Chicken Dinners
Predicting the outcome for PUBG games as part of an international evaluation
---
Talaal Mirza - 8166986
December 12, 2018

1.0 Context

1.1 PUBG?

PUBG (or PlayerUnknown's Battlegrounds) is a battle royale game in which ~100 players participate in an all out war against each other. In a round of PUBG these players have to jump out of a plane onto an island littered with weapons, armor, vehicles, medications, and other tools they will need to use to come out on top. The island itself can consist of mountains, cities, farms, villages, rivers, etc. and will be prone to random bombardments of artillery, all the while the playable area compresses forcing players into intense firefights and requiring them to make risky decisions. In the end of the game you are given a ranking of 100 to 1 presenting how successful you were in staying alive (100^{th} place being last).

1.2 Link to course content

The evaluation this report's work took part in asked the participants to accurately predict the rankings of 1 934 174 players in a round of PUBG. Competition participants would be required to test their skills of feature engineering, data analysis and overall ability to create an accurate prediction model. With this specific competition being a "playground" competition, there was a emphasis on learning/improving these skills and sharing ones findings with others.

2.0 Set-up

2.2 Datasets

The participants are provided two data sets, one for training and the other for testing. In both the training and testing data users are provided 28 features they can use to predict the outcome of a player's game with the training data also including the actual players' outcomes *[winPlacePerc]* (a decimal number where 1 represents 1^{st} place and 0 represents last place). As discussed earlier there are 1 934 174 player match records in the testing dataset and 4 446 966 player match records in the training set.

The features representing unique players and games include a player's ID *[Id]*, their group's ID *[groupId]* (represents the group players joined the game in) and the match's ID *[matchId]*. While these features can help identify players, their outcomes are to be predicted using the other 25 features which include things like how many other players someone kills in the match *[kills]*, how far a person travels in vehicles *[rideDistance]*, how many times they use healing tools *[heals]* and more[1].

[1] A full list of features and their descriptions as well as the datasets themselves is provided on the competition's website at: https://www.kaggle.com/c/pubg-finish-placement-prediction/data

2.1 The Evaluation

Submissions to the evaluation ask the participant to return a csv file containing the player IDs from the test data as well as the players' predicted placements. The website then scores the submission's accuracy using the mean absolute error, the smaller the error the higher you are placed in the competitions ranking.

3.0 Proposed algorithm

The plan was to use a variety of different pre-existing libraries and algorithms to create different models, evaluate their performance, and determine why one would be better than the other. Unfortunately due to time constraints and how long training was taking I settled on using the sci-kit learn library's linear scalar vector regression[2] and multi-layer perceptron regressor[3].

4.0 Final Implementation

The final implementation can be found on github at: github.com/TPineapples/ChickenDinners. NOTE that due to github's file size limit, the datasets will need to be downloaded from Kaggle[4] and fixed (see second paragraph). In order to run the implementation (assuming the data found in the *data* directory is available) simply run the *root.py* script found in */ChickenDinners/src/*.

In the event the datasets need to be redownloaded, they should be downloaded as *train.csv*, and *test.csv*, and must be placed in the *./data/train/* and *./data/test/* directories specifically. Once downloaded the *matchType_fix.py* script (found in */ChickenDinners/src/*) will need to be ran in order to ensure that the models will be able to read and fit the data, once its finished it should look something like Figure 1 (shown right). NOTE the */ChickenDinners/data/original/* directory isn't necessary for running the code.

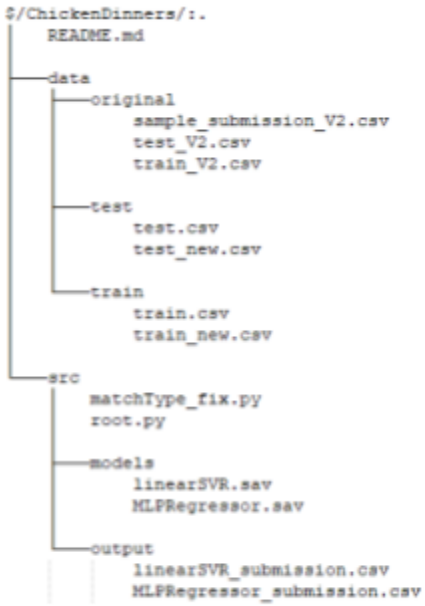

Figure 2: File structure for ChickenDinners directory

It is also important to note that the code runs on Python 3.7.1 and requires pandas 0.23.4 and scikit_learn 0.20.1. Any other versions may not allow the code to run properly.

