Aprendizagem 2023
Homework I — Group 003
(ist1107028, ist1107137)

Part I: Pen and paper

We collected four positive (P) observations,

$$\{x_1 = (A, 0), \quad x_2 = (B, 1), \quad x_3 = (A, 1), \quad x_4 = (A, 0)\}$$

and four negative (N) observations,

$$\{x_5 = (B, 0), \quad x_6 = (B, 0), \quad x_7 = (A, 1), \quad x_8 = (B, 1)\}$$

Consider the problem of classifying observations as positive or negative.

1. **Compute the F1-measure of a kNN with k = 5 and Hamming distance using a leave-one-out evaluation schema. Show all calculus.**

   We start by calculating Hamming distance between observations. The Hamming distance is the number of positions at which the corresponding symbols are different.
   Since we are workin with $k = 5$, we will consider the 5 nearest neighbors of each observation.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	-	2	**1**	**0**	**1**	**1**	**1**	2
x_2	2	-	**1**	2	**1**	**1**	**1**	**0**
x_3	**1**	**1**	-	**1**	2	2	**0**	**1**
x_4	**0**	2	**1**	-	**1**	**1**	**1**	2
x_5	**1**	**1**	2	**1**	-	**0**	2	**1**
x_6	**1**	**1**	2	**1**	**0**	-	2	**1**
x_7	**1**	**1**	**0**	**1**	2	2	-	**1**
x_8	2	**0**	**1**	2	**1**	**1**	**1**	-

Table 1: Hamming distance between observations, $k = 5$

Now that we have the Hamming distance between all observations, we must identify if the prediction is correct or not. We will consider the majority class of the 5 nearest neighbors for each observation.

Example: For x_1, the 5 nearest neighbors are x_3 and x_4 (which are positive), x_5, x_6 and x_7 (which are negative). The majority class is negative, therefore the prediction is incorrect.

We apply the same logic for the rest of the classes, ending up with the following table:

Observation	True Value	Mode	Prediction	Confusion Matrix Terminology
x_1	P	$\{2P, 3N\}$	N	FN
x_2	P	$\{1P, 4N\}$	N	FN
x_3	P	$\{3P, 2N\}$	P	TP
x_4	P	$\{2P, 3N\}$	N	FN
x_5	N	$\{3P, 2N\}$	P	FP
x_6	N	$\{3P, 2N\}$	P	FP
x_7	N	$\{4P, 1N\}$	P	FP
x_8	N	$\{2P, 3N\}$	N	TN

P - Positive observation; N - Negative observation

TP - True Positive; TN - True Negative; FP - False Positive; FN - False Negative

Table 2: Predictions for each observation, $k = 5$

With this table, we can know calculate the Precision, Recall and F1-measure using the following formulas:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{1}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2}$$

$$\text{F1-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Replacing the corresponding values in the formulas, we get:

for Precision (1) and Recall (2):

$$\text{Precision} = \frac{1}{1+3} = 0.25 \qquad \text{Recall} = \frac{1}{1+3} = 0.25$$

F1-measure (3):

$$\text{F1-measure} = 2 \times \frac{0.25 \times 0.25}{0.25 + 0.25} = 0.25$$

2. **Propose a new metric (distance) that improves the latter's performance (i.e., the F1-measure) by three fold.**

To improve the F1-measure, we propose using $k = 3$ instead of $k = 5$ and to atribute more weight to the difference between letters.

Example: When calculating the Hamming distance, if the corresponding y_1 values (letters) differ, it counts as 2 instead of 1. For y_2, it counts as 1, as in the previous exercise. Below is how we calculated some of the distances to conclude the example:

$$d(x_1, x_2) = 2 + 1 = 3 \quad \text{(both } y_1 \text{ and } y_2 \text{ differ)}$$
$$d(x_1, x_3) = 1 \quad \text{(only } y_2 \text{ differs)}$$
$$d(x_1, x_5) = 2 \quad \text{(only } y_1 \text{ differs)}$$

We apply this logic to the rest of the distances, endind up with the next table, where the colored values are the $k = 3$ nearest neighbors:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	-	3	**1**	**0**	2	2	**1**	3
x_2	3	-	2	3	**1**	**1**	2	**0**
x_3	**1**	2	-	**1**	3	3	**0**	2
x_4	**0**	3	**1**	-	2	2	**1**	3
x_5	2	**1**	3	2	-	**0**	3	**1**
x_6	2	**1**	3	2	**0**	-	3	**1**
x_7	**1**	2	**0**	**1**	3	3	-	2
x_8	3	**0**	2	3	**1**	**1**	2	-

Table 3: Hamming distance between observations, $k = 3$

Simply by looking at this, we can produce a new prediton table, like in the first exercise:

Observation	True Value	Mode	Prediction	Confusion Matrix Terminology
x_1	P	$\{2P, 1N\}$	P	TP
x_2	P	$\{0P, 3N\}$	N	FN
x_3	P	$\{2P, 1N\}$	P	TP
x_4	P	$\{2P, 1N\}$	P	TP
x_5	N	$\{1P, 2N\}$	N	TN
x_6	N	$\{1P, 2N\}$	N	TN
x_7	N	$\{3P, 0N\}$	P	FP
x_8	N	$\{1P, 2N\}$	N	TN

P - Positive observation; N - Negative observation

TP - True Positive; TN - True Negative; FP - False Positive; FN - False Negative

Table 4: Predictions for each observation, $k = 3$

We can now calculate the Precision (1), Recall (2) and F1-measure (3):

$$\text{Precision} = \frac{3}{3+1} = 0.75 \qquad \text{Recall} = \frac{3}{3+1} = 0.75$$

$$\text{F1-measure} = 2 \times \frac{0.75 \times 0.75}{0.75 + 0.75} = 0.75 = 3 \times 0.25$$

An additional positive observation was acquired, $\boldsymbol{x_9 = (B, 0)}$, and a third variable $\boldsymbol{y_3}$ was independently monitored, yielding estimates,

$$\boldsymbol{y_3|P = \{1.1, 0.8, 0.5, 0.9, 0.8\} \quad and \quad y_3|N = \{1, 0.9, 1.2, 0.9\}}$$

3. **Considering the nine training observations, learn a Bayesian classifier assuming: (i) $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$ are dependent; (ii) $\boldsymbol{y_1}$, $\boldsymbol{y_2}$ and $\boldsymbol{y_3}$ variable sets are independent and equally important; and (iii) $\boldsymbol{y_3}$ is normally distributed. Show all parameters.**

With the nine training observations, we can calculate the parameters for the Bayesian classifier. We will refer to the outcome, which can be positive or negative, P and N respectively, as 'class' or 'c'.

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

$$P(\text{class}|y_1, y_2, y_3) = \frac{P(y_1, y_2, y_3|\text{class}) \times P(\text{class})}{P(y_1, y_2, y_3)} \tag{4}$$

Since we know $\{y_1, y_2\}$ and $\{y_3\}$ are independent, we can rewrite $P(y_1, y_2, y_3)$ as $P(y_1, y_2) \times P(y_3)$. Rewriting (4) with this, results in:

$$P(\text{class}|y_1, y_2, y_3) = \frac{P(y_1, y_2|\text{class}) \times P(y_3|\text{class}) \times P(\text{class})}{P(y_1, y_2) \times P(y_3)} \tag{5}$$

We can now begin to compute these parameters.

$$\text{Priors:} \qquad P(P) = \frac{5}{9} \qquad P(N) = 1 - P(P) = \frac{4}{9}$$

To get the likelihoods, we start by calculating the **PMFs**:

$$\begin{aligned}
P((y_1 = A, y_2 = 0)|P) &= \frac{2}{5} \quad P((y_1 = A, y_2 = 0)|N) &= \frac{0}{4} \\
P((y_1 = A, y_2 = 1)|P) &= \frac{1}{5} \quad P((y_1 = A, y_2 = 1)|N) &= \frac{1}{4} \\
\\
P((y_1 = B, y_2 = 0)|P) &= \frac{1}{5} \quad P((y_1 = B, y_2 = 0)|N) &= \frac{2}{4} \\
P((y_1 = B, y_2 = 1)|P) &= \frac{1}{5} \quad P((y_1 = B, y_2 = 1)|N) &= \frac{1}{4}
\end{aligned}$$

And now the **PDFs**:

$$\begin{aligned} P((y_1 = A, y_2 = 0)) &= \frac{2}{9} \quad P((y_1 = B, y_2 = 0)) &= \frac{3}{9} \\ P((y_1 = A, y_2 = 1)) &= \frac{2}{9} \quad P((y_1 = B, y_2 = 1)) &= \frac{2}{9} \end{aligned}$$

