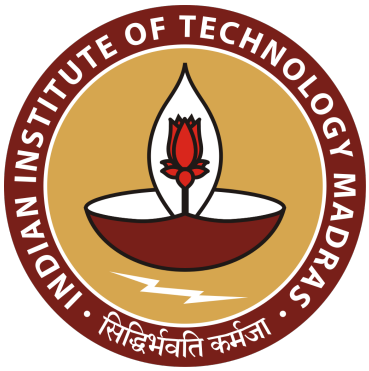
Programming Assignment 2
Classification Methods

CS5691

October 29, 2023

Balakumar R	**Girish Madhavan V**	**Gopalakrishnan T V**
ME20B043	ME20B072	ME20B075



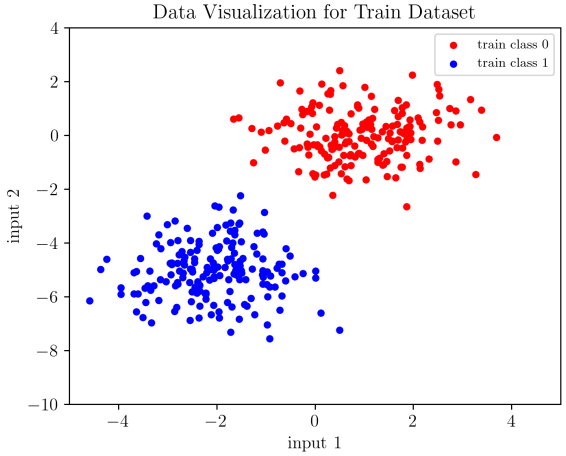
Contents

1 Confusion Matrix Definition:

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Table 1: Confusion Matrix

2 Dataset 1(a)



This dataset consists of **linearly separable** data for 2 classes. i.e, $N \times 2$.

File Name	Data Points
Train-14	360×2
Val-14	90×2
Test-14	50×2

Table 2:

2.1 K - Nearest Neighbours classification

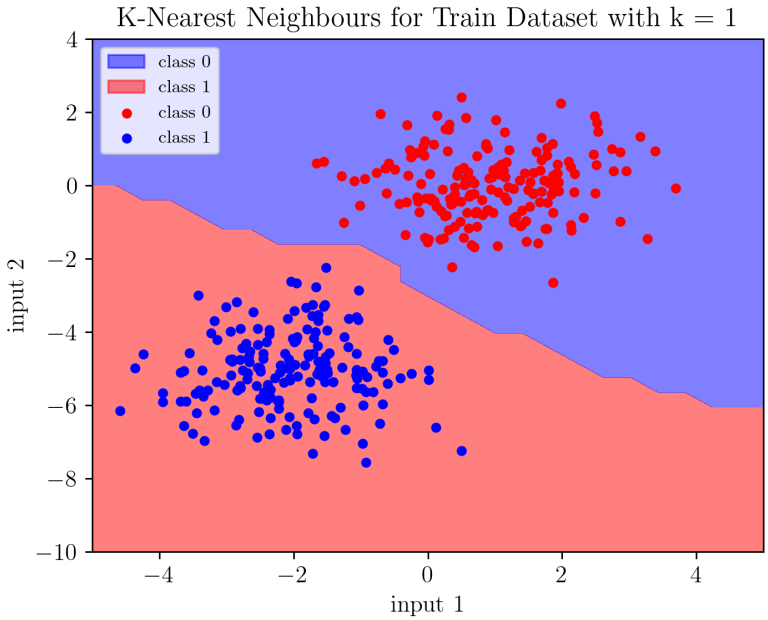
K (hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
1	100%	100%	100%
7	100%	100%	100%
15	100%	100%	100%

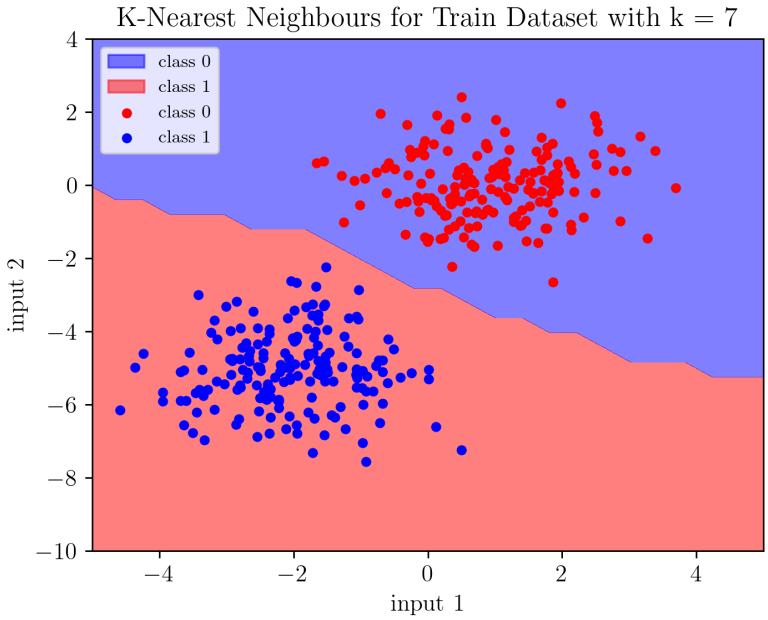
Table 3: KNN classification accuracies for varying hyperparameter k

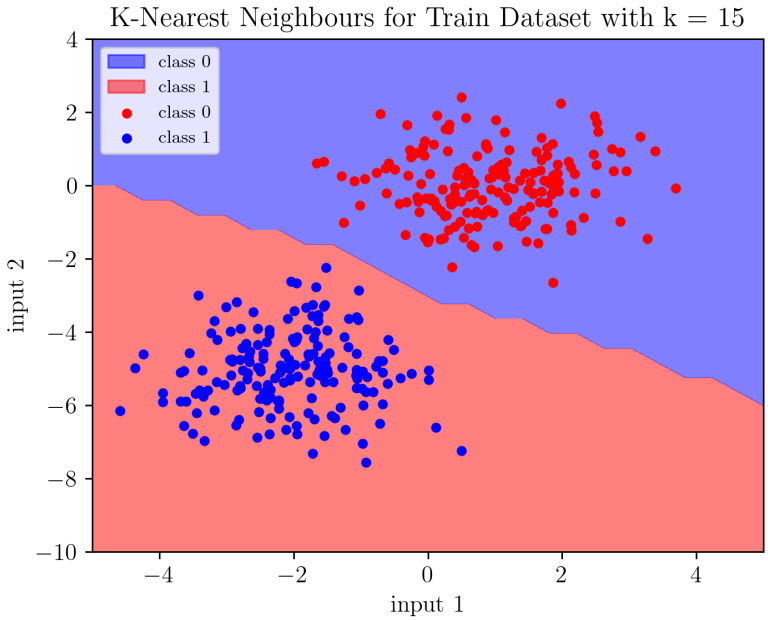
Confusion Matrices for best performing model : Train and Test Accuracy = 100%

$$\text{Training Dataset} \quad \begin{bmatrix} 182 & 0 \\ 0 & 178 \end{bmatrix} \qquad \text{Test Dataset} \quad \begin{bmatrix} 21 & 0 \\ 0 & 29 \end{bmatrix}$$

Plots :







2.1.1 Observations :

From the plots we can see that KNN performs extremely well for linearly separable data. It works with various values of hyperparameter K for KNN, giving a classification accuracy of 100%

2.2 Naive Bayes Classifier

Naive Bayes Classifier considers the features of the data matrix to be independent gaussians, hence the overall class conditional probability density of a datapoint can be represented using a diagonal covariance matrix. That is we say that there is no correlation between the features of the data matrix.

$$P(y_i|\bar{x}) = \frac{p(\bar{x}|y_i)P(y_i)}{p(\bar{x})}$$

For a given datapoint $p(\bar{x})$ remains unchanged

$$P(y_i) = \frac{N_i}{N}$$

Where N_i is number of points in class i & N is total number of data points

$$p(\bar{x}|y_i) = \frac{1}{\sqrt{(2\pi)^d|C_i|}} \cdot e^{\frac{-(\bar{x}-\mu_i)^T C_i^{-1}(\bar{x}-\mu_i)}{2}}$$

Where C_i is a diagonal positive definite matrix

The values for class covariance and mean can be obtained from sample mean and sample covariance corresponding to class i

