**I. Pen-and-paper** [13v]

Consider the following dataset:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 𝐷 | 𝑦1 | 𝑦2 | 𝑦3 | 𝑦4 | 𝑦5 | 𝑦6 |
| 𝐱1 | 0.24 | 0.36 | 1 | 1 | 0 | A |
| 𝐱2 | 0.16 | 0.48 | 1 | 0 | 1 | A |
| 𝐱3 | 0.32 | 0.72 | 0 | 1 | 2 | A |
| 𝐱4 | 0.54 | 0.11 | 0 | 0 | 1 | B |
| 𝐱5 | 0.66 | 0.39 | 0 | 0 | 0 | B |
| 𝐱6 | 0.76 | 0.28 | 1 | 0 | 2 | B |
| 𝐱7 | 0.41 | 0.53 | 0 | 1 | 1 | B |
| 𝐱8 | 0.38 | 0.52 | 0 | 1 | 0 | A |
| 𝐱9 | 0.42 | 0.59 | 0 | 1 | 1 | B |

1. Consider 𝐱1–𝐱7 to be training observations, 𝐱8–𝐱9 to be testing observations, 𝑦1– 𝑦5 to be input variables and 𝑦6 to be the target variable.

*Hint*: you can use *scipy.stats.multivariate\_normal* for multivariate distribution calculus

* 1. [3.5v] Learn a Bayesian classifier assuming: i) {𝑦1, 𝑦2}, {𝑦3, 𝑦4} and {𝑦5} sets of independent

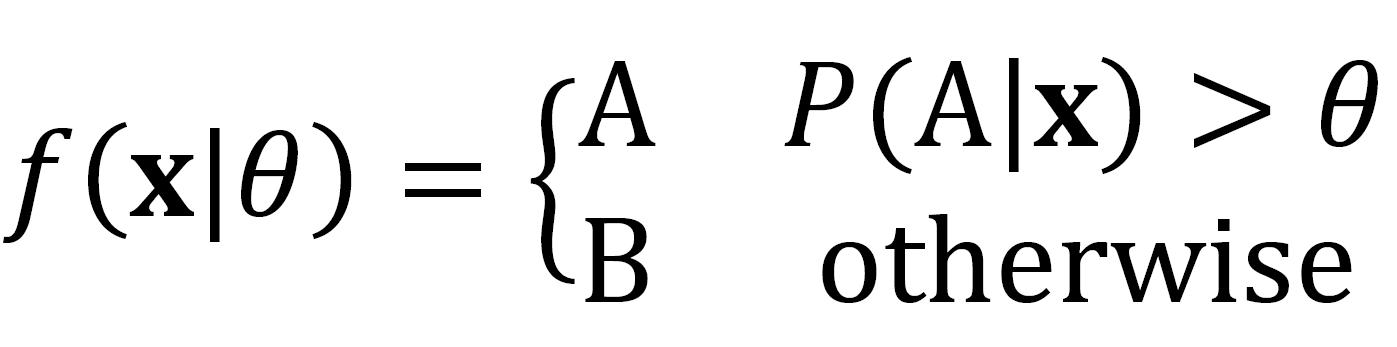
variables (e.g., 𝑦1 ⫫ 𝑦3 yet 𝑦1

⫫

𝑦2), and ii) 𝑦1 × 𝑦2 ∈ ℝ2 is normally distributed. Show all

parameters (distributions and priors for subsequent testing).

* 1. [2.5v] Under a MAP assumption, classify each testing observation showing all your calculus.
  2. [2v] Consider that the default decision threshold of 𝜃 = 0.5 can be adjusted according to



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

1. Let 𝑦1 be the target numeric variable, 𝑦2-𝑦6 be the input variables where 𝑦2 is binarized under an equal-width (equal-range) discretization. For the evaluation of regressors, consider a 3-fold

cross-validation over the full dataset (𝐱1- 𝐱9) without shuffling the observations.

* 1. [1v] Identify the observations and features per data fold after the binarization procedure.
  2. [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).

