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.**
   Using existing reviews, can we identify the sentiment of unseen reviews as positive or negative?

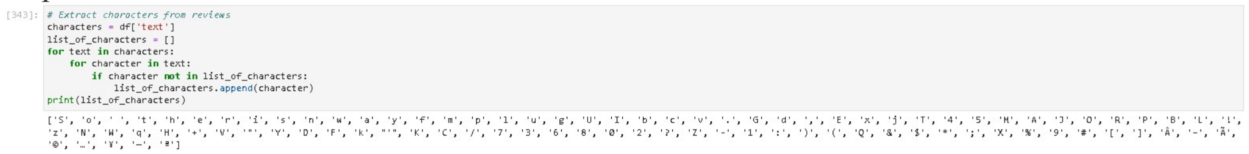
2. **Define the objectives or goals of the data analysis. Be sure the objectives or goals are reasonable within the scope of the research question and are represented in the available data.**
   The main goal is to train a neural network to recognize if a review of a product or service is positive or negative based on the words in the text of the review.

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.**
   I will be using a Recurrent Neural Network (RNN) to perform the sentiment analysis. RNNs depend on not just the input information, but the sequence of the inputs, to train the model. (Smartboost, 2020)

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.)**
     There are several unusual characters, including accented letters and punctuation marks. They are detailed in the screenshot below and will be removed in the cleaning process.



```
[343]: # Extract characters from reviews
characters = df['text']
list_of_characters = []
for text in characters:
    for character in text:
        if character not in list_of_characters:
            list_of_characters.append(character)
print(list_of_characters)
```

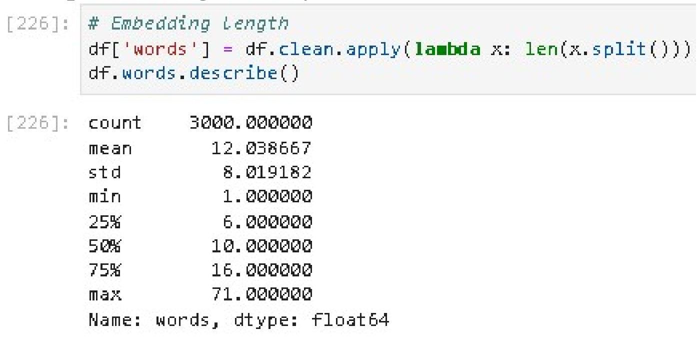
```
['S', 'o', ' ', 't', 'h', 'e', 'r', 'i', 's', 'n', 'w', 'a', 'y', 'f', 'm', 'p', 'l', 'u', 'g', 'U', 'I', 'b', 'c', 'v', '.', 'G', 'd', ',', 'E', 'x', 'j', 'T', '4', '5', 'M', 'A', 'J', 'O', 'R', 'P', 'B', 'L', '!',
'z', 'N', 'W', 'q', 'H', '+', 'V', '"', 'Y', 'D', 'F', 'k', "'", 'K', 'C', '/', '7', '3', '6', '8', '0', '2', '?', 'Z', '-', '1', ':', ')', '(', 'Q', '&', '$', '*', ';', 'X', '%', '9', '#', '[', ']', 'Â', '–', 'Ã',
'©', '…', '¥', '—', 'ª']
```

   - **vocabulary size**
     The vocabulary size is 5278 unique words in the reviews.

   - **proposed word embedding length**
     I propose a 9 word embedding length. One rule of thumb used to determine embedding length us using the fourth root of the total number of categories. (Malin, 2021) The vocabulary size is 5278 words, and the fourth root of 5278 is approximately 8.5, which I have rounded to 9.

