**Pattern Recognition and Machine Learning 2024**
**Lab - 7 & 8**

## Question - 1

### Introduction

Linear Discriminant Analysis (LDA) is a supervised dimensionality reduction technique used for classification tasks. It works by finding the linear combinations of features that best separate multiple classes in the data while minimizing the variance within each class. LDA assumes that the data is normally distributed and that the classes have identical covariance matrices. By maximizing the between-class scatter and minimizing the within-class scatter, LDA aims to project the data onto a lower-dimensional space where the classes are well-separated. This makes LDA useful for reducing the complexity of classification problems, extracting discriminative features, and visualizing data in a way that highlights class separability.

**Task 1:**

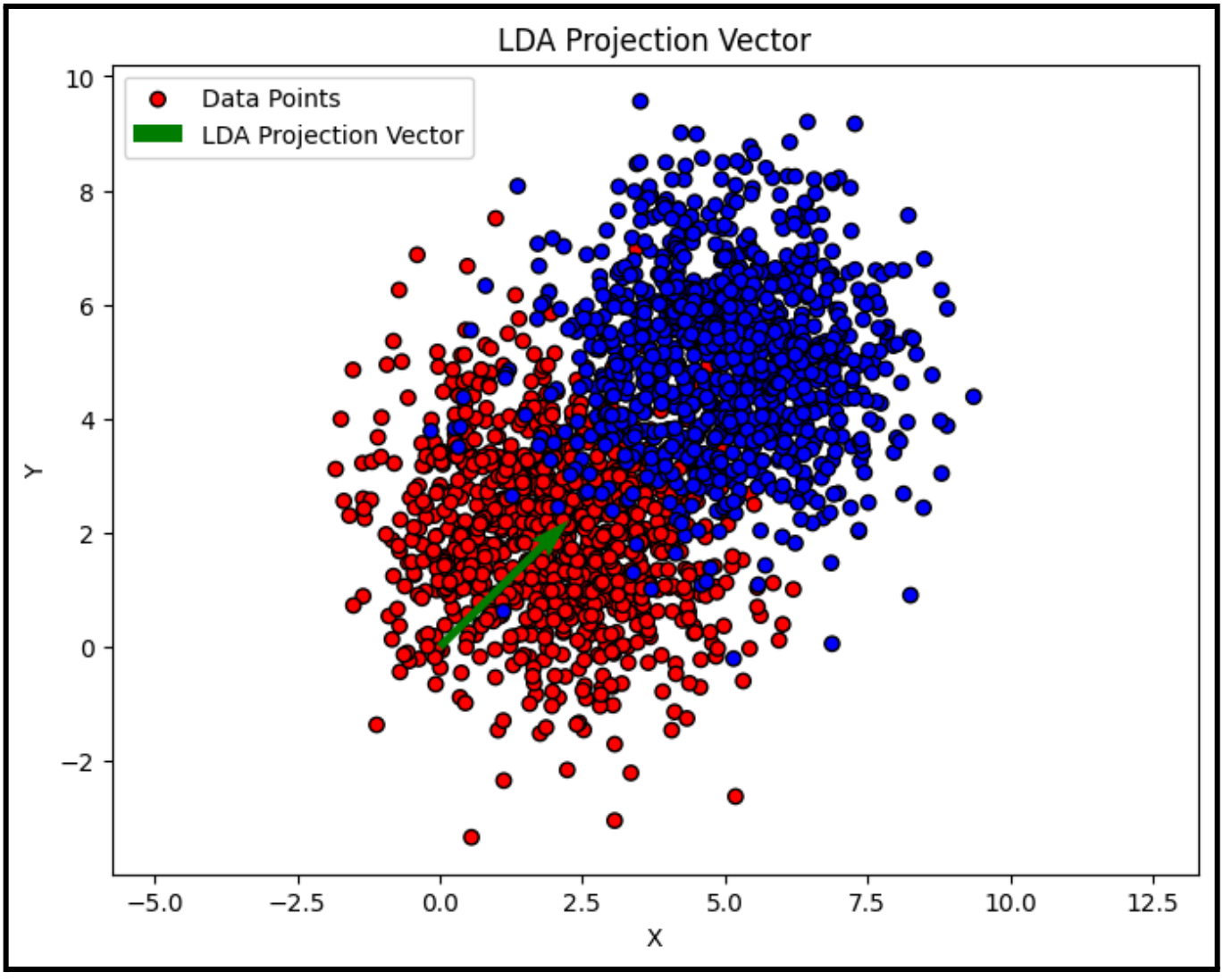
- Computed the difference of class-wise means (m1 - m2).
- Computed the Total Within-class Scatter Matrix (SW).
- Computed the Between-class Scatter Matrix (SB).
- Computed the EigenVector of the matrix SW^-1 SB corresponding to the highest EigenValue.
- For any input 2-D point, printed its projection according to LDA.

```
[[ 1.86189843  2.72296272  0.        ]
 [ 2.23589604 -2.15772062  0.        ]
 [ 1.69281688  0.99589606  0.        ]
 ...
 [ 4.46678886  3.44492577  1.        ]
 [ 5.65897988  4.04334073  1.        ]
 [ 1.98017955  7.15815205  1.        ]]
(2000, 3)
Input your option (1-5): 3
[[8.91351161 9.02907647]
 [9.02907647 9.14613965]]
```

