D213 Advanced Data Analytics Performance Task 2

Sean Simmons

WDU Data Analytics

MSDA D213

March 2023

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 dataset.

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.

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.

1. One research question that I will answer using neural network models is, "Can we determine customer sentiment from the wording of reviews using neural networks and the NLP technique?" This question is relevant to the real-world organization and sentiment analysis captured in this dataset because we are using a software review dataset to determine if a customer likes a product or not. I will use a tensorflow neural network to answer this question.

2. The goals of this analysis are to use a review dataset to determine whether a customer liked or disliked a product based on the word choices the user chose to use. We plan on

creating a positive or negative sentiment and training our model to place reviews in these categories. These goals are reasonable within the scope of this research question because we will use TensorFlow neural networks to analyze the wording of the reviews. The goals are represented in the data by the wording of the customer reviews and the review scores for software taken from amazon's dataset. Tensorflow and tokenization will serve as our main tools and techniques.

3. A neural network (NN) is a series of algorithms, trying to mimic the human brain, that attempt to identify underlying relationships in a dataset. The type of neural network capable of performing a text classification task that can be trained to produce useful predictions on text sequences in the selected data set is a recurrent neural network (RNN) (Pai 2020). RNN is used for sentiment analysis and text classification tasks that are related to this goal, including sequence labeling and speech tagging. Natural Language Processing (NLP) describes using a computer to manipulate the human language which is customer reviews in this instance. There are several types of NN's that could work, but RNN is the most straight forward and easiest option to use here.

Part II: Data Preparation

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

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

- **presence of unusual characters (e.g., emojis, non-English characters, etc.)**
- **vocabulary size**
- **proposed word embedding length**
- **statistical justification for the chosen maximum sequence length**

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

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

- **if the padding occurs before or after the text sequence**
- **a screenshot of a single padded sequence**

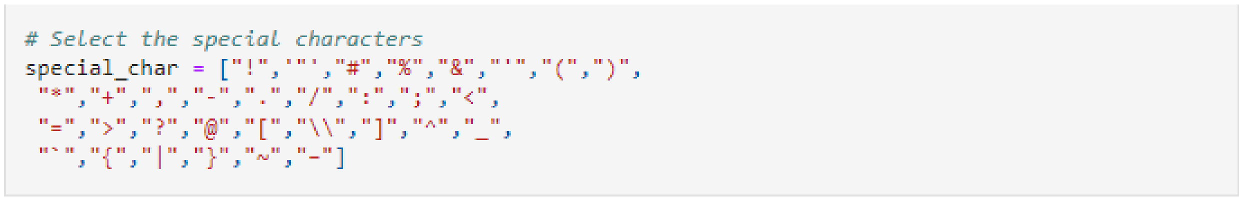
4. Identify how many categories of sentiment will be used and an activation function for the final dense layer of the network.

5. Explain the steps used to prepare the data for analysis, including the size of the training, validation, and test set split.

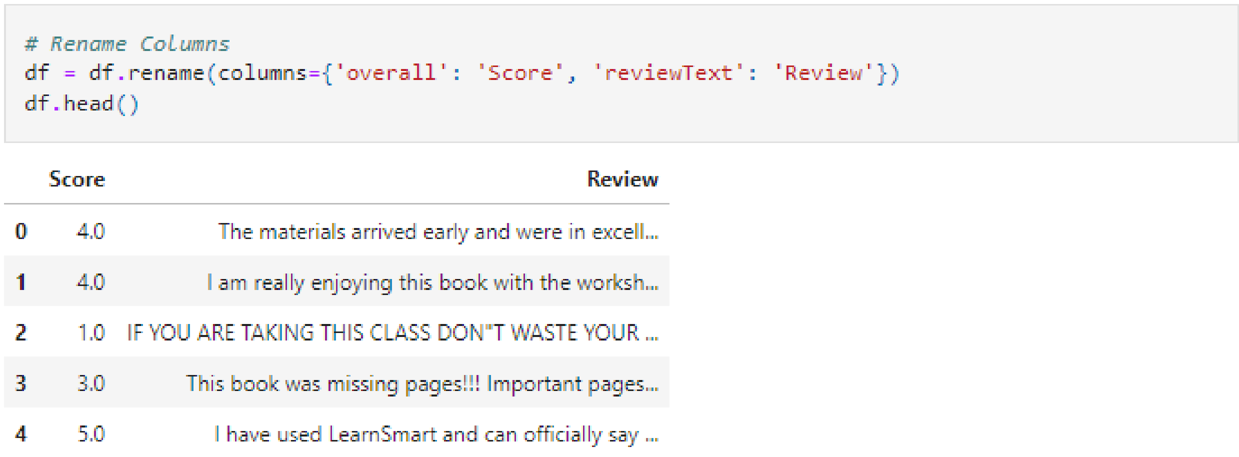
6. Provide a copy of the prepared dataset.

1. The exploratory data analysis is as follows:
    a. Presence of unusual characters (emojis, non-english, etc.)- We have symbols and non-text characters identified in the data table from the first few headers and we

need to go ahead and select all of the possible unusual characters. We can see the quotation mark and exclamation point in the first 5 rows:



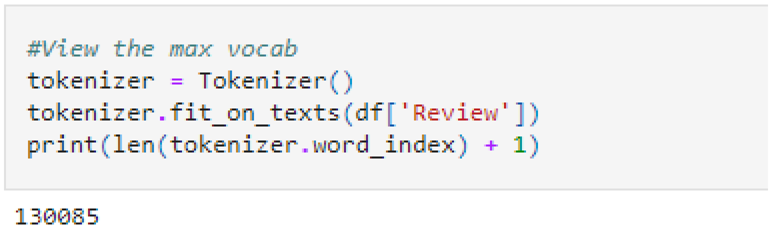
```
# Select the special characters
special_char = ["!",'"',"#","%","&","'","(",")",
 "*","+",",","-",".","/",":",";","<",
 "=",">","?","@","[","\\","]","^","_",
 "`","{","|","}","~","-"]
```



```
# Rename Columns
df = df.rename(columns={'overall': 'Score', 'reviewText': 'Review'})
df.head()
```

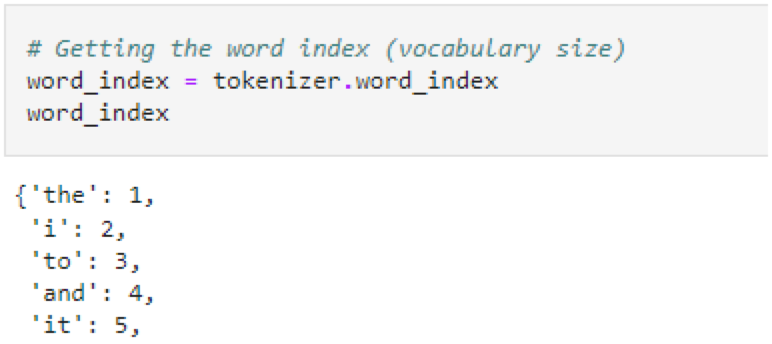
	Score	Review
0	4.0	The materials arrived early and were in excell...
1	4.0	I am really enjoying this book with the worksh...
2	1.0	IF YOU ARE TAKING THIS CLASS DON"T WASTE YOUR ...
3	3.0	This book was missing pages!!! Important pages...
4	5.0	I have used LearnSmart and can officially say ...

b. Vocabulary size- We have reviews of varying lengths, seen through the tokenizer word index function: 130085



```
#View the max vocab
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['Review'])
print(len(tokenizer.word_index) + 1)
```

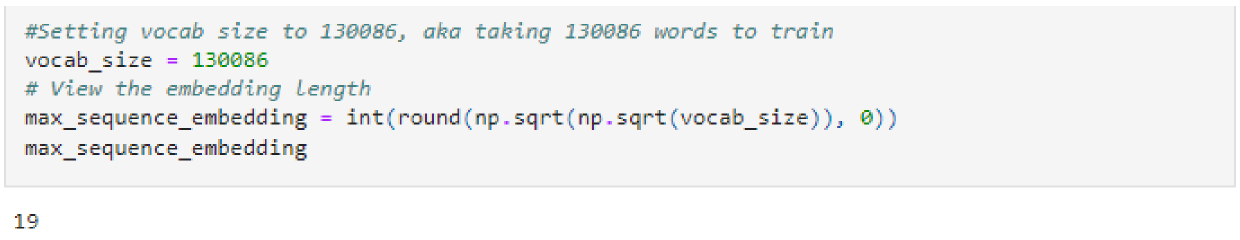
```
130085
```



```
# Getting the word index (vocabulary size)
word_index = tokenizer.word_index
word_index
```

```
{'the': 1,
 'i': 2,
 'to': 3,
 'and': 4,
 'it': 5,
```

c. Proposed word embedding length- Word embedding is the word positive in the vector. The length of word embedding is the fourth root of the vocabulary size. Here we find it to be 19:



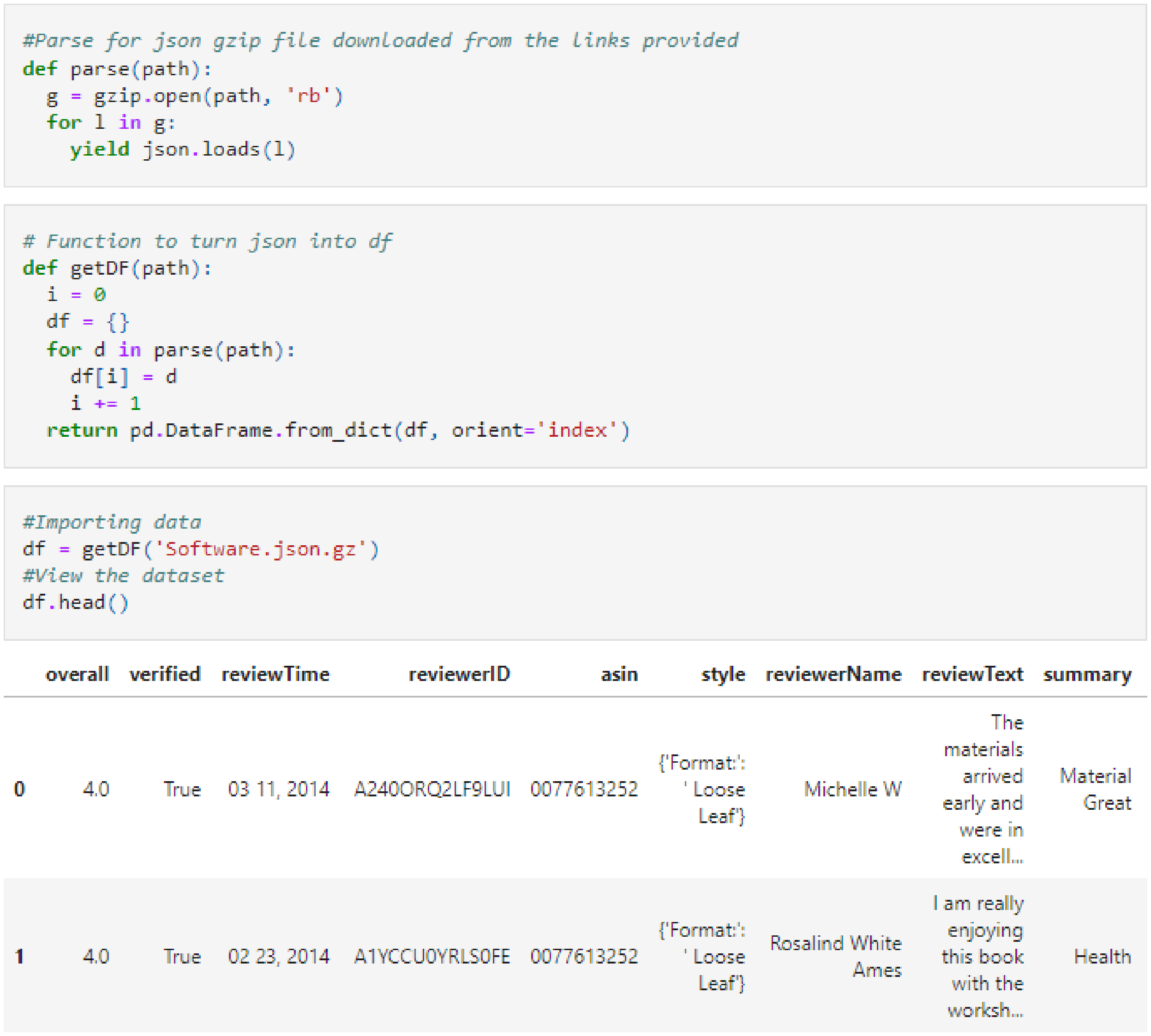
```
#Setting vocab size to 130086, aka taking 130086 words to train
vocab_size = 130086
# View the embedding length
max_sequence_embedding = int(round(np.sqrt(np.sqrt(vocab_size)), 0))
max_sequence_embedding
```

```
19
```

d. Statistical justification for the chosen maximum sequence length is the fourth root of the entire word_idex (vocabulary) size. We want the original available input data so our model is created from the actual dataset and does not skew itself by excluding words. Any inputs shorter than the maximum length will be addressed by padding. Please see the above snippet of code for the max sequence embedding size.

2. The goals of the tokenization are to separate the text into smaller pieces. The process is attempting to change the strings of words each into their own individual string, so one word equals one string. Once this is accomplished we can then run analysis on each word by itself rather than the whole string of words or a sentence. We also needed to replace special characters and pad the sequences.

a. The code generated is found in “D213_Task2_Code.ipynb“. I will post screenshots below as requested in the report rubric. The screenshots include annotations as comments to explain what the specific cell of code is for.



```
#Parse for json gzip file downloaded from the links provided
def parse(path):
  g = gzip.open(path, 'rb')
  for l in g:
    yield json.loads(l)
```

```
# Function to turn json into df
def getDF(path):
  i = 0
  df = {}
  for d in parse(path):
    df[i] = d
    i += 1
  return pd.DataFrame.from_dict(df, orient='index')
```

```
#Importing data
df = getDF('Software.json.gz')
#View the dataset
df.head()
```

	overall	verified	reviewTime	reviewerID	asin	style	reviewerName	reviewText	summary
0	4.0	True	03 11, 2014	A240ORQ2LF9LUI	0077613252	{'Format:': ' Loose Leaf'}	Michelle W	The materials arrived early and were in excell...	Material Great
1	4.0	True	02 23, 2014	A1YCCU0YRLS0FE	0077613252	{'Format:': ' Loose Leaf'}	Rosalind White Ames	I am really enjoying this book with the worksh...	Health

```
#New data frame with the new selected columns we need
df = df[['overall','reviewText']]
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 459436 entries, 0 to 459435
Data columns (total 2 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   overall     459436 non-null  float64
 1   reviewText  459370 non-null  object
dtypes: float64(1), object(1)
memory usage: 10.5+ MB
```

```
# Select the special characters
special_char = ["!",'"',"#","%","&","'","(",")",
 "*","+",",","-",".","/",":",";","<",
 "=",">","?","@","[","\\","]","^","_",
 "`","{","|","}","~","–"]
```

```
# Rename Columns
df = df.rename(columns={'overall': 'Score', 'reviewText': 'Review'})
df.head()
```

	Score	Review
0	4.0	The materials arrived early and were in excell...
1	4.0	I am really enjoying this book with the worksh...
2	1.0	IF YOU ARE TAKING THIS CLASS DON"T WASTE YOUR ...
3	3.0	This book was missing pages!!! Important pages...
4	5.0	I have used LearnSmart and can officially say ...

```
# Check for nulls
df.isna().any()
```

```
Score      False
Review      True
dtype: bool
```

```
# Remove nulls from where it says "True" above this cell
df = df.dropna(subset=['Review'])
```

```python
# Verify we do not have any nulls
df.isna().any()
```

```
Score     False
Review    False
dtype: bool
```

```python
# replace the special characters with spaces
for char in special_char:
 df['Review'] = df['Review'].str.replace(char, ' ',regex=True)
```

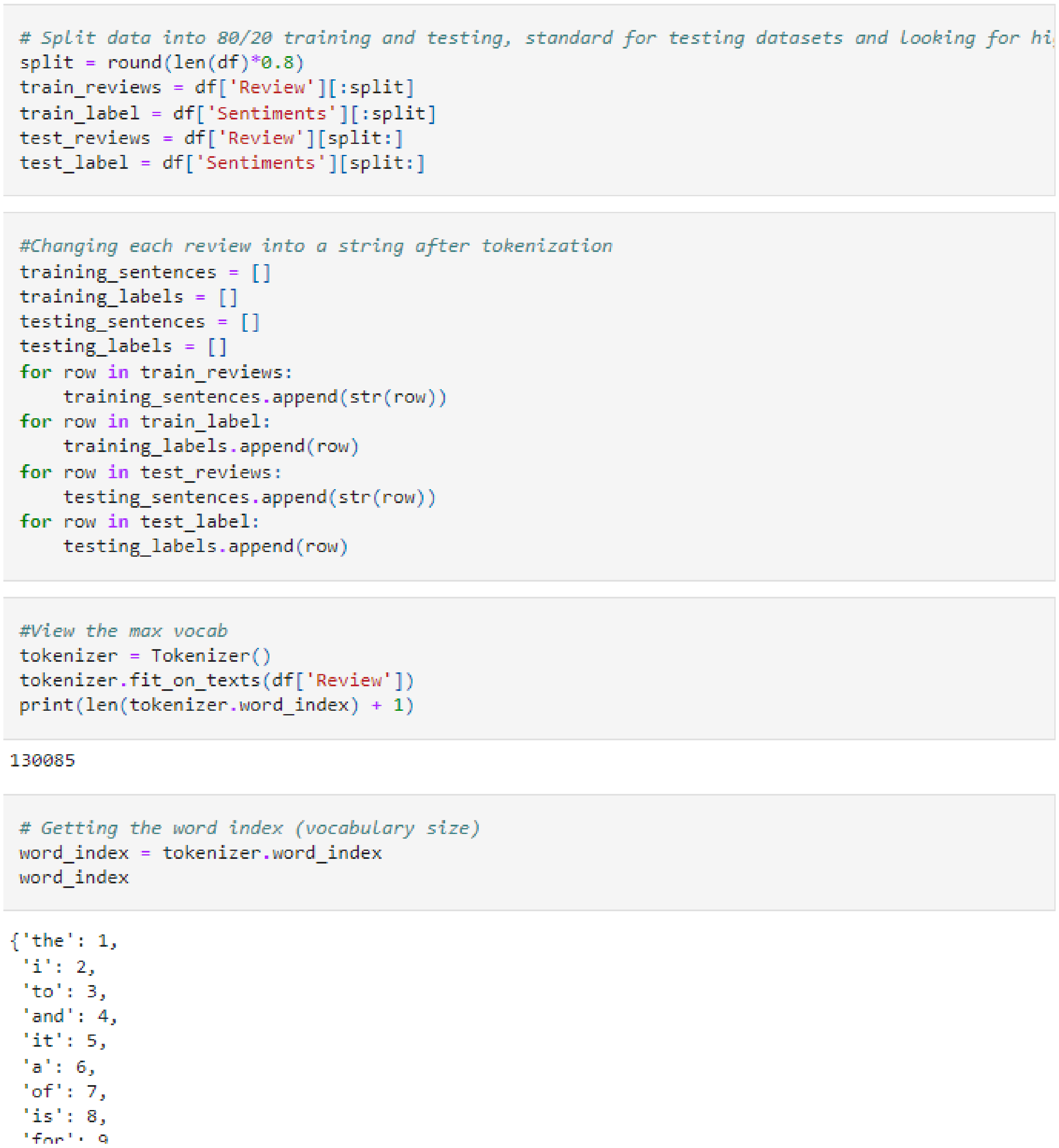
```python
# Replace capitals with lowercase letters
df['Review'] = df['Review'].str.lower()
df.head()
```

	Score	Review
0	4.0	the materials arrived early and were in excell...
1	4.0	i am really enjoying this book with the worksh...
2	1.0	if you are taking this class don t waste your ...
3	3.0	this book was missing pages important pages...
4	5.0	i have used learnsmart and can officially say ...

```python
# Create sentiments column from the dataset (Changing numeric review answers to positive and #
# 0 = negative, 1 = positive
df['Sentiments'] = df.Score.apply(lambda x: 0 if x in [1, 2] else 1)
df.head()
```

