Sentiment Analysis : using NLP for E-commerce data of Apparel reviews

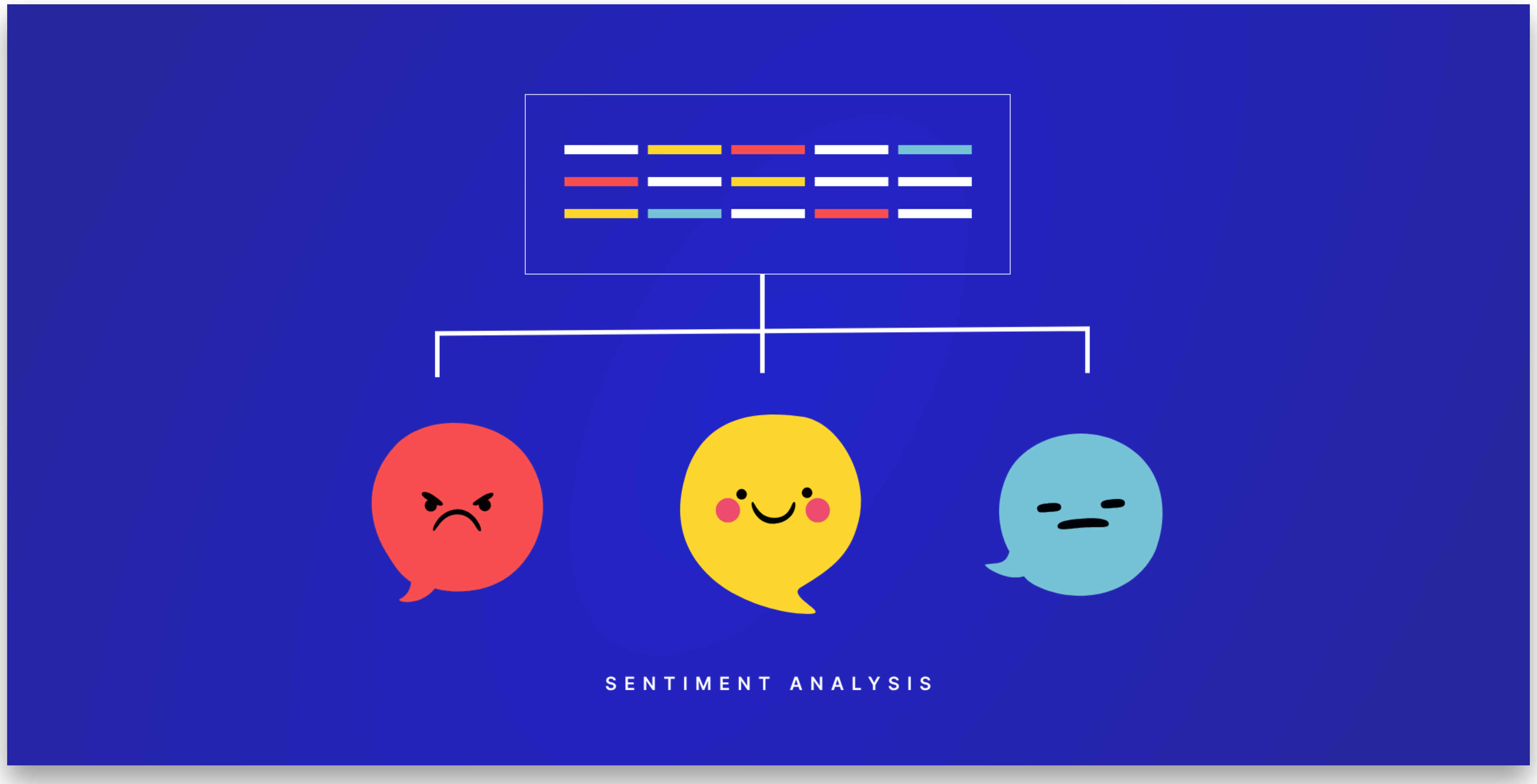
Data Science Diploma Program
Capstone - Sprint3

Presented By:-
Chandana Chaudhry

Introduction

Problem at hand : Understanding customer sentiments is of paramount importance in marketing strategies and product improvement and figuring out a way to use qualitative data quantitatively.

Sentiment Analysis : is the process of analyzing digital text to determine if the emotional tone of the message is positive, negative, or neutral.



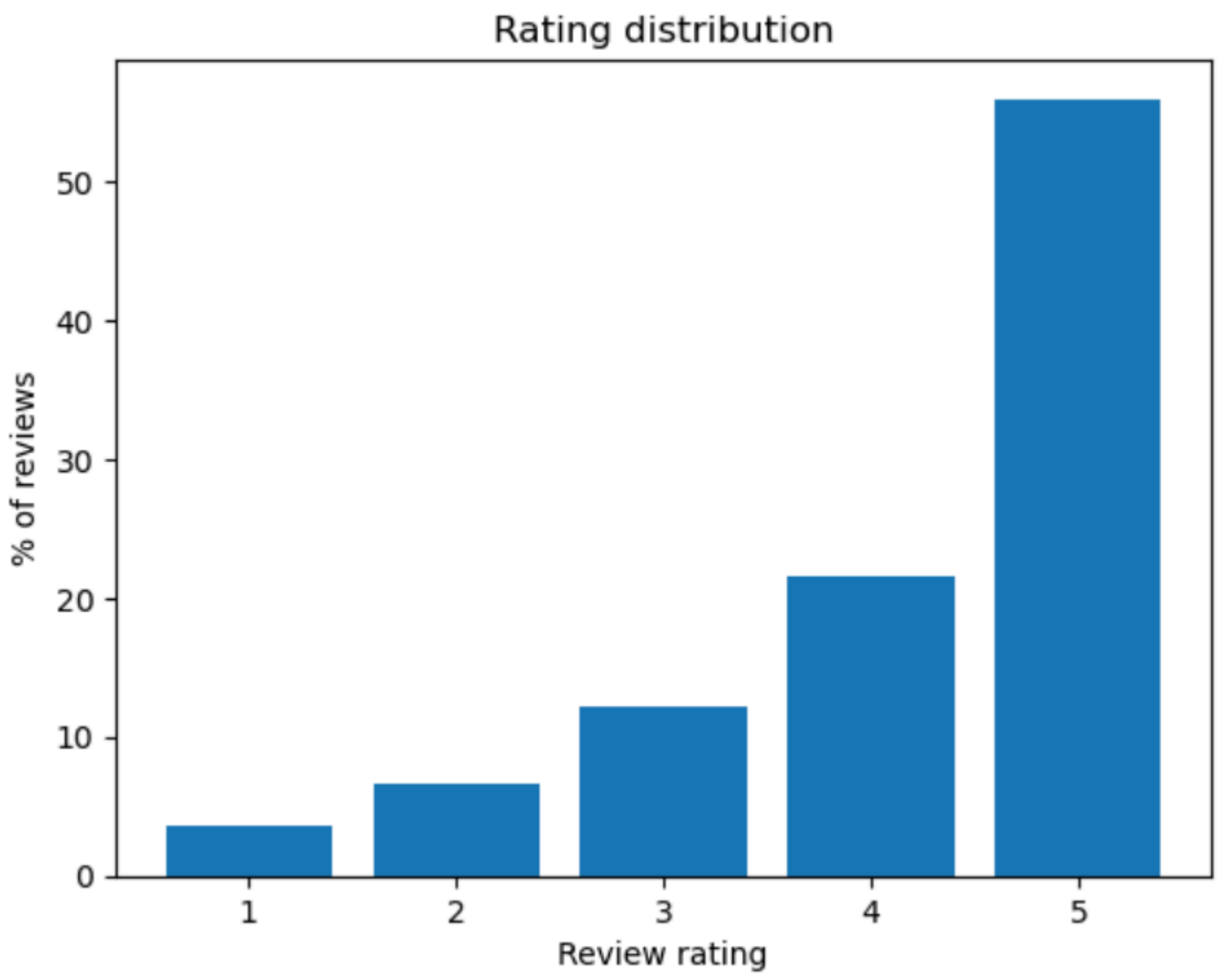
Our Approach/Objective : Sentiment Analysis using NLP with the help of using e-commerce data of apparel reviews.

