PREDICTIVE ANALYTICS USING BIG DATA TECNOLOGIES
(FOR THE ACME-FLYING USE CASE)

Big Data technologies handle volume, velocity and variety much better than traditional relational-based approaches. Furthermore, they follow a load-first model-later approach (opposed to model-first load-later as Data Warehousing solutions do) that provides flexibility for developing customized analytical data pipelines from diverse data sources.

In this lab we will continue working on a real setting based on the ACME flying use case. The sources we are going to consider are the following:

1. The data warehouse developed for the first lab, which provides a consolidated view of the AMOS and AIMS databases.
2. Additional CSV files provided from the ACMS system. ACMS consists of several aircraft built-in sensors that traces the events generated by a sensor during a flight.

The information you consolidated in the Data Warehouse talk about what happened; therefore, it is meant to produce descriptive analysis reports (either static or OLAP-based ones). In this project, we want to develop a complex analytical pipeline for predicting **if an aircraft is going to interrupt its operation unexpectedly in the next seven days (i.e., a predictive analysis)**. In the AMOS database, there exists a table called *Operation Interruption*, which reports about past events. This projects wants to predict (and avoid happening) potential future occurrences in this table.

Sources

We will limit the scope of the project given the time available. Therefore, consider the following constraints with regard to the data sources.

- The Data Warehouse: typically, when predicting operation interruption, a good share of the data warehouse KPIs would be included as features. In our case, we will simplify the problem and consider only three KPIs: flight hours (FH), flight cycles (FC) and delayed minutes (DM).
  - We provide a functioning Data Warehouse meeting the first lab's requirements. This is available in the *DW* database, in FIB's PostgreSQL server:
    - The parameters are the same as the ones used to connect to AMOS, AIMS. Only the name of the database is **DW**.
  - Realise this Data Warehouse records *AircraftUtilization* and *LogBookReporting* at the day level (and not per month). We introduced this change to facilitate this lab.
- The ACMS system: we provide the sensor data generated per flight. Since each aircraft has several subsystems, each of them with several sensors, we simplify the problem by focusing on the aircraft navigation subsystem (ATA code: **3453**) and assuming this subsystem has one single sensor. This data is gathered as follows:
  - The sensor generates an event every 5 minutes from take-off until landing, collecting the measurement of the sensor. This information is stored per flight in a single CSV file.
  - For this subsystem, we will thus work with CSV files.

- You will find a .ZIP file (*trainingData.zip*) containing 376 CSV files in the *resources* directory of the code skeleton.

Thus, the variables we are going to consider to predict if an aircraft is going to interrupt operations in the next 7 days are: the FH, FC and DM KPIs, and the aircraft navigation subsystem data collected as CSV files.

Processing

To implement the required data pipeline, you must create three data processing pipelines (i.e., management, analysis and evaluation) in order to predict the need for maintenance for an aircraft. These pipelines must be implemented in Spark using the PySpark library.

The designed Python code should adhere to the following instructions:

Data Management Pipeline

This pipeline generates a matrix where the rows denote the information of an aircraft **per day**, and the columns refer to the FH, FC and DM KPIs, and the average measurement of the 3453 sensor. Thus, this pipeline must:

- Read the sensor measurements (extracted from the CSV files) related to a certain aircraft *A* and average it per day.
- Once you have the average measurement of the sensor per day, enrich it with the KPIs related to *A* from the Data Warehouse (at the same granularity level).
- Importantly, since we are going to use a supervised learning algorithm (see the data analysis pipeline) we need to label each row with a label: either **unscheduled maintenance** or **no maintenance** predicted in the next 7 days for that flight.
- Generate a matrix with the gathered data and store it.

Data Analysis Pipeline

By using the data matrix created in the previous pipeline, we want to train a classifier. In the literature, we can find many papers reporting that decision trees perform well for predicting transport delays. Thus, we will use the MLlib library to train a decision tree. This pipeline must:

- Create two datasets (training and validation) and format them according to what is expected by the Decision Tree Classifier in MLlib,
- Create and store the validated model. The validation part must include traditional evaluation metrics: in this case, we propose accuracy and recall.

Run-time Classifier Pipeline

Finally, once the model has been created, we must create a pipeline that, given a new record (<aircraft, day, FH, FC, DM, AVG(sensor)>), predicts if this aircraft is going to go for unscheduled maintenance. This pipe must:

- Given an aircraft and a day, it replicates the data management pipeline (i.e., extracts the KPIs from the DW and computes the average measurement of the sensor for that day from the available CSVs) and prepares the tuple to be inputted into the model,

- Classifies the record and outputs maintenance / no maintenance.

IMPORTANT NOTE: Even if the analytical aspects of the problem fall out of scope, we do consider if you follow good practices when preparing the data to feed the model.

ADDITIONAL NOTE: Even if this lab is a simplified version of what would be required in a realistic case, it is still representative. In the following we list a set of characteristics that can be expected in a real scenario:

- Typically, the features used in the model are not decided beforehand. Instead, a first model is created and interpreted. Then, according to this interpretation, these pipelines are iteratively modified to consider new features and fine-tune the model. However, this part of the problem falls out of scope of this lab and we set the features beforehand.
- It is not a good practice to store the files generated by sensors in the local file system. Instead, a data lake (i.e., a database) should be deployed.
- A model might contain more features than the model we generated. However, from the data management point of view, it would entail more data flows consolidating variables into the matrix.
- The run-time classifier pipeline output would be implemented as an event in a dashboard. Thus, it would be one of the indicators / alarms / reports of a dashboard created with a visualization tool.

Deliverables

1) Three Python codes corresponding to each pipeline described above. The three scripts must be included in a single zip file.
    a. **<u>Important</u>**: The Python code must include comments to facilitate the rationale you followed. At the header of each file**, include an overall comment explaining WHAT are the steps implemented in the pipeline**, and refer to these steps when explaining the Spark code in the subsequent comments.
2) A PDF file (**one single A4 page, 2.5cm margins, font size 12, inline space 1.15**) with all assumptions made and justifying the decisions you made (if any).
    a. Sketch the three pipelines at a higher abstraction level. Use the notation seen in the lectures to describe the Spark job.
    b. For each pipeline, this file should elaborate on whatever other assumption not stated in the lab statement but that you followed. Thus, refer to any specificity of your solution that should help the lecturer to understand the decisions you made in your code that, otherwise, might look like controversial.

Assessment criteria:

i) Conciseness of explanations (only the first page will be considered in the evaluation)
ii) Understandability
iii) Coherence
iv) Soundness

Evaluation:

- 60% Deliverables
- 40% Exercises related to the project done individually in the day of the final exam