# Machine learning for classification (tree, bag, RF)

*M. Mougeot*

*ENSIIE, October 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 for the firt exercices

- The firt part of this document introduces how to program Binary classification Machine learning models using the Python language. In order to understand more deeply the properties of the models from a modelling point of view given the values of the hyper parameters of each method, the firt exercices use 2D simulated data, so that the boundaries of the classification area may be visualized.

## Instructions for the last exercice: the TP project.

- In the last exercice, the goal is to calibrate a machine learning decision model able to diagnose a binary medical output thanks to several covariables.
- This last practical work must be carried on with a 'group of two students'.
- This MAL project aims to developp a Jupyter notebook to solve a classification problem . 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 project.

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 math
```

# The data

## A labeled Data Set

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).

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.

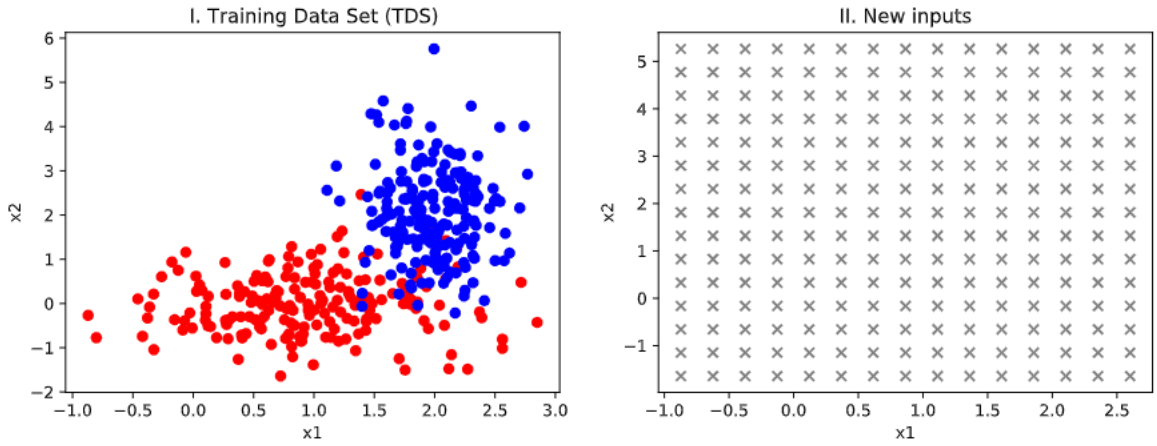


Figure 1: Data sets

# Classification Decision Tree

The *DecisionTreeClassifier*() of the library 'tree' implements the decision tree for classification.

## Model calibration

Following instructions calibrate a model on the training data set given inputs X and output Y.

```
# Decision Tree
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

tree = tree.DecisionTreeClassifier()
treefit = tree.fit(X, Y);
pY_train=treefit.predict_proba(X);

#Score and decision  on the training set
predxclass=np.argmax(pY_train,axis=1); #print(predclass)

#Accuracy
E_train=(G != predxclass).sum()/len(G)
#print("Error on the complete training set %5.2f->",E_train)
```

## Model description

The tree model can be displayed thanks to the following instructions (see annex for printed tree)

```
from sklearn.tree import export_text
r = export_text(treefit); #print(r)
```

## Score and decision boundaries

Compute the class prediction for all the inputs of the grid data set and visualize the decision boundaries.

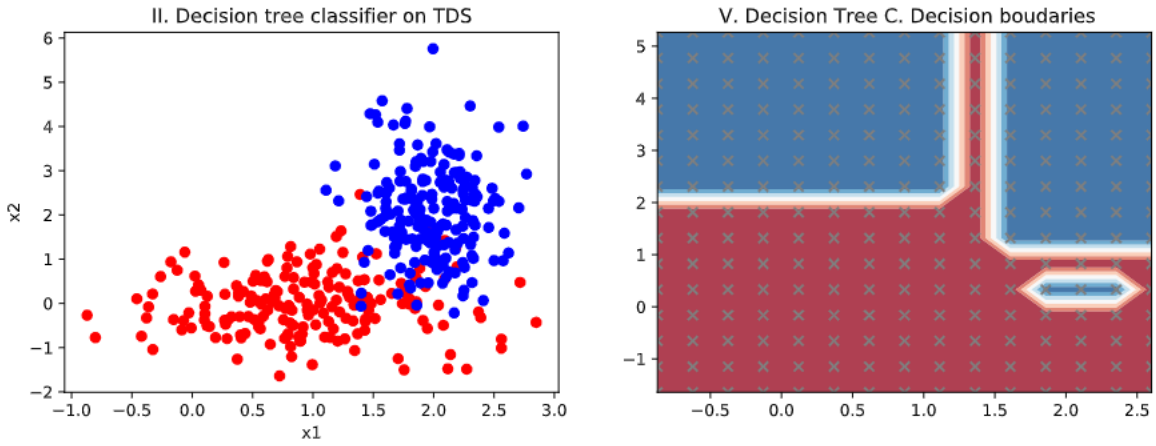


Figure 2: Data sets

## Meta parameters: split, leaf and deviance

What are the meta parameters of the decision tree ? What are the default values of the meta parameters of the $DecisionTreeClassifier()$ function ?

Modify the Python instructions in order to set the minimum sample split parameter to 20 and the minimum sample leaf to 10. Update the model taking into account the new parameter values and visualize the new decision boundaries as for Figure 3.

Modify the Python instructions in order to modify the tuning of the deviance parameter.

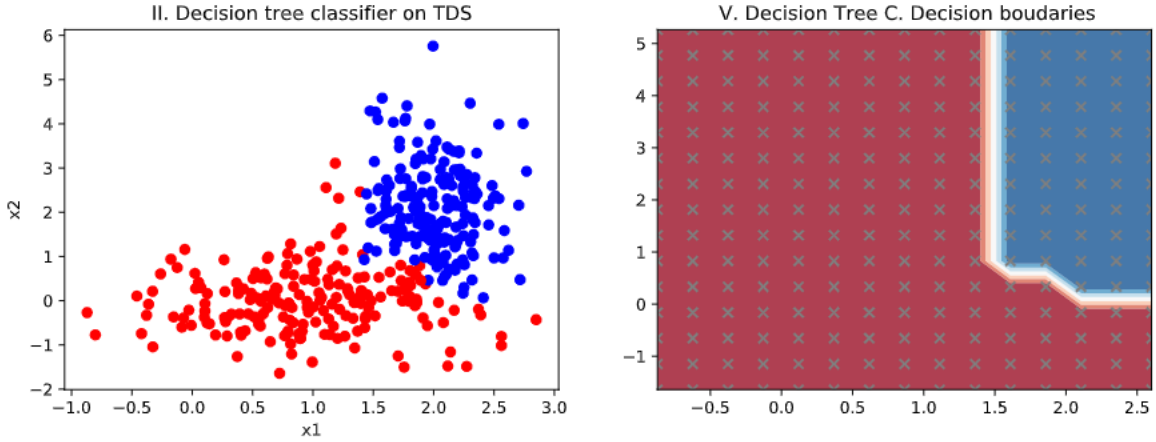


Figure 3: Data sets

How do you suggest to tune the meta parameters ?

# Bagging

## Model calibration

Following instructions calibrate a bagging model based on classification trees on the training data set given inputs X and output Y.

```
#Bagging
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text
from sklearn.ensemble import BaggingClassifier

treemod = tree.DecisionTreeClassifier()
bagmod=BaggingClassifier(base_estimator=treemod, n_estimators=10, random_state=0)
treemodfit=treemod.fit(X, y);
bagmodfit=bagmod.fit(X, y);

pY_train=bagmodfit.predict_proba(X);

#Score and decision  on the training set
predxclass=np.argmax(pY_train,axis=1); #print(predclass)
```

## Score and decision boundaries

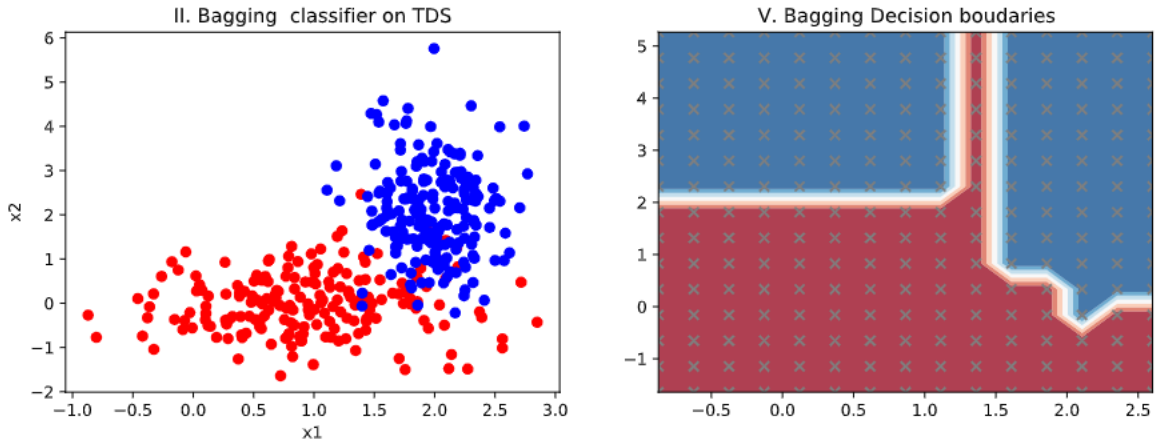


Figure 4: Data sets

## Meta parameters

What are the meta parameters of the bagging function applied to decision tree classifiers? Modify the default values of the Bagging parameters and check the differences on the decision boundaries graph.

# Random Forest

## Model calibration

Following instructions calibrate a bagging model based on classification trees on the training data set given inputs X and output Y.

```
#Random forest
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text

#tree = tree.DecisionTreeClassifier()
RF = RandomForestClassifier(max_depth=2, random_state=0)
RFfit = RF.fit(X, Y);
pY_train=RFfit.predict_proba(X);

#Score and decision computation on the training set
predxclass=np.argmax(pY_train,axis=1); #print(predclass)
```

## Score and decision boundaries

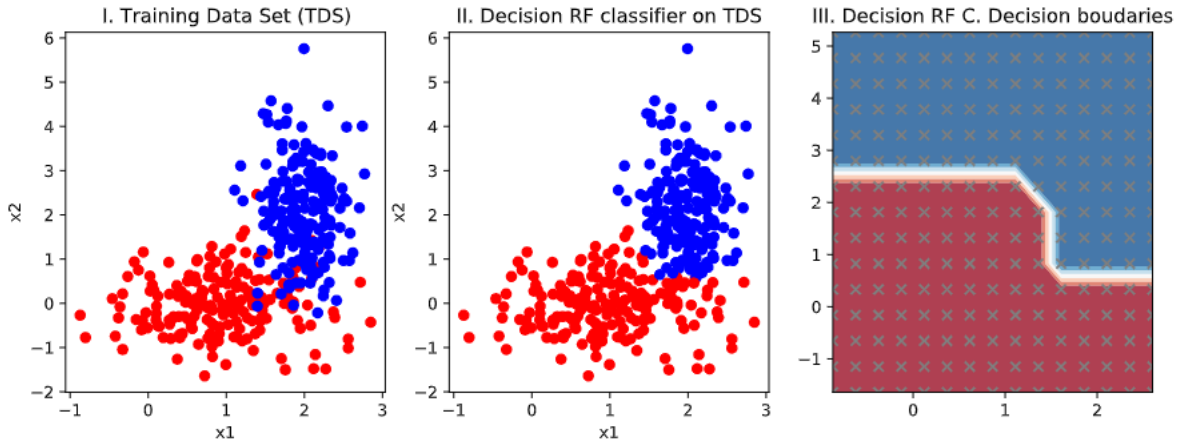


Figure 5: Data sets

## Meta parameters

What are the meta parameters of the random forest function applied to decision tree classifiers? Modify the default values of the Bagging parameters and check the differences on the decision boundaries graph.

# Random Forest

## Model calibration

Following instructions calibrate a random forest model based on classification trees on the training data set given inputs X and output Y.

```
#Random forest
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text

#tree = tree.DecisionTreeClassifier()
RF = RandomForestClassifier(max_depth=2, random_state=0, oob_score = True)
RFfit = RF.fit(X, Y);
```

## Model criteria

Several criteria are commonly used to evaluate the RF model as the global score (accuracy), theOOB (Out Of Bag) and the importance variables:

```
score=RF.score;
OOB=RF.oob_score_
IF=RF.feature_importances_
```

## Score and decision boundaries

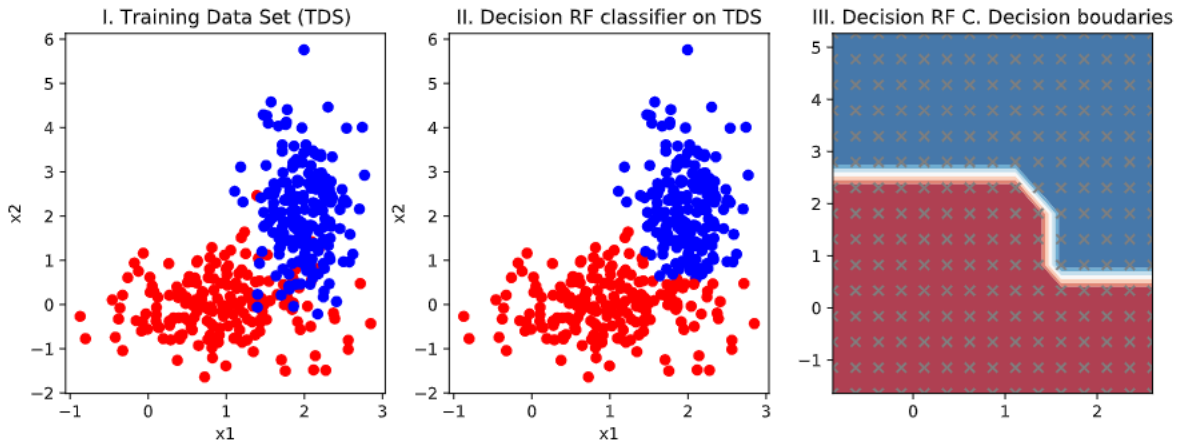


Figure 6: Random Forest

# Extra Trees

## Model calibration

Following instructions calibrate a random forest model based on classification trees on the training data set given inputs X and output Y.

```
#Random forest
from sklearn.ensemble import ExtraTreesClassifier

#tree = tree.DecisionTreeClassifier()
ExTC = ExtraTreesClassifier(max_depth=2, random_state=0)
ExTCfit = ExTC.fit(X, Y);
pY_train=ExTCfit.predict_proba(X);
```

## Model criteria

Several criteria are commonly used to evaluate the ExT model as the global score (accuracy), the OOB (Out Of Bag) score (in the case of bootstap samples are generated) and the importance variables:

```
from sklearn.ensemble import ExtraTreesClassifier
ExTC = ExtraTreesClassifier(max_depth=2, random_state=0,bootstrap=True,oob_score=True)
ExTCfit = ExTC.fit(X, Y);
pY_train=ExTCfit.predict_proba(X);
score=ExTC.score;

OOB=ExTC.oob_score_
IF=ExTC.feature_importances_
```

## Score and decision boundaries

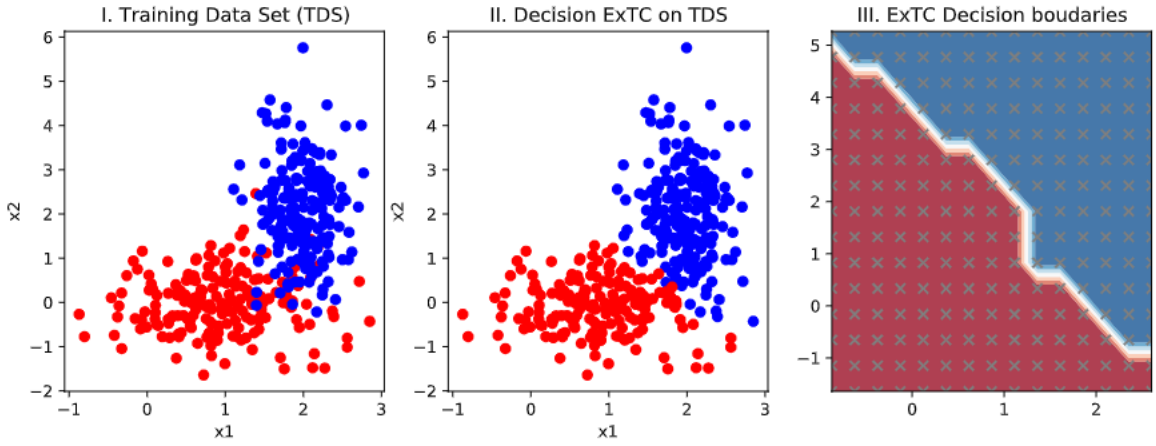


Figure 7: Extra Tree

# EXERCICE TO DELIVER. Classification 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 (all the models you have been studied in the MAL course up to now if appropriate) to be able to predict the value of the "chd" response variable (coronary heart disease) given 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();
```

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.

# Annexes

## Classification decision tree description

```
## |--- feature_1 <= 0.81
## |   |--- feature_0 <= 1.60
## |   |   |--- feature_0 <= 1.39
## |   |   |   |--- class: 0
## |   |   |--- feature_0 >  1.39
## |   |   |   |--- feature_0 <= 1.41
## |   |   |   |   |--- class: 1
## |   |   |   |--- feature_0 >  1.41
## |   |   |   |   |--- class: 0
## |   |--- feature_0 >  1.60
## |   |   |--- feature_1 <= 0.05
## |   |   |   |--- feature_1 <= -0.27
## |   |   |   |   |--- class: 0
## |   |   |   |--- feature_1 >  -0.27
## |   |   |   |   |--- feature_1 <= -0.21
## |   |   |   |   |   |--- class: 1
## |   |   |   |   |--- feature_1 >  -0.21
## |   |   |   |   |   |--- feature_0 <= 1.86
## |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |   |--- feature_0 >  1.86
## |   |   |   |   |   |   |--- feature_0 <= 1.95
## |   |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |   |--- feature_0 >  1.95
## |   |   |   |   |   |   |   |--- class: 0
## |   |   |--- feature_1 >  0.05
## |   |   |   |--- feature_0 <= 1.94
## |   |   |   |   |--- feature_1 <= 0.64
## |   |   |   |   |   |--- feature_1 <= 0.52
## |   |   |   |   |   |   |--- feature_0 <= 1.79
## |   |   |   |   |   |   |   |--- feature_1 <= 0.21
## |   |   |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |   |   |--- feature_1 >  0.21
## |   |   |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |   |   |--- feature_0 >  1.79
## |   |   |   |   |   |   |   |--- feature_1 <= 0.40
## |   |   |   |   |   |   |   |   |--- feature_0 <= 1.80
## |   |   |   |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |   |   |   |   |--- feature_0 >  1.80
## |   |   |   |   |   |   |   |   |   |--- feature_0 <= 1.87
## |   |   |   |   |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |   |   |   |   |--- feature_0 >  1.87
## |   |   |   |   |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |   |   |   |--- feature_1 >  0.40
## |   |   |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |--- feature_1 >  0.52
## |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |--- feature_1 >  0.64
## |   |   |   |   |   |--- feature_0 <= 1.84
## |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |--- feature_0 >  1.84
## |   |   |   |   |   |   |--- class: 0
## |   |   |   |--- feature_0 >  1.94
## |   |   |   |   |--- feature_0 <= 2.56
## |   |   |   |   |   |--- feature_0 <= 2.10
## |   |   |   |   |   |   |--- feature_0 <= 2.08
```

```
## |   |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |   |--- feature_0 >  2.08
## |   |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |   |--- feature_0 >  2.10
## |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |--- feature_0 >  2.56
## |   |   |   |   |   |--- class: 0
## |--- feature_1 >  0.81
## |   |--- feature_0 <= 1.42
## |   |   |--- feature_1 <= 1.98
## |   |   |   |--- class: 0
## |   |   |--- feature_1 >  1.98
## |   |   |   |--- feature_0 <= 1.30
## |   |   |   |   |--- class: 1
## |   |   |   |--- feature_0 >  1.30
## |   |   |   |   |--- class: 0
## |   |--- feature_0 >  1.42
## |   |   |--- feature_1 <= 1.42
## |   |   |   |--- feature_1 <= 1.40
## |   |   |   |   |--- feature_1 <= 0.84
## |   |   |   |   |   |--- feature_1 <= 0.82
## |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |--- feature_1 >  0.82
## |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |--- feature_1 >  0.84
## |   |   |   |   |   |--- feature_0 <= 1.60
## |   |   |   |   |   |   |--- feature_0 <= 1.49
## |   |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |   |--- feature_0 >  1.49
## |   |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |   |--- feature_0 >  1.60
## |   |   |   |   |   |   |--- feature_0 <= 2.04
## |   |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |   |--- feature_0 >  2.04
## |   |   |   |   |   |   |   |--- feature_0 <= 2.06
## |   |   |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |   |   |   |--- feature_0 >  2.06
## |   |   |   |   |   |   |   |   |--- feature_1 <= 1.24
## |   |   |   |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |   |   |   |--- feature_1 >  1.24
## |   |   |   |   |   |   |   |   |   |--- feature_1 <= 1.29
## |   |   |   |   |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |   |   |   |   |   |--- feature_1 >  1.29
## |   |   |   |   |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |--- feature_1 >  1.40
## |   |   |   |   |   |--- class: 0
## |   |   |--- feature_1 >  1.42
## |   |   |   |--- feature_1 <= 1.91
## |   |   |   |   |--- feature_1 <= 1.90
## |   |   |   |   |   |--- feature_1 <= 1.63
## |   |   |   |   |   |   |--- feature_1 <= 1.63
## |   |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |   |   |--- feature_1 >  1.63
## |   |   |   |   |   |   |   |--- class: 0
## |   |   |   |   |   |--- feature_1 >  1.63
## |   |   |   |   |   |   |--- class: 1
## |   |   |   |   |--- feature_1 >  1.90
## |   |   |   |   |   |--- class: 0
## |   |   |   |--- feature_1 >  1.91
```

```
## |   |   |   |   |--- class: 1
```

Classification decision tree visualisation