Aprendizagem 2023
Homework II – Group 090
(ist1103335, ist1103573)

Part I: Pen and paper

Consider the following dataset:

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 $\mathbf{x}_1$–$\mathbf{x}_7$ to be training observations, $\mathbf{x}_8$–$\mathbf{x}_9$ to be testing observations, $\mathbf{y}_1$–$\mathbf{y}_5$ to be input variables and $\mathbf{y}_6$ to be the target variable.
   Hint: you can use scipy.stats.multivariate_normal for multivariate distribution calculus

   (a) [3.5v] Learn a Bayesian classifier assuming: i) $\{\mathbf{y}_1, \mathbf{y}_2\}, \{\mathbf{y}_3, \mathbf{y}_4\}$ and $\{\mathbf{y}_5\}$ sets of independent variables (e.g., $\mathbf{y}_1 \perp \mathbf{y}_3$ yet $\mathbf{y}_1 \not\perp \mathbf{y}_2$), and ii) $\mathbf{y}_1 \times \mathbf{y}_2 \in \mathbb{R}^2$ is normally distributed. Show all parameters (distributions and priors for subsequent testing).

$$p(z|\mathbf{x}) = \frac{p(\mathbf{x}|z) \times p(z)}{p(\mathbf{x})}$$

$$p(z) : \textbf{priors} : p(y_6 = A) = \frac{3}{7}, p(y_6 = B) = \frac{4}{7}$$

$p(\mathbf{x}|z)$: **PMFs** :

z\x	$y_3 = 0, y_4 = 0$	$y_3 = 1, y_4 = 0$	$y_3 = 0, y_4 = 1$	$y_3 = 1, y_4 = 1$	$y_5 = 0$	$y_5 = 1$	$y_5 = 2$
A	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
B	$\frac{2}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0	$\frac{1}{4}$	$\frac{2}{4}$	$\frac{1}{4}$
Total	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{1}{7}$	$\frac{2}{7}$	$\frac{3}{7}$	$\frac{2}{7}$

$p(\mathbf{x}|z)$: **PDFs** :

$$\mu_{1,2} = \begin{bmatrix} \frac{sum(y_1)}{\#y_1} \\ \frac{sum(y_2)}{\#y_2} \end{bmatrix}, \Sigma_{1,2} = \begin{bmatrix} Var(y_1) & Cov(y_1, y_2) \\ Cov(y_2, y_1) & Var(y_2) \end{bmatrix}$$

$$p(y1, y2|A) : \mathcal{N}\left(\mu_{1,2|A}, \Sigma_{1,2|A}\right), \mu_{1,2|A} = \begin{bmatrix} 0.24 \\ 0.52 \end{bmatrix}, \Sigma_{1,2|A} = \begin{bmatrix} 0.0064 & 0.0096 \\ 0.0096 & 0.0336 \end{bmatrix}$$

$$p(y1, y2|B) : \mathcal{N}\left(\mu_{1,2|B}, \Sigma_{1,2|B}\right), \mu_{1,2|B} = \begin{bmatrix} 0.5925 \\ 0.3275 \end{bmatrix}, \Sigma_{1,2|B} = \begin{bmatrix} 0.0229 & -0.0098 \\ -0.0098 & 0.0314 \end{bmatrix}$$

(b) [2.5v] Under a MAP assumption, classify each testing observation showing all your calculus.

$$y = argmax\{P\left(A|x\right), P\left(B|x\right)\} \tag{1}$$

We just need to calculate the **likelihood** of the input leading to A and to B and multiply that by the **priors**. The we just pick the class with the biggest **posterior**:

$$posterior = likelihood \times prior,\ likelihood = p\left(x|z\right),\ prior = p\left(z\right) \tag{2}$$

$$p\left(y_1 = a, y_2 = b|A\right) = \mathcal{N}\left(\left(a,b\right), \mu_{1,2|A}, \Sigma_{1,2|A}\right)$$
$$p\left(y_1 = a, y_2 = b|B\right) = \mathcal{N}\left(\left(a,b\right), \mu_{1,2|B}, \Sigma_{1,2|B}\right)$$

