renduLab1

May 9, 2022

1 *****ML labs - report *****

Students : Théo Fontana - Vatosoa Razafiniary - 4IR SC Github access : https://github.com/Vatosoa285/TP_ML.git This report gathers the work done in the labs of the machine learning course. It has 3 parts that corresponds to the different labs seen : - lab 1: decision trees - lab 2: Perceptron - lab 3: multi-layer perceptron

The instructions of the labs being quite directive, we took the time to consult the resources provided (about maching learning notions or Python functions) in addition to our courses to carry out and understand the labs. For an easy understanding for the reader, we have put explanatory comments in our codes and displayed the results at runtime.

2 Lab 1 Evaluation

In the following, we consider the (binarized) Compas dataset that we studied in the Lab

[1]:
```
#usefull libraries
from sklearn import tree
from matplotlib import pyplot as plt # for a good visualization of the trees

import csv
import numpy as np
from utils import load_from_csv

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
```

[2]:
```
train_examples, train_labels, features, prediction = load_from_csv("./compass.
 ↪csv")
print ("We can see here that we work with binarized data :\n",train_examples,␣
 ↪"\n-------\n",train_labels)
```

```
We can see here that we work with binarized data :
 [[1 0 0 … 0 0 0]
 [1 0 1 … 0 0 0]
 [1 0 0 … 0 0 0]
 …
```

```
 [1 0 0 … 0 0 0]
 [1 0 0 … 0 0 0]
 [0 1 0 … 0 0 0]]
-------
 [1 0 0 … 1 0 1]
```

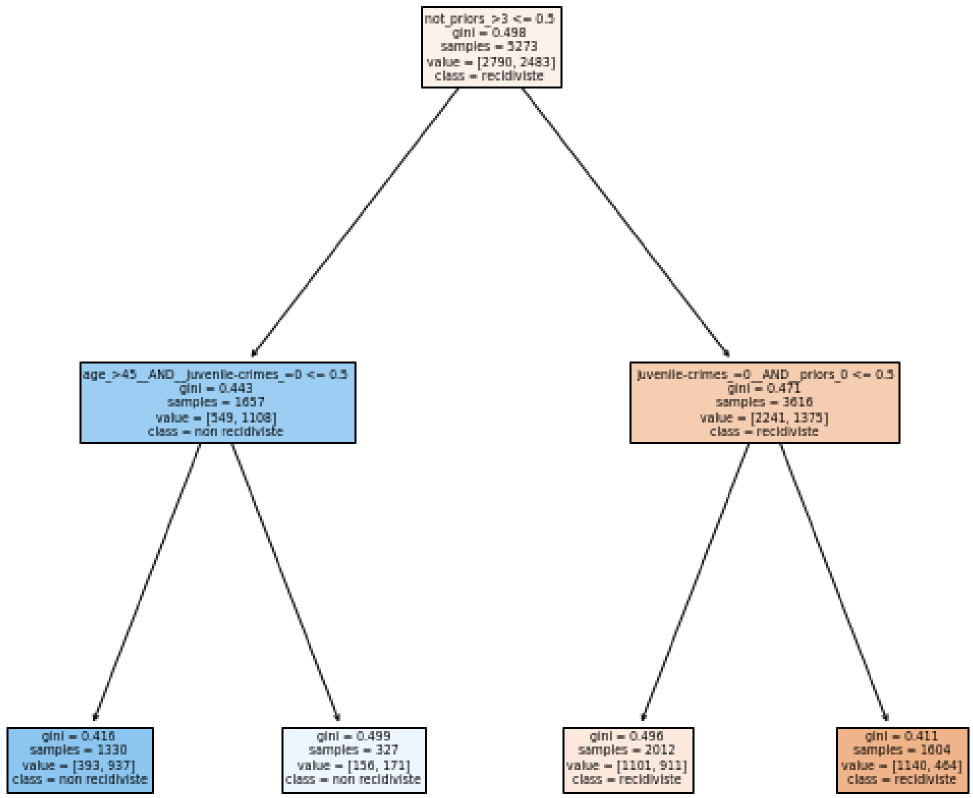
A decision tree configuration is a set of parameters that one can use to build decision trees. Propose 6 configurations that are likely to provide different topologies and caracteristics

