AUTOMATICALLY GENERATING ROBUST SIGNATURES USING A MACHINE LEARNING APPROACH TO UNVEIL ENCRYPTED VOIP TRAFFIC WITHOUT USING PORT NUMBERS, IP ADDRESSES AND PAYLOAD INSPECTION

by

Riyad Akla Alshammari

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
May 2012

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled “AUTOMATICALLY GENERATING ROBUST SIGNATURES USING A MACHINE LEARNING APPROACH TO UNVEIL ENCRYPTED VOIP TRAFFIC WITHOUT USING PORT NUMBERS, IP ADDRESSES AND PAYLOAD INSPECTION” by Riyad Akla Alshammari in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated: May 14, 2012

External Examiner:	Dr. Mourad Debbabi
Research Supervisor:	Dr. A. Nur Zincir-Heywood
Examining Committee:	Dr. Malcolm Heywood
	Dr. Srinivas Sampalli
Departmental Representative:	Dr. Michael McAllister

DALHOUSIE UNIVERSITY

DATE: May 14, 2012

AUTHOR: Riyad Akla Alshammari

TITLE: AUTOMATICALLY GENERATING ROBUST SIGNATURES USING A MACHINE LEARNING APPROACH TO UNVEIL ENCRYPTED VOIP TRAFFIC WITHOUT USING PORT NUMBERS, IP ADDRESSES AND PAYLOAD INSPECTION

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: Ph.D. CONVOCATION: October YEAR: 2012

Signature of Author

Table of Contents

List of Tables . . . . **ix**

List of Figures . . . . **xv**

Abstract . . . . **xxviii**

List of Abbreviations Used . . . . **xxix**

Acknowledgements . . . . **xxxiii**

Chapter 1 Introduction **1**

1.1 Motivation and Objectives 2

1.2 Thesis Contributions 9

1.3 Thesis Organization 10

Chapter 2 Literature Survey and Background **12**

2.1 Overview of VoIP Applications 12
2.1.1 Skype 13
2.1.2 Gtalk 14
2.1.3 Other Common Applications 15

2.2 Traffic Classification: Specifically P2P and VoIP 16
2.2.1 Machine Learning Methods for Classifying P2P Network Traffic 18
2.2.2 Port-based Methods for Classifying P2P Network Traffic . . . 21
2.2.3 Deep Packet Inspection (DPI) Methods for Classifying P2P Network Traffic 24
2.2.4 Open Source Network Traffic Classification Tools 25
2.2.5 Alternative Methods for Classifying P2P Network Traffic . . . 26
2.2.6 Importance of Robustness/Generalization for the Classification of Network Traffic 29

Chapter 3 Methodology: Learning Algorithms and Data Sets Employed **31**

3.1 Feature Sets 31
3.1.1 Packet Header-based Feature Set 32
3.1.2 Flow-based Feature Set 34

3.2 Labels, Training and Testing Data Sets 36

3.3 Robustness of Machine Learning Algorithms 36

3.4 Machine Learning Algorithms Deployed 37
3.4.1 C5.0 37
3.4.2 AdaBoost 37
3.4.3 Team-based Genetic Programming 38

3.5 Evaluation of Machine Learning Algorithms 40

3.6 Traces Deployed 43
3.6.1 Dalhousie Traces 43
3.6.2 ITALY Traces 43
3.6.3 NIMSII Traces 44
3.6.4 NIMSIII Traces 48
3.6.5 IPv6 Traces 49
3.6.6 Analysis of Traces 52

Chapter 4 Subset Sampling Methods for Selecting Training Data Sets 69

4.1 Uniform Random *N* Sampling Method 69

4.2 Stratified Sampling Method 70
4.2.1 Self Organizing Feature Maps (SOM) 71
4.2.2 Training Data Sets 75

4.3 Continuous Data Stream 77

4.4 Results of Experiments for Subset Sampling 78

4.5 Results of Experiments for the Best Subset Sampling Technique – Uniform Random Sampling of 6 Percent 81

4.6 Discussion of Results 90

Chapter 5 Exploring the Robustness/Generalization of the Classifiers 91

5.1 Results of Packet Header Experiments for Skype Identification – Robustness of Signatures 91

5.2 Results of Flow Experiments for Skype Identification – Robsutness of Signatures 100

5.3 Discussion of Results 109

Chapter 6 Evasion of the Robust Signatures . . . 112

6.1 Altering Skype Traffic – Evasion Attacks . . . 112

6.2 Evaluation of Evasion Attacks . . . 117
6.2.1 Discussion of Results . . . 123

6.3 Evading Public Tools . . . 124
6.3.1 Evading Attacks Against OpenDPI . . . 124
6.3.2 Evading Attacks Against Wireshark . . . 126

6.4 Discussion . . . 132

Chapter 7 Analysis of the Classifier Solutions . . . 133

7.1 Analysis of the Packet Header-based Approach for Skype Detection . 133
7.1.1 CPU Training Time . . . 133
7.1.2 Complexity of Solutions . . . 135
7.1.3 Best Solution for Each Classifier for the Packet Header-based Approach for Skype Detection . . . 140
7.1.4 FPR for the Best Solutions for Each Classifier based on the Packet Header Feature Set for Skype Detection . . . 147

7.2 Analysis of the Flow-based Approach for Solutions Generated by all Classifiers . . . 151
7.2.1 CPU Training Time . . . 151
7.2.2 Complexity of Solutions . . . 153
7.2.3 Analysis of the Best Solution from Each Classifier . . . 158
7.2.4 FPR for each of the Best Solutions for Each Classifier . . . 165

7.3 Sensitivity Analysis of Configuration Parameters . . . 169

7.4 Limitations . . . 172

Chapter 8 Ensemble Learning for Skype Identification . . . 174

8.1 The Proposed Ensemble Learner Neural Network Model for Skype Identification . . . 175

8.2 Results of the Ensemble Learning Model . . . 177

8.3 Discussion of Results . . . 184

Chapter 9 Multi-Class Classification for VoIP Applications . . . 185

9.1 Results of the VoIP Multi-Class Classification . . . 186

9.2 Analysis of the Classifier Solutions . . . 197

9.2.1 CPU Training Time 197
9.2.2 Complexity of Solutions 199
9.2.3 Analysis of the Best Solution from Each Classifier for Multi-Class VoIP Applications 202

9.3 Discussion 208

Chapter 10 Conclusion 209

10.1 Research Objectives Realized 209
10.1.1 Identifying a Suitable Method for Sampling a Training Data Set to Generate Robust Signatures 209
10.1.2 Exploring the Performance of Classifying Encrypted Traffic Using Only One Packet – Without any Temporal Information . . 210
10.1.3 Exploring the Performance of Classifying Encrypted Traffic Using Only One Flow – With Temporal Information 210
10.1.4 Exploring the Limits of Employing Machine Learning Algorithms in Order to Classify Encrypted Traffic Robustly 211
10.1.5 Testing the Ability of the Robust Signatures Against Evasion Attacks 211
10.1.6 Analysis of the Selected Features 212
10.1.7 Exploring the Performance of Ensemble Learning for Enhancing the Performance of the Signatures 212
10.1.8 Exploring the Ability of the Proposed Approach to Classify more than One VoIP Application 213

10.2 Key Contributions 213

10.3 Future Research Directions 215

Bibliography 217

Appendix A Results of Subset Sampling Methods 227

A.1 Uniform Random *N* Sampling 227

A.2 Stratified Sampling 240

A.3 Continuous Sampling 252

Appendix B One-way ANOVA Statistical Analysis Tests 261

B.1 Choosing the Sampling Method 261

B.2 Subset Sampling Using the Uniform Random Method with a 6 Percent Sampling Rate 263

B.3 Robustness of the Classifiers 264
B.3.1 Robustness of the Three ML Algorithms Using the Packet Header-based Feature Set 264
B.3.2 Robustness of the Three ML Algorithms Using the Flow-based Feature Set 265
B.3.3 Multi-Classes of the Three ML Algorithms Using the Flow-based Feature Set 266

Appendix C FPR for the Best Solutions for Each Classifier 268

C.1 FPR for Packet Header Feature Set for Skype Detection 268

C.2 FPR for Flow Feature Set for Skype Detection 268

Appendix D Range for the Robust Signatures 273

D.1 C5.0 Signatures 273

List of Tables

2.1 Flow Feature Set for the Clustering Technique Used by Erman et al. in [12] . . . 20

2.2 Flow Feature Set for the Semi-Supervised Classifier Used by Erman et al. in [53] . . . 20

2.3 Most Important Distinguishable Discriminators Identified by Moore et al. in [17] . . . 22

2.4 Network Traffic Grouped by Moore et al. in [17] . . . 22

2.5 Nine Identification Methods Used by Moore et al. in [9] . . . 23

2.6 Network Traffic Grouped by Karagiannis et al. in [13] . . . 27

3.1 Packet Header-based Features Employed; * Denotes that the Feature is Normalized by log . . . 33

3.2 Flow-based Features Employed . . . 35

3.3 C5.0 Parameterization . . . 41

3.4 Weka Parameterization for AdaBoost . . . 41

3.5 SBB-GP Parameterization . . . 42

3.6 Overview (in Millions) of Network Traces Employed . . . 52

4.1 Number of Flows Sampled for the Uniform Random *N* Sampling Method . . . 70

4.2 Parameters of the SOM . . . 72

4.3 Number of Flows for each Application in the Univ2007 Trace . . . 75

4.4 Number of Flows Sampled for the Stratified Sampling Method with Fixed *N* Records . . . 76

4.5 Number of Flows Sampled for the Stratified Sampling Method with Fixed *N* Percentage . . . 76

4.6 Number of Flows Sampled for a Continuous Data Stream . . . 78

4.7 Results of the Best Models for Each Classifier on the Validation Data for Subset Sampling Methods . . . 80

4.8 Best Model Results of the Subset Sampling Method (6% Uniform Random Sampling) for Skype Detection Based on Flow Feature Sets (Training and Testing University Data Sets) 87

4.9 Best Model Results for the Subset Sampling Method (6% Uniform Random Sampling) for Skype Detection Based on Flow Feature Sets (ITALY Data Sets) 87

4.10 Best Model Results for the Subset Sampling Method (6% Uniform Random Sampling) for Skype Detection Based on Flow Feature Sets (NIMSII Data Sets) 88

4.11 Best Model Results for the Subset Sampling Method (6% Uniform Random Sampling) for Skype Detection Based on Flow Feature Sets (NIMSIII Data Sets) 88

4.12 Best Model Results for the Subset Sampling Method (6% Uniform Random Sampling) for Skype Detection Based on Flow Feature Sets (IPv6 Data Sets) 89

5.1 Best Model Results on the Training Data Set for the Packet Header-based Feature Set for Skype Detection (Training and Testing on Dalhousie Data Sets) 97

5.2 Best Model Results on Testing for the Packet Header-based Feature Set for Skype Detection (ITALY Data Sets) 97

5.3 Best Model Results on Testing Sets for the Packet Header-based Feature Set for Skype Detection (NIMSII Data Sets) 98

5.4 Best Model Results on Testing for the Packet Header-based Feature Set for Skype Detection (NIMSIII Data Sets) 98

5.5 Best Model Results on Testing for the Packet Header-based Feature Set for Skype Detection (IPv6 Data Sets) 99

5.6 Best Model Results on Training Data Set for the Flow-based Feature Set for Skype Detection (Training and Testing on Dalhousie Data Sets) 106

5.7 Best Model Results on Testing for the Flow-based Feature Set for Skype Detection (ITALY Data Sets) 106

5.8 Best Model Results on Testing for the Flow-based Feature Set for Skype Detection (NIMSII Data Sets) 107

5.9 Best Model Results on Testing for the Flow-based Feature Set for Skype Detection (NIMSIII Data Sets) 107

5.10 Best Model Results on Testing for the Flow-based Feature Set for Skype Detection (IPv6 Data Sets) 108

6.1 Statistical Overview of Altered-Skype Audio Files 113

6.2 TP Rate for the Three Classifiers on each of the Altered-Skype Traces 122

6.3 OpenDPI Results on the NIMS Traces and Altered-Skype Traces 125

6.4 TP Rate for OpenDPI on each of the Altered-Skype Traces . . 125

7.1 Features Used by GP for the in/out-classes for Skype Based on the Packet Header Feature Set 141

7.2 Features Used by C5.0 for the in/out-classes for Skype Based on the Packet Header Feature Set 142

7.3 Features Used by AdaBoost as a Whole for Skype Based on the Packet Header Feature Set 142

7.4 Ranges of Values for each of the Flow Features for the Signatures Generated by C5.0 173

8.1 Results of Ensemble Learning (EL) for the Flow-based Feature Set for Skype Detection 178

8.2 Best Model Results on the Training Data Set for the Flow-based Feature Set for Skype Detection 178

8.3 Results of Ensemble Learning (EL) for the Flow-based Feature Set for Skype Detection 179

8.4 Best Model Results for the Flow-based Feature Set for Skype Detection 179

8.5 Results of Ensemble Learning (EL) for the Flow-based Feature Set for Skype Detection 180

8.6 Best Model Results for the Flow-based Feature Set for Skype Detection 180

8.7 Results of Ensemble Learning (EL) for the Flow-based Feature set for Skype Detection 181

8.8 Best Model Results for the Flow-based Feature Set for Skype Detection 182

8.9 Results of Ensemble Learning (EL) for the Flow-based Feature Set for Skype Detection 182

8.10 Best Model Results for the Flow-based Feature Set for Skype Detection 183

9.1 Results for the C5.0 Classifier - Multi-Class – All Traces 194

9.2 Results for the GP Classifier - Multi-Class – All Traces 195

9.3 Results for the AdaBoost Classifier - Multi-Class – All Traces . 196

9.4 Flow Features Utilized by C5.0 for Multi-Classes 203

9.5 Flow Features Utilized by GP for Multi-Classes 204

9.6 Flow Features Utilized by AdaBoost for Multi-Classes 204

B.1 A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms for the Subset Sampling Techniques on the Training Data Set 261

B.2 A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms for the Subset Sampling Techniques on the Training Data Set 262

B.3 A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms for the Subset Sampling Techniques on the Validation Data Set 262

B.4 A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms for the Subset Sampling Techniques on the Validation Data Set 262

B.5 A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms on the Subset Sampling Training Data Set 263

B.6 A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms on the Subset Sampling Training Data Set 263

B.7 A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms Using the Packet Header-based Feature Set on Training Data Set 264

B.8 A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms Using the Packet Header-based Feature Set on Training Data Set 264

B.9 A One-way ANOVA Statistical Analysis Test for the Mean DR for the three ML Algorithms Using the Flow-based Feature Set on Training Data Set 265

B.10 A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data set 265

B.11 A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Skype Classification 266

B.12 A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Skype Classification 266

B.13 A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Gtalk Classification 266

B.14 A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Gtalk Classification 267

B.15 A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Primus Classification 267

B.16 A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Primus Classification 267

C.1 Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Packet Header Feature Set on the Univ2007 Test Trace (FPR=0.03%) 268

C.2 Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Packet Header Feature Set on the Univ2010 Test Trace (FPR=2.4%) 269

C.3 Applications Wrongly Classified as Skype by the GP-based Signatures on the Packet Header Feature Set on the Univ2007 Test Trace (FPR=2.5%) 269

C.4 Applications Wrongly Classified as Skype by the GP-based Signatures on the Packet Header Feature Set on the Univ2010 Test Trace (FPR=65.1%) . . . 269

C.5 Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Packet Header Feature Set on the Univ2007 Test Trace (FPR=0.11%) . . . 270

C.6 Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Packet Header Feature set on the Univ2010 Test Trace (FPR=0.34%) . . . 270

C.7 Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=0.7%) . . . 270

C.8 Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=6.1%) . . . 271

C.9 Applications Wrongly Classified as Skype by the GP-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=9.9%) . . . 271

C.10 Applications Wrongly Classified as Skype by the GP-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=9.3%) . . . 271

C.11 Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=4.3%) . . . 272

C.12 Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=7.9%) . . . 272

List of Figures

2.1 Different Sub-methods Used by Moore et al. in [9] 23

3.1 Network Setup with Restrictions 45

3.2 Network Setup without Restrictions 45

3.3 Network Setup for Zfone Calls 46

3.4 Network Setup for Primus Calls 47

3.5 VPN Tunnel Setup 48

3.6 Packet Size Distribution for IPv6 Traces [101] 51

3.7 Number of Packets per Minute for TCP/UDP Protocols in Univ2007 Trace 54

3.8 Number of Bytes per Minute for TCP/UDP Protocols in Univ2007 Trace 55

3.9 Number of Packets per Minute for TCP/UDP Protocols in Univ2010 Trace 56

3.10 Number of Bytes per Minute for TCP/UDP Protocols in Univ2010 Trace 57

3.11 Number of Packets per Minute for TCP/UDP Protocols in ITALY Trace 58

3.12 Number of Bytes per Minute for TCP/UDP Protocols in ITALY Trace 59

3.13 Number of Packets per Minute for TCP/UDP Protocols in NIMSII Trace 60

3.14 Number of Bytes per Minute for TCP/UDP Protocols in NIMSII Trace 61

3.15 Number of Packets per Minute for TCP/UDP Protocols in NIMSIII Trace 62

3.16 Number of Bytes per Minute for TCP/UDP Protocols in NIMSIII Trace 63

3.17 Forward Direction of Mean of Inter-rrival Time vs. Std. of Inter-arrival for University, NIMS and ITALY Traces 65

3.18 Backward Direction of Mean of Inter-arrival Time vs. Std. of Inter-arrival for University, NIMS and ITALY Traces 66

3.19 Forward Direction of Mean of Packet Length vs. Std. of Packet Length for University, NIMS and ITALY Traces 67

3.20 Backward Direction of Mean of Packet Length vs. Std. of Packet Length for University, NIMS and ITALY Traces 68

4.1 Unified Distance Matrix 73

4.2 Distribution of Applications in each Neuron 74

4.3 Number of Packets for TCP/UDP Protocols in the Univ2007 Trace 77

4.4 DR and FPR Results for Skype Identification on the Training Data Sets Using the Uniform Random with 6 the Percent Sampling Method 81

4.5 Scatter Plot for the C5.0 Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR) 83

4.6 Scatter Plot for the AdaBoost Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR) 84

4.7 Scatter Plot for the GP Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR) 85

5.1 DR and FPR Results for Skype Packet Header Traffic on the Training Data Set 92

5.2 Scatter Plot for the C5.0 Classifier for Training Performance Using the Packet Header-based Feature Set for Skype Detection (DR versus FPR) 93

5.3 Scatter Plot for the AdaBoost Classifier for Training Performance Using the Packet Header-based Feature Set for Skype Detection (DR versus FPR) 94

5.4 Scatter Plot for the GP Classifier for Training Performance Using the Packet Header-based Feature Set for Skype Detection (DR versus FPR) 95

5.5 DR and FPR Results for Skype Flow Traffic on the Training Data Set 101

5.6 Scatter Plot for the C5.0 Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR) 102

5.7 Scatter Plot for the AdaBoost Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR) 103

5.8 Scatter Plot for the GP Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR) 104

5.9 Change in the DR for the Three Classifiers Using a Flow-based Feature Set for Skype Detection on the Test Traces Containing Skype Traffic 110

5.10 Change in the FPR for the Three Classifiers Using a Flow-based Feature Set for Skype Detection on Test Traces Containing Skype Traffic 111

6.1 Call Setup for Skype Evasion Attacks 113

6.2 Spectrum Analyzer Displays of Audio Signals 114

6.3 Forward Direction of Mean of Packet Length vs. Std. of Packet Length for NIMS Traces and Altered-Skype Traces 115

6.4 Backward Direction of Mean of Packet Length vs. Std. of Packet Length for NIMS Traces and Altered-Skype Traces . . 116

6.5 Performance of the C5.0-based Signatures Under Evasion Attacks by Using a Circos Graph 119

6.6 Performance of the AdaBoost-based Signatures Under Evasion Attacks by Using a Circos Graph 120

6.7 Performance of the GP-based Signatures Under Evasion Attacks by Using a Circos Graph 121

6.8 OpenDPI Testbed Setup 124

6.9 Wireshark Classifying SSH Packets Before Modifying Ports . . 127

6.10 Wireshark Misclassify SSH Packets After Modifying Ports . . 128

6.11 Wireshark Classified SSH Packets as HTTP Packets After Modifying Ports 129

6.12 Wireshark Could not Classify Skype Original Packets 130

6.13 Wireshark Could not Classify Altered-Skype Packets 131

7.1 Training Time (in Seconds) for Skype Detection Based on the Packet Header Feature Set 134

7.2 Number of Features Utilized for Each Classifier for Skype Classification Based on the Packet Header Feature Set 136

7.3 Number of Features Utilized for Each Classifier for non-Skype Classification Based on the Packet Header Feature Set 137

7.4 Number of Rules/Individuals Utilized for Each Classifier for Skype Classification Based on the Packet Header Feature Set 138

7.5 Number of Rules/Individuals Utilized for Each Classifier for non-Skype Classification Based on the Packet Header Feature Set 139

7.6 An Example of an AdaBoost Solution for Skype Based on the Packet Header Feature Set 144

7.7 An Example of a C5.0 Solution for Skype Based on the Packet Header Feature Set 145

7.8 An Example of a GP Solution for Skype Based on the Packet Header Feature Set 146

7.9 Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Packet Header Feature Set on the Univ2007 Test Trace (FPR=0.03%). 147

7.10 Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Packet Header Feature Set on the Univ2010 Test Trace (FPR=2.4%). 148

7.11 Applications Wrongly Classified as Skype by the GP-based Signatures on the Packet Header Feature Set on the Univ2007 Test Trace (FPR=2.5%). 148

7.12 Applications Wrongly Classified as Skype by the GP-based Signatures on the Packet Header Feature Set on the Univ2010 Test Trace (FPR=65.1%). 149

7.13 Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Packet Header Feature Set on the Univ2007 Test Test Trace (FPR=0.11%). 149

7.14 Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Packet Header Feature set on the Univ2010 Test Trace (FPR=0.34%). 150

7.15 Training Time (in Seconds) for Skype Detection Based on the Flow Feature Set 152

7.16 Number of Features Utilized for Each Classifier for Skype Classification Based on the Flow Feature Set 154

7.17 Number of Features Utilized for Each Classifier for non-Skype Classification Based on the Flow Feature Set 155

7.18 Number of Rules/Individuals Utilized for Each Classifier for Skype Classification Based on the Flow Feature Set 156

7.19 Number of Rules/Individuals Utilized for Each Classifier for non-Skype Classification Based on the Flow Feature Set . . . 157

7.20 Selected Flow Features of the GP-based Classifier for Skype Detection 159

7.21 Selected Flow Features of the AdaBoost-based Classifier for Skype Detection 159

7.22 Selected Flow Features for the C5.0-based Classifier for Skype Detection 160

7.23 An Example of an AdaBoost Solution for Skype Detection Based on the Flow Feature Set 161

7.24 An Example of a C5.0 Solution for Skype Detection Based on the Flow Feature Set 162

7.25 An Example of a GP Solution for Skype Detection Based on the Flow Feature Set 163

7.26 Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=0.7%). 165

7.27 Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=6.1%). 166

7.28 Applications Wrongly Classified as Skype by the GP-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=9.9%). 166

7.29 Applications Wrongly Classified as Skype by the GP-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=9.3%). 167

7.30 Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=4.3%). 167

7.31 Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=7.9%). 168

7.32 Sensitivity of C5.0 DR for Changing the Confidence Factor Parameter 169

7.33 Sensitivity of C5.0 FPR for Changing the Confidence Factor Parameter 170

7.34 Sensitivity of the Number of C5.0 Rules for Changing the Confidence Factor Parameter 170

8.1 Common Ensemble Learning Model 174

8.2 Representing the Flow Data Set to be Used by the Ensemble Learning Model 176

8.3 Second Step of the Ensemble Learning Model 176

9.1 DR and FPR Results for C5.0 or Multi-Class VoIP for the Flow-based Feature Set on the Training Data Sets 186

9.2 DR and FPR Results for GP or Multi-Class VoIP for the Flow-based Feature Set on the Training Data Sets 187

9.3 DR and FPR Results for AdaBoost or Multi-Class VoIP VoIP for the Flow-based Feature Set on the Training Data Sets . . 187

9.4 Scatter for the C5.0-based Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR) 188

9.5 Scatter for the C5.0-based Classifier for Training Performance using the Flow-based Feature set for Gtalk Detection (DR versus FPR) 189

9.6 Scatter for the C5.0-based Classifier for Training Performance Using the Flow-based Feature Set for Primus Detection (DR versus FPR) 189

9.7 Scatter for the GP-based Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR) 190

9.8 Scatter for the GP-based Classifier for Training Performance Using the Flow-based Feature Set for Gtalk Detection (DR versus FPR) 190

9.9 Scatter for the GP-based Classifier for Training Performance Using the Flow-based Feature Set for Primus Detection (DR versus FPR) 191

9.10 Scatter for the AdaBoost-based Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR) 191

9.11 Scatter for the AdaBoost-based Classifier for Training Performance Using the Flow-based Feature Set for Gtalk Detection (DR versus FPR) 192

9.12 Scatter for the AdaBoost-based Classifier for Training Performance Using the Flow-based Feature Set for Primus Detection (DR versus FPR) 192

9.13 Training Time (in Seconds) for VoIP Multi-Class Classification Based on the Flow Feature Set 198

9.14 Number of Features Utilized for Each Classifier for VoIP Multi-Class Classification Based on the Feature Set 200

9.15 Number of Rules Utilized for Each Classifier for VoIP Multi-Class Classification Based on the Flow Feature Set 201

9.16 An Example of an AdaBoost Solution for Multi-Class VoIP Application Classification Based on the Flow Feature Set 205

9.17 An Example of a C5.0 Solution for Multi-Class VoIP Application Classification Based on the Flow Feature Set 206

9.18 An Example of a GP Solution for Multi-Class VoIP Application Classification Based on the Flow Feature Set 207

A.1 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 30K Method on Training Data . . 227

A.2 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 30K Method on Validation Data . 228

A.3 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 60K Method on Training Data . . 228

A.4 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 60K Method on Validation Data . 229

A.5 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 100K Method on Training Data . 229

A.6 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 100K Method on Validation Data 230

A.7 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 200K Method on Training Data . 230

A.8 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 200K Method on Validation Data 231

A.9 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 400K Method on Training Data . 231

A.10 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 400K Method on Validation Data 232

A.11 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 800K Method on Training Data . 232

A.12 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 800K Method on Validation Data 233

A.13 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 1% Method on Training Data . . . 233

A.14 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 1% Method on Validation Data . . 234

A.15 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 2% Method on Training Data . . . 234

A.16 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 2% Method on Validation Data . . 235

A.17 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 3% Method on Training Data . . . 236

A.18 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 3% Method on Validation Data . . 236

A.19 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 4% Method on Training Data . . . 237

A.20 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 4% Method on Validation Data . . 237

A.21 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 5% Method on Training Data . . . 238

A.22 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 5% Method on Validation Data . . 238

A.23 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 6% Method on Training Data . . . 239

A.24 Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 6% Method on Validation Data . . 239

A.25 Results of Three Machine Learning Algorithms Using the Stratified Sampling 30K Method on Training Data 240

A.26 Results of Three Machine Learning Algorithms Using the Stratified Sampling 30K Method on Validation Data 240

A.27 Results of Three Machine Learning Algorithms Using the Stratified Sampling 60K Method on Training Data 241

A.28 Results of Three Machine Learning Algorithms Using the Stratified Sampling 60K Method on Validation Data 241

A.29 Results of Three Machine Learning Algorithms Using the Stratified Sampling 100K Method on Training Data 242

A.30 Results of Three Machine Learning Algorithms Using the Stratified Sampling 100K Method on Validation Data 242

A.31 Results of Three Machine Learning Algorithms Using the Stratified Sampling 200K Method on Training Data 243

A.32 Results of Three Machine Learning Algorithms Using the Stratified Sampling 200K Method on Validation Data 243

A.33 Results of Three Machine Learning Algorithms Using the Stratified Sampling 400K Method on Training Data 244

A.34 Results of Three Machine Learning Algorithms Using the Stratified Sampling 400K Method on Validation Data 244

A.35 Results of Three Machine Learning Algorithms Using the Stratified Sampling 800K Method on Training Data 245

A.36 Results of Three Machine Learning Algorithms Using the Stratified Sampling 800K Method on Validation Data 245

A.37 Results of Three Machine Learning Algorithms Using the Stratified Sampling 1% Method on Training Data 246

A.38 Results of Three Machine Learning Algorithms Using the Stratified Sampling 1% Method on Validation Data 246

A.39 Results of Three Machine Learning Algorithms Using the Stratified Sampling 2% Method on Training Data 247

A.40 Results of Three Machine Learning Algorithms Using the Stratified Sampling 2% Method on Validation Data 247

A.41 Results of Three Machine Learning Algorithms Using the Stratified Sampling 3% Method on Training Data 248

A.42 Results of Three Machine Learning Algorithms Using the Stratified Sampling 3% Method on Validation Data 248

A.43 Results of Three Machine Learning Algorithms Using the Stratified Sampling 4% Method on Training Data 249

A.44 Results of Three Machine Learning Algorithms Using the Stratified Sampling 4% Method on Validation Data 249

A.45 Results of Three Machine Learning Algorithms Using the Stratified Sampling 5% Method on Training Data 250

A.46 Results of Three Machine Learning Algorithms Using the Stratified Sampling 5% Method on Validation Data 250

A.47 Results of Three Machine Learning Algorithms Using the Stratified Sampling 6% Method on Training Data 251

A.48 Results of Three Machine Learning Algorithms Using the Stratified Sampling 6% Method on Validation Data 251

A.49 Results of Three Machine Learning Algorithms Using the Continuous 30K Method on Training Data 252

A.50 Results of Three Machine Learning Algorithms Using the Continuous 30K Method on Validation Data 252

A.51 Results of Three Machine Learning Algorithms Using the Continuous 60K Method on Training Data 253

A.52 Results of Three Machine Learning Algorithms Using the Continuous 60K Method on Validation Data 253

A.53 Results of Three Machine Learning Algorithms Using the Continuous 100K Method on Training Data 254

A.54 Results of Three Machine Learning Algorithms Using the Continuous 100K Method on Validation Data 254

A.55 Results of Three Machine Learning Algorithms Using the Continuous 200K Method on Training Data 255

A.56 Results of Three Machine Learning Algorithms Using the Continuous 200K Method on Validation Data 255

A.57 Results of Three Machine Learning Algorithms Using the Continuous 400K Method on Training Data 256

A.58 Results of Three Machine Learning Algorithms Using the Continuous 400K Method on Validation Data 256

A.59 Results of Three Machine Learning Algorithms Using the Continuous 800K Method on Training Data 257

A.60 Results of Three Machine Learning Algorithms Using the Continuous 800K Method on Validation Data 257

A.61 Results of Three Machine Learning Algorithms Using the Continuous 30 Minute Method on Training Data 258

A.62 Results of Three Machine Learning Algorithms Using the Continuous 30 Minute method on Validation Data 258

A.63 Results of Three Machine learning Algorithms Using the Continuous 60 Minutes Method on Training Data 259

A.64 Results of Three Machine Learning Algorithms Using the Continuous 60 Minute Method on Validation Data 259

A.65 Results of Three Machine Learning Algorithms Using the Continuous 90 Minutes Method on Training Data 260

A.66 Results of Three Machine Learning Algorithms Using the Continuous 90 Minute Method on Validation Data 260

D.1 Range for the min_fpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 273

D.2 Range for the mean_fpktl flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 274

D.3 Range for the max_fpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 275

D.4 Range for the std_fpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 276

D.5 Range for the min_bpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 277

D.6 Range for the mean_bpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 278

D.7 Range for the max_bpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 279

D.8 Range for the std_bpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 280

D.9 Range for the min_fiat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 281

D.10 Range for the mean_fiat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 282

D.11 Range for the max_fiat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 283

D.12 Range for the std_fiat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 284

D.13 Range for the min_biat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 285

D.14 Range for the mean_biat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 286

D.15 Range for the max_biat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 287

D.16 Range for the std_biat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 288

D.17 Range for the Duration Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 289

D.18 Range for the Total fpackets Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 290

D.19 Range for the Total fbytes Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 291

D.20 Range for the Total bpackets Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 292

D.21 Range for the Total bbytes Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set 293

Abstract

The identification of encrypted network traffic represents an important issue for network management tasks including quality of service, firewall enforcement and security. Traffic identification becomes more and more challenging as the traditional techniques such as port numbers or deep packet inspection are becoming ineffective against applications such as the Peer-to-Peer (P2P) Voice over Internet Protocol (VoIP), which uses non-standard ports and encryption. Thus, different approaches such as machine learning (ML) are explored in the literature for traffic classification. However, traffic classification represents a particularly challenging application domain for ML. Ideally, solutions should be both simple (hence efficient to deploy) and accurate. Recent advances in ML provide the opportunity to decompose the original problem into a subset of classifiers with non-overlapping behaviours, in effect providing further insight into the problem domain and increasing the throughput of solutions. Thus, this thesis presents a novel approach for generating robust signatures to classify P2P VoIP traffic using a ML-based approach, specifically with the C5.0, GP and AdaBoost classification algorithms. In this research, simple packet header feature sets and statistical flow feature sets are explored without using the IP addresses, source/destination ports and payload information to unveil the encrypted VoIP application in network traffic. In this context, what is meant by robust signatures are those which have been learned by training on one network are still valid when they are applied to traffic coming from different time periods, different networks (locations) as well as under evasion attacks that are designed to bypass such a classifier.

Results show that the performance of the automatically generated signatures does not degrade significantly when evaluated against the robustness criteria. These results demonstrate that flow-based statistical features (temporal information) with the use of a ML-based approach can achieve high classification accuracy and produce robust signatures. Furthermore, the results on the evasion experiments demonstrate that the performance of the signatures is very promising if a malicious user tries to alter the characteristics of VoIP (specifically, Skype) traffic to evade the classifier.

List of Abbreviations Used

AES Advanced Encryption Standard

AH Authentication Header

AIF Audio Interchange File Format

AIFF Audio Interchange File Format

AIM AOL Instant Messenger

ANOVA Analysis of Variance

AU Audio File format from Sun Microsystems

BLINC BLINd Classification

BMU Best Matching Unit

CAIDA Cooperative Association for Internet Data Analysis

CF Confidence Factor

CPU Central Process Unit

CVS Concurrent Versions System

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DNS Domain Name System

DPI Deep Packet Inspection

DR Detection Rate

E2E End-to-End

E2O End-to-Out

EL Ensemble Learning

ESP Encapsulated Security Payload

FCBF Fast Correlation-Based Filter

FN False Negative

FP False Positive

FPR False Positive Rate

FTP File Transfer Protocol

GA Genetic Algorithm

GB Giga Byte

GP	Genetic Programming
GPRS	General Packet Radio Service
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IDENT	Identification Protocol
IETF	Internet Engineering Task Force
IM	Instant Messaging
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IRC	Internet Relay Chat
ISP	Internet Service Provider
ITS	Information Technology Services Centre
KISS	Chi-Square Signatures (pronounced as KISS)
L2TP/IPSec	Layer 2 Tunneling Protocol Internet Protocol Security
libpcap	Packet Capture library
MAWI	Measurement and Analysis on the WIDE Internet
ML	Machine Learning
MP3	MPEG-2 Audio Layer III
MSN	Microsoft Messenger
NAT	Network Address Translation
Netbios	File Sharing and Name Resolution protocol
NIMS	Network Information Management and Security Lab
NM	Network Management
NN	Neural Network
NTP	Network Time Protocol
P2P	Peer-to-Peer
PDF	Probability Density Function

POP	Post Office Protocol
PSTN	Public Switched Telephone Network
RAM	Random Access Memory
Rlogin	Remote login protocol
RSA	Rivest Shamir Adleman
RTP	Real-time Transport Protocol
RTT	Round Trip Time
SBB	Symbiotic Bid-Based
SCP	Secure Copy Protocol
SIG	Signaling
SIP	Session Initiation Protocol
SGSN	Serving Gateway Support Node
SLTC	Self-Learning Traffic Classifier
SMB	Server Message Block
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
softTBB	soft Talk Broad Band
SOM	Self-Organizing Feature Maps
SSH	Secure Shell
SSL	Secure Socket Layer
SSL/TLS	Secure Socket Layer Transport Layer Security
STUN	Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)
SVM	Support Vector Machine
T1	1.544 megabits per second line rate
TCM	Time Correlation Metric
TCP	Transmission Control Protocol
TDG	Traffic Dispersion Graph
Telnet	a remote terminal access protocol
TN	True Negative
TP	True Positive

Tstat TCP Statistic and Analysis Tool

TV Television

UDP User Datagram Protocol

U-matrix Unified distance matrix

UMTS Universal Mobile Telecommunication System

VNC Virtual Network Computing

VoIP Voice over Internet Protocol

VPN Virtual Private Network

WiFi Wireless Fidelity

XMPP Extensible Messaging and Presence Protocol

YMSG Yahoo! Messenger

ZRTP Zimmermann Real-time Transport Protocol

Acknowledgements

I would like to thank my supervisor Dr. Nur Zincir-Heywood for her support and guidance throughout my graduate career. I am very fortunate to have her as my supervisor. She facilitated the opportunities to publish and present my research at well-known national and international venues. As well, I would like to express my gratitude and thanks to my committee members for their insightful questions and their invaluable feedback.

This thesis is dedicated to my family. I would like to thank my parents, brothers and sisters for their enduring encouragement, patience and unconditional support during my graduate study and I am not sure how I could repay them for it. My most heartfelt thanks go to my wife Hanoof for her love, strength and patience during the Ph.D. journey. She has been a strong pillar of support and inspiration. Last, but not least, I wish to thank all of my sponsors.

Chapter 1

Introduction

Traffic classification becomes a crucial requirement for network administrators to manage their network and help them to allocate expensive network bandwidth and resources to essential applications. Hence, network administrators are in need of efficient tools to manage, control and measure the traffic. However, managing network traffic requires huge resources to verify the traffic on the network, to ensure that organizational policies are met and to ensure security for the users. Identifying Internet Protocol (IP) Network traffic according to the application type has the capability of resolving some of the complicated network management problems for organizations (enterprises) such as managing bandwidth budget and ensuring quality of service objectives for critical applications.

Furthermore, traffic classification is important for defence applications since it can facilitate the assessment of security threats. Such a system is particularly useful from a law enforcement application perspective since most of the time users with malicious intentions try to hide their behaviour either in encrypted or covert tunnels. For example, should some law enforcement agencies need to intercept or capture malicious traffic (e.g. child pornography traffic) from an Internet Service Provider (ISP), then IP network traffic classification is a core part of the solution. Hypothetically, from a chief security officer's point of view any application that hides its behaviour and avoids detection by encrypting its payload and implementing many methods to bypass firewalls or proxies is a risky application for sensitive information. Thus, such an application can be viewed as a backdoor [1].

Thus, systems which can classify encrypted traffic represent a first step in identifying such malicious behaviours. Moreover such systems can be useful as a forensic tool to identify applications used by malicious users whose data is collected/captured by law-enforcement units. In this case, establishing the classification of traffic types can reflect the current utilization of applications and services in a given traffic trace.

In turn, this can help law-enforcement units to make a case for investigating the true intent of malicious users. Moreover, network engineering problems such as traffic shaping or workload modeling require the classification of network traffic. In some cases law enforcement agencies may want to 'tap' some actual phone call in so far as who called and when the call happened and will ask telephony companies to do so. In cases in which an individual is using a Voice over Internet Protocol (VoIP) such as Skype to communicate, then they request the information from the ISP. Therefore, traffic classification offers the option of identifying VoIP traffic in the ISP network. Moreover, based on the traffic classification approach employed, user privacy can be maintained.

1.1 Motivation and Objectives

The increasingly popular P2P VoIP applications have enjoyed huge success in the last few years. They are becoming a major communication service for enterprises and individuals since the cost of VoIP calls is much cheaper than the traditional Public Switched Telephone Networks (PSTNs). Voice and video quality are getting better and the communication is free of charge if placed directly from one VoIP end user to another. Moreover, through such services, the dynamic approach to circumvent restrictive network environments such as firewalls and Network Address Translation (NAT) boxes is possible. To date, there are many VoIP products which are able to provide high call quality such as Skype [2], Gtalk [3], Microsoft Messenger (MSN) [4], Primus [5] and Yahoo! Messenger (YMSG) [6]. Skype is a very popular P2P VoIP client developed in 2002 by the creators of Kazaa, which allows its users to communicate through voice calls, audio conferencing and text messages. Skype protocols are proprietary and an extensive use of cryptography is implemented by the Skype creators. Moreover, Skype employs a number of methods to circumvent NAT and firewall restrictions [7], which makes it difficult to distinguish from non-Skype traffic. On the other hand, Gtalk is an instant messenger developed by Google, which allows its users to place voice calls, send text messages, check emails and transfer files. Gtalk provides services very similar to those of MSN, YMSG, Primus and Skype since it has abilities for voice calls, instant messaging and buddy lists. In practice, it resembles Skype application since Gtalk encrypts its traffic; however the fundamental protocols and techniques employed are relatively distinctive. Thus, the goal of this thesis is to develop a model which distinguishes VoIP (mainly Skype, and Gtalk) traffic from non-VoIP traffic. An efficient classification of such VoIP traffic represents a fundamental issue for network management tasks such as managing bandwidth budgets and ensuring quality of service objectives. Naturally, the process of traffic classification has several unique challenges including: the non-standard utilization of ports, the embedding of services within encrypted channels, the dynamic port-to-application relationships, and the real-time nature of the domain.

Traditionally, two approaches are used to identify IP network traffic: the first approach is to use 'well-known' Transmission Control Protocol (TCP) and/or User Datagram Protocol (UDP) port numbers (visible in TCP or UDP headers) while the second approach includes more sophisticated techniques based on 'deep packet inspection' (DPI) within TCP or UDP payloads (visible payloads) looking for specific protocol signatures. Each approach relies on some assumptions, which are no longer accurate and has many disadvantages. The first approach assumes most applications always use well-known port numbers registered by the internet assigned numbers authority (IANA) [8]. However, this assumption becomes increasingly inaccurate when applications use non-standard ports to bypass firewalls or circumvent operating system restrictions. New applications such as Skype have not registered port numbers with IANA and assign port numbers dynamically. Moreover, the same port number can be used to transmit multiple applications, most notably port 80. Moore and Papagiannaki showed that classification based on the IANA port list is correct 70% of the time [9]. Madhukar and Williamson confirmed that port number analysis misclassifies 30-70% of their flow traffic [10].

On the other hand, the second approach, DPI, assumes the access to the payload of every packet. This technique can be extremely accurate when the payload is not encrypted. Sen et al. [11] demonstrates that classifying Peer-to-Peer (P2P) traffic based on payload signatures could reduce false positive and false negative rates by 5%. Moreover, Moore and Papagiannaki [9] showed that using the entire payload can classify 100% of packets correctly. However, DPI has many limitations. Firstly, governments may regulate the use of payload and enforce constraints on its use since it can violate some organizational privacy policies or go against related privacy legislation. Secondly, examining the payload of a packet at the network speed is a computationally expensive task since the speed of networks, i.e. packet volumes transmitted through a network, is increasing daily. Hence, deploying DPI, which works efficiently is challenging. Finally, the success of DPI is losing ground since new applications such as Skype or other VoIP or P2P traffic which use techniques such as protocol encapsulation, payload encryption, and protocol obfuscation, imply that the payload is opaque. Thus, other techniques are required to increase the accuracy of network traffic classification.

As discussed earlier, since the traditional methods are ineffective for the new emerging P2P VoIP applications, many studies in the literature employ machine learning (ML) techniques using statistical flow information (features). Such features are usually derived from the information on the transport layer, which does not depend on port numbers or payload inspections. Moreover they have shown promising results for classifying encrypted applications [12, 13, 14, 15, 16]. However, when ML is deployed to extract signatures from traffic data it requires training data (where the traffic is labelled by application, i.e. ground truth) and a feature set such as inter-arrival times or packet sizes to represent the traffic. Therefore, ML classifiers are trained on the data set to correlate sets of feature values with the class label to extract signatures (rules) to classify unknown traffic.

Recent research in this area focuses on the identification of efficient and effective classifiers. Different research groups have employed expert systems or various machine learning techniques such as Hidden Markov models, Naïve Bayesian models, AdaBoost, RIPPER, Decision Trees or Maximum Entropy methods to investigate this problem [9, 17, 18, 19, 20, 21, 22, 23, 24, 25]. Moreover, the limitations of port and payload based analysis have motivated the use of transport/flow layer statistics for traffic classification [12, 13, 15]. These techniques rely on the observation that different applications have distinct behaviour patterns on the network. However, in general all these efforts show that even though it is easier to apply such techniques to well-known public domain applications such as mail, more work is necessary to distinguish between Peer-to-Peer (P2P) or encrypted applications accurately. Moreover, P2P and encrypted applications such as Skype or Gtalk use port numbers dynamically as well as using the same port number for multiple applications.

There are three commonly used machine learning techniques. These are: supervised learning, unsupervised learning, and semi-supervised learning. Supervised learning is finding the correlation between the target class (labels) and the input feature to build a set of rules/models. Supervised learning requires labelled training data. While unsupervised learning is the clustering (grouping) of records which have similar characteristics according to the input features regardless of the classes (labels). Thus, unsupervised learning does not require labelled data. On the other hand, semi-supervised learning aims to understand how combining labelled and unlabelled data may change the learning behaviour, and how to design algorithms, which take advantage of such a combination [26]. In this research, the focus is on the application of supervised learning techniques to encrypted VoIP traffic classification, specifically classification of Skype and Gtalk traffic. The reason such an approach is taken is twofold: (i) automatically generating signatures (rules) is necessary to classify VoIP encrypted traffic, and (ii) automating the process of selecting the most appropriate features /attributes for those signatures becomes possible. In this thesis, I have employed three supervised machine learning algorithms: C5.0, AdaBoost and GP. The reasons I have employed these machine learning algorithms include the following: previous works have reported good performance from these learning models in their respective studies [21, 22, 25]. In the form of rules I have observed these models to give the best solutions under different network conditions [18, 19, 20, 27]. Moreover, all of these learning models are capable of choosing the best attributes from a given set. This is an important property, given that I am interested in analyzing which attributes are the best from a set of all possible attributes. Last, but not least, all three of these learning algorithms can generate solutions automatically in the form of rules which are easy to understand by human experts. I refer to these rules as the automatically generated signatures to identify the target traffic class (e.g. Skype) in a given traffic log file. This is a very important property in order to employ the generated rules as signatures to classify traffic in practice. Furthermore, these learning models (C5.0, GP and AdaBoost) provide human readable solutions, hence, the solutions they generate are not a black box to the system administrators or network engineers. Additionally, other ML methods (black box methods) such as Support Vector Machines (SVM) and Bayesian methods have significant memory overheads, which make all but the most advanced instances inappropriate whereas the advanced cases are not openly available. Particularly, Bayesian methods require a lot of expertise to extract their potential. Conversely C5.0 addresses the memory overheads of C4.5 [28] making for a very robust implementation. Likewise, AdaBoost and GP manage the memory very well.

The use of ML techniques requires two major steps. Firstly, features need to be defined to describe the traffic data to the ML algorithms. In this case, features can be based on packet header information or can be calculated over flows representing multiple packets. Secondly, the ML technique needs to be trained to find patterns to correlate features to known traffic classes (supervised learning) and create models/rules to classify traffic. Every ML technique has a different schema to associate with a specific set of features for building the final model/solution.

The number of network packets passing through high-speed links is massive and is affected by the applications used, the number of Internet users and the capacity of the links. As a result, sampling network traffic for traffic classification becomes a vital procedure for dealing with huge volumes of traffic where resources are limited (e.g. hard disk, memory). The most difficult part of sampling is to capture the behaviour of an application by observing a small number of packets/flows. Therefore this thesis investigates the challenging problem of sampling training data sets for the ML algorithms. Weiss [29] has already demonstrated that performance of the classifier is not impacted by restricting the learner to a subset of the exemplars during training. To study the effectiveness of ML algorithms to generate transportable/robust signatures, I use totally different data sets for training and testing the classifiers. Naturally, these traffic traces represent large data sets from a ML perspective. Thus, subset sampling is used to decouple the overall exemplar count from the subset over which training is conducted. I performed subset sampling to limit the memory and Central Process Unit (CPU) time required for training.

As discussed earlier, one of the objectives of this thesis is to develop a model, which distinguishes VoIP from non-VoIP traffic robustly without using IP addresses, TCP/UDP port numbers or payload information. I believe that this will enable the learning model to generalize from one network to another as well as potentially enabling such an approach to be employed for the classification of other encrypted applications. To achieve this goal, the approach consists of two main phases: the learning phase to identify signatures automatically and the classification phase. The goal of the learning phase is to correlate between network traffic flow/packet header-based features and the target class using the training data set. At the end of this phase, the best learner is selected based on the highest performance in terms of Detection Rate (DR) and False Positive Rate (FPR). Then, the classification phase is employed to test the robustness/generalization of the rules/signatures found in phase one on unseen data sets, which include different locations, different networks and different time periods. Furthermore, another objective of finding robust signatures is to identify which feature set (Packet header or Flow) is the most suitable for generating robust signatures to classify encrypted VoIP traffic. Since one of the thesis objectives is to find robust signatures in order to classify VoIP traffic, the robustness/generalization of the proposed approach is investigated by evaluating it against potential evasion attacks, i.e. attacks, which try to bypass the proposed traffic classifier. To the best of my knowledge, this is the first time in which the robustness/generalization of classifiers was evaluated under evasion attacks for traffic classification, specifically for Skype P2P VoIP.

Moreover, my proposed approach for identifying VoIP encrypted traffic such as Skype is data-driven and I present all possible attributes/features to the learning algorithms employed. In doing so, I aim to examine: (i) which features will be considered the most appropriate by each learning algorithm; and (ii) which features will be chosen by all of the learning models employed in this work. Therefore, I believe that this subset of features will give the most robust/generalized as well as the most appropriate ones, which can be used on real-life network traffic traces. Furthermore, in this thesis, my approach to the identification of encrypted VoIP traffic takes the form of a forensic analysis tool.

Finally, the three machine learning algorithms used in this thesis build different solutions from the training data set. Usually, solutions built by these algorithms cover parts of the example space which is represented by the training data set. As discussed earlier, having an informative training data set is not an easy task. On the other hand, using only one ML algorithm might not result in the best rule set either. For example, a rule/model classifies a hard example or outliers should be included in the rule sets beside the rule/model that classifies a large number of exemplars. Therefore, I investigated how to best combine the solutions provided by AdaBoost, C5.0 and GP to produce one classifier. This is called ensemble learning. The ensemble learning model involves the training of different classifiers followed by combining their outputs (using a method such as a neural network) into one classifier to classify new examples. In other words, ensemble learning transfers/combines the outputs of different classifiers into one classifier [30]. It should be noted here that this thesis focuses on three specific VoIP applications, namely Skype, Gtalk and Primus softphone. Since Gtalk and Primus softphone are new applications in the market, obtaining a data set for them is challenging. Therefore, a testbed was set up in the lab to generate traffic for these applications in order to evaluate the proposed approach in classifying multiple VoIP applications.

In summary, the primary objective of this thesis is to explore the robustness of automatically generated signatures for encrypted VoIP traffic. In this case, robustness analysis consists of unseen data sets including (i) different locations (networks), (ii) different time periods, and (iii) evasion attacks. The steps taken to achieve this are enumerated below.

1. The robust classification of encrypted VoIP traffic from a given traffic file without using IP addresses, port numbers and payload information.

2. The identification of a suitable method for sampling training data sets in order to find robust signatures/rules for classifying VoIP traffic;

3. An exploration of performance in classifying encrypted VoIP traffic when no temporal information is used, i.e. Packet Header-based features;

4. An exploration of performance in classifying encrypted traffic when temporal information is used, i.e. Flow-based features;

5. An exploration of limits of employing machine learning algorithms (C5.0, GP and AdaBoost) in order to classify encrypted VoIP traffic robustly;

6. The testing of the ability of the robust signatures against evasion attacks;

7. The analysis of which features are related to the classification target – VoIP encrypted traffic;

8. An exploration of performance of ensemble learning techniques for enhancing the performance of the signatures found by creating one classifier, i.e. combining the C5.0, AdaBoost and GP solutions based on the Flow-based features using a Neural Network based approach;

9. An exploration of capacity of the proposed approach for classifying more than one VoIP application – a multi-class classification based approach.

Such an approach raises several fundamental questions, including: how to establish the data on which 'general' – as opposed to network specific – classification signatures are identified; what representation and corresponding features to assume and what representation should the model of the classification assume to satisfy both real-time (potentially) and accuracy requirements. In this research, use is made of training and test data from entirely independent networks in order to provide some measure of classifier generalization (robustness). Issues of data representation are addressed by employing packet header-based features and flow-based features only without using IP addresses, port numbers and payload data. Specifically, a representation based on packet header information implies a low overhead and low computational cost when deriving corresponding features in real-time, whereas the construction of statistical flows is a much more involved process, but can still be achieved in near real-time. Flows are derived from a 5-tuple consisting of the protocol (TCP/UDP), the 'forward' and 'backward' IP addresses and the corresponding port numbers. When IP numbers match within a finite temporal window 'flow' statistics are calculated.

1.2 Thesis Contributions

In this thesis, I am interested in establishing how to deploy ML algorithms more effectively for practical network operation tasks such as traffic classification. To this end, an automatic classification system that is able to produce robust signatures for classifying encrypted VoIP traffic is developed. The proposed approach contributes to the identification of encrypted traffic by using a minimal set of features which can be utilized for analyzing and developing robust signatures. A significant contribution to the field of network traffic classification was achieved by finding lightweight models which are able to classify VoIP traffic robustly. To this end the novel contributions listed below are presented for consideration.

1. The identification of a suitable method for the challenging problems of sampling a training data set in order to find robust signatures and, hence, improve the performance of the classifiers.

2. The empirical determination that robust signatures can be found using ML techniques with statistical flow-based feature sets

3. The empirical examination of the robust signatures found by the proposed approach on traffic traces from different time periods, locations, networks infrastructures and also against evasion attacks.

1.3 Thesis Organization

In order to achieve the goals of this thesis, Chapter 2 introduces the background literature. It highlights several open source tools, which are able to classify network traffic based on the deep packet inspection method, the port numbers method and machine learning techniques. In addition, the chapter describes two commonly used VoIP applications, namely Skype and Gtalk.

Chapter 3 details the three supervised ML algorithms employed in this thesis (C5.0, AdaBoost and GP). The chapter explains the two feature sets employed. The first feature set is based on the information extracted from the packet header of a packet while the second feature set is based on the statistical information calculated from flows. Furthermore, the chapter defines the term robustness and describes the evaluation schema for evaluating the ML algorithms. Also, a detailed description of the network data sets (traces) employed in this research are presented in this chapter.

Chapter 4 presents the experimental studies performed on subset sampling. It describes in detail the three different techniques used for subset sampling and shows their performances on the test traces. The chapter concludes with a recommendation on the most suitable subset sampling method for finding robust signatures.

In chapter 5 experimental studies are presented comparing three ML algorithms (C5.0, AdaBoost and GP) in classifying VoIP traffic. First, classification techniques based on the packet header-based feature set are presented and then it is demonstrated that by adding the temporal information based on statistical information calculated from flows, i.e. flow-based feature set, not only is classification performance improved but also classification becomes more robust.

After finding the robust signatures, which are able to classify traces from different locations, networks and time periods the robustness evaluations can be used as well against potential evasion attacks. It is believed that the evasion attacks detailed in Chapter 6 is the first time such experiments have been performed against VoIP signatures. To this end, a description of evasion attacks is given in Chapter 6 followed by the details of the experimental setup and results.

Having demonstrated that the signatures found are robust not only on traces from different time periods, locations, and network infrastructures but also against evasion attacks, an analysis of the signatures is performed in detail in Chapter 7. In this chapter, the solutions are analyzed in terms of the time required to train the ML algorithms to obtain them, the number of features used, the number of signatures utilized and the false positive rate triggered to understand fully how the proposed approach solves the problem of classifying encrypted VoIP traffic. The analysis is done both for packet header-based signatures and flow-based signatures.

Chapter 8 introduces the possibility of enhancing the performance of the robust signatures found by combing the solutions of C5.0, AdaBoost and GP using an ensemble learning technique. In this thesis, a neural network based technique is employed to combine the three ML algorithms in an ensemble learner. Chapter 9 demonstrates the ability of the proposed approach to classify more than one VoIP application. Finally, chapter 10 draws conclusions and discusses future research directions.

Chapter 2

Literature Survey and Background

In this chapter, I give an overview of the VoIP applications employed in section 2.1 as well as summarizing previous works on Peer-to-Peer (P2P) and Voice over IP (VoIP) network traffic classification in section 2.2.

2.1 Overview of VoIP Applications

The Internet Protocol Suite, commonly known as TCP/IP, is organized into five layers. Four of these layers are protocols, which build on top of the hardware at the physical layer. These five layers are: (i) Physical Layer; (ii) Data link layer; (iii) Network layer; (iv) Transport layer; and (v) Application layer. The application layer, the top layer of the TCP/IP stack, controls the interaction between the operating system and its application. Users invoke application programs such as a Web browser to access services available through the Internet. The application communicates with one of the transport layer protocols to deliver data. The transport layer, layer 4, regulates the flow of information between two end nodes based on the protocol software. Two protocols, which are used commonly at this layer are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). The UDP protocol is an unreliable connectionless transport protocol. It has no mechanism for ensuring the arrival of packets, ordering the message, or providing information on when to terminate the connection. Furthermore, the UDP protocol has no flow control capability unlike the TCP protocol. The TCP protocol is a reliable and connection-oriented transport protocol. It includes an error-detection mechanism and supports three-way handshaking prior to sending messages to ensure that a virtual channel is established between the two communicating end nodes. The transport layer, layer 4, divides the stream of data into packets with a given destination address and forwards them to the network layer for transmission. The network layer manages the communication from one network node to another by using the Internet Protocol (IP). At this layer, the routing information informs the network on how to send the packet to its final destination and provides this service in a “best effort” way. The Data link layer, layer 2, interacts with the hardware, the Physical layer, to ensure the proper transmission of the data. In this thesis, I am interested in identifying encrypted VoIP applications at the Application layer by using information (feature sets) extracted from the Transport layer, the Network layer and the Data link layer as explained in detail in Chapter 3, section 3.1.

2.1.1 Skype

Skype [2] is a very popular P2P VoIP application developed in 2002. Skype allows its users to communicate through voice calls, audio conferencing and text messages. Although Skype client provides similar functions as MSN and Yahoo instant message applications, the fundamental protocol and techniques it operates are completely different. Since Skype protocols are proprietary and an extensive use of cryptography is implemented by the Skype creators, understanding the Skype protocol is a difficult task. Moreover, Skype employs a number of methods to circumvent NAT and firewall restrictions [7], which increases the difficulty of understanding the Skype protocol.

Skype is based on a P2P architecture with the user’s authentication based on a central architecture (client-server model via public key mechanisms). After authentication is completed, all communication is performed on the P2P network. Therefore, user information and search queries are stored and broadcast in a decentralized approach. On the P2P network, there are two types of nodes: ordinary nodes (hosts) and supernodes. An ordinary node is a Skype client, which can be used to communicate through the service provided by Skype. On the other hand, any node on the P2P network with sufficient CPU power, memory and network bandwidth is a suitable candidate for a supernode. A supernode is part of the decentralized Skype network, which can ease the routing of Skype traffic to bypass NATs and firewalls. Moreover, ordinary hosts have to connect to a supernode and register with the Skype login server in order to join the P2P network.

Skype uses TCP or UDP at the Transport layer to provide services to users such as voice and video calls, file transfer, chat and conference calls. For network communications, Skype prefers the UDP protocol when there are no restrictions. The communication among peers (users) on the P2P network is established via the Internet Protocol (IP) paradigm. However, in the case of NATs and firewall restrictions, Skype prefers using either port 80 (HTTP) or port 443 (HTTPS). Furthermore, Skype has the ability to route (overlay routing) traffic via its supernodes to circumvent the NATs and firewall restrictions. A more detailed description of the Skype protocol and internals can be found in [7, 31].

2.1.2 Gtalk

Gtalk [3] is an application developed by Google, which is based on Extensible Messaging, the Presence Protocol and the Jabber protocol (XMPP/Jabber) [32]. It provides many services to end users, these are: (i) voice communication, (ii) video communication, (iii) file transfer and (iv) chat services. The communication between users is established using a traditional end-to-end IP paradigm, but Gtalk call is routed through a relay node to ease the traversal of symmetric NATs and firewalls. Though Gtalk may relay on TCP and UDP at the Transport layer, preferably, communication data are carried over UDP. The user's authentication is performed by a client-server architecture using public key mechanisms. After a user (client) is authenticated all further communication is carried out with the 'nearest' Google server relay node. In this way, not only can the quality of service be guaranteed by Google but also both the scaling and control issues can be solved in a more seamless way. A Gtalk client stores all the user information on the server side and nothing on the client side. There is no direct communication with Gtalk clients to establish communication: the Gtalk clients on the caller side and callee side have to communicate through a Google server(s). Depending on the network restrictions, the communication between a Gtalk client and a Google server could use TCP or UDP. However, Gtalk prefers UDP and if it is behind a firewall or NAT then it uses HTTPS (TCP port 443). Furthermore, one of the main advantages of Gtalk is that Gtalk can benefit from the vast numbers of Google servers which are being used as relay nodes (super nodes) to ensure quality of service. Moreover, Gtalk is different from Skype in terms of preserving the bandwidth of the client machine by not letting the client to become a super node. A more detailed description of the Gtalk protocol can be found in [3, 33].

2.1.3 Other Common Applications

Yahoo! Messenger (YMSG) [6] uses a client-sever architecture for normal operations and a voice-chat service [34, 35]. The client needs to contact one Yahoo server and then route all subsequent communications through that server. It uses the Session Initiation Protocol (SIP) [36] for signalling and transmits the data via the Real-Time Transport Protocol (RTP) [37]. SIP is an application layer text based protocol for establishing multimedia sessions and adapts a client-server architecture. Several major companies, including Microsoft and Yahoo, have chosen SIP for signaling VoIP traffic. SIP messages can be transmitted over UDP, TCP or SSL. MSN and YMSG VoIP applications use SIP for creating, modifying and terminating voice sessions. However, in SIP voice, the voice and video communication are carried over the RTP protocol whose job it is to carry voice data from caller to callee.

Microsoft Messenger (MSN) [4] has a similar approach to that used by YMSG [34, 35]. It employs a client-server architecture for normal operations. MSN uses SIP for voice communication and transmits the data through RTP protocol. The major difference between YMSG and MSN is that YMSG needs only to contact one server and can provide all services via that server whereas MSN has different types of servers for each service it provides, including login, user search and voice. Moreover, its traffic is routed to the appropriate server based on the services requested by the client. Finally, MSN does not have any encryption capabilities while YMSG provides encryption. Gtalk is similar to YMSG and MSN for using a central server for user authentication. However, Gtalk is different from YMSG and MSN in its interior protocol design. YMSG and MSN depend on SIP and RTP to start, establish and end voice communications while Gtalk uses Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (STUN) [38] and XMPP/Jabber protocols to establish voice communications.

Furthermore, the Primus Enterprise VoIP network has developed a soft Talk Broad Band phone (softTBB) [5], which runs over SIP [36]. SIP is responsible for setting up, validating and completing calls over the Internet. The main components of the Primus Enterprise VoIP network are: (i) IP phone [a terminal (softTBB software) with native VoIP support and direct connection to the Internet]; (ii) Primus Voice Gateway with the ability to convert network signals from/to both telephony interfaces and VoIP protocols; and (iii) Primus SIP server, which is responsible for providing the management and administrative functions with essential support for routing calls across the network. For a call to a PSTN phone, the calls are routed to the nearest Primus Voice gateway for establishing communication and converting the calls between the VoIP network and the PSTN network.

2.2 Traffic Classification: Specifically P2P and VoIP

Traffic classification has been a consistently popular research topic in computer networks. However, with the increase in the number of VoIP, P2P and encrypted applications, interest has increased as well. Jordan [39] lists four questions, which can be used as a framework for guiding the use of traffic classification practices by ISPs in the United States and for determining whether the network management polices are reasonable / acceptable or unreasonable / unacceptable. The first two questions are relevant for the Internet Protocol Suite while the other two questions are related to traffic management at the ISP end (who should manage traffic and when it should be managed). The first two questions are the most related to this research and they are in the form of *where* and *what*. These two questions are *where* in the network layers the monitoring should be applied and *what* type of monitoring should be applied (e.g. blocking or terminating of a connection). Jordan suggested that the monitoring of network packets should not violate network layering, which is what this research does since the stated aim is to identify encrypted VoIP applications at the Application layer by employing feature sets obtained from the Transport layer, the Network layer and the Data link layer without using the IP addresses, source/destination ports and payload information. He suggested as well that the type of traffic monitoring should be associated with the enhancement or degradation of QoS, which is what this research does by identifying signatures that can be used to enhance or degrade VoIP traffic.

Two survey papers [40, 41] summarize the research in IP traffic classification from 1997 to 2008. Firstly, Nguyen and Armitage [40] focus on research in the literature, which use machine learning techniques to classify IP network traffic. They reviewed eighteen significant works from 2004 to 2008, which employ supervised learning techniques, unsupervised learning techniques and hybrid techniques, which combined supervised and unsupervised techniques for traffic classification. Secondly, Callado et al. [41] give details about the challenges in the area of IP traffic analysis and application classification. They reviewed papers from 1997 to 2008. They divided the techniques for traffic analysis and classification into two categories: packet-based and flow-based.

This chapter discusses on the research in the literature on automatic identification/classification of P2P applications and VoIP applications. The review is divided into five areas: (i) machine learning methods for classifying P2P network traffic (section 2.2.1); (ii) Port-based methods for classifying P2P network traffic (section 2.2.2); (iii) Deep Packet Inspection (DPI) methods for classifying P2P network traffic (section 2.2.3); (iv) Open source network traffic classification tools (section 2.2.4) and (v) Alternative methods for classifying P2P network traffic (section 2.2.5).

To the best of my knowledge, the focus of the literature for detecting VoIP traffic is on Skype traffic since Skype is the dominant P2P VoIP application with more than 246 millions users [42]. Therefore, Skype analysis has become popular in the last five years, in part due to the combination of the encrypted operation and dynamic nature of the port assignment making traditional methods of traffic identification insufficient. Baset and Schulzrinne [7] studied Skype during login, Network Address Translation (NAT) and firewall traversal, and call establishment under three different network arrangements in order to understand Skype behaviour and reverse engineer its protocol. Guha et al. [43] presented an experimental study of Skype in which they collected Skype traffic for five moths. They narrowed their approach to include only sessions relayed via a supernode. Rossi et al. [44] concentrated their analysis on the Skype signaling traffic while Yu et al. [45] inspected the characteristics of the Skype P2P overlay network. Suh et al. [46] monitored Skype relay nodes. They used statistical analysis based on inter-arrival time, byte size ratios and maximum cross correlation to detect Skype relay traffic. Furthermore, Cicco et al. [47] examined the Skype congestion control algorithm considering video call flow behaviour over TCP, taking into account network time variation.

2.2.1 Machine Learning Methods for Classifying P2P Network Traffic

In the literature, Bonfiglio et al. [48] present one of the earlier studies in classifying Skype traffic using supervised learning techniques. They introduced two approaches for classifying Skype traffic. The first approach is to classify Skype client traffic based on Pearson's Chi-Square (χ^2) test using information revealed from the message content randomness (e.g. the FIN and ID fields) introduced by the cypher and the header format. Their second approach is to classify Skype VoIP traffic based on the naïve Bayesian classifier using packet arrival rate and packet length. They obtained the best results when the first and second approaches were combined. They achieved approximately a 1% false positive rate and a 2-29% false negative rate, depending on the data sets. However, they employed a payload-based classification scheme as well and used a priori knowledge for UDP detection. Freire et al. [49] studied detecting Skype flows in Web traffic by using metrics derived from the χ^2 value and the Kolmogorov-Smirnov distance using features based on Web request/respone size, the number of requests and the time taken to detect Skype and Gtalk traffic. They achieved 100% DR and 5% FPR for both applications. Recently, Este et al. [50] applied Support Vector Machines (SVM) for classifying only TCP bi-directional flows relying mainly on the packet size as the main feature. The SVM models are able to classify multiple applications such as HTTP, HTTPS, BitTorrent, e-Donkey, Kazza, Gnutella and MSN. The classification method is based on two phases. For the first phase, they build a one-class SVM classifier in which the classifier can determine the application of the TCP flow-based on where the flow feature values fall in SVM hyperplanes (surfaces). For the second phase, they built a multi-class SVM based classifier. The second phase is called when the TCP flow falls inside more than one surface (class) in order to determine the correct application class. Furthermore, if the flow falls outside the surface, the flow is marked as 'unknown.' They tested their methods on three traces, which were captured from different locations. They were able to achieve high performance on the P2P e-Donkey flows but had poor results on other P2P applications such as Kazza and Gnutella. However, they did not look at the transportability/robustness of the SVM models.

Unsupervised learning methods have been used in network traffic classification as well. Bernaille et al. [15] used an unsupervised learning method to cluster the network traffic in order to label it according to the application protocols. They clustered the first five packets of TCP flows based on the packet size in each connection. They used the K-Means algorithm with the Euclidean distance to build an online classifier consisting of fifty clusters to classify only TCP network flows. The classifier begins by building the flows based on the 5-tuple (Protocol, Source/Destination IP addresses and Source/Destination port numbers) from the TCP header and calculating packet size. Then, the classifier searches all fifty clusters to label the coming traffic (new flow) according to application type. They were able to classify accurately more than 80% of the P2P application traffic. In particular, they achieved 95.24% accuracy for Kazaa and 84.2% accuracy for e-Donkey traffic. However, the classifier has problems handling similar flow sizes employed by different applications, basically labelling the flows the same way. Erman et al. [12] investigate K-Means and density-based spatial clustering of applications with noise (DBSCAN) and AutoClass algorithms. They used features based on bidirectional flows (Table 2.1) and evaluated the clustering algorithms on two traces: a public trace (Auckland IV2 [51]) and a trace collected at the University of Calgary. Results showed that the K-Means algorithm had an accuracy over 85% and was more suitable to the clustering of Web, P2P and FTP traffic since it was faster than AutoClass and DBSCAN. On another study, Erman et al. [52] addressed the problem of asymmetric flow data at the core of the network. They focused on TCP flows since partial information about the direction of TCP connections was available on the backbone traffic. They achieved an overall accuracy of 95% using K-Means when flow statistics from the server to the client were used. However, for P2P traffic they achieved 77% recall and 82% precision. Furthermore, they proposed an algorithm which could estimate flow statistics such as the number of bytes/packets of TCP flow from unidirectional traces where the direction of the flow was not observed.

Furthermore, Erman et al. [53] are among the first researchers to apply a semi-supervised technique for classifying such internet flow traffic as the Web, FTP, and P2P file sharing. The semi-supervised learning method consists of two steps. During the first step, the K-Means algorithm with Euclidean distance is used to cluster traffic. The clusters contain pre-labelled flows and unlabelled flows. The second step involves using the maximum likelihood estimate for the pre-label flows within a

Table 2.1: Flow Feature Set for the Clustering Technique Used by Erman et al. in [12]

Numbers	Features
1	Total number of packets
2	Mean packet size
3	Mean payload size excluding headers
4	Number of bytes transferred in forward direction
5	Number of bytes transferred in backward direction
6	Total number of bytes transferred
7	Mean inter-arrival time of packets

cluster to map the cluster into known traffic application classes. Clusters, which have no pre-label flows are mapped into the 'unknown class.' They applied the backward greedy feature selection algorithm to choose eleven flow features (Table 2.2). They achieved a best performance of ≈98% flow classification accuracy and ≈93% byte accuracy. Their performance metric is based only on accuracy, which is the number of correctly classified instances divided by the total number of instances, rather than providing classification results based on the false positive rate as well. Unfortunately, this may be misleading, especially on unbalanced data sets in which the data set consists of, say, two classes only (in a total of one hundred instances), 10 instances of FTP and 90 instances of P2P. By labelling everything as the major class, a classifier can achieve 90% accuracy but the false positive rate for P2P would be 100%.

Table 2.2: Flow Feature Set for the Semi-Supervised Classifier Used by Erman et al. in [53]

Numbers	Features
1	Total number of packets
2	Average packet size
3	Total bytes
4	Total header (transport plus network layer) bytes
5	Number of caller to callee packets
6	Total caller to callee bytes
7	Total caller to callee payload bytes
8	Total caller to callee header bytes
9	Number of callee to caller packets
10	Total callee to caller payload bytes
11	Total callee to caller header bytes

Recently, Iliofotou et al. [54] employed Traffic Dispersion Graphs (TDGs) for classifying P2P traffic (e.g. Gnutella, e-Donkey and BitTorrent). Their approach worked by grouping the first sixteen bytes of the payload using the K-Means algorithm. These bytes act as categorizing features ranging from 0 to 255. Then, the TDGs are used to classify the clusters. They applied their approach on backbone traffic collected from different sites and showed that they were able to classify 90% of P2P traffic with an average precision of 95%.

2.2.2 Port-based Methods for Classifying P2P Network Traffic

Port numbers have been used as a discriminator for identifying application traffic. Moore and Zuev [17] used the Naïve Bayes estimator to classify traffic into different categories. They distinguished eleven discriminators (Table 2.3), including the port number, out of 248 per-flow discriminators. They grouped network traffic into ten categories (Table 2.4), considering only complete TCP connections in their experiments. Their results showed that with the Naïve Bayes estimator they could achieve an average 65% accuracy, where accuracy is calculated as the total number of correct instances divided by the total number of instances. In addition, they enhanced the overall accuracy of the classification to 95% by using kernel-estimates combined with the Fast Correlation-Based Filter (FCBF) technique for discriminator reduction, but their best results on the P2P category was a 55.18% detection rate.

Table 2.3: Most Important Distinguishable Discriminators Identified by Moore et al. in [17]

Numbers	**Discriminators**	**Direction**
1	Port server	
2	# of pushed data packets	server-to-client
3	Initial window bytes	client-to-server
4	Initial window bytes	server-to-client
5	Average segment size	server-to-client
6	IP data bytes median	client-to-server
7	Actual data packets	client-to-server
8	Data bytes in the wire variance	server-to-client
9	Minimum segment size	client-to-server
10	RTT samples	client-to-server
11	Pushed data packets	client-to-server

Table 2.4: Network Traffic Grouped by Moore et al. in [17]

Classification	**Example Application**
BULK	FTP
DATABASE	postgres, sqlnet, oracle, ingres
INTERACTIVE	SSH, klogin, Rlogin, Telnet
MAIL	IMAP, POP2/3, SMTP
SERVICES	X11, DNS, IDENT, lDAP, NTP
WWW	WWW
P2P	Kazaa, BitTorrent, Gnutella
ATTACK	Internet worm and virus attacks
GAMES	Half-Life
MUTIMEDIA	Windows Media Player, Real

Furthermore, Moore et al. [9] extended the work to classify network flow traffic using nine distinct identification methods (Table 2.5) in order to study the classification of different applications. They used the standard port number as their first classification method. After that they used the header information for bidirectional flows as their second method. The final methods employ host rules for classifying the flows. The classification works by the flow going through different sub-methods for classification (Figure 2.1). They used port numbers and payload information for their methods. Furthermore, human interference is required when the flow has more than one match in order to identify the application accurately. They used accuracy as their only evaluation method, achieving 99.9% accuracy using the nine methods.

Table 2.5: Nine Identification Methods Used by Moore et al. in [9]

Identification Methods	**Example**
I Port-based classification (only)	–
II Packet header (including I)	simplex flows
III Single packet signature	May worm/virus
IV Single packet protocol	IDENT
V Signature on the first KByte	P2P
VI first KByte protocol	SMTP
VII Selected flow(s) protocol	FTP
VIII (All) Flow Protocol	VNC, CVS
IX Host history	Port-scanning

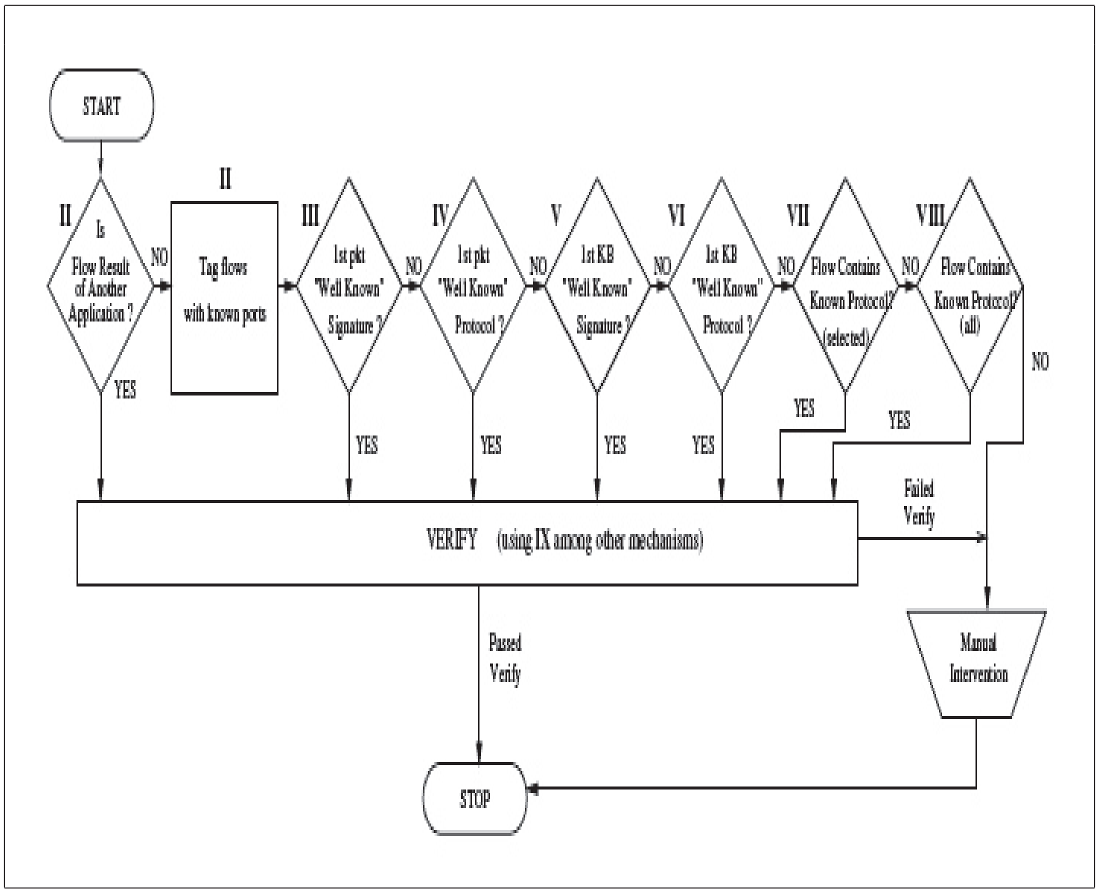

START
II
Is Flow Result of Another Application ?
NO
YES
II
Tag flows with known ports
III
1st pkt "Well Known" Signature ?
IV
1st pkt "Well Known" Protocol ?
V
1st KB "Well Known" Signature ?
VI
1st KB "Well Known" Protocol ?
VII
Flow Contains Known Protocol? (selected)
VIII
Flow Contains Known Protocol? (all)
NO
YES
VERIFY (using IX among other mechanisms)
Failed Verify
Passed Verify
Manual Intervention
STOP

Figure 2.1: Different Sub-methods Used by Moore et al. in [9]

Ehlert and Petgang [55] studied Skype signaling traffic looking for different patterns which would enable Skype to be detected. They built signatures based on port usage, packet sizes and payload content. They were able to identify Skype signaling traffic for Skype versions 1.4, 2.0 and 2.5. However, the use of port numbers and payload information makes their signatures less robust since the port number can be changed easily.

Bernaille and Teixeira [56] employed first clustering and then classification to the first few packets in each connection to identify Secure Socket Layer (SSL) connections and applications running over SSL. They used the first four packets of a TCP connection and represented it using the 5-tuple (destination/source IP addresses, destination/source port numbers and protocol) and the packet size. They achieved high performance on P2P traffic over SSL (a true positive of ≈86% for BitTorrent and ≈97% for e-Donkey). Li and Moore [57] employed C4.5 to classify different classes of network traffic. They used server/client port numbers in their feature sets. They achieved 99% recall for P2P flows. The P2P applications were Napster, Kazaa, e-Mule, Gnutella and e-Donkey. Recently, Li et al. [58] employed the C4.5 algorithm to classify network applications, using the first five packets of a connection. They classify bi-directional flows of the TCP and UDP protocols using features extracted from the packet headers including the source/destination port numbers. They reported a good performance for an online real-time classification of network traffic. However, when they tested the trained model on a data set different from the training data set, the performance of their classifier dropped significantly (e.g. on the interactive applications, such as SSH, the Recall was less than 30%).

2.2.3 Deep Packet Inspection (DPI) Methods for Classifying P2P Network Traffic

Zhang and Paxson [59] present one of the earlier studies of techniques based on matching patterns in the packet payloads. They developed an algorithm for classifying interactive network traffic based on matching patterns in the packet payloads, packet size, inter-arrival time and direction [59]. They developed signatures for classifying the traffic from seven interactive applications but they evaluated only the signatures for Telnet, Rlogin, SSH, FTP, and Root prompt, since there was no traffic for Napster and Gnutella in their test traces. They had a high performance on different test traces. Sen et al. [11] provide an approach based on available documentation and packet payload to derive application signatures for five P2P protocols: Gnutella, e-Donkey, DirectConnect, BitTorrent and Kazaa. They achieved high performance on all the test traces with less than 5% for both the false positive rate and false negative rate.

Chung et al. [60] compared three similarity metrics (Jaccard similarity, Cosine similarity, and Gaussian radius based function) for payload-based classification. They used accuracy as their evaluation metric. Results showed that the Jaccard Similarity had the best performance (95% accuracy on P2P traffic). Recently, Finamore et al. [61] proposed a deep packet inspection classifier for UDP traffic called KISS. KISS inspects the first byte of the payload and the packet header to derive statistical signatures by using Pearson's Chi-Square (χ^2) test. KISS adopts SVM for classification. KISS was able to identify a wide range of applications correctly, including DNS, RTP, P2P e-Mule, BitTorrent and P2P-TV applications with a 99% true positive rate. However, KISS is not able to classify encrypted traffic.

2.2.4 Open Source Network Traffic Classification Tools

There are several open source tools, which can classify traffic by using DPI or port numbers to classify network traffic. This section summarizes some of these tools. Wireshark and CoralReef are two well-known open source tools which use port numbers to label traffic. Wireshark [62], formerly known as Ethereal, is the most popular open-source, cross-platform network analysis tool. Network packets can be analyzed by Wireshark either online or offline. Wireshark makes of the libpcap [63] library for packet sniffing. Further, Wireshark is available for different platforms such as Unix-based, Windows-based and Apple machines. Wireshark labels traffic based on the standard port numbers. On the other hand, the CoralReef suite is a passive traffic monitoring tool which allows users to measure and analyze network traffic. The CoralReef software project is developed by the Cooperative Association for Internet Data Analysis (CAIDA) [64, 65]. CoralReef provides a large number of features at every layer of the network protocol stack and works on different operating system platforms.

Furthermore, DPI tools can classify network traffic by matching regular expressions with the payload bytes of packets. There are many open source tools such as OpenDPI and L7–filter. L7–filter [66] is a deep packet inspection tool designed for the Linux kernel. It uses regular expression matching of the first few packets of the application layer connection to classify traffic. It is capable of classifying many applications such as Kazza, BitTorrent, SSL, SSH, HTTP etc. The L7–filter is available as well in lighter version called 'L7–filter Userspace.' The IPOQUE company [67] has developed a software library which can classify network traffic based on pattern matching and statistical techniques. The IPOQUE company releases a lighter open source version called OpenDPI [68] which employs deep packet inspection to classify network traffic.

Furthermore, Bonfiglio et al. [48] added their method, which is based on the naïve Bayesian classifier using features such as packet length and inter arrival time to classify Skype traffic to the Tstat tool. Tstat is a real-time passive analysis tool derived form the 'tcptrace', which was developed by the Politecnico di Torino networking research group [69]. Tstat provides more than 80 different measures and statistics output to network traffic. Tstat is capable of classifying network traffic using deep packet inspection techniques and machine learning techniques as well (Naïve Bayesian classifier [48]). It inspects the packet payload bytes of one packet and matches them with the protocol's signatures to classify applications such as PPLive, BitTorrent, e-Mule, and Gnutella. If the application cannot be defined by inspecting one packet, then it inspects several packets. Using this technique, Tstat is able to identify applications such as MSN, XMPP/Jabber, Yahoo and SSL.

2.2.5 Alternative Methods for Classifying P2P Network Traffic

Many methods using host information, pattern matching and nonlinear recurrence plot-based approaches have been employed for classifying network traffic. Karagiannis et al. [13] designed a framework called BLINC, which was able to classify network flow traffic by analyzing patterns at three levels: (i) the social level; (ii) the functional level; and (iii) the application level. At the social level, they evaluate the popularity of a host by counting the number of communications to it at the Network layer. Basically, they counted the number of destination IPs interacting with the host. At the functional level, they evaluated the behaviour of the host as either a client or server in order to identify host functionality on the network by analyzing port usage at the Transport layer. Primarily, they investigated source/destination IP addresses and the source port numbers. Lastly, at the application level, they identified the application running on a host using two steps. In step one, they classified based on the source/destination IP addresses and source/destination port numbers. In step two, they refined the classification using graphs based on flow features such as protocol name, the number of packets and the number of bytes. In summary, they gathered flow information at the transport and network layers and investigated the relationships between host behaviour and the applications in order to classify flows. They captured three traces from different locations and developed deep packet inspection classifiers based on matching bit strings in order to set the ground truth of the traces. They grouped the flows into eleven categories (Table 2.6). They were able to identify 80% to 90% of the flows with an accuracy of 95%, where they defined accuracy as the percentage of correctly labelled traffic by BLINC. However, BLINC has a limitation of not being able to classify individual flows and may not be capable of classifying distinct applications [70].

Table 2.6: Network Traffic Grouped by Karagiannis et al. in [13]

Category	**Application/protocol**
WWW	WWW
P2P	FastTrack, e-Donkey2000, BitTorrent, Gnutella,WinMx, OpenNap, Soulseek, Ares, MP2P, Dirrect Connect, GoBoogy, Soribada, PeerEnabler
data (FTP)	FTP, database (MySQL)
Network management (NM)	DNS, Netbios, SMB, SNMP, NTP, spamassasin, GoToMyPc
mail	mail (SMTP, POP, IMAP, IDENTD)
news	news (NNTP)
chat/irc (chtirc)	IRC, MSN messenger, Yahoo messenger, AIM
streaming (strm)	mms (wmp), real, quicktime, shoutcast, vbrick streaming, logitech Video IM
gaming	(game) HalfLife, Age of Empires, etc.
Nonpayload	Packet without message
Unknown	Unknown applications

Suh et al. [46] concentrated on the classification of relayed traffic and monitored Skype traffic as an application using relay nodes. A relay node is part of the decentralized Skype network which can ease the routing of Skype traffic to bypass NATs and firewalls. They used several metrics based on features such as inter-arrival time, byte size ratios and maximum cross-correlation between two relayed bursts of packets to detect Skype relay traffic. Their results (a 96% true positive and 4% false positive) show the technique is reliable in recognizing relayed Skype sessions but it might not be appropriate for classifying all Skype VoIP traffic. Palmieri et al. [71] used nonlinear recurrence plot-based approach based on two flow features to classify traffic flows: average packet length and inter-arrival time variance. Their result on e-Donkey P2P flow traffic was a ∼78% true positive rate. Keralapura et al. [72] built a two-stage P2P traffic classifier called Self-Learning Traffic Classifier (SLTC) which used signature matching and pattern classification methods. In the first stage, they proposed a Time Correlation Metric (TCM) algorithm to identify either P2P nodes or P2P supernodes and classify in/out flows from these nodes as P2P flows. The TCM is based on the assumption that when a new node joins the P2P network, it tries to connect to a supernode. Then, the supernode would open a connection to more than one node and forward the information on the new node to other supernodes on the P2P network. The TCM algorithm monitors this temporal correlation from the incoming to outgoing connections to identify the P2P nodes. Results showed that the TCM is able to discover P2P nodes with 95% accuracy with 0% false positives. When the network flows go through the first stage, the matching P2P flows would be flagged as “Known” and no further process would be required. However, if there were no matching signatures, they would flag the flows as “unknown” and would forward layer 3 and layer 4 of the TCP/IP network packet information to the next stage for further analysis. In the second stage, they aggregated the row packets together and classified packets using signatures based on the destination IP, the destination port number, the number of packets in flow and the number of bytes in each packet. They achieved a 95% detection rate with 0% false positive rate for the P2P traffic.

2.2.6 Importance of Robustness/Generalization for the Classification of Network Traffic

In terms of the generalization/robustness of the classifier, Park et al. [73] pointed out the importance of finding a robust classifier. They used different data for training and testing for network traffic classification, which were different in terms of time periods and locations. Their scheme consisted of using Genetic Algorithms (GA) for feature set reduction and using a decision tree as a classifier. However, they did not provide the results of individual applications but rather they provided the overall results. Unfortunately, this can be misleading – particularly on unbalanced data sets when the data set consists of two classes (a small class and a major class) – when simply classifying everything as the major class; a classifier can achieve high accuracy. Hu et al. [74] built an approach based on behavioural profiling with a two-level matching method (host-level matching and flow-level matching) to identify P2P flow traffic, using BitTorrent and PPlive as the two case studies. Their flow features were based on the five flow tuples (srcIP, destIP, srcPort, destPort, protocol) and statistical flow features. They used the Apriori algorithm to achieve a compact set of flow patterns and build their rule sets using maximal association rules. They used accuracy as their evaluation criterion. They obtained an average accuracy for PPlive and BitTorrent when the validation data set was different from the training data set of ≈98% and ≈97% for PPlive over TCP and UDP respectively and ≈94% and ≈96% for BitTorrent over TCP and UDP, respectively. However, they did not report true positive and false positive rates. Moreover, they discussed how their methods could be evaded by malicious users, leaving any experiments to this end for future work. By contrast, Wright et al. [75] used traffic morphing techniques to make one application's traffic look similar to another application's traffic by padding the payload of a packet. They used accuracy as their only performance measurement and showed that the morphing techniques were able to reduce the accuracy of their own VoIP language classifier [76], from 71% to 30% and to reduce a web page classifier, designed by Liberatore et al. [77] from 98.4% to 4.5%. Then, the results of the classifiers employed were enhanced significantly when the morphed data included in the training data set and the experiments were repeated.

In this thesis, the aim is to use generic feature sets and let the employed machine learning algorithms identify the subsets of it for classifying any given application. The aim is to perform an investigation of C5.0, GP and AdaBoost based classifiers for the identification of VoIP (e.g. Skype) encrypted traffic as well as explore their robustness. The focus is on the robustness of the classifiers since it is important to demonstrate the robustness/generalization of such classifiers not only by evaluating them on unseen data from the same network but also by evaluating them against unseen data captured from different networks/locations at different time periods as well as against potential evasion (bypassing classifiers) attacks. In the literature, the evaluation of classifying network traffic traces using ML solutions against different network traffic traces or evasion attacks were reported, but to the best of authors' knowledge this is the first time that the generalization/robustness of classifiers was evaluated under the three aforementioned conditions for traffic classification specifically for encrypted P2P VoIP applications such as Skype.

Chapter 3

Methodology: Learning Algorithms and Data Sets Employed

In this research, the focus is on the application of supervised ML-based techniques to network traffic classification, specifically classification of Skype-encrypted VoIP traffic. To this end, three different supervised machine learning algorithms, AdaBoost, C5.0 and Genetic Programming (GP), are evaluated to generate signatures automatically in order to identify VoIP traffic robustly. The ML techniques require a number of steps such as selection of features, labelling of the data set, training of the learning algorithms and testing the solutions. The details of these steps and the data sets employed are presented in this chapter.

3.1 Feature Sets

Features consist of a small set of identifiers which are required to describe a data set. In other words, a feature set is a vector of information which describes each record of a data file that is input to a ML algorithm. The type of features is important in order for the machine learning algorithm to quantify the characteristic of the network traffic class. In this research, issues of data representation are addressed by employing packet header-based features and flow-based features but without using IP addresses, port numbers and payload data. Specifically, a representation based on packet header information implies a low computational cost when deriving corresponding features in real-time, whereas the construction of statistical flows is a much more involved process. However, this can be accomplished in near real-time. What is meant by flows is a communication between two network nodes where they share the same 5-tuple information (destination/source IP addresses, destination/source port numbers and protocol) for the two network nodes. Then, statistical information is calculated within a finite temporal window once the 5-tuple information is matched. Through the experiments, the best attribute set will be recommended for finding robust signatures based on the performance of the ML algorithms.

3.1.1 Packet Header-based Feature Set

In this case, each packet is described in terms of 29 packet header-based features (Table 3.1) in which the underlying principle is that features employed are simple and clearly defined within the networking community. They represent a reasonable benchmark feature set to which more complex features might be added in the future. To this end, Wireshark [62] is employed to process data sets and to generate features. As discussed earlier, this does not use the IP addresses, port numbers and payload data. The packet header feature set represents the TCP/IP Internet protocol stack where the information from the Transport, Network and Data link layers are gathered to identify applications running at the Application layer. Features 1 to 5 correspond to the Data link layer while features 6 to 13 represent the IP header (Network layer). Finally, features 14 to 29 are extracted from the two common protocols (TCP and UDP) which are running on the Transport layer.

