In the name of God

University of Tehran
College of Engineering
Faculty of ECE

Intelligent Systems

HW2

Reza Jahani

810198377

Fall 1401

Table of Contents

Question 1) Decision Tree

Part 1) Classifier Design

Feature : Color [5A, 5B] $S = -\frac{5}{10}\log_2\frac{5}{10} + \frac{5}{10}\log_2\frac{5}{10} = 1$

(Brown) [4A, 2B] $S_B = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.9183$

(Green) [1A, 3B] $S_G = -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} = 0.8113$

$\text{Gain (color)} = S - \sum_{v \in \{B,G\}} \frac{n_v}{n} S_v = 1 - \frac{6}{10} \times 0.9183 - \frac{4}{10} \times 0.8113 = 0.1245$

Feature : n feet

(2) [5A, 2B] $S_{(2)} = 0.8631$

(3) [0A, 3B] $S_{(3)} = 0$

$\text{Gain (n feet)} = 1 - \frac{7}{10} \times 0.8631 - \frac{3}{10} \times 0 = 0.3958$

Feature : Height

(tall) [3A, 3B] $S_{tall} = 1$

(short) [2A, 2B] $S_{short} = 1$

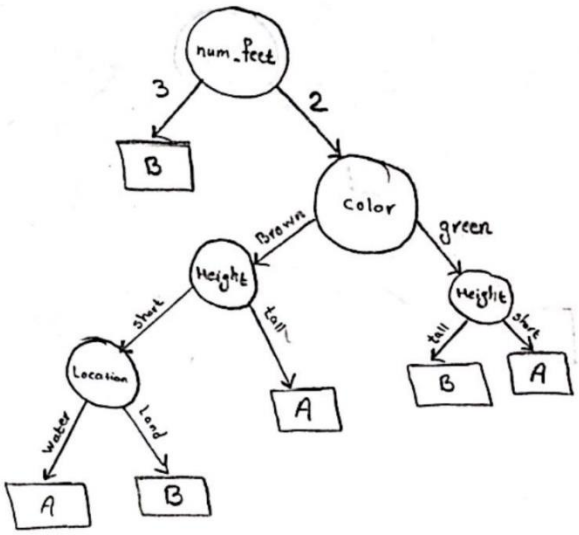
$\text{Gain (Height)} = 1 - \frac{6}{10} \times 1 - \frac{4}{10} \times 1 = 0$

Feature : Location

(Land) [2A, 2B] $S_L = 1$

(water) [3A, 3B] $S_w = 1$

Gain (Location) = 0



→ First node should be "num_feet"

3 Features remaining : [Color, Height, Location]

[5A, 2B] $S = 0.8631$

Gain (Color) = $0.8631 - \frac{5}{7} \times 0.7219 - \frac{2}{7} \times 1 = 0.1986$

Gain (Height) = $0.8631 - \frac{3}{7} \times 0.9183 - \frac{4}{7} \times 0.8113 = 0.059$

Gain (Location) = $0.8631 - \frac{3}{7} \times 0.9183 - \frac{4}{7} \times 0.8113 = 0.059$

→ Next node should be "Color"

2 Features remaining : [Height, Location]

[4A, 1B] $S = 0.7219$

Gain (Height) = $0.7219 - \frac{3}{5} \times 0 - \frac{2}{5} \times 1 = 0.3219$

Gain (Location) = $0.7219 - \frac{3}{5} \times 0.9183 - \frac{2}{5} \times 0 = 0.1709$

→ next node should be "Height".

Part 2) Classifier Evaluation

Using the classifier designed above which is observed in figure 1.2.1, each sample's label will be predicted.