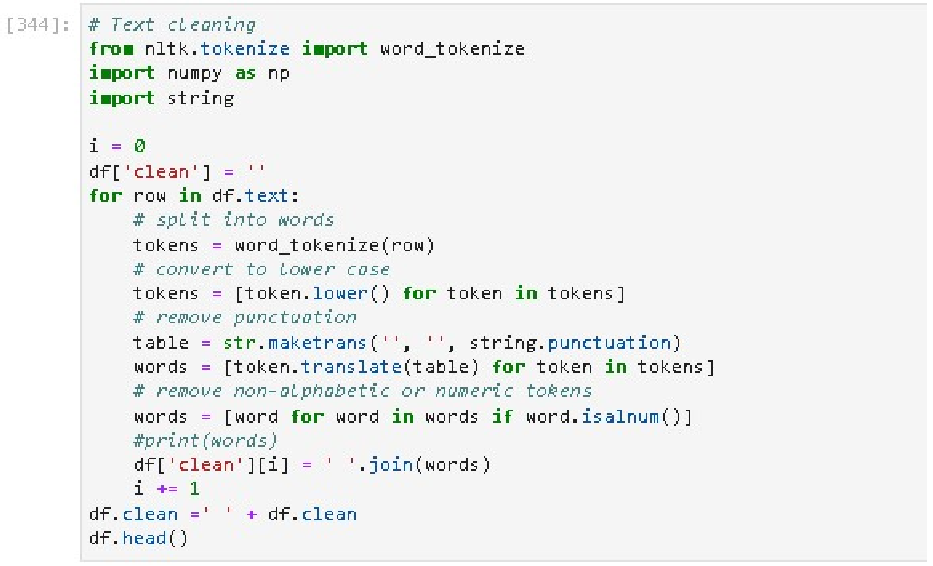
   - **statistical justification for the chosen maximum sequence length**
     The maximum sequence length is chosen so that none of the input data is cut short. We address any sequences that are naturally shorter than the maximum length by

padding them with white space to bring them to the maximum length. The maximum sequence length in my dataset is 71.

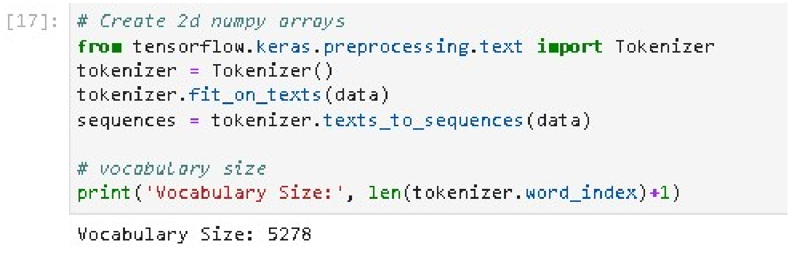


```
[226]: # Embedding length
       df['words'] = df.clean.apply(lambda x: len(x.split()))
       df.words.describe()

[226]: count    3000.000000
       mean       12.038667
       std         8.019182
       min         1.000000
       25%         6.000000
       50%        10.000000
       75%        16.000000
       max        71.000000
       Name: words, dtype: float64
```

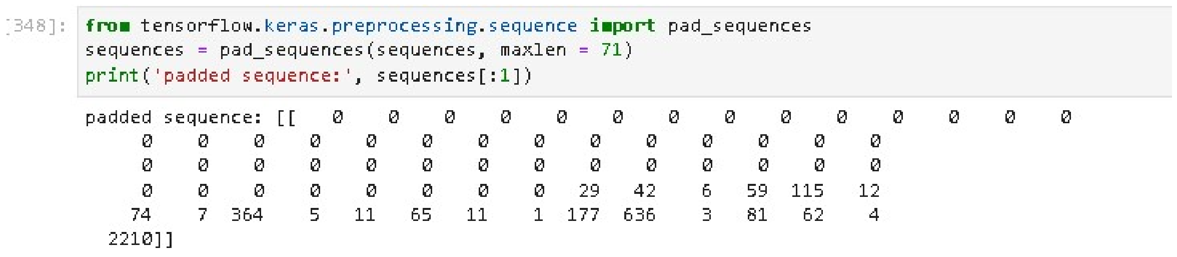
2. **Describe the goals of the tokenization process, including any code generated and packages that are used to normalize text during the tokenization process.**
   My goals in the tokenization process are to change the text to all lowercase, remove unusual characters and punctuation, split the reviews into words, change the text into sequences, and pad the sequences to make them all the maximum sequence length. Below are screenshots of the different packages and code segments in which I've used tokenization to achieve these goals:



```
[344]: # Text cleaning
       from nltk.tokenize import word_tokenize
       import numpy as np
       import string

       i = 0
       df['clean'] = ''
       for row in df.text:
           # split into words
           tokens = word_tokenize(row)
           # convert to lower case
           tokens = [token.lower() for token in tokens]
           # remove punctuation
           table = str.maketrans('', '', string.punctuation)
           words = [token.translate(table) for token in tokens]
           # remove non-alphabetic or numeric tokens
           words = [word for word in words if word.isalnum()]
           #print(words)
           df['clean'][i] = ' '.join(words)
           i += 1
       df.clean =' ' + df.clean
       df.head()
```



```
[17]: # Create 2d numpy arrays
      from tensorflow.keras.preprocessing.text import Tokenizer
      tokenizer = Tokenizer()
      tokenizer.fit_on_texts(data)
      sequences = tokenizer.texts_to_sequences(data)

      # vocabulary size
      print('Vocabulary Size:', len(tokenizer.word_index)+1)

      Vocabulary Size: 5278
```

