EXPLORING DATA LEAKAGE VIA SUPERVISED LEARNING

by

Amir Khaleghimoghaddam

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2020

I dedicate this thesis to my mother

Table of Contents

List of Tables

List of Figures

Abstract

Data security includes but not limited to, data encryption, tokenization, and key management practices that protect data across all applications and platforms. In this thesis, I aim to explore whether any data leakage takes place in data encryption when encrypted data is analyzed using supervised machine learning techniques. In the literature, researchers studied reverse engineering the encrypted data or brute forcing the attacks against encryption algorithms in order to study data leakage. However, in this research, my goal is not to reverse engineer or brute force the ciphertext, but to explore whether a supervised learning algorithm could identify a pattern that could potentially leak data in ciphertext. To this end, I analyze four encryption algorithms using five supervised learning techniques on four different datasets. The results show that as the encryption algorithms get stronger, the data leakage decreases, even though the data leakage is never zero percent.

List of abbreviations used

AI Artificial Intelligence

ML Machine Learning

NLP Natural Language Processing

TF-IDF Term Frequency – Inverse Document Frequency

CNN Convolutional Neural Networks

RNN Recurrent Neural Networks

dev development

LSTM Long Short-Term Memory

BOW Bag Of Words

KDES Keywords Detection and Expansion System

DES Data Encryption Standard

AES Advanced Encryption Standard

IDEA International Data Encryption Algorithm

RC Rivest's cipher

ECB Electronic Block Chaining

CBC Cipher Block Chaining

LPT Left Plain Text

RPT Right Plain Text

NIST National Institute of Standards and Technology

IV Initialization Vector

API Application Programming Interface

IT Information Technology

Acknowledgements

I would like to thank everyone who played a role during my master's research, specially my supervisor, Dr. Nur Zincir-Heywood who guided me throughout my research. Also, my committee members Dr. Srinivas Sampalli and Dr. Malcolm Heywood, each of whom has provided valuable advice to complete my thesis. Lastly, I would like to express my gratitude to my family, friends and NIMS Lab members who supported me during my research.

Chapter 1

Introduction

Nowadays, technology is evolving rapidly to make our lives easier and more efficient. As an outcome, there are all sorts of information, including daily messages, private information, security information, etc. are transferred through the internet.In order to secure this information from any threat, it is necessary to provide a way that only the intended recipients can read it. That is when encryption becomes useful. It is a technical term that ensures the data that is being sent to the internet is secure. In a nutshell, encryption is the process of encoding messages and/or information. The primary purpose of the encryption is to convert the actual text (plaintext) from the user to a ciphertext that only the receiver can read it. This process is usually done by using a key in the middle that only the sender and receiver have access to it. In this method, when a host sends a message to the client, it encrypts it with a private key, and when the message reaches the destination, the client can translate the encrypted message and revert it to the plaintext using its own private key. This key could be the same for both sender and receiver which is called Symmetric Encryption as it is illustrated in figure 1.1, or it can be different, which is called Asymmetric Encryption, as it is shown in figure 1.2. I will discuss these methods more in the methodology chapter. Also, there are different types of encryption developed for different reasons. In this thesis, I will cover four types of widely known symmetric encryption algorithms and do the experiments on them.

Moreover, Artificial Intelligence (AI) is another outcome of the recent developments, which is developing intelligence for machines to learn from the experience. A subset of AI is Machine Learning (ML), which is considered to be the most interesting concept in recent days. Using ML, a computer can be fed a huge amount of data, which is called a dataset, and the object is to learn and analyze them in an automated process. This process is highly dependent on the input data that the user provides to the machine learning algorithm. In other words, the more data we feed

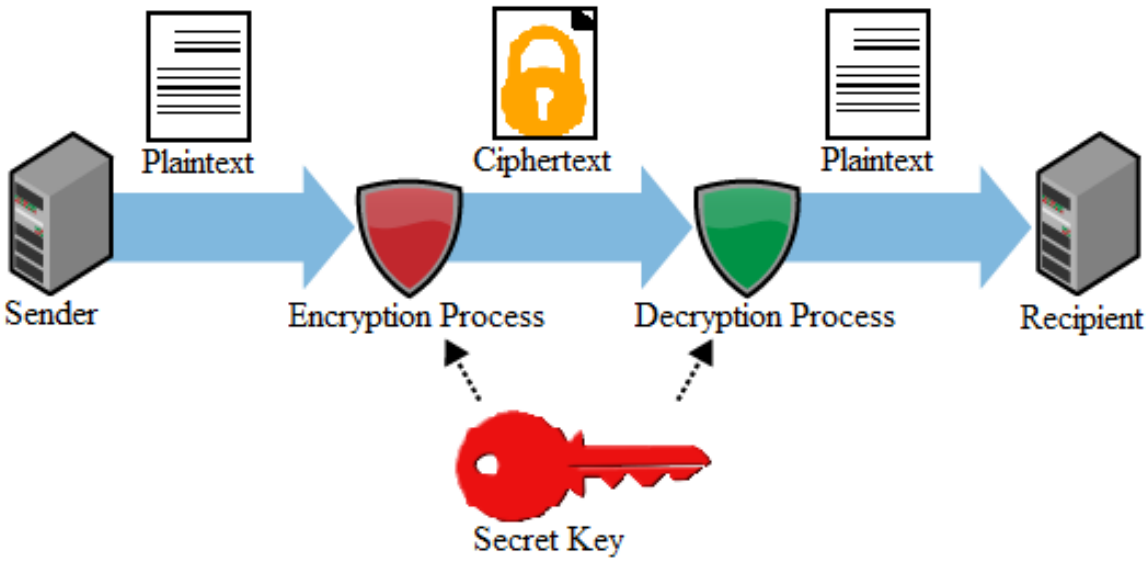


Figure (1.1) Symmetric Encryption process[36]

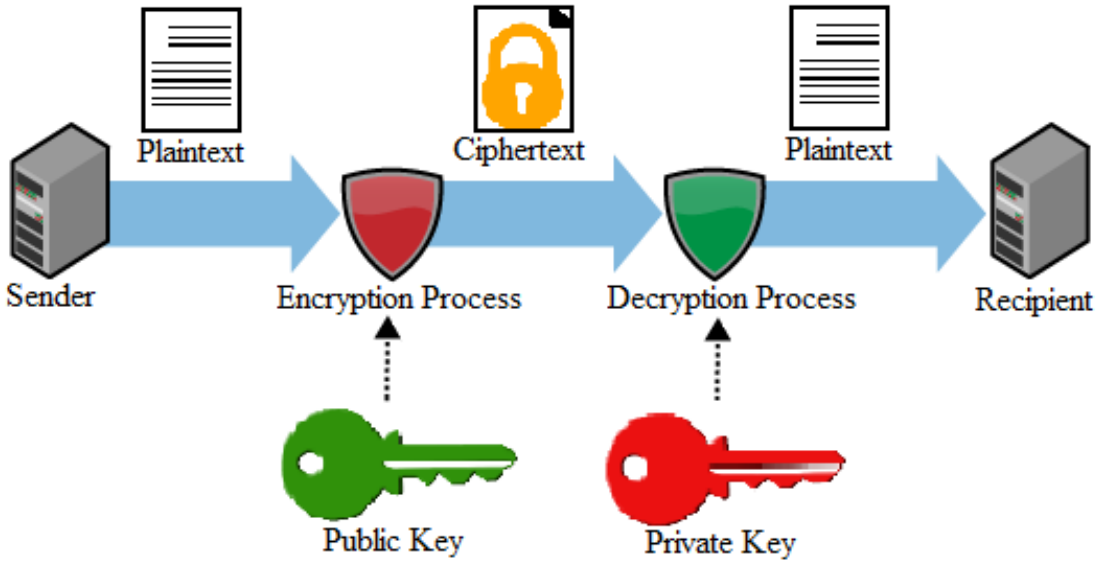


Figure (1.2) Asymmetric Encryption process[36]

to the algorithm, it can make the more computationally complicated decision. Hence the final solution is closer to reality. Machine Learning mainly contains two methods, which are Supervised Learning, Unsupervised Learning.

- Supervised Learning: Most of the work in the ML field is done via supervised learning. It maps data samples from the dataset to an output variable. In this case, the correct classification is already assigned to the training set. This process consists of two methods, called regression and classification. When the output of the classification is categorical, it is classification, and if the output is a continuous value, it will be a regression. [39]

- Unsupervised learning: In this case, users are not aware of the output class, and only input data is provided. The main purpose of this method is to find a structure or a pattern in the data that could lead to reasonable assumptions or results. [39]

My research objectives in this thesis is to explore whether any data leakage takes place in data encryption when encrypted data is analyzed using supervised machine learning techniques. The goal here is not to reverse engineer the encryption but to investigate whether there are any patterns in the encrypted data that could be identified using supervised learning techniques to analyze the encrypted data. For this purpose, I study if the topic of an encrypted text could be identified or not. Such an analysis could be useful for not only to explore data leakage in encrypted data but also could be useful for forensic analysis for cybersecurity purposes.

In this thesis, I will discuss the proposed system, which can learn from the encrypted data without any previous knowledge to identify the related topics of each ciphertext. This process is done by implementing several supervised learning algorithms on different sets of data using different types of encryption.

The rest of this thesis is organized as follows. Related work is discussed in Chapter 2. The classification algorithms, Datasets, Encryption algorithms, the approach taken and the tools that I used to implement the model are discussed in Chapter3. Evaluations and results are presented in Chapter 4. Finally, conclusions are drawn and future works are discussd in Chapter 5.

Chapter 2

Literature Review

Due to the complexity of privacy risks, analysis of services that provide security is now more essential than ever. Also, machine learning classification is widely used in numerous applications for text classification such as medical, finance, speech recognition, etc. Since privacy is a concern, the data need to remain confidential while transferring. Thus, the algorithm used to provide this confidentiality should be clear of any leakage. In this chapter, related works on encrypted text classification are covered. To this end, most of the work that has been done already, focus on preserving privacy during the training. Some of these works are investigated in the following.

2.1 Classification of plaintext data

Text classification is the task of assigning one or more labels to a text. This process makes it easier to sort. Natural Language Processing helps to find the essential keywords through this process so the machine learning algorithm can understand the text data. Here I am using four different datasets, and I will discuss the papers published on these datasets and their approach towards text classification in the following. Moreover, I will cover some of the techniques that can be used in order to make the raw text data understandable to machine learning algorithms and discuss some of the related works that have been done in that area.

2.1.1 Datasets

There are not many publicly available datasets that have sufficient records and can be used for classifying the raw text data. During my research, I found particularly four datasets that will be useful for this task. There are some papers published on these datasets, and most of them are focusing on data pre-processing and Natural Language processing techniques to achieve a high score on the classification task. These datasets are Kaggle Isis dataset [5], Kaggle News Category dataset [6], Kaggle Sentiment 140

dataset [7] and AG news dataset [2]. In 2018, Fernandez et al. [21] experimented on the Kaggle Isis dataset and used contextual meanings for better discrimination of radical content. For this purpose, they built classifiers to process contextual semantics, and in order to distinguish the terms that are related to radicalized rhetoric, have found terms for both target classes (Pro-ISIS and Non-Pro-ISIS). 15.7 percent of the terms belong to Pro-ISIS tweets, 5.6 percent to Non-Pro-ISIS, 47.9 Percent for Pro-ISIS and Non-Pro-Isis, and 30.8 Percent of the terms did not appear in any of the two classes. They used two different classifiers, one is unigrams only classifier, and another one is semantically enhanced classifier. The goal of making these two classifiers is to see if the semantics could improve the already existing methods by giving more information. They made a dataset with 27,400 tweets with each class containing 13,700 tweets. Using this approach, they were able to achieve the F1 score of 0.851.

In another study, Olga Fuks [22] experimented on Kaggle News Category dataset with a combination of TF-IDF method and word embeddings to predict the category of the news from their short descriptions and their headlines. She used both traditional ML and deep learning methods. For their word embedding, they used a Keras layer and considered only 30,000 most common words in the dataset. Moreover, truncated each example to a maximum length of 50 words. She used four different classifiers for the traditional ML method, Naive Bayes, Multinomial Logistic Regression, Kernel SVM, and Random Forest. Also, CNN and RNN were used for the deep ML method. They used pre-trained Glove word embeddings [26] with 100 dimensions for the part of the neural networks. After pre-processing, she used 113,342 samples with 25 labels. She divided the data into three parts of the train, dev, and test. The best result belonged to the ensemble of four neural network models, with a 68.85 percent accuracy score.

Moreover, Yogatama et al. [17] compared the complexity and error rate of discriminative and generative LSTM-based text classification models. In addition to their generative and discriminative datasets, they used Naive Bayes classifier, Kneser–Ney Bayes classifier, and Naive Bayes neural network. In their experiments, they set the word embedding dimension and the LSTM hidden dimension to 100. They discovered that discriminative models are faster in training and inference time. On the other

hand generativeModels are using the vocabulary of hundreds of thousand words, which makes them computationally intensive. They experimented on AG news, Sogou, Yelp Bin, Yelp Full, DBPed, and Yahoo datasets. Since I used the AG News dataset further on in my research, I am going to focus on the results based on this particular dataset. They divided this dataset into three parts, 115,000 records for the training set, 5,000 records for dev set, and 7,600 records for the test set. The highest accuracy score, which they achieved, was 92.5 percent using fasttext embeddings [10].They also tried zero-shot learning as another part of their experiments. In addition, Go et. al [9], illustrated that using emoticons as noisy labels in the training data is an effective way to improve machine learning algorithms. In their approach, they intend to classify the sentiment of Twitter messages using emoticons automatically. Their machine learning classifiers are Naive Bayes, Maximum Entropy (MaxEnt), and support vector machines (SVM). Moreover, their feature extractors are uni-grams, bigrams, unigrams and bigrams, and unigrams with part of speech tags. The side effect of their approach is that they need to strip out the emoticons while learning the classifier. After pre-processing the data, they ended up having 1.6 million tweets with 800,000 tweets in each class. They used word N-grams for their feature extraction. Their best result belongs to a combination of Unigram and Bigram, where the highest accuracy score is 83.0 percent.

2.1.2 Feature extraction

Since most machine learning algorithms cannot take in straight text, a numerical matrix should be created to represent the text. There are several approaches about how to extract features from a text document, which in this section, I will briefly go over the papers and researches that used these approaches. One of the most popular methods for feature extraction is using a bag of words representation. In 2017, James Barry [12], conducted research and a compared bag of words and neural networks-based approaches for sentiment classification. He used an LSTM model [44] for neural networks part, which is able to handle sequential data as well as pre-trained GloVe embeddings [26] and Word2vec embeddings [48]. He found that although neural networks outperform bag of words method, a bag of words still performs very well, and it has considerably shorter training time periods. Another research conducted by

Bijoyan Das et al. [13], which performed sentiment analysis on their datasets Using the bag of words model and Term Frequency–Inverse Document Frequency (TF-IDF) model. Also, they proposed a model that consists of a combination of TF-IDF and Word Negation. They used ten fold cross-validation and extracted 5000 features from their text corpus. Their results indicate that TF-IDF model and, in particular, the combination of TF-IDF and word negation will improve the accuracy score by a good percentage.

2.1.3 Classification algorithms

In supervised learning, a classification algorithm weighs the input features and assigns a label to them. In this section, I will discuss some researches that used various types of classification algorithms to classify the text data. In further, I will implement these classifiers on my approach. In 2018, Shou [53], investigates the performance of Naive Bayes classifier for his work. Their study proposes three Bayesian classifiers, which are Multinomial, Bernoulli, and Gaussian event models. They used two different datasets for their model $20newsgroup$ and $WebKB$. They implemented parameter tuning for each of the classifiers in their research using grid search and reported their results with Precision, Recall, and F1 score. At the end of their experiments, they have found out that the NB classifier with a multinomial event model outweighs the Bernoulli model, which Bernoulli itself outweighs the Gaussian event model. In the same year, Umnity et. al [50], conducted research and attempted to classify the reports based on complaints and non-complaint using the Logistic Regression classifier. They gathered their data from The people's online aspiration and complaints services (LAPOR!), which is an Indonesian governmental website, and they labeled the dataset themselves. After that, they performed pre-processing and used TF-IDF and CountVectorizer algorithms to extract features from the text. In the next step, they modeled the data using training data and classified the test data. In the final step, they measure the performance using precision, recall, and F1 score. Lastly, they were able to achieve a better result on Logistic regression using Countvectorizer than TF-IDF feature extraction. In 2017, Vora et al. [37], proposed a model to classify tweets based on the emotions expressed by their authors. They used word vectors to convert text into numbers that could be classified with machine learning algorithms. They gathered

data from their twitter API and automatically assigned labels to their dataset by using Hashtags. They have labeled their dataset into four different categories, Happy, Sad, Angry, and Surprise. They implemented NLP methods in order to pre-process their dataset and replaced the emojis and emoticons with a related text. They trained their model using Word2vec, Glove and Fasttext word embeddings with different dimensions and different number of estimators. Finally, they chose the Random Forest classifier to perform classification on their model. Also, out of 160138 samples,112096 samples as training data, and 48042 samples as test data. Based on their results, the Fasttext model with 300-dimension and 200 estimators outperforms the other models. In 2012, Deshmukh et al. [41], proposed an SVM approach for emotion recognition from text. They created their own dataset by extracting news headlines from major newspapers such as the York Times, CNN, and BBC News, as well as from the Google News search engine. They used emotions, namely, Anger, Disgust, Fear, Joy, Sad, and Surprise, as the labels for their dataset. They used six annotators to label their dataset (one for each emotion). They ended up using two different datasets, one development dataset which is consisted of 250 annotated headlines and a test dataset consisting of 1,000 annotated headlines. They used SVM for their classification and trained it with 250 news headlines, also evaluated it using 1,000 newspaper headlines. In the end, they compared their model by three similar existing models SWAT [32], UPAR7 [15] and UA [29] . They Illustrated that their SVM based classification on their data can be as good as the other three systems.

2.2 Encrypted text classification

During my research, I have found that there is not much research that has been done in this field yet. Most of the researches that try to find leakage in an encryption algorithm are trying to decrypt the encryption algorithm which is not possible yet and takes a huge amount of computational resource or trying to imply techniques like counting the frequency which is not useful for most of the modern and complicated encryption algorithms. I will go over some of the best works that I have found related to this task. In 2001, Matsui [34], conducted a research about linear cryptanalysis method for DES cipher. Using this method they were able to implement the first known-plaintext attack of the full 16 round DES cipher. They had to deal with various

problems resulting from non-randomness of the plaintext. They were successfuly able to attack the ciphers with a success rate of 99.9 percent. Yang et al. [23], published a paper in 2017 investigating black keywords that are being used frequently by the underground community to help them escape from being tracked by law enforcement. Since these are black keywords, there seems to be no specific pattern that they can recognize them, and that is why they call it *Klingon* (a constructed language in the fictional Star Trek universe, without reference to any dictionary). In order to find these keywords, they developed a KDES (Keywords Detection and Expansion System), which is trained with 478,879 black keywords, and they were able to achieve the accuracy of 94.3 percent on detecting the black keywords with it. They used the Baidu search engine and collaborated with them to gather their experiments. This collaboration led them to scan their indexed pages using their spider pool detection system from Aug 25th to Sep 10th, 2016, which in total yielded 2,733,728 SEO pages. Overall, they gathered 63,424 pages marked as "evil" by Baidu, including 60,000 porn pages and 3,424 gambling pages. The term "Evil" means that they are related to sex, gambling, dangerous goods, surrogacy, drug, faked sites and etc. They extracted black keywords from pages and removed the duplicates. They filtered the keywords and then expanded them to include more possible and related black keywords. In the next step, they developed a method to identify core words and clusters similar to the black keywords. Since they did not have any ground truth, they had to review the detected keywords manually, so they sampled 1,000 keywords randomly, and in the end, 943 keywords were classified as black correctly (94.3 percent accuracy). In 2006, Mason et al. [28], conducted a research to explore the keystream reuse which allows a practical attack when data being encrypted. In this case, keystream means a key that stream ciphers use to encrypt the data. It is a well-known fact that the keystream should not be used again because if keystream k is reused to encrypt two different plaintexts p and q then the $p \oplus k$ and $q \oplus k$ can be XORed together to recover $p \oplus q$ which is called the two-time pad problem. A feasible solution is that the string pair (p, q) such that $p \oplus q = x$. In their method they considered the probability distribution of p and q as independent. They tried to recover the plaintext using their smoothed n-gram Language model. They took $n = 7$ as their n-gram for plaintext string and estimated probabilities for p and q. They used Finite-State Language models and

Cross product of language models to build their method and had an investigation on feasible solutions. In their experiments, they found out that if the keystream is used more than twice, their method will work better. They captured their results in three different types of files, unstructured English text files (emails, with headers), English text files with text-based structure (HTML documents) and English text files with a binary structure (Microsoft Word documents). In the end, they illustrated the problem of reusing the keystream, and they were able to achieve over 99 percent accuracy in recovery in some instances and said their attack is general and can be applied easily to the other types of data. In another research, Islam et al. [35], they did an investigation on access pattern disclosure and formalized a query identity inference model. They also verified the effectiveness of their model by analyzing it on a real-world dataset. They illustrate by knowing first, underlying keywords for k of the queries in the sequence Q, and second, knowing $m \times m$ matrix M which in each cell it contains the expected probability of both i^{th} and j^{th} keywords appearing in a random document. Later on, they indicate, with using this information and probabilistic analysis they are able to build a model and test in on *Enron* dataset [30]. For this purpose they used 30109 documents from the *_sent_mail* folder of the Enron dataset. They determined the root of each word and discarded the most common words. They measured the accuracy of their model with different keyword set and query set sizes. They illustrate that their model was able to achieve more than 80 percent accuracy on the default settings. In 2010, O. Sharif et al. [47], published a paper regarding the classification of encryption algorithms using pattern recognition techniques. They used eight different classifiers and an accurate measure for this task. Also, The following block cipher algorithms, DES, IDEA, AES, and RC operating in ECB mode, were considered. They created two datasets, the first one has eight classes (with different key sizes), and the second one contains four classes (one for each encryption algorithm). Their optimal result for these two datasets after doing a ten fold cross-validation is 30.83 percent for the first dataset and 53.33 for the second one. They also discovered that their approach was not able to distinguish between the different versions of AES encryption (128, 194, and 256). In 2018 Kaggle initiated a challenge [3] which users were supposed to classify the encrypted text. Their dataset was a sub-sample of 20 newsgroup dataset [1] which they encrypted into 4 different

encryption difficulties. Luis Bronchal published a kernel in that challenge [4] and tried to classify the dataset without breaking the ciphers. Using this approach, he was able to achieve an almost 48 percent F1 score. Moreover, it seems like his approach was much more successful for the two simpler encryptions since he could get an F1 score of more than 60 percent for each one of them. They experimented with different Query and keyword set sizes to show the effectiveness of their model.

2.3 Summary

In this chapter, I covered the concept of text classification as well as related researches that motivated me towards the encrypted text classification. First, I discussed the datasets that are currently available and can be used in my approach. Also, I went through the researches that has been done on these datasets and the results that they were able to achieve using their approach. Then I briefly talked about two of the most common feature extraction methods for text, which are Bag Of Words (BOW) and Term Frequency–Inverse Document Frequency (TF-IDF). In the next step, the classification algorithms that are useful for the classification of text data have been discussed. The aforementioned researches show that Naive Bayes, Logistic Regression, Random Forest, and SVM are useful for the classification of text data. In the next step, I covered the researches, which provided an insight toward finding a leakage in the encrypted text data. For this purpose, different approaches have been taken, such as classifying the black keywords, trying to decrypt the text data by keystream reuse, probabilistic analysis of the keywords, or using pattern recognition techniques. In the end, I used them as a motivation to propose my approach.

Chapter 3

Methodology

In this chapter, I will discuss the proposed model and different types of encryption algorithms. The datasets mostly consist of short text records like twitter text which makes the task of identifying the topic more challenging. Since some of the datasets are difficult to classify even in plaintext, I implemented the same parameters for all the classifiers. In figure 3.1, A general overview of the steps that I took towards implementing the methodology is illustrated.

3.1 Text Classification

In recent years text classification has become more popular, and it has been used in a variety of applications. [43] The improvements of Natural Language Processing (NLP), have been led to attract researches attention. The two important steps of the text classification are feature extraction and pre-processing. Pre-processing means that the data would be converted into a format that can be predicted. For instance, there are some unnecessary words in many of the text documents and data sets such as stop-words, misspelling, slang, etc. which should not be considered for the classification task. Also, feature extraction, in this case, means transforming the text into numerical representation in the form of a vector. [31]

1. N-grams

   N-grams of texts are widely used in the text mining and natural language processing tasks. [49][45][51] They are a set of co-occurring words or characters inside a window. When computing an n-gram, typically, we move the window one or even more than one word or characters forward. [31] For instance, in the sentence "Amir is drinking coffee", "Amir" could be a word n-gram for a window with a word range of 1. In the same sentence, "Am" is also an N-gram of characters with a range of 2, which we call it bi-grams. [31] Character N-grams is the method that I used because we cannot identify any words in the

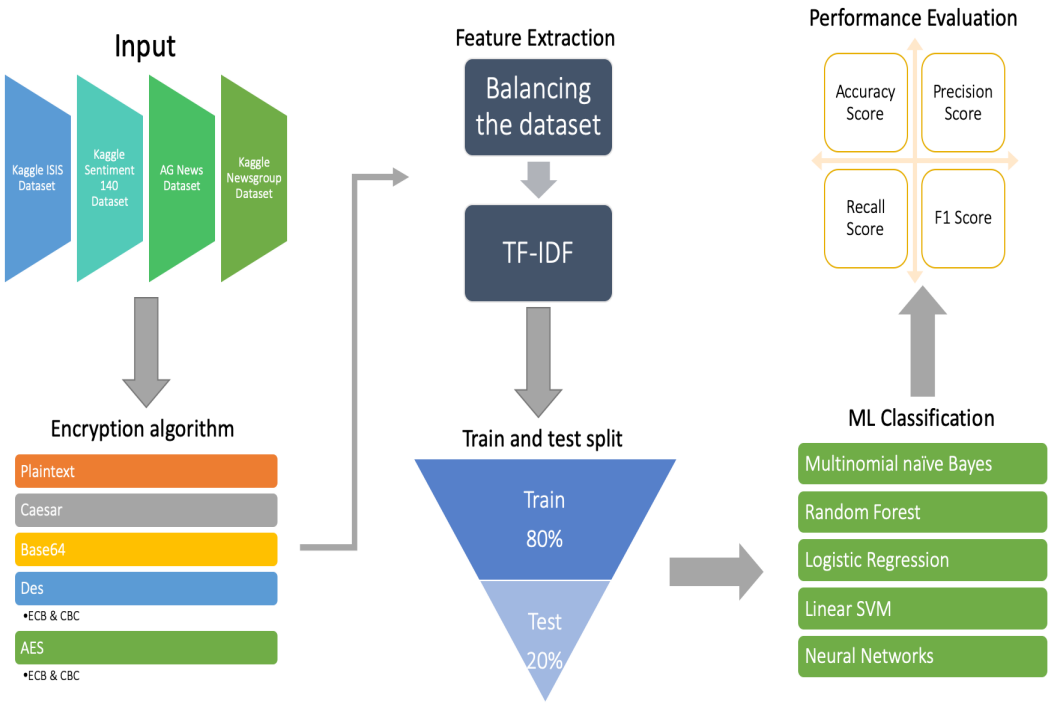


Figure (3.1) A general overview of methodology

ciphertext.

2. Text Vectorization

   Textual contents are needed to be converted to meaningful numerical representations, hence the machines would be able to understand them. This part is called feature representation. I am using TF-IDF method for that purpose. TF-IDF is one of the most common methods in NLP. It will convert the text documents into a matrix representation of vectors in a way that reflects the prominence of a word in the collection of documents. [46]

3. Term Frequency-Inverse Document Frequency (TF-IDF)

   One of the simplest ways of vectorizing the text documents is to just count the number of words in each document then assigning it to the feature space, which is called term frequency. K. Sparck Jones [27] purposed Inverse Document Frequency (IDF) method and combined it with term frequency to reduce the effect of common words in the text document. [31] TF-IDF determines a number for a word in each document indicating how relevant that word is in the document. By using this approach, it will create a numerical vector which represents each document. Thus, the documents with similar words will have similar representations. [40]

   Given a document collection D, a word w, and an individual document $d \epsilon D$, we calculate:

$$w_d = f_{w,d} * log(|D|/f_{w,D})$$

   Where $f_{w,d}$ equals the number of times w appears in d, $|D|$ is the size of the corpus, and $f_{w,D}$ equals the number of documents in which w appears in D (Salton & Buckley, 1988, Berger, et al, 2000).

