Machine Learning Engineer Nanodegree

Capstone Project – Winton Stock Market Challenge

John Yiu
10 October 2018

I. Definition

Project Overview

In the financial market, the movement of stock prices are never entirely predictable. Having a reliable way to forecast stock prices is crucial for most trading strategies. Many techniques have been developed to make predictions, ranging from fundamental techniques that involve analysing company financial statements to technical analysis which utilises indicators derived from the prices.

Different kind of market participants perform trades on the stock which causes the prices to vary intraday. These price movements form a non-stationary time series data. Although the series formed are often very noisy, one can extract signals from historical price trends in some occasions and forecast future prices.

In particular, price prediction using machine learning algorithms have gained a lot of interests in recent years, due to the success in leveraging data mining techniques in the other fields such as science and technology, as well as the increased availability of data. It is however known to be a very challenging problem.

Problem Statement

This project is based on the Winton Stock Market Challenge on Kaggle which require participants to perform predictions on asset returns from a proprietary dataset provided. We make use of artificial neural networks to identify hidden patterns in the data.

Metrics

Submissions to this competition are judged on weighted mean absolute error, where each return being predicted is compared with the actual return, scaled by its corresponding weight:

$$WMAE = \frac{1}{n}\sum_{i=1}^{n} w_i \cdot |y_i - \hat{y}_i|$$

where w_i is the weight associated with the return i, y_i is the predicted return, $\hat{y}_i$ is the actual return, and n is the number of predictions. These columns are discussed more thoroughly in the next section. A total of 62 returns have to be generated by our model for each 5-day window.

II. Analysis

Data Exploration

The data source for this project is provided as a comma separated file by Winton Capital and is downloaded from Kaggle. The training data consists of 40000 samples, where each row is an arbitrary stock at an arbitrary 5-day time window.

For each sample, 25 features which could be relevant to prediction, the returns ranging from 2 days before up till 2 days after the trading day, as well as 180-minute intraday returns, have been given. It is also worth noting that these returns are calculated by Winton, where the methodology is not revealed. However, they are designed to be a representation of real-world data. Finally, there are two weights for intraday and daily returns correspondingly as the project is scored on weighted mean absolute error. A graphical illustration is provided in Figure 1.

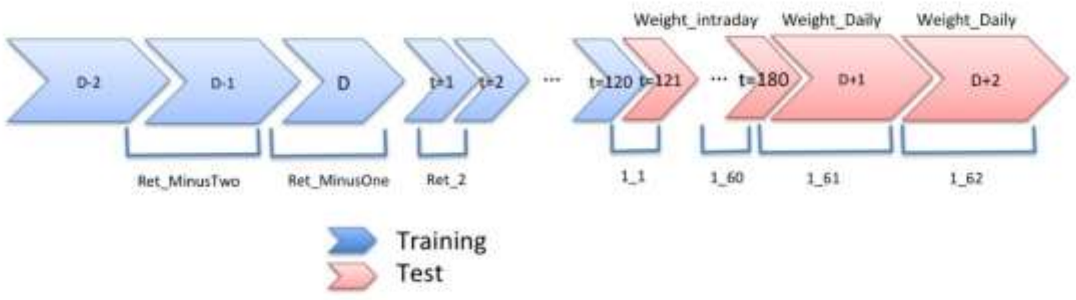

Figure 1: Graphical illustration of returns in the dataset, image source: Kaggle

Link to the dataset - https://www.kaggle.com/c/the-winton-stock-market-challenge/data

Exploratory Visualization

The distribution of daily returns is wider than the intraday returns, as one would have expected. Daily returns in the samples range from -60% to 80%, whereas intraday returns only range from about +/- 20%, this is shown in Figure 2.

The time series data formed by the given asset returns as shown in Figure 3 are quite noisy. The aim of our model is the make predictions on the returns starting from Ret_121, based on the features and historical returns, as illustrated in Figure 4.

