Aprendizagem 2023
Homework II – Group 28

Gonçalo Bárias (ist1103124) & Raquel Braunschweig (ist1102624)

## **Part I**: Pen and Paper

Consider the following dataset ($y_3$ - $y_5$ are all categorical variables and the domain of $y_2$ is $[0, 1]$):

| $D$ | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ |
|---|---|---|---|---|---|---|
| $\mathbf{x}_1$ | 0.24 | 0.36 | 1 | 1 | 0 | A |
| $\mathbf{x}_2$ | 0.16 | 0.48 | 1 | 0 | 1 | A |
| $\mathbf{x}_3$ | 0.32 | 0.72 | 0 | 1 | 2 | A |
| $\mathbf{x}_4$ | 0.54 | 0.11 | 0 | 0 | 1 | B |
| $\mathbf{x}_5$ | 0.66 | 0.39 | 0 | 0 | 0 | B |
| $\mathbf{x}_6$ | 0.76 | 0.28 | 1 | 0 | 2 | B |
| $\mathbf{x}_7$ | 0.41 | 0.53 | 0 | 1 | 1 | B |
| $\mathbf{x}_8$ | 0.38 | 0.52 | 0 | 1 | 0 | A |
| $\mathbf{x}_9$ | 0.42 | 0.59 | 0 | 1 | 1 | B |

1. **Consider $x_1$ - $x_7$ to be training observations, $x_8$ - $x_9$ to be testing observations, $y_1$ - $y_5$ to be input variables and $y_6$ to be the target variable.**
   ***Hint*: you can use `scipy.stats.multivariate_normal` for multivariate distribution calculus**

   (a) **Learn a Bayesian classifier assuming: i) $\{y_1, y_2\}$, $\{y_3, y_4\}$ and $\{y_5\}$ sets of independent variables (e.g., $y_1 \perp\!\!\!\perp y_3$ yet $y_1 \not\perp\!\!\!\perp y_2$), and ii) $y_1 \times y_2 \in \mathbb{R}^2$ is normally distributed. Show all parameters (distributions and priors for subsequent testing).**

   As stated by the question prompt, variable sets $\{y_1, y_2\}$, $\{y_3, y_4\}$ and $\{y_5\}$ are independent. Since we have seven training observations, we will use those to train a Bayesian classifier.

   We will refer to the outcome, which can be A or B, as class.

   To estimate $P(\text{class}|y_1, y_2, y_3, y_4, y_5)$, we can use Bayes' theorem:

$$P(\text{class}|y_1, y_2, y_3, y_4, y_5) = \frac{P(y_1, y_2, y_3, y_4, y_5|\text{class}) \times P(\text{class})}{P(y_1, y_2, y_3, y_4, y_5)} \tag{1}$$

   Since we know $\{y_1, y_2\}$, $\{y_3, y_4\}$ and $\{y_5\}$ are independent, we can rewrite $P(y_1, y_2, y_3, y_4, y_5)$ as $P(y_1, y_2) \cdot P(y_3, y_4) \cdot P(y_5)$. Rewriting (1) with this, results in:

$$P(\text{class}|y_1, y_2, y_3, y_4, y_5) = \frac{P(y_1, y_2|\text{class})P(y_3, y_4|\text{class})P(y_5|\text{class}) \times P(\text{class})}{P(y_1, y_2)P(y_3, y_4)P(y_5)} \tag{2}$$

   Given a new observation $O$, we are able to classify it by calculating $P(\text{class}|O)$ for all classes and selecting the class with the highest probability as our prediction.

$$
\begin{aligned}
\hat{z} &= \underset{c\in\{A,B\}}{\arg\max}\ \{P(\mathrm{c}|O)\} \\
&= \underset{c\in\{A,B\}}{\arg\max}\ \left\{\frac{P(y_1,y_2|c)P(y_3,y_4|c)P(y_5|c)\times P(c)}{P(y_1,y_2)P(y_3,y_4)P(y_5)}\right\} \\
&= \underset{c\in\{A,B\}}{\arg\max}\ \{P(y_1,y_2|c)P(y_3,y_4|c)P(y_5|c)\times P(c)\} \qquad \text{(we can remove parameters that do not depend on } c\text{)}
\end{aligned}
\tag{3}
$$

We can now begin to compute these parameters.

**Note:** Even though $P(y_1, y_2)$, $P(y_3, y_4)$ and $P(y_5)$ are not necessary to apply the model, we will still calculate them for the sake of showing all parameters.

Calculating $P(\mathrm{A})$, $P(\mathrm{B})$ and all parameters involving $y_1$ through $y_5$ is straightforward, since they can be infered from the table.

