Learn to predict win placement of PUBG

Stella Liu

06/28/2019

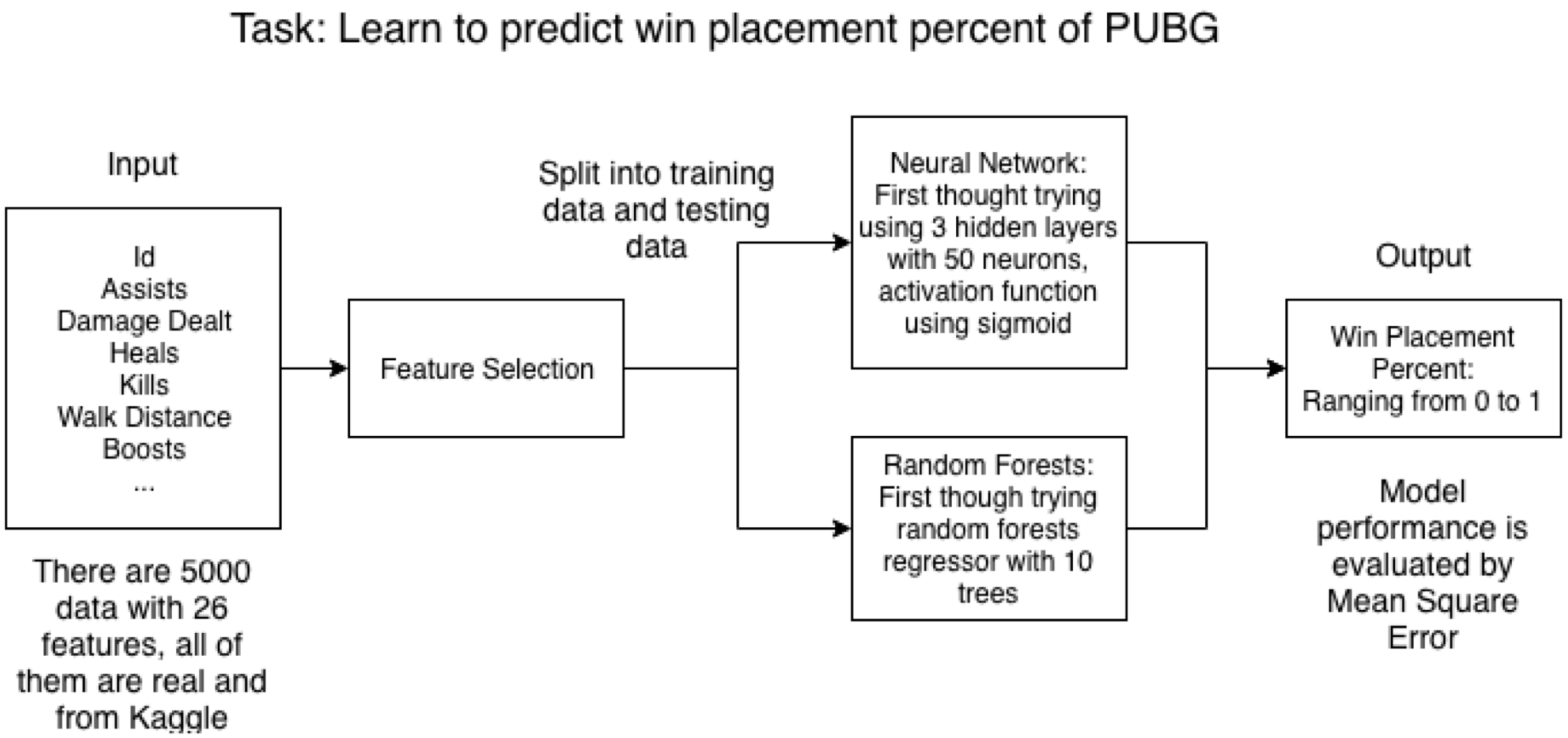
Problem Statement

‘PUBG’ Gets New 10-Man War Event Mode This Weekend
Credit: variety

PlayerUnknown's Battlegrounds(PUBG) is a battle royale game. 100 players are dropped onto an island empty-handed and must explore, scavenge, and eliminate other players until only one is left standing, all while the play zone continues to shrink.

In the game strategic gameplay is as important as shooting skills. What's the best strategy to win in PUBG? Should you sit in one spot and hide your way into victory, or do you need to be the top shot?

AI Approach


Task: Learn to predict win placement percent of PUBG
Input
Id
Assists
Damage Dealt
Heals
Kills
Walk Distance
Boosts
...
There are 5000 data with 26 features, all of them are real and from Kaggle
Feature Selection
Split into training data and testing data
Neural Network: First thought trying using 3 hidden layers with 50 neurons, activation function using sigmoid
Random Forests: First though trying random forests regressor with 10 trees
Output
Win Placement Percent: Ranging from 0 to 1
Model performance is evaluated by Mean Square Error

Implementation

Language: Python. There are sufficient machine learning libraries like sklearn, keras and I am familiar with it.

Dataset: Getting data from Kaggle. All of them are real and provided by PUBG official API. There are 4.36 million tuples and I will use 5000 most recent data for first attempt. It has 28 features and only some of them are related to target feature(win placement), so I need feature selection to eliminate meaningless features.

