PUBG Game Prediction

PlayerUnknown's Battlegrounds (PUBG) is a popular online multiplayer battle royale game. In a battle royale game, players fight to be the last person or team standing. In order to win a PUBG game, it is important to have a combination of skill, strategy, and luck.

Now to predict the outcome of a game we need to train our model on a large dataset containing all various parameters of a game or information of a player like the guns he used, average headshot rate, his group rank etc. The datasets we are going to use 29 features. These 29 features along with the output of the game will be provided to train the model.

```
0   Id               object
1   groupId          object
2   matchId          object
3   assists          int64
4   boosts           int64
5   damageDealt      float64
6   DBNOs            int64
7   headshotKills    int64
8   heals            int64
9   killPlace        int64
10  killPoints       int64
11  kills            int64
12  killStreaks      int64
13  longestKill      float64
14  matchDuration    int64
15  matchType        object
16  maxPlace         int64
17  numGroups        int64
18  rankPoints       int64
19  revives          int64
20  rideDistance     float64
21  roadKills        int64
22  swimDistance     float64
23  teamKills        int64
24  vehicleDestroys  int64
25  walkDistance     float64
26  weaponsAcquired  int64
27  winPoints        int64
28  winPlacePerc     float64
```

Dataset Information

In the project video each of the feature will be explained and what these features means.

Before creating a CatBoost Model we are going to perform Data Wrangling and Feature Engineering.

Data Wrangling

Data wrangling is the process of cleaning, organizing, and preparing data for analysis. In the context of the PUBG win prediction model, data wrangling would involve several steps to ensure that the data is in a usable form for the machine learning model.

Some of the data wrangling tasks that might be performed include:

Removing any unnecessary or irrelevant columns from the dataset

Handling missing values in the data (e.g. imputing missing values, dropping rows with missing values)

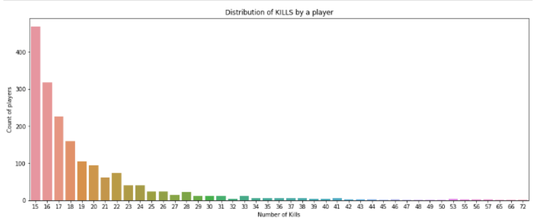
Converting categorical data, such as the type of equipment a player used, into numerical form (e.g. using one-hot encoding)

Splitting the data into training and testing sets

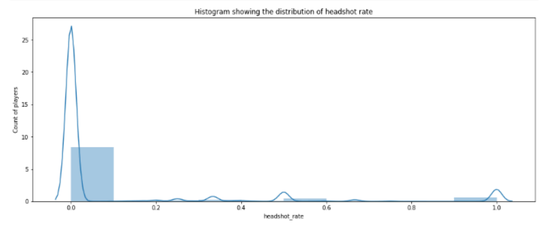
Performing these tasks is important because machine learning algorithms typically expect data to be in a specific format and can be sensitive to missing or incorrect values. By wrangling the data appropriately, we can ensure that the machine learning model is able to learn patterns in the data and make accurate predictions.

Now we can use this data for visualization as well which will help us get some insights about the game.

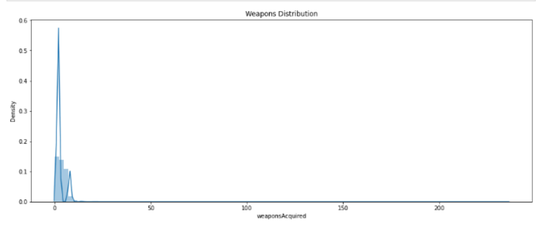
```
sns.countplot(df[df['kills']>=15]['kills']).set_title("Distribution of KILLS
plt.ylabel("Count of players")
plt.xlabel("Number of Kills")
plt.show()
```




```
# plot the headshot rate distribution
sns.distplot(df['headshot_rate'], bins =10).set_title("Histogram showing the
plt.ylabel("Count of players")
plt.show()
```




```
sns.distplot(df['weaponsAcquired'], bins=100).set_title("Weapons Distributio
plt.show()
```



These are just a few graphs that we can plot using the data. We can even perform EDA on this dataset. It all comes down to how creative you can be with the data. But for now, let's focus on our new task i.e., Feature engineering.

Feature engineering

Feature engineering is the process of creating new features or modifying existing features in a dataset in order to improve the performance of a machine learning model. In the context of the PUBG win prediction model, feature engineering might involve creating new features based on existing data or transforming existing features in a way that makes them more useful for the model.

This might include normalizing the features.

Dropping unnecessary features which would only lead to increase in models complexity

Creating features based on combinations of existing features: For example, creating a new feature that represents the total number of kills a player made in a game, by summing up the number of kills made with each type of weapon.

CatBoost Model

Let's first take a look at why we chose the CatBoost Model for this dataset.

CatBoost is effective for working with categorical data: The PUBG game includes a number of categorical features, such as the type of equipment a player used or the location on the map where a player was killed. CatBoost is particularly effective for handling categorical data, as it can automatically encode the categories as numerical values and handle missing values.

CatBoost is fast and easy to use: Training a CatBoost model is typically fast, and the library includes a number of built-in features that make it easy to use, such as automatic handling of missing values and support for parallelization. This can make it a good choice for quickly prototyping and testing models.

CatBoost is a powerful and accurate algorithm: In general, gradient boosting algorithms like CatBoost are known to be powerful and accurate, and they have been successful in a number of machine learning competitions. This makes them a good choice for many types of problems.

Given the characteristics of the PUBG game data and the strengths of the CatBoost algorithm, it is reasonable to consider using CatBoost for this type of problem.

After using CatBoost model for our dataset we can predict the performance using RMSE

Prediction

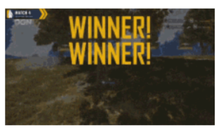
```
pred = model.predict(xtest)
```

```
rmse = np.sqrt(mean_squared_error(ytest, pred))
r2 = r2_score(ytest, pred)

print("Testing performance")

print("RMSE: {:.2f}".format(rmse))
print("R2: {:.2f}".format(r2))
```

```
Testing performance
RMSE: 0.08
R2: 0.93
```



Dataset: https://www.kaggle.com/datasets/ashishjangra27/pubg-games-dataset

Code: https://github.com/AshishJangra27/Machine-Learning-with-Python-GFG/blob/main/PUBG%20Game%20Winning%20Prediction/PUBG%20Game%20Prediction.ipynb