Data Management Pipeline:
In this pipeline we create the dataframe that we will use for training our predictive model. This dataframe needs to contain the following variables: *timeid*, *aircraftid*, FH, FC, DM and sensor average. Moreover, all rows will have to be labeled by whether the aircraft will need maintenance (1) or not (0) during the next 7 days. We start by getting the information of the CSV files using rdd's to parallelize this task. We use a series of map and reduce functions specified in the code to get a dataframe with <timeid> <aircraftif> and <Sensor_avg> as columns. As a remarkable point, the date needs to be obtained from each row of the files instead of the filename, because some flights take place at night and need to store their data in two different days.

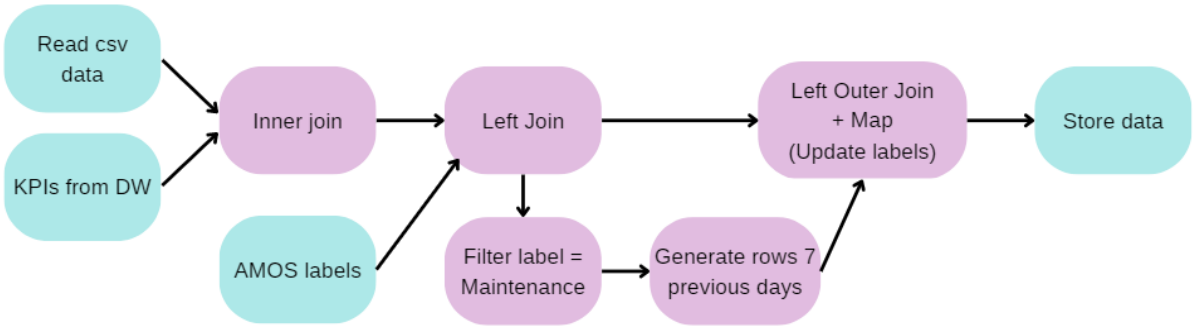
Then, we inner join the sensor data with the KPIs obtained from the DW. Next we get data from AMOS "operationinterruption" table filtered to the ones reported by subsystem 3453. We add a column <label> full of 1 because if the flight appears in this table means that it needed maintenance. We left join it with the previous dataframe created in order to give 0 value to the label of the rows that only exist in the first table (result null from the join) and 1 if they exist in both tables.

Lastly we adapt the dataframe taking into account that the prediction needs to give information about the next 7 days. To do so, we filter the samples with label 1 from our dataframe and for each row, another 6 samples with the same attributes are created for the 6 previous days. Then, the initial dataset is left outer joined with these new rows and a map function is applied so that only the rows that exist in both datasets have their label modified to 1.

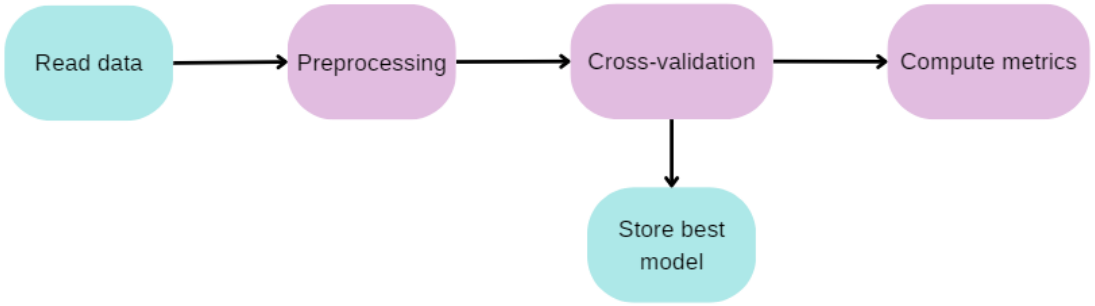
Data Analysis Pipeline:
In this pipeline we train and store the predictive model. Since we noticed that the amount of rows for each category was unbalanced, we added weights: the inverse of each class frequency. Then we prepared the data using vectorassembler and we split the data into two datasets (training and testing). Then we performed cross validation to train the decision tree classifier model avoiding overfitting. In the end, we save our best model and compute performance metrics (accuracy and recall) to check how well the model works.

Run-time Classifier Pipeline:
In this pipeline we predict whether a new input record is going to need unscheduled maintenance. First of all an input aircraft and date are asked to the user. Taking into account that the classification needs to be at run-time (new data could appear), it is necessary to reconnect to the DW and re-read all the CSV files. Then we filter both of them according to the input record and inner join them. To make sure the input record is appropriate, we added an exceptional error in case there is no information for those aircraft and date. Then we prepare the record as we did on the second pipeline with vectorassembler and we feed it to the loaded model to output the final prediction.

Data Management Pipeline:



Data Analysis Pipeline:



Run-time Classifier Pipeline: