Code Forces Rating Predictor

Satvik Ramaprasad
satviksunkam.ramaprasad@iiitb.org
IMT2016008

Raghavan G V
raghavan.gv@iiitb.org
IMT2016099

Ananth Shreekumar
ananth.shreekumar@iiitb.org
IMT2016129

***Abstract*—Competitive Programming is a mind sport where programmers are expected to solve challenging problems in a given timeframe. It is known for involving problem statements that require creative thinking on the part of the programmers. Competitive programmers are tested on their months of hard work and persistence, alongside their ability to solve a problem and build a code. This sport has been gaining popularity across the students of India since half a decade**

This paper is a report on a project that was conducted to predict a user's rating on Code Forces - A Competitive Programming Platform - using concepts from Machine Learning. We propose to create a model that will predict a user's rating based on the problems that he has solved. We solve two problems - given a user's data, predict his maximum rating using regression approaches, and given a user's data, predict his rank using classification techniques. Rank is determined by his rating.

I. Introduction

Programming is one the most sought-after skills in the World today. However, a good programmer must also know algorithms to solve problems and to come up with his own algorithms for an efficient solution.

Code Forces is a website where people can sign up and hone their skills in programming and algorithms in a language of their choice. The submitted solutions are run through a set of test cases to check their correctness. Codeforces also conducts weekly contests, during which the contestants are expected to code up a solution to the given problems. The programmer's rating is affected based on criteria - the number of problems solved, the time taken to solve, and the number of wrong submissions for a problem during the contest.

In this paper, we also present a novel approach on using the information about the quantity and quality of the problems that he has practiced all along and we want to predict the current rating of the user.

By doing this we give users a vague prediction of what his / her current rating will be according to her practice.

II. Dataset

Since this is a new problem, there was no dataset available which captured all the necessity "features". We had to generate our own dataset using a combination of the APIs provided by Codeforces and scraping some pages of their website. Given the time consuming nature of crawling/scraping, we decided to first select data attributes from which we can extract features.

A. User Dataset

Using Codeforces API, we obtained raw user data of all users on the platform. There were a total of 158K users from several countries. Given the time constraint, we decided to restrict ourselves to Indian users. Upon filtering, we obtained a total of 11.5K Indian users. We grouped users based on their organization. The total number of organizations turned out to be 959. We again filtered only on organizations which had more than 20 users. After all the filters, the dataset was reduced to 83 organizations and 6.4K users.

Data attributes collected

- Country
- Organization
- Handle
- Timestamp of user creation
- Contributions
- Number of Friends
- Rating
- Max Rating
- Rank

B. User - Submissions Dataset

Using Codeforces API, we obtained the total list of submissions by the user. This was used to perform a join between each user and problem submission data. This was crucial to aggregate and extract features from the submission data.

C. Problem Dataset

Using Codeforces API, we obtained raw problem data of all problems on the platform. There were a total of 4637 problems with varying difficulties. Since difficulty level of the problems were not provided by the API, we had to scrape it from the website.

Data attributes collected

- contestId
- index
- name
- type
- tags
- Difficulty

III. Feature Engineering

We performed a join on the user submission data and problem data based on problem id. We then performed statistical aggregation on the result.

We computed the mean, median and the mean of the top 20% of the problem's difficulty solved by the user. The total number of problems solved successfully was also extracted as a feature.

We classified problems into buckets based on their difficulty score and performed a count. We further classified problems based on their problem tags and performed a count.

Data Features Extracted

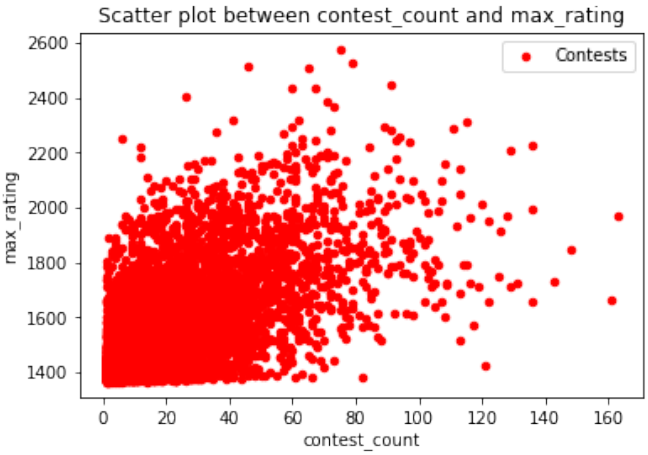
- avg_difficulty
- avg_difficulty20
- median_difficulty
- duration
- tags (set)
- difficulty_buckets (set)
- problem_count
- contest_count

