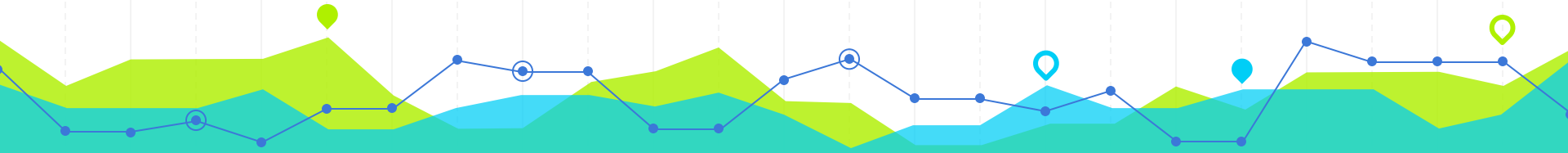
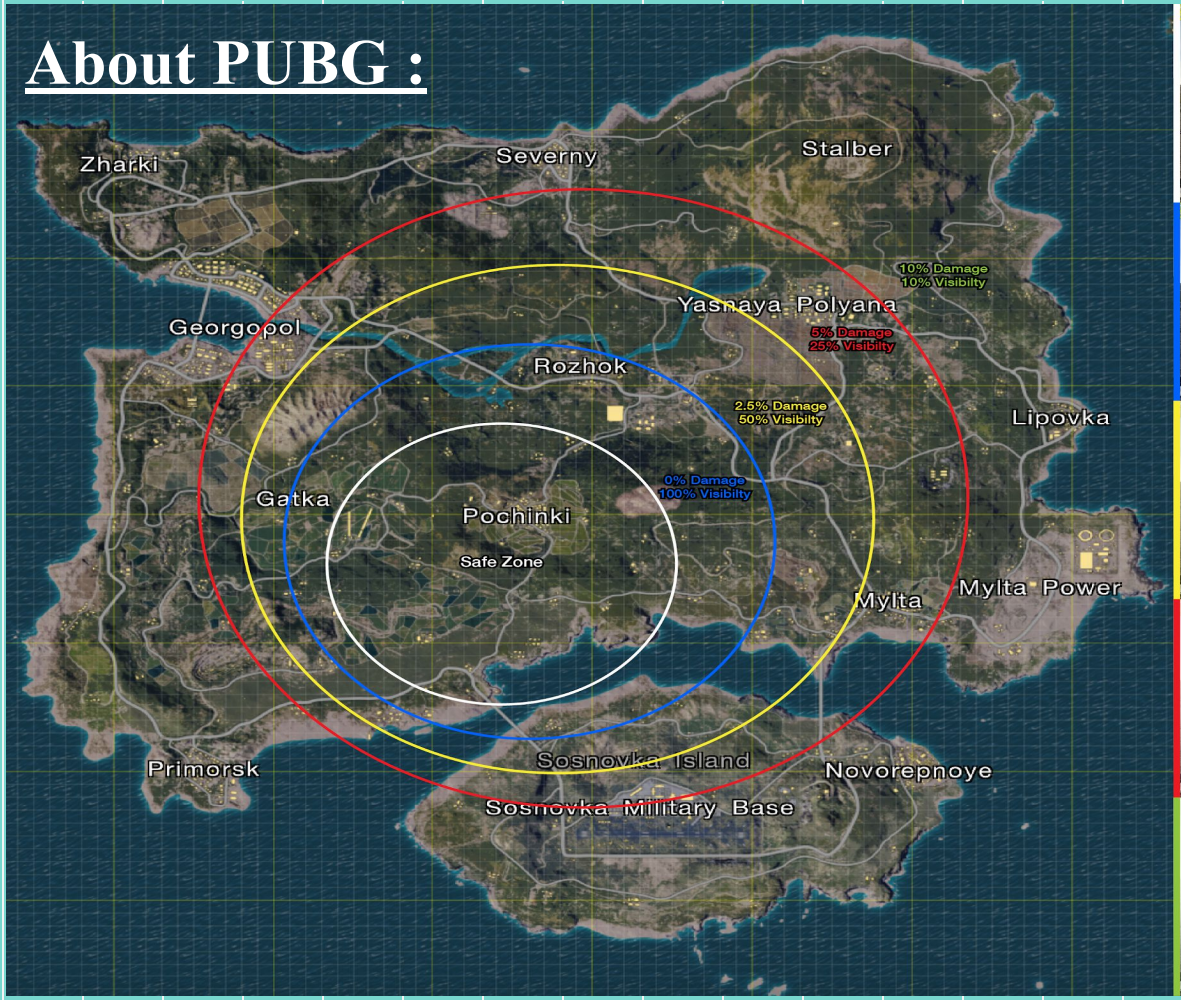