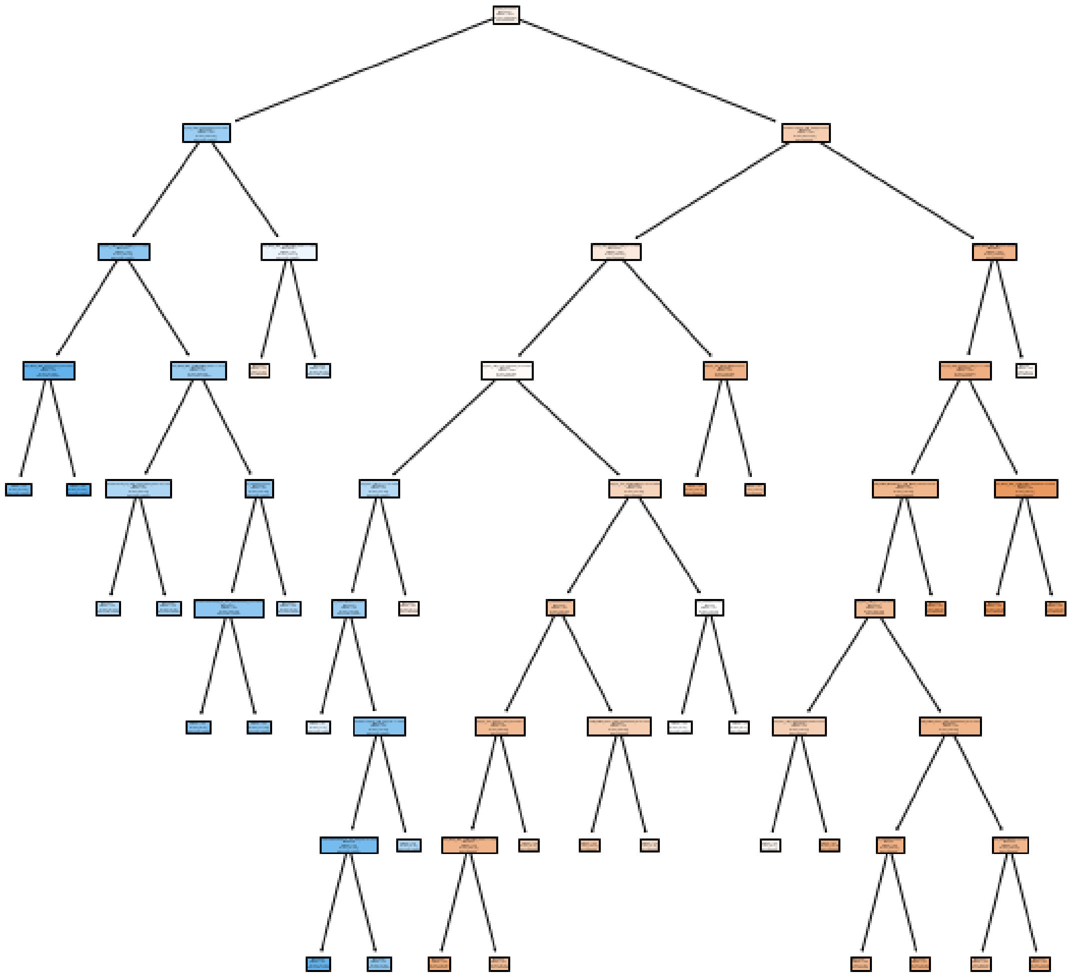
[3]:
```python
#our proposition : affecting different values for the parameters splitter,␣
↪max_depth and min_samples_leaf
treesParmeters = [
    #expected topology and caracteristic
    {'splitter':'best', 'max_depth': 2, 'min_samples_leaf': 1},
    {'splitter':'random', 'max_depth': 10, 'min_samples_leaf': 100},
    {'splitter':'best', 'max_depth': 5, 'min_samples_leaf': 20},
    {'splitter':'best', 'max_depth': 5, 'min_samples_leaf':5},
    {'splitter':'best', 'max_depth': 10, 'min_samples_leaf':10},
    {'splitter':'best', 'max_depth': 100, 'min_samples_leaf':60},
]
nbTree = len(treesParmeters)
print("We have configured :", nbTree, "different decision trees here")
```

