Aprendizagem 2021/22

Homework I – Group 6

I. Pen-and-paper

1) Bayesian classifier training method

To train this Bayesian classifier, we used the maximum *a posteriori* estimation for the class output prediction, which is defined by:

$$\boldsymbol{MAP = arg\,max_C\, P(C|x)}$$

$$\boldsymbol{P(C|x=(y_1,y_2,y_3,y_4)) = \frac{P(x=(y_1,y_2,y_3,y_4)|C)\times P(C)}{P(x=(y_1,y_2,y_3,y_4))}}$$

Assuming independence between $\{y_1\}$, $\{y_2\}$ and $\{y_3, y_4\}$ variable sets, we can get the following expression:

$$P(C|x=(y_1,y_2,y_3,y_4)) = \frac{P(y_1|C)\times P(y_2|C)\times P(y_3,y_4|C)\times P(C)}{P(y_1|C)\times P(y_2|C)\times P(y_3,y_4|C)\times P(C) + P(y_1|\bar{C})\times P(y_2|\bar{C})\times P(y_3,y_4|\bar{C})\times P(\bar{C})}$$

Since both $P(C=0\,|x)$ and $P(C=1|x)$ have the same denominator, the MAP can be defined by:

$$\boldsymbol{MAP = arg\,max_{C\in\{0,1\}}\{P(y_1|C)\times P(y_2|C)\times P(y_3,y_4|C)\times P(C)\}}$$

To evaluate this value, we must calculate each of these probabilities. Given that y1 and {y3, y4} are normally distributed and y2 is a categorical variable:

Normal univariate distribution

$$\boldsymbol{P(y_1|C)} = N(y|\mu,\sigma) = \frac{e^{-\frac{(y-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}},\quad \mu = \frac{1}{n}\sum_{i=1}^{n} y_i\,,\quad \sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i-\mu)^2}$$

Normal multivariate distribution

$$\boldsymbol{P(y_3,y_4|C)} = N(y_3,y_4|\mu,\Sigma) = \frac{e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}}{2\pi\sqrt{|\Sigma|}}$$

$$\mu = \begin{pmatrix}\mu_0 = \frac{1}{n}\sum_{i=1}^{n} y_{i,0}\\ \mu_1 = \frac{1}{n}\sum_{i=1}^{n} y_{i,1}\end{pmatrix},\quad \Sigma = \begin{pmatrix}\Sigma_{00} & \Sigma_{01}\\ \Sigma_{10} & \Sigma_{11}\end{pmatrix},\quad \Sigma_{a,b} = \frac{1}{n-1}\sum_{i=1}^{n}(y_{i,a}-\mu_0)\,(y_{i,b}-\mu_1)$$

Relative Frequency

$$\boldsymbol{P(y_2|C) = \frac{\#Observations\ of\ y_2}{\#Observations\ of\ C}}$$

As an example, these would be the steps required to predict the class of $x_1 = (y_1 = 0.6, y_2 = A, y_3 = 0.2, y_4 = 0.4)$:

$$\boldsymbol{arg\,max_{C\in\{0,1\}}\{P(y_1=0.6|C)\times P(y_2=A|C)\times P(y_3=0.2,y_4|C=0.4)\times P(C)\}}$$

$$\mu_{1|C=0} = \frac{1}{4}(0.6+0.1+0.2+0.1) = 0.25$$

$$\sigma_{1|C=0} = \sqrt{\frac{1}{4-1}[(0.6-0.25)^2+(0.1-0.25)^2+(0.2-0.25)^2+(0.1-0.25)^2]} = \sqrt{\frac{0.17}{3}} \approx 0.2380$$

$$P(y_1|C=0) = N(y_{1|C=0}|0.25, 0.2380) \text{ and similarly } P(y_1|C=1) = N(y_{1|C=1}|0.05, 0.2881)$$

$$P(y_2|C=0) = \frac{2}{4} = \frac{1}{2} \text{ and similarly } P(y_2|C=1) = \frac{1}{6}$$

$$\mu_{3,4|C=0} = \begin{pmatrix} \frac{1}{4}(0.2 - 0.1 - 0.1 + 0.8) \\ \frac{1}{4}(0.4 - 0.4 + 0.2 + 0.8) \end{pmatrix} = \begin{pmatrix} 0.2 \\ 0.25 \end{pmatrix}$$

$$\Sigma_{00} = \frac{(0.2-0.2)(0.2-0.25) + (-0.1-0.2)(-0.1-0.25) + (-0.1-0.2)(-0.2-0.25) + (0.8-0.2)(0.80-0.25)}{4-1}$$

$$P(y_3, y_4|C=0) = N\left(y_3, y_{4|C=0} \middle| \begin{pmatrix} 0.2 \\ 0.25 \end{pmatrix}, \begin{pmatrix} 0.18 & 0.18 \\ 0.18 & 0.25 \end{pmatrix}\right), \text{and similarly}$$

$$P(y_3, y_4|C=1) = N\left(y_3, y_{4|C=1} \middle| \begin{pmatrix} 0.1167 \\ 0.0833 \end{pmatrix}, \begin{pmatrix} 0.1097 & 0.1223 \\ 0.1223 & 0.2137 \end{pmatrix}\right)$$

Having all probabilities for both C=0 and C=1, we can find the arg max for x_1 and predict its class.

