TASK 1 – NATURAL LANGUAGE PROCESSING USING A RECURRENT NEURAL NETWORK

PDAN8412

SEPTEMBER 14, 2022

SEAN TEUCHERT
ST10041890

Contents

1. Introduction:

The dataset that has been selected for analysis and to build a natural language processing model consists of Amazon reviews. The data was recorded over a span of 18 years, including 35 million reviews up to March 2013. This information was recorded from the Stanford Network Analysis Project (SNAP). This dataset contains 1.8 million records specifically about product reviews and the reviews sentiment, it is a subset of the 35 million reviews dataset. Amazon receives large volumes of data on a daily basis, it will physically be impossible for one person or a group of people to collect and analyze all its customer feedback. A Deep Learning model can be implemented to analyze the customer reviews and distinguish between a positive or negative review. Amazon can use this model to determine what reviews are negative and focus on improving on those specific negative reviews for example: a product supplier may be supplying bad quality products. This results in a lot of customers leaving a bad review about the product. The deep learning model can identify these reviews are negative and what key words are common among the reviews like “bad quality product”. This information can help Amazon solve their business problems faster and increase customer satisfaction.

Author of the Dataset: Kritanjali Jain

Dataset link: https://www.kaggle.com/datasets/kritanjalijain/amazon-reviews

Please note:

A Jupyter Notebook was used to develop the initial code, but I used Visual Studio Code to write the finalized code. The file types are the same therefore the code can be run in Jupyter Notebook or Visual Studio Code.

2. What is Natural Language Processing?

Natural Language Processing (NLP) is a field of Artificial Intelligence, it involves computers to analyze, understand and derive meaning from human language in a meaningful way (DataRobot, 2016). The goal of NLP is to organize and structure knowledge to perform tasks like sentiment analysis or speech recognition. An example of NLP is providing a machine learning model a dataset that includes text and the sentiment of the text (positive or negative comment). Classification or deep learning algorithms can be used to analyze the data and build a model. The model can be used on a new dataset or new text to identify whether it is a positive or negative comment.

3. What is Deep Learning?

Deep Learning is a type of machine learning and artificial intelligence that recreates the way humans gain certain types of knowledge (Burns & Brush, 2021). This process involves statistics and predictive modeling. In simple terms, Deep Learning aims to automate predictive analytics by making use of artificial neural networks.

3.1. What is a Recurrent Neural Network?

A recurrent neural network (RNN) is a type of artificial neural network that uses sequential data or time series data. In a traditional neural network, all the inputs and outputs are independent of each other. In an RNN, the output from the previous step is fed into the current step as an input (IBM Cloud Education, 2020). This concept is ideal for Natural Language Processing because each word in a sentence or sequence may hold meaning and influence the next word in the sentence or sequence.

3.2. What is Long Short-Term Memory algorithm?

Long Short-Term Memory (LSTM) is a type of neural network capable of learning order dependence in a sequence prediction problems (Brownlee, 2017). The algorithm consists of four neural networks and different memory blocks called cells. The information is retained by the cells and the memory manipulations are done by gates known as forget gate, input gate and output gate (Chugh, 2021). The information that is no longer useful in the cell state is removed by the forget gate. The addition of useful information to the cell state is performed by the input gate. The task of extracting useful information from the current cell state and produce an output is performed by the output gate.

4. Why is the Dataset appropriate for a Recurrent Neural Network?

A recurrent neural network requires sequential data or time series data to accurately make predictions. The selected dataset is of Amazon reviews which consists of text data and the sentiment of the text. This dataset is appropriate for an RNN because it consists of sequential data. Sequential data is data that has some form of sequence or order to it. Text data falls under this category as the human language is a set of sequences that require a specific order to make sense. To perform Natural Language processing using an RNN, it is required to train the model using text and sentiment. Sentiment refers to a view or opinion that is held or expressed. This dataset consists of Amazon reviews in the form of text data and the reviews sentiment as an integer (1 – Negative & 2 - Positive). The dataset consists of a balanced number of positive and negative reviews, this means that we do not need to use SMOTE. The dataset meets the two requirements for using an RNN for NLP therefore, it is an appropriate dataset.

5. What is the question that the analysis will answer?

The purpose of the analysis is to answer the following questions:

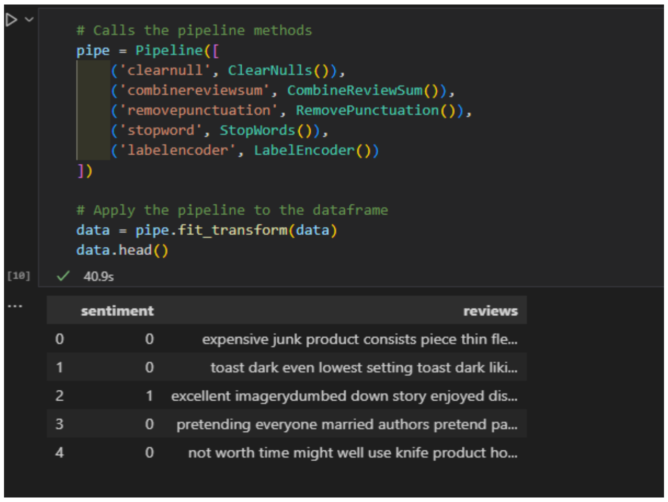
- Is it possible to identify sentiment from text specifically for Amazon product reviews?
- Can a machine learning model be used to categorize the reviews based on sentiment (positive or negative)?

6. Type of analysis performed:

- Descriptive Analysis has been performed to visualize the dataset
- Predictive Analysis using Recurrent Neural Networks (LSTM) has been used to build the model and predict sentiment from text data.

7. Exploratory Analysis and Data wrangling:

Before performing Exploratory Data Analysis (EDA), it is important to perform feature engineering or data wrangling. The reason for performing data wrangling before performing EDA is to remove data that is irrelevant to the problem domain, and it converts the relevant data into a format that is readable by EDA and machine learning model libraries. In this analysis, a pipeline was used to perform data wrangling. Figure 1 illustrates the pipeline call method:



```
# Calls the pipeline methods
pipe = Pipeline([
    ('clearnull', ClearNulls()),
    ('combinereviewsum', CombineReviewSum()),
    ('removepunctuation', RemovePunctuation()),
    ('stopword', StopWords()),
    ('labelencoder', LabelEncoder())
])

# Apply the pipeline to the dataframe
data = pipe.fit_transform(data)
data.head()
```

```
[10]  ✓ 40.9s
```

	sentiment	reviews
0	0	expensive junk product consists piece thin fle...
1	0	toast dark even lowest setting toast dark liki...
2	1	excellent imagerydumbed down story enjoyed dis...
3	0	pretending everyone married authors pretend pa...
4	0	not worth time might well use knife product ho...

