Machine learning for classification. First step.

M. Mougeot

ENSIIE, September 2023

Contents

Introduction

Goal of the practical sessions

- To understand classification 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.

Instructions

- The practical work must be carried on with a ‘group of two students’.
- The MAL project aims to developp a Jupyter notebook to solve a classification problem (the subject will be given soon). Your names have to be written in the first lines of the program file with comments. Please note that without this information, no grade will be attributed to the missing name porject.

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

```
import random as rd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import statistics as stat
import math
```

The data

With the help of function *gauss*() of the library `random`, or function *random*() of the library ‘numpy, simulate a two dimensional sample of size $N = 200$ as illustrated in Figure 1 (left). Following instructions may help:

```
N=500
xgauss1=[rd.gauss(0,1) for i in range(N)]
xgauss2=[rd.gauss(0,1)  for i in range(N)]
```

A labeled Data Set

Simulate a two dimensional sample composed of a mixture of two gaussians as illustrated in Figure 1 (center).

The group I (red points) contains $n = 100$ observations distributed as $\mathcal{N}(\left[\begin{array}{cc} 2.5 & 2.5 \end{array}\right], \left[\begin{array}{rr} 2 & -0.8 \\ -0.8 & 2 \end{array}\right])$. The label $Y = 1$ is defined for all observations of group I.

The group II (blue points) contains $n = 200$ observations distributed as $\mathcal{N}(\left[\begin{array}{cc} 0 & 0 \end{array}\right], \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right])$. The label $Y = 0$ is defined for all observations of group II.

- Compute the average, the standard deviation and the corelation of the input for the two classes. What do you observe compared to the theoretical parameters of the distribution ?

New data inputs without label

- In order to visualise the MAP decision boundaries. A grid of $N = 15 * 15$ inputs is generated with regularly spaced points computed on the support on the previous training data set.

```
#Eval data
x1_min, x1_max = min(x1), max(x1)
x2_min, x2_max = min(x2), max(x2)
Neval=15; h1= (x1_max-x1_min)/Neval; h2=(x2_max-x2_min)/Neval
x1Eval, x2Eval= np.meshgrid(np.arange(x1_min, x1_max, h1), np.arange(x2_min, x2_max, h2));
```

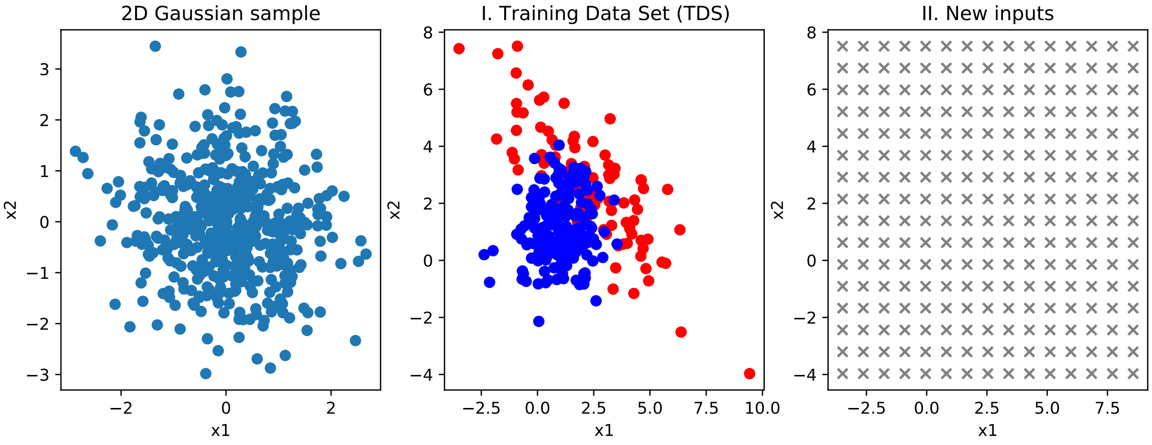

Figure 1: Data sets

A first model of machine learning. The naive Bayes model

Detailed implementation

The goal of this section is to write 'your own detailed python instructions' to implement the Naive Bayes model for 2D inputs. In this section, the use of the SciKit Learn library in not allowed (see later).

