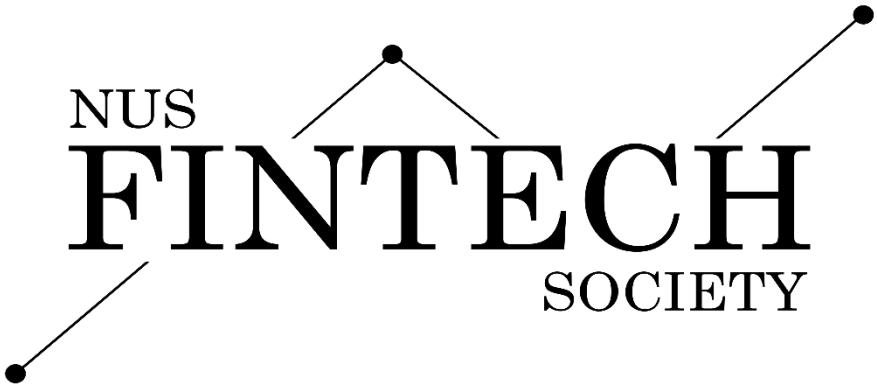

ALBATROSS STRATEGY

Authors

Rene Lim | Ng Jian Lai | Benedict Soh

NUS FINTECH SOCIETY
nusfintech@gmail.com

Contents

How to Run the Model

Our model consists of an ensemble of three models. All these models are controlled by a few variables at the beginning of the algorithm. It is ready to run after tweaking the parameters to your satisfaction. You can change the start and end dates, the resolution (current resolution is by the hour), and the choice of currency (currently self.currency = “EURUSD”). You can also change the size of data you want to input into our models by changing self.history_range. Should you have a better processing power you could perhaps change that to a larger number. Once all these variables are set, the model is ready to run.

Data Munging

We only selected the closing price, as we feel that it is the most important feature when it comes to price prediction. We ignored volume of trade as it does not affect our model decision making process. We also selected the top two closely correlated currency pairs to EURUSD. This can be increased to three or more features or changed to another currency pair.

Our first model is logistic regression. For the regression, we made use of other currency pairs (GBPUSD, EURJPY) to help us predict the movement of EURUSD, our main currency. The choice of these two currencies were decided using correlation coefficients. We plotted lags of EURUSD with 20 other currency pairs and chose the one that provided us with the highest correlation coefficient.

Normalising time series may be dangerous as it might result in out of bounds. Hence, we only used min-max scalar to scale our data. This is especially important if we need to compare between currency pairs, as the huge differences in ratio might introduce bias during training towards the one with a higher currency pair ratio.

Lags were also created for both our first constituent model log-model (ten lags) and second constituent model, the LSTM model (five lags) to take into account the trend within the time series.

Performance Measure

Accuracy of models were analysed during our research. It is not run during backtesting as it is difficult for us to keep track of it in the console. The models have an average of 61% accuracy, which is reasonable considering the volatility of the market.

To prevent overfitting, we ensembled three models: Logistic regression, LSTM neural network, random forest regressor. Random forest itself is an ensemble of decision trees. After each model makes their own predictions, we append the results to a decision list which has a max length of 3. Every model has a say in the buy-sell decision which will be further explained below.

For the LSTM component, the mean square error (MSE) was used as a hyperparameter performance measure to optimize the hyperparameters -- optimizer, cells, epoch and drop-out -- which is then used in LSTM.

```
# 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 datframe 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 hyperparametes
    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(i, input_shape = (1,5), return_sequences = True))
                    model.add(Dropout(d))
                    model.add(LSTM(i))
                    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 1: Gridsearch

Adagrad	100	200	0.15	0.000945

Figure 2: Best Optimizer with MSE of 0.000945

To avoid overfitting, we ensembled 3 different models for voting, with the usage being explained later in this document, each in their own special right.

Logistic regression takes into account the trend of highly correlated currency pair; hence it will be able to pick up unusual movements should it arise from other currency pairs. LSTM is chosen for its ability to avoid the problem of vanishing gradient as we continue to take in more data across time. Random forest regressor, already being an ensemble of models, makes use of averaging to reduce variance and bootstrapping to include many training datasets to ensure that we do not overfit the model.

We tried to retrain our model every time a new data is plugged into the algorithm. However, this took up a very long time for backtesting, as running LSTM's gridsearch alone takes more than four hours, hence we decided to train the model only once throughout the algorithm due to the runtime limitations. We could further improve it by allowing the model to retrain weekly instead of the pre-set resolution.

Risk Management

We made use of a modified version of majority voting to determine our buy-sell decision. Out of the three models, if only one model predicted an upward trend in price, we will purchase the currency with 70% of our assets. If two models predicted an upward trend, we purchase with 80% and if all three models produce the same results, we will purchase with 90% of our assets. To prevent multiple purchases in a single signal, we appended the currency purchased into a list, and we will only purchase the currency if it has not appeared on the list.

As long as there is more than one model with a sell signal, we will add the currency into a *shorts* list and short the currency. When there are currencies in said list, we will first find the average price of the currency. If the price of our currency is lower than average, we will buy back the currency. Else, we will continue to short it.

We considered a risk to reward ratio of 1:5. For the currencies that we go long, if prices fall 0.5% below the market average price or 0.1% above the average price, we will sell it. For our shortings, if the prices fall 0.1% below the market average price or 0.5% above it, we will sell it.