Silvia Fracalossi – 13180

Programming Data Infrastructure Project

Git Repository: https://gitlab.inf.unibz.it/Silvia.Fracalossi/pdi2019-project

2th September, 2019

1. Motivation Scenario

Most of the travellers already know where to go, once they reach the airports: they have been planning their dream vacation for weeks. However, people like students and young workers do not have, sometimes, enough time to organize themselves a vacation. What to do, rather than stay home?

WhereToFly is the platform that helps to solve this problem: it is an application to search for flights in all the United States. In the platform, a filter on the origin and destination airports can be utilized. The platform responds with a list of 15 flights from which the user can choose!

2. Data Description

The data of the project consist in several CSV, related with one another, all describing different entities in the flights environment. The files have been found and downloaded from two different sources:

- Kaggle, a web-site providing competitions and datasets;
- Transportation.gov, a platform offering datasets regarding US transportation information.

Therefore, the downloaded CSV files are the following four::

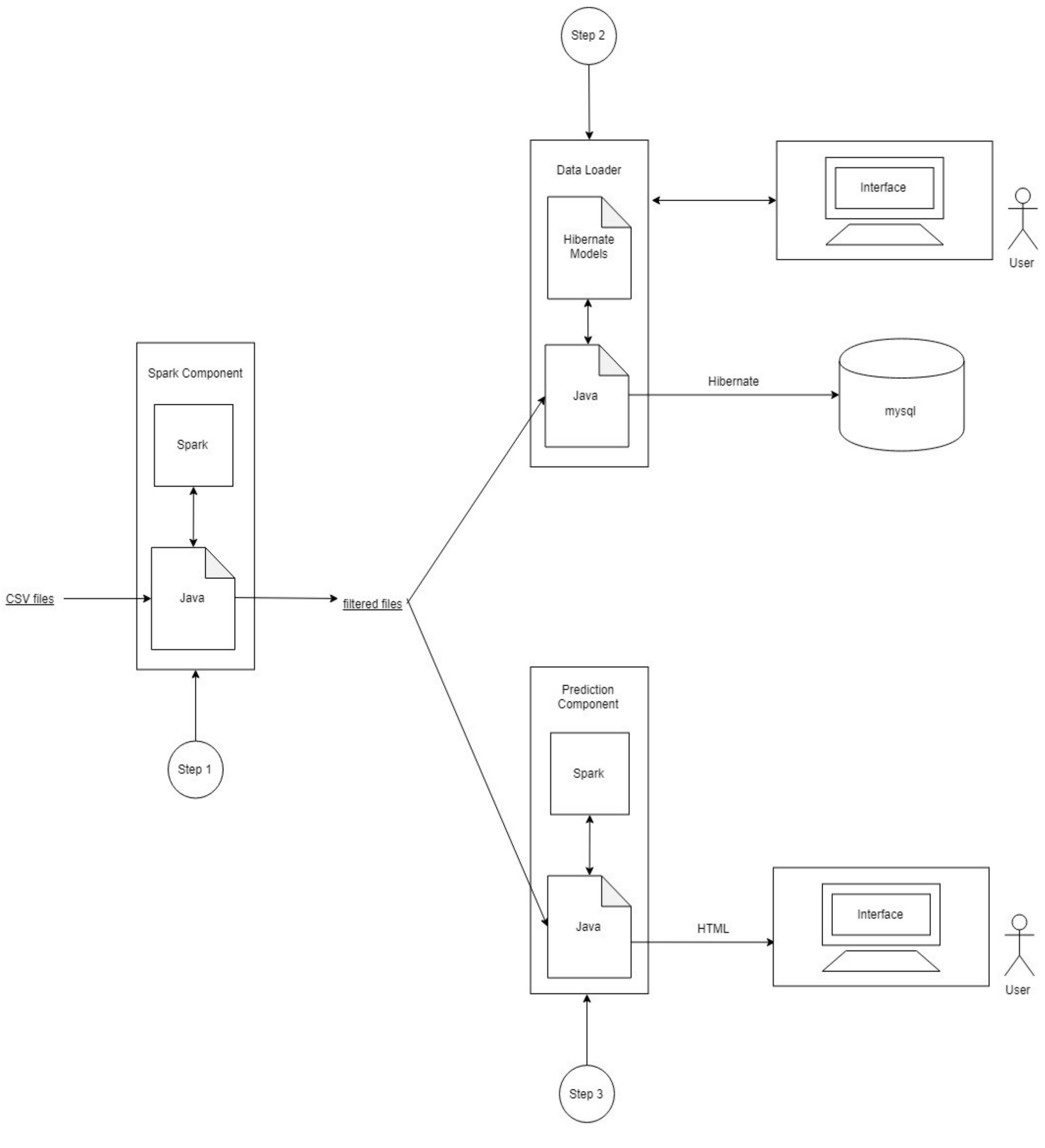
- Airlines.csv, from https://www.kaggle.com/fabiendaniel/predicting-flight-delays-tutorial; it contains the IATA code and name of the Airlines;
- Airports.csv, from https://www.kaggle.com/fabiendaniel/predicting-flight-delays-tutorial; it contains the IATA code, name and location of the Aiports;
- Flights.csv, from https://www.kaggle.com/fabiendaniel/predicting-flight-delays-tutorial; it contains the day, scheduled departure and arrival time, actual departure and arrival time, airline, origin and destination airports of the Flights;
- Transportation_airfare.csv, from https://data.transportation.gov/Aviation/Consumer-Airfare-Report-Table-6-Contiguous-State-C/yj5y-b2ir/data; it contains the fare per passenger, differentiated by year, quarter, origin city and destination city.

