**D213 Performance Assessment Task 1**

**SENTIMENT ANALYSIS FOR AMAZON REVIEWS**

Fahim A. Akbar Student ID 001434895 Masters Data Analytics (January 1, 2021) Program Mentor: Mandy Rasmuson (385) 428-5957x5957 [fakbar3@wgu.edu](mailto:fakbar3@wgu.edu)

**Part I: Research Question**

**A. Sentiment Analysis Purpose**

**A1. Research Question:**

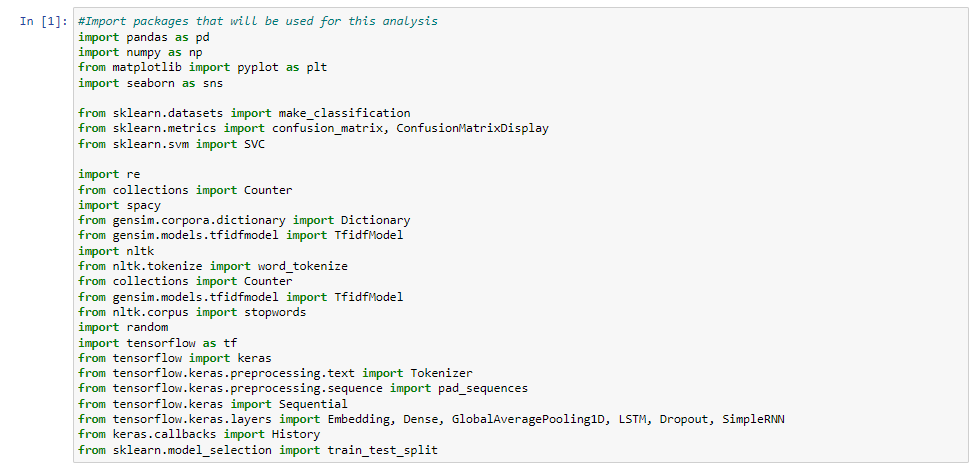
The research question that this analysis will focus on is if a neural network can accurately predict whether or not a customer would recommend a product or service, based on the reviews that the customers wrote.