The pre-processing step is crucial. However, the task is to deal with the ciphertext, which is consisted of different types of encoded characters. Therefore, we will not be able to distinguish the actual words in our document, hence pre-processing step is not going be helpful.

3.2 Encryption Algorithms

Encryption is the process of converting the plaintext into the ciphertext with a key. However, In the past, there used to be encryption algorithms that did not use a key like Caesar encryption. One of the crucial steps of having secure communication is cryptography. It provides services like confidentiality, data integrity, access control, authentication, and non-repudiation. It is important to select those encryption algorithms which will provide more security, accuracy, and effciency. The two main encryption methods are Probabilistic and deterministic. Deterministic encryption will always have the same ciphertext as the outcome because there is a one to one relationship between the keys and ciphertext. This process is done by repeating the encryption process for many times. On the other hand, the probabilistic encryption is way more secure compared to the deterministic encryption because it will add randomness to the encryption, which will produce a different ciphertext for the same plaintext after each encryption. [36] Below, we introduce some of the most popular encryption algorithms which I used to do the experiments on them:

1. Caesar Cipher

   Caesar Cipher is one of the classic and simple encryption algorithms available. It will use a fixed number to generate the letter. For instance, if n is the number, it will replace the letters with the one that is n places ahead of them. Encryption of a letter x by a shift n can be described mathematically:

   $$E_n(x) = (x + n)\ mod 26$$

   As can be seen, Caesar cipher can easily break using brute force because there are only 25 possibilities for the key available, which is easy to decrypt for nowadays computers. [20]

2. Base64 Encryption

   Base64 introduced to protect the emails. It can easily be decrypted since it does not use a key to encrypt the plaintext. It will encode binary data and transfers it to a representation of base 64. It will use the 64 printable characters in order to transfer the data across the media. The index value of the Base64 can be seen in table 3.1: [25]

Index	Char	Index	Char	Index	Char	Index	Binary
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
padding	=						

Table (3.1) Base64 index table

Since Base64 results are in the format of plaintext, it will be easier to send compared to the form of binary. Thus, Base64 encryption is being commonly used on the internet. [25]

3. Data Encryption Standard (DES)

   This standard introduced by IBM and adopted as a national standard in 1977. DES, takes a 64-bit plaintext and uses a 56-bit key to encipher it and produces 64-bit ciphertext messages. Considering key k ciphertext message block is: [19]

$$c = E_k(m)$$

   DES performs an Initial Permutation(IP), which will result in having two halves of the permuted block. These two are Left Plain Text (LPT) and Right Plain Text (RPT). In the next step, they will go through 16 rounds of the encryption process. Furthermore, there will be a Final Permutation which rejoins LPT and RPT and creates a combined block. Finally, we will have 64-bit block, which is the final ciphertext block.[19]

4. Advanced Encryption System (AES)

This algorithm is introduced by the National Institute of Standards and Technology (NIST) in 2000. The goal of publishing this algorithm was to replace DES due to its vulnerabilities. AES is supporting three different key sizes, namely 128, 192, and 256 bit. The size of each ciphertext block in the AES algorithm equals to 128 bits.

For each one of AES-128, AES-192, and AES-128 keys, AES would process encryption and decryption in 128 bits blocks. Since it is a symmetric method, the secret key should be known to both the sender and receiver. There are 10, 12, and 14 rounds of encryption for 128, 192, and 256 key sizes, respectively. [8]

For the last two encryption, we also have the option to choose between Electronic codebook (ECB) and Cipher Block Chaining (CBC). For the ECB encryption, each block is encrypted and decrypted separately. Using this model, the plaintext might show up in the block but not very clear, which makes it more vulnerable to attacks. However, the advantage of it is that you can encrypt and decrypt the blocks in parallel and process them simultaneously. On the other hand, CBC has another way to connect the blocks together. As can be seen in figure 3.2, instead of processing each block separately, they will be XOR'ed with the outcome of the previous encryption block. In this case, the parallel encryption is not possible because first, we need to encrypt one block, then we will be able to process the next block's encryption. In addition, it needs an Initialization Vector (IV), which is the same size as the block size. It will be XOR'ed with the first plaintext block, which will prevent the repetition of corresponding duplicate character sequences in the final ciphertext.[11]

3.3 Machine Learning Classifiers

Classification is a supervised learning method that learns from the input data to learn and classify the new data given to it. There are different types of machine learning algorithms that can be used for classification. However, in this work, I will focus on the following classification algorithms, since they are in the literature, closest to my research area. [14]

1. Multinomial Naive Bayes

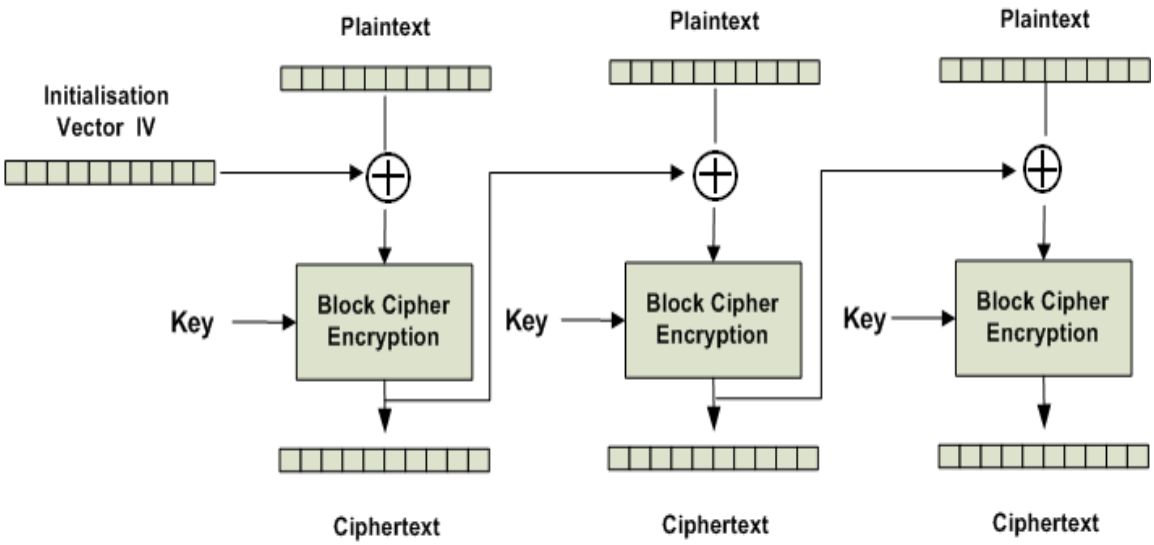


Figure (3.2) Cipher block chaining mode(CBC) Encryption mode [42]

Naive Bayes is the simplest algorithm. It assumes that all the variables in the dataset are independent and not correlated to each other. Naive Bayes is one of the most common classification algorithms because it is easy to use, and it will be efficient to get a good result. According to the Bayes rule, the probability of an example $E = (x_1, x_2, , x_n)$ being class c is:

$$P(c|E) = \frac{P(E|c)p(c)}{P(E)}$$

In this equation, c is evidence and E is hypothesis. We consider that features are independent, so the absence of a particular feature will not affect other features. It is called multinomial because it works with the data that could be count like word counts in text and follows a multinomial distribution. It is a very simple and fast classifier; however, it has the disadvantage of predictors being independent of each other, which is not true in most of the cases.[54]

2. Random Forest

   Random forest classifier is made by several random decision trees. An example of a Decision tree classification process is illustrated in figure 3.3. Firstly, the random forest builds a tree on a random sample of the data. Then, for each one of them, it selects a random set of features to generate the best split. They are a prediction, and by aggregating them, it will predict the new data. [33]

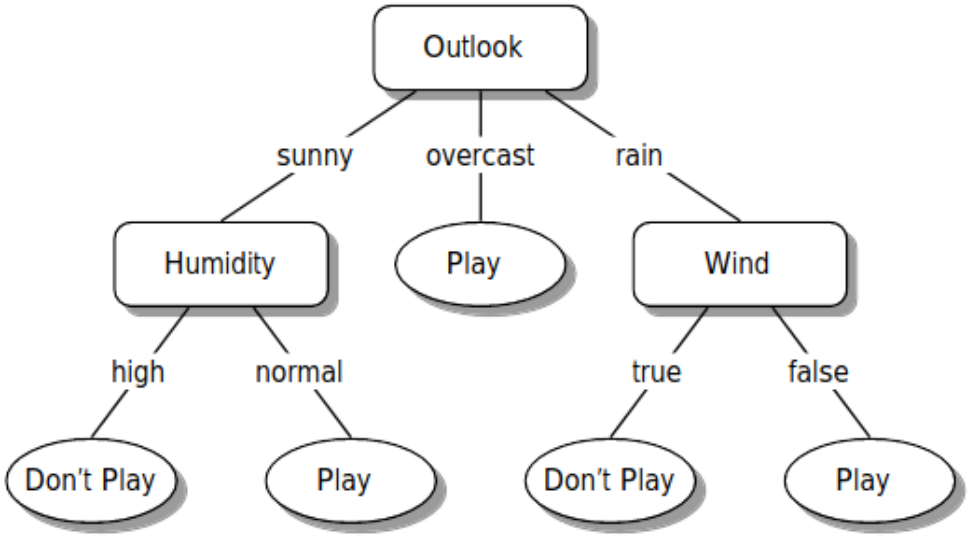


Figure (3.3) A decision tree for the "Golf" dataset. Branches correspond to the values of attributes; leaves indicate classifications.[24]

Building blocks of a random forest model are decision trees. Decision trees first will define a reasonable solution to our problem. Then they will limit the range of the solution to get the final answer. For instance, in figure 3.3 we can see a detailed example of which features do we need to play tennis. In addition, Random forest increases the diversity by choosing a random subset of features and performing them on a random set of the training data, which will help us to have a more robust result.[54]

3. Logistic Regression Logistic Regression Logistic Regression fits the data to the "Logistic Function" and returns the probability of occurrence. It is commonly used for multi-class classification tasks. It is a version of linear regression when the target is categorical. Prediction is made by getting the maximum likelihood, which provides a constant output. The logistic function is shown in figure 3.4. By maximizing the likelihood function, it will determine the parameters which will most likely produce the actual target. It uses a sigmoid function as the logistic function and maps the value between 0 and 1. Given x the value, the sigmoid function is: [18]

$$f(x) = \frac{1}{1+e^{-}(x)}$$

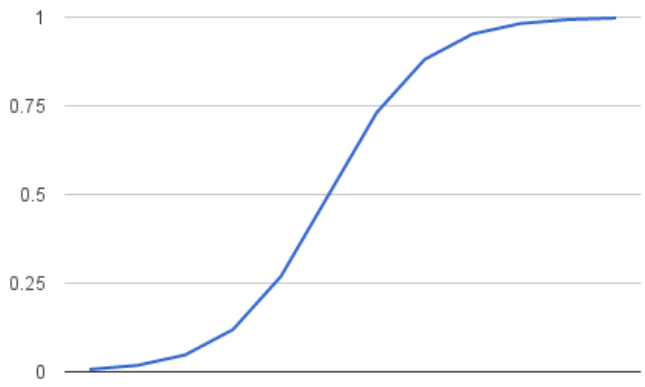


Figure (3.4) Logistic Function for Logistic Regression algorithm.

4. Linear SVM

   The support vector machine (SVM) is defined by a separating hyperplane, which classifies the data points in our dataset. The hyperplane and parameters of SVM are shown in figure 3.5. There are many possible ways to draw this hyperplane, but the objective of the SVM is to find the hyperplane that has the maximum distance between the datapoints of each class. The datapoints that are closer to the hyperplane are called support vectors since they specify the position of the hyperplane. For making a new prediction considering x as the input and x_i as each support vector, the equation is :

$$f(x) = B_0 + \sum_{i=0} a_i \times (x, x_i)$$

   In this equation, the coefficients B_0 and a_i are coming from the training data, using the learning algorithm.[16]

3.4 Datasets

1. kaggle Isis dataset

   This data set is intended to be a counterpoise to the How Isis Uses Twitter data set. That data set contains 17k tweets alleged to originate with "100+ pro-ISIS

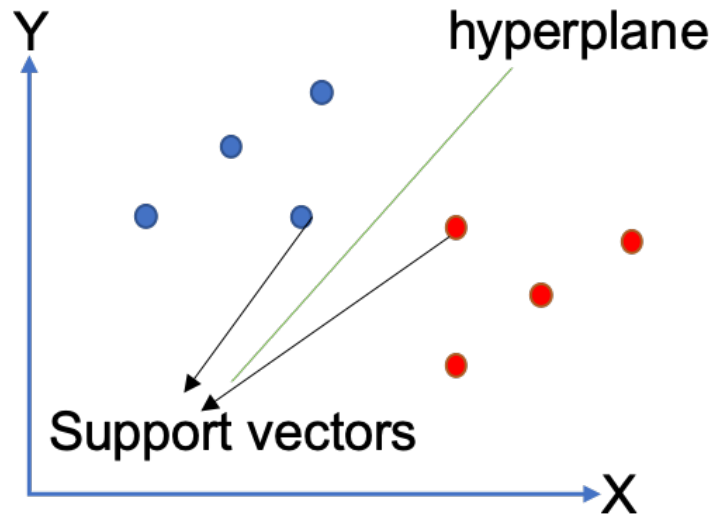


Figure (3.5) SVM classification parameters

Figure (3.6) Word cloud representation of Kaggle Isis dataset

Class	0	1
Items	15000	15000
Train(0.8)	**Test(0.2)**	
24000	6000	

Table (3.2) Kaggle Isis dataset split, Class 0 = Non-fanboy tweets, Class 1 = Fanboy tweets

Figure (3.7) Word cloud representation of Sentiment 140 dataset

fanboys". This new set contains 122k tweets collected on two separate days, 7/4/2016 and 7/11/2016, which contained any of the following terms, with no further editing or selection: "isis, isil, daesh, islamicstate, raqqa, Mosul, islamic state". Also, there are no "description, location, followers, numberstatuses" data columns. By looking at the word cloud representation in figure 3.6, it can be seen some words like Islamic State, ISI, Syria and etc. are used the most in these tweets, which is helpful to have a better understanding of the dataset. We use "tweets" column as our text data and "Fanboy" as the target. Each tweet has been labeled with 0 (Related Tweet) or 1 (Fanboy Tweet). I split the dataset into 80 percent train and 20 percent test, which can be found in table 3.2.

2. Sentiment 140

Sentiment 140 started as a class project at Stanford University which led to

Class	**0**	**4**
Items	15000	15000
Train(0.8)	**Test(0.2)**	
24000	6000	

Table (3.3) Sentiment 140 dataset, Class 0 = Negative tweets, Class 4 = Positive tweets

studying various aspects of sentiment classification. They automatically gathered and labeled the training data by labeling each tweet that contained positive emotions like :) as positive and the ones who had a negative emotion like :(as negative and twitter search API is used to collect these tweets. I used a subset of the original dataset, which can be found on the Kaggle website. This dataset contains 1.6 million extracted from twitter. Since the dataset consists of different topics, the classification task is already difficult on the plaintext. Also, the results should be reliable on plaintext so they can be compared more precisely with the plaintext while classifying the encrypted data. In order to get better performance on the dataset, I decided to use certain topics of the tweets and create a subset of the data to work on. For that purpose, I used the topic of technology and IT and extracted the tweets which contained the name of the most famous high tech companies like Facebook, Apple, Microsoft etc. The word cloud representation of this dataset illustrated in figure 3.7. The dataset is classified into two targets (0 = negative, 4 = positive), and they can be used to detect sentiment. I used 30000 tweets from that dataset and split it into 80 percent train and 20 percent test, which can be found in table 3.3.

3. News Category dataset

   In figure 3.9, , we can observe an example of the news category dataset. This dataset includes approximately 200k news headlines from Huffpots, which gathered from the year 2012 to 2018. Each headline corresponds to a category, and we chose four categories that had enough records to make a multi-class dataset. We used a short description of the news as our text data, and this dataset also contains 30000 records, and I split it into 80 percent train and 20 percent test, which is shown in table 3.4. Also, we used an additional feature (headline) to classify the records in a different part of our experiment to see how much

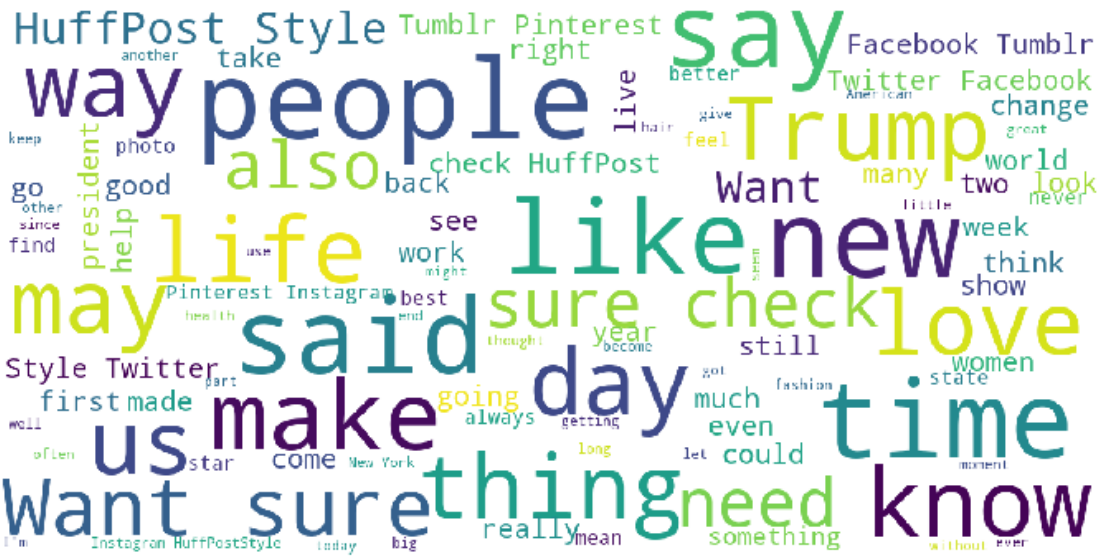


Figure (3.8) Word cloud representation of News Category dataset

Class	1	2	3	4
Items	7500	7500	7500	7500
Train (0.8)	**Test (0.2)**			
24000	6000			

Table (3.4) News category dataset, Class 1 = Entertainment, Class 2 = Politics, Class 3 = Style & Beauty, Class 4 = Wellness

difference an additional feature can make in the performance of our classifier. Moreover, figure 3.8 shows a word cloud representaion of this dataset.

4. AG News dataset [2]

   This dataset is collected from 1 million news articles. They have been gathered from more than 2000 news sources over a period of 1 year. The academic community provided this dataset for research purposes, and it is constructed by Xiang Zhang. They also used it as a text classification benchmark in a paper [52]. Figure 3.10 is illustrating the word cloud representation for this dataset. The dataset is constructed by choosing 4 largest classes, which are World, Sports, Business, Sci/Tech from the original corpus. It contains 120000 records, and there are 30000 records in each class. Also, I used the same strategy for this dataset and split it into 80 percent train and 20 percent test, which can be seen in table 3.5.

	category	headline	authors	link	short_description	date
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...	Melissa Jeltsen	https://www.huffingtonpost.com/entry/texas-ama...	She left her husband. He killed their children...	2018-05-26
1	ENTERTAINMENT	Will Smith Joins Diplo And Nicky Jam For The 2...	Andy McDonald	https://www.huffingtonpost.com/entry/will-smit...	Of course it has a song.	2018-05-26
2	ENTERTAINMENT	Hugh Grant Marries For The First Time At Age 57	Ron Dicker	https://www.huffingtonpost.com/entry/hugh-gran...	The actor and his longtime girlfriend Anna Ebe...	2018-05-26
3	ENTERTAINMENT	Jim Carrey Blasts 'Castrato' Adam Schiff And D...	Ron Dicker	https://www.huffingtonpost.com/entry/jim-carre...	The actor gives Dems an ass-kicking for not fi...	2018-05-26
4	ENTERTAINMENT	Julianna Margulies Uses Donald Trump Poop Bags...	Ron Dicker	https://www.huffingtonpost.com/entry/julianna-...	The "Dietland" actress said using the bags is ...	2018-05-26
...	...	...	...	...	...	...

Figure (3.9) Example of News Category dataset

Figure (3.10) Word cloud representation of AG News dataset

Class	**1**	**2**	**3**	**4**
Items	30000	30000	30000	30000
Train (0.8)	**Test (0.2)**			
96000	24000			

Table (3.5) AG news dataset with 4 different topics

3.5 Performance Metrics

1. Accuracy Score [38]

   This metric is used to evaluate the classification model using the fraction of predictions that model, successfully predicted. Following, is the formal definition of accuracy:

   $$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ Number\ of\ predictions}$$

   It can also be defined as below for Binary classification:

   $$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

   Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

2. Precision Score [38]

   Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Precision formula is as follows:

   $$Precision = \frac{TP}{TP+FP}$$

3. Recall Score [38]

   Recall is the ratio of correctly predicted positive observations to the all observations in actual class and the formula to calculate it is:

   $$Recall = \frac{TP}{TP+FN}$$

4. F1 Score [38]

   F1 can be considered as the weighted average of Precision and Recall. Since it takes both false positive and false negatives into account, it will be more useful in case of having an uneven class distribution. The equation for this metric is:

   $$F1\ Score = \frac{2\times(Recall\times Precision)}{(Recall+Precision)}$$

3.6 Tools and Environments used

For the implementation I wrote the code in **Python** language. I used **Pandas** package to read the data. In addition, **Labelencoder** Package from **Sklearn** is used to convert categorical target names to numerical so that the classifiers could easily categorize them, and for encrypting the text data, I used **Crypto** package. Also, I used Numpy pakckage to create an array as the input to feed to the classifiers. In order to convert the text into a vector, I used **Countvectorizer** and **TfidfTransformer** packages from Sklearn. Moreover, **train_test_split** function from Sklearn is used to split the data into train and test. Finally, I used **Multinomial Naive Bayes**, **Random Forest**, **Logistic Regression** and **Linear SVM** from Sklearn to classify the data, same as **accuracy_score**, **precision_score**, **recall_score** and **F1_score**.
Also, for the neural networks experiments I used **Matlab** which already included necessary tools inside of it.

3.7 The Approach

As I described earlier, the classification of ciphertext has more difficulties, which makes it different from a normal text classification task. More specifically, because my focus is to find patterns in ciphertext rather than mapping the ciphertext to the actual plaintext, I will not be able to implement NLP techniques such as stemming, lemmatization, named entity recognition and sentiment analysis. I used TF-IDF method as the feature representation to vectorize the data because the data is large enough and we know that the most frequent word is a less useful metric and the characters in the ciphertext are representing these words in a different order, since some words like' this',' a' occur very frequently across all documents. I balanced the dataset and removed the duplicated texts. Then, I split the dataset into train and test. In the next part, I encoded the text using the encryption function. For the feature extraction, after doing some experiments, I chose 5000 as the maximum number of features and the N-gram range of (3,6). Finally, I trained the classifier, calculated the scores (Accuracy, Precision, Recall, and F1), and draw the confusion matrices and learning curves. Also, in another experiment, I implemented neural

networks for this approach. I created a feedforward neural network in Matlab, and I demonstrated it with different neurons and different set of features to see the effect of neural networks in the approach. For this part, confusion matrices and learning curves are illustrated.

3.8 Summary

As summarized in this chapter, dealing with the ciphertext has a lot of unique challenges since we cannot implement the NLP techniques on them. Also, text classification steps such as n-grams and text vectorization have been discussed in this chapter. The experiments implemented on four different encryptions named Base64, Caesar, DES, and AES. for the last two encryptions, there are two modes ECB (Electronic Code Book) and CBC (Cipher Block Chaining). The experiments are done on four different datasets. Kaggle Isis and Sentiment 140 are binary, News Category, and AG News are multi-class datasets. The classification task on these datasets has been done without any parameter tuning to have a baseline score. The parameters are the same across all the datasets for the purpose of having a better comparison among the datasets and encryptions. In the end, a neural network experiment is done to see the difference of using a neural networks model on the task to check if it can perform better than regular classification on the task.

Chapter 4

Results

In this chapter, I will illustrate the results of our approach to see how each classifier performed on each dataset on different types of encryption. Also, we take a look at the deep learning results and compare them to our regular classification results.

4.1 Machine Learning Classifiers

1. Plaintext Results

   It can be seen that balancing the dataset made the results more precise in terms of their scores. In addition, It is obvious from table 4.1 that in almost all the datasets, Logistic Regression and Linear SVM outperformed the two other classification algorithms. For sentiment 140, the results are around 76 percent for Linear SVM, around 72 percent for Multinomial Naive Bayes, and around 70 percent for Random forest on the plaintext. Moving forward to the News Category dataset, the same trend is obvious in table 4.2. Again, the two best classifiers, Linear SVM, and Logistic Regression, are having a result of around 80 percent. However, in this dataset, Multinomial Naive Bayes outperforms the Random Forest with around 67 percent comparing to the 64. Also, by taking a look at the figure from table 4.3, by adding a feature to the News category dataset, it is obvious that all the classifiers had much better performance, and the results were increased 5-10 percent depending on the classification algorithm. The added feature is the headline, which contains more information about the tweets as a short text. Hence, it shows that having a longer text will boost the result of the classification task. Next, I did the experiments on AG News dataset in table 4.4.For Logistic Regression and Linear SVM the results are approximately 89 percent which considering that it is a multi-class dataset. The results on plaintext are surprisingly good. On the other hand, Random forest and Multinomial Naive Bayes were not as good with

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.7363	0.7693	0.7942	**0.8194**
Precision	0.7377	0.7708	0.7941	**0.8192**
Recall	0.7373	0.7704	0.7943	**0.8194**
F1	0.7363	0.7693	0.7941	**0.8193**
Test				
Accuracy	0.7231	0.6995	**0.7573**	0.7527
Precision	0.7245	0.7002	**0.7575**	0.7528
Recall	0.7241	0.7002	**0.7577**	0.7530
F1	0.7230	0.6995	**0.7573**	0.7526

Table (4.1) Results on train and test set for Sentiment 140 Dataset using Plaintext

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6603	0.7847	0.8439	**0.8818**
Precision	0.6783	0.8178	0.8455	**0.8833**
Recall	0.6605	0.7845	0.8438	**0.8817**
F1	0.6560	0.7850	0.8437	**0.8817**
Test				
Accuracy	0.6458	0.6773	0.7977	**0.7980**
Precision	0.6647	0.7092	0.7991	**0.7993**
Recall	0.6449	0.7850	0.7978	**0.7982**
F1	0.6403	0.6714	0.7975	**0.7979**

Table (4.2) Results on train and test set for News Category Dataset using Plaintext

around 72 and 76 percent, respectively. Finally, Kaggle Isis dataset seemed to have the best results on plaintext according to table 4.5. The F1 score for Linear SVM and Logistic regression is around 97 and 96 percent, respectively. Also, using Multinomial Naive Bayes the F1 score is around 87 percent, and for Random Forest is around 89 percent.

