Certificate in Quantitative Finance

Final Project

January 2022 Cohort

Deep Learning for Financial Time Series

Brad Smallwood

August 22, 2022

Contents

Abstract

Long-Short Term Memory models were used to classify directional movement of the USDCAD currency pair. A variety of technical indicators, commodity prices, index prices, and other currencies were used as features in this model. A mix of feature selection and dimension reduction methods were explored to reduce the number of features as well as explore the data further. Multiple LSTM models were fit to see which combination of features, and hyper parameters could best predict the direction of USDCAD. After identifying which LSTM model performed the best, as well as which set of features were most predictive, the predictions from this model were built into a secondary LSTM model which was used to determine if a position should be taken.

1 Introduction

1.1 Overview

For the final project I chose to work on the deep learning problem as I felt that the material I would learn over the course of completing this project would be the most relevant to my current position as a data scientist. Moreover, I wanted to combine what I learned in my previous role as an foreign exchange trader to to see if I can produce a viable trading strategy that could have been run on our desk. In the third exam I developed a very simple strategy using an XGBoost model, wherein the model would make predictions one day ahead as to whether a stock should be bought, or whether no action should be taken. While this approach might be fine as a tool for trading on my own portfolio, where I typically just find buying opportunities and hold the stock for several months to several years, it would not get very far when trading currencies and especially not in a professional environment. Instead, the directional movements (up, down, and no change) were predicted for the USDCAD currency pair with some additional adjustments of the labelling criteria to make the applicability of a trading system based on a machine learning model's predictions more realistic.

One of the major issues which prevents my approach from Exam 3 being applied to a real world FX trading desk (at least at the one I worked at) is the lack of take profits (TP), and stop losses (SL). My approach from Exam 3 ignores any TP or SL orders that I would have often used in practice when taking a position. Often times, it wouldn't even be a choice for me to use a stop loss order as I would be receiving angry phone calls from my manager, or even worse the folks in risk, if I were to not have a SL in place. For this project, the triple barrier labelling approach was used to overcome this shortcomings of traditional machine learning labelling when applied to financial problems. In addition to this, the labels were further augmented to a meta labelling approach where the algorithm would also learn the optimal bet sizing, as opposed to how it was done with Kelly optimal bets in Exam 3.

As per the requirements of this project, I've implemented multiple Long Short-Term Memory (LSTM) models to make classifications of directional movements. The features used in these models included technical indicators, as well as commodity prices, index prices, and other currencies. To reduce the large number of features being analyzed, dimension reduction techniques such as principal component analysis were used, as well as feature selection techniques such as the Boruta algorithm and self organizing maps. These approaches were compared to see which produced the most accurate predictions before moving onto a final LSTM model which used the predicted side of the the order as an additional feature in the model when predicting whether the order should be placed or not.

1.2 Data Collected

Daily data from January 1st, 2012 to June 30th, 2022 was collected through Oanda, an FX broker for retail traders. USDCAD open-high-low-close (OHLC) time bars were used as the main source of data in this project and were used to build all the technical indicators. In addition to this, OHLC data was also collected on the EURCAD and CADJPY pairs as well as WTI, gold, S& P 500, copper, and the US 10 year bond yield (all denominated in USD). The Euro and the Yen were chosen as features due to high volume of transaction and due to them being the primary currency traded in their respective sessions. WTI was chosen as there has historically been a strong relationship between oil prices and the Canadian dollar. The S& P 500 and US 10 year bond were chosen to understand the dynamics that impact the U.S. dollar, specifically how investor sentiment can cause rallies or declines. It is often observed that when there is a significant risk off move (i.e. pullback in equities) that we see investors purchasing bonds which needs to be done with U.S. dollars. This then can cause the U.S. to appreciate against the Canadian dollar. Similar dynamics can be observed in the Japanese Yen, another safe haven currency. This same dynamic can also be seen in gold which typically rallies when there is significant risk-off sentiment in the market. Lastly, copper prices were also included for largely the same reason as WTI. While the relationship between copper and the Canadian dollar is not going to be as strong as WTI, it might still be useful to see if a rally in a commodity such as copper is also related to a rally in the loonie. The historical closing prices for the USDCAD pair in the time frame of this analysis can be seen in the figure below.

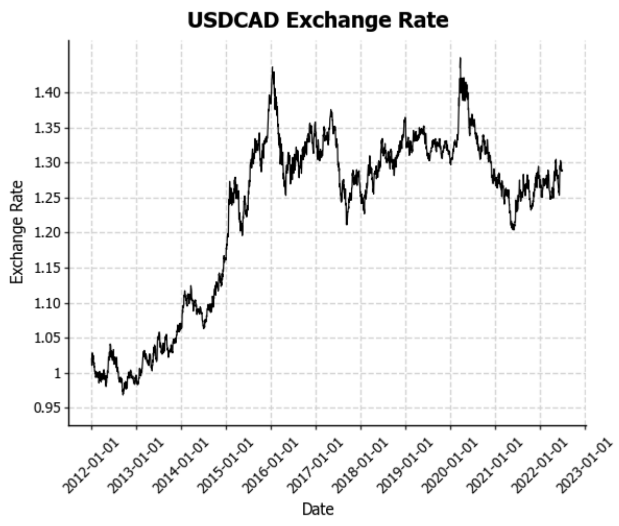


Figure 1: Historical USDCAD Closing Prices

1.3 Data Adjustments

Combining all of this data and organizing it into a wide formatted table resulted in some NaN values. Given we are using a deep learning technique on this data, know to be more data hungry, it would be preferable retain as many observations as possible. This can be accomplished with several possible imputation techniques, but care needs to be taken so as to not produce a data leak. For NaN values that were present in the USDCAD pair of the data due to a holiday in North America, or some form of an outage on Oanda, the entire row of data was dropped. For all other NaN values, produced due to no trading on that specific day for that pair, or any other reason, a last observation carried forward (LOCF) imputation was performed. LOCF was chosen to prevent any information about future prices being brought back to the time period where the NaN occurred. This was not done on the USDCAD data as it is realistic that my algorithm would not be able to trade some day due to markets being closed. There would however be days when North American markets would be open and I don't want my algorithm to be limited by holidays or closures in Europe or Asia.

2 Methodology

2.1 Triple Barrier Labelling and Meta-Labelling

If there was one thing working on Exam 3 and the final project have taught me, it would be that labelling schemes in financial machine learning are not a trivial decision. In most applications, defining exactly what I want my algorithm to learn has been straight forward. For example, in image classification we may want to determine if a picture is a hot dog or not. Applied naively, we could also classify returns in the same manner and teach our algorithm whether returns will be positive or negative, but when we consider the restrictions and other tools we have in place in the real world such as take profits and stop losses, we can begin to realize that a simplistic binary classification can't replicate what we want our algorithm to actually do. In his book, Advances in Financial Machine Learning, Marcos Lopez de Prado [dP18] presents a more realistic labelling scheme which addresses some of the shortcomings of traditional labelling.

The label on observation at time t has been defined as:

$$y_t = \begin{cases} -1 & r_{t,t+k} < -\tau_t \\ 0 & r_{t,t+k} < \tau_t \\ 1 & r_{t,t+k} \geq \tau_t \end{cases}$$

where $r_{t,t+k}$ is the return from time t to time $t+k$ and τ_t is the threshold decided at time t. To see why this definition is useful, we can imagine what might happen if we were an FX trader who just went long with a big position. Under this definition three things might happen, I would reach my TP level defined as the threshold τ_t, I wait for some given amount of time with the USDCAD not moving much, leading me to lose conviction in my view and closing my position early, or my view was wrong and my SL is triggered at the threshold $-\tau_t$. Moreover, as a trader I was always conscious of how far away I was placing my TPs and SLs. Typically, I would decide them based on the range of the USDCAD exchange rate over the last few sessions, and the volatility we had seen or might expect. The threshold τ_t is set dynamic in that it will be set at time t when the order is being entered. For this analysis I've calculated τ_t based on the 20 day exponentially weighted moving standard deviation. This labelling methodology is known as the Triple-Barrier method where $-\tau$ would form the lower barrier, τ would form the upper barrier, and should neither barrier be reached in a given amount of time then the vertical barrier will be touched.

Specifically in this project I have set the upper and lower barriers to be symmetric around the closing price of USDCAD at time t and those barriers are twice the exponentially weighted moving standard deviation over the last 20 trading session. In the original method proposed by Marcos Lopez de Prado, he used non time bars, but unfortunately that data is more difficult to access. In my version of this method I have used OHLC bars where the highs and the lows were monitored to determine if one of the thresholds had been breached. Unfortunately, there is also the scenario where both barriers could be breached, in which case my data does not allow me to know which was breached first. I have tried to adjust for this by looking at where the close the closing price was on the date of the breach but if the close is back between the barriers, then there is no way of knowing which label the observation should be assigned. In these cases the label was just set to 0. The more common reason for the label to be set to zero is when the vertical barrier is breached. I have set this vertical barrier to be 5 days after the trade was entered, meaning that if neither the upper nor the lower barrier is breached for 5 days then I exit the position with whatever return I can achieve at that time. An example of the three barrier approach can be seen in the figures below.

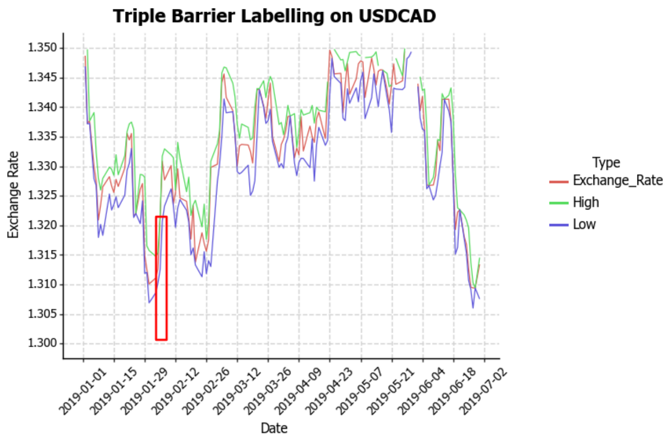


Figure 2: Example of Upper Barrier Being Breached

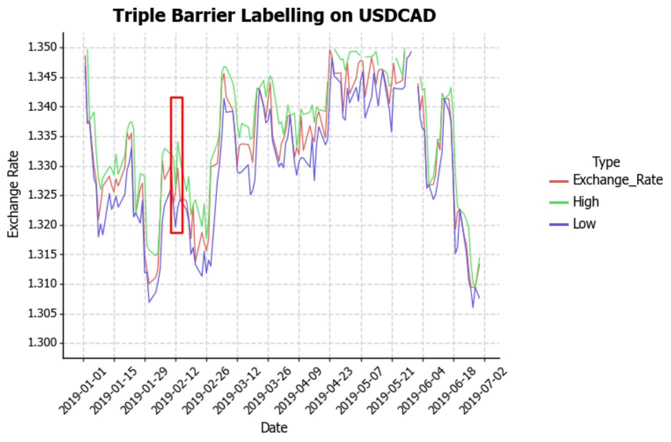


Figure 3: Example of Vertical Barrier Being Breached

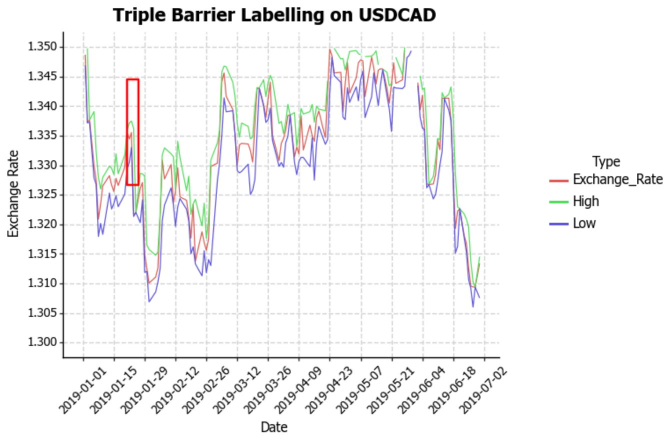


Figure 4: Example of Lower Barrier Being Breached

The Triple Barrier Labelling approach is already a significant improvement over what I used in Exam 3, but Marcos Lopez de Prado proposed another way to extend this idea even further to also address bet-sizing. While trading, I would subjectively size my own bets. For a trade I had a strong sense of conviction in I would take a larger position, but for trades I was less certain in I would take smaller positions. By using meta-labels, we can recreate the same type of thinking within our machine learning algorithm, After training our first set of models on the Triple Barrier Labels, I then use the predictions from the model as a feature within a secondary model. The labels of this secondary model have been defined as:

$$y_t^S = \begin{cases} 0 & y_t \neq \hat{y_t}^P \\ 1 & y_t = \hat{y_t}^P \end{cases}$$

where y_t is the correct label as according to the triple barrier approach from the primary model, and $\hat{y}_t^P$ is the predicted label from a model which was fit with the labels from the triple barrier approach. The the model was correct, then the new label will take the value of 1, if not 0. This new binary classification problem now results in a position only being opened if the label being predicted from the secondary model is 1 (i.e. the model feels strong about the trade). We could alternatively predict probabilities from this second model and use those to determine what size of a position the model could open. For example, we limit ourselves to a maximum of \$ 1,000,000 per trade, and the secondary model predicts a 75% probability that the first model's prediction of the top barrier being hit was correct, then we would open a long position with an exposure of \$ 750,000.

Focusing again on the class prediction of the second model, the meta labelling approach also gives rise to a nice interpretation of the metrics use to assess the predictive power of the model. There can now be four outcomes:

1. **True Positive:** $y_t = \hat{y}_t^S = 1$ (Open a position which was profitable)
2. **False Positive:** $y_t \neq \hat{y}_t^S = 1$ (Opened a position which was not profitable)
3. **True Negative:** $y_t = \hat{y}_t^S = 0$ (Did not open a position that would have been unprofitable)
4. **False Negative:** $y_t \neq \hat{y}_t^S = 0$ (Did not open a position that would have been profitable)

where $\hat{y}_t^S$ is the prediction to come from the secondary model. This will make a confusion matrix easy to interpret and also makes the use of recall, precision, and F1 as very natural metrics to use. The recall is then simply a measure of our models ability to correctly open profitable positions, while precision is the ratio between the model opening positions that are profitable and unprofitable. The F1 score remains the efficiency of our classifier. This also leads to a strategy to train the two models. The first model can be built to be better at recall, meaning it is better at correctly identifying which side of the trade to take (or no trade at all), even if it has a lot of false positives (low precision). This high false positive rate will then be corrected in the second model by applying the meta labels to the primary model.

2.2 Technical Indicators

Numerous technical indicators were included within this analysis. For the most part, they were calculated on the USDCAD pair, although some indicators were also applied to the EURCAD or CADJPY pairs.

2.2.1 Relative Strength Index

The relative strength index (RSI) is a momentum indicator which measures the speed and magnitude of the recent price changes in a security [Fer]. Its values range from 0 to 100 where the closer to 100 the index is, the more overbought the security is. For values below 30, the general consensus is that the security is oversold, and for values above 70 the general consensus is that the security is overbought. Mathematically, the RSI is defined as

$$RSI_{t+k} = 100 - \left(\frac{100}{1 + \left(\frac{AverageGain}{AverageLoss} \right)} \right)$$

where the average gains and average losses are calculate over the last k days.

2.2.2 Average True Range

The average true range (ATR) is a market volatility indicator based on the highs and lows of a security over a specified time frame [Haya]. The ATR is calculated as

$$ATR = \frac{1}{n} \sum_{i=1}^{n} TR_i$$

where $TR = max\left((H - L), |H - C_p|, |L - C_p|\right)$, and H, L, C_p correspond with the high, low, and closing price of the security.

2.2.3 Bollinger Bands

Bollinger Bands are a set of trend lines around the time series of a security [Hayb]. These bands have been calculated from a simple move average over a specified time frame. This technical indicator is generally used to generate buy and sell signals when it is believed that the security is overbought or oversold. The Bollinger Bands in this report have been calculated from the following formula.

$$BB_{Upper} = MA(k) + 2\sigma(k)$$
$$BB_{Lower} = MA(k) - 2\sigma(k)$$

where $MA(k)$ is the simple moving average of closing prices over the last k periods, and $\sigma(k)$ is the standard deviation of closing prices over the last k periods.

2.2.4 Stochastic Oscillator

The stochastic oscillator is a momentum indicator which compares the closing prices of a security to a range of closing prices for that same security, over a specified period of time [Hayc]. The measure is bound between 0 and 100, with the closer to 100 the value is, the more overbought it is considered. Typically a value of 80 is considered overbought, and a value below 20 is considered oversold. The stochastic oscillator is defined as

$$\%K = \left(\frac{C - L_k}{H_k - L_k} \right) \times 100$$

where C is the current closing price, L_k is the lowest price over the last k trading periods, and H_k is the highest traded price over the last k trading periods.

2.2.5 Drift Independent Volatility

Drift independent volatility (DIV) is a technical indicator which measures market volatility [DY00]. Unlike ATR, this indicator adjusted for jumps between closing prices of one day, and the opening prices of the next. This estimator has a lower variance compared to many other alternative volatility based technical indicators. It is defined in their paper as

$$V = V_0 + kV_C + (1 - k)V_{RS}$$

where V_0, V_C, and V_{RS} are defined as

$$V_0 = \frac{1}{n-1} \sum_{i=1} n(open_i - \bar(open))^2,$$

$$V_C = \frac{1}{n-1} \sum_{i=1} n(close_i - \bar(close))^2,$$

$$V_{RS} = \frac{1}{n} \left(u_i(u_i - c_i) + d_i(d_i - c_i)\right)$$

$$k = \frac{\alpha - 1}{\alpha + \frac{n+1}{n-1}}$$

V_RS is known as the Roger Satchell variance estimation with $u_i = \log(High_i) - \log(Open_i)$ and $d_i = \log(Low_i) - \log(Open_i)$. The authors of DIV recommend a value of 1.34 for k in their paper, which is what I used when calculating the DIV. An adjusted version of the DIV was found online and implemented in this project [Tan16].

2.2.6 Other Technical Indicators

Several other technical indicators from Exam 3 were also used in this analysis. They include:

- Simple Moving Averages
- Exponentially Weighted Moving Averages
- Momentums
- Exponentially Weighted Standard Deviations
- High Minus Low
- Open Minus Close
- Open Minus Previous Close
- Sign of Returns
- Returns

For the full list of indicators, please see the Boruta algorithm table at the end of this report.

2.3 Boruta Algorithm Feature Selection

The Boruta algorithm is a feature selection technique designed to final all relevant variables[MBK10]. The algorithm can use any tree based classifier such as a random forest or XGBoost. The algorithm works by creating randomly shuffled versions of the features, known as shadow features. The importance of each feature, including the shadow features is calculated, and the highest importance of all the shadow features is recorded. If one of the real features has a higher importance than the maximum importance of the shadow features, then it is kept in the dataset. Basically, a real feature is only kept if it is more useful than the best of the randomized features. To ensure that features aren't kept or removed just due to luck, multiple trials are performed for each feature and then modelled with a binomial distribution. If of the total number of trials, the feature is kept for most of them (a specifed significance level), then the feature is deemed to be actually important and is included in the final output of the algorithm. This method also allows for some features which are debatable as to whether they are relevant or not to be included in the final output, should the analyst decide to do so. Given that many repeated trials are being performed, I also adjusted the p-value with a Bonferroni correction. Finally, the model ranks the features and presents those which are more important for the prediction of our response variable.

2.4 Self Organizing Maps

A self-organizing map (SOM) is a type of neural network that is used as a unsupervised-learning technique. It begins by taking a vector of the features for all the data points and then from there it groups them together according to how similar those vectors are. The observations are then mapped to a two-dimensional grid which allows us to visualize how ”close” the observations are to one another [Wil19].

The algorithm operates as follows:

1. Initial weights for each node. These weights can be initialized randomly or with PCA. I've chosen to randomly initialize them.

2. Randomly select a vector of the feature data containing the observations from the training set.

3. Find the best matching unit (BMU). The BMU calculates the distance from each weight to the sample vector by iterating through all weight vectors. The weight are most like the input vector are considered the winner (the best matching unit).

4. We adjust the weights of the cells in the neighborhood of the BMU. Distance from the cell to the BMU is calculated usually based on Euclidean distance. Cells whose distance to the BTU is lower, have their weights adjusted more. Cells which are further away from the BMU get their weights adjusted less.

5. Repeat step 2 N times or until the adjustments of the weights stops changing. Once the weights of cells are no longer being adjusted based on new BMUs selected in each iteration, the results have converged.

In applying a SOM to the task of feature selection, I used a function written by the creators of the MiniSom Python package [Ath20] which uses the methodology found in the paper A New Feature Selection Methodology for Environmental Modelling Support: The Case of Thessaloniki Air Quality [NK18]. This feature selection method works by generating a SOM for each of the features in the dataset, and then after identifying the target of our dataset (the triple barrier label) it then proceeds through a set of steps which is in some ways reminiscent of the Boruta algorithm presented above. These steps are as follows:

1. Create N-1 random features and join them to the real features. These randomized features are similar to the approach taken in the Boruta Algorithm.

2. Normalize all features, including the shadow features.

3. A given feature column is then compared to the target element by element to ascertain how many elements are similar.

4. Initialize a hyperparamter α which acts as a threshold. This is a key step and is somewhat of an arbitrary choice. I have set it as 0.07, slightly higher than recommended by the authors. This produced a selection of features I subjectively believe to be better (see results).

5. Compare the corresponding elements of the target and the specified feature.

$$|Target_j - Feature_j| \leq \alpha$$

6. Assign these values to a new matrix S which has the same number of columns as the original but where each element is either equal to 1, very close to the target parameter, or zero, indicating that the two features are complementary.

7. Next we rank the features based on the sum of their values in the matrix S and select the feature which has the highest sum.

