LAB MANUAL
FOR

SOFTWARE PROJECT (4649304)

Prepared By

AMAN KUMAR KHANNA(195690693005)
VINA GAJERA(195690693017)
SHIVAM KUMAR PANDEY (195690693069)

Supervised by:
Prof. Bhavesh B. Prajapati

MASTER OF COMPUTER APPLICATION

Government MCA College
Khokhra, Ahmedabad 380008
Affiliated to
GUJARAT TECHNOLOGICAL UNIVERSITY
Academic Year 2020-21

Academic Year:2020-2021

GOVERNMENT MCA COLLEGE, Maninagar
(East) K.K.shastri Educational Campus
Khokhra Road,Ahmedabad – 380008, Gujarat

Date:26/04/2021

CERTIFICATE

This is to certify that **Aman Kumar Khanna(195690693005)** Student of MCA Semester-IV, Government MCA College Has Successfully completed his practical submission on subject "**Software Project (4649304)**"of **Semester-4 Master of Computer Application** of **Gujarat Technological University,** Ahmedabad during academic year **2020-2021**.

Prof. Bhavesh B Prajapati
Project Guide
Government MCA College
Maninagar

Prof. Bhavesh B Prajapati
Head &Assistant Professor IT
Government MCA College
Maninagar

Dr.Chetan B Bhatt
Principal
Government MCA College,
Maninagar

Academic Year:2020-2021

GOVERNMENT MCA COLLEGE, Maninagar
(East) K.K.shastri Educational Campus
Khokhra Road,Ahmedabad – 380008, Gujarat

Date:26/04/2021

CERTIFICATE

This is to certify that **Vina Gajera(195690693017)** Student of MCA Semester-IV, Government MCA College Has Successfully completed his practical submission on subject "**Software Project (4649304)**"of **Semester-4 Master of Computer Application** of **Gujarat Technological University,** Ahmedabad during academic year **2020-2021**.

Prof. Bhavesh B Prajapati
Project Guide
Government MCA College
Maninagar

Prof. Bhavesh B Prajapati
Head &Assistant Professor IT
Government MCA College
Maninagar

Dr.Chetan B Bhatt
Principal
Government MCA College,
Maninagar

Academic Year:2020-2021

GOVERNMENT MCA COLLEGE, Maninagar
(East) K.K.shastri Educational Campus
Khokhra Road,Ahmedabad – 380008, Gujarat

Date:26/04/2021

CERTIFICATE

This is to certify that **Shivam Kumar Pandey(195690693069)** Student of MCA Semester-IV, Government MCA College Has Successfully completed his practical submission on subject "**Software Project (4649304)**"of **Semester-4 Master of Computer Application** of **Gujarat Technological University,** Ahmedabad during academic year **2020-2021**.

Prof. Bhavesh B Prajapati
Project Guide
Government MCA College
Maninagar

Prof. Bhavesh B Prajapati
Head &Assistant Professor IT
Government MCA College
Maninagar

Dr.Chetan B Bhatt
Principal
Government MCA College,
Maninagar

ACKNOWLEDGEMENT

There are a number of people who have helped us during the course of this project without whom, this project would definitely have not been completed.

First foremost, we would like to thank our own College **Government MCA College (GMCA)** for teaching basics without which we would never had a direction regarding this project and which modules to include and develop this Project and help us in almost each and every situation and provided all the requirement including certificate.

Secondly, we would like to thank our guides:

Prof. Bhavesh B Prajapati our project guide who had not only taught us, but also guided and encouraged us all long into completing this project and helped in almost all the possible way they could and they are the person who taught us making diagram and explained the importance of each diagram making it what is today and they communicated all the needs of the project to us and supported us by giving us her valuable feedback at almost every instance and provided all the documents like certificate and helped in preparing presentation, Diagrams, and Data Dictionary.

Dr. Chetan B. Bhatt, principle of Government MCA College who encouraged us throughout the project and helped us during our semester.

We would also like to thank our family and friends who have morally motivated us throughout the project and never left our sides and who have seen us through both the good times as well as bad times of this project. Moreover they have been extremely patient with us. We give them our heartfelt thanks for this kind of gesture. Lastly, we would like to thank god for showing us his mercy, keeping us, our nears and dear ones safe and sound, and carrying us in the palm of his hand all throughout this project.

Aman Kumar Khanna
Vina Gajera
Shivam Kumar Pandey

ABSTARCT

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.

We 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. The dataset we will be using for Data analysis is taken from Kaggle.com. Our approach will be by analysing the dataset in correct and appropriate manner and get the results from it.

INDEX

1. OBJECTIVES

INTRODUCTION

- In this report, we will show our approach in exploring and predicting final leaderboard placements in PlayerUnknown's Battlegrounds matches.
- We first give background information on the video game and context for the problem.
- We do exploratory data analysis. Afterwards, we perform feature engineering, creating more insightful features that better predict the target variable.
- We discuss postprocessing of our data to decrease our error. We also discuss the interesting discoveries we made when solving this problem. Finally, we discuss future steps to improve our models.

WHAT IS PUBG ?

- PUBG stands for PlayerUnknown's Battlegrounds is an online multiplayer battle royale game developed and published by PUBG Corporation, a subsidiary of South Korean video game company Bluehole.
- Basically, the game is all about the battle. This is player vs player action-shooter-openworld game wherein you start with nothing and as time goes by, you will scavenge and collect weapons and equipment.
- The game is ultimately a battle to see the last player standing among 100 players on an 8 x 8 km (and many other size map) island.
- The mode of the game is: Solo, Duo(2) or Squad(4)

PROBLEM STATEMENT

- In a PUBG game, up to 100 players start in each match (**matchId**). Players (**Id**) 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.

- During the game, players can pick up different amunitions, revive downed-but-not-dead (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.

GOALS

We will perform the PUBG data analysis and answer the following questions :

- Does killing more people increases the chance of winning the game?
- How do we catch the fraudsters in the game?
- Can we predict the finishing position of a player in the game?

TASKS TO PERFORMED

- Exploratory Data Analysis

- Data Cleaning: Outlier Detection and Removal - Finding the fraudsters

- Feature Engineering

- Final Prediction

2. UNDERSTANDING DATA

PUBG DATASET DESCRIPTION

- We'll be using data collected by Kaggle via the PUBG Developer API.
- The dataset comprises of 65,000 games worth of anonymized player data, split into training and tests sets.
- For the purposes of this exploratory analysis we're only going to be looking at the training data set.
- The training set comes in the form of a ".CSV" file. This file contains 11,32,90,736 individual data points in "26 columns" and "43,57,336 rows".

PUBG DATASET COLUMNS

- **groupId** - Integer 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.
- **matchId** - Integer ID to identify match. There are no matches that are in both the training and testing set.
- **boosts** - Number of boost items used.
- **damageDealt** - Total damage dealt. Note: Self inflicted damage is subtracted.
- **DBNOs** - Number of enemy players knocked
- **headshotKills** - Number of enemies you killed with headshots.
- **heals** - Number of healing items used.
- **killPlace** - Your ranking in match in terms of number of enemy players killed.
- **killPoints** - Kills-based external ranking of player.
- **kills** - Number of enemy players killed.
- **killStreaks** - Max number of enemy players killed in a short amount of time. A Killstreak is earned when a player acquires a certain number of kills in a row without dying.

- **longestKill** - Longest distance between player and player killed at time of death. This may be misleading, as downing a - player and driving away may lead to a large longestKill stat.

- **numGroups** - Number of groups we have data for in the match.

- **revives** - Number of times you revived your teammates.

- **rideDistance** - Total distance traveled in vehicles (measured in meters).

