MTU

Student Name: Job Thomas Thekkekara

Student No: R00195427

For the module COMP9060_26651 as part of the

Master of Science in Data Science and Analytics, Department of Mathematics, 2021/22

Declaration of Authorship

I, Job Thomas Thekkekara, declare that the work submitted is my own.

- I declare that I have not obtained unfair assistance via use of the internet or a third party in the completion of this examination.
- I acknowledge that the Academic Department reserves the right to request me to present for oral examination as part of the assessment regime for this module.
- I confirm that I have read and understood the policy and procedures concerning academic honesty, plagiarism and infringements.
- I understand that where breaches of this declaration are detected, these will be reviewed under MTU (Cork) policy and procedures concerning academic honesty, plagiarism and infringements, as well as any other University regulations and policies which may apply to the case. I also understand that any breach of academic honesty is a serious issue and may incur penalties.
- EXAMINATION/ASSESSMENT MATERIAL MAY, AT THE DISCRETION OF THE INTERNAL EXAMINER, BE SUBMITTED TO THE UNIVERSITY'S PLAGIARISM DETECTION SOLUTION
- Where I have consulted the published work of others, this is always clearly attributed
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this work is entirely my own work
- I have acknowledged all main sources of help

Signed: Job Thomas Thekkekara

Date: 13/03/2022

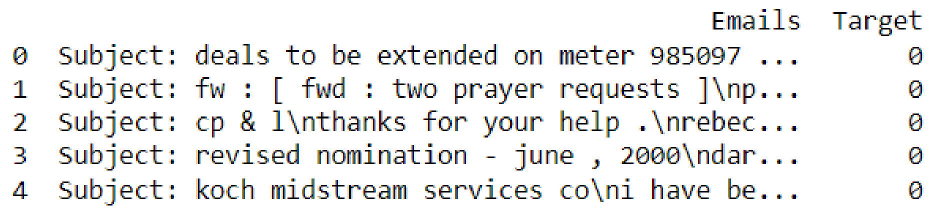
Introduction

In this era of technology, it is vital to reduce the chances of being attacked from spam emails. Natural language processing (NLP) a branch of artificial intelligence is extensively used to separate spam messages from non-spam or popularly known as ham messages. Hence the primary goal here is to pre-process the text data and to build a supervised classification pipeline to classify emails as spam and non-spam messages.

The dataset is a collection of public emails from Enron corporation. For a supervised machine learning algorithm, it is vital that the target labels are classified prior for the model training.

All the implementation is done using Python programming language. First, two sets of data files (ham and spam) were read into two separate pandas data frames. Ham data frame contained 3672 records and 2 columns- the email and the target label Ham. Spam data frame contained 1500 records and 2 columns. The target label in ham was assigned as 0 while the target label for spam was assigned as 1. Both the data frames were concatenated and shuffled for randomness to create one single data frame of all the emails. It contained 5172 emails.

First a check for duplicate records were done. 178 emails were found as duplicates and were removed, decreasing the total number of records to be 4994.



```
                                              Emails  Target
0  Subject: deals to be extended on meter 985097 ...       0
1  Subject: fw : [ fwd : two prayer requests ]\np...       0
2  Subject: cp & l\nthanks for your help .\nrebec...       0
3  Subject: revised nomination - june , 2000\ndar...       0
4  Subject: koch midstream services co\ni have be...       0
```

Fig1: Sample data of the data frame after reading

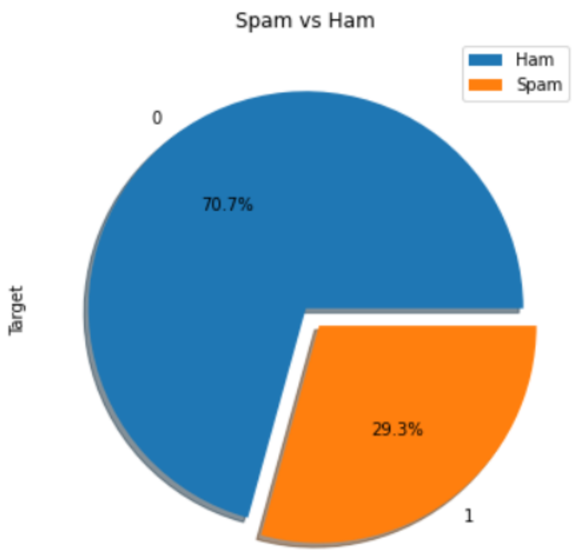


Fig 2: Distribution of ham and spam emails

It is also observed that the data set is imbalanced, as about 70% of the data contains ham messages. This would create certain difficulties in classifying minority class i.e., spam as the number of data points for training is less here.

Text Pre Processing

Lower Case Conversion: One of the most common pre-processing steps is the conversion of words into lower case. This makes sure that the number of words in the corpus does not become way too big with upper- and lower-case words. Although some time it may lead to loss of information in applications where emotions of a person is important, here it still can be applied.

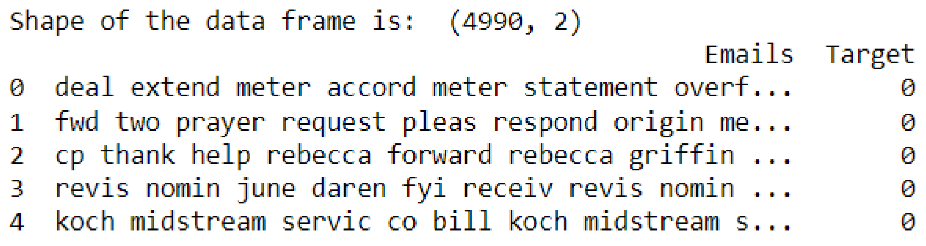
Removal of Punctuations: All punctuations from the text are removed as part of cleaning process. String library in python was used for this

Tokenization: Tokenization is the process of separating a piece of text into smaller chunks called tokens. A token can be a word, character or a set of characters. In a natural language text, a token is the feature of that text. Token occurrences in a document can be used as a vector for feature extraction and building machine learning models. Word tokenization is the most commonly used algorithm where the text is split into words based on a delimiter. Here word_tokenize library from nltk.tokenize is employed to do this task.