	Score	Review	Sentiments
0	4.0	the materials arrived early and were in excell...	1
1	4.0	i am really enjoying this book with the worksh...	1
2	1.0	if you are taking this class don t waste your ...	0
3	3.0	this book was missing pages important pages...	1
4	5.0	i have used learnsmart and can officially say ...	1

```python
# export the prepared data
df.to_csv('D213_Task2_cleaned.csv', index = False)
```



```
# Split data into 80/20 training and testing, standard for testing datasets and looking for hi
split = round(len(df)*0.8)
train_reviews = df['Review'][:split]
train_label = df['Sentiments'][:split]
test_reviews = df['Review'][split:]
test_label = df['Sentiments'][split:]
```

```
#Changing each review into a string after tokenization
training_sentences = []
training_labels = []
testing_sentences = []
testing_labels = []
for row in train_reviews:
    training_sentences.append(str(row))
for row in train_label:
    training_labels.append(row)
for row in test_reviews:
    testing_sentences.append(str(row))
for row in test_label:
    testing_labels.append(row)
```

```
#View the max vocab
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['Review'])
print(len(tokenizer.word_index) + 1)
```

```
130085
```

```
# Getting the word index (vocabulary size)
word_index = tokenizer.word_index
word_index
```

```
{'the': 1,
 'i': 2,
 'to': 3,
 'and': 4,
 'it': 5,
 'a': 6,
 'of': 7,
 'is': 8,
 'for': 9
```

```
#Setting vocab size to 130086, aka taking 130086 words to train
vocab_size = 130086
# View the embedding length
max_sequence_embedding = int(round(np.sqrt(np.sqrt(vocab_size)), 0))
max_sequence_embedding
```

```
19
```

```
# Embedding to 19 dimensions, given from the above code
embedding_dim = 19
# Max length of 150 words per review as a cut off
max_length = 150
#If review is bigger than 150 words, it will be truncated "post" or after the 150th word
trunc_type = 'post'
oov_tok = '<OOV>'
# Padding type "post" meaning each word will receive padding after, not before
padding_type = 'post'
```

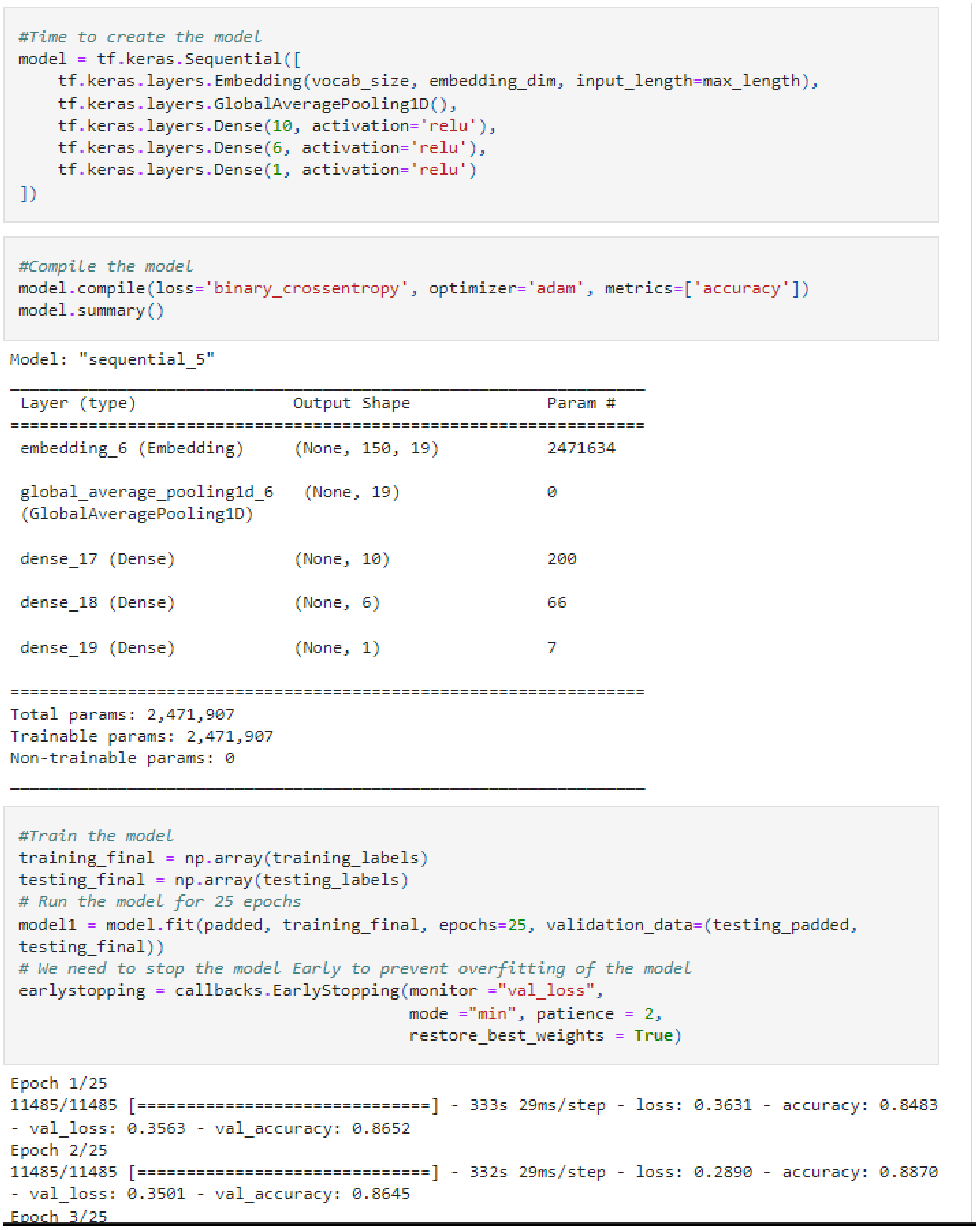
```
#Starting the tokenizer
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
# Fitting the tokenizer
tokenizer.fit_on_texts(training_sentences)
```

```
#Setting the sequences
sequences = tokenizer.texts_to_sequences(training_sentences)
#Setting the padding
padded = pad_sequences(sequences, maxlen=max_length, truncating=trunc_type)
testing_sentences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(testing_sentences, maxlen=max_length)
```

```
# Let’s check out the padded array
padded
print(df)
```

```
        Score                                             Review  Sentiments
0         4.0  the materials arrived early and were in excell...           1
1         4.0  i am really enjoying this book with the worksh...           1
2         1.0  if you are taking this class don t waste your ...           0
3         3.0  this book was missing pages     important pages...           1
4         5.0  i have used learnsmart and can officially say ...           1
...       ...                                                ...         ...
459431    2.0  no instructions      no help unless you want to...           0
459432    1.0                                        it s a joke           0
459433    5.0  i have multiple licenses of the antivirus  i h...           1
459434    5.0                                         good value           1
459435    5.0                       very nice designs easy to use           1

[459370 rows x 3 columns]
```



```
#Time to create the model
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(10, activation='relu'),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='relu')
])
```

```
#Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_6 (Embedding)     (None, 150, 19)           2471634

 global_average_pooling1d_6   (None, 19)               0
 (GlobalAveragePooling1D)

 dense_17 (Dense)            (None, 10)                200

 dense_18 (Dense)            (None, 6)                 66

 dense_19 (Dense)            (None, 1)                 7

=================================================================
Total params: 2,471,907
Trainable params: 2,471,907
Non-trainable params: 0
_________________________________________________________________
```

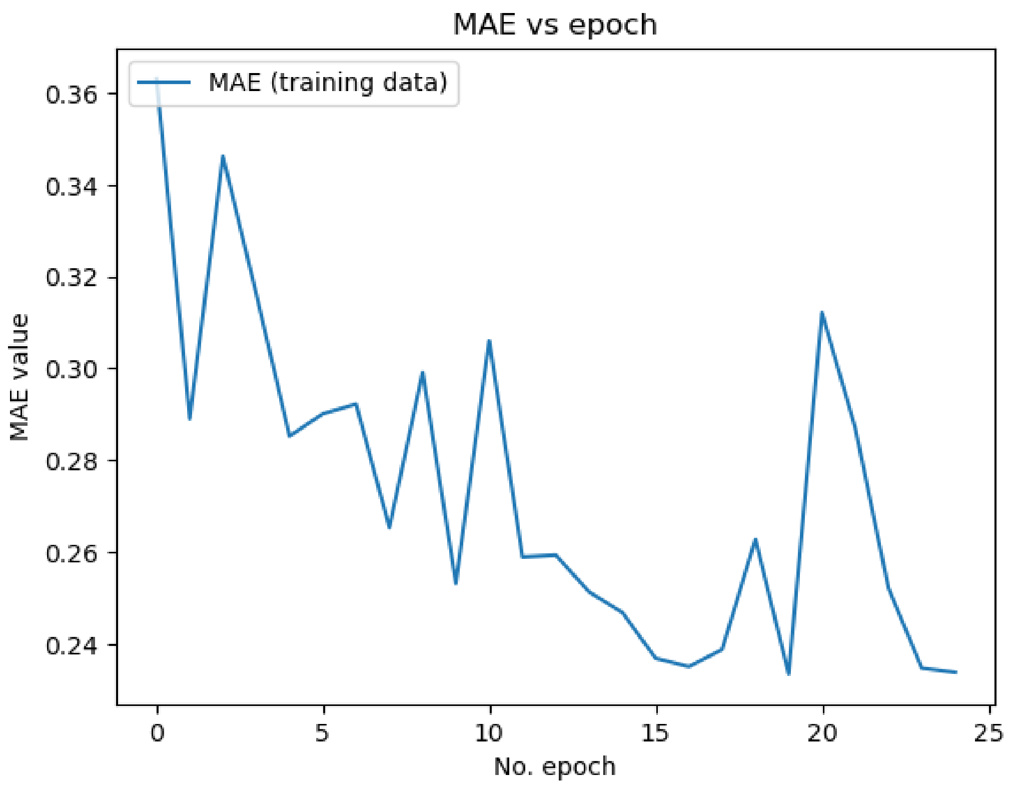
```
#Train the model
training_final = np.array(training_labels)
testing_final = np.array(testing_labels)
# Run the model for 25 epochs
model1 = model.fit(padded, training_final, epochs=25, validation_data=(testing_padded,
testing_final))
# We need to stop the model Early to prevent overfitting of the model
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 2,
                                        restore_best_weights = True)
```

```
Epoch 1/25
11485/11485 [==============================] - 333s 29ms/step - loss: 0.3631 - accuracy: 0.8483
- val_loss: 0.3563 - val_accuracy: 0.8652
Epoch 2/25
11485/11485 [==============================] - 332s 29ms/step - loss: 0.2890 - accuracy: 0.8870
- val_loss: 0.3501 - val_accuracy: 0.8645
Epoch 3/25
```