Figure 1: Pipeline call method

Please refer to Jupyter Notebook to view each pipeline method in Figure 1

The pipeline performs the following tasks during cleaning; removes all null records, combines the review_title and review_text columns, removes all punctuation, removes stop words and one hot encodes the sentiment column. When removing stop words from the reviews, it is important to include certain stop words because it can change the meaning of a sentence e.g.: “This product is not worth it”. By removing the word “not”, it will change the interpretation of the sentence therefore, the data analyst has specified which stop words to remove. After the dataset has been cleaned, the reviews column is lemmatized using the WordNetLemmatizer from the NLTK library. When customers leave reviews on products, they tend to use different forms of the same words that ultimately have the same meaning. Lemmatization is used to remove inflectional endings (parts added to the end of a base word) and to return the base or dictionary form of a word.

To improve the data analysts understanding of the dataset, Exploratory data analysis has been performed before creating the model. A count plot has been used to determine the total number of positive reviews vs total number of negative reviews. This information can be used to identify if the dataset has balanced samples. A balanced dataset of positive and negative reviews will result in higher model accuracy. A box and whisker plot are used to identify the number of outliers in the review's length. A histogram is used to identify the distribution of review's length and identify if the outliers will become a problem later. Two Wordclouds have been generated to help identify words that are common among the positive and negative reviews. This gives the analyst an idea of what words will hold more weightings in the model. A Wordcloud can be miss leading due to certain words being commonly used but they do not make sense by themselves. This is where a N-gram can be used to put these types of word into some context. The data analyst has made use of a three-word N-gram to improve their understanding. An example in the Jupyter notebook is the word "one", it holds a weirdly high weighting in the Wordcloud but does not make much sense by itself. If we look at the N-gram, a common phrase used in the reviews is "one best book". This gives more context to the word as its used in the phrase "one of the best books".

8. Overfitting and Underfitting:

Overfitting refers to a model that models the training data too well or has been over trained with a large amount of data. This can become a problem because the model will not be able to correctly analyze a new dataset. Underfitting refers to a model that cannot capture the underlying trend of the data, this is a result of using the incorrect model to analyze the data or by not providing enough test data. To combat overfitting and underfitting, I have made use of a method called "early_stopping" from the Keras Library, it is a method that allows the user to specify a large number of training epochs and stop training once the model performance stops improving on a holdout validation dataset (Brownlee, 2020). This method will ensure that the model has enough training data to avoid underfitting while making sure the model does not over train itself and become overfitted. I have also made sure that my dataset has a balance number of positive and negative reviews.

9. Table and graph Explanation:

Figure 2 illustrates the count plot for each sentiment in the dataset:

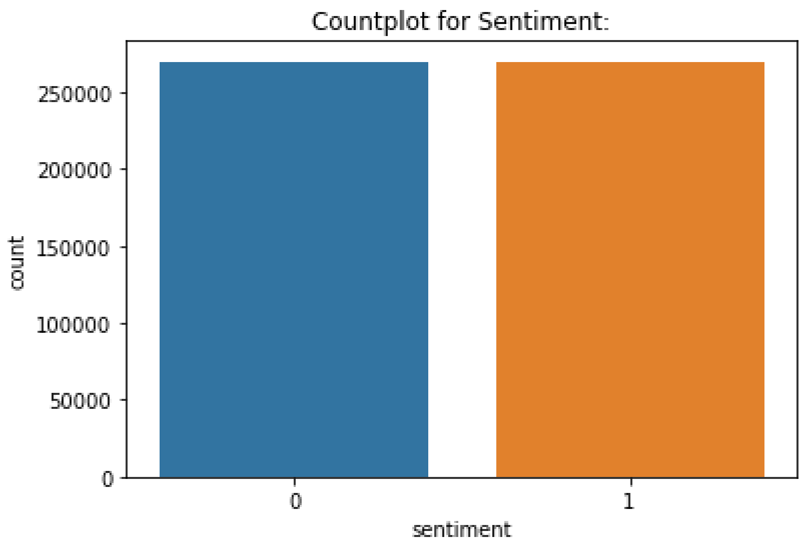


Figure 2: Count plot for Sentiment

This graph conveys the information about the total number of positive and negative reviews in the dataset. On the X-axis, the 0 refers to negative reviews and the 1 refers to positive reviews. This indicates that the dataset has a balance sample of negative and positive reviews. This information tells the analyst that SMOTE is not needed to rebalance the samples to combat overfitting.

Figure 3 illustrates a box and whiskers for review text length:

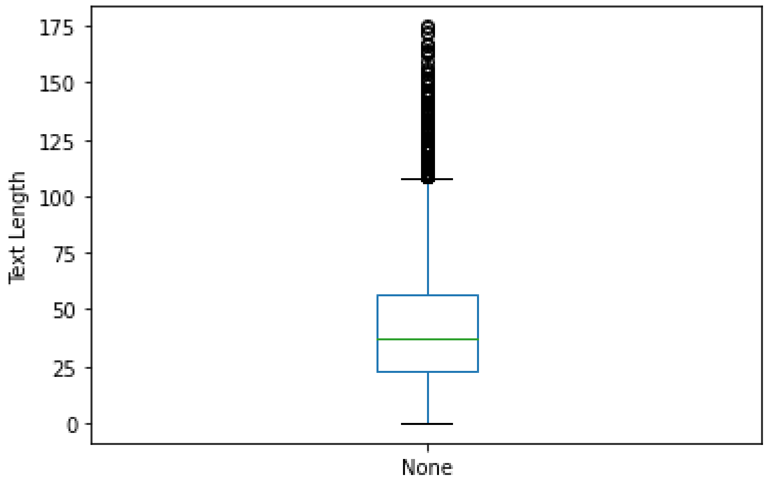


Figure 3: Box & Whisker for review text length

This graph conveys the information about the distribution of the reviews text length. It provides the analyst with information about any reviews that may be outliers due to the length of the review. It also tells the analyst that majority of the reviews fall within a text length smaller than 65 characters. This information is important because any outliers may result in poor model performance.

