GitHub Profile Analysis

ISM 6136 – Data Mining Project

Group 7

Harishma Parthiban

Ranjan Ravi

Sumitha Babu

Varsha Umed Dugar

Background

GitHub is a Git Repository hosting service used for code sharing. While Git is a command line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project. Since it is a cloud-based tool, the code is conveniently visible across the entire organization, facilitating every participant's contribution. It allows collaboration with developers from across the world. One of the most important elements in Git is its Version Control System (VCS) which records all the modifications made to the files so that if a specific version of files is required in the future, one can easily retrieve those from the history. Fork, Pull Request and Merge are the features which make GitHub so powerful. Forking enable us in copying a repository from one user's account to another. This helps us to take a project that we don't have write access to and modify it as per our needs. If we want to share our changes, then we can send a notification called pull request to the original owner which the owner can accept or reject. If the owner accepts the pull request, then the code gets merged with the original repo.

If we've ever applied for a technical role we've probably sent the company a link to our GitHub profile. The information on this profile can give a good indication of one's coding ability and fit within a team. The downside to all this information is that it may take a recruiter a long time to assess it. To save time, machine learning could potentially be used to automatically rate your coding ability.

Features of GitHub

1. Integrated Issue Tracking
2. Collaborative code review
3. Easily manage teams within organizations
4. Syntax highlighted code and rendered data
5. Security
6. Robust API
7. GitHub Pages

The above attributes would help in evaluating GitHub profiles of the interviewing candidates. This would show how active the candidate is and how well his coding strengths are.

Solution Methodology

As the Demand for qualified tech professionals are high, recruiters are looking for innovative ways to identify and source excellent candidates. One interesting way is to assess a potential employee's GitHub profile. GitHub profiles gives the recruiters a good indication of a candidate's ability to code and fit well in a team and conclude how relevant their skill sets are for a role.

GitHub profile allows a recruiter to view information by looking at repositories and contribution activity and recognize how experienced a candidate is. The recruiters would want to assess each candidate's profile to have a deep understanding of their capability. One disadvantage of manually assessing is that the recruiter may spend a longer time in evaluating each profile to determine their technical experience.

So, here we try to build a model to identify potential candidates based on their GitHub profiles in order to minimize the time and effort the recruiters put in during the recruitment process. Each profile is evaluated based on the scores assigned to them. The scores are calculated based on Total Stars, Total Forks, Total Followers, Repository count etc., of the candidates GitHub profile.

For the study, we took out dataset from the GitHub API. The dataset contains 11 attributes, and they are then further used in building a classification model to predict whether the candidate's profile is good, bad or average. We have collected the attributes of 2019 GitHub Profiles to build this classifier.

We have used the following classification algorithms – Multiclass Decision Jungle, Multiclass Neural Network.

Data Collection and Data Characteristics

We created a data collection model using PyGitHub APIs for pulling 2019 GitHub user profiles.

APIs used for collecting data

1. Search users API -To get users list in a given year range
2. Get user details API - To get details like followers, following, contributions.
3. Repos API -To get repo details for each user including number of stars, forks, languages used.

We bucketed the data being collected for each year starting from 2000 and pulled 100 profiles into each bucket.

The attributes of the dataset are total_cont, followers, following, repos, stars, forks, language, n_lang, foll_ratio, cont_repo_ratio, score, category and it's statistical summary can be viewed in Figure 1.

