Players Unknown Battlegrounds

Data Science Project

By Ameen Abdelghani

Dataset

Players Unknown Battleground is the 2017 video game of the year that began a whole new genre in video games that is currently dominating the market. It is currently the fifth best selling game of all time. The concept is simple, 100 players, optionally in teams, are dropped off on an island with nothing, they must scavenge the map for weapons and equipment with the map size shrinking to a random location, last man or team standing wins.. This genre is called battle royale, and has influences from the hit movie series *The Hunger Games* and the Japanese film *Battle Royale*.

I found on Kaggle 4 gb of data for Players Unknown Battlegrounds (aka PUBG), showing data for where the player died, how long a player lasted, how far they traveled, how many kills did they perform, and more. The data has no missing values given how the data is generated from a video game.

Problem

As this genre is very new, and has at the same time swept the world through PUBG and other hits like Fortnite, the overall best strategy to win a match is disputed. Unlike many other game concepts, only 1-4% (depending on team size) of players are actually considered winners. A game such as this takes more into consideration than simply shooting skills; strategy, patience, and positioning are crucially.

PUBG Corporation is not a Triple A game developer, they are rather small in scale, and despite the game being a large hit, the game lacks basic features and efficiencies you might expect in such a hit game. Key features such as tips, training modes, and tutorials are nonexistent in the game.

In recent months PUBG has been losing large amounts of playerbase to Fortnite, the current most played game (also of the battle royale genre). Adding features that might support interest in the game would improve player retention and increase player competitiveness.

Hypothesis

From the Kaggle data I aim to answer questions most players face when playing a match of PUBG. From the data I want to ask the following questions

1. Do players that get more kills, end up more likely to win a match?
2. Do players that travel less, end up more likely to win a match?
3. Where on the map do players end up dying the most?
4. Does using a vehicle in a match mean you do worse?

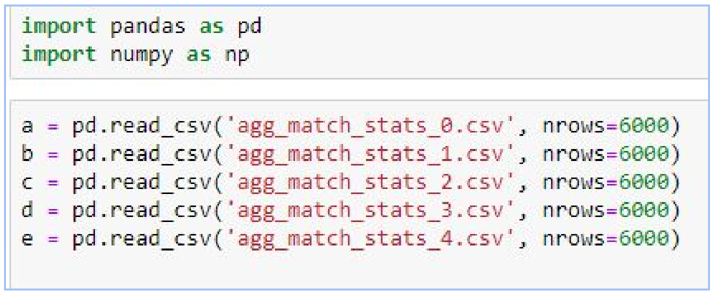
If you were to ask your average player some of these questions, you would find very mixed answers. People's strategies differ, and they base it off their experiences.

Uses

PUBG Corporation could use the results from my analysis to create a much needed tips system in the game that would promote players skills thus increasing player retention. The analysis could also benefit video game guide makers on the internet, commonly found on youtube. Esports is a booming industries, and many of the top players of PUBG would be able to utilize the analysis to gain a competitive advantage. Streamers on Twitch can test out these tips live and share their results with viewers to see if it improved their gameplay.