Figure 4 illustrates a histogram the frequency of review text length:

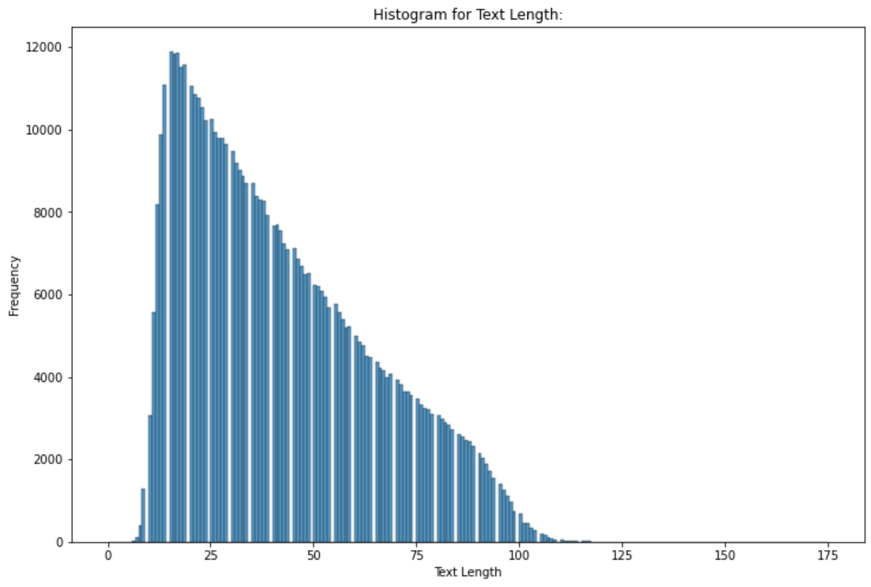


Figure 4: Histogram of review text length

This graph conveys a better picture of the distribution of the reviews by text length. The box and whiskers plot confirmed that the dataset contains outliers. The histogram gives the analyst a better idea of how many outliers there are.

Figure 5 illustrates a Wordcloud for positive reviews:

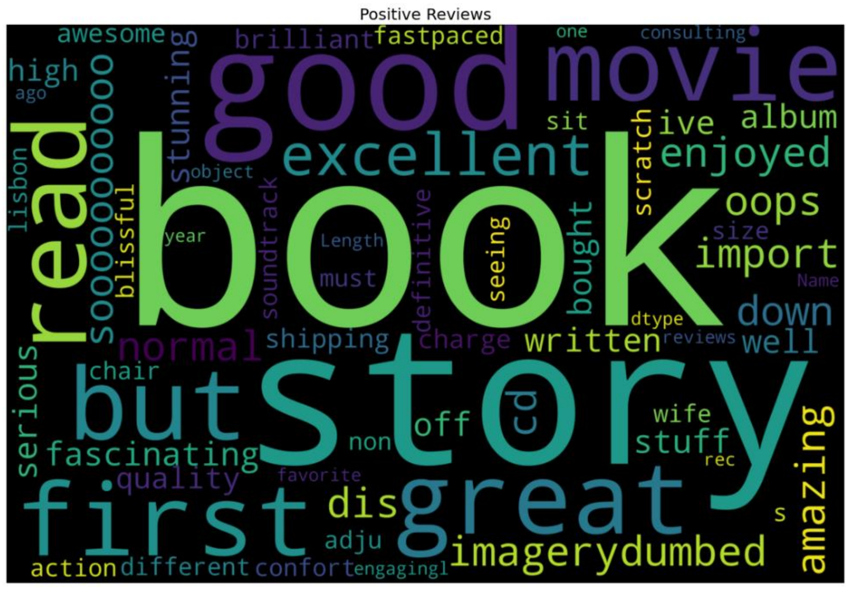


Figure 5: Wordcloud for positive reviews

This graph conveys the information about what words are frequently used in the positive reviews. It provides the analyst with information about the potential weightings each word may hold in the model’s algorithm.

Figure 6 illustrates a Wordcloud for negative reviews:

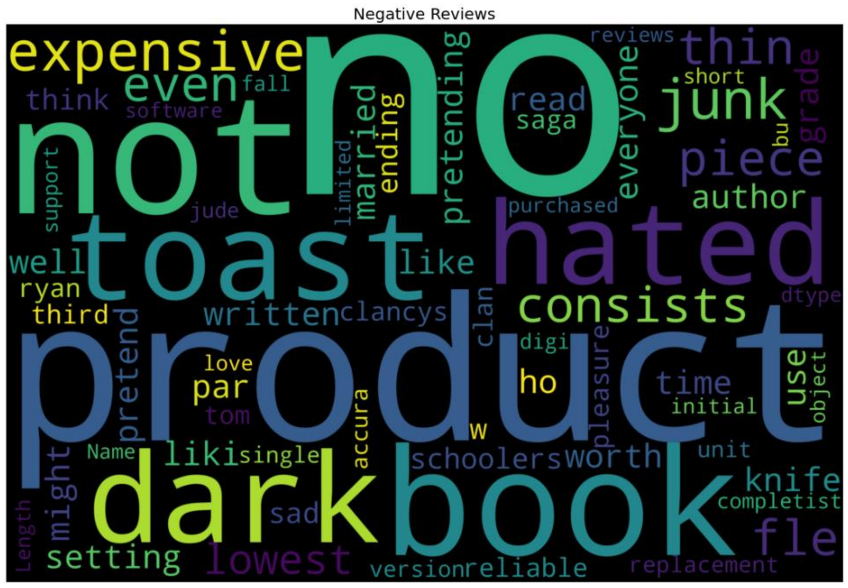


Figure 6: Wordcloud for negative reviews

This graph conveys the information about what words are frequently used in the negative reviews. It provides the analyst with information about the potential weightings each word may hold in the model's algorithm.

