Codeforces Rank and Rating Prediction using Machine Learning

Written by:

Aye Thanda (Tania) Htun

Contents

1. Abstract

This project explores the application of machine learning to predict codeforces user rankings and ratings based on their contest performance. The dataset, sourced from Kaggle, includes various contests.

The study employs both regression and classification models: regression predicts user ratings, while classification categorizes users into two rank types (0 or 1) based on a rating threshold of 2200. Initially, five different regression and five classification models were trained, and their performances were evaluated using relevant metrics.

The best-performing models—**Random Forest Regressor for regression** and **Gradient Boosting Classifier for classification**—were further optimized by tuning hyperparameters. The results of this study can be useful in various real-world applications, such as talent identification in HR departments

for competitive programming roles, student assessment in educational platforms, and automated ranking systems for online coding competitions.

2. Introduction

This project utilizes a dataset collected from Kaggle, containing contest results from codeforces, a widely recognized competitive programming platform. The dataset provides comprehensive information about the performance of participants across various contests.
The goal of this analysis is to predict two key variables:

- **Classification**: To predict "Rank-Type", a binary classification of participants' performance based on their rating.
- **Regression**: To predict "Rating", a continuous variable representing the skill rating of codeforces participants. This rating is a numeric measure that reflects the overall performance of users in programming contests.

Dataset Overview
The dataset includes the following columns:

1. **userid**: A unique identifier for each participant on codeforces.
2. **rating**: The participant's skill rating, an integer that signifies their performance in contests.
3. **rank-type**: A binary classification indicating whether a participant's rating is above 2200 (top-ranked) or not (non-top-ranked).
4. **contest-results**: Contest-related columns labeled as 'contest1' through 'contest10'. Each of these columns represents the results of different contests, with the values being four-digit numbers.

3. Methodology and Model Deployment

This section provides a detailed explanation of the data preprocessing steps, feature engineering, model selection, and evaluation methodologies employed throughout this project.
The goal is to predict two target variables: **rank-type** (binary classification) and **rating** (regression), based on contest performance data from codeforces participants.
The approach was structured as follows:

3.1 Data Preprocessing

The first step in the methodology was to preprocess the dataset to make it suitable for machine learning models:

Dataset Cleaning

The dataset was initially cleaned by removing duplicate records and handling missing values. The **drop_duplicates()** and **dropna() functions** were used to ensure that the data used for training was clean and free from redundancy and missing values.

Feature Transformation - Rank-Type Column

The original dataset contained a rank-type column with multiple categories (7-8 unique rank types). However, the classification task required a binary output: **0 (non-top-ranked)** and **1 (top-ranked)**.

seria	rank-type	rating	contest.	contest.	contest:	contest	contest:	contest	contest	contest	contest:	contest.
3143927301	Candidate Master	2115	2078	2055	2115	2047	2024	2010	1953	1936	2042	2045
1876577621	Master	2254	2194	2114	2152	2179	2211	2154	2170	2141	2157	2209
6397741793	Master	2344	2120	2206	2147	2234	2294	2090	2089	2072	2085	2114
3090123616	International Master	2224	2224	2222	2166	2116	2029	2113	2104	2096	2115	2163
9564162806	Legendary Grandmaster	2128	2128	2120	2072	2018	1963	2039	1932	1963	1960	1886
3796242163	Candidate Master	2139	1987	2040	2101	2139	2102	1996	1765	1734	1581	1769
8332225684	Master	2132	2114	2132	2094	2063	1960	1947	1987	1822	1716	1454
5454867391	Candidate Master	2294	2034	2016	1873	1892	1863	1825	2012	2164	2125	2208
7429241387	Newbie	2170	2117	2170	2149	2071	2084	2023	1998	1910	1815	1973
1747131862	Newbie	2242	2242	2005	2041	2029	1851	1756	1746	1643	1712	1507
1292939055	Pupil	2191	2147	2143	2191	2150	2107	2024	2095	2141	2065	2092
4227484046	Pupil	2231	2231	2100	2166	2099	2041	2003	1941	1891	1843	1833
6519694127	Master	2247	2247	2054	2025	2046	2137	2058	2000	2022	2022	2002

To achieve this, the **"rank-type"** column was transformed using a threshold rule based on the participants' **rating**:

- If a participant's **rating** was **greater than 2200**, their **rank-type** was set to **1** (top-ranked).
- If the rating was **2200 or lower**, the **rank-type** was set to **0** (non-top-ranked).

This step was essential to meet the project requirements of a binary classification model.

userid	rank-type	rating	contest1	contest2	contest3	contest4	contest5	contest6	contest7	contest8	contest9	contest10
7611210823	0	2242	2139	2158	2195	2108	2125	2213	2219	2231	2193	2201
5669498982	1	2673	2464	2556	2564	2601	2647	2645	2673	2666	2642	2544
9104862886	0	2146	2146	2121	2051	2012	2009	2029	1945	1926	1971	2049
5696423157	1	2728	2728	2599	2426	2506	2426	2477	2410	2491	2372	2215
9412604737	0	1992	1992	1777	1834	1923	1675	1798	1720	1940	1651	1289
2466737336	1	2478	2478	2200	2132	2119	2071	1947	1801	1775	1686	1619
7454794441	1	2731	2527	2495	2557	2527	2498	2445	2409	2408	2486	2560
9448189295	0	2244	2244	2157	2062	2158	2088	2020	1996	1987	2032	2066
4510669982	1	2898	2795	2779	2751	2770	2789	2741	2773	2864	2748	2704
4316527424	0	2121	2121	2084	2084	1998	1961	1977	1966	1920	2005	2118
8045623301	0	2266	2190	2237	2143	2218	2175	2205	2092	2127	2181	2251
6565997964	0	2356	2113	2245	2356	2215	2084	1987	1993	1716	1176	656
5486469395	0	2199	2070	1998	2038	2040	2051	2164	2123	2156	2065	2013
1139194040	0	2257	2257	2034	2086	2170	2182	2043	2154	2076	2148	2077
2996053410	1	2565	2565	2537	2508	2482	2406	2465	2467	2425	2377	2438

Feature Selection

The next step was to select the relevant features for both classification and regression tasks.

- **For Regression**: The target variable is "rating", and the features include all other columns except "rating", "rank-type", and "userid". The "rank-type" column will be removed as it is the target for the classification task.
- **For Classification**: The target variable is "rank-type", and the features include all other columns except "rank-type" and "userid". The "rating" column is excluded from the features because it was used to create the threshold for generating the "rank-type" label.

Data Splitting

The dataset was split into **training** and **testing** sets using an 80-20 split. This ensured that the models were trained on a substantial portion of the data while keeping enough unseen data for evaluating model performance.

Feature Scaling

To ensure that the machine learning models could handle the data effectively, all feature values were standardized using **StandardScaler**. This step transformed the features to have a mean of 0 and a standard deviation of 1, which is crucial for algorithms sensitive to feature scaling, such as Gradient Boosting and Random Forest.

3.2 Model Selection and Training

The project aimed to evaluate a variety of machine learning algorithms, using both **regression** and **classification** approaches.

Best Regression Model Selection

For the regression task (predicting the rating), five regression models were trained:

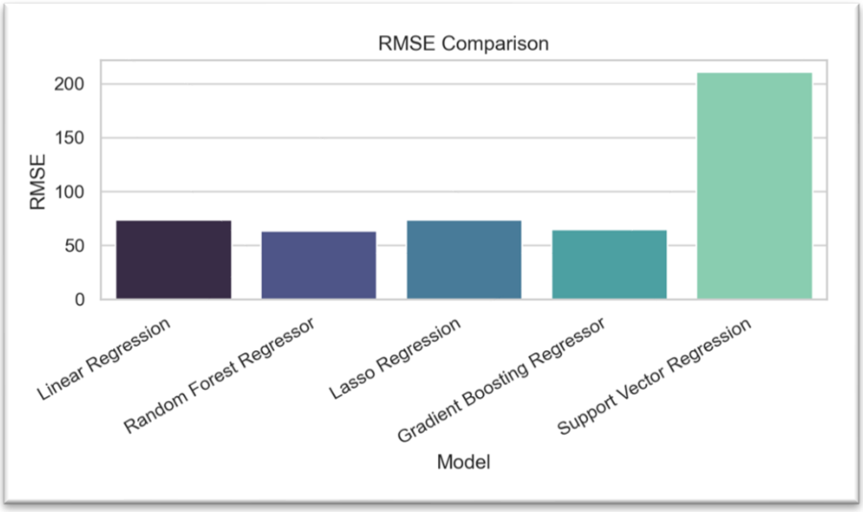
- **Linear Regression**: Basic model that assumes a linear relationship between input features and the target variable.
- **Random Forest Regressor:** An ensemble method using multiple decision trees.
- **Gradient Boosting Regressor:** An ensemble model that builds trees **sequentially**, focusing on correcting the errors of previous trees
- **Support Vector Regressor (SVR):** Based on Support Vector Machines (SVMs) and can model complex, non-linear relationships.
- **Lasso Regression:** Linear regression with **L1 regularization** (helps prevent overfitting and can eliminate unnecessary features).

