BIG DATA PROJECT

ANALYZING PUBG GAME DATA

Player Unknown's Battle Grounds (PUBG)

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.

The Game

TRENDING

Beta Release: March 2017.

Worldwide Release: December 2017

MASS USERBASE

227 million monthly players, 87 million daily players, 400 million players till date

WORLD RECORD

World record for most simultaneous players at once

REVENUE

113 million monthly revenue, ~700k earning from daily user spending

MULTI PLATFORM

Available on Windows, Android, iOS, and Xbox

VIDEO

2.03 billion minutes of viewing on Twitch

Features of Our Data

4.5 MILLION ROWS

26 COLUMNS

47k+ MATCH DATA

1.88 MILLION GROUPS

3 STRING VARIABLES,
23 INTEGER VARIABLES

DATA SCOURCE: KAGGLE

METADATA

MATCHID

ID to identify match.

GROUPID

ID to identify a group within a match.

NUMGROUPS

Number of groups we have data for in the match.

MATCHTYPE

String identifying the game mode that the data comes from. The standard modes are “solo”, “duo”, “squad”

KILLS

Number of enemy players killed.

HEADSHOTKILLS

Number of enemy players killed with headshots.

VEHICLEDESTROYS

Number of vehicles destroyed.

WEAPONSACQUIRED

Number of weapons picked up.

HEALS

Number of healing items used.

REVIVES

Number of times this player revived teammates.

SWIMDISTANCE

Total distance traveled by swimming measured in meters.

WINPLACEPERC

This is a percentile winning placement, where 1 corresponds to 1st place, and 0 corresponds to last place in the match.

Approach

Enhancing the game experience using **insights**

“Does killing more people increases the chance of winning the game?”

APPROACH

Using the correlation between the match winning percentage and number of kills to determine the relationship between the two.

Columns Used

WINPLACEPERC, KILLS

Data Pre-processing

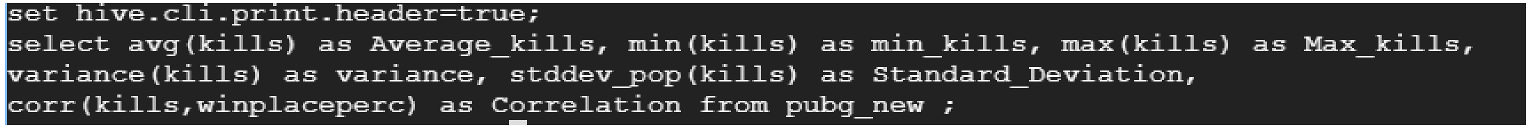
None

Tool Used

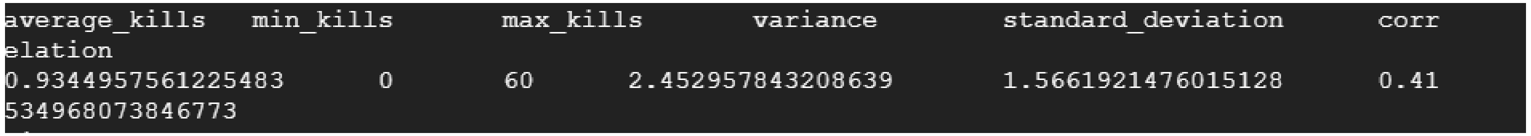
Hive

DATA ANALYSIS

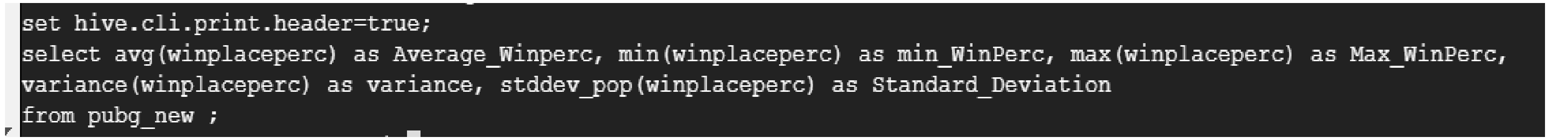
Data Preprocessing and Correlation



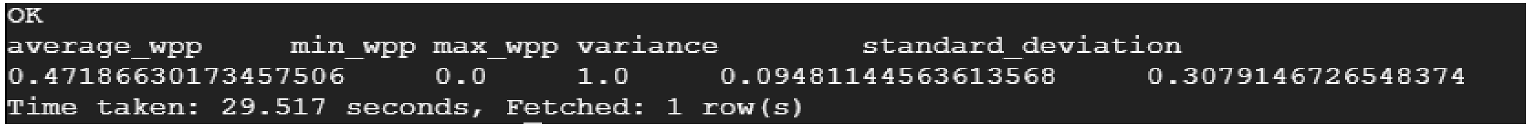
```
set hive.cli.print.header=true;
select avg(kills) as Average_kills, min(kills) as min_kills, max(kills) as Max_kills,
variance(kills) as variance, stddev_pop(kills) as Standard_Deviation,
corr(kills,winplaceperc) as Correlation from pubg_new ;
```



```
average_kills   min_kills        max_kills        variance          standard_deviation        corr
elation
0.9344957561225483      0        60       2.452957843208639        1.5661921476015128        0.41
534968073846773
```



```
set hive.cli.print.header=true;
select avg(winplaceperc) as Average_Winperc, min(winplaceperc) as min_WinPerc, max(winplaceperc) as Max_WinPerc,
variance(winplaceperc) as variance, stddev_pop(winplaceperc) as Standard_Deviation
from pubg_new ;
```



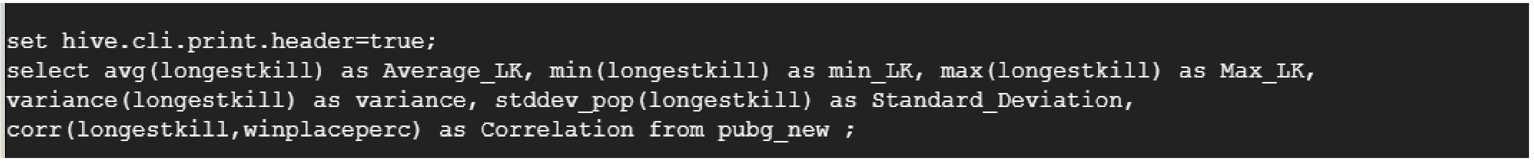
```
OK
average_wpp     min_wpp max_wpp variance        standard_deviation
0.47186630173457506     0.0     1.0     0.09481144563613568     0.3079146726548374
Time taken: 29.517 seconds, Fetched: 1 row(s)
```

	WINPLACEPERC	KILLS
Max	1	60
Min	0	0
Average	0.47	0.93
Standard Deviation	0.31	1.56
Variance	0.09	2.45
Missing Values	0	0
Correlation	**0.4153**	

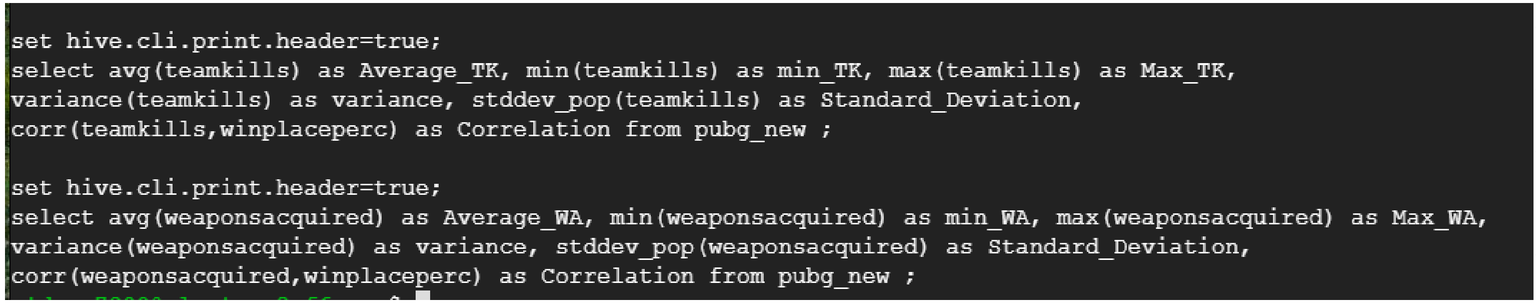
The correlation suggests that the relation between match winning percentage and number of kills is positive and suggests that more killing directly impact the winning placement. However, the correlation is not strong and is not a significant predictor of match winning.

