Predictive Analysis using Big Data Technologies

Contents

- Scenario
- Data processing pipelines
  - Data management
  - Data analysis
  - Run-time classifier
- Tools
  - Python 3
  - PySpark

Scenario

Objective

- Predictive maintenance of aircrafts

Is an aircraft going to interrupt its operation unexpectedly in the next seven days?

- Data sources
  - Semi-structured raw data
    - Sensor data gathered during flights
  - Structured data

Sources: Sensor Data

- We focus on the navigation subsystem
  - ATA code: 3453
- We provide 376 datasets containing time series data for this subsystem
  - Data is grouped per flight and provided in a single CSV file named as follows: *day-origin-destination-flightNumber-aircraft*
    - Example: `040713-ANS-RIV-8723-XY-YCV.csv`

      day aircraft
  - The file contains measurements every 5 minutes from take-off to landing
- Each row follows this schema:
  - `date;series;value`
  - Example: `2013-07-04 18:12:01.008;eventData;31.89`

Sources: Data Warehouse

- A unified standardized view of retrospective data
- We provide a functioning DW (with data)
  - A possible solution for Lab 1
  - The AircraftUtilization and LogBookReporting store data at a day granularity

Data processing

Data processing

We define three pipelines you must develop

- Data management pipeline
  - Prepares the data and outputs the matrix used in the data analysis pipeline
- Data analysis pipeline
  - Trains a classifier using the matrix generated in the previous step
- Run-time classifier pipeline
  - Given an aircraft and a day, it predicts whether that aircraft will require unexpected maintenance in the next seven days

The data management pipeline (I)

- Ingest, transform and combine the semi-structured and structured data sources
- The generated records must be per aircraft and day, of the form

  `<FH, FC, DM, AVG(sensor)>`

  - Process sensor data (compute the average sensor measurement per aircraft and day)
  - Enrich with KPIs (for that aircraft and day, extract from the Data Warehouse the requested KPIs)
    - Flight Hours (FH),
    - Flight Cycles (FC),
    - Delayed minutes (DM)

The data management pipeline (II)

- Ingest, transform and combine the semi-structured and structured data sources
  - Add a label (supervised learning)
    - (Unexpected) maintenance / no maintenance
    - Consider aircrafts for which you have sensor data
  - Generate a matrix
    - Combine all records to generate a single matrix/dataframe

FH	FC	DY	SensorAVG	Label
54	30	201	75	Maintenance
12	8	66	32	NoMaintenance
77	35	313	85	Maintenance

The data analysis pipeline

- Create two datasets
  - Training
  - Validation
- Use the training dataset to train (experiment with) at least two classifiers
  - Use algorithms from `spark.ml`
  - Evaluate the models
    - Compute recall and accuracy
- Store all the respective data
  - `spark.mlflow` to store/track models, hyperparameters and evaluation metrics
- Rank the models
  - E.g., using predictive accuracy
- Deploy the best model
  - Set as default the best model, so that it can be used in the 'run-time classifier pipeline'

Run-time classifier pipeline

- Given an aircraft and a day, it replicates the data management pipeline
  - Computes the **`avg(sensor)`**
  - Enriches this information with the DW KPIs
- Uses the default model (set as best in the previous pipeline) to predict whether the input record will require unscheduled maintenance in the next seven days
  - Print out the predictions

Additional remarks

- This is a data management lab
  - We will focus on data management aspects
  - This includes whether you follow good practices when preparing the data for the analysis and whether you properly manage/store the data/metadata generated in the analysis pipeline
- The practice is a simplified version of a real-life scenario
  - Feature-selection is not set beforehand
  - A model typically contains many more features
    - More features would entail more flows consolidating data as variables in the matrix
  - Data Lakes or Databases are used to store the data, not Local file systems
  - The run-time classifier pipeline is implemented as an event in a dashboard

Tools

- Python 3
- Spark (pyspark)
  - SparkSQL (DataFrames)
  - Mllib (DataFrame-based, spark.ml)

Timeline

- Start reading the manuals
- Set up the environment
  - Python 3
  - PySpark
  - Connect to the DW from PySpark
  - Read the CSVs from PySpark
- Conduct the training we suggest to get familiar with Spark
- Start devising the pipelines at the conceptual level