The Winton Stock Market Challenge

Sam Johnson

***Key Words*—Time series, data mining, stock market prediction, machine learning, forecasting**

I. Introduction

Predicting stock market returns has long been a challenging task for machine learning. The stock market has a large degree of random noise, which is caused by announcements, news, program trading, and emotional decisions. [3]

In this paper I compete in the *Winton Stock Market Challenge* [2], which is a Kaggle competition designed to evaluate one's ability at predicting stock market returns. The competition technically completed seven years ago but still allows late submissions. In this paper I describe my approach to the competition. Some of my approach was borrowed from the excellent notebook created and shared by Zonghao on Kaggle. [5]

II. Competition Description

Taken right from the *Data* section of the competition, here is the description of the prediction task:

"In this competition the challenge is to predict the return of a stock, given the history of the past few days.

We provide 5-day windows of time, days *D-2*, *D-1*, *D*, *D+1*, and *D+2*. You are given returns in days *D-2*, *D-1*, and part of day *D*, and you are asked 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. We provide 180 minutes of data, from t=1 to t=180. In the training set you are given the full 180 minutes, in the test set just the first 120 minutes are provided.

For each 5-day window, we also provide 25 features, *Feature_1* to *Feature_25*. These may or may not be useful in your prediction.

Each row in the dataset is an arbitrary stock at an arbitrary 5-day time window."

The image in figure 1 is taken from the competition and gives an overview of the data's structure.

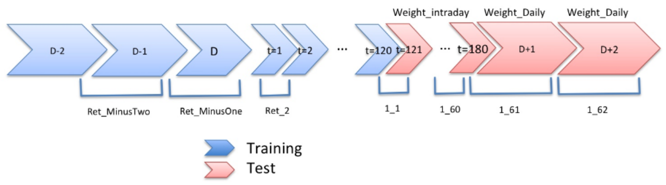

Fig. 1: Data Description

III. Evaluation

The loss function is given below:

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

Basically, we follow these steps:

1) Subtract the predicted return from the actual return.
2) Take the absolute value of the result.
3) Multiply the difference by some arbitrary weight known only by the competition designers.
4) Calculate the average loss over all predictions.

IV. Predicting Intraday Returns

The first task of the competition was to predict intraday returns. We are given 120 minutes of historical returns and asked to predict returns over the next 60 minutes.

In figures 2, 3, and 4 we display several examples of intraday returns. In each case, we mark the end of the known 120 minute period and the start of the unknown 60 minute period with a dashed red line.

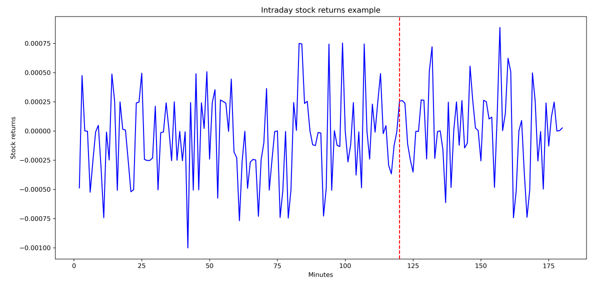

Fig. 2: Intraday Returns 1

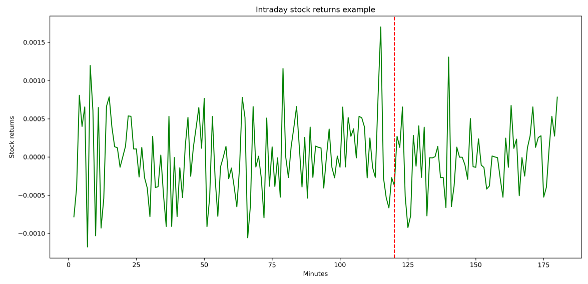

Fig. 3: Intraday Returns 2

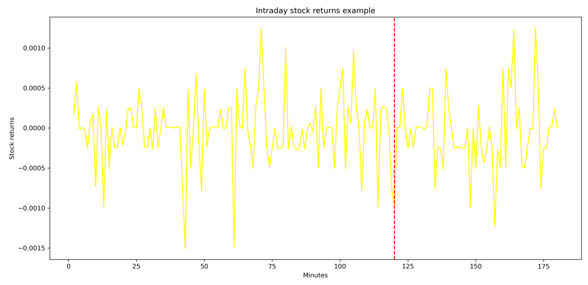

Fig. 4: Intraday Returns 3

In each case, we can see that the returns hover around zero and appear to be random. There are various forecasting models like ARIMA, RNN, or LSTM that could be considered for this task. However, there is also the added difficulty that all 60 return values must be predicted at once. These models aren't conducive to such a scenario. In the real world, we get return feedback every minute and would only have to predict the next minute rather than each of the next 60 minutes. Trying out one of these other models might make more sense in a real world scenario.

