Performance Assessment

WGU | D213

D213 Task 2

Narcisse, Laurie

2024

Contents

[**Part I: Research Question** 2](#_Toc159239373)

[A.  Describe the purpose of this data analysis by doing the following: 2](#_Toc159239374)

[1.  Summarize one research question that you will answer using neural network models and NLP techniques. Be sure the research question is relevant to a real-world organizational situation and sentiment analysis captured in your chosen data set(s). 2](#_Toc159239375)

[2.  Define the objectives or goals of the data analysis. Be sure the objectives or goals are reasonable within the scope of the research question and are represented in the available data. 2](#_Toc159239376)

[3.  Identify a type of neural network capable of performing a text classification task that can be trained to produce useful predictions on text sequences on the selected data set. 2](#_Toc159239377)

[**Part II: Data Preparation** 3](#_Toc159239378)

[B.  Summarize the data cleaning process by doing the following: 3](#_Toc159239379)

[1.  Perform exploratory data analysis on the chosen data set, and include an explanation of each of the following elements: 3](#_Toc159239380)

[2.  Describe the goals of the tokenization process, including any code generated and packages that are used to normalize text during the tokenization process. 20](#_Toc159239381)

[3.  Explain the padding process used to standardize the length of sequences. Include the following in your explanation: 22](#_Toc159239382)

[4.  Identify how many categories of sentiment will be used and an activation function for the final dense layer of the network. 23](#_Toc159239383)

[5.  Explain the steps used to prepare the data for analysis, including the size of the training, validation, and test set split (based on the industry average). 23](#_Toc159239384)

[6.  Provide a copy of the prepared data set. 23](#_Toc159239385)

[**Part III: Network Architecture** 24](#_Toc159239386)

[C.  Describe the type of network used by doing the following: 24](#_Toc159239387)

[1.  Provide the output of the model summary of the function from TensorFlow. 24](#_Toc159239388)

[2.  Discuss the number of layers, the type of layers, and the total number of parameters. 24](#_Toc159239389)

[3.  Justify the choice of hyperparameters, including the following elements: 25](#_Toc159239390)

[**Part IV: Model Evaluation** 26](#_Toc159239391)

[D.  Evaluate the model training process and its relevant outcomes by doing the following: 26](#_Toc159239392)

[1.  Discuss the impact of using stopping criteria to include defining the number of epochs, including a screenshot showing the final training epoch. 26](#_Toc159239393)

[2.  Assess the fitness of the model and any actions taken to address overfitting. 29](#_Toc159239394)

[3.  Provide visualizations of the model’s training process, including a line graph of the loss and chosen evaluation metric. 30](#_Toc159239395)

[4.  Discuss the predictive accuracy of the trained network using the chosen evaluation metric from part D3. 31](#_Toc159239396)

[**Part V: Summary and Recommendations** 32](#_Toc159239397)

[E.  Provide the code you used to save the trained network within the neural network. 32](#_Toc159239398)

[F.  Discuss the functionality of your neural network, including the impact of the network architecture. 32](#_Toc159239399)

[G.  Recommend a course of action based on your results. 32](#_Toc159239400)

[**Part VI: Reporting** 33](#_Toc159239401)

[H.  Show your neural network in an industry-relevant interactive development environment (e.g., a Jupyter Notebook). Include a PDF or HTML document of your executed notebook presentation. 33](#_Toc159239402)

[I.  Denote specific web sources you used to acquire segments of third-party code that was used to support the application. 33](#_Toc159239403)

# **Part I: Research Question**

## A.  Describe the purpose of this data analysis by doing the following:

### 1.  Summarize **one** research question that you will answer using neural network models and NLP techniques. Be sure the research question is relevant to a real-world organizational situation and sentiment analysis captured in your chosen data set(s).

Can we leverage neural networks and NLP to analyze the sentiment of customer reviews?

### 2.  Define the objectives or goals of the data analysis. Be sure the objectives or goals are reasonable within the scope of the research question and are represented in the available data.

In this report, we will employ the Python TensorFlow library to develop a predictive model aimed at discerning the sentiment of customer reviews. The approach relies on leveraging labeled datasets from customer feedback to train the model effectively.

