



PROJECT REPORT ON:
"EMAIL SPAM CLASSIFIER"

SUBMITTED BY
Safiya Firdose Khan

ACKNOWLEDGMENT

It is my deepest pleasure and gratification to present this report. Working on this project was an incredible experience that has given me a very informative knowledge regarding the data analysis process. All the required information and dataset are provided by **Flip Robo Technologies** (Bangalore) that helped me to complete the project. I want to thank my SME **KHUSHBOO GARG** for giving the dataset and instructions to perform the complete case study process.

Contents:

1. Introduction

- 1.1 Business Problem Framing:
- 1.2 Conceptual Background of the Domain Problem
- 1.3 Review of Literature
- 1.4 Motivation for the Problem Undertaken

2. Analytical Problem Framing

- 2.1 Mathematical/ Analytical Modeling of the Problem
- 2.2 Data Sources and their formats
- 2.3 Data Preprocessing Done
- 2.4 Data Inputs-Logic-Output Relationships
- 2.5 Hardware and Software Requirements and Tools Used

3. Data Analysis and Visualization

- 3.1 Identification of possible problem-solving approaches (methods)
- 3.2 Testing of Identified Approaches (Algorithms)
- 3.3 Key Metrics for success in solving problem under consideration
- 3.4 Visualization
- 3.5 Run and Evaluate selected models
- 3.6 Interpretation of the Results

4. Conclusion

4.1 Key Findings and Conclusions of the Study

4.2 Learning Outcomes of the Study in respect of Data Science

4.3 Limitations of this work and Scope for Future Work

5. Reference

1.INTRODUCTION

1.1 Business Problem Framing:

Today, spam has become a big internet issue and in recent times, it has been proven through statistics that spam has accounted more than 60% of all SMS movement. Furthermore, phishing spam messages has been causing problem for the security of end users, since they attempt to persuade them to surrender personal information such as passwords and bank account numbers, using parody messages which replicates the appearance of original message from trustworthy organizations. Data science can thus play a key role in this domain by protecting the users from such fraud SMS using various natural language processing techniques. Our goal is to build a prototype of spam filtering system that will classify spam SMS and ham (non spam) messages and thus help the client to distinguish between the two.

1.2 Conceptual Background of the Domain Problem

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,572 messages, tagged according being ham (legitimate) or spam.

Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the nonspam texts. It uses a binary type of classification containing the labels such as '**ham**' (nonspam) and **spam**. Application of this can be seen in Google Mail (GMAIL) where it segregates the spam emails in order to prevent them from getting into the user's inbox.

1.3 Review of Literature

Text classification is used to evaluate whether an incoming message or piece of mail should go directly to the inbox or the spam folder. It is the process of categorizing texts based on their contents. It is employed to arrange, organize, and classify text. Either manually or automatically can be done. Text is automatically classified by machine learning significantly more quickly than manually. Pre-labeled text is used by machine learning to learn the many connections between text and output. Each text was converted to a numerical representation in the form of a vector that represents the frequency of words in a predetermined lexicon using feature extraction. Most research has been conducted into detecting and filtering spam email using a variety of techniques.

1.4 Motivation for the Problem Undertaken

The project was provided to me by FlipRobo as a part of the internship program. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The motivation behind this project is to help the users by alerting them of the possibility of the spam SMS being received. The information obtained from the model can be used to report such contact and thus enhance the security of the user.

2. Analytical Problem Framing

2.1 Mathematical/ Analytical Modeling of the Problem

In this given problem the label can be either 0 or 1, where 0 denotes a 'ham' and 1 denotes a 'spam'. So clearly it is a binary classification problem and I have to use all classification algorithms while building the model. We would perform one type of supervised learning algorithms: Classification. Here, we will perform classification. Since there only 1 feature in the dataset, filtering the words is needed to prevent overfit. In order to determine the regularization parameter, throughout the project in classification part, we would first remove email, phone number, web address, spaces and stops words etc. In order to further improve our models, we also performed TFIDF in order to convert the tokens from the train documents into vectors so that machine can do further processing. I have used all the classification algorithms while building model then tuned the best model and saved the best model. At last, I have predicted whether the Email is spam or not, using the saved model.

2.2 Data Sources and their formats

The dataset consists of 5572 rows and 2 columns. Primarily, it has 5 columns, where 3 columns, i.e., "Unnamed: 2", "Unnamed: 3" and "Unnamed:4" are unnecessary. After deleting these 3 columns, the dataset consists of 2 columns, i.e., "v1" and "v2". I have renamed the column "v1" as "label", and the column "v2" as "message". Later the dataset is divided into two parts, training and testing. After determine the proper model, the model is applied to predict the target variable for the test dataset.

2.3 Data Preprocessing Done

Data processing and feature engineering are crucial in machine learning to build a prediction model. Furthermore, a model cannot be made without some data processing. For instance, as shown in the experiment, the model could not be trained before handling the missing values and converting the text in the dataset into numerical values. Hence, from the experiment, we saw that pre-processing the data does improve the prediction accuracy.

❖ **Encoding:** The target column, i.e., “label” is encoded into 0s and 1s.

```
In [16]: 1 # Label coding 0 and 1
          2
          3 email['label'].replace({'ham':0,'spam':1}, inplace=True)
```

❖ **Converting upper case to lower case:** I have converted all the uppercase text in the column “message” to lower case. It is an important technique as it helps in cleaning the data.

```
In [17]: 1 # Convert all messages to lower case
          2
          3 email['message'] = email['message'].str.lower()
```

❖ **Text Normalisation:** Let’s normalize the text data since messages from online forums usually contain inconsistent language, use of special characters in place of letters (e.g., @rgument), as well as the use of numbers to represent letters (e.g., n0t). So, let’s tackle such inconsistencies in data.