For the y_3, we know that it is normally distributed, so we start by calculating the mean and variance for each class:

$$\textbf{mean:} \quad \mu = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad\qquad \textbf{variance:} \quad \sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \mu)^2$$

Positive class:

$\mu_P = \frac{1.1+0.8+0.5+0.9+0.8}{5} = 0.82$

$\sigma_P^2 = \frac{(1.1-0.82)^2+(0.8-0.82)^2+(0.5-0.82)^2+(0.9-0.82)^2+(0.8-0.82)^2}{4} = 0.047$

Negative class:

$\mu_N = \frac{1+0.9+1.2+0.9}{4} = 1.0$

$\sigma_N^2 = \frac{(1-1)^2+(0.9-1)^2+(1.2-1)^2+(0.9-1)^2}{3} = 0.02$

Global:

$\mu = \frac{1.1+0.8+0.5+0.9+0.8+1+0.9+1.2+0.9}{9} = 0.9$

$\sigma^2 = \frac{(1.1-0.9)^2+(0.8-0.9)^2+(0.5-0.9)^2+(0.9-0.9)^2+(0.8-0.9)^2+(1-0.9)^2+(0.9-0.9)^2+(1.2-0.9)^2+(0.9-0.9)^2}{8} = 0.04$

With this, we get:

$$\mathcal{N}(\mu_{y_3} = 0.82, \sigma_{y_3}^2 = 0.47|P) \quad \mathcal{N}(\mu_{y_3} = 1, \sigma_{y_3}^2 = 0.02|N) \quad \mathcal{N}(\mu_{y_3} = 0.9, \sigma_{y_3}^2 = 0.04)$$

The normal distribution formula is given by:

$$P(y_z|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_z - \mu)^2}{2\sigma^2}\right) \tag{6}$$

Having the parameters for the normal distribution, we calculate the likelihoods for y_3 as per (6):

Positive class:

$$\begin{aligned}
P(y_3 = 1.1|\mu_P, \sigma_P^2) &= \frac{1}{\sqrt{2\pi \times 0.47}} \exp\left(-\frac{(1.1-0.82)^2}{2 \times 0.47}\right) \approx 0.535 \\
P(y_3 = 0.8|\mu_P, \sigma_P^2) &= \frac{1}{\sqrt{2\pi \times 0.47}} \exp\left(-\frac{(0.8-0.82)^2}{2 \times 0.47}\right) \approx 0.582 \\
P(y_3 = 0.5|\mu_P, \sigma_P^2) &= \frac{1}{\sqrt{2\pi \times 0.47}} \exp\left(-\frac{(0.5-0.82)^2}{2 \times 0.47}\right) \approx 0.522 \\
P(y_3 = 0.9|\mu_P, \sigma_P^2) &= \frac{1}{\sqrt{2\pi \times 0.47}} \exp\left(-\frac{(0.9-0.82)^2}{2 \times 0.47}\right) \approx 0.578
\end{aligned}$$

Negative class:

$$\begin{aligned}
P(y_3 = 1|\mu_N, \sigma_N^2) &= \frac{1}{\sqrt{2\pi \times 0.02}} \exp\left(-\frac{(1-1)^2}{2 \times 0.02}\right) \approx 2.821 \\
P(y_3 = 0.9|\mu_N, \sigma_N^2) &= \frac{1}{\sqrt{2\pi \times 0.02}} \exp\left(-\frac{(0.9-1)^2}{2 \times 0.02}\right) \approx 2.197 \\
P(y_3 = 1.2|\mu_N, \sigma_N^2) &= \frac{1}{\sqrt{2\pi \times 0.02}} \exp\left(-\frac{(1.2-1)^2}{2 \times 0.02}\right) \approx 1.038
\end{aligned}$$

Global:

$$\begin{aligned}
P(y_3 = 1.1|\mu, \sigma^2) &= \frac{1}{\sqrt{2\pi \times 0.04}} \exp\left(-\frac{(1.1-0.9)^2}{2 \times 0.04}\right) = 0.5 \\
P(y_3 = 0.8|\mu, \sigma^2) &= \frac{1}{\sqrt{2\pi \times 0.04}} \exp\left(-\frac{(0.8-0.9)^2}{2 \times 0.04}\right) = 0.125 \\
P(y_3 = 0.5|\mu, \sigma^2) &= \frac{1}{\sqrt{2\pi \times 0.04}} \exp\left(-\frac{(0.5-0.9)^2}{2 \times 0.04}\right) = 2 \\
P(y_3 = 0.9|\mu, \sigma^2) &= \frac{1}{\sqrt{2\pi \times 0.04}} \exp\left(-\frac{(0.9-0.9)^2}{2 \times 0.04}\right) = 0 \\
P(y_3 = 1.0|\mu, \sigma^2) &= \frac{1}{\sqrt{2\pi \times 0.04}} \exp\left(-\frac{(1.0-0.9)^2}{2 \times 0.04}\right) = 0.125 \\
P(y_3 = 1.2|\mu, \sigma^2) &= \frac{1}{\sqrt{2\pi \times 0.04}} \exp\left(-\frac{(1.2-0.9)^2}{2 \times 0.04}\right) = 1.125
\end{aligned}$$

