Proceedings of the 19th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 739–744
DOI: 10.15439/2024F4556
ISSN 2300-5963 ACSIS, Vol. 39

Decoding Financial Data: Machine Learning Approach to Predict Trading Actions

Yat Chun Fung, Bekzod Amonov
TU Dortmund University
Department of Statistics, Data Science
Dortmund, Germany
yatchun.fung@tu-dortmund.de, bekzod.amonov@tu-dortmund.de

***Abstract*—This paper presents a study on predicting stock trends using a dataset consisting of key financial indicators from 300 S&P 500 companies over a decade. Each company is characterized by 58 financial indicators along with their 1-year changes, offering valuable insights into potential trends. The objective is to develop predictive models to accurately forecast trading actions (buy, sell, hold) based on fundamental financial data. Three machine learning models—Random Forest, CatBoost, and XGBoost classifiers—were trained, employing two distinct voting mechanisms. The first voting mechanism was utilized in the competition, while the second was developed post-competition after the test labels were released. Notably, the second model was trained solely on the training data. The results demonstrate that both voting mechanisms effectively capture trends, as reflected by the average error cost measure, evaluated using the provided error cost matrix.**

I. Introduction

PREDICTING stock trends has long been a crucial aspect of financial analysis, enabling investors and traders to make informed decisions about buying, selling, or holding stocks. With the advent of advanced machine learning techniques, the ability to forecast stock movements based on fundamental financial data has significantly improved.

This research aims to enhance the accuracy of stock trend predictions by developing and training various machine learning models, including Random Forest, CatBoost, and XGBoost classifiers. Ultimately, the goal is to make informed buy, sell, and hold decisions based on financial indicators using the machine learning models developed in this research.

The composition of this study is as follows: Section 2 provides a brief description of the provided training data set. Section 3 introduces the methodologies, including data preprocessing techniques, models, and prediction techniques used in this study. Section 4 provides a descriptive analysis of the training dataset, evaluates the model designs (two voting mechanisms) and their outcomes, and examines the quality of the predictions. Finally, Section 5 offers conclusions and discusses potential directions for future research.

II. Data set description

The training dataset consists of 8,000 instances from 300 companies, each described by 58 financial indicators and their 1-year changes. These companies are categorized into 11 sectors, as indicated by the *Group* column, which is the only categorical feature in the dataset; all other features are numerical. The dataset includes two target variables: *Perform* and *Class*, with the primary objective being to predict *Class*. The *Perform* variable is numerical and reflects the company's stock market performance, while the *Class* variable is categorical, taking values of -1, 0, or 1, corresponding to sell, hold, or buy decisions, respectively.

There are two types of missing values in the dataset: "NA" and empty strings. "NA" indicates missing information, while empty strings represent non-applicable values. A total of 2,806 rows contain missing values.

III. Methodology

A. Data Preprocessing

1) Categorical Data Handling: For the only categorical feature, *Group*, one-hot encoding was applied to convert it into a numerical format. This process involved creating binary columns for each unique category within the *Group* feature, allowing the model to interpret categorical data as distinct numerical values without imposing any ordinal relationship. This encoding ensures that the categorical data is appropriately represented for analysis and model training [1].

2) Data Imputation: As mentioned above, there are two types of missing values ("NA" and empty strings) in the data set. For the former missing values, the MICE (Multiple Imputation by Chained Equations) method was used for imputing missing data. The process begins by initializing the missing values with a placeholder (such as the mean). Then, in an iterative manner, each variable with missing data is predicted based on the other variables in the dataset. This prediction is updated in each iteration, progressively refining the imputed values until the process converges to stable estimates. This iterative refinement helps ensure that the imputations are consistent with the underlying data structure [2].

However, instead of using all features as predictors, the top three most correlated features were selected for imputing the missing values after testing. This selective approach improves the quality of imputation by reducing the influence of less relevant variables, which ridge regression alone may not fully mitigate. By focusing on the most relevant relationships within the data, this method helps produce a more robust dataset for subsequent analysis.