PlayerUnknown's Battlegrounds Data Analysis Using Spark

Team 6: Beiwen Guo
Zhenyu Zhu

About PUBG :


Zharki
Severny
Stalber
10% Damage
10% Visibilty
Yasnaya Polyana
Georgopol
5% Damage
25% Visibilty
Rozhok
2.5% Damage
50% Visibilty
Lipovka
0% Damage
100% Visibilty
Gatka
Pochinki
Safe Zone
Mylta
Mylta Power
Primorsk
Sosnovka Island
Novorepnoye
Sosnovka Military Base

Our Goals

- Processing and visualizing matches data from the popular video game PUBG to determine if specific strategies or behaviors are related to winning the game.
- Gain more hands-on experience of coding with Scala.
- Learn to use Spark to build big data engineer project.
- With the assistance of Spark and Scala, train our skills of ingesting and processing data with cloud resources.

Use Case 1

Actor : Business User

Input :

Matches data (csv format)

Output :

DataFrame with clustering results.

Plots helping understanding data.

All players will be divided into several groups according to their travel preference and battle performance. We trained k means model within dataset we got from kaggle. Models can be fastly retrieved from AWS S3 bucket.

Use Case 2

Actor : Business User

Input :

Matches data (csv format)

Output :

DataFrame with prediction indicating if such a player is a winner in a single game.

Predictions will be made by our classifier which was built based on multilayer perceptron classifier from Spark MLlib.

Methodology

Filtering data

Discard data which will not be used and ingest data into an available format.

Spark MLlib

Building and training model using Spark MLlib.

AWS

Using AWS resources to process large scale data.

Apache Zeppelin

Using Apache Zeppelin to generate visualization output.

Hortonworks Data Cloud

With assistance of HDCloud, configure computing resources we need.

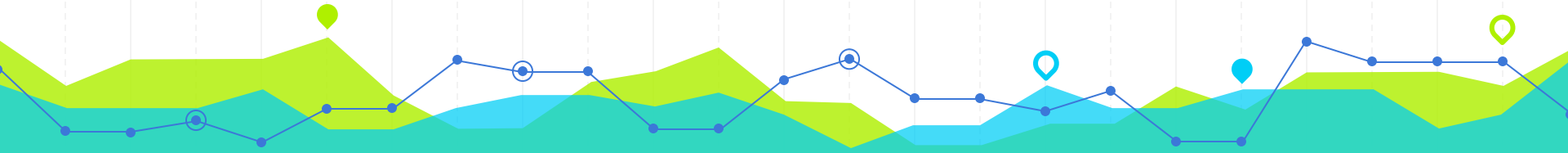
AWS & HORTONWORKS DATA CLOUD

Cloud Resource

Computing Cluster :Since Spark provides us with a cluster-computing framework and the data amount we were trying to processing is too big for our laptops. We deployed a spark cluster on AWS infrastructure with the assistance of HDC.