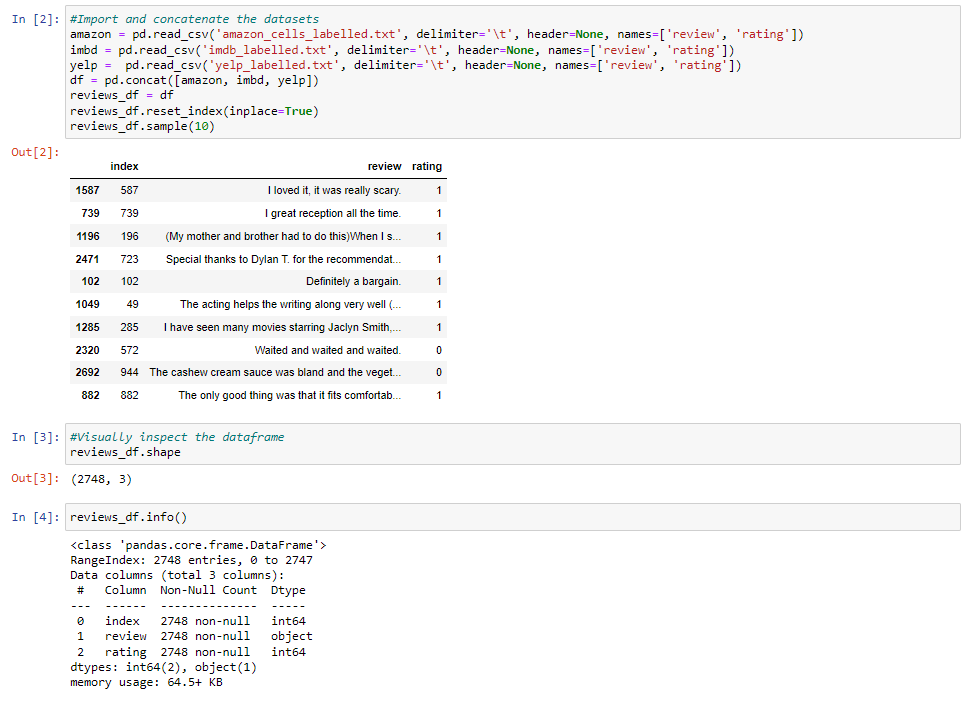
**A2. Goal**

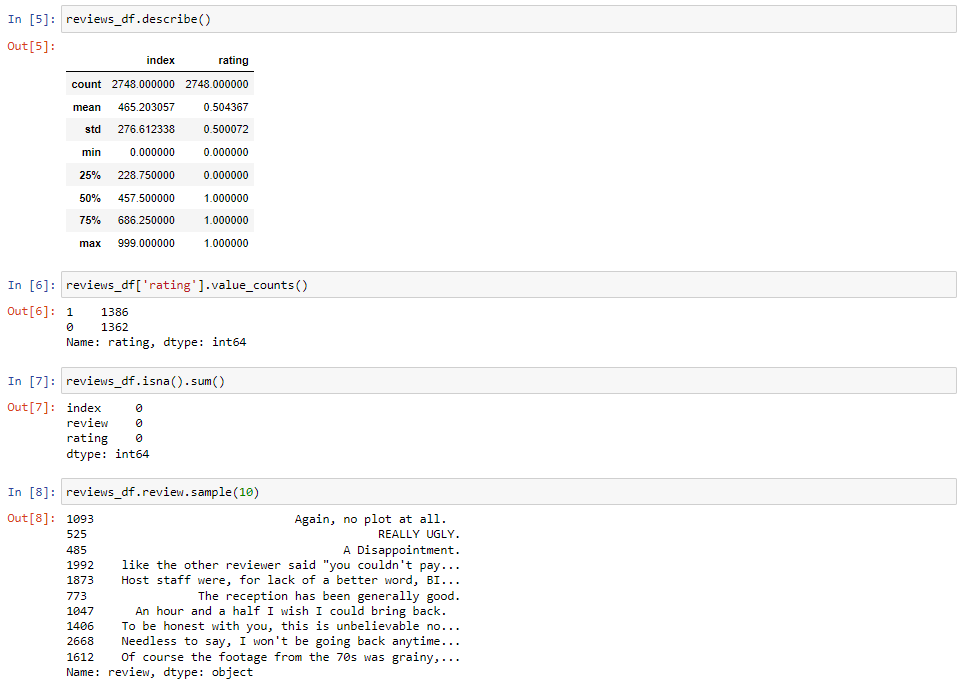
The goal of this data analysis is to create a neural network model capable of analyzing customer review text and assigning a score indicating its quality, represented as either good (1) or bad (0). This will be done by building a neural network model for training and evaluating a dataset that classifies text into positive and negative customer reviews across Amazon, IMDb, and Yelp.

**A3. Neural Network Type**

A Recurrent Neural Network (RNN) will be used for text classification and prediction generations on text sequences from the concatenated data. A Recurrent Neural Network (RNN) is a bi-directional artificial neural network designed to process sequential data. A Recurrent Neural Network (RNN) processes sequential data by maintaining hidden states, allowing them to capture patterns in sequences. In sentiment analysis, RNNs are used to analyze text data, considering the sequential nature of language. By processing words or phrases one at a time and retaining memory of previous inputs, RNNs can capture contextual dependencies (Li, 2018). This makes RNNs effective for understanding sentiment in longer textual contexts such as reviews, and is why I’ll be using an RNN for this analysis. The development of the model involves constructing a Sequential Keras model using Tensor Flow. The model implementation begins with an embedding layer, succeeded by several dense hidden layers. This architecture enables the model to learn from the organized text data, enabling it to discern relationships within the data. Subsequently, calculations are made based on these relationships, and a final output layer generates a predicted value for the review sentiment.







**Part II: Data Preparation**

**B. Data Exploration and Cleaning**

**B1. Exploratory Data Analysis**

After the 3 datasets were concatenated into 1, we begin explorative data analysis. For the purpose of this assessment, the dataset was assigned the name “reviews\_df.”



**Unusual characters:** This data contains all 26 English letters, all digits 0-9, and 25 special or non-English characters.

**Vocabulary size:** The vocabulary size for this dataset is 4,764 words

**Proposed word embedding length:** The embedding for this model requires two different dimensions, an input dimension of the word embedding length and an output dimension set as the maximum sequence length equal to the vocabulary size (4,764) and an output dimension. (Refer to Input 12 of the Jupyter notebook)

**Statistical justification for the chosen maximum sequence length:** For this model, the maximum sequence embedding length will be equal to the 4th root of the vocabulary size, which is 8. (Arat, 2019)

Maximum sequence embedding length = 4674 (vocabulary size) x 0.25;