8. We repeat through Step 5, until we reach the shadow features or the maximum number of 1's is reached.

I implemented this algorithm but probably wouldn't use it again. For starters, the randomness present in step 1 results in starkly different features being selected just by random chance alone. Between two runs of this algorithm, and using the same hyper parameter α, you could have both a completely different set of features, and a different number of features selected. The results are also highly dependent on the choice of α, as even a small change in this hyper parameter can produce drastically different results. α could likely be tuned, but it seems like it would be a wasted effort given the high degree of randomness inherent in the algorithm. Some of these issues might just be related to my dataset, and a different dataset might produce more stable results, but as for implementation in this project, I have more comfort in my other feature selection methods. Alternatively, a more qualitative review of self organizing feature map plots also seems more advisable. This selection algorithm might be reasonable in some cases, but it would need to be properly validated by the analyst and not taken at face value.

2.5 Principal Component Analysis

Simply put, Principal Components Analysis (PCA) is a method to find common movements in the data. They allow us to summarise a large number of features with a small set of variables that represent those features, and collectively explain the variability of the original dataset. For this reason PCA is used as a form of dimension reduction[GJ15]. Each of the principal components (PCs) are a linear combination of the features of the dataset. More precisely, take the 84 features used in the analysis which make up the set $X_1, X_2, ..., X_84$, the first principal component of this set is then the normalized linear combination of these features.

$$Z_1 = \phi_{1,1}X_1 + \phi_{2,1}X_2 + ... + \phi_{84,1}X_81$$

where $\sum_{i=1}^{84} \phi_{i,1}^2 = 1$ and the ϕ 's are called the loadings of the first principal component. These loadings are estimated via the maximization problem:

$$\max_{\{\phi_{1,1},...,\phi_{84,1}\}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{84} \phi_{j,1} x_{i,j} \right)^2 \right\} \quad s.t. \sum_{j=1}^{84} \phi_{j,1}^2 = 1$$

The first principal component described the direction of the features space which has the greatest variance, the second principal component the direction with the second most variance, and so on. By using a very large number of principal components, we would be able to basically explain all the variance observed in the feature set, but this would mean we have replaced our 84 easy to interpret features with 84 difficult to interpret principal components.

2.6 Long Short-Term Memory

Long Short-Term Memory models are a type of recurrent neural networks (RNN) that are better at learning long term dependencies in a time series. This makes this type of neural network useful when making classification of future events based on previous patterns, such as I am in this project. The LSTM is made up of memory cells, an input gate, an output gate, and a forget gate [Laz21]. Beginning with the input gate, it is responsible for taking in new information. Within this first gate the information is passed through usually a *sigmoid* or *tanh* (shown below respectively) activation function. These activation functions ensure that the information that is added is useful. This is given by the equation:

$$i_t = \sigma(W_i[h_{t-1}x_t] + b_f$$

$$\bar{c_t} = tanh(W_c[h_{t-1}x_t] + b_i$$

The memory cell is responsible for remembering previous states but the information it sees is controlled by the other gates.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \bar{c_t}$$

A forget gate removed the useless information about previous states. This acts as a filter to remove information that is less important.

$$f_t = \sigma(W_f \cdot [h_{t-1} x_t] + b_f$$

Lastly we have the output gate which regulates the final output values.

$$o_t = \sigma(W_o [h_{t-1} x_t] + b_o)$$

$$h_t = o_t \cdot thanh(C_t)$$

For the last layer of the LSTMs used in this project, a *softmax* activation function was used in the output gate so as to make a prediction about one of our three barriers. The exact design of the LSTMs is dependent on the dataset and will be presented in the next section.

3 Results

3.1 Exploratory Data Analysis

To begin the data exploration, the main features chosen for this project were analyzed. These were the closing prices of USDCAD, EURCAD, CADJPY, and all other commodities/indices which were chosen. The grid plot of these variables can be seen below over the full time frame of our analysis (January 2012 to June 2022).

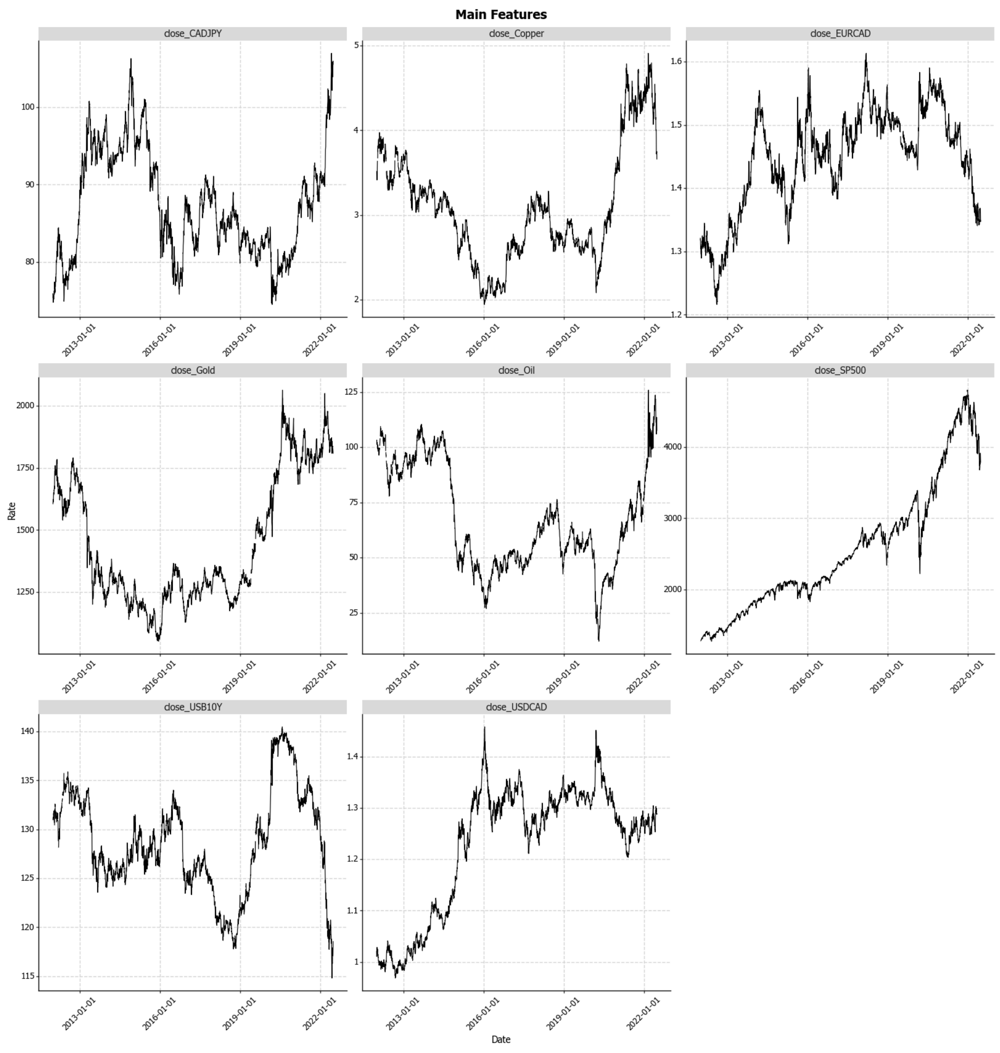


Figure 5: Time Series of Main Features

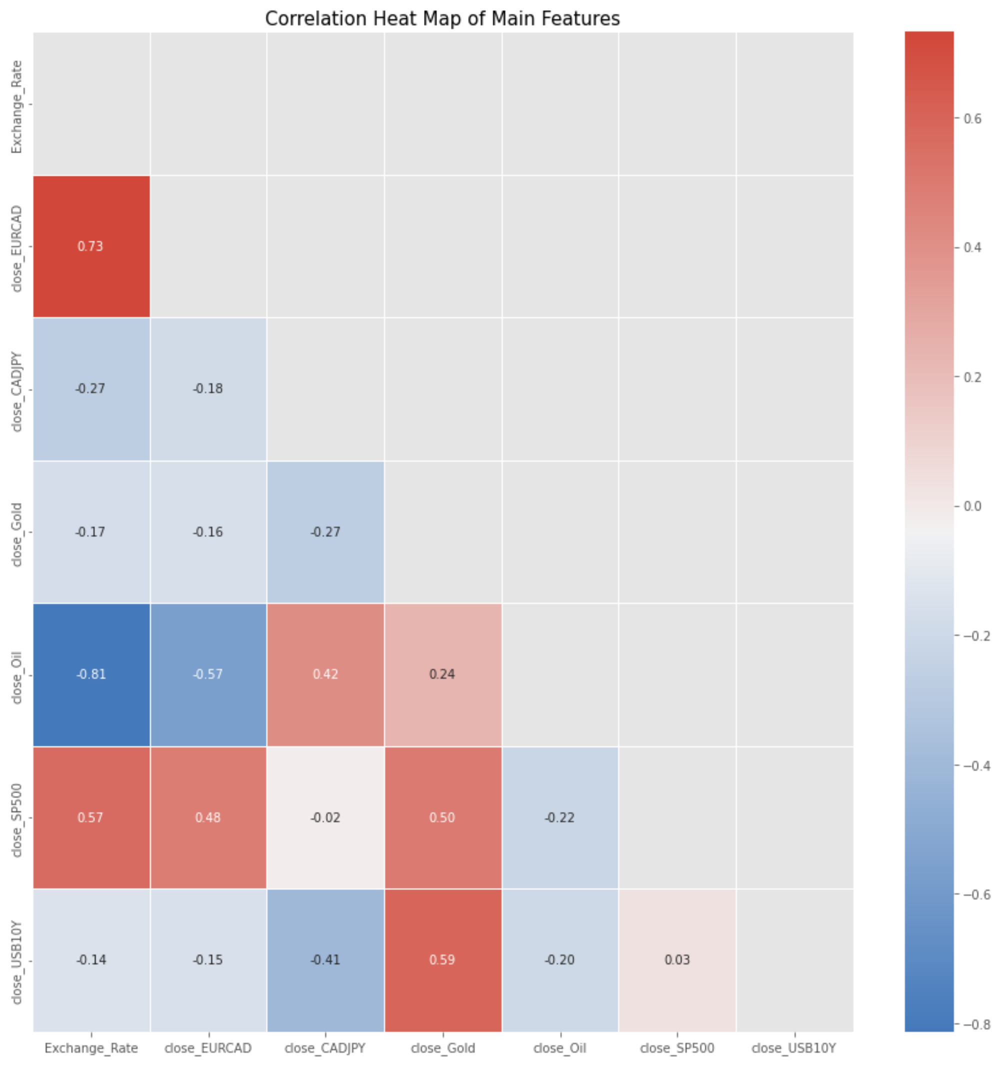


Figure 6: Main Features Correlation Heat Map

It seems clear that there is no long term relationship between USDCAD and some of the features included. This can also be seen in the correlation matrix of these same main features. USDCAD is most strongly correlated with EURCAD, most likely because any depreciation isolated to the Canadian side of the pair will result in a higher exchange rate across both pairs. Notably, the negative correlation between USDCAD and oils is quite strong as expected, in addition to there being a reasonable strong correlation between USDCAD and the S&P 500. It is worth noting that these correlations are not constant over time. For instance, historically the Canadian dollar was known to move with the price of oil and this can be seen up until around 2014. Following 2014, when we saw a large selloff in oil causing the price to collapse from $100 a barrel to around $50, much of the production operations in Canada scaled back. When oil prices recovered, it took longer to turn these fields back on, leading to a smaller amount of oil exports and a Canadian dollar that is less correlated with the price of oil. Another example can be seen in the S&P 500 and USDCAD. Usually we see the U.S. dollar rally when there is risk-off sentiment in the market, as we did at the beginning of the pandemic, but this has not been the case for the most recent selloff of in the equity markets. The loonie has been able to maintain its value against the U.S. dollar, largely due to other confounding factors such as inflation and interest rate differentials which have not been included in this analysis. Due to the changing correlations, the forget gate feature of LSTMs might be useful in creating a predictive model.

