I would like to use RNNs to predict stock prices. My clients are people who are seeking to maximize their retirement income and income for their future expenses, like college tuition for their children. I am going to use stock market data available on the website kaggle. My aim is to use recurrent neural networks to predict stock prices. I want to make models that take into account different time frames, such as weekly, and daily predictions. In the best case scenario, I want to produce a model that can predict stock prices at some accuracy above 60 %.

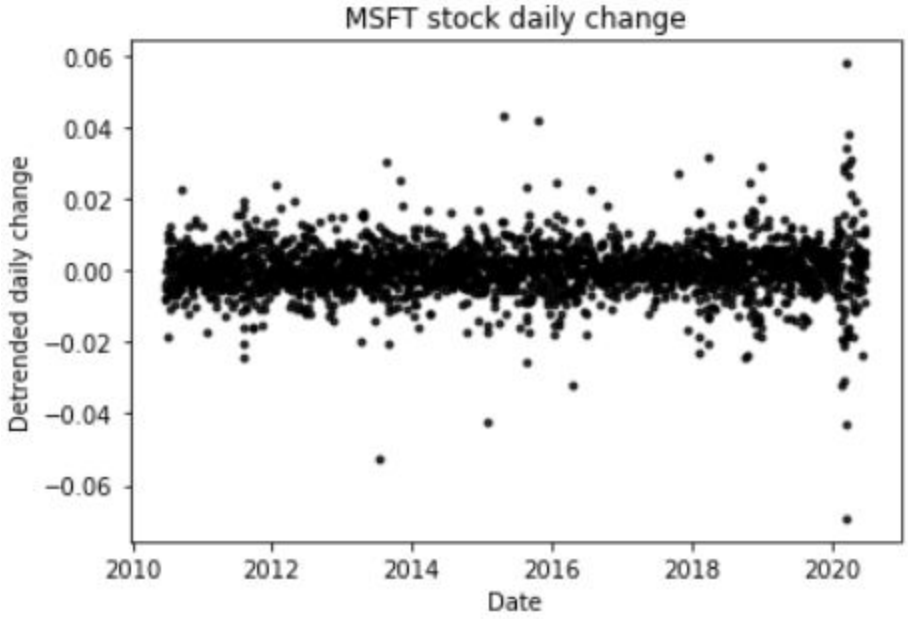
Towards this goal, I have downloaded a ten year dataset of Microsoft stock data. The data set includes the daily high price, the daily low price, the daily opening price, the daily closing price, and the volume. I have also downloaded a data set of analyst recommendations for the stock over the last eight years. I also downloaded Microsoft stock data from the German stock exchange Xetra. These data also included the daily high price, the daily low price, the daily opening price, the daily closing price, and the volume.

The above figure shows the Microsoft stock price over the selected interval. Note the exponential nature of the stock price.

I plan on trying to correct for the exponential change in stock price through two methods. The first method I would like to try is correcting for the exponential increase in stock price through taking the $\log_{10}$ of the stock price.

