The Winton Stock Market Challenge

by

Rajeev Kashetti (G01322331)

Sai Prashanth Reddy Kethiri (G01322333)

Abstract

Most of us commonly think that stock market is unpredictable due to variation in the variables of a company. One of the factors that inspired us to predict the stock market is our profound interest in the field of finance. In our project, we have used time series data and, also few machine learning algorithms like Multi-Layer Perceptron Regression, Linear Support Vector Regression, Keras Regression, Extreme Gradient Boost Regression (XGBoost), Random Forest Regression and our goal is to predict the stock market performance with a higher accuracy. Out of the listed models Linear SVR and MLP Regressor has provided us with the promising result, having a least Weighted Mean Absolute Error of 5.1.

1. Introduction

The Winton Stock Market Challenge was a competition hosted by Winton on Kaggle in 2016. The main task of this competition is to predict the interday and intraday return of a stock, given the history of the past few days. In our work, we have built different models to predict the stock ratings of a company.
The next section of this report describes the problem statement in brief. Chapter 3 summarizes the literature survey we have performed. Then the following chapters 4 and 5 will describe the methods we have used in the project and their respective results. And chapter 6 provides the conclusion for our project.

2. Problem Statement

Provided 5-day windows of time, days D-2, D-1, D, D+1, and D+2 of which returns in days D-2, D-1, and part of day D are given, we need to predict the returns in the rest of day D, and in days D+1 and D+2. During day D, there is intraday return data, which are the returns at different points in the day. The challenge is to predict the return of a stock, given the history of past few days.

3. Literature Survey

We have read a done some literature survey and read some articles online mentioned in Bibliography and came to a conclusion to implement the models described below in section 4.4 as models of our choice. Out of all the literature survey that we had done we found a research paper titled “Winton Stock Market Challenge: Using a Deep Learning Framework for Time Series Stock Market Prediction” by Stanford students more appealing. They have built two models where one model includes Denoising Wavelet Transforms and Fully Connected Neural Network Layers. Their final model includes Denoising Wavelet Transform, Stacked Auto-Encoders (SAE) and Layered Recurrent Neural Networks. The evaluation metrics that were used are cross entropy loss function and square difference loss function. Based on that we had tried to use the Recurrent Neural Network (RNN) to deal with the time series, but the RNN suffered from overfitting very badly. So we sticked with our initial thought that we got from earlier literature review i.e., to use models like Linear SVR, MLP Regressor etc. Paper by Jigar Patel [1] played a crucial role in choosing our models.

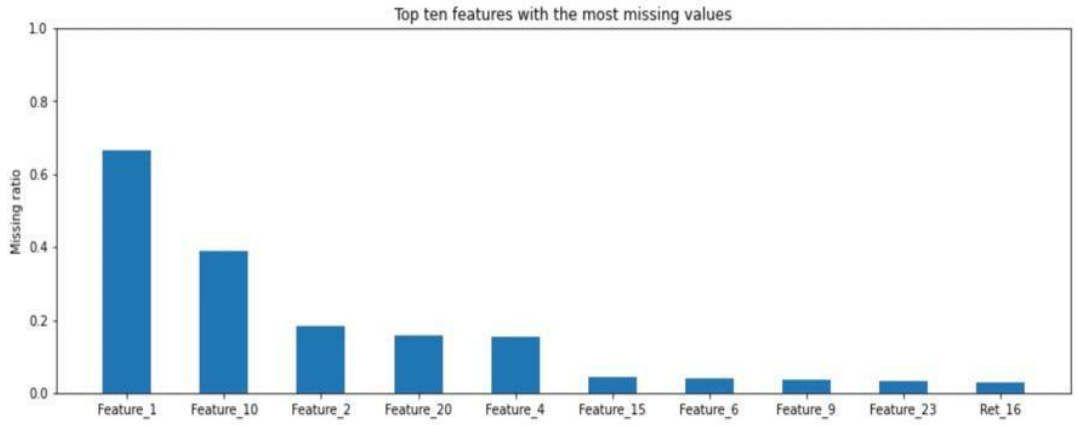
4. Methods and Techniques

4.1. Data Analysis

Here we will do initial data analysis. For each feature in the data set we look at the number of missing values, how many values are unique, how imbalanced the values are, how many potential outliers are contained in the values and if values in the training and test sets are disjoint.