2. Caesar Encryption

   Generally, Caesar encryption results are not so different from plaintext since Caesar algorithm just shifts the characters, and we are using character n-grams. Table 4.6 rrepresents the scores of the tweets encrypted by Caesar encryption method on the Sentiment 140 dataset. This figure reveals that the scores are almost identical to the plaintext scores using any of the four described classifiers. News Category dataset on tables 4.7 and 4.8, AG News dataset on table 4.9, and Kaggle Isis dataset on Table 4.10 are following the same trend. This also

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.8143	0.7786	0.9078	**0.9553**
Precision	0.8220	0.8186	0.9096	**0.9555**
Recall	0.8142	0.7786	0.9078	**0.9552**
F1	0.8144	0.7828	0.9079	**0.9553**
Test				
Accuracy	0.8037	0.7275	0.8782	**0.8930**
Precision	0.8131	0.7690	0.8816	**0.8945**
Recall	0.8039	0.7277	0.8784	**0.8933**
F1	0.8041	0.7327	0.8786	**0.8933**

Table (4.3) Results on train and test set for News Category with additional feature Dataset using Plaintext

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.7714	0.7687	0.9015	**0.9206**
Precision	0.7876	0.7722	0.9012	**0.9204**
Recall	0.7711	0.7685	0.9014	**0.9205**
F1	0.7663	0.7674	0.9011	**0.9204**
Test				
Accuracy	0.7613	0.7220	0.8873	**0.8933**
Precision	0.7773	0.7243	0.8873	**0.8933**
Recall	0.7625	0.7226	0.8877	**0.8936**
F1	0.7563	0.7201	0.8874	**0.8934**

Table (4.4) Results on train and test set for AG News Dataset using Plaintext

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.9031	0.9340	0.9736	**0.9897**
Precision	0.8843	0.9417	0.9727	**0.9889**
Recall	0.8636	0.8887	0.9597	**0.9849**
F1	0.8731	0.9109	0.9660	**0.9869**
Test				
Accuracy	0.8990	0.9201	0.9678	**0.9778**
Precision	0.8825	0.9260	0.9672	**0.9744**
Recall	0.8610	0.8712	0.9518	**0.9699**
F1	0.8708	0.8936	0.9591	**0.9721**

Table (4.5) Results on train and test set for Kaggle ISIS Dataset using Plaintext

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.7383	0.7741	0.7919	**0.8215**
Precision	0.7397	0.7792	0.7918	**0.8214**
Recall	0.7392	0.7759	0.7920	**0.8216**
F1	0.7383	0.7737	0.7918	**0.8214**
Test				
Accuracy	0.7253	0.6920	**0.7567**	0.7531
Precision	0.7270	0.6962	**0.7566**	0.7529
Recall	0.7267	0.6943	**0.7570**	0.7532
F1	0.7253	0.6916	**0.7566**	0.7530

Table (4.6) Results on train and test set for Sentiment 140 Dataset using Caesar encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6610	0.7971	0.8454	**0.8872**
Precision	0.6830	0.8260	0.8468	**0.8884**
Recall	0.6611	0.7973	0.8455	**0.8873**
F1	0.6569	0.7974	0.8453	**0.8872**
Test				
Accuracy	0.6520	0.6885	0.7948	**0.7983**
Precision	0.6742	0.7151	0.7959	**0.7987**
Recall	0.6513	0.6873	0.7944	**0.7979**
F1	0.6467	0.6831	0.7944	**0.7979**

Table (4.7) Results on train and test set for News Category Dataset using Caesar encryption

proves the consistency of our experiments, since I achieved the results that I expected already.

3. Base64 Encryption

   As we move forward to the more complicated encryptions, it is eminent that the scores will drop down, and the performance would not be the same. The results of the Base64 encryption algorithm on the Sentiment 140 dataset are shown in table 4.11. The results show that Logistic regression and Linear SVM classifiers could achieve the F1 score of 72 percent on the test set. Meanwhile, Multinomial Naive Bayes' performance was down by 4 percent comparing to the first two classifiers with a score of around 68 percent. Moreover, the Random forest score seems to be the lowest as it had a 65 percent F1 score on the test

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.8166	0.7829	0.9052	**0.9547**
Precision	0.8236	0.8191	0.9072	**0.9551**
Recall	0.8167	0.7830	0.9053	**0.9548**
F1	0.8170	0.7867	0.9054	**0.9548**
Test				
Accuracy	0.8113	0.7405	0.8887	**0.8985**
Precision	0.8191	0.7786	0.8911	**0.8994**
Recall	0.8110	0.7401	0.8885	**0.8984**
F1	0.8117	0.7443	0.8889	**0.8986**

Table (4.8) Results on train and test set for News Category with additional feature Dataset Caesar encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.7729	0.7708	0.9006	**0.9191**
Precision	0.7878	0.7745	0.9004	**0.9190**
Recall	0.7731	0.7708	0.9006	**0.9191**
F1	0.7678	0.7695	0.9004	**0.9190**
Test				
Accuracy	0.7666	0.7252	0.8878	**0.8940**
Precision	0.7821	0.7264	0.8873	**0.8936**
Recall	0.7661	0.7252	0.8878	**0.8940**
F1	0.7613	0.7230	0.8875	**0.8937**

Table (4.9) Results on train and test set for AG News Dataset using Caesar encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.8842	0.9608	0.9630	**0.9823**
Precision	0.8867	0.9609	0.9630	**0.9823**
Recall	0.8843	0.9608	0.9630	**0.9823**
F1	0.8840	0.9608	0.9630	**0.9823**
Test				
Accuracy	0.8837	0.9383	0.9508	**0.9597**
Precision	0.8856	0.9384	0.9508	**0.9597**
Recall	0.8833	0.9383	0.9508	**0.9597**
F1	0.8834	0.9383	0.9508	**0.9597**

Table (4.10) Results on train and test set for Kaggle Isis Dataset using Caesar encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6960	0.7300	0.7607	**0.7892**
Precision	0.6983	0.7412	0.7606	**0.7891**
Recall	0.6972	0.7329	0.7605	**0.7890**
F1	0.6958	0.7283	0.7605	**0.7891**
Test				
Accuracy	0.6846	0.6499	0.7208	**0.7213**
Precision	0.6875	0.6603	0.7208	**0.7211**
Recall	0.6864	0.6539	0.7206	**0.7209**
F1	0.6845	0.6475	0.7206	**0.7210**

Table (4.11) Results on train and test set for Sentiment 140 Dataset using Base64 encryption

set. The performance of the News Category dataset is illustrated in table 4.12. SVM did slightly a better job of classifying the encrypted text with the F1 score of around 74 percent, comparing to the Logistic regression with 73 percent on the test set. Random forest and Multinomial Naive Bayes come after these two with the F1 score of around 62 and 57 percent, respectively. In this case, having an additional feature helped a lot with the task. By looking at table 4.13, we can see that the performance of Linear SVM boosted by almost 12 percent with the F1 score of 86. It was the same for Logistic Regression with almost 83 percent F1 score. However, the boost was not as much for the remaining two classifiers Multinomial Naive Bayes with around 64 and Random Forest with almost 66 percent F1 score. For AG news dataset in table 4.14, Linear SVM and Logistic Regression had almost similar results with the F1 score of around 89 percent. In addition, by using Random forest and Multinomial Naive Bayes, the F1 score was around 73 and 76 percent, respectively. Also, it is obvious from table 4.15, that even with Base64 encryption, Kaggle Isis classifier still has the best results.

4. DES(ECB) Encryption

   Since this encryption is a little more complicated than we covered so far, it is expected to have much lower scores than the previous encryptions. In table 4.16, the best classification results belong to the Linear SVM and Logistic Regression, with almost 55 percent F1 score. Similarly, for Multinomial Naive Bayes and

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6000	0.8700	0.8043	**0.8497**
Precision	0.6175	0.8804	0.8046	**0.8508**
Recall	0.6002	0.8701	0.8043	**0.8497**
F1	0.5900	0.8708	0.8039	**0.8494**
Test				
Accuracy	0.5862	0.6312	0.7383	**0.7472**
Precision	0.5989	0.6327	0.7379	**0.7473**
Recall	0.5855	0.6302	0.7382	**0.7470**
F1	0.5738	0.6246	0.7379	**0.7466**

Table (4.12) Results on train and test set for News Category Dataset using Base64 encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6484	0.7254	0.8702	**0.9270**
Precision	0.7306	0.7804	0.8742	**0.9278**
Recall	0.6489	0.7251	0.8703	**0.9270**
F1	0.6432	0.7229	0.8703	**0.9270**
Test				
Accuracy	0.6375	0.6582	0.8340	**0.8557**
Precision	0.7270	0.7102	0.8389	**0.8573**
Recall	0.6355	0.6591	0.8340	**0.8557**
F1	0.6337	0.6568	0.8344	**0.8559**

Table (4.13) Results on train and test set for News Category Dataset with additional feature using Base64 encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.7724	0.7713	0.9015	**0.9194**
Precision	0.7875	0.7755	0.9013	**0.9193**
Recall	0.7728	0.7715	0.9016	**0.9195**
F1	0.7675	0.7705	0.9014	**0.9193**
Test				
Accuracy	0.7662	0.7309	0.8868	**0.8940**
Precision	0.7821	0.7333	0.8862	**0.8934**
Recall	0.7644	0.7299	0.8862	**0.8934**
F1	0.7606	0.7289	0.8861	**0.8933**

Table (4.14) Results on train and test set for AG News Dataset using Base64 encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.8701	0.9160	0.9578	**0.9853**
Precision	0.8701	0.9179	0.9578	**0.9853**
Recall	0.8701	0.9160	0.9578	**0.9853**
F1	0.8701	0.9159	0.9578	**0.9853**
Test				
Accuracy	0.8647	0.8875	0.9427	**0.9598**
Precision	0.8648	0.8889	0.9427	**0.9598**
Recall	0.8647	0.8877	0.9427	**0.9598**
F1	0.8647	0.8874	0.9427	**0.9598**

Table (4.15) Results on train and test set for Kaggle ISIS Dataset using Base64 encryption

Random Forest, the score is around 53 and 43, respectively. Also, it seems like the classifiers were not able to classify the News Category targets correctly. Looking at table 4.17, it can be understood that Logistic Regression performed better than the other classifiers with almost 52 percent F1 score. Linear SVM was the second best with around 50 percent. Multinomial Naive Bayes and Random forest both had the same score of around 37 percent. Considering this dataset is multi-class (4 classes), it can be said that the scores are better than a random prediction; hence, maybe the classifiers were able to find a pattern in the encrypted data for the classification task. In the next step, it is obvious from table 4.18 that adding a feature did not help the classification task since there's not much improvement in the F1 scores. Moreover, AG News dataset seems to fail the classification task as well. Because what can be seen in table 4.19, all the F1 scores for the classifiers are around 25 percent or less. While knowing that the dataset has only four classes, it means that it is basically a random selection of classes. On the other hand, the Kaggle Isis dataset in table 4.20, was the dataset that the classifiers had the best performance. Logistic Regression and Linear SVM classified almost 75 percent of the test records correctly. Although Multinomial Naive Bayes, with only 51 percent F1, seems to fail the task, Random Forest does a slightly better job in classification with around 58 percent. Considering the discussed results, it seems like there is the possibility of finding a pattern to classify encrypted text with the DES(ECB) method.

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6399	0.5662	0.6891	**0.6994**
Precision	0.6539	0.6366	0.6889	**0.6991**
Recall	0.6441	0.5537	0.6880	**0.6988**
F1	0.6354	0.4807	0.6882	**0.6989**
Test				
Accuracy	0.5418	0.5203	0.5478	**0.5483**
Precision	0.5476	0.5306	0.5472	**0.5478**
Recall	0.5446	0.5120	0.5467	**0.5476**
F1	0.5355	0.4285	0.5460	**0.5474**

Table (4.16) Results on train and test set for Sentiment 140 Dataset using DES(ECB) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.4513	0.4325	0.6994	**0.7222**
Precision	0.5419	0.5771	0.6985	**0.7217**
Recall	0.4511	0.4321	0.6993	**0.7221**
F1	0.4282	0.4056	0.6971	**0.7198**
Test				
Accuracy	0.3968	0.4038	**0.5273**	0.4998
Precision	0.4743	0.5068	**0.5235**	0.4960
Recall	0.3981	0.4054	**0.5278**	0.5003
F1	0.3684	0.3695	**0.5238**	0.4965

Table (4.17) Results on train and test set for News Category Dataset using DES(ECB) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.3651	0.4480	0.6895	**0.7245**
Precision	0.6136	0.5313	0.6937	**0.7263**
Recall	0.3645	0.4478	0.6895	**0.7246**
F1	0.3048	0.4453	0.6894	**0.7244**
Test				
Accuracy	0.2990	0.4013	**0.5267**	0.5140
Precision	0.4798	0.4662	**0.5320**	0.5154
Recall	0.3013	0.4022	**0.5265**	0.5139
F1	0.2361	0.3966	**0.5273**	0.5138

Table (4.18) Results on train and test set for News Category Dataset with additional feature using DES(ECB) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.5012	0.3822	0.5406	**0.5446**
Precision	0.5023	0.4604	0.5409	**0.5448**
Recall	0.5013	0.3827	0.5407	**0.5446**
F1	0.5009	0.3679	0.5404	**0.5443**
Test				
Accuracy	0.4509	0.3484	**0.4678**	0.4654
Precision	0.4513	0.4020	**0.4675**	0.4651
Recall	0.4506	0.3465	**0.4677**	0.4653
F1	0.4503	0.3278	**0.4674**	0.4649

Table (4.19) Results on train and test set for AG News Dataset using DES(ECB) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.5905	0.6392	0.8257	**0.8475**
Precision	0.5910	0.7642	0.8260	**0.8475**
Recall	0.5906	0.6409	0.8257	**0.8475**
F1	0.5901	0.5923	0.8256	**0.8475**
Test				
Accuracy	0.5145	0.6307	0.7493	**0.7497**
Precision	0.5141	0.7400	**0.7498**	0.7496
Recall	0.5140	0.6239	0.7488	**0.7495**
F1	0.5133	0.5765	0.7489	**0.7495**

Table (4.20) Results on train and test set for Kaggle ISIS Dataset using DES(ECB) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6617	0.5743	0.6828	**0.6877**
Precision	0.6616	0.6397	0.6831	**0.6875**
Recall	0.6618	0.5628	0.6814	**0.6870**
F1	0.6616	0.4994	0.6815	**0.6871**
Test				
Accuracy	0.5043	**0.5073**	0.5031	0.5043
Precision	**0.5040**	0.4960	0.5018	0.5035
Recall	**0.5040**	0.4982	0.5017	0.5034
F1	**0.5039**	0.4207	0.5008	0.5032

Table (4.21) Results on train and test set for Sentiment 140 Dataset using DES(CBC) encryption

5. DES(CBC) Encryption

   This encryption seems to be one of the most complicated encryptions we are covering so far. By having a glance at table 4.21, it can be understood that because the F1 scores for all the classifiers are 50 percent or below that. In addition, since this dataset is binary, it can be said that this approach fails for the Sentiment 140 dataset with DES(CBC) encryption. The News Category dataset, however, shows some better results than the Sentiment 140 in table 4.22. Achieving around 52 percent F1 score on the test set for Logistic Regression and around 50 percent for Linear SVM, it seems like these two classifiers did a slightly better job than random classification. Also, similar to the previous dataset adding the feature did not improve the scores as it can be seen in table 4.23. In table 4.24, it is clear that the results for AG news dataset are basically the same as DES(ECB). Hence, classifying this dataset with DES encryption seems to be a challenging task. Finally, form table 4.25, it can be seen that even for the dataset that had the best classification results so far, the scores are the same as a random classification. From this information, it can be said that this approach is not successful classifying DES(CBC) encrypted texts.

6. AES(ECB) Encryption

   Going towards the next encryption, the same trend as DES is evident for AES. For the Sentiment 140 dataset in table 4.26, it can be stated that all the classifiers failed in classification task since all the F1 scores are 50 percent or less on

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.5485	0.5879	0.6645	**0.6781**
Precision	0.5484	0.5976	0.6614	**0.6748**
Recall	0.5484	0.5876	0.6643	**0.6779**
F1	0.5482	0.5789	0.6611	**0.6735**
Test				
Accuracy	0.2840	0.3757	**0.4195**	0.3930
Precision	0.2840	0.3607	**0.4094**	0.3798
Recall	0.2844	0.3775	**0.4204**	0.3941
F1	0.2838	0.3467	**0.4133**	0.3844

Table (4.22) Results on train and test set for News Category Dataset using DES(CBC) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.5264	0.3752	0.6235	**0.6424**
Precision	0.5265	0.5057	0.6219	**0.6407**
Recall	0.5264	0.3750	0.6234	**0.6424**
F1	0.5264	0.3198	0.6207	**0.6399**
Test				
Accuracy	0.2645	0.3185	**0.3500**	0.3443
Precision	0.2645	**0.3662**	0.3426	0.3377
Recall	0.2646	0.3195	**0.3501**	0.3444
F1	0.2644	0.3195	**0.3451**	0.3401

Table (4.23) Results on train and test set for News Category Dataset with additional feature using DES(CBC) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.3783	0.2943	0.3852	**0.3858**
Precision	0.3782	0.3544	0.3851	**0.3857**
Recall	0.3783	0.2935	0.3852	**0.3858**
F1	0.3782	0.2391	0.3851	**0.3857**
Test				
Accuracy	0.2538	**0.2591**	0.2535	0.2531
Precision	0.2538	**0.2639**	0.2534	0.2530
Recall	0.2539	**0.2621**	0.2536	0.2532
F1	**0.2538**	0.1991	0.2534	0.2530

Table (4.24) Results on train and test set for AG News Dataset using DES(CBC) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6726	0.6198	0.6993	**0.7054**
Precision	0.6733	0.6474	0.6993	**0.7054**
Recall	0.6725	0.6192	0.6993	**0.7054**
F1	0.6722	0.6004	0.6993	**0.7054**
Test				
Accuracy	0.5145	0.5107	**0.5233**	0.5185
Precision	0.5150	0.5151	**0.5235**	0.5186
Recall	0.5150	0.5127	**0.5235**	0.5186
F1	0.5141	0.4916	**0.5233**	0.5186

Table (4.25) Results on train and test set for Kaggle ISIS Dataset using DES(CBC) encryption

the test set. In table 4.27, the results for the News category can be seen. Logistic Regression with around 45 percent F1 score and Linear SVM with almost 43 Percent were able to classify slightly better than a random classification on the test set. However, Multinomial Naive Bayes and Random Forest results are not useful for classification since their F1 score is close to a random classification score. Also, just like the DES encryption adding a feature did not improve the classification results as it can be illustrated in table 4.28. In addition, by looking at table 4.29, it can be stated that Logistic Regression and Linear SVM with almost 33 and 32 percent and Multinomial Naive Bayes with around 32 percent F1 on the test set had a similar result which can be considered as a better result than random classification. On the other hand, random forest with around 29 percent F1 failed on classifying this dataset. The best results can be seen in table 4.30. Linear SVM and Logistic Regression F1 score on the test set is around 69 Percent. It is obvious that even for Multinomial Naive Bayes and Random Forest, the results are above average. The F1 score for these classifiers is around 62 percent and 58 percent, respectively.

7. AES(CBC) Encryption

Based on the definition, AES(CBC) can be considered as the most complicated encryption in this research, hence it is expected to havce lower scores for this dataset. Sentiment 140 in table 4.26, follows the same trend as the AES(ECB) encryption and all the classifiers are not able to classify the records correctly.

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6499	0.5697	0.6762	**0.6829**
Precision	0.6496	0.6459	0.6765	**0.6826**
Recall	0.6490	0.5583	0.6749	**0.6821**
F1	0.6491	0.4867	0.6748	**0.6822**
Test				
Accuracy	0.5068	**0.5125**	0.5112	0.5067
Precision	0.5060	0.5008	**0.5099**	0.5061
Recall	0.5060	0.5003	**0.5099**	0.5061
F1	0.5059	0.4189	**0.5096**	0.5061

Table (4.26) Results on train and test set for Sentiment 140 Dataset using AES(ECB) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.4015	0.4204	0.6657	**0.6780**
Precision	0.5222	0.6393	0.6635	**0.6760**
Recall	0.4024	0.4214	0.6658	**0.6782**
F1	0.3608	0.3844	0.6635	**0.6743**
Test				
Accuracy	0.3398	0.3705	**0.4508**	0.4352
Precision	0.4188	**0.4944**	0.4456	0.4275
Recall	0.3360	0.3664	**0.4501**	0.4343
F1	0.2846	0.3124	**0.4472**	0.4295

Table (4.27) Results on train and test set for News Category Dataset using AES(ECB) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.3253	0.4044	0.6397	**0.6686**
Precision	0.6202	0.5038	0.6413	**0.6687**
Recall	0.3253	0.4044	0.6397	**0.6686**
F1	0.2425	0.4004	0.6382	**0.6672**
Test				
Accuracy	0.2700	0.3458	**0.4240**	0.4088
Precision	**0.4710**	0.4162	0.4251	0.4072
Recall	0.2700	0.3456	**0.4241**	0.4089
F1	0.1853	0.3345	**0.4229**	0.4075

Table (4.28) Results on train and test set for News Category Dataset with additional feature using AES(ECB) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.5138	0.5248	0.6001	**0.6212**
Precision	0.5216	0.6609	0.6007	**0.6213**
Recall	0.5138	0.5243	0.6001	**0.6213**
F1	0.5124	0.5155	0.6002	**0.6213**
Test				
Accuracy	0.3218	0.3167	**0.3285**	0.3242
Precision	0.3281	0.3734	**0.3283**	0.3237
Recall	0.3218	0.3183	**0.3282**	0.3239
F1	0.3213	0.2892	**0.3282**	0.3237

Table (4.29) Results on train and test set for AG News Dataset using AES(ECB) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6643	0.6487	0.7794	**0.8030**
Precision	0.7391	0.7440	0.7813	**0.8032**
Recall	0.6644	0.6489	0.7794	**0.8030**
F1	0.6359	0.6109	0.7791	**0.8029**
Test				
Accuracy	0.6500	0.6192	**0.6917**	0.6847
Precision	**0.7201**	0.6944	0.6920	0.6847
Recall	0.6496	0.6182	**0.6916**	0.6847
F1	0.6194	0.5774	**0.6915**	0.6847

Table (4.30) Results on train and test set for Kaggle ISIS Dataset using AES(ECB) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6566	0.5777	0.6809	**0.6867**
Precision	0.6565	0.6405	0.6814	**0.6865**
Recall	0.6566	0.5664	0.6793	**0.6859**
F1	0.6565	0.5070	0.6793	**0.6860**
Test				
Accuracy	0.5032	**0.5200**	0.5023	0.5017
Precision	0.5030	**0.5239**	0.5010	0.5010
Recall	0.5030	**0.5109**	0.5010	0.5010
F1	**0.5030**	0.4386	0.5002	0.5009

Table (4.31) Results on train and test set for Sentiment 140 Dataset using AES(CBC) encryption

Moving to the next dataset, we can see News Category Dataset results in table 4.32. All the classifiers had a better than average result on the test set. LogisticRegression with around 40 percent F1 score had the highest results, while Multi-nomial Naive Bayes, with almost 30 percent, had the lowest. Moreover, Random forest with 34 percent and Linear SVM with around 38 percent F1 were also tested. Similar to the previous encryption, we can see in figure 4.33 that adding the feature did not have a positive effect on classifying the News Category dataset. The classification results of AG News are shown in table 4.34. This dataset's results are basically the same as a random classification for this encryption because all the classifiers had similar results of 25 percent or less F1 score on the test set. In table 4.35, we can see unless the AES(ECB), the classifiers fail the classification task. With around 50 percent F1 score for all the classifiers, it can be said that the results are very close to random in this case.

4.2 Neural Networks

In another experiment, I examined the result of Neural Network on the dataset that had the best classification score so far, which is the Kaggle ISIS dataset. I did this experiment to check if using neural networks with a single hidden layer will improve the results that I already achieved using linear classifiers or not. For this task, I used three different neural networks for each encryption. The networks have one layer with

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.5545	0.5844	0.6709	**0.6785**
Precision	0.5540	0.6575	0.6686	**0.6755**
Recall	0.5543	0.5841	0.6706	**0.6780**
F1	0.5540	0.5650	**0.6675**	0.6738
Test				
Accuracy	0.2968	0.3820	**0.4070**	0.3868
Precision	0.2966	**0.4093**	0.4006	0.3776
Recall	0.2976	0.3838	**0.4089**	0.3890
F1	0.2968	0.3418	**0.4034**	0.3813

Table (4.32) Results on train and test set for News Category Dataset using AES(CBC) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.5249	0.3681	0.6157	**0.6365**
Precision	0.5250	0.5516	0.6137	**0.6344**
Recall	0.5249	0.3671	0.6158	**0.6366**
F1	0.5249	0.3058	0.6131	**0.6340**
Test				
Accuracy	0.2595	0.3122	0.3573	**0.3585**
Precision	0.2592	**0.3734**	0.3496	0.3520
Recall	0.2592	0.3156	0.3569	**0.3580**
F1	0.2591	0.2390	0.3520	**0.3541**

Table (4.33) Results on train and test set for News Category Dataset with additional feature using AES(CBC) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.5268	0.5775	0.5573	**0.5694**
Precision	0.5269	0.5928	0.5573	**0.5694**
Recall	0.5267	0.5774	0.5573	**0.5694**
F1	0.5267	0.5783	0.5573	**0.5694**
Test				
Accuracy	0.2493	**0.2598**	0.2487	0.2532
Precision	0.2498	**0.2602**	0.2488	0.2534
Recall	0.2495	**0.2601**	0.2488	0.2532
F1	0.2494	**0.2564**	0.2487	0.2532

Table (4.34) Results on train and test set for AG News Dataset with using AES(CBC) encryption

Classifier	**Multinomial NB**	**Random Forest**	**Logistic Regression**	**Linear SVM**
Train				
Accuracy	0.6780	0.6365	0.7000	**0.7067**
Precision	0.6783	0.6482	0.7000	**0.7067**
Recall	0.6779	0.6373	0.6999	**0.7067**
F1	0.6778	0.6301	0.6999	**0.7067**
Test				
Accuracy	0.5038	**0.5333**	0.5255	0.5235
Precision	0.5043	**0.5325**	0.5255	0.5235
Recall	0.5043	**0.5304**	0.5255	0.5235
F1	0.5037	0.5243	**0.5254**	0.5234

Table (4.35) Results on train and test set for Kaggle ISIS Dataset using AES(CBC) encryption

100, 300, and 600 nodes. The results show the Confusion Matrix of each network, along with the Learning Curve. I split the dataset for this task into 70 percent train, 20 percent test, and 10 percent validation to get the highest performance and used the whole dataset for a test. In addition, I set the maximum number of epochs to 200.

1. Plaintext

   As it can be seen from figure 4.1, the results of using a neural network with 100 nodes on the Kaggle Isis dataset are shown. The classifier was able to predict around 94 percent of the records correctly. Although the accuracy is high but compared to Logistic regression and Linear SVM, which has covered before, it is down by almost 4 percent. In addition, by increasing the number of nodes to 300 in figure 4.2, it can be seen that not only did it not improve the accuracy score, but also it dropped by 0.2 percent. Also, by looking at figure 4.3, the same trend is obvious, and the accuracy score is around 90 percent. Based on these results, it is obvious that increasing the number of nodes did not improve the results.

