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

BASES DE DADES AVANÇADES
PROJECT - BIG DATA - PREDICTIVE ANALYSIS

GRAU EN CIÈNCIA I ENGINYERIA DE DADES
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Sergio Cárdenas Gracia - 54175682X

Aina Luis Vidal - 26594595V

January 2023

1 Report

The scope of this project is to develop a complex analysis pipeline for predicting if an aircraft is going to interrupt its operation unexpectedly in the next seven days. For that, we will use the AMOS and AIMS databases (some of the required parameters are stored in the DW database too) and the ACMS system, which provides sensor data generated per flight and stores it in CSV files.

To train the model, we have used the CSV files that can be found in the *trainingData* directory. We have decided to remove the *281115-FUE-OVD-7445-XY-YJU.csv* and *281115-POM-BOG-1269-XY-YJU.csv* files, which represent the flights of aircraft *XY-YJU* on 28/11/2015. This decision has been made to make a real prediction in *pipeline3* for data that is not used in the training process.

In *pipeline1*, we create an empty data frame to fill it with the CSV files information (*value*, *timeid* and *aircraftid*) and once all the files are read, we group them by *timeid* and *aircraftid* (an aircraft can do more than one flight per day) and compute the average sensor value. We also create a data frame to store the KPI values from the DW and use an inner join to merge both data frames taking *timeid* and *aircraftid* as the primary key. When doing this inner join, some of the rows of the first data frame are discarded as we don't have the KPIs values for all the (*aircraftid*, *timeid*) pairs. Afterwards, we create a data frame to store flights from the AMOS database having an unscheduled interruption operation (*Delay*, *AircraftOnGround* or *Safety*) caused by sensor 3453 and use a cross-join to merge it with the same criteria as before. Finally, for all aircraft in the data frame, we add a label column to check whether unscheduled maintenance occurs in the following 7 next days ($unshceduledmaintenance = 1$) or not ($nomaintenance = 0$) and export the matrix to a CSV file for later use.

In *pipeline2*, we read the matrix from the CSV file and adapt the format so we have a column for the labels and another for all the features assembled. With that, we split the data into training and test sets with a 70/30 partition and train the model as a decision tree. The model is stored so we can make later predictions with it. We have decided to fix a seed based on our birthdays to obtain at every execution the same results. Using the test data, we compute the confusion matrix that has 20 TP, 47 TN, 2 FP and 3 FN and some evaluation metrics. We have computed the accuracy which is of 93.1% and then we have made a differentiation between true and negative recall and precision, so a better analysis can be done. The positive recall is 87.0%, the negative one is 95.9% while the positive precision is 90.9% and the negative one is 94.9%. In general, the results obtained are satisfactory: all metrics are above 85%, the accuracy is very high and negatives recall and precision values are higher than positive ones, so our model is more confident when predicting that an aircraft is not going to need an unscheduled operation interruption.

In *pipeline3*, we allow the user to insert the input *aircraftid* and *timeid*. With that, the information of this aircraft for this date (if it exists) is obtained as in *pipeline1*, using the DW database and the CSV files with the sensor information stored in the *Data* directory that contains all the data given for doing this project. The record obtained is stored in a matrix, which is adapted to the format commented on *pipeline2*. With that matrix and the usage of the previously trained model, the prediction is done. We have tried doing the prediction for the aircraft *XY-YJU* on 28/11/2015, the one that we have previously discarded from the training data, and the obtained prediction is the correct one. We have considered that when new sensor data is measured, it is stored in the *Data* folder, but the model is still the same. If we want to retrain to adapt it to new data, we just have to run *pipeline1* and *pipeline2* again having the new data in the *trainingData* folder.

The attached *Python* codes contain more specific information about how pipelines work and some technical details, so it is recommended to read them for complete understanding.

2 Pipeline sketches

2.1 Data Management Pipeline

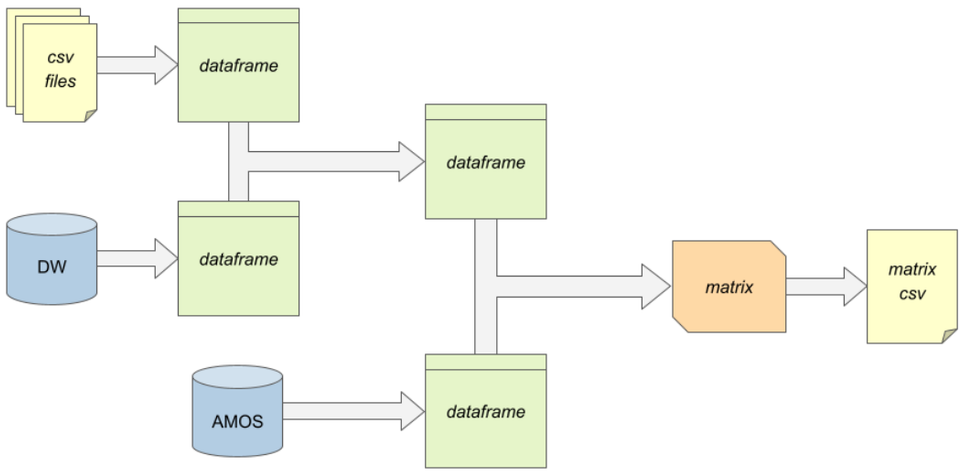

Figure 1: Sketch Data Management Pipeline

2.2 Data Analysis Pipeline

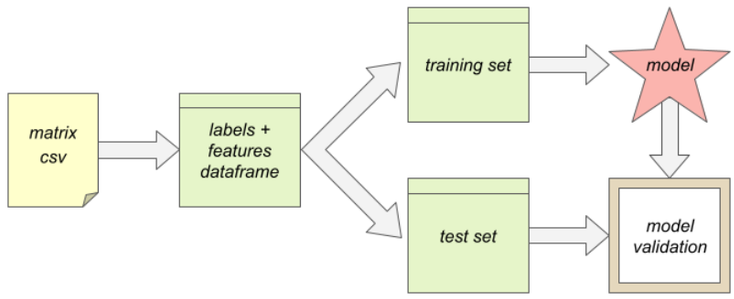

Figure 2: Sketch Data Analysis Pipeline

2.3 Run-time Classifier Pipeline

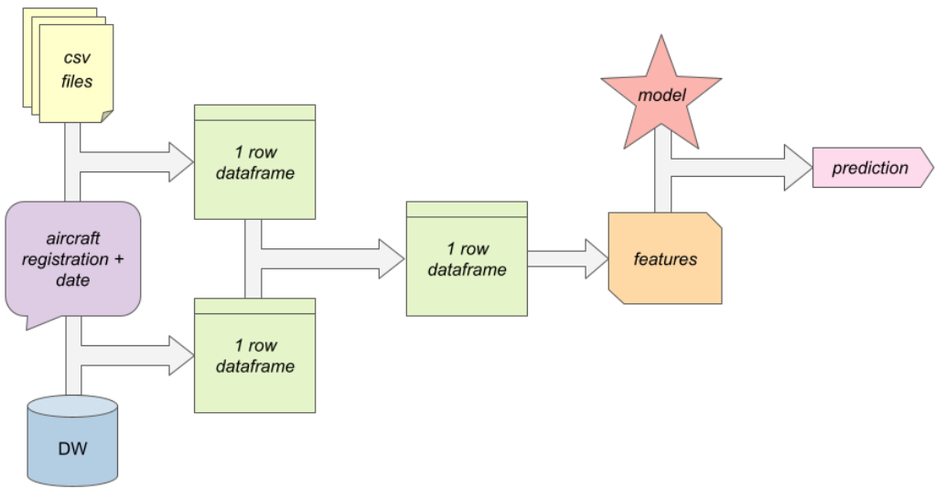

Figure 3: Sketch Run-time Classifier Pipeline