TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 35

I. Pen-and-paper

1) Aplicando a transformação dada pela basis function $\phi_j(x) = \|x\|_2^{\,j}$ e tendo em consideração que $\|x\|_2 = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}}$ obtemos a seguinte matriz X:

$$X = \phi = \begin{bmatrix} 1 & \|x_1\| & \|x_1\|^2 & \|x_1\|^3 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 1 & \|x_8\| & \|x_8\|^2 & \|x_8\|^3 \end{bmatrix} = \begin{bmatrix} 1 & 1.41421 & 2 & 2.82843 \\ 1 & 5.19615 & 27 & 140.29612 \\ 1 & 4.47214 & 20 & 89.44272 \\ 1 & 3.47166 & 14 & 52.38320 \\ 1 & 7.28011 & 53 & 385.84582 \\ 1 & 1.732.5 & 3 & 5.19615 \\ 1 & 2.82843 & 8 & 22.62742 \\ 1 & 9.21954 & 85 & 783.66128 \end{bmatrix}$$

Agora, sabendo que $Z = [1\ 3\ 2\ 0\ 6\ 4\ 5\ 7]^T$ o objetivo será encontrar o w que minimiza o erro quadrático, ou seja, o w tal que $\nabla_w E = 0$ onde $\nabla_w E = -2X^T(Z - X \cdot w)$

$$\nabla_w E = 0 \Leftrightarrow -2X^T(Z - X \cdot w) = 0 \Leftrightarrow X^T(Z - X \cdot w) = 0 \Leftrightarrow X^T \cdot Z = X^T \cdot X \cdot w$$
$$\Leftrightarrow w = (X^T \cdot X)^{-1} \cdot X^T \cdot Z$$

Assim, recorrendo às operações sobre matrizes da biblioteca *numpy* do Python (ver Appendix) obtemos:

$$w = (X^T \cdot X)^{-1} \cdot X^T \cdot Z = [\,4.58346 \quad -1.68716 \quad 0.33773 \quad -0.01331]^T$$

Pelo que:

$$\hat{z}(x) = f(x, w) = \sum_{j=0}^{3} w_j \cdot \phi_j(x) = 4.58346 + (-1.68716)\phi_1(x) + 0.33773\phi_2(x) + (-0.01331)\phi_3(x)$$

2) $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(output(x_i) - \hat{z}(x_i))^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(output(x_i) - f(x_i, w))^2}$

$$f(x_9, w) = 4.5834595 - 1.68715627 \cdot 2 + 0.33772727 \cdot 2^2 - 0.01330615 \cdot 2^3 = 2.45360687$$

$$f(x_{10}, w) = 4.5834595 - 1.68715627 \cdot \sqrt{6} + 0.33772727 \cdot 6 - 0.01330615 \cdot 6^{\frac{3}{2}} = 2.28159107$$

$$\therefore RMSE = \sqrt{\frac{1}{2}\sum_{i=9}^{10}(output(x_i) - f(x_i, w))^2} = \sqrt{\frac{1}{2}((2 - f(x_9, w))^2 + (4 - f(x_{10}, w))^2)} = 1.256720$$

3) Considerando uma *equal depth binarization* de y_3, com a mediana 3.5 após a sua ordenação crescente, obtemos o seguinte vetor de $y_3 = [0\ 1\ 1\ 0\ 1\ 0\ 0\ 1]^T$ onde 1 são os valores superiores à mediana e 0 os inferiores, do vetor original de y_3. Obtemos também o vetor $output = [N\ N\ N\ N\ P\ P\ P\ P]^T$ realizando a transformação apresentada no enunciado. Temos $y_1 \in \{0, 1, 2\}$ e $y_2 \in \{0, 1, 2\}$.

Começamos, então, por calcular os seguintes Information Gains para decidir qual característica/atributo colocar na raiz da árvore:

$IG(Z \mid y_1) = H(Z) - H(Z \mid y_1)$ $\quad IG(Z \mid y_2) = H(Z) - H(Z \mid y_2)$ $\quad IG(Z \mid y_3) = H(Z) - H(Z \mid y_3)$

- $H(Z) = -\sum_i (P(Z=i)\ \log_2 P(Z=i)) = -\left(\frac{1}{2}\cdot\log_2\frac{1}{2} + \frac{1}{2}\cdot\log_2\frac{1}{2}\right) = 1$
- $H(Z \mid y_1) = \frac{2}{8}\cdot H(Z \mid y_1 = 0) + \frac{4}{8}\cdot H(Z \mid y_1 = 1) + \frac{2}{8}\cdot H(Z \mid y_1 = 2)$