The text normalization steps that I performed are listed below: -

- Replaced email address with 'email'.
- Replaced URLs with 'webaddress'.
- Replaced money symbols with 'moneysymb'.
- Replaced 10 digit phone numbers (formats include parenthesis, spaces, no spaces, dashes) with 'phonenumbr'.
- Replaced numbers with 'numbr'.
- Removed punctuation.
- Replaced whitespace between terms with a single space.

- Removed Leading and trailing whitespace.

```
In [19]: 1 # Replace email address with 'email'
2 email['message'] = email['message'].str.replace(r'^.+@[^\.\.]*\.[a-z]{2,}$', 'emailaddress')
3
4
5 # Replace URLs with 'webaddress'
6 email['message'] = email['message'].str.replace(r'^http://[a-zA-Z0-9\-\.\.]+\.[a-zA-Z]{2,3}(/s*)?$', 'webaddress')
7
8
9 # Replace money symbols with 'moneysymb' (£ can be typed with ALT key + 156)
10 email['message'] = email['message'].str.replace(r'£|$', 'dollars')
11
12
13 # Replace 10 digit phone numbers (formats include parenthesis, spaces, no spaces, dashes) with 'phonenumbr'
14 email['message'] = email['message'].str.replace(r'^\([0-9]{3}\)?[0-9]{3}[0-9]{4}$', 'phonenumbr')
15
16
17 # Replace numbers with 'numbr'
18 email['message'] = email['message'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

```
In [20]: 1 # Remove punctuation
2 email['message'] = email['message'].str.replace(r'^[^\w\d\s]', ' ')
3
4
5 # Replace whitespace between terms with a single space
6 email['message'] = email['message'].str.replace(r'\s+', ' ')
7
8
9 # Remove Leading and trailing whitespace
10 email['message'] = email['message'].str.replace(r'^\s+|\s+$', ' ')
```

- ❖ **Stop word Removal:** Stop words are those words that are frequently used in both written and verbal communication and thereby do not have either a positive/negative impact on our statement. Stop words are a set of commonly used words in a language. Examples of stop words in English are “a”, “the”, “is”, “are” and etc. The intuition behind using stop words is that, by removing low information words from text, we can focus on the important words instead. For example, in the context of a search system, if your search query is “what is text processing?”, you want the search system to focus on surfacing documents that talk about text pre-processing over documents that talk about what is. This can be done by preventing all words from your

stop word list from being analyzed. Stop words are commonly applied in search systems, text classification applications, topic modelling, topic extraction and others.

```
In [22]: 1 # Remove stopwords
2 import string
3 import nltk
4 from nltk.corpus import stopwords
5
6 stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
7
8 email['message'] = email['message'].apply(lambda x: ' '.join(
9     term for term in x.split() if term not in stop_words))
```

❖ **Lemmatization:** Lemmatization on the surface is very similar to stemming, where the goal is to remove inflections and map a word to its root form. The only difference is that, lemmatization tries to do it the proper way. It doesn't just chop things off, it actually transforms words to the actual root. For example, the word “better” would map too “good”. It may use a dictionary such as word net for mapping or some special rule-based approaches. Here is an example of lemmatization in action using a WordNet-based approach:

	original_word	lemmatized_word
0	trouble	trouble
1	troubling	trouble
2	troubled	trouble
3	troubles	trouble

```
In [23]: 1 # Lemmatization
2
3 lem = WordNetLemmatizer()
4 email['message'] = email['message'].apply(lambda x: ' '.join(
5     lem.lemmatize(word) for word in x.split()))
```

❖ **Tf-Idf vectorization:** In NLP, tf-idf is an important measure and is used by algorithms like cosine similarity to find documents that are similar to a given search query.

- What is Term Frequency (tf): tf is the number of times a term appears in a particular document. So, it's specific to a document. A few of the ways to calculate tf is given below: -

$tf(t) = \text{No. of times term 't' occurs in a document.}$

- Inverse Document Frequency (idf): idf is a measure of how common or rare a term is across the entire corpus of documents. So, the point to note is that it's common to all the documents. If the word is common and appears in many documents, the idf value (normalized) will approach 0 or else approach 1 if it's rare. A few of the ways we can calculate idf value for a term is given below:

$idf(t) = \log_e \left[\frac{(1+n)}{(1 + df(t))} \right] + 1$ (default i: e smooth_idf = True)

and

$idf(t) = \log_e \left[n / df(t) \right] + 1$ (when smooth_idf = False)

TF-IDF

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$TF-IDF = TF(t, d) \times IDF(t)$$

Term frequency

Number of times term t appears in a doc, d

Inverse document frequency

$\log \frac{1 + n}{1 + df(d, t)} + 1$

n ← # of documents

$df(d, t)$ ← Document frequency of the term t

```
In [30]: 1 from sklearn.feature_extraction.text import TfidfVectorizer
2
3 tf = TfidfVectorizer()
4 features = tf.fit_transform(email["message"])
5 x = features
6 y = email["label"]
```

2.4 Data Inputs- Logic- Output Relationships

For this data's input and output logic, we will analyze the column "message", and study its relationship with the column "label".

2.5 Hardware and Software Requirements and Tools Used

While taking up the project we should be familiar with the Hardware and software required for the successful completion of the project. Here we need the following hardware and software.

Hardware required: -

- Processor — core i5 and above
- RAM — 8 GB or above
- SSD — 250GB or above

Software/s required: -

- Programming language: Python
- Distribution: Anaconda Navigator
- Browser based language shell: Jupyter Notebook

Libraries required :-

