Spam Email Classification

Review Of Minor Project

for E-3 2019 Admitted Batch

Submitted as a part of Minor Project

Telugu Amruta -S180995

Bandrapalli Sukanya S190767

Kundangi Shireesha -S190942

Gudipudi Natasha Divines-S190252

Under the Supervision of

Mr. S. Satish Kumar M. Tech,(Ph.D)

Assistant Professor

Department of Computer Science and Engineering.

Rajiv Gandhi University of Knowledge Technologies

Srikakulam – 532402

Abstract

In this generation, we are facing a lot through the internet. There are many pros and cons, but cons get the first place comparatively. People are using them for illegal and unethical conduct, phishing and fraud. Email is one such communication medium that comes to mind when we think of secure communication.

For example, people receive both spam and important mail. Here, there is some chance that we ignore the important mail. Because important mails are less compared to spam mail, For solving this kind of problem, we need some method that filters spam and non-spam (important) mail. So, it is needed to identify those spam emails that are fraud. This project will identify those spam by using machine learning algorithms. Nowadays, machine learning methods have been able to detect and filter out spam emails.

Contents

Chapter 1

Introduction

1.1 Introduction to Project

In this era of information technology, information sharing has become very easy and fast. No one wants to receive emails not related to their interests because they waste receivers time and resources. Spam is any irrelevant and unwanted message or email sent by the attacker to a significant number of recipients by using email or any other medium of information sharing. It has been found that many users have experienced financial losses as victims of email scams and others.

Spam mail is mainly meant as unnecessary or junk mail, where it causes issues for the users like spreading malware, typically advertising, and so on. Which exponential growth of the emails and unwanted emails using machine learning techniques.

1.2 Application

1. Email Service Providers Gmail: Utilizes machine learning models to filter spam emails, providing users with a cleaner inbox and protecting against phishing attempts. Yahoo Mail: Implements ML-based spam filters to enhance user experience and reduce the risk of malicious emails.

2. Desktop and Mobile Clients Applications like Mozilla Thunderbird and various mobile email apps use built-in spam filters to protect users from unwanted emails.

3. Fraud Prevention Detecting and blocking phishing attempts that target customers through email.

4. Corporate Email Security Organizations use spam filters to protect employees from phishing attacks, malware, and other security threats.

1.3 Motivation Towards Project

Spam classifications play a vital role in every human's life. It can be a direct or indirect way. Spam emails, also known as junk emails, are unsolicited messages sent in bulk, typically for advertising, phishing, spreading malware, or other malicious purposes. Spammers are very creative, and sometimes users may be tricked by them, leading them into getting scammed.

There are some cases where this situation of a user getting scammed effected so much that the user ended up taking their own life. So this project is to classify these emails and separate them from important ones. So that users can easily access their emails without getting spammed.

1.4 Problem Statement

Spammers are in continuous war with Email service providers. Email service providers implement various spam filtering methods to retain their users, and spammers are continuously changing patterns, using various embedding tricks to get through filtering.These filters can never be too aggressive because a slight misclassification may lead to important information loss for consumer. A rigid filtering method with additional reinforcements is needed to tackle this problem.

Chapter 2

Approach To Your Project

2.1 Explain About Your Project

Today, spam has become a huge problem. It has been estimated that around 55% of all emails are reported as spam, and the number has been growing steadily. Spam, also known as unsolicited bulk email, has led to the increasing use of email, as email provides the perfect way to send an unwanted advertisement or junk newsgroup posting at no cost to the sender. These chances are exploited by those irresponsible organizations, resulting in billions and millions of junk files in the user's mail box.

Spam has been a major concern given the offensive content of messages. spam is a waste of time. The end user is at risk of deleting legitimate mail by mistake. Moreover, spam also impacted the economy, which led some countries to adopt legislation.

In this project, machine learning techniques are used to detect the spam message in a message. Machine learning is where computers can learn to do something without the need to explicitly program them for the task. It uses data to produce a program to perform a task such as classification. A combination of algorithms is used to learn the classification rules from messages. After learning from the pre-labeled data, each of these algorithms predicts which class the unknown text may belong to, and the category predicted by the majority is considered final.

2.2 Data Set

The Spam Email Classification dataset is taken from the kaggle.The csv file contains 5572 rows, each row for each email. There are 2 columns.The first Column is about the category and second column is the email messages.The category column contains the indication of the mails like spam mail and ham mail.Thus, information regarding all 5172 emails are stored in a compact dataframe rather than as separate text files.

