Sardar Patel Institute of Technology, Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII (2022-2023)
OEIT6 - Data Analytics

Experiment: Classification

Name: Pushkar Sutar **Roll No. 2019110060**

Objective : Separating ham from spam in Enron email dataset.

Dataset Description:
The data set to be used is the one corresponding to the database "Enron Email dataset". Contains data of around 150 users with a total of around 500,000 email messages. Originally the dataset was made public by the Federal Commission US Energy Regulatory Agency during the investigation surrounding the collapse of the Enron company.
https://www.kaggle.com/datasets/juanagsolano/spam-email-from-enron-dataset
Attribute information -
email - contains an email body.
Spam - target, 0 for ham and 1 for spam.

Code and Output:

Problem 1.1 : Loading the dataset -

We first import all the necessary modules

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
```

We read the csv files and store it in variable emails.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
emails = pd.read_csv("/content/drive/MyDrive/DA/emails.csv")
```

We can take a peek at the dataset imported.

```
emails.head()
```

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1

```
emails.describe()
```

	spam
count	5728.000000
mean	0.238827
std	0.426404
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

```
emails.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5728 entries, 0 to 5727
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    5728 non-null   object
 1   spam    5728 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 89.6+ KB
```

Q. How many emails are there in the dataset?

```
emails.shape
```

```
(5728, 2)
```

From the .shape method we can see that there are 5728 emails in the dataset.

Q. How many of the emails are spam?

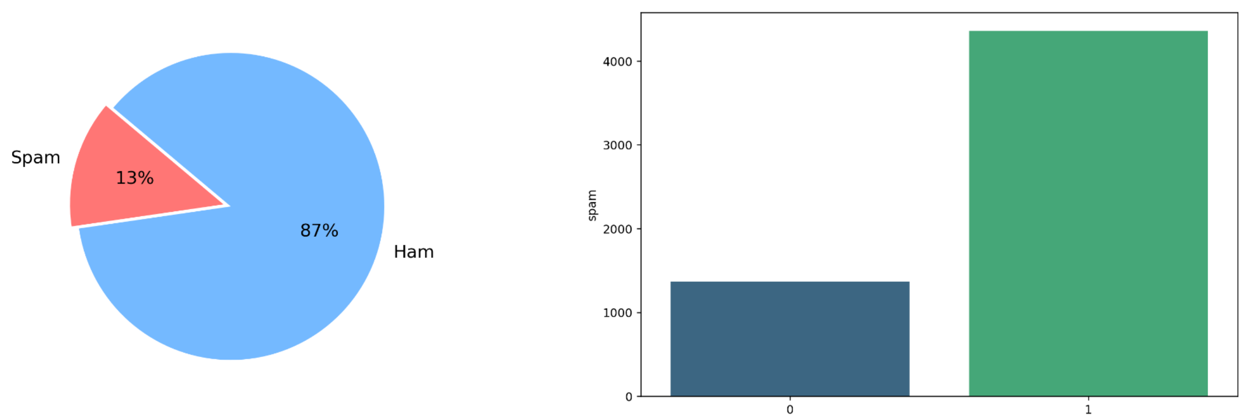
```
emails['spam'].value_counts()
```

```
0    4360
1    1368
Name: spam, dtype: int64
```

There are a total of 1368 spam emails in the dataset.

Pie chart and bar graph to visualise the spam and ham emails distribution.

```
labels = ['Spam', 'Ham']
sizes = [747, 4825]
custom_colours = ['#ff7675', '#74b9ff']

plt.figure(figsize=(20, 6), dpi=227)
plt.subplot(1, 2, 1)
plt.pie(sizes, labels = labels, textprops={'fontsize': 15}, startangle=140,
       autopct='%1.0f%%', colors=custom_colours, explode=[0, 0.05])

plt.subplot(1, 2, 2)
sns.barplot(x = data_set['spam'].unique(), y = data_set['spam'].value_counts(), palette= 'viridis')

plt.show()
```



The pie chart shows that there are 13% spam emails.