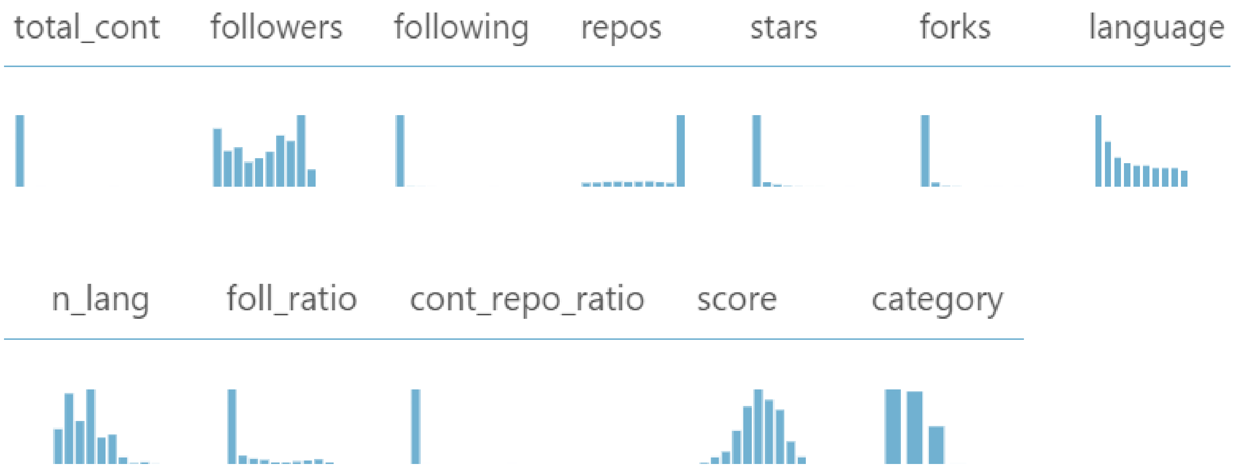


Figure 1. Attributes of Dataset

Description of the attributes

1. **total_cont -** Total number of contributions, a user has made till date.
2. **followers -** Total number of followers of a user.
3. **following** - Total number of profiles a user is following.
4. **repos** – Total number of repositories that the user has.
5. **stars** – Total stargazers count across repositories of a user.
6. **forks** – Total number of forks that a user's repositories have.
7. **language** – All the programming languages that a user has used.
8. **n_lang** – Count of the languages that a user knows/has used.
9. **foll_ratio** – Ratio between the total number of follower and following count.
10. **cont_repo_ratio** – Ratio of contributions over repositories.
11. **score -** The score given to each Github user profile.
12. **Category –** The category that the profile belongs to. (Above average and Below average).

Independent Variables

The IVs that we considered are as follows

1. Followers
2. Following
3. Repos
4. Stars
5. Forks
6. total_cont
7. n_lang.

Dependent Variable

We categorized the DV (category) based on the score calculated. Users are categorized as follows.

1. The profiles having scores >70 - Good
2. The profiles having score in the range 50-69 - Average
3. Th e profiles having score in the range 0-49 - Low

Score calculation/Feature selection

The score calculation was done based on the following factors.

1. User profile overall popularity – To calculate this, we considered the followers, star rate and repo counts.
   Star rate = Stars/Repos
   Profile's overall popularity = followers / repos + star rate
2. Repo popularity – To calculate this, we considered total stars, forks across the total number of repos.
   Repo popularity Stars + Forks / Repos
3. The number of contributions a user has made.
4. The number of languages a user has used.

Giving high weightage to factors 1 and 2, we calculated the overall score.

Data Pre-processing and Split

Since the data returned by the APIs were JSON (structured), we had clean data. But we had to normalize the independent variables for modelling, as the attributes like followers, contributions ranged from six to more than ten thousand. We split the data in the ratio 70:30, 70 percent being train data and 30 percent being test data.

Algorithm comparison

In this project, we are predicting the profile rating of various GitHub users. So, we can identify from the dataset that category is our target variable which indicates whether the profile is rated low, average or good. For analysing the same, we are considering the below algorithms:

- Multiclass Decision Jungle
- Multiclass Neural Network
- Multiclass Logistic Regression