Data Wrangling

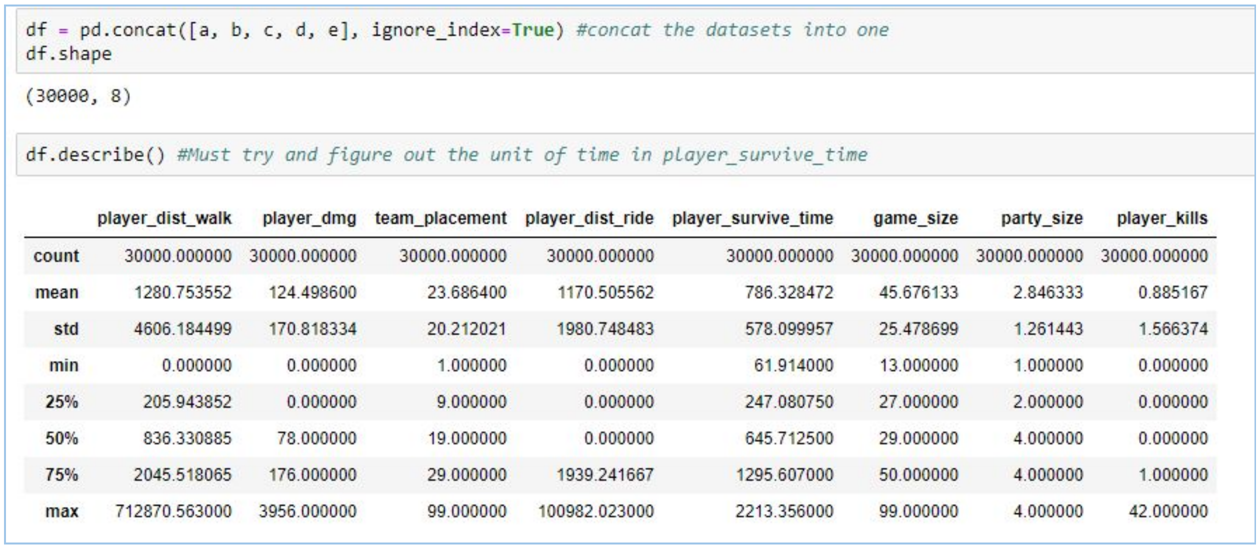
The dataset was in the form of a csv file from Kaggle. I began the project by reading the data using pandas. The data was split up in 5 files. I had to read each file in individual lines on my first trail doing the cleaning process. I came upon the issue that it took too much memory even for my fairly powerful Windows gaming desktop to work. To alleviate this issue I utilized the nrows method on pandas read csv function to only take in the first 6000 rows of each data set. Upon earlier inspection I found that 6000 rows of each file contained a fairly balanced amount of recordings from each game mode. I then decide to reduce the dataset to only pertinent columns, excluding the variables match_id, date, match_mode, player_name, and team_id.



```
import pandas as pd
import numpy as np
```

```
a = pd.read_csv('agg_match_stats_0.csv', nrows=6000)
b = pd.read_csv('agg_match_stats_1.csv', nrows=6000)
c = pd.read_csv('agg_match_stats_2.csv', nrows=6000)
d = pd.read_csv('agg_match_stats_3.csv', nrows=6000)
e = pd.read_csv('agg_match_stats_4.csv', nrows=6000)
```

I took each of these 5 files and used pandas concat function to combine then into a single file file while ignoring the index. I inspect the file though the describe function to gain a view of the descriptive statistics. Some likely outliers exist simply looking at the max amount of kills at 42. I reduce the dataset to contain only observations where the play moved greater than 5 units of distance, and 10 units of time. The actual metrics being used are not given on the kaggle website, the variance in the numbers leads to to assume the distance traveled is in meters, and the time survived is recorded in seconds. In my next step I then divide the survival time by 60 to convert them into minutes.



```
df = pd.concat([a, b, c, d, e], ignore_index=True) #concat the datasets into one
df.shape
```

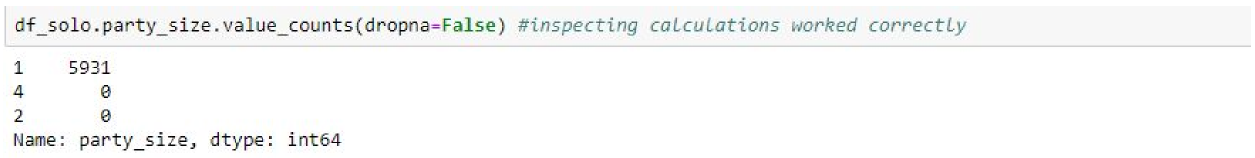
```
(30000, 8)
```

```
df.describe() #Must try and figure out the unit of time in player_survive_time
```

	player_dist_walk	player_dmg	team_placement	player_dist_ride	player_survive_time	game_size	party_size	player_kills
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	1280.753552	124.498600	23.686400	1170.505562	786.328472	45.676133	2.846333	0.885167
std	4606.184499	170.818334	20.212021	1980.748483	578.099957	25.478699	1.261443	1.566374
min	0.000000	0.000000	1.000000	0.000000	61.914000	13.000000	1.000000	0.000000
25%	205.943852	0.000000	9.000000	0.000000	247.080750	27.000000	2.000000	0.000000
50%	836.330885	78.000000	19.000000	0.000000	645.712500	29.000000	4.000000	0.000000
75%	2045.518065	176.000000	29.000000	1939.241667	1295.607000	50.000000	4.000000	1.000000
max	712870.563000	3956.000000	99.000000	100982.023000	2213.356000	99.000000	4.000000	42.000000