Due to the difficulties with predicting the next 60 minutes, we opt to simply predict a return of zero for each of the next 60 minutes. An example of this from *Intraday Returns 1* is shown in figure 5. We can see our predictions after the red dashed line, all of which are zero.

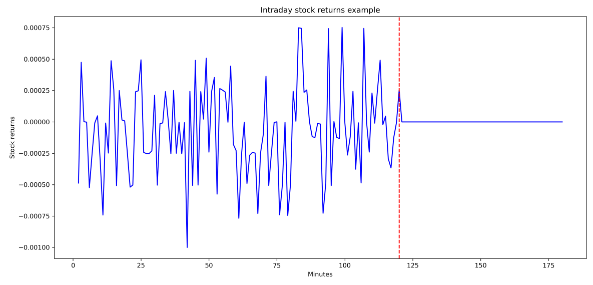

Fig. 5: Intraday Returns 1 with 0 Predicted

We also created a new feature for daily return prediction that is essentially the sum of the returns over the known 120 minutes of intraday return data.

V. Missing Values

In this next section we move towards daily return prediction. For this, we will use features *Feature_1* through *Feature_25*, whose labels are masked by the competition designers. Before performing daily return prediction, we need to preprocess and analyze the relevant features.

We begin by looking at the frequency of missing values in the dataset. In figure 6 we show the eight features with the most missing values. Given that other features have fewer missing values than these eight, this dataset really isn't too sparse. We deal with missing values by replacing them with the mean of the corresponding feature.

VI. Correlation Analysis

The next thing we do is look at the feature correlation matrix, which is shown in figure 7. It is

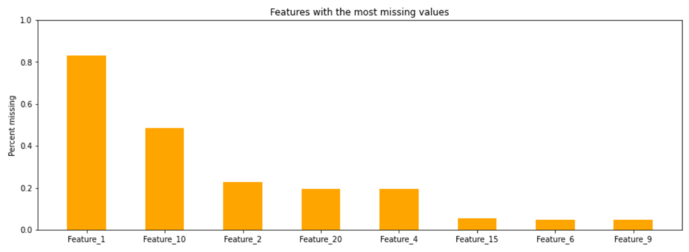

Fig. 6: Missing Values

shown in the form of a heatmap, with strong positive correlations shown in dark red and strong negative correlations shown in dark blue. Correlations of zero are shown in gray.

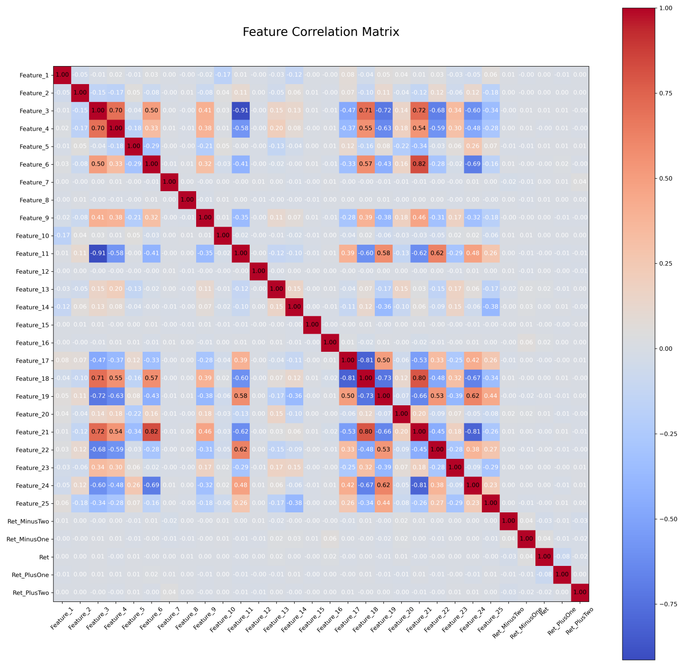

Fig. 7: Correlation Matrix

We can see that there is the occasional strong positive or strong negative correlation between features. If anything, however, there are far more extremely low correlations. In figure 8 we show the scatterplot between *Feature_6* and *Feature_21*. There is a rare strong positive correlation of 0.82 in this case.

