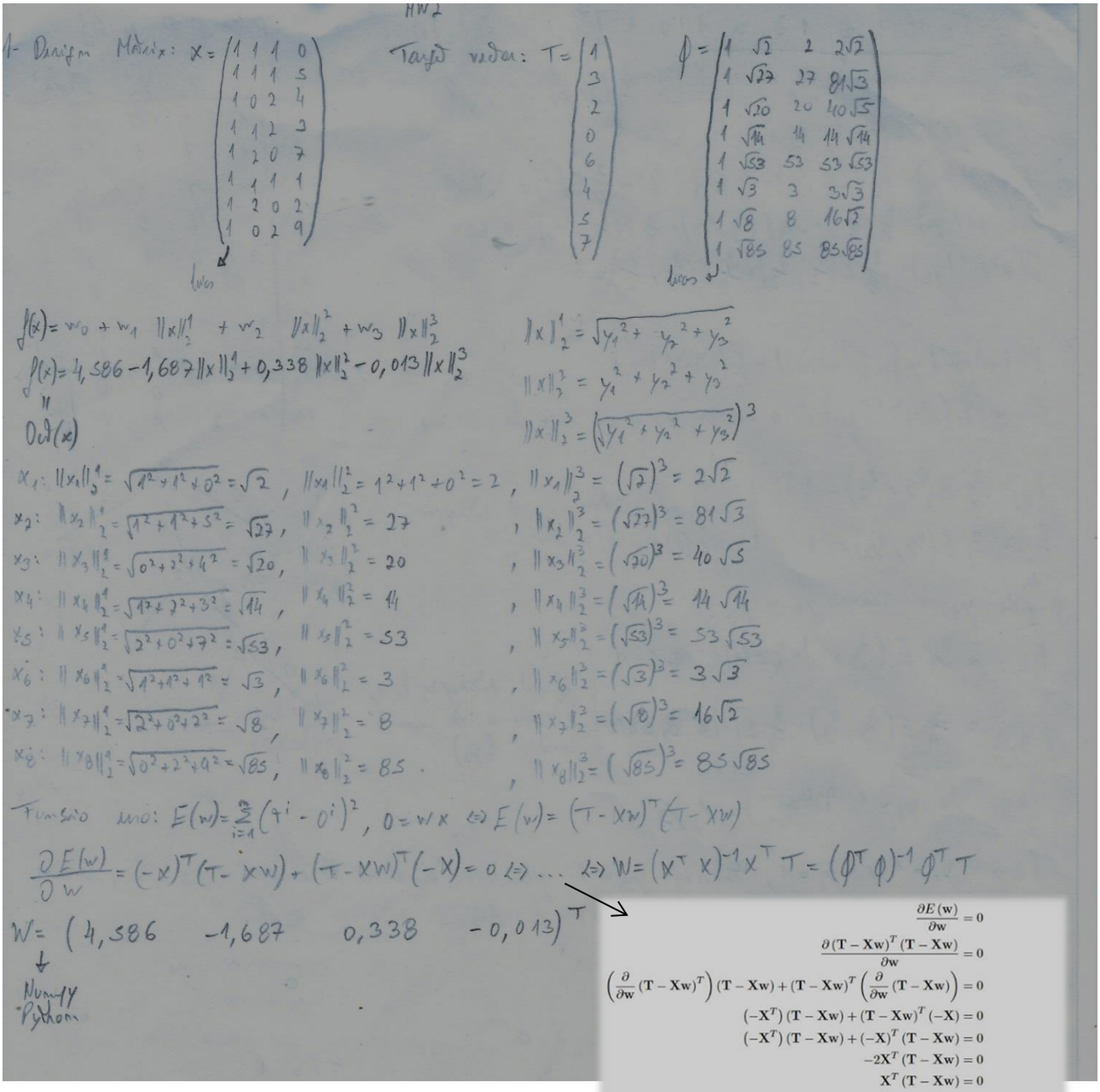
TÉCNICO LISBOA

Aprendizagem 2021/22

Homework II – Group 96

I. Pen-and-paper

1)



