(http://www.pieriandata.com)

Text Classification Assessment

This assessment is very much like the Text Classification Project we just completed, and the dataset is very similar.

The **moviereviews2.tsv** dataset contains the text of 6000 movie reviews. 3000 are positive, 3000 are negative, and the text has been preprocessed as a tab-delimited file. As before, labels are given as `pos` and `neg` .

We've included 20 reviews that contain either `NaN` data, or have strings made up of whitespace.

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

Task #1: Perform imports and load the dataset into a pandas DataFrame

For this exercise you can load the dataset from `'../TextFiles/moviereviews2.tsv'` .

In [1]:

```
import pandas as pd
import numpy as np

df = pd.read_csv('../TextFiles/moviereviews2.tsv', sep='\t')
df.head()
```

Out[1]:

	label	review
0	pos	I loved this movie and will watch it again. Or...
1	pos	A warm, touching movie that has a fantasy-like...
2	pos	I was not expecting the powerful filmmaking ex...
3	neg	This so-called "documentary" tries to tell tha...
4	pos	This show has been my escape from reality for ...

In [2]:

```
len(df)
```

Out[2]:

```
6000
```

Task #2: Check for missing values:

In [3]:

```
# Check for NaN values:
df.isnull().sum()
```

Out[3]:

```
label      0
review    20
dtype: int64
```

In [4]:

```
df.dropna(inplace = True)
```

In [5]:

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

Out[5]:

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

Task #3: Remove whitespace values:

In [11]:
```
blanks = []

for i, lb, rv in df.itertuples():
    if rv.isspace():
        blanks.append(i)

print(len(blanks))
```

```
0
```

Task #4: Take a quick look at the `label` column:

In [12]:
```
df['label'].head()
```

Out[12]:
```
0    pos
1    pos
2    pos
3    neg
4    pos
Name: label, dtype: object
```

In [23]:
```
df['label'].value_counts()
```

Out[23]:
```
pos    2990
neg    2990
Name: label, dtype: int64
```

Task #5: Split the data into train & test sets:

You may use whatever settings you like. To compare your results to the solution notebook, use `test_size=0.33, random_state=42`

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
X = df['review']
y = df['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33
```

Task #6: Build a pipeline to vectorize the date, then train and fit a model

You may use whatever model you like. To compare your results to the solution notebook, use `LinearSVC`.

In [18]:

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC

text_clf = Pipeline([('tfidf', TfidfVectorizer()), ('clf', LinearSVC())])

text_clf.fit(X_train, y_train)
```

Out[18]:

```
Pipeline(memory=None,
     steps=[('tfidf', TfidfVectorizer(analyzer='word', binary=False, d
ecode_error='strict',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='conten
t',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), norm='l2', preprocessor=None, smooth_idf=T
rue,...ax_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0))])
```

Task #7: Run predictions and analyze the results

In [19]:

```
# Form a prediction set
predictions = text_clf.predict(X_test)
```

In [20]:

```
# Report the confusion matrix
from sklearn.metrics import confusion_matrix

print(confusion_matrix(y_test, predictions))
```

```
[[900  91]
 [ 63 920]]
```

In [21]:

```
# Print a classification report
from sklearn.metrics import classification_report

print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

         neg       0.93      0.91      0.92       991
         pos       0.91      0.94      0.92       983

   micro avg       0.92      0.92      0.92      1974
   macro avg       0.92      0.92      0.92      1974
weighted avg       0.92      0.92      0.92      1974
```

In [22]:

```
# Print the overall accuracy
from sklearn.metrics import accuracy_score

print(accuracy_score(y_test, predictions))
```

```
0.9219858156028369
```

Great job!