```
!pip install nltk
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nltk in /usr/local/lib/python3.8/dist-packages (3.7)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.8/dist-packages (from nltk) (2022.6.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from nltk) (4.64.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.8/dist-packages (from nltk) (1.2.0)
Requirement already satisfied: click in /usr/local/lib/python3.8/dist-packages (from nltk) (7.1.2)
```

```
!pip install -q wordcloud
import wordcloud
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

```
#Importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.svm import SVC
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

# Importing the warnings
import warnings
warnings.filterwarnings('ignore')
```

```
#Loading the dataset
df = pd.read_csv("messages.csv",encoding='latin-1')
```

```
df.head()
```

	subject	message	label
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0
1	NaN	lang classification grimes , joseph e . and ba...	0
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0
3	risk	a colleague and i are researching the differin...	0
4	request book information	earlier this morning i was on the phone with a...	0

```
#Checking information of dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2893 entries, 0 to 2892
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   subject  2831 non-null   object
 1   message  2893 non-null   object
 2   label    2893 non-null   int64
dtypes: int64(1), object(2)
memory usage: 67.9+ KB
```

```
#Checking the shape of the dataset
print("Shape of the dataset:", df.shape)
```

```
Shape of the dataset: (2893, 3)
```

```
#Checking for the null values
df.isnull().values.any()
```

```
True
```

```
#Checkin for the null values in columns
df.isnull().sum()
```

```
subject    62
message     0
label       0
dtype: int64
```

62 row are missing in the subject columns that means 62 emails are without subject heading.

Here, not dropping Nan rows for subject column as it of no use in building model.

```
#Checking total number of mails
print("Count of label:\n",df['label'].value_counts())
```

```
Count of label:
 0    2412
1     481
Name: label, dtype: int64
```

```
# Note:- Here in our dataset 1 stands for Spam mail and 0 stands for not a spam mail.
#Checking the Ratio of labels
print("Not a Spam Email Ratio i.e. 0 label:",round(len(df[df['label']==0])/len(df['label']),2)*100,"%")
print("Spam Email Ratio that is 1 label:",round(len(df[df['label']==1])/len(df['label']),2)*100,"%")
```

```
Not a Spam Email Ratio i.e. 0 label: 83.0 %
Spam Email Ratio that is 1 label: 17.0 %
```

so here 17 % of the data is a spam email

```
#Creating the new column for length of message column
df['length'] = df.message.str.len()
df.head()
```

	subject	message	label	length
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0	2856
1	NaN	lang classification grimes , joseph e . and ba...	0	1800
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0	1435
		a colleague and i are researching the		

```
#Converting all messages to lower case
df['message'] = df['message'].str.lower()
df.head()
```

	subject	message	label	length
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0	2856
1	NaN	lang classification grimes , joseph e . and ba...	0	1800
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0	1435
		a colleague and i are researching the		

```
# regular expressions
# Replace email addresses with 'email'
df['message'] = df['message'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')
```

```
# Replace URLs with 'webaddress'
df['message'] = df['message'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','webaddress')

# Replace currency symbols with 'moneysymb' (£ can by typed with ALT key + 156)
df['message'] = df['message'].str.replace(r'£|\$', 'dollers')

# Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
df['message'] = df['message'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

# Replace numeric characters with 'numbr'
df['message'] = df['message'].str.replace(r'\d+(\.\d+)?', 'numbr')


# Remove punctuation
df['message'] = df['message'].str.replace(r'[^\w\d\s]', ' ')

# Replace whitespace between terms with a single space
df['message'] = df['message'].str.replace(r'\s+', ' ')

# Remove leading and trailing whitespace
df['message'] = df['message'].str.replace(r'^\s+|\s+?$', '')


# now re-checking the data
df.head()
```

	subject	message	label	length
0	job posting - apple-iss research center	content length numbr apple iss research center...	0	2856
1	NaN	lang classification grimes joseph e and barbar...	0	1800
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas sa...	0	1435
		a colleague and i are researching the		

```
#Removing the stopwords
import string
import nltk
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])

df['message'] = df['message'].apply(lambda x: " ".join(term for term in x.split() if term not in stop_words))