$$\begin{aligned}\mathbf{x}_8 : p\left(0.38, 0.52, 0, 1, 0|A\right) &= p\left(0.38, 0.52|A\right) \times p\left(0, 1|A\right) \times p\left(0|A\right) \\ &= \mathcal{N}\left(\left(0.38, 0.52\right), \mu_{1,2|A}, \Sigma_{1,2|A}\right) \times \frac{1}{3} \times \frac{1}{3} = 0.9847 \times \frac{1}{9} = 0.1094 \\ p\left(A|x8\right) &= 0.1094 \times \frac{3}{7} \times k = \mathbf{0.0469} \times k\end{aligned}$$

$$\begin{aligned}\mathbf{x}_8 : p\left(0.38, 0.52, 0, 1, 0|B\right) &= p\left(0.38, 0.52|B\right) \times p\left(0, 1|B\right) \times p\left(0|B\right) \\ &= 1.9624 \times \frac{1}{4} \times \frac{1}{4} = 0.1226 \\ p\left(B|x8\right) &= 0.1226 \times \frac{4}{7} \times k = \mathbf{0.0701} \times k\end{aligned}$$

Therefore, we conclude that x_8 should be classified as **B**.

$$\begin{aligned}\mathbf{x}_9 : p\left(0.42, 0.59, 0, 1, 1|A\right) &= p\left(0.42, 0.59|A\right) \times p\left(0, 1|A\right) \times p\left(1|A\right) \\ &= 0.4031 \times \frac{1}{3} \times \frac{1}{3} = 0.0448 \\ p\left(A|x9\right) &= 0.0448 \times \frac{3}{7} \times k = \mathbf{0.0191} \times k\end{aligned}$$

$$\begin{aligned}\mathbf{x}_9 : p\left(0.42, 0.59, 0, 1, 1|B\right) &= p\left(0.42, 0.59|B\right) \times p\left(0, 1|B\right) \times p\left(1|B\right) \\ &= 1.7286 \times \frac{1}{4} \times \frac{2}{4} = 0.2161 \\ p\left(B|x9\right) &= 0.2161 \times \frac{4}{7} \times k = \mathbf{0.1235} \times k\end{aligned}$$

Therefore, we conclude that x_9 should be classified as **B**.

(c) [2v] Consider that the default decision threshold of $\theta = 0.5$ can be adjusted according to

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

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

Under a maximum likelihood assumption, we consider p(Class|x) $\simeq$ p(x|Class), because we assume the priors are uniformly distributed.

$$p\left(A|x8\right) \simeq p\left(x8|A\right) = 0.1094$$
$$normalization: p\left(x8|A\right) = \frac{p\left(x8|A\right)}{p\left(x8|A\right) + p\left(x8|B\right)} = \textbf{0.4716}$$

$$\left(A|x9\right) \simeq p\left(x9|A\right) = 0.0448$$
$$normalization: p\left(x9|A\right) = \frac{p\left(x9|A\right)}{p\left(x9|A\right) + p\left(x9|B\right)} = \textbf{0.1717}$$

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

- At $\theta < 0.1717$: Accuracy = $\frac{1}{2}$
- At $0.1717 \leq \theta < 0.4716$: Accuracy = **1**
- At $\theta \geq 0.4716$: Accuracy = $\frac{1}{2}$

Therefore, thresholds in [0.1717, 0.47164[optimize testing accuracy.

2. Let $\mathbf{y}_1$ be the target numeric variable, $\mathbf{y}_2$-$\mathbf{y}_6$ be the input variables where $\mathbf{y}_2$ is binarized under an equal-width (equal-range) discretization. For the evaluation of regressors, consider a 3-fold cross-validation over the full dataset ($\mathbf{x}_1$-$\mathbf{x}_9$) without shuffling the observations.

   (a) [1v] Identify the observations and features per data fold after the binarization procedure.

   Given that y_2 is binarized under equal-width discretization, we need to determine two equal-width bins for y_2. We'll call them 1 and 2.

   Bin 1: 0 <= y_2 < 0.5

   Bin 2: 0.5 <= y_2 <= 1

   Now we'll assign observations to folds based on the bin to which their y_2 values belong. Considering the **3-cross validation**:

   Iteration 1

   Testing: x_7,x_8,x_9 (Bin 2)

   Training: x_1,x_2,x_4,x_5,x_6 (Bin 1) and x_3 (Bin 2)

   Iteration 2

   Testing: x_4,x_6,x_5 (Bin 1)

   Training: x_1,x_2 (Bin 1), x_3,x_7,x_8,x_9 (Bin 2)

   Iteration 3

   Testing: x_1,x_2,(Bin 1) and x_3 (Bin 2)

   Training: x_4,x_5,x_6 (Bin 1) and x_7,x_8,x_9 (Bin 2)

   (b) [4v] Consider a distance-weighted kNN with k = 3, Hamming distance (d), and 1/d weighting. Compute the MAE of this kNN regressor for the 1st iteration of the cross-validation (i.e. train observations have the lower indices).

