**Machine Learning Engineer Technical Challenge**

1. **Objective**

The main purpose of this challenge is to assess your skills **(1)** building a scalable data pipeline for feature engineering, **(2)** creating an API to serve those features, and **(3)** creating a prediction service that uses the created features.

1. **Data & technology**

* You will be handed a dataset from Kaggle containing the credit default risk, a jupyter notebook containing code with some feature engineering and the training of a machine learning model.
* We require you to use pyspark for the feature engineering. Therefore you will need to have it installed and configured in your environment (we recommend using a Docker image).

The columns and their description of the dataset used are described in the references, on the last page of this document.

1. **Problem context**

A data scientist has engineered five new features that he will use for the development of some ML model. The code used to compute these features is contained in the jupyter notebook 1 you were handed. The problem here is that these features are not coded in an efficient way, and it takes just too long to compute them for the whole credit risk dataset. You will need to help the data scientist compute these features in a scalable way, so that he or she (and the rest of the data scientists) can have these features available in a file or table to do things like building training datasets using them. Also, those features might be used by ML models on their online evaluations, so you will need to create an API to serve the features. Having said this, we will now go to the detailed instructions of the challenge.

1. **Instructions**

**4.1. Understand the features (notebook 1)**

Take a look at the jupyter notebook, where the data scientist explains the five functions he created and the basic machine learning model he trained. Try to understand the features he wants to compute, and the approach he had to compute them.

**4.2. Build data pipelines to compute these features and train machine learning model (using notebook 2)**

Using PySpark, build a data pipeline which extracts data from the credit risk dataset, and creates a table with six columns: the user id (id column) and the five features defined in the jupyter notebook. The pipeline should be built with the following considerations: The table created should contain all the rows of the credit risk dataset. The pipeline should be optimized, so that the table can be computed in a more reasonable amount of time. The table you create with the six columns containing the features and user id, can be stored in the way you find more convenient: on a file (avro, parquet, csv, etc.) or a SQL table, etc. Explain the reason of your choice. Are you sure (or at least to some point) of the quality of your output?

**4.3. Create an API to serve the features**

ML models might use those five features for online evaluation. We would like to have an API serving the features.

Create a local API which:

* receives: the **id** of a user
* returns:
* **nb\_previous\_loans**: number of loans granted to a given user, before the current loan.
* **avg\_amount\_loans\_previous**: average amount of loans granted to a user, before the current loan.
* **age**: user age in years.
* **years\_on\_the\_job**: years the user has been in employment.
* **flag\_own\_car**: flag that indicates if the user has his own car.

**Consider** that each user will have features at different moments of time, so you must bear in mind that only the most recent (last date per user) are of interest for the prediction.

**4.4. Create an API to make predictions**

Create an API that, given a user id, makes a prediction of credit risk using the model trained in the notebook 2. Consider using the features already created and available on the endpoint, in point 4.3.

At the end of notebook 2 it is shown how to load a model into memory from the already trained model and make a prediction.

**4.5. Automate ML Deployment Pipeline**

The model does not need to be the best, we will evaluate how this component will be implemented and integrated into MLOp Typical workload, here an [example](https://mlops-safe-deployment-pipeline.workshop.aws/). It is a plus using cloud Technologies such as AWS (Code Pipeline, State Machine).

**To consider:** *In this exercise there might be some things which are not a very good representation of a real life use case (for example: the structure/format of the source data, the architecture of the API and the data pipeline, the infrastructure, the QA considerations, etc.). Tell us what considerations you would have regarding this API for a real life situation.*

*In the event that the exercise is not resolved 100%, we will consider the form adopted for the resolution, proposed design, etc.*

**References:**

Columns of the dataset used in notebook 1.

**loan\_id:** loan number

**id**: client number

**code\_gender**: gender

**flag\_own\_car**: if users have a car

**flag\_own\_realty**: is there a property

**cnt\_children**: number of children

**amt\_income\_total**: annual income

**name\_income\_type**: income category

**name\_education\_type**: education level

**name\_family\_status**: marital status

**name\_housing\_type**: way of living

**flag\_mobil**: is there a mobile phone

**flag\_work\_phone**: is there a work phone

**flag\_phone**: is there a phone

**flag\_email**: is there an email

**occupation\_type**: occupation

**cnt\_fam\_members**: family size

**status**: 0->paid on time 1->Not paid on time

**birthday**: birthday

**job\_start\_date**: job start date

**loan\_date**: loan date

**loan\_amount**: loan amount