- **roadKills** - Number of enemy killed while travelling in a vehicle.

  swimDistance - Total distance traveled by swimming (measured in meters).

- **teamKills** - Number of times you are killed your 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. (Ranking where only winning matters).

- **winPlacePerc** - The target of prediction **(Target Variable)**. 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.

DATASET

```
[ ]  # csv file info
     train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4446966 entries, 0 to 4446965
Data columns (total 29 columns):
 #   Column           Dtype
---  ------           -----
 0   Id               object
 1   groupId          object
 2   matchId          object
 3   assists          int8
 4   boosts           int8
 5   damageDealt      float32
 6   DBNOs            int8
 7   headshotKills    int8
 8   heals            int8
 9   killPlace        int8
 10  killPoints       int16
 11  kills            int8
 12  killStreaks      int8
 13  longestKill      float32
 14  matchDuration    int16
 15  matchType        object
 16  maxPlace         int8
 17  numGroups        int8
 18  rankPoints       int16
 19  revives          int8
 20  rideDistance     float32
 21  roadKills        int8
 22  swimDistance     float32
 23  teamKills        int8
 24  vehicleDestroys  int8
 25  walkDistance     float32
 26  weaponsAcquired  int16
 27  winPoints        int16
 28  winPlacePerc     float32
dtypes: float32(6), int16(5), int8(14), object(4)
```

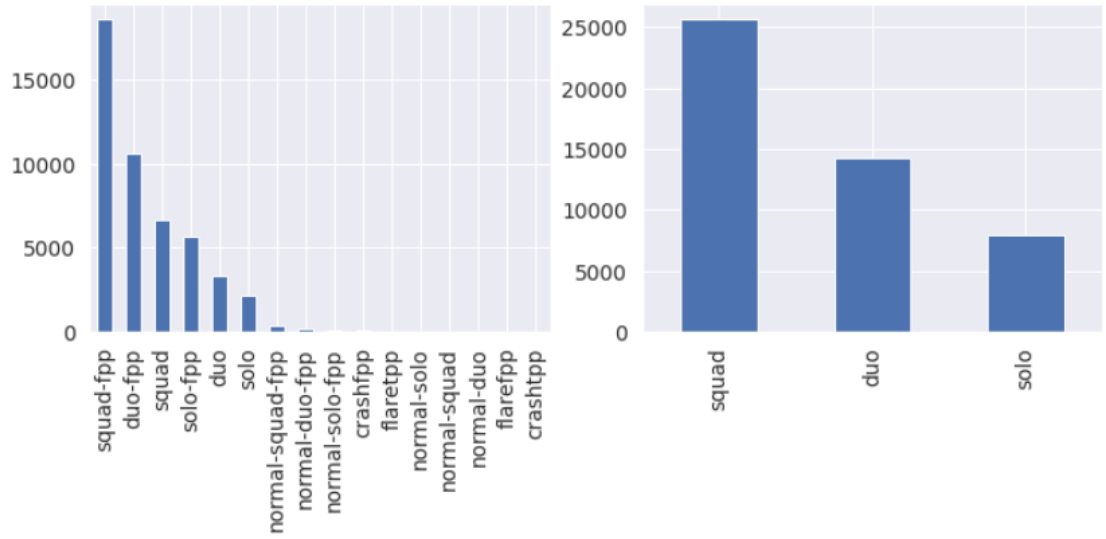
3. Exploratory Data Analysis

```
[ ]  # unique id, groupid, matchid
     for i in ['Id','groupId','matchId']:
         print(f'unique [{i}] count:', train[i].nunique())
```

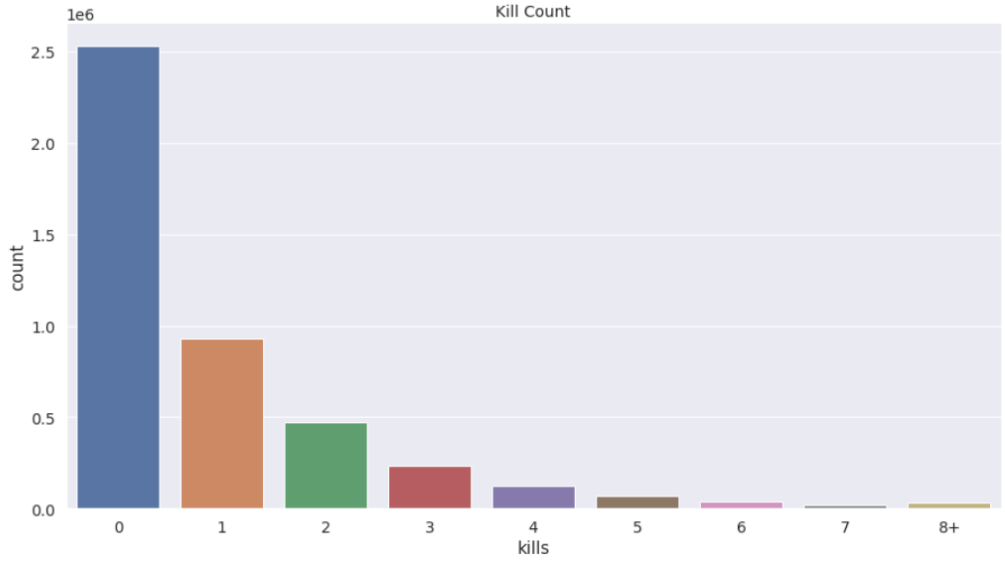
```
unique [Id] count: 4349359
unique [groupId] count: 2011668
unique [matchId] count: 47825
```

```
#PUBG offers 3 different game modes:
#Solo - One can play alone (solo,solo-fpp,normal-solo,normal-solo-fpp)
#Duo - Play with a friend (duo,duo-fpp,normal-duo,normal-duo-fpp,crashfpp,crashtpp)
#Squad - Play with 4 friends (squad,squad-fpp,normal-squad,normal-squad-fpp,flarefpp,flaretpp)
fig, ax = plt.subplots(1, 2, figsize=(12, 4))

train.groupby('matchId')['matchType'].first().value_counts().plot.bar(ax=ax[0])

mapper = lambda x: 'solo' if ('solo' in x) else 'duo' if ('duo' in x) or ('crash' in x) else 'squad'
train['matchType'] = train['matchType'].apply(mapper)
train.groupby('matchId')['matchType'].first().value_counts().plot.bar(ax=ax[1])
```

