Subarna Chowdhury Soma
SJSU ID: 014549587

1.(a) [10 Points]

In HW1, you graded the following scores obtained by section A on the curve.

47, 63, 71, 39, 47, 49, 43, 37, 81, 69, 38, 13, 29, 61, 49, 53, 57, 23, 58, 17, 73, 33, 29

Section B of the course taught by the same professor scored the following points out of 100.

20, 49, 85, 17, 33, 62, 93, 64, 37, 81, 22, 18, 45, 42, 14, 39, 67, 47, 53, 73, 58, 84, 21

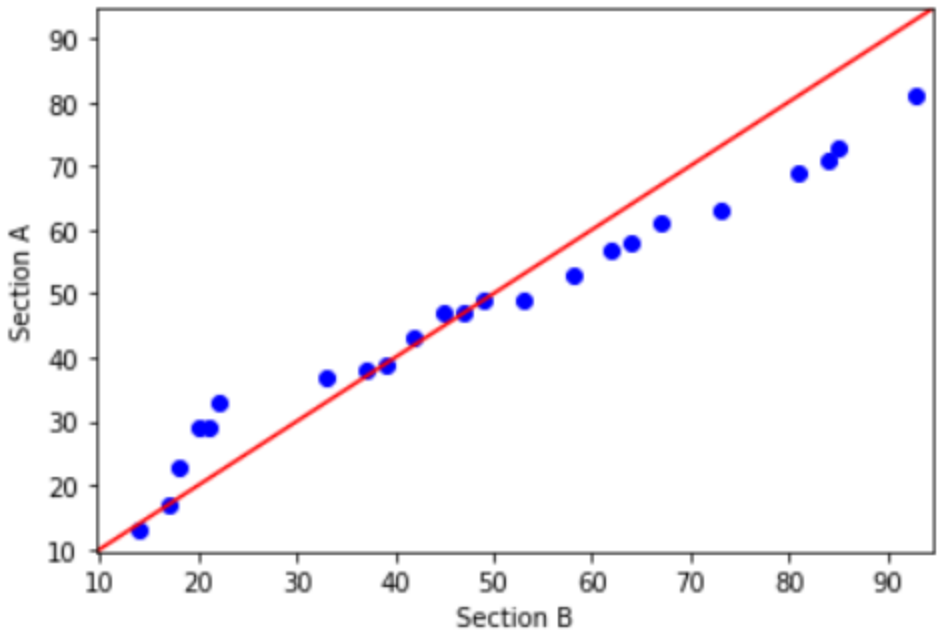
Write a python program to generate the quantile-quantile plot for the scores obtained by the students of the two sections.

Which section performed better? Justify your answer based on statistical analysis.

Answer:

File: hw2_QQplot_secAsecB.ipynb

Output:



Observation: Above is the Q-Q plot of scores from A and B section. From the plot it can be said that B section is performing better. The curve pattern suggest that are sparsely

placed in section A side whereas the central quantiles are more closely spaced in section B side. Moreover section B has 75th percentile value as 65.5, which is greater than the 75th percentile value (59.5) of section A. So, overall performance of section B is better.

(b) [10 Points]

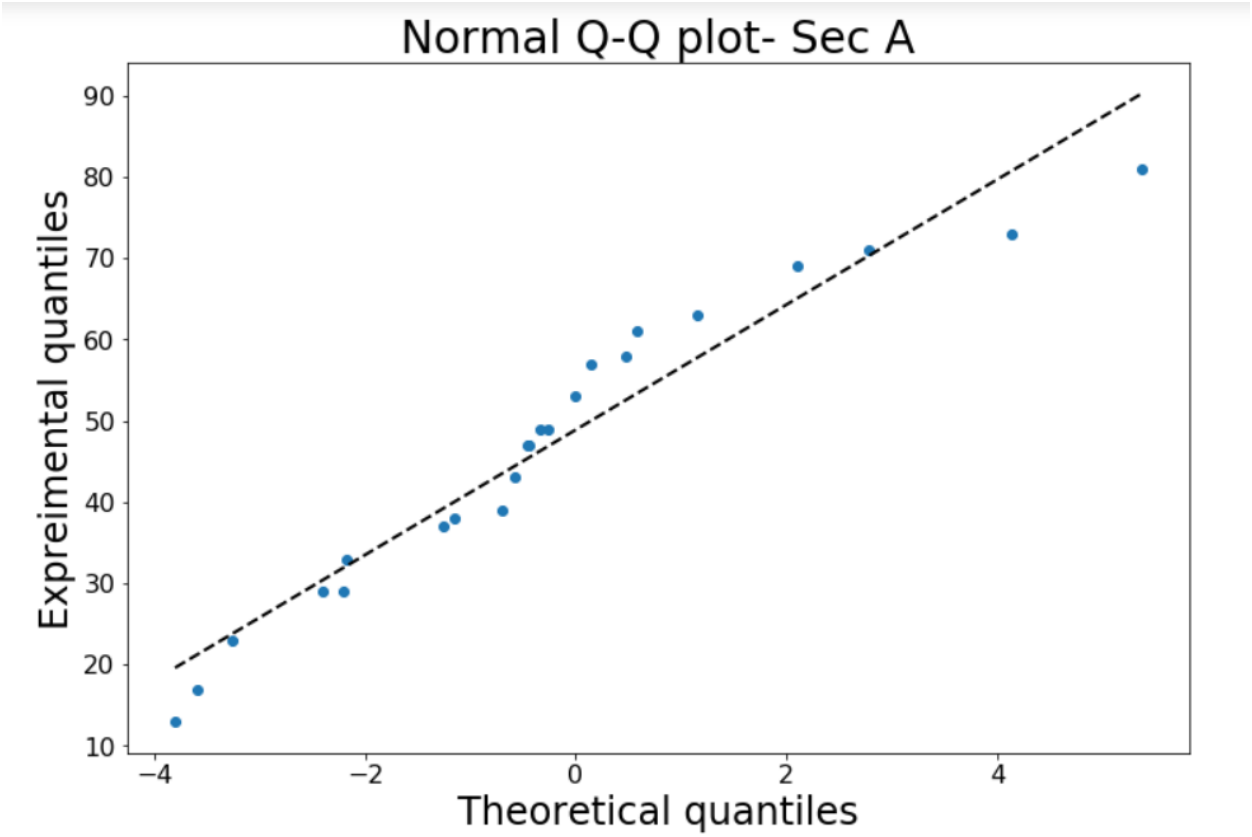
The scores obtained by which of the above two sections is closer to a normal distribution? [Hint: You can refer to https://scientificpythonsnippets.com/index.php/2-uncategorised/6-q-q-plot-in-python-to-test-if-data-is-normally-distributed and https://www.statisticshowto.datasciencecentral.com/q-q-plots/]