Table 3.1: Packet Header-based Features Employed; * Denotes that the Feature is Normalized by log

Number	**Feature Name**	**Description**
1	frame.time_delta	Delta time from previous captured packet
2	frame.pkt_len*	Packet Length
3	frame.len*	Frame Length
4	frame.cap_len*	Capture Length
5	frame.marked	Frame is marked
6	ip.len*	IP Header length
7	ip.flags	IP Flags
8	ip.flags.rb	IP Flags: Reserved bit
9	ip.flags.df	IP Flags: don't fragment
10	ip.flags.mf	IP Flags: More fragments
11	ip.frag.offset	IP Fragment offset
12	ip.ttl*	IP Time to live
13	ip.proto	IP Protocol
14	tcp.len*	TCP Segment Length
15	tcp.seq*	TCP Sequence number
16	tcp.nxtseq*	TCP Next sequence number
17	tcp.ack*	TCP Acknowledgement number
18	tcp.hdr_len*	TCP Header length
19	tcp.flags*	TCP Flags
20	tcp.flags.cwr	TCP Flags: Congestion Window Reduced
21	tcp.flags.ecn	TCP Flags: ECN-Echo
22	tcp.flags.urg	TCP Flags: Urgent
23	tcp.flags.ack	TCP Flags: Acknowledgment
24	tcp.flags.push	TCP Flags: Push
25	tcp.flags.reset	TCP Flags: Reset
26	tcp.flags.syn	TCP Flags: Syn
27	tcp.flags.fin	TCP Flags: Fin
28	tcp.window_size*	TCP Window size
29	udp.length*	UDP Length

3.1.2 Flow-based Feature Set

In this case, network traffic is represented using flow-based features. As discussed earlier, flows are bidirectional streams of packets between two hosts (client and server) where they share the same 5-tuple (source/destination IP addresses, source/destination port numbers and protocol). Client to server direction decides the forward direction. Moreover, flows are of limited duration. UDP flows are terminated by a flow time-out while TCP flows are terminated upon proper connection teardown or by a flow timeout, whichever occurs first. The flow time-out value (600 seconds) employed in this research is used as specified in [78]. The flow features employed are based on direction, inter-arrival time and inter-packet length. In this research, each network flow is described by a set of statistical features (Table 3.2). Here, a feature is a descriptive statistic which can be calculated from one or more packets, which provides a label (e.g. {Skype, non-Skype}) for each flow. To this end the NetMate tool [79] is employed to process packets, generate flows and compute feature values. Finally, only those UDP and TCP flows which have no less than one packet in each direction and transport no less than one byte of payload are considered. As discussed earlier, features such as IP addresses, source/destination port numbers and payload are excluded from the feature set to ensure that the results are not dependent upon such biased features.

Table 3.2: Flow-based Features Employed

	Abbreviation	**Feature Name**
1	proto	Protocol
2	Duration	Duration of the flow
3	fpackets	# Packets in forward direction
4	fbytes	# Bytes in forward direction
5	bpackts	# Packets in backward direction
6	bbytes	# Bytes in backward direction
7	min_fiat	Min forward inter-arrival time
8	mean_fiat	Mean forward inter-arrival time
9	max_fiat	Max forward inter-arrival time
10	std_fiat	Std deviation of forward inter-arrival times
11	min_biat	Min backward inter-arrival time
12	mean_biat	Mean backward inter-arrival time
13	max_biat	Max backward inter-arrival time
14	std_biat	Std deviation of backward inter-arrival times
15	min_fpkt	Min forward packet length
16	mean_fpkt	Mean forward packet length
17	max_fpkt	Max forward packet length
18	std_fpkt	Std deviation of forward packet length
19	min_bpkt	Min backward packet length
20	mean_bpkt	Mean backward packet length
21	max_bpkt	Max backward packet length
22	std_bpkt	Std deviation of backward packet length

3.2 Labels, Training and Testing Data Sets

In this research, the label of a data record is a class, which indicates the type of the IP traffic. Labels reflect the ground truth of a given data set. Thus, if the traffic type is known, a label is provided for each packet/ flow (data record) in the data sets.

Machine learning algorithms need to be trained using a data set (called training). Once they are trained, they give a solution, which consists of the rules or the model they generate. This solution (output) can be validated and tested on a test data set (previously unseen data instances). Sampling a representative subset data for training the machine learning algorithms is a difficult task in order to model or characterize the different behaviour of an application's traffic (e.g. Skype traffic). In this thesis, Chapter 4 describes how the training data sets are sampled. Moreover, test data sets are important for determining the performance of the best machine learning algorithms on unseen data/network traces and hence is an important step in identifying the robustness of the classifiers. Section 3.6 describes the test data sets employed in this thesis in more detail.

3.3 Robustness of Machine Learning Algorithms

In most cases in the literature [9, 15, 17, 22, 23, 70, 74, 75], researchers have evaluated the performance of their approaches on traces from the same network on which the model was trained, even though the testing data sets were unseen during training. It is important to evaluate the robustness/generalization of the solutions of the machine learning algorithms, specifically C5.0, GP and AdaBoost, in traffic classification (e.g. VoIP). In this thesis, robustness/generalization is defined from three perspectives: testing on (i) unseen data from different locations and network infrastructures; (ii) unseen data from different time periods and (iii) unseen altered data by padding/morphing, i.e. evasion attacks. It is believed that this is the first research to evaluate robustness of traffic classification signatures produced by ML algorithms on all three criteria. The quantification of the robustness of the signatures can be measure in terms of achieving moderate performance accuracy on test traces, which is higher than 50% (random guessing).

3.4 Machine Learning Algorithms Deployed

In this section, the three machine learning techniques employed in this thesis (C5.0, AdaBoost and GP) are summarized.

3.4.1 C5.0

C5.0 is the commercial tool [80] developed from the C4.5 decision tree algorithm. C5.0 incorporates new (relative to C4.5) technologies such as boosting and the construction of a cost-sensitive tree while still keeping all the functionalities of C4.5. The main improvements of C5.0 affect efficiency, otherwise the algorithm remains the same as C4.5 [28]. The C5.0 algorithm builds a decision tree by dividing the input spaces into local regions in a sequence of recursive splits. A split is pure if for all branches, for all exemplars choosing a branch belong to the same class after the split. It begins by calculating the entropy of input data (S); in Eq. 3.1. n represents the number of classes in the data and p_i corresponds to the proportion of exemplars that belong to class i.

$$E(S) = \sum_{i=1}^{n} -p_i log_2 p_i \tag{3.1}$$

If the split is not pure, then the exemplars are divided to minimize impurity and there can be multiple possible attributes on which a split can be done. The next step is to reduce the entropy by calculating the information gain for each attribute, A, in the input data, S, as in Eq. 3.2. *E(S)* is the entropy of all the input, and Sv is the number of instances which have value v for attribute A.

$$G(S, A) = E(S) - \sum_{v \in (A)} \frac{\mathsf{Sv}}{\mathsf{S}} E(Sv) \tag{3.2}$$

In other words, when a tree is constructed, at each step the split that results in the largest decrease in impurity is chosen. A more detailed explanation of the decision tree can be found in [81].

3.4.2 AdaBoost

The AdaBoost algorithm introduced by Freund and Schapire [82] in 1995 is a representative meta-learning algorithm of the Boosting family. The principle of AdaBoost is to convert or to boost weak learning (simple) classifiers into strong learning classifiers. Given a training data set, AdaBoost builds a complex classifier incrementally (Eq. 3.3).

$$H(X) = sign(\sum_{t=1}^{T}) \propto_t h_t(X) \tag{3.3}$$

The classifier *H(X)* is created by overlapping the performance of many weak simple classifiers, called decision stumps, for T times using a voting scheme. The decision stumps examine the feature set and return a decision tree with two leaves. Thus, the algorithm generates a set of hypotheses h_t such that each decision stump will return either +1 or –1. At the end of learning, *H(X)* represents a combination of these hypotheses with weights based on their credential factor $\propto_t$.

It constructs a complex classifier incrementally by overlapping the performance of possibly hundreds of simple classifiers using a voting scheme. These simple classifiers are called decision stumps. They examine the feature set and return a decision tree with two leaves. The leaves of the tree are used for binary classification and the root node evaluates the value of only one feature. Thus, each decision stump will return either +1 if the object is in-class, or –1 if it is out-class. Further information on the AdaBoost algorithm can be found in [81].

3.4.3 Team-based Genetic Programming

In this thesis, the form of genetic programming employed is based on the Symbiotic Bid-Based (SBB) paradigm of team-based GP. The SBB framework makes extensive use of coevolution [83, 84], with a total of three populations involved: a population of points, a population of learners, and a population of teams. Individuals comprising a team are specified by the team population, thus establishing a symbiotic relationship with the learner population. Only the subset of individuals indexed by an individual in the team population compete to bid against each other on training exemplars. The use of a symbiotic relation between teams and learners makes the credit assignment process more transparent than in the case of a population-wide competition between bids. Thus, variation operators may now be defined for the independent investigation of team composition (team population) and bidding strategy (learner population). The third population provides the mechanism for scaling evolution to large data sets.

In particular, the interaction between team and point population is formulated in terms of a competitive coevolutionary relation [85]. As such, the point population indexes a subset of the training data set under an active learning model (i.e. the subset indexed varies as classifier performance improves). Biases are enforced to ensure equal sampling of each class irrespective of their original exemplar class distribution [86], whereas the concept of Pareto competitive coevolution is used to retain points of most relevance to the competitive coevolution of teams.

The SBB model of evolution generates P_{gap} new exemplar indices in the point population and M_{gap} new teams in the team population at each generation. Individuals in the point population take the form of indices to the training data and are generated stochastically (subject to the aforementioned class balancing heuristic). New teams are created through variation operators applied to the current team population. Fitness evaluation evaluates all teams against all points with $(P_{size} - P_{gap})$ points and $(M_{size} - M_{gap})$ teams appearing in the next generation. Pareto competitive coevolution ranks the performance of teams in terms of a vector of outcomes, thus the Pareto non-dominated teams are ranked the highest [85]. Similarly, the points supporting the identification of non-dominated individuals (distinctions) are retained as well. In addition, use is made of competitive fitness sharing [87] in order to bias survival in favor of teams, which exhibit uniqueness in the non-dominated set (Pareto front).

Denoting the non-dominated and dominated points as $F(P)$ and $D(P)$, respectively, the SBB framework notes that as long as $F(P)$ contains fewer than $(P_{size}-P_{gap})$ points, all the points from $F(P)$ are copied into the next generation [83, 84]. An analogous process is repeated for the case of team selection, with $(M_{size} - M_{gap})$ individuals copied into the next generation. Under the condition where the (team) non-dominated set exceeds this number, the fitness sharing ranking employs $F(M)$ and $D(M)$ in place of $F(P)$ and $D(P)$, respectively. The resulting process of fitness sharing under a Pareto model of competitive coevolution has been shown to be effective at promoting solutions in which multiple models cooperate to decompose the original C class problem into a set of non-overlapping behaviours [83, 84].

Finally, the learner population of individuals expressing specific bidding strategies employs a linear representation. Bid values are standardized to the unit interval through the use of a sigmoid function, or $bid(y) = (1 + \exp(-y))^{-1}$, where y is the real valued result of program execution on the current exemplar. Variation operators take the form of the instructions add, delete, swap and mutate, applied with independent likelihood, under a uniform probability of selection. When an individual is no longer indexed by the team population it becomes extinct and is deleted from the learner population. Conversely, during evaluation of the team population, exactly M_{gap} children are created pairwise care of team-based crossover. Learners that are common to both child teams are considered to be the candidates for retention. Learners not common to the child teams are subject to stochastic deletion or modification, with corresponding tests for deletion/insertion at the learner population. The instruction set follows from that assumed in [83, 84] and consists of eight opcodes ($\{cos, exp, log, +, \times, -, \div, \%^1\}$) operating on up to eight registers, as per a linear GP representation. A more detailed description of the SBB-based GP learning model can be found in [83, 84].

3.5 Evaluation of Machine Learning Algorithms

In traffic classification, typically two metrics are used in order to quantify the performance of the classifier: Detection Rate (DR) and False Positive Rate (FPR). In this case, DR reflects the number of in-class (the ones which the algorithm aims to identify) packets/flows classified correctly and is calculated using Eq. 3.4; whereas FPR will reflect the number of out-class packets/flows classified incorrectly as in-class and is calculated using Eq. 3.5. Naturally, a high DR and a low FPR are the most desirable outcomes. False Negative, FN, implies that in-class traffic is classified as out-class traffic, whereas False Positive, FP, implies that out-class traffic is classified as in-class traffic. Furthermore, True Positive, TP, implies that in-class traffic is classified as in-class traffic, whereas True Negative, TN, implies that out-class traffic is classified as out-class traffic.

$$DR = \frac{TP}{TP + FN} \tag{3.4}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.5}$$

[1]% is a conditional operator that changes the sign of the opcode

All three candidate classifiers are trained on the training data using fifty runs to generate fifty different models for each run. Weka [88] is employed with default parameters to run AdaBoost. The Linux implementation in [80] is used with default parameters to run C5.0. Moreover, fifty runs of the C5.0 algorithm are performed using different confidence factors to generate different models for C5.0 and fifty runs of the AdaBoost algorithm are performed using different weight thresholds to generate different models for AdaBoost. Fifty runs of the GP algorithm are performed using different population initializations to generate different models. The default parameters of the C5.0, AdaBoost and GP classifiers are summarized in Tables 3.3, 3.4, and 3.5, respectively. The reason why all three candidate classifiers are trained on the training data using fifty runs to generate 50 different models for each run is to ensure that the results are based on statistically significant experiments as opposed to one-off trials. Furthermore, the non-dominated solutions were selected out of the fifty models. The non-dominated solutions are the distinctive solutions that ranked the highest in terms of high DR and low FPR out of all solutions/models. Then, the best learner out of the non-dominated learners is chosen based on the highest performance (the highest DR and the lowest FPR).

Table 3.3: C5.0 Parameterization

	Description	**Value**
r	Use rule-based classifiers	True
b	Use boosting	False
p	Use soft thresholds	True
e	Focus on errors	True
s	Find subset tests for discrete attributes	False
c	Confidence factor for pruning	5-54

Table 3.4: Weka Parameterization for AdaBoost

	Description	**Value**
classifier	The base classifier to be used	DecisionStump
numIterations	Number of iterations	10
seed	The random seed number	1
useResampling	Use resampling instead of reweighting	False
weightThreshold	Weight threshold (default 100)	10-250

Table 3.5: SBB-GP Parameterization

	Description	**Value**
P_{size}	Point population size	90
M_{size}	Team population size	90
t_{max}	Number of generations	30000
p_d	Probability of learner deletion	0.1
p_a	Probability of learner addition	0.2
μ_a	Probability of learner mutation	0.1
ω	Maximum team size	30
P_{gap}	Point generation gap	30
M_{gap}	Team generation gap	60

3.6 Traces Deployed

To show the effectiveness of the proposed approach, completely different data sets are employed for training and testing the classifiers. Moreover, solution robustness is assessed by training on a data set from one location (hereafter denoted Univ2007 trace) but by testing on data sets from other locations (Univ2007 Test partition, Univ2010, ITALY, NIMSII, NIMSIII and IPv6 traces, which were captured between 2000 and 2010).

3.6.1 Dalhousie Traces

Two Dalhousie traces were captured on the Dalhousie University Campus network by the Information Technology Services Centre (ITS) in January 2007 and May 2010 (Univ2007 and Univ2010). Dalhousie is one of the biggest universities in the Atlantic region of Canada. There are more than 15000 students and 3300 faculty and staff. The ITS is responsible for all the networking on the campus which includes more than 250 servers and 5000 computers. The Dalhousie network is connected to the Internet via a full-duplex T1 fiber link. Given privacy issues, data is filtered to scramble the IP addresses and each packet is truncated further to the end of the IP header so that all payload is excluded. Moreover, the checksums are set to zero since they could leak information from short packets. However, any information regarding the size of the packet is left intact. Moreover, both the Univ2007 and Univ2010 traces are labelled by ITS using a commercial classification tool called PacketShaper, which is a deep packet analyzer [89].

3.6.2 ITALY Traces

The ITALY data set consists of 96 hours of Skype Traffic over TCP and UDP protocols [90]. The data set was captured on the main link at the Politecnico di Torino University campus. TCP Statistic and Analysis Tool (Tstat) and the traffic classification method employed in [48] were used to label the traffic. As described in Chapter 2, section 2.2.1, the creators of this data set classified Skype traffic based on deep packet inspection and per-host analysis. In this thesis, all the Skype traces, which were captured in the Politecnico di Torino main link were employed: (i) End-to-End (E2E) voice only and voice-video class (Skype UDPE2E); (ii) Skype out calls (Skype UDPE2O); (iii) Signaling connections only (Skype UDPSIG); and (iv) End-to-End and Skype out calls (Skype TCPE2X). The first trace, which was captured over UDP, consists of voice only calls as well as voice plus video calls. The fourth trace was captured over TCP and consists of voice only calls.

3.6.3 NIMSII Traces

VoIP traffic was generated using different applications on a testbed set up in the NIMS Lab in 2009 at Dalhousie University. This testbed involved several PCs connected through the Internet and several network scenarios were emulated using many popular VoIP applications (e.g. Gtalk, Primus, Yahoo messenger). The focus was on Gtalk traffic and how Gtalk reacts to different network restrictions was observed. Moreover, the effects (if any) of different types of access technologies (i.e. WiFi versus Ethernet) were investigated as well as their different combinations. In 2009 over two hundred experiments were conducted, equivalent to more than fifty hours of VoIP traffic and non-VoIP traffic. This data set was made public at [91].

For this work, a Gtalk client was installed on each of the three Windows XP machines. The first machine was a Pentium 4 2.4 GHz Core 2 Duo with 2 GB RAM; the second machine was a Pentium 4 2 MHz Core 2 Duo with 2 GB RAM, and the third machine was a MacBook 2 GHz Intel Core 2 Duo with 2 GB RAM. Two machines had a 10/100 Mv/s Ethernet connection and the third machine had a wireless 10/100 Mv/s card. Furthermore, one was connected to a 1 GB/s network while the others were connected to a 10/100 Mb/s network. All three machines had Windows XP Service Packet 2 and all experiments were done using the Gtalk client version 1.0.0.104. In all experiments, Gtalk traffic was captured from both ends. In all cases the experiments were performed under several different network scenarios (Figures 3.1 and 3.2).

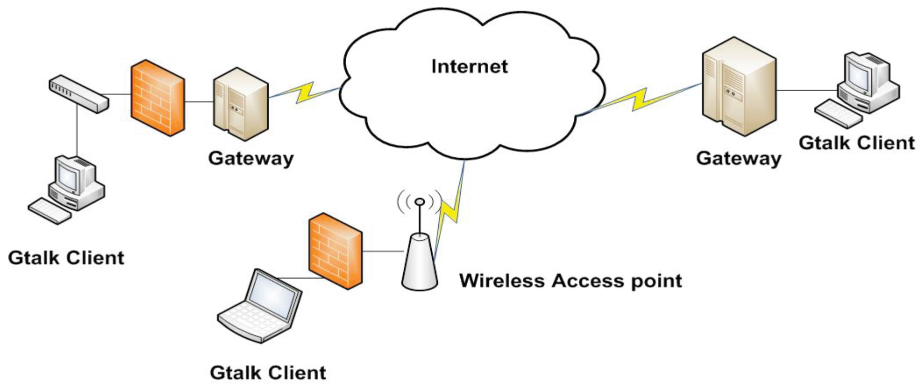


Figure 3.1: Network Setup with Restrictions

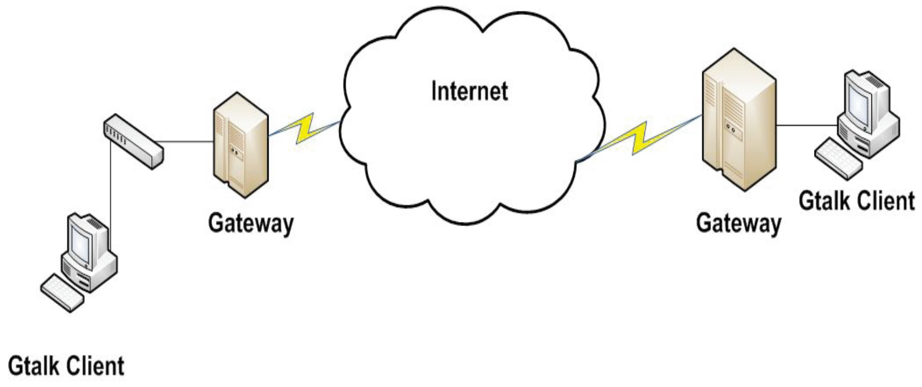


Figure 3.2: Network Setup without Restrictions

These scenarios include: i) Firewall restrictions at one user end and no restrictions at the other end; ii) Firewall restrictions at both ends; iii) No restrictions at both ends; iv) Use of wireless and wire-line connections; v) Blocking of all UDP connections, and vi) Blocking of all TCP connections. It should be noted here that during these experiments all the Internet communications went through the network's firewall. The firewall was configured to either block or to permit access to the following restrictions: i) block everything, or ii) permit limited well-known port numbers: 22, 53, 80 and 443. Wireshark [62] and NetPeeker [92] were used to monitor and control network traffic. NetPeeker was used to block ports and to allow either both TCP and UDP traffic, or only UDP or TCP traffic in order to analyze the behaviour of the Gtalk client. On the other hand, Wireshark was used to capture traffic from both ends of the communication.

The general call setup between the caller and callee for voice calls was as follows: caller transmits a standard audio file to callee. An English spoken text (male and female audio files) without noise and a sample rate of 8Hz was used, which encoded with 16 bits per sample and which can be downloaded at [93]. The wav-file was played and then the output of the Windows media player was used as input for Gtalk, Primus (soft Talk Broadband (softTBB)) and Yahoo messenger (Encrypted with Zfone in 2009) with clients using a microphone. Wireshark was used to capture the traffic from both users' ends.

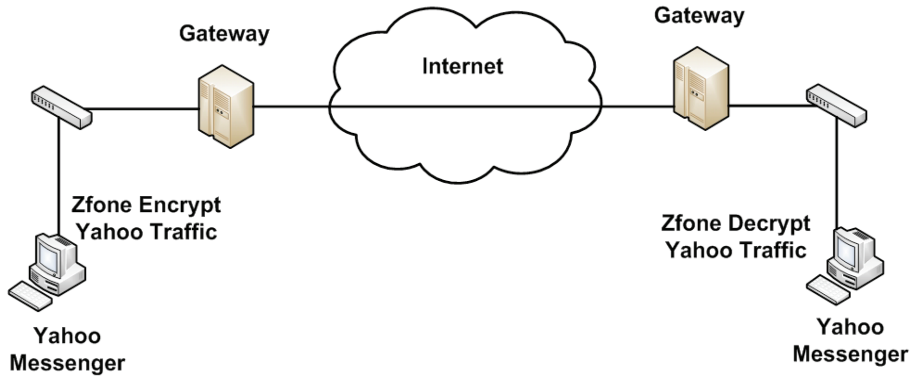


Figure 3.3: Network Setup for Zfone Calls

Furthermore, Yahoo messenger traffic was generated as well (encrypted with Zfone in 2009 only). Zfone traffic is another encrypted VoIP traffic, which was generated. Zfone [94] is software which secures VoIP calls over the Internet. Zfone works by intercepting all the unencrypted VoIP channel and protecting the VoIP channel securely by encrypting all the VoIP packets. Zfone is the user interface of the Zimmermann Real-time Transport Protocol (ZRTP) [95]. ZRTP uses Diffie-Hellman to exchange keys over the Real-time Transport Protocol (RTP) packet stream (creating secure RTP sessions). It encrypts the payload of a packet using standard cryptographic algorithms such as the Advanced Encryption Standard (AES) or Rivest Shamir Adleman (RSA) algorithms. Zfone was used to secure all Yahoo Messenger audio calls. Zfone detects Yahoo packets and encrypts them as they are sent by the caller machine and detects the encrypted packets received by the callee machine and decrypts them (Figure 3.3).

Non-encrypted VoIP traffic was generated using a Primus Session Initiation Protocol (SIP) client [5]. Primus softTBB was used to make calls to a Public Switched Telephone Network (PSTN) for voice services (hard line phone) and mobil cell phone. The softTBB client runs on a PC or a laptop and connects to the Primus SIP Network over the Internet. Depending on what is called, i.e. a mobile phone or a PSTN phone, the Primus SIP network routes the calls to the final destination differently.

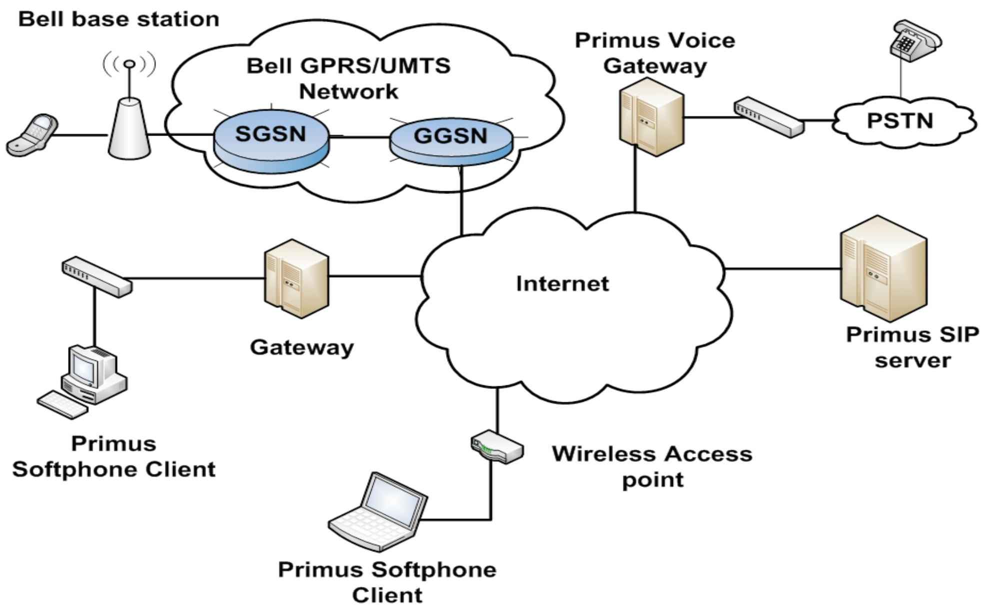


Figure 3.4: Network Setup for Primus Calls

A call to a PSTN phone is routed to the nearby Primus Voice gateway which converts the calls between the VoIP network and the PSTN network (Figure 3.4). For a call to a cell phone which is subscribed to the Bell General Packet Radio Service (GPRS) and Universal Mobile Telecommunication System (UMTS) network, the route is more complex. According to the GPRS/UMTS specification [96], the main components of the GPRS/UMTS network are base stations and gateways connected to the Internet. In this case, firstly, the cell phone registers with the base station. Then, the base station is connected to the Serving Gateway Support Node (SGSN), which is connected to the Gateway GPRS Support Node (GGSN) inside the Bell GPRA/UMTS network. Finally, the GGSN is the first node responsible for processing IP packets from the Internet to the mobile network and vice versa. To establish the call between a Primus softTBB and a Bell mobile phone, the call is routed through the Primus SIP network through the Internet to the Bell GGSN gateway. In all cases it was possible to listen to the call on the PSTN phone and the mobile phone. All communications are done without encryption and the traffic is captured using Wireshark only at the machine where softTBB is running, since permission to capture traffic with Primus or Bell companies were not an option. In this case, it was a deliberate choice not to encrypt the traffic in order to have different mixtures of VoIP traffic in the traces, i.e. both encrypted (Gtalk, Skype, and Yahoo with Zfone) and non-encrypted (Primus).

3.6.4 NIMSIII Traces

Further VoIP traffic was generated in 2010 using the same testbed used for the NIMSII traces. For the NIMSIII traces Gtalk and Primus traffic was generated using the same setup as in the NIMSII traces (section 3.6.3) and other background traffic, such as torrent traffic and web TV and radio, were included. However in 2010, Yahoo began to use encryption to secure its traffic. Therefore Zfone was not needed for encrypting the Yahoo traffic since it can be used only with a non-encrypted payload. Online web radio media stream (non-encrypted) traffic was captured as well. Also, the same methodology was used to capture a TV media stream channel broadcast on the web. Torrent traffic was captured as well by installing a Torrent client to download a free Linux operating system distribution. Several hours of traffic was captured for each application.

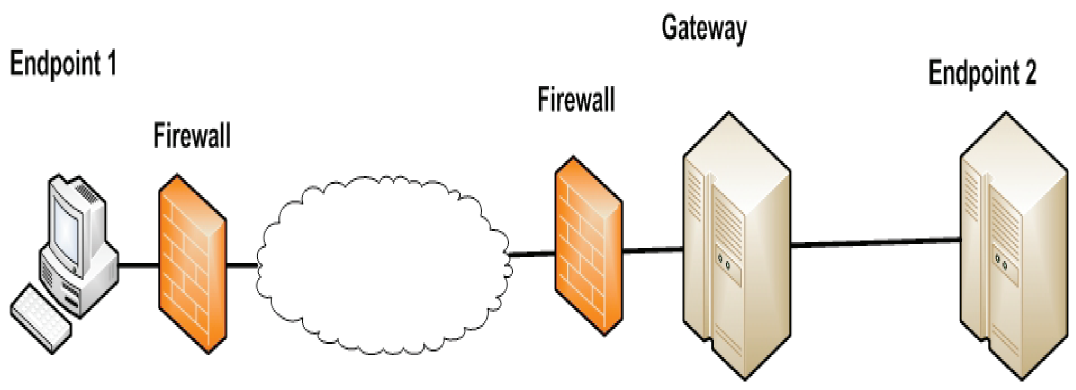


Figure 3.5: VPN Tunnel Setup

Due to wide usage of the Virtual Private Network (VPN) technology by companies and users who want to secure their communications, VPN traffic was generated as well and included in the NIMSIII test data sets. VPN technology can tunnel any traffic and quarantine the privacy and security between the two end points. Two VPN technologies are applied regularly for establishing VPN tunnels: Layer 2 Tunneling Protocol Internet Protocol Security (L2TP/IPSec) and Secure Socket Layer Transport Layer Security (SSL/TLS). Both technologies provide encrypted and secure communications with different implementations. L2TP enables the encapsulation of Ethernet frames into UDP packets. To ensure the privacy of the packets, L2TP is combined with IPSec. IPSec comes with two configurations: the (i) Encapsulated Security Payload (ESP) and the (ii) Authentication Header (AH). The ESP protects the payload by using encryption algorithms (e.g. Blowfish) while the AH protects the IP packet header by computing a cryptographic checksum and hashing. The ESP was chosen for two reasons: (i) authentication does not quarantine encryption of the payload; and (ii) the VoIP applications employed in this thesis encrypt their payload. The second common technology for setting up a VPN tunnel is SSL/TLS, which is used typically for securing HTTP connections but which can be used as well to create a tunnel for transfering any traffic between two machines. The Mac OS X implementation of L2TP/IPSec was used to set a connection between two machines for the first VPN technology and a commercial tool called VPN-X [97], which employs the second VPN technology was used as well (Figure 3.5). For both VPN technologies a VPN tunnel was set up between two machines and their traffic was captured. Using the two VPN setups it was possible to view and share files, to browse web pages and to send chat messages. This testbed traffic has been made publicly available to the research community as well [91].

3.6.5 IPv6 Traces

IPv6 is a version of IP and is intended to succeed IPV4 since the IPv4 address space is running out of addresses. The IPv6 address space is 128 bits long and provides privacy and security for communication on the network by applying encryption. IPv6 allows varieties of encryption algorithms. Kent and Atkinson define the IPv6 architecture [98]. Since IPv6 encrypts the packet and has a large address space which will not be running out in the next few years, IPv6 traces are included as test data sets to evaluate the robustness of the signatures. Public IPv6 traces captured by the Measurement and Analysis on the WIDE Internet working group (MAWI) [99] in 2000 and 2009 were used. The IPv6 traces are captured daily at an IPv6 cable connected to the 6Bone testbed (Figure 3.6). The 6Bone [100] testbed was established by the Internet Engineering Task Force (IETF) in 1996 to test IPv6 and ease the transition of the Internet to IPv6. The IPv6 traces total more than nine hours and include several applications running over TCP and UDP protocols such as HTTP, HTTPS, SMTP, POP3, SSH and DNS as well as traffic from other protocols (e.g. ICMP).

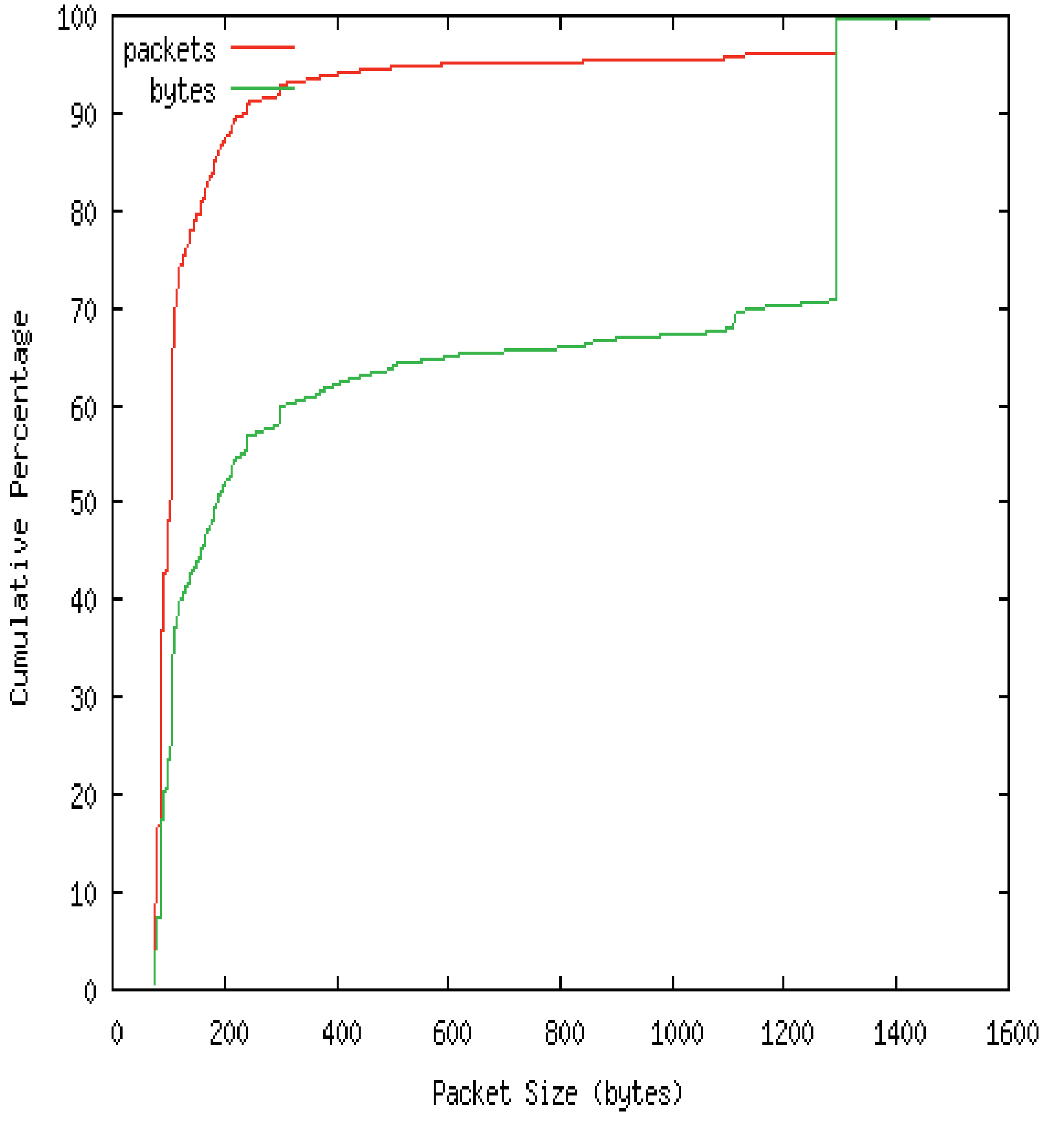


Figure 3.6: Packet Size Distribution for IPv6 Traces [101]

3.6.6 Analysis of Traces

The aim of this section is to have a better understanding of the characteristics of the data sets deployed in this thesis. The network traffic traces are examined in detail. First, the general properties of the traces are shown in terms of protocol usage and the number of packets/flows, then a more detailed analysis of the data is presented in terms of time, rate and bytes sent/received for each trace.

Brief statistics on the traffic data collected are given in Table 3.6. This shows that all the data sets have different general properties in terms of the number of packets and flows and the size of the traces.

Table 3.6: Overview (in Millions) of Network Traces Employed

	Packets	**Bytes**	**Flows**
Univ2007	336M	212,931M	28M
Univ2010	1,857M	1,347,191M	46M
Skype UDPE2E	39M	7,840M	295,811
Skype UDPE2O	3M	188M	2,601
Skype UDPSIG	71M	15,141M	50M
Skype TCPE2X	2M	302M	13,306
NIMSII: GTALK2009	34M	6,485M	190,665
NIMSII: PRIMUS2009	2M	384M	7,529
NIMSII: Zfone2009	1M	138M	28,553
NIMSIII: GTALK2010	384M	1,256M	14,847
NIMSIII: PRIMUS2010	7M	1,367M	21,802
NIMSIII: YAHOO2010	8M	1,080M	23,239
NIMSIII: Torrent2010	21M	17,791M	412,345
NIMSIII: Radio2010 stream	332,183	272M	2,236
NIMSIII: TV2010 stream	5M	4,941M	1,803
NIMSIII: VPN2010	32,079M	26,728M	74,302
IPv6	808M	825M	1,151

Also there is clear differentiation of the traces in terms of the use of the TCP/UDP protocols (Figures 3.7, 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14, 3.15 and 3.16). From these figures, it can be seen that the Univ2010 traces have a high number of packets transmitted with relatively big packet sizes whereas Univ2007 have a higher average size of packets but a lower number of packets transmitted compared to Univ2010. This shows the difference between these traces. Furthermore, the number of TCP packets/bytes have doubled in numbers from Univ2007 to Univ2010. These differences might be due to the changes of the applications' behaviour or the usage of the network by the users or both. Moreover, the Dalhousie (Univ2007 and Univ2010) traces have more TCP protocol usage than UDP protocol usage whereas the ITALY traces consist of mostly UDP usage. This might be due to the fact that the ITALY traces contain only Skype traffic. It should be noted here based on the analysis of the ITALY traces that Skype prefers the usage of UDP to TCP. The same trend for the VoIP applications protocol preferences can be seen as well in the NIMS traces where the UDP packets/bytes are more numerous than the TCP packets/bytes. Furthermore, most of the packets transmitted on the ITALY and NIMS traces are on average smaller in size compared to the University traces. This makes the ITALY and NIMS traces different from the University traces. The traces are from different locations, network infrastructures and cover different time periods ranging from 2006 to 2010 and are over different durations ranging from three hours to several days. Moreover, TCP and UDP have different trends in the traces, too.

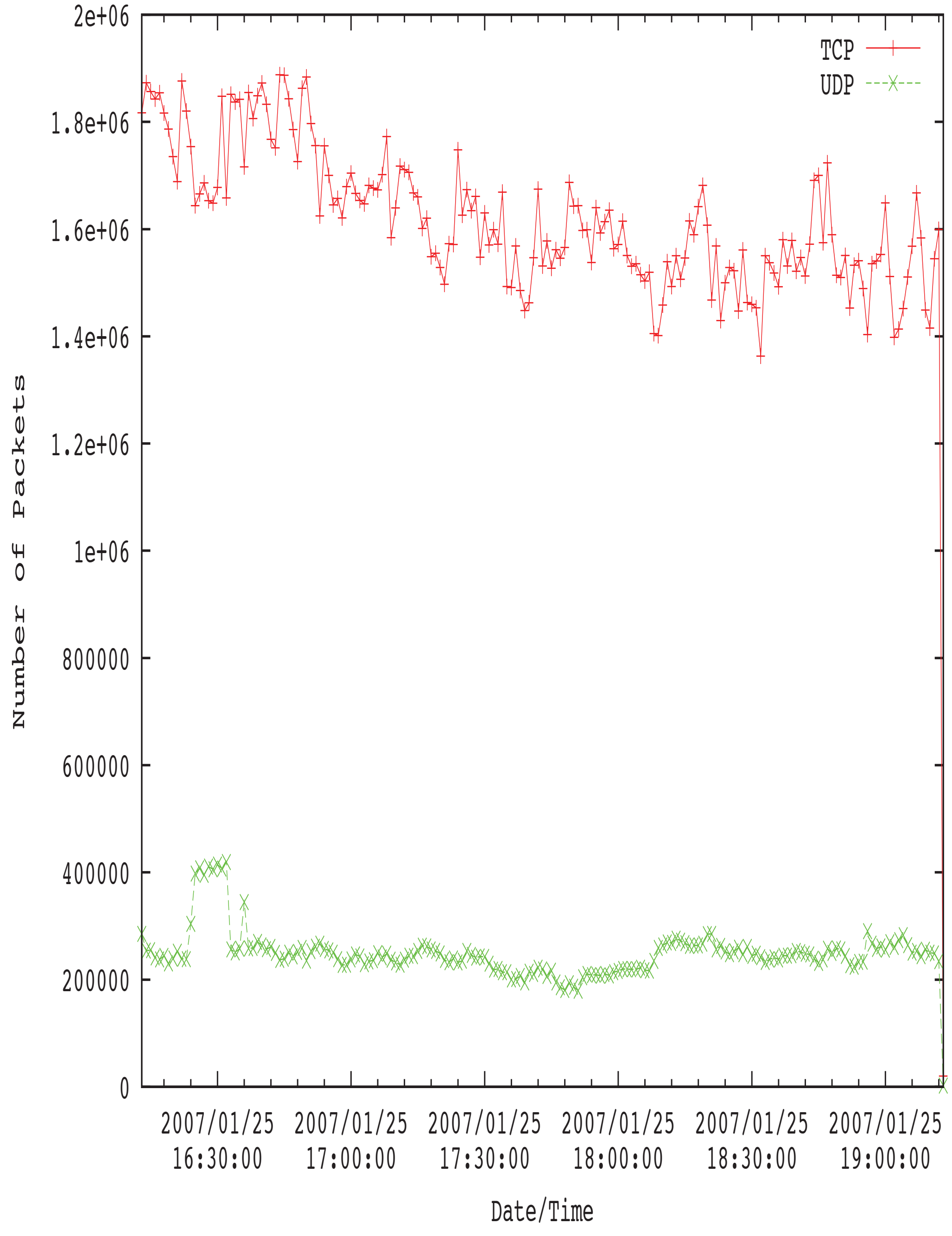


Figure 3.7: Number of Packets per Minute for TCP/UDP Protocols in Univ2007 Trace

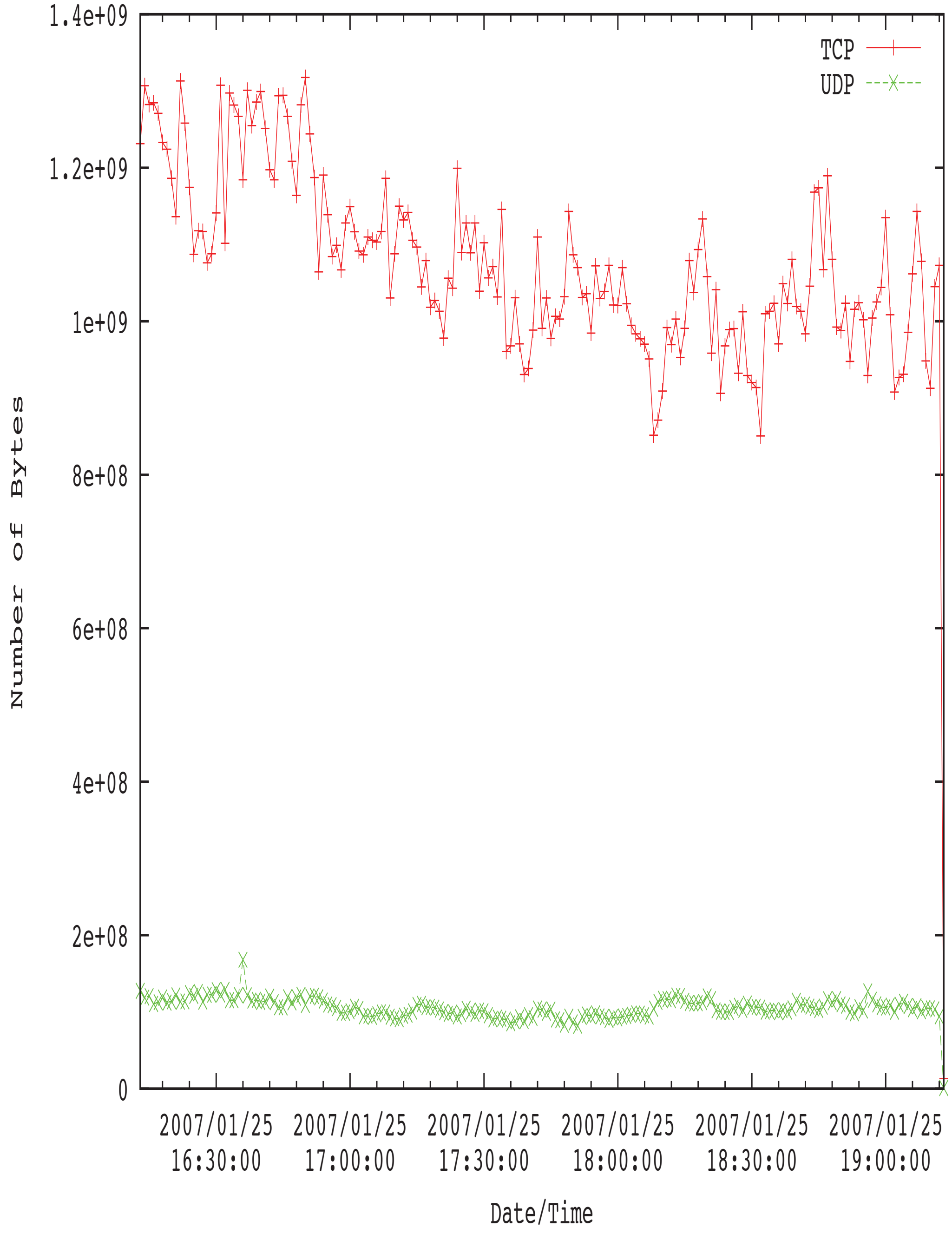


Figure 3.8: Number of Bytes per Minute for TCP/UDP Protocols in Univ2007 Trace

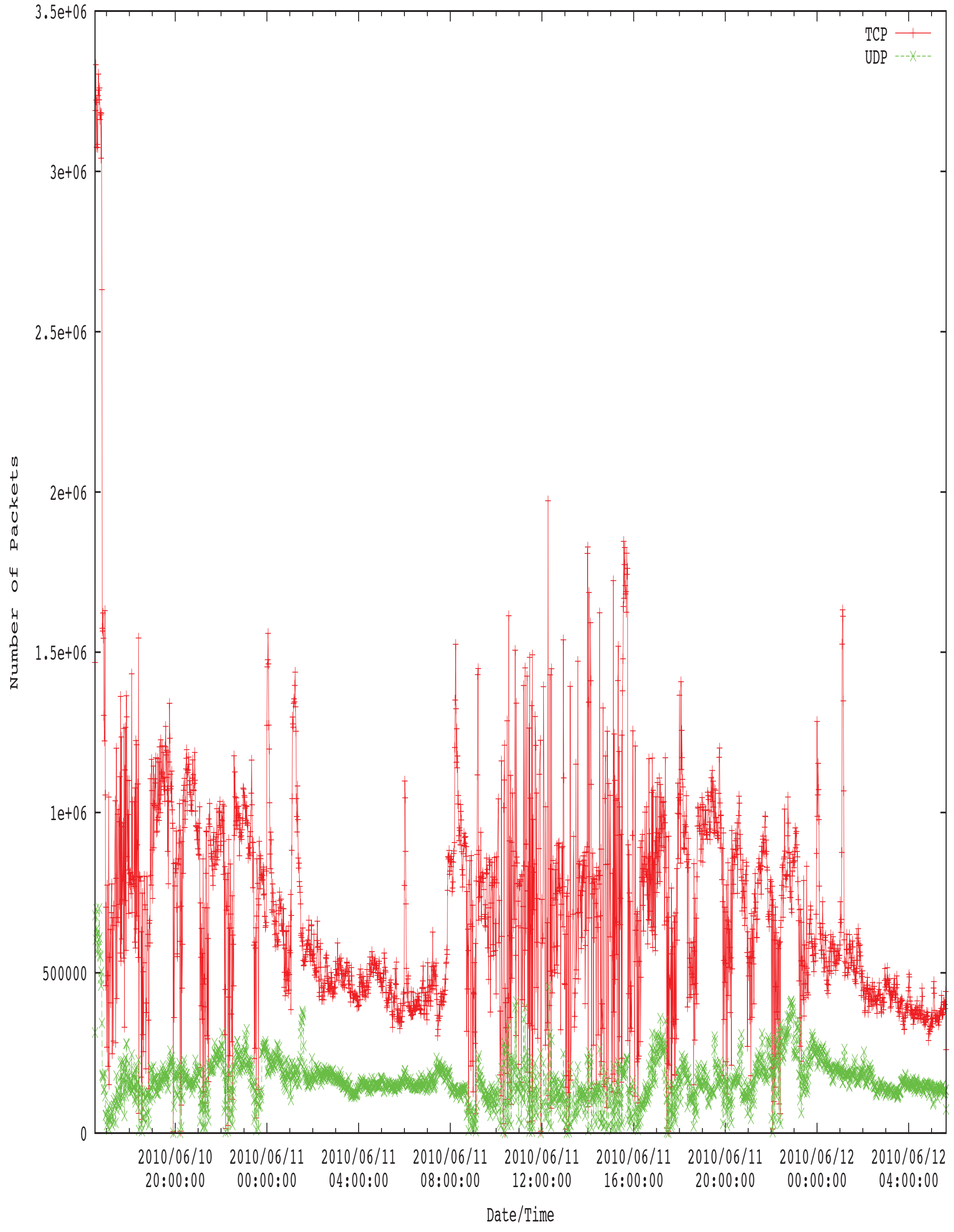


Figure 3.9: Number of Packets per Minute for TCP/UDP Protocols in Univ2010 Trace

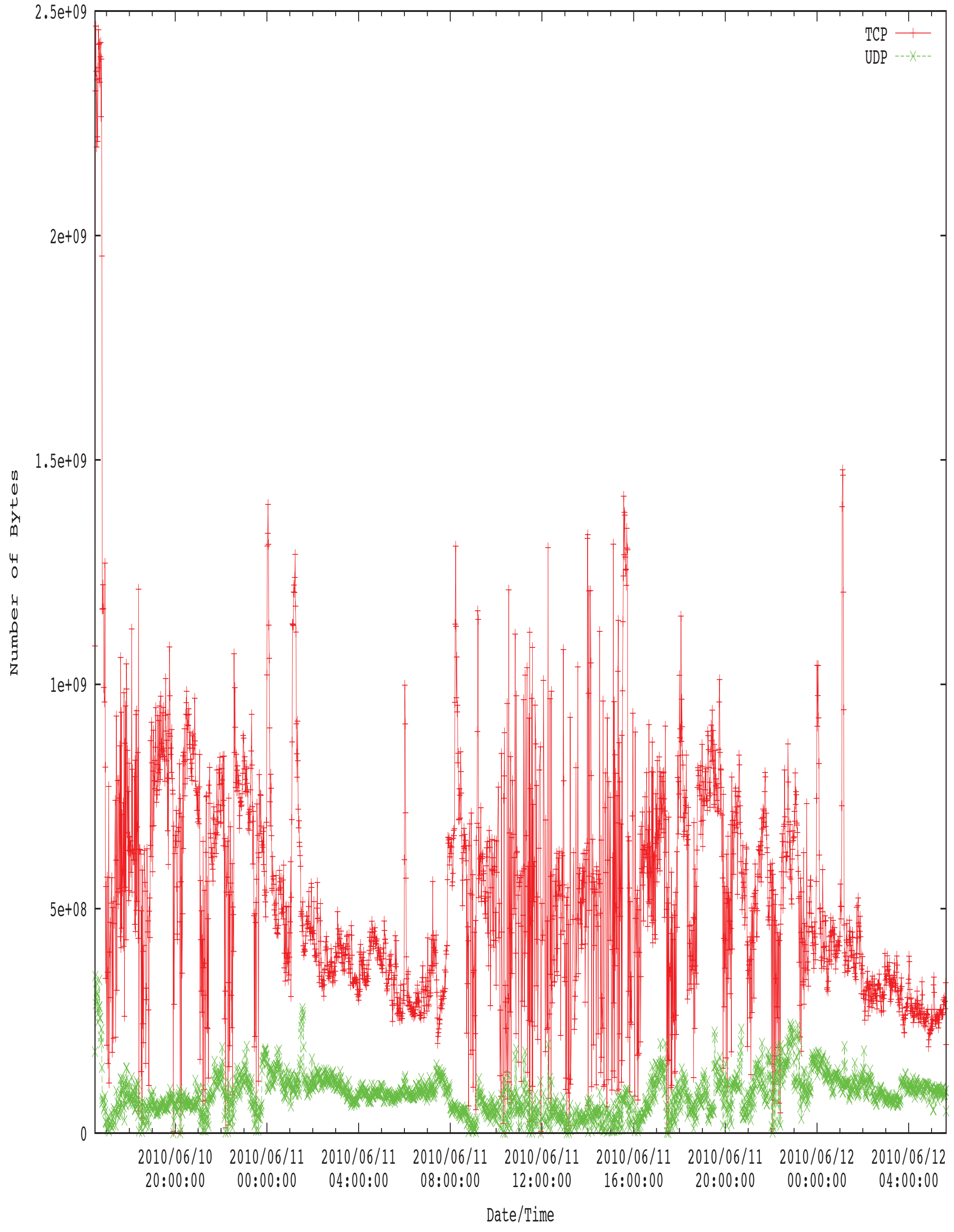


Figure 3.10: Number of Bytes per Minute for TCP/UDP Protocols in Univ2010 Trace

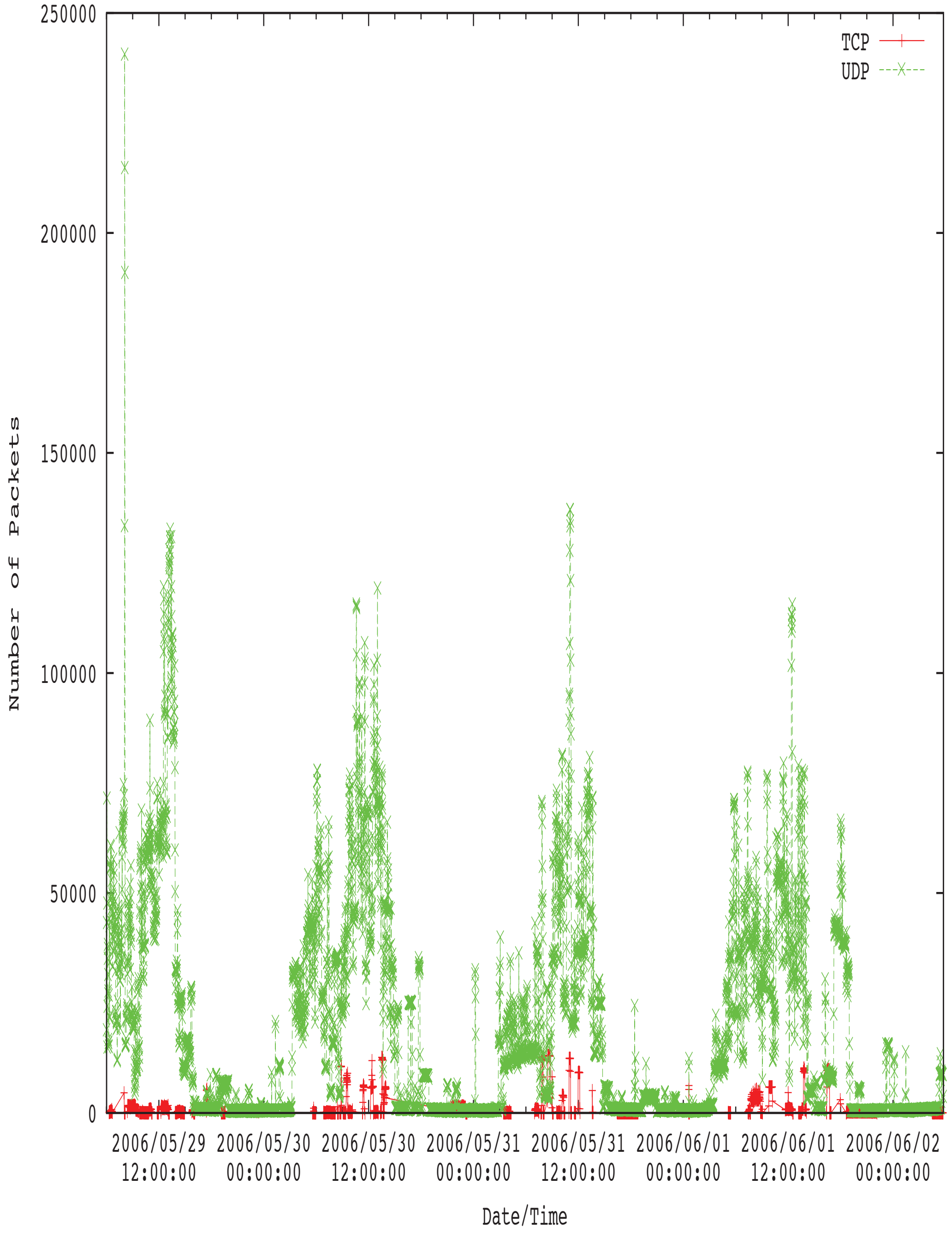


Figure 3.11: Number of Packets per Minute for TCP/UDP Protocols in ITALY Trace

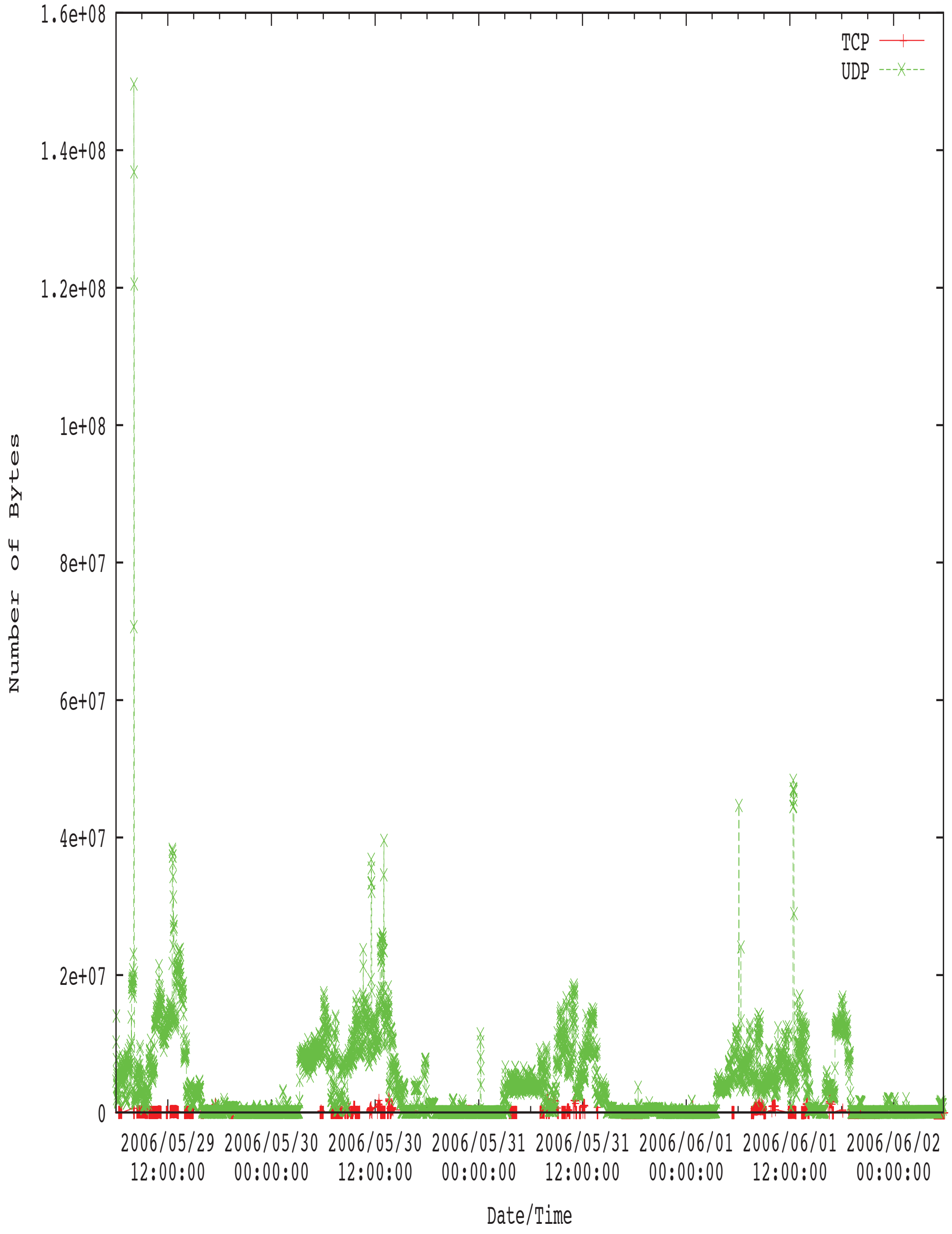


Figure 3.12: Number of Bytes per Minute for TCP/UDP Protocols in ITALY Trace

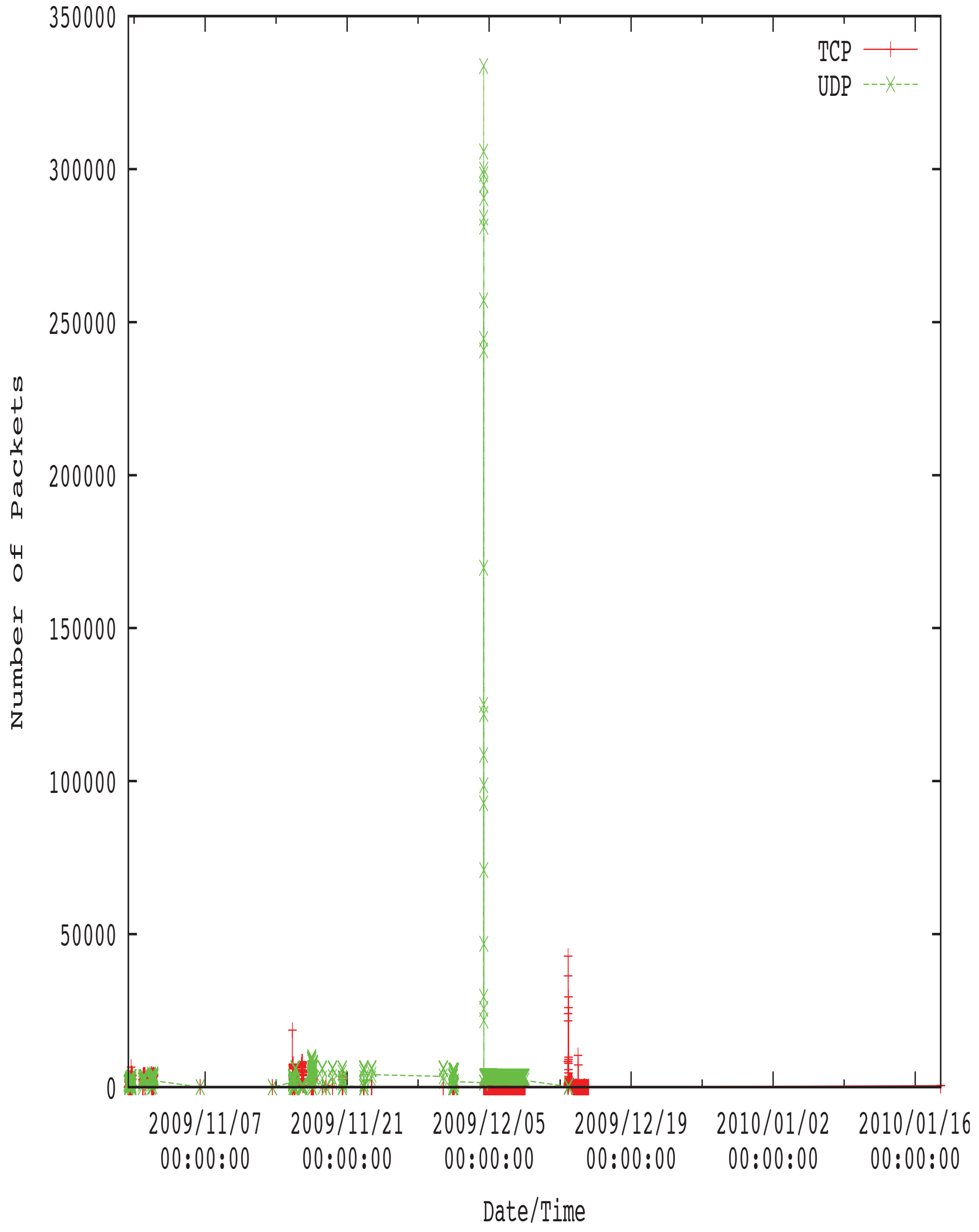


Figure 3.13: Number of Packets per Minute for TCP/UDP Protocols in NIMSII Trace

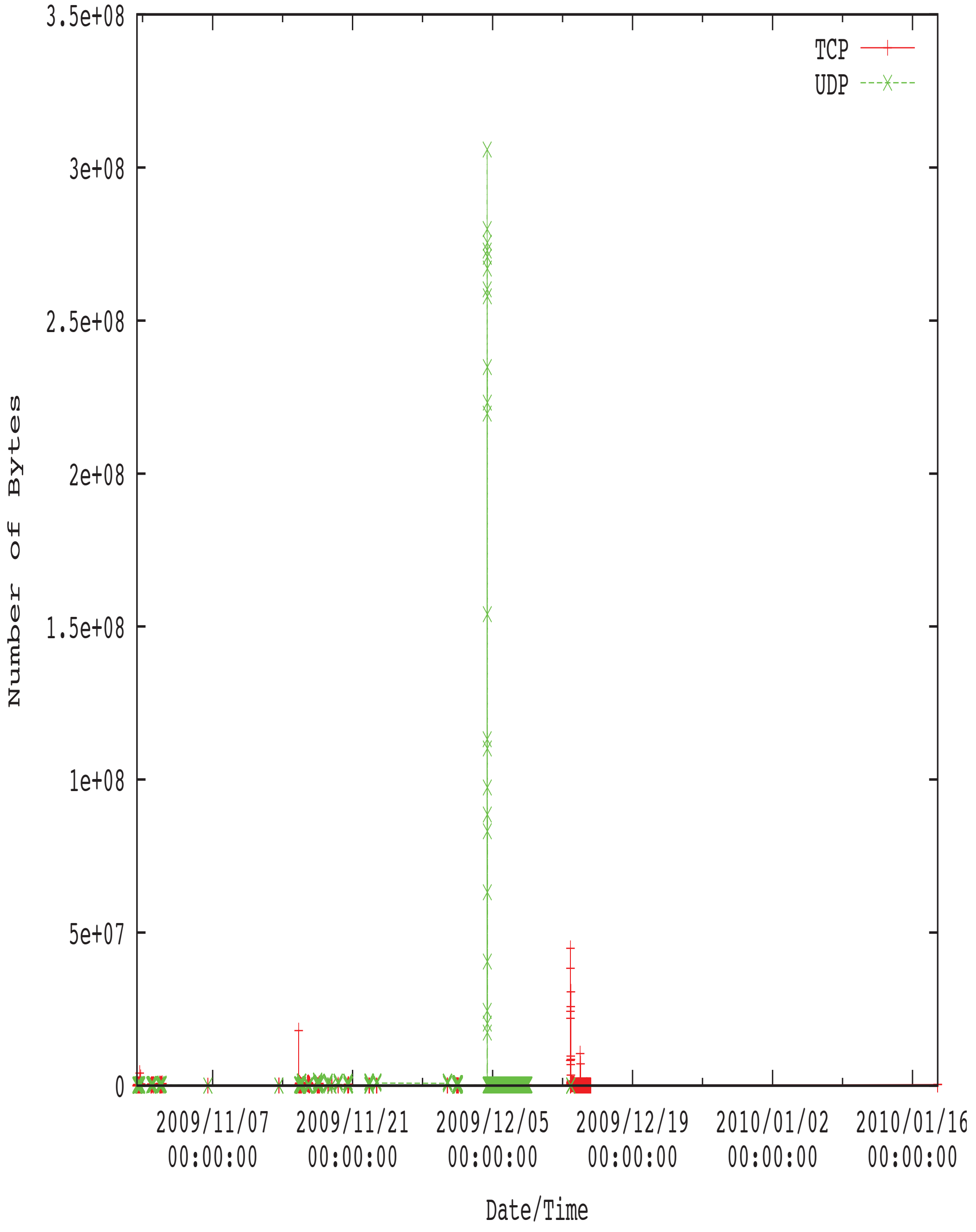


Figure 3.14: Number of Bytes per Minute for TCP/UDP Protocols in NIMSII Trace

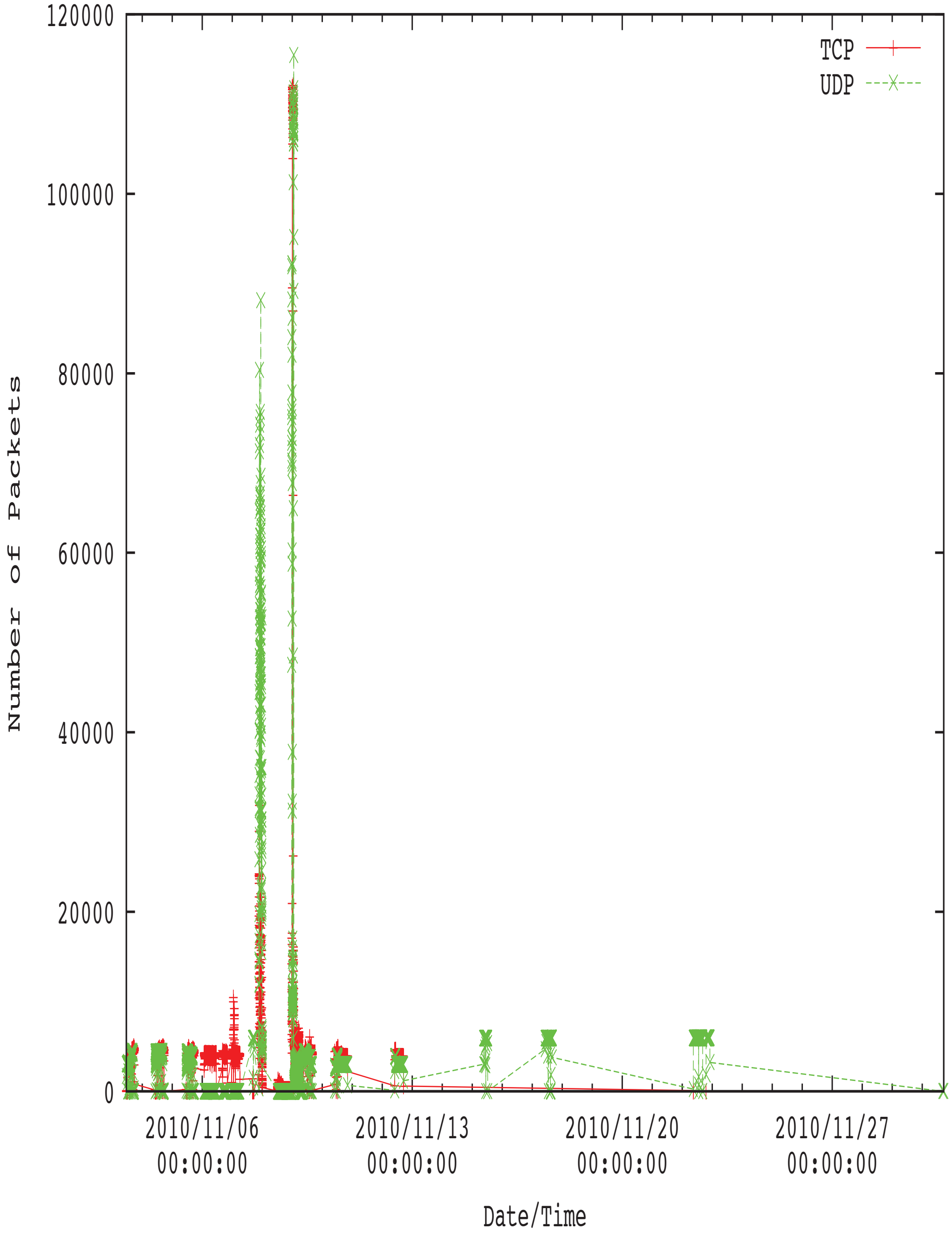


Figure 3.15: Number of Packets per Minute for TCP/UDP Protocols in NIMSIII Trace

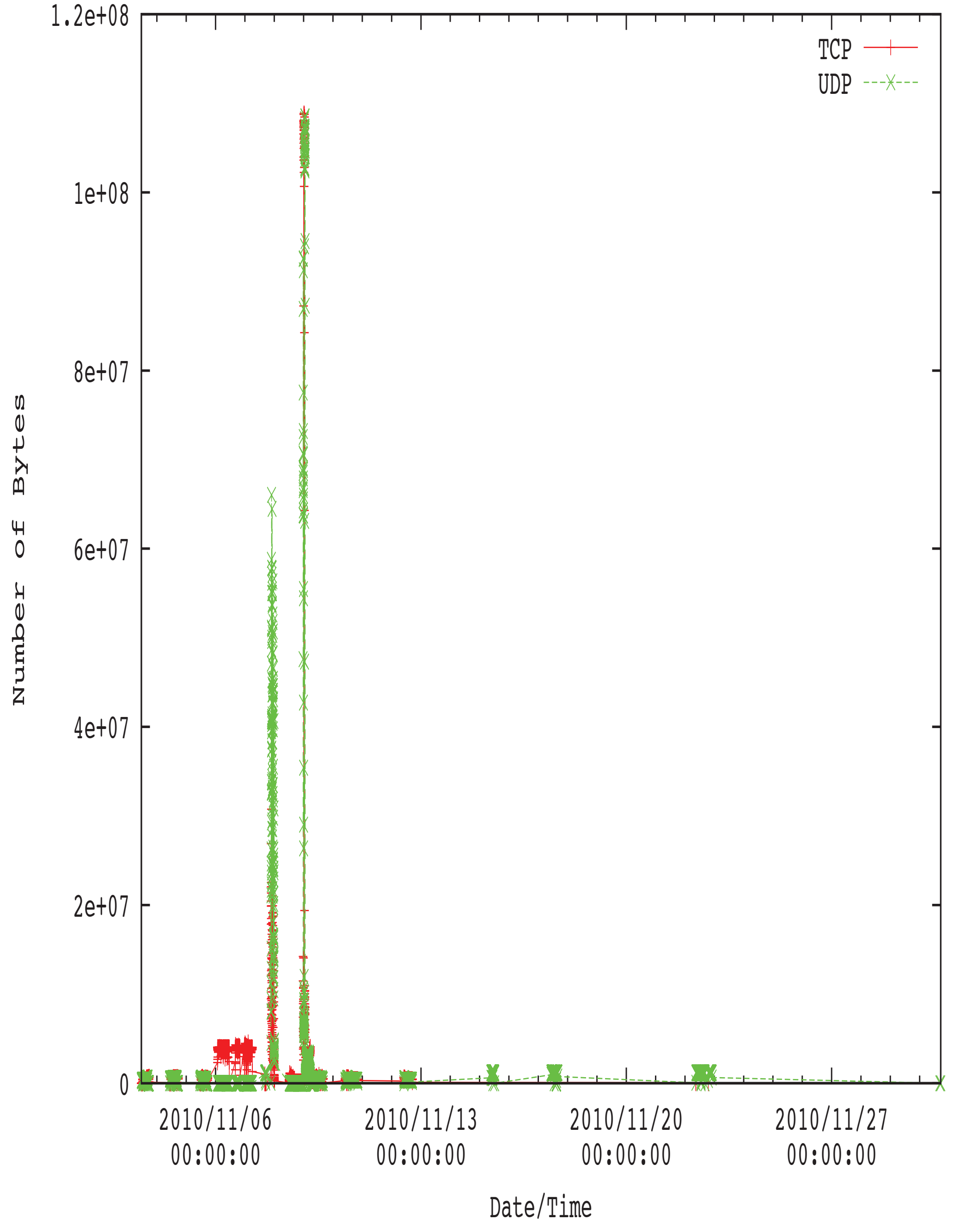


Figure 3.16: Number of Bytes per Minute for TCP/UDP Protocols in NIMSIII Trace

Looking at the traces from the perspective of flows, Figures 3.17 and 3.18 show scatter plots of the mean inter-arrival time and the standard deviation of inter-arrival time of each flow for the forward and the backward directions (client to server direction and server to client direction, respectively) while Figures 3.19 and 3.20 show scatter plots of the mean packet length and the standard deviation of packet length. In these figures, it can be seen that the Skype flows are notably diverse between the forward and backward directions for the inter-arrival time for all the data sets. This is due to the infrastructure of the network (e.g. the bandwidth, cable type etc.) and the behaviour of the Skype users in using Skype features such as text chat, voice chat, file transfers and so on. On the other hand, there is a symmetry between the forward and the backward directions for the packet length. Moreover, Skype flows overlap with non-Skype flows in the data sets plotted. This shows that Skype is mimicking the typical routines of other protocols. Hence, the problem of finding robust signatures for classifying Skype packets/flows on different data sets from different time periods and locations is a challenging task. Furthermore, the figures show that all the data sets are very good choices for testing the robustness/generalization of the signatures generated by the ML classifiers.

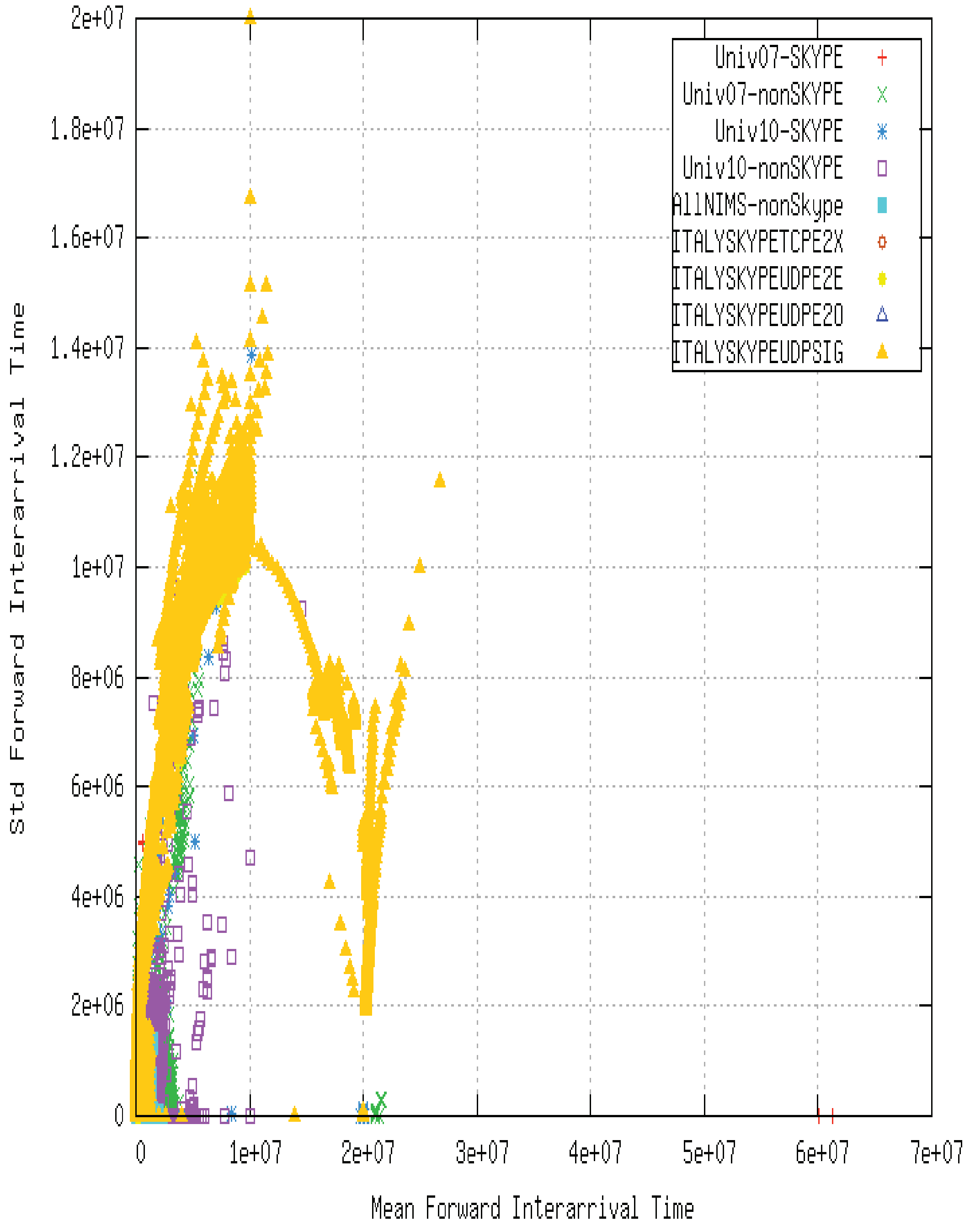


Figure 3.17: Forward Direction of Mean of Inter-rrival Time vs. Std. of Inter-arrival for University, NIMS and ITALY Traces

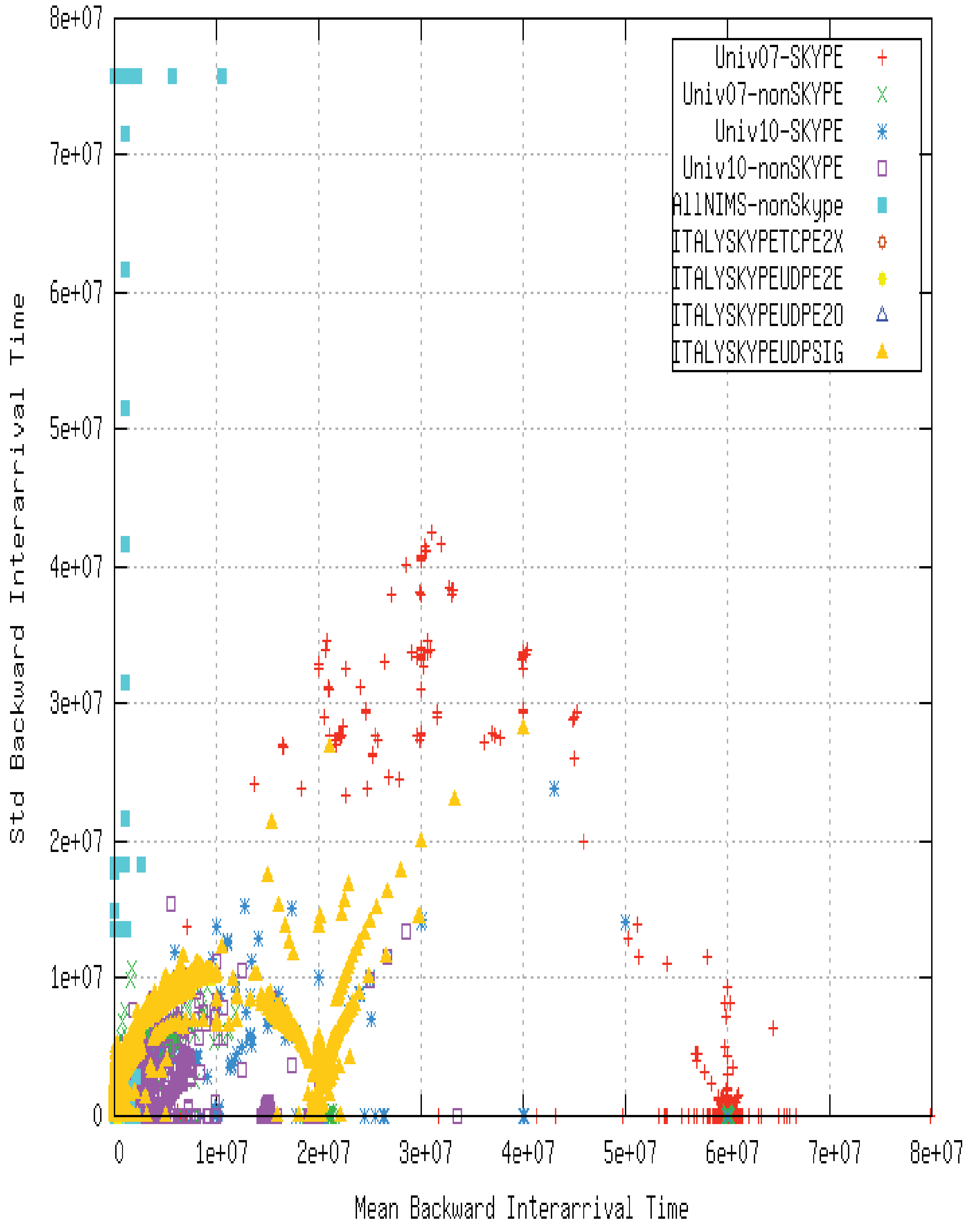


Figure 3.18: Backward Direction of Mean of Inter-arrival Time vs. Std. of Inter-arrival for University, NIMS and ITALY Traces

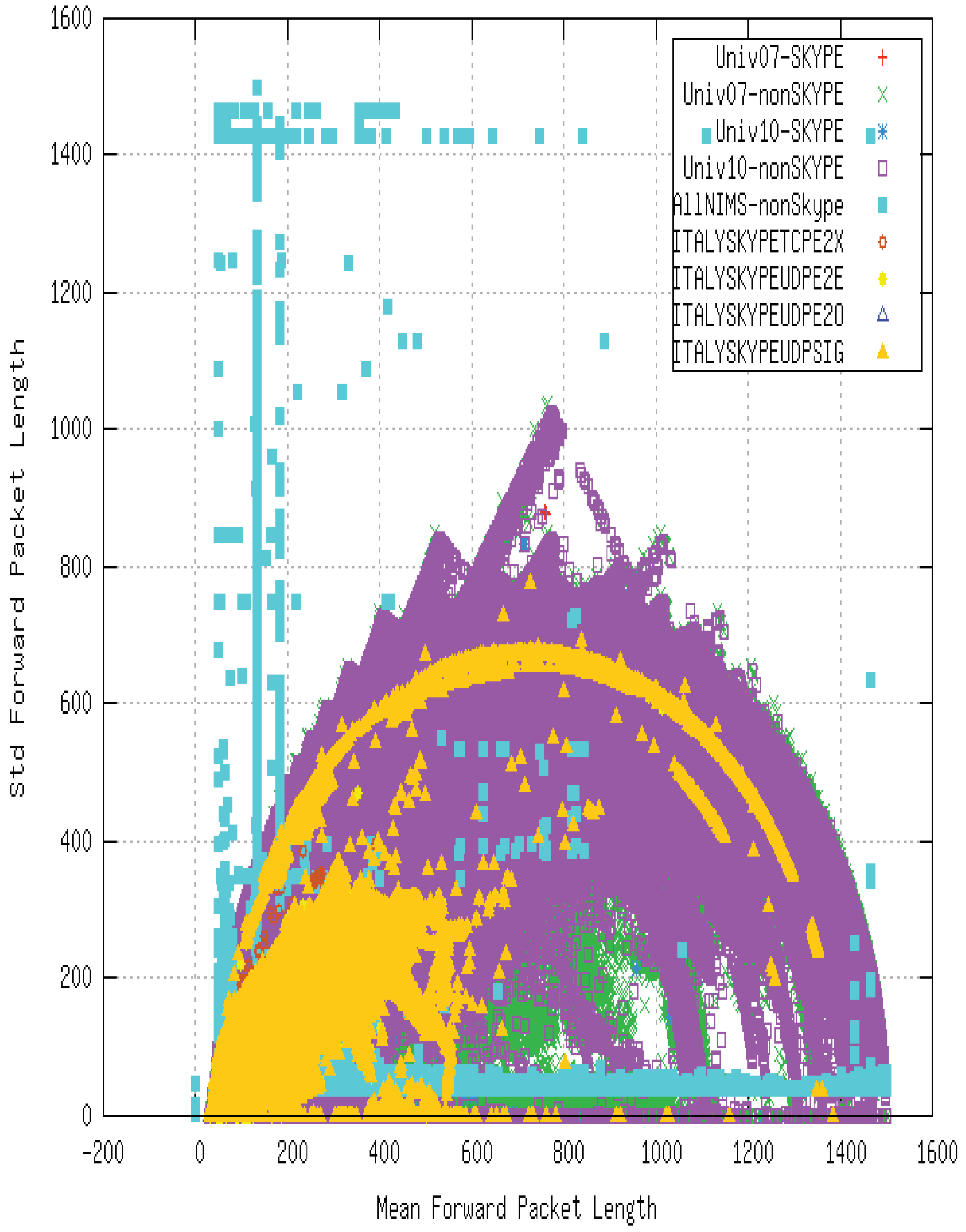


Figure 3.19: Forward Direction of Mean of Packet Length vs. Std. of Packet Length for University, NIMS and ITALY Traces

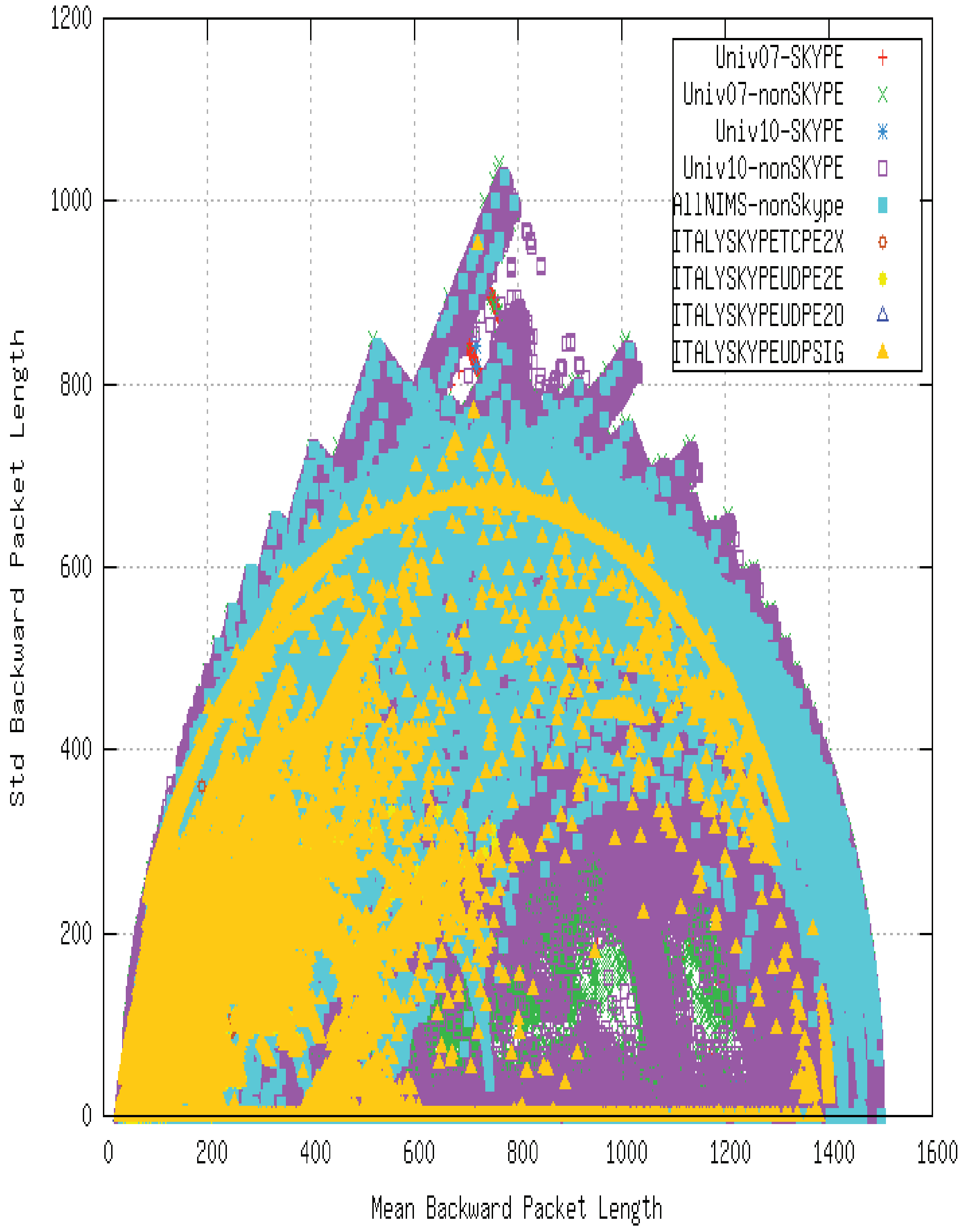


Figure 3.20: Backward Direction of Mean of Packet Length vs. Std. of Packet Length for University, NIMS and ITALY Traces

Chapter 4

Subset Sampling Methods for Selecting Training Data Sets

Weiss [29] has pointed out the importance for the subset sampling and its effects on the performance of the classifier during training. This chapter tackles the challenging problem of sampling training data sets for the machine learning algorithms where the most important part of the process is finding an informative sampled training data set for the ML algorithms. Three different techniques for sampling training data sets have been compared: (i) uniform random N sampling where N is either a fixed number of records (e.g. 30K, 60K, etc.); or N is a fixed percentage of records (e.g. 1%, 2%, etc.); (ii) stratified N sampling based on grouping where N is either a fixed number of records (e.g. 30K, 60K, etc.); or N is a fixed percentage of records (e.g. 1%, 2%, etc.) and (iii) continuous data streams of either a specific time period (e.g. 30 minutes, 60 minutes and 90 minutes of traffic) or N sampling records (e.g. 30K, 60K, etc.). All random samplings were done using uniform probability. Since the goal is to generate robust signatures for classifying unknown VoIP traffic, the training data set was limited to be a subset of Univ2007 and the remaining Univ2007 as well as Univ2010, NIMSII, NIMSIII and IPv6 and ITALY traces become the test data sets.