2. Caesar Encryption

   In figure 4.4, the results of a network with 100 nodes on Caesar encryption can be seen. This network was able to get around 92 percent accuracy on the test set. By increasing the number of nodes to 300 in figure 4.5, we can see an

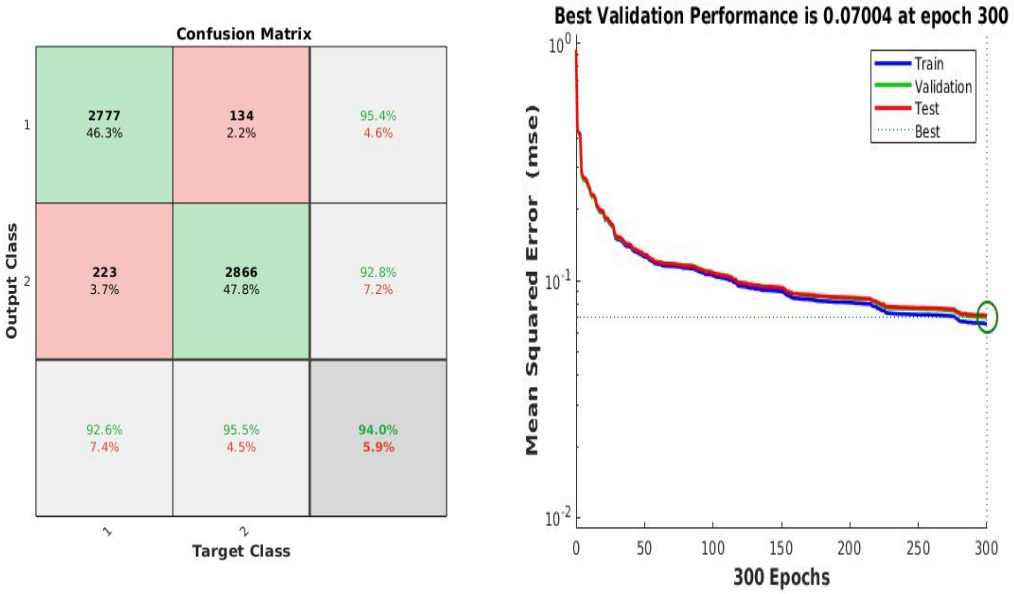


Figure (4.1) Kaggle Isis dataset plaintext confusion matrix (left) and learning curve (right) on neural networks with 100 nodes

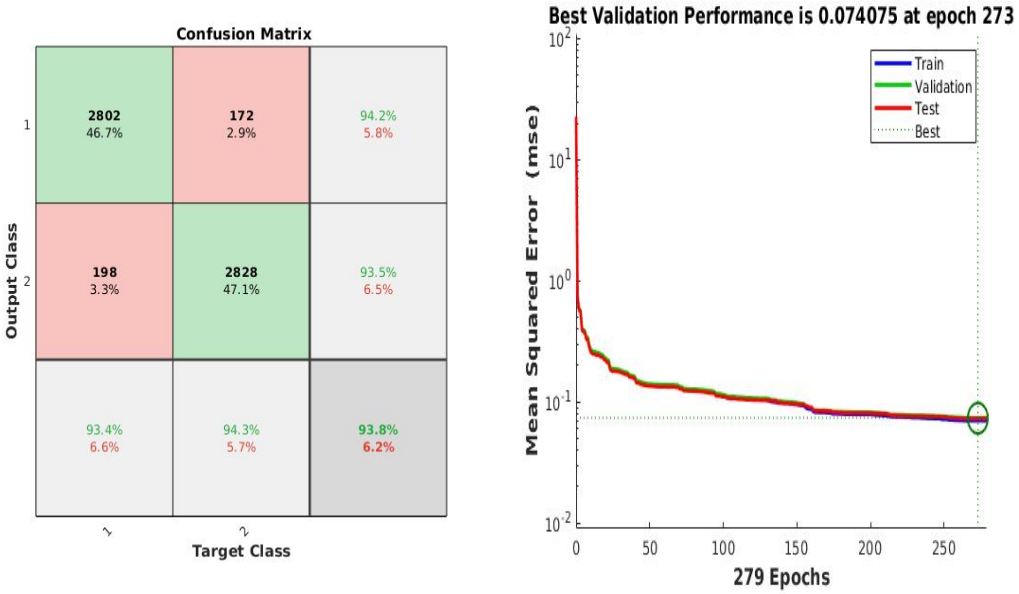


Figure (4.2) Kaggle Isis dataset plaintext confusion matrix (left) and learning curve (right) on neural networks with 300 nodes

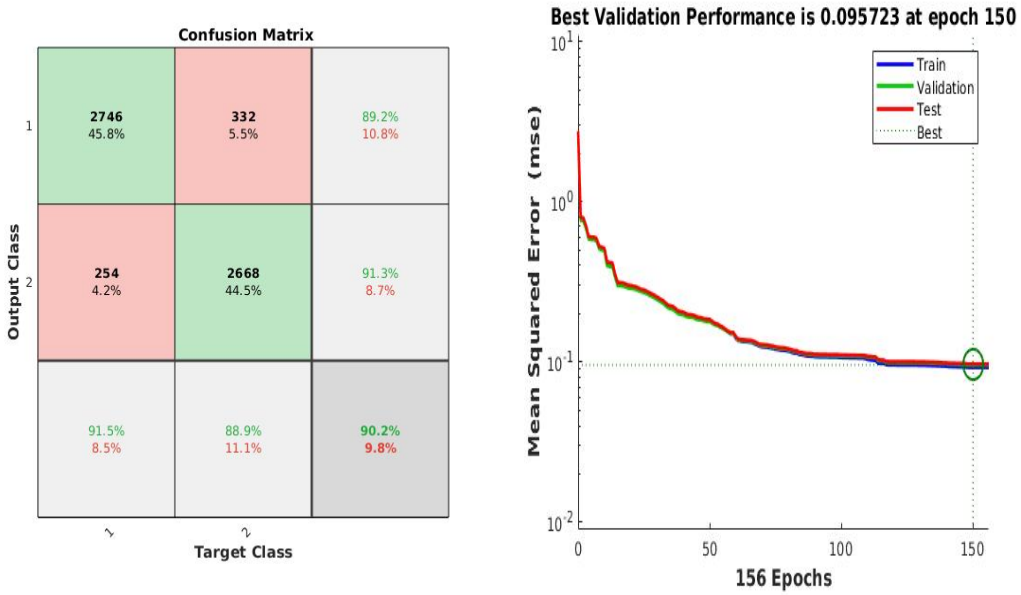


Figure (4.3) Kaggle Isis dataset plaintext confusion matrix (left) and learning curve (right) on neural networks with 600 nodes

increase in the accuracy score by 1 percent. However, increasing the number of nodes did not have any effect on the accuracy score in figure 4.6.

3. Base64 encryption

   Using the Base64 encryption in the neural networks experiment, it is evident that the best results belong to the network with 100 nodes in 4.7. Accuracy of this network on the test data is 93 percent. In addition, the accuracy for the networks with 300 nodes and 600 nodes are around 90 and 89 percent, respectively which can be seen in figures 4.8, 4.9. This means that increasing the number of features did not have a positive effect on the results for Base64 encryption.

4. DES(ECB) Encryption

   By a look at figure 4.10, it can be seen that using the DES(ECB) encryption with the 100 nodes network, it was able to achieve an accuracy score of 65 percent. In the next part for the 300 nodes network in figure 4.11, the accuracy is around 1 percent less than the previous network. However, an increase in the accuracy score can be seen for the 600 nodes network in figure 4.12. This network with 64.5 percent accuracy did a better job than the 300 nodes network

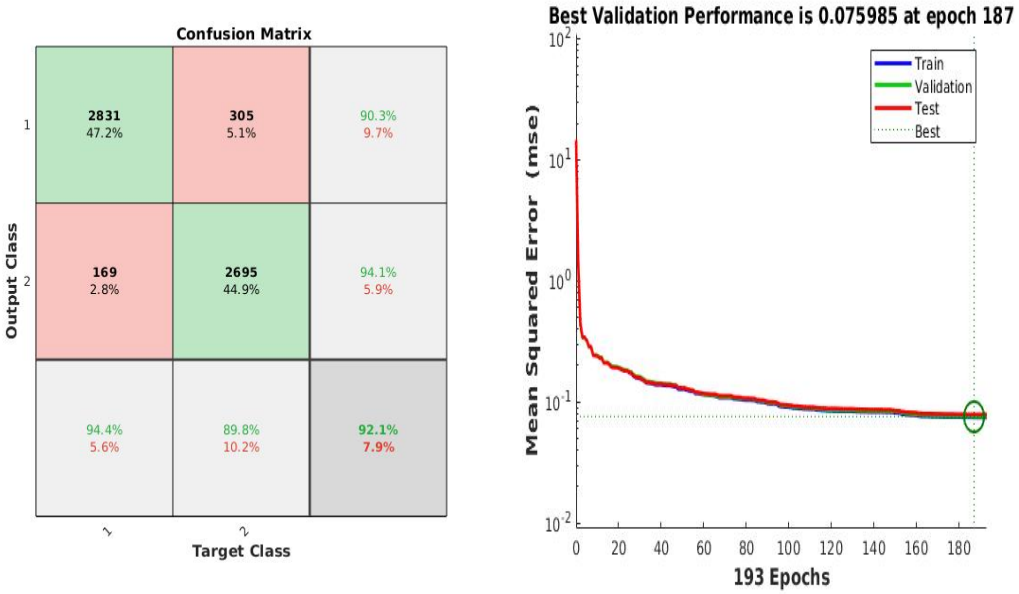


Figure (4.4) Kaggle Isis dataset Caesar confusion matrix (left) and learning curve (right) on neural networks with 100 nodes

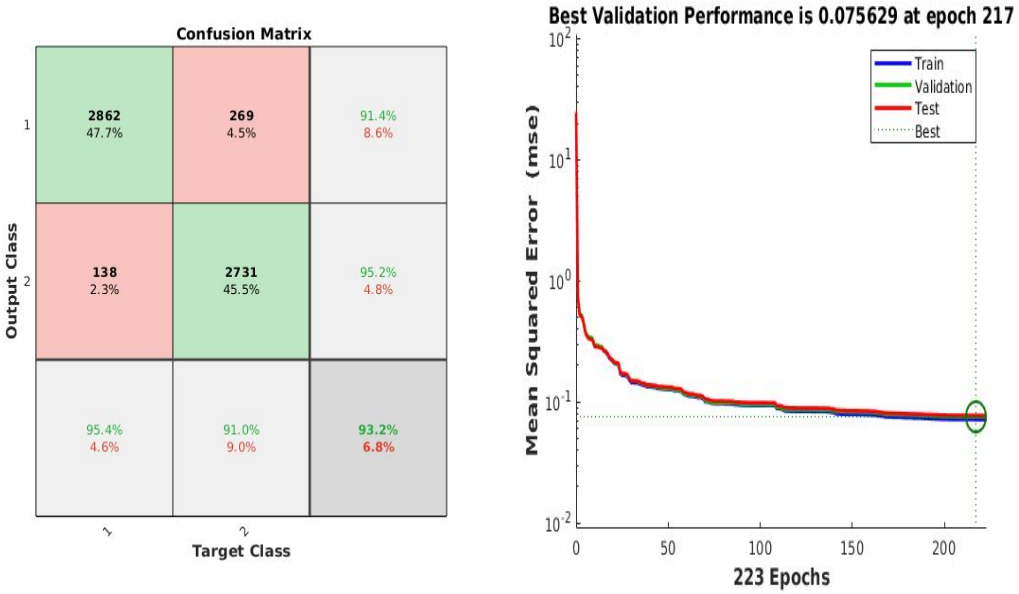


Figure (4.5) Kaggle Isis dataset Caesar confusion matrix (left) and learning curve (right) on neural networks with 300 nodes

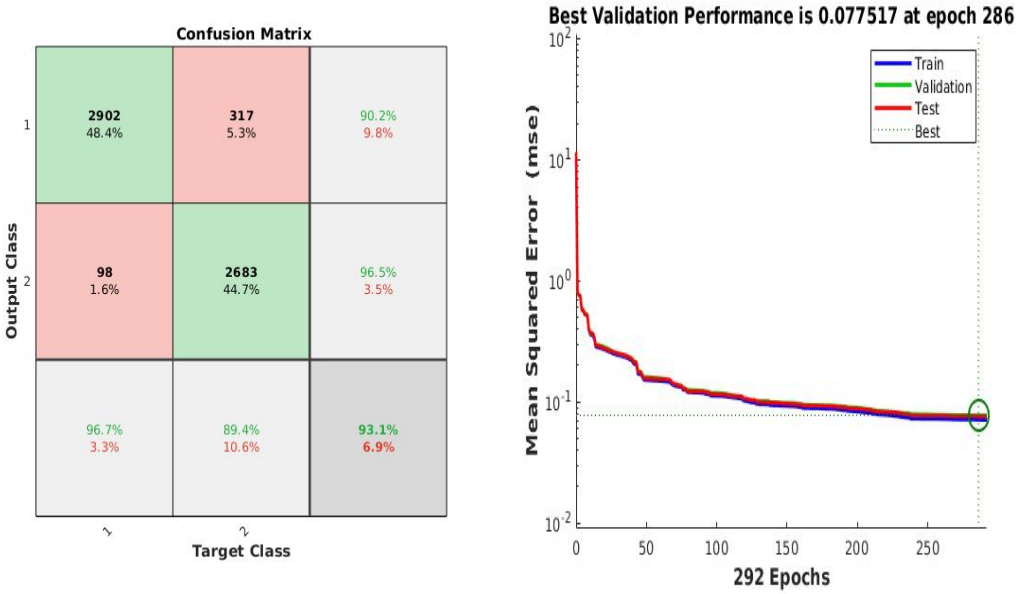


Figure (4.6) Kaggle Isis dataset Caesar confusion matrix (left) and learning curve (right) on neural networks with 600 nodes

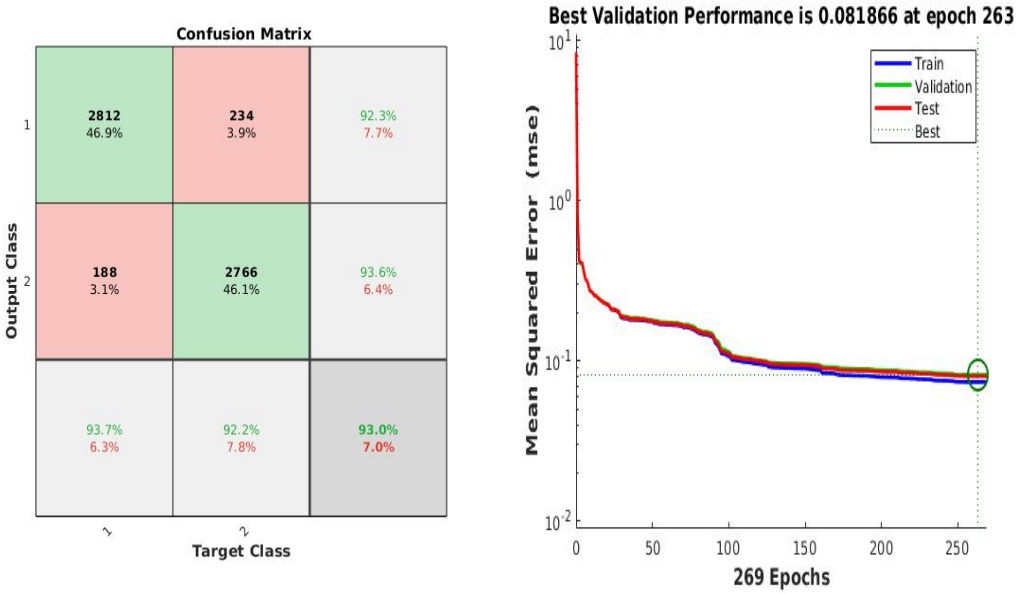


Figure (4.7) Kaggle Isis dataset Base64 confusion matrix (left) and learning curve (right) on neural networks with 100 nodes

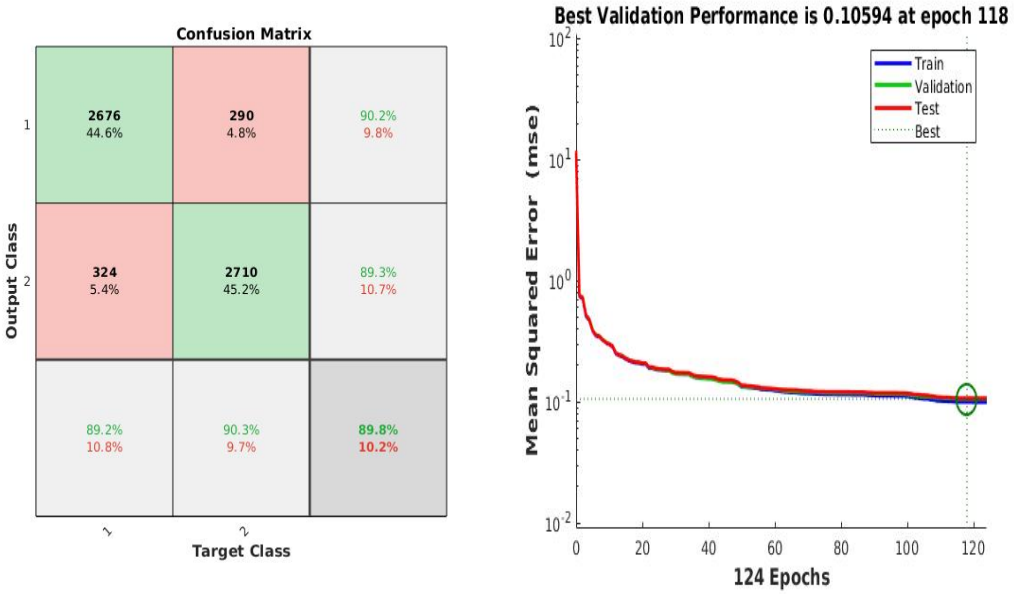


Figure (4.8) Kaggle Isis dataset Base64 confusion matrix (left) and learning curve (right) on neural networks with 300 nodes

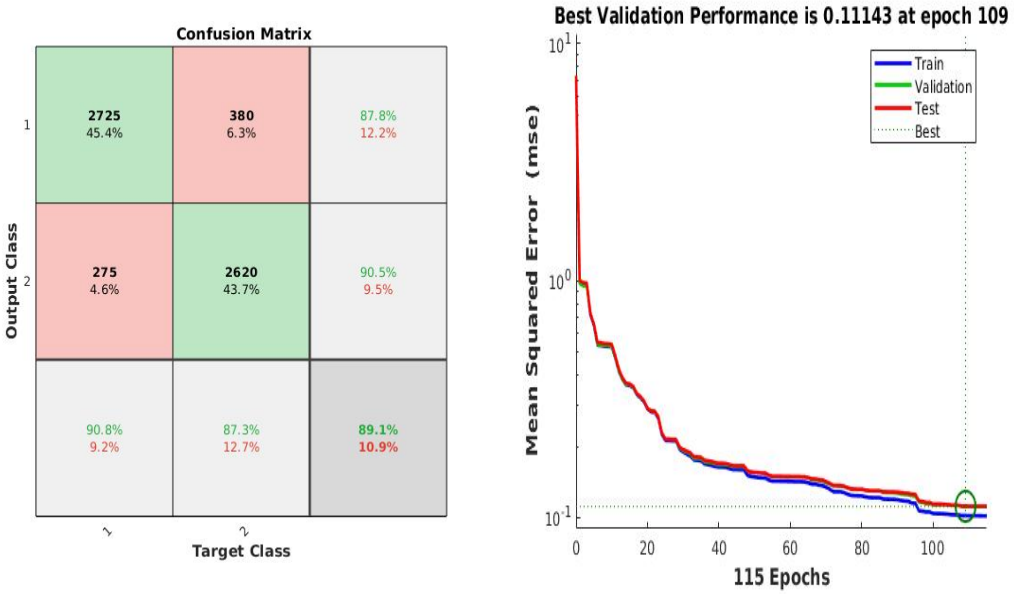


Figure (4.9) Kaggle Isis dataset Base64 confusion matrix (left) and learning curve (right) on neural networks with 600 nodes

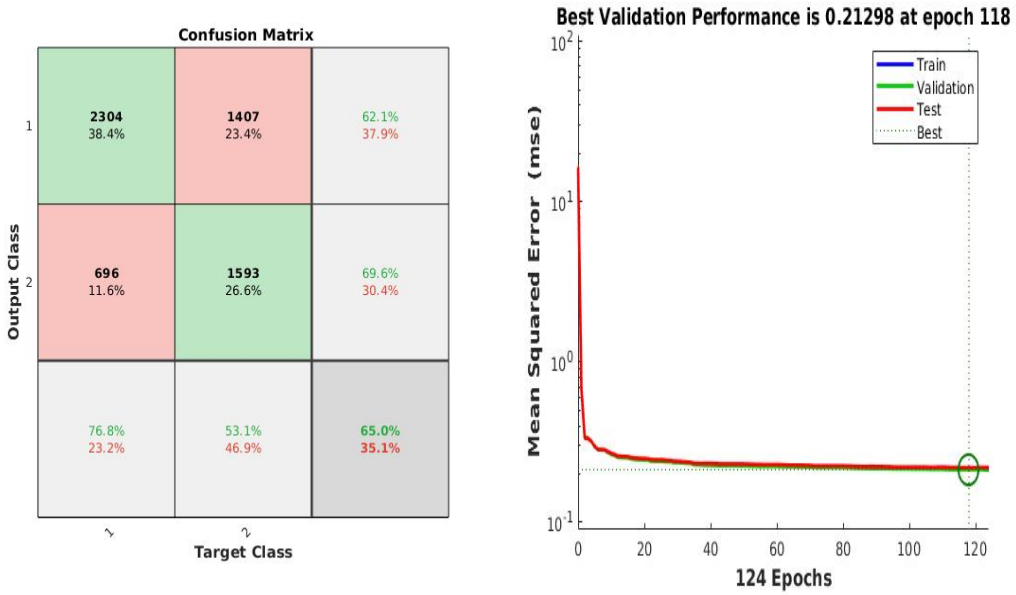


Figure (4.10) Kaggle Isis dataset DES(ECB) confusion matrix (left) and learning curve (right) on neural networks with 100 nodes

slightly, but it is still less than the first network.

5. DES(CBC) Encryption

   Moving to one of the more complicated encryptions, the results are dropping and they are close to a random classification as expected. In figure 4.13, we can see the classification accuracy score on the test data is around 55 percent. This score was almost the same for the 300 nodes and 600 nodes network with 55.5 percent and 55.3 accuracy score. They both are shown in the figures 4.14 and 4.15.

6. AES(ECB) Encryption

   For this encryption, the best accuracy score on the test set can be seen in figure 4.18, using the network with 600 nodes. In figure 4.16, the network with 100 nodes had and accuracy of 61.5 percent which was slightly better than the 300 nodes network with 59.9 percent accuracy score whcih can be seen in figure 4.17.

7. AES(CBC) Encryption

   The same trend as DES(CBC) encryption for the neural network can be seen

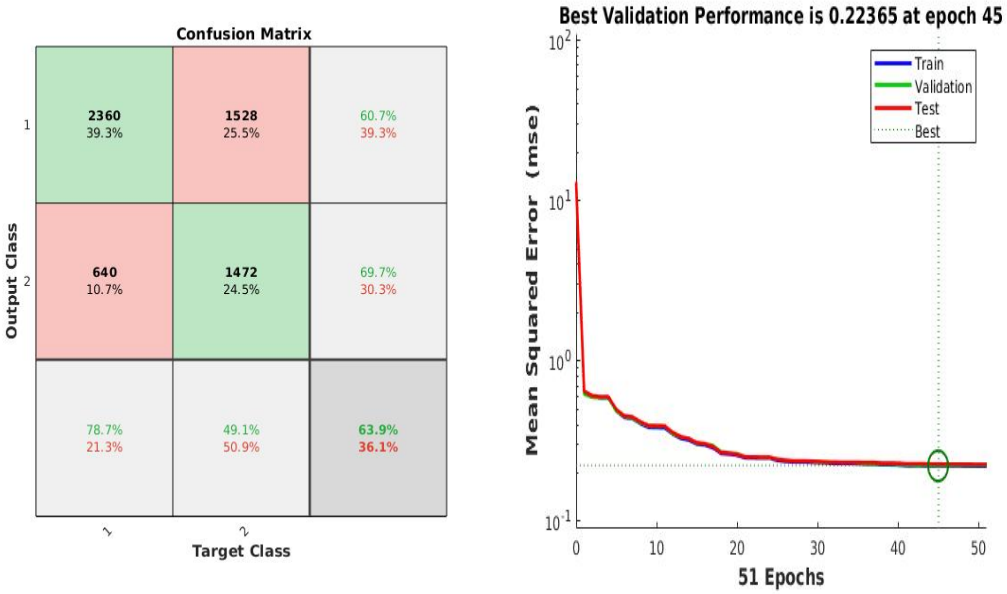


Figure (4.11) Kaggle Isis dataset DES(ECB) confusion matrix (left) and learning curve (right) on neural networks with 300 nodes

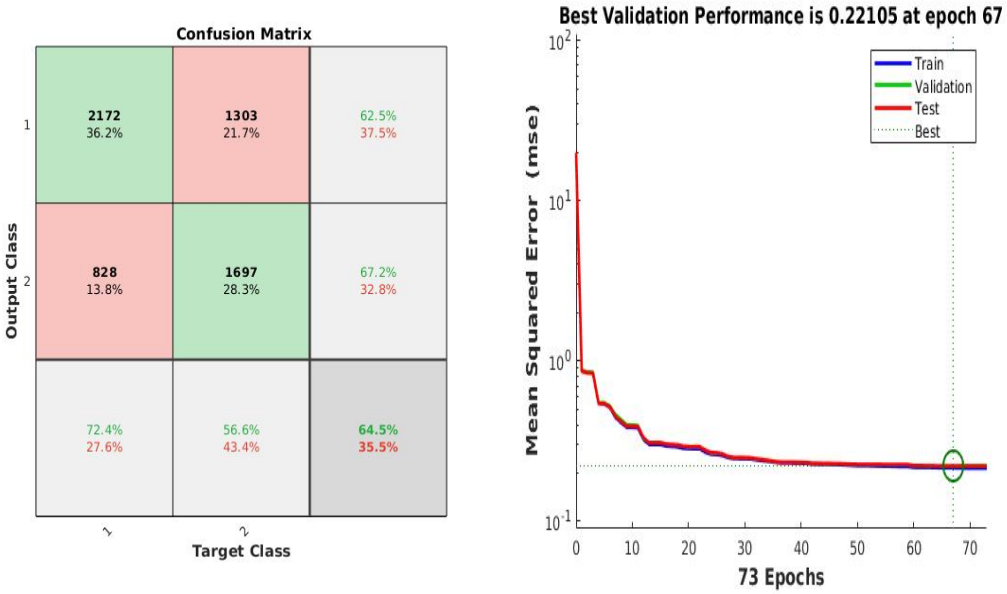


Figure (4.12) Kaggle Isis dataset DES(ECB) confusion matrix (left) and learning curve (right) on neural networks with 600 nodes

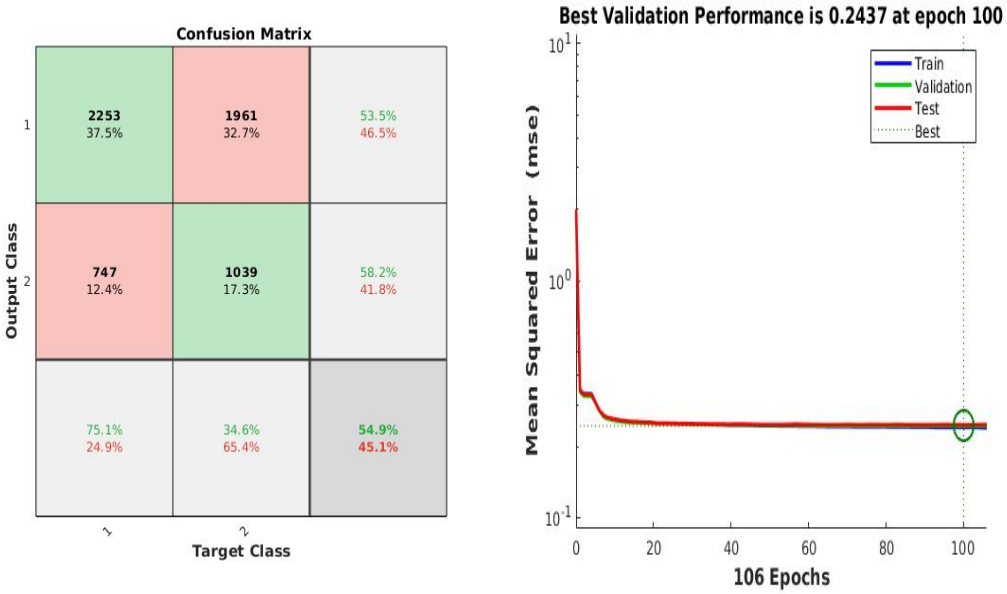


Figure (4.13) Kaggle Isis dataset DES(CBC) confusion matrix (left) and learning curve (right) on neural networks with 100 nodes