For the other type of missing values, represented by empty strings, a placeholder with extremely large value was used. These values were not imputed using MICE because they are not applicable to the specific rows in which they appear. By using a placeholder, we can isolate their influence on the models' performance, which is particularly effective for tree-based methods where such distinct values can be handled appropriately without distorting the analysis.

B. Model selection

1) Random Forest: Random Forest is an ensemble learning method, specifically a type of bagging, that aims to improve model stability and accuracy by aggregating multiple strong learners—in this case, decision trees. Each decision tree in a Random Forest is constructed using bootstrapped samples of the data, with random subsets of features considered at each node split. This approach enhances model diversity and helps prevent overfitting through majority voting [3].

The decision trees in Random Forests determine splits based on reducing Gini impurity, which allows the model to perform inherent feature selection. This not only improves the model's predictive power but also provides valuable insights into the relative importance of each feature in the dataset. Such insights are particularly useful in high-dimensional datasets like the one at hand, which contains 117 variables, where domain knowledge alone may not clearly indicate the most important features [4].

2) XGBoost: XGBoost (eXtreme Gradient Boosting) is another ensemble method employed in this research, specifically a boosting technique. It offers several features that make it particularly well-suited for this dataset. Firstly, XGBoost includes built-in regularization (both L1 and L2), which helps control model complexity and prevent overfitting—issues that are common with our models in this dataset. Secondly, its ability to handle missing data and its sparsity awareness are particularly advantageous, given that approximately 35% of the rows in this dataset contain missing values. This capability is crucial in managing complex datasets with a large number of features. Additionally, XGBoost's scalability and efficiency, enabled by innovations such as a sparsity-aware tree learning algorithm, parallel and distributed computing, and out-of-core computation, make it highly effective and time-efficient for hyperparameter tuning [5].

3) CatBoost: CatBoost (Categorical Boosting) is another ensemble method utilized in this research, specifically designed to excel in handling categorical variables within a boosting framework. It offers several features that make it particularly well-suited for this dataset. Firstly, CatBoost implements an ordered boosting technique, which mitigates the prediction shift problem commonly encountered in traditional gradient boosting methods. This leads to more accurate and stable models, particularly important given the complexity of our dataset. Secondly, CatBoost natively handles categorical data without requiring extensive preprocessing, making it an ideal choice for datasets like ours that include categorical variable *Group* [1].

4) MLP Classifier: The MLP (Multi-Layer Perceptron) is a type of feedforward artificial neural network composed of multiple layers of interconnected nodes, where each node is fully connected to every node in the subsequent layer [6]. In this research, the MLP is employed for the soft voting mechanism, which is used in the second model architecture.

C. Model Voting

1) Hard Voting: Hard voting is an ensemble technique where multiple models vote on predicted class labels, with the majority rule determining the final prediction. Each model contributes one vote, and the label with the most votes is selected. This method helps reduce overfitting and improve generalization by leveraging the strengths of different models [7]. This technique was employed in the first model architecture used during the competition.

2) Soft Voting: Soft voting is an ensemble technique where the predicted probabilities from multiple models are averaged to make a final prediction. Unlike hard voting, which considers only class labels, soft voting factors in each model's confidence, often leading to more accurate decisions [7]. This method reduces variance and bias, resulting in more robust performance across datasets. In this research, the MLP's weights are used for soft voting, which is employed in the second model architecture.

IV. Data Analysis

A. Descriptive Data Analysis

1) Data Distribution: The dataset exhibits an imbalanced *Class* distribution, with 47% of observations classified as *Class* 1 (Buy), 39% as *Class* -1 (Sell), and only 14% as *Class* 0 (Hold). The distribution of *Perform* appears symmetric and follows a bell-shaped curve, resembling a normal distribution, with a mean of 0.0341 and a variance of 0.0215 (both rounded to three sig. fig.).

According to Figure 1, observations can be categorized into three classes based on their performance values: high values (> 0.04) align with *Class* 1 (Buy), moderate values (between -0.015 and 0.04) align with *Class* 0 (Hold), and low values (< -0.015) align with *Class* -1 (Sell). Additionally, these thresholds suggest the potential for a regression task, where predicted values from regression models can be used to determine *Class* labels with the above thresholds.

