Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 745–749
DOI: 10.15439/2024F6695
ISSN 2300-5963 ACSIS, Vol. 39

Searching Stable Solutions For Stock Predictions: A Stacking Approach

Ty Gross, Arthur Allebrandt Werlang, Apeksha Poudel, Julian Roß
Technische Universität Dortmund
Department of Computer Science
Dortmund, Germany
{ty.gross, arthur.werlang, apeksha.poudel, julian.ross}@tu-dortmund.de

***Abstract*—The goal of the competition is to predict stock positions for holding, selling or buying stocks of companies from the S&P 500. Firstly the data is read in and the missing values are imputed with the median. Categorical data is one-hot encoded. A classification approach with mainly tree based methods is used. The models used are HistGradientBoosting, XGBoost, MLP and SVC whose parameters are chosen and modified through a grid search. For the stacking the models' prediction results are summed up and the result is mapped to the three positions. It is found that the result is a bit overfitted to the competition's test data which makes sense in regard to it being a competition. The stacking improves the score drastically. Concluding it can be said that machine learning models can hint in the right direction when it comes to handling stocks but fail at giving good financial advice.**

I. Introduction

THIS paper concludes the results of team "TUmany Data" in this year's Knowledge Pit data mining challenge titled "FedCSIS 2024 Data Science Challenge: Predicting Stock Trends".[1] A given dataset of the S&P 500 index consisting of financial indicators of 300 companies for the last 10 years has to be analysed. The results are predictions of buy, sell and hold positions. Firstly the preprocessing of the data has to be done. This includes the reading in, imputation of missing values and the encoding of the dataset. A median imputation method is used and the values are one-hot encoded.

In the second step, the data is analysed and the predictions are evaluated. Different classification and regression models are tested but it is noticed that the performance of classification boosting and classification tree models are better than the regression models. So based on their performance, only 4 of the models i.e HistgradientBoostingclassifier, XGBoost, MLP Regressor and SVC are finalized for data imputation and evaluation. The parameters for those models are tuned using Grid Search algorithm to find the optimal parameters. The models are then evaluated by first using the cross validation and then the error function provided by the competition.

In the third step, the results are stacked to refine the predictions and to better the score. For the stacking the sum of the different models is taken and the result is mapped to the three decisions of buying, selling and holding the stock. Because of the given penalty for wrongly predicting a stock, the mapping is skewed towards the holding position because it has the overall lowest cost.

[1]The challenge was held on the knowledgepit.ai platform. It was organised by the Conference on Computer Science and Intelligence Systems (FedCSIS) and sponsored by Yettel.Bank as well as the Conference on Computer Science and Intelligence Systems series. The authors did not benefit financially or in form of other endorsements from this challenge. For more information about the challenge, see [1]

II. Methods

In this paper, Python is primarily used as the programming language of choice due to its wide range of library support. The Pandas and Numpy libraries [2] are used to store and process the data and are commonly used in data science. Additionally, the Scikit-learn library [3] provided many of the model implementations used in this work. During initial experimentation, the MissForest imputation library is also utilized with minor changes to fix function names that have changed in its dependencies. However, this method proved to not be effective and is not included in the final prediction models (see III-B) [4].

III. Preprocessing

Before the dataset can be evaluated, it must first undergo a preprocessing step to transform the data into a format that can be evaluated easily. This section highlights the properties of the given dataset, as well as encoding and imputation techniques performed on the dataset during the preprocessing stage.

A. Dataset

The dataset for the competition consists of an 8000-line training CSV file, a 2000-line unlabeled testing data CSV file, and dictionary datasets that have names for the column names. In its initial state, the given dataset has two types of missing values: “empty” and “NA”. Each row of the data represents one financial statement. The columns contain 58 key financial indicators and a perform and class column that can only be found in the training data and not the test data. The class column contains three classes, "sell"/"hold"/"buy", labelled as "-1"/"0"/"1" accordingly.

Figure 1 gives a visual representation of the information stated above. From the graph, it is clear that some columns have a significantly higher number of NA/missing values compared to empty values. Columns such as I21, I48, I50, dI21, dI48, and dI50 exhibit particularly high counts of NA/missing

values. This visualization helped to identify which columns in the dataset required more attention for data imputation due to their high number of missing entries.

The financial indicators that comprise a majority of the columns are numerical values that are read in as floating point data types. The only column that is not numerical is the group column which has categorical datatypes, representing which financial sector the statement is from.

