**I. Pen-and-paper**

|  |  |  |
| --- | --- | --- |
|  | Class = 0 | Class = 1 |
|  |  |  |
|  |  |  |
|  |  |  |
| /y4 |  |  |

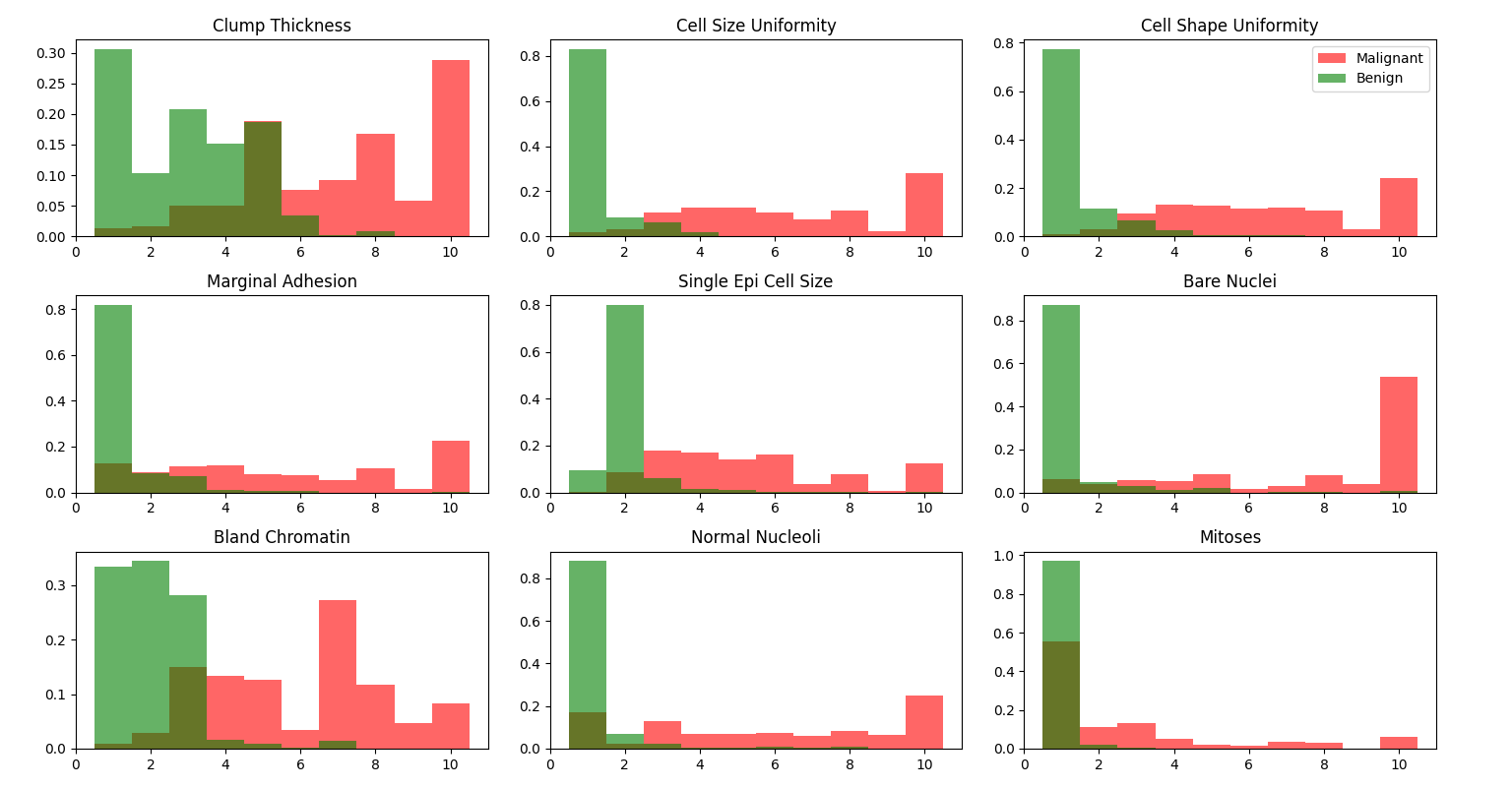
|  |  |  |  |
| --- | --- | --- | --- |
|  | Class(0) | Class(1) |  |
| x1 |  |  | 0 |
| x2 |  |  | 1 |
| x3 |  |  | 0 |
| x4 |  |  | 1 |
| x5 |  |  | 1 |
| x6 |  |  | 1 |
| x7 |  |  | 1 |
| x8 |  |  | 1 |
| x9 |  |  | 0 |
| x10 |  |  | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | | *Predicted* | |
|  |  | 0 | 1 |
| Real | 0 | 2 | 2 |
| 1 | 1 | 5 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Probabilities | Real Class | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
|  | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
|  | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
|  | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
|  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0.6996 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
|  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Threshold | - | 0.6 | 0.6 | 0.7 | 0.7 | 0.8 | 0.7 | 0.6 |

