Lab1

November 9, 2021

1 Lab Tasks

a. Study the classification problems (given in Section 2) using the info from Appendix B.

b. Study sklearn: you need to study DecisionTreeClassifier and sklearn.model_selection module (the latter allows us to estimate training accuracy rate and hold-out accuracy rate). A basic info is provided in Appendix A.

c. Train one-level decision trees and multi-level decision trees on the two data sets. Determine the accuracy rates of the resulting classifiers using the training set and hold- out validation1. Explain why there is a difference in the accuracy rates. Compare one- level decision trees and multi-level decision trees in terms of explainability.

d. Experiment with multi-level decision trees and error pre-pruning by changing the option min_samples_leaf from 0 to the size of the datasets (use some step). (The option min_samples_leaf determines the min number of training instances in the leaf nodes of the decision trees.) Estimate the accuracy rates of the resulting decision trees using the training set and hold-out validation. Plot the accuracy rates based on the training set and hold-out validation for min_samples_leaf from 1 to the size of the datasets with step of 5. Identify the regions of underfitting, optimality, and overfitting. Explain how you have identified these regions.

2 Lena Feiler - i6246119

[12]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn import tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from statistics import mean

# Import diabetis data file
diabetis_data = pd.read_csv('diabetes.csv')

# Import glass data file
glass_data = pd.read_csv('glass.csv')
```

2.1 Preparing data fram for training classification problems

[13]:
```
Y_diabetis = diabetis_data['class']
X_diabetis = diabetis_data.drop(['class'], axis = 1)

Y_glass = glass_data['class']
X_glass = glass_data.drop(['class'], axis = 1)
```

2.2 Setting and training decision-tree classifiers

train classifiers for diabetis data set

[14]:
```
# average accuracy of decision trees
def avg_accuracy(X, Y, max_depth, min_samples_leaf):
    num_nodes = []
    test_acc = []
    train_acc = []

    for i in range(1, 25) :
        # to compare differently generated decisoon trees
        X_train, X_test, Y_train, Y_test = train_test_split(X, Y,test_size=0.
 ↪34, random_state=i)
        clf = tree.DecisionTreeClassifier(criterion = 'entropy',
                                          max_depth = max_depth,
                                          min_samples_leaf = min_samples_leaf)
        clf = clf.fit(X_train, Y_train)
        num_nodes.append(clf.tree_.node_count)

        Yp_test = clf.predict(X_test)
        Yp_train = clf.predict(X_train)

        test_acc.append(accuracy_score(Y_test, Yp_test))
        train_acc.append(accuracy_score(Y_train, Yp_train))

    avg_test_acc = mean(test_acc)
    avg_train_acc = mean(train_acc)

    return avg_test_acc, avg_train_acc, num_nodes
```

3 Exercise C

Train one-level decision trees and multi-level decision trees on the two data sets. Determine the accuracy rates of the resulting classifiers using the training set and hold- out validation1. Explain why there is a difference in the accuracy rates. Compare one- level decision trees and multi-level decision trees in terms of explainability.

[15]:
```
# One layered decision tree for diabetis dataset
# as nothing was specified, no lower bound on the amount of samples needed to␣
 ↪create a leaf was used
avg_test_diab_one, avg_train_diab_one, X = avg_accuracy(X_diabetis, Y_diabetis,␣
 ↪max_depth = 1, min_samples_leaf = 1)

# Multi layered decision tree for diabetis dataset
avg_test_diab_multi, avg_train_diab_multi, X = avg_accuracy(X_diabetis,␣
 ↪Y_diabetis, max_depth = None, min_samples_leaf = 1)
```

[16]:
```
# One layered decision tree for glass dataset
# as nothing was specified, no lower bound on the amount of samples needed to␣
 ↪create a leaf was used
avg_test_glass_one, avg_train_glass_one, X = avg_accuracy(X_glass, Y_glass,␣
 ↪max_depth = 1, min_samples_leaf = 1)