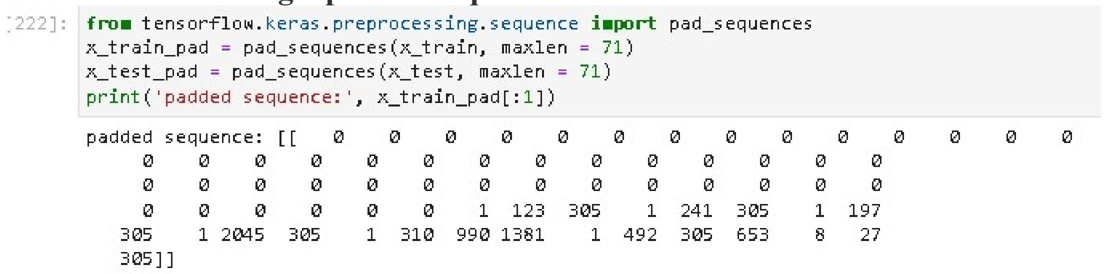


```
[348]: from tensorflow.keras.preprocessing.sequence import pad_sequences
       sequences = pad_sequences(sequences, maxlen = 71)
       print('padded sequence:', sequences[:1])

       padded sequence: [[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
            0    0    0    0    0    0    0    0    0    0    0    0    0    0
            0    0    0    0    0    0    0    0    0    0    0    0    0    0
            0    0    0    0    0    0    0    0   29   42    6   59  115   12
           74    7  364    5   11   65   11    1  177  636    3   81   62    4
         2210]]
```

3. **Explain the padding process used to standardize the length of sequences, including the following in your explanation:**
   Neural Networks require that all the input sequences are the same shape and size. Because naturally the sequences all contain a different number of words, padding is a technique used to standardize the inputs to one size for all. It inserts white space before or after a sequence less than the max length to bring it equal to the max length.

   - **if the padding occurs before or after the text sequence**
     The padding occurs before the text sequence, it is pre-padded.

   - **a screenshot of a single padded sequence**



```
[222]: from tensorflow.keras.preprocessing.sequence import pad_sequences
       x_train_pad = pad_sequences(x_train, maxlen = 71)
       x_test_pad = pad_sequences(x_test, maxlen = 71)
       print('padded sequence:', x_train_pad[:1])

       padded sequence: [[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
            0    0    0    0    0    0    0    0    0    0    0    0    0    0
            0    0    0    0    0    0    0    0    0    0    0    0    0    0
            0    0    0    0    0    0    1  123  305    1  241  305    1  197
          305    1 2045  305    1  310  990 1381    1  492  305  653    8   27
          305]]
```

4. **Identify how many categories of sentiment will be used and an activation function for the final dense layer of the network.**
   There will be two categories of sentiment in my model, 0 for a negative review and 1 for a positive review. The activation function in the final dense layer will be sigmoid because I am predicting a binary output. (Virahonda, 2020)

5. **Explain the steps used to prepare the data for analysis, including the size of the training, validation, and test set split.**
   1. Read the three datasets and combine them into one dataset.
   2. Check for unusual characters**.**
   3. Clean the data: split the reviews into words, remove unusual characters and punctuation, make all the text lowercase, and split the pandas series of the text into a list.
   4. Explore the data: determine the sequence length statistics and vocabulary size.
   5. Use Tokenizer to convert the text into sequences, and then pre-pad the sequences to the maximum sequence length..
   6. Create an 80/20 test and train split.

