Machine Learning with Networks *ECEN765.600, Fall 2018*

Midterm Report

Name: Di Tian

Claim: The original project idea —— "Food recognition algorithm" , has been replaced by a new project which comes from Kaggle Competition community. The reason for this change is because the original one is impractical and a little boring! I will introduce the new idea in this report.

● Introduction

Topic: PUBG Finish Placement Prediction

"*So, where we droppin' boys and girls?*

Battle Royale-style video games have taken the world by storm. 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.

PlayerUnknown's BattleGrounds (PUBG) has enjoyed massive popularity. With over 50 million copies sold, it's the fifth best selling game of all time, and has millions of active monthly players."

—— Kaggle Competition Community

In this project, competitors are given over 65,000 games' worth of anonymized player data, split into training and testing sets, and asked to predict final placement from final in-game stats and initial player ratings.

By exploring and modeling the given dataset, challengers are able to provide some very important insights about the best strategy to win the game. This is crucial to game development because a good understanding about the game itself helps developers to keep the game playable and balance.

Data

In a PUBG game, up to 100 players start in each match (matchId). Players can be on teams (groupId) which get ranked at the end of the game (winPlacePerc) based on how many other teams are still alive when they are eliminated. In game, players can pick up different munitions, revive downed-but-not-out (knocked) teammates, drive vehicles, swim, run, shoot, and experience all of the consequences -- such as falling too far or running themselves over and eliminating themselves.

Challengers are provided with a large number of anonymized PUBG game stats, formatted so that each row contains one player's post-game stats. The data comes from matches of all types: solos, duos, squads, and custom; there is no guarantee of there being 100 players per match, nor at most 4 player per group.

Challengers must create a model which predicts players' finishing placement based on their final stats, on a scale from 1 (first place) to 0 (last place).

More details about the data can be found in appendix.

Personal Goal

- Look the daily life from the perspectives of data scientist and machine learning engineer.
- Go through the entire machine learning workflow.
- Grasp some hands-on experience of machine learning project development. Such as using existing libraries (pandas, sklearn, seaborn), programming language(python) and consolidating the knowledge I learned from classes.
- Win a good placement!

Progress

Until now, the main progresses including:

① **Frame the learning problem and look at the big picture.**

Settled the learning tasks, learning type, evaluation metric and some matters need attention.

② **Data exploration.**

To gain some intuitions about this project, I visualized the given dataset by techniques including: target distribution, correlation matrix, correlation heatmap, feature histogram, feature scatters against target.

③ **Data pre-processing.**

To feed the models with best quality data, I did data pre-processing: data cleaning, feature selection, feature-engineering, standardization, normalization.

④ **Model comparison.**

To choose the best fit model, I used 10-fold cross-validation to compared the baseline of three models. They are linear regression, multi-layer perceptron and random forest regression. Result shows that random forest outperforms the other two.

⑤ **Prototyping.**

After determined which model to use, I started developing the prototype. In this process, I found out that by clustering the data and do some careful feature engineering before training the models, the final performance can be improved. Until now, the best score is about 0.049. The best score of this competition is about 0.0194. However, consider that I only used a small fraction of the dataset to train the model, there's still a lot of room for my model to improve.

Some details about my progress are described as following.

Overview

Learning task:

Given some game data, predict the **final placement** of each player. on a scale from 1 (first place) to 0 (last place). The final score will be evaluated by **mean absolute value** between target and prediction.

Learning type:

1. The goal is to predict the final placements, hence, supervised learning must be involved.
2. The given dataset includes 28 features and millions of records. Therefore, to handle the problem quickly and efficiently, we may need some unsupervised learning techniques (dimensionality reduction and visualization) to simplify the model.

Others:

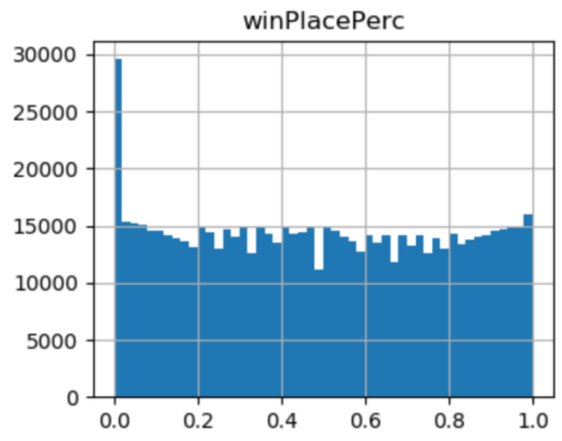
1. The given dataset includes more than millions of records and take up more than 800Mb space. Hence, some manipulation on the original dataset, such as sampling, compressing are needed.