# New column (clean_length) after puncuations,stopwords removal
df['clean_length'] = df.message.str.len()
df.head()
```

	subject	message	label	length	clean_length
0	job posting - apple-iss research center	content length numbr apple iss research center...	0	2856	2179
1	NaN	lang classification grimes joseph e barbara f ...	0	1800	1454
2	query : letter frequencies for text identifica...	posting inquiry sergei atamas satamas umabnet ...	0	1435	1064
3	risk	colleague researching differing degrees risk p...	0	324	210
4	request book information	earlier morning phone friend mine living south...	0	1046	629

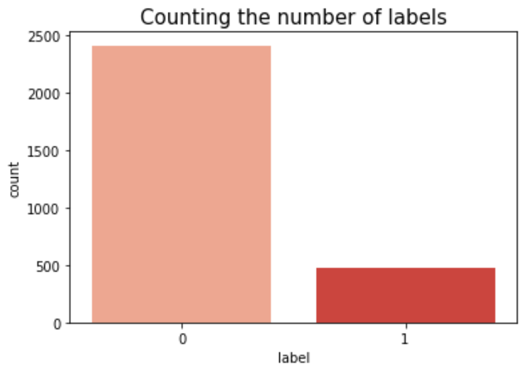
```
#Total length removal
print("Original Length:",df.length.sum())
print("Cleaned Length:",df.clean_length.sum())
print("Total Words Removed:",(df.length.sum()) - (df.clean_length.sum()))
```

```
Original Length: 9344743
Cleaned Length: 6767857
Total Words Removed: 2576886
```

```
#Graphical Visualisation for counting number of labels.
plt.figure(figsize=(6,4))
sns.countplot(df['label'],palette= 'Reds')
plt.title("Counting the number of labels",fontsize=15)
```

```
plt.xticks(rotation='horizontal')
plt.show()

print(df.label.value_counts())
```



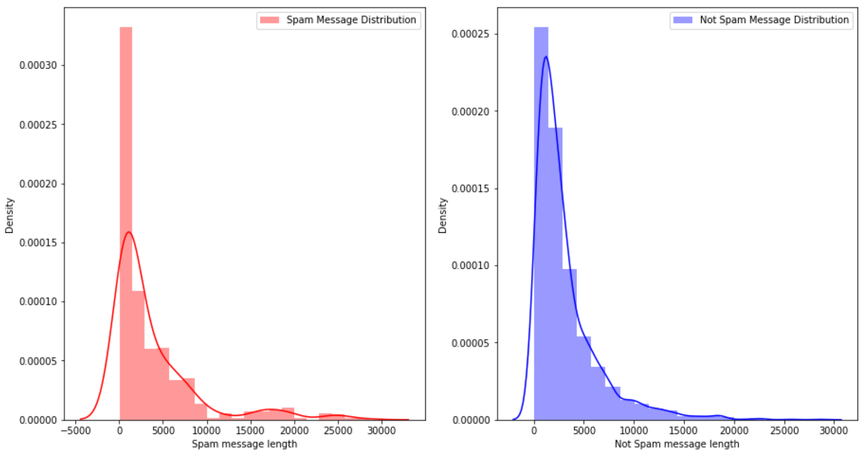

```
0    2412
1     481
Name: label, dtype: int64
```

```
#Message distribution before cleaning
f,ax = plt.subplots(1,2,figsize=(15,8))

sns.distplot(df[df['label']==1]['length'],bins=20, ax=ax[0],label='Spam Message Distribution',color='r')
ax[0].set_xlabel('Spam message length')
ax[0].legend()

sns.distplot(df[df['label']==0]['length'],bins=20, ax=ax[1],label='Not Spam Message Distribution',color='b')
ax[1].set_xlabel('Not Spam message length')
ax[1].legend()