Answer:

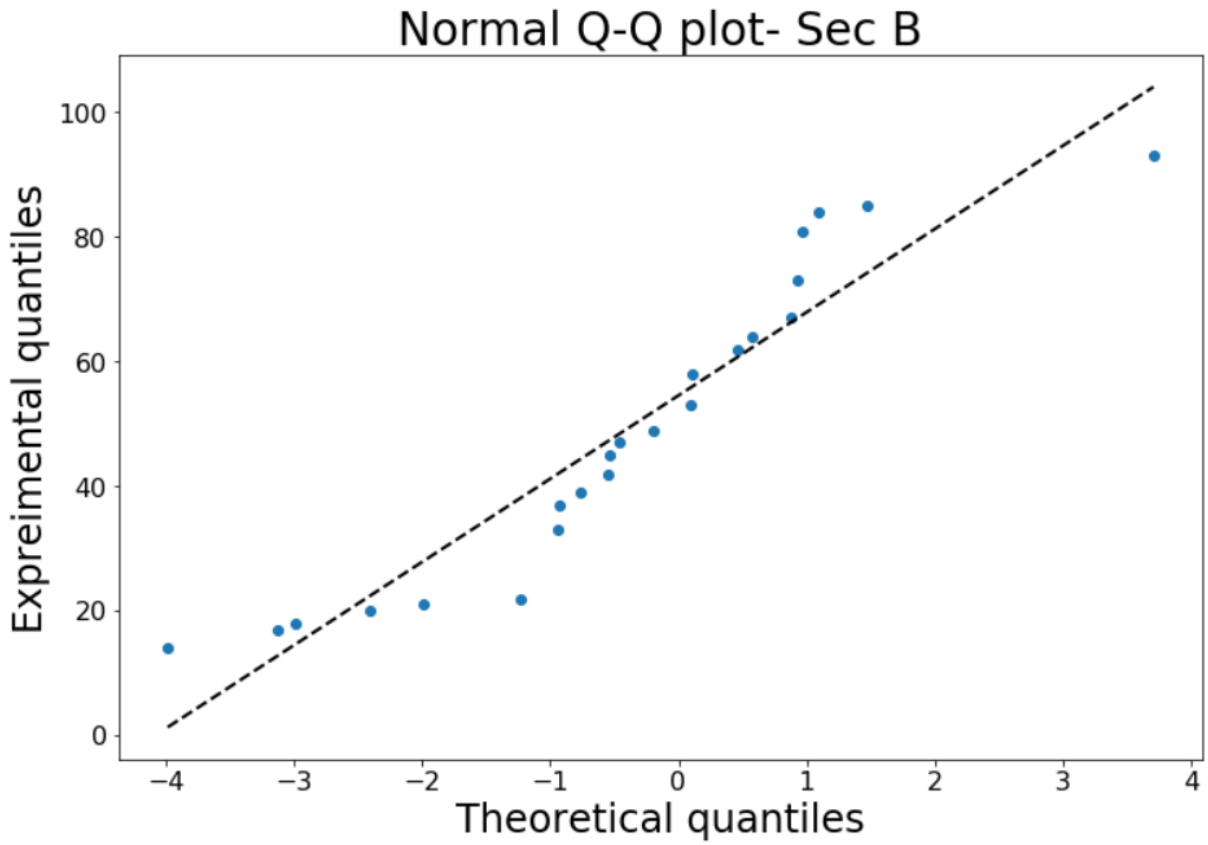
File: hw2_QQplot_NormalDistribution_secAsecB.ipynb

Plot:

1. Sec A:

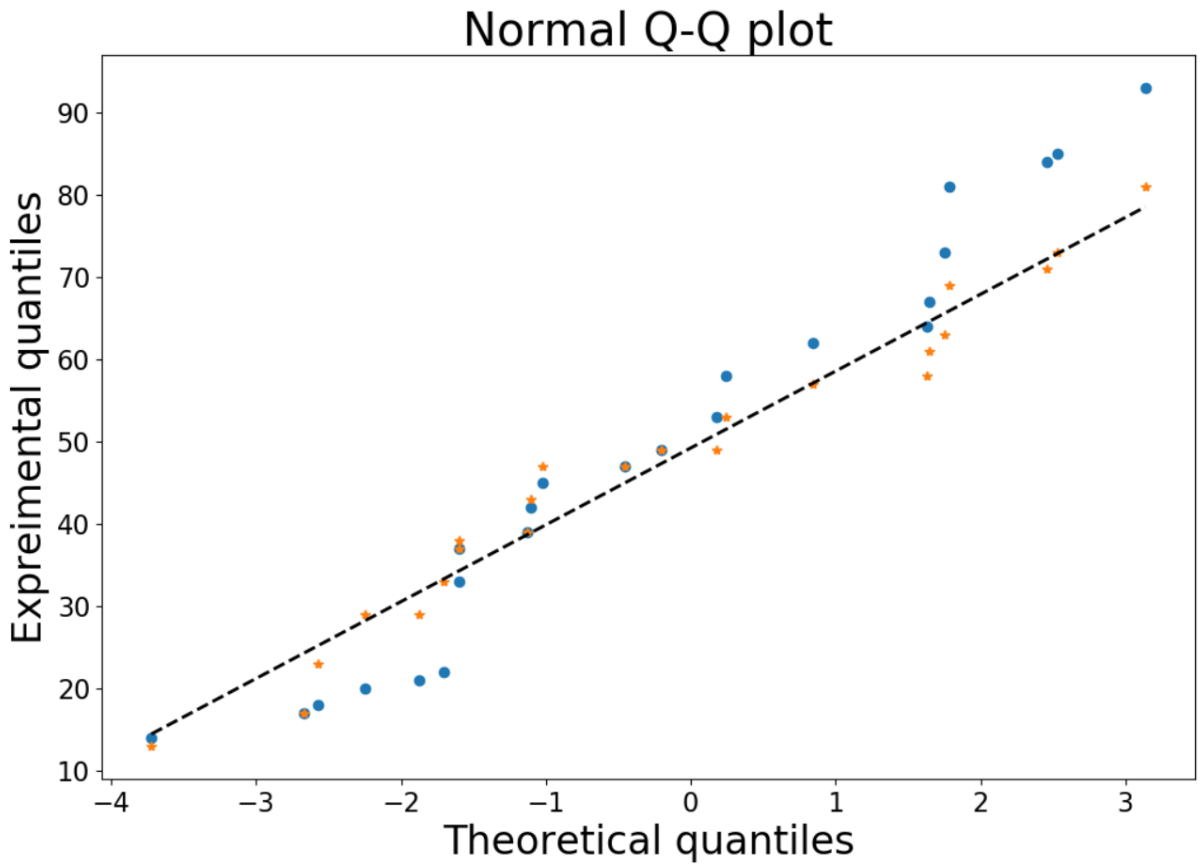


2. Sec B:



Observation: I have drawn two quantile plots using section A and section B scores. From the plots it is clear that section A data is closer to a normal distribution than section B data. Most of the section A data is approximately closely fitting the normal distribution curve whereas section B data is scattered and away from normal distribution curve compared to section A.

