```
from google.colab import drive
drive.mount('/content/drive')

     Mounted at /content/drive
```

```
import pandas as pd
import re
import os
```

```
df = pd.read_csv('/content/drive/MyDrive/NLP Project/fake-news/train.csv')
df.head()
```

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get	Consortiumnews.com	Why the Truth Might Get You Fired October	1

```
# Splits into x & y
x = df.drop('label',axis=1)
x.head()
```

	id	title	author	text
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...

```
y = df['label']
y.head()

     0    1
     1    0
     2    1
     3    1
     4    1
     Name: label, dtype: int64
```

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer,HashingVectorizer
```

```
df.shape

     (20800, 5)
```

```
df = df.dropna()
df.shape

     (18285, 5)
```

```
df.head(10)
```

	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Get You Fired	Consortiumnews.com	Get You Fired October	1

```
messages = df.copy()
```

```
#### After dropping null values some of the index get dropped so we can reset the index
messages.reset_index(inplace=True,drop=True)
messages.head(10)
```

	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

```
messages['text'][6]
```

```
'PARIS  —  France chose an idealistic, traditional   candidate in Sunday’s prima
ry to represent the Socialist and   parties in the presidential election this spr
ing. The candidate, Benoît Hamon, 49, who ran on the slogan that he would “make F
rance’s heart beat,” bested Manuel Valls, the former prime minister, whose campai
gn has promoted more   policies and who has a strong    background. Mr. Hamon app
eared to have won by a wide margin, with incomplete returns showing him with an e
stimated 58 percent of the vote to Mr. Valls’s 41 percent. “Tonight the left hold
s its head up high again it is looking to the future,” Mr. Hamon said, addressing
```

```
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()


import nltk
nltk.download('stopwords')
corpus = []
for i in range(len(messages)):
    review = re.sub('[^A-Za-z]',' ',messages['text'][i])
    review = review.lower()
    review = review.split()
    review = [ps.stem(word) for word in review if word not in stopwords.words('english')]
    review = ' '.join(review)
    corpus.append(review)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
corpus[3]
```

```
'video civilian kill singl us airstrik identifi rate civilian kill american airst
rik afghanistan higher us engag activ combat oper photo hellfir missil load onto
us militari reaper drone afghanistan staff sgt brian ferguson u air forc bureau a
bl identifi civilian kill singl us drone strike afghanistan last month biggest lo
ss civilian life one strike sinc attack medecin san frontier hospit msf last octo
b us claim conduct counter terror strike islam state fighter hit nangarhar provin
c missil septemb next day unit nation issu unusu rapid strong statement say strik
e kill civilian injur other gather hous celebr tribal elder return pilgrimag mecc
```

```
# Applying Tfidf Vectorizer
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
cv = TfidfVectorizer(max_features=5000,ngram_range=(1,3))
x = cv.fit_transform(corpus).toarray()
```

```
x.shape
```

```
(18285, 5000)
```

```
y = messages['label']
```

```
# Train Test Split
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.33,random_state=0)
```

```
cv.get_feature_names_out()[:20]
```

```
array(['aaron', 'abandon', 'abc', 'abe', 'abedin', 'abil', 'abl', 'abort',
       'abroad', 'absenc', 'absolut', 'absorb', 'absurd', 'abu', 'abus',
       'academ', 'academi', 'acceler', 'accept', 'access'], dtype=object)
```

```
cv.get_params()
```

```
{'analyzer': 'word',
 'binary': False,
 'decode_error': 'strict',
 'dtype': numpy.float64,
 'encoding': 'utf-8',
 'input': 'content',
 'lowercase': True,
 'max_df': 1.0,
 'max_features': 5000,
 'min_df': 1,
 'ngram_range': (1, 3),
 'norm': 'l2',
 'preprocessor': None,
 'smooth_idf': True,
 'stop_words': None,
 'strip_accents': None,
 'sublinear_tf': False,
 'token_pattern': '(?u)\\b\\w\\w+\\b',
 'tokenizer': None,
 'use_idf': True,
 'vocabulary': None}
```

```
count_df = pd.DataFrame(xtrain,columns=cv.get_feature_names_out())
count_df.head()
```

	aaron	abandon	abc	abe	abedin	abil	abl	abort	abroad	absenc	...	young
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.200698
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000

```
# See full source and example:
# http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