Data exploration

This is a large dataset with 28 features, so it is hard to visualize the whole picture. So I decided to start with the target value to get some insights about this problem.

➢ **Target —— "winPlacePerc"**

I first plot the histogram of "winPlacePerc" using pandas.hist function.



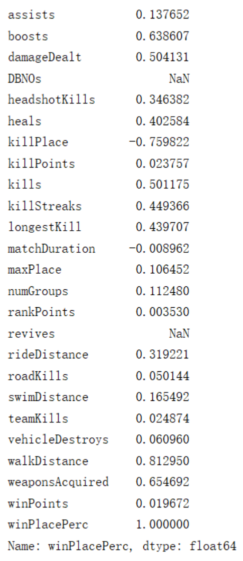
The overall distribution is a normal distribution as expected except that the number of 0 is as twice as others. I then reviewed the data description and found out that some game data is missed, so the placement is scaled in those existing records. Hence, we can always get a last placement in every game.

This distribution tells me that my model should be able to handle these 0 placements efficiently.

- **Feature correlation**

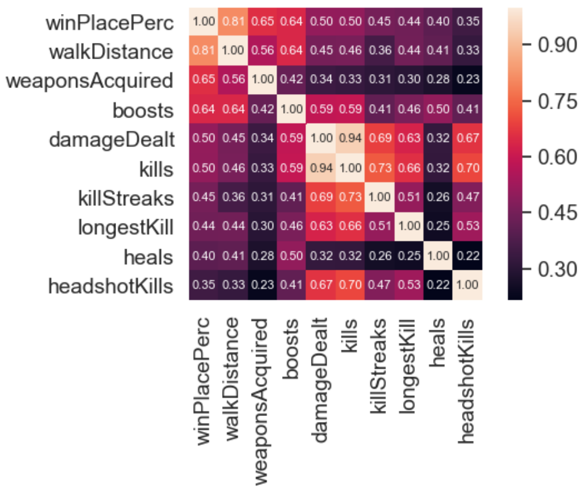
The next thing I did is to calculate the correlation matrix since it can provide important information to us about which feature is more important.

Then I printed the correlation matrix as following:



```
assists            0.137652
boosts             0.638607
damageDealt        0.504131
DBNOs                   NaN
headshotKills      0.346382
heals              0.402584
killPlace         -0.759822
killPoints         0.023757
kills              0.501175
killStreaks        0.449366
longestKill        0.439707
matchDuration     -0.008962
maxPlace           0.106452
numGroups          0.112480
rankPoints         0.003530
revives                 NaN
rideDistance       0.319221
roadKills          0.050144
swimDistance       0.165492
teamKills          0.024874
vehicleDestroys    0.060960
walkDistance       0.812950
weaponsAcquired    0.654692
winPoints          0.019672
winPlacePerc       1.000000
Name: winPlacePerc, dtype: float64
```

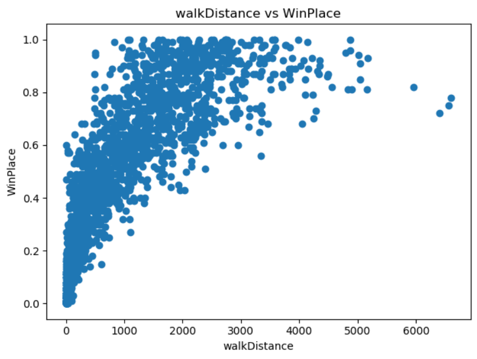
The table is a not very intuitive. Let's look at the heat map:



From this picture, we can clearly see that the ten most important features in predicting the final placement.

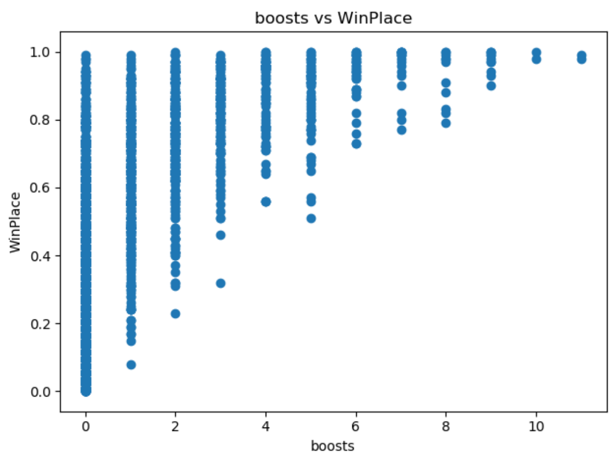
➢ Feature scatters

Then I grabbed a feature and plotted the scatter image against target value to get some intuition about the underlying distribution.



