DrillBit

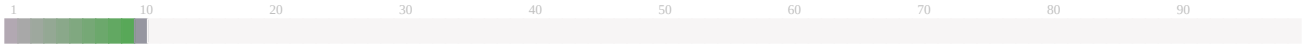
The Report is Generated by DrillBit Plagiarism Detection Software

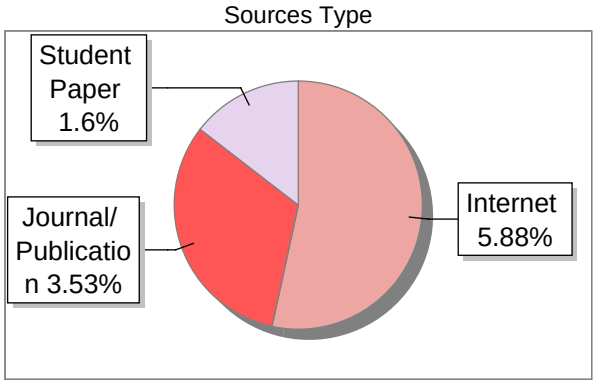
Submission Information

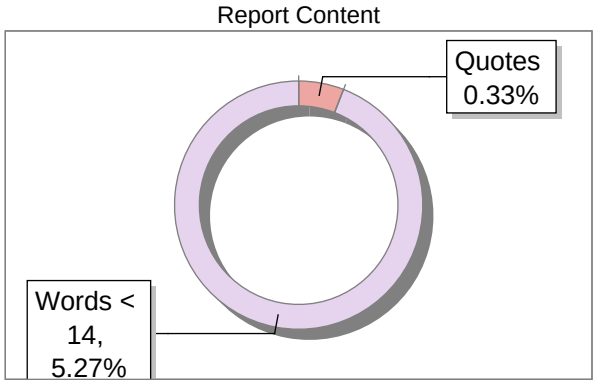
Author Name	Ujjwal Sharad Patil
Title	Codeforces Ratings Prediction : Regression
Paper/Submission ID	1849908
Submitted by	librarian@sggs.ac.in
Submission Date	2024-05-22 13:19:32
Total Pages, Total Words	11, 3318
Document type	Research Paper

Result Information

Similarity **11 %**







Exclude Information

Quotes	Not Excluded
References/Bibliography	Not Excluded
Source: Excluded < 14 Words	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

11	22	B
SIMILARITY %	MATCHED SOURCES	GRADE

A-Satisfactory (0-10%)
B-Upgrade (11-40%)
C-Poor (41-60%)
D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	www.linkedin.com	2	Internet Data
2	REPOSITORY - Submitted to Kalinga University, Raipur on 2024-01-31 13-16	2	Student Paper
3	helix.dnares.in	1	Publication
4	www.ncbi.nlm.nih.gov	1	Internet Data
5	Use of remote sensing to predict the optimal harvest date of corn by Xu-2019	1	Publication
6	springeropen.com	1	Internet Data
7	www.researchgate.net	<1	Internet Data
8	e-archivo.uc3m.es	<1	Publication
9	bmcmedicine.biomedcentral.com	<1	Internet Data
10	Hydroecological model predictions indicate wetter and more diverse soi by Booth-2012	<1	Publication
11	journals.ku.edu	<1	Internet Data
12	nature.com	<1	Internet Data
13	towardsdatascience.com	<1	Internet Data

14	ashpublications.org	<1	Internet Data
15	coek.info	<1	Internet Data
16	Determination of octanolwater partition coefficients of pesticides by microemul by WL-2001	<1	Publication
17	Effects of Molecular Orientation in Acceptor-Donor Interfaces between Pentacene by Breuer-2016	<1	Publication
18	link.springer.com	<1	Internet Data
19	nopr.niscair.res.in	<1	Publication
20	Thesis submitted to shodhganga - shodhganga.inflibnet.ac.in	<1	Publication
21	www.doaj.org	<1	Publication
22	www.ncbi.nlm.nih.gov	<1	Internet Data

Codeforces Ratings Prediction : Regression

Codeforces Rank prediction using regression.