C_0, C_1	**Training Accuracy**	**Validation Accuracy**	**Test Accuracy**
same	100%	100%	100%
different	100%	100%	100%

Table 4: Naive Bayes classification accuracies

Confusion Matrices for the best performing models : Train and Test Accuracy = 100%

$$\begin{matrix}\text{Training Dataset} \\ \begin{bmatrix} 182 & 0 \\ 0 & 178 \end{bmatrix}\end{matrix} \quad \begin{matrix}\text{Test Dataset} \\ \begin{bmatrix} 21 & 0 \\ 0 & 29 \end{bmatrix}\end{matrix}$$

Decision Boundary Equations :

Same Covariance Matrices

$$\begin{bmatrix} 2.935 & 6.031 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 16.948 = 0$$

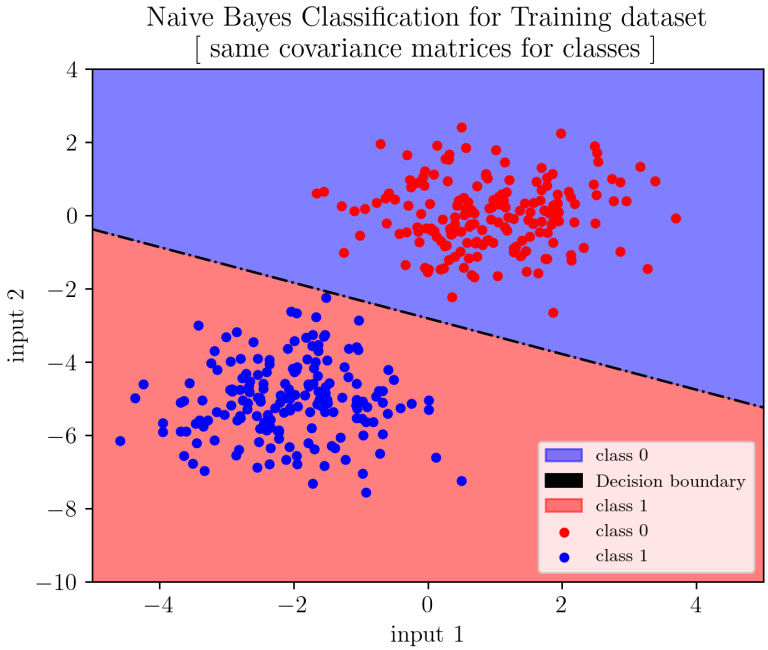
$$2.935x_1 + 6.031x_2 + 16.948 = 0$$

Different Covariance Matrices

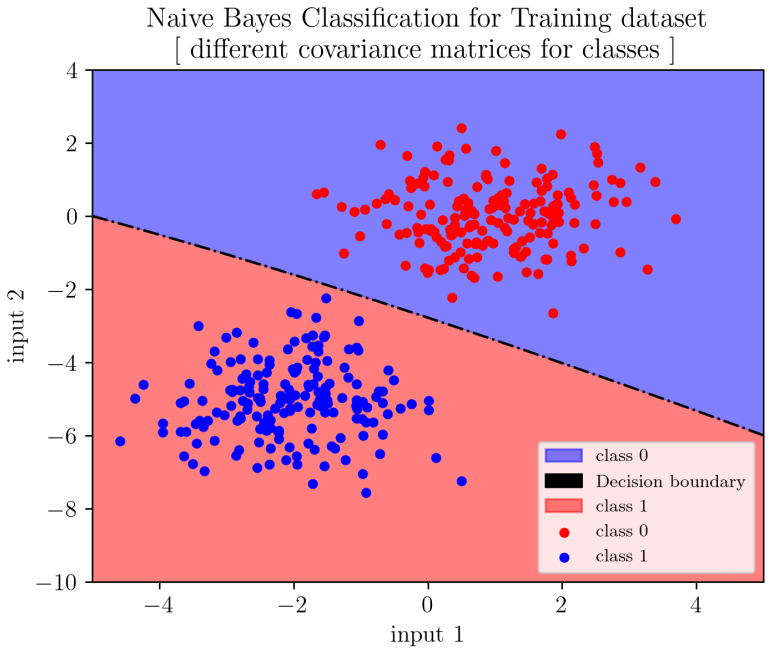
$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 0.092 & 0 \\ 0 & -0.121 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 3.316 & 4.805 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 14.274 = 0$$

$$0.092x_1^2 - 0.121x_2^2 + 3.316x_1 + 4.805x_2 + 14.274 = 0$$

Plots :
Same Covariance Matrices



Different Covariance Matrices



2.2.1 Observations:

1. In this case of linearly separable classes, the Naive Bayes classifier aligns perfectly with the separation between classes as it relies on the assumption of feature independence given the class label.

2. The model's ability to leverage linear decision boundaries aligns well with the distinctive class separation, leading to high classification accuracy. This can be seen in results as the data is separated with 100% accuracy.

2.3 Bayes classifier with hypersphere based Parzen window method for density estimation

The Parzen window is a statistical technique used for estimating probability density in data analysis. It involves placing a window function around each data point and then summing up these functions to estimate the probability density at a specific point. This estimation can be done using different shapes, such as hypercubes or hyperspheres. In our report, we concentrate on hypersphere estimation, which is a mathematical expression used to model this technique.

$$P(x/y_i) = \frac{K_i}{N_i V(h)}$$

$$K_i = \sum_{n=1}^{N} H(\frac{\bar{x} - \bar{x_n}}{h})$$

$$H\Big(\frac{\bar{x} - \bar{x_n}}{h}\Big) = \begin{cases} 1, & \text{if } \|\bar{x} - \bar{x_n}\| \leq h \\ 0, & \text{otherwise} \end{cases}$$

In the Parzen window method, we use a hypersphere with a radius "h" to estimate probability density. The formula for estimating the probability density is expressed as:

- V(h) represents the volume of the hypersphere with radius "h"
- H is the indicator function, taking values 0 or 1 depending on whether a data point falls inside or outside the hypersphere

In essence, we count the number of data points that are within the hypersphere centered around the point of interest. This approach allows us to estimate probability density using a frequentist approach, where we rely on observed data points to make our estimation. The choice of the radius "h" is a hyperparameter that influences the quality of the estimation.

h (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
3	100%	100%	100%
6.65	100%	100%	100%
10	100%	100%	100%
50	94.44%	94.44%	98%
100	51.94%	53.33%	46%

Table 5: Classification accuracies of the model on training data, validation data and training data for different values of hyperparameter

Confusion matrices for best model are as follows:

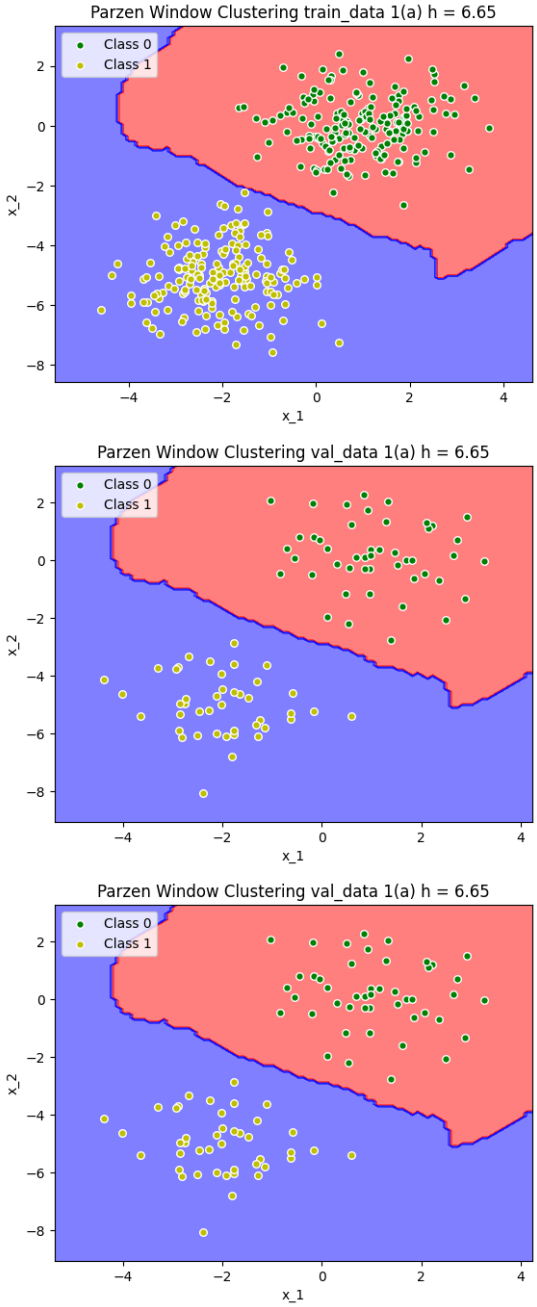
- **h = 6.65**, Train and Test Accuracy = 100%

$$\underset{\text{Training Dataset}}{\begin{bmatrix} 182 & 0 \\ 0 & 178 \end{bmatrix}} \quad \underset{\text{Test Dataset}}{\begin{bmatrix} 21 & 0 \\ 0 & 29 \end{bmatrix}}$$