3. General architecture of solution and used technologies

The technologies used in the project are Java as coding language, Spark as data parser, Hibernate as data storer, HTML as visualization language. The database engine is MySQL.

The application can be seen as a union of three different components, using two distinct technologies:

1. Spark Component, a Java application using *Spark* sessions, utilized to elaborate data and store its clean version;
2. Data Loader, a Java component using the *Hibernate* technology, utilized to store the parsed data into the database and to visualize the interface;
3. Prediction Component, a Java application using Spark sessions and Mllib, utilized to compute predictions on the flights delay.



4. Interface and System Functionalities

Below, a more detailed description of the application components introduced in Chapter 3.

– Step 1: Spark Component

The first component executed is the Spark Component. The goal of this part is to elaborate, clean and filter the data stored in the original CSV files, eventually storing them into different files and directories, according to the different entities they represent.

Therefore, the Spark technology has been used differently for the distinct files:

- Airlines.csv, the first file parsed. This file represents the Airline entity. No special elaboration has been made, the data was in the correct shape to be saved in the new file;
- Airports.csv, the second file parsed. This file represents the Airport entity. Rows with lacking information have been removed using a Spark Filter Function;
- Flights.csv, the third file parsed. This file represents both the Flight entity and the Route entity. Therefore, the file has been handled in two different ways:
    - Routes: Useless columns have been removed. Successively, a primary key has been created, by combining together origin and destination Airport IATA codes of the flight. Duplicates have been dropped;
    - Flights: Useless columns have been removed. Successively, the foreign key with the Route entity has been created by, again, joining origin and destination airport IATA codes. Date information, like day, month and year, have been joined in a single one. Finally, rows with null values have been dropped.
- Transportation_airfare.csv, the last file parsed. This file represents the Fare entity. Useless columns have been removed. On the Airport cities fields, two filters have been applied: one removing white spaces; the other one removing characters contained into brackets.

The files produced at the end have been stored into five different folders, one for each entity, all contained in the "parsed_data" folder.