Now we have all the parameters to apply the Bayesian classifier to new observations.

Consider now three testing observations,

$$\boldsymbol{\{(A, 1, 0.8), (B, 1, 1), (B, 0, 0.9)\}}$$

4. **Under a MAP assumption, classify each testing observation showing all your calculus.**

MAP (Maximum A Posteriori) is defined as:

$$\begin{aligned}\hat{z} &= \underset{c_i}{\arg\max} \; \{P(c_i|x)\} \\ &= \underset{c_i}{\arg\max} \; \left\{\frac{P(x|c_i) \times P(c_i)}{P(x)}\right\} \\ &= \underset{c_i}{\arg\max} \; \{P(x|c_i) \times P(c_i)\} \quad \text{Denominator will be the same for all calculated values.}\end{aligned} \tag{7}$$

To apply this assumption we use the likelihoods and the priors calculated in the lattest exercise. We must also calculate the missing values needed - $P(y_3 = 0.8|\mu_N, \sigma_N^2)$ and $P(y_3 = 1|\mu_P, \sigma_P^2)$.

$$\begin{aligned}P(y_3 = 0.8|\mu_N, \sigma_N^2) &= \frac{1}{\sqrt{2\pi \times 0.02}} \exp\left(-\frac{(0.8-1)^2}{2 \times 0.02}\right) \approx 2.821 \\ P(y_3 = 1|\mu_P, \sigma_P^2) &= \frac{1}{\sqrt{2\pi \times 0.47}} \exp\left(-\frac{(1-0.82)^2}{2 \times 0.47}\right) \approx 0.535\end{aligned}$$

With this, we can replace the values in the MAP formula (7) for each observation:

$\boxed{(A, 1, 0.8)}$

$$\begin{aligned}P(y_1 = A, y_2 = 1, y_3 = 0.8|P) &= P(y_1 = A, y_2 = 1|\text{P}) \times P(y_3 = 0.8|\text{P}) \times P(P) \\ &= \frac{1}{5} \times 0.582 \times \frac{5}{9} \approx 0.065\end{aligned}$$

$$\begin{aligned}P(y_1 = A, y_2 = 1, y_3 = 0.8|N) &= P(y_1 = A, y_2 = 1|\text{N}) \times P(y_3 = 0.8|\text{N}) \times P(N) \\ &= \frac{1}{4} \times 2.197 \times \frac{4}{9} \approx 0.488\end{aligned}$$

$$\begin{aligned}\hat{z}_{(A,1,0.8)} &= \underset{c \in \{P,N\}}{\arg\max} \; \{P(y_1 = A, y_2 = 1|c)P(y_3 = 0.8|c) \times P(c)\} \\ &= \arg\max \; \{P(y_1, y_2|P)P(y_3|P) \times P(P); P(y_1, y_2|N)P(y_3|N) \times P(N)\} \\ &= N\end{aligned}$$

$\boxed{(B, 1, 1)}$

$$\begin{aligned}P(y_1 = B, y_2 = 1, y_3 = 1|P) &= P(y_1 = B, y_2 = 1|\text{P}) \times P(y_3 = 1|\text{P}) \times P(P) \\ &= \frac{1}{5} \times 2.821 \times \frac{5}{9} \approx 0.313\end{aligned}$$

$$\begin{aligned}P(y_1 = B, y_2 = 1, y_3 = 1|N) &= P(y_1 = B, y_2 = 1|\text{N}) \times P(y_3 = 1|\text{N}) \times P(N) \\ &= \frac{1}{4} \times 0.535 \times \frac{4}{9} \approx 0.059\end{aligned}$$

$$\begin{aligned}\hat{z}_{(B,1,1)} &= \underset{c\in\{P,N\}}{\arg\max}\ \{P(y_1=B,y_2=1|c)P(y_3=1|c)\times P(c)\}\\ &= \arg\max\ \{P(y_1,y_2|P)P(y_3|P)\times P(P);P(y_1,y_2|N)P(y_3|N)\times P(N)\}\\ &= P\end{aligned}$$

