I081_Aniruddh_Kulkarni_NLP_Exp6i

May 28, 2023

1 Name: Aniruddh Kulkarni

2 Roll no: I081

3 Stream: CS (AI)

4 Division: I

5 Semester: 5th Semester

6 Batch: I-3

7 Subject: NLP

8 Assignment-6

[2]:
```
import warnings
warnings.filterwarnings('ignore')

# Generate and plot a synthetic imbalanced classification dataset
from collections import Counter
import numpy as np
import pandas as pd # to work with csv files

# matplotlib imports are used to plot confusion matrices for the classifiers
import matplotlib as mpl
import matplotlib.cm as cm
import matplotlib.pyplot as plt

# import feature extraction methods from sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction import _stop_words
from sklearn.feature_extraction.text import TfidfVectorizer

# pre-processing of text
import string
```

```python
import re

# import classifiers from sklearn
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

# import different metrics to evaluate the classifiers
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score

# from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn import metrics

# import time function from time module to track the training duration
from time import time

# importing required ml model libraries
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from keras.layers import Dense, Input, Flatten
from keras.layers import GlobalAveragePooling1D, Embedding
from keras.models import Sequential
from keras.preprocessing.text import Tokenizer

from sklearn import metrics

%pip install imbalanced-learn
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler
from imblearn.over_sampling import SMOTE
```

```
Collecting imbalanced-learn
  Using cached imbalanced_learn-0.10.1-py3-none-any.whl (226 kB)
Requirement already satisfied: numpy>=1.17.3 in
/Users/pushpakulkarni/miniconda3/envs/tensorflow/lib/python3.10/site-packages
(from imbalanced-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in
/Users/pushpakulkarni/miniconda3/envs/tensorflow/lib/python3.10/site-packages
(from imbalanced-learn) (1.10.1)
Requirement already satisfied: scikit-learn>=1.0.2 in
```

```
/Users/pushpakulkarni/miniconda3/envs/tensorflow/lib/python3.10/site-packages
(from imbalanced-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in
/Users/pushpakulkarni/miniconda3/envs/tensorflow/lib/python3.10/site-packages
(from imbalanced-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Users/pushpakulkarni/miniconda3/envs/tensorflow/lib/python3.10/site-packages
(from imbalanced-learn) (3.1.0)
Installing collected packages: imbalanced-learn
Successfully installed imbalanced-learn-0.10.1
Note: you may need to restart the kernel to use updated packages.
```

[4]:

```
our_data = pd.read_csv("Full-Economic-News-DFE-839861.csv" , encoding =␣
↪"ISO-8859-1" )
```

[5]:

```
our_data.head()
```

[5]:

```
     _unit_id  _golden _unit_state  _trusted_judgments   _last_judgment_at
0  842613455    False   finalized                    3  12/5/2015 17:48:27  \
1  842613456    False   finalized                    3  12/5/2015 16:54:25
2  842613457    False   finalized                    3  12/5/2015 01:59:03
3  842613458    False   finalized                    3  12/5/2015 02:19:39
4  842613459    False   finalized                    3  12/5/2015 17:48:27

   positivity  positivity:confidence relevance  relevance:confidence
0         3.0                 0.6400       yes                 0.640  \
1         NaN                    NaN        no                 1.000
2         NaN                    NaN        no                 1.000
3         NaN                 0.0000        no                 0.675
4         3.0                 0.3257       yes                 0.640

       articleid        date
0  wsj_398217788  1991-08-14  \
1  wsj_399019502  2007-08-21
2  wsj_398284048  1991-11-14
3  wsj_397959018  1986-06-16
4  wsj_398838054  2002-10-04

                                            headline  positivity_gold
0              Yields on CDs Fell in the Latest Week              NaN  \
1  The Morning Brief: White House Seeks to Limit …              NaN
2  Banking Bill Negotiators Set Compromise --- Pl…              NaN
3  Manager's Journal: Sniffing Out Drug Abusers I…              NaN
4  Currency Trading: Dollar Remains in Tight Rang…              NaN

   relevance_gold                                               text
0             NaN  NEW YORK -- Yields on most certificates of dep…
```

```
1             NaN  The Wall Street Journal Online</br></br>The Mo…
2             NaN  WASHINGTON -- In an effort to achieve banking …
3             NaN  The statistics on the enormous costs of employ…
4             NaN  NEW YORK -- Indecision marked the dollar's ton…
```

[6]:
```
our_data.shape # Number of rows (instances) and columns in the dataset
```

[6]:
```
(8000, 15)
```

[7]:
```
our_data["relevance"].unique()
```

[7]:
```
array(['yes', 'no', 'not sure'], dtype=object)
```

[8]:
```
our_data["relevance"].value_counts()
```

[8]:
```
relevance
no          6571
yes         1420
not sure       9
Name: count, dtype: int64
```

[9]:
```
our_data["relevance"].value_counts()/our_data.shape[0] # Class distribution in␣
↪the dataset
```

[9]:
```
relevance
no          0.821375
yes         0.177500
not sure    0.001125
Name: count, dtype: float64
```

[10]:
```
# convert label to a numerical variable
our_data = our_data[our_data.relevance != "not sure"] # removing the data where␣
↪we don't want relevance="not sure".
our_data.shape
```

[10]:
```
(7991, 15)
```

[11]:
```
our_data['relevance'] = our_data.relevance.map({'yes':1, 'no':0}) # relevant is␣
↪1, not-relevant is 0.
```

[12]:
```
our_data = our_data[["text","relevance"]] # Let us take only the two columns we␣
↪need.
our_data
```

[12]:
```
                                                text  relevance
0     NEW YORK -- Yields on most certificates of dep…          1
1     The Wall Street Journal Online</br></br>The Mo…          0
```

```
2     WASHINGTON -- In an effort to achieve banking …          0
3     The statistics on the enormous costs of employ…          0
4     NEW YORK -- Indecision marked the dollar's ton…          1
…                                                  …        …
7995  Secretary of Commerce Charles W. Sawyer said y…          1
7996  U.S. stocks inched up last week, overcoming co…          0
7997  Ben S. Bernanke cleared a key hurdle Thursday …          0
7998  The White House's push to contract out many fe…          0
7999  NEW YORK. April 17-Automobile stocks put on th…          0

[7991 rows x 2 columns]
```

[13]:
```
our_data.shape
```

[13]:
```
(7991, 2)
```

[14]:
```
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords = stopwords.words('english')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/pushpakulkarni/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[15]:
```
def clean(doc): # doc is a string of text
    doc = doc.replace("</br>", " ") # This text contains a lot of <br/> tags.
    doc = "".join([char for char in doc if char not in string.punctuation and␣
 ↪not char.isdigit()])
    doc = " ".join([token for token in doc.split() if token not in stopwords])
    # remove punctuation and numbers
    return doc
our_data['text'] = our_data['text'].apply(clean)
```

[16]:
```
def special_char(text):
  reviews = ''
  for x in text:
    if x.isalnum():
      reviews = reviews + x
    else:
      reviews = reviews + ' '
  return reviews
our_data['text'] = our_data['text'].apply(special_char)
```

[17]:
```
def convert_lower(text):
   return text.lower()
```

```python
our_data['text'] = our_data['text'].apply(convert_lower)
our_data['text'][1]
```

[17]:
```
'the wall street journal online the morning brief look days biggest news emailed
subscribers every business day sign email on friday evening congress town summer
recess americans heading midaugust weekend bush administration sent message
states the federal government make tougher national childrens insurance program
cover offspring middleincome families the state childrens health insurance
program created help children whose families couldnt afford insurance didnt
qualify medicaid administration officials tell new york times changes aimed
returning program low income focus assuring didnt become replacement private
insurance administration point man dennis smith wrote state officials saying
would new restrictions district columbia states including california new york
extend plan extend coverage children whose families make federal poverty levels
for family three family four under new limits child family making would spend
one year uninsured qualifying state wants extend coverage would assure
washington least children eligible schip medicaid enrolled one programs but
associated press reports state currently make assurances rachel klein deputy
director health policy advocacy group families usa tells ap since many families
threshold cant afford private insurance effect policy uninsured kids ann
clemency kohler deputy commissioner human services new jersey tells times
changes cause havoc program could jeopardize coverage thousands children states
already imposing waiting periods taking steps prevent parents moving children
private insurance schip currently serves million children washington post notes
the administrations new restrictions come program expires end next month
congress doesnt reauthorize subject larger political fight pits white house
democrats republicans congress state capitals'
```

[18]:
```python
x = our_data['text']
y = our_data['relevance']
our_data
```

[18]:

	text	relevance
0	new york yields certificates deposit offered m…	1
1	the wall street journal online the morning bri…	0
2	washington in effort achieve banking reform se…	0
3	the statistics enormous costs employee drug ab…	0
4	new york indecision marked dollars tone trader…	1
…	…	…
7995	secretary commerce charles w sawyer said yeste…	1
7996	us stocks inched last week overcoming concern …	0
7997	ben s bernanke cleared key hurdle thursday con…	0
7998	the white houses push contract many federal fu…	0
7999	new york april automobile stocks put best show…	0

[7991 rows x 2 columns]

[19]:
```
#BOW 1000 max feat
x1 = np.array(our_data.iloc[:,0].values)
y1 = np.array(our_data.relevance.values)
cv = CountVectorizer(max_features = 1000)
x1 = cv.fit_transform(our_data.text).toarray()
print("X.shape = ",x1.shape)
print("y.shape = ",y1.shape)
```

```
X.shape =  (7991, 1000)
y.shape =  (7991,)
```

[20]:
```
#BOW 5000 max feat
x2 = np.array(our_data.iloc[:,0].values)
y2 = np.array(our_data.relevance.values)
cv2 = CountVectorizer(max_features = 5000)
x2 = cv2.fit_transform(our_data.text).toarray()
print("X.shape = ",x2.shape)
print("y.shape = ",y2.shape)
```

```
X.shape =  (7991, 5000)
y.shape =  (7991,)
```

[21]:
```
#Bag of N Grams 1000 feat
x3 = np.array(our_data.iloc[:,0].values)
y3 = np.array(our_data.relevance.values)
count_vect = CountVectorizer(ngram_range=(2,3),max_features = 1000)

x3 = count_vect.fit_transform(our_data.text).toarray()

print("X.shape = ",x3.shape)
print("y.shape = ",y3.shape)
```

```
X.shape =  (7991, 1000)
y.shape =  (7991,)
```

[22]:
```
#Bag of N Grams 5000 feat
x4 = np.array(our_data.iloc[:,0].values)
y4 = np.array(our_data.relevance.values)
count_vect2 = CountVectorizer(ngram_range=(2,3),max_features = 5000)

x4 = count_vect2.fit_transform(our_data.text).toarray()

print("X.shape = ",x4.shape)
print("y.shape = ",y4.shape)
```

```
X.shape =  (7991, 5000)
y.shape =  (7991,)
```

[23]:
```
#TF-IDF 1000 feat
x5 = np.array(our_data.iloc[:,0].values)
y5 = np.array(our_data.relevance.values)

tfidf = TfidfVectorizer(max_features = 1000)
x5 = tfidf.fit_transform(our_data.text).toarray()

print("X.shape = ",x5.shape)
print("y.shape = ",y5.shape)
```

```
X.shape =  (7991, 1000)
y.shape =  (7991,)
```

[24]:
```
#TF-IDF 5000 feat
x6 = np.array(our_data.iloc[:,0].values)
y6 = np.array(our_data.relevance.values)

tfidf2 = TfidfVectorizer(max_features = 5000)
x6 = tfidf2.fit_transform(our_data.text).toarray()

print("X.shape = ",x6.shape)
print("y.shape = ",y6.shape)
```

```
X.shape =  (7991, 5000)
y.shape =  (7991,)
```

[25]:
```
#BoW 1000 feat
x_train1, x_test1, y_train1, y_test1 = train_test_split(x1, y1, test_size = 0.
 ↪3, random_state = 0, shuffle = True)
print(len(x_train1))
print(len(x_test1))

#BoW 5000 feat
x_train2, x_test2, y_train2, y_test2 = train_test_split(x2, y2, test_size = 0.
 ↪3, random_state = 0, shuffle = True)
print(len(x_train2))
print(len(x_test2))


#Bag of n gram 1000 feat
x_train3, x_test3, y_train3, y_test3 = train_test_split(x3, y3, test_size = 0.
 ↪3, random_state = 0, shuffle = True)
print(len(x_train3))
print(len(x_test3))

#Bag of n gram 5000 feat
```

```
x_train4, x_test4, y_train4, y_test4 = train_test_split(x4, y4, test_size = 0.
 ↪3, random_state = 0, shuffle = True)
print(len(x_train4))
print(len(x_test4))



#TF-IDF 1000 feat
x_train5, x_test5, y_train5, y_test5 = train_test_split(x5, y5, test_size = 0.
 ↪3, random_state = 0, shuffle = True)
print(len(x_train5))
print(len(x_test5))


#TF-IDF 5000 feat
x_train6, x_test6, y_train6, y_test6 = train_test_split(x6, y6, test_size = 0.
 ↪3, random_state = 0, shuffle = True)
print(len(x_train6))
print(len(x_test6))
```

```
5593
2398
5593
2398
5593
2398
5593
2398
5593
2398
5593
2398
```

[26]:

```
#NORMAL
X_train, X_test, Y_train, Y_test = train_test_split(x, y, random_state=1)
vect = CountVectorizer(preprocessor=clean)
X_train_dtm = vect.fit_transform(X_train)# use it to extract features from␣
 ↪training data
# transform testing data (using training data's features)
X_test_dtm = vect.transform(X_test)

n_words1 = x_test1.shape[1]
n_words2 = x_test2.shape[1]
n_words3 = x_test3.shape[1]
n_words4 = x_test4.shape[1]
n_words5 = x_test5.shape[1]
n_words6 = x_test6.shape[1]
```

[27]:
```
#create list of model and accuracy dicts
perform_list1 = [ ]
perform_list2 = [ ]
perform_list3 = [ ]
perform_list4 = [ ]
perform_list5 = [ ]
```

[28]:
```
def run_models(x_train, x_test, y_train, y_test, n_words):

  mdl1=''
  mdl2=''
  mdl3=''
  mdl4=''
  mdl5=''

#Multinomial Naive Bayes
  mdl1 = MultinomialNB(alpha=1.0,fit_prior=True)

#Logistic Regression
  mdl2 = LogisticRegression()

#Support Vector Classifer
  mdl3 = SVC()

#Random Forest
  mdl4 = RandomForestClassifier(n_estimators=100 ,criterion='entropy' ,␣
↪random_state=0)

#ANN
  mdl5 = Sequential()
  mdl5.add(Dense(50, input_shape=(n_words,), activation='relu'))
  mdl5.add(Dense(1, activation='sigmoid'))
  mdl5.compile(loss='binary_crossentropy', optimizer='adam',␣
↪metrics=['accuracy'])

#------------------------------------------------------------------------------------
  print()
  print("FOR NAIVE BAYES: ")
  print()
  mdl1.fit(x_train, y_train)
  y_pred = mdl1.predict(x_test)
  # Performance metrics

  accuracy = round(accuracy_score(y_test, y_pred) * 100, 2)

  # Get precision, recall, f1 scores
```

```
precision, recall, f1score, support = score(y_test, y_pred, average='micro')

print('Test Accuracy Score of Basic Naive Bayes Model:',accuracy)

print('Precision :',precision)

print('Recall :',recall)

print('F1-score :',f1score)

#calculate AUC of model
y_pred_prob = mdl1.predict_proba(x_test)[:, 1]
auc1 = metrics.roc_auc_score(y_test, y_pred_prob)
print("ROC_AOC_Score for Naive Bayes: ", auc1)


# Add performance parameters to list

perform_list1.append(dict([('Model', 'Naive Bayes'),
                           ('Test Accuracy', round(accuracy, 2)),('Precision',␣
↪round(precision, 2)),('Recall', round(recall, 2)),('F1', round(f1score,␣
↪2)),('ROC-AUC', round(auc1, 2))]))

#-----------------------------------------------------------------------------------------------
print()
print("FOR LOGISTIC REGRESSION: ")
print()
mdl2.fit(x_train, y_train)
y_pred2 = mdl2.predict(x_test)
# Performance metrics

accuracy2 = round(accuracy_score(y_test, y_pred2) * 100, 2)

# Get precision, recall, f1 scores

precision2, recall2, f1score2, support2 = score(y_test, y_pred2,␣
↪average='micro')

print('Test Accuracy Score of Basic Logistic Regression Model:',accuracy2)

print('Precision :',precision2)

print('Recall :',recall2)

print('F1-score :',f1score2)

#calculate AUC of model
```

```
  y_pred_prob = mdl2.predict_proba(x_test)[:, 1]
  auc2 = metrics.roc_auc_score(y_test, y_pred_prob)
  print("ROC_AOC_Score for Logistic Regression: ", auc2)

  # Add performance parameters to list

  perform_list2.append(dict([('Model', 'Logistic Regression'),
                            ('Test Accuracy', round(accuracy2,␣
↪2)),('Precision', round(precision2, 2)),('Recall', round(recall2, 2)),('F1',␣
↪round(f1score2, 2)),('ROC-AUC', round(auc2, 2))]))


#--------------------------------------------------------------------------------------------

  print()
  print("FOR LINEAR SVC: ")
  print()


  mdl3.fit(x_train, y_train)
  y_pred3 = mdl3.predict(x_test)
  # Performance metrics

  accuracy3 = round(accuracy_score(y_test, y_pred3) * 100, 2)

  # Get precision, recall, f1 scores

  precision3, recall3, f1score3, support3 = score(y_test, y_pred3,␣
↪average='micro')

  print('Test Accuracy Score of Basic Linear SVC Model:',accuracy3)

  print('Precision :',precision3)

  print('Recall :',recall3)

  print('F1-score :',f1score3)

  #calculate AUC of model
  #y_pred_prob = mdl3.predict_proba(x_test)[:, 1]
  auc3 = metrics.roc_auc_score(y_test, y_pred_prob)
  print("ROC_AOC_Score for Linear SVC: ", auc3)

  # Add performance parameters to list

  perform_list3.append(dict([('Model', 'Linear SVC'),
```

```
                              ('Test Accuracy', round(accuracy3,␣
↪2)),('Precision', round(precision3, 2)),('Recall', round(recall3, 2)),('F1',␣
↪round(f1score3, 2)),('ROC-AUC', round(auc3, 2))]))


#-----------------------------------------------------------------------------------------------


  print()
  print("FOR RANDOM FOREST: ")
  print()
  mdl4.fit(x_train, y_train)
  y_pred4 = mdl4.predict(x_test)
  # Performance metrics

  accuracy4 = round(accuracy_score(y_test, y_pred4) * 100, 2)



  # Get precision, recall, f1 scores

  precision4, recall4, f1score4, support4 = score(y_test, y_pred4,␣
↪average='micro')

  print('Test Accuracy Score of Basic Random Forest Model:',accuracy4)

  print('Precision :',precision4)

  print('Recall :',recall4)

  print('F1-score :',f1score4)

  #calculate AUC of model
  y_pred_prob = mdl4.predict_proba(x_test)[:, 1]
  auc4 = metrics.roc_auc_score(y_test, y_pred_prob)
  print("ROC_AOC_Score for Random Forest: ", auc4)



  # Add performance parameters to list

  perform_list4.append(dict([('Model', 'Random Forest'),
                              ('Test Accuracy', round(accuracy4,␣
↪2)),('Precision', round(precision4, 2)),('Recall', round(recall4, 2)),('F1',␣
↪round(f1score4, 2)),('ROC-AUC', round(auc4, 2))]))
```

```
#-----------------------------------------------------------------------------------------------

  print()
  print("FOR ANN: ")
  print()
  mdl5.summary()
  mdl5.fit(x_train, y_train, epochs=50, verbose=2)
  loss, acc = mdl5.evaluate(x_test, y_test, verbose=0)
  #calculate AUC of model
  #y_pred_prob = mdl5.predict_proba(x_test)[:, 1]
  auc5 = metrics.roc_auc_score(y_test, y_pred_prob)
  print("ROC_AOC_Score for ANN: ", auc5)
  print('Test Accuracy:',acc)
  perform_list5.append(dict([('Model', 'ANN'),('Test Accuracy', round(acc,␣
↪2)),('Loss',round(loss,2)),('ROC-AUC', round(auc5, 2))]))
```

[29]:

```
run_models(x_train1, x_test1, y_train1, y_test1, n_words1)
```

```
Metal device set to: Apple M1

FOR NAIVE BAYES:

Test Accuracy Score of Basic Naive Bayes Model: 67.76
Precision : 0.6776480400333611
Recall : 0.6776480400333611
F1-score : 0.6776480400333611
ROC_AOC_Score for Naive Bayes:  0.7237625305086854

FOR LOGISTIC REGRESSION:

Test Accuracy Score of Basic Logistic Regression Model: 77.36
Precision : 0.7735613010842368
Recall : 0.7735613010842368
F1-score : 0.7735613010842367
ROC_AOC_Score for Logistic Regression:  0.6703189880011768

FOR LINEAR SVC:

Test Accuracy Score of Basic Linear SVC Model: 81.61
Precision : 0.8160967472894078
Recall : 0.8160967472894078
F1-score : 0.8160967472894078
ROC_AOC_Score for Linear SVC:  0.6703189880011768

FOR RANDOM FOREST:
```

```
Test Accuracy Score of Basic Random Forest Model: 81.53
Precision : 0.8152627189324437
Recall : 0.8152627189324437
F1-score : 0.8152627189324437
ROC_AOC_Score for Random Forest:  0.7120753347961224
```

FOR ANN:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                50050

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 50,101
Trainable params: 50,101
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50

2023-05-28 00:13:37.947256: W
tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU
frequency: 0 Hz

175/175 - 3s - loss: 0.4458 - accuracy: 0.8183 - 3s/epoch - 20ms/step
Epoch 2/50
175/175 - 1s - loss: 0.3661 - accuracy: 0.8416 - 1s/epoch - 7ms/step
Epoch 3/50
175/175 - 1s - loss: 0.3111 - accuracy: 0.8707 - 1s/epoch - 8ms/step
Epoch 4/50
175/175 - 1s - loss: 0.2478 - accuracy: 0.9056 - 1s/epoch - 8ms/step
Epoch 5/50
175/175 - 1s - loss: 0.1861 - accuracy: 0.9397 - 1s/epoch - 8ms/step
Epoch 6/50
175/175 - 1s - loss: 0.1339 - accuracy: 0.9659 - 1s/epoch - 7ms/step
Epoch 7/50
175/175 - 1s - loss: 0.0908 - accuracy: 0.9837 - 1s/epoch - 7ms/step
Epoch 8/50
175/175 - 1s - loss: 0.0627 - accuracy: 0.9927 - 1s/epoch - 7ms/step
Epoch 9/50
175/175 - 1s - loss: 0.0434 - accuracy: 0.9957 - 1s/epoch - 7ms/step
Epoch 10/50
175/175 - 1s - loss: 0.0310 - accuracy: 0.9968 - 1s/epoch - 7ms/step
Epoch 11/50
175/175 - 1s - loss: 0.0226 - accuracy: 0.9970 - 1s/epoch - 7ms/step
```

```
Epoch 12/50
175/175 - 1s - loss: 0.0178 - accuracy: 0.9970 - 1s/epoch - 7ms/step
Epoch 13/50
175/175 - 1s - loss: 0.0147 - accuracy: 0.9970 - 1s/epoch - 7ms/step
Epoch 14/50
175/175 - 1s - loss: 0.0123 - accuracy: 0.9970 - 1s/epoch - 7ms/step
Epoch 15/50
175/175 - 1s - loss: 0.0106 - accuracy: 0.9973 - 1s/epoch - 7ms/step
Epoch 16/50
175/175 - 1s - loss: 0.0083 - accuracy: 0.9973 - 1s/epoch - 7ms/step
Epoch 17/50
175/175 - 1s - loss: 0.0082 - accuracy: 0.9971 - 1s/epoch - 8ms/step
Epoch 18/50
175/175 - 2s - loss: 0.0070 - accuracy: 0.9975 - 2s/epoch - 9ms/step
Epoch 19/50
175/175 - 1s - loss: 0.0060 - accuracy: 0.9979 - 1s/epoch - 8ms/step
Epoch 20/50
175/175 - 1s - loss: 0.0056 - accuracy: 0.9977 - 1s/epoch - 8ms/step
Epoch 21/50
175/175 - 1s - loss: 0.0057 - accuracy: 0.9979 - 1s/epoch - 7ms/step
Epoch 22/50
175/175 - 1s - loss: 0.0044 - accuracy: 0.9980 - 1s/epoch - 7ms/step
Epoch 23/50
175/175 - 1s - loss: 0.0046 - accuracy: 0.9979 - 1s/epoch - 7ms/step
Epoch 24/50
175/175 - 1s - loss: 0.0043 - accuracy: 0.9977 - 1s/epoch - 7ms/step
Epoch 25/50
175/175 - 1s - loss: 0.0037 - accuracy: 0.9982 - 1s/epoch - 7ms/step
Epoch 26/50
175/175 - 1s - loss: 0.0037 - accuracy: 0.9989 - 1s/epoch - 7ms/step
Epoch 27/50
175/175 - 1s - loss: 0.0036 - accuracy: 0.9986 - 1s/epoch - 7ms/step
Epoch 28/50
175/175 - 1s - loss: 0.0030 - accuracy: 0.9993 - 1s/epoch - 7ms/step
Epoch 29/50
175/175 - 1s - loss: 0.0026 - accuracy: 0.9996 - 1s/epoch - 8ms/step
Epoch 30/50
175/175 - 1s - loss: 0.0025 - accuracy: 0.9996 - 1s/epoch - 7ms/step
Epoch 31/50
175/175 - 1s - loss: 0.0019 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 32/50
175/175 - 1s - loss: 0.0017 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 33/50
175/175 - 1s - loss: 0.0016 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 34/50
175/175 - 1s - loss: 0.0016 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 35/50
175/175 - 1s - loss: 0.0015 - accuracy: 1.0000 - 1s/epoch - 7ms/step
```

```
Epoch 36/50
175/175 - 1s - loss: 0.0014 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 37/50
175/175 - 1s - loss: 0.0014 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 38/50
175/175 - 1s - loss: 0.0013 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 39/50
175/175 - 1s - loss: 0.0013 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 40/50
175/175 - 1s - loss: 0.0012 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 41/50
175/175 - 1s - loss: 0.0021 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 42/50
175/175 - 1s - loss: 0.0027 - accuracy: 0.9998 - 1s/epoch - 7ms/step
Epoch 43/50
175/175 - 1s - loss: 0.0013 - accuracy: 0.9998 - 1s/epoch - 7ms/step
Epoch 44/50
175/175 - 1s - loss: 0.0011 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 45/50
175/175 - 1s - loss: 0.0010 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 46/50
175/175 - 1s - loss: 9.9447e-04 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 47/50
175/175 - 1s - loss: 9.5039e-04 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 48/50
175/175 - 1s - loss: 9.1217e-04 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 49/50
175/175 - 1s - loss: 8.7074e-04 - accuracy: 1.0000 - 1s/epoch - 7ms/step
Epoch 50/50
175/175 - 1s - loss: 8.5015e-04 - accuracy: 1.0000 - 1s/epoch - 7ms/step
ROC_AOC_Score for ANN:  0.7120753347961224
Test Accuracy: 0.7814845442771912
```