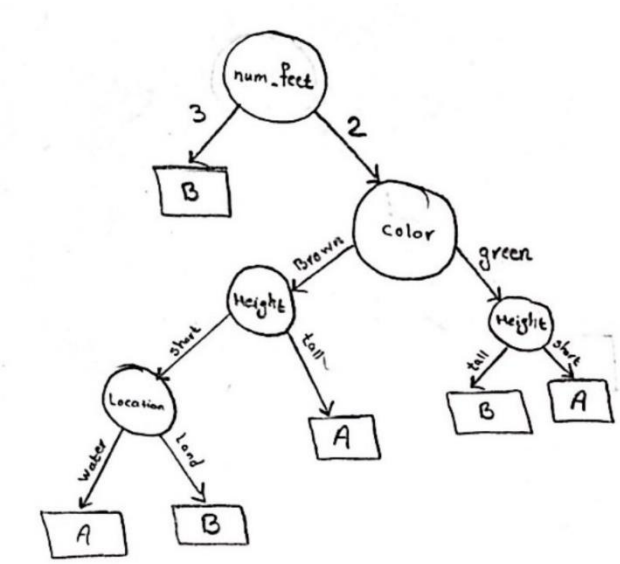


Figure 1.2.1

Sample	Predicted	Label
1	B	B
2	B	A
3	A	A
4	A	B
5	A	A

Table 1.2.1: Test Data

Accuracy = $\frac{3}{5} = 60\ \%$

True / Predicted	"A"	"B"
A	2	1
B	1	1

Table 1.2.2: Confusion Matrix

Part c) ID3 Greedy Approach

Since ID3 is a top-down greedy algorithm, employing only 2 features is not possible for designing the classifier. Because if it was, we would already reach it in the first design.

Part d) Classifier's Fitness Increment

To represent the reasons of overfitting in decision trees we can state 3 reasons.

1. Presence of noise
2. Lack of representative instances
3. Overfitting due to multiple comparison procedure

Most common approaches for avoiding overfitting in decision trees are listed below:

- Pre-Pruning
    - In this approach a smaller tree with fewer branches are generated.
- Post-Pruning
    - In this approach a tree is generated and then parts of it is removed.

Question 2) Decision Tree Algorithm Implementation

Part 1) Decision Tree Model Implementation

In this section we aim to implement the decision tree based on the theories which were derived in lecture notes and previous question. The model is implemented using object orientation in python where 2 classes of Node and Decision Tree are defined with their following methods. After implementation of the model, it is trained with different hyper parameters such as max depth of the tree and in each step, the accuracy and confusion matrix is displayed. As stated in the notebook itself, with increment of max depth, accuracy gets higher and model works with better performance.

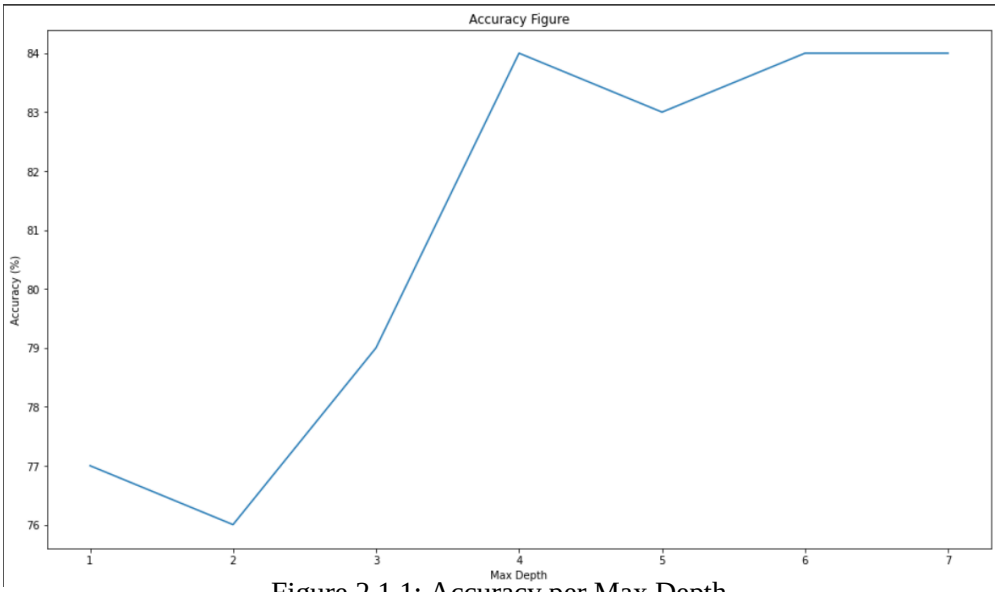


Figure 2.1.1: Accuracy per Max Depth

Below the reports from some steps of the training and evaluation is brought.

