**Programming Assignment02**

|  |
| --- |
| **Submission guide**  1. Write answer following questions in this file  2. Write your code using provided python script file   * You have to complete several functions under description * Do not use other packages that are not already imported in the script * Please check **TODO** |

This assignment uses MAGIC Gamma Telescope data set.

[Input]

1. fLength: continuous # major axis of ellipse [mm]

2. fWidth: continuous # minor axis of ellipse [mm]

3. fSize: continuous # 10-log of sum of content of all pixels [in #phot]

4. fConc: continuous # ratio of sum of two highest pixels over fSize [ratio]

5. fConc1: continuous # ratio of highest pixel over fSize [ratio]

6. fAsym: continuous # distance from highest pixel to center, projected onto major axis [mm]

7. fM3Long: continuous # 3rd root of third moment along major axis [mm]

8. fM3Trans: continuous # 3rd root of third moment along minor axis [mm]

9. fAlpha: continuous # angle of major axis with vector to origin [deg]

10. fDist: continuous # distance from origin to center of ellipse [mm]

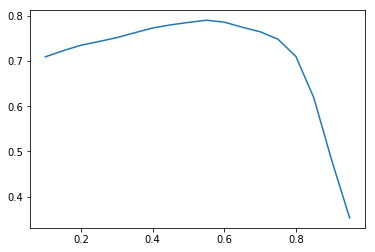
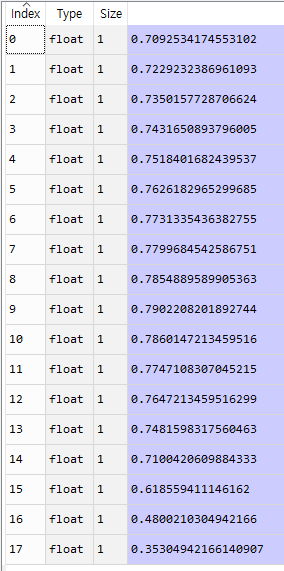
[Target]

11. class: g(gamma; signal), h(hadron; background)

The following code loads a dataset.

|  |
| --- |
| data=pd.read\_csv('https://drive.google.com/uc?export=download&id=1AoCh22pmLHhdQtYdYUAJJqOCwF9obgVO', sep='\t') |

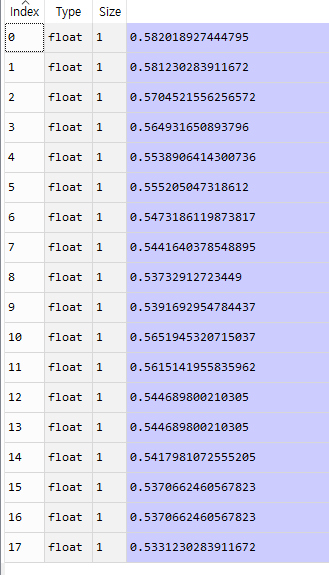
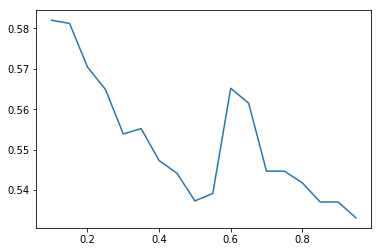
1. Logistic regression (14pts)
   1. Complete uploaded python code (10pts)
   2. Using MAGIC Gamma Telescope data set, calculate accuracy with varying cutoff for the final decision. cutoff . Draw a line plot (x=cutoff, y=accuracy). For this problem, the model is trained using trainX and accuracy is calculated using testX. (4pts)



