Machine learning for classification (Ensemble methods)

M. Mougeot

ENSIIE, September 2020

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 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.

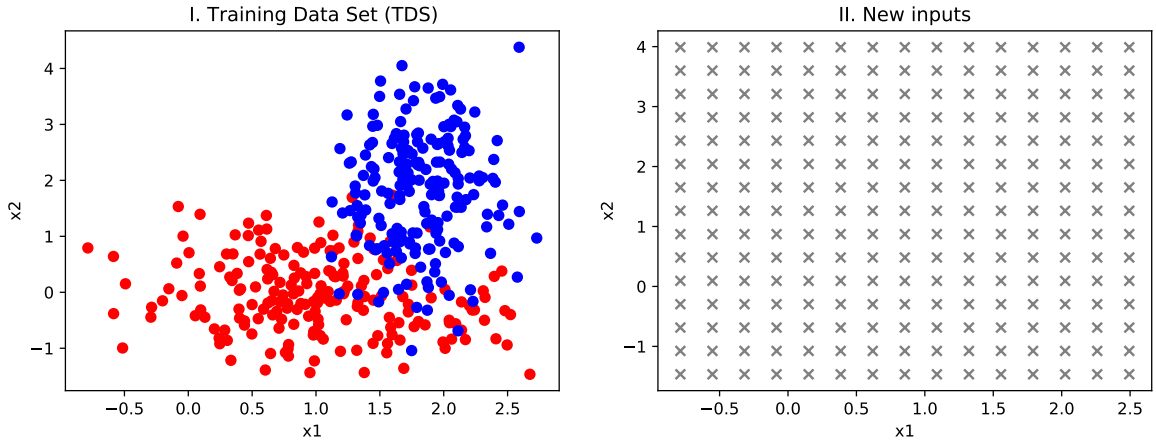

Figure 1: Random Forest

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

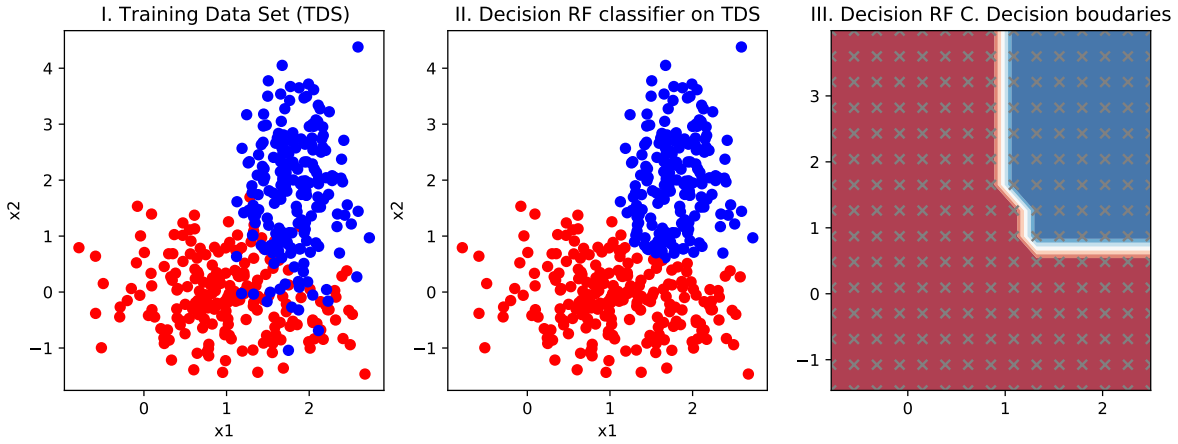

Figure 2: Random Forest

Adaboost

Model calibration

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

```
##Boosting
from sklearn.ensemble import AdaBoostClassifier
from sklearn import metrics

Ab = AdaBoostClassifier(n_estimators=100, random_state=0)
Abfit=Ab.fit(X, y)
y_pred=Abfit.predict(X);

E_all=(y != y_pred).sum()/len(y)
print("Boost Error on the complete training set %5.2f->",E_all)
```

```
## Boost Error on the complete training set %5.2f-> 0.0175
```