  $H(Z \mid y_1 = 0) = -\sum_i (P(Z=i \mid y_1=0)\ \log_2 P(Z=i \mid y_1=0)) =$

  $= -\left(\frac{1}{2}\cdot\log_2\frac{1}{2} + \frac{1}{2}\cdot\log_2\frac{1}{2}\right) = 1$

  $H(Z \mid y_1 = 1) = -\left(\frac{1}{4}\cdot\log_2\frac{1}{4} + \frac{3}{4}\cdot\log_2\frac{3}{4}\right) = 0.81128$

  $H(Z \mid y_1 = 2) = -(1\cdot\log_2 1) = 0$

  $\therefore\ H(Z \mid y_1) = \frac{2}{8}\cdot 1 + \frac{4}{8}\cdot 0.81128 + \frac{2}{8}\cdot 0 = 0.6556$

- $H(Z \mid y_2) = \frac{2}{8}\cdot H(Z \mid y_2 = 0) + \frac{3}{8}\cdot H(Z \mid y_2 = 1) + \frac{3}{8}\cdot H(Z \mid y_2 = 2)$

  $H(Z \mid y_2 = 0) = -(1\cdot\log_2 1) = 0$

  $H(Z \mid y_2 = 1) = -\sum_i (P(Z=i \mid y_2=0)\ \log_2 P(Z=i \mid y_2=0)) =$

  $= -\left(\frac{2}{3}\cdot\log_2\frac{2}{3} + \frac{1}{3}\cdot\log_2\frac{1}{3}\right) = 0.91830$

  $H(Z \mid y_2 = 2) = -\left(\frac{1}{3}\cdot\log_2\frac{1}{3} + \frac{2}{3}\cdot\log_2\frac{2}{3}\right) = 0.91830$

  $\therefore\ H(Z \mid y_2) = \frac{3}{8}\cdot 0.91830 + \frac{3}{8}\cdot 0.91830 = \frac{3}{4}\cdot 0.91830 = 0.6887$

- $H(Z \mid y_3) = \frac{1}{2}\cdot H(Z \mid y_3 = 0) + \frac{1}{2}\cdot H(Z \mid y_3 = 1)$

  $H(Z \mid y_3 = 0) = H(Z \mid y_3 = 1) = -\left(\frac{1}{2}\cdot\log_2\frac{1}{2} + \frac{1}{2}\cdot\log_2\frac{1}{2}\right) = 1$

  $\therefore\ H(Z \mid y_3) = \frac{1}{2}\cdot 1 + \frac{1}{2}\cdot 1 = 1$

Temos então:

$IG(Z \mid y_1) = H(Z) - H(Z \mid y_1) = 1 - 0.6556 = 0.3444$

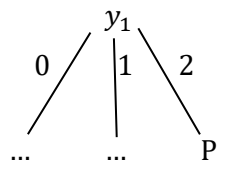
$IG(Z \mid y_2) = H(Z) - H(Z \mid y_2) = 1 - 0.6887 = 0.3113$

$IG(Z \mid y_3) = H(Z) - H(Z \mid y_3) = 1 - 1 = 0$

Deste modo, concluímos que a raiz da árvore será y_1 dado que tem maior Information Gain.

Quando y_1 é 2, o *output* é sempre P. Assim prosseguimos aos cálculos com o seguinte Data Set e troço da árvore:

x_i	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_6	1	1	0	P
x_8	0	2	1	P



- Tendo em conta apenas os x_i com $y_1 = 0$ (linhas de cor cinzenta):

  $H(Z) = -\sum_i (P(Z=i)\ \log_2 P(Z=i)) = -\left(\frac{1}{2}\cdot\log_2\frac{1}{2} + \frac{1}{2}\cdot\log_2\frac{1}{2}\right) = 1$

  $IG(Z \mid y_2) = H(Z) - H(Z \mid y_2) = 1 - 1 = 0 = IG(Z \mid y_3)$

- Tendo em conta apenas os x_i com $y_1 = 1$ (linhas de cor branca):

  $H(Z) = -\sum_i(P(Z=i)\ \log_2 P(Z=i)\ ) = -\left(\frac{3}{4}\cdot\log_2\frac{3}{4} + \frac{1}{4}\cdot\log_2\frac{1}{4}\right) = 0.81128$

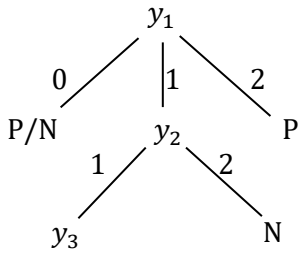
  $IG(\ Z \mid y_2\ ) = H(\ Z\ ) - H(\ Z \mid y_2\ ) = 0.81128 - 0.68872 = 0.12256$

