NUS FINTECH SOCIETY

AUD-USD STACKED ENSEMBLE MULTI-STRATEGY

Recurrent Neural Nets | Random Forest | Boosting | Bagging

Authors

Kaustubh Jagtap | Nicklaus Ong | Sung Zheng Jie

NUS FINTECH SOCIETY
nusfintech@gmail.com

Table of Contents

Initialize Algorithm

Our classification problem is phrased as follows:

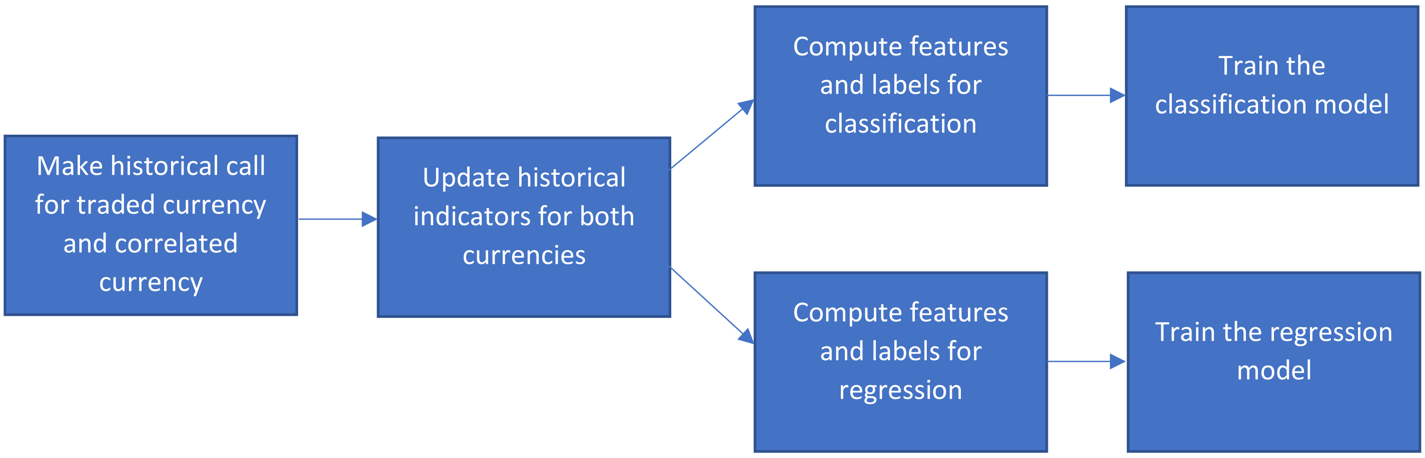
> "Given the relevant features from the market, can we predict whether price will increase (labelled as 1) or decrease (labelled as 0) in the following time interval?"

Our regression problem is phrased as follows:

> "Given the relevant features from the market, can we predict the actual price for the following time interval?"

Our strategy then combines the results of both these models, adds on more rules using extra technical indicators, and then places a trade.

The overall flow of the Initialize algorithm, therefore looks like this:

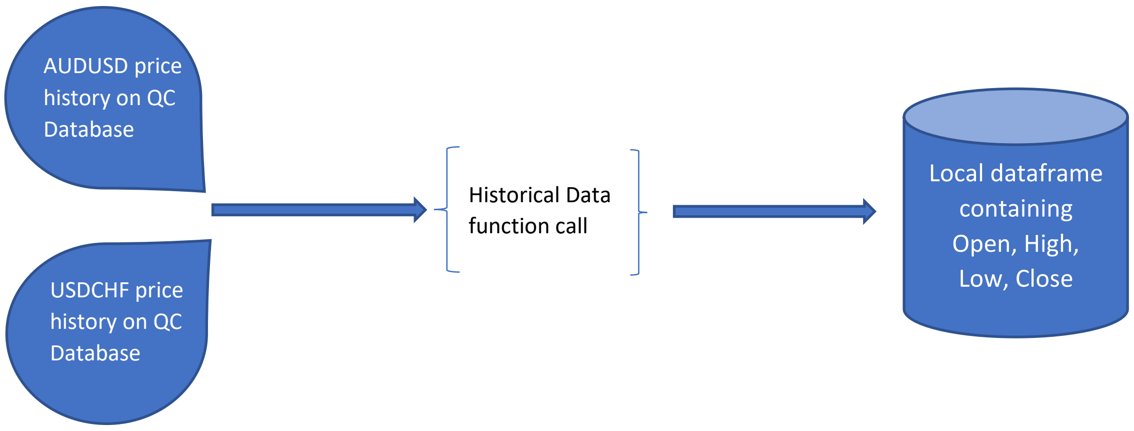

Make historical call for traded currency and correlated currency
Update historical indicators for both currencies
Compute features and labels for classification
Train the classification model
Compute features and labels for regression
Train the regression model

Data Processing

Our data processing methodology revolves around forming 2 dataframes and their corresponding labels – 1 for the classification and 1 for regression.