In regard to the features, the distribution of the *Group* variable is notably imbalanced. For the distribution of the numerical features, please refer to the table in the appendix. It can be observed that some of the distributions are right-skewed, indicating the presence of outliers.

2) Variable Relationships: The relationship between *Perform* and the categorical variable *Group* was examined using boxplots. The values of $Perform$ generally display similar distributions, with comparable central tendencies (median) and variability (IQR) across different groups. Notable exceptions include G3, which exhibits a negative median, and G8, which has large outliers. Additionally, G4 and G5 have outliers on

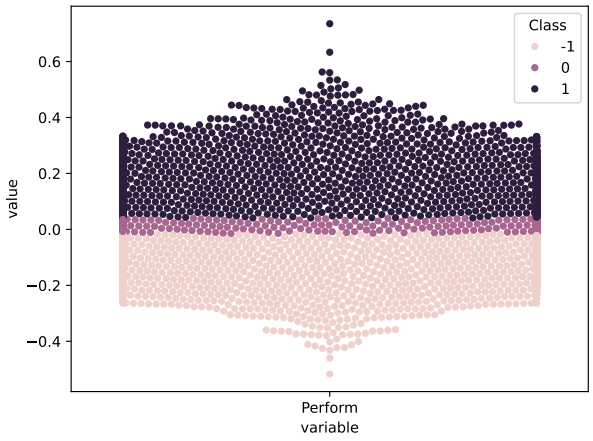

Fig. 1. Swarm plot showing the distribution of the *Perform* variable across different classes. The plot illustrates how the *Perform* values are distributed within each class. It highlights the separation of classes based on the *Perform* variable.

both ends—large and small—while G1 has one extremely high outlier.

The relationship between the target variable *Perform* and the numerical independent variables was analyzed by calculating the correlation coefficients. The highest absolute correlation was observed with variable I9 (Cash Flow from Operations to Total Assets), which had a value of 0.0762. These low correlation values indicate that there is essentially no significant linear relationship between the target variable *Perform* and the independent variables.

B. Modeling

1) Data Preprocessing: The same preprocessing steps were applied throughout the pipeline. The categorical variable was one-hot encoded into 11 binary columns, except in the case of CatBoost, which natively handles categorical variables. For numerical features, missing values (NAs) were first imputed using MICE, followed by standardization. Lastly, placeholders were added for other types of missing values (e.g., empty strings), with this step performed last to prevent any impact on the standardization process.

2) Custom Prediction Rule: A custom prediction rule was applied to derive the labels from the probability vector after voting. If the probabilities for both -1 and 1 are below 0.5, the model predicts 0 to minimize expected loss; otherwise, it selects -1 or 1 based on the higher probability. The derivation of this rule is illustrated in the equation below.

$$\text{Expected Error} = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{pmatrix} = \begin{pmatrix} \pi_2 + 2\pi_3 \\ \pi_1 + \pi_3 \\ 2\pi_1 + \pi_2 \end{pmatrix}$$

$$\text{argmin}_{-1,0,1}\text{Expected Error} \quad \Rightarrow \begin{cases} -1 & \text{if } \pi_1 \geq 0.5 \\ 0 & \text{otherwise} \\ 1 & \text{if } \pi_3 \geq 0.5 \end{cases}$$

where π_i stands for the probability for Class i, $i = \{-1, 0, 1\}$.

3) Model Architecture: For the first model, an ensemble learning approach was implemented by fitting three classification models: Random Forest, XGBoost, and CatBoost. First, the predicted labels were adjusted using custom prediction rules, followed by a voting process. The voting rules are as follows: If the predictions of all models coincide, any of the predictions may be chosen. In cases where all predictions differ, the prediction from the Random Forest classifier, which demonstrated the best preliminary results (only after the custom prediction rules), is selected. When the prediction from the strongest model (Random Forest) coincides with that of any weaker model (XGBoost and CatBoost), that prediction is chosen. If the predictions of the weaker models coincide but differ from the strongest model, the consensus of the weaker models is selected.