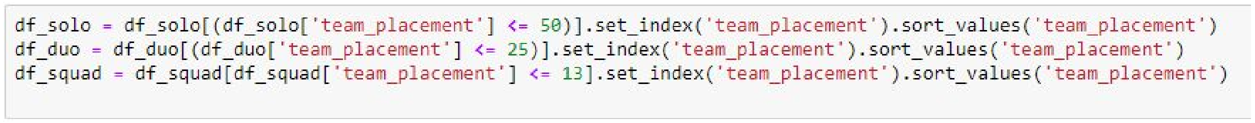
As for the reasoning for removing these recordings, I'm not very interested in the bottom half of team placements in a match. From experience playing the game, I can tell you that half of all teams usually are eliminated in the first 3 minutes of gameplay. The reasoning can be attributed to poor landing locations, not landing with your team so you end up getting ganged up on, poor accuracy, afk (away from keyboard) players, or poor loot finds. These variables while important, are not the aim of the analysis, and the data given will not unravel any insights on them.

From here I split up the data by game mode / party size. A game of Players Unknown Battlegrounds can be played in solo, duo, or squad mode. Squad mode contains up to 4 players which is what most players do, but a player can choose to go in with no teammates if willed. I also converted the data type of party_size into a category to reduce memory usage given that there's only 3 options here. I then set the index and sorted the value by team placement for each data set. Using the info method on each dataframe shows me the the total columns, observations, and data types of each column. I confirm once again that there are no null value and no surprises in the data. From here the data is ready to be saved into a newly cleaned csv file.

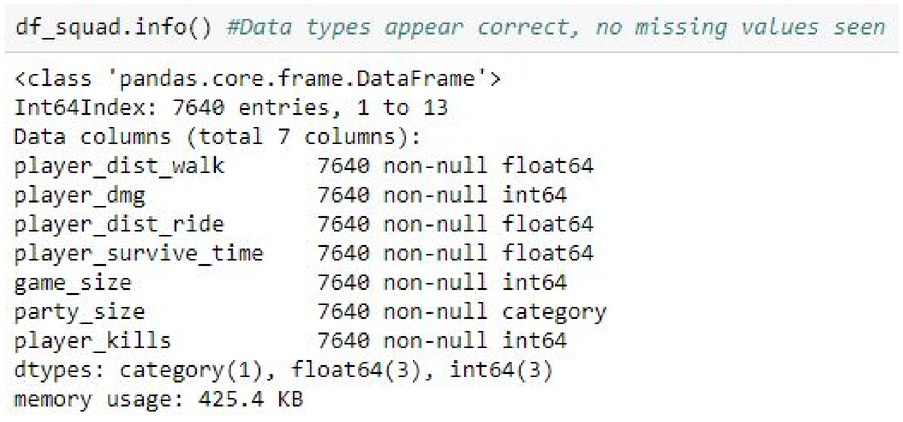


```
df_solo.party_size.value_counts(dropna=False) #inspecting calculations worked correctly
```

```
1    5931
4       0
2       0
Name: party_size, dtype: int64
```

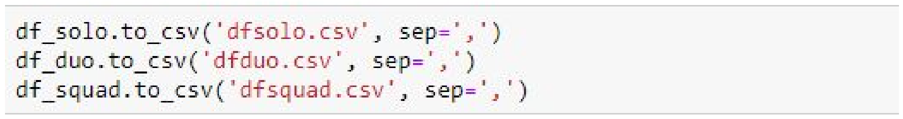


```
df_solo = df_solo[(df_solo['team_placement'] <= 50)].set_index('team_placement').sort_values('team_placement')
df_duo = df_duo[(df_duo['team_placement'] <= 25)].set_index('team_placement').sort_values('team_placement')
df_squad = df_squad[df_squad['team_placement'] <= 13].set_index('team_placement').sort_values('team_placement')
```



```
df_squad.info() #Data types appear correct, no missing values seen
```