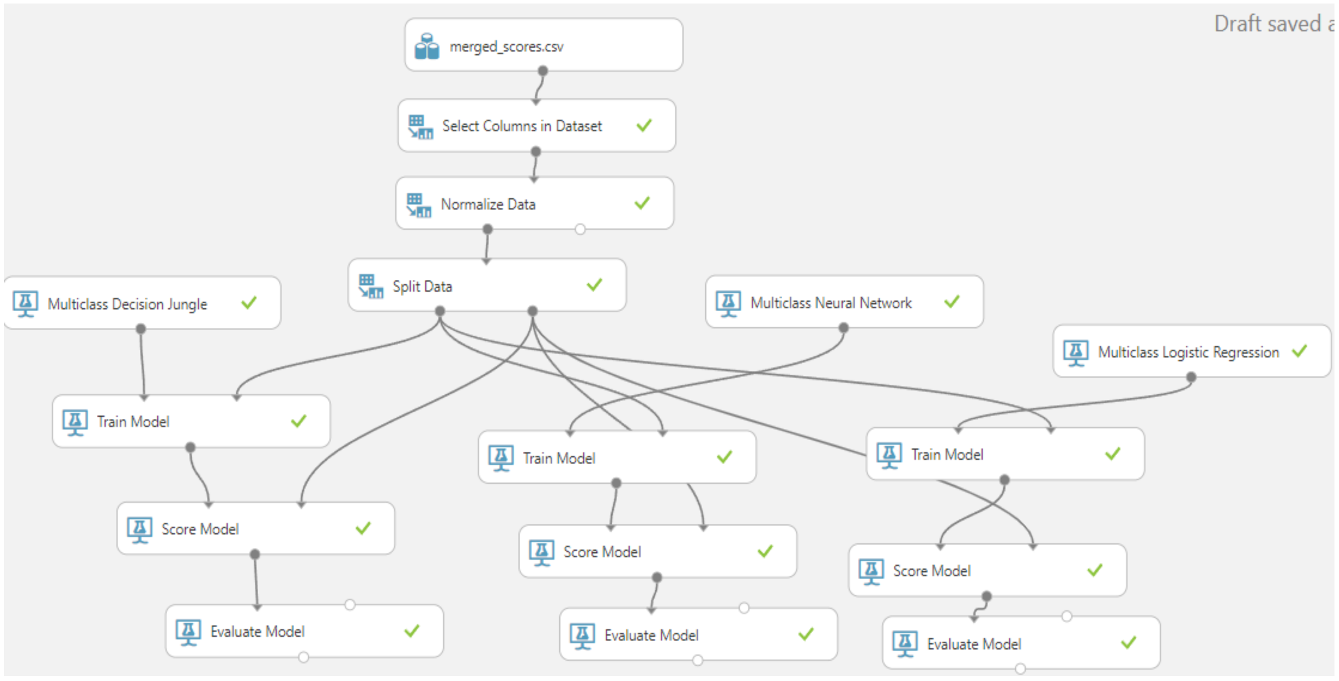


Figure 2. Machine Learning Model

Starting with the Multiclass Decision Jungle, which is a supervised learning algorithm, when running with default parameters, we achieved an accuracy of 73.7% with a recall value of 0.47 and precision value of 0.47. We then tried to tune the default parameters by selecting the resampling method as bagging with a single parameter mode, Number of decisions DAGs as 25, Maximum depth of the DAGs as 60 and number of optimisation steps per decision DAG layer as 2200. This gave an accuracy of 76.4% with a recall value of 0.50 and precision value of 0.50.

Next, Multiclass Neural Network was selected with the default parameters as number of hidden nodes as 100, learning rate as 0.1, hidden layer set to fully connected, number of learning iterations as 100. This gave an accuracy of 71% with recall and precision values of 0.42. Tweaking the number of hidden nodes and number of learning iterations to 200 and 300 respectively gave an accuracy of 72% with a recall and precision values of 0.44, which shows that there's no significant change.

After that, we trained the data on Multiclass logistic regression model with default parameters, resulting an accuracy of 70.7% with recall and precision values of 0.41. Tweaking the model by giving more weightage on L2 regularization did not improve the accuracy significantly.

The Model created can be found in Figure 2.

