**Data preprocessing:**

To preprocess the data, we began by checking the data types. We then converted the date into a datetime format. The data was checked for null values, any of which were removed. This analysis does not require extensive preprocessing to run with the LSTM model.

**Description of feature engineering and the feature selection, including decision-making process:**

Date and Adjusted Close price (Adj\_Close) were chosen as the features for this model to best portray the predictions for each stock. The adjusted close price was chosen over the close price because the adjusted close is a more accurate representation of the stock’s value. The close price only reflects the cost of the shares at the end of the day. Adjusted close accounts for other things such as dividends, stock splits, and new stock offerings. scaled the data to normalize the data in a 0 to 1 range. Then converted the data into a Numpy array before reshaping the data to fit the 3D model.

**Description of how data was split into training and testing sets:**

The adjusted closing price was extracted into a new dataframe, then converted into a time series. 80% of the data was then split into the training set and the remaining 20% into the testing set. Data was group by 60-day segments to train the model. The data was then converted in to a Numpy array which is the format accepted by Tensorflow for training, then reshaped into a three-dimensional array to work with the LSTM model. The remaining 20% of normalized data was processed for the testing sets in a similar fashion as the training sets

\*Insert the picture of the training portion here\*

**Explanation of model choice, including limitations and benefits:**

For this project, an LSTM model was necessary to perform this analysis. It is difficult to train RNNS to capture long-term dependencies because the gradients tend to either vanish or explode. This is referred to as the vanishing gradient problem, where the gradient shrinks the further back in time it goes. Too small a gradient won’t allow for good machine learning. Due to this, a vanilla RNN was excluded after the first analysis attempt.

Instead, a Long Short-Term Memory (LSTM) was chosen for this model because unlike other recurrent neural networks, the LSTM model has a large memory capacity and is able to store past information. LSTM is one type of RNN used to learn order dependence in sequence predictions. Unlike traditional RNN’s, the LSTM model has gates that control the flow of information. An LSTM model has the capability to learn which data is or is not important within the sequence. These models are great for stock predictions because the future of a stock price is dependent on the price history.

There are a few potential draw backs of using the LSTM model. The main drawbacks for this model are;

* The training process is longer
* They require more memory to train (cannot be done in cloud due to scaling)
* Prone to overfitting

**Explanation of changes in model choice (if changes occurred between the Segment 2 and Segment 3 deliverables):**

The original model choice was a normal RNN until we realized we were working with Timeseries data and that a standard RNN would be unable to retain enough information to properly train the model. We then chose a LSTM rather than a GRU as this is the most common practice for stock prediction

* Model choice updated to reflect all stocks of interest – CHWY, ELAN, FRPT, PETQ
  + tickers were implemented which allows for filter based on that ticker

**Description of how they have trained the model thus far, and any additional training that will take place:**

* The model was trained by fitting it to the previously separated testing set data. To do this, an optimizer and loss function was applied.
* Insert picture HERE \*
* For this project, the “adam” optimizer for its fast results and works well with large datasets.
* The model was then fit to training sets using a batch\_size of 1 and run for 5 epochs.

**Description of current accuracy score:**

* The LSTM model uses a root mean square error (RMSE) metric to determine the accuracy and performance of the model. The close to 0 that the RMSE score is, the more accurate the model is performing. When running the RMSE for this model, our team ended with a score of 1.0159546093459928, which indicated that the model is performing well.

Normal RNNs are like people with short term memory issues, there is a loop that allows them to use past information before coming to a final output (like a thought), however they can only use recent previous info rather than long term info. Gradients are values used to update a neural networks weights

To fix this issue, LSTM’s and GRU’s (gated recurrent unit) were created.

* They have internal mechanisms, aka gates, that regulate the flow of information and are able to learn which data is important and to be kept within a sequence
* By doing this, the info can be passed through many sequences in order to make predictions