IMDB Sentiment analysis

Problem Statement:

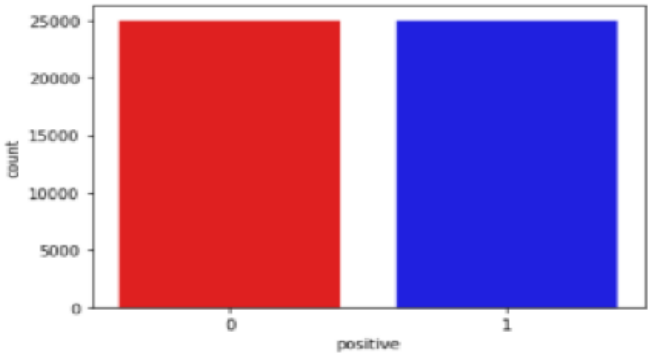
Sentiment analysis is an important technique for natural language processing that can help businesses interpret a consumer's perception of a product, service or brand. Businesses can benefit from sentiment analysis as it can help detect polarity from consumer response (good vs bad), sort your data, and identify trends in real-time. A common example of using sentiment analysis is on customer reviews. A related technique of sentiment analysis is intent classification where machine learning attempts to understand the intent behind a response. This was the goal behind the sentiment analysis of the IMDB dataset. How accurately could we predict whether a review is positive or negative based on the wording? The dataset is available from several resources; this dataset was retrieved from the Kaggle website.

Data cleaning and EDA:

The original dataset contained 50,000 rows and 2 columns. The two columns, 'review' and 'sentiment', are both type object. We needed to predict the 'sentiment' column, so an extra column was created with dummy variables so that we could have type integer. There were no missing values and positive and negative values were balanced.

```
[10] sns.countplot(x='positive', data=data, palette = ["red","blue"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f32a4bfffd0>
```




```
[11] data.review_length[data["sentiment"] == "positive"].describe()
```

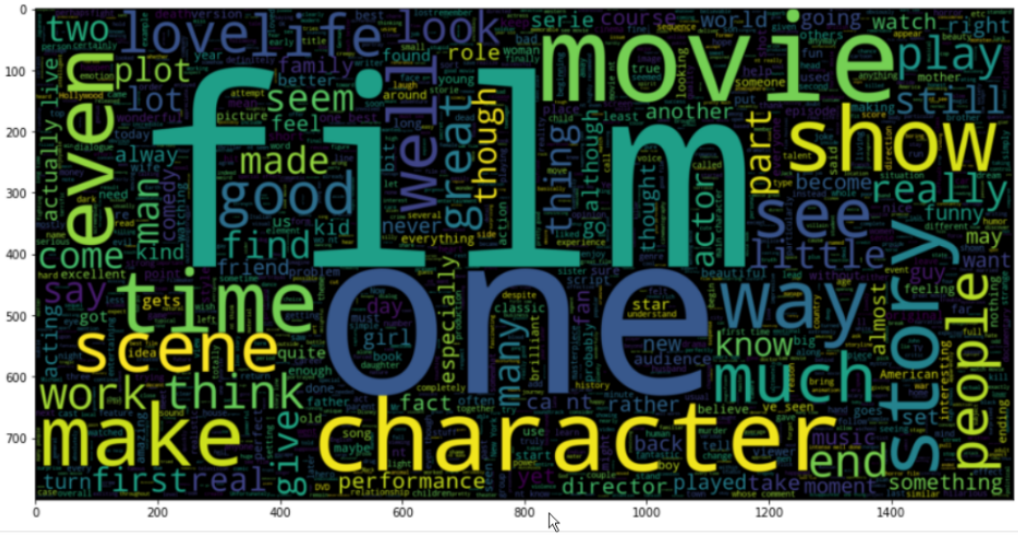
```
count    25000.000000
mean       232.849320
std        177.497046
min         10.000000
25%        125.000000
50%        172.000000
75%        284.000000
max       2470.000000
Name: review_length, dtype: float64
```

Before any modeling can be done, the text from the review section needed to be cleaned. This includes removing HTML tags, stopwords, and punctuation. The HTML and punctuation needed to be removed so that the text can be used in the ML models.