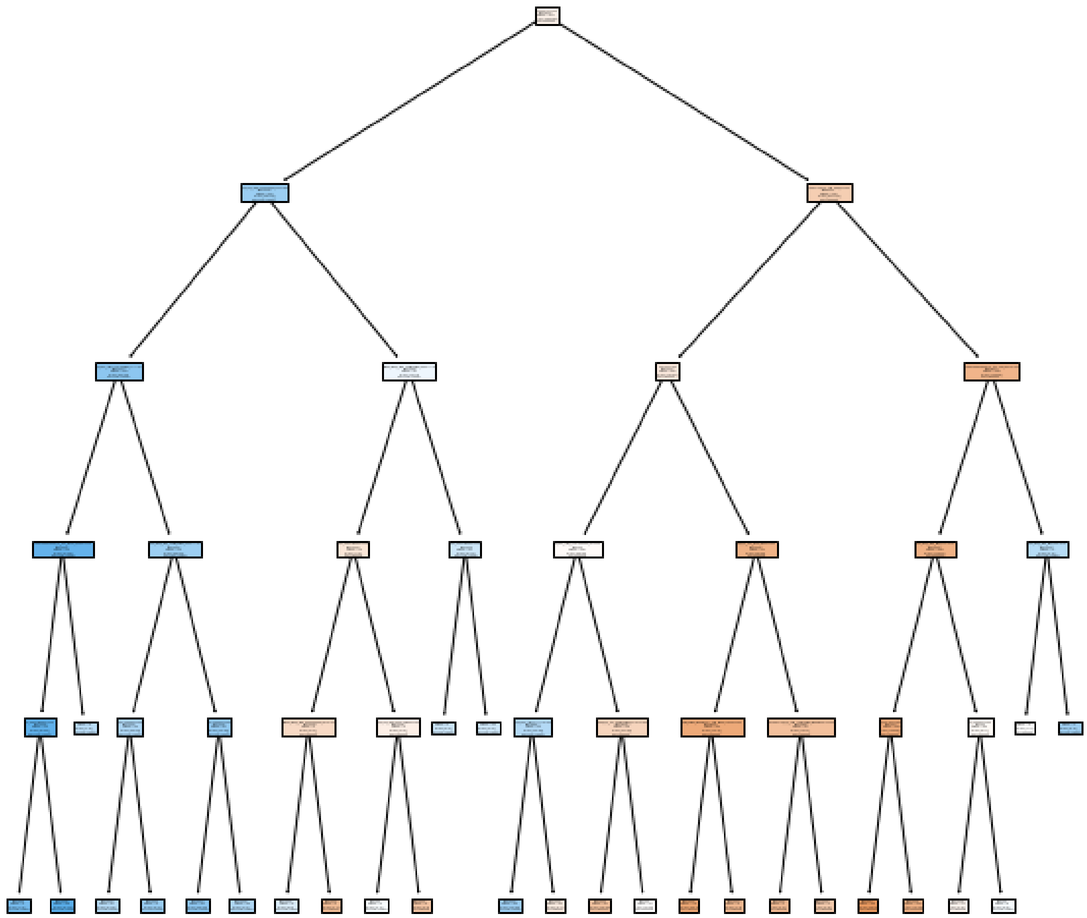
```
We have configured : 6 different decision trees here
```

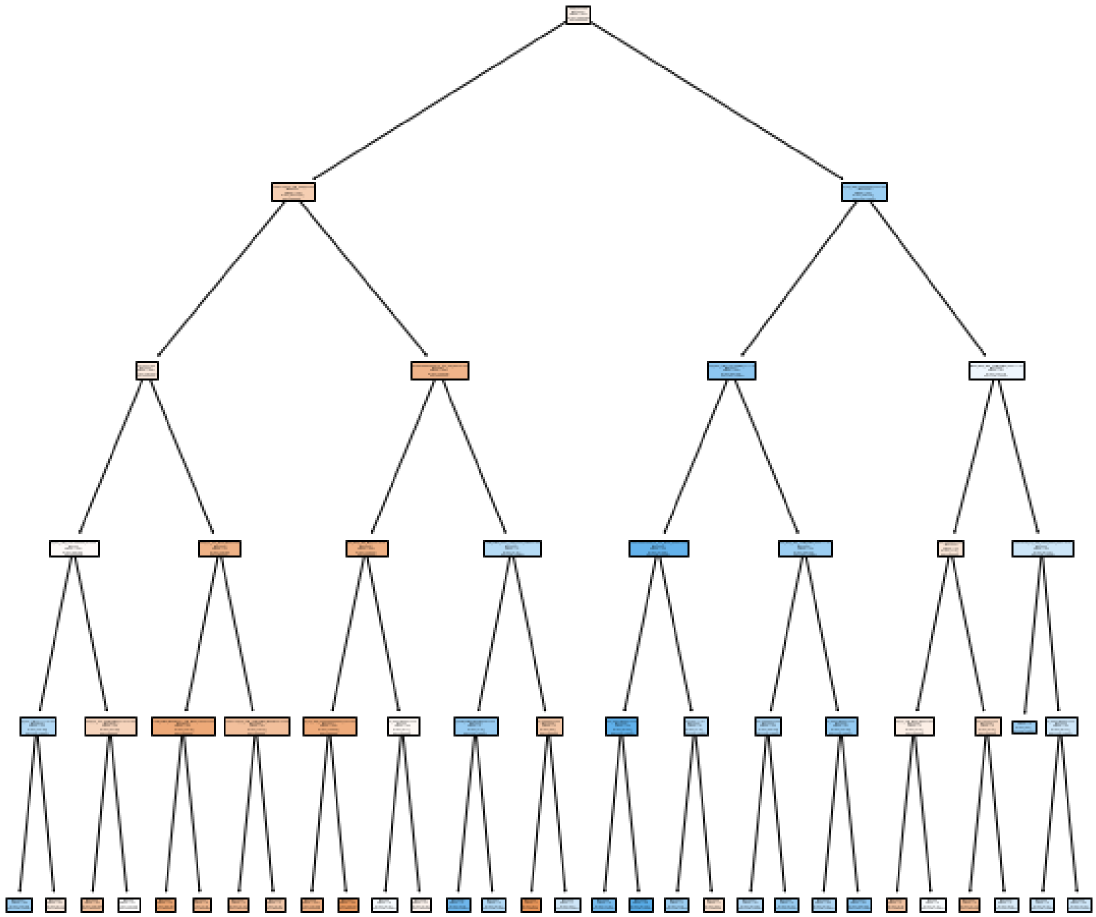
Train a decision tree for each of the previous configurations on the full dataset