plt.show()
```

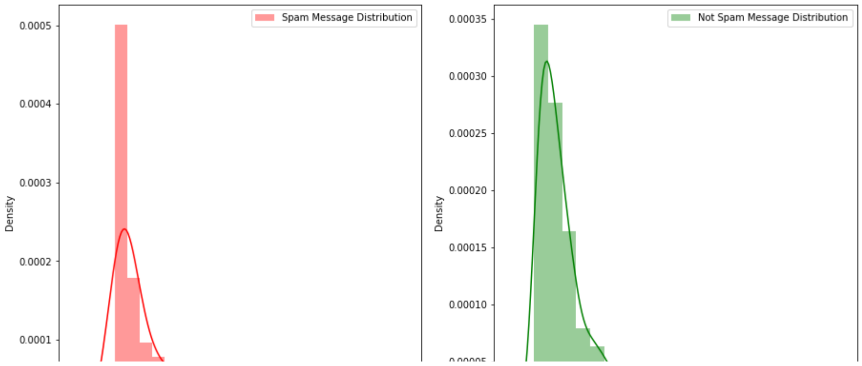



```
#Message distribution after cleaning
f,ax = plt.subplots(1,2,figsize=(15,8))

sns.distplot(df[df['label']==1]['clean_length'],bins=20, ax=ax[0],label='Spam Message Distribution',color='r')
ax[0].set_xlabel('Spam message length')
ax[0].legend()

sns.distplot(df[df['label']==0]['clean_length'],bins=20, ax=ax[1],label='Not Spam Message Distribution',color='g')
ax[1].set_xlabel('Not a Spam message length')
ax[1].legend()

plt.show()
```

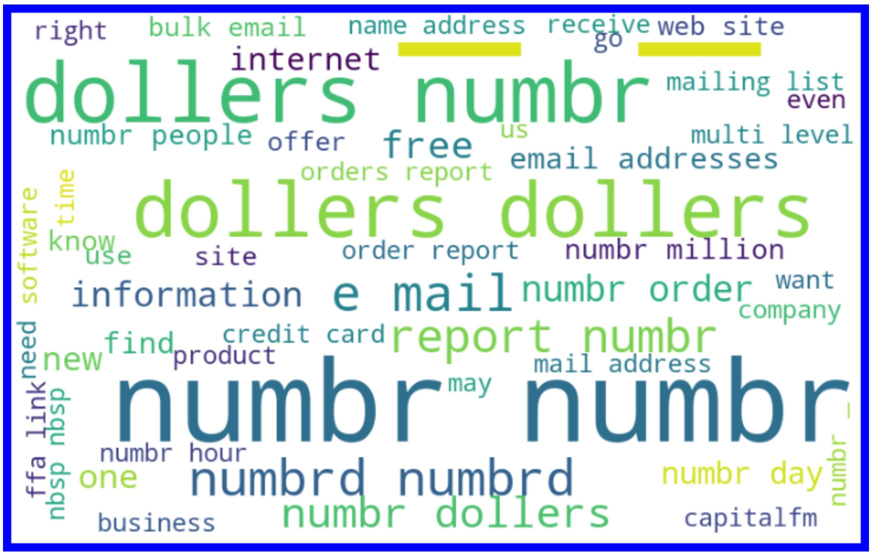



```
#Getting sense of loud words in spam
from wordcloud import WordCloud


spams = df['message'][df['label']==1]

spam_cloud = WordCloud(width=800,height=500,background_color='white',max_words=50).generate(' '.join(spams))

plt.figure(figsize=(10,8),facecolor='b')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```




```
#Getting sense of loud words in not-spam
from wordcloud import WordCloud

not_spams = df['message'][df['label']==0]

spam_cloud = WordCloud(width=800,height=500,background_color='white',max_words=50).generate(' '.join(not_spams))

plt.figure(figsize=(10,8),facecolor='b')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```


```
# Converting the text into vectors using TF-IDF, as text cannot be the input in the model
# 1. Convert text into vectors using TF-IDF
# 2. Instantiate MultinomialNB classifier
# 3. Split feature and label


tf_vec = TfidfVectorizer()

naive = MultinomialNB()

SVM = SVC(C=1.0, kernel='linear', degree=3 , gamma='auto')

decision = DecisionTreeClassifier()
classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")

clf = LogisticRegression()


features = tf_vec.fit_transform(df['message'])

X = features
y = df['label']


