Sentiment Analysis Model

In [2]:
```
# import libraries
import pandas as pd
import numpy as np
from textblob import TextBlob

# check versions
print('pandas version:', pd.__version__)
print('numpy version:', np.__version__)
print('TextBlob version:', np.__version__)
```

```
pandas version: 1.4.2
numpy version: 1.21.5
TextBlob version: 1.21.5
```

1. Get the stemmed data using the same process you did in Week 3.

In [4]:
```
df = pd.read_csv('labeledTrainData.tsv',sep='\t')
df.head()
```

Out[4]:

	id	sentiment	review
0	5814_8	1	With all this stuff going down at the moment w...
1	2381_9	1	\The Classic War of the Worlds\" by Timothy Hi...
2	7759_3	0	The film starts with a manager (Nicholas Bell)...
3	3630_4	0	It must be assumed that those who praised this...
4	9495_8	1	Superbly trashy and wondrously unpretentious 8...

In [6]:
```
## convert all text to lowercase
df = df.applymap(lambda s: s.lower() if type(s) == str else s)
df.head()
```

Out[6]:

	id	sentiment	review
0	5814_8	1	with all this stuff going down at the moment w...
1	2381_9	1	\the classic war of the worlds\" by timothy hi...
2	7759_3	0	the film starts with a manager (nicholas bell)...
3	3630_4	0	it must be assumed that those who praised this...
4	9495_8	1	superbly trashy and wondrously unpretentious 8...

In [8]:
```
## remove all but alphanumeric
import re
df['review'] = df['review'].apply(lambda x: re.sub('[^A-Za-z0-9]', ' '
df.head()
```

Out[8]:

	id	sentiment	review
0	5814_8	1	with all this stuff going down at the moment w...
1	2381_9	1	the classic war of the worlds by timothy hi...
2	7759_3	0	the film starts with a manager nicholas bell ...
3	3630_4	0	it must be assumed that those who praised this...
4	9495_8	1	superbly trashy and wondrously unpretentious 8...

In [9]:
```
## remove stop words
## download library
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

stop_words = stopwords.words('english')
df['review'] = df['review'].apply(lambda x: ' '.join([word for word in
df.head()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\alexi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[9]:

	id	sentiment	review
0	5814_8	1	stuff going moment mj started listening music ...
1	2381_9	1	classic war worlds timothy hines entertaining ...
2	7759_3	0	film starts manager nicholas bell giving welco...
3	3630_4	0	must assumed praised film greatest filmed oper...
4	9495_8	1	superbly trashy wondrously unpretentious 80 ex...

In [10]:
```
## apply NLTK's porterstemmer
## download library
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()

## create tokenized words
tokenized_words = df['review']
## apply stemmer
[ps.stem(word) for word in tokenized_words]
df.head()
```

Out[10]:

	id	sentiment	review
0	5814_8	1	stuff going moment mj started listening music ...
1	2381_9	1	classic war worlds timothy hines entertaining ...
2	7759_3	0	film starts manager nicholas bell giving welco...
3	3630_4	0	must assumed praised film greatest filmed oper...
4	9495_8	1	superbly trashy wondrously unpretentious 80 ex...

2. Split this into training and test set.

In [11]:
```
## setting target
y = df.review
y.head()
```

Out[11]:
```
0    stuff going moment mj started listening music ...
1    classic war worlds timothy hines entertaining ...
2    film starts manager nicholas bell giving welco...
3    must assumed praised film greatest filmed oper...
4    superbly trashy wondrously unpretentious 80 ex...
Name: review, dtype: object
```

In [12]:
```
x = df.drop('review',axis=1)
x.head()
```

Out[12]:

	id	sentiment
0	5814_8	1
1	2381_9	1
2	7759_3	0
3	3630_4	0
4	9495_8	1

In [31]:
```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn import metrics
from sklearn.model_selection import KFold, cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

## standardizer
standardizer = StandardScaler()

## create logistic regression object
logit = LogisticRegression()

## split into training and test set
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

print("original df:", df.shape)
print("training set input", x_train.shape)
print("training set output", y_train.shape)
print("testing set input", x_test.shape)
print("testing set output", y_test.shape)
```