After training these models, the performance was evaluated using metrics such as **Root Mean Squared Error (RMSE)**, **Mean Absolute Error (MAE)**, and **R-squared (R^2)** score.

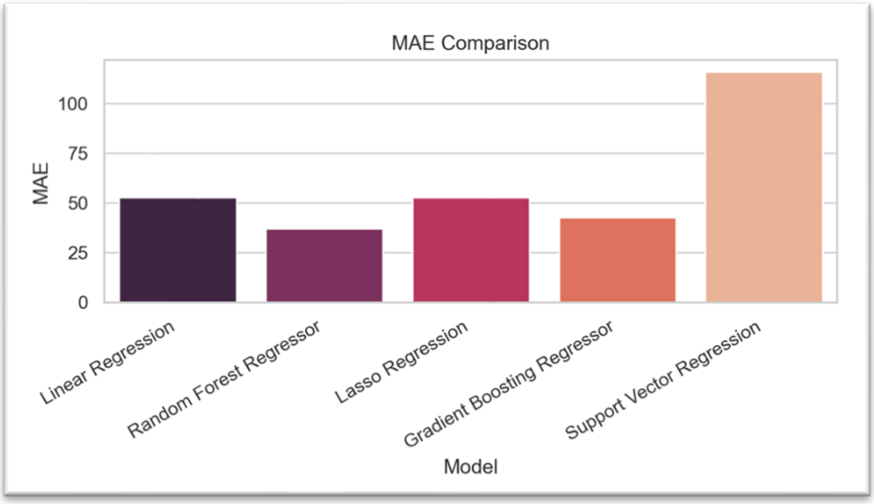
Regression Models	**MSE**	**MAE**	**R^2 Score**
Linear Regression	73.69	52.81	0.9260
Random Forest Regressor	**63.59**	**37.17**	**0.9449**
Lasso Regression	73.64	52.73	0.9261
Gradient Boosting Regressor	64.76	42.81	0.9428
Support Vector Regression	211.14	116.06	0.3923

<u>Data Visualization on Metrics of 5 Regression Models</u>

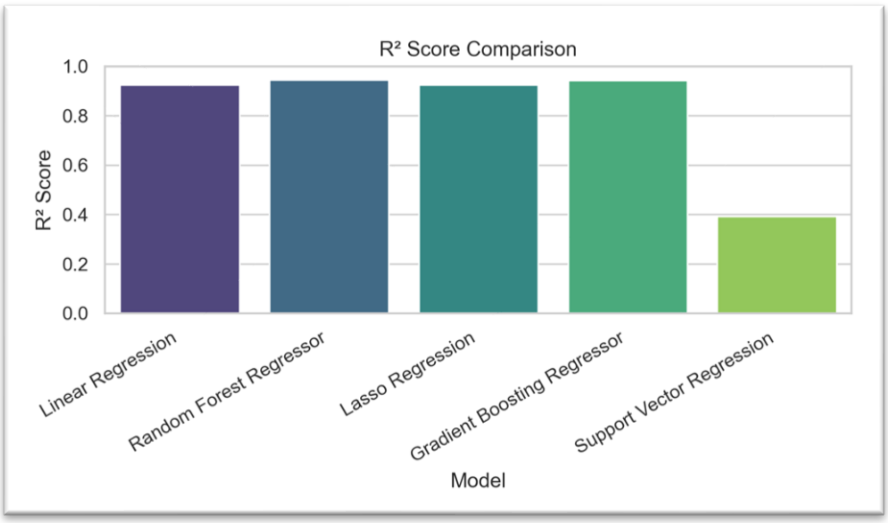
Mean Squared Error (MSE) results of 5 Regressors



Mean Absolute Error (MAE) results of 5 Regressors



R² Score results of 5 Regressors



Analyzing the results based on the three key metrics:

- **Mean Squared Error (MSE)** → Lower is better.
- **Mean Absolute Error (MAE)** → Lower is better.
- **R^2 Score** → Higher is better (closer to 1 is ideal).

✓ **Best Model → Random Forest Regressor**

- Lowest MSE (63.59)
- Lowest MAE (37.17)
- Highest R^2 Score (0.9449)

Best Classification Model Selection

For the **classification task** (predicting the rank-type), five classification models were tested:

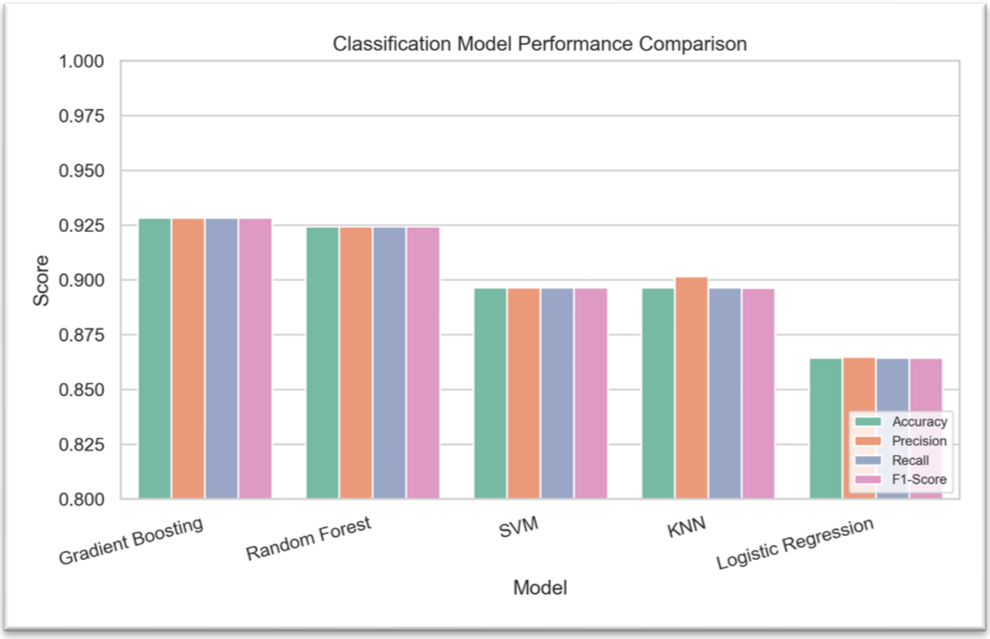
- **Logistic Regression:** A simple, fast, and interpretable linear model best for binary classification.
- **Random Forest Classifier:** An ensemble of decision trees that improves accuracy and reduces overfitting.
- **Gradient Boosting Classifier:** Builds trees sequentially, where each tree corrects the errors of the previous one.

- **Support Vector Machine (SVM) Classifier:** Finds the best boundary (hyperplane) to separate classes.
- **K-Nearest Neighbors (KNN) Classifier:** A simple, instance-based method that classifies based on the majority label of the nearest neighbors.

The models were evaluated using **accuracy**, **precision**, **recall**, and **F1-score** metrics.

Classification Models	Accuracy	Precision	Recall	F1-Score
Logistics Regression	0.8645	0.8648	0.8645	0.8645
Random Forest Classifier	0.9243	0.9243	0.9243	0.9243
Support Vector Machine (SVM)	0.8964	0.8965	0.8964	0.8964
KNN	0.8964	0.9016	0.8964	0.8962
Gradient Boosting	**0.9283**	**0.9284**	**0.9283**	**0.9283**

Data Visualization of 5 Classifiers' Metrics



<u>Model Analysis</u>

Top Performer: Gradient Boosting

- Best scores in all four metrics (Accuracy, Precision, Recall, F1-Score).
- Very well-balanced: high prediction power and reliable classification.
- Ideal if performance is your top priority (but can be slower to train than others).

