**I. Pen-and-paper**

1. Class = z

; ; p(z=0|x) = 1- p(z=1|x)

p(x|z=0) = p(y1|z=0)p(y2|z=0)p(y3,y4|z=0); p(x|z=1)= p(y1|z=1)p(y2|z=1)p(y3,y4|z=1)

p(z)= 4/10 (=0,4) se z=0; 6/10 (=0,6) se z=1

**p(y1|z=0):**

;

**p(y1|z=1):**

;

p(y2=α|z=0)= 2/4 (=0,5) se α=A; ¼ (=0,25) se α=B; ¼ (=0,25) se α=C.

p(y2= α|z=1)= 1/6 (=0,1667) se α=A; 2/6 (=0,3333) se α=B; 3/6 (=0,5) se α=C.

**p(y3,y4|z=0):**

*; ;*

*;* Det(∑) = 0,18\*0,25-0,18\*0,18 = 0,0126

*;*

**p(y3,y4|z=0) =**  )

**p(y3,y4|z=1):**

*; ;*

*;* Det(∑) = 0,1097\*0,2137 - 0,1223\*0,1223= 0,0085

*;*

**p(y3,y4|z=1) = )**

**Conclusion:**

* ; p(z=0|x) = 1- p(z=1|x)
* p(x|z=0) = p(y1|z=0)p(y2|z=0)p(y3,y4|z=0); p(x|z=1)= p(y1|z=1)p(y2|z=1)p(y3,y4|z=1),where:
* p(z)= 4/10 (=0,4) se z=0; 6/10 (=0,6) se z=1
* ;
* p(y2=α|z=0)= 2/4 (=0,5) se α=A; ¼ (=0,25) se α=B; ¼ (=0,25) se α=C.
* p(y2= α|z=1)= 1/6 (=0,1667) se α=A; 2/6 (=0,3333) se α=B; 3/6 (=0,5) se α=C.
* p(y3,y4|z=0) =
* p(y3,y4|z=1) = )

1. Replacing y1, y2, y3, y4 for each cada xi, and taking into account that p(z=0)=0,4 e p(z=1)=0,6:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | p(y1|z=0) | p(y1|z=1) | p(y2|z=0) | p(y2|z=1) | p(y3, y4|z=0) | p(y3,y4|z=1) | p(x,z=0) | p(x,z=1) | p(x) | p(z=1,x) | ^z | z |
| x1 | 0,5686 | 0,2239 | 0,5 | 0,1667 | 1,2074 | 1,2119 | 0,1373 | 0,0271 | 0,1644 | 0,1650 | 0 | 0 |
| x2 | 1,3741 | 1,3641 | 0,25 | 0,3333 | 0,4603 | 0,9567 | 0,0633 | 0,2610 | 0,3243 | 0,8049 | 1 | 0 |
| x3 | 1,6393 | 1,2092 | 0,5 | 0,1667 | 0,7066 | 0,6079 | 0,2317 | 0,0735 | 0,3052 | 0,2409 | 0 | 0 |
| x4 | 1,3741 | 1,3641 | 0,25 | 0,5 | 0,5124 | 0,2030 | 0,0704 | 0,0831 | 0,1535 | 0,5413 | 1 | 0 |
| x5 | 1,6393 | 0,9503 | 0,25 | 0,3333 | 1,1743 | 1,2071 | 0,1925 | 0,2294 | 0,4219 | 0,5437 | 1 | 1 |
| x6 | 0,5686 | 1,2092 | 0,25 | 0,5 | 0,3338 | 0,6698 | 0,0190 | 0,2430 | 0,2620 | 0,9275 | 1 | 1 |
| x7 | 0,1162 | 0,6620 | 0,25 | 0,5 | 0,7066 | 0,6079 | 0,0082 | 0,1207 | 0,1289 | 0,9363 | 1 | 1 |
| x8 | 1,6393 | 1,2092 | 0,25 | 0,3333 | 1,0847 | 0,8408 | 0,1778 | 0,2033 | 0,3812 | 0,5335 | 1 | 1 |
| x9 | 1,3741 | 0,6620 | 0,5 | 0,1667 | 0,2174 | 0,3880 | 0,0598 | 0,0257 | 0,0854 | 0,3007 | 0 | 1 |
| x10 | 0,2807 | 0,9503 | 0,25 | 0,5 | 1,0804 | 1,1252 | 0,0303 | 0,3208 | 0,3511 | 0,9136 | 1 | 1 |