We define potential outliers as values which differs from the mean by more than 3 standard deviations.

	0	Unique	Unique %	Imbalance	Imbalance %	Outlier	Outlier %
0	Feature_1	11	0.0275	2651	6.6275	0	0.0000
1	Feature_2	30855	77.1375	1	0.0025	62	0.1550
2	Feature_3	38764	96.9100	1	0.0025	269	0.6725
3	Feature_4	32280	80.7000	1	0.0025	80	0.2000
4	Feature_5	10	0.0250	6943	17.3575	0	0.0000
5	Feature_6	38068	95.1700	1	0.0025	1018	2.5450
6	Feature_7	824	2.0600	114	0.2850	0	0.0000
7	Feature_8	33	0.0825	4178	10.4450	0	0.0000
8	Feature_9	37	0.0925	5863	14.6575	237	0.5925
9	Feature_10	7	0.0175	14437	36.0925	778	1.9450
10	Feature_11	39014	97.5350	1	0.0025	605	1.5125
11	Feature_12	102	0.2550	1596	3.9900	0	0.0000
12	Feature_13	11	0.0275	4715	11.7875	0	0.0000
13	Feature_14	39273	98.1825	1	0.0025	552	1.3800
14	Feature_15	921	2.3025	61	0.1525	888	2.2200
15	Feature_16	3	0.0075	39100	97.7500	290	0.7250
16	Feature_17	39355	98.3875	1	0.0025	486	1.2150
17	Feature_18	39433	98.5825	1	0.0025	894	2.2350
18	Feature_19	38811	97.0275	1	0.0025	15	0.0375
19	Feature_20	10	0.0250	7008	17.5200	0	0.0000
20	Feature_21	38983	97.4575	1	0.0025	950	2.3750
21	Feature_22	38656	96.6400	1	0.0025	118	0.2950
22	Feature_23	38290	95.7250	1	0.0025	1434	3.5850
23	Feature_24	39275	98.1875	1	0.0025	798	1.9950
24	Feature_25	39346	98.3650	1	0.0025	1527	3.8175
25	Ret_MinusTwo	40000	100.0000	1	0.0025	635	1.5875
26	Ret_MinusOne	40000	100.0000	1	0.0025	689	1.7225
27	Ret_Agg	40000	100.0000	1	0.0025	673	1.6825
28	Ret_Agg_Std	40000	100.0000	1	0.0025	611	1.5275
29	Ret_Std	40000	100.0000	1	0.0025	684	1.7100
30	Ret_PlusOne	40000	100.0000	1	0.0025	625	1.5625
31	Ret_PlusTwo	40000	100.0000	1	0.0025	655	1.6375



Besides being categorical, Feature_6 and Feature_7 are also distinct between the training and the test data sets. However, Feature_6 is highly unique, whereas Feature_7 only contains relatively few unique values.

This leaves Feature_7 as a special feature. We will try investigating this feature more thoroughly, by grouping returns by values of Feature_7 and look for a relationship in the sign of the returns to see if equal values of Feature_7 has correlated returns.

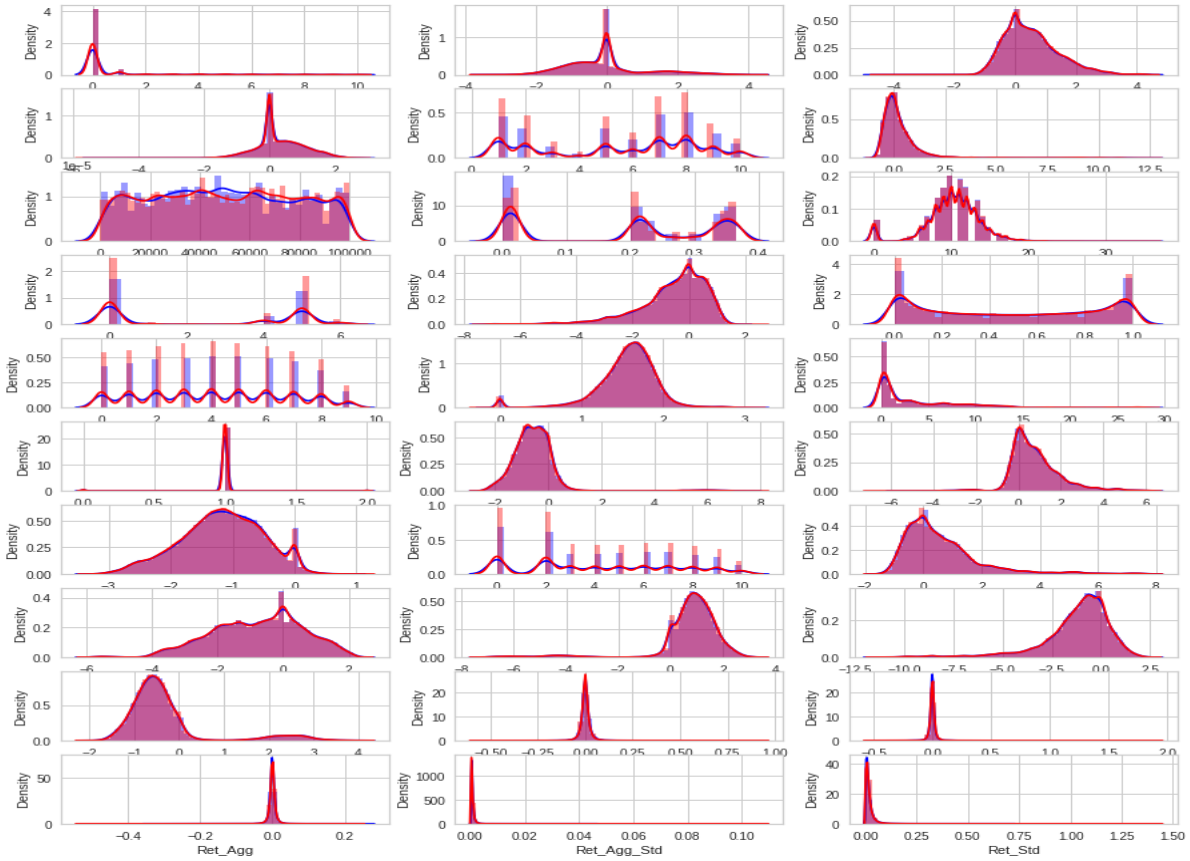
We observe that the sign of the returns within each group is almost 100% correlated. Regardless of what the nature of Feature_7 is, we must account for this relationship when doing model cross-validation to avoid data leakage. Most likely, Feature_7 has a relationship to time, which would explain why returns within the groups are correlated.

4.2. Data Preprocessing

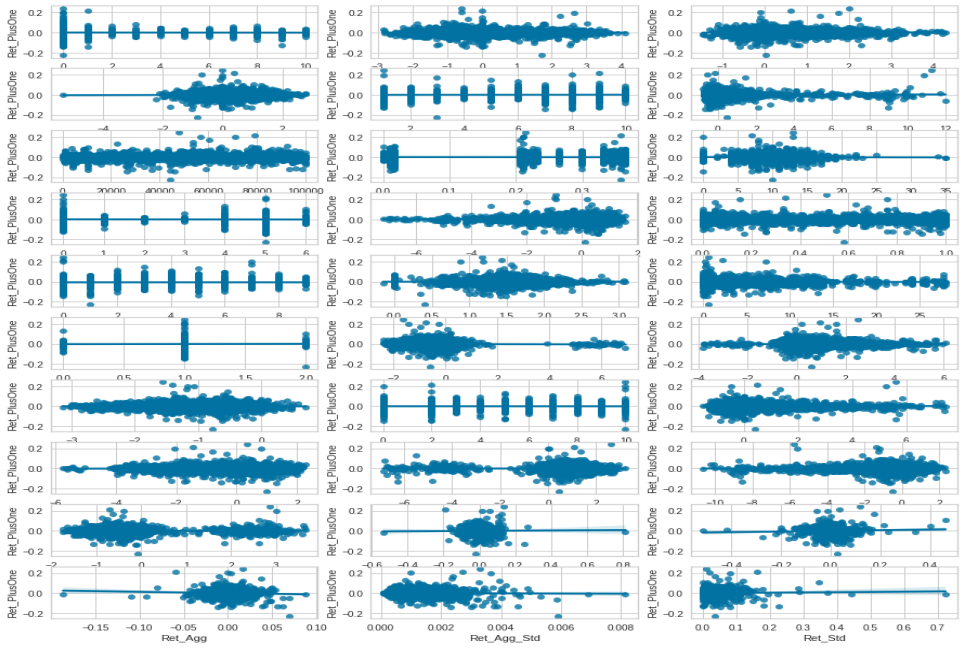
4.2.1. Feature Selection

We have considered all the features in the dataset and categorized them into numerical and categorical features. These categorical features are further split into ordered and unordered categorical features.