Strong Contender: Random Forest

- Almost as good as Gradient Boosting.
- Slightly lower but still highly consistent across all metrics.
- Tends to be faster and more interpretable than Gradient Boosting.

Support Vector Machine & KNN

- Both perform better than Logistic Regression, but a bit below tree-based models.
- KNN has slightly higher Precision, suggesting it makes fewer false positives.
- SVM is balanced, clean, and close in all metrics — good if the dataset is small and well-scaled.

Logistic Regression

- The lowest performer across all metrics.
- Still decent at ~86%, but might not be suitable if higher accuracy is required.
- Best used as a baseline or when model interpretability is crucial.

✓ **Best Model → Gradient Boosting Classifier**

- Highest Accuracy (92.83%)
- Highest Precision (92.84%)
- Highest Recall (92.83%)
- Highest F1-Score (92.83%)

Hyperparameter Tuning

To enhance model performance, hyperparameter tuning was applied. Specifically, the **number of estimators** (n_estimators) was tuned for both the **Random Forest Regressor** and **Gradient Boosting Classifier** models.

These are the metrics of **Random Forest Regressor** based on 3 estimators (100, 200 and 300).

Random Forest Regressor			
N_estimator	**MSE**	**MAE**	**R^2 Score**
100	***63.59***	***37.17***	***0.9449***
200	64.28	37.63	0.9437
300	64.74	37.74	0.9429

Model Performance Summary (Different n_estimators):

- 100 Estimators → Best Performance
  - Lowest MSE (63.59) and MAE (37.17)
  - Highest R^2 Score (0.9449)
  - Most accurate and reliable among the three.
- 200 & 300 Estimators → Slight Drop
  - MSE and MAE slightly increased.
  - R^2 Score decreased to 0.9437 and 0.9429, indicating minor performance decline.
  - More estimators did not improve the model.

✓ **Best estimator for Regression = 100**

These are metrics of **classification model** based on 3 estimators (100, 200 and 300).

Gradient Boosting Classifier				
N_estimator	**Accuracy**	**Precision**	**Recall**	**F1-Score**
100	0.9283	0.9284	0.9283	0.9283
200	0.9323	0.9323	0.9323	0.9323
300	***0.9363***	***0.9363***	***0.9363***	***0.9363***

Performance Summary (Different n_estimators):

- **300 Estimators → Best Overall Performance**
  - Highest Accuracy, Precision, Recall, and F1-Score (0.9363)
  - Shows consistent improvement with more estimators.
- 200 Estimators → Moderate Performance
  - Slight increase across all metrics (0.9323) compared to 100.
  - Indicates better learning with added estimators.
- 100 Estimators → Lowest Performance
  - Still strong (0.9283), but slightly behind higher settings.

✓ **Best estimator for Classification** = 300: offers the **best balance** and most accurate classification.

4. Model Evaluation

According to the previous training, final selected models with best estimators are:

- ✓ **Regression**: Random Forest Regressor (n_estimator=100)
- ✓ **Classification**: Gradient Boosting Classifier (n_estimator=300)

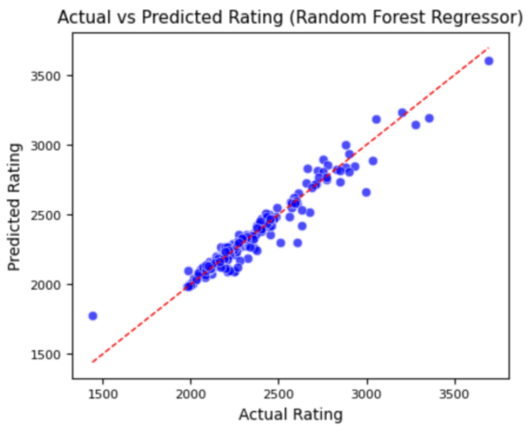
Random Forest Regressor Model Evaluation

The **Random Forest Regressor** was evaluated based on three key regression metrics:

- Root Mean Squared Error (RMSE) = 63.59
- Mean Absolute Error (MAE) = 37.17
- R^2 Score = 0.9449

Regression Model Results

Metric	Value
RMSE	63.59
MAE	37.17
R² Score	0.9449



Performance Summary

- **Low RMSE (63.59)** → Indicates small average prediction errors.
- **Low MAE (37.17)** → Suggests high accuracy in predicting actual values.
- **High R^2 Score (0.9449)** → Explains **94.49%** of the variance in the target variable.