- **h = 10**, Train and Test Accuracy = 100%

$$\underset{\text{Training Dataset}}{\begin{bmatrix} 182 & 0 \\ 0 & 178 \end{bmatrix}} \quad \underset{\text{Test Dataset}}{\begin{bmatrix} 21 & 0 \\ 0 & 29 \end{bmatrix}}$$

Plots :



2.3.1 Observations:

The Bayes classifier with Parzen Window estimation performs optimally with a radius of 6.65, striking a balance between fitting the training data and avoiding overfitting. Lower radius values tend to overfit, resulting in significantly lower test data performance due to increased sensitivity to training data. The decision surface's clustering around a class can be easily influenced by slight changes in the radius hyperparameter. This observation underscores the importance of fine-tuning hyperparameters for achieving a well-generalizing model.

2.4 Bayes classifier with KNN method for estimation of class-conditional probability density function, for K=10 and K=20 neighbors

$$P(x/y_i) = \frac{K}{N_i V(h)}$$

In the KNN method, we use a hypersphere K to estimate probability density. The formula for estimating the probability density is expressed as:

- V(h) represents the volume of the hypersphere with radius of K^{th} nearest neighbour
- If the volume V(h) is less, the point lies in that group

K (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
10	100%	100%	100%
20	100%	100%	100%

Table 6: Classification accuracies of the model on training data, validation data and training data for different values of hyperparameter

Confusion matrices for best model are as follows:

- **K = 10**

Training Dataset
Accuracy = 100%

$$\begin{bmatrix} 182 & 0 \\ 0 & 178 \end{bmatrix}$$

Test Dataset
Accuracy = 100%

$$\begin{bmatrix} 21 & 0 \\ 0 & 29 \end{bmatrix}$$

- **K = 20**

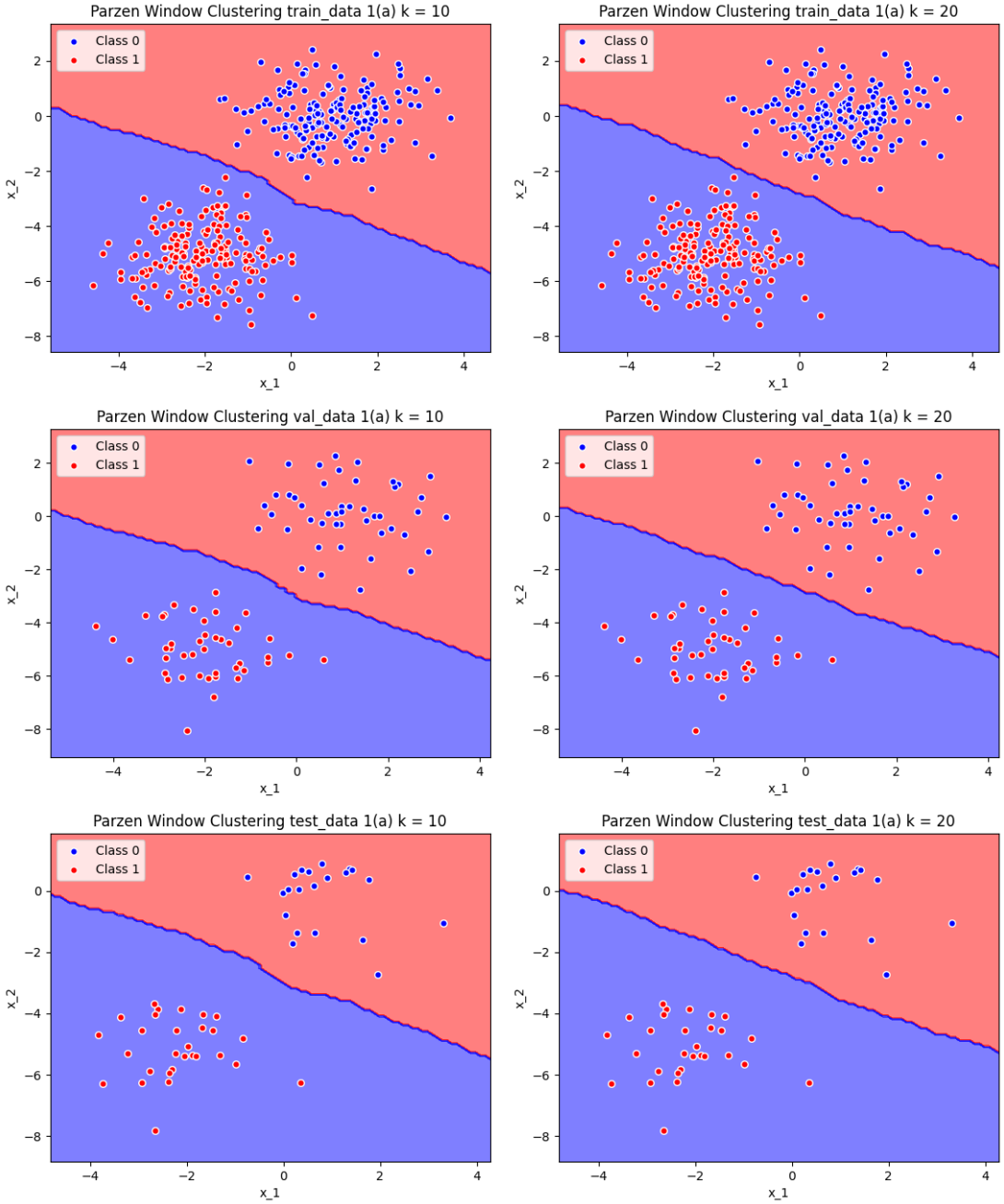
Training Dataset
Accuracy = 100%

$$\begin{bmatrix} 182 & 0 \\ 0 & 178 \end{bmatrix}$$

Test Dataset
Accuracy = 100%

$$\begin{bmatrix} 21 & 0 \\ 0 & 29 \end{bmatrix}$$

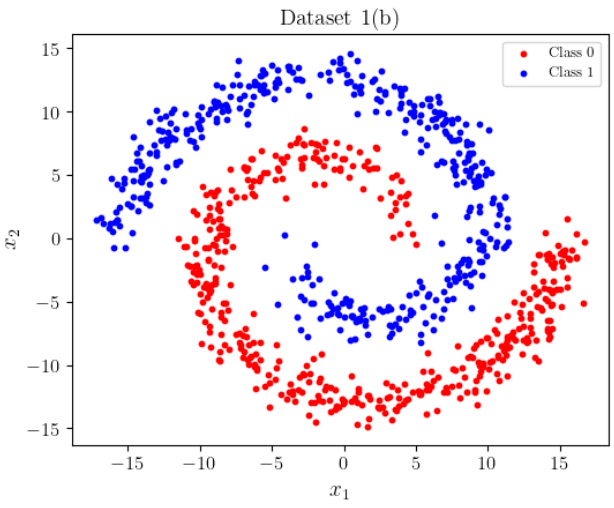
Plots :



2.4.1 Observations:

The performance of the Bayes classifier with K-nearest neighbor likelihood estimation remains consistent across different values of k. This consistency is due to the linear separability of the data, where small changes in the decision boundary do not significantly affect the model's performance.

3 Dataset 1(b)



This dataset consists of **nonlinearly separable** data set for 2 classes. i.e, $N \times 2$.

File Name	Data Points
Train-9	842×2
Val-9	238×2
Test-9	120×2

Table 7:

3.1 K - Nearest Neighbours classification

In this method of classification, we determine the class of a test point using the majority class label of K neighboring points around our test point.