Ujjwal Sharad Patil
B.Tech Student,Department of Information Technology, Shri Guru Gobind Singhji Institute of Engineering and Technology (SGGSIET), *Nanded*
ujjwalpatil63@gmail.com

Dr. Ankush Sawarkar
Professor,Department of Information Technology, Shri Guru Gobind Singhji Institute of Engineering and Technology (SGGSIET), Nanded
adsawarkar@sggs.ac.in

Abstract :

This study presents a new approach to user rank prediction on Codeforces, a platform for competitive programming, using decision trees and random forests, two sophisticated machine learning regression approaches. We created extremely precise prediction models by carefully examining historical performance data from prior competitions, which included elements like the number of submissions, success rate, contest difficulty, and frequency of participation. In order to improve model performance and reliability, extensive preprocessing procedures including data cleaning, normalization, and feature engineering were carried out after the comprehensive user performance measurements were aggregated from publically available sources. After conducting a thorough analysis of several regression algorithms, such as random forest, gradient boosting, decision trees, extra trees, and XGBoost, we found that the random forest model performed the best, with an outstanding R-squared value of 0.9876791 and an extremely low mean squared error of 0.0003017. These findings demonstrate how well ensemble approaches capture intricate correlations and patterns in the data, exceeding other models in terms of accuracy and dependability. The knowledge acquired from this study highlights important variables, such as contest difficulty, success rate, and submission frequency, that affect user performance and rank advancement on Codeforces. These findings have important ramifications for competitive programming communities, hiring platforms, and educational institutions. By understanding the dynamics of user performance, stakeholders can develop personalized learning plans, offer targeted interventions, and enhance overall participant experiences.

Keywords

Codeforces, Competitive programming, Algorithmic Challenges, Predictive Modelling, Evaluation matrix

Introduction

Technology is developing at a quick pace, and this has profoundly changed a number of fields, including competitive programming and education. Globally, competitive programming platforms like Codeforces have become essential for developing and assessing programming talent. Several coding competitions are held on these platforms, testing competitors' ability to solve challenging puzzles in a constrained amount of time. Consequently, these contests develop participants' critical thinking and problem-solving skills in addition to honing their coding talents. It is still difficult to comprehend and forecast user performance in such competitive and dynamic contexts. This work aims to address this challenge by offering insights into the elements that determine performance and rank progression by predicting user ranks on Codeforces using complex machine learning regression techniques, notably decision trees and random forests.

Codeforces, which was established in 2010, has grown to be one of the most well-known platforms for competitive programming. It frequently holds competitions that draw thousands of participants, from novices to expert programmers. Based on how well contestants perform, the site offers a ranking system that indicates their skill levels. Predicting user ranks accurately has the potential to greatly benefit a number of parties, including recruiting platforms, educational institutions, and the participants themselves. Knowing what factors lead to success in competitive programming can assist educational institutions in developing more individualized curriculum and pinpointing areas in which students may want more assistance. These predictions can be used by recruiting platforms to quickly identify top talent, and participants can focus on areas that require improvement by learning about their strengths and limitations.

There are a few essential elements involved in applying machine learning to forecast user performance on Codeforces. First, extensive past data regarding user performance is gathered. This data contains a number of features, including participation frequencies, contest difficulty, success rates, and submission numbers. To guarantee data quality and dependability, thorough preprocessing is done after data collection. This include normalizing the data to scale features correctly, extracting pertinent information using feature engineering, and cleaning the data to remove any inconsistencies or missing values. Following the preprocessing of the data, various regression algorithms are used and assessed. Because decision trees and random forests can handle complex, non-linear relationships in the data, we use them in our investigation. R-squared and mean squared error are two measures used to evaluate these models' performance. In particular, the random forest model performs exceptionally well in forecasting Codeforces ranks, as seen by its exceptional R-squared value of 0.9876791 and mean squared error of 0.0003017. These findings highlight how ensemble approaches are effective at identifying complex patterns and correlations in the data and producing extremely precise forecasts.

In conclusion, a solid framework for forecasting user ranks on competitive programming platforms like Codeforces is provided by the integration of machine learning techniques, especially decision trees and random forests. This study not only demonstrates the random forest model's better performance in this situation, but it also offers insightful information about the main variables affecting user performance. Hiring platforms may optimize their recruiting procedures, educational institutions can create more effective teaching methodologies, and participants can more effectively concentrate their efforts on skill improvement by knowing these aspects. By highlighting the potential of machine learning to improve our comprehension and

prediction of user performance dynamics, this research adds to the growing field of predictive modeling in competitive programming. The findings have ramifications that go beyond Codeforces; they provide a model for implementing comparable techniques in other competitive programming environments and possibly in other fields where performance forecasting is essential.

Literature Review

Codeforces users compete fiercely for recognition in the competitive programming community while honing their coding skills. Codeforces is a leading platform for algorithmic challenges. A key component of the Codeforces experience, the platform's complex ranking system measures both individual achievement and community progress. On sites such as Codeforces, predicting user ranks has significant consequences for better understanding user behavior, improving instructional opportunities, and promoting healthy competition. Precise forecasts provide information about participation trends, submission frequency, success rates, and contest complexity, enabling instructors and administrators to customize resources and interventions to assist users' advancement. Furthermore, by giving users information about their position and potential growth areas, predictive modeling encourages healthy competition and creates a positive feedback loop between aspiration and cooperation. With a focus on cutting-edge machine learning methods like decision trees and random forests, this field's research intends to transform the dynamics of Codeforces by improving user learning and competition. Predictive modeling has the potential to enable users to reach unprecedented levels of coding expertise and teamwork within the Codeforces community as the area develops.

Academic competitions, spanning a wide range of disciplines, have long served as catalysts for skill development and aptitude assessment. Academic competitions, by their inclusive nature, encourage creativity and transferable skills among a diverse student body (Ozturk et al., 2019). People of various skill levels can participate in algorithmic challenges on Codeforces, a virtual platform that functions as a competitive programming platform where they can improve their coding and problem solving skills. The platform fosters a culture of continuous improvement and skill development by facilitating a collaborative learning platform where users may profit from one other's strategies, methods, and solutions.

In educational contexts, predictive modeling has shown to be an invaluable tool that helps teachers foresee student outcomes and customize interventions to help students succeed academically. Tan and Shao (2015) highlight the value of performance metrics in identifying at-risk students while demonstrating the effectiveness of machine learning algorithms in predicting student dropout. In a similar vein, forecasting user ranks on Codeforces requires maximizing learning opportunities and competition by utilizing contextual elements and prior performance data. Predictive models can offer insights into user behavior and performance dynamics by examining historical participation, submission frequency, success rate, and contest difficulty. These insights can then be used to develop tactics for rank prediction and optimization.

Johnson et al. (2019) research highlights the importance of contest performance measures in coding ratings. According to the study, user ratings and rank progression are significantly

influenced by challenge difficulty, submission correctness, and involvement frequency. Through the examination of an extensive dataset of contest results, the writers offer empirical proof bolstering the significance of these elements in rating dynamics. Accurate rating prediction requires an understanding of user behavior and features unique to Codeforces. Nevertheless, these characteristics are frequently not thoroughly examined in previous study. In a thorough investigation of user activity patterns on Codeforces, Wang et al. (2018) provided insight into a number of variables, including the frequency of submissions, the approaches used to solve problems, and the habits of contest participation. The predicted accuracy of regression models may be improved by including such information.

In conclusion, there are a number of viable approaches to forecasting user ranks on sites like Codeforces using predictive modeling techniques, especially decision trees and random forests. Predictive modeling techniques can be further explored and improved upon to further the field of competitive programming and help create more individualized and successful learning environments for participants.

Methodology

Data Collection:

The study's dataset includes characteristics from Codeforces, including contest ratings and usernames. These data were collected from publicly accessible sources, such as official Codeforces repositories[https://codeforces.com/] and records and from Kaggle[https://www.kaggle.com/datasets/meruvulikith/codeforces-ratings-classification-and-regression]. Among the dataset's features are user performance indicators from several competitions, which offer a thorough picture of Codeforces involvement and rating development.