4.1 Uniform Random *N* Sampling Method

Random N packets are sampled with uniform probability from the Univ2007 trace where it was divided into two classes. The two classes are Skype, representing the in-class, and non-Skype, representing the out-class. The non-Skype class includes all the network applications in the traces. Since the focus is to differentiate Skype traffic from non-Skype traffic, six training data sets with different N number of packets were sampled randomly with uniform probability. For example, when N is equal to 30K, 15000 flow records from Skype and 15000 flow records from non-Skype classes were sampled randomly for a total of 30000 records. For the fixed number of records, six different N values were used: 30K, 60K, 100K, 200K, 400K and 800K. Six other different training data sets were sampled randomly with uniform probability where N represents a fixed percentage of records. Six different N values were used. These are: 1%, 2%, 3%, 4%, 5% and 6%. For instance, when N is equal to 1%, 1% of flow records from Skype and 1% of flow records from non-Skype classes were sampled randomly. Table 4.1 lists the total number of records sampled randomly for each training data set.

Table 4.1: Number of Flows Sampled for the Uniform Random N Sampling Method

N **records**	**# of Skype**	**# of non-Skype**	**Total**
30k	15,000	15,000	30,000
60k	30,000	30,000	60,000
100k	50,000	50,000	100,000
200k	100,000	100,000	200,000
400K	200,000	200,000	400,000
800k	400,000	400,000	800,000
1 Percent (1%)	82,547	207,250	289,797
2 Percent (2%)	165,095	414,501	579,596
3 Percent (3%)	247,643	621,752	869,395
4 Percent (4%)	330,191	829,002	1,159,193
5 Percent (5%)	412,739	1,036,253	1,448,992
6 Percent (6%)	495,286	1,243,504	1,738,790

4.2 Stratified Sampling Method

Stratified sampling uses a priori information to improve the estimation performance of classification methods by using grouping techniques. This section investigates whether including applications which exhibit behaviour similar to the Skype application to the training data set makes any difference in the training performance or not. The Univ2007 data sets will be grouped so that each cluster contains data with similar properties. After that the classes (network applications e.g, FTP, HTTP etc.) in each cluster are determined in order to select them for constructing the training data set. Self-Organizing Feature Maps (SOM), which are introduced by Kohonen [102], are employed to build the clusters using the Univ2007 data sets. SOM is a well-known unsupervised machine learning model based on a neural network model for clustering and visualization of high dimensional data into a topographical two-dimensional grid structure [103].

4.2.1 Self Organizing Feature Maps (SOM)

Self organizing feature maps are unsupervised neural networks which transform arbitrarily high dimensional input data space (n dimensional input data vector) into a low dimensional space which is most commonly a two dimensional array of neurons. The aim of the SOM is to discover the fundamental structure of the input data space (feature map) while maintaining the properties of the input space. SOM builds a topologically preserving map which presents a visual arrangement of the neighbouring relationships of the points in the input data set where a human can simply notice groups/clusters and relations.

The learning process of the SOM starts by selecting at random a sample vector x from the input and calculating all the weight vectors based on a distance measurement. The neuron whose weight vector has the minimum distance to the input vector x becomes the Best Matching Unit (BMU), Eq. 4.1:

$$\|x - m_c\| = min\{\|x - m_i\|\} \tag{4.1}$$

where x is the input vector, w the weight vector, c is the BMU and $\|\|$ is the distance measure. In this thesis, the Euclidian distance was used. After finding the BMU, all neurons (weight vectors) are updated to make the BMU moving closer to the input vector, Eq. 4.2:

$$w_i(t+1) = w_i(t) + \alpha(t)h_{ci}(t)[x(t) - w_i(t)] \tag{4.2}$$

where *wi(t)* is the weight vector which specifies the location of the output unit index, i, in the data space at time t, α is the learning rate and h_{ci} is the neighbourhood kernel close to the c (BMU). After convergence is reached, the resulting map is ordered topologically. Further details about the SOM can be found in [104]. In order to apply SOM, the input data set needs to be normalized to prevent certain variables (features, e.g. the minimum forward packet length (min_fpktl) value) from having a higher impact than the other variables. This normalization will transform all the variables to be within the range of 0 to 10 (log normalization).

In this thesis, the SOM PAK package and the SOM Toolbox [105, 106] are used to carry out the SOM-based experiments. Clustering using a SOM involves experimenting with parameters. The main parameters are the map dimensions, the number of iterations and the learning rate. The map dimensions have an effect on the number of clusters (units) the SOM generates. Since the data sets are relatively large, a bigger map size is needed. In this case, it is 6x6 (36 map units). The parameters for training the map are listed in Table 4.2.

Table 4.2: Parameters of the SOM

Parameters	**values**
X dimension	6
Y dimension	6
radius1	2
radius2	1
data Length (rlen_1)	rlen_1 multiply by 100
data Length (rlen_2)	rlen_1 multiply by 1000
alpha type	inverse_t
neighourhood	gaussian
alpha1	0.5
alpha2	0.05
topology	hexa

After training the SOM is finished, a Unified distance matrix (U-matrix) is used to visualize the grouping structure of high weight vectors between neurons. U-matrix is a color-heated map, which plots the distances of SOM neurons (Figure 4.1). The color-heated map ranges from dark red through shades of yellow and green to dark blue, where red implies high values and blue implies low values. A dark red color means a large distance between neurons, which indicates heterogeneous neighborhoods while a dark blue color means a small distance which indicates homogenous neighborhoods. The dark color can represent the cluster separators while the light color can represent the clusters. This color schema is useful when trying to find clusters in the data set without a priori knowledge.

Although the SOM is an unsupervised learning algorithm, in post training the Univ2007 data is input to SOM to find out the following: (i) the number of applications in each neuron; (ii) how spread is the Skype application on the map; and

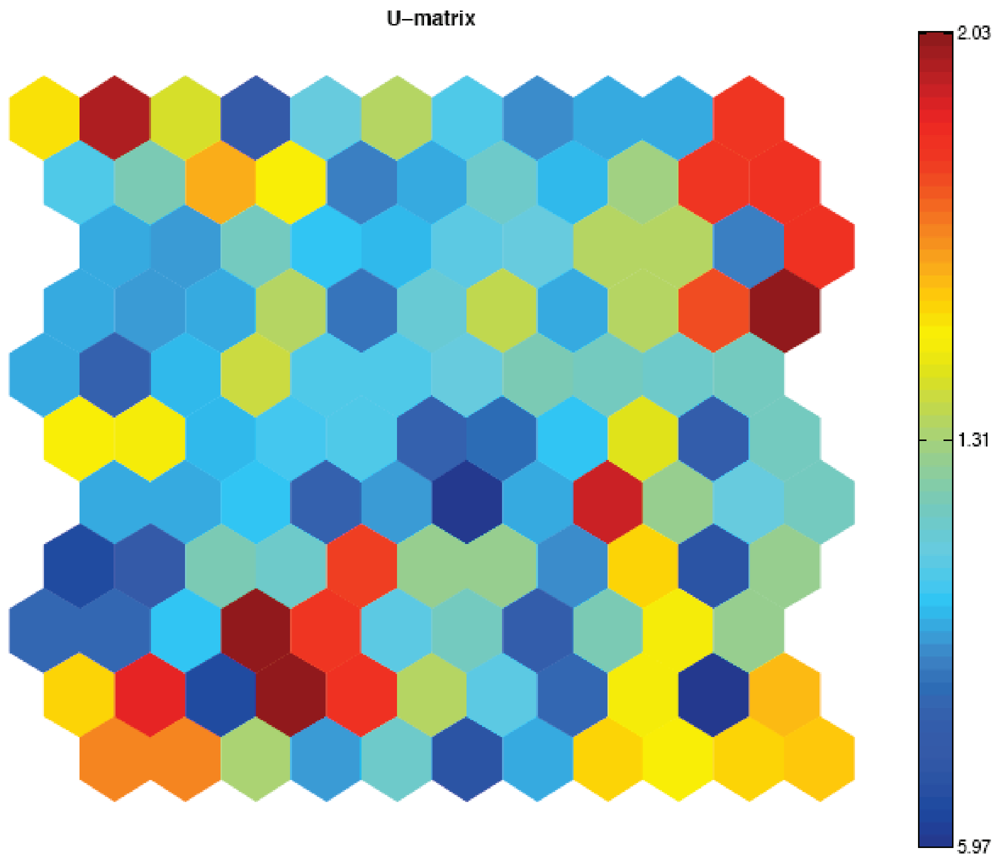


Figure 4.1: Unified Distance Matrix

(iii) how many applications are similar to Skype. Figure 4.2 shows that Skype is in 11 neurons and shared similarities with eight applications (FTP, SSH, MAIL, DNS, HTTP, HTTPS, MSN and OTHER). This is because a Skype application is trying to mimic the behaviour of other applications to avoid detection. Thus, based on this observation, the FTP, SSH, MAIL, DNS, HTTP, HTTPS, MSN and OTHER applications are going to be used as the out-class when sampling the training data set for the stratified sampling method ('a priori' method).

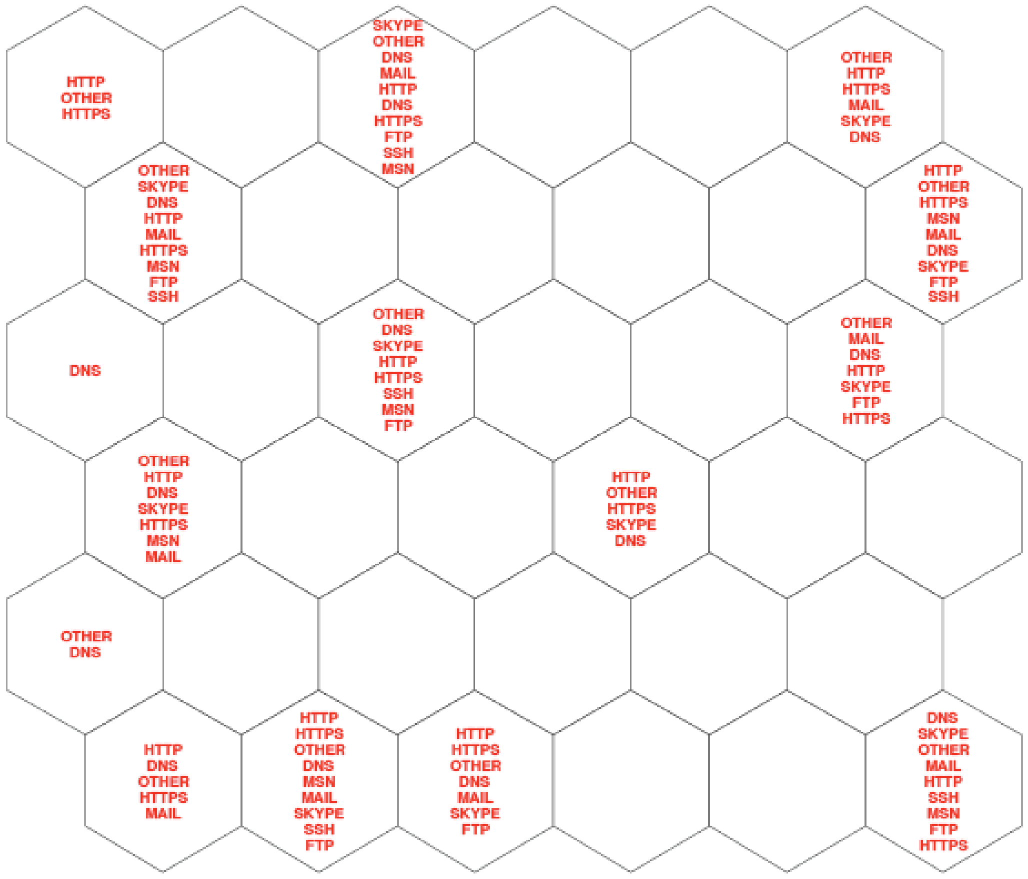


Figure 4.2: Distribution of Applications in each Neuron

4.2.2 Training Data Sets

Table 4.3 lists the number of flow records for each of the nine applications (FTP, SSH, MAIL, DNS, HTTP, HTTPS, MSN, Skype and OTHER) on the Univ2007 data set. In this case, based on the SOM classification, data is sampled randomly with uniform probability in different N fixed size samples and different N percentages from nine different applications (FTP, SSH, MAIL, DNS, HTTP, HTTPS, MSN, Skype and OTHER) to form the out/in-class for the training data set (Tables 4.4 and 4.5). For the N fixed size training data sets as in section 4.1, the data set is balanced so that the in-class and out-class have the same number of flows. For the fixed size of N records, the number of out-class flows is divided by the number of out-class applications (e.g. for the 30K classes 15000/8=1875 flows for each class). If an application has a fewer number of flows in the data set than the number necessary for sampling, then, the missing flows would be sampled randomly from the OTHER class. For instance, there were 7,684 FTP flows in the Univ2007 data set, when sampled for the 200K class training data set. The number of FTP flows is less than the 12,500 allocation for the FTP application so the missing flows would be taken from the OTHER class.

Table 4.3: Number of Flows for each Application in the Univ2007 Trace

Applications	**Number of Flows**
FTP	7,684
SSH	18,993
MAIL	359,430
DNS	5,032,876
HTTP	5,670,386
HTTPS	1,144,505
MSN	344,408
OTHER	8,146,792
SKYPE	8,254,782

Table 4.4: Number of Flows Sampled for the Stratified Sampling Method with Fixed *N* Records

Applications	**30K**	**60K**	**100K**	**200K**	**400K**	**800K**
FTP	1,875	3,750	6,250	7,500	7,500	7,500
SSH	1,875	3,750	6,250	12,500	18,000	18,000
MAIL	1,875	3,750	6,250	12,500	25,000	50,000
DNS	1,875	3,750	6,250	12,500	25,000	50,000
HTTP	1,875	3,750	6,250	12,500	25,000	50,000
HTTPS	1,875	3,750	6,250	12,500	25,000	50,000
MSN	1,875	3,750	6,250	12,500	25,000	50,000
OTHER	1,875	3,750	6,250	17,500	49,500	124,500
SKYPE	15,000	30000	50,000	100,000	200,000	400,000
TOTAL	30,000	60,000	100,000	200,000	400,000	800,000

Table 4.5: Number of Flows Sampled for the Stratified Sampling Method with Fixed *N* Percentage

Applications	**1%**	**2%**	**3%**	**4%**	**6%**	
FTP	76	153	230	307	384	461
SSH	189	379	569	759	949	1,139
MAIL	3,594	7,188	10,782	14,377	1,797	21,565
DNS	50,328	100,657	150,986	201,315	251,643	301,972
HTTP	56,703	113,407	170,111	226,815	283,519	340,223
HTTPS	11,445	22,890	34,335	45,780	57,225	68,670
MSN	3,444	6,888	10,332	13,776	17,220	20,664
OTHER	81,467	162,935	244,403	325,871	407,339	488,807
SKYPE	82,547	165,095	247,643	330,191	412,739	495,286
TOTAL	289,793	579,592	869,391	1,159,191	1,448,989	1,738,787

4.3 Continuous Data Stream

Network traffic traces are a real-time continuous stream of packets, which are ordered explicitly by the timestamp of the packets. Typically, these continuous data streams have unique characteristics, which depict the network infrastructure and user behaviour. In Figure 4.3, there are different peeks for TCP and UDP traffic for the Univ2007. For instance, at 16:30 PM there is an increase in the number of UDP packets while there is a decrease in the number of TCP packets. Moreover, the number of TCP packets fluctuates on the trace, which shows that the Dalhousie users tend to use more applications that run on TCP than UDP.

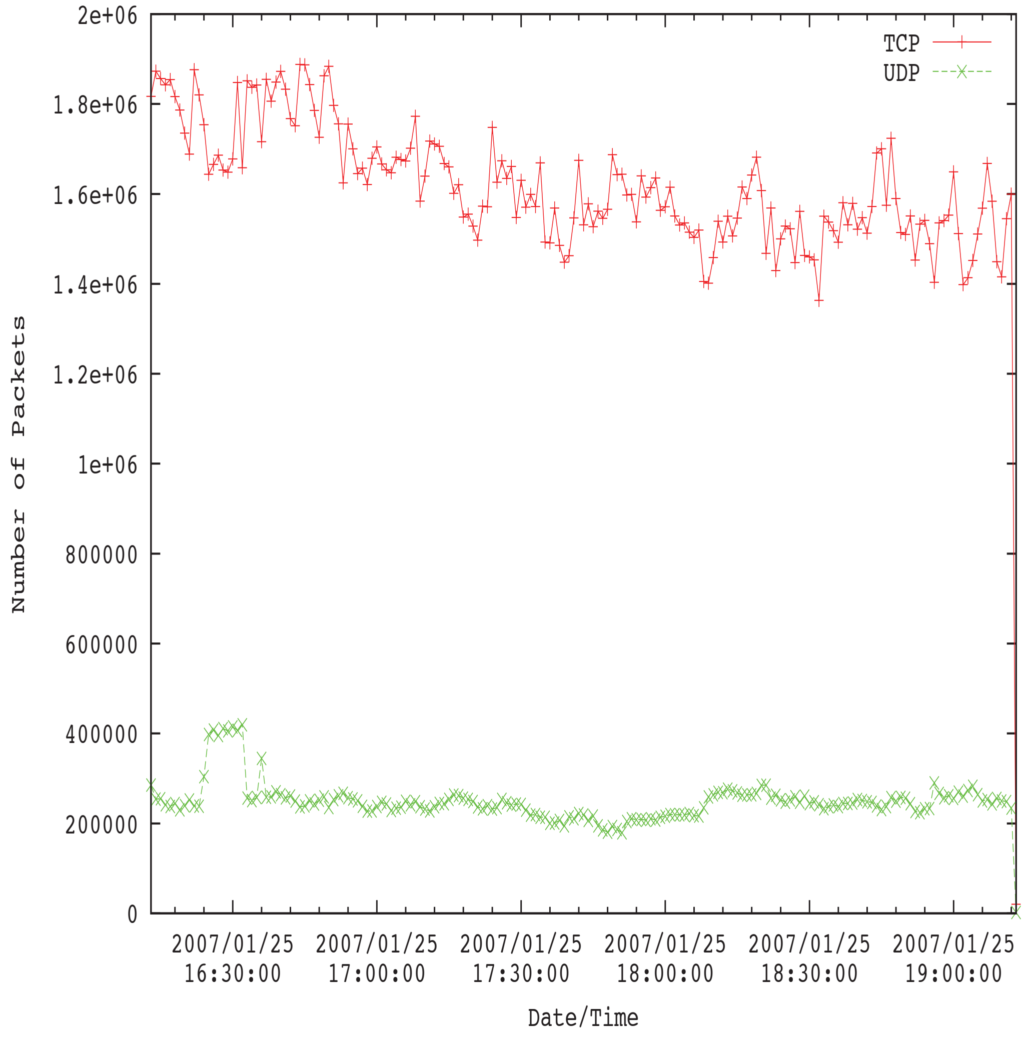


Figure 4.3: Number of Packets for TCP/UDP Protocols in the Univ2007 Trace

To capture this behaviour in the training data set the flows were sampled based on the order in which they arrived in two different techniques (Table 4.6). The first technique used is to sample a fixed size number of records as in the previous two sections (4.1 and 4.2). For example, for the "First 30K" method the first 15000 Skype flow records and the first 15000 non-Skype flow records were selected. The second technique used is involves sampling over a continuous time period (e.g. first 30 minutes, first 60 minutes, etc.).

Table 4.6: Number of Flows Sampled for a Continuous Data Stream

Sampling Methods	**# of Skype**	**# of non-Skype**	**Total**
First 30k records	15,000	15,000	30,000
First 60k records	30,000	30,000	60,000
First 100k records	50,000	50,000	100,000
First 200k records	100,000	100,000	200,000
First 400K records	200,000	200,000	400,000
First 800k records	400,000	400,000	800,000
First 30 minutes	1,283,805	3,235,084	4,518,889
First 60 minutes	2,683,941	6,645,860	9,329,801
First 90 minutes	4,227,140	10,327,200	14,554,340

4.4 Results of Experiments for Subset Sampling

In total, 33 training data sets were sampled using the three subset sampling techniques. The sizes of the training data sets vary from thousands of flow records to millions of flow records (e.g. sizes from 30,000 flow records to 14,554,340 flow records). For these experiments each classifier was trained initially on the training data set (which is a subset sampled from the Univ2007 traces) using the same feature set. The flow feature set was chosen since the size of the data sets is smaller compared to the size of the data generated by the packet header feature set. Then each trained model (C5.0, AdaBoost and GP) was tested on a validation data set, namely, a subset of the Univ2010 test traces. The validation data set consists of randomly sampled (with uniform probability) 1000 flow records of ten applications from the Univ2010 trace for a total of 10000 records. The ten applications were FTP, SSH, MAIL, DNS, HTTP, HTTPS, MSN, P2P, Skype and OTHER. The validation data was used to evaluate the most appropriate subset sampling method for generating generalized/robust signatures not to evaluate the performance of the ML algorithms. Furthermore, the size of the test data sets is huge and therefore, evaluating the 33 training data sets on the test data sets would have required a very long time. Thus, the best training data set is selected through the results of the validation process. Once the best one is selected, it is then evaluated on the test data sets. Since the Univ2010 data set is a real network trace captured from the same location as the Univ2007 traces but at a different time period and contains many applications, it is the most suitable one for validating and testing the robustness of the classifiers. The training performances and the validation performances of the 50 runs on the 33 training data sets are given using density of distribution/box plots for the Uniform Random N Sampling method, Stratified sampling method and Continuous sampling method in Appendices A.1, A.2 and A.3, respectively.

It should be noted here that the highest performance solution was selected based on the DR and FPR on the training results for GP, AdaBoost and C5.0 to be evaluated on the validation data sets. That is to say, all model construction took place on the Univ2007 Training partitions. The validation data set, which is a subset of Univ2010 where none were encountered during training, was used for the post training evaluation. Table 4.7 lists the performances of the best models obtained on the validation data. Results show that on the validation data partitions, C5.0 turns out to provide the stronger performance with consistently lower FPR and higher DR while using the Uniform Random sampling with the 6 Percent (97% DR and 0.04% FPR). Moreover, a one-way ANOVA test was employed to compare the performance of the subset sampling techniques based on the DR and FPR (e.g. one-way ANOVA with n=50 data points). Results of the one-way ANOVA statistical analysis test in Appendix B.1 show that the mean of 50 runs of C5.0-based classifiers using the Uniform Random sampling with 6 the Percent is statistically significantly better than the mean of the other ML algorithms on the training and validation data sets. Thus the Uniform Random sampling method with 6 Percent was chosen as the method for sampling the training data set.

Table 4.7: Results of the Best Models for Each Classifier on the Validation Data for Subset Sampling Methods

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Uniform Random *N* Sampling						
30K	0.76	0.04	0.73	0.08	0.76	0.07
60K	0.72	0.04	0.73	0.08	0.76	0.04
100K	0.72	0.05	0.73	0.08	0.76	0.07
200K	0.72	0.05	0.73	0.08	0.76	0.07
400K	0.73	0.04	0.73	0.08	0.76	0.08
800K	0.74	0.06	0.73	0.08	0.74	0.05
1%	0.73	0.05	0.68	0.05	0.76	0.06
2%	0.73	0.05	0.68	0.05	0.76	0.07
3%	0.72	0.05	0.68	0.05	0.74	0.04
4%	0.72	0.03	0.68	0.05	0.75	0.06
5%	0.65	0.03	0.68	0.05	0.75	0.04
6%	0.97	0.04	0.68	0.05	0.74	0.06
Stratified Sampling						
30K	0.75	0.06	0.68	0.05	0.76	0.07
60K	0.76	0.08	0.68	0.05	0.77	0.08
100K	0.75	0.05	0.68	0.05	0.76	0.09
200K	0.75	0.05	0.68	0.05	0.76	0.08
400K	0.74	0.07	0.68	0.05	0.76	0.09
800K	0.68	0.04	0.68	0.05	0.77	0.06
1%	0.68	0.05	0.68	0.07	0.76	0.06
2%	0.67	0.06	0.68	0.05	0.76	0.07
3%	0.60	0.05	0.68	0.05	0.76	0.09
4%	0.66	0.04	0.68	0.05	0.75	0.07
5%	0.72	0.04	0.68	0.05	0.76	0.05
6%	0.71	0.04	0.68	0.07	0.75	0.08
Continuous data streams						
30K	0.67	0.05	0.61	0.06	0.74	0.08
60K	0.72	0.04	0.68	0.05	0.76	0.08
100K	0.64	0.04	0.68	0.05	0.77	0.06
200K	0.74	0.09	0.68	0.05	0.75	0.07
400K	0.69	0.04	0.68	0.05	0.75	0.06
800K	0.69	.07	0.71	0.06	0.75	0.07
30 minutes	0.68	0.05	0.68	0.05	0.75	0.07
60 minutes	0.68	0.04	0.70	0.08	0.75	0.07
90 minutes	0.65	0.04	0.68	0.05	0.77	0.06

4.5 Results of Experiments for the Best Subset Sampling Technique – Uniform Random Sampling of 6 Percent

Since the Uniform Random subset sampling with 6 the Percent has the best performance on the validation data set, the performance of the three trained models (C5.0, AdaBoost and GP) was tried out on the test data sets, namely the Univ2007 test traces and the Univ2010, NIMSII, NIMSIII, IPv6 and ITALY traces.

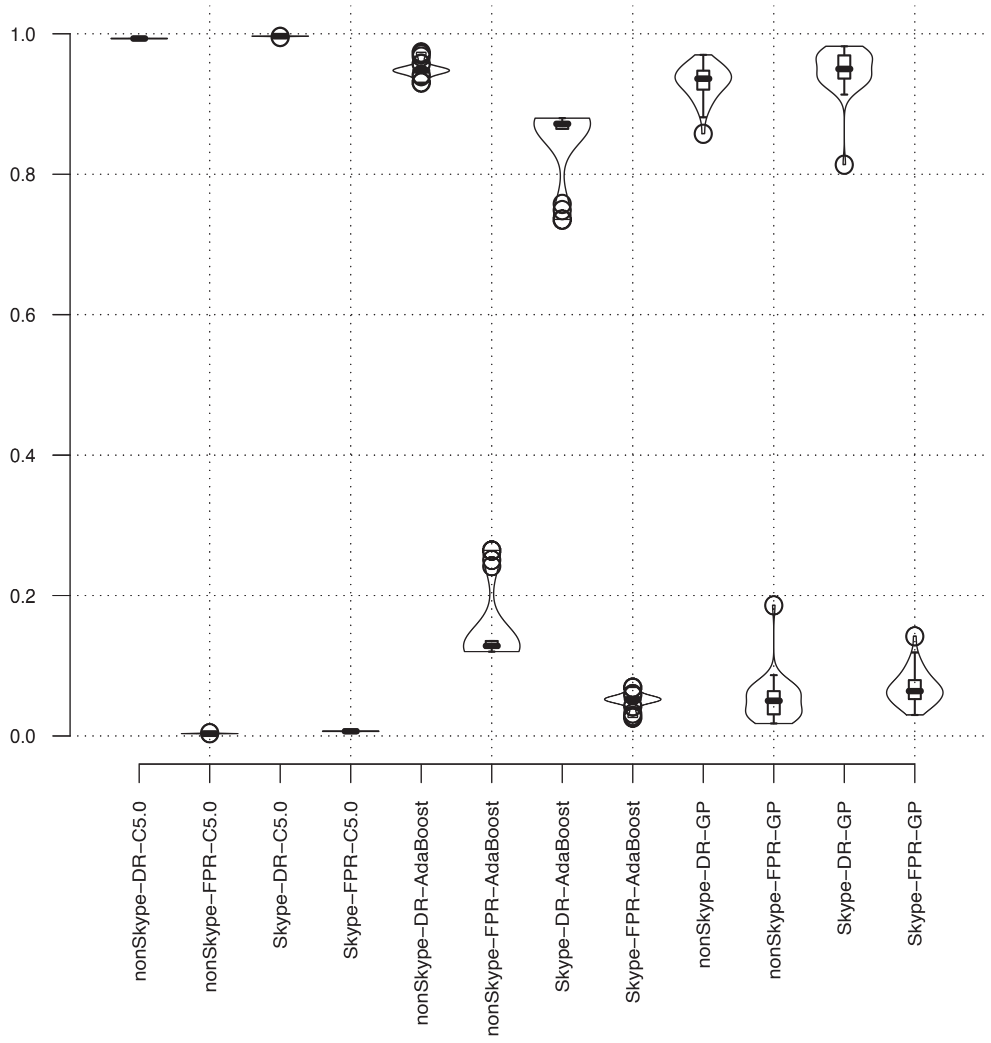


Figure 4.4: DR and FPR Results for Skype Identification on the Training Data Sets Using the Uniform Random with 6 the Percent Sampling Method

The training performances for the fifty runs were plotted using density of distribution/box plots (Figure 4.4). Results show that on the training data partitions, C5.0 turns out to provide the stronger performance with a consistently lower FPR and higher DR for all the fifty runs based on a one-way ANOVA statistical analysis test (Appendix B.2). Moreover, the box-plot of the C5.0 solutions has a very small range which implies that most of the C5.0 models have similar performance. Hence, the fifty models found by C5.0 are highly similar to each other. By contrast, AdaBoost and GP have different ranges of DR and FPR. This implies that both algorithms found different solutions on different runs. The best performing solution was selected based on the DR and FPR on the training results for GP, AdaBoost and C5.0 to be evaluated on the test data sets. In other words, all model construction took place on the Univ2007 Training partitions. Post training evaluation was conducted on Univ2007 test, Univ2010, NIMS, IPv6 and ITALY traces where none were encountered during training. Naturally, the Univ2007 Test partition reflects the training behaviour more closely than the Univ2010, IPv6, all NIMS, IPv6 and ITALY data sets.

To select the best trained model for each machine learning algorithm the training performance was plotted as a scatter plot for each of the three machine learning algorithms. Figures 4.5, 4.6 and 4.7 summarize these solutions. For Skype, there are five solutions which are non-dominated for GP, four which are non-dominated for AdaBoost and four which are non-dominated for C5.0. The solution with the highest performance in terms of high DR and low FPR was selected out of these non-dominated solutions for GP, AdaBoost and C5.0 and then evaluated on the test data sets.

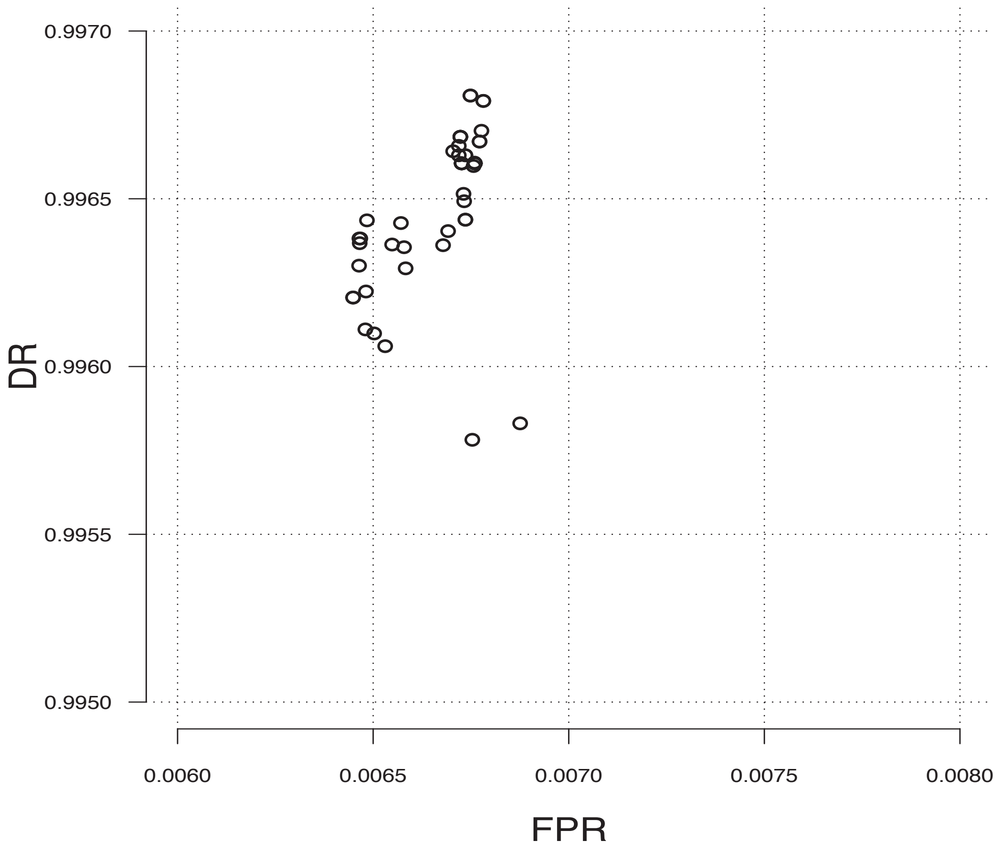


Figure 4.5: Scatter Plot for the C5.0 Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR)

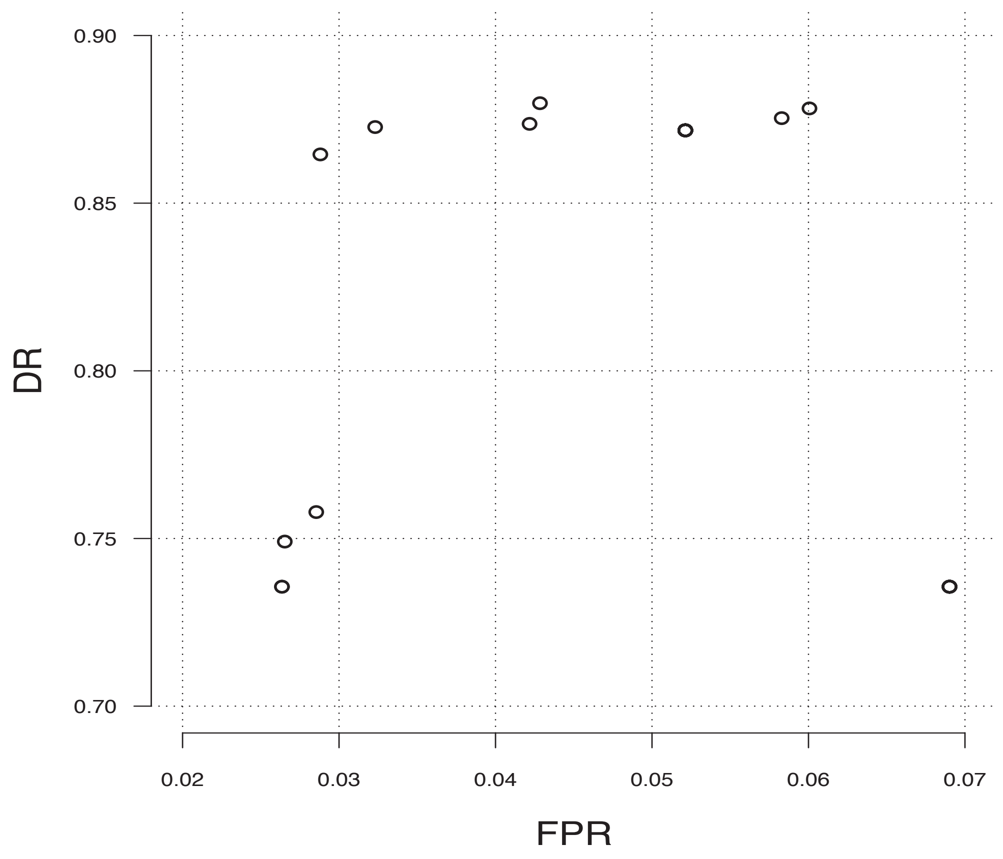


Figure 4.6: Scatter Plot for the AdaBoost Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR)

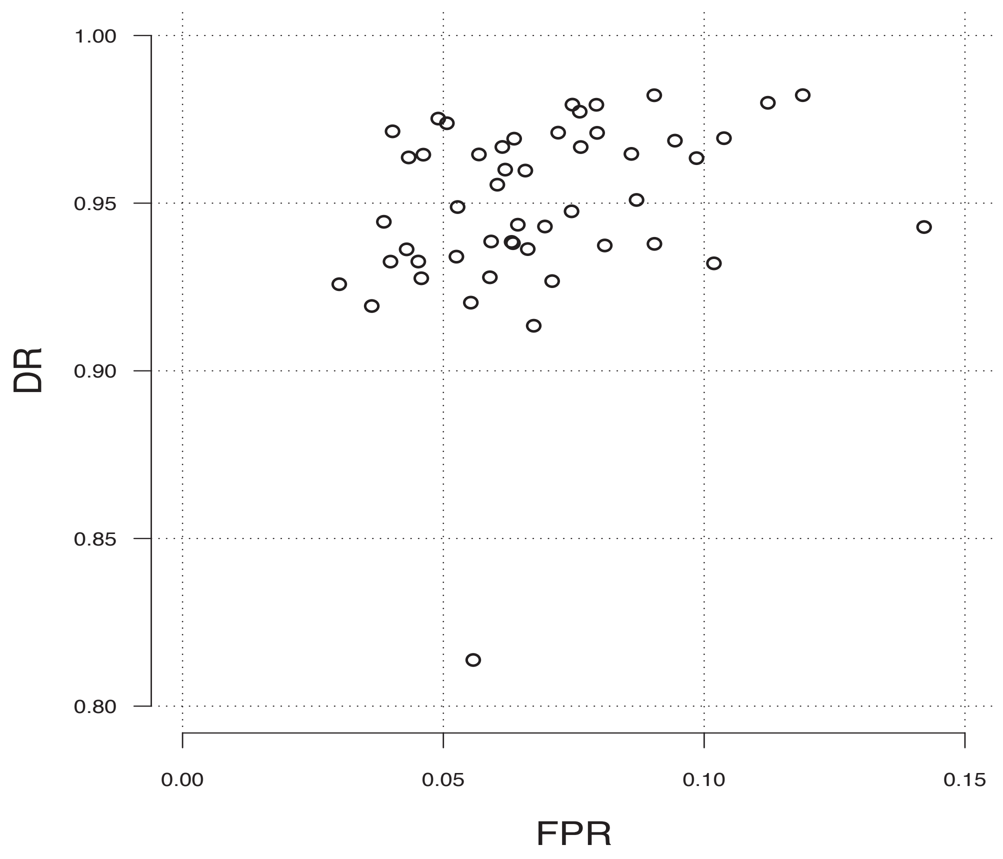


Figure 4.7: Scatter Plot for the GP Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR)

Tables 4.8, 4.9, 4.10, 4.11 and 4.12 show the results of the best models for the three machine learning algorithms on the unseen Univ2007 test trace and the independent test traces (Univ2010, NIMSII, NIMSIII, IPv6 and ITALY traces). Under the Univ2007 and Univ2010 test partitions, C5.0 appears to provide the stronger performance with consistently better DR and FPR than AdaBoost and GP. Introducing the entirely independent test set – ITALY traces – indicated that C5.0 had overlearned the properties implicit in the training partition. Indeed most Skype flow traffic was misclassified as non-Skype. Moreover, GP was observed to provide best case performance under the independent ITALY test partition. Results show that the GP classifier performs better than the other classifiers on the ITALY test trace while C5.0 performs better on the Univ2010 test trace. The GP classifier achieved 86% DR and 8% FPR on the Univ2010, 97% DR on the ITALY-UDPE2E, 100% DR on the ITALY-UDPE2O and 88% on the ITALY-UDPSIG. The C5.0 classifier achieved 83% DR and 4% FPR on the Univ2010, 61% DR on the ITALY-UDPE2E, 53% DR on the ITALY-UDPE2O and 78% on the ITALY-UDPSIG test traces. In contrast, AdaBoost provides the best results on the NIMS and IPv6 data sets (Tables 4.10, 4.11 and 4.12). The next best classifier after AdaBoost on the NIMS and IPv6 data set is C5.0. Thus, there is no best classifier which can have consistent performance on all the test data sets. Moreover, none of the classifiers could detect any of ITALY-TCPE2X since there was no Skype TCP flows in the training data set (because there were no such labelled traffic in the Univ2007 data set).

Table 4.8: Best Model Results of the Subset Sampling Method (6% Uniform Random Sampling) for Skype Detection Based on Flow Feature Sets (Training and Testing University Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample (subset of Univ2007)						
Non-SKYPE	0.993	0.004	0.957	0.120	0.936	0.031
SKYPE	0.993	0.005	0.957	0.120	0.969	0.064
Univ2007 Test data sets						
Non-SKYPE	0.993	0.005	0.957	0.120	0.936	0.031
SKYPE	0.995	0.007	0.880	0.043	0.969	0.064
Univ2010 Test data sets						
Non-SKYPE	0.956	0.169	0.932	0.189	0.922	0.144
SKYPE	0.831	0.044	0.811	0.068	0.856	0.078

Table 4.9: Best Model Results for the Subset Sampling Method (6% Uniform Random Sampling) for Skype Detection Based on Flow Feature Sets (ITALY Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
ITALY TCPE2X Test data sets						
Non-SKYPE	N/A	1.00	N/A	1.00	N/A	1.00
SKYPE	0.000	N/A	0.000	N/A	0.000	N/A
ITALY UDPE2E Test data sets						
Non-SKYPE	N/A	0.390	N/A	0.794	N/A	0.033
SKYPE	0.610	N/A	0.206	N/A	0.967	N/A
ITALY UDPE2O Test data sets						
Non-SKYPE	N/A	0.947	N/A	0.977	N/A	0.003
SKYPE	0.053	N/A	0.023	N/A	0.997	N/A
ITALY UDPSIG Test data sets						
Non-SKYPE	N/A	0.216	N/A	0.304	N/A	0.125
SKYPE	0.784	N/A	0.696	N/A	0.875	N/A

Table 4.10: Best Model Results for the Subset Sampling Method (6% Uniform Random Sampling) for Skype Detection Based on Flow Feature Sets (NIMSII Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
NIMSII GTALK2009 Test data sets						
Non-SKYPE	0.999	N/A	0.999	N/A	0.542	N/A
SKYPE	N/A	0.001	N/A	0.001	N/A	0.458
NIMSII PRIMUS2009 Test data sets						
Non-SKYPE	1.00	N/A	1.00	N/A	0.680	N/A
SKYPE	N/A	0.000	N/A	0.000	N/A	0.320
NIMSII ZFONE2009 Test data sets						
Non-SKYPE	0.996	N/A	0.940	N/A	0.624	N/A
SKYPE	N/A	0.004	N/A	0.060	N/A	0.376

Table 4.11: Best Model Results for the Subset Sampling Method (6% Uniform Random Sampling) for Skype Detection Based on Flow Feature Sets (NIMSIII Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
NIMSIII GTALK2010 Test data sets						
Non-SKYPE	0.997	N/A	0.997	N/A	0.418	N/A
SKYPE	N/A	0.003	N/A	0.003	N/A	0.582
NIMSIII PRIMUS2010 Test data sets						
Non-SKYPE	0.972	N/A	0.808	N/A	0.673	N/A
SKYPE	N/A	0.028	N/A	0.192	N/A	0.327
NIMSIII YAHOO2010 Test data sets						
Non-SKYPE	0.932	N/A	0.933	N/A	0.373	N/A
SKYPE	N/A	0.068	N/A	0.067	N/A	0.627
NIMSIII RADIO2010 Test data sets						
Non-SKYPE	0.997	N/A	0.997	N/A	0.997	N/A
SKYPE	N/A	0.003	N/A	0.003	N/A	0.003
NIMSIII TORRENT2010 Test data sets						
Non-SKYPE	0.949	N/A	0.934	N/A	0.846	N/A
SKYPE	N/A	0.051	N/A	0.066	N/A	0.154
NIMSIII TV2010 Test data sets						
Non-SKYPE	0.994	N/A	1.00	N/A	0.983	N/A
SKYPE	N/A	0.006	N/A	0.000	N/A	0.017
NIMSIII VPN2010 Test data sets						
Non-SKYPE	0.987	N/A	1.00	N/A	1.00	N/A
SKYPE	N/A	0.013	N/A	0.000	N/A	0.000

Table 4.12: Best Model Results for the Subset Sampling Method (6% Uniform Random Sampling) for Skype Detection Based on Flow Feature Sets (IPv6 Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
IPv6 2000 Test data sets						
Non-SKYPE	0.999	N/A	1.00	N/A	0.000	N/A
SKYPE	N/A	0.001	N/A	0.000	N/A	1.00
IPv6 2009 Test data sets						
Non-SKYPE	0.889	N/A	1.00	N/A	0.000	N/A
SKYPE	N/A	0.111	N/A	0.000	N/A	1.00

4.6 Discussion of Results

The primary motivation addressed in this chapter is the challenging problem of sampling training data sets for ML algorithms in order to generate robust signatures for detecting VoIP (and specifically Skype) traffic. The effect of using three sampling techniques was investigated with a total of 33 training data sets using three machine learning algorithms (AdaBoost, GP and C5.0). To do so, traces from University, WIDE (MAWI), NIMS lab and ITALY repositories were used. The aforementioned learning algorithms were evaluated using traffic flow-based features.

These results demonstrate that the signatures, which the AdaBoost, GP and C5.0 classifiers generated during training, are robust (transportable) enough to be tested on real world network traces. Furthermore, it is possible to have a well generalized (robust) rules set and a generic attributes set which can be employed to identify encrypted traffic – such as Skype traffic – using a 6% uniform random sampling method. However, the most noticeable weakness of the signatures is the Skype TCP data. Therefore, to enhance the performance of the signatures found, it was decided to add Skype TCP data to the training data sets. Thus, in the next chapter, randomly 6% of the ITALY TCPE2X trace is sampled randomly with uniform probability and added to the chosen training data sets described in this chapter, since there are no any Skype TCP flows in the Univ2007 traces.

Chapter 5

Exploring the Robustness/Generalization of the Classifiers

In this chapter, the effectiveness of my proposed approach for finding robust signatures is discussed. To this end, test data sets which are completely different from the training data sets are used in order to provide some measure of classifier generalization (robustness). Thus, solution robustness is assessed by training on a data set from one location (the Univ2007 trace) but testing on data sets from different locations, different networks and different time periods (Univ2007 Test partition, Univ2010, NIMSII, NIMSIII, ITALY and IPv6, which were captured in 2007, 2010, 2009, 2010, 2006, and in both 2000 and 2009 for IPv6, respectively). Issues of data representation are addressed by employing packet header-based features and flow-based features but without using IP addresses, port numbers and payload data.

5.1 Results of Packet Header Experiments for Skype Identification – Robustness of Signatures

In this set of experiments the objective is to identify Skype on a packet per packet basis using only the features given in Table 3.1. The training data set is generated by sampling randomly with a uniform probability of 6% from both classes (in-class and out-class). Moreover, 6% of traffic from the ITALY (Skype TCPE2X) data is sampled and added to the training data set to represent the TCP based Skype behaviour during training, too. In total, the training data set consists of 20,142,770 packets.

The results are presented in Figure 5.1 and the one-way ANOVA statistical analysis test in Appendix B.3.1 illustrates that the GP-based classification approach is much better on average than other algorithms employed in identifying the Skype packets based on the training data set. The violin plot demonstrates the diversity of the performance in terms of DR and FPR for all fifty models on the training data set for each ML algorithm (Figure 5.1). On average, GP is much better than the other ML algorithms on the training data sets for Skype in terms of having a high DR.

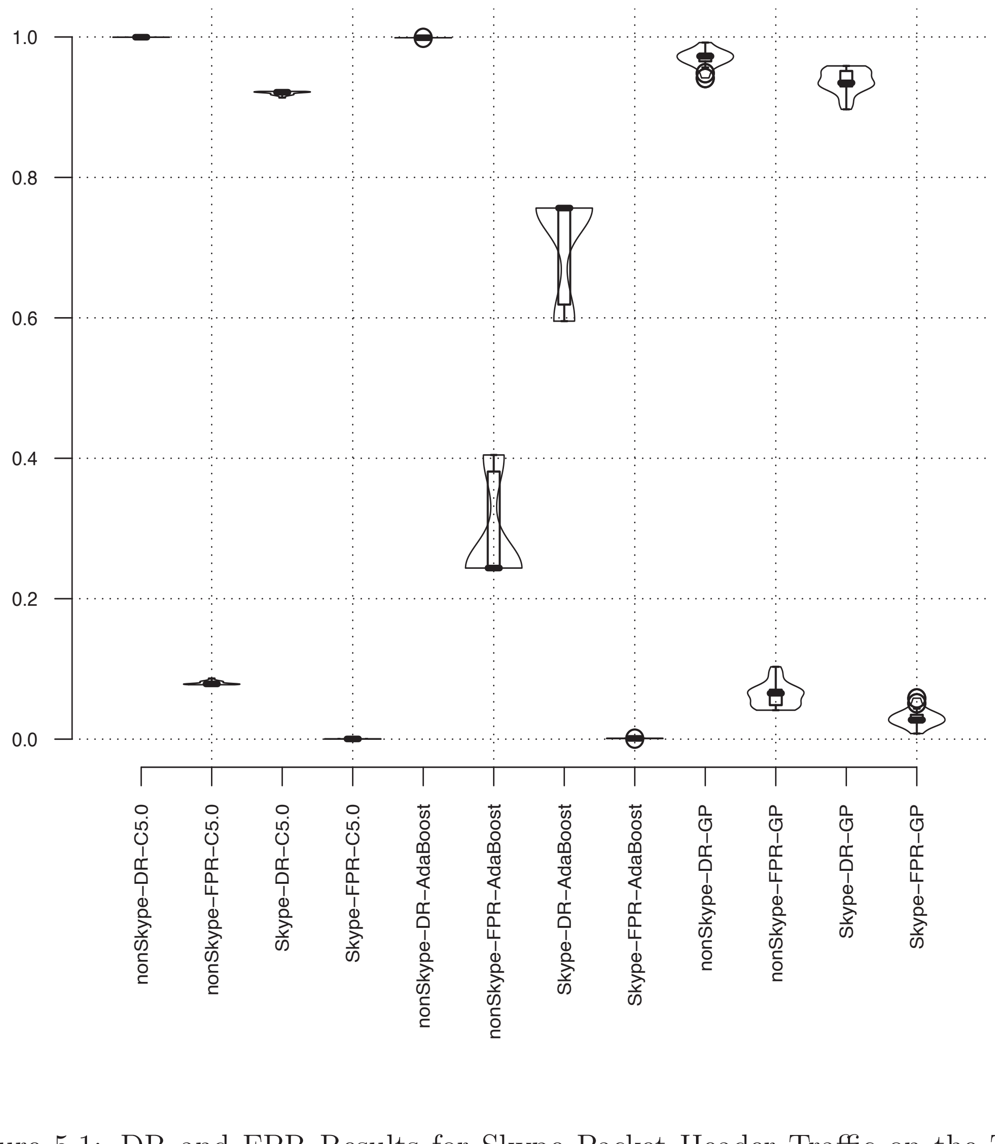


Figure 5.1: DR and FPR Results for Skype Packet Header Traffic on the Training Data Set

Figures 5.2, 5.3 and 5.4 summarize the performance of trained models for each ML algorithm. For Skype, there are nine solutions, which are non-dominated for GP, one which is non-dominated for AdaBoost and four which are non-dominated for C5.0. The best performing solution in terms of high DR and low FPR out of these non-dominated solutions was selected for GP, AdaBoost and C5.0, and then evaluated on the test data sets.

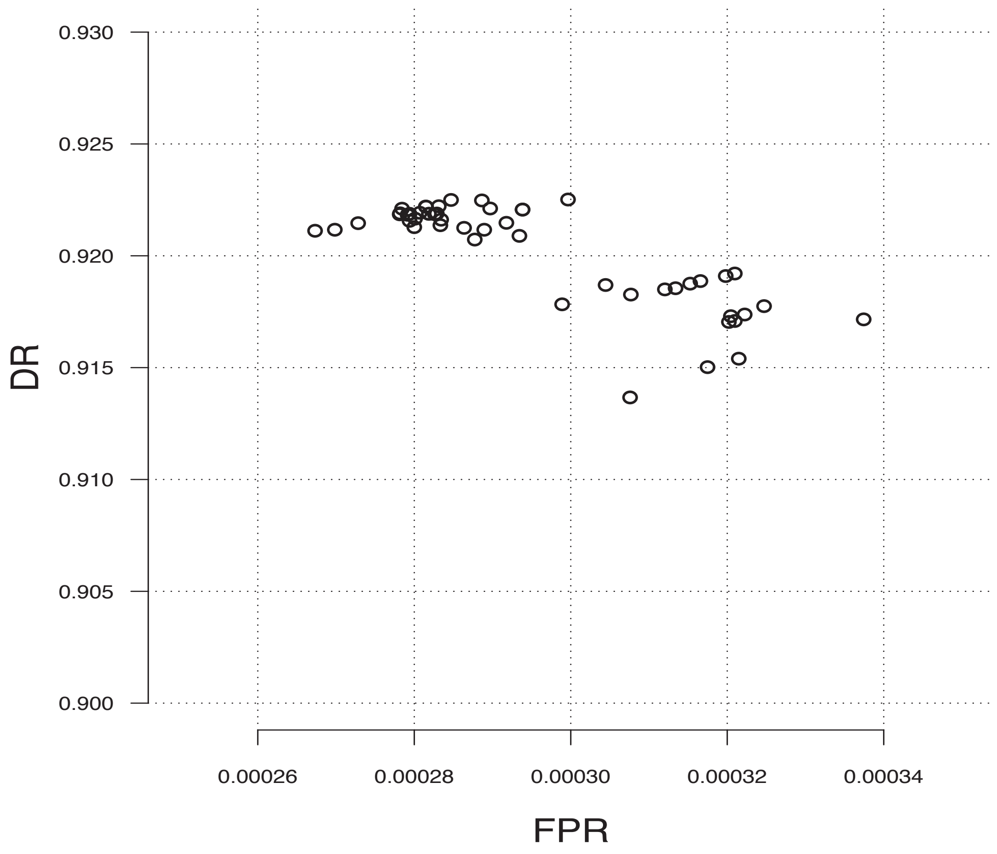


Figure 5.2: Scatter Plot for the C5.0 Classifier for Training Performance Using the Packet Header-based Feature Set for Skype Detection (DR versus FPR)

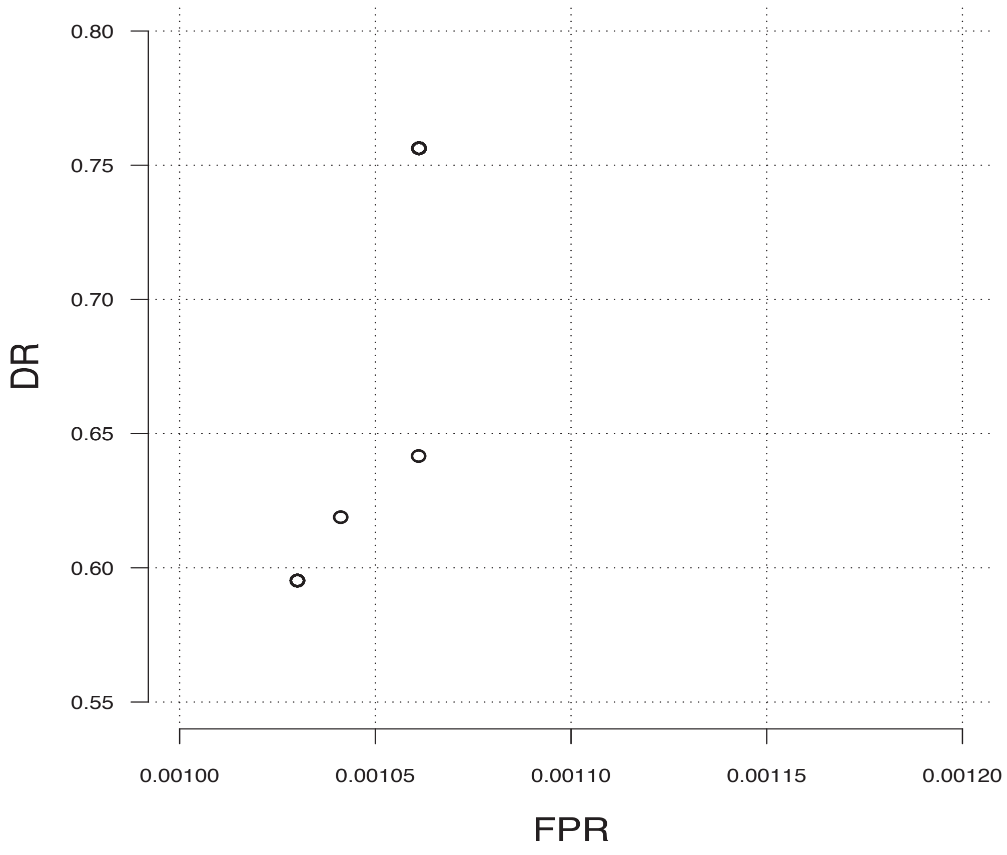


Figure 5.3: Scatter Plot for the AdaBoost Classifier for Training Performance Using the Packet Header-based Feature Set for Skype Detection (DR versus FPR)

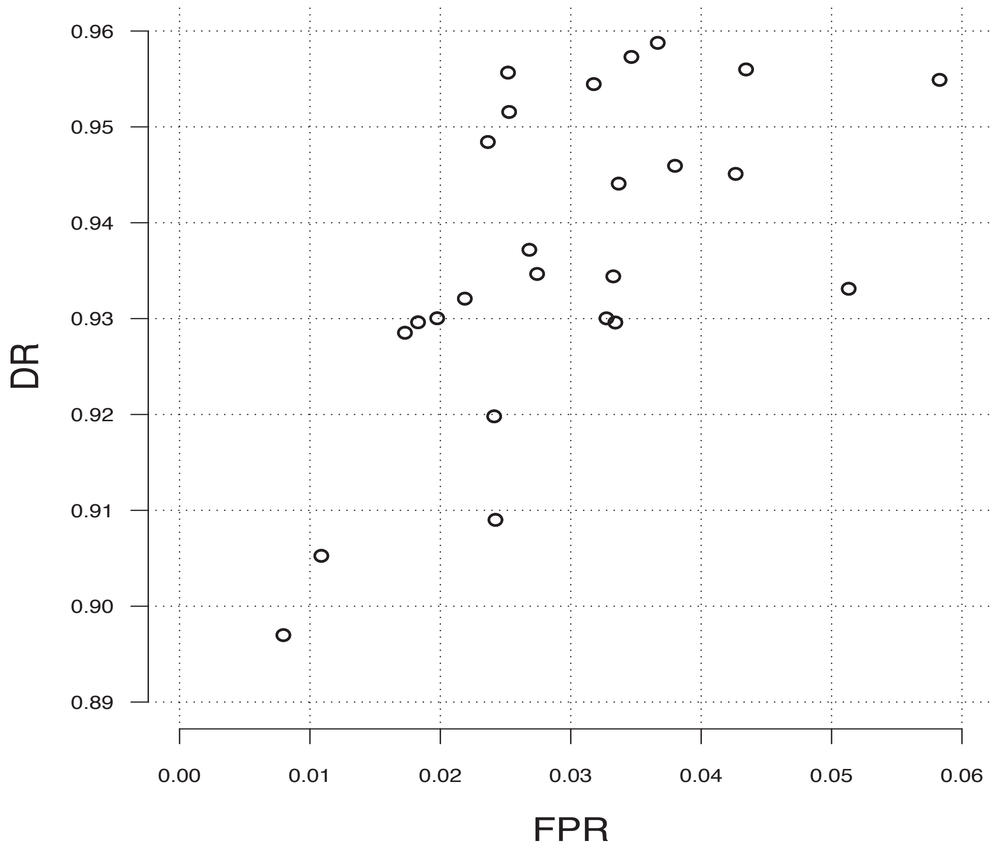


Figure 5.4: Scatter Plot for the GP Classifier for Training Performance Using the Packet Header-based Feature Set for Skype Detection (DR versus FPR)

For the Skype packet header, Tables 5.1, 5.2, 5.3, 5.4, and 5.5 list the results for the three ML algorithms on the training and independent test traces. All models return a high DR for Skype traffic on the training data set and C5.0 and GP appear to provide the strongest performance with a consistently better FPR on the Univ2007 test partition. Naturally, the Univ2007 test partition reflects the training behaviour the most of all the test data sets.

Introducing the entirely independent test sets – Univ2010, ITALY, IPv6 and all NIMS data sets – indicates that all classifiers have overlearned the properties implicit in the training partition. However, GP and C5.0 were observed to provide a best case performance under the ITALY independent test partitions. All learning algorithms performed poorly under the Univ2010 and NIMS data sets.

Results show that the C5.0 classifier performs better than the other classifiers on the majority of ITALY traces (Table 5.2). The C5.0 classifier achieves $\approx$99% DR on the ITALY Test TCPE2X trace, $\approx$100% DR on the ITALY Test UDPE2E trace and $\approx$99% DR on the ITALY Test UDPSIG trace. By contrast, GP achieves 100% DR on the ITALY Test UDPE2O trace. Moving to the NIMS traces (NIMSII and NIMSIII), AdaBoost performed better than C5.0 and GP, most noticeably on the TORRENT2010 and VPN2010 data sets (Tables 5.3 and 5.4). In this case, robustness results illustrate the performance of the packet header-based approach in identifying Skype traffic based on a single packet (packet by packet) without using IP addresses, port numbers or payload information. It should be noted here that the FP rate for Skype and DR for non-Skype are zero in the ITALY traces and the DR for Skype and FP rate for non-Skype are zero in the NIMSII, NIMSIII and IPv6 traces because the ITALY traces contain only Skype traffic while the NIMSII, NIMSIII and IPv6 traces contain only non-Skype traffic.

Table 5.1: Best Model Results on the Training Data Set for the Packet Header-based Feature Set for Skype Detection (Training and Testing on Dalhousie Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample (subset of Univ2007)						
Non-SKYPE	1.00	0.077	0.999	0.244	0.975	0.044
SKYPE	0.923	0.000	0.756	0.001	0.956	0.025
Univ2007 Test data sets						
Non-SKYPE	1.00	0.131	1.00	0.245	0.975	0.066
SKYPE	0.869	0.000	0.755	0.001	0.934	0.025
Univ2010 Test data sets						
Non-SKYPE	0.976	0.940	0.997	0.994	0.349	0.212
SKYPE	0.06	0.024	0.006	0.003	0.788	0.651

Table 5.2: Best Model Results on Testing for the Packet Header-based Feature Set for Skype Detection (ITALY Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
ITALY TCPE2X Test data sets						
Non-SKYPE	N/A	0.008	N/A	0.243	N/A	0.014
SKYPE	0.992	N/A	0.757	N/A	0.986	N/A
ITALY UDPE2E Test data sets						
Non-SKYPE	N/A	0.001	N/A	0.13	N/A	0.008
SKYPE	0.999	N/A	0.87	N/A	0.992	N/A
ITALY UDPE2O Test data sets						
Non-SKYPE	N/A	0.064	N/A	0.43	N/A	0.000
SKYPE	0.936	N/A	0.57	N/A	1.00	N/A
ITALY UDPSIG Test data sets						
Non-SKYPE	N/A	0.006	N/A	0.217	N/A	0.015
SKYPE	0.994	N/A	0.783	N/A	0.985	N/A

Table 5.3: Best Model Results on Testing Sets for the Packet Header-based Feature Set for Skype Detection (NIMSII Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
NIMSII GTALK2009 Test data sets						
Non-SKYPE	0.032	N/A	0.067	N/A	0.023	N/A
SKYPE	N/A	0.968	N/A	0.933	N/A	0.977
NIMSII PRIMUS2009 Test data sets						
Non-SKYPE	0.033	N/A	0.001	N/A	0.000	N/A
SKYPE	N/A	0.967	N/A	0.999	N/A	1.00
NIMSII ZFONE2009 Test data sets						
Non-SKYPE	0.008	N/A	0.018	N/A	0.002	N/A
SKYPE	N/A	0.992	N/A	0.982	N/A	0.998

Table 5.4: Best Model Results on Testing for the Packet Header-based Feature Set for Skype Detection (NIMSIII Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
NIMSIII GTALK2010 Test data sets						
Non-SKYPE	0.05	N/A	0.081	N/A	0.046	N/A
SKYPE	N/A	0.95	N/A	0.919	N/A	0.954
NIMSIII PRIMUS2010 Test data sets						
Non-SKYPE	0.03	N/A	0.000	N/A	0.000	N/A
SKYPE	N/A	0.97	N/A	1.00	N/A	1.00
NIMSIII YAHOO2010 Test data sets						
Non-SKYPE	0.115	N/A	0.121	N/A	0.02	N/A
SKYPE	N/A	0.885	N/A	0.879	N/A	0.98
NIMSIII RADIO2010 Test data sets						
Non-SKYPE	0.386	N/A	0.308	N/A	0.724	N/A
SKYPE	N/A	0.614	N/A	0.692	N/A	0.276
NIMSIII TORRENT2010 Test data sets						
Non-SKYPE	0.653	N/A	0.793	N/A	0.267	N/A
SKYPE	N/A	0.347	N/A	0.207	N/A	0.733
NIMSIII TV2010 Test data sets						
Non-SKYPE	0.818	N/A	0.844	N/A	0.588	N/A
SKYPE	N/A	0.182	N/A	0.156	N/A	0.412
NIMSIII VPN2010 Test data sets						
Non-SKYPE	0.983	N/A	0.985	N/A	0.000	N/A
SKYPE	N/A	0.017	N/A	0.015	N/A	1.00

Table 5.5: Best Model Results on Testing for the Packet Header-based Feature Set for Skype Detection (IPv6 Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
IPv6 2000 Test data sets						
Non-SKYPE	0.436	N/A	0.192	N/A	0.153	N/A
SKYPE	N/A	0.564	N/A	0.808	N/A	0.847
IPv6 2009 Test data sets						
Non-SKYPE	0.327	N/A	0.144	N/A	0.18	N/A
SKYPE	N/A	0.673	N/A	0.856	N/A	0.82

5.2 Results of Flow Experiments for Skype Identification – Robsutness of Signatures

In this set of experiments, the objective is to identify Skype on a flow by flow basis using only the set of features given in Table 3.2. The training data set is generated by sampling randomly with a uniform probability of 6% from both classes (in-class and out-class). Moreover, since there were no Skype TCP flows in Univ2007, 6% of the ITALY (Skype TCPE2X) traffic was sampled. In total, the training data set consists of 1,739,588 flows.

Figure 5.5 summarizes solutions for the three ML algorithms on the training trace: all model construction takes place on the Univ2007 training partition. Testing evaluation was conducted under the Univ2007 Test partition, Univ2010, ITALY, NIMSII, NIMSIII and IPv6 traces where none was encountered during training. Naturally, the Univ2007 Test partition will reflect the training behavior more closely than the other test network traces.

Results presented in Figure 5.5 and the one-way ANOVA statistical analysis test in Appendix B.3.2 illustrates that the C5.0-based classification approach is much better than other algorithms employed in identifying the Skype flow traffic based on the training data set. The violin plot demonstrates the diversity of performance in terms of DR and FPR for all fifty models on the training data sets for each ML algorithm (Figure 5.5). On average, C5.0 is much better than other ML algorithms on the test data sets in terms of a high DR and low FPR.

To select the best trained model for each ML algorithm the training performance was plotted using the scatter plot for each of the three ML algorithms. Figures 5.6, 5.7 and 5.8 summarize the solutions. For Skype, there are four solutions which are non-dominated for GP, four which are non-dominated for AdaBoost and seven which are non-dominated for C5.0. The best performing solution in terms of high DR and low FPR was selected out of these non-dominated solutions for GP, AdaBoost and C5.0 and then evaluated on the test data sets.

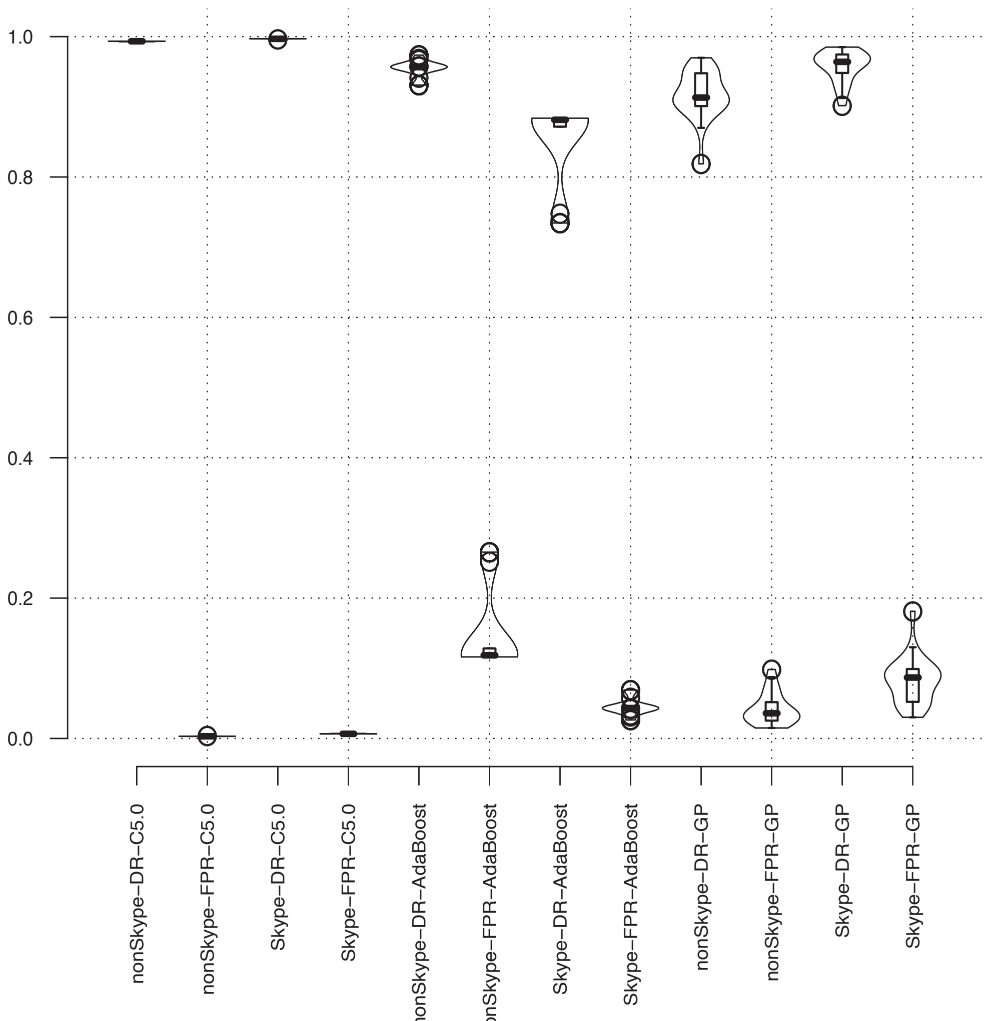


Figure 5.5: DR and FPR Results for Skype Flow Traffic on the Training Data Set

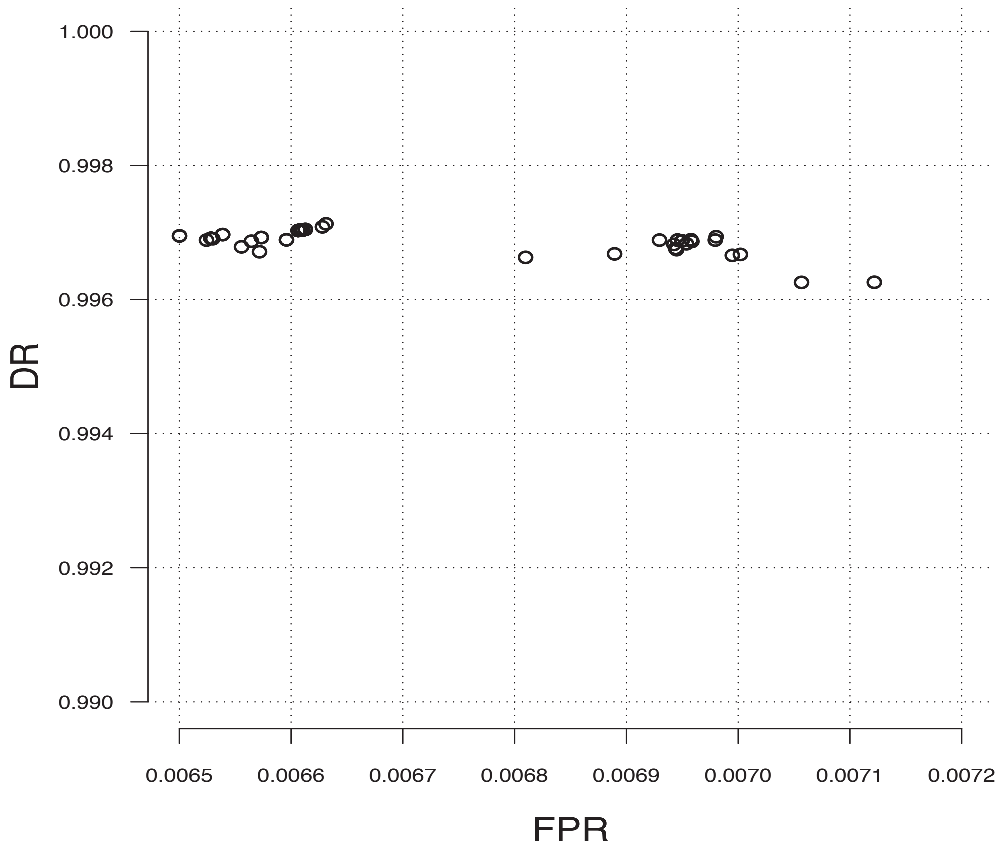


Figure 5.6: Scatter Plot for the C5.0 Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR)

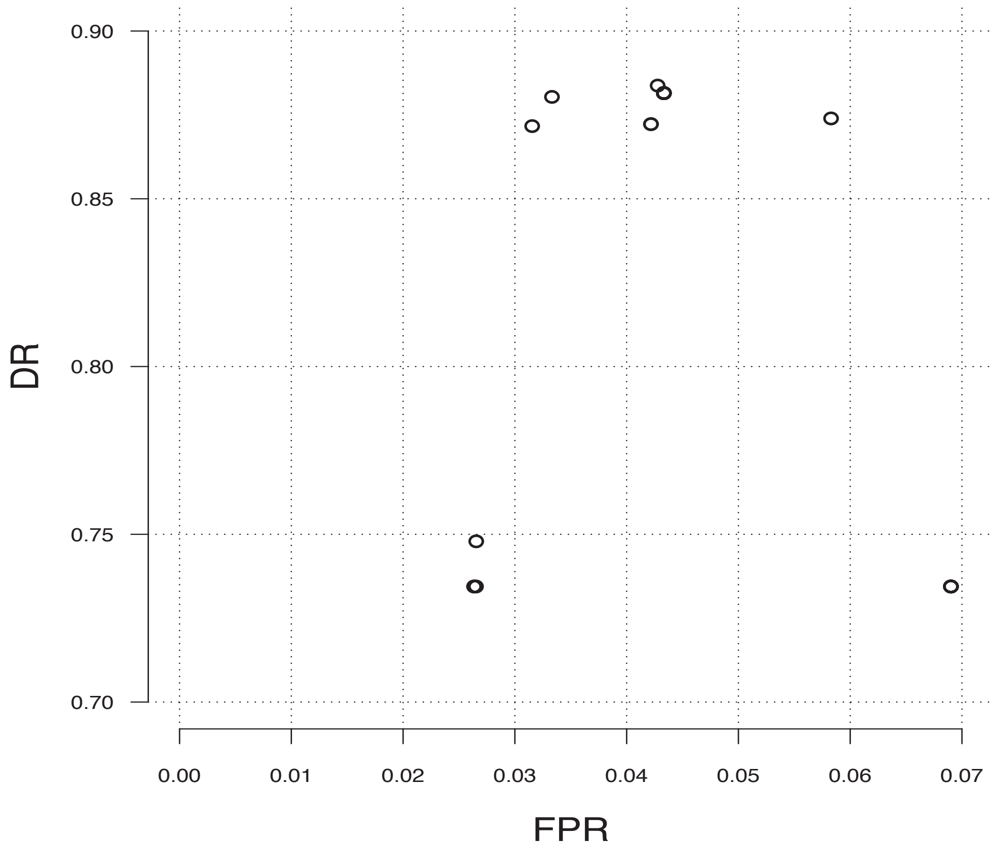


Figure 5.7: Scatter Plot for the AdaBoost Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR)

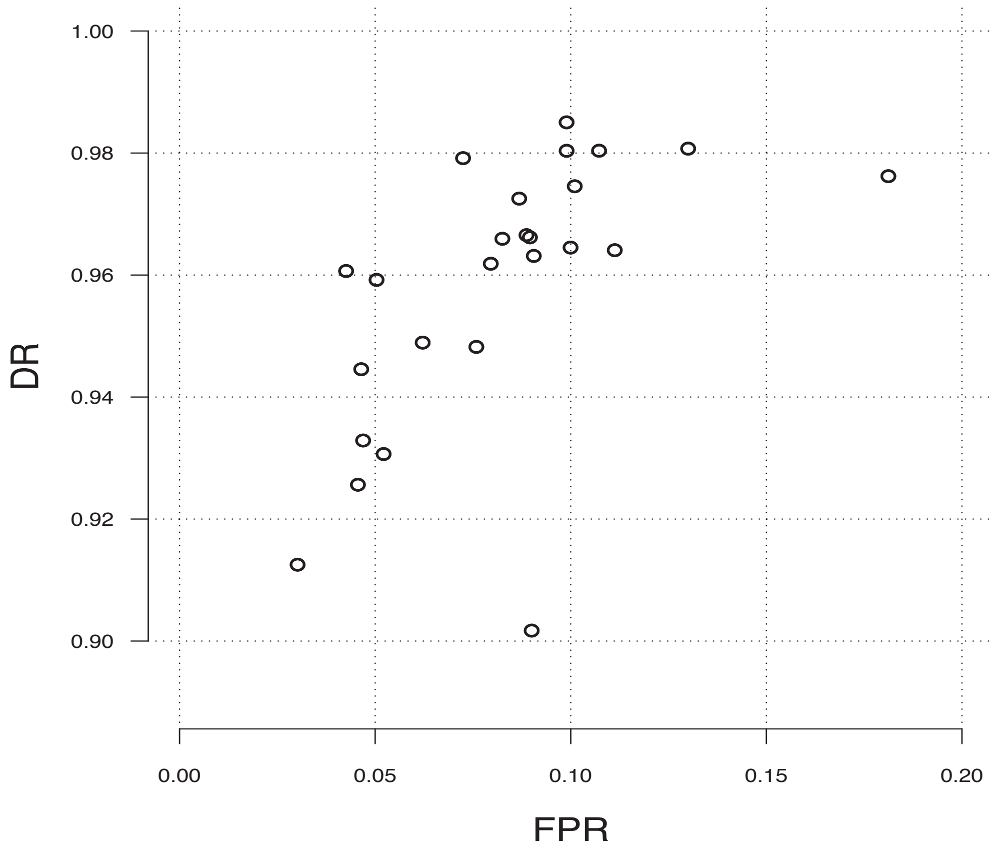


Figure 5.8: Scatter Plot for the GP Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR)

The results are summarized in terms of accuracy. Tables 5.6, 5.7, 5.8, 5.9 and 5.10 list the results for the three ML algorithms on the training and independent test traces. For Skype flows, results show that the C5.0-based classifier performs better than the other classifiers on the Univ2007 Test partition while the GP-based classifier performs better than all the others on the Univ2010 test traces. The C5.0-based classifier achieves ≈100% DR and ≈1% FPR on the Univ2007 Test partition and ≈80% DR and ≈6% FPR on the Univ2010 traces. The GP-based classifier achieves ≈98% DR and ≈10% FPR on the Univ2007 Test partition and ≈86% DR and ≈9% FPR on the Univ2010 traces (Table 5.6). By introducing the entirely independent test sets in terms of location, time and network infrastructure (the Univ2010, ITALY, NIMSII & III and IPv6 traces), the performance of the classifiers were evaluated from a different perspective. In this case, as the results in Table 5.7 show, AdaBoost had overlearned the properties implicit in the training partition since the performance of the AdaBoost-based classifier dropped to a 0% DR on the ITALY traces. Moreover, C5.0 and GP based classifiers were observed to provide good performances on both the Univ2010 and ITALY traces, whereas the AdaBoost and the C5.0 based classifiers achieved very good performances on the NIMSII, NIMSIII and IPv6 test data sets.

The C5.0-based classifier achieved ≈99% DR on the ITALY-TCPE2X traces, ≈89% DR on the ITALY-UDPE2E traces, ≈53% DR on the ITALY-UDPE2O traces and ≈92% on the ITALY-UDPSIG whereas GP achieves ≈71% DR on the ITALY-TCPE2X traces, ≈61% DR on the ITALY-UDPE2E traces, ≈94% DR on the ITALY-UDPE2O traces and ≈61% on the ITALY-UDPSIG (Table 5.7). However, AdaBoost performs better than C5.0 and GP on the NIMS test traces and the IPv6 traces (Tables 5.8, 5.9 and 5.10).

Table 5.6: Best Model Results on Training Data Set for the Flow-based Feature Set for Skype Detection (Training and Testing on Dalhousie Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample (subset of Univ2007)						
Non-SKYPE	0.993	0.003	0.957	0.118	0.901	0.020
SKYPE	0.997	0.007	0.882	0.043	0.980	0.099
Univ2007 Test data sets						
Non-SKYPE	0.993	0.004	0.957	0.117	0.901	0.019
SKYPE	0.996	0.007	0.883	0.043	0.981	0.099
Univ2010 Test data sets						
Non-SKYPE	0.939	0.203	0.921	0.184	0.907	0.140
SKYPE	0.797	0.061	0.816	0.079	0.860	0.093

Table 5.7: Best Model Results on Testing for the Flow-based Feature Set for Skype Detection (ITALY Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
ITALY TCPE2X Test data sets						
Non-SKYPE	N/A	0.014	N/A	1.00	N/A	0.293
SKYPE	0.986	N/A	0.000	N/A	0.707	N/A
ITALY UDPE2E Test data sets						
Non-SKYPE	N/A	0.113	N/A	0.794	N/A	0.387
SKYPE	0.887	N/A	0.206	N/A	0.613	N/A
ITALY UDPE2O Test data sets						
Non-SKYPE	N/A	0.470	N/A	0.976	N/A	0.065
SKYPE	0.530	N/A	0.024	N/A	0.935	N/A
ITALY UDPSIG Test data sets						
Non-SKYPE	N/A	0.079	N/A	0.302	N/A	0.388
SKYPE	0.921	N/A	0.698	N/A	0.612	N/A

Table 5.8: Best Model Results on Testing for the Flow-based Feature Set for Skype Detection (NIMSII Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
NIMSII GTALK2009 Test data sets						
Non-SKYPE	0.787	N/A	0.999	N/A	0.751	N/A
SKYPE	N/A	0.213	N/A	0.001	N/A	0.249
NIMSII PRIMUS2009 Test data sets						
Non-SKYPE	1.00	N/A	1.00	N/A	0.820	N/A
SKYPE	N/A	0.000	N/A	0.000	N/A	0.180
NIMSII ZFONE2009 Test data sets						
Non-SKYPE	0.962	N/A	0.938	N/A	0.810	N/A
SKYPE	N/A	0.038	N/A	0.062	N/A	0.190

Table 5.9: Best Model Results on Testing for the Flow-based Feature Set for Skype Detection (NIMSIII Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
NIMSIII GTALK2010 Test data sets						
Non-SKYPE	0.809	N/A	0.998	N/A	0.464	N/A
SKYPE	N/A	0.191	N/A	0.002	N/A	0.536
NIMSIII PRIMUS2010 Test data sets						
Non-SKYPE	0.998	N/A	0.650	N/A	0.220	N/A
SKYPE	N/A	0.002	N/A	0.350	N/A	0.780
NIMSIII YAHOO2010 Test data sets						
Non-SKYPE	0.927	N/A	0.931	N/A	0.581	N/A
SKYPE	N/A	0.073	N/A	0.069	N/A	0.419
NIMSIII RADIO2010 Test data sets						
Non-SKYPE	0.997	N/A	0.997	N/A	0.997	N/A
SKYPE	N/A	0.003	N/A	0.003	N/A	0.003
NIMSIII TORRENT2010 Test data sets						
Non-SKYPE	0.952	N/A	0.927	N/A	0.821	N/A
SKYPE	N/A	0.048	N/A	0.073	N/A	0.179
NIMSIII TV2010 Test data sets						
Non-SKYPE	0.989	N/A	1.00	N/A	0.955	N/A
SKYPE	N/A	0.011	N/A	0.000	N/A	0.045
NIMSIII VPN2010 Test data sets						
Non-SKYPE	0.987	N/A	1.00	N/A	0.595	N/A
SKYPE	N/A	0.013	N/A	0.000	N/A	0.405

Table 5.10: Best Model Results on Testing for the Flow-based Feature Set for Skype Detection (IPv6 Data Sets)

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
IPv6 2000 Test data sets						
Non-SKYPE	0.999	N/A	1.00	N/A	0.136	N/A
SKYPE	N/A	0.001	N/A	0.000	N/A	0.864
IPv6 2009 Test data sets						
Non-SKYPE	0.889	N/A	1.00	N/A	0.875	N/A
SKYPE	N/A	0.111	N/A	0.000	N/A	0.125

5.3 Discussion of Results

In this chapter, the robustness of the solutions generated automatically by the AdaBoost, GP and C5.0 based classifiers were explored for identifying encrypted VoIP traffic, specifically Skype, in a given traffic trace. To do so, traffic traces from Dalhousie University, NIMS lab, WIDE (IPv6) and ITALY repositories were employed. Furthermore, the classification based approach can be employed with either the packet header only feature set or the flow feature set. Both feature sets were investigated on the traces employed and have showed that the flow-based features performed better than the packet header-based features. It should be noted here that the classification based approach does not use IP addresses, port numbers and payload data.

The flow results demonstrate that the C5.0 classifier is the most consistent performer across all test and training conditions while being competitive with AdaBoost using the NIMS and IPv6 traces. This shows not only that the model which the C5.0 classifier learned during training is robust (generalized) enough to be tested on real world network traces, but also verifies that accurate differentiation between Skype and non-Skype traffic is possible without employing port numbers, IP addresses and payload information. These results demonstrate as well that to achieve high detection and low false positive rates, temporal information is necessary in the case of Skype. The packet header-based feature set performs well only on the traces in which the training data set and test data set are from the same network. This suggests that the packet header-based classifier is more suitable for use on the same or similar network infrastructures while the flow-based classifier performs well on different traces. This suggests that the flow-based classifier is more suitable for working robustly on traces from different networks or network infrastructures. This is the most important difference between the flow-based features and the single packet header-based features. In short, these results suggest that the flow-based classification system trained on data from one network can be employed to run on a different network without new training.

Figures 5.9 and 5.10 list the performance of the three classifiers on traces which contain only Skype traffic from different time periods. The time periods range from 2006 to 2010. It is clear that the C5.0-based classifier has the best performance followed by the GP-based classifier. It should be noted here that the training data set was selected from the Univ2007 trace. The signatures found by C5.0 were able to achieve 87% DR on traces, which were captured a year earlier (the ITALY traces) and 86% DR on traces which were captured three years later. Even though Skype has changed updates over the years, from version 2.5 in 2006 [55] to version 5.0 in 2010 [107], the C5.0-based classifier was able to identify Skype with a high DR and low FPR. Thus, the C5.0-based signatures can generalize well from one network to another and from different time periods as well as different locations. These results show that signatures based on flow feature sets can adapt to the changing conditions of the applications and networks and therefore, they can generalize well, i.e. they are robust.

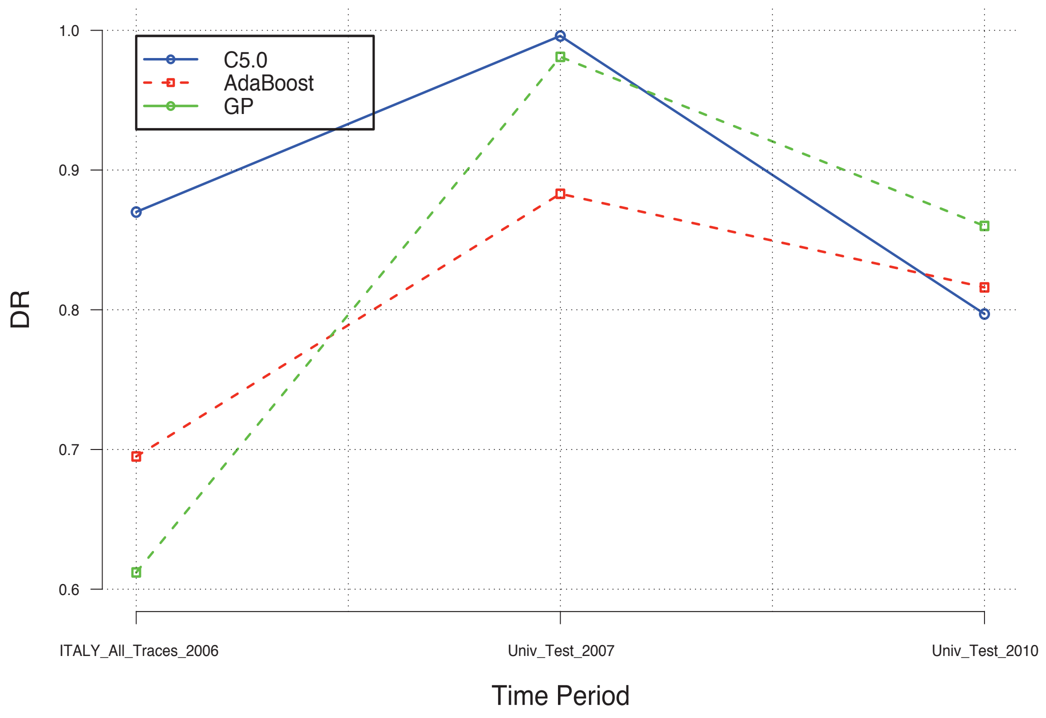


Figure 5.9: Change in the DR for the Three Classifiers Using a Flow-based Feature Set for Skype Detection on the Test Traces Containing Skype Traffic

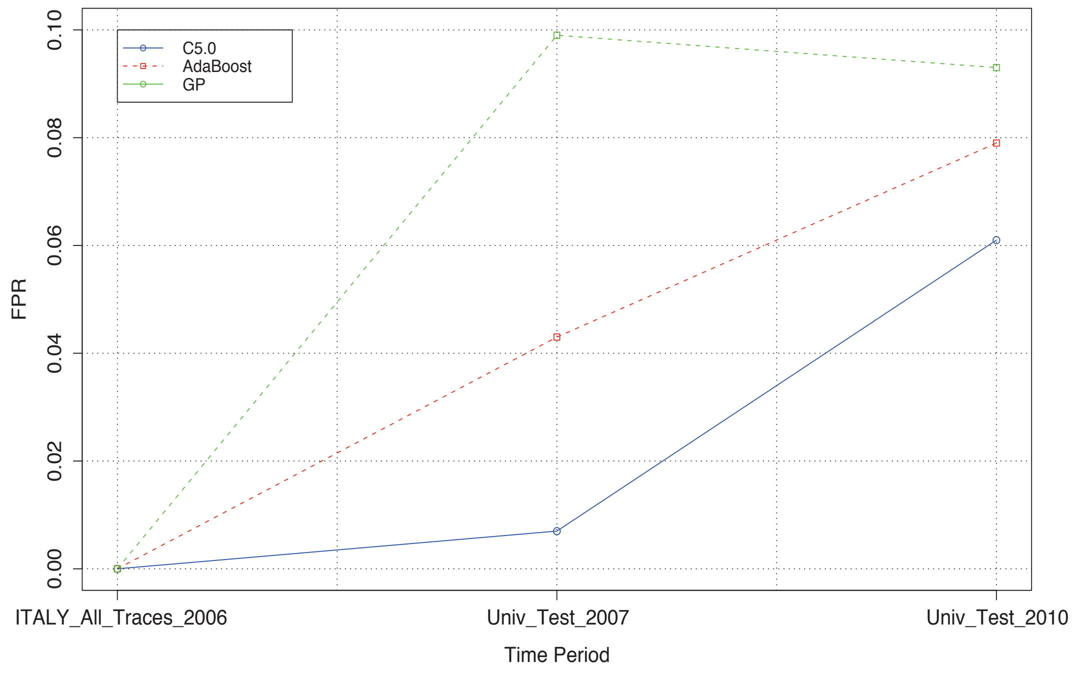


Figure 5.10: Change in the FPR for the Three Classifiers Using a Flow-based Feature Set for Skype Detection on Test Traces Containing Skype Traffic

Chapter 6

Evasion of the Robust Signatures

In this chapter, the robustness/generalization of the proposed approach using ML algorithms with a flow-based feature set is examined by evaluating it against potential evasion attacks, which could circumvent the traffic classifier. It is believed that this is the first time the robustness/generalization of such an approach has been investigated against evasion attacks for VoIP traffic classification, specifically for Skype. To this end, the data sets employed and the experiments performed will be described followed by a discussion of the results.

6.1 Altering Skype Traffic – Evasion Attacks

In this section, the performance of the signatures generated by C5.0, AdaBoost and GP against the padding of the Skype VoIP network traffic is investigated. The same speech corpus audio files used in the generation of the NIMS Lab traces was used. The Speex encoder [108] was used to change the bit rate of the audio file from 8Hz (the original bit rate of the audio file employed) to a range from 5Hz to 14Hz as well as using a sound file convertor [109] to change the formatting type of the audio file. The audio formatting files employed were Macintosh sound formats (AIF and AIFF), a Sun Microsystems file format (AU) and MPEG-2 Audio Layer III (MP3). Table 6.1 lists the properties of the modified audio files.

