MAFS6100: Short-term Alpha Forecasting via ML/DL Techniques

Summary Report

Supervisor: Prof. Hong song Chou.

Author: Bokai CAO, Howard CHENG, Anthony WAH

Abstract

In this report, we will study the prediction accuracy of various short-term price forecasting models built with machine learning and deep learning models. The 3 main types of ML/DL models of interest are Multi-layer Perceptron, Linear Regression, and Random Forest Classification. We will study how these models perform on predicting the short-term price movements of 3 stocks from Taiwan Stock Exchange based on several commonly used performance metrics. Also, since the models we have chosen are dealing with either classification or regression problem, we will propose a new method to compare the accuracy of these 3 models.

The report will include several main sections in the following passage, namely: 1) Methodology, where data and details of experiments will be mentioned; 2) Analysis result, where performance of each model under different sets of experiments will be detailed; 3) Result Comparison, in which models will be compared using the new method we mentioned; 4) Conclusion, and 5) Potential next steps.

Methodology

Data

In this report, we will use the tick-level data of the 3 Taiwan stocks (0050, 2330, 2603) and their associated single stocks futures, from 2020-07-01 to 2021-05-31, for our analysis. The secondly snapshotted limit order book (LOB) data will be used as inputs to generate a set of 280 features which includes market microstructure, technical indicators and futures leading indicators (Total number of features will be 480 if we include futures data as well). Using this set of features, we will try to predict the X-second forward average return, which is defined as average of all forward returns (calculated using mid-price snapshots) of the next X seconds. X may range from 30 to 900.

Procedures

First, we will carry out labels analytics to see how the target labels are distributed. From there, we will propose how we should classify different classes to be used in our classification problems. Also, we will conduct features engineering to study how the features are correlated.

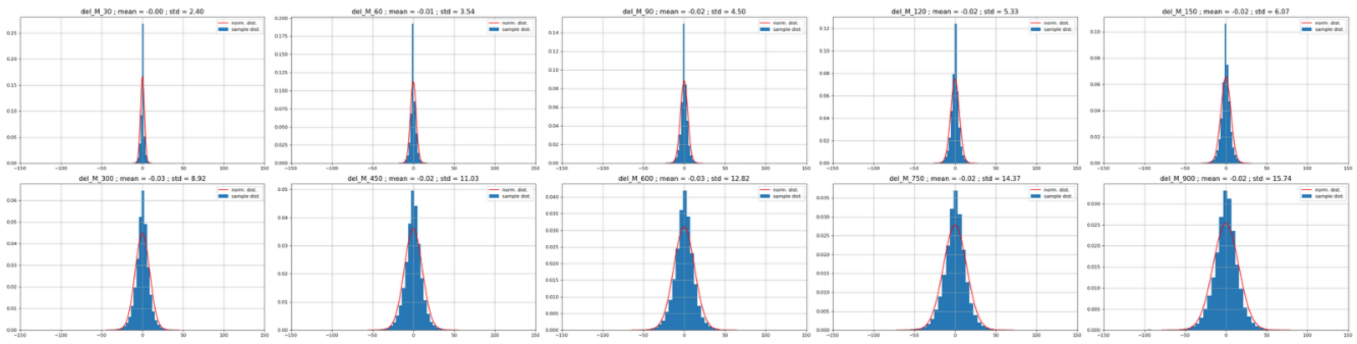
Then, we will carry out a preliminary training for each model. In this step, we will demonstrate how the performance of each model is measured and how they will be used in the next section. Also, this will help us get a first impression on how these models may perform on the target data set.

In the next section, which is the most important step, we will conduct a performance stability analysis. To construct an effective and accurate prediction model for building a trading strategy, both prediction accuracy and stability are the key elements. Firstly, we will train the model using data of day 0 to day t, in which t may range from 10 to 90. Then we will measure the performance of the model on data of day t+1 using the metrics detailed in the previous section and repeat the step again using data of day 1 to day t+1 for training another model. In short, we are updating the model by "rolling over" to new date whether new data is available, and then measure it prediction accuracy on the next day. By plotting the performance metrics on each day against the dates, we will be able to analyze the stability of each model.

Finally, we will compare the accuracy of each model by slightly modifying the regression approach into a classification model. In general, only predictions with an extremely positive/negative value will be considered as a positive/negative classification. Then we will compare the accuracy of all the 3 model.

