Week 5: Brandon Mather DSC550-T301

In [2]:
```
import pandas as pd
import numpy as np
import unicodedata
import sys
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction import DictVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn import tree
```

In [5]:
```
#1 Get the stemmed data using the same process you did in Week 3.
data = pd.read_csv("labeledTrainData.tsv", header=0, \
                   delimiter="\t", quoting=3)
```

In [6]:
```
review = data["review"]
```

In [7]:
```
def decapitalizer(string: str) -> str:
    return string.lower()
```

In [8]:
```
noncapital = [decapitalizer(string) for string in review]
```

In [9]:
```
punctuation = dict.fromkeys(i for i in range(sys.maxunicode)
    if unicodedata.category(chr(i)).startswith('P'))
```

In [10]:
```
noperiods = [string.translate(punctuation) for string in noncapital]
```

In [11]:
```
stop_words = stopwords.words('english')
```

In [12]:
```
nostopwords = [word for word in noperiods if word not in stop_words]
```

In [13]:
```
porter = PorterStemmer()
```

In [14]:
```
appliedstemmer = [porter.stem(word) for word in nostopwords]
```

In [15]:
```
#2 Split this into a training and test set.
x = appliedstemmer
y = data.sentiment
```

In [16]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2)
```

In [17]:
```
#3 Fit and apply the tf-idf vectorization to the training set.
tfidf_vectorizer = TfidfVectorizer()
```

In [18]:
```
tfidf_train = tfidf_vectorizer.fit_transform(x_train)
```

In [19]:
```
#4 Apply but DO NOT FIT the tf-idf vectorization to the test set (Why?).
tfidf_test = tfidf_vectorizer.transform(x_test)
```

In [17]:
```
#5 Train a logistic regression using the training data.
logit = LogisticRegression()
```

In [18]:
```
model = logit.fit(tfidf_train, y_train)
```

In [21]:
```
#6 Find the model accuracy on test set.
score = accuracy_score(y_test,x_test)
score
```

```
C:\Users\brand\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:217: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  score = y_true == y_pred
```

Out[21]:
```
0.0
```

In [26]:
```
#7 Create a confusion matrix for the test set predictions.
pred = model.predict(tfidf_test)
```

In [27]:
```
confusion_matrix(y_test, pred)
```

Out[27]:
```
array([[2178,  290],
       [ 267, 2265]], dtype=int64)
```

In [30]:
```
#8 Get the precision, recall, and F1-score for the test set predictions.
precision_score(y_test, pred)
```

Out[30]:
```
0.8864970645792564
```

In [31]:
```
recall_score(y_test, pred)
```

Out[31]:
```
0.8945497630331753
```

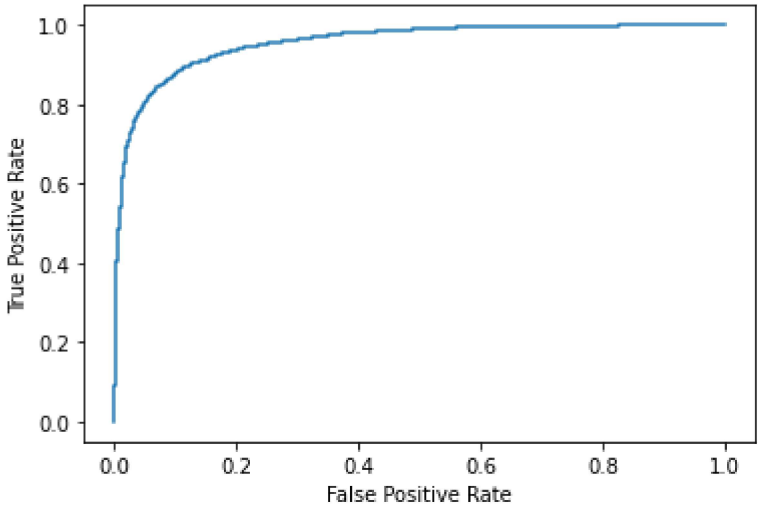
In [32]:
```
f1_score(y_test, pred)
```

Out[32]:
```
0.890505209357185
```

In [36]:
```
#9 Create a ROC curve for the test set.
y_pred_proba = logit.predict_proba(tfidf_test)[::,1]
```

In [37]:
```
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
```

In [38]:
```
plt.plot(fpr,tpr)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [ ]:
```
#10 Pick another classification model you learned about this week and repeat steps (5)
```

In [3]:
```
#Decision Tree
clf = tree.DecisionTreeClassifier()
```

In [21]:
```
clf = clf.fit(tfidf_train, y_train)
```

In [24]:
```
Y_pred = clf.predict(tfidf_test)
```

In [25]:
```
score_2 = accuracy_score(y_test, Y_pred)
score_2
```

Out[25]:
```
0.707
```

In [26]:
```
confusion_matrix(y_test, Y_pred)
```

Out[26]:
```
array([[1809,  750],
       [ 715, 1726]], dtype=int64)
```

In [27]:
```
precision_score(y_test, Y_pred)
```

Out[27]:
```
0.697092084006462
```

In [28]:
```
recall_score(y_test, Y_pred)
```

Out[28]:
```
0.7070872593199509
```

In [29]:
```
f1_score(y_test, Y_pred)
```

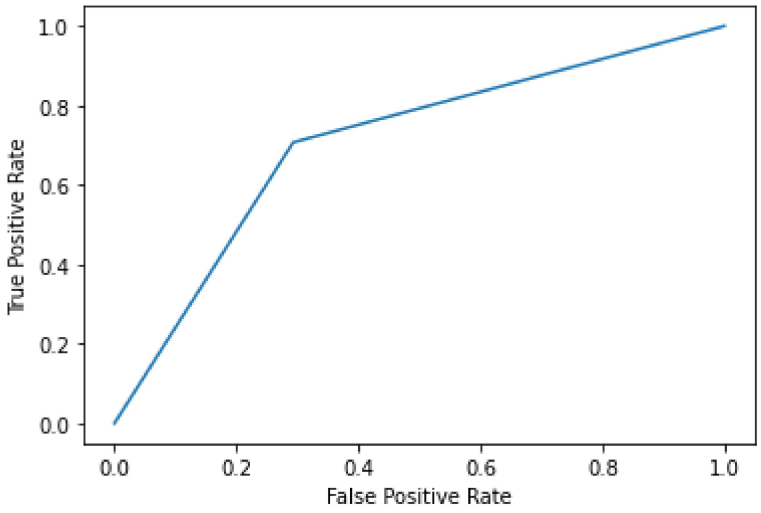
Out[29]:
```
0.7020540980272524
```

In [30]:
```
y_pred_proba_2 = clf.predict_proba(tfidf_test)[::,1]
```

In [31]:
```
fpr_2, tpr_2, _ = metrics.roc_curve(y_test,  y_pred_proba_2)
```

In [32]:
```
plt.plot(fpr_2,tpr_2)
plt.ylabel('True Positive Rate')
```

```
plt.xlabel('False Positive Rate')
plt.show()
```



In [ ]: