Capturing stock movement dynamics following earnings data release using Machine Learning techniques

Joseph Ye

This is a project progress report on analysing and predicting stock price movements following the release of a company's quarterly financial statements. The student has observed how different companies during the same earnings season could react drastically differently to very similar earnings data and has set out to use machine learning models to hopefully capture and take advantage of the non-linear relations between near term stock price movements post the data release and a company's financial reports data, by also considering the prevailing market conditions as well as the price momentum of each individual stock.

To achieve such a goal the student has been building an end-to-end machine learning program in Python that performs data pre-processing, feature selection, model hyperparameter tuning, as well as model calibration and prediction. The majority of this program has already been built and this report will present the project progress by giving an overview on each functional part of the program.

This report describes what has been done so far in the project and does not provide in-depth descriptions on publicly known terminologies nor mathematical depiction of certain scenarios.

It needs to be said that the student initially set out to predict a stock's price movement shortly after the release of quarterly reports (stock's opening price after release, and other short term price movements post earnings release). This proved to be not fruitful. The student put this down to the fact that short term stock movements post earnings release can be impacted by a lot of non-numerical factors, such as earnings calls and insider dealings, as well as behavioural factors that potentially go against what our data may otherwise have suggested, such as the typical "buy the rumour, sell the news' kind of behaviours of stock investors.

Model Features Generation

This section explains how the student has chosen his current set of features as inputs to the supervised learning models.

The student has chosen in total 384 S&P500 companies for analysis. The chosen time frame was between the fourth financial quarter of 1996 (1996 Q4) and the second financial quarter of 2018 (2018 Q2). The chosen companies were in continuous operation as well as being a constituent of the S&P500 index during at least half of the chosen time framework. As a result some companies have contributed more to the final population of test data points and others less, although this should not be a problem.

While the outputs of the learning model are the near term stock movements (%change), the input to the models consists of the following sets of data:

- Financial statements data
- Earnings Surprise data
- Price movements data
- Momentum indicator data

Financial Statements data

The following 24 metrics from financial reports have been chosen as training data.

Cash (or equivalent) account	Short Term Debt
Total Equity	Long Term Debt
Total Liability	Long Term Debt / Capital
Net Income	Long Term Debt / Equity
Net Income Available to Common	Long Term Debt / Total Asset
Operating Income	Total Debt / Capital
Operating Margin	Total Debt / Equity
Pre tax Income	Total Debt / Total Asset
Profit Margin	Return on Asset
Revenue	Return on Common Equity
Common Equity / Total Asset	Earnings per share (adjusted and diluted)
Net Debt / EBIT	Income Tax Expense

These metrics have been preprocessed in order for them to make sense to the learning models and new features are created through feature engineering.

Specially:

a. According to the principle of *Vertical Analysis*, certain items from quarterly financial statements should be represented as a percentage of the same quarter's *asset* or *revenue* level. Such a simple technique determines the relative weight of each item and its share in asset resources or revenue generation.

   The student has normalised all items from the Income Statement and Cashflow Statement against *revenue* and normalised all items from the Balance Sheet against *asset*;

b. Instead of using the (processed) financial reports data directly as model inputs, the student calculate
   (a) the change in value of the 24 financial metric from the quarter before (quarterly change), as well as,
   (b) change in metric values from the same quarter the year before (yearly change).

   Most of the metrics have their differences calculated through simple subtraction by metric value of the previous quarter; the only exception is with *Revenue* whose difference is calculated as %change from a previous quarter.

   Therefore, 48 features (24 representing metrics' quarterly changes and 24 representing metrics' yearly changes) have been constructed directly from financial reports metric data:

Cash (or equivalent) account_Q_change	Short Term Debt_Q_change
Total Equity_Q_change	Long Term Debt_Q_change
Total Liability_Q_change	Long Term Debt / Capital_Q_change
Net Income_Q_change	Long Term Debt / Equity_Q_change
Net Income Available to Common_Q_change	Long Term Debt / Total Asset_Q_change
Operating Income_Q_change	Total Debt / Capital_Q_change
Operating Margin_Q_change	Total Debt / Equity_Q_change
Pre tax Income_Q_change	Total Debt / Total Asset_Q_change
Profit Margin_Q_change	Return on Asset_Q_change
Revenue_Q_change	Return on Common Equity_Q_change
Common Equity / Total Asset_Q_change	Earnings per share (adjusted and diluted) _Q_change
Net Debt / EBIT_Q_change	Income Tax Expense_Q_change
Cash (or equivalent) account_Y_change	Short Term Debt_Y_change
Total Equity_Y_change	Long Term Debt_Y_change
Total Liability_Y_change	Long Term Debt / Capital_Y_change
Net Income_Y_change	Long Term Debt / Equity_Y_change
Net Income Available to Common_Y_change	Long Term Debt / Total Asset_Y_change
Operating Income_Y_change	Total Debt / Capital_Y_change
Operating Margin_Y_change	Total Debt / Equity_Y_change
Pre tax Income_Y_change	Total Debt / Total Asset_Y_change
Profit Margin_Y_change	Return on Asset_Y_change
Revenue_Y_change	Return on Common Equity_Y_change
Common Equity / Total Asset_Y_change	Earnings per share (adjusted and diluted) _Y_change
Net Debt / EBIT_Y_change	Income Tax Expense_Y_change

Note:

Initially a larger set of financial report metrics had been chosen and used. This included other commonly known metrics such as Price/EPS ratio, Price/Book ratio, etc. However, some metrics were dropped because they had suffered from missing data. *Handling missing data is on its own a technical subject to be considered and possibly explored.*

Earnings Surprise data

Earnings Surprise represents how much a company's actual reported Earnings Per Share (EPS) is more (or less) than the average of a selected group of stock analysts' estimates on that quarter's EPS: **reported EPS – market estimated EPS**.

The student initially calculated Earnings Surprise as %change between reported EPS and market estimated EPS. This had turned out to be problematic because (a) %change was too volatile as a very small change when the actual EPS levels were close to zero would lead to a large %change, and (b) it was not easy to deal with the change of signs which is a common problem.

Note:
From a more mathematically rigorous point of view, we should find the metric value difference of two adjacent quarters as absolute difference if the metric values follow a normal distribution where as we should use %change instead if the metric values follow lognormal distribution. However the student doesn't think such a consideration matters a lot to the machine learning model and instead has chosen to pay more attention the more practical problems such as the two points outlined above.

The three features constructed by the student under this category are:

- Current quarter's Earnings Surprise (reported EPS – market estimated EPS);
- Difference between current quarter's Earnings Surprise and that of the previous quarter;
- Difference between current quarter's Earnings Surprise and the average Earnings surprise of the preceding three quarters;

Note:

The student has also prepared for Guidance data which represents how a company's management would expect the company to perform in the coming quarters. Guidance data is currently not used as model input because they are too sparse.

Price Movement data

The student is currently forecasting 60-day forward movement of a given company's stock price from a start date shortly after data release. The chosen start date can vary depending on the student's program's settings.

The following data related to recent stock movements have been constructed as the model input features:

- %change from a stock's closing price prior to data release, to the stock's opening price immediately after data release;
- %change from a stock's opening price immediately after data release, to the stock's closing price on the chosen forecast start date (price forecast can be chosen to start 2 days after data release)

- %change from S&P500 index’s closing price prior to data release, to S&P500 index’s opening price immediately after data release;

- %change from S&P500 index’s opening price immediately after data release, to the S&P500 index’s price on the chosen forecast start date;

The purpose of including stock movements around the time as well as shortly after the release of earnings data is to provide information to the learning models about the market’s immediate reactions after data release.

The purpose of including the movement of S&P500 market index around the time as well as shortly after the release of earnings data is to help paint a picture to the models on the health of the general market as well as the prevailing collective mood of the investor population.

Momentum Indicators

For each quarter of every company, the following momentum indicator values have been calculated on the same day the financial reports were released

- 9-day Relative Strength Index (RSI)
- 30-day Relative Strength Index
- 5-day Moving Average / 50-day Moving Average
- 5-day Moving Average /200-day Moving Average
- 50-day Moving Average / 200-day Moving Average

All these indicators should in a way measure how a stock’s recent short term movements compare to its historical movements further back in time.

The inclusion of momentum indicators is to allow the prediction process of future stock movements to take into account a stock’s recent movement trend.

Other Economic Data

The student has partially collected and planned to use the following data to add to the input features as part of future work:

- Stocks’ trading volume data

- More momentum index data for individual stocks

- Momentum index data of broad market indexes such as S&P 500

- Momentum index data of stock index for stocks of a particular sector (such as healthcare stock index)

- 3 month US Treasury Bill yield vs 10 year US Treasury Note yield

Data Pre-processing

By the start of data preprocessing, all the training data has been collected and put into a matrix-like data structure where each row represents an n-dimensional training data point, indexed by a company name and a historical quarter name, whereas each column represents the data of a feature from all the companies. It should look something like this:

	Feature 1	**Feature 2**	......	**Feature n**
Apple **2018 Q2**	0.53	0.73		0.94
Apple **2018 Q1**	0.73	0.43		0.4
:	:	:		:
Apple **1996 Q1**	0.5	0.1	......	0.93
JPM **2018 Q2**	0.01	0.45		0.44
JPM **2018 Q1**	0.01	0.45		0.44
:	:	:		:

The two main tasks involved in this stage are (a) to mitigate the impact of extreme outliers present mainly in the financial reports data, as well as (b) to normalise financial reports data of different stocks so that they can be considered comparable from the model's perspective.

Winsorization

Winsorization is employed to reduce the number of outliers present in the financial reports data.

Note:

Winsorization is applied to the data of every financial metric feature at the per-company level. It is not applied to the collective data of all companies.

The principle of winsorization is simple: replace all the data points that are above the 95% percentile with the 95% percentile data point and replace the data points that are below the 5% (1 – 95%) percentile with the 5% percentile data point. However this is only an example. In reality there is no fixed rule out there that suggests the best threshold to use and this is a challenge.

In order to find a threshold that maximises the prospect of purging outliers without overdoing it, the student has come up with the following (possibly novel) scheme:

For the list of data of a certain feature that belongs to one of the companies, sort the data in descending order

⇨ *Start winsorization at 100% percentile and gradually reduce the percentile threshold down to 90% with a step side of 0.1%;*

- ⇨ *For each percentile threshold, perform winsorization on this list of data by replacing both the lower and upper end of the data up to the threshold point. Calculate the standard deviation of the processed data list and record its standard deviation. Repeat until all candidate thresholds have been gone through;*
- ⇨ *Sort the list of standard deviations in descending order;*
- ⇨ *Find the standard deviation value that has the largest difference with the standard deviation value before it.*
- ⇨ *The chosen threshold value corresponds to this standard deviation.*

I will be able to explain on this scheme more clearly in person.

Standardization

Similar to winsorization, data standardization is carried out on each set of company's data, feature by feature.

The rationale behind this processing is that, certain blue chip large cap companies, or certain companies from non-cyclical sectors may have little variation in its financial metrics from quarter to quarter whereas most mid cap growth companies may see the same financial metrics swing up or down regularly. Either case is considered norm for respective companies but when putting data of these companies together and put them under the same feature column they may cause confusion to the models if they hadn't been standardized to similar levels.

The student has decided this step is an optional step and will review its true impact and value when the time is right.