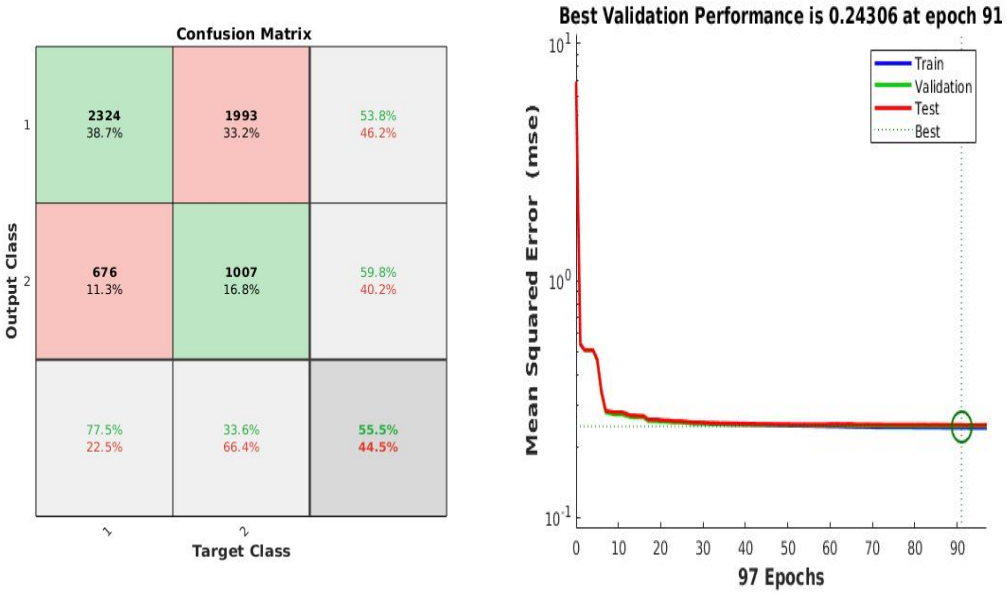


Figure (4.14) Kaggle Isis dataset DES(CBC) confusion matrix (left) and learning curve (right) on neural networks with 300 nodes

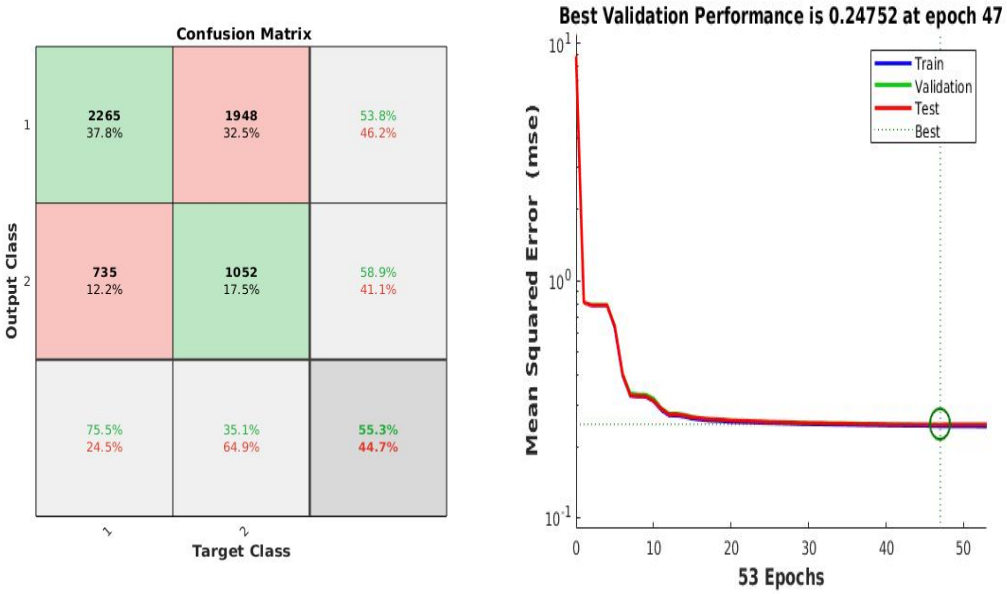


Figure (4.15) Kaggle Isis dataset DES(CBC) confusion matrix (left) and learning curve (right) on neural networks with 600 nodes

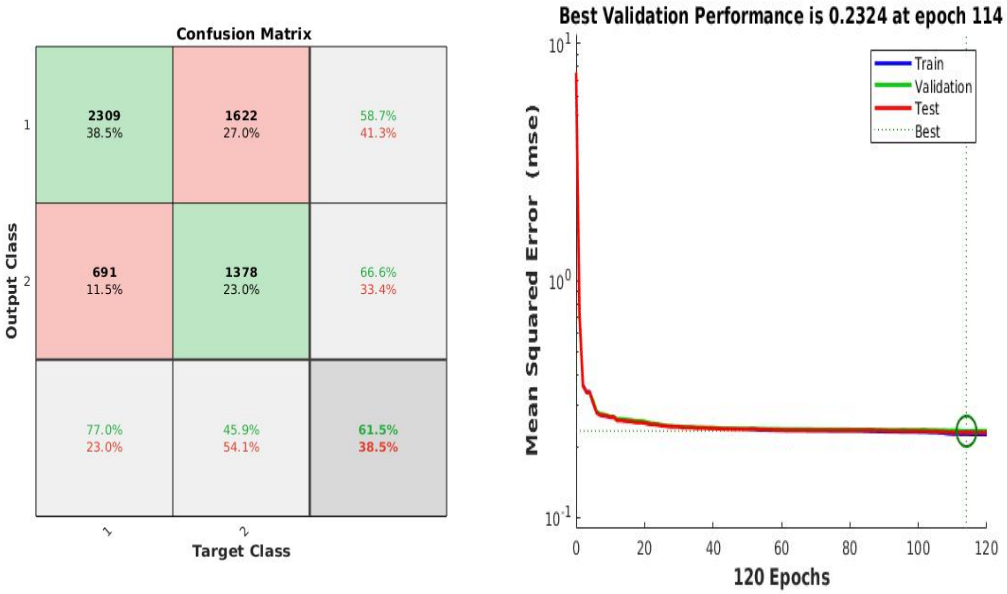


Figure (4.16) Kaggle Isis dataset AES(ECB) confusion matrix (left) and learning curve (right) on neural networks with 100 nodes

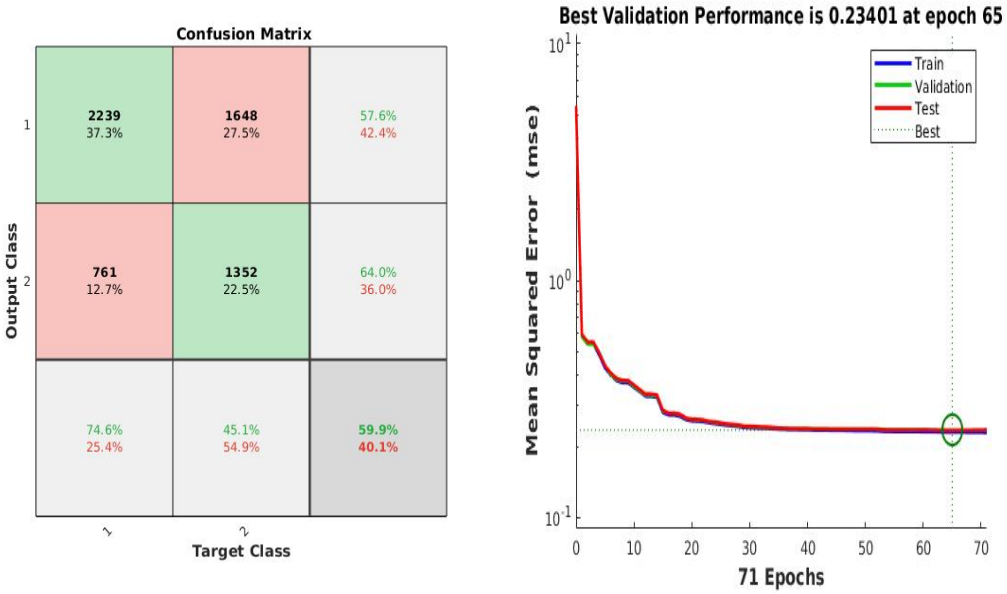


Figure (4.17) Kaggle Isis dataset AES(ECB) confusion matrix (left) and learning curve (right) on neural networks with 300 nodes

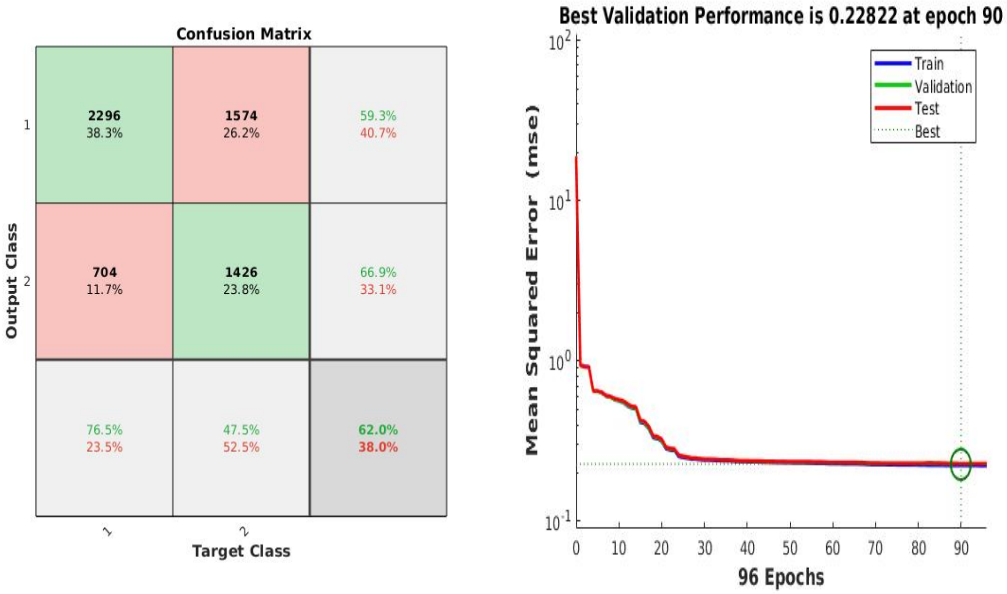


Figure (4.18) Kaggle Isis dataset AES(ECB) confusion matrix (left) and learning curve (right) on neural networks with 600 nodes

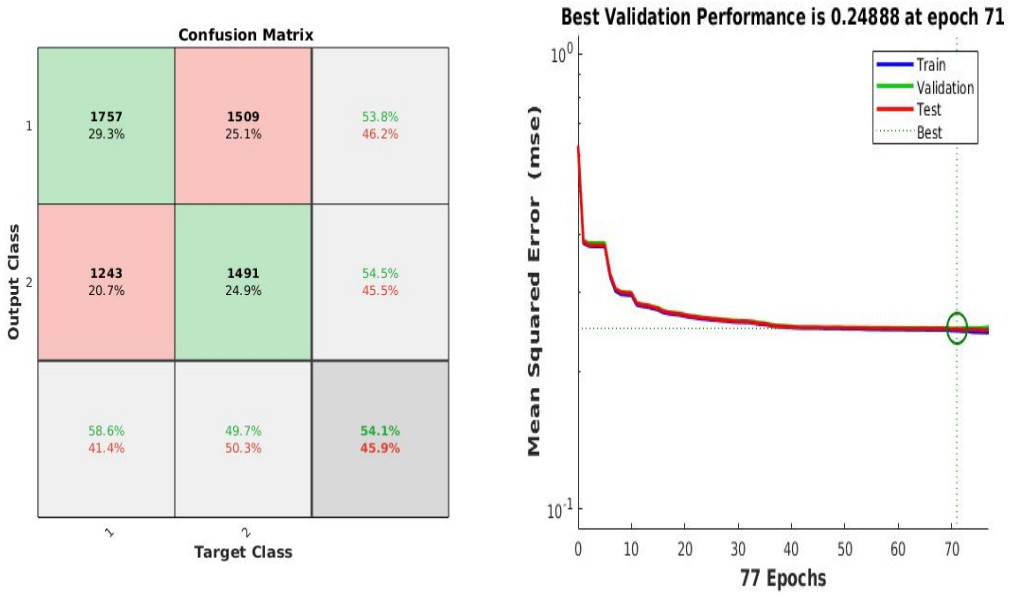


Figure (4.19) Kaggle Isis dataset AES(CBC) confusion matrix (left) and learning curve (right) on neural networks with 100 nodes

for AES(CBC) as well. Experimenting on the network with 100 nodes in figure 4.19, the accuracy score is around 54 percent. For this encryption, increasing the number of nodes had a positive effect slightly on the accuracy score. In figure 4.20, the accuracy of the 300 nodes network is 54.6 percent, and for the 600 nodes network in figure 4.14, the accuracy score is around 56 percent.

From the observations that has been covered so far, it seems like using neural networks did not help with the classification task since the results of the linear classifiers are better than the neural networks classification results. Depending on the encryption method, on average the accuracy score was almost 5 percent less than the best linear model. Even with increasing the number of nodes in order to make a more complicated network, the scores are still less and not comparable to the linear models.

4.3 Summary

In this section, I developed the method of different types of encryption. The results are shown for different classifiers, namely, Multinomial Naive Bayes, Random Forest, Logistic Regression, Linear SVM, and Neural Networks. For the first four classifiers,

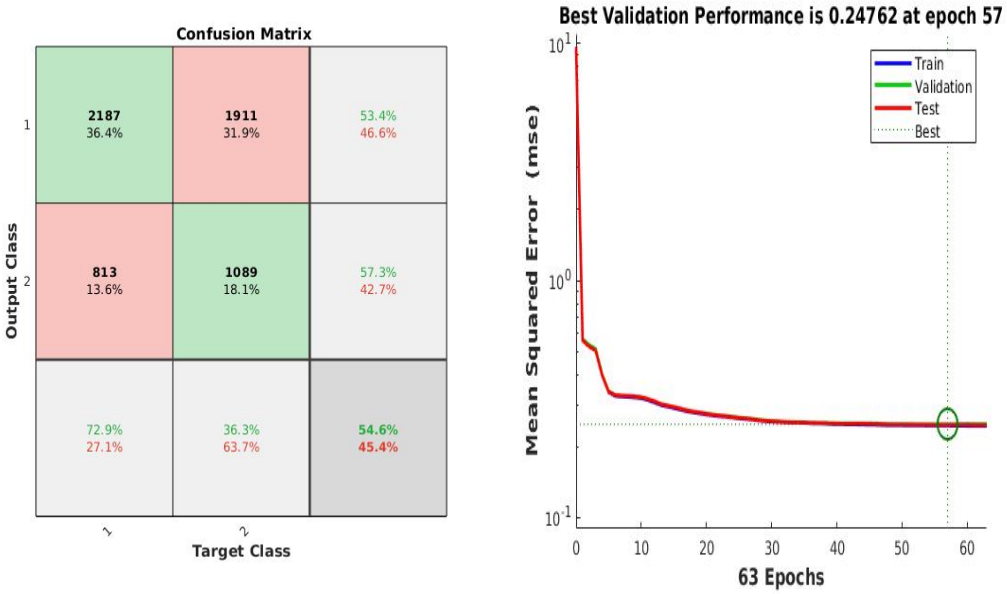


Figure (4.20) Kaggle Isis dataset AES(CBC) confusion matrix (left) and learning curve (right) on neural networks with 300 nodes

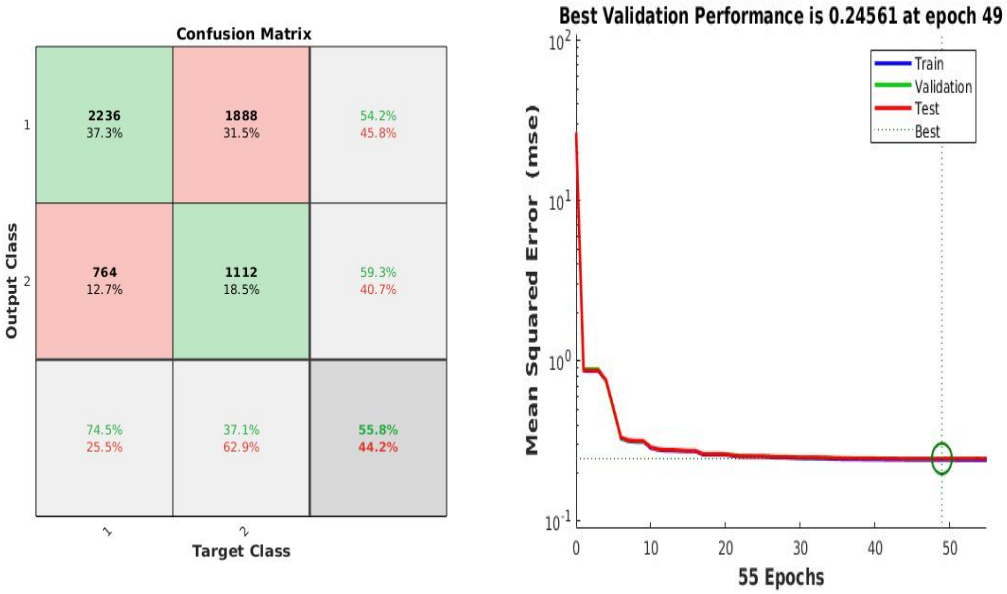


Figure (4.21) Kaggle Isis dataset AES(CBC) confusion matrix (left) and learning curve (right) on neural networks with 600 nodes

I gathered the results for both train and test set using accuracy, precision, recall, and f1 score. Based on the results, it is apparent that most of the encryption algorithms had some sort of data leakage. For the classic encryption methods, this approach is very successful since the result of Caesar, and Base64 encryptions were almost the same as plaintext. Also, for the modern encryptions, namely AES and DES, it can be seen that there is still some leakage when the ECB method is used. However, using the CBC method for the last two encryptions, the results are almost near the baseline (random classification), so it can be stated that the approach is not very effective for this type of encryption. In the part of the neural networks, I first attempted to get the best performance using train, test, and validation and saved the model. Then I developed the model on the test data to get the confusion matrix and shown them for each encryption method. Neural networks are also following the same trend as previous classifiers. However, the results are not as promising as the ones that I have shown before. The results are shown as histograms in the appendix A.

Chapter 5

Conclusion and future work

In this thesis, my goal is to explore data leakage in encrypted data. I built a supervised learning, classification, based model that aims to find the related topic of each ciphertext, which is encoded using one of the four aforementioned encryption algorithms. It is a simple approach that analyzes cypertext in order to explore any data leakage in the encrypted data. Evaluations show that data leakage decreases as the encryption algorithms get stronger, even though the data leakage is never zero percent.

5.1 Conclusion

The proposed data analysis model could be implemented quickly and does not need any prior knowledge of the encryption algorithms or the datasets. As I mentioned earlier, the pre-processing phase of the ciphertext is not the same as plaintext because the traditional NLP techniques could not be implemented on encrypted data. Hence, the pre-processing part of this research only consists of extracting the features using character n-grams and representing them using TF-IDF. Moreover, following machine learning literature, training datasets (Kaggle Sentiment 140, Kaggle Newsgroup, Kaggle Isis, AG News) are balanced to minimize any data related biases. Additionally, I encoded the data using the same key for each encryption and then fed the training data to the classification algorithms (Multinomial Naive Bayes, Random Forest, Logistic Regression, and Linear SVM). Based on the range of n-grams and the TF-IDF scores, the goal here was to explore whether the classifier would be able to find a pattern on the ciphertext that could generalize and use it for the unseen data to predict the related topic (class) of that ciphertext. Furthermore, to discover the effect of deep learning, I implemented a feedforward neural network with a single hidden layer and different number of neurons (100, 300, 600) and presented the results. The proposed model can be used with any other encryption algorithm on any other text

classification dataset. Finally, based on the results, it can be stated that:

- The model is able to find the leakage successfully using the classic encryption algorithms.
- The performance of the model on the modern encryption algorithms DES and AES is above average, which means it could still find some leakage using these algorithms. However, if these encryption algorithms use CBC mode, data leakage decreases, i.e. the model is not able to identify the topic correctly.
- Using a neural network with the specified features did not improve the results. However, the results achieved with this method were following the same trend as the other classification algorithms.
- These results are gathered from various datasets which are both binary and multiclass and all of them have the same trend in terms of the results. Thus, this model can be used to test the strength of the encryption algorithm.

5.2 Future Works

The following summarizes the potential future work directions:

- One of the limitations that I encountered through this research was finding an accurate labelled dataset. There are few publicly available datasets for the purpose of encrypted data leakage analysis. Thus, generating labelled datasets for this purpose would be an interesting direction to pursue.
- Further analysis on the structure of the data could also be an area that could shed more light into the analysis of data leakage on encrypted data. For this model I tried a fixed key size for each encryption however, this work can be extended to stream ciphers which will use a byte at a time to encrypt instead of a block of bytes. Also, using different key sizes can be explored.
- Neural Networks are a wide area to explore for any classification task. So, by using a more complex neural network, there may be a chance of achieving a higher score.

- Another area to study is the further analysis of parameter tuning and sensitivity of the different classifiers including deep learning that are used in this research.

Appendix A

Plots

A.1 Plaintext Results

A.2 Caesar encryption results

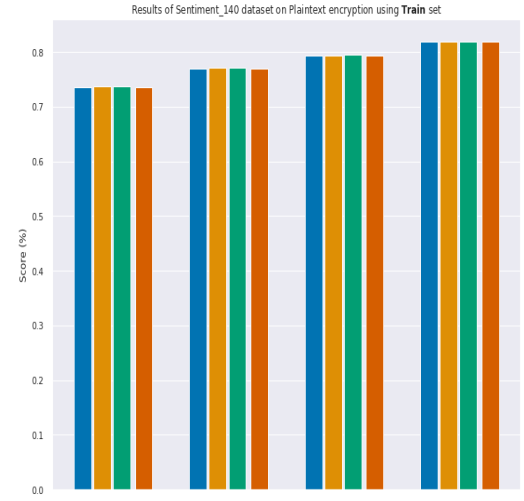
A.3 Base64 encryption results

A.4 DES(ECB) encryption results

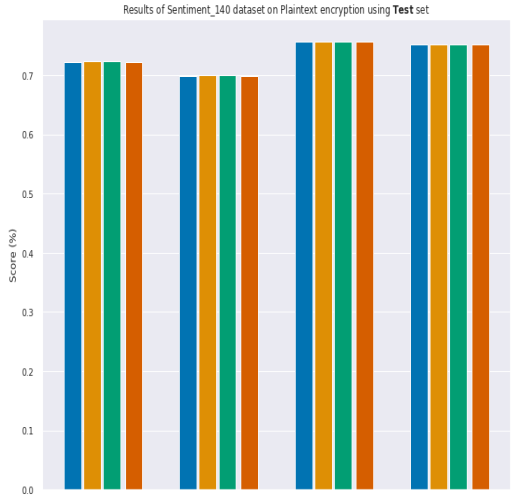
A.5 DES(CBC) encryption results

A.6 AES(ECB) encryption results

A.7 AES(CBC) encryption results

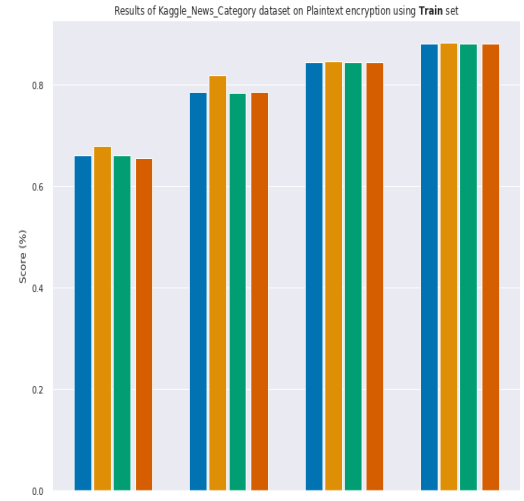


	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.7363	0.7693	0.7942	0.8194
Precision	0.7377	0.7708	0.7941	0.8192
Recall	0.7373	0.7704	0.7943	0.8194
F1	0.7363	0.7693	0.7941	0.8193

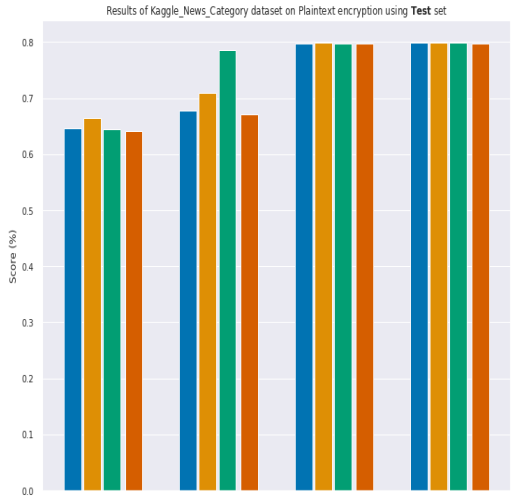


	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.7231	0.6995	0.7573	0.7527
Precision	0.7245	0.7002	0.7575	0.7528
Recall	0.7241	0.7002	0.7577	0.753
F1	0.723	0.6995	0.7573	0.7526

Figure (A.1) Sentiment 140 dataset Plaintext results on train and test

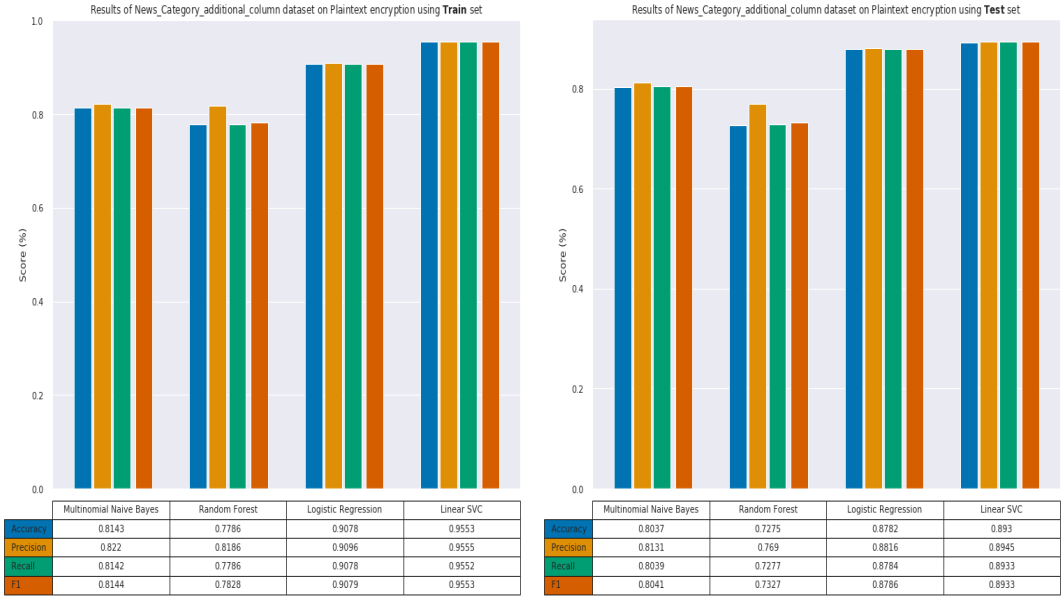


	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6603	0.7847	0.8439	0.8818
Precision	0.6783	0.8178	0.8455	0.8833
Recall	0.6605	0.7845	0.8438	0.8817
F1	0.656	0.785	0.8437	0.8817



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6458	0.6773	0.7977	0.798
Precision	0.6647	0.7092	0.7991	0.7993
Recall	0.6449	0.785	0.7978	0.7982
F1	0.6403	0.6714	0.7975	0.7979

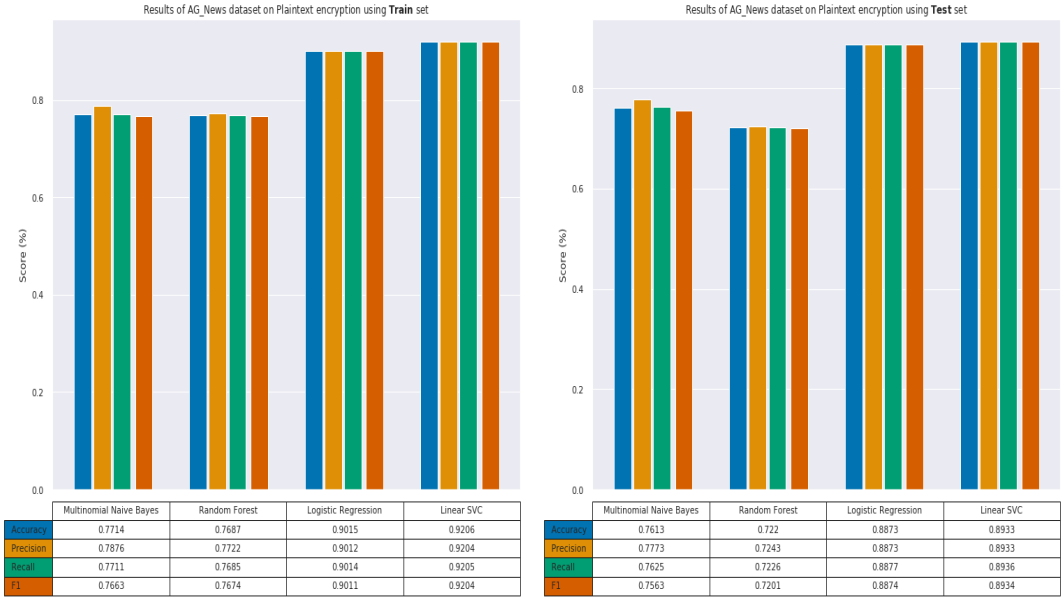
Figure (A.2) Kaggle News Category dataset Plaintext results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.8143	0.7786	0.9078	0.9553
Precision	0.822	0.8186	0.9096	0.9555
Recall	0.8142	0.7786	0.9078	0.9552
F1	0.8144	0.7828	0.9079	0.9553