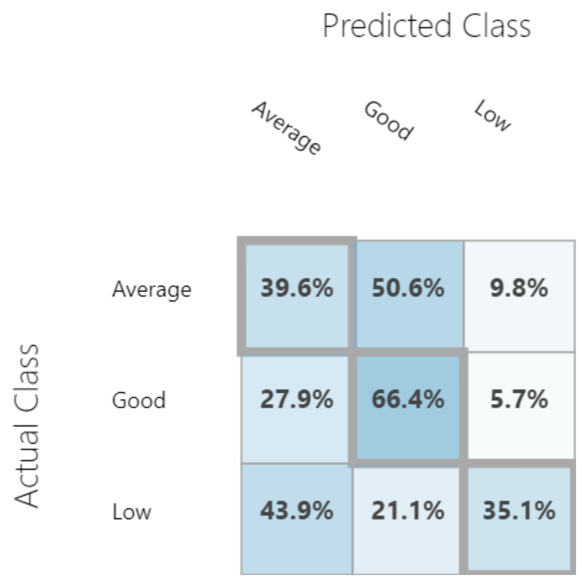
Results

Summing up, we can see that Multiclass Decision Jungle has performed better with an accuracy of 76.5%. Considering the problem, we are dealing with, for identifying the best algorithm, we have considered accuracy, precision and performance matrix as the important features. Therefore, based on the observations and interpretation, we can conclude that Decision Jungle model can be used to predict the profile category.

The following are the results of modelling.

Multiclass Decision Jungle

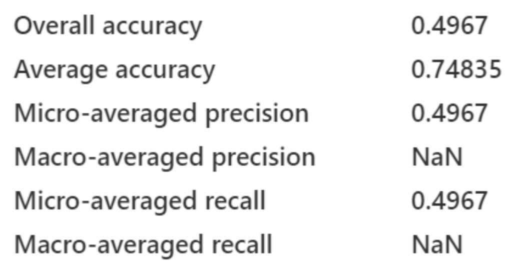
Confusion Matrix



Actual Class \ Predicted Class	Average	Good	Low
Average	39.6%	50.6%	9.8%
Good	27.9%	66.4%	5.7%
Low	43.9%	21.1%	35.1%

Figure 3. Confusion Matrix for Multiclass Decision Jungle Model

Metrics

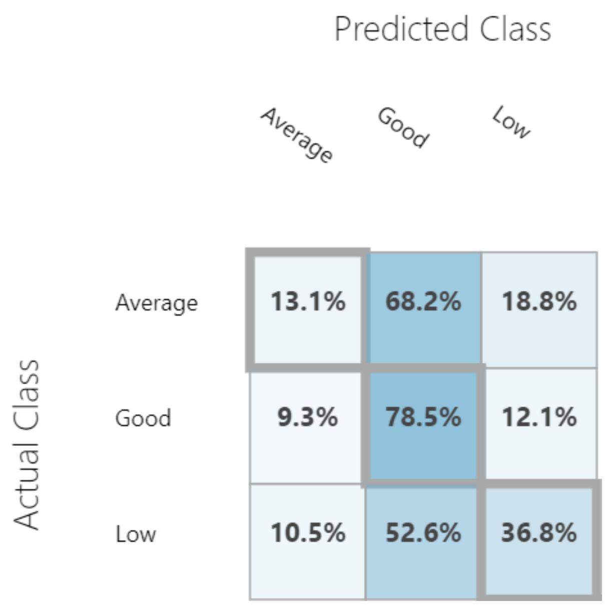


Overall accuracy	0.4967
Average accuracy	0.74835
Micro-averaged precision	0.4967
Macro-averaged precision	NaN
Micro-averaged recall	0.4967
Macro-averaged recall	NaN

Figure 4. Metrics of Multiclass Decision Jungle Model

Multiclass Neural Network

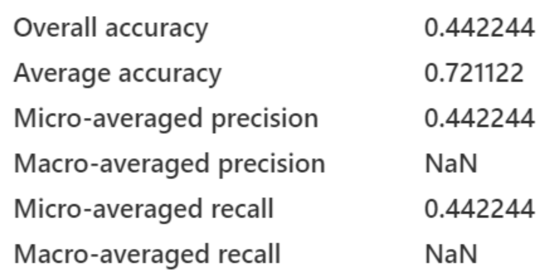
Confusion Matrix



Actual Class \ Predicted Class	Average	Good	Low
Average	13.1%	68.2%	18.8%
Good	9.3%	78.5%	12.1%
Low	10.5%	52.6%	36.8%