```
In [1]: 1 import numpy as np
        2 import pandas as pd
        3 import seaborn as sns
        4 import matplotlib.pyplot as plt
        5
        6
        7 import string
        8 import nltk
        9 from wordcloud import WordCloud
       10 from collections import Counter
       11 from sklearn.feature_extraction.text import TfidfVectorizer
       12
       13
       14 import warnings
       15 warnings.filterwarnings('ignore')
```

- ✓ **Pandas:** Pandas is a popular Python-based data analysis toolkit which can be imported using `import pandas as pd`. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a numpy matrix array. This makes pandas a trusted ally in data science and machine learning.
- ✓ **Seaborn:** Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.
- ✓ **Matplotlib:** `matplotlib.pyplot` is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.
- ✓ **Scikit Learn:** This is the most important library for Machine Learning since it contains various Machine Learning Algorithms which are used in this project. Scikit Learn also contains Preprocessing library which is used in data preprocessing. Apart from this, it contains a very useful joblib library for serialization purpose using which the final model has been saved in this project.

- ✓ **NLTK:** Natural language tool kit is one of the most used libraries for building NLP projects.

3. Data Analysis and Visualization

3.1 Identification of possible problem-solving approaches (methods)

I have made the messages more appropriate so that we'll get less words to process and get more accuracy. I have replaced email address with 'email', URLs with 'webaddress', money symbols with 'moneysymb', replaced 10 digit phone numbers (formats include parenthesis, spaces, no spaces, dashes) with 'phonenumber', replaced numbers with 'numbr', removed punctuation, replaced whitespace between terms with a single space and removed leading and trailing whitespace. Thus, I have tried to make the messages, small and more appropriate as much as possible.

3.2 Testing of Identified Approaches (Algorithms)

