



# ALBATROSS STRATEGY

Authors

Rene Lim | Ng Jian Lai | Benedict Soh

NUS FINTECH SOCIETY  
[nusfintech@gmail.com](mailto:nusfintech@gmail.com)

## Summary

This report applies what we learnt during this semester to create a profitable model for trading. It provides insights on why we make such decisions, such as why we chose to ensemble three models, how we chose our hyperparameters, the limitations we faced and how we improved our models and what we could do in the future to improve it should time and resources allow.

## Contents

Summary .....	1
Ideas Used in Class .....	3
Selection of Models.....	6
Selection of Forex and Updating of Model .....	8
Live Testing .....	9
Decisions Made by Model.....	10

## Ideas Used in Class

Our algorithm made use of several concepts taught in class.

Firstly, we manually ensemble three models to lower the variance and reduce the likelihood of overfitting our models. Despite only learning homogeneous ensemble methods such as random forest, we extended our knowledge and made use of heterogeneous ensemble methods by assembling logistic regression, LSTM neural network and random forest regressor together.

Next, we also went through several data transformation processes such as normalising, scaling and converting to categorical variables. We made use of min-max scalar which will be further explained in our model write-up. Our dependent variable was converted from continuous to categorical for decision making. With two categories and depending on the prices, we came up with a rule to decide if this outcome spells a buy or sell decision.

Our algorithm also made use of a modified version of majority voting. Depending on how many of our models gave out a buy signal, we will purchase currencies with a proportionate amount of our assets.

We explored the correlation between variables in our logistic regression on the basis of the correlation coefficient between two different currency pairs and only choosing the closely related ones to help us predict the price movement of the currency we intend to invest in. For instance, a correlation matrix was plotted with the price of EURUSD against that of seven other currencies across ten lags, with EURJPY and GBPUSD consistently following closely.

Through exploratory data analytics, we explored popular cryptocurrencies, stocks and forex and realised that prices of bitcoins plunged \$13 billion in the market. Most of the assets and cryptocurrencies show neither a hopeful nor an increasing trend. Forex on the other hand appears to be more stable and consistent. With the top currency pairs occupying more than 30% of the market share, we decided to delve

into EURUSD as our main investment for more effective learning and application to our model.

