Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 751–754 ISSN 2300-5963 ACSIS, Vol. 39
DOI: 10.15439/2024F6884

# Experimenting with manual and automated data mining pipelines on the FedCSIS 2024 Data Science Challenge

Luisa Buck, Marc Furier, Okan Mert Göktepe, Jusztina Judák, Max Lautenbach, Gregor Münker
University of Mannheim
Mannheim, Germany
Email: {luisa.buck, marc.furier, okan.mert.goektepe, judak.jusztina, max.lautenbach, gregor.muenker}
@students.uni-mannheim.de

***Abstract*—This paper reviews the 5th-best solution and results of the FedCSIS 2024 Data Science Challenge, which aimed to predict stock trends using financial indicators. It details the preprocessing, modelling, and tuning approaches and demonstrates, as well as the methods and techniques used to address the prediction problem effectively. Subsequently, the results of different experiments, including hyperparameter optimization on preprocessing steps and switching between different prediction targets, could be compared to manual experiments. Overall, a manually experienced model could be found to outperform hyperparameter-tuned pipelines.**

## I. Introduction

PUBLIC data science competitions and benchmarks support companies in deciding on today's data science methodology. They show the best-of-breed data science methods in specific fields and compare the most recent methodologies [2]. This year's FedCSIS 2024 data science challenge[1] targeted the financial markets. The challenge aimed to predict stock trends based on given financial indicators and one-year data per stock. The stock trends were predicted as three classes, which makes up a classification task. The given financial indicators are also represented in a tabular way, which opens up a variety of data mining methods.

The following paper will review our team's proceedings and results, "Pattern Pioneers", scoring 5th in the overall competition. As the team consisted of six members, various data mining methods were used to get the best scores. This paper aims to review the applied methods and share the overall experience of the application. The applied methods range from simple statistical methods like Naive Bayes to challenge-winning methods like XGBoost. It also includes the methodology of stacking as well as various preprocessing steps. Hyperparameter optimization was also utilized in the challenge but with the addition of including the whole data mining pipeline within the search space. TPOT and FLAML were included during the competition to utilize and compare automated machine-learning approaches.

The paper will be structured as follows. The first part will include a challenge review, data inspection, preprocessing, and modelling. Afterwards, the results section will discuss how different methods were used in the given challenge. The work on the results was part of the class "Data Mining II". Therefore, the number of methods tested and the time for work were limited, limiting the time for exploitation of different approaches.

[1]https://knowledgepit.ai/fedcsis-2024-challenge/

## II. Data Mining Process

### A. Challenge Review

The FedCSIS 2024 Data Science Challenge's topic was predicting stock trends buy, sell, and hold, making the data mining problem a classification problem. The dataset consisted of 300 S&P 500 companies, their stock trends and financial indicators collected over multiple years from the companies' financial statements. The only information given about the companies was their industry, but there was no information about the particular company. Therefore, the only data source that was legally usable was the FedCSIS 2024 dataset. The submissions to the challenge were scored on the stock trend with a type of mean absolute error, which could only be implied as the three stock trends were encoded numerically. Further information can be found within table I.

### B. Data Inspection

The given data consisted of 10000 data points with the industry, 58 financial indicators, 58 1-year deltas of the financial indicators and two target columns. On the one side, the target column "Class" contained the stock trend, which was also the one in charge of the challenge leaderboard. On the other side, another column named perform was included in the dataset. This column was a risk-return performance measure. In the early part of the project, it was visible that the following rule set could discretize this risk-return measure:

This strict rule has already opened up a regression problem that could be used instead of classification. In addition, as the column "Class" was already encoded in -1, 0 and 1, the initial classification problem could also be transformed into a regression problem. On Figure 1. we can see the distribution of the instances based on the Performance attribute, colored by the actual class from the training data.

TABLE I
PERFORM DISCRETIZATION RULES

| Perform | Class |
|---|---|
| $x < -0.015$ | Sell |
| $-0.015 < x < 0.04$ | Hold |
| $x > 0.04$ | Buy |