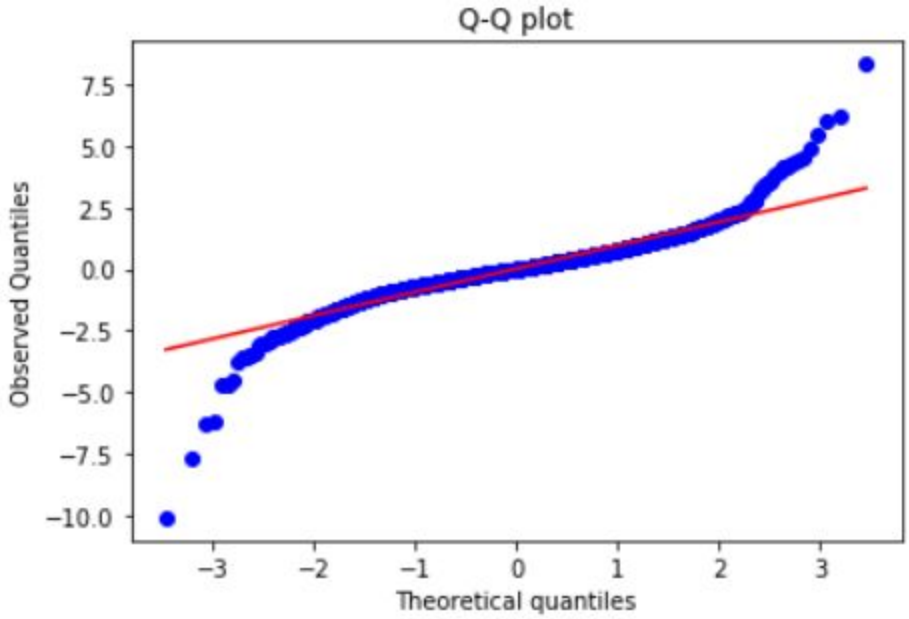
The above figure shows the $\log_{10}$ of the Microsoft stock price at its close. Note that the data still show some non-linearities.

The next step in this analysis will be to examine the daily change in the $\log_{10}$ of the closing price.



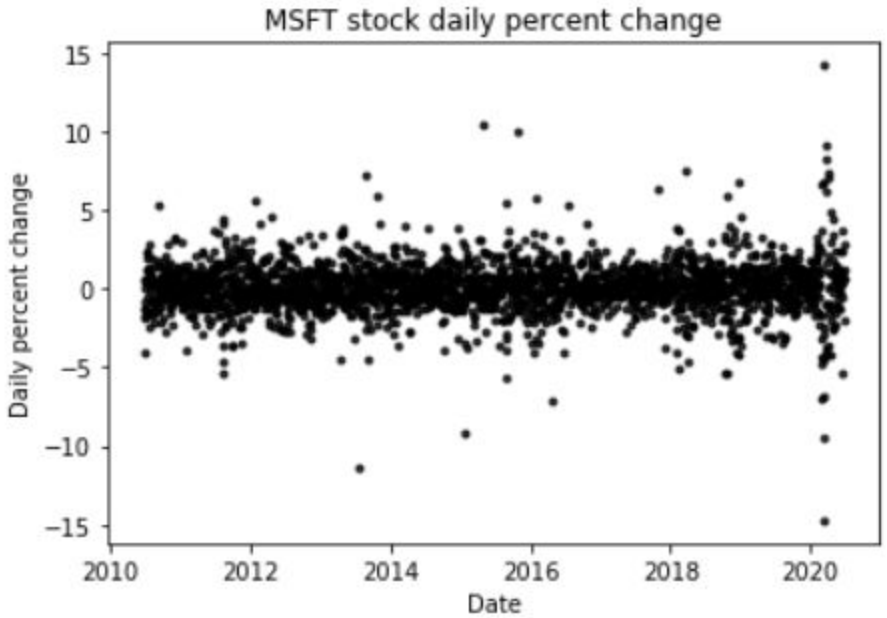
The above figure shows the daily change of $\log_{10}$ of the Microsoft stock price at its close.

This step produces largely detrended data. The q-q plot of this data when normalized shows a distribution of returns that approximates a normal distribution between -2 and 2 standard deviations from the mean. However beyond those values, the distribution has much flatter tails than a normal distribution does.