We have 3 observations of A and 4 observations of B, out of a total of 7 training observations. Therefore,

$$P(\mathrm{A}) = \tfrac{3}{7} \quad P(\mathrm{B}) = \tfrac{4}{7}$$

In a similiar manner we can obtain the probabilities for $y_5$,

$$P(y_5 = 0) = \tfrac{2}{7} \quad P(y_5 = 1) = \tfrac{3}{7} \quad P(y_5 = 2) = \tfrac{2}{7}$$

Now for the conditional probabilities of $y_5$,

$$
\begin{aligned}
P(y_5 = 0\,|\mathrm{A}) = \tfrac{1}{3}, \quad P(y_5 = 1\,|\mathrm{A}) = \tfrac{1}{3}, \quad P(y_5 = 2\,|\mathrm{A}) = \tfrac{1}{3} \\
P(y_5 = 0\,|\mathrm{B}) = \tfrac{1}{4}, \quad P(y_5 = 1\,|\mathrm{B}) = \tfrac{2}{4}, \quad P(y_5 = 2\,|\mathrm{B}) = \tfrac{1}{4}
\end{aligned}
$$

For the four possible combinations of $y_3$ and $y_4$ we can follow the same logic as above,

$$
\begin{aligned}
P(y_3 = 0,\ y_4 = 0) = \tfrac{2}{7}, \quad P(y_3 = 0,\ y_4 = 1) = \tfrac{2}{7} \\
P(y_3 = 1,\ y_4 = 0) = \tfrac{2}{7}, \quad P(y_3 = 1,\ y_4 = 1) = \tfrac{1}{7}
\end{aligned}
$$

Finally, considering each class and the four possible combinations of $y_3$ and $y_4$, we can use the table to calculate the following:

$$
\begin{aligned}
P(y_3 = 0,\ y_4 = 0\,|\mathrm{A}) = \tfrac{0}{3}, \quad P(y_3 = 0,\ y_4 = 1\,|\mathrm{A}) = \tfrac{1}{3} \\
P(y_3 = 1,\ y_4 = 0\,|\mathrm{A}) = \tfrac{1}{3}, \quad P(y_3 = 1,\ y_4 = 1\,|\mathrm{A}) = \tfrac{1}{3} \\
P(y_3 = 0,\ y_4 = 0\,|\mathrm{B}) = \tfrac{2}{4}, \quad P(y_3 = 0,\ y_4 = 1\,|\mathrm{B}) = \tfrac{1}{4} \\
P(y_3 = 1,\ y_4 = 0\,|\mathrm{B}) = \tfrac{1}{4}, \quad P(y_3 = 1,\ y_4 = 1\,|\mathrm{B}) = \tfrac{0}{4}
\end{aligned}
$$

Calculating now the parameters related to the variable set $\{y_1, y_2\}$. We know that $(y_1, y_2)$ follows a Multivariate Gaussian Distribution. Therefore,

$$P\left((y_1, y_2)|\boldsymbol{\mu}, \Sigma\right) = \mathcal{N}\left((y_1, y_2)|\boldsymbol{\mu}, \Sigma\right) \tag{4}$$

We can use the observations we have to approximate a value for the 2-dimensional mean vector ($\boldsymbol{\mu}$) and the covariance matrix ($\Sigma$).

$$\boldsymbol{\mu} = \frac{1}{7}\sum_{i=1}^{7}\begin{bmatrix} y_{1,i} \\ y_{2,i} \end{bmatrix} = \frac{1}{7}\left(\begin{bmatrix} 0.24 \\ 0.36 \end{bmatrix} + \begin{bmatrix} 0.16 \\ 0.48 \end{bmatrix} + \begin{bmatrix} 0.32 \\ 0.72 \end{bmatrix} + \begin{bmatrix} 0.54 \\ 0.11 \end{bmatrix} + \begin{bmatrix} 0.66 \\ 0.39 \end{bmatrix} + \begin{bmatrix} 0.76 \\ 0.28 \end{bmatrix} + \begin{bmatrix} 0.41 \\ 0.53 \end{bmatrix}\right)$$
$$= \begin{bmatrix} 0.4414 \\ 0.41 \end{bmatrix}$$

$$\Sigma_{00} = \frac{1}{N-1}\sum_{i=1}^{N}(y_{1,i} - \mu_1)^2 = \frac{1}{7-1}\left[(0.24 - 0.4414)^2 + \cdots + (0.41 - 0.4414)^2\right] \approx 0.0491$$

$$\Sigma_{11} = \frac{1}{N-1}\sum_{i=1}^{N}(y_{2,i} - \mu_2)^2 = \frac{1}{7-1}\left[(0.36 - 0.41)^2 + \cdots + (0.53 - 0.41)^2\right] \approx 0.0375$$

$$\Sigma_{01} = \Sigma_{10} = \frac{1}{N-1}\sum_{i=1}^{N}(y_{1,i} - \mu_1)(y_{2,i} - \mu_2)$$
$$= \frac{1}{7-1}\left[(0.24 - 0.4414)(0.36 - 0.41) + \cdots + (0.41 - 0.4414)(0.53 - 0.41)\right]$$
$$\approx -0.0211$$

$$\Sigma = \begin{bmatrix} \Sigma_{00} & \Sigma_{10} \\ \Sigma_{01} & \Sigma_{11} \end{bmatrix} = \begin{bmatrix} 0.0491 & -0.0211 \\ -0.0211 & 0.0375 \end{bmatrix}$$