S3 Bucket: Reliable storage solution for our trained models.

	vCPU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
c4.4xlarge	16	30	EBS Only	$0.796 per Hour
c4.xlarge	4	7.5	EBS Only	$0.199 per Hour



Clustering Model

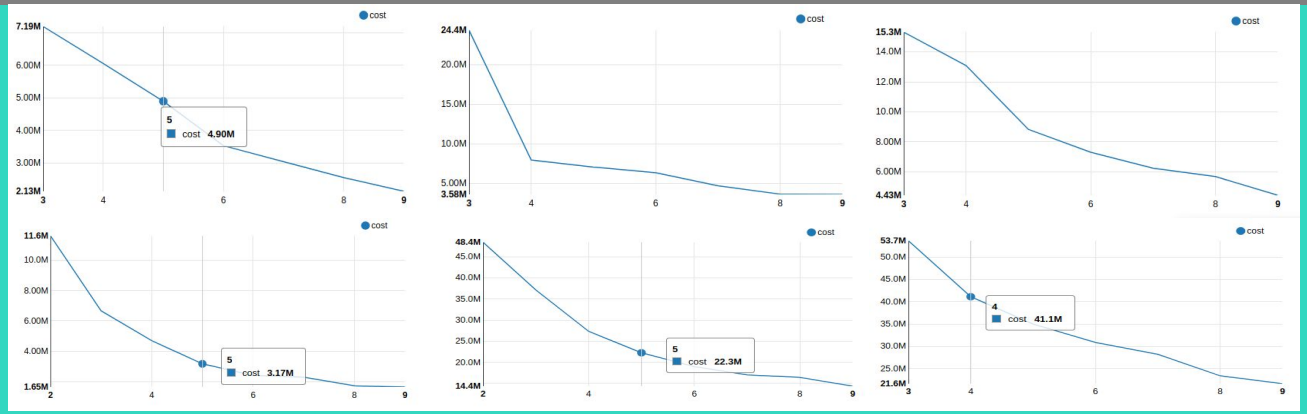
From Spark MLlib

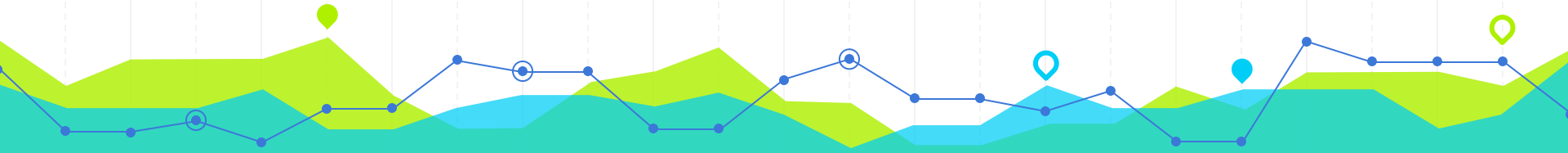
1

K-Means

The work we have done on clustering can be divided into two parts. One is clustering according to player's travel preference, the other one is clustering according to player's battle performance. Each part is also divided into 3 subsets according to the game mode (Solo, Duo or Squad). In all, we have 6 models for clustering.

For each model we used elbow method to determine the number of k.





