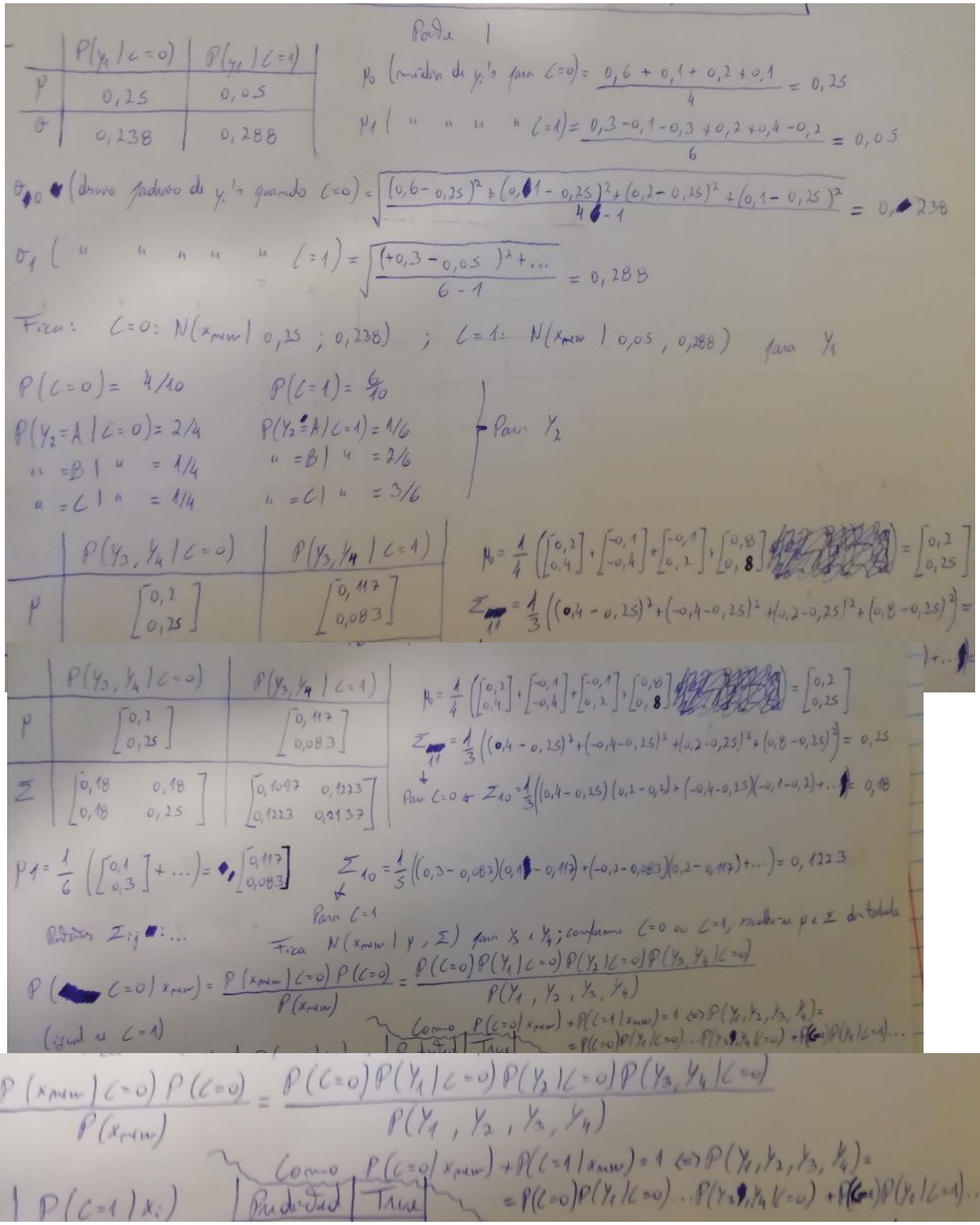
TÉCNICO LISBOA

Aprendizagem 2021/22

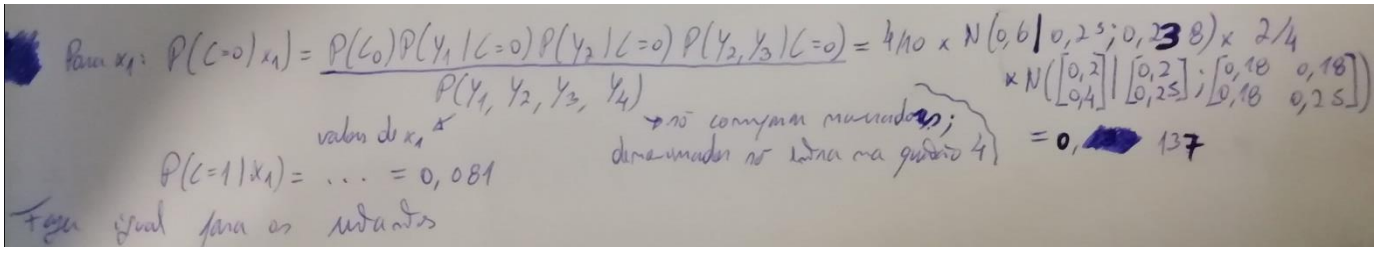