DATA ANALYSIS

Data Preprocessing

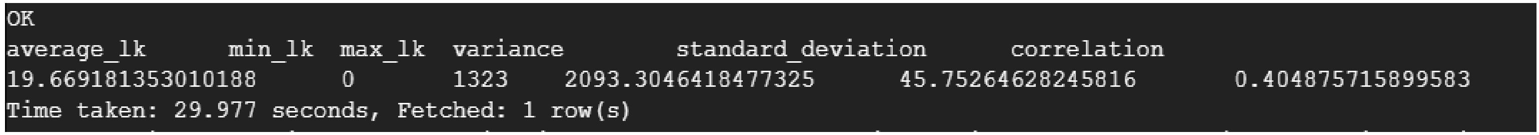


```
set hive.cli.print.header=true;
select avg(longestkill) as Average_LK, min(longestkill) as min_LK, max(longestkill) as Max_LK,
variance(longestkill) as variance, stddev_pop(longestkill) as Standard_Deviation,
corr(longestkill,winplaceperc) as Correlation from pubg_new ;
```

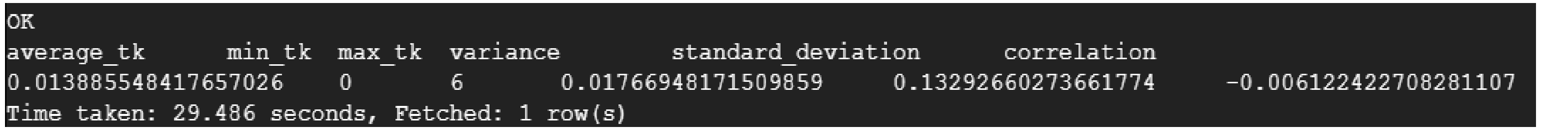


```
set hive.cli.print.header=true;
select avg(teamkills) as Average_TK, min(teamkills) as min_TK, max(teamkills) as Max_TK,
variance(teamkills) as variance, stddev_pop(teamkills) as Standard_Deviation,
corr(teamkills,winplaceperc) as Correlation from pubg_new ;

set hive.cli.print.header=true;
select avg(weaponsacquired) as Average_WA, min(weaponsacquired) as min_WA, max(weaponsacquired) as Max_WA,
variance(weaponsacquired) as variance, stddev_pop(weaponsacquired) as Standard_Deviation,
corr(weaponsacquired,winplaceperc) as Correlation from pubg_new ;
```

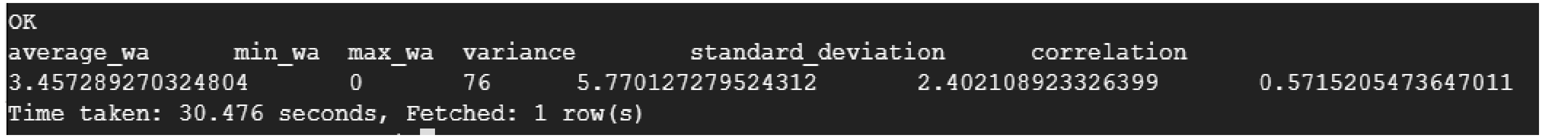
	LONGEST KILL	TEAM KILLS	WEAPONS ACQUIRED
Max	1323	6	76
Min	0	0	0
Average	19.66	0.013	3.45
Standard Deviation	45.75	0.13	2.40
Variance	2093.30	0.017	5.77
Missing Values	0	0	0
Correlation with win percentage	**0.40**	**-0.006**	**0.57**



```
OK
average_lk      min_lk  max_lk  variance        standard_deviation      correlation
19.669181353010188      0       1323    2093.3046418477325      45.75264628245816       0.404875715899583
Time taken: 29.977 seconds, Fetched: 1 row(s)
```



```
OK
average_tk      min_tk  max_tk  variance        standard_deviation      correlation
0.013885548417657026    0       6       0.01766948171509859     0.13292660273661774     -0.006122422708281107
Time taken: 29.486 seconds, Fetched: 1 row(s)
```



```
OK
average_wa      min_wa  max_wa  variance        standard_deviation      correlation
3.457289270324804       0       76      5.770127279524312       2.402108923326399       0.5715205473647011
Time taken: 30.476 seconds, Fetched: 1 row(s)
```

“Can we predict the finishing position of a player in the game?”

APPROACH

Regression Problem: design and test a model using regression algorithms to predict the final position of the player at the end of the game.