Aside from these other features, I also calculated several technical indicators. A subset of those indicators can be seen in the figure below in the grid of line plots, as well as the correlation matrix.

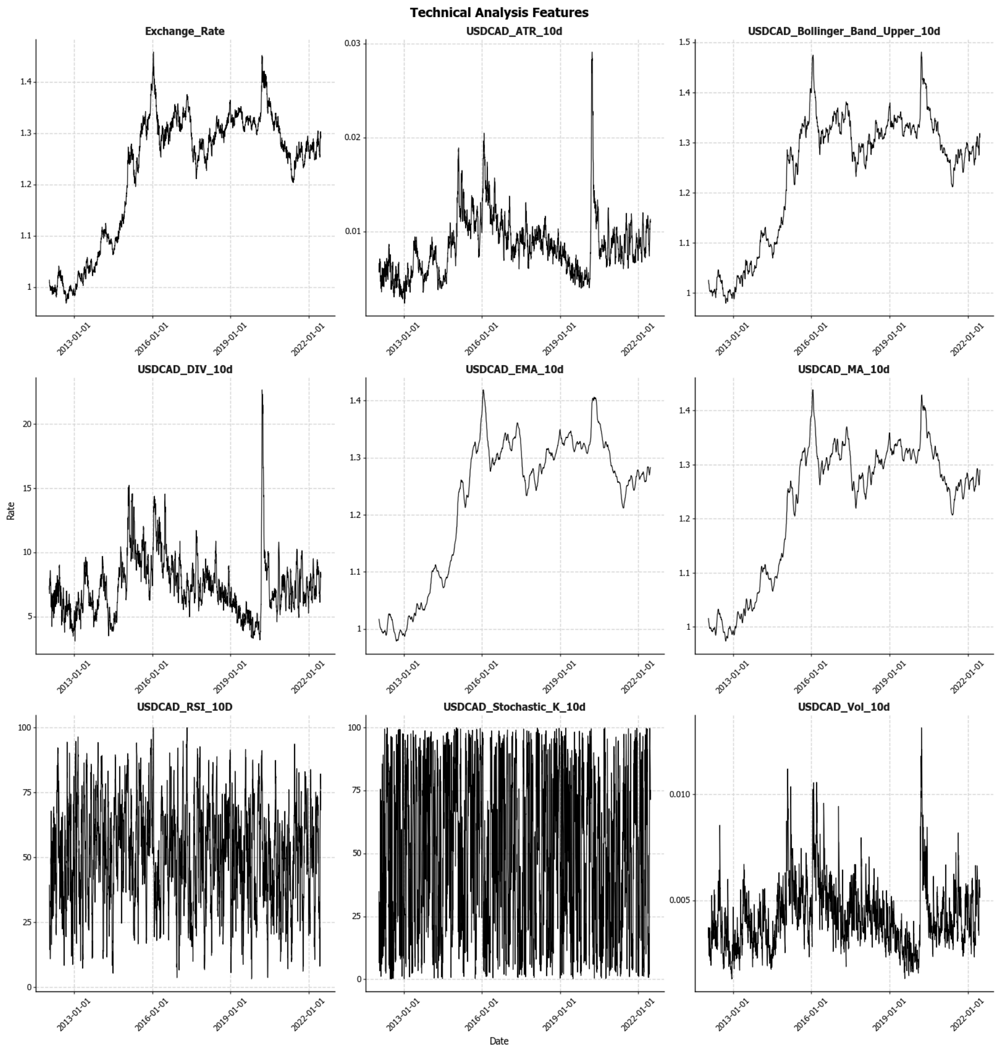


Figure 7: Grid Plot of Technical Analysis Features

From this plot, it can be plainly seen that several indicators will likely be strongly correlated with one another. For example, the Average True Range indicator and the Drift Independent volatility have very similar plots. Furthermore, keeping both the relative strength index and the stochastic K oscillator might be redundant. The plots presented here are only for the versions which were calculated over the last 10 days, but other lags were also investigated. Below is the correlation matrix for the same set of data, but also including the triple barrier label.

As expected, there is a strong positive correlation between the average true range and the drift independent volatility. Furthermore we can also see a strong negative correlation between the USDCAD exchange rate and the volatility as measured by an EWMA. This negative correlation disappears when corrected using drift independent volatility.

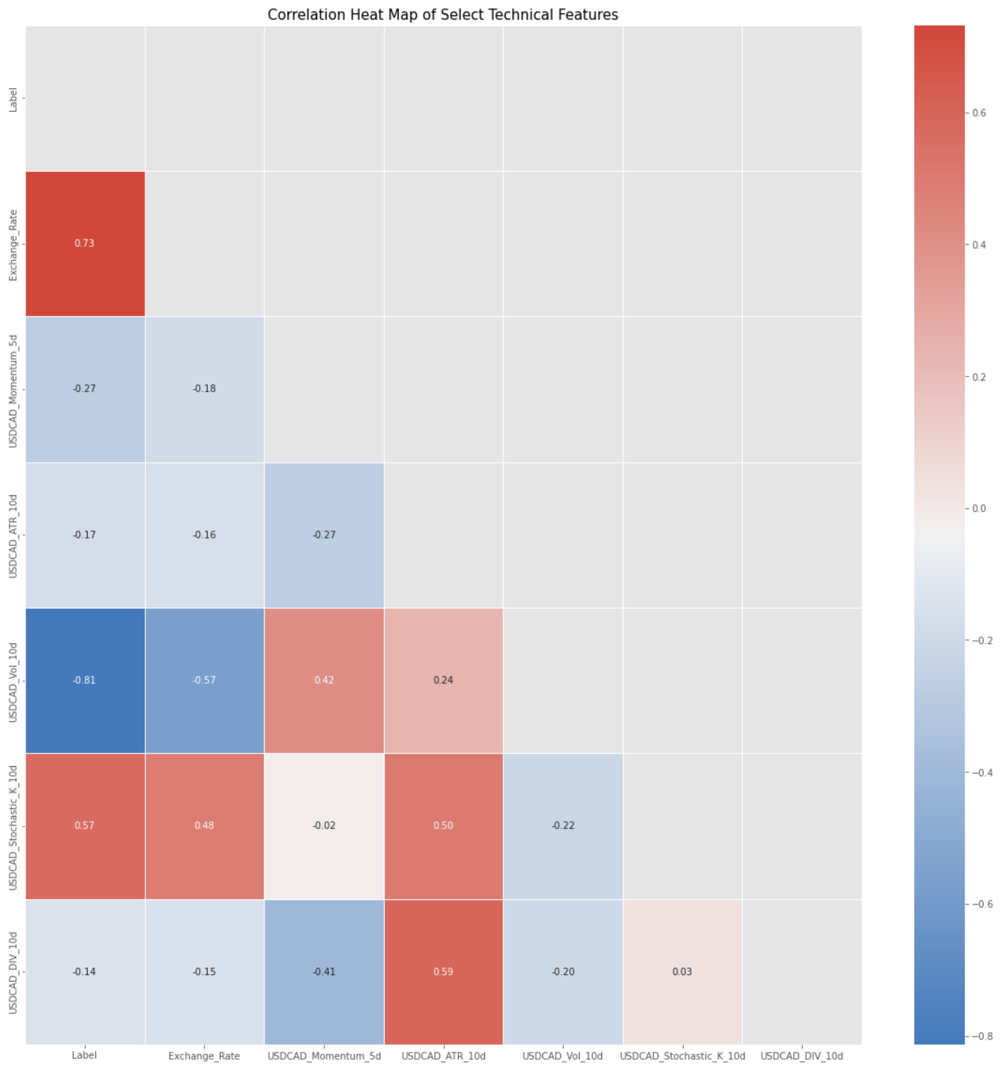


Figure 8: Technical Indicators Correlation Heat Map

3.2 Label Imbalance and Train-Test Split

A 70, 20, 10 splitting scheme was used to divide the dataset into the train, validation, and test sets respectively. When evaluating the primary model candidates, only the validation set was used and the test set was held in reserve to evaluate the performance of the secondary model based on meta labelling. At this step in the analysis, the class imbalance of the triple barrier labels were also investigated. This imbalance can be seen in the figure below.

As seen in this figure, there a slight class imbalance between the lower barrier (-1) and the upper barrier (+1), but there is a significant imbalance between both horizontal barriers and the vertical barrier. To remedy this, two approaches were investigated. The first, was using the synthetic minority oversampling technique (SMOTE) to over sample the lower barrier and the vertical barrier classes. This was chosen over under-sampling the upper barrier class because deep learning models like the LSTM are known to be more data hungry, and we already had a limited number of observations on a large number of features. The second technique investigated was directly adjusting the class weights during model fitting. This should have produced similar results as the SMOTE method but due to complications of applying those class weights during the hyper parameter tuning with certain packages, I opted instead to perform this correction before working on the LSTM.

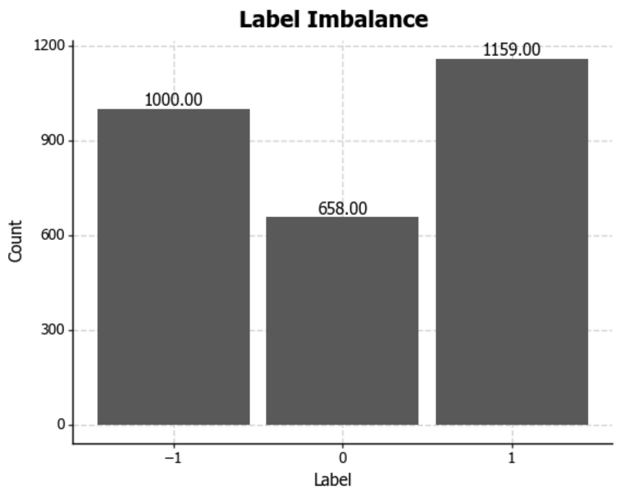


Figure 9: Class Imbalance of Triple Barrier Label

The SMOTE correction was only applied to the training set, leaving the imbalance as is in the validation and test sets of data. Before moving on to perform feature selection, copies of the training set, validation set, and test set were made where scaling and normalizing was done. The scaling and normalizing was performed individually on each set after the splits had been made so as to ensure that there wouldn't be any data leakage. Lastly, the label needed to be re-encoded as 0, 1, 2 before many of the algorithms I would be using would work. The lower barrier was reclassified as 0, the vertical barrier as 1, and the upper barrier as 2.