# This function prints and plots the confusion matrix. Normalization can br applied by setting normalize=True
```

```
import matplotlib.pyplot as plt
def plot_confusion_matrix(cm,classes,normalize=True,title='Confusion matrix',cmap=plt.cm.Blues):

    plt.imshow(cm,interpolation='nearest',cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks,classes,rotation=45)
    plt.yticks(tick_marks,classes)
    if normalize:
        cm = cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
        print('Normalized Confusion matrix')
    else:
        print('Confusion matrix without Normalization')

    thresh = cm.max()/2
    for i,j in itertools.product(range(cm.shape[0]),range(cm.shape[1])):
        plt.text(j,i,cm[i,j],
                 horizontalalignment = 'center',
                 color = 'white' if cm[i,j] > thresh else 'black')

    plt.tight_layout()
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
```

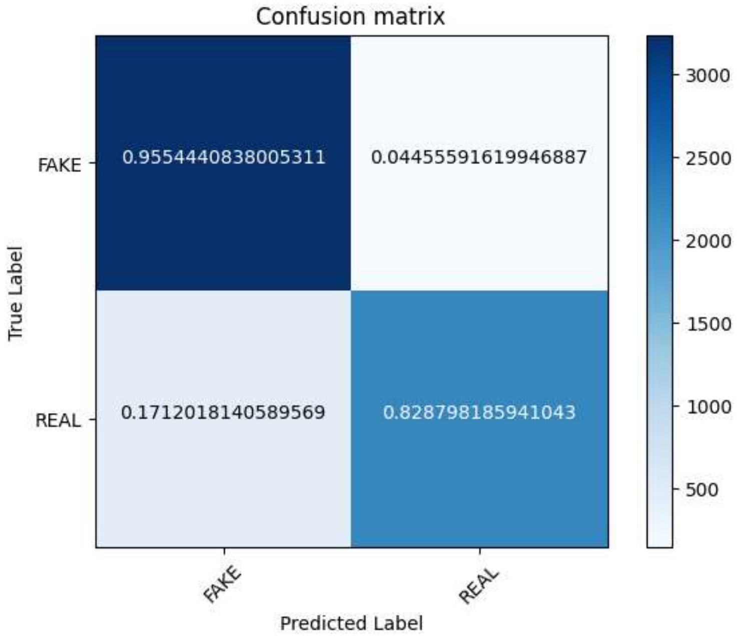
MultinomialNB Algorithm

```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
```

```
from sklearn import metrics
import numpy as np
import itertools
```

```
classifier.fit(xtrain,ytrain)
ypred = classifier.predict(xtest)
score = metrics.accuracy_score(ytest,ypred)
print('accuracy :   %0.3f'% score)
cm = metrics.confusion_matrix(ytest,ypred)
plot_confusion_matrix(cm,classes=['FAKE','REAL'])
```

```
accuracy :   0.900
Normalized Confusion matrix
```



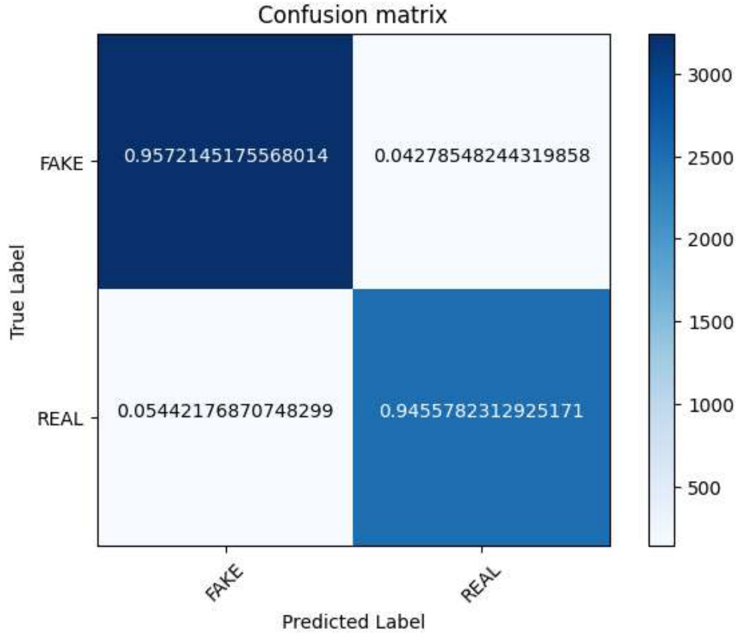
Passive Aggressive Classifier Algorithm

```
from sklearn.linear_model import PassiveAggressiveClassifier
linear_clf = PassiveAggressiveClassifier(n_iter_no_change=50)
```