Columns Used

WINPLACEPERC, All Columns

Data Pre-processing

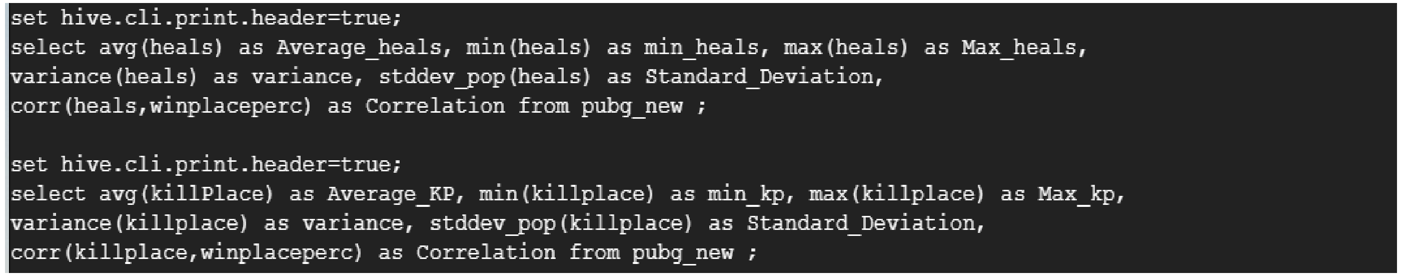
Data standardization

Tool Used

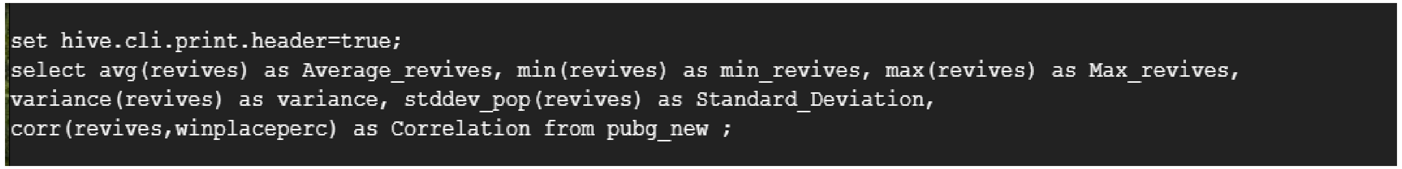
Hive, Spark

DATA ANALYSIS

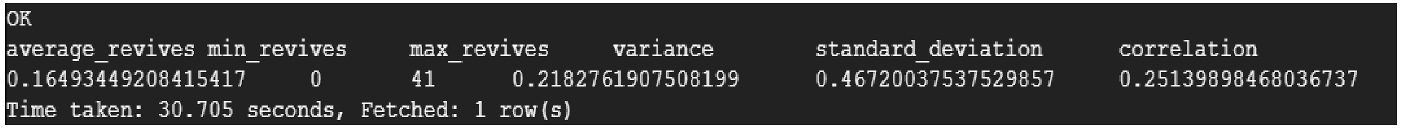
Data Preprocessing



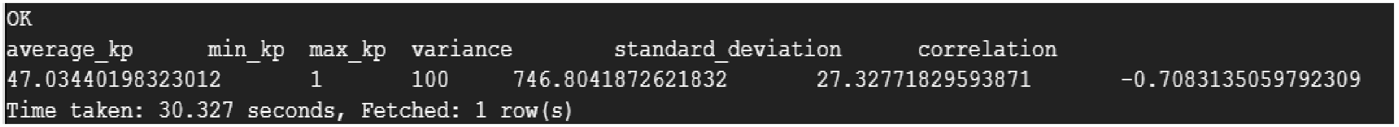
```
set hive.cli.print.header=true;
select avg(heals) as Average_heals, min(heals) as min_heals, max(heals) as Max_heals,
variance(heals) as variance, stddev_pop(heals) as Standard_Deviation,
corr(heals,winplaceperc) as Correlation from pubg_new ;

set hive.cli.print.header=true;
select avg(killPlace) as Average_KP, min(killplace) as min_kp, max(killplace) as Max_kp,
variance(killplace) as variance, stddev_pop(killplace) as Standard_Deviation,
corr(killplace,winplaceperc) as Correlation from pubg_new ;
```



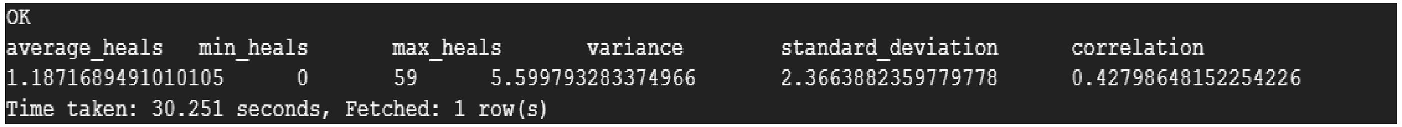
```
set hive.cli.print.header=true;
select avg(revives) as Average_revives, min(revives) as min_revives, max(revives) as Max_revives,
variance(revives) as variance, stddev_pop(revives) as Standard_Deviation,
corr(revives,winplaceperc) as Correlation from pubg_new ;
```



```
OK
average_revives min_revives     max_revives     variance        standard_deviation      correlation
0.16493449208415417     0       41      0.2182761907508199      0.46720037537529857     0.25139898468036737
Time taken: 30.705 seconds, Fetched: 1 row(s)
```



```
OK
average_kp      min_kp  max_kp  variance        standard_deviation      correlation
47.03440198323012       1       100     746.8041872621832       27.32771829593871       -0.7083135059792309
Time taken: 30.327 seconds, Fetched: 1 row(s)
```

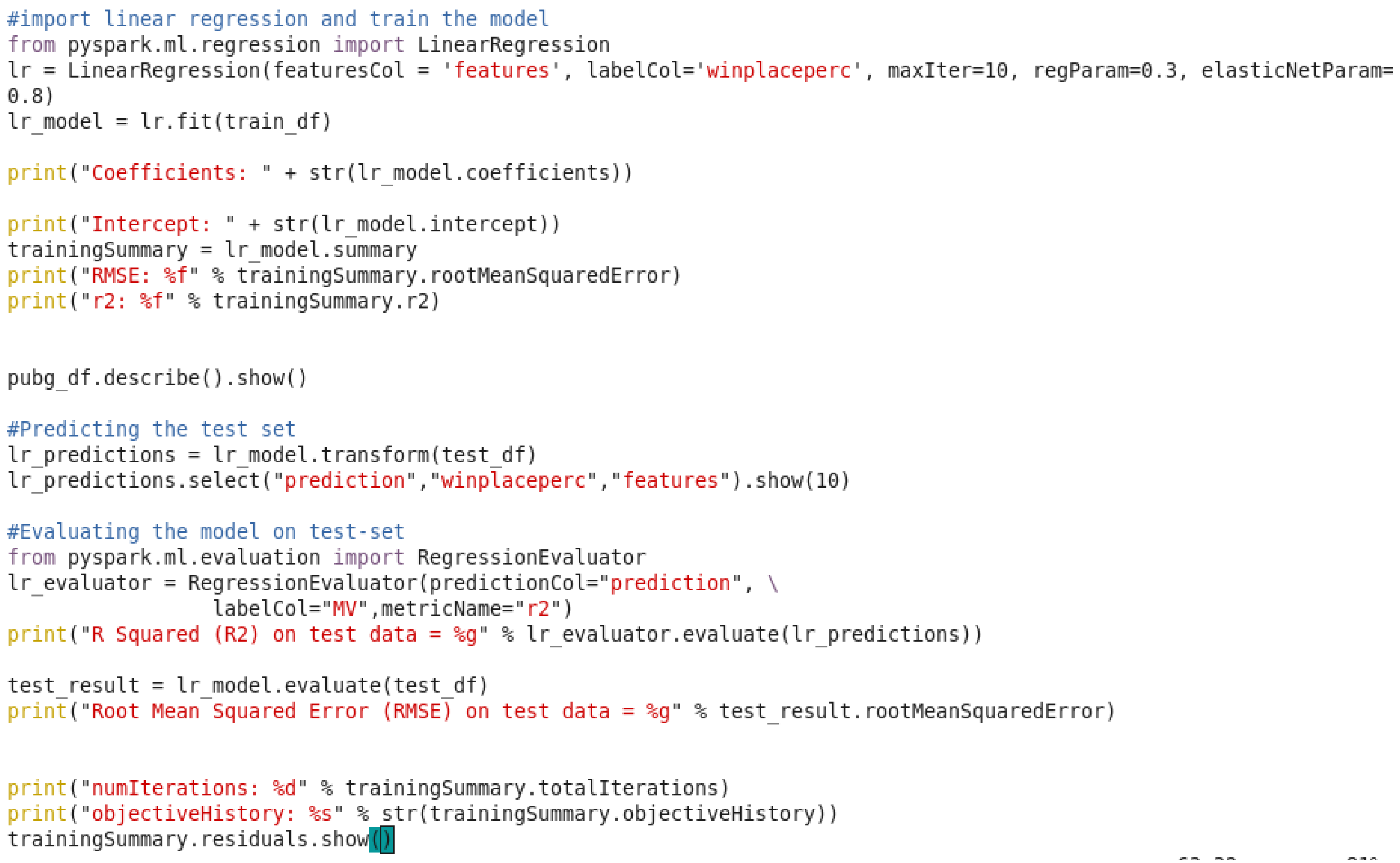


```
OK
average_heals   min_heals       max_heals       variance        standard_deviation      correlation
1.1871689491010105      0       59      5.599793283374966       2.3663882359779778      0.42798648152254226
Time taken: 30.251 seconds, Fetched: 1 row(s)
```

