**I. Pen-and-paper**

1. Calendar

   Description automatically generated  
   A picture containing diagram

   Description automatically generated  
   Text

   Description automatically generated  
   Text

   Description automatically generated  
   Text

   Description automatically generated  
   Diagram

   Description automatically generated
2. Text

   Description automatically generated  
   Text

   Description automatically generated  
   Text

   Description automatically generated with medium confidence  
   Text

   Description automatically generated  
   A picture containing text, night sky

   Description automatically generated  
   Text, arrow

   Description automatically generated  
   A picture containing text, electronics, keyboard

   Description automatically generated



Uma imagem com texto, janela

Descrição gerada automaticamente

**Uma imagem com seta

Descrição gerada automaticamente**

**Uma imagem com texto

Descrição gerada automaticamente**

**Uma imagem com texto

Descrição gerada automaticamente**

**4)**

Uma imagem com texto

Descrição gerada automaticamente

**Uma imagem com texto

Descrição gerada automaticamente**

**Uma imagem com texto

Descrição gerada automaticamente**

**II. Programming and critical analysis**

**5)**

Chart

Description automatically generated with low confidenceChart

Description automatically generated

6) Para um limite de confiança alpha = 0.05, tem-se que kNN > NB, uma vez que o respetivo p-value é menor que alpha, i.e., 0.00024 < p-value . Obtivemos este resultado com a realização da normalização das variáveis.

7) A diferença observável da precisão preditiva entre o kNN e o Naïve Bayes podem ser explicas por:

- O kNN é melhor para determinar semelhança entre observações que é o que fizemos nos exercícios anteriores devido à sua localidade já que limita o número de observações em estudo enquanto o Naïve Bayes é melhor para tratar muitas observações e para fazer previsões em tempo real pois gera probabilidades para cada classe e, portanto, caso haja introdução de novos dados o valor das probabilidades é alterado.

-O kNN tem maior tendência de ocorrer overfit e a quanto mais abrangente for o valor dado ao kNN menos preciso será a previsão

- O kNN tem a tendência a ter menos outliers, pois trata-se de observações filtradas e limitadas, enquanto o Naïve Bayes faz previsão com todas as observações possíveis.

**III. APPENDIX**

import pandas as pd  
import numpy as np  
  
#\*####################################################################################  
#\* 5)  
#\*####################################################################################  
  
#\* Import Data  
from scipy.io.arff import loadarff  
data = loadarff("pd\_speech.arff")  
df = pd.DataFrame(data[0])  
df['class'] = df['class'].str.decode('utf-8')  
num\_columns = df.shape[1]  
df['class'] = pd.to\_numeric(df["class"])  
  
#\* Pre-Process Data: scale data (relevant for kNN)  
from sklearn.preprocessing import MinMaxScaler, StandardScaler  
df\_scaled = df.copy()  
df\_scaled.iloc[:, 0:num\_columns-1] = MinMaxScaler().fit\_transform(df.iloc[:, 0:num\_columns-1])  
  
#\* Partition Data  
kNN\_X, kNN\_y = df\_scaled.iloc[:, 0:num\_columns-1], df\_scaled["class"]  
NB\_X, NB\_y = df.iloc[:, 0:num\_columns-1], df\_scaled["class"]  
  
#\* Classifier  
from sklearn.neighbors import KNeighborsClassifier  
kNN\_predictor = KNeighborsClassifier(n\_neighbors=5, weights="uniform", metric="minkowski")  
from sklearn.naive\_bayes import GaussianNB  
NB\_predictor = GaussianNB()  
  