Ex (x1):

; p(y1|z=1)=

p(y2|z=0) = p(A|z=0)=0,5; p(y2|z=1)=p(A|z=1)=0,1667

p(y3,y4|z=0) = = 1,2074

p(y3,y4|z=1) = )=1,2119

p(z=1|x1)=

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Prev.** | |
| **0** | **1** |
| **Real** | **0** | TN=2 | FP=1 |
| **1** | FN=2 | TP=5 |

1. ;;

;

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Probability Threshold (and corresponding ^z) | | | | | | | | | |
|  | p(z=1|x) | 0,2 | 0,25 | 0,3 | 0,35 | 0,40 | 0,45 | 0,50 | 0,55 | 0,60 |
| x1 | 0,1650 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| x2 | 0,8049 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| x3 | 0,2409 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| x4 | 0,5413 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| x5 | 0,5437 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| x6 | 0,9275 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| x7 | 0,9363 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| x8 | 0,5335 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| x9 | 0,3007 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| x10 | 0,9136 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | Accuracy | 0,6 | 0,8 | 0,8 | 0,7 | 0,7 | 0,7 | 0,7 | 0,6 | 0,6 |

R: By analysis, the threshold that optimizes training accuracy is approximately 0,275 (average between 0,25 and 0,3 thresholds, both of which give maximum training accuracy). However, there is a small number of training samples, so this optimal threshold may not reflect that of the entire population.

**II. Programming and critical analysis**

1. Taking into account that the left bar, in each two-bin set, corresponds to the amount of samples of each value (1-10) that occur when the tumor is benign; and the right bar to the amount of samples of each value (1-10) that occur when the tumor is malign:



Average testing accuracy for k= {3: 0.9693094629156012, 5: 0.9765984654731458, 7: 0.9736572890025575}

Testing variance for k= {3: 0.0005350425462037332, 5: 0.00033885994278936107, 7: 0.0002737968310864877}

Average training accuracy for k= {3: 0.9793405894971002, 5: 0.9803164640766928, 7: 0.9790148565980772}

Training variance for k= {3: 1.1164339652572781e-05, 5: 7.881279576867631e-06, 7: 1.0235208147054918e-05}

[average\_training\_accuracy+training\_variance - average\_testing\_accuracy+testing\_variance] for k= {3: 0.009507248374947874, 5: 0.0033870199403345236, 7: 0.00509400597258014}

R: k=5 has the least difference between training and testing accuracies; therefore it is less susceptible to the overfitting risk, as has neither too little neighbors to give a too-close neighbor too much influence on the results nor too many neighbors to make “noise”.

1. Average k=3 accuracy: {testing: 0.9693094629156012; training: 0.9793405894971002};

k=3 variance: {testing: 0.0005350425462037332; training: 1.1164339652572781e-05};

k=3 average\_accuracy+variance: {testing: 0.9698445054618049 ; training: 0.9793517538367528};

Multinomial Bayesian variance: {testing: 0.00118111916465974 ; training: 1.9980029609651145e-05};

Average Multinomial Bayesian accuracy: {testing: 0.9049019607843135; training: 0.9054837001138741};

Multinomial Bayesian average\_accuracy+variance: {testing: 0.9060830799489733; training: 0.9055036801434838};

Training\_pvalue= 4.46e-13 < 0.05 => Cannot Reject Null Hypothesis

Testing\_pvalue= 0.00017 < 0.05 => Cannot Reject Null Hypothesis

R: Considering the difference between training and testing accuracies, the small variance of both models (whose trust interval still mantains the superiority of one above the other), and considering that we can reject neither testing nor training pvalues, the k=3 kNN model is statistically better than the Naïve Bayes (multinomial assumption) model.