Stop words removal: Some of the most commonly used words in English such as 'the' 'is', 'to' etc does not add or add very less value to the analysis and at the same time masks the usage of some other important words. Hence these words are removed from the text so that the model can be trained and tested on words that are important for the analysis. Common stop words are available in Python libraries for usage. Stop words library from nltk.corpus is used to remove the common words. Additionally, words such as 'subject','re','fw' etc are added to the stop words list to remove them

Stemming: Stemming is the process of reducing a word into its root form by removing the affixes. Often the root forms of words have more importance and hence stemming is done.

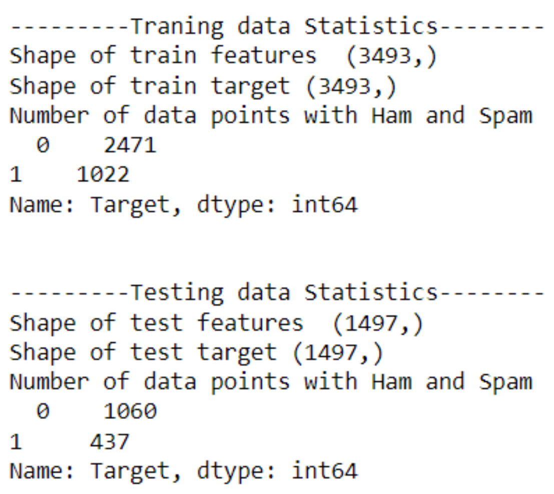
Additional pre processing steps such as removing words that contain numbers, removing words with only one letter, removing empty tokens etc was also performed to enhance the text processing. Weblinks were not removed as it contains words like http and www that mostly occurs in spam emails. Hence to avoid information loss they have been retained. Finally, the tokens were joined back together for feature extraction.



```
Shape of the data frame is:  (4990, 2)
                                              Emails  Target
0  deal extend meter accord meter statement overf...       0
1  fwd two prayer request pleas respond origin me...       0
2  cp thank help rebecca forward rebecca griffin ...       0
3  revis nomin june daren fyi receiv revis nomin ...       0
4  koch midstream servic co bill koch midstream s...       0
```

Fig3: Data frame after pre processing

After pre processing the data was split into training and testing data sets. Sklearn train_test_split functionality was used for this. 30% of the data was set aside for testing purpose of the model.



```
---------Traning data Statistics--------
Shape of train features  (3493,)
Shape of train target (3493,)
Number of data points with Ham and Spam
  0    2471
1    1022
Name: Target, dtype: int64


---------Testing data Statistics--------
Shape of test features  (1497,)
Shape of test target (1497,)
Number of data points with Ham and Spam
  0    1060
1     437
Name: Target, dtype: int64
```

Fig 4: Train and test data statistics

Total number of data points in train were 3493 (2471 as ham and 1022 as spam). Total number of test data points were 1497 (1060 as ham and 437 as spam). Testing data was kept as a separate entity and data exploration and model selection was done only on training data

Data Exploration

For data exploration a new feature 'Length' was created that carries the length of each email.

Using that, the average length of Spam messages was found to be 780, while the average length of non-spam/ham messages were found to be 497.

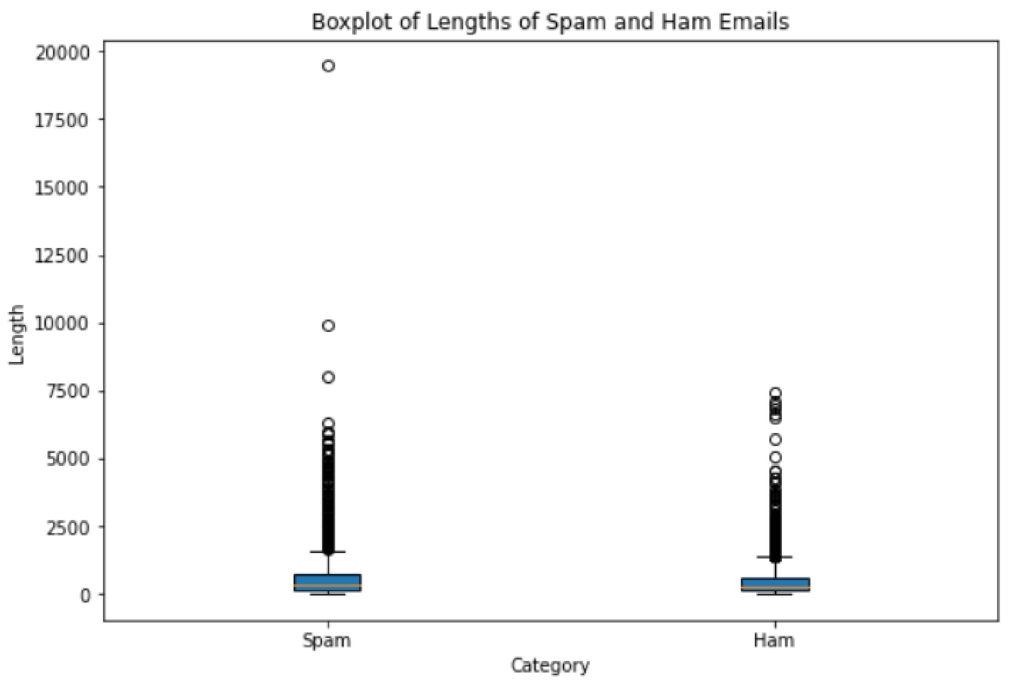
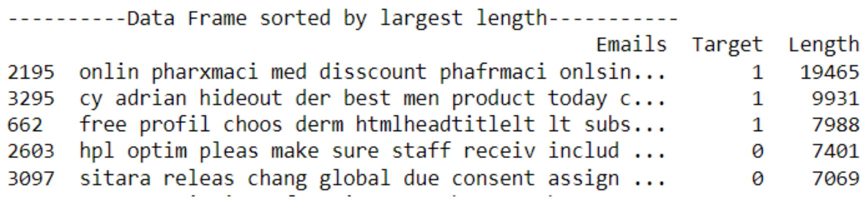


