Aprendizagem 2023
Homework I – Group 036
(ist1103248, ist1103327)

Part I: Pen and paper

1. a)

1a)

Teorema de Bayes

$$P(Y_6 \mid Y_1, Y_2, Y_3, Y_4, Y_5) = \frac{P(Y_1, Y_2, Y_3, Y_4, Y_5 \mid Y_6) \times P(Y_6)}{P(Y_1, Y_2, Y_3, Y_4, Y_5)}$$

Pela in – dependência descrito na i)

$$= \frac{P(Y_1, Y_2 \mid Y_6) \times P(Y_3, Y_4 \mid Y_6) \times P(Y_5 \mid Y_6) \times P(Y_6)}{P(Y_1, Y_2) \times P(Y_3, Y_4) \times P(Y_5)}$$

Priors: $P(Y_6 = A) = \frac{3}{7}$, $P(Y_6 = B) = \frac{4}{7}$

Probability mass functions:

$P(Y_3, Y_4 \mid Y_6)$: $P(Y_3 = 0, Y_4 = 0 \mid A) = \frac{0}{3}$

$P(Y_3 = 0, Y_4 = 1 \mid A) = \frac{1}{3}$

$P(Y_3 = 1, Y_4 = 0 \mid A) = \frac{1}{3}$

$P(Y_3 = 1, Y_4 = 1 \mid A) = \frac{1}{3}$

$P(Y_3 = 0, Y_4 = 0 \mid B) = \frac{2}{4}$

$P(Y_3 = 0, Y_4 = 1 \mid B) = \frac{1}{4}$

$P(Y_3 = 1, Y_4 = 0 \mid B) = \frac{1}{4}$

$P(Y_3 = 1, Y_4 = 0 \mid B) = \frac{0}{4}$

$P(Y_5 \mid Y_6)$: $P(Y_5 = 0 \mid A) = \frac{1}{3}$

$P(Y_5 = 1 \mid A) = \frac{1}{3}$

$P(Y_5 = 2 \mid A) = \frac{1}{3}$

$P(Y_5 = 0 \mid B) = \frac{1}{4}$

$P(Y_5 = 1 \mid B) = \frac{2}{4}$

$P(Y_5 = 2 \mid B) = \frac{1}{4}$

Probability Density Function:

$P(Y_1, Y_2 \mid Y_6)$ => vamos ter de calcular os valores médios μ e matrizes de covariância Σ

Para a classe A: [0.24, 0.36] [0.16, 0.48] [0.32, 0.72]

$$\mu = \begin{bmatrix} \text{média } y_1 \\ \text{média } y_2 \end{bmatrix} = \begin{bmatrix} \frac{0.24+0.16+0.32}{3} \\ \frac{0.36+0.48+0.72}{3} \end{bmatrix} = \begin{bmatrix} 0.24 \\ 0.52 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} var(y_1) & cov(y_1, y_2) \\ cov(y_1, y_2) & var(y_2) \end{bmatrix} \quad var(x) = \frac{\sum(x_i^2) - n\bar{x}^2}{n-1}$$

$$cov(y_1, y_2) = \frac{\sum(y_1 - \bar{y_1}) \times (y_2 - \bar{y_2})}{n-1}$$

$$var(y_1) = \frac{(0.24^2 + 0.16^2 + 0.32^2) - 3 \times \left(\frac{0.72}{3}\right)^2}{2}$$

$$= \frac{0.1856 - 3 \times \frac{0.5184}{9}}{2} = \frac{0.0128}{2} = 0.0064$$

$$var(y_2) = \frac{(0.36^2 + 0.48^2 + 0.72^2) - 3 \times \left(\frac{1.56}{3}\right)^2}{2}$$

$$= \frac{0.8784 - 3 \times \frac{2.4336}{9}}{2} = \frac{0.0672}{2} = 0.0336$$

$$cov(y_1, y_2) = \frac{(0.24-0.24)\times(0.36-0.52) + (0.16-0.24)\times(0.48-0.52) + (0.32-0.24)\times(0.72-0.52)}{2}$$

$$= \frac{0 + 0.032 + 0.016}{2} = \frac{0.0192}{2} = 0.0096$$

$$\Sigma = \begin{bmatrix} 0.0064 & 0.0096 \\ 0.0096 & 0.0336 \end{bmatrix}$$

$$N\left(\mu = \begin{bmatrix} 0.24 \\ 0.52 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.0064 & 0.0096 \\ 0.0096 & 0.0336 \end{bmatrix} \middle| A\right)$$