What's concerning is that the correlations between our dependant variables; *Ret_PlusOne* and *Ret_PlusTwo*. We can view these correlations in the two last rows of the matrix. Many of these

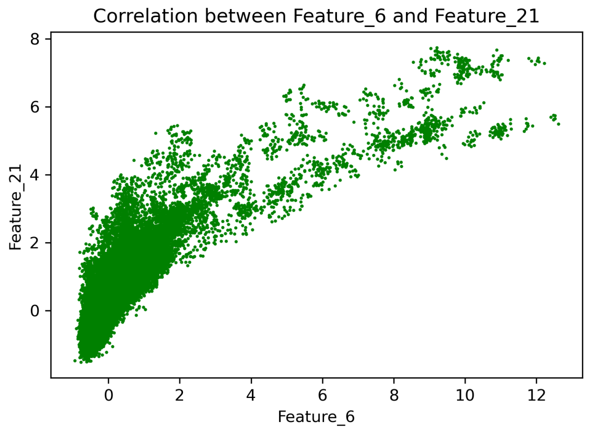

Fig. 8: Variable Scatterplot

correlation coefficients are zero while none of them reach greater than an absolute value of 0.08. This is concerning in that it will be difficult to predict our dependant variables using features that show no linear relationship with them. Nonlinear relations might still exist that aren't caught by the correlation matrix.

VII. Feature Selection

Because there were so many irrelevant features, I decided to perform feature selection and fit a final model using only the most useful features. I used the XGBoost [1] classifier to perform feature selection. It is somewhat similar to the random forest classifier. It works by fitting a certain number of decision trees. Each additional decision tree is fit on the error remaining after the prior decision tree's addition to the model [4]. After the model is fit, the individual trees can be looked at to determine how much information gain each of the features has contributed to the model.

I display XGBoost's calculated feature importances in figure 9. We can see that several features make up the bulk of total importance.

I opted to select the 15 most important features for the final model.

VIII. Results

Using my chosen features, I once again fit an XGBoost model on the training dataset to predict

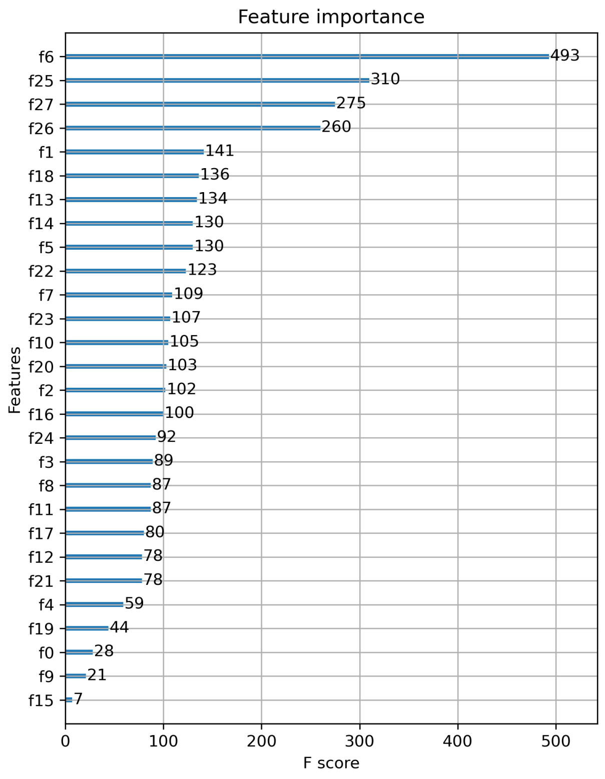

Fig. 9: XGBoost Feature Importance

Ret_PlusOne and *Ret_PlusTwo*.