In this nlp-based project, we need to predict the targets which is binary. I have converted the text into vectors using TF-IDF vectorizer and separated our feature and labels. Then, I have built the model using One Vs Rest Classifier. Among all the algorithms that I have used for this purpose, I have chosen LinearSVC as best algorithm for our final model, as it is performing well, as compared to other algorithms. I have used following algorithms and evaluated them.

- Logistic Regression
- Decision Tree Classifier
- Linear Support Vector Machine Classifier
- AdaBoost Classifier
- KNeighbors Classifier
- Multinomial Naive Bayes Classifier

From all of these above models, Linear Support Vector Machine Classifier is giving me good performance.

3.3 Key Metrics for success in solving problem under consideration

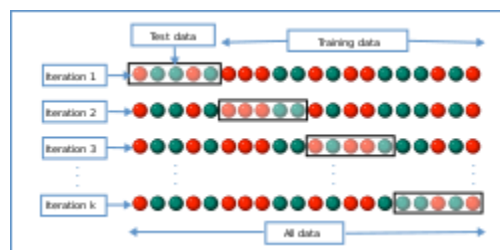
I have used the following metrics for evaluation:

- **Accuracy score:** Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition: $\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$ For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

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





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

- **Cross Validation Score:** Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

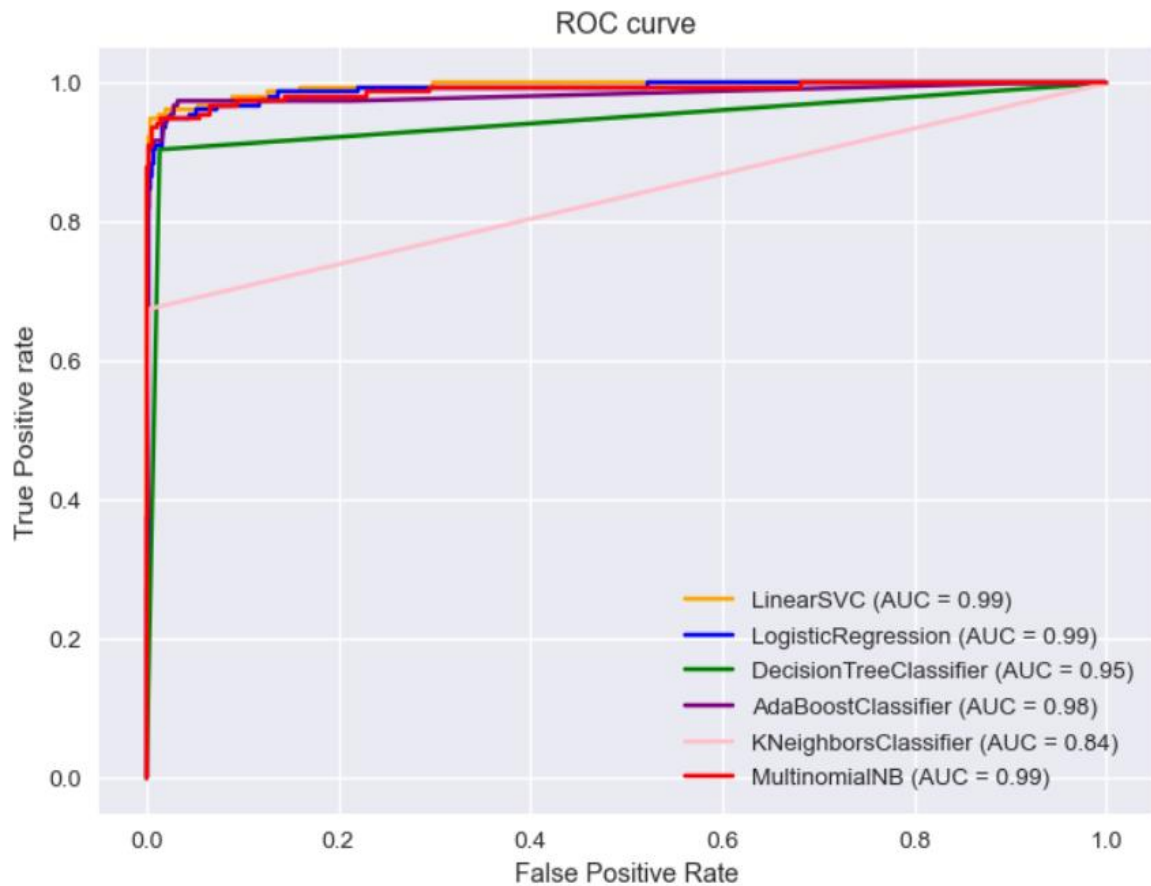


- **Confusion matrix:** A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data

for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. Well, it is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values. It is extremely useful for measuring Recall, Precision, Specificity, Accuracy, and most importantly AUC-ROC curves. Let's understand TP, FP, FN, TN in terms of pregnancy analogy. Log loss: Log Loss is the most important classification metric based on probabilities. ... For any given problem, a lower log-loss value means better predictions.

		Actual Values	
		1	0
Predicted Values	1	TRUE POSITIVE  You're pregnant	FALSE POSITIVE  You're pregnant TYPE 1 ERROR
	0	FALSE NEGATIVE  You're not pregnant TYPE 2 ERROR	TRUE NEGATIVE  You're not pregnant

- **AUC-ROC CURVE:** AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1.



3.4 Visualizations

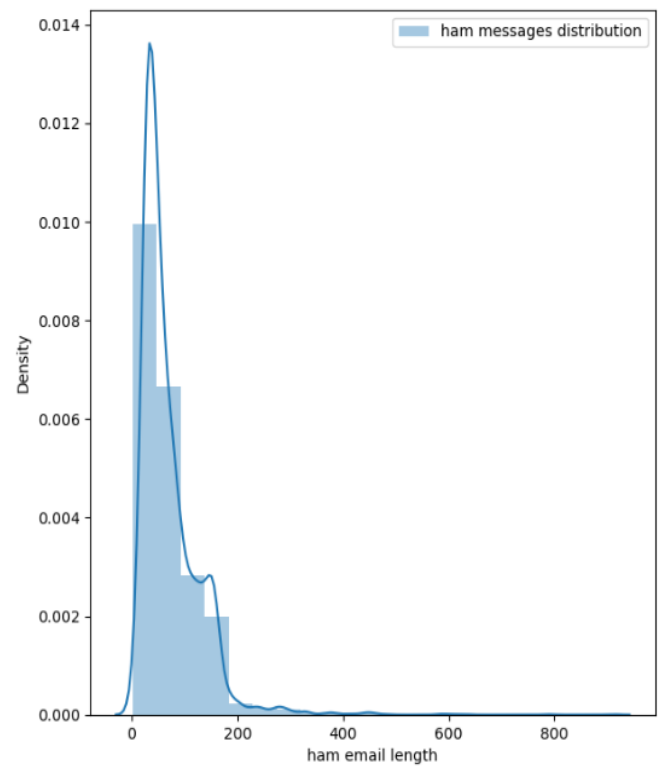
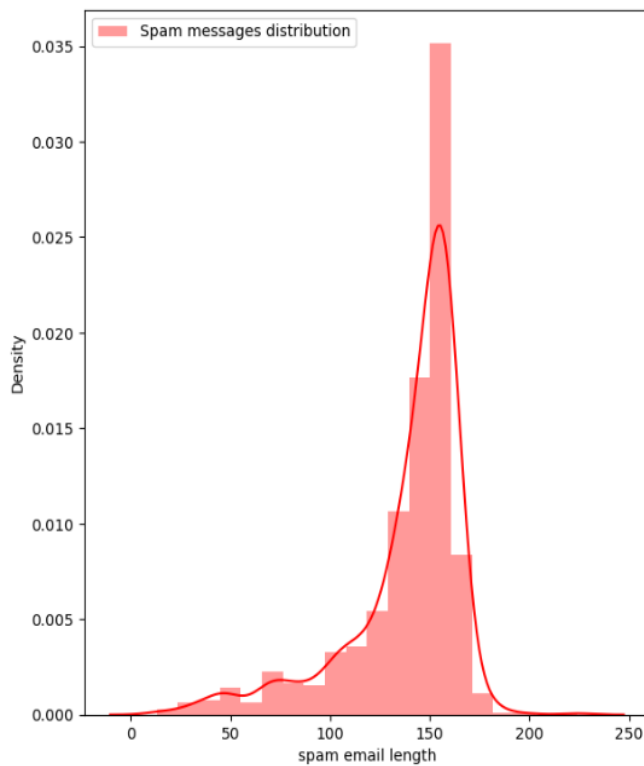
Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed.

The following distribution plot gives us an idea about Message distribution before cleaning.