The full correlation matrix of all features used in this project can be found at the end of this report, although it can be difficult to read the labels. A more clean image can be seen in the accompanying notebook.

3.3 Feature Selection

3.3.1 Boruta Algorithm Results

The features selected by the Boruta algorithm can be found in a table in the Appendix. Only variables which were ranked one were selected and all remaining features which ranked between 2 and 21 were dropped. The variables which ranked 2 were closer to the top of the list on the right hand column while the variables which ranked closer to 21 are near the bottom. The rank is simply a grouping of variables based on how far away they were from the confirmation region. The confirmation region was specified at a 0.01 level. In total, 54 features were kept and the remaining were removed.

3.3.2 Self Organizing Map Results

The first self organizing map was fitted with a learn rate of 0.5 and a sigma of 1. As it turns out these were very good guesses from the start and it was difficult to find a lower quantization error when tuning these hyperparameters. The first feature map with some selected features can be seen in the figure below.

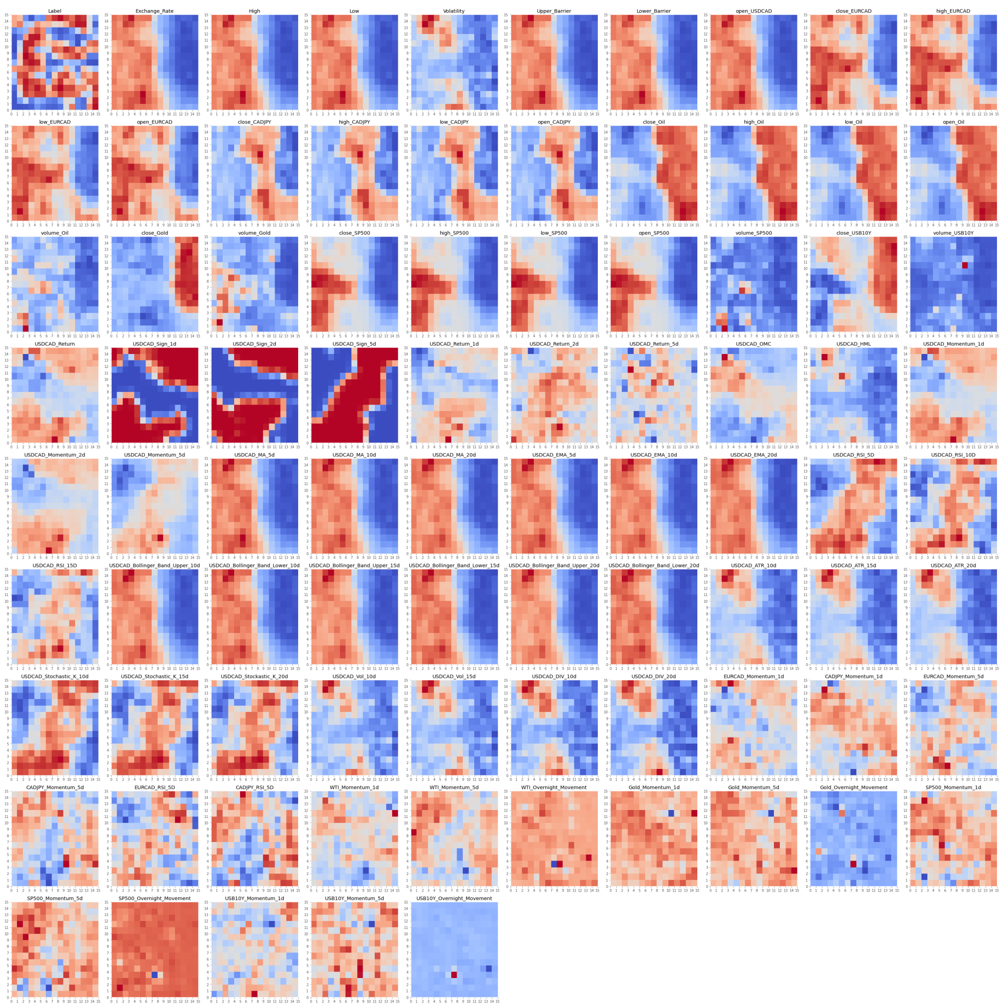


Figure 10: Self Organizing Feature Map

Even without tuning the SOM's hyperparameters, we can get some good separation in some features. Specifically we can see that the sign of the returns over one day for USDCAD has two very dark regions, a stark separation. Furthermore several panels look almost identical. All the OHLC data for USDCAD is almost identical, and it also shares a lot of similarities with the moving averages and the Bollinger bands. All the lags look essentially the same, giving no clear indication which lag would be the best from a visual inspection alone.

Following this first inspection, I tried tuning the SOMs hyper parameters (learning rate, and sigma) to see if the features that didn't see much seperation could be improved. The results of this work are below.

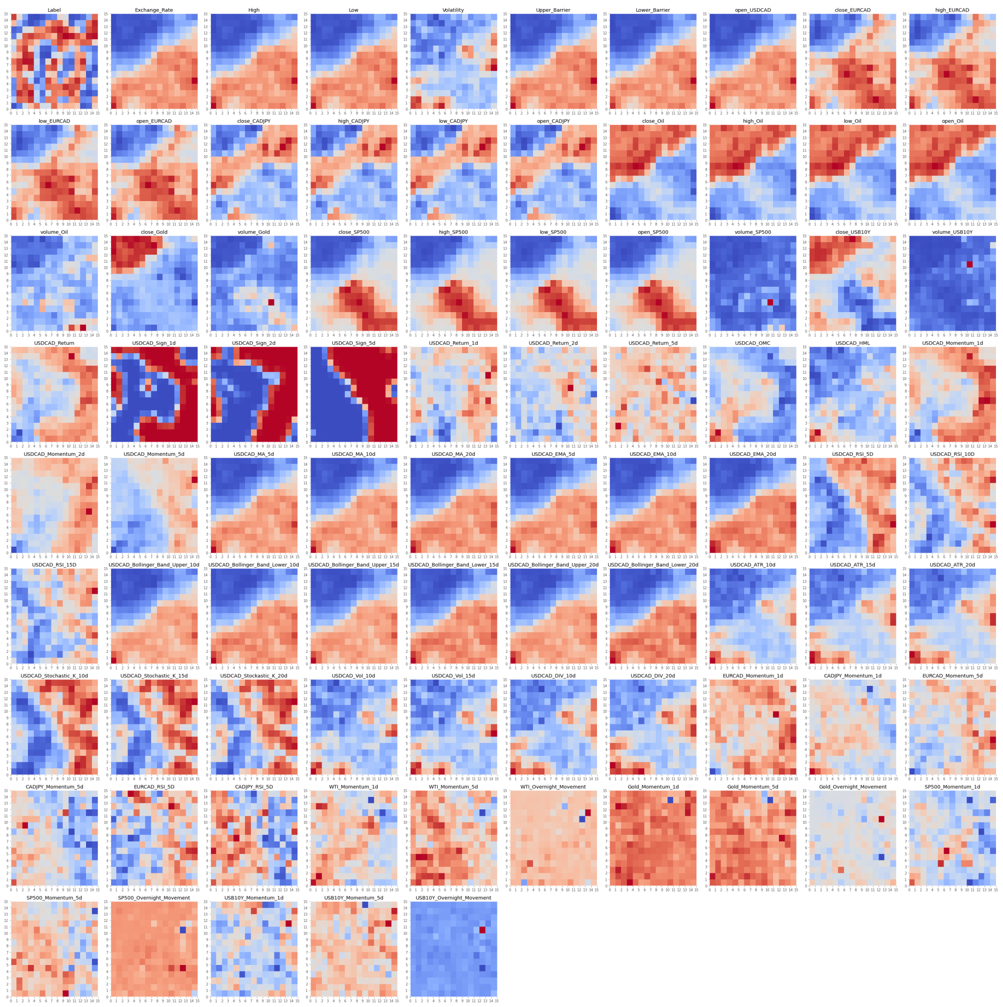


Figure 11: Optimized Self Organizing Feature Map

By tuning the hyper parameters I didn't achieve a significant improvement in my result with only a minor reduction in the quantization error. The effect this had on the self organizing map feature plots was also trivial. Some of the graphs now seemed to be rotated but the separations appear largely the same. Using the SOM feature selection algorithm outlined in the methodology section, I was able to reduce the 84 features down to 9. These final features being RSI for USDCAD over the last five days, change in WTI between yesterdays close and the open, the low of the day for USDCAD, RSI for USDCAD over the last fifteen days, change in the price of the U.S. government 10 year between yesterdays close and the open, the sign on the one day return of USDCAD, the closing price of WTI, and the volume trading for the S&P 500.

Lastly, it is important to note I chose a slightly higher hyper parameter α than is recommended in [NK18]. I chose this higher α because I was only receiving one or two features out of the algorithm otherwise, and they also typically weren't features that the Boruta algorithm would have considered important. Due to the conflicting selection of features being produced by this algorithm, I decided it would be more prudent to raise the threshold of the algorithm to include feature I had more confidence in.

3.3.3 Principal Component Analysis Results

Initially, 50 principal components were fit to the full feature set to determine how much of the total variance each of these PCs explained. This can be visually seen in the figure below.

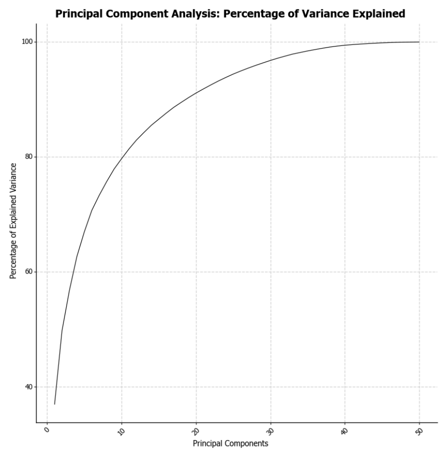


Figure 12: Principal Component Analysis

It took approximately 40 principal components to explain 99% of the total variance, and while this would reduce the number of features being entered into the LSTM from 84 down to 40, it seemed more prudent to reduce this number further. Alternatively, 20 principal components could explain 90% of the total variance, again cutting the number of features being entered into the model by half. Explaining 90% of the variance and including only 20 variables seemed like a better trade off to come from this analysis. This can be further confirmed from the Scree plot below, where after 20 PCs the marginal variance explained is very small.

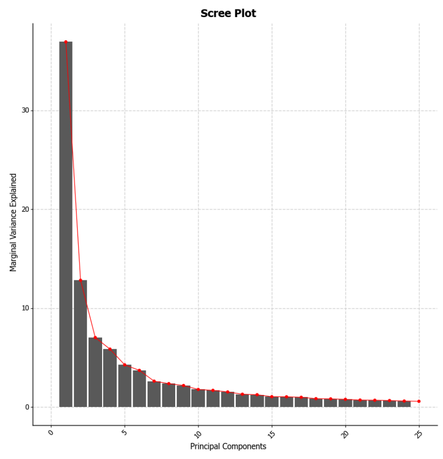


Figure 13: Scree Plot of Principal Components

3.4 Long Short-Term Memory Models (Primary Models)

Three LSTM models were built using the data from the three feature selection methods. A large parameter space was attempted on some of these LSTMs, but unfortunately due to computational constraints, these parameter spaces ended up being limited. Several other runs which did not make it into the final notebook were also performed but the results often took too long to run so the parameter spaced needed to be reduced to what is presented in the notebook. The hyper parameters adjusted in each LSTM included:

- Learning Rate: $[0.001, 0.01]$
- Dropout Rate: $[0.2, 0.4]$
- Units: $[32, 64, 128]$
- Batch Size: 50
- Epochs: $[50, 100]$
- Hidden Layers: 1
- Optimizer: *Adam*
- Activation: *Tanh*

I had wanted to include more hidden layers but unfortunately the runtime exploded to several hours as soon as I included a second hidden layer. It is likely the predictions from these LSTMs could be improved given more time to train.