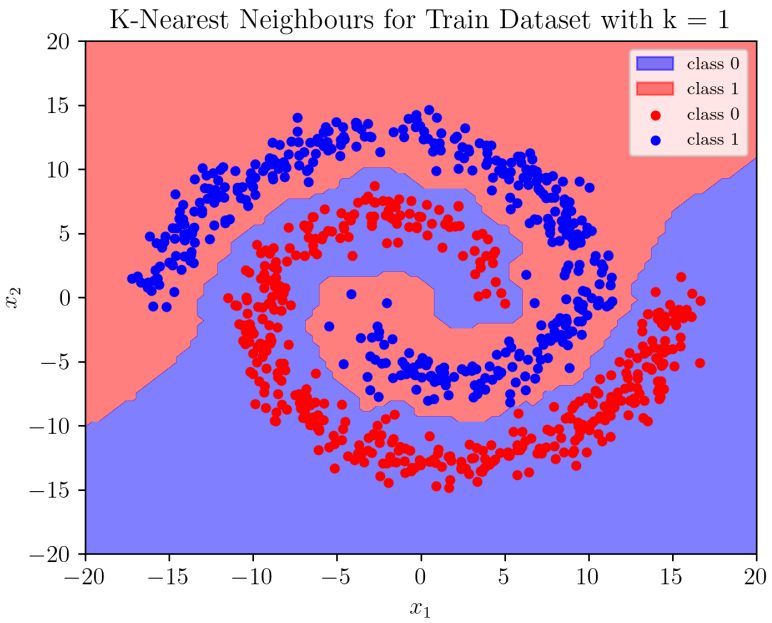
K (hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
1	100%	100%	100%
7	100%	99.58%	100%
15	99.88%	99.58%	99.17%

Table 8: KNN classification accuracies for varying hyperparameter k

Confusion Matrices for the best performing model : Train and Test accuracy = 100%

$$\text{Training Dataset} \quad \begin{bmatrix} 435 & 0 \\ 0 & 407 \end{bmatrix} \qquad \text{Test Dataset} \quad \begin{bmatrix} 68 & 0 \\ 0 & 52 \end{bmatrix}$$

Plots :



3.1.1 Observations :

KNN is still capable of identifying the classes pretty accurately. KNN also performs better as compared to the Bayes Classifier

3.2 Bayes Classifier

Bayes Classifier considers the features of the data matrix to follow gaussian distribution, it is represented as follows :

$$P(y_i|\bar{x}) = \frac{p(\bar{x}|y_i)P(y_i)}{p(\bar{x})}$$

for a given datapoint $p(\bar{x})$ remains unchanged

$$P(y_i) = \frac{N_i}{N}$$

Where N_i is number of points in class i and N is total number of data points

$$p(\bar{x}|y_i) = \frac{1}{\sqrt{(2\pi)^d|C_i|}} \cdot e^{\frac{-(\bar{x}-\mu_i)^T C_i^{-1}(\bar{x}-\mu_i)}{2}}$$

Where C_i is a full covariance matrix
The values for class covariance and mean can be obtained from sample mean and sample covariance corresponding to class i

C_0, C_1	**Training Accuracy**	**Validation Accuracy**	**Test Accuracy**
different	71.85%	75.21%	70%

Table 9: Bayes classification accuracies

Confusion Matrices for the best performing model : Accuracy : Train = 71.85% & Test = 70%

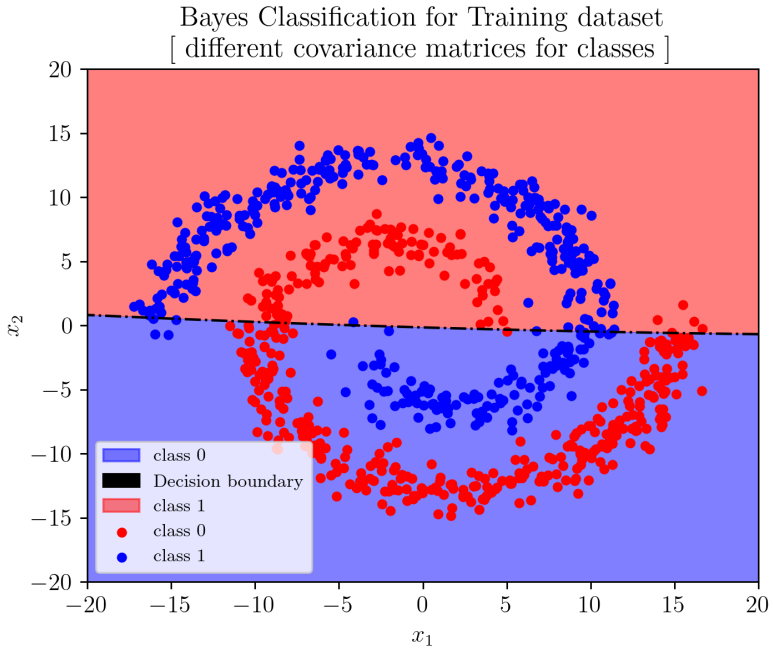
$$\begin{matrix}\text{Training Dataset} \\ \begin{bmatrix} 317 & 118 \\ 119 & 288 \end{bmatrix}\end{matrix} \quad \begin{matrix}\text{Test Dataset} \\ \begin{bmatrix} 50 & 18 \\ 18 & 34 \end{bmatrix}\end{matrix}$$

Decision Boundary Equation:

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 0.000116 & 0.00000143 \\ 0.00000143 & -0.000154 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -0.00781 & -0.20862 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 0.033886 = 0$$

$$0.000116x_1^2 + 0.00000286x_1x_2 - 0.000154x_2 - 0.00781x_1 - 0.20862x_2 - 0.033886 = 0$$

Plots :
Different Covariance Matrices



3.2.1 Observations :

Bayes Classifier Suffers in this case because it is only suitable for classifying linearly separable classes. As its decision boundary surface for two class problem can only be a second order equation at best, i.e the decision boundary can be a line, parabola, ellipse or a hyperbola. Thereby it cannot construct other nonlinear decision boundaries, hence the poor classification results.

3.3 Bayes classifier with a GMM for each class, using full covariance matrices

In this subsection, the classification task is done using Bayes classifier with a Gaussian Mixture Models, where each class is represented by **Q** Gaussians. The Bayes classifier, often employed in machine learning and pattern recognition, provides a probabilistic approach to classify data points into different categories or classes. When combined with a GMM, which models the underlying data distribution as a mixture of Gaussian components, we can create a powerful and flexible classification framework.
The method used to obtain results is : **Expectation-Maximization** method.

The **Expectation-Maximization (EM)** algorithm is an iterative optimization technique used to estimate the parameters of statistical models. EM consists of two main steps: the E-step (Expectation) and the M-step (Maximization).

1. **Initialization**

   Before starting the EM algorithm, we initialize the model parameters using the **k-means** clustering algorithm. It estimates the initial means and covariances of the clusters. The hyperparameter $\mathbf{k} = \mathbf{Q}$. N_q is number of points in q^{th} cluster.

$$\begin{aligned}
\bar{\theta} &= \left\{ w_q, \bar{\mu_q}, c_q \right\}_{q=1}^{Q} \\
\bar{\mu_q} &= \text{Average of examples in } q^{th} \text{ cluster} \\
c_q &= \frac{\sum_{x_n} (\bar{x_n} - \bar{\mu_q})(x_n - \bar{\mu_q})^t}{N_q} \\
w_q &= \frac{N_q}{N}
\end{aligned}$$

2. **E-step (Expectation)**

   In the **E-step**, we calculate the expected value of the log-likelihood function with respect to the current estimate of the model parameters, given the observed data. This step involves computing the posterior distribution over the hidden variables.

   For a given model with parameters θ and observed data X, the E-step aims to find the responsibility function γ_{nq}:

$$\gamma_{nq} = \frac{w_q \, \mathcal{N}(\bar{x_n} \bar{\mu_q}, c_q)}{\sum_{m=1}^{Q} w_m \, \mathcal{N}(\bar{x_n} \bar{\mu_m}, c_m)}$$

3. **M-step (Maximization)**

   In the M-step, we maximize the expected log-likelihood from the E-step with respect to the model parameters. This step involves updating the model parameters to maximize the expected log-likelihood.