<u>Making the historical call:</u>

This involves 2 Quantconnect function calls for the 150-day History of AUDUSD (traded currency) and USDCHF (correlated currency). A visualization of this is given below:


AUDUSD price history on QC Database
USDCHF price history on QC Database
Historical Data function call
Local dataframe containing Open, High, Low, Close

Updating historical indicators:

Once the historical call has been made, we have the Open, High, Low and Close prices of the past 150 days. We then iterate over this data and update our technical indicators. The indicators we chose for were representative of each of the four facets of the market – trend, momentum, volume and volatility.

Market Facet	Indicator	Remarks
Trend	Exponential Moving Average	This moving average puts more weight to recent observations.
Momentum	Moving Average Convergence Divergence	Shows the relationship between the fast- and slow-moving averages.
	Stochastic Oscillator	Compares the closing price of the currency pair with the range of its prices over a certain period of time.
Volume	Relative Strength Index	Measures the magnitude of recent price changes to analyse overbought or oversold conditions.
Volatility	Bollinger Bands	A set of lines plotted 2 standard deviations away from the simple moving average.
	Standard Deviation	Measures the variance of the currency pair's price from its mean.

Features for classification

The data processing from the previous section is common to both the regression and classification feature forming. This section, however, is specific to the classification model.

Computing previous price changes:

The next step involves computing the previous price changes for every row. This means that for every row in our dataset, price_shifted_by_n represents the difference in price between (today-n) and (today-n+1). We can visualize this better using the following diagram:

Day	Price	Price_shifted_by_1	Price_shifted_by_2
1	100	-	-
2	97	-3	-
3	102	5	-3
4	90	-12	5
5	95	5	-12
6	98	3	5
7	105	7	3

We then combined the close price changes for our traded currency as well as the correlated currency, and added the technical indicators. Our final features for the classification dataframe are below:

Computing features from Technical Indicators:

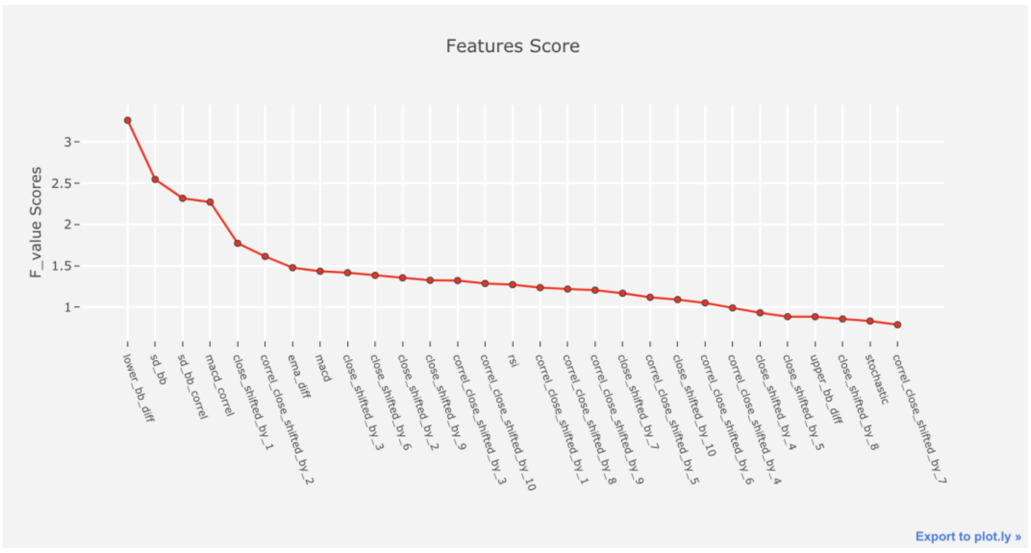
We didn't plainly use the technical indicators, as we wanted to optimize our model to predict the next direction of price movement. Intuitively, when we wanted to tell just the up/down direction, it made sense to get the relative position of actual price from the calculated indicators. These were our computed features:

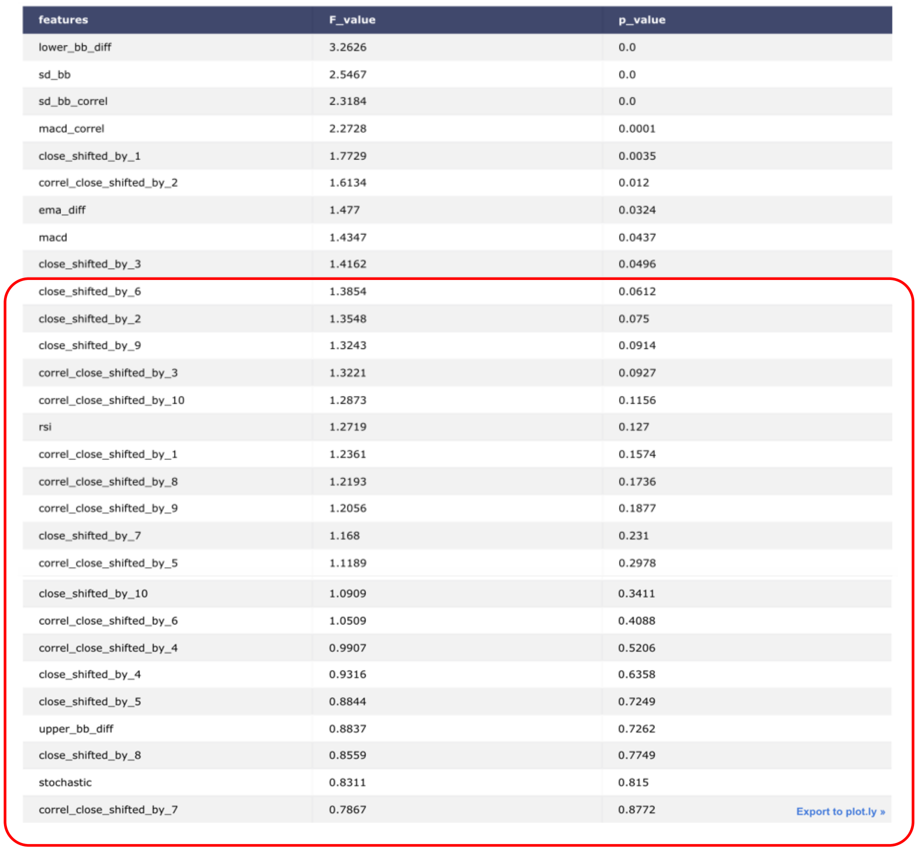
Computed Feature	**Explanation**
Lower_BB_Diff	Difference between upper bollinger band and the close price
Upper_BB_Diff	Difference between lower bollinger band and the close price
EMA_Diff	Difference between EMA and close price
MACD	Left as is, since it is already a relative rather than absolute value

Features Selection (Classification)

Statistical Reasoning

To select important features for the direction prediction model, we use the f_value scoring function as a statistical measure for classifiers. We evaluated the significance of each feature at the 5% significance level. The f_value scores are plotted using the line graph employed in plotly. The higher the f_value score, the more significant the feature is. We have also plotted a dataframe of the p_value for each feature to further analyse and identify the insignificant features.





features	F_value	p_value
lower_bb_diff	3.2626	0.0
sd_bb	2.5467	0.0
sd_bb_correl	2.3184	0.0
macd_correl	2.2728	0.0001
close_shifted_by_1	1.7729	0.0035
correl_close_shifted_by_2	1.6134	0.012
ema_diff	1.477	0.0324
macd	1.4347	0.0437
close_shifted_by_3	1.4162	0.0496
close_shifted_by_6	1.3854	0.0612
close_shifted_by_2	1.3548	0.075
close_shifted_by_9	1.3243	0.0914
correl_close_shifted_by_3	1.3221	0.0927
correl_close_shifted_by_10	1.2873	0.1156
rsi	1.2719	0.127
correl_close_shifted_by_1	1.2361	0.1574
correl_close_shifted_by_8	1.2193	0.1736
correl_close_shifted_by_9	1.2056	0.1877
close_shifted_by_7	1.168	0.231
correl_close_shifted_by_5	1.1189	0.2978
close_shifted_by_10	1.0909	0.3411
correl_close_shifted_by_6	1.0509	0.4088
correl_close_shifted_by_4	0.9907	0.5206
close_shifted_by_4	0.9316	0.6358
close_shifted_by_5	0.8844	0.7249
upper_bb_diff	0.8837	0.7262
close_shifted_by_8	0.8559	0.7749
stochastic	0.8311	0.815
correl_close_shifted_by_7	0.7867	0.8772

The section of the table boxed out in red represents features that are not statistically significant at the 5% significance level as they have p-values less than 0.05. This is a considerable amount of features to drop. As such, for our feature selection, we decided to factor fundamental reasoning on top of statistical measures to weed out the truly insignificant features in our feature selection.

Fundamental Reasoning

Successive Days

Logically, our features require the prices for a successive amount of days in order to predict the direction of next day's price. Hence, if we were to drop any day's price during the successive period, the data will lose the true representation of the trend in prices. As such, it is fundamental that we maintain the features for successive days' prices.

Relevant Indicators

The indicators used in our data have to be useful and logical. From the results of the statistical measure, we can see that `rsi` and `stochastic` are both insignificant.