$\boxed{(B,0,0.9)}$

$$\begin{aligned}P(y_1=B,y_2=0,y_3=0.9|P) &= P(y_1=B,y_2=0|\mathrm{P})\times P(y_3=0.9|\mathrm{P})\times P(P)\\ &= \frac{1}{5}\times 0.578\times\frac{5}{9}\approx 0.064\end{aligned}$$

$$\begin{aligned}P(y_1=B,y_2=0,y_3=0.9|N) &= P(y_1=B,y_2=0|\mathrm{N})\times P(y_3=0.9|\mathrm{N})\times P(N)\\ &= \frac{2}{4}\times 2.197\times\frac{4}{9}\approx 0.488\end{aligned}$$

$$\begin{aligned}\hat{z}_{(B,0,0.9)} &= \underset{c\in\{P,N\}}{\arg\max}\ \{P(y_1=B,y_2=0|c)P(y_3=0.9|c)\times P(c)\}\\ &= \arg\max\ \{P(y_1,y_2|P)P(y_3|P)\times P(P);P(y_1,y_2|N)P(y_3|N)\times P(N)\}\\ &= N\end{aligned}$$

Therefore, the predictions for each observation are N, P and N respectively.

At last, consider only the following sentences and their respective connotations,

$$\boldsymbol{\{(\text{"}\textit{Amazing run}\text{"},P),(\text{"}\textit{I like it}\text{"},P),(\text{"}\textit{Too tired}\text{"},N),(\text{"}\textit{Bad run}\text{"},N)\}}$$

5. **Using a naïve Bayes under a ML assumption, classify the new sentence "I like to run". For the likelihoods calculation consider the following formula,**

$$\boldsymbol{P(T_i|c)=\frac{freq(t_i)+1}{N_c+V}}$$

where $\mathbf{t_i}$ represents a certain term i, V the number of unique terms in the vocabulary, and $\mathbf{N_c}$ the total number of terms in class c. Show all calculus.

Vocabulary:	$\{amazing, run, I, like, it, to, tired, bad\}$	$Total = 8$
Positive class:	$\{amazing, run, I, like, it\}$	$N_P = 5$
Negative class:	$\{to, tired, bad, run\}$	$N_P = 4$

Now we calculate the likelihoods with the given expression:

$$P(\text{I}|P) = \frac{freq(I)+1}{5+8} = \frac{1+1}{13} \approx 0.154 \qquad P(\text{I}|N) = \frac{freq(I)+1}{4+8} = \frac{0+1}{12} \approx 0.083$$

$$P(\text{Like}|P) = \frac{freq(Like)+1}{5+8} = \frac{1+1}{13} \approx 0.154 \quad P(\text{Like}|N) = \frac{freq(Like)+1}{4+8} = \frac{0+1}{12} \approx 0.083$$

$$P(\text{to}|P) = \frac{freq(to)+1}{5+8} = \frac{0+1}{13} \approx 0.077 \qquad P(\text{to}|N) = \frac{freq(to)+1}{4+8} = \frac{0+1}{12} \approx 0.083$$

$$P(\text{run}|P) = \frac{freq(run)+1}{5+8} = \frac{1+1}{13} \approx 0.154 \qquad P(\text{run}|N) = \frac{freq(run)+1}{4+8} = \frac{1+1}{12} \approx 0.167$$

ML assumption can be defined as follows:

$$\begin{aligned} \hat{z} &= \arg\max_{c_i} \{P(c_i|x)\} \\ &= \arg\max_{c_i} \left\{\frac{P(x|c_i) \times P(c_i)}{P(x)}\right\} \\ &= \arg\max_{c_i} \{P(x|c_i)\} \end{aligned} \tag{8}$$

We must caclulate the probabilities for the sentence "I like to run" for both classes.

$$\begin{aligned} P(\text{"I like to run"}|P) &= P(I|P) \cdot P(\text{like}|P) \cdot P(\text{to}|P) \cdot P(\text{run}|P) \\ &= 0.154 \times 0.154 \times 0.077 \times 0.154 \approx 0.00028 \end{aligned}$$

$$\begin{aligned} P(\text{"I like to run"}|N) &= P(I|N) \cdot P(\text{like}|N) \cdot P(\text{to}|N) \cdot P(\text{run}|N) \\ &= 0.083 \times 0.083 \times 0.083 \times 0.167 \approx 0.000095 \end{aligned}$$