References : *https://aws.amazon.com/what-is/sentiment-analysis/*
Image src: *https://exemplary.ai/blog/sentiment-analysis*

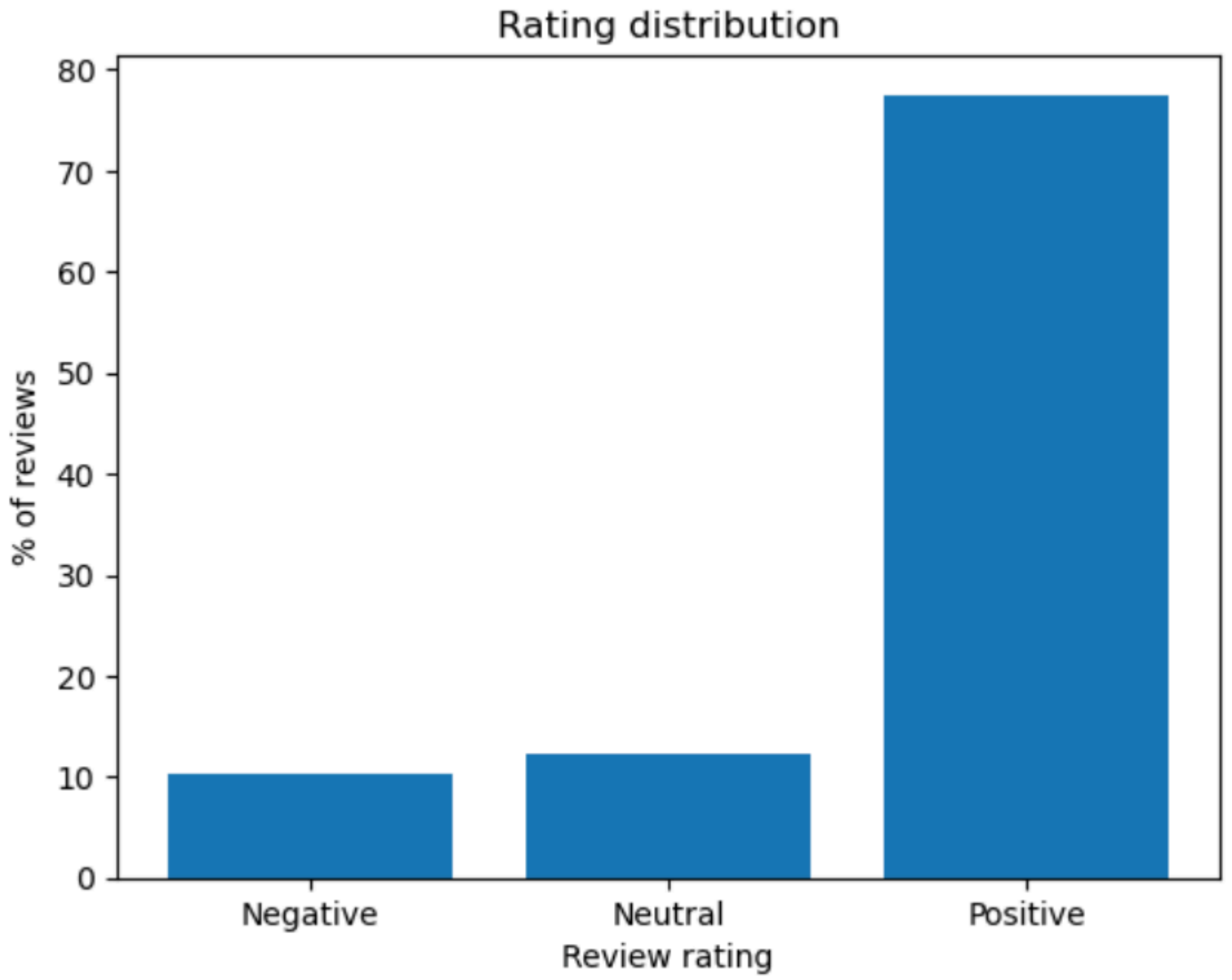
Exploratory Data Analysis and Insights

- Shape of the Data Set is 23486 rows and 11 columns
- 5 Object type columns and 6 int types
- The data set contains ‘Title : 3810’ and ‘Review_text : 845’ null values
- Classification problem with ‘Rating’ as the target variable
- After plotting a heat map, there is high correlation between ‘Rating’ and ‘Recommendation_IND’.
- The data at hand had to be processed into three classification : ‘Positive’ , ‘Neutral’ and ‘Negative’. Initially it had 5 classes
- Also, the data at hand was imbalanced.

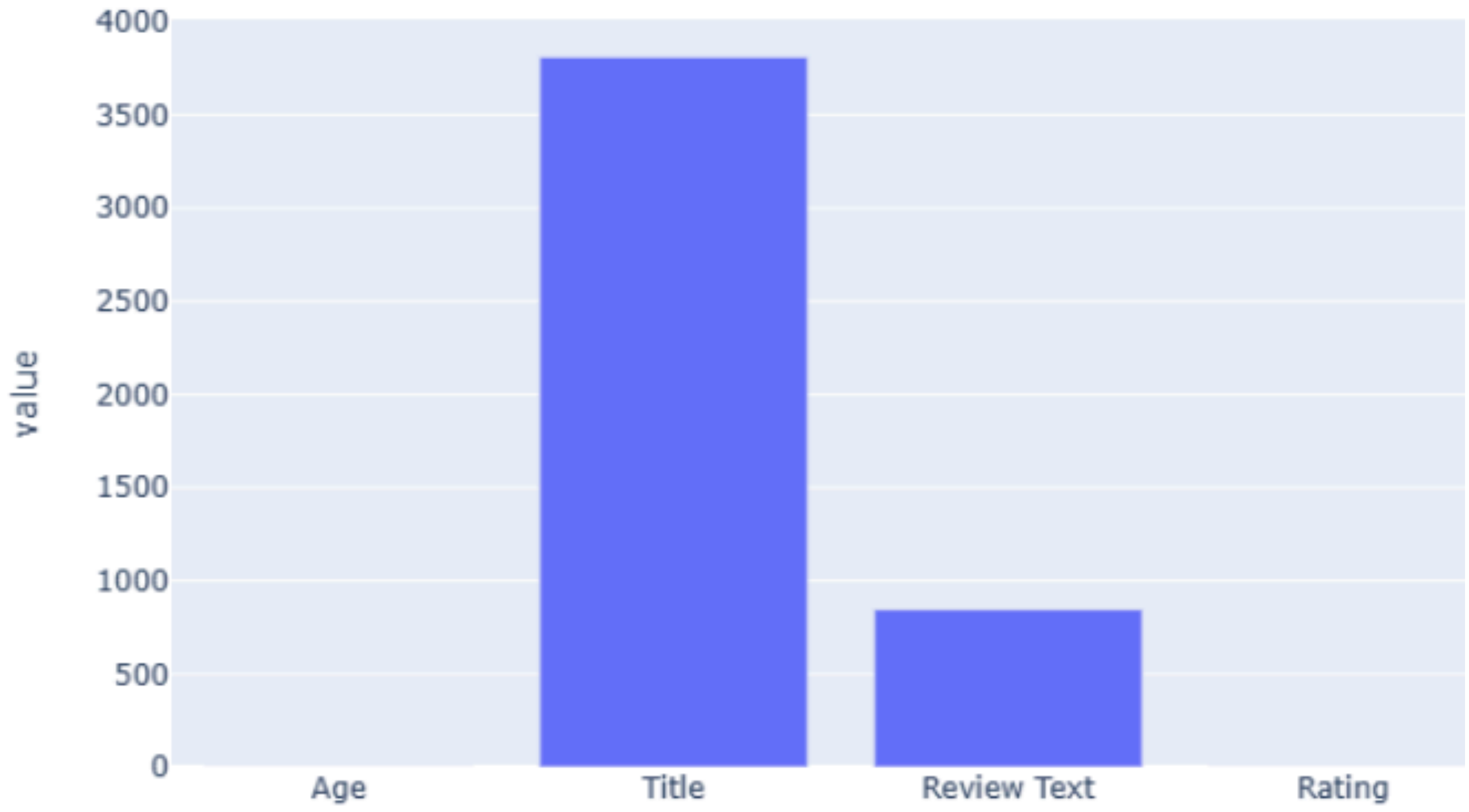
Exploratory Data Analysis and Insights



The rating distribution before processing



The rating distribution after processing



Null value distribution

Data Preprocessing :

