Building a Smarter AI-Powered Spam Classifier

The use of internet has been extensively increasing over the past decade and it continues to be on the ascent. Hence it is apt to say that the Internet is gradually becoming an integral part of everyday life. Internet usage is expected to continue growing and e-mail has become a powerful tool intended for idea and information exchange. Negligible time delay during transmission, security of the data being transferred, low costs are few of the multifarious advantages that e-mail enjoys over other physical methods. However there are few issues that spoil the efficient usage of emails. Spam email is one among them [1]. In recent years, spam email or more properly, Unsolicited Bulk Email (UBE) is a widespread problem on the Internet. Spam email is so cheap to send, that unsolicited messages are sent to a large number of users indiscriminately. When a large number of spam messages are received, it is necessary to take a long time to identify spam or non-spam email and their email messages may cause the mail server to crush.

To solve the spam problem, there have been several attempts to detect and filter the spam email on the client-side. In previous research, many Machine Learning (ML) approaches are applied to the problem, including Bayesian classifiers as Naive Bayes, C4.5 and Support Vector Machine (SVM) etc. In these approaches, Bayesian classifiers obtained good results by many researchers so that it widely applied to several filtering software's. However, almost approaches learn and find the distribution of the feature set in only the spam and the non-spam messages.

Smarter AI-Powered Spam Classifier is an advanced email filtering system designed to enhance your email experience by effectively identifying and managing spam messages. This innovative solution leverages cutting-edge artificial intelligence and machine learning technologies to accurately detect and categorize spam, minimizing the clutter in your inbox while ensuring that legitimate emails reach your primary folder. With its adaptive learning capabilities, the Smarter AI-Powered Spam Classifier continually refines its algorithms to stay ahead of evolving spam techniques, providing you with a more efficient and secure email communication environment. Say goodbye to the frustration of sifting through unwanted emails, thanks to this intelligent spam classification system.

What is spaCy

spaCy is an open-source natural language processing (NLP) library designed for processing and analyzing text data. It is written in Python and provides tools and resources for various NLP tasks, including tokenization, part-of-speech tagging, named

entity recognition, dependency parsing, and more. spaCy is known for its speed and efficiency, making it a popular choice for NLP tasks in research and industry applications. It also supports multiple languages and has pre-trained models for various languages, making it a versatile tool for working with text data.

Naive Bayes

Naive Bayes is a classification algorithm that's particularly well-suited for text classification tasks, like spam detection or sentiment analysis. It's based on Bayes' theorem, which is a fundamental concept in probability theory. Naive Bayes can be quite effective in text classification and spam detection. It calculates the probability of a data point belonging to a particular class based on the probabilities of its individual features. There are different variants of Naive Bayes, including Gaussian, Multinomial, and Bernoulli, which are suitable for different types of data.

Training: You start by training the model with a labeled dataset. For example, if you're building a spam filter, you'd provide it with a dataset of emails labeled as spam or not spam. The algorithm learns from this data.

Feature Independence: Naive Bayes makes the "naive" assumption that all features (words in the case of text) are independent of each other. This means that the presence or absence of one word doesn't affect the presence or absence of another word.

Calculating Probabilities: Given a new, unlabeled data point (e.g., a new email), Naive Bayes calculates the probability that it belongs to each class (e.g., spam or not spam) based on the occurrence of its features (words). It uses Bayes' theorem to do this.

Classification: The algorithm assigns the data point to the class with the highest probability.

Decision Tree C4.5

C4.5 (Classifier for 4.5) is a decision tree algorithm used in machine learning and data mining. It is designed for classification tasks and is named after its creator, Ross Quinlan. C4.5 constructs a decision tree from a dataset, where each internal node represents a test on an attribute, each branch represents an outcome of that test, and each leaf node represents a class label.

Attribute Selection: C4.5 selects the best attribute to split the dataset based on a metric called information gain. It measures how much information an attribute provides about the classification.

Recursive Partitioning: The dataset is divided into subsets based on the selected attribute. This process continues recursively until a stopping criterion is met, such as a maximum tree depth or a threshold for data purity.

