EFREI PARIS
M2APP-BDIA

Applications of Big Data (ADDE92)
Project : Home Credit Risk Classification

Authors :
Farès FADILI
Nour-Eddine OUBENAMI
Adrien TARCY

Referent :
Medina HADJEM

The goal of this project is to apply some concepts & tools seen in the three sessions of this course; this project is organized into three parts :
1) Building Classical ML projects with respect to basic ML Coding best practices
2) Integrate MLFlow to your project
3) Integrate ML Interpretability to your project

January 23, 2023

Table of contents

Introduction

Dataset (Finance use case) :

Dataset of Home Credit Risk Classification:
https://www.kaggle.com/c/home-credit-default-risk/data
We did not use all the datasets available on Kaggle, only the main data set :
⇒application_train.csv
⇒application_test.csv
We also used a reduced version of these datasets

Requirements :

Linux OS is recommended, an IDE with last python version (using of anaconda environment for example).

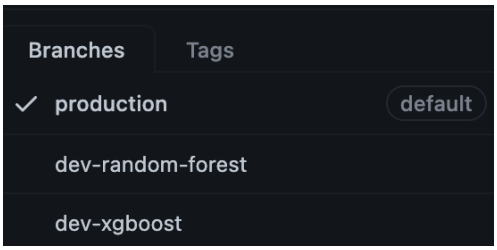
1. Build an ML Project for Home Credit Risk Classification

1.1. Coding best practice for production ready code

Throughout the project, we use GIT for team collaboration, code and model versioning, we divide the machine learning project workflow into different scripts (data preparation, feature engineering, models training, predict), we use templates or create our own, a conda environment for all of your libraries (or any other package/environment management like poetry), and Sphyx library for documentation.

1.2. Choose of one classical ML algorithm

We tested all three models (see Git branches), but after various tests we chose Gradient Boosting because of the simplicity of its implementation and the good results it brought.



The pipeline solution we chose was Apache Air Flow, as it is easy to use, performs well and can be used for ML-OPS.

2. Integrate ML Flow to the Project

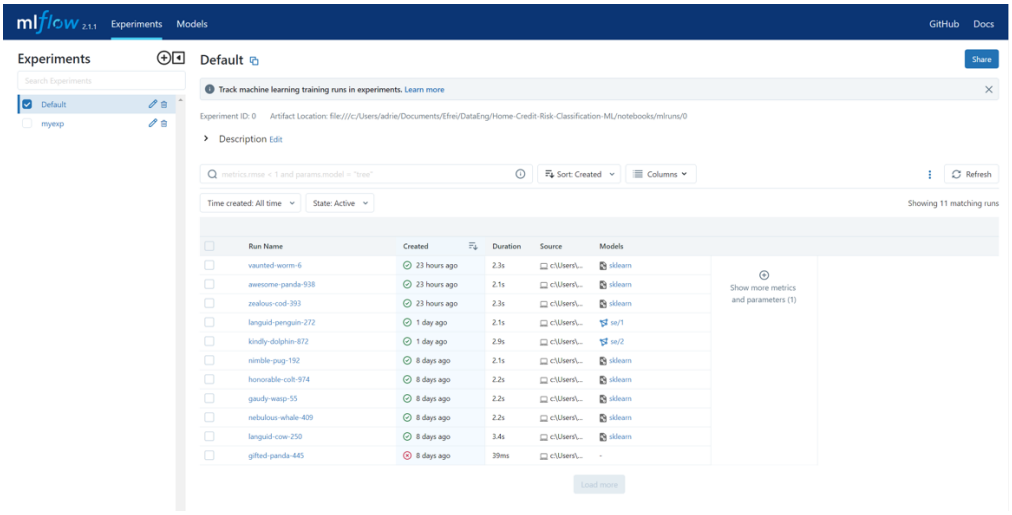
2.1. Track parameters & metrics of the model

We record the accuracy, the loss and our Gradient Boosting model:

