In [1]:
```
# Importing the necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
# Load dataset

df = pd.read_csv('spam.tsv', sep='\t')
```

In [3]:
```
df.head()
```

Out[3]:

	label	message	length	punct
0	ham	Go until jurong point, crazy.. Available only ...	111	9
1	ham	Ok lar... Joking wif u oni...	29	6
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	6
3	ham	U dun say so early hor... U c already then say...	49	6
4	ham	Nah i don't think he goes to usf, he lives aro...	61	2

In [4]:
```
# This is to count the total number of records

df.count()
```

Out[4]:
```
label      5572
message    5572
length     5572
punct      5572
dtype: int64
```

In [5]:
```
# This shows that we have no missing data

df.isna().sum()
```

Out[5]:
```
label      0
message    0
length     0
punct      0
dtype: int64
```

In [6]:
```
df.describe()
```

Out[6]:

	length	punct
count	5572.000000	5572.000000
mean	80.489950	4.177495
std	59.942907	4.623919
min	2.000000	0.000000
25%	36.000000	2.000000
50%	62.000000	3.000000
75%	122.000000	6.000000
max	910.000000	133.000000

DATA PREPROCESSING AND ANALYSIS.

In [7]:
```
# This shows the value count of ham and sham messages. The spam and ham data is imbalanced.

df['label'].value_counts()
```

Out[7]:
```
ham     4825
spam     747
Name: label, dtype: int64
```

In [8]:
```
ham = df[df['label'] == 'ham']
spam = df[df['label'] == 'spam']
```

In [9]:
```
ham.shape, spam.shape
```

Out[9]:
```
((4825, 4), (747, 4))
```

In [10]:
```
ham = ham.sample(spam.shape[0])
```

In [11]:
```
# This shows that the spam and ham messages are balanced.

ham.shape, spam.shape
```

Out[11]:
```
((747, 4), (747, 4))
```

In [12]:
```
# I combine both ham and spam into 'data'

data = ham.append(spam, ignore_index=True)
```

In [13]:
```
data.shape
```

Out[13]:
```
(1494, 4)
```

In [14]:
```
# This shows a visualisation of the length of messages between ham and spam.

plt.hist(data[data['label'] == 'ham']['length'], bins=100, alpha=0.7, label='Ham', color='green')
plt.hist(data[data['label'] == 'spam']['length'], bins=100, alpha=0.7, label='Spam', color='orange')
plt.legend(loc='upper right')
plt.show()
```

In [15]:
```
# This shows a visualisation of the Punctuation marks in messages between ham and spam.

plt.hist(data[data['label'] == 'ham']['punct'], bins=100, alpha=0.7, label='Ham', color='green')
plt.hist(data[data['label'] == 'spam']['punct'], bins=100, alpha=0.7, label='Spam', color='orange')
plt.legend(loc='upper right')
plt.show()
```

In [ ]:
```

```

SPLIT DATA INTO TESTING AND TRAINING SETS

In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
# We segregate our data into testing/training sets. 30 percent ('0.3') of data will be on the testing sets
# and the rest in the training sets.

X_train, X_test, y_train, y_test = train_test_split(data['message'], data['label'], test_size = 0.3, random_st
```

In [18]:
```
X_train.shape
```

Out[18]:
```
(1045,)
```

In [19]:
```
X_test.shape
```

Out[19]:
```
(449,)
```

In [ ]:
```

```

BUILDING THE RANDOM FOREST MODEL.

In [20]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
```

In [21]:
```
# Creating a pipeline object which we will pass 'tfidf vectorizer' and "random forest classifier"

classifier = Pipeline([("tfidf", TfidfVectorizer()) , ("classifier", RandomForestClassifier(n_estimators=100))]
```

In [22]:
```
# train the model

classifier.fit(X_train, y_train)
```

Out[22]:
```
Pipeline(steps=[('tfidf', TfidfVectorizer()),
                ('classifier', RandomForestClassifier())])
```

In [ ]:
```

```

EVALUATE THE MODEL (RANDOM FOREST)