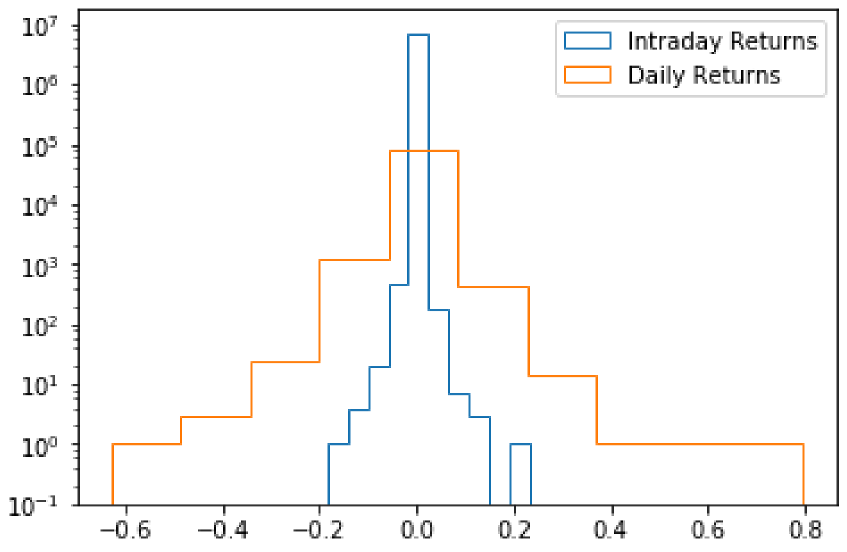

Figure 2: Distribution of daily versus intraday returns, note that the y-axis is in log scale.

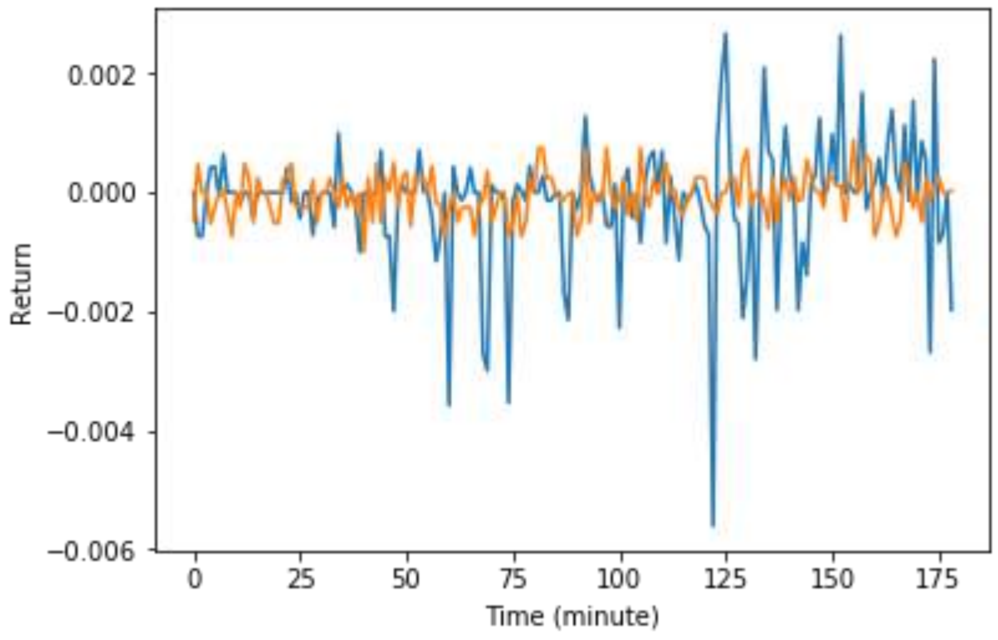

Figure 3: Intraday returns over time for sample assets.

