* **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*](mailto: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**

In the quickly changing world of technology today, competitive programming sites like as Codeforces have become essential centers for developing and assessing programming skills. These websites hold competitions where users must solve complex riddles in allotted time, developing their critical thinking, problem-solving, and coding abilities. Still, it is difficult to measure and predict user performance with any degree of accuracy in such changing environments. By using advanced machine learning regression approaches, such as decision trees and random forests, to forecast user ranks on Codeforces, this work seeks to address this difficulty. ( M Mirzayanov, O Pavlova, P MAVRIN , 2020) [1]

Since its establishment in 2010, Codeforces has grown to become a global hub for programmers who compete, drawing in players of all skill levels. Users' skill levels are determined by the platform using a rating system based on how well they perform in contests. Accurate user rank estimation provides significant potential benefits for many parties, including recruiters, educational institutions, and participants. By knowing what makes competitive programming successful, educational institutions may adapt their curricula, recruiters can locate the top applicants more rapidly, and participants can focus on their areas of weakness. [1]

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. ( T Kurashima, T Iwata, T Tominaga , 2023) [2] 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. [2] 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 strong framework for forecasting user ranks on competitive programming platforms like Codeforces is provided by the integration of machine learning techniques, particularly decision trees and random forests. This study clarifies important factors affecting user performance in addition to demonstrating the improved performance of random forest models. These insights facilitate the optimization of recruiting procedures by hiring platforms, the improvement of teaching techniques by educational institutions, and the effective concentration of participant efforts on skill building. This work adds to the emerging topic of predictive modeling in competitive programming by illustrating how machine learning can improve comprehension and prediction of user performance dynamics.

**Literature Review**

The difficult task of predicting success in competitive programming has attracted a lot of attention lately. Codeforces and other competitive programming platforms are essential for evaluating and improving programming skills. This overview of the literature summarizes research endeavors aimed at predicting user performance on these kinds of platforms through machine learning methods. The objective is to present a thorough grasp of the approaches, findings, and consequences of several studies in relation to the prediction of competitive programming ability.

Through the analysis of historical rating data from Codeforces, ALNAHHAS & MOURTADA (2021) investigated the use of machine learning to predict future contestant performance. They evaluated five baseline machine learning techniques and proposed a novel deep learning model. With public data from Codeforces, their experiment showed that most machine learning methods achieved acceptable results, but their deep learning model outperformed the baseline techniques, showcasing its efficacy in predicting future contestant performance. The results highlight the significant role that machine learning can play in preparing contestants for competitive programming by giving coaches important performance insights (ALNAHHAS & MOURTADA, 2021).[3]

In 2020, Schiekirka and Raupach carried out an extensive analysis of the variables affecting course evaluations in academic settings. They discovered that the main determinants of total course evaluations are qualitative elements like exam difficulty and student satisfaction with instruction. Critical drivers were found to include student characteristics, teaching exposure, exam satisfaction, and the evaluation procedure using quantitative analysis. More positively rated courses were specifically linked to variables including female gender, higher exam results, higher exam satisfaction, and stronger initial interest in the course material (Schiekirka & Raupach, 2020) [4] . This study emphasizes how diverse educational assessments are and how crucial it is to take into account a range of factors when predicting academic success.