   The M-step updates the parameters as follows:

$$\begin{aligned}
\bar{\theta^{new}} &= \left\{ w_q^{new}, \bar{\mu_q}^{new}, c_q^{new} \right\}_{q=1}^{Q} \\
N_q &= \sum_n \gamma_{nq} \\
\bar{\mu_q}^{new} &= \frac{\sum_{x_n} \gamma_{nq} \, \bar{x_n}}{N_q} \\
c_q^{new} &= \frac{\sum_{x_n} \gamma_{nq} \, (\bar{x_n} - \bar{\mu_q}^{new})(x_n - \bar{\mu_q}^{new})^t}{N_q} \\
w_q^{new} &= \frac{N_q}{N}
\end{aligned}$$

4. **Iterative Process** The EM algorithm iteratively alternates between the E-step and the M-step until convergence. The convergence is typically determined by a change in the log-likelihood or a tolerance threshold.

$$\Delta\mathcal{L} = \mathcal{L}(\mathcal{D}/\theta^{new}) - \mathcal{L}(\mathcal{D}/\theta^{old})$$

Q (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**# of Paramters per class**
3	98.10%	98.32%	18
5	99.64%	100%	30
8	99.76%	100%	48
10	100%	100%	60
15	100%	100%	90
20	100%	99.58%	120

Table 10: Classification accuracies of the model on training data and validation data for different values of hyperparameter

From Table 10, we can infer that the $\mathbf{Q = 10}$, $\mathbf{Q = 15}$ are the best models as the validation accuracy as well as training accuracy is **100%**. The decision region for the plots are given in Figure 1.

3.3.1 Confusion matrix for best performing model:

Q = 10

1. Training data:
   - **Accuracy:** 100%
   - **Confusion Matrix:** $\begin{bmatrix} 435 & 0 \\ 0 & 407 \end{bmatrix}$
2. Test data:
   - **Accuracy:** 100%
   - **Confusion Matrix:** $\begin{bmatrix} 68 & 0 \\ 0 & 52 \end{bmatrix}$

Q = 15

1. Training data:
   - **Accuracy:** 100%
   - **Confusion Matrix:** $\begin{bmatrix} 435 & 0 \\ 0 & 407 \end{bmatrix}$
2. Test data:
   - **Accuracy:** 100%
   - **Confusion Matrix:** $\begin{bmatrix} 68 & 0 \\ 0 & 52 \end{bmatrix}$

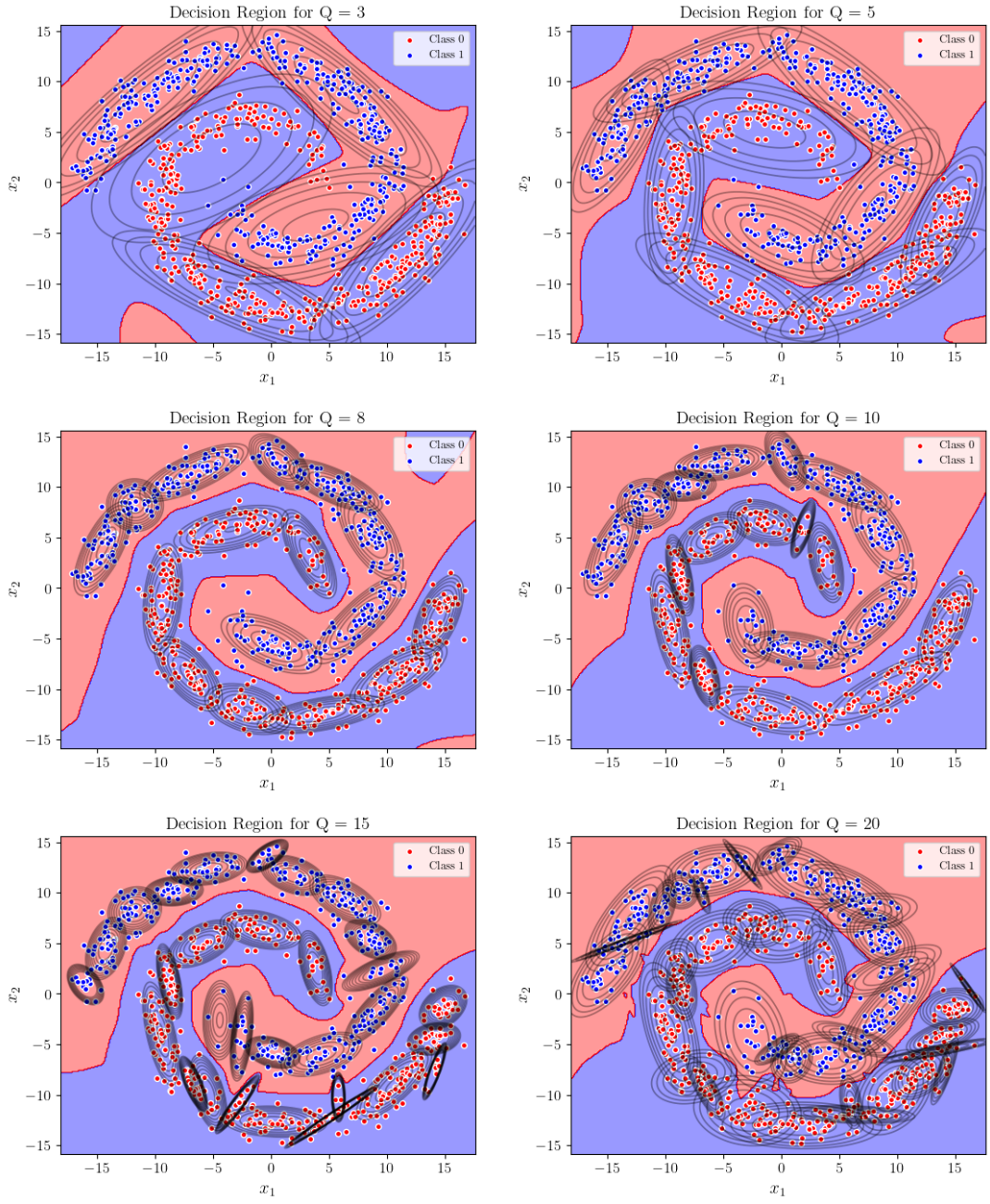


Figure 1: GMM decision plots for various **Q** with full covariance

3.3.2 Observations:

From Table 10 and Figure 1, we can observe that:

- Q = 3, 5, is underfitting the dataset
- Q = 8, 10, performs decently well but many a times the accuracy dips from 100%
- Q = 20, 15, performs the best all times and is therefore a better fit for the dataset.The decision region is also almost similar.
- As the data set in non-linearly separable, the decision boundary obtained from a GMM tends to be soft, probabilistic, and exhibits intricate shapes. Thus it offers capturing complex, non-linear relationships in the data, and so decision boundary is highly adaptable to the underlying data distribution.

3.4 Bayes classifier with a GMM for each class, using diagonal covariance matrices

In this subsection, the classification task is done using Bayes classifier with a Gaussian Mixture Models, where each class is represented by **Q** Gaussians. The difference between this and last subsection is that Covariance matrix here is **diagonal matrix**.

The method used to obtain results is: **Expectation-Maximization** method.(Section : 3.3)

Q (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**# of Paramters per class**
3	90.14252%	88.2353%	15
5	99.049881%	98.739496%	25
8	99.16865%	99.57983%	40
10	99.76247%	100%	50
15	100%	100%	75
20	100%	100%	100

Table 11: Classification accuracies of the model on training data and validation data for different values of hyperparameter

From Table 11, we can infer that the $\mathbf{Q = 15}$, $\mathbf{Q = 20}$ are the best models as the validation accuracy as well as training accuracy is **100%**. The decision region for the plots are given in Figure 2.