2.3 Prediction technique

For prediction,some classifiers are used. Some of them are Naive Bayes classifier, Logistic regression, KNN classification ,Random forest. First the all the extra spaces, special characters, unwanted symbols, repeated symbols are cleaned. Next the data is split into training and testing. By using classifiers,the accuracy can be found.

2.3.1 Naive Bayes classifier

The Naive Bayes classifier is a probabilistic model that uses Bayes' theorem with the assumption of feature independence. It's highly effective for text classification tasks like spam detection, calculating the probability of an email being spam based on word frequencies. Despite its simplicity, it performs well and is computationally efficient.

2.3.2 Logistic regression

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value.Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification.

2.3.3 Random forest

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

2.3.4 KNN Classification

KNN is a classification algorithm. It comes under supervised algorithms. All the data points are assumed to be in an n-dimensional space. And then based on neighbors the category of current data is determined based on the majority. Euclidian distance is used to determine the distance between points.

2.4 Graphs

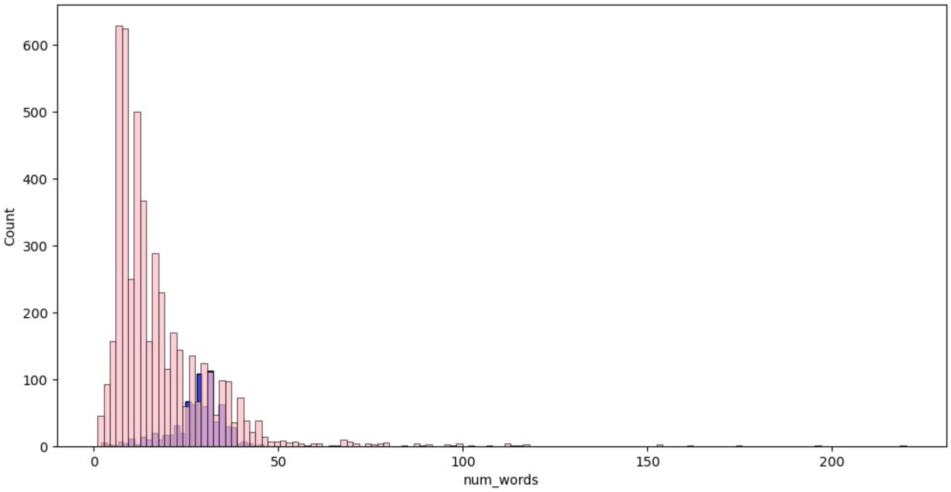


Figure 2.1: Number Of Words

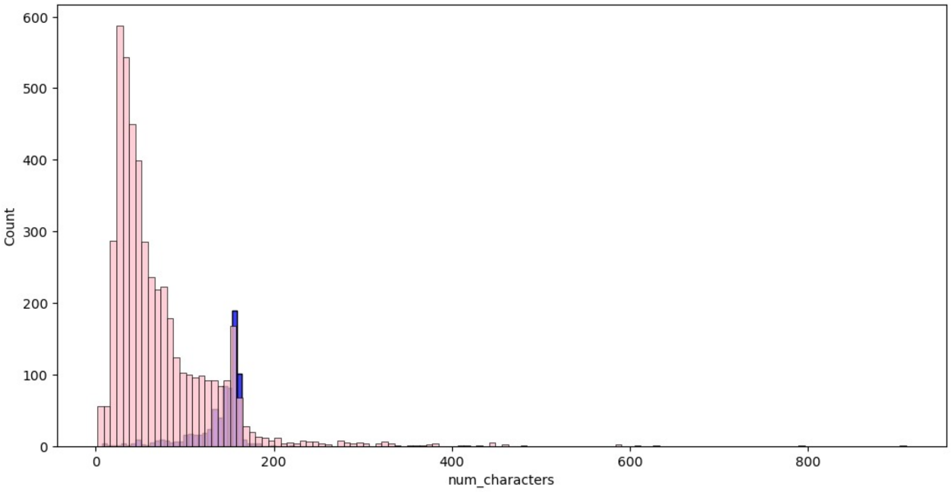


Figure 2.2: Number Of Characters

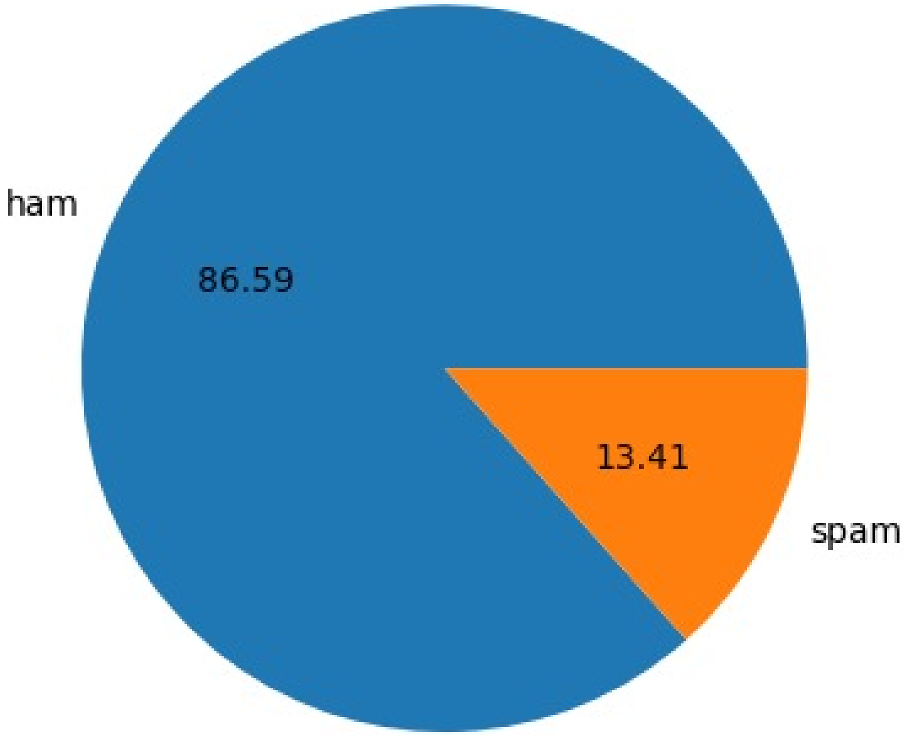


Figure 2.3: Spam Mails VS Ham Mails

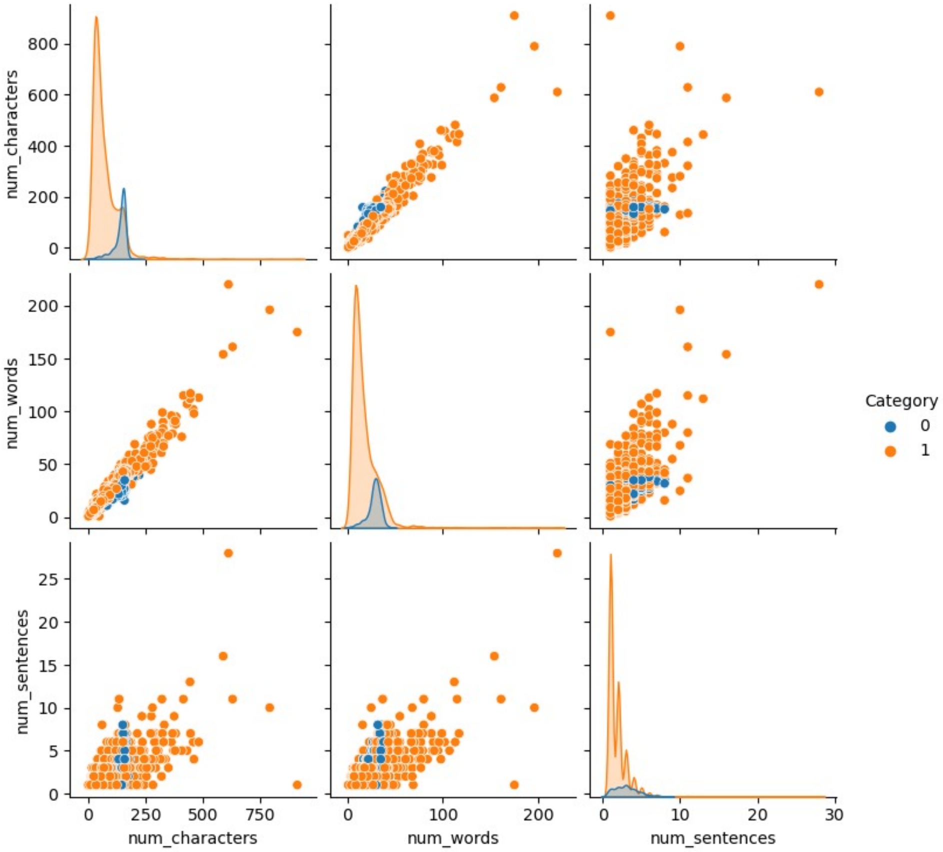


Figure 2.4: Scatterplot of Number of sentences, words,characters

2.5 Visualization

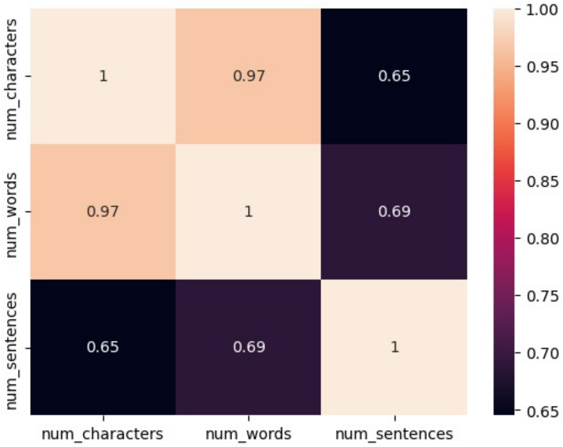


Figure 2.5: Visualization Example

Chapter 3

Code

3.1 Explain Your Code With Outputs

we are importing pandas for data manipulating Numpy for calculations , Pyplot for graphs, Seaborn for effective plots

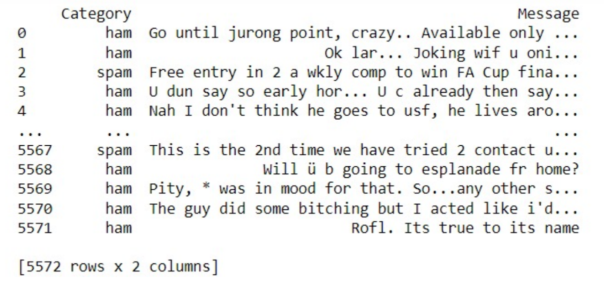
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score
```

3.2 Reading Data Sample

Now I want to see some rows and columns

To see the first 5 rows, we use df.head(5) function

```
file1 =r"C:\Users\Sireesha\Downloads\mail_data .csv"

data = pd.read_csv(file1, encoding='latin-1')

print(data)
```



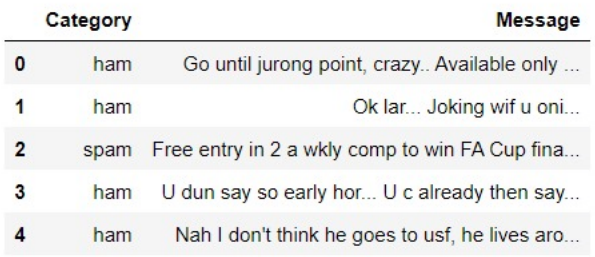