B. Data Imputation

There are multiple imputation methods for imputing missing values. In this section, a brief analysis is done to determine the best imputation method for this particular dataset. For that, a handful of imputation methods are tested against a few machine learning algorithms. The results are measured with the testing error described in the challenge (see [5] and [1]). The following imputation methods are analysed:
Imputation with mean, median, mode, random values, using a missing forest regression as well as disregarding all missing values.

The following machine learning algorithms are used to compute the testing error:
Decision Tree Regressor, Random Forest Regressor, SVR, Decision Tree MLP, Decision Tree Gradient Boosting, Bayes Ridge Regressor, Gradient Boosting, Hist Gradient Boosting, Decision Tree Methods, Ada Boost Regressor, Bagging, Gaussian Naive Bayes Negressor, SVC and an MLP.

The results of this testing matrix can be found in graphic 2. It can be seen that the tree-based methods generally perform better than the non-tree-based methods. This finding is independent of the choice of the imputation and is further built on in the evaluating process (see section IV). The decision tree gradient boosting had the best results with the imputation method mean and missing forest. When comparing the imputation methods, it can be seen that the random imputation and the mode imputation perform the worst. This is unsurprising for the random imputation. The bad result for the mode can be explained by the random-like imputation of the values. This is because most of the values occur once and are not correlated in any way with the missing values, thus being random-like. Disregarding data on a large scale generally makes the evaluation of the final model worse, and thus, this approach is not further pursued. The last three remaining imputation methods do not differ much. All three methods of regression, median, and mean, calculate similar values based on the existing values. Because of the simplicity, in the following evaluation, the median is used as an imputation method.

In addition to imputing the missing values, indicator columns are added to the dataset to indicate whether a value are imputed or not. This ensures that the model does not lose information about whether a value is missing when it is making its prediction.

C. Encoding non-numerical data

Since many models require numerical data to function, as the first step, the categorical values have to be encoded numerically. A one-hot encoding method is used for the "Group" column, which converted the previous string values into an indicator column for each possible group. In the indicator column, if a financial statement belongs to a group, it is marked with a one in the group's column; otherwise, it is set to zero. This ensures that ML models requiring numerical data function properly on this dataset while the information about the "Group" column is maintained.

IV. Evaluating models

Once that dataset is put through the preprocessing stage, it is ready to be fed into various models. In this section, the various models and tuning methods are introduced. Then, the cross-validation method used is discussed. Finally, the evaluation metric is used to determine how well a model performs is outlined.

A. Classification vs Regression / models

Different Classification and regression models, such as Gradientboosting, Decision Tree, ADABoosting, MLP, XGBoost, Missforest, Linear Regression, etc., are applied to impute and evaluate the data. It is observed that classification tree-based models and boosting models delivered superior performance than the regression models and thus are selected to be further tuned.

Based on the performance, following models are used for further analysis:

- HistGradientBoosting Classifier [6]
- XGBoost [7]
- MLP Regressor [8]
- SVC (Support Vector Classifier) [9]

The hyperparameters of these models are tuned using the Grid-Search. Grid search is a traditional method of hyperparameters optimization, which simply makes a complete search over a given subset of the hyperparameters space of the training algorithm. In other words, the grid search algorithm is a complete brute-force and takes a too long time to execute [10].

Table I presents the parameter grid that are used and the best parameters are obtained from grid search for the above given evaluated models. This comparison highlights the specific hyperparameters and their optimal values that resulted in the best performance for each model, facilitating our further analysis and evaluation of error rates.

B. Cross-validation

For the evaluation of the models, a five-fold cross-validation is used. This means that the training data is split into five equally large test train splits. Using cross-validation helps reduce inconsistencies that can happen when data is split in an inconsistent way. For example, fitting a model and evaluating it against the outliers will have a much different outcome than fitting the model to the outliers and testing against the normal data. Thus, the variance for the testing error is reduced.