1. The kNN model’s performance depends severely on the type, number, and weight of neighbors. Since there were so many neighbors to compare the training set to, but not enough that the training set was thrown off by irrelevant features, it was able to have a high training and testing accuracy.

The multinomial Naïve Bayes assumes that dataset features are mutually independent; however, as with seen on the histogram, plenty of features are associated with each other (Cell Size Uniformity is lowest when Marginal Adhesion is lowest for a benign cancer, and Cell Shape Uniformity is highest when Bare Nuclei is highest for a malign cancer, for example).

**III. APPENDIX**

import numpy as np; from scipy.io import arff; import pandas as pd; from sklearn import metrics

from sklearn.neighbors import KNeighborsClassifier; import plotly.graph\_objects as go

from sklearn.model\_selection import StratifiedKFold, KFold; import statistics; from scipy import stats;

from sklearn.naive\_bayes import MultinomialNB; from plotly.subplots import make\_subplots;

tableVar = pd.DataFrame(arff.loadarff("breast.w.arff")[0]).dropna()

fig = make\_subplots(rows=3, cols=3, subplot\_titles=list(tableVar.columns) )

tableVar["Class"] = tableVar["Class"].apply(lambda y : y.decode("utf-8"))

#================================================ awnser5

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "benign")]["Clump\_Thickness"] , nbinsx=20),row=1,col=1)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "malignant")]["Clump\_Thickness"] , nbinsx=20),row=1,col=1)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "benign")]["Cell\_Size\_Uniformity"] , nbinsx=20),row=1,col=2)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "malignant")]["Cell\_Size\_Uniformity"] , nbinsx=20),row=1,col=2)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "benign")]["Cell\_Shape\_Uniformity"] , nbinsx=20 ),row=1,col=3)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "malignant")]["Cell\_Shape\_Uniformity"] , nbinsx=20 ),row=1,col=3)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "benign")]["Marginal\_Adhesion"] , nbinsx=20),row=2,col=1)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "malignant")]["Marginal\_Adhesion"] , nbinsx=20),row=2,col=1)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "benign")]["Single\_Epi\_Cell\_Size"] , nbinsx=20),row=2,col=2)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "malignant")]["Single\_Epi\_Cell\_Size"] , nbinsx=20),row=2,col=2)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "benign")]["Bare\_Nuclei"] , nbinsx=20),row=2,col=3)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "malignant")]["Bare\_Nuclei"] , nbinsx=20),row=2,col=3)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "benign")]["Bland\_Chromatin"] , nbinsx=20),row=3,col=1)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "malignant")]["Bland\_Chromatin"] , nbinsx=20),row=3,col=1)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "benign")]["Normal\_Nucleoli"] , nbinsx=20),row=3,col=2)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "malignant")]["Normal\_Nucleoli"] , nbinsx=20),row=3,col=2)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "benign")]["Mitoses"] , nbinsx=20),row=3,col=3)

fig.add\_trace( go.Histogram(x=tableVar[(tableVar["Class"] == "malignant")]["Mitoses"] , nbinsx=20),row=3,col=3)

for e in range(10):

    if (e==0):fig["layout"]["xaxis"]["title"]= "Value"; fig["layout"]["yaxis"]["title"]= "Count"

    else:fig["layout"][f"xaxis{e}"]["title"]= "Value"; fig["layout"][f"yaxis{e}"]["title"]= "Count"

#comment v in order to check the rest

#fig.show()