```
     Category                                            Message
0         ham  Go until jurong point, crazy.. Available only ...
1         ham                      Ok lar... Joking wif u oni...
2        spam  Free entry in 2 a wkly comp to win FA Cup fina...
3         ham  U dun say so early hor... U c already then say...
4         ham  Nah I don't think he goes to usf, he lives aro...
...       ...                                                ...
5567     spam  This is the 2nd time we have tried 2 contact u...
5568      ham               Will ü b going to esplanade fr home?
5569      ham  Pity, * was in mood for that. So...any other s...
5570      ham  The guy did some bitching but I acted like i'd...
5571      ham                         Rofl. Its true to its name

[5572 rows x 2 columns]
```

Figure 3.1: DataSet Reading

3.3 Reading The Head Of The Dataset

Finding the first head(5)

```
print(data.head(5))
```



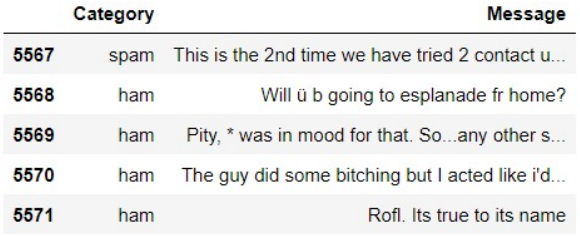
	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Figure 3.2: Head of The Dataset

3.4 Reading The Tail Of The Dataset

Finding the first tail(5)