Pruning: After the initial tree is constructed, C4.5 employs pruning to reduce the tree's complexity and improve its generalization by removing branches that don't contribute significantly to accuracy.

Handling Missing Values: C4.5 can handle missing attribute values during the tree construction process.

Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of interconnected neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron in one layer is connected to every neuron in the subsequent layer.
MLPs are used for a variety of machine learning tasks, including classification and regression. They can learn complex relationships within data by adjusting the weights of the connections between neurons during training. The hidden layers in MLPs enable them to model non-linear patterns in data.
MLPs have been a fundamental component of deep learning and are often used in more complex neural network architectures. They are known for their versatility and have been applied to a wide range of applications in fields like image recognition, natural language processing, and more.

RELATED WORK

Email spam is one of the major problems of the today’s Internet, bringing financial damage to companies and annoying individual users. Among the approaches developed to stop spam, filtering is the one of the most important technique. Spam mail, also called unsolicited bulk e-mail or junk mail that is sent to a group of recipients who have not requested it. The task of spam filtering is to rule out unsolicited e-mails automatically from a user's mail stream. These unsolicited mails have already caused

many problems such as filling mailboxes, engulfing important personal mail, wasting network bandwidth, consuming users time and energy to sort through it, not to mention all the other problems associated with spam [11]. Two methods of machine classification were described in paper [4]. The first one is done on some rules defined manually. The typical example is the rule based expert systems. This kind of classification can be used when all classes are static, and their components are easily separated according to some features. The second one is done using machine learning techniques. Paper [15] formalizes a problem of clustering of spam message collection through criterion function. The criterion function is a maximization of similarity between messages in clusters, which is defined by k-nearest neighbor algorithm. Genetic algorithm including penalty function for solving clustering problem is offered. Classification of new spam messages coming to the bases of antispam system. A novel distributed data mining approach, called Symbiotic Data Mining (SDM) [7] that unifies ContentBased Filtering (CBF) with Collaborative Filtering (CF) is described. The goal is to reuse local filters from distinct entities in order to improve personalized filtering while maintaining privacy. In paper [26] the effectiveness of email classifiers based on the feed forward back propagation neural network and Bayesian classifiers are evaluated. Results are evaluated using accuracy and sensitivity metrics. The results show that the feed forward back propagation network algorithm classifier provides relatively high
accuracy and sensitivity that makes it competitive to the best known classifiers. A fully Bayesian approach to soft clustering and classification using mixed membership models based on the assumptions on four levels: population, subject, latent variable, and sampling scheme was implemented in [8]. In paper [1]-[3], automatic anti-spam filtering becomes an important member of an emerging family of junk-filtering tools for the Internet, which will include tools to remove advertisements. The author separate distance measures for numeric and nominal variables, and are then combined into an overall distance measure. In another method, nominal variables are converted into numeric variables, and then a distance measure is calculated using all variables. Paper [24] analyzes the computational complexity and scalability of the algorithm, and tests its performance on a number of data sets from various application domains. The social networks of spammers [12] by identifying communities of harvesters with high behavioral similarity using spectral clustering. The data analyzed was collected through Project Honey Pot
[14], a distributed system for monitoring harvesting and spamming. The main findings are (1) that most spammers either send only phishing emails or no phishing emails at all, (2) that most communities of spammers also send only phishing emails or no phishing emails at all, and (3) that several groups of spammers within communities exhibit coherent temporal behavior or have similar IP addresses [5].

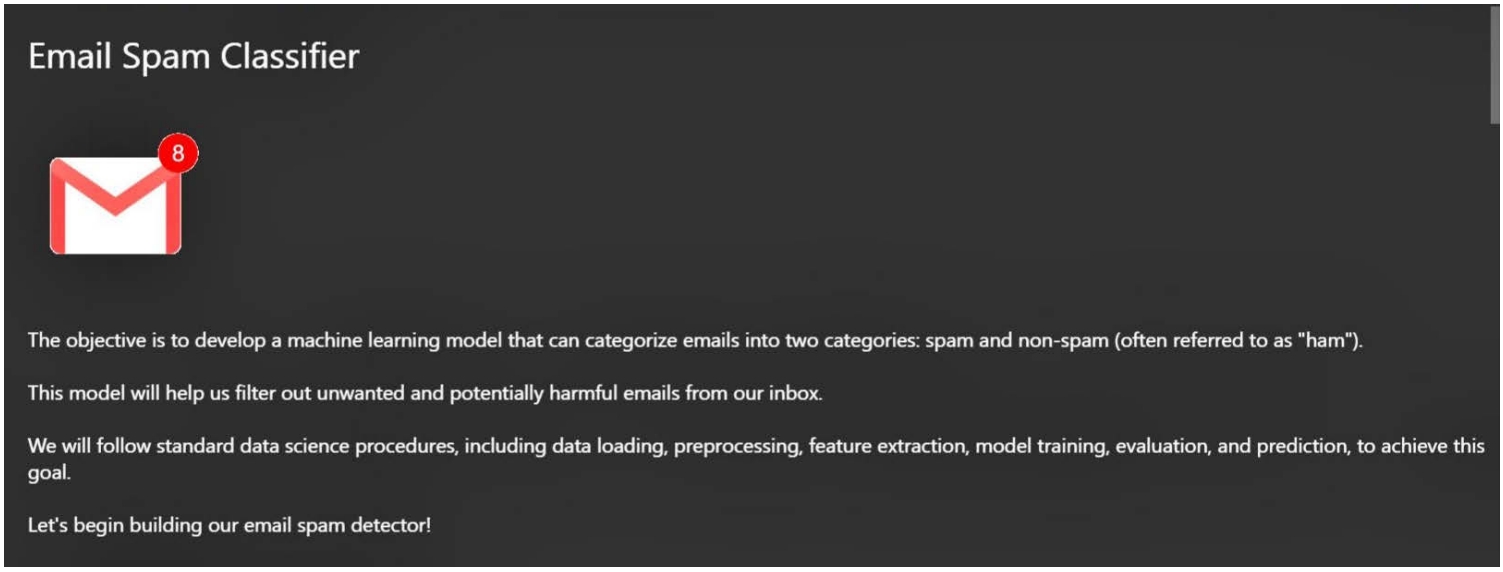
It is demonstrated that both methods obtain significant generalizations from a small number of examples; that both methods are comparable in generalization performance on problems of this type; and that both method asset reasonably efficient, even with fairly large training sets [6]. Spam classification [21] is done through Linear Discriminant Analysis by creating a bag-of words document for every Web site.

The Proposed Approach For Spam Email Classifi-cation

In this proposed model, the input is two publicly available datasets and one proposed email dataset is pre-processed using tokenization, stemming and lemmatization techniques. Afterward, the dataset is splited into train, and test by 80%, 20% respectively. Subsequently,
The spam detection is a text classification problem, That'swhy only standard ML models are used for feature engineering rather than Deep Learning (DL) techniques, dueto the viable execution proficiency interest. Furthermore, the features are extracted using two sate of the art modules namely Count-Vectorizer and TFIDF-Vectorizer and use different Hyper parameter optimization techniques for selecting the best set of parameters for a classification model.

Datasets

In this study, we use two publicly available text-based datasets. And also prepared the proposed dataset with the help of state-of-the-art, web scrapping, and personal email data. The proposed dataset is labeled with ham or spam emails. The datasets were in CSV format. The following are the two publicly available datasets, Ling Spam[14]. And UCI SMS SPAM dataset [15]. Both of these were not pre-processed and included links, numbers, and other noise in them.



Importing Necessary Libraries

```
# Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Load and Explore the Dataset

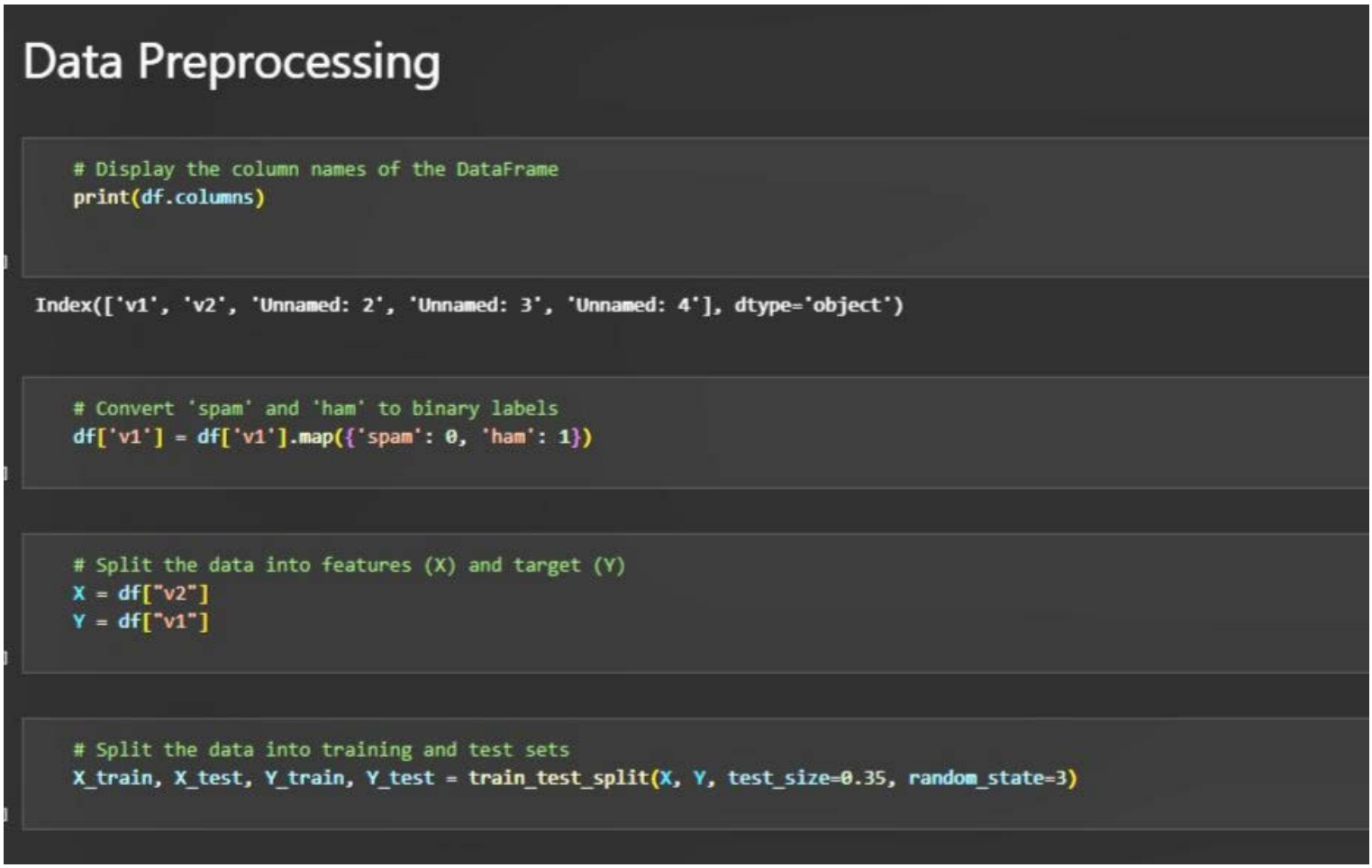
```
# Load the dataset
df = pd.read_csv("/kaggle/input/sms-spam-