Before preprocessing:



Regression plots



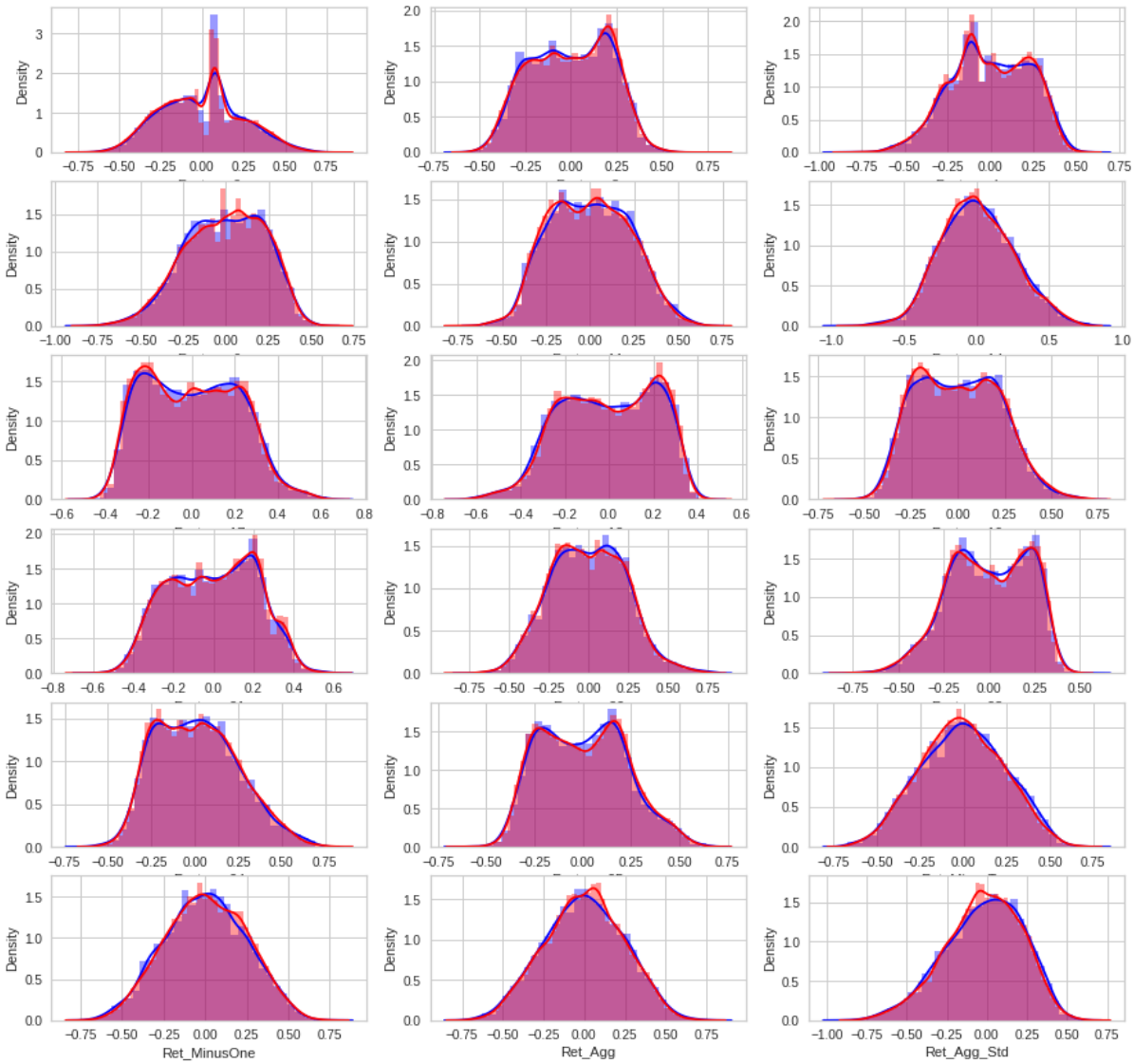
4.2.2. Preprocessing

We have tried many models to predict the time series of the intra-day stock returns. None of them beat the all-zero prediction, because intraday one min data is likely to be too noisy to be predicted by the data given in this competition. Hence we will only output predictions for {id}_61 and {id}62 and output 0 for all of {id}{1-60}. We also removed the imbalanced features. We decided to take the sum of all returns in the time series as a single feature representing the stock return in the first two hours in the current day and aggregate the intraday return features Ret_2 Ret_121 as a new feature called Ret_Agg.