# Programming and critical analysis [7v]

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. [3v] 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):

import pandas as pd

from scipy.io.arff import loadarff

from sklearn.model\_selection import StratifiedKFold

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

# Read ARFF file

data = loadarff('column\_diagnosis.arff')

df = pd.DataFrame(data[0])

df['class'] = df['class'].str.decode('utf-8')

X = df.drop('class', axis=1)

y = df['class']

stratified\_cv = StratifiedKFold(n\_splits=10, shuffle=True, random\_state=0)

knn\_accuracies, nb\_accuracies = [], []

knn\_classifier = KNeighborsClassifier(n\_neighbors=5)

nb\_classifier = GaussianNB()

# Iterate through each fold of stratified cross-validation

for train\_index, test\_index in stratified\_cv.split(X, y):

    X\_train, X\_test = X.iloc[train\_index], X.iloc[test\_index]

    y\_train, y\_test = y.iloc[train\_index], y.iloc[test\_index]

    # Train both classifiers

    knn\_classifier.fit(X\_train, y\_train)

    nb\_classifier.fit(X\_train, y\_train)

    # Predict and evaluate 𝑘NN

    knn\_pred = knn\_classifier.predict(X\_test)

    knn\_accuracy = accuracy\_score(y\_test, knn\_pred)

    knn\_accuracies.append(knn\_accuracy)

    # Predict and evaluate Gaussian Naïve Bayes

    nb\_pred = nb\_classifier.predict(X\_test)

    nb\_accuracy = accuracy\_score(y\_test, nb\_pred)

    nb\_accuracies.append(nb\_accuracy)

* 1. Plot two boxplots with the fold accuracies for each classifier.

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 4))

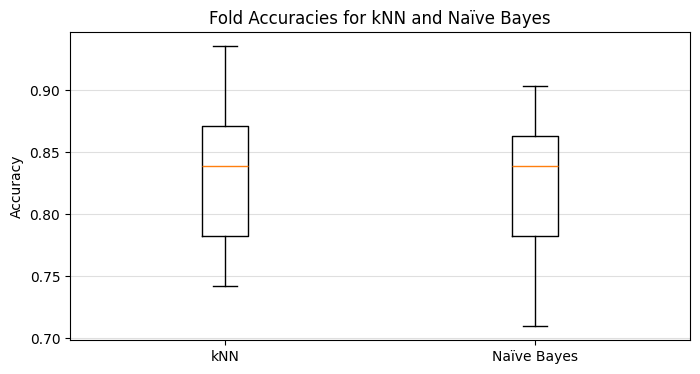
plt.boxplot([knn\_accuracies, nb\_accuracies], labels=['kNN', 'Naïve Bayes'])

plt.title('Fold Accuracies for kNN and Naïve Bayes')

plt.ylabel('Accuracy')

plt.grid(axis='y', alpha=0.4)

plt.show()



* 1. Using *scipy*, test the hypothesis “𝑘NN is statistically superior to naïve Bayes regarding

accuracy”, asserting whether is true.

from scipy import stats

# H0: 𝑘NN is statistically equal to Naïve Bayes regarding accuracy

# H1: 𝑘NN is statistically superior to Naïve Bayes regarding accuracy

t\_statistic, p\_value = stats.ttest\_rel(knn\_accuracies, nb\_accuracies, alternative='greater')

alpha = 0.05  # Significance level

if p\_value < alpha:

    print('Para níveis de signicância até 0.05, a hipótese nula (𝑘NN é estatisticamente igual a Naïve Bayes em termos de precisão)'

          '\né rejeitada, ou seja, "𝑘NN is statistically superior to Naïve Bayes regarding accuracy" confirma-se.')

else:

    print('Para níveis de signicância até 0.05, a hipótese nula (𝑘NN é estatisticamente igual a Naïve Bayes em termos de precisão)',

          '\nnão pode ser rejeitada, ou seja, "𝑘NN is statistically superior to Naïve Bayes regarding accuracy" é falso.')