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.8037	0.7275	0.8782	0.893
Precision	0.8131	0.769	0.8816	0.8945
Recall	0.8039	0.7277	0.8784	0.8933
F1	0.8041	0.7327	0.8786	0.8933

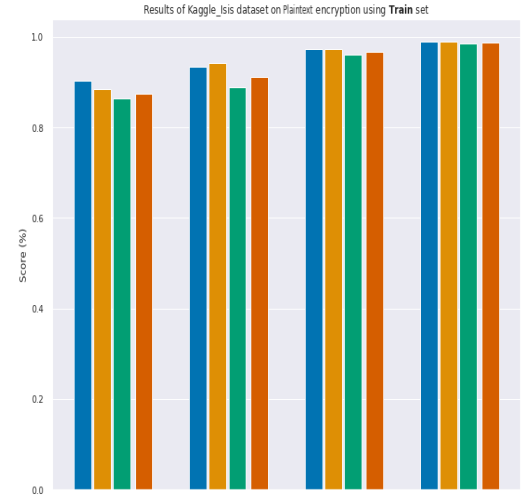
Figure (A.3) Kaggle News Category with additional feature dataset, Plaintext results on train and test



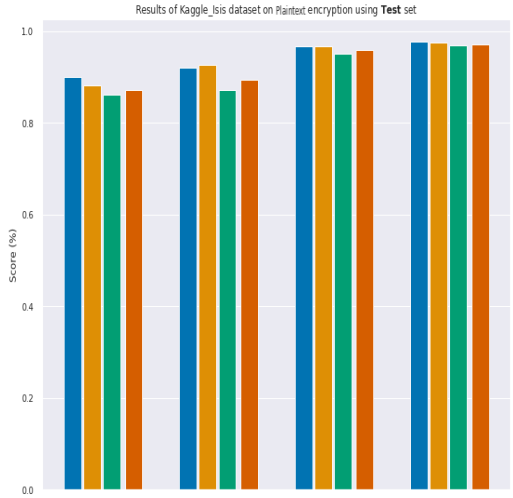
	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.7714	0.7687	0.9015	0.9206
Precision	0.7876	0.7722	0.9012	0.9204
Recall	0.7711	0.7685	0.9014	0.9205
F1	0.7663	0.7674	0.9011	0.9204

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.7613	0.722	0.8873	0.8933
Precision	0.7773	0.7243	0.8873	0.8933
Recall	0.7625	0.7226	0.8877	0.8936
F1	0.7563	0.7201	0.8874	0.8934

Figure (A.4) AG News Plaintext dataset results on train and test

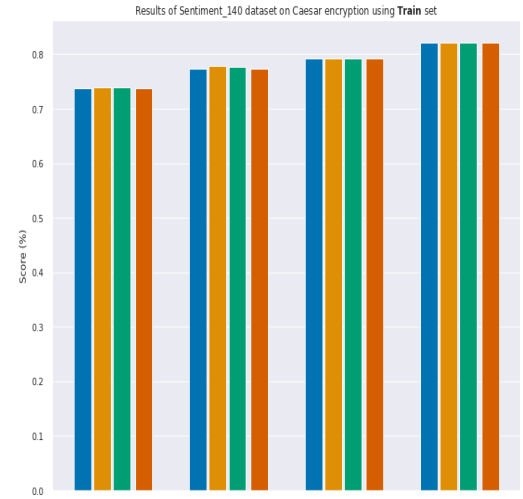


	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.9031	0.934	0.9736	0.9897
Precision	0.8843	0.9417	0.9727	0.9889
Recall	0.8636	0.8887	0.9597	0.9849
F1	0.8731	0.9109	0.966	0.9869

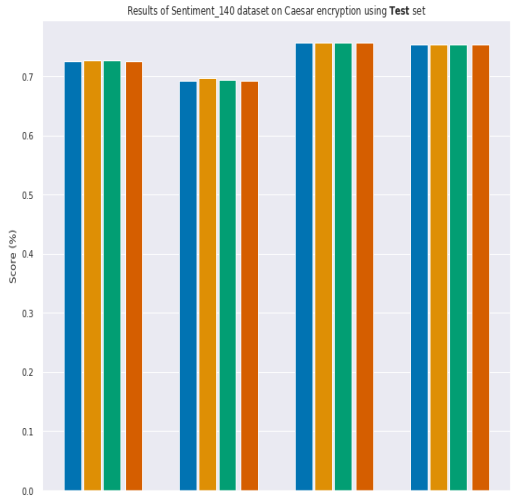


	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.899	0.9201	0.9678	0.9778
Precision	0.8825	0.926	0.9672	0.9744
Recall	0.861	0.8712	0.9518	0.9699
F1	0.8708	0.8936	0.9591	0.9721

Figure (A.5) Kaggle Isis dataset Plaintext results on train and test

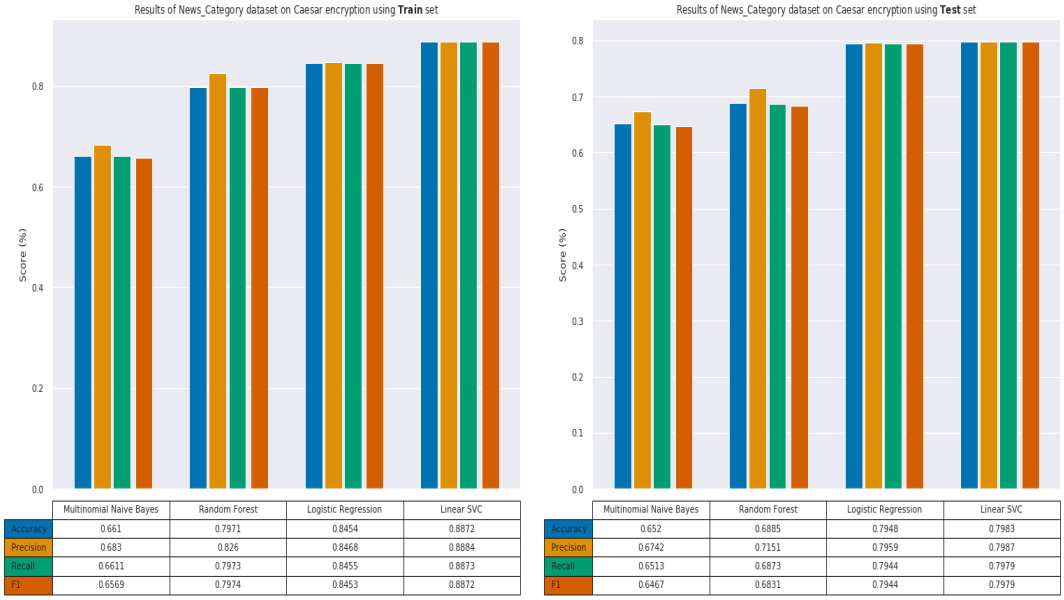


	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.7383	0.7741	0.7919	0.8215
Precision	0.7397	0.7792	0.7918	0.8214
Recall	0.7392	0.7759	0.792	0.8216
F1	0.7383	0.7737	0.7918	0.8214



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.7253	0.692	0.7567	0.7531
Precision	0.727	0.6962	0.7566	0.7529
Recall	0.7267	0.6943	0.757	0.7532
F1	0.7253	0.6916	0.7566	0.753

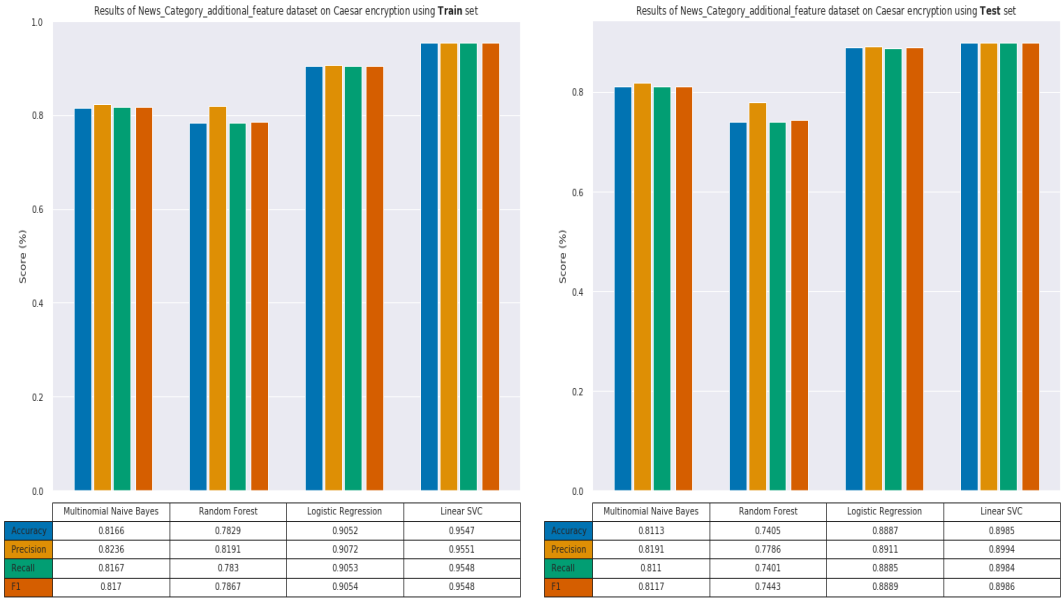
Figure (A.6) Sentiment 140 dataset Caesar encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.661	0.7971	0.8454	0.8872
Precision	0.683	0.826	0.8468	0.8884
Recall	0.6611	0.7973	0.8455	0.8873
F1	0.6569	0.7974	0.8453	0.8872

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.652	0.6885	0.7948	0.7983
Precision	0.6742	0.7151	0.7959	0.7987
Recall	0.6513	0.6873	0.7944	0.7979
F1	0.6467	0.6831	0.7944	0.7979

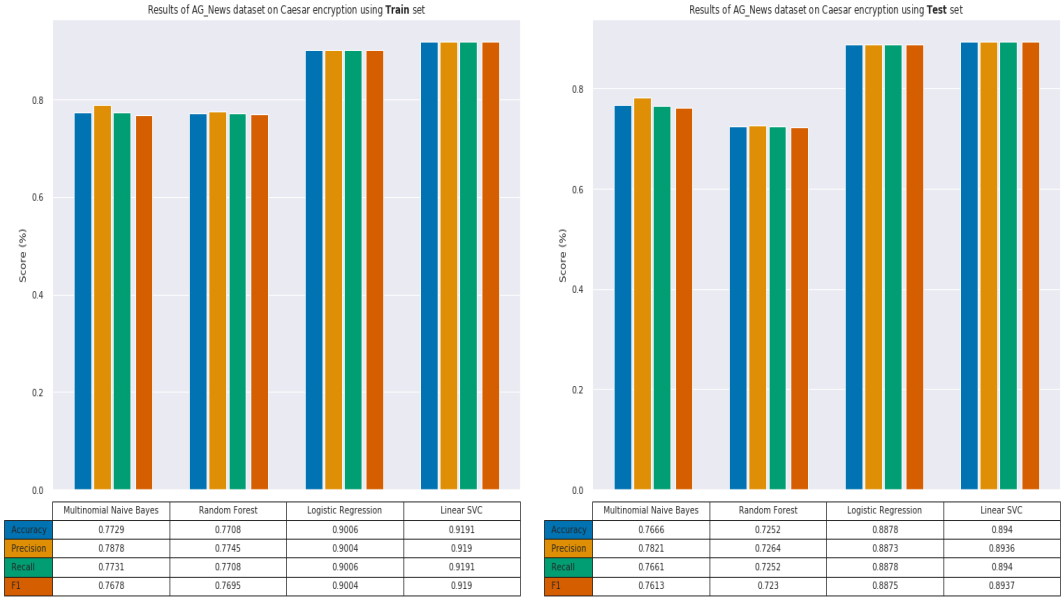
Figure (A.7) News Category dataset Caesar encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.8166	0.7829	0.9052	0.9547
Precision	0.8236	0.8191	0.9072	0.9551
Recall	0.8167	0.783	0.9053	0.9548
F1	0.817	0.7867	0.9054	0.9548

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.8113	0.7405	0.8887	0.8985
Precision	0.8191	0.7786	0.8911	0.8994
Recall	0.811	0.7401	0.8885	0.8984
F1	0.8117	0.7443	0.8889	0.8986

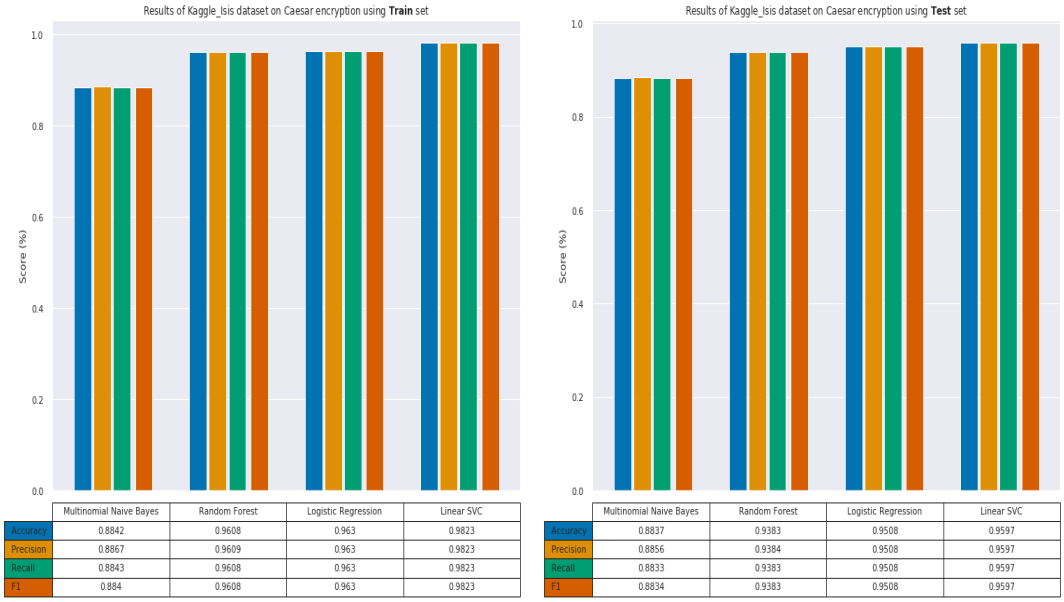
Figure (A.8) News Category with additional feature dataset Caesar encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.7729	0.7708	0.9006	0.9191
Precision	0.7878	0.7745	0.9004	0.919
Recall	0.7731	0.7708	0.9006	0.9191
F1	0.7678	0.7695	0.9004	0.919

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.7666	0.7252	0.8878	0.894
Precision	0.7821	0.7264	0.8873	0.8936
Recall	0.7661	0.7252	0.8878	0.894
F1	0.7613	0.723	0.8875	0.8937

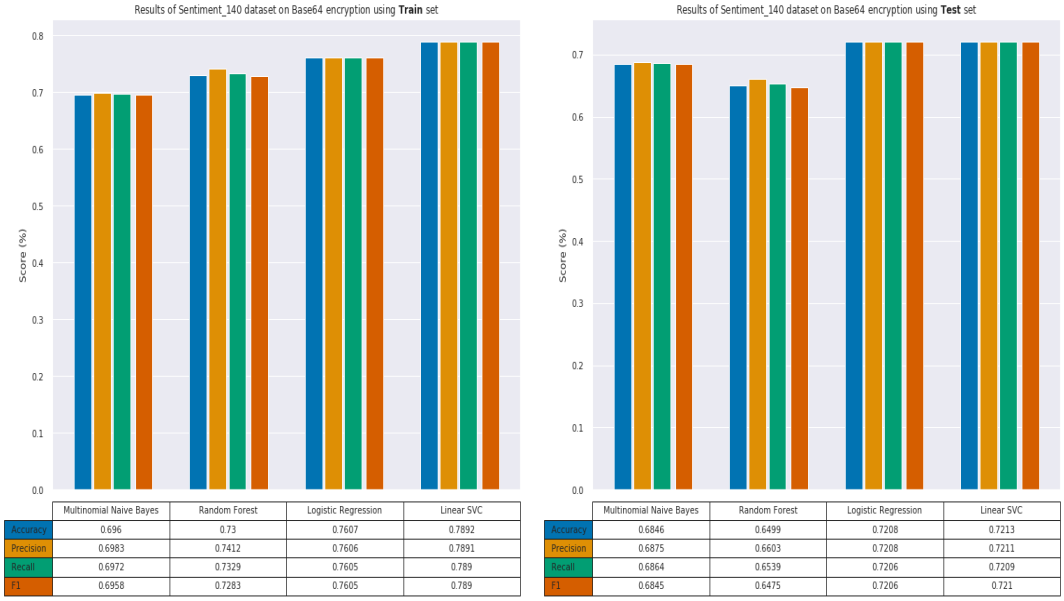
Figure (A.9) AG News dataset Caesar encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.8842	0.9608	0.963	0.9823
Precision	0.8867	0.9609	0.963	0.9823
Recall	0.8843	0.9608	0.963	0.9823
F1	0.884	0.9608	0.963	0.9823

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.8837	0.9383	0.9508	0.9597
Precision	0.8856	0.9384	0.9508	0.9597
Recall	0.8833	0.9383	0.9508	0.9597
F1	0.8834	0.9383	0.9508	0.9597

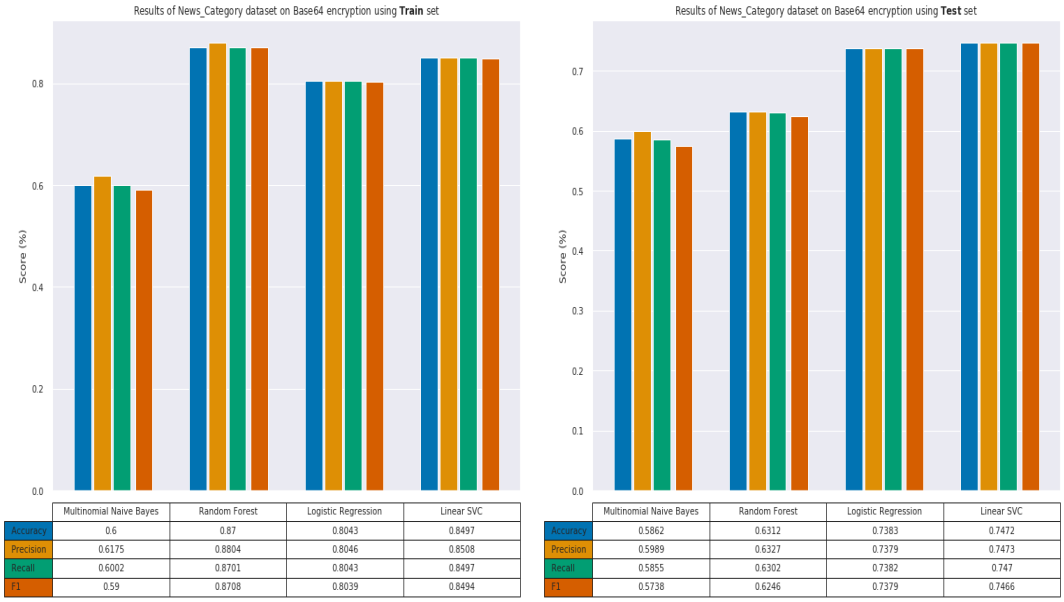
Figure (A.10) Kaggle Isis dataset Caesar encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.696	0.73	0.7607	0.7892
Precision	0.6983	0.7412	0.7606	0.7891
Recall	0.6972	0.7329	0.7605	0.789
F1	0.6958	0.7283	0.7605	0.789

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6846	0.6499	0.7208	0.7213
Precision	0.6875	0.6603	0.7208	0.7211
Recall	0.6864	0.6539	0.7206	0.7209
F1	0.6845	0.6475	0.7206	0.721

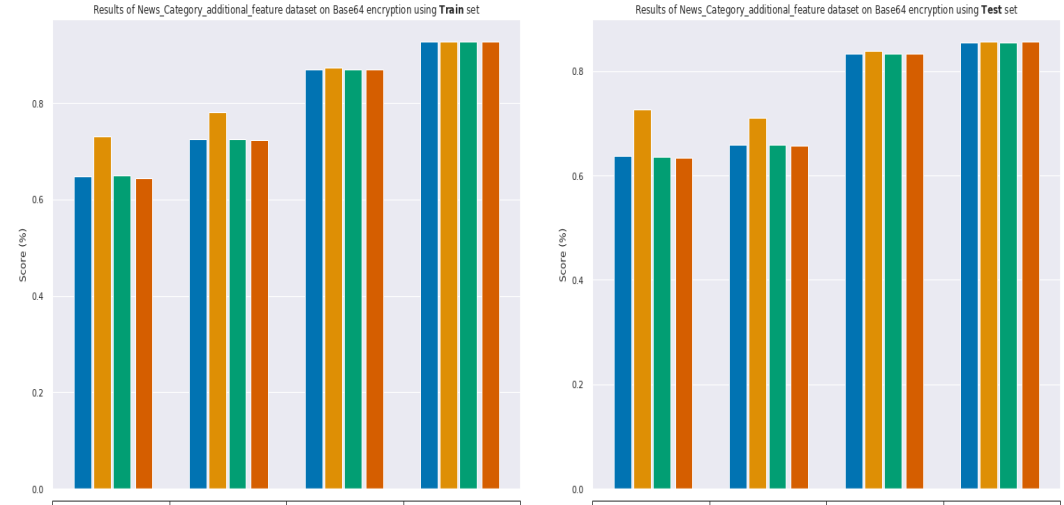
Figure (A.11) Sentiment 140 dataset Base64 encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6	0.87	0.8043	0.8497
Precision	0.6175	0.8804	0.8046	0.8508
Recall	0.6002	0.8701	0.8043	0.8497
F1	0.59	0.8708	0.8039	0.8494

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5862	0.6312	0.7383	0.7472
Precision	0.5989	0.6327	0.7379	0.7473
Recall	0.5855	0.6302	0.7382	0.747
F1	0.5738	0.6246	0.7379	0.7466

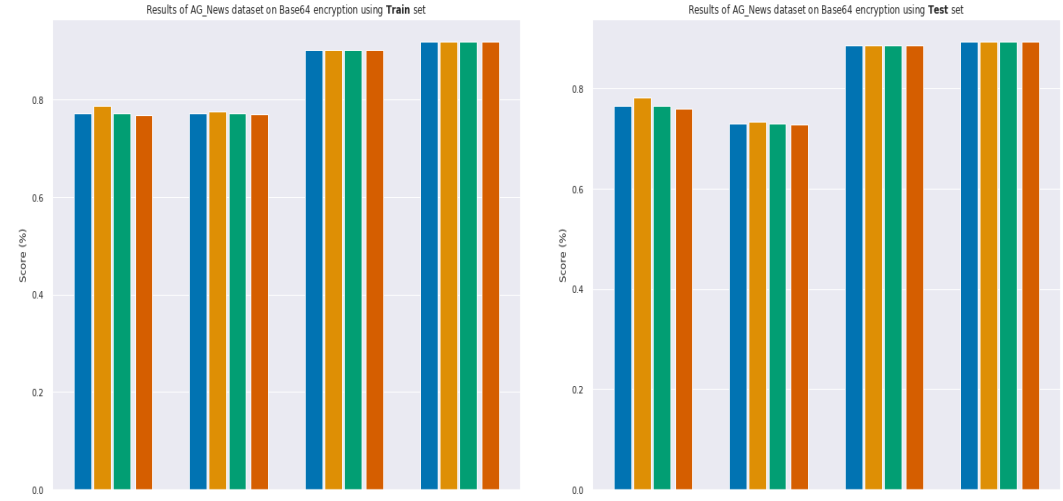
Figure (A.12) News Category dataset Base64 encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6484	0.7254	0.8702	0.927
Precision	0.7306	0.7804	0.8742	0.9278
Recall	0.6489	0.7251	0.8703	0.927
F1	0.6432	0.7229	0.8703	0.927

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6375	0.6582	0.834	0.8557
Precision	0.727	0.7102	0.8389	0.8573
Recall	0.6355	0.6591	0.834	0.8557
F1	0.6337	0.6568	0.8344	0.8559

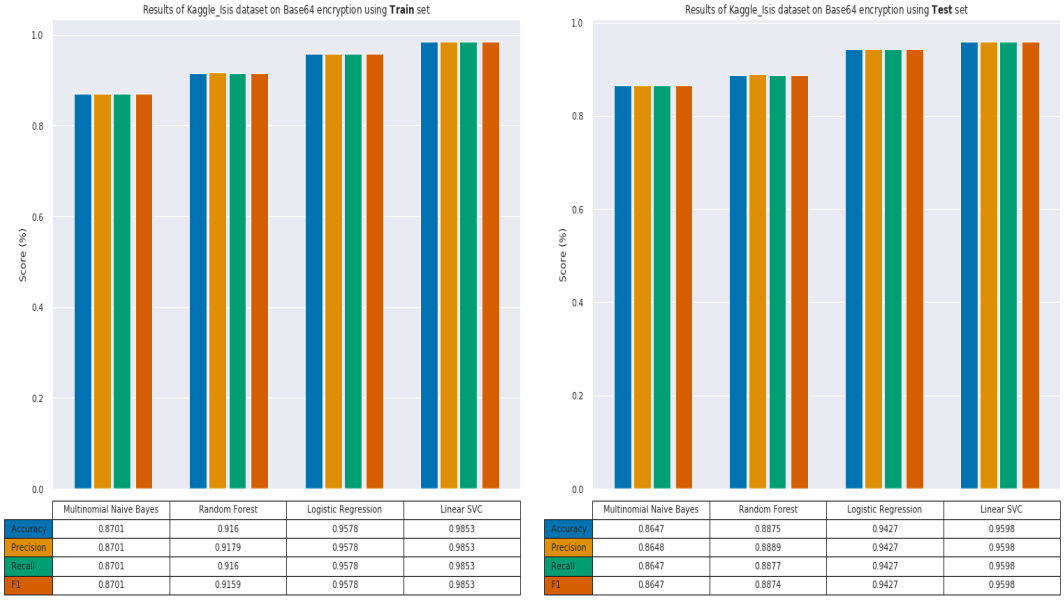
Figure (A.13) News Category with additional feature dataset Base64 encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.7724	0.7713	0.9015	0.9194
Precision	0.7875	0.7755	0.9013	0.9193
Recall	0.7728	0.7715	0.9016	0.9195
F1	0.7675	0.7705	0.9014	0.9193

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.7662	0.7309	0.8868	0.894
Precision	0.7821	0.7333	0.8862	0.8934
Recall	0.7644	0.7299	0.8862	0.8934
F1	0.7606	0.7289	0.8861	0.8933

