

Aprendizagem 2022  
Homework II – Group 019  
Diogo Gaspar 99207, Rafael Oliveira 99311

**Part I: Pen and paper**

1. **Compute the recall of a distance-weighted  $k$ NN with  $k = 5$  and distance  $d(x_1, x_2) = \text{Hamming}(x_1, x_2) + \frac{1}{2}$  using leave-one-out evaluation schema (i.e., when classifying one observation, use all remaining ones).**

For starters, it is worth noting that, in this context, the **Hamming distance** between two observations  $x_1$  and  $x_2$  is defined as the number of attributes that differ between them.

Knowing this, we can now create an  $8 \times 8$  matrix (as can be seen below), where each entry represents the Hamming distance ( $+\frac{1}{2}$ ) between two observations. This matrix is symmetric, of course. Each column  $i$ , here, will have  $8 - 1 = 7$  associated entries, each representing the distance  $d$  between the observation  $x_i$  and the remaining 7 observations: we will, then, pick the  $k = 5$  nearest neighbors according to said distance, classifying  $x_i$  in a **distance-weighted** manner.

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$
$x_1$	×	5/2	3/2	1/2	3/2	3/2	3/2	5/2
$x_2$	5/2	×	3/2	5/2	3/2	3/2	3/2	1/2
$x_3$	3/2	3/2	×	3/2	5/2	5/2	1/2	3/2
$x_4$	1/2	5/2	3/2	×	3/2	3/2	3/2	5/2
$x_5$	3/2	3/2	5/2	3/2	×	1/2	5/2	3/2
$x_6$	3/2	3/2	5/2	3/2	1/2	×	5/2	3/2
$x_7$	3/2	3/2	1/2	3/2	5/2	5/2	×	3/2
$x_8$	5/2	1/2	3/2	5/2	3/2	3/2	3/2	×

Table 1: Distance  $d$  between observations - in teal, a given observation's  $k$  nearest neighbors

For each observation  $x_i$ , the  $k$  nearest neighbors are, therefore, the ones represented in teal in the table above. Instead of predicting a class given the neighbors' mode, we'll want to choose it in a distance-weighted manner here: this means that, for each observation  $x_i$ , we'll want to compute the **weighted majority vote** of its  $k$  nearest neighbors, where the weight of each neighbor is given by:

$$w_{i,j} = \frac{1}{d(x_i, x_j)}$$

where  $x_j$  is one of the  $k$  nearest neighbors of  $x_i$ . Considering the data gathered up until now, we'll have the following conclusions regarding the model's classification for each instance:

	Weighted distance to $N$	Weighted distance to $P$	Predicted Class
$x_1$	$3 * 1/(3/2) = 2$	$1/(3/2) + 1/(1/2) = 8/3$	$P$
$x_2$	$3 * 1/(3/2) + 1/(1/2) = 4$	$1/(3/2) = 2/3$	$N$
$x_3$	$1/(3/2) + 1/(1/2) = 8/3$	$3 * 1/(3/2) = 2$	$N$
$x_4$	$3 * 1/(3/2) = 2$	$1/(3/2) + 1/(1/2) = 8/3$	$P$
$x_5$	$1/(3/2) + 1/(1/2) = 8/3$	$3 * 1/(3/2) = 2$	$N$
$x_6$	$1/(3/2) + 1/(1/2) = 8/3$	$3 * 1/(3/2) = 2$	$N$
$x_7$	$1/(3/2) = 2/3$	$3 * 1/(3/2) + 1/(1/2) = 4$	$P$
$x_8$	$3 * 1/(3/2) = 2$	$1/(3/2) + 1/(1/2) = 8/3$	$P$

Table 2: Distance weighting for each observation

We'll have, given the data gathered above, the following confusion matrix:

		Real	
		$P$	$N$
Projected	$P$	2	2
	$N$	2	2

Figure 1: Confusion Matrix

Moreover, the **recall** of a classifier is defined as the ratio between the number of true positives and the number of true positives plus the number of false negatives that the classifier makes. Looking at the confusion matrix above, we can assert that the associated recall will, therefore, be:

$$R = \frac{TP}{TP + FN} = \frac{2}{2 + 2} = \frac{2}{4} = 0.5$$

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

Considering both variable sets,  $\{y_1, y_2\}$  and  $\{y_3\}$ , to be independent and equally important, it'll make sense to train a Bayesian classifier here, such that (and utilizing Bayes' theorem):

$$\forall_{c \in \{N, P\}}, \quad P(C = c | y_1, y_2, y_3) = \frac{P(y_1, y_2, y_3 | C = c)P(C = c)}{P(y_1, y_2, y_3)} \quad (1)$$

More so, since  $\{y_1, y_2\}$  and  $\{y_3\}$  are independent (and  $y_1$  and  $y_2$  are dependent), we can rewrite the above as:

$$\forall_{c \in \{N, P\}}, \quad P(C = c | y_1, y_2, y_3) = \frac{P(y_1, y_2 | C = c)P(y_3 | C = c)P(C = c)}{P(y_1, y_2)P(y_3)} \quad (2)$$

A Bayesian classifier's goal here will be to maximize the numerator of the above equation, since we'll want to find the "maximum posterior hypothesis" (i.e.,  $P(C = c)$ , here), and such a maximization will be independent of  $P(y_1, y_2, y_3)$ . As such, we know that we'll be looking for the following:

$$\operatorname{argmax}_{c \in \{N, P\}} P(y_1, y_2 | C = c)P(y_3 | C = c)P(C = c) \quad (3)$$

For starters, we can note that, from the given training set:

$$P(\textcolor{red}{C} = \textcolor{red}{N}) = \frac{4}{9}, \quad P(\textcolor{teal}{C} = \textcolor{teal}{P}) = \frac{5}{9}$$

We also know that  $y_3$  is normally distributed, meaning we'll have:

$$\forall_{c \in \{N, P\}}, \quad P(y_3 | C = c) \sim \mathcal{N}(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

By also looking at the given training set (which includes the new, ninth sample), we'll be able to extrapolate the following probabilities:

$$C = N : P(C = N) = \frac{4}{9}$$

- $P(y_1 = A, y_2 = 0 | C = N) = 0$
- $P(y_1 = A, y_2 = 1 | C = N) = \frac{1}{4}$
- $P(y_1 = B, y_2 = 0 | C = N) = \frac{2}{4} = \frac{1}{2}$
- $P(y_1 = B, y_2 = 1 | C = N) = \frac{1}{4}$

Regarding  $y_3$  and  $N$  labeled observations, we'll have the following parameters:

$$\mu = \frac{1 + 0.9 + 1.2 + 0.8}{4} = 0.975$$

$$\sigma^2 = \frac{1}{4-1} \sum_{i=1}^4 (y_{3,i} - \mu)^2 = 0.029$$

Therefore:

$$P(y_3 | C = N) \sim \mathcal{N}(x | 0.975, 0.029)$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.029} \exp\left(-\frac{(y_3 - 0.975)^2}{2 \cdot 0.029}\right)$$

$$C = P : P(C = P) = \frac{5}{9}$$

- $P(y_1 = A, y_2 = 0 | C = P) = \frac{2}{5}$
- $P(y_1 = A, y_2 = 1 | C = P) = \frac{1}{5}$
- $P(y_1 = B, y_2 = 0 | C = P) = \frac{1}{5}$
- $P(y_1 = B, y_2 = 1 | C = P) = \frac{1}{5}$

Regarding  $y_3$  and  $P$  labeled observations, we'll have the following parameters:

$$\mu = \frac{1.2 + 0.8 + 0.5 + 0.9 + 0.8}{5} = 0.840$$

$$\sigma^2 = \frac{1}{5-1} \sum_{i=1}^5 (y_{3,i} - \mu)^2 = 0.063$$

Therefore:

$$P(y_3 | C = P) \sim \mathcal{N}(x | 0.840, 0.063)$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.063} \exp\left(-\frac{(y_3 - 0.840)^2}{2 \cdot 0.063}\right)$$