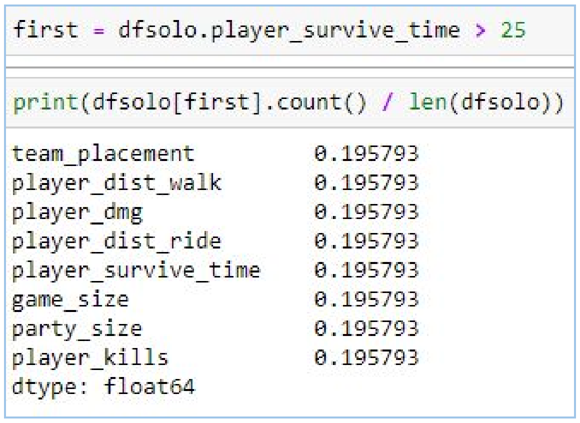
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7640 entries, 1 to 13
Data columns (total 7 columns):
player_dist_walk       7640 non-null float64
player_dmg             7640 non-null int64
player_dist_ride       7640 non-null float64
player_survive_time    7640 non-null float64
game_size              7640 non-null int64
party_size             7640 non-null category
player_kills           7640 non-null int64
dtypes: category(1), float64(3), int64(3)
memory usage: 425.4 KB
```



```
df_solo.to_csv('dfsolo.csv', sep=',')
df_duo.to_csv('dfduo.csv', sep=',')
df_squad.to_csv('dfsquad.csv', sep=',')
```

Exploratory Data Analysis

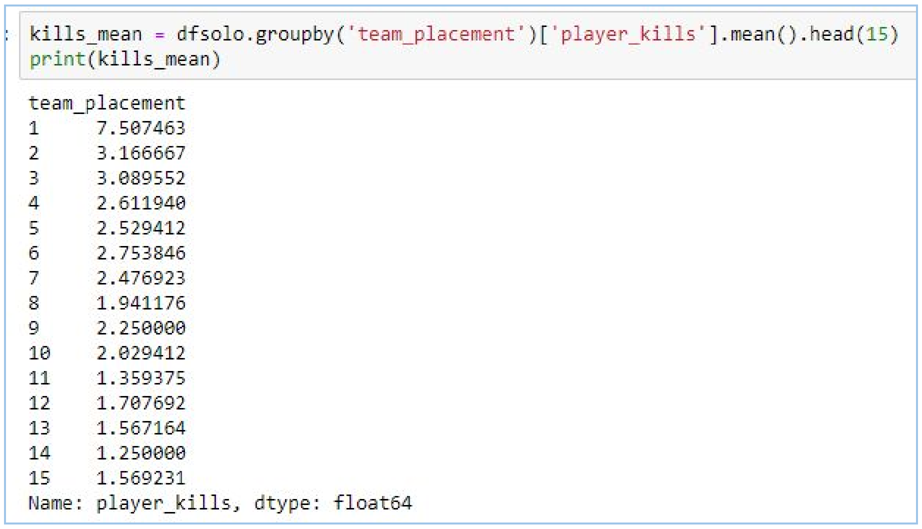
From here I begin some simple exploratory data analysis. The goal here is to find interesting trends and distributions of the data. With some filtering techniques using boolean operations and groupby sorting, I discover the following. 20% of the top half of players in a solo match will survive longer than 25 minutes in a match. So the top 10 players can expect a match to go beyond that time frame. The average kills for the top 2-7 players in a solo's match are between 2-3 kills, but first place average is 7.5, well above the rest.