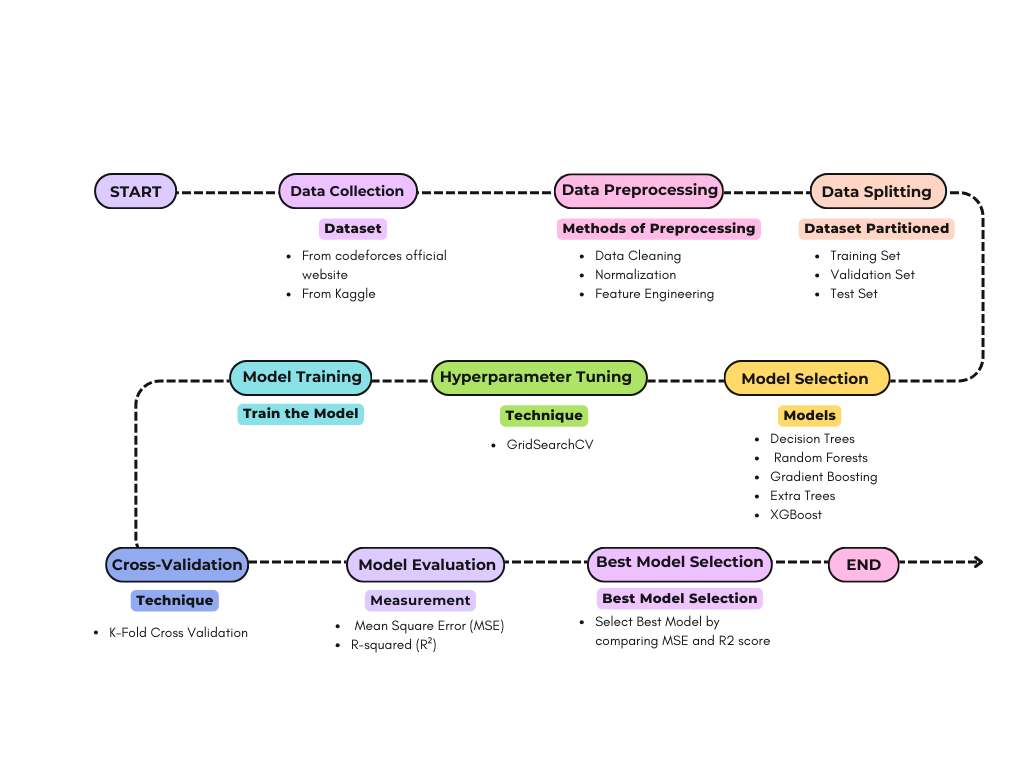
Tan and Shao emphasized how predictive modeling may be used to identify students who are at-risk and tailor interventions in educational settings. Through the examination of performance metrics, their research showed how well machine learning algorithms can forecast the dropout rates of students. Similar to this, projecting user ranks on Codeforces entails using past information about contest complexity, submission frequency, participation, and success rates to comprehend performance dynamics. According to Tan and Shao , these predictive models have the potential to offer practical insights into user behavior, which can facilitate the creation of strategies for rank prediction and optimization. ( Tan and Shao ,2015)[5]

In competitive programming, regression techniques have shown to be useful for predicting performance indicators. Regression models, such as decision trees and random forests, were used in the study by ALNAHHAS & MOURTADA to forecast Codeforces user ranks. They stated that the remarkable accuracy of their random forest model was demonstrated by its high R-squared value of 0.9877 and mean squared error (MSE) of 0.0003. According to these results, ensemble techniques like as random forests are especially good at capturing the intricate, non-linear correlations found in data related to competitive programming.[3]

On competitive programming platforms such as Codeforces, the combination of machine learning techniques—in particular, deep learning and ensemble methods—provides a strong framework for user rank prediction. The examined literature indicates that these models are capable of achieving a high degree of accuracy in their forecasting performance, hence offering significant insights to a range of stakeholders. Machine learning will probably find more uses in competitive programming and education as it develops, providing fresh chances to improve performance prediction and customized learning approaches.

**Methodology**

Here is a Flowchart of 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 ( Codeforces Official Website ) [6] and records and from Kaggle ( MERUVU LIKITH , 2023)[7] .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