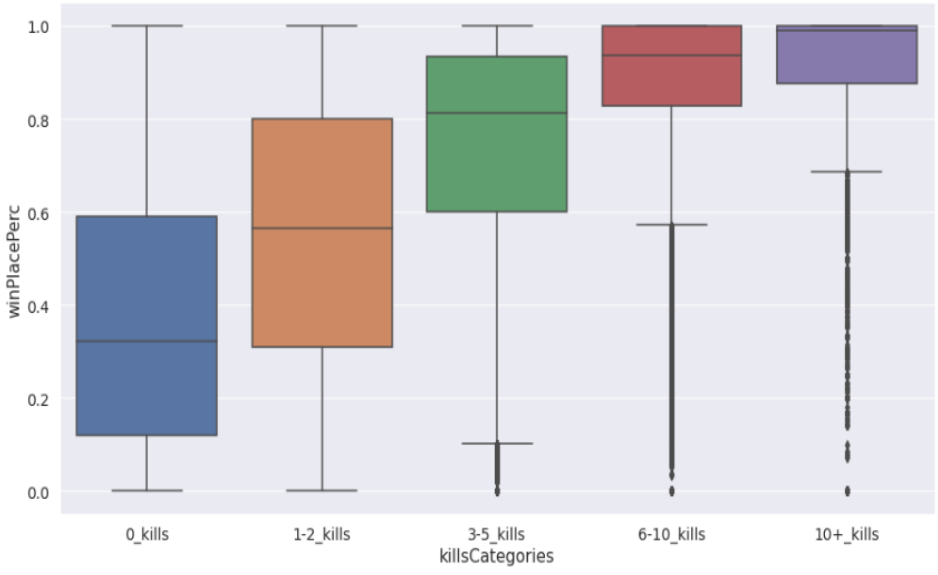
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2cc5980d10>
```



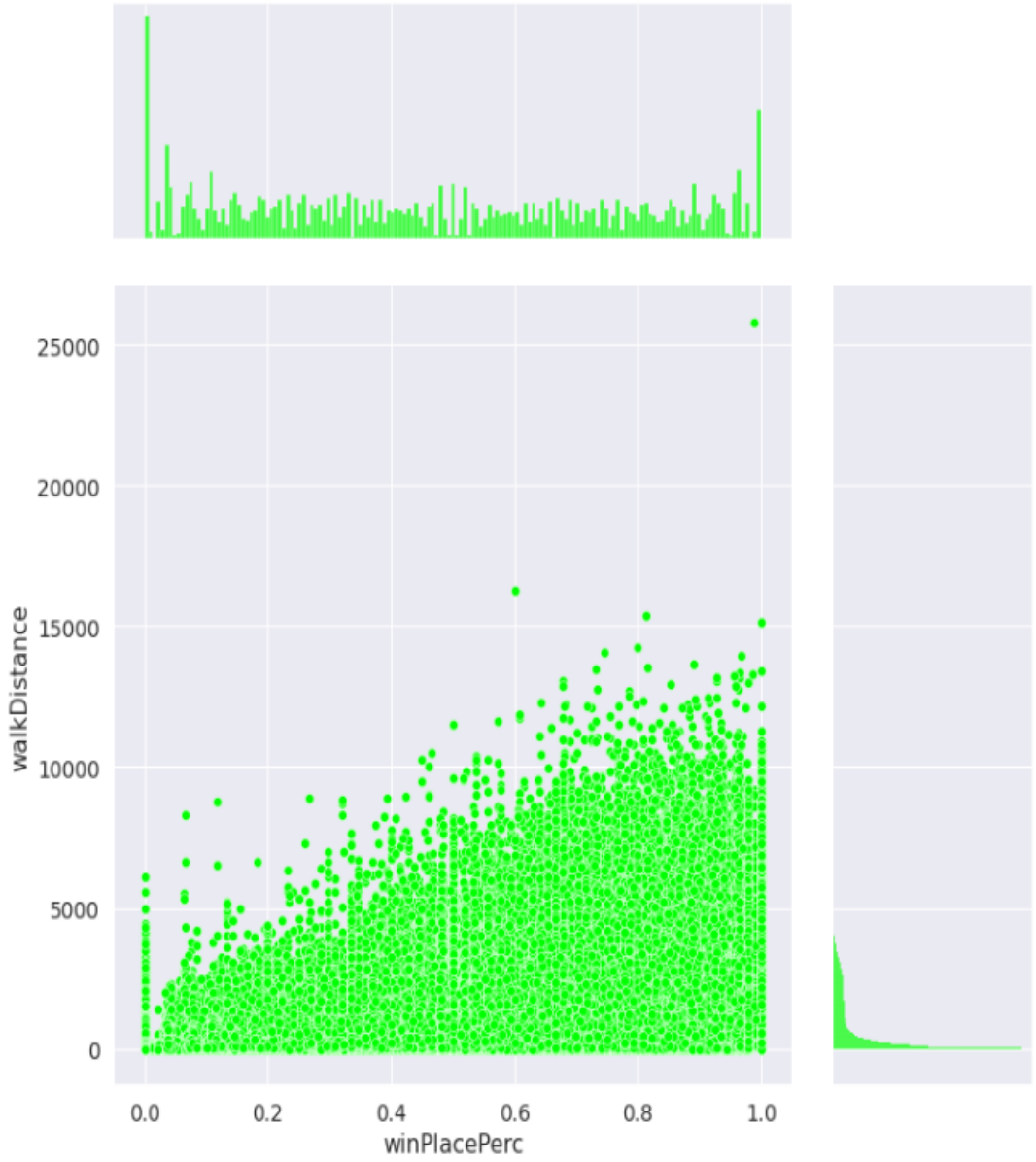
```
[ ]  #killcount
     data = train.copy()
     data.loc[data['kills'] > data['kills'].quantile(0.99)] = '8+'
     plt.figure(figsize=(15,8))
     sns.countplot(data['kills'].astype('str').sort_values())
     plt.title("Kill Count",fontsize=14)
     plt.show()
```



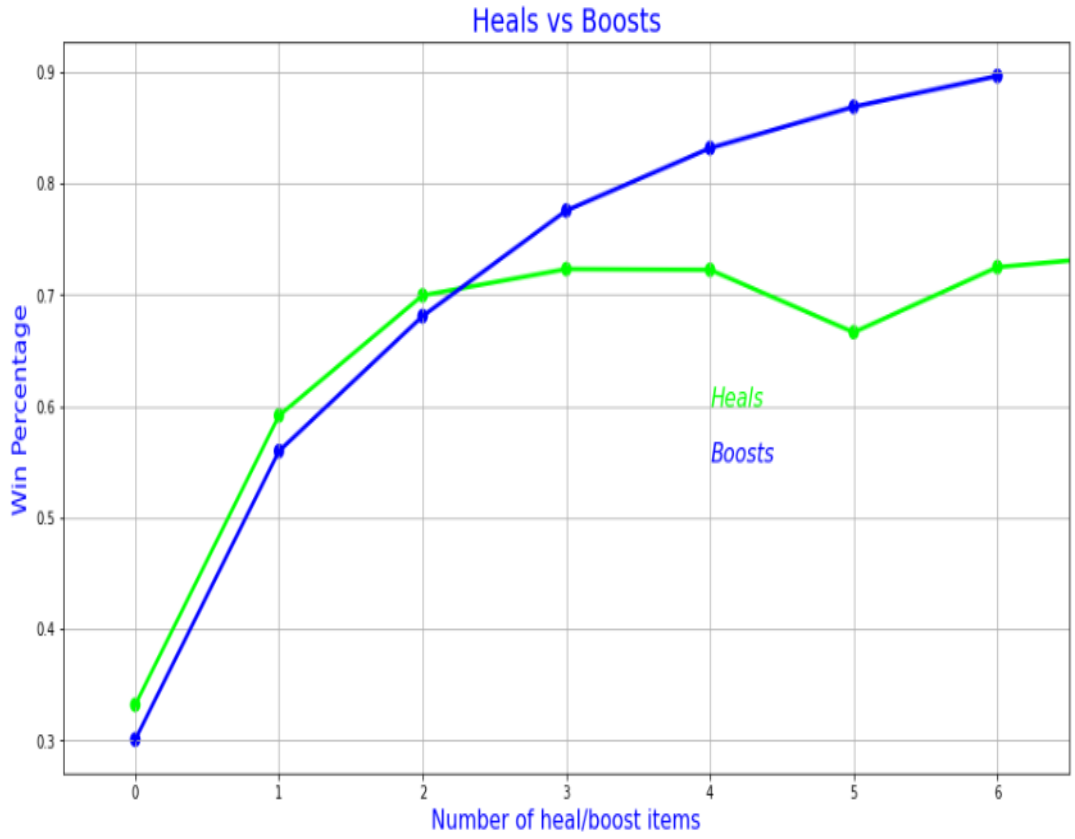

```
[ ]  #Effect of Killing on Winning percentage
     kills = train.copy()
     kills['killsCategories'] = pd.cut(kills['kills'], [-1, 0, 2, 5, 10, 60], labels=['0_kills','1-2_kills', '3-5_kills', '6-10_kills', '10+_kills'])
     plt.figure(figsize=(15,8))
     sns.boxplot(x="killsCategories", y="winPlacePerc", data=kills)
     plt.show()
```



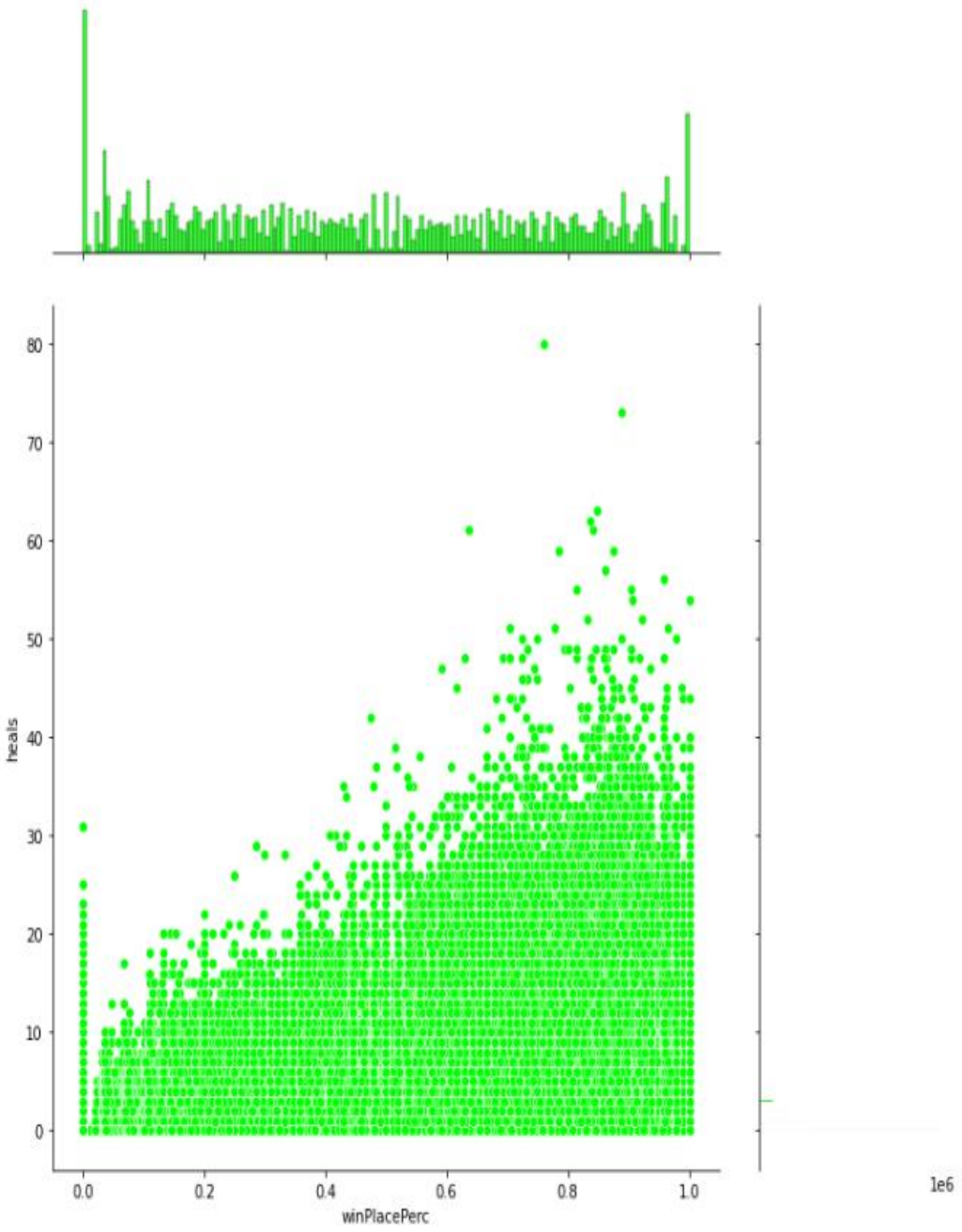
```
#longer you're alive, the more you run and more is the chance of winning.
sns.jointplot(x="winPlacePerc", y="walkDistance",  data=train, height=10, ratio=3, color="lime")
plt.show()
```



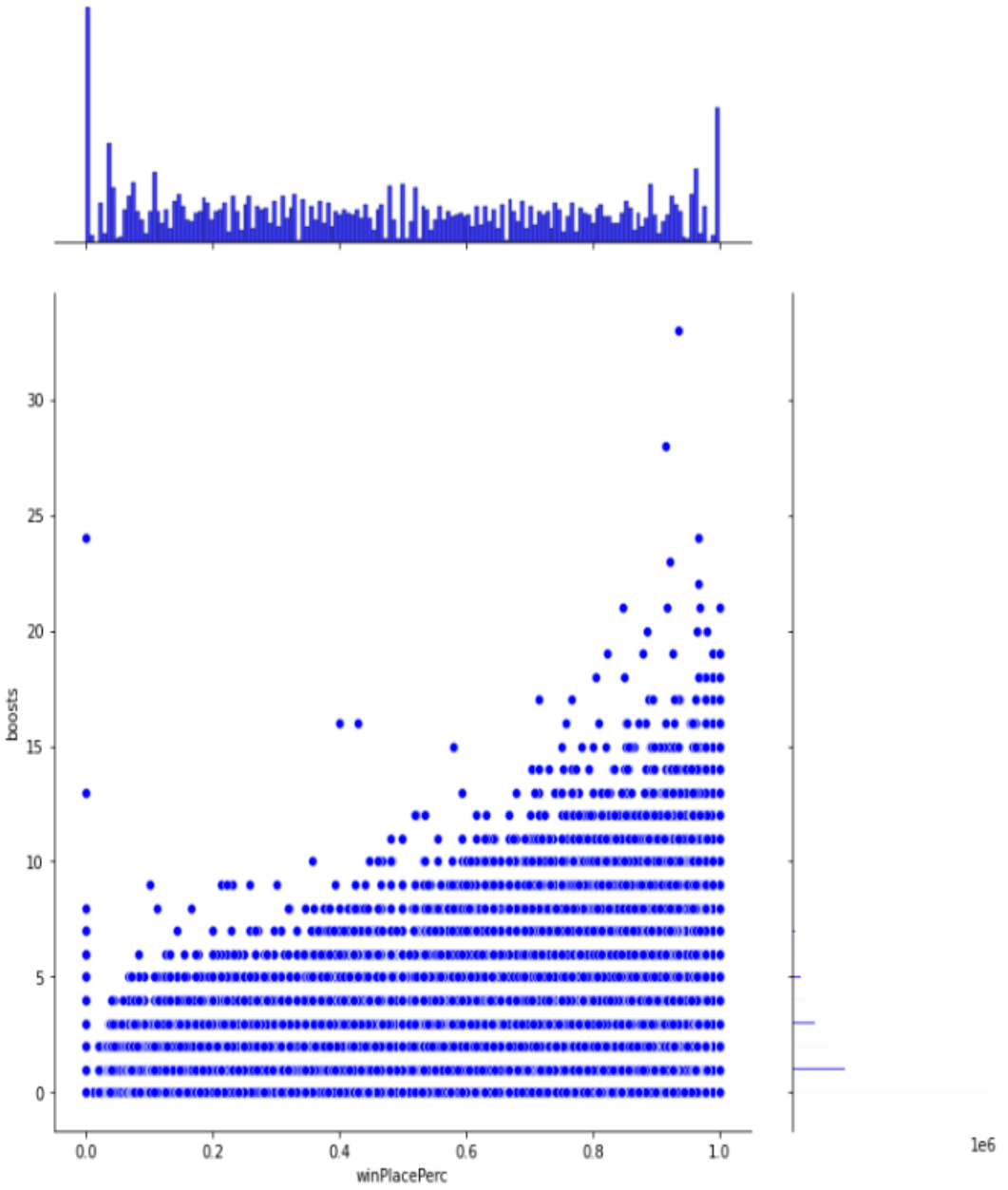
```
[ ] #Analysing Healing and Boosting: Effect of Healing & Boosting on Winning Percentage
    data = train.copy()
    data = data[data['heals'] < data['heals'].quantile(0.99)]
    data = data[data['boosts'] < data['boosts'].quantile(0.99)]

