**Part I:  Research Question**

A1: Our research question is how we can minimize customer churn when looking through customer reviews. We will minimize customer churn by building a neural network to parse customer reviews. Our neural network will be able to identify customer review sentiment. Unhappy customers will be flagged and then contacted by our customer happiness outreach team.

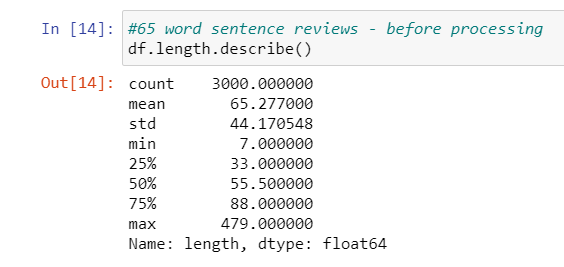
A2: The goal of our data analysis is to build a TensorFlow model which identifies review sentiment based on customer reviews. Through building this model, we can then sift through our customer’s reviews to identify if our customer base is happy to minimize customer churn. We can then use our model to identify sentiment based on new sentences.

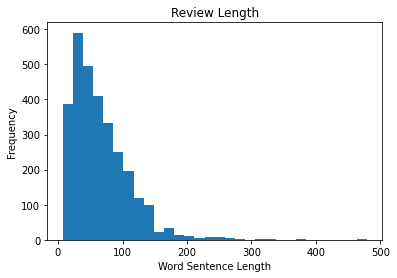
A3: One type of neural network we will be using in order to perform text classification is the LSTM model. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.

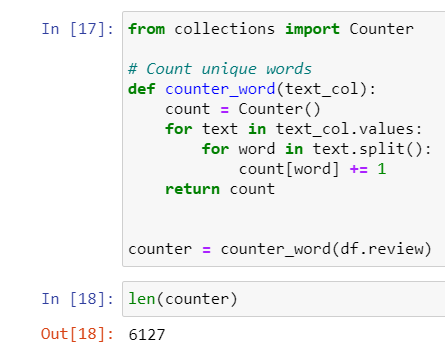
**Part II:  Data Preparation**

B1:  
Our review data contains punctuation such as html separators \t, \n, csv separators, and some non-English characters. Before processing, our reviews contain an average of 65 words per sentence. We have a vocabulary size of 6127 words.

**Pre-Processing:**

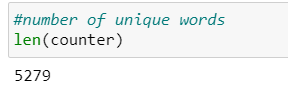


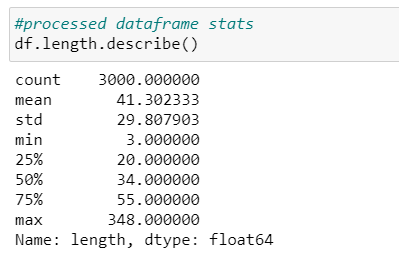


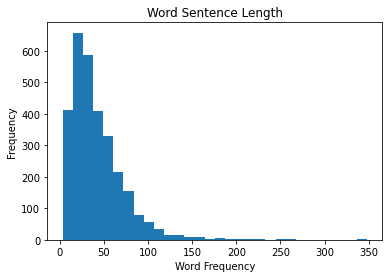


After processing our data, we get a unique vocabulary size of 5279. We set our proposed word embedding length to the number of unique words. In this case we will use 5279 for our proposed word embedding length. After processing our data, the sentence length has an average of length of 41 words. We will select 41 for our maximum sentence length. We can statistically support this average through graphing the sentence frequency length below.