Replacing the values in (8), we get:

$$\begin{aligned} \hat{z}_{\text{"I like to run"}} &= \arg\max_{c \in \{P,N\}} \{P(I|c) \cdot P(\text{like}|c) \cdot P(\text{to}|c) \cdot P(\text{run}|c)\} \\ &= \arg\max \{0.00028, 0.000095\} = P \end{aligned}$$

Part II: Programming

Consider the heart-disease.csv dataset available at the course webpage's homework tab. Using sklearn, apply a 5-fold stratified cross-validation with shuffling (random_state = 0) for the assessment of predictive models along this section.

(1) Compare the performance of a kNN with k = 5 and a Naive Bayes with Gaussian assumption (consider all remaining parameters as default):

a. Plot two boxplots with the fold accuracies for each classifier. Is there one more stable than the other regarding performance? Why do you think that is the case? Explain.

```python
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

# Read the dataset
df = pd.read_csv("./heart-disease.csv")
X, y = df.drop("target", axis=1), df["target"]

# Define cross-validation strategy
folds = StratifiedKFold(n_splits=5, 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 = ["#1f77b4", "#E40071"]
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()
```

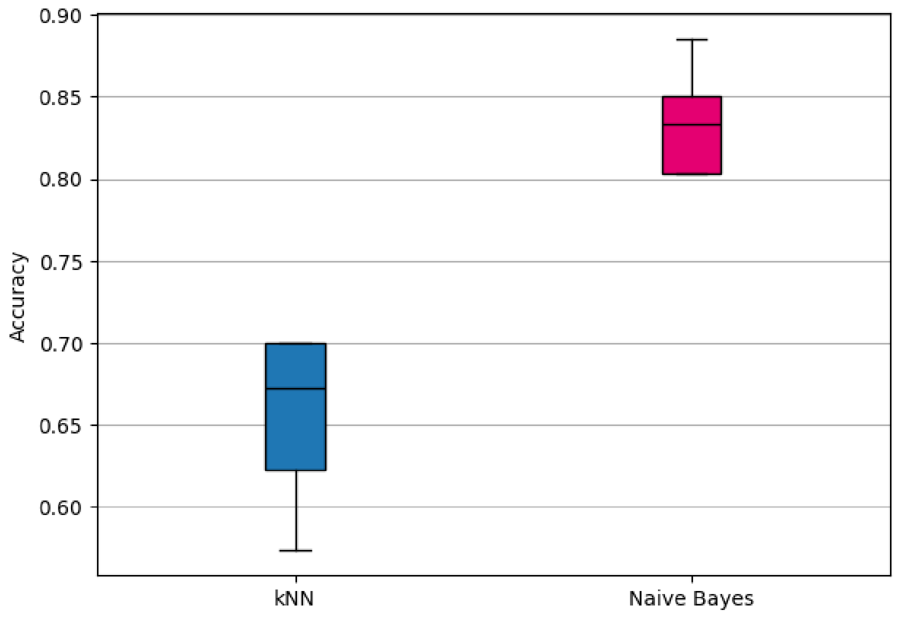

Figure 1: Boxplot of the fold accuracies for each classifier

None have the models contain outliers, but, regarding performance, the naïve Bayes model apears to be more stable. We make this statment becouse, as we can see in the graph, its boxplot is narrower, which means that its accuracy values are clustered closely around the median, and it also has shorter whiskers, indicating more consistence perform and, therefore, better stability. Possible reasons that explain why kNN performs better compared to the naïve Bayes model are:

- kNN is able to handle variable dependencies (naïve Bayes assumes that the features are conditionally independent given the class) and variables that are not normally distributed (naïve Bayes with Gaussian assumption assumes all variables follow this specific distribution).
- Naive Bayes can struggle to estimate probabilities accurately when there are too few data points, which may result in incorrect or even zero probabilities. Additionally, if the class distribution is imbalanced, it can lead to biased predictions because the prior probabilities used in the model's calculations may be skewed.

