In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('spam.csv')
```

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

Out[3]:

	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

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
v1               0
v2               0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
dtype: int64
```

In [6]:
```
df['v1'].value_counts()
```

Out[6]:
```
ham     4825
spam     747
Name: v1, dtype: int64
```

In [7]:
```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], inplace=True, axis=1)
```

In [8]:
```
df
```

Out[8]:

	v1	v2
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 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

In [9]:
```
email_df = df.where((pd.notnull(df)),'')
```

In [10]:
```
email_df.loc[email_df['v1'] == 'spam', 'v1',] = 0
email_df.loc[email_df['v1'] == 'ham', 'v1',] = 1
```

In [11]:
```
from sklearn.model_selection import train_test_split
X = email_df['v2']
y = email_df['v1']
```

In [12]:
```
y
```

Out[12]:
```
0       1
1       1
2       0
3       1
4       1
       ..
5567    0
5568    1
5569    1
5570    1
5571    1
Name: v1, Length: 5572, dtype: object
```

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=3)
```

In [14]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [15]:
```
feature_extraction = TfidfVectorizer(min_df = 1, stop_words='english', lowercase='True')

X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)

# convert Y_train and Y_test values as integers

y_train = y_train.astype('int')
y_test = y_test.astype('int')
```

In [16]:
```
print(X_train_features)
```

```
  (0, 742)      0.32207229533730536
  (0, 3962)     0.24116082432124387
  (0, 4279)     0.38930423610458832
  (0, 6580)     0.20305518394534605
  (0, 3375)     0.32207229533730536
  (0, 2116)     0.38519642807943744
  (0, 3126)     0.44030352345644808
  (0, 3251)     0.25888050029555985
  (0, 3369)     0.21816477736422235
  (0, 4497)     0.29108876331514199
  (1, 4045)     0.3804311983169959
  (1, 6850)     0.43060158942777422
  (1, 6397)     0.47691368595540388
  (1, 6422)     0.56525090766544626
  (1, 7420)     0.35055697107320353
  (2, 934)      0.49175984657232773
  (2, 2103)     0.42972812260098503
  (2, 3899)     0.40088501350982736
  (2, 2220)     0.41348465293484624
  (2, 5806)     0.49175984657232773
  (3, 6121)     0.49038631688693604
  (3, 1595)     0.59270918541494291
  (3, 1838)     0.37086806414877708
  (3, 7430)     0.52026335711003087
  (4, 2523)     0.74193190914596392
  :     :
  (4452, 2116)  0.3092200696948929
  (4453, 1000)  0.6760129013031282
  (4453, 7250)  0.5787739591782677
  (4453, 1758)  0.4561000564008829
  (4454, 3019)  0.4261890999786
  (4454, 2080)  0.3809693742808703
  (4454, 3078)  0.3447559300591444
  (4454, 1995)  0.4166919007849217
  (4454, 1050)  0.3193206011160604
  (4454, 7323)  0.3116626383410377
  (4454, 5351)  0.4261890999786
  (4455, 1148)  0.38998123077430413
  (4455, 6413)  0.38998123077430413
  (4455, 6341)  0.25697343671652706
  (4455, 2755)  0.32263237545940581
  (4455, 7335)  0.29159496226395065
  (4455, 7384)  0.30284819955557642
  (4455, 2102)  0.31364683845260087
  (4455, 4235)  0.30616657078392584
  (4455, 3745)  0.16807158405536876
  (4455, 4755)  0.35860460546223444
  (4456, 6098)  0.53043503132991551
  (4456, 6114)  0.53043503132991551
  (4456, 1386)  0.44600336316446079
  (4456, 4541)  0.48821933148688146
```

In [17]:
```
from sklearn.linear_model import LogisticRegression as LR
```

In [18]:
```
lr = LR()
lr.fit(X_train_features, y_train)
```

Out[18]:
```
LogisticRegression()
```

In [19]:
```
y_lr = lr.predict(X_test_features)
```

In [22]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_lr, y_test)
```

Out[22]:
```
0.9623318385650225
```

In [23]:
```
from sklearn.neighbors import KNeighborsClassifier as KNN
knn= KNN()
knn.fit(X_train_features, y_train)
```

Out[23]:
```
KNeighborsClassifier()
```

In [24]:
```
y_knn = knn.predict(X_test_features)
```

In [25]:
```
accuracy_score(y_knn, y_test)
```

Out[25]:
```
0.9058295964125560
```

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier as RF
rf = RF()
rf.fit(X_train_features, y_train)
```

In [27]:
```
y_rf = rf.predict(X_test_features)
```

In [28]:
```
accuracy_score(y_rf, y_test)
```

Out[28]:
```
0.9784753363228700
```

In [29]:
```
from sklearn.tree import DecisionTreeClassifier as DTC
dtc = DTC()
```

In [30]:
```
dtc.fit(X_train_features, y_train)
y_dtc = dtc.predict(X_test_features)
accuracy_score(y_dtc, y_test)
```

Out[30]:
```
0.9650224215246637
```

In [31]:
```
from sklearn.ensemble import AdaBoostClassifier as ADA
ada = ADA()
ada.fit(X_train_features, y_train)
```

Out[31]:
```
AdaBoostClassifier()
```

In [32]:
```
y_ada = ada.predict(X_test_features)
```

In [33]:
```
accuracy_score(y_ada, y_test)
```

Out[33]:
```
0.9704035874439462
```

RANDOM FOREST CLASSIFIER

In [ ]:
```

```