Final Project Report:

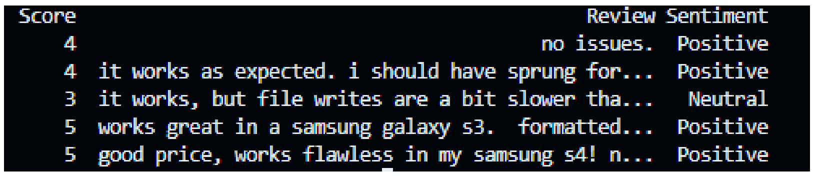
LSTM Sentiment Model

Summary

This project aims to develop a sentiment analysis model using LSTM neural networks to predict the sentiment of given text as either positive, negative, or neutral. The model was trained on over 75,000 Amazon reviews from three Kaggle datasets, which were preprocessed and cleaned. The model architecture includes an input layer, an LSTM layer with 128 units, a dense layer with 64 units, a dropout layer with a rate of 0.5, and a dense output layer with three units for the three classes of sentiment. After fine-tuning, the model achieved an accuracy of 80.63%. This model can be used to understand customer opinions on specific aspects of products and analyze sentiment in other contexts. Future iterations of this model could include the use of more complex models or the incorporation of more data to increase the accuracy of the model.

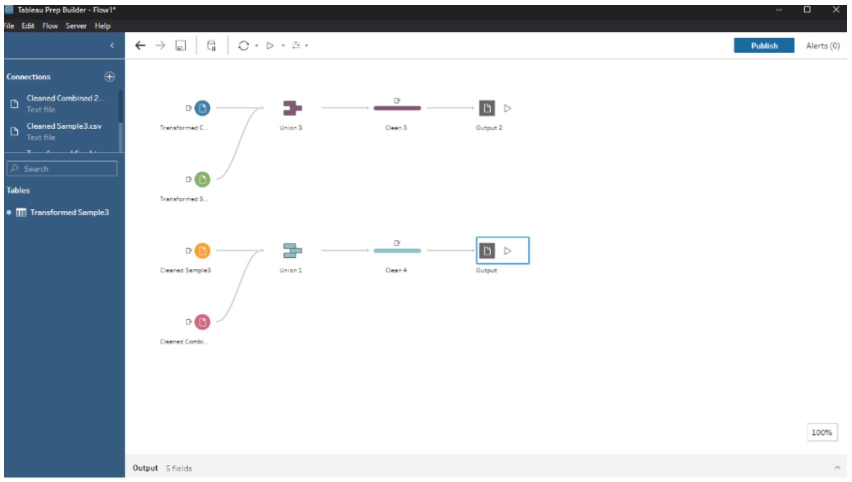
Data Cleaning

a. This step is very crucial when it comes to creating your own sentiment analysis model.
b. It is important that the columns we do not need are eliminated from the dataset so we can just focus on the important aspects.
c. For reference, the code used to clean the datasets is titled *"Data Cleaning.py"* in the repository.
d. After cleaning the dataset, the structure of the data set should look like the figure below.



```
Score                                              Review Sentiment
    4                                          no issues.  Positive
    4  it works as expected. i should have sprung for...  Positive
    3  it works, but file writes are a bit slower tha...   Neutral
    5  works great in a samsung galaxy s3.  formatted...  Positive
    5  good price, works flawless in my samsung s4! n...  Positive
```

e. Once each data set is cleaned, we take those into Tableau Prep and combine all three data sets.

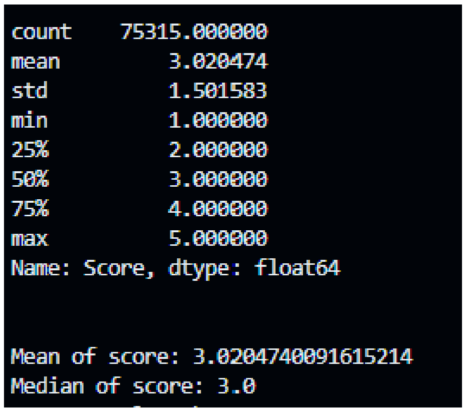


f. Next, we then clean the distribution to better balance the data.
g. After all this is complete, we can then move on to analyzing the cleaned and combined sample.

Descriptive Analysis

In our descriptive analysis we will be looking to analyze the mean and median of the score, the text length as important details can be presented regarding the input set for our model, and a word cloud to see what possible category of review we are seeing mainly, or the common words being presented.

a. To start, a general analysis must be run on the Score so we can see that are of importance are the count, mean, and median. What results is shown below.