Label Analytics

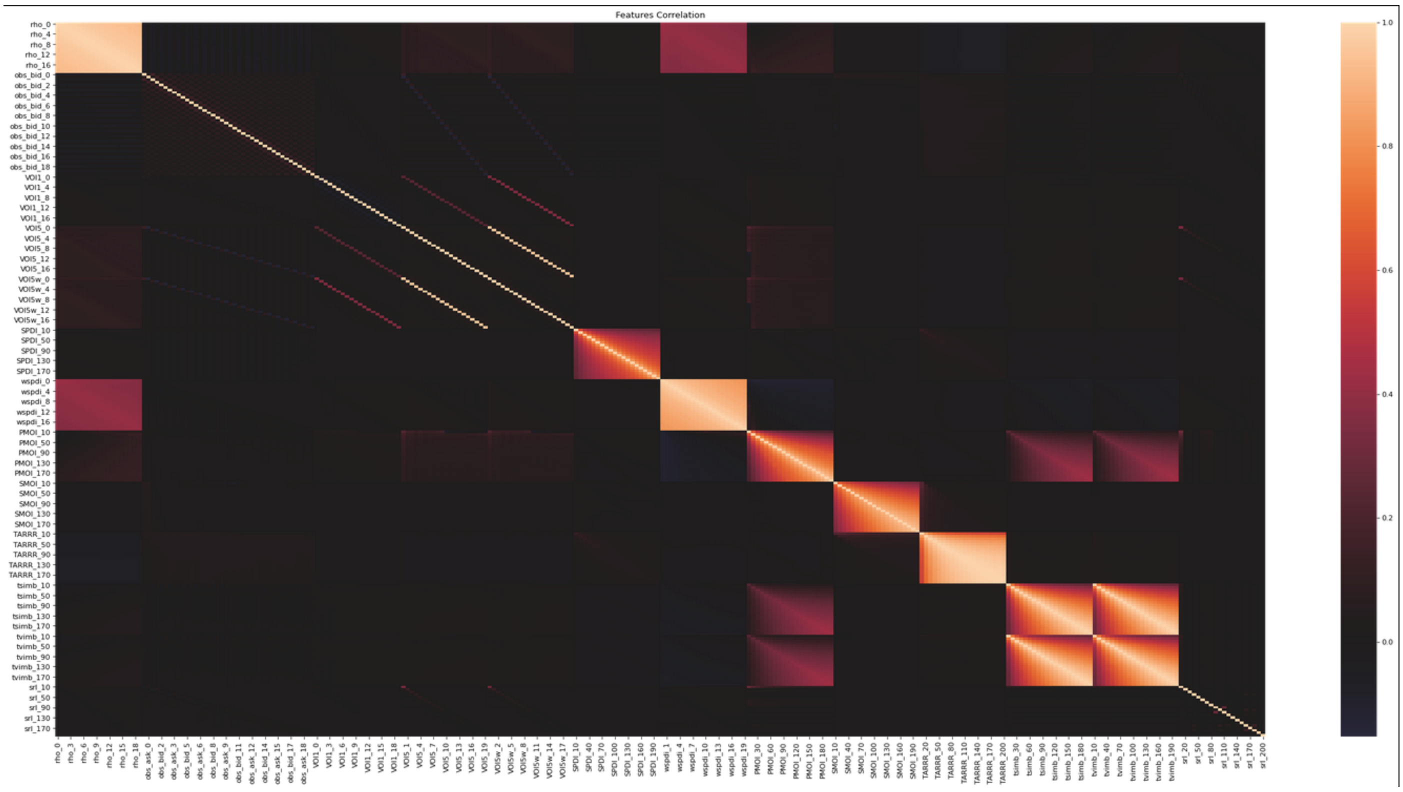
In the below figure (using stock 0050 for illustration), we can see that the variances of forward returns are higher under a longer forward time, which can be observed from the fatter tails. Also, forward returns of shorter forward time tend to be clustered around zero more tightly, while that of longer forward time tend to have more non-zero values. One may expect a more meaningful result from accurate prediction of non-zero values than that of zero values under a price prediction problem, since accurate prediction of zero values can hardly improve trading result but that of non-zero can significantly boost profit.



Feature Engineering

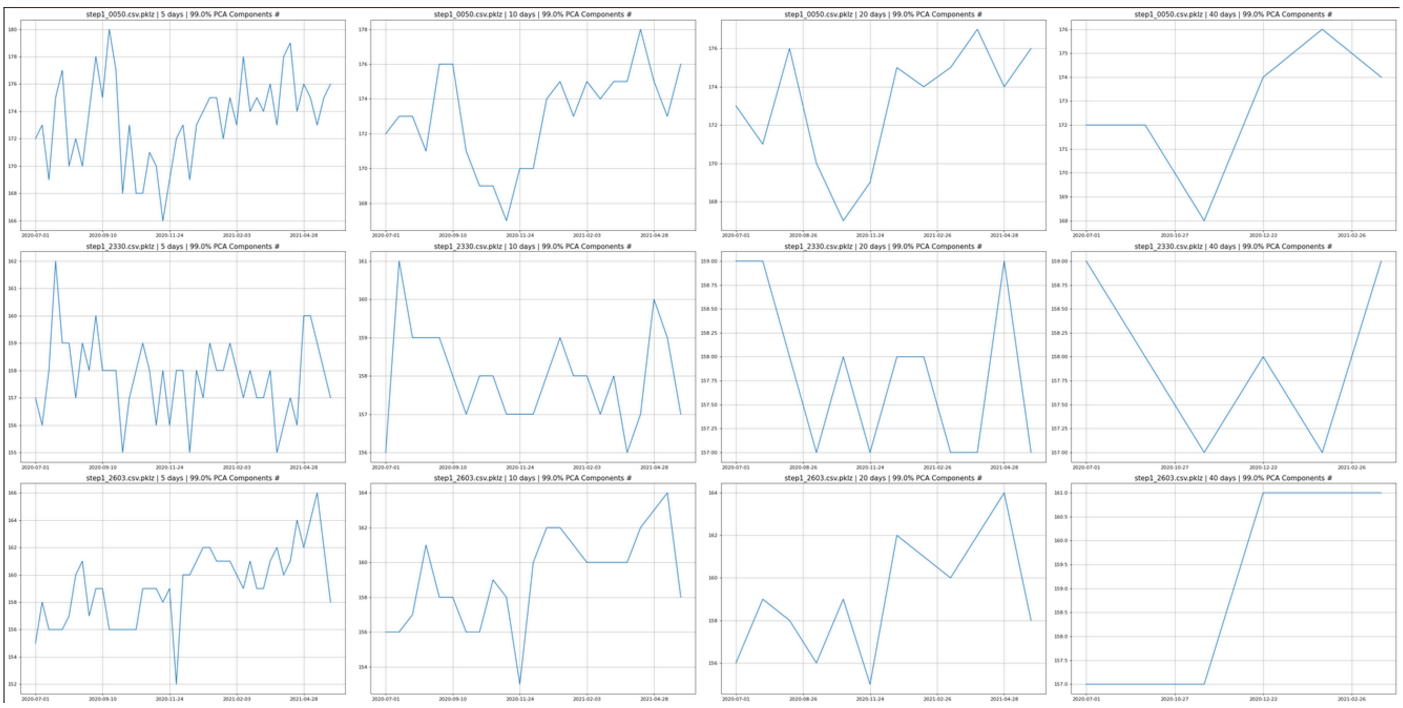
Features Correlation Matrix

Correlation matrix is constructed as shown below to compare the correlation between features in each stock. Since most of the spaces in the matrixes are in dark color implying close to zero correlation with other features, it may not be possible to select a few features which contain majority of the information in the features dataset. Approach of principal component analysis is therefore adopted instead to reduce number of features and the runtime of model.