Models

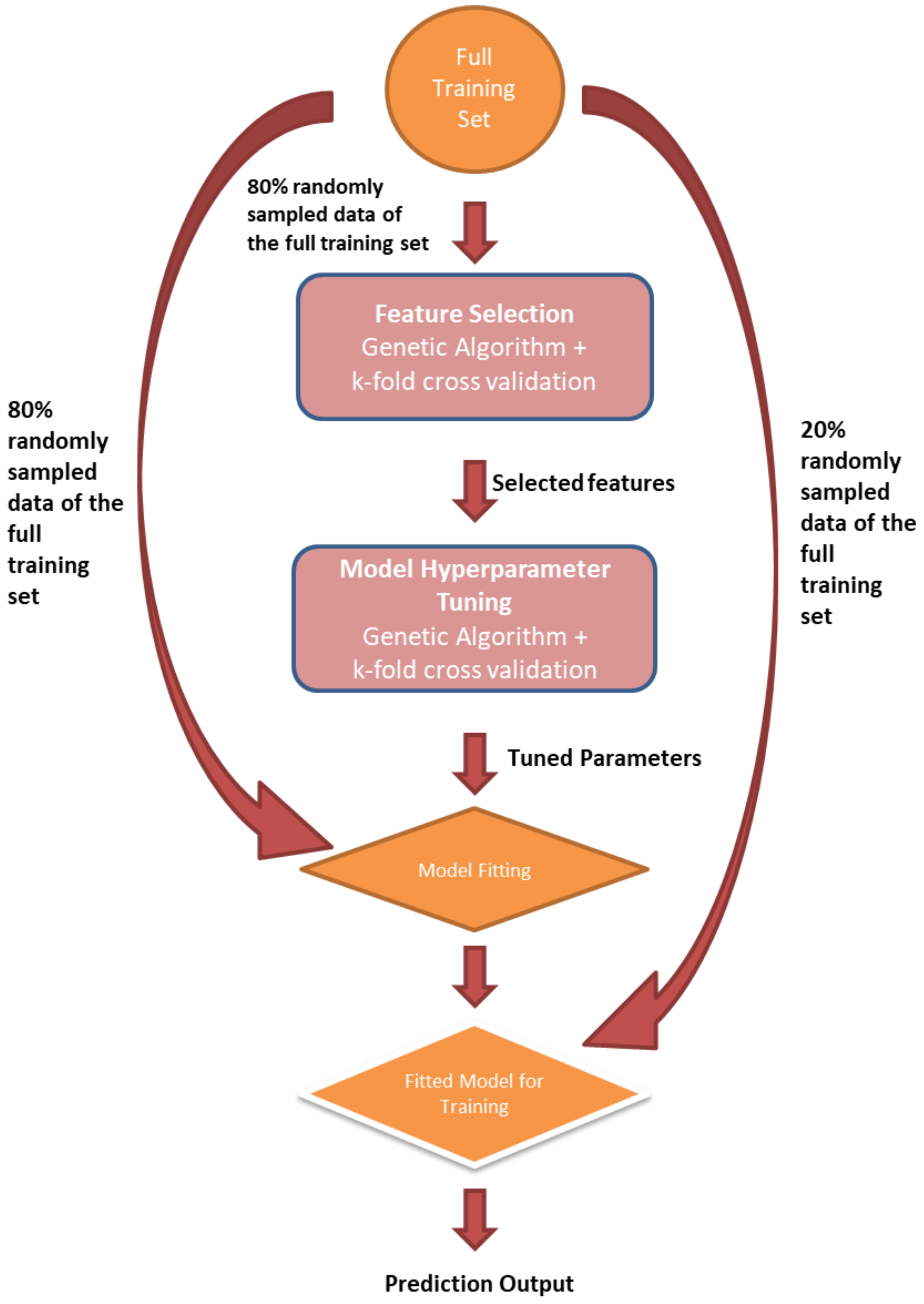
The student has put in place and experimented two supervised learning models so far: Multi-layer Perceptron (MLP) and Extreme Gradient Boosting Trees (XGBoost). The MLP algorithm was obtained from the Teras Python package whereas the XGBoost was obtained from the XGB Python package. It would be interesting to experiment other models should the student find success with these two baseline models.

Both MPL and XGBoost are capable of classification and regression tasks both of which the student has experimented. The student has not managed to perform a detailed comparison on these two models' outputs but based on XGBoost's relative speedy performance and its recent success in Kaggle competitions the student is using XGBoost as the primary model for the time being.

Another key reason XGBoost is primarily used is its ability to work with features of different scales, meaning that it's not necessary to standardize all the features. This is one of the properties of any tree-based learning models.

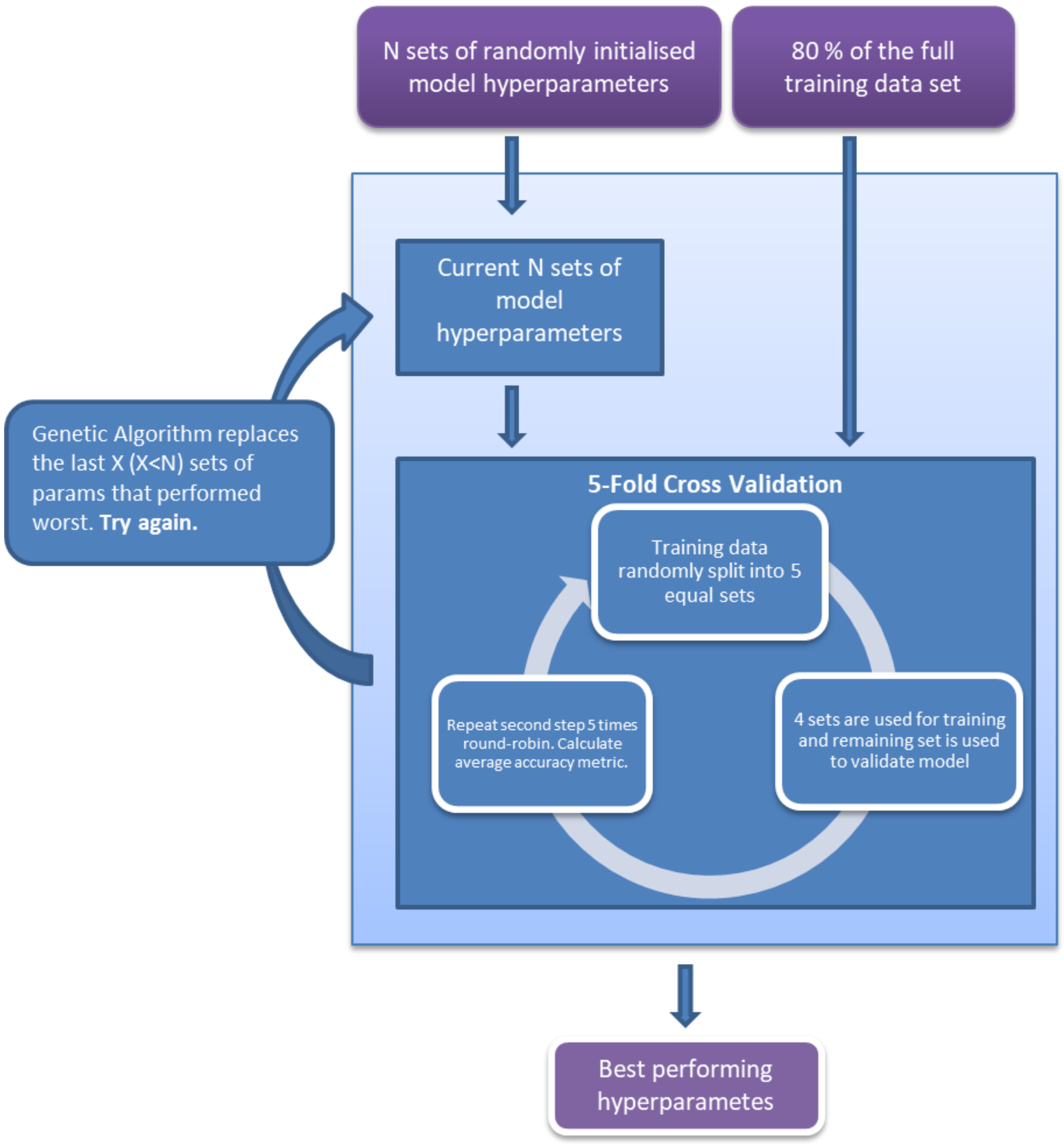
It has been said in the literature that XGBoost also handles missing data well because each node on the decision trees has three branches instead of two. They are left, right, and unknown. That being said, the student does not rely on this property of the model and has not tested its performance in this regard.