We will now look at total word and character count per email.

```python
emails['Total Words'] = emails['text'].apply(lambda x: len(x.split()))

def count_total_words(text):
    char = 0
    for word in text.split():
        char += len(word)
    return char

def count_chars(text):

    return len(text)


emails['Total Chars']= emails['text'].apply(count_chars)
```

```python
emails.head()
```

	text	spam	Total Words	Total Chars
0	Subject: naturally irresistible your corporate...	1	324	1484
1	Subject: the stock trading gunslinger fanny i...	1	89	598
2	Subject: unbelievable new homes made easy im ...	1	87	448
3	Subject: 4 color printing special request add...	1	98	500
4	Subject: do not have money , get software cds ...	1	52	235

```python
maxval = emails['Total Chars'].max()



print("Maximum Characters: ")
print(maxval)
```

```
Maximum Characters:
43952
```

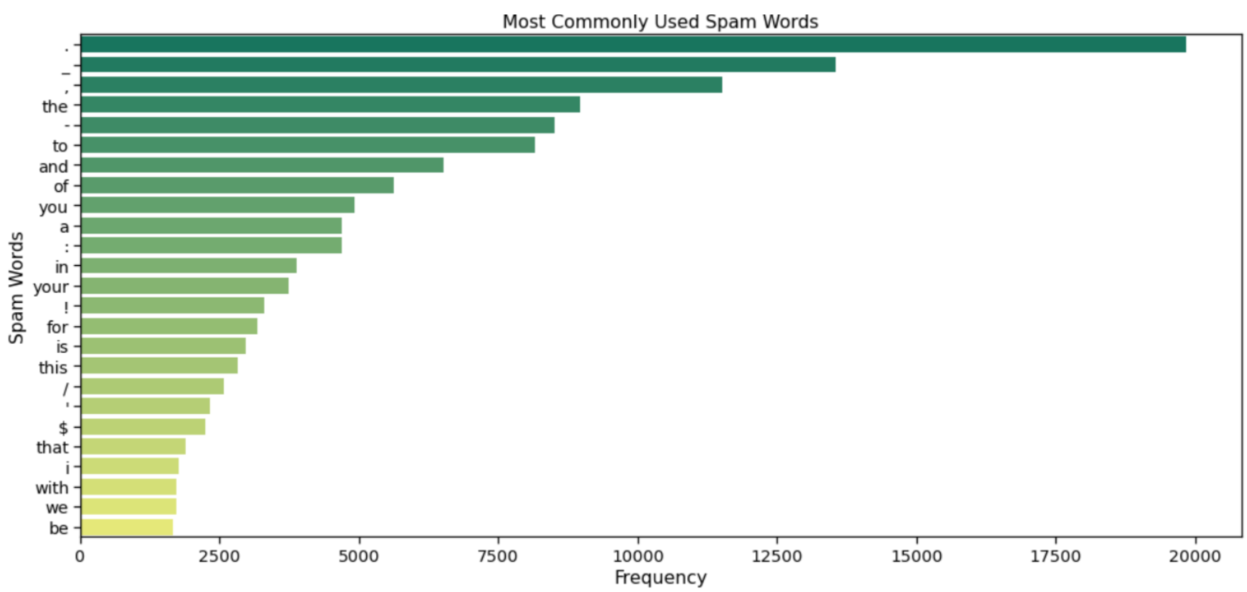
Q.The nchar() function counts the number of characters in a piece of text. How many characters are in the longest email in the dataset (where longest is measured in terms of the maximum number of characters)?
The longest email contains a total of 43952 characters.

Q. Which word appears at the beginning of every email in the dataset? Respond as a lower-case word with punctuation removed.
The word 'subject' appears at the beginning of every email in the dataset.