In these experiments, the impacts of two different approaches for altering the audio on the transmission of the Skype VoIP calls were investigated. The network setup used was the same as with the NIMSII Gtalk experiments in section 3.6.3. A Skype client was installed on the Windows XP machines. The general call setup was the modified audio file played for ten minutes and then the output of the Windows media player was used as input for the Skype client using a microphone (Figure 6.1). Wireshark was used for capturing the traffic on both ends of the Skype communication. These traffic traces (both altered and unaltered Skype traffic) were mixed with the lab traces

Table 6.1: Statistical Overview of Altered-Skype Audio Files

Audio File	**Duration**	**Size in Bytes**	**Number of Flows**
Original (8Hz)	49s	791kb	2,396
Altering Bitrate (5Hz)	1:18s	791KB	3,188
Altering Bitrate (6Hz)	1:05s	791KB	2,863
Altering Bitrate (7Hz)	56s	791KB	2,324
Altering Bitrate (9Hz)	43s	791KB	3,494
Altering Bitrate (10Hz)	39s	791KB	3,605
Altering Bitrate (11Hz)	35s	791KB	2,478
Altering Bitrate (12Hz)	32s	791KB	2,821
Altering Bitrate (13Hz)	30s	791KB	3,235
Altering Bitrate (14Hz)	28s	791KB	3,123
Altering format (AIF)	49s	791KB	4,937
Altering format (AIFF)	49s	791KB	3,097
Altering format (AU)	49s	8.7MB	2,476
Altering format (MP3)	49s	795KB	2,041

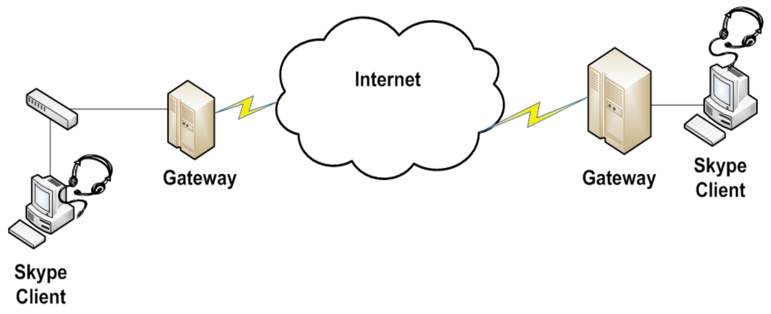


Figure 6.1: Call Setup for Skype Evasion Attacks

described in sections 3.6.3 and 3.6.4.

Figure 6.2 plots the audio spectrum of four audio files. These files (from top to bottom of the figure) are the altered 6Hz Bitrate audio file, the original audio file, the altered 9Hz Bitrate audio file and the altered 14Hz Bitrate audio file. The original audio file was an 8Hz female voice audio file. The orange rectangle boxes on the altered signals indicate the differences between the altered data and the original data caused by the padding, the method used for the evasion attacks. As can be seen from these plots, the padding led to the production of different outputs, but it is very subtle. Furthermore, the altering of the original audio files changed the number of flows Skype generated when these audio files were employed in the Skype voice chat

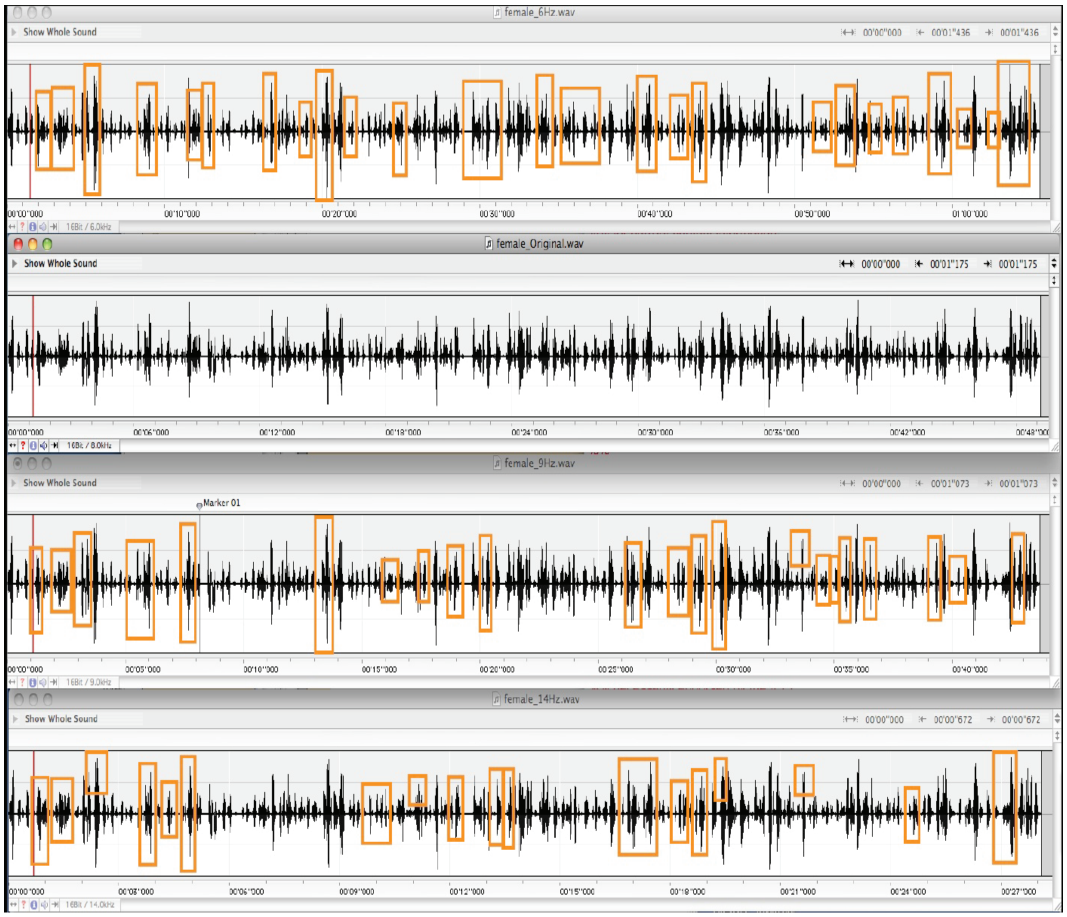


Figure 6.2: Spectrum Analyzer Displays of Audio Signals

service (Table 6.1). However, the altered Skype flows look similar to the Lab traffic. Figures 6.3 and 6.4 show the success of the evasion attacks in changing the Skype traffic into other similar application traffic.

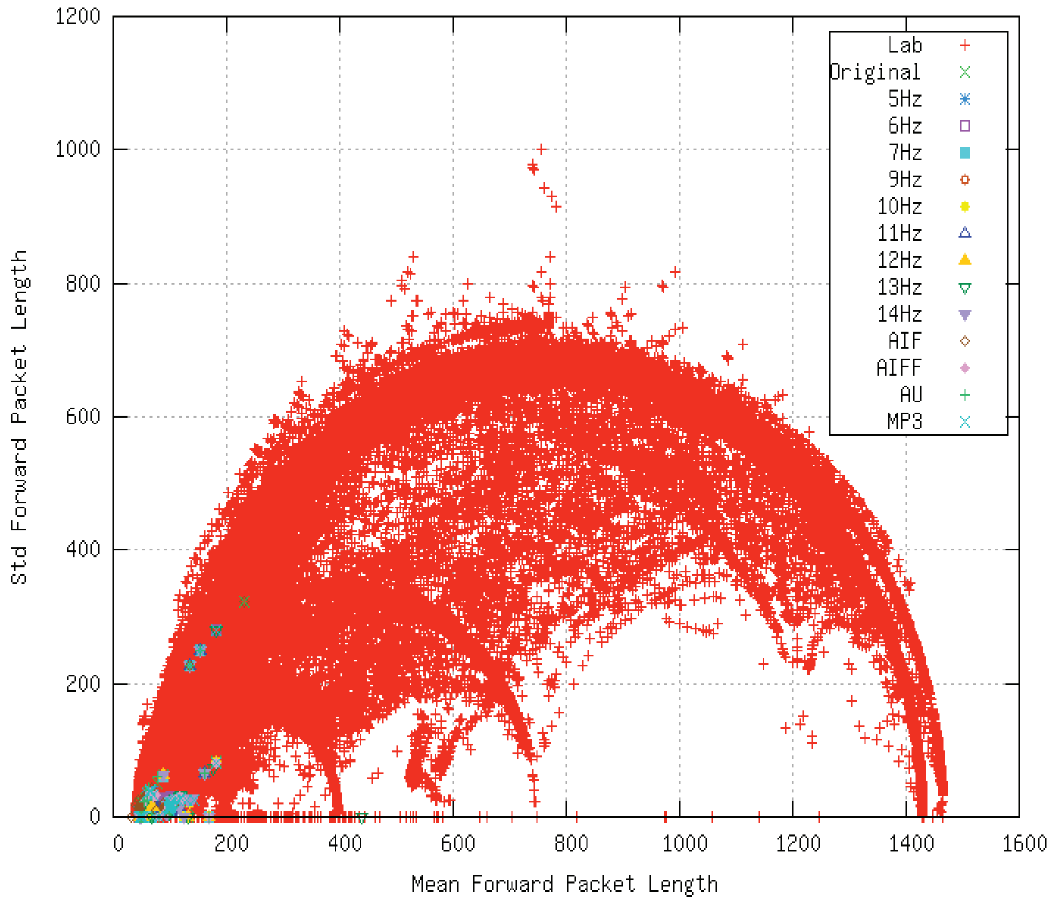


Figure 6.3: Forward Direction of Mean of Packet Length vs. Std. of Packet Length for NIMS Traces and Altered-Skype Traces

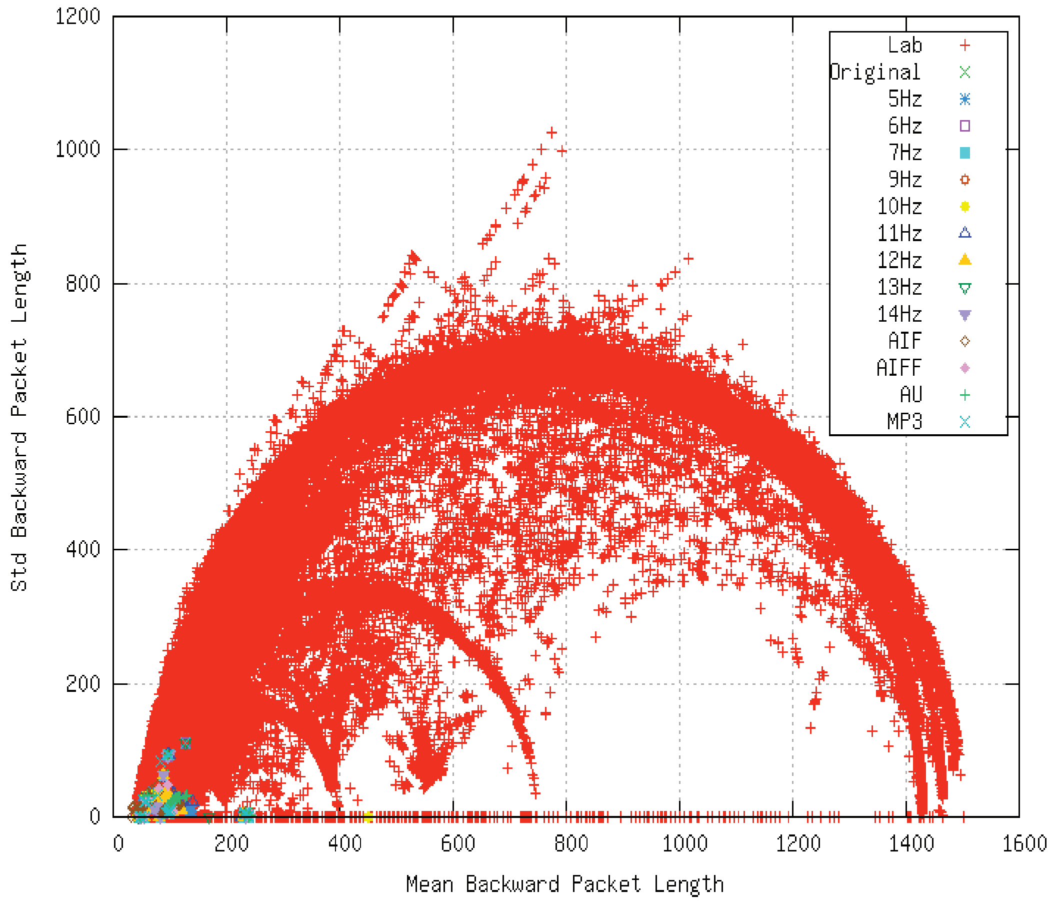


Figure 6.4: Backward Direction of Mean of Packet Length vs. Std. of Packet Length for NIMS Traces and Altered-Skype Traces

6.2 Evaluation of Evasion Attacks

The objective here is to evaluate the effectiveness of the C5.0, AdaBoost and GP based classifiers (Flow-based models) trained on the Univ2007 training data set employed in Chapter 5, section 5.2 on both the original Skype traffic (not altered) traces and the evaded Skype traffic (altered) traces. As discussed earlier, both of these traces were generated in the NIMS lab. Figures 6.5, 6.6 and 6.7 show DR and FPR for C5.0, AdaBoost and GP based classifiers both on the Skype original flows mixed with the NIMS traces and the Skype altered flows mixed with the NIMS traces. The reason the NIMS traces were used as the out-class was to see the performance of the signatures on a data set in which there were additional VoIP applications as well as other encrypted and non-encrypted applications. Moreover, the same network infrastructures were used to generate the NIMS traces and the Skype altered traffic.

Figure 6.5 represents the confusion matrix of the C5.0 classifier on the Skype and Skype altered traffic using a circus graph. The inner circle represents TP, FP, TN, and FN performances. The circle above it contains four segments. These segments represent the classes (Skype and Altered-Skype) and the dominant performance metric. Starting clockwise from six o'clock to nine o'clock represents the performance of Skype in cyan, nine o'clock to twelve o'clock represents the performance of Altered-Skype in red, twelve o'clock to six o'clock represents the TP rate and TN rate in pink and blue, respectively. The TP and TN rates are in this inner circle since these two rates dominate the other rates, which are FP and FN rates. Distinct ribbons determine the ratio layout between the classes and the performance metric. The ribbon represents a quantitative measurement, where a larger ribbon means a higher value and smaller ribbon means a lower value. The FP rate is represented by the green color while the FN rate is represented by the yellow color. The outer two circles provide the value for the TP, FN, TN and FP rates. The C5.0-based classifier has a very high performance based on the size of the ribbons starting from the TP and TN segments for both the Skype and Altered-Skype classes. For instance, there are two wide cyan ribbons (Skype) coming from the pink TP segment and the blue TN segment, while there are two tiny cyan ribbons coming from the green FP segment and the yellow FN segment. The same behaviour can be seen for the Altered-Skype class. In other words, C5.0-based signatures show very promising performance for original Skype traffic (the cyan color in Figure 6.5) with ≈91% DR and ≈5% FPR for in-class traffic (Skype) and ≈95% DR and ≈9% FPR for out-class traffic (non-Skype). For Altered-Skype traffic (the red color in Figure 6.5), the C5.0-based classifier achieved ≈85% DR and ≈5% FPR for in-class traffic (Altered-Skype) and ≈95% DR and ≈15% FPR for out-class traffic (non-Skype).

Figure 6.6 shows the performance of the AdaBoost-based classifier. Again, the ribbons starting from the TP and TN segments are wide for both classes, whereas the ribbons starting from the FP and FN segments are very narrow which implies that the percentages of FP and FN are very small. The AdaBoost-based classifier achieves ≈85% DR and ≈6% FPR for in-class traffic (Altered-Skype) and ≈94% DR and ≈15% FPR for out-class traffic (non-Skype). For original Skype traffic (the cyan color in Figure 6.6), the AdaBoost-based classifier achieves ≈91% DR and ≈6% FPR for in-class traffic (Skype) and ≈94% DR and ≈9% FPR for out-class traffic (non-Skype).

In contrast, the sizes of the ribbons starting from the TP and TN segments for the GP-based classifier (Figure 6.7) are narrower (smaller percentage value) compared to the C5.0-based classifier (higher percentage value) in Figure 6.5, and the AdaBoost-based classifier in Figure 6.6. Moreover, the ribbons starting from the FP segments for the Skype and Altered-Skype classes are relatively wide (relatively smaller percentage). The GP-based classifier achieves ≈94% DR and ≈23% FPR for in-class traffic (Skype) and ≈77% DR and ≈6% FPR for out-class traffic (non-Skype). For Altered-Skype traffic (the red color in Figure 6.7), GP achieved ≈85% DR and ≈23% FPR for in-class traffic (Altered-Skype) and ≈77% DR and ≈15% FPR for out-class traffic (non-Skype).

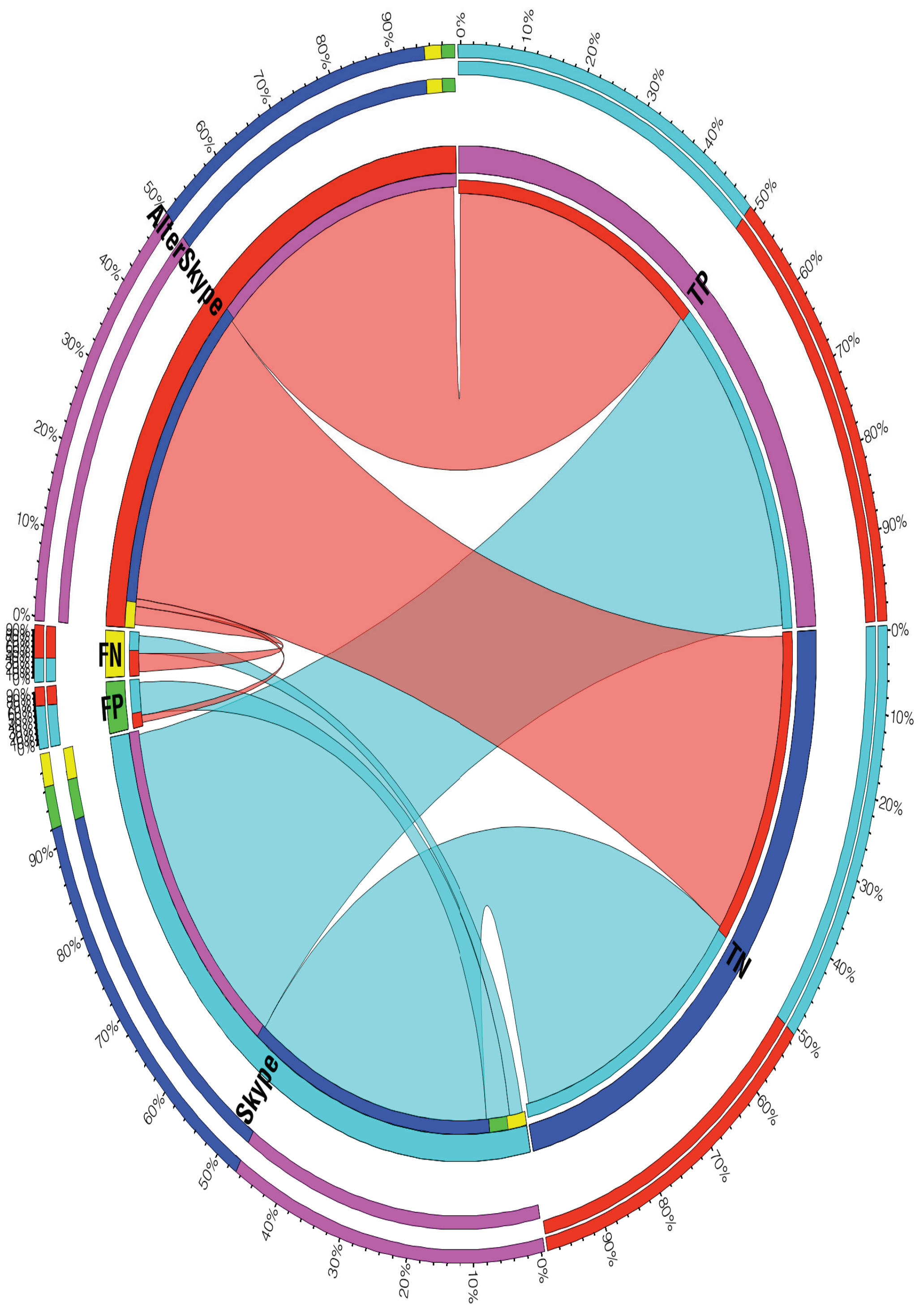


Figure 6.5: Performance of the C5.0-based Signatures Under Evasion Attacks by Using a Circos Graph

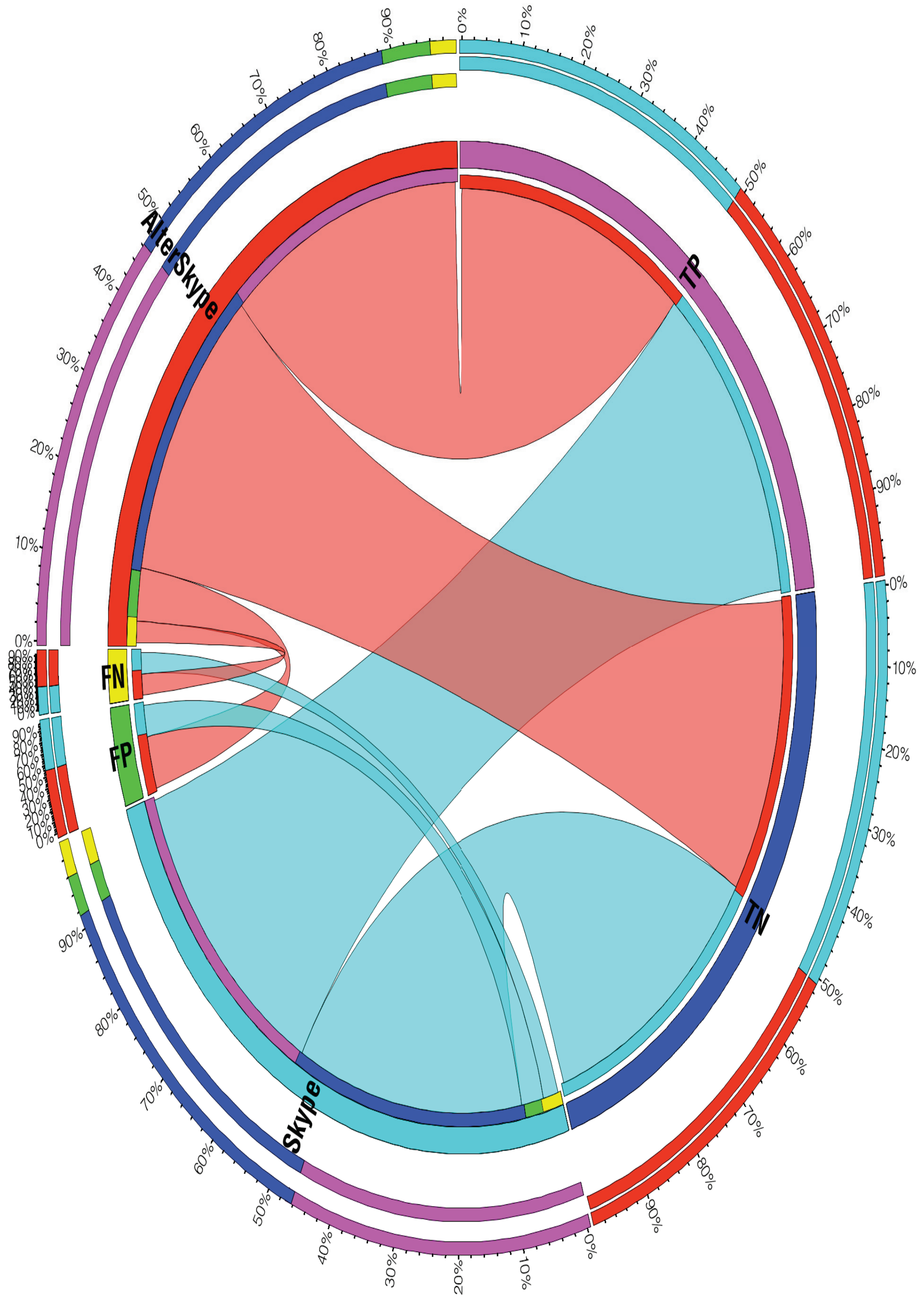


Figure 6.6: Performance of the AdaBoost-based Signatures Under Evasion Attacks by Using a Circos Graph

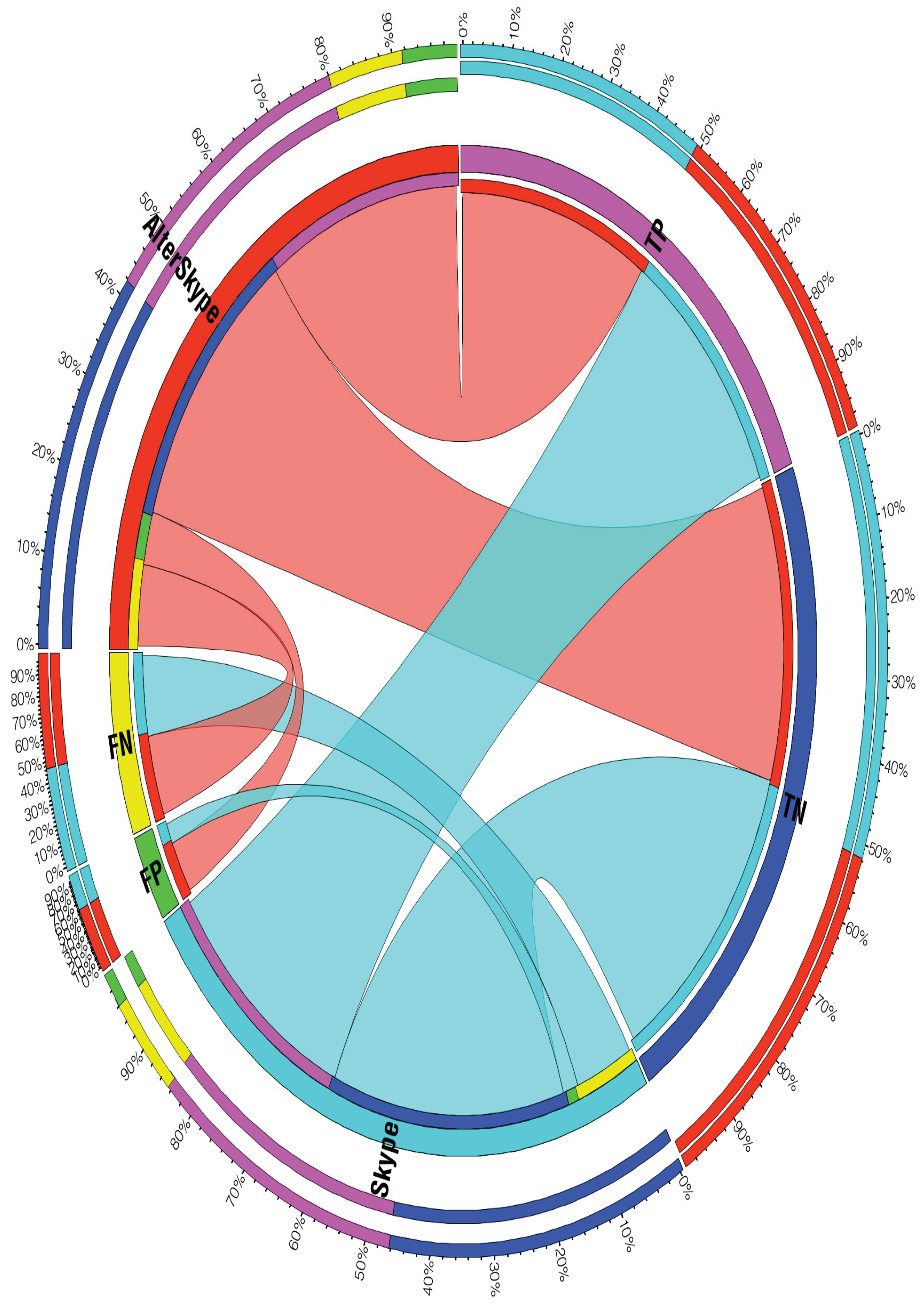


Figure 6.7: Performance of the GP-based Signatures Under Evasion Attacks by Using a Circos Graph

Table 6.2 lists the performance of the three ML-based classifiers on each of the flow data sets generated by altering the audio files. The lowest performance occurred when the audio file bit rate was changed from 8Hz to 9Hz (TP ≈76%). In terms of modifying the encoding format, the MP3 caused the lowest performance (≈82% TP rate for the C5.0-based classifier, ≈82% TP rate for the AdaBoost-based classifier and ≈76% TP rate for the GP-based classifier). Specifically, the GP-based classifier achieved the highest performance on five of the Skype altered traces (9Hz, 10Hz, 11Hz, 14Hz and AIF), while the AdaBoost-based classifier achieved the highest performance on two of the Skype altered traces (6Hz and 13Hz) and C5.0 achieved the highest performance on five of the Skype altered traces (5Hz, 7Hz, 12Hz, AIFF and AU). Both the C5.0 and AdaBoost based classifiers achieved higher performance than the GP-based classifier on the MP3 trace.

Table 6.2: TP Rate for the Three Classifiers on each of the Altered-Skype Traces

	C5.0	**AdaBoost**	**GP**
5Hz bitrate	0.875	0.870	0.841
6Hz bitrate	0.796	0.832	0.807
7Hz bitrate	0.815	0.811	0.768
9Hz bitrate	0.758	0.751	0.769
10Hz bitrate	0.886	0.806	0.888
11Hz bitrate	0.86	0.851	0.890
12Hz bitrate	0.84	0.816	0.832
13Hz bitrate	0.823	0.898	0.859
14Hz bitrate	0.822	0.911	0.921
AIF encoding	0.896	0.869	0.924
AIFF encoding	0.88	0.877	0.827
AU encoding	0.878	0.874	0.840
MP3 encoding	0.816	0.816	0.756

6.2.1 Discussion of Results

The modification of the audio files (Figure 6.2) has a big effect on how Skype transmits the packets. Hence, the characteristic of the flows changes, which makes them more similar to other protocols (Figures 6.3 and 6.4). With the success in padding the Skype payload by altering audio files to mimic the behaviour of other protocols, the performance of the signatures generated by C5.0 dropped by 7% on average (Table 6.2).

Previously published research [74, 75] in the field claimed that evasion attacks against ML-based classifiers and / or statistical features could succeed easily, resulting in performances below 50% (random guessing). However, these experiments indicate that such classifiers are not as easy to evade as claimed before. Indeed, the performance of the traffic classifiers does drop against such attacks but these results do indicate that well chosen training data sets and features can improve the generalization of such learning and data mining techniques even against evasion attacks. The results of this research suggest that the signatures generated by using the three machine learning classifiers do not become ineffective when faced with evasion attacks, i.e. altered Skype flows. Instead, they can still classify, albeit with a 7% decrease in their performance. This suggests that the signatures are robust for classifying Skype traffic regardless of the effect of locations, time periods or padding payload packets. These results are consistent with the performance of the C5.0, AdaBoost and GP based classifiers on the test traces (Chapter 5, section 5.2).

6.3 Evading Public Tools

In this section, the performance of the Deep Packet Inspection tools on the Skype and Altered-Skype traffic used in section 6.3.1 is discussed.

6.3.1 Evading Attacks Against OpenDPI

The publicly available OpenDPI is one of the common DPI tools used to classify traffic, as explained in section 2.2.4. Since OpenDPI requires the packet payload in order to classify the packets, the same traces used to evaluate the robustness of the ML algorithms against evasion attacks in section 6.1 were employed. Both the original Skype traffic generated in the NIMS lab and the altered-Skype traffic traces were used in order to evaluate the performance of the OpenDPI tool.

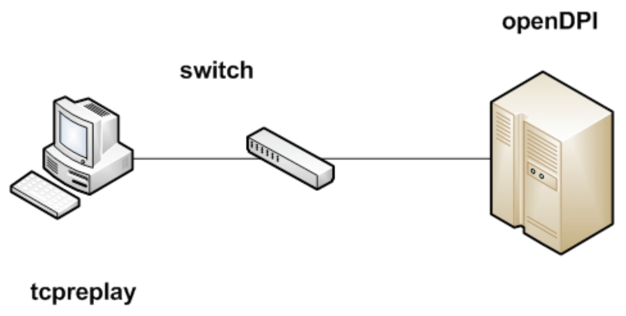


Figure 6.8: OpenDPI Testbed Setup

The network testbed setup for this experiment as shown in Figure 6.8 is self explanatory. Tables 6.3 and 6.4 show the performance of the OpenDPI tool in classifying the traces. As can be seen from these results, OpenDPI failed to classify the Altered-Skype traces. This is due to the fact that the signatures employed by OpenDPI for classifying Skype are not effective on this data set. Also, these results indicate that there exists a need for alternative techniques (such as the one proposed in this thesis) for classifying VoIP traffic.

Table 6.3: OpenDPI Results on the NIMS Traces and Altered-Skype Traces

	DR	**FPR**
Original Skype Traces		
Non-SKYPE	0.01	1
Altered-SKYPE	0.0	0.99
Altered-Skype Traces		
Non-SKYPE	0.01	1
Altered-SKYPE	0.0	0.99

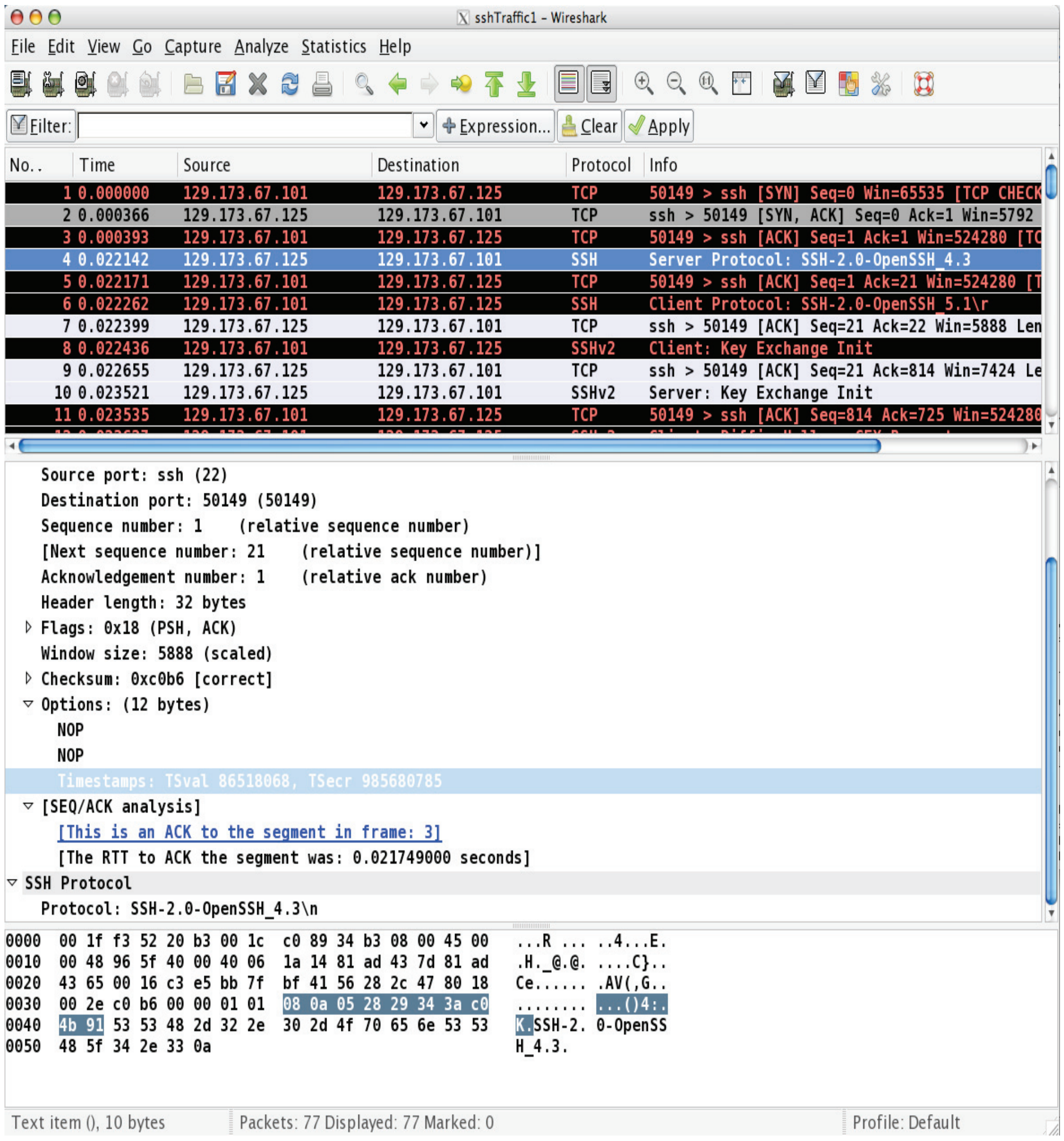
Table 6.4: TP Rate for OpenDPI on each of the Altered-Skype Traces

5Hz bitrate	0.0
6Hz bitrate	0.0
7Hz bitrate	0.0
9Hz bitrate	0.0
10Hz bitrate	0.0
11Hz bitrate	0.0
12Hz bitrate	0.0
13Hz bitrate	0.0
14Hz bitrate	0.0
AIF encoding	0.0
AIFF encoding	0.0
AU encoding	0.0
MP3 encoding	0.0

6.3.2 Evading Attacks Against Wireshark

The purpose of this experiment is to show the effectiveness of Wireshark type traffic analyzers, which inspect all the header (including the IP addresses and TCP/UDP port numbers) as well as the payload information, as traffic classification systems. Since Wireshark can be used as a network analysis tool to label traffic according to application type, the following experiments can show the ability of Wireshark to classify an encrypted application such as SSH. An SSH session was run in a controlled environment in which all SSH traffic was defined to be direct communication between a client machine and an SSH server (hector.cs.dal.ca) connected via the Internet. The client computer connected to the SSH server on port 22. Tcpdump [110] was running to capture traffic on the client machine. Figure 6.9 shows the results of running the Wireshark tool on the captured SSH trace. Indeed, Wireshark was able to label all of the SSH packets.

Furthermore, to demonstrate how Wireshark uses signatures based on port numbers to label SSH traffic, two experiments were run in which the port number in the SSH trace was modified from port 22 to port 2200 in the first experiment and from port 22 to port 80 in the second experiment using tcprewrite [111] (a Unix tool for rewriting packets in a pcap file), and then Wireshark was run again. Figure 6.10 shows the result of the first experiment in which Wireshark failed to detect any of the SSH packets. Figure 6.10 shows the result of the second experiment in which Wireshark classified SSH packets as HTTP packets. Wireshark was run as well on the Original Skype and Altered-Skype traffic. Figures 6.12 and 6.13 show the results in which Wireshark failed to detect any of Skype packets (Original or Altered). These experiments illustrate that Wireshark depends on well-known port numbers for classifying the traffic. Thus, a new approach is necessary, which does not depend on port numbers where the classifier/analyzer cannot be evaded just by changing the port number where the application runs.



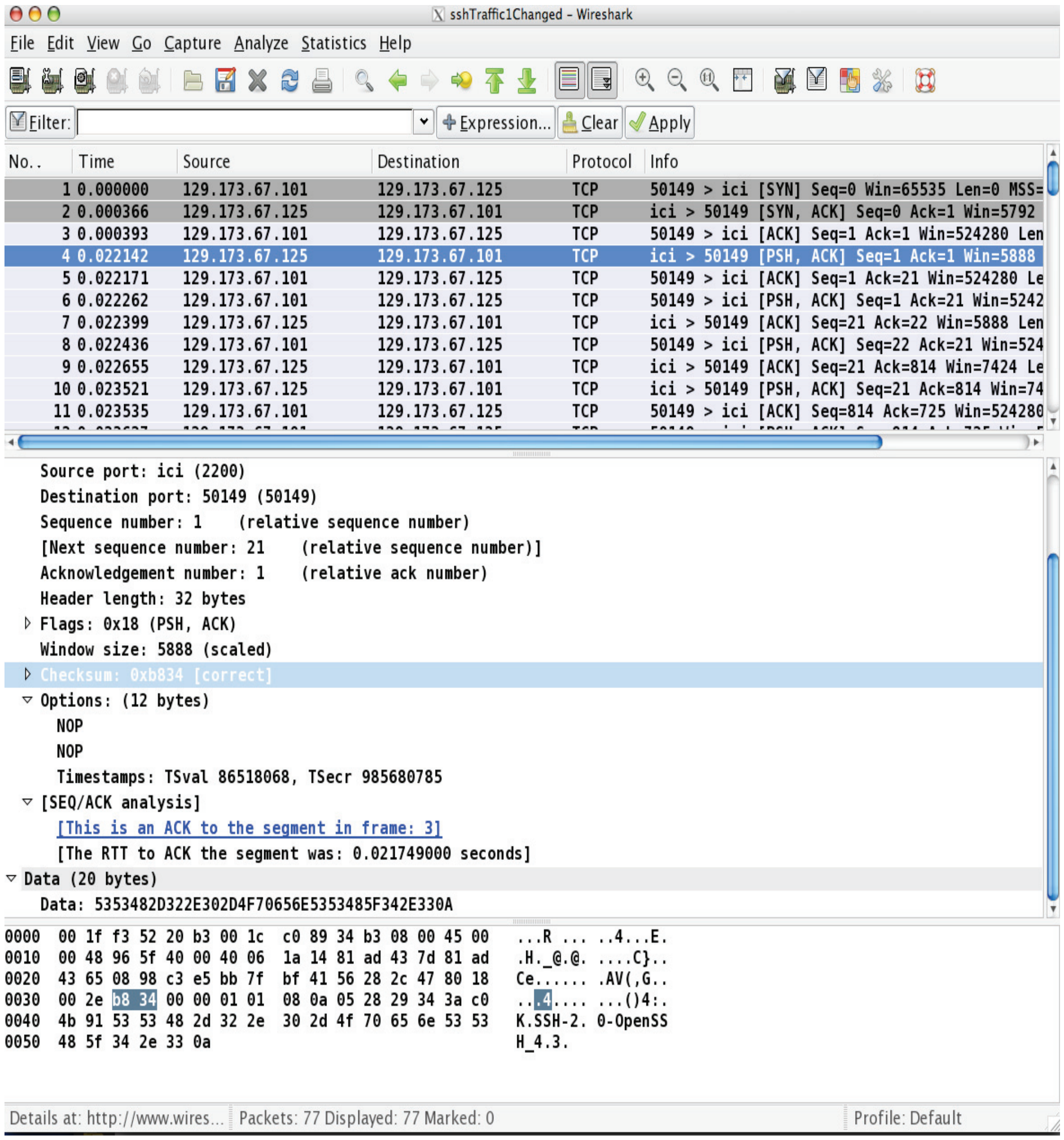
```
No..  Time      Source          Destination     Protocol Info
   1 0.000000  129.173.67.101  129.173.67.125  TCP      50149 > ssh [SYN] Seq=0 Win=65535 [TCP CHECK
   2 0.000366  129.173.67.125  129.173.67.101  TCP      ssh > 50149 [SYN, ACK] Seq=0 Ack=1 Win=5792
   3 0.000393  129.173.67.101  129.173.67.125  TCP      50149 > ssh [ACK] Seq=1 Ack=1 Win=524280 [TC
   4 0.022142  129.173.67.125  129.173.67.101  SSH      Server Protocol: SSH-2.0-OpenSSH_4.3
   5 0.022171  129.173.67.101  129.173.67.125  TCP      50149 > ssh [ACK] Seq=1 Ack=21 Win=524280 [T
   6 0.022262  129.173.67.101  129.173.67.125  SSH      Client Protocol: SSH-2.0-OpenSSH_5.1\r
   7 0.022399  129.173.67.125  129.173.67.101  TCP      ssh > 50149 [ACK] Seq=21 Ack=22 Win=5888 Len
   8 0.022436  129.173.67.101  129.173.67.125  SSHv2    Client: Key Exchange Init
   9 0.022655  129.173.67.125  129.173.67.101  TCP      ssh > 50149 [ACK] Seq=21 Ack=814 Win=7424 Le
  10 0.023521  129.173.67.125  129.173.67.101  SSHv2    Server: Key Exchange Init
  11 0.023535  129.173.67.101  129.173.67.125  TCP      50149 > ssh [ACK] Seq=814 Ack=725 Win=524280
```

```
    Source port: ssh (22)
    Destination port: 50149 (50149)
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 21    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 32 bytes
  ▷ Flags: 0x18 (PSH, ACK)
    Window size: 5888 (scaled)
  ▷ Checksum: 0xc0b6 [correct]
  ▽ Options: (12 bytes)
      NOP
      NOP
      Timestamps: TSval 86518068, TSecr 985680785
  ▽ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 3]
      [The RTT to ACK the segment was: 0.021749000 seconds]
▽ SSH Protocol
    Protocol: SSH-2.0-OpenSSH_4.3\n
```

```
0000  00 1f f3 52 20 b3 00 1c  c0 89 34 b3 08 00 45 00   ...R .... ..4...E.
0010  00 48 96 5f 40 00 40 06  1a 14 81 ad 43 7d 81 ad   .H._@.@. ....C}..
0020  43 65 00 16 c3 e5 bb 7f  bf 41 56 28 2c 47 80 18   Ce...... .AV(,G..
0030  00 2e c0 b6 00 00 01 01  08 0a 05 28 29 34 3a c0   ........ ...()4:.
0040  4b 91 53 53 48 2d 32 2e  30 2d 4f 70 65 6e 53 53   K.SSH-2. 0-OpenSS
0050  48 5f 34 2e 33 0a                                  H_4.3.
```

Text item (), 10 bytes | Packets: 77 Displayed: 77 Marked: 0 | Profile: Default

Figure 6.9: Wireshark Classifying SSH Packets Before Modifying Ports


sshTraffic1Changed - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

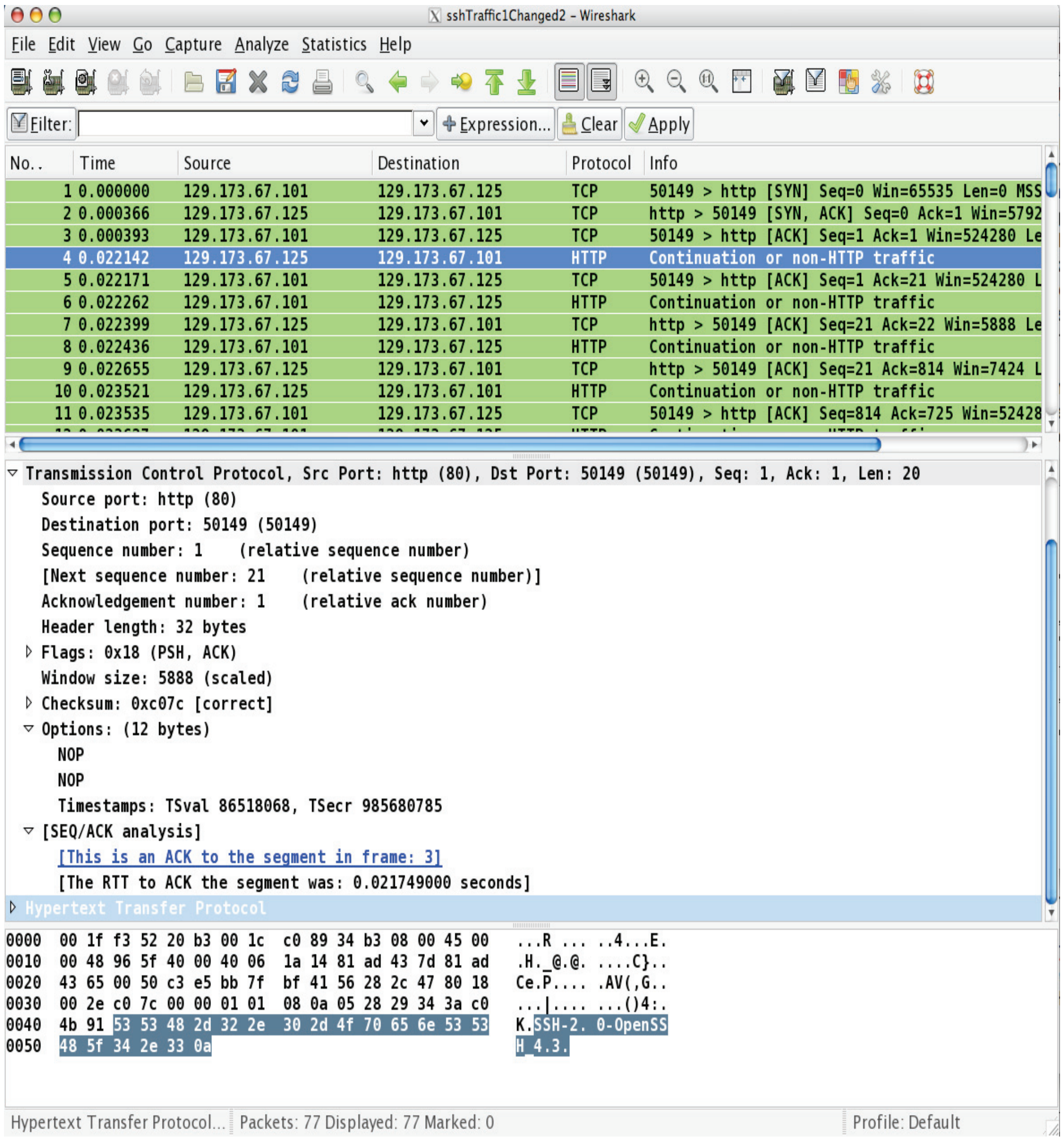
```
No..  Time      Source          Destination     Protocol  Info
   1 0.000000   129.173.67.101  129.173.67.125  TCP       50149 > ici [SYN] Seq=0 Win=65535 Len=0 MSS=
   2 0.000366   129.173.67.125  129.173.67.101  TCP       ici > 50149 [SYN, ACK] Seq=0 Ack=1 Win=5792
   3 0.000393   129.173.67.101  129.173.67.125  TCP       50149 > ici [ACK] Seq=1 Ack=1 Win=524280 Len
   4 0.022142   129.173.67.125  129.173.67.101  TCP       ici > 50149 [PSH, ACK] Seq=1 Ack=1 Win=5888
   5 0.022171   129.173.67.101  129.173.67.125  TCP       50149 > ici [ACK] Seq=1 Ack=21 Win=524280 Le
   6 0.022262   129.173.67.101  129.173.67.125  TCP       50149 > ici [PSH, ACK] Seq=1 Ack=21 Win=5242
   7 0.022399   129.173.67.125  129.173.67.101  TCP       ici > 50149 [ACK] Seq=21 Ack=22 Win=5888 Len
   8 0.022436   129.173.67.101  129.173.67.125  TCP       50149 > ici [PSH, ACK] Seq=22 Ack=21 Win=524
   9 0.022655   129.173.67.125  129.173.67.101  TCP       ici > 50149 [ACK] Seq=21 Ack=814 Win=7424 Le
  10 0.023521   129.173.67.125  129.173.67.101  TCP       ici > 50149 [PSH, ACK] Seq=21 Ack=814 Win=74
  11 0.023535   129.173.67.101  129.173.67.125  TCP       50149 > ici [ACK] Seq=814 Ack=725 Win=524280
```

```
  Source port: ici (2200)
  Destination port: 50149 (50149)
  Sequence number: 1    (relative sequence number)
  [Next sequence number: 21    (relative sequence number)]
  Acknowledgement number: 1    (relative ack number)
  Header length: 32 bytes
▷ Flags: 0x18 (PSH, ACK)
  Window size: 5888 (scaled)
▷ Checksum: 0xb834 [correct]
▽ Options: (12 bytes)
    NOP
    NOP
    Timestamps: TSval 86518068, TSecr 985680785
▽ [SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 3]
    [The RTT to ACK the segment was: 0.021749000 seconds]
▽ Data (20 bytes)
  Data: 5353482D322E302D4F70656E5353485F342E330A
```

```
0000  00 1f f3 52 20 b3 00 1c  c0 89 34 b3 08 00 45 00   ...R .... ..4...E.
0010  00 48 96 5f 40 00 40 06  1a 14 81 ad 43 7d 81 ad   .H._@.@. ....C}..
0020  43 65 08 98 c3 e5 bb 7f  bf 41 56 28 2c 47 80 18   Ce...... .AV(,G..
0030  00 2e b8 34 00 00 01 01  08 0a 05 28 29 34 3a c0   ..4..... ...()4:.
0040  4b 91 53 53 48 2d 32 2e  30 2d 4f 70 65 6e 53 53   K.SSH-2. 0-OpenSS
0050  48 5f 34 2e 33 0a                                  H_4.3.
```

```
Details at: http://www.wires...  Packets: 77 Displayed: 77 Marked: 0  Profile: Default
```

Figure 6.10: Wireshark Misclassify SSH Packets After Modifying Ports


sshTraffic1Changed2 – Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

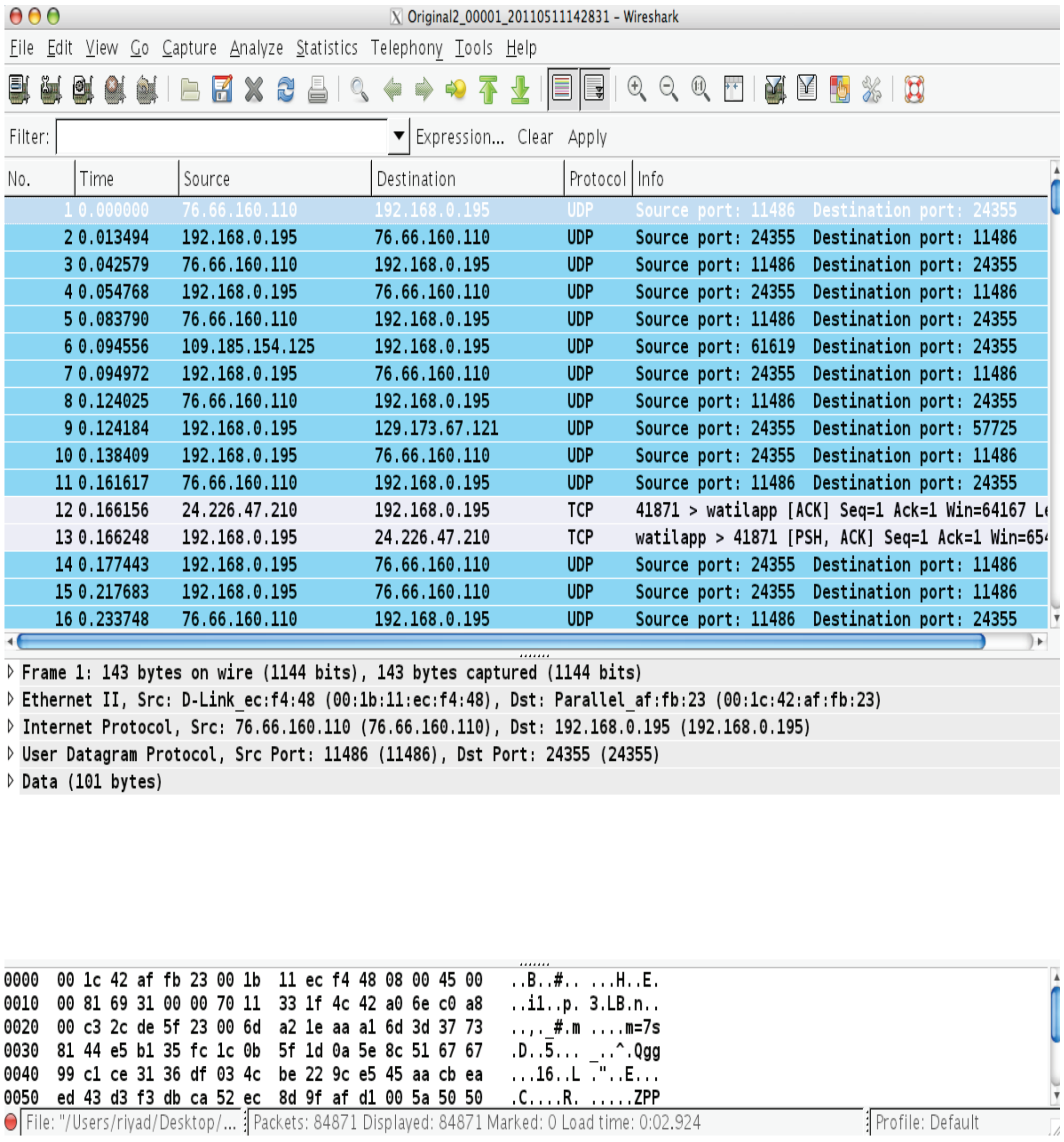
```
No..  Time      Source          Destination     Protocol Info
   1 0.000000   129.173.67.101  129.173.67.125  TCP      50149 > http [SYN] Seq=0 Win=65535 Len=0 MSS
   2 0.000366   129.173.67.125  129.173.67.101  TCP      http > 50149 [SYN, ACK] Seq=0 Ack=1 Win=5792
   3 0.000393   129.173.67.101  129.173.67.125  TCP      50149 > http [ACK] Seq=1 Ack=1 Win=524280 Le
   4 0.022142   129.173.67.125  129.173.67.101  HTTP     Continuation or non-HTTP traffic
   5 0.022171   129.173.67.101  129.173.67.125  TCP      50149 > http [ACK] Seq=1 Ack=21 Win=524280 L
   6 0.022262   129.173.67.101  129.173.67.125  HTTP     Continuation or non-HTTP traffic
   7 0.022399   129.173.67.125  129.173.67.101  TCP      http > 50149 [ACK] Seq=21 Ack=22 Win=5888 Le
   8 0.022436   129.173.67.101  129.173.67.125  HTTP     Continuation or non-HTTP traffic
   9 0.022655   129.173.67.125  129.173.67.101  TCP      http > 50149 [ACK] Seq=21 Ack=814 Win=7424 L
  10 0.023521   129.173.67.125  129.173.67.101  HTTP     Continuation or non-HTTP traffic
  11 0.023535   129.173.67.101  129.173.67.125  TCP      50149 > http [ACK] Seq=814 Ack=725 Win=52428
```

```
▽ Transmission Control Protocol, Src Port: http (80), Dst Port: 50149 (50149), Seq: 1, Ack: 1, Len: 20
    Source port: http (80)
    Destination port: 50149 (50149)
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 21    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 32 bytes
  ▷ Flags: 0x18 (PSH, ACK)
    Window size: 5888 (scaled)
  ▷ Checksum: 0xc07c [correct]
  ▽ Options: (12 bytes)
      NOP
      NOP
      Timestamps: TSval 86518068, TSecr 985680785
  ▽ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 3]
      [The RTT to ACK the segment was: 0.021749000 seconds]
▷ Hypertext Transfer Protocol
```

```
0000  00 1f f3 52 20 b3 00 1c  c0 89 34 b3 08 00 45 00   ...R ... ..4...E.
0010  00 48 96 5f 40 00 40 06  1a 14 81 ad 43 7d 81 ad   .H._@.@. ....C}..
0020  43 65 00 50 c3 e5 bb 7f  bf 41 56 28 2c 47 80 18   Ce.P.... .AV(,G..
0030  00 2e c0 7c 00 00 01 01  08 0a 05 28 29 34 3a c0   ...|.... ...()4:.
0040  4b 91 53 53 48 2d 32 2e  30 2d 4f 70 65 6e 53 53   K.SSH-2. 0-OpenSS
0050  48 5f 34 2e 33 0a                                  H_4.3.
```

Hypertext Transfer Protocol... | Packets: 77 Displayed: 77 Marked: 0 | Profile: Default

Figure 6.11: Wireshark Classified SSH Packets as HTTP Packets After Modifying Ports


Original2_00001_20110511142831 - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

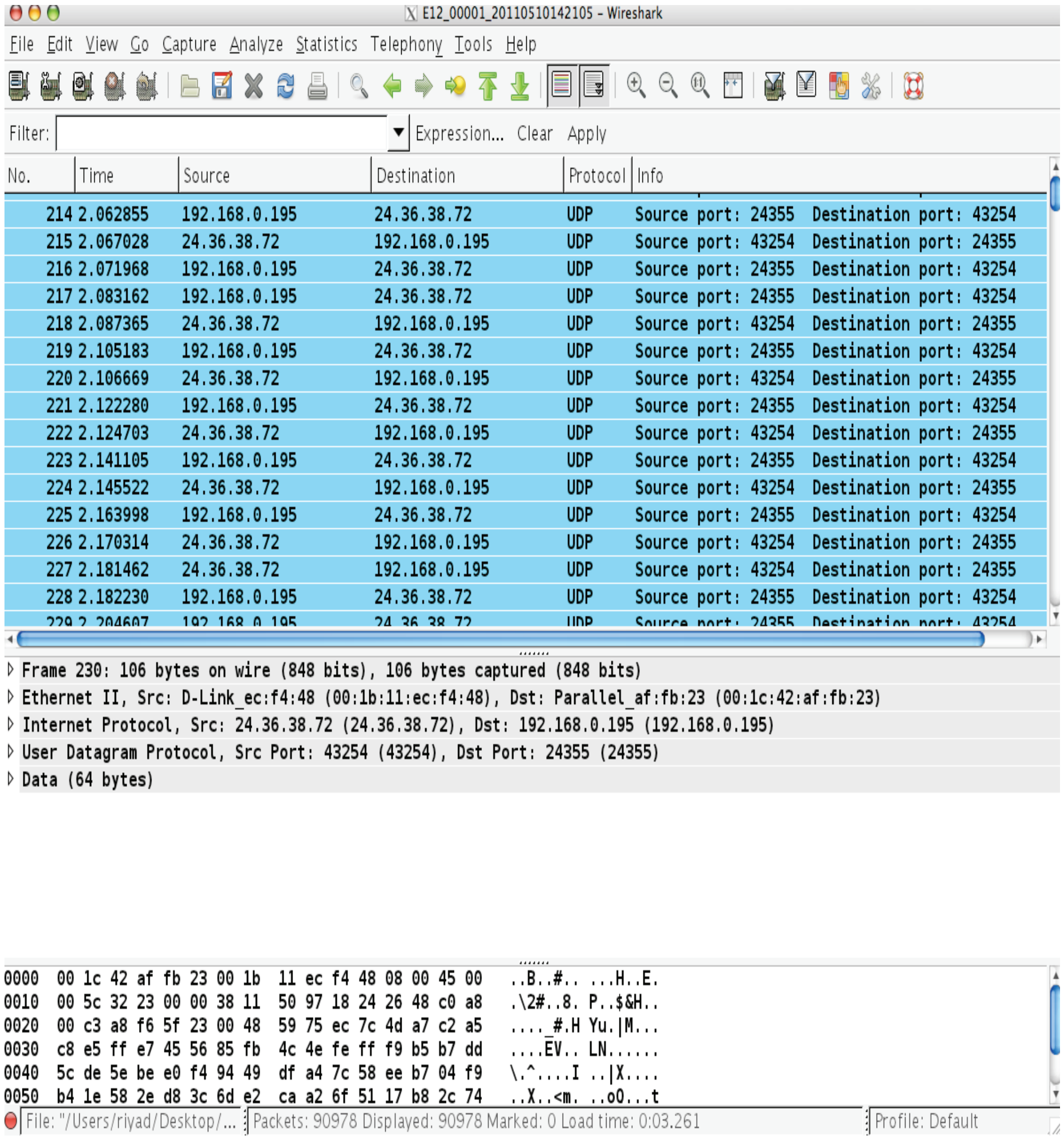
```
No.  Time      Source          Destination     Protocol Info
   1 0.000000  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355
   2 0.013494  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
   3 0.042579  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355
   4 0.054768  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
   5 0.083790  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355
   6 0.094556  109.185.154.125 192.168.0.195   UDP      Source port: 61619  Destination port: 24355
   7 0.094972  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
   8 0.124025  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355
   9 0.124184  192.168.0.195   129.173.67.121  UDP      Source port: 24355  Destination port: 57725
  10 0.138409  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
  11 0.161617  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355
  12 0.166156  24.226.47.210   192.168.0.195   TCP      41871 > watilapp [ACK] Seq=1 Ack=1 Win=64167 L
  13 0.166248  192.168.0.195   24.226.47.210   TCP      watilapp > 41871 [PSH, ACK] Seq=1 Ack=1 Win=65
  14 0.177443  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
  15 0.217683  192.168.0.195   76.66.160.110   UDP      Source port: 24355  Destination port: 11486
  16 0.233748  76.66.160.110   192.168.0.195   UDP      Source port: 11486  Destination port: 24355

▷ Frame 1: 143 bytes on wire (1144 bits), 143 bytes captured (1144 bits)
▷ Ethernet II, Src: D-Link_ec:f4:48 (00:1b:11:ec:f4:48), Dst: Parallel_af:fb:23 (00:1c:42:af:fb:23)
▷ Internet Protocol, Src: 76.66.160.110 (76.66.160.110), Dst: 192.168.0.195 (192.168.0.195)
▷ User Datagram Protocol, Src Port: 11486 (11486), Dst Port: 24355 (24355)
▷ Data (101 bytes)

0000  00 1c 42 af fb 23 00 1b  11 ec f4 48 08 00 45 00   ..B..#.. ...H..E.
0010  00 81 69 31 00 00 70 11  33 1f 4c 42 a0 6e c0 a8   ..i1..p. 3.LB.n..
0020  00 c3 2c de 5f 23 00 6d  a2 1e aa a1 6d 3d 37 73   ..,._#.m ....m=7s
0030  81 44 e5 b1 35 fc 1c 0b  5f 1d 0a 5e 8c 51 67 67   .D..5... _..^.Qgg
0040  99 c1 ce 31 36 df 03 4c  be 22 9c e5 45 aa cb ea   ...16..L ."..E...
0050  ed 43 d3 f3 db ca 52 ec  8d 9f af d1 00 5a 50 50   .C....R. .....ZPP

File: "/Users/riyad/Desktop/... | Packets: 84871 Displayed: 84871 Marked: 0 Load time: 0:02.924 | Profile: Default
```

Figure 6.12: Wireshark Could not Classify Skype Original Packets


E12_00001_20110510142105 - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

```
No.  Time      Source          Destination     Protocol Info
 214 2.062855  192.168.0.195   24.36.38.72     UDP      Source port: 24355  Destination port: 43254
 215 2.067028  24.36.38.72     192.168.0.195   UDP      Source port: 43254  Destination port: 24355
 216 2.071968  192.168.0.195   24.36.38.72     UDP      Source port: 24355  Destination port: 43254
 217 2.083162  192.168.0.195   24.36.38.72     UDP      Source port: 24355  Destination port: 43254
 218 2.087365  24.36.38.72     192.168.0.195   UDP      Source port: 43254  Destination port: 24355
 219 2.105183  192.168.0.195   24.36.38.72     UDP      Source port: 24355  Destination port: 43254
 220 2.106669  24.36.38.72     192.168.0.195   UDP      Source port: 43254  Destination port: 24355
 221 2.122280  192.168.0.195   24.36.38.72     UDP      Source port: 24355  Destination port: 43254
 222 2.124703  24.36.38.72     192.168.0.195   UDP      Source port: 43254  Destination port: 24355
 223 2.141105  192.168.0.195   24.36.38.72     UDP      Source port: 24355  Destination port: 43254
 224 2.145522  24.36.38.72     192.168.0.195   UDP      Source port: 43254  Destination port: 24355
 225 2.163998  192.168.0.195   24.36.38.72     UDP      Source port: 24355  Destination port: 43254
 226 2.170314  24.36.38.72     192.168.0.195   UDP      Source port: 43254  Destination port: 24355
 227 2.181462  24.36.38.72     192.168.0.195   UDP      Source port: 43254  Destination port: 24355
 228 2.182230  192.168.0.195   24.36.38.72     UDP      Source port: 24355  Destination port: 43254
 229 2.204607  192.168.0.195   24.36.38.72     UDP      Source port: 24355  Destination port: 43254
```

```
▷ Frame 230: 106 bytes on wire (848 bits), 106 bytes captured (848 bits)
▷ Ethernet II, Src: D-Link_ec:f4:48 (00:1b:11:ec:f4:48), Dst: Parallel_af:fb:23 (00:1c:42:af:fb:23)
▷ Internet Protocol, Src: 24.36.38.72 (24.36.38.72), Dst: 192.168.0.195 (192.168.0.195)
▷ User Datagram Protocol, Src Port: 43254 (43254), Dst Port: 24355 (24355)
▷ Data (64 bytes)
```

```
0000  00 1c 42 af fb 23 00 1b  11 ec f4 48 08 00 45 00   ..B..#.. ...H..E.
0010  00 5c 32 23 00 00 38 11  50 97 18 24 26 48 c0 a8   .\2#..8. P..$&H..
0020  00 c3 a8 f6 5f 23 00 48  59 75 ec 7c 4d a7 c2 a5   ...._#.H Yu.|M...
0030  c8 e5 ff e7 45 56 85 fb  4c 4e fe ff f9 b5 b7 dd   ....EV.. LN......
0040  5c de 5e be e0 f4 94 49  df a4 7c 58 ee b7 04 f9   \.^....I ..|X....
0050  b4 1e 58 2e d8 3c 6d e2  ca a2 6f 51 17 b8 2c 74   ..X..<m. ..oQ...t
```

File: "/Users/riyad/Desktop/... Packets: 90978 Displayed: 90978 Marked: 0 Load time: 0:03.261 Profile: Default

Figure 6.13: Wireshark Could not Classify Altered-Skype Packets

6.4 Discussion

This chapter has demonstrated how easy it is to evade public tools used for traffic classification which are based on the DPI method (OpenDPI) and the port number based method (Wireshark). The results of these experiments show the importance and the need of the proposed approach for finding robust signatures which are difficult to evade. To do so, it was necessary to evaluate the robustness/generalization of the solutions provided by the ML algorithms C5.0, GP and AdaBoost in Skype traffic classification on unseen altered data by padding/morphing, i.e. evasion attacks. These experimental results suggest that the proposed approach provides over 80% DR on average for Skype VoIP traffic classification even if a user alters the characteristics of the Skype traffic maliciously. However, the three ML-based classifiers used in this thesis build different solutions and their performances are different on the test traces. Thus, there is no dominant classifier for all of the test data sets, even though the C5.0-based classifier on average has better performance than the AdaBoost and the GP based classifiers. Therefore, the automatically generated signatures are examined in more detail in Chapter 7 in order to understand better how the proposed approach solves the problem of classifying VoIP Skype traffic robustly. In Chapter 8 the performance of the three ML-based classifiers will be investigated when they are combined together (Ensemble Learning).

Chapter 7

Analysis of the Classifier Solutions

In this chapter, the aim is to analyze the solutions generated by the classification-based system in terms of Central Processing Unit (CPU) training time and complexity of the solutions. The complexity analysis is based on the number of features and the number of rules generated by each classifier. Furthermore, the best solution for each classifier is discussed in terms of the flow/packet header feature sets, generated signatures (rules/patterns) and the FPR to understand how the classification based approach identifies the Skype flows. The approach adopted is to include as wide a feature set as possible and let the 'embedded' properties of the various learning algorithms establish which subset of features to employ. Given this capability, the features selected by each learning algorithm are reviewed by class for both C5.0 and GP. On the other hand, the summary for AdaBoost is not straightforward, since AdaBoost is an adaptive boosting meta-learning algorithm. Thus, it will be limited to the total set of features utilized, independent of the class. The analysis is done using an Intel Xeon 2.67 GHz Quad 16 Core chip with 48 GB of RAM for the the flow/packet header feature sets.

7.1 Analysis of the Packet Header-based Approach for Skype Detection

This section will analyze the solutions generated by classification for the Packet Header-based approach to understand how Skype packets are identified.

7.1.1 CPU Training Time

Figure 7.1 summarizes the CPU training time. The steadiness of GP in finding fast solutions is readily apparent. The GP-based classifier is the fastest ML algorithm by building its model in ≈3560 seconds on average. Again, the CPU training time for C5.0 is extremely fast as well (≈9140 seconds on average). Both GP and C5.0 are particularly impressive in finding very good solutions (see the results in Chapter 5, section 5.1). By contrast, AdaBoost takes a longer time for training (on average $\approx$ 12100 seconds, which is 3 hours, 21 minutes) compared to GP (60 minutes on average) and C5.0 (2 hours, 32 minutes on average). Indeed, GP and C5.0 can find effective solutions with large training sizes ($\approx$20 million records) in few hours.

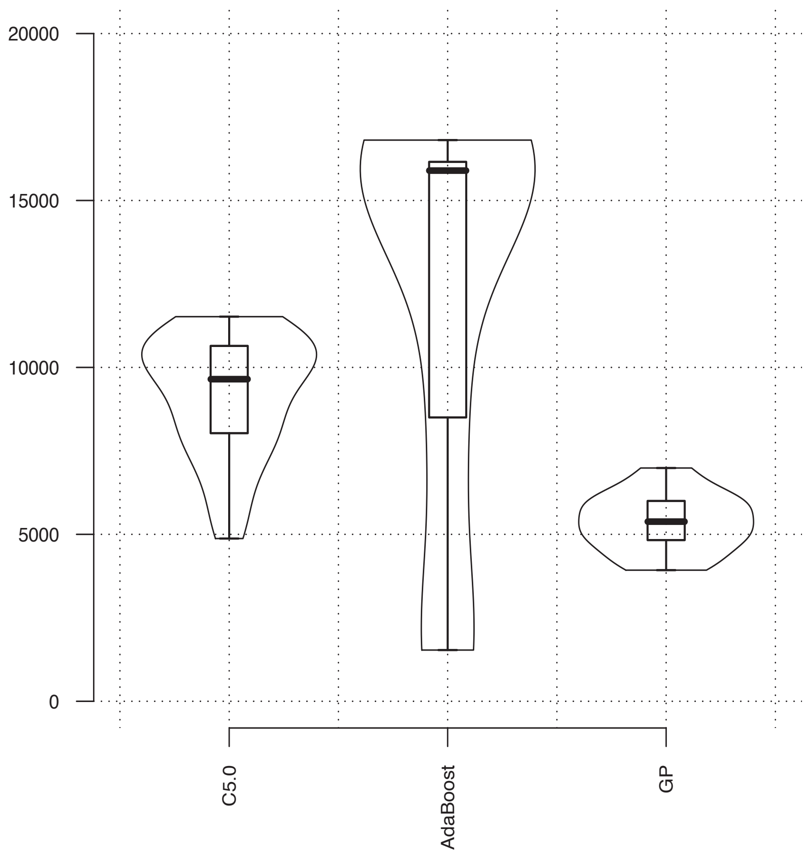


Figure 7.1: Training Time (in Seconds) for Skype Detection Based on the Packet Header Feature Set

7.1.2 Complexity of Solutions

The AdaBoost-based classifier utilizes on average a lower total count of attributes for Skype (in-class) and (out-class) traffic than either of the GP or C5.0 based classifiers (Figures 7.2 and 7.3). The GP-based classifier uses the largest set of attributes as a whole or by class for Skype detection whereas, on average, the C5.0-based classifier uses the second largest set of attributes as a whole or by class for Skype detection. In terms of solution complexity, the GP-based classifier generates fewer individuals (teams) on average (7 for non-Skype and 8 for Skype) for detecting Skype packet traffic while the AdaBoost-based classifier finds the second simplest solution (8 rules for non-Skype and 8 rules for Skype) and the C5.0-based classifier finds the most complex solution for Skype (on average 283 rules for non-Skype and 464 rules for Skype), Figures 7.4 and 7.5.

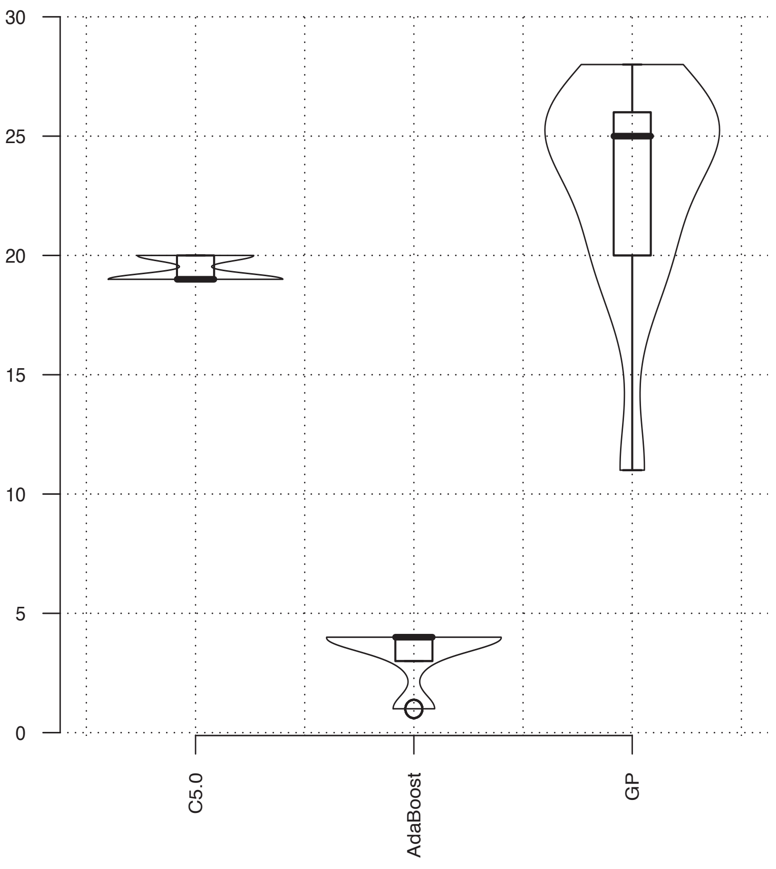


Figure 7.2: Number of Features Utilized for Each Classifier for Skype Classification Based on the Packet Header Feature Set

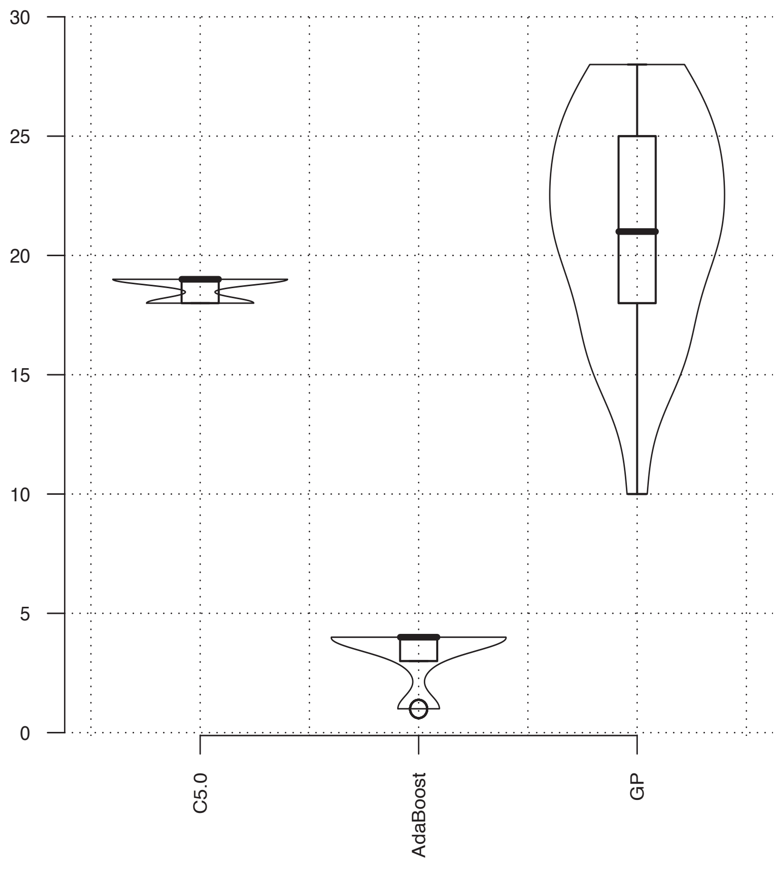


Figure 7.3: Number of Features Utilized for Each Classifier for non-Skype Classification Based on the Packet Header Feature Set

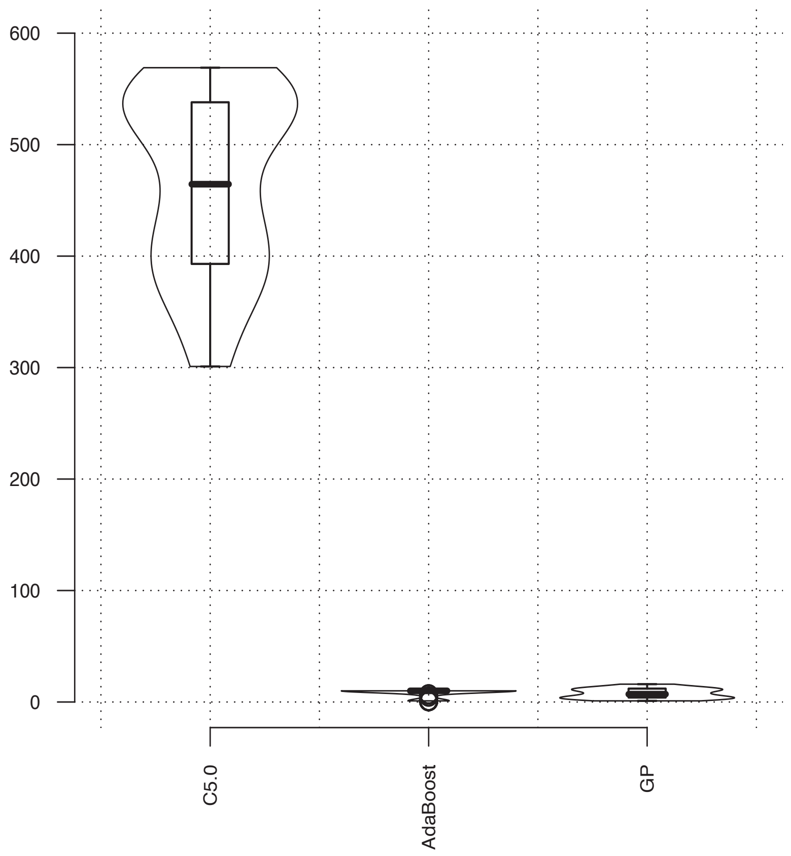


Figure 7.4: Number of Rules/Individuals Utilized for Each Classifier for Skype Classification Based on the Packet Header Feature Set

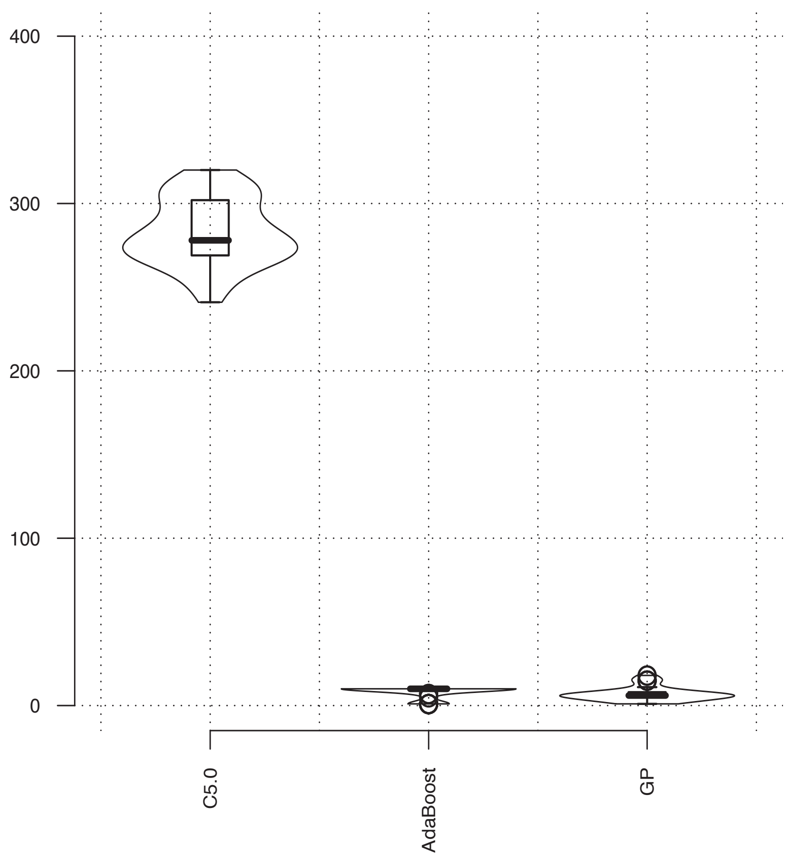


Figure 7.5: Number of Rules/Individuals Utilized for Each Classifier for non-Skype Classification Based on the Packet Header Feature Set

7.1.3 Best Solution for Each Classifier for the Packet Header-based Approach for Skype Detection

For the number of features utilized based on the packet header feature set (attributes) for Skype detection, Tables 7.1, 7.2 and 7.3 summarize these findings for the GP, C5.0 and AdaBoost based classifiers, respectively. Clearly, AdaBoost uses the lowest number of attributes relative to GP and C5.0 for Skype detection. Conversely, GP uses the largest set of attributes as a whole or by class. Each classifier identifies attributes unique to its own solution as well. For example, AdaBoost focuses mainly on attributes based on the frame header (Table 7.3). GP is the only model to make use of '*frame.marked*' for Skype detection (Table 7.1). Furthermore, GP and C5.0 discover the hierarchy of the network protocol stack layers (a five layer network stack) and build their model for selecting attributes from each of the network protocol stack layers, which include the Data Link layer (the frame header), the Network layer (IP header) and the Transport layer (TCP/UDP header) for Skype detection.

Table 7.1: Features Used by GP for the in/out-classes for Skype Based on the Packet Header Feature Set

	in-class	**out-class**
1	frame.time_delta	frame.time_delta
2	frame.pkt_len	frame.pkt_len
3	frame.len	frame.len
4	frame.cap_len	frame.cap_len
5	frame.marked	frame.marked
6	ip.len	ip.len
7	ip.flags	ip.flags
8	ip.flags.rb	ip.flags.rb
9	ip.flags.df	ip.flags.df
10	ip.flags.mf	ip.flags.mf
11	ip.frag.offset	ip.frag.offset
12	ip.proto	ip.ttl
13	tcp.window_size	tcp.window_size
14	udp.length	udp.length
15	tcp.seq	tcp.seq
16	tcp.nxtseq	tcp.nxtseq
17	tcp.len	tcp.ack
18	tcp.hdr_len	tcp.hdr_len
19	tcp.flags	tcp.flags
20	tcp.flags.cwr	tcp.flags.cwr
21	tcp.flags.ecn	tcp.flags.ecn
22	tcp.flags.urg	tcp.flags.urg
23	tcp.flags.push	tcp.flags.push
24	tcp.flags.reset	tcp.flags.reset
25	tcp.flags.syn	tcp.flags.syn
26	tcp.flags.fin	tcp.flags.fin
27		tcp.flags.ack

Table 7.2: Features Used by C5.0 for the in/out-classes for Skype Based on the Packet Header Feature Set

	in-class	**out-class**
1	frame.time_delta	frame.time_delta
2	frame.pkt_len	frame.pkt_len
3	frame.cap_len	frame.cap_len
4	ip.len	ip.len
5	ip.flags	ip.flags
6	ip.flags.df	ip.flags.df
7	ip.ttl	ip.ttl
8	tcp.len	tcp.len
9	tcp.seq	tcp.seq
10	tcp.nxtseq	tcp.nxtseq
11	tcp.ack	tcp.ack
12	tcp.hdr_len	tcp.hdr_len
13	tcp.flags	tcp.flags
14	tcp.flags.push	tcp.flags.push
15	tcp.flags.reset	tcp.flags.reset
16	tcp.flags.syn	tcp.flags.syn
17	tcp.flags.fin	tcp.flags.fin
18	tcp.window_size	tcp.window_size
19	udp.length	udp.length

Table 7.3: Features Used by AdaBoost as a Whole for Skype Based on the Packet Header Feature Set

	in-class	**out-class**
1	frame.time_delta	frame.time_delta
2	frame.len	frame.len
3	frame.cap_len	frame.cap_len
4	tcp.window_size	tcp.window_size

Also of interest is the high level of overlap in shared attributes among the three ML algorithms. C5.0 shared three of the four attributes utilized by AdaBoost and all C5.0 attributes shared with GP. What is certainly clear, however, is that a significant degree of preference exists in attribute selection relative to the ML model; thus, attempting to provide a limited ‘hand crafted’ set of attributes is likely to be counter-productive.

In terms of solution complexity for Skype packet header solutions AdaBoost generates ten signatures for both Skype traffic and non-Skype traffic, whereas C5.0 employs 461 signatures for Skype classification and 287 signatures to classify non-Skype traffic. The SBB-based GP uses nine individuals for Skype classification and nine for non-Skype. Figures 7.6, 7.7 and 7.8 show part of the C5.0, AdaBoost and GP solutions for the detection of Skype traffic based on the packet header feature set. Moreover, in terms of the number of packets each model can classify per second, the C5.0-based model leads the competition. In this case, the GP solution can classify $\approx$ 790000 packets/second, the AdaBoost-based model can classify $\approx$ 1.16e+06 packets/second and the C5.0-based model can classify $\approx$ 3.17e+09 packets/second in terms of classifying the data sets offline without any special treatment for speed-ups. However, the number of records processed per second can be improved by using either a faster machine or parallel computing to implement the signatures, i.e. the solutions of the ML algorithms. For instance, parallel computing can be implemented in the case of GP by processing simultaneously each individuals/teams of the GP solution. Hence, the results of each individuals/teams can be obtained at the same time in order to find the winner individuals/teams and label the traffic.

```
Decision Stump

Classifications

frame.time_delta <= 0.0012155 : NOTSKYPE
frame.time_delta > 0.0012155 : SKYPE
frame.time_delta is missing : NOTSKYPE

Class distributions

frame.time_delta <= 0.0012155
NOTSKYPE    SKYPE
0.9929608899388909      0.007039110061109188
frame.time_delta > 0.0012155
NOTSKYPE    SKYPE
0.08998482013626576     0.9100151798637343
frame.time_delta is missing
NOTSKYPE    SKYPE
0.9828019681503587      0.017198031849641335


Weight: 4.82

Decision Stump

Classifications

frame.cap_len <= 4.3630785 : NOTSKYPE
frame.cap_len > 4.3630785 : SKYPE
frame.cap_len is missing : NOTSKYPE

Class distributions

frame.cap_len <= 4.3630785
NOTSKYPE    SKYPE
0.7148325695260668      0.2851674304739333
frame.cap_len > 4.3630785
NOTSKYPE    SKYPE
9.032747622759881E-4    0.999096725237724
frame.cap_len is missing
NOTSKYPE    SKYPE
0.5583330534039108      0.4416669465960891


Weight: 1.25
```

Figure 7.6: An Example of an AdaBoost Solution for Skype Based on the Packet Header Feature Set

```
Rule 287: (3271/22, lift 1.0)
	frame.time_delta <= 0.000876
	frame.pkt_len > 4.672829
	frame.pkt_len <= 4.882802
	ip.ttl > 4.682131
	ip.ttl <= 4.70048
	tcp.window_size <= 8.651549
	->  class NOTSKYPE  [0.993]

Rule 288: (47628/2, lift 58.1)
	frame.time_delta <= 4e-06
	frame.cap_len > 4.060443
	->  class SKYPE  [1.000]

Rule 289: (122526, lift 58.1)
	frame.cap_len > 4.060443
	tcp.hdr_len <= 2.995732
	->  class SKYPE  [1.000]

Rule 290: (2203, lift 58.1)
	frame.time_delta > 0.000473
	frame.pkt_len <= 6.202536
	frame.cap_len > 4.060443
	ip.ttl > 4.75359
	ip.ttl <= 4.812184
	tcp.seq > 11.71681
	tcp.window_size > 8.651549
	tcp.window_size <= 9.847076
	->  class SKYPE  [1.000]
```

Figure 7.7: An Example of a C5.0 Solution for Skype Based on the Packet Header Feature Set

```
R[0] <- R[0] R[4] exp
R[7] <- R[7] R[0] cos
R[3] <- R[3] R[7] diff
R[5] <- R[5] I[4] cos
R[0] <- R[0] I[2] div
R[0] <- R[0] I[28] diff
R[1] <- R[1] I[0] log
R[6] <- R[6] R[3] log
R[1] <- R[1] I[0] sum
R[7] <- R[7] I[15] log
R[3] <- R[3] R[3] diff
R[2] <- R[2] I[25] exp
R[0] <- R[0] R[6] diff
R[1] <- R[1] R[6] prod
R[3] <- R[3] I[27] mod
R[7] <- R[7] I[3] exp
R[0] <- R[0] R[5] log
R[3] <- R[3] R[0] diff
R[0] <- R[0] R[3] diff
R[0] <- R[0] R[1] cos
R[1] <- R[1] R[6] prod
R[5] <- R[5] I[1] cos
R[0] <- R[0] R[6] div
```

Figure 7.8: An Example of a GP Solution for Skype Based on the Packet Header Feature Set

7.1.4 FPR for the Best Solutions for Each Classifier based on the Packet Header Feature Set for Skype Detection

Figures 7.9, 7.10, 7.11, 7.12, 7.13 and 7.14 list the application packets which C5.0, GP and AdaBoost misclassified as Skype packets for the Univ2007 test and Univ2010 traces employed (See Appendix C.1 for the tables). Given the fact that Skype runs over TCP or UDP, this is to be expected. C5.0, GP and AdaBoost are mostly classifying the HTTP and 'OTHER' classes as Skype since Skype uses TCP and UDP for setting up communication calls.