Additionally, from the fundamental point of view, `rsi` and `stochastic` do not provide much value because having past days price changes gave an indication of the momentum of price movement, and hence these 2 indicators were superfluous.

Eventually, we ended up dropping `rsi` and `stochastic` due to the above reasons and `upper_bb_diff` because it was statistically insignificant. Moreover, we are using `upper_bb_diff` in our Risk Management Strategy, hence dropping in from our features set does not lead to much information loss.

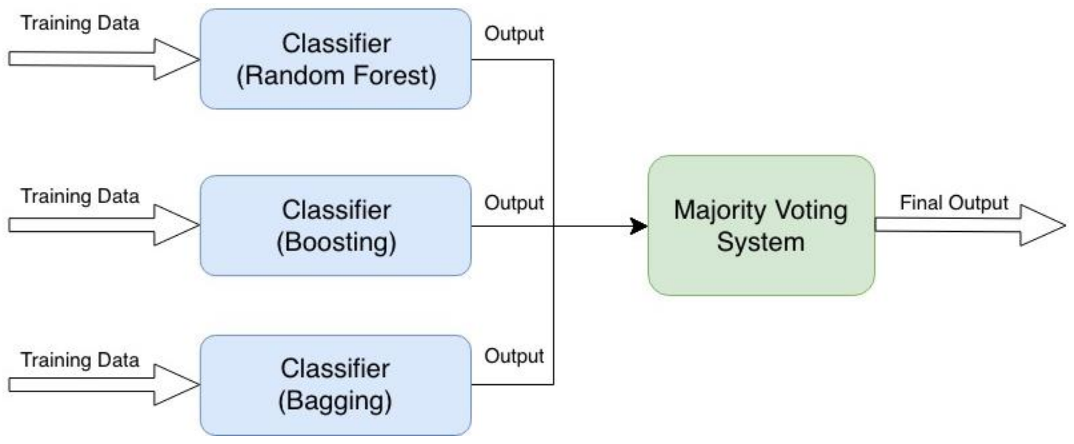
Final Dataframe for classification

close_shifted_by_1	**close_shifted_by_6**	**correl_close_shifted_by_1**	**correl_close_shifted_by_6**
close_shifted_by_2	**close_shifted_by_7**	**correl_close_shifted_by_2**	**correl_close_shifted_by_7**
close_shifted_by_3	**close_shifted_by_8**	**correl_close_shifted_by_3**	**correl_close_shifted_by_8**
close_shifted_by_4	**close_shifted_by_9**	**correl_close_shifted_by_4**	**correl_close_shifted_by_9**
close_shifted_by_5	**close_shifted_by_10**	**correl_close_shifted_by_5**	**correl_close_shifted_by_10**
MACD	**Lower_BB_Diff**	**Sd_BB**	**EMA_Diff**

The labels for this dataframe are the direction change for close prices (1 if up and 0 if down).

Model Building – Classification

We built ensemble models using Bagging, Boosting and Random Forest. The diagram below illustrates the process and mechanism of the stack ensemble.



To fine-tune, we used a custom-built (from scratch) grid-search, with the following search space:

Max_depth = [10, 15, 20, 30]

Num_estimators = [100, 200, 300, 500]

Criterion = ["gini", "entropy"]

To measure the performance of the 3 machine learning classifiers, I used a time series split, rather than a traditional k-fold split. This breaks the time series into 2 consecutive datasets, and then does a 70-30 train-test routine on each half. To illustrate how this works, consider an array of time series data below:

$$[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]$$

Now, we will pass along this dataset twice:

Iteration 1 – Train on $[1, 2, 3, 4, 5, 6, 7]$, test on $[8, 9, 10]$. Record the accuracy.

Iteration 2 – Train on $[11, 12, 13, 14, 15, 16, 17\ ]$, test on $[18, 19, 20]$. Record the accuracy.

The final accuracy is calculated as the average of both accuracies. The reason for doing this is to preserve the 'consecutive-ness' of the time series data. I used a split with a value of 4.

