Prepare Data

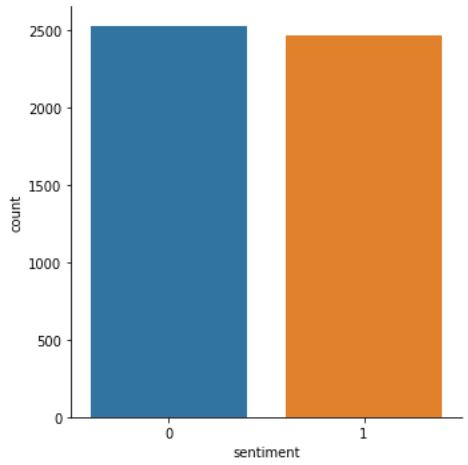
```
# set up
import nltk
from nltk.corpus import stopwords
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
import seaborn as sb

df = pd.read_csv('IMDB Dataset.csv')
stopwords = list(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwords, binary=True)
# set up X and y
X = vectorizer.fit_transform(df.review)
y = df.sentiment

#divide into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, random_state=1234)


#print out graph of distribution of class
sb.catplot(x="sentiment", kind="count", data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7ff8a5b0b250>
```



Dataset explanation -

This is a dataset compiled of imdb reviews that can either be categorized as positive (1) or negative (0). This means that the model should be able to predict the basic sentiment of a review as either a postive one or a negative on. We can see from the plot above that there are almost an equal amount of positive and negative reviews in the training data. This means that it should not be skewed necessarily in any particular direction.

Naive Bayes

```
#  naive bayes
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.naive_bayes import MultinomialNB

naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, y_train)

pred = naive_bayes.predict(X_test)
print(confusion_matrix(y_test, pred))
accuracy_score(y_test,·pred)
```

```
[[464  28]
 [140 366]]
0.8316633266533067
```

Logistic Regression

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

clf = LogisticRegression(C=2.5, n_jobs=4, solver='lbfgs', random_state=17, verbose=1)
clf.fit(X_train, y_train)
pred2 = clf.predict(X_test)
print(confusion_matrix(y_test, pred2))
accuracy_score(y_test, pred2)
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[[433  59]
 [ 56 450]]
[Parallel(n_jobs=4)]: Done   1 out of   1 | elapsed:    1.4s finished
0.8847695390781564
```

Neural Network

```
# neural network
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(15, 2), random_state=1)
classifier.fit(X_train, y_train)
pred3 = classifier.predict(X_test)
print(confusion_matrix(y_test, pred3))
accuracy_score(y_test, pred3)
```

```
[[421  71]
 [ 63 443]]
0.8657314629258517
```

Analysis:

If we look at the accuracy scores for all three algorithms on the same training dataset, we can see that the top performer in terms of accuracy is suprisingly Logisitc regression at 88.4%. The next best performer was the neural network at 86.5% and finally the naive bayes had the least accuracy at 83.1%. This is rather suprisingly because in most cases neural networks is usually the strongest performer, but here is was only the second best performer.

It is important to note that without preprocessing the data (as I had done in earlier attempts), Neural networks had the worst performance by a large amount with the accuracy being around 40%. After only preprocessing the data, that accuracy shot up to over double at over 80%, indicating the effects of leaving stopwords inside each review. Yet on the other hand the perfornmances of the other two alorightms decreased slightly after preprocessing the data and removing stop words fromt eh reviews.

Interstingly if we look at the confusion matrix produced for each alorightm, the number of incorrect positives and negatives (aka false positives and false negatives), the latter two algorithms have relatively similar amounts, but the naive bayes has a drastic change between the FP and FN values with the FP value being just over 1/7th the FN value. This indicates that the training set may be swekwed to haveing more negative reviews than positive ones.

✓ 8s completed at 1:55 PM