Aprendizagem 2021/22

Homework I – Group 024

I. Pen-and-paper

1) Do enunciado obtemos os seguintes valores de treino:

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

Para aprender o modelo de regressão polinomial, é necessário calcular:

$$f(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{3} w_j \cdot \phi_j(\mathbf{x}) = \Phi \cdot \mathbf{w} \qquad (1)$$

Onde a matriz Φ, obtida através da função $\phi_j(\mathbf{x}) = \|\mathbf{x}\|_2{}^j$, é dada por:

$$\Phi = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \cdots & \phi_3(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_8) & \cdots & \phi_3(\mathbf{x}_8) \end{bmatrix} = \begin{bmatrix} \|\mathbf{x}_1\|_2{}^0 & \cdots & \|\mathbf{x}_1\|_2{}^3 \\ \vdots & \ddots & \vdots \\ \|\mathbf{x}_8\|_2{}^0 & \cdots & \|\mathbf{x}_8\|_2{}^3 \end{bmatrix}$$

Concretizando:

$$\Phi = \begin{bmatrix} 1 & 1.4142 & 2.0 & 2.82843 \\ 1 & 5.1961 & 27 & 140.296 \\ 1 & 4.4721 & 20 & 89.4427 \\ 1 & 3.7417 & 14 & 52.3832 \\ 1 & 7.2801 & 53 & 385.846 \\ 1 & 1.1731 & 3.0 & 5.19615 \\ 1 & 2.8284 & 8.0 & 22.6274 \\ 1 & 9.2195 & 85 & 783.661 \end{bmatrix}$$

Por sua vez, o vetor de pesos ($\mathbf{w}$) para regressões polinomiais é obtido através da expressão:

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

Assim,

$$\mathbf{w} = \begin{bmatrix} 4.5835 \\ -1.687 \\ 0.3377 \\ -0.013 \end{bmatrix}$$

Logo, pela equação (1):

$$f(\mathbf{x}, \mathbf{w}) = \begin{bmatrix} 2.8353 \\ 3.0686 \\ 2.6027 \\ 2.3019 \\ 5.0662 \\ 2.6053 \\ 2.2122 \\ 7.3080 \end{bmatrix}$$

Aprendizagem 2021/22

Homework I – Group 024

2) Os valores de teste são:

$$X = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \qquad \mathbf{z} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

Pelo mesmo raciocínio do exercício anterior, calcula-se a matriz Φ:

$$\Phi = \begin{bmatrix} 1 & 2.00 & 4 & 8.00 \\ 1 & 2.45 & 6 & 14.7 \end{bmatrix}$$

No entanto, utiliza-se o vetor **w** anterior porque pretendemos testar o erro do modelo obtido anteriormente. Pela equação (1), obtém-se:

$$f(\mathbf{x}, \mathbf{w}) = \begin{bmatrix} 2.454 \\ 2.282 \end{bmatrix} = \hat{\mathbf{z}}$$

Comparando o valor real ($\mathbf{z}$) com o previsto ($\hat{\mathbf{z}}$), o *Root Mean Square Error* (RMSE) é calculado ($N = 2$):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(z_i - \hat{z}_i)^2}{N}} \approx 1.2567$$

3) Para tornar y_3 numa variável binária e com profundidades iguais entre valores, foi escolhida a divisão:

$$y_{3_i} = \begin{cases} A, & y_{3_i}(antigo) < 4 \\ B, & y_{3_i}(antigo) \geq 4 \end{cases}$$

Assim,

$$\mathbf{y}_3 = [A, B, B, A, B, A, A, B]^T$$

Pelo enunciado,

$$\mathbf{t} = [N, N, N, N, P, P, P, P]^T$$

Aplicando o algoritmo *ID3*, começamos por calcular os ganhos de informação (*IG*) para todas as variáveis. A que tiver maior *IG* será escolhida para ser nó na árvore de decisão. Para $\mathbf{y}_3$, pela fórmula:

$$IG(\mathbf{t}, \mathbf{y}_3) = H(\mathbf{t}) - H(\mathbf{t}|\mathbf{y}_3) = \sum_{x \in X} -p(x) log_2\left(p(x)\right) - \sum_{t \in T} p(t) H(t)$$