The primary objective of our data analysis endeavors is to gain insights into customer preferences by identifying products that have garnered positive evaluations. Through this process, I aim to better understand customer sentiment and preferences, enabling businesses to tailor their offerings to meet customer expectations effectively.

### 3.  Identify a type of neural network capable of performing a text classification task that can be trained to produce useful predictions on text sequences on the selected data set.

For text classification tasks like sentiment analysis, it's essential to consider the multifaceted nature of emotions and intentions conveyed within textual data. Sentiment analysis goes beyond merely identifying polarity (positive, negative, or neutral) to encompass a wide spectrum of emotions, including but not limited to anger, joy, sadness, urgency, and indifference. By delving deeper into the textual content, sentiment analysis can provide valuable insights into the diverse range of sentiments expressed, offering a nuanced understanding of the underlying emotions and intentions.

For my Natural Language Processing (NLP) model, I will leverage the Python TensorFlow and Keras libraries. Utilizing an NLP model offers the most effective approach for discerning the nuanced emotions embedded within textual data. The overarching goal of our data analysis initiative is to develop a predictive model capable of forecasting the sentiment of forthcoming customer reviews. This will be achieved through the utilization of labeled data derived from past customer feedback.

Moreover, by identifying customers' preferred products based on positive product evaluations, our aim is to provide the company with deeper insights into its clientele. This understanding enables businesses to tailor their offerings more effectively to meet customer expectations and enhance overall satisfaction.

# **Part II: Data Preparation**

## B.  Summarize the data cleaning process by doing the following:

### 1.  Perform exploratory data analysis on the chosen data set, and include an explanation of each of the following elements:

#### •   presence of unusual characters (e.g., emojis, non-English characters)

#### •   vocabulary size

#### •   proposed word embedding length

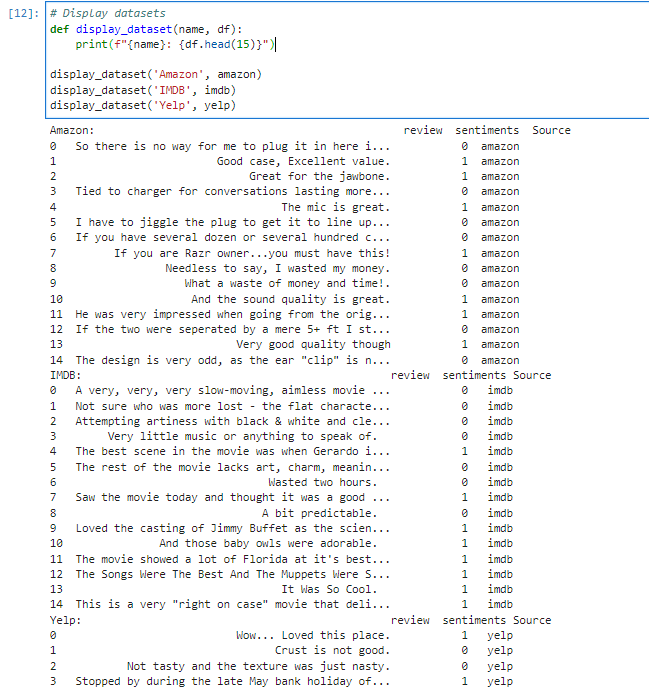
#### •   statistical justification for the chosen maximum sequence length

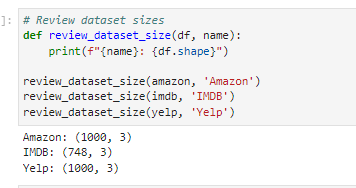
The amalgamation of three datasets, labeled from Amazon, IMDB, and Yelp, will be consolidated into a singular dataframe for analytical purposes. These datasets encompass:

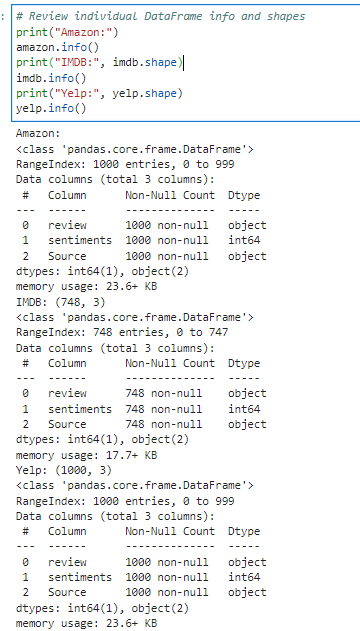
* Amazon Product Data Collection
* UCSD Recommender Systems Data Collection
* UCI Sentiment Labeled Sentences Data Collection











A screenshot of a computer program

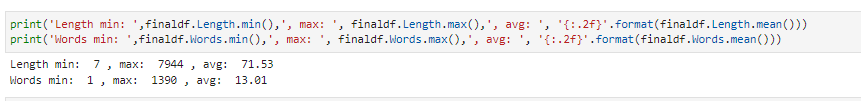
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer code

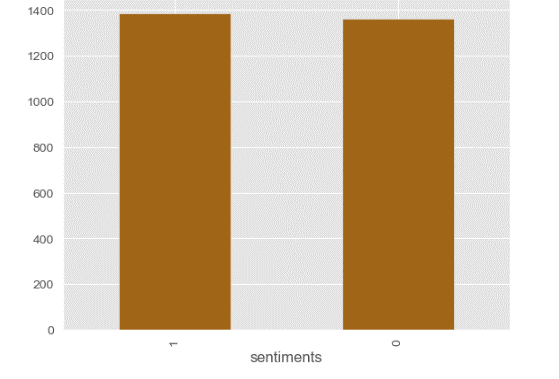
Description automatically generated

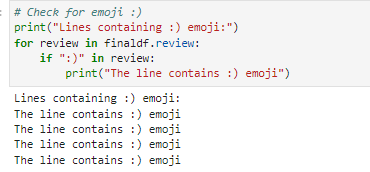
A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated





A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer code with colorful text

Description automatically generated

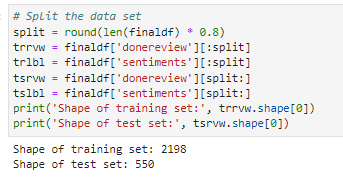
A graph with a bar graph

Description automatically generated

A graph of a bar graph

Description automatically generated with medium confidence

### 2.  Describe the goals of the tokenization process, including any code generated and packages that are used to normalize text during the tokenization process.

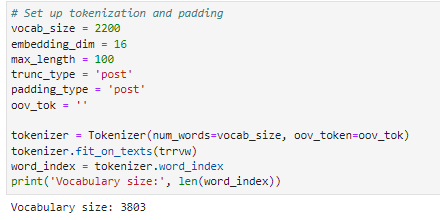


First, I split the dataset into training and testing sets. To accomplish this, I calculated the index for splitting by taking 80% of the total length of the finaldf DataFrame. This index represented where to divide the dataset into training and testing portions.

Then, I assigned the reviews ('donereview' column) and corresponding sentiments ('sentiments' column) from the finaldf DataFrame to the training set variables (trrvw and trlbl) using slicing. The training set contained the first 80% of the reviews and sentiments.

Similarly, I assigned the remaining reviews and sentiments from the finaldf DataFrame to the testing set variables (tsrvw and tslbl) using slicing. The testing set contained the last 20% of the reviews and sentiments.

After splitting the dataset, I printed out the shapes of the training and testing sets to verify the split. The training set consisted of 2198 samples, while the testing set consisted of 550 samples.



I set up tokenization and padding for the dataset. First, I defined a vocabulary size of 2200. This choice was based on the fact that the training set contained 2198 samples, and I wanted to ensure that the vocabulary size was large enough to cover most of the unique words in the training data while also allowing some flexibility for out-of-vocabulary tokens.

Additionally, I specified an embedding dimension of 16, which determines the size of the embedding vectors for each word. A smaller embedding dimension was chosen to reduce computational complexity while still capturing meaningful relationships between words.

For tokenization, I instantiated a Tokenizer object with the specified vocabulary size and an out-of-vocabulary token (oov\_tok) set to an empty string. This tokenizer was then fitted on the training reviews (trrvw) to generate a word index, which maps each word to a unique integer.