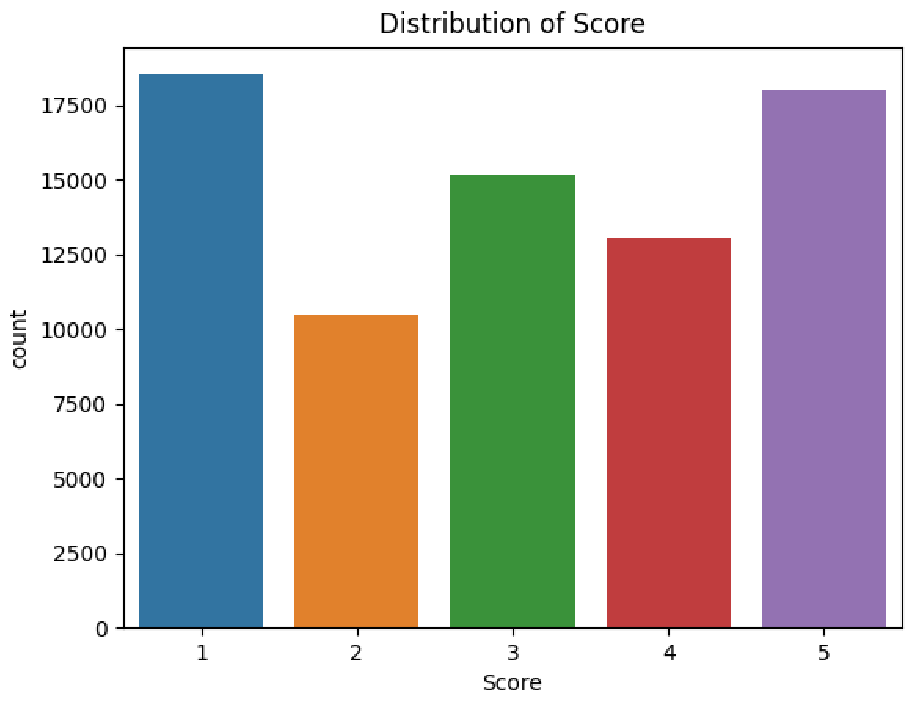


```
count    75315.000000
mean         3.020474
std          1.501583
min          1.000000
25%          2.000000
50%          3.000000
75%          4.000000
max          5.000000
Name: Score, dtype: float64


Mean of score: 3.0204740091615214
Median of score: 3.0
```

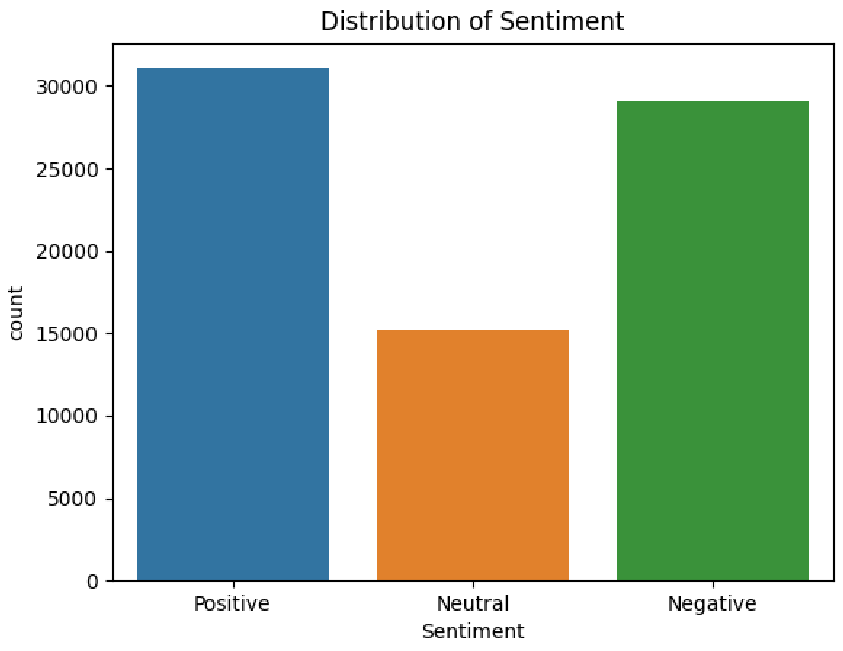
b. Next, a word cloud is created with inspiration from a Kaggle participant that allows us to identify key words and possibly understand what category or product the reviews are stemming from.

Here we can see that this seems that food products or coffee products from Amazon are prevalent and seems to carry a positive generality.

c. The next step in our analysis involves plotting the distribution for the score and sentiment.