Calibration of the model given a training data set

Write the Pyhton instructions to compute: - the prior probability for each given class - the Likelihood probability with each input for each class - the posterior probability of each observation - the predictive class with the Maximum A Posteriori (MAP) decision using the calibrated naive Bayes model. Each observation is assigned to the class with the highest probability.

Figure 2 illustrates the different steps of the method.

Evaluation of the model

Number of errors, number of False Positive and False negative observations

Write the instructions to compute the values of the indicators on the training data set for the naive Bayes model.

Accuracy, Precision, recall and F1-score

Write the instructions to compute the values of the indicators on the training data set for the naive Bayes model.

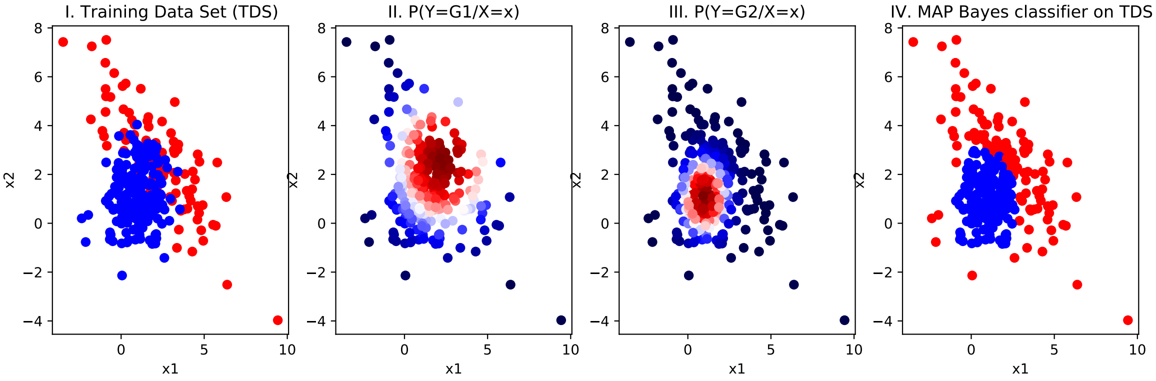

Figure 2: The naive Bayes model: from initial data to MAP decision

Class Prediction for new inputs using the calibrated naive Bayes model

- In order to visualise the MAP decision boundaries. A grid of $N = 15 * 15$ inputs is generated with regularly spaced points computed on the support on the previous training data set.

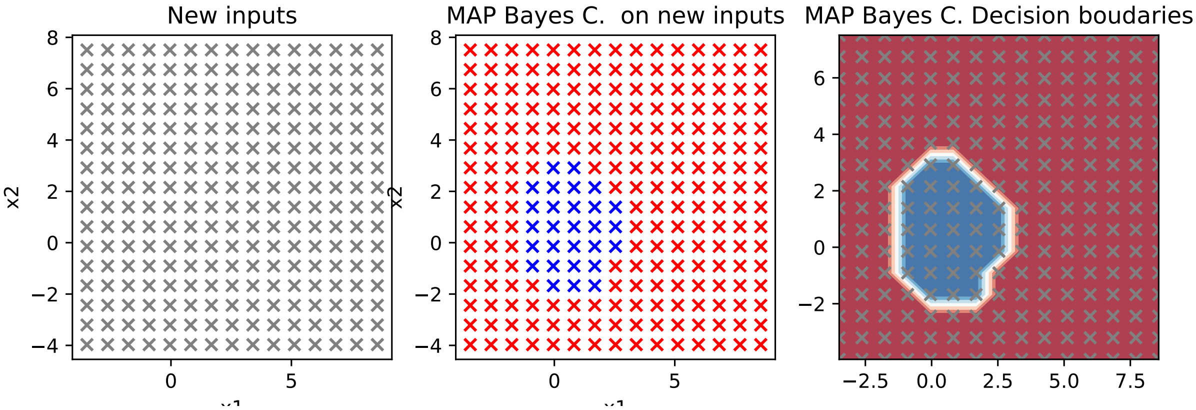

Figure 3: The naive Bayes model: from initial data to MAP decision

The Scikit learn library

The scikit learn library provides many machine learning functions to solve applications. The purpose of this section is to learn how to use sklearn functions to implement, calibrate, and evaluate machine learning models. This work begins with the naive Nayes model which is implemented by the $GaussianNB()$ function of library `sklearn`.. Dedicated libraries need to be upload in the python environment before their use.

