TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 6

I. Pen-and-paper

1) The following are the original design matrix ($\boldsymbol{X}$) and the output vector ($\boldsymbol{z}$):

$$\boldsymbol{X} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 5 \\ 1 & 0 & 2 & 4 \\ 1 & 1 & 2 & 3 \\ 1 & 2 & 0 & 7 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 2 \\ 1 & 0 & 2 & 9 \end{bmatrix} \quad \boldsymbol{z} = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 0 \\ 6 \\ 4 \\ 5 \\ 7 \end{bmatrix}$$

As requested in the assignment, the function $\boldsymbol{\varphi_j(x) = ||x||_2^j}$ was applied to all entries of X (except for the first column, given the fact it's the bias vector), resulting in the matrix $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & \sqrt{2} & 2 & 2\sqrt{2} \\ 1 & 3\sqrt{3} & 27 & 81\sqrt{3} \\ 1 & 2\sqrt{5} & 20 & 40\sqrt{5} \\ 1 & \sqrt{14} & 14 & 14\sqrt{14} \\ 1 & \sqrt{53} & 53 & 53\sqrt{53} \\ 1 & \sqrt{3} & 3 & 3\sqrt{3} \\ 1 & 2\sqrt{2} & 8 & 16\sqrt{2} \\ 1 & \sqrt{85} & 85 & 85\sqrt{85} \end{bmatrix}$$

Given the following formula for the weight vector:

$$\boldsymbol{w = (\Phi^T \Phi)^{-1} \Phi^T z}$$

The vector $\boldsymbol{w}$**= (4.58352122, -1.6872048, 0.33773733, -0.01330674)** is returned, and, as such, the expression for the regression function is as follows:

$$\boldsymbol{f(x, w) = \sum_{j=0}^{3} w_j \times ||x||_2^j = 4.58352122 - 1.6872048\,||x||_2 + 0.33773733\,||x||_2^2 - 0.01330674\,||x||_2^3}$$

2) Using the function obtained in **1)**, the following outputs ($\boldsymbol{t_i}$) were obtained:

	o_i	$t_i = f(x, w)$
x_9	2	2.4356
x_{10}	4	2.2816

Using the following formula for the root mean squared error (RMSE):

$$\boldsymbol{RMSE(t, o) = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} (t_i - o_i)^2}}$$

We conclude that the RMSE for our testing data is approximately **1.2567**.

TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 6

3) For this ID3 decision tree, the node choice was made based on which variable had the highest information gain (IG). For those computations, the following formulas were used:

$$IG(z \mid y_i) = H(z) - H(z \mid y_i)$$

$$H(z \mid y_i) = \sum_{v \in y_i} P(y_i = v) \times H(z \mid y_i = v)$$

$$H(z) = -\sum_{v \in z} P(z = v) \times log_2\,(P(z = v))$$

$$H(z \mid y_i = k) = -\sum_{v \in z \mid y_i = k} P(z = v \mid y_i = k) \times log_2\,(P(z = v \mid y_i = k))$$

For y_3's equal depth binarization, as the median is $\frac{3+4}{2} = 3.5$, all values under 3.5 were assigned the value 0, and those over 3.5 (never equal because these are integer values) were assigned 1.

The transformed data set and the initial IG values are shown in the following tables ($\mathbf{H(z) = 1}$):

	y_1	y_2	y_3	**Output**
x_1	1	1	0	N
x_2	1	1	1	N
x_3	0	2	1	N
x_4	1	2	0	N
x_5	2	0	1	P
x_6	1	1	0	P
x_7	2	0	0	P
x_8	0	2	1	P

y_i	y_1	y_2	y_3
$H(z \mid y_i = 0)$	1	0	1
$H(z \mid y_i = 1)$	0.8113	0.9183	1
$H(z \mid y_i = 2)$	0	0.9183	N/A
$H(z \mid y_i)$	0.6556	0.6887	1
$IG(z \mid y_i)$	**0.3444**	0.3113	0

Since the variable with higher information gain is $\boldsymbol{y_1}$, that will be the first node in our decision tree.