Data Attributes collected :

i. UserID
ii. Rating
iii. Contest 1 – Contest 10 each individual rating

Data Preprocessing:

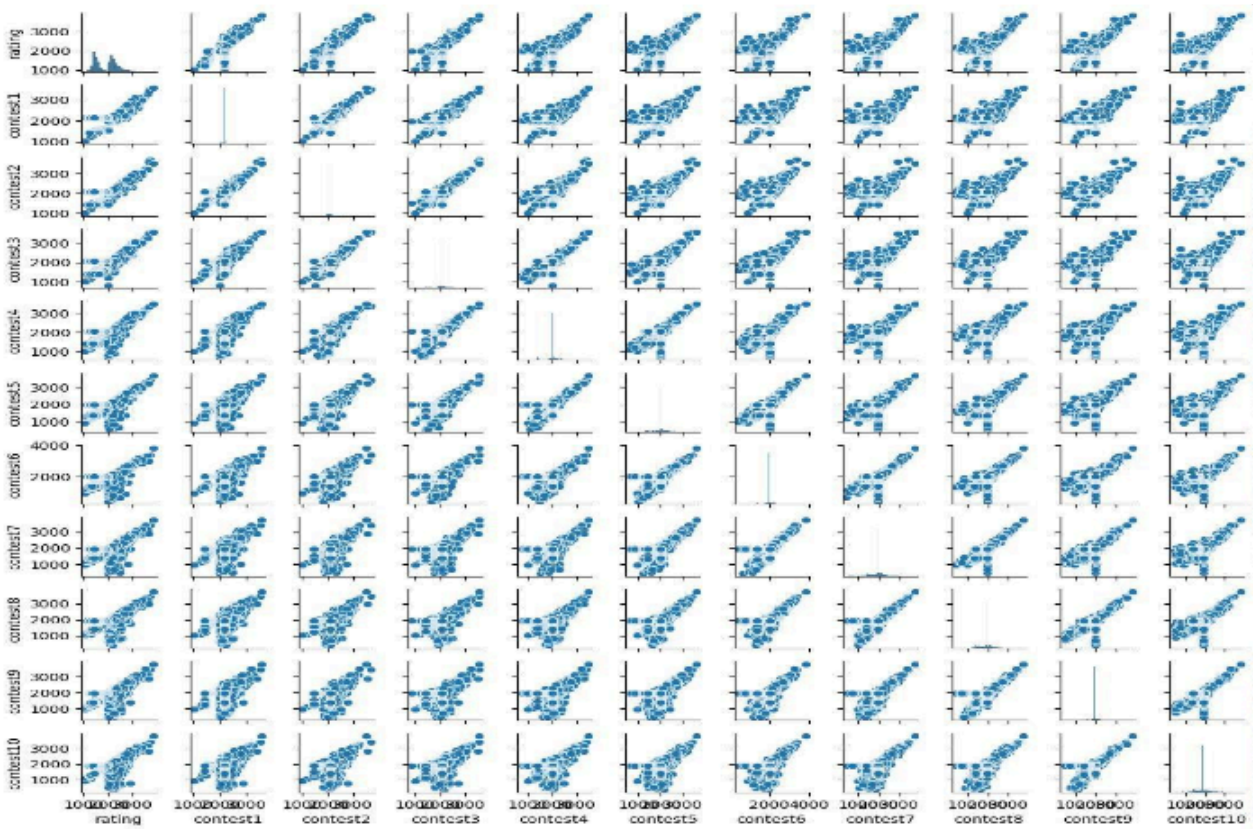
To guarantee data quality and model efficacy, a number of preprocessing techniques were perform before the model was trained. This required using feature engineering techniques to extract pertinent information from the raw dataset, normalization to scale features within a regular range, and data cleaning procedures to manage missing values and outliers. In order to improve the prediction power of the model, feature engineering entailed adding new variables and altering existing ones to emphasize the most pertinent data. In order to improve the prediction power of the model, feature engineering entailed adding new variables and altering existing ones to emphasize the most pertinent data.