Para a classe B: [0.54, 0.11] [0.66, 0.39] [0.76, 0.28] [0.41, 0.53]

$$\mu = \begin{bmatrix} 0.5925 \\ 0.3275 \end{bmatrix} \quad var(y_1) = 0.02289 \quad cov(y_1, y_2) = -0.0097583$$

$$var(y_2) = 0.03149$$

$$\Sigma = \begin{bmatrix} 0.02289 & -0.0097583 \\ -0.0097583 & 0.03149 \end{bmatrix}$$

$$\mathcal{N}\left(\mu = \begin{bmatrix} 0.5925 \\ 0.3275 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.02289 & -0.0097583 \\ -0.0097583 & 0.03149 \end{bmatrix} \middle| B\right)$$

Nota: não vamos ver os priors de $P(y_1, y_2)$, $P(y_3, y_4)$ e $P(y_5)$ já que não vamos precisar de usar nos próximos exercícios.

b)

b) MAP:

$$h_{MAP} = \text{argmax } p(D|h) \times p(h)$$

$$x_8 = [0.38, 0.52, 0, 1, 0]^T$$

$$P(x_8|A) \times P(A) = P(y_1 = 0.38, y_2 = 0.52 | A) \times P(y_3 = 0, y_4 = 1 | A) \times P(y_5 = 0 | A) \times P(y_6 = A)$$

$$P(y_1 = 0.38, y_2 = 0.52 | A) = \mathcal{N}\left(\begin{bmatrix} 0.38 \\ 0.52 \end{bmatrix} \middle| \mu = \begin{bmatrix} 0.24 \\ 0.52 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.0064 & -0.0096 \\ -0.0096 & 0.0336 \end{bmatrix}\right)$$

$$* \; \mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{m/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \times \Sigma^{-1} \times (x-\mu)\right)$$

$$|\Sigma| = 0.0064 \times 0.0336 - 0.0096 \times 0.0096 = 0.00012288$$

$$\Sigma^{-1} = \frac{1}{\underset{0.00012288}{|\Sigma|}} \begin{bmatrix} 0.0336 & -0.0096 \\ -0.0096 & 0.0064 \end{bmatrix} = \begin{bmatrix} 273.4375 & -78.125 \\ -78.125 & 52.083 \end{bmatrix}$$

$$X - \mu = \begin{bmatrix} 0.38 \\ 0.52 \end{bmatrix} - \begin{bmatrix} 0.24 \\ 0.52 \end{bmatrix} = \begin{bmatrix} 0.14 \\ 0 \end{bmatrix}$$

$$P(Y_1 = 0.38, Y_2 = 0.52 | A) = \frac{1}{(2\pi)^{2/2}\sqrt{0.00012288}} \exp\left(-\frac{1}{2} \begin{bmatrix} 0.14 \\ 0 \end{bmatrix}^T \begin{bmatrix} 273.4375 & -78.125 \\ -78.125 & 52.083 \end{bmatrix} \begin{bmatrix} 0.14 \\ 0 \end{bmatrix}\right)$$

$$= 14.3575 \exp\left(-\frac{1}{2} \times 5.3594\right)$$

$$\approx 0.9847$$

$$P(X8|A) \times P(A) = 0.9847 \times \frac{1}{3} \times \frac{1}{3} \times \frac{3}{7} = \underline{0.04689}$$

$$P(X8|B) \times P(B) = P(Y_1 = 0.38, Y_2 = 0.52 | B) \times P(Y_3 = 0, Y_4 = 1 | B) \times P(Y_5 = 0 | B) \times P(Y_6 = B)$$

$P(Y_1 = 0.38, Y_2 = 0.52 | B)$ = usando scipy.stats.multivariate_normal $= 1.9624$

$$P(X8|B) \times P(B) = 1.9624 \times \frac{1}{4} \times \frac{1}{4} \times \frac{4}{7} = 0.07009$$

$$P(X9|A) \times P(A) = P(Y_1 = 0.42, Y_2 = 0.59 | A) \times P(Y_3 = 0, Y_4 = 1 | A) \times P(Y_5 = 1 | A) \times P(A)$$

$P(Y_1 = 0.42, Y_2 = 0.59 | A)$ = usando scipy.stats.multivariate_normal $= 0.4031$

$$P(X9|A) \times P(A) = 0.4031 \times \frac{1}{3} \times \frac{1}{3} \times \frac{3}{7} = 0.0192$$

$$P(X9|B) \times P(B) = P(Y_1 = 0.42, Y_2 = 0.59 | B) \times P(Y_3 = 0, Y_4 = 1 | B) \times P(Y_5 = 1 | B) \times P(B)$$

$P(Y_1 = 0.42, Y_2 = 0.59 | B)$ = usando multivariate_normal de scipy.stats $= 1.7286$

$$P(X9|B) \times P(B) = 1.7286 \times \frac{1}{4} \times \frac{2}{4} \times \frac{4}{7} = 0.1235$$

Logo o argmax h_{MAP} de x_8 vai ser entre $P(x8|A) \times P(A) = 0.04689$ e $P(x8|B) \times P(B) = \underline{0.07009}$. Classificamos x_8 então como tendo output B.

O argmax h_{MAP} de x_9 vai ser entre $P(x9|A) \times P(A) = 0.0192$ e $P(x9|B) \times P(B) = 0.1235$. Classificamos x_9 então como tendo output B.

```python
# Este codigo foi usado para calcular P(Y1 = 0.38, Y2 = 0.52 | B)

import numpy as np
```

```python
from scipy.stats import multivariate_normal

mu = np.array([0.5925, 0.3275])
cov_matrix = np.array([[0.02289, -0.0097583], [-0.0097583, 0.03149]])

point = np.array([0.38, 0.52])

mvn = multivariate_normal(mean=mu, cov=cov_matrix)

pdf_value = mvn.pdf(point)

print("PDF at point (0.38, 0.52):", pdf_value)
```

```python
# Este codigo foi usado para calcular P(Y1 = 0.42, Y2 = 0.59 | A)

import numpy as np
from scipy.stats import multivariate_normal

mu = np.array([0.24, 0.52])
cov_matrix = np.array([[0.0064, 0.0096], [0.0096, 0.0336]])

point = np.array([0.42, 0.59])

mvn = multivariate_normal(mean=mu, cov=cov_matrix)

pdf_value = mvn.pdf(point)

print("PDF at point (0.42, 0.59):", pdf_value)
```

```python
# Este codigo foi usado para calcular P(Y1 = 0.42, Y2 = 0.59 | B)

import numpy as np
from scipy.stats import multivariate_normal

mu = np.array([0.5925, 0.3275])
cov_matrix = np.array([[0.02289, -0.0097583], [-0.0097583, 0.03149]])

point = np.array([0.42, 0.59])

mvn = multivariate_normal(mean=mu, cov=cov_matrix)

pdf_value = mvn.pdf(point)

print("PDF at point (0.42, 0.59):", pdf_value)
```

c)

c) Primeiro vamos calcular $P(A|x)$ sob um pressuposto de MAXIMUM LIKELIHOOD $= p(D|h)$.