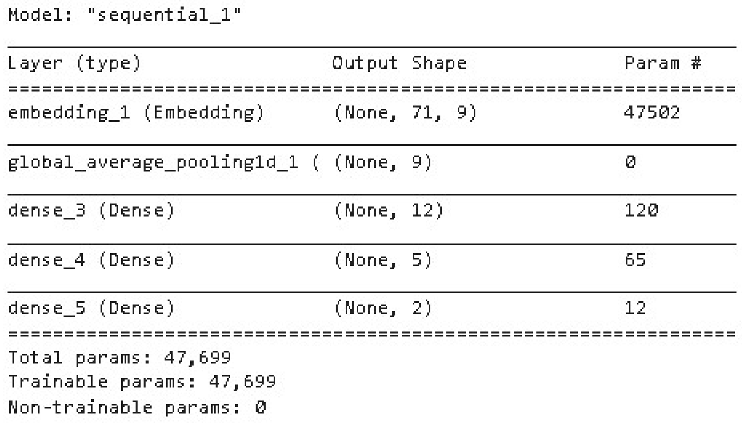
6. **Provide a copy of the prepared dataset.**
   The prepared data is provided in the following documents: Keim D213 Task Two x train,

Keim D213 Task Two x test, Keim D213 Task Two y train, and Keim D213 Task Two y test.

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.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 71, 9)             47502
_________________________________________________________________
global_average_pooling1d_1 ( (None, 9)                 0
_________________________________________________________________
dense_3 (Dense)              (None, 12)                120
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 65
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 12
=================================================================
Total params: 47,699
Trainable params: 47,699
Non-trainable params: 0
```

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

Layer one: This is an Embedding layer. It is the input layer and has 47502 parameters.
Layer two: This is a pooling layer. It is a hidden layer with 0 parameters.
Layer three: A Dense hidden layer with 120 parameters.
Layer four: A Dense hidden layer with 65 parameters.
Layer five: A Dense output layer with 12 parameters.

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

- **activation functions**
  I chose a relu activation function for the hidden layers because it is the default activation in multi-layer neural networks. (Brownlee, A Gentle Introduction to the Rectified Linear Unit (ReLU), 2020)
  I chose a sigmoid activation function for the output layer, because it is good for predicting the probability in binary class outputs. (Virahonda, 2020)
- **number of nodes per layer**
  For my Embedding layer, I chose 71 nodes because that is my maximum sequence length. (Brownlee, How to Use Word Embedding Layers for Deep Learning with Keras, 2021) For the hidden dense layers I chose 12 and 5 respectively because one approximation of the hidden layer nodes is the square root of the product of the input layer nodes times the output layer nodes. (Sachdev, 2020) For the output layer nodes, I chose 2 because I want a binary output.
- **loss function**
  I chose the binary_crossentropy loss function because I am performing a binary classification.
- **optimizer**
  I chose the adam optimizer because it is common and easy to implement. (Virahonda, 2020)
- **stopping criteria**

I used EarlyStopping with patience set to 2 so that the model would stop after two epochs with no improvement. This helps to prevent the model from overfitting. After some trial and error, I also defined the number of epochs to be 50 instead of the keras default of 10.

- **evaluation metric**

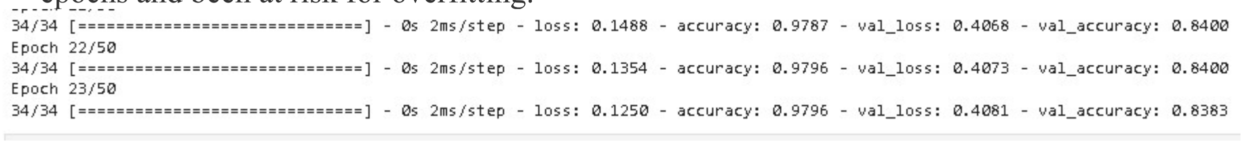
  I used 'accuracy' as my metric to evaluate my model for performance on the training set and overfitting on the test set. The training and validation accuracies of my final epoch were 98.0% and 83.8% respectively.

Part IV: Model Evaluation

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

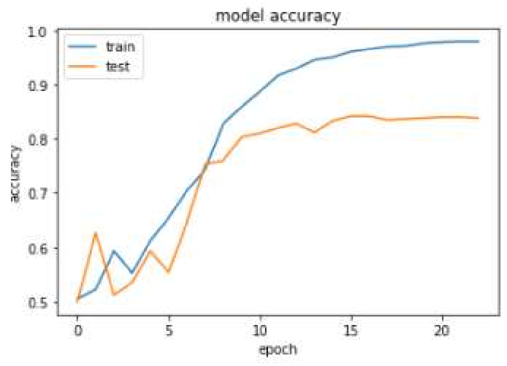
1. **Discuss the impact of using stopping criteria instead of defining the number of epochs, including a screenshot showing the final training epoch.**

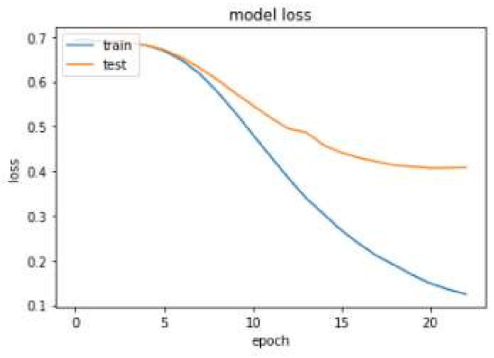
   Using a combination of EarlyStopping and epoch definition let my model train to its full potential without overfitting. It trained to 23/50 epochs. If I hadn't defined the number of epochs at greater than 23, my model wouldn't have as good an accuracy score. At the same time, if I had not included a stopping criteria, it would have trained the full 50 epochs and been at risk for overfitting.