Toolkits: For RF, I will use sklearn.ensemble.RandomForestRegressor. For NN, I will use Keras to build neural network.

Evaluation

The model performance is evaluated on Mean Square Error between predicted win placement and the observed data in the dataset.

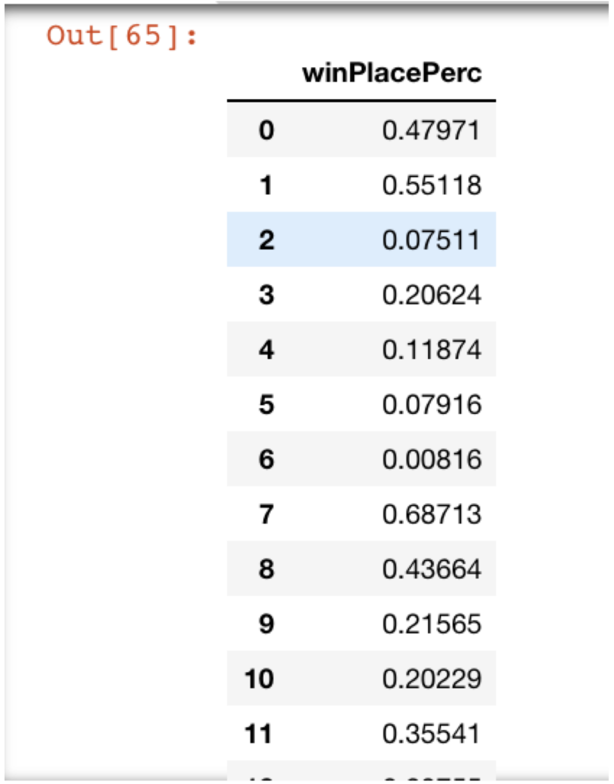
The expected MSE should be at most 10%, since I use 1000 examples for testing, it should guarantee that models get precision better than 90%.

Evaluation Results

In this experiment, I split the data set into two parts: training data(4000, 80%) and testing data(1000, 20%). During EDA and feature engineering, 5 main features are used for tuning Random Forest and NN: walkDistance, boostAndheals, weaponsAcquired, damageDealt and killPlace

Random Forest:
For these 1000 testing data, the MSE is 0.01430938005544915.

Neural Network:
Without batch normalization, the MSE is 0.3223
After applying BN layer, the MSE is 0.02727401074800811.



Out[65]:

	winPlacePerc
0	0.47971
1	0.55118
2	0.07511
3	0.20624
4	0.11874
5	0.07916
6	0.00816
7	0.68713
8	0.43664
9	0.21565
10	0.20229
11	0.35541

Discussion of Results

The result is better than I expected, both method get prediction with error less than 0.03.

At first, I used the architecture from proposal: model was built in sequential mode, five neurons for the first layer, input shape is (5,). The next three layers are all hidden layers, each with 50 neurons. All the layers are using RELU as activation function. The model is compiled by mean square error, sgd for optimizer. However, the training process is always unsatisfied, the fitting loss is 0.3223 and the testing error is about 0.33. No matter how I changed the hidden layers, neurons, activation function, the result did not change much. After doing more experiments, I found that it was due to overfitting. After applying dropout and batch normalization, the performance got much better.

For future work, I will try more data, there are 4.36 million data from Kaggle. If I can use all of the data, the result might be more real and reasonable. Also I can try gradient boosting on those data. However, as the size increases, time cost and memory cost will become a big problem. I need to find out how to reduce time and space consuming.

Conclusions

The task is to predict win placement in PUBG based on players’ gaming data. Random forest and Neural network both predict successfully, they all have error less than 0.03.

From this project, I found that Random Forest is more suitable for less data than Neural Network, since there might be overfitting in NN and it is easier to train RF.
Since I only applied 5 features to train the model, it is uncertain that whether RF performs better than NN.
When building NN, especially for fewer examples and features, overfitting problem must be considered. Both randomly droping neurons connection and normalizing batch activation values can help avoid overfitting, while BN have more benefits. It can accelerate convergence, reduce network insensitivity to initialization weights and allow large learning rate.

References

[1]. Artificial Intelligence 2E Foundations of Computational Agents, Chapter 7.2-6
[2]. Gradient Boosting VS Random Forests,
URL: https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80
[3]. The privacy pro's guide to explainability in machine learning,
URL: https://iapp.org/news/a/the-privacy-pros-guide-to-explainability-in-machine-learning/
[4]. Gradient Boosting vs Random forest,
URL: https://stackoverflow.com/questions/46190046/gradient-boosting-vs-random-forest
[5]. 3.2.4.3.2. sklearn.ensemble.RandomForestRegressor,
URL:http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
[6]. Decision Trees and Random Forests,
URL: https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991
[7]. Wikipedia: Random Forest, Gradient Boosting, Neural Networks, Decision Tree, PlayerUnknown's Battlegrounds
[8]. PUBG Finish Placement Prediction,
URL: https://www.kaggle.com/c/pubg-finish-placement-prediction
[9]. Data Analytics Made Accessible Chapter 4, 6, 8

Thanks very much for viewing!