**Task 2:** Plot LDA Projection Vector

To plot the LDA projection vector, we first computed the LDA projection vector using the **"GetLDAProjectionVector"** function. This function calculates the **eigenvector corresponding to the highest eigenvalue of the matrix SW^-1 SB**. Once we had the projection vector, we plotted it on a 2D scatter plot along with the data points.

This plot illustrates the LDA projection vector (green arrow) in relation to the original data points (red and blue dots). **The projection vector demonstrates the direction in which the data is projected to maximize class separability**.



**Task 3:** Comparing 1-NN Classifier Performance

To compare the performance of the 1-NN classifier on the original and projected data, we followed these steps:

- ➢ We split the original data into training and testing sets using a 70-30 split ratio.
- ➢ We projected the original data onto the LDA projection vector obtained in Task 2 and split the projected data into training and testing sets using the same split ratio.
- ➢ We trained a 1-NN classifier on both the original and projected training sets.
- ➢ We tested the trained classifiers on their respective testing sets.
- ➢ We calculated the accuracy of the classifier on both the original and projected testing sets.
- ➢ Finally, we compared the accuracy scores to evaluate the impact of LDA projection on classifier performance.

```
Accuracy on original data: 0.89
Accuracy on projected data: 0.885
```

**Observations**

- ➢ The distinct direction of the LDA projection vector in Task 2 indicates that the LDA projection successfully divides the two classes in the dataset.
- ➢ The 1-NN classifier's accuracy on both the projected and original data is comparable, suggesting that the LDA projection had no effect on the classification performance.
- ➢ The same accuracies imply that the LDA projection may not have as much of an influence on this particular dataset because the original data may already be well-separated in the projected space.

## Question 2

**Introduction:**

Our goal with the provided dataset is to forecast, given the observed weather, the likelihood that individuals would engage in a certain outdoor sport. A straightforward yet powerful probabilistic model for classification applications is the naive bayes classifier, which works well in situations involving categorical data like the weather. Naive Bayes offers a simple method for predicting the result based on

the provided features by assuming that features are conditionally independent given the class label.