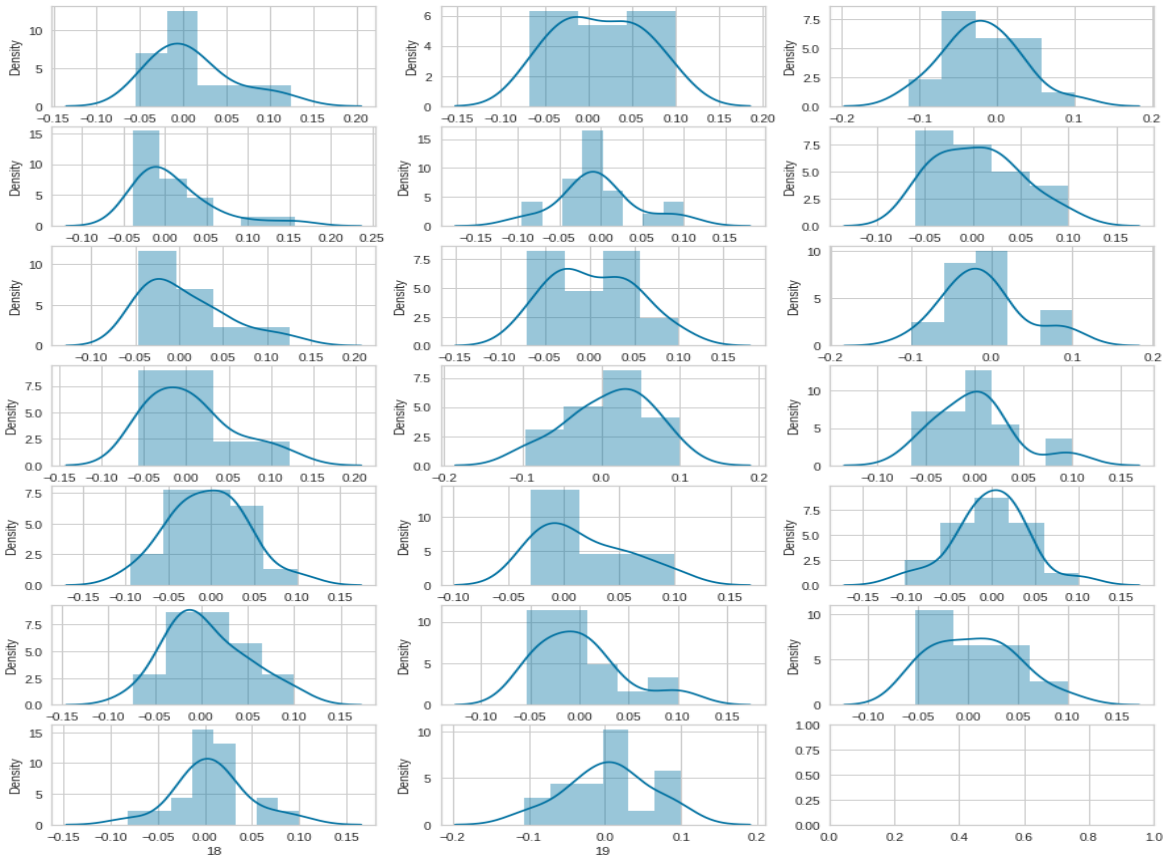
In case of numerical features, they are scaled using [5, 95] quantile range. We also have removed the values which lies more than three standard deviations from the center to improve our handling of outliers.

We have encoded the ordered categorical features using Ordinal Encoder, so that the order is preserved. Coming to unordered categorical features, we removed correlations between the features by using PCA whitening and then one-hot encoded them.

After preprocessing numerical features: Distribution



After preprocessing categorical features:



4.3. Initial Approach

During the early stages of our project, we have considered the methods like,

- **Autoregressive model**

  An autoregressive model predicts future behavior based on past behavior. It is used for forecasting when there is some relation between values in a time series and the values that precede and succeed them. The process is basically a linear regression of the data in the current series against one or more past values in the same series.