$$p(D|h) = \frac{p(D|h) \times P(h)}{P(h)} \leftarrow \text{de b)}$$

$$\cancel{p(x8|A) = p(x8|A) \times p(A}$$

$$p(x8|A) = \frac{0.04689}{3/7} = 0.1094$$

$$p(x8|B) = \frac{0.07009}{4/7} = 0.1227$$

$$P(x9|A) = \frac{0.0192}{3/7} = 0.0448$$

$$P(x9|B) = \frac{0.1235}{4/7} = 0.2161$$

Vamos agora normalizar para garantir que $\sum_{h \in you t} p(h|x) = 1$

$$P(A|x8) = \frac{p(x8|A)}{p(x8|A) + p(x8|B)} = \frac{0.1094}{0.1094 + 0.1227} = 0.4713$$

$$P(A|x9) = \frac{P(x9|A)}{P(x9|A) + p(x9|B)} = \frac{0.0448}{0.0448 + 0.2161} = 0.1717$$

Para termos uma accuracy de 100% queremos que x8 seja classificado como A e que x9 seja classificado como B. Logo $P(A|x8) > \theta$ e $P(A|x9) \leq \theta$, ou seja, $P(A|x9) \leq \theta < P(A|x8)$

$\Rightarrow \theta \in [0.1717, 0.4713[$.

2. a)

2a) Como a discretização de y_2 tem de ser de equal range binário, vamos separar em:

$$\begin{cases} 0, & \text{se } 0 \le y_2 \le 0.5 \\ 1, & \text{se } 0.5 < y_2 \le 1 \end{cases}$$

Vamos dividir os observações em 3 folds, cada um com a mesma dimensão, sem shuffling

Fold 1

D	y_1	y_2	y_3	y_4	y_5	y_6
x_1	0.24	0	1	1	0	A
x_2	0.16	0	1	0	1	A
x_3	0.32	1	0	1	2	A

Fold 2

D	y_1	y_2	y_3	y_4	y_5	y_6
x_4	0.54	0	0	0	1	B
x_5	0.66	0	0	0	0	B
x_6	0.76	0	1	0	2	B

Fold 3

D	y_1	y_2	y_3	y_4	y_5	y_6
x_7	0.41	1	0	1	1	B
x_8	0.38	1	0	1	0	A
x_9	0.42	1	0	1	1	B

b)

b) Como as train observations têm os índices mais baixos, as nossas observações de teste vão ser as do fold 3 : x_7, x_8, x_9

Vamos então para cada uma destas observações a hamming distance (o número de elementos ~~em~~ nos quais os inputs diferem).

X7:

$H(x_i, x_j)$	x_1	x_2	x_3	x_4	x_5	x_6
x_7	4	4	2	2	3	4
			$\downarrow$ 0.32	$\downarrow$ 0.54	$\downarrow$ 0.66	

As 3 observações com menos distância vão ser x_3, x_4 e x_5.

Podemos então calcular a estimativa de ~~x8~~ x_7 para y_1 com weighted mean estimate $= \dfrac{\frac{1}{d_1} y_1 + \frac{1}{d_2} y_2 + \frac{1}{d_3} y_3}{\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3}}$

$$\hat{z} = \frac{\frac{1}{2} \times 0.32 + \frac{1}{2} \times 0.54 + \frac{1}{3} \times 0.66}{\frac{1}{2} + \frac{1}{2} + \frac{1}{3}} = 0.4875$$

X8:

$H(x_i, x_j)$	x_1	x_2	x_3	x_4	x_5	x_6
x_8	2	4	1	4	3	5
	$\downarrow$ 0.24		$\downarrow$ 0.32		$\downarrow$ 0.66	

$$\hat{z} = \frac{\frac{1}{2} \times 0.24 + \frac{1}{1} \times 0.32 + \frac{1}{3} \times 0.66}{\frac{1}{2} + \frac{1}{1} + \frac{1}{3}} = 0.36$$

X9:

$H(x_i, x_j)$	x_1	x_2	x_3	x_4	x_5	x_6
x_9	4	4	2	2	3	4
			$\downarrow$ 0.32	$\downarrow$ 0.54	$\downarrow$ 0.66	

$$\hat{z} = \frac{\frac{1}{2} \times 0.32 + \frac{1}{2} \times 0.54 + \frac{1}{3} \times 0.66}{\frac{1}{2} + \frac{1}{2} + \frac{1}{3}} = 0.4875$$

