**PUBG Players Finishing Placement Prediction**

Description

Create a model that predicts players’ finishing placement based on their final stats, on a scale of 1 (first place) to 0 (last place).

**Problem Statement**:

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.

You are provided with a large number of anonymous PUBG game stats, formatted so that each row contains one player’s post-game stats. The data comes from matches of all types, i.e., solos, duos, squads, and custom. There is no guarantee of having 100 players per match, or at the most 4 players per group. In the game, a player can find and use a variety of weapons, items, and vehicles. Players must also avoid the shrinking “circle”, which pushes players closer together on the map.

**Domain**: Gaming

**Analysis to be done**: Perform data preprocessing, build a deep learning prediction model.

**Content**:

**Dataset columns and definition**:

* **DBNOs** - Number of enemy players knocked.
* **assists** - Number of enemy players this player damaged that were killed by teammates.
* **boosts**- Number of boost items used.
* **damageDealt**- Total damage dealt. Note: Self-inflicted damage is subtracted.
* **headshotKills** - Number of enemy players killed with headshots.
* **heals**- Number of healing items used.
* **Id**- Player’s Id
* **killPlace**- Ranking in match of number of enemy players killed.
* **killPoints** - Kills-based external ranking of player. (Think of this as an Elo ranking where only kills matter.) If there is a value other than -1 in rank points, then any 0 in kill points should be treated as a “None”.
* **killStreaks**- Max number of enemy players killed in a short amount of time.
* **kills** - Number of enemy players killed.
* **longestKill** - Longest distance between player and player killed at the time of death. This may be misleading, as downing a player and driving away may lead to a large longestKill stat.
* **matchDuration**- Duration of match in seconds.
* **matchId** - ID to identify match. There are no matches that are in both the training and testing set.
* **matchType** - String identifying the game mode that the data comes from. The standard modes are “solo”, “duo”, “squad”, “solo-fpp”, “duo-fpp”, and “squad-fpp”; other modes are from events or custom matches.
* **rankPoints**- Elo-like ranking of players. This ranking is inconsistent and is being deprecated in the API’s next version, so use with caution. Value of -1 takes place of “None”.
* **revives**- Number of times this player revived teammates.
* **rideDistance**- Total distance traveled in vehicles measured in meters.
* **roadKills**- Number of kills while in a vehicle.
* **swimDistance** - Total distance traveled by swimming measured in meters.
* **teamKills**- Number of times this player killed a teammate.
* **vehicleDestroys**- Number of vehicles destroyed.
* **walkDistance** - Total distance traveled on foot measured in meters.
* **weaponsAcquired**- Number of weapons picked up.
* **winPoints**- Win-based external ranking of player. (Think of this as an Elo ranking where only winning matters.) If there is a value other than -1 in rankPoints, then any 0 in winPoints should be treated as a “None”.
* **groupId** - ID to identify a group within a match. If the same group of players plays in different matches, they will have a different groupId each time.
* **numGroups** - Number of groups we have data for in the match.
* **maxPlace** - Worst placement we have data for in the match. This may not match with numGroups, as sometimes the data skips over placements.
* **winPlacePerc**- The target of prediction. This is a percentile winning placement, where 1 corresponds to 1st place, and 0 corresponds to last place in the match. It is calculated off of maxPlace, not numGroups, so it is possible to have missing chunks in a match.

**Steps to perform**:

Perform exploration of the data and apply feature engineering on the data. Follow up with a deep learning model to predict players finishing placement.

**Tasks**:

1. Data Exploration:

* Check mean, quantile, and max kills
* Draw inference by plotting kill counts
* Check signs of cheating

1. Feature Engineering:

* Creating new features from existing to depict data in a better way
* Expressing relative rank instead of absolute in the form of percentage
* Dropping redundant features
* Dropping cheaters
* Grouping and aggregating features

1. Model Creation:

* Create a sequential model with Keras using Tensorflow backend
* Take four dense layers, three dropout layers, three batch normalizations, and use PreLu as an activation function