Below is a flow chart showing how different components are put together within the model framework:


Full Training Set
80% randomly sampled data of the full training set
Feature Selection
Genetic Algorithm + k-fold cross validation
Selected features
Model Hyperparameter Tuning
Genetic Algorithm + k-fold cross validation
Tuned Parameters
80% randomly sampled data of the full training set
20% randomly sampled data of the full training set
Model Fitting
Fitted Model for Training
Prediction Output

Hyperparameter Tuning

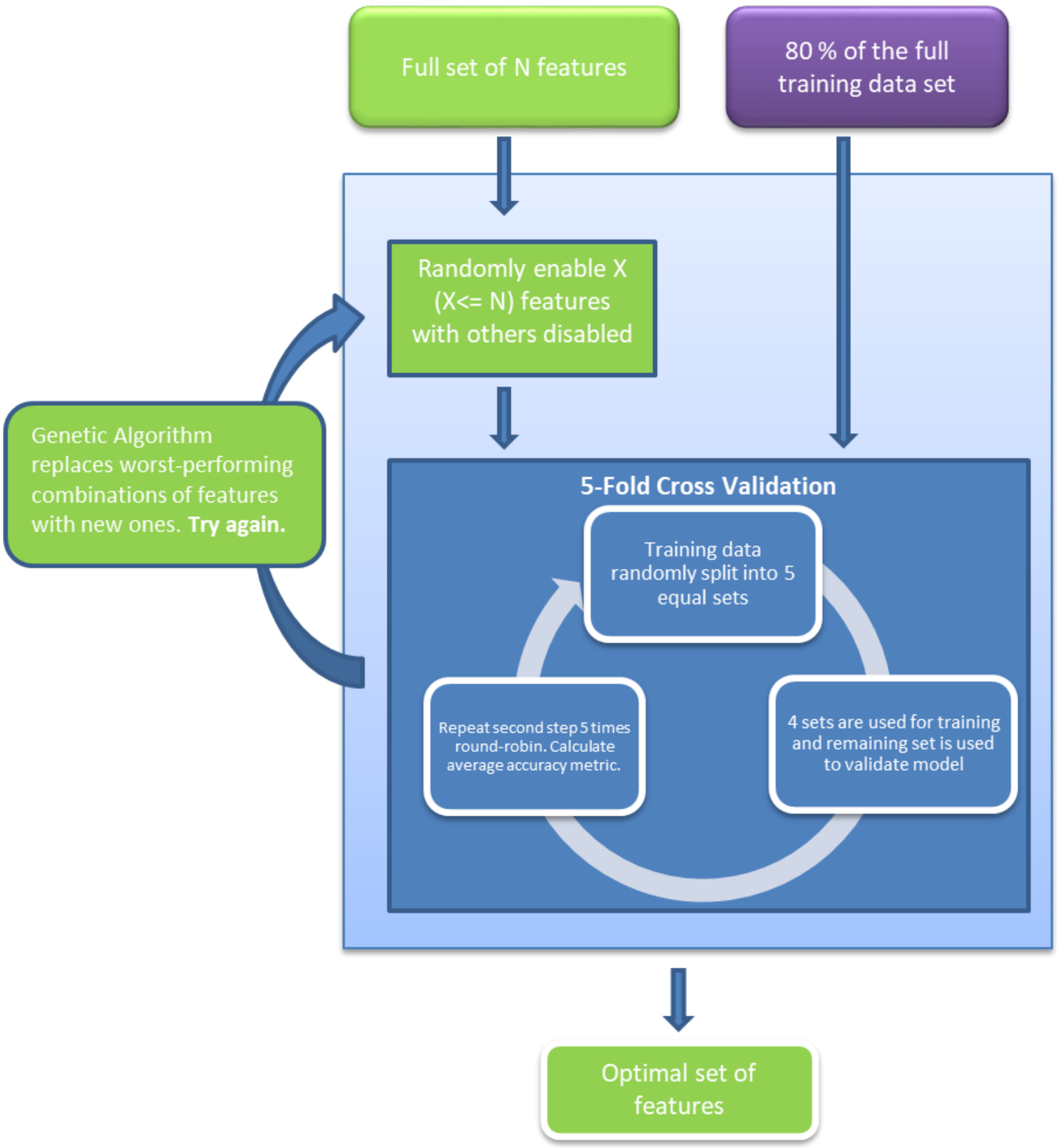
Correctly tuning the hyperparameters of supervised learning models is key to success of any machine learning project. Instead of using brute force methods to try and evaluate every single combination of the model parameters (within ranges), the student is using Genetic Algorithm to quickly and efficiently produce the highest performing set of model parameters. The whole process follows the logic diagram below:


N sets of randomly initialised model hyperparameters
80 % of the full training data set
Current N sets of model hyperparameters
Genetic Algorithm replaces the last X (X<N) sets of params that performed worst. **Try again.**
5-Fold Cross Validation
Training data randomly split into 5 equal sets
4 sets are used for training and remaining set is used to validate model
Repeat second step 5 times round-robin. Calculate average accuracy metric.
Best performing hyperparametes

It needs to be pointed out that only 80% randomly sampled data out of the full training set is used for any tasks to do with model training which includes feature selection, hyperparameter tuning and model calibration. The remaining 20% of data which is to be used as test data has no contact with the model tuning and calibration steps under any circumstances to ensure they are completely unseen by the models.

