In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import gensim
```

```
/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/pandas/core/computation/expressions.py:20
: UserWarning: Pandas requires version '2.7.3' or newer of 'numexpr' (version '2.7.1' currently install
ed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

In [2]:
```
# pip install gensim
```

In [3]:
```
messages = pd.read_csv('spam.csv',encoding = 'latin-1')
messages = messages.drop(labels = ['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis = 1)
messages.columns = ['labels','text']
messages.head()
```

Out[3]:

	labels	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...

In [4]:
```
messages['text_clean'] = messages['text'].apply(lambda x : gensim.utils.simple_preprocess(x))
messages.head()
```

Out[4]:

	labels	text	text_clean
0	ham	Go until jurong point, crazy.. Available only ...	[go, until, jurong, point, crazy, available, o...
1	ham	Ok lar... Joking wif u oni...	[ok, lar, joking, wif, oni]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	[free, entry, in, wkly, comp, to, win, fa, cup...
3	ham	U dun say so early hor... U c already then say...	[dun, say, so, early, hor, already, then, say]
4	ham	Nah I don't think he goes to usf, he lives aro...	[nah, don, think, he, goes, to, usf, he, lives...

In [5]:
```
messages['text_clean'][0]
```

Out[5]:
```
['go',
 'until',
 'jurong',
 'point',
 'crazy',
 'available',
 'only',
 'in',
 'bugis',
 'great',
 'world',
 'la',
 'buffet',
 'cine',
 'there',
 'got',
 'amore',
 'wat']
```

In [6]:
```
# convert lables into numerical format

messages['labels'] = messages['labels'].map({'ham':1,'spam':0})
messages.head()
```

Out[6]:

	labels	text	text_clean
0	1	Go until jurong point, crazy.. Available only ...	[go, until, jurong, point, crazy, available, o...
1	1	Ok lar... Joking wif u oni...	[ok, lar, joking, wif, oni]
2	0	Free entry in 2 a wkly comp to win FA Cup fina...	[free, entry, in, wkly, comp, to, win, fa, cup...
3	1	U dun say so early hor... U c already then say...	[dun, say, so, early, hor, already, then, say]
4	1	Nah I don't think he goes to usf, he lives aro...	[nah, don, think, he, goes, to, usf, he, lives...

In [7]:
```
messages.shape
```

Out[7]: (5572, 3)

In [8]:
```
X = messages['text_clean']
y = messages['labels']
```

In [9]:
```
# split data into train and test data
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
```

In [12]:
```
# Train the word2vec model
w2v_model = gensim.models.Word2Vec(X_train,vector_size = 100,window = 5,min_count = 2)
```

In [14]:
```
w2v_model.wv.index_to_key
```

```
'number',
'way',
'yes',
'give',
'more',
'prize',
'www',
'hey',
'doing',
'had',
'make',
'should',
'say',
'said',
'won',
'after',
'really',
'message',
'yeah',
'right',
```

In [18]:
```
# Find the most similar word to input word
w2v_model.wv.most_similar('birthday')
```

Out[18]:
```
[('back', 0.9991286993026733),
 ('me', 0.9990671873092651),
 ('miss', 0.9990516304969788),
 ('one', 0.999048113822937),
 ('and', 0.9990416765213013),
 ('feel', 0.9990386366844177),
 ('life', 0.9990307688713074),
 ('even', 0.999019205570221),
 ('amp', 0.9990180134773254),
 ('so', 0.999013364315033)]
```

In [19]:

```
words = set(w2v_model.wv.index_to_key)
```

In [20]:

```
len(words)
```

Out[20]:

```
3391
```

In [30]:

```
X_train_vec = np.array([np.array([w2v_model.wv[i] for i in ls if i in words]) for ls in X_train])
X_test_vec = np.array([np.array([w2v_model.wv[i] for i in ls if i in words]) for ls in X_test])
```

```
<ipython-input-30-1769936cebc3>:1: VisibleDeprecationWarning: Creating an ndarray from ragged nested se
quences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is d
eprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
  X_train_vec = np.array([np.array([w2v_model.wv[i] for i in ls if i in words]) for ls in X_train])
<ipython-input-30-1769936cebc3>:2: VisibleDeprecationWarning: Creating an ndarray from ragged nested se
quences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is d
eprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
  X_test_vec = np.array([np.array([w2v_model.wv[i] for i in ls if i in words]) for ls in X_test])
```

In [32]:

```
for i ,v in enumerate(X_train_vec):
    print(len(X_train.iloc[i]),len(v)) # printing lenght of sentence and lenght of sentense vector
```

```
22 20
23 23
9 8
4 3
24 20
23 23
25 15
12 12
17 17
3 3
9 9
9 8
6 6
19 19
22 22
4 3
19 17
23 23
26 23
```

In [35]:

```
# average the word vectors for each sentence (if we provide un even sized vectors, we may get an err
# (assign zero if model did not learn anything from training)
X_train_vec_avg = []
for v in X_train_vec:
    if v.size:
        X_train_vec_avg.append(v.mean(axis = 0))
    else:
        X_train_vec_avg.append(np.zeros(100,dtype = float))
X_test_vec_avg = []
for v1 in X_test_vec:
    if v1.size:
        X_test_vec_avg.append(v1.mean(axis = 0))
    else:
        X_test_vec_avg.append(np.zeros(100,dtype = float))
```

In [36]:

```
for i ,v in enumerate(X_train_vec_avg):
    print(len(X_train.iloc[i]),len(v))
```

```
22 100
23 100
9 100
4 100
24 100
23 100
25 100
12 100
17 100
3 100
9 100
9 100
6 100
19 100
22 100
4 100
19 100
23 100
26 100
```

In [45]:

```
# fit RF model

from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf_model = rf.fit(X_train_vec_avg,y_train.values)
y_pred = rf_model.predict(X_test_vec_avg)
```

In [46]:

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

pr = precision_score(y_test,y_pred)
rc = recall_score(y_test,y_pred)
acc = accuracy_score(y_test,y_pred)

print("Precision : ", pr)
print("Recall : ", rc)
print("Accuracy : ", acc)
```

```
Precision :  0.9754350051177073
Recall :  0.9875647668393782
Accuracy :  0.967713004484305
```

In [ ]:

```

```