```
first = dfsolo.player_survive_time > 25
```

```
print(dfsolo[first].count() / len(dfsolo))
```

```
team_placement         0.195793
player_dist_walk       0.195793
player_dmg             0.195793
player_dist_ride       0.195793
player_survive_time    0.195793
game_size              0.195793
party_size             0.195793
player_kills           0.195793
dtype: float64
```



```
kills_mean = dfsolo.groupby('team_placement')['player_kills'].mean().head(15)
print(kills_mean)
```

```
team_placement
1     7.507463
2     3.166667
3     3.089552
4     2.611940
5     2.529412
6     2.753846
7     2.476923
8     1.941176
9     2.250000
10    2.029412
11    1.359375
12    1.707692
13    1.567164
14    1.250000
15    1.569231
Name: player_kills, dtype: float64
```

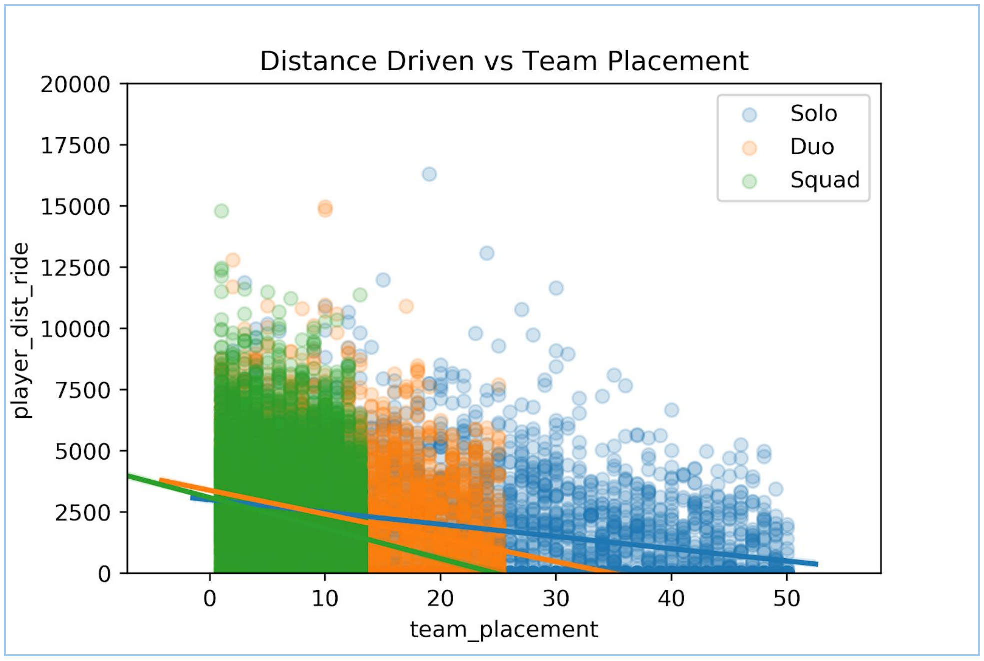
Using pair plots I see that most of the kills happen within the first 2000 meters of distance walked in a match. Seaborn's regression plot shows me that driving a vehicle results in better placement in squads and duo game modes, but has little help in solo matches.

```
#5
f, ax = plt.subplots(1, 1)
sns.regplot(x='team_placement', y='player_dist_ride', data=dfsolo, label ='Solo', scatter_kws={'alpha':0.2})

sns.regplot(x='team_placement', y='player_dist_ride', data=dfduo, label ='Duo', scatter_kws={'alpha':0.2})

sns.regplot(x='team_placement', y='player_dist_ride', data=dfsquad, label ='Squad', scatter_kws={'alpha':0.2})