```

In [26]: 1 # Message distribution BEFORE cleaning
2
3 f,ax = plt.subplots(1, 2, figsize = (15,8))
4
5 sns.distplot(email[email['label']=='1']['length'], bins=20, ax=ax[0], label='Spam messages distribution', color='r')
6 ax[0].set_xlabel('spam email length')
7 ax[0].legend()
8
9
10 sns.distplot(email[email['label']=='0']['length'], bins=20, ax=ax[1], label='ham messages distribution')
11 ax[1].set_xlabel('ham email length')
12 ax[1].legend()
13
14 plt.show()

```

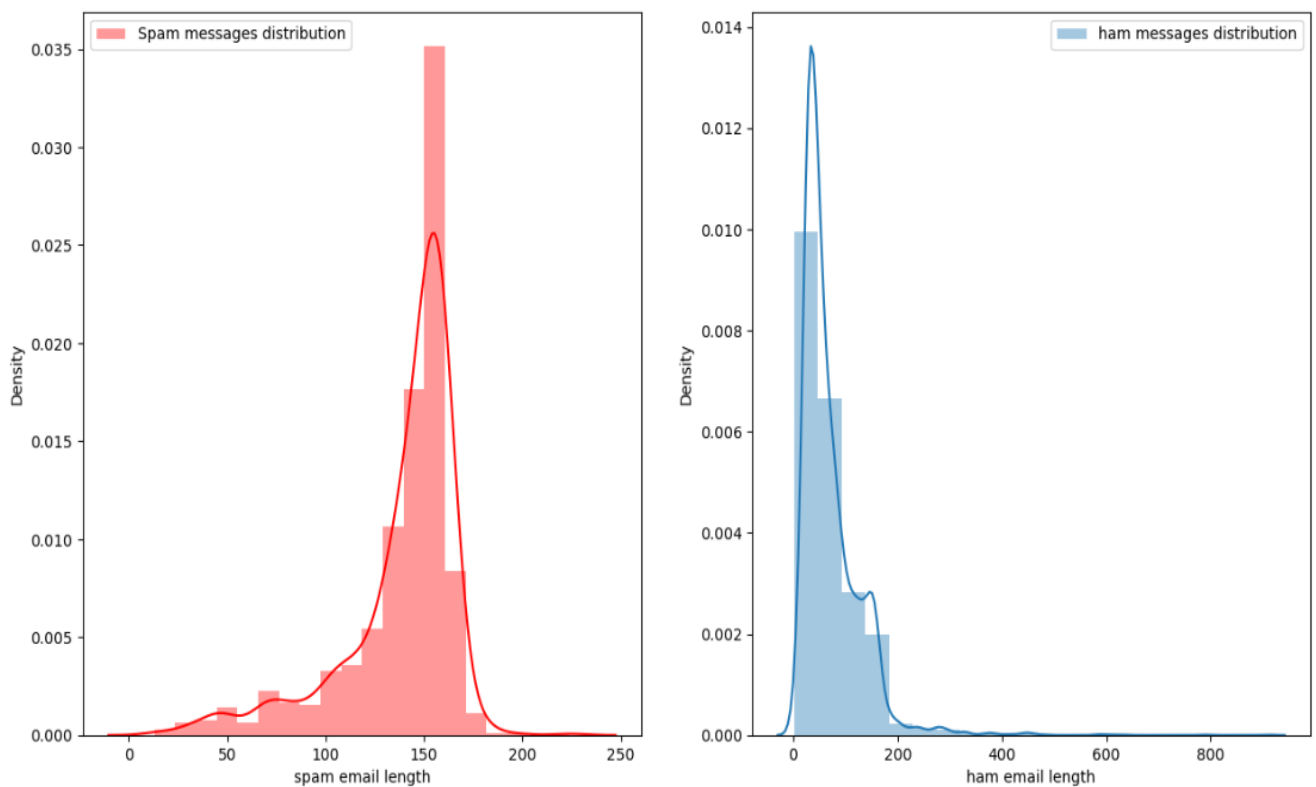


The following distribution plot gives us an idea about Message distribution after cleaning.

```

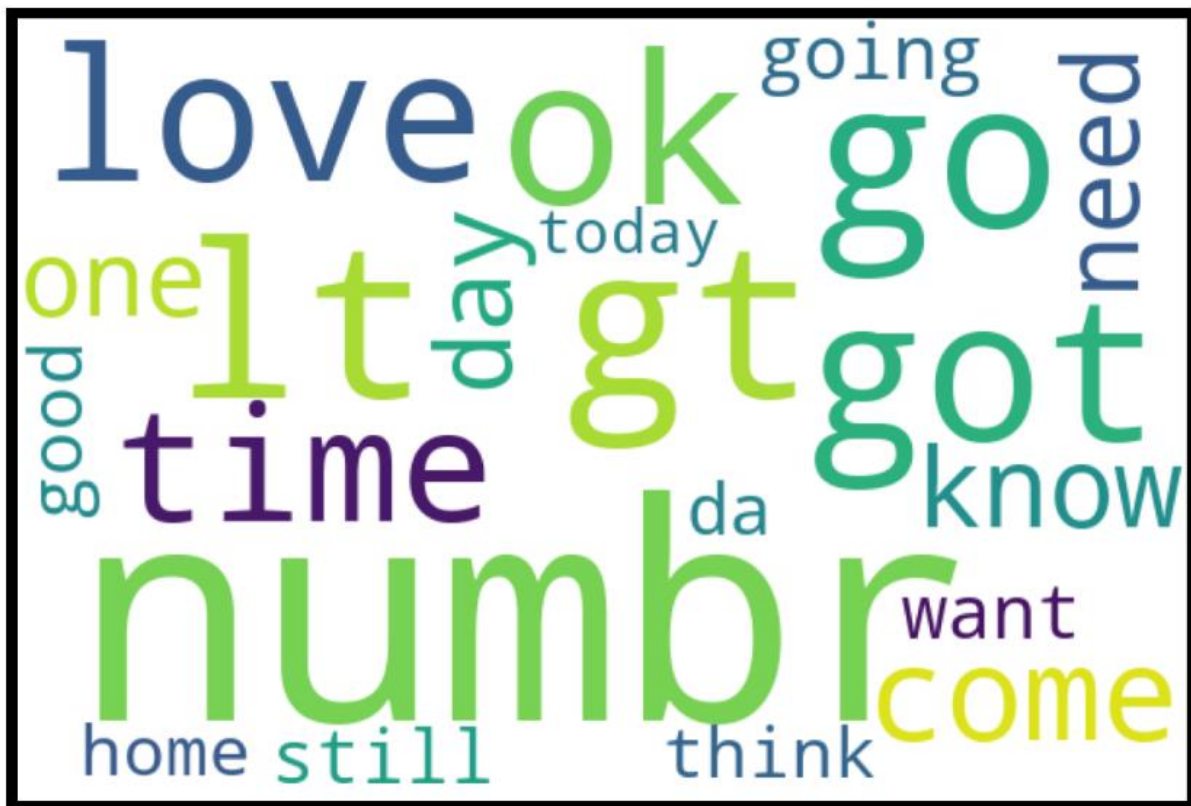
In [27]: 1 # Message distribution AFTER cleaning
2
3 f,ax = plt.subplots(1, 2, figsize = (15,8))
4
5 sns.distplot(email[email['label']==1]['length'], bins=20, ax=ax[0], label='Spam messages distribution', color='r')
6 ax[0].set_xlabel('spam email length')
7 ax[0].legend()
8
9
10 sns.distplot(email[email['label']==0]['length'], bins=20, ax=ax[1], label='ham messages distribution')
11 ax[1].set_xlabel('ham email length')
12 ax[1].legend()
13
14 plt.show()

```



Word Cloud: Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analyzing data from social network websites.