Fig 5: Boxplot showing length of email in spam and ham

The above boxplot shows that there some extreme outliers exist in both ham and spam messages in terms of length. The most extreme outlier in spam message was about online pharmacy medical discounts. This was kind of an advertisement with many random words and drug names in it.



```
----------Data Frame sorted by largest length-----------
                                                   Emails  Target  Length
2195  onlin pharxmaci med disscount phafrmaci onlsin...       1   19465
3295  cy adrian hideout der best men product today c...       1    9931
662   free profil choos derm htmlheadtitlelt lt subs...       1    7988
2603  hpl optim pleas make sure staff receiv includ ...       0    7401
3097  sitara releas chang global due consent assign ...       0    7069
```

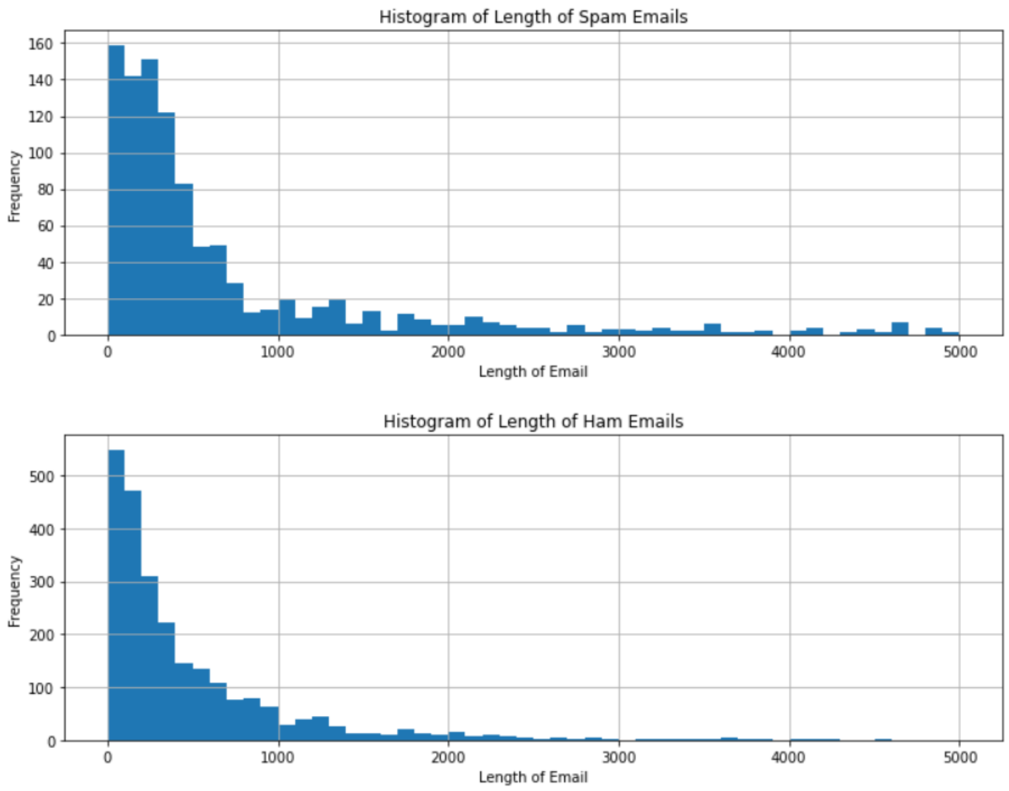
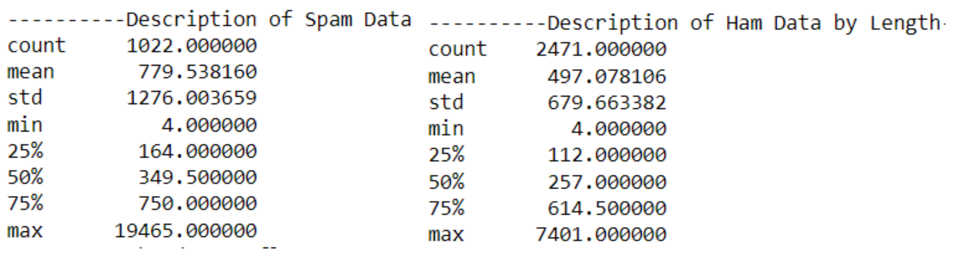


Fig 7: Frequency Distribution of length of email in spam and ham

On checking the frequency distribution of spam and ham messages right skewness is seen and most of the email length was found to be less than 1000 characters. Interestingly it was also observed that many of the spam emails had longer length when compared to non-spam emails.



```
----------Description of Spam Data      ----------Description of Ham Data by Length
count     1022.000000                   count    2471.000000
mean       779.538160                   mean      497.078106
std       1276.003659                   std       679.663382
min          4.000000                   min         4.000000
25%        164.000000                   25%       112.000000
50%        349.500000                   50%       257.000000
75%        750.000000                   75%       614.500000
max      19465.000000                   max      7401.000000
```

Fig 8: Statistics of length of email in spam and ham

```
----------Description of Entire Data by Length-
count     3493.000000
mean       579.721729
std        905.161568
min          4.000000
25%        125.000000
50%        290.000000
75%        657.000000
max      19465.000000
```

Fig 9: Distribution of length of email in entire data

The mean and standard deviation of spam messages were found to be much higher.