# Multi layered decision tree for glass dataset
avg_test_glass_multi, avg_train_glass_multi, X = avg_accuracy(X_glass, Y_glass,␣
 ↪max_depth = None, min_samples_leaf = 1)
```

[17]:
```
#calcultaing the difference in accuracy from one-level to mutli-level
delta_acc_diab = avg_test_diab_multi - avg_test_diab_one
delta_acc_glass = avg_test_glass_multi - avg_test_glass_one
```

[18]:
```
print('Average values for the Diabetis data set, with a (1/3) hold-out␣
 ↪validation')
print('One-level decison tree accuracy for hold-out validation set:' ,␣
 ↪avg_test_diab_one)
print('One-level decison tree accuracy for training data set:' ,␣
 ↪avg_train_diab_one)
print('Multi-level decison tree accuracy for hold-out validation set:' ,␣
 ↪avg_test_diab_multi)
print('Multi-level decison tree accuracy for training data set:' ,␣
 ↪avg_train_diab_multi)
print()
print('Average values for the Glass data set, with a (1/3) hold-out validation')
print('One-level decison tree accuracy for hold-out validation set:' ,␣
 ↪avg_test_glass_one)
print('One-level decison tree accuracy for training data set:' ,␣
 ↪avg_train_glass_one)
print('Multi-level decison tree accuracy for hold-out validation set:' ,␣
 ↪avg_test_glass_multi)
print('Multi-level decison tree accuracy for training data set:' ,␣
 ↪avg_train_glass_multi)
print()
```

```
print('Difference in accuracy of one and multi layered decision trees using the␣
 ↪hold-out validation set:')
print('On diabetis data set: ', delta_acc_diab)
print('On glass data set: ', delta_acc_glass)
```

```
Average values for the Diabetis data set, with a (1/3) hold-out validation
One-level decison tree accuracy for hold-out validation set: 0.7080152671755725
One-level decison tree accuracy for training data set: 0.7382246376811594
Multi-level decison tree accuracy for hold-out validation set:
0.7037213740458015
Multi-level decison tree accuracy for training data set: 1.0

Average values for the Glass data set, with a (1/3) hold-out validation
One-level decison tree accuracy for hold-out validation set: 0.4183789954337899
One-level decison tree accuracy for training data set: 0.45005910165484636
Multi-level decison tree accuracy for hold-out validation set:
0.6529680365296804
Multi-level decison tree accuracy for training data set: 1.0

Difference in accuracy of one and multi layered decision trees using the hold-
out validation set:
On diabetis data set:  -0.00429389312977102
On glass data set:  0.23458904109589046
```

3.0.1 Answer (c) Accuracy and Explainability of the classification problem

Looking at the difference in accuracy printed above, we can clearly see, that, for the diabetis data set, there exist a small decrease in the accuracy of the classification when we go from a one-level decision tree to a multi-level decision tree. Since the one-level classification works better in this case, we can assume, that the attribute used to classify the instances has a high entropy. This allows an accurate classification of instances, when using this feature. Furthermore, the multi-level decision tree leads to overfitting, since the data is already well classified.

For the glass data set on the other hand, we can see an increase almost 25% when using a multi-level decision tree. This shows, that we need more attributes to correctly classify a certain instance of the data set. This also means, that, other than in the diabetis data set, we have no attribute with a very high entropy, and therefore our one-level decision tree is not very accurate.

In terms of explainability, we can say that one-level decision trees are easily explainable since they only use one feature to classify the given data instances. Therefore, explaining the classification process is very straight forward and easily understandable. Multi-level decision trees are more complex since they use several features to classify the given instances. Hence, explaining this process is also more complex than in the case of the on-level decision tree.

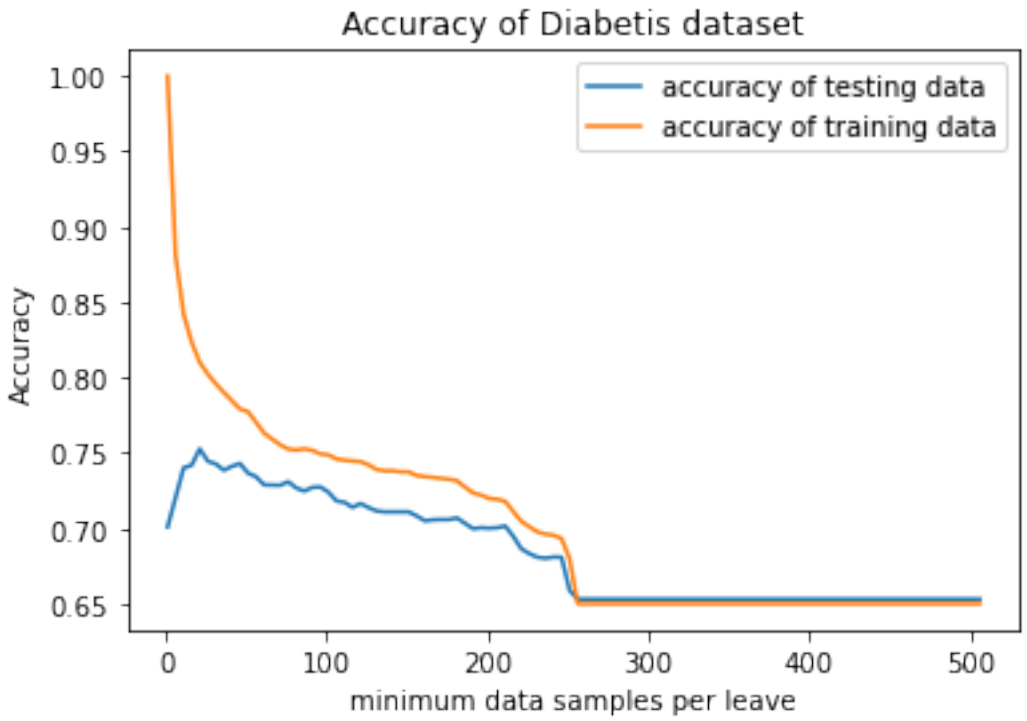
4 Exercise D

Experimenting with multi-level decision trees

[19]:
```
# experiment performed on diabetis data set
acc_res_train = []
acc_res_test = []