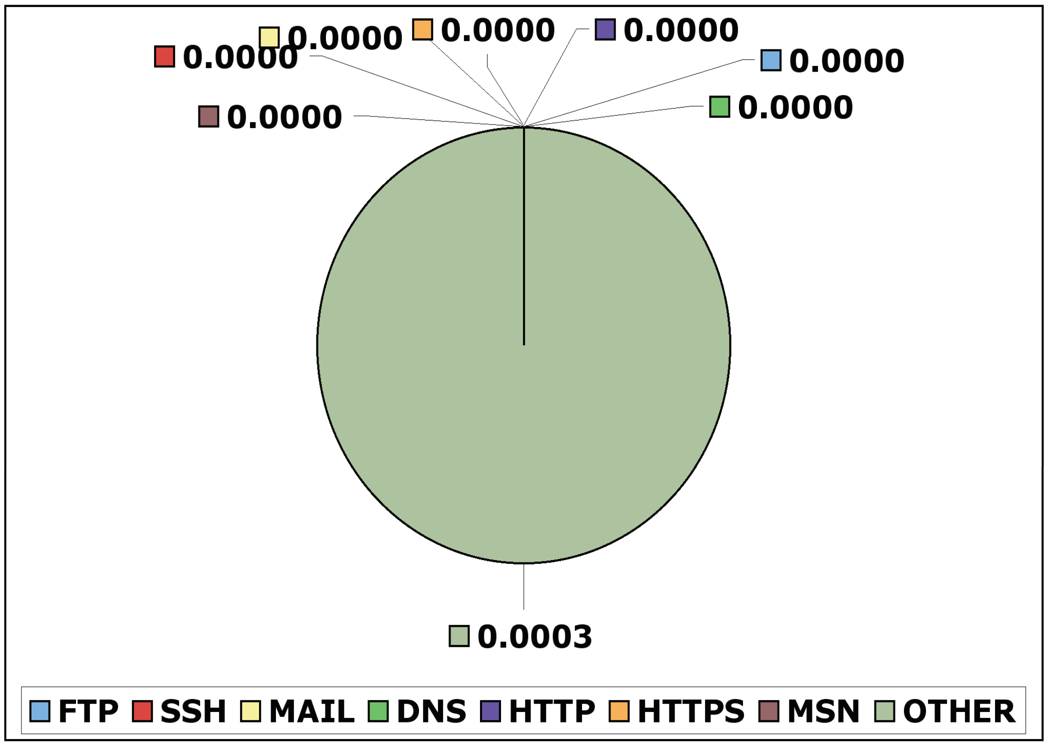


Figure 7.9: Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Packet Header Feature Set on the Univ2007 Test Trace (FPR=0.03%).

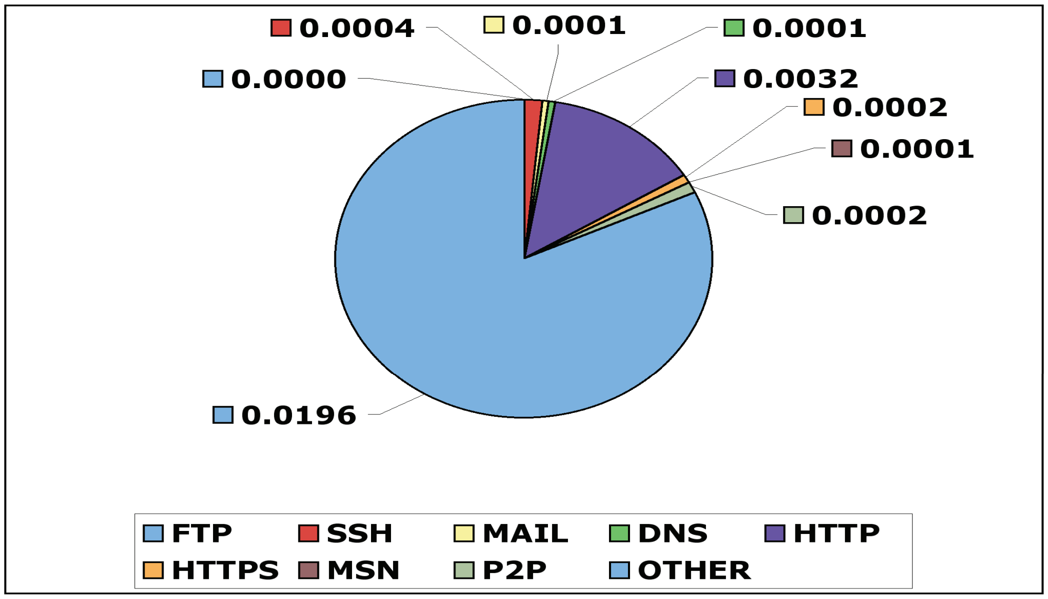


Figure 7.10: Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Packet Header Feature Set on the Univ2010 Test Trace (FPR=2.4%).

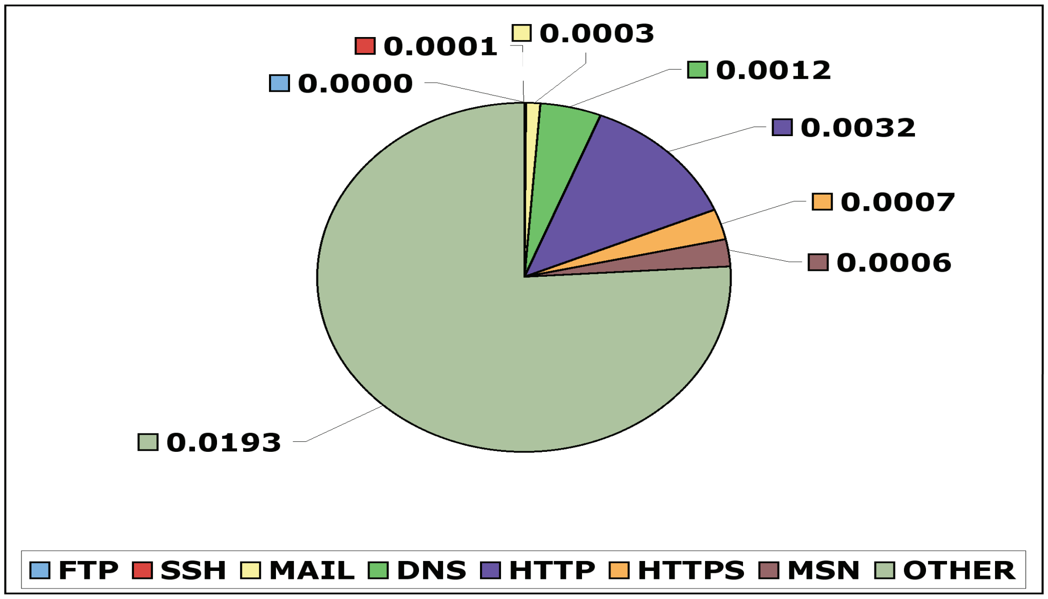


Figure 7.11: Applications Wrongly Classified as Skype by the GP-based Signatures on the Packet Header Feature Set on the Univ2007 Test Trace (FPR=2.5%).

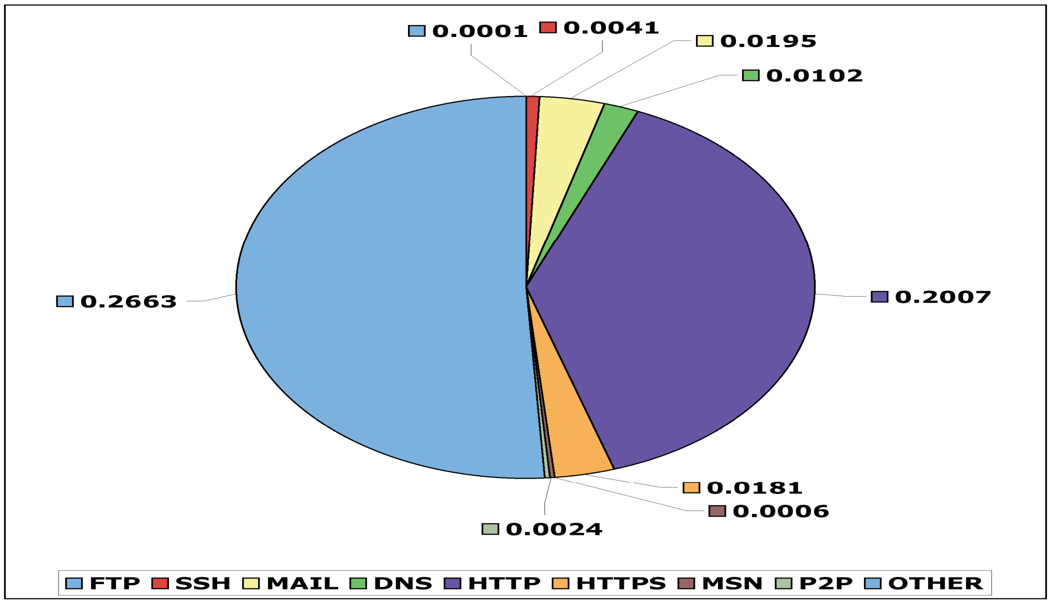


Figure 7.12: Applications Wrongly Classified as Skype by the GP-based Signatures on the Packet Header Feature Set on the Univ2010 Test Trace (FPR=65.1%).

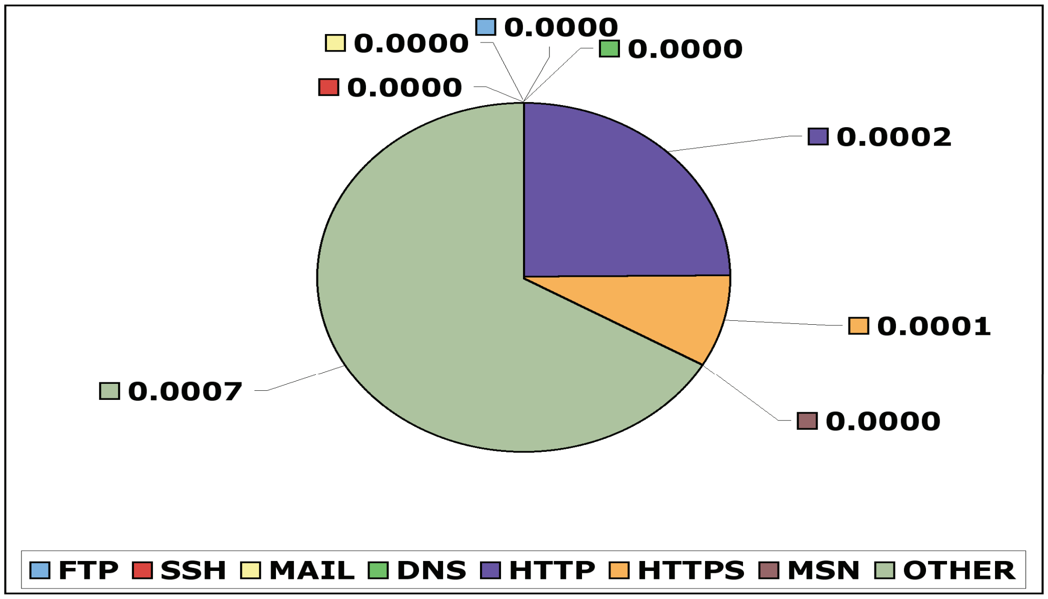


Figure 7.13: Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Packet Header Feature Set on the Univ2007 Test Test Trace (FPR=0.11%).

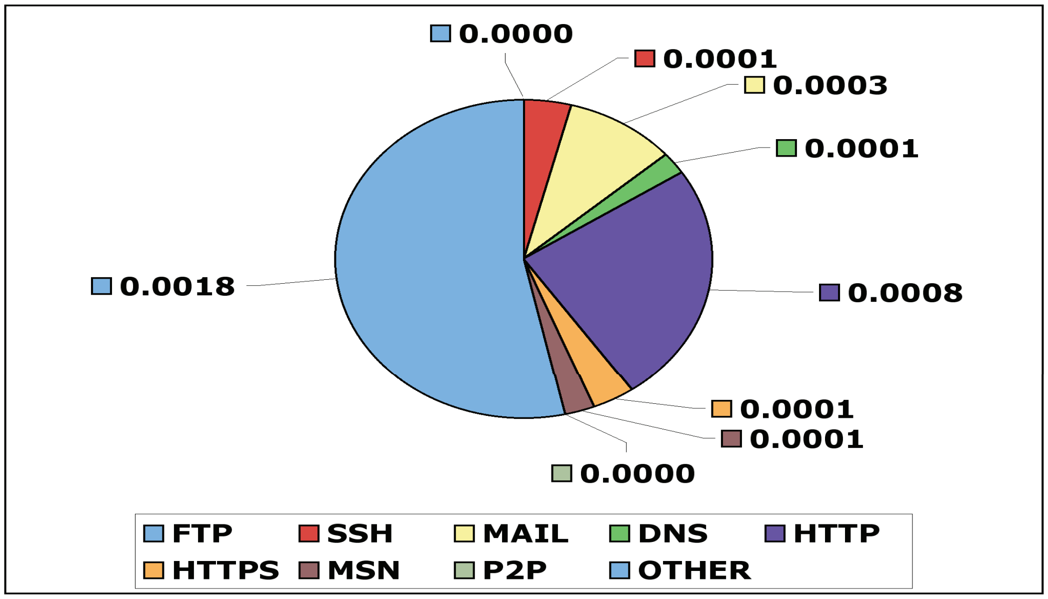


Figure 7.14: Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Packet Header Feature set on the Univ2010 Test Trace (FPR=0.34%).

7.2 Analysis of the Flow-based Approach for Solutions Generated by all Classifiers

7.2.1 CPU Training Time

Figure 7.15 summarizes the CPU training time. AdaBoost is the fastest compared to GP and C5.0 for building its model, as shown in Figure 7.15. However, its performance is very low. By contrast, the CPU training time for C5.0 is very fast and particularly impressive in finding very good solutions (see the results in Chapter 5, section 5.2). The GP-based classifier takes a longer time for training (on average 1 hour, 20 minutes) compared to C5.0 and AdaBoost (on average less than 8 minutes). Indeed, C5.0 can find effective solutions with large training sizes (≈1.7 million records) in a few minutes.

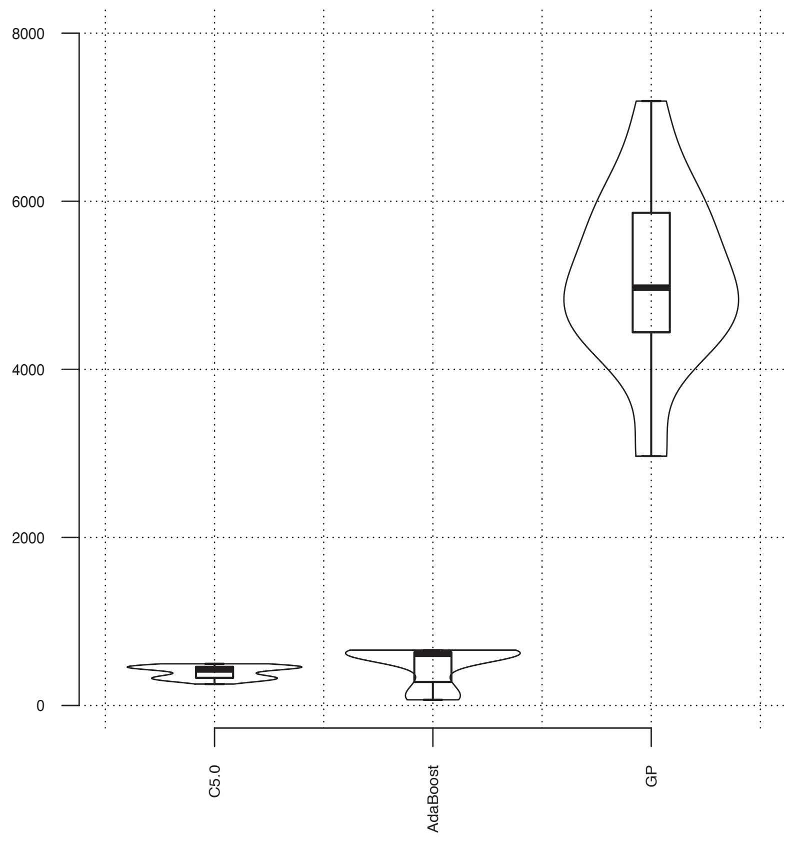


Figure 7.15: Training Time (in Seconds) for Skype Detection Based on the Flow Feature Set

7.2.2 Complexity of Solutions

AdaBoost uses on average a lower total count of attributes for Skype (in-class) and (out-class) relative to either GP or C5.0 (Figures 7.16 and 7.17). Conversely, C5.0 uses the largest set of attributes as a whole or by class for Skype detection whereas, on average, GP uses the second largest set of attributes as a whole or by class for Skype detection. In terms of solution complexity, GP generates fewer individuals/teams on average (ten for non-Skype and six for Skype) to detect Skype flow traffic while AdaBoost finds a simpler solution (eight rules for non-Skype and eight rules for Skype) and C5.0 finds the most complex solution for Skype (on average 213 rules for non-Skype and 238 rules for Skype), as shown in Figures 7.18 and 7.19.

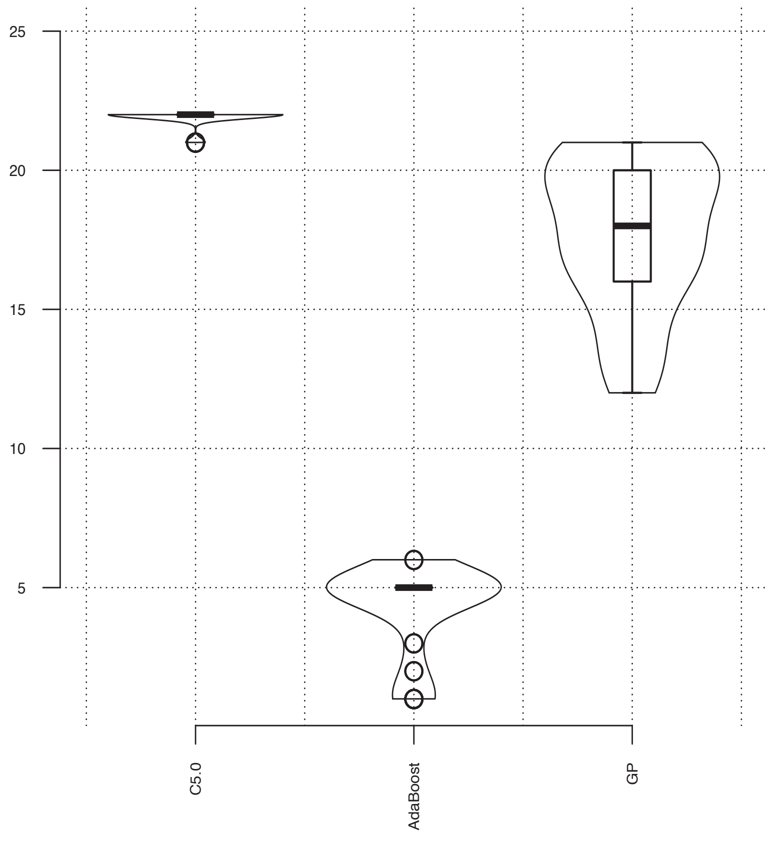


Figure 7.16: Number of Features Utilized for Each Classifier for Skype Classification Based on the Flow Feature Set

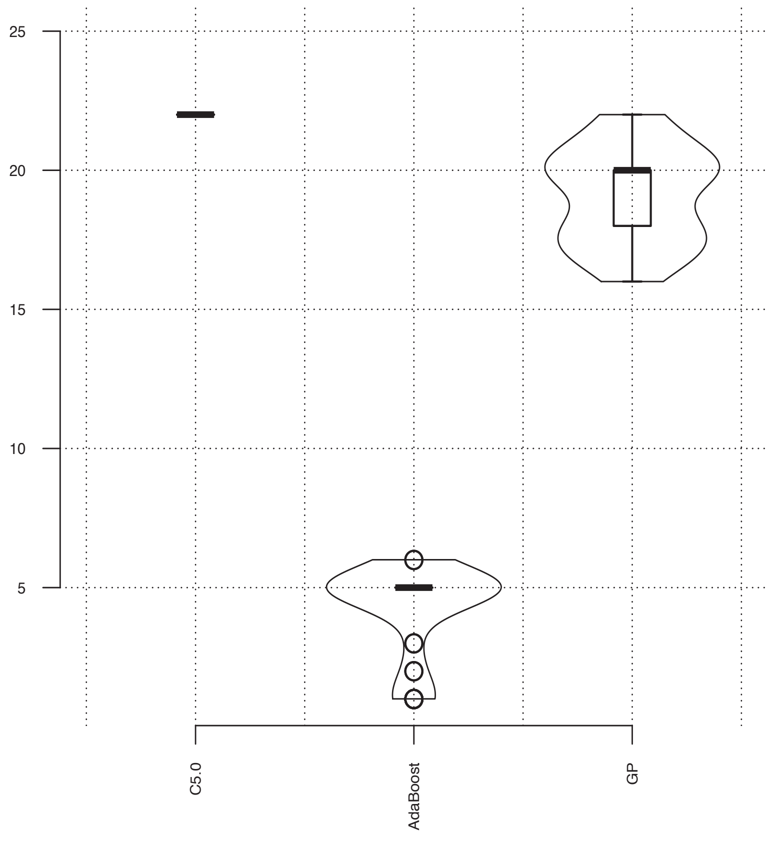


Figure 7.17: Number of Features Utilized for Each Classifier for non-Skype Classification Based on the Flow Feature Set

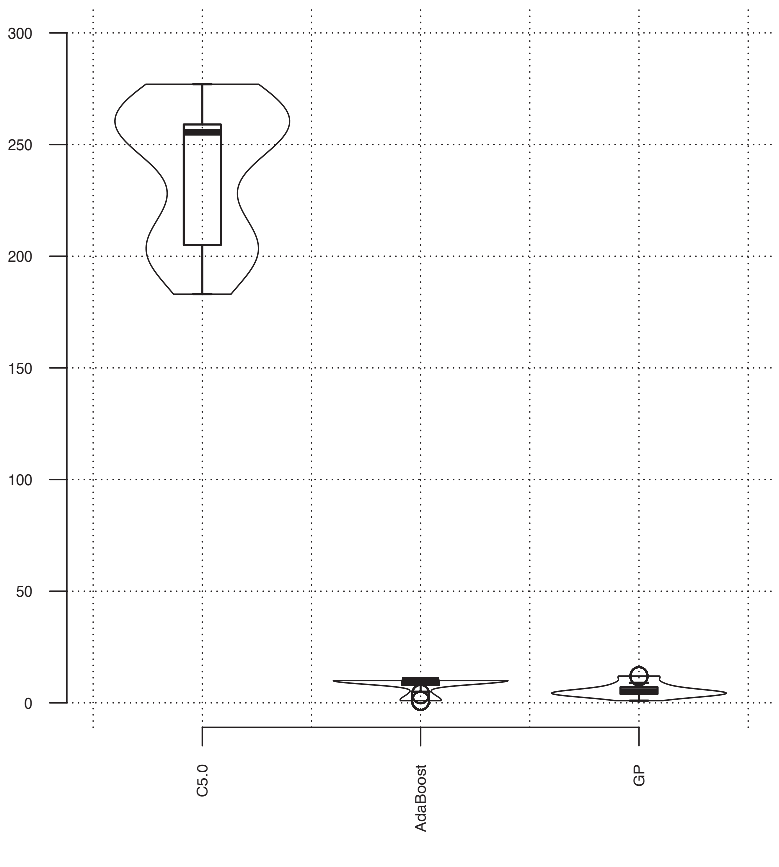


Figure 7.18: Number of Rules/Individuals Utilized for Each Classifier for Skype Classification Based on the Flow Feature Set

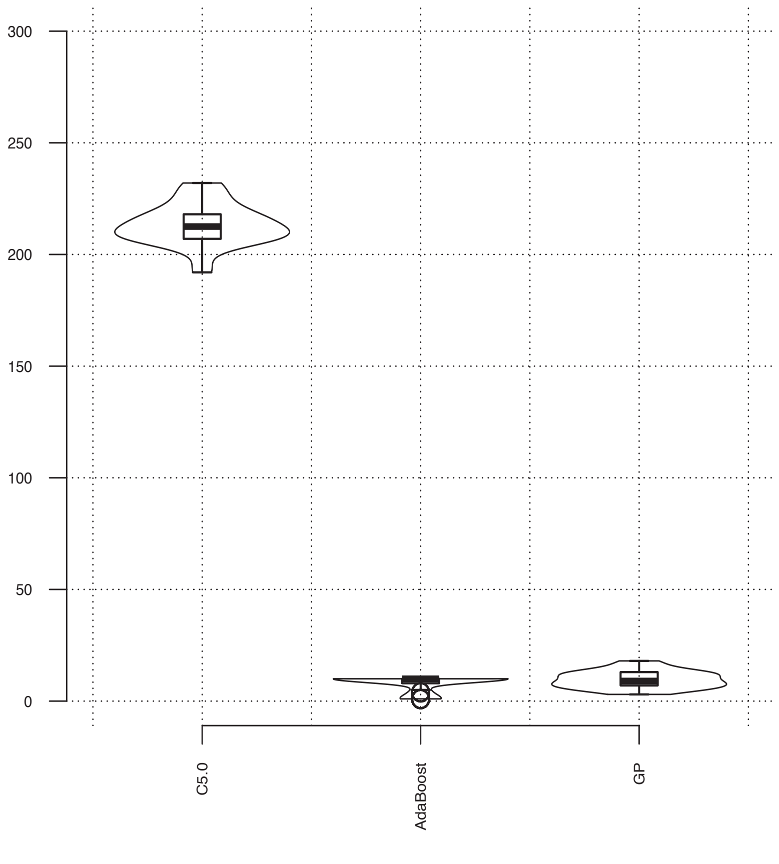


Figure 7.19: Number of Rules/Individuals Utilized for Each Classifier for non-Skype Classification Based on the Flow Feature Set

7.2.3 Analysis of the Best Solution from Each Classifier

For Skype, Figures 7.20, 7.21 and 7.22 show the feature selection properties of the GP, AdaBoost and C5.0 classifiers, respectively. Clearly, AdaBoost uses a lower count of attributes for Skype detection. Conversely, C5.0 uses the largest set of attributes as a whole or by class. In addition each classier identifies features unique to its own solution. For example, AdaBoost chooses the Duration features since the Skype application depends on the quality of the services for the end user; GP utilizes the features based on packet size and inter-arrival time intuitively, which makes sense since the Skype application is mostly interactive (user–user and user-machine); see Figure 7.20. These features may give more insight into the behaviour of the users and can provide more information for predicting what the payload might be. By contrast C5.0 uses all of the features to build its model (22 features). The overlap in shared attributes among the machine learning models is high with four features shared between AdaBoost and GP for Skype detection. It is clear that a significant degree of preference exists in feature selection relative to the machine learning model, so trying to give a specific limited set of flow features for Skype identification is likely to be counter-productive.

AdaBoost generates ten signatures for both Skype traffic and non-Skype traffic; C5.0 employs 187 rules for identifying Skype traffic and 215 rules (signatures) for non-Skype traffic and GP uses five individuals for Skype classification and eleven for non-Skype. In short, the simplicity of the GP and AdaBoost solutions does not appear to be traded off for classifier complexity while still achieving very good performance, in effect emphasizing the significance of support for problem decomposition in this application. Figures 7.23, 7.24 and 7.25 show part of the AdaBoost, C50 and GP solutions for detection Skype traffic based on the flow feature set.

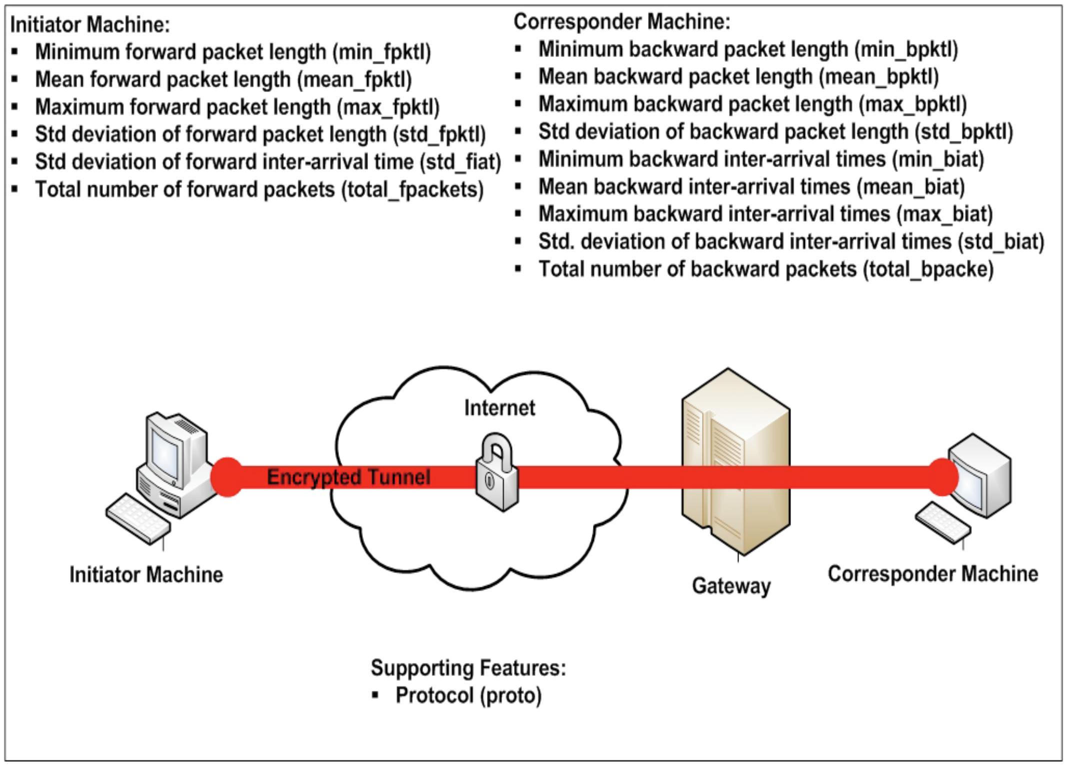

Initiator Machine:
▪ Minimum forward packet length (min_fpktl)
▪ Mean forward packet length (mean_fpktl)
▪ Maximum forward packet length (max_fpktl)
▪ Std deviation of forward packet length (std_fpktl)
▪ Std deviation of forward inter-arrival time (std_fiat)
▪ Total number of forward packets (total_fpackets)
Corresponder Machine:
▪ Minimum backward packet length (min_bpktl)
▪ Mean backward packet length (mean_bpktl)
▪ Maximum backward packet length (max_bpktl)
▪ Std deviation of backward packet length (std_bpktl)
▪ Minimum backward inter-arrival times (min_biat)
▪ Mean backward inter-arrival times (mean_biat)
▪ Maximum backward inter-arrival times (max_biat)
▪ Std. deviation of backward inter-arrival times (std_biat)
▪ Total number of backward packets (total_bpacke)
Internet
Encrypted Tunnel
Initiator Machine
Gateway
Corresponder Machine
Supporting Features:
▪ Protocol (proto)

Figure 7.20: Selected Flow Features of the GP-based Classifier for Skype Detection

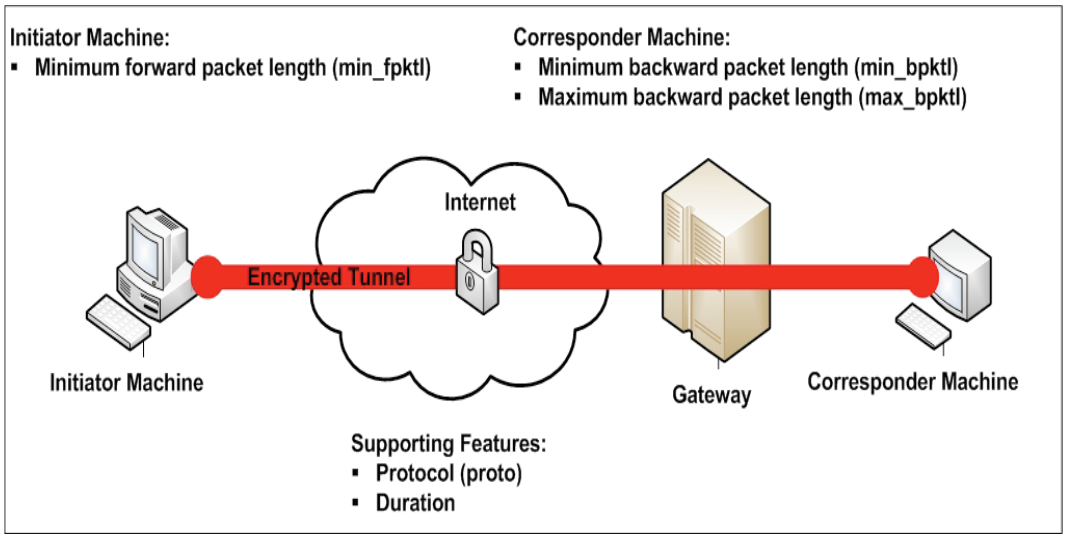


Figure 7.21: Selected Flow Features of the AdaBoost-based Classifier for Skype Detection

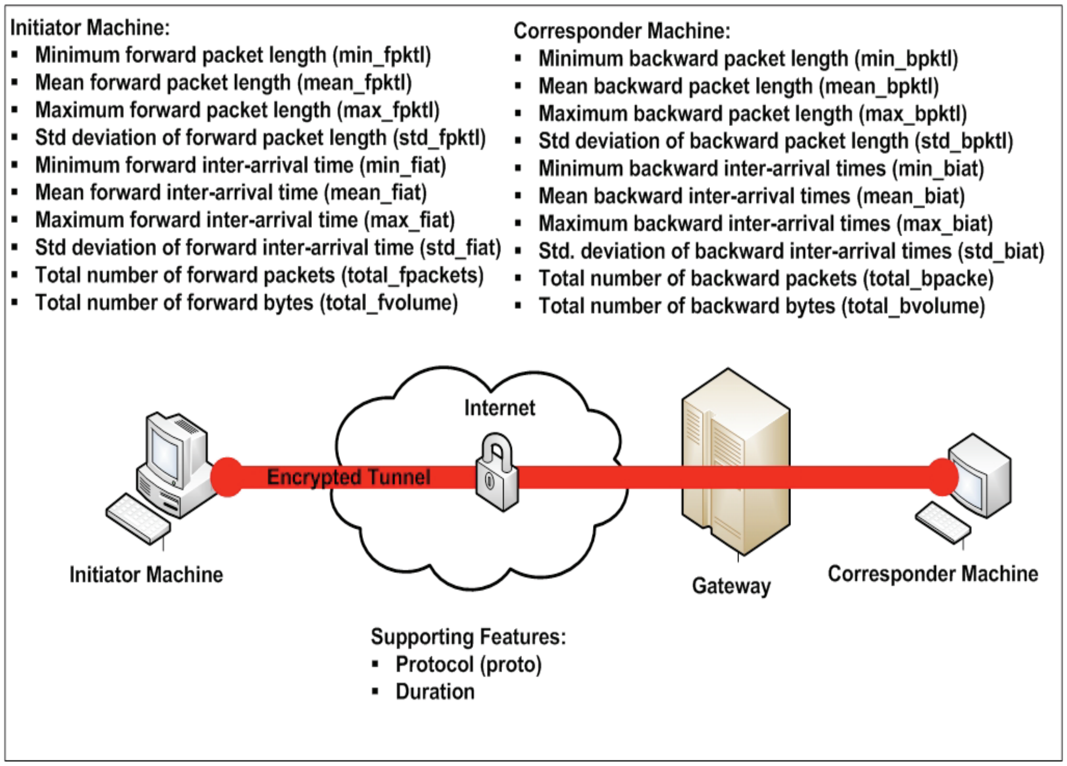

Initiator Machine:
Minimum forward packet length (min_fpktl)
Mean forward packet length (mean_fpktl)
Maximum forward packet length (max_fpktl)
Std deviation of forward packet length (std_fpktl)
Minimum forward inter-arrival time (min_fiat)
Mean forward inter-arrival time (mean_fiat)
Maximum forward inter-arrival time (max_fiat)
Std deviation of forward inter-arrival time (std_fiat)
Total number of forward packets (total_fpackets)
Total number of forward bytes (total_fvolume)
Corresponder Machine:
Minimum backward packet length (min_bpktl)
Mean backward packet length (mean_bpktl)
Maximum backward packet length (max_bpktl)
Std deviation of backward packet length (std_bpktl)
Minimum backward inter-arrival times (min_biat)
Mean backward inter-arrival times (mean_biat)
Maximum backward inter-arrival times (max_biat)
Std. deviation of backward inter-arrival times (std_biat)
Total number of backward packets (total_bpacke)
Total number of backward bytes (total_bvolume)
Internet
Encrypted Tunnel
Initiator Machine
Gateway
Corresponder Machine
Supporting Features:
Protocol (proto)
Duration

Figure 7.22: Selected Flow Features for the C5.0-based Classifier for Skype Detection

```
Decision Stump

Classifications

max_bpktl <= 62.5 : SKYPE
max_bpktl > 62.5 : NOTSKYPE
max_bpktl is missing : NOTSKYPE

Class distributions

max_bpktl <= 62.5
NOTSKYPE      SKYPE
0.19062681872503764  0.8093731812749624
max_bpktl > 62.5
NOTSKYPE      SKYPE
0.8978431996029191   0.10215680039708083
max_bpktl is missing
NOTSKYPE      SKYPE
0.714826729087577    0.28517327091242295


Weight: 1.95

Decision Stump

Classifications

proto <= 11.5 : NOTSKYPE
proto > 11.5 : SKYPE
proto is missing : NOTSKYPE
```

Figure 7.23: An Example of an AdaBoost Solution for Skype Detection Based on the Flow Feature Set

```
Rule 215: (879366/168737, lift 1.1)
        min_fpktl <= 139
        min_bpktl > 48
        ->  class NOTSKYPE  [0.808]

Rule 216: (9456/3, lift 3.5)
        std_fpktl <= 0
        max_bpktl <= 321
        std_bpktl > 80
        max_fiat <= 26620
        mean_biat > 34983
        mean_biat <= 330261
        ->  class SKYPE  [1.000]

Rule 217: (1801, lift 3.5)
        min_fpktl > 131
        duration <= 126901
        total_fvolume <= 132
        total_bvolume <= 63
        ->  class SKYPE  [0.999]

Rule 218: (798, lift 3.5)
        min_fiat > 0
        min_fiat <= 3
        ->  class SKYPE  [0.999]

Rule 219: (6044/3, lift 3.5)
        min_fpktl <= 94
        std_fpktl <= 2
        min_bpktl > 61
        min_bpktl <= 79
        max_bpktl > 244
        max_bpktl <= 298
        std_bpktl > 103
        ->  class SKYPE  [0.999]

Rule 220: (782, lift 3.5)
        min_fpktl > 34
        min_fpktl <= 50
        std_fpktl > 2
        min_bpktl > 46
        min_bpktl <= 79
        mean_bpktl > 164
        max_bpktl <= 595
        proto > 6
        total_fpackets <= 8
        total_bvolume > 359
        ->  class SKYPE  [0.999]
```

Figure 7.24: An Example of a C5.0 Solution for Skype Detection Based on the Flow Feature Set

```
R[5]  <-  R[5]  I[3]  cos
R[3]  <-  R[3]  R[2]  mod
R[5]  <-  R[5]  R[1]  cos
R[2]  <-  R[2]  R[1]  cos
R[4]  <-  R[4]  R[0]  log
R[1]  <-  R[1]  I[1]  mod
R[4]  <-  R[4]  I[0]  log
R[2]  <-  R[2]  R[4]  div
R[5]  <-  R[5]  I[20]  cos
R[6]  <-  R[6]  I[6]  div
R[4]  <-  R[4]  R[3]  cos
R[1]  <-  R[1]  I[17]  div
R[4]  <-  R[4]  R[2]  prod
R[4]  <-  R[4]  I[4]  log
R[6]  <-  R[6]  R[5]  diff
R[6]  <-  R[6]  R[0]  log
R[4]  <-  R[4]  I[5]  exp
R[2]  <-  R[2]  R[4]  cos
R[6]  <-  R[6]  I[20]  sum
R[7]  <-  R[7]  I[11]  exp
R[3]  <-  R[3]  R[3]  diff
R[4]  <-  R[4]  R[2]  cos
R[1]  <-  R[1]  R[5]  exp
R[5]  <-  R[5]  R[6]  div
R[0]  <-  R[0]  R[0]  log
R[2]  <-  R[2]  I[15]  diff
R[5]  <-  R[5]  R[0]  sum
R[1]  <-  R[1]  R[0]  sum
R[4]  <-  R[4]  R[4]  sum
R[2]  <-  R[2]  I[14]  prod
R[2]  <-  R[2]  I[0]  sum
R[1]  <-  R[1]  R[6]  sum
R[7]  <-  R[7]  R[6]  diff
R[4]  <-  R[4]  I[3]  cos
R[2]  <-  R[2]  I[5]  mod
R[1]  <-  R[1]  R[4]  log
R[2]  <-  R[2]  I[18]  exp
R[0]  <-  R[0]  R[4]  cos
R[5]  <-  R[5]  R[4]  sum
R[0]  <-  R[0]  R[5]  mod
```

Figure 7.25: An Example of a GP Solution for Skype Detection Based on the Flow Feature Set

In summary, ML algorithms such as AdaBoost, GP and C5.0 select the most appropriate attributes (among the set given) to build their classifier model. This information is used to determine that five flow attributes from a set of 22 are used in the experiments for identifying Skype traffic using AdaBoost whereas sixteen flow attributes (again, out of 22) are used by GP (Figures 7.20 and 7.21).

Intuitively, what these algorithms learned from the data makes sense. In order to identify Skype traffic correctly the classifier needs to explore both forward and backward directions of the traffic. Each direction has its unique pattern given that an Initiator machine (starts connection) and a Corresponder machine (responds to the connection) operate differently. The features listed in Figures 7.20, 7.21 and 7.22 are what the learning model used to discover the Skype P2P VoIP encrypted tunnel. These features are separated into two groups: (i) attributes from Initiator to Corresponder (Forward direction); and (ii) attributes from Corresponder to Initiator (Backward direction). The attributes in the forward direction are based on the packet length and inter-arrival time since the forward direction depends on the initiator requesting information (e.g. min_fpktl, mean _fpktl, max_fpktl, std_fpktl, min_fiat, mean_fiat, max_fiat and std_fiat). The attributes in the backward direction are based on the packet length and the inter-arrival time since the backward direction is based on the Corresponder side responding to the Initiator requests (min_bpktl, mean _bpktl, max_bpktl, std_bpktl, min_biat, mean_biat, max_biat and std_biat).

For example, the minimum length for a packet is affected in part by the length of the request made by the Initiator. The standard deviation for a packet length is a measurement of request variation, i.e. the different commands used by the Initiator. In other words, the standard deviation measures the spread of the packet length, which can indicate the different commands which Initiators run. Consequently, the minimum and standard deviations of packet length measurements can shed some light on the behaviour of the Initiator in choosing commands to do the work and can provide more information for predicting what the payload might be. Such an indication can be very useful for network/system administrators since encrypted content can prevent the detection of anomalous activities, which can harm the system or steal sensitive data.

Furthermore, in these experiments, the GP model can process $\approx$ 760000 flow records per second, the AdaBoost model can process ≈ 430000 flow records per second while the C5.0 model can process ≈ 3.51e+09 flow records per second for classifying off-line data sets. Again, the models can classify more flow records per second if parallel computing is used to implement the signatures/model or they are run on a faster machine (more CPU power). Again, the parallel computing can increase the speed of the signatures in particular in the case of GP by computing each individuals/teams solution simultaneously.

7.2.4 FPR for each of the Best Solutions for Each Classifier

Figures 7.26, 7.27, 7.28, 7.29, 7.30 and 7.31 list the application flows which C5.0, GP and AdaBoost misclassify as Skype flows for the university traces employed (See Appendix C.2 for the tables). Given the fact that Skype runs over TCP or UDP, this is to be expected. The C5.0 and AdaBoost Flow-based classifiers tend to classify HTTP, DNS, P2P and 'OTHER' DNS traffic as Skype while GP tends to classify DNS, P2P and 'OTHER' as Skype since Skype uses UDP and TCP protocols to set up communication calls.

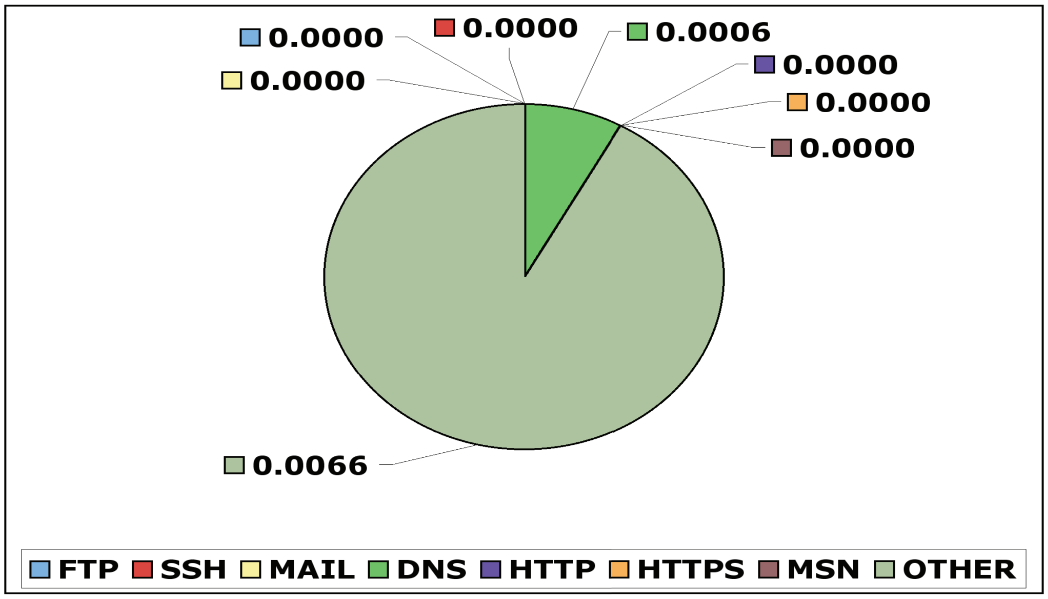


Figure 7.26: Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=0.7%).

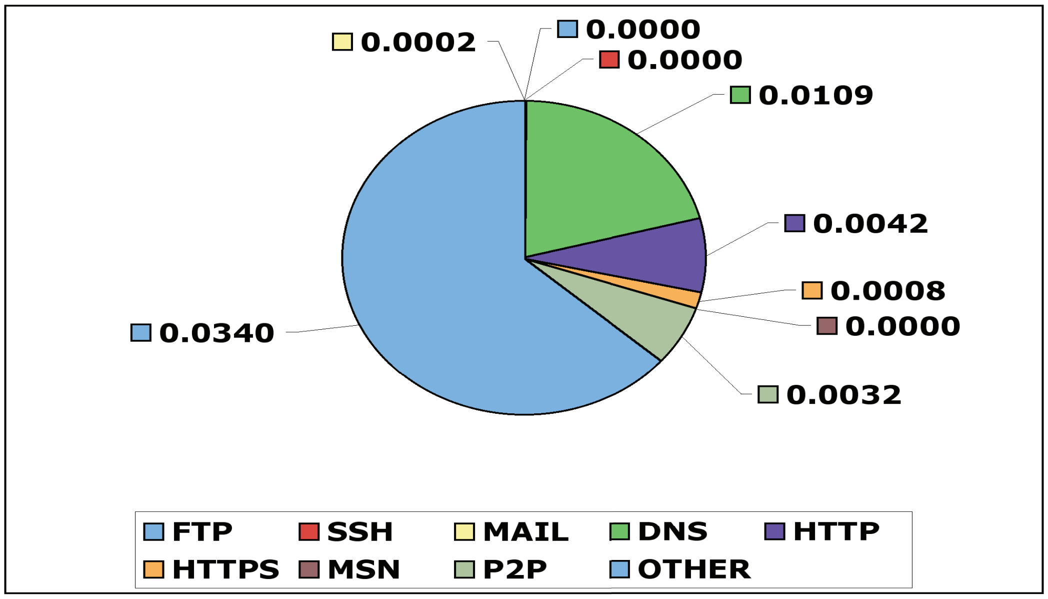


Figure 7.27: Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=6.1%).

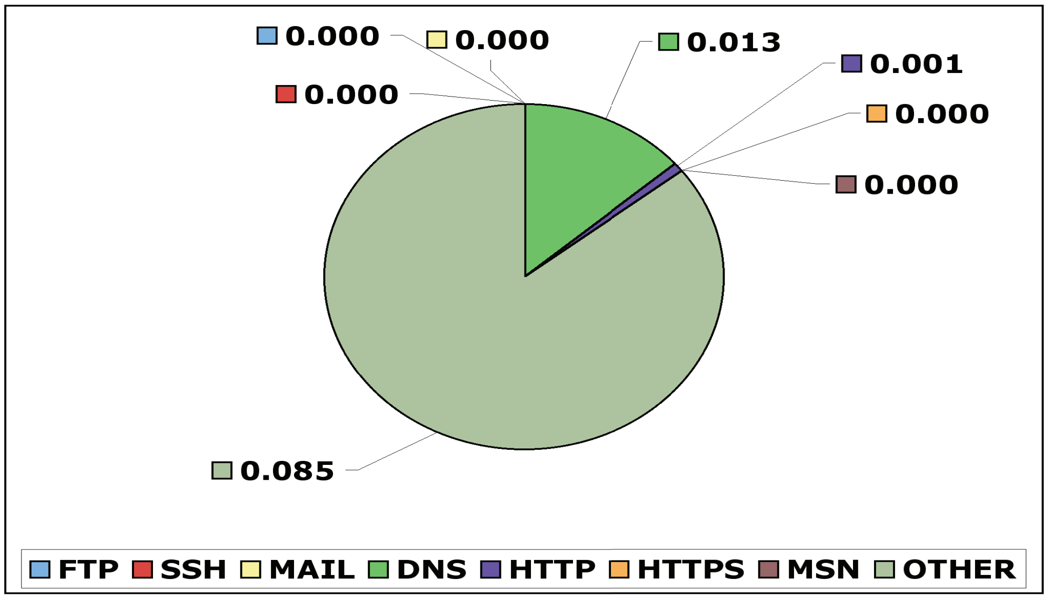


Figure 7.28: Applications Wrongly Classified as Skype by the GP-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=9.9%).

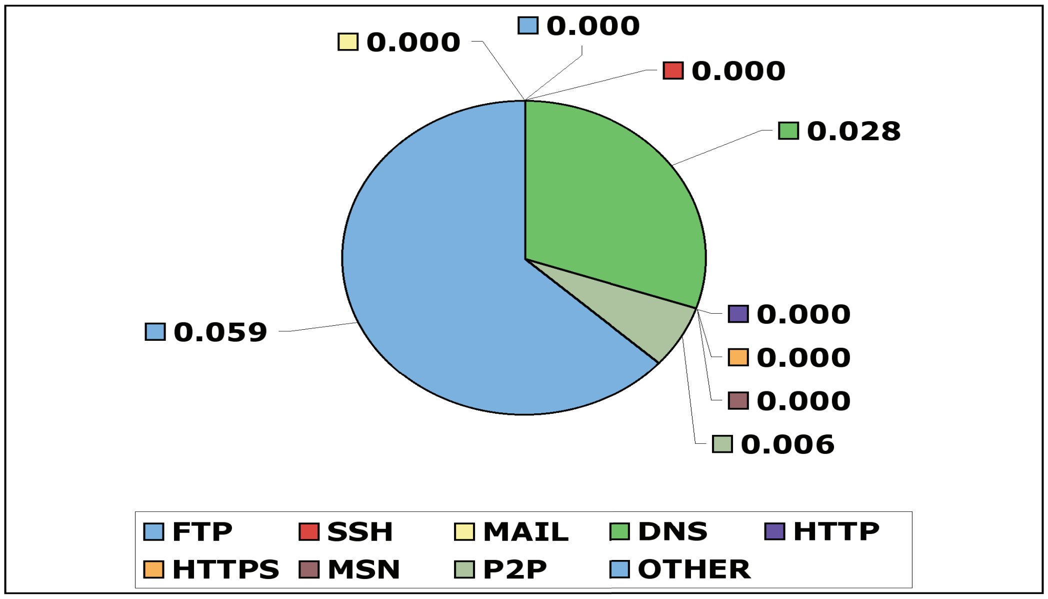


Figure 7.29: Applications Wrongly Classified as Skype by the GP-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=9.3%).

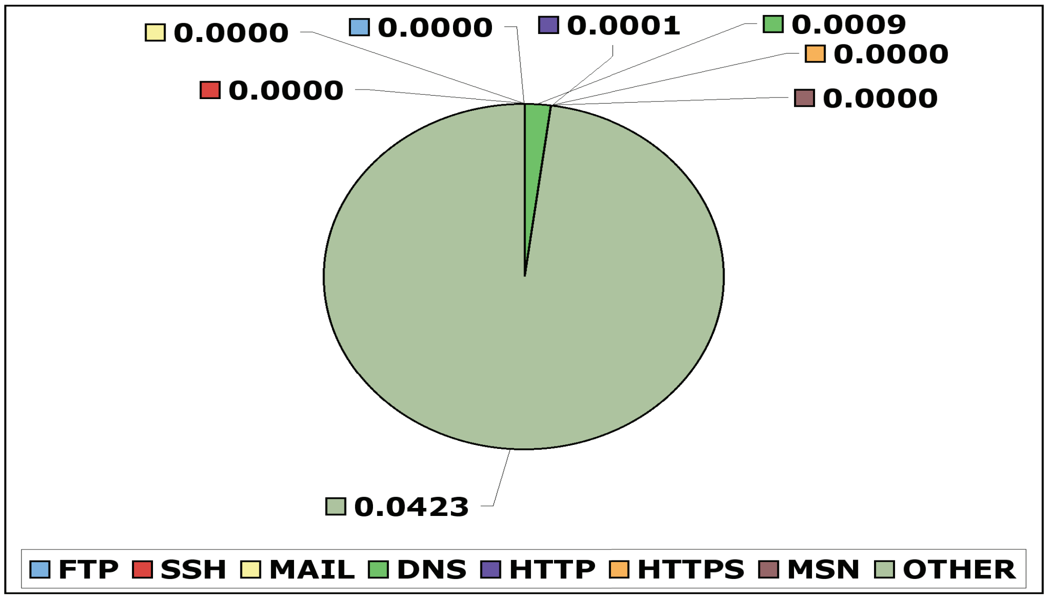


Figure 7.30: Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=4.3%).

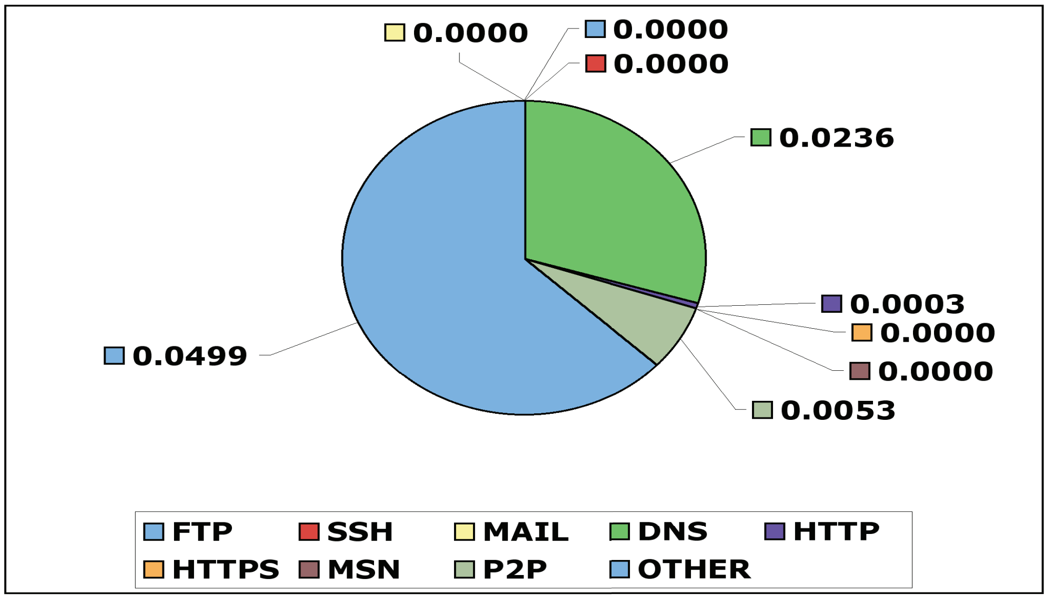


Figure 7.31: Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=7.9%).

7.3 Sensitivity Analysis of Configuration Parameters

This section discusses the effect of the parameters on the performance of the C5.0 and the GP based classifiers, since they are the top two performers in the experiments. The default parameters for the C5.0-based classifier have been used because the aim is not to find the best parameter set, but to investigate whether such a classifier will work out of the box and what its performance would be under such circumstances. The most important parameter to 'tune' under C5.0 is the pruning/confidence factor, where this has a direct impact on the resulting model complexity. Model complexity is related to generalization and performance/real-time operation. Figures 7.32, 7.33 and 7.34 summarize the impact of varying the confidence factor (CF) on the DR, FPR and the number of rules.[1] Essentially as the CF increases (less pruning) the resulting C5.0 model becomes more specific; the FPR decreases and the number of rules increases. However, in this particular data set, there is no change in DR. The best DR and FPR appear in the 5–45 CF interval. After this, any further improvement to FPR is negligible. Unfortunately, this comes with the higher number of rules under C5.0 (more than 270 rules). The sweet spot in this configuration happens to correspond to the default parameterization for C5.0.

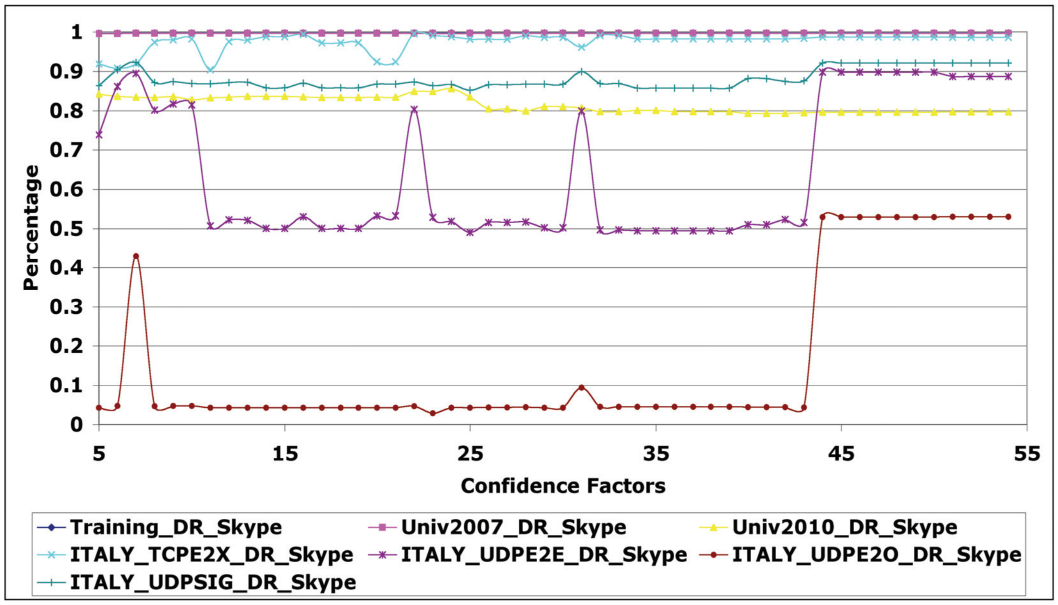


Figure 7.32: Sensitivity of C5.0 DR for Changing the Confidence Factor Parameter

[1]Skype detection task and flow features.

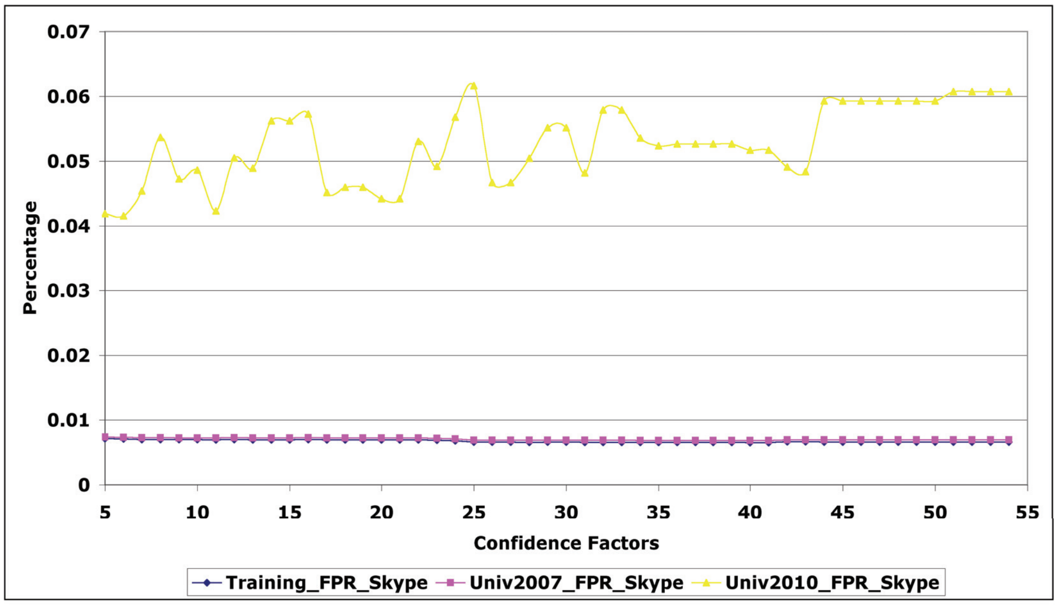


Figure 7.33: Sensitivity of C5.0 FPR for Changing the Confidence Factor Parameter

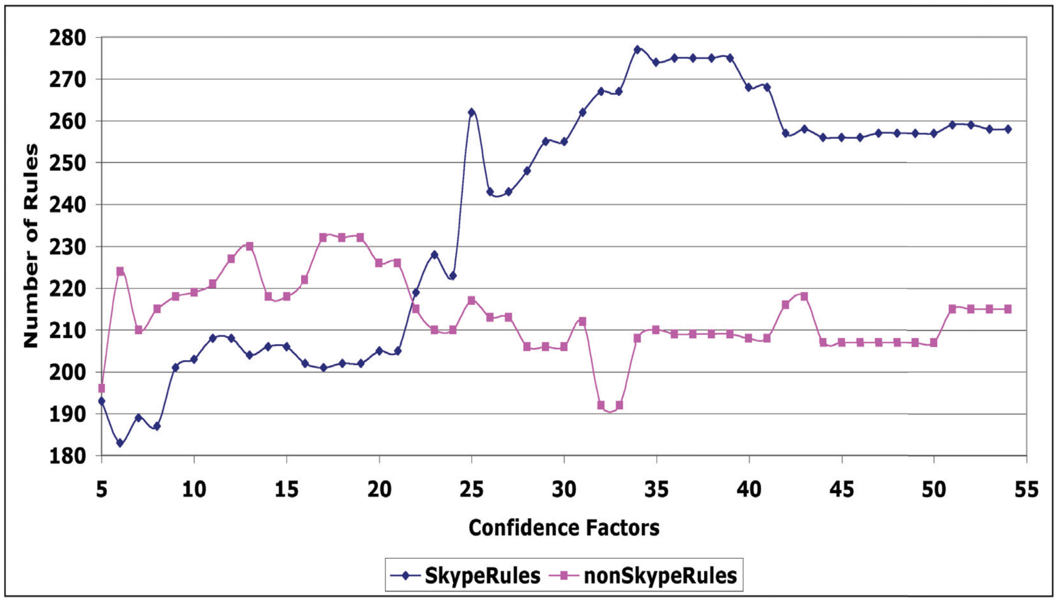


Figure 7.34: Sensitivity of the Number of C5.0 Rules for Changing the Confidence Factor Parameter

The principle property of variation in models of evolutionary computation, in GP case, is the seed parameters used to initialize the various stochastic processes behind population initialization and search operators. This source of variation is addressed through the use of multiple runs (fifty in this case) and is reported throughout the aforementioned experimental study. Secondary parameters might include the number of generations, population size, team size, program length, frequency of applying search operators and total number of registers. Of these, team size and program length are free to adapt. As long as runs do not approach the team size and program length limits, the choice of such parameters is independent of the parameterization, as is the case here. Any generational limit is fixed mostly to reflect the computational overhead of the task at hand. That is to say, as long as sufficient evaluations are performed to reach a performance plateau, the value of considering further evaluations needs to be traded off against the law of diminishing returns. The same can be said regarding the size of the point population used to sample from the wider training set. The larger the point population, the more significant the computational overhead is in performing any single fitness evaluation – particularly under the Pareto formulations as used here. Conversely the smaller the point population, the greater the sensitivity is to any single sample from the point population. Previous research has demonstrated that even when there is an order of magnitude difference in point population size there is little impact on the quality of the solutions evolved [112]. The remaining parameters have been studied extensively by the linear GP literature. In particular, the work of Brameier and Banzhaf identifies Max. Register Count as the single most significant parameter on linear GP; see Chapter 7 in [113]. Such a result is independent of problem domain as it reflects the ratio of registers to instructions per program. In short, the SBB algorithm adapts program length and team size to the problem domain at hand. Changes to point population size and team population size have little impact beyond some nominal figure (fifty individuals) and, within the limit, are sensitive to the total computational cost of performing a run. Similarly, a generational limit is imposed such that performance plateaus before a run are considered to have completed execution.

7.4 Limitations

No signature-based method in traffic classification can be perfect, given that there can always be some new (unseen) applications. Thus, the major challenge for traffic classification in general is evasion. All classification methods can be evaded. For example, a payload-based approach can be evaded by encrypting the packet payload and a port-based approach can be evaded by changing the port numbers dynamically. However, approaches based on flow statistics using packet size and inter-arrival time attributes are sensitive theoretically to altering these attributes. If attackers want to evade the proposed method they can modify the size of the packets in the entire connection by padding the packet payloads randomly. The accuracy of the proposed method might be decreased if those features which depend on packet size were modified from the application behaviour. However, it is not that easy to obfuscate application behaviour without presenting a large amount of overhead.

Another limitation of any classification system is obtaining (generating) the training data set. The generality and accuracy of the classifier depends on the quality of the training data sets. A meaningful and representative training data set is hard to find and generating one is resource and time consuming. Moreover, since the classifier generates the signatures automatically from the training data set, the accuracy of the classifier might decrease if the signatures/models from the trained classifiers are applied to network traffic which have different characteristics or behaviour (such as new applications which are developed or old applications which change their behaviour). Indeed, in such cases, the signatures/models need to be updated by retraining the classifiers. That is why it is very important to conduct robustness analysis on such classifiers. For instance, in this thesis, C5.0 signatures have the best consistent performance in the robustness criteria and the signatures can classify Skype P2P VoIP traffic in a trace robustly if the characteristic of the flow features in the trace fall within the range described in Table 7.4 (See Appendix D.1 for the figures).

Table 7.4: Ranges of Values for each of the Flow Features for the Signatures Generated by C5.0

Feature Name	**Range**
Duration	*value* <= 5987061
fpackets	*value* <= 1218
fbytes	*value* <= 2512
bpackts	*value* <= 1011
bbytes	*value* <= 2877
min_fiat	472914 <= *value* > 472914
mean_fiat	47 <= *value* > 420485
max_fiat	47 <= *value* > 26620
std_fiat	67009 <= *value* > 6202
min_biat	*value* <= 569517
mean_biat	*value* <= 330261
max_biat	*value* <= 20162
std_biat	*value* <= 85003
min_fpkt	*value* > 30
mean_fpkt	512 <= *value* > 47
max_fpkt	344 <= *value* > 47
std_fpkt	*value* <= 548
min_bpkt	870 <= *value* > 30
mean_bpkt	1299 <= *value* > 38
max_bpkt	1341 <= *value* > 42
std_bpkt	633 < *value* > 587

Chapter 8

Ensemble Learning for Skype Identification

The common model for ensemble learning is depicted in Figure 8.1, where each classifier (*1* to N) is trained on the input data and produces an output. Then, the outputs of all classifiers (*1* to N) are combined to produce the ensemble learning predication.

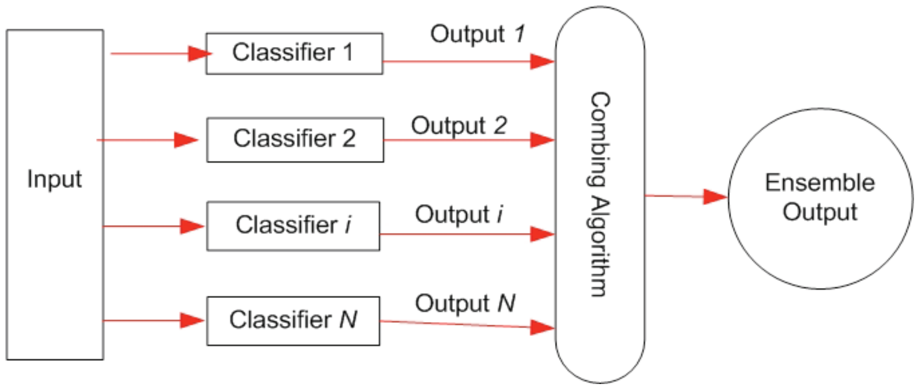


Figure 8.1: Common Ensemble Learning Model

There exist several methods for building an ensemble learning model such as boosting, bagging and stack generalization. Boosting methods consist of training a classifier sequentially on training exemplars which have been filtered by the (*1* to N) classifiers. A famous example of a boosting method is the AdaBoost classifier designed by Freund and Schapire [82]. The bagging (Bootstrap) method involves the aggregation of multiple classifiers (predictors) by using voting schema to generate an aggregated classifier [114] which is based on the bootstrap technique. The third common method of ensemble learning is stack generalization. Stack generalization consists of two layers in which the outputs of the first layer are employed as inputs to a second layer [115]. The first layer consists of different classifiers with different parameters which are trained mainly on the same training data sets. Then, the outputs of the first layer are fed to the second layer where a combining model is used to produce the ensemble learning prediction value. Since each one of the ML algorithms (AdaBoost, C5.0 and GP) employed in this thesis has a different learning approach, the stack generalization method is the most appropriate method to be chosen out of the three common ensemble learning methods. Therefore, combining AdaBoost, C5.0 and GP into one model using the stack generalization method where a Neural Network is employed at the second layer. The reason behind using a Neural Network (NN) based approach for ensemble learning is that the usefulness of using a NN with ensemble learning has been demonstrated already ([116, 117]) and is widely known.

8.1 The Proposed Ensemble Learner Neural Network Model for Skype Identification

To combine the three machine learning approaches the following steps are necessary. The first step is to transform the flow data sets from 22 features (the number of flow features) to new data sets where the number of features depends on the solutions produced by the three machine learning algorithms. The second step involves building the new ensemble learning model using a Neural Network based approach on the newly transformed data sets.

Figure 8.2 depicts the first step. The solutions generated by C5.0, AdaBoost and GP are represented as “if-then statement” rules. Then, these rules will be used to transfer the flow training data sets from 22 features (columns) into a new data set in which each column represents a binary output since a binary classification (either Skype or non-Skype) is performed.

The second step is where the selection of the optimal rules happens. A Neural Network (NN) is run on the new data sets, which consist of the output of a diverse pool of rules (solutions). The hidden layer and the output layer of the NN are deployed with a sigmoid function. The reason for using sigmoid and setting the output range for the sigmoid between -0.8 and 0.8 is to prevent over-fitting (over learning). The Levenberg-Marquardt algorithm [118] with conjugate gradient approach, which is a variant of the backpropagation algorithm, is used for the NN. At each generation the rules which minimize the error rate on the training data set are kept based on a threshold. Hence, with each generation the size of the data set is reduced.

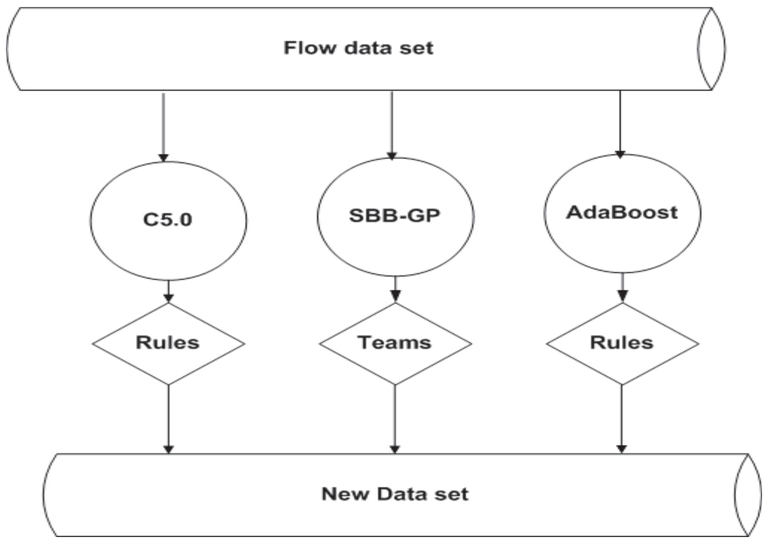


Figure 8.2: Representing the Flow Data Set to be Used by the Ensemble Learning Model

```
begin
  $Threshold = 0$
  for $i := 1$ to $sizeofdata$ step 1 do
      $Threshold = Threshold + 0.005$
      $TraintheNeuralNetwork(NN)$
      if $validationError > StoppingCriteria$ then $Break$
      fi
      $GetWeightFromNN$
      if $weight > Threshold$ then $KeepRule$
      fi
  end
```

Figure 8.3: Second Step of the Ensemble Learning Model

The pseudo code is given in Figure 8.3. At the first generation, the threshold is initialized to 0.005. Then, training of the Neural Network (NN) starts. Two outputs of the NN are considered. The validation error and the weight of each input after training. The first output is used as the stopping criterion for the NN. The second output is a vector of weights for each of the inputs. Only the inputs (rules) which pass the threshold are kept. A MATLAB [119] version 7.5.0.338 on a Pentium 4 2.4 GHz Core 2 Duo chip with 2 GB of RAM was used to train the NN and test the ensemble model prediction on the test data sets.

8.2 Results of the Ensemble Learning Model

Since the performance of the flow feature set is better than the packet header feature set, the models generated by AdaBoost, C5.0 and GP-based on the flow feature set were chosen for the Ensemble model experiments. Due to the memory limitation on the machine where MATLAB runs, the training data set used in Chapter 5 could not be used (1,739,588 flows). Therefore, in these experiments, the training data set generated by the uniform random sampling of 30K records from Chapter 4 was employed during training. As done in the previous experiments, the solution robustness was assessed by training on a data set from one location (Univ2007 training partition) and testing on data sets from other locations (Univ2007 test partition, Univ2010, ITALY, IPv6 and NIMS traces). In short, the performances of AdaBoost, GP and C5.0 as individual classifiers were compared against the performance of the ensemble model where the three ML algorithms are employed together to build the ensemble learning prediction model. Furthermore, the three classifier based ensemble learning model is compared as well against an ensemble model based on only one classifier.

Table 8.1 shows the performance on the training data and the test data sets of the University traces for the Ensemble Model when a NN model is used to combine all classifiers (EL 3-classifiers), for C5.0 rules only (EL C5.0), for AdaBoost rules only (EL AdaBoost) and for GP individuals only (EL GP). Table 8.2 shows the performance of C5.0, GP and AdaBoost on the same data sets (as in Table 8.1) but without ensemble learning. In these experiments (Table 8.2) the classifier giving the best performance is C5.0 with 98.8% DR and 1.5% FPR on the Univ2007 traces and with 86.6% DR and 5.9% FPR on the Univ2010 traces. The new approach (EL 3-classifiers) increases the DR for the Univ2007 data set by 1% compared to the best classifier, C5.0, and reduces the FPR on the Univ2010 traces by 1% again compared to running the C5.0-based classifier on its own. The EL 3-classifier model achieves 99.0% DR and 1.5% FPR on the Univ2007 traces and 84.4% DR and 4.9% FPR on the Univ2010 traces. According to these results, it is clear that the combining of the three machine learning algorithms is better than each machine learning algorithm on its own on the university traces.

Since finding the optimum training data set for testing the robustness of the classifiers is a hard task as illustrated in Chapter 4, none of the three machine learning algorithms trained on this smaller data set generalized well to the unseen data sets from other networks/infrastructures. For the ITALY traces, GP has the best performance compared to AdaBoost, C5.0 and the ensemble model (Tables 8.3 and 8.4). For the NIMSII traces, C5.0 has the best performance on the GTALK and ZFONE traces while the performance of the EL 3-classifiers was the best on the PRIMUS traces (Tables 8.5 and 8.6). AdaBoost performs the best on the NIMSIII traces (Tables 8.7 and 8.8) while the AdaBoost and ensemble learning models achieves 100% DR for the IPv6 traces (Tables 8.9 and 8.10).

Table 8.1: Results of Ensemble Learning (EL) for the Flow-based Feature Set for Skype Detection

	EL 3-classifiers		**EL C5.0**		**EL AdaBoost**		**EL GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample (subset of Univ2007)								
non-Skype	0.989	0.006	0.988	0.005	0.925	0.023	0.985	0.006
Skype	0.994	0.011	0.995	0.012	0.977	0.075	0.994	0.015
Univ2007 Test data sets								
non-Skype	0.985	0.010	0.985	0.010	0.984	0.023	0.980	0.010
Skype	0.990	0.015	0.990	0.015	0.977	0.016	0.990	0.020
Univ2010 Test data sets								
non-Skype	0.951	0.156	0.862	0.138	0.866	0.141	0.953	0.173
Skype	0.844	0.049	0.862	0.138	0.859	0.134	0.827	0.047

Table 8.2: Best Model Results on the Training Data Set for the Flow-based Feature Set for Skype Detection

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training Sample (subset of Univ2007)						
Non-SKYPE	0.988	0.008	0.955	0.115	0.924	0.023
SKYPE	0.992	0.012	0.885	0.045	0.977	0.076
Univ2007 Test data sets						
Non-SKYPE	0.985	0.012	0.957	0.120	0.924	0.025
SKYPE	0.988	0.015	0.880	0.043	0.975	0.076
Univ2010 Test data sets						
Non-SKYPE	0.941	0.134	0.931	0.187	0.939	0.135
SKYPE	0.866	0.059	0.813	0.069	0.865	0.061

Table 8.3: Results of Ensemble Learning (EL) for the Flow-based Feature Set for Skype Detection

	EL 3-classifiers		**EL C5.0**		**EL AdaBoost**		**EL GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**
ITALY TCPE2X Test data sets								
non-Skype	N/A	1.000	N/A	1.000	N/A	1.000	N/A	1.000
Skype	0.000	N/A	0.000	N/A	0.000	N/A	0.000	N/A
ITALY UDPE2E Test data sets								
non-Skype	N/A	0.761	N/A	0.416	N/A	0.912	N/A	0.605
Skype	0.239	N/A	0.584	N/A	0.088	N/A	0.395	N/A
ITALY UDPE2O Test data sets								
non-Skype	N/A	0.953	N/A	0.061	N/A	0.993	N/A	0.624
Skype	0.047	N/A	0.939	N/A	0.007	N/A	0.376	N/A
ITALY UDPSIG Test data sets								
non-Skype	N/A	0.486	N/A	0.311	N/A	0.328	N/A	0.401
Skype	0.514	N/A	0.689	N/A	0.672	N/A	0.599	N/A

Table 8.4: Best Model Results for the Flow-based Feature Set for Skype Detection

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
ITALY TCPE2X Test data sets						
Non-SKYPE	N/A	1.000	N/A	1.000	N/A	1.000
SKYPE	0.000	N/A	0.000	N/A	0.000	N/A
ITALY UDPE2E Test data sets						
Non-SKYPE	N/A	0.994	N/A	0.794	N/A	0.008
SKYPE	0.006	N/A	0.206	N/A	0.992	N/A
ITALY UDPE2O Test data sets						
Non-SKYPE	N/A	0.994	N/A	0.976	N/A	0.000
SKYPE	0.006	N/A	0.024	N/A	1.000	N/A
ITALY UDPSIG Test data sets						
Non-SKYPE	N/A	0.399	N/A	0.302	N/A	0.022
SKYPE	0.601	N/A	0.698	N/A	0.978	N/A

Table 8.5: Results of Ensemble Learning (EL) for the Flow-based Feature Set for Skype Detection

	EL 3-classifiers		**EL C5.0**		**EL AdaBoost**		**EL GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**
NIMSII GTALK2009 Test data sets								
non-Skype	0.985	N/A	0.990	N/A	0.990	N/A	0.983	N/A
Skype	N/A	0.015	N/A	0.010	N/A	0.010	N/A	0.017
NIMSII PRIMUS2009 Test data sets								
non-Skype	0.997	N/A	0.991	N/A	0.990	N/A	0.985	N/A
Skype	N/A	0.003	N/A	0.009	N/A	0.010	N/A	0.015
NIMSII ZFONE2009 Test data sets								
non-Skype	0.959	N/A	0.984	N/A	0.984	N/A	0.918	N/A
Skype	N/A	0.041	N/A	0.016	N/A	0.016	N/A	0.082

Table 8.6: Best Model Results for the Flow-based Feature Set for Skype Detection

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
NIMSII GTALK2009 Test data sets						
Non-SKYPE	0.990	N/A	0.999	N/A	0.947	N/A
SKYPE	N/A	0.010	N/A	0.001	N/A	0.053
NIMSII PRIMUS2009 Test data sets						
Non-SKYPE	0.989	N/A	1.000	N/A	0.170	N/A
SKYPE	N/A	0.011	N/A	0.000	N/A	0.830
NIMSII ZFONE2009 Test data sets						
Non-SKYPE	0.983	N/A	0.938	N/A	0.470	N/A
SKYPE	N/A	0.017	N/A	0.062	N/A	0.530

Table 8.7: Results of Ensemble Learning (EL) for the Flow-based Feature set for Skype Detection

	EL 3-classifiers		**EL C5.0**		**EL AdaBoost**		**EL GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**
NIMSIII GTALK2010 Test data sets								
non-Skype	0.945	N/A	0.952	N/A	0.952	N/A	0.942	N/A
Skype	N/A	0.055	N/A	0.048	N/A	0.048	N/A	0.058
NIMSIII PRIMUS2010 Test data sets								
non-Skype	0.814	N/A	0.872	N/A	0.839	N/A	0.918	N/A
Skype	N/A	0.186	N/A	0.128	N/A	0.161	N/A	0.082
NIMSIII YAHOO2010 Test data sets								
non-Skype	0.915	N/A	0.923	N/A	0.924	N/A	0.883	N/A
Skype	N/A	0.085	N/A	0.077	N/A	0.076	N/A	0.117
NIMSIII RADIO2010 Test data sets								
non-Skype	0.997	N/A	0.997	N/A	0.997	N/A	0.997	N/A
Skype	N/A	0.003	N/A	0.003	N/A	0.003	N/A	0.003
NIMSIII TORRENT2010 Test data sets								
non-Skype	0.890	N/A	0.758	N/A	0.782	N/A	0.904	N/A
Skype	N/A	0.110	N/A	0.242	N/A	0.218	N/A	0.096
NIMSIII TV2010 Test data sets								
non-Skype	0.992	N/A	0.992	N/A	0.991	N/A	0.992	N/A
Skype	N/A	0.008	N/A	0.008	N/A	0.009	N/A	0.008
NIMSIII VPN 2010 Test data sets								
non-Skype	0.986	N/A	1.000	N/A	1.000	N/A	0.986	N/A
Skype	N/A	0.014	N/A	0.000	N/A	0.000	N/A	0.014

Table 8.8: Best Model Results for the Flow-based Feature Set for Skype Detection

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
NIMSIII GTALK2010 Test data sets						
Non-SKYPE	0.952	N/A	0.997	N/A	0.941	N/A
SKYPE	N/A	0.048	N/A	0.003	N/A	0.059
NIMSIII PRIMUS2010 Test data sets						
Non-SKYPE	0.871	N/A	0.650	N/A	0.336	N/A
SKYPE	N/A	0.129	N/A	0.350	N/A	0.664
NIMSIII YAHOO2010 Test data sets						
Non-SKYPE	0.924	N/A	0.933	N/A	0.290	N/A
SKYPE	N/A	0.076	N/A	0.067	N/A	0.710
NIMSIII RADIO2010 Test data sets						
Non-SKYPE	0.997	N/A	0.997	N/A	0.997	N/A
SKYPE	N/A	0.003	N/A	0.003	N/A	0.003
NIMSIII TORRENT2010 Test data sets						
Non-SKYPE	0.897	N/A	0.934	N/A	0.562	N/A
SKYPE	N/A	0.103	N/A	0.066	N/A	0.438
NIMSIII TV2010 Test data sets						
Non-SKYPE	0.991	N/A	1.000	N/A	0.992	N/A
SKYPE	N/A	0.009	N/A	0.000	N/A	0.008
NIMSIII VPN2010 Test data sets						
Non-SKYPE	0.987	N/A	1.000	N/A	0.959	N/A
SKYPE	N/A	0.013	N/A	0.000	N/A	0.041

Table 8.9: Results of Ensemble Learning (EL) for the Flow-based Feature Set for Skype Detection

	EL 3-classifiers		**EL C5.0**		**EL AdaBoost**		**EL GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**
IPv6 2000 Test data sets								
non-Skype	1.000	N/A	1.000	N/A	1.000	N/A	1.000	N/A
Skype	N/A	0.000	N/A	0.000	N/A	0.000	N/A	0.000
IPv6 2009 Test data sets								
non-Skype	1.000	N/A	1.000	N/A	1.000	N/A	1.000	N/A
Skype	N/A	0.000	N/A	0.000	N/A	0.000	N/A	0.000

Table 8.10: Best Model Results for the Flow-based Feature Set for Skype Detection

	C5.0		**AdaBoost**		**GP**	
	DR	**FPR**	**DR**	**FPR**	**DR**	**FPR**
IPv6 2000 Test data sets						
Non-SKYPE	0.999	N/A	1.000	N/A	1.000	N/A
SKYPE	N/A	0.001	N/A	0.000	N/A	0.000
IPv6 2009 Test data sets						
Non-SKYPE	0.889	N/A	1.000	N/A	1.000	N/A
SKYPE	N/A	0.111	N/A	0.000	N/A	0.000

8.3 Discussion of Results

Limited by the computer memory and MATLAB license, it was not possible to use the training data set employed in Chapters 4 and 5 to test the affect of the ensemble learning model on improving the robustness of the machine learning algorithms. However, the results from the ensemble learning approach are promising and show the effectiveness of combining three machine learning based systems using a Neural Network for classifying Skype encrypted traffic on the university traces. The ensemble learning approach increases the DR by 1% and reduces the FPR by 1% on the University test data sets compared to the best classifier (in this case C5.0) based on the flow feature set.

Chapter 9

Multi-Class Classification for VoIP Applications

In this chapter, an approach which is based on ML algorithms and statistical flow-based features is described and demonstrated in order to find signatures for classifying more than one VoIP application. Three VoIP applications were employed: Skype, Gtalk and Primus softphone. To show the effectiveness of the proposed approach, evaluations are performed on different training and test data sets. The solution's robustness is assessed by sampling the training data from two data sets (Univ2007 and NIMSII traces) while testing occurred on traces from different locations (Univ2007 and NIMSII test partitions, Univ2010, IPv6 and ITALY, which were captured in 2007, 2009, 2010, 2000-2009 and 2006, respectively). In these experiments, the training data set is labelled into multi-classes depending on the VoIP applications (SKYPE, GTALK, PRIMUS, and non-VoIP). It should be noted here that 6% of the GTALK2009 and PRIMUS2009 data sets were sampled and added to the training data set as described in Chapter 5. Thus, the training data set consists of SKYPE, non-VoIP, GTALK, and PRIMUS applications where each of them contains the following flow numbers: 646521, 1235055, 11417, and 451, respectively.

As discussed earlier in this thesis, the proposed approach to identifying VoIP traffic such as Skype, Gtalk and Primus is data-driven, and therefore, all the possible attributes/features are presented to the learning algorithms employed. In doing so the aim is to examine: (i) which features will be considered the most appropriate by each learning algorithm and (ii) which features will be chosen by all of the learning models employed in this research. The answer to these two questions will provide the subset of features which could be the most robust/generalized as well as the most appropriate ones which can be used for real-life network traffic traces.

9.1 Results of the VoIP Multi-Class Classification

Figures 9.1, 9.2 and 9.3 summarize solutions for the three machine learning algorithms on the training data sets. All model construction takes place on the training partition (Univ2007 and NIMSII). Testing evaluation is carried on under the Univ2007 and NIMSII test partitions and the Univ2010, ITALY, NIMSIII and IPv6 traces which are not seen during training. Normally, the Univ2007 test partition and NIMSII test partitions will reflect the training behavior more closely than the other test network traces.

The violin plots demonstrate the diversity of performance in terms of DR and FPR for all fifty models on the training data sets for each machine learning algorithm. The results presented in Figure 9.1 and the one-way ANOVA statistical analysis test in Appendix B.3.3 illustrates that the C5.0-based classification approach is much better on average than the other algorithms employed in identifying the Skype, Gtalk and Primus flow traffic based on the training data set. GP is competitive with C5.0 for all classes (Figure 9.2) and has different ranges of DR and FPR which implies that the GP-based classifier finds different solutions at different runs. AdaBoost performs the worst (Figure 9.3) failing to identify any of the Gtalk.

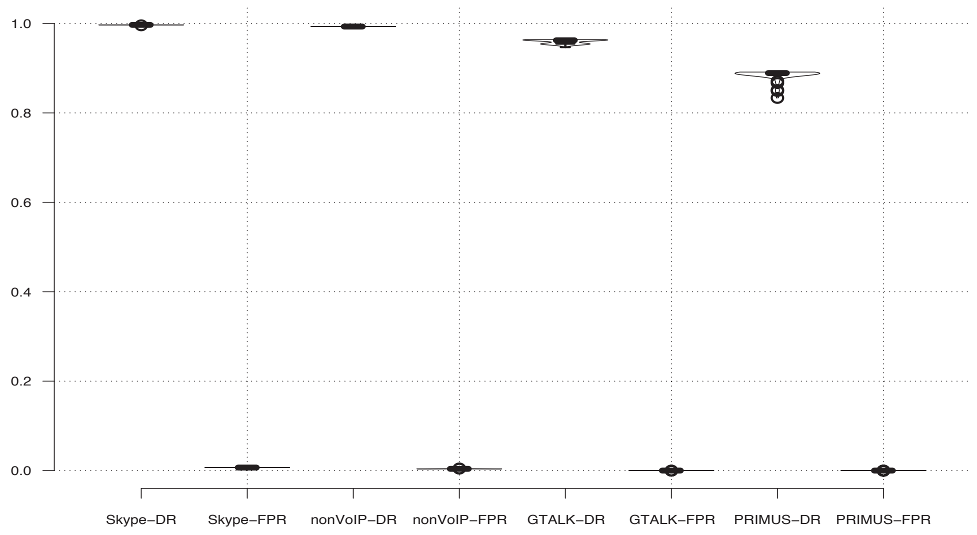


Figure 9.1: DR and FPR Results for C5.0 or Multi-Class VoIP for the Flow-based Feature Set on the Training Data Sets

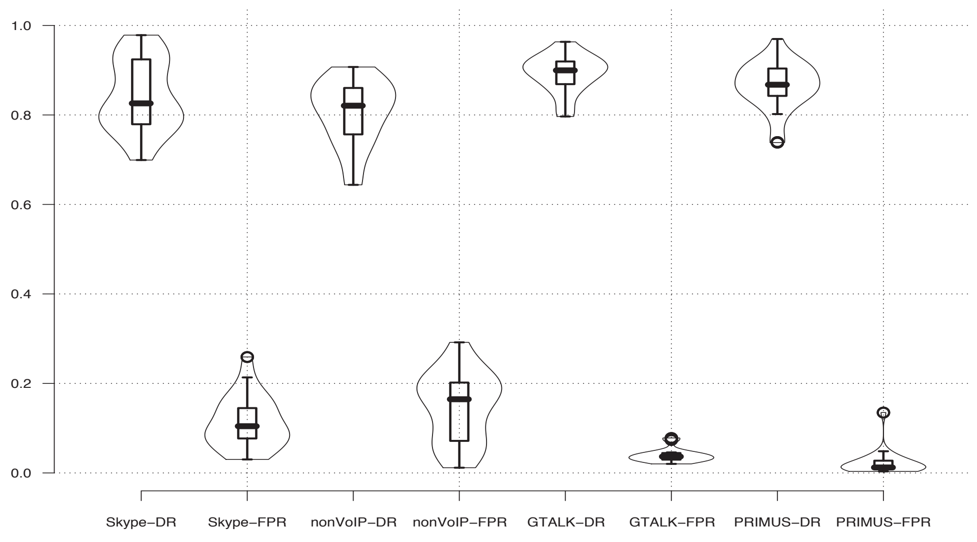


Figure 9.2: DR and FPR Results for GP or Multi-Class VoIP for the Flow-based Feature Set on the Training Data Sets

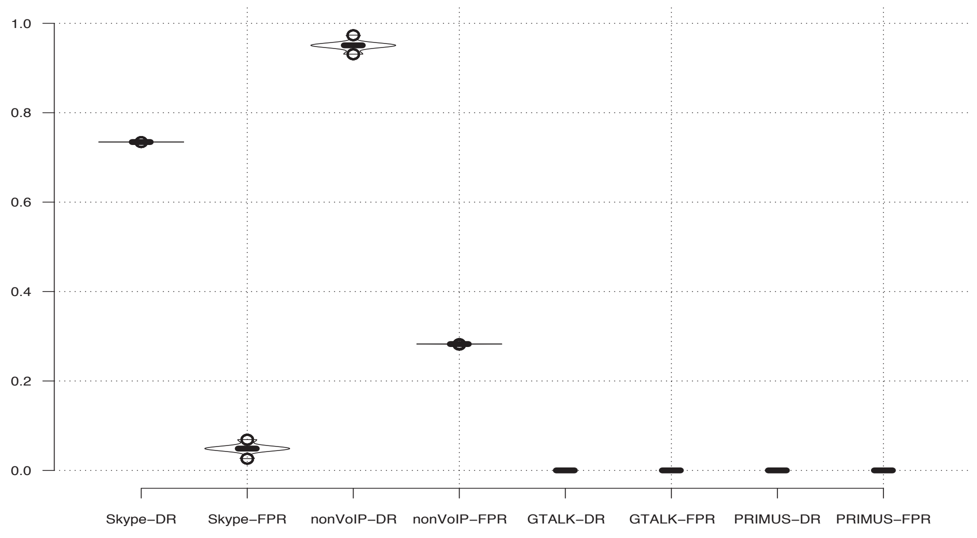


Figure 9.3: DR and FPR Results for AdaBoost or Multi-Class VoIP VoIP for the Flow-based Feature Set on the Training Data Sets

To select the best trained model for each ML algorithm, their training performance is plotted in a scatter plot for each. Figures 9.4, 9.5, 9.6, 9.7, 9.8, 9.9, 9.10, 9.11, 9.12 summarize the solutions for the C5.0, GP and AdaBoost algorithms. The best performing solution in terms of high DR and low FPR is selected out of these non-dominated solutions for GP, AdaBoost and C5.0 and then evaluated on the test data sets. For C5.0, there are eight solutions which are non-dominated, for GP, seven and for AdaBoost, one.