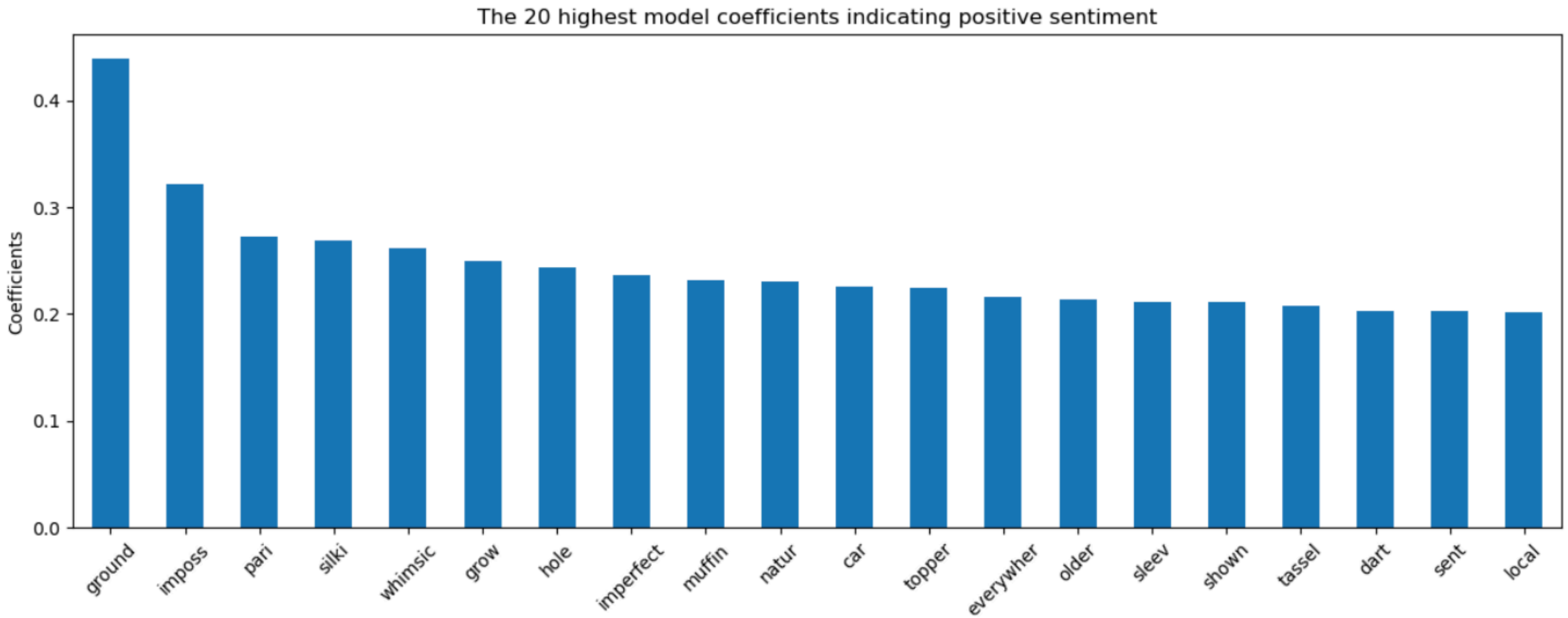
- Dropping the Title column, as it is not relevant to the task of predicting the rating of a review given the text of the review.
- Removing any rows that are missing the rating label.
- Binarizing the rating column into three categories: Negative, Neutral, and Positive. Dropping the original rating column, as it is no longer needed.
- Removing any rows that are missing any values after the previous steps.
- Tokenizing data using NLP BagofWords technique and Tokenizer in Keras for LSTM
- Concatenating 'Tokens' to our data frame before model fit step.

Baseline Model or Model 1 : Logistic Regression

Logistic regression is one of the most basic (yet effective) tools we have for classifying categorical data.

We plotted the top 20 tokens with highest Log.coefficients

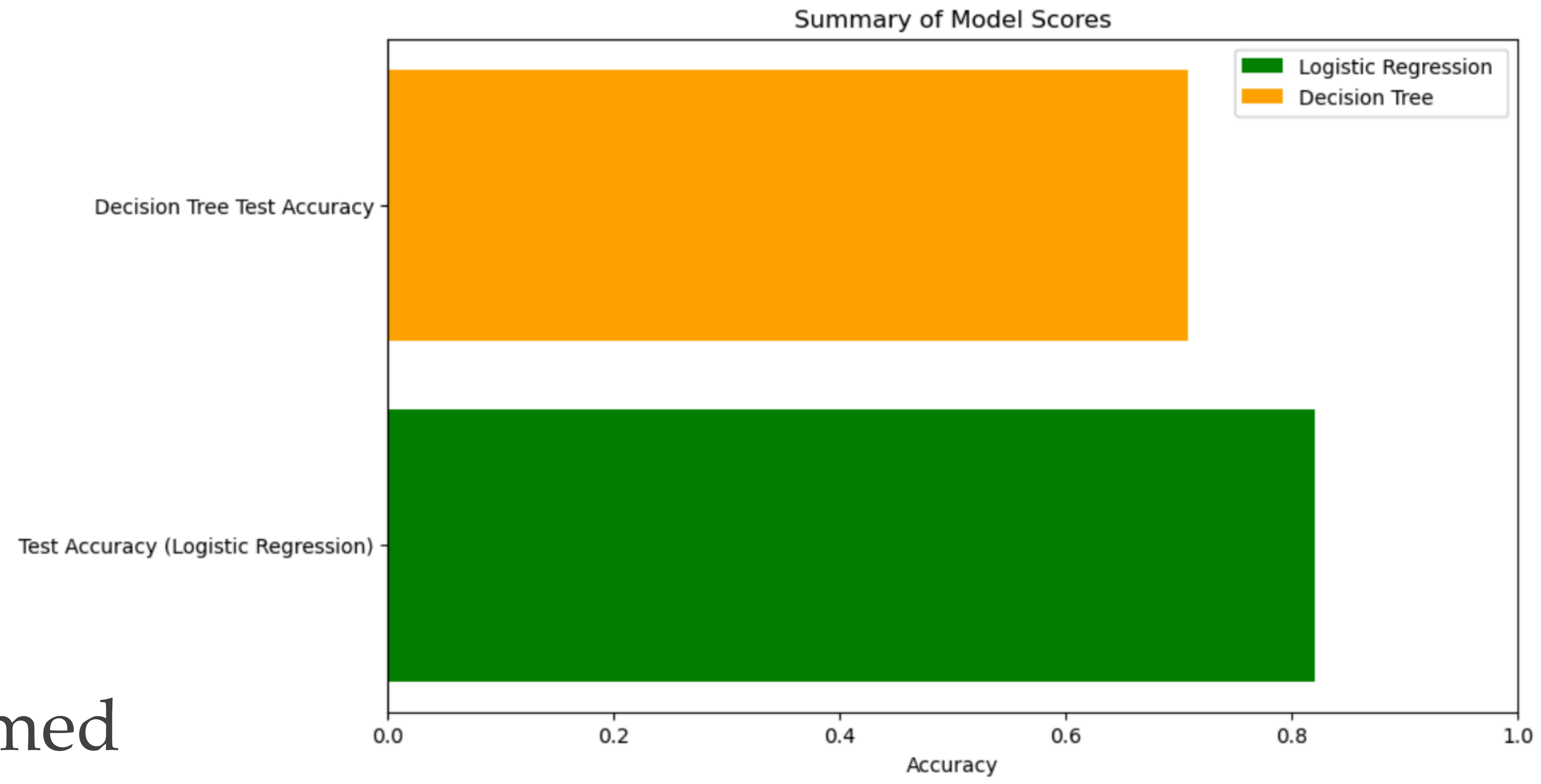
Logistic regression Accuracy Score : 82%



Model 2 : Decision Trees

- Performed using pipeline and with PCA (n_components = 0.90) preserving 90% of variance
- Max_depth = 10
- Min_sample leaf = 1
- Model Accuracy = 70%