Classification Model

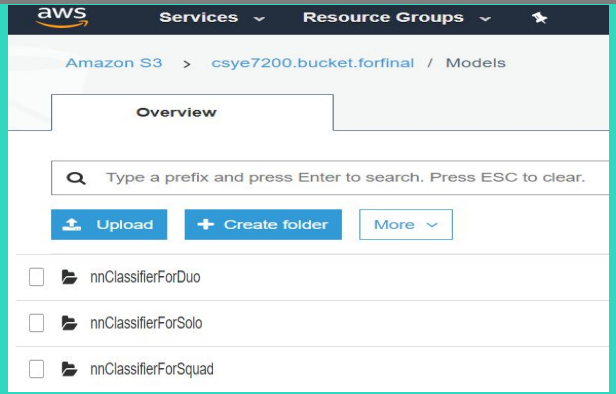
From Spark MLlib

2

Multilayer perceptron classifier

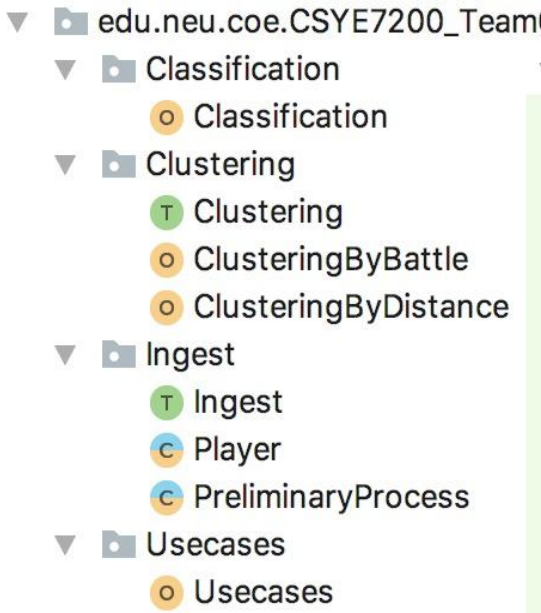
Like what we did for clustering, we also separated players by game mode for classification. So we also have three models for determining if a player is a winner in a single game.

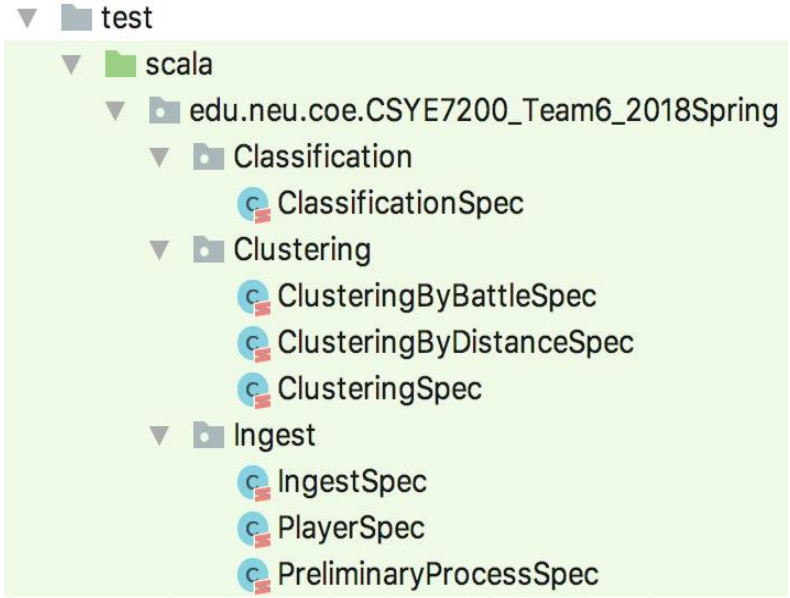
We tuned the models for several times to get the parameters we eventually used.
Although we are running all this on a 4 nodes cluster, it was still a very time consuming job.



Programing In Scala

- PreliminaryProcess
- Ingest
- Player
- Clustering
- ClusteringByBattle
- ClusteringByDistance
- Classification
- Usecases
- Unit Test





Data Source :

★ All data are from kaggle called PUBG Match Deaths and Statistics.

There are approximately 10 million rows and 20 GB in all.