After fitting the tokenizer, I obtained the word index and printed out the vocabulary size. Despite setting the vocabulary size to 2200, the actual vocabulary size turned out to be 3803. This result indicates that the tokenizer discovered more unique words in the training data than initially expected.

### 3.  Explain the padding process used to standardize the length of sequences. Include the following in your explanation:

#### •   if the padding occurs before or after the text sequence

#### •   a screenshot of a single padded sequence

A screenshot of a computer

Description automatically generated

I used the tokenizer to convert the textual reviews in the training dataset (trrvw) into sequences of tokens. This is done by calling tokenizer.texts\_to\_sequences(trrvw). Each word in the reviews is replaced by its corresponding index in the tokenizer's word index.

Next, I padded these sequences to ensure that they all have the same length. This is important because neural networks typically require inputs of fixed size. I used the pad\_sequences function, passing in the sequences of tokens (train\_sequences), the maximum length (max\_length), and parameters for padding and truncating (padding\_type and trunc\_type). The resulting padded\_train\_sequences contained sequences of the same length, with padding added as necessary.

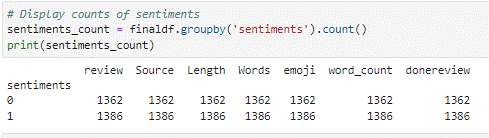
Then, I converted the labels for the training data (trlbl) into numpy arrays, which are suitable for use in machine learning models.

Similarly, I performed the same steps for the testing dataset (tsrvw), converting the textual reviews to sequences of tokens, padding them, and converting the labels to numpy arrays (tslbl\_final).

Finally, I printed out some of the training sequences for inspection. These sequences show the reviews converted into tokenized form. This provides a glimpse into how the textual data has been transformed into numerical sequences for training the model.

I also checked the lengths of both training and testing datasets along with their corresponding labels. This step ensures that the number of reviews matches the number of labels for both the training and testing datasets. If the lengths didn't match, it could indicate a problem with the data preparation or loading process.

### 4.  Identify how many categories of sentiment will be used and an activation function for the final dense layer of the network.



### 5.  Explain the steps used to prepare the data for analysis, including the size of the training, validation, and test set split (based on the industry average).

To prepare the data for analysis, I followed several steps using the required applications. First, I filled a Pandas dataframe with three datasets: individual information from Amazon and IMDb, and the corresponding review labels. This allowed me to examine the structure of the supplied data, gaining insights into its composition. Analyzing individual DataFrames from Amazon and IMDb provided a deeper understanding of their contents and formats.

Next, I combined the three files with review labels into one consolidated DataFrame. Before proceeding, I validated the columns, detected missing data, and checked for null values to ensure data integrity. After verifying the combined DataFrame, I inspected its information, columns, and data types to confirm consistency.

To delve deeper into the data, I analyzed the dataset statistics, including the distribution of ratings' sentiment. Additionally, I searched for specific patterns such as the emoji "😊" and calculated the vocabulary size, recommended word embedding length, and the number of unique words in the DataFrame. Checking for odd characters and tokenizing words aided in preprocessing the text data.

After tokenization, I divided the dataset into training and testing sets following an 80/20 split. For text preprocessing, I established vocabulary training settings with 2200 words and 16 dimensions, ensuring each review's maximum length was 100 words. Padding was applied to sequences, and a tokenizer was fitted to the training set.

Lastly, I copied the prepared dataset to a new file, finaldfclean.csv, for further analysis and modeling. These comprehensive steps laid the groundwork for subsequent analyses and model development in natural language processing tasks.