1. Max Depth = 3

Accuracy = 79%

True / Predicted	“0”	“1”
0	61	10
1	18	45

Confusion Matrix

2. Max Depth = 5

Accuracy = 83%

True / Predicted	“0”	“1”
0	64	7
1	16	47

Confusion Matrix

3. Max Depth = 7

Accuracy = 84%

True / Predicted	“0”	“1”
0	67	4
1	18	45

Confusion Matrix

Part 2) Developing The Decision Tree Algorithm

When it comes to training of the decision tree, based on the derived model, some features are chosen to be employed for prediction of label based on max depth and this fact limits the performance of the tree in prediction of label because some of the feature may be held unused when they can seem useful in labels prediction.

Employing Bagging or Random Forest which in fact is using a number of decision trees which are trained with different configurations and then taking a majority vote among all the results can lead to better accuracy.

Let us have 3 decision trees that each of them use different features to predict the label and the final result of the prediction from each tree will be [1,1,0]. In this case the final prediction will be “1” which has occurred twice among the trees. Obviously this action increases the accuracy since more features and more trees are involved in label prediction.

Part 3) Employing Random Forest

In this section we aim to implement the algorithm stated in previous section.
As already explained and as it is clear based on the driven code the Random Forrest algorithm is implemented and below results were obtained in each situation of number of trees.

- Random Forrest with 3 Trees

Accuracy = 82.09%

True / Predicted	"0"	"1"
0	73	6
1	18	37

Confusion Matrix

- Random Forest with 6 Trees

Accuracy = 81,34%

True / Predicted	"0"	"1"
0	74	5
1	20	35

Confusion Matrix

- Random Forest with 15 Trees

Accuracy = 82.84%

True / Predicted	"0"	"1"
0	73	6
1	17	38

Confusion Matrix

- This section is implemented in the Google Colab file named ***HW2_810198377.ipynb*** with title of **Question 2.**

Question 3) Learning per Criteria

KNN Classifier

Part 1) Classifier Design

In this section the KNN model is designed with Euclidian Distance parameter. The model is based on 2 functions and it is clear in the code. As instructed the dataset is separated to 2 parts and the model's performance is evaluated.

```
** KNN Models **
####################
K=10
Accuracy = 74.0%
**Confusion Matrix**
[[12  0  0]
 [ 1  8  5]
 [ 1  2  5]]
------------------------
K=5
Accuracy = 71.0%
**Confusion Matrix**
[[12  0  0]
 [ 1  8  5]
 [ 2  2  4]]
------------------------
K=1
Accuracy = 88.0%
**Confusion Matrix**
[[12  0  0]
 [ 1 10  3]
 [ 0  0  8]]
------------------------
K=20
Accuracy = 74.0%
**Confusion Matrix**
[[12  0  0]
 [ 0  7  7]
 [ 1  1  6]]
------------------------
```

Figure 3.1.1

- K = 10

Accuracy = 74%

True / Predicted	“0”	“1”	“2”
0	12	0	0
1	1	8	5
2	1	2	5

Confusion Matrix

- K = 5

Accuracy = 71%

True / Predicted	“0”	“1”	“2”
0	12	0	0
1	1	8	5
2	2	2	4

Confusion Matrix

- K = 1

Accuracy = 88%

True / Predicted	“0”	“1”	“2”
0	12	0	0
1	1	10	3
2	0	0	8

Confusion Matrix

- K = 20

Accuracy = 74%

True / Predicted	“0”	“1”	“2”
0	12	0	0
1	0	7	7
2	1	1	6

Confusion Matrix

Part 2) Possession to each Category of Class Probability Calculation

As instructed in the manual, in this section we aim to configure the probability of prediction of labels in each training setting. For this purpose, we employed a simple algorithm to check the K nearest neighbor's labels and calculate the probability that a test data is whether classified correctly or by mistake. These count plots can be observed in figure 3.2.1 to figure 3.2.4

