Text Classification Assessment

Goal: Given a set of text movie reviews that have been labeled negative or positive

For more information on this dataset visit http://ai.stanford.edu/~amaas/data/sentiment/

Complete the tasks in bold below!

Task: Perform imports and load the dataset into a pandas DataFrame For this exercise you can load the dataset from `'../DATA/moviereviews.csv'`.

In [2]:

```
# CODE HERE
```

In [47]:

```
import numpy as np
import pandas as pd
```

In [48]:

```
df = pd.read_csv("D:\\Study\\Programming\\python\Python course from udemy\\Udemy - 2022 Python for Machine Learning & Data Science Masterclass\\20 - Naive Bayes Classification and Natural Language Processing\\31640132-moviereviews.csv")
```

In [49]:

```
df.head()
```

Out[49]:

	label	review
0	neg	how do films like mouse hunt get into theatres...
1	neg	some talented actresses are blessed with a dem...
2	pos	this has been an extraordinary year for austra...
3	pos	according to hollywood movies made in last few...
4	neg	my first press screening of 1998 and already i...

TASK: Check to see if there are any missing values in the dataframe.

In [50]:

```
#CODE HERE
df.isnull().sum()
```

Out[50]:

```
label      0
review    35
dtype: int64
```

In [8]:

Out[8]:

```
label       0
review     35
dtype: int64
```

TASK: Remove any reviews that are NaN

In [51]:

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

In [52]:

```
df.isnull().sum()
```

Out[52]:

```
label     0
review    0
dtype: int64
```

TASK: Check to see if any reviews are blank strings and not just NaN. Note: This means a review text could just be: "" or " " or some other larger blank string. How would you check for this? Note: There are many ways! Once you've discovered the reviews that are blank strings, go ahead and remove them as well. Click me for a big hint

In [53]:

```
df['review'].str.isspace().sum()
```

Out[53]:

```
27
```

In [18]:

Out[18]:

```
27
```

In [54]:

```
df[df['review'].str.isspace()]
```

Out[54]:

	label	review
57	neg	
71	pos	
147	pos	
151	pos	
283	pos	
307	pos	
313	neg	
323	pos	
343	pos	
351	neg	
427	pos	
501	neg	

	label	review
633	pos	
675	neg	
815	neg	
851	neg	
977	neg	
1079	neg	
1299	pos	
1455	neg	
1493	pos	
1525	neg	
1531	neg	
1763	neg	
1851	neg	
1905	pos	
1993	pos	

In [19]:

```

```

Out[19]:

	label	review
57	neg	
71	pos	
147	pos	
151	pos	
283	pos	
307	pos	
313	neg	
323	pos	
343	pos	
351	neg	
427	pos	
501	neg	
633	pos	
675	neg	
815	neg	
851	neg	
977	neg	
1079	neg	
1299	pos	
1455	neg	
1493	pos	
1525	neg	
1531	neg	
1763	neg	
1851	neg	

	label	review
1851	neg	
1905	pos	
1993	pos	

In [55]:

```
df1 = df[df['review'].str.isspace()]
df1
```

Out[55]:

	label	review
57	neg	
71	pos	
147	pos	
151	pos	
283	pos	
307	pos	
313	neg	
323	pos	
343	pos	
351	neg	
427	pos	
501	neg	
633	pos	
675	neg	
815	neg	
851	neg	
977	neg	
1079	neg	
1299	pos	
1455	neg	
1493	pos	
1525	neg	
1531	neg	
1763	neg	
1851	neg	
1905	pos	
1993	pos	

In [56]:

```
df = df[~df['review'].str.isspace()]
df
```

Out[56]:

	label	review
0	neg	how do films like mouse hunt get into theatres...
1	neg	some talented actresses are blessed with a dem...
2	pos	this has been an extraordinary year for austra...

	label	review
3	pos	according to hollywood movies made in last few...
4	neg	my first press screening of 1998 and already i...
...	...	...
1995	pos	i like movies with albert brooks , and i reall...
1996	pos	it might surprise some to know that joel and e...
1997	pos	the verdict : spine-chilling drama from horror...
1998	pos	i want to correct what i wrote in a former ret...
1999	pos	a couple of months ago , when i first download...

