

I. Pen-and-paper

1)

HW 2

1- Design Matrix: $X = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 5 \\ 1 & 0 & 2 & 4 \\ 1 & 1 & 2 & 3 \\ 1 & 2 & 0 & 7 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 2 \\ 1 & 0 & 2 & 9 \end{pmatrix}$

Target vector: $T = \begin{pmatrix} 1 \\ 3 \\ 2 \\ 0 \\ 6 \\ 4 \\ 5 \\ 7 \end{pmatrix}$

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

$f(x) = w_0 + w_1 \|x\|_2^4 + w_2 \|x\|_2^2 + w_3 \|x\|_2^3$

$f(x) = 4,586 - 1,687 \|x\|_2^4 + 0,338 \|x\|_2^2 - 0,013 \|x\|_2^3$

$\text{Obj}(x)$

$x_1: \|x_1\|_2^4 = \sqrt{1^2+1^2+0^2} = \sqrt{2}, \|x_1\|_2^2 = 1^2+1^2+0^2 = 2, \|x_1\|_2^3 = (\sqrt{2})^3 = 2\sqrt{2}$

$x_2: \|x_2\|_2^4 = \sqrt{1^2+1^2+5^2} = \sqrt{27}, \|x_2\|_2^2 = 27, \|x_2\|_2^3 = (\sqrt{27})^3 = 81\sqrt{3}$

$x_3: \|x_3\|_2^4 = \sqrt{0^2+2^2+4^2} = \sqrt{20}, \|x_3\|_2^2 = 20, \|x_3\|_2^3 = (\sqrt{20})^3 = 40\sqrt{5}$

$x_4: \|x_4\|_2^4 = \sqrt{1^2+2^2+3^2} = \sqrt{14}, \|x_4\|_2^2 = 14, \|x_4\|_2^3 = (\sqrt{14})^3 = 14\sqrt{14}$

$x_5: \|x_5\|_2^4 = \sqrt{2^2+0^2+7^2} = \sqrt{53}, \|x_5\|_2^2 = 53, \|x_5\|_2^3 = (\sqrt{53})^3 = 53\sqrt{53}$

$x_6: \|x_6\|_2^4 = \sqrt{1^2+1^2+1^2} = \sqrt{3}, \|x_6\|_2^2 = 3, \|x_6\|_2^3 = (\sqrt{3})^3 = 3\sqrt{3}$

$x_7: \|x_7\|_2^4 = \sqrt{2^2+0^2+2^2} = \sqrt{8}, \|x_7\|_2^2 = 8, \|x_7\|_2^3 = (\sqrt{8})^3 = 16\sqrt{2}$

$x_8: \|x_8\|_2^4 = \sqrt{0^2+2^2+9^2} = \sqrt{85}, \|x_8\|_2^2 = 85, \|x_8\|_2^3 = (\sqrt{85})^3 = 85\sqrt{85}$

Função erro: $E(w) = \sum_{i=1}^8 (t_i - o_i)^2, 0 = wX \Leftrightarrow E(w) = (T - Xw)^T (T - Xw)$

$\frac{\partial E(w)}{\partial w} = (-X)^T (T - Xw) + (T - Xw)^T (-X) = 0 \Leftrightarrow \dots \Leftrightarrow W = (X^T X)^{-1} X^T T = (\Phi^T \Phi)^{-1} \Phi^T T$

$W = \begin{pmatrix} 4,586 & -1,687 & 0,338 & -0,013 \end{pmatrix}^T$

NumPy
Python

$\frac{\partial E(w)}{\partial w} = 0$

$\frac{\partial (T - Xw)^T (T - Xw)}{\partial w} = 0$

$\left(\frac{\partial}{\partial w} (T - Xw)^T \right) (T - Xw) + (T - Xw)^T \left(\frac{\partial}{\partial w} (T - Xw) \right) = 0$

$(-X^T) (T - Xw) + (T - Xw)^T (-X) = 0$

$(-X^T) (T - Xw) + (-X)^T (T - Xw) = 0$

$-2X^T (T - Xw) = 0$

$X^T (T - Xw) = 0$

$X^T T - X^T Xw = 0$

$X^T Xw = X^T T$

$w = (X^T X)^{-1} X^T T$

2)

