Machine Learning Engineer Nanodegree
Capstone Project

Hafidz Zulkifli
March 24, 2017

Predict Intraday Returns of Stock Market

Definition

Project Overview

Predicting the stock market has always been something that everyone has been trying to accomplish ever since the market itself was introduced. The incentives are high – as those who can predict the stock market could ensure that returns for investment can significantly be increased, and stocks which are predicted to not perform can also be avoided to negate losses.

The race is on for the industry to attain higher accuracy in being able to predict the market. However, given that information itself now travels faster as compared to the olden days – the challenge is on for analysts to be able to consume huge amounts of data and have it processed with so little time.

Traditionally, regressions methods such as ARIMA is the go to technique for analysts to go about building a predictive model for time series data. This was back in the day when computing power wasn't sufficient to compute data at a large scale as compared to now.

With recent development and advancements in big data and computing power - deep learning now provide us with the ability to process data without prior understanding of the features being inserted – thus seems to be the ideal methodology on how one should approach building a machine learning classifier meant for stock prediction.

Problem Statement

In this project, our goal is to predict stock market behavior given historical stock performance and a host of other features which are masked. Our data [2] is derived from Winton Stock Market Challenge competition hosted on Kaggle [1].

The problem is challenging given that it relies on time series data, with most of the fields being given labels which are non-descriptive. In general, there are features that describes

the stock, the returns of the stock for a time interval, and weights. The outcome to be predicted are return values for a set of future intervals.
Furthermore, the stock market movement are known to really volatile as a result of speculation (for which we have no existing data of).

The Kaggle competition itself asks the competitors to predict 2 things:

a) The intraday return
b) The subsequent daily return

By the end of the challenge (the word challenge and competition are used interchangeably in this documentation), a lot of the top Kagglers ended up predicting only the daily return, while keeping the intraday return at zero (no return). This was a sound strategy to them as they have seen little to now increase in the final score of the intraday outcome.

Given these challenges, one approach that can be used is to use recurrent neural network (RNN). A neural network typically would not require much data understanding to be able to work optimally (although being able to do prior massaging would help).

RNN - specifically the long short-term memory (LSTM) variant of RNN; has been proven to perform well for learning sequences from time-series data. A typical problem with the standard RNN architecture is a phenomenon called the vanishing gradient problem. By using LSTM however, we are able overcome this issue.

In this capstone, we will be focusing our efforts in improving the result by way of using a recurrent neural network to process historical sequential intraday returns data. As for daily returns, we will put less emphasis on it and instead use existing or out-of-the-box method to derive the predicted outcome.

Metrics

Each submission in Kaggle will be evaluated using the Weighted Mean Absolute Error. As such, we will also be using the same metrics to evaluate our model while offline. Each model predicted return is compared with the actual return. The formula is then

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

where w_i is the weight associated with the return, Weight_Intraday, Weight_Daily for intraday and daily returns, i_i, y_i is the predicted return, $\hat{y}_i$ is the actual return, and n is the number of predictions.

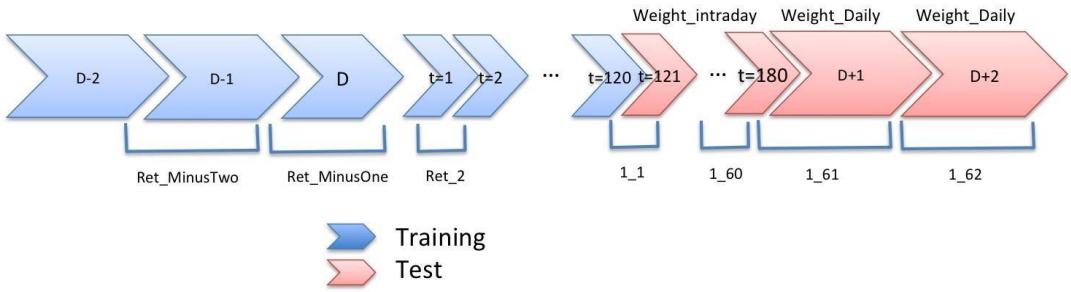
The weights for the training set are given in the training data, while the weights for the test set are unknown.

Analysis

Data Exploration and Visualization

To solve the given problem of predicting the future, we are given historical data of the past few days. The datasets are available from Kaggle.

We are given 5 days in total, days D-2, D-1, D, D+1, and D+2 respectively.



The data in D-2, D-1 and part of day D will be used for training, while the rest of remaining days are to be used as test.

Return data are available only for training datasets (D-2,D-1 and part of day D).

In total there are 25 features provided for the whole 5 day duration to be used in modelling.

Below are description on the files available for the problem.

1. **train.csv:**

- Feature_1 - Feature_25
- Ret_MinusTwo, Ret_MinusOne
- Ret_2 - Ret_120
- Ret_121 - Ret_180: **target variables**
- Ret_PlusOne, Ret_PlusTwo: **target variables**
- Weight_Intraday, Weight_Daily

2. test.csv:

- Feature_1 - Feature_25
- Ret_MinusTwo, Ret_MinusOne
- Ret_2 - Ret_120

3. sample_submission.csv

The last file mentioned above is to be used as reference for the output file that will need to submitted to Kaggle for scoring purposes.

Below are the description of the features available in the train and test dataset.

