

Machine learning for classification (tree, bag, RF)

M. Mougeot

ENSIIE, September 2020

Contents

Introduction	2
Goal of the practical sessions	2
Warnings and Advices	2
Instructions	2
The data	3
A labeled Data Set	3
Classification Decision Tree	3
Model calibration	3
Model description	3
Score and decision boundaries	4
Meta parameters: split, leaf and deviance	4
Bagging	5
Model calibration	5
Score and decision boundaries	5
Meta parameters	5
Random Forest	6
Model calibration	6
Score and decision boundaries	6
Meta parameters	6
Cassification models for a real application: the Heart dataset.	7
Annexes	8
Classification decision tree description	8
Classification decision tree visualisation	9

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 interpret ‘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 develop appropriate models for answering to a given operational 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 develop 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 project.

For this practical session, the following libraries need to 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.

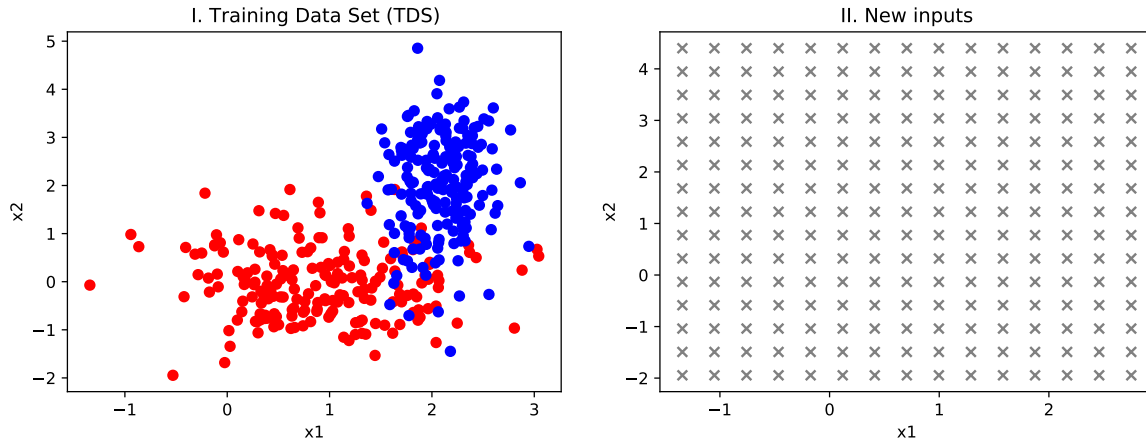


Figure 1: Data sets

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.

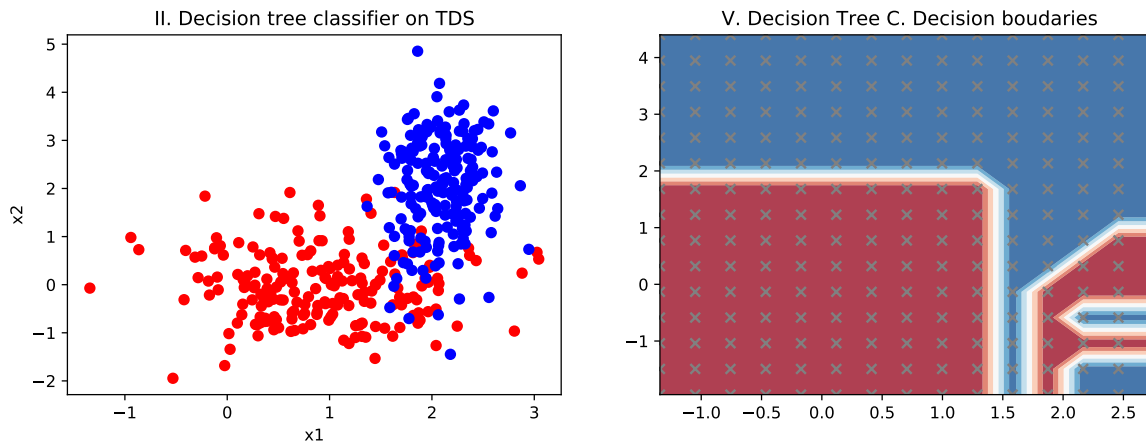


Figure 2: Data sets

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.