The results of the classification models are as below:

Model	Number of Estimators	Split 1 Accuracy	Split 2 Accuracy	Split 3 Accuracy	Split 4 Accuracy	Overall Accuracy
Random Forest	100	0.6875	0.5625	0.5300	0.5250	0.5763
	300	0.7500	0.6875	0.5800	0.5415	0.6398
	500	0.8125	0.6250	0.5725	0.5415	0.6379
	700	0.8125	0.6250	0.5725	0.5415	0.6379
Bagging	100	0.8125	0.5625	0.5525	0.5200	0.6119
	300	0.8125	0.5625	0.5525	0.5200	0.6119
	500	0.8125	0.6250	0.5570	0.5200	0.6286
	700	0.8125	0.5625	0.5525	0.5200	0.6119
Boosting	100	0.7000	0.5625	0.5525	0.5250	0.5850
	300	0.7000	0.5625	0.5525	0.5250	0.5850
	500	0.7000	0.5625	0.5530	0.5305	0.5865
	700	0.7000	0.5625	0.5530	0.5305	0.5865

Our accuracy results show that the predictive capability of our models generally decrease with each time series split. This makes sense, as training on data from very far back may not be relevant, as the financial markets are constantly changing. A comforting fact was that our accuracy was consistently above 55%, which told us that our model was not simply randomly guessing. From the research papers that we had read, we knew that 55% win rate was enough for high returns, given a robust risk management strategy; hence we were confident with our classification models overall.

Overfitting

The financial Markets are extremely random, and this renders our model prone to overfitting as it may latch on to otherwise nonsensical patterns. Our Random Forest model is the best candidate to tackle this, since it drops features by to prevent overfitting. However, we found a better way to mitigate this problem – to combine the 3 classifiers to form a stack ensemble. This works by a majority voting, where if 2 or more models predict a direction change, our overall model takes that to be the label.

Confusion Matrix

Our final classification decision was made using a majority voting decision from the 3 stacked ensemble methods. The results are summarized in the accuracy table and confusion matrix below:

Correctly Classified	Wrongly Classified	Total Predictions
132	74	206
Overall Model Accuracy: 64.07%		

		Predicted Class		
		Up	Down	Recall
Actual Class	Up	54	31	63.53%
	Down	43	78	
	Precision	55.67%		Recall

The above confusion matrix indicates that the accuracy from the stacked ensemble is 64.1% - higher than any of the individual models, which indicates that our thinking is right. However, there are still many False Positives, and we would have to account for them in our risk management strategy to ensure that we make money in the long run.

Features for Regression

For the regression model, we are concerned with the absolute price rather than the relative positions of price compared to the technical indicators. Hence the features used were:

Open	Close	Low	High	RSI
Open_shifted_by_1	Close_shifted_by_1	Low_shifted_by_1	High_shifted_by_1	MACD
Open_shifted_by_2	Close_shifted_by_2	Low_shifted_by_2	High_shifted_by_2	Upper_BB
Open_shifted_by_3	Close_shifted_by_3	Low_shifted_by_3	High_shifted_by_3	Lower_BB
Open_shifted_by_4	Close_shifted_by_4	Low_shifted_by_4	High_shifted_by_4	Std_Dev_BB
Open_shifted_by_5	Close_shifted_by_5	Low_shifted_by_5	High_shifted_by_5	Stochastic Oscillator
Open_shifted_by_6	Close_shifted_by_6	Low_shifted_by_6	High_shifted_by_6	EMA

Here, the shifted prices refer to the actual price (whether it is open, high, low or close) from n days ago, rather than the difference.

Note on Features Selection (Regression)

Note: We did not employ features selection for our Regression Model because the Neural Network has the capability of reducing the extent of overfitting through the adjustment of the dropout rate. Moreover, Neural Networks are known for their ability to handle a large amount of features.

Model Building – Regression

In our price prediction analysis, we incorporated the LSTM regression model. It predicts today's close price which will then be compared against the yesterday's price. This yesterday's price is obtained on every `OnData` daily initialization which includes a historical call.

Model Setup

The model is built once in the `Initialise` function because the window frame for the tracking lasted only 2 weeks from November 7th to November 21th. This is a relatively short window frame considering that our buy-sell strategy only operates on a daily basis.

Data Transformation

We use the `np.reshape()` function to reshape the `scaled_features` from 2-dimension to 3-dimension in order to fit the data into the LSTM model.

Model Tuning

We want to maximise the performance and accuracy of the model. Hence, we finetune the hyperparmeters of the model. These hyperparmeters include: number of training epochs and number of cells (neurons).

Hyperparameters	Experimented Values
Number of Training Epochs	[100, 200]
Number of Neurons	[100, 200]

Note: We did not experiment with many values because it will reduce the overall computational efficiency. A lot of time is needed to run and analyse the performance of every possible permutation of the hyperparameter values.