In [23]:
```
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
import seaborn as sns
```

In [24]:
```
y_pred1 = classifier.predict(X_test)
```

In [25]:
```
print(classification_report(y_test, y_pred1))
```

```
              precision    recall  f1-score   support

         ham       0.90      1.00      0.95       227
        spam       1.00      0.88      0.94       222

    accuracy                           0.94       449
   macro avg       0.95      0.94      0.94       449
weighted avg       0.95      0.94      0.94       449
```

In [26]:
```
# Compute the confusion matrix
conf_mat1 = confusion_matrix(y_test, y_pred1)

# Display the confusion matrix
print("Confusion Matrix:")
print(conf_mat1)
```

```
Confusion Matrix:
[[227   0]
 [ 26 196]]
```

In [27]:
```
# Plot the confusion matrix as a heatmap
plt.figure(figsize=(6, 6))
sns.heatmap(conf_mat1, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=['Ham', 'Spam'], yticklabels=

# Add labels and title
plt.xlabel('Predicted Labels', fontsize=14)
plt.ylabel('True Labels', fontsize=14)
plt.title('Confusion Matrix', fontsize=16)

# Display the plot
plt.show()
```

In [ ]:
```

```

BUILDING THE SVM MODEL.

In [28]:
```
from sklearn.svm import SVC
```

In [29]:
```
# Creating a pipeline object which we will pass 'tfidf vectorizer' and "random forest classifier"

svm = Pipeline([("tfidf", TfidfVectorizer()) , ("classifier", SVC(C = 100, gamma='auto'))]
```

In [30]:
```
# train the model

svm.fit(X_train, y_train)
```

Out[30]:
```
Pipeline(steps=[('tfidf', TfidfVectorizer()),
                ('classifier', SVC(C=100, gamma='auto'))])
```

In [ ]:
```

```

EVALUATE THE MODEL (SUPPORT VECTOR MACHINE)

In [31]:
```
y_pred2 = svm.predict(X_test)
```

In [32]:
```
print(classification_report(y_test, y_pred2))
```

```
              precision    recall  f1-score   support

         ham       0.92      0.96      0.94       227
        spam       0.96      0.91      0.94       222

    accuracy                           0.94       449
   macro avg       0.94      0.94      0.94       449
weighted avg       0.94      0.94      0.94       449
```

In [33]:
```
# Compute the confusion matrix
conf_mat2 = confusion_matrix(y_test, y_pred2)

# Display the confusion matrix
print("Confusion Matrix:")
print(conf_mat2)
```

```
Confusion Matrix:
[[219   8]
 [ 20 202]]
```

In [34]:
```
# Plot the confusion matrix as a heatmap
plt.figure(figsize=(6, 6))
sns.heatmap(conf_mat2, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=['Ham', 'Spam'], yticklabels=

# Add labels and title
plt.xlabel('Predicted Labels', fontsize=14)
plt.ylabel('True Labels', fontsize=14)
plt.title('Confusion Matrix', fontsize=16)

# Display the plot
plt.show()
```


In [ ]:
```

```

TESTING BOTH MODELS ON DATASET

In [35]:
```
# This are the test data which we will try to predict on each model.

test1 = ['Hello, You are learning natural language Processing']
test2 = ['Your free ringtone is waiting to be collected. Simply text the password "MIX" to 85069 to verify. Ge
test3 = ['Hope you are doing good and learning new things !']
test4 = ['Congratulations, You won a lottery ticket worth $1 Million ! To claim call on 446677']
```

In [36]:
```
# This shows the prediction of messages for Random Forest

print(classifier.predict(test1))
print(classifier.predict(test2))
print(classifier.predict(test3))
print(classifier.predict(test4))
```

```
['ham']
['spam']
['ham']
['spam']
```

In [37]:
```
# This also shows the prediction of messages for SVM

print(svm.predict(test1))
print(svm.predict(test2))
print(svm.predict(test3))
```

```
['ham']
['spam']
['ham']
```

In [ ]:
```

```