```
print(data.tail(5))
```



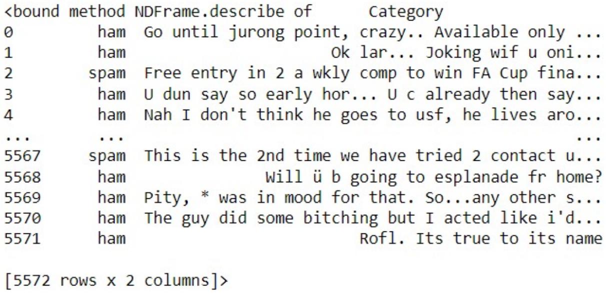
	Category	Message
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

Figure 3.3: Tail Of The Dataset

3.5 Reading The info Of The Dataset

As we have read the dataset and found out the head and tails, its better to know the information of the dataset.

```
print(data.info())
```



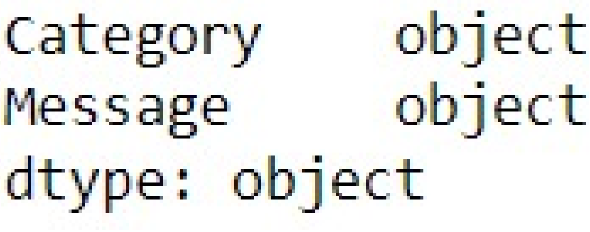
```
<bound method NDFrame.describe of      Category
0         ham  Go until jurong point, crazy.. Available only ...
1         ham                      Ok lar... Joking wif u oni...
2        spam  Free entry in 2 a wkly comp to win FA Cup fina...
3         ham  U dun say so early hor... U c already then say...
4         ham  Nah I don't think he goes to usf, he lives aro...
...       ...                                                ...
5567     spam  This is the 2nd time we have tried 2 contact u...
5568      ham               Will ü b going to esplanade fr home?
5569      ham  Pity, * was in mood for that. So...any other s...
5570      ham  The guy did some bitching but I acted like i'd...
5571      ham                         Rofl. Its true to its name

[5572 rows x 2 columns]>
```

Figure 3.4: Info Of The Dataset

3.6 Finding the datatypes Of The Dataset

Finding what type our data is

```
print(data.dtypes)
```



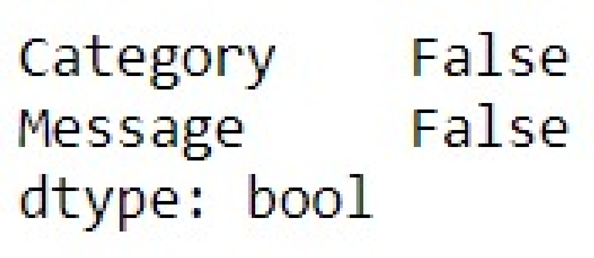
```
Category    object
Message     object
dtype: object
```

Figure 3.5: Datatypes

3.7 Finding The Null Values

It is important to learn about null values.Null values may be the values which ruin the accuracy of the code.

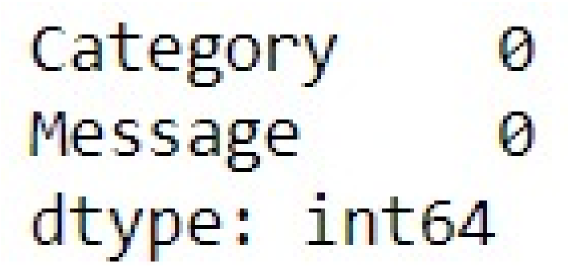
```
print(data.isnull().any())
```



```
Category    False
Message     False
dtype: bool
```

Figure 3.6: isnull().any()

```
print(data.isnull().sum())
```

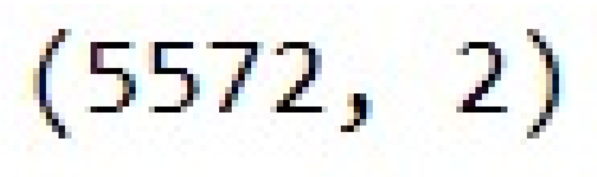


```
Category    0
Message     0
dtype: int64
```

Figure 3.7: isnull().sum()

3.8 Finding The Shape Of The Dataset

