**import** scipy.io **as** sio  
**import** sklearn.model\_selection **as** sms  
**import** sklearn.naive\_bayes **as** snb  
**import** sklearn.neighbors **as** sn  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**import** matplotlib.colors **as** mc  
**import** math  
  
*#1*iris = []  
**with** open(**'iris.data'**) **as** file:  
 **for** line **in** file:  
 iris.append(line.rstrip().split(**','**))  
iris.data = iris[**'iris.data'**]  
iris.target = iris[**'iris.target'**]  
train, test, train\_targets, test\_targets = sms.train\_test\_split(iris.data, iris.target, test\_size=0.5, random\_state=42)  
print(**"learning samples: "**,train)  
print(**"test samples: \n"**,test)  
*#1  
  
  
#2*print(**" \n 2"**)  
clf = tree.DecisionTreeClassifier()  
clf = clf.fit(train, train\_targets)  
dot\_data = StringIO()  
tree.export\_graphviz(clf, out\_file=dot\_data)  
graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())  
graph.write\_pdf(**"tree.pdf"**)  
*#2  
  
  
#3*print(**" \n 3"**)  
y = clf.predict(test)  
gini\_score = clf.score(test,test\_targets)  
print(**"classifier efficiency:"**,gini\_score)  
print(**"faulty classified: "**, (y != test\_targets).sum())  
*#3  
  
  
#4*print(**" \n 4"**)  
print(clf)  
print(**'deciding criterion'**, clf.get\_params()[**'criterion'**])  
*#4  
  
  
#5*print(**" \n 5"**)  
y=clf.predict(iris.data)  
print(**'learning set efficiency: '**)  
popr\_zaklas = (iris.target==y).sum()  
print(**'Correctly classified:'**,popr\_zaklas)  
print(float(popr\_zaklas)/len(y)\*100,**"%"**)  
*#5  
  
  
#6*tree.export\_graphviz(clf, out\_file=dot\_data)  
graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue()) graph.write\_pdf(**"tree2.pdf"**)  
clf = tree.DecisionTreeClassifier(max\_depth=2)  
clf = clf.fit(train, train\_targets)  
y = clf.predict(test) print(**'tree depth: 2:'**)  
print(**'learning set efficiency: '**)  
popr\_zaklas = (y == test\_targets).sum()  
print(**'Correctly classified: '**,popr\_zaklas)  
print(float(popr\_zaklas)/len(y)\*100,**"%"**) dot\_data = StringIO()  
tree.export\_graphviz(clf, out\_file=dot\_data)  
graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())  
graph.write\_pdf(**"tree3.pdf"**)  
*#6  
  
  
#7*print(**" \n 7"**)  
entropy = tree.DecisionTreeClassifier(criterion=**'entropy'**)  
*#7  
  
  
#8*print(**" \n 8"**)  
entropy = entropy.fit(train, train\_targets)  
entropy\_score = entropy.score(test,test\_targets)  
print(**'entropy efficiency: '**,entropy\_score)  
print(**'gini efficiency'**, gini\_score)  
*#8  
  
  
#9*print(**" \n 9"**)  
clr = tree.DecisionTreeClassifier(min\_samples\_leaf=3, max\_leaf\_nodes=9)  
clr.fit(train, train\_targets)  
print(**'restrictions: Min\_samples\_leaf = 3 oraz Max\_leaf\_nodes = 9'**)  
print(**'efficiency: '**, round(clr.score(test, test\_targets)\*100, 2))  
*#9  
  
  
#10*print(**" \n 10"**)  
train, test, train\_targets, test\_targets = train\_test\_split(iris.data[:, 0:1], iris.target, test\_size=0.5, random\_state=42)  
clf = tree.DecisionTreeClassifier()  
clf.fit(train, train\_targets)  
print(**'2 first atributes qualifier'**)  
print(**'efficiency: '**, round(clf.score(test, test\_targets)\*100, 2))  
*#10*