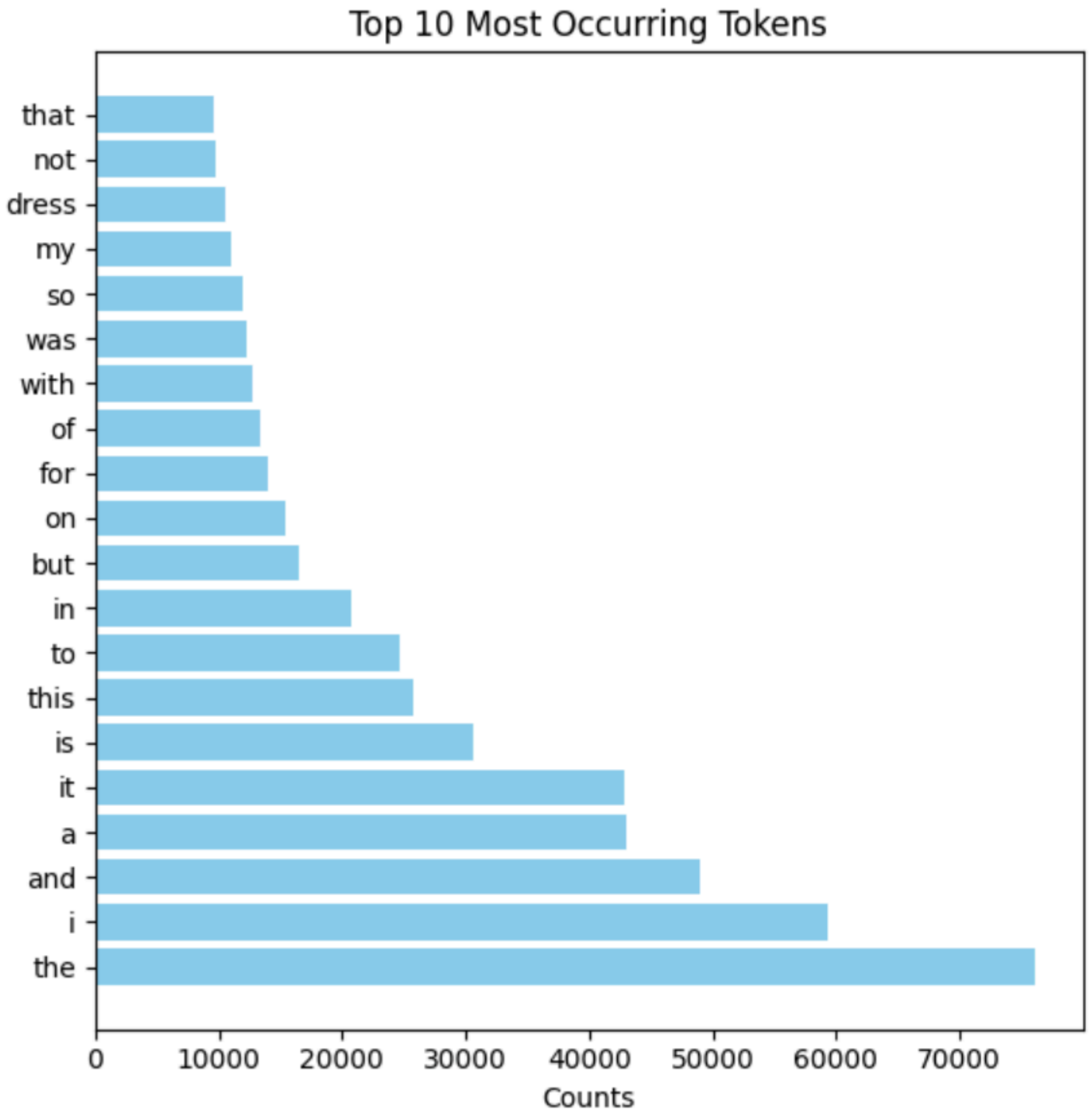
Concluding that Logistic Regression performed better