```
linear_clf.fit(xtrain,ytrain)
ypred = linear_clf.predict(xtest)
score = metrics.accuracy_score(ytest,ypred)
print('accuracy :   %0.3f'% score)
```

```
cm = metrics.confusion_matrix(ytest,ypred)
plot_confusion_matrix(cm,classes=['FAKE','REAL'])
```

```
accuracy :   0.952
Normalized Confusion matrix
```



Multinomial Classifier with HPT

```
classifier = MultinomialNB(alpha=0.1)
```

```
previous_score = 0
for alpha in np.arange(0,1,0.1):
    sub_classifier = MultinomialNB(alpha=alpha)
    sub_classifier.fit(xtrain,ytrain)
    ypred = sub_classifier.predict(xtest)
    score = metrics.accuracy_score(ytest,ypred)
    if score > previous_score:
        classifier = sub_classifier
    print('Alpha: {}, score : {}'.format(alpha,score))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py:629: FutureWarning: The default value for `force_alpha` will change to `T
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/naive_bayes.py:635: UserWarning: alpha too small will result in numeric errors, setting
  warnings.warn(
Alpha: 0.0, score : 0.9022369511184756
Alpha: 0.1, score : 0.9017398508699255
Alpha: 0.2, score : 0.9020712510356255
Alpha: 0.30000000000000004, score : 0.9022369511184756
Alpha: 0.4, score : 0.9020712510356255
Alpha: 0.5, score : 0.9014084507042254
Alpha: 0.6000000000000001, score : 0.9015741507870754
Alpha: 0.7000000000000001, score : 0.9012427506213753
Alpha: 0.8, score : 0.9009113504556753
Alpha: 0.9, score : 0.9005799502899752
```

```
# Get features names
feature_names = cv.get_feature_names_out()
```

```
feature_names
```

```
array(['aaron', 'abandon', 'abc', ..., 'zionist', 'zone', 'zu'],
      dtype=object)
```

```
classifier.feature_log_prob_[0] # (It returns coefficient & replaced by .coef_)
```

```
array([ -9.12693158,  -8.85404287,  -8.37578731, ..., -10.02440559,
        -8.72415806, -11.33915105])
```

```
# Most real
sorted(zip(classifier.feature_log_prob_[0],feature_names),reverse=True)[:20]
```

```
[(-4.8806604137777105, 'mr'),
 (-5.139321704939019, 'said'),
 (-5.4174508002237785, 'trump'),
 (-5.945904009383865, 'mr trump'),
 (-6.010439748361563, 'presid'),
 (-6.0597181356282315, 'state'),
 (-6.176438578227852, 'ms'),
 (-6.186978921465968, 'would'),
 (-6.190664081844223, 'one'),
 (-6.203719010990082, 'peopl'),
 (-6.213971635691953, 'new'),
 (-6.216849578098212, 'year'),
 (-6.31677136316521, 'time'),
 (-6.330066788761383, 'like'),
 (-6.405502286042062, 'report'),
 (-6.469793020402705, 'also'),
 (-6.506211037844363, 'say'),
 (-6.528582134080173, 'news'),
 (-6.532878319242008, 'american'),
 (-6.539461379556004, 'polic')]
```

```
# Most Fake
sorted(zip(classifier.feature_log_prob_[0],feature_names))[:20]
```

```
[(-11.384569945135524, 'auf'),
 (-11.384569945135524, 'en el'),
 (-11.384569945135524, 'fli zone'),
 (-11.384569945135524, 'html'),
 (-11.384569945135524, 'http co'),
 (-11.384569945135524, 'http www'),
 (-11.384569945135524, 'infowar life'),
 (-11.384569945135524, 'pic twitter com'),
 (-11.384569945135524, 'ufo'),
 (-11.384569945135524, 'utm'),
 (-11.363715819078537, 'brain forc'),
 (-11.362130833778155, 'twitter com'),
 (-11.348963751279618, 'oligarchi'),
 (-11.348833232513876, 'infowar com'),
 (-11.33915104590614, 'zu'),
 (-11.329820752669196, 'ein'),
 (-11.327340283772534, 'ist'),
 (-11.307187908874361, 'como'),
 (-11.306995492990557, 'kadzik'),
 (-11.295088488931421, 'una')]
```

Hashing Vectorizer

```
hs_vectorizer = HashingVectorizer(n_features=15000,alternate_sign=False) # altenate_sign = False replaces non_negative = True
x = hs_vectorizer.fit_transform(corpus).toarray()
```