	HEALS	KILLPLACE	REVIVES
Max	59	100	41
Min	0	1	0
Average	1.18	47.03	0.16
Standard Deviation	2.36	27.32	0.47
Variance	5.59	746.80	0.22
Missing Values	0	0	0
Correlation with win percentage	**0.43**	**-0.71**	**0.25**

DATA ANALYSIS

Linear Regression Code

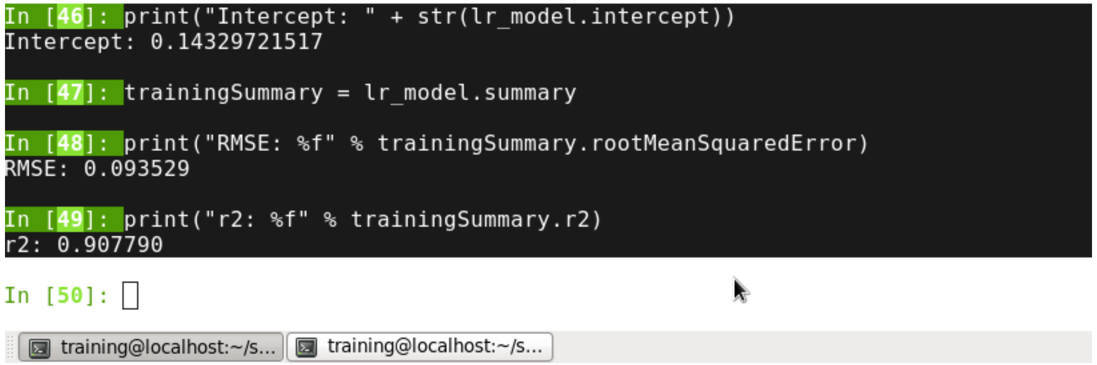


```
#import linear regression and train the model
from pyspark.ml.regression import LinearRegression
lr = LinearRegression(featuresCol = 'features', labelCol='winplaceperc', maxIter=10, regParam=0.3, elasticNetParam=
0.8)
lr_model = lr.fit(train_df)

print("Coefficients: " + str(lr_model.coefficients))

print("Intercept: " + str(lr_model.intercept))
trainingSummary = lr_model.summary
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("r2: %f" % trainingSummary.r2)


pubg_df.describe().show()

#Predicting the test set
lr_predictions = lr_model.transform(test_df)
lr_predictions.select("prediction","winplaceperc","features").show(10)

#Evaluating the model on test-set
from pyspark.ml.evaluation import RegressionEvaluator
lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
                 labelCol="MV",metricName="r2")
print("R Squared (R2) on test data = %g" % lr_evaluator.evaluate(lr_predictions))

test_result = lr_model.evaluate(test_df)
print("Root Mean Squared Error (RMSE) on test data = %g" % test_result.rootMeanSquaredError)


print("numIterations: %d" % trainingSummary.totalIterations)
print("objectiveHistory: %s" % str(trainingSummary.objectiveHistory))
trainingSummary.residuals.show()
```

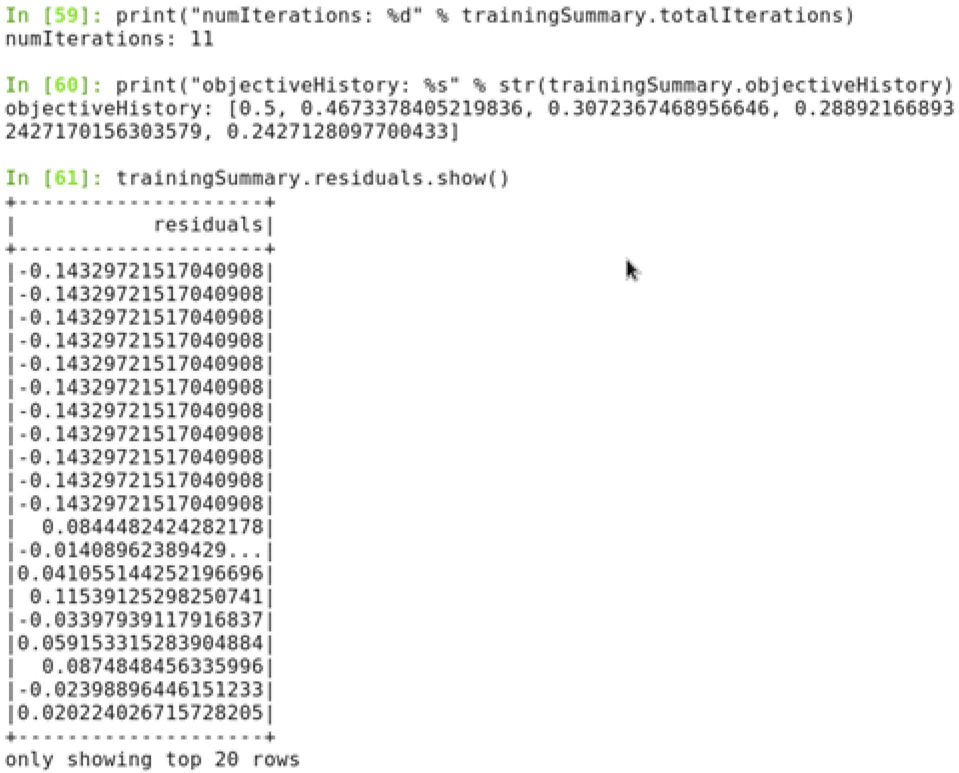
Running linear regression on the entire dataset to develop a model to predict the winning percentage based on various parameters.

DATA ANALYSIS

Linear Regression Output



```
In [46]: print("Intercept: " + str(lr_model.intercept))
Intercept: 0.14329721517

In [47]: trainingSummary = lr_model.summary

In [48]: print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
RMSE: 0.093529

In [49]: print("r2: %f" % trainingSummary.r2)
r2: 0.907790

In [50]:
```