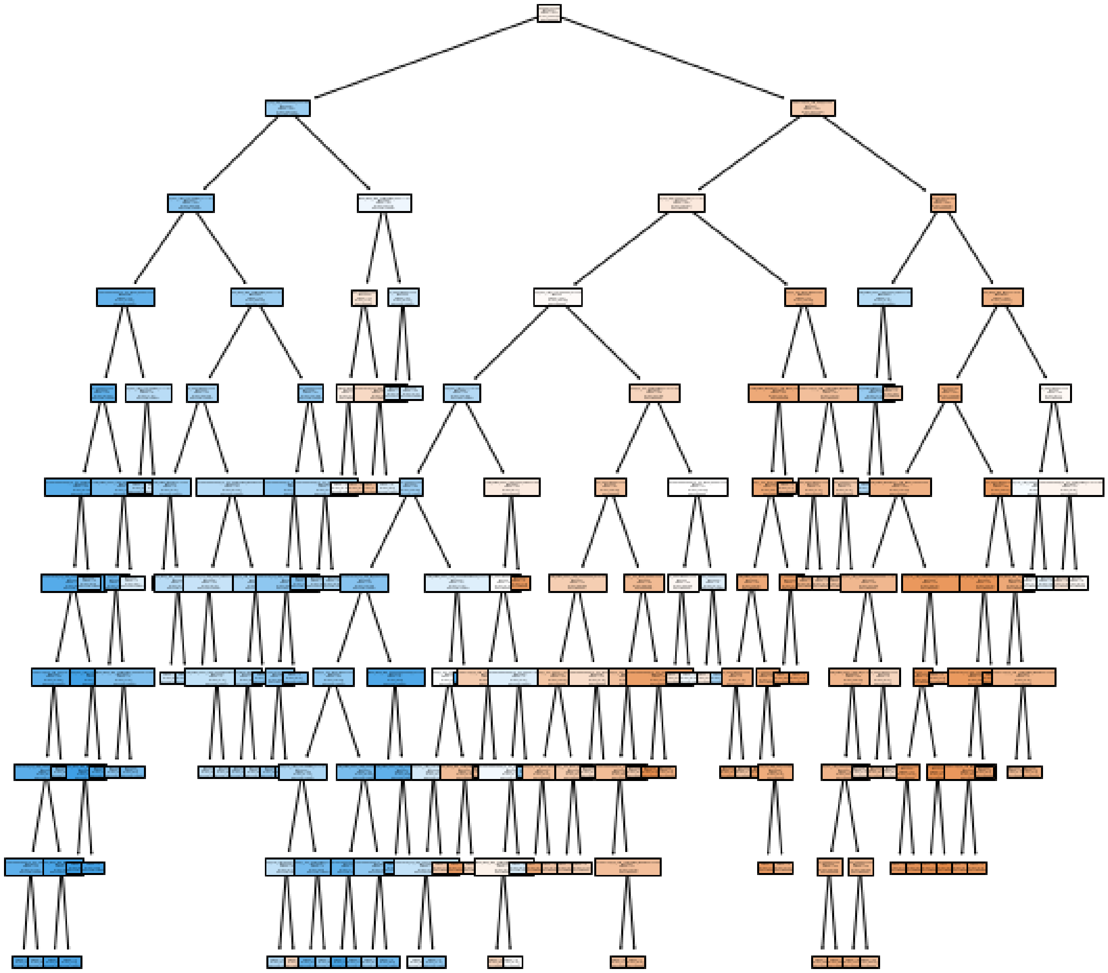
[4]:
```python
for i in range (nbTree) :
    parameters= treesParmeters[i]
    clf = tree.DecisionTreeClassifier( #while creating each tree, we work with␣
↪the parameters configured above
        splitter=parameters['splitter'],
        max_depth =parameters['max_depth'],
        min_samples_leaf =parameters['min_samples_leaf']
    )
    clf = clf.fit(train_examples, train_labels)
    plt.figure(figsize=(10,10))
    tree.plot_tree(clf,
                   feature_names= (features),
                   class_names= ("recidiviste", "non recidiviste"),
                   filled=True)
    plt.show()
    # Result : we can visually check that we obtain 6 different trees
    #Each tree is coherent with the expected topology and characteristic␣
↪parametered
```