```
x.shape
```

```
(18285, 15000)
```

```
x
```

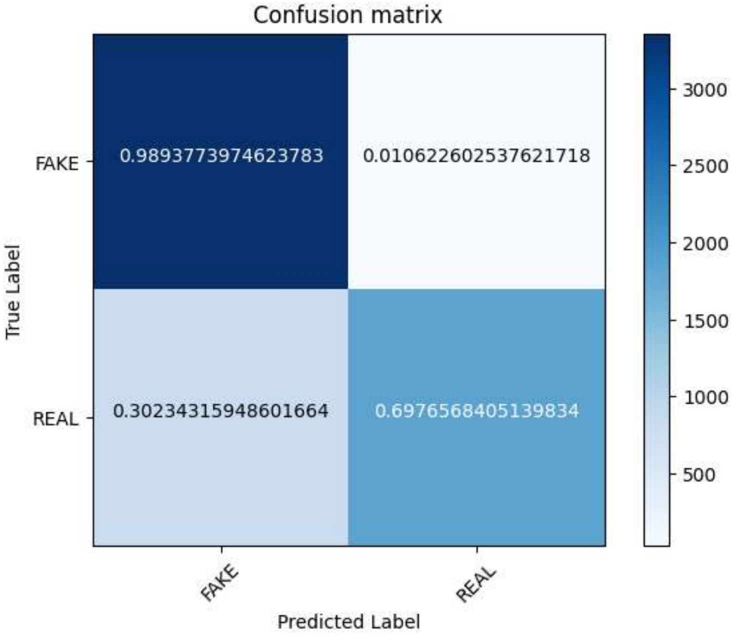
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
# Train Test Split
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.33,random_state=0)
```

```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(xtrain,ytrain)
ypred = classifier.predict(xtest)
```

```
score = metrics.accuracy_score(ytest,ypred)
print('accuracy :   %0.3f'% score)
cm = metrics.confusion_matrix(ytest,ypred)
plot_confusion_matrix(cm,classes=['FAKE','REAL'])
```

```
accuracy :   0.861
Normalized Confusion matrix
```



Embedding Vectorizer

```
import tensorflow as tf
from tensorflow.keras.layers import Embedding,LSTM,Dense
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.callbacks import EarlyStopping
```

```
# Vocabulary size
voc_size = 5000
```

Onehot Representation

```
onehot_repr = [one_hot(words,voc_size) for words in corpus]
print(onehot_repr)
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

```
sent_length = 20
embeded_docs = pad_sequences(onehot_repr,maxlen=sent_length)
print(embeded_docs)
```

```
[[ 436 4898  435 ... 2097  311  848]
 [3773 3230 1009 ... 1880 1321 3125]
 [2418 4616 1503 ... 1003 1865 4910]
 ...
 [3908  842 1313 ... 1572 1818 1795]
 [ 289 2086 3661 ... 3860  144 3937]
 [  92  445 1965 ... 1759  936 4691]]
```

```
embeded_docs[0]
```

```
array([ 436, 4898,  435, 2237, 4501,  830, 4048, 1552,  571, 2305,  359,
         66,  848,  333, 3997, 1510, 3356, 2097,  311,  848], dtype=int32)
```

```
len(embeded_docs)
```

```
18285
```

```
import numpy as np
xfinal = np.array(embeded_docs)
yfinal = np.array(y)
```

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(xfinal,yfinal,test_size=0.33,random_state=42)
```

LSTM Model Training

```
# Creating model
embedding_vector_features = 40
model = Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(LSTM(100))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 20, 40)            200000

 lstm (LSTM)                 (None, 100)               56400

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 256,501
Trainable params: 256,501
Non-trainable params: 0
_________________________________________________________________
```