# Train and predict for naive bayes model
X_train,x_test,Y_train,y_test = train_test_split(X,y,random_state=42)

#test_size=0.20 random_state=42 test_size=0.15

naive.fit(X_train,Y_train)
y_pred= naive.predict(x_test)



print ('Final score = > ', accuracy_score(y_test,y_pred))
```

```
    Final score = >  0.8342541436464088
```

```
# Train and predict for SVM model
X_train,x_test,Y_train,y_test = train_test_split(X,y,random_state=42)

#test_size=0.20 random_state=42 test_size=0.15



SVM.fit(X_train,Y_train)
y_pred = SVM.predict(x_test)

print ('Final score = > ', accuracy_score(y_test,y_pred))
```

```
    Final score = >  0.9875690607734806
```

```
# train and predict for the Decision tree model
X_train,x_test,Y_train,y_test = train_test_split(X,y,random_state=42)
decision.fit(X_train,Y_train)
#test_size=0.20 random_state=42 test_size=0.15

y_pred = decision.predict(x_test)
print ('Final score = > ', accuracy_score(y_test,y_pred))
```

```
    Final score = >  0.9585635359116023
```

```
# train and predict uisng random forest classifier
X_train,x_test,Y_train,y_test = train_test_split(X,y,random_state=42)
classifier.fit(X_train, Y_train)
#test_size=0.20 random_state=42 test_size=0.15
```

```
y_pred= classifier.predict(x_test)
print ('Final score = > ', accuracy_score(y_test,y_pred))
```

```
Final score = >  0.9709944751381215
```

```
# train and predict uisng logistic regression
X_train,x_test,Y_train,y_test = train_test_split(X,y,random_state=42)
clf.fit(X_train, Y_train)
#test_size=0.20 random_state=42 test_size=0.15
y_pred= clf.predict(x_test)
print ('Final score = > ', accuracy_score(y_test,y_pred))
```

```
Final score = >  0.9475138121546961
```

```
y_pred
```

```
array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
tree.plot_tree(decision)
```