Sendo X o subconjunto de **t** em que todo o x é igual, e $p(x)$ a proporção desse subconjunto em relação a **t**. Neste caso, **t** divide-se em 2 subconjuntos: A e B, logo:

$$H(\mathbf{t}) = -\frac{4}{8} log_2\left(\frac{4}{8}\right) - \frac{4}{8} log_2\left(\frac{4}{8}\right) = 1$$

T é o subconjunto de $\mathbf{y}_3$ em que todo o t é igual, e $p(t)$ a proporção desse subconjunto em relação a **t**. Assim,

$$IG(\mathbf{t}, \mathbf{y}_3) = 1 - \left(\frac{4}{8} H(A) + \frac{4}{8} H(B)\right)$$

Onde:

$$H(A) = -\frac{2}{4} log_2\left(\frac{2}{4}\right) - \frac{2}{4} log_2\left(\frac{2}{4}\right) = 1, \quad H(B) = H(A) = 1$$

Logo:

$$IG(\mathbf{t}, \mathbf{y}_3) = 1 - 1 = 0$$

Seguindo o mesmo raciocínio, obteve-se:

$$IG(\mathbf{t},\mathbf{y}_2) = 1 - \left[\frac{2}{8}\left(-1.\,log_2(1)\right) + \frac{3}{8}\left(-\frac{2}{3}log_2\left(\frac{2}{3}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right)\right) + \frac{3}{8}\left(-\frac{2}{3}log_2\left(\frac{2}{3}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right)\right)\right] \approx 0.31$$

$$IG(\mathbf{t},\mathbf{y}_1) = 1 - \left[\frac{2}{8}\left(-\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right)\right) + \frac{4}{8}\left(-\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right)\right) + \frac{2}{8}\left(-1.\,log_2(1)\right)\right] \approx 0.33$$

Conclui-se que $\mathbf{y}_1$ tem o maior ganho de informação, logo é escolhido para base da árvore. Observa-se também que a entropia quando $\mathbf{y}_1 = 2$ é nula, logo para este ramo da árvore infere-se que terá sempre *output* = *P*.

Para o ramo $\mathbf{y}_1 = 1$, utilizando as fórmulas anteriores:

$$H(\mathbf{y}_1 = 1) = -\frac{3}{4}log_2\left(\frac{3}{4}\right) - \frac{1}{4}log_2\left(\frac{1}{4}\right) = 0.8113$$

$$IG(\mathbf{y}_1 = 1, \mathbf{y}_2) = 0.8113 - \left[\frac{3}{4}\left(-\frac{2}{3}log_2\left(\frac{2}{3}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right)\right) + \frac{1}{4}\left(-1.\,log_2(1)\right)\right] \approx 0.123$$

$$IG(\mathbf{y}_1 = 1, \mathbf{y}_3) = 0.8113 - \left[\frac{3}{4}\left(-\frac{2}{3}log_2\left(\frac{2}{3}\right) - \frac{1}{3}log_2\left(\frac{1}{3}\right)\right) + \frac{1}{4}\left(-1.\,log_2(1)\right)\right] \approx 0.123$$

Como o ganho de informação é igual em $\mathbf{y}_2$ e $\mathbf{y}_3$, dado que o *ID3* é um algoritmo *greedy*, é escolhido o primeiro valor encontrado, que é $\mathbf{y}_2$. Como a entropia de $\mathbf{y}_2 = 2$ é nula, este ramo tem *output* = *N*. Para $\mathbf{y}_2 = 0$ não existem valores, logo o resultado é inconclusivo (*?*). Para $\mathbf{y}_2 = 1$, observa-se a variável $\mathbf{y}_3$ (não faz sentido calcular IG com apenas uma variável), e conclui-se que: quando $\mathbf{y}_3 = A$, o resultado é inconclusivo; quando $\mathbf{y}_3 = B$, *output* = *N*.