Therefore, $P(y_1, y_2) \sim \mathcal{N}\left((y_1, y_2)|\boldsymbol{\mu} = \begin{bmatrix} 0.4414 \\ 0.41 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.0491 & -0.0211 \\ -0.0211 & 0.0375 \end{bmatrix}\right)$.
We can repeat the process for both classes (A and B).
Starting with A:

$$\boldsymbol{\mu} = \frac{1}{3}\sum_{i=1}^{3}\begin{bmatrix} y_{1,i}|\mathrm{A} \\ y_{2,i}|\mathrm{A} \end{bmatrix} = \frac{1}{3}\left(\begin{bmatrix} 0.24 \\ 0.36 \end{bmatrix} + \begin{bmatrix} 0.16 \\ 0.48 \end{bmatrix} + \begin{bmatrix} 0.32 \\ 0.72 \end{bmatrix}\right) = \begin{bmatrix} 0.24 \\ 0.52 \end{bmatrix}$$

$$\Sigma_{00} = \frac{1}{N-1}\sum_{i=1}^{N}(y_{1,i}|\mathrm{A} - \mu_1)^2 = \frac{1}{3-1}\left[(0.24 - 0.24)^2 + \cdots + (0.32 - 0.24)^2\right] \approx 0.0064$$

$$\Sigma_{11} = \frac{1}{N-1}\sum_{i=1}^{N}(y_{2,i}|\mathrm{A} - \mu_2)^2 = \frac{1}{3-1}\left[(0.36 - 0.52)^2 + \cdots + (0.72 - 0.52)^2\right] \approx 0.0336$$

$$\Sigma_{01} = \Sigma_{10} = \frac{1}{N-1}\sum_{i=1}^{N}(y_{1,i}|\mathrm{A} - \mu_1)(y_{2,i}|\mathrm{A} - \mu_2)$$
$$= \frac{1}{3-1}\left[(0.24 - 0.24)(0.36 - 0.52) + \cdots + (0.32 - 0.24)(0.72 - 0.52)\right] \approx 0.0096$$

$$\Sigma = \begin{bmatrix} \Sigma_{00} & \Sigma_{10} \\ \Sigma_{01} & \Sigma_{11} \end{bmatrix} = \begin{bmatrix} 0.0064 & 0.0096 \\ 0.0096 & 0.0336 \end{bmatrix}$$

Therefore, $P(y_1, y_2|\mathrm{A}) \sim \mathcal{N}\left((y_1, y_2)|\boldsymbol{\mu} = \begin{bmatrix} 0.24 \\ 0.52 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.0064 & 0.0096 \\ 0.0096 & 0.0336 \end{bmatrix}\right)$.
And now the B:

$$\boldsymbol{\mu} = \frac{1}{4}\sum_{i=1}^{4}\begin{bmatrix} y_{1,i}|\mathrm{B} \\ y_{2,i}|\mathrm{B} \end{bmatrix} = \frac{1}{4}\left(\begin{bmatrix} 0.54 \\ 0.11 \end{bmatrix} + \begin{bmatrix} 0.66 \\ 0.39 \end{bmatrix} + \begin{bmatrix} 0.76 \\ 0.28 \end{bmatrix} + \begin{bmatrix} 0.41 \\ 0.53 \end{bmatrix}\right) = \begin{bmatrix} 0.5925 \\ 0.3275 \end{bmatrix}$$

$$\Sigma_{00} = \frac{1}{N-1}\sum_{i=1}^{N}(y_{1,i}|\mathrm{B} - \mu_1)^2 = \frac{1}{4-1}\left[(0.54 - 0.5925)^2 + \cdots + (0.41 - 0.5925)^2\right] \approx 0.0229$$

$$\Sigma_{11} = \frac{1}{N-1}\sum_{i=1}^{N}(y_{2,i}|\mathrm{B} - \mu_2)^2 = \frac{1}{4-1}\left[(0.11 - 0.3275)^2 + \cdots + (0.53 - 0.3275)^2\right] \approx 0.0315$$

$$\Sigma_{01} = \Sigma_{10} = \frac{1}{N-1}\sum_{i=1}^{N}(y_{1,i}|\mathrm{B} - \mu_1)(y_{2,i}|\mathrm{B} - \mu_2)$$

$$= \frac{1}{4-1}\left[(0.54 - 0.5925)(0.11 - 0.3275) + \cdots + (0.41 - 0.5925)(0.53 - 0.3275)\right] \approx -0.0098$$

$$\Sigma = \begin{bmatrix} \Sigma_{00} & \Sigma_{10} \\ \Sigma_{01} & \Sigma_{11} \end{bmatrix} = \begin{bmatrix} 0.0229 & -0.0098 \\ -0.0098 & 0.0315 \end{bmatrix}$$

Therefore, $P(y_1, y_2|\mathrm{B}) \sim \mathcal{N}\left((y_1, y_2)|\boldsymbol{\mu} = \begin{bmatrix} 0.5925 \\ 0.3275 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.0229 & -0.0098 \\ -0.0098 & 0.0315 \end{bmatrix}\right)$.