#\* Cross-Validation and Cumulative Confusion Matrixes  
from sklearn.model\_selection import StratifiedKFold  
from sklearn.metrics import confusion\_matrix  
folds = StratifiedKFold(n\_splits=10)  
kNN\_cumulative = np.array([[0, 0], [0, 0]])  
for train\_k, test\_k in folds.split(kNN\_X, kNN\_y):  
 kNN\_X\_train, kNN\_X\_test = kNN\_X.iloc[train\_k], kNN\_X.iloc[test\_k]  
 kNN\_y\_train, kNN\_y\_test = kNN\_y.iloc[train\_k], kNN\_y.iloc[test\_k]  
  
 kNN\_predictor.fit(kNN\_X\_train, kNN\_y\_train)  
 kNN\_y\_pred = kNN\_predictor.predict(kNN\_X\_test)  
 #? qual é Healthy e qual é Parkinsons?  
 kNN\_cm = np.array(confusion\_matrix(kNN\_y\_test, kNN\_y\_pred, labels=[0,1]))  
 kNN\_cumulative[0][0] += kNN\_cm[0][0]  
 kNN\_cumulative[0][1] += kNN\_cm[0][1]  
 kNN\_cumulative[1][0] += kNN\_cm[1][0]  
 kNN\_cumulative[1][1] += kNN\_cm[1][1]  
kNN\_cumulative\_confusion = pd.DataFrame(kNN\_cumulative, index=['Healthy', 'Parkinsons'], columns=['Predicted Healthy', 'Predicted Parkinsons'])  
NB\_cumulative = np.array([[0, 0], [0, 0]])  
for train\_k, test\_k in folds.split(NB\_X, NB\_y):  
 NB\_X\_train, NB\_X\_test = NB\_X.iloc[train\_k], NB\_X.iloc[test\_k]  
 NB\_y\_train, NB\_y\_test = NB\_y.iloc[train\_k], NB\_y.iloc[test\_k]  
  
 NB\_predictor.fit(NB\_X\_train, NB\_y\_train)  
 NB\_y\_pred = NB\_predictor.predict(NB\_X\_test)  
 #? qual é Healthy e qual é Parkinsons?  
 NB\_cm = np.array(confusion\_matrix(kNN\_y\_test, kNN\_y\_pred, labels=[0,1]))  
 NB\_cumulative[0][0] += NB\_cm[0][0]  
 NB\_cumulative[0][1] += NB\_cm[0][1]  
 NB\_cumulative[1][0] += NB\_cm[1][0]  
 NB\_cumulative[1][1] += NB\_cm[1][1]  
NB\_cumulative\_confusion = pd.DataFrame(NB\_cumulative, index=['Healthy', 'Parkinsons'], columns=['Predicted Healthy', 'Predicted Parkinsons'])  
  
#\* Plot  
import seaborn as sns  
import matplotlib.pyplot as plt  
sns.heatmap(kNN\_cumulative\_confusion, annot=True, fmt='g')  
plt.title(label="kNN")  
plt.show()  
sns.heatmap(NB\_cumulative\_confusion, annot=True, fmt='g')  
plt.title(label="Naive Bayes")  
plt.show()  
  
#\*####################################################################################  
#\* 6)  
#\*####################################################################################  
#\* Test Hypothesis

from sklearn.model\_selection import cross\_val\_score  
kNN\_fold\_accs = cross\_val\_score(kNN\_predictor, kNN\_X, kNN\_y, cv = 10, scoring="accuracy")  
NB\_fold\_accs = cross\_val\_score(kNN\_predictor, NB\_X, NB\_y, cv = 10, scoring="accuracy")  
print(kNN\_fold\_accs)  
print(NB\_fold\_accs)  
  
from scipy import stats  
# kNN is better than NB?  
res = stats.ttest\_rel(kNN\_fold\_accs, NB\_fold\_accs, alternative='greater')  
print("kNN > NB? pval=", res.pvalue)  
# kNN is worse than NB?  
res = stats.ttest\_rel(kNN\_fold\_accs, NB\_fold\_accs, alternative='less')  
print("kNN < NB? pval=", res.pvalue)  
# kNN is differs from NB?  
res = stats.ttest\_rel(kNN\_fold\_accs, NB\_fold\_accs, alternative='two-sided')  
print("kNN != NB? pval=", res.pvalue)

**END**