- **Moving Average**

  It's a method for analyzing data points by calculating the averages of different subsets of the entire data set. The first element of the moving average is obtained by taking the average of the initial fixed subset of the number series, given a series of numbers and a fixed subset size. The subset is then "shifted forward," i.e., the first number in the series is excluded and the next value in the subset is included.

- **ARIMA**

  Auto-Regressive Integrated Moving Average is a generalization of the simpler Auto-Regressive Moving Average and adds the notion of integration. This is an effective machine learning algorithm for performing time series forecasting. Parameter tuning for ARIMA consumes a significant amount of time which can be reduced by setting auto ARIMA which automatically selects the best combination of (p, q, d) that provides the least error. With ARIMA using past data to understand the pattern in time series the model captures an increasing trend in the series. Although these predictions using this technique are better than many implementations of machine learning models, these predictions are still not usually close to the real values. As is normally the case, the model will capture trends in the series but not focus on the seasonal part.

- **SARIMA**

  Seasonal Auto-Regressive Integrated Moving Average is an extension of ARIMA that supports the direct modeling of the seasonal component in univariate data.

All these are implemented by creating dataframe with columns ret_2 through ret_180, which are then split into train and validation sets to verify predictions.

These approaches didn't give expected results. This is due to stock prices not having a particular trend or seasonality. Instead it highly depends on what is currently going on in the market and therefore the prices rise and fall. As a result forecasting techniques like ARIMA, SARIMA, and Prophet will not show reliable results for stocks in general and especially with the dataset of this report. We also tried classification models like KNN. However as is seen in the data analysis section there was no significant improvement in RMSE due to the dependence of features on Feature_7.

4.4. Models

These are the different models we have built.

4.4.1. Multi-layer Perceptron Regression

It is a class of feedforward artificial neural network that generates a set a outputs from a set of inputs. It is a combination of multiple perceptron models. An MLP contains three layers of nodes: an input layer, a hidden layer, and an output layer. MLP makes use of backpropagation for training.

4.4.2. Linear Support Vector Regression

It is an algorithm that is used to solve regression problems. These problems involve the task of deriving a mapping function which would approximate from input variables to a continuous output variable.

4.4.3. Keras Regression

It is a type of supervised machine learning algorithm used to predict a continuous variable.

4.4.4. Extreme Gradient Boosting

XGBoost is an efficient implementation of gradient boosting that can be used for regression predictive modelling. It is a decision tree based ensemble machine Learning algorithm that uses a gradient boosting framework. XGBoost consists of base learners that uniform error, which are all combined so that the errors are cancelled out and gives the final prediction.

4.4.5. Random Forest Regression

It is a supervised learning algorithm that uses ensemble learning method for regression. The algorithm operates by constructing several decision trees at training time and outputting the average of prediction of all the individual trees. It has an effective method for estimating missing data and maintains accuracy even when the large portion of data is missing.

5. Discussion and Results

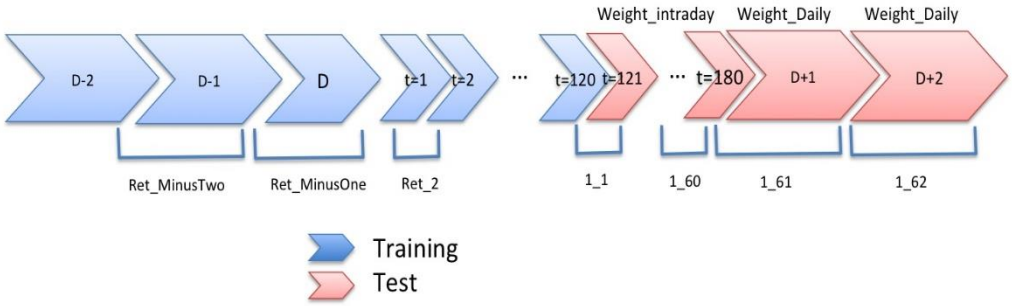
5.1. Dataset

Each row in the dataset is an arbitrary stock at an arbitrary 5-day window time. For each 5-day window, we are provided 25 features and 180 minutes of data (from t=1 to t=180). Training dataset consists of 180 minutes of data while test dataset has only the first 120 minutes. The columns that are present in the dataset are,

