Aprendizagem 2021/22

Homework I – Group XXX

I. Pen-and-paper

1) First, we transformed the design matrix using the basis function

$$X = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 5 \\ 1 & 0 & 2 & 4 \\ 1 & 1 & 2 & 3 \\ 1 & 2 & 0 & 7 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 2 \\ 1 & 0 & 2 & 9 \end{bmatrix} \xrightarrow[\phi(x)]{} \Phi = \begin{bmatrix} 1 & 1.414 & 2 & 2.828 \\ 1 & 5.196 & 27 & 140.3 \\ 1 & 4.472 & 20 & 89.44 \\ 1 & 3.742 & 14 & 52.38 \\ 1 & 7.280 & 53 & 385.8 \\ 1 & 1.732 & 3 & 5.196 \\ 1 & 2.828 & 8 & 22.63 \\ 1 & 9.220 & 85 & 783.7 \end{bmatrix} \quad \Phi^{T} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1.41 & 5.196 & 4.47 & 3.74 & 7.28 & 1.7 & 2.8 & 9.22 \\ 2 & 27 & 20 & 14 & 53 & 3 & 8 & 85 \\ 2.82 & 140.3 & 89.4 & 52.4 & 386 & 5.2 & 23 & 784 \end{bmatrix}$$

Then, we will calculate the vector W that minimizes the square-error loss function, this vector is given by the following expression

$$W = (\Phi^{T}\Phi)^{-1}\Phi^{T}\ \text{z where z} = [1 \quad 3 \quad 2 \quad 0 \quad 6 \quad 4 \quad 5 \quad 7]^{T} \text{(output vector)}$$

So, for the training dataset we obtain that $W = [4.5835 \quad -1.6872 \quad 0.3377 \quad -0.01331]^{T}$

2) To measure the differences between the observed values and the predictions given by the model, in order to test it, we'll use the RMSE

For that we'll need to calculate the predictions of our model:

$$X_{train} = \begin{bmatrix} 1 & 2 & 0 & 0 \\ 1 & 1 & 2 & 1 \end{bmatrix} \xrightarrow[\phi(x)]{} \Phi_{\text{train}} = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 2.449 & 6 & 14.6969 \end{bmatrix}$$

Each prediction is given by the following polynomial regression model $\hat{z} = \Phi^{t}.\text{w} = [2.4536 \quad 2.2816]^{T}$

Therefore, the RMSE is:

$$RMSE = \sqrt{\frac{\sum_{i=9}^{10}(z_i - \hat{z}_i)^2}{2}} = \sqrt{\frac{(2-2.4536)^2 + (4-2.2816)^2}{2}} = 1.256$$

3) Firstly, we computed the equal depth binarization of y_3 , in which we used the median of its train values as the criteria for the binarization, $y_3\ median = 3.5$, and the class targets t_i, so we obtained the following table:

To learn a decision tree using ID3, we need to calculate the information gain (IG) of each variable, which is given by $IG = H(t) - H(t|y_i)\ where\ i = \{1,2,3\}$, and H the entropy.

Probabilities of the training dataset:

	$y_{3_{new}}$	t
x_1	0	N
x_2	1	N
x_3	1	N
x_4	0	N
x_5	1	P
x_6	0	P
x_7	0	P
x_8	1	P

$P(y_1 = 0)$	0.25
$P(y_1 = 1)$	0.5
$P(y_1 = 2)$	0.25
$P(t = N\|y_1 = 0)$	0.5
$P(t = N\|y_1 = 1)$	0.75
$P(t = N\|y_1 = 2)$	0
$P(t = P\|y_1 = 0)$	0.5
$P(t = P\|y_1 = 1)$	0.25
$P(t = P\|y_1 = 2)$	1

$P(y_2 = 0)$	0.25
$P(y_2 = 1)$	0.375
$P(y_2 = 2)$	0.375
$P(t = N\|y_2 = 0)$	1
$P(t = N\|y_2 = 1)$	0.666667
$P(t = N\|y_2 = 2)$	0.666667
$P(t = P\|y_2 = 0)$	0
$P(t = P\|y_2 = 1)$	0.333333
$P(t = P\|y_2 = 2)$	0.333333

$P(t = N\|y_3 = 0)$	0.5
$P(t = N\|y_3 = 1)$	0.5
$P(t = N\|y_3 = 0)$	0.5
$P(t = N\|y_3 = 1)$	0.5
$P(t = P\|y_3 = 0)$	0.5
$P(t = P\|y_3 = 1)$	0.5