1938 rows × 2 columns

In [57]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1938 entries, 0 to 1999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   label   1938 non-null   object
 1   review  1938 non-null   object
dtypes: object(2)
memory usage: 45.4+ KB
```

In [22]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1938 entries, 0 to 1999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   label   1938 non-null   object
 1   review  1938 non-null   object
dtypes: object(2)
memory usage: 45.4+ KB
```

TASK: Confirm the value counts per label:

In [66]:

```
#CODE HERE
df['label'].value_counts()
```

Out[66]:

```
neg    969
pos    969
Name: label, dtype: int64
```

In [24]:

Out[24]:

```
pos    969
neg    969
Name: label, dtype: int64
```

EDA on Bag of Words

Bonus Task: Can you figure out how to use a CountVectorizer model to get the top 20 words (that are not english stop words) per label type? Note, this is a bonus task as we did not show this in the lectures. But a quick

stop words, per label type. Note, this is a bonus task as we did not show this in the lectures. But a quick cursory Google search should put you on the right path. Click me for a big hint

In [45]:

```
#CODE HERE
```

In [67]:

```
from  sklearn.feature_extraction.text import CountVectorizer
```

In [69]:

```
cv = CountVectorizer(stop_words='english')
```

In [73]:

```
matrix = cv.fit_transform(df[df['label']=='neg']['review'])
freqs = zip(cv.get_feature_names_out(), matrix.sum(axis=0).tolist()[0])
# sort from largest to smallest
print("Top 20 words used for Negative reviews.")
print(sorted(freqs, key=lambda x: -x[1])[:20])
```

```
Top 20 words used for Negative reviews.
[('film', 4063), ('movie', 3131), ('like', 1808), ('just', 1480), ('time', 1127), ('good'
, 1117), ('bad', 997), ('character', 926), ('story', 908), ('plot', 888), ('characters',
838), ('make', 813), ('really', 743), ('way', 734), ('little', 696), ('don', 683), ('does
', 666), ('doesn', 648), ('action', 635), ('scene', 634)]
```

In [ ]:

In [55]:

```
Top 20 words used for Negative reviews.
[('film', 4063), ('movie', 3131), ('like', 1808), ('just', 1480), ('time', 1127), ('good'
, 1117), ('bad', 997), ('character', 926), ('story', 908), ('plot', 888), ('characters',
838), ('make', 813), ('really', 743), ('way', 734), ('little', 696), ('don', 683), ('does
', 666), ('doesn', 648), ('action', 635), ('scene', 634)]
```

In [74]:

```
matrix = cv.fit_transform(df[df['label']=='pos']['review'])
freqs = zip(cv.get_feature_names_out(), matrix.sum(axis=0).tolist()[0])
# sort from largest to smallest
print("Top 20 words used for Positive reviews.")
print(sorted(freqs, key=lambda x: -x[1])[:20])
```

```
Top 20 words used for Positive reviews.
[('film', 5002), ('movie', 2389), ('like', 1721), ('just', 1273), ('story', 1199), ('good
', 1193), ('time', 1175), ('character', 1037), ('life', 1032), ('characters', 957), ('way
', 864), ('films', 851), ('does', 828), ('best', 788), ('people', 769), ('make', 764), ('
little', 751), ('really', 731), ('man', 728), ('new', 702)]
```

In [56]:

```
Top 20 words used for Positive reviews.
[('film', 5002), ('movie', 2389), ('like', 1721), ('just', 1273), ('story', 1199), ('good
', 1193), ('time', 1175), ('character', 1037), ('life', 1032), ('characters', 957), ('way
', 864), ('films', 851), ('does', 828), ('best', 788), ('people', 769), ('make', 764), ('
little', 751), ('really', 731), ('man', 728), ('new', 702)]
```

Training and Data

TASK: Split the data into features and a label (X and y) and then preform a train/test split. You may use whatever

settings you like. To compare your results to the solution notebook, use `test_size=0.20, random_state=101`

In [ ]:

```

```

In [76]:

```
from sklearn.model_selection import train_test_split

X = df['review']
y = df['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=10
1)
```

Training a Mode

TASK: Create a PipeLine that will both create a TF-IDF Vector out of the raw text data and fit a supervised learning model of your choice. Then fit that pipeline on the training data.

In [78]:

```
#CODE HERE
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
```

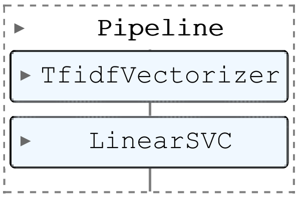
In [79]:

```
pipe = Pipeline([('tfidf', TfidfVectorizer()),('svc', LinearSVC()),])
```

In [80]:

```
# Feed the training data through the pipeline
pipe.fit(X_train, y_train)
```

Out[80]:



In [76]:

```

```

Out[76]:

```
Pipeline(steps=[('tfidf', TfidfVectorizer()), ('svc', LinearSVC())])
```

TASK: Create a classification report and plot a confusion matrix based on the results of your PipeLine.