#===================================== awnser6 ====================================================

x=[]

for i in range(len(tableVar)):x += [list(tableVar.iloc[i][0:9])]

x= np.array(x);y = list(tableVar["Class"]);y= np.array(y)

for el in range(len(y)):

    if y[el] == "benign":y[el] = 0

    else:y[el] = 1

skf = StratifiedKFold(n\_splits=10,random\_state=78,shuffle=True)

d\_acc\_test = {3: [], 5: [], 7: []};d\_acc\_train={3: [], 5: [], 7: []}

d\_p\_7\_train = {3: []}; d\_p\_7\_test = {3: []};

for train\_index, test\_index in skf.split(x, y):

    X\_train, X\_test = x[train\_index], x[test\_index]; y\_train, y\_test = y[train\_index], y[test\_index]

    for n in [3,5,7]:

        knn = KNeighborsClassifier(n\_neighbors=n); knn.fit(X\_train, y\_train)

        y\_pred\_test = knn.predict(X\_test); d\_acc\_test[n].append(metrics.accuracy\_score(y\_test, y\_pred\_test))

        y\_pred\_train = knn.predict(X\_train); d\_acc\_train[n].append(metrics.accuracy\_score(y\_train, y\_pred\_train))

        if (n == 3):

            clf = MultinomialNB(); clf.fit(X\_train, y\_train)

            d\_p\_7\_test[3].append(metrics.accuracy\_score(clf.predict(X\_test),y\_test))

            d\_p\_7\_train[3].append(metrics.accuracy\_score(clf.predict(X\_train),y\_train))

#print(d\_p\_7\_test); #print(d\_p\_7\_train) #print(d\_acc\_test); #print(d\_acc\_train)

def calculus(d):

    d\_medias = {}; d\_variancias = {}; d\_sum\_var\_medias = {}

    for e in [3,5,7]:

        d\_medias[e]=np.average(d[e]);d\_variancias[e]=statistics.variance(d[e]);d\_sum\_var\_medias[e]=d\_medias[e]+d\_variancias[e]

    #print(d\_medias); print(d\_variancias); #print("d\_sum\_var\_medias", d\_sum\_var\_medias)

    return d\_sum\_var\_medias

#print("test", calculus(d\_acc\_test)) #print("train", calculus(d\_acc\_train))

d\_dif = {}

for e in [3,5,7]:

    d\_dif[e] = calculus(d\_acc\_train)[e] - calculus(d\_acc\_test)[e]

#Resposta#print("d\_dif", d\_dif)

#================================================ awnser7 ==================================================

#Note: values reused from previous quest

media\_test\_7= np.average(d\_p\_7\_test[3]); media\_train\_7= np.average(d\_p\_7\_train[3]) #print("media\_test\_7", media\_test\_7); print("media\_train\_7", media\_train\_7)

v\_test\_7 = statistics.variance(d\_p\_7\_test[3]); v\_train\_7 = statistics.variance(d\_p\_7\_train[3]) #print("v\_test\_7", v\_test\_7); print("v\_train\_7", v\_train\_7)

sum\_var\_med\_7\_test = media\_test\_7 + v\_test\_7; sum\_var\_med\_7\_train = media\_train\_7 + v\_train\_7 #print("sum\_var\_med\_7\_test", sum\_var\_med\_7\_test); print("sum\_var\_med\_7\_train", sum\_var\_med\_7\_train)

diff\_sums\_7 = sum\_var\_med\_7\_train - sum\_var\_med\_7\_test #print(diff\_sums\_7)

value1, pvalue1 = stats.ttest\_rel(d\_p\_7\_test[3], d\_acc\_test[3]) #print("value1=",value1, "value1=",pvalue1);

value2, pvalue2 = stats.ttest\_rel(d\_p\_7\_train[3], d\_acc\_train[3]) #print("value2=",value2,"pvalue2=", pvalue2)

"""Conclusions

if pvalue1 > 0.05: print('pvalue\_test: fail to reject H0')

else:print('pvalue\_test: reject H0')

if pvalue2 > 0.05: print('pvalue\_train: fail to reject H0')

else:print('pvalue\_train: reject H0') """

**END**