2) Using the formula obtained in **1)**, the following values were obtained:

x_i	$P(C)P(y_1\|C)P(y_2\|C)P(y_3,y_4\|C)$		Predicted class	Real class
	C = 0	**C = 1**		
x_1	**0.1373**	0.0271	0	0
x_2	0.0633	**0.2610**	1	0
x_3	**0.2317**	0.0735	0	0
x_4	0.0704	**0.0831**	1	0
x_5	0.1925	**0.2294**	1	1
x_6	0.0190	**0.2430**	1	1
x_7	0.0008	**0.1207**	1	1
x_8	0.1778	**0.2033**	1	1
x_9	**0.0598**	0.0257	0	1
x_{10}	0.0303	**0.3208**	1	1

The resulting confusion matrix is drawn as follows:

		Predicted class	
		Positive	**Negative**
Real class	**Positive**	**#TP = 5**	**#FN = 1**
	Negative	**#FP = 2**	**#TN = 2**

3) The F1 score is calculated as follows:

$$\textbf{F1 Score} = \frac{\mathbf{2}}{\frac{\mathbf{1}}{\textbf{Recall}} + \frac{\mathbf{1}}{\textbf{Precision}}} = \frac{2}{\frac{\#TP + \#FN}{\#TP} + \frac{\#TP + \#FP}{\#TP}} = \frac{2}{\frac{6}{5} + \frac{7}{5}} = \frac{\mathbf{10}}{\mathbf{13}} \approx 0.7692$$

4) In **1)**, our calculations only required the computation of the numerators of the posterior probabilities. However, to find the actual value of each posterior probability and, as a result, the optimal decision probability threshold for training accuracy, computing the expression's denominator was also needed. The chosen thresholds corresponded to $\{P(C = 1|x_i): i = 1, 2, ..., 10\} \cup \{0, 1\}$, where the predicted class was 1 only if $P(C = 1|x_i) >$ threshold, for each of the 12 thresholds. These were used since they correspond to transition values when applied to the x_i that originated them, giving us a more precise idea of the ideal threshold. The results are shown in the following table:

x_i	Real class	$P(C = 1\|x_i)$	0	0.1650	0.2409	0.3007	0.5335	0.5413	0.5437	0.8049	0.9136	0.9275	0.9363	1
x_1	0	0.1650	FP	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**
x_2	0	0.8049	FP	FP	FP	FP	FP	FP	FP	**TN**	**TN**	**TN**	**TN**	**TN**
x_3	0	0.2409	FP	FP	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**
x_4	0	0.5413	FP	FP	FP	FP	FP	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**	**TN**
x_5	1	0.5437	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	FN	FN	FN	FN	FN	FN
x_6	1	0.9275	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	FN	FN	FN
x_7	1	0.9363	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	FN	FN
x_8	1	0.5335	**TP**	**TP**	**TP**	**TP**	FN	FN	FN	FN	FN	FN	FN	FN
x_9	1	0.3007	**TP**	**TP**	**TP**	FN	FN	FN	FN	FN	FN	FN	FN	FN
x_{10}	1	0.9136	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	**TP**	FN	FN	FN	FN
$\textbf{Accuracy} = \frac{\#TP + \#TN}{\#Observ.}$			0.6	0.7	**0.8**	0.7	0.6	0.7	0.6	0.7	0.6	0.5	0.4	0.4

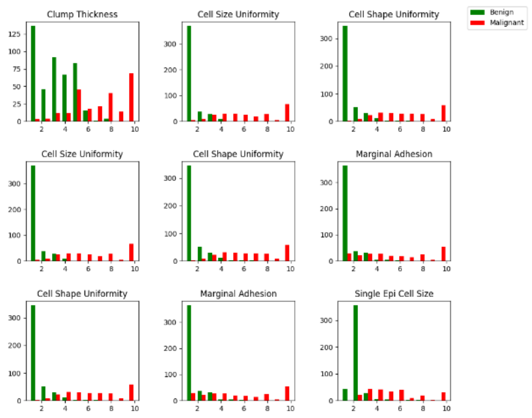
From this table, we conclude that the optimal decision probability threshold for training accuracy among those tested is **0.2409**, with an accuracy of **0.8**.