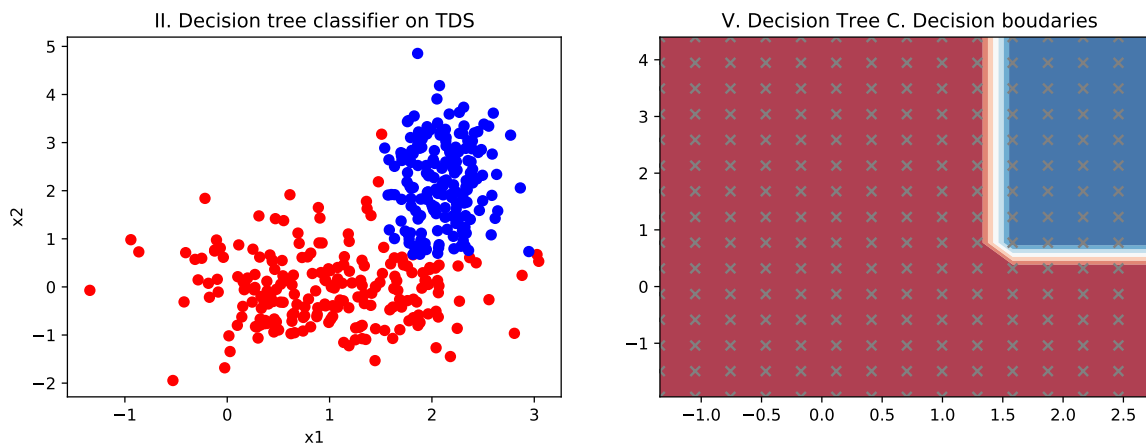


Figure 3: Data sets

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(predxclass)
```

Score and decision boundaries

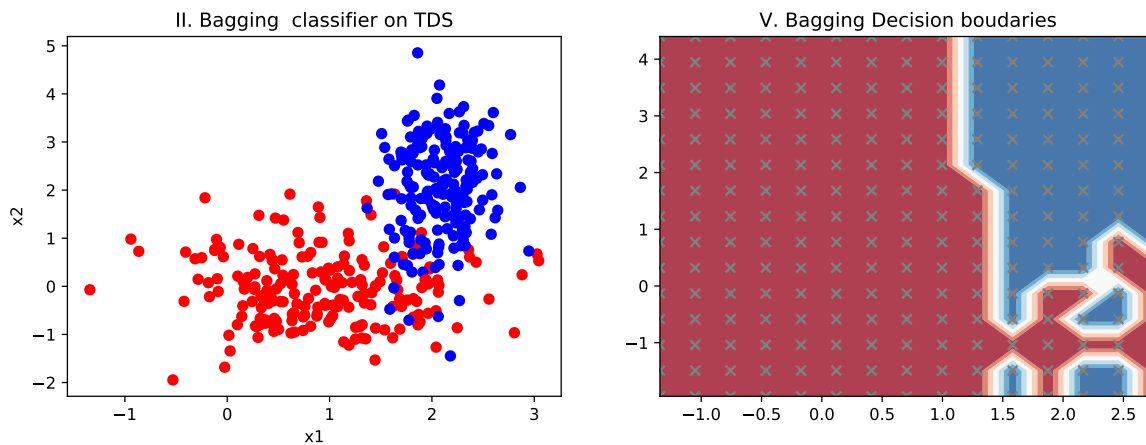


Figure 4: Data sets

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(predxclass)
```