2- $\text{Out}(x_9) = 4,586 - 1,687(x_1) + 0,338(x_4) - 0,013(x_8) = 2,46$
 $\text{Out}(x_{10}) = 4,586 - 1,687(\sqrt{6}) + 0,338(6) - 0,013(6\sqrt{6}) = 2,291$
 $\text{RMSE} = \sqrt{\frac{(2 - 2,46)^2 + (4 - 2,291)^2}{2}} = 1,251$
 $x_9: \|x_9\|_2 = \sqrt{2^2 + 0^2 + 0^2} = 2, \|x_9\|_2^2 = 4, \|x_9\|_2^3 = (2)^3 = 8$
 $x_{10}: \|x_{10}\|_2 = \sqrt{4^2 + 2^2 + 1^2} = \sqrt{21}, \|x_{10}\|_2^2 = 21, \|x_{10}\|_2^3 = (\sqrt{21})^3 = 21\sqrt{21}$

3)

3- $y_3' = [0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1]^T$ $\text{output}' = [N \ N \ N \ N \ P \ P \ P \ P \ N \ P]$

Só se lembrarmos os dados de treino, pois sabemos que os dados de teste são dados na aprendizagem.

$E(\text{output}') = -(\frac{1}{2} \log \frac{1}{2} + \frac{1}{2} \log \frac{1}{2}) = 1$

$E(\text{output}' | y_1) = \frac{2}{8} E(\frac{1}{2}, \frac{1}{2}) + \frac{4}{8} E(\frac{3}{4}, \frac{1}{4}) + \frac{2}{8} E(1) = 0,656$

$E(\text{output}' | y_2) = \frac{2}{8} E(1) + \frac{3}{8} E(\frac{2}{3}, \frac{1}{3}) + \frac{3}{8} E(\frac{2}{3}, \frac{1}{3}) = 0,689$

$E(\text{output}' | y_3) = \frac{4}{8} E(\frac{1}{2}, \frac{1}{2}) + \frac{4}{8} E(\frac{1}{2}, \frac{1}{2}) = 1$

$IG(\text{output}' | y_i) = E(\text{output}') - E(\text{output}' | y_i)$

Como $E(\text{output}' | y_1) \neq 0$ então, y_1 é a raíz da árvore.

$E_{2,2} = \frac{2}{2} E(\frac{1}{2}, \frac{1}{2}) = 1$ } indy., pois têm igual entropia
 $E_{2,3} = \frac{2}{2} E(\frac{1}{2}, \frac{1}{2}) = 1$ } $P(P) = P(N) = 50\%$

quando $y_1 = 0$, temos y_2 e y_3

$E_{3,2} = \frac{3}{4} E(\frac{2}{3}, \frac{1}{3}) + \frac{1}{4} E(1) = 0,689$

$E_{3,3} = \frac{3}{4} E(\frac{2}{3}, \frac{1}{3}) + \frac{1}{4} E(1) = 0,689$ } igual entropia, logo, escolhemos de menor índice (y_2)

quando $y_1 = 1$, temos y_2 e y_3

Para $y_1 = 1$ e $y_2 = 1$, temos $y_3 = 1$ que leva à classe N e para $y_3 = 0$, temos 50% de prob. de pertencer a N e 50% de prob. de pertencer a P.

| | 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 |
| x_9 | 2 | 0 | 0 | N |
| x_{10} | 1 | 2 | 1 | P |