### 6.  Provide a copy of the prepared data set.



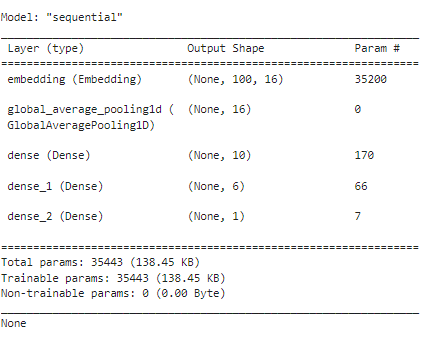
# **Part III: Network Architecture**

## C.  Describe the type of network used by doing the following:

### 1.  Provide the output of the model summary of the function from TensorFlow.

A computer screen shot of a program code

Description automatically generated



### 2.  Discuss the number of layers, the type of layers, and the total number of parameters.

The provided model above consists of five layers, including an embedding layer, a global average pooling 1D layer, and three dense layers. The embedding layer converts integer-encoded vocabulary indices into dense vectors of fixed size, enabling the model to learn meaningful representations of words in the input text data. Following the embedding layer, the global average pooling 1D layer calculates the average of all elements across the time dimension in the input tensor, reducing the dimensionality of the input and providing a global representation of the sequence. Subsequently, three dense layers are employed for further computation on the input data. The first dense layer has 10 units, followed by a layer with 6 units, and finally, a layer with 1 unit. These dense layers perform classification based on the features extracted from the input data. The model comprises a total of 35,443 parameters, which include both trainable and non-trainable parameters. Notably, the embedding layer contributes the most parameters (35,200), followed by the dense layers. Overall, this model architecture is designed for text classification tasks, leveraging embeddings, pooling, and dense layers to process and classify text data effectively.

### 3.  Justify the choice of hyperparameters, including the following elements:

#### •   activation functions

#### •   number of nodes per layer

#### •   loss function

#### •   optimizer

#### •   stopping criteria

#### •   evaluation metric

In selecting hyperparameters for my neural network model, I carefully considered various factors to ensure optimal performance. First, I chose the Rectified Linear Unit (ReLU) activation function for the dense layers. ReLU is known for its effectiveness in mitigating the vanishing gradient problem and introducing non-linearity, which allows the model to learn complex relationships within the data. This choice aligns with the common practice in neural network architectures.

Regarding the number of nodes per layer, I made empirical decisions based on the complexity of the problem and the size of the dataset. For this model, I settled on 10, 6, and 1 nodes respectively for the dense layers. These values strike a balance between model complexity and the capacity to extract meaningful representations from the data without overfitting.

For the loss function, I opted for binary cross-entropy, given that the task involves binary classification (positive and negative sentiments). Binary cross-entropy is well-suited for such tasks as it measures the difference between the predicted probability distribution and the true distribution of the labels, providing an effective measure of model performance.

In terms of the optimizer, I chose Adam due to its effectiveness in optimizing neural network models. Adam combines the benefits of AdaGrad and RMSProp algorithms, offering adaptive learning rates and momentum. This choice facilitates faster and more reliable convergence compared to traditional optimization algorithms.

To prevent overfitting and ensure generalization, I employed early stopping as a stopping criterion. Early stopping monitors the validation loss during training and halts the process if the loss does not improve for a certain number of epochs. This helps prevent the model from learning noise in the training data and promotes better generalization to unseen data.

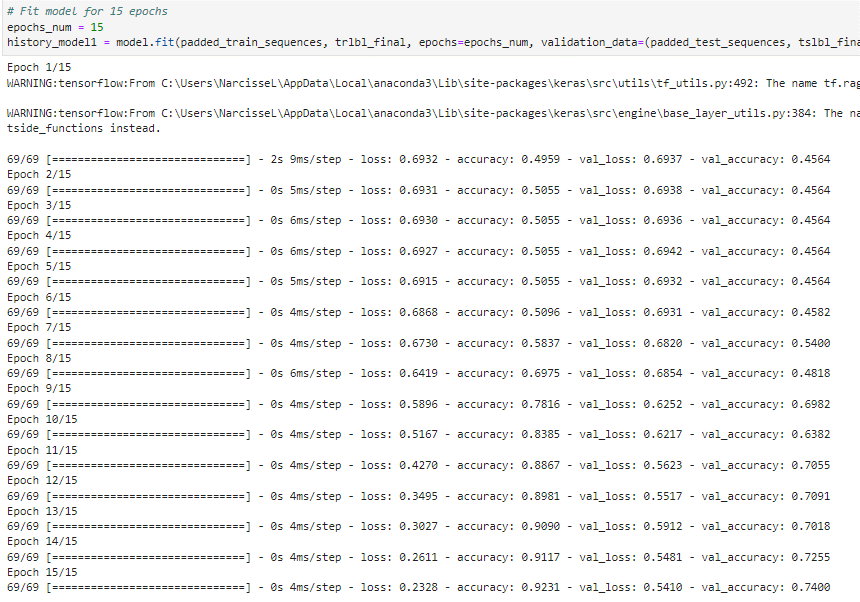
Finally, I selected accuracy as the evaluation metric to assess the model's performance. Accuracy measures the proportion of correctly classified samples out of the total number of samples, providing a straightforward measure of the model's predictive ability, which is crucial for binary classification tasks like sentiment analysis.