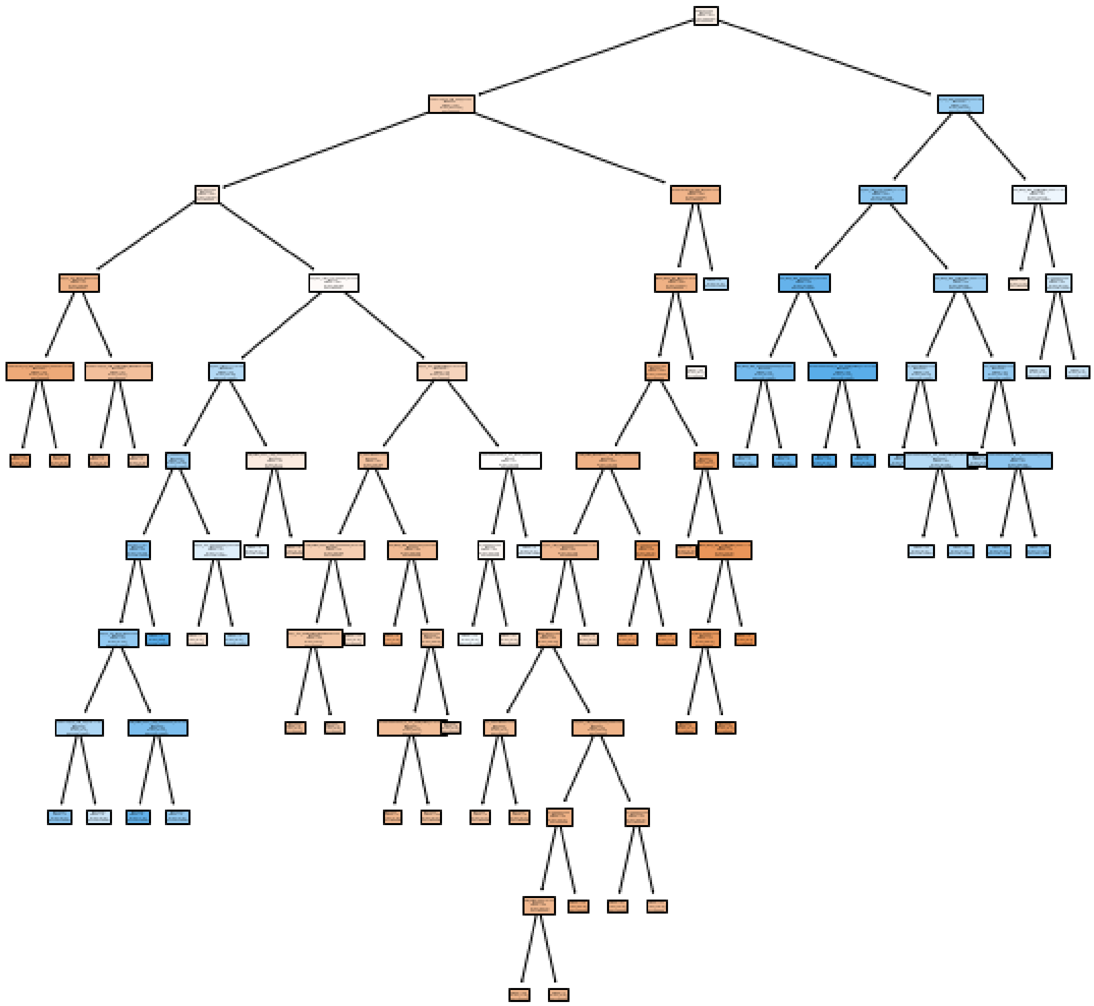

not_priors_>3 <= 0.5
gini = 0.498
samples = 5273
value = [2790, 2483]
class = recidiviste
age_>45__AND__juvenile-crimes_=0 <= 0.5
gini = 0.443
samples = 1657
value = [549, 1108]
class = non recidiviste
juvenile-crimes_=0__AND__priors_0 <= 0.5
gini = 0.471
samples = 3616
value = [2241, 1375]
class = recidiviste
gini = 0.416
samples = 1330
value = [393, 937]
class = non recidiviste
gini = 0.499
samples = 327
value = [156, 171]
class = non recidiviste
gini = 0.496
samples = 2012
value = [1101, 911]
class = recidiviste
gini = 0.411
samples = 1604
value = [1140, 464]
class = recidiviste











Propose an evaluation in terms of training and testing accuracies using 5-cross validation on two decision trees that have different typologies

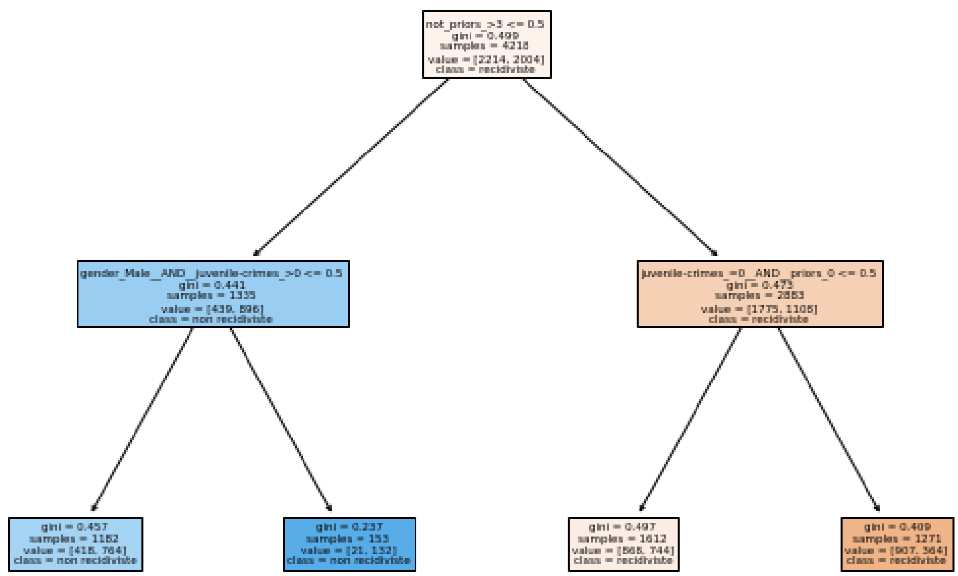
[5]:
```
# We kept the two first topologies
treesParmeters = treesParmeters[:2]
nbTree = len(treesParmeters)
```