- **Feature_1 to Feature_25:** these are different features relevant to the prediction.
- **Ret_MinusTwo:** this column represents the returns from the close of trading on day

D-2 to the close of trading on day D-1 (one 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 (half day).
- **Ret_2 to Ret_120:** these are the returns over approximately one minute on day D. It is the return between t=1 and t=2.
- **Ret_121 t Ret_180:** intraday returns over approximately one minute on day D. (Target Variable)
- **Ret_PlusOne:** return from the time Ret_180 is measured on day D to closing of trading on day D+1. (Target Variable)
- **Ret_PlusTwo:** this is the return from the close of trading on day D+1 to closing of trading on day D+1. (Target Variable)
- **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)



5.2. Evaluation Metrics

The evaluation metric that we have used to measure the performance of our model is Weighted Mean Absolute Error and Mean Squared Error, and we made prediction on ret_60 feature using the models described above and got the following MSE and WMAE values

Weighted Mean Absolute Error: It is the mean of absolute values of the individual prediction errors on over all instances in the test set.

$$WMAE = \frac{1}{n}\sum_{i=1}^{n} w_i - |y_i - \widehat{y_i^2}|$$

where, w_i is the weight associated with the return

i, y_i is the predicted return

$\hat{y}_i$ is the actual return

n is the number of predictions

Mean Squared Error: It is the measure of average of square of errors i.e., average squared difference between the estimated values and the actual value.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

5.3. Results

We made prediction on ret_60 feature using the models described above and got the following MSE and WMAE values

S. No.	Model	No. of Epochs	Loss% (MSE)	Error (WMAE)
1	Linear SVR	**-**	6.1	5.1
2	MLP Regressor	**-**	6.3	5.3
3	Keras Regressor	10	0	0
4	Sequential Model	10	8.99	5.64
5	XGBoost Regressor	5	9.48	5.74
6	Random Forest Regressor	**-**	9.67	7.6

6. Conclusion

Using the models we have built in this project, we can predict the stock return for the interday and the intraday. Out of all the models we have built, Linear Support Vector Regressor and MLP Regressor provided us with the better results. We have secured the minimum value for mean absolute error by using both Linear Support Vector Regression and MLP Regression.

6.1. Directions for Future Work

It would be interesting to further investigate about ordinal vs nominal categorical variables, like coming up with a solution to auto identify them based n the distribution. The next step for our project could be, exploring and implementing a model based on layered Long Short-Term Memory (LSTM) networks. It is known that the older stock prices could potentially impact current stock prices. Consider an example, where a company got sued for some ill activity few months back and the company could still be experiencing depreciation in stock value. Due to some long-term dependencies, the use of LSTMs could significantly improve model performance.

Bibliography

1. Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques – Jigar Patel, Sahil Saha, Priyanka Thakkar, K Kotecha
   https://www.sciencedirect.com/science/article/pii/S0957417414004473?casa_token=epWYxnJFyDgAAAAA:yTitUkbuHNNRtXPP0FI0EbgcicI-RxMESrpV90CIeDDMc6PQnKQWfXn7YCW7kLD8GPCOzSwDYew
2. S. Zemke, “On developing a financial prediction system: Pitfall and possibilities,” Proceedings of DMLL-2002 Workshop, ICML, Sydney, Australia, 2002
3. Stock Price Forecasting by Hybrid Machine Learning Techniques – Tsai and Wang
4. Stock Price Prediction with help of python and fbprophet – Rakshith Ratan
   https://medium.com/datadriveninvestor/stock-price-prediction-with-the-help-of- python-and-fbprophet-prophet-library-part-1-3-f55ebff0b624
5. Aishwarya Singh. 2018. Build High Performance Time Series Models using Auto ARIMA in Python and R. (August 2018). Retrieved March 23, 2020 from https://www.analyticsvidhya.com/blog/2018/08/auto-arima-time-series- modeling-python-r/
6. https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/
7. https://www.youtube.com/playlist?list=PLvcbYUQ5t0UHOLnBzl46_Q6QKtFgfMGc3
8. https://github.com/ritvikmath/Time-Series-Analysis/blob/master/Stock%20Forecasting.ipynb
9. https://machinelearningmastery.com/multi-output-regression-models-with-python/