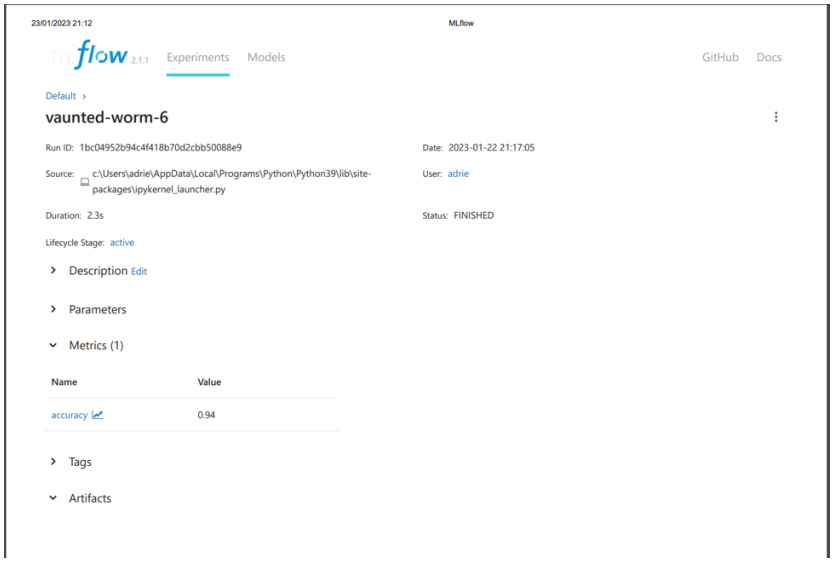
```
mlflow.log_metric("accuracy", model.score(x_test, y_test))
# Log the log loss of the model
mlflow.log_metric("log_loss", log_loss(y_test,model.predict_proba(x_test)))
# Log the model
signature = infer_signature(x_train, clf.predict(x_train))
mlflow.sklearn.log_model(model, "model_test",signature = signature)
```

2.2. Display the results in the local ML Flow UI

Once the program has been launched, the various metrics are available at the following address: http://localhost:5000.



Here is the output of one of the experiments on MLFlow:



2.3. Package the code in a reusable and reproducible model format

We have created a file MLProject; this file contains a command to call a Python file as well as parameters (arguments) and calls a file environment.yaml.
The environment.yaml file allows to download only the necessary libraries. In our case, these are the Pandas, Numpy, Scikit-Learn, Joblib, Pip and MLFlow libraries.

```yaml
! environment.yml
name: base
channels:
  - defaults
dependencies:
  - pandas
  - numpy
  - scikit-learn
  - joblib
  - pip
  - pip:
    - mlflow
```

2.4. Deploy the model into a local REST server that will enable to score predictions

We have developed a Rest MLFlow API. It allows to create, list and get experiments and executions. It also allows to log metrics and artifacts. To search for experiments on the MLFlow server, you just have to send a POST request to localhost:5000/api/2.0/mlflow/experiments/search.

The deployment of the model is done via the Rest_API.py file.

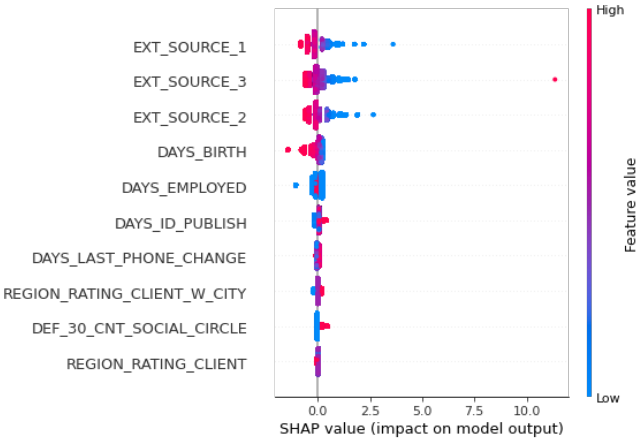
Example of Rest API function:

```python
def search_experiments(self):
    """Get all experiments."""
    url = self.base_url + "/experiments/search"
    r = requests.get(url)
    experiments = None
    if r.status_code == 200:
        experiments = r.json()["experiments"]
    return experiments
```

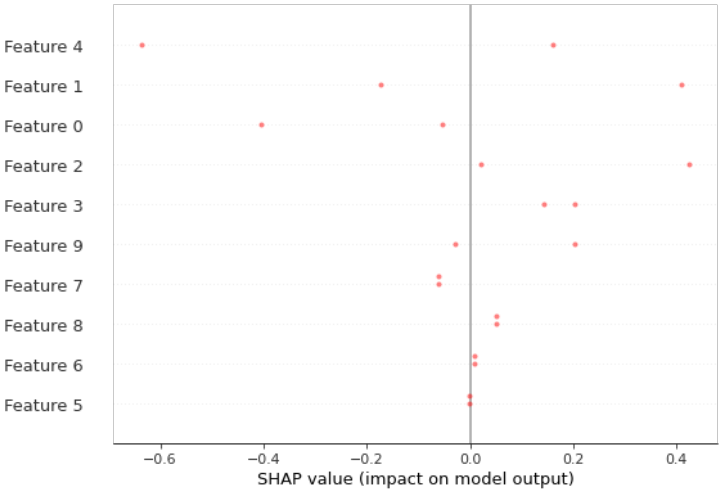
3. Integrate SHAP Library to the Project

Explanations are available in the reports/figures report. Here are some examples:

Comparison of features on the impact on the model



SHAP values (impact on the model output) of features on a specific point of the dataset



Depedencies between the different features