# Lab 1: Decision Tree

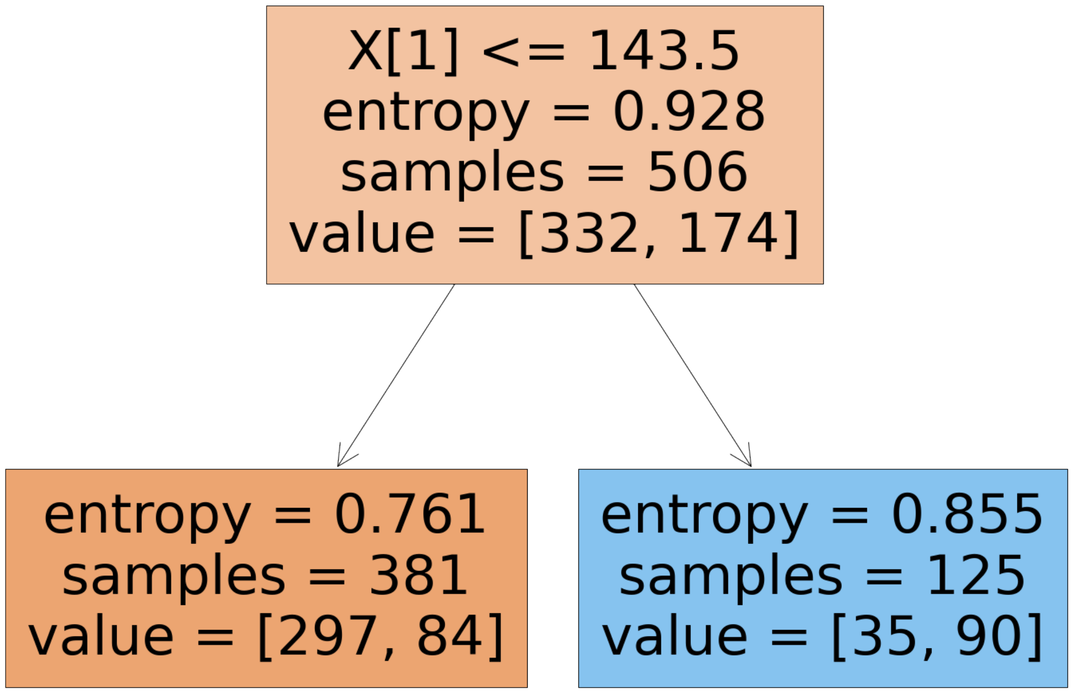
Martin Michaux i6220118

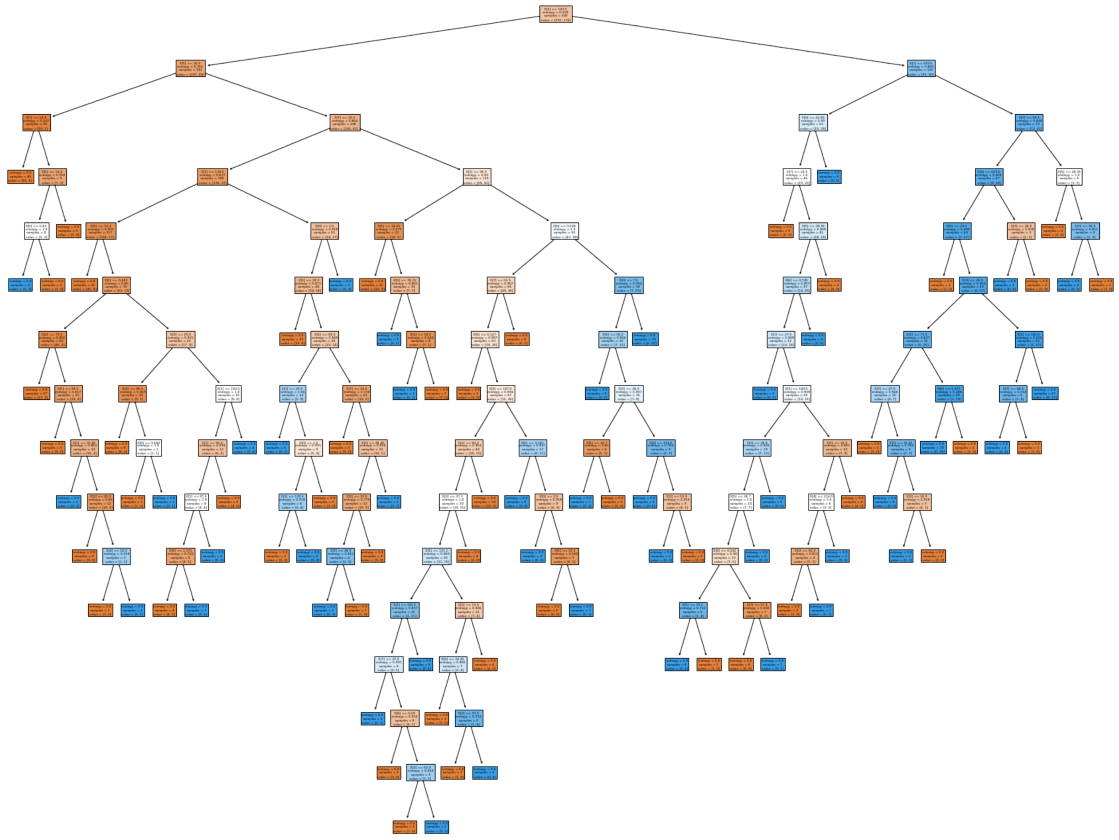
November 9, 2020

[15]:
```
#Question A:
import pandas as pd
data = pd.read_csv('diabetes.csv')
#output matrix Y
Y = data['class']
#input matrix X
X = data.drop(['class'],axis=1)


#Question B:
from sklearn import tree
#n represents the size of the training set, here for the diabete it is 768
n = 768
#creating the decision tree classifier from the data
clf = tree.DecisionTreeClassifier(criterion = 'entropy', min_samples_leaf = n)
#train the classifier
clf = clf.fit(X, Y)

#output Y of prediction of the input X on our classifier
Yp = clf.predict(X)
from sklearn.metrics import accuracy_score
#calculates the accuracy of that output from the true Y output
acc = accuracy_score(Y, Yp)

from sklearn.model_selection import train_test_split
#get training data set and test data set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.34,␣
 ↪random_state=10)


#Question C:

#one level decision tree:
#creating the classifier of the decision tree but only with 1 node depth
clfOneLevel = tree.DecisionTreeClassifier(criterion = 'entropy',
max_depth = 1)
#training this classifier with the train data set
```

```python
clfOneLevel = clfOneLevel.fit(X_train, Y_train)
#display the decision tree
from matplotlib import pyplot as plt
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clfOneLevel,filled=True)
#output Y of prediciton of the X input of the train data set
YpXtrain = clfOneLevel.predict(X_train)
#accuracy of that output from the Y output training data set
accOneLevelTraining = accuracy_score(Y_train, YpXtrain)