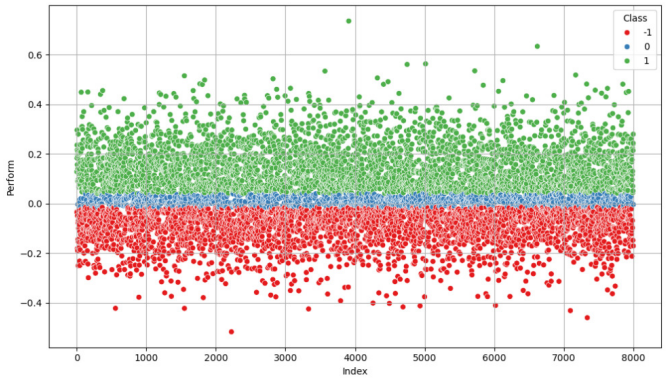


Fig. 1. Performance colored by Class

All in all, this resulted in a total of 117 influencing columns. The dataset consisted of 116 numeric and one categorical influencers. The categorical influencer industry consisted of eleven distinct values like energy or health care and therefore had no ordering, which makes it a nominal variable. FedCSIS has already provided a train-test data splitting. The split was an 80-20 split with no indication of shuffling or the timestamp of the collected data point. The test split was not consisting any of the performance-based columns "Class" or "Perform". Instead, this dataset was meant to define a live leaderboard. In addition, an unknown subset of the test set was used to evaluate the submission's overall performance.

Within the dataset description of FedCSIS, missing values are already outlined. There were two types of missing values in the dataset: one is non-available, and the other is non-applicable. Further information on which values remarks on which type of missing value was not conducted. When concentrating on the training dataset, only 5194 rows do not contain any missing value. When changing the axis, 75 columns contain at least one missing value. In addition, the most missing values are 1564, with a mean of 243 per column. This implies that the missing values are conducted by more than one column. Instead, they are spread across the whole dataset.

### C. Preprocessing

The following various preprocessing steps will be outlined, as those were used in different approaches, leading to the result of our work. Except for the outlier removal, all preprocessing steps were used within Python 3.9 and the package scikit-learn[2]. The outlier removal was done by using the package imblearn[3]. The data inspection already leads to a couple of implications. First, as only one dataset contains every needed influencer and target column, no data transformation is needed. As mentioned, there was no legal possibility of extending the database through other data sources, such as stock price, because there was no inference on the particular company.

To evaluate the dataset at scale and not exceed 500 submissions, a private 80-20 split was done before the preprocessing. This relied on the 80-20 of FedCSIS. All shown pipelines of the later results were trained again on the full training dataset to use a maximum number of data. The missing values are an issue of the dataset that must first be resolved on the split dataset. Therefore, an imputation or deletion could be considered. As already outlined, the missing values are highly probable to be spread across the dataset; a deletion leads to a row loss of approx. 35%. In addition, this approach invokes the question of how to handle missing values in the test sets. Therefore, the imputation approach was chosen.

As already outlined, the goal of the work was to achieve results within a small given time frame; this work only contained simple imputation methods. On the one side, missing values were imputed with the mean and median of the remaining values of a column. On the other side, missing values were replaced by 0 or 999. The number 999 was intended to work as an outlier, marking a value as missing. In manual experiments kNN imputing was also tested.

In the second step of the preprocessing, the data was scaled. Therefore, a min-max scaler, the more robust standard scaler [1], and the quantile range-based robust scaler [4] were used in experiments. In this step, it is mentioned that the imputation with 999 will change the underlying distribution. Any effects of this will have to be considered during the evaluation of the results. This data scaling was done to all 116 numerical influencers. In addition, the industry encoded strings. As many data mining algorithms, especially numeric ones like linear regression or neural networks, do not support categorical values, the industry had to be encoded numerically. As this column contained nominal values, one correct way is to use a one-hot encoding, especially because there are only eleven distinct values [3].

In order to create a model later with good generalization capabilities, the outlier removal and feature selection steps were integrated after the necessary preprocessing steps. As the dataset contained 117 influencers, manual outlier detection based on outlier plots was not considered. Concerning the efficiency of the work, an isolation forest was utilized. A simple f-test based on linear regression, correlation or wrapper methods were used for the feature selection. This was wrapped into a k-best selector method. On classification approaches, additionally, sampling techniques like SMOTE were used.

[2]https://scikit-learn.org/stable/
[3]https://imbalanced-learn.org/stable/

### D. Modelling

The core of the data mining pipeline remained the models and their settings. As already outlined, two types of problems could be solved within the challenge. On the one hand, there is regression and, on the other hand, a classification problem. Therefore, applicable models had to be chosen for each of the problems. Nevertheless, the models used, namely XGBoost, LightGBM, Random Forest and Support Vector Machine, supported both problems. A logistic regression was used in the first experiments, which is only capable of classification problems.