```
print(data.shape)
```

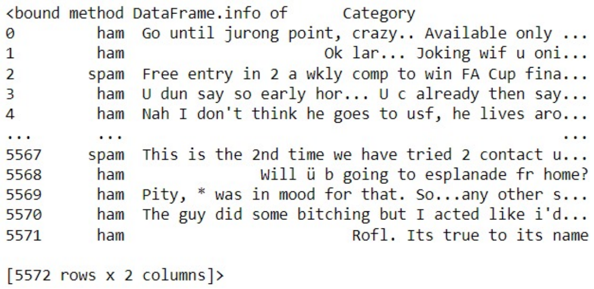


```
(5572, 2)
```

Figure 3.8: Shape Of The Dataset

3.9 Description Of The Dataset

```
print(data.describe)
```



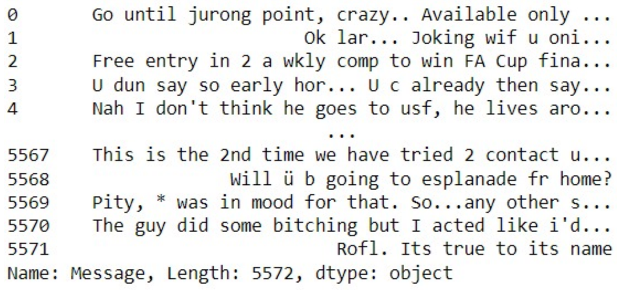
```
<bound method DataFrame.info of      Category                                            
0         ham  Go until jurong point, crazy.. Available only ...
1         ham                      Ok lar... Joking wif u oni...
2        spam  Free entry in 2 a wkly comp to win FA Cup fina...
3         ham  U dun say so early hor... U c already then say...
4         ham  Nah I don't think he goes to usf, he lives aro...
...       ...                                                ...
5567     spam  This is the 2nd time we have tried 2 contact u...
5568      ham               Will ü b going to esplanade fr home?
5569      ham  Pity, * was in mood for that. So...any other s...
5570      ham  The guy did some bitching but I acted like i'd...
5571      ham                         Rofl. Its true to its name

[5572 rows x 2 columns]>
```

Figure 3.9: Describe Of The Dataset

3.10 Feature Extraction From The Dataset

```
data.loc[data['Category'] == 'spam', 'Category',] = 0

data.loc[data['Category'] == 'ham', 'Category',] = 1

X = data['Message']

Y = data['Category']
```

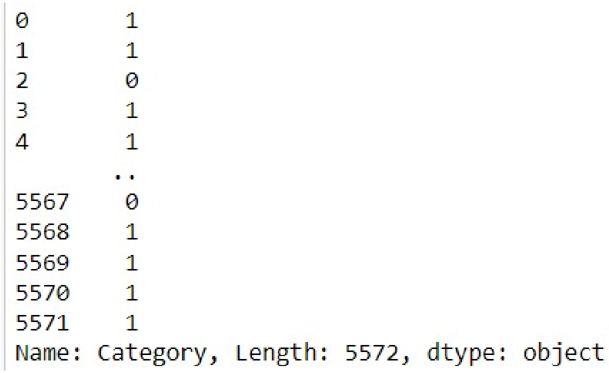
```
print(X)
```



```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
                              ...                        
5567    This is the 2nd time we have tried 2 contact u...
5568                 Will ü b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                           Rofl. Its true to its name
Name: Message, Length: 5572, dtype: object
```

Figure 3.10: X feature Of The Dataset