Although not necessary for classifying samples (since Expression 3 does not depend on Expression 2's denominator), we include the following below for completeness's sake:

- $P(y_1 = A, y_2 = 0) = 2/9$
- $P(y_1 = A, y_2 = 1) = 2/9$
- $P(y_1 = B, y_2 = 0) = 3/9$
- $P(y_1 = B, y_2 = 1) = 2/9$

$$\mu_{y_3} = \frac{1.2 + 0.8 + \dots + 1.2 + 0.8}{9} = 0.9000$$

$$\sigma_{y_3}^2 = \frac{1}{9-1} \sum_{i=1}^9 (y_{3,i} - \mu)^2 = 0.0475$$

Therefore:

$$P(y_3) \sim \mathcal{N}(x | 0.9000, 0.0475)$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.0475} \exp\left(-\frac{(y_3 - 0.9000)^2}{2 \cdot 0.0475}\right)$$

The model is now ready to be used to classify new observations.

3. Under a **MAP assumption**, compute  $P(Positive|x)$  of each testing observation.

Considering Bayes' theorem, we know  $P(Positive|x)$  can also be written as follows:

$$P(Positive|x) = \frac{P(x|Positive)P(Positive)}{P(x)}$$

Since  $x$  has two independent variable sets,  $\{y_1, y_2\}$  and  $\{y_3\}$ , we are able to write the above expression as follows (note that  $x$ 's  $y_1$  value will be written as  $y_1$ , and so on, for simplicity's sake):

$$P(Positive|x) = \frac{P(y_1, y_2|Positive)P(y_3|Positive)P(Positive)}{P(y_1, y_2)P(y_3)}$$

The **prior** ( $P(Positive)$ ) shows up in the equation, of course, since we are working under a **MAP assumption**: if we were working under a **MLE assumption**, we would be dropping the prior from the equation.

Let us consider the three samples given in the question's statement:

$$x'_1 = \begin{pmatrix} A \\ 1 \\ 0.8 \end{pmatrix}, \quad x'_2 = \begin{pmatrix} B \\ 1 \\ 1 \end{pmatrix}, \quad x'_3 = \begin{pmatrix} B \\ 0 \\ 0.9 \end{pmatrix}$$

Considering the training observations, we can gather that:

$x'_1$	$x'_2$	$x'_3$
Among all (four) <b>negative</b> training samples, there is <b>1</b> with features $y_1 = A \wedge y_2 = 1$ and <b>1</b> with feature $y_3 = 0.8$ .	Among all (four) <b>negative</b> training samples, there is <b>1</b> with features $y_1 = B \wedge y_2 = 1$ and <b>1</b> with feature $y_3 = 1$ .	Among all (four) <b>negative</b> training samples, there is <b>2</b> with features $y_1 = B \wedge y_2 = 0$ and <b>1</b> with feature $y_3 = 0.9$ .
Among all (five) <b>positive</b> training samples, there is <b>1</b> with features $y_1 = A \wedge y_2 = 1$ and <b>2</b> with feature $y_3 = 0.8$ .	Among all (five) <b>positive</b> training samples, there is <b>1</b> with features $y_1 = B \wedge y_2 = 1$ and <b>0</b> with feature $y_3 = 1$ .	Among all (five) <b>positive</b> training samples there is <b>1</b> with features $y_1 = B \wedge y_2 = 0$ and <b>1</b> with feature $y_3 = 0.9$ .
Among <u>all</u> (nine) training samples, there are <b>2</b> with features $y_1 = A \wedge y_2 = 1$ , and <b>3</b> with feature $y_3 = 0.8$ .	Among <u>all</u> (nine) training samples, there are <b>2</b> with features $y_1 = B \wedge y_2 = 1$ and <b>1</b> with feature $y_3 = 1$ .	Among <u>all</u> (nine) training samples, there are <b>3</b> with features $y_1 = B \wedge y_2 = 0$ and <b>2</b> with feature $y_3 = 0.9$ .