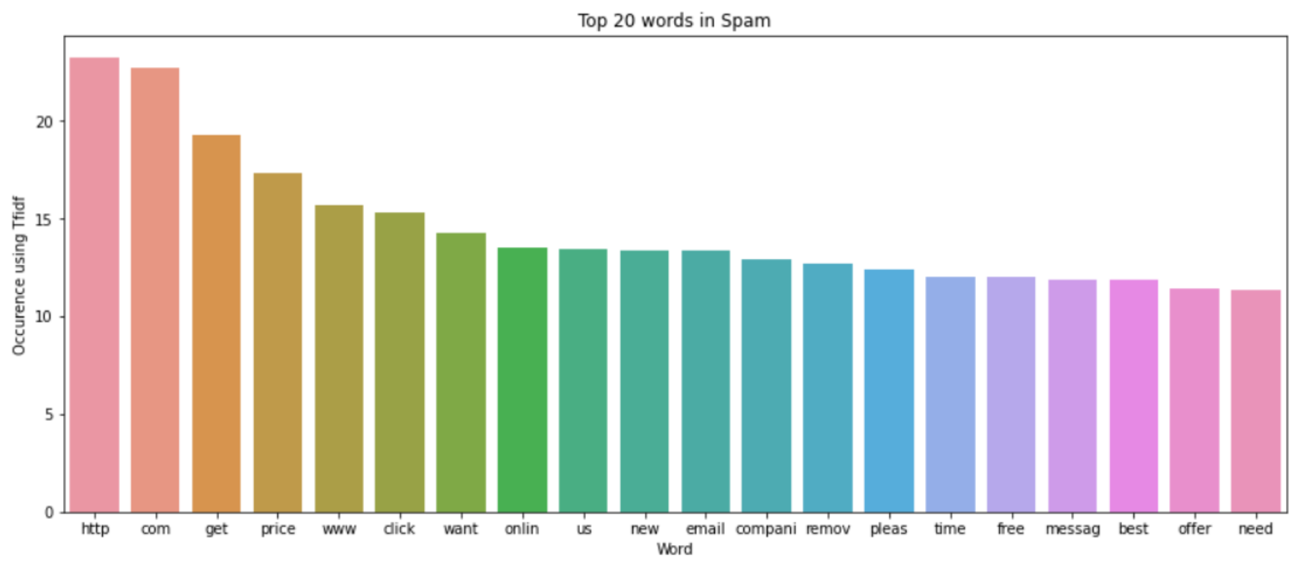


Fig 10: Most frequently used words in spam

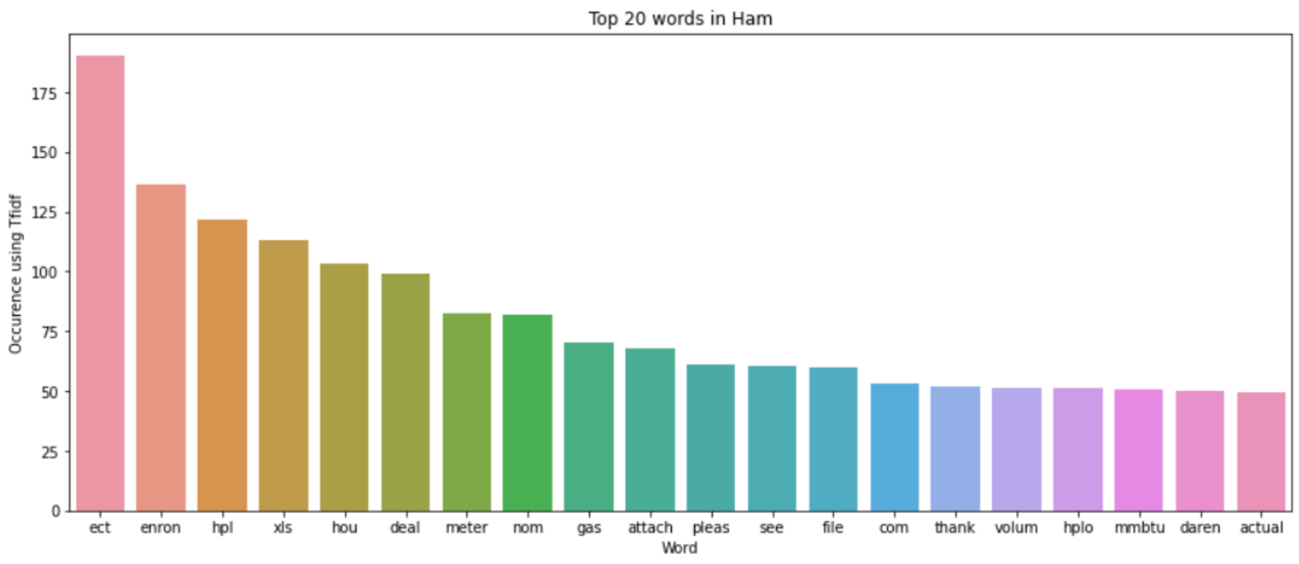


Fig 11: Most frequently used words in Ham

The most used words in ham emails were the company name and general words that employees exchange in emails in a workplace e.g.: please, attach, file, thank etc. The most used words in spam emails like http, price, click, new, best, offer etc are usually found in click baits and advertisements. It also proves that spam emails generally contain weblinks to click as the most common words were http, www and com.

Feature Extraction

Feature extraction is the most important part in natural language processing. As machine learning models do not understand human language and texts it is important to convert the text into numbers. Here words are the features. There are several ways to do feature extraction. Two of the most popular methods are:

1) Bag of words model
2) TF-IDF

Bag of words model: The main concept in this method is to take the entire text and count its frequency of occurrence. Finally, it maps each word with its corresponding frequency. The default model treats all words equally and cares only about how many times a word has occurred. This may cause some bias as random words can have high frequency. Sklearn package gives CountVectorizer module that implements bag of words model. On fitting training features to CountVectorizer, it creates a dictionary of tokens (words) that maps each single token to its relative position. On transforming training features, an encoded feature vector is produced containing the frequency of occurrence of each word. Test features are transformed on this, so that each word in test data also mapped to the same column as training set.