```
34/34 [==============================] - 0s 2ms/step - loss: 0.1488 - accuracy: 0.9787 - val_loss: 0.4068 - val_accuracy: 0.8400
Epoch 22/50
34/34 [==============================] - 0s 2ms/step - loss: 0.1354 - accuracy: 0.9796 - val_loss: 0.4073 - val_accuracy: 0.8400
Epoch 23/50
34/34 [==============================] - 0s 2ms/step - loss: 0.1250 - accuracy: 0.9796 - val_loss: 0.4081 - val_accuracy: 0.8383
```

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.**

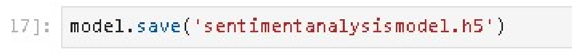
   My model had a training accuracy of 98.0% and a validation accuracy of 83.8%. I used a combination of stopping criteria and epoch definition to address overfitting.

4. **Discuss the predictive accuracy of the trained network.**

   My model correctly predicts its target 83.8% of the time. Its prediction loss is 0.41. We want a number as close to 0 as possible for loss, so this is not the best number, but not the worst either.

Part V: Summary and Recommendations

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



```
17]: model.save('sentimentanalysismodel.h5')
```

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

My model used 2400 reviews with their recorded sentiments to train, and 600 reviews with their recorded sentiments to validate. I used a Recurrent Neural Network with five layers to analyze if reviews were positive or negative based on the words used in the reviews. The impact of this architecture is that my model is tuned to predict sentiment on unseen reviews with 83.8% accuracy.

G. Recommend a course of action based on your results

I recommend using this model to assign sentiment to reviews that are text only, and need an accompanying positive or negative sentiment to be attached.

Data acquired at: (Kotzias et. al, 2015)

Third party code sources: (Brownlee, Display Deep Learning Model Training History in Keras, 2019)**,** (Onstott, 2020)**,** (Virahonda, 2020)**,** (Elleh, 2021)

Works Cited

Brownlee, J. (2019 , October 3). *Display Deep Learning Model Training History in Keras*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/

Brownlee, J. (2020, August 20). *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Brownlee, J. (2021, February 2). *How to Use Word Embedding Layers for Deep Learning with Keras*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

Elleh, F. (2021). Retrieved from https://wgu.webex.com/wgu/ldr.php?RCID=835a355c56619d28923337bcaa040a70

Kotzias et. al. (2015). *From Group to Individual Labels using Deep Features.* KDD. Retrieved from KDD.

Malin, M. (2021, February 10). *Why You Should Always Use Feature Embeddings With Structured Datasets*. Retrieved from Towards Data Science: https://towardsdatascience.com/why-you-should-always-use-feature-embeddings-with-structured-datasets-7f280b40e716

Onstott, M. (2020, January 29). *Sentiment Analysis and Classification of Amazon, IMDb, and Yelp Reviews*. Retrieved from github: https://monstott.github.io/sentiment_analysis_and_classification_of_amazon_imdb_and_yelp_reviews

Sachdev, H. S. (2020, January 23). *Choosing number of Hidden Layers and number of hidden neurons in Neural Networks*. Retrieved from LinkedIn: https://www.linkedin.com/pulse/choosing-number-hidden-layers-neurons-neural-networks-sachdev/

Smartboost. (2020, July 7). *Deep Learning Vs Neural Nework: What's the Difference?* Retrieved from Smartboost: https://smartboost.com/blog/deep-learning-vs-neural-network/

Virahonda, S. (2020, October 8). *An easy tutorial about Sentiment Analysis with Deep Learning and Keras*. Retrieved from Towards Data Science: https://towardsdatascience.com/an-easy-tutorial-about-sentiment-analysis-with-deep-learning-and-keras-2bf52b9cba91