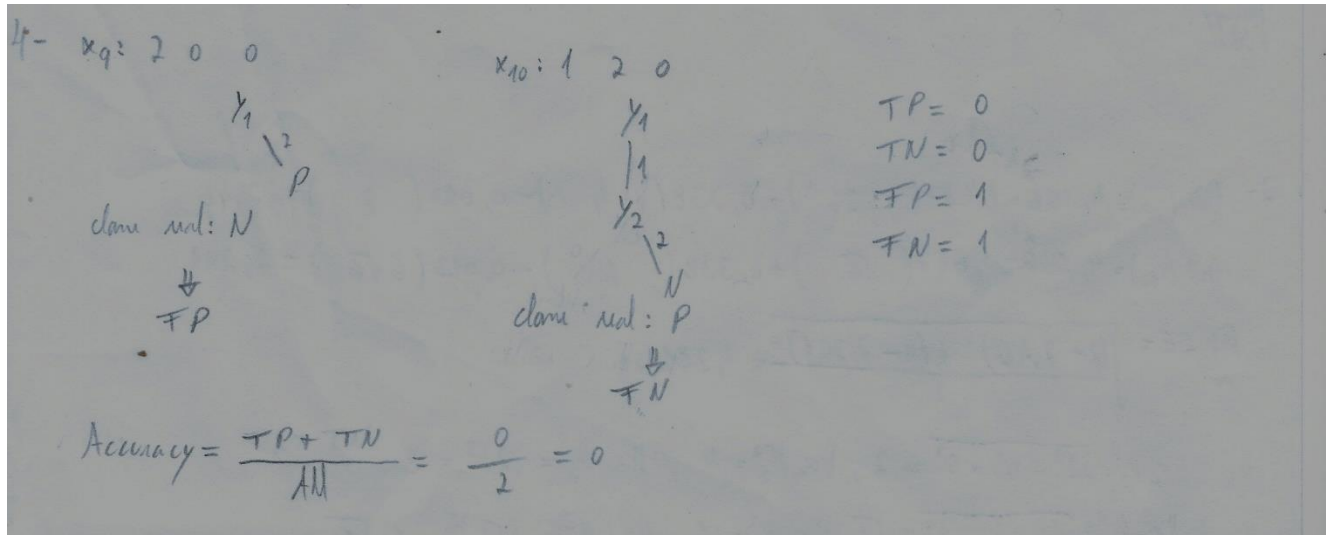
```

graph TD
    Y1((y1)) -- 0 --> Y2_0((y2))
    Y1 -- 1 --> Y2_1((y2))
    Y2_0 -- 1 --> Y3_0_1((y3))
    Y2_0 -- 2 --> Y3_0_2((y3))
    Y2_1 -- 1 --> Y3_1_1((y3))
    Y2_1 -- 2 --> Y3_1_2((y3))
    Y3_0_1 -- 0 --> N1[N]
    Y3_0_1 -- 1 --> N1
    Y3_0_2 -- 0 --> N1
    Y3_0_2 -- 1 --> N1
    Y3_1_1 -- 0 --> N2[N]
    Y3_1_1 -- 1 --> N2
    Y3_1_2 -- 0 --> P1[P]
    Y3_1_2 -- 1 --> P1
  
```

$P(P) = 50\%$
 $P(N) = 50\%$

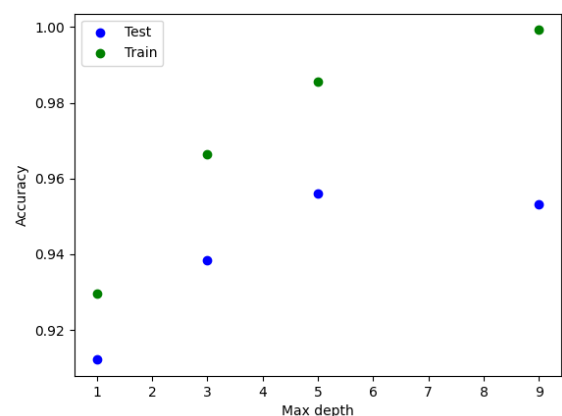
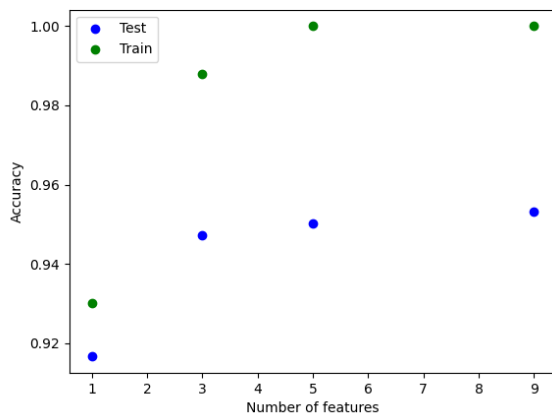
$P(P) = 50\%$
 $P(N) = 50\%$

4)



II. Programming and critical analysis

- 5) A *accuracy* aumenta com o aumento do número de *features*, apesar de aumentar cada vez menos. Em relação à *depth*, a *accuracy* aumenta até *depth*=5, mas diminui ligeiramente para *depth*=9.



- 6) Com o aumento do número de *features*, aumenta a quantidade de dados disponível para relacionar com o output, obtendo-se uma *accuracy* maior, já que é possível prever os dados com maior rigor. Não existe grande diferença entre ter 3, 5 e 9 *features*, pois a adição de mais *features*, cada vez menos relevantes, acaba por não trazer grande significância para a classificação dos dados.

Até uma *depth* de 5 a *accuracy* aumenta, diminuindo ligeiramente para *depth* igual 9. Com o aumento da *depth*, consideramos um nível maior de especificidade, o que acaba por não ser benéfico neste caso, já que nos leva ao *overfitting* (diferença entre *train* e *test accuracy* aumenta), daí a diminuição da *accuracy* com *depth* igual a 9.