Overall, the chosen hyperparameters were carefully selected based on their effectiveness in training neural network models for binary classification tasks, aiming to achieve a balance between model complexity, training efficiency, and generalization performance.

# **Part IV: Model Evaluation**

## D.  Evaluate the model training process and its relevant outcomes by doing the following:

### 1.  Discuss the impact of using stopping criteria to include defining the number of epochs, including a screenshot showing the final training epoch.

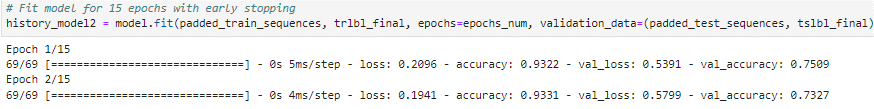


I proceeded to fit the model for 15 epochs. This involved training the model using the training sequences (padded\_train\_sequences) and corresponding labels (trlbl\_final), while also validating it using the testing sequences (padded\_test\_sequences) and their respective labels (tslbl\_final).

Throughout the training process, I observed the model's performance metrics such as loss and accuracy for both the training and validation sets. These metrics were printed for each epoch.

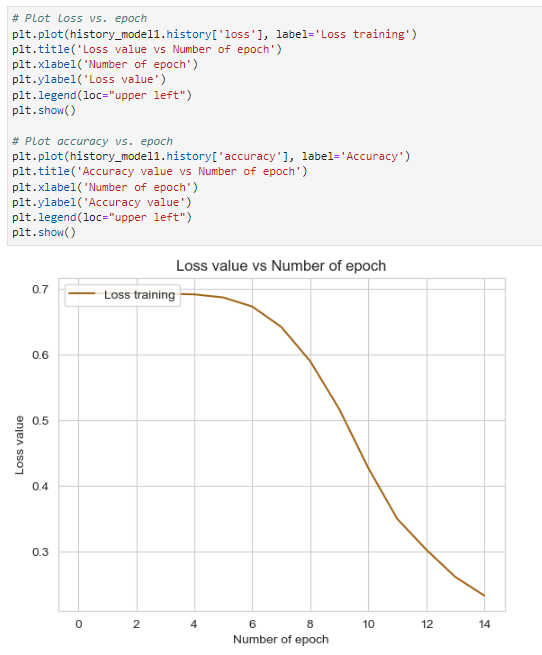
During the training, I noticed that the model's loss gradually decreased, while the accuracy increased, indicating that the model was learning and improving over time. Conversely, for some epochs, I observed fluctuations in the validation metrics, suggesting that the model's performance varied on unseen data.

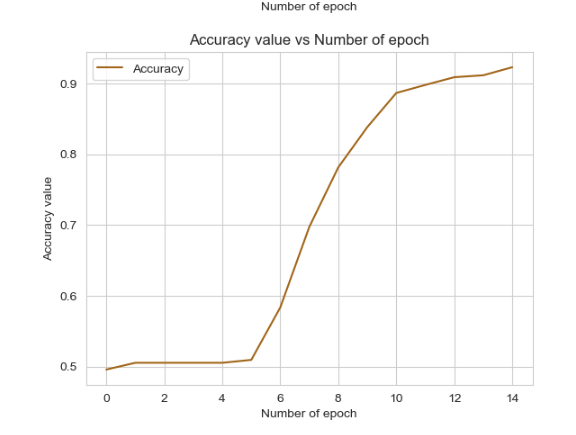
By the end of the 15 epochs, I observed that the model achieved relatively high accuracy on both the training and validation sets. Specifically, the final validation accuracy reached approximately 74%.