#output Y of prediciton of the X input of the test data set
YpOneLevel = clfOneLevel.predict(X_test)
#accuracy of that output from the Y output training data set
accOneLevelHoldOutVal = accuracy_score(Y_test, YpOneLevel)

#multi level decision tree:
#creating the classifier of the decision tree with no maximum node depth
clfMultiLevel = tree.DecisionTreeClassifier(criterion = 'entropy',
max_depth = None)
#training this classifier with the train data set
clfMultiLevel = clfMultiLevel.fit(X_train, Y_train)
#show decision tree
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clfMultiLevel,filled=True)
#output Y of prediciton of the X input of the train data set
YpXtrain = clfMultiLevel.predict(X_train)
#accuracy of that output from the Y output training data set
accMultiLevelTraining = accuracy_score(Y_train, YpXtrain)
#output Y of prediciton of the X input of the test data set
YpMultiLevel = clfMultiLevel.predict(X_test)
#accuracy of that output from the Y output training data set
accMultiLevelHoldOutVal = accuracy_score(Y_test, YpMultiLevel)

print('1) One level decision tree training accuracy: ' , accOneLevelTraining)
print('2) One level decision tree hold-out validation accuracy: ' ,␣
 ↪accOneLevelHoldOutVal)
print('1) Multi level decision tree training accuracy: ' , accMultiLevelTraining)
print('2) Multi level decision tree hold-out validation accuracy: ' ,␣
 ↪accMultiLevelHoldOutVal)
```

```
1) One level decision tree training accuracy:  0.7648221343873518
2) One level decision tree hold-out validation accuracy:  0.7213740458015268
1) Multi level decision tree training accuracy:  1.0
2) Multi level decision tree hold-out validation accuracy:  0.7251908396946565
```