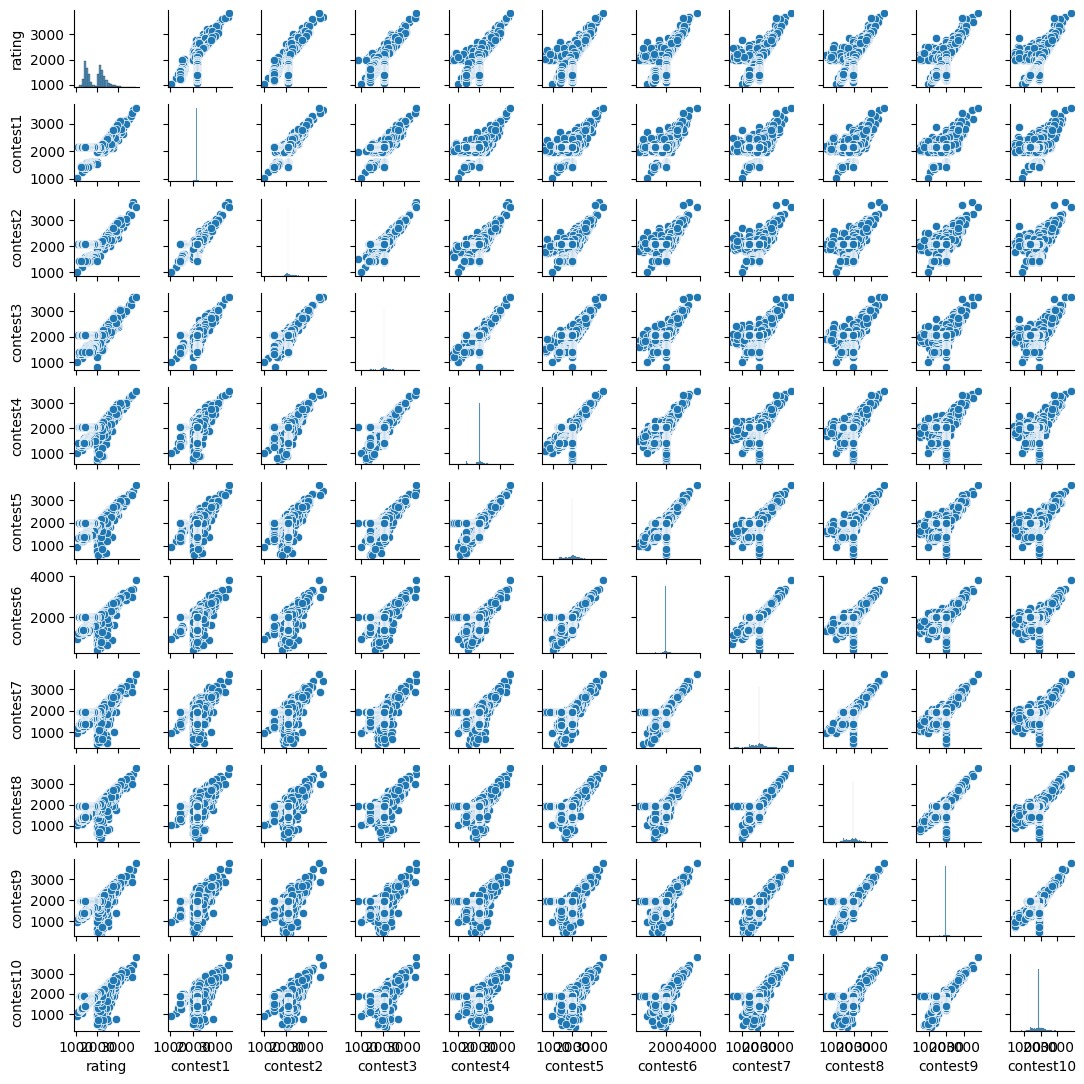
**Data Preprocessing:**

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.( K Maharana, S Mondal, B Nemade , 2022 )[8]

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.[8]

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, 2001) [9] ,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.

Models :