3.4.1 LSTM with Boruta Selected Features

The best estimator found for the LSTM built with the Boruta selected features had parameters with the values:

- Learning Rate: 0.001
- Dropout Rate: 0.4
- Units: 64
- Batch Size: 50
- Epochs: 50

The classification report for this LSTM is in the table below.

Table 1: **Classification Report for LSTM with Boruta Selected Features**

	Precision	Recall	F1-Score	Support
0 (Lower Barrier)	0.39	0.60	0.48	208
1 (Vertical Barrier)	0.21	0.07	0.10	147
2 (Upper Barrier)	0.37	0.36	0.37	188
Accuracy			0.37	543
Macro Average	0.33	0.34	0.32	543
Weighted Average	0.34	0.37	0.34	543

This classifier had a low accuracy, and while it did a decent job of correctly predicting that the lower barrier would be triggered, it did an abysmal job predicting the vertical barrier. Furthermore, the precision of predictions for the lower barrier was also fairly low, indicating it would over predict the class and produce

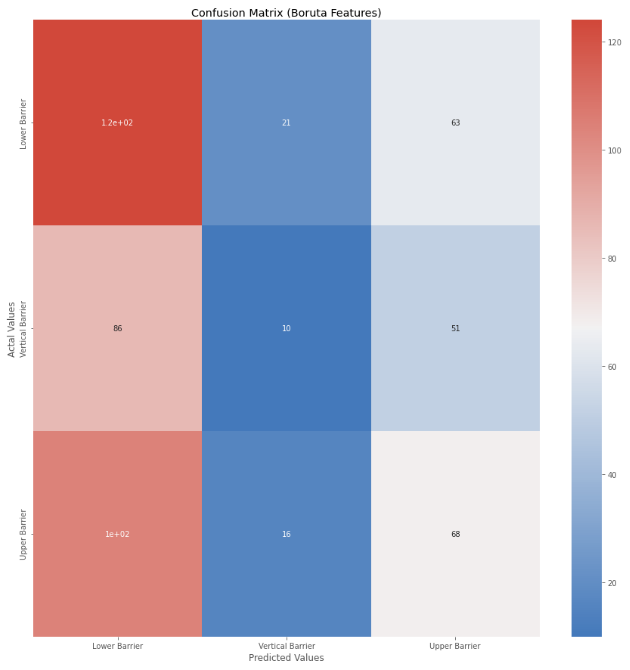


Figure 14: Boruta Confusion Matrix

many false positives. While not as poor as the predictions for the vertical barrier, it also had a relatively low recall for the upper barrier. The confusion matrix confirms these findings from the classification report.

3.4.2 LSTM with Self Organizing Map Selected Features

The best estimator found for the LSTM built with the SOM selected features had parameters with values that match the Boruta LSTM.

- Learning Rate: 0.001
- Dropout Rate: 0.4
- Units: 64
- Batch Size: 50
- Epochs: 50

The classification report for the LSTM with SOM selected features is in the table below.

Table 2: **Classification Report for LSTM with SOM Selected Features**

	Precision	Recall	F1-Score	Support
0 (Lower Barrier)	0.00	0.00	0.00	208
1 (Vertical Barrier)	0.00	0.00	0.00	147
2 (Upper Barrier)	0.35	1.00	0.51	188
Accuracy			0.35	543
Macro Average	0.12	0.33	0.17	543
Weighted Average	0.12	0.35	0.18	543

This classifier was completely overfit, and predicted only one class even though it used the same underlying label data that the other LSTMs were built on. The classes were equal within the data set after using SMOTE, yet this LSTM still only predicted that the lower barrier would be breached. It is likely something within the settings or data is wrong for this to occur, but after checking, the same results were again reproduced. This LSTM was not considered when determining which would be most appropriate to use for the final primary model.

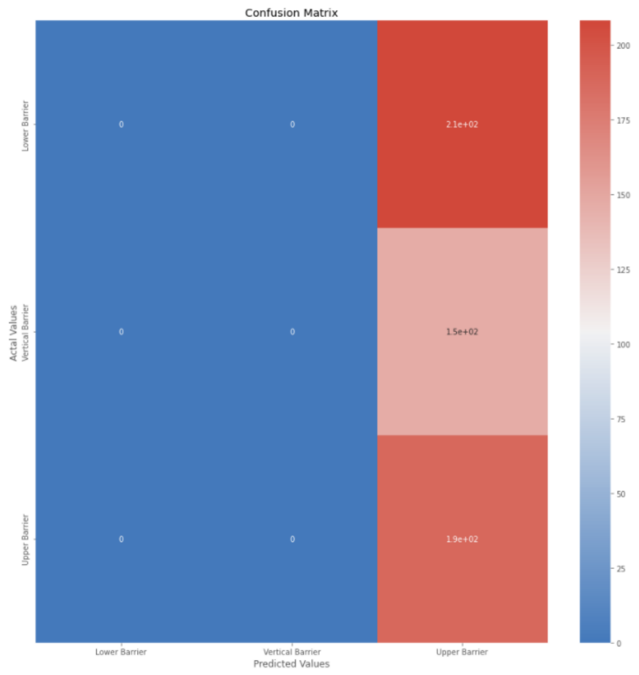


Figure 15: SOM Confusion Matrix

3.4.3 LSTM with PCA Dimension Reduction of Features

The best estimator found for the LSTM built on the principal components of the original feature set was again similar to the other two LSTMs. The only difference was the higher learning rate used for this model.

- Learning Rate: 0.01
- Dropout Rate: 0.4
- Units: 64
- Batch Size: 50
- Epochs: 50

The classification report for the LSTM with principal component features is in the table below.

Table 3: **Classification Report for LSTM with PC Data**

	Precision	Recall	F1-Score	Support
0 (Lower Barrier)	0.39	0.36	0.37	208
1 (Vertical Barrier)	0.24	0.23	0.24	147
2 (Upper Barrier)	0.41	0.46	0.44	188
Accuracy			0.36	543
Macro Average	0.35	0.33	0.35	543
Weighted Average	0.36	0.36	0.36	543

This LSTM model had predictions which were more similar to those of the Boruta LSTM. While all the recalls are still relatively poor for these predictions, they weren't as drastically different as those which came out of the Boruta LSTM. The recall for the upper barrier is reasonable in this model, but the recall for the lower barrier and vertical barrier are still poor. Furthermore, the predictions still had poor precision, indicating that numerous false positives are being predicted. This is confirmed in the confusion matrix below.

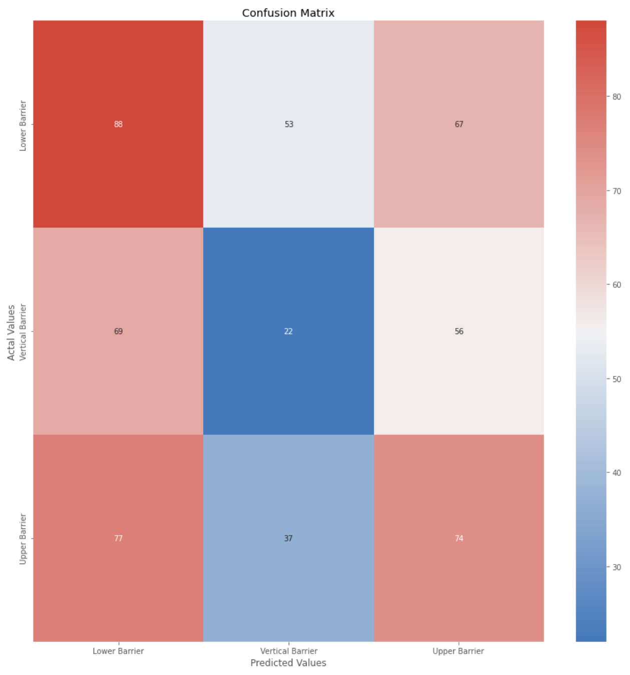


Figure 16: PCA LSTM Confusion Matrix

Given that the performance of the Boruta LSTM and PCA LSTM were similar, it doesn't make much of a difference which is chosen. As the original strategy was to choose a model which had a high recall and low precision, expecting the secondary model to correct for that low precision, I have chosen to use the Boruta feature set in the secondary model due to it having a slightly higher average recall.

3.5 Long Short-Term Memory Model (Secondary)

3.5.1 Secondary Model Metrics and Analysis

Up until this point, the test set still had not been used to validate any of the results. It had been held in reserve to prevent the possibility of anything being leaked when building the primary models. The test set will now be used to validate the results of this secondary model built with the features selected from the Boruta algorithm, and the predictions from the LSTM model built on that same feature set. After constructing the meta label, I analyzed the class imbalance and found that class 0 was under represented. This can be seen in the figure below.

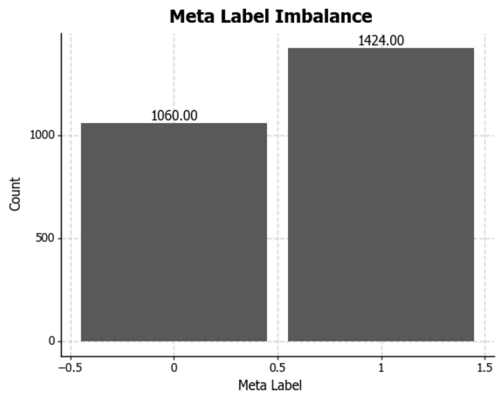


Figure 17: Meta-Label Class Imbalance

To remedy this, I calculated class weights which could be added when fitting the secondary LSTM. I avoided using the SMOTE method again as I wanted to eliminate as many sources of potential date leaks as possible. While the method may have been fine to use, I decided to go with what seemed like a safer alternative. The confusion matrix for the predictions from the secondary model is presented below.

Table 4: **Classification Report for Secondary Model**

	Precision	Recall	F1-Score	Support
0 (Don't Trade)	0.40	0.13	0.20	145
1 (Trade)	0.34	0.69	0.46	95
Accuracy			0.35	240
Macro Average	0.37	0.41	0.33	240
Weighted Average	0.38	0.35	0.30	240

As seen in the classification report above, there are still clear deficiencies in the secondary model. Notably, the model tends to trade more than it should. While the recall of decisions to trade was very good, it had a poor precision. This indicates that the model is over predicting that class and producing many false positives. The recall of the other class, not to trade, was very low. It missed most of the situations where it would have been better not to trade. Overall the F1 scores for both these metrics are very low, indicating the poor efficiency of the predictions. These findings are confirmed by the confusion matrix as seen below.