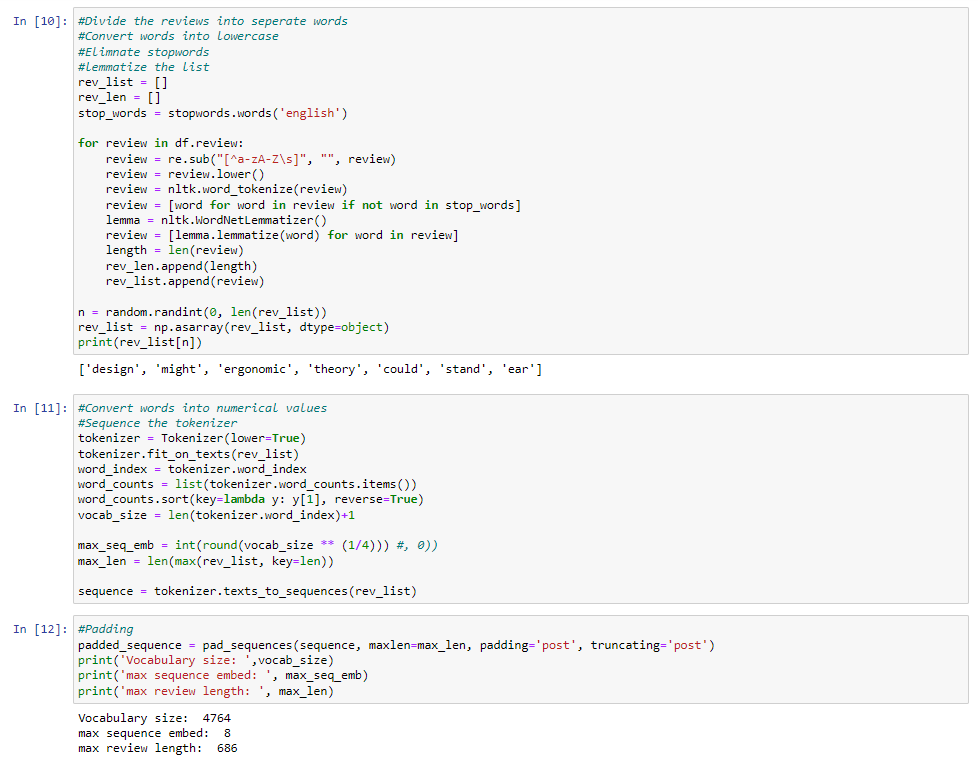
Maximum sequence embedding length = 8

While this is general guideline, I’ve had success in this model using the 4th root of the number of categories to base the embedding vector size.

**B2. Tokenization Process Goals**

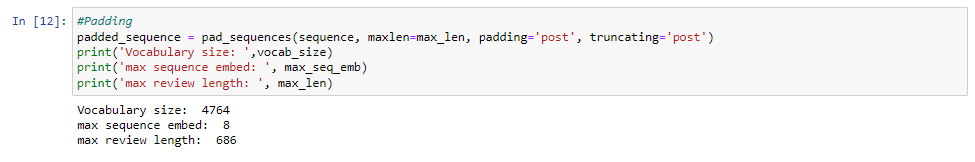
Tokenization is done to preprocess text data for machine learning. It involves cleaning the text by removing stop-words and punctuation marks. Furthermore, Tokenization assigns a unique token to each word and breaks down sentences into individual words for classification purposes. First, every review is divided into a list of separate words. These words are subsequently converted into lowercase, eliminating the impact of capitalization and ensuring text standardization. From each tokenized list, stop words are eliminated, leaving behind a refined list comprising unique words that provide essential information and context to the text. Next, the list is lemmatized to ensure that different forms of a word are unified. In order to ready the tokenized text for input into the neural network, each word needs to be converted into a numerical value. This is achieved by assigning an index number to each word in the vocabulary. These numerical values are then used to replace the corresponding words in the list.

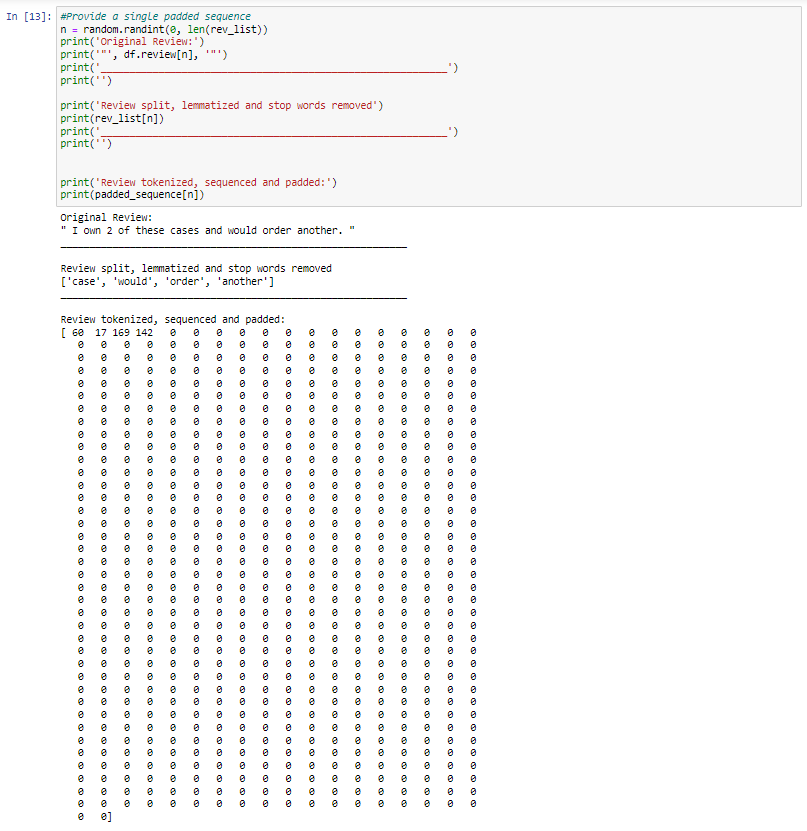
The tokenization of the data is showcased below:



**B3. Padding Process**

Due to the differing lengths and word counts in each review, padding will be done so the length of the resulting sequence of indexes will be more uniform. This is achieved by adding 0 values to the list until all reviews have the same length. In this model, padding will occur at the end of the sequences.