X[1] <= 143.5
entropy = 0.928
samples = 506
value = [332, 174]
entropy = 0.761
samples = 381
value = [297, 84]
entropy = 0.855
samples = 125
value = [35, 90]



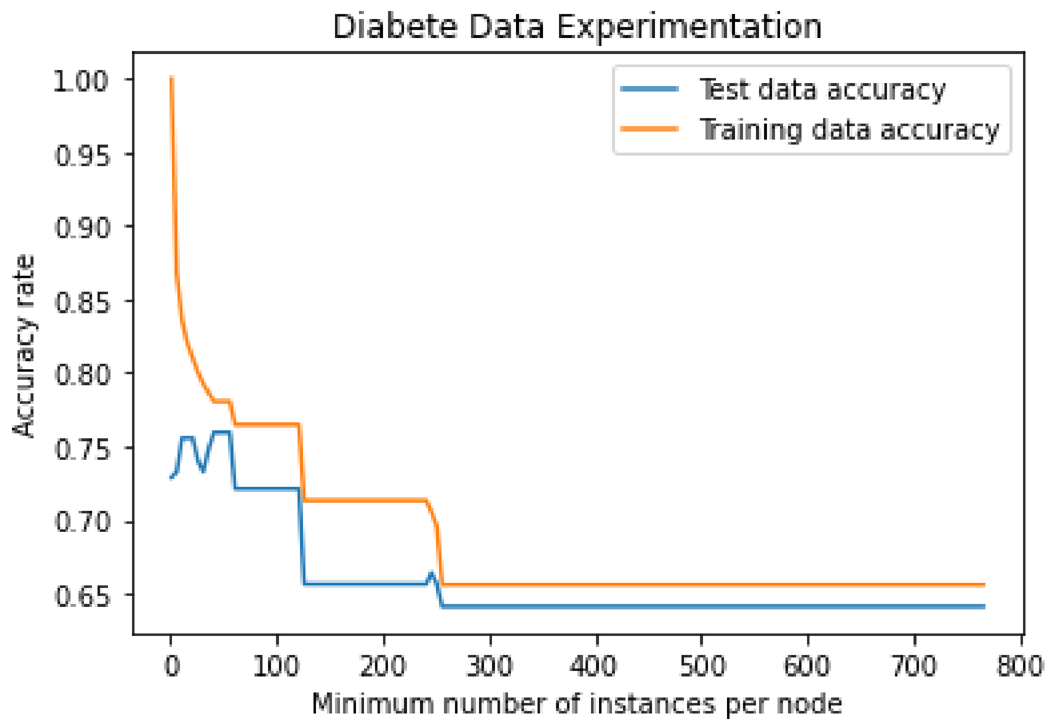
Explanation:

1) Training a decision tree with only one level (a depth of max 1 node) on the training set should have an accuracy lower than the one of the multi level, so if we use only one level in that tree, the decision trees only contains 1 node (which is the one that gives the higher accuracy for one level). For a training of a multi level decision tree on the training set, it is obvious that, having no maximum depth has a minimum number of instances per leaf that gives an accuracy of 1. The difference between the two accuracies of the one level decision tree is due to the fact that the classifier of 1 node is not perfect, and the classifier of a none max depth is "perfect" for the training data set.

2) Testing a decision tree with one level (for the hold-out validation) show that the accuracy of only one node is not that bad, the more node we "add" (increase the max depth) the more accurate the testing should be (until we reach the optimal point) With multi level (for the hold-out validation) the accuracy is almost the same than in one level, which proves that there exists an overfitting because after reaching the optimality (in terms of max depth), the accuracy of the test decreases
and the one of the instance set increase.