From this picture, we can see that "walkDistance" is positively correlated to "winPlacePerc". And the distribution appears linearity to some extent. **So linear regression model is a potential candidate for this problem.**

Let's take a look at another feature:



This is the scatter image of feature “boosts” against target. We can see that “boosts” is a discrete feature. In fact, in this problem, there are many other discrete features. **From this point, I decided to also try decision tree regressor because it is good at dealing with discrete points.**

The two features I showed above are represents of two major kinds of features. I then looked at the histograms and scatters of all other features, which are included in appendix.

Data pre-processing

So far, two potential models are selected. They are liner regression and decision tree regressor. Although decision tree regression doesn’t require too much data pre-processing steps, we need to do some careful preparation to assure the data feed to linear regression and other potential models are appropriate.

Data cleaning

The first step is data cleaning. By look at the data, I found out that there are some missing values in target, so I get rid of the corresponding records.

Text and categorical features

In this dataset, the text and categorical features including “Id”, “matchId”, “matchType” and “gourpId”. I then realized that this dataset include data from several game mode: solo, duo, squad and custom. **Considering the different strategies players may use in different game modes, I decided to partition the dataset by “matchType” and first focus on the particular mode——solo.** This also simplified the modeling process, which speeds up the whole development cycle.

For solo mode, “Id”, “matchId” and “groupId” is useless. So I simply drop out these features. Also, since we focus on “solo” mode, “matchType” can’t provide any information neither. So it was discarded too.

Feature scaling

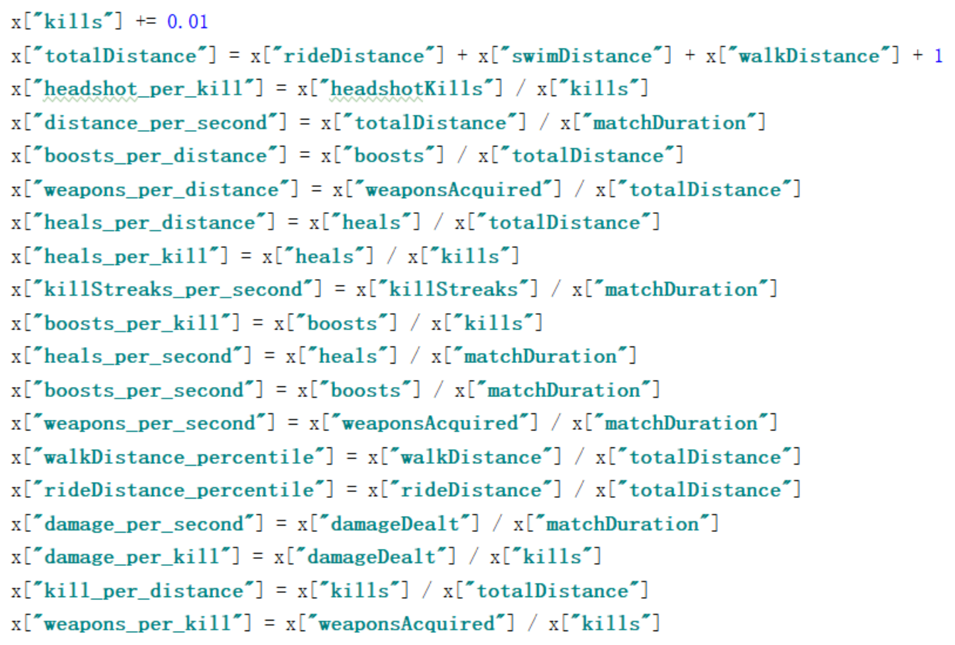
Since different scale range may hurt the performance of some machine learning models, we need to normalize the data. Also, some algorithm requires centered data, so we also need to standardize the data. So the data pre—processing pipeline is:

① Feature-wise standardization

② Feature-wise normalization(l2)