   The MAE is calcuted by: MAE = $\frac{1}{n}\Sigma|y_{\text{true}_i} - y_{\text{pred}_i}|$

   So we start by calculating the Hamming distances.

For x_7:

$d_{x_1} = 4$, $d_{x_2} = 4$, $d_{x_3} = 2$, $d_{x_4} = 2$, $d_{x_5} = 3$, $d_{x_6} = 4$

For x_8:

$d_{x_1} = 2$, $d_{x_2} = 4$, $d_{x_3} = 1$, $d_{x_4} = 4$, $d_{x_5} = 3$, $d_{x_6} = 5$

For x_9:

$d_{x_1} = 4$, $d_{x_2} = 4$, $d_{x_3} = 2$, $d_{x_4} = 2$, $d_{x_5} = 3$, $d_{x_6} = 4$

Therefore we have the 3 nearest neighbours and respective weights as such. This way we can predict the value of y_1:

For x7:

Weight for x_3: $\frac{1}{d_{x_3}} = 0.5$

Weight for x_4: $\frac{1}{d_{x_4}} = 0.5$

Weight for x_5: $\frac{1}{d_{x_5}} \approx 0.3333$

$$y_{\text{pred}_7} = \frac{0.32 \cdot 0.5 \;\; 0.54 \cdot 0.5 \;\; 0.66 \cdot 0.3333}{0.5 + 0.5 + 0.3333} \approx 0.487$$

For x_8:

Weight for x_1: $\frac{1}{d_{x_1}} = 0.5$

Weight for x_3: $\frac{1}{d_{x_3}} = 1$

Weight for x_5: $\frac{1}{d_{x_5}} \approx 0.3333$

$$y_{\text{pred}_8} = \frac{0.24 \cdot 0.5 \;\; 0.32 \cdot 1 \;\; 0.66 \cdot 0.3333}{0.5 + 1 + 0.3333} \approx 0.3602$$

For x_9:

Weight for x_3: $\frac{1}{d_{x_3}} = 0.5$

Weight for x_4: $\frac{1}{d_{x_4}} = 0.5$

Weight for x_5: $\frac{1}{d_{x_5}} \approx 0.3333$

$$y_{\text{pred}_9} = \frac{0.32 \cdot 0.5 \;\; 0.54 \cdot 0.5 \;\; 0.66 \cdot 0.3333}{0.5 + 0.5 + 0.3333} \approx 0.487$$

$$\begin{aligned} MAE &= \frac{1}{3}\left(|y_{\text{true}_7} - y_{\text{pred}_7}| \;\; |y_{\text{true}_8} - y_{\text{pred}_8}| \;\; |y_{\text{true}_9} - y_{\text{pred}_9}|\right) \\ &= \frac{1}{3}\left(|0.41 - 0.487| \;\; |0.38 - 0.3602| \;\; |0.42 - 0.487|\right) \\ &\approx 0.0550 \end{aligned}$$