Vamos agora calcular a $MAE = \frac{1}{n} \sum_{k} |z_k - \hat{z}_k|$

$$X_7 = |z_7 - \hat{z}_7| = |0.41 - 0.4875| = 0.0775$$

$$X_8 = |z_8 - \hat{z}_8| = |0.38 - 0.361| = 0.02$$

$$X_9 = |z_9 - \hat{z}_9| = |0.42 - 0.4875| = 0.0675$$

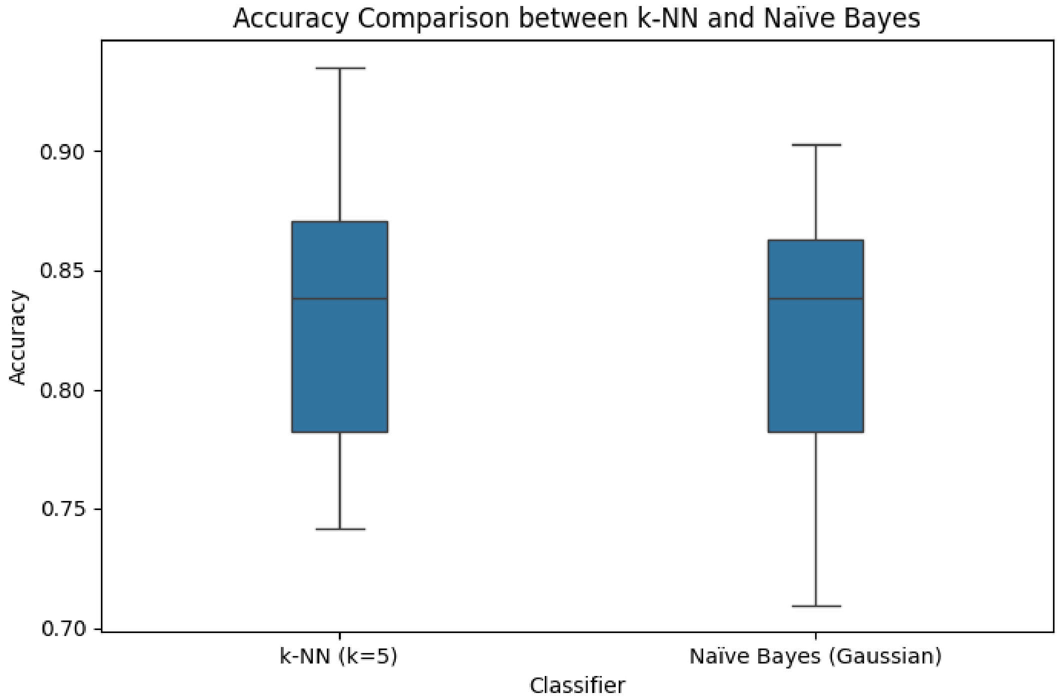
$$MAE = \frac{1}{3} \times (0.0775 + 0.02 + 0.0675) = 0.055$$

Part II: Programming

1. a)

```python
from scipy.io.arff import loadarff
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

data = loadarff('column_diagnosis.arff')

df = pd.DataFrame(data[0])

le = LabelEncoder()
df['class'] = le.fit_transform(df['class'])

x = df.drop('class', axis=1)
y = df['class']

cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)

knn_classifier = KNeighborsClassifier(n_neighbors=5)
nb_classifier = GaussianNB()

knn_accuracies = []
nb_accuracies = []

for train_idx, test_idx in cv.split(x, y):
    x_train, x_test = x.iloc[train_idx], x.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    knn_classifier.fit(x_train, y_train)
    knn_pred = knn_classifier.predict(x_test)
    knn_accuracy = accuracy_score(y_test, knn_pred)
    knn_accuracies.append(knn_accuracy)

    nb_classifier.fit(x_train, y_train)
    nb_pred = nb_classifier.predict(x_test)
    nb_accuracy = accuracy_score(y_test, nb_pred)
    nb_accuracies.append(nb_accuracy)

results_df = pd.DataFrame({'Classifier': ['k-NN (k=5)'] * 10 + ['Naive Bayes (
    Gaussian)'] * 10, 'Accuracy': knn_accuracies + nb_accuracies})

plt.figure(figsize=(8, 5))
sns.boxplot(x='Classifier', y='Accuracy', data=results_df, width=0.2)
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison between k-NN and Naive Bayes')
plt.savefig("Exercicio1.png")
plt.show()

_, p_value = stats.ttest_rel(knn_accuracies, nb_accuracies, alternative="
    greater")
```

```python

print("is k-NN statistically superior to Naive Bayes regarding accuracy? (p-
    value =", p_value, ")")
```