1. Bernoulli naïve Bayes. (29 pts)
   1. Complete uploaded python code. (20pts)
   2. First, you have to binarize training set (trainX) of MAGIC Gamma Telescope data set. Each column is converted to binary variable based on the average value. If a value is greater than average, set a value as 1. Otherwise, set a value as 0. Then, using new binarized dataset, calculate (. (5pts)

|  |  |  |
| --- | --- | --- |
|  | Class g | Class h |
|  | 0.673525 | 0.246916 |
|  | 0.734746 | 0.266355 |
|  | 0.901277 | 0.0624299 |
|  | 0.407967 | 0.58 |
|  | 0.286337 | 0.698505 |
|  | 0.627509 | 0.503178 |
|  | 0.658524 | 0.449907 |
|  | 0.501723 | 0.500561 |
|  | 0.459761 | 0.163551 |
|  | 0.849483 | 0.152897 |

* 1. Based on the calculated , calculate probability of class g for each test sample (testX) and calculate accuracy for testX with varying cutoff. Prior probabilities of classes are proportional to ratios of classes in training set. Use the same of set of cutoff values used in problem (1)-B. Draw a line plot (x=cutoff, y=accuracy). (4pts)



1. Nearest neighbor (31pts)
   1. Complete uploaded python code (20pts)
   2. What are accuracies of the test set by -NN with varying parameter settings for glass dataset.

Use trainX as training set and testX as test set. (3pts)

|  |  |  |
| --- | --- | --- |
| Distance measure |  | Accuracy |
| Euclidean | 3 | 0.732912723449001 |
| 5 | 0.7268664563617245 |
| 7 | 0.7166140904311251 |
| Manhattan | 3 | 0.7302839116719243 |
| 5 | 0.7158254468980021 |
| 7 | 0.7145110410094637 |

* 1. Instead of uniform weights for the nearest neighbors, use the weights proportional to the inverse of distances of the nearest neighbors from the test point. Use trainX as training set and testX as test set. Compare two types of -NN based on the results. (4pts)

|  |  |  |
| --- | --- | --- |
| Distance measure |  | Accuracy |
| Euclidean | 3 | 0.731335436382755 |
| 5 | 0.7302839116719243 |
| 7 | 0.7245005257623555 |
| Manhattan | 3 | 0.7302839116719243 |
| 5 | 0.7208201892744479 |
| 7 | 0.7200315457413249 |

* 1. Based on mean and standard deviation values of explanatory variables using training set, standardize variables for both trainX and testX. Then, apply k-NN and weighted k-NN with varying parameter settings. Use trainX as training set and testX as test set. Compare two types of k-NN based on the results. (4pts)

[-NN]

|  |  |  |
| --- | --- | --- |
| Distance measure |  | Accuracy |
| Euclidean | 3 | 0.7618296529968455 |
| 5 | 0.7455310199789695 |
| 7 | 0.7326498422712934 |
| Manhattan | 3 | 0.7668243953732913 |
| 5 | 0.7534174553101998 |
| 7 | 0.7400105152471083 |

[weighted -NN]

|  |  |  |
| --- | --- | --- |
| Distance measure |  | Accuracy |
| Euclidean | 3 | 0.767875920084122 |
| 5 | 0.7557833859095688 |
| 7 | 0.7439537329127235 |
| Manhattan | 3 | 0.7733964248159831 |
| 5 | 0.7634069400630915 |
| 7 | 0.7518401682439537 |

1. k-means clustering (26 pts)
   1. Complete uploaded python code. (15pts)
   2. Divide the whole MAGIC Gamma Telescope data set into two clusters using k-means clustering. Calculate centroid of two clusters. (5pts)

|  |  |  |
| --- | --- | --- |
|  | Cluster 1 | Cluster 2 |
|  | 88.048 | 36.3163 |
|  | 33.7374 | 16.5572 |
|  | 3.2097 | 2.63782 |
|  | 0.255435 | 0.441104 |
|  | 0.143819 | 0.249129 |
|  | -17.2162 | 1.93828 |
|  | 25.2966 | 3.3672 |
|  | 1.03077 | -0.130358 |
|  | 20.4842 | 31.1307 |
|  | 272.596 | 155.482 |

* 1. Calculate homogeneity and completeness using true labels of MAGIC Gamma Telescope data set (6pts)

**Homogeneity= 0.006466120803295339**

**Completeness= 0.10999214355318188**