Noah Hanka

Swapna Gokhale

Honors Project Final Report

Github Popularity Predicting

Introduction:

The premise of this project and research was to determine if the popularity of a github repository, which can be measured through three parameters, number of watchers, number of gazers, and number of forks, is able to be numerically predicted using other known statistics of a repository. These other statistics of the repository include all of the following: commits, branches, releases, contributors, size, totalIssues, openIssues, totalPullRequests, openPullRequests. To determine this, a sample of github data from JavaScript based repositories was analyzed. This data had statistics of over 58,000 repositories, which should be plenty to detect a pattern on.

Data Processing and Overview:

The data was processed through Python, using the pandas library, and then the metrics were graphs to get a sense of their ranges, and how useful the data would be. An example is shown below.

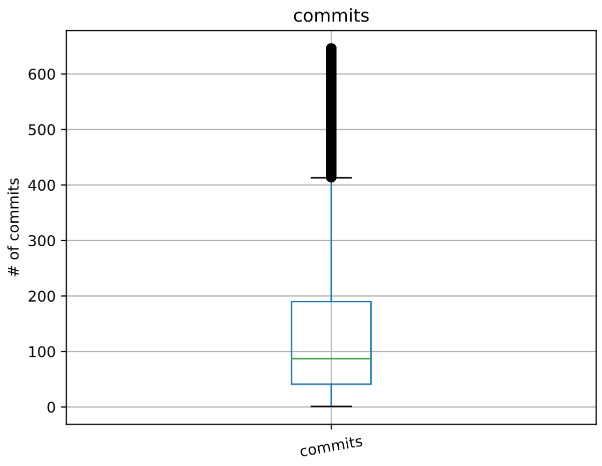


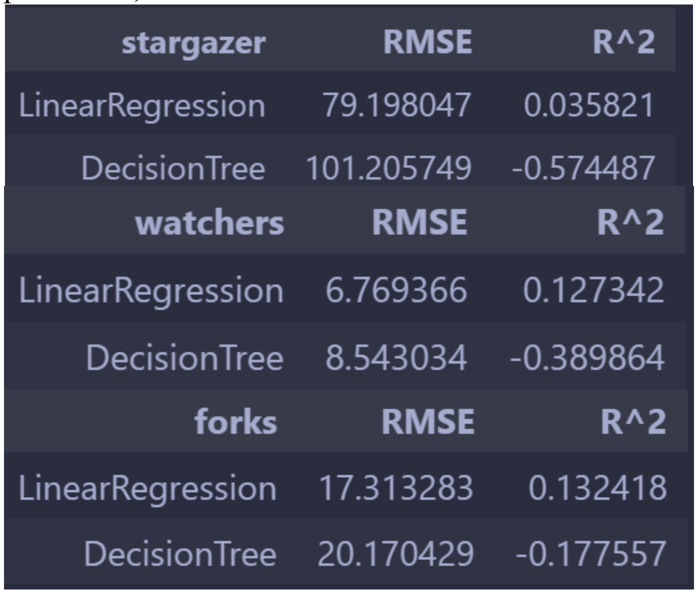
Fig. 1 - Boxplot showing the number of commits

Unfortunately, as seen in the boxplot above, and for many of the other metrics, the data was very scattered and there were lots of outliers. As a result, this would be impactful on the accuracy of the model predictions.

Machine Learning Model and Results:

In this dataset, the dependent variables we are trying to predict are continuous numeric values, so both linear regression and decision tree models were created to see if one model could be more efficient at predicting the other. In order to ensure an accurate testing result, a little bit of preprocessing was done. First, the data was split into a 50/50 training test split. The training set is what the models would be trained on, and the test split to verify results. This helps ensure the models were not overfitted to a specific set of data. Furthermore, for each of the popularity metrics, when creating a model on them, outliers (any data points stretching outside the first and third quartile by 1.5 times the interquartile range, as is the most common method of determining outliers) were ignored.

Here are the results in terms of the R^2 score, and the root mean squared error, which is the average error of each prediction, for each of the metrics.



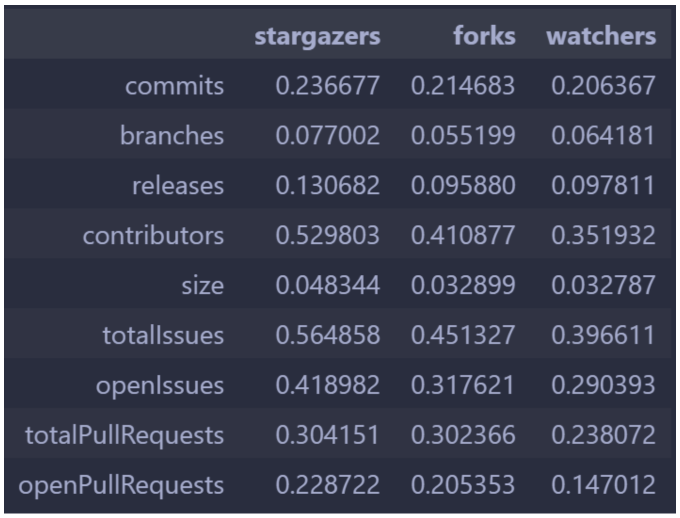
stargazer	RMSE	R^2
LinearRegression	79.198047	0.035821
DecisionTree	101.205749	-0.574487

watchers	RMSE	R^2
LinearRegression	6.769366	0.127342
DecisionTree	8.543034	-0.389864

forks	RMSE	R^2
LinearRegression	17.313283	0.132418
DecisionTree	20.170429	-0.177557

Fig. 2 - The RMSE and R^2 score for the models ran on the test data sets.

To get an understanding, the R^2 score is a percent measurement showcasing the correlation between the predictions, and the actual results, and unfortunately for both models there was not a very strong correlation that could be found. Despite this, it is still interesting to see that some correlation is prevalent, especially in the Linear Regression models, for predicting watchers and forks. There is an R^2 present of around 13%, which although is not great, is a sign of some form of correlation. Another thing we can look at is although the predictions were not great, which metrics were the most correlated to the popularity metrics. We can view this through a correlation matrix, which showcases through a percentage how strongly related two variables are. Here it is:



	stargazers	forks	watchers
commits	0.236677	0.214683	0.206367
branches	0.077002	0.055199	0.064181
releases	0.130682	0.095880	0.097811
contributors	0.529803	0.410877	0.351932
size	0.048344	0.032899	0.032787
totalIssues	0.564858	0.451327	0.396611
openIssues	0.418982	0.317621	0.290393
totalPullRequests	0.304151	0.302366	0.238072
openPullRequests	0.228722	0.205353	0.147012

Fig 3. - The correlation matrix between the repository metrics, and the popularity variables.

As shown in figure 3, the best factors for considering popularity were the number of contributors, total number of issues, and the total number of pull requests. Something that was surprising is that the number of commits was found to have a much lower correlation, and the number of branches was found to have almost no correlation at all.

Overall, despite not being able to train a model to accurately predict the popularity of a github repository, some interesting insight was found into how all of these metrics relate, and how they interact with one another. Perhaps in a follow up, other independent variables could be observed to how they impact popularity, and a larger variety of models could be trained to see if one could produce a more accurate fit.

If you would like to see more on this project, ironically it is hosted on GitHub. You can see if for yourself here: https://github.com/noah-hanka/ghML.git