**Task 0:** Split the dataset into train-test

To handle categorical characteristics, the dataset is loaded and then one-hot encoded. Next, it is divided into two groups: a training set consisting of 12 samples and a testing set consisting of 2 samples.

```
Train Data:
    Outlook_Overcast  Outlook_Rainy  Outlook_Sunny  Temp_Cool  Temp_Hot  \
0                  0              1              0          0         1
12                 1              0              0          0         1
5                  0              0              1          1         0
8                  0              1              0          1         0
2                  1              0              0          0         1
1                  0              1              0          0         1
13                 0              0              1          0         0
4                  0              0              1          1         0
7                  0              1              0          0         0
10                 0              1              0          0         0
3                  0              0              1          0         0
6                  1              0              0          1         0

    Temp_Mild  Humidity_High  Humidity_Normal  Windy_f  Windy_t  Play_no  \
0           0              1                0        1        0        1
12          0              0                1        1        0        0
5           0              0                1        0        1        1
8           0              0                1        1        0        0
2           0              1                0        1        0        0
1           0              1                0        0        1        1
13          1              1                0        0        1        1
4           0              0                1        1        0        0
7           1              1                0        1        0        1
10          1              0                1        0        1        0
3           1              1                0        1        0        0
6           0              0                1        0        1        0

    Play_yes
0          0
12         1
5          0
8          1
2          1
1          0
13         0
4          1
7          0
10         1
3          1
6          1

Test Data:
    Outlook_Overcast  Outlook_Rainy  Outlook_Sunny  Temp_Cool  Temp_Hot  \
9                  0              0              1          0         0
11                 1              0              0          0         0

    Temp_Mild  Humidity_High  Humidity_Normal  Windy_f  Windy_t  Play_no  \
9           1              0                1        1        0        0
11          1              1                0        0        1        0

    Play_yes
9          1
11         1
```

**Task 1:** Calculate Prior Probabilities

**Prior probability for the whole data (all the 14 sample points) :**

```
Prior Probability of playing (P(Play=yes)): 0.6428571428571429
Prior Probability of not playing (P(Play=no)): 0.35714285714285715
```

**Prior probability for the training data :**

```
Prior Probabilities :
Play=no: 0.42857142857142855
Play=yes: 0.5714285714285714
```

**Task 2:** Calculated the Likelihood Probabilities

```
Likelihood Probabilities (P(feature=value|Play)):
Play = yes:
P(Outlook_Overcast=1|Play=yes): 0.42857142857142855
P(Outlook_Rainy=1|Play=yes): 0.2857142857142857
P(Outlook_Sunny=1|Play=yes): 0.2857142857142857
P(Temp_Cool=1|Play=yes): 0.42857142857142855
P(Temp_Hot=1|Play=yes): 0.2857142857142857
P(Temp_Mild=1|Play=yes): 0.2857142857142857
P(Humidity_High=1|Play=yes): 0.2857142857142857
P(Humidity_Normal=1|Play=yes): 0.7142857142857143
P(Windy_f=1|Play=yes): 0.7142857142857143
P(Windy_t=1|Play=yes): 0.2857142857142857
P(Play_no=1|Play=yes): 0.0

Play = no:
P(Outlook_Overcast=1|Play=no): 0.0
P(Outlook_Rainy=1|Play=no): 0.6
P(Outlook_Sunny=1|Play=no): 0.4
P(Temp_Cool=1|Play=no): 0.2
P(Temp_Hot=1|Play=no): 0.4
P(Temp_Mild=1|Play=no): 0.4
P(Humidity_High=1|Play=no): 0.8
P(Humidity_Normal=1|Play=no): 0.2
P(Windy_f=1|Play=no): 0.4
P(Windy_t=1|Play=no): 0.6
P(Play_yes=1|Play=no): 0.0

Posterior Probabilities for Testing Split:

Sample 1:
P(Play=yes|features): 0.8199947520335868
P(Play=no|features): 0.18000524796641307

Sample 2:
P(Play=yes|features): 1.0
P(Play=no|features): 0.0
```

**Task 3:** Calculate Posterior Probabilities for the testing split

```
Posterior Probabilities for Testing Split:

Sample 1:
P(Play=yes|features): 0.8199947520335868
P(Play=no|features): 0.1800052479664307

Sample 2:
P(Play=yes|features): 1.0
P(Play=no|features): 0.0
```