    f,ax1 = plt.subplots(figsize =(15,8))
    sns.pointplot(x='heals',y='winPlacePerc',data=data,color='lime',alpha=0.8)
    sns.pointplot(x='boosts',y='winPlacePerc',data=data,color='blue',alpha=0.8)
    plt.text(4,0.6,'Heals',color='lime',fontsize = 16,style = 'italic')
    plt.text(4,0.55,'Boosts',color='blue',fontsize = 16,style = 'italic')
    plt.xlabel('Number of heal/boost items',fontsize = 16,color='blue')
    plt.ylabel('Win Percentage',fontsize = 16,color='blue')
    plt.title('Heals vs Boosts',fontsize = 20,color='blue')
    plt.grid()
    plt.show()
```



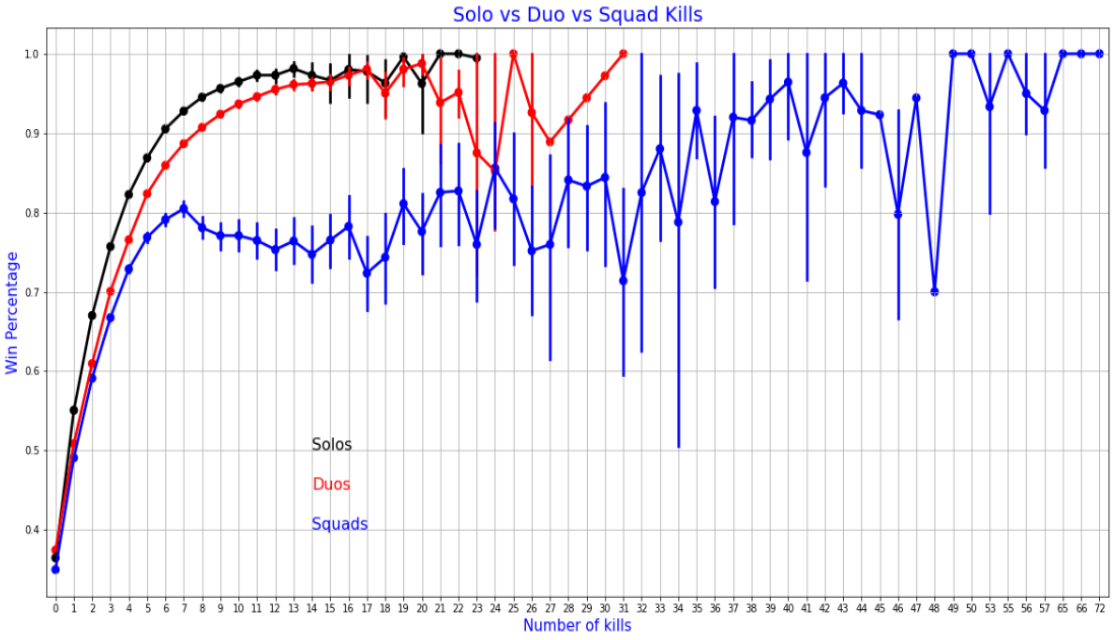
```
[ ] #heals
    sns.jointplot(x="winPlacePerc", y="heals", data=train, height=10, ratio=3, color="lime")
    plt.show()
```



```
[ ]  #boosts
     sns.jointplot(x="winPlacePerc", y="boosts", data=train, height=10, ratio=3, color="blue")
     plt.show()
```



```
[ ] f,ax1 = plt.subplots(figsize =(20,10))
    sns.pointplot(x='kills',y='winPlacePerc',data=solos,color='black',alpha=0.6)
    sns.pointplot(x='kills',y='winPlacePerc',data=duos,color='red',alpha=0.6)
    sns.pointplot(x='kills',y='winPlacePerc',data=squads,color='blue',alpha=0.6)
    plt.text(14,0.5,'Solos',color='black',fontsize = 16)
    plt.text(14,0.45,'Duos',color='red',fontsize = 16)
    plt.text(14,0.4,'Squads',color='blue',fontsize = 16)
    plt.xlabel('Number of kills',fontsize = 15,color='blue')
    plt.ylabel('Win Percentage',fontsize = 15,color='blue')
    plt.title('Solo vs Duo vs Squad Kills',fontsize = 20,color='blue')
    plt.grid()
    plt.show()
```



4. Data Cleaning: Outlier Detection and Removal

```
[ ] # Check row with NaN value
    train[train['winPlacePerc'].isnull()]
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills	heals	killPlace	killPoints	kills
2744604	f70c74418bb064	12dfbede33f92b	224a123c53e008	0	0	0.0	0	0	0	1	0	0

```
[ ] #Is it even possible to kill more than 40 people by acquiring more than 55 weapons and maintaining a total distance of less than 100m?
    train[(train['kills'] >= 40) & (train['weaponsAcquired'] > 55) & (train['_totalDistance'] < 100.0)]
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills	heals	killPlace	killPoints	kills
156599	746aa7eabf7c86	5723e7d8250da3	f900de1ec39fa5	21	0	5479.0	0	12	7	4	0	48
672993	da31f191ace8ed	ce9a3c4950a8f2	17dea22cefe62a	10	0	5793.0	0	5	2	1	0	57
770454	2ade4369bccd12	9f9e64a3db8384	e024bf51bf1799	12	0	5557.0	0	7	4	1	0	55
1378200	f241fdbb4f94c6	fadbbc4cbb3e06	e024bf51bf1799	9	0	3707.0	0	5	1	5	0	41

```
    #Is it even possible to kill more than 40 people without using any heals?
    train[(train['kills'] >= 40) & (train['heals'] == 0)]
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills	heals	killPlace	killPoints	kills
160254	15622257cb44e2	1a513eeecfe724	db413c7c48292c	1	0	4033.0	0	40	0	1	1000	42
2105633	770c02791306c4	48ca6706a90e10	6ee2c835176181	8	0	4272.0	40	34	0	1	0	43
2316123	dbc81aa64a7e3d	c50fe5c54e8bb6	5d58307bae9b50	8	0	4106.0	31	3	0	1	0	40
2601666	436d1530e9eb00	5c974c2bb9d9f0	fe7043ee6221c8	5	0	4347.0	0	31	0	1	0	53
3924729	579949f753978c	108f4c00d80882	ff9cd80c0d8fb7	1	2	3680.0	0	35	0	1	1000	40

```
[ ] # List of Hitman who made more than 10 kills and all the kills were done by headshot(perfect kill)
    display(train[(train['_headshot_rate'] == 1) & (train['kills'] >=10)].shape)
    train[(train['_headshot_rate'] == 1) & (train['kills'] >= 10)].head(10)
```

(24, 31)