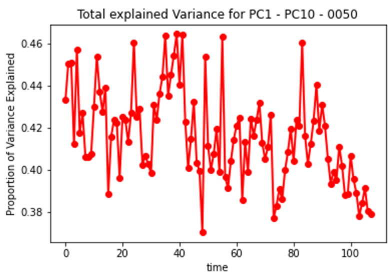


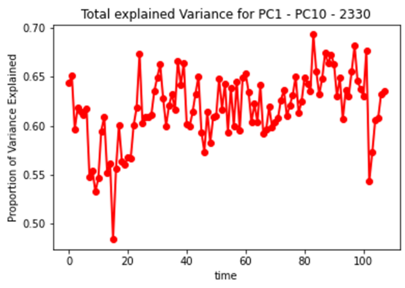
Principal Component Stability Analysis

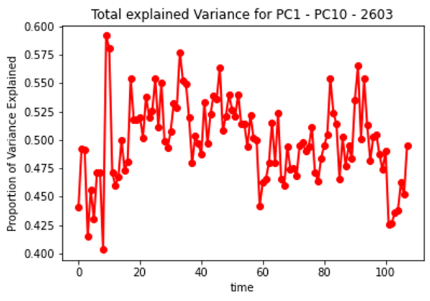
We adopted two approaches to study the stability of principal component analysis. In the first approach, we study the number of principal components required to explain 99% variance of a stock in particular window of time. For stock 0050.TW, it requires number of components between 170 and 190 to explain 99% variance in the rolling window of time. (5 days, 10 days, 20 days and 40 days) Meanwhile, it requires respectively 150 – 160 principal components and 150 – 170 principal components to explain 99% of variance for stock 2330.TW and 2603.TW.



For the second approach of stability analysis, the explained variance of the transformed test data is examined. In the rolling training and testing dataset window, principal component analysis model is trained with the first 6 days of features, and the 7th day of features data (test data) will be casted into the eigenvectors of the precedent trained principal component model. The total explained variance that the 10 principal components account for when casting the 7th day's data into principal component analysis is relatively stable: The total explained variance when stock 0050.TW test data are casted into 10 precedent trained principal components is ranging from 37% to 46% while that for stock 2330.TW and 2603.TW are separately between 48% to 69% and between 40% to 59%.





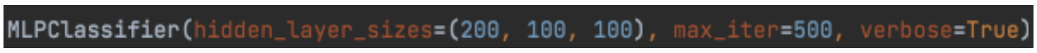


Neural Networks (Bokai CAO)

I mainly focused on neural networks methods, therefore I tried deep learning models including MLP, LSTM, CNN_LSTM and so on. The features I used are both 280 columns. Both of models are evaluated by classification precision. The Y_label, stock return, are classified into three classes using 25th quantile and 75th quantile.

Multilayer Perceptron

A multilayer perceptron is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable. MLPs are useful in research and application for their ability to solve problems stochastically, which often allows approximate solutions for extremely complex problems like fitness approximation. As a simple deep neural network structure, it's worth a try.

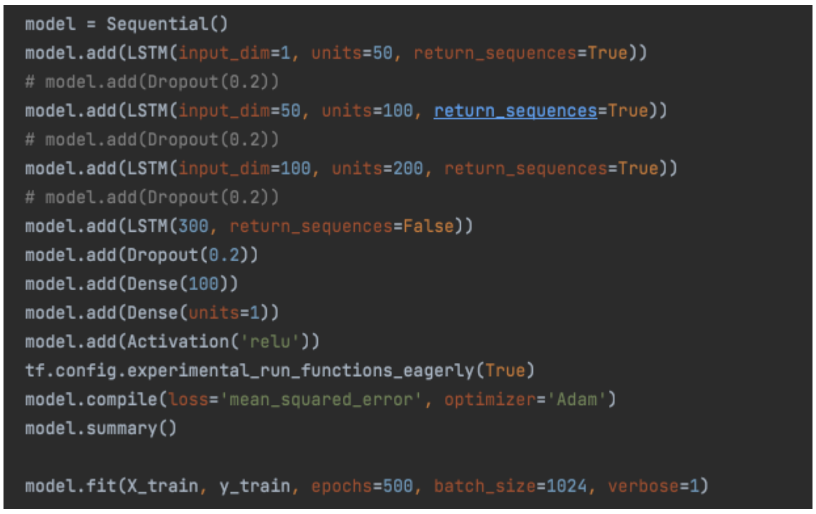


```
MLPClassifier(hidden_layer_sizes=(200, 100, 100), max_iter=500, verbose=True)
```

In this project, we use three layers and each of them contains 200, 100, 100 neurons and set maximum iteration as 500.

Long short-term memory (LSTM)

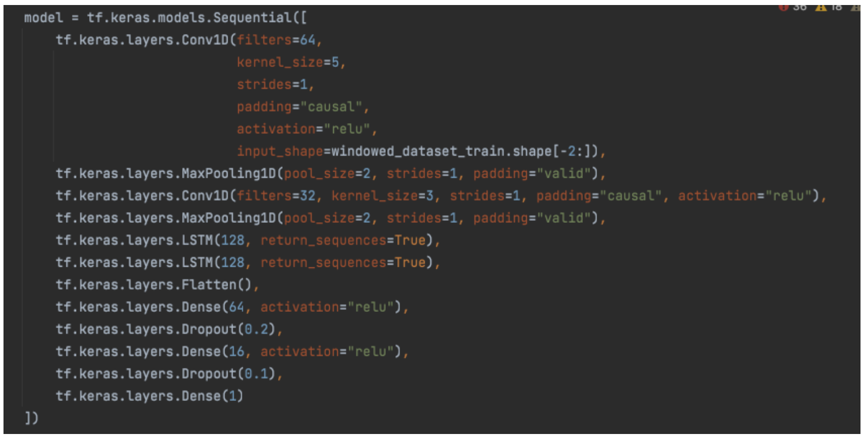
Long short-term memory is a recurrent neural network architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. Although each cell consists of four fully connected layers which means a lot of computation facing a large time series dataset like this, it improves the long-term dependence problem in RNN, and always outperforms of time recurrent neural network. I also tried different structures like different number of layers and neurons.



```
model = Sequential()
model.add(LSTM(input_dim=1, units=50, return_sequences=True))
# model.add(Dropout(0.2))
model.add(LSTM(input_dim=50, units=100, return_sequences=True))
# model.add(Dropout(0.2))
model.add(LSTM(input_dim=100, units=200, return_sequences=True))
# model.add(Dropout(0.2))
model.add(LSTM(300, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(100))
model.add(Dense(units=1))
model.add(Activation('relu'))
tf.config.experimental_run_functions_eagerly(True)
model.compile(loss='mean_squared_error', optimizer='Adam')
model.summary()

model.fit(X_train, y_train, epochs=500, batch_size=1024, verbose=1)
```