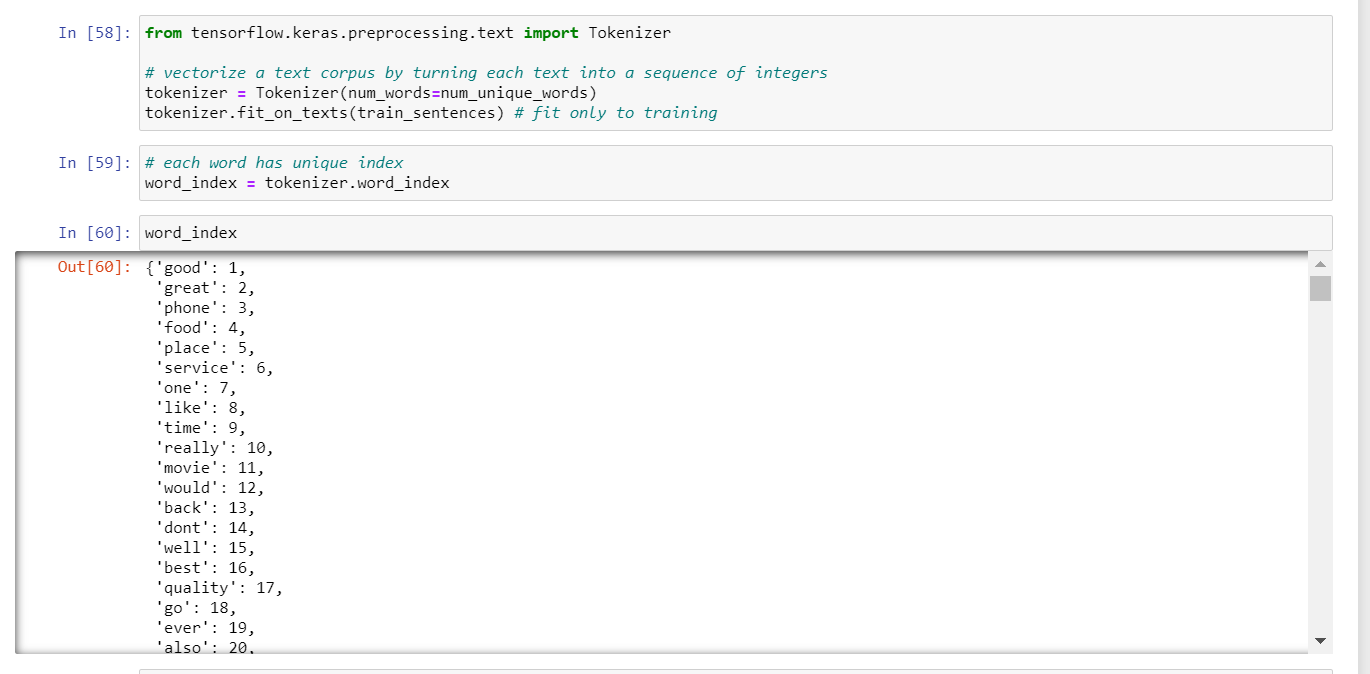
**Post-Processing:**

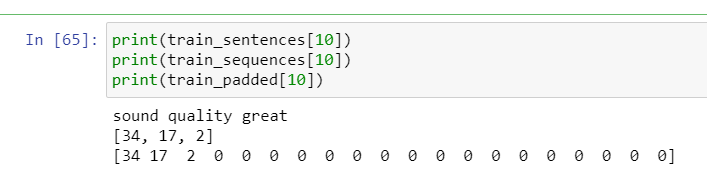






B2: The goal of the tokenization process is to create an index for each word in our review data. Chakravarthy (2020) describes tokenization as breaking sentences into raw words or chunks. We can tokenize our review data through using the Tokenizer function from the tensorflow.preprocessing.text library. Tokenizer allows us to set the number of words in our word index. In our case we set the word index count to the number of unique words and fit our model to the training review data.

  
  
B3: The padding process for our tensorflow model works by either adding zeros in the begginng or end of our index. By padding the data, we are making sure that our index legnth is uniform for each word.For our model we put the padding at the end of each word index.



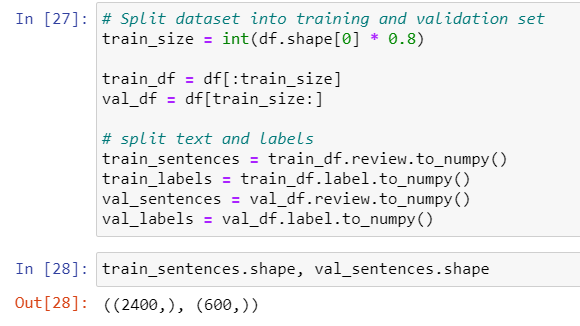
B4: For our model we will be using two categories of sentiment either positive or negative. This binary setniment allows for us to use a sigmoid activation. A sigmoid activiation allows us to either lean towards a binary outcome similar to logistic regression.

B5: In order to process our data, we first had to run a function that removes any string punctuation through the use of regular expression. After removing string punctuation, we then removed stop words using the stopwords function from the NLTK library. Once we have processed the review data, we can then split our data into training and testing data. We used 80 percent of our data for training and 20 percent of our data for testing.

