E1 213 Pattern Recognition and Neural Network
Assignment 01 Report
Group 24

SAPTARSHI MANDAL

22925
MTech(SP)
saptarshiman@iisc.ac.in

HARSHIT DUBEY
23043
MTech(AI)
harshitdubey@iisc.ac.in

OINDRILA HALDAR
22501
MTech(SP)
oindrilah@iisc.ac.in

March 2024

1 Regression Task

Q1: Multilinear Regression

The given problem is a Multi-linear Regression Problem with 3 targets.

The problem is solved by solving an Emperical Risk Minimization using Gradient Descent Method.

Let Dataset $D = \{(X_j, Y_j)\}_{j=1}^{N}$

$$X_j = \begin{pmatrix} x_j^1 \\ x_j^2 \\ x_j^3 \\ x_j^4 \\ x_j^5 \\ x_j^6 \\ x_j^7 \\ x_j^8 \\ x_j^9 \\ x_j^{10} \end{pmatrix}; Y_j = \begin{pmatrix} y_j^1 \\ y_j^2 \\ y_j^3 \end{pmatrix}$$

Loss:

- ℓ_2 Loss : $||h(x) - y||_2^2$

For ℓ_2 Loss:

Gradient of Loss w.r.t. each y^i is:

$$\nabla_W L = \frac{1}{N} \sum_{j=1}^{N} 2(W_i^T X_j - y_j^i) X_j$$

where, W_i is the Weight Vector corresponding to i^{th} dimension of target(Y) : y^i

After the optimization, the Correlation Plots for different dimensions of target y are given in Figure. 1:

The Results are given in the Table. **??**

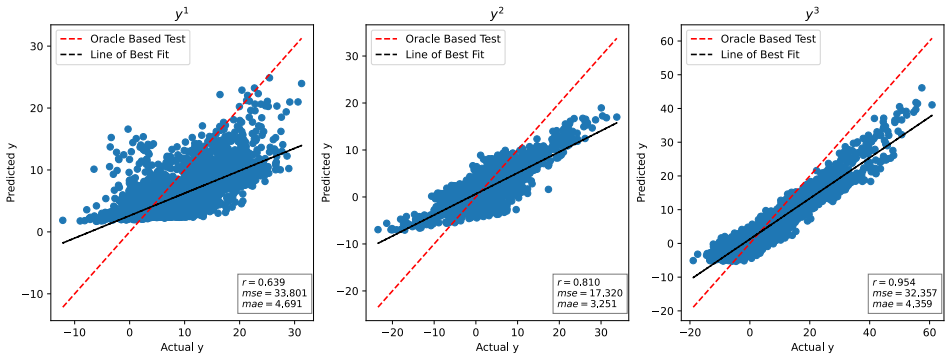
Learning Rates	mse	mae	r
0.1	**8.423, 8.581, 8.631**	**2.318, 2.342, 2.349**	**0.889, 0.891, 0.971**
0.01	10.705, 10.202, 9.374	2.624, 2.552, 2.446	0.860, 0.871, 0.968
0.001	33.801, 17.320, 32.357	4.691, 3.251, 4.359	0.639, 0.810, 0.954

Table 1: Multilinear Regression Results

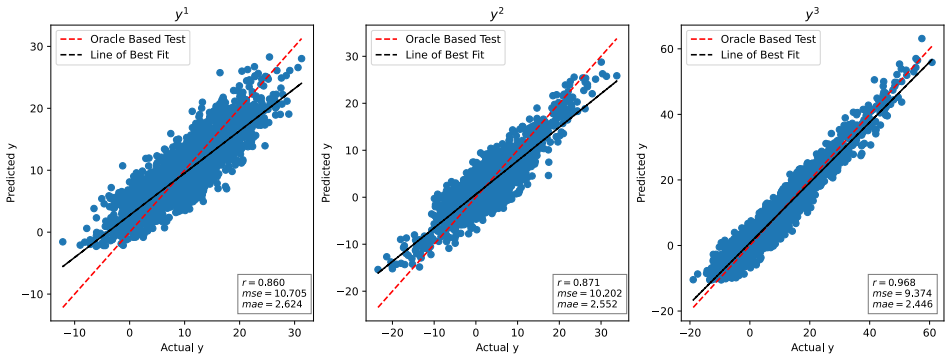
The Loss plots are given in Figure. 2

Conclusions based on learning rate:

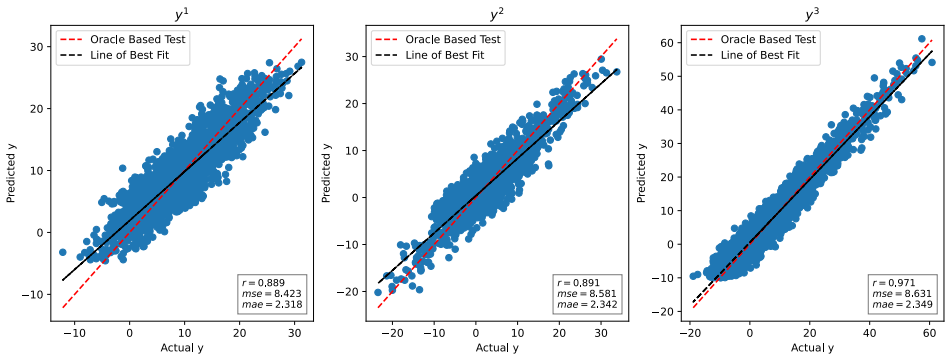
1. **Mean Squared Error (MSE)**: There is a direct correlation between the learning rate and the MSE. As the learning rate decreases, the MSE increases, indicating a rise in the prediction error.



(a) learning rate = 0.001



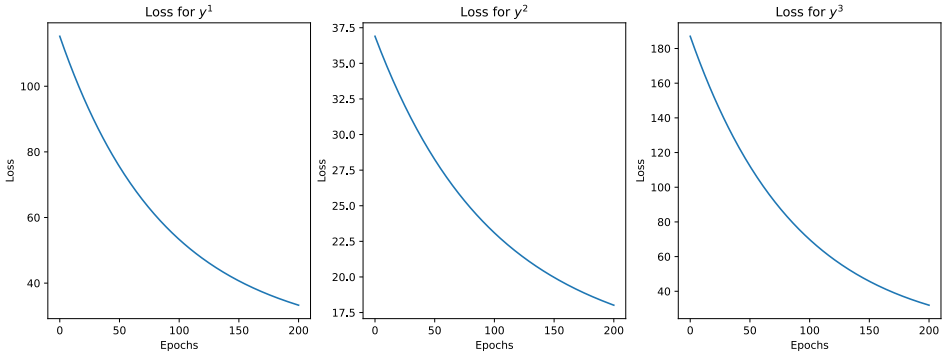
(b) learning rate = 0.01



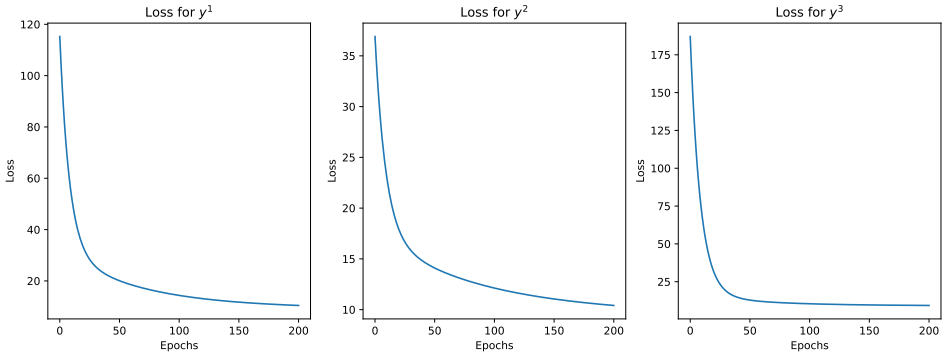
(c) learning rate = 0.1

Figure 1: Correlation Comparison of Different Learning Rates

2. **Mean Absolute Error (MAE)**: The MAE follows a similar trend to the MSE. A decrease in the learning rate results in an increase in the MAE, suggesting larger deviations in the model's predictions.



(a) learning rate = 0.001



(b) learning rate = 0.01

Figure 2: Loss Comparison of Different Learning Rates

3. **Correlation Coefficient (r)**: The r values exhibit an inverse relationship with the learning rate. A decrease in the learning rate leads to a decrease in the r values, implying a weaker correlation between the predicted and actual values.

These trends are consistent across all three dimensions. It can be inferred that a higher learning rate may enhance model performance concerning these metrics. However, an excessively high learning rate (for e.g. Learning rate = 1) may cause the model to overshoot the optimal point, emphasizing the importance of balance in selecting the learning rate.