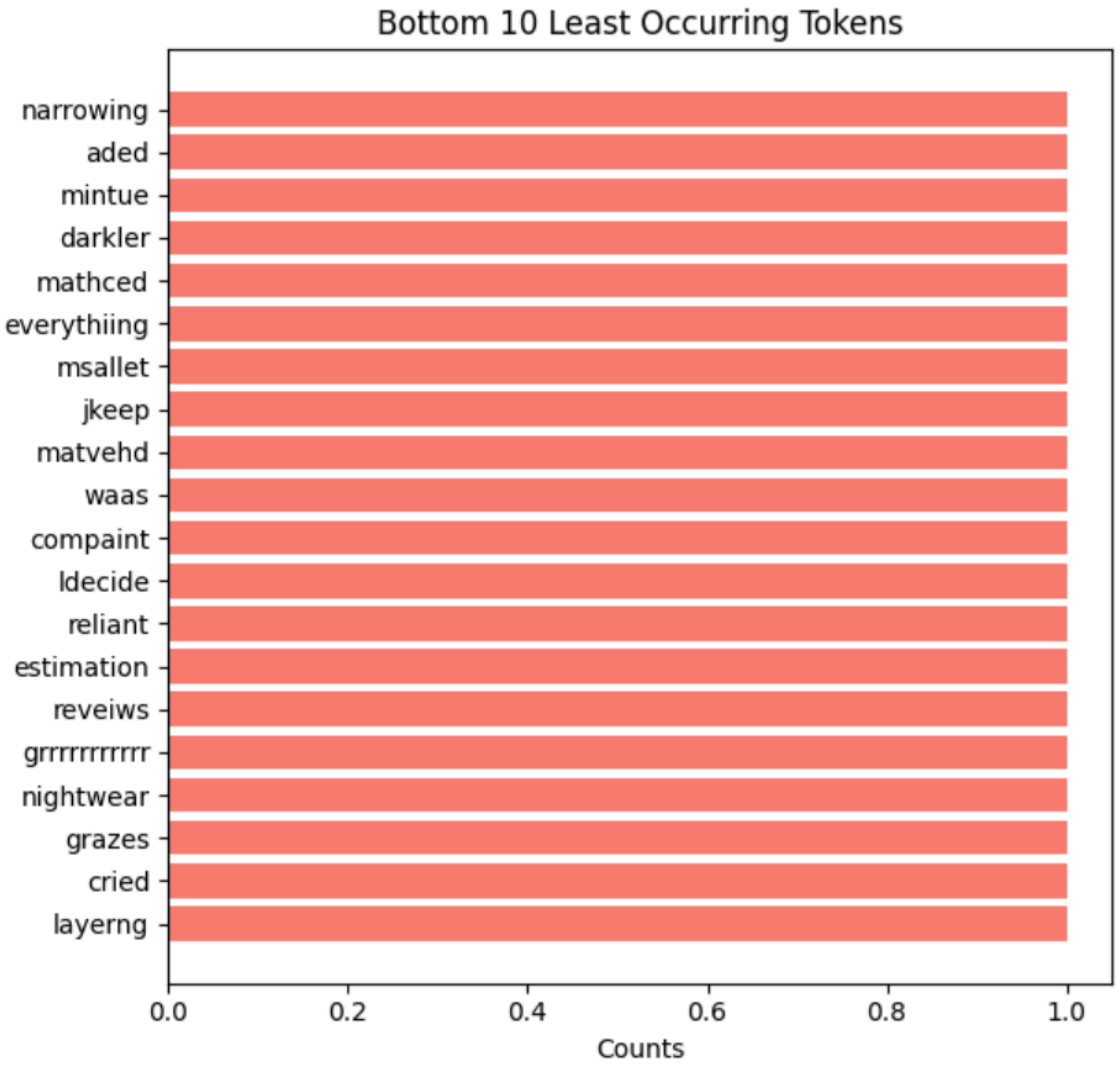


Model 3 : LSTM Neural Network

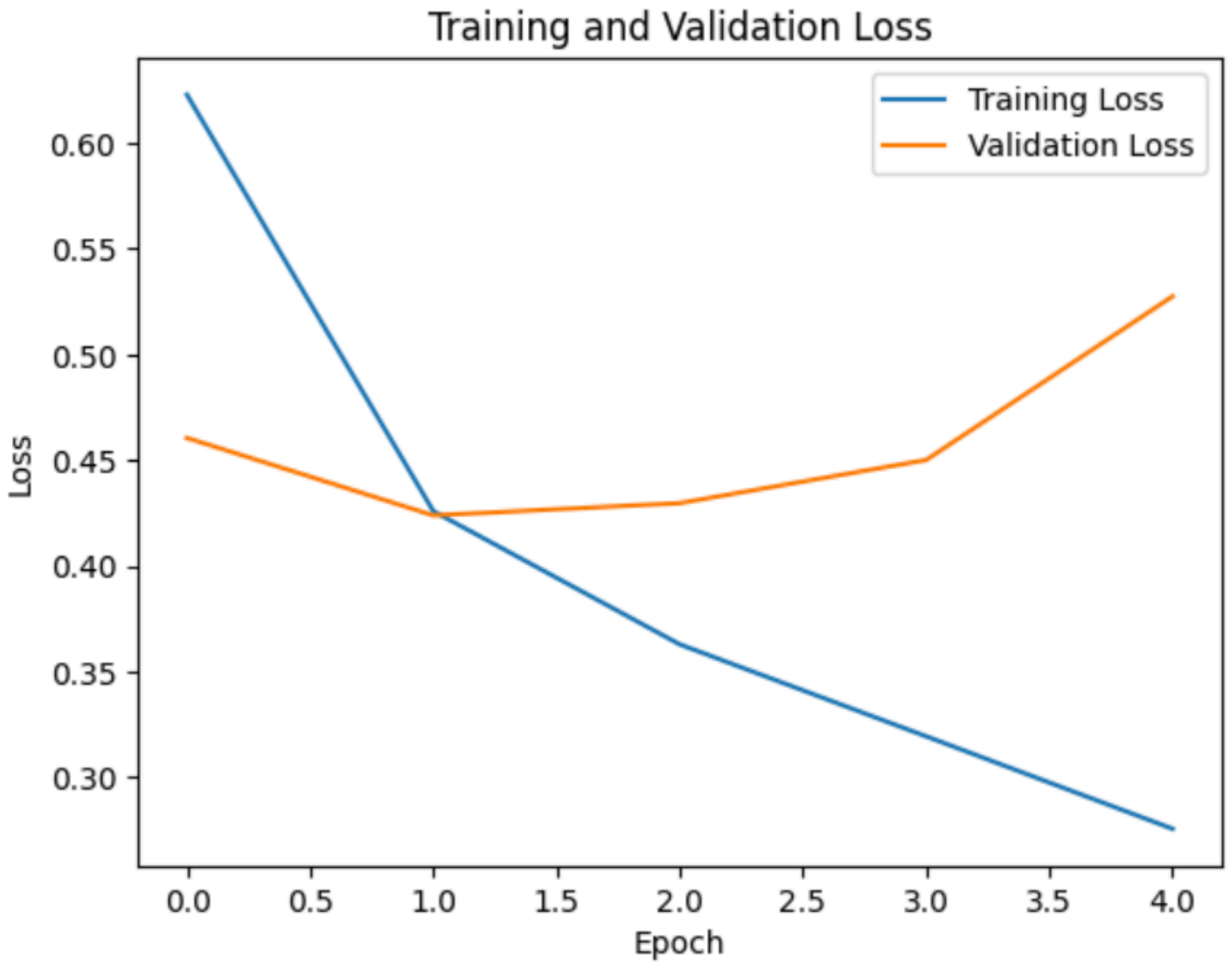
- Tokenization: Review_Text is tokenized using the `Tokenizer` class from Keras. Text corpus into sequences of integers
- Padding Sequences : The sequences are padded to ensure that they all have the same length using
- One-Hot Encoding: The target variable 'Sentiment' is one-hot encoded using
- Train_Test_Split : The dataset is split into training and testing sets using. Test data : 20%
- Defining the LSTM Model
- Compiling the Model: The model is compiled with Adam optimizer and categorical crossentropy loss function. The metric for evaluation is accuracy.
- Training the Model: The model is trained on the training data for 10 epochs with a batch size of 32.
- Early Stopping: to prevent overfitting by monitoring the validation loss
- Evaluation : The model is evaluated on the testing data, and the test accuracy is printed.
- Testing on review string

Tokens

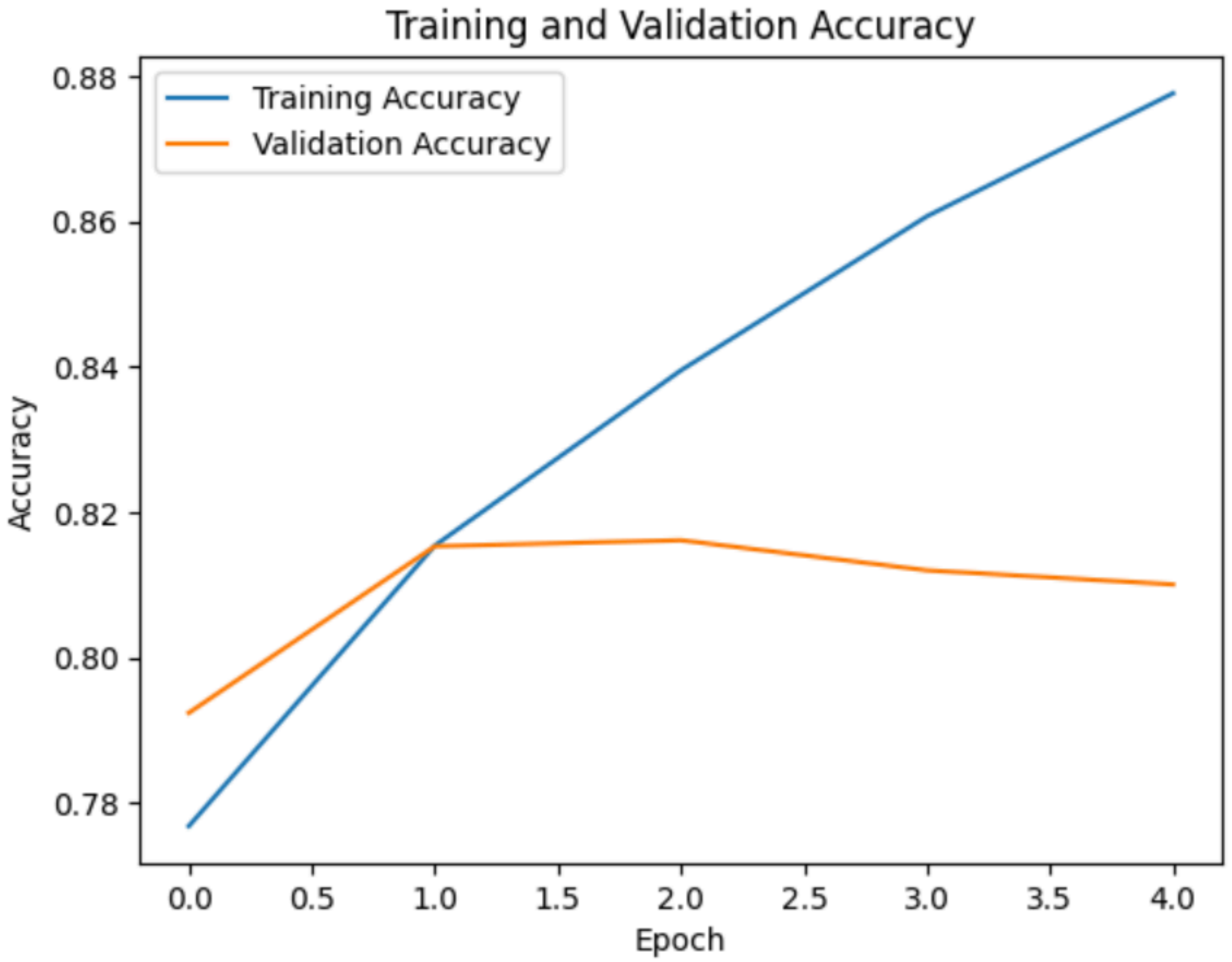

Top 10 Most Occurring Tokens
that
not
dress
my
so
was
with
of
for
on
but
in
to
this
is
it
a
and
i
the
0
10000
20000
30000
40000
50000
60000
70000
Counts


Bottom 10 Least Occurring Tokens
narrowing
aded
mintue
darkler
mathced
everythiing
msallet
jkeep
matvehd
waas
compaint
ldecide
reliant
estimation
reveiws
grrrrrrrrrr
nightwear
grazes
cried
layerng
0.0
0.2
0.4
0.6
0.8
1.0
Counts