Field Name	Description
Feature_1 to Feature_25	different features relevant to prediction
Ret_MinusTwo	this is the return from the close of trading on day D-2 to the close of trading on day D-1 (i.e. 1 day)
Ret_MinusOne	this is the return from the close of trading on day D-1 to the point at which the intraday returns start on day D (approximately 1/2 day)
Ret_2 to Ret_120	these are returns over approximately one minute on day D. Ret_2 is the return between t=1 and t=2.
Ret_121 to Ret_180	intraday returns over approximately one minute on day D. ***target variables.**
Ret_PlusOne	this is the return from the time Ret_180 is measured on day D to the close of trading on day D+1. (approximately 1 day). ***target variables.**
Ret_PlusTwo	this is the return from the close of trading on day D+1 to the close of trading on day D+2 (i.e. 1 day) ***target variables.**
Weight_Intraday	weight used to evaluate intraday return predictions Ret 121 to 180
Weight_Daily	weight used to evaluate daily return predictions (Ret_PlusOne and Ret_PlusTwo).

A pre-requisite in using neural network for machine learning is the amount of data – as NN typically needs huge amount of data to produce a decent result. Alternatively, if the data set is small in size; other methods might be more suitable – such SVM or linear regression.

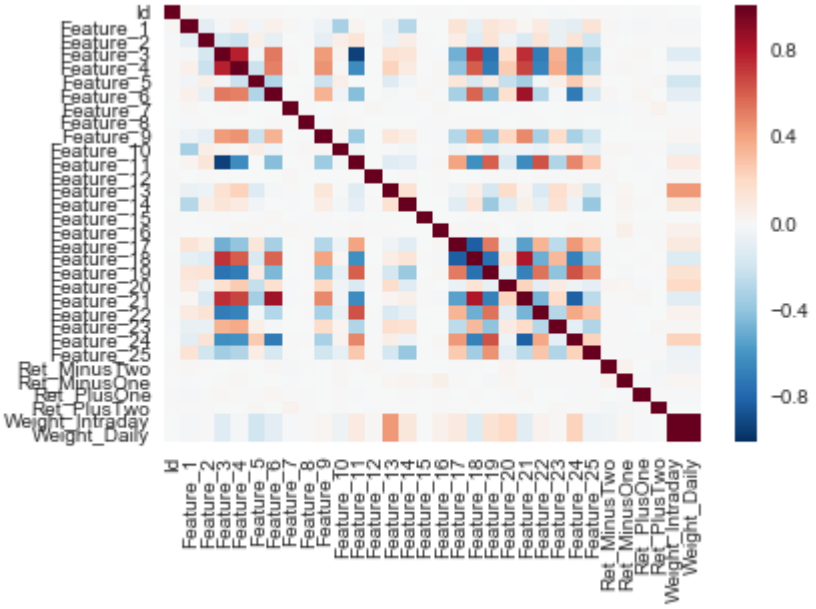
The training sample itself contain 40,000 rows of data. The test data meanwhile has 120,000 rows of data. With these amount, we should be able to use recurrent neural network as our method of choice.

The table below shows some basic statistics on the non-sequential features that were provided. In some (ie Feature_1), we can see that a bulk of the values are NULL; and some like Feature 5 and 7 – are fully populated. In the discussion forum we learn that some Feature_5 is somehow related to month and Feature_7 is the date_id. There were no other details for the other features.

	count	mean	std	min	50%	max
Id	40000	20000.5	11547.14972	1	20000.5	40000
Feature_1	6687	3.59025	2.798532	1	3	10
Feature_2	30854	-0.117558	1.23625	-3.44052	-0.38916	4.17515
Feature_3	38763	0.558392	0.902233	-4.64353	0.437228	4.530405
Feature_4	32279	0.405572	0.799082	-5.4406	0.403516	2.953163
Feature_5	40000	5.482775	2.942324	1	6	10
Feature_6	38067	0.430972	1.498274	-0.93664	0.055564	12.60989
Feature_7	40000	49244.97153	28242.40972	338	48457	99861
Feature_8	39531	0.196958	0.138485	0.0098	0.2138	0.365
Feature_9	38125	10.680289	2.850634	0	11	36
Feature_10	20529	4.744703	0.865096	1	5	6
Feature_11	39013	-0.572244	1.246347	-7.35912	-0.32492	1.786886
Feature_12	38904	0.498508	0.351855	0	0.5	1
Feature_13	39406	4.238162	2.570493	0	4	9
Feature_14	39272	1.588524	0.316907	-0.14927	1.611754	3.161848
Feature_15	37859	3.891381	5.035073	0.021679	1.292008	28.01811
Feature_16	39390	1.007362	0.085488	1	1	2
Feature_17	39354	-0.549725	0.936833	-2.61399	-0.59905	7.683857
Feature_18	39432	0.803059	1.165442	-5.75805	0.587005	6.352352
Feature_19	38810	-1.205438	0.642426	-3.29291	-1.16933	0.898236
Feature_20	32174	5.267359	2.549227	2	5	10
Feature_21	38982	0.605593	1.319158	-1.515	0.308468	7.73702
Feature_22	38655	-0.773089	1.389229	-5.81991	-0.69911	2.284991
Feature_23	38289	0.799833	1.28804	-7.22139	0.96258	3.228906
Feature_24	39274	-1.20929	1.739656	-11.4422	-0.86844	2.526654
Feature_25	39345	-0.329675	0.958661	-1.90388	-0.55155	4.020332
Ret_MinusTwo	40000	0.000784	0.028279	-0.53628	0.000112	0.894024
Ret_MinusOne	40000	-0.000803	0.030569	-0.51472	-0.00067	0.852139
Ret_PlusOne	40000	-0.00021	0.025039	-0.62769	-0.00026	0.795602
Ret_PlusTwo	40000	0.000012	0.02416	-0.45078	-0.00026	0.303038
Weight_Intraday	40000	1.50E+06	2.06E+05	1.00E+06	1.48E+06	2.76E+06
Weight_Daily	40000	1.88E+06	2.57E+05	1.25E+06	1.85E+06	3.45E+06