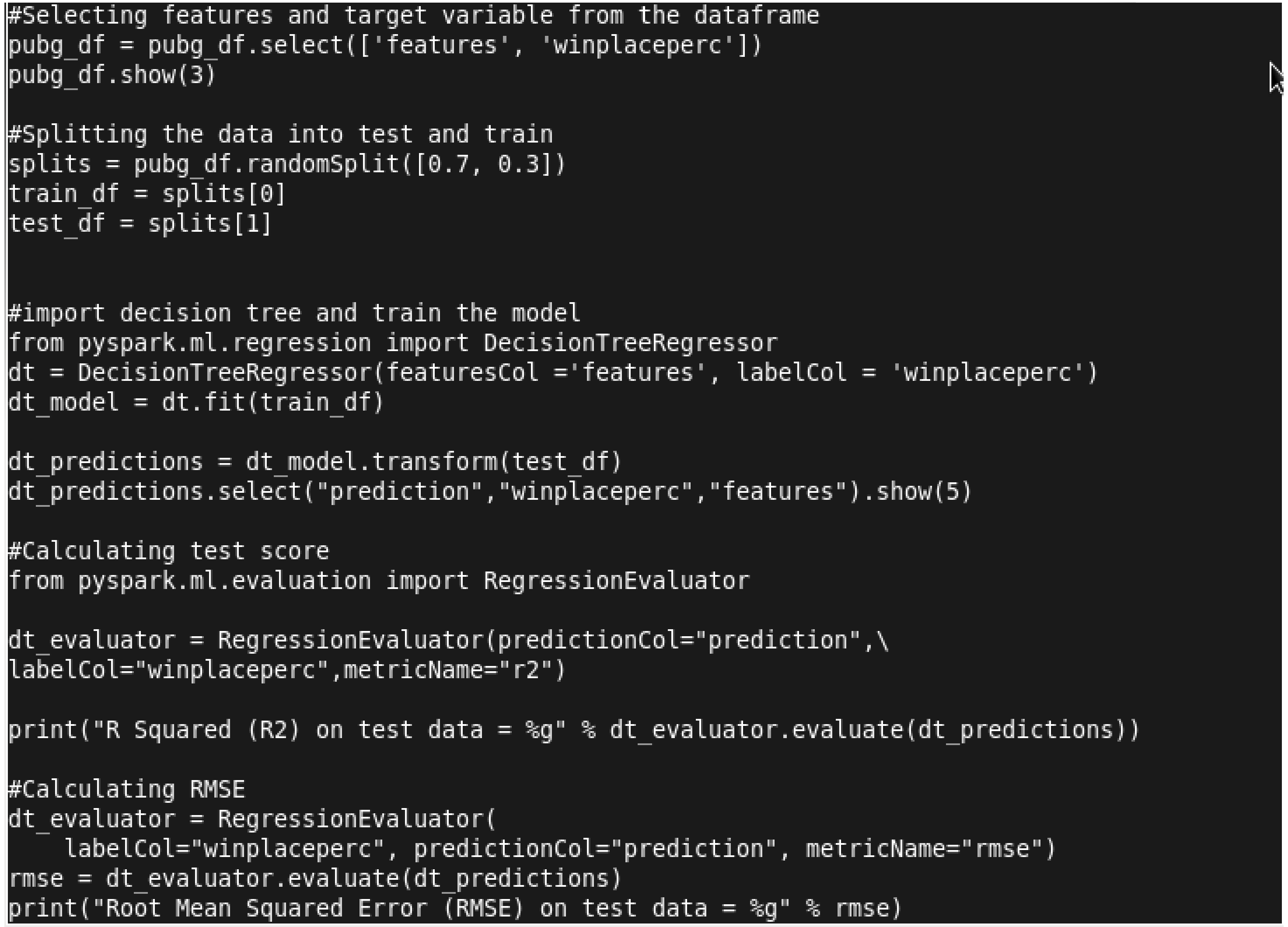
```
In [59]: print("numIterations: %d" % trainingSummary.totalIterations)
numIterations: 11

In [60]: print("objectiveHistory: %s" % str(trainingSummary.objectiveHistory)
objectiveHistory: [0.5, 0.4673378405219836, 0.3072367468956646, 0.28892166893
2427170156303579, 0.2427128097700433]

In [61]: trainingSummary.residuals.show()
+--------------------+
|           residuals|
+--------------------+
|-0.14329721517040908|
|-0.14329721517040908|
|-0.14329721517040908|
|-0.14329721517040908|
|-0.14329721517040908|
|-0.14329721517040908|
|-0.14329721517040908|
|-0.14329721517040908|
|-0.14329721517040908|
|-0.14329721517040908|
|-0.14329721517040908|
|  0.0844482424282178|
|-0.01408962389429...|
|0.041055144252196696|
| 0.11539125298250741|
|-0.03397939117916837|
|0.059153315283904884|
|  0.0874848456335996|
|-0.02398896446151233|
|0.020224026715728205|
+--------------------+
only showing top 20 rows
```

The regression model provides with an R Square of 0.90

Based on the predictors we have included in the data we can say that the position of a player can be accurately predicted based on the various game parameters.

DATA ANALYSIS

Decision Tree Code

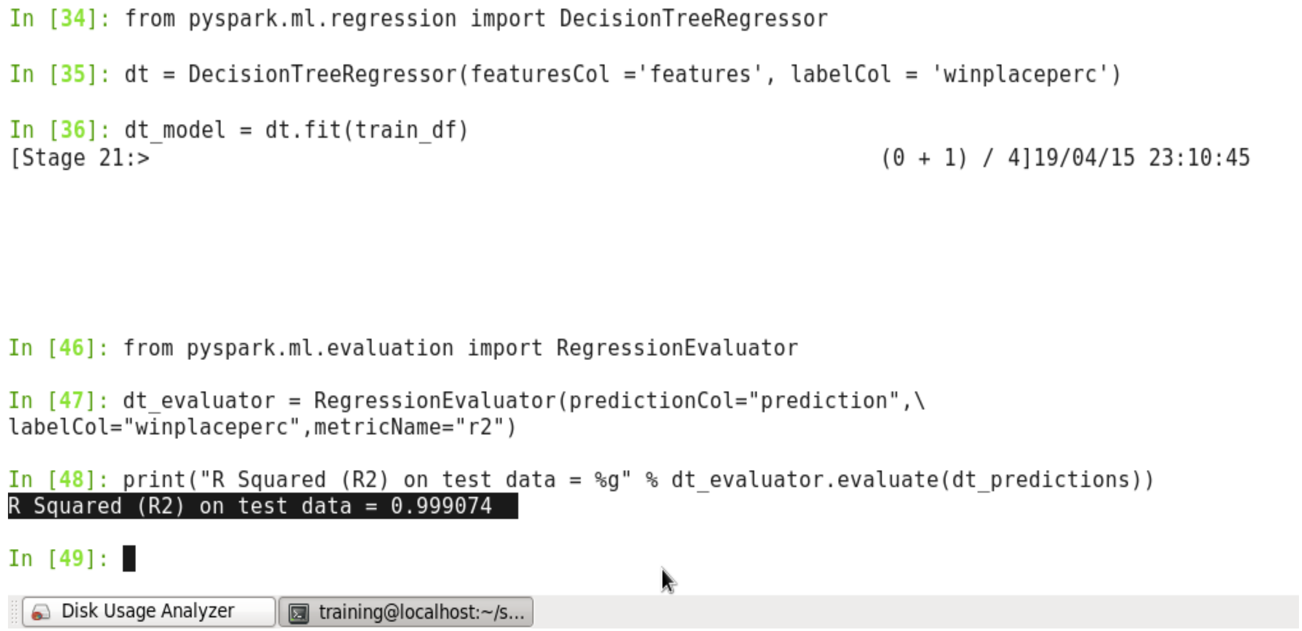


```
#Selecting features and target variable from the dataframe
pubg_df = pubg_df.select(['features', 'winplaceperc'])
pubg_df.show(3)

#Splitting the data into test and train
splits = pubg_df.randomSplit([0.7, 0.3])
train_df = splits[0]
test_df = splits[1]


#import decision tree and train the model
from pyspark.ml.regression import DecisionTreeRegressor
dt = DecisionTreeRegressor(featuresCol ='features', labelCol = 'winplaceperc')
dt_model = dt.fit(train_df)

dt_predictions = dt_model.transform(test_df)
dt_predictions.select("prediction","winplaceperc","features").show(5)

#Calculating test score
from pyspark.ml.evaluation import RegressionEvaluator

dt_evaluator = RegressionEvaluator(predictionCol="prediction",\
labelCol="winplaceperc",metricName="r2")

print("R Squared (R2) on test data = %g" % dt_evaluator.evaluate(dt_predictions))

#Calculating RMSE
dt_evaluator = RegressionEvaluator(
    labelCol="winplaceperc", predictionCol="prediction", metricName="rmse")
rmse = dt_evaluator.evaluate(dt_predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

Running Decision Tree regression on the entire dataset to develop a model to predict the winning percentage based on various parameters.

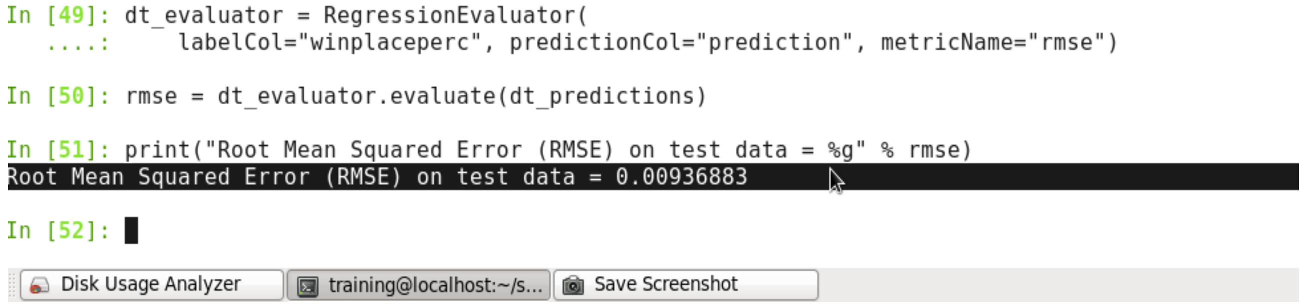
DATA ANALYSIS

Decision Tree Output