Aprendizagem 2021/22

Homework I – Group 6

II. Programming and critical analysis

5) Using the code shown in the appendix, the following histograms were created:



6) After running a 10-fold cross validation with **seed = 6**, the following results were obtained:

k	Accuracy
3	0.9707
5	0.9707
7	0.9736

The less susceptible value of k ∈ {3,5,7} to the overfitting risk is **k = 3**. This comes from the fact that, the lower the value of k, the smaller the number of neighbours the model will consider. As such**, the predictions will become too localised and highly prone to the influence of biased samples** (since they'll consider only the nearest neighbours), which might give high accuracy values if we test the model with the sample set used to train it but return low accuracies when testing the model with new samples. Thus, considering the much bigger size of the training data when compared to these values of k, **higher values of k are recommended to balance the effects of uncommon outcomes.**

7) Given that accuracy estimates are normally distributed, we tested the hypothesis "3NN is statistically superior to Naïve Bayes (multinomial assumption)" by computing the p-value using Student's t-test with the following parameters:

- $\mu_0 \approx 0.9707$ is the 3NN's average accuracy (obtained in **6)**) and $\mu_1 \approx 0.9077$ is the Naïve Bayes' average accuracy;
- H_0 (null hypothesis): $\mu_0 - \mu_1 = 0$ and H_1 (alternative hypothesis): $\mu_0 - \mu_1 > 0$;

This computation returned a p-value $\approx 1.4808 \times 10^{-5}$, which is far lower than the usual significance thresholds (1%, 5% and 10%), meaning we should accept the alternative hypothesis. In conclusion, in this context, **3NN is indeed, statistically, superior to Naïve Bayes (multinomial assumption).**

8) The histograms that were plotted in **5)** suggest there's some positive correlation between (at least, some of the) variables, which implies the dependence between them, and, in turn, the Naïve Bayes assumption, might be flawed, resulting in a lower accuracy when using this training model.

The statistical superiority of kNN when compared to Naïve Bayes found in **7)** may come from the assumption, in the Naïve Bayes model, that the data has a multinomial distribution, which is never tested. As such, the lower the accuracy of the assumption, the lower the accuracy of the prediction itself.

III. APPENDIX

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import KFold
from matplotlib import pyplot as plt
from scipy import stats
from scipy.io import arff
import numpy as np
import pandas as pd

def getHistograms():
    mask = df["Class"] == "benign"
    df_ben, df_mal = df[mask], df[~mask]
    features = list(df.columns)[:-1]
    fig, axs = plt.subplots(3, 3)
    fig.tight_layout()
    for i in range(3):
        for j in range(3):
            axs[i,j].hist(
                      [df_ben[features[i+j]],df_mal[features[i+j]]],bins=10,color=["green","red"],
                      label=[" Benign","Malignant"])
            axs[i,j].set_title(features[i+j].replace("_"," "))
    handles, labels = axs[i,j].get_legend_handles_labels()
    fig.legend(handles, labels, loc='upper right')
    plt.show()

def classifier_accuracies (clf):
    acc = []
    for train,test in kf.split(df):
        # Split data in train and test
        df_train = pd.DataFrame([df.iloc[i] for i in train])
        df_train_features = df_train.iloc[:,:-1]
        df_train_label = df_train.iloc[:,[-1]]
        df_train_label_01 = [1 if elem == "malignant" else 0 for elem in df_train_label["Class"]]

        df_test = pd.DataFrame([df.iloc[i] for i in test])
        df_test_features = df_test.iloc[:,:-1]
        df_test_label = df_test.iloc[:,[-1]]
        df_test_label_01 = [1 if elem == "malignant" else 0 for elem in df_test_label["Class"]]

        clf.fit(df_train_features.values,df_train_label_01)
        acc.append(clf.score(df_test_features.values,df_test_label_01))

    return acc

def knnVSnaive():
    knn3 = KNeighborsClassifier(n_neighbors = 3)
    naive_bayes = MultinomialNB()
    knn_acc = classifier_accuracies(knn3)
    naive_acc = classifier_accuracies(naive_bayes)
    _, pvalue = stats.ttest_rel(knn_acc, naive_acc, alternative="greater")

data = arff.loadarff('breast.w.arff')
df = pd.DataFrame(data[0])
df['Class'] = df['Class'].str.decode('utf-8')
kf = KFold(n_splits=10,shuffle=True,random_state=6)

getHistograms() # Exercise 5
knn_accuracies = [np.mean(classifier_accuracies(KNeighborsClassifier(n_neighbors = k)) for k in
[3,5,7]] # Exercise 6
knnVSnaive() # Exercise 7
```

END