For better visualization I have also created a combined plot [below]. Here “*” is representing section A and “o” is representing section B.

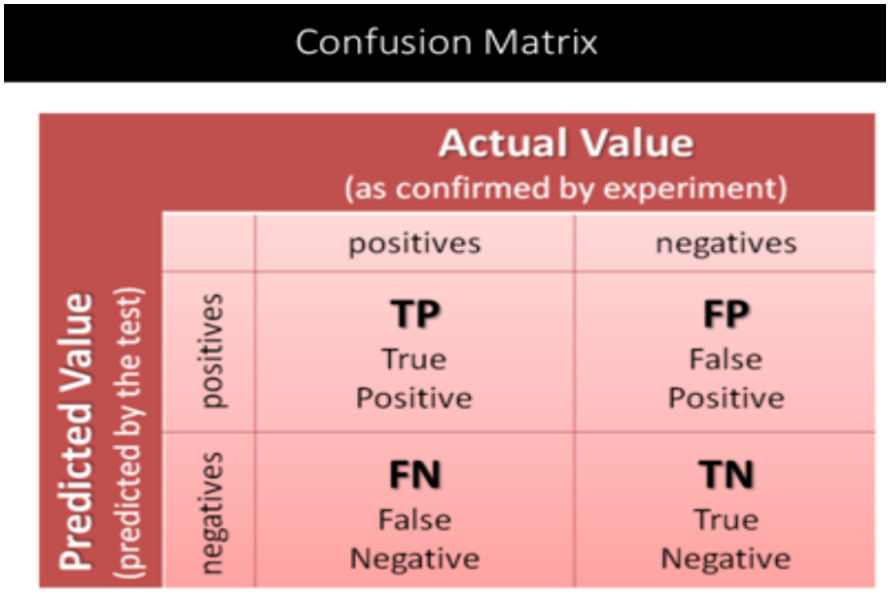


2. (a) [5 Points]

Derive an equation for accuracy in terms of Specificity and Sensitivity. Give an interpretation of the equation.

Answer:

From confusion matrix [screenshot given below from class lecture slide], we know the following formulas:



Total # of positive = TP + FN = P

Total # of negative = FP + TN = N

FPR = FP/N

Accuracy = (TP+TN)/P+N ……(1)

Sensitivity = TP/P ……..(2)

Specificity = 1 - FPR = 1- FP/N = (N-FP)/N = (FP+TN-FP)/N = TN/N ……(3)

Now from (1) =>

Accuracy = (TP/P+N) + (TN/P+N)

= (TP/P * P/P+N) + (TN/N * N/P+N)

= (Sensitivity * P/P+N) + (Sensitivity * N/P+N) [from (2), (3)].....(4)

= 1/P+N [P * Sensitivity + N * Specificity]

Finally, here is the equation for accuracy in terms of Specificity and Sensitivity,

Accuracy = 1/P+N [Total # of positive * Sensitivity + Total # of negative * Specificity]

So, if we take total of Sensitivity times total # of positive and Specificity time Total # of negative and make a ratio by dividing total # of positive and Total # of negative, we get accuracy.

Again from equation (4), we get a different perspective of relationship among accuracy, sensitivity and specificity.

P/P+N = Probability of positive's. Let's call it X

N/P+N = Probability of negative's = 1- X

So, equation (4) can be reformed as:

Accuracy = (Sensitivity) * X + (Sensitivity) * (1-X)

= (Sensitivity) * (Probability of positive's) + (Sensitivity) * (1-Probability of positive's)

Therefore, accuracy can be calculated from sensitivity and specificity if probability of positive's is known. It can be interpreted from the equation that, even if both the sensitivity and specificity are high for an event, it does not mean accuracy will be equally high as well.

2. (b) [15 Points]

Choose one of the cleaned datasets at
https://www.kaggle.com/annavictoria/ml-friendly-public-datasets

. Split it into training and test data. Apply any two ML algorithms that you learned in class. You can use R or Python to implement them. Which one of the algorithm fares better? Use as many evaluation metrics as possible to discuss the performance of the algorithms.

Answer:

Dataset: https://www.kaggle.com/uciml/adult-census-income/download

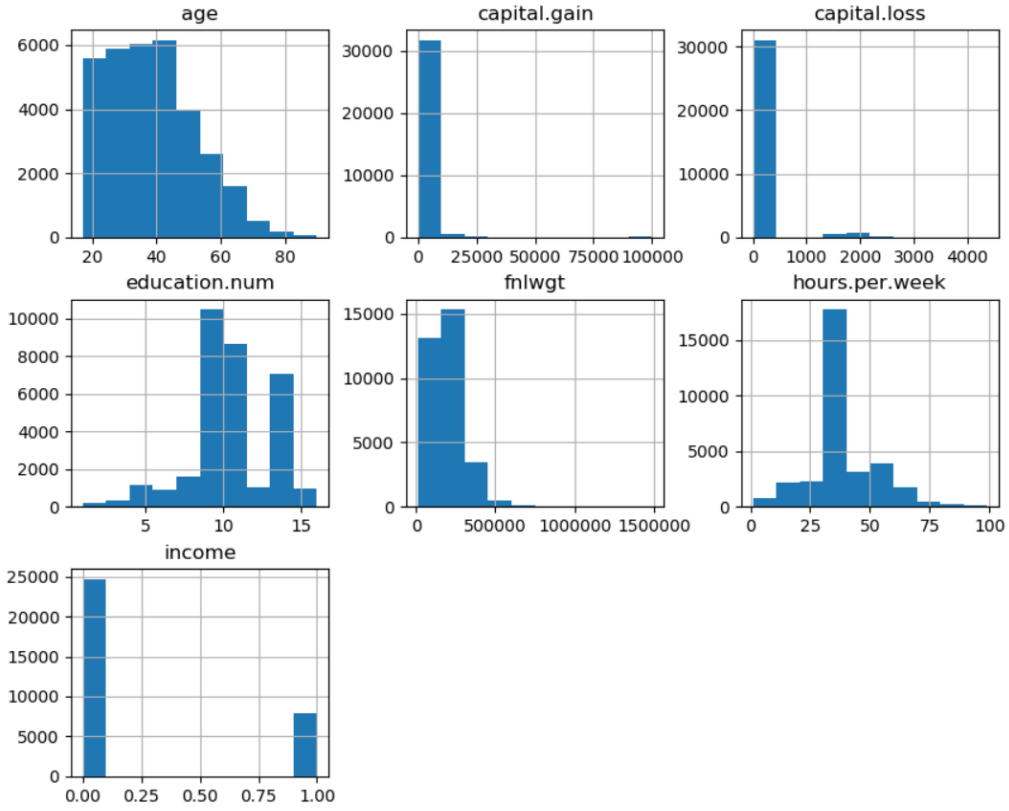
I have used adult-census-income data set for implementing and comparing my algorithms. The data set is large with 32561 rows and with size 4MB. Target column is 'income' and is classified as: [>50K, <=50K] = [1,0]. I have used Logistic regression and decision tree algorithms for classification to generate the decision boundaries between two classes. For evaluation I have splitted data set into training and test data with 70:30 ratio and calculated accuracy comparing with test data. I have also created confusion matrix for evaluating implemented algorithms.