3.4.1 Confusion matrix for best performing model:

- **Q = 15**

  1. Training data:
     - **Accuracy:** 100%
     - **Confusion Matrix:** $\begin{bmatrix} 435 & 0 \\ 0 & 407 \end{bmatrix}$
  2. Test data:
     - **Accuracy:** 99.16667%
     - **Confusion Matrix:** $\begin{bmatrix} 67 & 1 \\ 0 & 52 \end{bmatrix}$

- **Q = 20**

  1. Training data:
     - **Accuracy:** 100%
     - **Confusion Matrix:** $\begin{bmatrix} 435 & 0 \\ 0 & 407 \end{bmatrix}$
  2. Test data:
     - **Accuracy:** 100%
     - **Confusion Matrix:** $\begin{bmatrix} 68 & 0 \\ 0 & 52 \end{bmatrix}$

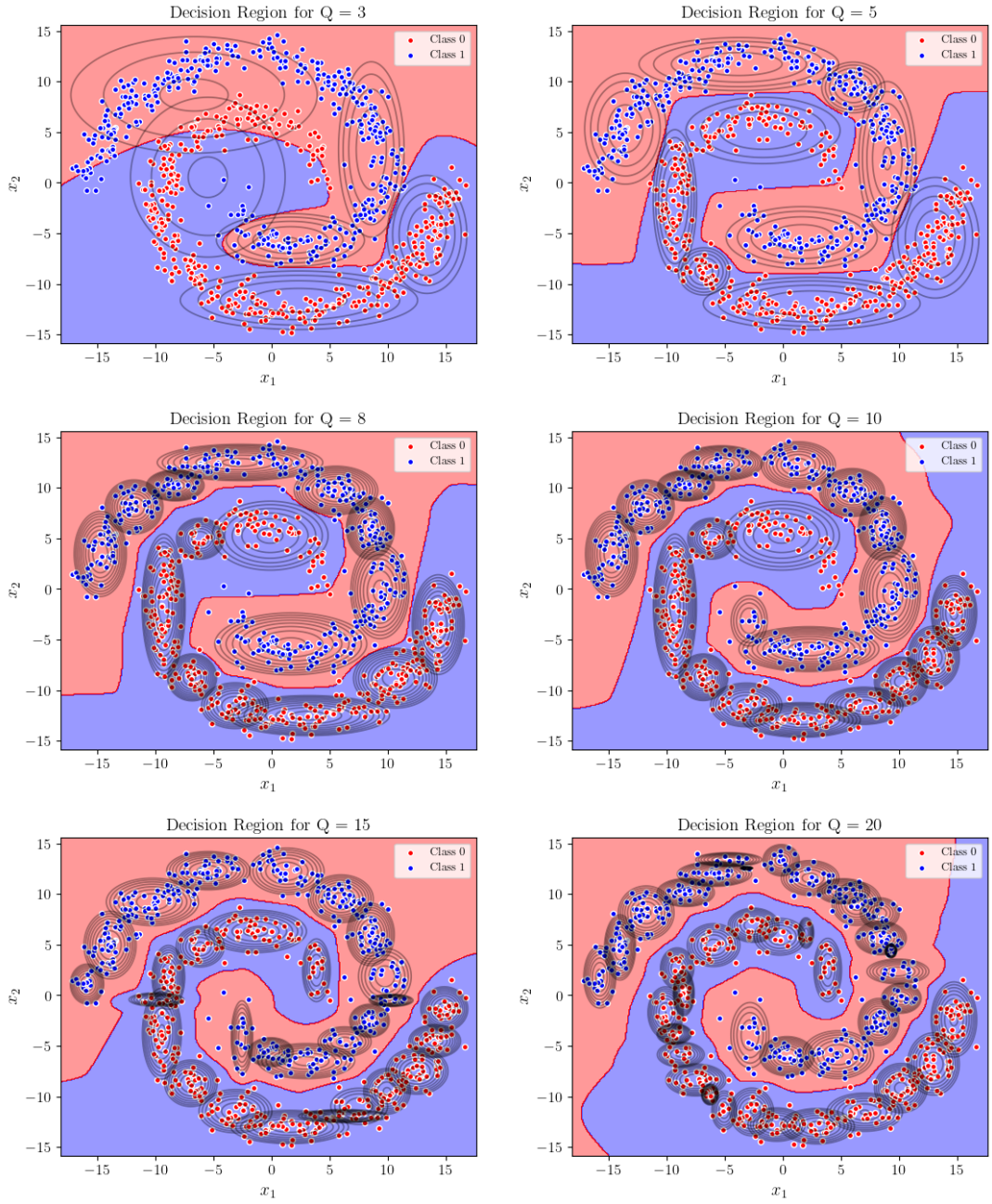


Figure 2: GMM decision plots for various **Q** with diagonal covariance

3.4.2 Observations:

From Table 11 and Figure 2, we can observe that:

- Q=3 gaussian is underfitting the dataset
- Q=5,8 performs decently well but many a times the accuracy dips from 100%
- Q=10,15 performs the best all times and is therefore a better fit for the dataset.The decision region is also almost similar.
- The difference between full and diagonal covariance can be observed in the shape of level curves where the latter has curves parallel to axes.
- In diagonal covariance matrices, it is assumed that the features (dimensions) are independent of each other within each component. Thus the decision regions in these GMM's tend to be axis-aligned, forming hyperellipses.

3.5 Bayes classifier with hypersphere based Parzen window method for density estimation

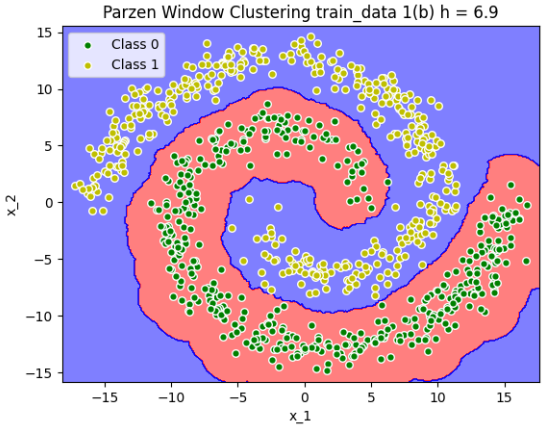
h (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
3	100%	100%	100%
6.9	99.88%	100%	100%
10	99.88%	99.58%	99.17%
50	90.38%	91.18%	85.83%
100	75.53%	77.73%	71.67%

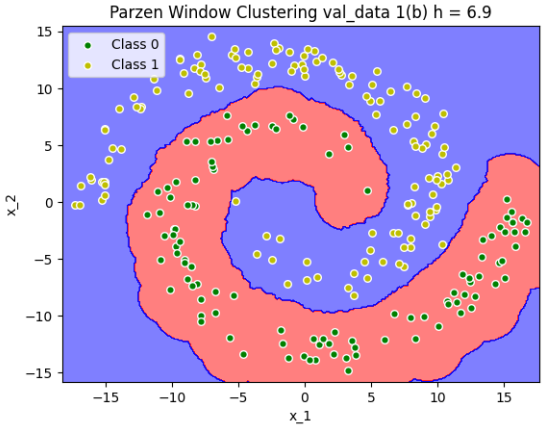
Table 12: Classification accuracies of the model on training data, validation data and training data for different values of hyperparameter

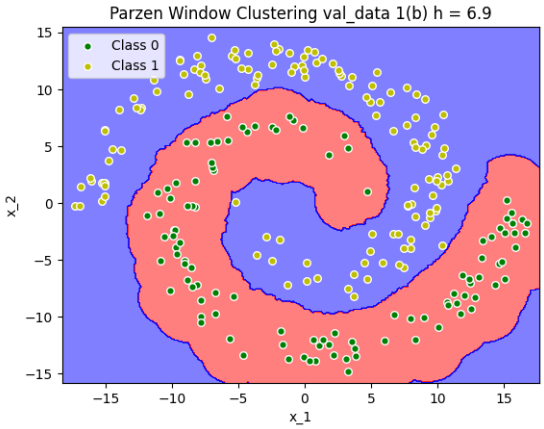
The accuracy and confusion matrix for best model is as follows:

- **h = 3**
  1. Training data:
     - **Accuracy:** 100%
     - **Confusion Matrix:** $\begin{bmatrix} 435 & 0 \\ 0 & 407 \end{bmatrix}$
  2. Test data:
     - **Accuracy:** 100%
     - **Confusion Matrix:** $\begin{bmatrix} 68 & 0 \\ 0 & 52 \end{bmatrix}$
- **h = 6.9**
  1. Training data:
     - **Accuracy:** 99.88%
     - **Confusion Matrix:** $\begin{bmatrix} 435 & 0 \\ 1 & 406 \end{bmatrix}$
  2. Test data:
     - **Accuracy:** 100%
     - **Confusion Matrix:** $\begin{bmatrix} 68 & 0 \\ 0 & 52 \end{bmatrix}$

Plots :







3.5.1 Observations:

The Bayes classifier with Parzen Window estimation excels when using a radius of 6.9. Smaller Parzen hyperspheres tend to overfit, leading to a noticeable drop in test data performance compared to training data for lower radius values.