Here is an example of a single padded sequence:



**B4. Sentiment Categories**

In this data set, the reviews are made up of binary data, where 0 indicates a negative review and 1 indicates a positive review. The binary nature of the dataset makes binary cross-entropy a suitable choice for a loss function, especially in the context of binary classification. Furthermore, the model must produce a binary output for it to be compatible with the specified loss function during compilation. To achieve this, the model requires a final dense layer with a single output unit. Because the output consists of just one unit, which must be either 0 or 1, the ideal activation for this final layer is "sigmoid.” This will return a value ranging from 0 to 1.

**B5. Data Preparation Steps**

By now, the review data has been tokenized, sequenced, and padded into an array that can be utilized for a neural network model. For model training, the data will be split into training and testing sets. 80% of the data will be allocated for training purposes, while the remaining 20% will be utilized for testing. In addition, a validation set is required for model learning, and will be applied within the model's fit method. I’ll be utilizing the "validation\_split" parameter set to 0.2, which represents 20% of the data. This ensures that training data will be shuffled between each epoch so the model can train and validate different arrangements of the dataset on each epoch.

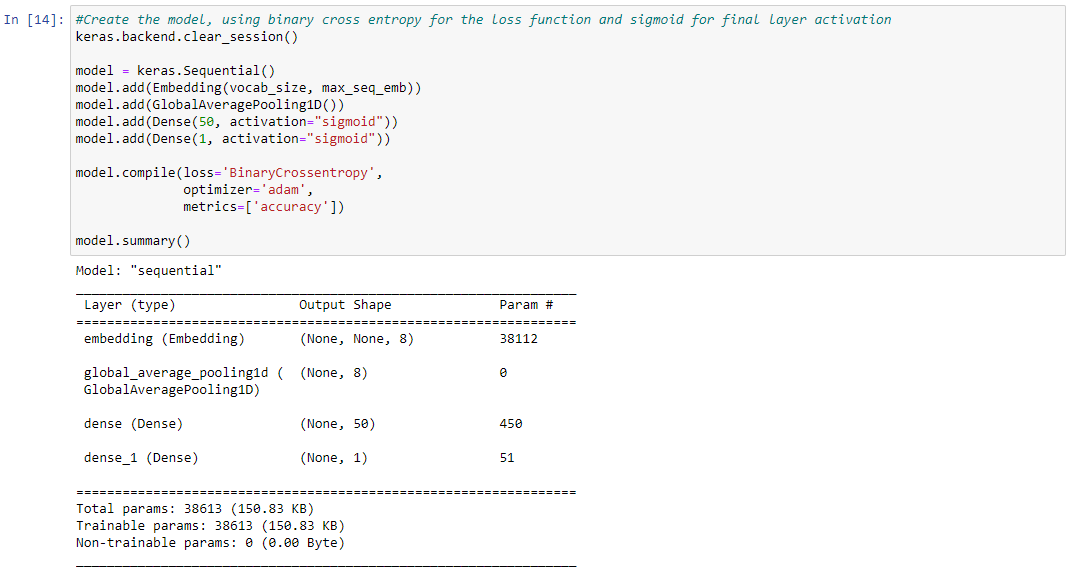
**B6. Copy of Prepared Data Set**

The preparation of the dataset was performed in Python using a Jupyter notebook environment. The Jupyter notebook file is attached to the task submission. A pdf copy of the notebook and a txt. file of code used is provided with the task submission as well.

**Part III: Network Architecture**

**C. Neural Network**

**C1. Output of the Model Summary**



**C2. Model Layers**

The neural network I created was build as a sequential model from Keras, and has four layers. The first layer is an embedding layer with the input dimension matching the vocabulary size and the output dimension set to previously calculated maximum sequence embedding length of 8. This layer contains 38,112 parameters. After the embedding layer, there is a Global Average Pool 1D layer, which is used for down sampling of the input representation by taking the maximum value over the time dimension (Brownlee, 2019). The last two layers are two Dense layers using sigmoid activations: the initial one with 50 nodes, and the second serving as the output layer with a single node. Overall, the model has 38,613 trainable parameters.