Aprendizagem 2021/22

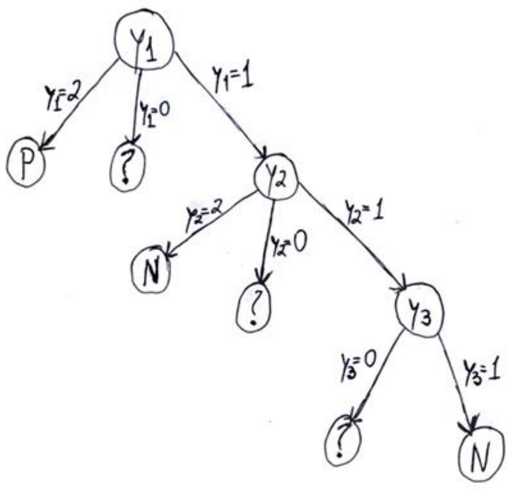
Homework I – Group XXX

-$H(t|y_1) = H(t|y_1 = 0) + H(t|y_1 = 1) + H(t|y_1 = 2) = 0.25 + 0.405639 + 0 = 0.655639$
- $H(t|y_2) = H(t|y_2 = 0) + H(t|y_2 = 1) + H(t|y_2 = 2) = 0 + 0.344361 + 0.344361 = 0.688722$
- $H(t|y_3) = H(t|y_3 = 0) + H(t|y_3 = 1) = 0.5 + 0.5 = 1$

Information Gain: $IG(y_1) = 1 - 0.655639 = 0.344361; IG(y_1) = 1 - 0.688722 = 0.311278;$
-$IG(y_3) = 1 - 1 = 0$

	y3	t
x1	0	0
x2	1	0
x6	0	1

We can conclude that the variable that will be used as root for the decision tree is y_1 as it is the variable with the highest IG, and $y_1 = 2$ always leads to $t = P$, and $y_0 = 0$ to uncertain (?). So, we will need to study now only the cases that $y_1 = 1$, which are on the following table:



	y2	y3	t
x1	1	0	N
x2	1	1	N
x4	2	0	N
x6	1	0	P

From this, we can conclude that $y_2 = 0$ is uncertain, and that $y_2 = 2$ leads to $t = N$. Since the information gain of y_2 and y_3 is the same (using analogous calculus from above), we choose the add y_2 to the decision tree, which leaves us with the final table:

Here, we can conclude that when $y_3 = 1$, then $t = 0$, and with analogous methods, it is clear that when $y_3 = 0$ the information gain will be 0, so that will lead to uncertainty. With all this information, we are now able to create the decision tree:

4) Using the "test" dataset on the decision tree obtained earlier (x_9 and x_{10}) we get a prediction $t_9 = P$, $t_{10} = N$(by following the branch), opposed to the true values of $t_9 = N$, $t_{10} = P$, so, the accuracy will be:

$$accuracy = \frac{\#\text{true positives} + \#\text{true negatives}}{\#\text{total observations}} = \frac{0}{2} = 0\%$$

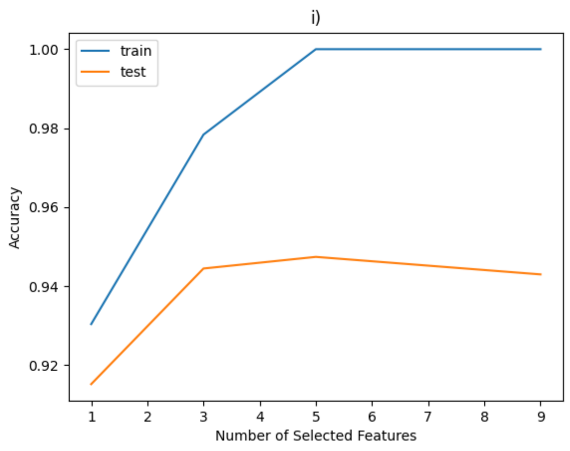
II. Programming and critical analysis

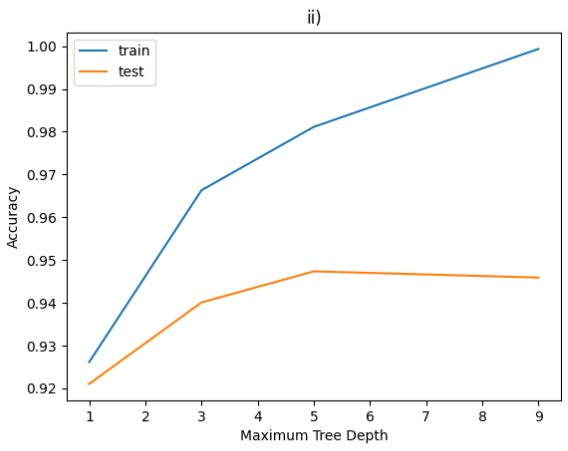
5) Using 10-fold cross validation for splitting the data into training and testing data, we trained our decision tree model in two different ways:

i) number of selected features ∈ {1,3,5,9}

ii) maximum tree depth ∈ {1,3,5,9}

After fitting the model, the score is calculated for both training and testing data. The score of each model can be compared in the given plots:





6) In both graphs we can observe that the values of the training and testing accuracy are somewhat correlated, this observation may be explained by:

- Increasing the depth/number of features leads us to an increase of the test data accuracy (these variables are correlated with the test accuracy). By choosing a lower value for these parameters we a restraining how much information/detail the model has to learn the problem, therefore by lowering the restrictions leads to an increase in learning ability.
- Increasing the depth/number of features leads us to an increase of the training data accuracy (these variables are correlated with the training accuracy). By increasing the parameters, we are giving the model more room, to learning all dataset irregularities instead of learning the problem that the dataset represents. Therefore, the increase of these variables leads to a gather overfitting risk, that represents a gather score in training data.

7) We can observe that as we increase the depth, the overfitting risk grows, since the model starts to learn quite well the training set (the accuracy of the training data increases). To some extent the accuracy of the test data follows the increase in training data, but when the maximum tree depth is above 5, it slightly drops, which leads us to conclude that the model isn't really leaning the problem at hand, but it's actually learning the dataset noise of the training data (overfitting). With this in mind, we believe that the best choice for the value of depth is 5 (best fit).

Aprendizagem 2021/22

Homework I – Group XXX

III. APPENDIX

5)

```
dataset = pandas.DataFrame(arff.loadarff(<path>)[0])
dataset["Class"] = dataset["Class"].str.decode('utf-8')
inputs = dataset.drop(columns=["Class"]).values; outputs = dataset["Class"].values
k_fold = KFold(n_splits=10, random_state=13, shuffle=True)

def calculateDictMean(dictionary):
       return [sum(dictionary[key])/len(dictionary[key]) for key in dictionary]

trainingAccuracy_i, testingAccuracy = {"1" : [], "3" : [], "5": [], "9": []}
    maxFeatures    [1, 3, 5, 9]:
    inputsNew =              (mutual_info_classif,  =maxFeatures).              (inputs ,outputs)
for        in
        train, test    k_fold.     (dataset):
        SelectKBest              k             fit_transform
   for  treeClassifier = tree.                        (          =          ,           =maxFeatures)
            split
            DecisionTreeClassifier criterion "entropy" max_features
        treeClassifier.    (inputsNew[train], outputs[train])
           fit

        trainingAccuracy_i[  append (maxFeatures)].score  (treeClassifier.    (inputsNew[train],
outputs[train]))
              str              append               score
        testingAccuracy_i[   (maxFeatures)].       (treeClassifier.    (inputsNew[test],
outputs[test]))
trainingAccuracyMean_i, testingAccuracyMean_i = calculateDictMean(trainingAccuracy_i),
calculateDictMean(testingAccuracy_i)

trainingAccuracy_ii, testingAccuracy_ii = {"1" : [], "3" : [], "5": [], "9": []}
for depth in [1, 3, 5, 9]:
    for train, test in k_fold.split(dataset):
        treeClassifier = tree.DecisionTreeClassifier(criterion="entropy", max_depth=depth)
        treeClassifier.fit(inputs[train], outputs[train])

        trainingAccuracy_ii[str(depth)].append(treeClassifier.score(inputs[train], outputs[train]))
        testingAccuracy_ii[str(depth)].append(treeClassifier.score(inputs[test], outputs[test]))

trainingAccuracyMean_ii, testingAccuracyMean_ii = calculateDictMean(trainingAccuracy_ii),
calculateDictMean(testingAccuracy_ii)

plt.plot([1, 3, 5, 9], trainingAccuracyMean_i); plt.plot([1, 3, 5, 9], testingAccuracyMean_i)
plt.show(); plt.clf()

plt.plot([1, 3, 5, 9], trainingAccuracyMean_ii); plt.plot([1, 3, 5, 9], testingAccuracyMean_ii)
plt.show()
```