

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.,  $y_1 \perp y_3$  yet  $y_1 \not\perp y_2$ ), and ii)  $y_1 \times y_2 \in \mathbb{R}^2$  is normally distributed. Show all parameters (distributions and priors for subsequent testing).

Gonçalo

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

Gonçalo

- (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?

Raquel

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.

Raquel

- (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<sup>st</sup> iteration of the cross-validation (i.e. train observations have the lower indices).

Raquel

## 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.

```
1 import matplotlib.pyplot as plt, pandas as pd
2 from sklearn.model_selection import StratifiedKFold, cross_val_score
3 from sklearn.neighbors import KNeighborsClassifier
4 from sklearn.naive_bayes import GaussianNB
5 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 # Define cross-validation strategy
14 folds = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)
15
16 # Initialize classifiers
17 knn_predictor = KNeighborsClassifier(n_neighbors=5)
18 nb_predictor = GaussianNB()
19
20 # Evaluate classifiers
21 knn_accs = cross_val_score(knn_predictor, X, y, cv=folds, scoring="accuracy")
22 nb_accs = cross_val_score(nb_predictor, X, y, cv=folds, scoring="accuracy")
23
24 # Plot boxplots
25 plt.figure(figsize=(7, 5))
26 b_plot = plt.boxplot(
27     [knn_accs, nb_accs], patch_artist=True, labels=["kNN", "Naive Bayes"]
28 )
29
30 colors = ["#f8766d", "#00bfc4"]
31 for patch, color in zip(b_plot["boxes"], colors):
32     patch.set_facecolor(color)
33 for median in b_plot["medians"]:
34     median.set_color("black")
35
36 plt.ylabel("Accuracy")
37 plt.grid(axis="y")
38 plt.show()
```

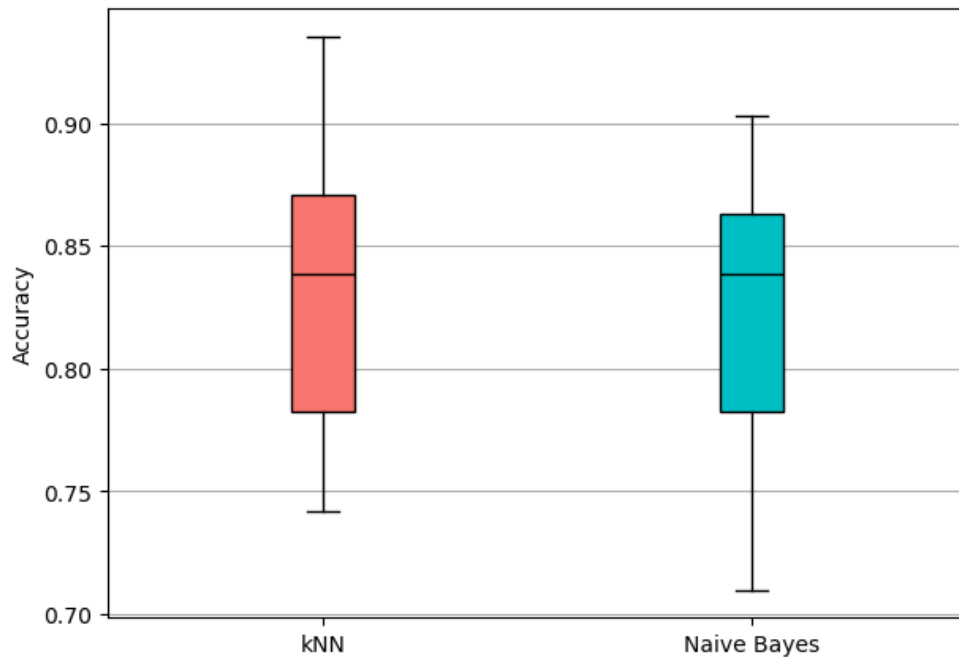


Figure 1: Boxplots with the fold accuracies of  $k$ NN ( $k = 5$ ) and Naïve Bayes

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

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

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

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

```
1 from scipy.stats import ttest_rel
2
3 # Is knn better than naive bayes?
4 res = ttest_rel(knn_accs, nb_accs, alternative="greater")
5 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.

```
1 import numpy as np, matplotlib.pyplot as plt, pandas as pd, seaborn as sns
2 from sklearn.model_selection import StratifiedKFold
3 from sklearn.neighbors import KNeighborsClassifier
4 from sklearn.metrics import confusion_matrix
5 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()

```

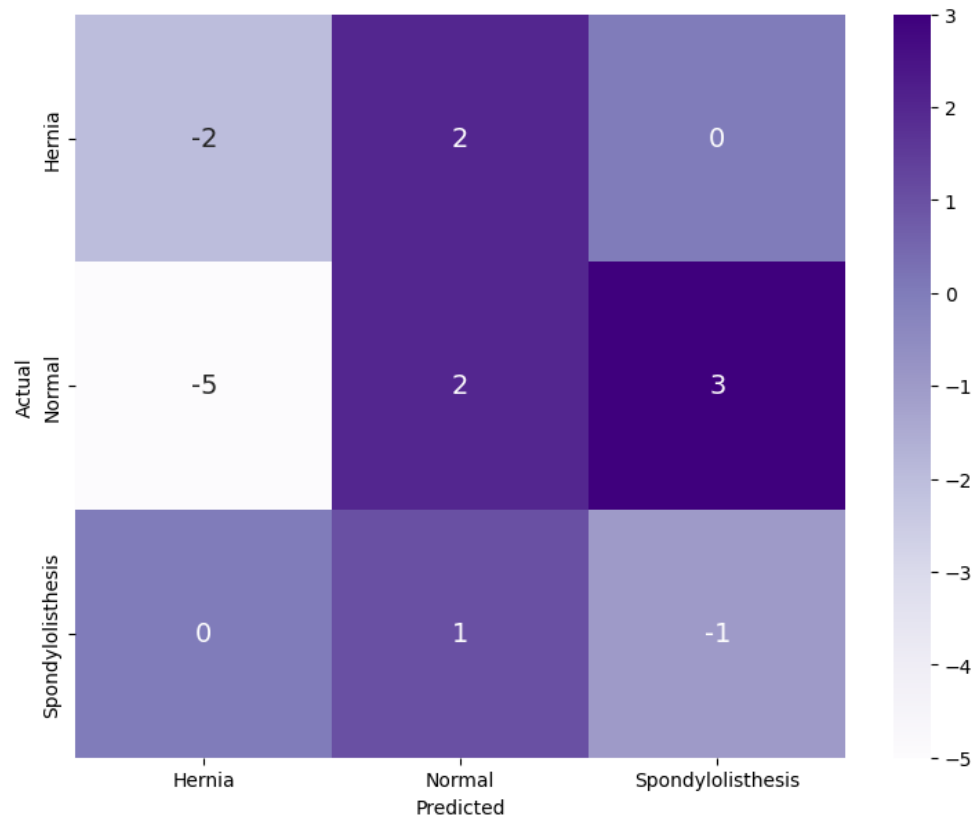


Figure 2: Confusion Matrix Differences Between  $k=1$  and  $k=5$  k-Nearest Neighbors ( $k$ NN) Classifiers

Blah

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:

- Variable dependencies (inadequacy of independence assumption).
- Variables not normally distributed (inadequacy of Gaussian assumption). Probability estimates from a limited number of observations (e.g., inadequate estimates, null probabilities).
- Imbalanced class creating biases in MAP estimates via priors.