Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
from google.colab import files
```

```
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
```

```
    read_reply_from_input(message_id, timeout_sec)
```

```
!unzip imdb-dataset-of-50k-movie-reviews.zip
```

```
    ---> 97       time.sleep(0.025)
```

```
import pandas as pd

df = pd.read_csv('IMDB Dataset.csv')
```

```
df
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
...	...	...
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

50000 rows × 2 columns

```
# get cols of the df
df.columns
```

```
Index(['review', 'sentiment'], dtype='object')
```

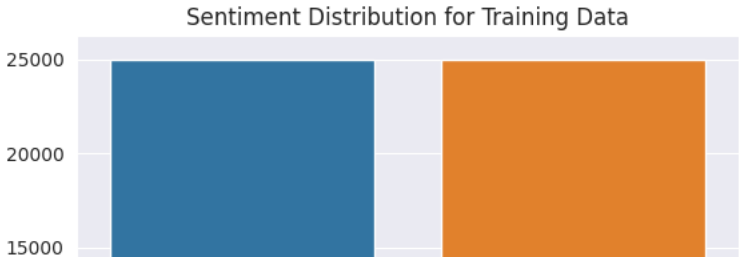
```
# sentiment is the y value, review is the x value. We split with an 80/20 split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df['review'],
                                                    df['sentiment'],
                                                    test_size=0.2,
                                                    random_state=42)
```

```
import seaborn as sns

sns.set_style('darkgrid')
sns.countplot(x='sentiment', data=df).set(title='Sentiment Distribution for Training Data')
```

```
[Text(0.5, 1.0, 'Sentiment Distribution for Training Data')]
```



Describe the data set and what the model should be able to predict.

This data set contains IMDB reviews that have a sentiment assigned to them - either positive or negative based on the review's content.

The model should be able to predict accurately the sentiment of a review given the review's content. There is an 80/20 split between training/testing data for the model to predict on.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import CountVectorizer

# vectorize the text data
vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# make a Naive Bayes model - no smoothing
nb_model = MultinomialNB()
nb_model.fit(X_train_vec, y_train)

# make predictions on the test data
y_pred = nb_model.predict(X_test_vec)

# get accuracy
accuracy = accuracy_score(y_pred, y_test)
print("Naive Bayes Accuracy without Laplace Smoothing: " , accuracy)

# make a Naive Bayes model - with Laplace smoothing
nb_model = MultinomialNB(alpha=100)
nb_model.fit(X_train_vec, y_train)

# make predictions on the test data
y_pred = nb_model.predict(X_test_vec)

# get accuracy
accuracy = accuracy_score(y_pred, y_test)
print("Naive Bayes Accuracy with Laplace Smoothing: " , accuracy)
```

```
Naive Bayes Accuracy without Laplace Smoothing:  0.8488
Naive Bayes Accuracy with Laplace Smoothing:  0.8343
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

# normalize the features
scaler = StandardScaler(with_mean=False)
X_train_vec_scaled = scaler.fit_transform(X_train_vec)
X_test_vec_scaled = scaler.transform(X_test_vec)

# make a Logistic Regression model - 500 iterations
clf = LogisticRegression(random_state=0, max_iter=110).fit(X_train_vec_scaled, y_train)

# make predictions on the test data
y_pred = clf.predict(X_test_vec_scaled)

# get accuracy
accuracy = accuracy_score(y_pred, y_test)
print("Logistic Regression Accuracy - 110 iterations: " , accuracy)

# The following gave the same results; the number of iterations were used to get a local
# minimum # of iterations required for convergence:
# 100 - too few, no convergence, warning given
```

```
# 101 - too few
# 110 - converged
```

```
Logistic Regression Accuracy - 110 iterations:  0.8662
```

```
from sklearn.neural_network import MLPClassifier

# make a Neural Network classifier
clf = MLPClassifier(hidden_layer_sizes=(75,), max_iter=1000, alpha=0.0001, random_state=42)
clf.fit(X_train_vec, y_train)

# make predictions on the test data
y_pred = clf.predict(X_test_vec)

# get accuracy
accuracy = accuracy_score(y_pred, y_test)
print("Neural Network Accuracy: ", accuracy)

# This took 29 minutes and 6 seconds
```

```
Neural Network Accuracy:  0.8877
```

Analysis

I tried three different approaches when it came to classification of movie review sentiment - Naive Bayes, Logistic Regression, and Neural Networks under the sklearn library.

In order to use the text data for classification, I used a tokenizer to convert the words accordingly for numerical input.

From my analysis, all three models performed fairly well (with >80% accuracy) in increasing order of accuracy. I'll discuss these in detail below.

Naive Bayes

Naive Bayes was the simplest model and I used two accuracies - with and without laplace smoothing (set at 100). The accuracies are shown below:

- Naive Bayes Accuracy without Laplace Smoothing: 0.8488
- Naive Bayes Accuracy with Laplace Smoothing: 0.8343

As can be seen, the accuracy is lower the higher Laplace smoothing is. This can happen if the dataset is not sparse, i.e., if most of the features occur frequently in the training set (which it does, as there are only two values when it comes to analysis).

Logistic Regression

Logistic Regression was the next model; unlike Naive Bayes, I only changed the number of max iterations before convergence. The accuracies are shown below:

- Logistic Regression Accuracy - 110 iterations: 0.8662

An interesting issue I noted when I was first testing out this model is that convergence wasn't reached by the default number of max iterations which was 100. Initially, I decided to set it very high at around 500 so I could achieve convergence. But once I got the same accuracy at 250, I decided to keep lowering the number of epochs until the accuracy remained the same, while convergence could still occur. In the end, 110 max iterations was my rough estimate, and it yielded a higher accuracy than Naive Bayes.

Neural Network - 75 layers, 1000 max iterations, 0.0001 learning rate

Neural Network took by and far the longest time of any of the other models, and for good reason - it's a very complex model that doesn't rely on pure regression alone. But it still yielded the highest accuracy of any of the other models. The accuracy is below:

- Neural Network Accuracy: 0.8877

Even though it took the longest time and had the most complex structure, it performed very well for the time invested, and it performed the best of any other model. Plus, neural networks are generally best for classification, so there is that as well.

Conclusion

Overall, I had a good time working with the different models to try and figure out what was going on with each during this process.