```
In [34]: from pyspark.ml.regression import DecisionTreeRegressor

In [35]: dt = DecisionTreeRegressor(featuresCol ='features', labelCol = 'winplaceperc')

In [36]: dt_model = dt.fit(train_df)
[Stage 21:>                                                   (0 + 1) / 4]19/04/15 23:10:45
```

```
In [46]: from pyspark.ml.evaluation import RegressionEvaluator

In [47]: dt_evaluator = RegressionEvaluator(predictionCol="prediction",\
labelCol="winplaceperc",metricName="r2")

In [48]: print("R Squared (R2) on test data = %g" % dt_evaluator.evaluate(dt_predictions))
R Squared (R2) on test data = 0.999074

In [49]: 
```



```
In [49]: dt_evaluator = RegressionEvaluator(
   ....:     labelCol="winplaceperc", predictionCol="prediction", metricName="rmse")

In [50]: rmse = dt_evaluator.evaluate(dt_predictions)

In [51]: print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
Root Mean Squared Error (RMSE) on test data = 0.00936883

In [52]: 
```

The regression model provides with an R Square of 0.99

Based on the predictors we have included in the data we can say that the position of a player can be accurately predicted based on the various game parameters.

“How do we catch the cheaters in the game?”

APPROACH

Using various logical conditions based on game knowledge to determine cheaters in the game.

Columns Used

WINPLACEPERC, KILLS, RIDE DISTANCE, WALK DISTANCE

Data Pre-processing

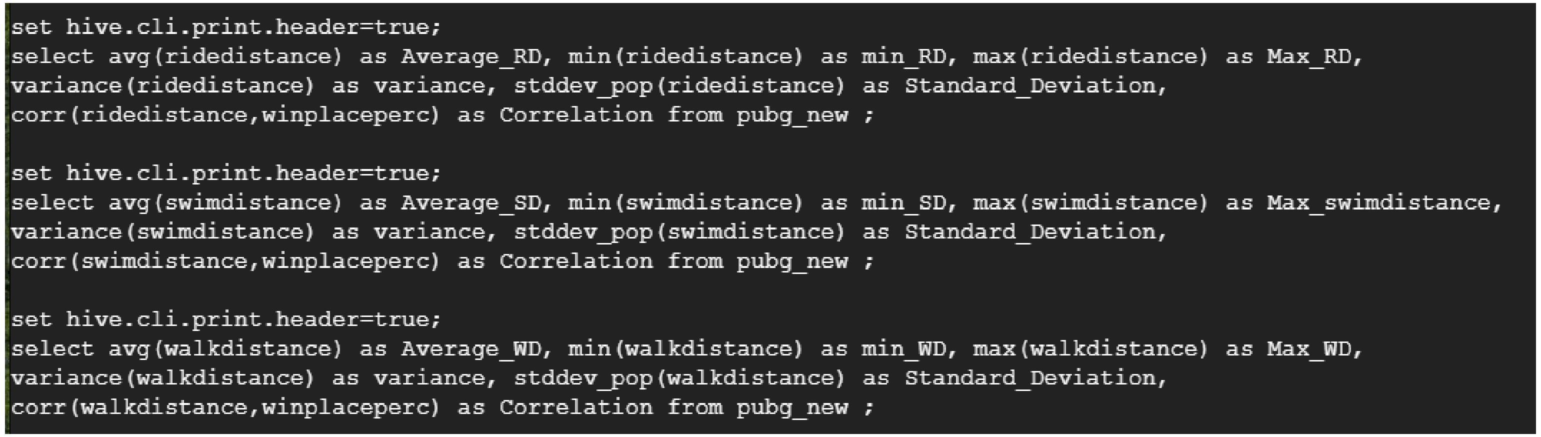
None

Tool Used

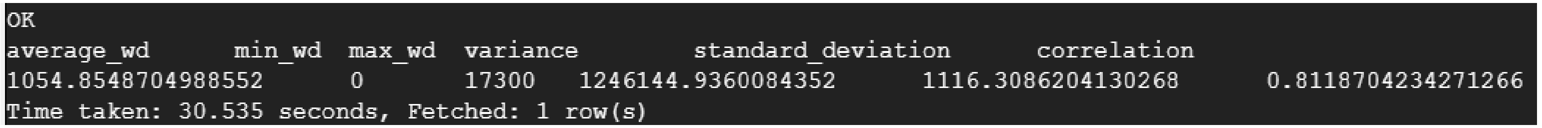
Hive

DATA ANALYSIS

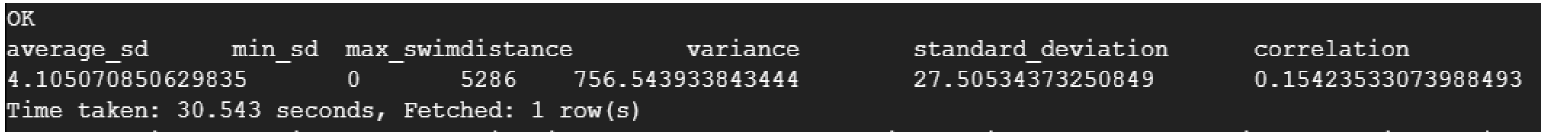
Data Preprocessing



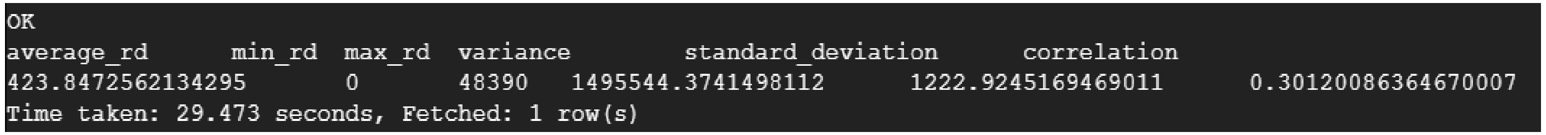
```
set hive.cli.print.header=true;
select avg(ridedistance) as Average_RD, min(ridedistance) as min_RD, max(ridedistance) as Max_RD,
variance(ridedistance) as variance, stddev_pop(ridedistance) as Standard_Deviation,
corr(ridedistance,winplaceperc) as Correlation from pubg_new ;