collection-dataset/spam.csv", encoding='ISO-8859-1')
```

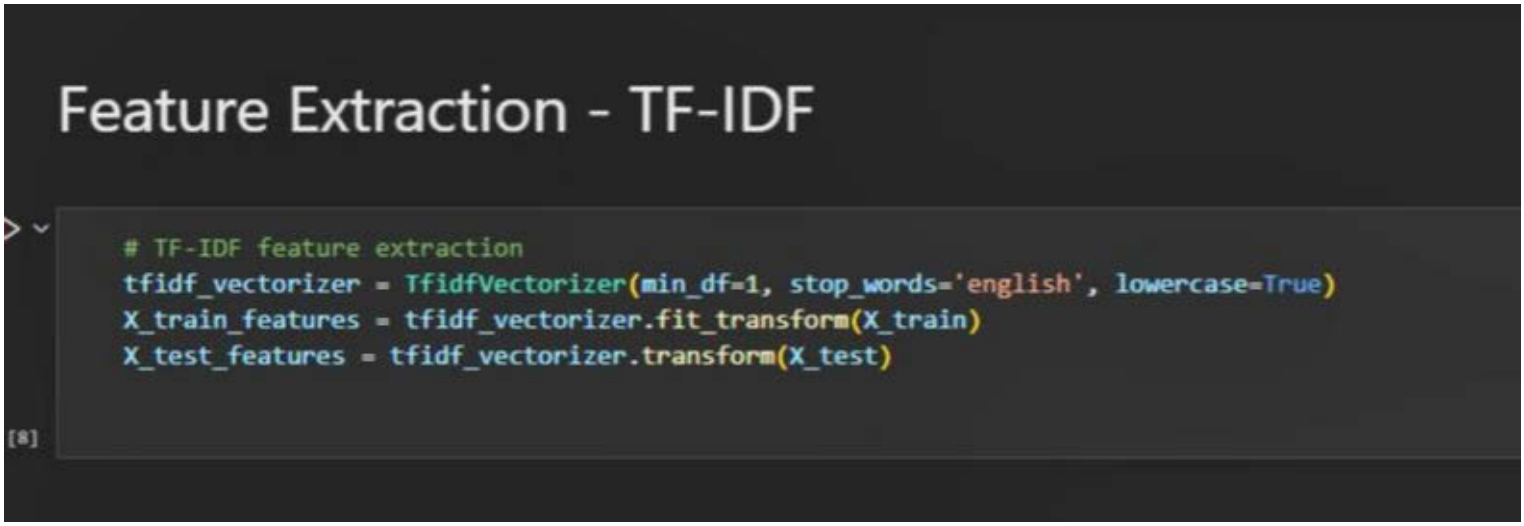
```
# Display the first few rows of the dataset
df.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Data Pre-processing