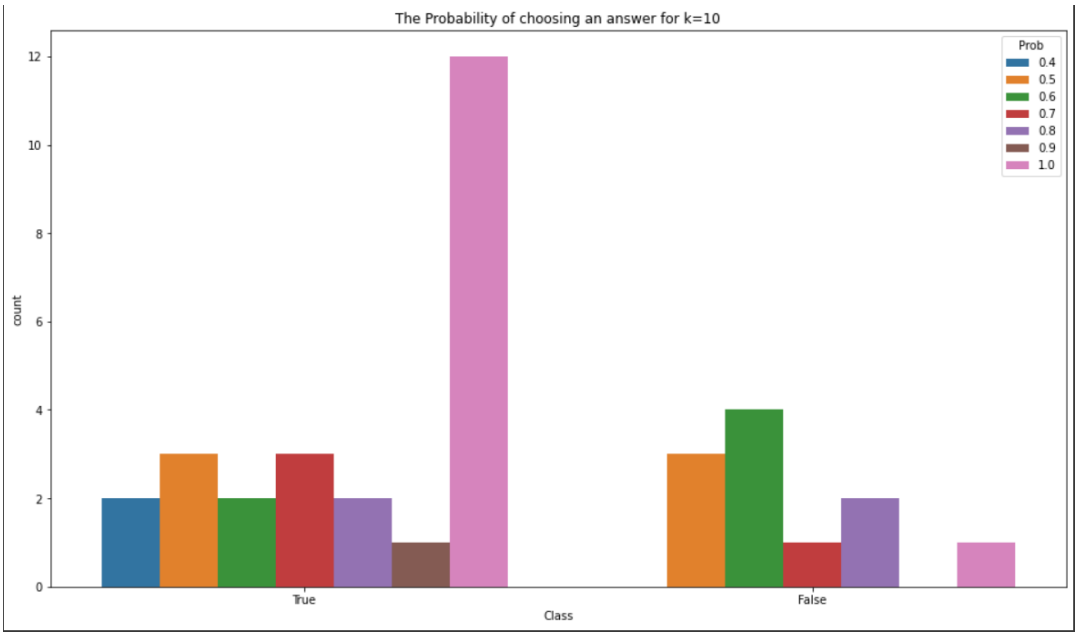


Figure 3.2.1: Count plot for K=10

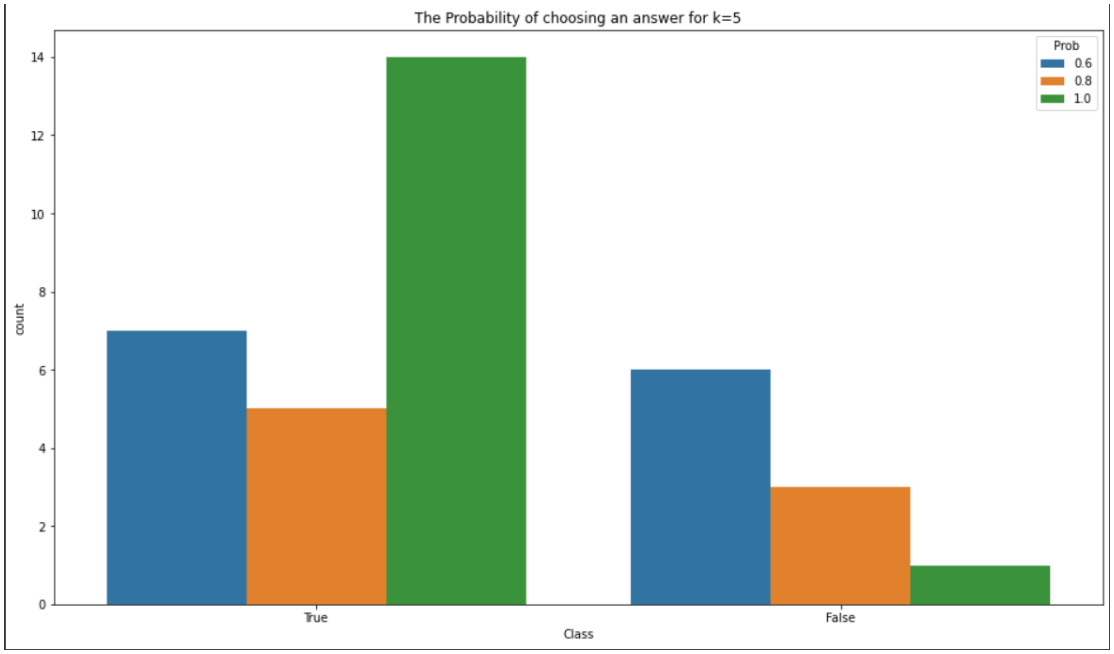


Figure 3.2.2: Count Plot for K=5

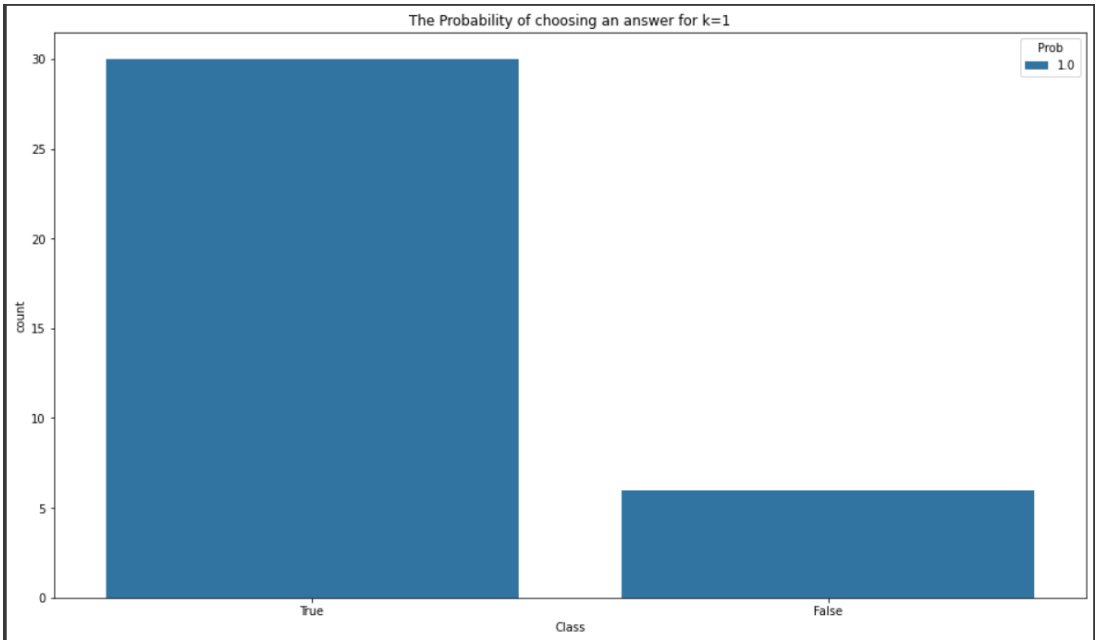


Figure 3.2.3: Count Plot for K=1

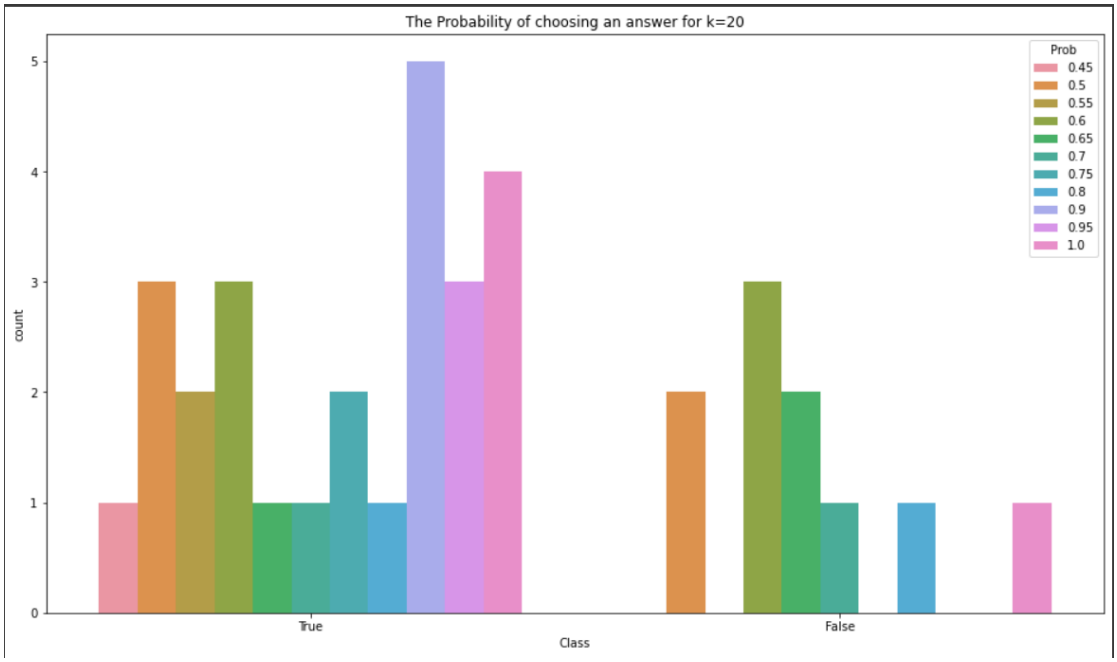


Figure 3.2.4: Count Plot for K=20

As it can be observed, when K decreases, since less neighbors are involved in deciding the label of test data, unique values of probability decreases. For instance, for K=1 only probability value is 1.
By taking a glance at the plots, it can easily be deduced that for K=1 the model performs better than other situations since only 5 data were mistakenly predicted.