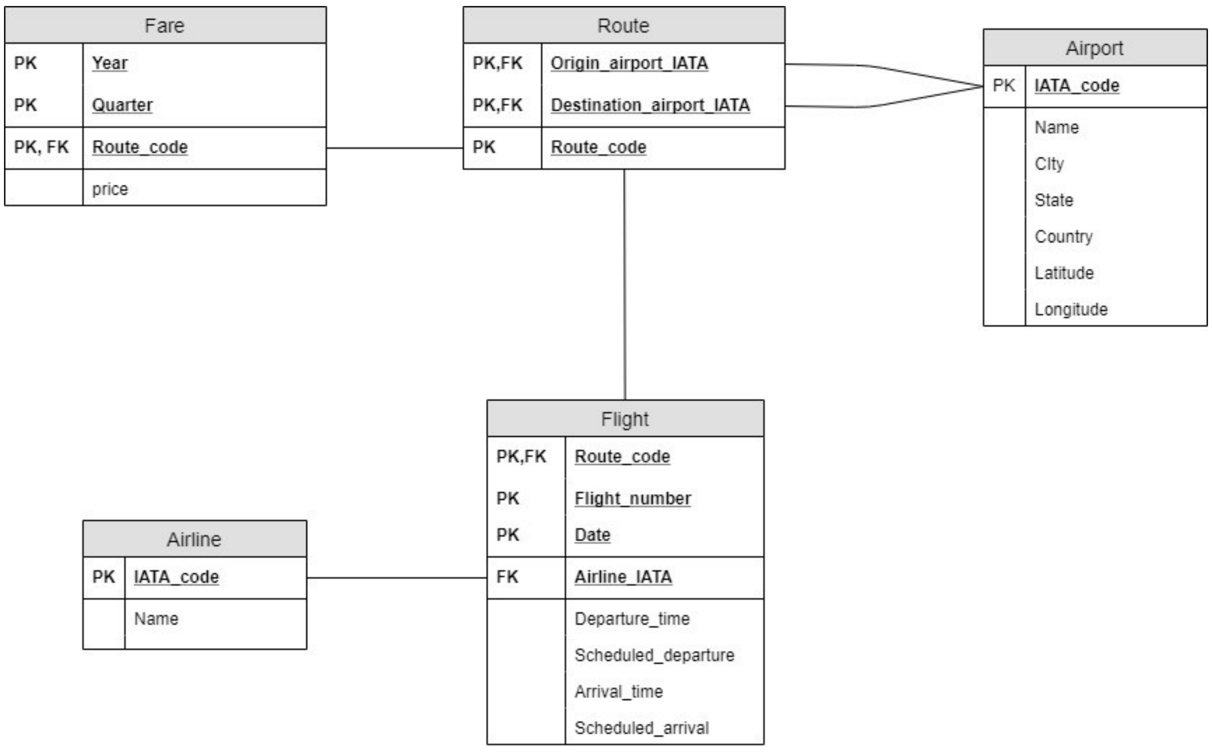
– Step 2: Data Loader

The second component is called Data Loader. This part of the project is responsible for reading the files produced by the Spark Component and storing the content into the MySQL database, using the Hibernate models and methods. Moreover, it manages and shows the interface of the application.

– 2.1.: Data Handler

Therefore, a Java-Hibernate Model for each entity has been created. A Controller file and a MySQL Service file for each entity have been coded, in order to handle the several calls to the database. Thanks to a configured session object, the data have been stored to the MySQL database.

Thus, the data structure of the database can be represented with the following E/R model:



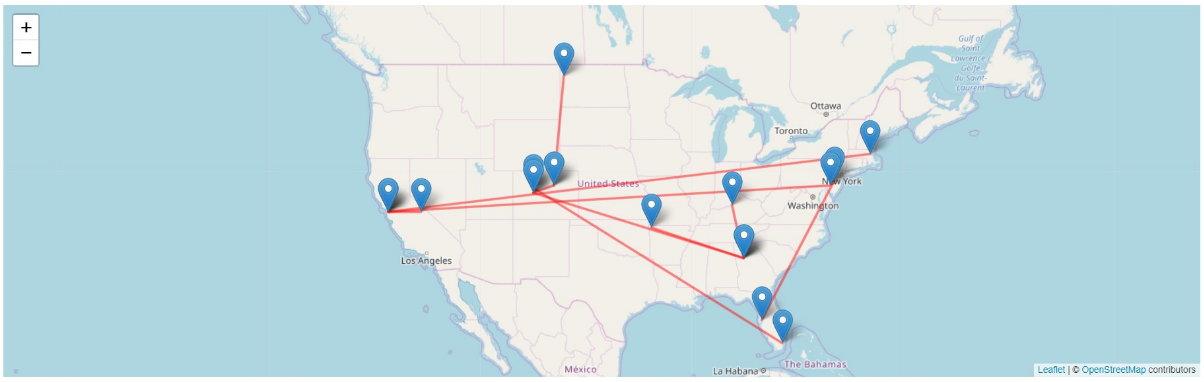
– 2.2: Interface

In the customer interface, the user can look for flights and see the different routes between the airports. As mentioned before, a filtering system will be implemented to make the user look for a flight based on the starting airport and on the destination airport.