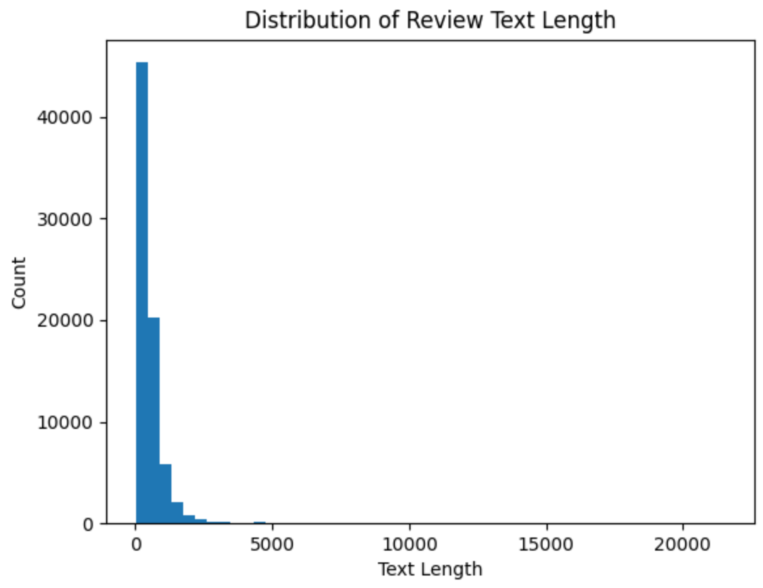


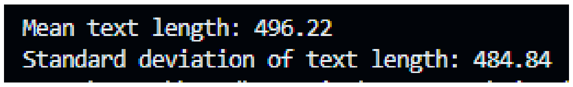
There is almost an even score distribution but, we do have to consider that scores 1 and 2 will count as negative while 4 and 5 will become positive which can throw the distribution off so further score distribution cleaning may or may not be done.



This distribution is not too compromising so we can continue.

d. Reviewing the length of text will be the next step in our analysis that way we can set an input length limit in our LSTM model.





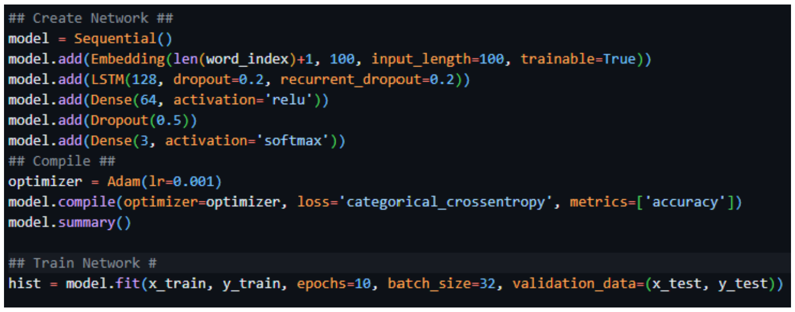
```
Mean text length: 496.22
Standard deviation of text length: 484.84
```

Through this distribution we can see count and text length. This distribution also shows that reviews are not going over 5000 words and typically have a mean of about 500 words so setting the input limit to 1000 will not cause issues.

Predictive Analysis

Here is where the LSTM model is created utilizing various libraries such as sklearn, TensorFlow, pandas, pickle, keras, pandas, and NumPy.

a. LSTM creation
   i. Create a DataFrame from the cleaned sample.
   ii. Run the .astype(str) on the text column to make sure the text is read as string.
   iii. Convert the text to a sequence of integers using tokenizer, sequences, and padded sequences.
   iv. Dummies are created for the sentiment labels, then we can proceed with splitting the dataset into training and testing.
   v. The model creation is next where we can specify units, parameters, and layers for the model. Model pictured below.



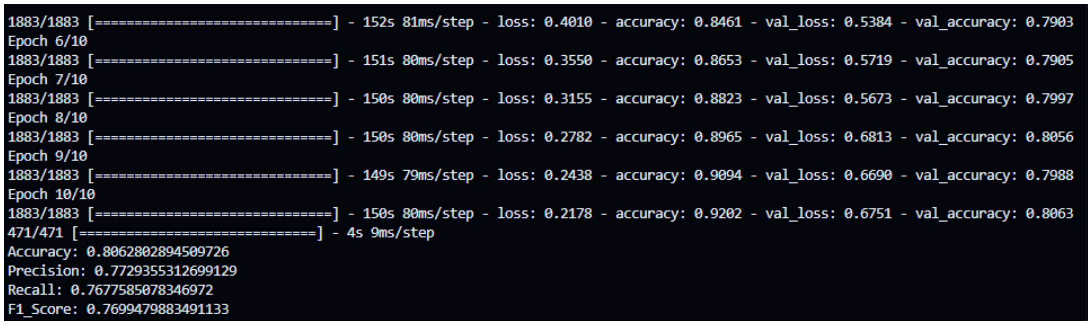
```
## Create Network ##
model = Sequential()
model.add(Embedding(len(word_index)+1, 100, input_length=100, trainable=True))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
## Compile ##
optimizer = Adam(lr=0.001)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

## Train Network #
hist = model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_test, y_test))
```

   vi. Once all this is set, we can run and evaluate the model using the general machine learning model principles.