Metric Based Learning

Part 1) Learning Method Evaluation

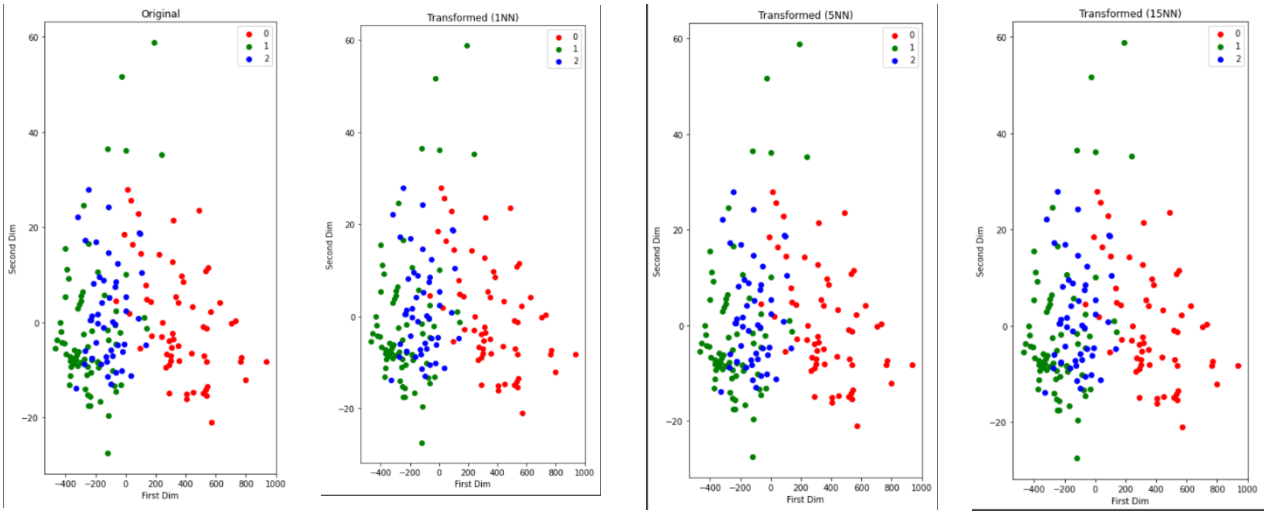
LMNN which stands for Large Margin Nearest Neighbors, aims for keeping the K nearest neighbors close to the same class while keeping other classes away with a large margin. This learning method employs a specific type of distance metric. Some of it's argument to mention are "k" value for deciding how many neighbors to involve in prediction, "init" for initialization of the linear transformation or "pca" for initialization of the transformations and ….

On the other hand, LFDA which stands for Local Fisher Decrement Analysis (Employed for supervised dimensionality reduction) is a linear supervised dimensionality reduction method. It is mostly used when struggling with multimodality. Some of its parameters or constraints to mention can be "n_components" which is the final reduced space dimension, "k" value for number of neighbors and ….

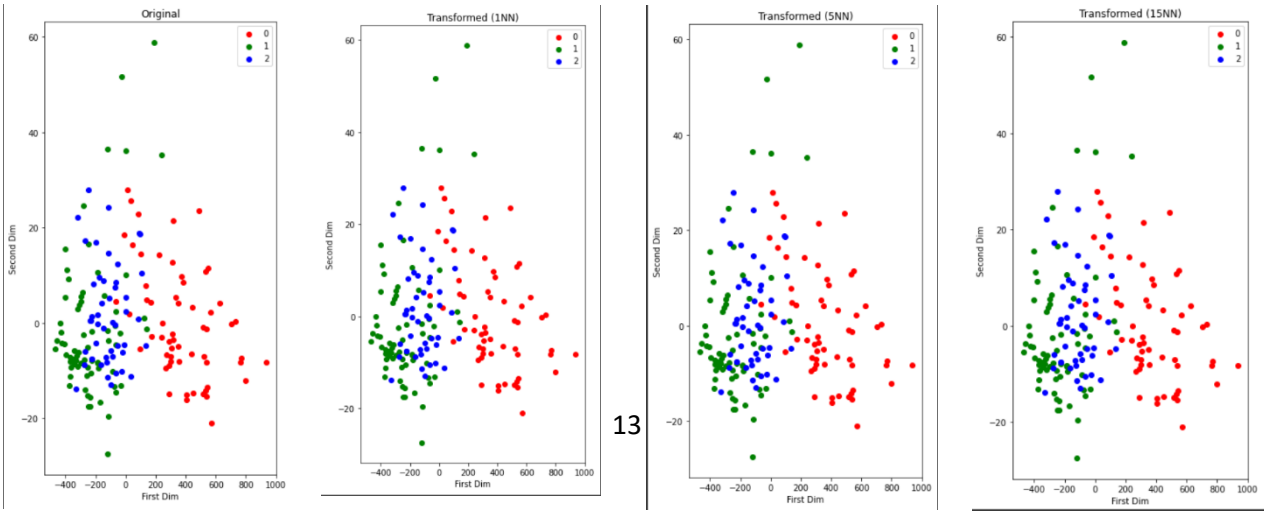
Part 2) Mapped Data Plot

1. K parameter in LMNN as already explained, is used for considering K neighbors to predict the label of the data based on its own specific process of LMNN while in LFDA, this parameter is employed as one of the effective parameters of local scaling method which leads to dimensionality reduction.
2. After applying PCA on the dataset to reduce the dimension, LFDA and LMNN methods are employed and the result is plotted.

- LFDA



- LMNN



Part 3) Classifier's Performance Comparison

As it can be observed when K=15 the best mapping with the best margin is occurring so the model would have a better performance. In this section we aim to apply the KNN model on the last mapped dataset.

```
LMMN K=15
Accuracy=97.0%
** Confusion Matrix **
[[ 9  1  0]
 [ 0 17  0]
 [ 0  0  9]]
```

As observed above, the final result of this action can be seen with the accuracy parameter and the confusion matrix.

Part 4) Correlation

As instructed and asked in the project manual, the correlation matrix between the features has been obtained and plotted and can be observed in figure 3.4.1