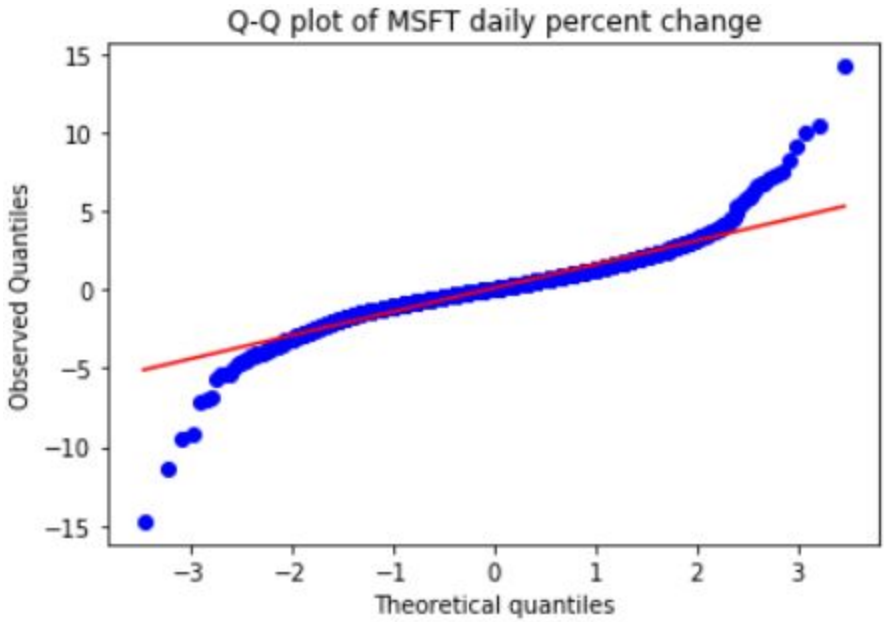
Notice that at quantiles above and below 2 and -2 the observed quantiles occur less frequently than would be predicted in a normal distribution.

The other method I wanted to examine for stock market prediction was the percent daily change in Microsoft stock price.



The above figure shows the daily percent change between Microsoft stock closing prices.

The q-q plot of the daily percent change in the Microsoft stock closing price produced a distribution that was similar to the $\log_{10}$ transformed data, but the magnitude of the tails was greater.



My next steps in building a preliminary recurrent neural network model were download analyst ratings of Microsoft stock. Yahoo had a dataset of analyst ratings of MIcrosoft stock extending roughly eight years. This dataset consisted of the firm, the rating, and whether the stock had been upgraded or downgraded. I converted these columns containing qualitative variables into multiple columns via one hot encoding. I used these columns along with the change in volume of traded Microsoft stock and the percentage change in Microsoft stock to see if I could predict where the stock would be higher in three days based on the information in the 60 preceding days of trading.

I created a recurrent neural network that runs for 10 epochs. I used 3 layers of long short term memory followed by two dense layers with the final dense layer predicting whether or not the stock would increase or decrease in 1 day based on the events of the preceding 60 days. The model currently takes into account the aforementioned analyst ratings, and daily changes in stock open, close, high, and low prices, as well as trading volume calculated as percent changes the previous day's closing price and trading volume.

The model currently can predict whether the validation test set will increase or decrease with accuracies that range from about 48 to 60 % with a mean of about 54 %.

```
 2/25 [=>............................] - ETA: 4s - loss: 0.9560 - accurac
y: 0.5312WARNING:tensorflow:Method (on_train_batch_end) is slow compared
to the batch update (0.131725). Check your callbacks.
24/25 [============================>..] - ETA: 0s - loss: 0.9107 - accurac
y: 0.5085WARNING:tensorflow:From C:\Users\webster\Anaconda3\lib\site-pack
ages\tensorflow\python\ops\resource_variable_ops.py:1817: calling BaseRes
ourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops)
with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
INFO:tensorflow:Assets written to: models/RNN_trial-01-0.56.hdf5.model\as
sets
25/25 [==============================] - 7s 274ms/step - loss: 0.9106 - a
ccuracy: 0.5085 - val_loss: 0.6929 - val_accuracy: 0.5588
```

The above figure shows the output from the RNN, note that the validation accuracy (val_accuracy) is 55.88 %.

A drawback of the current model is that the validation test set is relatively small and the model would likely benefit from more data. The dataset is limited by the timescale of the analyst ratings. I would also like to add in other data from other stocks, like Amazon (AMZN), or Google (GOOGL) to see if this will improve performance. I am also thinking of trying causal convolutional networks to determine if this improves model performance.