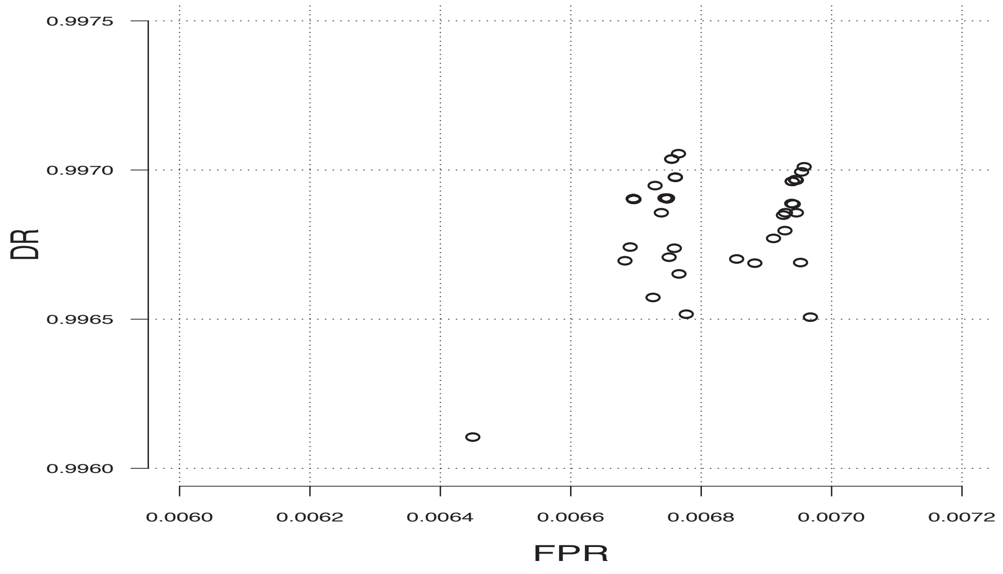


Figure 9.4: Scatter for the C5.0-based Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR)

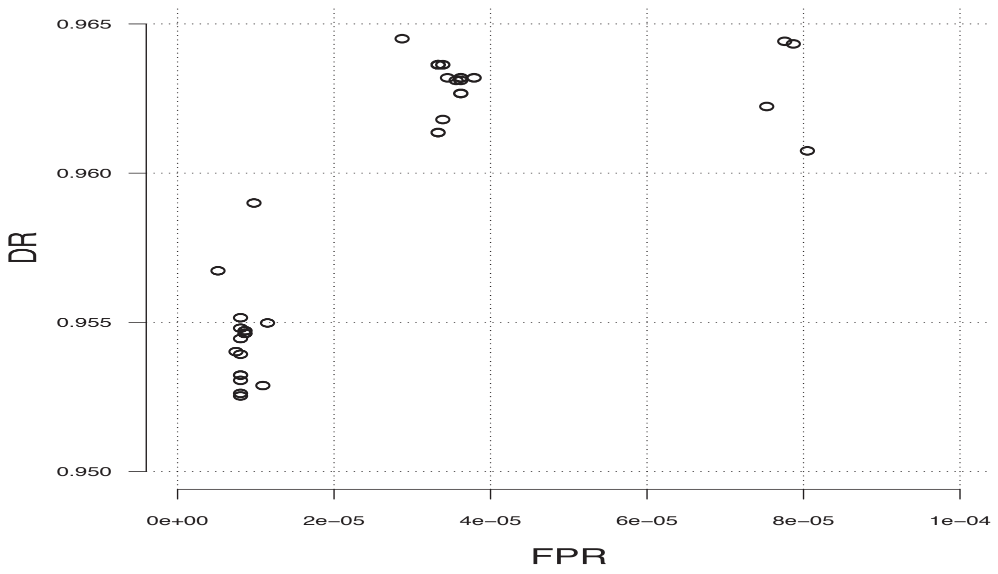


Figure 9.5: Scatter for the C5.0-based Classifier for Training Performance using the Flow-based Feature set for Gtalk Detection (DR versus FPR)

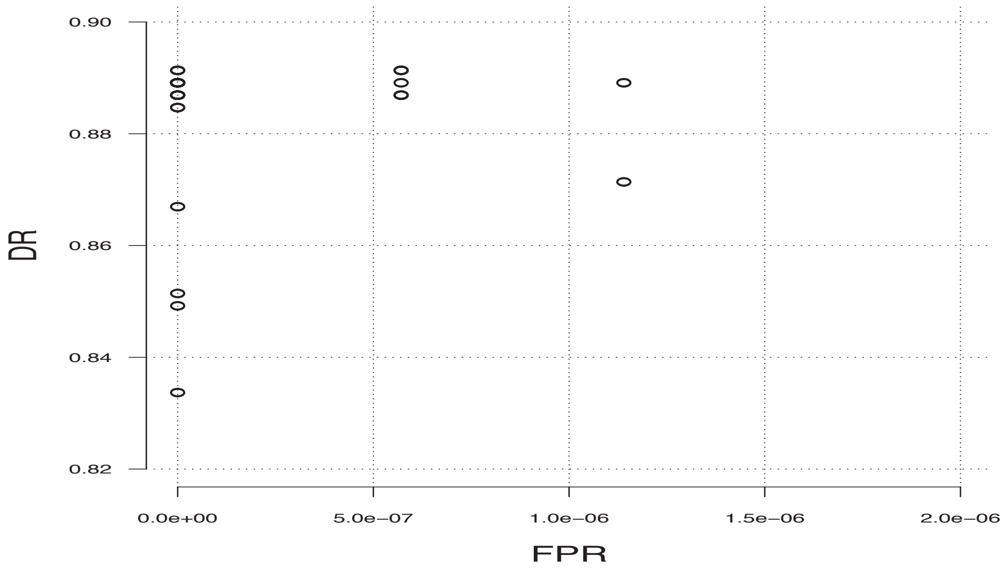


Figure 9.6: Scatter for the C5.0-based Classifier for Training Performance Using the Flow-based Feature Set for Primus Detection (DR versus FPR)

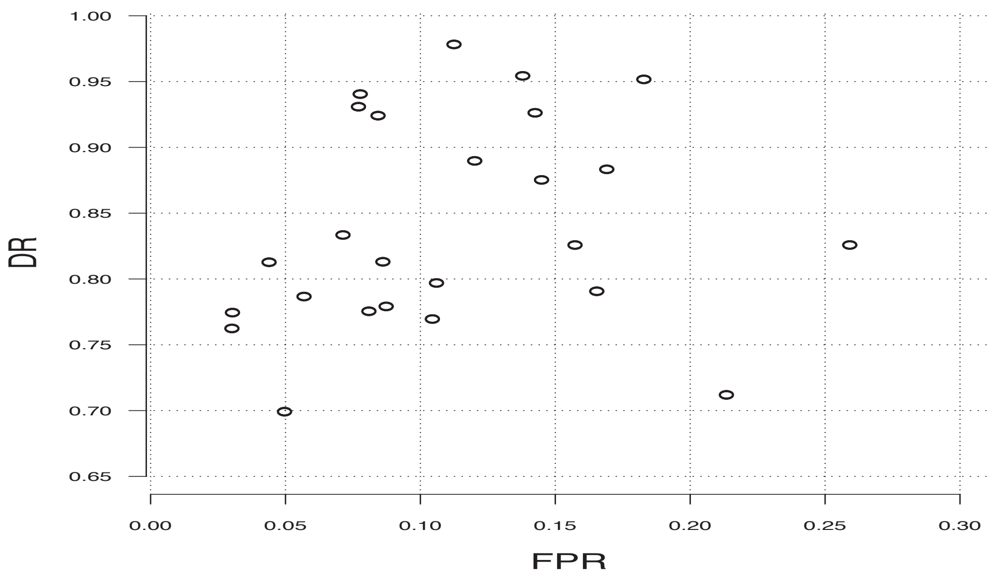


Figure 9.7: Scatter for the GP-based Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR)

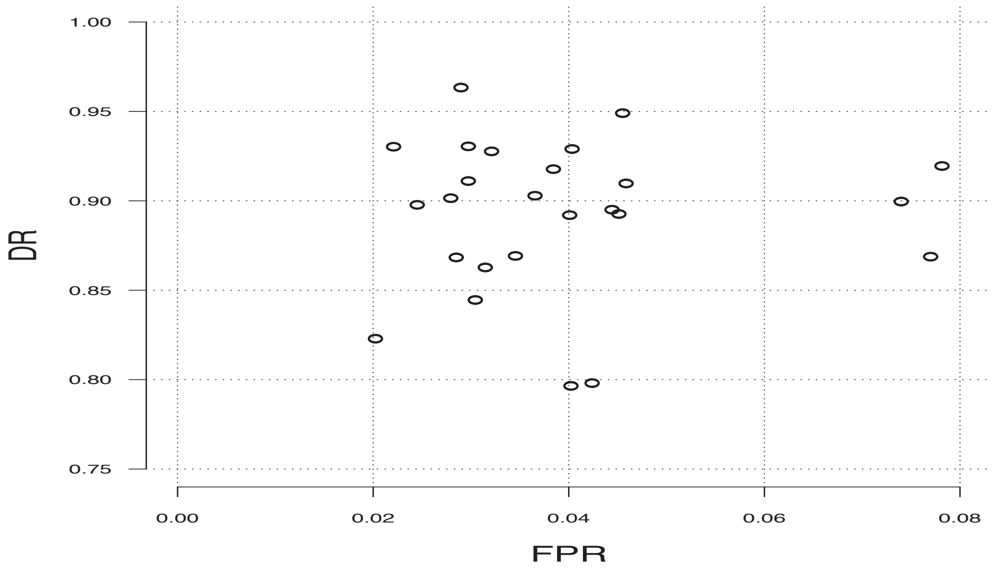


Figure 9.8: Scatter for the GP-based Classifier for Training Performance Using the Flow-based Feature Set for Gtalk Detection (DR versus FPR)

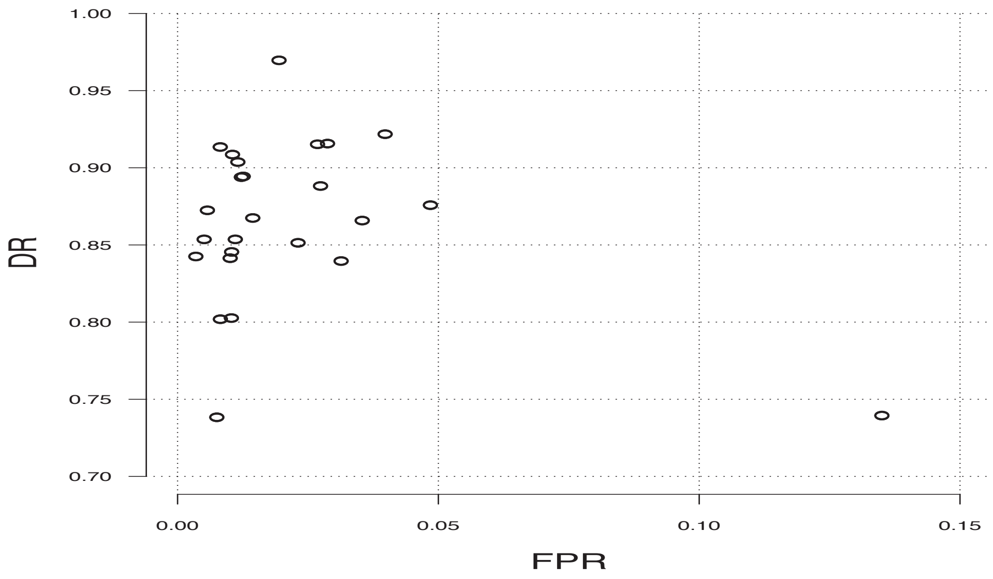


Figure 9.9: Scatter for the GP-based Classifier for Training Performance Using the Flow-based Feature Set for Primus Detection (DR versus FPR)

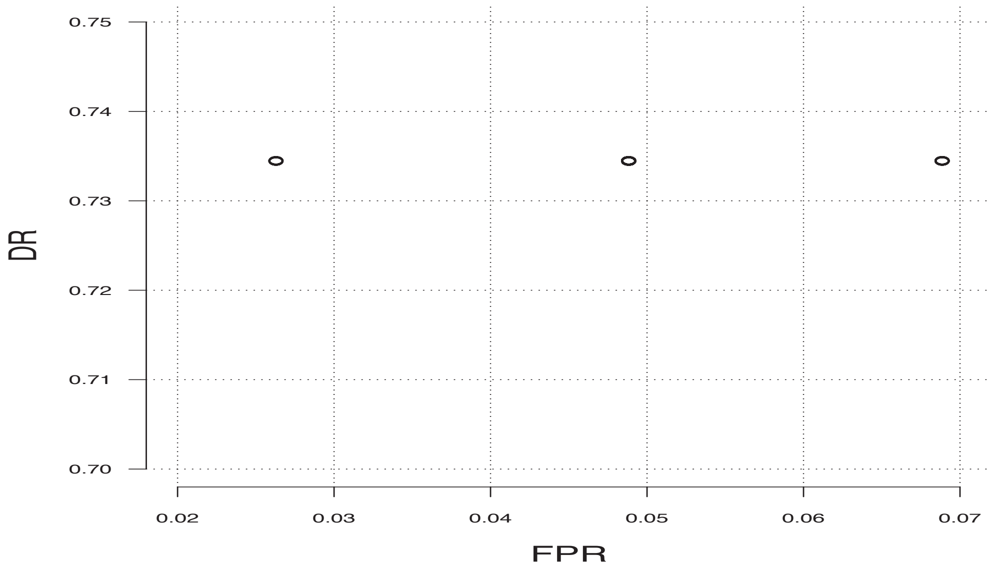


Figure 9.10: Scatter for the AdaBoost-based Classifier for Training Performance Using the Flow-based Feature Set for Skype Detection (DR versus FPR)

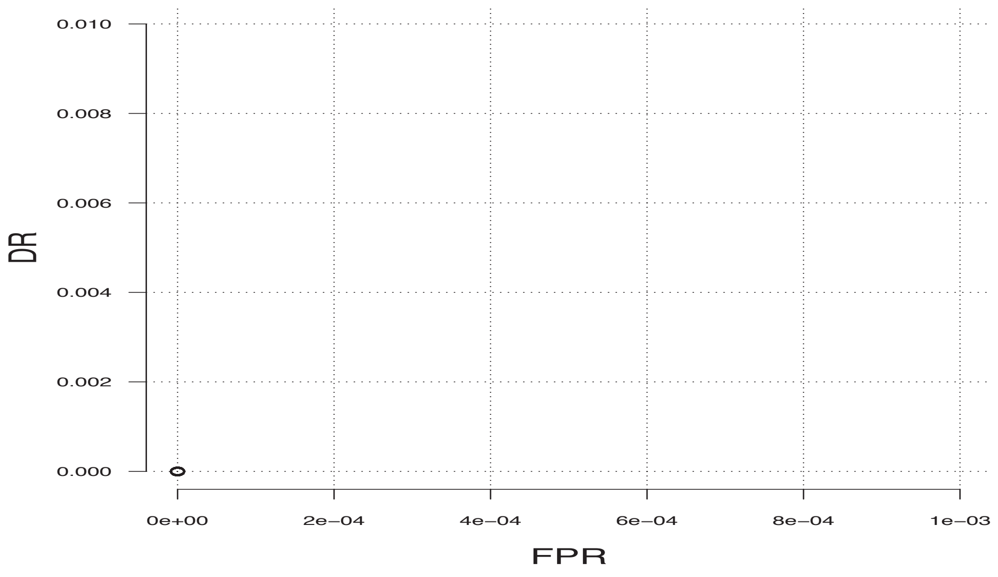


Figure 9.11: Scatter for the AdaBoost-based Classifier for Training Performance Using the Flow-based Feature Set for Gtalk Detection (DR versus FPR)

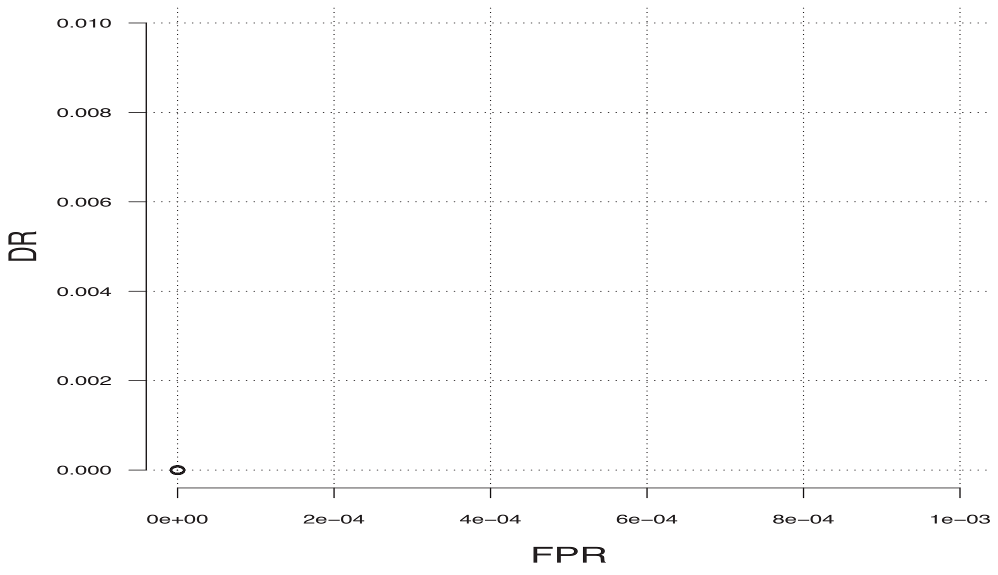


Figure 9.12: Scatter for the AdaBoost-based Classifier for Training Performance Using the Flow-based Feature Set for Primus Detection (DR versus FPR)

The results are summarized in terms of DR and FPR. Tables 9.1, 9.2 and 9.3 list the results for the three machine learning algorithms on the training and independent test traces. In this case, results show that C5.0 performs much better than the GP and AdaBoost algorithms in classifying multiple VoIP applications. For Skype C5.0 achieves ≈100% DR and ≈1% FPR on the Univ2007 Test partition, ≈80% DR and ≈4% FPR on the Univ2010 traces, ≈91% DR on the ITALY-TCPE2X traces, ≈83% DR on the ITALY-UDPE2E traces, ≈58% DR on the ITALY-UDPE2O traces and ≈86% on the ITALY-UDPSIG. Since the ITALY traces consist only of Skype traffic, the FP rate for Skype and DR for non-Skype was 'non-applicable'. For Gtalk, C5.0 achieved ≈96% DR and ≈0% FPR on the NIMSII traces and ≈91% DR and ≈0% FPR on the NIMSIII traces. For Primus, C5.0 achieved ≈94% DR and ≈0% FPR on the NIMSII traces and ≈92% DR and ≈0% FPR on the NIMSIII traces. The C5.0 classifier is the most consistent performer across all test and training conditions while being competitive with GP for Skype detection under the university traces and Gtalk detection under the NIMS traces. This shows not only that the model which the C5.0 classifier learned during training is robust (generalized) enough to be tested on real world network traces, but also verifies that accurate differentiation between multiple VoIP applications is possible without employing port numbers, IP addresses and payload information.

Table 9.1: Results for the C5.0 Classifier - Multi-Class – All Traces

	SKYPE		**Non-VoIP**		**GTALK**		**PRIMUS**	
Data Sets	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training	0.997	0.007	0.993	0.004	0.962	0.000	0.951	0.086
Univ2007	0.996	0.007	0.993	0.004	N/A	0.000	N/A	0.000
Univ2010	0.803	0.038	0.962	0.197	N/A	0.001	N/A	0.000
ITALY TCPE2X	0.913	N/A	N/A	0.086	N/A	0.001	N/A	0.000
ITALY UDPE2E	0.834	N/A	N/A	0.099	N/A	0.032	N/A	0.035
ITALY UDPE2O	0.576	N/A	N/A	0.057	N/A	0.029	N/A	0.338
ITALY UDPSIG	0.863	N/A	N/A	0.095	N/A	0.040	N/A	0.002
GTALK2009	N/A	0.002	N/A	0.035	0.963	N/A	N/A	0.000
PRIMUS2009	N/A	0.000	N/A	0.056	N/A	0.002	0.942	N/A
ZFONE2009	N/A	0.059	0.751	N/A	N/A	0.164	N/A	0.027
GTALK2010	N/A	0.012	N/A	0.074	0.914	N/A	N/A	0.000
PRIMUS2010	N/A	0.022	N/A	0.049	N/A	0.014	0.915	N/A
YAHOO2010	N/A	0.082	0.902	N/A	N/A	0.016	N/A	0.000
RADIO2010	N/A	0.003	0.986	N/A	N/A	0.012	N/A	0.000
TORRENT2010	N/A	0.040	0.921	N/A	N/A	0.039	N/A	0.000
TV2010	N/A	0.008	0.987	N/A	N/A	0.005	N/A	0.000
VPN2010	N/A	0.000	1.000	N/A	N/A	0.000	N/A	0.000
IPv6 2000	N/A	0.000	1.000	N/A	N/A	0.000	N/A	0.000
IPv6 2009	N/A	0.000	1.000	N/A	N/A	0.000	N/A	0.000

Table 9.2: Results for the GP Classifier - Multi-Class – All Traces

	SKYPE		**Non-VoIP**		**GTALK**		**PRIMUS**	
Data Sets	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training	0.941	0.078	0.977	0.051	0.963	0.029	0.954	0.008
Univ2007	0.950	0.098	0.858	0.048	N/A	0.029	N/A	0.016
Univ2010	0.816	0.046	0.930	0.085	N/A	0.122	N/A	0.001
ITALY TCPE2X	0.802	N/A	N/A	0.198	N/A	0.000	N/A	0.000
ITALY UDPE2E	0.759	N/A	N/A	0.086	N/A	0.030	N/A	0.125
ITALY UDPE2O	0.234	N/A	N/A	0.002	N/A	0.452	N/A	0.312
ITALY UDPSIG	0.623	N/A	N/A	0.230	N/A	0.132	N/A	0.015
GTALK2009	N/A	0.004	N/A	0.033	0.962	N/A	N/A	0.000
PRIMUS2009	N/A	0.001	N/A	0.016	N/A	0.001	0.983	N/A
ZFONE2009	N/A	0.043	0.669	N/A	N/A	0.288	N/A	0.000
GTALK2010	N/A	0.051	N/A	0.048	0.901	N/A	N/A	0.000
PRIMUS2010	N/A	0.074	N/A	0.040	N/A	0.002	0.884	N/A
YAHOO2010	N/A	0.000	0.910	N/A	N/A	0.074	N/A	0.017
RADIO2010	N/A	0.001	0.999	N/A	N/A	0.000	N/A	0.000
TORRENT2010	N/A	0.006	0.932	N/A	N/A	0.044	N/A	0.019
TV2010	N/A	0.001	0.982	N/A	N/A	0.008	N/A	0.009
VPN2010	N/A	0.053	0.947	N/A	N/A	0.000	N/A	0.000
IPv6 2000	N/A	0.000	1.000	N/A	N/A	0.000	N/A	0.000
IPv6 2009	N/A	0.000	1.000	N/A	N/A	0.000	N/A	0.000

Table 9.3: Results for the AdaBoost Classifier - Multi-Class – All Traces

	SKYPE		**Non-VoIP**		**GTALK**		**PRIMUS**	
Data Sets	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**	**DR**	**FPR**
Training	0.734	0.049	0.951	0.266	0.000	0.000	0.000	0.000
Univ2007	0.747	0.027	0.973	0.253	N/A	0.000	N/A	0.000
Univ2010	0.710	0.067	0.933	0.290	N/A	0.000	N/A	0.000
ITALY TCPE2X	0.000	N/A	N/A	1.000	N/A	0.000	N/A	0.000
ITALY UDPE2E	0.206	N/A	N/A	0.794	N/A	0.000	N/A	0.000
ITALY UDPE2O	0.025	N/A	N/A	0.975	N/A	0.000	N/A	0.000
ITALY UDPSIG	0.735	N/A	N/A	0.265	N/A	0.000	N/A	0.000
GTALK2009	N/A	0.005	N/A	0.995	0.000	N/A	N/A	0.000
PRIMUS2009	N/A	0.001	N/A	0.999	N/A	0.000	0.000	N/A
ZFONE2009	N/A	0.235	0.765	N/A	N/A	0.000	N/A	0.000
GTALK2010	N/A	0.000	N/A	1.000	0.000	N/A	N/A	0.000
PRIMUS2010	N/A	0.321	N/A	0.679	N/A	0.000	0.000	N/A
YAHOO2010	N/A	0.002	0.998	N/A	N/A	0.000	N/A	0.000
RADIO2010	N/A	0.001	0.999	N/A	N/A	0.000	N/A	0.000
TORRENT2010	N/A	0.058	0.942	N/A	N/A	0.000	N/A	0.000
TV2010	N/A	0.001	0.999	N/A	N/A	0.000	N/A	0.000
VPN2010	N/A	0.013	0.987	N/A	N/A	0.000	N/A	0.000
IPv6 2000	N/A	0.001	0.999	N/A	N/A	0.000	N/A	0.000
IPv6 2009	N/A	0.111	0.889	N/A	N/A	0.000	N/A	0.000

9.2 Analysis of the Classifier Solutions

In this section, the solutions generated by the classification-based systems are analyzed in terms of CPU training time, the number of features employed and the number of rules/solutions generated in order to understand how multiple P2P VoIP applications are classified. Again, the summary for AdaBoost is not simple and so it will be limited to the total number of attributes utilized, regardless of the class.

9.2.1 CPU Training Time

Figure 9.13 summarizes the CPU training time using an Intel Xeon 2.67 GHz Quad 16 Core chip with 48 GB of RAM. On average, C5.0 is faster than GP and AdaBoost in building its model, as shown in Figure 9.13. The CPU training time for C5.0 is not only fast but also particularly impressive in finding very good solutions (see the results in section 9.1). GP takes the longest time for training (on average 2 hours, 30 minutes) compared to seven minutes on average for C5.0 and eight minutes on average for AdaBoost. Indeed, C5.0 can find effective solutions with large training sizes ($\approx$1.9 million flows) in a few minutes.

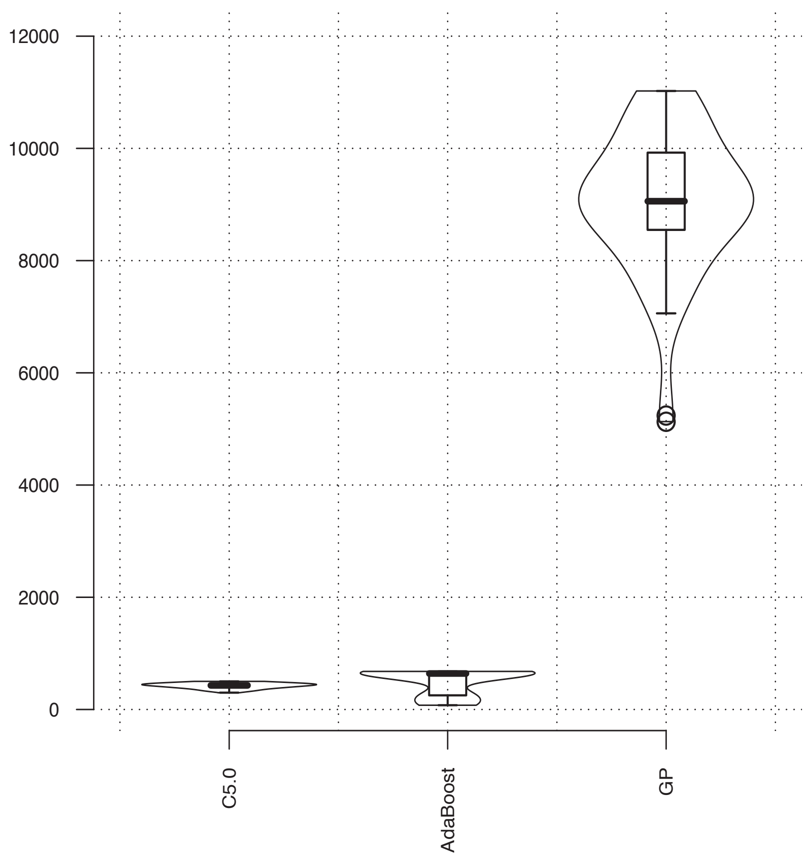


Figure 9.13: Training Time (in Seconds) for VoIP Multi-Class Classification Based on the Flow Feature Set

9.2.2 Complexity of Solutions

In this case, AdaBoost uses on average a lower total count of attributes for all VoIP applications relative to either GP or C5.0 (Figure 9.14). In contrast, C5.0 uses all of the 22 attributes (the largest set of attributes) for Skype, non-VoIP and Gtalk while using fourteen attributes for Primus. GP, on average, uses the second largest set of attributes: eighteen for Skype and Primus, twenty for Gtalk and twent-one for non-VoIP. In terms of solution complexity, GP generates fewer individuals on average (ten for non-VoIP, four for Skype, six for Gtalk and five for Primus) in detecting Skype flow traffic while AdaBoost finds the simpler solution which is, on average, competitive with GP (eight rules). C5.0 gives the most complex solution on average (317 for non-VoIP, 217 for Skype, 32 for Gtalk and nine for Primus), see Figure 9.15.

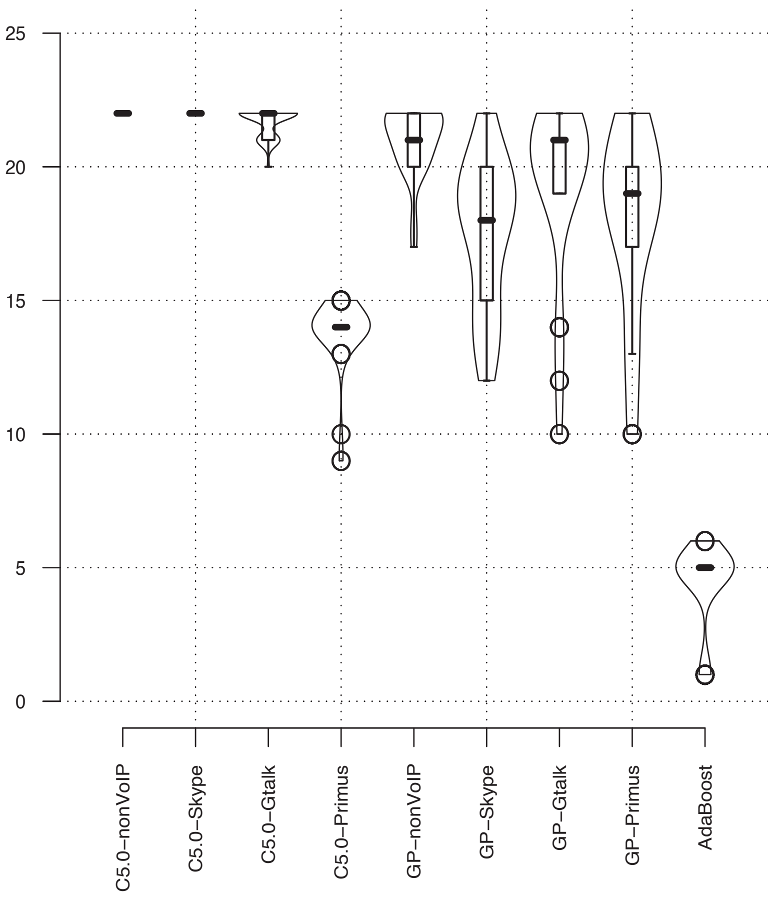


Figure 9.14: Number of Features Utilized for Each Classifier for VoIP Multi-Class Classification Based on the Feature Set

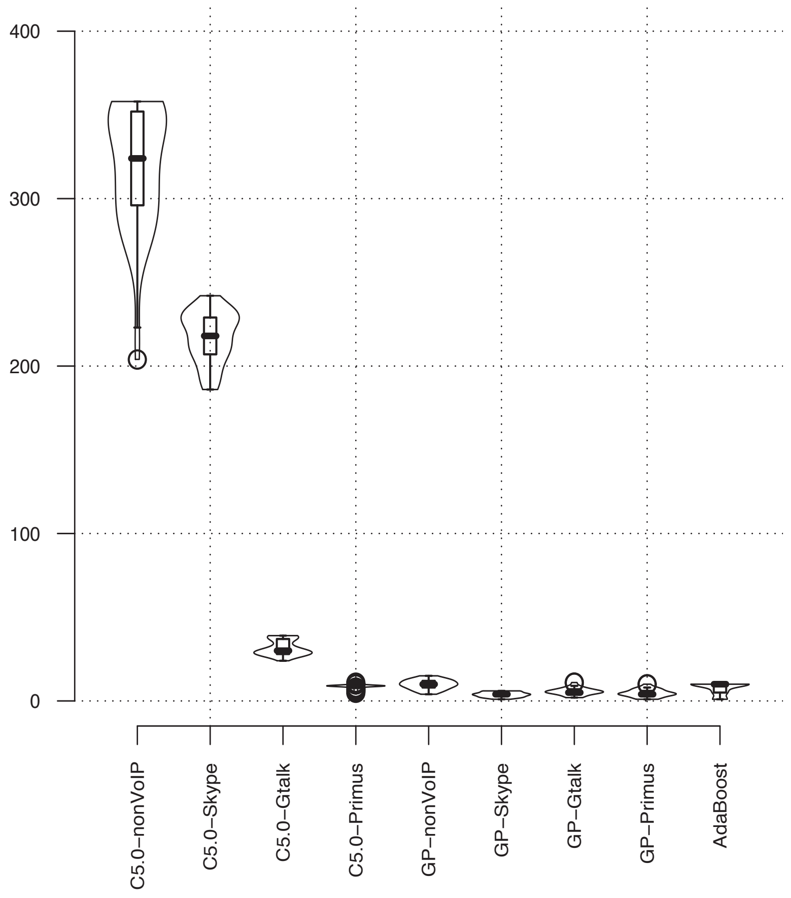


Figure 9.15: Number of Rules Utilized for Each Classifier for VoIP Multi-Class Classification Based on the Flow Feature Set

9.2.3 Analysis of the Best Solution from Each Classifier for Multi-Class VoIP Applications

For multi-class VoIP solutions, Tables 9.4, 9.5 and 9.6 summarize the number of features utilized by C5.0, GP and AdaBoost, respectively. Clearly, AdaBoost uses the lowest number of attributes relative to GP and C5.0 for classification. In contrast, C5.0 uses the largest set of attributes by class. However, the C5.0 solution has the best performance on these data sets compared to the GP and the AdaBoost solutions. In these experiments the C5.0-based classifier utilizes all of the features (22 flow features) to build signatures (rules) for the Skype and non-VoIP applications while using only fifteen features for Primus and twenty-one for Gtalk. The fifteen features focus on the packet size and time since VoIP applications depend strongly on the quality of service provided to their users. These features may provide more insight on the indicators of quality of service (high quality audio and video calls) to the VoIP application developers for improving their products.

Table 9.4: Flow Features Utilized by C5.0 for Multi-Classes

Skype	**non-VoIP**	**GTALK**	**PRIMUS**
min_fpktl	min_fpktl	min_fpktl	min_fpktl
mean_fpktl	mean_fpktl	mean_fpktl	mean_fpktl
max_fpktl	max_fpktl	max_fpktl	max_fpktl
std_fpktl	std_fpktl	std_fpktl	
min_bpktl	min_bpktl	min_bpktl	min_bpktl
mean_bpktl	mean_bpktl	mean_bpktl	mean_bpktl
max_bpktl	max_bpktl	max_bpktl	max_bpktl
std_bpktl	std_bpktl	std_bpktl	std_bpktl
min_fiat	min_fiat	min_fiat	min_fiat
mean_fiat	mean_fiat	mean_fiat	mean_fiat
max_fiat	max_fiat	max_fiat	
std_fiat	std_fiat	std_fiat	std_fiat
min_biat	min_biat	min_biat	min_biat
mean_biat	mean_biat	mean_biat	
max_biat	max_biat	max_biat	
std_biat	std_biat		
duration	duration	duration	duration
proto	proto	proto	
total_fpackets	total_fpackets	total_fpackets	total_fpackets
total_fvolume	total_fvolume	total_fvolume	total_fvolume
total_bpackets	total_bpackets	total_bpackets	
total_bvolume	total_bvolume	total_bvolume	total_bvolume

Table 9.5: Flow Features Utilized by GP for Multi-Classes

Skype	**non-VoIP**	**GTALK**	**PRIMUS**
min_fpktl	min_fpktl	min_fpktl	min_fpktl
mean_fpktl	mean_fpktl	mean_fpktl	mean_fpktl
max_fpktl	max_fpktl	max_fpktl	max_fpktl
std_fpktl	std_fpktl	std_fpktl	std_fpktl
min_bpktl	min_bpktl	min_bpktl	min_bpktl
mean_bpktl	mean_bpktl	mean_bpktl	mean_bpktl
max_bpktl		max_bpktl	
	std_bpktl	std_bpktl	std_bpktl
min_fiat	min_fiat	min_fiat	min_fiat
	mean_fiat	mean_fiat	mean_fiat
	max_fiat	max_fiat	max_fiat
std_fiat	std_fiat	std_fiat	std_fiat
	min_biat	min_biat	min_biat
mean_biat	mean_biat	mean_biat	mean_biat
max_biat	max_biat	max_biat	max_biat
std_biat	std_biat	std_biat	std_biat
duration	duration	duration	duration
proto	proto	proto	proto
total_fpackets	total_fpackets	total_fpackets	total_fpackets
total_fvolume	total_fvolume	total_fvolume	total_fvolume
total_bpackets	total_bpackets	total_bpackets	total_bpackets
total_bvolume	total_bvolume	total_bvolume	total_bvolume

Table 9.6: Flow Features Utilized by AdaBoost for Multi-Classes

Features
duration
proto
max_bpktl
std_fiat
total_fpackets

```
Decision Stump

Classifications

max_bpktl <= 62.5 : SKYPE
max_bpktl > 62.5 : notVoIP
max_bpktl is missing : notVoIP

Class distributions

max_bpktl <= 62.5
SKYPE   notVoIP  GTALK   PRIMUS
0.808242089758833  0.19036041951340704     0.0013974907277599332
     0.0
max_bpktl > 62.5
SKYPE   notVoIP  GTALK   PRIMUS
0.10127242113679873     0.8900705021258275 0.00831033236716461 5
     3.467443702091999E-4
max_bpktl is missing
SKYPE   notVoIP  GTALK   PRIMUS
0.28323735724913474     0.7099740904538909 0.006531055485709783
     2.574968112645434E-4


Weight: 1.89

Decision Stump

Classifications

proto <= 11.5 : notVoIP
proto > 11.5 : SKYPE
proto is missing : notVoIP

Class distributions

proto <= 11.5
SKYPE   notVoIP  GTALK   PRIMUS
0.005908121626949904    0.950762270807104  0.04302540997858238
     3.0419758736366874E-4
proto > 11.5
SKYPE   notVoIP  GTALK   PRIMUS
0.5734725916985552 0.40785890063096675     0.017403369134131673
     0.0012651385363464638
proto is missing
SKYPE   notVoIP  GTALK   PRIMUS
0.4067605857896047 0.567327145437835   0.02492938961222175 6
     9.82879160338412E-4


Weight: 0.77
```

Figure 9.16: An Example of an AdaBoost Solution for Multi-Class VoIP Application Classification Based on the Flow Feature Set

In terms of solution complexity, AdaBoost generates ten signatures. C5.0 employs 223 signatures for Skype classification, 307 signatures for non-VoIP classification, 28 signatures for Gtalk classification and eleven signatures for Primus classification. GP uses eighteen individuals for Skype classification, seven individuals for non-VoIP classification, eleven individuals for Gtalk classification and eight for Primus classification. Figures 9.16, 9.17 and 9.18 show parts of the C5.0, AdaBoost and GP solutions for classifying multiple VoIP applications based on the flow feature set.

```
Rule 530: (1103988/624685, lift 1.5)
      mean_fiat <= 35603
      ->  class SKYPE  [0.434]

Rule 531: (7241/1, lift 153.1)
      min_bpktl > 38
      min_bpktl <= 77
      std_bpktl <= 693
      min_fiat > 2
      duration > 2.099204e+08
      ->  class GTALK  [1.000]

Rule 532: (959, lift 153.0)
      min_fpktl <= 84
      min_bpktl <= 383
      min_fiat > 10
      min_biat <= 4
      total_bvolume > 19676
      ->  class GTALK  [0.999]

Rule 533: (336, lift 152.7)
      min_fpktl > 47
      max_fpktl <= 48
      min_bpktl > 39
      max_bpktl > 59
      std_bpktl <= 30
      total_bpackets > 8
      ->  class GTALK  [0.997]
```

Figure 9.17: An Example of a C5.0 Solution for Multi-Class VoIP Application Classification Based on the Flow Feature Set

```
R[0] <- R[0] I[3] exp
R[7] <- R[7] R[2] diff
R[5] <- R[5] I[19] cos
R[5] <- R[5] I[21] log
R[0] <- R[0] I[1] diff
R[0] <- R[0] R[6] div
R[3] <- R[3] I[19] diff
R[3] <- R[3] R[4] exp
R[4] <- R[4] I[3] diff
R[4] <- R[4] R[0] diff
R[0] <- R[0] I[8] diff
R[6] <- R[6] I[14] log
R[5] <- R[5] R[2] div
R[4] <- R[4] I[16] cos
R[1] <- R[1] I[4] cos
R[6] <- R[6] R[4] sum
R[4] <- R[4] R[5] log
R[3] <- R[3] R[3] sum
R[7] <- R[7] R[0] prod
R[6] <- R[6] R[2] sum
R[7] <- R[7] I[13] prod
R[2] <- R[2] R[6] log
R[0] <- R[0] R[5] log
R[4] <- R[4] R[4] sum
R[3] <- R[3] I[7] exp
R[4] <- R[4] I[11] sum
R[0] <- R[0] R[2] log
R[4] <- R[4] I[17] diff
R[2] <- R[2] R[2] cos
R[6] <- R[6] I[12] diff
R[6] <- R[6] R[3] cos
R[0] <- R[0] R[5] prod
R[4] <- R[4] R[6] sum
R[3] <- R[3] I[13] cos
R[3] <- R[3] I[16] div
R[3] <- R[3] R[3] exp
R[6] <- R[6] R[2] cos
R[1] <- R[1] R[6] diff
R[0] <- R[0] R[1] log
R[1] <- R[1] R[7] sum
R[5] <- R[5] I[3] cos
R[1] <- R[1] R[7] sum
R[0] <- R[0] I[20] prod
```

Figure 9.18: An Example of a GP Solution for Multi-Class VoIP Application Classification Based on the Flow Feature Set

9.3 Discussion

As seen in the above results, the C5.0-based classifier has the most consistent performance compared to the AdaBoost and the GP classifiers whether classifying an individual VoIP application such as Skype with high DR and low FPR or classifying more than one VoIP application (Gtalk and Primus). These results demonstrate that the model that the C5.0 classifier learned during training is robust (generalized) enough to be tested on real-world network traces. Furthermore, these results verify that the accurate differentiation among multiple VoIP applications is possible by using flow features based on packet size, time and direction while not employing port numbers, IP addresses and payload information. Last, but not least, these results demonstrate that it is possible to have a generic attribute set which can be employed to identify multiple VoIP applications such as Skype, Gtalk and Primus.

Chapter 10

Conclusion

The primary motivation behind the proposed thesis is the challenging problem of identifying network traffic, specifically encrypted traffic, according to the application type. Network traffic classification is viewed as an essential task of any network operations management group since it is used to manage bandwidth budgets and to ensure quality of service objectives for critical applications. This thesis employs a ML-based approach to overcome the insufficiency of traditional approaches (port number and payload signature approaches) for classifying network traffic without using IP addresses, TCP/UDP port numbers or payload information.

10.1 Research Objectives Realized

To be able to achieve the goal of robust classification of VoIP encrypted traffic eight steps (milestones) were pursued. These are summarized in the following.

10.1.1 Identifying a Suitable Method for Sampling a Training Data Set to Generate Robust Signatures

Given that a ML-based approach is employed in this thesis, one of the more important factors in ML algorithms is the use of a training data set which represents the network data well. Since the amount of traffic on a network link is massive (could be terabytes per day, 10^{12} bytes), sampling an informative training data set becomes a hard task. Therefore, in addressing this issue, three different sampling techniques were studied in Chapter 4. These subset sampling methods are: (i) uniform random N sampling, where N is either a fixed number of records (e.g. 30K, 60K, etc.) or N is a fixed percentage of records (e.g. 1%, 2%, etc.); (ii) stratified N sampling based on grouping, where N is either a fixed number of records (e.g. 30K, 60K, etc.) or N is a fixed percentage of records (e.g. 1%, 2%, etc.); (iii) continuous data streams of either a specific time period (e.g. 30 minutes, 60 minutes and 90 minutes of traffic) or N

sampling records (e.g. 30K, 60K, etc.). As a result of these three subset sampling techniques, thirty-three training data sets were sampled to explore these methods to understand which one would be more suitable for generating robust signatures for classifying VoIP traffic, specifically Skype. To this end, three ML algorithms (AdaBoost, C5.0 and GP) were trained on all the training data sets, which were sampled from Univ2007 network traffic traces and tested on validation data sampled from the Univ2010 traces. Results indicate that Uniform Random Sampling with 6% of the records is the most appropriate method for achieving this objective.

10.1.2 Exploring the Performance of Classifying Encrypted Traffic Using Only One Packet – Without any Temporal Information

Feature sets are important for the ML algorithms, enabling them to group the characteristics of the network traffic into labels. Twenty-nine features extracted from the packet header were employed in this case, as described in Chapter 3, section 3.1.1. The main objective of using such a feature set based only on the packet header information is to test the possibility of generating signatures which can classify VoIP Skype traffic on the fly with low overhead and computational cost. Results of these experiments demonstrate that identifying Skype traffic on the fly using information extracted only from a single packet is possible, however the signatures generated are not robust. In other words, they work best when the training data set and test data set are from the same network.

10.1.3 Exploring the Performance of Classifying Encrypted Traffic Using Only One Flow – With Temporal Information

As discussed earlier, a flow feature is a descriptive statistic which can be calculated from one or more packets for each flow. The flow feature set depends on the direction, packet size and inter-arrival time of packets which make up the flow. To this end, 22 flow features were computed and described in Chapter 3, section 3.1.2. Features such as IP addresses, source/destination port numbers and payload are excluded from the feature set to ensure that the results were not dependent on such biased features. The resulting statistical flow feature set adds a computational overhead but can be achieved in near real-time and represents a benchmark feature set for more complex features which can be added in the future. Results of these experiments show that the performance of the ML algorithms improved significantly when temporal information was employed compared with the packet header-based feature set. Using this method, it is possible to classify encrypted VoIP traffic robustly without using port numbers, IP addresses and payload information. In other words, the signatures which are generated by this approach are able to classify such traffic on networks different from the ones on which they are trained.

10.1.4 Exploring the Limits of Employing Machine Learning Algorithms in Order to Classify Encrypted Traffic Robustly

Since the objective is to generate robust signatures as defined in Chapter 3, section 3.3, three different ML algorithms (C5.0, AdaBoost and GP) were employed in this thesis for generating robust signatures for classifying VoIP traffic. The robustness of AdaBoost, C5.0 and GP were investigated by training them on parts of Univ2007 but evaluating their performances on data sets captured from different locations, different networks and different time periods in Chapter 5. The test data sets were the Univ2007 Test partition, Univ2010, NIMS II, NIMSIII, ITALY and IPv6, which were captured in 2007, 2010, 2009, 2010, 2006, and in both 2000 and 2009 for IPv6, respectively. The results in Chapter 5 demonstrate that the ability to generate robust signatures automatically which are able to identify Skype with high DR and low FPR on different network traces and without retraining is possible. As well, these results show that the signatures are able to identify Skype even though many different Skype versions were in use/circulation from 2006 to 2010, e.g. version 2.5 in 2006 [55] to version 5.0 in 2010 [107]. Furthermore, the quantification of the robustness of the signatures can be measure in terms of achieving a DR that is higher than or equal to 80% and a FPR that is lower than 6% on test traces.

10.1.5 Testing the Ability of the Robust Signatures Against Evasion Attacks

In Chapter 6 the robust signatures generated in Chapter 5 were evaluated against evasion attacks. The signatures were evaluated on unseen altered data (evasion data) by using morphing techniques (padding the payload). Even though, papers [74, 75] in the literature claimed that ML algorithms using statistical features can be evaded easily, the empirical results obtained in this research suggest that this is not the case. These results suggest that when an informative training data set is provided, ML algorithms employing statistical flow-based features can generate robust signatures even against evasion attacks. However, it should be noted here that such attacks do cause a performance drop in the classification, but cannot be successful all the time.

10.1.6 Analysis of the Selected Features

Results from Chapter 5 show that the GP and C5.0 classifiers performed better than AdaBoost on both the packet header feature set and flow-based feature sets. Since the proposed approach in this thesis is data-driven, all the features were presented to the ML algorithms so that they could select the most suitable ones from the given set. By analyzing the rules/models generated by C5.0 and GP in Chapter 7 it can be seen that C5.0 selects 19 features out of 29 and GP selects 26 features out of 29 for classifying Skype using packet header-based information. Analysis of the flow-based solutions reveals that the GP-based classifier selects 16 out of 22 flow features whereas the C5.0-based classifier employs all of the 22 flow features in its solution (rule set). Furthermore, the analysis of these selected features show that the ML algorithms attempt to understand Skype application behaviour by differentiating it at both ends of the communication (Chapter 7).

10.1.7 Exploring the Performance of Ensemble Learning for Enhancing the Performance of the Signatures

Each ML algorithm employed in this thesis uses a different method for building its solution. In chapter 8 a technique for combining the different solutions produced by the ML algorithms into one model was explored: an ensemble learning technique based on stack generalization was employed. Stack generalization is based on two layers where the outputs of the first layer (e.g. the solutions of the three ML algorithms) are fed to the second layer where the integration of different ML solutions occur. In this research, a NN was employed at the second layer to produce the ensemble learning model. Results show that an improvement of the generalization of the AdaBoost, C5.0 and GP solutions can be achieved when they are combined into one model using ensemble learning.

10.1.8 Exploring the Ability of the Proposed Approach to Classify more than One VoIP Application

Chapter 9 explores the ability of the three ML algorithms to classify robustly and accurately (high DR and low FPR) more than one P2P VoIP application. Three VoIP applications, namely Skype, Gtalk and Primus softphone, were employed to evaluate the ability of the proposed approach for this problem. A testbed was set up in the Dalhousie NIMS lab to generate traffic for the Gtalk and Primus applications in 2009 and 2010 since there were no public traces available for these two applications. Results of these experiments show that the C5.0-based classifier can generate robust signatures not only to classify a single VoIP application, namely Skype, but also to classify other VoIP applications such as Gtalk and Primus.

10.2 Key Contributions

In this thesis, the main goal is to investigate the robustness of the models/signatures generated automatically by a ML-based approach – specifically C5.0, GP and AdaBoost – for distinguishing encrypted VoIP traffic (namely Skype) from non-encrypted traffic in a given traffic trace, robustness meaning that the classifier is trained on a data set from one network traffic trace but tested on:

1. unseen data from different locations/networks;
2. unseen data from different time periods;
3. unseen data which is altered by padding/morphing (evasion attacks)

To explore this, three supervised ML algorithms, C5.0 AdaBoost and GP, were employed. It should be noted here that these learning algorithms are not opaque but rather their solutions can be analyzed to provide signatures (rules) which can be understood easily by human experts. Moreover, these learning algorithms are well known for selecting the most appropriate features from a given set for a given task. Furthermore, the classification based approach was employed using only packet header attributes or flow attributes.

In this research, the C5.0-based classification approach performed the best on the given data sets. In the best case scenario the C5.0-based classifier achieved 99.6% DR with 0.7%FPR (when trained on one network but tested on another) in detecting Skype traffic. These results show that the classification based system trained on data from one network can be employed to run on a different network without new training. Additionally, the results of evasion attack traces based on morphing techniques in Chapter 6 showed that C5.0 provided over 80% DR on average for Skype VoIP traffic classification on maliciously altered Skype traffic. Moreover, the ensemble learning technique (Chapter 8) can be applied to increase the DR by 1% while reducing the FPR by 1% as well.

The C5.0-based classifier achieved ≈100% DR and ≈1% FPR on the Univ2007 Test partition, ≈80% DR and ≈6% FPR on the Univ2010 traces, ≈99% DR on the ITALY-TCPE2X traces, ≈89% DR on the ITALY-UDPE2E traces, ≈53% DR on the ITALY-UDPE2O traces and ≈92% on the ITALY-UDPSIG. It should be noted here that the ITALY traces only have Skype traffic. Furthermore, the C5.0-based classifier identified IPv6 and VPN flows with high DR and low FPR. C5.0 achieved ≈99% DR and 0% FPR on the NIMSIII VPN2010 traces, ≈100% DR and 0% FPR on the IPv6 2000 traces and ≈89% DR and 0% FPR on the IPv6 2009 traces. With regard to evasion attacks, the signatures generated by the C5.0-based classifier from a statistical feature set and a well chosen training data set were proven to be robust and not easy to evade. The C5.0-based classifier achieved ≈91% DR and ≈5% FPR on the Original Skype flows and ≈85% DR and ≈5% FPR on the Altered-Skype flows.

In summary, this thesis has proposed and developed a classification system which is able to generate robust signatures automatically for classifying VoIP traffic, specifically Skype traffic, by addressing the aforementioned research objectives highlighted in section 10.1. As a result of this research, three main contributions have been made which are detailed below.

1. The empirical exploration of a subset sampling method for the challenging problem of sampling a training data set for ML algorithms for classifying encrypted VoIP traffic.

2. The empirical exploration of the difference between feature sets with and without temporal information. Then, based on these features, the evaluation of the

robustness of automatically generated signatures on traffic traces from different time periods, locations, networks infrastructures as well as against evasion attacks.

3. The analysis of the robust signatures and features employed to gain an insight into the problem of encrypted VoIP traffic classification.

10.3 Future Research Directions

Given the results obtained in this thesis, there are many future research directions which can be pursued. These are listed below.

1. Obtaining network traces for research purposes is a lengthy task and requires a lot of time and effort. Furthermore, finding public network traces with accessible payloads is even harder due to privacy issues. However, network traffic can be generated in a lab environment. Therefore, future work might follow similar lines to obtain more traces for performing more tests on different and/or even larger data sets in order to continue to evaluate the robustness of the proposed approach.

2. An interesting direction might be to combine the packet header feature sets with the flow feature sets. Such a combination can allow a coarse-grained parallelism which can increase the accuracy of detecting encrypted traffic.

3. Network traces are huge in size and a subset sampling method is an essential step in finding an informative training data which can be input to ML algorithms in order to generate robust signatures. Therefore, another research direction which might be followed would be to explore larger percentage sizes for randomly sampling training data sets with uniform probability in order to define a minimum and maximum range which can be used for subset sampling.

4. Employing a pruning algorithm for the selection of the optimal rules for ensemble learning and training the ensemble learning model on a large training data set (e.g. the training data set employed in Chapter 5) would be another interesting future research direction.

5. VoIP applications such as Skype are appearing on new wireless gadget devices. It would be interesting to evaluate the proposed approach on such devices when they are running Skype (or other VoIP applications) and connected using WiFi as well as testing on other VoIP applications installed on these devices.

6. Use of different operators for the GP opcodes rather than the standard arithmetic operators used in this research (e.g. conditional operators $>$, $>=$, $<$, $<=$).

7. Another area of interest might be exploring the possibilities for integrating the proposed approach with other approaches such as those employing host-based behaviour or approaches based on unsupervised learning (e.g. clustering algorithms).

8. Testing the performance of the generated signatures on other IPv6, VPN and Altered-VoIP traces.

9. Finally, there is always the potential for additional evaluation of the proposed approach under other encrypted applications.

Bibliography

[1] P. Biondi and F. Desclaux. Silver needle in the skype. *Black Hat 2006 Multimedia - Presentation, Audio and Video Archives*, March 2006.

[2] Skype. http://www.skype.com/useskype/.

[3] Google. Google talk (Gtalk), last accessed October, 2009. http://www.google.com/talk/.

[4] Microsoft. MSN messenger, last accessed October, 2009. http://webmessenger.msn.com/.

[5] Primus Telecommunications Canada Inc. Primus Softphone Client, last accessed October, 2009. http://www.primus.ca/en/residential/talkbroadband/talkBroadband-softphone.htm.

[6] Yahoo! Yahoo messenger, last accessed October, 2009. http://messenger.yahoo.com/.

[7] S. A. Baset and H. G. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–11, April 2006.

[8] IANA. Internet Assigned Numbers Authority, last accessed October, 2009. http://www.iana.org/assignments/port-number.

[9] Andrew W. Moore and Konstantina Papagiannaki. Toward the accurate identification of network applications. In *Passive and Active Network Measurement: Proceedings of the Passive &Active Measurement Workshop*, pages 41–54, 2005.

[10] A. Madhukar and C. Williamson. A longitudinal study of p2p traffic classification. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. MASCOTS 2006. 14th IEEE International Symposium on*, pages 179 – 188, Sept. 2006.

[11] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 512–521, New York, NY, USA, 2004. ACM.

[12] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. In *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286, New York, NY, USA, 2006. ACM Press.

[13] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. BLINC: multilevel traffic classification in the dark. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 229–240, New York, NY, USA, 2005. ACM Press.

[14] Riyad Alshammari and A. Nur Zincir-Heywood. Can encrypted traffic be identified without port numbers, ip addresses and payload inspection? *Computer Networks*, 55(6):1326 – 1350, 2011.

[15] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic classification on the fly. *SIGCOMM Comput. Commun. Rev.*, 36(2):23–26, 2006.

[16] C. Bacquet, A.N. Zincir-Heywood, and M.I. Heywood. Genetic optimization and hierarchical clustering applied to encrypted traffic identification. In *Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on*, pages 194 –201, april 2011.

[17] Andrew W. Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 50–60, New York, NY, USA, 2005. ACM Press.

[18] R. Alshammari and A. N. Zincir-Heywood. A flow based approach for ssh traffic detection. *Proceedings of the IEEE International Conference on System, Man and Cybernetics - SMC'2007*, 2007.

[19] Riyad Alshammari and A. Nur Zincir-Heywood. Investigating two different approaches for encrypted traffic classification. In *PST '08: Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust*, pages 156–166, Washington, DC, USA, 2008. IEEE Computer Society.

[20] R. Alshammari and N. Zincir-Heywood. Generalization of signatures for ssh encrypted traffic identification. In *Computational Intelligence in Cyber Security, 2009. CICS '09. IEEE Symposium on*, pages 167–174, 30 2009-April 2 2009.

[21] J Early, C Brodley, and C Rosenberg. Behavioral authentication of server flows. In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 46–55, 2003.

[22] Patrick Haffner, Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. ACAS: automated construction of application signatures. In *MineNet '05: Proceeding of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 197–202, New York, NY, USA, 2005. ACM Press.

[23] Annie De Montigny-Leboeuf. Flow Attributes For Use In Traffic Characterization. *CRC Technical Note No. CRC-TN-2005-003*, 2005.

[24] Charles Wright, Fabian Monrose, and Gerald M. Masson. HMM profiles for network traffic classification. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 9–15, New York, NY, USA, 2004. ACM Press.

[25] Nigel Williams, Sebastian Zander, and Grenville Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *SIGCOMM Comput. Commun. Rev.*, 36(5):5–16, 2006.

[26] N.N. Pise and P. Kulkarni. A survey of semi-supervised learning methods. In *Computational Intelligence and Security, 2008. CIS '08. International Conference on*, volume 2, pages 30 –34, Dec. 2008.

[27] Riyad Alshammari. Automatically classifying encrypted network traffic: A case study of ssh. *Masters Thesis, Faculty of Computer Science, Dalhousie University*, May 2008.

[28] J.R. Quinlan. see5-comparison, last accessed February, 2011. http://www.rulequest.com/see5-comparison.html.

[29] Gary M. Weiss and Foster J. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *J. Artif. Intell. Res. (JAIR)*, 19:315–354, 2003.

[30] Christian Gagné, Michèle Sebag, Marc Schoenauer, and Marco Tomassini. Ensemble learning for free with evolutionary algorithms? In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1782–1789, New York, NY, USA, 2007. ACM.

[31] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, and D. Rossi. Tracking down skype traffic. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 261–265, April 2008.

[32] Jabber Software Foundation. Extensible Messaging and Presence Protocol (XMPP), last accessed October, 2009. http://www.ietf.org/rfc/rfc3920.txt.

[33] Riyad Alshammari and A. Nur Zincir-Heywood. Insight into the gtalk protocol. *Technical Reports*, CS-2010-02, March 2010. Available from: `http://www.cs.dal.ca/research/techreports/cs-2010-02`.

[34] R.B. Jennings, E.M. Nahum, D.P. Olshefski, D. Saha, Zon-Yin Shae, and C. Waters. A study of internet instant messaging and chat protocols. *Network, IEEE*, 20(4):16 –21, July-Aug. 2006.

[35] L.C.A. de Menezes Filho, M.L. da Costa, R.L. Belem, and E.J.M. Arruda Filho. Performance and quality of service on free softwares for voip. In *Optical Internet and Next Generation Network, 2006. COIN-NGNCON 2006. The Joint International Conference on*, pages 49 –53, July 2006.

[36] J. Rosenberg and H. Schulzrinne. SIP: Session Initiation Protocol, June 2002. http://www.ietf.org/rfc/rfc3263.txt.

[37] R. Frederick H. Schulzrinne, S. Casner and V. Jacobsos. RTP: A Transport Protocol for Real-Time Applications, July 2003. http://www.ietf.org/rfc/rfc3550.txt.

[38] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN: Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), March 2003. http://www.ietf.org/rfc/rfc3489.txt.

[39] S. Jordan. Four questions that determine whether traffic management is reasonable. In *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, pages 137 –140, June 2009.

[40] T.T.T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys Tutorials, IEEE*, 10(4):56 –76, fourth 2008.

[41] A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, and D. Sadok. A survey on internet traffic identification. *Communications Surveys Tutorials, IEEE*, 11(3):37 –52, quarter 2009.

[42] eBay. Skype Reaches 10 Million Concurrent Users, last accessed May, 2010. http://seekingalpha.com/article/50328-ebay-watch-59-earnings-growth-skype-reaches-10-million-concurrent-users.

[43] Saikat Guha, Neil Daswani, and Ravi Jain. An experimental study of the skype peer-to-peer voip system. In *IPTPS'06: The 5th International Workshop on Peer-to-Peer Systems*. Microsoft Research, Feb. 2006. Available from: `http://saikat.guha.cc/pub/iptps06-skype.pdf`.

[44] D. Rossi, M. Mellia, and M. Meo. Following skype signaling footsteps. In *Telecommunication Networking Workshop on QoS in Multiservice IP Networks, 2008. IT-NEWS 2008. 4th International*, pages 248 –253, Feb. 2008.

[45] Yanfeng Yu, Dadi Liu, Jian Li, and Changxiang Shen. Traffic identification and overlay measurement of skype. In *Computational Intelligence and Security, 2006 International Conference on*, volume 2, pages 1043 –1048, Nov. 2006.

[46] D. K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley. Characterizing and detecting relayed traffic: A case study using skype. *in INFOCOM 06: Proceedings of the 25th IEEE International Conference on Computer Communications*, Apr. 2006.

[47] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. Skype video responsiveness to bandwidth variations. In *NOSSDAV '08: Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 81–86, New York, NY, USA, 2008. ACM.

[48] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, and Paolo Tofanelli. Revealing skype traffic: when randomness plays with you. *SIGCOMM Comput. Commun. Rev.*, 37(4):37–48, 2007.

[49] E.P. Freire, A. Ziviani, and R.M. Salles. Detecting skype flows in web traffic. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 89 –96, April 2008.

[50] Alice Este, Francesco Gringoli, and Luca Salgarelli. Support vector machines for tcp traffic classification. *Computer Networks*, 53(14):2476 – 2490, 2009.

[51] WAND. Network Research Group, WITS: Auckland IV, last accessed October, 2011. http://www.wand.net.nz/wits/auck/4/.

[52] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, and Carey Williamson. Identifying and discriminating between web and peer-to-peer traffic in the network core. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 883–892, New York, NY, USA, 2007. ACM.

[53] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, Ira Cohen, and Carey Williamson. Offline/realtime traffic classification using semi-supervised learning. *Perform. Eval.*, 64:1194–1213, October 2007.

[54] Marios Iliofotou, Hyun chul Kim, Michalis Faloutsos, Michael Mitzenmacher, Prashanth Pappu, and George Varghese. Graption: A graph-based p2p traffic classification framework for the internet backbone. *Computer Networks*, 55(8):1909 – 1920, 2011. Available from: `http://www.sciencedirect.com/science/article/pii/S1389128611000430`.

[55] S. Ehlert, S. Petgang, T. Magedanz, and D. Sisalem. Analysis and signature of skype voip session traffic. *in CIIT 2006: 4th IASTED International Conference on Communications, Internet, and Information Technology*, page 8389, Nov./Dec. 2006.

[56] L. Bernaille and R. Teixeira. Early recognition of encrypted applications. *Passive and Active Measurement Conference (PAM), Louvain-la-neuve, Belgium*, April 2007.

[57] Wei Li and A.W. Moore. A machine learning approach for efficient traffic classification. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS '07. 15th International Symposium on*, pages 310 –317, October 2007.

[58] Wei Li, Marco Canini, Andrew W. Moore, and Raffaele Bolla. Efficient application identification and the temporal and spatial stability of classification schema. *Comput. Netw.*, 53(6):790–809, 2009.

[59] Y. Zhang and V. Paxson. Detecting back doors. In *Proceedings of the 9th USENIX Security Symposium*, pages 157–170. ACM Press, 2000.

[60] Jae Yoon Chung, Byungchul Park, Y.J. Won, J. Strassner, and J.W. Hong. An effective similarity metric for application traffic classification. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 286 –292, April 2010.

[61] A. Finamore, M. Mellia, M. Meo, and D. Rossi. Kiss: Stochastic packet inspection classifier for udp traffic. *Networking, IEEE/ACM Transactions on*, 18(5):1505 –1515, October 2010.

[62] Wireshark, last accessed September, 2008. http://www.wireshark.org/.

[63] Libpcap, last accessed September, 2008. http://www.tcpdump.org/.

[64] CAIDA. The Cooperative Association for Internet Data Analysis, last accessed September, 2011. http://www.caida.org/.

[65] David Moore, Ken Keys, Ryan Koga, Edouard Lagache, and K. C. Claffy. The coralreef software suite as a tool for system and network administrators. In *Proceedings of the 15th USENIX conference on System administration*, pages 133–144, Berkeley, CA, USA, 2001. USENIX Association.

[66] l7 filter, last accessed March, 2008. http://l7-filter.sourceforge.net/.

[67] IPOQUE. Bandwidth Management with Deep Packet Inspection, last accessed April, 2011. http://www.ipoque.com/.

[68] OpenDPI. the Open Source version of ipoque's DPI engine, last accessed April, 2011. http://www.opendpi.org/.

[69] Tstat. TCP STatistic and Analysis Tool, last accessed September, 2011. http://tstat.tlc.polito.it.

[70] Charles V. Wright, Fabian Monrose, and Gerald M. Masson. On inferring application protocol behaviors in encrypted network traffic. *J. Mach. Learn. Res.*, 7:2745–2769, 2006.

[71] Francesco Palmieri and Ugo Fiore. A nonlinear, recurrence-based approach to traffic classification. *Comput. Netw.*, 53(6):761–773, 2009.

[72] Ram Keralapura, Antonio Nucci, and Chen-Nee Chuah. A novel self-learning architecture for p2p traffic classification in high speed networks. *Computer Networks*, 54(7):1055 – 1068, 2010.

[73] Junghun Park, Hsiao-Rong Tyan, and C.-C.J. Kuo. Ga-based internet traffic classification technique for qos provisioning. In *Intelligent Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP '06. International Conference on*, pages 251 –254, Dec. 2006.

[74] Yan Hu, Dah-Ming Chiu, and John C. S. Lui. Profiling and identification of p2p traffic. *Comput. Netw.*, 53(6):849–863, 2009.

[75] Charles V. Wright, Scott E. Coull, and Fabian Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *Proceedings of the Network and Distributed Security Symposium - NDSS '09*, February 2009.

[76] Charles V. Wright, Lucas Ballard, Fabian Monrose, and Gerald M. Masson. Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob? In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 4:1–4:12, Berkeley, CA, USA, 2007. USENIX Association.

[77] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 255–263, New York, NY, USA, 2006. ACM.

[78] IETF. http://www3.ietf.org/proceedings/97apr/97apr-final/xrtftr70.htm.

[79] NetMate. http://www.ip-measurement.org/tools/netmate/.

[80] J.R. Quinlan. see5-info, last accessed July, 2010. http://www.rulequest.com/see5-info.html.

[81] E. Alpaydin. *Introduction to Machine Learning.* MIT Press, 2004.

[82] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[83] P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with Symbiotic Bid-based Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 363–370, 2008.

[84] John A. Doucette, Andrew R. Mcintyre, Peter Lichodzijewski, and Malcolm I. Heywood. Symbiotic coevolutionary genetic programming: a benchmarking study under large attribute spaces. *Genetic Programming and Evolvable Machines*, 13(1):71–101, March 2012.

[85] E.D. de Jong. A monotonic archive for pareto-coevolution. *Evolutionary Computation*, 15(1):61–93, 2007.

[86] J. Doucette and M.I. Heywood. GP Classification under Imbalanced Data Sets: Active Sub-sampling and AUC Approximation. In *European Conference on Genetic Programming*, volume 4971 of *Lecture Notes in Computer Science*, pages 266–277, 2008.

[87] Christopher D. Rosin and Richard K. Belew. New methods for competitive coevolution. *Evol. Comput.*, 5(1):1–29, 1997.

[88] The University of Waikato. WEKA Software. http://www.cs.waikato.ac.nz/ml/weka/.

[89] PacketShaper, last accessed March, 2008. http://www.packeteer.com/products/p-acketshaper/.

[90] Skype Traces. Telecommunication Networks Group - Politecnico di Torino, last accessed August, 2009. http://tstat.tlc.polito.it/traces-skype.shtml.

[91] Riyad Alshammari. Downloading the NIMS Data Sets, last accessed September, 201 1. http://web.cs.dal.ca/ riyad/Site/Download.html.

[92] Net Peeker. NetPeeker, last accessed October, 2009. http://www.net-peeker.com.

[93] Signalogic. Speech Codec Wav Samples, last accessed October, 2009. http://www.signalogic.com/index.pl?page=codec_samples.

[94] Phil Zimmermann. The Zfone Project, last accessed October, 2009. http://zfoneproject.com/.

[95] P. Zimmermann, Ed. A. Johnston, and J. Callas. ZRTP: Media Path Key Agreement for Secure RTP, January 2010. http://tools.ietf.org/html/draft-zimmermann-avt-zrtp-17.

[96] ETSI. Digital cellular telecommunications system (Phase 2+), General Packet Radio Service (GPRS), Overall description of the GPRS radio interface, Stage 2 (GSM 03.64, Version 7.0.0, Release 1999).

[97] BirdsSoft. VPN-X, last accessed March, 2011. http://birdssoft.com/.

[98] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol, November 1998. http://www.ietf.org/rfc/rfc2401.txt.

[99] MAWI. MAWI Working Group Traffic Archive. http://tracer.csl.sony.co.jp/mawi/.

[100] R. Fink and R. Hinden. 6bone (IPv6 Testing Address Allocation), March 2004. http://tools.ietf.org/html/rfc3701.

[101] MAWI. IPv6 Traffic Trace Information, MAWI Working Group Traffic Archive, last accessed March, 2011. http://mawi.wide.ad.jp/mawi/samplepoint-D/2009/200901011800.html.

[102] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464 –1480, Sep. 1990.

[103] A. Ultsch. Data mining and knowledge discovery with emergent self-organizing feature maps for multivariate time series. In *Kohonen Maps*, pages 33–46. Elsevier, 1999.

[104] T. Kohonen. self-organizing maps. *3rd ed. Springer Series in Information Sciences*, 30:501, 2001.

[105] E. Alhoniemi J. Vesanto, J. Himberg and J. Parhankangas. Som toolbox for matlab. *tech. rep., Helsinki University of Technology*, 2000.

[106] Jussi Hynninen Teuvo Kohonen and Jari Kangas. Som pak: The self-organizing map program package. *tech. rep., Helsinki University of Technology*, 1996.

[107] Skype. Skype Garage, last accessed September, 2011. http://blogs.skype.com/garage/windows/.

[108] J.-M. Valin and C. Montgomery. Improved noise weighting in CELP coding of speech - applying the Vorbis psychoacoustic model to Speex. *In Audio Engineering Society Convention*, May 2006. http://www.speex.org.

[109] NCH software. Switch Audio Converter for Mac, last accessed May, 2011. http://www.nch.com.au/switch/index.html.

[110] Tcpdump, last accessed September, 2008. http://www.tcpdump.org/.

[111] Aaron Turner. tcprewrite – Rewrite the packets in a pcap file, last accessed September, 2008. http://tcpreplay.synfin.net/tcprewrite.html.

[112] R. Curry. Towards efficient training on large datasets for genetic programming. *Masters Thesis, Faculty of Computer Science, Dalhousie University*, 2004. http://www.cs.dal.ca/~mheywood/Thesis/RCurry.pdf.

[113] Markus F. Brameier and Wolfgang Banzhaf. *Linear Genetic Programming (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[114] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24:123–140, August 1996.

[115] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[116] Yong Liu, Xin Yao, and Tetsuya Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4:380–387, 2000.

[117] Xin Yao and Yong Liu. Making use of population information in evolutionary artificial neural networks. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 28(3):417 –425, Jun. 1998.

[118] Martin Fodslette Møller. Original contribution: A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.*, 6:525–533, April 1993.

[119] MathWorks. MATLAB – The Language of Technical Computing, last accessed March, 2008. http://www.mathworks.com/products/matlab/index.html.

Appendix A

Results of Subset Sampling Methods

A.1 Uniform Random *N* Sampling

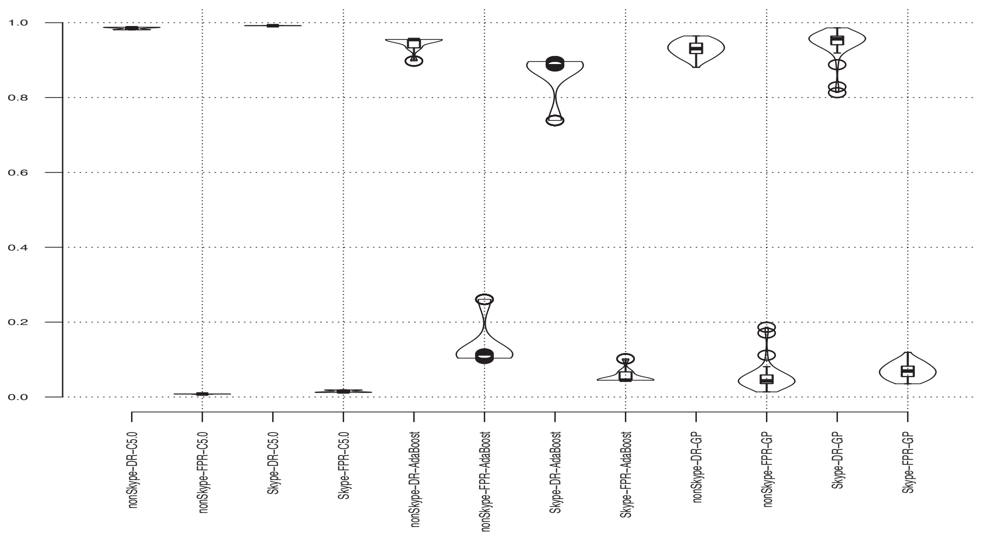


Figure A.1: Results of Three Machine Learning Algorithms Using the Uniform Random *N* Sampling 30K Method on Training Data

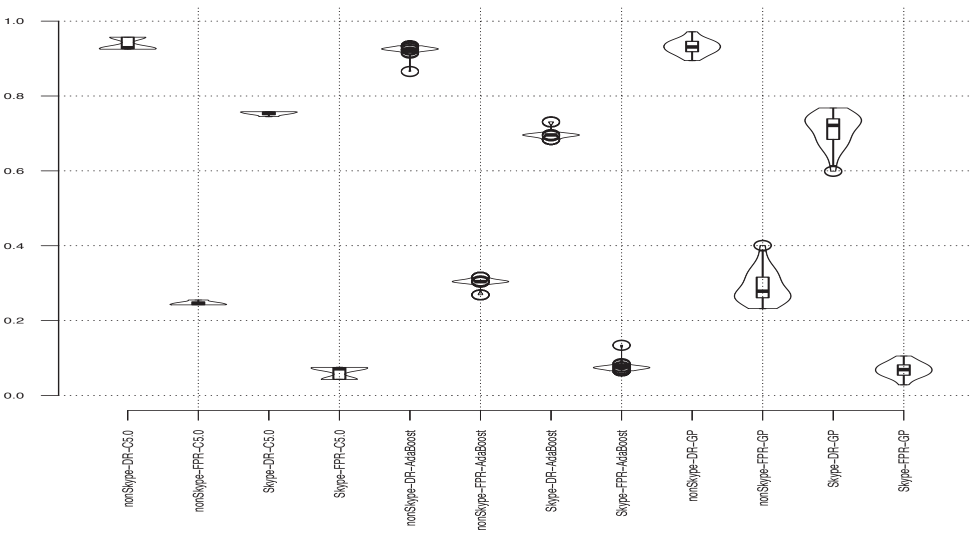


Figure A.2: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 30K Method on Validation Data

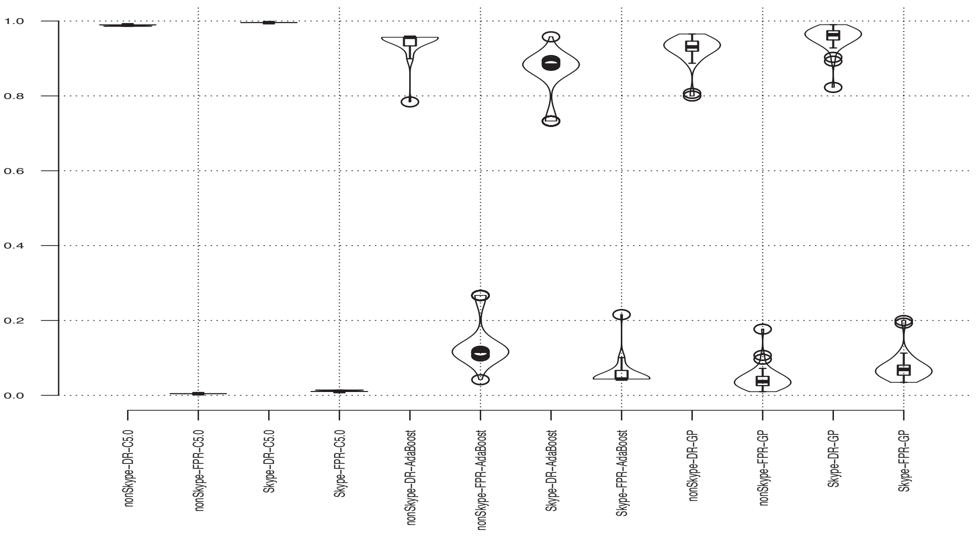


Figure A.3: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 60K Method on Training Data

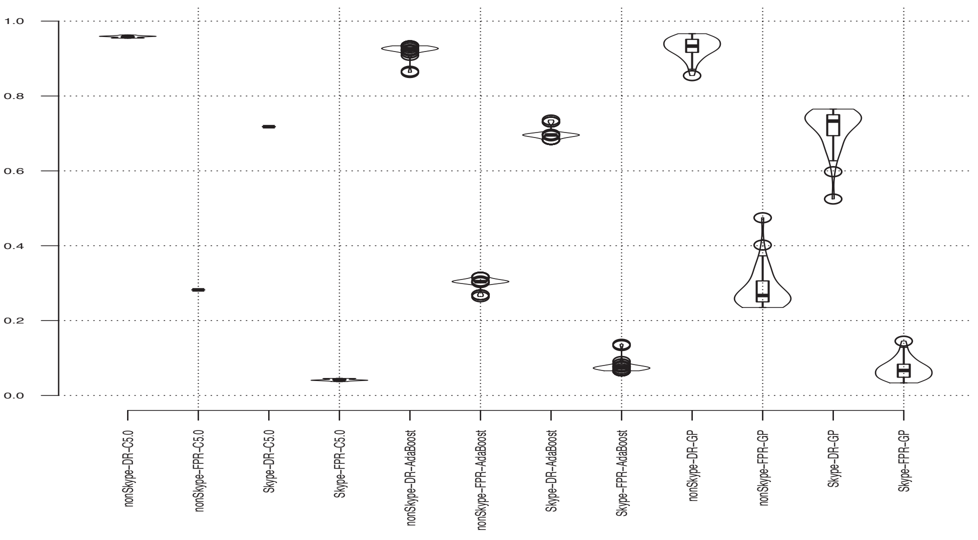


Figure A.4: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 60K Method on Validation Data

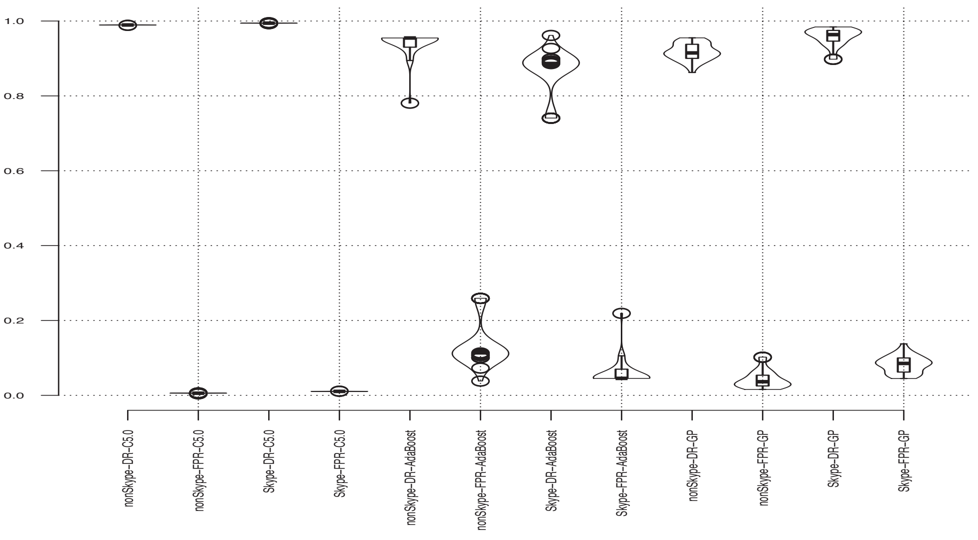


Figure A.5: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 100K Method on Training Data

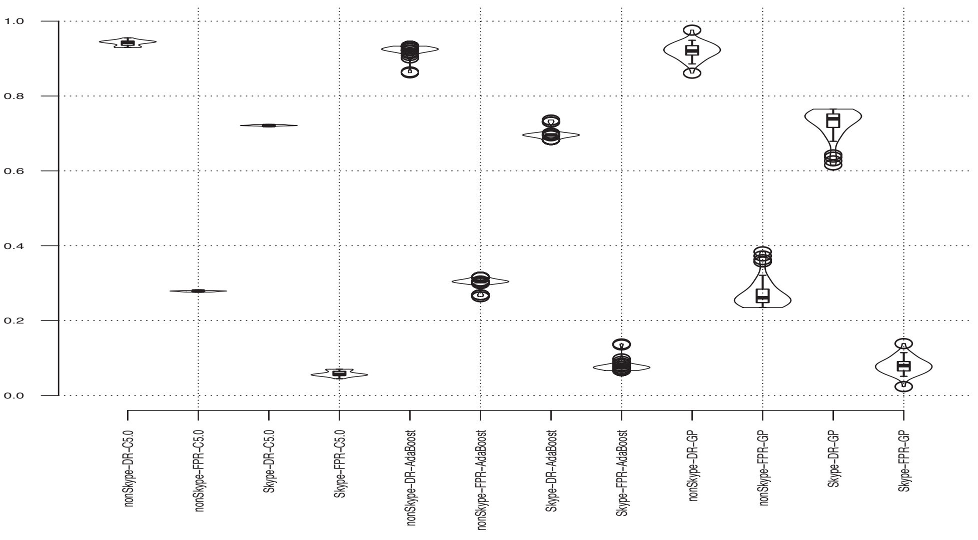


Figure A.6: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 100K Method on Validation Data

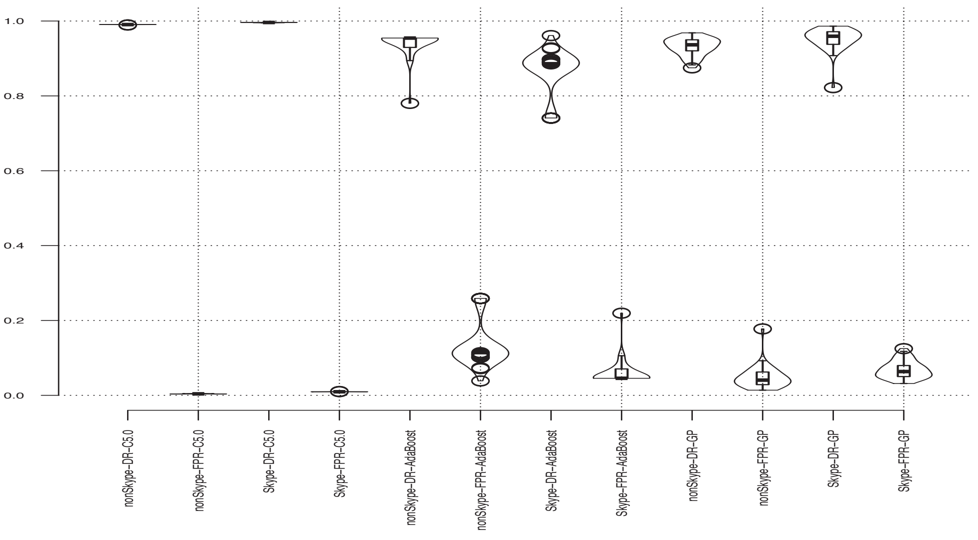


Figure A.7: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 200K Method on Training Data

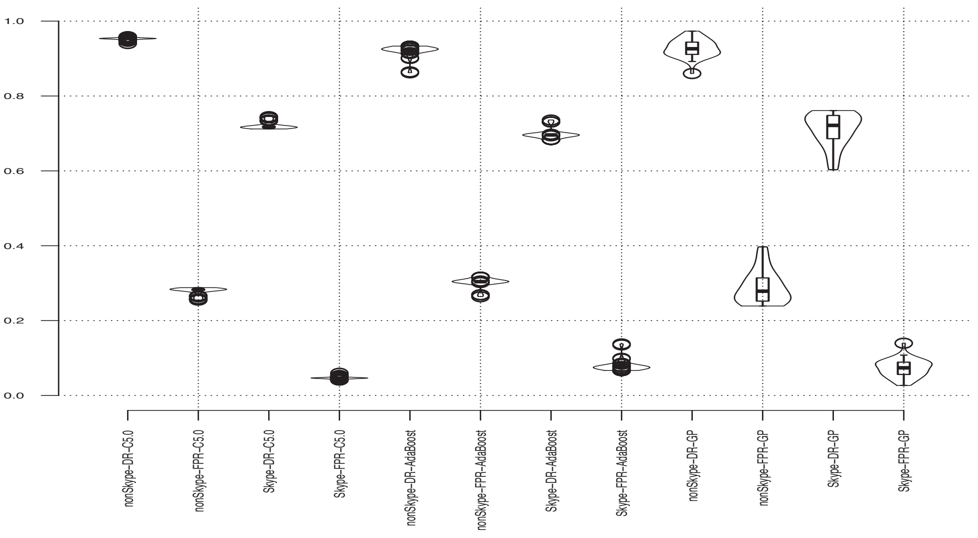


Figure A.8: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 200K Method on Validation Data

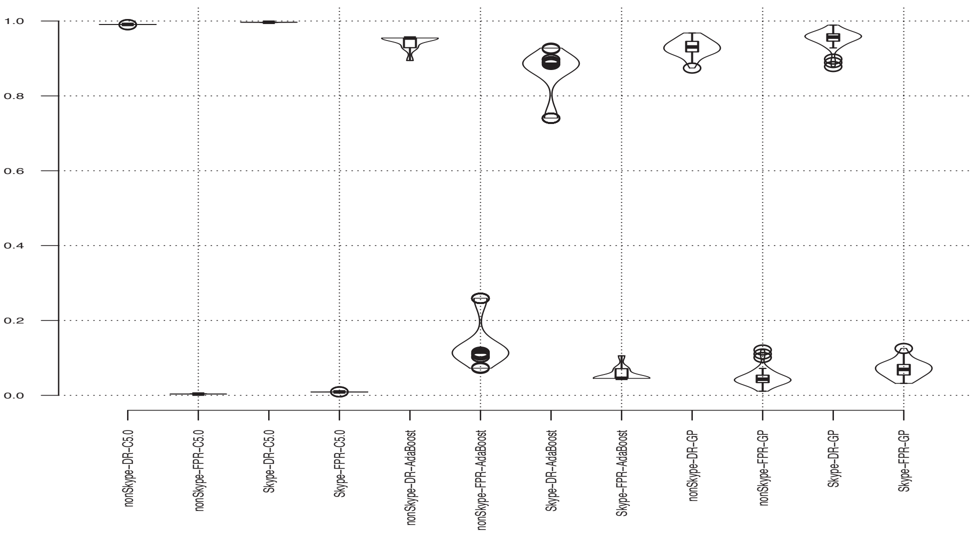


Figure A.9: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 400K Method on Training Data

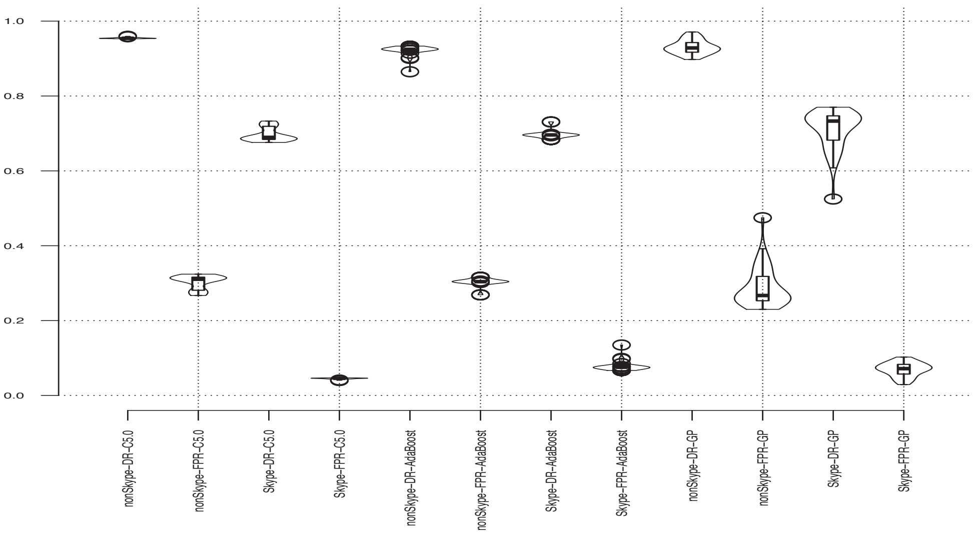


Figure A.10: Results of Three Machine Learning Algorithms Using the Uniform Random *N* Sampling 400K Method on Validation Data