Score and decision boundaries

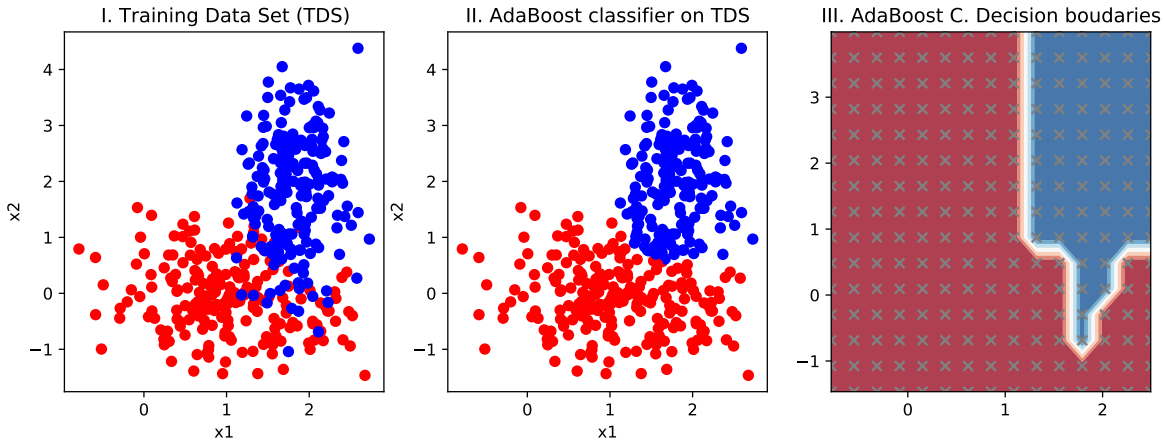

Figure 3: Ada boost

Structure of the successive Adaboost trees

```
import matplotlib.pyplot as plt

Ad = AdaBoostClassifier(n_estimators=200, random_state=0)
Ad.fit(X, y)
```

```
## AdaBoostClassifier(n_estimators=200, random_state=0)
```

```
ad_staged=Ad.staged_predict(X)

Ad_seq_errors = []
for  Ad_train_predict in Ad.staged_predict(X):
    Ad_seq_errors.append(metrics.accuracy_score(Ad_train_predict, y))

plt.figure(figsize=(15, 5))
```

```
plt.plot(Ad_seq_errors); plt.title('Adaboost underlaying tree Accuracy')
plt.ylabel('Error'); plt.xlabel('Number of Trees')
```

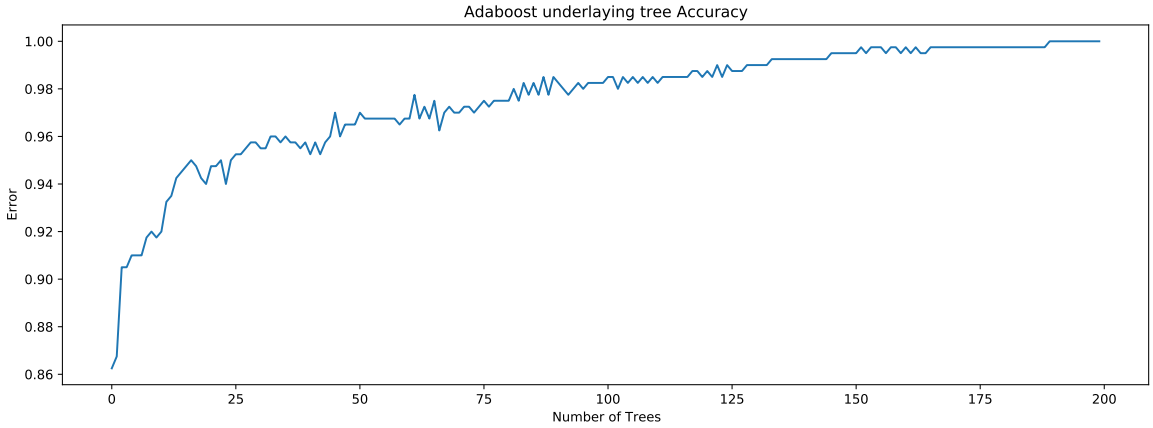

Figure 4: Ada boost successive tree accuracy

Gradient Boosting

Model calibration

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

```
from sklearn.datasets import make_classification
from sklearn.ensemble import GradientBoostingClassifier
GB = GradientBoostingClassifier(random_state=0)
GBfit=GB.fit(X, y)
y_pred=GBfit.predict(X)
E_all=(y != y_pred).sum()/len(y)
print("Gradient Boosting Error on the complete training set %5.2f->",E_all)
```

```
## Gradient Boosting Error on the complete training set %5.2f-> 0.0075
```

Score and decision boundaries

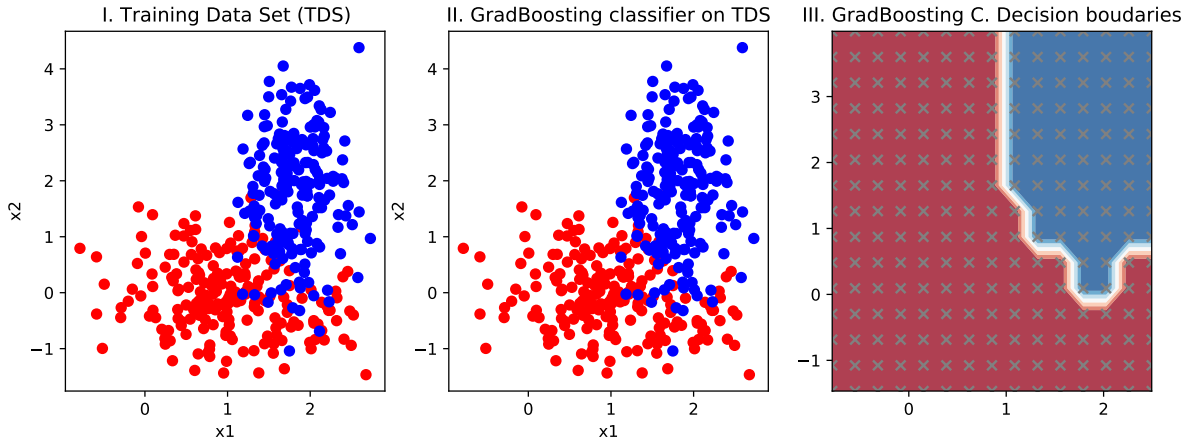

Figure 5: Gradient boosting

Structure of the successive Adaboost trees

```
import matplotlib.pyplot as plt
GB = GradientBoostingClassifier(random_state=0)
GB.fit(X, y)
```

```
## GradientBoostingClassifier(random_state=0)
```

```
GB_staged=GB.staged_predict(X)

GB_seq_errors = []
for  GB_train_predict in GB.staged_predict(X):
    GB_seq_errors.append(metrics.accuracy_score(GB_train_predict, y))

plt.figure(figsize=(15, 5))
plt.plot(GB_seq_errors); plt.title('Gradient Boosting underlaying tree Accuracy')
plt.ylabel('Error'); plt.xlabel('Number of Trees')
```

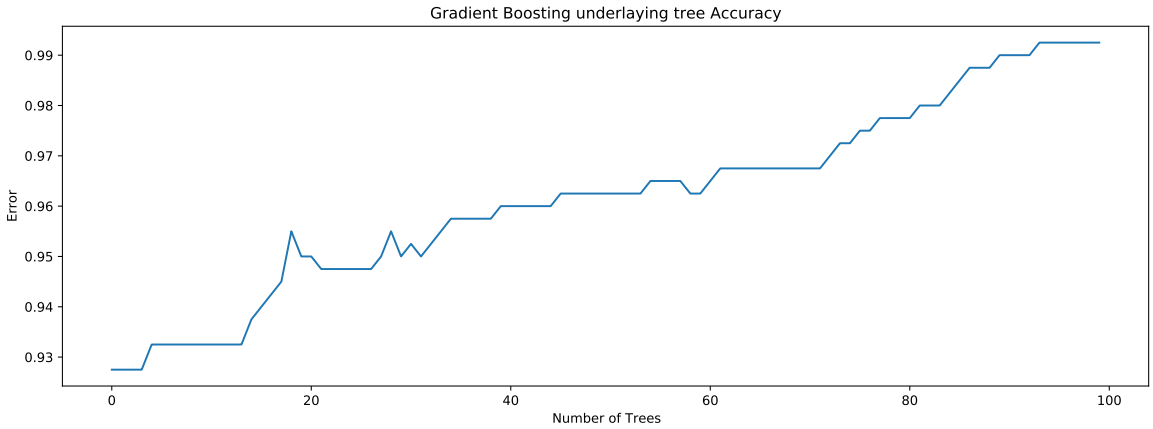

Figure 6: Gradient Booisting successive tree accuracy

Stacking

Implement the stacking aggregation model given the previous classifiers: naive bayes, ADL, QDL, logistic regression, classification trees.

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 to be able to predict the value of the "chd" response variable (coronary heart disease) givne the other variables ($p = 10$).

Compare and study the following classifiers: classification tree, random forest, adaboost, gradient boost and stacking.

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.