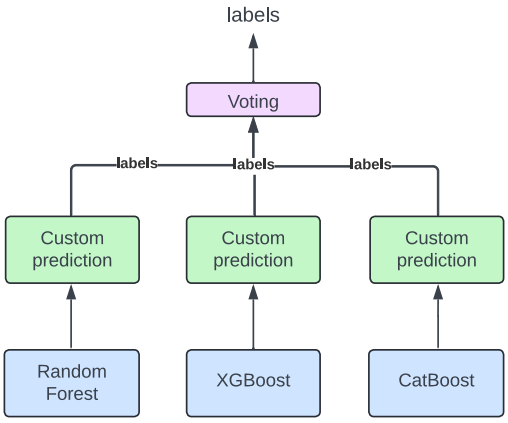

Fig. 2. Voting mechanism where predictions are adjusted based on custom rules. The Random Forest prediction is favored if all predictions differ, while the consensus of weaker models is chosen when they align but differ from the strongest model

For the second model, an ensemble learning approach by fitting three classification models was implemented: Random Forest, XGBoost, and CatBoost. Each model produces a probability vector, representing the predicted probabilities for each class. These three probability vectors are then used as inputs to a MLP classifier. The MLP has a single hidden layer consisting of 20 neurons, with the output layer corresponding to the true class labels.

After training the MLP, we extract the weight matrix associated with the connections between the input layer (9 nodes corresponding to the probabilities from the three models) and the hidden layer (20 neurons). This weight matrix is of size 9x20. To derive feature importance, we sum the weights row-wise across the matrix, resulting in a 9x1 column vector.

This column vector is then used as a set of weights to perform weighted soft voting, where each probability vector from the three models is multiplied by its corresponding weight. The

final predicted class label is determined by applying custom prediction rules to the aggregated weighted probabilities.

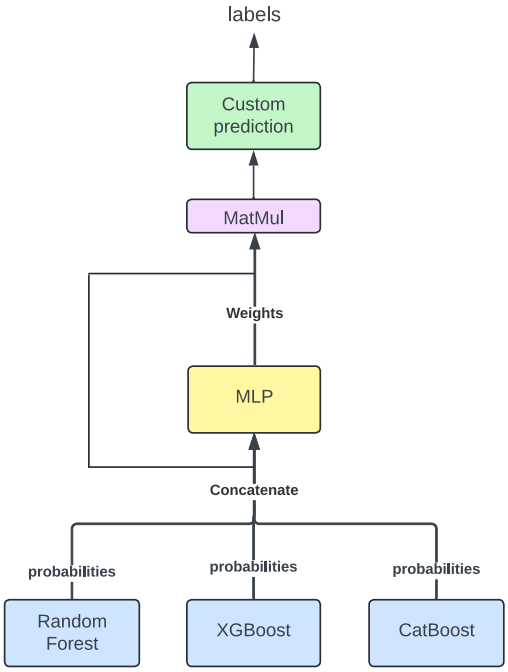

Fig. 3. The model architecture combines probability vectors from Random Forest, XGBoost, and CatBoost models, which are input into an MLP with one hidden layer. The extracted weight matrix is used for weighted soft voting to determine the final class label.

C. Results

1) Model Evaluation and Error Diagnosis: Among the three models (Random Forest, XGBoost, and CatBoost), XGBoost demonstrates the best overall performance, achieving the highest accuracy , weighted precision, weighted recall, and F1-score. It also has the lowest error (both preliminary and final) according to the cost matrix (0.8218 and 0.8355). Specifically for Class -1, XGBoost has the second-highest precision but the highest recall, indicating its strong ability to identify Class -1 observations. However, the lower precision compared to CatBoost suggests a potential issue with overfitting. Despite this, XGBoost achieves the highest F1-score for Class -1, balancing its recall and precision effectively.