In filtering of spam, the pre-processing of the textual information is very critical and important. Main objective of text data pre processing is to remove data which do not give useful information regarding the class of the document. Furthermore we also want to remove data that is redundant. Most widely used data cleaning steps in the textual retrieval tasks are removing of stop words and performing stemming to reduce the vocabulary [2]. In addition to these two steps we also removed the words that have length lesser than or equal to two.





Model Training (Random Forest)

```
# Model training
model = RandomForestClassifier(n_estimators=100, random_state=3)
model.fit(X_train_features, Y_train)
```

[9]

```
RandomForestClassifier
RandomForestClassifier(random_state=3)
```

Model Evaluation (Random Forest)

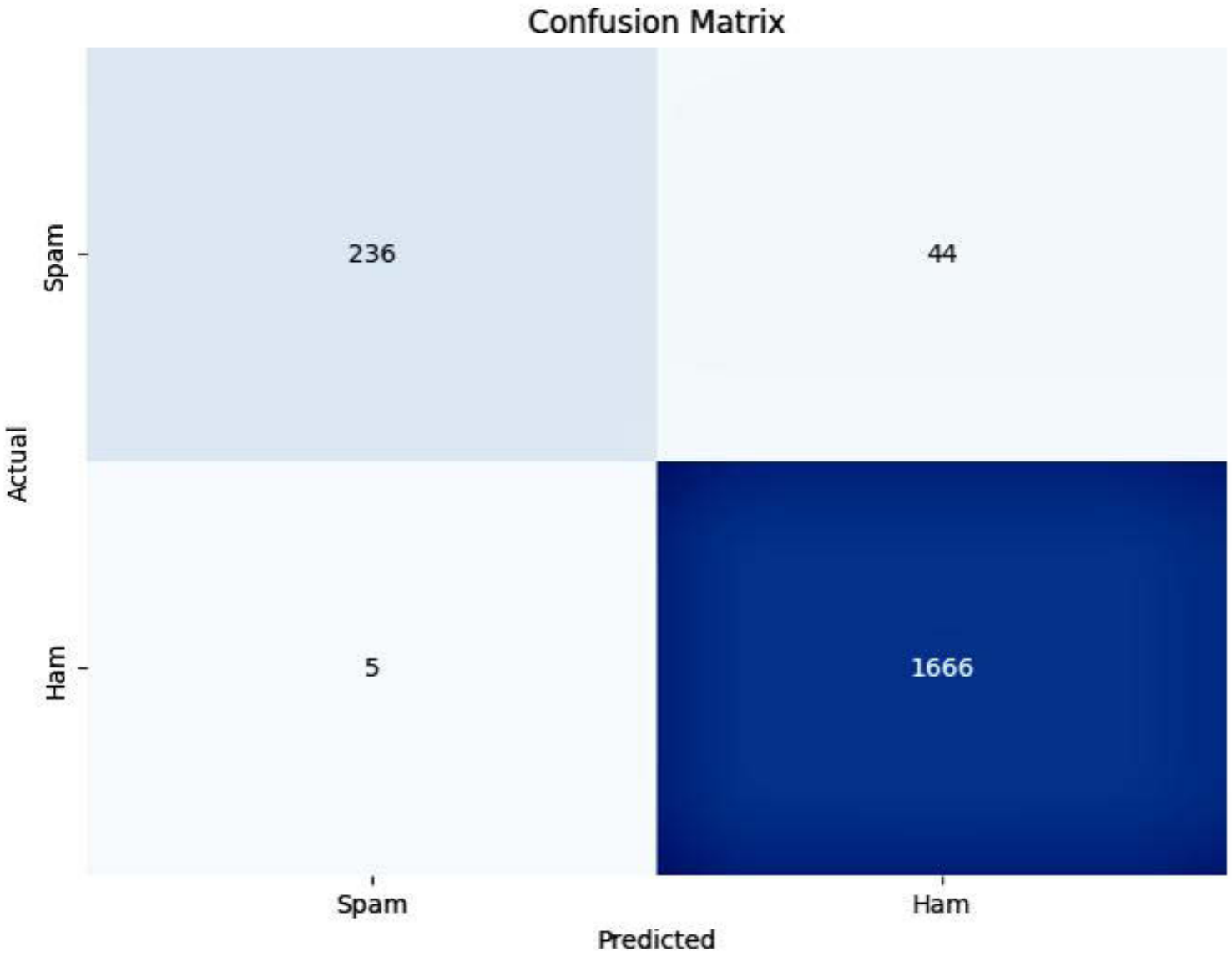
```
prediction_on_training_data = model.predict(X_train_features)
accuracy_on_training_data = accuracy_score(Y_train, prediction_on_training_data)

prediction_on_test_data = model.predict(X_test_features)
accuracy_on_test_data = accuracy_score(Y_test, prediction_on_test_data)
```

[10]

Confusion Matrix Visualization(Random Forest Classifier)

