Forex Data Prediction via Recurrent Neural Network Deep Sequence Modeling

Authors: Edmund Adewu, Diego Kervabon, Mohammad Mahmud, Oleksandr Taradachuk, Eugene Sajor

January 9, 2022

1 Introduction

The subject of our final project is the prediction of Forex price changes. For the uninitiated, Forex - short for "foreign exchange market" - is a global decentralized market that primarily facilitates the trading of various currency pairs. The value of any single currency, much like the value of any single stock, can fluctuate over time. Our group decided to try predicting the movement of currency prices over a predetermined period of time. To that end, we analyzed the price changes of the British Pound Sterling relative to the Japanese yen over the course of a decade.

2 Data Utilized

2.1 Data Pre-Processing Cleanup

Data Pre-processing is a data mining technique that involves altering raw data into a more usable format. Raw data tends to be inconsistent, incomplete, and have incorrect or missing information. In order to prevent low quality data from affecting our model's results, proper data pre-processing is required.

We used the Data Integration technique for our pre-processing. This involves converting JSON data into CSV data. Since polygon API restricts us from getting all the data at once, we used data integration to combine data from multiple API calls. Our first step was to create a CSV file with 4 hour data points. We used the CSV file from our local system to train our model. Our second step was to create a database to access the data from anywhere, not just our local system. The data base used is MongoDB that holds 4 hour data points. The 4 hour data point includes timestamp, OHLC (open, high, low, and closing) prices, the trading volume (v), the volume weighted average (vw), and the average OHLC. In addition, we created a 1 hour data set using the same technique, however, we were not fully able to train our model on it due to usage limits on Google Colab.

As previously mentioned, the British Pound Sterling and the Japanese yen were evaluated over a 10-year period, from January 2010 to December 2020. The data was retrieved from the Polygon API. A function was used to access the Polygon API in 1 hour time-frames in the span of a month. Each monthly batch came in as a JSON file which then underwent conversion to a pandas DataFrame before being further converted into a CSV file. A total of 68,293 data points, covering the time period from 2010-01 to 2020-12 in 1 hour time frames, were generated.

2.2 Input Data Generation

Seeing various different models online, we found issue with the fact that they used prices as inputs to their model. Prices are sensitive to trends which make it unclear what a the next price could be and are generally inconsistent. Instead, we chose to use percent changes in price as inputs to the model as it bounds the data and provides consistency. Additionally, percent changes in price make much more sense to use than prices because when we analyze trends and movements in the markets through changes in price - not prices on their own. Percent changes in price were calculated as the percent of the current price needed to get to the next price.

3 Experimentation

3.1 Long-Term Price Prediction Over Time Model

Our first model worked by predicting price movement over a desired time-frame. This was done using a Stacked LSTM architecture with dropout layers to prevent over-fitting. A visual of of the architecture can be seen below:

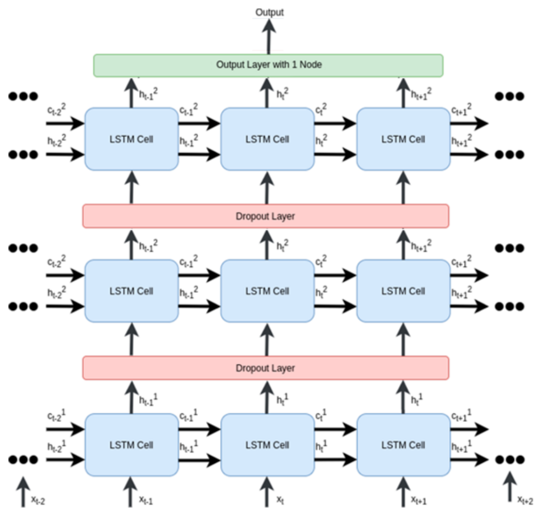


Figure 1: Price Prediction Over Time Model

LSTM cells were picked because they solve the vanishing gradient problem found in Recurrent Neural Networks and are usually the first choice when dealing with time-series data such as market prices. We chose to stack three LSTM layers on top of each other to add layers of abstraction with the intent of having the model create an intermediary analysis of the data similar to adding layers to a Dense network. Dropout layers, layers where input nodes have a probability of being dropped out and not being used for the current prediction, have been shown to be effective at preventing over-fitting which is why we included them in between each LSTM layer. In the final model, our dropout layer had a dropout rate of 20%. We trained our model using the Adam optimizer and total training lasted for 120 epochs. Loss during training can be seen below:

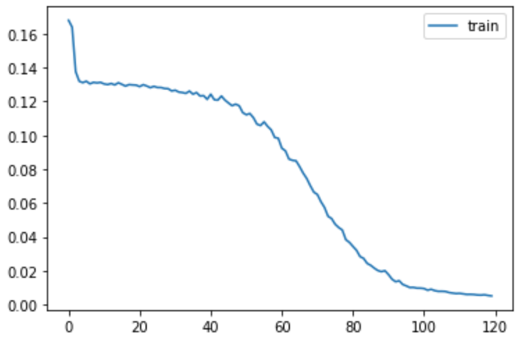


Figure 2: Price Prediction Over Time Loss

Interestingly enough, the sigmoid shape of the loss function could indicate that we are over-fitting to the training data and the results seem to support that. This could be alleviated by increasing the dropout rate, perhaps to 50%, or decreasing the number of LSTM layers however further experimentation is required to be certain.

3.1.1 Testing the Model & Findings

When it came to testing the model, we saw many online models being flawed in the sense that they were not predicting prices based on their own predictions. Instead, they simply ran the model for each slice of the test data which would not work with real-time data as future prices are unknown. Experimentation with the price prediction over time model led us to the conclusion that long-term price prediction was not feasible given the amount of external variables at play that compounded over time. As we can see from the figure below, after getting through the training data, the model quickly falls off and has trouble predicting into the future:

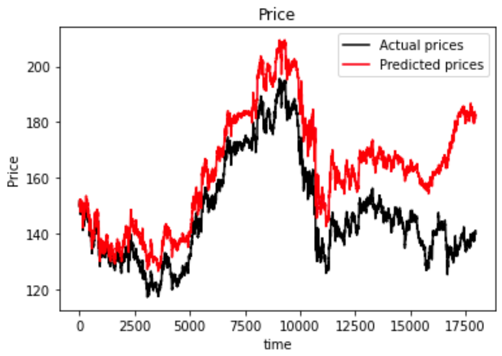


Figure 3: Price Prediction Over Time

The graph above is the actual price movement of GBP/JPY and was created by taking the same initial price and applying the percent changes of the actual price movements as well as the predicted price changes. The resulting vectors of prices were then plotted to achieve the above graph.

3.2 Short Term Categorical Price Prediction Model

The aforementioned impracticality of the long-term price prediction model led us to pursue a model that could be used as more of a trading tool. Based on previous prices, we attempted to predict not only the price increase or decrease in the next time frame, but also the magnitude of said movement. Because the output was categorical, the model would also be able to provide confidence metrics which could have been leveraged to find optimal times to enter the market. This model had a very similar architecture to our first model, however its final dense layer now had 7 inputs for each of the possible price movements we categorized. The new architecture can be seen below:

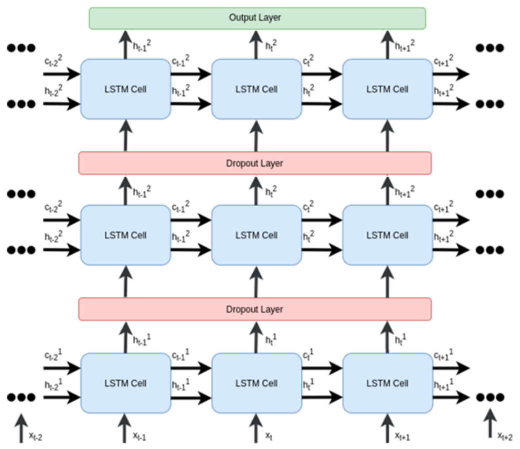


Figure 4: Categorical Price Prediction Model

3.2.1 Training the Model

When using outputs for prediction, a softmax is applied to get a probability distribution of which prediction is most likely the true one. Because softmax isn't applied in the actual architecture, we had to make sure the optimizer, Adamax in this case, applied a softmax over the outputs before calculating loss. The categorical model was trained for 300 epochs and the loss function is very similar to the initial loss function we saw with the price prediction over time model as can be seen below:

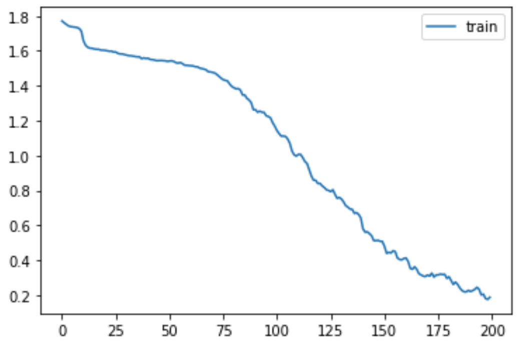


Figure 5: Categorical Price Prediction Loss

3.2.2 Testing the Model & Findings

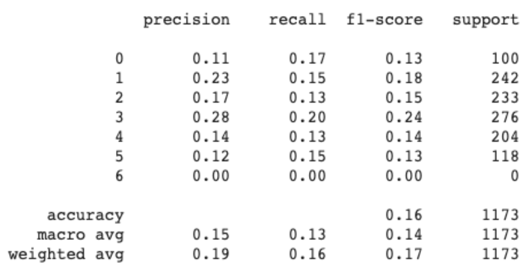
The approach we took for this model resulted in much better performance than random-choice (25.4% vs. 13.6%) however was comparable to simply choosing the most likely outcome of the data-set (25.4% vs. 24.1%). Utilizing the confidence of the model, which was thought to be promising, resulted in accuracy dropping all the way down to 14.1% which made it comparable to random-choice. There are many reasons we believe this is happening and we reflect on that in the Future Work section.

3.2.3 Confusion Matrix

The confusion matrix below, shows our model's performance in analyzing Forex market price movements as categorical data. The left side of the plot represents the actual and the bottom represents the predicted. It is also important to note that categories were determined by a factor of the standard deviation since the data was normally distributed which allowed us to get fairly equal categories. The categories are represented as

- 0 - Big Move Down (:-0.97*z]
- 1 - Medium Move Down (-0.97*z:-0.35*z]
- 2 - Small Move Down (-0.35*z:0)
- 3 - Small Move Up (0:0.35*z)
- 4 - Medium Move Up [0.35*z:0.97*z)
- 5 - Big Move Up [0.97*z:)
- 6 - No Movement

The metric that matters the most for our model is accuracy since it determines the number of correctly predicted price movements whether it be a big move up, big move down, medium move up, medium move down, small move up, small move down or no movement at all. Precision and recall don't have as much of a bearing since classifying something falsely as moving up has the same consequences as falsely classifying something as moving down. Looking through the model we can see that there were 0 predictions for category 6 which states that there was no movement in the market which makes sense since the Forex market is always active. Small moves up were the most accurately identifiable category and had the highest precision which could be a result of it being a slightly bigger group than the other.



```
              precision    recall  f1-score   support

           0       0.11      0.17      0.13       100
           1       0.23      0.15      0.18       242
           2       0.17      0.13      0.15       233
           3       0.28      0.20      0.24       276
           4       0.14      0.13      0.14       204
           5       0.12      0.15      0.13       118
           6       0.00      0.00      0.00         0

    accuracy                           0.16      1173
   macro avg       0.15      0.13      0.14      1173
weighted avg       0.19      0.16      0.17      1173
```

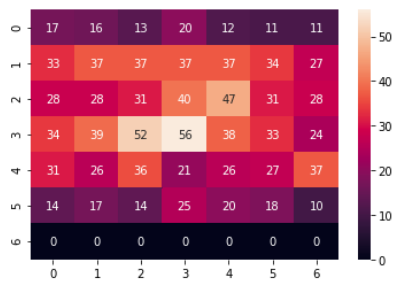


Figure 6: Confusion Matrix of Categorical Prediction

4 Future Work

One main focus of future work would be making the model multivariate instead of simply relying on percent changes in price as it currently does. Perhaps including volume, different time-frames, or even news would help in making accurate predictions and avoiding over-fitting. Categorical Price Prediction requires further testing and experimentation of hyper-parameters as we were limited by time and usage limits imposed by Google Colab. Hourly data might provide more interesting results as the higher amount of data would be better suited for deep-learning. As it stands, however, our models support markets being unpredictable though need further testing.