[6]:
```
for i in range (nbTree) :
    parameters= treesParmeters[i]
    clf = tree.DecisionTreeClassifier(
        splitter=parameters['splitter'],
        max_depth =parameters['max_depth'],
        min_samples_leaf =parameters['min_samples_leaf']
    )
```

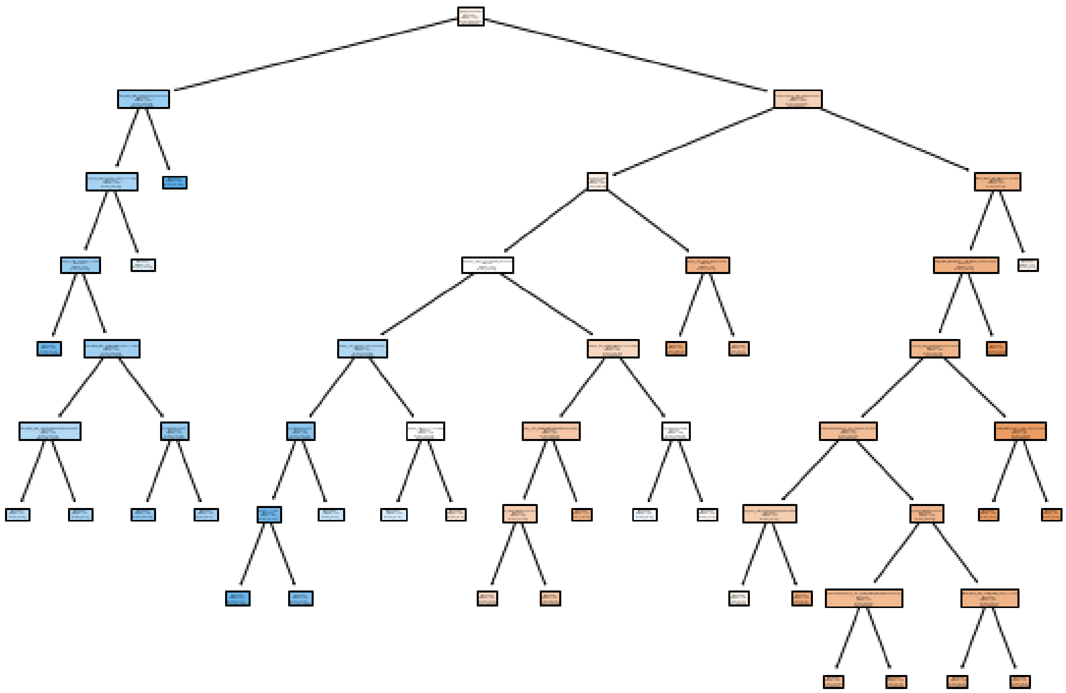
```
    X_train, X_test, y_train, y_test = train_test_split(train_examples,␣
↪train_labels, test_size=0.2, random_state=42)

    clf = clf.fit(X_train, y_train)
    scores = cross_val_score(clf, X_test, y_test, cv=2)
    print("The tree number %0.2i has %0.2f T accuracy with a standard deviation␣
↪of %0.2f" % (i,scores.mean(), scores.std()))
    plt.figure(figsize=(10,7))
    tree.plot_tree(clf,
                        feature_names= (features),
                        class_names= ("recidiviste", "non recidiviste"),
                        filled=True)
    plt.show()
```

```
The tree number 00 has 0.66 T accuracy with a standard deviation of 0.02
```