[17]:
```
#Question D
```

```python
#list of the accuracy rate of the test data
accListTest=[]
#list of the accuracy rate of the training data
accListTrain=[]
#list of the number minimum of instance per node
iX = []
for i in range(1, n, 5):
    #creating the classifier of the decision tree with a number minimum of␣
 ↪instance per node (between 1 and n)
    clf = tree.DecisionTreeClassifier(criterion = 'entropy',min_samples_leaf = i)
    #training this classifier with the train data set
    clf = clf.fit(X_train, Y_train)
    #output Y of prediciton of the X input of the test data set
    YpTest = clf.predict(X_test)
    #accuracy of that output from the Y output training data set
    accTest = accuracy_score(Y_test, YpTest)
    #add that accuracy to the accuracy of the test list
    accListTest.append(accTest)
    #output Y of prediciton of the X input of the train data set
    YpTrain = clf.predict(X_train)
    #accuracy of that output from the Y output training data set
    accTrain = accuracy_score(Y_train,YpTrain)
    #add that accuracy to the accuracy of the training list
    accListTrain.append(accTrain)
    #add the info of the number minimum of instance per node
    iX.append(i)

import matplotlib.pyplot as plt2
#plot the data found for the instance and test data compared to the number␣
 ↪minimum of instance per node
plt2.plot(iX, accListTest, label = "Test data accuracy")
plt2.plot(iX, accListTrain, label = "Training data accuracy")
plt2.legend(loc='upper right')

plt2.xlabel('Minimum number of instances per node')

plt2.ylabel('Accuracy rate')

plt2.title('Diabete Data Experimentation')

plt2.show()
```



Overfitting: in range of x [1,+-50] In that range (mostly in the beginning), we can see that the accucary of the instance data set (in orange) is much more high that the accuracy of the test set (in blue). This is due to the fact that because the decision tree (the classifier) has been created from the instance data, the less number of instance per node, the more the instance data accurate is. This is why, with a small minimum number of instance per node, we can clearly see that there is a huge difference with the accuracy of the test data.

Optimality : where x = 60, 125, 250 These are optimal points because at those Xs, this is where the accuracy of the test data and the accuracy of the instance data are almost equal, which means that the classifier with that number minimum of instance per node has a good evaluation since the test data has the same accuracy as the instance data

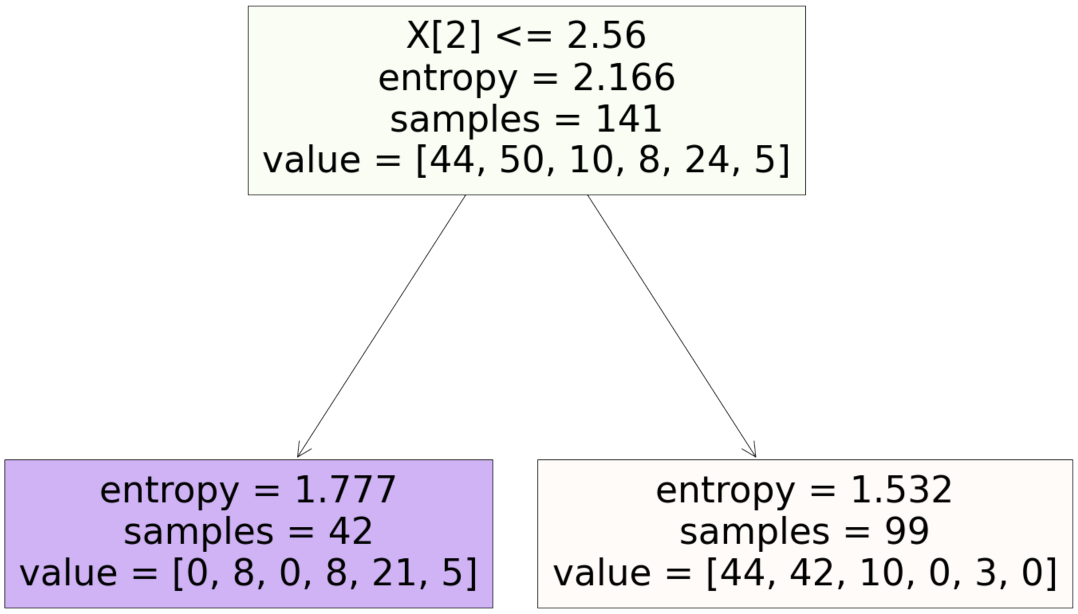
Underfitting : in range x [+-250, 768] When “we” reach that region, we can clearly see that both accuracies are very low and stagnate, which means that the classifier is underfitting the 2 data types because if we decrease the number minimum of instance per node, the accuracies will both increase