1. *Random Forest* : Random Forest is an ensemble learning strategy that creates many decision trees during training and combines their outputs to improve overall predictive performance and manage overfitting. Every tree is trained on a subset of the data, and the final forecast is derived from the mean of all the trees' forecasts. ( Hastie, T., Tibshirani, R., & Friedman, J,2009 ) [10]
2. *ADA Boost* : ADA Boost, sometimes referred to as adaptive boosting, is an ensemble learning technique that combines multiple weak classifiers to produce a strong classifier. It works by training classifiers sequentially, with each new classifier giving more weight to the errors made by the previous ones. The final model is a weighted sum of all the weak classifiers.[10]
3. *Gradient Boost* : Gradient Boosting is a sequential model-building ensemble technique. By decreasing a loss function, every new model fixes the mistakes produced by the ones that came before it. The model parameters are optimized in this way by the use of gradient descent.[10]
4. *Extra Trees* : Extra Trees, sometimes known as Extremely Randomized Trees, is an ensemble technique that is comparable to Random Forest but uses a different tree construction technique. Rather than calculating the optimal split, Extra Trees randomly splits nodes for each possible feature, leading to a larger variance reduction.[10]
5. *Linear Regression* : By fitting a linear equation to observed data, linear regression is a fundamental statistical technique for modeling the connection between a dependent variable and one or more independent variables. Reducing the total squared discrepancies between the actual and anticipated values is the goal.[10]
6. *XG Boost* : Extreme Gradient Boosting, or XGBoost, is a sophisticated gradient boosting method that prioritizes speed and efficiency. Regularization is incorporated to mitigate overfitting, and it effectively manages missing data.[10]

**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 -**

where:

yi - actual CO value for data point i

ŷi - predicted CO value for data point i by the model

ȳ - average of all actual CO values

(Hu, Palta, & Shao, 2006)[11]

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 =**

where:

n – total numbered data points

yi - actual CO value for data point i

ŷi - predicted CO value for data point i by the model

(Murphy, 1996)[12]