[30]:
```
run_models(x_train2, x_test2, y_train2, y_test2, n_words2)
```

```
FOR NAIVE BAYES:

Test Accuracy Score of Basic Naive Bayes Model: 67.26
Precision : 0.6726438698915763
Recall : 0.6726438698915763
F1-score : 0.6726438698915763
ROC_AOC_Score for Naive Bayes:  0.732863682716648

FOR LOGISTIC REGRESSION:

Test Accuracy Score of Basic Logistic Regression Model: 76.36
Precision : 0.7635529608006673
```

```
Recall : 0.7635529608006673
F1-score : 0.7635529608006673
ROC_AOC_Score for Logistic Regression:  0.6642259718302652

FOR LINEAR SVC:

Test Accuracy Score of Basic Linear SVC Model: 81.65
Precision : 0.8165137614678899
Recall : 0.8165137614678899
F1-score : 0.81651376146789
ROC_AOC_Score for Linear SVC:  0.6642259718302652

FOR RANDOM FOREST:

Test Accuracy Score of Basic Random Forest Model: 81.61
Precision : 0.8160967472894078
Recall : 0.8160967472894078
F1-score : 0.8160967472894078
ROC_AOC_Score for Random Forest:  0.7122764973529101

FOR ANN:

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 50)                250050

 dense_3 (Dense)             (None, 1)                 51

=================================================================
Total params: 250,101
Trainable params: 250,101
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
175/175 - 2s - loss: 0.4286 - accuracy: 0.8216 - 2s/epoch - 13ms/step
Epoch 2/50
175/175 - 1s - loss: 0.3048 - accuracy: 0.8570 - 1s/epoch - 8ms/step
Epoch 3/50
175/175 - 2s - loss: 0.1978 - accuracy: 0.9203 - 2s/epoch - 9ms/step
Epoch 4/50
175/175 - 2s - loss: 0.1042 - accuracy: 0.9750 - 2s/epoch - 9ms/step
Epoch 5/50
175/175 - 2s - loss: 0.0487 - accuracy: 0.9936 - 2s/epoch - 9ms/step
Epoch 6/50
175/175 - 1s - loss: 0.0232 - accuracy: 0.9971 - 1s/epoch - 8ms/step
Epoch 7/50
```

```
175/175 - 1s - loss: 0.0139 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 8/50
175/175 - 1s - loss: 0.0094 - accuracy: 0.9984 - 1s/epoch - 8ms/step
Epoch 9/50
175/175 - 1s - loss: 0.0072 - accuracy: 0.9986 - 1s/epoch - 8ms/step
Epoch 10/50
175/175 - 2s - loss: 0.0057 - accuracy: 0.9989 - 2s/epoch - 9ms/step
Epoch 11/50
175/175 - 1s - loss: 0.0043 - accuracy: 0.9991 - 1s/epoch - 8ms/step
Epoch 12/50
175/175 - 1s - loss: 0.0045 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 13/50
175/175 - 1s - loss: 0.0039 - accuracy: 0.9991 - 1s/epoch - 8ms/step
Epoch 14/50
175/175 - 1s - loss: 0.0029 - accuracy: 0.9993 - 1s/epoch - 8ms/step
Epoch 15/50
175/175 - 2s - loss: 0.0020 - accuracy: 0.9995 - 2s/epoch - 9ms/step
Epoch 16/50
175/175 - 1s - loss: 0.0016 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 17/50
175/175 - 1s - loss: 0.0015 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 18/50
175/175 - 1s - loss: 0.0012 - accuracy: 1.0000 - 1s/epoch - 9ms/step
Epoch 19/50
175/175 - 1s - loss: 0.0010 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 20/50
175/175 - 1s - loss: 0.0011 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 21/50
175/175 - 1s - loss: 0.0015 - accuracy: 0.9995 - 1s/epoch - 8ms/step
Epoch 22/50
175/175 - 1s - loss: 8.4189e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 23/50
175/175 - 1s - loss: 7.0274e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 24/50
175/175 - 1s - loss: 6.4155e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 25/50
175/175 - 2s - loss: 6.1541e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 26/50
175/175 - 2s - loss: 5.8428e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 27/50
175/175 - 1s - loss: 5.4582e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 28/50
175/175 - 1s - loss: 5.2623e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 29/50
175/175 - 1s - loss: 5.0085e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 30/50
175/175 - 1s - loss: 4.7653e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 31/50
```

```
175/175 - 1s - loss: 4.6845e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 32/50
175/175 - 1s - loss: 4.4904e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 33/50
175/175 - 1s - loss: 4.3328e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 34/50
175/175 - 2s - loss: 4.2661e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 35/50
175/175 - 2s - loss: 4.6435e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 36/50
175/175 - 1s - loss: 0.0018 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 37/50
175/175 - 2s - loss: 0.0016 - accuracy: 0.9996 - 2s/epoch - 9ms/step
Epoch 38/50
175/175 - 2s - loss: 7.0251e-04 - accuracy: 0.9998 - 2s/epoch - 9ms/step
Epoch 39/50
175/175 - 1s - loss: 4.1090e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 40/50
175/175 - 1s - loss: 3.9432e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 41/50
175/175 - 1s - loss: 3.8360e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 42/50
175/175 - 1s - loss: 3.8224e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 43/50
175/175 - 1s - loss: 3.6629e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 44/50
175/175 - 1s - loss: 3.5805e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 45/50
175/175 - 1s - loss: 3.6010e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 46/50
175/175 - 1s - loss: 3.3603e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 47/50
175/175 - 1s - loss: 3.3450e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 48/50
175/175 - 2s - loss: 3.2842e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 49/50
175/175 - 2s - loss: 3.1774e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 50/50
175/175 - 1s - loss: 3.1439e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
ROC_AOC_Score for ANN:  0.7122764973529101
Test Accuracy: 0.7789824604988098
```

[31]:

```
run_models(x_train3, x_test3, y_train3, y_test3, n_words3)
```

```
FOR NAIVE BAYES:

Test Accuracy Score of Basic Naive Bayes Model: 76.65
```

```
Precision : 0.7664720600500416
Recall : 0.7664720600500416
F1-score : 0.7664720600500416
ROC_AOC_Score for Naive Bayes:  0.6779253745053785

FOR LOGISTIC REGRESSION:

Test Accuracy Score of Basic Logistic Regression Model: 79.36
Precision : 0.7935779816513762
Recall : 0.7935779816513762
F1-score : 0.7935779816513762
ROC_AOC_Score for Logistic Regression:  0.652991682567811

FOR LINEAR SVC:

Test Accuracy Score of Basic Linear SVC Model: 81.9
Precision : 0.8190158465387823
Recall : 0.8190158465387823
F1-score : 0.8190158465387823
ROC_AOC_Score for Linear SVC:  0.652991682567811

FOR RANDOM FOREST:

Test Accuracy Score of Basic Random Forest Model: 81.69
Precision : 0.816930775646372
Recall : 0.816930775646372
F1-score : 0.816930775646372
ROC_AOC_Score for Random Forest:  0.7088642920182651

FOR ANN:

Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 50)                50050

 dense_5 (Dense)             (None, 1)                 51

=================================================================
Total params: 50,101
Trainable params: 50,101
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
175/175 - 2s - loss: 0.5454 - accuracy: 0.7733 - 2s/epoch - 10ms/step
Epoch 2/50
175/175 - 1s - loss: 0.4212 - accuracy: 0.8307 - 1s/epoch - 8ms/step
```

```
Epoch 3/50
175/175 - 1s - loss: 0.3727 - accuracy: 0.8464 - 1s/epoch - 7ms/step
Epoch 4/50
175/175 - 1s - loss: 0.3328 - accuracy: 0.8663 - 1s/epoch - 8ms/step
Epoch 5/50
175/175 - 1s - loss: 0.2924 - accuracy: 0.8909 - 1s/epoch - 8ms/step
Epoch 6/50
175/175 - 1s - loss: 0.2539 - accuracy: 0.9094 - 1s/epoch - 8ms/step
Epoch 7/50
175/175 - 1s - loss: 0.2173 - accuracy: 0.9299 - 1s/epoch - 8ms/step
Epoch 8/50
175/175 - 1s - loss: 0.1844 - accuracy: 0.9451 - 1s/epoch - 7ms/step
Epoch 9/50
175/175 - 1s - loss: 0.1558 - accuracy: 0.9567 - 1s/epoch - 7ms/step
Epoch 10/50
175/175 - 1s - loss: 0.1314 - accuracy: 0.9667 - 1s/epoch - 7ms/step
Epoch 11/50
175/175 - 1s - loss: 0.1104 - accuracy: 0.9751 - 1s/epoch - 7ms/step
Epoch 12/50
175/175 - 1s - loss: 0.0929 - accuracy: 0.9798 - 1s/epoch - 8ms/step
Epoch 13/50
175/175 - 1s - loss: 0.0791 - accuracy: 0.9836 - 1s/epoch - 7ms/step
Epoch 14/50
175/175 - 1s - loss: 0.0671 - accuracy: 0.9864 - 1s/epoch - 7ms/step
Epoch 15/50
175/175 - 1s - loss: 0.0575 - accuracy: 0.9898 - 1s/epoch - 7ms/step
Epoch 16/50
175/175 - 1s - loss: 0.0500 - accuracy: 0.9918 - 1s/epoch - 7ms/step
Epoch 17/50
175/175 - 1s - loss: 0.0431 - accuracy: 0.9932 - 1s/epoch - 8ms/step
Epoch 18/50
175/175 - 1s - loss: 0.0374 - accuracy: 0.9936 - 1s/epoch - 8ms/step
Epoch 19/50
175/175 - 1s - loss: 0.0329 - accuracy: 0.9945 - 1s/epoch - 7ms/step
Epoch 20/50
175/175 - 1s - loss: 0.0288 - accuracy: 0.9957 - 1s/epoch - 7ms/step
Epoch 21/50
175/175 - 1s - loss: 0.0254 - accuracy: 0.9964 - 1s/epoch - 7ms/step
Epoch 22/50
175/175 - 1s - loss: 0.0227 - accuracy: 0.9966 - 1s/epoch - 7ms/step
Epoch 23/50
175/175 - 1s - loss: 0.0202 - accuracy: 0.9971 - 1s/epoch - 7ms/step
Epoch 24/50
175/175 - 1s - loss: 0.0182 - accuracy: 0.9971 - 1s/epoch - 7ms/step
Epoch 25/50
175/175 - 1s - loss: 0.0165 - accuracy: 0.9975 - 1s/epoch - 7ms/step
Epoch 26/50
175/175 - 1s - loss: 0.0149 - accuracy: 0.9979 - 1s/epoch - 7ms/step
```

```
Epoch 27/50
175/175 - 1s - loss: 0.0136 - accuracy: 0.9979 - 1s/epoch - 7ms/step
Epoch 28/50
175/175 - 1s - loss: 0.0124 - accuracy: 0.9979 - 1s/epoch - 7ms/step
Epoch 29/50
175/175 - 1s - loss: 0.0114 - accuracy: 0.9980 - 1s/epoch - 8ms/step
Epoch 30/50
175/175 - 1s - loss: 0.0103 - accuracy: 0.9984 - 1s/epoch - 8ms/step
Epoch 31/50
175/175 - 1s - loss: 0.0095 - accuracy: 0.9984 - 1s/epoch - 7ms/step
Epoch 32/50
175/175 - 1s - loss: 0.0088 - accuracy: 0.9984 - 1s/epoch - 7ms/step
Epoch 33/50
175/175 - 1s - loss: 0.0082 - accuracy: 0.9986 - 1s/epoch - 7ms/step
Epoch 34/50
175/175 - 1s - loss: 0.0076 - accuracy: 0.9986 - 1s/epoch - 7ms/step
Epoch 35/50
175/175 - 1s - loss: 0.0071 - accuracy: 0.9987 - 1s/epoch - 7ms/step
Epoch 36/50
175/175 - 1s - loss: 0.0067 - accuracy: 0.9987 - 1s/epoch - 7ms/step
Epoch 37/50
175/175 - 1s - loss: 0.0062 - accuracy: 0.9987 - 1s/epoch - 7ms/step
Epoch 38/50
175/175 - 1s - loss: 0.0059 - accuracy: 0.9987 - 1s/epoch - 7ms/step
Epoch 39/50
175/175 - 1s - loss: 0.0055 - accuracy: 0.9986 - 1s/epoch - 7ms/step
Epoch 40/50
175/175 - 1s - loss: 0.0053 - accuracy: 0.9987 - 1s/epoch - 7ms/step
Epoch 41/50
175/175 - 1s - loss: 0.0050 - accuracy: 0.9989 - 1s/epoch - 7ms/step
Epoch 42/50
175/175 - 1s - loss: 0.0047 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 43/50
175/175 - 1s - loss: 0.0045 - accuracy: 0.9989 - 1s/epoch - 7ms/step
Epoch 44/50
175/175 - 1s - loss: 0.0043 - accuracy: 0.9987 - 1s/epoch - 7ms/step
Epoch 45/50
175/175 - 1s - loss: 0.0047 - accuracy: 0.9986 - 1s/epoch - 7ms/step
Epoch 46/50
175/175 - 1s - loss: 0.0043 - accuracy: 0.9987 - 1s/epoch - 7ms/step
Epoch 47/50
175/175 - 1s - loss: 0.0037 - accuracy: 0.9989 - 1s/epoch - 7ms/step
Epoch 48/50
175/175 - 1s - loss: 0.0036 - accuracy: 0.9989 - 1s/epoch - 7ms/step
Epoch 49/50
175/175 - 1s - loss: 0.0035 - accuracy: 0.9987 - 1s/epoch - 7ms/step
Epoch 50/50
175/175 - 1s - loss: 0.0034 - accuracy: 0.9987 - 1s/epoch - 7ms/step
```

```
ROC_AOC_Score for ANN:  0.7088642920182651
Test Accuracy: 0.7806505560874939
```