```
print(Y)
```

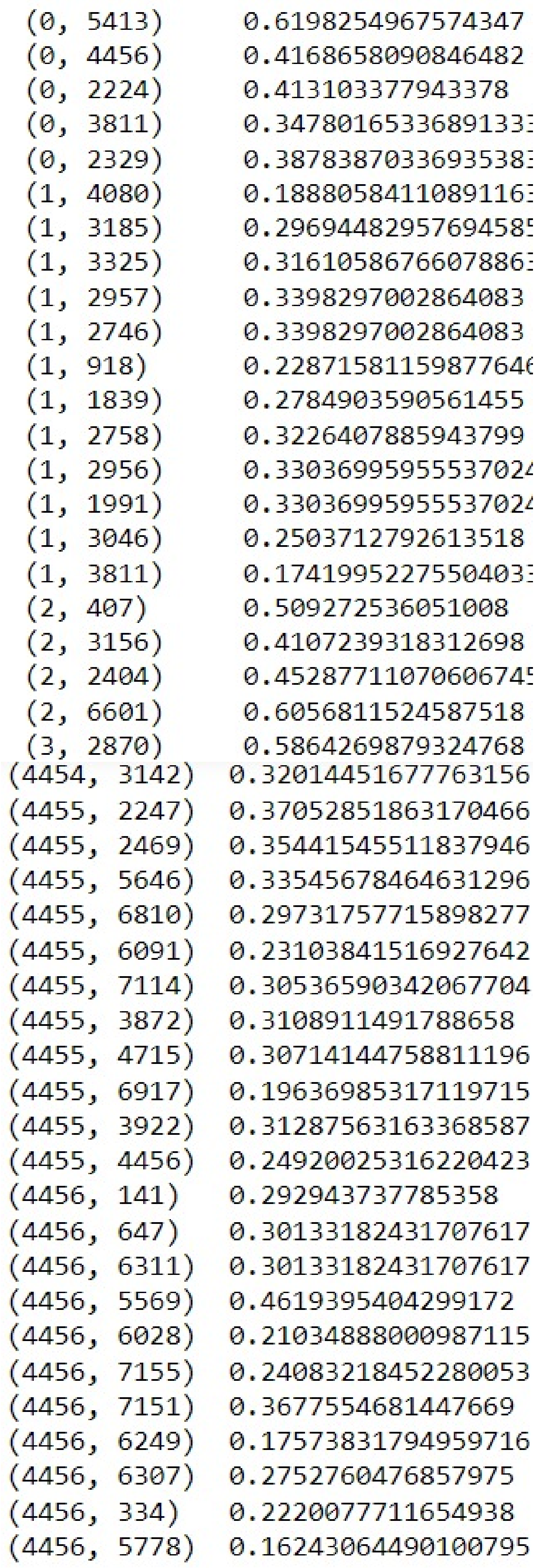


```
0       1
1       1
2       0
3       1
4       1
       ..
5567    0
5568    1
5569    1
5570    1
5571    1
Name: Category, Length: 5572, dtype: object
```

Figure 3.11: Y feature Of The Dataset

3.11 Training The Model

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
 ↪ random_state=3)
feature_extraction = TfidfVectorizer(min_df = 1, stop_words='english',
 ↪ lowercase='True')
X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)
Y_train = Y_train.astype('int')
Y_test = Y_test.astype('int')
print(X_train_features)
```



```
  (0, 5413)     0.6198254967574347
  (0, 4456)     0.4168658090846482
  (0, 2224)     0.413103377943378
  (0, 3811)     0.34780165336891333
  (0, 2329)     0.38783870336935383
  (1, 4080)     0.18880584110891163
  (1, 3185)     0.29694482957694585
  (1, 3325)     0.31610586766078863
  (1, 2957)     0.3398297002864083
  (1, 2746)     0.3398297002864083
  (1, 918)      0.22871581159877646
  (1, 1839)     0.2784903590561455
  (1, 2758)     0.3226407885943799
  (1, 2956)     0.33036995955537024
  (1, 1991)     0.33036995955537024
  (1, 3046)     0.2503712792613518
  (1, 3811)     0.17419952275504033
  (2, 407)      0.509272536051008
  (2, 3156)     0.4107239318312698
  (2, 2404)     0.45287711070606745
  (2, 6601)     0.6056811524587518
  (3, 2870)     0.5864269879324768
(4454, 3142)  0.32014451677763156
(4455, 2247)  0.37052851863170466
(4455, 2469)  0.35441545511837946
(4455, 5646)  0.33545678464631296
(4455, 6810)  0.29731757715898277
(4455, 6091)  0.23103841516927642
(4455, 7114)  0.30536590342067704
(4455, 3872)  0.3108911491788658
(4455, 4715)  0.30714144758811196
(4455, 6917)  0.19636985317119715
(4455, 3922)  0.31287563163368587
(4455, 4456)  0.24920025316220423
(4456, 141)   0.292943737785358
(4456, 647)   0.30133182431707617
(4456, 6311)  0.30133182431707617
(4456, 5569)  0.4619395404299172
(4456, 6028)  0.21034888000987115
(4456, 7155)  0.24083218452280053
(4456, 7151)  0.3677554681447669
(4456, 6249)  0.17573831794959716
(4456, 6307)  0.2752760476857975
(4456, 334)   0.222007771165493 8
(4456, 5778)  0.16243064490100795
```