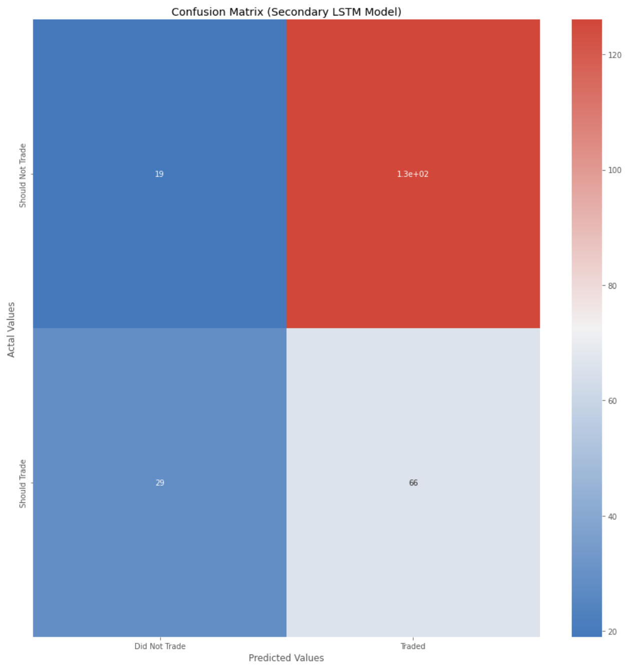


Figure 18: Secondary Model Confusion Matrix

3.5.2 Risk and Return Analysis

To conclude this report I have performed a basic analysis on the returns of the two trading strategies, the first being based only on the primary model, and the second being based on the combination of both the primary and the secondary model. This analysis was performed on the test set which covered the time frame of between the end of July 2021, and the middle of June 2022. While this length of time might be a reasonable size, it covers a very unique period in financial markets history with the pandemic, soaring inflation, and a war in Europe. Given all of these significant events, it would be prudent to conduct a more thorough cross validation and backtesting than what was performed in this project, but unfortunately time was not permitting. To begin, the cumulative returns of the two trading strategies were calculated and are presented below.

Here we can see that the primary model would have achieved a higher return had it been allowed to run on its own. The two models were almost in sync up until 2022 when the secondary model began to predict do not trade signals more regularly. Notably, there is a period from March 2022 to May 2022 when it predicted a string of do not trade signals. Because of this, the two cumulative return series began to deviate from one another. Eventually, the secondary model began predicting signals to trade again. At that time we began to see the cumulative returns rise, before we began to make several losing trades at the beginning of June. Had the secondary model correctly identified these as times not to trade, then we would have seen its cumulative returns not dip in the same way we see the primary model's returns, but unfortunately the secondary model was only predicting trade signals at this time.

Other portfolio metrics were also calculated for both strategies and are presented in the table below. As expected, the risk of the secondary model which combines the predictions of the first had a lower level of risk than the primary model. Unfortunately, the risk adjusted return for the primary model was greater than that of the secondary model. This can be seen in the Sharpe and Sortino ratios. While the secondary model produced a less risky trading strategy, it also resulted in a significantly less profitable trading strategy. The two strategies also had the same max drawdown, with the secondary model unable to prevent the worst losing streak which occurred in the primary model. Regardless, the max drawdown of either strategy is reasonable. Plots summarizing this information can also be found below.

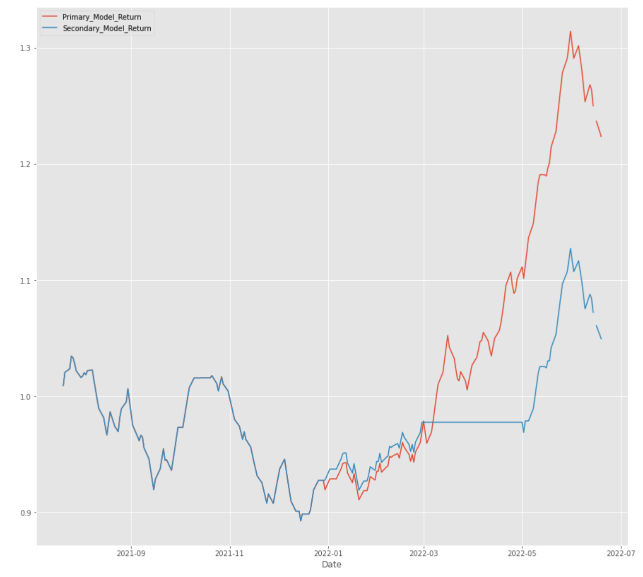


Figure 19: Cumulative Returns of Both Strategies

Table 5: **Model Risk and Return Summary**

	Primary Model	Secondary Model
Total Moths	11	11
Annual Return	22.7%	4.6%
Cumulative Return	21.5%	4.4%
Annual Volatility	12.5%	10.6%
Sharpe Ratio	1.76	0.48
Calmar Ratio	1.63	0.33
Stability	0.34	0.07
Max Drawdown	-14.0%	-14.0%
Omega Ratio	1.28	1.08
Sortino Ratio	2.70	0.71
Tail Ratio	1.05	1.04
Daily VaR	-1.4%	-1.3%

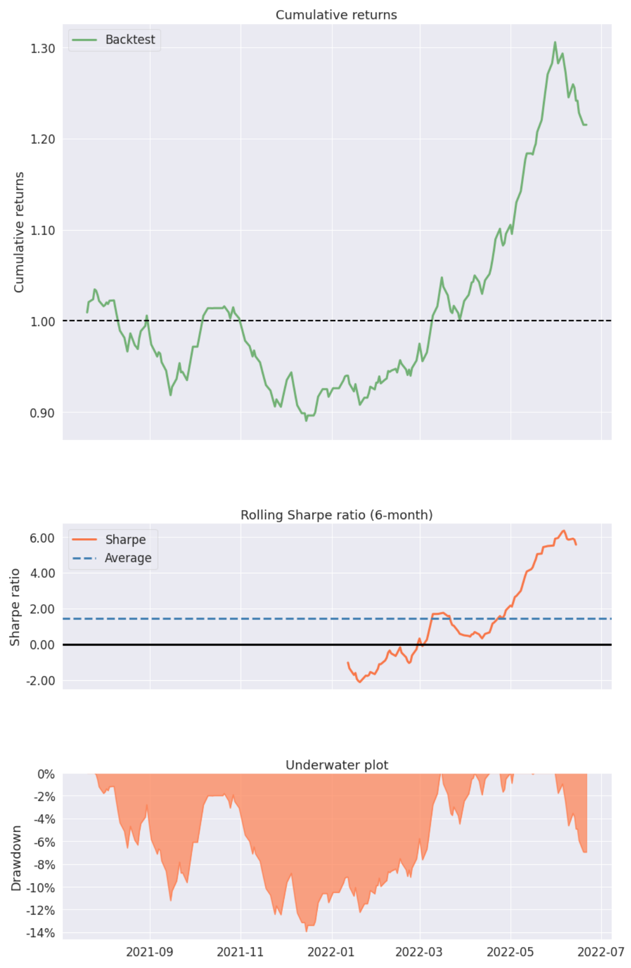


Figure 20: Risk-Return Plots for Primary Model

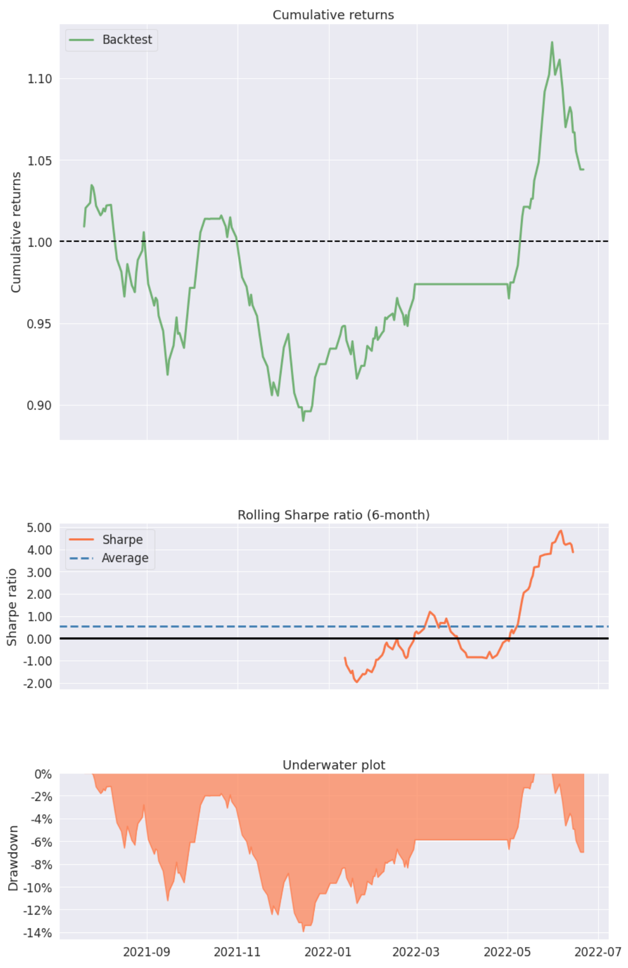


Figure 21: Risk-Return Plots for Secondary Model

4 Conclusion

4.1 Future Considerations

There are several refinements that I think should be made in a future version of this analysis. First, the LSTM's had poor predictive power. It would be unreasonable to expect the LSTM to overcome the general lack of signal within the data, but modest improvements should be possible. Secondly, it would likely be more prudent to use a different type of data such as tick or dollar bars. Tick level data might have better properties which the memory features of an LSTM model could use more effectively. Finally, it would also likely be beneficial to use alternative data sources such as web searches or news headlines. If this were to be integrated with the tick level data, then the feature set might be far more predictive than what was used in this project.

Aside from the model and features, a third area for improvement would be in the backtesting procedures. The validation and testing performed in this report was too basic to be reliable. A much more thorough a cross-validation backtest, such as the one proposed by Marcos Lopez de Prado in Advances in Financial Machine Learning [dP18] would be preferable. He proposes a backtesting procedure involving a purge of the beginning of the test set to ensure no data is being leaked,and a combinatorial based splitting procedure. Unfortunately due to time constraints, these features were not able to be implemented in this project, but it would be advisable to implement a more robust system such as this before even moving on to the paper trading.

4.2 Summary

In this project, I analyzed a decade of daily OHLC FX data and constructed numerous features based on both technical analysis indicators, and economic theory. From this large feature set, I identified the most promising features using three distinct methods. The Boruta algorithm, a feature selection algorithm based on self organizing maps, and principal component analysis was used to reduce the total number of features in the original set down to what was used to train LSTM models. I produced three LSTMs which were able to make predictions on whether the USDCAD exchange rate would break one of three barriers. Through the triple barrier labelling scheme, I was able to construct a trading algorithm which would behave in a fashion that would be more reliable in a professional environment than a more simplistic labelling scheme. Furthermore, the labels were improved again by constructing a meta-labelling scheme, wherein the predictions from the first LSTM model are fed into a secondary model as an additional feature. The results across all these models were presented and discussed, and then followed up with a analysis of the risk and return of both trading strategies. The final findings indicate that the secondary model actually made the performance of the trading strategy worse, and it would have been better to only use the predictions from the primary model. While these results are not what I had expected, I also believe both of the underlying models should be refined more before giving up on meta labelling. While the annualized returns from the primary model are impressive, final judgement should be deferred until a more robust backtesting strategy is implemented.