In [84]:

```
#CODE HERE
from sklearn.metrics import classification_report,confusion_matrix
```

In [85]:

```
from mlxtend.plotting import plot_confusion_matrix
```

In [86]:

```
preds = pipe.predict(X_test)
```

In [87]:

```
print(classification_report(y_test,preds))
```

```
              precision    recall  f1-score   support

         neg       0.81      0.86      0.83       191
         pos       0.85      0.81      0.83       197

    accuracy                           0.83       388
   macro avg       0.83      0.83      0.83       388
weighted avg       0.83      0.83      0.83       388
```

In [80]:

```
```

```
              precision    recall  f1-score   support

         neg       0.81      0.86      0.83       191
         pos       0.85      0.81      0.83       197

    accuracy                           0.83       388
   macro avg       0.83      0.83      0.83       388
weighted avg       0.83      0.83      0.83       388
```

In [90]:

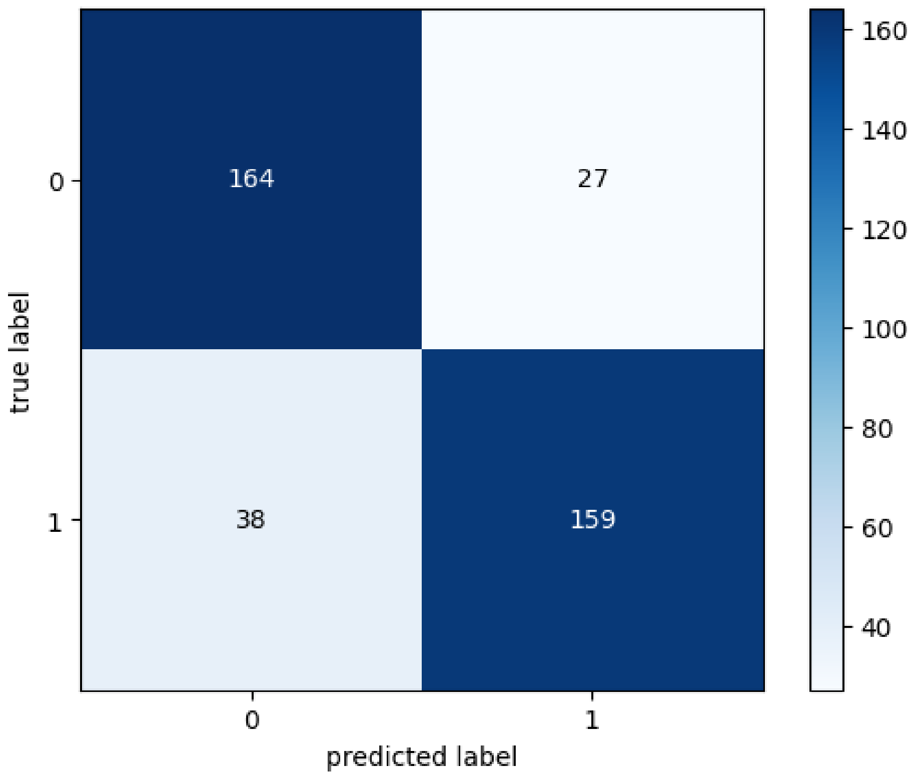
```
mm= confusion_matrix(y_test,preds)
mm
```

Out[90]:

```
array([[164,  27],
       [ 38, 159]], dtype=int64)
```

In [92]:

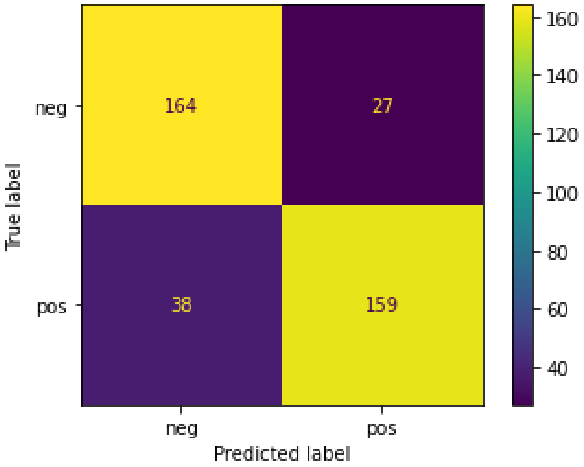
```
plot_confusion_matrix(mm,colorbar=True);
```



In [81]:

Out[81]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f370d0b790>
```



Great job!