FairMoney - Test machine learning engineer

Intro

You will be given a small dataset credit.csv with data of customers applying for a loan. The goal is to predict their defaulting behavior. The data structure described in the Data section of this document.

The goal is to write scripts to train simple models to predict the default behavior of customers, and build a minimalist webserver to serve the trained models.

The test should be written in python, and sent as a zip file, with a README.md file with instructions to run the training scripts and deploy the web server.

We will evaluate the quality of the implementation, the ease of training reproducibility and server / model deployment, and every additional comment (next steps, technical choices, scripts, evaluation reports…).

We expect you to spend about 2 hour on this test (30min for simple model training, and 1h30 to write the web server and clean the training pipeline).

Data

The target is the variable default.
The data has the following structure:

- Observation_id: unique observation id
- Checking_balance: Status of existing checking account. (German currency)
- Savings_balance: Savings account/bonds (German currency)
- Installment_rate: Installment rate in percentage of disposable income
- Personal_status: Personal status and sex
- Residence_history: Present residence since
- Installment_plan: Other installment plans
- Existing_credits: Number of existing credits at this bank
- Dependents: Number of people being liable to provide maintenance for
- Default: 1 is a good loan, 2 is a defaulting one.

Specifications

- Training

    - A script to split the csv file into train / test files (70% into train.csv / 30% into test.csv)
    - A script to train a model and save it locally (format: {experiment_id}_{training_timestamp})'
    - (optional) A script to output a quick model training report (AUC, GINI, ROC curves, Precision / recall…)
- Deployment
    - The webserver will have the following routes
        - GET /models -> Outputs a list of model_id that are trained and available
        - POST /output/<model_id> -> Returns the output of the model on the data sent one data instance (the input data could be a row from the csv file in json format, or whatever is more convenient)
        - (optional) POST /outputs/<model_id> -> Outputs the outputs of the model on a batch of data
- (optional) Integration
    - Write a script which loads the data from the test.csv file, query the web server and outputs an AUC score for the test set.