Figure 7 illustrates two N-grams for positive and negative reviews:

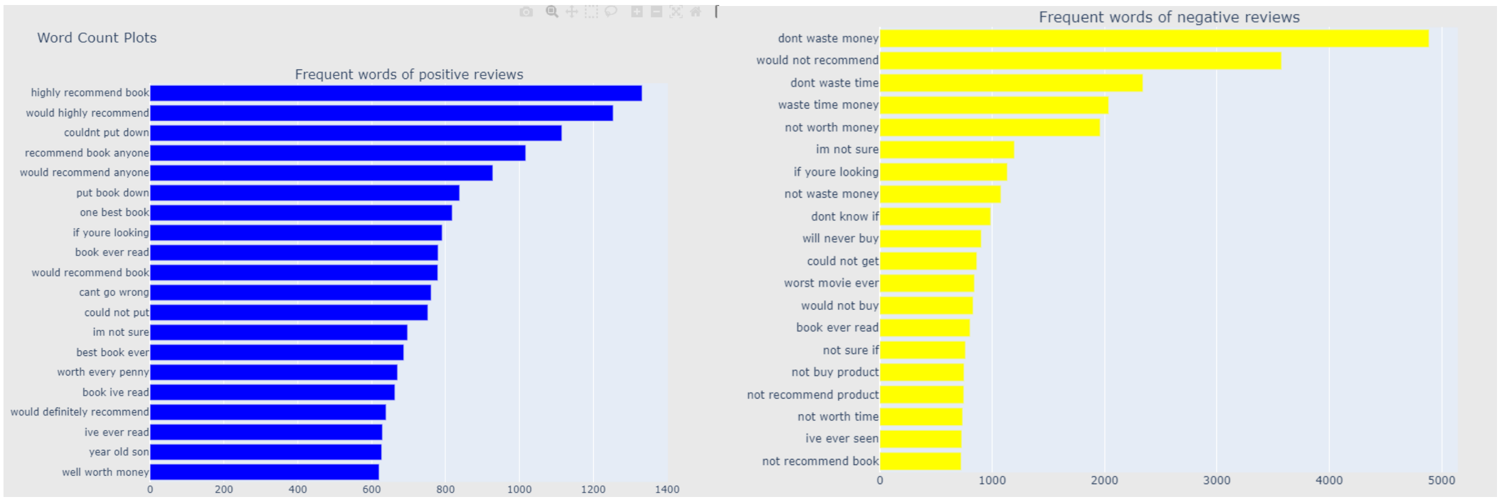


Figure 7: N-grams for positive and negative reviews

This graph conveys information about what words are commonly used together or as a phrase. This provides the analyst with a better idea of why certain words have a high frequency in the Wordcloud. An example is the word “not”, by itself it can be misleading but when looking at the N-gram it is clear that its commonly used to construct the phrase “Do not buy this product” or “I do not recommend this book”

Figure 8 illustrates a line graph for the model’s training and validation accuracy:

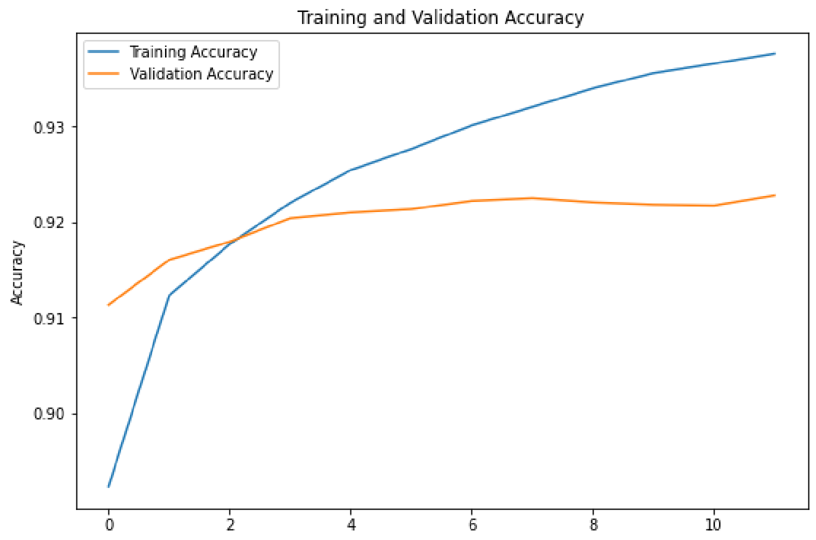


Figure 8: Line graph for model's training and validation accuracy

The line graph conveys information about how the training and validation accuracy changes per epoch. It provides the analyst with a visual representation of how the model’s training and validation data’s accuracy is improving over time. This also provides the analyst with a graphical representation of the model’s learning curve.

Figure 9 illustrates a line graph for model's training and validation loss:

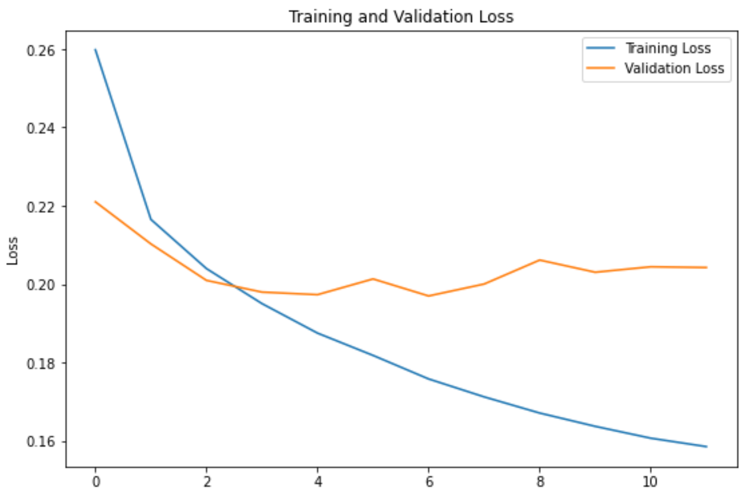


Figure 9: Line graph for model's training and validation loss

The line graph conveys information about how the training and validation loss changes per epoch. It provides the analyst with a visual representation of how the model's training and validation data's loss is decreasing over time. This also provides the analyst with a graphical representation of the model's learning curve.