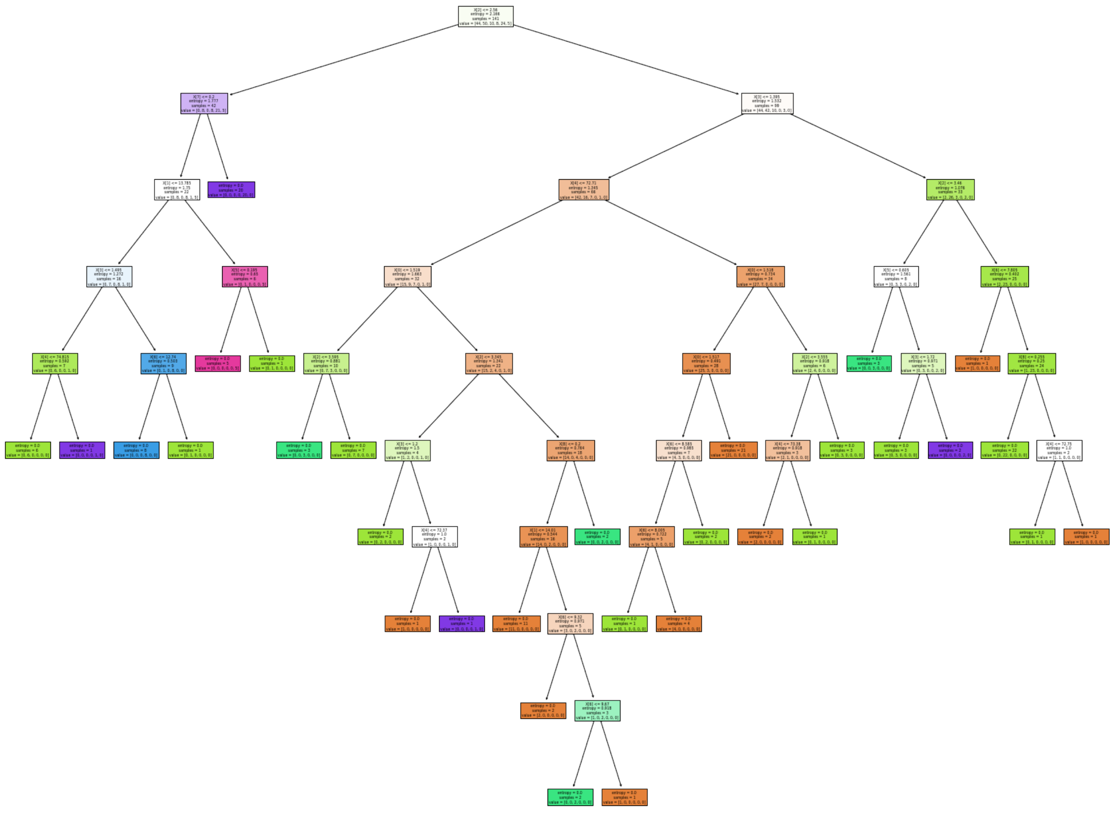
[18]:
```
import pandas as pd
data = pd.read_csv('glass.csv')
#output matrix Y
Y = data['class']
#input matrix X
X = data.drop(['class'],axis=1)
```

```
#Question B:
from sklearn import tree
#n represents the size of the training set, here for the diabete it is 214
n = 214
#creating the decision tree classifier from the data
clf = tree.DecisionTreeClassifier(criterion = 'entropy', min_samples_leaf = n)
#train the classifier
clf = clf.fit(X, Y)

#output Y of prediction of the input X on our classifier
Yp = clf.predict(X)
from sklearn.metrics import accuracy_score
#calculates the accuracy of that output from the true Y output training data set
acc = accuracy_score(Y, Yp)

from sklearn.model_selection import train_test_split
#get training data set and test data set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.34,␣
 ↪random_state=10)



#Question C:

#one level decision tree:
#creating the classifier of the decision tree but only with 1 node depth
clfOneLevel = tree.DecisionTreeClassifier(criterion = 'entropy',
max_depth = 1)
#training this classifier with the train data set
clfOneLevel = clfOneLevel.fit(X_train, Y_train)
#show decision tree
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clfOneLevel,filled=True)
#output Y of prediciton of the X input of the train data set
YpXtrain = clfOneLevel.predict(X_train)
#accuracy of that output from the Y output training data set
accOneLevelTraining = accuracy_score(Y_train, YpXtrain)
#output Y of prediciton of the X input of the test data set
YpOneLevel = clfOneLevel.predict(X_test)
#accuracy of that output from the Y output training data set
accOneLevelHoldOutVal = accuracy_score(Y_test, YpOneLevel)

#multi level decision tree:
#creating the classifier of the decision tree with no maximum node depth
clfMultiLevel = tree.DecisionTreeClassifier(criterion = 'entropy',
max_depth = None)
#training this classifier with the train data set
clfMultiLevel = clfMultiLevel.fit(X_train, Y_train)
```

```
#show decision tree
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clfMultiLevel,filled=True)
#output Y of prediciton of the X input of the train data set
YpXtrain = clfMultiLevel.predict(X_train)
#accuracy of that output from the Y output training data set
accMultiLevelTraining = accuracy_score(Y_train, YpXtrain)
#output Y of prediciton of the X input of the test data set
YpMultiLevel = clfMultiLevel.predict(X_test)
#accuracy of that output from the Y output training data set
accMultiLevelHoldOutVal = accuracy_score(Y_test, YpMultiLevel)
print("1)One level decision tree training accuracy: " , accOneLevelTraining)
print("2)One level decision tree hold-out validation accuracy: " ,␣
 ↪accOneLevelHoldOutVal)
print("1)Multi level decision tree training accuracy: " , accMultiLevelTraining)
print("2)Multi level decision tree hold-out validation accuracy: " ,␣
 ↪accMultiLevelHoldOutVal)
```