[32]:
```
run_models(x_train4, x_test4, y_train4, y_test4, n_words4)
```

```
FOR NAIVE BAYES:

Test Accuracy Score of Basic Naive Bayes Model: 75.65
Precision : 0.7564637197664721
Recall : 0.7564637197664721
F1-score : 0.7564637197664721
ROC_AOC_Score for Naive Bayes:  0.6900736161934696

FOR LOGISTIC REGRESSION:

Test Accuracy Score of Basic Logistic Regression Model: 78.15
Precision : 0.7814845704753962
Recall : 0.7814845704753962
F1-score : 0.7814845704753962
ROC_AOC_Score for Logistic Regression:  0.641203905576854

FOR LINEAR SVC:

Test Accuracy Score of Basic Linear SVC Model: 81.65
Precision : 0.8165137614678899
Recall : 0.8165137614678899
F1-score : 0.81651376146789
ROC_AOC_Score for Linear SVC:  0.641203905576854

FOR RANDOM FOREST:

Test Accuracy Score of Basic Random Forest Model: 81.44
Precision : 0.8144286905754796
Recall : 0.8144286905754796
F1-score : 0.8144286905754796
ROC_AOC_Score for Random Forest:  0.6943974483750601

FOR ANN:

Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 50)                250050

 dense_7 (Dense)             (None, 1)                 51
```

```
=================================================================
Total params: 250,101
Trainable params: 250,101
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
175/175 - 2s - loss: 0.4924 - accuracy: 0.8178 - 2s/epoch - 13ms/step
Epoch 2/50
175/175 - 1s - loss: 0.3374 - accuracy: 0.8561 - 1s/epoch - 8ms/step
Epoch 3/50
175/175 - 1s - loss: 0.2306 - accuracy: 0.9097 - 1s/epoch - 8ms/step
Epoch 4/50
175/175 - 1s - loss: 0.1364 - accuracy: 0.9596 - 1s/epoch - 8ms/step
Epoch 5/50
175/175 - 1s - loss: 0.0761 - accuracy: 0.9828 - 1s/epoch - 8ms/step
Epoch 6/50
175/175 - 1s - loss: 0.0428 - accuracy: 0.9945 - 1s/epoch - 8ms/step
Epoch 7/50
175/175 - 1s - loss: 0.0266 - accuracy: 0.9977 - 1s/epoch - 8ms/step
Epoch 8/50
175/175 - 1s - loss: 0.0184 - accuracy: 0.9980 - 1s/epoch - 8ms/step
Epoch 9/50
175/175 - 1s - loss: 0.0136 - accuracy: 0.9982 - 1s/epoch - 8ms/step
Epoch 10/50
175/175 - 2s - loss: 0.0104 - accuracy: 0.9982 - 2s/epoch - 9ms/step
Epoch 11/50
175/175 - 1s - loss: 0.0082 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 12/50
175/175 - 1s - loss: 0.0069 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 13/50
175/175 - 1s - loss: 0.0061 - accuracy: 0.9986 - 1s/epoch - 8ms/step
Epoch 14/50
175/175 - 1s - loss: 0.0051 - accuracy: 0.9986 - 1s/epoch - 8ms/step
Epoch 15/50
175/175 - 1s - loss: 0.0040 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 16/50
175/175 - 1s - loss: 0.0035 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 17/50
175/175 - 1s - loss: 0.0031 - accuracy: 0.9989 - 1s/epoch - 8ms/step
Epoch 18/50
175/175 - 1s - loss: 0.0027 - accuracy: 0.9989 - 1s/epoch - 8ms/step
Epoch 19/50
175/175 - 1s - loss: 0.0026 - accuracy: 0.9991 - 1s/epoch - 8ms/step
Epoch 20/50
175/175 - 1s - loss: 0.0022 - accuracy: 0.9991 - 1s/epoch - 8ms/step
Epoch 21/50
175/175 - 1s - loss: 0.0018 - accuracy: 0.9995 - 1s/epoch - 8ms/step
Epoch 22/50
```

```
175/175 - 1s - loss: 0.0016 - accuracy: 0.9995 - 1s/epoch - 8ms/step
Epoch 23/50
175/175 - 1s - loss: 0.0015 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 24/50
175/175 - 2s - loss: 0.0013 - accuracy: 0.9998 - 2s/epoch - 9ms/step
Epoch 25/50
175/175 - 1s - loss: 0.0012 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 26/50
175/175 - 1s - loss: 0.0011 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 27/50
175/175 - 1s - loss: 0.0010 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 28/50
175/175 - 1s - loss: 9.7206e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 29/50
175/175 - 1s - loss: 9.4516e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 30/50
175/175 - 2s - loss: 9.2674e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 31/50
175/175 - 2s - loss: 8.5656e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 32/50
175/175 - 2s - loss: 0.0012 - accuracy: 0.9998 - 2s/epoch - 9ms/step
Epoch 33/50
175/175 - 2s - loss: 0.0015 - accuracy: 0.9998 - 2s/epoch - 10ms/step
Epoch 34/50
175/175 - 1s - loss: 7.4323e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 35/50
175/175 - 1s - loss: 7.0301e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 36/50
175/175 - 1s - loss: 6.6723e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 37/50
175/175 - 1s - loss: 6.6146e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 38/50
175/175 - 1s - loss: 6.1080e-04 - accuracy: 1.0000 - 1s/epoch - 9ms/step
Epoch 39/50
175/175 - 1s - loss: 5.8384e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 40/50
175/175 - 1s - loss: 5.6445e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 41/50
175/175 - 1s - loss: 5.6741e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 42/50
175/175 - 2s - loss: 5.3851e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 43/50
175/175 - 2s - loss: 5.1114e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 44/50
175/175 - 1s - loss: 4.7539e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 45/50
175/175 - 1s - loss: 4.5255e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 46/50
```

```
175/175 - 1s - loss: 4.3577e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 47/50
175/175 - 1s - loss: 4.2424e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 48/50
175/175 - 1s - loss: 4.0912e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 49/50
175/175 - 1s - loss: 3.9300e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 50/50
175/175 - 1s - loss: 3.8001e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
ROC_AOC_Score for ANN:  0.6943974483750601
Test Accuracy: 0.7664720416069031
```

[33]:
```
run_models(x_train5, x_test5, y_train5, y_test5, n_words5)
```

```
FOR NAIVE BAYES:

Test Accuracy Score of Basic Naive Bayes Model: 82.19
Precision : 0.8219349457881568
Recall : 0.8219349457881568
F1-score : 0.8219349457881568
ROC_AOC_Score for Naive Bayes:  0.7280538045885993

FOR LOGISTIC REGRESSION:

Test Accuracy Score of Basic Logistic Regression Model: 81.65
Precision : 0.8165137614678899
Recall : 0.8165137614678899
F1-score : 0.81651376146789
ROC_AOC_Score for Logistic Regression:  0.7340770533987752

FOR LINEAR SVC:

Test Accuracy Score of Basic Linear SVC Model: 81.65
Precision : 0.8165137614678899
Recall : 0.8165137614678899
F1-score : 0.81651376146789
ROC_AOC_Score for Linear SVC:  0.7340770533987752

FOR RANDOM FOREST:

Test Accuracy Score of Basic Random Forest Model: 81.78
Precision : 0.8177648040033361
Recall : 0.8177648040033361
F1-score : 0.8177648040033361
ROC_AOC_Score for Random Forest:  0.7247009015105796

FOR ANN:
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 50)                50050

 dense_9 (Dense)             (None, 1)                 51

=================================================================
Total params: 50,101
Trainable params: 50,101
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
175/175 - 2s - loss: 0.4745 - accuracy: 0.8237 - 2s/epoch - 11ms/step
Epoch 2/50
175/175 - 1s - loss: 0.4022 - accuracy: 0.8253 - 1s/epoch - 8ms/step
Epoch 3/50
175/175 - 1s - loss: 0.3799 - accuracy: 0.8303 - 1s/epoch - 8ms/step
Epoch 4/50
175/175 - 1s - loss: 0.3634 - accuracy: 0.8385 - 1s/epoch - 8ms/step
Epoch 5/50
175/175 - 1s - loss: 0.3489 - accuracy: 0.8487 - 1s/epoch - 8ms/step
Epoch 6/50
175/175 - 1s - loss: 0.3313 - accuracy: 0.8595 - 1s/epoch - 8ms/step
Epoch 7/50
175/175 - 1s - loss: 0.3156 - accuracy: 0.8688 - 1s/epoch - 8ms/step
Epoch 8/50
175/175 - 1s - loss: 0.2970 - accuracy: 0.8784 - 1s/epoch - 8ms/step
Epoch 9/50
175/175 - 1s - loss: 0.2795 - accuracy: 0.8917 - 1s/epoch - 8ms/step
Epoch 10/50
175/175 - 1s - loss: 0.2621 - accuracy: 0.9008 - 1s/epoch - 8ms/step
Epoch 11/50
175/175 - 1s - loss: 0.2441 - accuracy: 0.9117 - 1s/epoch - 8ms/step
Epoch 12/50
175/175 - 1s - loss: 0.2272 - accuracy: 0.9190 - 1s/epoch - 8ms/step
Epoch 13/50
175/175 - 1s - loss: 0.2107 - accuracy: 0.9304 - 1s/epoch - 9ms/step
Epoch 14/50
175/175 - 1s - loss: 0.1957 - accuracy: 0.9394 - 1s/epoch - 8ms/step
Epoch 15/50
175/175 - 1s - loss: 0.1789 - accuracy: 0.9455 - 1s/epoch - 8ms/step
Epoch 16/50
175/175 - 1s - loss: 0.1646 - accuracy: 0.9549 - 1s/epoch - 8ms/step
Epoch 17/50
175/175 - 2s - loss: 0.1511 - accuracy: 0.9601 - 2s/epoch - 9ms/step
```

```
Epoch 18/50
175/175 - 2s - loss: 0.1376 - accuracy: 0.9671 - 2s/epoch - 10ms/step
Epoch 19/50
175/175 - 2s - loss: 0.1254 - accuracy: 0.9714 - 2s/epoch - 9ms/step
Epoch 20/50
175/175 - 1s - loss: 0.1141 - accuracy: 0.9784 - 1s/epoch - 8ms/step
Epoch 21/50
175/175 - 1s - loss: 0.1033 - accuracy: 0.9802 - 1s/epoch - 8ms/step
Epoch 22/50
175/175 - 1s - loss: 0.0933 - accuracy: 0.9843 - 1s/epoch - 8ms/step
Epoch 23/50
175/175 - 1s - loss: 0.0841 - accuracy: 0.9880 - 1s/epoch - 7ms/step
Epoch 24/50
175/175 - 1s - loss: 0.0756 - accuracy: 0.9891 - 1s/epoch - 7ms/step
Epoch 25/50
175/175 - 1s - loss: 0.0680 - accuracy: 0.9909 - 1s/epoch - 8ms/step
Epoch 26/50
175/175 - 1s - loss: 0.0612 - accuracy: 0.9934 - 1s/epoch - 7ms/step
Epoch 27/50
175/175 - 1s - loss: 0.0548 - accuracy: 0.9950 - 1s/epoch - 8ms/step
Epoch 28/50
175/175 - 1s - loss: 0.0487 - accuracy: 0.9966 - 1s/epoch - 8ms/step
Epoch 29/50
175/175 - 1s - loss: 0.0434 - accuracy: 0.9964 - 1s/epoch - 8ms/step
Epoch 30/50
175/175 - 1s - loss: 0.0390 - accuracy: 0.9977 - 1s/epoch - 8ms/step
Epoch 31/50
175/175 - 1s - loss: 0.0346 - accuracy: 0.9980 - 1s/epoch - 7ms/step
Epoch 32/50
175/175 - 1s - loss: 0.0313 - accuracy: 0.9984 - 1s/epoch - 7ms/step
Epoch 33/50
175/175 - 1s - loss: 0.0279 - accuracy: 0.9989 - 1s/epoch - 8ms/step
Epoch 34/50
175/175 - 1s - loss: 0.0249 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 35/50
175/175 - 1s - loss: 0.0224 - accuracy: 0.9986 - 1s/epoch - 8ms/step
Epoch 36/50
175/175 - 2s - loss: 0.0200 - accuracy: 0.9987 - 2s/epoch - 9ms/step
Epoch 37/50
175/175 - 1s - loss: 0.0180 - accuracy: 0.9991 - 1s/epoch - 8ms/step
Epoch 38/50
175/175 - 1s - loss: 0.0163 - accuracy: 0.9989 - 1s/epoch - 8ms/step
Epoch 39/50
175/175 - 1s - loss: 0.0147 - accuracy: 0.9993 - 1s/epoch - 8ms/step
Epoch 40/50
175/175 - 1s - loss: 0.0131 - accuracy: 0.9995 - 1s/epoch - 8ms/step
Epoch 41/50
175/175 - 1s - loss: 0.0120 - accuracy: 0.9995 - 1s/epoch - 9ms/step
```

```
Epoch 42/50
175/175 - 1s - loss: 0.0109 - accuracy: 0.9996 - 1s/epoch - 8ms/step
Epoch 43/50
175/175 - 1s - loss: 0.0097 - accuracy: 0.9995 - 1s/epoch - 8ms/step
Epoch 44/50
175/175 - 1s - loss: 0.0089 - accuracy: 0.9993 - 1s/epoch - 8ms/step
Epoch 45/50
175/175 - 2s - loss: 0.0082 - accuracy: 0.9993 - 2s/epoch - 9ms/step
Epoch 46/50
175/175 - 1s - loss: 0.0073 - accuracy: 0.9995 - 1s/epoch - 8ms/step
Epoch 47/50
175/175 - 1s - loss: 0.0066 - accuracy: 0.9993 - 1s/epoch - 8ms/step
Epoch 48/50
175/175 - 1s - loss: 0.0063 - accuracy: 0.9993 - 1s/epoch - 8ms/step
Epoch 49/50
175/175 - 1s - loss: 0.0057 - accuracy: 0.9995 - 1s/epoch - 8ms/step
Epoch 50/50
175/175 - 1s - loss: 0.0050 - accuracy: 0.9995 - 1s/epoch - 8ms/step
ROC_AOC_Score for ANN:  0.7247009015105796
Test Accuracy: 0.7739782929420471
```