Figure 10 illustrates a confusion matrix for the model's predicted outputs

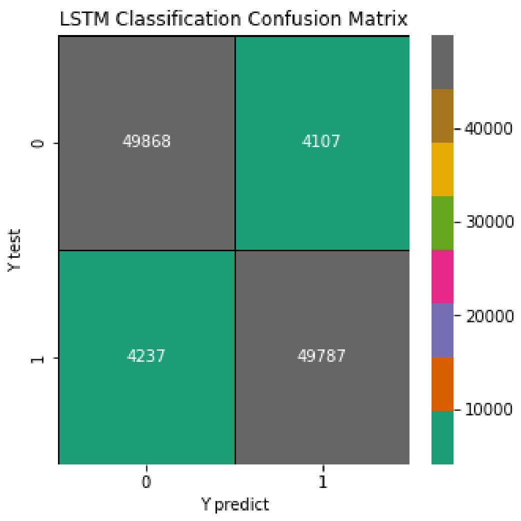


Figure 10: Confusion Matrix for model predictions

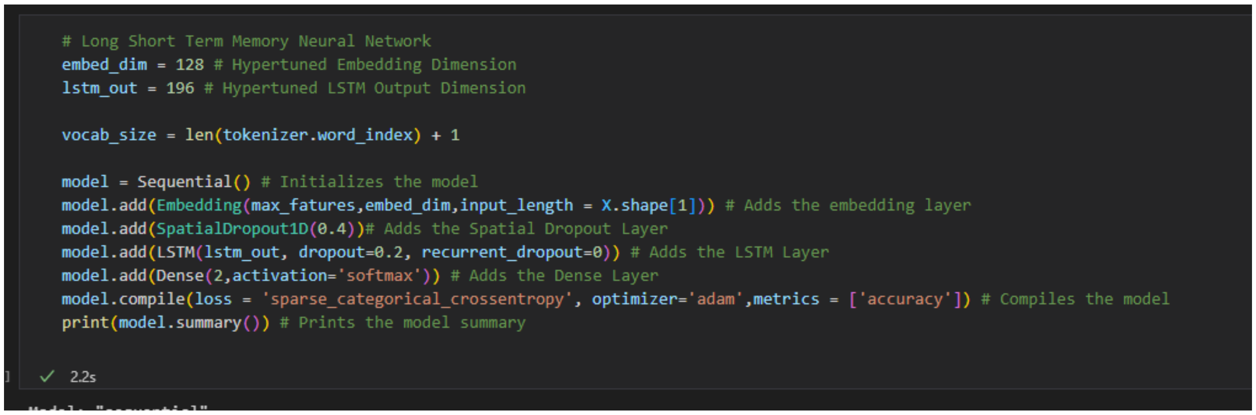
The confusion matrix conveys information about how well the model can make predictions. After making predictions using the test data, the confusion matrix produces an output of how many predictions were correctly predicted and how many were incorrectly predicted. On both axis, 0 refers to negative reviews and 1 refers to positive reviews. After analyzing the confusion matrix, we can see that the model was able to correctly predict 49 868 negative reviews and 49 787 positive reviews. The model incorrectly predicted 4107 reviews as positive (known as false-positives) and 4237 reviews as negative (known as false-negatives). This information can be used to evaluate the model's performance.

10. Hyper tuning the Model:

A LSTM model has four variables that are Hyperparameters, they are the following variables: embed_dim, lstm_out, batch_size and dropout. Due to the nature of LSTM, there is no method to tune the hyperparameters, it is the responsibility of the data analyst to find the best hyperparameters through trial and error. For this Amazon reviews model, the best hyper parameters are embed_dim = 128, lstm_out = 196, batch_size = 64 and dropout = 0.2. A test model was built at the end of the Jupyter notebook with untuned hyperparameters and it achieved an accuracy of 84% while the model with tuned hyperparameters achieved an accuracy of 92%. Tuning the Hyperparameters resulted in an 8% accuracy increase.

11. Interpreting the Results:

Figure 11 illustrates the model that produced the best results:



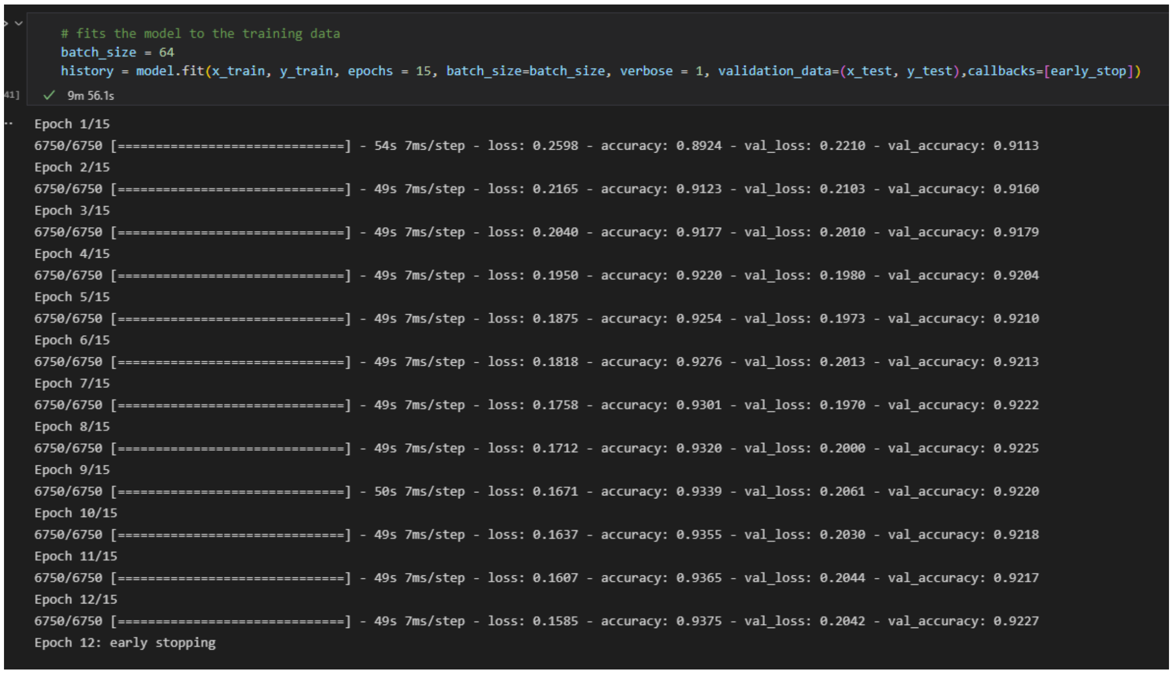
```python
# Long Short Term Memory Neural Network
embed_dim = 128 # Hypertuned Embedding Dimension
lstm_out = 196 # Hypertuned LSTM Output Dimension

vocab_size = len(tokenizer.word_index) + 1

model = Sequential() # Initializes the model
model.add(Embedding(max_fatures,embed_dim,input_length = X.shape[1])) # Adds the embedding layer
model.add(SpatialDropout1D(0.4))# Adds the Spatial Dropout Layer
model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0)) # Adds the LSTM Layer
model.add(Dense(2,activation='softmax')) # Adds the Dense Layer
model.compile(loss = 'sparse_categorical_crossentropy', optimizer='adam',metrics = ['accuracy']) # Compiles the model
print(model.summary()) # Prints the model summary
```

```
✓ 2.2s
```

Figure 11: LSTM Model

The LSTM model above produced accurate and consistent results. The hyperparameters that were tuned are embed_dim, lstm_out, dropout. After hyper tuning and combining the review_title and review_text column the model was able to achieve an accuracy of 92%.