set hive.cli.print.header=true;
select avg(swimdistance) as Average_SD, min(swimdistance) as min_SD, max(swimdistance) as Max_swimdistance,
variance(swimdistance) as variance, stddev_pop(swimdistance) as Standard_Deviation,
corr(swimdistance,winplaceperc) as Correlation from pubg_new ;

set hive.cli.print.header=true;
select avg(walkdistance) as Average_WD, min(walkdistance) as min_WD, max(walkdistance) as Max_WD,
variance(walkdistance) as variance, stddev_pop(walkdistance) as Standard_Deviation,
corr(walkdistance,winplaceperc) as Correlation from pubg_new ;
```



```
OK
average_wd      min_wd  max_wd  variance        standard_deviation      correlation
1054.8548704988552      0       17300   1246144.9360084352      1116.3086204130268      0.8118704234271266
Time taken: 30.535 seconds, Fetched: 1 row(s)
```



```
OK
average_sd      min_sd  max_swimdistance        variance        standard_deviation      correlation
4.105070850629835       0       5286    756.543933843444        27.50534373250849       0.15423533073988493
Time taken: 30.543 seconds, Fetched: 1 row(s)
```

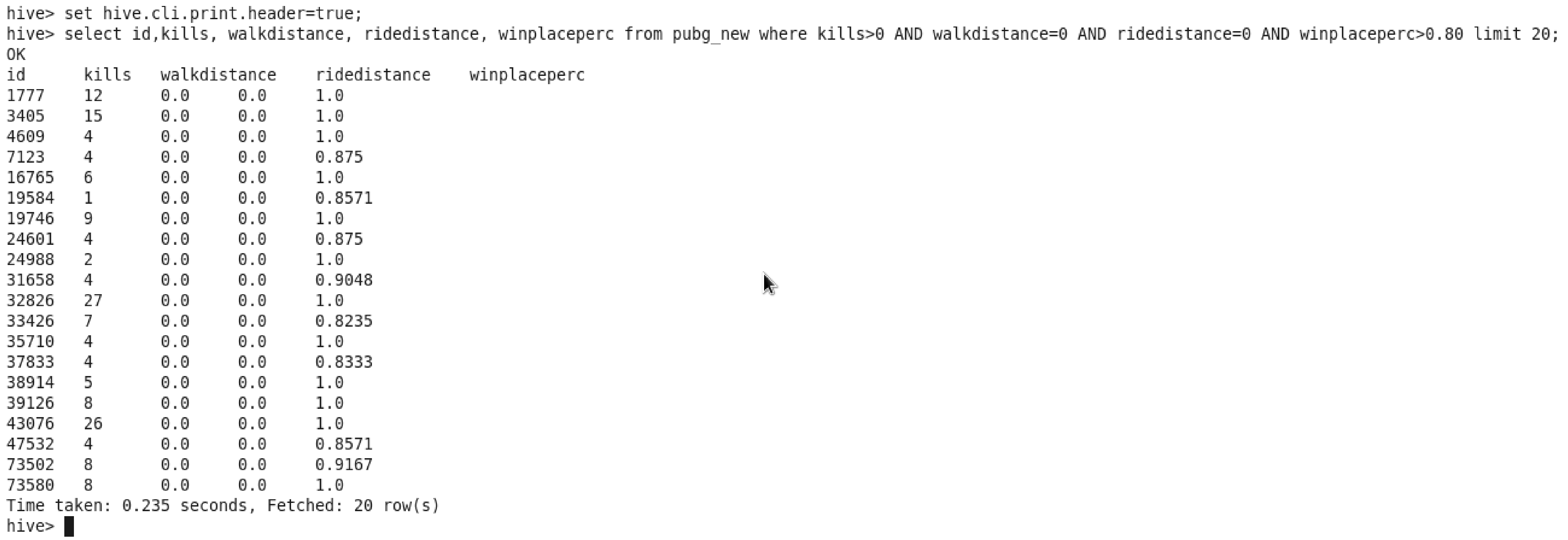


```
OK
average_rd      min_rd  max_rd  variance        standard_deviation      correlation
423.8472562134295       0       48390   1495544.3741498112      1222.9245169469011      0.30120086364670007
Time taken: 29.473 seconds, Fetched: 1 row(s)
```

	RIDE DISTANCE	SWIM DISTANCE	WALK DISTANCE
Max	48390	5286	17300
Min	0	0	0
Average	423.84	4.11	1054.85
Standard Deviation	1222.92	27.50	1116.30
Variance	1495544	756.54	1246144
Missing Values	0	0	0
Correlation with win percentage	**0.30**	**0.15**	**0.81**

DATA ANALYSIS

Finding cheaters using logical conditions

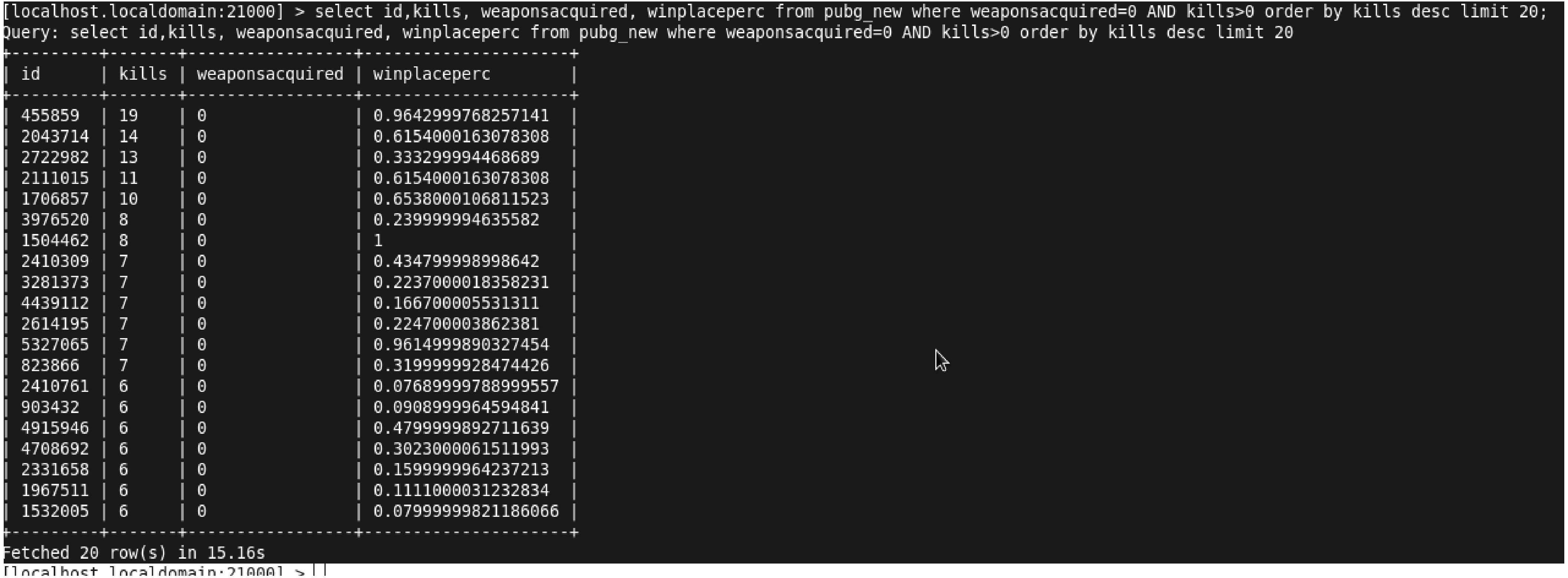