3.6 Bayes classifier with K-nearest neighbours method for estimation of class-conditional probability density function, for K=10 and K=20

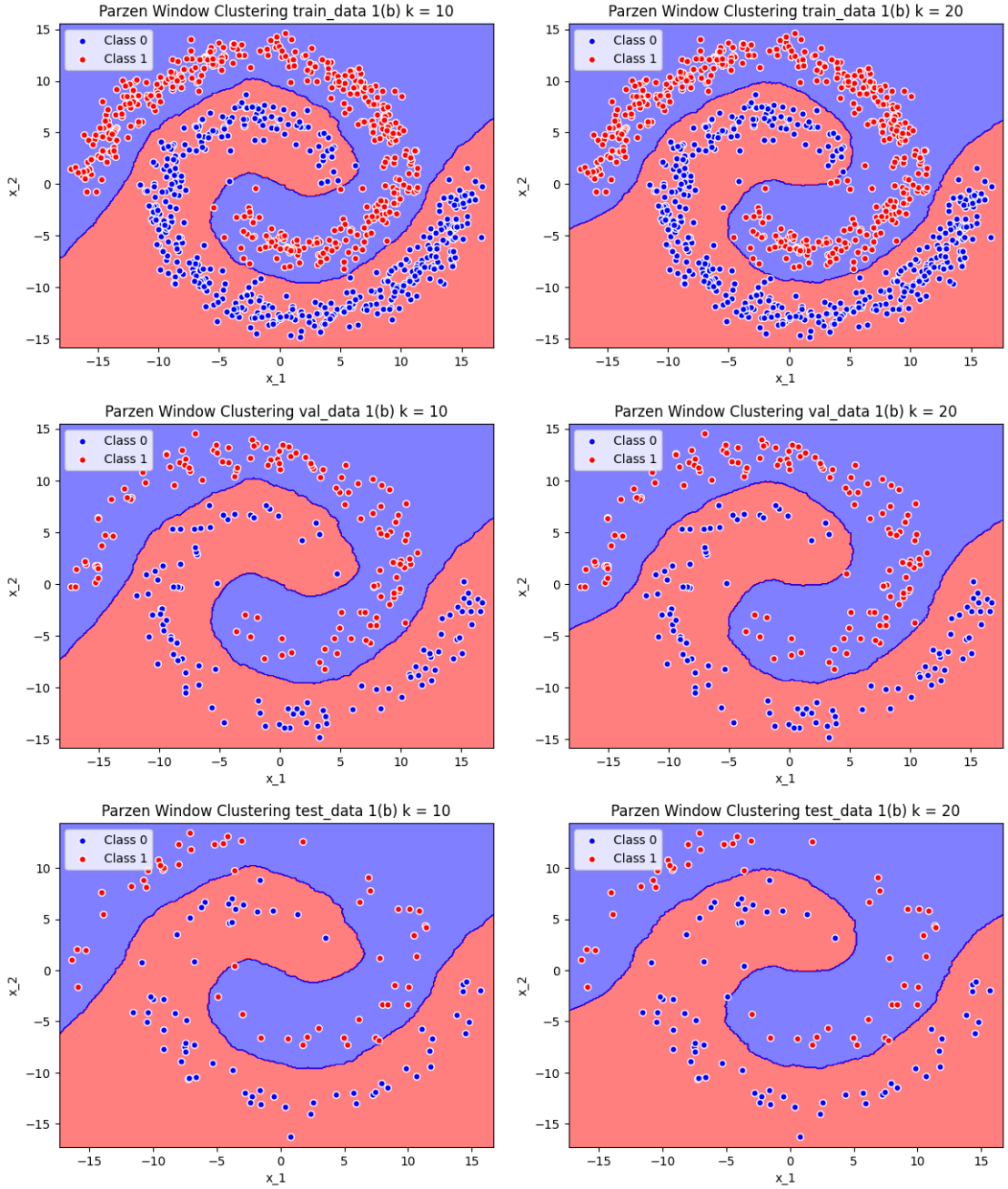
K (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
10	99.64%	99.58%	99.17%
20	99.29%	99.16%	97.5%

Table 13: Classification accuracies of the model on training data, validation data and training data for different values of hyperparameter

The accuracy and confusion matrix for best model is as follows:

- **K = 10**
  1. Training data:
     - **Accuracy:** 99.64%
     - **Confusion Matrix:** $\begin{bmatrix} 434 & 1 \\ 2 & 405 \end{bmatrix}$
  2. Test data:
     - **Accuracy:** 99.17%
     - **Confusion Matrix:** $\begin{bmatrix} 67 & 1 \\ 0 & 52 \end{bmatrix}$
- **K = 20**
  1. Training data:
     - **Accuracy:** 99.29%
     - **Confusion Matrix:** $\begin{bmatrix} 431 & 4 \\ 2 & 405 \end{bmatrix}$
  2. Test data:
     - **Accuracy:** 97.5%
     - **Confusion Matrix:** $\begin{bmatrix} 67 & 1 \\ 2 & 50 \end{bmatrix}$

Plots :



3.6.1 Observations:

The Bayes classifier with K-nearest neighbor likelihood estimation exhibits consistent performance across different values of k. This stability is because slight changes in the decision boundary have minimal impact on model performance, as the decision boundary typically lies near the midpoint between clusters of the respective class data.

4 Dataset 2

This Dataset consists of Image data set for 5 classes
(Dimension of feature vector is 81) i.e, $N \times 81$

File Name	Data Points
train data	3500×81
val data	1000×81
test data	500×81

4.1 K - Nearest Neighbours Classification

Similar to section 1.1 we compute the nearest **K** neighbors using euclidean distance and assign a class to our datapoint same as the maximum repeated classes in those k neighboring points.

K (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**Test Accuracy**
1	100%	48.40%	43.80%
7	66.20%	51.70%	52.60%
15	62.23%	55.00%	50.80%

Confusion Matrices for best performing model : Train = 62.23% & Test = 50.80%

Training Dataset

$$\begin{bmatrix} 365 & 47 & 116 & 32 & 140 \\ 34 & 328 & 245 & 61 & 32 \\ 19 & 36 & 566 & 58 & 21 \\ 11 & 38 & 215 & 425 & 11 \\ 61 & 24 & 86 & 35 & 494 \end{bmatrix}$$

Test Dataset

$$\begin{bmatrix} 34 & 10 & 19 & 5 & 32 \\ 6 & 32 & 46 & 14 & 2 \\ 4 & 6 & 73 & 13 & 4 \\ 0 & 4 & 45 & 51 & 0 \\ 14 & 4 & 14 & 4 & 64 \end{bmatrix}$$

4.2 Bayes Classifier

Similar to section 2.2 we obtain the covariance and mean for each class using the sample covariance and sample means of each class. The class for any datapoint is determined using

$$\text{class label of } \bar{x} = \arg\max_{y_i} p(\bar{x}|y_i)P(y_i)$$

We vary across all classes and compute the above value, we assign the class corresponding to maximum function value to our datapoint.

Covariance Matrix	**Training Accuracy**	**Validation Accuracy**	**Test Accuracy**
different	85.4%	53.5%	56.00%

Confusion Matrices for best performing model : Train = 85.4% & Test = 56%

Training Dataset

$$\begin{bmatrix} 545 & 30 & 17 & 22 & 86 \\ 33 & 574 & 38 & 24 & 31 \\ 17 & 17 & 617 & 27 & 22 \\ 8 & 19 & 32 & 629 & 12 \\ 21 & 27 & 15 & 13 & 624 \end{bmatrix}$$

Test Dataset

$$\begin{bmatrix} 44 & 18 & 5 & 7 & 26 \\ 10 & 63 & 11 & 11 & 5 \\ 4 & 21 & 49 & 19 & 7 \\ 0 & 9 & 25 & 61 & 5 \\ 16 & 6 & 6 & 9 & 63 \end{bmatrix}$$

4.3 Bayes classifier with a GMM for each class, using full covariance matrices

In this subsection, the classification task is done using Bayes classifier with a Gaussian Mixture Models, where each class is represented by **Q** Gaussians.

The method used to obtain results is: **Expectation-Maximization** method.(Section : 3.3)

Q (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**# of Paramters per class**
1	85.4%	53.5%	6
2	94.77%	51.6%	12
3	98.57%	45.9%	18
4	99.37%	43.4%	24
5	99.88%	37.1%	30
6	100%	39%	36
7	99.85%	39.1%	42
10	99.88%	26.8%	60
15	100%	23%	90
20	100%	20.5%	120

Table 14: Classification accuracies of the model on training data and validation data for different values of hyperparameter

From Table 14, we can infer that the $\mathbf{Q = 1}$, and to some extent $\mathbf{Q = 2}$ are the best models as the validation accuracy is greatest among the rest.