Calibration and fit of the model using Scikit learn

Use the following instructions to calibrate the naive Bayes model given the complete data set using the functions of Sklearn.

```
from sklearn.naive_bayes import GaussianNB
#initial data set: X (inputs), y (output).
gnb = GaussianNB();
gnbfit=gnb.fit(X, y);
y_pred = gnbfit.predict(X)
```

Model parameter analysis.

Use the attributes *class_count_*, *class_prior_*, *classes_*, *theta_*, *sigma_* of 'GaussianNb()' to recover the parameters of the naive Bayes model. Compare the values of the field with the values obtained by direct computation in the previous section. Are there any differences ? What is your explanation ?

How can it be useful to analyse the parameter values of the model ?

First Evaluation of the model on training data

Accuracy, precision, recall and F1-score

Use the appropriate 'metrics' attributes of the 'sklearn' library to compute the accuracy, the precision, the recall and the F1 score on the training data set for the naive Bayes model. Compare the computed values with the previous section. Conclusion.

```
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(Y, y_pred))
```

```
## Accuracy: 0.8533333333333334
```

AUC indicator.

Use the appropriate 'metrics' attributes of the 'sklearn' library to compute the Area Under the Roc Curve (AUC).

The ROC curve

Withe the help of the following instructions, compute and visualize the ROC curve on the training data set as in Figure 4.

```
y_proba = gnbfit.predict_proba(X)[:,1];
fpr, tpr, thresholds = metrics.roc_curve(y, y_proba);
figure = plt.figure(figsize=(4,2));
plt.plot(fpr,tpr, linewidth = 2);
plt.title('Naive Bayes. Roc curve')
plt.xlabel('fpr');plt.ylabel('tpr')
plt.plot(tpr,tpr,"k--", linewidth = 2);
plt.grid(linestyle = 'dashed'); plt.show();
```

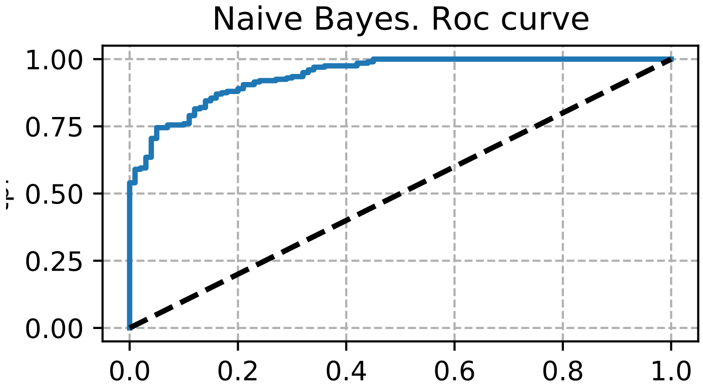

Figure 4: The ROC curve

```
auc = metrics.roc_auc_score(y, y_proba); #print(auc)
```

Evaluation of the predictive power/ capabilities of the model

One of the main purpose of predictive models is to be able to provide good performances on new observations, which have not been already used in model calibration. As the amount of labeled data is most often limited, a commun approach is to split the initial data in two sets: one set to calibrate the model (the training data set), one set to evaluate the performance of the model on independant labeled data (the test data set).

Cross validation

Split randomly the initial data set $\mathcal{D}$ into two sub data sets $\mathcal{D}_{Train}$ and $\mathcal{D}_{Test}$ containing respectely 2/3 and 1/3 of observations. Calibrate the naive Bayes model using $\mathcal{D}_{Train}$ then compute the perfomances on $\mathcal{D}_{Train}$ and $\mathcal{D}_{Test}$.

The following instructions may help to calibrate and evaluate the performances of the naive Bayes model given train and test data sets using the functions of Sci-Kit Learn.

```
#initial data set: X (inputs), y (output).
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=0)
gnb = GaussianNB();
gnbfit=gnb.fit(X_train, y_train);
y_pred = gnbfit.predict(X_test)
E_test=(y_test != y_pred).sum()/len(y_test)
print("Error on the test data set %5.2f->",E_test)
```

Repeat the previous methodology $K = 5$ times (i.e. K random splits of both test and training data with similar proportions) and provide a graph of both training and test performances for the different repetitions. Conclusion.

$Kfold$ methodology

Using the Kfold attribute of the model_selection of sklearn,

```
from sklearn.model_selection import KFold
```

compute the different values of the classification criteria and reproduce Figure 5.

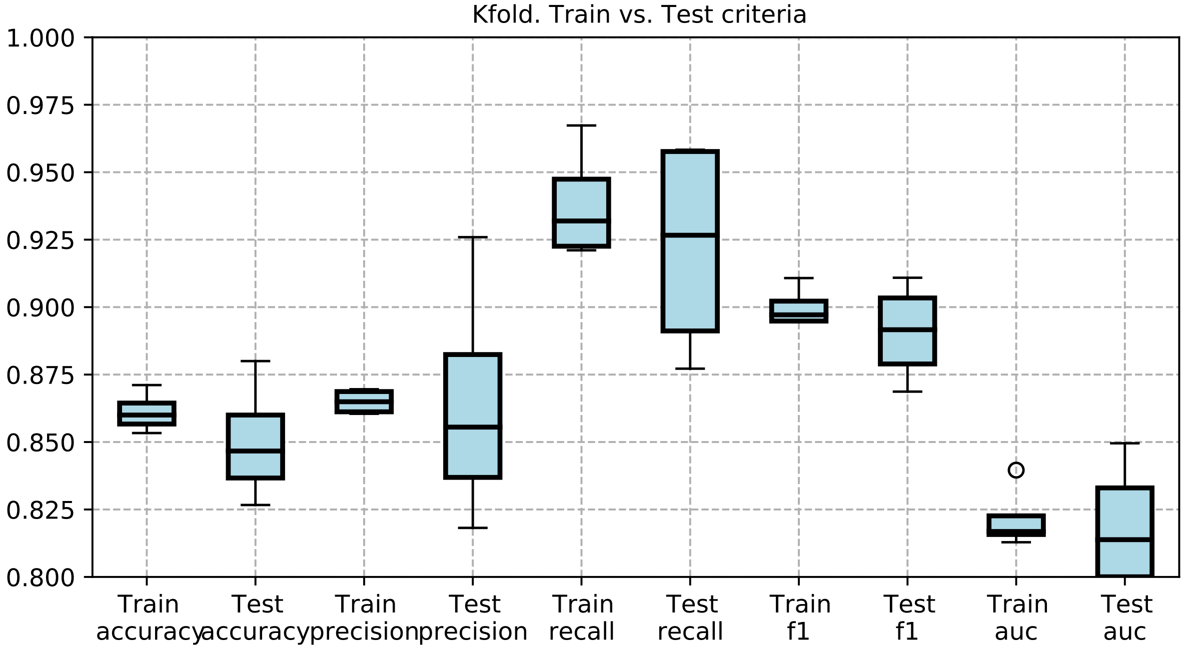

Figure 5: K fold criteria evaluation

Conclusions

Based on the different values of the criteria (Accuracy, precision, recall, F1-score, auc) computed for the training and the test data sets, provide your own conclusions on the ability of the naive Bayes model, and more generally of a machine learning model to perform predictions.

Classification learning machines

Linear Discriminant Analysis

In this section, the linear Discriminant method (LDA) is used to built the predictive model.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components=2);
```

With the help of the sklearn library, calibrate a LDA model on the training data set. What are the available parameter and attributes of the `LinearDiscriminantAnalysis()` ? Display, study and comment the parameters (means and covariance) of the statistical/ machine learning LDA model? Compute the accuracy, precision, recall and f1-score on the training data set. Visualize the decision boundaries of the calibrated model.

Quadratic Discriminant Analysis

In this section, the Quadratic Discriminant method (QDA) is used to built the predictive model.

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
qda = QuadraticDiscriminantAnalysis();
```

With the help of the sklearn library, calibrate a QDA model on the training data set. What are the available parameters and attributes of the `QuadraticDiscriminantAnalysis()` ? Display, study and comment the parameters (means and covariance) of the statistical/ machine learning QDA model? Compute the accuracy, precision, recall and f1-score on the training data set. Visualize the decision boundaries of the calibrated model.

Logistic regression

In this section, the logistic regression (LogReg) is used to built the predictive model.

```
from sklearn import linear_model
logreg = linear_model.LogisticRegression(C=1e5);
```

With the help of the sklearn library, calibrate a logistic regression model (chose the value of K) on the training data set, What are the available parameters and attributes of the ‘LogisticRegression() ? Display, study and comment the parameters of the statistical/ machine learning logistic regression model? Compute the accuracy, precision, recall and f1-score on the training data set. Visualize the decision boundaries of the calibrated model.

KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5);
```

With the help of the sklearn library, calibrate a KNN model (chose the value of K) on the training data set, What are the available parameters and attributes of the `KNeighborsClassifier()` ? Display, study and comment the parameters of the statistical/ machine learning KNN regression model? Compute the accuracy, precision, recall and f1-score on the training data set. Visualize the decision boundaries of the calibrated model.

What is the distance used by default ? Are the inputs automatically scaled ? What is the impact of scaling or not scaling the inputs on the classification? Test different values of K. How can you chose the best value of K ?

How to choose an appropriate classifier ?

A frequently asked question is to select the best classification model. With no theoretical assumption on the data, it is necessary and useful to evaluate the ability of a set of classification machines and to select the “best”” one. To answer to this question, one should choose the evaluation criteria.

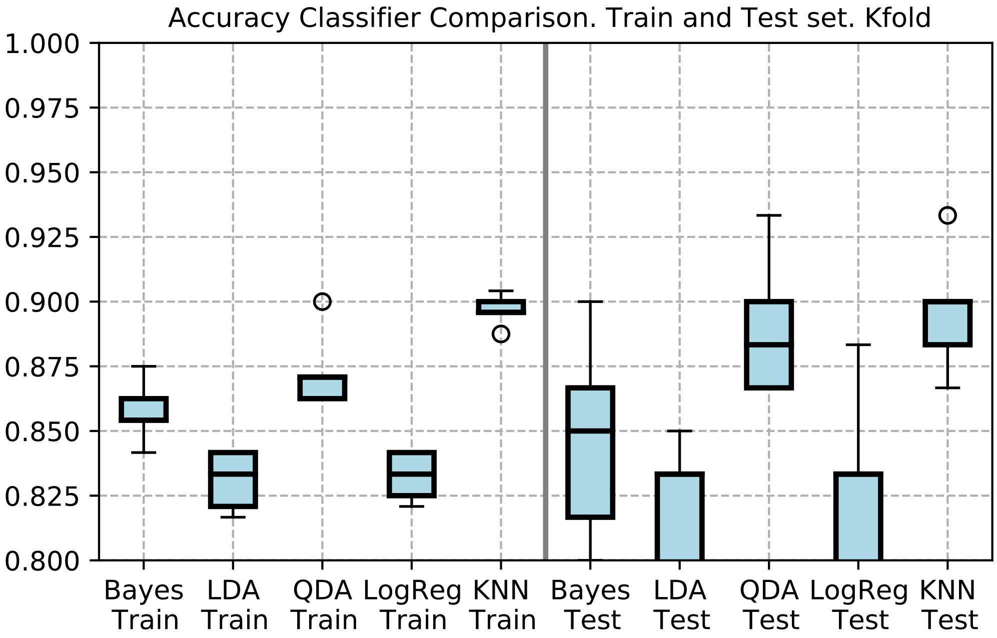

Figure 6: Accuracy study between classifiers

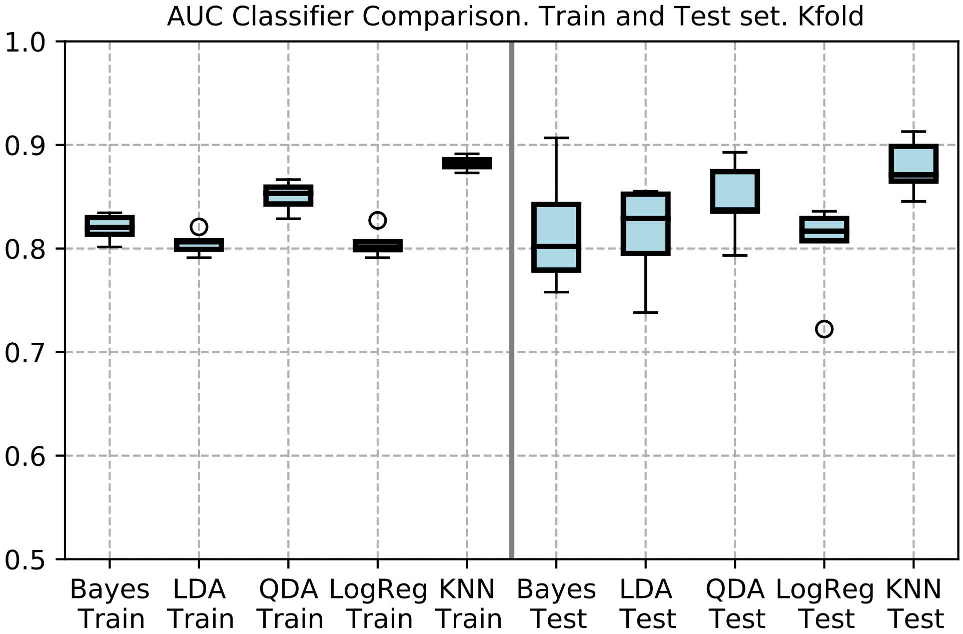

Figure 7: AUC study between classifiers

Next step:

on your own . . .

Use python to generate your own 2D data as in Figure 8 and evaluate the previous studied classifiers on your data set.

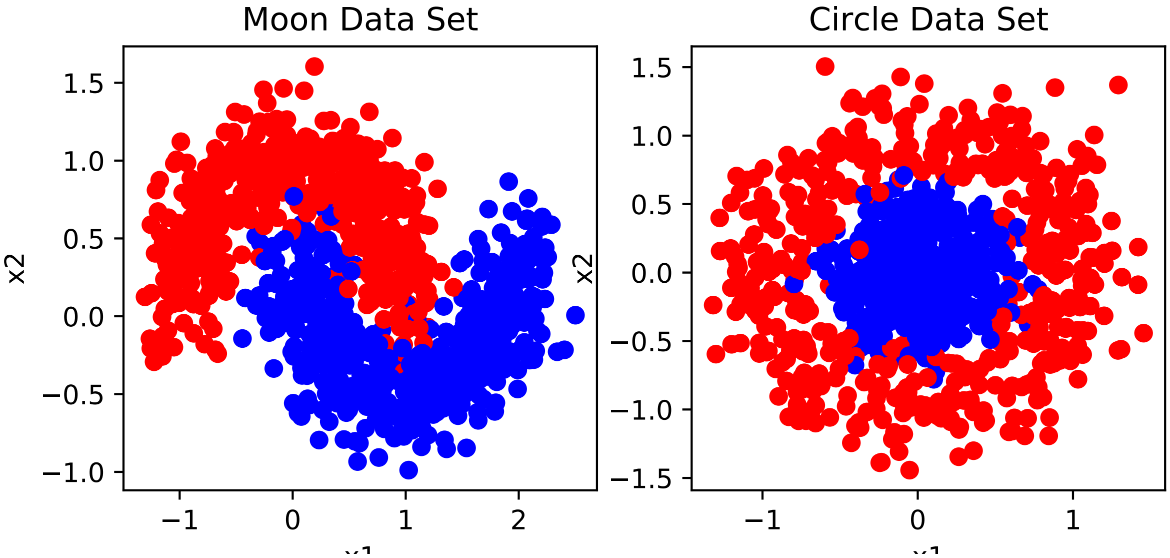

Figure 8: New data sets

Cassification models for a real application: the Heart dataset.

The “SAheart.txt” dataset stored a $n = 462$ sample of males in a heart-disease high-risk region of the Western Cape, South Africa (see “SAheartinfo.txt” file for more information). The aim of the work of this section is to study classification models to be able to predict the value of the “chd” response variable (coronary heart disease) givne the other variables $(p = 10)$.

Load and extract the quantitative co-variables (X) and the target response (Y) of this dataset.