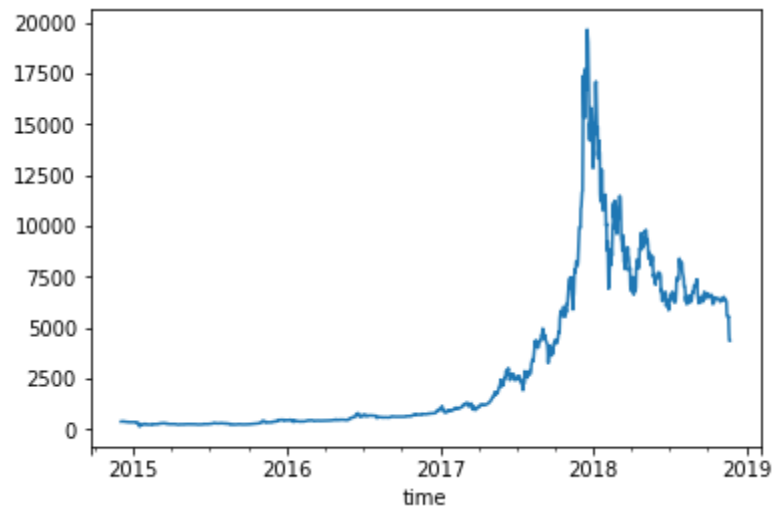


Figure 1: Plunging Prices of BTC

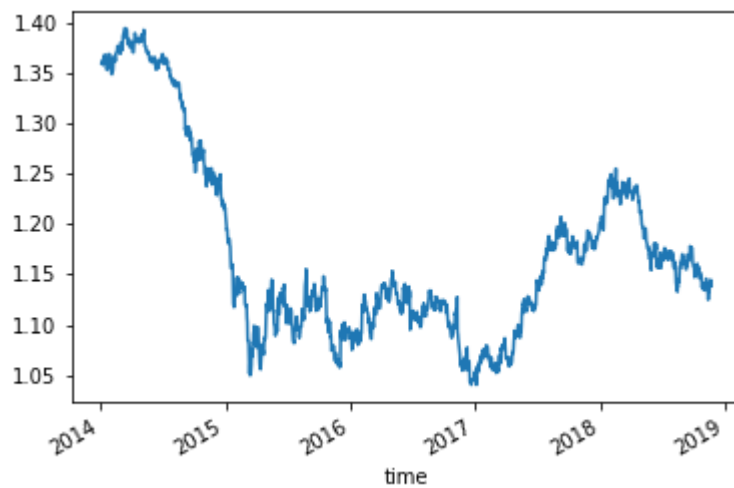


Figure 2: Relatively Stable Prices of Forex

Lastly, we conducted gridsearch to discover optimal hyperparameters for our LSTM using nested loops.

```

# Make cells and epochs to be used in grid search.
cell1 = [50, 100]
epochs = [50, 100, 150, 200]
opt = ['RMSprop', 'Adagrad']
dropout = [0.10, 0.15, 0.2, 0.25, 0.3]
# creating a dataframe to store final results of cross validation for different combination of cells and epochs
df = pd.DataFrame(columns=['cell', 'epoch', 'dropout', 'opt', 'mse'])

# Loop for every combination of hyperparameters
for i in cell1:
    for j in epochs:
        for d in dropout:
            cvscores = []
            print(" Trying " + str(o) + " " + str(i) + " " + str(j) + " " + " " + str(d))

            for train_index, test_index in tscv.split(X_data):
                X_train, X_test = X_data[train_index], X_data[test_index]
                Y_train, Y_test = Y_data[train_index], Y_data[test_index]

                X_train= np.reshape(X_train, (X_train.shape[0],1,X_train.shape[1]))
                # print("X input to LSTM : " + str(X_train))
                X_test= np.reshape(X_test, (X_test.shape[0],1,X_test.shape[1]))
                #print("Y input to LSTM : "+ str(Y_train))

                #print("start seq modeling to find best hyperpara ")
                model = Sequential()
                model.add(LSTM(1, input_shape = (1,5), return_sequences = True))
                model.add(Dropout(d))
                model.add(LSTM(1))
                model.add(Dropout(d))
                model.add(Dense(1))
                model.compile(loss = 'mean_squared_error', optimizer = o, metrics = ['mean_squared_error'])
                model.fit(X_train, Y_train, epochs = j, verbose = 0)
                # print("end seq modeling to find best hyperpara ")

                scores = model.evaluate(X_test, Y_test, verbose = 0)
                # print("indi score ")

                cvscores.append(scores[1])
                # print("append score")

            MSE= np.mean(cvscores)
            #print("find MSE")

            df = df.append({'cell':i, 'epoch':j, 'dropout':d, 'opt': o, 'mse':MSE}, ignore_index=True)

print(df)

```

Figure 3: Gridsearch

## Selection of Models

As mentioned earlier, our algorithm is an ensemble of three trading models, namely Logistic Regression, LSTM and Random Forest.

Logistic regression was selected as it could use the information available between several highly correlated currencies to predict the price movement of the currency we want to invest in, instead of solely focusing on one currency. We do so by using the lags of data to predict future prices.

Our second model is built using LSTM, as they are very good at holding long term memories. In other words, the prediction of the  $n^{th}$  sample in a sequence of test samples can be influenced by an input that was provided several steps back. Preserving the long term dependencies in a network is done by its gating mechanisms. The network can store or release memory in real time through gating mechanisms. Working with currencies with fluctuating rates makes LSTM a good choice as it is capable of modeling sequential or temporal aspects of the data, making it widely used for texts, videos, and in our case, time-series.