```
In [29]: 1 # Getting sense of loud words in ham
2
3 from wordcloud import WordCloud
4
5 hams = email['message'][email['label']==0]
6
7 spam_cloud = WordCloud(width=600, height=400, background_color='white', max_words=20).generate(' '.join(hams))
8
9 plt.figure(figsize=(10,8), facecolor='k')
10 plt.imshow(spam_cloud)
11 plt.axis('off')
12 plt.tight_layout(pad=0)
13 plt.show()
```



3.5 Run and Evaluate selected models

1. Finding best random state

```
In [31]: 1 from sklearn.model_selection import train_test_split
2 from sklearn.metrics import mean_absolute_error
3 from sklearn.metrics import mean_squared_error
4 from sklearn.metrics import r2_score
5 from sklearn.metrics import accuracy_score
6 from sklearn import metrics
```

```
In [32]: 1 from sklearn.linear_model import LogisticRegression
2
3 maxAccu = 0
4 maxRS = 0
5 for i in range(1,200):
6     xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=.25, random_state=i)
7     LR = LogisticRegression()
8     LR.fit(xtrain, ytrain)
9     pred = LR.predict(xtest)
10    acc=accuracy_score(ytest, pred)
11    if acc>maxAccu:
12        maxAccu=acc
13        maxRS=i
14 print("Best accuracy is ", maxAccu, " on Random_state ", maxRS)
```

Best accuracy is 0.9798994974874372 on Random_state 148

```
In [33]: 1 xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=.25, random_state=maxRS)
```

```
In [34]: 1 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
2 from sklearn.metrics import roc_curve, roc_auc_score
3 from sklearn.model_selection import cross_val_score
4 from sklearn.metrics import log_loss
```

2. Model Building

1) Logistic Regression:

```
In [37]: 1 a = accuracy_score(ytest, ypred)
2 c = cross_val_score(LR,x,y,cv=3).mean()
3 loss = log_loss(ytest,ypred)
4
5 print("accuracy_score : ",a,"\n","cross validation score :",c,"\n","logloss:",loss)
```

accuracy_score : 0.9798994974874372
cross validation score : 0.9675164077234503
logloss: 0.6942484837399793

```
In [38]: 1 confusion_matrix(ytest,ypred)
```

```
Out[38]: array([[1233,    3],
               [  25,  132]], dtype=int64)
```

2) Decision Tree Classifier:

```
In [44]: 1 a = accuracy_score(ytest, ypred)
2 c = cross_val_score(dt,x,y,cv=3).mean()
3 loss = log_loss(ytest,ypred)
4
5 print("accuracy_score : ",a,"\n","cross validation score :",c,"\n","logloss:",loss)

accuracy_score : 0.9770279971284996
cross validation score : 0.9741566883246491
logloss: 0.7934346287103062
```

```
In [45]: 1 confusion_matrix(ytest,ypred)
```

```
Out[45]: array([[1219, 17],
               [ 15, 142]], dtype=int64)
```

3) Linear Support Vector Machine Classifier:

```
In [50]: 1 a = accuracy_score(ytest, ypred)
2 c = cross_val_score(svc,x,y,cv=3).mean()
3 loss = log_loss(ytest,ypred)
4
5 print("accuracy_score : ",a,"\n","cross validation score :",c,"\n","logloss:",loss)

accuracy_score : 0.9885139985642498
cross validation score : 0.9822325903847369
logloss: 0.3967141572942304
```

```
In [51]: 1 confusion_matrix(ytest,ypred)
```

```
Out[51]: array([[1233, 3],
               [ 13, 144]], dtype=int64)
```

4) Ada Boost Classifier:

```
In [56]: 1 a = accuracy_score(ytest, ypred)
2 c = cross_val_score(abc,x,y,cv=3).mean()
3 loss = log_loss(ytest,ypred)
4
5 print("accuracy_score : ",a,"\n","cross validation score :",c,"\n","logloss:",loss)

accuracy_score : 0.9863603732950467
cross validation score : 0.9764894282806607
logloss: 0.47109831291674453
```

```
In [57]: 1 confusion_matrix(ytest,ypred)
```

```
Out[57]: array([[1232, 4],
               [ 15, 142]], dtype=int64)
```


5) K-Neighbors Classifier:

```
In [62]: 1 a = accuracy_score(ytest, ypred)
          2 c = cross_val_score(knn,x,y,cv=3).mean()
          3 loss = log_loss(ytest,ypred)
          4
          5 print("accuracy_score : ",a,"\n","cross validation score :",c,"\n","logloss:",loss)

accuracy_score : 0.9303661162957645
cross validation score : 0.9090088434668306
logloss: 2.405069138769804
```

```
In [63]: 1 confusion_matrix(ytest,ypred)

Out[63]: array([[1236,    0],
               [  97,   60]], dtype=int64)
```

6) Multinomial Naive Bayes Classifier:

```
In [68]: 1 a = accuracy_score(ytest, ypred)
          2 c = cross_val_score(mnb,x,y,cv=3).mean()
          3 loss = log_loss(ytest,ypred)
          4
          5 print("accuracy_score : ",a,"\n","cross validation score :",c,"\n","logloss:",loss)

accuracy_score : 0.9806173725771715
cross validation score : 0.967336616906829
logloss: 0.6694522345029359
```

```
In [69]: 1 confusion_matrix(ytest,ypred)

Out[69]: array([[1236,    0],
               [  27,  130]], dtype=int64)
```

Comparison of different models:

```
In [72]: 1 model = ["LogisticRegression", "DecisionTreeClassifier", "LinearSupportVectorMachineClassifier", "AdaBoostClassifier", "KNeigh
```

```
In [73]: 1 classifier = pd.DataFrame({"MODEL":model, "Accuracy score":accuracy, "Cross_validation_Score":cross_val, "Difference":diff, "Log  
2  
3 classifier
```

Out[73]:

	MODEL	Accuracy score	Cross_validation_Score	Difference	Logloss
0	LogisticRegression	0.979899	0.967516	0.012383	0.694248
1	DecisionTreeClassifier	0.977028	0.974157	0.002871	0.793435
2	LinearSupportVectorMachineClassifier	0.988514	0.982233	0.006281	0.396714
3	AdaBoostClassifier	0.986360	0.976489	0.009871	0.471098
4	KNeighborsClassifier	0.930366	0.909009	0.021357	2.405069
5	MultinomialNaiveBayesClassifier	0.980617	0.967337	0.013281	0.669452

As we can see, **Linear Support Vector Machine Classifier** is the best model.

Hyperparameter Tuning:

```
In [75]: 1 from sklearn.model_selection import RandomizedSearchCV
2
3 param = {'loss': ['hinge', 'squared_hinge'],
4          'C': [0.1, 1, 10, 100, 1000],
5          'multi_class': ['ovr', 'crammer_singer'],
6          'class_weight': ['dict', 'balanced']}
```

```
In [76]: 1 rand_search = RandomizedSearchCV(svc, param_distributions=param, cv=2)
```

```
In [77]: 1 rand_search.fit(xtrain, ytrain)
```

```
Out[77]: ▸ RandomizedSearchCV
          ▸ estimator: LinearSVC
              ▸ LinearSVC
```

```
In [78]: 1 rand_search.best_params_
```

```
Out[78]: {'multi_class': 'crammer_singer',
          'loss': 'hinge',
          'class_weight': 'balanced',
          'C': 10}
```

```
In [79]: 1 svc = LinearSVC(multi_class='crammer_singer', loss='hinge', class_weight='balanced', C=10)
2 svc.fit(xtrain, ytrain)
3
4 ypred1= svc.predict(xtest)
```

```
In [80]: 1 print(" Accuracy score :",accuracy_score(ytest,ypred1),"\\n","="*80,"\\n Cross_validation_Score :",
2         cross_val_score(svc,x,y,cv=3).mean(),"\\n","="*80,"\\n Classification report :",classification_report(ytest,ypred1),
3         "="*80,"\\n Confusion matrix :",confusion_matrix(ytest,ypred1))
```

Accuracy score : 0.9877961234745154

=====

Cross_validation_Score : 0.9816942806425479

=====

Classification report :

	precision	recall	f1-score	support
0	0.99	1.00	0.99	1236
1	0.98	0.91	0.94	157
accuracy			0.99	1393
macro avg	0.98	0.95	0.97	1393
weighted avg	0.99	0.99	0.99	1393

=====

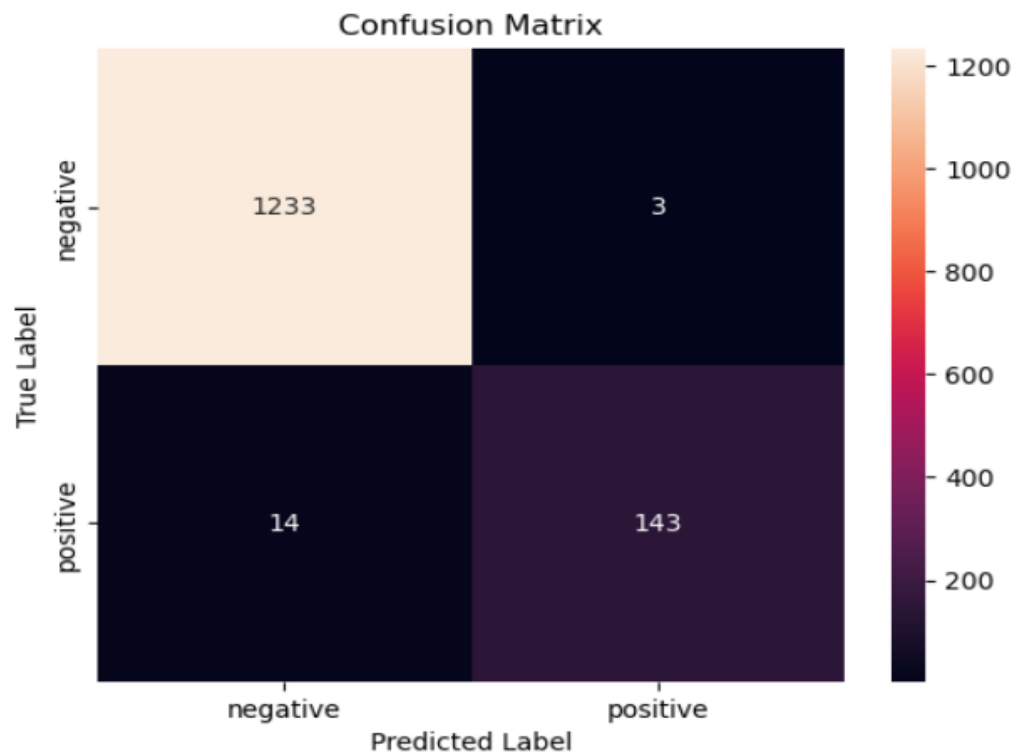
Confusion matrix :