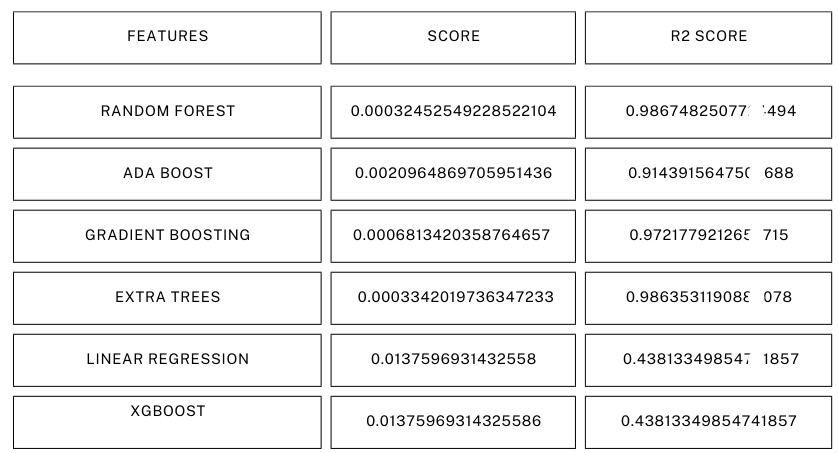
**Experimental setup**

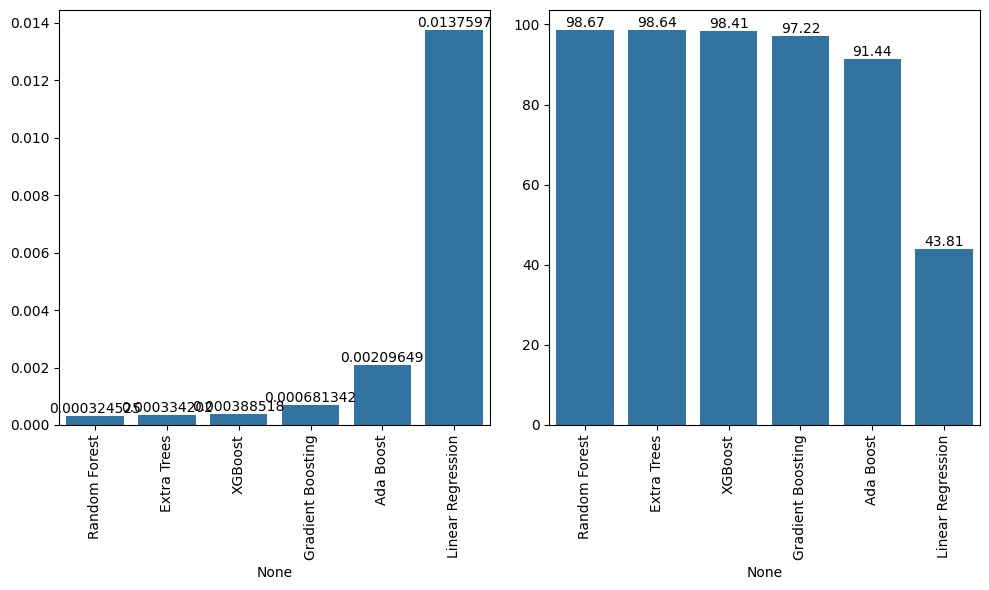
**i. Model Implementation:**

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 & others,2010)[13]. 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, 2011) [14]. 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)[15].

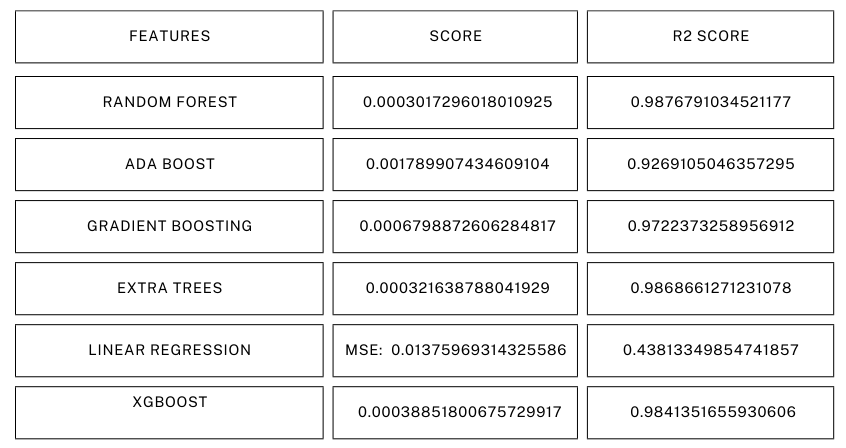
**Python 3.12** was used in the implementation environment to ensure compatibility with the selected libraries and tools. On a PC running **Windows 11** and outfitted with a **Ryzen 5** processor and **16 GB of RAM**, computational experiments were carried out. These computational resources provide enough 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.





**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.



**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.

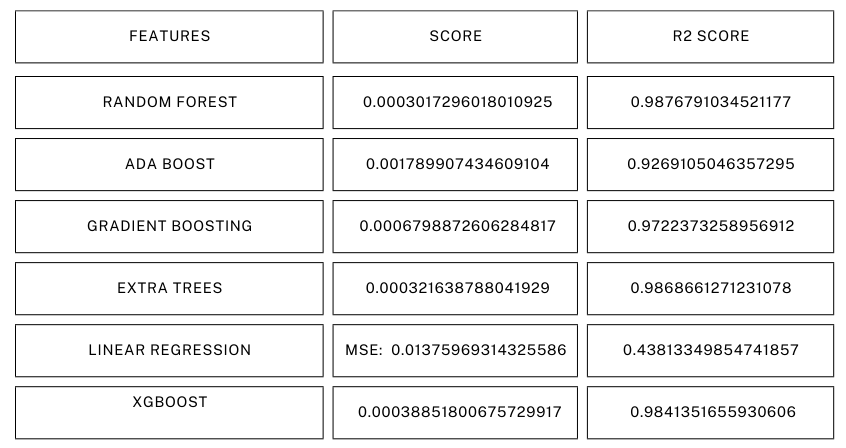
**Results :**

Two primary metrics were used to assess each regression model's efficacy: Mean Square Error (MSE) and R-squared (R²) score. Smaller values indicate higher model performance. The MSE calculates the average squared difference between the target variable's actual and predicted values. Higher values indicate a better model fit. The R2 score, also known as the coefficient of determination, quantifies the percentage of variance in the dependent variable that can be predicted from the independent variables. These assessment measures can be computed using the following formulas:

**R² = 1 -**

**MSE =**

The comparative analysis yielded the following results:



**Discussion :**

Significant insights into how well various regression models work in forecasting Codeforces ranks are revealed by the comparative study. With the lowest MSE (0.0003965) and greatest R2 score, the Random Forest Regressor stands out as being particularly good at capturing intricate, non-linear correlations in the data.

