xTech @ bip.

Data Science Job Simulation

Introduction & setup

This document serves as a presentation to the work carried out during the Job Simulation proposed by xTech BIP in the field of Data Science.

The programming language used for this challenge is **Python** (version: *3.8.10*). As a preliminary step to be able to properly execute the code is required to install the **Scikit-Learn** and **Pandas** libraries. This operation can be made manually or by executing the following command:

```
pip3 install -r requirements.txt
```

Once the working environment has been properly configured, it is possible to run the file using the following command:

```
python main.py
```

Workflow

Step 1: Problem & data assessment
The first thing tried to do was to get a better understanding of the proposed problem starting from the provided information.In particular, in order to improve the quality of the customer base the machine learning model required must allow the identification of *bad payers*. For this purpose, two datasets (*"Client_Personal_Data.xlsx"* and *"Client_History.xlsx"*) and a dictionary of data (*"DataDictionary_v28.09.xlsx"*) containing information on the various customers were provided.

Step 2: Data preparation
Intensive work has been done on the data at this stage. The first step was to convert the two *xlsx* datasets to *csv* format, much easier to manipulate in Python. Subsequently, on the basis of the information provided, the various dataset fields have been studied in order to identify any anomalies in the data and determine those attributes that most help to characterize a bad payer. The following operations were therefore carried out:

- ***Data selection*:** removal of attributes considered unsuitable or not relevant for the realization of the model. The following attributes have therefore been discarded as little determining in the identification of a bad payer: TERRITORY, COUNTRY, GENDER, AGE, PERSONAL_DATA, ACTIVATION_DATE. It should be highlighted that ACTIVATION_CHANNEL, initially discarded as an attribute, has subsequently been reinstated in the development of the model because its inclusion has led to a slight improvement in the score. Moreover in this phase the ID_CLIENT attribute has been maintained for both datasets since it is used like point of conjunction (primary key) between the information contained in the two files;

- ***Data cleaning:*** control and management of missing values, outlier removal and reformatting of data contained in datasets (*string* to *integer* or *float* to *integer*).

Step 3: Data enrichment
Generate a new dataset ("*dataset.csv*") from the data collected up to this point. Thanks to the ID_CLIENT value contained in the two datasets, the data are gathered in a single csv file on which the training and testing phases of the model will be carried out subsequently.
In addition, at this stage the information is reworked in order to acquire more relevance for the training phase of the model:

- Removal of ID_CLIENT attribute (not required for model training);

- Union of the contained information of the two datasets (a single record for ID_CLIENT);

- Number of active services per customer;

- Calculation of the average number of days of late payment (this value has been tested both as *float* and as *integer*, being more significant in the second format);

- Calculation of the total number of days on which services were used (this value has been tested both in *daily* and *monthly* format, being more significant in the first format;

- Percentage value of the services on which the customer is found to be in default at least once (this value has been tested both in *numerical* and *percentage* format, being more significant in the second format).

Step 4: Model Training & Evaluation
Being a binary classification problem, after having defined the main evaluation parameters, the model chosen for the realization of the challenge is a logistic regression algorithm. For more details

on the output of the script and the results obtained, see section ***Output & performance***.

Step 5: Deployment

Once the required objectives have been reached, the execution of the code has been automated, in particular:

- The model obtained after the first execution is saved ("*model.pickel*") so that it can be used for subsequent executions;

- The generated dataset file is saved, as well as the csv files resulting from the conversions of the respective xlsx files. Please note that in case of data manipulations in the original datasets csv files will not be automatically regenerated.

These minor but substantial changes allow for much faster consecutive model executions (tests showed that the first run takes about 70-80 seconds to generate the various files, while subsequent runs take about 2 to 3 seconds).

Output & performance

Below is an example of output generated by the model (since the dataset on which the model operates is splitted in train and test in a different way for each execution of the code, the values may slightly differ between one execution and the other).

```
PREDICTION SAMPLES
Predicetd: KO - Bad payer - Expected: KO - Bad payer
Predicetd: OK – Good payer - Expected: OK – Good payer

MODEL METRICS
Accuracy score:     0.97603
Recall score:       0.89295
```

The section dedicated to PREDICTION SAMPLES has been inserted to demonstrate the actual functioning of the model developed. In particular, two records were randomly taken from the dataset.csv file (whose data has therefore already been prepared) whose expected outputs are known. A prediction was then made on the information provided to verify that the output predicted by the model matched the expected output.
The section MODEL METRICS instead shows the values of the metrics chosen for the evaluation of the model. Considering the problem presented, the metrics that it was considered appropriate to use are *accuracy score* and *recall score*.
In order to explain the reasons for such choices and their relative goodness it is necessary to introduce briefly the concept of *confusion matrix* for a classification model.
In practice, the confusion matrix represents all the possible outcomes of a prediction carried out by a machine learning model. In the specific case of a binary classification algorithm the possible outcomes are the following:

		EXPECTED VALUE	
		Positive	Negative
PREDICTED VALUE	Positive	*True Positive (TP)*	*False Positive (FP)*
	Negative	*False Negative (FN)*	*True Negative (TN)*

A good parameter to understand the goodness of the developed model is definitely the accuracy score. This parameter represents the ratio of the correct predictions of the algorithm to the number of total predictions. In practice we are given a decimal value of how precise the model is in general predictions. In more technical terms we have that:

$$Accuracy\ score = \frac{TP+TN}{TP+TN+FP+FN} = 0.97$$

The developed model recorded an accuracy score of about 0.97, so about 97% of the predictions made were found to be correct.
However, the purpose of the model is to identify in the most accurate way possible the cases of morosity (which in the confusion matrix are represented by only TP) and not even those of "non

morosity" (represented by TN). It was therefore necessary to identify a metric that would give greater prominence to this dynamic. The choice of this parameter fell on the recall score:

$$Recall\,score = \frac{TP}{TP + FN} = 0.89$$

So the recall score analyzes the relationship between the only cases of delay predicted (TP) compared to cases of total delay (TP and FN). In this way the report considers only the cases of interest for the request of the challenge, obtaining a more truthful and representative result (about 89%) of the predictive abilities of the model with respect to the proposed problem.