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).**

   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 kNN with k = 3, Hamming distance (*d*), and 1 / *d* weighting. Compute the MAE of this kNN regressor for the 1^{st} 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 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.**

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

# 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"]

# Initialize classifiers
knn_classifier = KNeighborsClassifier(n_neighbors=5)
naive_bayes_classifier = GaussianNB()

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

# Evaluate classifiers
knn_accuracies = cross_val_score(knn_classifier, X, y, cv=cv, scoring='accuracy')
naive_bayes_accuracies = cross_val_score(naive_bayes_classifier, X, y, cv=cv, scoring='accuracy')

# Plot boxplots
plt.boxplot([knn_accuracies, naive_bayes_accuracies], labels=['kNN', 'Naive Bayes'])
plt.title('Classifier Comparison')
plt.ylabel('Accuracy')
plt.show()
```

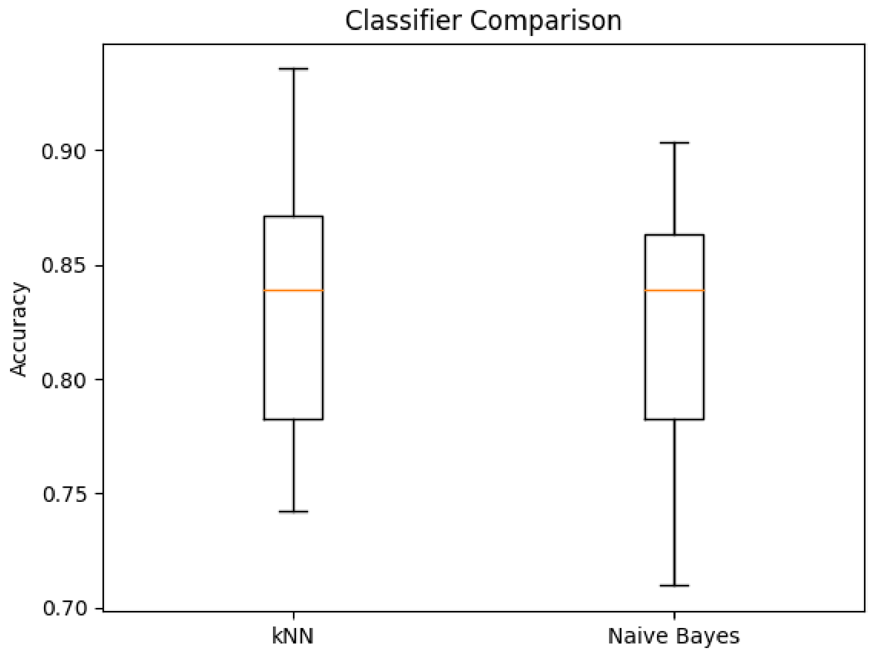

Figure 1: 10-Fold Cross-Validation Results for kNN and Naive Bayes on Column Diagnosis Dataset

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

```
# Perform paired t-test
t_statistic, p_value = ttest_rel(knn_accuracies, naive_bayes_accuracies)

# Check the p-value
if p_value < 0.05:
    print("Reject null hypothesis: kNN is statistically superior to Naive Bayes in terms of accuracy")
else:
    print("Fail to reject null hypothesis: No significant difference in accuracy between kNN and Naive Bayes")
```

Answer: Fail to reject null hypothesis: No significant difference in accuracy between kNN and Naive Bayes

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.**

```
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

# Read the ARFF file and prepare data
data = loadarff("./data/column_diagnosis.arff")
```

```python
df = pd.DataFrame(data[0])
df["class"] = df["class"].str.decode("utf-8")
X, y = df.drop("class", axis=1), df["class"]