Figure 12 illustrates the accuracies per epoch when training the model:



```python
# fits the model to the training data
batch_size = 64
history = model.fit(x_train, y_train, epochs = 15, batch_size=batch_size, verbose = 1, validation_data=(x_test, y_test),callbacks=[early_stop])
```

```
✓ 9m 56.1s

Epoch 1/15
6750/6750 [==============================] - 54s 7ms/step - loss: 0.2598 - accuracy: 0.8924 - val_loss: 0.2210 - val_accuracy: 0.9113
Epoch 2/15
6750/6750 [==============================] - 49s 7ms/step - loss: 0.2165 - accuracy: 0.9123 - val_loss: 0.2103 - val_accuracy: 0.9160
Epoch 3/15
6750/6750 [==============================] - 49s 7ms/step - loss: 0.2040 - accuracy: 0.9177 - val_loss: 0.2010 - val_accuracy: 0.9179
Epoch 4/15
6750/6750 [==============================] - 49s 7ms/step - loss: 0.1950 - accuracy: 0.9220 - val_loss: 0.1980 - val_accuracy: 0.9204
Epoch 5/15
6750/6750 [==============================] - 49s 7ms/step - loss: 0.1875 - accuracy: 0.9254 - val_loss: 0.1973 - val_accuracy: 0.9210
Epoch 6/15
6750/6750 [==============================] - 49s 7ms/step - loss: 0.1818 - accuracy: 0.9276 - val_loss: 0.2013 - val_accuracy: 0.9213
Epoch 7/15
6750/6750 [==============================] - 49s 7ms/step - loss: 0.1758 - accuracy: 0.9301 - val_loss: 0.1970 - val_accuracy: 0.9222
Epoch 8/15
6750/6750 [==============================] - 49s 7ms/step - loss: 0.1712 - accuracy: 0.9320 - val_loss: 0.2000 - val_accuracy: 0.9225
Epoch 9/15
6750/6750 [==============================] - 50s 7ms/step - loss: 0.1671 - accuracy: 0.9339 - val_loss: 0.2061 - val_accuracy: 0.9220
Epoch 10/15
6750/6750 [==============================] - 49s 7ms/step - loss: 0.1637 - accuracy: 0.9355 - val_loss: 0.2030 - val_accuracy: 0.9218
Epoch 11/15
6750/6750 [==============================] - 49s 7ms/step - loss: 0.1607 - accuracy: 0.9365 - val_loss: 0.2044 - val_accuracy: 0.9217
Epoch 12/15
6750/6750 [==============================] - 49s 7ms/step - loss: 0.1585 - accuracy: 0.9375 - val_loss: 0.2042 - val_accuracy: 0.9227
Epoch 12: early stopping
```

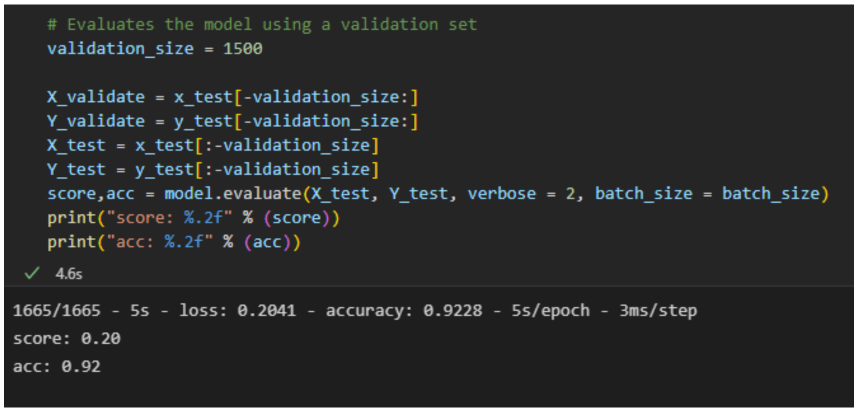
Figure 12: Model accuracy & loss during training

Before interpreting the model results, it is important to note that the data analyst has set up a method called “early_stopping”. This is a callback that will stop the model from over training which allows the data analyst to specify a large number of epochs to get the best results while still combatting over fitting. The model’s accuracy improved after each epoch and the loss decreased after each epoch. This shows that the model is working correctly and is learning after each epoch. An accuracy of 0.93 or 93% after 12 epochs is good and indicates that the model should make correct predictions majority of the time. A loss function of 0.20 after 12 epochs is good. This indicates that there is a low number of errors when comparing the predicted output to training output. The test model at the end of the Jupyter Notebook that consists of untuned hyperparameters and only the review_text column resulted in an accuracy of 84%. The data analyst was able to increase the model’s accuracy by 8% by turning the hyperparameters and adding review_title into the model.

12. Evaluating the Model:

The results have been evaluated using the model's accuracy, loss function, confusion matrix, and sklearn metrics library.