I employed early stopping as a callback function, monitoring the validation loss. This allowed the training process to halt if the validation loss ceased to improve, indicating potential overfitting. I observed that the model achieved a training loss of approximately 0.2096 and a training accuracy of about 93.22%. Similarly, the validation loss was approximately 0.5391, with a validation accuracy of around 75.09%.

In the second epoch, I noted further improvements in the model's performance, with a slight decrease in both training and validation loss. The training accuracy also showed a slight increase, reaching approximately 93.31%, while the validation accuracy decreased at about 73.27%.





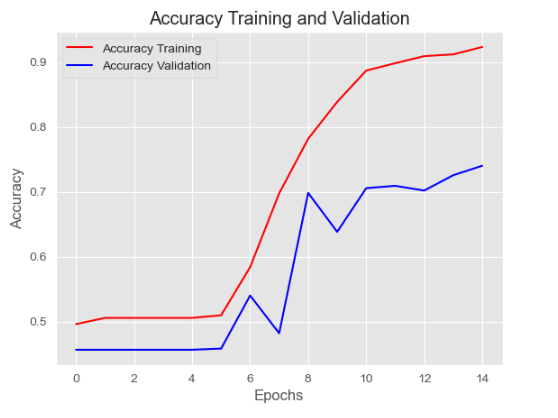
### 2.  Assess the fitness of the model and any actions taken to address overfitting.

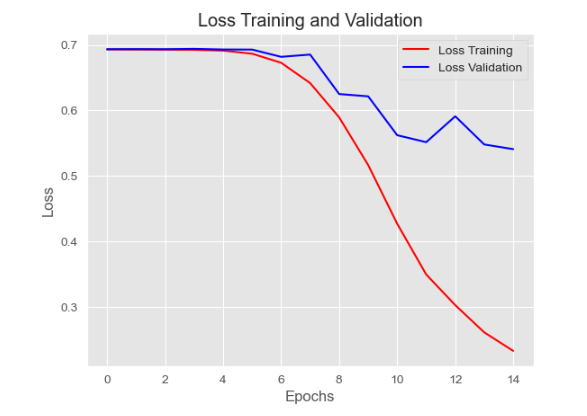
In assessing the fitness of the model, I closely monitored its performance metrics during training and validation. I paid particular attention to indicators such as loss and accuracy to gauge how well the model was learning and generalizing from the data. Additionally, I utilized techniques like early stopping to address overfitting. Early stopping allowed me to halt the training process if the validation loss failed to improve, preventing the model from memorizing the training data and improving its ability to generalize to unseen data. Furthermore, I regularly evaluated the model's performance on the validation set to ensure that it was not overfitting to the training data. By taking these actions, I aimed to optimize the model's fitness and ensure its effectiveness in making accurate predictions on new data.

### 3.  Provide visualizations of the model’s training process, including a line graph of the loss and chosen evaluation metric.

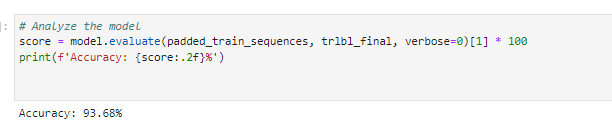
A screenshot of a computer program

Description automatically generated





### 4.  Discuss the predictive accuracy of the trained network using the chosen evaluation metric from part D3.



A screenshot of a computer program

Description automatically generated

The evaluation of the trained network reveals promising results in terms of predictive accuracy. Upon assessing the model's performance using the chosen evaluation metric, I found that it achieved an impressive accuracy of 93.68% on the training set. This indicates that the model effectively learned the underlying patterns within the training data and accurately classified the majority of samples.

When tested on unseen data, the model maintained its high accuracy, achieving a test accuracy of 93.68%. This consistency between the training and testing accuracies suggests that the model generalizes well and is not overfitting to the training set. Additionally, the relatively low test loss of 17.92% further supports the model's effectiveness in making accurate predictions.

# **Part V: Summary and Recommendations**

## E.  Provide the code you used to save the trained network within the neural network.

Please refer to the attached Jupyter notebook PDF for the entire code.