For each different value of y_1, there is a different outcome:

- If $\boldsymbol{y_1 = 0}$, each output has a probability of $\frac{1}{2}$, so the result is inconclusive. As such, we chose the predicted output in this situation to be P;
- If $\boldsymbol{y_1 = 2}$, all samples return P as output, so the resulting node in the tree is also P;
- If $\boldsymbol{y_1 = 1}$, a new, trimmed data set is given (see the table on the next page, left hand side).

Within this data set, the information gain must be calculated for both variables ($\mathbf{H(z) \approx 0.8113}$):

Aprendizagem 2021/22

Homework I – Group 6

	y_2	y_3	**Output**
x_1	1	0	N
x_2	1	1	N
x_4	2	0	N
x_6	1	0	P

y_i	y_2	y_3
$H(z \mid y_i = 0)$	N/A	0.9183
$H(z \mid y_i = 1)$	0.9183	0
$H(z \mid y_i = 2)$	0	N/A
$H(z \mid y_i)$	0.6887	0.6887
$IG(z \mid y_i)$	**0.1226**	**0.1226**

Since both variables have an equal information gain, $\boldsymbol{y_2}$ was the chosen variable due to its lower index.

Like previously done for y_1, each different value of y_2 returns a different outcome:

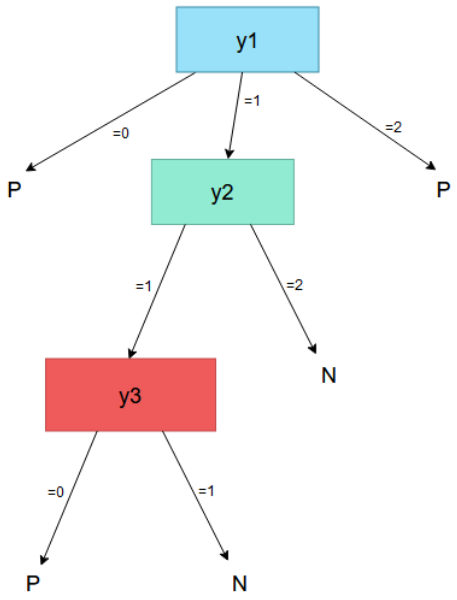
- If $\boldsymbol{y_2 = 2}$, the only sample returns N, so that will be our predicted output;
- If $\boldsymbol{y_2 = 1}$, the new data set is given as follows:

	y_3	Output
$\underline{x_1}$	0	N
x_2	1	N
x_6	0	P

Finally, for y_3:

- If $\boldsymbol{y_3 = 0}$, yet again, each output has a probability of $\frac{1}{2}$, so our chosen predicted output was P;
- If $\boldsymbol{y_3 = 1}$, the only sample returns N, so that will be our predicted output.

The resulting decision tree is thus drawn as follows:



TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 6

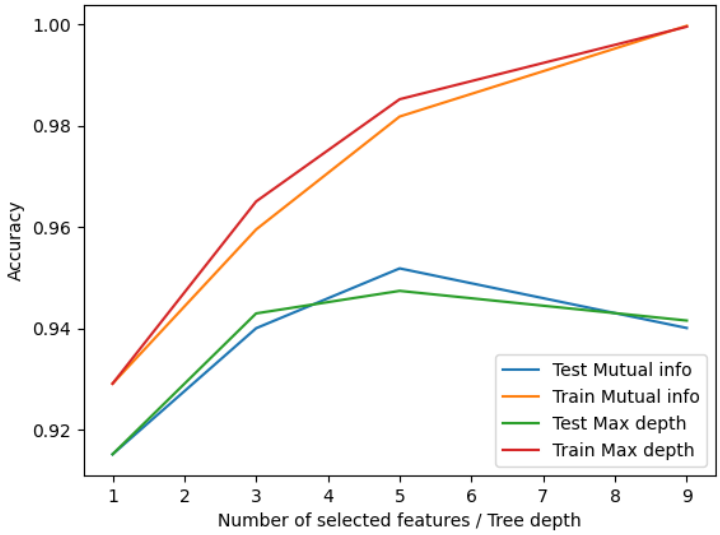
4) Using the previously learned decision tree, the following outputs (t_i) were obtained:

	o_i	t_i
x_9	N	P
x_{10}	P	N

Naturally, the accuracy for this testing set is **0**.

