Ukrainian Catholic University
Risk Management, Spring 2021

Stock returns prediction

June 6, 2021

Sofiia Garkot, Kostyantyn Hrytsyuk, Oksana Matskiv

1 Introduction

The task of prediction of stock is very well known on the horizon of econometrics and machine learning. Although it is old and well investigated, a lot of investment companies are still in need of specialists who will accurately develop a model of a stock price in order to be able to choose the set of companies to invest in.

One of such investment companies is Winton stating itself as a data-driven investment management company. Recently (5 years ago actually) it has organized a sophisticated challenge on Kaggle [1] in order to find unique and creative approaches in the community and present to it the day-to-day challenges they face.

2 Data

The data was prepared by an offering investment company and divided into two subsets: the train and the test part. The targets are not given in the case of the test subset, that is why we cannot compare our approaches to the ones presented on the leader-board.

The target is defined as a vector of size 122 of returns presented for 1 stock on different time periods. The entries 1 to 120 present returns over approximately one minute on day 0. The entry № 121 of a target vector presents the returns from the last time period on day 0 to the close of trading on day 1 (next one). The last entry represents the returns from the close of trading on day 1 to the close of trading on day 2.

The data presented in this challenge describe the returns of a stock during a time period of 2 days before trading as well as on different timestamps on the day of trading (day 0). I addition, every stock is characterized by a set of 25 features relevant to the prediction. The characteristics are not described explicitly.

The performance of the participants is evaluated on the test dataset based on weighted mean average error.

3 Methodology

For predicting returns on stocks, the following models were applied:

- Penalized linear regressions (Ridge, Lasso, Elastic Net),
- Random Forest,
- Gradient Boosted regression trees,
- Support Vector Machines,
- Neural networks: shallow vs. deep.

To compare the results with the leader-board we used the same metric as the authors of a dataset propose - Weighted Mean Average Prediction Error ($MSPE$) and R^2_{oos} were computed as follows:

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

However, due to unnormalized weights randing from .. to ... we decided to additionally present the following metrics :

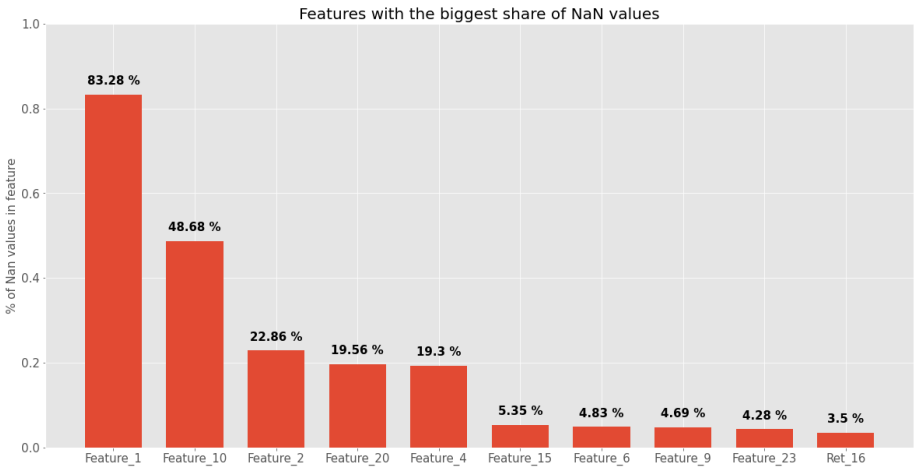
- MAE: Mean Absolute Error
- WNMAE: Weighted MAE with normalized weights
- RMSE: Root Mean Squared Error
- R2 score
- MAE: Mean Average Error

Due to the lack of response variables in the test subset on the challenge we were not able to compare our metrics to the one on the leader-board. However, we decided to average our performance on 5 folds for the Neural Networks approach. Every fold contains 80 percent of training data for training a model and 20 percent for its evaluation.

For all other models we used train/test random split of data in proportion 80/20. After we applied 4 folds cross validation for train set. Test set was used for models evaluation.

3.1 Handling missing values

The biggest challenge in this task was presence of NaN values in data. They are presented both in explanatory and predictor variables. Below, we are showing top 10 columns with the biggest share of NaN values.



To deal with this challenge, we preprocessed our data with such a technique as Imputer. It is a sklearn built-in tool [2] that replaces the absent values with actual ones based on a predefined algorithm.