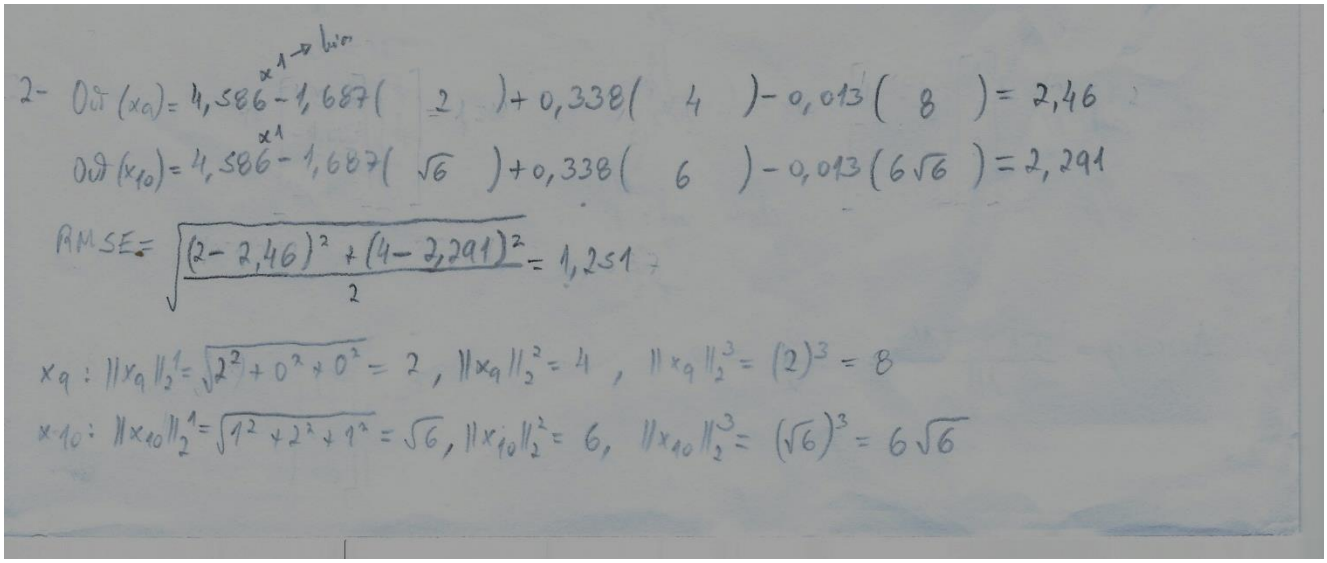
$$
\begin{aligned}
\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} &= 0 \\
\frac{\partial (\mathbf{T}-\mathbf{X}\mathbf{w})^T(\mathbf{T}-\mathbf{X}\mathbf{w})}{\partial \mathbf{w}} &= 0 \\
\left(\frac{\partial}{\partial \mathbf{w}}(\mathbf{T}-\mathbf{X}\mathbf{w})^T\right)(\mathbf{T}-\mathbf{X}\mathbf{w}) + (\mathbf{T}-\mathbf{X}\mathbf{w})^T\left(\frac{\partial}{\partial \mathbf{w}}(\mathbf{T}-\mathbf{X}\mathbf{w})\right) &= 0 \\
(-\mathbf{X}^T)(\mathbf{T}-\mathbf{X}\mathbf{w}) + (\mathbf{T}-\mathbf{X}\mathbf{w})^T(-\mathbf{X}) &= 0 \\
(-\mathbf{X}^T)(\mathbf{T}-\mathbf{X}\mathbf{w}) + (-\mathbf{X})^T(\mathbf{T}-\mathbf{X}\mathbf{w}) &= 0 \\
-2\mathbf{X}^T(\mathbf{T}-\mathbf{X}\mathbf{w}) &= 0 \\
\mathbf{X}^T(\mathbf{T}-\mathbf{X}\mathbf{w}) &= 0 \\
\mathbf{X}^T\mathbf{T} - \mathbf{X}^T\mathbf{X}\mathbf{w} &= 0 \\
\mathbf{X}^T\mathbf{X}\mathbf{w} &= \mathbf{X}^T\mathbf{T} \\
\mathbf{w} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{T}
\end{aligned}
$$