In the Home page of the application, there is a button to access the analytics, called “Show Analytics”: by clicking it, the HTML page of the project analytics is displayed. The page includes five queries about the data collected in the database:

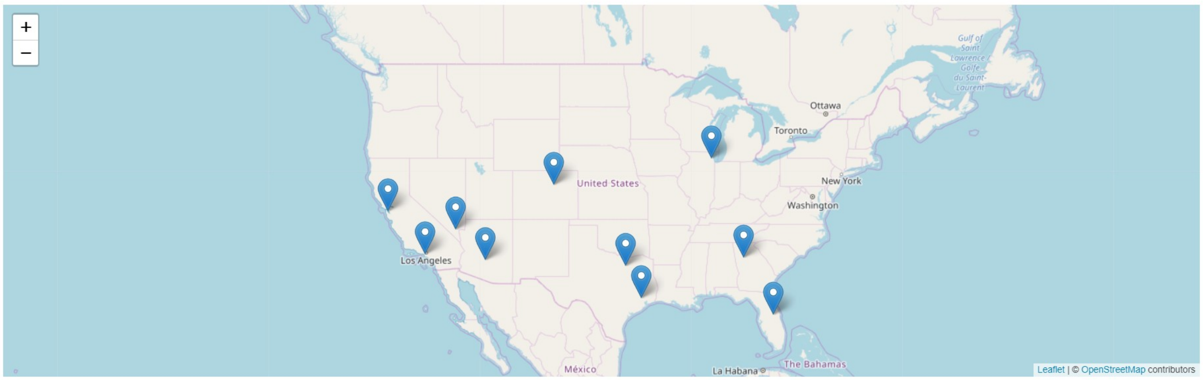
1. What are the most expensive routes?

Question 1: What are the most expensive routes?



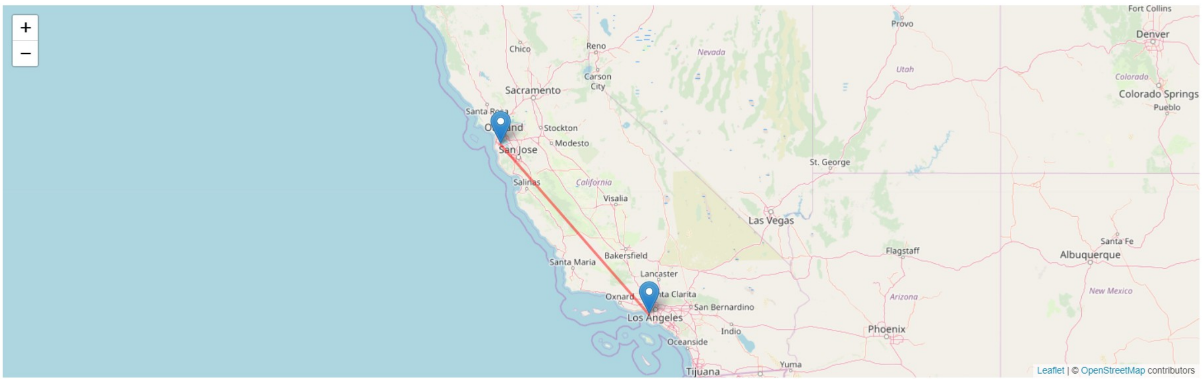
2. What are the airports with the highest number of flights?

Question 2: What are the airports with the highest number of flights?

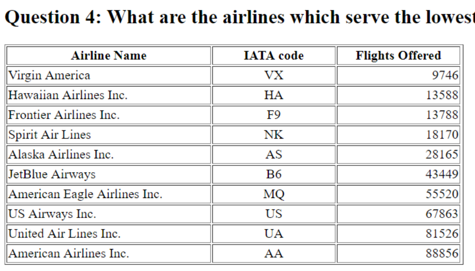


3. What are the two most connected airports?

Question 3: What are the two most connected airports?