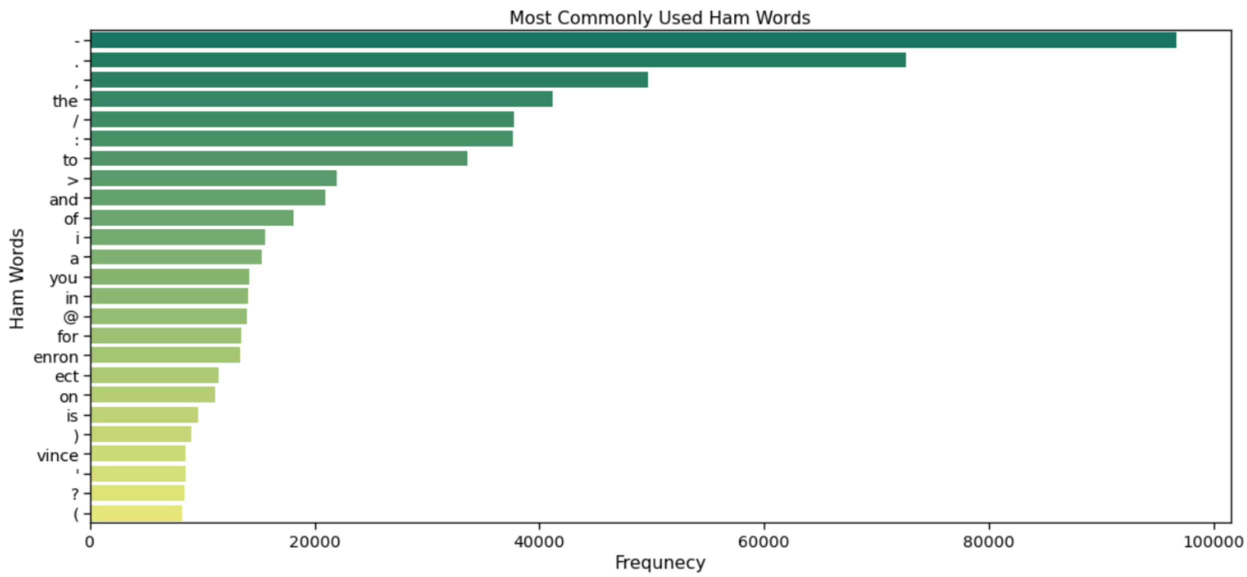
We now check the frequency of words by category to verify if they have any significance in determining whether the email is spam or not.

```
all_spam_words = []
for sentence in emails[emails['spam'] == 1]['text'].to_list():
    for word in sentence.split():
        all_spam_words.append(word)

df = pd.DataFrame(Counter(all_spam_words).most_common(25), columns= ['Word', 'Frequency'])

sns.set_context('notebook', font_scale= 1.3)
plt.figure(figsize=(18,8))
sns.barplot(y = df['Word'], x= df['Frequency'], palette= 'summer')
plt.title("Most Commonly Used Spam Words")
plt.xlabel("Frequency")
plt.ylabel("Spam Words")
plt.show()
```

```
all_ham_words = []
for sentence in emails[emails['spam'] == 0]['text'].to_list():
    for word in sentence.split():
        all_ham_words.append(word)

df = pd.DataFrame(Counter(all_ham_words).most_common(25), columns= ['Word', 'Frequency'])

sns.set_context('notebook', font_scale= 1.3)
plt.figure(figsize=(18,8))
sns.barplot(y = df['Word'], x= df['Frequency'], palette= 'summer')
plt.title("Most Commonly Used Ham Words")
plt.xlabel("Frequnecy")
plt.ylabel("Ham Words")
plt.show()
```



We can observe symbols like ! and $ are usually a part of spam mails.



Q.Could a spam classifier potentially benefit from including the frequency of the word that appears in every email?

Yes -- the number of times the word appears might help us differentiate spam from ham.

No -- the word appears in every email so this variable would not help us differentiate spam from ham.

If we check the frequency of words we could get potential sets of words that are usually a part of spam mails. With this we can better understand and differentiate spam from ham.

Problem 2.1 - Preparing the corpus

```
import nltk

nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

Follow the standard steps to build and pre-process the corpus:

1) Build a new corpus variable called corpus.
2) Using tm_map, convert the text to lowercase.
3) Using tm_map, remove all punctuation from the corpus.
4) Using tm_map, remove all English stopwords from the corpus.
5) Using tm_map, stem the words in the corpus.
6) Build a document term matrix from the corpus, called dtm.

```
emails['text'] = emails['text'].apply(lambda x: ' '.join(x.split(' ')[1:]))
```

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
#Every mail starts with 'Subject :' will remove this from each text

emails['text']=emails['text'].map(lambda text: text[1:])
emails['text'] = emails['text'].map(lambda text:re.sub('[^a-zA-Z0-9]+', ' ',text)).apply(lambda x: (x.lower()).split())
ps = PorterStemmer()
corpus=emails['text'].apply(lambda text_list:' '.join(list(map(lambda word:ps.stem(word),(list(filter(lambda text:text not in set(stopwords.words('english')),text_list)))))))

# Creating the Bag of Words model
```

```
emails['corpus']=corpus
```

We have removed all the punctuations, converted the text to lowercase, removed all the stopwords and have prepared the corpus.

```
from nltk.corpus import stopwords

# Make a list of english stopwords
stopwords = nltk.corpus.stopwords.words("english")

print(len(stopwords))
```

```
179
```

The length of the stopwords from nltk is 179.