Note, of course, that the probabilities regarding  $y_1$  and  $y_2$  features may be calculated in a discrete manner, directly utilizing the values gathered above:

$\mathbf{x}'_1$	$\mathbf{x}'_2$	$\mathbf{x}'_3$
$P(y_1 = A, y_2 = 1 \textcolor{red}{N}) = 1/4$	$P(y_1 = B, y_2 = 1 \textcolor{red}{N}) = 1/4$	$P(y_1 = B, y_2 = 0 \textcolor{red}{N}) = 2/4$
$P(y_1 = A, y_2 = 1 \textcolor{teal}{P}) = 1/5$	$P(y_1 = B, y_2 = 1 \textcolor{teal}{P}) = 1/5$	$P(y_1 = B, y_2 = 0 \textcolor{teal}{P}) = 1/5$
$P(y_1 = A, y_2 = 1) = 2/9$	$P(y_1 = B, y_2 = 1) = 2/9$	$P(y_1 = B, y_2 = 0) = 3/9$

The same cannot be said, for  $y_3$ , however, with it being normally distributed: we'll need to recall its parameters (which were computed in the previous question), and then will we be able to calculate the needed probabilities regarding these samples'  $y_3$  values:

$$\begin{array}{lll} \mu_{y_3, \text{Negative}} = 0.975 & \mu_{y_3, \text{Positive}} = 0.840 & \mu_{y_3} = 0.9000 \\ \sigma_{y_3, \text{Negative}}^2 = 0.029 & \sigma_{y_3, \text{Positive}}^2 = 0.063 & \sigma_{y_3}^2 = 0.0475 \end{array}$$

With  $y_3$  being normally distributed, we can calculate  $P(y_3|\text{Negative})$ ,  $P(y_3|\text{Positive})$  and  $P(y_3)$ :

$$\forall_{c \in \{\text{Negative}, \text{Positive}\}}, \quad P(y_3|c) = \frac{1}{\sqrt{2\pi\sigma_{y_3,c}^2}} \exp\left(-\frac{(y_3 - \mu_{y_3,c})^2}{2\sigma_{y_3,c}^2}\right)$$

$$P(y_3) = \frac{1}{\sqrt{2\pi\sigma_{y_3}^2}} \exp\left(-\frac{(y_3 - \mu_{y_3})^2}{2\sigma_{y_3}^2}\right)$$

$\mathbf{x}'_1$	$\mathbf{x}'_2$	$\mathbf{x}'_3$
$P(y_3 = 0.8 \textcolor{red}{N}) \approx 1.3816,$	$P(y_3 = 1 \textcolor{red}{N}) \approx 2.3176,$	$P(y_3 = 0.9 \textcolor{red}{N}) \approx 2.1261,$
$P(y_3 = 0.8 \textcolor{teal}{P}) \approx 1.5694,$	$P(y_3 = 1 \textcolor{teal}{P}) \approx 1.2972,$	$P(y_3 = 0.9 \textcolor{teal}{P}) \approx 1.5447,$
$P(y_3 = 0.8) \approx 1.6476$	$P(y_3 = 1) \approx 1.6476$	$P(y_3 = 0.9) \approx 1.8305$

We can, therefore, assert that:

$$P(Positive|x'_1) = \frac{1/5 \times 1.5694 \times 5/9}{2/9 \times 1.6476}$$

$$\approx 0.4763$$

$$P(Positive|x'_2) = \frac{1/5 \times 1.2972 \times 5/9}{2/9 \times 1.6476}$$

$$\approx 0.3937$$

$$P(Positive|x'_3) = \frac{1/5 \times 1.5447 \times 5/9}{3/9 \times 1.8305}$$

$$\approx 0.2813$$

$$P(Negative|x'_1) = \frac{1/4 \times 1.3816 \times 4/9}{2/9 \times 1.6476}$$

$$\approx 0.4193$$

$$P(Negative|x'_2) = \frac{1/4 \times 2.3176 \times 4/9}{2/9 \times 1.6476}$$

$$\approx 0.7033$$

$$P(Negative|x'_3) = \frac{2/4 \times 2.1261 \times 4/9}{3/9 \times 1.8305}$$

$$\approx 0.7743$$

As can be easily noted, the probabilities above (for each sample) do not add up to 1, even though intuitively they should: that happens because the values being calculated above are, in fact, **likelihoods**, not probabilities! To calculate the actual probabilities, we need to normalize the likelihoods by dividing them by the sum of all the likelihoods, as will be shown below (note that only  $P(Positive|x)$  will be calculated, for the sake of conciseness):

