Machine learning & anomaly detection

M. Mougeot

ENSIIE, November 2020

Contents

Introduction **2**
- Goal of the practical sessions . . . 2
- Warnings and Advices . . . 2
- Python libraries . . . 2

The data **3**
- Data set 1 . . . 3
- Data set 2 . . . 3

Isolation forest **4**

Local Outlier factor **5**

Density based Anomaly detection **6**

Introduction

Goal of the practical sessions

- To understand anomaly detection machine learning methods, from a methodological and practical point of view.
- To apply models and to tune the appropriate parameters on several data sets using the 'Python' language.
- To interprete 'Python' outputs.

Warnings and Advices

- The goal of this practical session is not "just to program with Python"" but more specifically to understand the framework of Modeling, to learn how to developp appropriate models for answering to a given operationnal question on a given data set. The MAL course belongs to the **Data Science courses** . For each MAL practical session, you should **first understand** the mathematical and statistical backgrounds, **then write your own program with 'Python'** to practically answer to the questions.

Python libraries

For this practical session, the following librairies need bo be uploaded in the python environment.

```
import numpy as np
from sklearn.ensemble import IsolationForest
from sklearn import datasets
from sklearn.datasets import make_blobs
from sklearn import preprocessing
import matplotlib.pyplot as plt
from numpy import quantile, where, random
import statistics as stat
import random as rd
```

The data

Data set 1

```
n=400;
X0,y0=datasets.make_moons(n_samples=n,noise=0.05);
X0 = preprocessing.scale(X0)
```

Add 10% of uniform distributed observations to the previous data to built the data set 1.

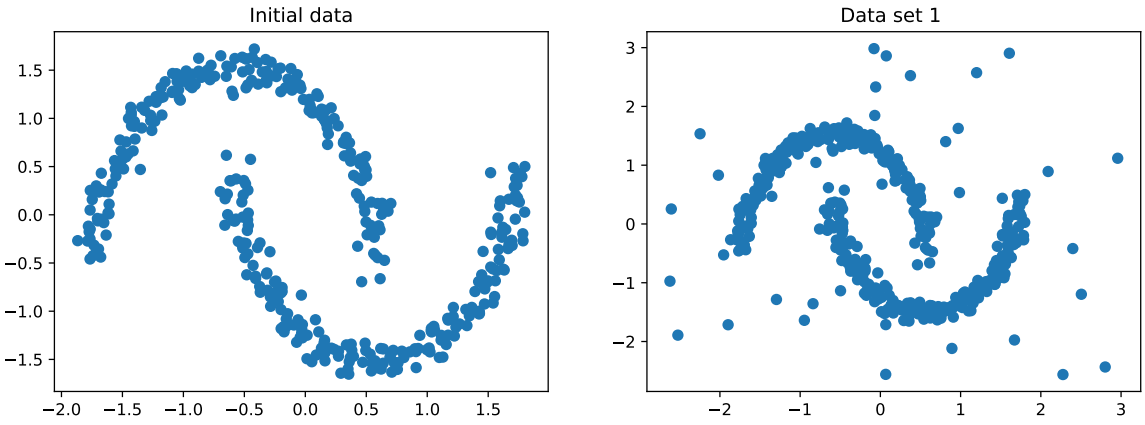

Figure 1: Data (left: initial data; right: data with noise.

Data set 2

```
NG1=200;
mx1G1=0; mx2G1=0; sx1G1=1; sx2G1=1; rhoxG1=0.95;
mu1 = [mx1G1, mx2G1];
covxG1=[[sx1G1**2, rhoxG1*sx1G1*sx2G1], [rhoxG1*sx1G1*sx2G1, sx1G1**2]];
XG1 = np.random.multivariate_normal(mu1,covxG1,NG1);
X0 = preprocessing.scale(XG1);
```

Add 10% of uniform distributed observations to the previous data to built the data set 1.

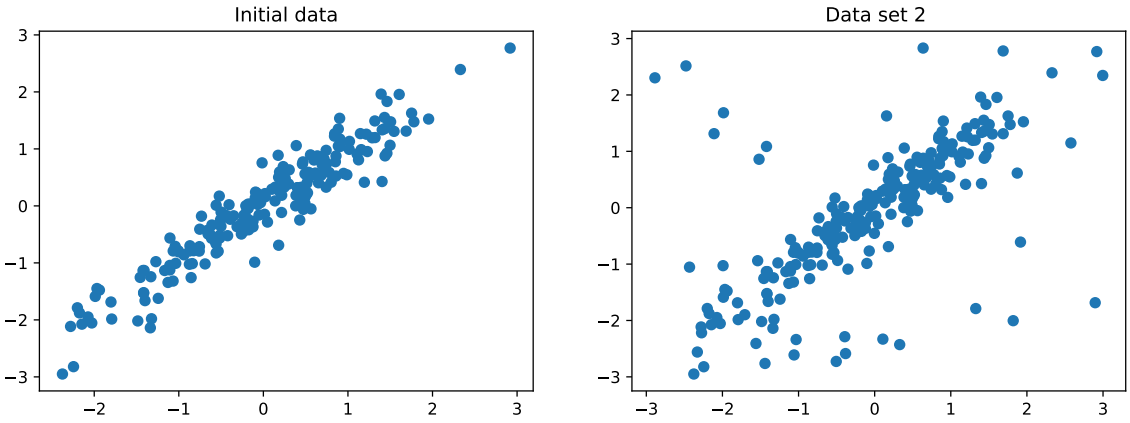

Figure 2: Data (left: initial data; right: data with noise.

Isolation forest

-Study the help of the IsolationForest() function of the scikitlearn library and use the function to detect ouliers in the previous data sets 1 and 2.

```
#Anomaly detection with score
iforest = IsolationForest(n_estimators=100)
tmp=iforest.fit(X)
scores = iforest.score_samples(X)
```

-Detect the outliers with a score lower that the quantile value of order 5%.

-Data set 1:

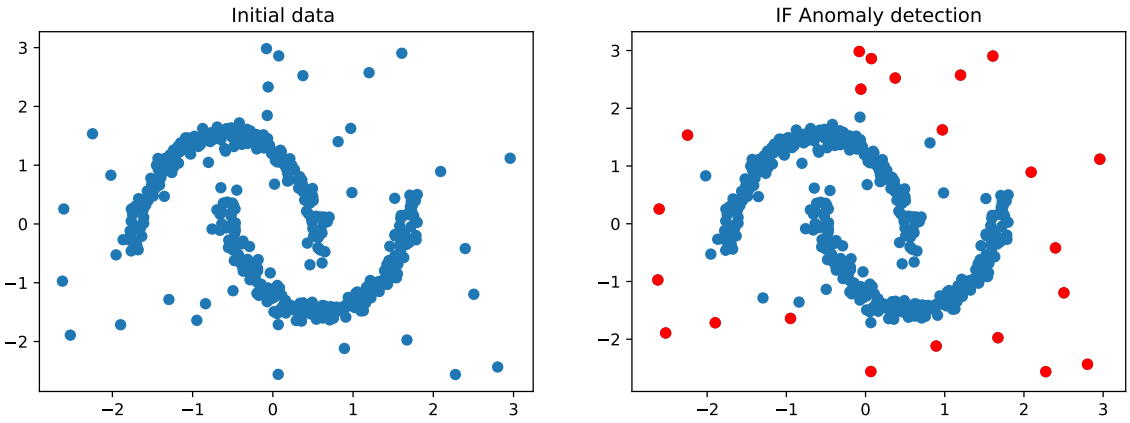

Figure 3: Anomaly detection

-Data set 2:

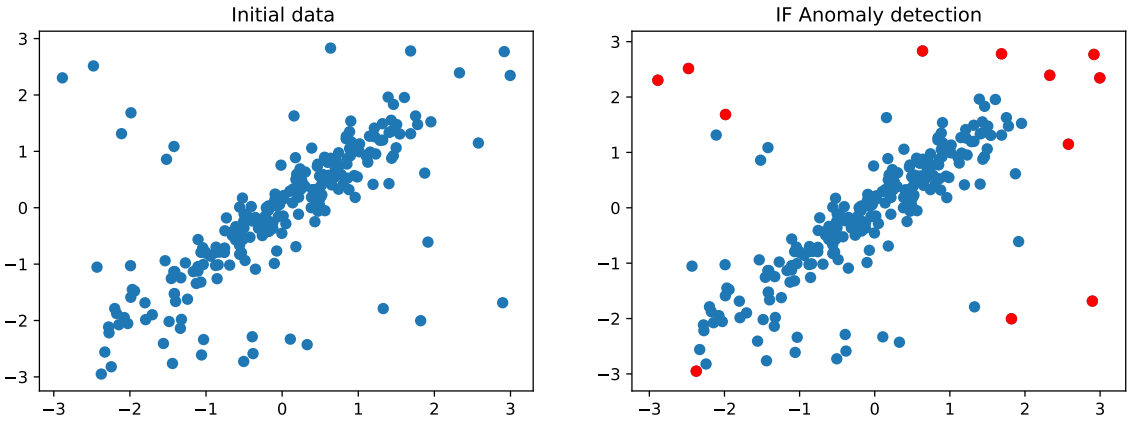

Figure 4: Anomaly detection

Local Outlier factor

Study the help of the `LocalOutlierFactor()` function of the scikitlearn library and use the function to detect ouliers in the previous data sets 1 and 2.

```
from sklearn.neighbors import LocalOutlierFactor
lof = LocalOutlierFactor(n_neighbors=20, contamination=.01)
y_pred = lof.fit_predict(X)
```

-Detect the ouliers with a score equaled to -1.