I show the XGBoost predictions plotted against the actual values in figures 10 and 11. The first figure shows predictions for *Ret_PlusOne* while the second figure shows predictions for *Ret_PlusTwo*.
We can see that these predictions aren't amazing. In both figures, we can see that the actual values have a much greater range than the predicted values. This leads to poor prediction accuracy.

I tried a couple of other experiments with both the XGBoost model and the Random Forest Regression model with and without top 15 feature selection. I submitted the results to Kaggle for each of these experiments. A table of the results is shown in figure 12.

We can see that the Zero Baseline actually outperformed all of the models except for XGBoost

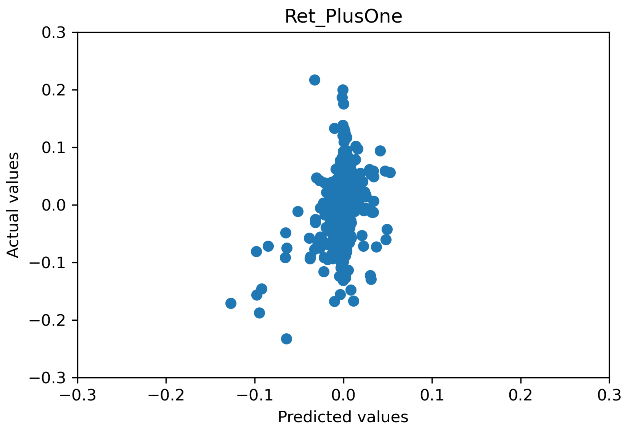

Fig. 10: XGBoost *Ret_PlusOne* Scatterplot

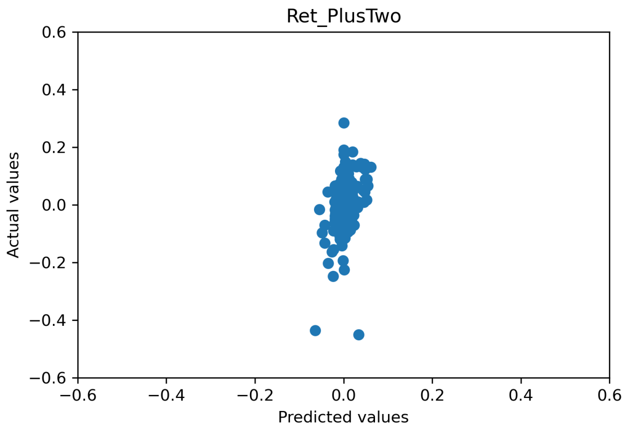

Fig. 11: XGBoost *Ret_PlusTwo* Scatterplot

with feature selection. Even then, XGBoost with feature selection only performed slightly better than the Zero Baseline and slightly worse than the competition winning score. Overall, the XGBoost model achieved a score that would have placed 50th on the private leaderboard, as of 12/09/2022.

IX. Conclusion

Here's a quick recap of what we've covered so far. We began by introducing the competition to predict stock market returns. We introduced the loss function as the weighted mean absolute error. We showed some intraday returns to demonstrate how difficult they were to predict. We moved on to predicting the returns over the next two days. We replaced missing values with the mean value. We looked at the correlation matrix, which was

Model	Top 15 Feat.?	WMAE
Zero Baseline	N/A	1728.62
XGBoost	No	1734.55
XGBoost	Yes	1728.24
Random Forest	Yes	1767.57
Random Forest	No	1761.4
Competition Winner	Unknown	1727.53

Fig. 12: Competition Results

concerning due to its lack of correlation with the target variables.

Finally, we fitted several models and found that the predictions, although good from a competition perspective, weren't accurate in any absolute sense. This leads us to the key takeaway that the features given by the competition designers weren't useful in predicting future stock returns. We might reasonably suspect that predicting stock market returns in general, regardless of features, is difficult, considering the large degree of random noise. Future work could include a search for better features outside of the competition to see if better prediction is possible.

REFERENCES

[1] Xgboost documentation.
[2] The winton stock market challenge, Dec 2015.
[3] Will Kenton. Noise, Jul 2022.
[4] Abhishek Mishra. How xgboost works, 2019.
[5] Zonghao. Predicting stock returns by xgboost, Sep 2020.