For Class 0, XGBoost achieves the highest recall and precision among the models. This can be attributed to the fact that it is the only model that makes Class 0 predictions (thanks to the boosting algorithm probably), producing 8 predictions in total. However, only 2 of these predictions are correct, resulting in a precision of 0.25. The F1-score of 0.01 reflects XGBoost's challenges in both identifying Class 0 data and accurately predicting it. Despite these difficulties, XGBoost still has the highest F1-score for Class 0 compared to the other models. The poor performance for Class 0 can be explained by the limited number of observations (1136 out of 8000, or 14.2%), which provides insufficient data for the model to effectively learn the patterns in the predictors. For Class 1, XGBoost has the lowest recall but the highest precision among the models. Overall, it ties with Random Forest for the second-highest F1-score. Compared to Class 0 and Class -1, the recall, precision, and F1-score for Class 1 are significantly higher across all three models. This can be attributed to the abundance of Class 1 observations in the training data (3768 out of 8000, or 47.1%), which provides ample instances for the models to effectively learn the patterns associated with this class.

For CatBoost, all performance metrics except the weighted F1-score rank second best. CatBoost excels in identifying and predicting Class 1, as indicated by its highest F1-score among the three models as shown in Table I. However, it struggles with Class 0, as evidenced by a 0 F1-score, despite assigning more importance to Class 0 during training. This may be due to the majority classes having stronger signals, making their patterns easier to learn. As CatBoost's boosting algorithm focuses on minimizing overall error, it might prioritize the majority classes, leading to fewer correct predictions for the minority class. Additionally, CatBoost performs the worst for Class -1, as shown by its lowest F1-score for that class. Its final result is 0.8465, which is the second best among the three models.

For Random Forest, the F1 scores for Classes -1, 0, and 1 all rank second, indicating a balanced performance across the classes. Despite this apparent balance, Random Forest has the lowest accuracy, weighted precision and recall, with a final result of 0.8575, the worst among the models. A notable issue is the model's failure to predict any instances of Class 0, which makes up 14% of the dataset. This problem likely arises due to the nature of the data, which may lack strong, consistent signals in the predictors. As a result, the decision trees within the Random Forest may produce very different results, causing the majority voting process to fail.

The bagging algorithm used in Random Forest could further exacerbate this issue. By training on different subsets of data, the model may inadvertently reduce the representation of Class 0 even more, leading to its omission in the final predictions. Additionally, if the predictors for Class 0 are weak or overlap significantly with other classes, the trees may not learn to split on features that identify Class 0 effectively.

Our first final model achieved a score of 0.8059, placing 4th on the leaderboard. Compared to the three individual component models, the final model has lower accuracy, weighted recall, and F1-score. However, it demonstrates a higher weighted precision, which contributes to the improved score. This precision boost is also influenced by our custom prediction rules, where misclassifying Class 0 results in a lower error (only 1), making it less costly.

The custom prediction rule dictates that the model will predict Class 0 unless the confidence (probability) for either Class -1 or Class 1 is high enough (>= 0.5). This approach

significantly increased the number of Class 0 predictions while reducing the predictions for Classes -1 and 1 (148 predictions for Class -1, 1204 for Class 0, and 648 for Class 1).

For Class 0, the precision is lower than that of XGBoost. This is because the custom prediction rule leads to many Class 0 predictions even when the model is not confident. Consequently, many of these predictions are made not because the model is certain about Class 0, but because the probabilities for Classes -1 and 1 are not sufficiently high. This uncertainty reduces the precision for Class 0. On the other hand, the recall for Class 0 increases significantly due to the higher number of Class 0 predictions.

For Class 1, the first final model achieves better precision than XGBoost (which had the highest precision for Class 1 among the individual models). This improvement is because the number of Class 1 predictions decreases, but the correct predictions for Class 1 do not decrease as much, thanks to hard voting. This results in a higher precision for Class 1.

Regarding Class -1, the precision remains the same as CatBoost (the model with the highest precision for Class -1). This is because the decrease in correct predictions for Class -1 is more pronounced than the overall decrease in the number of Class -1 predictions. This outcome can be attributed to the limited number of Class -1 instances in the training data. Even when the model correctly identifies Class -1, the confidence often falls below 0.5, leading to these predictions being overridden by the custom rule. As a result, the precision for Class -1 does not improve.