Interestingly, if we look at the return on the previous 2 days (Ret_MinusTwo, Ret_MinusOne) and the next 2 days (Ret_PlusOne, Ret_PlusTwo), we see that the average and median value for the returns are close to zero. This very much explain the rationale we mentioned previously as to why zero based prediction benchmark itself is hard to beat.

Below is a correlation heatmap between the non-sequential features, and the daily returns (both past and future). For the below diagram, we have removed the intraday returns.



We can see few relationships between the features, both positively and negatively. However there seem to be no correlation at all between the returns and the features.

Next, we will dive more into the sequential features – which we are interested in improving for this project. We begin by analyzing the distribution of returns that we get for intraday trading. From the diagram below, we can see that typically the returns are close to zero, on the positive size. We could probably conclude that by default it's acceptable to set our intraday returns as zero.

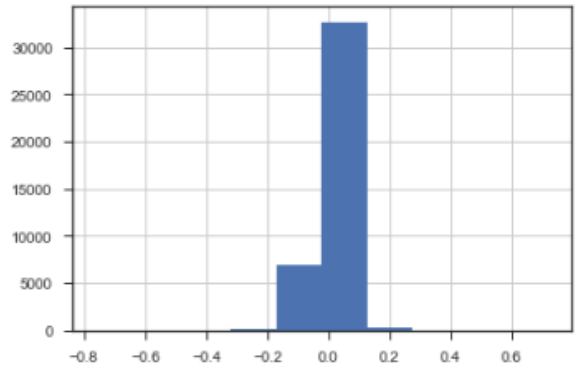


Figure 1: Intraday return histogram

Next, we try to understand the intraday trading volatility, based on historical data. We take a look at a single day data from one of the stocks provided.

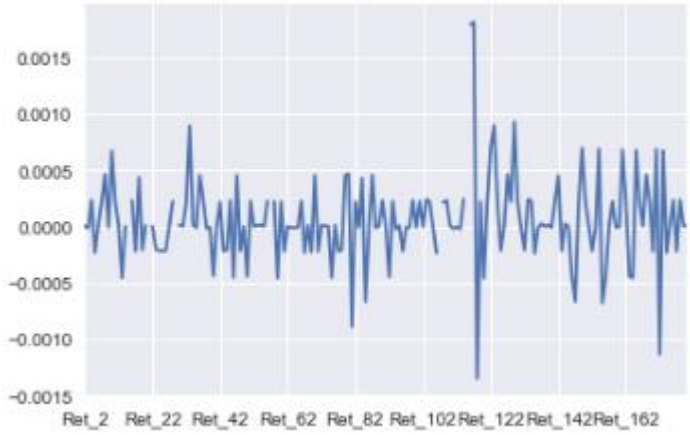


Figure 2: A sample of intraday returns

In the above we can see that the trend as a whole is rather volatile with few missing data. If we take the mean of each minute intraday trade, we get the below diagram.

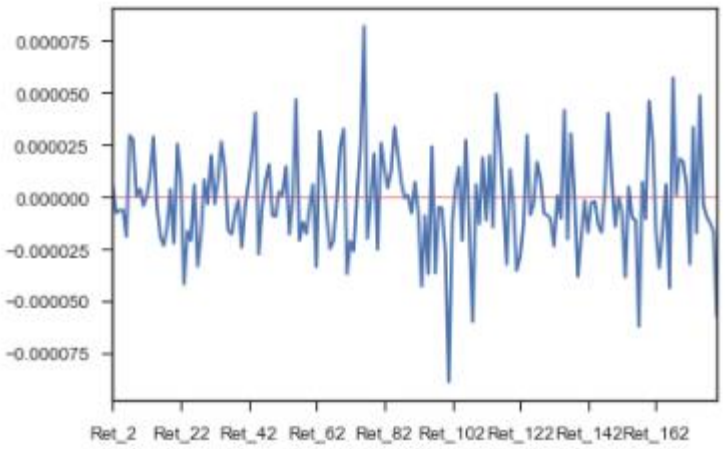


Figure 3: Mean intraday returns by the minute

As we can see, the trend is indeed volatile, with no obvious trend emerging. Flipping our vantage, we look at the mean of the returns for each row.