result_X = []

for n in range(1,(int) (len(diabetis_data) * 0.66 + 1), 5): #steps of 5
    avg_test, avg_train, num_nodes = avg_accuracy(X_diabetis, Y_diabetis,␣
 ↪max_depth = None, min_samples_leaf = n)
    acc_res_train.append(avg_train)
    acc_res_test.append(avg_test)
    result_X.append(n)
```

[20]:
```
# Plot the accuracy rates based on the training set and hold-out validation
plt.plot(result_X,acc_res_test, label = "accuracy of testing data")
plt.plot(result_X,acc_res_train, label = "accuracy of training data")

plt.title("Accuracy of Diabetis dataset")
plt.ylabel("Accuracy")
plt.xlabel("minimum data samples per leave")
plt.legend()
```

[20]:
```
<matplotlib.legend.Legend at 0x7f9b0f4d41f0>
```



[21]:
```
# experiment performed on glass data set
acc_res_train = []
acc_res_test = []

result_X = []

for n in range(1,(int) (len(glass_data) * 0.66 + 1), 5): #steps of 5
    avg_test, avg_train, num_nodes = avg_accuracy(X_glass, Y_glass, max_depth =␣
 ↪None, min_samples_leaf = n)

    acc_res_train.append(avg_train)
    acc_res_test.append(avg_test)
    result_X.append(n)
```

[22]:
```
plt.plot(result_X,acc_res_test, label = "testing data")
plt.plot(result_X,acc_res_train, label = "training data")

plt.title("Accuracy of Glass dataset")
plt.ylabel("Accuracy")
plt.xlabel("minimum data samples per leave")
plt.legend()
```

[22]: <matplotlib.legend.Legend at 0x7f9b101cef40>

4.0.1 Answer (d) Experiment with multi-level decision trees and error pre-pruning

Note: we start with a minimum of 1 data sample per leave since we receive an errror when setting min_samples_leaf = 0 into the decisionTreeClassifier() function. Therefore, any leave can have from min_samples_leaf(=1) - (size of data set) many data samples.

We now identify the regions of underfitting, optimality, and overfitting using the two plots above. Diabaetis Data set: From the graph we can see that for a minimum amount of 70 - 100 samples per leave, the accuracy of both the testing and the training data set remains more or less constant. Therefore we have reached our optimality value around this point. Before this region, so for minimally 0 - 70 samples per leave the data is overfit. This can be seen by observing that the two graphs diverge. So, the accuracy on the training data is very high since a lot of instances are very precisely classified but the accuracy of the testing data is low due to the overfit. Furthermore, after the area of optimality, the data is underfit (at least 80 samples per leave), which we can observe since the accuracy on both data sets decreases after this point until they reach a point where they do not decrease any further.

Glass Data set: For the Glass dataset, for minimally 0 - 20 samples per leaf, the data is overfit (the training data has a very high accuracy and the testing data a low one, since the classification is too focused on the training data set). Our optimality lies in the region of minimum 20-40 data samples per leave, since both data sets stay more or less constant around their current value and do not noticeably decrease or increase. With at minimally 40 samples per leaf, our data is underfit and the accuracy of both the training and test data decreases significantly.