Big Data Management - Project Presentation

Del corso del

Prof. Fabrizio Montecchiani

Maiorani Simone, 346627

Venturi Marco, 346951

Perugia, Anno Accademico 2022/2023
Università degli Studi di Perugia
Corso di laurea magistrale in Ingegneria Informatica e Robotica
Curriculum Data Science
Dipartimento di Ingegneria

Contents

1. Introduction

The project is designed for a startup that deals with comparing and recommending smartphones to final users. It carries out big data analytics to study smartphones and to provide the end user with summary statistics in order to be able to recommend the type of smartphone that best suits him in the right way. The dataset used in our project is MobilePriceClassification.

The **goal** of our project is to make predictions on the possible price range to which a smartphone can belong. The possible price ranges are **low**, **cheap**, **medium** and **high**. Furthermore, the charts containing the statistics common to all the smartphones belonging to the output class are displayed. For example, if after the data entered by the user the output price range is "medium" then statistics belonging to "medium" class smartphones will also appear. The model used for the prediction is a multiclass classifier, specifically **Softmax Regression** (multinomial logistic regression).

The model is trained by taking the dataset from HDFS as input. For the realization of the charts **Plotly** is used, a library of high-level declarative charts. These charts are based on the results taken from processing the SQL queries used to extrapolate information from our dataset. Specifically, the statistics concern the number of megapixels, the amount of RAM memory and the battery capacity.

The entire project is tested both in a local environment and in a distributed environment. Specifically, the cluster is configured as follows:

- **Master Node** which also contains the Namenode, the Secondary Namenode, the Datanode (1 of 2), the Node Manager and the Resource Manager (the latter two required for YARN).

- **Worker Node** that only plays the role of Datanode (2 of 2).

Both versions are available at the following link: GitHub.

2. Dataflow and Technology

The data project are uploaded and generated over a Hadoop HDFS cluster. Those data are replicated with a factor of 3 unit, because the dataset is accessed many times for analytics tasks. The HDFS scheduler configuration are set up for schedule the jobs as **capacity scheduler** type with two queues, one for development purpose, called **dev**, and another for production purpose, called **prod**.

Our outputs data has two different processes. The first process generates a classification regression model, while the second uses queries to extract useful statistics informations about phones.

2.1 Dataset

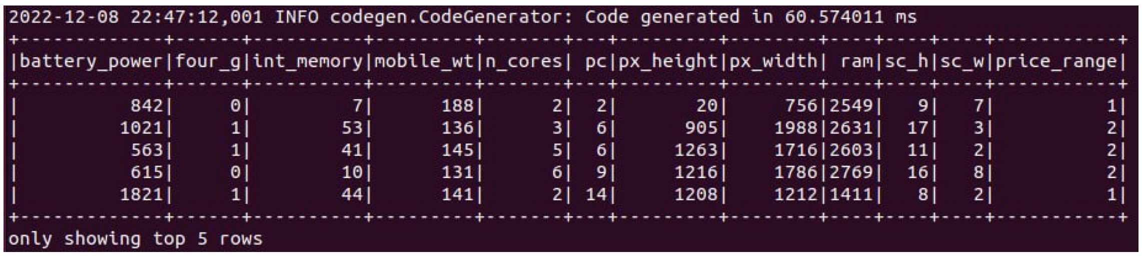
The dataset goal is to predict phone's price range over four different categories. It contains twentyone features. The project goal is not to implement a fully efficient model, so we decide to use just a subset of features:

• *battery_power*	Int: mAh
• *fc*	Front camera | Int: Mega pixel
• *four_g*	Boolean : has or not
• *int_memory*	Internal memory | Int: Gb
• *mobile_wt*	Weight of mobile phone | Int: grams
• *ppi*	px per inches | Int
• *ram*	Int: Mb
• *inch*	screen inches | Double cm
• *price_range*	categorical | Int: 0,1,2,4

Inch and ppi features are acquired from sc_w, sc_h and inch, px_width respectively.