Homework I – Group 96

I. Pen-and-paper

1)



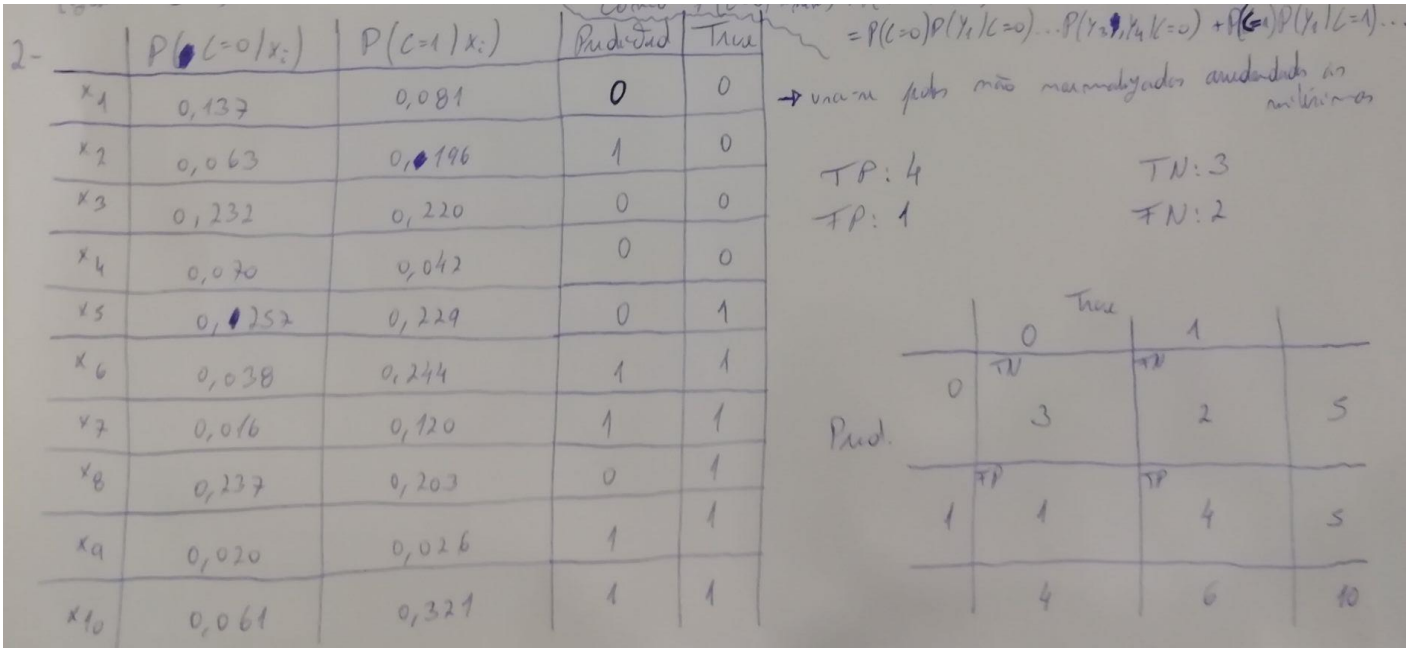
2)



TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 96



	$P(C=0\mid x_i)$	$P(C=1\mid x_i)$	Predicted	True
x_1	0,137	0,081	0	0
x_2	0,063	0,196	1	0
x_3	0,232	0,220	0	0
x_4	0,070	0,042	0	0
x_5	0,257	0,229	0	1
x_6	0,038	0,244	1	1
x_7	0,016	0,120	1	1
x_8	0,237	0,203	0	1
x_9	0,020	0,026	1	1
x_{10}	0,061	0,321	1	1

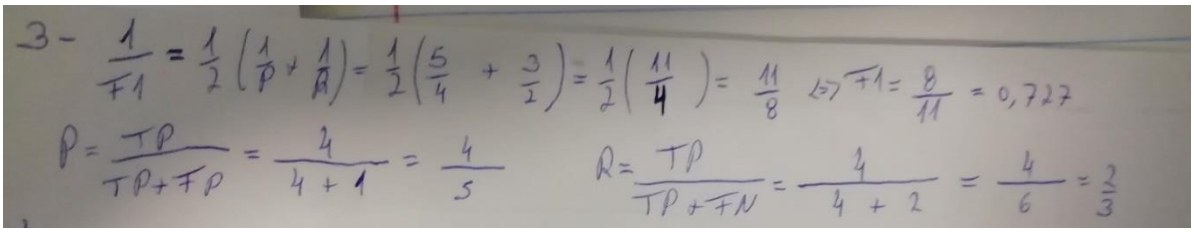
→ usamos pesos não normalizados anteriormente

TP: 4 TN: 3

FP: 1 FN: 2

Pred. \ True	0	1	
0	TN 3	FN 2	5
1	FP 1	TP 4	5
	4	6	10

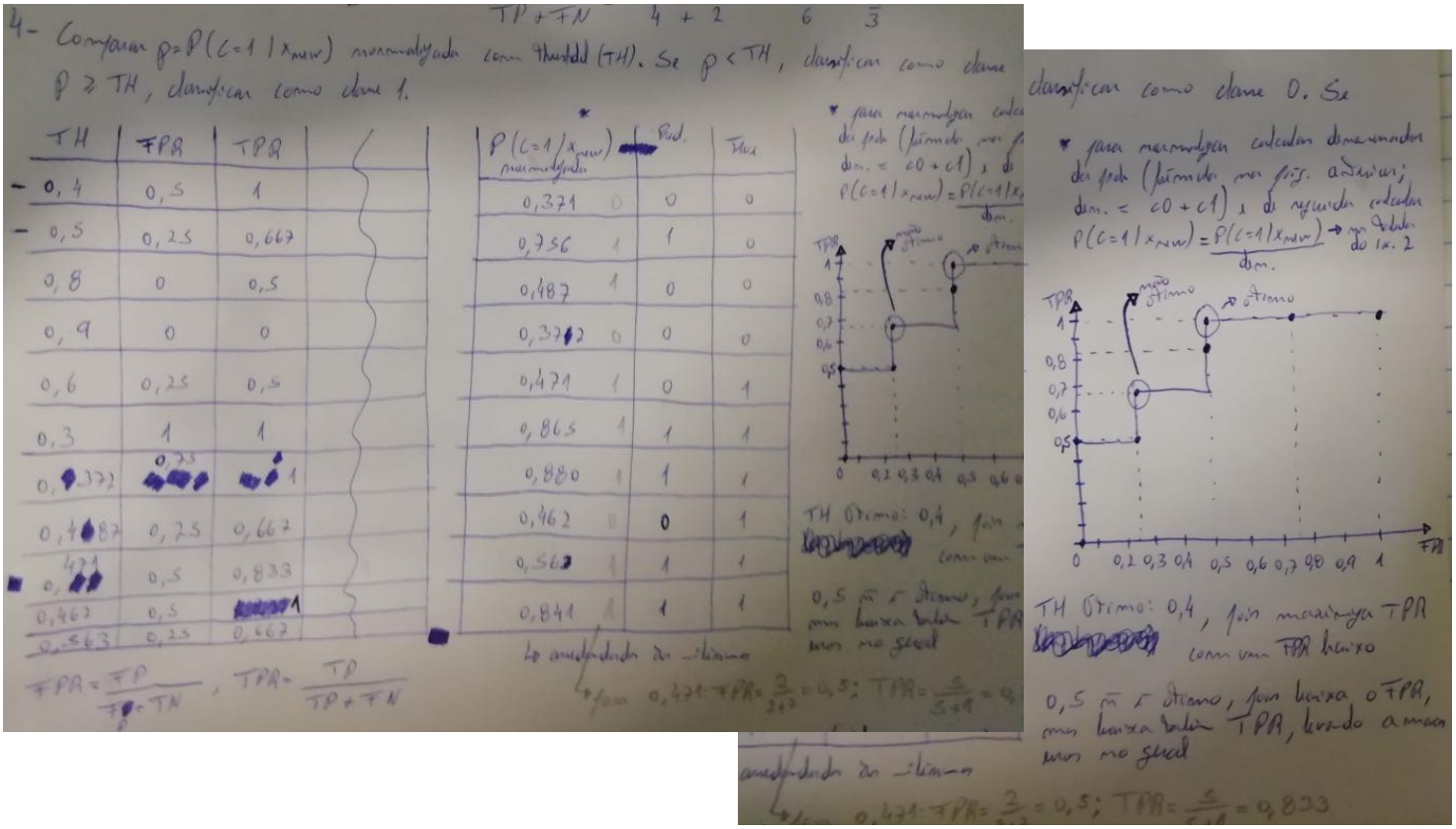
3)