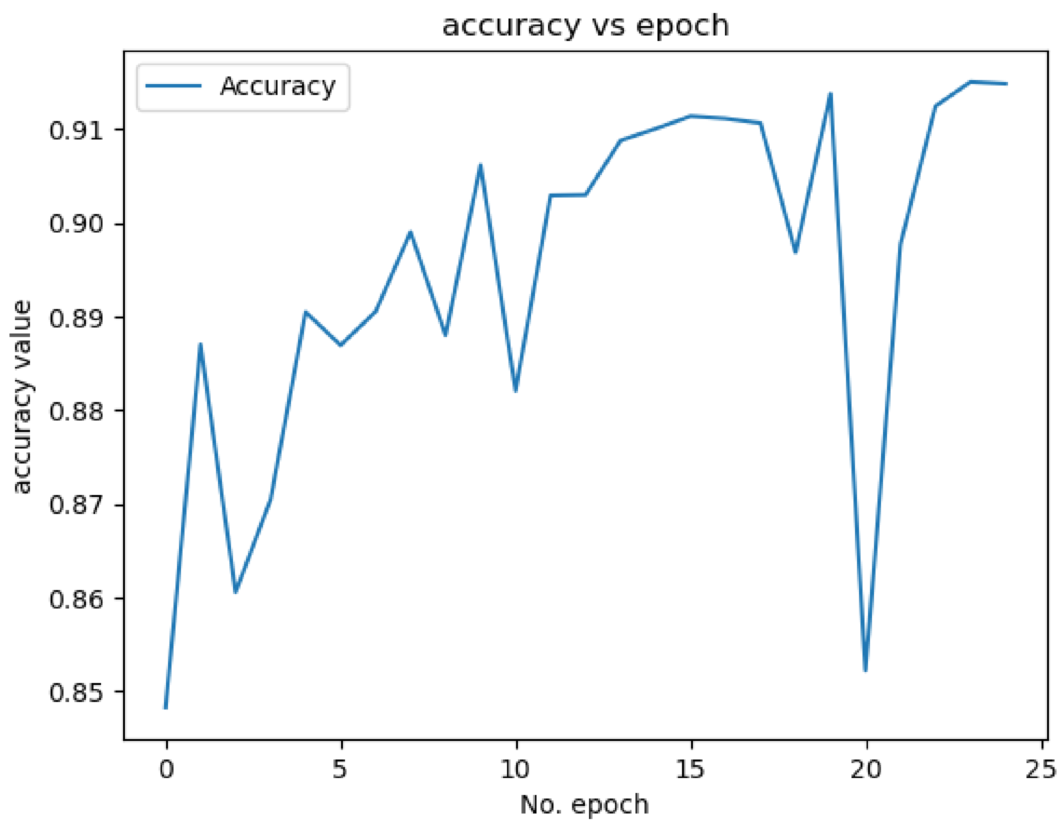
```
# Time to run the model again and see our accuracy
model2 = model.fit(padded, training_final, epochs=25, validation_data=(testing_padded,
                                                                      testing_final),
                                                                      callbacks = [earlystopp
```

```
Epoch 1/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2312 - accuracy: 0.9153
- val_loss: 0.3715 - val_accuracy: 0.8639
Epoch 2/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2261 - accuracy: 0.9165
- val_loss: 0.3883 - val_accuracy: 0.8708
Epoch 3/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2349 - accuracy: 0.9138
- val_loss: 0.3749 - val_accuracy: 0.8652
```

```
# epoch vs Mean Absolute Error (MAE)
mpl.plot(model1.history['loss'], label='MAE (training data)')
mpl.title('MAE vs epoch')
mpl.ylabel('MAE value')
mpl.xlabel('No. epoch')
mpl.legend(loc="upper left")
mpl.show()
```

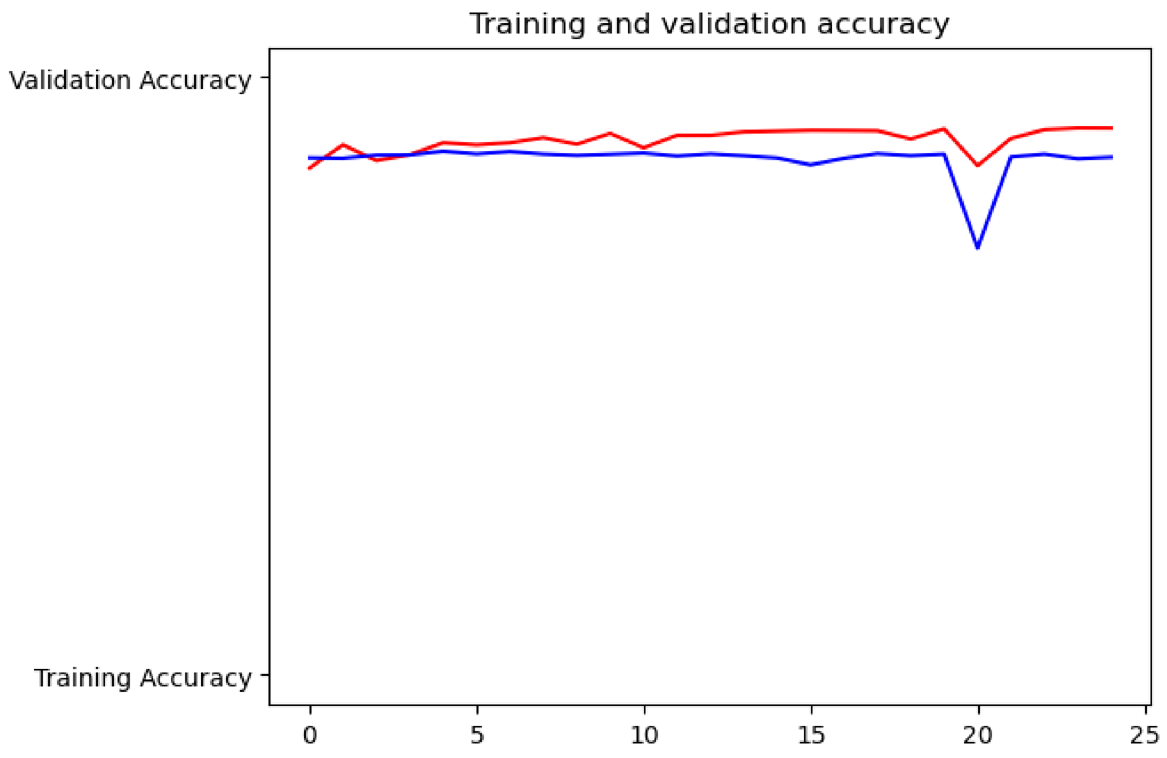


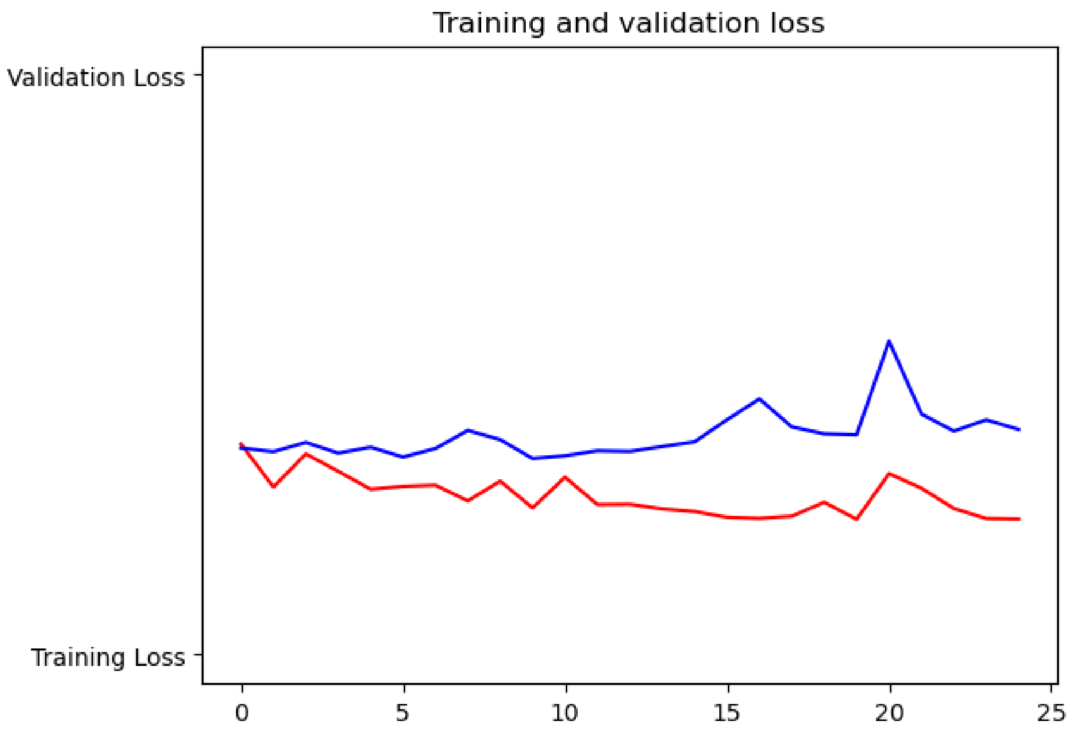
```python
#Check our accuracy per Epoch
mpl.plot(model1.history['accuracy'], label='Accuracy')
mpl.title('accuracy vs epoch')
mpl.ylabel('accuracy value')
mpl.xlabel('No. epoch')
mpl.legend(loc="upper left")
mpl.show()
```



```
#Training versus Accuracy
acc = model1.history['accuracy']
val_acc = model1.history['val_accuracy']
loss = model1.history['loss']
val_loss = model1.history['val_loss']
epochs=range(len(acc))
mpl.plot(epochs, acc, 'r', 'Training Accuracy')
mpl.plot(epochs, val_acc, 'b', 'Validation Accuracy')
mpl.title('Training and validation accuracy')
mpl.figure()
mpl.plot(epochs, loss, 'r', 'Training Loss')
mpl.plot(epochs, val_loss, 'b', 'Validation Loss')
mpl.title('Training and validation loss')
mpl.figure()
```

```
<Figure size 640x480 with 0 Axes>
```





```
<Figure size 640x480 with 0 Axes>
```

```
#Exports
train_reviews.to_csv('D213_Task2_train_reviews.csv', index = False)
train_label.to_csv('D213_Task2_train_label.csv', index = False)
test_reviews.to_csv('D213_Task2_test_reviews.csv', index = False)
test_label.to_csv('D213_Task2_test_label.csv', index = False)
```

```
df.to_csv('D213_Task2_df_Ready_to_Run.csv', index = False)
```

```
#Saving our model
model.save('D213_Task2_Model.h5')
```