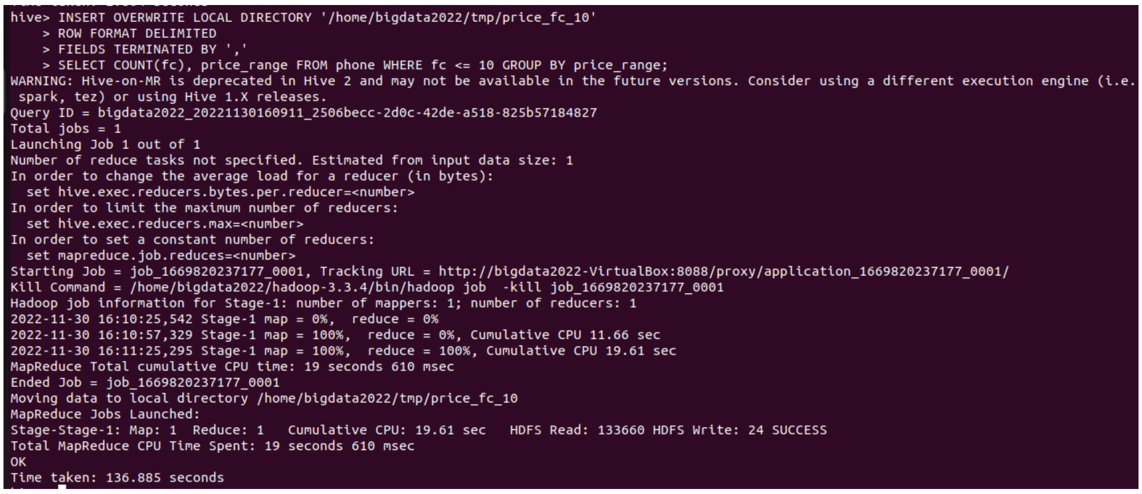
2.2 Tecnologies and Libraries

Model Engineering is generated using Apache Spark distributed mode to train, evaluate and test model and for features engineering too, while the data extracted from dataset about phones statistics are acquired using Apache Hive in is local mode.



```
2022-12-08 22:47:12,001 INFO codegen.CodeGenerator: Code generated in 60.574011 ms
+-------------+------+----------+---------+-------+---+---------+--------+----+----+----+-----------+
|battery_power|four_g|int_memory|mobile_wt|n_cores| pc|px_height|px_width| ram|sc_h|sc_w|price_range|
+-------------+------+----------+---------+-------+---+---------+--------+----+----+----+-----------+
|          842|     0|         7|      188|      2|  2|       20|     756|2549|   9|   7|          1|
|         1021|     1|        53|      136|      3|  6|      905|    1988|2631|  17|   3|          2|
|          563|     1|        41|      145|      5|  6|     1263|    1716|2603|  11|   2|          2|
|          615|     0|        10|      131|      6|  9|     1216|    1786|2769|  16|   8|          2|
|         1821|     1|        44|      141|      2| 14|     1208|    1212|1411|   8|   2|          1|
+-------------+------+----------+---------+-------+---+---------+--------+----+----+----+-----------+
only showing top 5 rows
```

Features selected in Spark



```
hive> INSERT OVERWRITE LOCAL DIRECTORY '/home/bigdata2022/tmp/price_fc_10'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > SELECT COUNT(fc), price_range FROM phone WHERE fc <= 10 GROUP BY price_range;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e.
 spark, tez) or using Hive 1.X releases.
Query ID = bigdata2022_20221130160911_2506becc-2d0c-42de-a518-825b57184827
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1669820237177_0001, Tracking URL = http://bigdata2022-VirtualBox:8088/proxy/application_1669820237177_0001/
Kill Command = /home/bigdata2022/hadoop-3.3.4/bin/hadoop job  -kill job_1669820237177_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-11-30 16:10:25,542 Stage-1 map = 0%,  reduce = 0%
2022-11-30 16:10:57,329 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 11.66 sec
2022-11-30 16:11:25,295 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 19.61 sec
MapReduce Total cumulative CPU time: 19 seconds 610 msec
Ended Job = job_1669820237177_0001
Moving data to local directory /home/bigdata2022/tmp/price_fc_10
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 19.61 sec   HDFS Read: 133660 HDFS Write: 24 SUCCESS
Total MapReduce CPU Time Spent: 19 seconds 610 msec
OK
Time taken: 136.885 seconds
```

Example of HQL

Real time predictions are made possible by using Openscoring, a middleware for predictive analytics applications, while for charts visualization we use Plotly.js. All the outputs are then merged in a single client-server structure made using **HTML client** and a **Node.js** server.