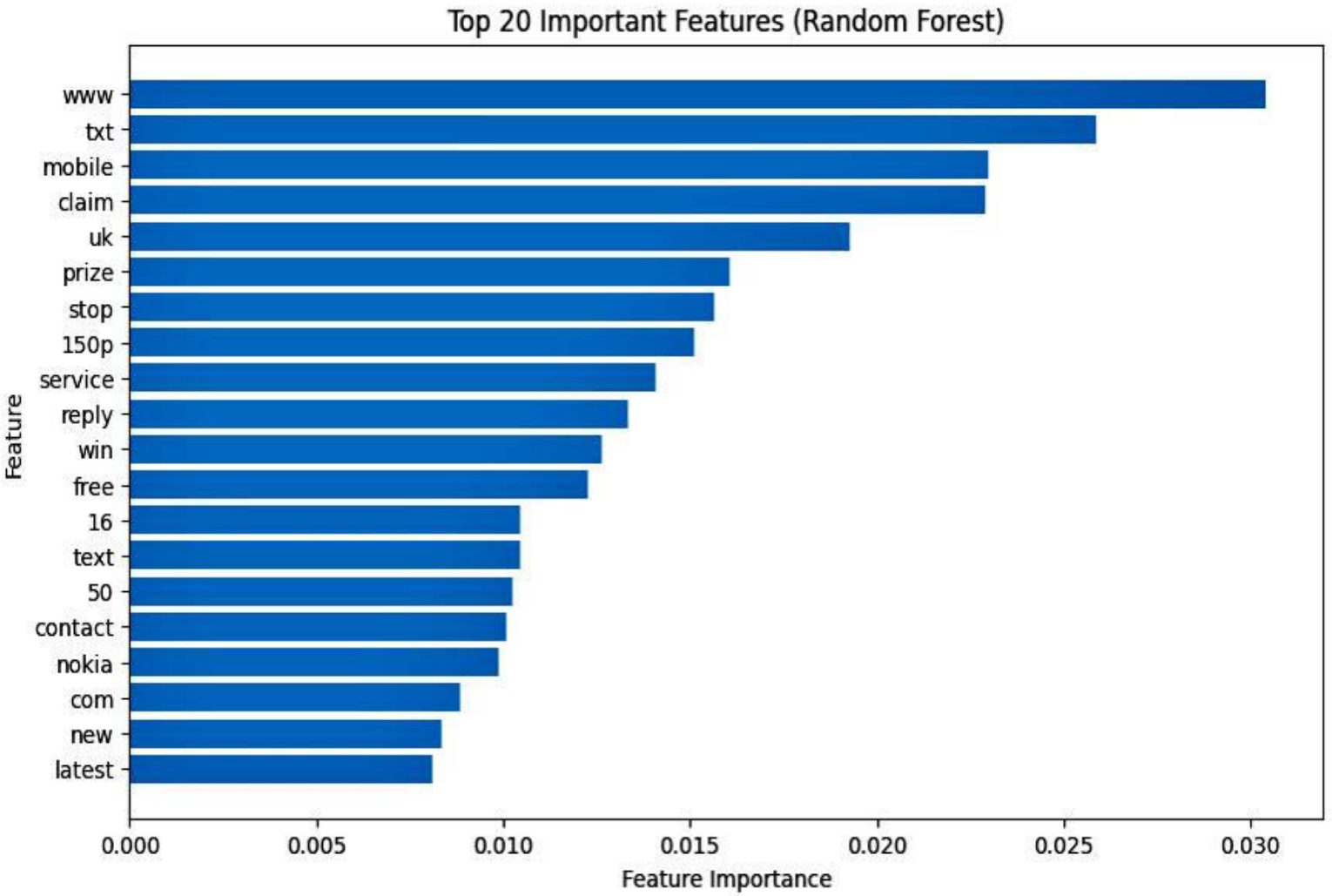
```
# Confusion Matrix Visualization
conf_matrix = confusion_matrix(Y_test, prediction_on_test_data)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False,
            xticklabels=['Spam', 'Ham'], yticklabels=['Spam', 'Ham'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



Feature Importance Visualization (Random Forest)

```
feature_importance = model.feature_importances_
feature_names = tfidf_vectorizer.get_feature_names_out()
sorted_idx = np.argsort(feature_importance)[-20:]  # Top 20 important features

plt.figure(figsize=(10, 6))
plt.barh(range(len(sorted_idx)), feature_importance[sorted_idx], align="center")
plt.yticks(range(len(sorted_idx)), [feature_names[i] for i in sorted_idx])
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Top 20 Important Features (Random Forest)")
plt.show()
```



Make Predictions on New Input (Random Forest Classifier)