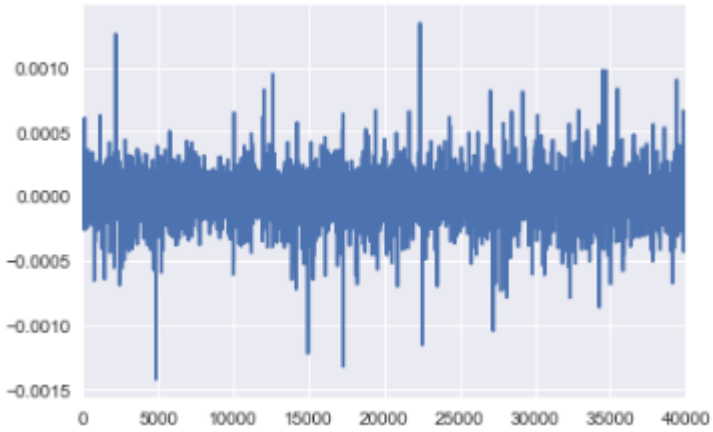


Figure 4: Mean intraday returns by each row

We see that on average the intraday returns are typically within the 0.0005 to -0.0005 range. The standard deviation figure below confirms this as well.

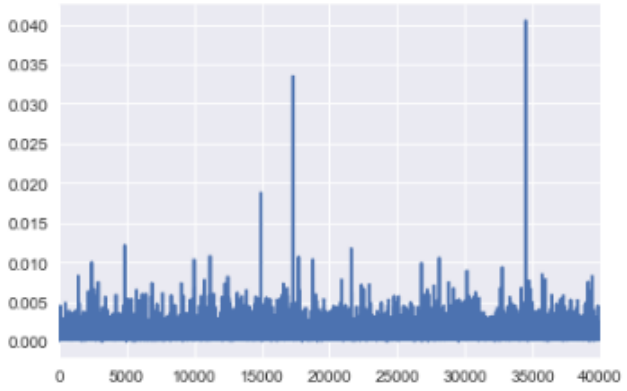


Figure 5: Standard deviation of intraday returns by row

The previous few figures gave us an understanding of the intraday returns for each row and what are the typical ranges that it revolves in. Let us now compare this and view the data from minutes return perspective.

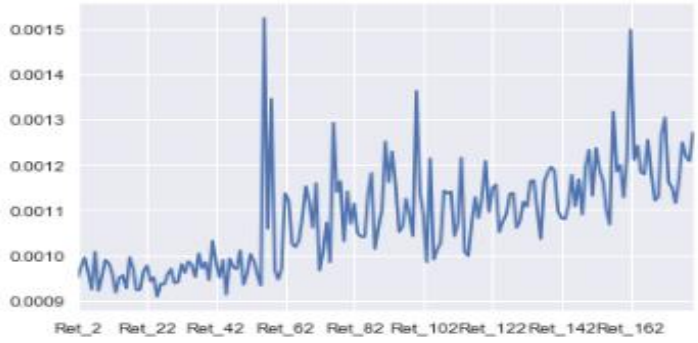


Figure 6: Standard deviation by minutes return

The above figures show that the returns standard deviation seems to increase over time. In other words - over time, returns get more volatile.

The second observation is the volatility spike that occurs around Ret_50, and how prior to that, there is very little change in deviation. Meaning, if one wants to sell or buy a share - it is preferable to do so in early morning instead of waiting until mid-day, as it is less predictable. This information seems interesting, and it might be worthwhile to include this as a new feature in our dataset during modelling stage later.

Finally we want to take a look at whether time plays a part in determining returns. From our previous findings, both during analysis and research stage – we've concluded that some of the masked features provided indicated month (Feature_5) and date (Feature_7).

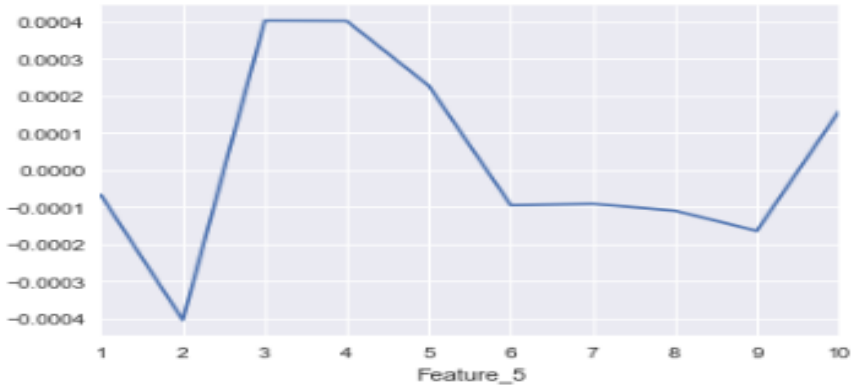


Figure 7: Mean returns over the months (Feature_5)

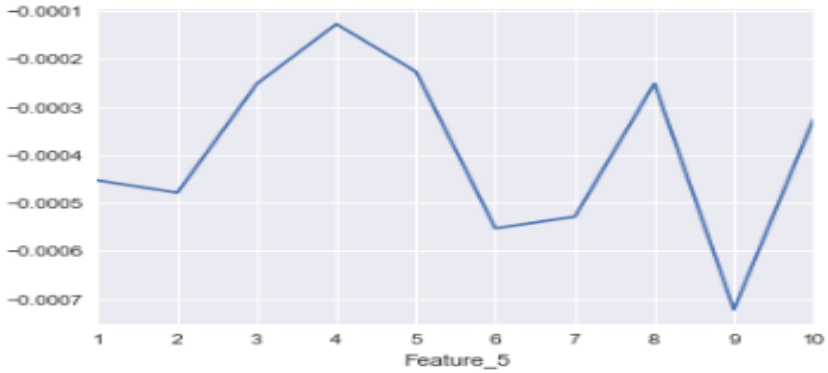


Figure 8: Median return over the months (Feature_5)

In the above figure, we derive return by taking an average of Ret_MinusOne, Ret_MinusTwo, Ret_PlusOne, and Ret_PlusTwo for each entry. We then plotted the mean and median return by month to see whether there's a trend or not. Based on the figures, we see a similar trending peak on month 3 and 4, with declining slope by month 6. We can conclude then that month does in fact play a factor in whether the returns will be increasing or decreasing.