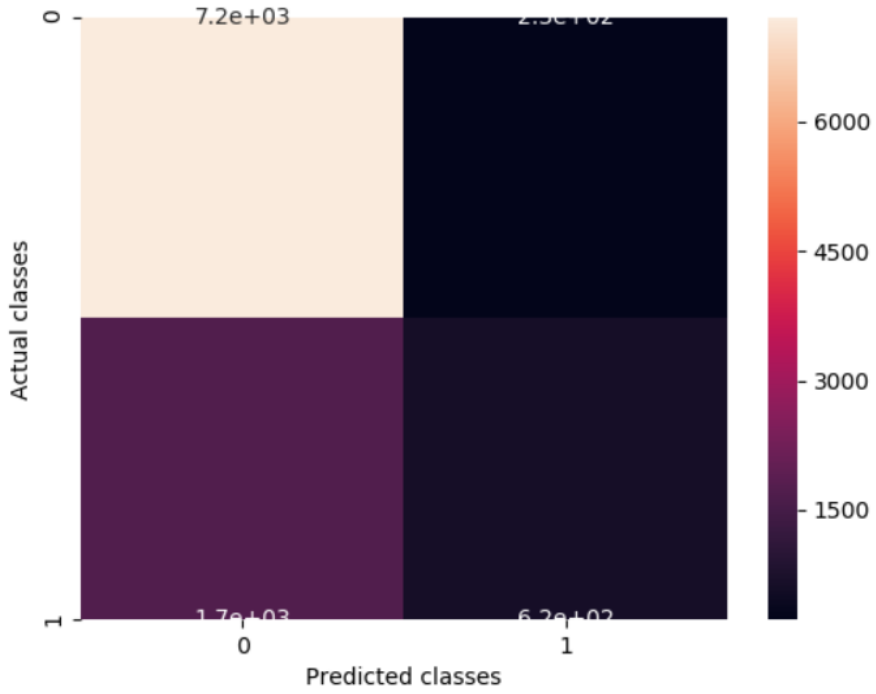
ML Algorithm 1: Logistic Regression classifier

File: KaggleData_LogisticRegression.ipynb [Replace the 'filepath' variable with your local file path and filename]

Output:

```
Total data count: 32561
Training data count: 22792
Test data count: 9769
Class distribution of training data: 0    17279
1     5513
Name: income, dtype: int64
Accuracy with down sampled data: 80.19244549083837 %
Confusion Matrix [[7210  231]
 [1704  624]]
```





ML Algorithm 2: Decision Tree classifier

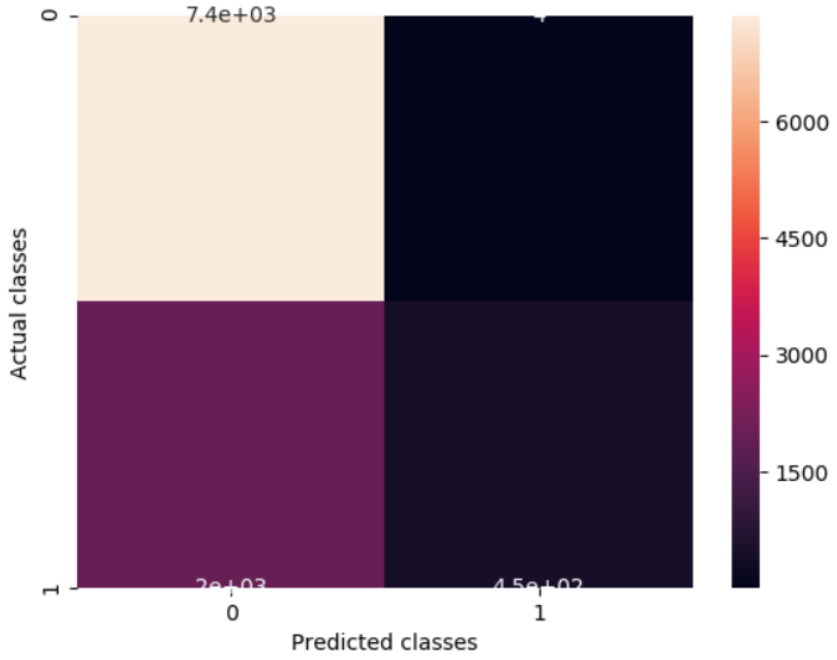
File: hw2_KaggleData_DecisionTree.ipynb [Replace the 'filepath' variable with your local file path and filename]