Part II: Programming

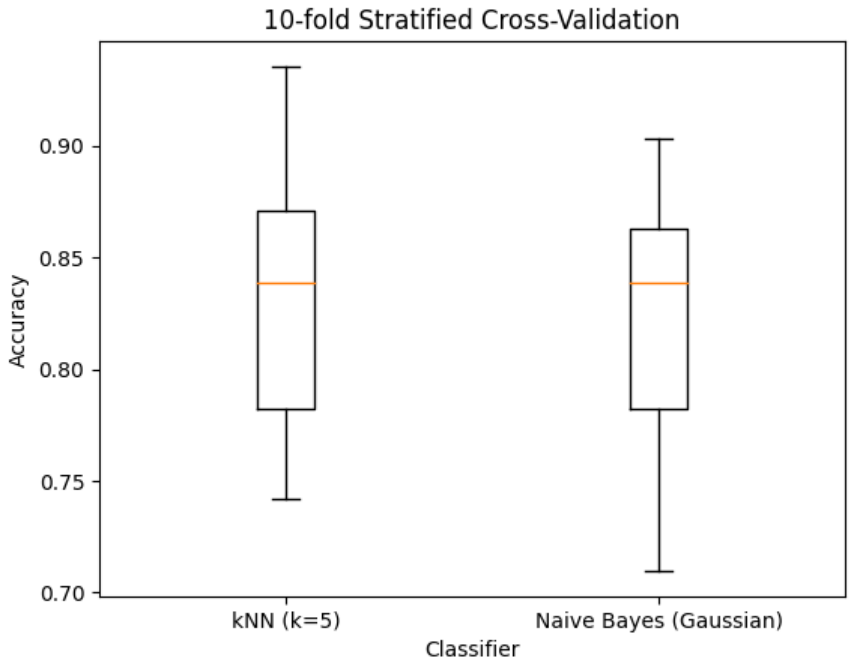
1. Compare the performance of kNN with $k = 5$ and naïve Bayes with Gaussian assumption (consider all remaining parameters for each classifier as sklearn's default):

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

   We used the following program:

```python
#a) plot two boxplost with the fold accuracies for kNN with k=5 and naive Bayes with Gaussian assumptions
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier

classifiers = [
    KNeighborsClassifier(n_neighbors=5),
    GaussianNB()
]

classifier_names = [
    'kNN (k=5)',
    'Naive Bayes (Gaussian)'
]

classifier_accuracies = []

for classifier in classifiers:
    accuracies = cross_val_score(classifier, X, y, cv=stratified_kfold)
    classifier_accuracies.append(accuracies)

plt.boxplot(classifier_accuracies)
plt.xticks(np.arange(1, len(classifier_names) + 1), classifier_names)
plt.xlabel('Classifier')
plt.ylabel('Accuracy')
plt.title('10-fold Stratified Cross-Validation')
plt.show()
```

   And we obtained the boxplots:



b. Using scipy, test the hypothesis "kNN is statistically superior to naïve Bayes regarding accuracy", asserting whether is true.

To do this we did:

```python
#b) Using scipy, test the hypothesis "kNN is statistically superior to naïve Bayes regarding accuracy", assert

from scipy import stats

p_value = stats.ttest_rel(classifier_accuracies[0], classifier_accuracies[1], alternative='greater').pvalue

# Define the significance level
alpha = 0.05

if p_value < alpha:
    result = "k-NN is statistically superior to Naive Bayes"
else:
    result = "k-NN is not statistically superior to Naive Bayes"

print("P-value:", p_value)
print("Hypothesis Test Result:", result)
```

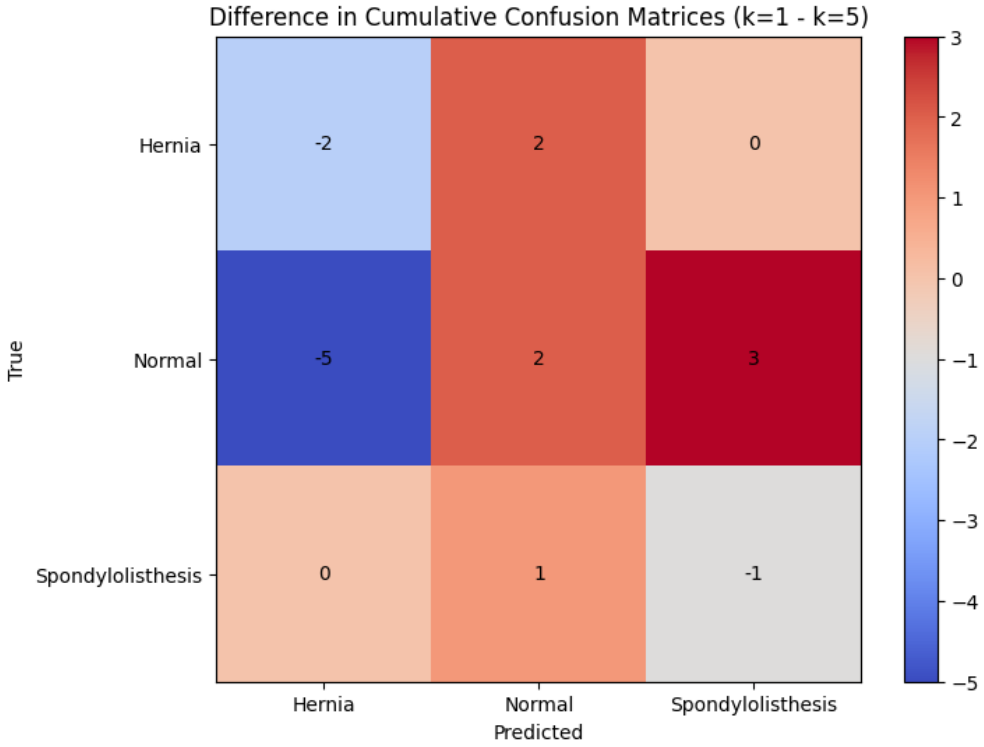
With this we determined that the p_value is approximately 0.19 and therefore we conclude that k-NN **is not statistically superior** to Naive Bayes regarding accuracy.