II. Programming and critical analysis

5) Using the code in Appendix, we plotted the following graph (using seed = 6):



6) Using Pearson's correlation coefficient, we verify that there is a very high positive correlation between the accuracy of both methods (over 0.95). This is also confirmed by direct analysis of the previous graph.

This result is expected because in both cases we are increasing the amount of information taken into consideration when training the classifier. In the first method we gradually increase the number of selected features and in the second we increase the maximum tree depth. Both actions will prevent underfitting and able us to get better estimates.

The correlation can be further explained by the fact that, the deeper our tree is, the more variables/features it will use, so it's expected that the accuracies for each tree depth are, in some way, conditioned by the number of variables used.

7) By running the program, specified in the Appendix, multiple times, we infer that the tree depth that maximizes accuracy is usually 5.

We chose 5 because is the most stable value for maximum tree depth that also results in high levels of accuracy. A value of 9 could also produce in better results, but it's more inconsistent and sometimes suffers from overfitting, which leads to a lower accuracy.

III. APPENDIX

```
from sklearn.feature_selection import mutual_info_classif, SelectKBest
from sklearn.model_selection import StratifiedKFold, cross_validate
import matplotlib.pyplot as plt
from scipy.io import arff
from sklearn import tree
import pandas as pd

def loadDataFrame():
    data = arff.loadarff('breast.w.arff')
    df = pd.DataFrame(data[0])
    df['Class'] = df['Class'].str.decode('utf-8')
    return df

def splitFeatureLabel(x):
    x_features, x_label = x.iloc[:,:-1], x.iloc[:,[-1]]
    x_label_01 = [1 if elem == "malignant" else 0 for elem in x_label["Class"]]
    return x_features, x_label, x_label_01

def runDecisionTree(df_features,df_labels,n,cv):
    clf = tree.DecisionTreeClassifier(max_depth=n)
    scores = cross_validate(clf, df_features,df_labels,cv=cv,scoring="accuracy",return_train_score =
True)
    test_score = scores["test_score"].mean()
    train_score = scores["train_score"].mean()
    return test_score,train_score

def runMutualInfoDecisionTree(cv,df_features, df_label):
    test_scores, train_scores = [], []
    for i in [1,3,5,9]:
        sel_cols = SelectKBest(mutual_info_classif,k=i)
        sel_cols.fit(df_features, df_label_01)
        test_score,train_score =
runDecisionTree(df_features[sel_cols.get_feature_names_out()],df_label,i,cv)
        test_scores.append(test_score)
        train_scores.append(train_score)
    return test_scores,train_scores

def runMaxDepthDecisionTree(cv,df_features, df_label):
    test_scores, train_scores = [], []
    for i in [1,3,5,9]:
        test_score,train_score = runDecisionTree(df_features,df_label,i,cv)
        test_scores.append(test_score)
        train_scores.append(train_score)
    return test_scores,train_scores

def plotAndSaveGraph(test_scores_mutual, train_scores_mutual,test_scores_depth,train_scores_depth):
    plt.plot([1,3,5,9],test_scores_mutual,label="Test Mutual info")
    plt.plot([1,3,5,9],train_scores_mutual,label="Train Mutual info")
    plt.plot([1,3,5,9],test_scores_depth,label="Test Max depth")
    plt.plot([1,3,5,9],train_scores_depth,label="Train Max depth")
    plt.xlabel('Number of selected features / Tree depth'); plt.ylabel('Accuracy')
    plt.legend(loc='lower right')
    plt.savefig("plot.png")

if __name__ == "__main__":
    df = loadDataFrame()
    df_features, df_label, df_label_01 = splitFeatureLabel(df)
    cv = StratifiedKFold(n_splits = 10, shuffle = True, random_state = 6)
    test_scores_mutual,train_scores_mutual = runMutualInfoDecisionTree(cv,df_features, df_label)
    test_scores_depth,train_scores_depth = runMaxDepthDecisionTree(cv,df_features, df_label)
    plotAndSaveGraph(test_scores_mutual, train_scores_mutual,test_scores_depth,train_scores_depth)
```

END