# Initialize StratifiedKFold with 10 folds and shuffling
folds = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)

# Create kNN classifiers with k=1 and k=5
knn_1 = KNeighborsClassifier(
    n_neighbors=1, weights="uniform", metric="euclidean"
)
knn_5 = KNeighborsClassifier(
    n_neighbors=5, weights="uniform", metric="euclidean"
)

labels = ["Hernia", "Normal", "Spondylolisthesis"]
cm_1, cm_5 = np.zeros((3, 3)), np.zeros((3, 3))
for train_k, test_k in folds.split(X, y):
    X_train, X_test = X.iloc[train_k], X.iloc[test_k]
    y_train, y_test = y.iloc[train_k], y.iloc[test_k]

    # Fit kNN classifiers and assess
    knn_1.fit(X_train, y_train)
    knn_5.fit(X_train, y_train)
    knn_1_pred, knn_5_pred = knn_1.predict(X_test), knn_5.predict(X_test)
    cm_1 += np.array(confusion_matrix(y_test, knn_1_pred, labels=labels))
    cm_5 += np.array(confusion_matrix(y_test, knn_5_pred, labels=labels))

# Calculate cumulative confusion matrices
cm_diff = cm_1 - cm_5
cm_diff_df = pd.DataFrame(cm_diff, index=labels, columns=labels)

# Plot the differences
plt.figure(figsize=(9, 7))
sns.heatmap(
    cm_diff_df, cmap="Purples", annot=True, annot_kws={"fontsize": 14}, fmt="g"
)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

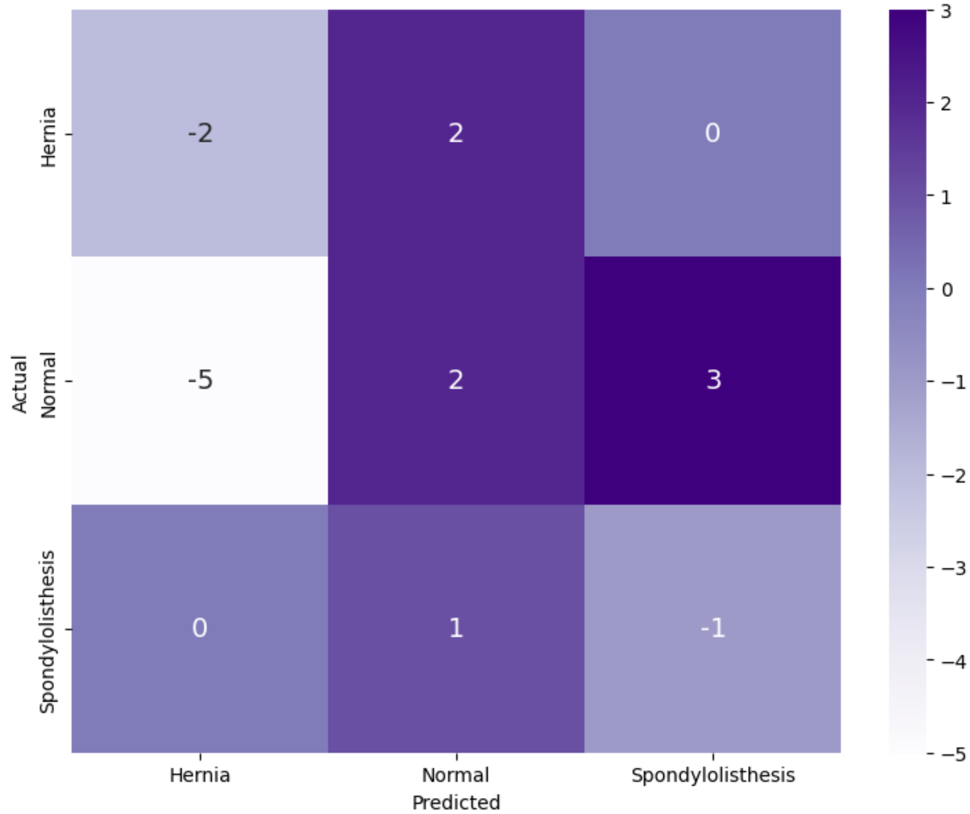

Figure 2: Confusion Matrix Differences Between k=1 and k=5 k-Nearest Neighbors (kNN) Classifiers

Blah

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

   Gonçalo