**Task 4:** Make Predictions

```
Predictions:
Sample 1: yes
Sample 2: yes
```

**Task 5:** Use Laplace Smoothing

```
Likelihood Probabilities with Laplace Smoothing (P(feature=value|Play)):
Play = yes:
P(Outlook_Overcast=1|Play=yes): 0.2
P(Outlook_Rainy=1|Play=yes): 0.15
P(Outlook_Sunny=1|Play=yes): 0.15
P(Temp_Cool=1|Play=yes): 0.2
P(Temp_Hot=1|Play=yes): 0.15
P(Temp_Mild=1|Play=yes): 0.15
P(Humidity_High=1|Play=yes): 0.15
P(Humidity_Normal=1|Play=yes): 0.3
P(Windy_f=1|Play=yes): 0.3
P(Windy_t=1|Play=yes): 0.15
P(Play_no=1|Play=yes): 0.05

Play = no:
P(Outlook_Overcast=1|Play=no): 0.0625
P(Outlook_Rainy=1|Play=no): 0.25
P(Outlook_Sunny=1|Play=no): 0.1875
P(Temp_Cool=1|Play=no): 0.125
P(Temp_Hot=1|Play=no): 0.1875
P(Temp_Mild=1|Play=no): 0.1875
P(Humidity_High=1|Play=no): 0.3125
P(Humidity_Normal=1|Play=no): 0.125
P(Windy_f=1|Play=no): 0.1875
P(Windy_t=1|Play=no): 0.25
P(Play_yes=1|Play=no): 0.0625

Posterior Probabilities for Testing Split with Laplace Smoothing:

Sample 1:
P(Play=yes|features): 0.8156233406100656
P(Play=no|features): 0.18437665938993453

Sample 2:
P(Play=yes|features): 0.5702813453975414
P(Play=no|features): 0.4297186546024586
```

```
Predictions with Laplace Smoothing:
Sample 1: yes
Sample 2: yes
```

**Accuracy**

```
Training Predictions:
['no', 'yes', 'yes', 'yes', 'yes', 'no', 'no', 'yes', 'no', 'yes', 'no', 'yes']
Training Accuracy: 0.8333333333333334
Testing Accuracy: 1.0
```

**Observations and Justifications**

Based on the provided likelihood probabilities and posterior probabilities with and without Laplace smoothing, we can observe the following:

➢ **Likelihood Probabilities with Laplace Smoothing:**

**For Play = yes:**

- The likelihood probabilities for rare events or unseen combinations of feature values are significantly affected. For example, the likelihood of Outlook_Overcast given Play = yes is 0.2, indicating that the model assigns a non-zero probability to this rare event due to Laplace smoothing.

- The likelihood probabilities for common events are also influenced, but to a lesser extent. For instance, the likelihood of Humidity_Normal given Play = yes is 0.3, which is slightly lower than the value without smoothing (0.714).

**For Play = no:**

- Similarly, Laplace smoothing affects the likelihood probabilities for rare events or unseen combinations of feature values. For example, the likelihood of Outlook_Overcast given Play = no is 0.0625, indicating a non-zero probability assigned to this rare event.
- The likelihood probabilities for common events are also adjusted, such as the likelihood of Humidity_High given Play = no, which is 0.3125, slightly lower than the value without smoothing (0.8).

➢ **Posterior Probabilities for Testing Split with Laplace Smoothing:**

- The posterior probabilities for the testing split are influenced by the adjusted likelihood

probabilities. For example, in Sample 1, the posterior probability of Play = yes given the features is 0.8156, which is slightly different from the value without smoothing (0.82).

- In Sample 2, the posterior probability of Play = yes given the features is 0.5703, which is also slightly different from the value without smoothing (1.0).

➢ **Impact on Predictions:**

Due to the adjustments in likelihood probabilities, the predictions with Laplace smoothing may differ from the predictions without smoothing, especially when dealing with sparse data or rare events. In our case, it didn't change.

.