[34]:
```
run_models(x_train6, x_test6, y_train6, y_test6, n_words6)
```

```
FOR NAIVE BAYES:

Test Accuracy Score of Basic Naive Bayes Model: 81.78
Precision : 0.8177648040033361
Recall : 0.8177648040033361
F1-score : 0.8177648040033361
ROC_AOC_Score for Naive Bayes:  0.7198468373873985

FOR LOGISTIC REGRESSION:

Test Accuracy Score of Basic Logistic Regression Model: 81.69
Precision : 0.816930775646372
Recall : 0.816930775646372
F1-score : 0.816930775646372
ROC_AOC_Score for Logistic Regression:  0.7449502965694226

FOR LINEAR SVC:

Test Accuracy Score of Basic Linear SVC Model: 81.69
Precision : 0.816930775646372
Recall : 0.816930775646372
F1-score : 0.816930775646372
ROC_AOC_Score for Linear SVC:  0.7449502965694226
```

```
FOR RANDOM FOREST:

Test Accuracy Score of Basic Random Forest Model: 81.48
Precision : 0.8148457047539617
Recall : 0.8148457047539617
F1-score : 0.8148457047539617
ROC_AOC_Score for Random Forest:  0.7292817101375464

FOR ANN:

Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 50)                250050

 dense_11 (Dense)            (None, 1)                 51

=================================================================
Total params: 250,101
Trainable params: 250,101
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
175/175 - 2s - loss: 0.4816 - accuracy: 0.8200 - 2s/epoch - 12ms/step
Epoch 2/50
175/175 - 1s - loss: 0.3889 - accuracy: 0.8262 - 1s/epoch - 8ms/step
Epoch 3/50
175/175 - 1s - loss: 0.3441 - accuracy: 0.8453 - 1s/epoch - 8ms/step
Epoch 4/50
175/175 - 1s - loss: 0.2985 - accuracy: 0.8680 - 1s/epoch - 8ms/step
Epoch 5/50
175/175 - 1s - loss: 0.2498 - accuracy: 0.8974 - 1s/epoch - 8ms/step
Epoch 6/50
175/175 - 1s - loss: 0.2017 - accuracy: 0.9281 - 1s/epoch - 8ms/step
Epoch 7/50
175/175 - 1s - loss: 0.1583 - accuracy: 0.9524 - 1s/epoch - 8ms/step
Epoch 8/50
175/175 - 2s - loss: 0.1229 - accuracy: 0.9687 - 2s/epoch - 9ms/step
Epoch 9/50
175/175 - 2s - loss: 0.0934 - accuracy: 0.9807 - 2s/epoch - 9ms/step
Epoch 10/50
175/175 - 2s - loss: 0.0704 - accuracy: 0.9896 - 2s/epoch - 10ms/step
Epoch 11/50
175/175 - 1s - loss: 0.0530 - accuracy: 0.9950 - 1s/epoch - 8ms/step
Epoch 12/50
175/175 - 1s - loss: 0.0405 - accuracy: 0.9964 - 1s/epoch - 8ms/step
Epoch 13/50
```

```
175/175 - 1s - loss: 0.0315 - accuracy: 0.9975 - 1s/epoch - 8ms/step
Epoch 14/50
175/175 - 1s - loss: 0.0247 - accuracy: 0.9982 - 1s/epoch - 8ms/step
Epoch 15/50
175/175 - 1s - loss: 0.0199 - accuracy: 0.9986 - 1s/epoch - 8ms/step
Epoch 16/50
175/175 - 1s - loss: 0.0163 - accuracy: 0.9986 - 1s/epoch - 8ms/step
Epoch 17/50
175/175 - 1s - loss: 0.0135 - accuracy: 0.9986 - 1s/epoch - 8ms/step
Epoch 18/50
175/175 - 1s - loss: 0.0112 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 19/50
175/175 - 2s - loss: 0.0096 - accuracy: 0.9989 - 2s/epoch - 9ms/step
Epoch 20/50
175/175 - 2s - loss: 0.0080 - accuracy: 0.9993 - 2s/epoch - 9ms/step
Epoch 21/50
175/175 - 2s - loss: 0.0071 - accuracy: 0.9993 - 2s/epoch - 9ms/step
Epoch 22/50
175/175 - 2s - loss: 0.0060 - accuracy: 0.9993 - 2s/epoch - 10ms/step
Epoch 23/50
175/175 - 2s - loss: 0.0052 - accuracy: 0.9998 - 2s/epoch - 10ms/step
Epoch 24/50
175/175 - 2s - loss: 0.0046 - accuracy: 0.9998 - 2s/epoch - 9ms/step
Epoch 25/50
175/175 - 2s - loss: 0.0039 - accuracy: 0.9998 - 2s/epoch - 9ms/step
Epoch 26/50
175/175 - 2s - loss: 0.0034 - accuracy: 0.9998 - 2s/epoch - 9ms/step
Epoch 27/50
175/175 - 1s - loss: 0.0030 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 28/50
175/175 - 1s - loss: 0.0028 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 29/50
175/175 - 1s - loss: 0.0025 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 30/50
175/175 - 1s - loss: 0.0022 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 31/50
175/175 - 1s - loss: 0.0018 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 32/50
175/175 - 1s - loss: 0.0017 - accuracy: 0.9996 - 1s/epoch - 8ms/step
Epoch 33/50
175/175 - 1s - loss: 0.0015 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 34/50
175/175 - 1s - loss: 0.0014 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 35/50
175/175 - 1s - loss: 0.0012 - accuracy: 0.9998 - 1s/epoch - 9ms/step
Epoch 36/50
175/175 - 1s - loss: 0.0011 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 37/50
```

```
175/175 - 1s - loss: 9.7679e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 38/50
175/175 - 1s - loss: 8.9839e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 39/50
175/175 - 2s - loss: 7.9541e-04 - accuracy: 0.9998 - 2s/epoch - 9ms/step
Epoch 40/50
175/175 - 2s - loss: 7.4248e-04 - accuracy: 0.9998 - 2s/epoch - 9ms/step
Epoch 41/50
175/175 - 1s - loss: 6.9646e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 42/50
175/175 - 1s - loss: 6.1476e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 43/50
175/175 - 1s - loss: 5.5358e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 44/50
175/175 - 1s - loss: 5.1071e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 45/50
175/175 - 1s - loss: 4.6611e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 46/50
175/175 - 1s - loss: 4.3868e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 47/50
175/175 - 1s - loss: 4.0239e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 48/50
175/175 - 1s - loss: 3.7140e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 49/50
175/175 - 1s - loss: 3.5070e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
Epoch 50/50
175/175 - 1s - loss: 3.4548e-04 - accuracy: 1.0000 - 1s/epoch - 8ms/step
ROC_AOC_Score for ANN:  0.7292817101375464
Test Accuracy: 0.7710592150688171
```

[35]:

```
perform_list = perform_list1 + perform_list2 + perform_list3 + perform_list4 +␣
↪perform_list5
model_performance_I = pd.DataFrame(data=perform_list)
model_performance_I = model_performance_I[['Model', 'Test Accuracy',␣
↪'Precision', 'Recall', 'F1', 'Loss']]
model_performance_I
```

[35]:

	Model	Test Accuracy	Precision	Recall	F1	Loss
0	Naive Bayes	67.76	0.68	0.68	0.68	NaN
1	Naive Bayes	67.26	0.67	0.67	0.67	NaN
2	Naive Bayes	76.65	0.77	0.77	0.77	NaN
3	Naive Bayes	75.65	0.76	0.76	0.76	NaN
4	Naive Bayes	82.19	0.82	0.82	0.82	NaN
5	Naive Bayes	81.78	0.82	0.82	0.82	NaN
6	Logistic Regression	77.36	0.77	0.77	0.77	NaN
7	Logistic Regression	76.36	0.76	0.76	0.76	NaN
8	Logistic Regression	79.36	0.79	0.79	0.79	NaN

9	Logistic Regression	78.15	0.78	0.78	0.78	NaN
10	Logistic Regression	81.65	0.82	0.82	0.82	NaN
11	Logistic Regression	81.69	0.82	0.82	0.82	NaN
12	Linear SVC	81.61	0.82	0.82	0.82	NaN
13	Linear SVC	81.65	0.82	0.82	0.82	NaN
14	Linear SVC	81.90	0.82	0.82	0.82	NaN
15	Linear SVC	81.65	0.82	0.82	0.82	NaN
16	Linear SVC	81.65	0.82	0.82	0.82	NaN
17	Linear SVC	81.69	0.82	0.82	0.82	NaN
18	Random Forest	81.53	0.82	0.82	0.82	NaN
19	Random Forest	81.61	0.82	0.82	0.82	NaN
20	Random Forest	81.69	0.82	0.82	0.82	NaN
21	Random Forest	81.44	0.81	0.81	0.81	NaN
22	Random Forest	81.78	0.82	0.82	0.82	NaN
23	Random Forest	81.48	0.81	0.81	0.81	NaN
24	ANN	0.78	NaN	NaN	NaN	1.50
25	ANN	0.78	NaN	NaN	NaN	1.89
26	ANN	0.78	NaN	NaN	NaN	1.45
27	ANN	0.77	NaN	NaN	NaN	2.12
28	ANN	0.77	NaN	NaN	NaN	1.01
29	ANN	0.77	NaN	NaN	NaN	1.69