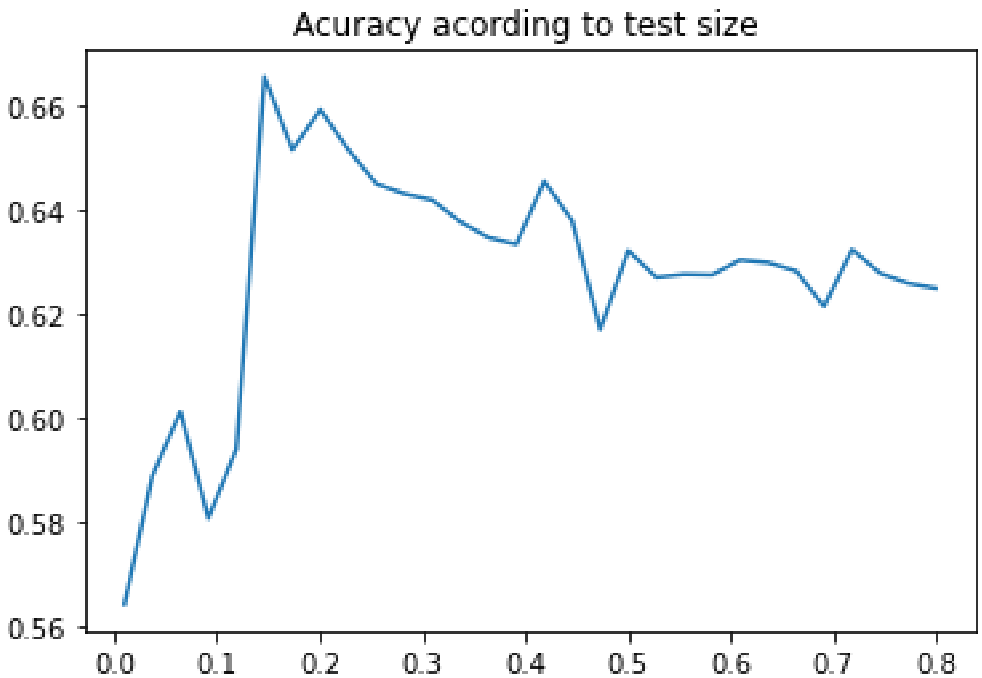

```
The tree number 01 has 0.64 T accuracy with a standard deviation of 0.01
```



Propose an experimental study that shows the transition phase from underfitting to overfitting

[7]:
```
parameters= treesParmeters[0]
acuracy =[]
test_sizes=np.linspace(0.01,0.8,30)
for test_size in test_sizes :
    clf = tree.DecisionTreeClassifier(
        splitter=parameters['splitter'],
        max_depth =parameters['max_depth'],
        min_samples_leaf =parameters['min_samples_leaf']
    )
    X_train, X_test, y_train, y_test = train_test_split(train_examples,␣
 ↪train_labels, test_size=test_size, random_state=42)

    clf = clf.fit(X_train, y_train)
    scores = cross_val_score(clf, X_test, y_test, cv=2)
    acuracy.append(scores.mean())
    #print(f"for a test size of {1/i} we have an accuracy of {scores.mean()}␣
 ↪with a standard deviation of {scores.std()}")
plt.title('Acuracy acording to test size ')
plt.plot(test_sizes,acuracy)
plt.show()
```



2.0.1 Remarks

- With small tests size, we have bad result because our graph is to close of our data
  - **this is overfitting**
- With high tests size, our results start to be be lower because the graph is constructed with too few data
  - **this is underfitting**

Construct the confusion matrix on a particular good configuration (after explaining your choice)

[8]:
```
parameters= treesParmeters[0]

clf = tree.DecisionTreeClassifier(
        splitter=parameters['splitter'],
        max_depth =parameters['max_depth'],
        min_samples_leaf =parameters['min_samples_leaf']
    )
X_train, X_test, y_train, y_test = train_test_split(train_examples,␣
 ↪train_labels, test_size=test_size, random_state=42)

clf = clf.fit(X_train, y_train)
#confusion_matrix(y_train, y_test)
```

Provide an evaluation of the fairness of the model based on the False Positive Rate

[ ]:
```
```