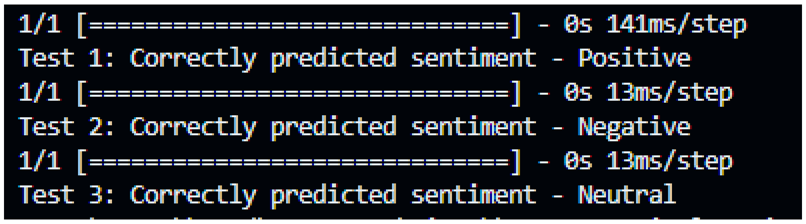
b. Model Evaluation
   i. In our evaluation we will be looking at four metrics: Accuracy, Precision, Recall, and F1 Score.
   ii. The model will be run on 10 epochs and should return to us the metrics we want.
   iii. Pickle will also be used to save the model and allow us to test it against our own text for future use cases.
   iv. Model Evaluation Results

Our model returns an accuracy of 80.63% which is pretty good and something you may see returned in other models. Precision, Recall, and F1 Score all hover around the same score. Overall, model looks like something we can test new text against.

c. Testing model against new text



```
1883/1883 [==============================] - 152s 81ms/step - loss: 0.4010 - accuracy: 0.8461 - val_loss: 0.5384 - val_accuracy: 0.7903
Epoch 6/10
1883/1883 [==============================] - 151s 80ms/step - loss: 0.3550 - accuracy: 0.8653 - val_loss: 0.5719 - val_accuracy: 0.7905
Epoch 7/10
1883/1883 [==============================] - 150s 80ms/step - loss: 0.3155 - accuracy: 0.8823 - val_loss: 0.5673 - val_accuracy: 0.7997
Epoch 8/10
1883/1883 [==============================] - 150s 80ms/step - loss: 0.2782 - accuracy: 0.8965 - val_loss: 0.6813 - val_accuracy: 0.8056
Epoch 9/10
1883/1883 [==============================] - 149s 79ms/step - loss: 0.2438 - accuracy: 0.9094 - val_loss: 0.6690 - val_accuracy: 0.7988
Epoch 10/10
1883/1883 [==============================] - 150s 80ms/step - loss: 0.2178 - accuracy: 0.9202 - val_loss: 0.6751 - val_accuracy: 0.8063
471/471 [==============================] - 4s 9ms/step
Accuracy: 0.8062802894509726
Precision: 0.7729355312699129
Recall: 0.7677585078346972
F1_Score: 0.7699479883491133
```