We used two variations of this tool:

1. Simple Imputer with "mean" strategy [3]

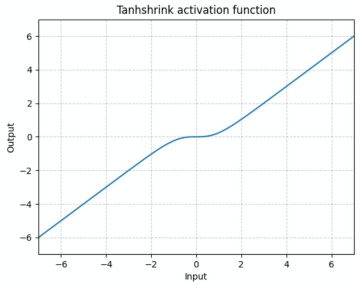
   This algorithm is pretty straightforward. It replaces all NaN values in the column by the mean value of all available values. The drawback of this approach is that all absent values will be replaced with the same constant. And it can affect the results of future predictions.

2. KNN Imputer [4]

   This version of Imputer is based on K Nearest Neighbors algorithm. For each missing value it computes the mean value of K non-NaN Neighbors. Using it, we are taking into account variance that is present in each feature.

3.2 Neural Networks

We tested variety of architectures in order to choose the one that performs the best. On this task shallow neural networks performed better than deep. The number of input neurons equals to 161, the number of hidden states equals 1 and is of width 128, the width of output vector is 22. The activation function of choice is Tanhshrink that has the ability to smooth the data around 0.



Due to computational and time limitations we used only 50 epochs for training. Also, we observed the trend for neural network to overfit, that is why we used early stopping technique with patience indicator equal to 10.

3.3 Penalized linear regressions

The difference from the OLS is that in penalized linear regression we are using penalizing with norm. The weight of this norm is defined by the parameter α, which is used in hyper parameter tuning for all 3 models.

3.3.1 Ridge

In Ridge regression L2 regularization is used. Also, for this model in parameter tuning, we used next parameters:

- Solver - algorithm used for the computations;
- Fit Intercept - boolean flag to define whether to fit intercept or not ;
- Normalize - if Fit intercept is true, then this boolean flag tells whether to normalize regressors or not .

3.3.2 Lasso

In Lasso regression L1 regularization is used. If in this model $\alpha = 0$, then it will be equal to OLS. For this model in parameter tuning, we used only α. $FitIntercept$ and $Normalize$ are predefined as $True$

3.3.3 Elastic Net

Elastic Net - is the combination of two models above. The impact of each of them is defined by parameter $l1_ratio$.

We used range $[0.95, 1]$ for this parameter.

Table 1: Averaged results on 5 folds.

Method	WMAE	WNMAE	RMSE	R2	MAE
Neural Networks	34.46	0.0	0.05	-57.93	0.01
Best leaderboard model	1727.53575	-	-	-	-

Table 2: Averaged results with 4 CV folds and 1 test fold. KNN Imputer

Method	WMAE	RMSE	R2	MAE
Ridge	1771.09	0.0045	-0.0001	0.0011
Lasso	1771.25	0.0045	-0.0001	0.0011
Elastic Net	1771.25	0.0045	-0.0001	0.0011
Decision Tree Regressor	1765.6	0.0044	0	0.0011
Random Forest	1664.36	0.004	0.15	0.001
Best leaderboard model	1727.53575	-	-	-

4 Results

The performance of Neural Networks is compared by upper described metrics to the best score gained on the leader-board during the challenge.

5 Conclusion

In our research, we aimed to highlight available tools and technologies for solving problems presented in challenge [1].

After completing this work, We can state that the best model is Random Forest in a combination of preprocessing data with KNN Imputer and PCA dimensionality reduction.

As an area for opportunities for future enhancement of our work we see next things:

- Working on new algorithms for filling NaN values

  We used Imputers to fill missing values based on column values. Since in predicted values we always have time series, we can use row-based mean value to fill NaN. In that case, we will avoid cross-stock influence.

- Since we know that tree-based models provide the best results, we can make deeper investigation into models and their parameters. Due to the big number of parameters to tune, there is a possibility to obtain better results using other hyper parameters.

- In our work, we tried to apply Gradient Boosting Regressor and Support Vector Machine. In some cases, we faced with errors raised by the sklearn API. A better understanding of the nature of those errors and knowledge of how to deal with them.

In the conclusion, we can state that there is a number of ways to optimize obtained results and provide better scores for the initial problem.

6 Appendix

References

[1] Winton. The winton stock market challenge. *https://www.kaggle.com/c/the-winton-stock-market-challenge/overview/description*, 2016.

[2] Imputer. *https://jaquesgrobler.github.io/online-sklearn-build/modules/generated/sklearn.preprocessing.Imputer.html*.

[3] SimpleImputer. *https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html*.

[4] KNNImputer. *https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html*.