We now have all the parameters necessary to apply the Bayesian classifier to new observations.

(b) **Under a MAP assumption, classify each testing observation showing all of your calculus.**

Since we will need the $(y_1, y_2)$ pdf values for the following exercises, let's calculate them using `scipy.multivariate_normal`, just like it was suggested in the hint:

```
from scipy.stats import multivariate_normal

x8 = multivariate_normal.pdf([0.38, 0.52], mean=[0.4414, 0.41],
                             cov=[[0.0491, -0.0211], [-0.0211, 0.0375]])
x9 = multivariate_normal.pdf([0.42, 0.59], mean=[0.4414, 0.41],
                             cov=[[0.0491, -0.0211], [-0.0211, 0.0375]])

x8_A = multivariate_normal.pdf([0.38, 0.52], mean=[0.24, 0.52],
                               cov=[[0.0064, 0.0096], [0.0096, 0.0336]])
x8_B = multivariate_normal.pdf([0.38, 0.52], mean=[0.5925, 0.3275],
                               cov=[[0.0229, -0.0098], [-0.0098, 0.03149]])
x9_A = multivariate_normal.pdf([0.42, 0.59], mean=[0.24, 0.52],
                               cov=[[0.0064, 0.0096], [0.0096, 0.0336]])
x9_B = multivariate_normal.pdf([0.42, 0.59], mean=[0.5925, 0.3275],
                               cov=[[0.0229, -0.0098], [-0.0098, 0.03149]])
```

From the code above we get the following:

$$P(y_1 = 0.38, y_2 = 0.52) \approx 3.6225, \quad P(y_1 = 0.42, y_2 = 0.59) \approx 2.5387$$

$$P(y_1 = 0.38, y_2 = 0.52|A) \approx 0.9847, \quad P(y_1 = 0.38, y_2 = 0.52|B) \approx 1.9623$$
$$P(y_1 = 0.42, y_2 = 0.59|A) \approx 0.4031, \quad P(y_1 = 0.42, y_2 = 0.59|B) \approx 1.7285$$

Since we are under a Maximum a Posteriori assumption, we can just use the expression at (3) and replace the values for each of the testing observations ($x_8$ and $x_9$).

**Starting** with $x_8$:

$$P(y_1 = 0.38, y_2 = 0.52|A)P(y_3 = 0, y_4 = 1|A)P(y_5 = 0|A) \times P(A) = 0.9847 \times \frac{1}{3} \times \frac{1}{3} \times \frac{3}{7} \approx 0.04689$$
$$P(y_1 = 0.38, y_2 = 0.52|B)P(y_3 = 0, y_4 = 1|B)P(y_5 = 0|B) \times P(B) = 1.9623 \times \frac{1}{4} \times \frac{1}{4} \times \frac{4}{7} \approx 0.07008$$

$$\begin{aligned}\hat{z}_{x_8} &= \underset{c\in\{A,B\}}{\arg\max}\ \{P(y_1 = 0.38, y_2 = 0.52|c)P(y_3 = 0, y_4 = 1|c)P(y_5 = 0|c) \times P(c)\} \\ &= \arg\max\ \{P(y_1, y_2|A)P(y_3, y_4|A)P(y_5|A) \times P(A); P(y_1, y_2|B)P(y_3, y_4|B)P(y_5|B) \times P(B)\} \\ &= B\end{aligned}$$

And **now** with $x_9$:

$$P(y_1 = 0.42, y_2 = 0.59|A)P(y_3 = 0, y_4 = 1|A)P(y_5 = 1|A) \times P(A) = 0.4031 \times \frac{1}{3} \times \frac{1}{3} \times \frac{3}{7} \approx 0.0192$$
$$P(y_1 = 0.42, y_2 = 0.59|B)P(y_3 = 0, y_4 = 1|B)P(y_5 = 1|B) \times P(B) = 1.7285 \times \frac{1}{4} \times \frac{2}{4} \times \frac{4}{7} \approx 0.1235$$

$$\begin{aligned}\hat{z}_{x_9} &= \underset{c\in\{A,B\}}{\arg\max}\ \{P(y_1 = 0.42, y_2 = 0.59|c)P(y_3 = 0, y_4 = 1|c)P(y_5 = 1|c) \times P(c)\} \\ &= \arg\max\ \{P(y_1, y_2|A)P(y_3, y_4|A)P(y_5|A) \times P(A); P(y_1, y_2|B)P(y_3, y_4|B)P(y_5|B) \times P(B)\} \\ &= B\end{aligned}$$

**Therefore**, we conclude that under a MAP assumption, observations $x_8$ and $x_9$ will be classified with B and B, respectively.