9 CLASS BALANCING

[36]:
```
our_data["relevance"].value_counts()
```

[36]:
```
relevance
0    6571
1    1420
Name: count, dtype: int64
```

[37]:
```
our_data.isnull().sum()
```

[37]:
```
text         0
relevance    0
dtype: int64
```

[38]:
```
#NO SAMPLING
x8 = x6
y8 = y6

counter = Counter(y8)
print('Count: ',counter)
```

```
Count:  Counter({0: 6571, 1: 1420})
```

[39]:
```
#Under Sampling
x9 = x6
y9 = y6

counter = Counter(y9)
print('Before Under Sampling: ',counter)

# transform the dataset
undersample = RandomUnderSampler(sampling_strategy=0.5)  #current sample in␣
↪minority/0.5 = updated samples in majority class
x9, y9 = undersample.fit_resample(x9, y9)

counter = Counter(y9)
print('After Under Sampling: ',counter)
```

```
Before Under Sampling:  Counter({0: 6571, 1: 1420})
After Under Sampling:  Counter({0: 2840, 1: 1420})
```

[40]:
```
#Over Sampling
x10 = x6
y10 = y6

counter = Counter(y10)
print('Before Over Sampling: ',counter)

# transform the dataset
oversample = RandomOverSampler(sampling_strategy=0.5)   #0.5 * majority class =␣
↪updates sample count in minority
x10, y10 = oversample.fit_resample(x10, y10)

counter = Counter(y10)
print('After Over Sampling: ',counter)
```

```
Before Over Sampling:  Counter({0: 6571, 1: 1420})
After Over Sampling:  Counter({0: 6571, 1: 3285})
```

[41]:
```
#SMOTE
x11 = x6
y11 = y6

counter = Counter(y11)
print('Before SMOTE: ',counter)

# transform the dataset
smote = SMOTE(sampling_strategy=0.7)         # 0.7*samples in majority =␣
↪updated sample count in minority class
x11, y11 = smote.fit_resample(x11, y11)
```

```
counter = Counter(y11)
print('After SMOTE: ',counter)
```

```
Before SMOTE:  Counter({0: 6571, 1: 1420})
After SMOTE:  Counter({0: 6571, 1: 4599})
```

[42]:
```
#Train test split for class balancing
#No sampling
x_train8, x_test8, y_train8, y_test8 = train_test_split(x8, y8, test_size = 0.
↪3, random_state = 0, shuffle = True)
print(len(x_train8))
print(len(x_test8))

#Under Sampling
x_train9, x_test9, y_train9, y_test9 = train_test_split(x9, y9, test_size = 0.
↪3, random_state = 0, shuffle = True)
print(len(x_train9))
print(len(x_test9))


#Over Sampling
x_train10, x_test10, y_train10, y_test10 = train_test_split(x10, y10, test_size␣
↪= 0.3, random_state = 0, shuffle = True)
print(len(x_train10))
print(len(x_test10))

#SMOTE
x_train11, x_test11, y_train11, y_test11 = train_test_split(x11, y11, test_size␣
↪= 0.3, random_state = 0, shuffle = True)
print(len(x_train11))
print(len(x_test11))
```

```
5593
2398
2982
1278
6899
2957
7819
3351
```

[43]:
```
#Normal
n_words8 = x_test8.shape[1]
n_words9 = x_test9.shape[1]
n_words10 = x_test10.shape[1]
n_words11 = x_test11.shape[1]
```

```
[44]: #No sampling
run_models(x_train8, x_test8, y_train8, y_test8, n_words8)
```

```
FOR NAIVE BAYES:

Test Accuracy Score of Basic Naive Bayes Model: 81.78
Precision : 0.8177648040033361
Recall : 0.8177648040033361
F1-score : 0.8177648040033361
ROC_AOC_Score for Naive Bayes:  0.7198468373873985

FOR LOGISTIC REGRESSION:

Test Accuracy Score of Basic Logistic Regression Model: 81.69
Precision : 0.816930775646372
Recall : 0.816930775646372
F1-score : 0.816930775646372
ROC_AOC_Score for Logistic Regression:  0.7449502965694226

FOR LINEAR SVC:

Test Accuracy Score of Basic Linear SVC Model: 81.69
Precision : 0.816930775646372
Recall : 0.816930775646372
F1-score : 0.816930775646372
ROC_AOC_Score for Linear SVC:  0.7449502965694226

FOR RANDOM FOREST:

Test Accuracy Score of Basic Random Forest Model: 81.48
Precision : 0.8148457047539617
Recall : 0.8148457047539617
F1-score : 0.8148457047539617
ROC_AOC_Score for Random Forest:  0.7292817101375464

FOR ANN:

Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 50)                250050

 dense_13 (Dense)            (None, 1)                 51

=================================================================
```

```
Total params: 250,101
Trainable params: 250,101
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
175/175 - 2s - loss: 0.4752 - accuracy: 0.8225 - 2s/epoch - 12ms/step
Epoch 2/50
175/175 - 2s - loss: 0.3846 - accuracy: 0.8262 - 2s/epoch - 9ms/step
Epoch 3/50
175/175 - 2s - loss: 0.3388 - accuracy: 0.8486 - 2s/epoch - 9ms/step
Epoch 4/50
175/175 - 2s - loss: 0.2902 - accuracy: 0.8723 - 2s/epoch - 9ms/step
Epoch 5/50
175/175 - 2s - loss: 0.2394 - accuracy: 0.9033 - 2s/epoch - 9ms/step
Epoch 6/50
175/175 - 2s - loss: 0.1904 - accuracy: 0.9349 - 2s/epoch - 9ms/step
Epoch 7/50
175/175 - 2s - loss: 0.1490 - accuracy: 0.9566 - 2s/epoch - 9ms/step
Epoch 8/50
175/175 - 2s - loss: 0.1137 - accuracy: 0.9719 - 2s/epoch - 10ms/step
Epoch 9/50
175/175 - 2s - loss: 0.0865 - accuracy: 0.9839 - 2s/epoch - 10ms/step
Epoch 10/50
175/175 - 1s - loss: 0.0659 - accuracy: 0.9918 - 1s/epoch - 8ms/step
Epoch 11/50
175/175 - 1s - loss: 0.0499 - accuracy: 0.9948 - 1s/epoch - 8ms/step
Epoch 12/50
175/175 - 1s - loss: 0.0387 - accuracy: 0.9966 - 1s/epoch - 8ms/step
Epoch 13/50
175/175 - 1s - loss: 0.0298 - accuracy: 0.9975 - 1s/epoch - 8ms/step
Epoch 14/50
175/175 - 1s - loss: 0.0236 - accuracy: 0.9980 - 1s/epoch - 8ms/step
Epoch 15/50
175/175 - 1s - loss: 0.0191 - accuracy: 0.9980 - 1s/epoch - 8ms/step
Epoch 16/50
175/175 - 1s - loss: 0.0156 - accuracy: 0.9986 - 1s/epoch - 8ms/step
Epoch 17/50
175/175 - 1s - loss: 0.0131 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 18/50
175/175 - 1s - loss: 0.0107 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 19/50
175/175 - 1s - loss: 0.0093 - accuracy: 0.9986 - 1s/epoch - 8ms/step
Epoch 20/50
175/175 - 1s - loss: 0.0081 - accuracy: 0.9987 - 1s/epoch - 8ms/step
Epoch 21/50
175/175 - 1s - loss: 0.0070 - accuracy: 0.9996 - 1s/epoch - 8ms/step
Epoch 22/50
175/175 - 1s - loss: 0.0061 - accuracy: 0.9993 - 1s/epoch - 8ms/step
```

```
Epoch 23/50
175/175 - 1s - loss: 0.0053 - accuracy: 0.9996 - 1s/epoch - 8ms/step
Epoch 24/50
175/175 - 1s - loss: 0.0047 - accuracy: 0.9996 - 1s/epoch - 8ms/step
Epoch 25/50
175/175 - 1s - loss: 0.0042 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 26/50
175/175 - 1s - loss: 0.0037 - accuracy: 0.9996 - 1s/epoch - 8ms/step
Epoch 27/50
175/175 - 1s - loss: 0.0033 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 28/50
175/175 - 1s - loss: 0.0031 - accuracy: 0.9996 - 1s/epoch - 8ms/step
Epoch 29/50
175/175 - 1s - loss: 0.0027 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 30/50
175/175 - 1s - loss: 0.0024 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 31/50
175/175 - 1s - loss: 0.0022 - accuracy: 0.9996 - 1s/epoch - 8ms/step
Epoch 32/50
175/175 - 1s - loss: 0.0021 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 33/50
175/175 - 1s - loss: 0.0019 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 34/50
175/175 - 1s - loss: 0.0017 - accuracy: 0.9998 - 1s/epoch - 9ms/step
Epoch 35/50
175/175 - 1s - loss: 0.0016 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 36/50
175/175 - 2s - loss: 0.0015 - accuracy: 0.9998 - 2s/epoch - 9ms/step
Epoch 37/50
175/175 - 1s - loss: 0.0014 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 38/50
175/175 - 1s - loss: 0.0013 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 39/50
175/175 - 1s - loss: 0.0013 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 40/50
175/175 - 1s - loss: 0.0012 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 41/50
175/175 - 1s - loss: 0.0010 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 42/50
175/175 - 1s - loss: 9.3183e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 43/50
175/175 - 1s - loss: 8.6835e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 44/50
175/175 - 1s - loss: 8.1570e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 45/50
175/175 - 1s - loss: 7.9774e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 46/50
175/175 - 1s - loss: 7.3990e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
```

```
Epoch 47/50
175/175 - 1s - loss: 7.0206e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 48/50
175/175 - 1s - loss: 6.8851e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 49/50
175/175 - 1s - loss: 6.3399e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
Epoch 50/50
175/175 - 1s - loss: 6.0208e-04 - accuracy: 0.9998 - 1s/epoch - 8ms/step
ROC_AOC_Score for ANN:  0.7292817101375464
Test Accuracy: 0.7693911790847778
```