This trend is not easily seen when the data is analyzed on the daily basis (as can be seen below) .

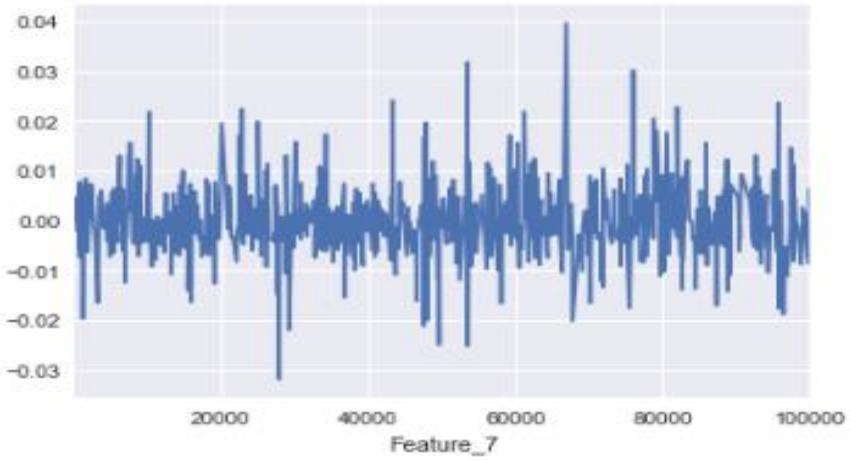


Figure 9: Average daily intraday return over days (Feature_7)

However we do get to see some pattern when we convert the dates into sequences of 7 to simulate day of week.

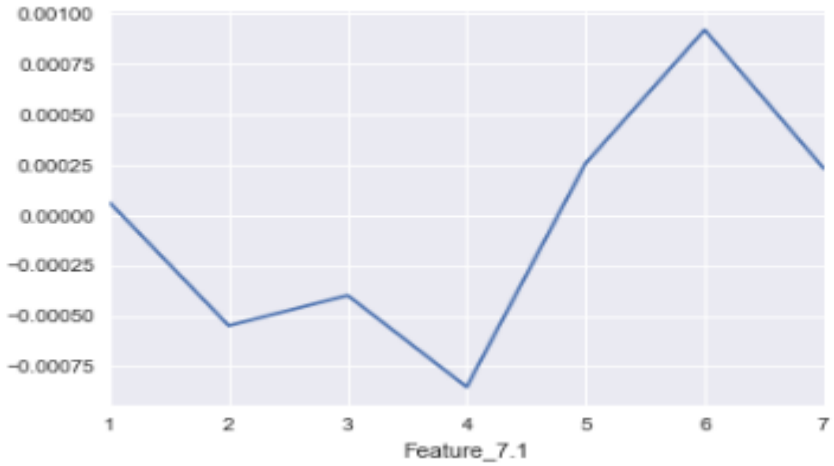


Figure 10: Mean return on day of week.

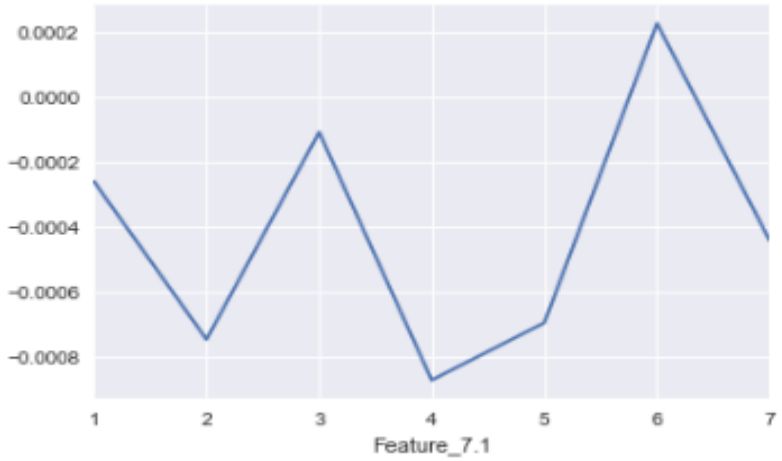


Figure 11: Median return on day of week

The above two figures show that there are days where the returns are typically higher and ones where it is generally lower. However, it's not exactly easy to speculate on the reason – since we do not know exactly the date.