TF-IDF model: This model is more advanced, and it based on how relevant a word is to a document in a collection of documents. It doesn’t merely count the occurrence of words. It has two components

TF- How many times a word occurs in a document. The log frequency of weight term ‘t’ in document ‘d’ is **LOG(tf)**

IDF- This is used to account for less frequent words which are more informative and penalize the words which are more frequent that hardly convey any meaning. It has two steps

1) Calculate document frequency – No of documents in which word has occurred. (df)
2) Total document count N

IDF = LOG(N/df)

E.g.: If the total number of documents were 100. A word ‘the’ occurs 95 times and the word ‘cancer’ occurs 2 times.

Idf(the) = log (100/95) =0.02, idf(cancer) = log(100/2) = 1.7

Thus, it penalizes the more frequent unimportant word

TF-IDF = LOG(1+tf) * LOG(N/df)

Sklearn package provides TfidfVectorizer module that implements this algorithm. The train set features is fit transformed on this while the test set features is transformed on the same.

After fitting and transforming training data it is ready for model building and selection.

Model Selection

For model selection the transformed training data is given to 4 different classification models for comparison. The models selected were Logistic Regression, Multinomial Naïve Bayes, Support vector Machine and Decision tree. These are some of the most commonly used basic classification models.

Logistic Regression is a statistical model that finds an equation to predict the outcome of a binary variable. Unlike linear regression logistic regression can be used for classification of categorical data. LR uses log odds ratio and an iterative maximum likelihood method to fit the model. In NLP logistic regression is the base line supervised machine learning algorithm for classification

Multinomial Naïve Bayes: Naïve bayes is a classification model which assumes that each of the features it uses are conditionally independent of one another given some class. This model is based on Bayes theorem. Multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier which uses a multinomial distribution for each of the features. It is one of the most commonly used algorithms in NLP.

Support Vector Machine: SVM is a supervised machine algorithm that is commonly used for classification. The main goal in SVM is to find the best hyperplane that can separate n-dimensional space into classes so that when a new data point comes it can be put in the right class.

Decision Tree: In this model the predictor space is segmented into a number of simple regions. This segmenting is done by a set of splitting rules and criterions. As the process of decision making based on rules to segment the predictor space can be summarized and visualized in the form of an inverted tree starting from root to branches to leaves, such methods are known as decision tree methods.

Cross validation is a technique to measure the effectiveness of the model created. It is a re-sampling technique of the data to evaluate the performance of the model. Cross validation helps to avoid overfitting by re-sampling the training data again and again. Also, it can give out an average accuracy expected from the model by using only the training data. There are different techniques for this. But one of the most common is K- Folds cross validation K-Folds Cross validation. This technique is very easy to understand because every data point in our data set can come in both training and validation set by re-sampling. This method becomes very important when we have less data points to train and test. It follows the following steps:

1) Split the training data set randomly into K sets or folds. The value of K should not be too high or too low

2) Train the required model using k-1 data sets and validate on the remaining fold. For instance, if the data set is divided into 6 folds, The first 5 folds become the training data and 6th fold

becomes the validation data in first case. In the second case, 1st fold becomes the validation data and remaining becomes the train set

3) This process is continued until every fold becomes the test.

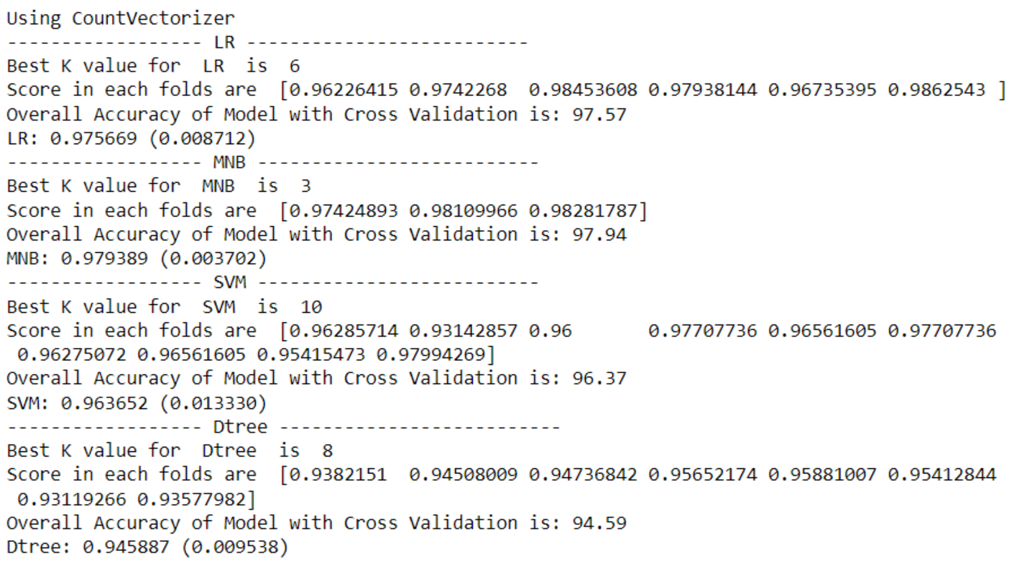
4) Finally, the average of results obtained in all the folds is calculated. This will be used to assess the model performance.

Here for cross validation optimum value of K for each model is also calculated.