From the table we can identify that the decision probability threshold that optimizes training accuracy is 0,7. This means that we can classify as being of Class 0 if , or Class 1 otherwise.

**II. Programming and critical analysis**



1. From this data we can see that K=5 has better accuracy, and therefore is less susceptible to overfitting.

|  |  |  |  |
| --- | --- | --- | --- |
| K | 3 | 5 | 7 |
| Accuracy | 0.9692668371696506 | 0.9721867007672635 | 0.9707161125319693 |

1. The hypotheses H0, “kNN is statistically inferior or equal to Naïve Bayes (multinomial assumption)”, returned a P-Value of 0.0003537432054576055. From this we can safely reject H0 for a significance level of 0.0004 and accept H1,” kNN is statistically superior to Naïve Bayes (multinomial assumption)”.
2. Two reasons that explain the difference in performance between kNN and Naïve Bayes are:
3. Naïve Bayes assumes that every variable is independent, but looking at **7)**, that doesn’t seem to be the case;
4. Not enough data to train Naïve Bayes enough to correctly predict the class.

**III. APPENDIX**

import **matplotlib.pyplot** as **plt**

import **numpy** as **num**

from **scipy.io** import **arff**

from **scipy.stats** import **ttest\_rel**

from **sklearn.neighbors** import **KNeighborsClassifier**

from **sklearn.model\_selection** import **KFold, cross\_val\_score**

from **sklearn.naive\_bayes** import **MultinomialNB**

file = **open**("breast.w.arff", "r")

**data**, **meta** = **arff.loadarff**(file)

**#5)**

bins = [1,2,3,4,5,6,7,8,9,10,11]

for **features**, i in zip(**meta**.names()[:-1], range(1,10)):

**plt.subplot**(3,3,i)

dh = [**data**[(**data**["Class"]==b'benign')][**features**], **data**[(**data**["Class"]==b'malignant')][**features**]]

**plt.hist**(dh, bins=bins, align='left', color=['g','r'], label=['Benign','Malignant'], alpha=0.6, histtype='stepfilled', density=True)

**if** i==3: **plt.legend**()

**plt.title**(**features**.replace("\_"," "))

**plt.show**()

**#6)**

**input** = **data**[**meta**.names()[:-1]].tolist()

**output** = **data**["Class"].tolist()

**kFol** = **KFold**(n\_splits=10, shuffle=True, random\_state=47)

**crossKNN**=[[],[],[]]

for i,k in zip(range(3,8,2),range(3)):

**classifier** = **KNeighborsClassifier**(n\_neighbors=i weights='uniform', metric='euclidean')

**crossKNN**[k] = **cross\_val\_score**(**classifier**, **input**, **output**, scoring='accuracy', cv = **kFol**)

kErr = **num.average**(**crossKNN**[k])

**print**("Accuracy K={}: {}".format(i,kErr))

**#7)**

**classifier** = **MultinomialNB**()

**crossNB** = **cross\_val\_score**(**classifier**, **input**, **output**, scoring='accuracy', cv = **kFol**)

pval = **ttest\_rel**(**crossKNN**[0], **crossNB**, alternative='greater').pvalue

**print**("p-value:",pval)

**END**