TÉCNICO LISBOA

Aprendizagem 2021/22

Homework II – Group 053

I. Pen-and-paper

1) $f(x,w) = \sum_{j=0}^{3} w_j \cdot \varphi_j(x)$, where $\varphi_j(x) = ||x||_2^j$.

$$\varphi_{\text{train}} = \begin{pmatrix} 1 & \varphi_{11}(x) = ||x_1||_2^1 & \cdots & \varphi_{81}(x) = ||x_8||_2^1 \\ 1 & \vdots & \ddots & \vdots \\ 1 & \varphi_{13}(x) = ||x_1||_2^3 & \cdots & \varphi_{83}(x) = ||x_8||_2^3 \end{pmatrix}; ||x_{\text{train}}||_2 = \begin{pmatrix} 1.4142135623730951 \\ 5.196152422706632 \\ 4.47213595499958 \\ 3.7416573867739413 \\ 7.280109889280518 \\ 1.7320508075688772 \\ 2.8284271247461903 \\ 9.219544457292887 \end{pmatrix}$$

Given that $W = {\varphi_{\text{train}}}^{\dagger} z_{\text{train}}$, where $\varphi^{\dagger} = ({\varphi_{\text{train}}}^{T} \cdot \varphi_{\text{train}})^{-1} \cdot {\varphi_{\text{train}}}^{T}$ is the pseudo-inverse of φ_{train}, this gives us that:

$$W = {\varphi_{\text{train}}}^{\dagger} \cdot \begin{pmatrix} 1 \\ 3 \\ 2 \\ 0 \\ 6 \\ 4 \\ 5 \\ 7 \end{pmatrix} = \begin{pmatrix} 4.58352122 \\ -1.6872048 \\ 0.33773733 \\ -0.01330674 \end{pmatrix}.$$

Note that we only consider the training set in this regression model learning process.

2) Considering the testing set, we have that:

$$\varphi_{\text{test}} = \begin{pmatrix} 1 & 2 & 4 & 8 \\ 1 & 2.44948974 & 6 & 14.69693846 \end{pmatrix} \quad ; \quad \hat{z}_{\text{test}} = \varphi_{\text{test}} \cdot W = \begin{pmatrix} 2.45360697 \\ 2.28158593 \end{pmatrix}$$

$$z_{\text{test}} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$

From here we can calculate the *Root Mean Square Error (RMSE)* by:

$$\text{RMSE}(\hat{z}_{\text{test}}, z_{\text{test}}) = \sqrt{\frac{1}{2}\sum_{i=1}^{2} (\hat{z}_{\text{test}_i} - z_{\text{test}_i})^2} = 1.2567231583983312.$$

This is a high value considering a dataset of size two.

3) The variable y_3 has a substantial range of values with low frequency each. Therefore, to prevent overfitting from using these statistically insignificant values, we resort to variable discretization. In order to obtain an equal depth binarization of y_3, we choose 4 as the splitting point. The resulting training vector is [0 1 1 0 1 0 0 1].

Because we're working with ID3 algorithm, for each node we select the variable that grants highest Information Gain (IG).

$IG(X \mid y_j) = I(X) - I(X \mid y_j)$, X is a partition, y_j is a variable and I is the information entropy where $I(X \mid y_j)$ is the conditional entropy of X given the value of the y_j feature.

TÉCNICO LISBOA

Aprendizagem 2021/22

Homework II – Group 053

For the root:

$IG(\mathrm{X} \mid y_1) = I(X) - I(\mathrm{X} \mid y_1) = 1 - 0.65564 = 0.34436$

$IG(\mathrm{X} \mid y_2) = I(X) - I(\mathrm{X} \mid y_2) = 1 - 0.688722 = 0.311278$

$IG(\mathrm{X} \mid y_3) = I(X) - I(\mathrm{X} \mid y_3) = 1 - 1 = 0$

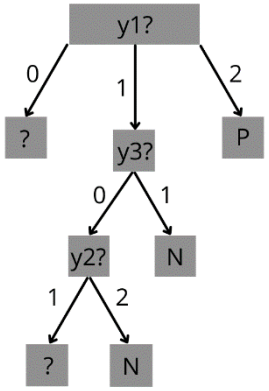
For the branch "y_1=1":

$IG(\mathrm{X} \mid y_1) = I(\{x_1, x_2, x_4, x_6\}) - I(\mathrm{X} \mid y_1) = 0$

$IG(\mathrm{X} \mid y_2) = I(\{x_1, x_2, x_4, x_6\}) - I(\mathrm{X} \mid y_2) = 0.811278 - 0.68872 = 0.12256$

$IG(\mathrm{X} \mid y_3) = I(\{x_1, x_2, x_4, x_6\}) - I(\mathrm{X} \mid y_3) = 0.811278 - 0.68872 = 0.12256$

To standardize, since for y2 and y3 the IG is the same, we select the one with smaller index (i.e., y2).



Branches <y1=0> and <y1=1, y3=0, y2=1> are undecidable because there are ambiguous values in the dataset (same input variables but different outcome). This is a consequence of the small dimension of the dataset as well as the exponential growth of decision tree.

4) Expected for x9: P — Actual: N (output$_9$ = 2, therefore t_9 = N)

Expected for x10: N — Actual: P (output$_{10}$ = 4, therefore t_{10} = P)

So, $Accuracy = \frac{0}{2} = 0$.

The poor performance of the model is a consequence of the small data set as predicted.

Y1 has 3 possible values, y2 has 3 as well and y3 has 2 possible values (post discretization) which results in 18 possible combinations, while we only have 8 training x's.

This inevitably results in overfitting of the model. One possible solution to prevent this would be to aggregate values of the variables in larger bins to reduce the impact of the dimensionality problem (e.g., if we proceeded to binarization of both y1 and y2 the total possible combinations would be reduced to 8).

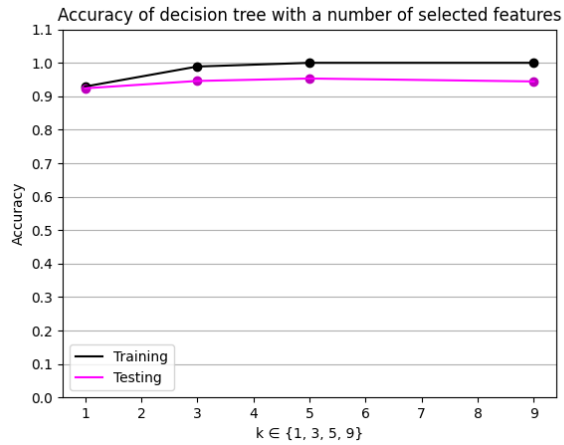
TÉCNICO LISBOA

Aprendizagem 2021/22

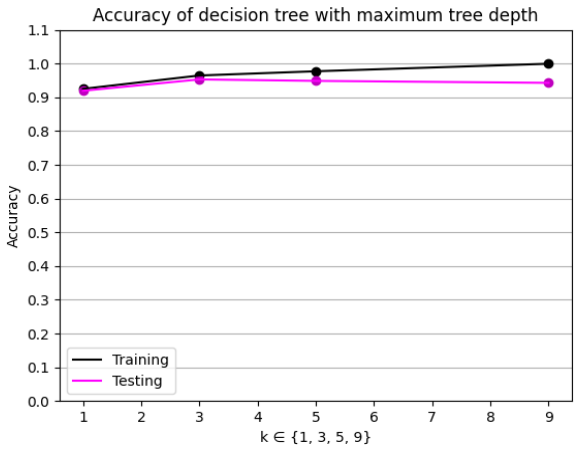
Homework II – Group 053

II. Programming and critical analysis

5) i.



ii.



i. $Pearson\ correlation\ =\ 0.9611$ $\quad p-value\ for\ testing\ non-correlation\ =\ 0.0389$

ii. $Pearson\ correlation\ =\ 0.6481$ $\quad p-value\ for\ testing\ non-correlation\ =\ 0.3519$

From the data and the plot, we can see the evident correlation between training and testing for selected features. Its p-value means we accept the correlation for any level of significance above 3.9%.

For maximum tree depth we cannot claim there is a correlation between training and testing results, based on the correlation coefficient and the high p-value for non-testing.

6) When selecting features from the dataset we obtain progressively better results from one to three to five features and slightly worse ones for all the nine. On the other hand, for the training dataset the results just get better and better. This means that from 5 to 9 features arises an overfitting problem, which may be a consequence of either statistical insignificance of the variables selected or dimensionality being too big for the size of the dataset. The model fails to generalize properly.