Figure 11 illustrates the model evaluate using accuracy and loss:



```
# Evaluates the model using a validation set
validation_size = 1500

X_validate = x_test[-validation_size:]
Y_validate = y_test[-validation_size:]
X_test = x_test[:-validation_size]
Y_test = y_test[:-validation_size]
score,acc = model.evaluate(X_test, Y_test, verbose = 2, batch_size = batch_size)
print("score: %.2f" % (score))
print("acc: %.2f" % (acc))
```

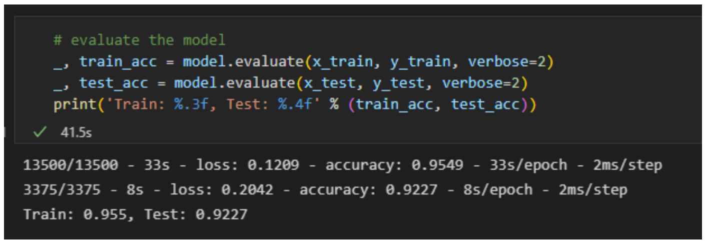
✓ 4.6s

```
1665/1665 - 5s - loss: 0.2041 - accuracy: 0.9228 - 5s/epoch - 3ms/step
score: 0.20
acc: 0.92
```

Figure 13: Model evaluation using validation data

The model has been evaluated using a validation dataset that is taken from the y_test variable which was not included in the model training. This ensures that the model is not evaluating against the data it learnt from and producing a false accuracy. Accuracy refers to how often we can expect our model to correctly predict an outcome. An accuracy score of 0.92 is a good and this means that majority of the time it will make accurate predictions. Loss refers to the loss function of the model, this measures how different predicted outputs compare to the expected outputs (Kamali, 2021). The model has a loss function of 0.20 which is a good value for loss. The lower the loss the better the model will perform due to low number of errors.

Figure 12 illustrates difference between evaluation with training and test data:



```
# evaluate the model
_, train_acc = model.evaluate(x_train, y_train, verbose=2)
_, test_acc = model.evaluate(x_test, y_test, verbose=2)
print('Train: %.3f, Test: %.4f' % (train_acc, test_acc))
```

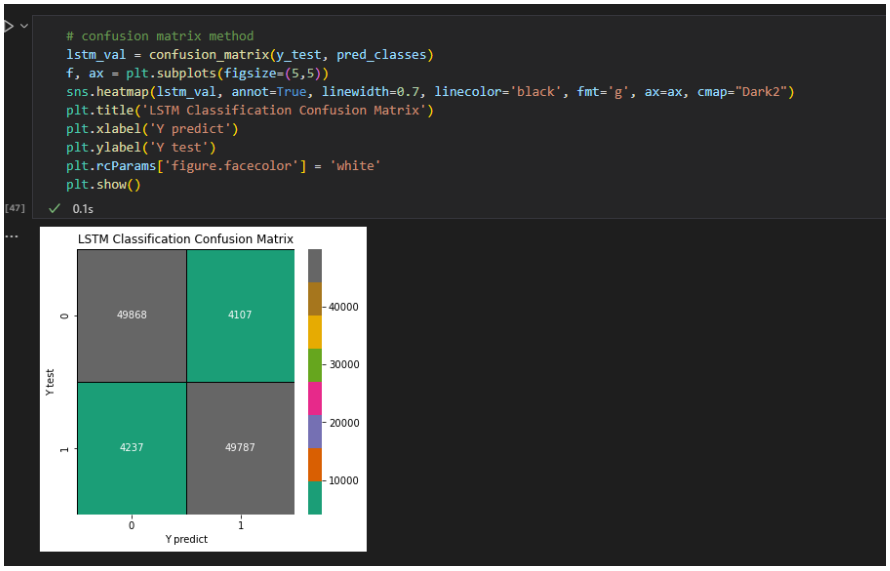
✓ 41.5s

```
13500/13500 - 33s - loss: 0.1209 - accuracy: 0.9549 - 33s/epoch - 2ms/step
3375/3375 - 8s - loss: 0.2042 - accuracy: 0.9227 - 8s/epoch - 2ms/step
Train: 0.955, Test: 0.9227
```

Figure 14: Evaluation using training and test data

When evaluating the model, it is better to use separate test data. If we look at figure 12 as an example, the model’s accuracy is 0.95 when evaluating the results against the training data that was used during training. When using the test data that was kept separate to the model’s training data, we get a more accurate result of 0.92. This is due to the model learning and evaluating against itself.

Figure 13 illustrates the confusion matrix:

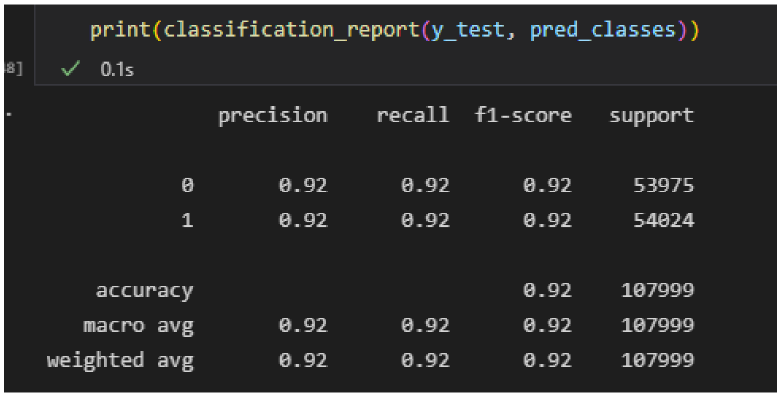


```
# confusion matrix method
lstm_val = confusion_matrix(y_test, pred_classes)
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(lstm_val, annot=True, linewidth=0.7, linecolor='black', fmt='g', ax=ax, cmap="Dark2")
plt.title('LSTM Classification Confusion Matrix')
plt.xlabel('Y predict')
plt.ylabel('Y test')
plt.rcParams['figure.facecolor'] = 'white'
plt.show()
```

Figure 15: Confusion Matrix for predicted values

We can further investigate the model’s prediction performance visually using a confusion matric. A confusion matrix illustrates how many predicts were incorrectly predicted. From the confusion matrix we can see, the model only predicted 4017 negative reviews incorrectly and 4237 positive reviews incorrectly. We can assume the model performed well due to the low number of incorrectly predicted values when compared to the high number of correctly predicted values.

Figure 14 illustrates the sklearn classification report:



```
print(classification_report(y_test, pred_classes))
✓ 0.1s

              precision    recall  f1-score   support

           0       0.92      0.92      0.92     53975
           1       0.92      0.92      0.92     54024

    accuracy                           0.92    107999
   macro avg       0.92      0.92      0.92    107999
weighted avg       0.92      0.92      0.92    107999
```

Figure 16: Classification Report

From sklearn's classification report, we can get the precision, recall and f1_score scores for the model. The precision score measures the proportion of positively predicted labels that are actually correct (Kumar, 2022). A precision score of 0.92 for predicted negative reviews, 0.92 for predicted positive reviews is a good score. This means that majority of the positive predicted values are correct.