Model Training

```
# Finally Training
model.fit(xtrain,ytrain,validation_data=(xtest,ytest),epochs=40,batch_size=64)
```

```
192/192 [==============================] - 2s 11ms/step - loss: 0.0125 - accuracy: 0.9963 - val_loss: 1.1235 - val_accuracy: 0.8272
Epoch 13/40
192/192 [==============================] - 2s 9ms/step - loss: 0.0061 - accuracy: 0.9984 - val_loss: 1.1751 - val_accuracy: 0.8151
```

```
     192/192 [==============================] - 2s 9ms/step - loss: 5.4964e-04 - accuracy: 0.9999 - val_loss: 1.8742 - val_accuracy: 0.819
     Epoch 26/40
     192/192 [==============================] - 2s 8ms/step - loss: 5.6867e-04 - accuracy: 0.9999 - val_loss: 1.9102 - val_accuracy: 0.8189
     Epoch 27/40
     192/192 [==============================] - 2s 13ms/step - loss: 5.3845e-04 - accuracy: 0.9999 - val_loss: 1.8645 - val_accuracy: 0.819
     Epoch 28/40
     192/192 [==============================] - 2s 11ms/step - loss: 5.9110e-04 - accuracy: 0.9999 - val_loss: 1.9822 - val_accuracy: 0.819
     Epoch 29/40
     192/192 [==============================] - 2s 10ms/step - loss: 4.9761e-04 - accuracy: 0.9999 - val_loss: 2.0332 - val_accuracy: 0.819
     Epoch 30/40
     192/192 [==============================] - 2s 9ms/step - loss: 5.1185e-04 - accuracy: 0.9999 - val_loss: 1.9935 - val_accuracy: 0.8179
     Epoch 31/40
     192/192 [==============================] - 2s 9ms/step - loss: 5.5276e-04 - accuracy: 0.9999 - val_loss: 2.0916 - val_accuracy: 0.8179
     Epoch 32/40
     192/192 [==============================] - 2s 10ms/step - loss: 5.1772e-04 - accuracy: 0.9999 - val_loss: 2.1180 - val_accuracy: 0.818
     Epoch 33/40
     192/192 [==============================] - 2s 10ms/step - loss: 5.2730e-04 - accuracy: 0.9999 - val_loss: 2.1218 - val_accuracy: 0.819
     Epoch 34/40
     192/192 [==============================] - 3s 14ms/step - loss: 5.0920e-04 - accuracy: 0.9999 - val_loss: 2.1805 - val_accuracy: 0.818
     Epoch 35/40
     192/192 [==============================] - 2s 12ms/step - loss: 5.2431e-04 - accuracy: 0.9999 - val_loss: 2.1787 - val_accuracy: 0.819
     Epoch 36/40
     192/192 [==============================] - 2s 8ms/step - loss: 4.6137e-04 - accuracy: 0.9999 - val_loss: 2.1826 - val_accuracy: 0.8174
     Epoch 37/40
     192/192 [==============================] - 2s 9ms/step - loss: 5.0112e-04 - accuracy: 0.9999 - val_loss: 2.2482 - val_accuracy: 0.8184
     Epoch 38/40
     192/192 [==============================] - 2s 10ms/step - loss: 5.2321e-04 - accuracy: 0.9999 - val_loss: 2.1798 - val_accuracy: 0.818
     Epoch 39/40
     192/192 [==============================] - 2s 10ms/step - loss: 5.1808e-04 - accuracy: 0.9999 - val_loss: 2.2539 - val_accuracy: 0.818
     Epoch 40/40
     192/192 [==============================] - 2s 9ms/step - loss: 5.0617e-04 - accuracy: 0.9999 - val_loss: 2.2680 - val_accuracy: 0.8177
```

Performance Metrics And Accuracy

```
ypred = model.predict(xtest)
ypred = np.where(ypred>0.5,1,0)
```

```
     189/189 [==============================] - 1s 3ms/step
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(ytest,ypred)
```

```
     array([[2832,  587],
            [ 513, 2103]])
```

```
from sklearn.metrics import accuracy_score
accuracy_score(ytest,ypred)
```

```
     0.8177299088649544
```