  $IG(\ Z \mid y_3\ ) = H(\ Z\ ) - H(\ Z \mid y_3\ ) = 0.81128 - 0.68872 = 0.12256$

No primeiro caso, obtemos um Information Gain igual a 0 para ambos os casos, o que não nos permite selecionar um comportamento determinista em relação ao resultado e, por isso, tanto pode ser P como N. Escolhemos y_2 para raiz da subárvore quando $y_1 = 1$ pois, havendo um empate no Information Gain, optamos pelo índice menor. Quando $y_2 = 2 \mid y_1 = 1$ obtemos sempre o *output* N, ficando a faltar y_3 no ramo $y_2 = 1 \mid y_1 = 1$.

Novo Data Set:

x_i	y_1	y_2	y_3	output
x_1	1	1	0	N
x_2	1	1	1	N
x_6	1	1	0	P



- Tendo agora em conta este Novo Data Set temos (quando $y_2 = 1 \mid y_1 = 1$):

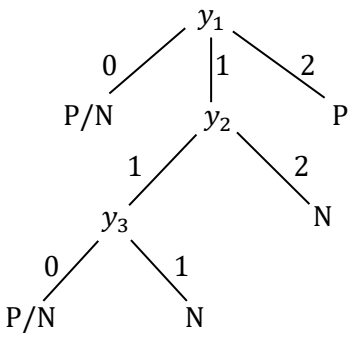
  $H(Z) = -\sum_i(P(Z=i)\ \log_2 P(Z=i)\ ) = -\left(\frac{2}{3}\cdot\log_2\frac{2}{3} + \frac{1}{3}\cdot\log_2\frac{1}{3}\right) = 0.91830$

  $H(\ Z \mid y_3 = 0\ ) = -\left(\frac{1}{2}\cdot\log_2\frac{1}{2} + \frac{1}{2}\cdot\log_2\frac{1}{2}\right) = 1$

  $H(\ Z \mid y_3 = 1\ ) = -1\cdot\log_2 1 = 0$

Tendo as entropias e observando a tabela, verificamos que quando $y_3 = 0$ existem 1 caso N e um caso P pelo que a entropia é 1, ou seja, não conseguimos definir um output. Sempre que $y_3 = 1$ o output é N, logo a entropia é 0.

<u>Árvore Final:</u>



4) Segundo a nossa árvore temos:

Para x_9:

. $z_9 = output(x_9) = N$

. $\widehat{z_9} = P$

. Logo, temos um False Positive, FP.

Para x_{10}:

. $z_{10} = output(x_{10}) = P$

. $\widehat{z_{10}} = N$

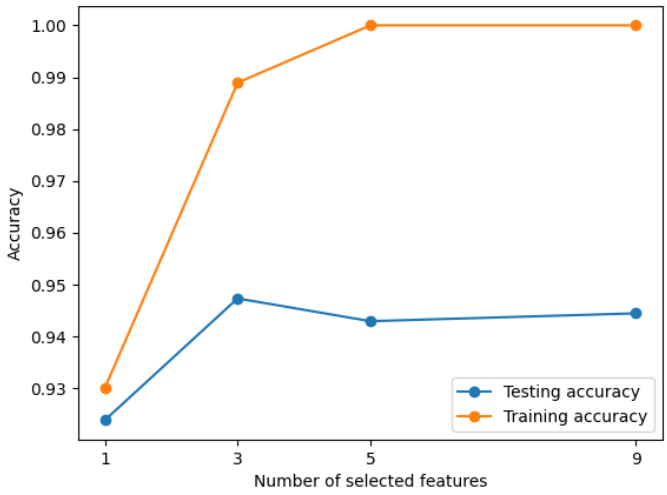
. Logo, temos um False Negative, FN.

Assim temos:

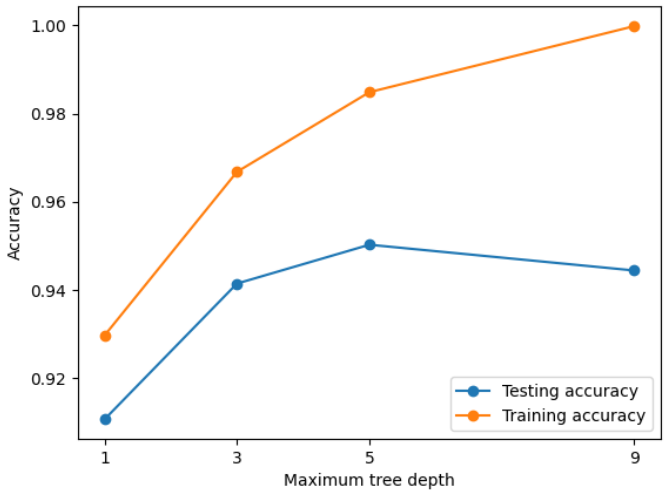
$$\therefore Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{0}{2} = 0$$