Reference:

★ https://www.kaggle.com/skihikingkevin/pubg-match-deaths

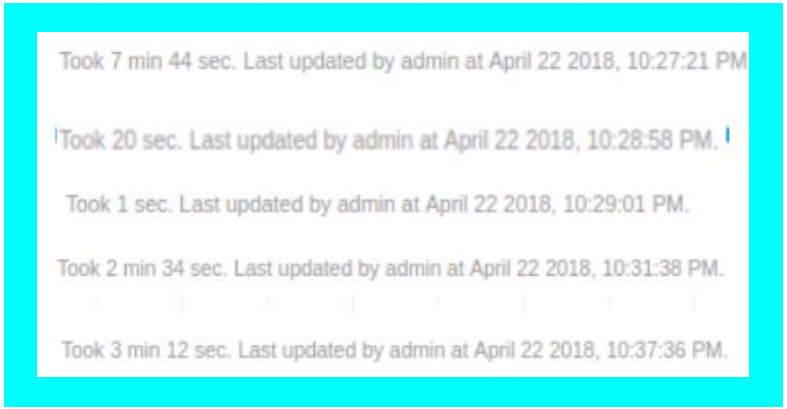
Acceptance Criteria Report

- The speed of our system to finish a visualization work should be faster than 20s/GB.

18 charts took 831s for 8GB data.

831 / 18 / 8 = 5.77

The speed of our system is 5.77s / GB.



Acceptance Criteria Report

- Classification model can determine if the user is a winner. The accuracy should be at least 70%.

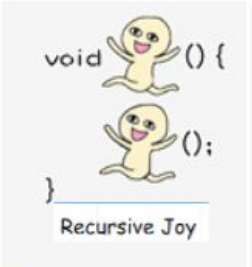
$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

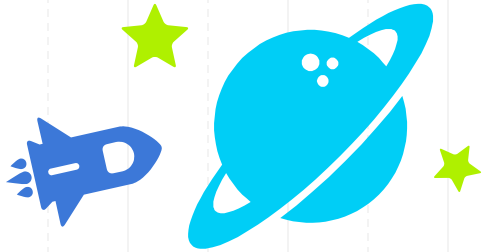
```
res64: Double = 0.6446023989338072

res65: Double = 0.5163511187607573
```

```
Test set accuracy = 0.9891577563669826
Test set accuracy = 0.9785052639146727
Test set accuracy = 0.9597469032400867
```




```
for(n ← Range(0, 100)) yield
```



A Little Surprise

Hacker Detector

```
1|[49.8169372623138...|[0.49816937262313...|     1.0|
      tpp|        1|           0|        0|          0.0|3409.8781700000004|     1484|        15| WG-Qun301323278|          612.186| 100076|          37|[0.0,0.0,0.0,3409...|[0.0,0.0,0.0,0.90...|
0|[49.8169372623138...|[0.49816937262313...|     1.0|
      tpp|        1|           1|        0|          0.0|         7311.286|     3639|        37|664301849-QUN-KK| 1329.5810000000001| 100093|           1|[1.0,0.0,0.0,7311...|[1.20854080418411...|
1|[47.8045245724561...|[0.47804524572456...|     1.0|
      tpp|        1|           1|        0|          0.0|        10071.751|     4109|        39| QQqun-648314099|          1195.715| 100087|           1|[1.0,0.0,0.0,1007...|[9.13803081805472...|
1|[47.9878564125485...|[0.47987856412548...|     1.0|
      tpp|        1|           0|        0|          0.0|         7661.149|     3105|        31|    liujiachuang| 1508.3010000000002| 100041|           1|[0.0,0.0,0.0,7661...|[0.0,0.0,0.0,0.91...|
1|[49.8169372623138...|[0.49816937262313...|     1.0|
      tpp|        1|           0|        0|          0.0|         9212.909|     3663|        36|   WG_QUN_567435|          1670.192| 100057|           1|[0.0,0.0,0.0,9212...|[0.0,0.0,0.0,0.91...|
1|[49.8169372623138...|[0.49816937262313...|     1.0|
```

Demo Link :

https://ec2-34-207-189-154.compute-1.amazonaws.com/

THANKS!

Repo Link :

https://github.com/beiwen/CSYE7200_FinalProject_Team06