**Processing Data:**

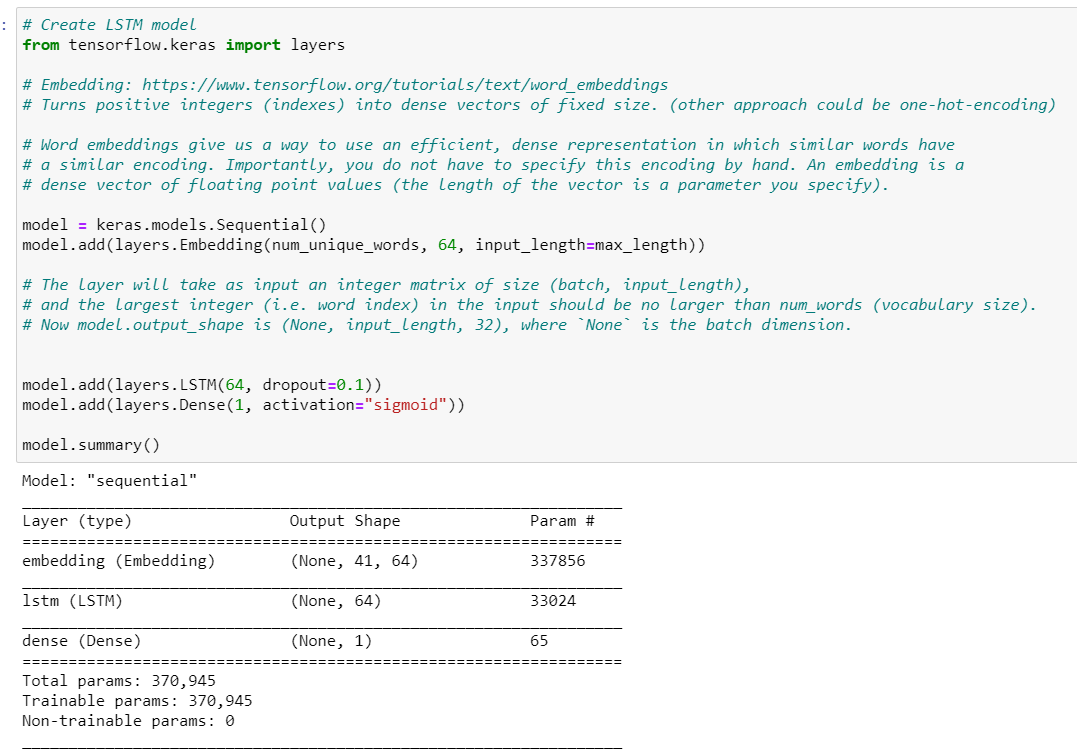


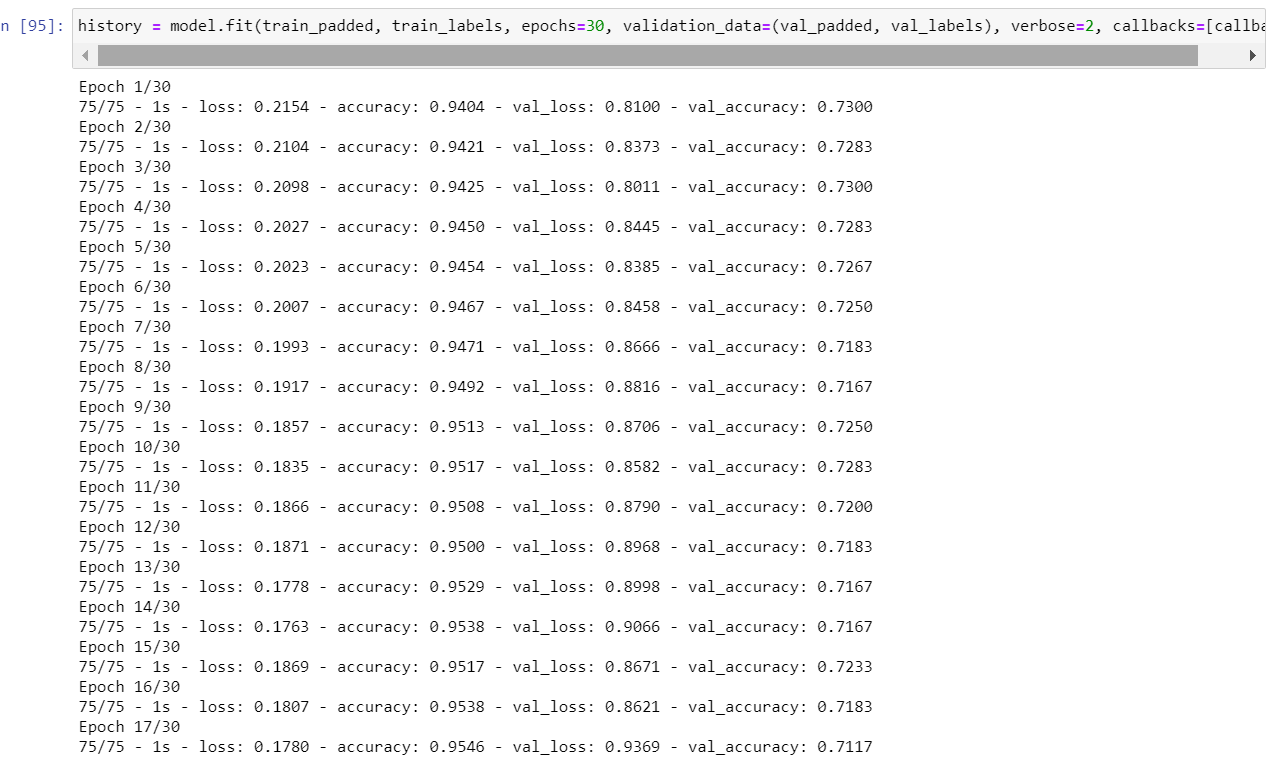
**Train/Test Split:**



**Part III:  Network Architecture**

**C1:**





C2:

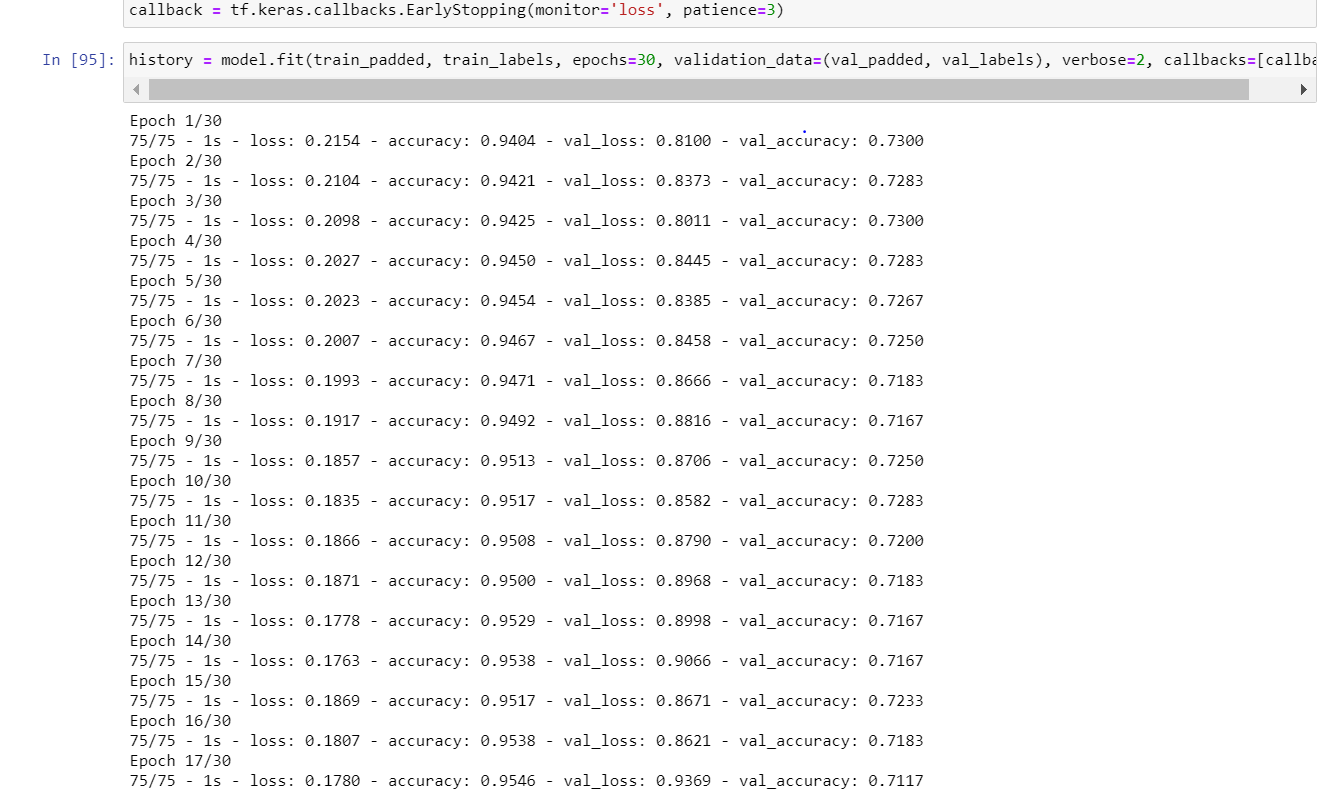
For our TensorFlow model we have three layers, starting with the word embedding layer. Word embedding allows us to see which words have a similar encoding. I set the output equal to the number of unique words. In our embedding layer we can set the shape of the embedding vector. In our case I chose the vector shape of 64 due to the size of our data. I set the input length equal to our average sentence length. Next, I added an LSTM layer with the number of output units as 64. We set the dropout length equal to 10 percent. Finally, I added a density layer set to a sigmoid function looking for an output of 1.