[45]:
```
#Under Sampling
run_models(x_train9, x_test9, y_train9, y_test9, n_words9)
```

```
FOR NAIVE BAYES:

Test Accuracy Score of Basic Naive Bayes Model: 72.85
Precision : 0.7284820031298904
Recall : 0.7284820031298904
F1-score : 0.7284820031298903
ROC_AOC_Score for Naive Bayes:  0.7564270214709913

FOR LOGISTIC REGRESSION:

Test Accuracy Score of Basic Logistic Regression Model: 72.61
Precision : 0.7261345852895149
Recall : 0.7261345852895149
F1-score : 0.7261345852895149
ROC_AOC_Score for Logistic Regression:  0.7587254454088626

FOR LINEAR SVC:

Test Accuracy Score of Basic Linear SVC Model: 73.0
Precision : 0.7300469483568075
Recall : 0.7300469483568075
F1-score : 0.7300469483568076
ROC_AOC_Score for Linear SVC:  0.7587254454088626

FOR RANDOM FOREST:

Test Accuracy Score of Basic Random Forest Model: 72.3
Precision : 0.7230046948356808
Recall : 0.7230046948356808
F1-score : 0.7230046948356808
ROC_AOC_Score for Random Forest:  0.7427892302421197

FOR ANN:
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_14 (Dense)            (None, 50)                250050

 dense_15 (Dense)            (None, 1)                 51

=================================================================
Total params: 250,101
Trainable params: 250,101
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
94/94 - 2s - loss: 0.6314 - accuracy: 0.6559 - 2s/epoch - 17ms/step
Epoch 2/50
94/94 - 1s - loss: 0.5308 - accuracy: 0.7401 - 743ms/epoch - 8ms/step
Epoch 3/50
94/94 - 1s - loss: 0.4445 - accuracy: 0.8005 - 744ms/epoch - 8ms/step
Epoch 4/50
94/94 - 1s - loss: 0.3667 - accuracy: 0.8551 - 755ms/epoch - 8ms/step
Epoch 5/50
94/94 - 1s - loss: 0.2955 - accuracy: 0.9027 - 817ms/epoch - 9ms/step
Epoch 6/50
94/94 - 1s - loss: 0.2330 - accuracy: 0.9393 - 883ms/epoch - 9ms/step
Epoch 7/50
94/94 - 1s - loss: 0.1819 - accuracy: 0.9608 - 829ms/epoch - 9ms/step
Epoch 8/50
94/94 - 1s - loss: 0.1421 - accuracy: 0.9789 - 872ms/epoch - 9ms/step
Epoch 9/50
94/94 - 1s - loss: 0.1107 - accuracy: 0.9893 - 889ms/epoch - 9ms/step
Epoch 10/50
94/94 - 1s - loss: 0.0860 - accuracy: 0.9953 - 805ms/epoch - 9ms/step
Epoch 11/50
94/94 - 1s - loss: 0.0683 - accuracy: 0.9973 - 871ms/epoch - 9ms/step
Epoch 12/50
94/94 - 1s - loss: 0.0544 - accuracy: 0.9987 - 868ms/epoch - 9ms/step
Epoch 13/50
94/94 - 1s - loss: 0.0436 - accuracy: 0.9993 - 849ms/epoch - 9ms/step
Epoch 14/50
94/94 - 1s - loss: 0.0356 - accuracy: 0.9993 - 816ms/epoch - 9ms/step
Epoch 15/50
94/94 - 1s - loss: 0.0297 - accuracy: 0.9997 - 877ms/epoch - 9ms/step
Epoch 16/50
94/94 - 1s - loss: 0.0250 - accuracy: 0.9993 - 902ms/epoch - 10ms/step
Epoch 17/50
94/94 - 1s - loss: 0.0215 - accuracy: 0.9993 - 854ms/epoch - 9ms/step
```

```
Epoch 18/50
94/94 - 1s - loss: 0.0183 - accuracy: 0.9997 - 778ms/epoch - 8ms/step
Epoch 19/50
94/94 - 1s - loss: 0.0160 - accuracy: 0.9993 - 743ms/epoch - 8ms/step
Epoch 20/50
94/94 - 1s - loss: 0.0138 - accuracy: 0.9997 - 756ms/epoch - 8ms/step
Epoch 21/50
94/94 - 1s - loss: 0.0122 - accuracy: 0.9997 - 747ms/epoch - 8ms/step
Epoch 22/50
94/94 - 1s - loss: 0.0109 - accuracy: 0.9993 - 756ms/epoch - 8ms/step
Epoch 23/50
94/94 - 1s - loss: 0.0096 - accuracy: 0.9997 - 741ms/epoch - 8ms/step
Epoch 24/50
94/94 - 1s - loss: 0.0090 - accuracy: 0.9993 - 739ms/epoch - 8ms/step
Epoch 25/50
94/94 - 1s - loss: 0.0078 - accuracy: 0.9997 - 754ms/epoch - 8ms/step
Epoch 26/50
94/94 - 1s - loss: 0.0072 - accuracy: 0.9993 - 762ms/epoch - 8ms/step
Epoch 27/50
94/94 - 1s - loss: 0.0064 - accuracy: 0.9997 - 788ms/epoch - 8ms/step
Epoch 28/50
94/94 - 1s - loss: 0.0059 - accuracy: 0.9997 - 795ms/epoch - 8ms/step
Epoch 29/50
94/94 - 1s - loss: 0.0059 - accuracy: 0.9993 - 741ms/epoch - 8ms/step
Epoch 30/50
94/94 - 1s - loss: 0.0051 - accuracy: 0.9997 - 739ms/epoch - 8ms/step
Epoch 31/50
94/94 - 1s - loss: 0.0045 - accuracy: 0.9997 - 760ms/epoch - 8ms/step
Epoch 32/50
94/94 - 1s - loss: 0.0045 - accuracy: 0.9993 - 863ms/epoch - 9ms/step
Epoch 33/50
94/94 - 1s - loss: 0.0045 - accuracy: 0.9993 - 832ms/epoch - 9ms/step
Epoch 34/50
94/94 - 1s - loss: 0.0039 - accuracy: 0.9993 - 877ms/epoch - 9ms/step
Epoch 35/50
94/94 - 1s - loss: 0.0035 - accuracy: 0.9997 - 820ms/epoch - 9ms/step
Epoch 36/50
94/94 - 1s - loss: 0.0035 - accuracy: 0.9993 - 800ms/epoch - 9ms/step
Epoch 37/50
94/94 - 1s - loss: 0.0034 - accuracy: 0.9993 - 827ms/epoch - 9ms/step
Epoch 38/50
94/94 - 1s - loss: 0.0029 - accuracy: 0.9997 - 872ms/epoch - 9ms/step
Epoch 39/50
94/94 - 1s - loss: 0.0033 - accuracy: 0.9993 - 868ms/epoch - 9ms/step
Epoch 40/50
94/94 - 1s - loss: 0.0033 - accuracy: 0.9993 - 807ms/epoch - 9ms/step
Epoch 41/50
94/94 - 1s - loss: 0.0031 - accuracy: 0.9993 - 799ms/epoch - 9ms/step
```

```
Epoch 42/50
94/94 - 1s - loss: 0.0023 - accuracy: 0.9997 - 811ms/epoch - 9ms/step
Epoch 43/50
94/94 - 1s - loss: 0.0029 - accuracy: 0.9993 - 782ms/epoch - 8ms/step
Epoch 44/50
94/94 - 1s - loss: 0.0028 - accuracy: 0.9993 - 779ms/epoch - 8ms/step
Epoch 45/50
94/94 - 1s - loss: 0.0027 - accuracy: 0.9993 - 772ms/epoch - 8ms/step
Epoch 46/50
94/94 - 1s - loss: 0.0020 - accuracy: 0.9997 - 784ms/epoch - 8ms/step
Epoch 47/50
94/94 - 1s - loss: 0.0019 - accuracy: 0.9997 - 836ms/epoch - 9ms/step
Epoch 48/50
94/94 - 1s - loss: 0.0024 - accuracy: 0.9993 - 833ms/epoch - 9ms/step
Epoch 49/50
94/94 - 1s - loss: 0.0022 - accuracy: 0.9993 - 853ms/epoch - 9ms/step
Epoch 50/50
94/94 - 1s - loss: 0.0017 - accuracy: 0.9993 - 861ms/epoch - 9ms/step
ROC_AOC_Score for ANN:  0.7427892302421197
Test Accuracy: 0.6690140962600708
```

[46]:
```
#Over Sampling
run_models(x_train10, x_test10, y_train10, y_test10, n_words10)
```

```
FOR NAIVE BAYES:

Test Accuracy Score of Basic Naive Bayes Model: 73.55
Precision : 0.735542779844437
Recall : 0.735542779844437
F1-score : 0.7355427798444371
ROC_AOC_Score for Naive Bayes:  0.7851504008941004

FOR LOGISTIC REGRESSION:

Test Accuracy Score of Basic Logistic Regression Model: 76.73
Precision : 0.7673317551572539
Recall : 0.7673317551572539
F1-score : 0.767331755157254
ROC_AOC_Score for Logistic Regression:  0.8348795876929473

FOR LINEAR SVC:

Test Accuracy Score of Basic Linear SVC Model: 86.95
Precision : 0.8694622928643896
Recall : 0.8694622928643896
F1-score : 0.8694622928643896
ROC_AOC_Score for Linear SVC:  0.8348795876929473
```

```
FOR RANDOM FOREST:

Test Accuracy Score of Basic Random Forest Model: 91.58
Precision : 0.9157930334798783
Recall : 0.9157930334798783
F1-score : 0.9157930334798783
ROC_AOC_Score for Random Forest:  0.9334171552418794

FOR ANN:

Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_16 (Dense)            (None, 50)                250050

 dense_17 (Dense)            (None, 1)                 51

=================================================================
Total params: 250,101
Trainable params: 250,101
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
216/216 - 2s - loss: 0.5694 - accuracy: 0.7039 - 2s/epoch - 11ms/step
Epoch 2/50
216/216 - 2s - loss: 0.4262 - accuracy: 0.8100 - 2s/epoch - 8ms/step
Epoch 3/50
216/216 - 2s - loss: 0.3193 - accuracy: 0.8778 - 2s/epoch - 8ms/step
Epoch 4/50
216/216 - 2s - loss: 0.2392 - accuracy: 0.9191 - 2s/epoch - 8ms/step
Epoch 5/50
216/216 - 2s - loss: 0.1791 - accuracy: 0.9478 - 2s/epoch - 9ms/step
Epoch 6/50
216/216 - 2s - loss: 0.1319 - accuracy: 0.9684 - 2s/epoch - 8ms/step
Epoch 7/50
216/216 - 2s - loss: 0.0977 - accuracy: 0.9825 - 2s/epoch - 8ms/step
Epoch 8/50
216/216 - 2s - loss: 0.0719 - accuracy: 0.9913 - 2s/epoch - 8ms/step
Epoch 9/50
216/216 - 2s - loss: 0.0529 - accuracy: 0.9948 - 2s/epoch - 8ms/step
Epoch 10/50
216/216 - 2s - loss: 0.0397 - accuracy: 0.9970 - 2s/epoch - 8ms/step
Epoch 11/50
216/216 - 2s - loss: 0.0301 - accuracy: 0.9977 - 2s/epoch - 8ms/step
Epoch 12/50
216/216 - 2s - loss: 0.0231 - accuracy: 0.9991 - 2s/epoch - 8ms/step
```

```
Epoch 13/50
216/216 - 2s - loss: 0.0184 - accuracy: 0.9988 - 2s/epoch - 8ms/step
Epoch 14/50
216/216 - 2s - loss: 0.0150 - accuracy: 0.9991 - 2s/epoch - 8ms/step
Epoch 15/50
216/216 - 2s - loss: 0.0119 - accuracy: 0.9991 - 2s/epoch - 8ms/step
Epoch 16/50
216/216 - 2s - loss: 0.0103 - accuracy: 0.9991 - 2s/epoch - 8ms/step
Epoch 17/50
216/216 - 2s - loss: 0.0085 - accuracy: 0.9994 - 2s/epoch - 8ms/step
Epoch 18/50
216/216 - 2s - loss: 0.0075 - accuracy: 0.9993 - 2s/epoch - 8ms/step
Epoch 19/50
216/216 - 2s - loss: 0.0063 - accuracy: 0.9997 - 2s/epoch - 9ms/step
Epoch 20/50
216/216 - 2s - loss: 0.0054 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 21/50
216/216 - 2s - loss: 0.0042 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 22/50
216/216 - 2s - loss: 0.0037 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 23/50
216/216 - 2s - loss: 0.0032 - accuracy: 0.9999 - 2s/epoch - 9ms/step
Epoch 24/50
216/216 - 2s - loss: 0.0028 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 25/50
216/216 - 2s - loss: 0.0031 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 26/50
216/216 - 2s - loss: 0.0027 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 27/50
216/216 - 2s - loss: 0.0039 - accuracy: 0.9996 - 2s/epoch - 8ms/step
Epoch 28/50
216/216 - 2s - loss: 0.0021 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 29/50
216/216 - 2s - loss: 0.0026 - accuracy: 0.9996 - 2s/epoch - 8ms/step
Epoch 30/50
216/216 - 2s - loss: 0.0024 - accuracy: 0.9996 - 2s/epoch - 8ms/step
Epoch 31/50
216/216 - 2s - loss: 0.0018 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 32/50
216/216 - 2s - loss: 0.0012 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 33/50
216/216 - 2s - loss: 0.0016 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 34/50
216/216 - 2s - loss: 0.0010 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 35/50
216/216 - 2s - loss: 0.0014 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 36/50
216/216 - 2s - loss: 0.0013 - accuracy: 0.9997 - 2s/epoch - 8ms/step
```

```
Epoch 37/50
216/216 - 2s - loss: 0.0013 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 38/50
216/216 - 2s - loss: 0.0012 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 39/50
216/216 - 2s - loss: 0.0010 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 40/50
216/216 - 2s - loss: 6.4344e-04 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 41/50
216/216 - 2s - loss: 6.8497e-04 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 42/50
216/216 - 2s - loss: 9.4828e-04 - accuracy: 0.9997 - 2s/epoch - 9ms/step
Epoch 43/50
216/216 - 2s - loss: 4.7308e-04 - accuracy: 0.9999 - 2s/epoch - 10ms/step
Epoch 44/50
216/216 - 2s - loss: 7.3303e-04 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 45/50
216/216 - 2s - loss: 0.0029 - accuracy: 0.9994 - 2s/epoch - 8ms/step
Epoch 46/50
216/216 - 2s - loss: 0.0012 - accuracy: 0.9994 - 2s/epoch - 8ms/step
Epoch 47/50
216/216 - 2s - loss: 9.6085e-04 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 48/50
216/216 - 2s - loss: 4.9224e-04 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 49/50
216/216 - 2s - loss: 4.8324e-04 - accuracy: 0.9999 - 2s/epoch - 9ms/step
Epoch 50/50
216/216 - 2s - loss: 5.6266e-04 - accuracy: 0.9999 - 2s/epoch - 9ms/step
ROC_AOC_Score for ANN:  0.9334171552418794
Test Accuracy: 0.8562732338905334
```