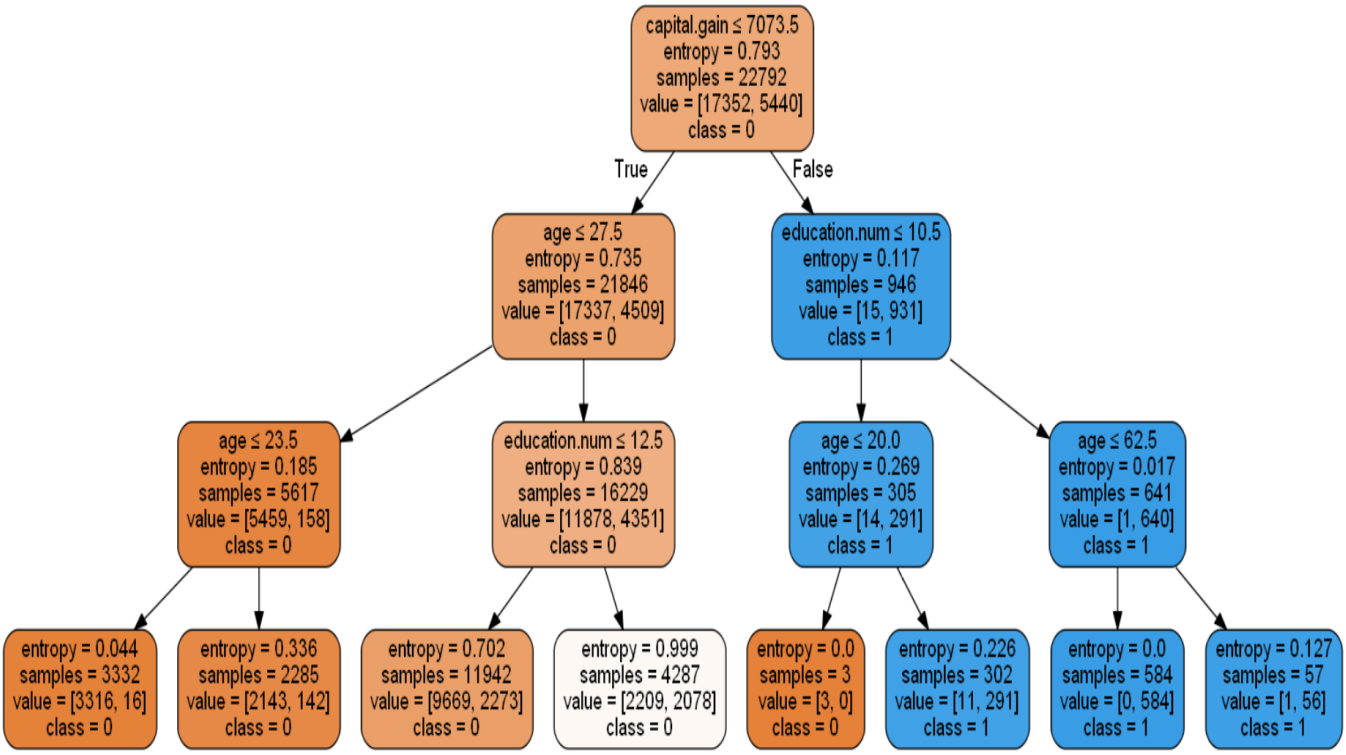
Output:

```
       capital.loss  hours.per.week native.country  income
0              4356              40  United-States       0
1              4356              18  United-States       0
2              4356              40  United-States       0
3              3900              40  United-States       0
4              3900              40  United-States       0
...             ...             ...            ...     ...
32556             0              40  United-States       0
32557             0              38  United-States       0
32558             0              40  United-States       1
32559             0              40  United-States       0
32560             0              20  United-States       0

[32561 rows x 15 columns]
Index(['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss',
       'hours.per.week'],
      dtype='object')
feature columns: ['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss', 'hours.per.week']
Accuracy: 79.95700685842972 %
Confusion Matrix [[7364    4]
 [1954  447]]
```

```
Max depth: 1, Accuracy: 0.76 (+/- 0.00)
Max depth: 2, Accuracy: 0.76 (+/- 0.01)
Max depth: 3, Accuracy: 0.76 (+/- 0.03)
Max depth: 4, Accuracy: 0.76 (+/- 0.00)
Max depth: 5, Accuracy: 0.77 (+/- 0.05)
Max depth: 6, Accuracy: 0.77 (+/- 0.03)
Max depth: 7, Accuracy: 0.77 (+/- 0.03)
Max depth: 8, Accuracy: 0.77 (+/- 0.03)
Max depth: 9, Accuracy: 0.77 (+/- 0.03)
Max depth: 10, Accuracy: 0.77 (+/- 0.03)
Max depth: 11, Accuracy: 0.77 (+/- 0.03)
Max depth: 12, Accuracy: 0.77 (+/- 0.03)
Max depth: 13, Accuracy: 0.77 (+/- 0.03)
Max depth: 14, Accuracy: 0.77 (+/- 0.04)
Max depth: 15, Accuracy: 0.76 (+/- 0.03)
Max depth: 16, Accuracy: 0.76 (+/- 0.04)
Max depth: 17, Accuracy: 0.76 (+/- 0.04)
Max depth: 18, Accuracy: 0.76 (+/- 0.04)
Max depth: 19, Accuracy: 0.75 (+/- 0.04)
```





Observation: From the accuracy calculation value, it is observed that Logistic Regression performs better with ≈81%. With Decision Tree, the accuracy is ≈80% for maximum depth 3 and entropy. Actually this large data set is not fully cleaned. More fine tuning on each column and value can be done to improve the accuracies. For decision tree algorithm, I have also implemented a tree level based comparison to show the increasing/decreasing accuracy with the change of tree level. In the confusion matrix plot, I have predicted class in X-axis and Y-axis represents the actual classes. From the observed logistic regression confusion matrix, it is seen that, 7.2e+03 times the model correctly predicted class 0 [<=50K] when the class was actually 0 and 6.3e+5 times the model has predicted class 1 [>50K] perfectly. However for decision tree confusion matrix, the model has correctly prediction class 0 [<=50K] for 7.4e+03 times and class 1 for 4.5e+03 times.

3.(a) [5 points]

Listing which problem domains are best suited for each, briefly explain in your own words, the pros and cons of

- Logistic Regression
- K-NN

- SVM
- Naïve Bayes
- Decision Trees

Answer:

Logistic Regression:

Applicable for the problems where the data obey the following properties:

1. The relation between independent and dependent variables should be linear.
2. The variance of the errors should be constant.
3. Independent variables cannot be co-linear.

Pros:

Multi-class classifications can be done.

Cons:

Outliers badly affect the accuracy. Non-linear classifications cannot be done.

K-NN:

There is no training involved in KNN, but it requires lots of computation for classification. So it should be applied when sample size is not large.

Pros:

No training time, only two parameters, k and the distance function need to be properly selected.

Cons:

KNN requires lots of computations during runtime. It is also affected by outliers.

SVM:

Applicable for labeled data. It performs well for large features with few training data.

Pros:

It is less prone to outliers.

Cons:

Sometimes appropriate kernel functions cannot be found.

Naive Bayes:

It is well suited for small data set.

Pros:

It performs fast and can work with high dimensions.

Cons:

It may not perform well if assumption of independence is violated.

Decision Trees:

Applicable for problems where the independent variables have discrete values. We form the classification tree for dependent variables with discrete values.

Pros:

It provides an easy explanation how the prediction is being made without any assumptions on the distribution of the data.

Cons:

Overfitting is the worst feature of decision trees.

(b) [15 points]

Follow the tutorial on Naïve Bayes classifier at
https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

. Write your own code for Naïve Bayes Classification of the Breast Cancer dataset that you used for HW1. Comment on the performance of Naïve Bayes as compared with Logistic Regression you used in HW1.