References

[Ath20] Evangelos Athanasakis. Feature selection based on self organizing maps. https://github.com/JustGlowing/minisom/blob/master/examples/FeatureSelection.ipynb, 2020.

[dP18] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley, New Jersey, 2018.

[DY00] Qiang Zhang Dennis Yang. Drift-independent volatility estimation based on high, low, open, and close prices. *Journal of Business*, 73(3), 2000.

[Fer] Jason Fernando. ”relative strength index (rsi)”. https://www.investopedia.com/terms/r/rsi.asp#:~:text=The%20relative%20strength%20index%20(RSI)%20is%20a%20momentum%20indicator%20used,the%20price%20of%20that%20security.&text=The%20RSI%20can%20do%20more%20than%20point%20to%20overbought%20and%20oversold%20securities.htm. Accessed: 10.08.2022.

[GJ15] Trevor Hastie Robert Tibshirani Gareth James, Daniela Witten. *An Introduction to Statistical Learning with Applications in R*. Springer, New York, 2015.

[Haya] Adam Hayes. ”average true range (atr)”. https://www.investopedia.com/terms/a/atr.asp. Accessed: 10.08.2022.

[Hayb] Adam Hayes. ”bollinger band”. https://www.https://www.investopedia.com/terms/b/bollingerbands.asp. Accessed: 10.08.2022.

[Hayc] Adam Hayes. ”stochastic oscillator”. https://www.https://www.https://www.investopedia.com/terms/s/stochasticoscillator.asp. Accessed: 10.08.2022.

[Laz21] Francesca Lazzeri. *Machine Learning for Time Series Forecasting with Python*. Wiley, 2021.

[MBK10] Witold R. Rudnicki Miron B. Kursa. Feature selection with the boruta package. *Journal of Statistical Software*, 36(11), 2010.

[NK18] Kostas Karatzas Nikos Katsifarakis. A new feature selection methodology for environmental modelling support: The case of thessaloniki air quality. *International Symposium on Environmental Software Systems*, 2018.

[Tan16] Adrian Tang. Compute yang zhang volatility given ohlc data. https://gist.github.com/tangabc/d9700366bd75ce49b1f4ea7a1a1030bf, 2016.

[Wil19] Paul Wilmott. *Machine Learning - An Applied Mathematics Introduction*. Panda Ohana Publishing, 2019.

A Appendix

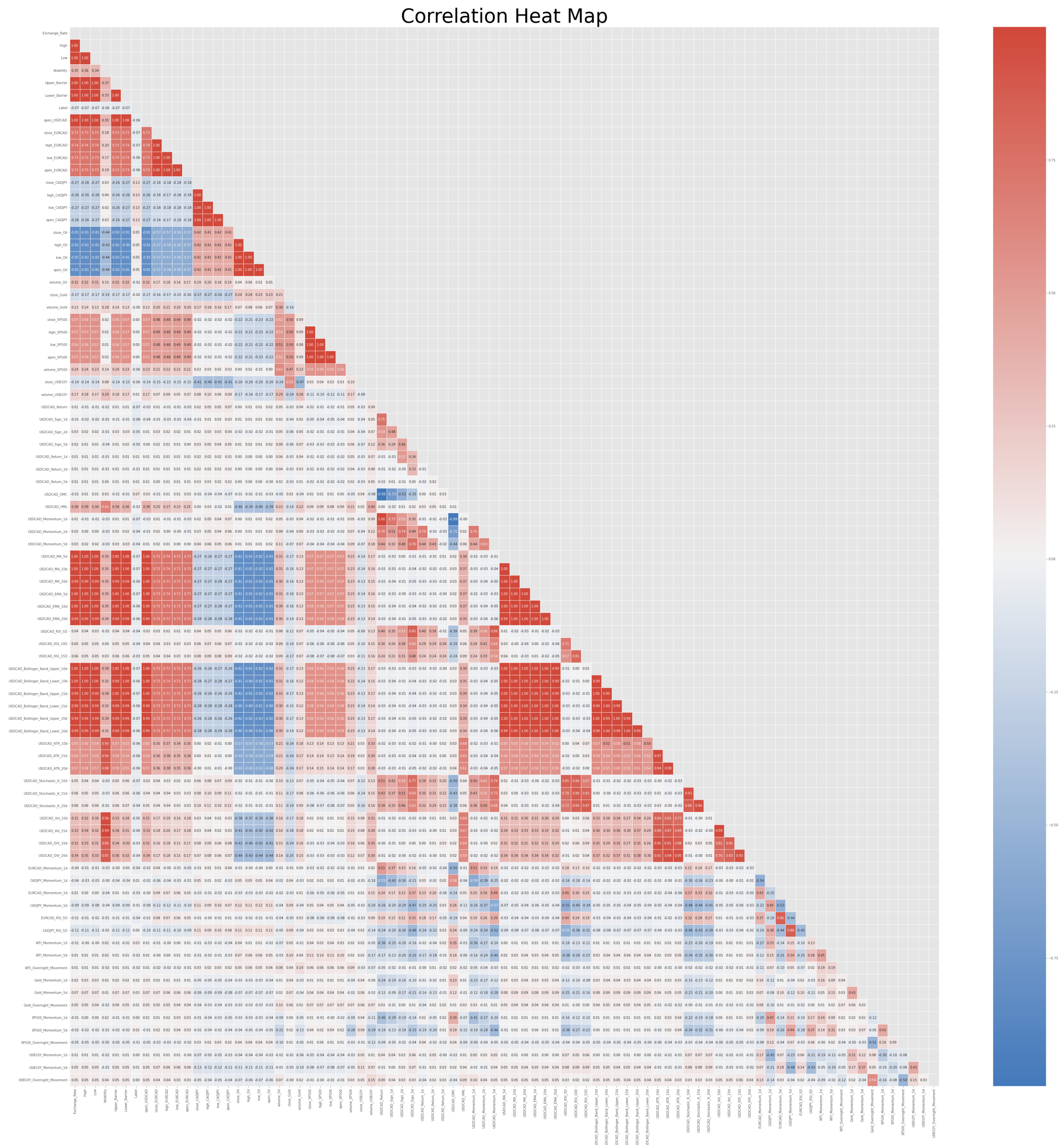
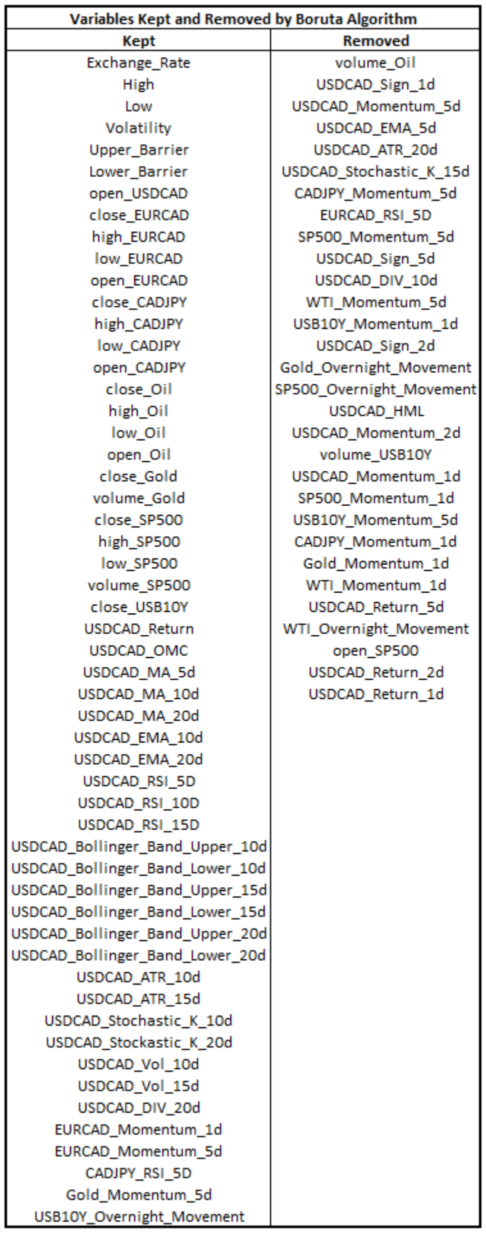


Figure 22: Correlation Heat Map of All Features



Variables Kept and Removed by Boruta Algorithm	
Kept	**Removed**
Exchange_Rate	volume_Oil
High	USDCAD_Sign_1d
Low	USDCAD_Momentum_5d
Volatility	USDCAD_EMA_5d
Upper_Barrier	USDCAD_ATR_20d
Lower_Barrier	USDCAD_Stochastic_K_15d
open_USDCAD	CADJPY_Momentum_5d
close_EURCAD	EURCAD_RSI_5D
high_EURCAD	SP500_Momentum_5d
low_EURCAD	USDCAD_Sign_5d
open_EURCAD	USDCAD_DIV_10d
close_CADJPY	WTI_Momentum_5d
high_CADJPY	USB10Y_Momentum_1d
low_CADJPY	USDCAD_Sign_2d
open_CADJPY	Gold_Overnight_Movement
close_Oil	SP500_Overnight_Movement
high_Oil	USDCAD_HML
low_Oil	USDCAD_Momentum_2d
open_Oil	volume_USB10Y
close_Gold	USDCAD_Momentum_1d
volume_Gold	SP500_Momentum_1d
close_SP500	USB10Y_Momentum_5d
high_SP500	CADJPY_Momentum_1d
low_SP500	Gold_Momentum_1d
volume_SP500	WTI_Momentum_1d
close_USB10Y	USDCAD_Return_5d
USDCAD_Return	WTI_Overnight_Movement
USDCAD_OMC	open_SP500
USDCAD_MA_5d	USDCAD_Return_2d
USDCAD_MA_10d	USDCAD_Return_1d
USDCAD_MA_20d	
USDCAD_EMA_10d	
USDCAD_EMA_20d	
USDCAD_RSI_5D	
USDCAD_RSI_10D	
USDCAD_RSI_15D	
USDCAD_Bollinger_Band_Upper_10d	
USDCAD_Bollinger_Band_Lower_10d	
USDCAD_Bollinger_Band_Upper_15d	
USDCAD_Bollinger_Band_Lower_15d	
USDCAD_Bollinger_Band_Upper_20d	
USDCAD_Bollinger_Band_Lower_20d	
USDCAD_ATR_10d	
USDCAD_ATR_15d	
USDCAD_Stochastic_K_10d	
USDCAD_Stockastic_K_20d	
USDCAD_Vol_10d	
USDCAD_Vol_15d	
USDCAD_DIV_20d	
EURCAD_Momentum_1d	
EURCAD_Momentum_5d	
CADJPY_RSI_5D	
Gold_Momentum_5d	
USB10Y_Overnight_Movement	

Figure 23: Boruta Algorithm Results