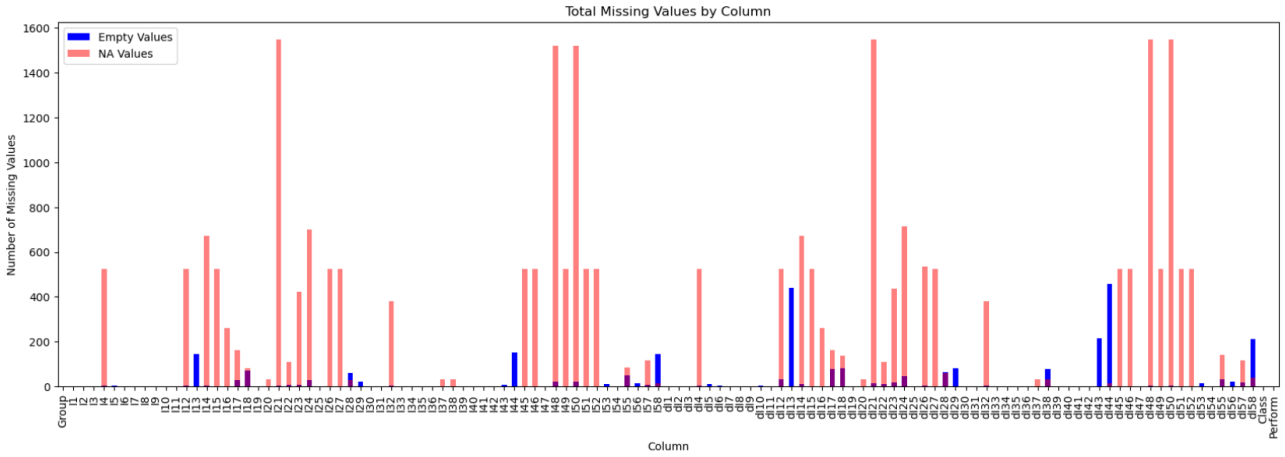

Fig. 1. Total missing values by column

TABLE I
PARAMETER METRICES AND BEST PARAMETERS FOR ALL EVALUATED MODELS.

Model	Parameter	Values Tested	Best Value
HistGradientBoostingClassifier	Learning rate	0.001, 0.01, 0.05, 0.1	0.001
	Max Depth	5, 7, 10, 20, 50	20
	Min samples leaf	5, 10, 20, 30	10
XGBoost	Learning rate	0.01, 0.1, 1	0.01
	Max Depth	2, 5, 10	5
	Number of Estimators	100, 500, 1000	100
MLP	Hidden layer sizes	(116,232,116), (116,116), (116, 232), (116)	(116,232,116)
	Activation	logistic, ReLU, tanh	logistic
	Solver	adam	adam
	Alpha	0.0001, 0.005	0.005
	Learning rate	constant, adaptive	constant
	Learning rate init	0.001, 0.005, 0.01	0.001
SVC	C	0.5, 1, 2, 5	1
	Kernel	linear, poly, rbf, sigmoid	rbf

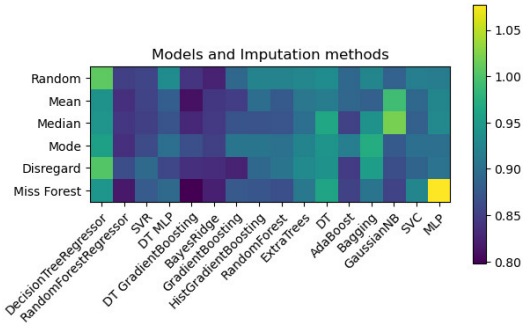

Fig. 2. Results of the imputation analysis. Lower is better.

C. Evaluation (Calculating Error)

The same error function provided by the competition is used to evaluate the models. The error function is calculated based on the matrix in Figure 3. When the predicted class matches the expected class, zero is added to the error, predicting one off from the expected class results in one added to the error, and predicting two off from the expected class adds two to the error. Finally, the error function divides the confusion matrix output by the number of predictions to get the average error. The decision to use the same error calculation as the competition is made to ensure that the same function is being optimized by the models as the competition's scoring.

	-1	0	1
-1	0	1	2
0	1	0	1
1	2	1	0

Fig. 3. Confusion Matrix showing error for predicted vs. actual class value.

V. Stacking

A method of model stacking is used to optimize our further predictions. Model stacking allows for merging the predictions of multiple types of models into one final output by feeding the output of previous models into another model [11]. By stacking multiple models with different parameters and imputation methods, an incorrect prediction by one model can be mitigated by the other models being stacked. This can, however, also lead to worse predictions in the case in which one model is poorly chosen, skewing the overall results to being incorrect.

A. Choosing Models for Stacking

After evaluating multiple models, both using the training data and cross-validation, and the provided evaluation from the submissions to the competition website, different models are picked based on their performance.