C3:

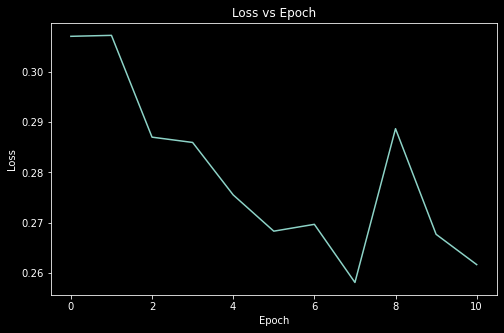
* We will use a sigmoid activation node because we are looking for a binary outcome with sentiment.
* We picked 64 nodes per layer because of the accuracy when compared to 32 and 128. 64 node layers performed better than 32 and 128.
* We chose a loss function of binary cross entropy because it supports our binary classification outcome.
* We will use an Adam optimization due to its ability to easily handle gradient descent. There’s a momentum aspect tied to Adam optimization which allows it to not get stuck in local minimum of loss. The Adam optimizer works well with sequential models because the optimizer itself is a sequence formula.
* Our stopping criteria will be set to loss with a criterion of 3. Our model will have 3 chances to score lower than our minimum loss value or model will stop.
* Our stopping evaluation metric is accuracy because this is a good metric for binary classification.

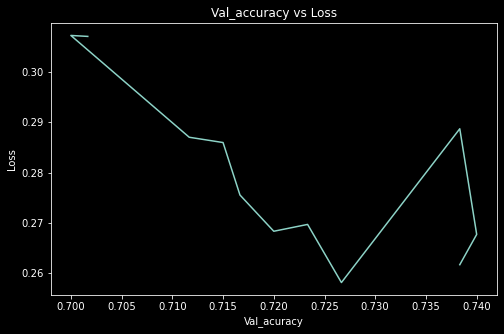
**Part IV:  Model Evaluation**

**D1: In order to avoid our model from overfitting we can use the stopping criteria instead of defining the number of epochs.** Either loss/accuracy values can be monitored by early stopping call back function. If the loss is being monitored, training comes to halt when there is an increment observed in loss values. Or, if accuracy is being monitored, training comes to halt when there is decrement observed in accuracy values. For our model we chose to observe loss and set the patience to 3 steps.



D2:





D3: Due to our model’s accuracy sitting higher than the val\_accuracy, we can conclude that our model is overfitting. In order to improve our model, we can either add additional dropout layers or continue experimenting with parameters. Accuracy sits at around 90 percent while val\_acruacy sits at near 75 percent.

D4: In total predicative accuracy tied to our val\_accuracy at around 75 percent. I would say our model is accurate but could use improvement. I would suggest adding additional dropping layers to prevent overfitting.

**Part V:  Summary and Recommendations**

**F:** Overall, our LSTM is the best functional network for our project. LSTMs look at one element at a time in a sequential matter. Compare this to a transformer which looks at the whole sequence at once. LSTMs are a subset of the Recurrent Neural Network (RNNs), while GRUs are the other major RNN. The key difference between GRU and LSTM is the way the models input and forget data. Phi (2020) describes GRU's as having two gates reset and update while LSTM has three gates input, output, and forget. If the dataset is small, then a GRU is preferred. Otherwise, LSTM is better for bigger datasets. Through using an LSTM model, we can identify negative customer reviews to minimize churn.

G: Based on our results, I would use our LSTM model in order to parse customer reviews. Our model sits at around 75 percent for accuracy. I would continue to improve upon our model by adding additional dropout layers or changing the parameters for our neural network. We can then use our model to identify unhappy customers to minimize customer churn.

**References:**

Chakravarthy, S. (2020, July 10). *Tokenization for natural language processing*. Medium. https://towardsdatascience.com/tokenization-for-natural-language-processing-a179a891bad4.

Phi, M. (2020, June 28). *Illustrated guide TO LSTM's And GRU's: A step by step explanation*. Medium. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.