**C3. Hyperparameters and Evaluation metric:**

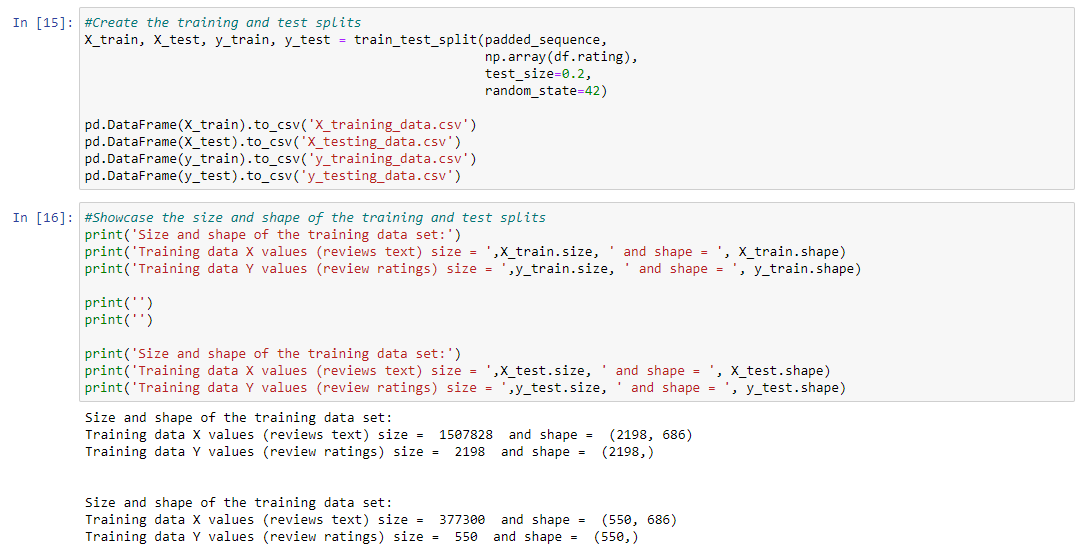
**Activation functions:** Only the two dense layers need an activation function. In both instances, the sigmoid function is employed. The S-shaped curve function produces outputs between 0 and 1 that centered on 5, aligning well with the model's output requirements. Number of nodes per layer: The input layer is the embedding layer, and needs nodes equivalent to the vocabulary size. The Global Average Pool 1D layer doesn't need a specific node count. In the subsequent Dense layer, 50 nodes were chosen after experimenting with values between 5 and 200, aiming for model simplicity. Further nodes beyond 50 didn't significantly enhance accuracy due to trial and error. The final layer is the output layer, and only requires one node. This was determined by using binary cross entropy for the loss function.

**Loss function:** The selected loss function for this model was Binary Cross Entropy. This was chosen due to its efficiency in predicting binary values that align with the target values of 0 or 1 based on the reviews in the original dataset. (cite BCE)

**Optimizer:** The Adam optimizer (adaptive moment estimation) was chosen for this model due to its widespread acceptance in various neural networks. This efficient function is adaptable to diverse data types and shapes, requiring minimal parameter changes. (CITE Adam)

**Stopping criteria:** In this model, the Early Stopping callback is used with a patience setting of 5. To make sure that the model comprehensively trains through the available dataset, a large number of epochs were selected. However, the large volume of epochs could lead to prolonged runtime and overfitting. To mitigate overfitting, various patience values ranging from 0 to 100 were experimented with, and a value of 5 proved optimal. This choice allowed ample time for the model to train while preventing overfitting by stopping before the loss and validation began to diverge.

**Evaluation metric:** Because the goal of the model is predicting the likelihood of a negative or positive review, accuracy was deemed the most important, and was used as an evaluation for training.