In summary, we've learned the following:

1. Analyzing the returns by the date and minute is hard since overall there is no obvious pattern that one can derive due to the volatility of intraday trading.
2. However, features like mean, median and standard deviation can provide us with a way to signal the volatility.
3. When analyzed on weekly and monthly basis, we do see some peaks and valley on the returns.
4. While we have not analyzed the exact importance of each feature yet, item 2 and 3 does seem like a simple way to derive meaning from our data and could help us in our modelling later.

Algorithms and Techniques

The problem at hand is in a nutshell a time series prediction. For time series, the order of which the sequence of data flows in is important, as the next data that goes after it is related to the prior data.

Traditional neural networks can't do this [3], as the model has no memory or concept of persistency in maintaining the previous data that it has processed thus far.

Recurrent neural networks address this issue, since it uses a loop within its network in order for it to be able to store previous information.

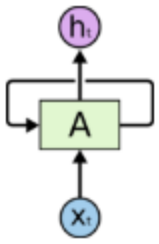


Figure 12: Loop within the network

From a mathematical standpoint, the RNN would take each element of a sequence, multiply the element by a matrix, and then sum the result with the previous output from the network. This can be summarized by the following equation.

$$h_t = activation(X_t W_x + h_{t-1} W_h)$$

There are many types of RNN out there. The ones that we are interested in using for this problem however is called Long Short Term Memory (LSTM). LSTM is popular in problems such as context recognition as it is able to store memory or a concept from a sequence far back in the past to be applied in the present. Other types of RNN fall short of this due to a phenomenon called the vanishing gradient problem (which LSTM can overcome).

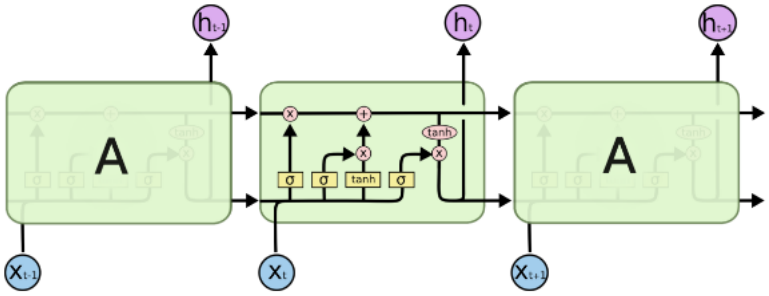


Figure 13: Repeating modules within an LSTM.

The above figure show what an LSTM module look like in principle. The LSTM itself has multiple variants of implementation. Each variant has been proven to work [4] better at some task and not in others. The variant that we will be using is called the Gated Recurrent Unit (GRU), introduced by Cho, et al (2014) [5].

Our approach in applying the algorithm and solve the problem is as follows:

1. Create additional features as mentioned in the previous section for more context.
2. Use a simple 2-layer network RNN with default hyperparameters to establish our first model benchmark.

Benchmark

For benchmarking purposes, we will compare our model performance against the following scenarios:

1. Zero prediction. A zero prediction is a situation when returns have no changes over time. This is chosen as our lower benchmark – so that we know how bad our model performs against no prediction. The current score for a zero prediction is shown below.

Figure 14: Score for zero prediction

2. The public leaderboard for this respective Kaggle challenge. Given that this problem originated from Kaggle, it is naturally thus to compare the performance of my model against the others who have competed. This is chosen as our 2nd benchmark, as a way to evaluate how good our model is when compared against other top Kagglers. The below screenshot shows the current public leaderboard score for this challenge.

Figure 15: Score for first place in the public leaderboard

Our goal thus is to not only to produce a model that can perform better than a zero predictor (which by the way, is no mean feat) – but to also get as close as possible (if not beat) the highest score in the leaderboard.

Methodology

Data Preprocessing

For modelling, the data was first preprocessed according using the below steps:

1. Created new features based on our earlier exploration and analysis. They are intraday mean, intraday standard deviation, intraday mean and day of week.
2. Imputed missing data. Data that are blank or missing are replaced with zeroes.
3. Rearrange columns to fit into format desired by model. Some columns were dropped since it's not relevant to our modelling stage.
4. Divide training data into train and validation set using a 70:30 ratio.
5. Padding was applied at both our encoder and decoder to simplify our model.

Missing from the above steps are normalization, which is important especially if sigmoid or tanh were used as the activation function. However, since our input data are mixed between sequences and non-sequential data, we've decided to omit this step for the time being.

Implementation

Our approach to implementing the model will be to use a sequence to sequence neural architecture, where essentially there needs to be at least 2 recurrent neural networks (RNN). The encoder, processes the input while the decoder, generates the output. A basic architecture of this is depicted below. Each box in the below picture represents a cell of the RNN, typically a GRU cell or an LSTM cell.

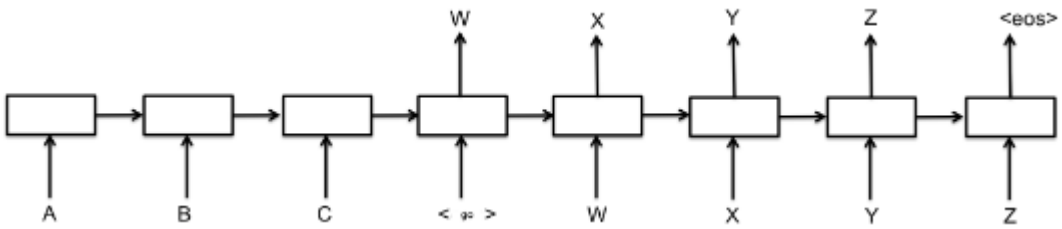


Figure 16: Basic sequence to sequence RNN

To illustrate how it would work, consider the below example of input and target (output sequence).