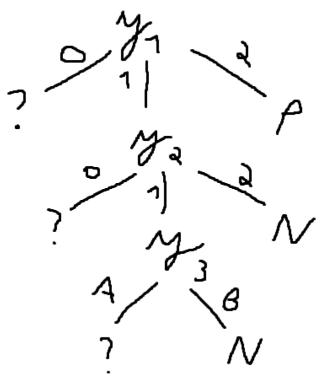
Para o ramo $\mathbf{y}_1 = 0$:

$$H(\mathbf{y}_1 = 0) = -\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right) = 1$$

$$IG(\mathbf{y}_1 = 0, \mathbf{y}_2) = 1 - \left[1.\left(-\frac{1}{2}log_2\left(\frac{1}{2}\right) - \frac{1}{2}log_2\left(\frac{1}{2}\right)\right)\right] = 0 = IG(\mathbf{y}_1 = 0, \mathbf{y}_3)$$

Com ganhos de informação nulos, nada se conclui para este ramo (resultados inconclusivos).

Desenho da árvore:

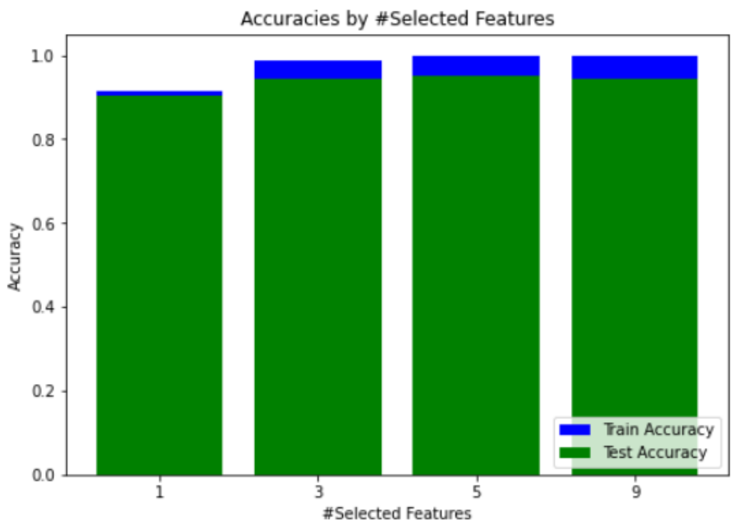


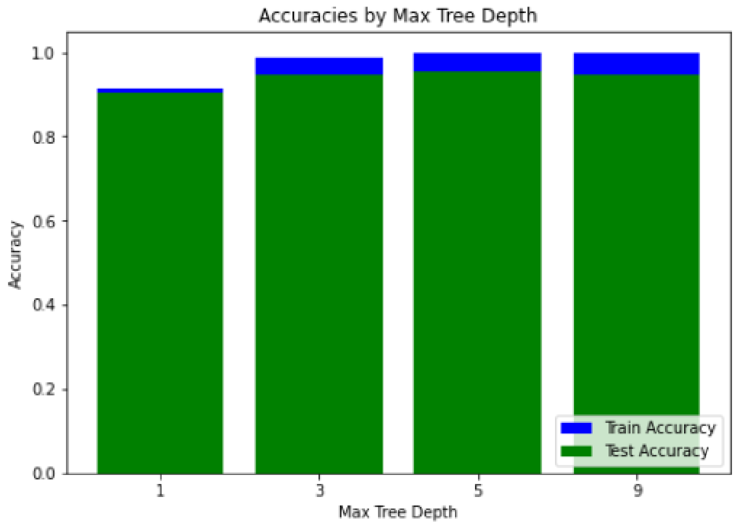
4) *Accuracy* = 0% (*1 False Positive; 1 False Negative*).

Aprendizagem 2021/22
Homework I – Group 024

II. Programming and critical analysis

5)





6) Como existem variáveis cujo ganho de informação é menor, ou seja, que são menos importantes para aprender a árvore, observa-se que a partir de 5 variáveis consideradas a *accuracy* se mantém constante.