LSTM : Training and Validation loss and accuracy

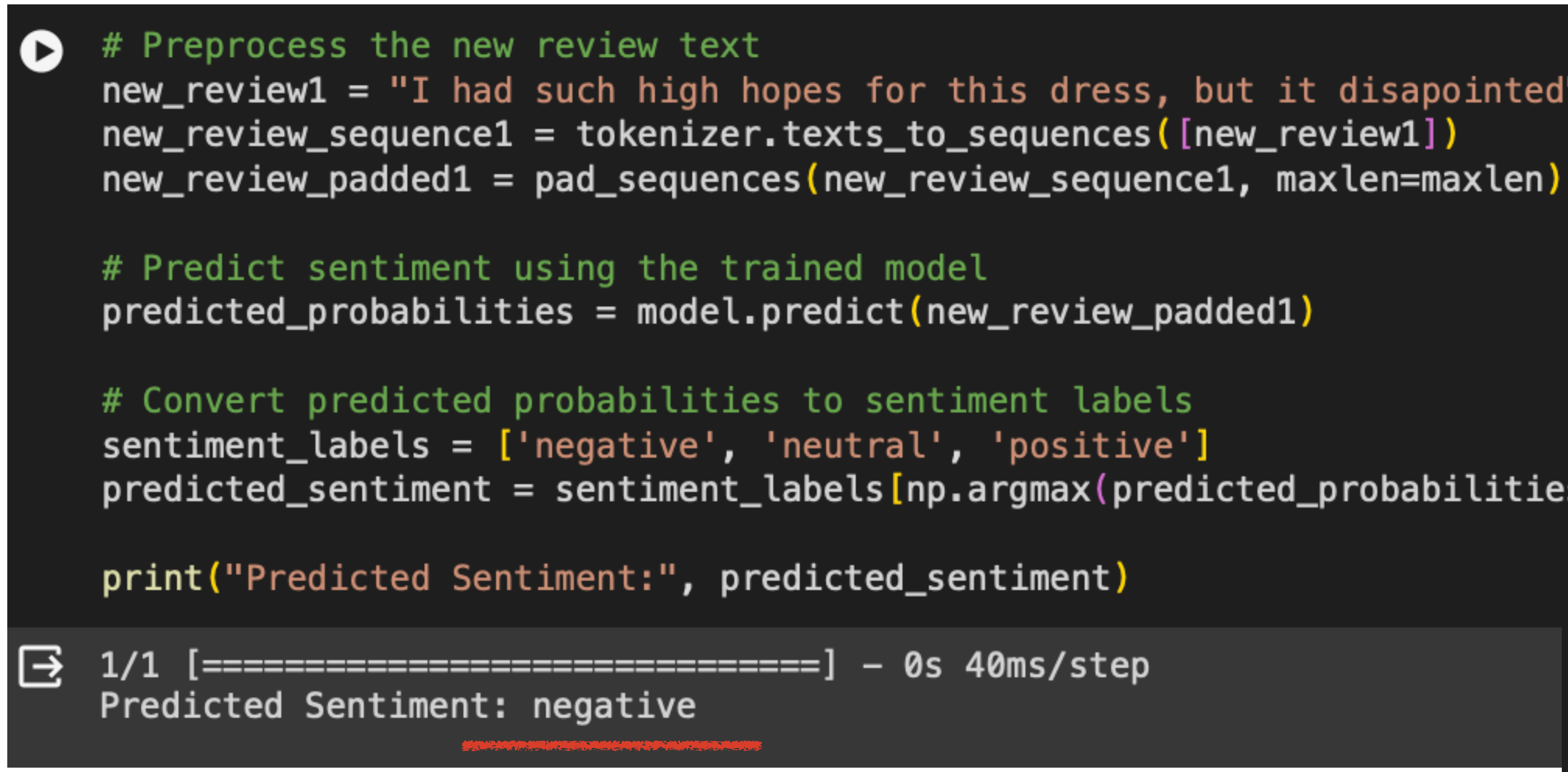


Training loss vs Validation Loss



Training Accuracy vs Validation Validation

Testing Model : Unseen Sample

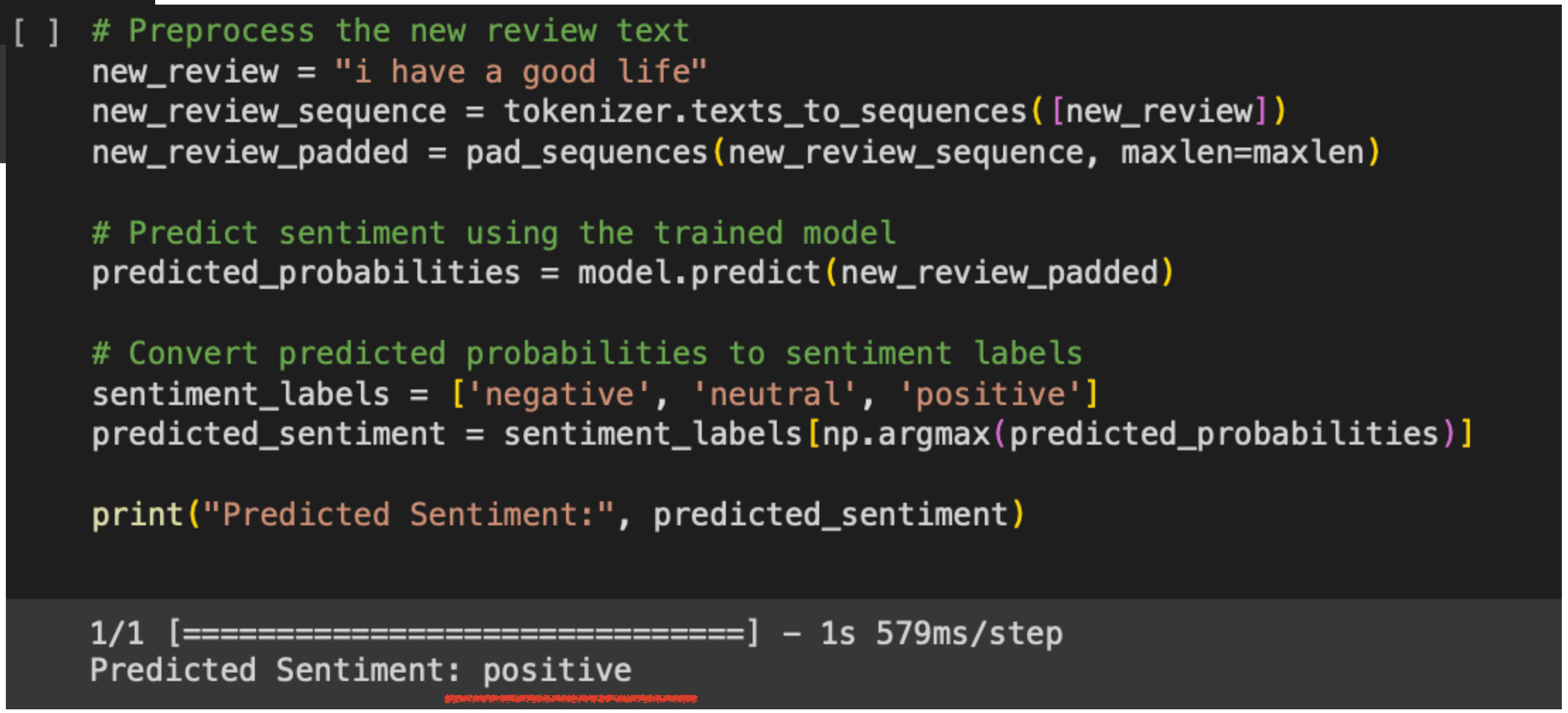


```
# Preprocess the new review text
new_review1 = "I had such high hopes for this dress, but it disapointed"
new_review_sequence1 = tokenizer.texts_to_sequences([new_review1])
new_review_padded1 = pad_sequences(new_review_sequence1, maxlen=maxlen)

# Predict sentiment using the trained model
predicted_probabilities = model.predict(new_review_padded1)

# Convert predicted probabilities to sentiment labels
sentiment_labels = ['negative', 'neutral', 'positive']
predicted_sentiment = sentiment_labels[np.argmax(predicted_probabilities)]

print("Predicted Sentiment:", predicted_sentiment)
```

```
1/1 [==============================] - 0s 40ms/step
Predicted Sentiment: negative
```