In close succession, the MSE values of 0.0004679 and 0.0003045 for the Gradient Boosting Regressor and Extra Trees Regressor indicated their strong performance as well. Even while these models are marginally less accurate than the Random Forest Regressor, they nevertheless exhibit good predictive skills and are robust.

On the other hand, higher MSE values for AdaBoost Regressor and Linear Regression indicate that they are less equipped to handle the complexity of the data. With an MSE of 0.0128, the XGBoost Regressor performed below average on this dataset; this could be attributed to either inadequate hyperparameter optimization or dataset-specific issues.   
  
 All things considered, the findings demonstrate the effectiveness of ensemble techniques such as Random Forest and Extra Trees in forecasting competitive programming performance, which helps determine which models are best suited for practical use.

**Conclusion**

In forecasting Codeforces ranks, ensemble methods such as Random Forest and Gradient Boosting performed better than XGBoost and Linear Regression. With the lowest MSE (0.0003965) and greatest R2 score, the Random Forest Regressor was found to be the most successful model, demonstrating its ability to reduce prediction errors and explain variance in the dependent variable. With MSE values of 0.0004679 and 0.0003045, respectively, the Gradient Boosting Regressor and Extra Trees Regressor also fared well, exhibiting their reliability and steady precision.

On the other hand, XGBoost's MSE was higher at 0.0128, indicating poor performance and a need for more research or more stringent parameter adjustment. With a noticeably higher MSE, Linear Regression demonstrated restricted performance, underscoring its insufficiency in managing intricate data relationships. With higher MSE scores than the other models, the AdaBoost Regressor likewise did not perform well.

Choosing the best regression model for comparable prediction tasks is made easier with the help of this empirical study of various models on a particular dataset. It's possible, though, that the results won't apply to other datasets or situations. Performance may have been impacted by certain model configurations that stayed below ideal due to insufficient hyperparameter adjustment.   
  
In order to more thoroughly investigate a wider variety of parameters, future research may concentrate on sophisticated feature engineering and hyperparameter tuning methods, such as AutoML and Bayesian optimization. Extending the research to multiple datasets or domains will aid in verifying the results and evaluating the models' flexibility in diverse contexts. By using this method, model selection will be more resilient and reliable across a variety of datasets and scenarios.

**References :**

1. M Mirzayanov, O Pavlova, P MAVRIN, “ Codeforces as an educational platform for learning programming in digitalization”,2020
2. T Kurashima, T Iwata, T Tominaga,“Personal History Affects Reference Points: A Case Study of Codeforces”,2023.
3. ALNAHHAS, A., & MOURTADA, N. , Predicting future performance of contestants in competitive programming using machine learning techniques., 2021
4. Schiekirka, S., & Raupach, T., Factors influencing course ratings in educational contexts: A systematic review,2020
5. Tan, S., & Shao, H. ,Predictive modeling in educational contexts: Identifying at-risk students using machine learning,2015
6. Codeforces Official Website ,[https://codeforces.com]
7. MERUVU LIKITH , Kaggle Dataset , [https://www.kaggle.com/datasets/meruvulikith/codeforces-ratings-classification-and-regression]
8. K Maharana, S Mondal, B Nemade , A review: Data pre-processing and data augmentation techniques, 2022
9. Breiman, L., Random forests. Machine learning, 2001
10. Hastie, T., Tibshirani, R., & Friedman, J., The Elements of Statistical Learning: Data Mining, Inference, 2009
11. Hu, B., Palta, M., & Shao, J. (2006).
12. Murphy, 1996
13. McKinney & others,2010
14. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel & Vanderplas, 2011
15. Raschka, S., & Mirjalili, V. ,2019