b) Ao fazer um teste de hipotese com

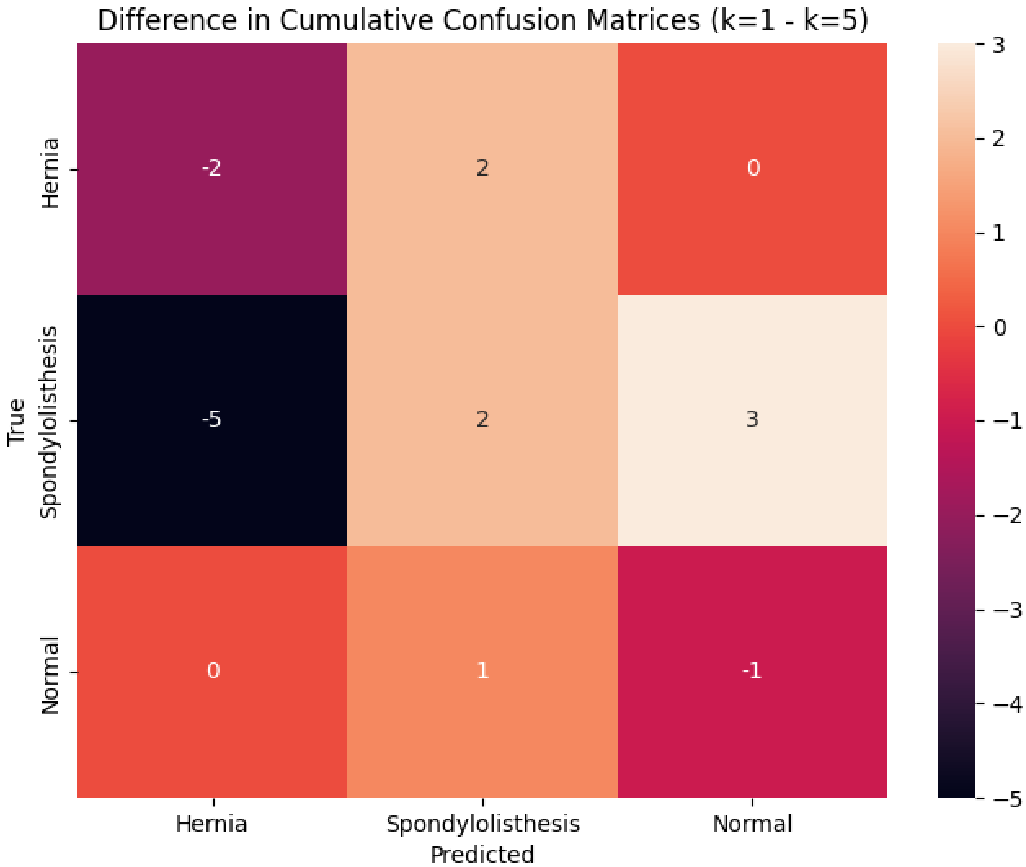
$$H_0 : accuracy_{kNN} = accuracy_{NaiveBayes} \tag{1}$$

$$H_1 : accuracy_{kNN} > accuracy_{NaiveBayes} \tag{2}$$

Obtivemos o p-value = 0.19042809062064092, o que significa que não rejeitamos H_0 para os valores de significância usuais (1%, 5%, 10%). Concluindo assim, que não é possível afirmar que kNN é estatisticamente superior a Naïve Bayes em termos de accuracy, sendo também impossível tirar outras conclusões sem mais testes de hipotese.