```
hive> set hive.cli.print.header=true;
hive> select id,kills, walkdistance, ridedistance, winplaceperc from pubg_new where kills>0 AND walkdistance=0 AND ridedistance=0 AND winplaceperc>0.80 limit 20;
OK
id      kills   walkdistance    ridedistance    winplaceperc
1777    12      0.0     0.0     1.0
3405    15      0.0     0.0     1.0
4609    4       0.0     0.0     1.0
7123    4       0.0     0.0     0.875
16765   6       0.0     0.0     1.0
19584   1       0.0     0.0     0.8571
19746   9       0.0     0.0     1.0
24601   4       0.0     0.0     0.875
24988   2       0.0     0.0     1.0
31658   4       0.0     0.0     0.9048
32826   27      0.0     0.0     1.0
33426   7       0.0     0.0     0.8235
35710   4       0.0     0.0     1.0
37833   4       0.0     0.0     0.8333
38914   5       0.0     0.0     1.0
39126   8       0.0     0.0     1.0
43076   26      0.0     0.0     1.0
47532   4       0.0     0.0     0.8571
73502   8       0.0     0.0     0.9167
73580   8       0.0     0.0     1.0
Time taken: 0.235 seconds, Fetched: 20 row(s)
hive> 
```

The game has been designed in such a way that walking is an essential part of playing the game. It is impossible that any player that is killing someone is not walking at all. The users that fall in these conditions are probably cheating or it is a glitch in the data.

DATA ANALYSIS

Finding cheaters using logical conditions

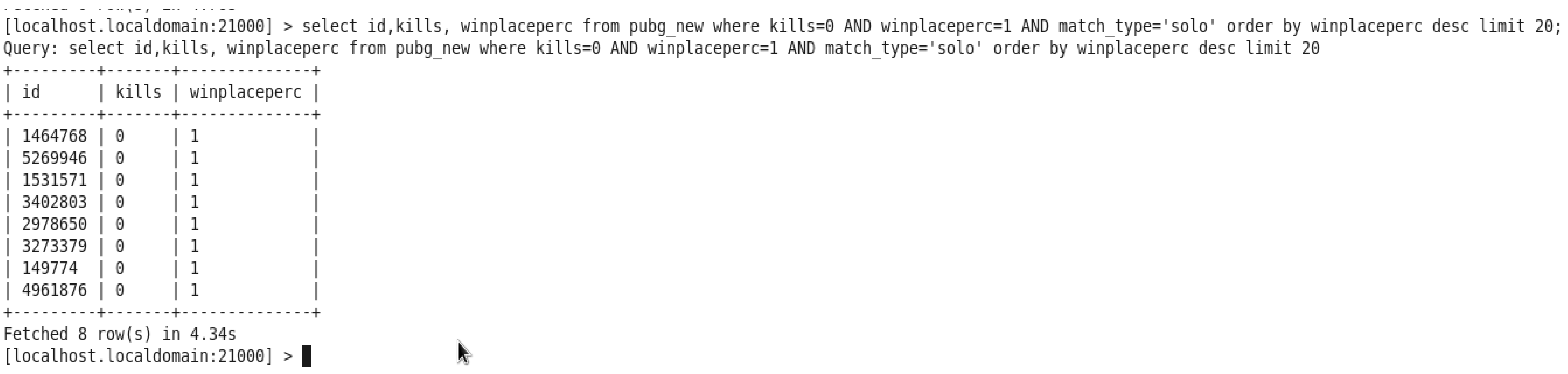


```
[localhost.localdomain:21000] > select id,kills, weaponsacquired, winplaceperc from pubg_new where weaponsacquired=0 AND kills>0 order by kills desc limit 20;
Query: select id,kills, weaponsacquired, winplaceperc from pubg_new where weaponsacquired=0 AND kills>0 order by kills desc limit 20
+---------+-------+-----------------+---------------------+
| id      | kills | weaponsacquired | winplaceperc        |
+---------+-------+-----------------+---------------------+
| 455859  | 19    | 0               | 0.9642999768257141  |
| 2043714 | 14    | 0               | 0.6154000163078308  |
| 2722982 | 13    | 0               | 0.333299994468689   |
| 2111015 | 11    | 0               | 0.6154000163078308  |
| 1706857 | 10    | 0               | 0.6538000106811523  |
| 3976520 | 8     | 0               | 0.239999994635582   |
| 1504462 | 8     | 0               | 1                   |
| 2410309 | 7     | 0               | 0.434799998998642   |
| 3281373 | 7     | 0               | 0.2237000018358231  |
| 4439112 | 7     | 0               | 0.166700005531311   |
| 2614195 | 7     | 0               | 0.224700003862381   |
| 5327065 | 7     | 0               | 0.9614999890327454  |
| 823866  | 7     | 0               | 0.3199999928474426  |
| 2410761 | 6     | 0               | 0.07689999788999557 |
| 903432  | 6     | 0               | 0.0908999964594841  |
| 4915946 | 6     | 0               | 0.4799999892711639  |
| 4708692 | 6     | 0               | 0.3023000061511993  |
| 2331658 | 6     | 0               | 0.1599999964237213  |
| 1967511 | 6     | 0               | 0.1111000031232834  |
| 1532005 | 6     | 0               | 0.07999999821186066 |
+---------+-------+-----------------+---------------------+
Fetched 20 row(s) in 15.16s
[localhost.localdomain:21000] >
```

The game is essentially won by killing more people and being the last one standing. weapons help the players do that. It is nearly impossible to kill so many people and come close to winning without acquiring any weapons. These players appear to be cheaters in the game.

DATA ANALYSIS

Finding cheaters using logical conditions



```
[localhost.localdomain:21000] > select id,kills, winplaceperc from pubg_new where kills=0 AND winplaceperc=1 AND match_type='solo' order by winplaceperc desc limit 20;
Query: select id,kills, winplaceperc from pubg_new where kills=0 AND winplaceperc=1 AND match_type='solo' order by winplaceperc desc limit 20
+---------+-------+--------------+
| id      | kills | winplaceperc |
+---------+-------+--------------+
| 1464768 | 0     | 1            |
| 5269946 | 0     | 1            |
| 1531571 | 0     | 1            |
| 3402803 | 0     | 1            |
| 2978650 | 0     | 1            |
| 3273379 | 0     | 1            |
| 149774  | 0     | 1            |
| 4961876 | 0     | 1            |
+---------+-------+--------------+
Fetched 8 row(s) in 4.34s
[localhost.localdomain:21000] >
```

The game ends by killing the last opponent playing the game. the last person has to kill a person to end the game. Thus, the players who are winning the game without killing anyone are likely to be cheaters.