CNN-LSTM

Since the results of LSTM does not look so good, I consider adding a CNN to extract features from high dimension data. In my code, I use 64 filters to do the feature extraction and set ReLU as activation function, then use max-pooling layer to pooling, some Dropout layer to overcome overfitting. And use the output as the input of LSTM. SGD as optimizer and MSE as loss function. Attention mechanism is also introduced to the model structure, we sum the output with a weight of each time step of LSTM and output it through softmax layer to obtain an attention coefficient and further get the attention value. Train LSTM by seq2seq and use backpropagation-through-time to propagate error.



```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv1D(filters=64,
                           kernel_size=5,
                           strides=1,
                           padding="causal",
                           activation="relu",
                           input_shape=windowed_dataset_train.shape[-2:]),
    tf.keras.layers.MaxPooling1D(pool_size=2, strides=1, padding="valid"),
    tf.keras.layers.Conv1D(filters=32, kernel_size=3, strides=1, padding="causal", activation="relu"),
    tf.keras.layers.MaxPooling1D(pool_size=2, strides=1, padding="valid"),
    tf.keras.layers.LSTM(128, return_sequences=True),
    tf.keras.layers.LSTM(128, return_sequences=True),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(16, activation="relu"),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(1)
])
```

Rolling Windows MLP

Since MLP performs best among above three models, I also applied rolling window techniques on MLP. Rolling window means using previous window_size days data to forecast the next day. Then there are number of days models need to be fitted and forecasted. The rolling window size is set as 20.

Results and conclusion

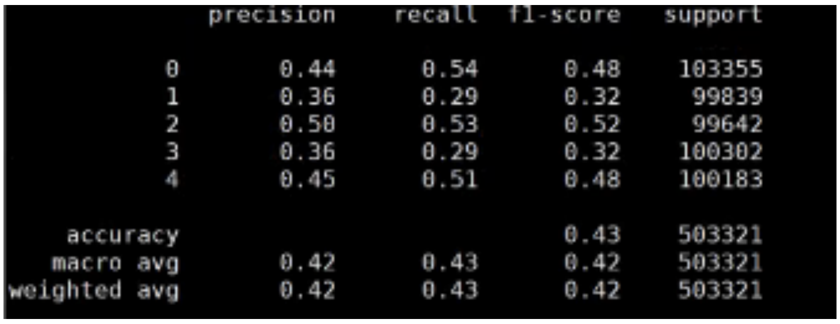
Here are the classification precision results, since the experiments on 0050 I did are most complete, the table below shows its precision.

Model	Class 0	Class 1	Class 2	Time-window
MLP	0.54	0.59	0.55	60
MLP	0.53	0.60	0.53	90
MLP	0.53	0.57	0.53	120
MLP	0.51	0.57	0.52	150
MLP	0.50	0.56	0.52	300
MLP	0.55	0.53	0.50	600
LSTM	0.57	0.55	0.51	60
LSTM	0.53	0.55	0.53	120
LSTM	0.51	0.56	0.49	300
CNN-LSTM	0.55	0.57	0.57	60
CNN-LSTM	0.51	0.59	0.52	120
CNN-LSTM	0.53	0.56	0.52	300
RollingWindow	0.59	0.61	0.57	60
RollingWindow	0.55	0.63	0.57	90
RollingWindow	0.54	0.63	0.53	120

RollingWindow	0.55	0.59	0.51	150
RollingWindow	0.54	0.62	0.55	300
RollingWindow	0.53	0.53	0.50	600

Conclusions

Five classes classification was also tried, using MLP and set percentile [20*(i-1) th, 20*(i) th] as the class i. Here is the classification precision of time window 60 of 0050, a satisfactory classification result could also be seen.



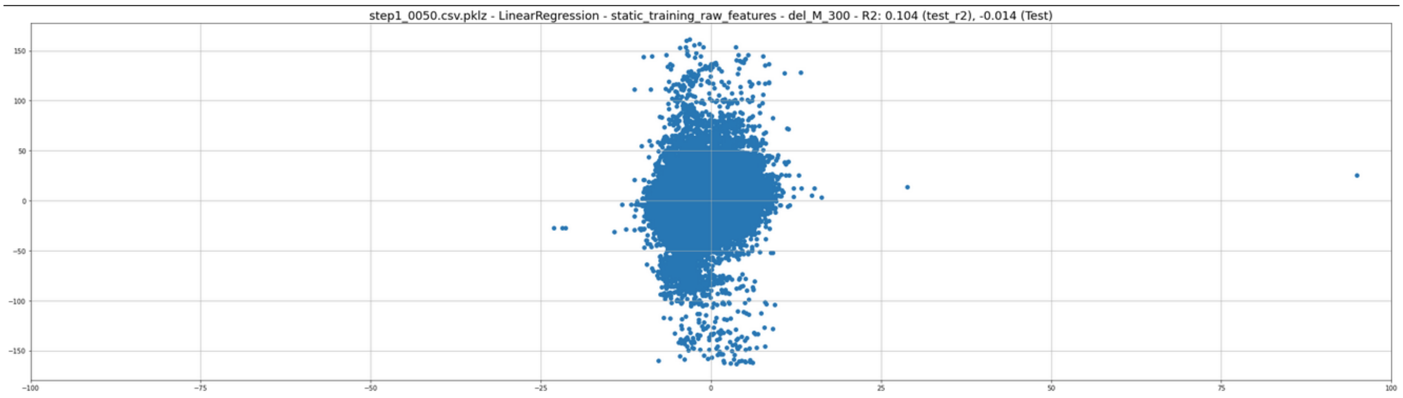
```
              precision    recall  f1-score   support

           0       0.44      0.54      0.48    103355
           1       0.36      0.29      0.32     99839
           2       0.50      0.53      0.52     99642
           3       0.36      0.29      0.32    100302
           4       0.45      0.51      0.48    100183

    accuracy                           0.43    503321
   macro avg       0.42      0.43      0.42    503321
weighted avg       0.42      0.43      0.42    503321
```

Classification precision becomes larger as time window gets smaller. And although LSTM, CNN-LSTM is more complex with much more parameters, it doesn't outperform MLP. The reason may could be conduct to I didn't fine tune the parameters and model structure are not well enough. And when different rolling window size applied in different stocks, it shows different patterns. For the rolling window size, increase proper size within limit will bring a better model. As the conclusion, the rolling window MLP with window size 20 fitting 60 seconds return best. In further research, I could try more models on different stocks to see their effects.

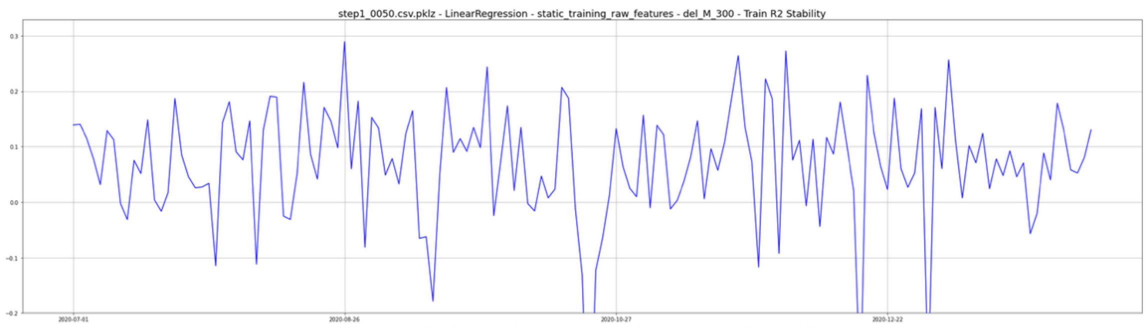
Regression (Anthony WAH)

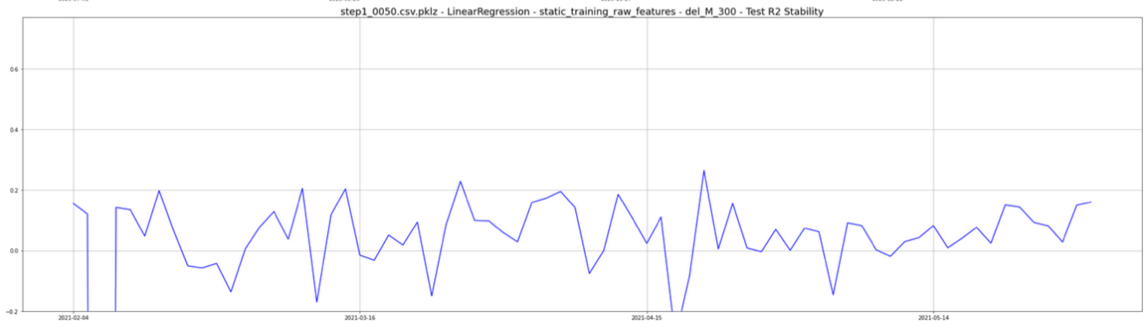
We have tried to apply the Ordinary Least Square (OLS), Lasso Regression and Ridge Regression model to the problem, on each stock respectively. The 280 input features of each stock are defined as the independent variables, while the forward average returns of different time length are defined as the dependent variables and will be run in different setups. In the below passage, OLS model will be used for visual illustration. Graphs of other models can be generated from the code provided and the instructions are given in the readme document.

In the initial setting, we split the data into 60% training set (first 60% of days in the data set) and 40% testing. After fitting the training set data into the model, we computed the prediction for the testing set data using the trained model and the result is shown below. In this example figure, stock 0050 and forward 300s average returns is used. R-square is 0.104 for training set and -0.014 for testing set, meaning that the model failed to explain the testing data after fitting the training data into the model.



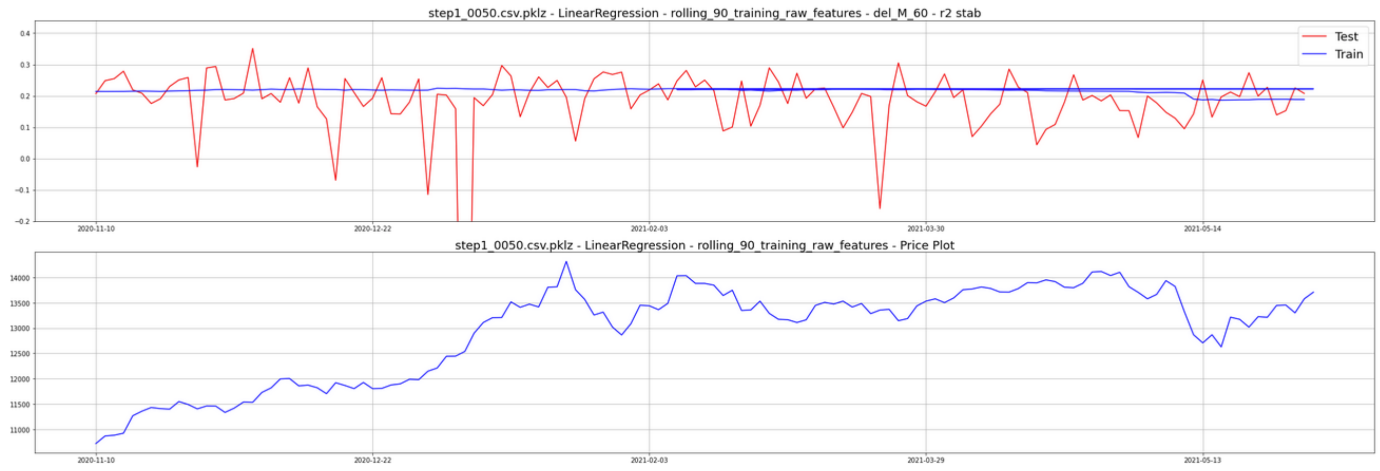
The testing R-square is not showing a significant result. We may also breakdown the R-square by applying the trained model to each day in the testing set and see how the R-square varies over time:





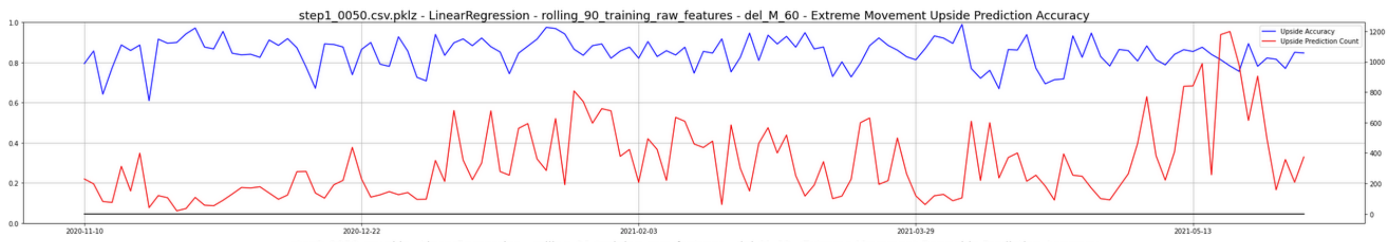
Again, in the training it may seem that the R-square is revolving around 0 – 0.2 with occasional downward spike to below 0, but the figure goes below 0 much more frequently when it comes to the testing set. It shows that the model is effective in explaining the forward returns, but it may suffer from overfitting problem which causes a lower R-square in the current setup.

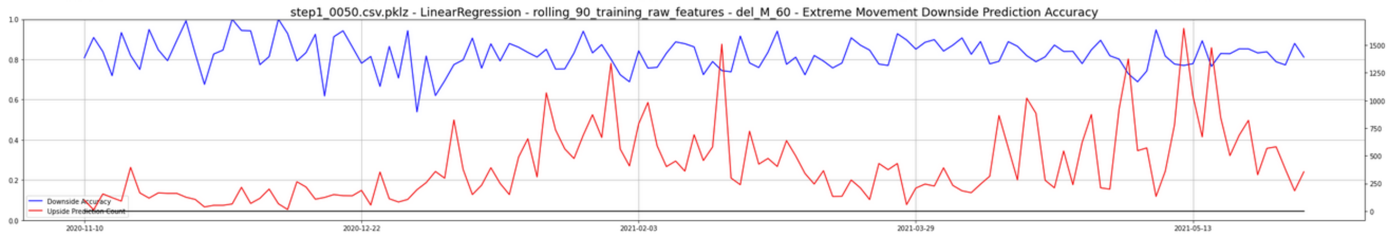
In the next step, we applied the rolling dates approach introduced in the methodology section. The stability of prediction has improved significantly in terms of both the R-square and the stability. In this example figure, stock 0050 and forward 60s average returns are trained with a rolling window of 90 days. Also, we tried to include the futures data for training in this part.

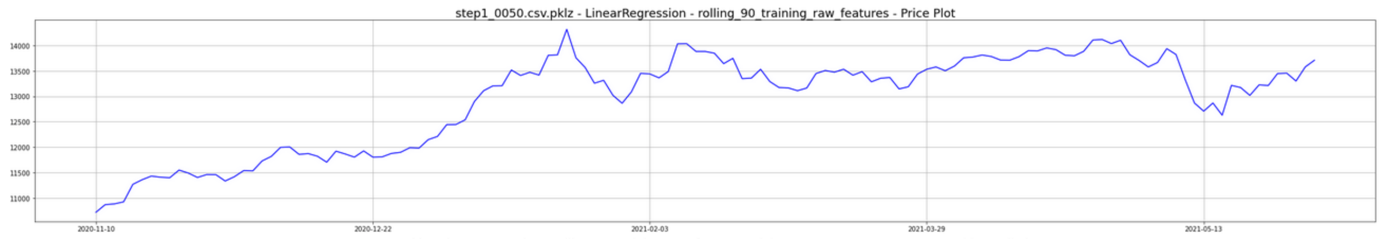


It is easily observed that the training set R-square has improved much more as it is staying above 0.2 all the time throughout the period. Also, testing set R-square may seem to have spikes downwards as usual, but it can be observed that in most of the time it is revolving around 0.2 and 0.3. As a result, we can conclude that training the model with a rolling timeframe outperforms training the model statically.

In the last step, we will try to modify the regression problem into a classification problem. We will try to see if the extreme predictions on the testing set turns out to be positive/negative returns that match our expectation. For instance, we will pick the x-th and (100 – x)-th percentile returns from the training set and define it as the extreme prediction (say, if x=15, 85th percentile returns in the training set = 4.5 bps will be selected and any prediction in the testing set that is > 4.5 bps will be predicted as positive returns). In the example below, stock 0050 and forward 60s average returns are trained with a rolling window of 90 days, and we have selected 15th and 85th percentile as the negative and positive returns boundary.







The upper graph represents positive predictions and the lower one represents negative predictions. The blue line is the prediction accuracy while red line is the prediction count on that day, meaning number of predictions classified as extreme market predictions. We can easily tell from the graph that both positive and negative predictions maintain a high accuracy at around 80% over the entire period. Moreover, by comparing the prediction accuracy and the price graph of 0050 over the period, we can see that the satisfactory performance is stable under both uptrend, downtrend and oscillating markets.

Conclusion

We have tried to apply OLS model to a static training problem with a 6-4 train-test split ratio, as well as a rolling training problem with 90 days rolling window. Apart from evaluating the R-square performance, we have also calculated the prediction accuracy of extreme market predictions. It is concluded that among all approaches under OLS framework, extreme prediction works the best as it can accurately predict the directly of the market in the short-term (60s), with a high accuracy of ~80% under the rolling training model. It will help a lot in building short-term trading strategies when it comes to directional strategies in this sense.

Results of OLS/Lasso/Ridge regression, other stocks, forward returns and rolling window size can be generated and visualized according to the instructions detailed in the readme file.