Figure 5. Confusion Matrix for Multiclass Neural Network Model

Metrics

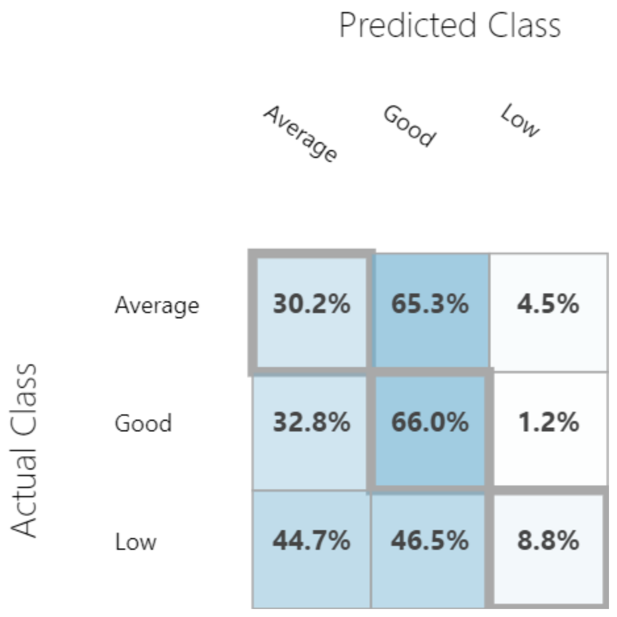


Overall accuracy	0.442244
Average accuracy	0.721122
Micro-averaged precision	0.442244
Macro-averaged precision	NaN
Micro-averaged recall	0.442244
Macro-averaged recall	NaN

Figure 6. Metrics of Multiclass Neural Network Model

Multiclass Logistic Regression Classification

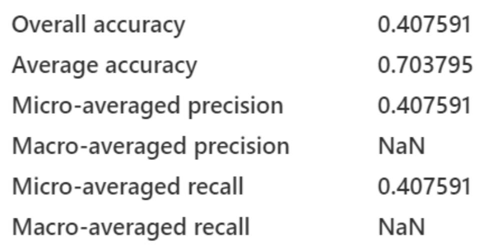
Confusion Matrix



Actual Class \ Predicted Class	Average	Good	Low
Average	30.2%	65.3%	4.5%
Good	32.8%	66.0%	1.2%
Low	44.7%	46.5%	8.8%

Figure 7. Confusion Matrix for Multiclass Logistic Regression Classification Model

Metrics



Overall accuracy	0.407591
Average accuracy	0.703795
Micro-averaged precision	0.407591
Macro-averaged precision	NaN
Micro-averaged recall	0.407591
Macro-averaged recall	NaN

Figure 8 Metrics of Multiclass Logistic Regression Classification Model

Future works

The scoring that we provided could be done in many ways as well. For example, a rater could possibly the languages used, instead of just considering the number of languages used. In practice, this scoring system can differ for different companies or teams. For example, if a data engineering team is rating a user, the team can give a completely different rating based on the skills that they are looking for. This is because the skills required for each team are different. In other words, the rating would capture how well a user would perform in a specific team and not how good they are at coding in general. The features we have used so far are just associated with coding ability. Making higher number of contributions doesn’t make one a good coder. It is just that good coders tend to make a lot of contributions. This model can be improved by creating features from the actual code written. For example, we could identify if a user has created test suite, used a consistent naming convention, or has sufficiently commented their code.

Cost-Benefit Analysis

This entire process of analysing the GitHub profiles while hiring people, specifically in the technical field is being carried out manually by either Engineers in the team or Hiring managers, which means it involves more time and resources. With this project, as the process is being automated, the time and resource utilization can be cut down heavily.

Reference

Yan Hu, Jun Zhang, Xiaomei Bai, Shuo Yu and Zhuo Yang, “Influence analysis of Github repositories”, *Springer Plus*