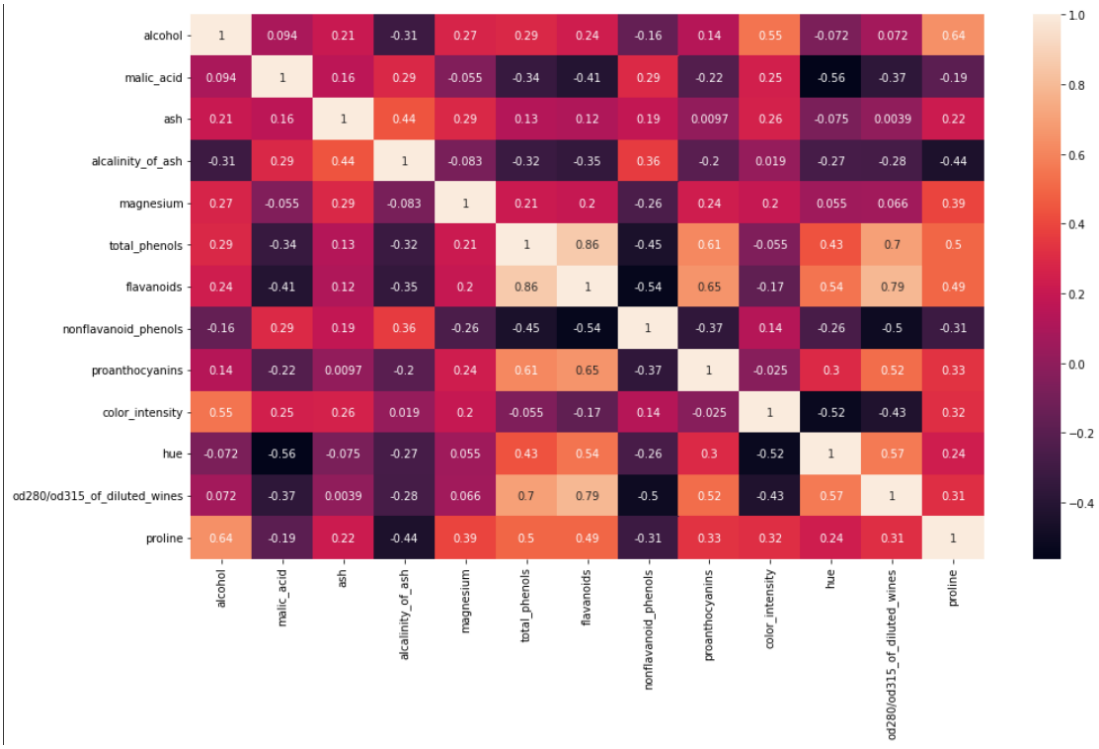


Figure 3.4.1: Correlation Matrix

Now we will obtain the correlation matrix between two new features that was gained after dimensionality reduction and applying other methods for mapping the data points.

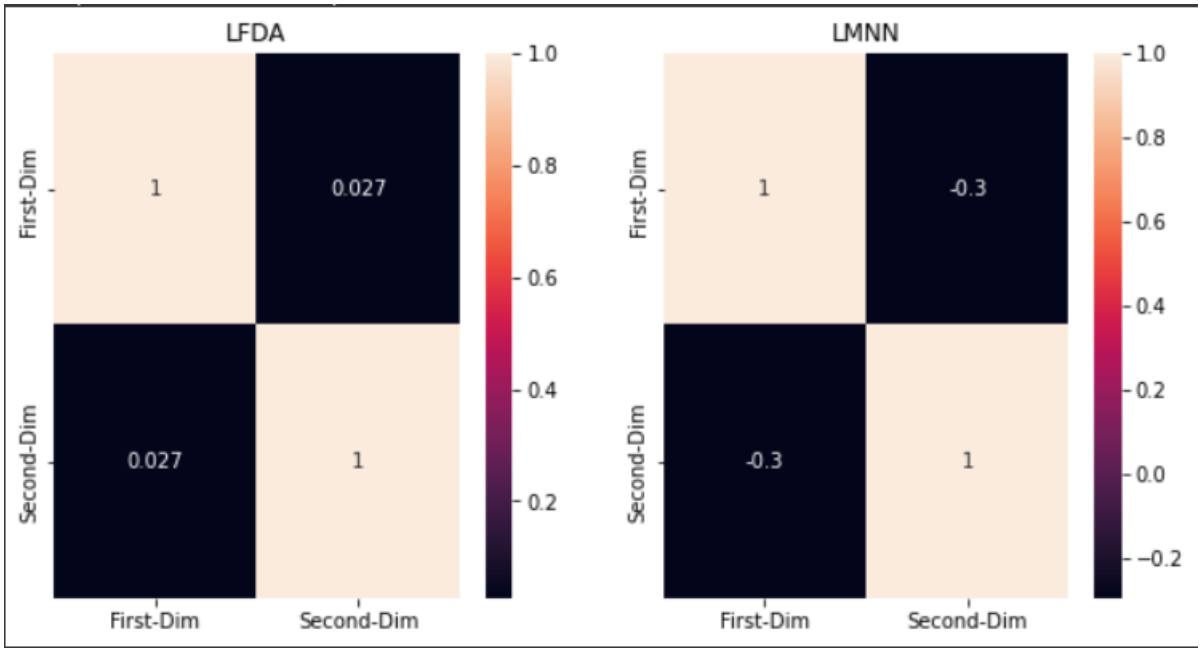


Figure 3.4.2: Correlation Matrix for Mapped Points

- The difference between LFDA result and LMNN result is obvious that in latter method, the features are still dependent which makes the model work better.
- In addition, the feature columns in LMNN method provide an overall summary of depending features among those 13 features in the beginning.

Part 5) GMML

This method employs Euclidian Distance metric for categorizing the dataset into different groups. By providing a geometric insight about the problem, and reformulating the distance metric from the beginning and basic principles, a closed-form solution for an unconstrained and smooth non-convex problem is presented.

One of the most significant differences with LMNN can be pointed out as the distance metric used. As already explained LMNN employs Mahalanobis distance metric and in the end it succeeds in providing an appropriate margin for those data points which have different classes while grouping and decreasing the distance between points with the same class.

- This section is implemented in the Google Colab file named ***HW2_810198377.ipynb*** with title of **Question 3.**

Additional Section

- Implementations and codes are stored in a google colab file named ***HW2_810198377.ipynb***.