Using CountVectorizer

All analysis on training data was done based on models with default parameters

By using CountVectorizer on training data and building the model, following results were obtained.



```
Using CountVectorizer
------------------ LR -------------------------
Best K value for  LR  is  6
Score in each folds are  [0.96226415 0.9742268  0.98453608 0.97938144 0.96735395 0.9862543 ]
Overall Accuracy of Model with Cross Validation is: 97.57
LR: 0.975669 (0.008712)
------------------ MNB -------------------------
Best K value for  MNB  is  3
Score in each folds are  [0.97424893 0.98109966 0.98281787]
Overall Accuracy of Model with Cross Validation is: 97.94
MNB: 0.979389 (0.003702)
------------------ SVM -------------------------
Best K value for  SVM  is  10
Score in each folds are  [0.96285714 0.93142857 0.96       0.97707736 0.96561605 0.97707736
 0.96275072 0.96561605 0.95415473 0.97994269]
Overall Accuracy of Model with Cross Validation is: 96.37
SVM: 0.963652 (0.013330)
------------------ Dtree -------------------------
Best K value for  Dtree  is  8
Score in each folds are  [0.9382151  0.94508009 0.94736842 0.95652174 0.95881007 0.95412844
 0.93119266 0.93577982]
Overall Accuracy of Model with Cross Validation is: 94.59
Dtree: 0.945887 (0.009538)
```

Fig 12: Optimum K value and training accuracy for all 4 models after using countvectorizer

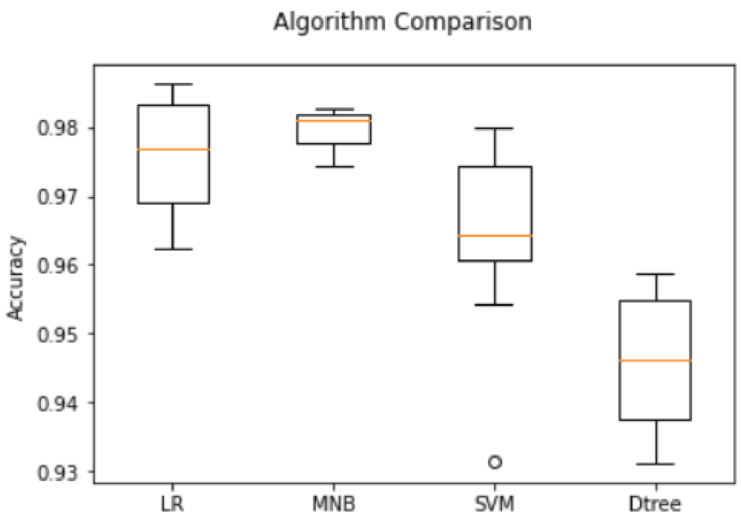
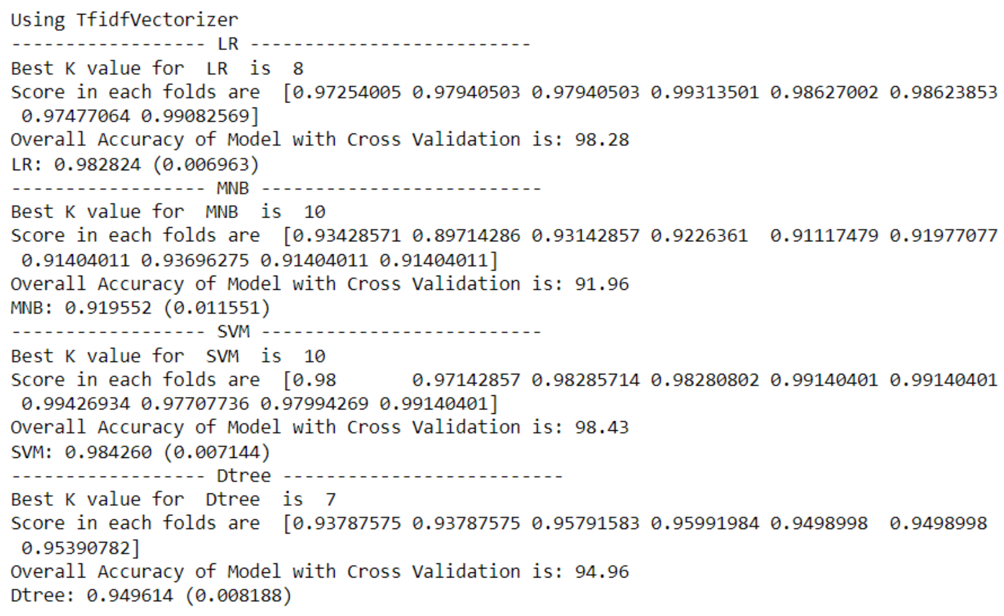


Fig 13: Box plots showing comparison by training accuracy for all models

Multinomial Naïve Bayes classifier was found to be performing better than others in terms of accuracy metric.

Using TfidfVectorizer



```
Using TfidfVectorizer
------------------ LR -------------------------
Best K value for  LR  is  8
Score in each folds are  [0.97254005 0.97940503 0.97940503 0.99313501 0.98627002 0.98623853
 0.97477064 0.99082569]
Overall Accuracy of Model with Cross Validation is: 98.28
LR: 0.982824 (0.006963)
------------------ MNB -------------------------
Best K value for  MNB  is  10
Score in each folds are  [0.93428571 0.89714286 0.93142857 0.9226361  0.91117479 0.91977077
 0.91404011 0.93696275 0.91404011 0.91404011]
Overall Accuracy of Model with Cross Validation is: 91.96
MNB: 0.919552 (0.011551)
------------------ SVM -------------------------
Best K value for  SVM  is  10
Score in each folds are  [0.98       0.97142857 0.98285714 0.98280802 0.99140401 0.99140401
 0.99426934 0.97707736 0.97994269 0.99140401]
Overall Accuracy of Model with Cross Validation is: 98.43
SVM: 0.984260 (0.007144)
------------------ Dtree -------------------------
Best K value for  Dtree  is  7
Score in each folds are  [0.93787575 0.93787575 0.95791583 0.95991984 0.9498998  0.9498998
 0.95390782]
Overall Accuracy of Model with Cross Validation is: 94.96
Dtree: 0.949614 (0.008188)
```

Fig 14: Optimum K value and training accuracy for all 4 models after using Tfidfvectorizer

Here, Multinomial Naïve bayes was performing comparatively poorly but Logistic Regression and SVM models were performing extremely well on accuracy metric. Final choice came down to Logistic Regression or SVM models as these two models had the highest training accuracy.