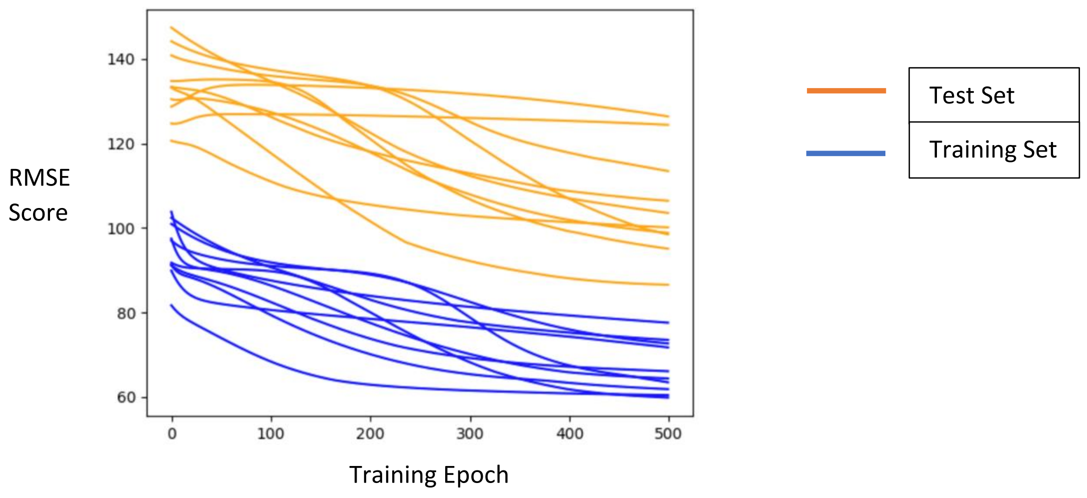
<u>Evaluate base on RMSE</u>

RMSE (Root Mean Square Error)

- It represents the sample standard deviation of the differences between predicted values and observed values (called residuals).
- $$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$
- RMSE is used as it punishes large errors and results in a score that is in the same units as the forecast data.

Tuning the Number of Training Epoch

The figure below shows test and train RMSE scores when it was run for varying number of training epoch – from 0 to 500.



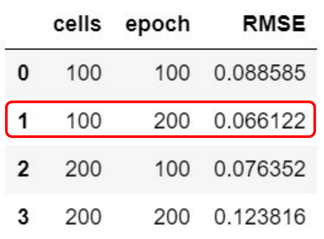
The results clearly show a downward trend in RMSE over the training epochs for almost all of the experimental runs. This is a good sign, as it shows the model is learning the problem and has some predictive skill.

Hence, the results suggest that more training epochs will result in a more skillful model. As such, we utilized a high training epoch value of 200 instead of 100. We avoided a very high epoch value (above 200) due to high computational workload and possible overfitting.

Tuning the Number of Neurons

The number of neurons affect the learning capacity of the network. Generally, more neurons would be able to learn more structure from the problem at the cost of longer training time. More learning capacity also creates the problem of potentially overfitting the training data.

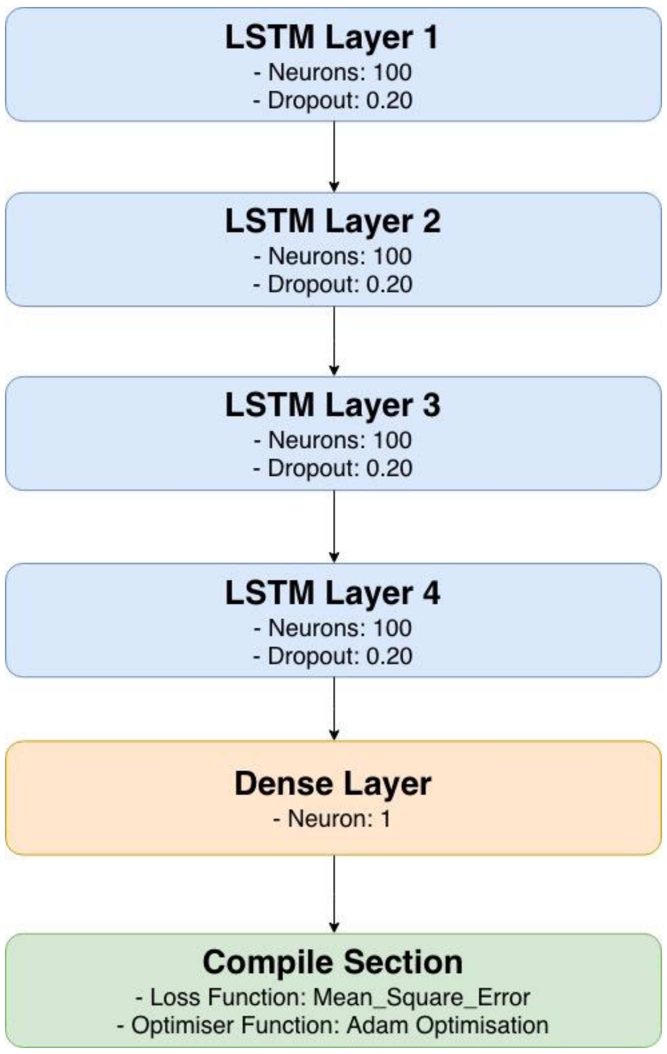
Optimised Parameters



	cells	epoch	RMSE
0	100	100	0.088585
1	100	200	0.066122
2	200	100	0.076352
3	200	200	0.123816

We ran the entire set of experimented hyperparameter values and gathered the associated RMSE scores in a dataframe. From the dataframe, the optimized values are 200 cells and 200 training epochs, since they have the lowest RMSE score. Hence, we used these hyperparameters to build our final LSTM model.