For the second final model, which was developed after the competition, the final result was 0.797. The model exhibits higher or equal precision across all three classes compared to the first final model, thanks to the soft voting approach that takes into account the probabilities from each component model when making decisions. Specifically, the model made 168 predictions for Class -1, 1,415 for Class 0, and 417 for Class 1.

Compared to the first model, the second model produces even more Class 0 predictions, likely to minimize the error when the model is uncertain. It also makes more correct predictions for Class -1, resulting in increased precision and recall for that class. However, Model 2 makes fewer predictions for Class 1. Most of the predictions that were reduced were originally misclassified, leading to an increase in precision but a decrease in recall for Class 1. The overall improvement in the score is likely due to the increased precision for Classes -1 and 1, as well as the increased number of Class 0 predictions, reflecting the model's cautious approach when uncertain.

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT MODELS ACROSS VARIOUS (WEIGHTED) METRICS

	Acc.	Precision	Recall	F1-Score	Prelim.	Final
Baseline	0.1425	0.0203	0.1425	0.0355	0.8564	0.8575
RF	0.5000	0.4244	0.5000	0.4458	0.8465	0.8575
XGBoost	0.5100	0.4713	0.5100	0.4659	0.8218	0.8355
CatBoost	0.5055	0.4381	0.5055	0.4317	0.8564	0.8465
Final Model 1	0.3155	0.4948	0.3155	0.3211	0.6980	0.8059
Final Model 2	0.2810	0.5345	0.2810	0.2923	0.7673	0.7970

Remark: Acc. denotes accuracy, Prelim. refers to the preliminary score on the leaderboard, and Final represents the final score on the leaderboard

TABLE II
COMPARATIVE F1-SCORES BY CLASS FOR DIFFERENT MODELS

	Baseline	RF	XGBoost	CatBoost	Final Model 1	Final Model 2
-1	0.00	0.41	0.46	0.34	0.17	0.21
0	0.25	0.00	0.01	0.00	0.24	0.25
1	0.00	0.61	0.61	0.63	0.47	0.38

TABLE III
COMPRATIVE PRECISION SCORES BY CLASS FOR DIFFERENT MODELS

	Baseline	RF	XGBoost	CatBoost	Final Model 1	Final Model 2
-1	0.00	0.48	0.49	0.52	0.52	0.58
0	0.14	0.00	0.25	0.00	0.15	0.15
1	0.00	0.51	0.52	0.50	0.58	0.61

TABLE IV
COMPARATIVE RECALL SCORES BY CLASS FOR DIFFERENT MODELS

	Baseline	RF	XGBoost	CatBoost	Final Model 1	Final Model 2
-1	0.00	0.35	0.44	0.26	0.10	0.13
0	1.00	0.00	0.01	0.00	0.63	0.73
1	0.00	0.77	0.72	0.86	0.40	0.27

V. CONCLUSION

This study investigated stock trend prediction using key financial indicators from 300 S&P 500 companies. Three machine learning models—Random Forest, CatBoost, and XGBoost—were employed with two distinct voting mechanisms. While XGBoost delivered the best overall performance, our custom Model 1 and Model 2 achieved better final results by effectively managing prediction uncertainties. Model 2, developed post-competition, demonstrated further improvements in precision across all classes, underscoring the effectiveness of soft voting. Despite the challenges posed by class imbalance, particularly for Class 0 and -1, our approach successfully captured significant trends, as reflected in the final scores. Looking forward, a promising direction would be to explore stacking as an ensemble method, using the probability vectors from the three models as input to an MLP classifier to produce a refined probability vector, followed by custom prediction rules to determine the final labels.

REFERENCES

[1] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," *Advances in Neural Information Processing Systems*, vol. 31, pp. 6638–6648, 2018.

[2] S. Van Buuren and K. Groothuis-Oudshoorn, "Mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, vol. 45, no. 3, 2011.

[3] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[4] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[5] 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*. ACM, 2016, pp. 785–794.

[6] K. Gurney, *An introduction to neural networks*. CRC press, 1997.

[7] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, 2004.