To solve the problem, three evolution steps of the pipeline modelling were done. First, manual experiments were used to create a baseline and an initial experience for the models and preprocessing on the dataset. Second, fully automated experiments were utilized to get good results efficiently. In this step, the results rely on TPOT[4] and the package FLAML[5]. TPOT can cover the preprocessing steps so that no manual experiments with the preprocessing and TPOT are done. In contrast, the package FLAML only provides hyperparameter tooling that tests different models. The tweak in FLAML against other automated machine-learning packages is that FLAML prioritizes fast-processing hyperparameters. Therefore, FLAML fits rather well to gather good results in a time-efficient process. The third and last evolution step utilized hyperparameter tuning. All of the tuning within the work was done on a 5-fold cross-validation to maximize the generalization capabilities of the model. This was important because the dataset for the final ranking was unknown. The decision was to use Bayesian tuning to follow the idea of efficiency. Within experiments, GridSearchCV and RandomSearchCV were also performed. The underlying search spaces were chosen iteratively.

In addition to the hyperparameter of the models, all preprocessing steps were included in the tuning. Therefore, the preprocessing could be split into necessary and optional steps. The necessary steps are the scaling and encoding. For those steps, the search space consists of the presented ones in subsection II-C. The optional outlier removal and feature selection steps could be deselected within the hyperparameter tuning. When they were selected, the search space of the contamination $c$ of the isolation forest was defined as

$$0 < c < 0.5; c \in \mathbb{Q} \tag{1}$$

and the search space of the $k$ in k-best feature selection as

$$5 < k < 120; k \in \mathbb{Z}. \tag{2}$$

With this step, a small automated machine-learning pipeline could have been built. One minor addition was also introduced within this third evolution step: stacking the best models. This was related to good empirical performances in prior Kaggle competitions.

[4]https://epistasislab.github.io/tpot/

[5]https://microsoft.github.io/FLAML/

## III. Results

Throughout the project, we experimented with various models. In this section, we will discuss the three evolution steps of the pipeline modeling.

The first test was based on logistic regression, excluding NA and NaN values, which achieved an accuracy of 0.49 and an F1-Score of 0.45, with a score of 0.88 and no successful predictions for class 0. Imputing NA values with the median and deleting NaN values slightly reduced accuracy and F1-Score to 0.48 and 0.44, respectively. When both NA and NaN values were imputed with their group or column medians, accuracy improved but the F1-Score remained unchanged. Outlier detection using Isolation Forest improved accuracy to 0.50, F1-Score to 0.46, and reduced error to 0.85. Applying SMOTE for handling class imbalance reduced accuracy to 0.46 but improved error to 0.77, enabling predictions for class 0. Further experiments with feature selection based on SelectKBest and removing highly correlated features reduced the score to 0.74 and decreased accuracy and F1-Score to 0.43. The FedCSIS leaderboard showed a score of 0.84. Further techniques like PCA and SelectFromModel did not improve results.

We experimented with Support Vector Regression (SVR) for performance prediction due to its robustness to outliers and effectiveness with high-dimensional datasets. The best results were obtained with rbf kernel, Gamma set to 'scale' and C set to 0.1, resulting in a validation score of 0.89. However, when applied to the submission dataset, the score dropped to 0.79. Switching from median to K-Nearest Neighbors (KNN) imputation with two neighbors did not enhance performance, resulting in a score of 0.84. We also detected the importance of the Group attribute regarding the performance so we introduced group-based approaches. Group-based preprocessing, including mean, median, and KNN imputations, and numerical attribute scaling within groups, yielded the best result with the score of 0.80, though still lower than the initial preprocessing strategy. Despite extensive experimentation, combining OneHotEncoding, median imputation, and standard scaling, alongside an SVR model with gamma set to 'scale' and C set to 0.1, provided the best results.

For XGBoost, The final model, trained with a learning rate of 0.05, 500 estimators, and a maximum depth of 5, achieved an F1-Score of 0.61 and a validation cost of 0.5956. The submission dataset score was 0.7822. Applying SMOTE to address class imbalance and using SelectKBest for feature selection did not yield improvements, leading to the exclusion of these steps. Furthermore, we experimented with new feature generation. These new features were created based on domain-specific knowledge and combined with existing features to enhance the model's performance, but the scores worsened. Therefore, this step was also excluded.

In addition, the Random Forest approach was utilized in manual experiments. The best classification score on the FedCSIS Submission Set was 0.8861, though it did not reach the top leaderboard positions. Regarding testing and other prediction

targets, the work was focused on predicting the risk-reward performance. Switching to this target lowered the leaderboard score from 0.8861 to 0.7970, indicating high potential. That is why the following steps focussed on the regression problem of the "Perform" column.

