# Machine Learning Assignment

## Introduction

A ML flow/pipeline consists of a sequence of stages until the modeling one. In Optasia, every pipeline step plays a crucial role in our ML platform and the 'Feature engineering' step is obviously an integral part of it. You should implement one of the two following projects, however, feel free to do both if you prefer so.

## 1) Feature Engineering API

The objective of this assignment is to create a feature engineering module that generates some features/variables that are useful for modeling in our business case. The python module should be called via an API endpoint, returning some features/variables for the customers of the given dataset(cvas_data.json) file.

### Technical problem description

**Goal**: The goal of this project is to create an API (preferably with fastAPI framework) for a feature engineering process.

**Tech stack**:
Python
Docker (preferably compose)

**Endpoints**: The API should have two endpoints. The first one should return the results of the 'feature engineering' process in a JSON format.
The second should check the health of the API returns only {status: "UP"} if the API is up and running.

**Input**: The input data is given in JSON format (the attached cvas_data.json file)

**Output**: The output data should be in JSON format.

**Data**:
The JSON file includes transactional data/ loan records from a bunch of customers.

**Columns Details**:
1. ***customer_ID***: A unique identifier for the customer. Customers may have more than one loan
2. ***loan_date***: The date of loan.
3. ***amount***: The loan amount that was credited
4. ***term***: A Categorical variable indicating the kind of loan. Its values are "long" and "short".
5. ***fee***: The fee of the loan.
6. ***loan_status***: A categorical variable indicating if the loan was paid back(0) or defaulted(1).
7. ***annual_income***: The customer's annual income

**Docker**: The API should run in a docker container.

**Unit tests/Logger**: The implementation of some unit tests as well as logger will be considered plus.

**Tip**: If you want to explore more about feature engineering techniques you could use the featuretools library for automated feature engineering process.

**Extra**: If you want to do more complex things with the API feel free. For example you could implement an extra endpoint to do something else.

## 2) Feature Engineering with PySpark

The objective of this assignment is to implement a distributed system that handles csv data, applies transformations and feature engineering and persists them in parquet format. The system should consist of the
following two modules:
1. A distributed file system (Hadoop cluster) where the csv dataset-file will be stored and the resulting parquet files will be persisted.
2. A spark cluster that will run on top of Hadoop and will process the csv data in order to generate new features that will be stored in parquet files.

### Technical problem description

**Goal**: The goal of this project is to create a (py)spark job to make feature engineering process in a distributed manner.

**Tech stack**:
Spark
Python
Docker (preferably compose)

**Input**: The input data is given in csv format (the attached cvas_data.csv file)

**Output**: The output data will be in parquet format.

**Data**:
The csv file includes transactional data/ loan records from a bunch of customers.

**Details**:
The first main objective is to setup the Hadoop + Spark cluster. If you have some familiarity with dockers you may setup a multi-node cluster with the help of this technology. Otherwise you can setup a single-node cluster at a single VM. Spark can be setup over yarn or standalone.
You could find more info at the below sites and repositories:
https://hadoop.apache.org/docs/r3.2.1/
https://archive.apache.org/dist/hadoop/common/
https://spark.apache.org/releases/spark-release-3-1-1.html
https://archive.apache.org/dist/spark/

The input data is given in CSV format (the attached cvas_data.csv file). A spark job should be submitted at Spark cluster that will read the input csv data and produce the parquet output with columns:

**Columns Details**:
1. ***customer_ID***: A unique identifier for the customer. Customers may have more than one loan
2. ***loan_date***: The date of loan.
3. ***amount***: The loan amount that was credited
4. ***term***: A Categorical variable indicating the kind of loan. Its values are "long" and "short".
5. ***fee***: The fee of the loan.
6. ***loan_status***: A categorical variable indicating if the loan was paid back(0) or defaulted(1).
7. ***annual_income***: The customer's annual income

**Docker**: The job should be run in a docker container or in a VM.

**Unit tests/Logger**: The implementation of some unit tests as well as a logger will be considered plus.

## DELIVERABLE

With this assignment we want to see how you formulate a solution for the problem and how you implement it code-wise. You should implement the one of the two projects. You could try to do both but it is not mandatory. The final deliverable should include the following:

• A private repo in GitHub. We only need read-only access to the repo

(invite as collaborator **sofoklis-7@hotmail.com**).

• A **README.md** file with instructions of how to setup, build and execute all modules.

Your deliverable will be examined based on the requirements of the assignment. Expect to be asked about your design and implementation choices, assumptions and issues that you may have encountered.

In case you have any questions/problems with the project please contact: **sofoklis.tolikas@optasia.com**