Feature Selection

Feature selection allows the student to perform dimensionality reduction on the input data. In machine learning having more input features doesn't translate into better performance or accuracy. It's necessary to remove features that are too correlated to other features, or features that are not meaningfully related to the outputs. There are many methods to perform feature selections but the student has chosen to continue using the Genetic Algorithm routines that have already been created for hyperparameter tuning to select meaningful input features. The flow chart below describes how this is done in the program:



Full set of N features
80 % of the full training data set
Randomly enable X (X<= N) features with others disabled
Genetic Algorithm replaces worst-performing combinations of features with new ones. Try again.
5-Fold Cross Validation
Training data randomly split into 5 equal sets
4 sets are used for training and remaining set is used to validate model
Repeat second step 5 times round-robin. Calculate average accuracy metric.
Optimal set of features

Results

So far the student has formulated the project both as a regression problem by predicting the actual 60-day stock movement after earnings data release, and as a classification problem by predicting if stock has gone up or down from the start date's level after 60 days.

Below are test scenarios that the student has conducted on:

- The student has performed predictions using the entire data set of 19,000 data points (spanning 20 years of data with 384 companies), as well as performing the same tests using a subset of these data points all of which belong to the same industry sector. These sectors are: Energy, Financial, Industrial, Technology, Utilities, Basic Materials, Communications, Consumer Cyclical, and Consumer Non-cyclical.

- The student has also performed prediction tests using only data that are from year 2009 and later;

- In the regression setting, in addition to using 60-day stock price changes (stock returns) as output, the student has also tried altering the output data by turning it into 60-day excess returns instead, i.e., the 60-day stock price changes adjusted by the 60-day price change of the S&N500 index over the same period of time;

Overall, the student has achieved very similar results in all the different permutation of scenarios by either using MLP or XGBoost. As a classification problem, the prediction success rate hovers around **55%**. As a regression problem, the prediction results have a Mean Absolute Error (MAE) of **7%**. Such results are by no means spectacular but seem to be coincide with a lot of research results with similar context in the literature [1] [2] [3] although there are also researches that purport to show more superior results [4].

Scenario 1:

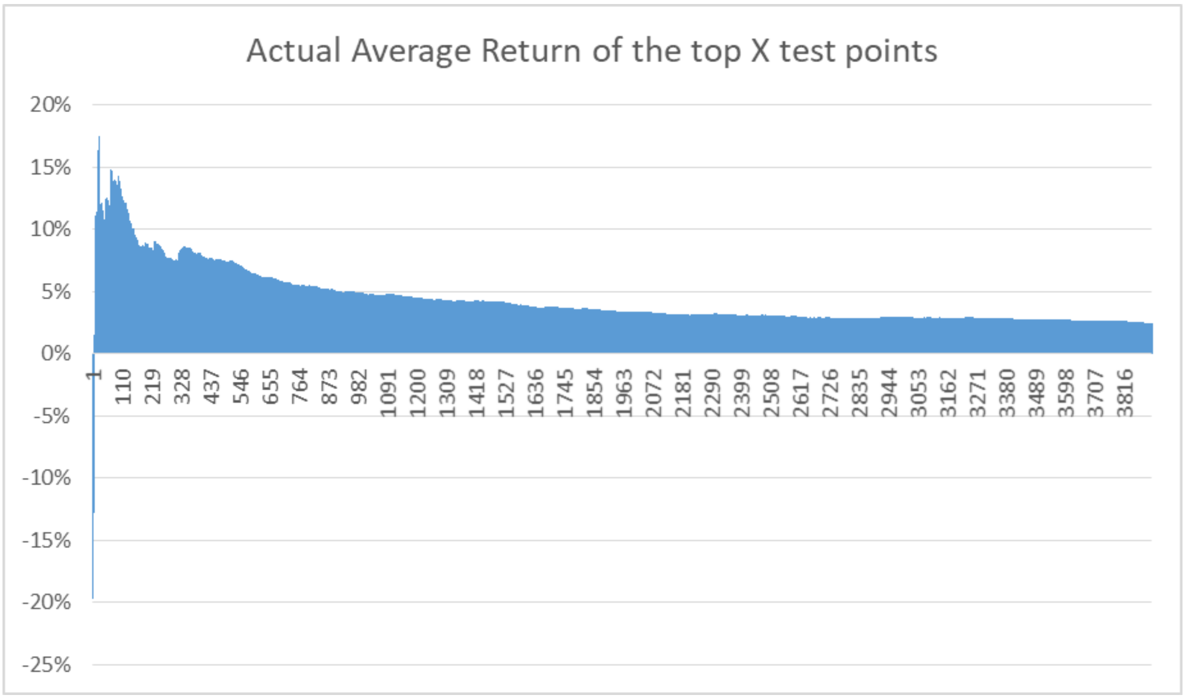
Recently, the student has had an interesting observation on the results of the regression problem:

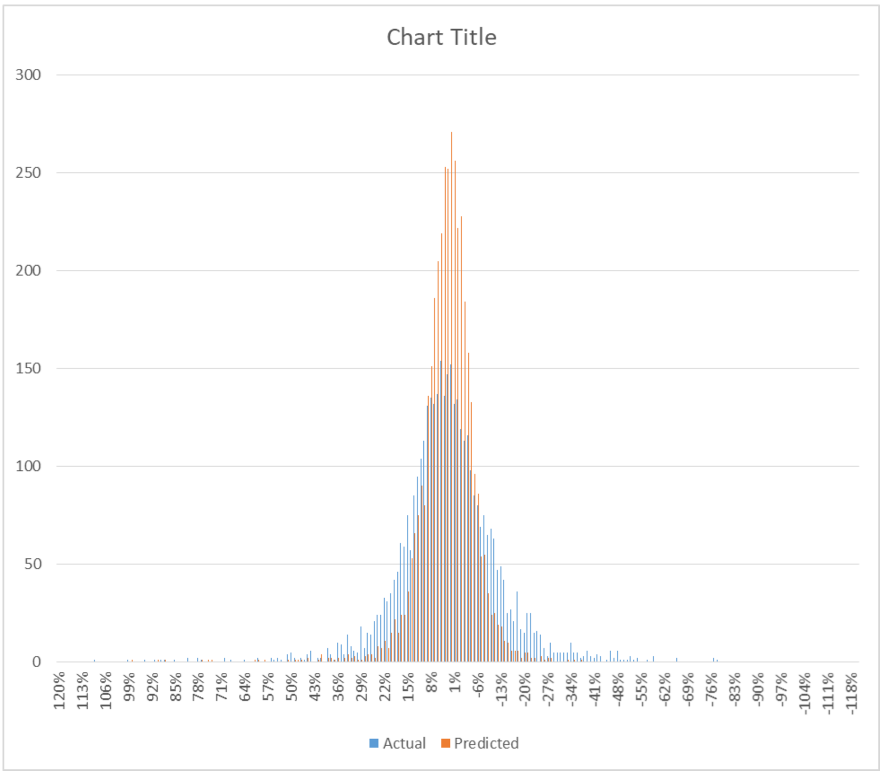
- Firstly, the 20% randomly chosen test data points whose size is N are ranked in **descending** order according to the value of their **predicted outputs**.

- Secondly, compute the **average** of the **actual output** of the first X test points with X increasing from 1 up to N. This is equivalent to forming a number of portfolios out of the top X stocks, with X increasing from 1 up to N, and finding the average return of these portfolios.

- Thirdly, plot a chart on the average actual output of the top X test points, the chart is most of the time in descending order. More importantly, the average actual output of the top x' points where x' is small compared to N, is generally high enough to warrant a buy-and-hold investment.

An example illustration is given in here:





The first chart demonstrates the student's observation outlined above.

The second chart shows the distribution of the actual outputs and that of the predicted outputs. The predictions concentrate a lot around the 0% point resulting in large absolute differences from the actual outputs when compared on a point-by-point basis.

Most of the time, the **R-squared** value between predicted outputs when they are ranked in descending order, and the average actual output of the top X test points (with the test points having been ranked according to predicted outputs) with X increasing from 1 to N, ranges in the region of **0.45 to 0.75**.

This observation seems to suggest that, although the model outputs are poor in predicting the actual 60-day stock movements post data release on a point-by-point basis, they could be somehow capturing the averaged stock movements.

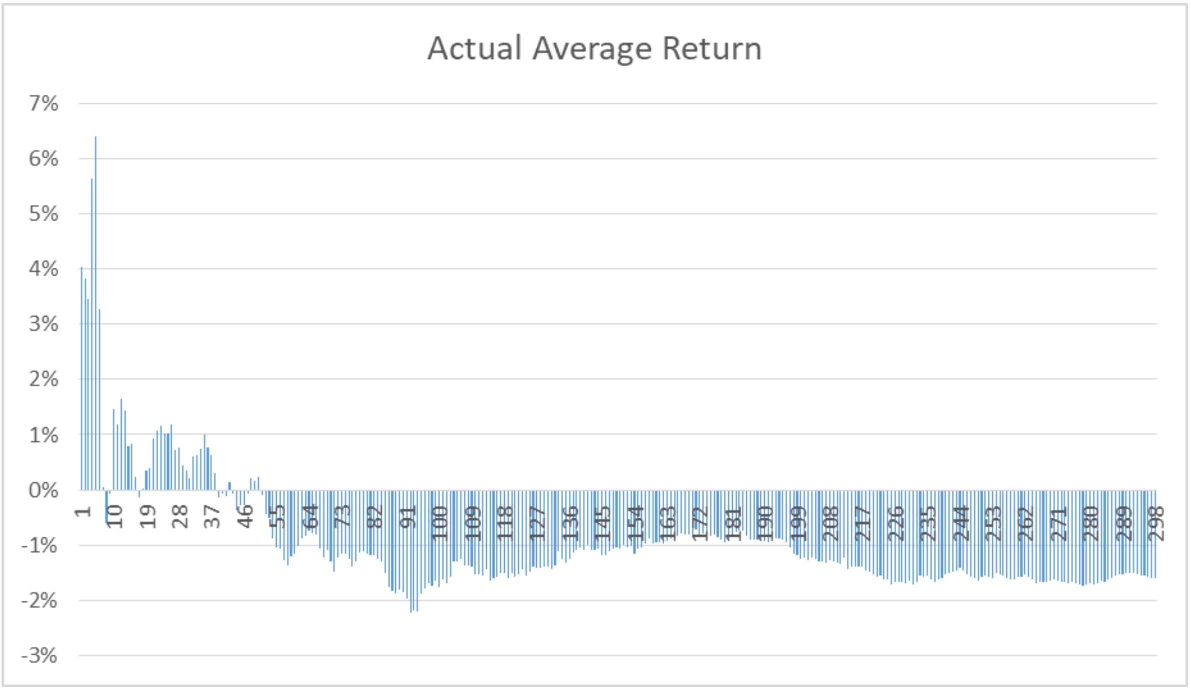
The student has not been able to explain this phenomenon or whether it was real or not.