Testing negative sentiment correctly

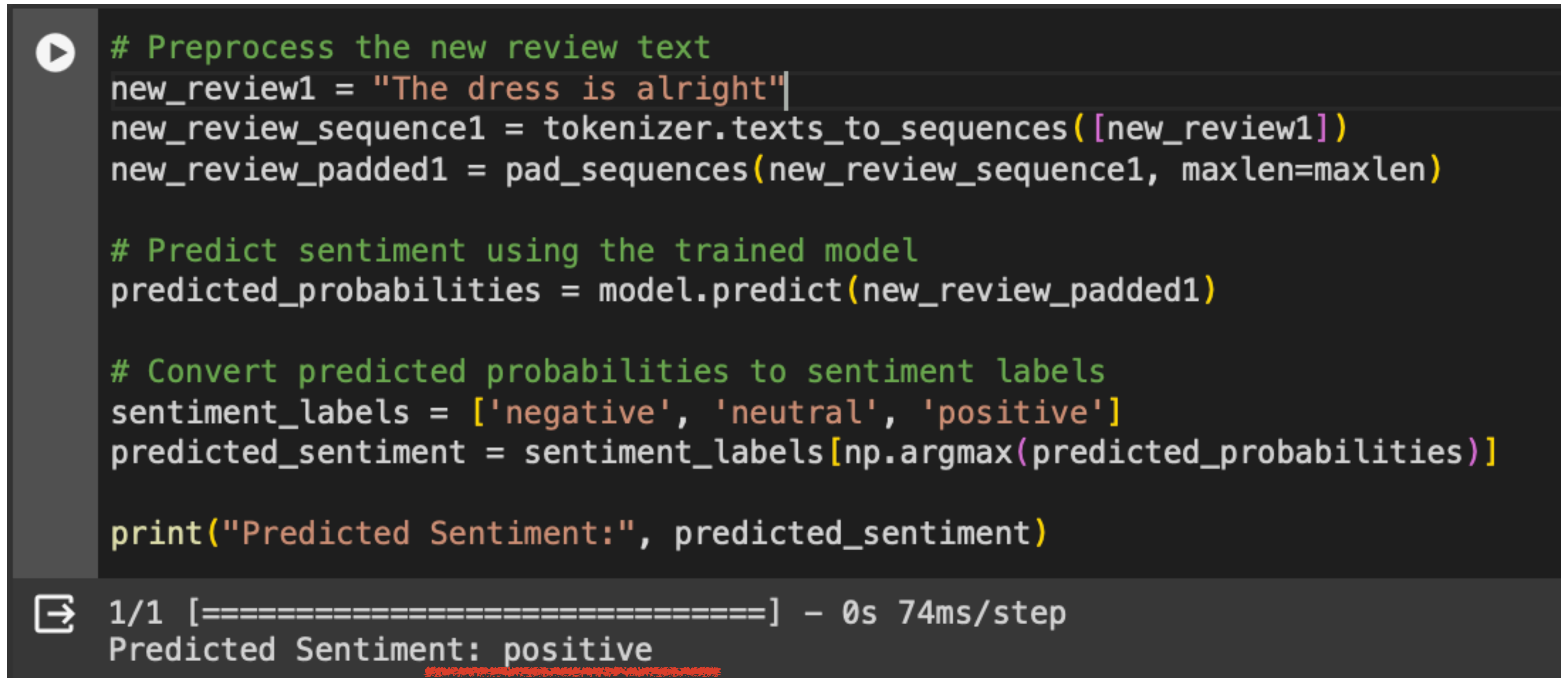
Testing positive sentiment correctly



```
# Preprocess the new review text
new_review = "i have a good life"
new_review_sequence = tokenizer.texts_to_sequences([new_review])
new_review_padded = pad_sequences(new_review_sequence, maxlen=maxlen)

# Predict sentiment using the trained model
predicted_probabilities = model.predict(new_review_padded)

# Convert predicted probabilities to sentiment labels
sentiment_labels = ['negative', 'neutral', 'positive']
predicted_sentiment = sentiment_labels[np.argmax(predicted_probabilities)]

print("Predicted Sentiment:", predicted_sentiment)
```

```
1/1 [==============================] - 1s 579ms/step
Predicted Sentiment: positive
```

Incorrect Classification

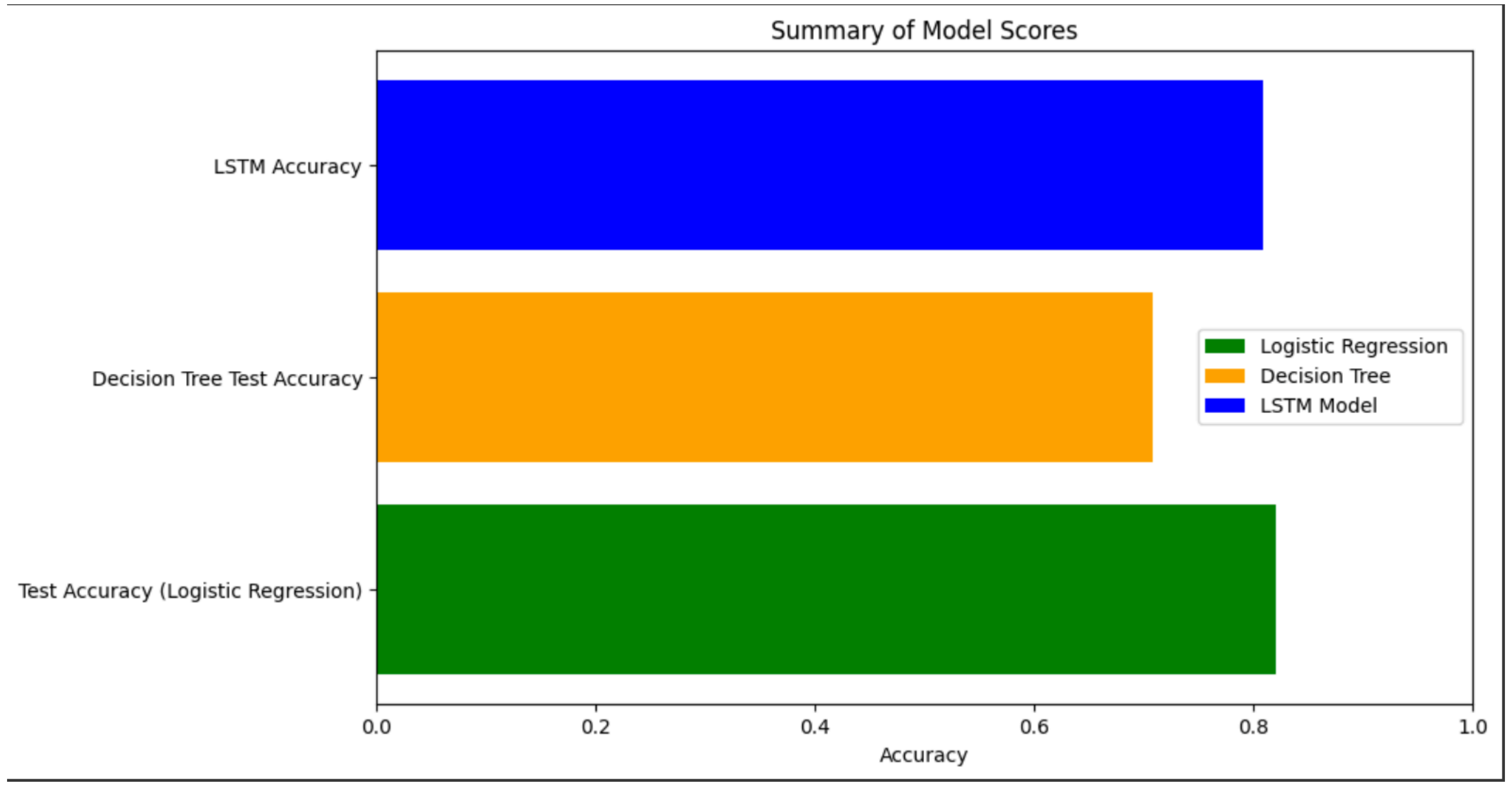


```
# Preprocess the new review text
new_review1 = "The dress is alright"
new_review_sequence1 = tokenizer.texts_to_sequences([new_review1])
new_review_padded1 = pad_sequences(new_review_sequence1, maxlen=maxlen)

# Predict sentiment using the trained model
predicted_probabilities = model.predict(new_review_padded1)

# Convert predicted probabilities to sentiment labels
sentiment_labels = ['negative', 'neutral', 'positive']
predicted_sentiment = sentiment_labels[np.argmax(predicted_probabilities)]

print("Predicted Sentiment:", predicted_sentiment)
```

```
1/1 [==============================] - 0s 74ms/step
Predicted Sentiment: positive
```