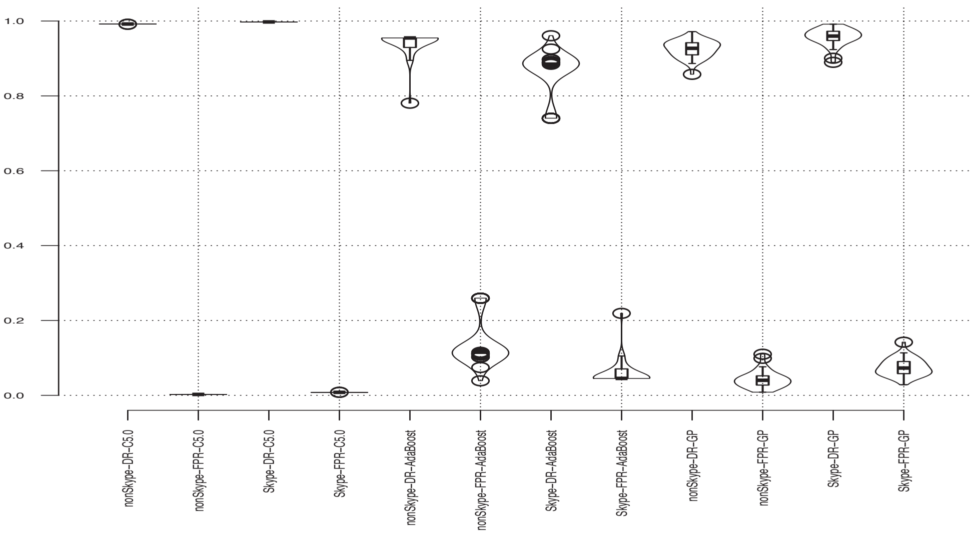


Figure A.11: Results of Three Machine Learning Algorithms Using the Uniform Random *N* Sampling 800K Method on Training Data

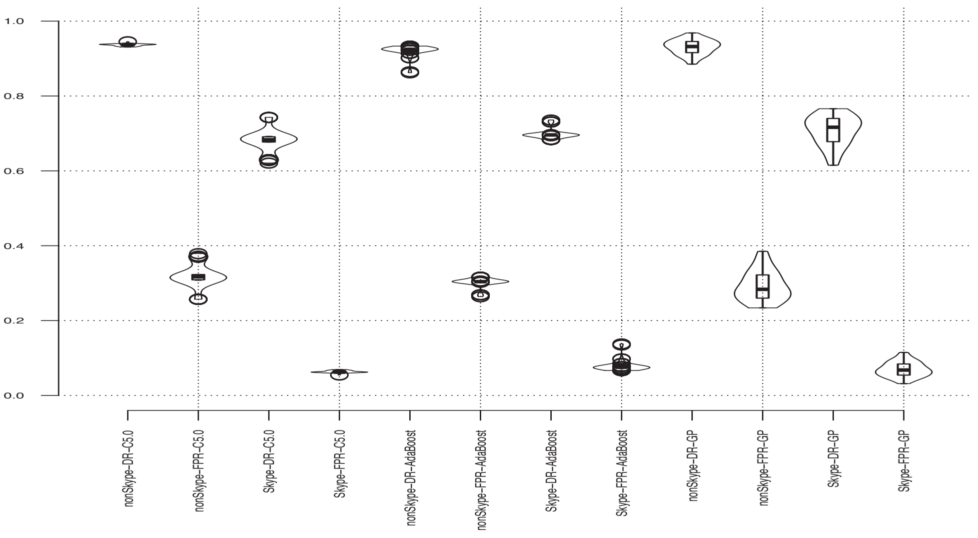


Figure A.12: Results of Three Machine Learning Algorithms Using the Uniform Random *N* Sampling 800K Method on Validation Data

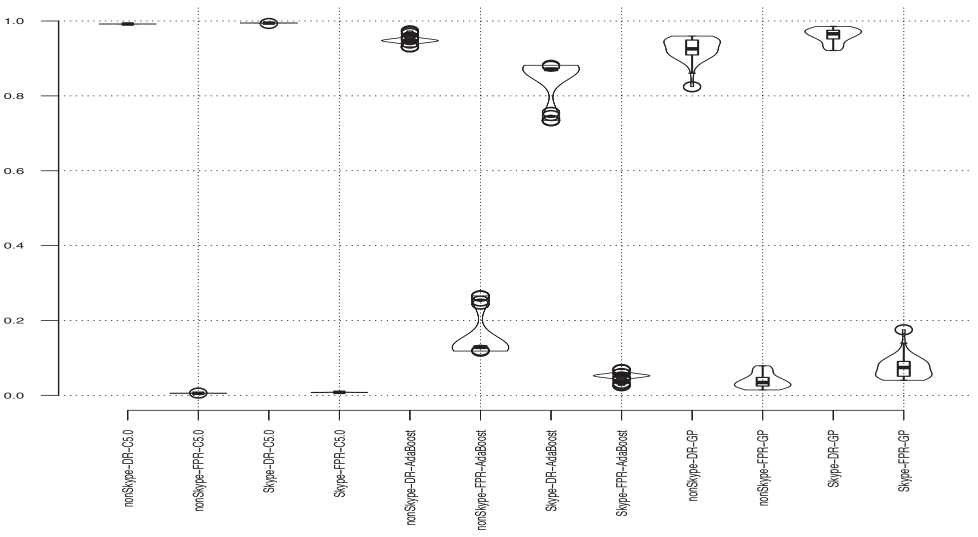


Figure A.13: Results of Three Machine Learning Algorithms Using the Uniform Random *N* Sampling 1% Method on Training Data

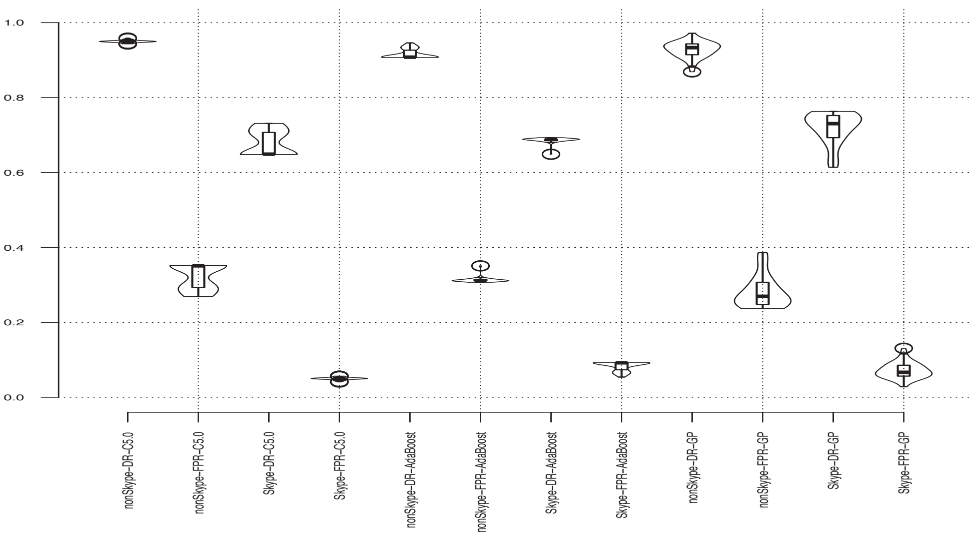


Figure A.14: Results of Three Machine Learning Algorithms Using the Uniform Random *N* Sampling 1% Method on Validation Data

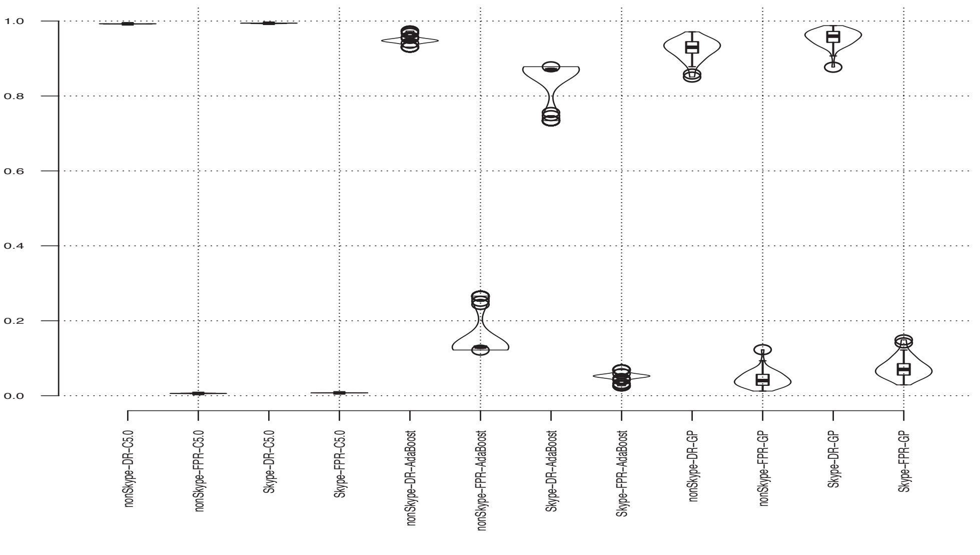


Figure A.15: Results of Three Machine Learning Algorithms Using the Uniform Random *N* Sampling 2% Method on Training Data

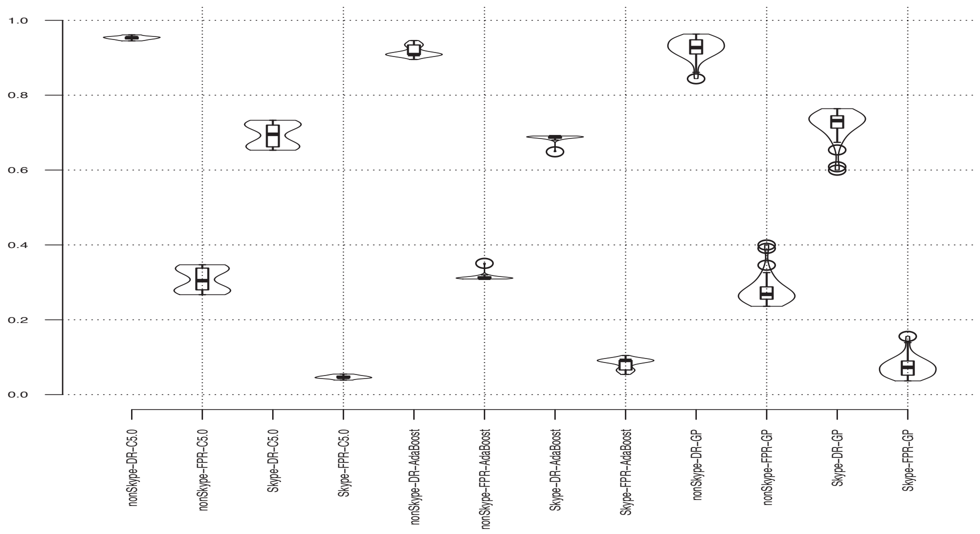


Figure A.16: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 2% Method on Validation Data

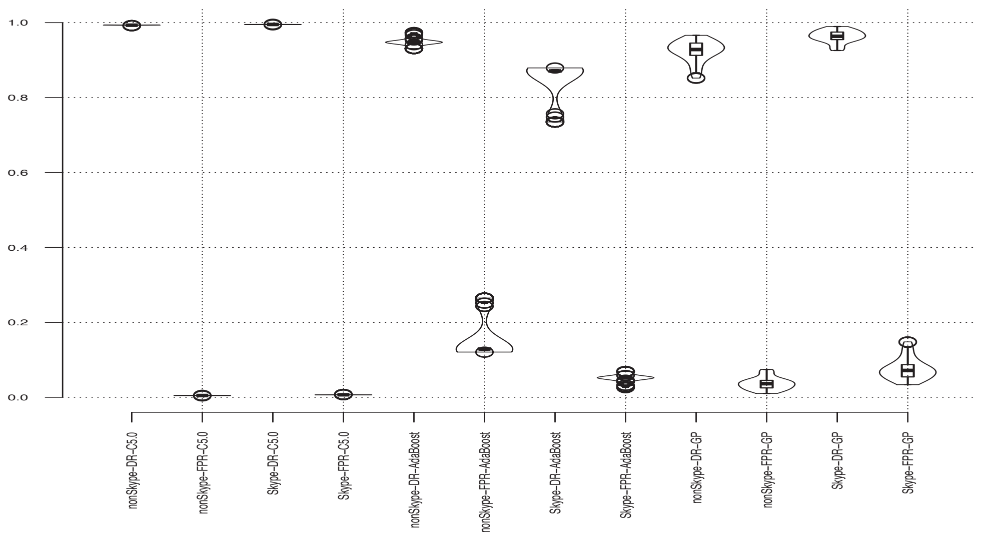


Figure A.17: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 3% Method on Training Data

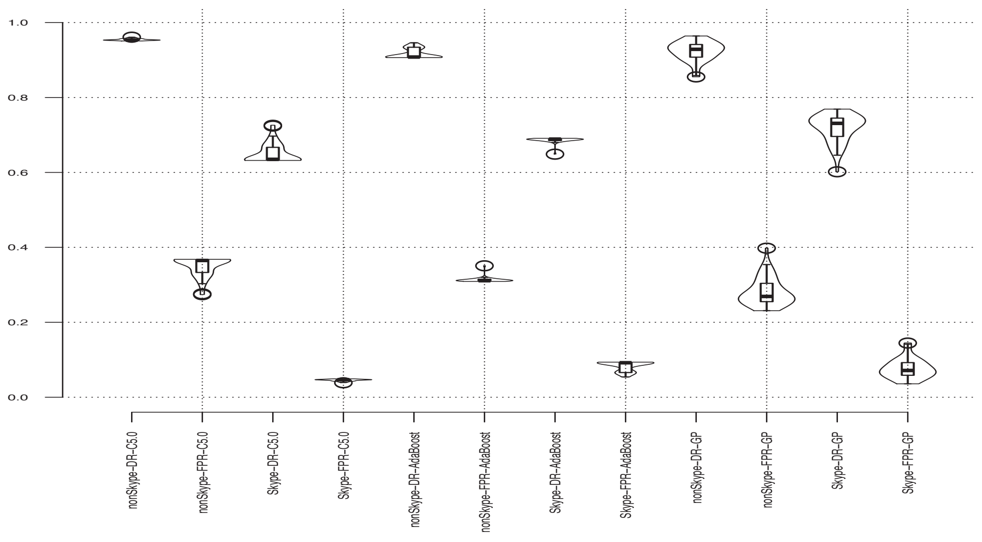


Figure A.18: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 3% Method on Validation Data

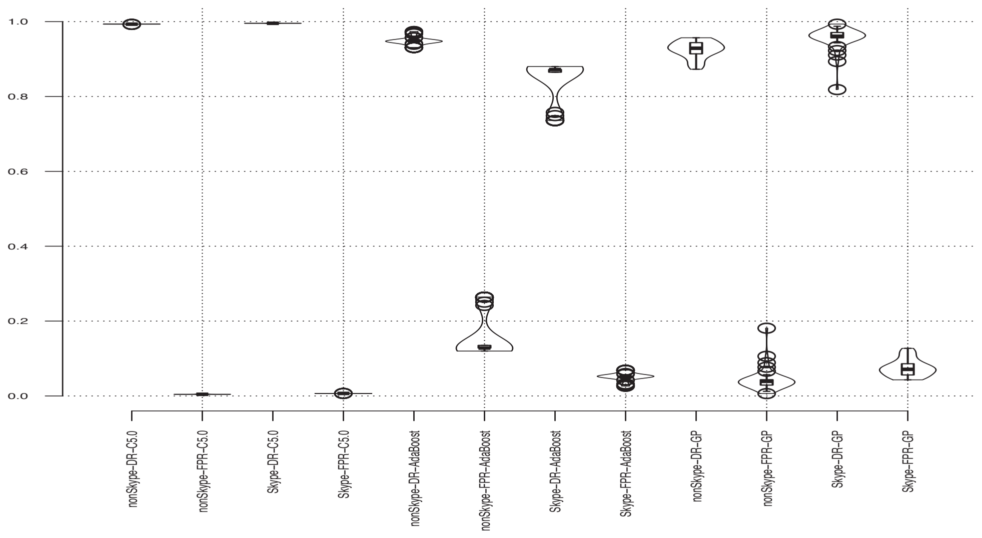


Figure A.19: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 4% Method on Training Data

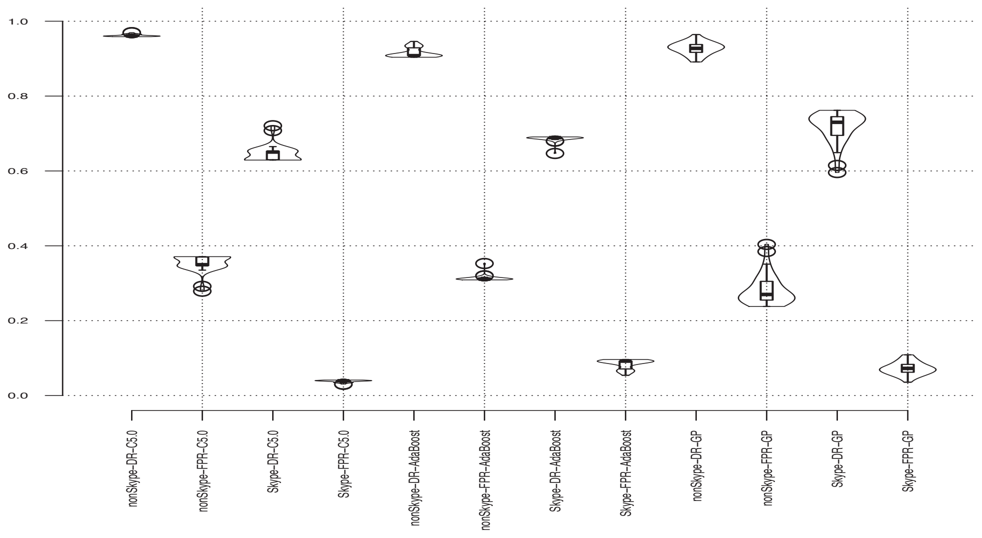


Figure A.20: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 4% Method on Validation Data

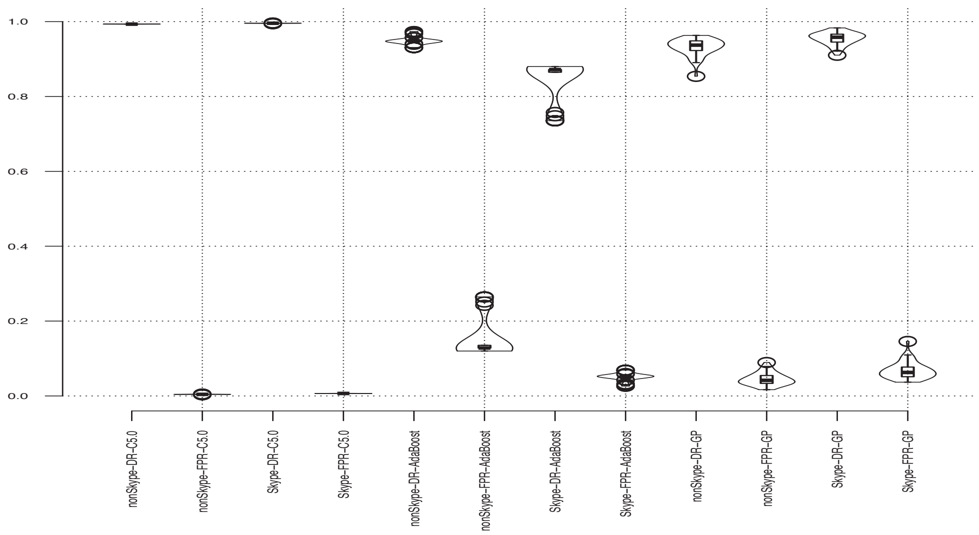


Figure A.21: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 5% Method on Training Data

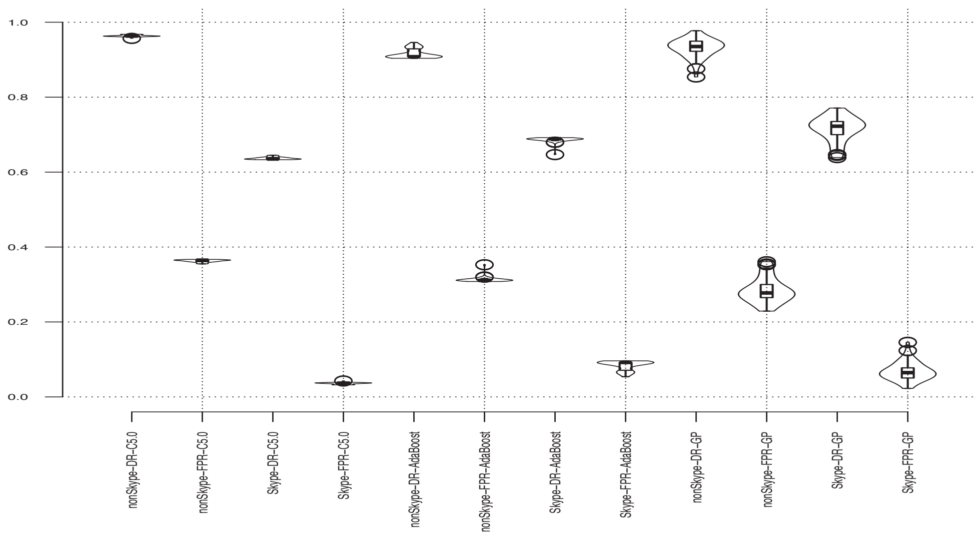


Figure A.22: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 5% Method on Validation Data

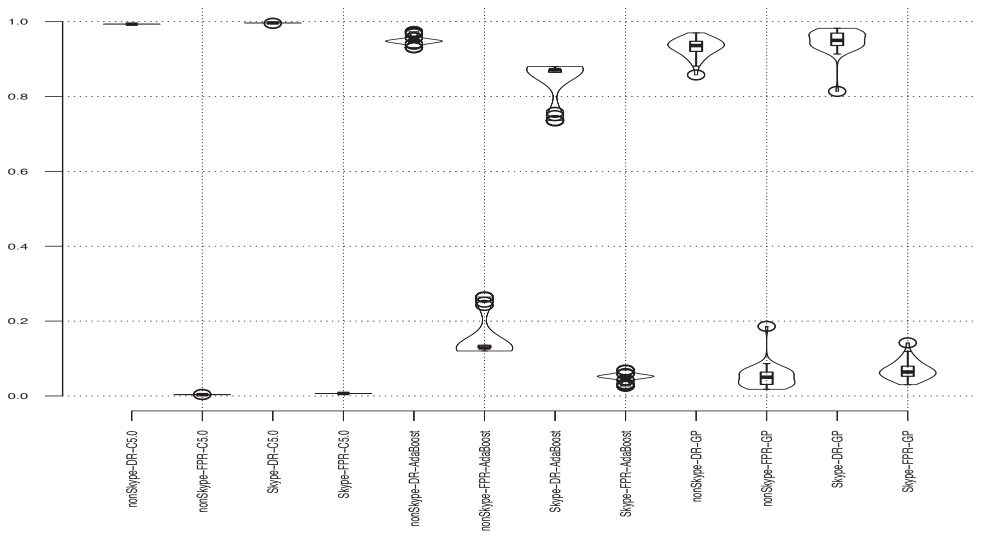


Figure A.23: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 6% Method on Training Data

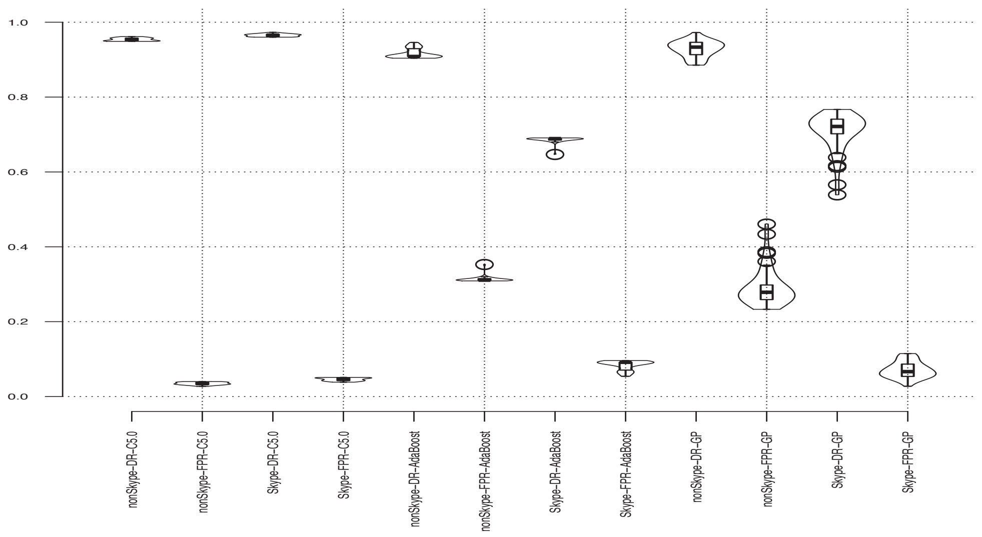


Figure A.24: Results of Three Machine Learning Algorithms Using the Uniform Random N Sampling 6% Method on Validation Data

A.2 Stratified Sampling

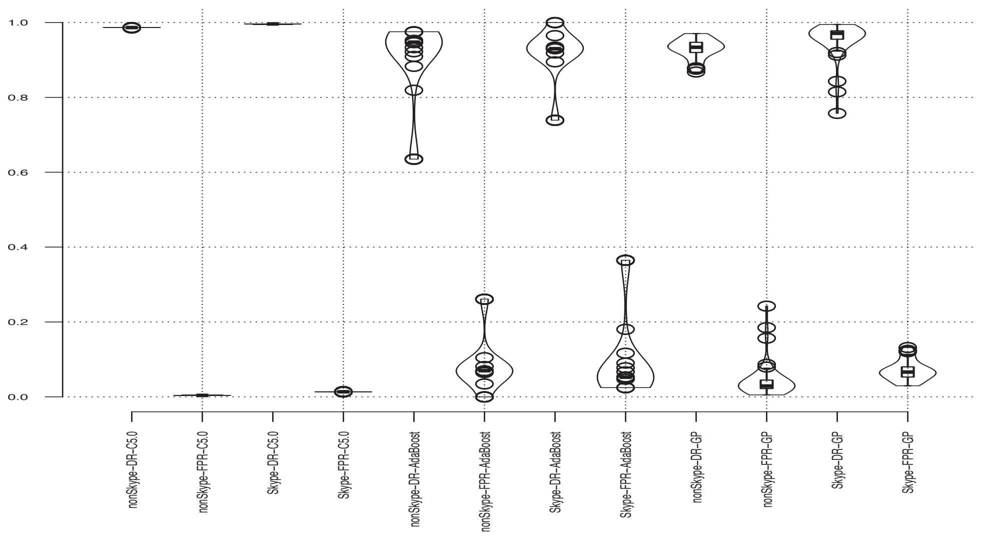


Figure A.25: Results of Three Machine Learning Algorithms Using the Stratified Sampling 30K Method on Training Data

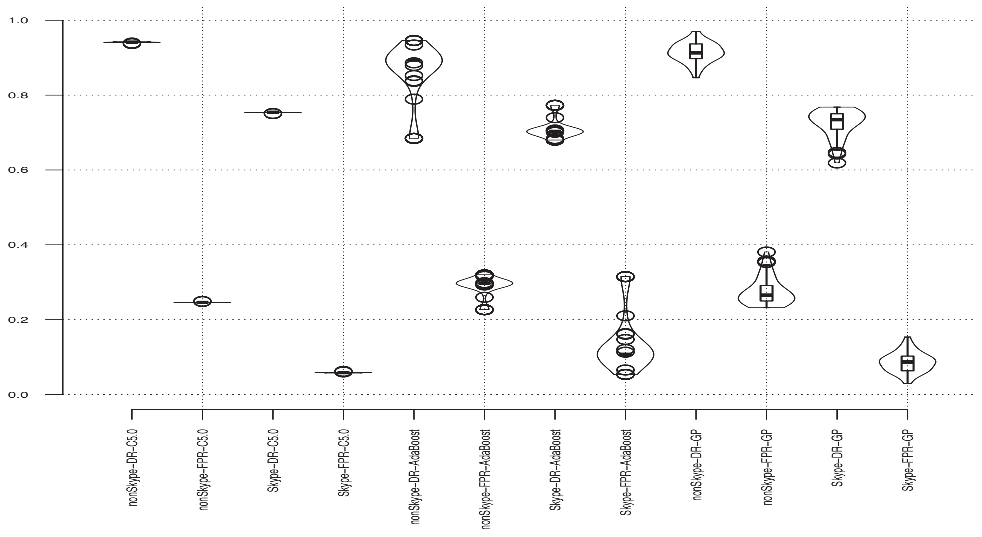


Figure A.26: Results of Three Machine Learning Algorithms Using the Stratified Sampling 30K Method on Validation Data

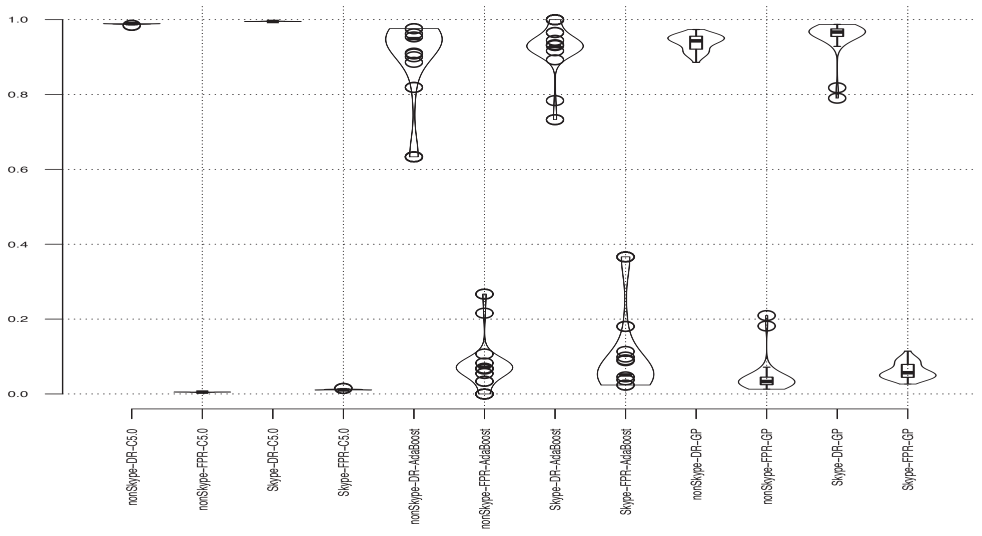


Figure A.27: Results of Three Machine Learning Algorithms Using the Stratified Sampling 60K Method on Training Data

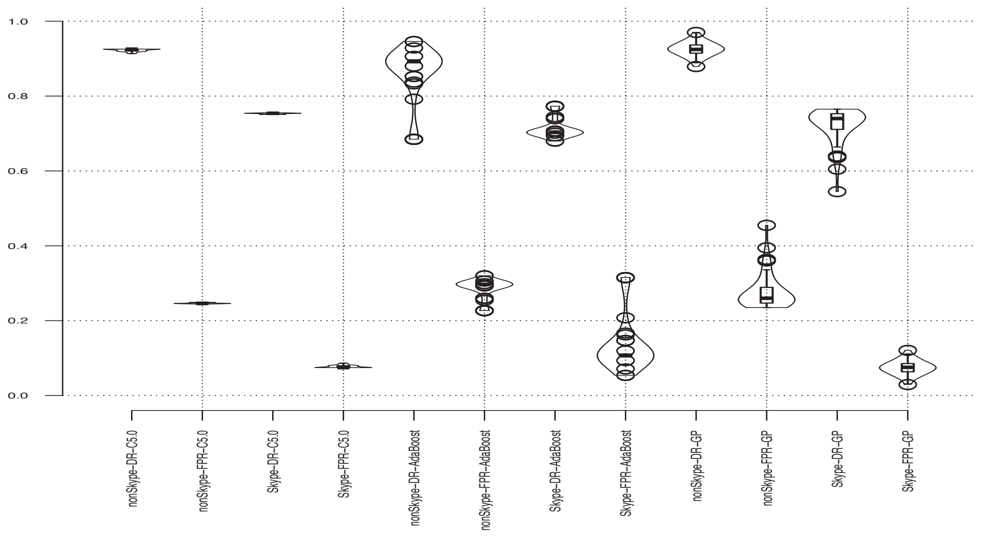


Figure A.28: Results of Three Machine Learning Algorithms Using the Stratified Sampling 60K Method on Validation Data

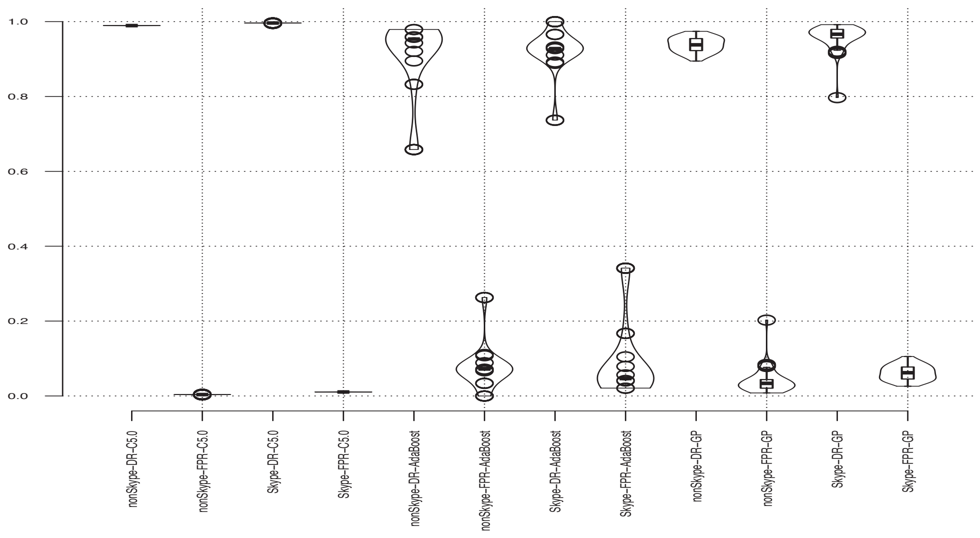


Figure A.29: Results of Three Machine Learning Algorithms Using the Stratified Sampling 100K Method on Training Data

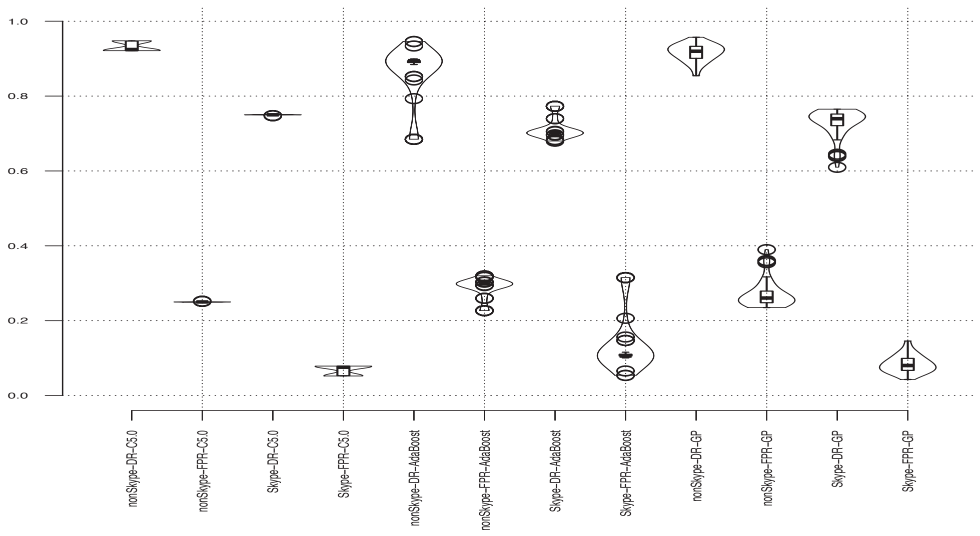


Figure A.30: Results of Three Machine Learning Algorithms Using the Stratified Sampling 100K Method on Validation Data

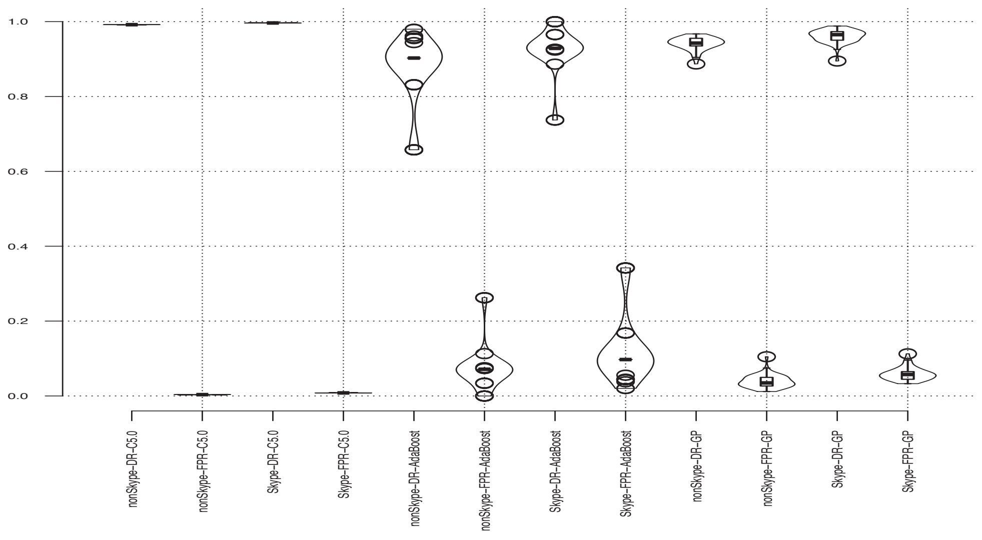


Figure A.31: Results of Three Machine Learning Algorithms Using the Stratified Sampling 200K Method on Training Data

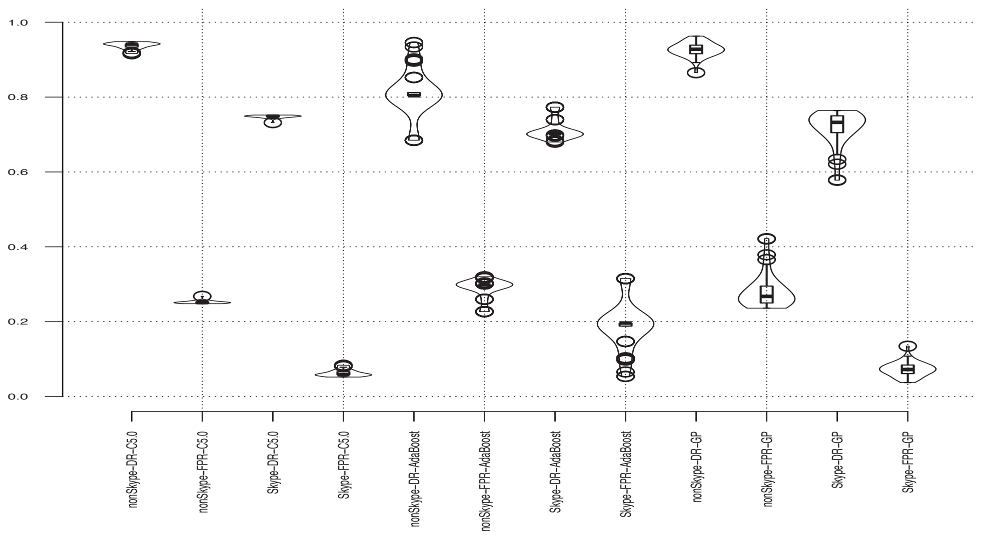


Figure A.32: Results of Three Machine Learning Algorithms Using the Stratified Sampling 200K Method on Validation Data

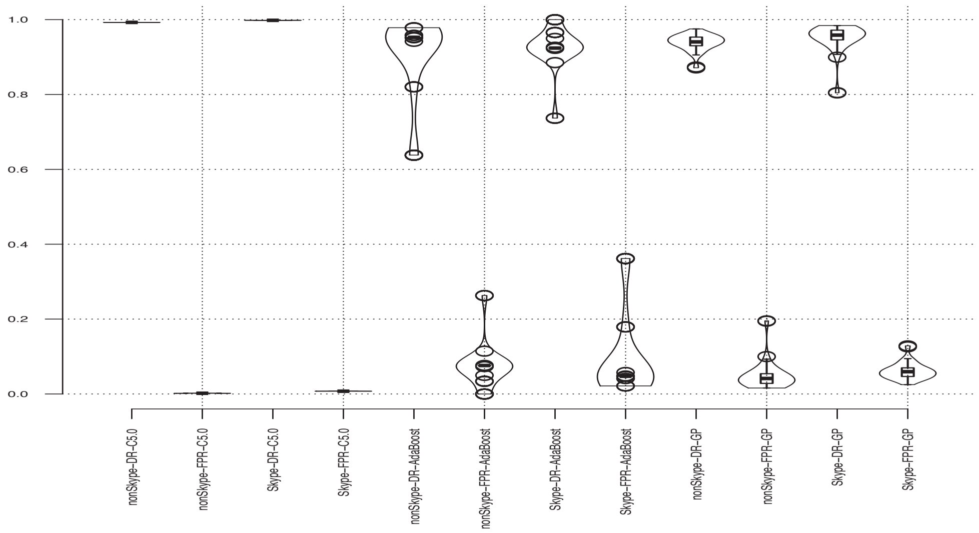


Figure A.33: Results of Three Machine Learning Algorithms Using the Stratified Sampling 400K Method on Training Data

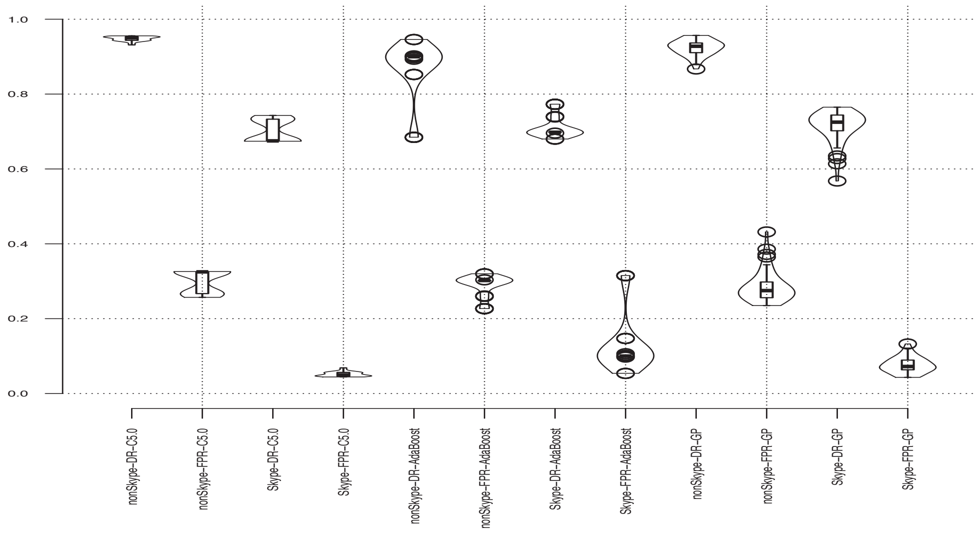


Figure A.34: Results of Three Machine Learning Algorithms Using the Stratified Sampling 400K Method on Validation Data

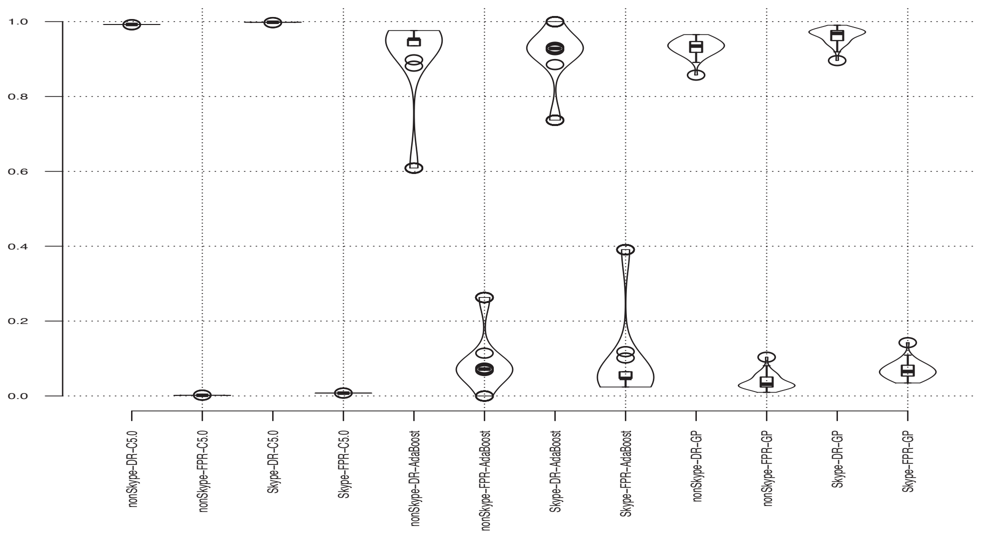


Figure A.35: Results of Three Machine Learning Algorithms Using the Stratified Sampling 800K Method on Training Data

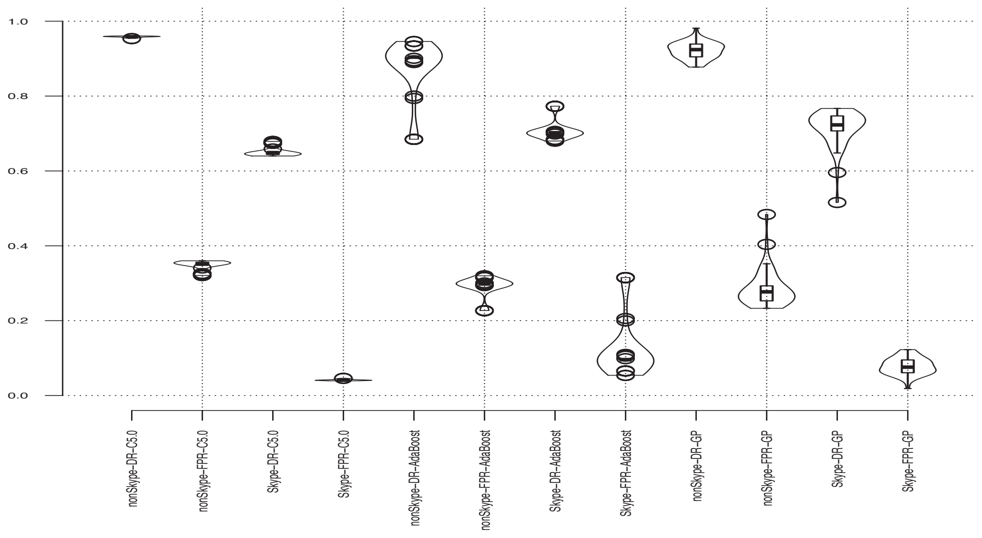


Figure A.36: Results of Three Machine Learning Algorithms Using the Stratified Sampling 800K Method on Validation Data

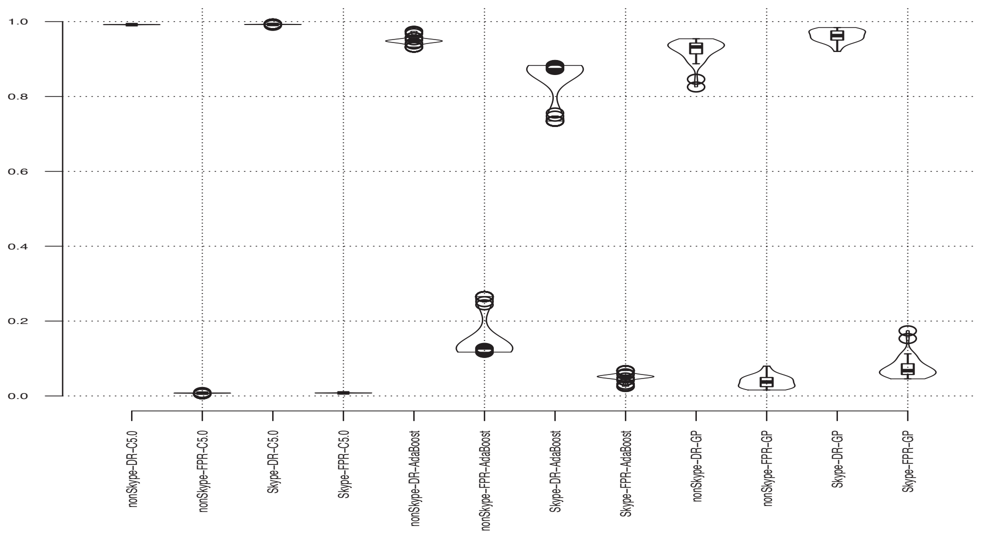


Figure A.37: Results of Three Machine Learning Algorithms Using the Stratified Sampling 1% Method on Training Data

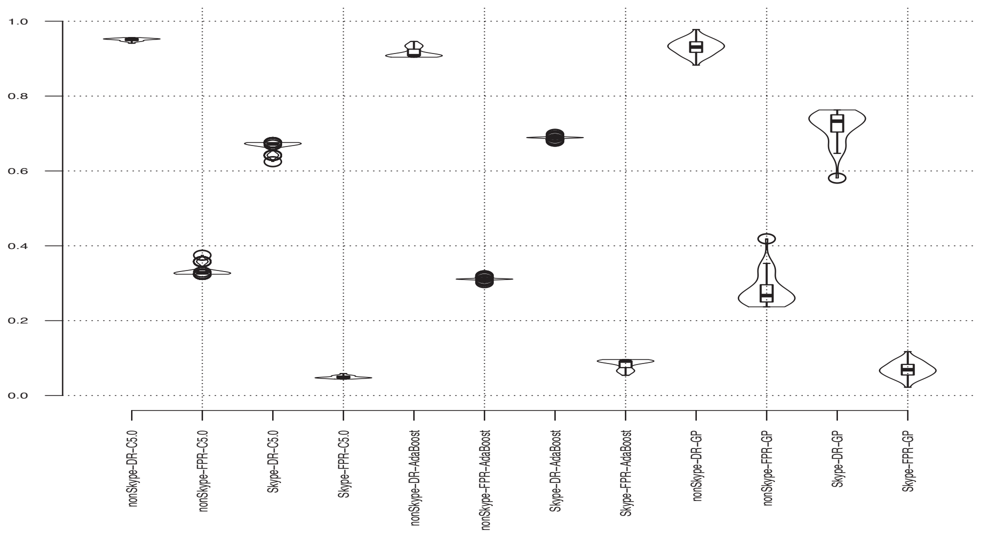


Figure A.38: Results of Three Machine Learning Algorithms Using the Stratified Sampling 1% Method on Validation Data

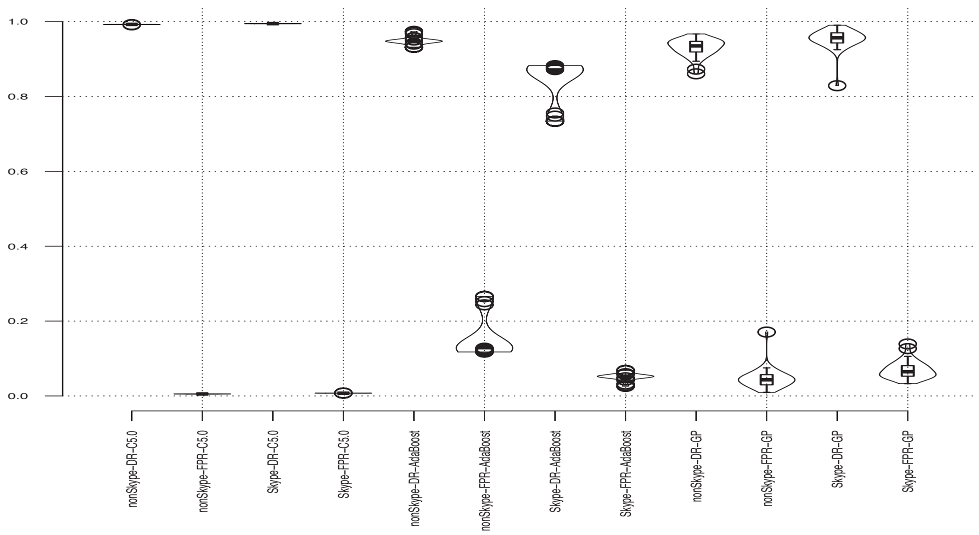


Figure A.39: Results of Three Machine Learning Algorithms Using the Stratified Sampling 2% Method on Training Data

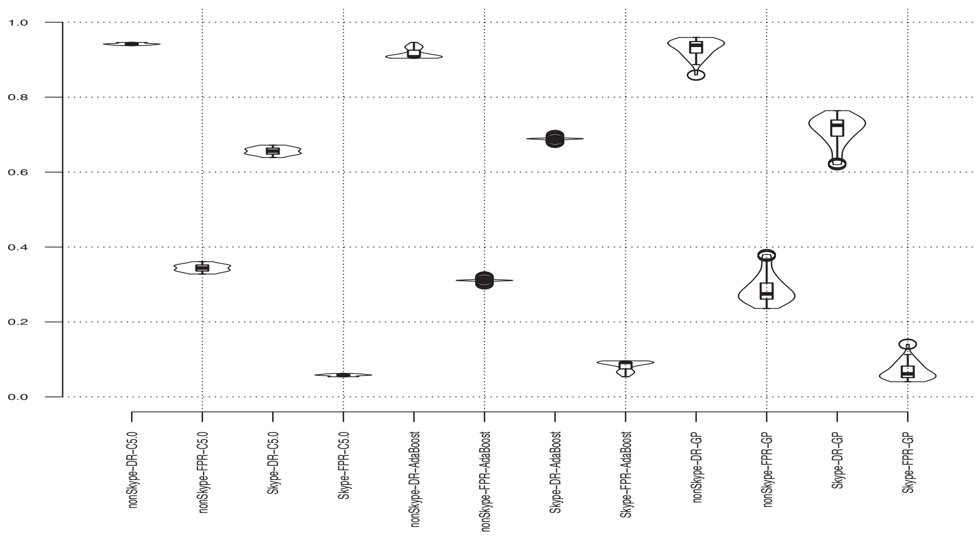


Figure A.40: Results of Three Machine Learning Algorithms Using the Stratified Sampling 2% Method on Validation Data

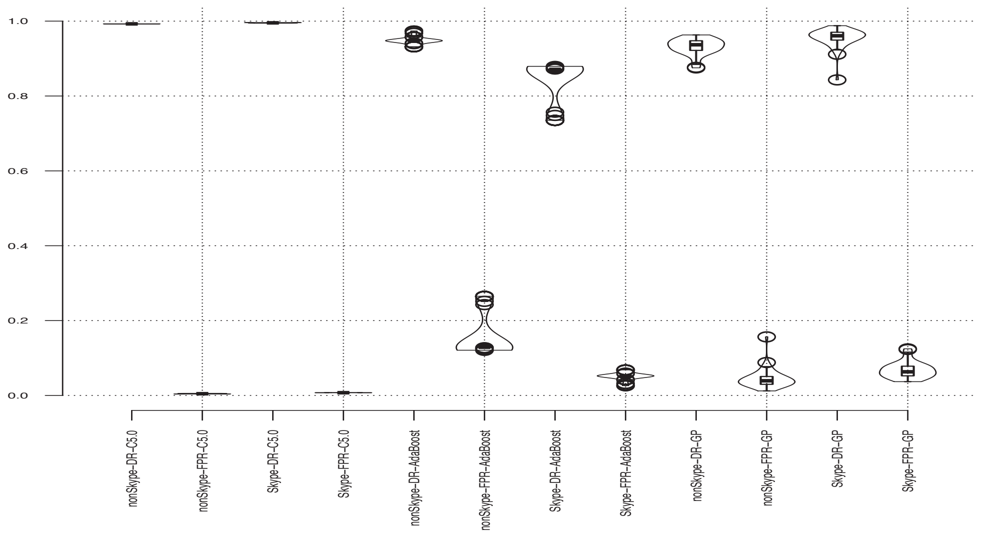


Figure A.41: Results of Three Machine Learning Algorithms Using the Stratified Sampling 3% Method on Training Data

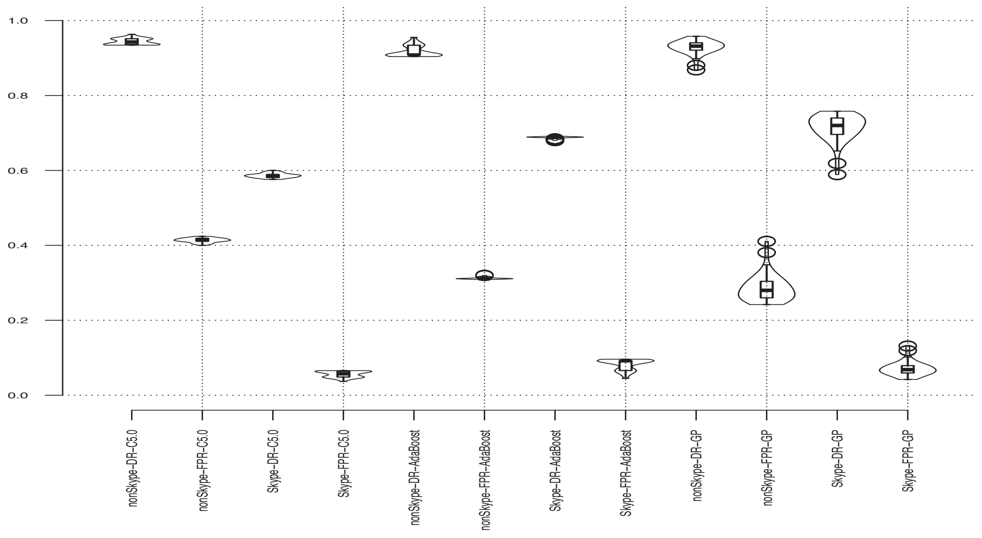


Figure A.42: Results of Three Machine Learning Algorithms Using the Stratified Sampling 3% Method on Validation Data

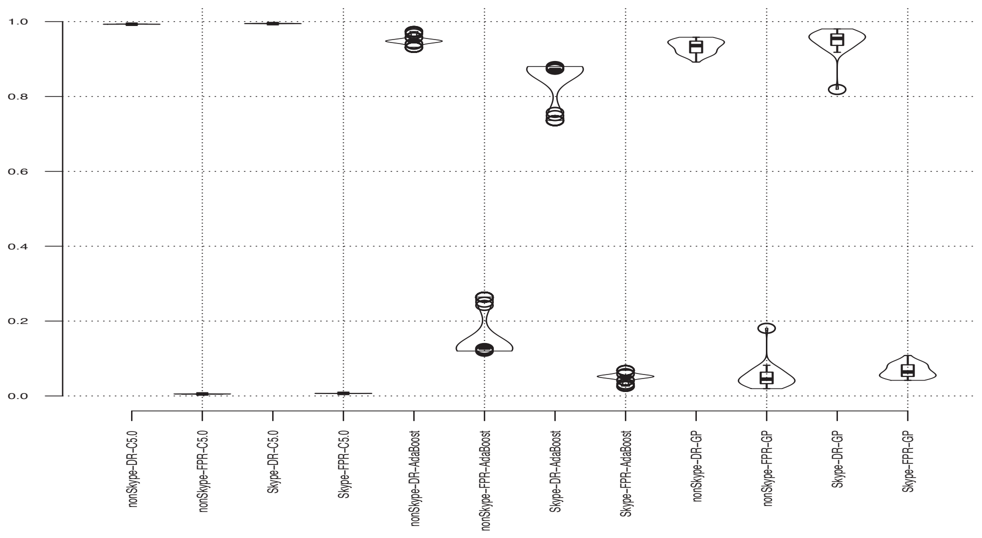


Figure A.43: Results of Three Machine Learning Algorithms Using the Stratified Sampling 4% Method on Training Data

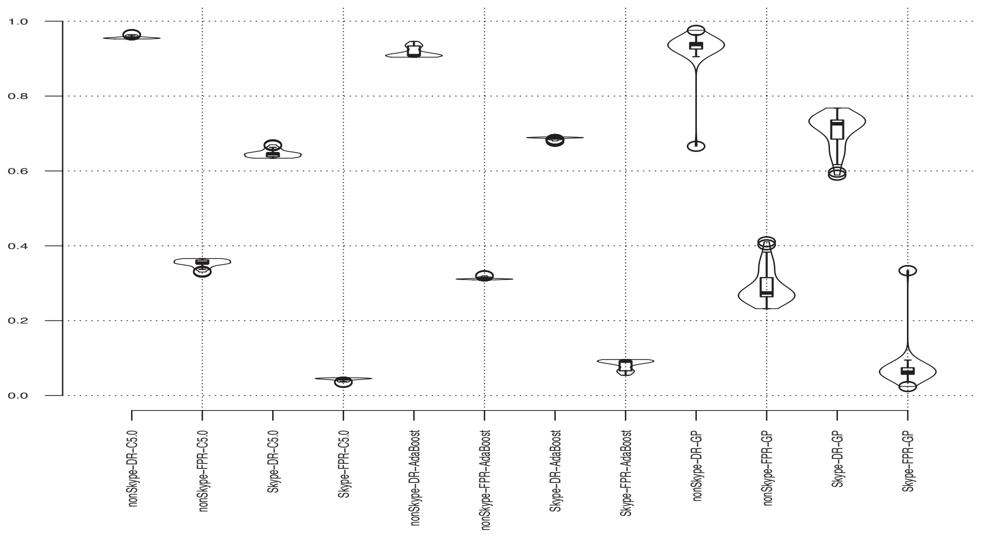


Figure A.44: Results of Three Machine Learning Algorithms Using the Stratified Sampling 4% Method on Validation Data

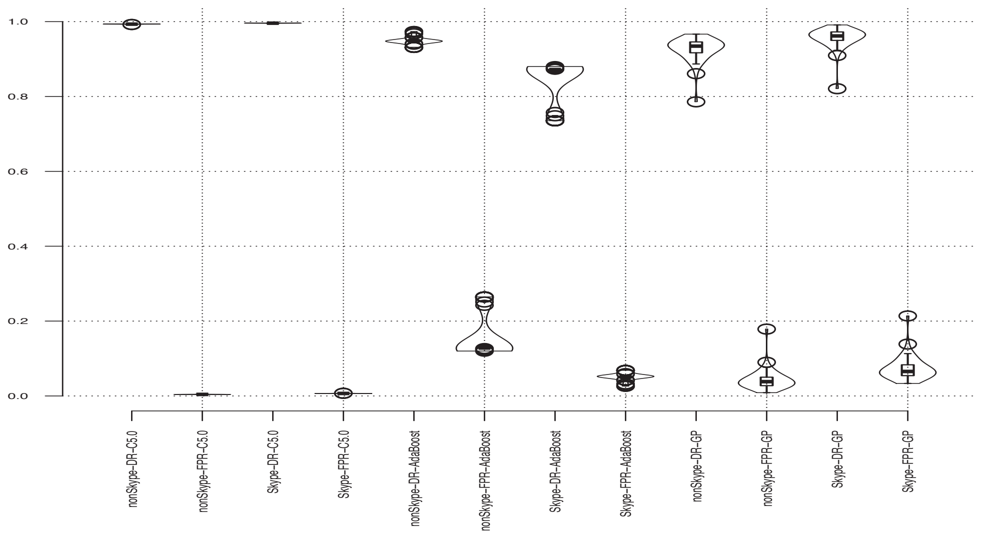


Figure A.45: Results of Three Machine Learning Algorithms Using the Stratified Sampling 5% Method on Training Data

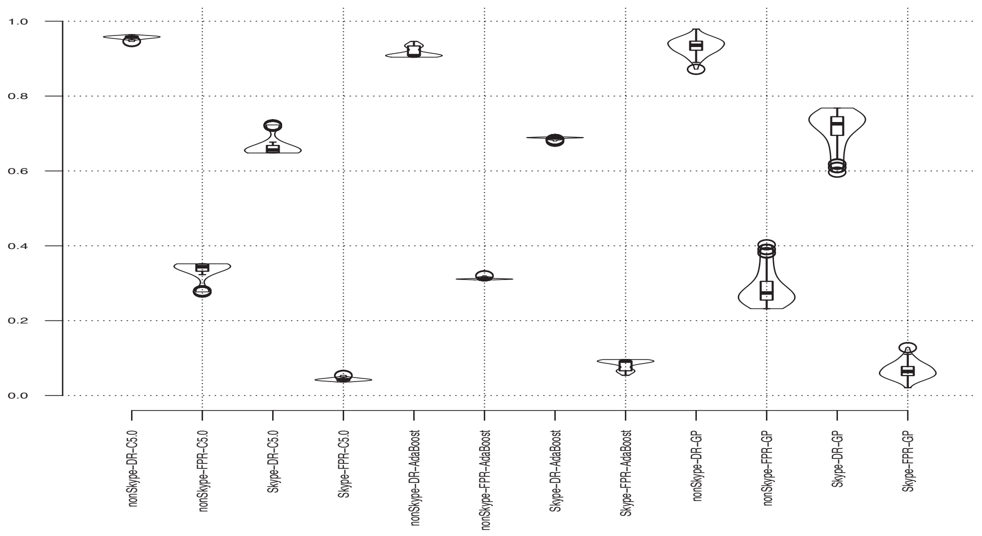


Figure A.46: Results of Three Machine Learning Algorithms Using the Stratified Sampling 5% Method on Validation Data

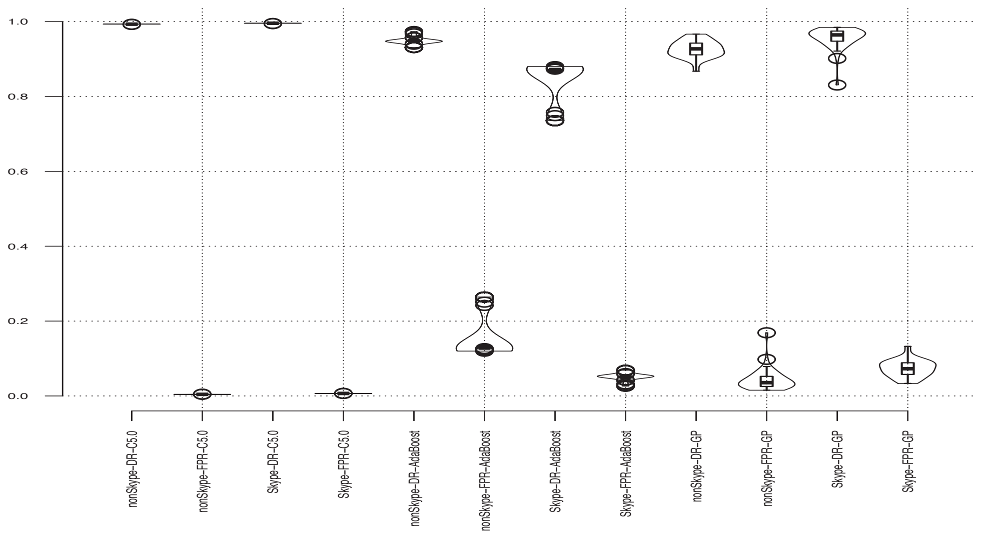


Figure A.47: Results of Three Machine Learning Algorithms Using the Stratified Sampling 6% Method on Training Data

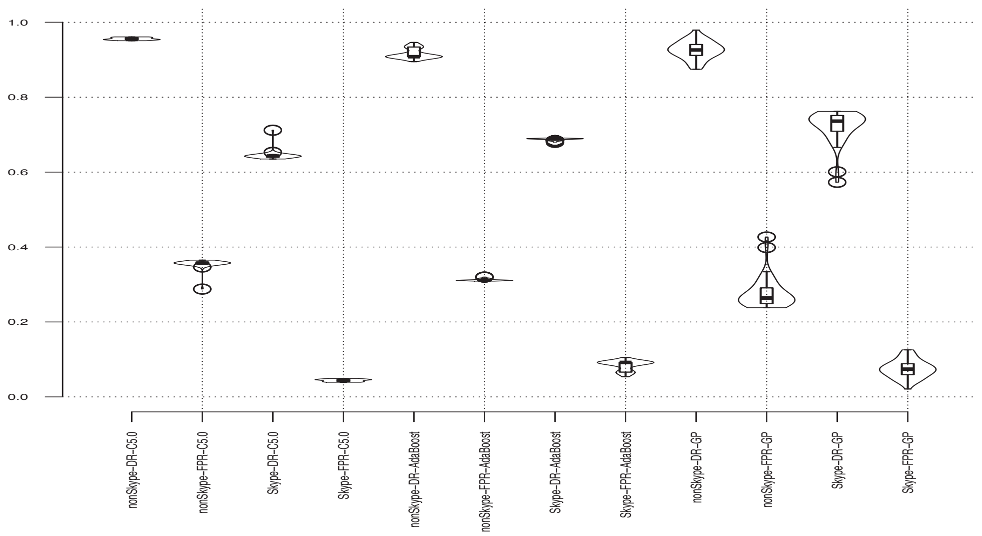


Figure A.48: Results of Three Machine Learning Algorithms Using the Stratified Sampling 6% Method on Validation Data

A.3 Continuous Sampling

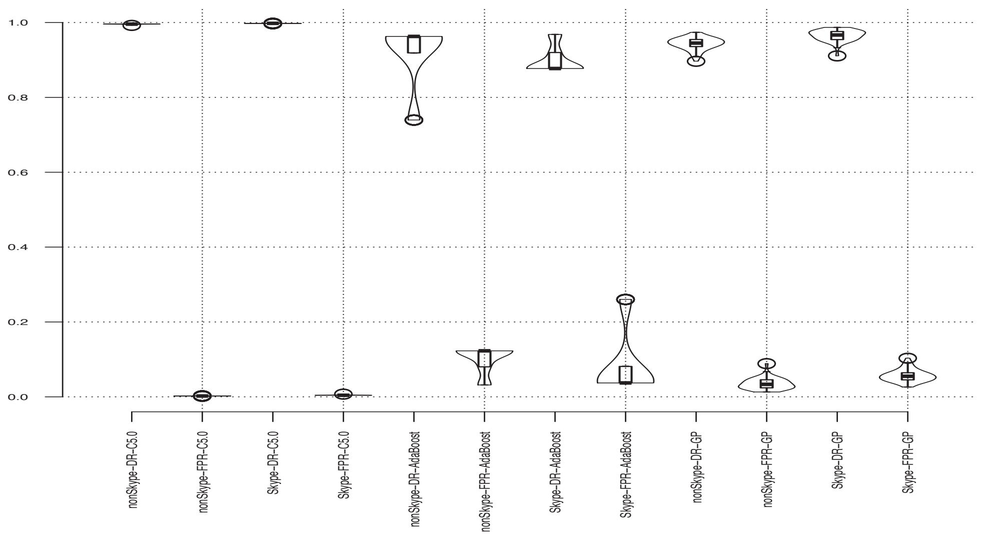


Figure A.49: Results of Three Machine Learning Algorithms Using the Continuous 30K Method on Training Data

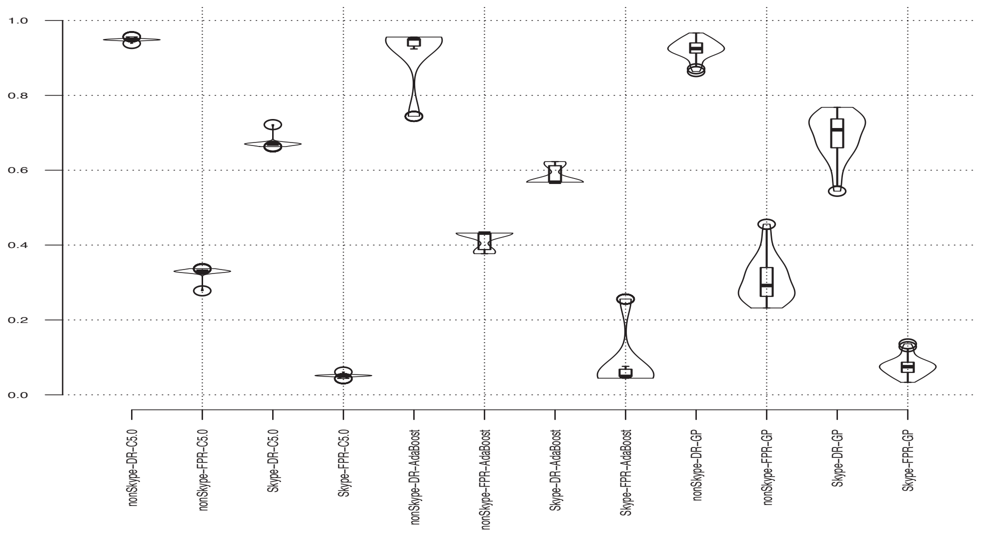


Figure A.50: Results of Three Machine Learning Algorithms Using the Continuous 30K Method on Validation Data

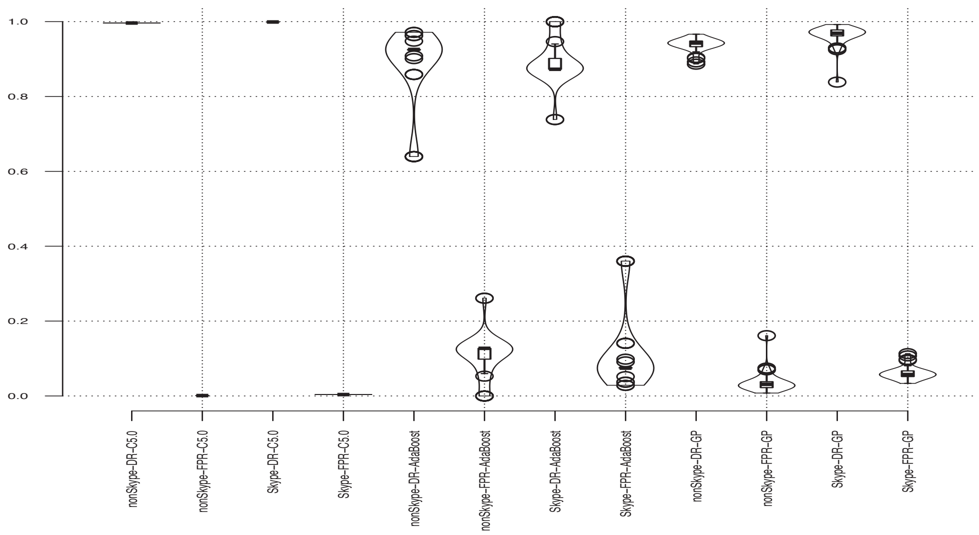


Figure A.51: Results of Three Machine Learning Algorithms Using the Continuous 60K Method on Training Data

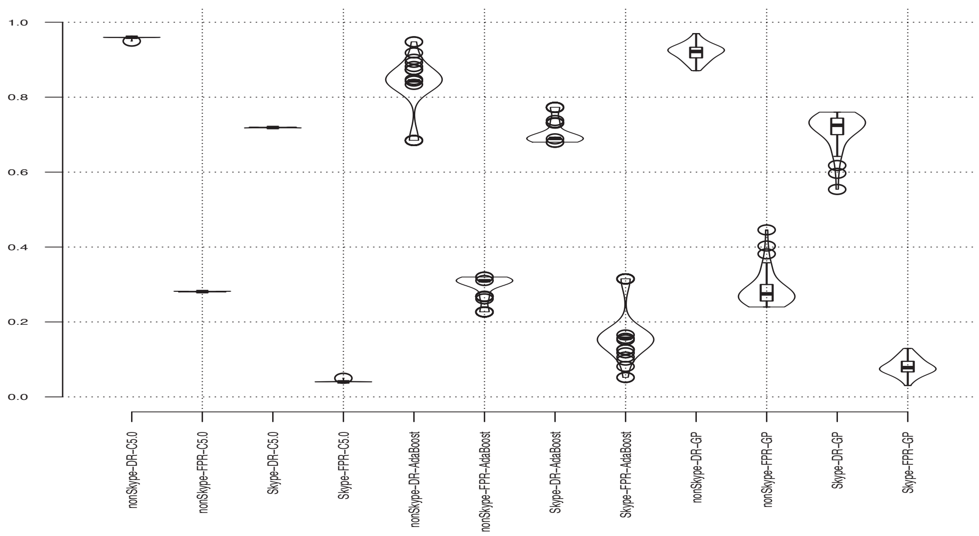


Figure A.52: Results of Three Machine Learning Algorithms Using the Continuous 60K Method on Validation Data

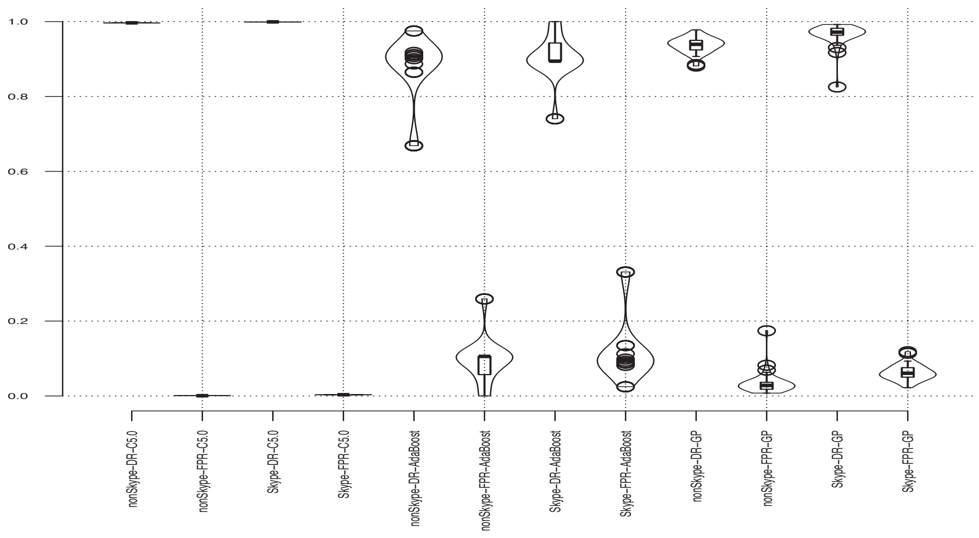


Figure A.53: Results of Three Machine Learning Algorithms Using the Continuous 100K Method on Training Data

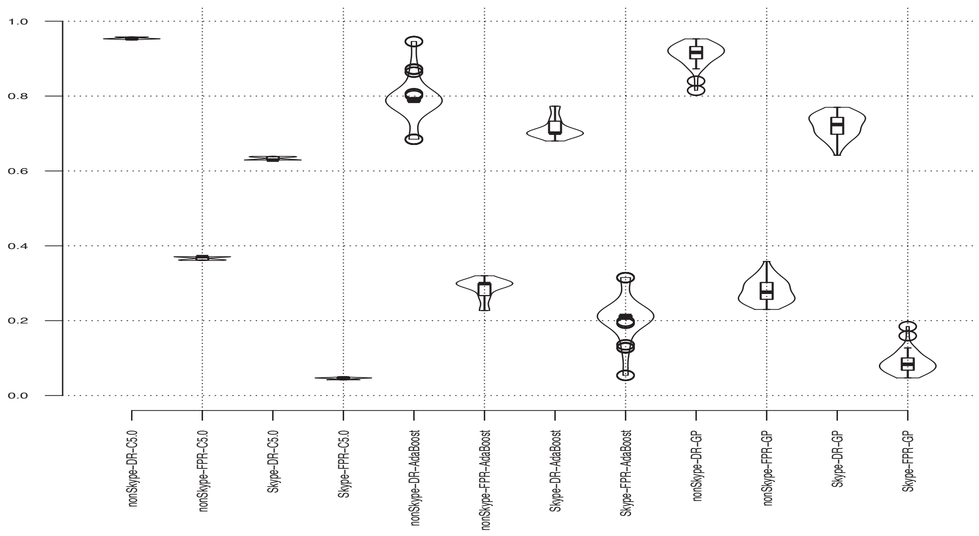


Figure A.54: Results of Three Machine Learning Algorithms Using the Continuous 100K Method on Validation Data

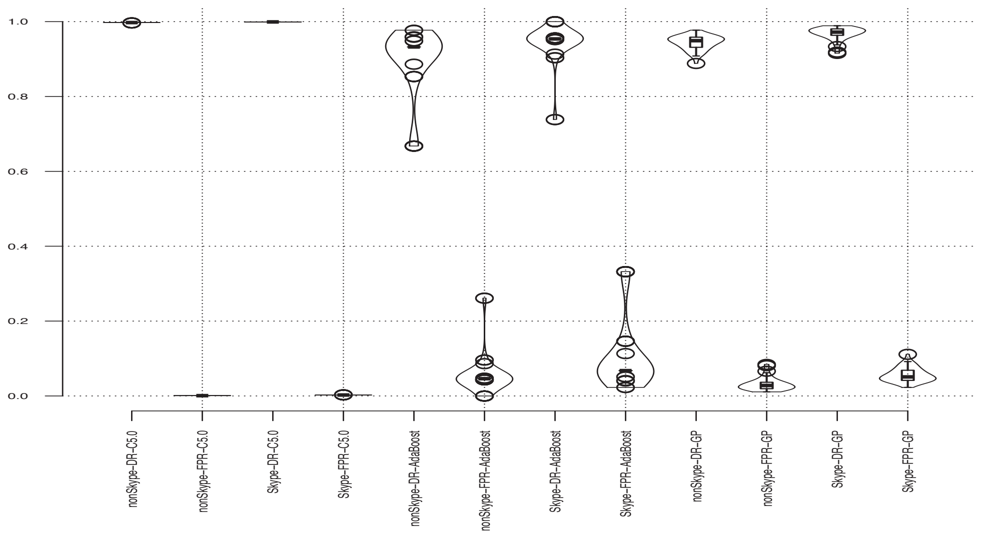


Figure A.55: Results of Three Machine Learning Algorithms Using the Continuous 200K Method on Training Data

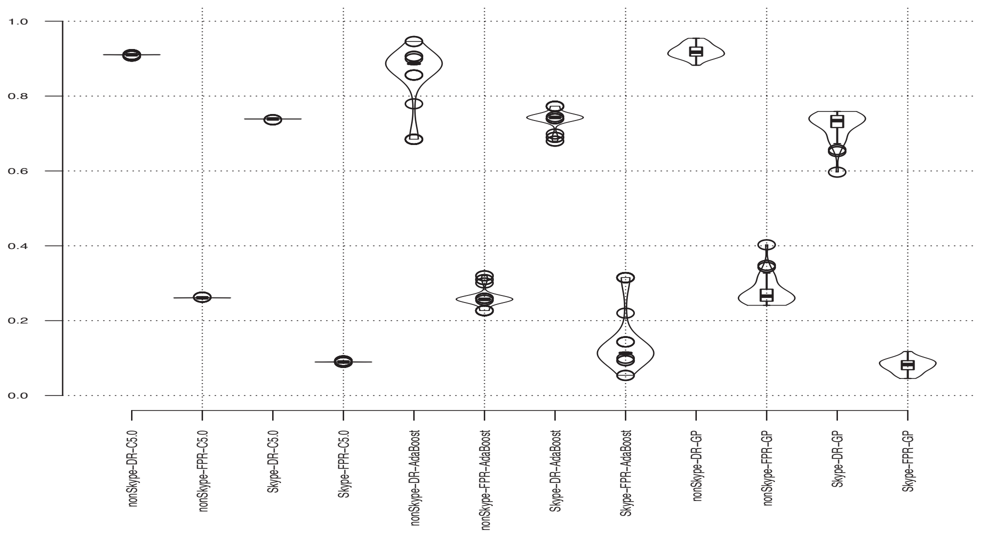


Figure A.56: Results of Three Machine Learning Algorithms Using the Continuous 200K Method on Validation Data

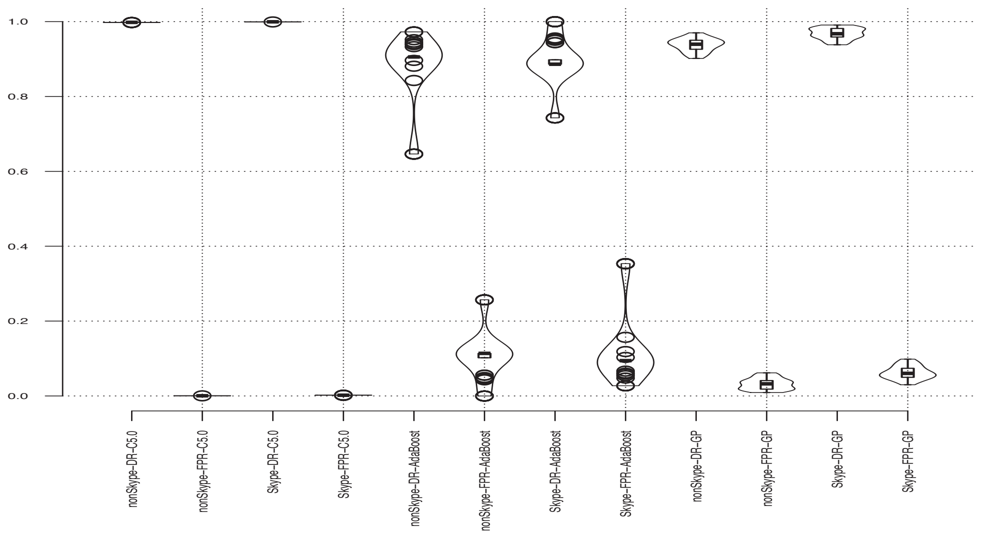


Figure A.57: Results of Three Machine Learning Algorithms Using the Continuous 400K Method on Training Data

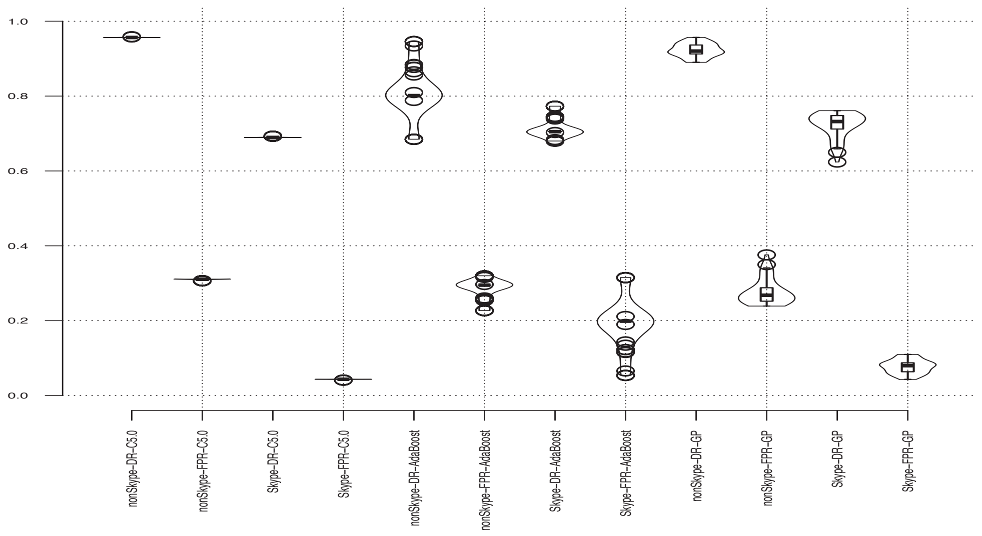


Figure A.58: Results of Three Machine Learning Algorithms Using the Continuous 400K Method on Validation Data

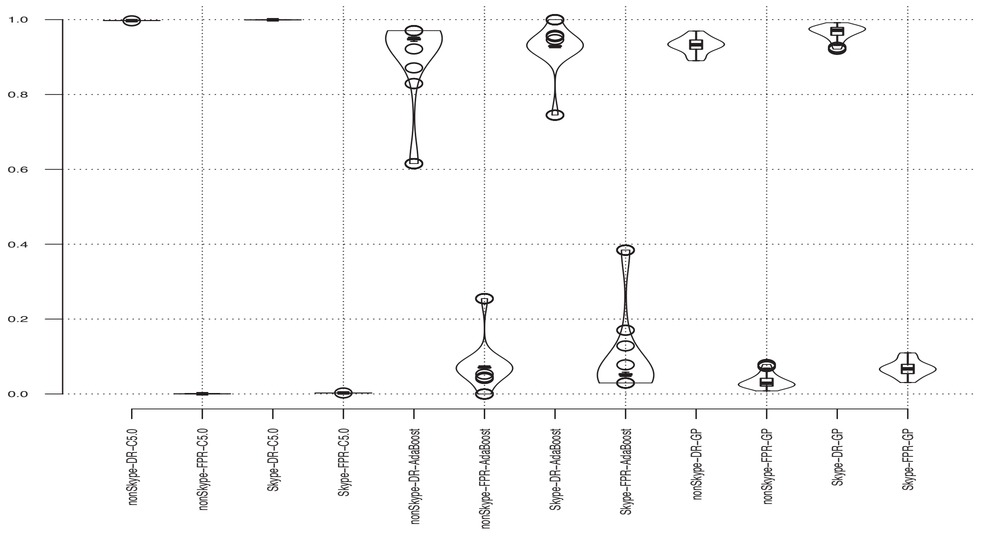


Figure A.59: Results of Three Machine Learning Algorithms Using the Continuous 800K Method on Training Data

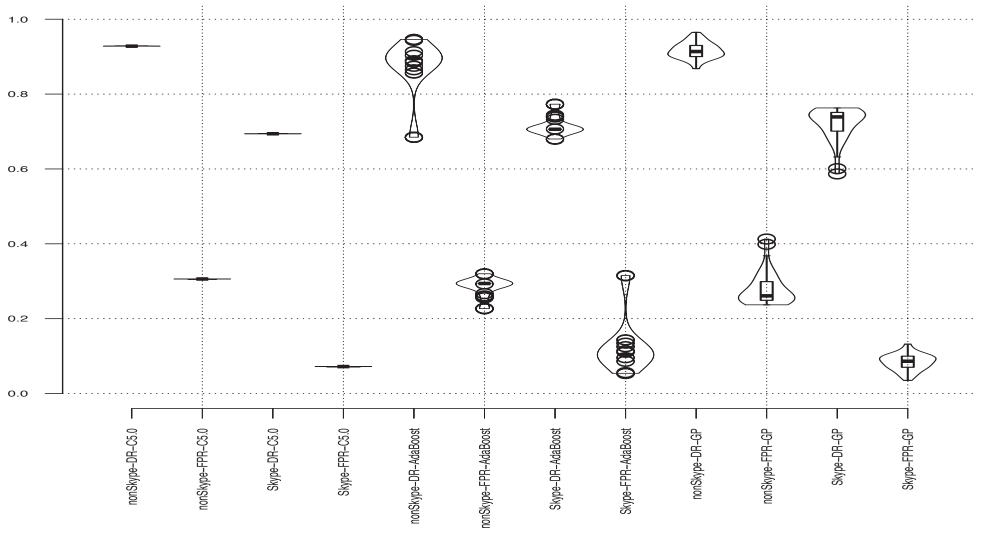


Figure A.60: Results of Three Machine Learning Algorithms Using the Continuous 800K Method on Validation Data

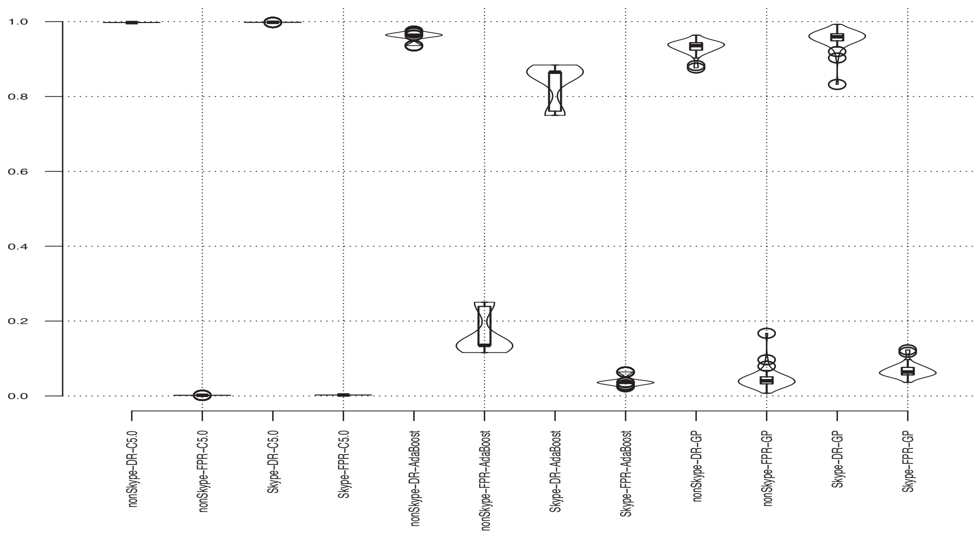


Figure A.61: Results of Three Machine Learning Algorithms Using the Continuous 30 Minute Method on Training Data

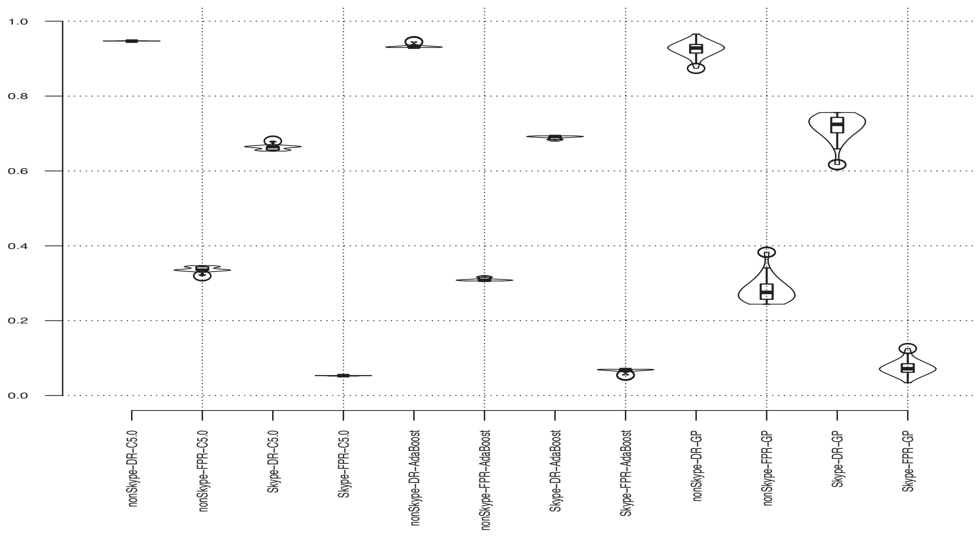


Figure A.62: Results of Three Machine Learning Algorithms Using the Continuous 30 Minute method on Validation Data

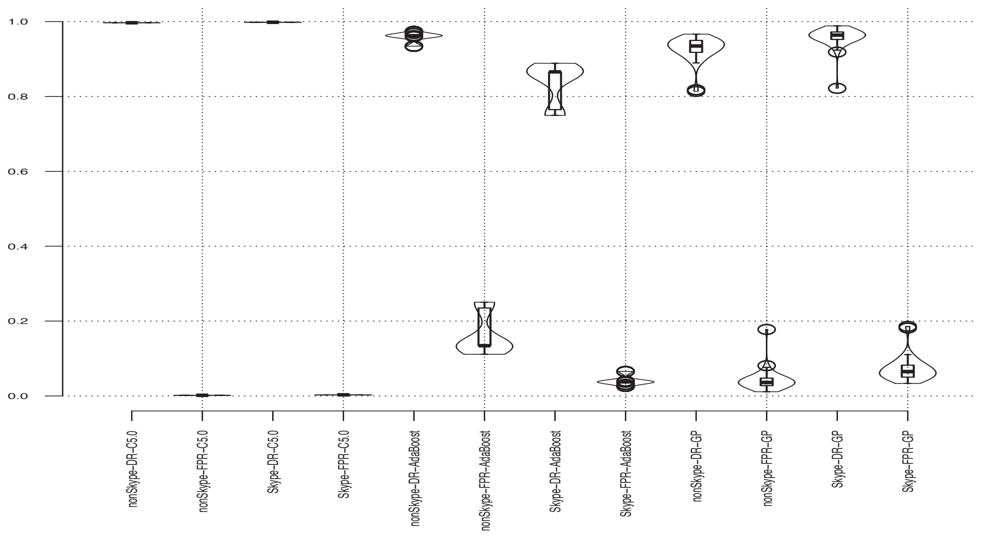


Figure A.63: Results of Three Machine learning Algorithms Using the Continuous 60 Minutes Method on Training Data

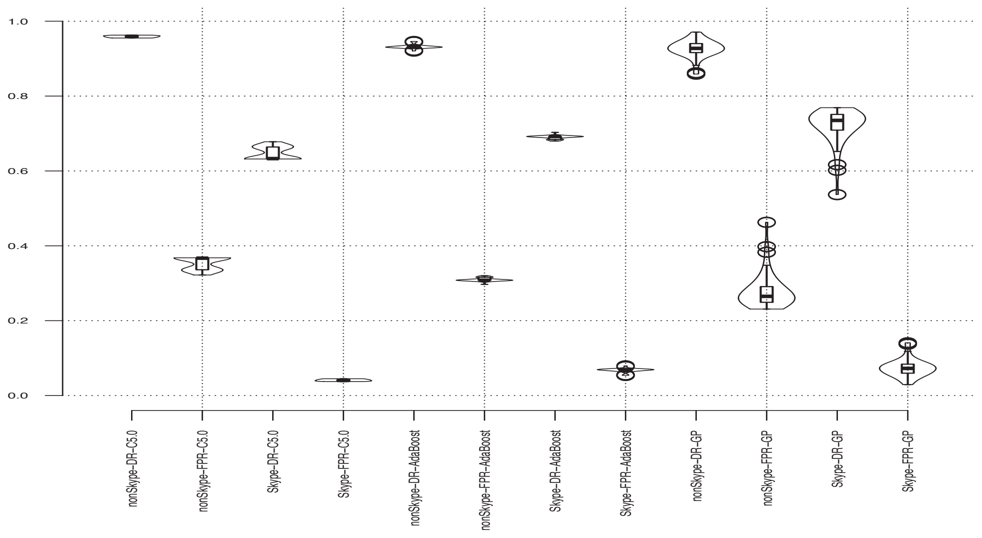


Figure A.64: Results of Three Machine Learning Algorithms Using the Continuous 60 Minute Method on Validation Data

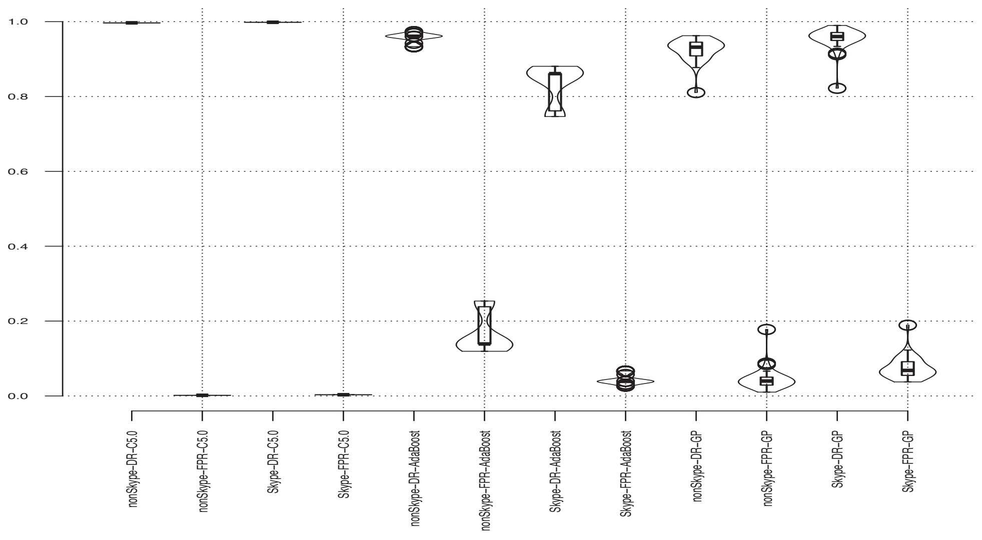


Figure A.65: Results of Three Machine Learning Algorithms Using the Continuous 90 Minutes Method on Training Data

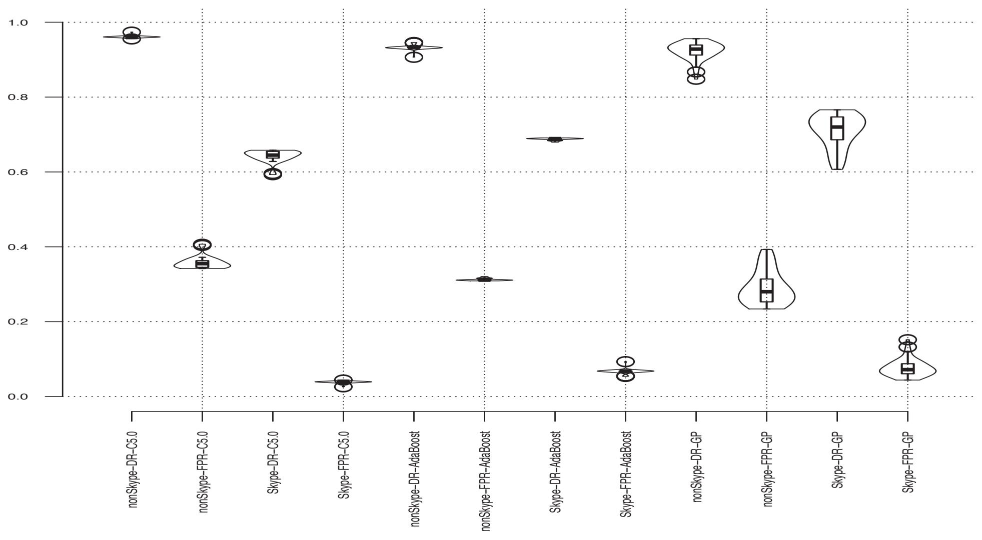


Figure A.66: Results of Three Machine Learning Algorithms Using the Continuous 90 Minute Method on Validation Data

Appendix B

One-way ANOVA Statistical Analysis Tests

A one-way ANOVA statistical analysis test was employed to test the Null hypothesis that all the means of the DRs and FPRs have an equal mean.