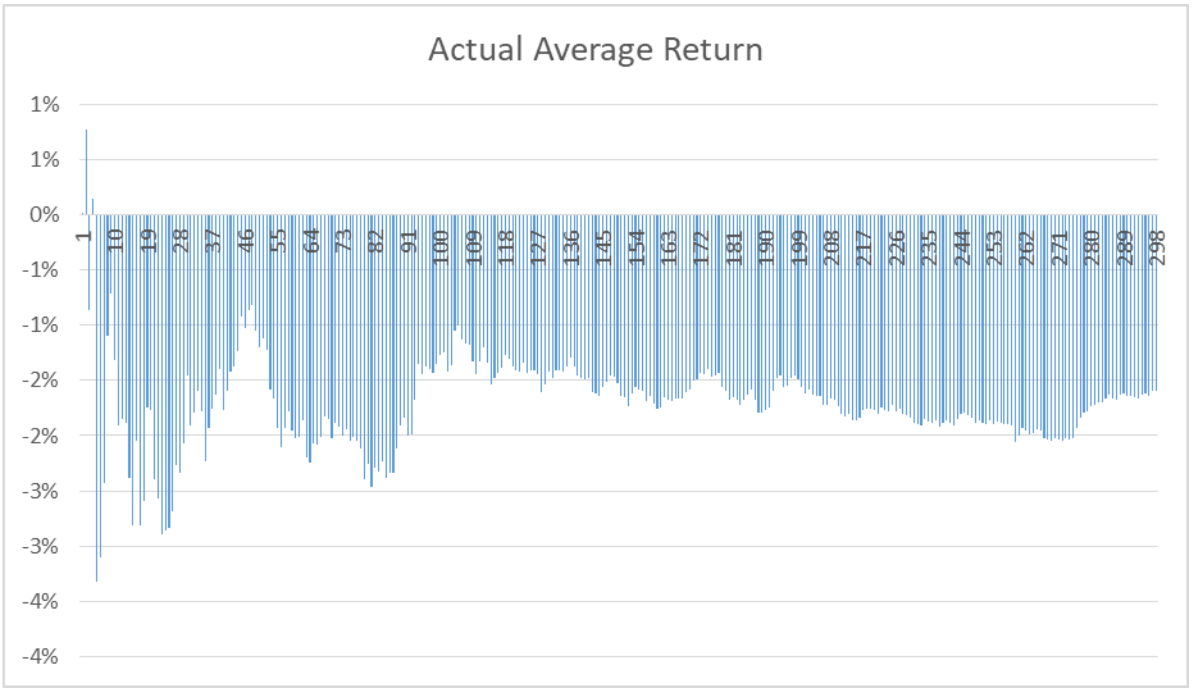
Scenario 2:

Following Walter's recent suggestion, the student has performed the same prediction exercise from a different perspective.

Instead of randomly partitioning the full training set on an 80/20 basis for training and testing, the student has selected all the data that belong in Q2 2018 and put them aside as test data; all the data that do not belong in Q2 2018 (the rest of all the data) are randomly shuffled first and then used to train the model. Such an exercise is aimed at replicating a scenario that is more close to real life applications.

Somehow, the student has so far not been able to replicate the same kind of pattern as he did in the earlier exercises. Below are two example charts created out of two separate runs of modelling:





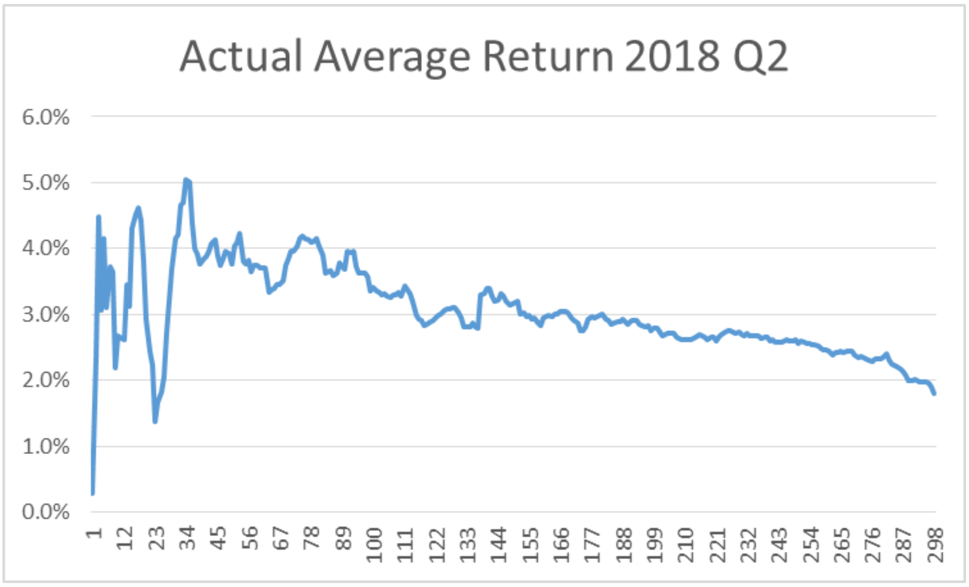
Scenario 3:

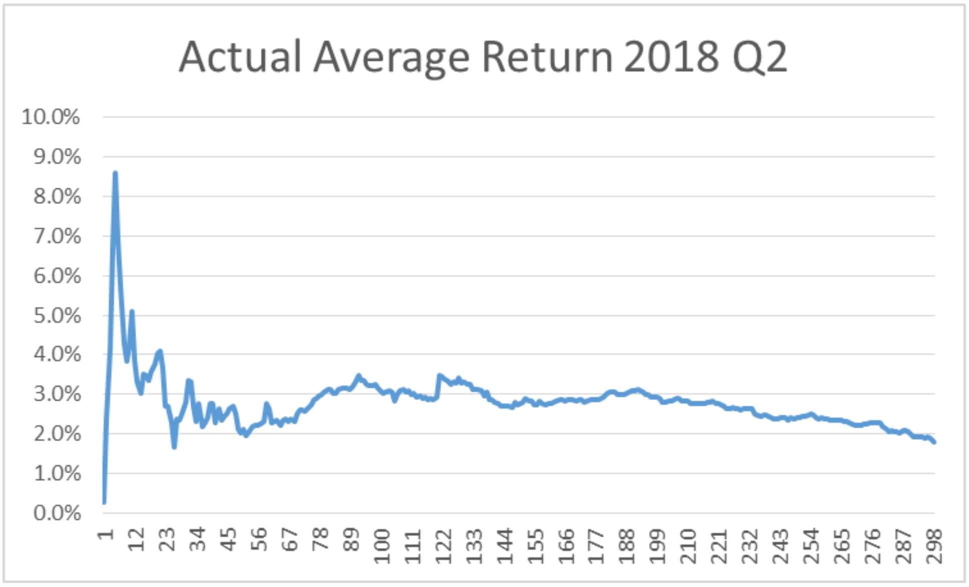
Recognising the fact that the student is trying to capture an underlying distribution of the data using the predicted outputs, he has suspected that the poor results in Scenario 2 might be due to the small size of the ***test population***. As a result, the student has made a small modification on how the training and test data set is organised:

a. As in Scenario 2, the student no longer randomly samples data into the training and test set from the full data set of 22 years and instead only chooses 'future data' as the test set.
b. The last quarter in the full training data population is 2018 Q2 and this quarter has been chosen as the 'future data'.
c. Unlike in Scenario 2, the student has chosen data from 2016, 2017, and 2018 as the test set, and all the data prior to 2016 as the training set.
d. Once predictions have been carried out over all the test data of 2016, 2017 and 2018, those predictions that belong in 2018 Q2 are extracted and analysed specifically.

Same as in the previous scenarios, all test data are ranked according to their predicted stock price changes in descending order, and the average of the real stock price changes of the top N data points are calculated and plotted.

Below are two plots of the Average Stock Returns of the top N stocks created in two different runs, with all the data points (companies) belonging in 2018 Q2.



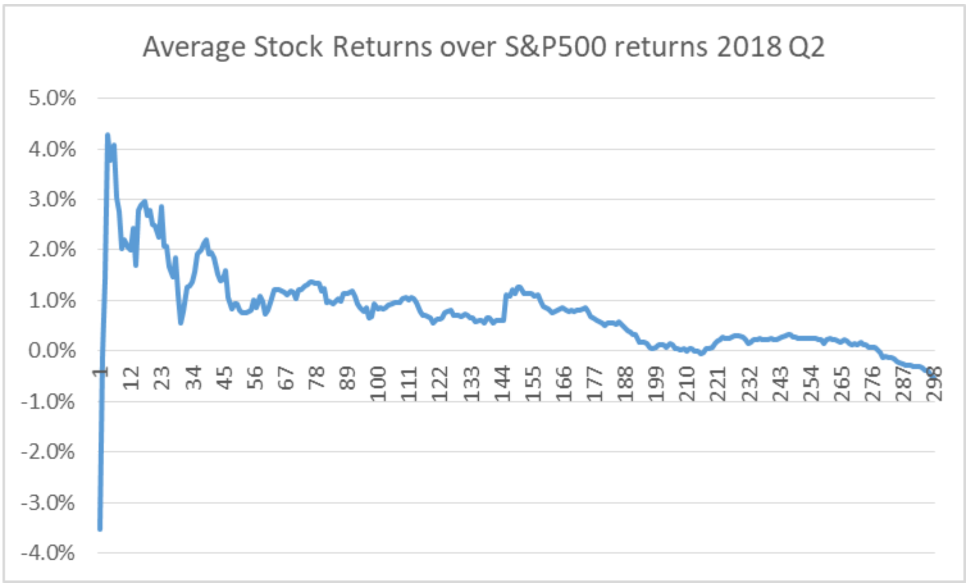


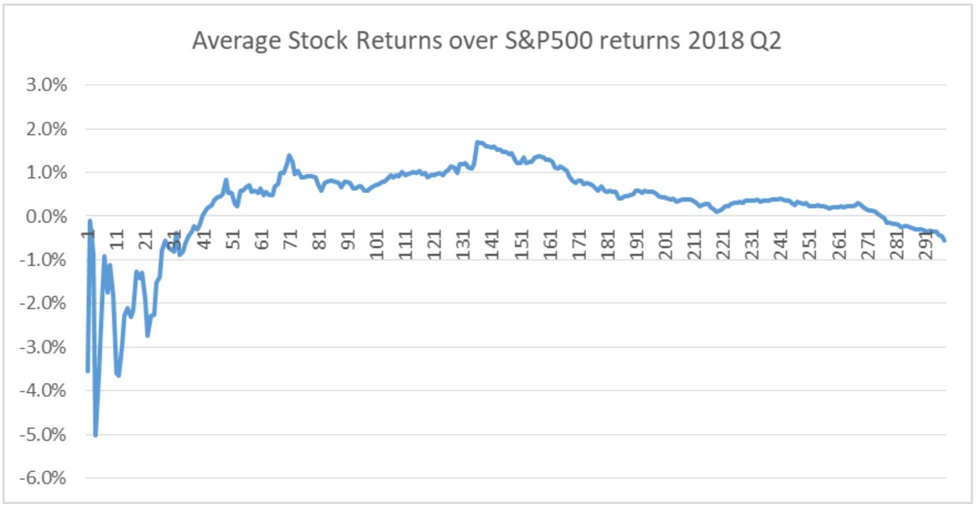
Although not perfect, these two plots seem to have captured the top performing combination of stocks (lets' call them portfolio here).

It is possible to say that, when all the stocks in 2018 Q2 are ranked according to their predicted outputs in descending order, investing in a portfolio that consists of the top N stocks in this quarter will be most definitely more profitable than investing in a portfolio with the middle group of N stocks or the last N stocks.

Next, for completeness, the student has performed the same kind of tests but this time chosen to forecast what we call risk-adjusted returns, which is the stock price change minus the S&P500 price change over the same period of time.

The following plots came from two separate runs:





This time we can see that the model is less stable and can produce dubious results.

The student has been trying to understand what may be contributing to the model instability.

References:

[1] Chen, Zhou, Dai, “A LSTM-based method for stock returns prediction: A case study of China stock market”, 2015 IEEE International Conference on Big Data (Big Data)

[2] Chou, Nguyen, “Forward Forecast of Stock Price Using Sliding-Window Metaheuristic-Optimized Machine-Learning Regression”, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 14, NO. 7, JULY 2018

[3] Zheng, Jin, “Using AI to Make Predictions on Stock Market”, Stanford University

[4] Nikola Milosevic , “Equity forecast: Predicting long term stock price movement using machine learning”, Manchester University