Creating model using Dropout

```
from tensorflow.keras.layers import Dropout
embedding_vector_features = 40
model = Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(Dropout(0.3))
model.add(LSTM(100))
model.add(Dropout(0.3))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
# Finally Training
model.fit(xtrain,ytrain,validation_data=(xtest,ytest),epochs=40,batch_size=64)
```

```
192/192 [==============================] - 2s 11ms/step - loss: 0.0210 - accuracy: 0.9928 - val_loss: 1.0171 - val_accuracy: 0.8287
Epoch 18/40
192/192 [==============================] - 2s 10ms/step - loss: 0.0145 - accuracy: 0.9959 - val_loss: 1.1287 - val_accuracy: 0.8293
Epoch 19/40
192/192 [==============================] - 2s 9ms/step - loss: 0.0176 - accuracy: 0.9937 - val_loss: 1.0220 - val_accuracy: 0.8255
Epoch 20/40
192/192 [==============================] - 2s 9ms/step - loss: 0.0168 - accuracy: 0.9940 - val_loss: 1.0410 - val_accuracy: 0.8257
Epoch 21/40
192/192 [==============================] - 2s 10ms/step - loss: 0.0171 - accuracy: 0.9944 - val_loss: 1.0114 - val_accuracy: 0.8295
Epoch 22/40
192/192 [==============================] - 2s 12ms/step - loss: 0.0104 - accuracy: 0.9968 - val_loss: 1.1488 - val_accuracy: 0.8247
Epoch 23/40
192/192 [==============================] - 3s 16ms/step - loss: 0.0123 - accuracy: 0.9960 - val_loss: 1.1200 - val_accuracy: 0.8287
Epoch 24/40
192/192 [==============================] - 2s 10ms/step - loss: 0.0133 - accuracy: 0.9956 - val_loss: 1.1589 - val_accuracy: 0.8292
Epoch 25/40
192/192 [==============================] - 2s 9ms/step - loss: 0.0128 - accuracy: 0.9957 - val_loss: 1.0667 - val_accuracy: 0.8323
Epoch 26/40
192/192 [==============================] - 2s 11ms/step - loss: 0.0153 - accuracy: 0.9951 - val_loss: 1.1451 - val_accuracy: 0.8268
Epoch 27/40
192/192 [==============================] - 2s 8ms/step - loss: 0.0103 - accuracy: 0.9972 - val_loss: 1.1031 - val_accuracy: 0.8330
Epoch 28/40
192/192 [==============================] - 2s 9ms/step - loss: 0.0098 - accuracy: 0.9967 - val_loss: 1.1441 - val_accuracy: 0.8298
Epoch 29/40
192/192 [==============================] - 3s 13ms/step - loss: 0.0092 - accuracy: 0.9964 - val_loss: 1.3472 - val_accuracy: 0.8245
Epoch 30/40
192/192 [==============================] - 3s 14ms/step - loss: 0.0094 - accuracy: 0.9964 - val_loss: 1.2192 - val_accuracy: 0.8321
Epoch 31/40
192/192 [==============================] - 2s 9ms/step - loss: 0.0089 - accuracy: 0.9972 - val_loss: 1.0868 - val_accuracy: 0.8297
Epoch 32/40
192/192 [==============================] - 2s 8ms/step - loss: 0.0128 - accuracy: 0.9959 - val_loss: 1.1479 - val_accuracy: 0.8313
Epoch 33/40
192/192 [==============================] - 2s 9ms/step - loss: 0.0099 - accuracy: 0.9966 - val_loss: 1.1582 - val_accuracy: 0.8295
Epoch 34/40
192/192 [==============================] - 2s 9ms/step - loss: 0.0094 - accuracy: 0.9974 - val_loss: 1.2831 - val_accuracy: 0.8272
Epoch 35/40
192/192 [==============================] - 2s 9ms/step - loss: 0.0070 - accuracy: 0.9971 - val_loss: 1.2319 - val_accuracy: 0.8230
Epoch 36/40
192/192 [==============================] - 2s 10ms/step - loss: 0.0074 - accuracy: 0.9977 - val_loss: 1.1732 - val_accuracy: 0.8305
Epoch 37/40
192/192 [==============================] - 3s 15ms/step - loss: 0.0091 - accuracy: 0.9967 - val_loss: 1.3404 - val_accuracy: 0.8254
Epoch 38/40
192/192 [==============================] - 2s 11ms/step - loss: 0.0071 - accuracy: 0.9979 - val_loss: 1.3458 - val_accuracy: 0.8267
Epoch 39/40
192/192 [==============================] - 2s 9ms/step - loss: 0.0082 - accuracy: 0.9973 - val_loss: 1.2115 - val_accuracy: 0.8267
Epoch 40/40
192/192 [==============================] - 2s 10ms/step - loss: 0.0066 - accuracy: 0.9983 - val_loss: 1.3034 - val_accuracy: 0.8244
<keras.callbacks.History at 0x7d0d8a994400>
```

Performance Metrics And Accuracy

```
ypred = model.predict(xtest)
ypred = np.where(ypred>0.5,1,0)
```

```
    189/189 [==============================] - 1s 3ms/step
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(ytest,ypred)
```

```
    array([[2759,  660],
           [ 400, 2216]])
```

```
from sklearn.metrics import accuracy_score
accuracy_score(ytest,ypred)
```

```
    0.824357912178956
```

✓ 0s completed at 7:02 PM ● ×