4. What are the airlines which serve the lowest number of flights?

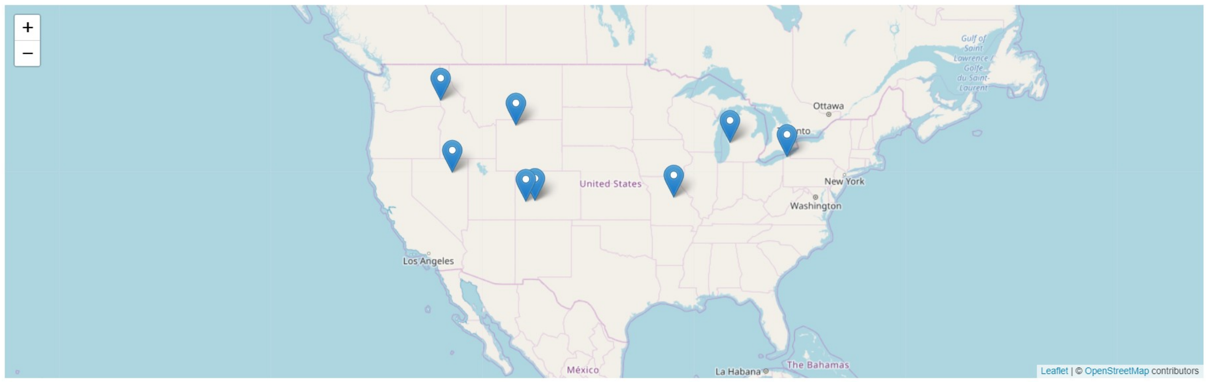


Question 4: What are the airlines which serve the lowest number of flights?

Airline Name	IATA code	Flights Offered
Virgin America	VX	9746
Hawaiian Airlines Inc.	HA	13588
Frontier Airlines Inc.	F9	13788
Spirit Air Lines	NK	18170
Alaska Airlines Inc.	AS	28165
JetBlue Airways	B6	43449
American Eagle Airlines Inc.	MQ	55520
US Airways Inc.	US	67863
United Air Lines Inc.	UA	81526
American Airlines Inc.	AA	88856

5. What are the airports causing the greatest departure delay per flight?

Question 5: What are the airports causing the greatest departure delay per flight?


+
−
Gulf of Saint Lawrence Golfe du Saint-Laurent
Ottawa
nto
United States
New York
Washington
Los Angeles
México
La Habana
The Bahamas
Leaflet | © OpenStreetMap contributors

The other button in the interface is called “Explore Flights”. By clicking it, another Java window will appear, containing two Java Combo Box, one showing the origin airports, the other the destination airports. After the selection of the two airports, by clicking the “Search” button, 15 flights will be printed in the console.

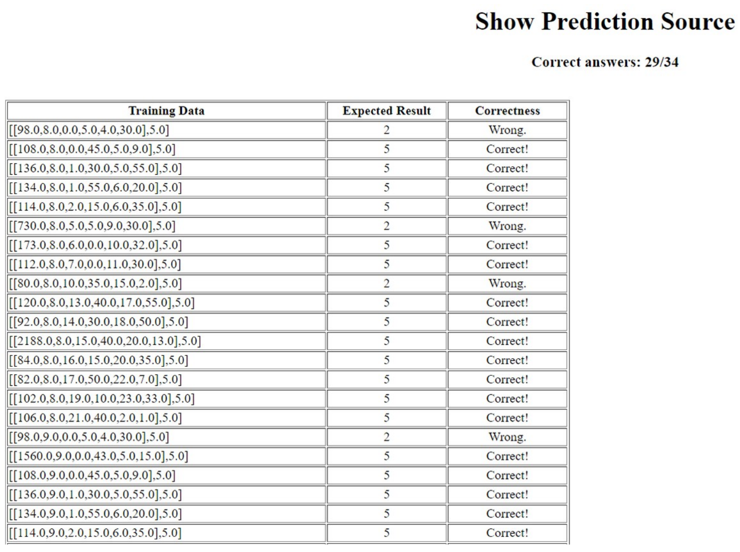
– **Step 3: Prediction Component**

The third and last component is called Prediction Loader. This part of the project is responsible for predicting the flights delays. Spark and SparkML are the technologies used to cover the functionality.

The flights have been sampled directly from the "parsed_data/flights" folder. Only the flights covering the "ANC_SEA" route have been chosen, in order to analyse a congested route and to have a more specific analysis of the delays, since the starting airport would always be the same. After parsing the data, the most relevant information have been selected as "features":

- Flight Number;
- Day;
- Departure Time;
- Arrival Time.

Depending on the scheduled departure time and the actual departure time, the flights have been classified in "delay" and "in time". After the model has been created, the prediction has been done on other flights information. At the end of the execution, the results have been displayed in an HTML file, including the visualization of the correctness.