```
original df: (25000, 3)
training set input (20000, 2)
training set output (20000,)
testing set input (5000, 2)
testing set output (5000,)
```

3. Fit and apply the tf-idf vectorization to the training set.

In [32]:
```
standardizer.fit(x_train)
```

Out[32]:
```
StandardScaler()
```

In [17]:
```
from sklearn.feature_extraction.text import TfidfVectorizer as tfidfv
tfidf = tfidfv()
feature_matrix = tfidf.fit_transform(x_train, y_train)
feature_matrix
```

Out[17]:
```
<2x2 sparse matrix of type '<class 'numpy.float64'>'
        with 2 stored elements in Compressed Sparse Row format>
```

In [20]:
```
feature_matrix.toarray()
```

Out[20]:
```
array([[1., 0.],
       [0., 1.]])
```

In [33]:
```
feature_matrix_std = standardizer.transform(x_train)
```

4. Apply but DO NOT FIT the tf-idf vectorization to the test set. (Why?)

If we apply the tf-idf vectorization to the test set, then we are applying it to the entire df. Additionally, the test set is only 20% of the dataset and not a large enough sample to yield significant enough assumptions of our model quality. In this case, we are treating the test set as unknown data.

In [34]:
```
feature_test_std = standardizer.transform(x_test)
```

5. Train a logistic regression using the training data.

In [24]:
```
from sklearn.linear_model import LogisticRegression
```

I received a similar memory error in Google Colab, but the IDE is new to me, so I could've done something wrong. I tried to adjust the dtypes but also couldn't process. My laptop has 64gb of memory, and I attempted to run this on my work's technical laptop through Spyder with the same issues. Was there another way to optimize the models? Code below was not ran due to the error of the model but theorized based on model results.

In [ ]:
```
## received similar memory error in Google Colab
logit.fit(x_train, y_train)
```

6. Find the model accuracy on the test set.

In [ ]:
```
y_pred = logreg.predict(x_test)
print('Test Set Logicist Regression Accuracy': {:.2f}'.format(logreg.s
```

7. Create a confusion matrix for the test set predictions

In [ ]:
```
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(x_test, y_pred)
```

8. Get the precision, recall, and F1-score for the test set predictions

In [ ]:
```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import make_classification

## precision
cross_val_score(logit, x_test, y_test, scoring = "precision")
```

In [ ]:
```
## recall
cross_val_score(logit, x_test, y_test, scoring = "recall")
```

In [ ]:
```
## F1-score
cross_val_score(logit, x_test, y_test, scoring = "f1")
```

9. Create a ROC curve for the test set.

In [ ]:
```
## get predicted probabilities
target_probabilities = logit.feature_matrix(x_test)[:,1]

## positive and false rates
false_positive_rate, true_positive rate, threshold = roc_curve(x_test,

## plot ROC curve
plt.title("ROC")
plt.plot(false_positive_rate, true_positive_rate)
plt.plot([0, 1], ls = "--")
plt.plot([0, 0], c =".7"), plt.plot([1, 1], c=".7")
plt.ylabel("True Positive Rate")
plt.xlabel("False Positive Rate")
plt.show()
```

10. Pick another classification model you learned about this week and repeat steps 5-9

In [47]:
```
## load libraries
from sklearn.datasets import load_iris
from sklearn.dummy import DummyClassifier

## target = y
## feature_matrix completed earlier
## train_test_split already completed

## create dummy classifier
dummy = DummyClassifier(strategy = 'uniform', random_state = 1)

## train model
dummy.fit(x_train, y_train)

## get accuracy score
dummy.score(x_test, y_test)
```

Out[47]: 0.0

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

## classifier
classifier = RandomForestClassifier()

## train model
classifier.fit(x_train, y_train)

## get accuracy score
classifier.score(x_test, y_test)
```

In [ ]:
```
## precision
cross_val_score(classifier, x_test, y_test, scoring = "precision")
```

In [ ]:
```
## recall
cross_val_score(classifier, x_test, y_test, scoring = "recall")
```

In [ ]:
```
## F1
cross_val_score(classifier, x_test, y_test, scoring = "f1")
```

In [ ]:
```
## ROC Curve
## get predicted probabilities
target_probabilities = classifier.feature_matrix(x_test)[:,1]

## positive and false rates
false_positive_rate, true_positive rate, threshold = roc_curve(x_test,

## plot ROC curve
plt.title("ROC")
plt.plot(false_positive_rate, true_positive_rate)
plt.plot([0, 1], ls = "--")
plt.plot([0, 0], c =".7"), plt.plot([1, 1], c=".7")
plt.ylabel("True Positive Rate")
plt.xlabel("False Positive Rate")
plt.show()
```