Score and decision boundaries

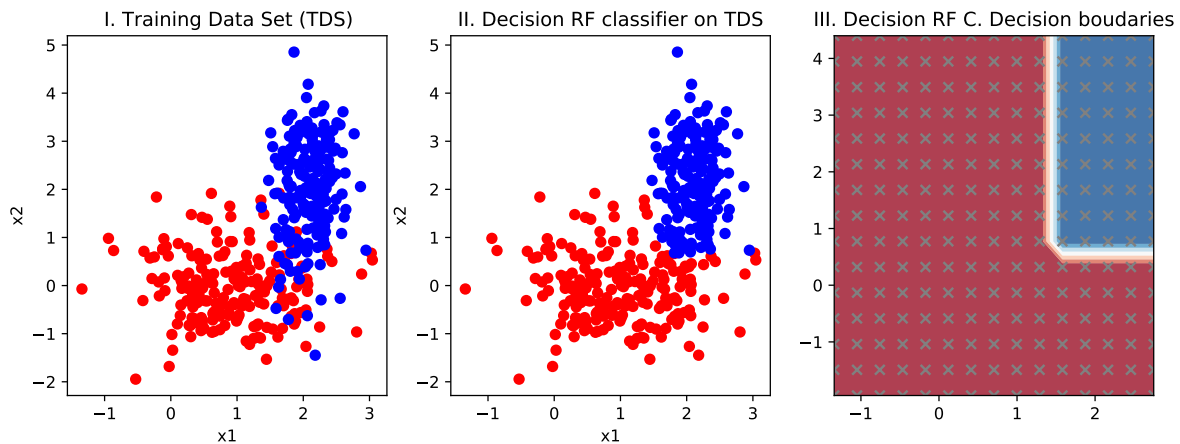


Figure 5: Data sets

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.

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();
```

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_0 <= 1.54
## | |--- feature_1 <= 2.05
## | | |--- feature_1 <= 1.56
## | | | |--- class: 0
## | | |--- feature_1 > 1.56
## | | | |--- feature_1 <= 1.64
## | | | | |--- class: 1
## | | | |--- feature_1 > 1.64
## | | | | |--- class: 0
## | |--- feature_1 > 2.05
## | | |--- class: 1
## |--- feature_0 > 1.54
## | |--- feature_1 <= 0.68
## | | |--- feature_0 <= 1.59
## | | | |--- class: 1
## | | |--- feature_0 > 1.59
## | | | |--- feature_1 <= 0.13
## | | | | |--- feature_0 <= 2.05
## | | | | | |--- feature_0 <= 1.80
## | | | | | | |--- feature_0 <= 1.77
## | | | | | | |--- feature_1 <= -0.05
## | | | | | | | |--- class: 0
## | | | | | | |--- feature_1 > -0.05
## | | | | | | | |--- class: 1
## | | | | | | |--- feature_0 > 1.77
## | | | | | | | |--- class: 1
## | | | | | | |--- feature_0 > 1.80
## | | | | | | | |--- class: 0
## | | | | |--- feature_0 > 2.05
## | | | | | |--- feature_1 <= -0.19
## | | | | | | |--- feature_1 <= -0.74
## | | | | | | |--- feature_1 <= -1.21
## | | | | | | | |--- class: 1
## | | | | | | |--- feature_1 > -1.21
## | | | | | | | |--- class: 0
## | | | | | | |--- feature_1 > -0.74
## | | | | | | | |--- class: 1
## | | | | | | |--- feature_1 > -0.19
## | | | | | | | |--- class: 0
## | | | | |--- feature_1 > 0.13
## | | | | | |--- feature_1 <= 0.47
## | | | | | | |--- feature_1 <= 0.42
## | | | | | | |--- feature_0 <= 1.96
## | | | | | | | |--- feature_0 <= 1.65
## | | | | | | | | |--- class: 0
## | | | | | | | |--- feature_0 > 1.65
## | | | | | | | | |--- feature_0 <= 1.92
## | | | | | | | | | |--- class: 1
## | | | | | | | | |--- feature_0 > 1.92
## | | | | | | | | | |--- feature_1 <= 0.13
## | | | | | | | | | | |--- class: 0
## | | | | | | | | | |--- feature_1 > 0.13
## | | | | | | | | | | |--- class: 1
## | | | | | | |--- feature_0 > 1.96
```


[illegible]

Classification decision tree visualisation