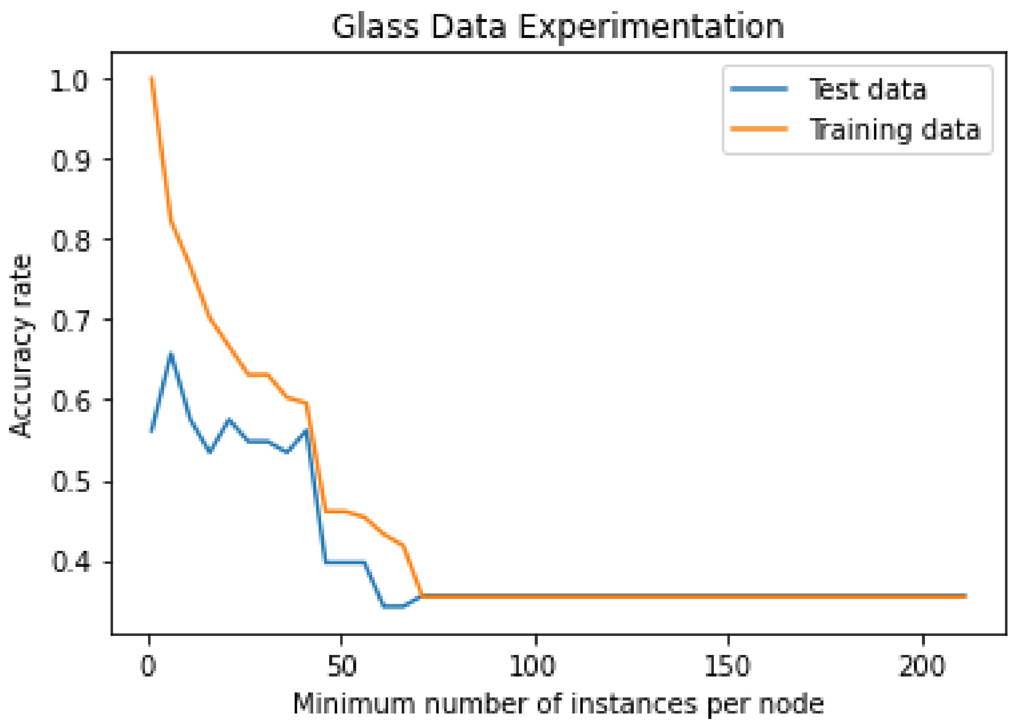
```
1)One level decision tree training accuracy:  0.46099290780141844
2)One level decision tree hold-out validation accuracy:  0.4246575342465753
1)Multi level decision tree training accuracy:  1.0
2)Multi level decision tree hold-out validation accuracy:  0.589041095890411
```