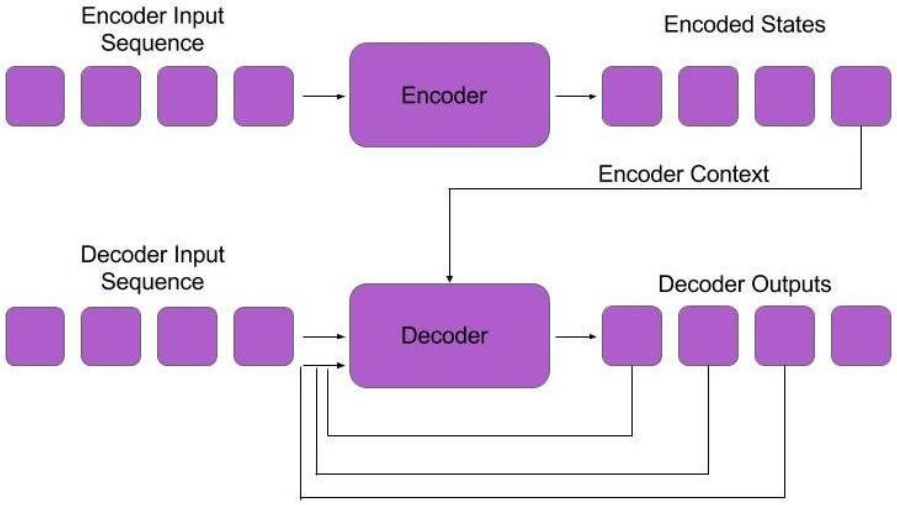


Figure 17: The encoder outputs an encoded context. The decoder processes this along with the target sequence. A feedback loop is also formed from the decoder output

To simplify our architecture even further, we will be wrapping our multiple RNNs into a single cell by using tf.nn.rnn_cell.MultiRNNCell function provided by Tensorflow. A diagram of this is depicted below.

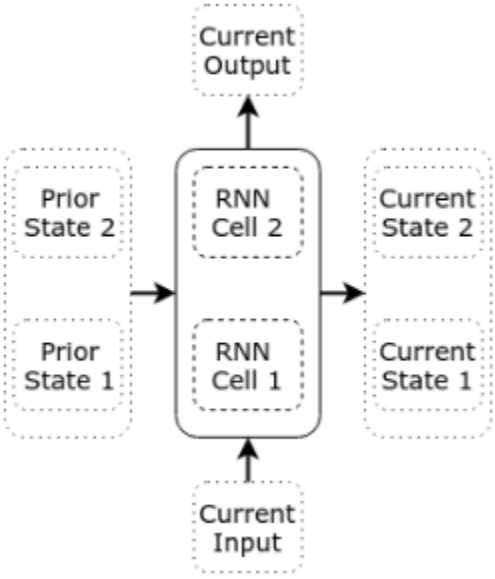


Figure 18: Wrapping multiple cell into one

In the encoder step, the model converts an input sequence (such as an English sentence or historical stock prices) into a fixed representation. In the decoder step, the model is trained on both the output sequence (such as translated sentence or future stock price) as well as the fixed representation from the encoder. Since the decoder model sees an encoded representation of the

input sequence as well as the translation sequence, it can make more intelligent predictions about future values based on current value.

Both of our encoder and decoder were based on RNN. During training, its main objective is to look at the translated input sequence, and predict the next value in the decoder sequence. After training is completed, prediction is done by encoding the historical stock prices using the encoder and running the network in the generation mode.

