```
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
```