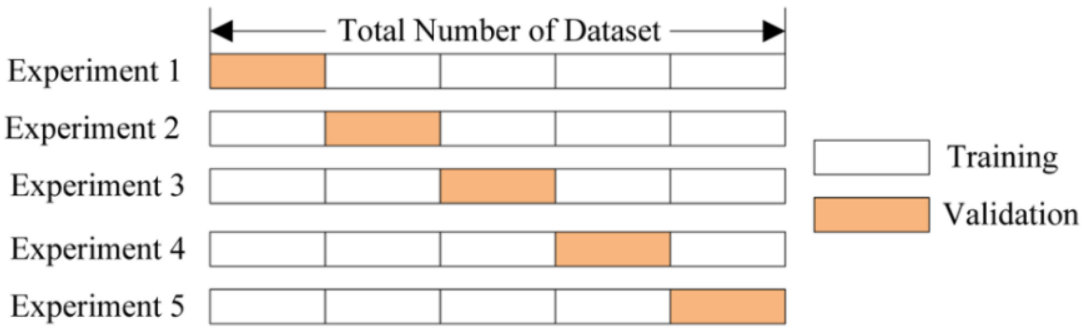
Random Forest & Gradient Boosting (Howard CHENG)

Random forest and Gradient Boosting Prediction with application of PCA

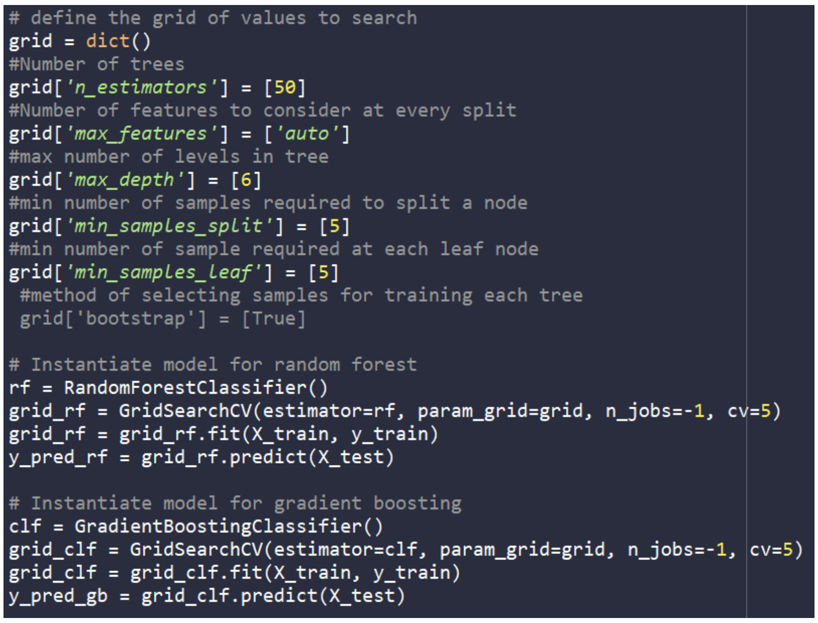
For training and testing random forest and gradient boosting models, 480 features in total from our stock and stock futures data are transformed to 10 principal components used to train and test the random forest and gradient boosting stock return prediction model in rolling basis: first 6 days data are used for training and validation, while the 7th day data is used for testing. In each trial, principal component analysis is performed on the first 6th days' data after standardization to reduce dimensions of features from 480 to 10 and obtain a principal component analysis model to be used to transform the standardized test data on the 7th day. The training data features transformed into 10 principal components will be used as input variables to train random forest and gradient boosting model for each period of returns. In this section for random forest and gradient boosting model, only 60-seconds, 300-seconds and 900-seconds periods of stock return are chosen to represent short-term, median-term and long-term stock return. In order to construct a classification problem, each day return is classified into 3 classes, where class 0, 1 and 2 respectively represents those return lower than 30 percentiles, between 30 percentiles to 70 percentiles and higher than 70 percentiles of the returns on a day.

Model configuration for both random forest and gradient boosting model are set as below:
n_estimators = 50, which is the number of trees in the forest
max_features = auto, which suggests max number of features considered for splitting a node. "auto" in here means no feature subset selection is performed in the trees.
max_depth = 6, which is max number of levels in each decision tree
min_samples_split = 5, which is min number of data points placed in a node before the node is split
min_samples_leaf = 5, which is min number of data points allowed in a leaf node

Cross validation (CV) technique is used in training the random forest and gradient boosting models. Approach a machine learning problem, we split our data into a training and a testing set. In K-fold cross validation, training set is further split into K number of subsets, called folds. The model is then fitted by K times iteratively, where the data is trained on K-1 of the folds each time and evaluate on the Kth fold. After k-times of the training, the performance on each of the folds is averaged to come up with final validation metrics for the model. 5-fold cross validation technique is adopted to train our models.



5 Fold Cross Validation (Source)



```
# define the grid of values to search
grid = dict()
#Number of trees
grid['n_estimators'] = [50]
#Number of features to consider at every split
grid['max_features'] = ['auto']
#max number of levels in tree
grid['max_depth'] = [6]
#min number of samples required to split a node
grid['min_samples_split'] = [5]
#min number of sample required at each leaf node
grid['min_samples_leaf'] = [5]
 #method of selecting samples for training each tree
 grid['bootstrap'] = [True]

# Instantiate model for random forest
rf = RandomForestClassifier()
grid_rf = GridSearchCV(estimator=rf, param_grid=grid, n_jobs=-1, cv=5)
grid_rf = grid_rf.fit(X_train, y_train)
y_pred_rf = grid_rf.predict(X_test)

# Instantiate model for gradient boosting
clf = GradientBoostingClassifier()
grid_clf = GridSearchCV(estimator=clf, param_grid=grid, n_jobs=-1, cv=5)
grid_clf = grid_clf.fit(X_train, y_train)
y_pred_gb = grid_clf.predict(X_test)
```

With the random forest model and gradient boosting model trained with the first 6 days data, the models are tested with the 7th day's data, which have been transformed using the principal component analysis model trained in the previous step. The prediction performance for training and testing data are represented in the metrics of precision, recall, support and F1 score. The stability of principal component analysis is also examined by casting the 7th day's data to the principal component analysis model trained with the first 6 days' data.

Results

From the classification performance statistics summarized as below, it is shown that gradient boosting model demonstrates better performance in general and the models' performance diverges for different time window of return. Several window of time of return (60s, 300s and 900s) is chosen to represent short-term return, medium-term return and long-term return.
For stock 0050.TW and 2603.TW, it is shown that both random forest model and gradient boosting exhibit the best performance for predicting short-term return, where slightly higher precision and recall for three classes are observed for gradient boosting model.