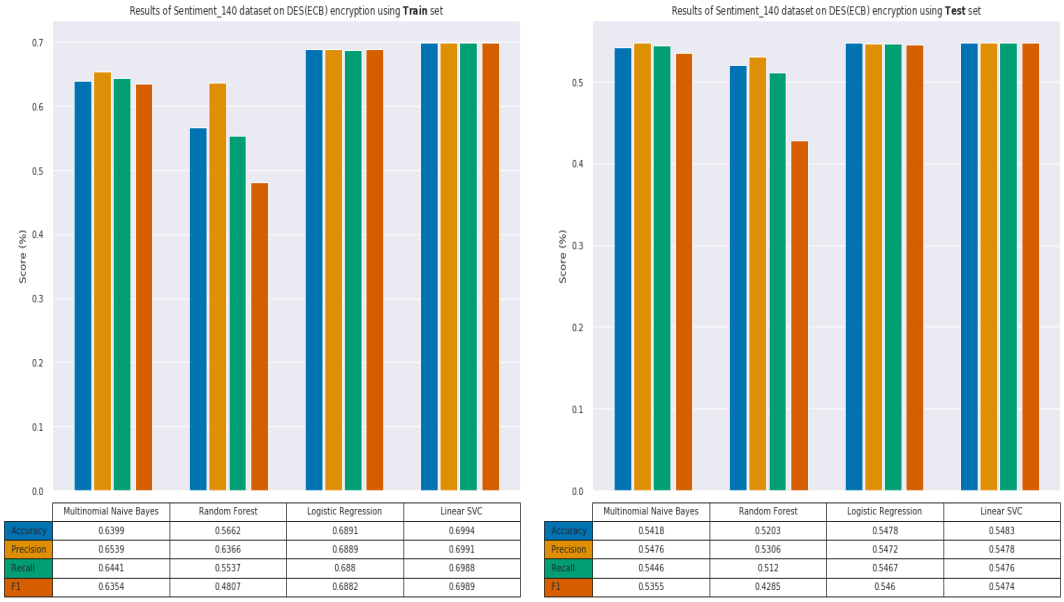
Figure (A.14) AG News dataset Base64 encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.8701	0.916	0.9578	0.9853
Precision	0.8701	0.9179	0.9578	0.9853
Recall	0.8701	0.916	0.9578	0.9853
F1	0.8701	0.9159	0.9578	0.9853

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.8647	0.8875	0.9427	0.9598
Precision	0.8648	0.8889	0.9427	0.9598
Recall	0.8647	0.8877	0.9427	0.9598
F1	0.8647	0.8874	0.9427	0.9598

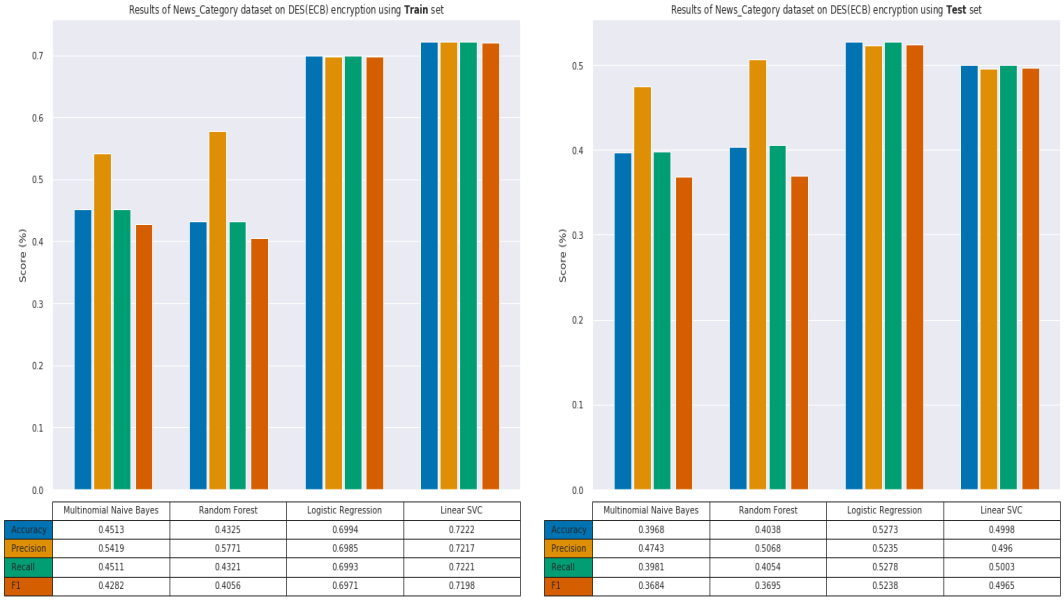
Figure (A.15) Kaggle Isis dataset Base64 encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6399	0.5662	0.6891	0.6994
Precision	0.6539	0.6366	0.6889	0.6991
Recall	0.6441	0.5537	0.688	0.6988
F1	0.6354	0.4807	0.6882	0.6989

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5418	0.5203	0.5478	0.5483
Precision	0.5476	0.5306	0.5472	0.5478
Recall	0.5446	0.512	0.5467	0.5476
F1	0.5355	0.4285	0.546	0.5474

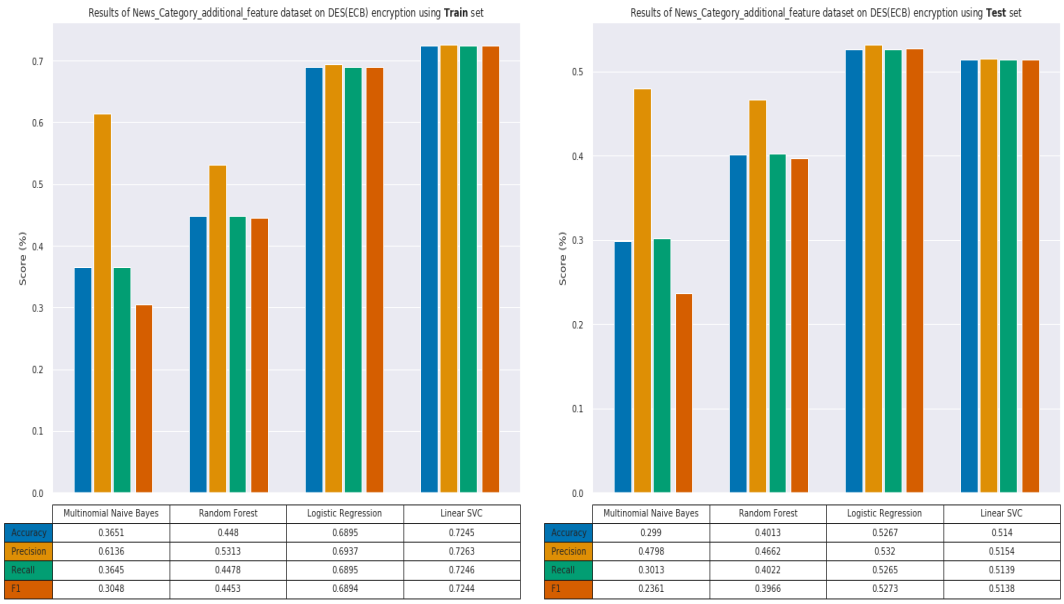
Figure (A.16) Sentiment 140 dataset DES(ECB) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.4513	0.4325	0.6994	0.7222
Precision	0.5419	0.5771	0.6985	0.7217
Recall	0.4511	0.4321	0.6993	0.7221
F1	0.4282	0.4056	0.6971	0.7198

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.3968	0.4038	0.5273	0.4998
Precision	0.4743	0.5068	0.5235	0.496
Recall	0.3981	0.4054	0.5278	0.5003
F1	0.3684	0.3695	0.5238	0.4965

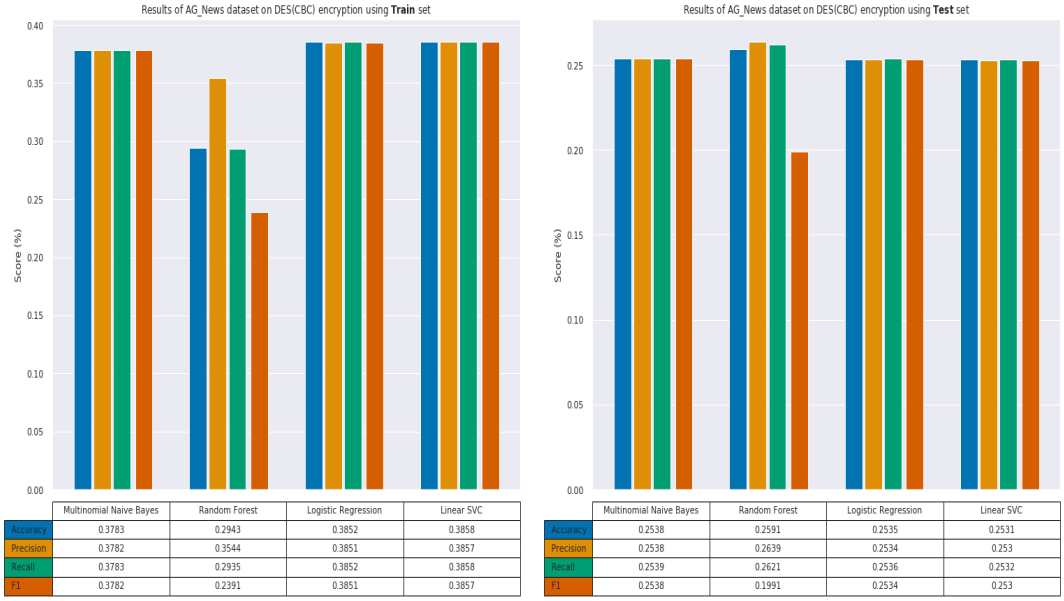
Figure (A.17) News Category dataset DES(ECB) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.3651	0.448	0.6895	0.7245
Precision	0.6136	0.5313	0.6937	0.7263
Recall	0.3645	0.4478	0.6895	0.7246
F1	0.3048	0.4453	0.6894	0.7244

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.299	0.4013	0.5267	0.514
Precision	0.4798	0.4662	0.532	0.5154
Recall	0.3013	0.4022	0.5265	0.5139
F1	0.2361	0.3966	0.5273	0.5138

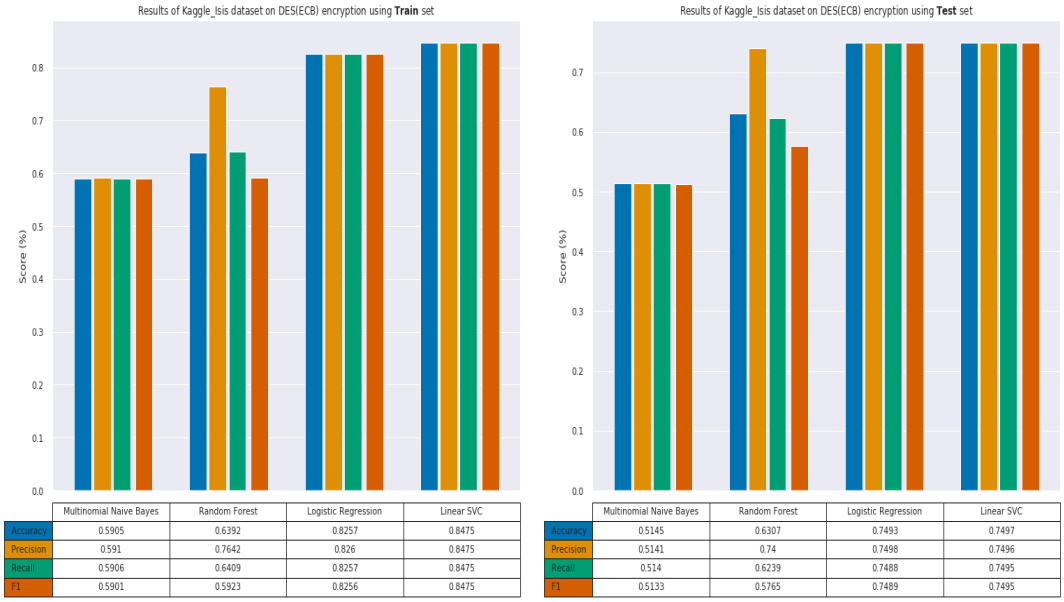
Figure (A.18) News Category with additional feature dataset DES(ECB) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.3783	0.2943	0.3852	0.3858
Precision	0.3782	0.3544	0.3851	0.3857
Recall	0.3783	0.2935	0.3852	0.3858
F1	0.3782	0.2391	0.3851	0.3857

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.2538	0.2591	0.2535	0.2531
Precision	0.2538	0.2639	0.2534	0.253
Recall	0.2539	0.2621	0.2536	0.2532
F1	0.2538	0.1991	0.2534	0.253

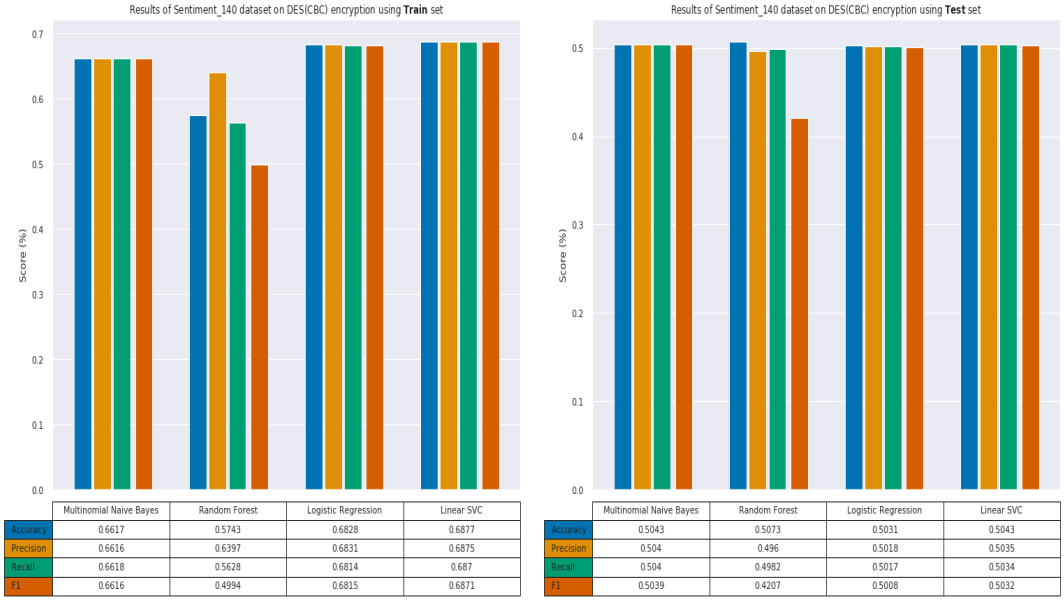
Figure (A.19) AG News dataset DES(ECB) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5905	0.6392	0.8257	0.8475
Precision	0.591	0.7642	0.826	0.8475
Recall	0.5906	0.6409	0.8257	0.8475
F1	0.5901	0.5923	0.8256	0.8475

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5145	0.6307	0.7493	0.7497
Precision	0.5141	0.74	0.7498	0.7496
Recall	0.514	0.6239	0.7488	0.7495
F1	0.5133	0.5765	0.7489	0.7495

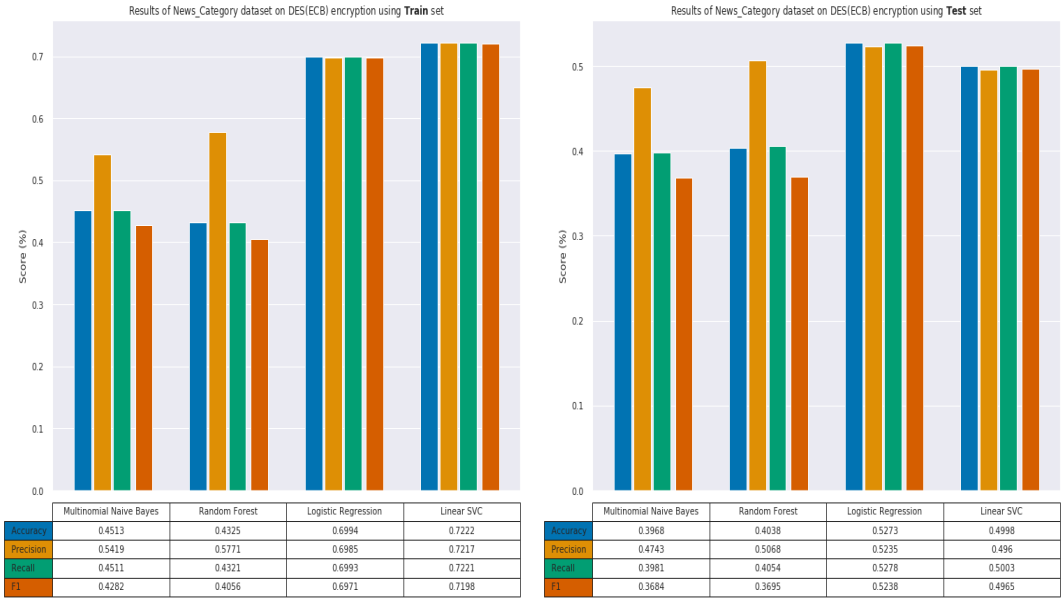
Figure (A.20) Kaggle Isis dataset DES(ECB) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6617	0.5743	0.6828	0.6877
Precision	0.6616	0.6397	0.6831	0.6875
Recall	0.6618	0.5628	0.6814	0.687
F1	0.6616	0.4994	0.6815	0.6871

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5043	0.5073	0.5031	0.5043
Precision	0.504	0.496	0.5018	0.5035
Recall	0.504	0.4982	0.5017	0.5034
F1	0.5039	0.4207	0.5008	0.5032

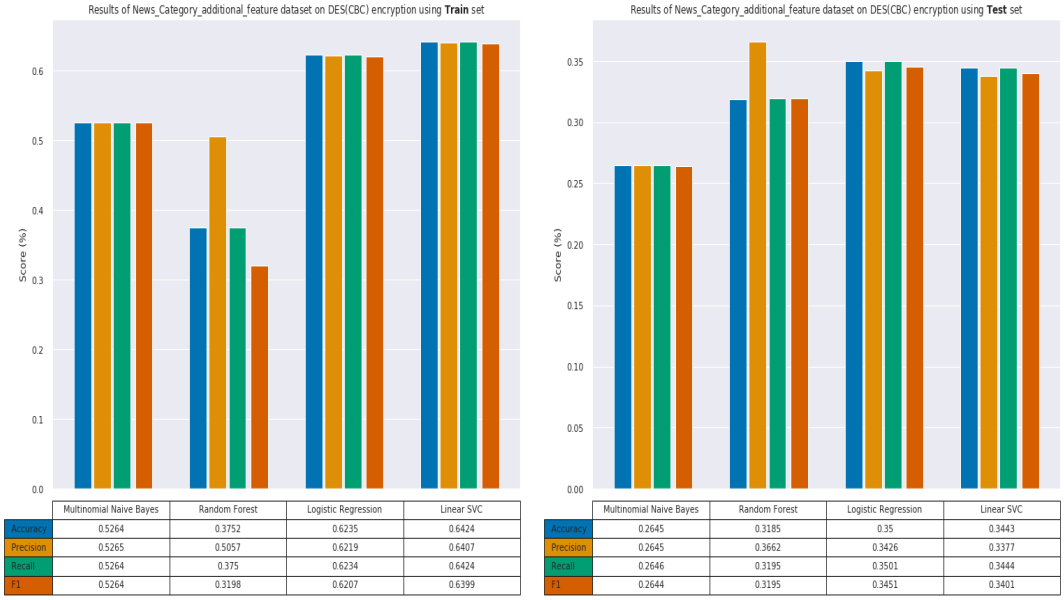
Figure (A.21) Sentiment 140 dataset DES(CBC) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.4513	0.4325	0.6994	0.7222
Precision	0.5419	0.5771	0.6985	0.7217
Recall	0.4511	0.4321	0.6993	0.7221
F1	0.4282	0.4056	0.6971	0.7198

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.3968	0.4038	0.5273	0.4998
Precision	0.4743	0.5068	0.5235	0.496
Recall	0.3981	0.4054	0.5278	0.5003
F1	0.3684	0.3695	0.5238	0.4965

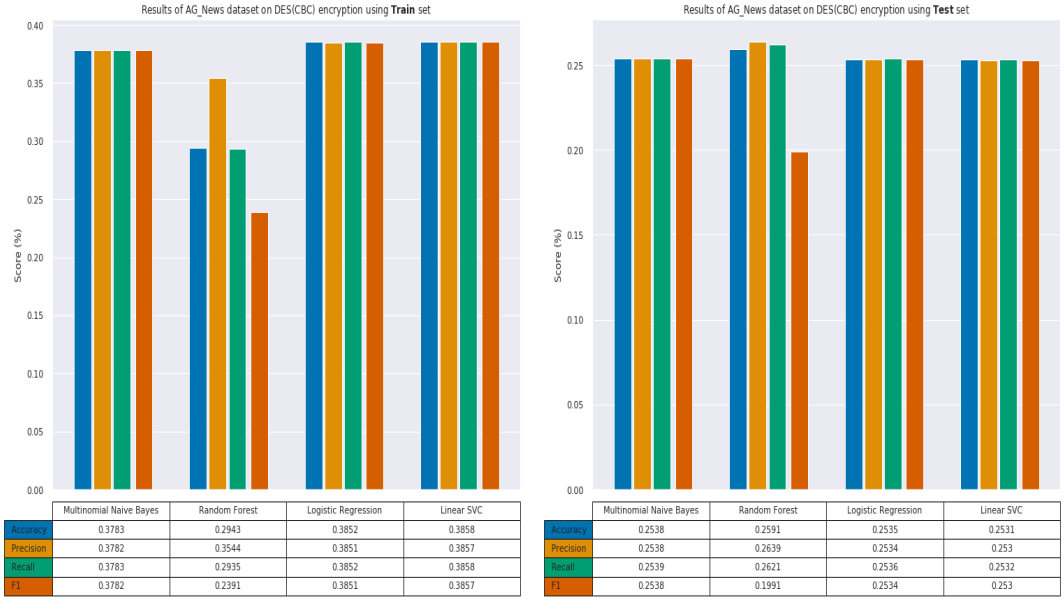
Figure (A.22) News Category dataset DES(CBC) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5264	0.3752	0.6235	0.6424
Precision	0.5265	0.5057	0.6219	0.6407
Recall	0.5264	0.375	0.6234	0.6424
F1	0.5264	0.3198	0.6207	0.6399

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.2645	0.3185	0.35	0.3443
Precision	0.2645	0.3662	0.3426	0.3377
Recall	0.2646	0.3195	0.3501	0.3444
F1	0.2644	0.3195	0.3451	0.3401

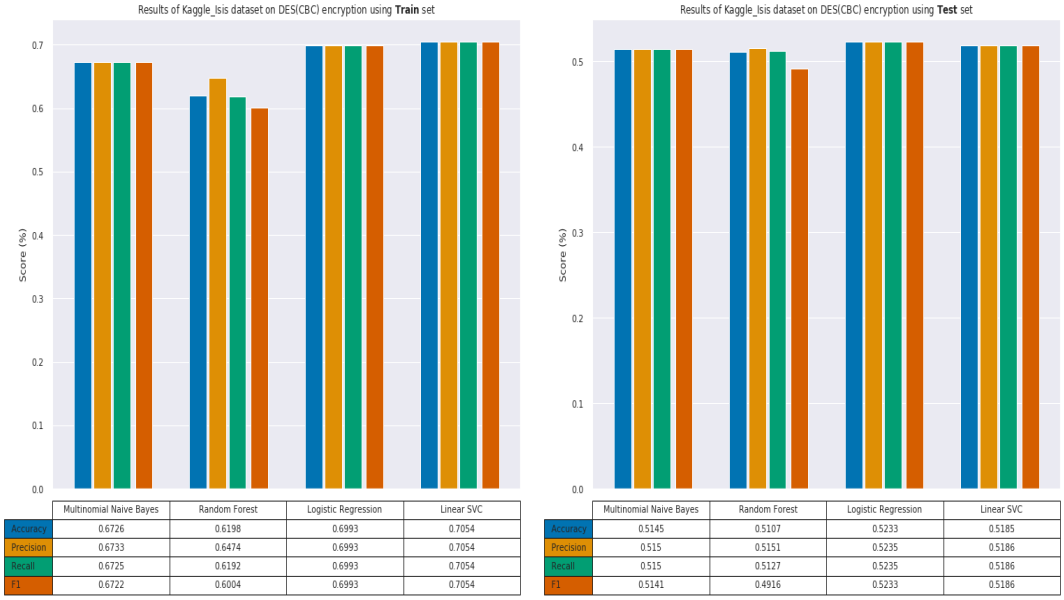
Figure (A.23) News Category with additional feature dataset DES(CBC) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.3783	0.2943	0.3852	0.3858
Precision	0.3782	0.3544	0.3851	0.3857
Recall	0.3783	0.2935	0.3852	0.3858
F1	0.3782	0.2391	0.3851	0.3857

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.2538	0.2591	0.2535	0.2531
Precision	0.2538	0.2639	0.2534	0.253
Recall	0.2539	0.2621	0.2536	0.2532
F1	0.2538	0.1991	0.2534	0.253

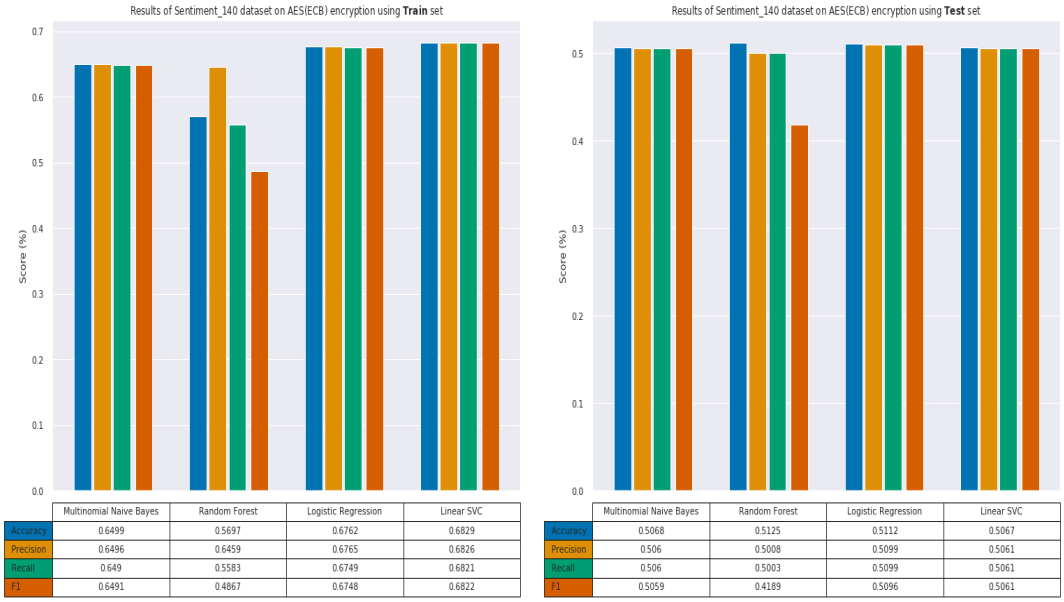
Figure (A.24) AG News dataset DES(CBC) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6726	0.6198	0.6993	0.7054
Precision	0.6733	0.6474	0.6993	0.7054
Recall	0.6725	0.6192	0.6993	0.7054
F1	0.6722	0.6004	0.6993	0.7054

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5145	0.5107	0.5233	0.5185
Precision	0.515	0.5151	0.5235	0.5186
Recall	0.515	0.5127	0.5235	0.5186
F1	0.5141	0.4916	0.5233	0.5186

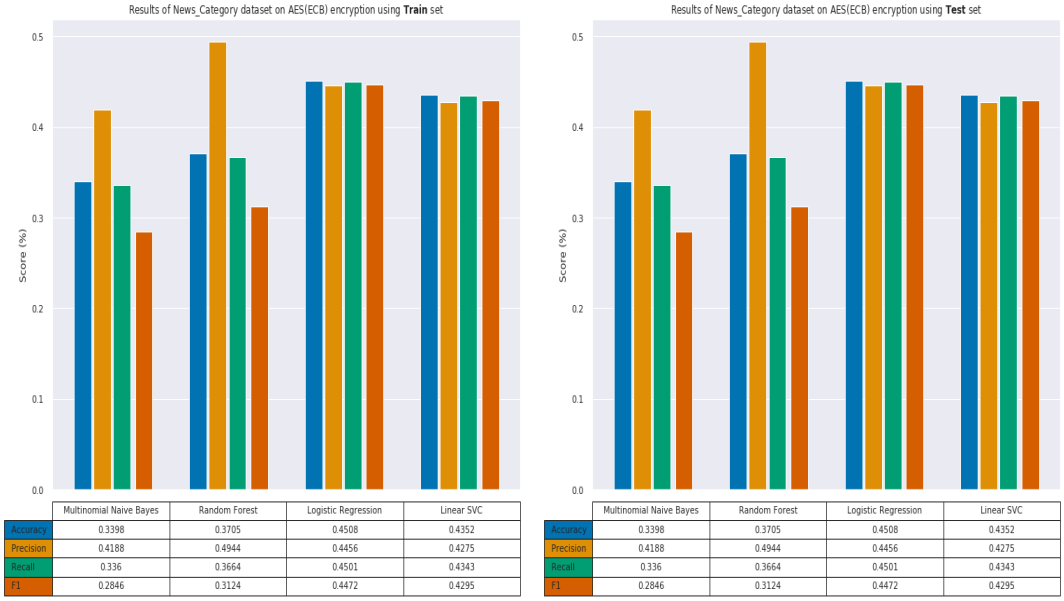
Figure (A.25) Kaggle Isis dataset DES(CBC) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6499	0.5697	0.6762	0.6829
Precision	0.6496	0.6459	0.6765	0.6826
Recall	0.649	0.5583	0.6749	0.6821
F1	0.6491	0.4867	0.6748	0.6822