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills	heals	killPlace	killPoints
281570	ab9d7168570927	add05ebde0214c	e016a873339c7b	2	3	1212.0	8	10	0	1	0
346124	044d18fc42fc75	fc1dbc2df6a887	628107d4c41084	3	5	1620.0	13	11	3	1	1424
871244	e668a25f5488e3	5ba8feabfb2a23	f6e6581e03ba4f	0	4	1365.0	9	13	0	1	1579
908815	566d8218b705aa	a9b056478d71b2	3a41552d553583	2	5	1535.0	10	10	3	1	1393
963463	1bd6fd288df4f0	90584ffa22fe15	ba2de992ec7bb8	2	6	1355.0	12	10	2	1	1543
1079403	1c245ed99b5f96	e42d09a9b8463a	5cec236bce68eb	0	5	1218.0	8	11	3	1	0
1167959	c4f80d4be5c561	b4a7892189b5dd	c7f7733ebbd447	0	4	1065.0	6	10	1	1	1391
1348164	474a641f0a4bcb	2fdad3ca6fb3c0	114499c82f35d7	1	5	1319.0	11	12	1	1	0
1380385	202ce6a55119c5	2df66861f597b4	496700c29a5d44	1	4	1150.0	4	11	1	1	1000
1483199	9d483f7cbb34d4	db5867bc814191	69495e3c478eb9	0	10	1478.0	8	13	2	1	0

```
# Create feature killsWithoutMoving
train['_killsWithoutMoving'] = ((train['kills'] > 0) & (train['_totalDistance'] == 0))

# Check players who kills without moving
display(train[train['_killsWithoutMoving'] == True].shape)
train[train['_killsWithoutMoving'] == True].head(10)
```

```
(1535, 32)
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills	heals	killPlace
1824	b538d514ef2476	0eb2ce2f43f9d6	35e7d750e442e2	0	0	593.000000	0	0	3	18
6673	6d3a61da07b7cb	2d8119b1544f87	904cecf36217df	2	0	346.600006	0	0	6	33
11892	550398a8f33db7	c3fd0e2abab0af	db6f6d1f0d4904	2	0	1750.000000	0	4	5	3
14631	58d690ee461e9d	ea5b6630b33d67	dbf34301df5e53	0	0	157.800003	0	0	0	69
15591	49b61fc963d632	0f5c5f19d9cc21	904cecf36217df	0	0	100.000000	0	1	0	37
20881	40871bf43ddac7	2cea046b7d1dce	0600f86f11c6e4	0	0	506.100006	4	1	3	7
23298	b950836d0427da	1f735b1e00d549	ad860f4e162bbc	1	0	1124.000000	0	4	1	7
24640	aeced11d46de19	d4009ffa95bb4f	73f3ed869c9171	2	0	529.900024	0	2	11	12
25659	6626c4d47cffa0	ee3fe5c0d917c3	341341834b7941	0	1	128.899994	0	1	6	53
30079	869331b90bfa3f	869ea3ad036e53	fa373e28ff5062	0	0	85.559998	0	0	0	46

```
#Is it even posssible to snipe (kill) someone from a distance > 1Km in this game?
train[train['longestKill'] >= 1000]
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs
202281	88e2af7d78af5a	34ddeede52c042	4346bc63bc67fa	0	3	783.900024	5
240005	41c2f5c0699807	9faecf87ab4275	634edab75860b3	5	0	1284.000000	8
324313	ef390c152bcc3d	30fd444be3bbc1	4f7f8d6cf558b4	2	0	1028.000000	0
656553	9948b058562163	c8cb8491112bf6	0104eeb664494d	6	0	1410.000000	17
803632	4e7e6c74e3c57d	94698690918933	da91b0c3d875f8	0	0	196.800003	0
895411	1f5ba6e0cfb968	512ea24b831be3	5fb0d8b1fc16cf	4	0	1012.000000	11
1172437	303a93cfa1f46c	8795d39fd0df86	9c8962b58bb3e3	2	1	329.299988	0
1209416	528659ff1c1aec	7d1ba83423551d	ea9386587d5888	0	6	1640.000000	0
1642712	91966848e08e2f	0ee4fbd27657c9	17dea22cefe62a	3	2	2103.000000	0
2015559	5ff0c1a9fab2ba	2d8119b1544f87	904cecf36217df	3	3	1302.000000	0
2122128	42df3102cb540b	7d9b2be15b355b	610d78f3affd2e	5	0	2500.000000	0

```
[ ]  # Players who got more than 10 roadKills
     train[train['roadKills'] > 10]
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills
2733926	c3e444f7d1289f	489dd6d1f2b3bb	4797482205aaa4	0	0	1246.0	0	0
2767999	34193085975338	bd7d50fa305700	a22354d036b3d6	0	0	1102.0	0	0
2890740	a3438934e3e535	1081c315a80d14	fe744430ac0070	0	8	2074.0	0	1
3524413	9d9d044f81de72	8be97e1ba792e3	859e2c2db5b125	0	3	1866.0	0	5

```
# How is it even possible that a player is able to ride and kill without walking even a single meter ?
train[(train['walkDistance'] == 0) & (train['rideDistance'] > 0) & (train['kills'] > 0)]
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills
219621	49e86bcbb74918	5bf4ac7f5f89a0	05a97d10e3cf81	0	0	100.000000	1	0
1051447	c43116d1c3d4e5	bd09c97b67800a	aad60eee7eeb79	0	0	102.199997	0	0
1053213	4a8011f19b0e0e	037c89b0cef255	238171821a1d0e	1	0	100.000000	1	0
1166930	553f199b5538c5	560c2ba9dcbbd0	afb05e37d884fc	2	0	168.500000	1	0
1479750	bcdf0f6c5d08bf	ea2ed2cc5955dd	2b59ed013eaf15	0	0	100.000000	1	0
2221273	a280df87d394fc	61042f8c96538c	a0e8fe0599288c	0	0	100.000000	1	0
2382410	adbda59d2f11cd	343d453ffb18c8	cbb5f864c6f532	0	0	100.000000	1	0
2471705	b968373063f5a8	536e9c19659ce8	61c90034dc8174	0	0	93.470001	1	0
2907639	2bf1d34142358b	71c15011651348	08dfa30b9390f7	4	0	578.000000	0	0
3778837	ca3ca72549b07f	68a0c84f3b09aa	f72f06cafb3349	0	0	144.300003	2	0
3878026	98b29b9a3c1d2d	5367c310b33253	32b26ac193c9ee	0	0	100.000000	1	0
4223250	5e7e5c37b5bfaa	4b8e96fa341568	73cb1c5f2685f3	0	0	100.000000	1	0
4261511	5151e02af2b0ef	ab966381a28eac	560ec48b1f3371	0	0	272.799988	0	0

```
# What was the player doing in the game when total distance travelled by him/her is 0?
train[(train['_totalDistance'] == 0)]
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs
29	ac5b57ff39979c	857cc55b2b6001	e019e04dee4f19	0	0	0.0	0
116	6adb021f5165ff	58e5500bd40898	de5c692fe25a73	0	0	0.0	0
151	a2bbe20aa8789d	926e8a09bab249	e36e4203ed4831	0	0	0.0	0
237	baaa694658e085	d034728f22cff7	fa71620624d3e7	0	0	0.0	0
283	3ab8128e6bcbe6	bb52a209f2e938	aabd2650b129e2	0	0	0.0	0
...	...	...	...	...	...	...	...
4446843	56f1ff8594a328	63008ee9676bc3	b3f07953e112dc	0	0	0.0	0
4446849	19d2b52c9e17a3	f8a952ce9ab7aa	2afb2889026133	0	0	0.0	0
4446905	f93dfebceecc59	c54ca5bb9df107	e1da3a1ee799ef	0	0	0.0	0
4446926	e0c791ea3c4644	e18240dcce54f9	827639896a20cb	0	0	0.0	0
4446958	837349af7e8a35	58bc4104935623	2001300d4f5787	0	0	0.0	0

95835 rows × 32 columns