The purpose of removing the stopwords is so that the models don't use those words to predict the sentiment. Before removing them, the three most common word among the reviews was 'the'. Aftwards, the most common word became 'movie'. Further analysis shows the common words between positive and negative reviews

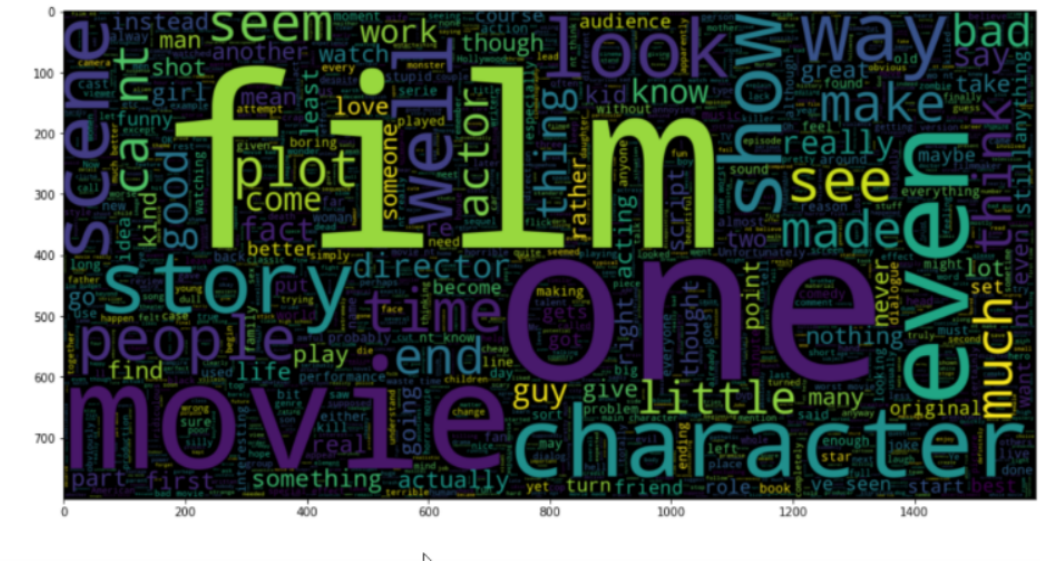
```
plt.figure(figsize = (15,12))
wc_pos = WordCloud(max_words = 1500, width = 1600 , height = 800).generate(" ".join(data[data.sentiment == 'positive'].clean_review))
plt.tight_layout(pad=0) #shrink the size of the border
plt.imshow(wc_pos, interpolation = 'bilinear')
#displayed image appear more smoothly
```

```
<matplotlib.image.AxesImage at 0x7f32441559d0>
```




```
[27] plt.figure(figsize = (15,12)) # Negative Review Text
     wc_neg = WordCloud(max_words = 2000, width = 1600 , height = 800).generate(" ".join(data[data.sentiment == 'negative'].clean_review))
     plt.imshow(wc_neg, interpolation = 'bilinear')
```

```
<matplotlib.image.AxesImage at 0x7f322783a050>
```



Modeling:

Two ML models were used for this project. The first was a simple Decision Tree Classifier. Due to component limitations, only 2000 rows were trained and evaluated. After the model was tested, the model accuracy score was 0.6425.

The second model used was BERT. BERT (Bidirectional Encoder Representations from Transformers), is a pretrained, transformer-based machine learning model developed by Google in 2018. Transformers use an attention mechanism to process all tokens at the same time. This helps determine the significance of each input. BERT essentially trains by randomly masking words and predicting them based on the context. The model is made to learn from words in all positions. These are better than traditional NLP models that use word embedding and are bad at interpreting context, as well as other models such as LSTM (Long short-term memory) that is only trained by predicting from left to right.

There are several BERT Models to choose from. The one I chose was small BERT which has less transformers than the original BERT, but trains faster.

Epoch 1/3 525/525	140s 249ms/step	loss: 0.4566 binary_accuracy: 0.7616 -	val_loss: 0.3442 val_binary_accuracy: 0.8508
Epoch 2/3 525/525	132s 251ms/step	loss: 0.3017 binary_accuracy: 0.8693	val_loss: 0.3226 val_binary_accuracy: 0.8524
Epoch 3/3 525/525	132s 252ms/step	loss: 0.2299 binary_accuracy: 0.9064	val_loss: 0.3450 val_binary_accuracy: 0.8642

Test Set:

188/188	19s 99ms/step	loss: 0.3404	binary_accuracy: 0.8662

Conclusion:
The BERT Model was a significant improvement compared to the Decision Tree Classifier in terms of accuracy. This was to be expected. For future research, I would compare extra models such as random forest and other BERT models. I would also change some of the hyperparameters such as adding more epochs. It is likely however

that the loss will increase as well. I would also train on a larger dataset. Although this dataset contained 50,000 rows, I was computationally limited to using 30,000.