II. Programming and critical analysis

5) i)



ii)



6) Tendo em conta os gráficos acima apresentados (resultantes do código no Appendix), observamos que a accuracy aumenta para a Training Data quer com o aumento do número de features selecionadas quer com o aumento da profundidade da árvore. Tal como seria de esperar, a árvore terá maior profundidade com mais features e avaliará melhor os casos de treino dado que estará melhor "treinada" para tais casos.

Em contraste com o aumento da accuracy na Training Data, temos uma oscilação da accuracy na Testing Data. Verificamos que existe uma subida até um determinado número de features e até uma determinada profundidade, porém, a partir de tal, a accuracy tende a diminuir.

Deste modo, conseguimos concluir que um número pequeno de features e de profundidade gera resultados com menos accuracy devido à existência de um eventual *underfitting*, ou seja, o modelo não está totalmente bem adaptado aos dados quer de treino quer de teste por não ser complexo o suficiente. Para um número grande de features e profundidade, a diminuição da accuracy na Testing Data e aumento da accuracy na Training Data justifica-se devido ao *overfitting*, ou seja, o modelo adaptou-se tão bem aos dados de treino que se torna ineficaz perante novos dados/testes.

7) A partir da análise do gráfico, concluímos que a profundidade máxima que traz melhores resultados é 5 pois esta apresenta uma accuracy bastante alta para os casos de treino e, ao mesmo tempo, maximiza a accuracy para os casos de teste. Deste modo, o modelo não só está bem adaptado à Training Data (accuracy de 98.5%) como também é capaz de prever os casos de teste com uma accuracy de 95%.

III. APPENDIX

```
# IMPORTS are omitted

# Conditions
GROUP_NUMBER = 35
k = [1,3,5,9]

data = arff.loadarff('breast.w.arff')
df = pd.DataFrame(data[0])
df = df.dropna()
data = df.drop(columns=["Class"]).values
results = df[df.keys()[-1]].astype('string').values

accuracies_test_f, accuracies_train_f, accuracies_test_d, accuracies_train_d = [], [], [], []

kfold = KFold(n_splits=10, shuffle=True, random_state=GROUP_NUMBER)

for i in k:
    clf1 = DecisionTreeClassifier(max_features=i, random_state=GROUP_NUMBER) # 5i.
    clf2 = DecisionTreeClassifier(max_depth=i, random_state=GROUP_NUMBER) # 5ii.

    new_data = SelectKBest(score_func=mutual_info_classif, k=i).fit_transform(data, results)
    scores_f = cross_validate(clf1, new_data, results, scoring="accuracy", return_train_score=True,
cv=kfold)
    scores_d = cross_validate(clf2, data, results, scoring="accuracy", return_train_score=True, cv=kfold)

    new_acc_test_f, new_acc_train_f = mean(scores_f['test_score']), mean(scores_f['train_score'])
    new_acc_test_d, new_acc_train_d = mean(scores_d['test_score']), mean(scores_d['train_score'])
    accuracies_test_f.append(new_acc_test_f) ; accuracies_train_f.append(new_acc_train_f)
    accuracies_test_d.append(new_acc_test_d) ; accuracies_train_d.append(new_acc_train_d)

    print("Accuracy: Test-{} # Train-{} for {} features\nAccuracy: Test-{} # Train-{} for {} max depth\n"
        .format(new_acc_test_f, new_acc_train_f, i, new_acc_test_d, new_acc_train_d, i))

# Plot and Save Selected Features plot
test = plt.plot(k, accuracies_test_f, marker='o') ; train = plt.plot(k, accuracies_train_f, marker='o')
plt.xticks(k)
plt.ylabel("Accuracy") ; plt.xlabel("Number of selected features")
plt.legend(["Testing accuracy", "Training accuracy"], loc='lower right')
plt.savefig("graph_f.png")
# Plot and Save Maximum Depth plot
test.pop(0).remove() ; train.pop(0).remove() ; plt.gca().set_prop_cycle(None) # delete old lines
test = plt.plot(k, accuracies_test_d, marker='o') ; train = plt.plot(k, accuracies_train_d, marker='o')
plt.xlabel("Maximum tree depth")
plt.savefig("graph_d.png")
```

Código auxiliar para a Parte I:
https://github.com/martimfasantos/Apre/blob/main/homework2/pol_regression.py

END