```
[Text(0.8492647058823529, 0.9827586206896551, 'X[18089] <= 0.015\ngini =
0.266\nsamples = 2169\nvalue = [1827, 342]'),
 Text(0.7612745098039215, 0.9482758620689655, 'X[41154] <= 0.009\ngini =
0.157\nsamples = 1901\nvalue = [1738, 163]'),
 Text(0.6950980392156862, 0.9137931034482759, 'X[32052] <= 0.076\ngini =
0.12\nsamples = 1853\nvalue = [1734, 119]'),
 Text(0.6411764705882353, 0.8793103448275862, 'X[8614] <= 0.053\ngini =
0.1\nsamples = 1828\nvalue = [1732, 96]'),
 Text(0.5803921568627451, 0.8448275862068966, 'X[19527] <= 0.068\ngini =
0.08\nsamples = 1804\nvalue = [1729, 75]'),
 Text(0.5372549019607843, 0.8103448275862069, 'X[23195] <= 0.019\ngini =
0.065\nsamples = 1780\nvalue = [1720, 60]'),
 Text(0.4980392156862745, 0.7758620689655172, 'X[49044] <= 0.029\ngini =
0.057\nsamples = 1771\nvalue = [1719, 52]'),
 Text(0.4666666666666667, 0.7413793103448276, 'X[51790] <= 0.083\ngini =
0.049\nsamples = 1760\nvalue = [1716, 44]'),
 Text(0.45098039215686275, 0.7068965517241379, 'X[7268] <= 0.044\ngini =
0.043\nsamples = 1755\nvalue = [1716, 39]'),
 Text(0.4196078431372549, 0.6724137931034483, 'X[11366] <= 0.082\ngini =
0.039\nsamples = 1750\nvalue = [1715, 35]'),
 Text(0.403921568627451, 0.6379310344827587, 'X[39836] <= 0.082\ngini =
0.036\nsamples = 1747\nvalue = [1715, 32]'),
 Text(0.3607843137254902, 0.603448275862069, 'X[30114] <= 0.082\ngini =
0.032\nsamples = 1741\nvalue = [1713, 28]'),
 Text(0.3215686274509804, 0.5689655172413793, 'X[11408] <= 0.076\ngini =
0.027\nsamples = 1735\nvalue = [1711, 24]'),
 Text(0.2901960784313726, 0.5344827586206896, 'X[39107] <= 0.017\ngini =
0.023\nsamples = 1728\nvalue = [1708, 20]'),
 Text(0.27450980392156865, 0.5, 'X[52548] <= 0.118\ngini = 0.021\nsamples =
1726\nvalue = [1708, 18]'),
 Text(0.25882352941176473, 0.46551724137931033, 'X[30119] <= 0.152\ngini =
0.018\nsamples = 1724\nvalue = [1708, 16]'),
 Text(0.24313725490196078, 0.43103448275862066, 'X[28346] <= 0.135\ngini =
0.016\nsamples = 1722\nvalue = [1708, 14]'),
 Text(0.20392156862745098, 0.39655172413793105, 'X[25295] <= 0.09\ngini =
0.014\nsamples = 1719\nvalue = [1707, 12]'),
 Text(0.17254901960784313, 0.3620689655172414, 'X[37433] <= 0.15\ngini =
0.012\nsamples = 1716\nvalue = [1706, 10]'),
 Text(0.1568627450980392, 0.3275862068965517, 'X[46093] <= 0.015\ngini =
0.01\nsamples = 1715\nvalue = [1706, 9]'),
 Text(0.1411764705882353, 0.29310344827586204, 'X[10017] <= 0.144\ngini =
0.009\nsamples = 1714\nvalue = [1706, 8]'),
 Text(0.12549019607843137, 0.25862068965517243, 'X[49937] <= 0.102\ngini =
0.008\nsamples = 1713\nvalue = [1706, 7]'),
 Text(0.10980392156862745, 0.22413793103448276, 'X[34059] <= 0.08\ngini =
0.007\nsamples = 1712\nvalue = [1706, 6]'),
 Text(0.09411764705882353, 0.1896551724137931, 'X[488] <= 0.062\ngini =
0.006\nsamples = 1711\nvalue = [1706, 5]'),
 Text(0.0784313725490196, 0.15517241379310345, 'X[14728] <= 0.177\ngini =
0.005\nsamples = 1710\nvalue = [1706, 4]'),
 Text(0.06274509803921569, 0.1206896551724138, 'X[36598] <= 0.068\ngini =
0.004\nsamples = 1709\nvalue = [1706, 3]'),
 Text(0.047058823529411764, 0.08620689655172414, 'X[27775] <= 0.22\ngini =
0.002\nsamples = 1708\nvalue = [1706, 2]'),
 Text(0.03137254901960784, 0.05172413793103448, 'X[21822] <= 0.073\ngini =
0.001\nsamples = 1707\nvalue = [1706, 1]'),
 Text(0.01568627450980392, 0.017241379310344827, 'gini = 0.0\nsamples =
```

```
# Checking Classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       585
           1       0.99      0.73      0.84       139

    accuracy                           0.95       724
   macro avg       0.97      0.87      0.91       724
weighted avg       0.95      0.95      0.94       724
```

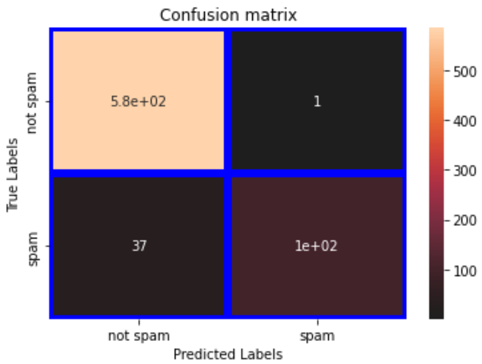
```
# plot confusion matrix heatmap
conf_mat = confusion_matrix(y_test,y_pred)

ax=plt.subplot()

sns.heatmap(conf_mat,annot=True,ax=ax,linewidths=5,linecolor='b',center=0)

ax.set_xlabel('Predicted Labels');ax.set_ylabel('True Labels')

ax.set_title('Confusion matrix')
ax.xaxis.set_ticklabels(['not spam','spam'])
ax.yaxis.set_ticklabels(['not spam','spam'])
plt.show()
```




```
# SVM performs best among all the classification models
```

Colab paid products - Cancel contracts here

✓ 0s completed at 15:25