(c) **Consider that the default decision threshold of $\theta = 0.5$ can be adjusted according to**

$$f(\mathbf{x}|\theta) = \begin{cases} A, & P(A|\mathbf{x}) > \theta \\ B, & \text{otherwise} \end{cases}$$

**Under a maximum likelihood assumption, what thresholds optimize testing accuracy?**

Since we know that $P(A|y_1, y_2, y_3, y_4, y_5) + P(A|y_1, y_2, y_3, y_4, y_5)$ must be equal to 1, we need to normalize the values. Therefore, we get these new updated values for the posteriors:

$$\begin{aligned}P(A|y_1, y_2, y_3, y_4, y_5) &= \frac{P(A|y_1, y_2, y_3, y_4, y_5)}{P(A|y_1, y_2, y_3, y_4, y_5) \ + \ P(B|y_1, y_2, y_3, y_4, y_5)} \\ P(B|y_1, y_2, y_3, y_4, y_5) &= \frac{P(B|y_1, y_2, y_3, y_4, y_5)}{P(B|y_1, y_2, y_3, y_4, y_5) \ + \ P(A|y_1, y_2, y_3, y_4, y_5)}\end{aligned} \tag{5}$$

**Note:** To make the equations easier to read, we simplified "$y_1 = 0.38,\ y_2 = 0.52,\ y_3 = 0,\ y_4 = 1,\ y_5 = 0$" by only writing "$x_8$" and "$y_1 = 0.42,\ y_2 = 0.59,\ y_3 = 0,\ y_4 = 1,\ y_5 = 1$" by only writing "$x_9$", given that those variable values characterize each observation.

On question 1.b) we calculated the following values:

$$P(A) = \tfrac{3}{7}, \quad P(B) = \tfrac{4}{7}$$

$$P(x_8|A) \times P(A) \approx 0.04689, \quad P(x_8|B) \times P(B) \approx 0.07008$$
$$P(x_9|A) \times P(A) \approx 0.0192, \quad P(x_9|B) \times P(B) \approx 0.1235$$

**Note:** According to the *FAQ*, we can use the **likelihood** as a **rough proxy for posteriors**. Therefore, we will proceed with the assumption that $P(A|x)$ is roughly equivalent to $P(x|A)$, implying that the likelihood will serve as an approximation for the posteriors

Hence why, we need to calculate the likelihoods:

$$P(x_8|A) = 0.10941, \quad P(x_8|B) = 0.12264$$
$$P(x_9|A) = 0.0448, \quad P(x_9|B) = 0.216125$$

To obtain the posteriori values, we simply substitute the previously calculated values into the formula at equation (5) to normalize them. As stated before, we are assuming $P(A|x) = P(x|A)$ and $P(B|x) = P(x|B)$:

$$P(A|x_8) = \frac{0.10941}{0.10941+0.12264} \approx 0.4715 \quad P(B|x_8) = \frac{0.12264}{0.12264+0.10941} \approx 0.5285$$
$$P(A|x_9) = \frac{0.0448}{0.0448+0.216125} \approx 0.1717 \quad P(B|x_9) = \frac{0.216125}{0.216125+0.0448} \approx 0.8283$$

As observed, we now have $P(A|x_8) + P(B|x_8) = 1$ and $P(A|x_9) + P(B|x_9) = 1$, indicating that we have obtained normalized probabilities.

Our goal is to get optimal testing accuracy of 100%. To accomplish this, we have to look into the accuracy formula, given by:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

For the fraction to be of value 1, our observations need to be only true positives and true negatives. Therefore, we requise observations $x_8$ and $x_9$ to be classified as A and B, respectively. This leads to the conclusion that the following inequalities, when solved will yield us the interval of values that the threshold ($\theta$) can take so that the accuracy is 100%.

$$P(A|x_8) > \theta \Rightarrow x_8 \text{ gets classified as A}$$
$$P(A|x_9) \leq \theta \Rightarrow x_9 \text{ gets classified as B}$$

$$P(A|x_9) \leq \theta < P(A|x_8) \Rightarrow 0.1717 \leq \theta < 0.4715 \Rightarrow \theta \in [0.1717, 0.4715[ \tag{6}$$

2. **Let $y_1$ be the target numeric variable, $y_2$ - $y_6$ be the input variables where $y_2$ is binarized under an equal-width (equal-range) discretization. For the evaluation of regressors, consider a 3-fold cross-validation over the full dataset ($x_1$ - $x_9$) without shuffling the observations.**

   (a) **Identify the observations and features per data fold after the binarization procedure.**

   To do the **binarization procedure** with an **equal-width discretization**, we need to divide $y_2$ into two intervals. Which are:

   $$\text{interval}_1 = [0, 0.5] \qquad \text{interval}_2 = [0.5, 1]$$

   **Here is the binarization of $y_2$** based on those intervals:

| $D$ | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ |
|---|---|---|---|---|---|---|
| $x_1$ | 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 |
| $x_4$ | 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 |
| $x_7$ | 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 |

**Subsequently**, we proceed to determine our folds. In this case, we will have three folds, each containing three observations. This choice aligns with the prompt instructions of using a 3-fold cross-validation approach.