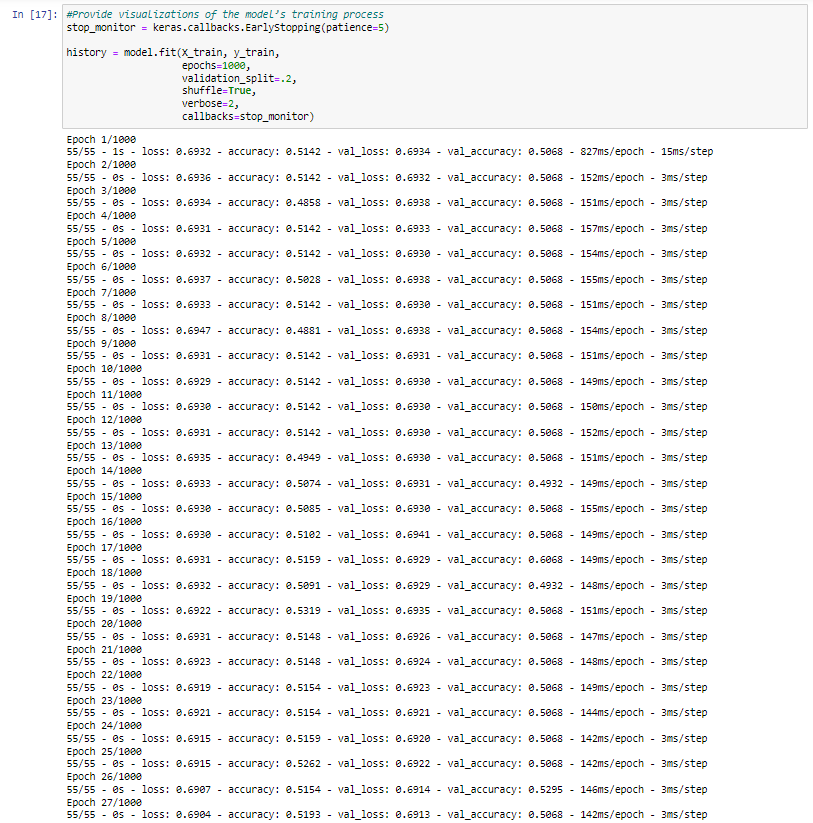


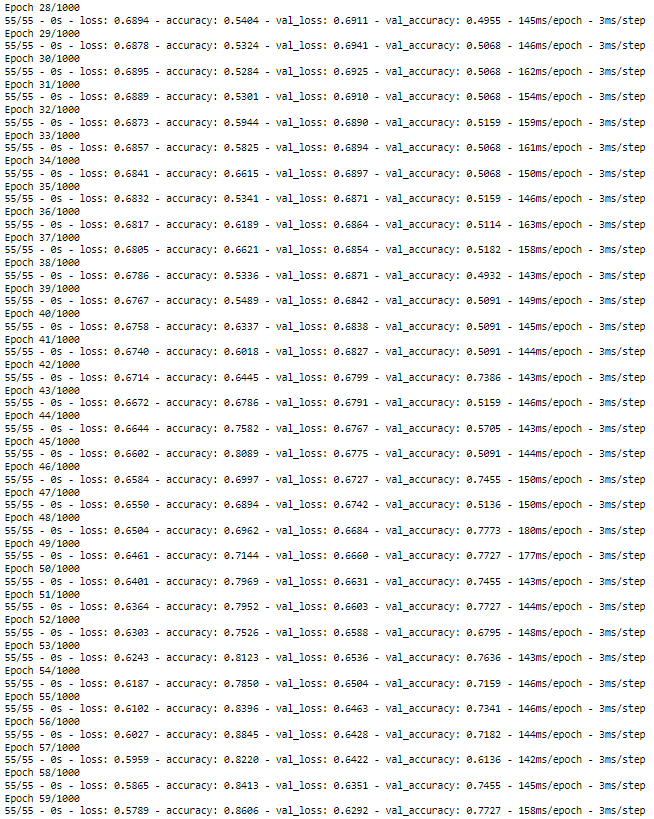
**Part IV: Model Evaluation**

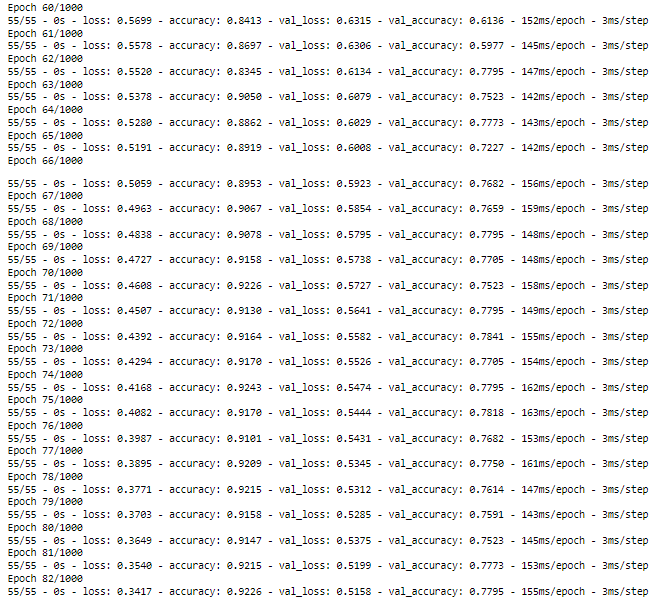
**D. Evaluation of model training process and outcomes**

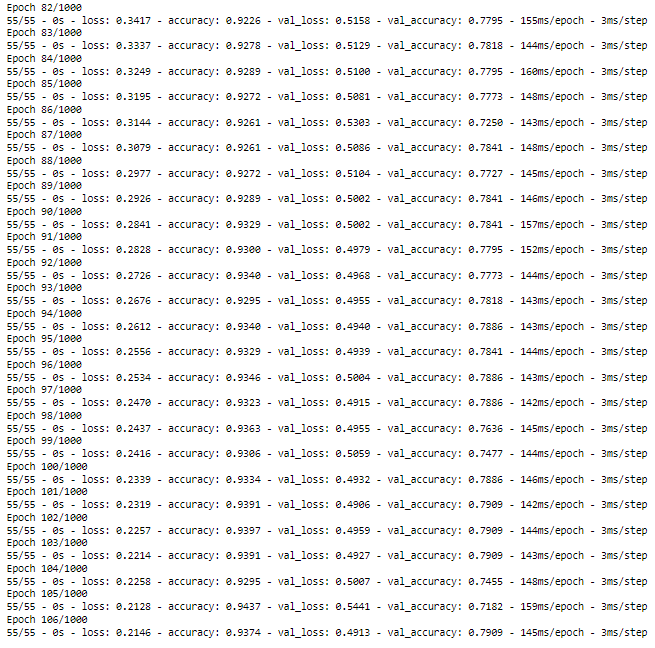
**D1. Impact of Stopping Criteria**

Having the stopping criteria established, as shown in C3, allowed for a sufficient number of epochs for effective training. Furthermore, this helped reduce loss, and enhanced accuracy without the risk of excessive training that could have lead to overfitting or surpassing the point of improvement. For this model, the maximum number of epochs was set at 1000. However, based on the defined stopping criteria, the training process concluded by epoch 106. The screenshot below shows the final training epoch:







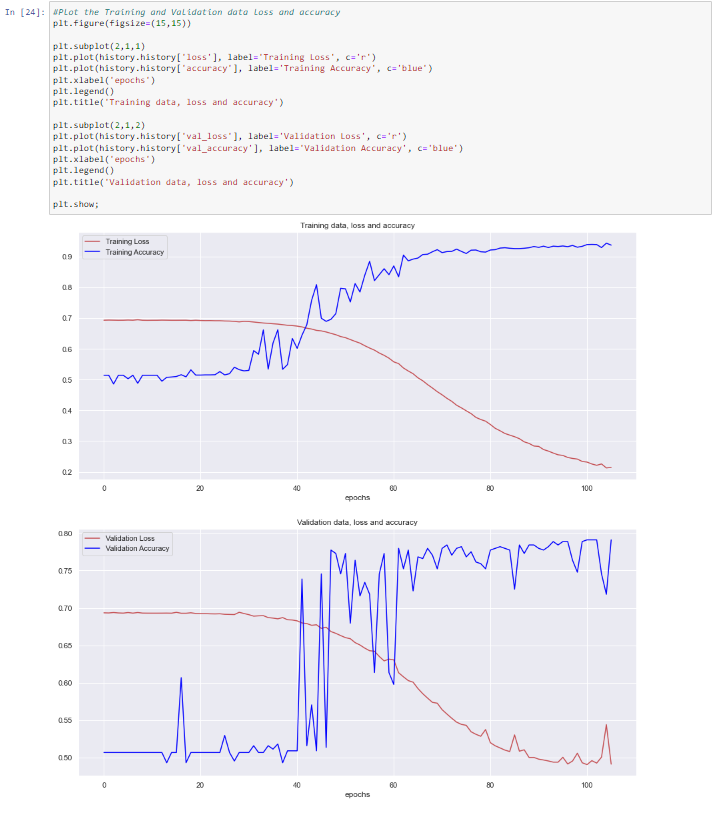


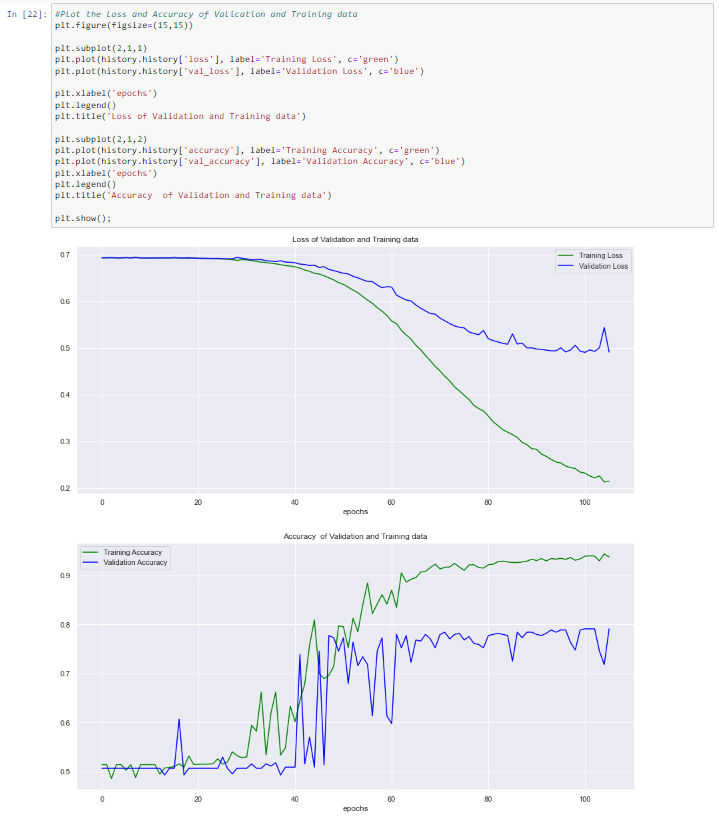
**D2. Model Fitness**

The model appears to be fit without any major signs of overfitting. A primary measure for avoiding overfitting was using validation data in conjunction with the training data for model fitting. This allowed for examination of validation data alongside each epoch, making any patterns apparent that would provide insight into whether overfitting is occurring. A crucial pattern to observe in this model was the relationship between loss values and validation data loss values. While loss values should decrease, the onset of overfitting causes the validation data loss values to diverge from the training loss values and start rising. To prevent overfitting in this model, complexity was introduced only if it enhanced accuracy and did not exhibit signs of overfitting. Furthermore, deploying a stopping criterion as shown in sections C3 and D1 was crucial to prevent overfitting. This restricted the model from continuing training beyond a point where further improvements were unattainable, thereby mitigating the risk of overfitting.

**D3. Model Visualizations**

Visualizations of the model’s training process are showcased below. These include a line graph of the loss and the chosen evaluation metric, being a confusion matrix.