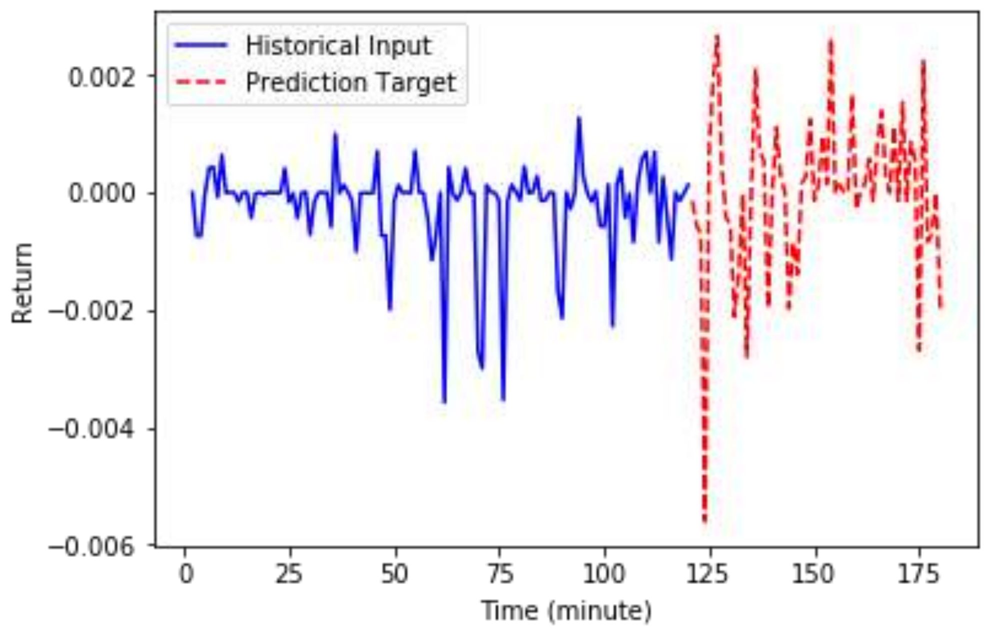

Figure 4: Historical input returns and prediction target for a sample asset, prediction should be provided from Ret_121 onwards.

Algorithms and Techniques

The problem is being structured as a regression, a stock returns prediction model is created using artificial neural networks, where the 25 anonymous features and various historical returns are being fed as inputs, and the output will be 60 ticks of intraday minute returns, together with 2 daily returns. An LSTM model is employed due to its effectiveness on time series data. Depending on the testing results the model may be split into versions to specialise in predicting intraday and daily returns separately, details on this is included in the next section.

Benchmark

The performance of our model is evaluated by the *weighted mean absolute error* of its predicted stock returns. A zero-return benchmark, which essentially predicts the asset price to have zero percentage returns (relative to the previous timestep) throughout the testing window, is used to compare against our solution. In other words, the zero-benchmark model simply uses the asset price from today day as its prediction for tomorrow's closing price, which is relatively naïve but may not be trivial to outperform. Our deep learning model should have a lower weighted mean absolute error if it is able to effectively capture any hidden patterns in the input features.

III. Methodology

In this section we document the inputs and outputs of our model, the model implementation, including its hyperparameters, as well as the metrics for measuring the model performance.

Data Preprocessing

To ensure the model can be trained effectively, the dataset is pre-processed before being used by our model. The features and returns are not always available for every sample. These missing fields have been replaced with zeros, so effectively we assume an asset price is unchanged (zero return) for our model training purpose. In addition, since the feature columns do not vary with time, in order to generate an input which can be processed by an LSTM model, we repeated all features by the number of input returns available (119 intraday, 2 daily) such that, for each timestep of each sample, there are 25 anonymous features from the original dataset, plus one return, i.e. 26 features in total.

Due to the sizable differences between intraday and daily returns, we have eventually chosen to segregate the regression problem into two parts. Two separate models are created for prediction intraday and daily returns respectively. For the intraday model the inputs consist of the 119 intraday returns (timesteps), whereas the model for daily return prediction only uses 2 daily returns as its input. Together with the 25 features, the shape of input matrices for each sample stock in the daily model and the intraday model are (119 x 26) and (2 x 26) respectively.

Implementation

The samples are split into training and testing set to avoid statistical bias in our final results. 8000 samples are used in the testing set, which constitutes 20% of the dataset. In the model training phase, 32000 samples are used, of which 8000 (25%) of them are being used as validation set for the purpose of tuning hyperparameters.

Both the intraday and daily prediction model are built using the Keras library, with Tensorflow backend. Initially, a single model is used to perform predictions on daily and intraday returns altogether. The initial neural network consists of an LSTM layer with 50 neurons, with one dropout and dense layer, using linear activation function and Adam optimizer. However, the results, which are detailed in the next section, are unsatisfactory. Later on, the model is broken into two parts which allow daily and intraday stock returns to be predicted separately. Numerous attempts have then been made to fine tune the hyperparameters, a model checkpoint has been added to the training process such that only the best weights are saved throughout the rather time-consuming process.

Refinement

Considerable effort has been made to improve the data processing code to a state where it is both efficient and utilises the available functions in the pandas and numpy library. The project would likely be enhanced further in the future therefore any utility logic has been refactored into functions to improve reusability.