$$\text{Fold}_1 = x_1\, x_2\, x_3 \qquad \text{Fold}_2 = x_4\, x_5\, x_6 \qquad \text{Fold}_3 = x_7\, x_8\, x_9$$

**So** our datasets will be:

**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) **Consider a distance-weighted *k*NN with k = 3, Hamming distance (*d*), and 1 / *d* weighting. Compute the MAE of this *k*NN regressor for the $1^{st}$ iteration of the cross-validation (i.e. train observations have the lower indices).**

The formula for the **weight** ($w_i$) and the **weighted average** ($f$), considering that k=3, are the following:

$$w_i(x_{new}) = \begin{cases} \frac{1}{d(x_{new}, x_i)}, & x_{new} \neq x_i \\ 1, & \text{otherwise} \end{cases} \quad \Bigg| \quad \hat{z}_{x_{new}} = f(x_{new}) = \frac{\sum_{i=1}^{n} w_i(x_{new}) f(x_i)}{\sum_{i=1}^{n} w_i(x_{new})} \tag{7}$$

And the formula for the **mean absolute error** is given by:

$$MAE(\hat{z}, z) = \frac{1}{n} \sum_{i=1}^{n} |z_i - \hat{z}_i| \tag{8}$$

As stated in the prompt, we will use Folds 1 and 2 for training, reserving Fold 3 for testing. Let's start by computing the **Hamming distances**, for $x_7$, $x_8$ and $x_9$:

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
|---|---|---|---|---|---|---|
| $H(x_7, x_j)$ | 4 | 4 | **2** | **2** | **3** | 4 |
| $H(x_8, x_j)$ | **2** | 4 | **1** | 4 | **3** | 5 |
| $H(x_9, x_j)$ | 4 | 4 | **2** | **2** | **3** | 4 |

Now let's determine the **three closest neighbors** (lowest Hamming distance values) for each test observation:

- For $x_7$ it is $x_3$, $x_4$ and $x_5$
- For $x_8$ it is $x_1$, $x_3$ and $x_5$
- For $x_9$ it is $x_3$, $x_4$ and $x_5$

The next step is calculating their **weighted average**. By replacing the formula on (7), we get the following values:

$$\hat{z}_{x_7} = f(x_7) = \frac{\frac{1}{2} \cdot 0.32 + \frac{1}{2} \cdot 0.54 + \frac{1}{3} \cdot 0.66}{\frac{1}{2} + \frac{1}{2} + \frac{1}{3}} = 0.4875$$

$$\hat{z}_{x_8} = f(x_8) = \frac{\frac{1}{2} \cdot 0.24 + \frac{1}{1} \cdot 0.32 + \frac{1}{3} \cdot 0.66}{\frac{1}{2} + \frac{1}{1} + \frac{1}{3}} = 0.36$$

$$\hat{z}_{x_9} = f(x_9) = \frac{\frac{1}{2} \cdot 0.32 + \frac{1}{2} \cdot 0.54 + \frac{1}{3} \cdot 0.66}{\frac{1}{2} + \frac{1}{2} + \frac{1}{3}} = 0.4875$$

Now we have the estimates for the testing observations along with their real values:

$$\hat{z} = \left(\hat{z}_{x_7}, \hat{z}_{x_8}, \hat{z}_{x_9}\right) = (0.4875, 0.36, 0.4875)$$
$$z = (0.41, 0.38, 0.42)$$

**Finally,** let's compute the mean absolute error by using the formula on (8):

$$MAE(\hat{z}, z) = \frac{1}{3}\left(|0.41 - 0.4875| + |0.38 - 0.36| + |0.42 - 0.4875|\right) = 0.055$$

## **Part II**: Programming and critical analysis

Considering the `column_diagnosis.arff` dataset available at the course webpage's homework tab. Using `sklearn`, apply a 10-fold stratified cross-validation with shuffling (`random_state=0`) for the assessment of predictive models along this section.

1. **Compare the performance of *k*NN with k = 5 and Naïve Bayes with Gaussian assumption (consider all remaining parameters for each classifier as `sklearn`'s default):**

   (a) **Plot two boxplots with the fold accuracies for each classifier.**

```
import matplotlib.pyplot as plt, pandas as pd
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from scipy.io.arff import loadarff

```

```
# Read the ARFF file and prepare data
data = loadarff("./data/column_diagnosis.arff")
df = pd.DataFrame(data[0])
df["class"] = df["class"].str.decode("utf-8")
X, y = df.drop("class", axis=1), df["class"]

# Define cross-validation strategy
folds = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)

# Initialize classifiers
knn_predictor = KNeighborsClassifier(n_neighbors=5)
nb_predictor = GaussianNB()

# Evaluate classifiers
knn_accs = cross_val_score(knn_predictor, X, y, cv=folds, scoring="accuracy")
nb_accs = cross_val_score(nb_predictor, X, y, cv=folds, scoring="accuracy")

# Plot boxplots
plt.figure(figsize=(7, 5))
b_plot = plt.boxplot(
    [knn_accs, nb_accs], patch_artist=True, labels=["kNN", "Naive Bayes"]
)

colors = ["#f8766d", "#00bfc4"]
for patch, color in zip(b_plot["boxes"], colors):
    patch.set_facecolor(color)
for median in b_plot["medians"]:
    median.set_color("black")

plt.ylabel("Accuracy")
plt.grid(axis="y")
plt.show()
```