$$P(Positive|x'_1) = \frac{0.4763}{0.4763 + 0.4193} \approx 0.5318$$

$$P(Positive|x'_2) = \frac{0.3937}{0.3937 + 0.7033} \approx 0.3589$$

$$P(Positive|x'_3) = \frac{0.2813}{0.2813 + 0.7743} \approx 0.2665$$

4. Given a binary class variable, the default decision threshold of  $\theta = 0.5$ ,

$$f(x|\theta) = \begin{cases} \textit{Positive} & \text{if } P(\textit{Positive}|x) > \theta \\ \textit{Negative} & \text{otherwise} \end{cases}$$

can be adjusted. Which decision threshold – 0.3, 0.5 or 0.7 – optimizes testing accuracy?

As given by the question's statement, we know the actual class values of the testing observations. Moreover, we've calculated  $P(\textit{Positive}|x)$  for each of them in the previous question's answer, so we can easily calculate the accuracy of the classifier for each of the three decision thresholds (considering only these three testing samples):

	Class	$P(\textit{Positive} x)$	$\theta = 0.3$	$\theta = 0.5$	$\theta = 0.7$
$x'_1$	<i>Positive</i>	0.5318	<i>Positive</i>	<i>Positive</i>	<i>Negative</i>
$x'_2$	<i>Positive</i>	0.3589	<i>Positive</i>	<i>Negative</i>	<i>Negative</i>
$x'_3$	<i>Negative</i>	0.2665	<i>Negative</i>	<i>Negative</i>	<i>Negative</i>

Table 3:  $f(x|\theta)$  for varying thresholds, given three testing samples

Here, the classifier's accuracy, considering the table above, amounts to:

$$\text{Accuracy}(\theta = 0.3) = 1$$

$$\text{Accuracy}(\theta = 0.5) = 2/3$$

$$\text{Accuracy}(\theta = 0.7) = 1/3$$

As it can be seen above, a threshold of  $\theta = 0.3$  yields a higher accuracy than both 0.5 and 0.7, hence it should be chosen in order to optimize testing accuracy.



## Part II: Programming

The code utilized in order to answer this section's first two questions answers can be found in this report's appendix. There's both a "general" code section, which can be found initially, and two functions, `first` and `second`, utilized to answer the respective questions - the first to plot the confusion matrices, while the second to test the given hypothesis.

5. Using **sklearn**, considering a 10-fold stratified cross validation (**random=0**), plot the cumulative testing confusion matrices of  $k$ NN (uniform weights,  $k = 5$ , Euclidean distance) and Naïve Bayes (Gaussian assumption). Use all remaining classifier parameters as default.

We ended up utilizing `seaborn`'s heatmap function to plot the required confusion matrices side-by-side.