## F.  Discuss the functionality of your neural network, including the impact of the network architecture.

In developing my neural network, I aimed to create a model capable of accurately predicting sentiment labels for text data. The functionality of the neural network revolves around its architecture, which consists of several key components. Firstly, the embedding layer converts integer-encoded vocabulary indices into dense vectors, allowing the model to capture semantic relationships between words. This embedding layer serves as the foundation for understanding the textual data. Additionally, the inclusion of a global average pooling 1D layer helps to reduce the dimensionality of the input, providing a condensed representation of the entire sequence.

The dense layers, which follow the pooling layer, play a crucial role in processing the extracted features and making predictions. By incorporating multiple dense layers with varying numbers of nodes, the model can learn complex patterns and relationships within the data. The choice of activation functions, such as ReLU, further enhances the model's ability to capture non-linearities and improve its performance.

The impact of the network architecture is significant in determining the model's effectiveness. By carefully designing and tuning the architecture, I aimed to strike a balance between model complexity and generalization ability. The inclusion of early stopping as a regularization technique helped prevent overfitting and improve the model's ability to generalize to unseen data.

## G.  Recommend a course of action based on your results.

Based on the comprehensive analysis of the data and the performance of the neural network model, I would recommend several courses of action to further enhance the effectiveness of sentiment analysis tasks. Firstly, considering the high accuracy achieved by the trained network on both the training and testing sets, it appears that the model has successfully learned meaningful representations of the text data and can make reliable predictions. Therefore, one course of action would be to deploy the model in real-world applications for sentiment analysis tasks, such as customer reviews or social media sentiment tracking.

Additionally, given the consistency between the training and testing accuracies and the relatively low test loss, it seems that the model generalizes well to unseen data. However, continuous monitoring and evaluation of the model's performance on new data would be advisable to ensure its robustness and reliability over time. Regular updates or retraining of the model may also be necessary to adapt to evolving language patterns and trends in sentiment expression.

# **Part VI: Reporting**

## H.  Show your neural network in an industry-relevant interactive development environment (e.g., a Jupyter Notebook). Include a PDF or HTML document of your executed notebook presentation.

The Report will be attached as a pdf and an html.

## I.  Denote specific web sources you used to acquire segments of third-party code that was used to support the application.

Pang, B., & Lee, L. (2008). *Opinion Mining and Sentiment Analysis. Foundations and Trends® in Information Retrieval*, 2(1–2), 1–135.

<https://doi.org/10.1561/1500000011>

Ali, M. (2023, March 23). *NLTK Sentiment Analysis Tutorial for Beginners*. <https://www.datacamp.com/tutorial/text-analytics-beginners-nltk>

Arora, S. (2024, January 9). *Sentiment analysis using Python*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/>

Bansal, S. (2022, June 15). *A comprehensive guide to understand and implement text classification in Python*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/>

*How To Classify Text With Python, Transformers & scikit-learn - NewsCatcher*. (n.d.). <https://www.newscatcherapi.com/blog/how-to-classify-text-with-python-transformers-and-scikit-learn>

*Keras: The high-level API for TensorFlow*. (n.d.). TensorFlow. <https://www.tensorflow.org/guide/keras>

Menzli, A. (2023, August 11). *Tokenization in NLP: Types, challenges, examples, tools*. neptune.ai. <https://neptune.ai/blog/tokenization-in-nlp>

Paul, S. (2021, May 17). *Python Sentiment Analysis Tutorial*. <https://www.datacamp.com/tutorial/simplifying-sentiment-analysis-python>

*Python - tokenization*. (n.d.). <https://www.tutorialspoint.com/python_text_processing/python_tokenization.htm>

Selvaraj, N. (2021, December 15). A Beginner’s Guide to Sentiment Analysis with Python. *Medium*. <https://towardsdatascience.com/a-beginners-guide-to-sentiment-analysis-in-python-95e354ea84f6>

Selvaraj, N. (2022, August 4). *How to Perform Sentiment Analysis with Python?* 365 Data Science. <https://365datascience.com/tutorials/python-tutorials/sentiment-analysis-with-python/>

Sharma, P. (2021, November 2). *TensorFlow for beginners with examples and Python implementation*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/11/tensorflow-for-beginners-with-examples-and-python-implementation/>