-Data set 1:

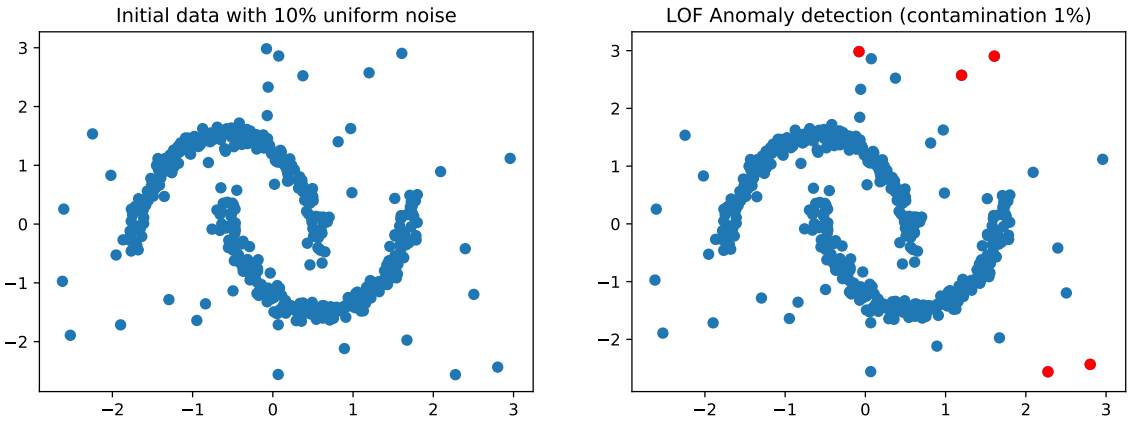

Figure 5: Anomaly detection

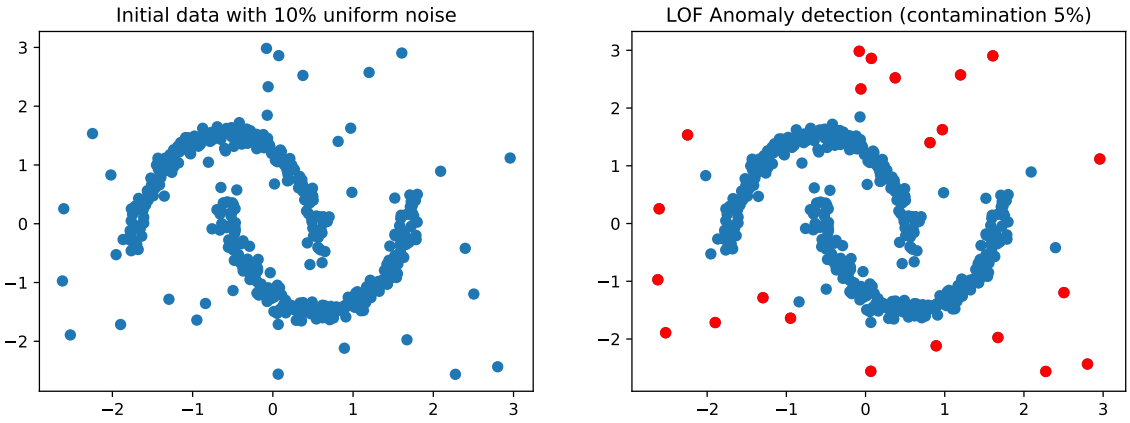

Figure 6: Anomaly detection

-Data set 2:

-Detect the ouliers with a score equaled to -1.

