PUBG Win Points Prediction with Spark

Ailun Zhu

az8ec

Motivation – Business Interest

eSports

the Next Billion-dollar Industry

2 billion
By 2020

696 million
In 2018

385 million
In 2018

PUBG

Most Popular Game in 2017

2 million
Download/month

+500 million
revenue

How Data Science Is

Revolutionizing the Future of eSports

- ❑ Identify players' strengths
- ❑ Predict opponents' strategies
- ❑ Build up dream teams
- ❑ Attract capital and talent
- ❑ Online gambling and betting

Code Snippet

Load Data & Visualization

```
bucket='odl-spark19spds6003-001/az8ec'
data_key = 'small train.csv'
data_location = 's3://{}/{}'.format(bucket, data_key)
pd.read_csv(data_location)
```

```
df_subset=df.sample(n=10000)
```

```
df_subset.hist()
plt.show()
```

Machine Learning Pipeline

```
print("Pearson's r(killPlace, killPoints) = {}".format(df.corr("killPlace", "killPoints"))
print("Pearson's r(killPlace,kills) = {}".format(df.corr("killPlace", "kills")))
print("Pearson's r(killPlace,killStreaks) = {}".format(df.corr("killPlace", "killStreaks")
print("Pearson's r(killPlace, longestKill) = {}".format(df.corr("killPlace", "longestKill"
```

```
df=df.select("killPoints", "winPoints")
```

```
seed = 42
(testDF, trainingDF) = df.randomSplit((0.20, 0.80), seed=seed)
print ('training set N = {}, test set N = {}'.format(trainingDF.count(),testDF.count()))
```

Vectorization

```
from pyspark.ml.linalg import Vectors, VectorUDT
```

```
sqlc.registerFunction("oneElementVec", lambda d: Vectors.dense([d]), returnType=VectorUDT(
trainingDF = trainingDF.selectExpr("killPoints", "oneElementVec(winPoints) as winPoints")
testDF = testDF.selectExpr("killPoints", "oneElementVec(winPoints) as winPoints")

print(testDF.orderBy(testDF.killPoints.desc()).limit(5))
```

```
trainingDF = trainingDF.withColumnRenamed("killPoints", "label").withColumnRenamed("winPoi
testDF = testDF.withColumnRenamed("killPoints", "label").withColumnRenamed("winPoints", "f
```

Modeling

```
from pyspark.ml.regression import LinearRegression, LinearRegressionModel
lr = LinearRegression()
lrModel = lr.fit(trainingDF)
```

```
type(lrModel)
```

```
predictionsAndLabelsDF = lrModel.transform(testDF)
print(predictionsAndLabelsDF.orderBy(predictionsAndLabelsDF.label.desc()).take(5))
```

Visualization

In [13]:
```
df_subset.hist()
plt.show()
```