$$\frac{1}{F1} = \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right) = \frac{1}{2}\left(\frac{5}{4} + \frac{3}{2}\right) = \frac{1}{2}\left(\frac{11}{4}\right) = \frac{11}{8} \Rightarrow F1 = \frac{8}{11} = 0{,}727$$

$$P = \frac{TP}{TP+FP} = \frac{4}{4+1} = \frac{4}{5} \qquad R = \frac{TP}{TP+FN} = \frac{4}{4+2} = \frac{4}{6} = \frac{2}{3}$$

4)



4- Comparar $p = P(C=1 \mid x_{new})$ normalizada com threshold (TH). Se $p < TH$, classificar como classe 0. Se $p \geq TH$, classificar como classe 1.

TH	FPR	TPR
0,4	0,5	1
0,5	0,25	0,667
0,8	0	0,5
0,9	0	0
0,6	0,25	0,5
0,3	1	1
0,372	0,25	1
0,487	0,25	0,667
0,421	0,5	0,833
0,463	0,5	1
0,563	0,25	0,667

$P(C=1\mid x_{new})$ normalizada	Pred.	True
0,371	0	0
0,756	1	0
0,487	0	0
0,3712	0	0
0,471	0	1
0,865	1	1
0,880	1	1
0,462	0	1
0,563	1	1
0,841	1	1