```
# How can you swim for more than 2 km without breathing?
train[train['swimDistance'] >= 2000]
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	H
177973	c2e9e5631f4e54	23213058f83abe	f01eb1073ef377	0	5	78.120003	1	
274258	ba5e3dfb5a0fa0	383db055216ec2	d6e13468e28ab4	0	4	53.320000	0	
1005337	d50c9d0e65fe2a	4996575c11abcb	66840259242 9f8	0	1	503.000000	4	
1195818	f811de9de80b70	d08ddf7beb6252	8a48703ab52ec8	0	7	352.299988	3	
1227362	a33e917875c80e	5b72674b42712b	5fb0d8b1fc16cf	0	1	589.200012	3	
1889163	bd8cc3083a9923	1d5d17140d6fa4	8e2e6022d6e5c8	0	0	0.000000	0	
2065940	312ccbb27b99aa	47c7f4d69e2fb1	b4b11756321f3a	1	3	49.590000	0	
2327586	8773d0687c6aae	b17f46f9f6666c	56ee5897512c86	3	1	474.399994	2	
2784855	a8653b87e83892	383db055216ec2	d6e13468e28ab4	1	4	843.799988	5	
3359439	3713b36e1ba9e1	1f7aed9240864a	584447ed875c85	0	0	0.000000	0	
3513522	aff482b8c08486	383db055216ec2	d6e13468e28ab4	0	4	109.800003	0	
4132225	2496e3223a8b5d	78980ab36f7642	23ec7dd5546022	0	0	0.000000	0	

```
#acquiring guns in game
display(train[train['weaponsAcquired'] >= 80].shape)
train[train['weaponsAcquired'] >= 80].head()
```

```
(21, 32)
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs
233643	7c8c83f5f97d0f	b33b210a52a2f8	2e8a0917a71c43	0	0	67.110001	0
588387	c58e3e0c2ba678	3d3e6100c07ff0	d04dbb98249f76	0	1	175.300003	1
1437471	8f0c855d23e4cd	679c3316056de8	fbaf1b3ae1d884	1	0	100.000000	0
1449293	db54cf45b9ed1c	898fccaeeb041d	484b4ae51fe80f	0	0	0.000000	0
1462706	be4ff9afaa5bb1	abb73dd57619fa	f900de1ec39fa5	22	0	5377.000000	0

```
# 40 or more healing items used
display(train[train['heals'] >= 40].shape)
train[train['heals'] >= 40].head(10)
```

```
(135, 32)
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs
18405	63ab976895d860	927eeba5614c4f	69473402649f11	0	2	0.0	0
54463	069ddee7c9d26a	58ab5a1ce8e06f	942416b6caf21e	1	4	182.0	0
126439	c45bd6917146e2	81ab9f863957cb	4335664c6716fa	0	2	0.0	0
259351	86910c38335c2f	2738398928d28c	7d2911e944bfaa	0	10	0.0	0
268747	a007734fbc6ebf	5bf702dfa1e5d4	ad6b5669d33a2c	0	5	0.0	0
269098	a0891dbc2950ea	dde848d90491ba	b4fd3348551b73	0	2	0.0	0
284195	91a2fb00455eb3	f639b09774c5b1	65b73c71653822	0	3	123.0	0
300204	1f4f2efc86bfcb	3d668492d1fca9	d3638466a43d38	0	6	175.0	2
349908	7725ad71ad2ff7	4b2a7cf86d1546	cfa2775c9ef944	3	0	2348.0	0
375156	d64866c78ebcb0	aa0f089ae6430c	4dbc4ebba33ec6	0	7	278.5	3

```
[ ]  # Remove outliers
     train.drop(train[train['heals'] >= 40].index, inplace=True)
```

```
[ ]  train.shape
```

```
     (4349359, 32)
```

```
[ ]  train.to_csv('/content/drive/My Drive/Colab/DS/cleaned_data.csv', index=False)
```

```
[ ]  # import pandas as pd
     cleaned_data = pd.read_csv('/content/drive/My Drive/Colab/DS/cleaned_data.csv')
     cleaned_data = reduce_mem_usage(cleaned_data)
```

```
     Memory usage of dataframe is 1032.82 MB --> 377.46 MB (Decreased by 63.5%)
```

5. FEATURE ENGINEERING

Adding/Removing some new Features and finding their correlation with the winPlacePer

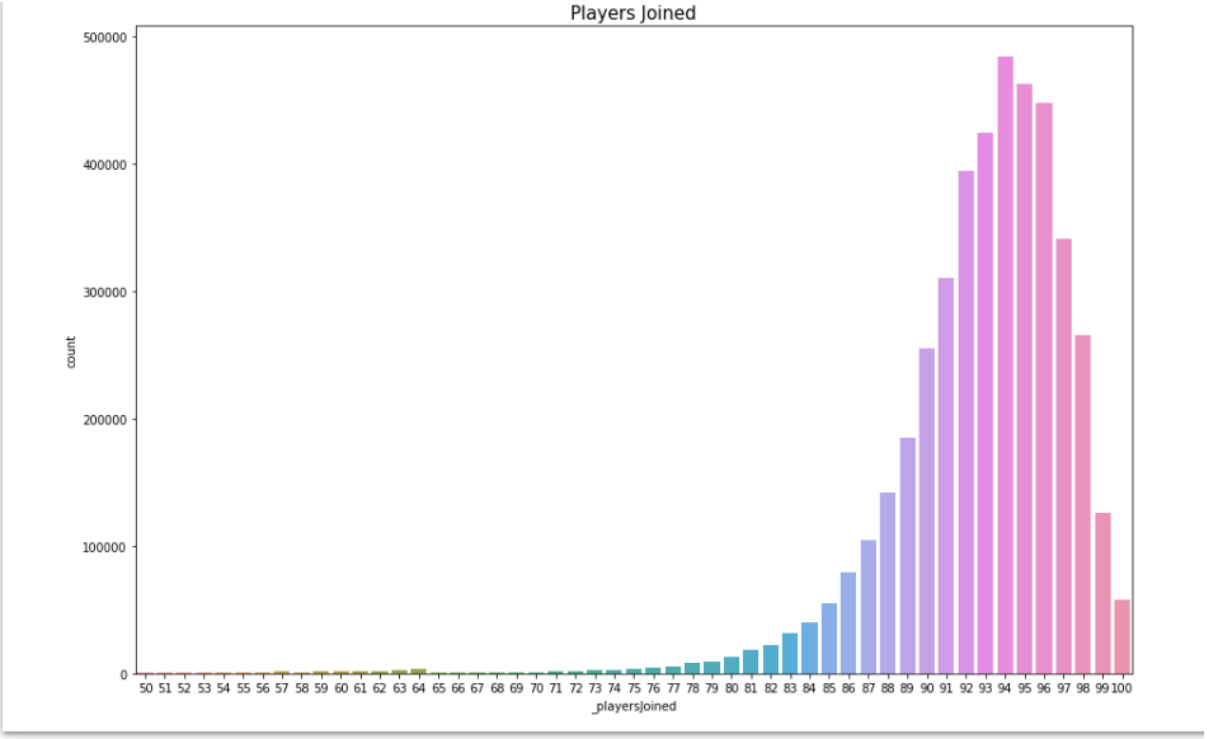
A game in PUBG can have up to 100 players fighting each other. But most of the times a game isn't "full". There is no variable that gives us the number of players joined. So lets create one.

```
# Create normalized features
cleaned_data['_killsNorm'] = cleaned_data['kills']*((100-cleaned_data['_playersJoined'])/100 + 1)
cleaned_data['_damageDealtNorm'] = cleaned_data['damageDealt']*((100-cleaned_data['_playersJoined'])/100 + 1)
cleaned_data['_maxPlaceNorm'] = cleaned_data['maxPlace']*((100-cleaned_data['_playersJoined'])/100 + 1)
cleaned_data['_matchDurationNorm'] = cleaned_data['matchDuration']*((100-cleaned_data['_playersJoined'])/100 + 1)
# Compare standard features and normalized features
to_show = ['Id', 'kills','_killsNorm','damageDealt', '_damageDealtNorm', 'maxPlace', '_maxPlaceNorm', 'matchDuration', '_matchDurationNorm']
cleaned_data[to_show][0:11]
```

```
match = cleaned_data.groupby('matchId')
cleaned_data['_killsPerc'] = match['kills'].rank(pct=True).values
cleaned_data['_killPlacePerc'] = match['killPlace'].rank(pct=True).values
cleaned_data['_walkDistancePerc'] = match['walkDistance'].rank(pct=True).values
cleaned_data['_damageDealtPerc'] = match['damageDealt'].rank(pct=True).values
cleaned_data['_walkPerc_killsPerc'] = cleaned_data['_walkDistancePerc'] / cleaned_data['_killsPerc']
cleaned_data.head()
```