B.1 Choosing the Sampling Method

Table B.1: A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms for the Subset Sampling Techniques on the Training Data Set

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Columns	15.7262	98	0.1605	125.4834	0
Error	6.2036	4851	0.0013		
Tota	21.9297	4949			

Table B.2: A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms for the Subset Sampling Techniques on the Training Data Set

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Columns	5.3618	98	0.0547	39.5126	0
Error	6.7171	4851	0.0014		
Tota	12.0790	4949			

Table B.3: A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms for the Subset Sampling Techniques on the Validation Data Set

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Columns	8.2791	98	0.0845	117.6585	0
Error	3.4831	4851	7.1801e-04		
Tota	11.7622	4949			

Table B.4: A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms for the Subset Sampling Techniques on the Validation Data Set

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Columns	4.9024	98	0.0500	63.1761	0
Error	3.8411	4851	7.9182e-04		
Tota	8.7435	4949			

Table B.5: A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms on the Subset Sampling Training Data Set

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Columns	0.6439	2	0.3219	236.1756	1.2316e-46
Error	0.2004	147	0.0014		
Tota	0.8443	149			

Table B.6: A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms on the Subset Sampling Training Data Set

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
textbfColumns	0.1019	2	0.0509	233.4894	2.3364e-46
Error	0.0321	147	2.1819e-04		
Tota	0.1340	149			

B.2 Subset Sampling Using the Uniform Random Method with a 6 Percent Sampling Rate

Table B.7: A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms Using the Packet Header-based Feature Set on Training Data Set

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Columns	1.5609	2	0.7805	438.5611	1.0889e-62
Error	0.2616	147	0.0018		
Tota	1.8225	149			

Table B.8: A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms Using the Packet Header-based Feature Set on Training Data Set

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
textbfColumns	0.0281	2	0.0141	316.7734	5.0874e-54
Error	0.0065	147	4.4363e-05		
Tota	0.0346	149			

B.3 Robustness of the Classifiers

B.3.1 Robustness of the Three ML Algorithms Using the Packet Header-based Feature Set

Table B.9: A One-way ANOVA Statistical Analysis Test for the Mean DR for the three ML Algorithms Using the Flow-based Feature Set on Training Data Set

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Columns	0.6150	2	0.3075	209.2938	9.7519e-44
Error	0.2160	147	0.0015		
Tota	0.8310	149			

Table B.10: A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data set

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
textbfColumns	0.1433	2	0.0717	182.2266	1.5870e-40
Error	0.0578	147	3.9333e-04		
Tota	0.2012	149			

B.3.2 Robustness of the Three ML Algorithms Using the Flow-based Feature Set

Table B.11: A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Skype Classification

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Columns	1.7421	2	0.8710	422.4915	1.1344e-61
Error	0.3031	147	0.0021		
Tota	2.0451	149			

Table B.12: A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Skype Classification

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
textbfColumns	0.2781	2	0.1391	123.0332	4.0233e-32
Error	0.1662	147	0.0011		
Tota	0.4443	149			

B.3.3 Multi-Classes of the Three ML Algorithms Using the Flow-based Feature Set

Table B.13: A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Gtalk Classification

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Columns	28.6823	2	14.3411	2.3786e+04	2.5960e-185
Error	0.0886	147	6.0293e-04		
Tota	28.7709	149			

Table B.14: A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Gtalk Classification

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
textbfColumns	0.0521	2	0.0260	323.0056	1.5878e-54
Error	0.0118	147	8.0593e-05		
Tota	0.0639	149			

Table B.15: A One-way ANOVA Statistical Analysis Test for the Mean DR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Primus Classification

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
Columns	28.7183	2	14.3591	1.4973e+04	1.3539e-170
Error	0.1410	147	9.5903e-04		
Tota	28.8593	149			

Table B.16: A One-way ANOVA Statistical Analysis Test for the Mean FPR for the Three ML Algorithms Using the Flow-based Feature Set on Training Data Set for Primus Classification

Source	**SS**	**df**	**MS**	**F**	**Prob>F**
textbfColumns	0.0165	2	0.0083	36.4825	1.3641e-13
Error	0.0333	147	2.2669e-04		
Tota	0.0499	149			

Appendix C

FPR for the Best Solutions for Each Classifier

C.1 FPR for Packet Header Feature Set for Skype Detection

C.2 FPR for Flow Feature Set for Skype Detection

Table C.1: Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Packet Header Feature Set on the Univ2007 Test Trace (FPR=0.03%)

Applications	**Value**
FTP	0.0000
SSH	0.0000
MAIL	0.0000
DNS	0.0000
HTTP	0.0000
HTTPS	0.0000
MSN	0.0000
OTHER	0.0003

Table C.2: Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Packet Header Feature Set on the Univ2010 Test Trace (FPR=2.4%)

Applications	**Value**
FTP	0.0000
SSH	0.0004
MAIL	0.0001
DNS	0.0001
HTTP	0.0032
HTTPS	0.0002
MSN	0.0001
P2P	0.0002
OTHER	0.0196

Table C.3: Applications Wrongly Classified as Skype by the GP-based Signatures on the Packet Header Feature Set on the Univ2007 Test Trace (FPR=2.5%)

Applications	**Value**
FTP	0.0000
SSH	0.0001
MAIL	0.0003
DNS	0.0012
HTTP	0.0032
HTTPS	0.0007
MSN	0.0006
OTHER	0.0193

Table C.4: Applications Wrongly Classified as Skype by the GP-based Signatures on the Packet Header Feature Set on the Univ2010 Test Trace (FPR=65.1%)

Applications	**Value**
FTP	0.0001
SSH	0.0041
MAIL	0.0195
DNS	0.0102
HTTP	0.2007
HTTPS	0.0181
MSN	0.0006
P2P	0.0024
OTHER	0.2663

Table C.5: Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Packet Header Feature Set on the Univ2007 Test Trace (FPR=0.11%)

Applications	**Value**
FTP	0.0000
SSH	0.0000
MAIL	0.0000
DNS	0.0000
HTTP	0.0002
HTTPS	0.0001
MSN	0.0000
OTHER	0.0007

Table C.6: Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Packet Header Feature set on the Univ2010 Test Trace (FPR=0.34%)

Applications	**Value**
FTP	0.0000
SSH	0.0001
MAIL	0.0003
DNS	0.0001
HTTP	0.0008
HTTPS	0.0001
MSN	0.0001
P2P	0.0000
OTHER	0.0018

Table C.7: Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=0.7%)

Applications	**Value**
FTP	0.0000
SSH	0.0000
MAIL	0.0000
DNS	0.0006
HTTP	0.0000
HTTPS	0.0000
MSN	0.0000
OTHER	0.0066

Table C.8: Applications Wrongly Classified as Skype by the C5.0-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=6.1%)

Applications	**Value**
FTP	0.0000
SSH	0.0000
MAIL	0.0002
DNS	0.0109
HTTP	0.0042
HTTPS	0.0008
MSN	0.0000
P2P	0.0032
OTHER	0.0340

Table C.9: Applications Wrongly Classified as Skype by the GP-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=9.9%)

Applications	**Value**
FTP	0.000
SSH	0.000
MAIL	0.000
DNS	0.013
HTTP	0.001
HTTPS	0.000
MSN	0.000
OTHER	0.085

Table C.10: Applications Wrongly Classified as Skype by the GP-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=9.3%)

Applications	**Value**
FTP	0.000
SSH	0.000
MAIL	0.000
DNS	0.028
HTTP	0.000
HTTPS	0.000
MSN	0.000
P2P	0.006
OTHER	0.059

Table C.11: Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Flow Feature Set on the Univ2007 Test Trace (FPR=4.3%)

Applications	**Value**
FTP	0.0000
SSH	0.0000
MAIL	0.0000
DNS	0.0009
HTTP	0.0001
HTTPS	0.0000
MSN	0.0000
OTHER	0.0423

Table C.12: Applications Wrongly Classified as Skype by the AdaBoost-based Signatures on the Flow Feature Set on the Univ2010 Test Trace (FPR=7.9%)

Applications	**Value**
FTP	0.0000
SSH	0.0000
MAIL	0.0000
DNS	0.0236
HTTP	0.0003
HTTPS	0.0000
MSN	0.0000
P2P	0.0053
OTHER	0.0499

Appendix D

Range for the Robust Signatures

D.1 C5.0 Signatures

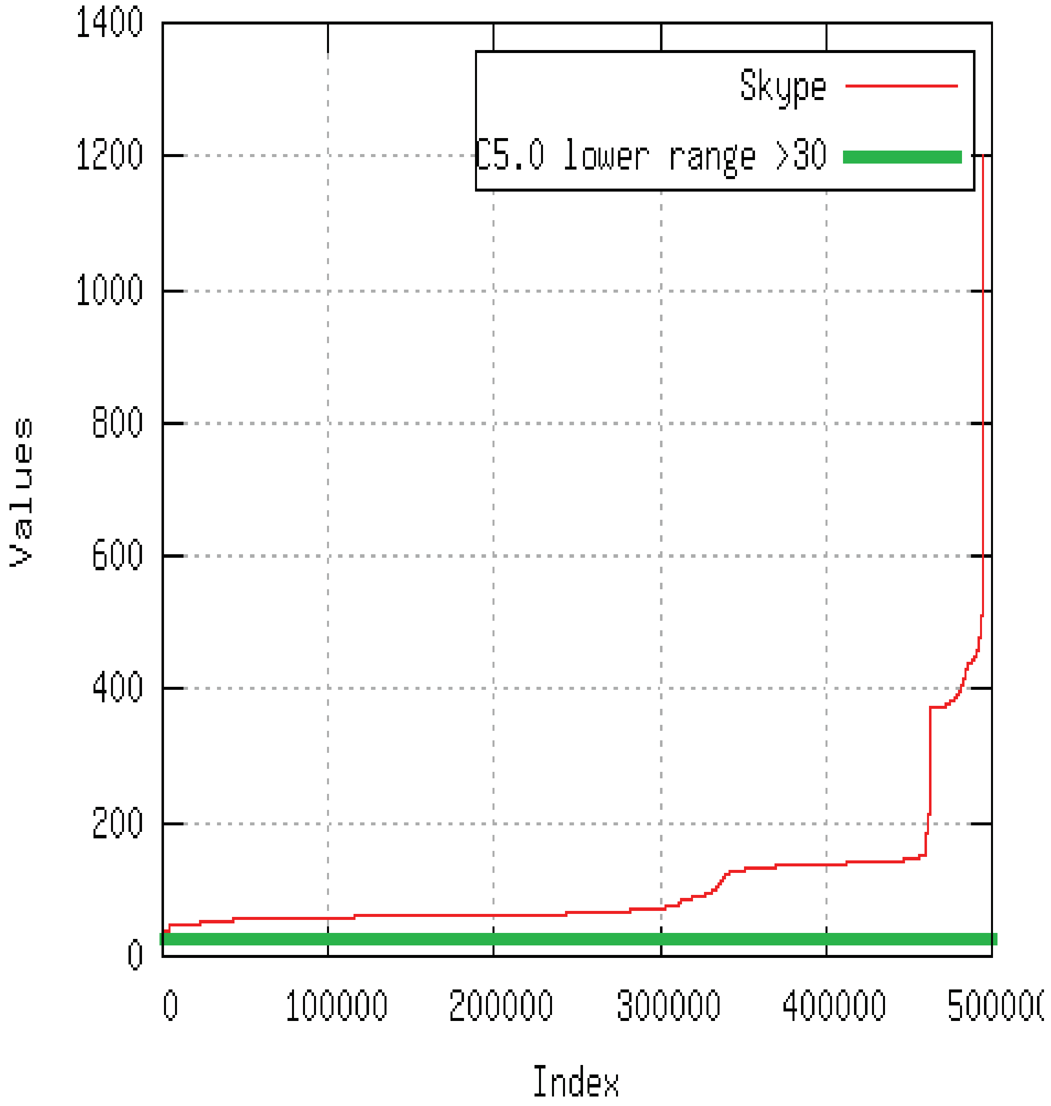


Figure D.1: Range for the min_fpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

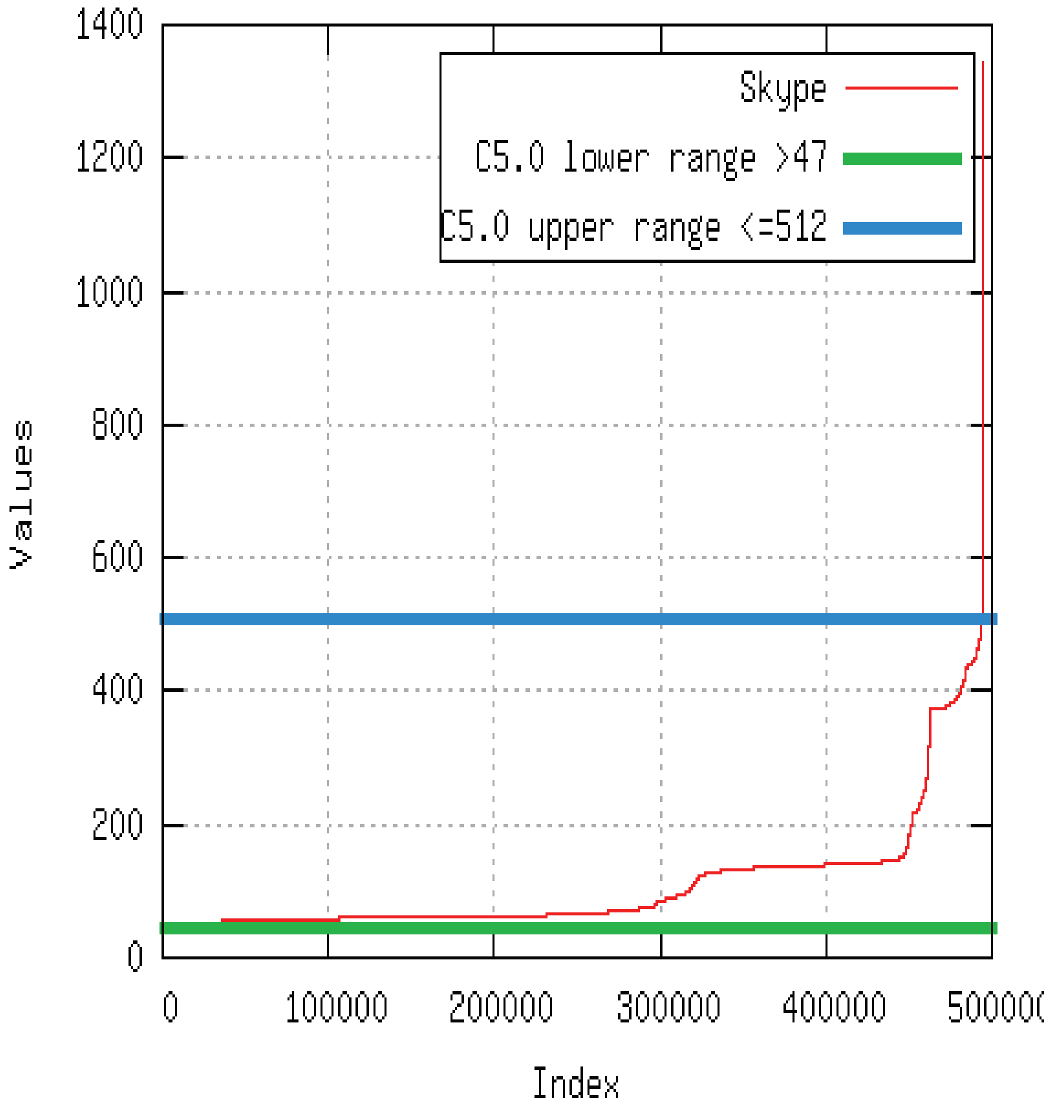


Figure D.2: Range for the mean_fpktl flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

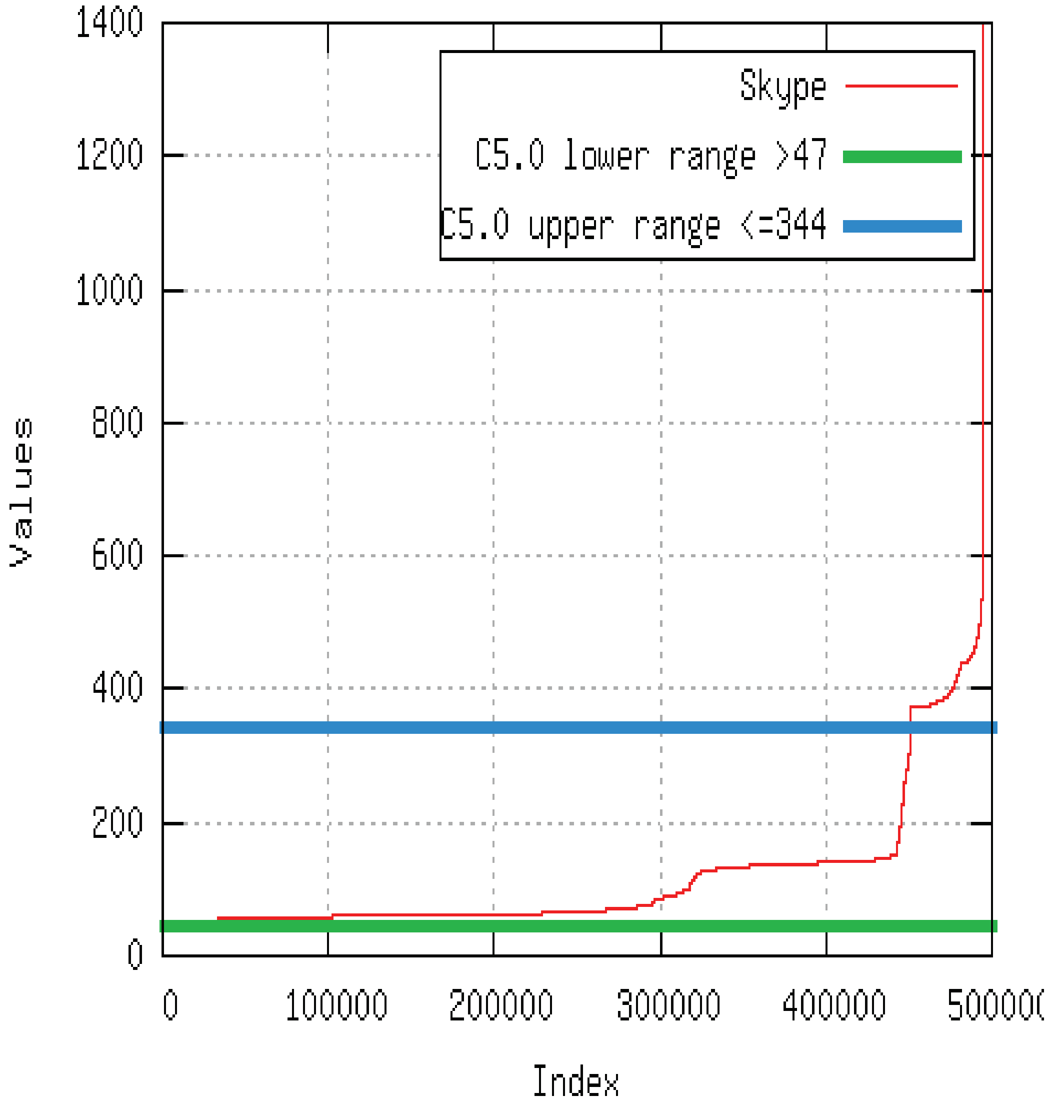


Figure D.3: Range for the max_fpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

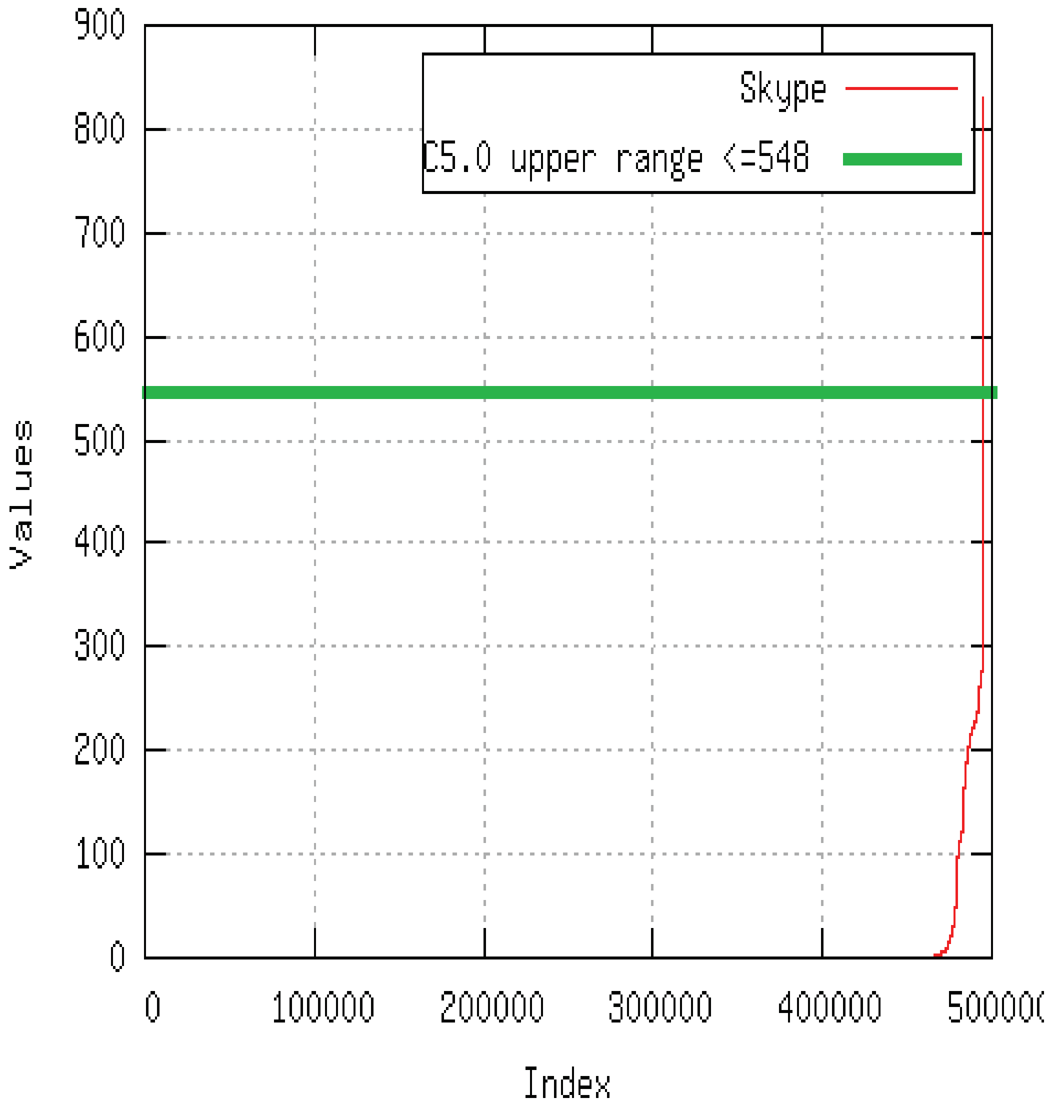


Figure D.4: Range for the std_fpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

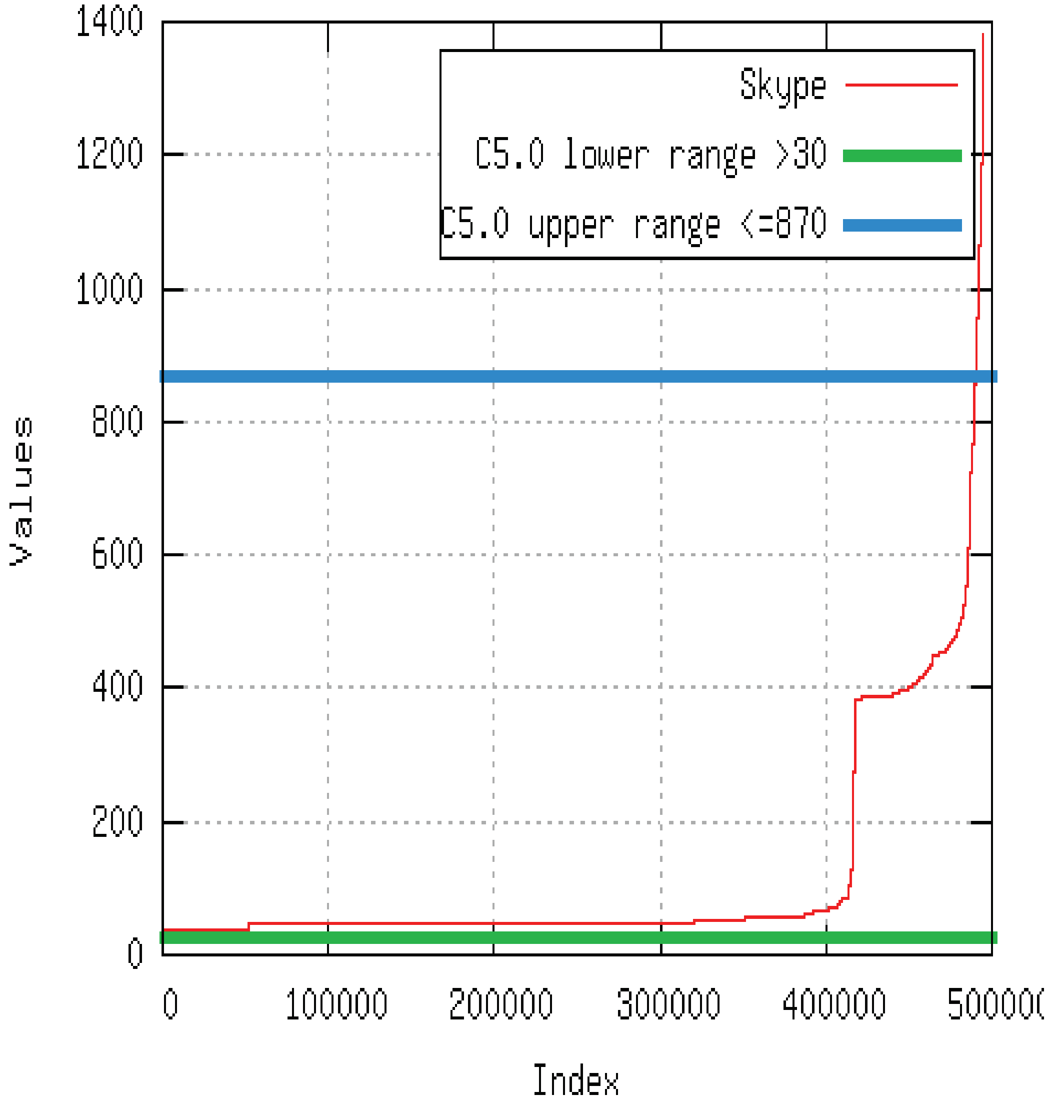


Figure D.5: Range for the min_bpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

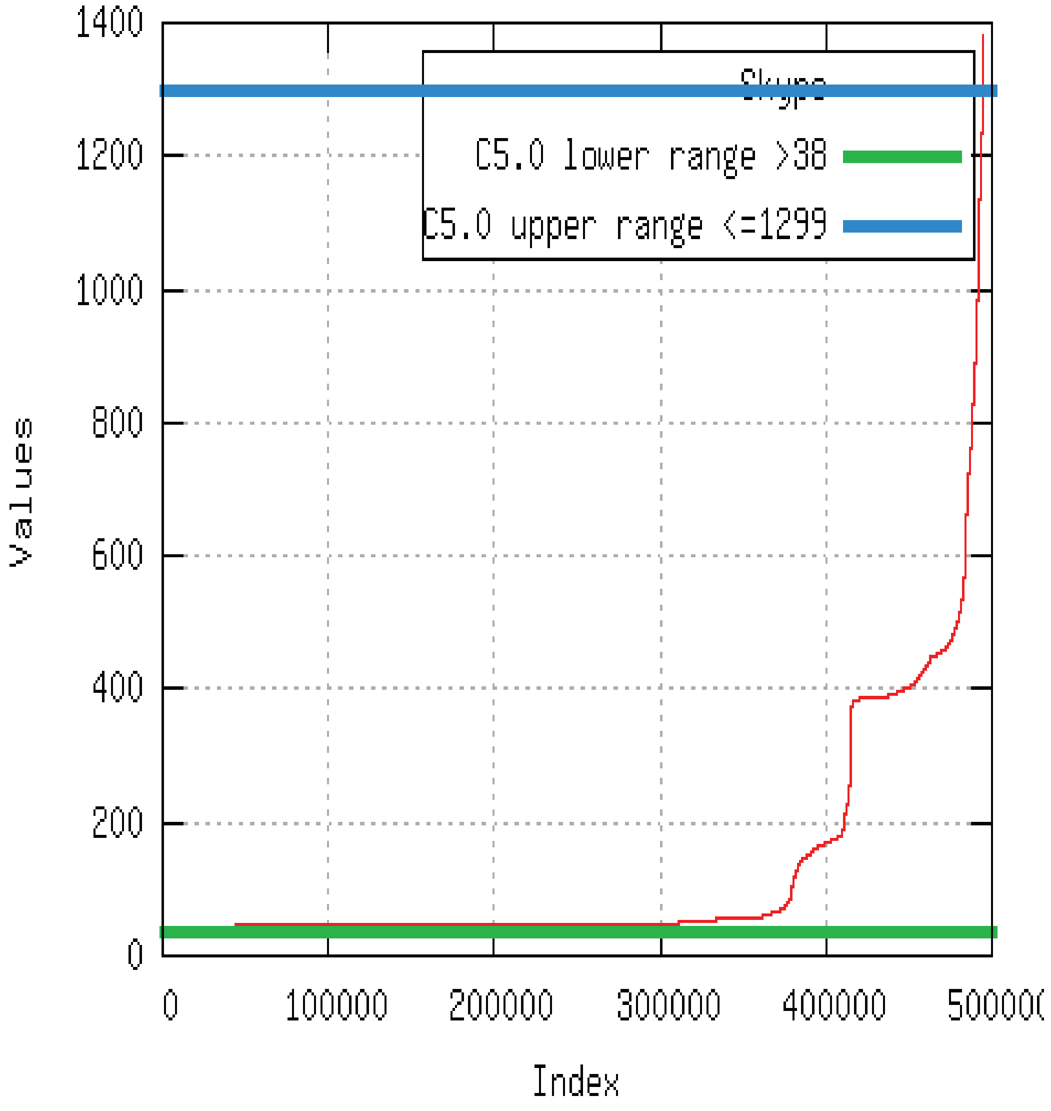


Figure D.6: Range for the mean_bpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

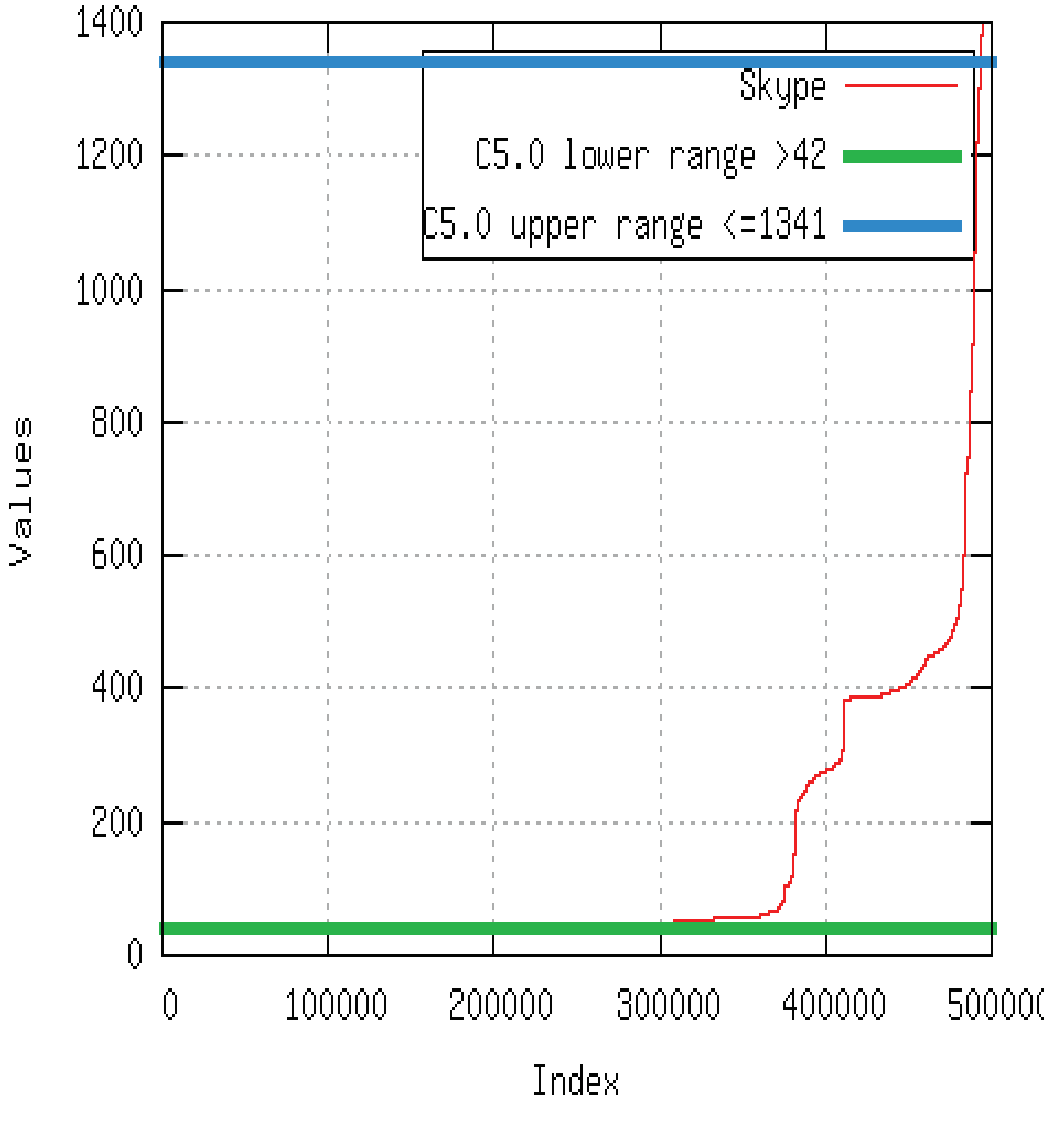


Figure D.7: Range for the max_bpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

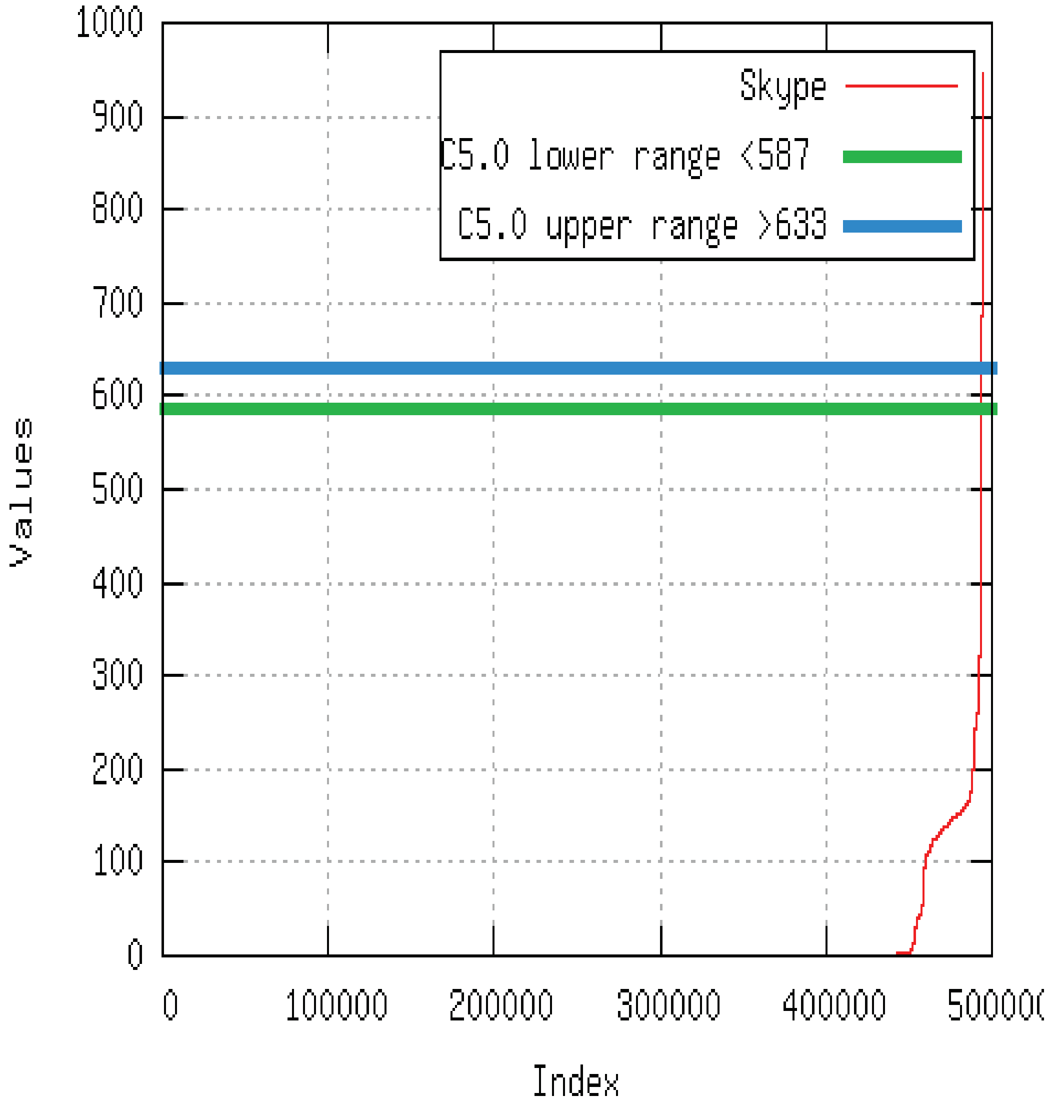


Figure D.8: Range for the std_bpktl Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

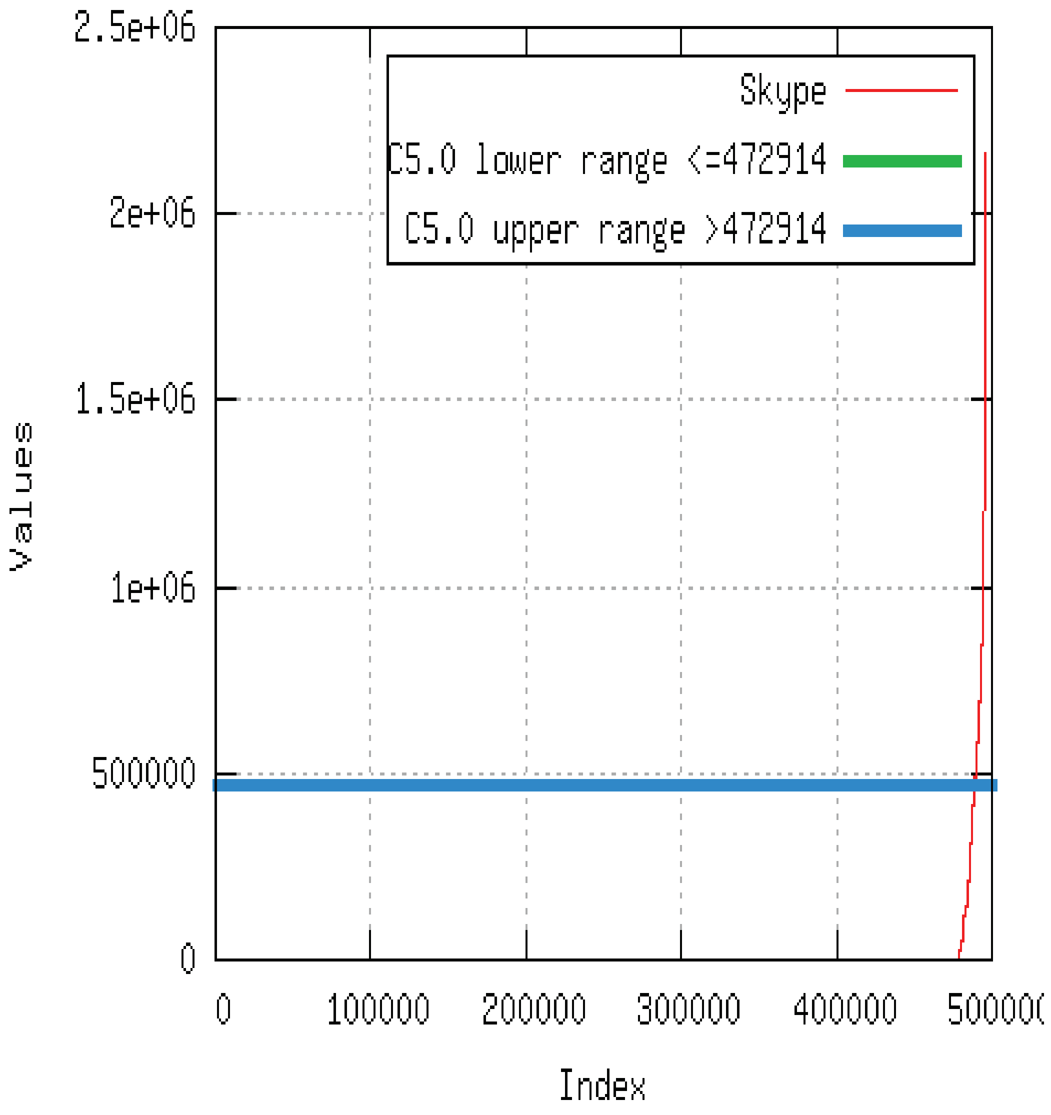


Figure D.9: Range for the min_fiat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

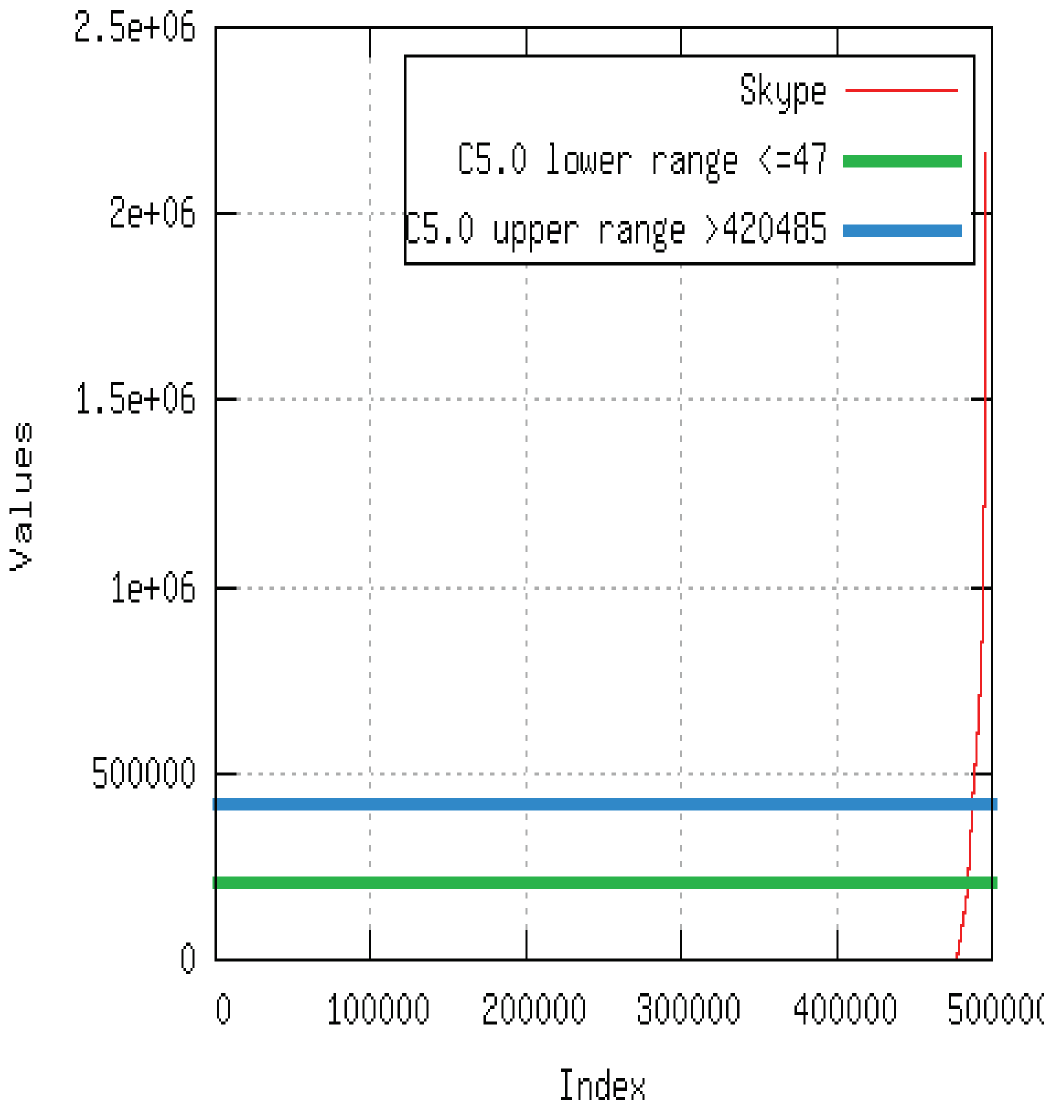


Figure D.10: Range for the mean_fiat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

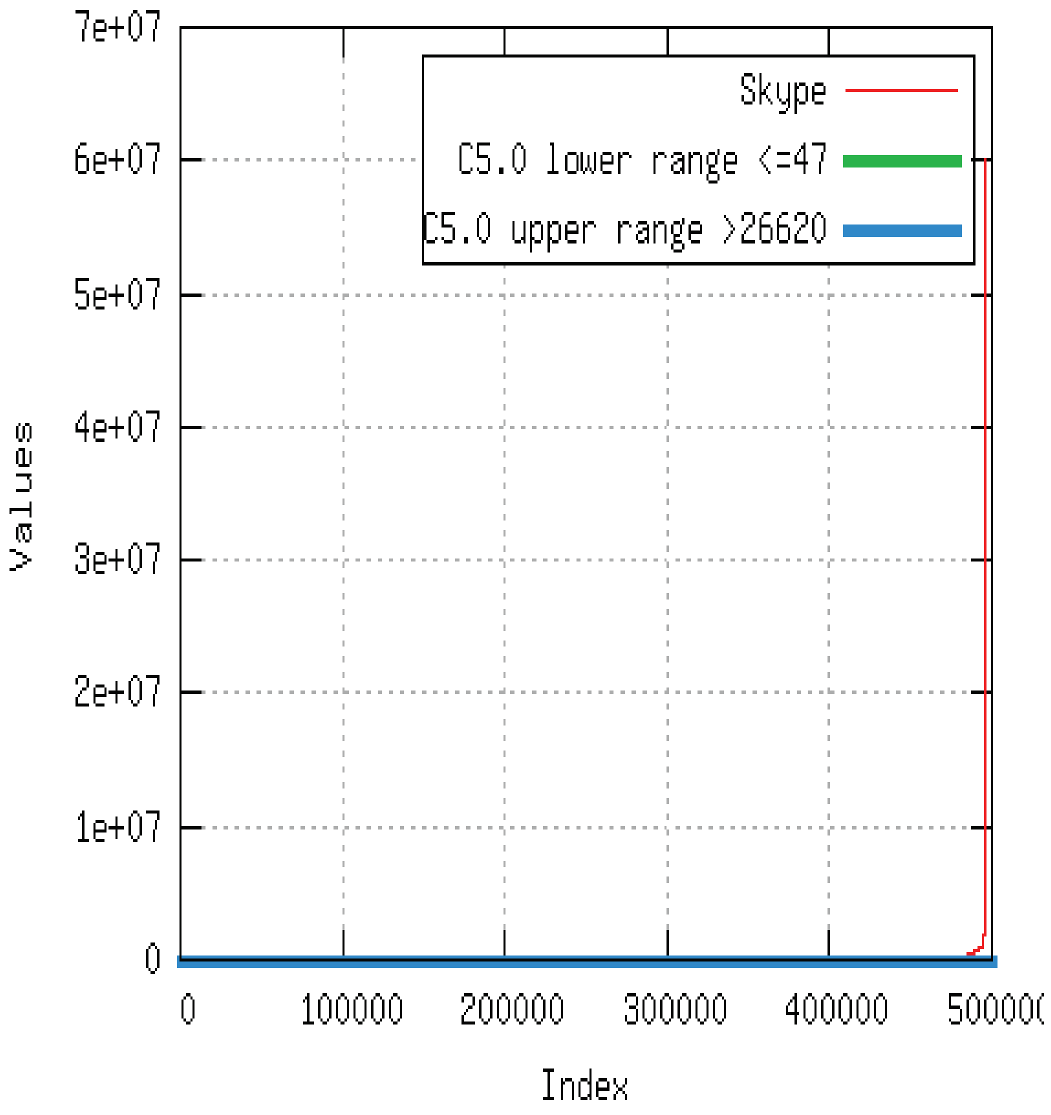


Figure D.11: Range for the max_fiat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

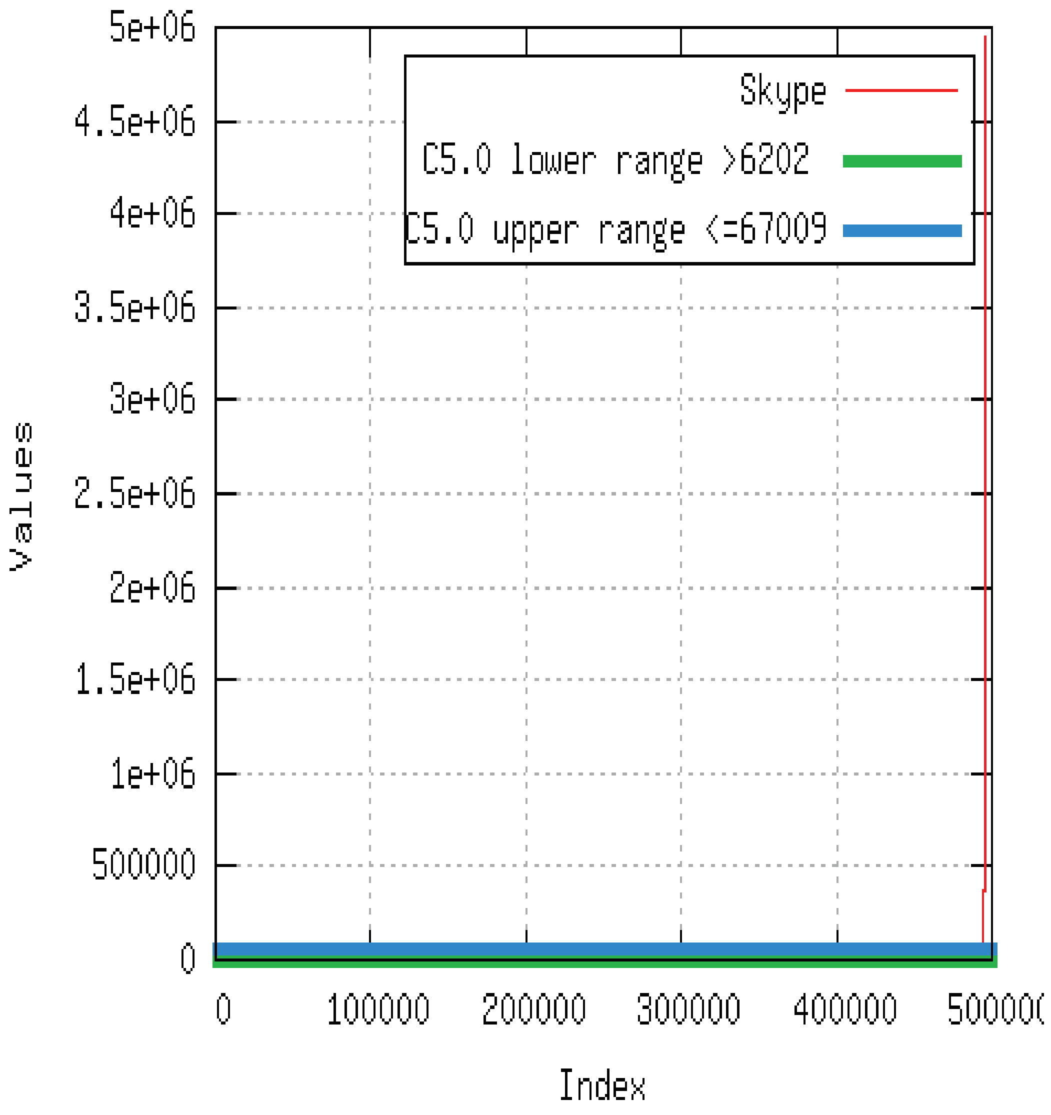


Figure D.12: Range for the std_fiat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

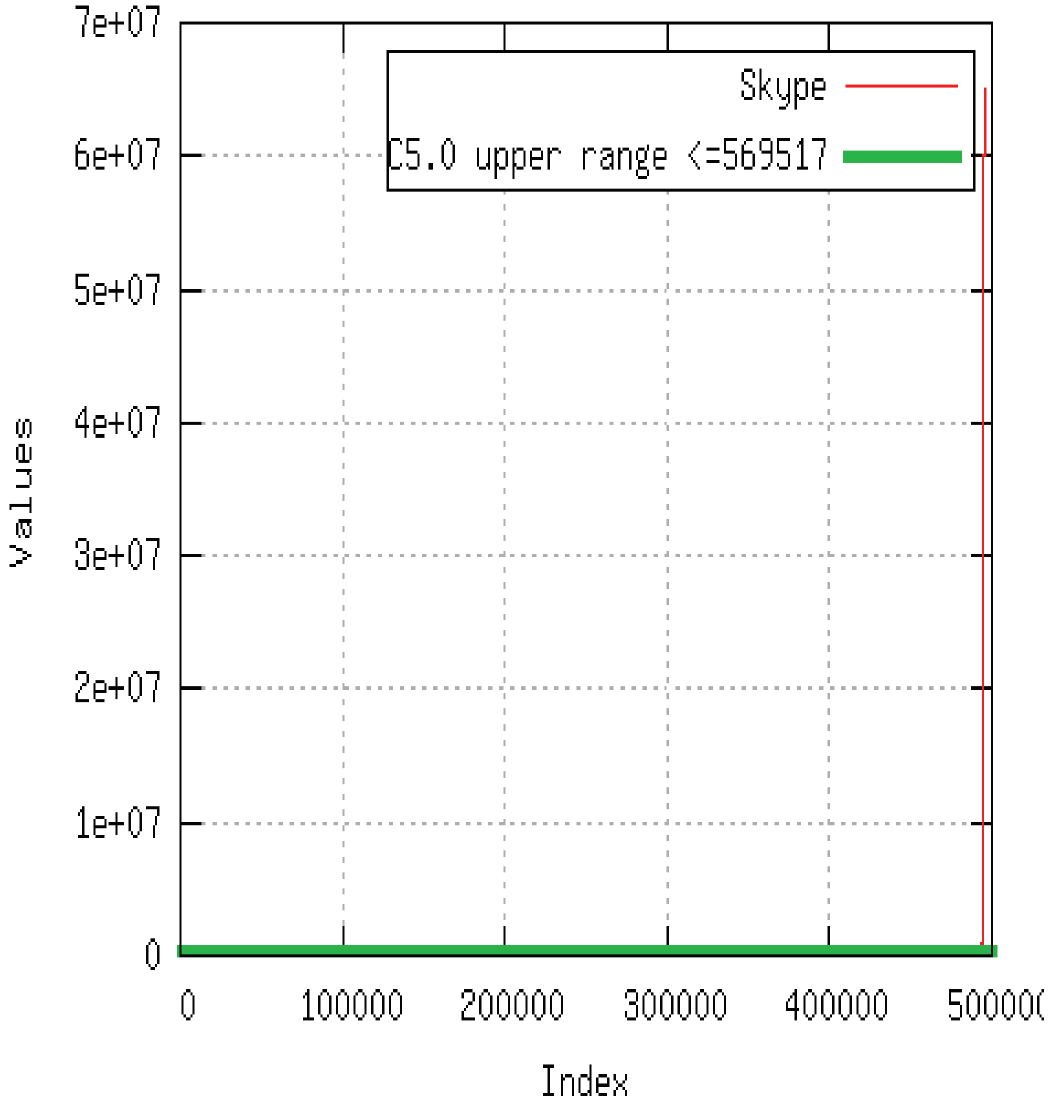


Figure D.13: Range for the min_biat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

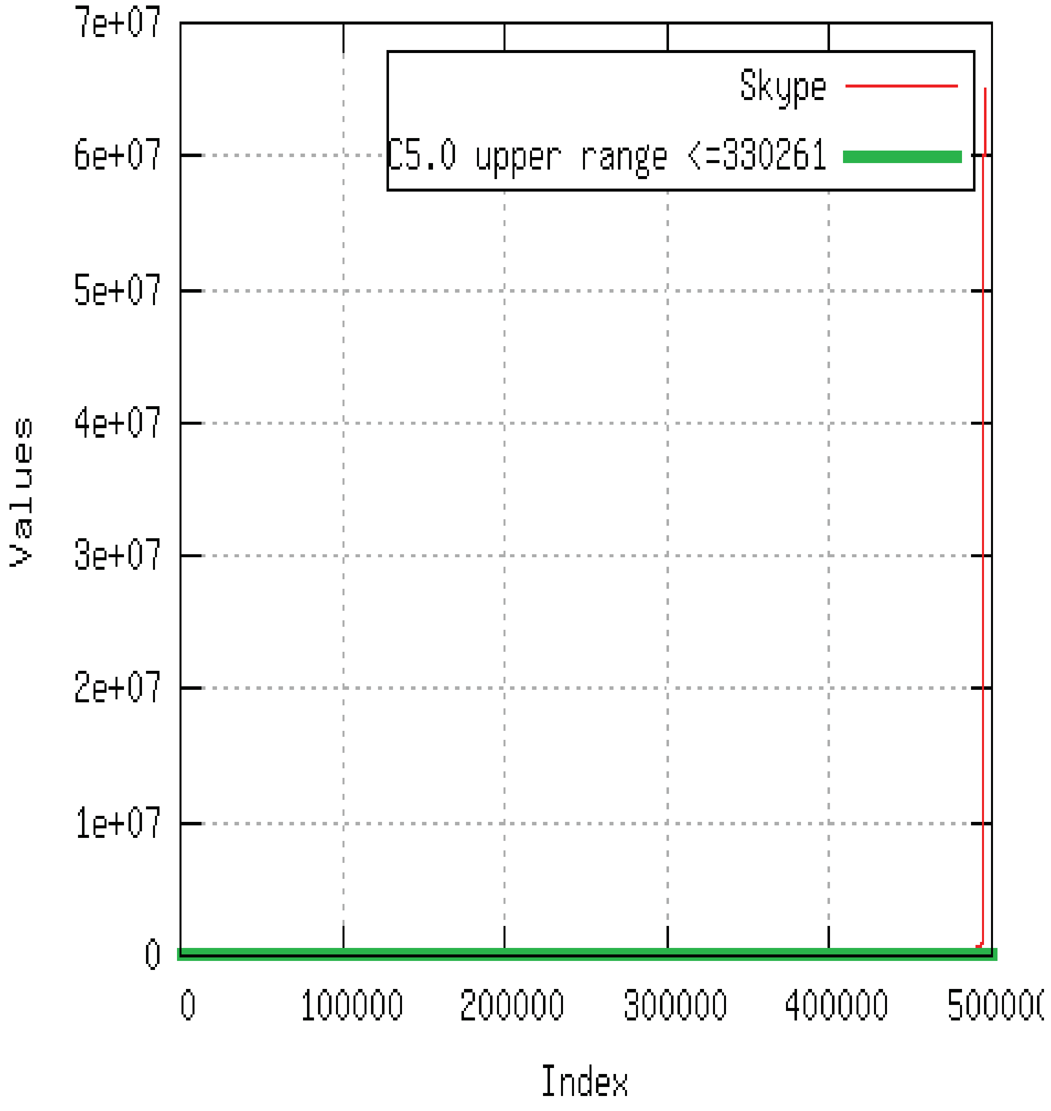


Figure D.14: Range for the mean_biat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

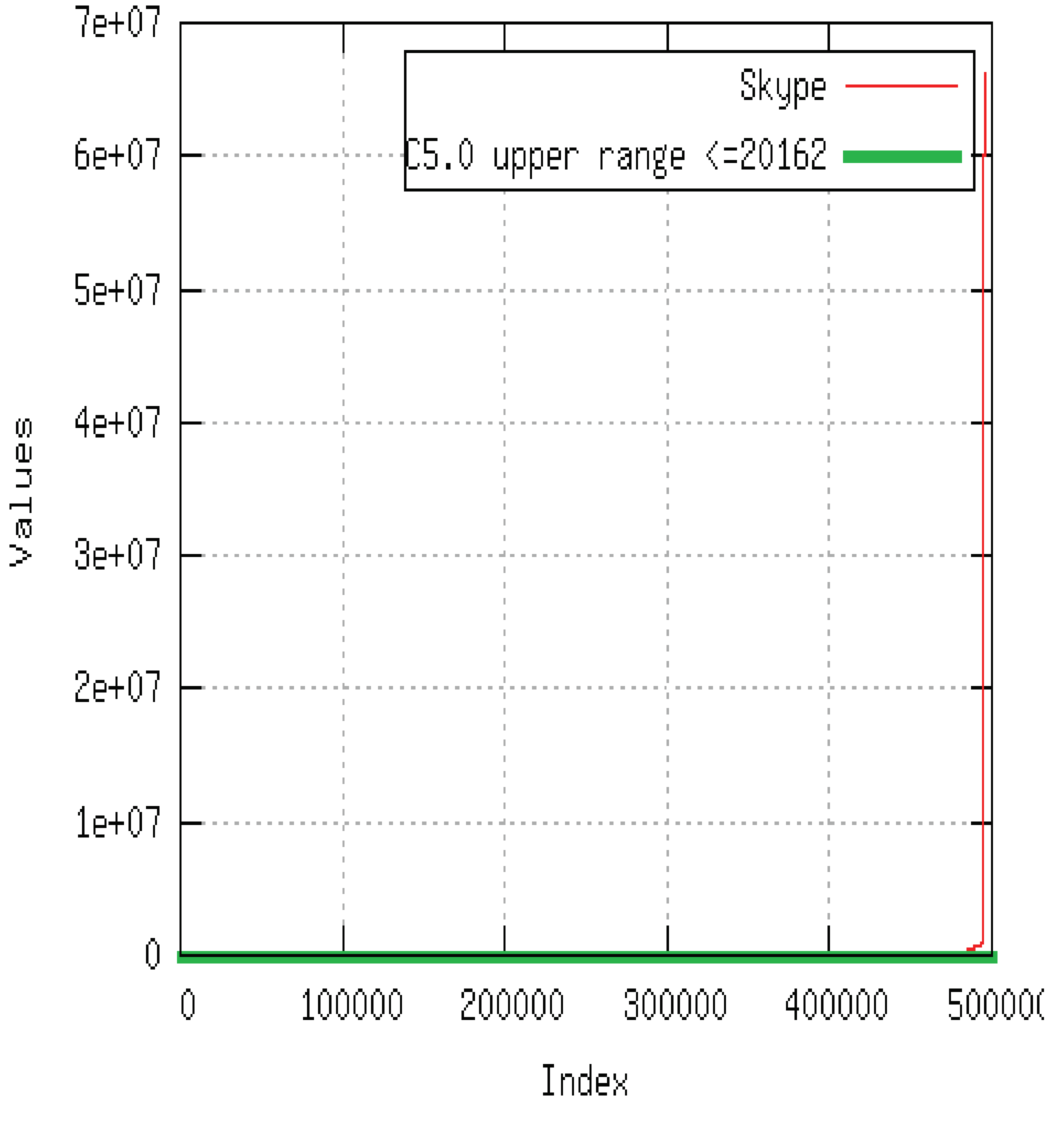


Figure D.15: Range for the max_biat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

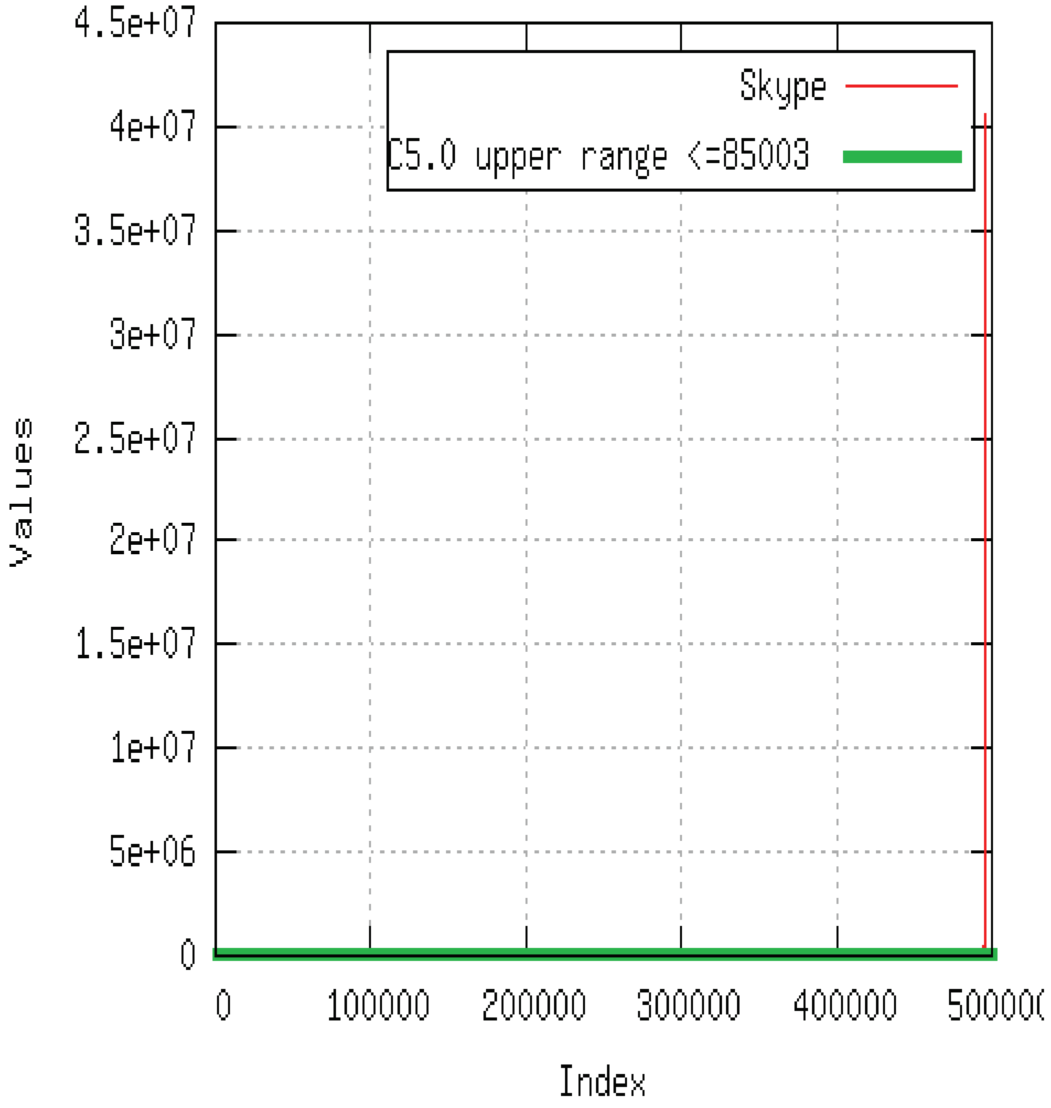


Figure D.16: Range for the std_biat Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

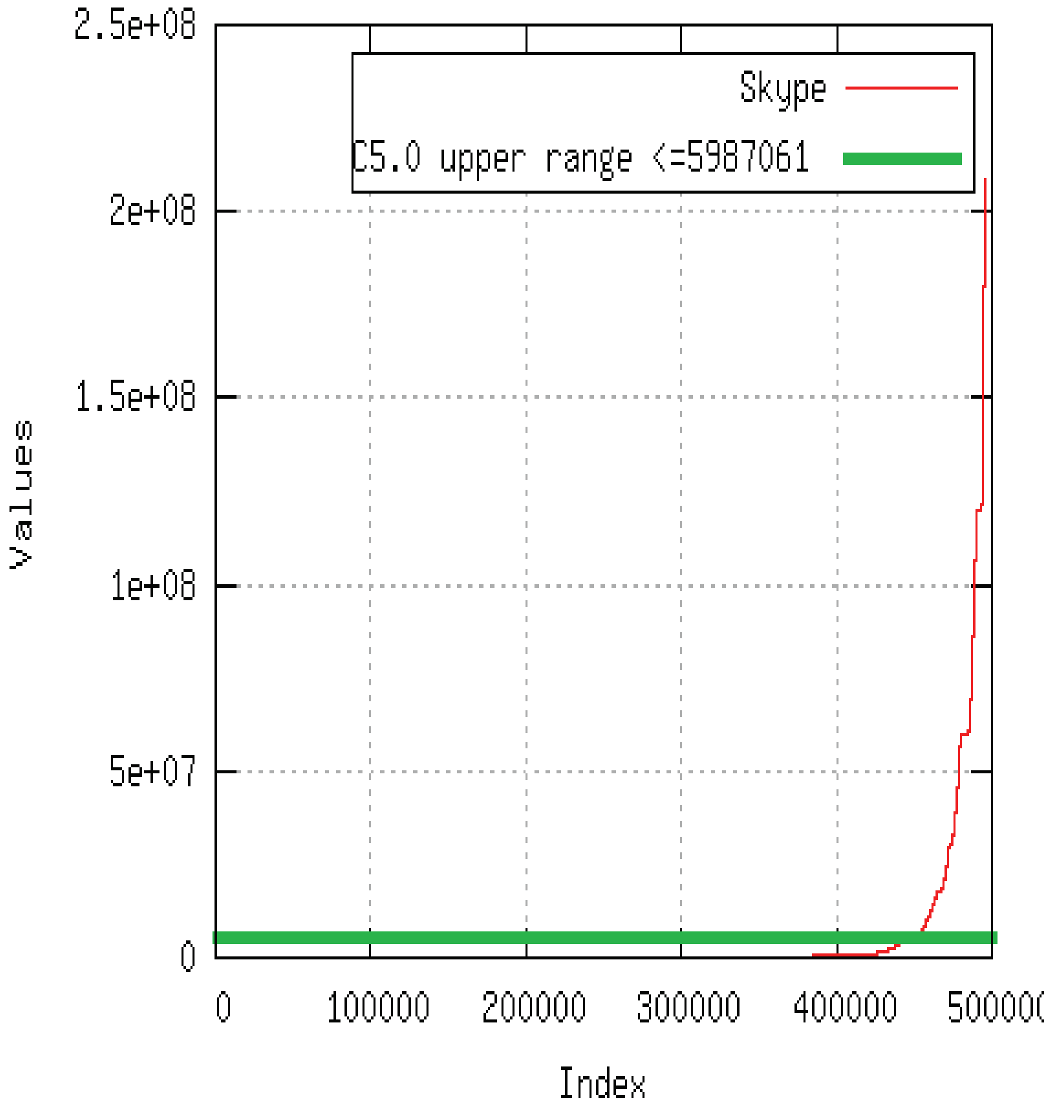


Figure D.17: Range for the Duration Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

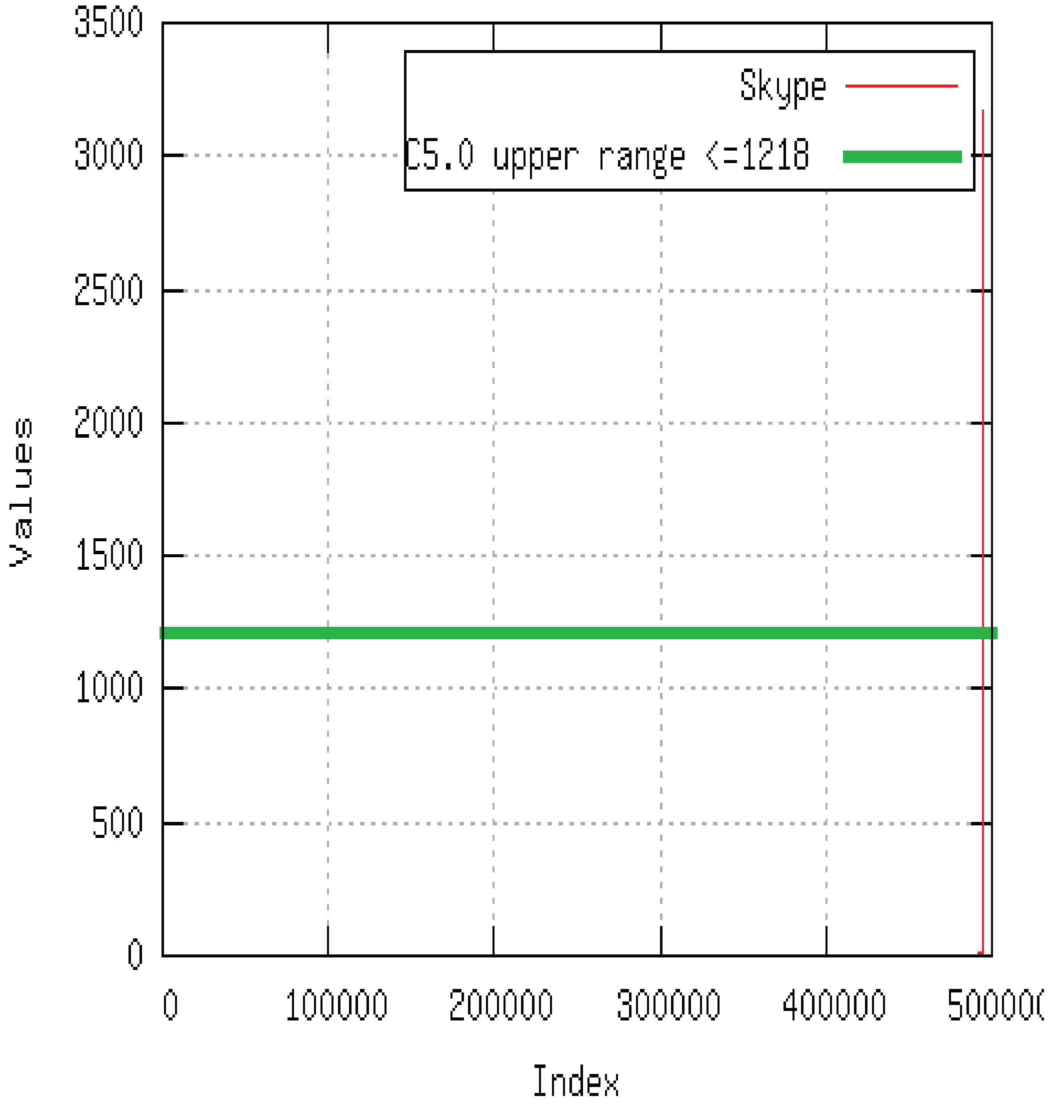


Figure D.18: Range for the Total fpackets Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

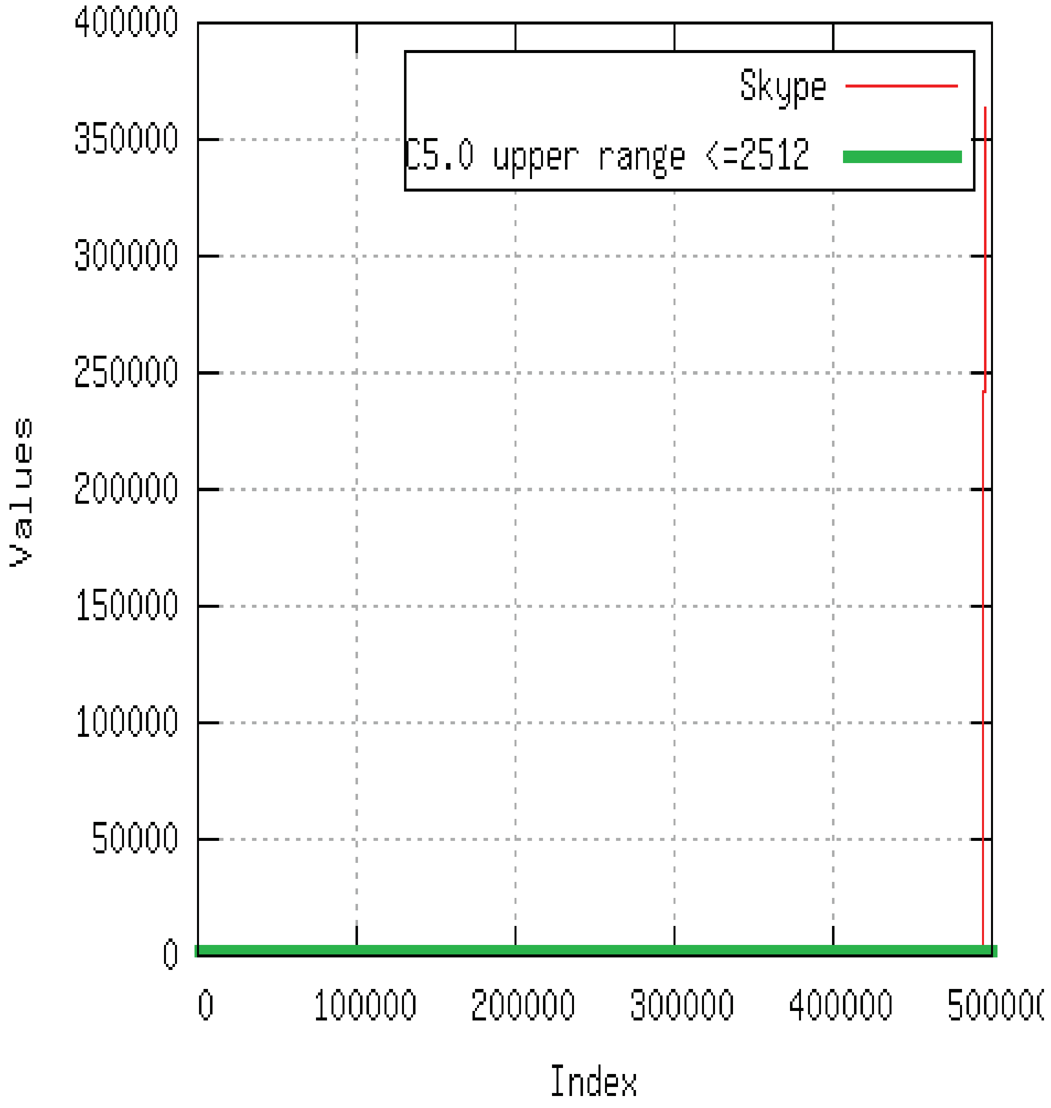


Figure D.19: Range for the Total fbytes Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

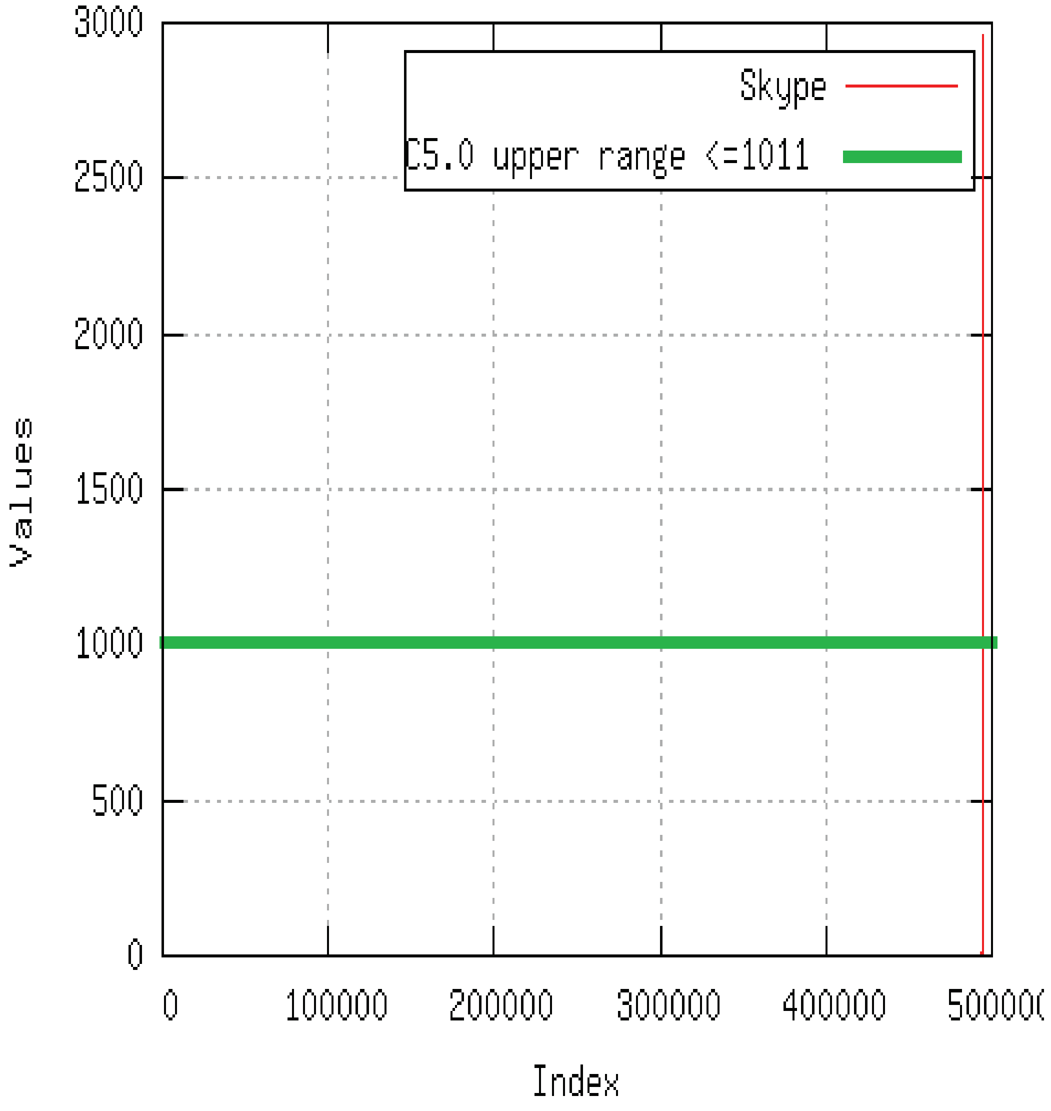


Figure D.20: Range for the Total bpackets Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set

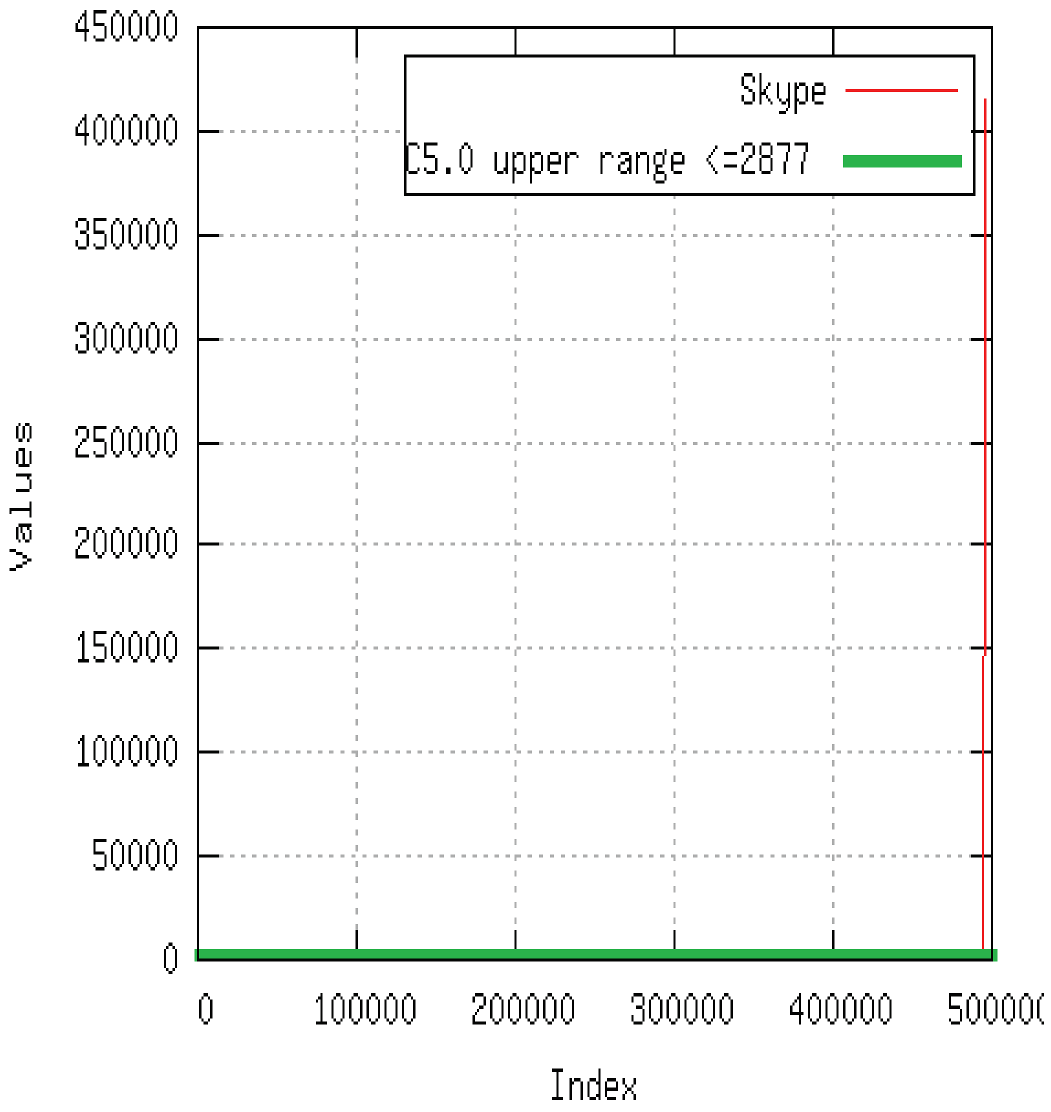


Figure D.21: Range for the Total bbytes Flow Feature for the C5.0 Robust Signatures Based on the Training Data Set