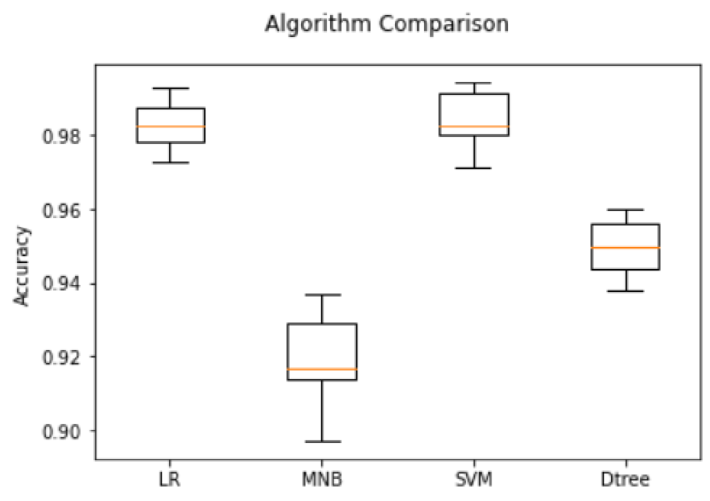


Fig 15: Box plots showing comparison by training accuracy for all models

Hyper Parameter Tuning

Model tuning is conducted by tweaking hyper parameters to improve the performance of the model. Hyperparameters are used to evaluate the optimal parameters of the model. GridSearchCv is a tool provided by Sklearn package. It uses a different combination of all the hyperparameters values specified and calculates the performance for each combination and selects the best value for hyperparameters. Cross validation is also performed with GridSearchCv. To implement this, a model pipeline is created with TfidfVectorizer() for feature extraction and the list of models are fed into the pipeline. Then GridSearchCv with cross validation is performed on this pipeline of algorithms. The scoring metric given was accuracy and AUC (area under the curve). Hence GridSearchCv will score models based on this with optimally tuned hyperparameters.

On default parameters (without explicitly specifying dictionary of hyperparameters) for each model, GridSearchCv gave extremely good result. SVM model was found to be the best model with a **training accuracy of 99.84%**

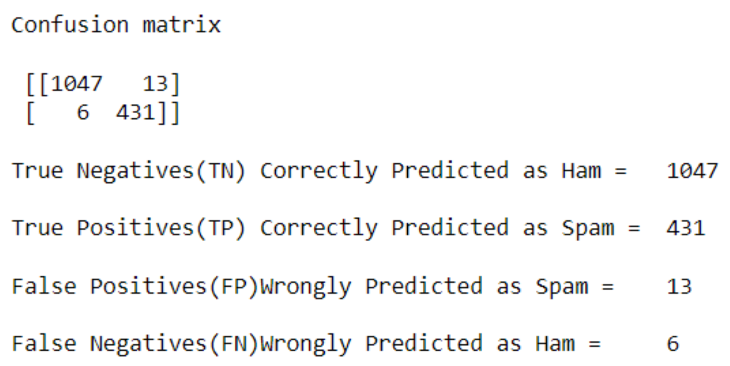
A more extensive hyperparameter tuning was conducted by specifying different hyper parameters to find the best model. But the training accuracy did not improve much, and it was computationally intensive. Hence decided not to proceed with that model for testing. The observation was that using a pipeline for model selection using GridSearchCV has improved the results significantly.

The best model was saved into disk using pickle function. Hence this pre trained model can be used again at any time without doing all the training steps. Only the best model SVM was used to for predictions using test data, and it gave a **test accuracy of 98.73%**

Model Evaluation

After finding the best model SVM and predicting the test results using test features, it is necessary evaluate the model by comparing it with original test data target values. There are several metrics for this model evaluation.

Confusion matrix is a table that is used to evaluate the performance of a classification model. On printing confusion matrix, following result was obtained. In any classification model there would be a positive class and a negative class. Here positive class is defined as ‘Spam’ (1) credit and negative class is defined as ‘Ham’ (0).



```
Confusion matrix

 [[1047   13]
 [   6  431]]

True Negatives(TN) Correctly Predicted as Ham =   1047

True Positives(TP) Correctly Predicted as Spam =  431

False Positives(FP)Wrongly Predicted as Spam =    13

False Negatives(FN)Wrongly Predicted as Ham =     6
```

Fig 16: Confusion Matrix produced after on testing data for best model SVM

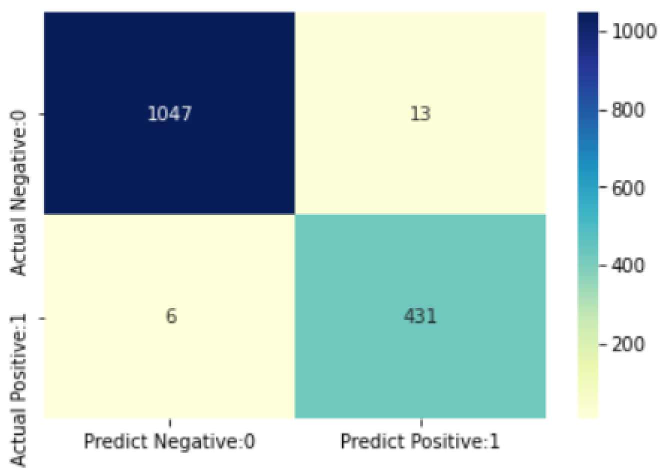


Fig 17: Confusion Matrix visual representation

From confusion matrix following interpretations can be made

Accuracy of the model is given by $TP+TN\ /TP+TN+FP+FN$

Accuracy metric depicts how many were correctly predicted by the model out of the total predictions. Here accuracy of the model on test data was found to be **98.73%**

Recall/Sensitivity= $TP/\ TP+F$N

Recall tells us how many actual positive classes were there in the entire testing data and out of that how many were predicted correctly. **Recall was found to be 98.63%**

Precision= $TP/TP+F$P

Precision tells how much good the model is in predictions. Precision talks only about predicted classes. Here **precision was found to be 97.07%**

There is a trade-off between precision and recall metrics in machine learning model. When precision increases recall decreases. And if cut offs were changes to increase recall, precision goes down.