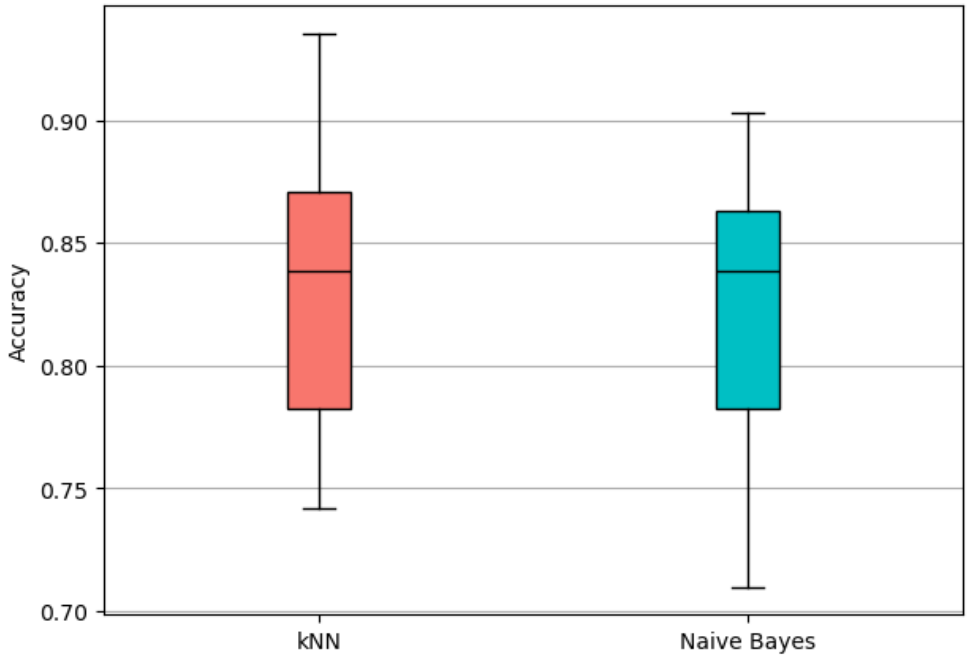


Figure 1: Boxplots with the fold accuracies of *k*NN (k = 5) and Naïve Bayes

(b) **Using `scipy`, test the hypothesis "*k*NN is statistically superior to Naïve Bayes regarding accuracy", asserting whether is true.**

We will consider the null hypothesis and alternate hypothesis below and perform a right-tailed test using the accuracies obtained in the previous answer,

$$H_0 : \text{accuracy}_{k\text{NN}} = \text{accuracy}_{\text{Naïve Bayes}}$$
$$H_1 : \text{accuracy}_{k\text{NN}} > \text{accuracy}_{\text{Naïve Bayes}}$$

```
from scipy.stats import ttest_rel

# Is kNN better than Naive Bayes?
res = ttest_rel(knn_accs, nb_accs, alternative="greater")
print("Is kNN > Naive Bayes? pval =", res.pvalue)
```

Using `scipy` we get a p-value of, approximately, 0.190428 = 19.0428 %.

This means we cannot reject the hypothesis $H_0$ at common significance levels (1%, 5% and 10%).

**Therefore,** we cannot assert that *k*NN is statistically superior to Naïve Bayes. We also cannot state that the hypothesis on the statement is outright false without checking other statistical tests.

2. **Consider two *k*NN predictors with k = 1 and k = 5 (uniform weights, Euclidean distance, all remaining parameters as default). Plot the differences between the two cumulative confusion matrices of the predictors. Comment.**

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

```
6
7  # Read the ARFF file and prepare data
8  data = loadarff("./data/column_diagnosis.arff")
9  df = pd.DataFrame(data[0])
10 df["class"] = df["class"].str.decode("utf-8")
11 X, y = df.drop("class", axis=1), df["class"]
12
13 # Initialize StratifiedKFold with 10 folds and shuffling
14 folds = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)
15
16 # Create kNN classifiers with k=1 and k=5
17 knn_1 = KNeighborsClassifier(n_neighbors=1)
18 knn_5 = KNeighborsClassifier(n_neighbors=5)
19
20 labels = ["Hernia", "Normal", "Spondylolisthesis"]
21 cm_1, cm_5 = np.zeros((3, 3)), np.zeros((3, 3))
22 for train_k, test_k in folds.split(X, y):
23     X_train, X_test = X.iloc[train_k], X.iloc[test_k]
24     y_train, y_test = y.iloc[train_k], y.iloc[test_k]
25
26     # Fit kNN classifiers and assess
27     knn_1.fit(X_train, y_train)
28     knn_5.fit(X_train, y_train)
29     knn_1_pred, knn_5_pred = knn_1.predict(X_test), knn_5.predict(X_test)
30     cm_1 += np.array(confusion_matrix(y_test, knn_1_pred, labels=labels))
31     cm_5 += np.array(confusion_matrix(y_test, knn_5_pred, labels=labels))
32
33 # Calculate cumulative confusion matrices
34 cm_diff = cm_1 - cm_5
35 cm_diff_df = pd.DataFrame(cm_diff, index=labels, columns=labels)
36
37 # Plot the differences
38 plt.figure(figsize=(9, 7))
39 sns.heatmap(
40     cm_diff_df, cmap="Purples", annot=True, annot_kws={"fontsize": 14}, fmt="g"
41 )
42 plt.xlabel("Predicted")
43 plt.ylabel("Real")
44 plt.show()
```