Based on those factors, the best-performing models are chosen to be stacked, going from the assumption that the better-performing models would result in the best possible stack. The models that performed the best on the competition leaderboard and against the training data are the HistGradientBoosting Classifier, XGBoost, and an MLP Regressor.

B. Stacking the Models

Different stacking methods are tested - firstly the mode from the results of different models was taken, and where more than one mode is found, the prediction is set to zero. The intention behind this is to minimize the error function, as wrong hold predictions are less costly than wrong buy or sell predictions. Based on that, a new method, using the sum of many different results was devised.

It is noted, during a trial and error phase stacking the models, that a more robust method for the stacking is needed. As a result, based on the fact that the sum of n model predictions would go from $-n$ to n and could then be mapped, using the Pandas map function, so that results close to zero - meaning the models either agreed on the zero prediction or disagreed heavily - were set to zero, while results close to $-n$ or n, to -1 or 1 respectively. Since the competition score is calculated based on how far away the prediction is from the actual value (see 3), choosing zero when models disagree results in a lower penalty for an incorrect prediction. This method also considers all models equally as important, so no models are favoured, not even the better-performing ones.

In a practical example, when stacking six models, one would get results varying from -6 to 6. From these results, any values between -2 and 2 are set to 0, while values equal or above 3 and below -3 are set to 1 and -1 respectively. This way, it is ensured that for any buy or sell decision, at least 50% of the models agreed on that decision, which increased the accuracy of the predictions on the leaderboard substantially.

VI. Conclusion

While preprocessing the data, it is found that the imputation method only plays a minor role when evaluating this dataset. After a brief analysis, a median imputation is decided.

The final evaluation for the submitted predictions is a preliminary score of 0.7030 and a final score of 0.8304. The final stacking of the predictions relied on evaluating our models based on their leaderboard performance since the limited training data might not be an accurate representation of the final testing data. In hindsight, however, picking the models based on their leaderboard score might have led to the end results of an overfitting of the specific part of the test dataset that was used to calculate the error. This explains the discrepancy between the initial results and the end scores.

Further analysis should experiment with different imputation methods for missing and NA values rather than treating all of these values as the same. Additionally, the indicator columns used to mark missing/NA values should be modified to specify whether the imputed value was previously missing or NA.

For a more extensive research a regression approach with the "perform" column could be done. This would give the model more degrees of freedom to predict the score and to make a decision based on that score.

Generally, it can also be said that predicting stocks is a very hard task and can not be reliably done. Events that influence the stock market in politics and many other fields that are part of the global economy can not be foreseen. Therefore, the prediction on the basis of old data can not account for such events and changes. For a given stock, a volatile score could be assigned that captures the dependency on global events and represents its stability in changing times. For this, the dataset could be extended with additional data. For example, a score that captures the change for each stock regarding similar events to feed the machine learning algorithms the dependency for specific fields. For this reason, outside data could also be integrated into the dataset to help improve the predictions since other events outside of the financial world might have an influence on the stock's value.

Acknowledgment

We would like to thank Prof. Emmanuel Müller, TU Dortmund University and the Research Center Trustworthy Data Science and Security, as well as our tutors Simon Klüttermann, Michel Lang, Steffen Maletz, and Jonas Rieger, who introduced us to the challenge and supported us throughout the competition. We'd also like to thank FedCSIS for organizing the challenge and for inviting us to write this paper.

References

[1] A. M. Rakicevic, P. D. Milosevic, I. T. Dragovic, A. M. Poledica, M. M. Zukanovic, A. Janusz, and D. Slezak, "Predicting stock trends using common financial indicators: A summary of fedcsis 2024 data science challenge held on knowledgepit.ai platform," in *Proceedings of FedCSIS 2024*, 2024.

[2] T. pandas development team, "pandas-dev/pandas: Pandas (latest version)," *10.5281/zenodo.3509134*, Apr. 2024.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] Y. S. Y. Hindy, "Missforest (version 2.5.5)." https://pypi.org/project/MissForest/, Mar. 2024.
[5] FedCSIS, "Data science challenge: Predicting stock trends," 2024.
[6] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.
[7] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, ACM, Aug. 2016.
[8] G. E. Hinton, "Connectionist learning procedures," *Artif. Intell.*, vol. 40, pp. 185–234, 1989.
[9] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Adv. Large Margin Classif.*, vol. 10, 06 2000.
[10] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.
[11] B. Pavlyshenko, "Using stacking approaches for machine learning models," in *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pp. 255–258, 2018.