b. The packages imported for tokenization are

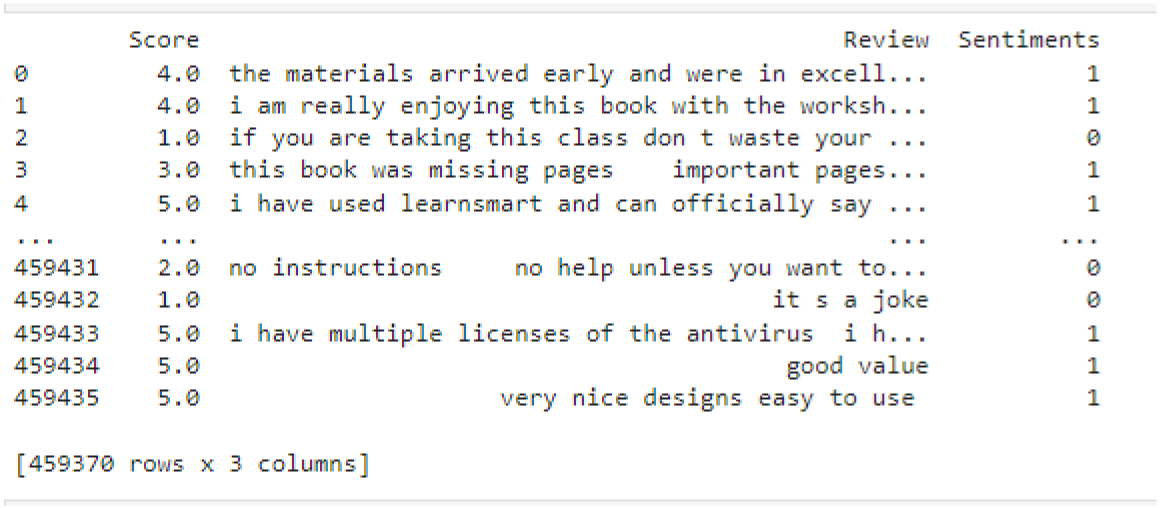
   i. **import** tensorflow **as** tf - Tensorflow will be our main package to handle the modeling
   ii. **from** tensorflow.keras.preprocessing.text **import** Tokenizer - To tokenize our data
   iii. **from** tensorflow.keras.preprocessing.sequence **import** pad_sequences - For padding our inputs that are less than the maximum length
   iv. **from** keras **import** callbacks - To aid in modeling

c. The packages needed to normalize the data and prepare for tokenization
   i. **import** json - To read the json file
   ii. **import** pandas **as** pd - To create a dataframe with our data
   iii. **import** gzip - To unzip our file
   iv. **import** numpy **as** np - Used in basic math calculations
   v. **import** matplotlib.pyplot **as** mpl - For our visualizations
   vi. **import** matplotlib.image **as** mpimg - To save the visualization
   vii. **%matplotlib** inline
   viii. **import** seaborn **as** sb - Further visualizations

3. The padding process used to standardize the length of the sequences improves the performance of our model by making all input sentences the same size. We used the pad_sequences package to pad our data for us.

   a. The padding occurs after the text sequence. In our code this can be seen as ‘post.’

   b. A screenshot of the single padded sequence can be seen with the review “good value” which is less than the maximum length and is shown in the padded array.



```
        Score                                             Review  Sentiments
0         4.0  the materials arrived early and were in excell...           1
1         4.0  i am really enjoying this book with the worksh...           1
2         1.0  if you are taking this class don t waste your ...           0
3         3.0  this book was missing pages    important pages...           1
4         5.0  i have used learnsmart and can officially say ...           1
...       ...                                                ...         ...
459431    2.0  no instructions     no help unless you want to...           0
459432    1.0                                       it s a joke           0
459433    5.0  i have multiple licenses of the antivirus  i h...           1
459434    5.0                                         good value           1
459435    5.0                       very nice designs easy to use           1

[459370 rows x 3 columns]
```

4. There are two categories of sentiment that will be used and the activation function for the final dense layer of the network, 0 = negative, 1 = positive. We have further broken them down by changing numeric review answers to positive and negative sentiments. review 0-2 negative, 3-5 positive. An appropriate fitting activation function for the final dense layer of the network is relu activation because relu runs a piecewise linear activation for binary sentiments, outputting 1 for positive and 0 for everything else (negative) (*ReLU (Rectified Linear Unit) Activation Function* 2021*)*.

5. The steps used to prepare the data for analysis is as follows:
    a. Load packages needed
    b. Parse for json gzip file downloaded from the links provided
    c. Turn json into df
    d. Import the data as a date frame
    e. View the data for EDA
    f. Create the new data frame with the new selected columns we need
    g. Rename the columns
    h. Check the columns for nulls
    i. Drop the nulls
    j. Verify nulls are mitigated

k. Create sentiments column from the dataset (Changing numeric review answers to positive and negative sentiments. review 0-2 negative, 3-5 positive. 0 = negative, 1 = positive
l. Select the special characters and replace them with spaces
m. Replace capitals with lowercase letters
n. Export the prepared data
o. Split into training/test. We are using an 80% split model where 80% of the data is used for training and 20% for testing/validation.
p. Set maximum word lengths, Tokenize the data, pad it
q. Create and compile the model
r. Validate the model with MSE and select for the number of epochs from accuracy

6. The final prepared dataset has been included as “'D213_Task2_df_Ready_to_Run.csv”

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.

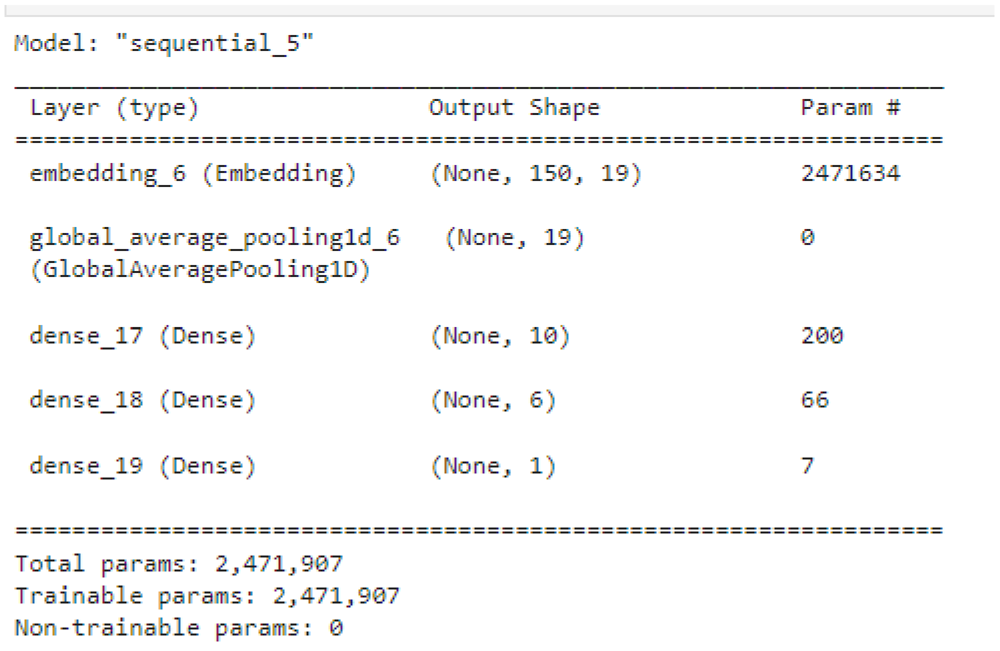
2. Discuss the number of layers, the type of layers, and total number of parameters.

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

- **activation functions**
- **number of nodes per layer**
- **loss function**
- **optimizer**

- **stopping criteria**
- **evaluation metric**

1. The output of the model summary from TensorFlow is:



```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_6 (Embedding)     (None, 150, 19)           2471634

 global_average_pooling1d_6   (None, 19)               0
 (GlobalAveragePooling1D)

 dense_17 (Dense)            (None, 10)                200

 dense_18 (Dense)            (None, 6)                 66

 dense_19 (Dense)            (None, 1)                 7

=================================================================
Total params: 2,471,907
Trainable params: 2,471,907
Non-trainable params: 0
```

2. The summary information of the model is as follows:
   a. The number of layers: 5
   b. The type of layers include embedding, globalaveragepooling1d, and 3 dense layers.
   c. The total number of parameters is 2,471,907, all of which are trainable.

3. The choice of hyperparameters is summarized below:
   a. Activation function: I used the Rectified Linear Activation (ReLu) function because it is very simple to implement and has the ability to deal with hidden layers. This activation function is also very efficient when applied to binary sentiments, which is what we are looking for.

b. Number of Nodes per layer: We are trying to predict a 0 or 1, so we have gone with 2 nodes per layer based on the problem we are handling.

c. Loss Function: This model uses binary cross-entropy because we have 2 sentiments (positive and negative). Binary cross-entropy measures the difference between the predicted probability and true label in this binary classification.

d. Optimizer: The optimizer used in this model is the adam optimizer. The adam algorithm updates network weights iteratively based on our training data. It is an extension of the stochastic gradient descent. This optimizer is very easy to implement and intuitive to use, while reducing overfitting, making it the ideal choice of optimizer for this model.

e. Stopping Criteria: Early stopping is used here to prevent the neural network from overfitting the training data. Early stopping therefore improves the generalization capabilities of the final model by monitoring the accuracy score and stops training when the validation score stops improving. The patience is how many epochs the model can go without improving and is determined by early stopping (Brownlee 2018).

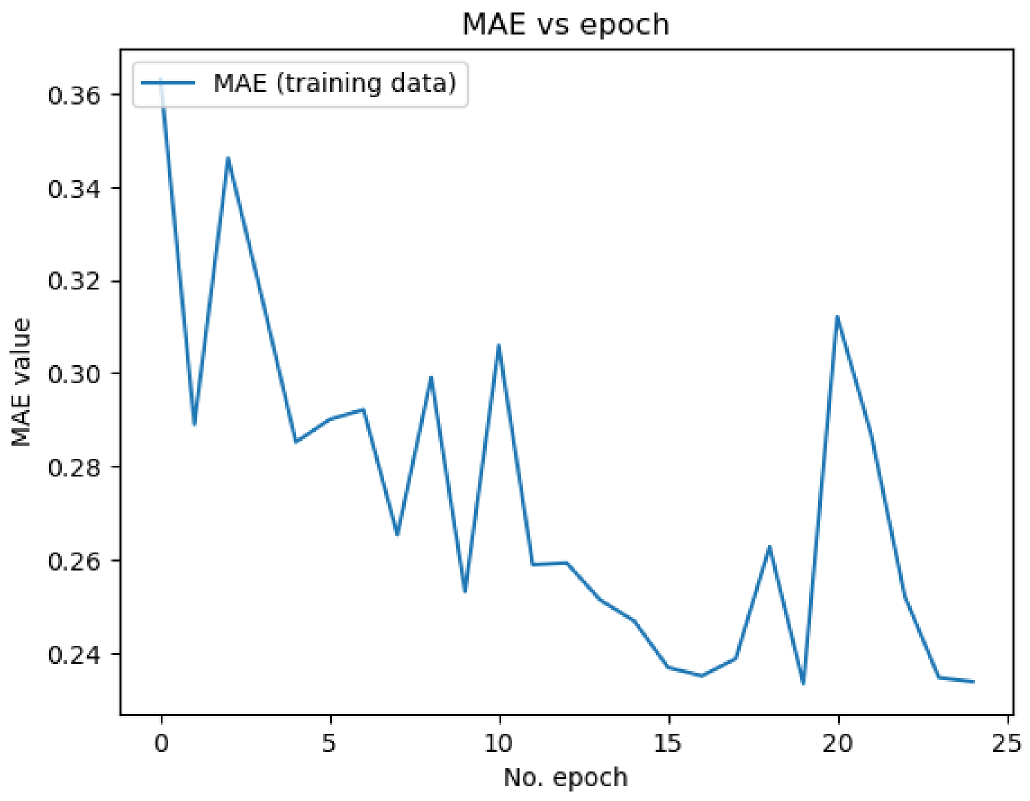
f. Evaluation Metric: Accuracy is used to evaluate our metric. The accuracy represents how our model performs and can be broken down and seen below with the epoch accuracy and visualizations. Our final model is model2:

```
11485/11485 [==============================] - 339s 30ms/step - loss: 0.2347 - accuracy: 0.9150
- val_loss: 0.4049 - val_accuracy: 0.8637
Epoch 25/25
11485/11485 [==============================] - 341s 30ms/step - loss: 0.2338 - accuracy: 0.9148
- val_loss: 0.3883 - val_accuracy: 0.8663
```