Feature Engineering

To provide more information for our potential models, I created some new features. Which are show below:



```
x["kills"] += 0.01
x["totalDistance"] = x["rideDistance"] + x["swimDistance"] + x["walkDistance"] + 1
x["headshot_per_kill"] = x["headshotKills"] / x["kills"]
x["distance_per_second"] = x["totalDistance"] / x["matchDuration"]
x["boosts_per_distance"] = x["boosts"] / x["totalDistance"]
x["weapons_per_distance"] = x["weaponsAcquired"] / x["totalDistance"]
x["heals_per_distance"] = x["heals"] / x["totalDistance"]
x["heals_per_kill"] = x["heals"] / x["kills"]
x["killStreaks_per_second"] = x["killStreaks"] / x["matchDuration"]
x["boosts_per_kill"] = x["boosts"] / x["kills"]
x["heals_per_second"] = x["heals"] / x["matchDuration"]
x["boosts_per_second"] = x["boosts"] / x["matchDuration"]
x["weapons_per_second"] = x["weaponsAcquired"] / x["matchDuration"]
x["walkDistance_percentile"] = x["walkDistance"] / x["totalDistance"]
x["rideDistance_percentile"] = x["rideDistance"] / x["totalDistance"]
x["damage_per_second"] = x["damageDealt"] / x["matchDuration"]
x["damage_per_kill"] = x["damageDealt"] / x["kills"]
x["kill_per_distance"] = x["kills"] / x["totalDistance"]
x["weapons_per_kill"] = x["weaponsAcquired"] / x["kills"]
```

Note that some features are added with a small number, this is to avoid the “divided by zero” errors.

Model comparison

Having previous knowledge about this dataset, I decided to try three models:

linear regression, multi-layer perceptron and random forest.

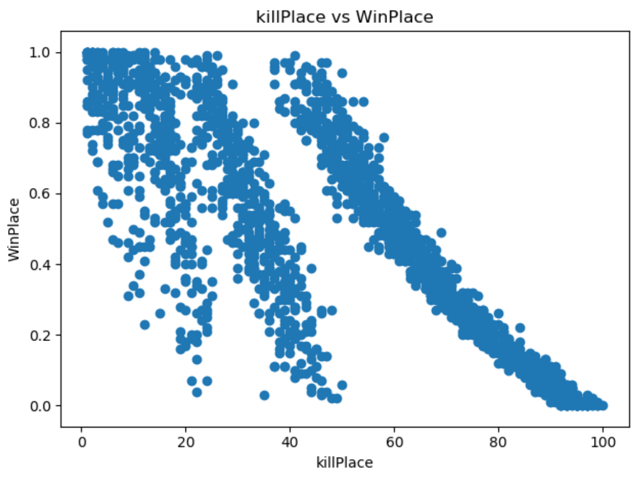
The prediction result is shown in the picture below.

```
The score of linear regression is:  0.06794536449410567 (mean) 0.0011370732389557635 (std)
The score of multi-layer perceptron is:  0.1977994658403675 (mean) 0.07453040403746548 (std)
The score of random forest is:  0.05149098512798025 (mean) 0.0007869520155139286 (std)
```

We can see that random forest is the best fit for this problem. Its mean absolute error reached about 0.0514 which means that the average placement prediction error is about 5.

Prototype

The difference between the prototype and the experimental model is that I trained different models to make prediction for different data. The inspiration comes from the scatter plot of feature "killPlace" as shown below:



We can see that when "killPlace" is greater than 50, the final place is almost linearly correlated with "killPlace". However, at the part where "killPlace" is greater than 40 but smaller than 50, we can see that the same "killPlace" can have totally different placement. **I then made an assumption that different players may use completely different strategies to play the game.** For example, some players like to fight with enemies very often which gives them a better "killPlace" but also increases the likelihood of being killed. Other players, especially those experienced group tend to avoid unnecessary conflicts but only combat their enemies at vital moment. Hence, you can see a player with 60 "killPlace" wining the game as well as another lose the game with 40 "killPlace".

Then I decided to train a k-means model to discover this underlying pattern. And the result shows that after clustering the data and feed different class to different models, the result is improved.

```
The score of random forest is 0.04972216439728906 (mean) 0.0004583463373594644 (std)
```

Existing challenges

1. Until now, all experimental results are obtained from a single game mode "solo". **Whether the conclusion can be applied to other modes is still a question.**
2. Although we have found that there are some underlying differences between the playing styles. I**t still is a challenge to better reflect these differences (the clustering doesn't improve the result tremendously).**
3. Although we didn't feed all data to train the model which means that the model still has room to improve, the result of our model is much lower than the best score in this competition. **We need to explore more about the problem and the dataset to figure out whether there are better models or better feature engineering than the current one.**

Next stage

To address the challenges above, I will do the following things:

1. Increase the volume of the train set.
2. Apply the model we get from "solo" mode on other mode. See the result and decide if there is needs to develop different model to fit different modes.
3. Try more models or combination of different models. At the time when I wrote this report, I found out that if we combine k-means clustering and linear regression, the result is very close to the result of random forest! This implies that maybe some novel combination of different models can surprise us.
4. Dive into the dataset deeper to gain more insights about the features. This could help us establish a better feature engineering.