4.3.1 Confusion matrix for best performing model:

Q = 1

1. Training data:
   - **Accuracy:** 85.4%
   - **Confusion Matrix:**
$$\begin{bmatrix} 545 & 30 & 17 & 22 & 86 \\ 33 & 574 & 38 & 24 & 31 \\ 17 & 17 & 617 & 27 & 22 \\ 8 & 19 & 32 & 629 & 12 \\ 21 & 27 & 15 & 13 & 624 \end{bmatrix}$$
2. Test data:
   - **Accuracy:** 52.6%
   - **Confusion Matrix:**
$$\begin{bmatrix} 83 & 24 & 18 & 10 & 65 \\ 22 & 108 & 34 & 23 & 13 \\ 23 & 51 & 88 & 28 & 10 \\ 8 & 32 & 37 & 113 & 10 \\ 23 & 15 & 13 & 6 & 143 \end{bmatrix}$$

4.3.2 Observations:

From Table 14 , we can observe that:

- Q=1 gaussian best model as it has greater validation accuracy

- Q=2, 3 performs decently well with the image dataset
- All other Q's overfit the data set with training accuracy $\approx 100\%$.
- In high-dimensional spaces such as the image dataset with **81** features, the model has become overparameterized, leading to overfitting. This resulted in poor generalization to new data.
- As dataset is small compared to number of features, the model doesn't generalize well. GMMs assume that the data is generated from a mixture of Gaussian distributions which may not align with the complex and diverse patterns and features present in image dataset, as it can be highly non-Gaussian.
- The GMM model has low accuracy in validation due to the nature of GMMs modeling the global statistical properties of data rather than capturing local features or detailed patterns within the data, such as those found in this image dataset.

4.4 Bayes classifier with a GMM for each class, using diagonal covariance matrices

In this subsection, the classification task is done using Bayes classifier with a Gaussian Mixture Models, where each class is represented by **Q** Gaussians. The difference between this and last subsection(4.3) is that Covariance matrix here is **diagonal matrix**.

The method used to obtain results is: **Expectation-Maximization** method.(Section : 3.3)

Q (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**# of Paramters per class**
1	55.4%	52.9%	5
2	56.57%	52.7%	10
3	57.71429%	53.2%	15
4	60.37%	52.4%	20
5	62.89%	51.5%	25
6	64.34%	51.8%	30
7	65.66%	50.6%	35
10	70.63%	51.6%	50
15	77.86%	53.0%	75
20	82.54%	52.5%	100

Table 15: Classification accuracies of the model on training data and validation data for different values of hyperparameter

From Table 15, we can infer that the $\mathbf{Q = 3}$, and to some extent $\mathbf{Q = 15}$ are the best models as the validation accuracy is greatest among the rest.

4.4.1 Confusion matrix for best performing model:

Q = 3

1. Training data:
   - **Accuracy:** 57.71429%
   - **Confusion Matrix:**
$$\begin{bmatrix} 319 & 113 & 84 & 31 & 153 \\ 40 & 400 & 135 & 81 & 44 \\ 28 & 102 & 443 & 96 & 31 \\ 4 & 107 & 151 & 410 & 28 \\ 105 & 66 & 59 & 22 & 448 \end{bmatrix}$$
2. Test data:
   - **Accuracy:** 51.4%
   - **Confusion Matrix:**
$$\begin{bmatrix} 84 & 30 & 23 & 9 & 54 \\ 9 & 97 & 49 & 28 & 17 \\ 11 & 29 & 115 & 30 & 15 \\ 2 & 36 & 44 & 112 & 6 \\ 31 & 21 & 19 & 5 & 124 \end{bmatrix}$$

4.4.2 Observations:

From Table 15 , we can observe that:

- Q = 3 gaussian best model as it has greater validation accuracy
- Q = 1, 2 performs decently well with the image dataset.
- All other Q's overfit the data set with training accuracy $\approx 100\%$.
- In high-dimensional spaces such as the image dataset with **81** features, the model has become overparameterized, leading to overfitting. This resulted in poor generalization to new data.
- In diagonal covariance matrices, it is assumed that the features (dimensions) are independent of each other within each component. Thus the decision regions in these GMM's tend to be axis-aligned, forming hyperellipses.
- The GMM model has low accuracy in validation due to the nature of GMMs modeling the global statistical properties of data rather than capturing local features or detailed patterns within the data, such as those found in this image dataset.

4.5 Bayes classifier with hypersphere based Parzen window method for density estimation

h (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
0.5	55.85%	50.35%	49.90%
1	31.38%	32.33%	30.46%
2	20.01%	20.02%	20.04%

Table 16: Classification accuracies of the model on training data, validation data and training data for different values of hyperparameter

The accuracy and confusion matrix for best model is as follows:

- **h = 0.5**
  1. Training data:
     - **Accuracy:** 55.85%
     - **Confusion Matrix:**
$$\begin{bmatrix} 280 & 23 & 161 & 330 & 202 \\ 25 & 216 & 377 & 46 & 36 \\ 7 & 6 & 591 & 58 & 38 \\ 3 & 16 & 308 & 360 & 13 \\ 37 & 10 & 120 & 26 & 507 \end{bmatrix}$$
  2. Test data:
     - **Accuracy:** 49.90%
     - **Confusion Matrix:**
$$\begin{bmatrix} 27 & 5 & 26 & 6 & 35 \\ 7 & 21 & 56 & 12 & 4 \\ 1 & 2 & 88 & 4 & 5 \\ 0 & 2 & 51 & 47 & 0 \\ 10 & 3 & 18 & 3 & 66 \end{bmatrix}$$

4.5.1 Observations :

The Bayes classifier with Parzen Window estimation yields unsatisfactory results due to the sparse and non-smooth nature of the image data. This makes it challenging for Parzen hyperspheres to effectively estimate class conditional probability. Additionally, controlling density estimation is problematic because class conditional data can be widely dispersed, leading to poor performance.

4.6 Bayes classifier with K-nearest neighbours method for estimation of class-conditional probability density function, for K=10 and K=20

K (Hyperparameter)	**Training Accuracy**	**Validation Accuracy**	**Testing Accuracy**
10	59.45%	53.25%	50.50%
20	57.02%	52.25%	49.70%

Table 17: Classification accuracies of the model on training data, validation data and training data for different values of hyperparameter

The accuracy and confusion matrix for best model is as follows:

- **K = 10**
  1. Training data:
     - **Accuracy:** 59.45%
     - **Confusion Matrix:**
$$\begin{bmatrix} 301 & 40 & 140 & 48 & 170 \\ 24 & 262 & 302 & 69 & 43 \\ 12 & 16 & 562 & 76 & 34 \\ 3 & 24 & 223 & 437 & 13 \\ 44 & 15 & 96 & 27 & 518 \end{bmatrix}$$
  2. Test data:
     - **Accuracy:** 50.50%
     - **Confusion Matrix:**
$$\begin{bmatrix} 33 & 3 & 22 & 10 & 31 \\ 5 & 28 & 50 & 15 & 2 \\ 1 & 4 & 74 & 15 & 6 \\ 1 & 2 & 44 & 53 & 0 \\ 11 & 3 & 16 & 6 & 64 \end{bmatrix}$$
- **K = 20**
  1. Training data:
     - **Accuracy:** 57.02%
     - **Confusion Matrix:**
$$\begin{bmatrix} 275 & 33 & 169 & 39 & 183 \\ 27 & 220 & 349 & 68 & 36 \\ 6 & 10 & 580 & 71 & 33 \\ 5 & 20 & 255 & 407 & 13 \\ 36 & 16 & 114 & 21 & 513 \end{bmatrix}$$
  2. Test data:
     - **Accuracy:** 49.70%
     - **Confusion Matrix:**
$$\begin{bmatrix} 28 & 4 & 23 & 10 & 34 \\ 7 & 22 & 51 & 17 & 3 \\ 1 & 3 & 79 & 10 & 7 \\ 1 & 1 & 45 & 53 & 0 \\ 10 & 4 & 15 & 5 & 66 \end{bmatrix}$$

4.6.1 Observations :

The Bayes classifier with K-nearest neighbor likelihood estimation outperforms the Parzen window method. It allows users to fine-tune the model by setting the number of samples and finding the volume, enabling the inclusion of more data in each window compared to the Parzen approach. However, a noticeable bias toward classes with more data is observed, indicating the K-nearest neighbor density estimation's difficulty in handling imbalanced classes.