$FPR = \frac{FP}{FP+TN}$, $TPR = \frac{TP}{TP+FN}$

TH ótimo: 0,4, pois maximiza TPR com um FPR baixo

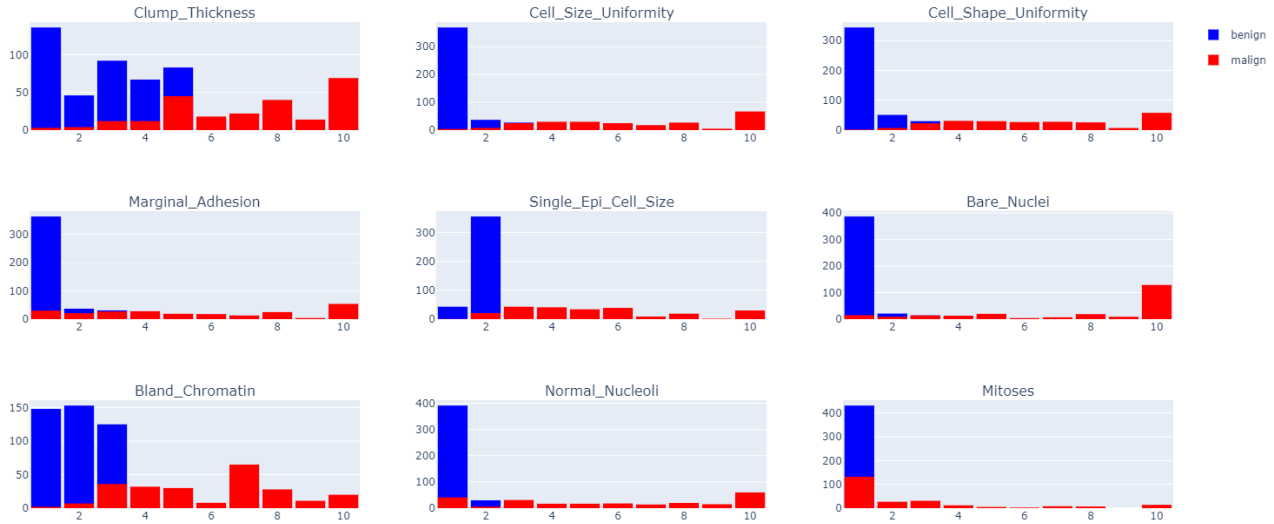
0,5 ou 0,7 ótimo, pois baixa o FPR, mas baixa bastante o TPR, levando a maus resultados no geral

Aprendizagem 2021/22

Homework I – Group 96

II. Programming and critical analysis

5)



6) 3NN Mean Test Accuracy = 96.6%, 5NN Mean Test Accuracy = 97.1%, 7NN Mean Test Accuracy = 96.9%

O 7NN é menos suscetível ao risco de overfitting, porque considera os 7 vizinhos mais próximos do novo elemento a classificar, levando a uma análise melhor, já que acaba por não ser tão influenciado por dados mal classificados junto do novo elemento. Quanto menor o nº de vizinhos, maior o grau de overfitting. Conforme se aumenta o nº de vizinhos, o overfitting diminui, mas a partir de certa altura começa a haver underfitting.

7) Ttest pvalue = 0.0004326 = 0.04326%. Como o pvalue é menor que o threshold de 1%, 5% e 10%, podemos rejeitar a hipótese nula de accuracies iguais entre o 3NN e o Naïve Bayes (multinomial assumption).

8) Com os dados obtidos anteriormente, concluímos que o KNN tem uma accuracy maior que o Naïve Bayes. Uma razão para isto acontecer é o facto do KNN lidar melhor com o overfitting do que o Naïve Bayes, levando a uma melhor classificação dos dados de teste. Outra razão para isto acontecer é devido ao Naïve Bayes assumir que as features são independentes quando na verdade isso pode não acontecer.