For the maximum tree depth there is not a significant correlation between training and testing data sets' results. Also, the training results only get to approximately 1 at depth nine (increasing almost linearly), while the maximum accuracy for testing occurs at depth three. Therefore, the increasing tree depth from three onward may come at an overfitting cost, and below depth three we lose predictive accuracy.

7) Based on the average accuracy and the variance being below 0.001 for our testing results, we can conclude that the most efficient strategy is setting a maximum depth of three since after that we obtain worse results, and it also takes more time to build the tree.

However, this is a supervised learning model, and we wish to diagnose as many malignant cases as possible (high sensitivity expected). Thus, we might want to choose the model based on the sensitivity results.

Sensitivity for maximum depth in testing:

[0.94811711468827, 0.9230943987735001, 0.9145956876027694, 0.8946858508544769]

This indicates that the highest recall rate is obtained for a depth of one, and that should therefore be the one chosen if we are not very worried about false positives.

TÉCNICO LISBOA

Aprendizagem 2021/22

Homework II – Group 053

III. APPENDIX

```
# 2.5.i & 2.5.ii
values_range = [1, 3, 5, 9]
def model_score(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    results = model.predict(X_test)
    return accuracy_score(results, y_test)
def plot_graph(acc_vectors, plot_title, file_name):
    for i in range(len(values_range)):
        plt.plot(values_range[i], acc_vectors[0][i], 'ko')
        plt.plot(values_range[i], acc_vectors[1][i], 'mo')
    plt.plot(values_range, acc_vectors[0], label="Training", color="black")
    plt.plot(values_range, acc_vectors[1], label="Testing", color="magenta")
    plt.xlabel("k ∈ {1, 3, 5, 9}")
    plt.ylabel("Accuracy")
    plt.grid(axis = "y")
    plt.yticks(np.arange(0, 1.2, 0.1))
    plt.title(f"Accuracy of decision tree with {plot_title}")
    plt.legend()
    plt.savefig(f"acc_{file_name}.png")

data = arff.loadarff('breast.w.arff')
df = pd.DataFrame(data[0])
df['Class'] = df['Class'].str.decode('utf8') # remove weird string
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
scores_training_mi, scores_testing_mi = {}, {} # mutual information
scores_training_md, scores_testing_md = {}, {} # max depth
for i in range(len(values_range)):
    j = values_range[i]
    scores_training_mi[j], scores_testing_mi[j], scores_training_md[j], scores_testing_md[j] = [],
[], [], []
k_fold = KFold(n_splits=10, shuffle=True, random_state=76)
for train_index, test_index in k_fold.split(X):
    for i in range(len(values_range)):
        j = values_range[i]
        selector = SelectKBest(mutual_info_classif, k=j)
        X_new = selector.fit_transform(X, y)
        X_train , X_test = X_new[train_index,:], X_new[test_index,:]
        y_train , y_test = y[train_index].values, y[test_index].values
        scores_training_mi[j] += [model_score(DecisionTreeClassifier(criterion="entropy"), X_train,
X_train, y_train, y_train)]
        scores_testing_mi[j] += [model_score(DecisionTreeClassifier(criterion="entropy"), X_train,
X_test, y_train, y_test)]
```

```
        scores_training_md[j] += [model_score(DecisionTreeClassifier(criterion="entropy",
max_depth=j), X_train, X_train, y_train, y_train)]
        scores_testing_md[j] += [model_score(DecisionTreeClassifier(criterion="entropy",
max_depth=j), X_train, X_test, y_train, y_test)]
acc_train_mi, acc_test_mi, acc_train_md, acc_test_md = [], [], [], []
for i in range(len(values_range)):
    j = values_range[i]
    acc_train_mi += [sum(scores_training_mi[j])/len(scores_training_mi[j])]
    acc_test_mi += [sum(scores_testing_mi[j])/len(scores_testing_mi[j])]
    acc_train_md += [sum(scores_training_md[j])/len(scores_training_md[j])]
    acc_test_md += [sum(scores_testing_md[j])/len(scores_testing_md[j])]

#2.6
mi = [acc_train_mi, acc_test_mi]
md = [acc_train_md, acc_test_md]
plot_graph(mi, "a number of selected features", "mutualinformation")
plt.clf()
plot_graph(md, "maximum tree depth", "maxdepth")
```

END