Classification error was found to be 1.27%. That is, only 1.27% of data points were misclassified. Here 19 data points were misclassified.

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1060
           1       0.97      0.99      0.98       437

    accuracy                           0.99      1497
   macro avg       0.98      0.99      0.98      1497
weighted avg       0.99      0.99      0.99      1497
```

Fig 18: Classification report of the best model

Classification report clearly tells that there were 1060 data points for Ham and 437 for spam in test data. In a highly imbalanced dataset such as this it is important to look into precision and recall of minority class instead of just checking accuracy. It is observed that precision of minority class spam is slightly less than of ham. But still 97% is an extremely good result considering the imbalance.

ROC-AUC Curve

Roc- Receiver operating characteristic and AUC- Area under the curve. ROC curve shows true positive (y axis) vs false positive (x axis) rate for the model. If AUC is more the model is better.

The area under the curve was found out to be 99.83% which is really good.

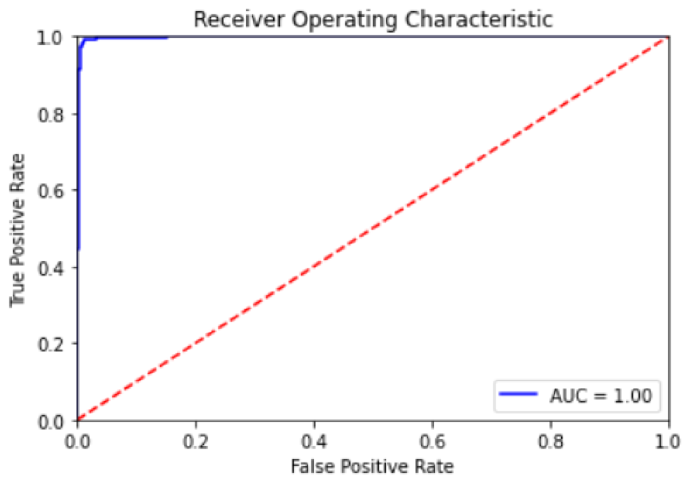


Fig 18: Roc-Auc Curve for svm model

References

https://medium.com/@cristhianboujon/how-to-list-the-most-common-words-from-text-corpus-using-scikit-learn-dad4d0cab41d

https://www.analyticsvidhya.com/blog/2021/11/top-7-cross-validation-techniques-with-python-code/

https://stackoverflow.com/questions/38692520/what-is-the-difference-between-fit-transform-and-transform-in-sklearn-countvecto

https://machinelearningmastery.com/training-validation-test-split-and-cross-validation-done-right/

https://www.kaggle.com/prashant111/naive-bayes-classifier-in-python/notebook

https://stackoverflow.com/questions/39163354/evaluating-logistic-regression-with-cross-validation

https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/

https://medium.datadriveninvestor.com/improve-the-text-classification-results-with-a-suitable-preprocessing-step-gridsearchcv-and-f19cb3e182a3 https://towardsdatascience.com/how-to-tune-multiple-ml-models-with-gridsearchcv-at-once-9fcebfcc6c23

https://www.analyticsvidhya.com/blog/2022/01/nlp-tutorials-part-ii-feature-extraction/

https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

https://link.springer.com/referenceworkentry/10.1007/978-1-4899-7993-3_565-2?noAccess=true

https://link.springer.com/chapter/10.1007/978-3-540-30549-1_43

https://arxiv.org/abs/2005.14627

Appendix

Make changes in code to change file path and restart kernel to run all cells in jupyter notebook. OR

Execute Notebook in the following order:

First execute function definitions in each cell

- Function to read files - **read_file**(path_ham,path_spam)
- Function to check data structure- **data_check**(df)
- Function to pre-process data - **text_cleaning(**final_df,stop_words=False,stem=False)
- Function for Length Calculation & Distribution Plots - **data_exploration(**x)
- Function To Analyze Emails by Length - **largest**(df)
- Function To Find Top Words in Ham and Spam - **top_words**(df_train_vec)
- Function to Compare models Based on CountVectorized Data Using Cross Validation- **modelselection_cv**(models,features_train_cv,result_train)
- Function to Compare models Based on TfidfVectorized Data Using Cross Validation- **modelselection_tf**(models,features_train_tf,result_train)
- Function for Model Selection - **model_selection**(features_train_cv,features_train_tf,result_train)
- Function for Creating Model Pipeline and Using GridSearchCv - **model_pipeline**(features_train_g,result_train)
- Function for Creating Model Pipeline and Using GridSearchCv For More Hyper Parameter Tuning - **hyperparameter_tuning_extra**(features_train_g,result_train)
- Function To Save and Load Model - **save_model**(best_model), **import_model**()
- Function For Final Model Evaluation - **model_evaluation**(best_svm,features_test_g,result_test)

Next Run the cells in the order given below

- **Execute Main Implementation Cell** – This includes importing of libraries, selecting **file paths** and function calls for **read_file(), data_check(), text_cleaning() .** Also splitting of data into train and test and transformation of data using count vectorizer and tfidfvectorizer happens here.

- Execute function call for EDA cell - **data_exploration()**
- Execute function call for further EDA cell - **largest()**
- Execute Function Call to Find Important Words cell – **top_words()**
- Execute Function Call for Model Selection Using Cross Validation cell – **model_selection()**
- Execute Function Call To Find Best Model Using Pipeline And GridSearchCv- **model_pipeline()**
- Execute Function Call To Save and Load Best Model- **save_model(), import_model()**
- Execute Function Call To Evaluate Best Model - **model_evaluation()**