Output:

Para níveis de signicância até 0.05, a hipótese nula (𝑘NN é estatisticamente igual a Naïve Bayes em termos de precisão) não pode ser rejeitada, ou seja, "𝑘NN is statistically superior to Naïve Bayes regarding accuracy" é falso.

1. [2.5v] 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

from sklearn.metrics import confusion\_matrix

# Initialize confusion matrices

knn1\_cumulative = np.array([[0,0,0], [0,0,0], [0,0,0]])

knn5\_cumulative = np.array([[0,0,0], [0,0,0], [0,0,0]])

classes = ["Hernia", "Normal", "Spondylolisthesis"]

knn1 = KNeighborsClassifier(n\_neighbors=1, weights='uniform', metric='euclidean')

knn5 = KNeighborsClassifier(n\_neighbors=5, weights='uniform', metric='euclidean')

# Iterate through each fold of stratified cross-validation

for train\_index, test\_index in stratified\_cv.split(X, y):

    X\_train, X\_test = X.iloc[train\_index], X.iloc[test\_index]

    y\_train, y\_test = y.iloc[train\_index], y.iloc[test\_index]

    # Train both classifiers

    knn1.fit(X\_train, y\_train)

    knn5.fit(X\_train, y\_train)

    # Make predictions and calculate confusion matrix for 𝑘NN1

    knn1\_pred = knn1.predict(X\_test)

    knn1\_cm = confusion\_matrix(y\_test, knn1\_pred, labels=classes)

    knn1\_cumulative += np.array(knn1\_cm)

    # Make predictions and calculate confusion matrix for 𝑘NN5

    knn5\_pred = knn5.predict(X\_test)

    knn5\_cm = confusion\_matrix(y\_test, knn5\_pred, labels=classes)

    knn5\_cumulative += np.array(knn5\_cm)

# Calculate the difference between cumulative confusion matrices

difference\_matrix = knn1\_cumulative - knn5\_cumulative

print("Difference between cumulative confusion matrices (𝑘NN1 - 𝑘NN5):\n\n",

      pd.DataFrame(difference\_matrix, index=classes, columns=classes),

      "\n\nNote:\tColumns are predicted values\n\tLines are test values")

Output:

Difference between cumulative confusion matrices (𝑘NN1 - 𝑘NN5):

Hernia Normal Spondylolisthesis

Hernia -2 2 0

Normal -5 2 3

Spondylolisthesis 0 1 -1

Note: Columns are predicted values

Lines are test values

**Comentário** sobre os resultados obtidos:

* Para a classe 'Hernia', o modelo com k=5 previu corretamente mais 2 instâncias, enquanto o modelo com k=1 previu incorretamente mais 2 instâncias de 'Hernia' como 'Normal'.
* Para a classe 'Normal', o modelo com k=1 previu corretamente mais 2 instâncias e previu incorretamente mais 3 instâncias de 'Normal' como 'Spondylolisthesis', no entanto, o modelo com k=5 previu incorretamente mais 5 instâncias de 'Normal' como 'Hernia'.
* Para a classe 'Spondylolisthesis', o modelo com k=5 previu corretamente mais 1 instância, enquanto o modelo com k=1 previu incorretamente mais 1 instância de 'Spondylolisthesis' como 'Normal'.

Assim, pode-se concluir que o modelo com k=5 teve melhor performance para as classes 'Hernia' e 'Spondylolisthesis', enquanto o modelo com k=1 teve melhor performance para a classe 'Normal'.

1. [1.5v] Considering the unique properties of *column\_diagnosis*, identify three possible difficulties of naïve Bayes when learning from the given dataset.

* Assume independência condicional entre as variáveis. Essa suposição pode ser muito restritiva, pois nem sempre é verdadeira.
* Assume que os dados seguem uma distribuição gaussiana (normal), o que pode não ser apropriado caso os dados não obedeçam a essa distribuição.
* O uso de um conjunto de observações limitado (pode levar a estimativas inadequadas ou probabilidades nulas).

# END