```
[[1233  3]
 [ 14 143]]
```

```
In [81]: 1 log_loss(ytest,ypred1)
```

Out[81]: 0.42150868449804274

```
In [82]: 1 conf_mat = confusion_matrix(ytest, ypred1)
2         class_label = ["negative", "positive"]
3         email = pd.DataFrame(conf_mat, index = class_label, columns = class_label)
4         sns.heatmap(email, annot = True, fmt="d")
5         plt.title("Confusion Matrix")
6         plt.xlabel("Predicted Label")
7         plt.ylabel("True Label")
8         plt.show()
```



After Hyperparameter Tuning, we got an accuracy score of 98.77%.

Saving the model:

```
In [85]: 1 import joblib
2         3 joblib.dump(svc, 'Email Spam Classifier.pkl')
```

```
Out[85]: ['Email Spam Classifier.pkl']
```

```
In [86]: 1 loadmodel = joblib.load('Email Spam Classifier.pkl')
         2 loadmodel.predict(xtest)
```

```
Out[86]: array([0, 1, 0, ..., 0, 0, 0], dtype=int64)
```

```
In [87]: 1 conclusion = pd.DataFrame([loadmodel.predict(xtest)[:], ytest[:]], index=['Predicted', 'Original'])
         2 conclusion
```

```
Out[87]:
```

	0	1	2	3	4	5	6	7	8	9	...	1383	1384	1385	1386	1387	1388	1389	1390	1391	1392
Predicted	0	1	0	0	0	1	1	0	1	1	...	0	0	0	0	0	0	0	0	0	0
Original	0	1	0	0	0	1	1	0	1	1	...	0	0	0	0	0	0	0	0	0	0

2 rows × 1393 columns

3.6 Interpretation of the Results

Many machine learning algorithms are used to predict. However, the prediction accuracy of these algorithms depends heavily on the given data when training the model. If the data is in bad shape, the model will be overfitted and inefficient, which means that data pre-processing is an important part of this experiment and will affect the final results. Thus, multiple combinations of pre-processing methods need to be tested before getting the data ready to be used in training. After analyzing every model, Linear Support Vector Machine Classifier Model shows good accuracy and cross validation with least difference. On performing hyperparameter tuning, its accuracy reached 98.77%.

4. Conclusion

4.1 Key Findings and Conclusions of the Study

Machine learning algorithms can be extensively applied in the field of spam filtering. Substantial work have been done to improve the effectiveness of spam filters for classifying emails as either ham (valid messages) or spam (unwanted messages) by means of ML classifiers. They have the ability to recognize distinctive characteristics of the contents of emails.

4.2 Learning Outcomes of the Study in respect of Data Science

I found that the dataset was quite interesting to handle. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analyzed. New analytical techniques of machine learning can be used in property research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove unrealistic values and stopwords. This study is an exploratory attempt to use machine learning algorithms to identify whether an email is spam or ham.

4.3 Limitations of this work and Scope for Future Work

Summarily, the open research problems in email spam filtering are as follows:

- Lack of effective strategy to handle the threats to the security of the spam filters. Such an attack can be causative or exploratory, targeted or indiscriminate attack.
- The inability of the current spam filtering techniques to effectively deal with the concept drift phenomenon.

- Majority of the existing email spam filters does not possess the capacity to incrementally learn in real-time. Conventional spam email classification techniques are no longer viable to cope in real-time environment that is characterized by evolving data streams and concept drift.
- Failure of many spam filters to reduce their false positive rate.
- Development of more efficient image spam filters. Most spam filters can only classify spam messages that are text. However, many savvy spammers send spam email as text embedded in an image (stego image) thereby making the spam email to evade detection from filters.
- The need to develop adapted, scalable, and integrated filters by applying ontology and semantic web to spam email filtering.
- Lack of filters that have the capacity to dynamically update the feature space. Majority of the existing spam filters are unable to incrementally add or delete features without re-creating the model totally to keep abreast of current trends in email spam filtering.
- The need to apply deep learning to spam filtering in order to exploit its numerous processing layers and many levels of abstraction to learn representations of data.
- The inevitable need to design spam filters with lower processing and classification time using Graphics Processing Unit (GPU) and Field-Programmable Gate Array (FPGA) with their advantage of low power consumption, reconfigurability, and real-time processing capability for real-time processing and classification.

5. Reference

- <https://www.javatpoint.com/nlp>
- <https://www.educative.io/answers/preprocessing-steps-in-natural-language-processing-nlp>
- <https://www.youtube.com/watch?v=5ctbvkAMQO4>

- <https://www.youtube.com/watch?v=X2vAabgKiuM>