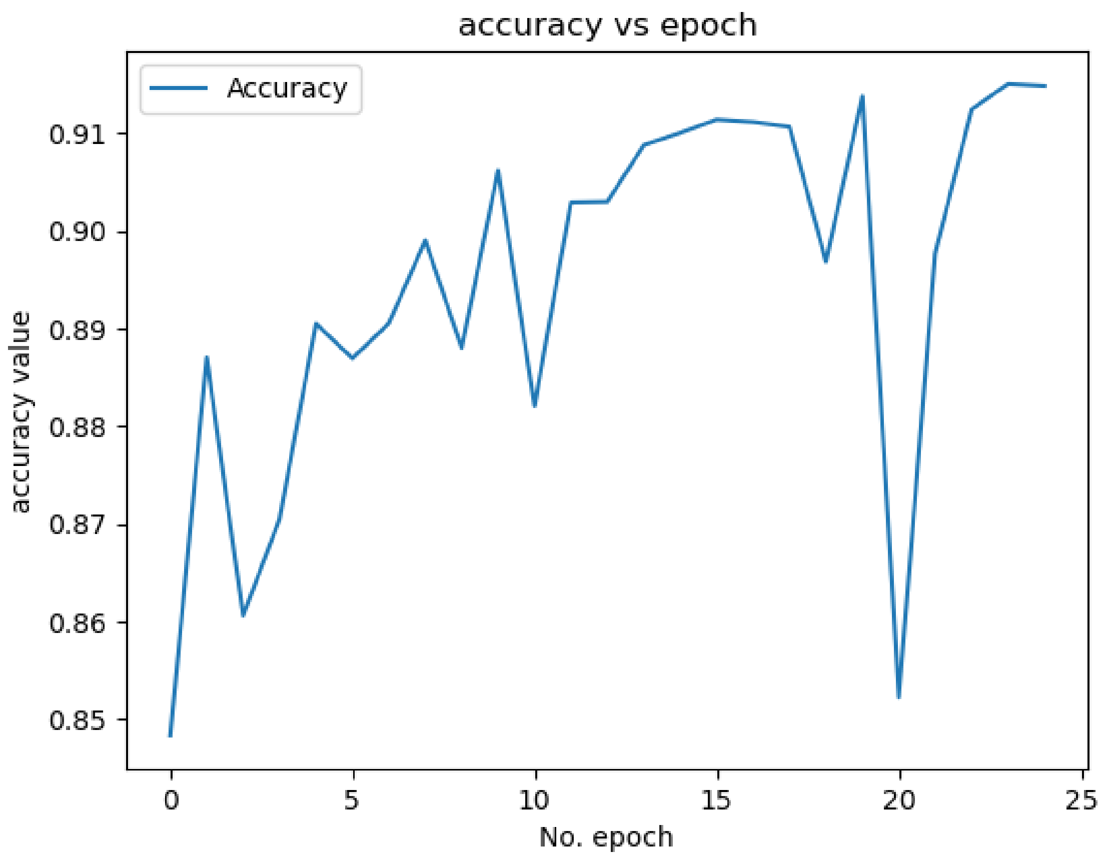
```
# Time to run the model again and see our accuracy
model2 = model.fit(padded, training_final, epochs=25, validation_data=(testing_padded,
                                                                        testing_final),
                                                                        callbacks = [earlystopp
```

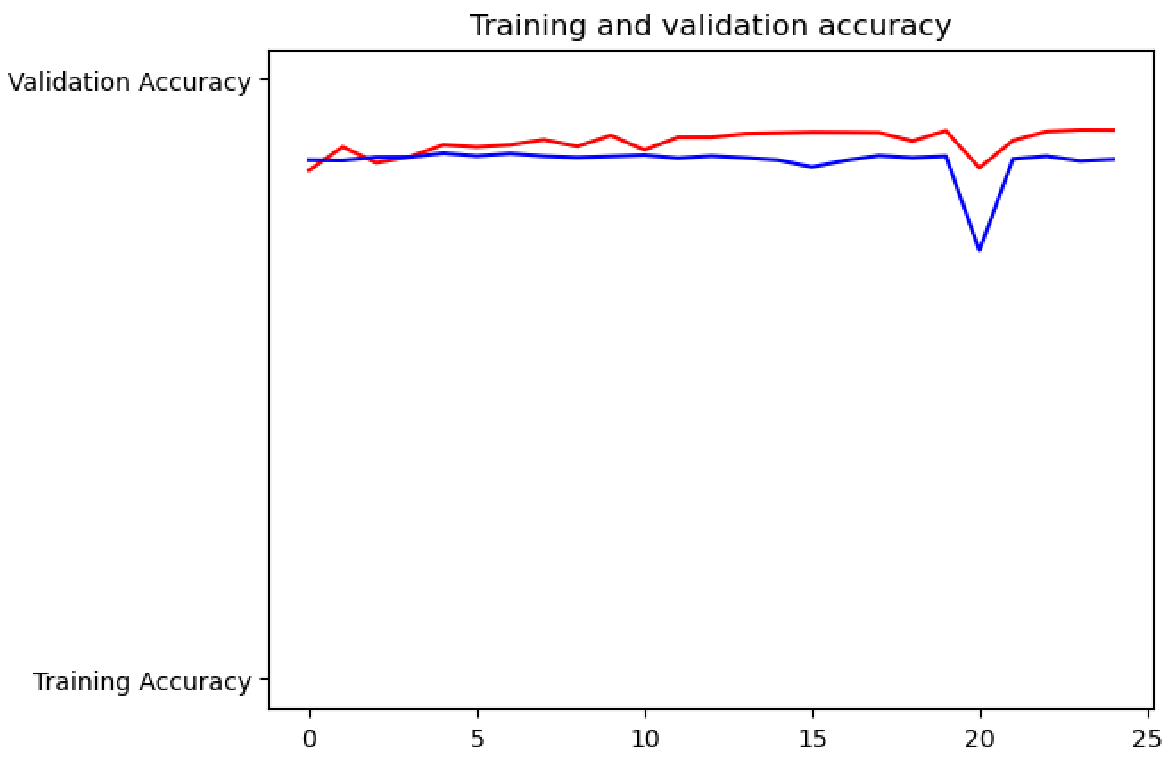
```
Epoch 1/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2312 - accuracy: 0.9153
- val_loss: 0.3715 - val_accuracy: 0.8639
Epoch 2/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2261 - accuracy: 0.9165
- val_loss: 0.3883 - val_accuracy: 0.8708
Epoch 3/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2349 - accuracy: 0.9138
- val_loss: 0.3749 - val_accuracy: 0.8652
```

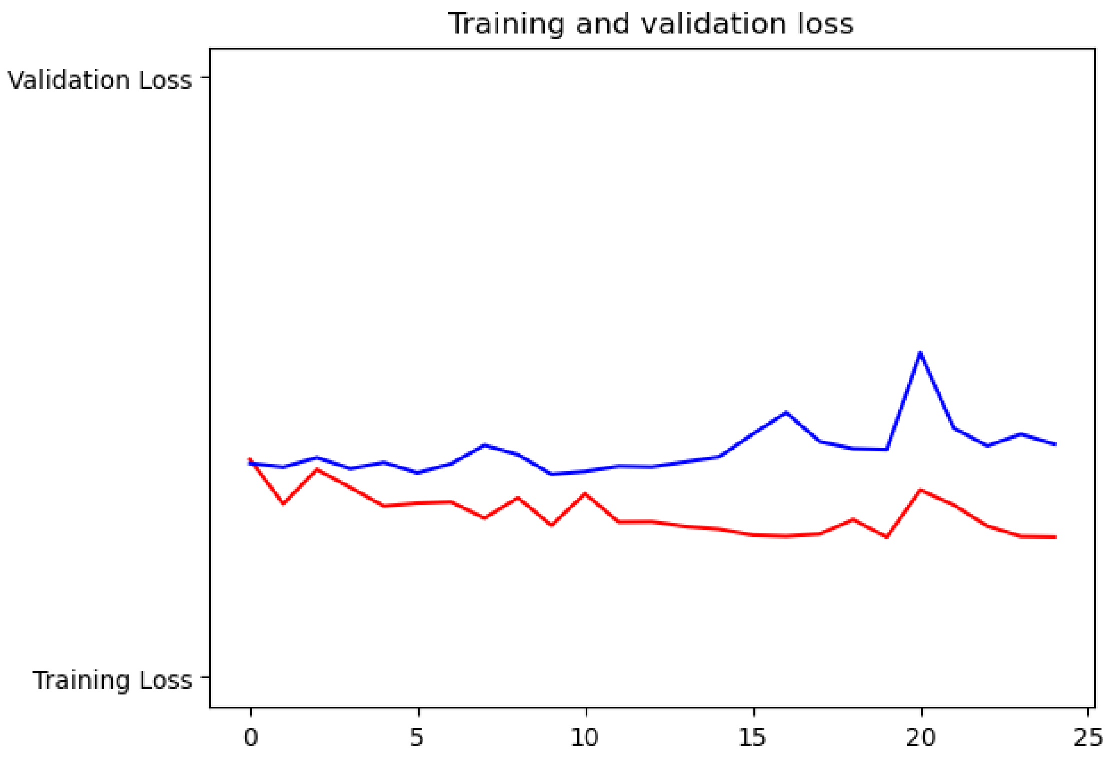
```
# epoch vs Mean Absolute Error (MAE)
mpl.plot(model1.history['loss'], label='MAE (training data)')
mpl.title('MAE vs epoch')
mpl.ylabel('MAE value')
mpl.xlabel('No. epoch')
mpl.legend(loc="upper left")
mpl.show()
```



```
#Check our accuracy per Epoch
mpl.plot(model1.history['accuracy'], label='Accuracy')
mpl.title('accuracy vs epoch')
mpl.ylabel('accuracy value')
mpl.xlabel('No. epoch')
mpl.legend(loc="upper left")
mpl.show()
```