Recall score represents the model's ability to correctly predict the positives out of actual positives (Kumar, 2022).A recall score of 0.92 for negative reviews, 0. 92 for positive reviews is a good score. This tells us that the model is good at identifying actual positive out of all positives that exist within the dataset.

F1_score represents the model score as a function of precision and recall score. F1_score is a performance metric that gives equal weight to both the Precision and Recall for measuring its performance in terms of accuracy (Kumar, 2022). A f1_score of 0.92 for negative reviews, 0.92 for positive reviews is a good score. This tells us that the model is optimized, and it is providing accurate outputs.

13. Possible more appropriate methods:

A possible method that may increase the accuracy of the model is by using the Bidirectional LSTM algorithm. This method sends sequence information in both directions, backwards (future to past) or forwards (past to future). This is known as back propagation or forward propagation. This will improve the model's ability to learn from new inputs while still taking older inputs into consideration which will increase the accuracy of predictions.

14. Summary of Code:

The model created follows the structure of Figure 15:

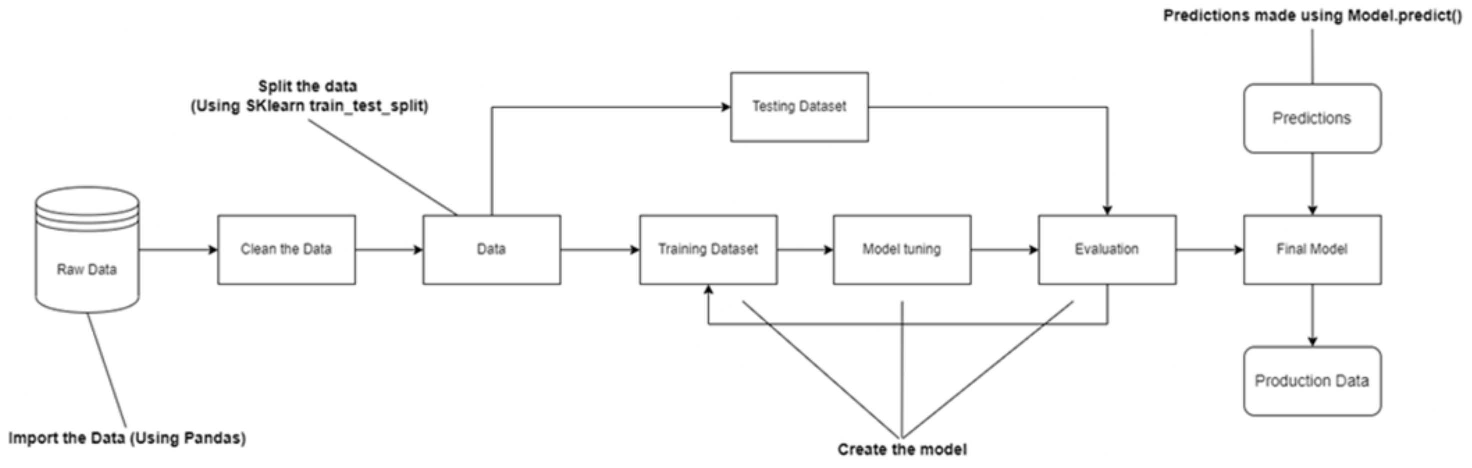


Figure 17: Model diagram

The raw data is first imported using Pandas. The data is visualized using graphs to gain an understanding of any potential trends. The data is then cleaned using Featured engineering to eliminate any null values, remove punctuation, combine columns, remove stop words and one-hot encode. The next step is to split the dataset into training and test data. The training data is used to train the model and the algorithm selected to train the model is Long Shot-Term Memory (LSTM). After building the model, the hyperparameters are tuned through trial and error. After tuning the hyperparameters, the model is evaluated using the test data and the sklearn libraries (classification report, accuracy, score, confusion matrix.

15. Conclusion:

After visualizing the dataset, it contained a balanced number of positive and negative reviews. The most common phrase used in the negative reviews was “don’t waste money” and the most common phrase used in the positive reviews was “highly recommend book”. Once the model was built using text data from Amazon reviews, it achieved an accuracy of 92% and a loss of 0.20. This model performed much better when compared to the test model (untuned hyperparameters) that had an accuracy of 84% and a loss of 0.36. The model that achieved an accuracy of 92% can be used to successfully identify sentiment within text and categorize whether the review is positive or negative. The model is also able to predict sentiment without providing a review title. This will allow companies like Amazon to analyze large datasets of reviews, identify sentiment within the reviews and categorize the reviews based on if it is positive, neutral, or negative. Using this model, a company like Amazon will be able to analyze customer feedback and highlight areas of improvement that their customers are not happy with. Ultimately this will increase customer satisfaction and increase profits in the long term. The outcome achieve from this model is that LSTM and RNN’s can be used to predict the sentiment of a review. All the analysis questions are answered by the results produced by the model and the model was able to successfully categorize the reviews.

16. References

Brownlee, J., 2017. *Machine Learning Mastery.* [Online]
Available at: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/#:~:text=Long%20Short%2DTerm%20Memory%20(LSTM,complex%20area%20of%20deep%20learning.
[Accessed 14 09 2021].

Brownlee, J., 2020. *Machine Learning Mastery.* [Online]
Available at: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/
[Accessed 12 09 2022].

Burns, E. & Brush, K., 2021. *TechTarget.* [Online]
Available at: https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network
[Accessed 12 09 2022].

Chugh, A., 2021. *GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/
[Accessed 14 09 2022].

DataRobot, 2016. *What is Natural Language Processing?.* [Online]
Available at: https://www.datarobot.com/blog/what-is-natural-language-processing-introduction-to-nlp/
[Accessed 12 09 2022].

IBM Cloud Education, 2020. *IBM.* [Online]
Available at: https://www.ibm.com/cloud/learn/recurrent-neural-networks
[Accessed 12 09 2022].

Kamali, K., 2021. *Training Galaxy Project.* [Online]
Available at: https://training.galaxyproject.org/archive/2021-06-01/topics/statistics/tutorials/RNN/tutorial.html#:~:text=A%20loss%20function%20measur

es%20how,used%20to%20train%20the%20model.
[Accessed 14 09 2022].

Kumar, A., 2022. *Vital Flux.* [Online]
Available at: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/
[Accessed 06 06 2022].