b. Report the accuracy of both models, this time scaling the data with a Min-Max scaler before training the models. Explain the impact that this preprocessing step has on the performance of each model, providing an explanation for the results.

```
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 sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline

# Read the ARFF file and prepare data
```

```python
df = pd.read_csv("./heart-disease.csv")
X, y = df.drop("target", axis=1), df["target"]

# Define cross-validation strategy
folds = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)

# Create pipelines for scaling
knn_pipeline = Pipeline([("scaler", MinMaxScaler()), ("knn",
    KNeighborsClassifier(n_neighbors=5))])
nb_pipeline = Pipeline([("scaler", MinMaxScaler()), ("nb", GaussianNB())])

# Evaluate classifiers
knn_accs = cross_val_score(knn_pipeline, X, y, cv=folds, scoring="accuracy")
nb_accs = cross_val_score(nb_pipeline, 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 = ["#1f77b4", "#E40071"]
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()
```

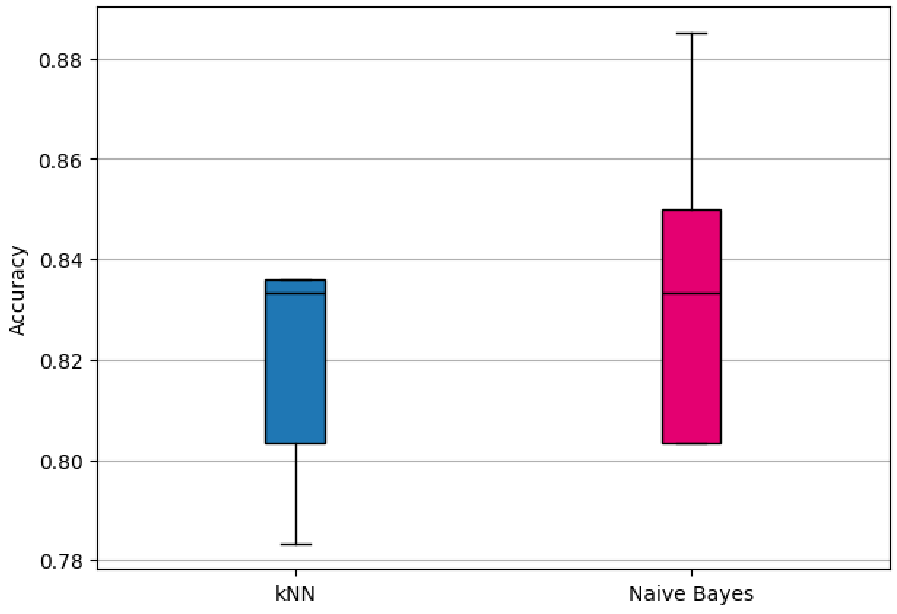

Figure 2: Accuracy of both models with Min-Max scaling

This pre-processing step worsens the performances of both models. The possible reasons for this outcome is that scaling might make irrelevant features more prominent, reducing both models' effectiveness.

- For kNN, the Min-Max scaler alters the distances that the model relies on, which changes the neighbors that are considered "close", leading to incorret classifications. Scaling makes all feature ranges the same, which can decrease the impact of features that were originally more significant, leading to worse performance.
- For naïve Bayes, Min-Max scaling can distort the features, making them not normally distributed, which leads to inaccurate probability estimates.

c. Using scipy, test the hypothesis "the kNN model is statistically superior to naive Bayes regarding accuracy", asserting whether it is true.

```python
from scipy.stats import ttest_rel

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

Since the p-value (p-value = 0.83), is greater than the usual significance levels, (e.g., $\alpha = 0.05$), we cannot reject the null hypothesis, thus we can't afirm that the kNN model is statistically superior to naïve Bayes regarding accuracy. In order to be able to make a correct statement about this hypothesis, it would be necessary to obtain more statistical tests.

(2) a 80-20 train-test split, vary the number of neighbors of a kNN classifier using k = $\{1, 5, 10, 20, 30\}$. Additionally, for each k, train one classifier using uniform weights and distance weights.

a. Plot the train and test accuracy for each model

```python
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Read the dataset
df = pd.read_csv("./heart-disease.csv")
X, y = df.drop("target", axis=1), df["target"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=0)

# Standardize the features
scaler = StandardScaler().fit(X_train)
X_train_scaled, X_test_scaled = scaler.transform(X_train), scaler.transform(
    X_test)
```

```python

# Define different values of k
number_of_neighbors = [1, 5, 10, 20, 30]

# Initialize lists to store accuracy results
train_acc_uniform, test_acc_uniform = [], []
train_acc_distance, test_acc_distance = [], []