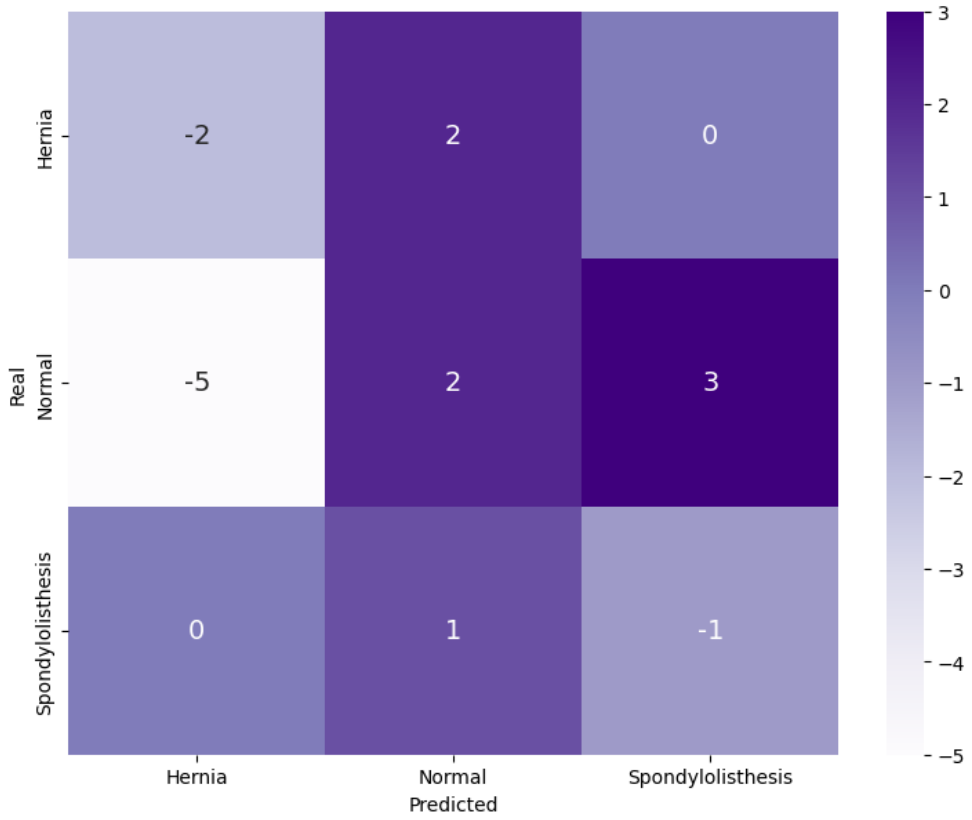


Figure 2: Confusion Matrix Differences Between k=1 and k=5 k-Nearest Neighbors (*k*NN) Classifiers

Upon examination of the difference matrix derived from the cumulative confusion matrices, it is evident that the *k*NN model with k=5 surpasses the performance of k=1.

For each cell in this matrix, we can determine which model (k=1 or k=5) had the most observations by noting if the cell value is negative or positive. A positive value indicates that k=1 had the most observations, while a negative value indicates that k=5 was the one with more observations.

This superiority of the k=5 model is shown by the negative sum of the main diagonal elements, signifying higher accuracy. Additionally, the fact that the sum of incorrect predictions (false positives and false negatives) is positive suggests that k=5 has fewer miss classifications.

**Therefore**, utilizing k=5 appears to be the preferable choice over k=1 for this task.

3. **Considering the unique properties of `column_diagnosis`, identify three possible difficulties of Naïve Bayes when learning from the given dataset.**

   Here are three possible difficulties of Naïve Bayes when learning from the given dataset, in no particular order:

   - To apply Naïve Bayes, we assume that all variables were independent of one another, which might not be case, thus explaining a potential problem that would hinder its accuracy.
   - There may be an inadequacy of Gaussian assumption, because the data used (`column_diagnosis.arff`) is not very big (around 300 observations, with only some being used for training), and so there is not much data for probability density function/probability mass function approximations.

- Probabilities are estimated based on the number of occurences on the given training data, and so there might be imbalanced number of occurences for some given class. This might affect the value of the prior, creating biases in a Maximum a Priori assumption.