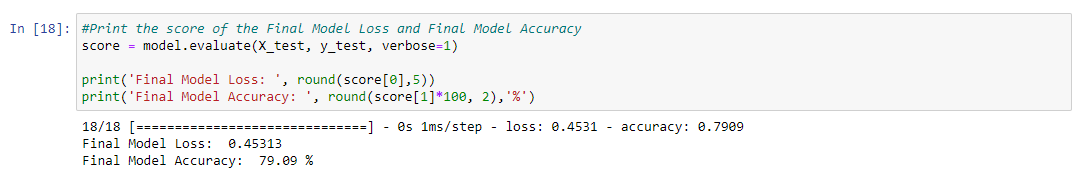






**D4. Trained network predictive accuracy**

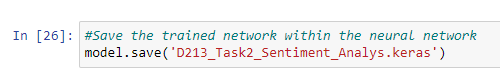
The following code was used to prince the score of the final model loss and final model accuracy:



The model generated could accurately predict the correct outcome 79.09% of the time when used against test data. The confusion matrix from section D3 was generated to see how accurately the model predicted each review outcome. For each response option, it correctly predicted a positive review (1) 204 out of 259 times (78.8%), and for negative reviews, it correctly predicted a 0 value 231 out of 291 times (79.4%).

**Part V: Summary and Recommendations**

**E. Code used to save the trained network with neural network**



The code used for saving the trained network with the neural network is provided above and was performed in Python using a Jupyter notebook environment.

**F. Neural Network Functionality**

The neural network was trained using 2,198 text reviews and their corresponding scores. It was successful in accurately predicting future scores from text inputs, and achieved a prediction accuracy of 79.09% based on 550 test data observations. Given the high level of accuracy, this model can now be used to predict scores based on new customer reviews and further aid in business decision making strategies. Since customer satisfaction is imperative in nearly all businesses, it is essential to recognize and rectify any internal issues that impact customer contentment and satisfaction with the company and its products and services. To my surprise, the model was slighting more accurate in predicting negative reviews by a factor of 1%. This model can be further trained on negative reviews to help pinpoint problematic aspects by analyzing predicted scores from customer feedback and reviews. Additionally, it can identify the company's strengths, providing valuable insights for allocating additional resources or conducting further studies to sustain ongoing success. The model's network architecture is tailored for sentiment analysis, leveraging the suitability of an RNN model for text classification. The layers are specifically designed to accommodate binary score outputs. Efficient optimizers and loss functions ensure effective training, driving improvements in essential metrics while preventing overfitting.

**G. Recommend course of action**

Because the model shows a high level of accuracy, I recommend that business decision-makers begin using this model for analyzing customer feedback to create reporting mechanisms and metrics for text-based reviews. For a much more fine tuned model, I would highly recommend a larger dataset with more reviews across Amazon, IMDb, and Yelp. The dataset could also include additional company product reviews, such as Steam or TripAdvisor. A larger dataset with higher product and service diversity would also have the added benefit of training the model based an expanded vocabulary and larger library of unique words.

**Part V: Reporting**

**H. Full code used**

The following code was performed in Python using a Jupyter notebook environment. The Jupyter notebook file is attached to the task submission. A pdf copy of the notebook and a text file of code used is provided with the task submission as well.

**I. Sources**

**Works Cited**

Arat, M.M. (2019) *How to use embedding layer and other feature columns together in a network using Keras?*, *Mustafa Murat ARAT*. Available at: https://mmuratarat.github.io/2019-06-12/embeddings-with-numeric-variables-Keras#:~:text=Embedding%20Dimensionality&text=Jeremy%20Howard%20provides%20a%20general,So%20it’s%20kind%20of%20experimental (Accessed: 20 October 2023).

Brownlee, J. (2019) *A gentle introduction to pooling layers for Convolutional Neural Networks*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/ (Accessed: 20 October 2023).

Brownlee, J. (2021) *How to use word embedding layers for deep learning with keras*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/ (Accessed: 20 October 2023).

*Introducing tensorflow feature columns* (2017) *- Google for Developers Blog - News about Web, Mobile, AI and Cloud*. Available at: https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html (Accessed: 20 October 2023).

Li, S. (2018) *A beginner’s guide on sentiment analysis with RNN*, *Medium*. Available at: https://towardsdatascience.com/a-beginners-guide-on-sentiment-analysis-with-rnn-9e100627c02e (Accessed: 20 October 2023).

**Additional Third-Party Code used:**

<https://www.kaggle.com/code/ngyptr/python-nltk-sentiment-analysis>

<https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer>

<https://stackoverflow.com/questions/51956000/what-does-keras-tokenizer-method-exactly-do>

<https://stackoverflow.com/questions/54493738/keras-difference-between-averagepooling1d-layer-and-globalaveragepooling1d-laye>

<https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea>

<https://www.kaggle.com/code/rafetcan/recurrent-neural-n-rnn-tutorial-for-beginners>

<https://www.kaggle.com/code/thebrownviking20/intro-to-recurrent-neural-networks-lstm-gru>

<https://stackoverflow.com/questions/39883331/plotting-learning-curve-in-keras-gives-keyerror-val-acc>

<https://stackoverflow.com/questions/59840289/model-evaluate-in-keras>

<https://stackoverflow.com/questions/46287403/is-there-a-way-to-implement-early-stopping-in-keras-only-after-the-first-say-1>