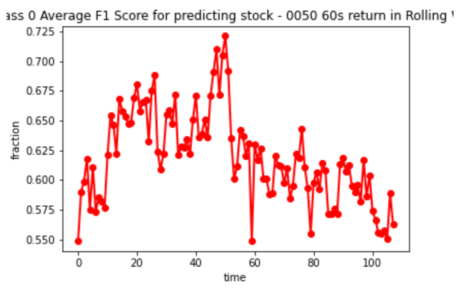
For stock 2330.TW, it is shown that both random forest model and gradient boosting exhibit the best performance for predicting medium-term return, where slightly higher F-score (harmonic mean of the model's precision and recall) for three classes are observed for gradient boosting model.

		0050.TW								
	Window of Time for Return Prediction	60s			300s			900s		
	Class	0	1	2	0	1	2	0	1	2
Gradient Boosting	Precision	47.02%	46.89%	48.06%	41.57%	42.09%	42.36%	34.43%	42.50%	35.40%
	Recall	47.80%	46.12%	48.14%	38.79%	46.78%	38.81%	29.88%	51.25%	29.88%
	F1- Score	47.24%	46.36%	47.94%	39.88%	44.14%	40.27%	31.63%	46.27%	32.10%
	Support	2701	3739	2713	2738	3672	2743	2687	3756	2710
Random Forest	Precision	48.81%	45.72%	50.03%	43.62%	42.03%	44.83%	36.39%	42.33%	36.77%
	Recall	42.37%	53.75%	43.88%	33.01%	57.42%	33.44%	21.71%	67.15%	21.64%
	F1- Score	44.96%	49.19%	46.27%	37.00%	48.30%	37.60%	26.21%	51.64%	26.44%
	Support	2701	3739	2713	2738	3672	2743	2687	3756	2710

		2330.TW								
	Window of Time for Return Prediction	60s			300s			900s		
	Class	0	1	2	0	1	2	0	1	2
Gradient Boosting	Precision	37.57%	79.92%	36.49%	40.62%	53.98%	39.99%	37.48%	44.59%	36.68%
	Recall	19.85%	90.69%	18.10%	42.23%	51.88%	40.50%	38.51%	41.97%	38.85%
	F1- Score	24.17%	84.56%	22.27%	40.37%	52.21%	39.15%	37.47%	42.76%	37.34%
	Support	1522	10628	1559	3447	6821	3441	3907	5909	3894
Random Forest	Precision	26.33%	78.10%	20.11%	45.68%	53.41%	46.02%	39.27%	44.44%	39.00%
	Recall	3.19%	98.61%	2.69%	33.99%	65.66%	31.00%	34.03%	53.41%	32.22%
	F1- Score	4.59%	86.76%	3.74%	35.91%	57.98%	33.40%	35.54%	47.96%	34.52%
	Support	1522	10628	1559	3447	6821	3441	3907	5909	3894

		2603.TW								
	Window of Time for Return Prediction	60s			300s			900s		
	Class	0	1	2	0	1	2	0	1	2
Gradient Boosting	Precision	40.57%	56.52%	40.88%	36.51%	44.72%	36.93%	33.36%	44.09%	31.11%
	Recall	35.79%	60.48%	38.70%	34.81%	48.40%	34.48%	30.74%	49.85%	27.24%
	F1- Score	37.07%	58.00%	38.86%	35.19%	45.97%	34.85%	31.50%	46.06%	28.33%
	Support	3280	6126	3283	3757	5195	3737	3758	5193	3738
Random Forest	Precision	39.88%	54.70%	43.00%	38.52%	43.82%	38.50%	33.77%	43.09%	31.31%
	Recall	21.08%	76.62%	24.86%	25.98%	64.68%	24.45%	20.86%	67.25%	17.39%
	F1- Score	25.11%	63.38%	28.52%	29.65%	51.53%	28.19%	23.98%	51.42%	20.90%
	Support	3280	6126	3283	3757	5195	3737	3758	5193	3738

Below please find an example graph of stock 0050, Average F1 score of class 0 over time. (The remaining graphs are included in the folder Howard Graphs, attaching all graphs will overload the document).



Conclusion

In this report, we have tried 3 different main approach in tackling the short-term price forecasting problem, namely neural networks, regression, and random forest. Neural networks and random forest are classification problems while OLS and other linear models are regression problems. From the above analysis, we can arrive several important conclusions:

- Rolling training approach can predict the price movement much more accurately and the accuracy is more stable over the period.
- Prediction on short time frame (60s forward returns) tends to be more accurate than that of longer time frame.

However, to compare the accuracy among all the 3 models, we may need to alter the regression problem by a bit such that it becomes a classification problem. In the regression analysis section, we have shown how it can be done and the result is appealing. We may conclude that a classification solution approached by a regression model outperforms the other 2 models with a stable accuracy of ~80% over the entire period. However, we also need to consider that the regression approach only targeted on accuracy in predicting the sign of market movement given the extreme prediction, while the other 2 models are predicting the exact cluster of the market forward returns. The result may be different if we modify the classification approach of the 2 models to be the same as the regression model. Still, if one wants to build a directional trading strategy using one of the 3 models, OLS model with a classification approach should still be considered as the best model as it can predict the direction of market most accurately in the short term.

Next Steps

Based on the above results, one may build a trading strategy using the predictions from each model. For example, the strategy may long the stock and close the positive evenly across the time window (say, sell the stock evenly over 60s if the prediction on 60s forward average return is positive), and vice versa. After backtesting the result of the strategy built on all 3 models, one may compare the sharpe ratio, winning ratio of trades and average profits of each trade to conclude on the prediction power comparison among all 3 models. Note that transaction cost is not significant in this analysis as the main target is to compare model accuracy.

Also, one may apply the study the prediction power of the 3 models on other assets, say, index futures, forex, and cryptocurrencies. Trading stocks may face a problem of high margin interest rate in shorting the stock when the prediction tells that the forward returns may be negative. However, in forex market, cryptocurrencies or even index futures, both long and short incurs a much lower cost in interest rate and transaction fees as there are other products available for these asset classes, such as CFDs (Contract For Differences), perpetual futures, perpetual swaps … etc. With these advantages, it may imply that construction of trading strategy from these models will be easier since the model result itself can better reflect the realistic profitability with a lower transaction cost.