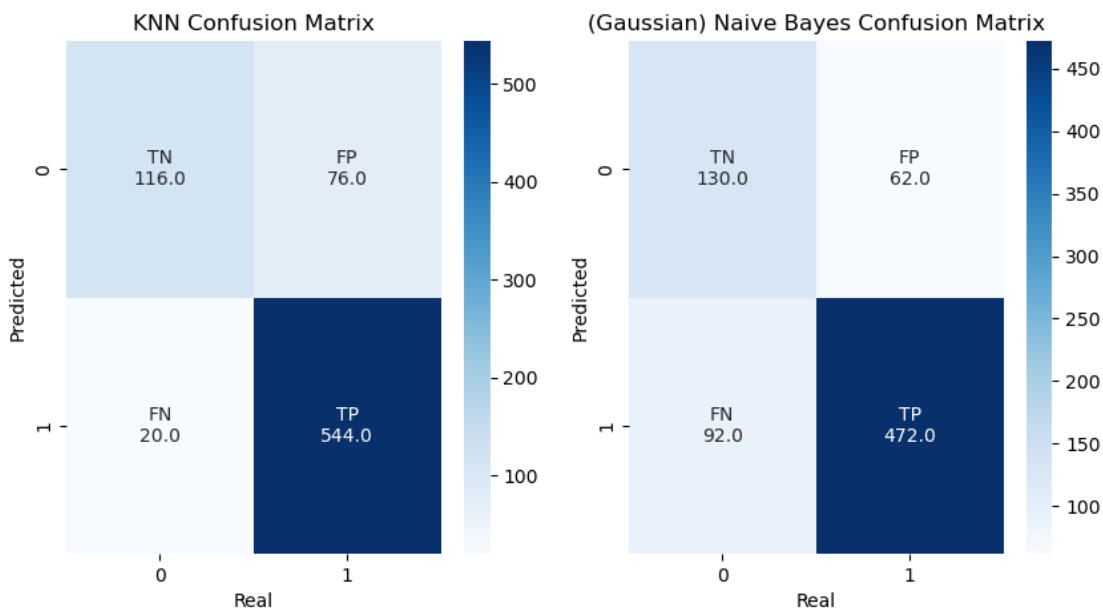


Figure 2: Cumulative testing confusion matrices of  $k$ NN and Naïve Bayes

As a relevant note, we opted to **normalize** our data, utilizing `sklearn`'s `StandardScaler`, in order to avoid overfitting the data (even more so while having relatively small training and testing sample subsets). As such, both this question's answer **and the following ones** will be based on normalized data.

**6. Using `scipy`, test the hypothesis “ $k$ NN is statistically superior to Naïve Bayes regarding accuracy”, asserting whether it is true.**

*All the comments written below are evidently regarding a  $k$ NN implementation such as the one suggested in the previous question’s statement - that is, considering  $k = 5$ .*

`scipy.stats` provides the `ttest_rel` function, which can be utilized to test a hypothesis given two related samples - here, both  $k$ NN’s and Naïve Bayes’ scores are inherently related, since we’re working with the same dataset on both classifiers, hence we can use this function to test the hypothesis.

The considered hypotheses were, considering the problem in hands:

- $H_0$ :  $k$ NN’s accuracy is not statistically superior to Naïve Bayes’
- $H_1$ :  $k$ NN’s accuracy is statistically superior to Naïve Bayes’

As a side-note, we’ve considered, in absence of a given confidence level in the question’s statement, a confidence level of  $1 - \alpha = 0.95$ .

After performing the test (utilizing the `alternative = 'greater'` parameter, in order to test the afore-mentioned hypotheses), we obtained a  $p$ -value of  $\approx 0.0013$  and a  $t$ -statistic of  $\approx 4.1109$ , which leads us to assert that, given  $\alpha = 0.05$ , we must reject the null hypothesis. As such, considering  $\alpha = 0.05$  and normalized data, we can affirm that  $k$ NN’s accuracy is statistically superior to Naïve Bayes’ accuracy.

*Note: Although we used  $\alpha = 0.05$ , these results still hold for all other usual significance levels, i.e.,  $\alpha = 0.01$  and  $\alpha = 0.10$ .*

**7. Enumerate three possible reasons that could underlie the observed differences in predictive accuracy between  $k$ NN and Naïve Bayes.**

*Note: the **homework's FAQ** states that the answer could mention as low as two reasons, even though the original question's statement asks for three. We opted to answer the question as stated in the FAQ.*

In the previous question, we were able to assert that, regarding accuracy,  $k$ NN is indeed statistically superior to Naïve Bayes (considering a significance level of 5%, of course, and for  $k = 5$ ). There are a couple of reasons that could underlie this observation:

- **Naïve Bayes' assumption of independence between features.**

Naïve Bayes' assumption of independence between features is, in fact, a very strong assumption, which is obviously not always true. As such, Naïve Bayes' assumption of independence between features could be detrimental to its accuracy, since it could lead to a loss of information regarding the dataset's features.