To our initial surprise, when run independently across 2018 Jan - 2018 Dec, our Logistic Regression and LSTM models yielded the same results, suggesting that both models have the same final loss rate, win rate, profit, etcetera. We humbly conclude that these phenomena is due to both models being able to predict the new datapoint to the same degree of accuracy, of being either more or less, than the last available datapoint. While it might seem redundant, at this point, to keep both models in the ensemble, we have to recognise the ability of logistic regression to pick up unusual movements of other currency pairs, as it takes into account the trend of highly correlated currency pair. As we cannot dismiss the issue of vanishing gradient, LSTM helps us avoid said problem while we continue taking more data across time.

The random forest model, our simplest one yet, was built on data from the past 300 days, and it requires the past three days of data to predict the next day prices. When

run independently, it gave a surprising \$8k returns as compared to its counterparts with only \$1k return in profits across 2018.

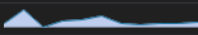
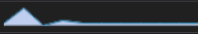
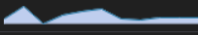
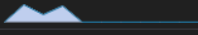

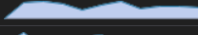
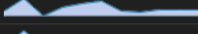
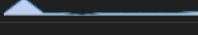
Overall Statistics				<a href="#">Download Results</a>
Total Trades	29	Average Win	0.77%	
Average Loss	-0.30%	Compounding Annual Return	11.391%	
Drawdown	3.100%	Expectancy	0.331	
Net Profit	8.692%	Sharpe Ratio	1.521	
Loss Rate	62%	Win Rate	38%	
Profit-Loss Ratio	2.55	Alpha	0.249	
Beta	-10.022	Annual Standard Deviation	0.058	
Annual Variance	0.003	Information Ratio	1.244	
Tracking Error	0.058	Treynor Ratio	-0.009	
Total Fees	\$98.23			

Figure 4: Performance of Random Forest Regressor



## Selection of Forex and Updating of Model

Unfortunately, due to the run-time limitations in both back-test and live, we are unable to run a model that carries out reinforced learning. However, it can be updated by the following steps:

For logistic regression, we analysed the accuracy of models using the two independent variables. Once the accuracy of such model falls below a certain threshold, we can dynamically change the two variables that are used to predict the price movement of our EURUSD by using the next highest correlated currency pairs.

For LSTM, we can input the new set of time series data of our currency pair EURUSD and run the grid search, use MSE to compare the accuracies of each iteration, and store each result as an entry in a dataframe. Thereafter, we shall select the row with the lowest MSE from the resulting dataframe, where each column in that row holds the optimum hyperparameter respectively to be used with the new updated dataset.

If not for the runtime restriction, this hyperparameter search will run every time when there is a new data entry and when the model is run to predict the next price. While these hyperparameters might remain optimal for a period of time, it might change as time goes by or with unfamiliar trends or change in prices.

With more time, we would retrain our model every week. Under our initialisation function, we would start trading every 30 minutes after the market opens. Furthermore, we can retrain our models at the end of every week.

Whenever reinforced learning is included, the ensemble crashed when running in both back-testing and live. This might be caused by a long amount of time (more than four hours) to run LSTM's gridsearch alone to select the optimum hyperparameters.

## Live Testing



The performance of our backtest model is quite similar to that of during live testing, with a return of -0.7%. While observing the returns every day during the live testing from round one, we realised that backtesting performed better most of the time. Backtesting returned positive rates almost all the time but live testing returned negative rates. We suspect that this is due to the steep and long fall from 11-12 November onwards after gaining a quick increase in gradient of rising. As such, we lost money as the price continued to fall when our models predicted it to rise by gradient. Hence, we should always be careful to conclude that positive backtesting performance of a model is a good model for live.

## Decisions Made by Model

On 11 - 12 November, our model took notice of the climb in positive gradient where the value was rising and profited from this. However, being unable to anticipate the sudden change in direction, we lost approximately the same amount the very next period and for a while thereafter. As we can see, our algorithm is very sensitive to trend of rising value, but less so to falling value.

Our models mostly made use of trend trading. We use past few data to predict the rise of prices in the future. However, our logistic regression model considers the trend of other currencies as well, which are not displayed in the logs. Should we be given the ability to view the prices, our ensemble of three still makes purchase decisions based on the signals given out by each model i.e. if one model says buy, we buy with 70%, if two models say buy, we buy with 80%. As such, it is very complex for us to see why our models make said decisions.