III. APPENDIX

```
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold
from sklearn import neighbors
from sklearn.metrics import accuracy_score
from statistics import mean
from scipy import stats
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.naive_bayes import MultinomialNB

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]

fig = make_subplots(rows=3, cols=3, subplot_titles=("Clump_Thickness",
"Cell_Size_Uniformity", "Cell_Shape_Uniformity", "Marginal_Adhesion",
"Single_Epi_Cell_Size", "Bare_Nuclei", "Bland_Chromatin", "Normal_Nucleoli",
"Mitoses"))

ben = df[df["Class"] == "benign"]
mal = df[df["Class"] == "malignant"]

fig.add_trace(go.Histogram(
    x=ben["Clump_Thickness"],
    bingroup=1, name="benign", legendgroup='group1', marker_color='#0000FF'),
row=1, col=1)

fig.add_trace(go.Histogram(
    x=mal["Clump_Thickness"],
    bingroup=1, name="malign", legendgroup='group2', marker_color="#FF0000"),
row=1, col=1)

fig.add_trace(go.Histogram(
    x=ben["Cell_Size_Uniformity"],
    bingroup=1, name="benign", legendgroup='group1', showlegend=False,
marker_color='#0000FF'), row=1, col=2)

fig.add_trace(go.Histogram(
    x=mal["Cell_Size_Uniformity"],
    bingroup=1, name="malign", legendgroup='group2', showlegend=False,
marker_color="#FF0000"), row=1, col=2)

fig.add_trace(go.Histogram(
    x=ben["Cell_Shape_Uniformity"],
    bingroup=1, name="benign", legendgroup='group1', showlegend=False,
marker_color='#0000FF'), row=1, col=3)

fig.add_trace(go.Histogram(
    x=mal["Cell_Shape_Uniformity"],
    bingroup=1, name="malign", legendgroup='group2', showlegend=False,
marker_color="#FF0000"), row=1, col=3)

fig.add_trace(go.Histogram(
    x=ben["Marginal_Adhesion"],
    bingroup=1, name="benign", legendgroup='group1', showlegend=False,
marker_color='#0000FF'), row=2, col=1)

fig.add_trace(go.Histogram(
    x=mal["Marginal_Adhesion"],
    bingroup=1, name="malign", legendgroup='group2', showlegend=False,
marker_color="#FF0000"), row=2, col=1)

fig.add_trace(go.Histogram(
```

```
    x=ben["Single_Epi_Cell_Size"],
    bingroup=1, name="benign", legendgroup='group1', showlegend=False,
marker_color='#0000FF'), row=2, col=2)

fig.add_trace(go.Histogram(
    x=mal["Single_Epi_Cell_Size"],
    bingroup=1, name="malign", legendgroup='group2', showlegend=False,
marker_color="#FF0000"), row=2, col=2)

fig.add_trace(go.Histogram(
    x=ben["Bare_Nuclei"],
    bingroup=1, name="benign", legendgroup='group1', showlegend=False,
marker_color='#0000FF'), row=2, col=3)

fig.add_trace(go.Histogram(
    x=mal["Bare_Nuclei"],
    bingroup=1, name="malign", legendgroup='group2', showlegend=False,
marker_color="#FF0000"), row=2, col=3)

fig.add_trace(go.Histogram(
    x=ben["Bland_Chromatin"],
    bingroup=1, name="benign", legendgroup='group1', showlegend=False,
marker_color='#0000FF'), row=3, col=1)

fig.add_trace(go.Histogram(
    x=mal["Bland_Chromatin"],
    bingroup=1, name="malign", legendgroup='group2', showlegend=False,
marker_color="#FF0000"), row=3, col=1)

fig.add_trace(go.Histogram(
    x=ben["Normal_Nucleoli"],
    bingroup=1, name="benign", legendgroup='group1', showlegend=False,
marker_color='#0000FF'), row=3, col=2)

fig.add_trace(go.Histogram(
    x=mal["Normal_Nucleoli"],
    bingroup=1, name="malign", legendgroup='group2', showlegend=False,
marker_color="#FF0000"), row=3, col=2)

fig.add_trace(go.Histogram(
    x=ben["Mitoses"],
    bingroup=1, name="benign", legendgroup='group1', showlegend=False,
marker_color='#0000FF'), row=3, col=3)

fig.add_trace(go.Histogram(
    x=mal["Mitoses"],
    bingroup=1, name="malign", legendgroup='group2', showlegend=False,
marker_color="#FF0000"), row=3, col=3)

fig.update_layout(
    barmode="overlay",
    bargap=0.1)

fig.show()  # Pergunta 5

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

kf10 = KFold(n_splits=10)
model = neighbors.KNeighborsClassifier(n_neighbors=3, weights='uniform', p=2)
gnb = MultinomialNB()
```

TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 96

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

acc_knn = []
acc_gnb = []
i = 1
for train_index, test_index in kf10.split(df):
    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]

    model.fit(X_train, y_train)
    gnb.fit(X_train, y_train)
    acc_knn += [accuracy_score(y_test, model.predict(X_test))]
    acc_gnb += [accuracy_score(y_test, gnb.predict(X_test))]
    i += 1

print("KNN: ", mean(acc_knn), " GNB: ", mean(acc_gnb))  # Pergunta 6

print(stats.ttest_rel(acc_knn, acc_gnb))  # Pergunta 7
```

END

Nota: Não houve paridade de esforço entre os elementos do grupo. Eu, Afonso Ferreira - 96832, fiz este homework sozinho.