The utilization of TPOT and FLAML seemed promising. Using FLAML with a cost-efficient approach should lower the time-to-value for best-in-class results. Nevertheless, both packages could not outperform the XGBoost classification or Random Forest regression approaches. The TPOT approach scored a best score of 0.8336 while running much longer than the FLAML package. This behaviour is reasonable, as FLAML has a specific time budget for gathering results. In a maximum of 30 minutes of training the FLAML package achieved a score of 0.8119 on the submission dataset.

In the last step and after three of four weeks in the competition, the goals was to tweak the last performance out of the models. Therefore, hyperparameter tuning, like in subsection II-D presented, was used. In addition to the scikit-learn models, FLAML was integrated as another machine-learning model. Overall, the best-performing model within this step was a random forest with a score of 0.7921 on the leaderboard. Other models like XGBoost or even the FLAML approach were behind this pipeline.

Like already outlined, we explored the use of stacking to combine the strengths of multiple predictors. Unfortunately, this idea was implement near to the deadline of the competition and is therefore very rudimentary. Specifically, we employed the default scikit-learn StackingRegressor with the default underlying RidgeCV regression algorithm, feeding in the predictions from our best-performing LightGBM and Random Forest models from the hyperparameter tuning before, selected based on their optimal preprocessing configurations. This approach allowed us to capture more nuanced patterns and relationships in the data, leveraging the diverse strengths of each model. The resulting stacking model yielded a score of 0.7921, demonstrating the potential benefits of combining multiple predictors. Stacking has shown promise in this project, and it could be effectively used with an even broader range of models and meta-learners in future projects to enhance our predictive capabilities further.

Overall, this left three best models: a randomly experienced XGBoost Classifier, a hyperparameter-tuned pipeline based on Random Forest and a rudimentary stacking approach. Both more sophisticated approaches were experienced as outperformed slightly by the XGBoost Classifier. As the FedCSIS challenge allowed three models for the final evaluation, those three were handed in randomly to catch the best performance experienced and the models with the highest expected generalization. The final score was 0.8076, which is very close to the results experienced before. Unfortunately, the FedCSIS challenge platform is publishing which model of the three was the best performing. It is worth mentioning that good generalization was vital in getting a top 5 rank in the challenge. The best score of 0.7822 was only 42nd on the public leaderboard, with the best score being 0.5921.

## IV. Conclusion

Through extensive experimentation with various models and preprocessing techniques, we explored various techniques to tweak the performance to the top 5 on a final leaderboard. Different preprocessing strategies, outlier detection methods, and hyperparameter tuning influenced each model's performance. Although improvements were made, some methods like group-specific preprocessing for SVR and feature selection for XGBoost did not enhance performance. In addition, the evolution steps produced better reproducible results, but they never passed the XGBoost classifier. Overall, our best results were achieved using a combination of careful preprocessing and model-specific adjustments.

## References

[1] Christo El Morr, Manar Jammal, Hossam Ali-Hassan, and Walid EI-Hallak. *Machine Learning for Practical Decision Making: A Multidisciplinary Perspective with Applications from Healthcare, Engineering and Business Analytics*. Springer International Publishing, 2022. ISBN 9783031169908. doi: 10.1007/978-3-031-16990-8. URL http://dx.doi.org/10.1007/978-3-031-16990-8.

[2] Frederic Lardinois, Matthew Lynley, and John Mannes. Google is acquiring data science community kaggle. URL https://techcrunch.com/2017/03/07/google-is-acquiring-data-science-community-kaggle/?guccounter=1&guce_referrer=aHR0cHM6Ly9kZS53aWtpcGVkaWEub3JnLw&guce_referrer_sig=AQAAAEu9gSzQHtMGz1fxcvTfrr5VGV41GmfxVdjjnmodYOzlHNhlxLXWNY7by5UshvhMOqu7rfB4Qcx05Z5fi8vMGelVAxyorBLu--6UN1lxAG_nNgSdNy1MNv9L3m92Fxlz8kIr5YF1Kjv9z2ErFaqh3qeHzl_2_QiWylNrJMEJsK4L.

[3] Pau Rodríguez, Miguel A. Bautista, Jordi Gonzàlez, and Sergio Escalera. Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing*, 75:21–31, July 2018. ISSN 0262-8856. doi: 10.1016/j.imavis.2018.04.004. URL http://dx.doi.org/10.1016/j.imavis.2018.04.004.

[4] Andre Ye and Zian Wang. *Modern Deep Learning for Tabular Data: Novel Approaches to Common Modeling Problems*. Apress, 2023. ISBN 9781484286920. doi: 10.1007/978-1-4842-8692-0. URL http://dx.doi.org/10.1007/978-1-4842-8692-0.