Assignment No. 04

Name : Tejas Shyam Fulumbarkar

Subject : Machine Learning

Class : TE-IT (B)

[2]:
```
import numpy as np
import pandas as pd
```

[3]:
```
df = pd.read_csv('../input/sms-spam-collection-data-set/SMSSpamCollection',sep='\t',names=['label','text'])
```

[4]:
```
df
```

[4]:

	label	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

[5]:
```
df.shape
```

[5]:
```
(5572, 2)
```

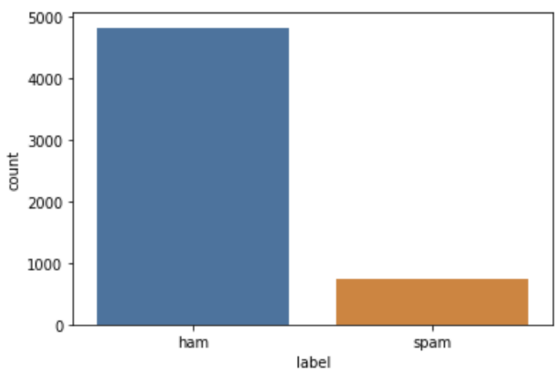
[6]:
```
import nltk #!pip install nltk
```

[7]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[7]:
```
True
```

```
[10]: from nltk.corpus import stopwords
      swords = stopwords.words('english')

[11]: clean = [word for word in word_tokenize(sent) if word not in swords]

[12]: clean

[12]: ['How', 'friends', '?']

[13]: # Stemming words with NLTK
      from nltk.stem import PorterStemmer
      ps = PorterStemmer()
      clean = [ps.stem(word) for word in word_tokenize(sent)
               if word not in swords]
      clean

[13]: ['how', 'friend', '?']

[14]: sent = 'Hello friends! How are you? We will learning python today'

[15]: def clean_text(sent):
          tokens = word_tokenize(sent)
          clean = [word for word in tokens if word.isdigit() or word.isalpha()]
          clean = [ps.stem(word) for word in clean
                   if word not in swords]
          return clean

[16]: clean_text(sent)

[16]: ['hello', 'friend', 'how', 'we', 'learn', 'python', 'today']

[17]: # Pre-processing
      from sklearn.feature_extraction.text import TfidfVectorizer

[18]: tfidf = TfidfVectorizer(analyzer=clean_text)

[19]: x = df['text']
      y = df['label']

[20]: x_new = tfidf.fit_transform(x)

[21]: x.shape

[21]: (5572,)

[22]: x_new.shape

[22]: (5572, 6513)

[23]: # tfidf.get_feature_names()

[24]: import seaborn as sns
      sns.countplot(x=y)

[24]: <AxesSubplot:xlabel='label', ylabel='count'>
```



[25]:
```
#cross validation
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_new,y,test_size=0.25,random_state=1)
```

[26]:
```
print(f"Size of splitted data")
print(f"x_train {x_train.shape}")
print(f"y_train {y_train.shape}")
print(f"y_test {x_test.shape}")
print(f"y_test {y_test.shape}")
```

```
Size of splitted data
x_train (4179, 6513)
y_train (4179,)
y_test (1393, 6513)
y_test (1393,)
```

[27]:
```
from sklearn.naive_bayes import GaussianNB
```

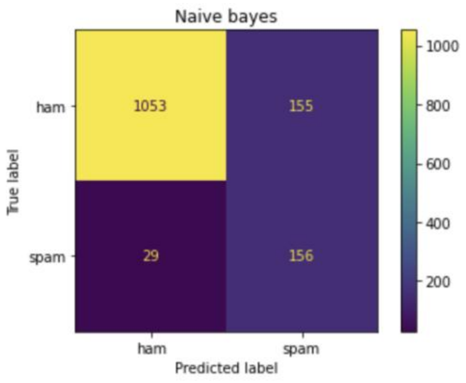
[28]:
```
nb = GaussianNB()
nb.fit(x_train.toarray(),y_train)
y_pred_nb = nb.predict(x_test.toarray())
```

[29]:
```
y_test.value_counts()
```

t[29]:
```
ham     1208
spam     185
Name: label, dtype: int64
```

[30]:
```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
```

[31]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_nb)
plt.title('Naive bayes')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_nb)}")
print(classification_report(y_test,y_pred_nb))
```



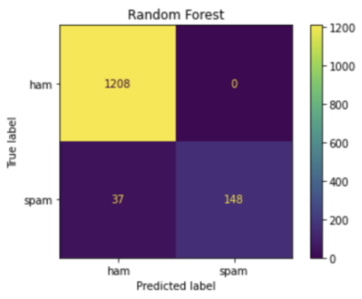
```
Accuracy is 0.867910983488873
              precision    recall  f1-score   support

         ham       0.97      0.87      0.92      1208
        spam       0.50      0.84      0.63       185

    accuracy                           0.87      1393
   macro avg       0.74      0.86      0.77      1393
weighted avg       0.91      0.87      0.88      1393
```

[32]:
```python
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier(random_state=1)
model_rf.fit(x_train,y_train)
```

[32]:
```
RandomForestClassifier(random_state=1)
```

[33]:
```python
y_pred_rf = model_rf.predict(x_test) #float
```

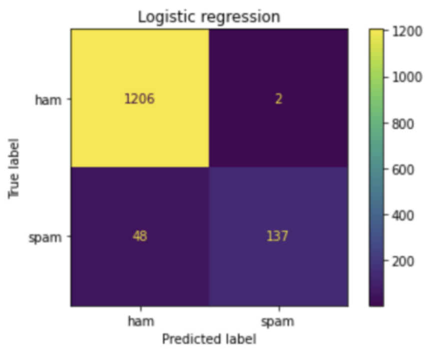
[34]:
```python
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_rf)
plt.title('Random Forest')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_rf)}")
print(classification_report(y_test,y_pred_rf))
```



```
Accuracy is 0.9734386216798278
              precision    recall  f1-score   support

         ham       0.97      1.00      0.98      1208
        spam       1.00      0.80      0.89       185

    accuracy                           0.97      1393
   macro avg       0.99      0.90      0.94      1393
weighted avg       0.97      0.97      0.97      1393
```

[35]:
```
from sklearn.linear_model import LogisticRegression
model_lr = LogisticRegression(random_state=1)

model_lr.fit(x_train,y_train)
y_pred_lr = model_lr.predict(x_test)
```

[36]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_lr)
plt.title('Logistic regression')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_lr)}")
print(classification_report(y_test,y_pred_lr))
```



```
 Accuracy is 0.9641062455132807
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98      1208
        spam       0.99      0.74      0.85       185

    accuracy                           0.96      1393
   macro avg       0.97      0.87      0.91      1393
weighted avg       0.96      0.96      0.96      1393
```