_totalDistance	_headshot_rate	_killsWithoutMoving	_playersJoined	_killsNorm	_damageDealtNorm	_maxPlaceNorm
244.800003	0.0	0.0	94	0.00	0.000000	29.68
1445.044556	0.0	0.0	90	0.00	100.617001	28.60
161.800003	0.0	0.0	93	0.00	72.760000	53.50
202.699997	0.0	0.0	91	0.00	35.861002	33.79
49.750000	0.0	0.0	94	1.06	106.000000	102.82

_killsPerc	_killPlacePerc	_walkDistancePerc	_damageDealtPerc	_walkPerc_killsPerc
0.297872	0.638298	0.468085	0.148936	1.571429
0.250000	0.633333	0.555556	0.433333	2.222222
0.279570	0.505376	0.344086	0.419355	1.230769
0.274725	0.824176	0.230769	0.296703	0.840000
0.659574	0.478723	0.180851	0.563830	0.274194



There are a few matches with less than 75 players that cannot be displayed here. As you can see most of the matches are nearly packed and have almost 100 players.

6. Data Analysis/Algorithm

DECISION TREE

- Decision tree algorithm is category of supervised learning.
- Decision tree algorithm can be used to solve regression problems and classification problems.
- It's motto is create a model which predicts the value of a target parameter based on different input parameter.
- Decision tree can be used in predicting the dependent variable like fog and rain.

RANDOM FOREST

- Random forest algorithm is very easy to measure the relative importance of every feature on the prediction.
- It can be used for both regression and classification tasks and that it's easy to view the relative importance it assigns to the input features.
- Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity- weighted average of missing values.

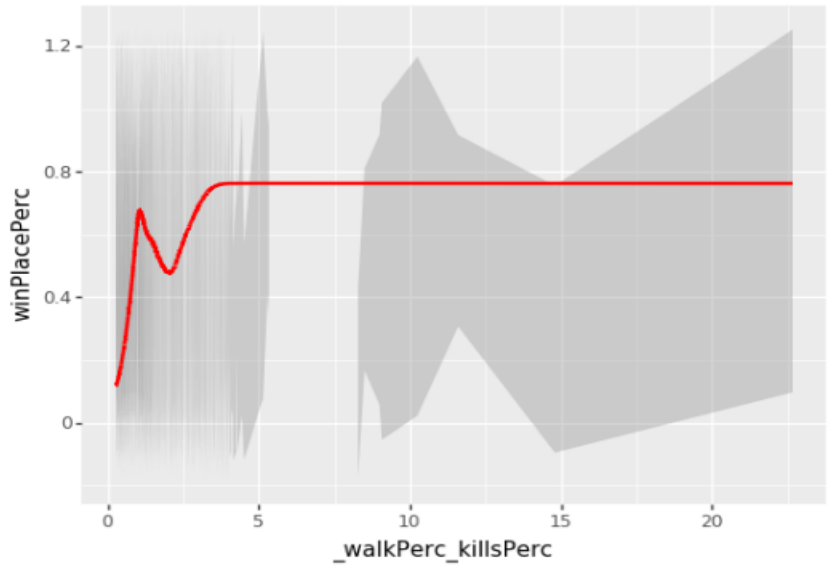
LOGISTIC REGRESSION

- The use of logistic regression modeling has exploded during the past decade for prediction and forecasting.
- Logistic regression allows one to predict a discrete outcome, such as whether it will rain today or not, from a set of variables that may be continuous, discrete, dichotomous, or a mix of any of these.

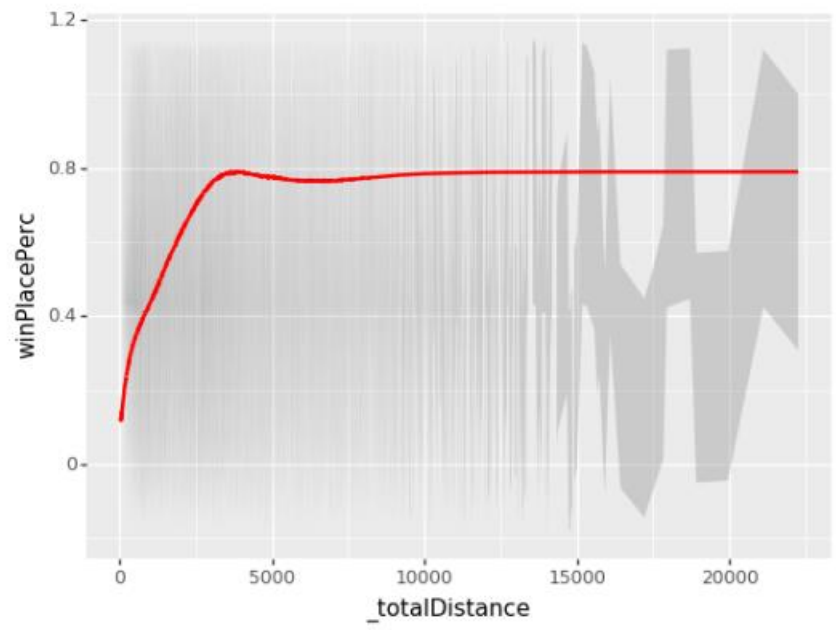
7. CONCLUSION

" PLAYERS WITH MORE THAN 10 KILLS AND HAVE TRAVELLED DISTANCE MORE THAN 15000 ARE PREDICTED TO BE WINNING THE GAME AND GAIN THE 1ST POSITION "

```
from plotnine import *
from pandas_summary import DataFrameSummary
x_all = get_sample(highly_corr, 100000)
ggplot(x_all, aes('_walkPerc_killsPerc','winPlacePerc'))+stat_smooth(se=True, colour='red', method='mavg')
```




```
# Plot the predictive quality of walkDistance
x_all = get_sample(highly_corr, 100000)
ggplot(x_all, aes('_totalDistance','winPlacePerc'))+stat_smooth(se=True, colour='red', method='mavg')
```



8. REFERENCES

https://www.kaggle.com/

https://machinelearningmastery.com/machine-learning-in-python-step-by-step/

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.hist.html#matplotlib.pyplot.hist

https://www.youtube.com/user/edurekaIN/playlists