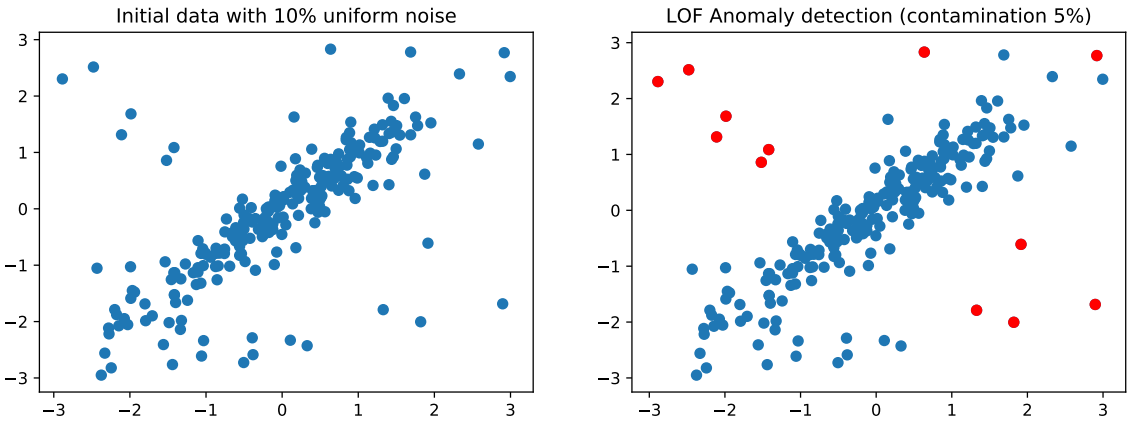

Figure 7: Anomaly detection

Density based Anomaly detection

Study the help of the `EllipticEnvelope()` function of the scikitlearn library.

```
from sklearn.covariance import EllipticEnvelope
mod=EllipticEnvelope(contamination=0.05);
y_pred =mod.fit_predict(X);
```

-Detect the outliers with a score value equaled to -1.

-Data set 1:

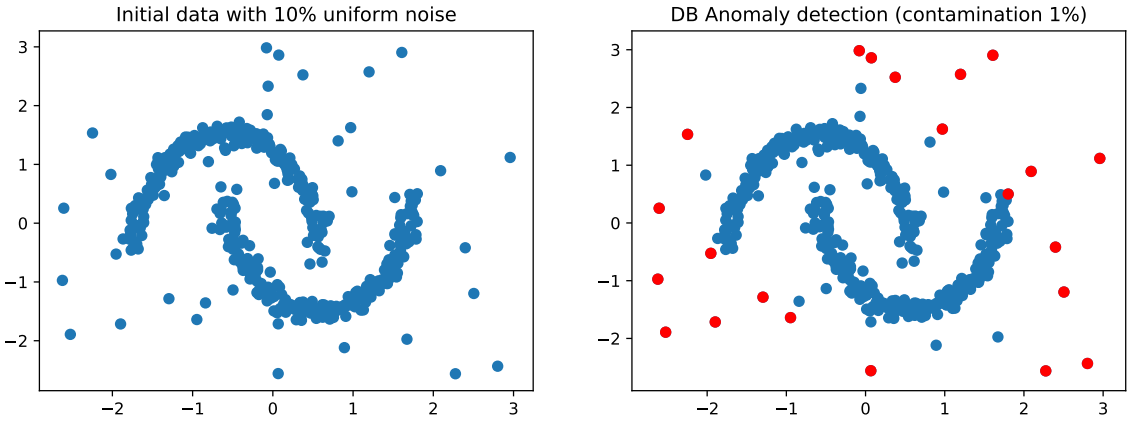

Figure 8: Anomaly detection.

-Data set 2:

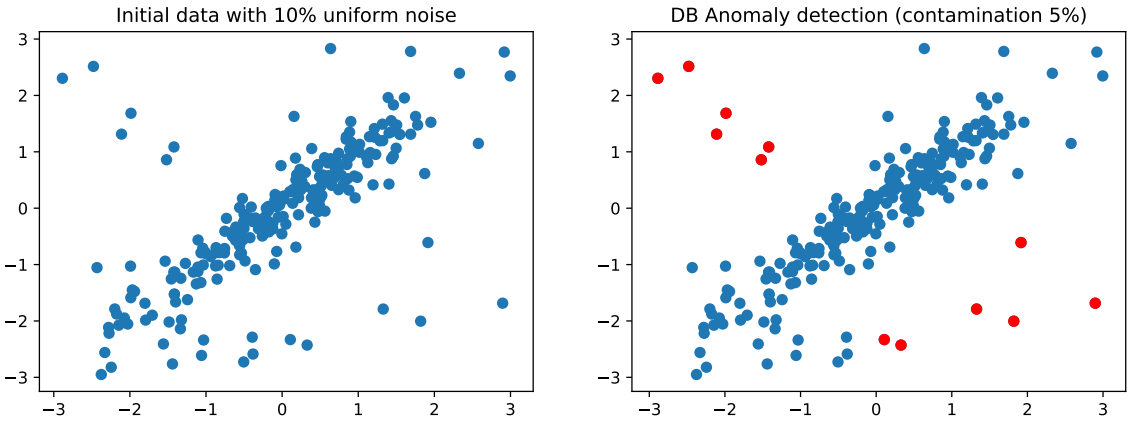

Figure 9: Anomaly detection.