Answer:

File: hw2_NaiveBayesClassifier.ipynb [Replace the 'filepath' variable with your local file path and filename]

Output:

```
def main():
    ## Give your file path with name here. Now local path is given
    filename = 'C:/Users/subar/Downloads/CMPE-255 Sec 99 - Data Mining/Home Works/MyData.csv'
    data = loadCsvData(filename)
    trainData, testData = train_test_split(data, train_size=0.7, random_state=100)
    print("Split {0} rows into train={1} and test={2} rows".format(len(data), len(trainData), len(testData)))

    # prepare  naive bayes model
    summaries = class_summary(trainData)

    # test model
    predictions = getNVPredictionValues(summaries, testData)

    # Accuracy calculation
    print('Accuracy with Naive Bayes Model:', (accuracy_score(testData, predictions) * 100), "%")

main()
```

```
Split 683 rows into train=478 and test=205 rows
Accuracy with Naive Bayes Model: 96.09756097560975 %
```

Observation: On the same breast cancer data set, Naive Bayes algorithm has performed better with accuracy 96%. Accuracy with Logistic Regression was 94%.

4. [20 Points]

Manually generate the decision tree (as much as possible) for the following subset from a large dataset using the ID3 algorithm. Show the information gain computation at each stage. Then generate the decision tree programmatically using R or Python. Submit code and the decision tree so generated.

Windy?	Air Quality Good?	Hot?	Play Tennis?
No	No	No	No
Yes	No	Yes	Yes
Yes	Yes	No	Yes
Yes	Yes	Yes	No

Answer: Let PT denote Play Tennis. Here, there are 2 values: Yes 2 times and No 2 times. So, entropy of playing tennis, E(PT) = - 2/4 log (2/4) - 2/4 log(2/4) = 1.

Now, let IG denote information gain.

IG (PT, Wind) = E(PT) - ¾ E(PT, Wind = Y) - ¼ E(PT, Wind = N)

= 1 - ¾ (- ⅔ log(⅔) - ⅓ log(⅓)) - 0 = 1 - ¾ (0.389 + 0.528) = 0.313

IG (PT, Air) = E(PT) - 2/4 E(PT, Air = Y) - 2/4 E(PT, Air = N)

= 1 - ½ (-½ log (½) - ½ log (½)) - ½ (-½ log (½) - ½ log (½)) = 0

IG (PT, Hot) = E(PT) - 2/4 E(PT, Hot = Y) - 2/4 E(PT, Hot = N)

= 1 - ½ (-½ log (½) - ½ log (½)) - ½ (-½ log (½) - ½ log (½)) = 0

So, information gain with respect to wind is maximum among the 3 attributes. Therefore, wind is selected as the first level attribute (root) in the decision tree.

Now, Wind = N does not need any more branching. We need to calculate E(PT, Wind = Y) = - ⅔ log(⅔) - ⅓ log(⅓) = 0.918.

IG (PT, Wind = Y, Air) = E(PT, Wind = Y) - ⅔ E(PT, Wind = Y, Air = Y) - ⅓ E(PT, Wind = Y, Air = N) = 0.918 - ⅔ (-½ log (1/2) - ½ log (1/2)) - 0 = 0.251

IG (PT, Wind = Y, Hot) = E(PT, Wind = Y) - ⅔ E(PT, Wind = Y, Hot = Y) - ⅓ E(PT, Wind = Y, Hot = N) = 0.918 - ⅔ (-½ log (1/2) - ½ log (1/2)) - 0 = 0.251

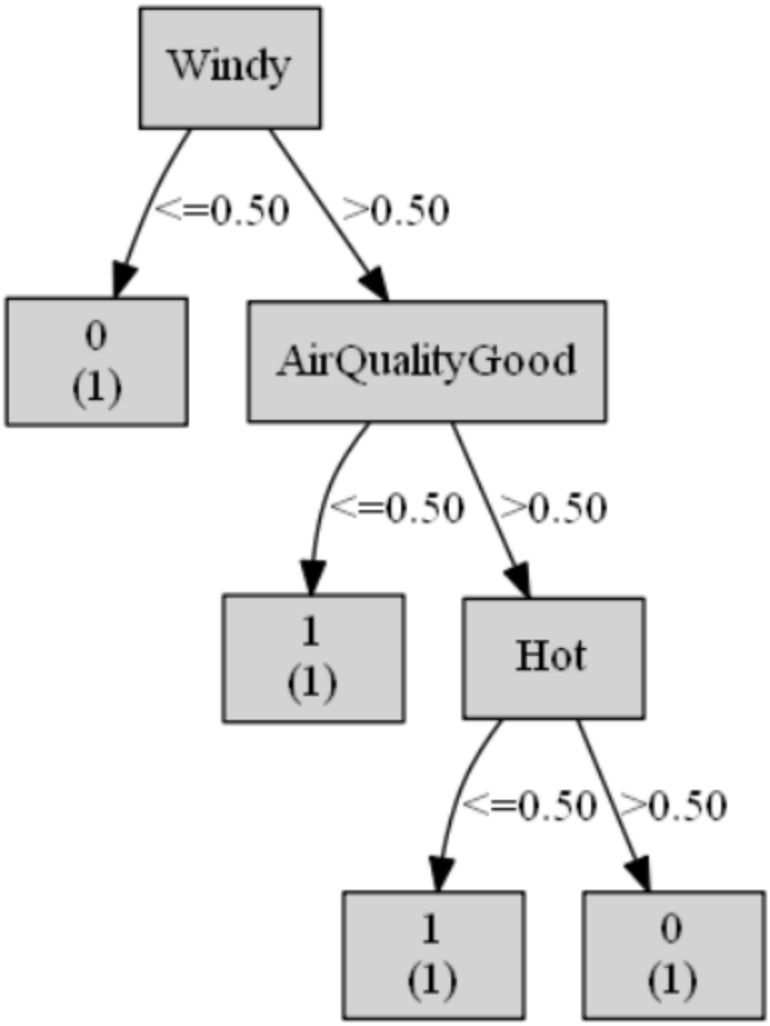
So we can select either Hot or Air as the next level attribute under Wind = Y branch in the decision tree and the other attributes at the last level.