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5068	0.5125	0.5112	0.5067
Precision	0.506	0.5008	0.5099	0.5061
Recall	0.506	0.5003	0.5099	0.5061
F1	0.5059	0.4189	0.5096	0.5061

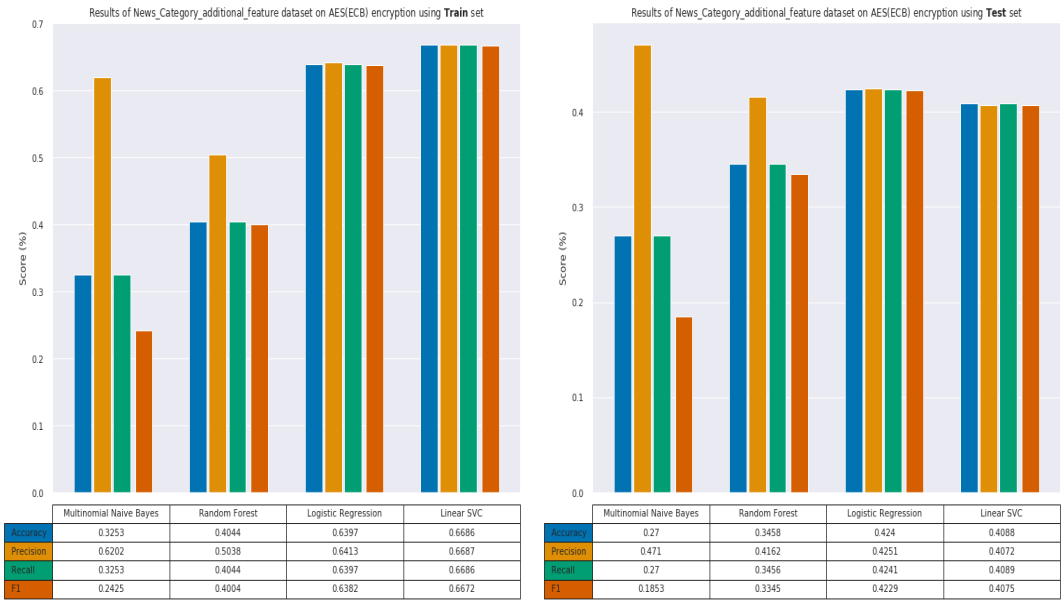
Figure (A.26) Sentiment 140 dataset AES(ECB) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.3398	0.3705	0.4508	0.4352
Precision	0.4188	0.4944	0.4456	0.4275
Recall	0.336	0.3664	0.4501	0.4343
F1	0.2846	0.3124	0.4472	0.4295

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.3398	0.3705	0.4508	0.4352
Precision	0.4188	0.4944	0.4456	0.4275
Recall	0.336	0.3664	0.4501	0.4343
F1	0.2846	0.3124	0.4472	0.4295

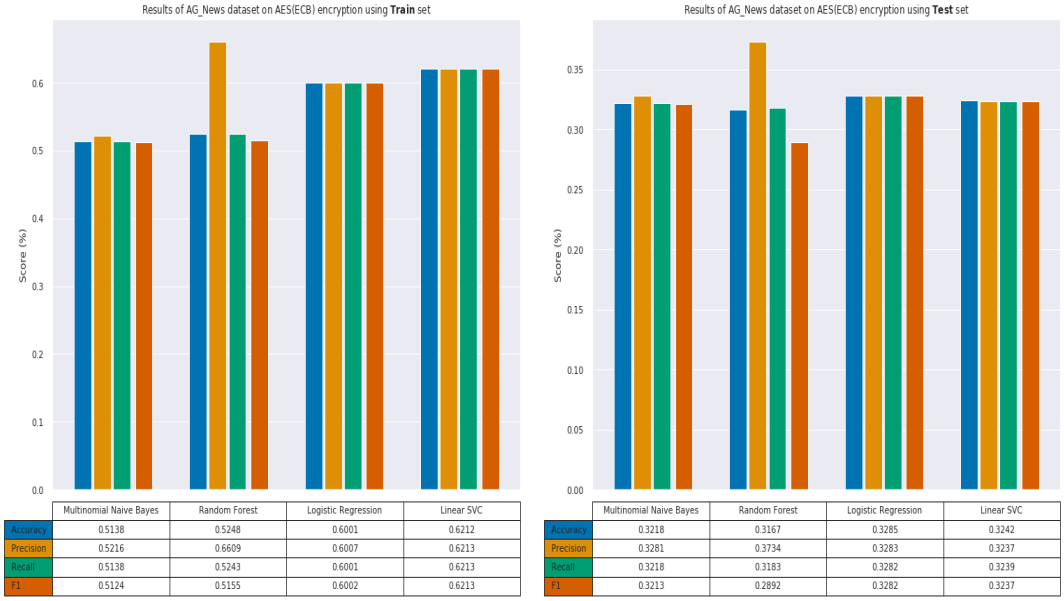
Figure (A.27) News Category dataset AES(ECB) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.3253	0.4044	0.6397	0.6686
Precision	0.6202	0.5038	0.6413	0.6687
Recall	0.3253	0.4044	0.6397	0.6686
F1	0.2425	0.4004	0.6382	0.6672

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.27	0.3458	0.424	0.4088
Precision	0.471	0.4162	0.4251	0.4072
Recall	0.27	0.3456	0.4241	0.4089
F1	0.1853	0.3345	0.4229	0.4075

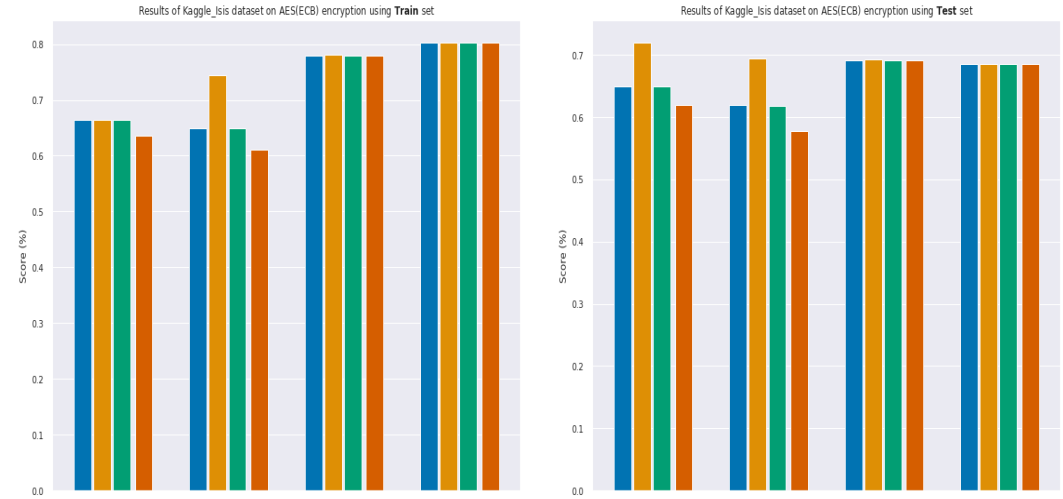
Figure (A.28) News Category with additional feature dataset AES(ECB) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5138	0.5248	0.6001	0.6212
Precision	0.5216	0.6609	0.6007	0.6213
Recall	0.5138	0.5243	0.6001	0.6213
F1	0.5124	0.5155	0.6002	0.6213

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.3218	0.3167	0.3285	0.3242
Precision	0.3281	0.3734	0.3283	0.3237
Recall	0.3218	0.3183	0.3282	0.3239
F1	0.3213	0.2892	0.3282	0.3237

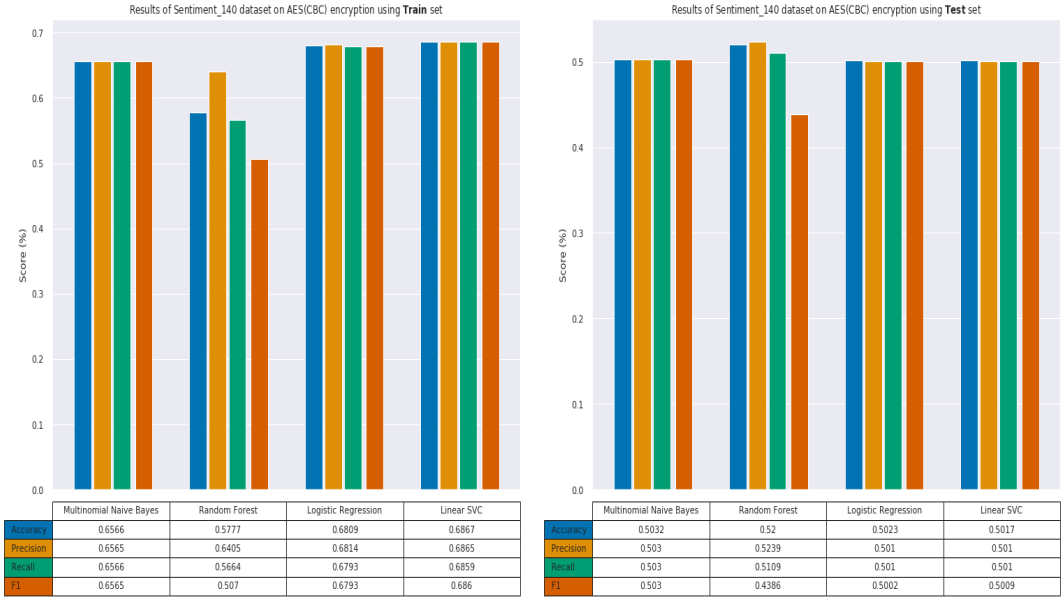
Figure (A.29) AG News dataset AES(ECB) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6643	0.6487	0.7794	0.803
Precision	0.6643	0.744	0.7813	0.8032
Recall	0.6644	0.6489	0.7794	0.803
F1	0.6359	0.6109	0.7791	0.8029

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.65	0.6192	0.6917	0.6847
Precision	0.7201	0.6944	0.692	0.6847
Recall	0.6496	0.6182	0.6916	0.6847
F1	0.6194	0.5774	0.6915	0.6847

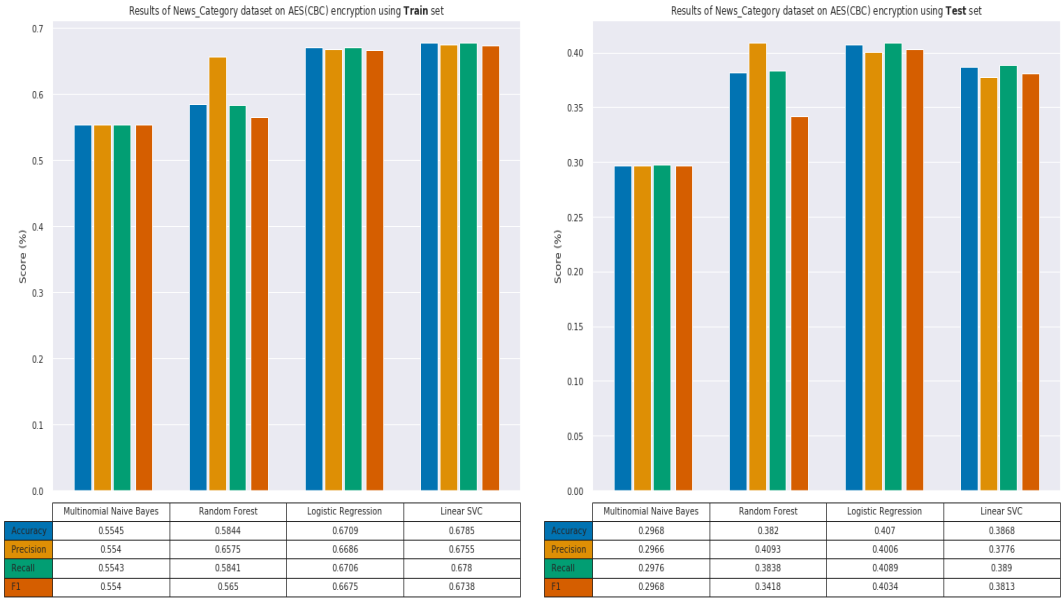
Figure (A.30) Kaggle Isis dataset AES(ECB) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.6566	0.5777	0.6809	0.6867
Precision	0.6565	0.6405	0.6814	0.6865
Recall	0.6566	0.5664	0.6793	0.6859
F1	0.6565	0.507	0.6793	0.686

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5032	0.52	0.5023	0.5017
Precision	0.503	0.5239	0.501	0.501
Recall	0.503	0.5109	0.501	0.501
F1	0.503	0.4386	0.5002	0.5009

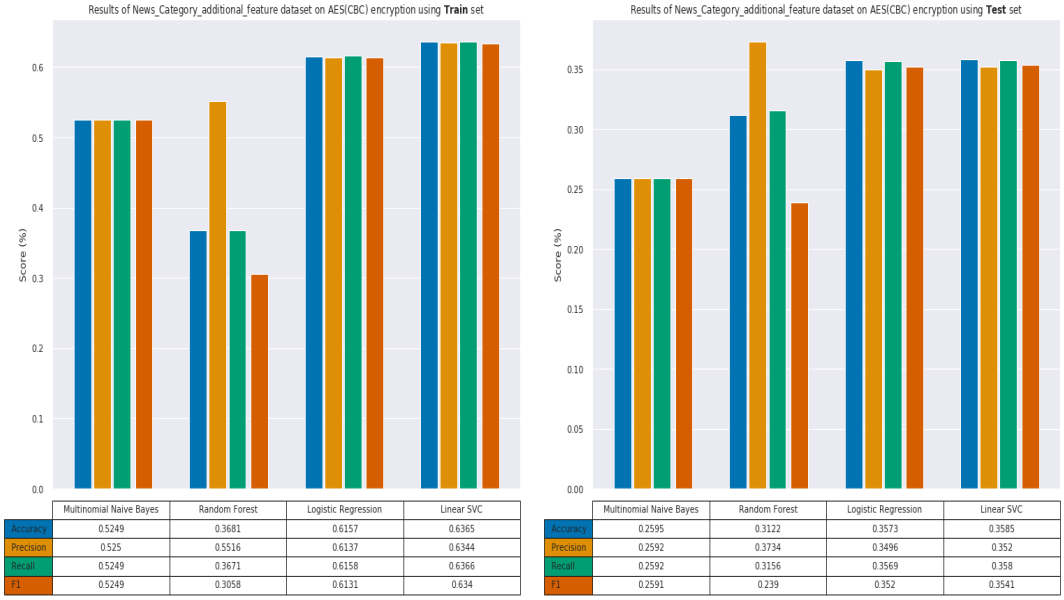
Figure (A.31) Sentiment 140 dataset AES(CBC) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5545	0.5844	0.6709	0.6785
Precision	0.554	0.6575	0.6686	0.6755
Recall	0.5543	0.5841	0.6706	0.678
F1	0.554	0.565	0.6675	0.6738

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.2968	0.382	0.407	0.3868
Precision	0.2966	0.4093	0.4006	0.3776
Recall	0.2976	0.3838	0.4089	0.389
F1	0.2968	0.3418	0.4034	0.3813

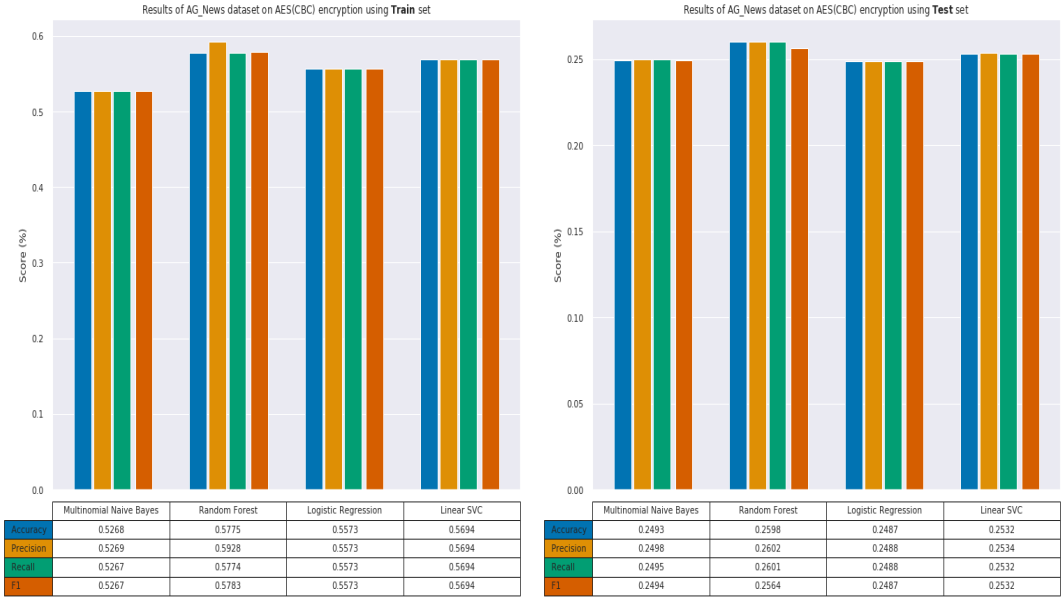
Figure (A.32) News Category dataset AES(CBC) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5249	0.3681	0.6157	0.6365
Precision	0.525	0.5516	0.6137	0.6344
Recall	0.5249	0.3671	0.6158	0.6366
F1	0.5249	0.3058	0.6131	0.634

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.2595	0.3122	0.3573	0.3585
Precision	0.2592	0.3734	0.3496	0.352
Recall	0.2592	0.3156	0.3569	0.358
F1	0.2591	0.239	0.352	0.3541

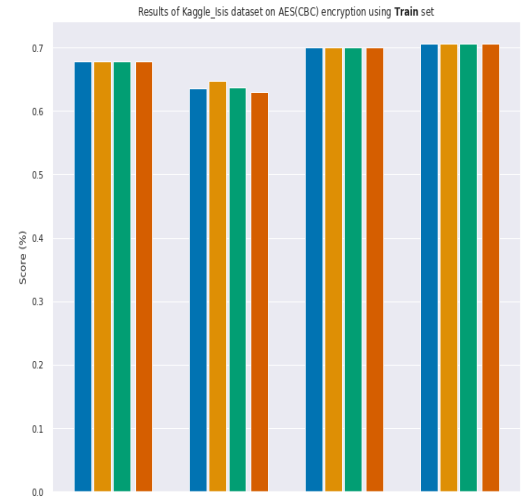
Figure (A.33) News Category with additional feature dataset AES(CBC) encryption results on train and test



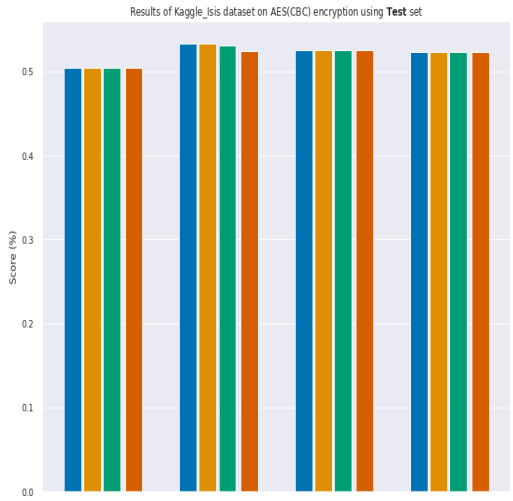
	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5268	0.5775	0.5573	0.5694
Precision	0.5269	0.5928	0.5573	0.5694
Recall	0.5267	0.5774	0.5573	0.5694
F1	0.5267	0.5783	0.5573	0.5694

	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.2493	0.2598	0.2487	0.2532
Precision	0.2498	0.2602	0.2488	0.2534
Recall	0.2495	0.2601	0.2488	0.2532
F1	0.2494	0.2564	0.2487	0.2532

Figure (A.34) AG News dataset AES(CBC) encryption results on train and test



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.678	0.6365	0.7	0.7067
Precision	0.6783	0.6482	0.7	0.7067
Recall	0.6779	0.6373	0.6999	0.7067
F1	0.6778	0.6301	0.6999	0.7067



	Multinomial Naive Bayes	Random Forest	Logistic Regression	Linear SVC
Accuracy	0.5038	0.5333	0.5255	0.5235
Precision	0.5043	0.5325	0.5255	0.5235
Recall	0.5043	0.5304	0.5255	0.5235
F1	0.5037	0.5243	0.5254	0.5234

Figure (A.35) Kaggle Isis dataset AES(CBC) encryption results on train and test

Bibliography

[1] 20 newsgroup dataset. http://qwone.com/~jason/20Newsgroups,Accessed: 2020-02-24.

[2] Ag's corpus of news articles. http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html,Accessed: 2020-01-29.

[3] Kaggle 20 newsgroup ciphertext challenge. https://www.kaggle.com/c/20-newsgroups-ciphertext-challenge,Accessed: 2020-02-24.

[4] Kaggle 20 newsgroup ciphertext challenge luis bronchal notebook. https://www.kaggle.com/lbronchal/without-breaking-ciphers-0-48-1b,Accessed: 2020-02-24.

[5] Kaggle isis dataset. https://www.kaggle.com/fifthtribe/how-isis-uses-twitter,Accessed: 2020-02-24.

[6] Kaggle news category dataset. https://www.kaggle.com/rmisra/news-category-dataset,Accessed: 2020-02-24.

[7] Kaggle sentiment 140 dataset. https://www.kaggle.com/kazanova/sentiment140,Accessed: 2020-02-24.

[8] Ako Muhammad Abdullah. Advanced encryption standard (aes) algorithm to encrypt and decrypt data. 2017.

[9] Lei Huang Alec Go, Richa Bhayani. Twitter sentiment classification using distant supervision. 2009.

[10] Piotr Bojanowski Tomas Mikolov Armand Joulin, Edouard Grave. Bag of tricks for efficient text classification. 2016.

[11] Faudziah Ahmad Ashwak Mahmood Alabaichi, Ramlan Mahmood and Mohammed S. Mechee. Randomness analysis on blowfish block cipher using ecb and cbc modes. 2013.

[12] James Barry. Sentiment analysis of online reviews usingbag-of-words and lstm approaches. 2017.

[13] Sarit Chakraborty Bijoyan Das. An improved text sentiment classification model using tf-idf and next word negation. 2013.

[14] Yunqian Ma Cha Zhang. *EnsembleMachine LearningCha Zhang · Yunqian MaEditorsMethods and Applications*. 2012.

[15] F.R. Chaumartin. A knowledge-based system for headline sentiment tagging. 2007.

[16] Chih-Jen Lin Sathiya Sathiya Keerthi Sellamanickam Sundararajan profile imageS. Sundararajan Cho-Jui Hsieh, Kaiwei Chang. A dual coordinate descent method for large-scale linear svm. 2008.

[17] Wang Ling Dani Yogatama, Chris Dyer and Phil Blunsom. Generative and discriminative text classification with recurrent neural network. 2017.

[18] Mitchel Klein David G. Kleinbaum. *Logistic Regression.* 3 edition, 2002.

[19] D.Coppersmith. The data encryption standard (des) and its strength against attacks. 1994.

[20] Siti Ummi Masruroh Feri Fahrianto and Nopan Ziro Ando. Encrypted sms application on android with combination of caesar cipher and vigenere algorithm. 2014.

[21] Miriam Fernandez and Harith Alani. Contextual semantics for radicalisation detection on twitter. 2018.

[22] Olga Fuks. Classification of news dataset. 2018.

[23] Kun Du Zhou Li Hai-Xin Duan XiaoDong Su Guang Liu Zhifeng Geng Jianping Wu H. T. Yang, Xiulin Ma. How to learn klingon without a dictionary: Detection and measurement of black keywords used by the underground economy. 2017.

[24] Mark A. Hall. *Correlation-based Feature Selection for Machine Learning.* 1999.

[25] Robbi Rahim2 Heri Nurdiyanto1 and Nur Wulan. Symmetric stream cipher using triple transposition key method and base64 algorithm for security improvement. 2017.

[26] Christopher D. Manning Jeffrey Pennington, Richard Socher. Glove: Global vectors for word representation. 2014.

[27] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. 1972.

[28] Jason Eisner Adam Stubblefield Joshua Mason, Kathryn Watkins. A natural language approach to automated cryptanalysis of two-time pads. 2006.

[29] Singleton M. Wicentowski R. Katz, P. Swat-mp: The semeval-2007 systems for task 5 and task 14. 2007.

[30] B. Klimt and Y. Yang. Introducing the enron corpus. 2004.

[31] J. Heidarysafa Mendu Barnes & Brown Kowsari, Meimandi. Text classification algorithms: A survey. 2019.

[32] Navarro B. Vazquez S. Montoyo A. Kozareva, Z. Ua-zbsa: A headline emotion classification through web information. 2007.

[33] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. 2002.

[34] Mitsuru Matsui. Linear cryptanalysis method for des cipher. 2001.

[35] Murat Kantarcioglu Mohammad Saiful Islam, Mehmet Kuzu. Poster: inference attacks against searchable encryption protocols. 2011.

[36] Abdulkadir Hassan Disina Zahraddeen A. Pindar Nur Shafinaz Ahmad Shakir Mustafa Mat Deris Muhammad Faheem Mushtaq, Sapiee Jamel. A survey on the cryptographic encryption algorithms. 2017.

[37] Kavita Kelkar Parth Vora, Mansi Khara. Classification of tweets based on emotions using word embedding and random forest classifiers. 2017.

[38] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[39] Annamma Abraham R. Sathya. Comparison of supervised and unsupervised learning algorithms for pattern classification. 2013.

[40] Juan Ramos. Using tf-idf to determine word relevance in document queries s. 2003.

[41] D. K. Kirange Ratnadeep R. Deshmukh. Emotion classification of news headlines using svm. 2012.

[42] Rhouma Rhouma. Cryptanalysis of a spatiotemporal chaotic image/video cryptosystem. 2008.

[43] S.S Sehra. A review paper on algorithms used for text classification. 2013.

[44] Jürgen Schmidhuber Sepp Hochreiter. Long short-term memory. 1997.

[45] Velasquez F. Stamatatos E. Gelbukh A. & Chanona-Hernández L. Sidorov, G. Syntactic n-grams as machine learning features for natural language processing. expert systems with applications. 2014.

[46] & Shashi M Singh, A. K. Vectorization of text documents for identifying unifiable news articles. 2019.

[47] S.P. Mansoor Suhaila O. Sharif, L.I. Kuncheva. Classifying encryption algorithms using pattern recognition techniques. 2010.

[48] Greg Corrado Jeffrey Dean Tomas Mikolov, Kai Chen. Distributed representations of words and phrasesand their compositionality. 2013.

[49] Agrawal A. & Rath S. K. Tripathy, A. Classification of sentiment reviews using n-gram machine learning approach. expert systems with applications. 2016.

[50] DesiRamayanti UmniySalamah. Implementation of logistic regression algorithm for complaint text classification in indonesian ministry of marine and fisheries. 2018.

[51] Miao D. Chauchat J.-H. Zhao R. & Li W. Wei, Z. N-grams based feature selection and text representation for chinese text classification. international journal of computational intelligence systems. 2009.

[52] Yann A Lecun Xiang Zhang, Junbo Zhao. Character-level convolutional networks for textclassification. 2015.

[53] Shuo Xu. Bayesian naive bayes classifiersto text classification. 2018.

[54] Harry Zhang. The optimality of naive bayes. 2004.