- **Unknown distribution of the dataset's features.**

Naïve Bayes' assumption of Gaussian distribution of the dataset's features is, again, a very strong assumption, which is not always true, even more prevalent in a relatively small dataset with a relatively large amount of features, such as this one. As such, by assuming a Gaussian distribution for **all** of the dataset's features, Naïve Bayes is very likely to underfit the dataset (that is, “not only won't it perform well on the training data, nor will it be able to generalize to new data”), since it is not taking into account the dataset's features' distributions, which could be very different from a Gaussian one.

## Appendix

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.model_selection import StratifiedKFold
6 from sklearn.neighbors import KNeighborsClassifier
7 from sklearn.naive_bayes import GaussianNB
8 from sklearn.metrics import accuracy_score
9 from sklearn.metrics import confusion_matrix
10 from scipy.io.arff import loadarff
11 from scipy.stats import ttest_rel
12 from sklearn.preprocessing import StandardScaler
13
14 # Load data
15 data = loadarff('data/pd_speech.arff')
16 df = pd.DataFrame(data[0])
17 df['class'] = df['class'].str.decode('utf-8')
18
19 X = df.drop('class', axis=1)
20 y = df['class']
21
22 # Create 10-fold stratified cross validation
23 skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)
24
25 # Create classifiers
26 knn = KNeighborsClassifier(n_neighbors=5, weights='uniform', metric='euclidean')
27 nb = GaussianNB()
28
29 knn_cm = np.zeros((2, 2))
30 nb_cm = np.zeros((2, 2))
31
32 knn_scores = np.array([])
33 nb_scores = np.array([])
34
35 for train_index, test_index in skf.split(X, y):
36     X_train, X_test = X.iloc[train_index], X.iloc[test_index]
37     y_train, y_test = y.iloc[train_index], y.iloc[test_index]
38
39     # Normalize data
40     scaler = StandardScaler().fit(X_train)
41     X_train = scaler.transform(X_train)
42     X_test = scaler.transform(X_test)
43
44     knn.fit(X_train, y_train)
45     nb.fit(X_train, y_train)
46
47     knn_pred = knn.predict(X_test)
48     nb_pred = nb.predict(X_test)
49
50     # used in the first question
51     knn_cm += confusion_matrix(y_test, knn_pred)
52     nb_cm += confusion_matrix(y_test, nb_pred)
53
54     # used in the second question
```

```

55 knn_scores = np.append(knn_scores, accuracy_score(y_test, knn_pred))
56 nb_scores = np.append(nb_scores, accuracy_score(y_test, nb_pred))
57
58 def first():
59     """Used to answer the first question"""
60     def create_plot(axis, labels, title, cm):
61         plot = sns.heatmap(cm, annot=labels, cmap='Blues', ax=axis, fmt='s')
62         plot.set_title(title)
63         plot.set_ylabel('Predicted')
64         plot.set_xlabel('Real')
65         return plot
66
67     def label_heatmap(cm):
68         return np.asarray(
69             [f"{name}\n{count}" for name, count in zip(GROUP_NAMES, cm.flatten())]
70             ).reshape(2, 2)
71
72     GROUP_NAMES = ['TN', 'FP', 'FN', 'TP']
73     KNN_LABELS = label_heatmap(knn_cm)
74     NB_LABELS = label_heatmap(nb_cm)
75     fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
76     p1 = create_plot(ax1, KNN_LABELS, 'KNN Confusion Matrix', knn_cm)
77     p2 = create_plot(ax2, NB_LABELS, '(Gaussian) Naive Bayes Confusion Matrix', nb_cm)
78     plt.show()
79     # save the plot to a file
80     fig.savefig('assets/hw2-2.1.png')
81
82 def second():
83     """Used to answer the second question"""
84     return ttest_rel(knn_scores, nb_scores, alternative='greater')
85
86 # First programming question
87 first()
88
89 # Second programming question
90 print(second())

```