- 7) Escolhemos uma *depth* de 5, porque é a que permite obter uma maior *accuracy*, de entre todas as *depths* estudadas. A *depth* de 9 acabou por não ter uma *accuracy* maior devido ao *overfitting*, daí não ser uma boa escolha. Teríamos ainda uma melhor conclusão se considerássemos todas as *depths*.

III. APPENDIX

```
import pandas as pd
from sklearn import metrics
from sklearn import tree
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
from sklearn.model_selection import StratifiedKFold
from statistics import mean
import numpy as np

df = pd.read_csv("test.csv", dtype={"Clump_Thickness": int,
                                     "Cell_Size_Uniformity": int,
                                     "Cell_Shape_Uniformity": int,
                                     "Marginal_Adhesion": int,
                                     "Single_Epi_Cell_Size": int,
                                     "Bare_Nuclei": float,
                                     "Bland_Chromatin": int,
                                     "Normal_Nucleoli": int,
                                     "Mitoses": int,
                                     "Class": str}, na_values=["?"])

df = df.dropna()

columns = ["Clump_Thickness", "Cell_Size_Uniformity", "Cell_Shape_Uniformity",
           "Marginal_Adhesion", "Single_Epi_Cell_Size", "Bare_Nuclei", "Bland_Chromatin",
           "Normal_Nucleoli", "Mitoses", "Class"]
features = columns[:-1]
classes = ["benign", "malign"]

Y = df["Class"]
X = df.drop(columns=["Class"])

Y = [0 if x == "benign" else 1 for x in Y]
Y = np.ravel(pd.DataFrame(Y))

# usar os random_state para gerar sempre a mesma arvore e splits
# usamos o stratified porque existem muito mais targets como benign e nao malign
# assim o stratifiedkfold tem uma proporcao de amostras em cada split
kf = StratifiedKFold(n_splits=10, random_state=132, shuffle=True)

acc_test1 = []
acc_train1 = []

n_features = [1, 3, 5, 9]
for i in n_features:
    mean_acc_test1 = []
    mean_acc_train1 = []

    for train_index, test_index in kf.split(X, Y):
        X_train = X.iloc[train_index].loc[:, features].values
        X_test = X.iloc[test_index][features].values
        Y_train = Y[train_index]
        Y_test = Y[test_index]

        select = SelectKBest(score_func=mutual_info_classif, k=i)
        X_train_new = select.fit_transform(X_train, Y_train)
        X_test_new = select.transform(X_test)
```

Aprendizagem 2021/22
Homework II – Group 96

```
model1 = tree.DecisionTreeClassifier(random_state=0)
model1 = model1.fit(X_train_new, Y_train)

Y_pred_train = model1.predict(X_train_new)
mean_acc_train1 += [metrics.accuracy_score(Y_pred_train, Y_train)]

Y_pred = model1.predict(X_test_new)
mean_acc_test1 += [metrics.accuracy_score(Y_pred, Y_test)]

acc_train1 += [mean(mean_acc_train1)]
acc_test1 += [mean(mean_acc_test1)]

plt.scatter(n_features, acc_test1, color="blue", label="Test")
plt.scatter(n_features, acc_train1, color="green", label="Train")
plt.legend()
plt.xlabel("Number of features")
plt.ylabel("Accuracy")
plt.show() # grafico 5-i)

acc_test2 = []
acc_train2 = []

max_depth = n_features
for i in max_depth:
    mean_acc_test2 = []
    mean_acc_train2 = []

    for train_index, test_index in kf.split(X, Y):
        X_train2 = X.iloc[train_index].loc[:, features].values
        X_test2 = X.iloc[test_index][features].values
        Y_train2 = Y[train_index]
        Y_test2 = Y[test_index]

        model2 = tree.DecisionTreeClassifier(max_depth=i, random_state=0)
        model2 = model2.fit(X_train2, Y_train2)

        Y_pred_train2 = model2.predict(X_train2)
        mean_acc_train2 += [metrics.accuracy_score(Y_pred_train2, Y_train2)]

        Y_pred2 = model2.predict(X_test2)
        mean_acc_test2 += [metrics.accuracy_score(Y_pred2, Y_test2)]

    acc_train2 += [mean(mean_acc_train2)]
    acc_test2 += [mean(mean_acc_test2)]

plt.scatter(max_depth, acc_test2, color="blue", label="Test")
plt.scatter(max_depth, acc_train2, color="green", label="Train")
plt.legend()
plt.xlabel("Max depth")
plt.ylabel("Accuracy")
plt.show() # grafico 5-ii)

print(acc_test1)
print(acc_test2)
```

END