Random Forest Regressor demonstrates **strong predictive power** with **low error rates** and **high explained variance**, making it well-suited for this regression task.

Gradient Boosting Classifier Model Evaluation

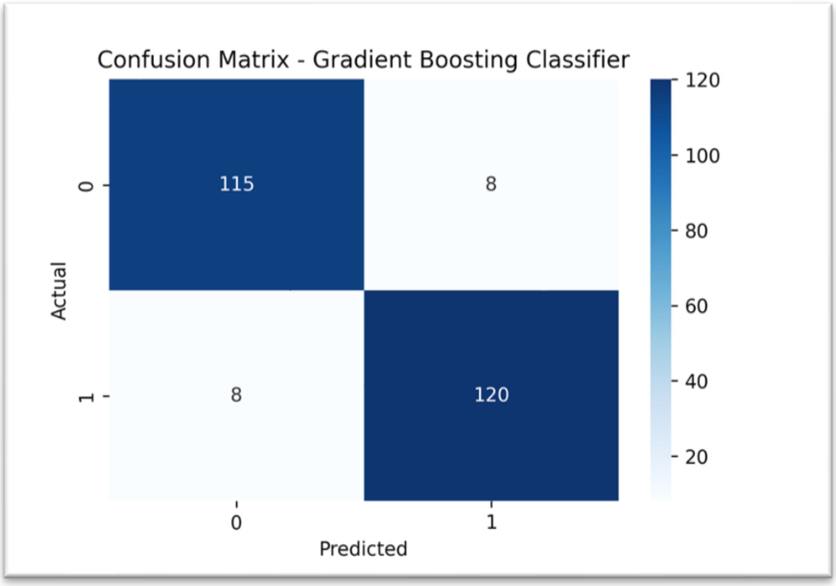
The **Gradient Boosting Classifier** was evaluated using multiple classification metrics, including accuracy, precision, recall, and F1-score. Below is the interpretation of the model's performance.

```
===== Classification Model: Gradient Boosting Classifier =====
Accuracy: 0.9363

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93       123
           1       0.94      0.94      0.94       128

    accuracy                           0.94       251
   macro avg       0.94      0.94      0.94       251
weighted avg       0.94      0.94      0.94       251
```



Gradient Boosting Classifier Model Evaluation Summary –
Overall Accuracy: 93.63%
Reflects the model's strong overall classification ability across both rank-types.
Class 0 (rank-type = 0):

- Precision: 0.93 → 93% of predicted class 0 instances are correct.
- Recall: 0.93 → The model correctly identifies 93% of all actual class 0 instances.
- F1-Score: 0.93 → Indicates balanced and reliable performance for class 0.

Class 1 (rank-type = 1):

- Precision: 0.94 → 94% of predicted class 1 instances are accurate.
- Recall: 0.94 → The model captures 94% of all true class 1 cases.
- F1-Score: 0.94 → Highlights excellent and consistent classification of class 1.

Classification Report Summary

- **High Accuracy (94%) →** Overall, the model performs strongly across both classes.
- **Balanced Precision & Recall (93–94%) →** No major bias toward either class**.**
- **Consistent Scores →** Macro and weighted averages show stable performance across class distributions.

Gradient Boosting Classifier model is robust, balanced, and suitable for classifying both categories effectively.

5. Conclusion

Regression – Predicting Rating
The **Random Forest Regressor** delivers **outstanding predictive performance**, characterized by **very low error rates** (MSE and MAE) and a **high R^2 score**, indicating excellent explained variance. These results confirm the model's **strong suitability and reliability** for regression tasks involving rating predictions.