Testing neutral sentiment incorrectly

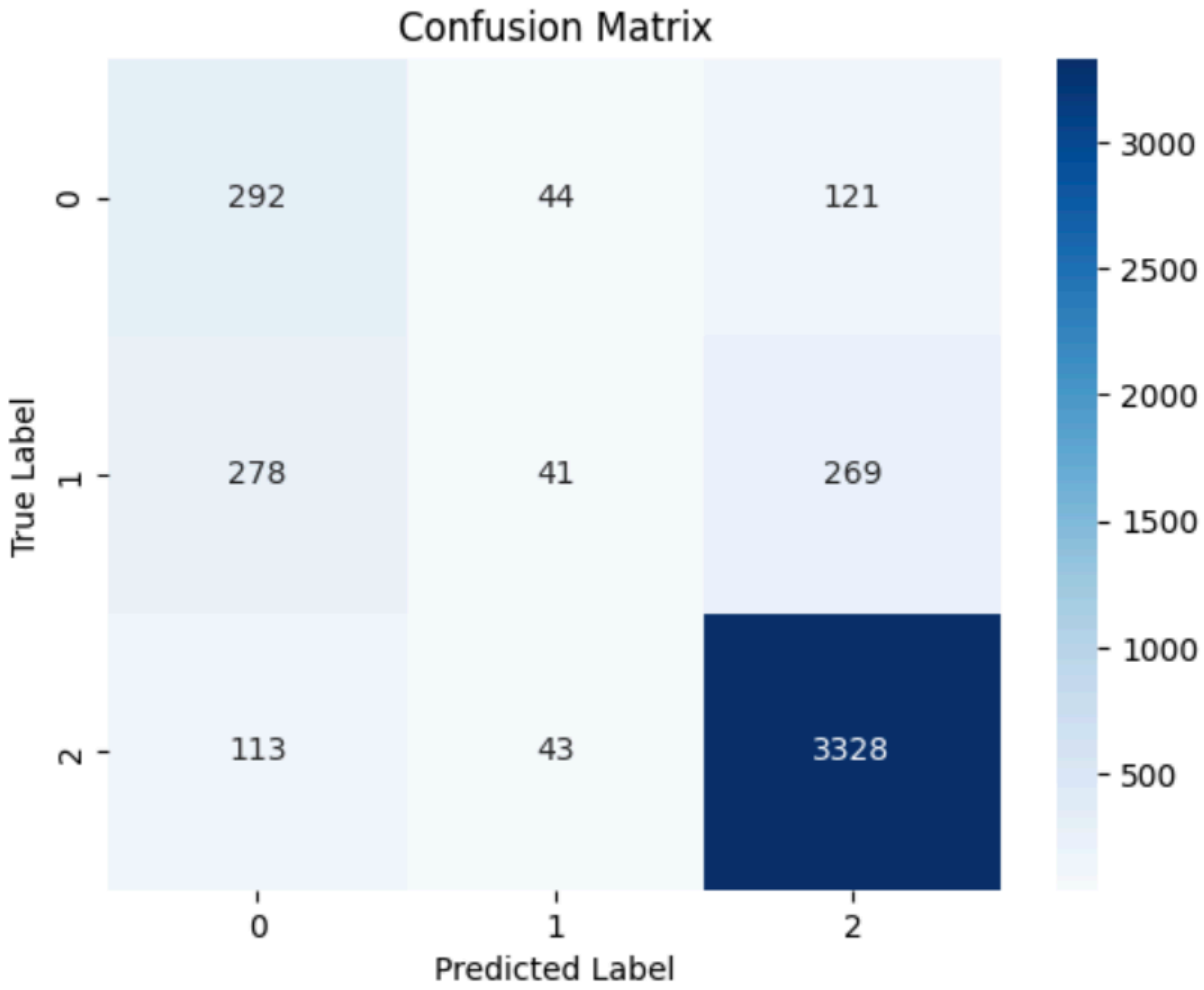
Model Comparison : Accuracy Score



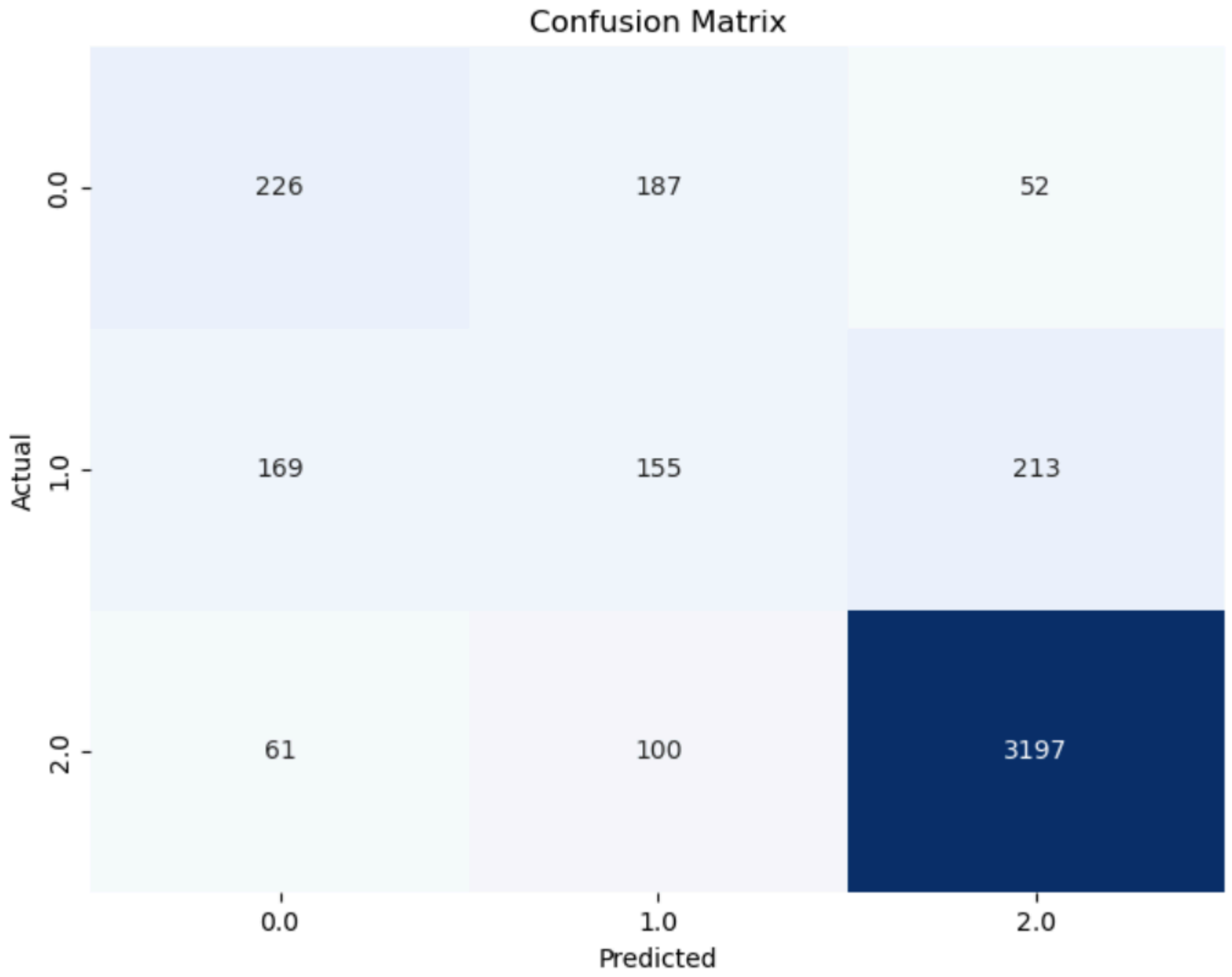
- Logistic_reg =82%
- Decision Tree = 71%
- LSTM_model = 81.25%

Conclusive of Logistic regression and LSTM having similar accuracy

Confusion Matrix



LSTM Model



Logistic Regression

Next Iterations

- Fine Tune LSTM model
- Saving logistic regression model and creating a web app to display the model for sentiment prediction
- Compile all work and organize Jupyter notebooks

Thank you and References

- https://www.kaggle.com/datasets/nicapotato/womens-ecommerce-clothing-reviews
- **Sentiment Classification :**https://www.researchgate.net/publication/323545316_Statistical_Analysis_on_E-Commerce_Reviews_with_Sentiment_Classification_using_Bidirectional_Recurrent_Neural_Network , **paper pdf link : https://arxiv.org/pdf/1805.03687.pdf**
- https://aws.amazon.com/what-is/sentiment-analysis/
- https://medium.com/@zaiinn440/attention-is-all-you-need-the-core-idea-of-the-transformer-bbfa9a749937
- **Attention is all you need :** https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- **Case Analysis - Twitter** : https://www.academia.edu/31874952/Twitter_Sentiment_Analysis_
- Sentiment Analysis using CNN : https://computingonline.net/files/journals/1/archieve/IJC_2022_21_2_10.pdf
- https://towardsdatascience.com/multiclass-text-classification-using-lstm-in-pytorch-eac56baed8df