plt.ylim(0, 20000)
plt.title('Distance Driven vs Team Placement')
ax.legend()
```



Correlations & Inferential Statistics

Pandas pearson correlation function shows the pearson correlation between all variables in a dataset. I’m looking for the variables that have the greatest correlation with team placements. Values with a low pearson correlation are of interest. The lower the number for team placement (number #1 !), the better. For all game modes, the top features come out to be player kills / damage, distance walked, and distance driven with values between -.2 and -.55 depending on the game mode.

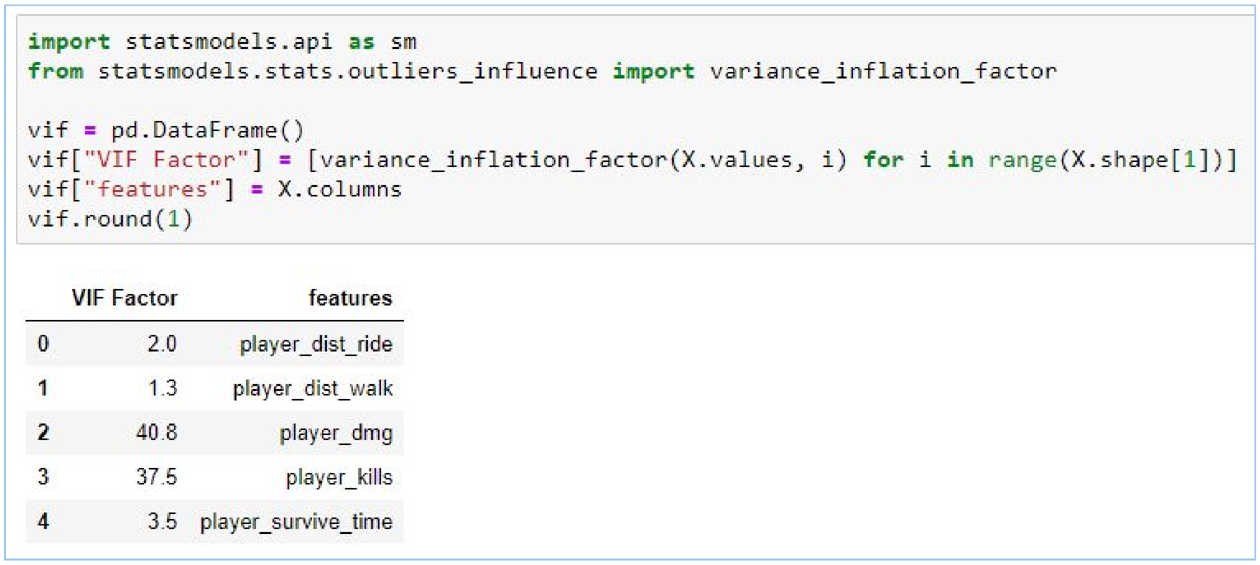
Using the Stats module of the Scipy library, I was able to conduct a few T-tests. From this I found that using vehicles in a solo match does not necessarily entail a better survival time by observing a p value of 6%, while in duo or squad mode it does. This makes sense because in solo matches you don’t have a teammate to support you in the passenger or back seats to protect you while you drive, you simply become a loud easy target.

Building A Model

In this step my goal is to see if I can build a classifier model to predict if a player were score first place or not given their stats in a Solo’s match. Full disclosure there aren’t many practical uses for this and is mostly a test of techniques.

To create dummy output variables I apply Numpy’s where method to convert all team placements that aren’t first place into 0, and all first place placements remain 1. Upon my first attempt at this model I discovered that I had a highly imbalanced dataset with far too many non first place scores, as to be expected as this was 1% of the dataset. So I go back to my original dataset and pull a larger amount of rows, taking a good chunk of first place observations to balance the data set so first place makes up 20%; 31,317 first place, and 125,268 non first places. We then separate the output variable (team placement) from the dependent variables.

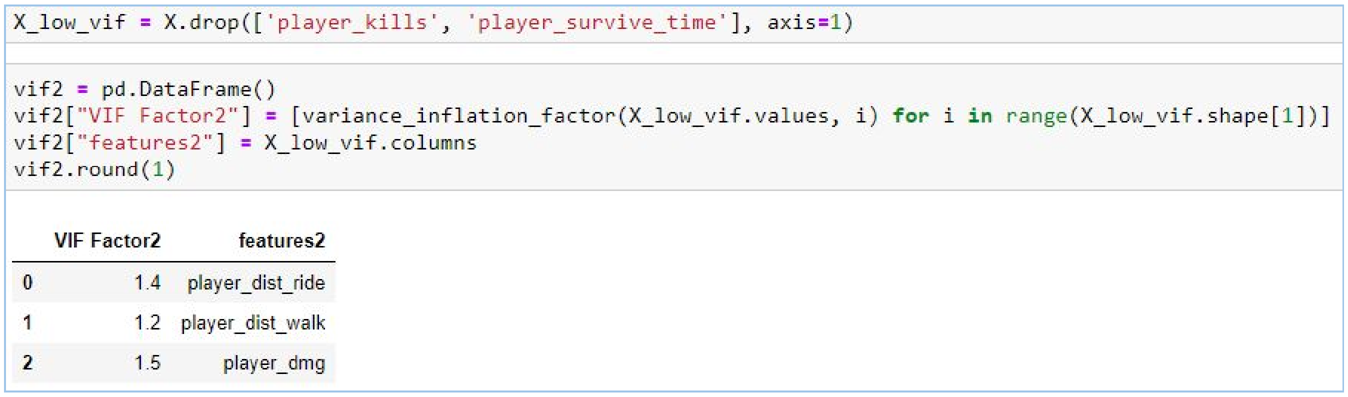
I apply two feature selection techniques: variance inflation factor and information value. VIF explains the colinearity variables are giving a data set, and IV gives the predicting power of each variable for the output variable.