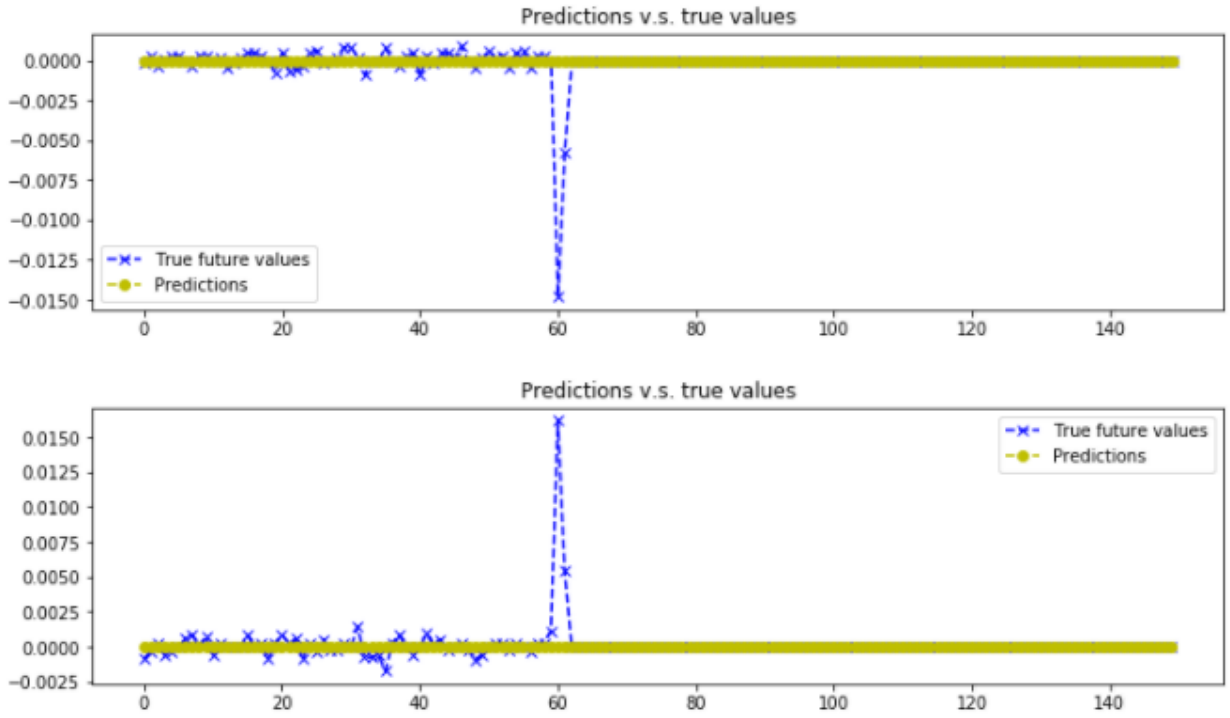
Result

Model Evaluation

After setting up our Tensorflow, we've trained the model over 1000 iterations. The result can be referred to below.

In the diagram above, test scores are taken at a longer interval compared to the train loss. After around 800 iterations, the training errors have plateau-ed at log -5.

We then proceeded to run the model across the validation data set. Below are the results:

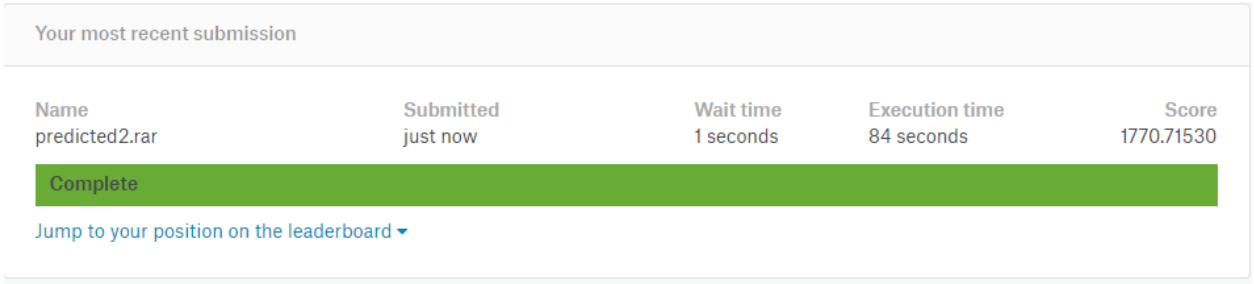


Note that for both training and validation dataset – we have padded the data to be of equal with the incoming sequence (150). The predicted sequence is only 62 in length – thus the graph above should only be evaluated from point 1 to point 62.

In addition, the last two predicted points are of different granularity compared to the prior points. Recall the that point 61 and 62 are returns for T+1 and T+2 instead of intraday returns. This explains the vast differences in returns compared to the other points.

Model Validation

Predicted outcomes are compiled and are scored in Kaggle. Result can be referred to below.



As a comparison, recall that our benchmark is as follows:

Zero Prediction	**1st Place**
1770.03211	1698.234

Thus it can be seen that the current submission was not able to get close to the top positions, much less surpass the zero prediction.

Conclusion

Reflection and Improvement

To conclude this report, we make the following observations:

1. The model tends to generate a single value as the prediction. While preparing the submission file to Kaggle, we noticed that the model generates only a single value for all the output.
2. However due to time constraints, we didn't investigate this further.
3. Several hypotheses could be explored in the future:
    a. Padding of zeroes would have made the NN to summarize that whatever value to predicted must be close to zero. Using bucketing or dynamic padding would probably solve this issue.
    b. The current decoder input did not deploy a feedback mechanism while learning. Using other types of function such as embedding_rnn_seq2seq would enable us to use feedback input at the decoder.
4. The main challenge that was faced during preparation of this report was learning Tensorflow. Certain concepts such as how data needs to be prepared before using certain classes in Tensorflow, understanding data flow between nodes – are some of the difficulties faced.
5. It's encouraging to see that even without doing heavy feature engineering we could achieve a result close to the zero prediction (which is according to some Kagglers is not as easy as it seems for this competition).
6. One of the way the result could be improved is to prepare separate model for intraday returns and daily returns.

References

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

[2]: https://www.kaggle.com/c/the-winton-stock-market-challenge/data

[3]: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[4]: http://jmlr.org/proceedings/papers/v37/jozefowicz15.pdf

[5]: http://arxiv.org/pdf/1406.1078v3.pdf

[6]: http://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

[7]: https://github.com/llSourcell/seq2seq_model_live/blob/master/2-seq2seq-advanced.ipynb

[8]: https://github.com/guillaume-chevalier/seq2seq-signal-prediction

[9]: https://indico.io/blog/sequence-modeling-neuralnets-part1/

[10]: http://suriyadeepan.github.io/2016-12-31-practical-seq2seq/