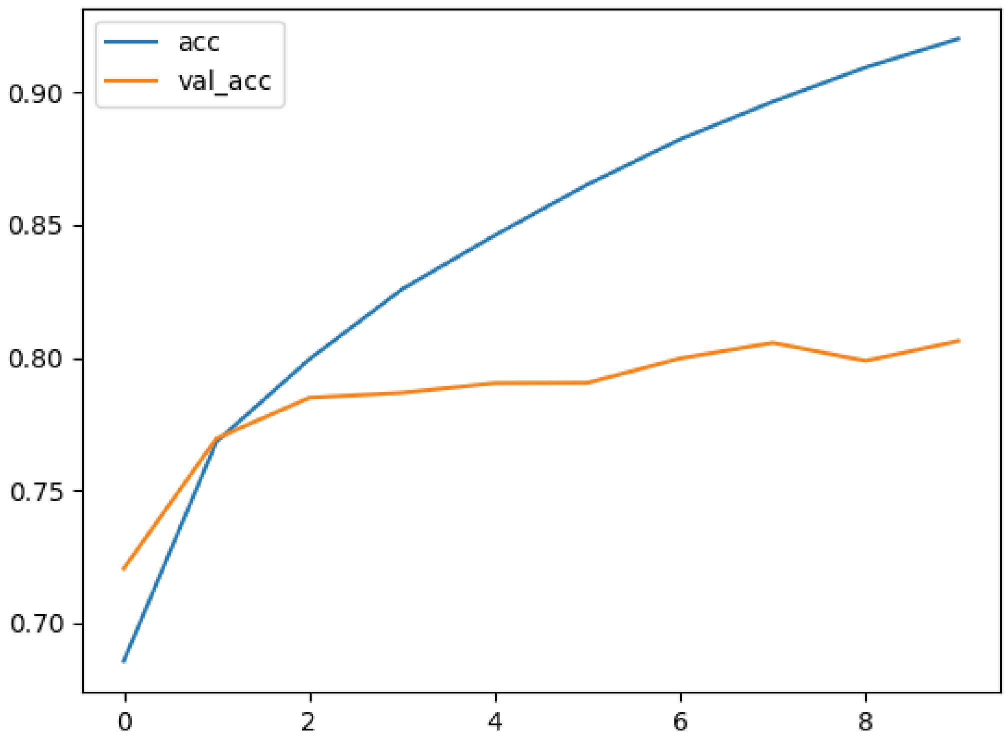


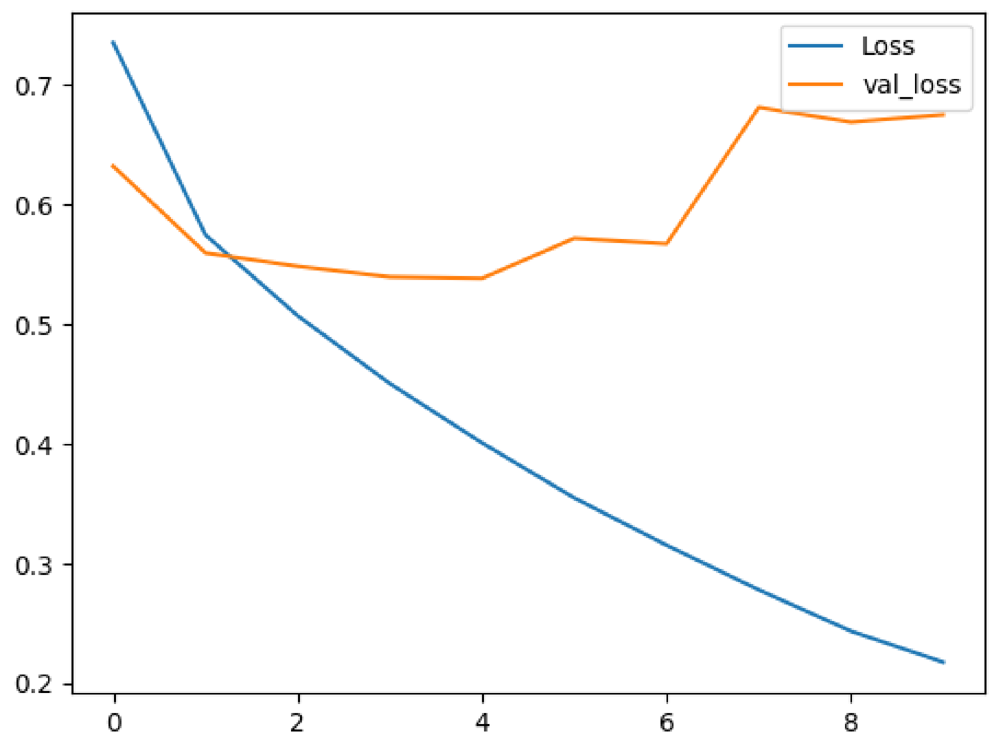
```
1/1 [==============================] - 0s 141ms/step
Test 1: Correctly predicted sentiment - Positive
1/1 [==============================] - 0s 13ms/step
Test 2: Correctly predicted sentiment - Negative
1/1 [==============================] - 0s 13ms/step
Test 3: Correctly predicted sentiment - Neutral
```

We test the model against new text and return correct predictions of sentiment.

While this looks promising there is still much that can be improved to adjust this model accordingly and refine it to be even better.

e. Accuracy and Loss Charts


acc
val_acc
0.90
0.85
0.80
0.75
0.70
0
2
4
6
8



Findings

There are several positive takeaways regarding the potential applications of this model. When it comes to predicting sentiment, we can expect it to perform well even with the introduction of new text. The model's accuracy rating of 80.63% is a good starting point and there are still several machine learning techniques available to enhance it, such as adding more layers to the model and fine-tuning hyperparameters. Additionally, the model can benefit from the inclusion of additional data through data mining. It is very important to note that when running this model on large datasets, computing power is a factor to consider.

Overall, this model can be leveraged by companies to predict the sentiment of product reviews, leading to an improvement in product quality. Furthermore, with slight modifications, the model can be applied to other areas of business to target areas for improvement. This model proves to be one that can be utilized on many fronts and implementations of it can vary greatly for a company's use case.