Observa-se também um aumento da *accuracy* de treino proporcionalmente à profundidade máxima da árvore, uma vez que esta se ajusta mais ao conjunto de treino, ou seja, a árvore considera mais opções em vez de as generalizar.

7) A profundidade máxima = 5 é a escolhida, pois tem a maior *accuracy* de teste, sendo a diferença entre as *accuracies* de treino e de teste semelhante à de profundidade máxima = 3. Continuando o raciocínio do exercício anterior, quando a árvore considera demasiadas opções, existe uma maior probabilidade de *overfitting*, o que acontece para a profundidade máxima = 9, pois a *accuracy* de teste diminui e a de treino aumenta, quando comparadas com as de profundidade máxima = 5.

III. APPENDIX

Aprendizagem 2021/22

Homework I – Group 024

```
from scipy.io import arff
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_validate
from sklearn.tree import DecisionTreeClassifier

cancer = arff.loadarff(r'breast.w.arff')
df = pd.DataFrame(cancer[0])
df.dropna(inplace=True)
df = df.replace(df['Class'][0], 0)
x = 0
while df['Class'][x] == 0:
    x += 1
df = df.replace(df['Class'][x],1)
x=df[['Clump_Thickness','Cell_Size_Uniformity','Cell_Shape_Uniformity','Marginal_Adhesion','Single_E
pi_Cell_Size','Bare_Nuclei','Bland_Chromatin','Normal_Nucleoli','Mitoses']]
y = df['Class']

train_accuracy_mean = []
test_accuracy_mean = []

for i in [1, 3, 5, 9]:
    x_new = SelectKBest(chi2, k=i).fit_transform(x, y)

    cv_clf=KFold(n_splits=10,random_state=24,shuffle=True)

    clf = DecisionTreeClassifier()
    cv_results = cross_validate(clf,x_new,y,cv=cv_clf,scoring='accuracy',return_train_score=True)

    train_accuracy_mean.append(cv_results["train_score"].mean())
    test_accuracy_mean.append(cv_results["test_score"].mean())

X = ["1", "3", "5", "9"]
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(X, train_accuracy_mean, color = 'b')
ax.bar(X, test_accuracy_mean, color = 'g')
ax.legend(labels=['Train Accuracy', 'Test Accuracy'], loc=4)
ax.set_xlabel('#Selected Features')
ax.set_ylabel('Accuracy')
ax.set_title('Accuracies by #Selected Features')
plt.show()
from scipy.io import arff
import pandas as pd
```

```
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_validate
from sklearn.tree import DecisionTreeClassifier

cancer = arff.loadarff(r'breast.w.arff')
df = pd.DataFrame(cancer[0])
df.dropna(inplace=True)
df = df.replace(df['Class'][0], 0)
x = 0
while df['Class'][x] == 0:
    x += 1
df = df.replace(df['Class'][x],1)
x=df[['Clump_Thickness','Cell_Size_Uniformity','Cell_Shape_Uniformity','Marginal_Adhesion','Single_E
pi_Cell_Size','Bare_Nuclei','Bland_Chromatin','Normal_Nucleoli','Mitoses']]
y = df['Class']

train_accuracy_mean = []
test_accuracy_mean = []

for i in [1, 3, 5, 9]:
    clf = DecisionTreeClassifier(max_depth=i)

    cv_clf=KFold(n_splits=10,random_state=24,shuffle=True)
    cv_results = cross_validate(clf, x, y, cv=cv_clf, scoring='accuracy', return_train_score=True)

    train_accuracy_mean.append(cv_results["train_score"].mean())
    test_accuracy_mean.append(cv_results["test_score"].mean())

X = ["1", "3", "5", "9"]
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(X, train_accuracy_mean, color = 'b')
ax.bar(X, test_accuracy_mean, color = 'g')
ax.legend(labels=['Train Accuracy', 'Test Accuracy'], loc=4)
ax.set_xlabel('Max Tree Depth')
ax.set_ylabel('Accuracy')
ax.set_title('Accuracies by Max Tree Depth')
plt.show()
```

END