TÉCNICO LISBOA

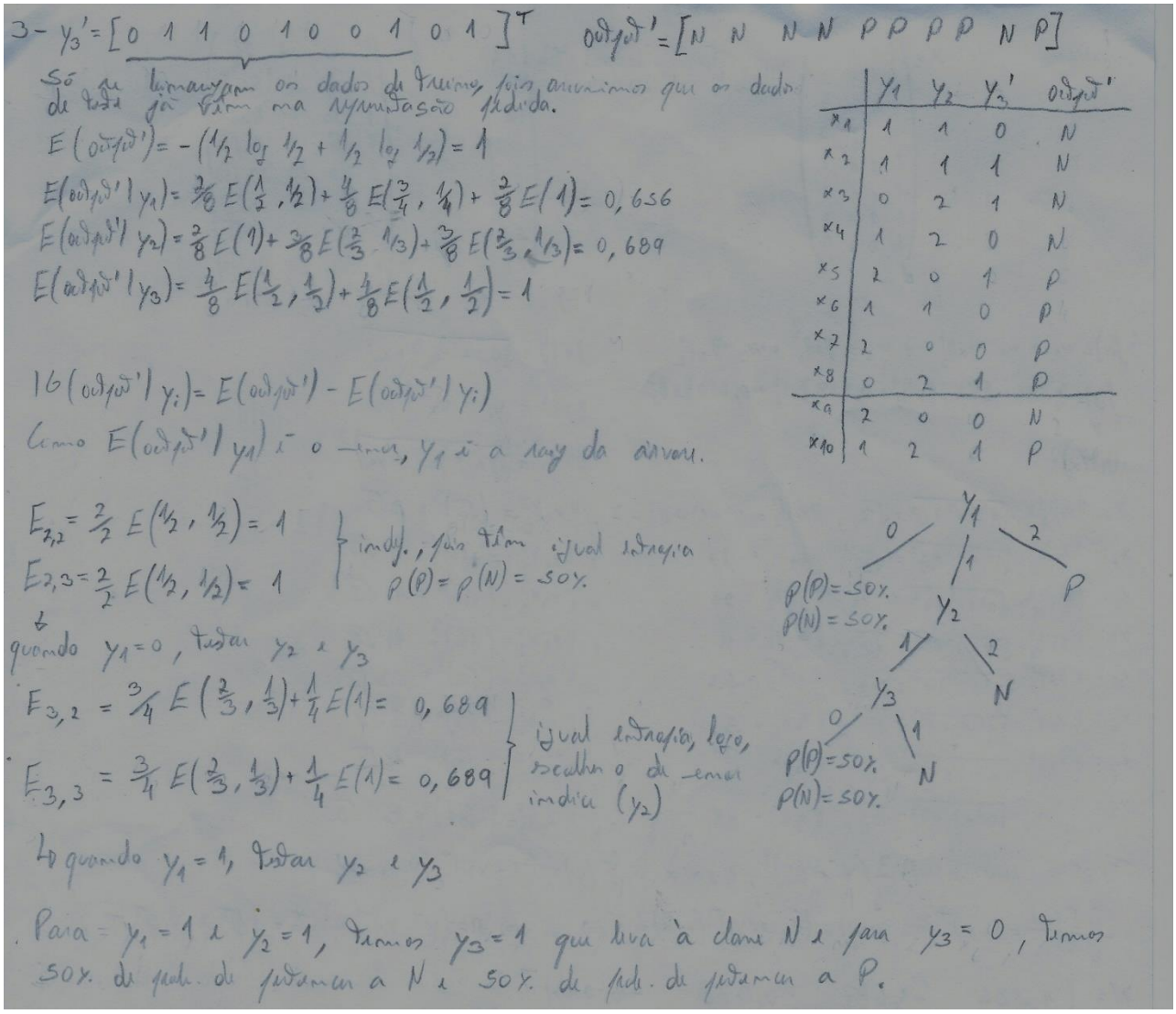
Aprendizagem 2021/22

Homework II – Group 96

2)