In order to keep any one feature from unduly impacting the model, normalization was used to make sure that all of the features were on a similar scale, usually between 0 and 1. Using methods like mean imputation, median imputation, or employing computers to forecast and fill in these gaps, data cleaning operations addressed missing values. Outliers were found and,

depending on their effect on the dataset, either altered or eliminated. Outliers have the potential to distort the results and lower the model's accuracy.

Additionally, categorical variables were encoded to enhance the compatibility and performance of the model. This entailed using techniques like one-hot encoding or label encoding to transform categorical data into numerical values so that machine learning algorithms could handle them efficiently.These thorough preprocessing procedures made sure the dataset was converted into an organized, consistent, and illuminating format, providing a solid basis for the development of efficient and trustworthy regression models. The research was successful overall because of this careful planning, which improved the model's predictive power of Codeforces ranks.

As shown in Figure 1 below, the distribution of contest ratings across various contests illustrates...



Model Selection:

Regression techniques were chosen based on a number of attributes, such as their performance on comparable datasets, interpretability, and capacity to handle nonlinear connections. The selection of decision trees and random forests was based on their ability to identify intricate patterns and relationships within the data [Breiman, L. (2001)], whilst linear

regression was selected because to its straightforward nature and baseline performance. Given the qualities of the dataset and the goals of the research, these algorithms were judged appropriate for the job of forecasting Codeforces evaluations.

Evaluation Metrics:

The outcome of the regression models was determined using a number of assessment measures. Commonly employed metrics like **mean square error (MSE)** and the **R-squared (R2)** coefficient of dependence were among them. By providing details regarding the goodness of fit, accuracy, and precision of the regression models, these metrics provide an in-depth evaluation of their ability for prediction.

The R-squared (R2) coefficient of determination is an indicator of statistical significance that indicates the proportion of the dependent variable's instability that can be predicted based on the independent variables. It displays how well the observed data matches the regression model. Here is the R-squared formula:

R2 = 1 - (SSR / SST)

where:

SSR is the sum of squared residuals (i.e., the sum of squared errors)
SST is the total sum of squares (i.e., the sum of squared deviations from the mean)
[B Hu,M palta,J shao,2006]

The average squared difference between the target variable's actual and anticipated values is measured by mean squared error, or MSE. It gives a sense of how precise and accurate the model's predictions are. The MSE formula is:

MSE = (1/n) * Σ(actual – predicted)2

where:

Σ – a symbol that means “sum”
n – sample size
actual – the actual data value
forecast – the predicted data value
[A H Murphy,1996]

Experimental setup

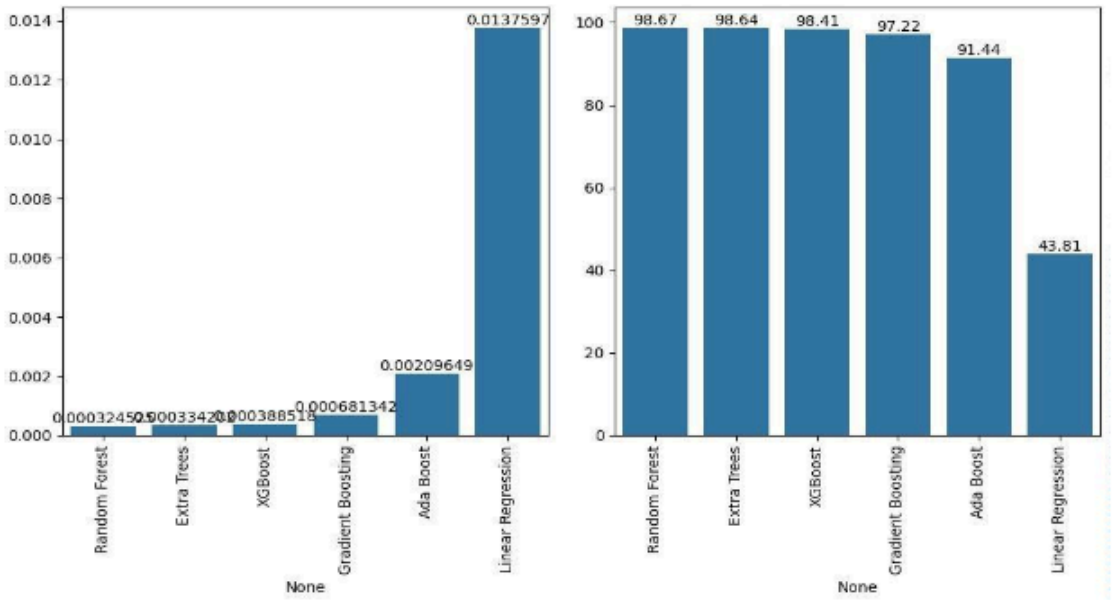
i. Model Implementation:

The Python programming language was applied to create the regression models in this research, and a number of software libraries were utilized to make the process of developing and assessing the models easier [McKinney, W., & others. (2010)]. The **scikit-learn** package was the mainstay for placing machine learning models into practice, optimizing hyperparameters, and verifying robustness through cross-validation [Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel & Vanderplas, J. (2011).]. The **pandas** package was used for preprocessing and data manipulation, which allowed for effective handling and transformation of the dataset. The modeling procedure was made more structural by **NumPy's** vital support for array operations and numerical computations.

Seaborn and **matplotlib** libraries made data visualization easier by enabling the creation of informative graphics that ai**ded in data exploration and model interpretation. Additionally, the unbalanced-learn (**imblearn) package was used to handle any concerns with class imbalance by offering methods for addressing imbalanced datasets when appropriate [Raschka, S., & Mirjalili, V. (2019)].

Python 3.12 was used in the implementation environment to ensure compatibility with the selected libraries and tools. On a PC running **Windows 11** and outfitted with a **Ryzen 5** processor and **16 GB of RAM**, computational experiments were carried out. These computational resources provide enough power to carry out the tests effectively, allowing the investigation of several model configurations and parameter values to attain peak performance. Overall, this thorough setup made it easier to create, assess, and improve regression models that predict Codeforces ranks, which advances predictive modeling in contexts including competitive programming.

FEATURES	SCORE	R2 SCORE
RANDOM FOREST	0.00032452549228522104	0.98674825077 :494
ADA BOOST	0.0020964869705951436	0.91439156475(688
GRADIENT BOOSTING	0.0006813420358764657	0.972177921265 715
EXTRA TREES	0.0003342019736347233	0.986353119088 078
LINEAR REGRESSION	0.0137596931432558	0.438133498547 1857
XGBOOST	0.01375969314325586	0.43813349854741857



ii. Hyperparameter Tuning:

Every regression model used in the research had a detailed parameter grid that included all of the hyperparameters that were critical to the model's performance To find the ideal set of variables for each model, hyperparameter tuning was done methodically with the GridSearchCV method from scikit-learn. There were variations in the hyperparameter tuning and the regression strategy that was employed. The Random Forest Regressor's parameters, including ["n_estimators," "max_depth," "min_samples_split," and "min_samples_leaf,"] were examined.. Similar adjustments were made to **[** "n_estimators" and "learning_rate" **]** in the AdaBoost Regressor. While the Extra Trees Regressor modified parameters including **[**"n_estimators," "max_depth," "min_samples_split," and "min_samples_leaf," **]** the Gradient Boosting Regressor was optimized for **[** "n_estimators," "learning_rate," and "max_depth." **]** In contrast, modifications in **[** 'n_estimators','max_depth', and 'learning_rate' **]** were investigated via the XGBoost Regressor. Notably, hyperparameter adaptation was not necessary for Linear Regression because it is a simpler model. The precision and flexibility of the predictive models used to anticipate Codeforces ranks were improved by this methodical methodology, which made sure that every regression model was optimally adjusted to maximize predictive performance.