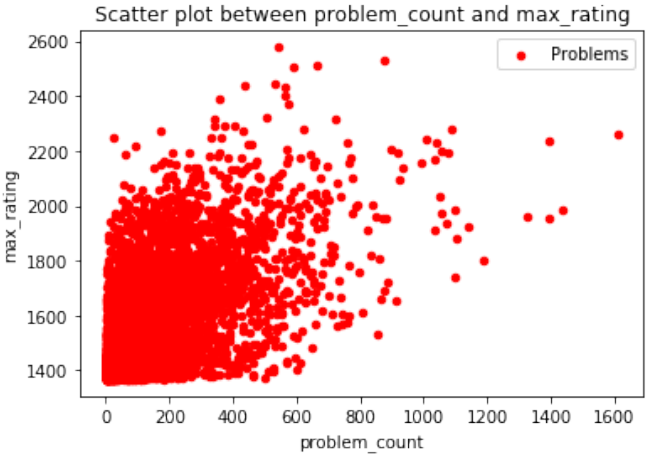
IV. Data Visualisation

One of the major consequences of this project was that we could empirically prove that a person generally has a higher rating if he has solved more problems than his peers. We plotted the rating alongside the number of problems solved for the users in the dataset to achieve this.

V. Models and Training

We tried multiple models to get an idea of which performs better.





A. Rating Predictor

Rating prediction is a regression problem. We built Linear Regression, Random Forest and XG Boost classifiers on the dataset.

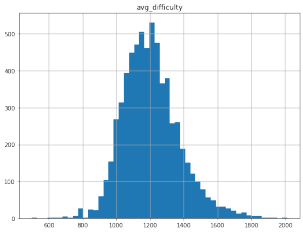
B. Rank Predictor

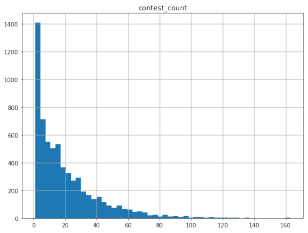
Rank prediction is a classification problem. We built Logistic Regression, Random Forest and XG Boost classifiers on the dataset.

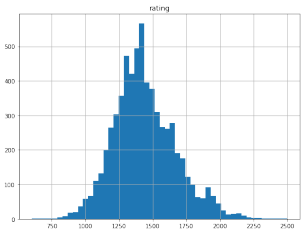
VI. Results and Analysis

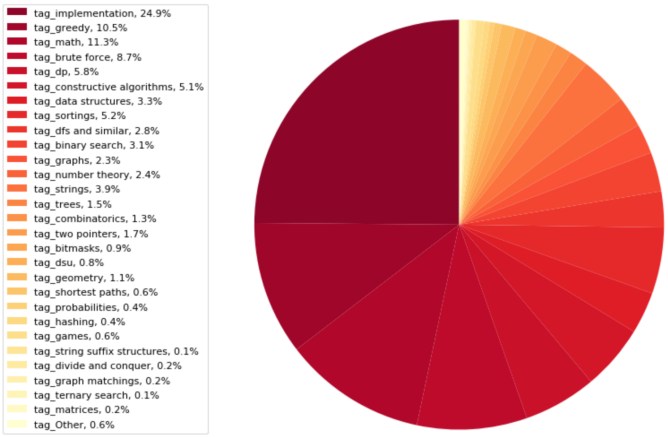
A. Rating Predictor

Since rating prediction is a regression problem, we used R2 score and RMS Error as our metric. XGBoost algorithm gave our best score with a R2 score of 0.82 and RMS Error of 75.11. Codeforces on an average gives around -100 to $+100$ rating change to a contestant after every contest based on relative performance. Since our RMS Error is around 75 points we could certainly tell that we are only one or two contests behind the actual rating of the user.









B. Rank Predictor

Rank prediction is classification problem where the rank of the given user is predicted. As we clearly can see that predicting a user as 'expert' (category - 4) when he is 'candidate master' (category - 5) is not a very bad result as they both are very close to each other in the ranking graph. But since the model considers these both as completely independent categories we won't be able to get a good classification. The best score that we could get was using logistic regression which gave us the mean score of 0.604.

VII. Issues Faced

Data collection turned out to be tricky and time consuming. Network issues slowed down our data crawling. Moreover codeforces blocked our IP addresses for repeated requests. We resorted to using VPNs to get through the restriction. We also tried parallizing our requests using multiple threads to speed up the process.

VIII. Future Scope

1. All the countries in the World could be included to obtain a better dataset to build a model.
2. We could try to predict a user's rating sometime in the future if he continues at the same level of effort.

Acknowledgment

The authors would like to thank professors for their continued guidance, support and encouragement. Further the authors would like to thank the teaching assistants for their valuable advice and encouragement.

References

[1] Codeforces api documentation. [Online]. Available: https://codeforces.com/api/help
[2] "Scikit-learn documentation." [Online]. Available: http://scikit-learn.org/stable/documentation.html
[3] "Pandas documentation." [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/
[4] "Numpy documentation." [Online]. Available: https://docs.scipy.org/doc/
[5] "Xgboost documentation." [Online]. Available: https://xgboost.readthedocs.io/en/latest/
[6] "Xgboost documentation." [Online]. Available: https://readthedocs.org/projects/beautiful-soup-4/