2.3 Libraries and Frameworks

• *Apache Hadoop*	3.3.4
• *Apache Spark*	3.3.2
• *Apache Hive*	2.3.9
• *PySpark*	2.16.1
• *Openscoring Server*	2.1.1
• *PySpark2PMML*	2.2.0
• *JPMML − SparkML*	2.2.0
• *node.js*	18.12.1
• *Plotly.js*	2.16.1
• *Bootstrap*	5.2

3. Execution Instructions

3.1 Running the Project

The program can be launched by running the **runProject.sh** script that contains all the scripts and commands necessary for the correct execution of the program, specifically:

- runHadoop.sh
- runSparkApp.sh
- runHive.sh

Each script run the homonymous framework with all the connected shell command for the achievement of tasks. If we talk about Hadoop, the script only runs the command about start-all.sh, while for Spark it also runs all the commands about output folder position and xml to pmml file extension change. runHive.sh script file instead runs only the queries contained inside hive_querys.sql file. runProject.sh also contains lines to execute all server side application.

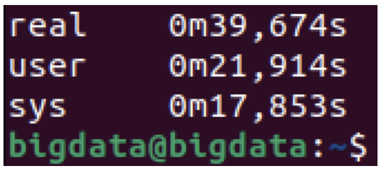
The program can be terminated by running the **stopProject.sh** which contains the script for stopping Hadoop and commands to close server side application execution.

3.2 How It Works

After the program has started, a Chrome window is launched. By visiting the "**http://localhost:8081/index**" page, you access our interface where the user can enter data in a form and launch the prediction. After the prediction process, the client also display the charts.

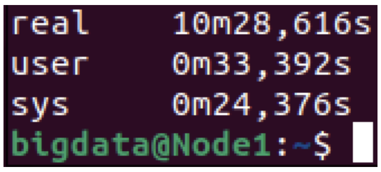
3.3 Time Execution

Here are the **compute times** for Hive and Spark in both their local and distributed versions.



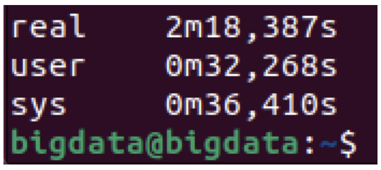
```
real	0m39,674s
user	0m21,914s
sys	0m17,853s
bigdata@bigdata:~$
```

Hive: Local Computation



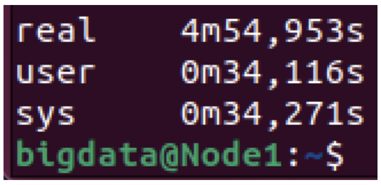
```
real	10m28,616s
user	0m33,392s
sys	0m24,376s
bigdata@Node1:~$
```

Hive: Cluster Computation



```
real	2m18,387s
user	0m32,268s
sys	0m36,410s
bigdata@bigdata:~$
```

Spark: Local Computation



```
real	4m54,953s
user	0m34,116s
sys	0m34,271s
bigdata@Node1:~$
```

Spark: Cluster Computation

We can see how in this case the execution time on a cluster mode is greater than on a local mode. This is because HDFS does not perform well with small datasets (without counting the addition of communication time between the two nodes).

4. Limits and Possible Improvements

Regarding **data quality**, the system could be improved by adding more recent data inside the dataset and removing historical ones, because the data refer to 2017, and they don't retrive current phones statistics.

Regarding **Hive Engine**, it could be updated with the Spark engine. By default Hive execution engine is set to mr, that stand for Map Reduce engine, and it should be changed because it does not support any more cluster execution mode. Spark supports Hive cluster mode, but to apply this improvement we should use Spark 2.0.0 for version compatibility reason.

Regarding **Hadoop Cluster Configurations**, we should move the SecondaryNamenode to a different machine otherwise fault tolerance is not guaranteed. Having both in the same node is a single point of failure.