Model Structure



In the Model Structure, there are 4 LSTM Layers consisting of 100 neurons with Dropout Rate of 0.20. The dropout rate is included to slow down learning of the recurrent LSTM model and reduce any potential overfitting problems. This is followed by a Dense Layer containing only 1 neuron. In the compile section, we use the "Mean_Squared_Error" as the Loss Function and the "Adam Optimisation" as the optimizer function.

The number of neurons is determined by the hyperparameter finetuning. The dropout rate, loss function and optimizer function, as well as the number of layers are all determined by trial and error.

Note on Correlation between Features

When selecting features, it was important for us to avoid including features that were correlated with each other, as this would defeat the principle of parsimony – keeping every model as simple as possible. This is why we chose technical indicators that each represent different aspects of the markets, which ensures that they are not inter-correlated. To clarify, our correlated currency prices are correlated with the price of our underlying security – the label; they are not correlated with each other. Hence, including them in the feature set in fact boosted our model performance.

Note on Feature Scaling

Since the magnitudes of the difference features are different – for example, RSI is in the range of 0 to 100 while the price fluctuates around the 0.8 to 1 region. Scikit-Learn's min-max scaler standardizes the data array in the following manner:

$$newNum = \frac{thisDatapoint - minOfArray}{maxOfArray - minOfArray}$$

Applying the above formula to every point in each column will convert it to a value between 0 and 1, and hence the ML model will be fair in its weighting of each feature.

It is important to note that the same X and Y minMaxScaler when training and when predicting to ensure consistency in the inputs for the model. Furthermore, we inverse transform our outputs to obtain their real value.

OnData Algorithm

In this algorithm, the data gets updated on a daily basis. The updates include: (1) Making a historical call for yesterday's prices, (2) Appending and Popping the Features, (3) Updating the Indicators' Values. In this section, we will focus on 2 important aspects: (1) Retraining and (2) Risk Management Strategy.

Retraining

During our research process, we initially implemented a rolling window model retraining – this meant that at every new data arrival, we would pop the oldest value from our features, append the latest value and retrain both our models. However, we decided to drop this as our trading window was only for 2 weeks, which was not long enough for retraining to make a significant difference. In fact, it only took up additional computational power and made our backtesting process extremely time consuming. However, if we were to deploy our algorithm for the long run, we would implement a monthly retraining where we pop the oldest 30 datapoints, append the latest 30 and retrain. This technique would provide a meaningful update to our models.

Risk Management Strategy

We decided on the use of various indicators in our risk management strategy. The following table provides explanation on the value provided by each indicator used in the strategy.

Value that each Indicator brings to the Strategy

RSI	RSI values of 70 or above indicate that a security is becoming overbought or overvalued, and therefore, may be primed for a trend reversal or corrective pullback in price. An RSI reading of 30 or below is commonly interpreted as indicating an oversold or undervalued condition that may signal a trend change or corrective price reversal to the upside.
Upper_BB_Diff **Lower_BB_Diff**	Bollinger Bands are used to identify potential overbought/oversold areas as well as the volatility of the markets. In an uptrend, long positions are made near the lower Bollinger Band. In a downtrend, short positions are made near the upper Bollinger Band. When the Bollinger Bands is in a squeeze, it signals the market is "ready" to breakout.
MACD	The moving average convergence divergence is a lagging indicator used to follow trends. It consists of two exponential moving averages and a histogram. The most important signal of the MACD is when the trigger line crosses the MACD up or down. This gives us a signal that a trend might be emerging in the direction of the cross.

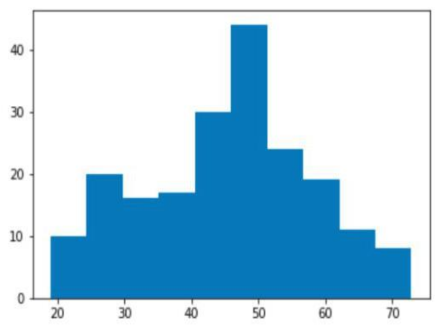
	Bearish Divergence – price increasing and the MACD recording lower highs Bullish Divergence – price decreasing and the MACD recording higher lows

Determining the Appropriate Threshold based on Historical Data

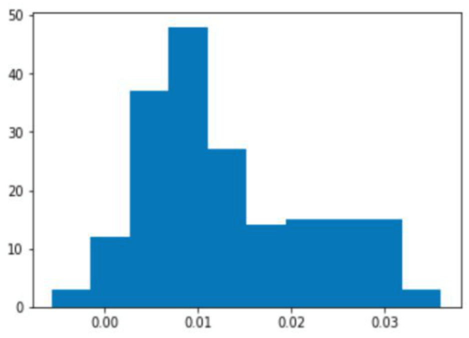
After understanding how the selected indicators complement each other in the risk management strategy, we then had to determine the appropriate threshold values for each indicator. These values are not randomly determined but are instead contextualized in the `AUDUSD` forex market. Using the past 220 days Historical Data, we conducted a thorough Exploratory Data Analysis in order to understand the distribution of various indicators.