In the final version of our models, both the intraday and daily prediction model have 200 neurons in the LSTM layer, with 2 Dense layers (each with their corresponding dropout layers at 20%), using the Adam optimser and Relu activation function. In particular, a marking layer has been added in front of the LSTM layer to mask any inputs that are zero-ed out at the pre-processing stage to ensure a fair representation on the unavailable data points. As the evaluation metric for this competition is weighted mean absolute error, mean absolute error is chosen as the loss function during the model training process. Prediction results from these final models are discussed in the next section.

IV. Results

Model Evaluation and Validation

The parameters being used in the final model is summarised in Table 5.

Model	**LSTM Neurons**	**Size of Dense layer**	**Loss function**	**Activation function**	**Optimiser**
Daily return	100	200	Mean absolute error (MAE)	Relu	Adam
Intraday return	250	500	Mean absolute error (MAE)	Relu	Adam

Table 5: A summary of hyperparameters being used in the models.

The models are trained across 3 epochs, with a batch size of 500, since the validation loss does not improve beyond the first epoch. Mean absolute error is chosen as the loss function, our intraday model achieved an error of 0.0006341, whereas our daily model achieved an error of 0.0155. These values are very close the metrics reported by the validation set, which has mean absolute errors of 0.0006326 and 0.0154 respectively.

The number of units in the dense layers are two times that of the LSTM neurons. The intraday return model is configured to have a higher number of neurons because of the larger input and output sizes compared to the daily model. Increasing the number of neurons or the number of Dense layers do not have observable effects on the MAE of the validation set. The choice of optimiser do not have much effect on the MAE either, as long as enough epochs is being run to ensure the model is sufficiently trained, Adam is finally chosen has it has a better performance relative to the others we tested, e.g. RMS, SGD.

Justification

If we compare the daily model error against the standard deviation of daily returns (0.025039 for Ret_PlusOne, 0.024160 for Ret_PlusTwo), it is not too significant. However, the results are then compared against the zero-benchmark model, which uses zero for all its stock returns prediction. The results from both the daily and intraday model rather disappointingly perform only as good as the zero-benchmark. The intraday results are included in Figure 6.

The final models have not adequately solved the problem as their prediction capabilities are only matching the zero-benchmark. One should note that although the zero-benchmark may appear to give a relatively low error, it can never be used in a real-world trading strategy.

```
Weighted Mean Abosolute Error on testing set:
Model:  0.000599515506184
Zero benchmark:  0.000599515506184
```

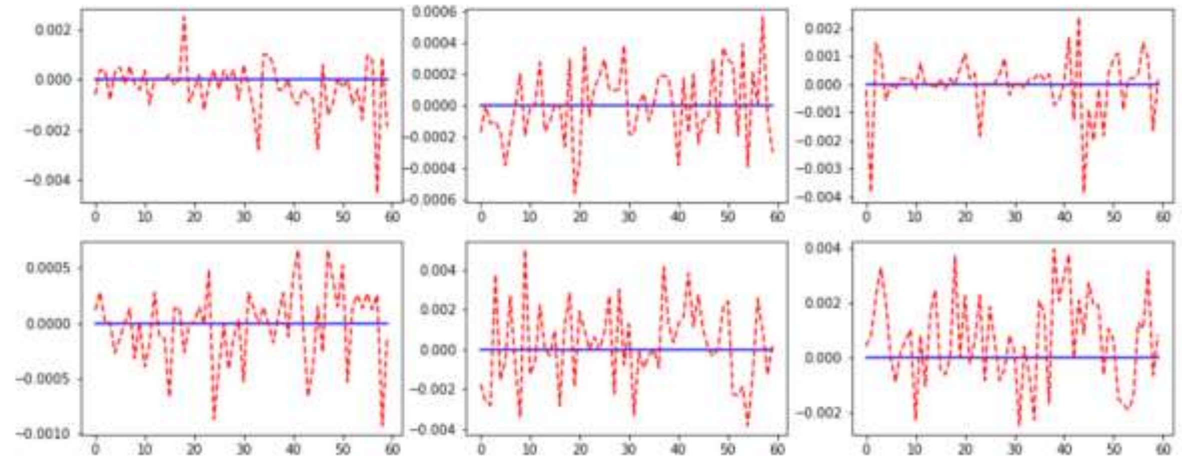

Figure 6: Results of the intraday model on a few random samples, blue lines are the model predictions whereas the red lines show the actual returns. Based on the mean absolute error function we chose to optimise against, our model only performs as good as the zero benchmark.