2.
```python
import numpy as np
from scipy.io.arff import loadarff
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

data = loadarff('column_diagnosis.arff')

df = pd.DataFrame(data[0])

le = LabelEncoder()
```

```python
df['class'] = le.fit_transform(df['class'])

x = df.drop('class', axis=1)
y = df['class']

cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)

# O KNeighborsClassifier ja usa a Euclidean distance e os pesos uniformes por
    default
knn1 = KNeighborsClassifier(n_neighbors=1)
knn5 = KNeighborsClassifier(n_neighbors=5)

cumulative_cm1 = np.zeros((3, 3))
cumulative_cm5 = np.zeros((3, 3))

for train_idx, test_idx in cv.split(x, y):
    x_train, x_test = x.iloc[train_idx], x.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    knn1.fit(x_train, y_train)
    knn5.fit(x_train, y_train)

    y_pred1 = knn1.predict(x_test)
    y_pred5 = knn5.predict(x_test)

    cm1 = confusion_matrix(y_test, y_pred1)
    cm5 = confusion_matrix(y_test, y_pred5)

    cumulative_cm1 += cm1
    cumulative_cm5 += cm5

diff_cm = cumulative_cm1 - cumulative_cm5

class_names = ["Hernia", "Spondylolisthesis", "Normal"]

plt.figure(figsize=(8, 6))
sns.heatmap(diff_cm, annot=True, fmt='g', cbar=True, xticklabels=class_names,
    yticklabels=class_names)
plt.title("Difference in Cumulative Confusion Matrices (k=1 - k=5)")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.savefig("Exercicio2.png")
plt.show()
```



Em cada célula da matriz da diferença entre matrizes de confusão é possível ver qual dos modelos tem mais observações para essa célula em específico: se for um valor positivo kNN com k = 1 tem mais observações, se for um valor negativo é o kNN com k = 5 e se for um valor nulo têm o mesmo número de observações. Ao analisar a matrix da differença entre as matrizes de confusão, concluiu-se que o kNN com k = 5 é melhor do que o kNN com k = 1 em termos de accuracy, isto é acertou mais vezes, já que a soma dos valores na diagonal é um número negativo (-2 + 2 - 1 = -1). Para além disso, também é possível observar que o kNN com k = 5 teve menos previsões incorretas do que o kNN com k = 1 uma vez que o total de previsões erradas é um número positivo (2 + 0 + 3 - 5 + 0 + 1 = 1). Por fim, é possível concluir que o kNN com k = 5 é ligeiramente melhor do que o kNN = 1 neste caso.

3. 1. Uma das características da utilização de Naïve Bayes é assumir que os dados são independentes entre si. Para este conjunto de dados isso pode não se verificar já que estamos a lidar com dados relativos à saúde. Isso torna-se ainda mais notável para o nosso conjunto porque alguns dados referem-se ao mesmo orgão, como é o caso de pelvic_incidence, pelvic_tilt e pelvic_radius.

   2. Naïve Bayes é normalmente utilizado com variáveis categoricas. No nosso caso todas as variáveis possuem dados numéricos continuos e podem não seguir uma distribuição Gaussiana como Naïve Bayes prevê. Desta forma a utilização de Naïve Bayes pode levar a previsões pouco precisas.

   3. Por fim, a utilização de Naïve Bayes também pode trazer problemas para conjuntos de dados que tenham as suas classes desbalanceadas. Como funciona através de probabilidades pode acabar por atribuir uma baixa probabilidade a uma classe que tenha uma frequência relativa baixa, mesmo sendo uma classe

importante. No nosso caso, de 309 amostras, apenas 59 se referem a Disk Hernia (aproximandamente 19%), 149 a Spondylolisthesis (aproximadamente 48%) e 99 a Normal (aproximandamente 32%).