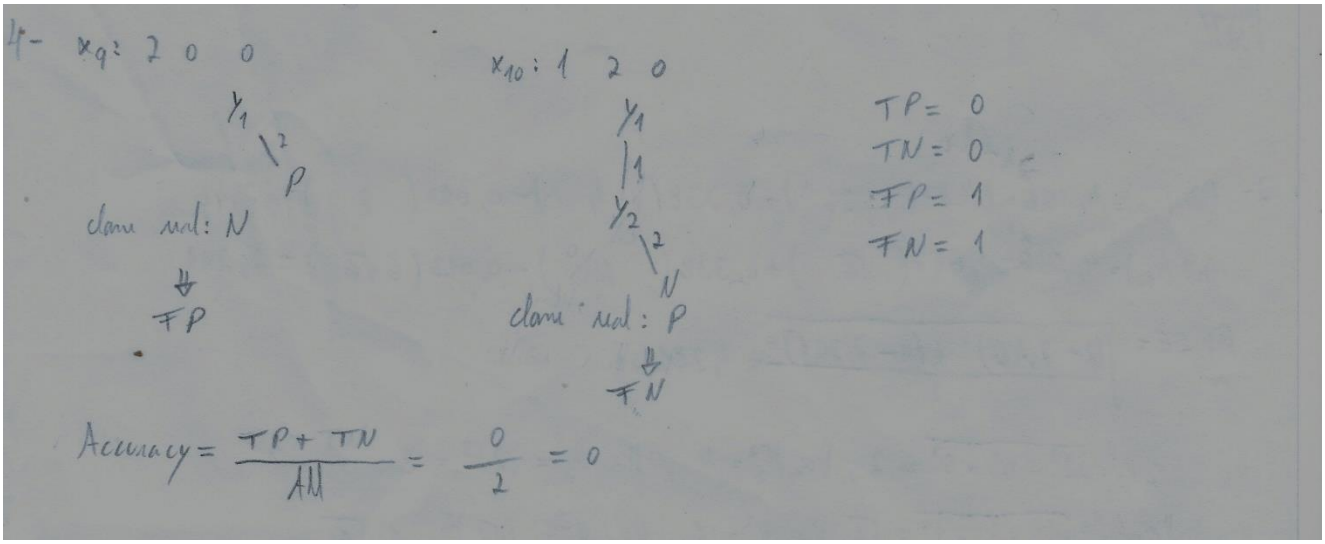
3)



Aprendizagem 2021/22

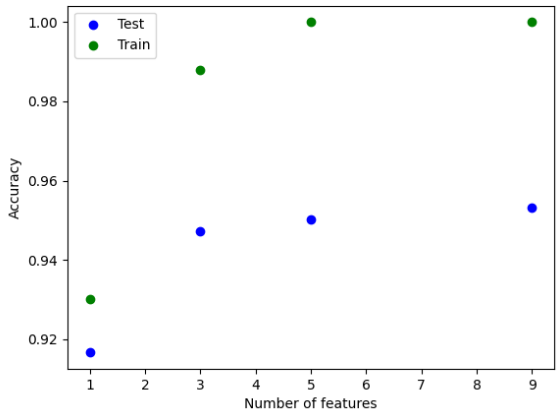
Homework II – Group 96

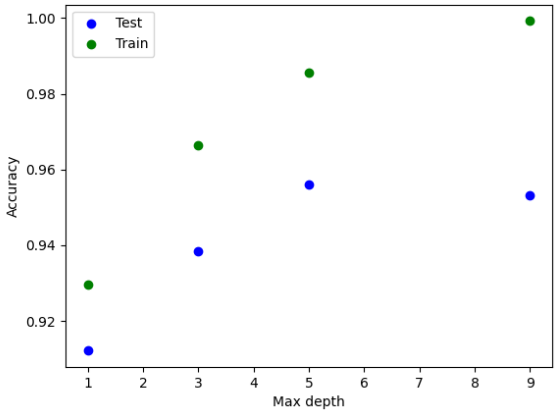
4)



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*.

TÉCNICO LISBOA

Aprendizagem 2021/22

Homework II – Group 96

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)
```

```
        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