```
#Application SA Heart
####################################
import pandas as pd
import numpy as np
tab = pd.read_csv('SAheart.txt')
#print(tab)
np.shape(tab)
```

```
## (462, 11)
```

```
Y=tab["chd"]
Xnum=tab.loc[:,['sbp','tobacco','ldl','adiposity','typea','obesity','alcohol','age']]
X=Xnum.to_numpy();
```

Take home messages

Write your own take home messages after this practical session.

Annex

Gaussian and Grid data simulation

```
import numpy as np
import matplotlib.pyplot as plt

#2D Gaussian observations
NG=500;
muG=[2,3];
covG=[[ 1,0.9], [0.9,2]];
XG = np.random.multivariate_normal(muG,covG,NG);

#New data: data on a grid
Neval=15;
x1=XG[:,0]; x2=XG[:,1];
x1_min, x1_max = min(x1), max(x1)
x2_min, x2_max = min(x2), max(x2)
h1= (x1_max-x1_min)/Neval; h2=(x2_max-x2_min)/Neval
x1Grid, x2Grid= np.meshgrid(np.arange(x1_min, x1_max, h1), np.arange(x2_min, x2_max, h2));
```

Graph examples

```
#Graphs
figure = plt.figure(figsize=(6,3));
ax = plt.subplot(1,2, 1);
ax.scatter(x1,x2,c="blue")
ax.set_title('I. Training Data Set (TDS)')
ax.set_xlabel('x1'); ax.set_ylabel('x2')
plt.grid()
#plt.colorbar()

ax = plt.subplot(1,2,2)
ax.scatter(x1Grid,x2Grid,c='gray',marker='x')
ax.set_title('II. New Grid inputs')
ax.set_xlabel('x1')
ax.set_ylabel('x2')
```

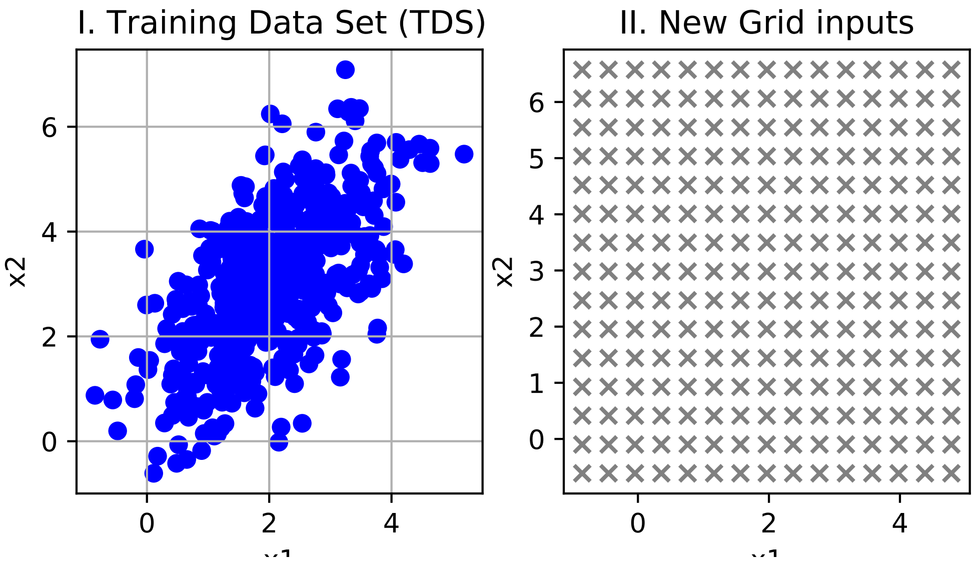

Figure 9: Data plot

Using the usual indicators (Accuracy, Precision, recall and F1-score) and the ROC curve, compare the classification decision tree, the bagging and the random forest models.