Decision Tree Code File: hw2_DecisionTree_TennisPlay.ipynb [Replace the ‘filepath’ variable with your local file path and filename]

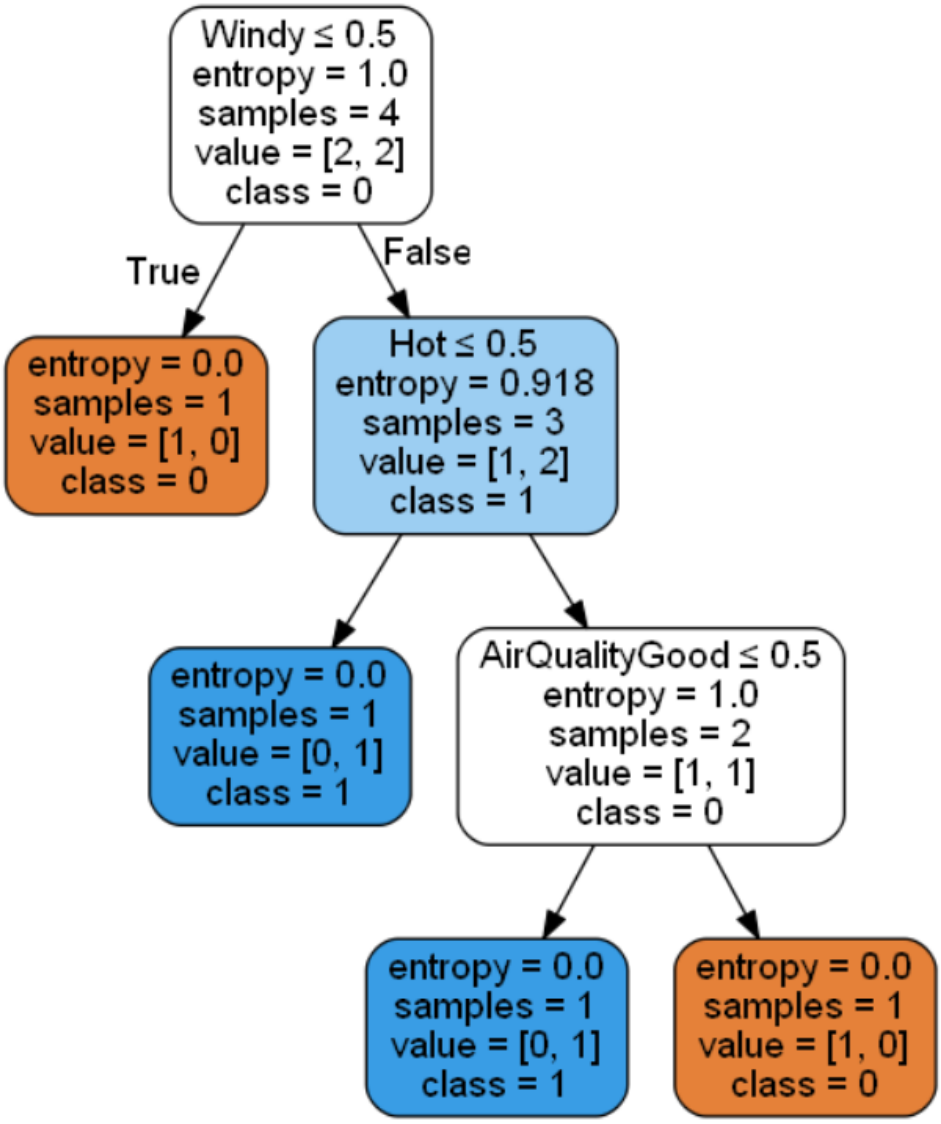
Observation: Manually calculated and code generated decision trees are same. Information gain calculation is also same. In python Sklearn library uses CART algorithm but gives better visualization of tree. I have used ID3Estimator library to build the decision tree using ID3 algorithm. But for better visualization attaching the image of CART algorithm output with entropy. Both have the same output. IG (AirQuality) and IG(Hot) information gain is same. So anyone can go in level 2 or 3.

Output:

1. Using ID3Estimator - ID3



2. Using Sklearn -CART



5. (a) [10 Points]

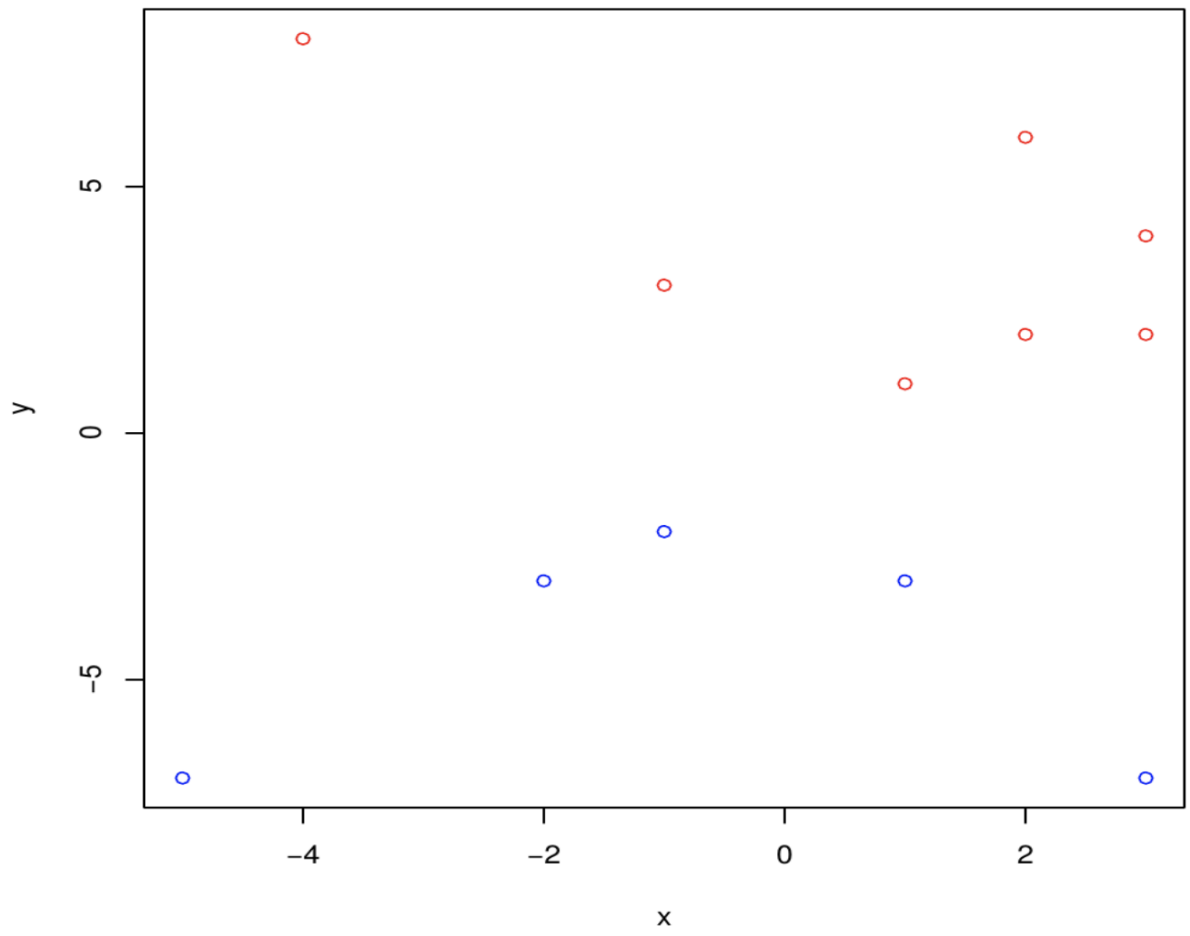
Assume we have only two features in our training dataset that is already classified into class C1 and class C2. The transposes of the feature vectors are given below for each class:

C1: [2 6], [1 1], [3 4], [3 2], [-4 8], [-1 3], [2 2]

C2: [-5 -7], [-2 -3], [-1 -2], [3 -7], [1 -3]

Giving details and derivations, determine the (i) number of support vectors from each class (ii) the support vectors (iii) the equation of the hyperplanes (straight lines in this case) defining the sides of the margin (iv) length of the maximum margin (v) equation of the maximum margin linear classifier.