In our research environment, we plotted a histogram to analyse the spread of values for each indicator. The distributions are plotted below:

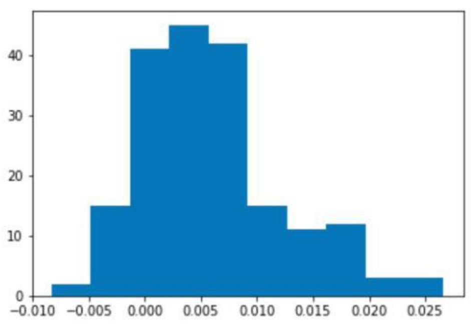
RSI Indicator



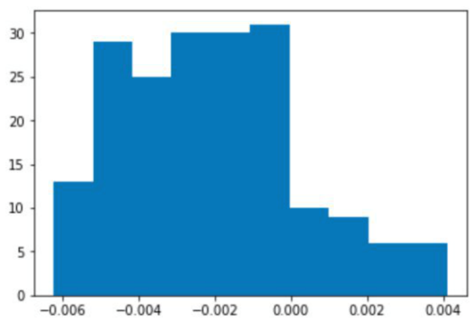
Upper BB



Lower BB



MACD



Constructing the Buy and Sell Signals

The buy and sell signals will be constructed using the indicators as well as information from the above distributions.

RSI Indicator	Buy: RSI value is between 55 and 70 Sell: RSI value is between 20 and 65
Upper BB Difference	Sell: 0 <= Upper BB – current price <= 0.025
Lower BB Difference	Buy: 0 <= current price – Lower BB <= 0.02
MACD	Buy: MACD > 0.001 Sell: MACD < -0.001

The indicators and their threshold values will then be combined to form holistic buy and sell signals. All conditions must be fulfilled for the signal to be positive.	
Buy Signal	• 55 < RSI Value < 70 • 0 <= current price – Lower BB <= 0.02 • MACD > 0.01
Sell Signal	• 20 < RSI Value < 65 • 0 <= Upper BB – current price <= 0.025 • MACD < -0.001

Finalising the Risk Management Strategy

In our Risk Management Strategy, we utilized the predictive power of both the classification (direction prediction) model as well as the regression (price prediction) model. On top of the machine learning models, we also used the buy and sell signals determined by indicators and their threshold values to guide our strategy into making better decisions.

Our strategy is considered to be relatively stringent and conservative as we aim to maximise gains and minimize losses. The following information explains the methodology of our strategy:

Summary: *(1) Direction Prediction (2) Price Prediction (3) Buy and Sell Signals are incorporated into the Risk Management Strategy*	
Long Position	Our algorithm will adopt long position in either of these 2 scenarios: (1) Upward Direction in prices **AND** today's predicted close price is 1.015 times greater than the current price (2) Upward Direction in prices **AND** positive buy signal
Short Position	Our algorithm will adopt short position in either of these 2 scenarios: (1) Downward Direction in prices **AND** today's predicted close price is 0.995 times lesser than the current price (2) Downward Direction in prices **AND** positive sell signal

Stop Loss – Take Profit (When do we liquidate)

*Cost Basis refers to the average price of the currencies in the current portfolio.

Long Portfolio	*Stop Loss*: Current Price <= 0.98 * Cost Basis *Take Profit*: Current Price >= 1.03 * Cost Basis
Short Portfolio	*Stop Loss*: Current Price <= 0.98 * Cost Basis *Take Profit*: Current Price >= 1.03 * Cost Basis

The weightage for Stop Loss and Take Profit are the same for both the Long and Short Portfolio. We assigned a higher weightage to `Take Profit` because for every `Take Profit` and `Stop Loss` committed, the amount that we profit will outweigh the amount that we loss. This ensures that our net profit does not go into the negative region.

Assuming we win 50% of our trades, our take profit is always 3% and our stop loss is -2%. In the long run, we will profit at least 1%. It is important to note, however, that the accuracy of our forex trading model is 64%, greater than the 50% mark. This makes our strategy extremely robust. Hence, it will be expected that we would enjoy a more than 1% profit over time.