2. Consider two kNN 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.

   We used this programm:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder

knn_k1 = KNeighborsClassifier(n_neighbors=1, weights='uniform', metric='euclidean')
knn_k5 = KNeighborsClassifier(n_neighbors=5, weights='uniform', metric='euclidean')

cm_k1, cm_k5= np.array([[0,0,0],[0,0,0],[0,0,0]]), np.array([[0,0,0],[0,0,0],[0,0,0]])
X = df.iloc[:, :-1].values

label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df.iloc[:, -1])

for train_index, test_index in stratified_kfold .split(X, y):

    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    knn_k1.fit(X_train, y_train)
    knn_k5.fit(X_train, y_train)
    y_pred_1 = knn_k1.predict(X_test)
    y_pred_5 = knn_k5.predict(X_test)
    cm_k1+= confusion_matrix(y_test, y_pred_1)
    cm_k5+= confusion_matrix(y_test, y_pred_5)

cm_diff = cm_k1 - cm_k5

plt.figure(figsize=(8, 6))
plt.imshow(cm_diff, cmap='coolwarm', interpolation='nearest')
plt.colorbar()
plt.title("Difference in Cumulative Confusion Matrices (k=1 - k=5)")
plt.xticks(np.arange(3), ['Hernia', 'Normal', 'Spondylolisthesis'])
plt.yticks(np.arange(3), ['Hernia', 'Normal', 'Spondylolisthesis'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

And obtained the matrix:



We can take some conclusions about each class:

Hernia Class: - The k=1 classifier is less accurate in correctly identifying "Hernia" cases, with 2 fewer true negatives compared to k=5. - The k=1 classifier tends to make more false positive predictions for "Normal" cases, with 2 more false positives compared to k=5.

Normal Class: - The k=1 classifier is less accurate in correctly identifying "Normal" cases, with 5 fewer true negatives compared to k=5. - The k=1 classifier performs better in correctly identifying "Normal" cases, with 2 more true positives compared to k=5. - The k=1 classifier is less accurate in identifying "Spondylolisthesis" cases, with 3 more false negatives compared to k=5.

Spondylolisthesis Class: - There is no significant difference in predictions for the "Hernia" class between the two classifiers (indicated by 0 difference). - The k=1 classifier is less accurate in identifying "Normal" cases, with 1 more false negative compared to k=5. - The k=1 classifier performs slightly worse in correctly identifying "Spondylolisthesis" cases, with 1 fewer true positive compared to k=5.

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

   1. **Assumption of Independence**: Naive Bayes assumes that the features used for classification are conditionally independent, given the class label. In this dataset, it's unlikely that all features are truly independent. For example, features like "pelvic_incidence," "pelvic_tilt," "lumbar_lordosis_angle," and "sacral_slope" may have some degree of correlation because they are related to spinal measurements. Violation of the independence assumption can lead to suboptimal classification results.

   2. **Continuous Features**: Naive Bayes typically works well with discrete or categorical data. While it can handle continuous features, it assumes that the continuous data follows a specific

probability distribution (e.g., Gaussian Naive Bayes assumes a Gaussian distribution). If the actual data distribution significantly deviates from these assumptions, the algorithm's performance may be negatively affected.

3. **Class Imbalance**: If there is a significant class imbalance in the dataset, where one class is much more frequent than others, Naive Bayes can be biased towards the majority class. "Hernia" might be less common than "Spondylolisthesis" or "Normal". Naive Bayes may struggle to make accurate predictions for minority classes in such cases.

End note: do not forget to also submit your Jupyter notebook