Q.How many terms are in dtm?

```
count=0
for i in range(5728):
    count=count+len(corpus[i].split())
print(count)
```

```
878314
```

The document term matrix consists of 878314 terms.

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
X = cv.fit_transform(corpus.values).toarray()
y = data_set.iloc[:, 1].values
```

Problem 3.1 - Building the model

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
X_train, X_test, y_train, y_test = train_test_split(
         X, y, test_size = 0.3, random_state = 100)
```

Performing the test train split on the dataset.

CART Algorithm -

```
spamCART = DecisionTreeClassifier()

# Train Decision Tree Classifer
spamCART = spamCART.fit(X_train,y_train)
spamtrain=spamCART.predict(X_train)
#Predict the response for test dataset
y_pred =  spamCART.predict(X_test)
```

```
#CART train metrics
from sklearn.metrics import accuracy_score, precision_score, recall_score,f1_score

print('Accuracy score: ', accuracy_score(y_train, spamtrain))
print('Precision score: ', precision_score(y_train, spamtrain))
print('Recall score: ', recall_score(y_train, spamtrain))
print("F1 score: ",f1_score(y_train,spamtrain))
```

```
Accuracy score:  1.0
Precision score:  1.0
Recall score:  1.0
F1 score:  1.0
```

Q. What is the training set accuracy of spamCART, using a threshold of 0.5 for predictions?
The training set accuracy of spamCART is 100%.

Random Forest Algorithm-

```
from sklearn.ensemble import RandomForestClassifier

 # create regressor object
spamRF = RandomForestClassifier(n_estimators = 100, random_state = 0)

# fit the regressor with x and y data
spamRF=spamRF.fit(X_train,y_train)
spamRFtrain=spamRF.predict(X_train)
```

```
y_pred1 = spamRF.predict(X_test)
```

```
#train accuracy for Spam RF

from sklearn.metrics import accuracy_score, precision_score, recall_score,f1_score

print('Accuracy score: ', accuracy_score(y_train, spamRFtrain))
print('Precision score: ', precision_score(y_train, spamRFtrain))
print('Recall score: ', recall_score(y_train, spamRFtrain))
print("F1 score: ",f1_score(y_train,spamRFtrain))
```

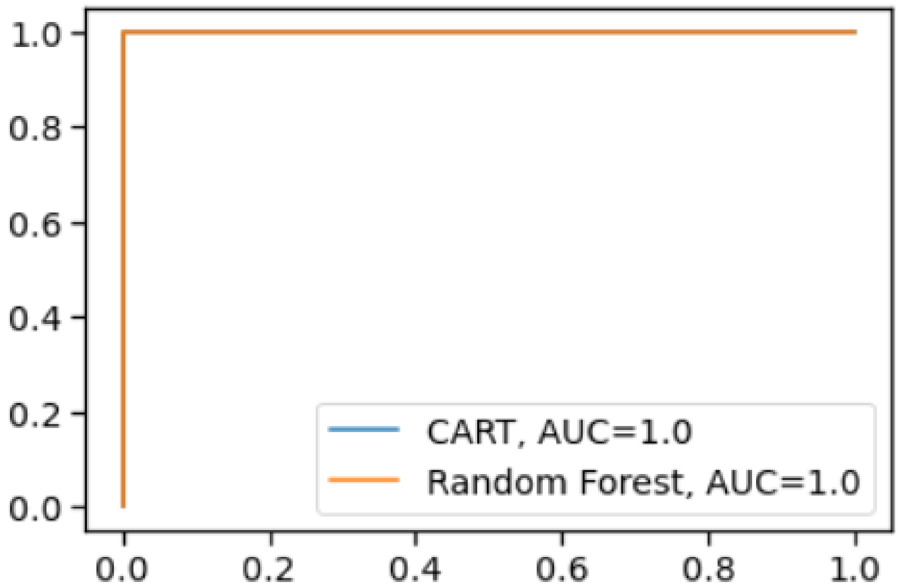
```
Accuracy score:  1.0
Precision score:  1.0
Recall score:  1.0
F1 score:  1.0
```

Q. What is the training set accuracy of spamRF, using a threshold of 0.5 for predictions? (Remember that your answer might not match ours exactly, due to random behaviour in the random forest algorithm on different operating systems.)
The training set accuracy of spamRF is also 100%.