Figure 3.12: X Train Features

3.12 Training Models

Logistic Regression

```
 model = LogisticRegression()
 model.fit(X_train_features, Y_train)
 LogisticRegression(C=1.0, class_weight=None, dual=False,
  ↪  fit_intercept=True,
                  intercept_scaling=1, l1_ratio=None, max_iter=100,
                  multi_class='auto', n_jobs=None, penalty='l2',
                  random_state=None, solver='lbfgs', tol=0.0001,
                   ↪  verbose=0,
                  warm_start=False)
 prediction_on_training_data = model.predict(X_train_features)
 accuracy_on_training_data = accuracy_score(Y_train,
  ↪  prediction_on_training_data)
 print('Accuracy Score : ', accuracy_on_training_data)
 print('Precision score: ', format(precision_score(Y_train,
  ↪  predictions)))
print('Recall score: ', format(recall_score(Y_train, predictions)))
print('F1 score: ', format(f1_score(Y_train, predictions)))
print('\nConfusion Matrix :\n', confusion_matrix(Y_train, predictions))
```

```
 #output:
 Accuracy score:  0.9670181736594121
 Precision score:  0.9642857142857143
 Recall score:  0.9989650711513584
 F1 score:  0.98131910026687
```

```
Confusion Matrix :
 [[ 449  143]
 [   4 3861]]
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
model1= RandomForestClassifier()
model1.fit(X_train_features, Y_train)
prediction_on_training_data = model1.predict(X_train_features)
accuracy_on_training_data = accuracy_score(Y_train,
 ↪  prediction_on_training_data)
print('Accuracy on training data : ', accuracy_on_training_data)



#output: Accuracy on training data :  1.0
```

Naive Bayes Classification

```
import time
from sklearn.naive_bayes import GaussianNB
start_time = time.time()
NB = GaussianNB()
NB.fit(X_train_features.toarray(),Y_train)
end_time = time.time()
print("Training time:", end_time - start_time, "seconds")

#output:Training time: 0.891627311706543 seconds
```

```
prediction_on_training_data = NB.predict(X_train_features.toarray())
accuracy_on_training_data = accuracy_score(Y_train,
 ↪  prediction_on_training_data)
```

```
print('Accuracy on training data : ', accuracy_on_training_data)


#output:Accuracy on training data :  0.9347094458155571
```

KNN Classification

```
from sklearn.feature_extraction.text import TfidfVectorizer
knn = KNeighborsClassifier()
knn.fit(X_train_features, Y_train)
prediction_on_training_data = knn.predict(X_train_features)
accuracy_on_training_data = accuracy_score(Y_train,
↪  prediction_on_training_data)
print('Accuracy on training data : ', accuracy_on_training_data)


#output: Accuracy on training data :  0.9201256450527261
```

3.13 Predictive System

Building A Predictive Model

```
input_mail = ["I've been searching for the right words to thank you for
↪  this breather. I promise i wont take your help for granted and will
↪  fulfil my promise. You have been wonderful and a blessing at all
↪  times"]
# convert text to feature vectors
input_data_features = feature_extraction.transform(input_mail)
# making prediction
prediction = model.predict(input_data_features)
print(prediction)
if (prediction[0]==1):
```

```
print('Ham mail')

else:

print('Spam mail')
```

```
output: Ham mail
```

Chapter 4

Conclusion and Future Work

4.0.1 Conclusion

From the results obtained we can conclude that an ensemble machine learning model is more effective in detection and classification of spam than any individual algorithms. We can also conclude that Logistic Regression is the best compared to other algorithms and classifications used. And finally we can say that spam detection can get better if machine learning algorithms are combined and tuned to needs.

4.0.2 Future Work

There are numerous applications to machine learning and natural language processing and when combined they can solve some of the most troubling problems concerned with texts. This application can be scaled to intake text in bulk so that classification can be done more effectively in some public sites. Other contexts such as negative, phishing, malicious, etc,. can be used to train the model to filter things such as public comments in various social sites. This application can be converted to online type of machine learning system and can be easily updated with latest trends of spam and other mails so that the system can adapt to new types of spam emails and texts