Comparison between Actual Values and Predicted Values (Random Forest Regressor)

contest1	contest2	contest3	contest4	contest5	contest6	contest7	contest8	contest9	contest10	Actual_Rating	Predicted_Rating
-0.089697	0.014509	0.538302	0.342588	0.633083	0.311375	0.477553	0.517282	0.366068	0.637305	2318	2354.98
-0.204286	-0.647113	-0.148340	0.133267	0.183965	0.541058	0.747027	0.578036	0.342803	0.109238	2316	2286.34
-0.870601	-1.240153	-1.324890	-0.632995	-0.820160	-0.847639	-0.743609	-0.649847	-0.547058	-0.346933	2029	2031.03
0.003672	0.107297	-0.071189	-0.180713	0.052516	0.095827	-0.102329	0.255077	0.243930	0.520783	2235	2280.58
-0.624446	-0.683422	-0.372077	-0.812413	-0.743481	-0.978381	-0.822064	-0.608278	-0.378391	-0.490725	2087	2099.27
...	...	...	...	...	...	...	...	...	...	...	...
-0.076965	-0.271925	0.742752	0.492102	0.034259	0.240703	0.303588	0.382982	0.755745	0.748868	2371	2353.27
-0.942749	-1.361181	-1.201448	-1.563723	-0.560913	-0.349404	-0.719732	-0.426014	-0.340587	-0.341974	2017	2045.15
0.733647	0.462314	0.738894	1.254627	0.771835	1.629400	1.326909	1.610865	1.767744	1.502541	2997	2661.62
-0.026037	-0.308234	-0.518664	-0.502170	-0.108144	-0.243397	-0.828886	-1.650700	-1.951642	-2.260865	2228	2232.17
-0.450441	-0.651148	-1.459904	-0.827364	-0.491537	-0.434210	-0.163728	-0.230960	-0.328954	-0.024638	2128	2141.31

<u>**Classification – Predicting Rank-Type**</u>

The **Gradient Boosting Classifier** exhibits **exceptional classification performance**, achieving **high accuracy** and **well-balanced precision, recall, and F1-scores** across both classes. This makes it a **highly effective model** for real-world applications such as HR recruitment systems, ranking algorithms in competitive platforms, and employee performance assessments. Its ability to precisely classify rank-types enhances decision-making processes in data-driven and evaluation-heavy environments.

Comparison between Actual Values and Predicted Values (Gradient Boosting Classifier)

contest1	contest2	contest3	contest4	contest5	contest6	contest7	contest8	contest9	contest1	Actual_RankType	Predicted_RankType
-0.089697	0.014509	0.538302	0.342588	0.633083	0.311375	0.477553	0.517282	0.366068	0.63730	1	1
-0.204286	-0.647113	-0.148340	0.133267	0.183965	0.541058	0.747027	0.578036	0.342803	0.10923	1	1
-0.870601	-1.240153	-1.324890	-0.632995	-0.820160	-0.847639	-0.743609	-0.649847	-0.547058	-0.34693	0	0
0.003672	0.107297	-0.071189	-0.180713	0.052516	0.095827	-0.102329	0.255077	0.243930	0.52078	1	1
-0.624446	-0.683422	-0.372077	-0.812413	-0.743481	-0.978381	-0.822064	-0.608278	-0.378391	-0.49072	0	0
...	...	...	...	...	...	...	...	...	..	...	...
-0.076965	-0.271925	0.742752	0.492102	0.034259	0.240703	0.303588	0.382982	0.755745	0.74886	1	1
-0.942749	-1.361181	-1.201448	-1.563723	-0.560913	-0.349404	-0.719732	-0.426014	-0.340587	-0.34197	0	0
0.733647	0.462314	0.738894	1.254627	0.771835	1.629400	1.326909	1.610865	1.767744	1.50254	1	1
-0.026037	-0.308234	-0.518664	-0.502170	-0.108144	-0.243397	-0.828886	-1.650700	-1.951642	-2.26086	1	1
-0.450441	-0.651148	-1.459904	-0.827364	-0.491537	-0.434210	-0.163728	-0.230960	-0.328954	-0.024638	0	0

================================== **Thank you** ====================================