[47]:
```
#SMOTE
run_models(x_train11, x_test11, y_train11, y_test11, n_words11)
```

```
FOR NAIVE BAYES:

Test Accuracy Score of Basic Naive Bayes Model: 72.93
Precision : 0.7293345270068636
Recall : 0.7293345270068636
F1-score : 0.7293345270068636
ROC_AOC_Score for Naive Bayes:  0.8127680486061473

FOR LOGISTIC REGRESSION:

Test Accuracy Score of Basic Logistic Regression Model: 79.47
Precision : 0.7946881527902119
Recall : 0.7946881527902119
```

```
F1-score : 0.7946881527902119
ROC_AOC_Score for Logistic Regression:  0.8737589935433976

FOR LINEAR SVC:

Test Accuracy Score of Basic Linear SVC Model: 89.94
Precision : 0.8994330050731125
Recall : 0.8994330050731125
F1-score : 0.8994330050731125
ROC_AOC_Score for Linear SVC:  0.8737589935433976

FOR RANDOM FOREST:

Test Accuracy Score of Basic Random Forest Model: 89.94
Precision : 0.8994330050731125
Recall : 0.8994330050731125
F1-score : 0.8994330050731125
ROC_AOC_Score for Random Forest:  0.9535620437314709

FOR ANN:

Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_18 (Dense)            (None, 50)                250050

 dense_19 (Dense)            (None, 1)                 51

=================================================================
Total params: 250,101
Trainable params: 250,101
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
245/245 - 3s - loss: 0.5591 - accuracy: 0.7197 - 3s/epoch - 13ms/step
Epoch 2/50
245/245 - 2s - loss: 0.3954 - accuracy: 0.8267 - 2s/epoch - 8ms/step
Epoch 3/50
245/245 - 2s - loss: 0.2960 - accuracy: 0.8878 - 2s/epoch - 9ms/step
Epoch 4/50
245/245 - 2s - loss: 0.2280 - accuracy: 0.9240 - 2s/epoch - 9ms/step
Epoch 5/50
245/245 - 2s - loss: 0.1788 - accuracy: 0.9440 - 2s/epoch - 8ms/step
Epoch 6/50
245/245 - 2s - loss: 0.1399 - accuracy: 0.9628 - 2s/epoch - 8ms/step
Epoch 7/50
245/245 - 2s - loss: 0.1096 - accuracy: 0.9752 - 2s/epoch - 8ms/step
```

```
Epoch 8/50
245/245 - 2s - loss: 0.0845 - accuracy: 0.9844 - 2s/epoch - 8ms/step
Epoch 9/50
245/245 - 2s - loss: 0.0654 - accuracy: 0.9916 - 2s/epoch - 8ms/step
Epoch 10/50
245/245 - 2s - loss: 0.0496 - accuracy: 0.9945 - 2s/epoch - 8ms/step
Epoch 11/50
245/245 - 2s - loss: 0.0384 - accuracy: 0.9967 - 2s/epoch - 8ms/step
Epoch 12/50
245/245 - 2s - loss: 0.0293 - accuracy: 0.9985 - 2s/epoch - 9ms/step
Epoch 13/50
245/245 - 2s - loss: 0.0236 - accuracy: 0.9992 - 2s/epoch - 8ms/step
Epoch 14/50
245/245 - 2s - loss: 0.0187 - accuracy: 0.9991 - 2s/epoch - 8ms/step
Epoch 15/50
245/245 - 2s - loss: 0.0156 - accuracy: 0.9991 - 2s/epoch - 8ms/step
Epoch 16/50
245/245 - 2s - loss: 0.0126 - accuracy: 0.9994 - 2s/epoch - 8ms/step
Epoch 17/50
245/245 - 2s - loss: 0.0106 - accuracy: 0.9994 - 2s/epoch - 8ms/step
Epoch 18/50
245/245 - 2s - loss: 0.0090 - accuracy: 0.9992 - 2s/epoch - 8ms/step
Epoch 19/50
245/245 - 2s - loss: 0.0083 - accuracy: 0.9991 - 2s/epoch - 8ms/step
Epoch 20/50
245/245 - 2s - loss: 0.0066 - accuracy: 0.9995 - 2s/epoch - 8ms/step
Epoch 21/50
245/245 - 2s - loss: 0.0061 - accuracy: 0.9992 - 2s/epoch - 8ms/step
Epoch 22/50
245/245 - 2s - loss: 0.0049 - accuracy: 0.9994 - 2s/epoch - 8ms/step
Epoch 23/50
245/245 - 2s - loss: 0.0043 - accuracy: 0.9994 - 2s/epoch - 8ms/step
Epoch 24/50
245/245 - 2s - loss: 0.0038 - accuracy: 0.9995 - 2s/epoch - 8ms/step
Epoch 25/50
245/245 - 2s - loss: 0.0036 - accuracy: 0.9994 - 2s/epoch - 8ms/step
Epoch 26/50
245/245 - 2s - loss: 0.0028 - accuracy: 0.9996 - 2s/epoch - 8ms/step
Epoch 27/50
245/245 - 2s - loss: 0.0024 - accuracy: 0.9996 - 2s/epoch - 8ms/step
Epoch 28/50
245/245 - 2s - loss: 0.0019 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 29/50
245/245 - 2s - loss: 0.0023 - accuracy: 0.9996 - 2s/epoch - 8ms/step
Epoch 30/50
245/245 - 2s - loss: 0.0016 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 31/50
245/245 - 2s - loss: 0.0015 - accuracy: 0.9999 - 2s/epoch - 8ms/step
```

```
Epoch 32/50
245/245 - 2s - loss: 0.0013 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 33/50
245/245 - 2s - loss: 0.0027 - accuracy: 0.9995 - 2s/epoch - 8ms/step
Epoch 34/50
245/245 - 2s - loss: 0.0022 - accuracy: 0.9996 - 2s/epoch - 8ms/step
Epoch 35/50
245/245 - 2s - loss: 0.0034 - accuracy: 0.9994 - 2s/epoch - 9ms/step
Epoch 36/50
245/245 - 2s - loss: 0.0023 - accuracy: 0.9997 - 2s/epoch - 9ms/step
Epoch 37/50
245/245 - 2s - loss: 0.0014 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 38/50
245/245 - 2s - loss: 0.0012 - accuracy: 0.9996 - 2s/epoch - 8ms/step
Epoch 39/50
245/245 - 2s - loss: 0.0012 - accuracy: 0.9997 - 2s/epoch - 9ms/step
Epoch 40/50
245/245 - 2s - loss: 6.5497e-04 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 41/50
245/245 - 2s - loss: 3.2070e-04 - accuracy: 1.0000 - 2s/epoch - 8ms/step
Epoch 42/50
245/245 - 2s - loss: 3.1179e-04 - accuracy: 1.0000 - 2s/epoch - 8ms/step
Epoch 43/50
245/245 - 2s - loss: 2.5902e-04 - accuracy: 1.0000 - 2s/epoch - 9ms/step
Epoch 44/50
245/245 - 2s - loss: 2.6052e-04 - accuracy: 1.0000 - 2s/epoch - 8ms/step
Epoch 45/50
245/245 - 2s - loss: 2.8767e-04 - accuracy: 1.0000 - 2s/epoch - 8ms/step
Epoch 46/50
245/245 - 2s - loss: 0.0013 - accuracy: 0.9997 - 2s/epoch - 8ms/step
Epoch 47/50
245/245 - 2s - loss: 0.0018 - accuracy: 0.9996 - 2s/epoch - 8ms/step
Epoch 48/50
245/245 - 2s - loss: 0.0022 - accuracy: 0.9999 - 2s/epoch - 8ms/step
Epoch 49/50
245/245 - 2s - loss: 2.4163e-04 - accuracy: 1.0000 - 2s/epoch - 8ms/step
Epoch 50/50
245/245 - 2s - loss: 3.4206e-04 - accuracy: 1.0000 - 2s/epoch - 8ms/step
ROC_AOC_Score for ANN:  0.9535620437314709
Test Accuracy: 0.8743658661842346
```

[48]:
```
perform_list = perform_list1 + perform_list2 + perform_list3 + perform_list4 +␣
 ↪perform_list5
model_performance_II = pd.DataFrame(data=perform_list)
model_performance_II = model_performance_II[['Model', 'Test Accuracy',␣
 ↪'Precision', 'Recall', 'F1', 'Loss']]
model_performance_II
```

[48]:

	Model	Test Accuracy	Precision	Recall	F1	Loss
0	Naive Bayes	67.76	0.68	0.68	0.68	NaN
1	Naive Bayes	67.26	0.67	0.67	0.67	NaN
2	Naive Bayes	76.65	0.77	0.77	0.77	NaN
3	Naive Bayes	75.65	0.76	0.76	0.76	NaN
4	Naive Bayes	82.19	0.82	0.82	0.82	NaN
5	Naive Bayes	81.78	0.82	0.82	0.82	NaN
6	Naive Bayes	81.78	0.82	0.82	0.82	NaN
7	Naive Bayes	72.85	0.73	0.73	0.73	NaN
8	Naive Bayes	73.55	0.74	0.74	0.74	NaN
9	Naive Bayes	72.93	0.73	0.73	0.73	NaN
10	Logistic Regression	77.36	0.77	0.77	0.77	NaN
11	Logistic Regression	76.36	0.76	0.76	0.76	NaN
12	Logistic Regression	79.36	0.79	0.79	0.79	NaN
13	Logistic Regression	78.15	0.78	0.78	0.78	NaN
14	Logistic Regression	81.65	0.82	0.82	0.82	NaN
15	Logistic Regression	81.69	0.82	0.82	0.82	NaN
16	Logistic Regression	81.69	0.82	0.82	0.82	NaN
17	Logistic Regression	72.61	0.73	0.73	0.73	NaN
18	Logistic Regression	76.73	0.77	0.77	0.77	NaN
19	Logistic Regression	79.47	0.79	0.79	0.79	NaN
20	Linear SVC	81.61	0.82	0.82	0.82	NaN
21	Linear SVC	81.65	0.82	0.82	0.82	NaN
22	Linear SVC	81.90	0.82	0.82	0.82	NaN
23	Linear SVC	81.65	0.82	0.82	0.82	NaN
24	Linear SVC	81.65	0.82	0.82	0.82	NaN
25	Linear SVC	81.69	0.82	0.82	0.82	NaN
26	Linear SVC	81.69	0.82	0.82	0.82	NaN
27	Linear SVC	73.00	0.73	0.73	0.73	NaN
28	Linear SVC	86.95	0.87	0.87	0.87	NaN
29	Linear SVC	89.94	0.90	0.90	0.90	NaN
30	Random Forest	81.53	0.82	0.82	0.82	NaN
31	Random Forest	81.61	0.82	0.82	0.82	NaN
32	Random Forest	81.69	0.82	0.82	0.82	NaN
33	Random Forest	81.44	0.81	0.81	0.81	NaN
34	Random Forest	81.78	0.82	0.82	0.82	NaN
35	Random Forest	81.48	0.81	0.81	0.81	NaN
36	Random Forest	81.48	0.81	0.81	0.81	NaN
37	Random Forest	72.30	0.72	0.72	0.72	NaN
38	Random Forest	91.58	0.92	0.92	0.92	NaN
39	Random Forest	89.94	0.90	0.90	0.90	NaN
40	ANN	0.78	NaN	NaN	NaN	1.50
41	ANN	0.78	NaN	NaN	NaN	1.89
42	ANN	0.78	NaN	NaN	NaN	1.45
43	ANN	0.77	NaN	NaN	NaN	2.12
44	ANN	0.77	NaN	NaN	NaN	1.01
45	ANN	0.77	NaN	NaN	NaN	1.69

```
46                     ANN          0.77        NaN      NaN    NaN   1.76
47                     ANN          0.67        NaN      NaN    NaN   1.75
48                     ANN          0.86        NaN      NaN    NaN   1.20
49                     ANN          0.87        NaN      NaN    NaN   1.20
```

10 Output

Feature vectorization	Multinomial Naïve bayes				Logistic regression				Linear SVC				Random Forest				ANN			
	Max Features = 1000		Max Features =5000		Max Features = 1000		Max Features = 5000		Max Features = 1000		Max Features = 5000		Max Features = 1000		Max Features = 5000		Max Features = 1000		Max Features = 5000	
	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC
Bag Of Words	67.81	0.722	67.26	0.73	77.65	0.665	76.4	0.661	81.57	N/A	81.69	N/A	81.65	0.716	81.48	0.71	79	N/A	77.6	N/A
Bag Of n-grams	76.56	0.676	75.31	0.691	79.36	0.65	77.65	0.639	81.82	N/A	81.65	N/A	81.69	0.71	81.36	0.705	78.1	N/A	76.35	N/A
TF-IDF	82.11	0.727	81.65	0.719	81.65	0.73	81.73	0.743	81.69	N/A	81.65	N/A	81.61	0.718	81.44	0.732	0.778	N/A	0.768	N/A

Class Balancing	Multinomial Naïve bayes		Logistic regression		Linear SVC		Random Forest		ANN	
	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC
No Sampling	81.65	0.719	81.73	0.743	81.65	N/A	81.44	0.732	77.39	N/A
Under Sampling	73.79	0.768	74.41	0.78	74.18	N/A	72.69	0.762	66.5	N/A
Over Sampling	73.49	78.67	76.6	0.84	87.52	N/A	91.17	0.94	83.39	N/A
SMOTE	72.81	0.8	78.51	0.86	90.15	N/A	89.26	0.94	86.3	N/A

11 We can observe that TF-IDF (feature=1000) with Naive Bayes gives the most optimal accuracy at 82.11 %

For the second case, we can observe that Over sampling with Random Forest gives highest accuracy at 91.17%

Oversampling involves randomly duplicating examples from the minority class and adding them to the training dataset.