---

[2] sklearn's LinearSVR - https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html
[3] sklearn's MLPRegressor - https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
[4] Kaggle Datasets - https://www.kaggle.com/c/pubg-finish-placement-prediction/data

5.0 Results

Before I started working on how I would go about implementing my model I determined figured that it would important to see which features would affect the players' outcomes the most. To do so I used the pandas library[5] , and to apply the correlation visually I used the seaborn library's heatmap function[6] as shown in Figure 2.

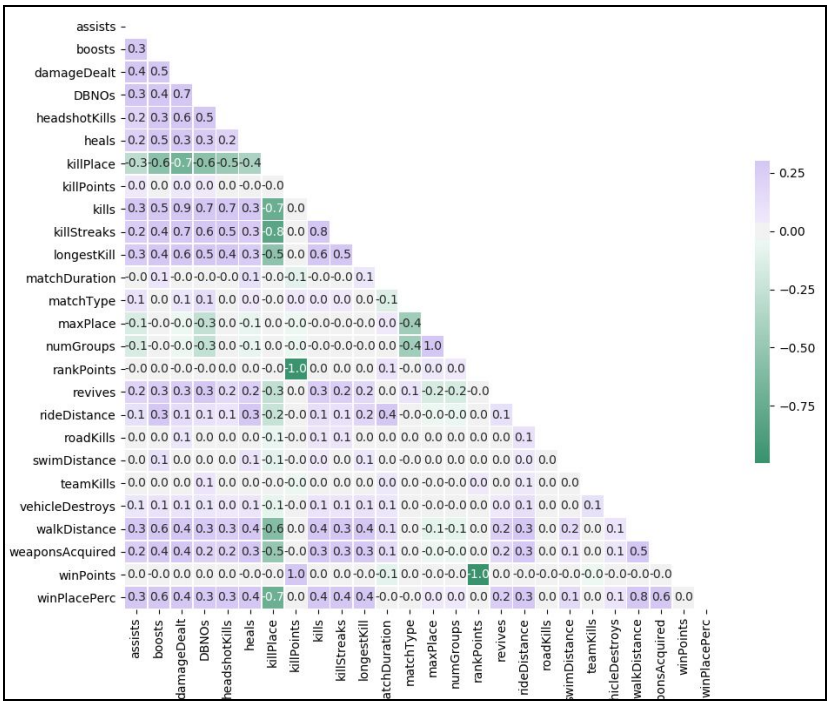

Figure 2: Pearson correlations for training data

Figure 1 shows the pearson correlations for training data, showing how "relationships between pairs of continuous variables" are linearly correlated[7]. When looking at the bottom row of the figure we can see which features affect the *[winPlacePerc]*, the most. More specifically, we can see that *[killPlace]*, strongly-negatively correlates with *[winPlacePerc]*, while *[walkDistance]*, *[weaponsAcquired]*, and *[boosts]*, strongly-positively correlates with *[winPlacePerc]*. We should expect a good model would take these features into most consideration when predicting.

And while the developed models do show the same correlation characteristics they still fall short. When uploaded to the competition and submitted the linearSVR presented a mean absolute error of 0.1181 while the MLPRegressor had a mean absolute error of 0.0756, placing me at the time of uploading the submission, at 552nd place.

[5] Pandas - https://pandas.pydata.org/
[6] Seaborn - https://seaborn.pydata.org/index.html
[7] Kent State University: SPSS Tutorials - https://libguides.library.kent.edu/SPSS/PearsonCorr

6.0 Conclusion

While these models aren't the best they are pointing in the right direction. Some strategies that may help them predict better would be to divide the data into categories and train separate models catering to them. I would also try using different libraries like tensorflow or LightGBM.

References

[1] Kaggle: PUBG Finish Placement Prediction
https://www.kaggle.com/c/pubg-finish-placement-prediction
[2] sklearn's LinearSVR
https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html
[3] sklearn's MLPRegressor
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
[4] Kaggle Datasets https://www.kaggle.com/c/pubg-finish-placement-prediction/data
[5] Pandas https://pandas.pydata.org/
[6] Seaborn https://seaborn.pydata.org/index.html
[7] Kent State University: SPSS Tutorials https://libguides.library.kent.edu/SPSS/PearsonCorr