Checking training set AUC.

```
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
fpr, tpr, _ = metrics.roc_curve(y_train,  spamtrain)
auc = round(metrics.roc_auc_score(y_train,  spamtrain), 4)
plt.plot(fpr,tpr,label="CART, AUC="+str(auc))

fpr, tpr, _ = metrics.roc_curve(y_train, spamRFtrain)
auc = round(metrics.roc_auc_score(y_train, spamRFtrain), 4)
plt.plot(fpr,tpr,label="Random Forest, AUC="+str(auc))

#add legend
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7faec6517590>
```



Q. What is the training set AUC of spamCART?
What is the training set AUC of spamRF?
The training set AUC for both the models is 1 indicating that the model's predictions on the training set are 100% always correct.

Problem 4.1 Evaluating on test data

```
#CART train accuracy

from sklearn.metrics import accuracy_score, precision_score, recall_score,f1_score

print('Accuracy score: ', accuracy_score(y_test, y_pred))
print('Precision score: ', precision_score(y_test, y_pred))
print('Recall score: ', recall_score(y_test, y_pred))
print("F1 score: ",f1_score(y_test, y_pred))
```

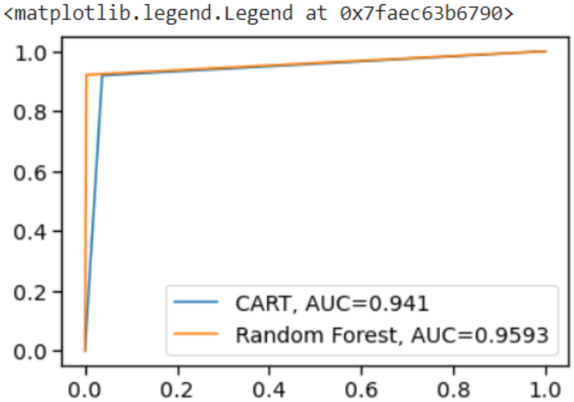
```
Accuracy score:  0.951716114019779
Precision score:  0.8926174496644296
Recall score:  0.9193548387096774
F1 score:  0.905788876276958
```

```python
#Random Forest train accuracy
from sklearn.metrics import accuracy_score, precision_score, recall_score,f1_score

print('Accuracy score: ', accuracy_score(y_test, y_pred1))
print('Precision score: ', precision_score(y_test, y_pred1))
print('Recall score: ', recall_score(y_test, y_pred1))
print("F1 score: ",f1_score(y_test, y_pred1))
```

```
Accuracy score:  0.9778941244909831
Precision score:  0.9900990099009901
Recall score:  0.9216589861751152
F1 score:  0.954653937947494
```

```python
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred)
auc = round(metrics.roc_auc_score(y_test,  y_pred), 4)
plt.plot(fpr,tpr,label="CART, AUC="+str(auc))

fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred1)
auc = round(metrics.roc_auc_score(y_test,  y_pred1), 4)
plt.plot(fpr,tpr,label="Random Forest, AUC="+str(auc))

#add legend
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7faec63b6790>
```



Q. What is the testing set accuracy of spamCART, using a threshold of 0.5 for predictions?
The testing set accuracy of spamCART is 95.17%.

Q. What is the testing set AUC of spamCART?
The testing set AUC of spamCART is 0.94 which is pretty good.

Q. What is the testing set accuracy of spamRF, using a threshold of 0.5 for predictions?
The testing set accuracy of spamRF is 97.7%.

Q. What is the testing set AUC of spamRF?

The testing set AUC of spamRF is 0.95.

Q. Which model had the best testing set performance, in terms of accuracy and AUC?
-CART or Random Forest
In terms of accuracy, the random forest algorithm performed better.

Conclusions :

- AUC measures the degree of separability between the two classes (spam and ham). Random Forest algorithm provides a better separability.
- In terms of accuracy also, the random forest algorithm performs better.
- If we evaluate the model based on f1 score, it is clear that random forest beats the CART algorithm.
- Preprocessing the data by removing stopwords, punctuations, converting every word in lowercase, and other techniques also helped to significantly improve the accuracy.