```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["features"] = X.columns
vif.round(1)
```

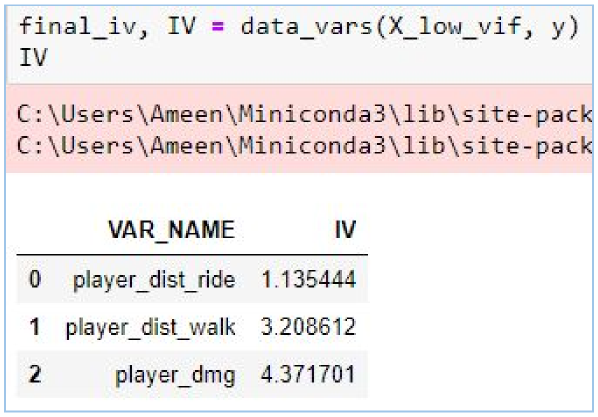
	VIF Factor	features
0	2.0	player_dist_ride
1	1.3	player_dist_walk
2	40.8	player_dmg
3	37.5	player_kills
4	3.5	player_survive_time



```
X_low_vif = X.drop(['player_kills', 'player_survive_time'], axis=1)

vif2 = pd.DataFrame()
vif2["VIF Factor2"] = [variance_inflation_factor(X_low_vif.values, i) for i in range(X_low_vif.shape[1])]
vif2["features2"] = X_low_vif.columns
vif2.round(1)
```

	VIF Factor2	features2
0	1.4	player_dist_ride
1	1.2	player_dist_walk
2	1.5	player_dmg



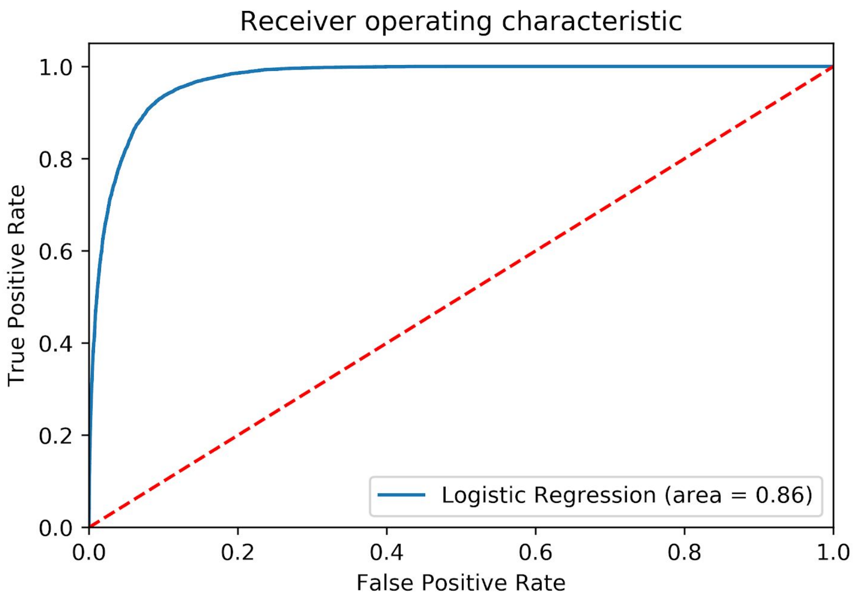
```
final_iv, IV = data_vars(X_low_vif, y)
IV
```

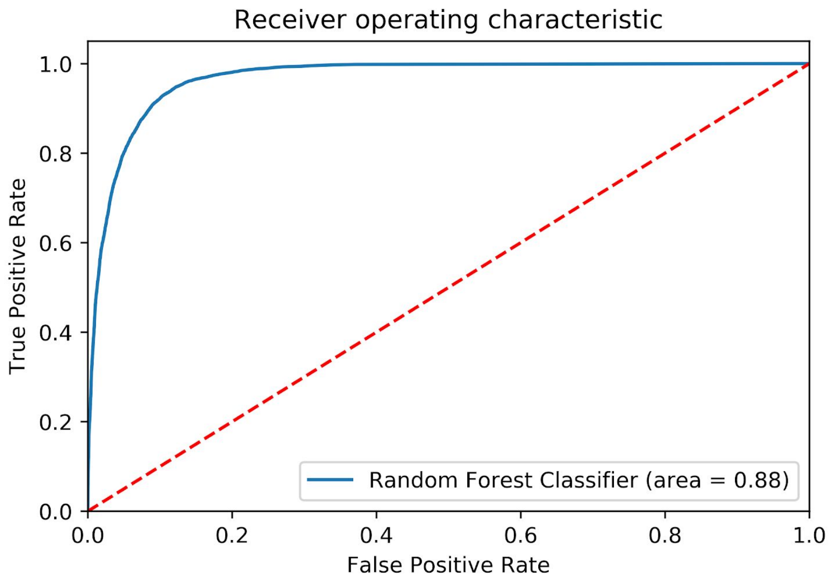
```
C:\Users\Ameen\Miniconda3\lib\site-pack
C:\Users\Ameen\Miniconda3\lib\site-pack
```

	VAR_NAME	IV
0	player_dist_ride	1.135444
1	player_dist_walk	3.208612
2	player_dmg	4.371701

As you can see the remaining features are the distance driven, distance walked, and player damage. From here I can split the data into a training and test set and apply fit my two models of choice to compare. Im choosing logistic regression and comparing it to a random forest classifier. My function run_regression_accuracy takes training and test data, and instantiates a logistic regression model, fits, and predicts the data. It prints out a confusion matrix, a classification report, and displays a receiver operating characteristic chart (ROC Curve), as well as shows the area under the curve score. For a classification report these three metrics are a suitable choice to compare a models performance. I apply an equivalent function to the data but with a random forest model, and the result is an auc score of 86% under the logistic regression model, and 88%

under the random forest classifier. With the three features mentioned I can accurately predict if a player will be first place in a match or not.





Results

From the analysis I can conclude that commiting to damage and kills, moving around, and utilizing vehicles (for duo and squad) are the attributes of a first place winner in Players Unknown Battlegrounds. This is to be expected due to pulling off more damage and kills will results in better loot, and clearing of areas. Staying mobile ensures you are scouting your surroundings and finding people first. Utilizing vehicles means not getting zones, first to loot buildings, and superior positioning. A greater walking distance will typically result of course by being in the later end of the match, but typically in the final minutes of the match there isn't a great deal of walking happening, as can be seen in the data. This debunks the common practice of hiding in a building for the majority of a match, often remote from the central action, and never making a sound in order to attempt a prolonged survival, but resulting in missed opportunities.

Future Endeavors

Newer datasets have begun coming out on containing more variables, which may shed light on other features that are commonly seen in the top placements. For example data on player accuracy can show just how important that is to winning a game. Location data can be used to determine fighting hotspots and where zone likely end.

I'm Ameen Abdelghani, thank you for reading my report on my first complete data science project. Feel free to reach out to me on LinkedIn if you have feedback on my work, cheers!