# Train and evaluate KNN with 'uniform' weights
for k in number_of_neighbors:
    knn_uniform = KNeighborsClassifier(n_neighbors=k, weights="uniform")
    knn_uniform.fit(X_train_scaled, y_train)
    train_acc_uniform.append(knn_uniform.score(X_train_scaled, y_train))
    test_acc_uniform.append(knn_uniform.score(X_test_scaled, y_test))

# Train and evaluate KNN with 'distance' weights
for k in number_of_neighbors:
    knn_distance = KNeighborsClassifier(n_neighbors=k, weights="distance")
    knn_distance.fit(X_train_scaled, y_train)
    train_acc_distance.append(knn_distance.score(X_train_scaled, y_train))
    test_acc_distance.append(knn_distance.score(X_test_scaled, y_test))

# Plot the results
plt.figure(figsize=(16, 6))

# Plot distance accuracy
plt.subplot(1, 2, 1)
plt.plot(
    number_of_neighbors,
    test_acc_distance,
    label='Test Accuracy',
    marker='D',
    color='#E40071'
)

plt.plot(
    number_of_neighbors,
    train_acc_distance,
    label='Train Accuracy',
    marker='o',
    color='#1f77b4')

plt.xlabel('Number of neighbors')
plt.ylabel('Distance')
plt.title('KNN Accuracy with Distance Weights')
plt.grid(True)
plt.legend()

# Plot uniform accuracy
plt.subplot(1, 2, 2)
plt.plot(
    number_of_neighbors,
    test_acc_uniform,
    label='Test Accuracy',
    marker='D',
    color='#E40071'
)

plt.plot(
    number_of_neighbors,
    train_acc_uniform,
    label='Train Accuracy',
    marker='o',
    color='#1f77b4'
)

# Add labels and title
plt.xlabel('Number of neighbors')
plt.ylabel('Uniform')
plt.title('KNN Accuracy with Uniform Weights')
plt.grid(True)
plt.legend()
plt.show()
```

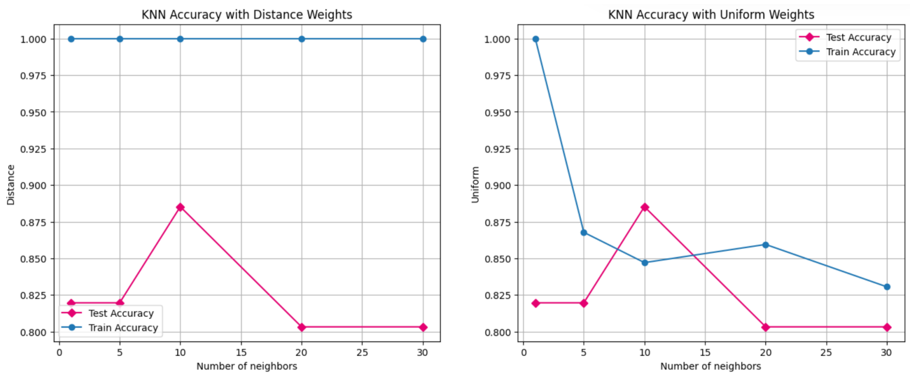

Figure 3: Train and test accuracy for each model

b. Explain the impact of increasing the neighbors on the generalization ability of the models.

As the number of neighbors (k) increases, both models experience a shift in generalization ability. With uniform weights, as k increases, the model smooths its decision boundaries, reducing training accuracy and improving generalization until it peaks at k = 10. Beyond that, both training and test accuracy drop, indicating underfitting. In the case of distance weights, the model maintains perfect training accuracy across all k, suggesting overfitting persists due to the emphasis on closer neighbors. However, the test accuracy also peaks at $k = 10$, similar to uniform weights. After this, increasing k causes underfitting in both models, as they fail to generalize well to the test data. In short, k = 10 is optimal, with larger k values leading to a loss of model generalization in both cases.

(3) Considering the unique properties of the heart-disease.csv dataset, identify two possible difficulties of the naïve Bayes model used in the previous exercises when learning from the given dataset.

Two possible difficulties are:

- Naïve Bayes assumes class conditional independence, which can lead to a loss of accuracy and the unrealistic assumption of independence between features, such as age and cholesterol. The presence of a heart disease may be more strongly correlated with certain features than others, which would not be captured by the model.
- The model assumes that all features follow a specific distribution, which may not accurately reflect the dataset's true distribution, which would lead to inaccurate probability estimates and incorrect classifications.