V. Conclusion

Free-Form Visualization

As an attempt to gain further insight from our results, it is not too surprising that our model converges to the zero-benchmark solution, given that the zero benchmark can provide a rather robust estimate. Further, since the competition is evaluated in weight mean absolute error, over a long period of time, zero-benchmark can in fact reach a decent local minimum in the given stock market data. A number of stock price movement illustration has been included in Figure 7 to demonstrate this.

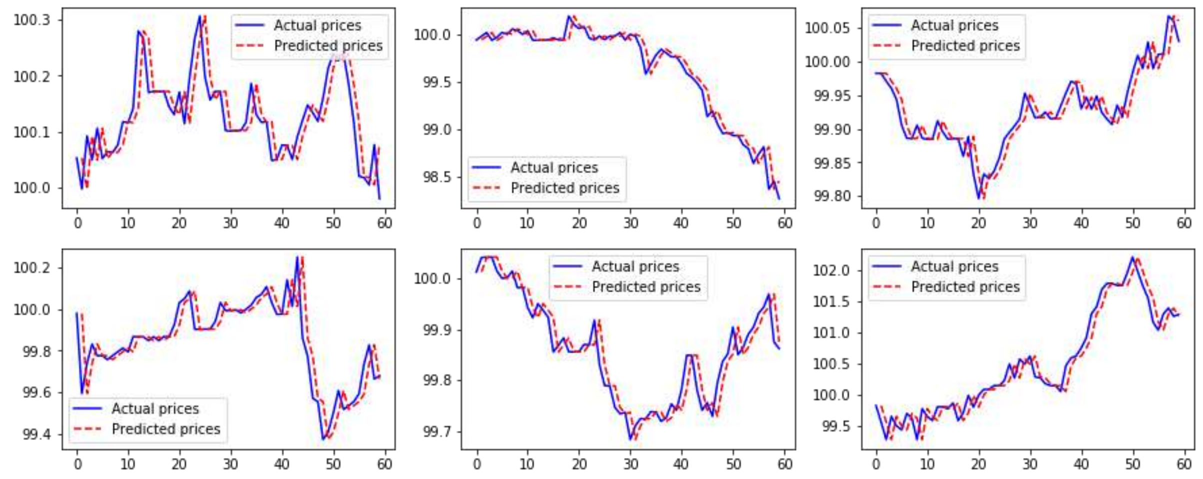

Figure 7: Sample stock price movement and the prediction using zero-benchmark. As one can see, the zero benchmark is able to achieve a good accuracy if mean absolute error is considered.

Reflection

The Winton Stock Market Challenge has proven to be a very challenging problem to be solved. Part of it is due to the fact that signals in the stock markets are intrinsically very noisy. Theoretically, one should be able to develop a neural network which has the capability of capturing hidden patterns in the stock returns time series data. However, in this project, we implemented the model following the plan in the capstone proposal, but the signal to noise ratio seems to be rather low.

Despite the final results of the project is somewhat disappointing, it has been a very good learning experience. I now further appreciate how difficult a deep learning project in practice can be. The project gave me an opportunity to carry out a machine learning project almost without any guidance from the others, from processing the data, to model development, and the final results analysis. These should hopefully be useful as I work on more machine learning projects in the future.

This project is in particular difficult because that there exists a naïve benchmark which requires a non-trivial effort to outperform. Originally, in the project proposal we intended to use a random uniform distribution to compare against the performance of our model. However, the idea is eventually abolished as such a benchmark is too trivial to beat, not to mention such a model does not make much theoretical sense either.

This competition is also slightly complicated by the fact that the Features being provided are not labelled, in other words one cannot develop any intuition from it to see which feature would be more specific to the problem. Further researches on the discussion forum for this competition show that even the top performers in this challenge only managed to beat the zero-benchmark by less than 1%. In fact, many submissions are out-performed by the zero-benchmark.

Improvement

A few aspects of this project can be improved further. For example, feature scaling could be done on our dataset to provide better convergence in our models; some of the features may not have a significant correlation with the stock returns, removing them might improve the performance of our models. It is also worthwhile to explore solutions that do not involve the use of a neural network, such as logistic regression. Additionally, the loss function being used to optimise our model should have taken the sample weights into account. The final solution does not have weights incorporated as they actually worsen the results based on how the neural network is currently set up.

References

https://www.kaggle.com/c/the-winton-stock-market-challenge

http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0180944

https://www.sciencedirect.com/science/article/pii/S2405918818300060