Show Prediction Source

Correct answers: 29/34

Training Data	Expected Result	Correctness
[[98.0,8.0,0.0,5.0,4.0,30.0],5.0]	2	Wrong.
[[108.0,8.0,0.0,45.0,5.0,9.0],5.0]	5	Correct!
[[136.0,8.0,1.0,30.0,5.0,55.0],5.0]	5	Correct!
[[134.0,8.0,1.0,55.0,6.0,20.0],5.0]	5	Correct!
[[114.0,8.0,2.0,15.0,6.0,35.0],5.0]	5	Correct!
[[730.0,8.0,5.0,5.0,9.0,30.0],5.0]	2	Wrong.
[[173.0,8.0,6.0,0.0,10.0,32.0],5.0]	5	Correct!
[[112.0,8.0,7.0,0.0,11.0,30.0],5.0]	5	Correct!
[[80.0,8.0,10.0,35.0,15.0,2.0],5.0]	2	Wrong.
[[120.0,8.0,13.0,40.0,17.0,55.0],5.0]	5	Correct!
[[92.0,8.0,14.0,30.0,18.0,50.0],5.0]	5	Correct!
[[2188.0,8.0,15.0,40.0,20.0,13.0],5.0]	5	Correct!
[[84.0,8.0,16.0,15.0,20.0,35.0],5.0]	5	Correct!
[[82.0,8.0,17.0,50.0,22.0,7.0],5.0]	5	Correct!
[[102.0,8.0,19.0,10.0,23.0,33.0],5.0]	5	Correct!
[[106.0,8.0,21.0,40.0,2.0,1.0],5.0]	5	Correct!
[[98.0,9.0,0.0,5.0,4.0,30.0],5.0]	2	Wrong.
[[1560.0,9.0,0.0,43.0,5.0,15.0],5.0]	5	Correct!
[[108.0,9.0,0.0,45.0,5.0,9.0],5.0]	5	Correct!
[[136.0,9.0,1.0,30.0,5.0,55.0],5.0]	5	Correct!
[[134.0,9.0,1.0,55.0,6.0,20.0],5.0]	5	Correct!
[[114.0,9.0,2.0,15.0,6.0,35.0],5.0]	5	Correct!

5. Conclusion and Lessons Learned

The project is a collection of a Spark filtering component, a Spark prediction component, a Java-Hibernate data manager and loader, communicating with plain files and a MySQL database, a Java interface and HTML+JavaScript visualization pages.

The most important lesson I have learned in the development regards Hibernate.

The main advantage of Hibernate is the mapping and its HQL queries: both during the code development and also in the Analytics queries implementation, I have noticed that once the database entity have been mapped to the database classes, the insertion of them in the MySQL database and the creation of HQL queries is really facilitated. Especially in the Analytics queries, several joins between tables have been spared thanks to the relations between entities specified throughout the files.

The disadvantage of this technologies I met is related to the insertion speed. The greatest file in the project contained 5,000,000 to be stored in the MySQL database. The speed of the computation on my laptop equals 35,000 insertion per hour.

In conclusion, I think Hibernate is a really useful tool when it comes to queries facilitation and little sets insertion. Perhaps, for the initial data insertion, another technology could have been better. For a daily rows insertion, Hibernate is a great tool.

An improvement to the project I would like to apply is the management of the Spark and the Prediction components directly from the interface, by calling prompt commands already hard-coded in the application, triggered by some buttons in the interface. Another improvement could be implementing more filters in the “Explore Flight” window functionality.

6. Project Configuration

The software that have been installed in order to run the project are the following ones:

- XAMPP (including *mysql-phpmyadmin* plugin) and the related dependencies;
- Spark and the related dependencies;
- Intellij environment and the related dependencies.

Instructions used to set up and run the WhereToFly application:

- Start Apache and MYSQL servers from XAMPP;
- Create a database called “wheretofly”;
- Created an account “silvia” with password “aPassword!”. Grant the new account the privileges to access and edit “wheretofly” database.
- Download the repository from the Unibz GitLab;
- Make sure the “data” folder is in the main directory of the project;
- Make sure the “ex_out” and “parsed_data” folders are not in the main directory;
- Access the SparkComponent folder:
    - Execute “mvn package”, having Java version 12 configured;
    - Execute “spark-submit --class Main --deploy-mode client --master local target/SparkComponent-1.0.0.jar”
- After the execution, a folder called “parsed_data” containing 5 folders has been created;
- Access the DataLoader application through the IntelliJ environment:
    - Build project;
    - Run project;
- Access the PredictionComponent application by accessing the folder:
    - Execute “mvn package”, having Java version 8 configured;
    - Execute “spark-submit --class SparkDriverLRegression --deploy-mode client --master local target/LRJar-1.0.0.jar”.