Training and validation accuracy
Validation Accuracy
Training Accuracy
0
5
10
15
20
25


Training and validation loss
Validation Loss
Training Loss
0
5
10
15
20
25

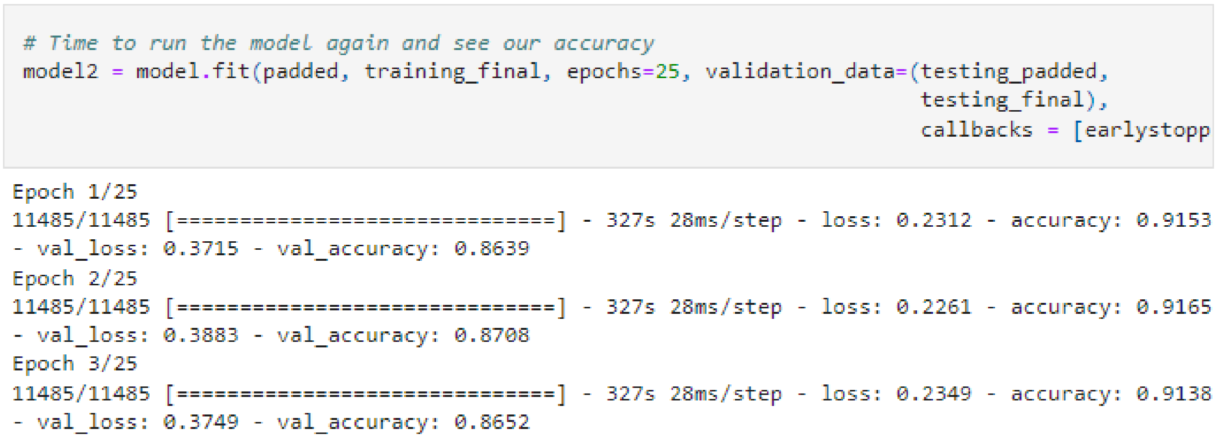
Part IV: Model Evaluation

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

> **1. Discuss the impact of using stopping criteria instead of defining the number of epochs, including a screenshot showing the final training epoch.**
>
> **2. Provide visualizations of the model's training process, including a line graph of the loss and chosen evaluation metric.**
>
> **3. Assess the fitness of the model and any measures taken to address overfitting.**
>
> **4. Discuss the predictive accuracy of the trained network.**

1. The impact of using stopping criteria instead of defining the number of epochs is that when you do it by the number of epochs, it sets a hard stop at that arbitrarily chosen number. This requires manual verification of the training and validation scores. In contrast, using early stopping stops the training as soon as the training score stops improving, taking the verification process out of our hands and into a more statistically sound method. Early stopping uses patience, the number of epochs the model runs through without improving, to decide when to stop training. Our model stops at 3 epochs

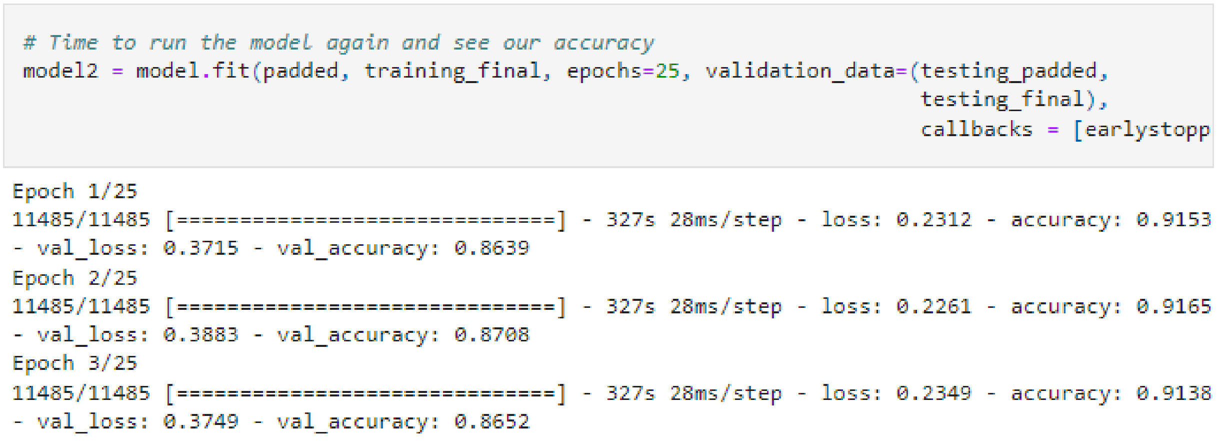
in the final training to prevent overfitting. A screenshot of the final training epoch is as follows:



```
# Time to run the model again and see our accuracy
model2 = model.fit(padded, training_final, epochs=25, validation_data=(testing_padded,
                                                                       testing_final),
                                                                       callbacks = [earlystopp
```

```
Epoch 1/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2312 - accuracy: 0.9153
- val_loss: 0.3715 - val_accuracy: 0.8639
Epoch 2/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2261 - accuracy: 0.9165
- val_loss: 0.3883 - val_accuracy: 0.8708
Epoch 3/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2349 - accuracy: 0.9138
- val_loss: 0.3749 - val_accuracy: 0.8652
```

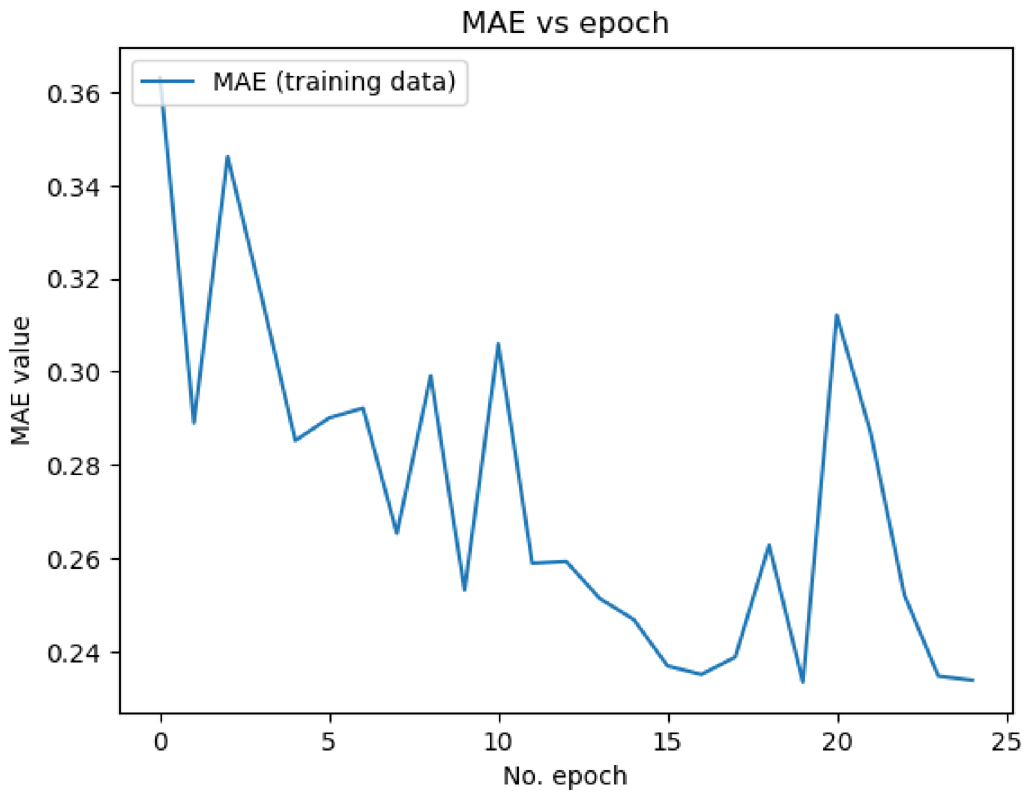
2. The visualizations below represent the training process. At the end is a line graph of the loss and chosen evaluation metrics. Mean Absolute Error (MAE) is the measure of error between the paired observations and used as our error metric in our first visualization. We then use the accuracy measurements from the model seen as val_accuracy.



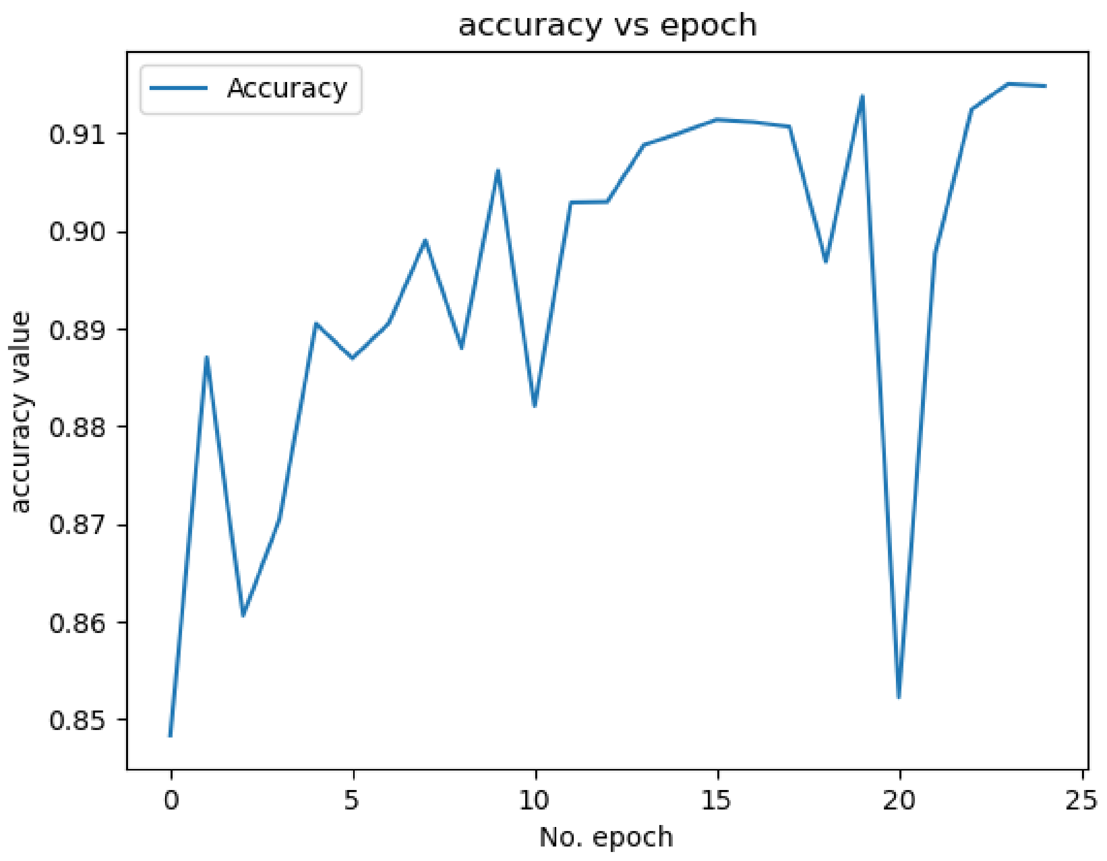
```
# Time to run the model again and see our accuracy
model2 = model.fit(padded, training_final, epochs=25, validation_data=(testing_padded,
                                                                       testing_final),
                                                                       callbacks = [earlystopp
```