```
input_your_mail = "Keep yourself safe for me because I need you and I miss you already and I envy everyone that see's you in real life"
input_data_features = tfidf_vectorizer.transform([input_your_mail])
prediction = model.predict(input_data_features)
if prediction[0] == 1:
    print("Ham Mail")
else:
    print("Spam Mail")
```

```
Ham Mail
```

Removal of Stopwords in Text Pre-processing

Stopwords are a set of words that do not value a text example ‘a’,’an’,’the’ these are the words that occur very frequently in our text data, but they are of no use. Many libraries have compiled stop words for various languages and we can use them directly and for any specific use case if we feel we can also add a more specific set of stop words to the list.

```
from nltk.corpus import
stopwords
",
".join(stopwords.words('english'
```

Code for removal of stop words Before stop words removal we need tokenized text

```
#defining function for
tokenization
import re
#whitespace tokenizer
from nltk.tokenize import
WhitespaceTokenizer
def tokenization(text):
    tk = WhitespaceTokenizer()
    return tk.tokenize(text)

#applying function to the
column for making tokens in
both Training and Testing data
df['tokenised_clean_msg']=
df['clean_msg'].apply(lambda
x: tokenization(x))
```

Now stop words removal

```
#importing nlp library
import nltk
nltk.download('stopwords')
#Stop words present in the
library
stopwords =
nltk.corpus.stopwords.words('eng

#defining the function to
remove stopwords from
tokenized text
def remove_stopwords(text):
    output= [i for i in text
if i not in stopwords]
    return output

#applying the function for
removal of stopwords
df['cleaned_tokens']=
df['tokenised_clean_msg'].apply(
x:remove_stopwords(x))
```

Stemming in Text Pre-processing

It is a text standardization technique where a word is reduced to its stem/base word. Example: “jabbing” → “jab” and “kicking” → “kick”. The main aim for stemming is that we can reduce the vocab size before inputting it into any machine learning model.

```
#importing the Stemming
function from nltk library
from nltk.stem.porter import
PorterStemmer
#defining the object for
stemming
porter_stemmer =
PorterStemmer()

#defining a function for
stemming
def stemming(text):
  stem_text =
[porter_stemmer.stem(word) for
word in text]
  return stem_text

# applying function for
stemming
df['cleaned_tokens']=df['cleaned
x: stemming(x))
```

Spam detection

Spam detection is the process of identifying and filtering out unsolicited, irrelevant, or potentially harmful messages or content, often found in various communication channels such as email, messaging apps, or online forums. It involves the use of algorithms, techniques, and heuristics to automatically recognize and prevent spam, ensuring that users are protected from unwanted or potentially malicious content.Once the various pre-processing operations (second step) have been performed on the input data and the quality datahas been obtained, different machine learning algorithms can be used to train and evaluate the results.

		Predicted classes				
		TRUE	FALSE		TP	135
					TN	177
Actual classes	TRUE	135	53	188	FP	10
	FALSE	10	177	187	FN	53
		145	230			

Fig. 4. Confusion matrix extraction from the final Naive Bayes (NB) model

		Predicted classes				
		TRUE	FALSE		TP	148
					TN	185
Actual classes	TRUE	148	32	180	FP	10
	FALSE	10	185	195	FN	32
		158	217			

Fig. 5. Confusion matrix extraction from the final Decision Tree C45 model

CONCLUSION

Email spam classification has received a tremendous attention by majority of the people as it helps to identify the unwanted information and threats. Therefore, most of the researchers pay attention in finding the best classifier for detecting spam emails. From the obtained results, fisher filtering and runs filtering feature selection algorithms performs better classification for many classifiers. The Rnd tree classification algorithm applied on relevant features after fisher filtering has produced more than 99% accuracy in spam detection. This Rnd tree classifier is also tested with test dataset which gives accurate results than other classifiers for this spam dataset.