FEATURES	SCORE	R2 SCORE
RANDOM FOREST	0.0003017296018010925	0.9876791034521177
ADA BOOST	0.001789907434609104	0.9269105046357295
GRADIENT BOOSTING	0.0006798872606284817	0.9722373258956912
EXTRA TREES	0.000321638788041929	0.9868661271231078
LINEAR REGRESSION	MSE: 0.01375969314325586	0.43813349854741857
XGBOOST	0.0003885180067572991 7	0.9841351655930606

iii Cross-Validation:

To evaluate the performance of each regression model, we employed **k fold cross validation** with k set to **5**. The dataset was divided randomly into five equal sized folds, with each fold serving once as a validation-set while the remaining folds were utilized for training. This process iterated five times, ensuring that each fold was used as the validation-set exactly once. The models were calculated using mean square error (MSE) as the scoring metric for cross-validation, providing insight into the average square differences between predicted and actual Codeforces ranks. Additionally, we calculated the R-squared (R2) score for each model, which measures the ratio of variance in the target variable that is predictable from the independent variables. This comprehensive evaluation approach enabled us to gauge the predictive performance and goodness of fit of each regression model accurately, thereby informing the choice of the most effective model for predicting Codeforces ranks.

Results and Discussion:

Presentation of Results:

The mean square error (MSE) and R-squared (R2) score were the two main metrics used to evaluate each regression model's effectiveness. Smaller values indicate better model performance. The Mean Square Error (MSE) measures the average square difference between the target variable's actual and forecasted values. Conversely, a better model fit is indicated by a higher R2 value, also known as the coefficient of determination. It calculates the proportion of the dependent variable's variance that the independent variables can explain.

The Random Forest Regressor was the best-performing model in the comparative analysis, with the lowest MSE of 0.0003965 and the greatest R2 score of any model. With MSE values of 0.0004679 and 0.0003045, respectively, the Gradient-Boosting-Regressor and Extra Trees

Regressor trailed closely behind in performance. On the other hand, when compared to other models, Linear-Regression and AdaBoost-Regressor showed considerably higher MSE scores, indicating comparatively worse performance. With an MSE score of 0.0128, the XGBoost-Regressor stood out as having less than ideal performance on this specific dataset. These results offer insightful data about how well each regression model predicts Codeforces ranks in comparison, which helps choose the best model to use in real-world applications.

Conclusion

Ensemble techniques like Random Forest and Gradient-Boosting performed better than XGBoost and Linear Regression in target variable prediction. It was determined that the Random Forest Regressor was the best-performing model with the lowest mean square error (MSE) of 0.0003965 and the highest R-squared (R^2) score. This implies that it can effectively lower prediction errors and explain a significant portion of the variance in the dependent variable. With MSE values of 0.0004679 and 0.0003045, respectively, the Gradient Boosting Regressor and Extra Trees Regressor were also doing well. These models' robust R2 values persisted, indicating their consistent ability to generate accurate projections.

On the other hand, with an MSE score of 0.0128, XGBoost's performance was subpar, indicating a need for more investigation or stricter parameter tweaks to improve its effectiveness. With a significantly larger MSE, linear regression showed limited performance, underscoring its limits in managing intricate data correlations and capturing the subtle associations seen in the dataset. Additionally, AdaBoost Regressor's MSE scores were greater than those of the other models, indicating that it performed quite poorly. The performance of different regression models on a particular dataset is empirically analyzed in this paper, which is helpful in choosing the right model for comparable predicting tasks. By contrasting the benefits and drawbacks of various regression algorithms with regard to prediction accuracy,

Findings from a single dataset, however, might not apply universally to other datasets or situations. Because of the limited generalizability of the data, care should be used when extrapolating these findings to other contexts. Furthermore, it's possible that some model configurations were left suboptimal due to incomplete parameter space exploration during hyperparameter tuning, which could have an effect on the models' performance.

In order to further improve model performance, future research initiatives can concentrate on examining more complex feature engineering techniques and hyperparameter tweaking strategies. Methods like automated machine learning (AutoML) and Bayesian optimization could be used to more effectively investigate a larger range of hyperparameters. Additionally, broadening the scope of the study to encompass a range of datasets or domains will aid in confirming the validity of the conclusions and evaluating the adaptability of the models in various scenarios. By doing so, more may be learned about the predictive power of regression models and how well they work in a variety of contexts. This will help to guarantee that the models selected will function dependably no matter what dataset or domain they are used to.