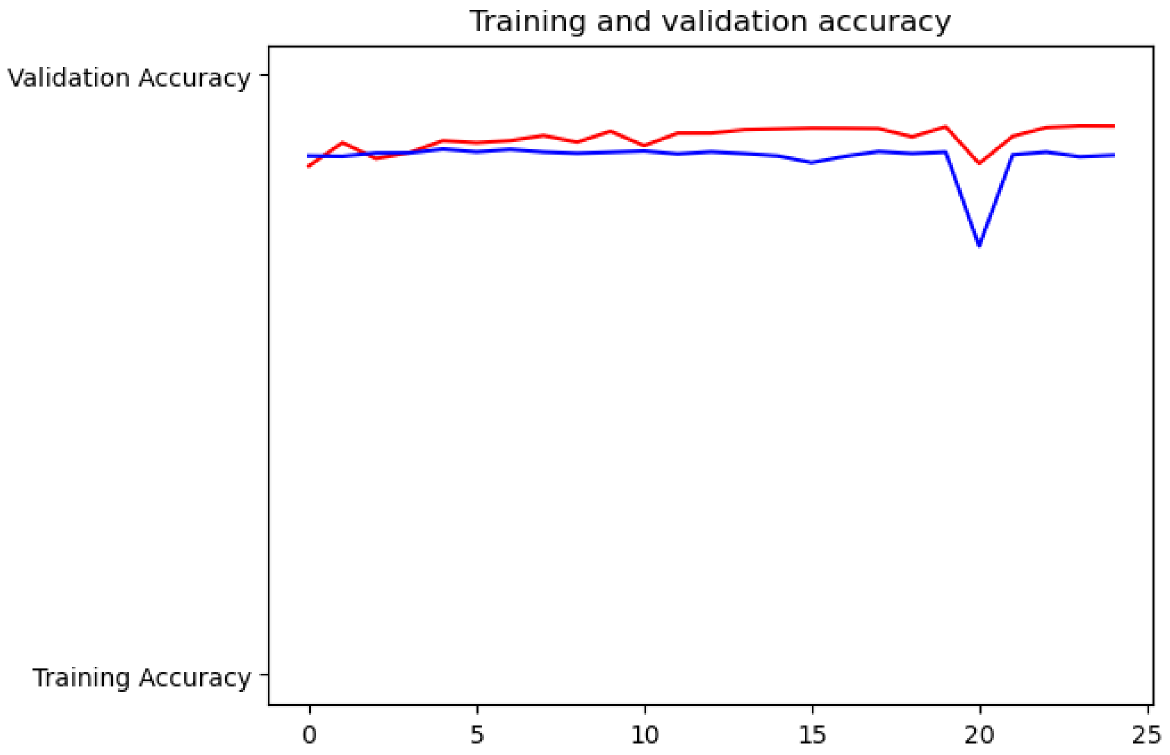
```
Epoch 1/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2312 - accuracy: 0.9153
- val_loss: 0.3715 - val_accuracy: 0.8639
Epoch 2/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2261 - accuracy: 0.9165
- val_loss: 0.3883 - val_accuracy: 0.8708
Epoch 3/25
11485/11485 [==============================] - 327s 28ms/step - loss: 0.2349 - accuracy: 0.9138
- val_loss: 0.3749 - val_accuracy: 0.8652
```

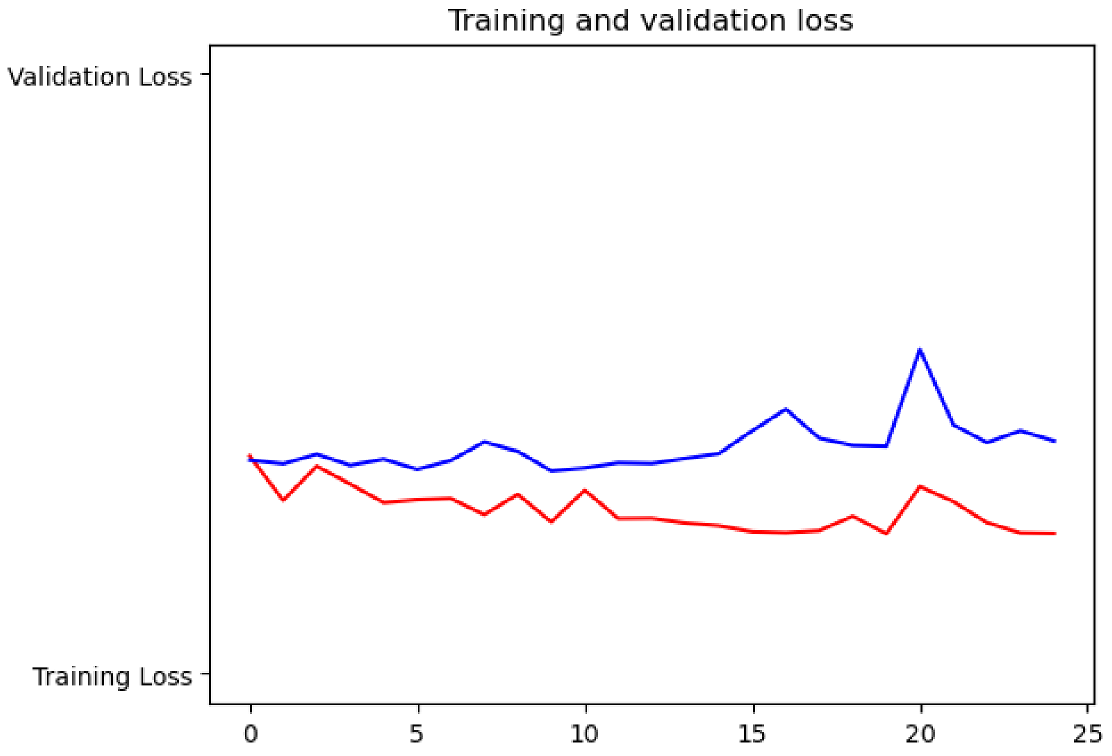
```
# epoch vs Mean Absolute Error (MAE)
mpl.plot(model1.history['loss'], label='MAE (training data)')
mpl.title('MAE vs epoch')
mpl.ylabel('MAE value')
mpl.xlabel('No. epoch')
mpl.legend(loc="upper left")
mpl.show()
```



```
#Check our accuracy per Epoch
mpl.plot(model1.history['accuracy'], label='Accuracy')
mpl.title('accuracy vs epoch')
mpl.ylabel('accuracy value')
mpl.xlabel('No. epoch')
mpl.legend(loc="upper left")
mpl.show()
```




Training and validation accuracy
Validation Accuracy
Training Accuracy
0
5
10
15
20
25


Training and validation loss
Validation Loss
Training Loss
0
5
10
15
20
25

3. The fitness of the model is represented in the line graphs and shows how well our model fits, or how accurate it is. Overfitting is when the model cannot accurately represent the underlying shape of the data. Overfitting has been addressed by using the adam optimizer, which reduces overfitting through several methods listed earlier in this report. Early stopping is also deployed here to reduce overfitting. Both of these techniques are explained in detail in section C3d and e. Our model stops at 3 epochs to prevent overfitting.

4. The predictive accuracy of the trained network is shown through the accuracy value and the line graphs comparing the accuracy. The prediction loss is 0.2349 and the value_loss is 0.3749. The accuracy stops at .9138 and the val_accuracy is .8652. From these results we can say we have 86.5% accuracy in predicting customer sentiment based on the words in the customer review. From the line graphs we can see there is a large dropoff of accuracy after 3, indicating that is an ideal stopping point. Finally from the line graphs showing training and validation accuracy/loss we can see our model follows the trend closely and is within an acceptable accuracy.

Part V: Summary and Recommendations

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

The code used to save the trained network within the neural network has been saved as "D213_Task2_Code.ipynb"

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

The neural network's functionality can be described as being trained and able to analyze customer feedback and identify sentiments in the review texts submitted by customers. The neural network has a high predictive accuracy and the impact is that the sentiment analysis can be used by the organization to figure out which products customers liked based on positive and negative reviews. NLP was used to break up and analyze customer sentiments from the reviews. It was trained with actual reviews. We used 80% of the data file for training, which comes to 367,496 reviews. This model has been tuned to perform predictions on new customer reviews to identify the sentiment as negative or positive with an 86.5% accuracy after 3 epochs.

We used tensorflow to create the keras model and the embedding layer was chosen from the parameters we found. The layers of the model can be found in the model summary section in this report, but total to be 5. Finally, the model used ReLu activation. The result of this network architecture is a model that can accurately answer our research question and identify customer sentiments from the wording of reviews. By using this analysis, the organization will be able to appropriately respond to any concerns in the reviews and have a fast way to analyze the sentiment of customer reviews.

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

A course of action based on the results here I recommend is to take this trained model and use it to analyze and predict customer sentiments from reviews. I would use this model to predict how customers feel about the product to better inform how to respond to customer reviews, how to alter their product or business plan, and how they digest customer reviews. The organization can do this by gathering the customer reviews and running this model with them.

Part VI: Reporting

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

The notebook presentation has been attached as "D213_Task1_Code" as a html and "D213_Task2_Code_pdfBackup" as a pdf, able to be viewed according to your preference and what your machine is best suited for.

I. List the web sources used to acquire data or segments of third-party code to support the application.

References

Brownlee, J. (2018, December 9). *Use Early Stopping to Halt the Training of Neural Networks At the Right Time*. Machine Learning Mastery. https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

Justifying recommendations using distantly-labeled reviews and fined-grained aspects

Jianmo Ni, Jiacheng Li, Julian McAuley

Empirical Methods in Natural Language Processing (EMNLP), 2019.

Pai, A. (2020, May 25). *What is Tokenization | Tokenization In NLP*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

J. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

References

Pai, A. (2020, May 25). *What is Tokenization | Tokenization In NLP*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

ReLU (Rectified Linear Unit) Activation Function. (2021, December 30). OpenGenus IQ: Computing Expertise & Legacy. https://iq.opengenus.org/relu-activation/

K. Demonstrate professional communication in the content and presentation of your submission.

This aspect of the rubric is evaluated through the entirety of this report and I hope professionalism has shown continuously.