Answer: First, we plot the features shown in the figure below:



From the plot, we can intuitively understand that there are 3 support vectors: 1 vector from class C1: (1, 1) and 2 vectors from class C2: (1, -3) and (-1, -2).

Now, let us try to find the hyperplane such that **w**.**x** - b >= 1 for any vector in class C1 and **w**.**x** - b <= -1 for any vector in class C2.

Now, for support vectors, |**w**.**x** - b| = 1.

So, for support vector (1, 1), $w_1 + w_2 - b = 1$ … (1)

For support vector (1, -3), $w_1 - 3w_2 - b = -1$ … (2)

For support vector (-1, -2), $- w_1 - 2w_2 - b = -1$ … (3)

By subtracting (2) from (1), we get $4w_2 = 2$, => $w_2 = ½$.

By adding (2) and (3), we get $-5w_2 - 2b = -2$, => $b = - ¼$.

So, $w_1 + w_2 - b = 1$, => $w_1 = ¼$.

Therefore, the equations for defining the sides of the hyperplane are:

$$w_1x_1 + w_2x_2 - b = 1, => ¼ x_1 + ½ x_2 + ¼ = 1 => ¼ x_1 + ½ x_2 - ¾ = 0$$

And

$$w_1x_1 + w_2x_2 - b = -1, => ¼ x_1 + ½ x_2 + ¼ = -1 => ¼ x_1 + ½ x_2 + 5/4 = 0$$

The maximum margin is 2/|**w**|, which is 2/sqrt((1/4)^2 + (1/2)^2) = 8 / sqrt(5) = 3.57777

The equation of maximum margin classifier is $w_1x_1 + w_2x_2 - b = 0$, i.e. $¼x_1 + ½x_2 + ¼ = 0$.

(b) [10 Points]

Follow the tutorials on SVM at

https://chrisalbon.com/machine_learning/support_vector_machines/find_support_vectors/ and
https://towardsdatascience.com/breast-cancer-classification-using-support-vector-machine-svm-a510907d4878

After thoroughly reading the tutorials, without referring to them, write your own code to list the support vectors, their indices, and the number of support vectors in each class of the breast cancer dataset.

Answer:

File: hw2_SVM.ipynb [Replace the 'filepath' variable with your local file path and filename]

Output:

Accuracy with down sampled data: 95.1219512195122 %

Classification: precision recall f1-score support

0	0.97	0.95	0.96	130
1	0.92	0.95	0.93	75
accuracy			0.95	205
macro avg	0.95	0.95	0.95	205
weighted avg	0.95	0.95	0.95	205

Support Vectors:

[[4. 3. 2.]

 [6. 1. 3.]

 [5. 3. 2.]

 [5. 4. 5.]

 [4. 3. 3.]

 [4. 2. 4.]

 [5. 2. 2.]

 [8. 4. 4.]

 [5. 1. 4.]

 [6. 2. 1.]

 [4. 4. 4.]

 [2. 3. 2.]

 [3. 1. 4.]

 [3. 1. 2.]

 [5. 1. 3.]

 [4. 1. 2.]

 [1. 3. 1.]

[8. 4. 6.]

[5. 7. 7.]

[1. 1. 4.]

[5. 4. 4.]

[5. 3. 4.]

[6. 1. 1.]

[6. 2. 3.]

[1. 1. 3.]

[6. 9. 7.]

[3. 4. 5.]

[1. 1. 1.]

[4. 1. 1.]

[3. 1. 1.]

[7. 6. 3.]

[8. 4. 10.]

[2. 7. 10.]

[7. 4. 6.]

[10. 8. 4.]

[10. 5. 8.]

[4. 5. 5.]

[6. 1. 3.]

[9. 6. 9.]

[5. 8. 8.]

[6. 6. 7.]

[3. 3. 6.]

[5. 4. 4.]

[6. 3. 2.]

[10. 3. 6.]

[7. 3. 2.]

[8. 3. 8.]

[10. 7. 7.]

[6. 5. 4.]

[4. 8. 6.]

[2. 5. 7.]

[10. 2. 2.]

[7. 4. 5.]

[5. 3. 3.]

[10. 8. 8.]

[8. 5. 6.]

[4. 8. 7.]

[4. 2. 3.]

[8. 3. 5.]

[6. 10. 5.]

[5. 3. 5.]

[8. 2. 3.]

[4. 7. 8.]

[10. 8. 7.]

[4. 10. 4.]

[8. 10. 5.]

[10. 8. 10.]

[7. 2. 4.]

[2. 3. 4.]

[8. 4. 4.]

[5. 3. 4.]

[6. 5. 5.]

[8. 10. 10.]

[6. 8. 7.]

[9. 4. 5.]

[5. 7. 10.]

[10. 10. 6.]

[5. 10. 6.]

[5. 8. 4.]

[10. 4. 3.]

[8. 8. 7.]

[5. 10. 8.]

[3. 10. 7.]

[8. 8. 9.]

[10. 10. 8.]

[8. 7. 5.]

[6. 10. 10.]

[7. 6. 10.]

[5. 4. 6.]

[8. 4. 7.]

[4. 5. 5.]

[7. 9. 4.]

[10. 10. 10.]

[8. 10. 3.]

[2. 5. 3.]

[1. 5. 8.]

[10. 6. 4.]

[6. 10. 7.]

[7. 5. 10.]

[3. 10. 3.]

[1. 4. 3.]

[10. 4. 2.]

[5. 10. 10.]

[8. 7. 4.]

[5. 3. 3.]

[3. 10. 8.]

[10. 4. 7.]

[5. 6. 6.]

[10. 6. 6.]]

Indices of Support Vectors: [12 16 27 37 55 66 70 73 78 96 98 101 102 104 111 116 121 122 125 127 140 141 143 145 146 157 158 160 162 163 164 167 168 169 172 177 178 183 189 194 195 197 198 199 204 206 208 209 210 211 216 219 220 222 226 227 228 229 230 235 236 239 242 243 244 251 252 255 256 259 262 264 265 268 271 273 276 277 278 279 283 284 285 286 288 289 293 294 300 301 302 304 306 307 308 309 310 311 312 313 314 319 320 321 322 323 325 326 327]

Number of support vectors for each class: [30 79]