Explanation: The explanation of the training of the decision tree here is partially the same as the one of the Diabete data. (except it's less accurate since we had less instances data) The difference between the accuracy rates is still due to the fact that the deeper we go in the decision tree, the higher the accuracy should be, but once reaching the optimality, the accuracy of the training data still increase and the one of the test deacreses. Also, the accuracy of the test and the instance set for the one level here is not very accurate, but, we can crearly see that the multi level decision tree is not that bad for the test compared to the instance data set, which proves that the classifier is not too bad.

[19]:
```
#Question D

#list of the accuracy rate of the test data
accListTest=[]
#list of the accuracy rate of the training data
accListTrain=[]
#list of the number minimum of instance per node
iX = []
for i in range(1, n, 5):
    #creating the classifier of the decision tree with a number minimum of␣
 ↪instance per node (between 1 and n)
    clf = tree.DecisionTreeClassifier(criterion = 'entropy',min_samples_leaf = i)
    #training this classifier with the train data set
    clf = clf.fit(X_train, Y_train)
    #output Y of prediciton of the X input of the test data set
    YpTest = clf.predict(X_test)
    #accuracy of that output from the Y output training data set
    accTest = accuracy_score(Y_test, YpTest)
    #add that accuracy to the accuracy of the test list
    accListTest.append(accTest)
    #output Y of prediciton of the X input of the train data set
    YpTrain = clf.predict(X_train)
    #accuracy of that output from the Y output training data set
    accTrain = accuracy_score(Y_train,YpTrain)
    #add that accuracy to the accuracy of the training list
    accListTrain.append(accTrain)
    #add the info of the number minimum of instance per node
    iX.append(i)

import matplotlib.pyplot as plt
#plot the data found for the instance and test data compared to the number␣
 ↪minimum of instance per node
plt.plot(iX, accListTest, label = "Test data accuracy")
plt.plot(iX, accListTrain, label = "Training data accuracy")
plt.legend(loc='upper right')

plt.xlabel('Minimum number of instances per node')

plt.ylabel('Accuracy rate')

plt.title('Glass Data Experimentation')

plt.show()
```



Overfitting: in range of x [1,+-10] In that range (mostly in the beginning), we can see that the accucary of the instance data set (in orange) is much more high that the accuracy of the test set (in blue). This is due to the fact that because the decision tree (the classifier) has been created from the instance data, the less number of instance per node, the more the instance data accurate is. This is why, with a small minimum number of instance per node, we can clearly see that there is a huge difference with the accuracy of the test data.

Optimality : where x = 45 These are optimal points because at those Xs, this is where the accuracy of the test data and the accuracy of the instance data are almost equal, which means that the classifier with that number minimum of instance per node has a good evaluation since the test data has the same accuracy as the instance data

Underfitting: in range of x [+-45,214] When “we” reach that region, we can clearly see that both accuracies are very low and stagnate, which means that the classifier is underfitting the 2 data types because if we decrease the number minimum of instance per node, the accuracies will both increase