Conclusions: loss vs epoch for learning rate:

- Learning Rate 0.001: Gradual loss reduction over epochs indicates steady learning and gradual prediction improvement.
- Learning Rate 0.01: Rapid loss reduction and early stabilization suggest faster initial learning but potential early convergence.

These trends are consistent across all dimensions (y1, y2, y3). Higher learning rates may lead to faster learning but risk early convergence and missing the global minimum. Lower rates ensure thorough solution space exploration, potentially leading to better models.

Q2: Generalised Regression with polynomial kernel

The given problem is a Regression Problem with with 3 targets. The problem is solved by using Polynomial Kernels and analysis is done by increasing the degree of polynomial kernel. The Performance Varies Greatly with the degree. Let Dataset $D = \{(X_j, Y_j)\}_{j=1}^{N}$

$$X_j = \begin{pmatrix} x_j^1 \\ x_j^2 \end{pmatrix}; Y_j = \begin{pmatrix} y_j^1 \\ y_j^2 \\ y_j^3 \end{pmatrix}$$

Let the polynomial function be : $\phi(X_j)$
Loss : ℓ_2 Loss : $||h(x) - y||_2^2$ where, h(x) = predicted target, y = actual target
For each dimension of target (y^i):
For ℓ_2 Loss:

$$L = \frac{1}{N}\sum_{j=1}^{N}(W_i^T\phi(X_j) - y_j^i)^2$$

Gradient of Loss w.r.t. each y^i is:

$$\nabla_{W_i} L = \frac{1}{N}\sum_{j=1}^{N} 2(W_i^T\phi(X_j) - y_j^i)\phi(X_j)$$

where, W_i is the Weight Vector corresponding to i^{th} dimension of target(Y) : y^i
The following Polynomial Kernels are used:

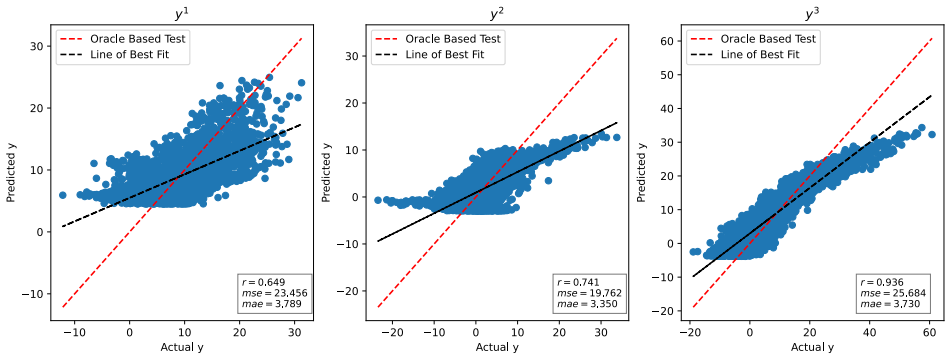
$$\phi_1(X_j) = \begin{pmatrix} x_j^1 \\ x_j^2 \\ x_j^1 x_j^2 \\ (x_j^1)^2 \\ (x_j^2)^2 \\ 1 \end{pmatrix} \quad \phi_2(X_j) = \begin{pmatrix} x_j^1 \\ x_j^2 \\ x_j^1 x_j^2 \\ (x_j^1)^2 \\ (x_j^2)^2 \\ (x_j^1)^2(x_j^2) \\ (x_j^2)^2(x_j^1) \\ (x_j^1)^3 \\ (x_j^1)^3 \\ 1 \end{pmatrix}$$

where ,$\phi_1(X_j)$ and $\phi_2(X_j)$ are polynomial kernels of degree 2 and 3 respectively **For $\phi_1(X_j)$ and $\phi_2(X_j)$ the Correlation Plots for different dimensions of target y are given in Figure. 3:**
Observations:

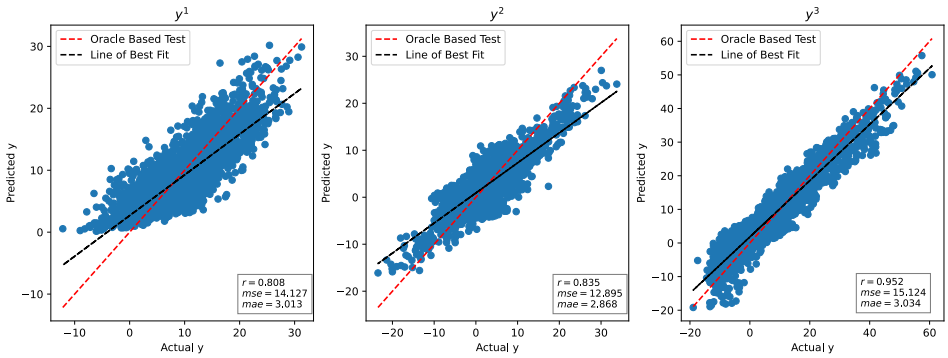
1. From the above curves we can see that the Cubic Kernel is Performing Better than the Quadratic Kernel. This Observation can indicate that the dataset exhibits higher-order non-linearities than quadratic.

Now, Fixing the kernel the Learning Rate is varied. Two different Learning Rates were chosen - 0.1 and 0.01. The Loss plots are given in Figure. 4
Observation:

1. From the plots we can see the a higher learning rate (0.1) is proving to be better.



(a) Quadratic Kernel (degree=2)

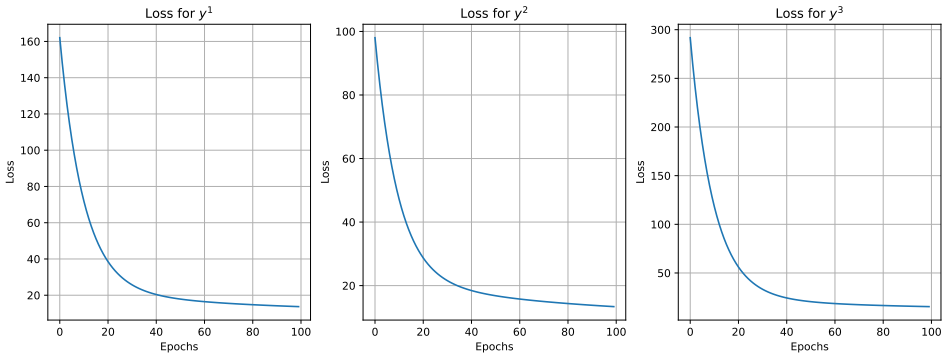


(b) Cubic Kernel (degree=3)

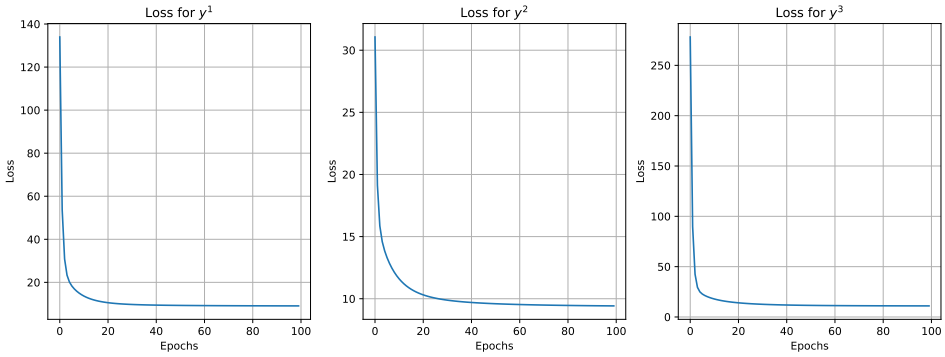
Figure 3: Correlation Comparison for different Kernels

2. Also, we can observe from the loss vs epochs curves that due to higher learning rate, the loss is decreasing faster.

In all problems, number of epochs is chosen as 100.



(a) learning rate = 0.01



(b) learning rate = 0.1

Figure 4: Comparison of Quadratic vs Cubic Kernel

Q3. Generalised Regression with non-polynomial kernel

The given problem is a Regression Problem with a single dimensional target. The problem is solved by using Non-Polynomial Kernels and analysis is done by changing the kernel.

Let the polynomial function be : $\phi(.)$.

Like the previous questions the loss and the gradient equation can be given as follows:

For ℓ_2 Loss:

$$L = \frac{1}{N}\sum_{j=1}^{N}(W_i^T\phi(X_j) - y)^2$$

Gradient of Loss w.r.t. each y^i is:

$$\nabla_W L = \frac{1}{N}\sum_{j=1}^{N}2(W^T\phi(X_j) - y)\phi(X_j)$$

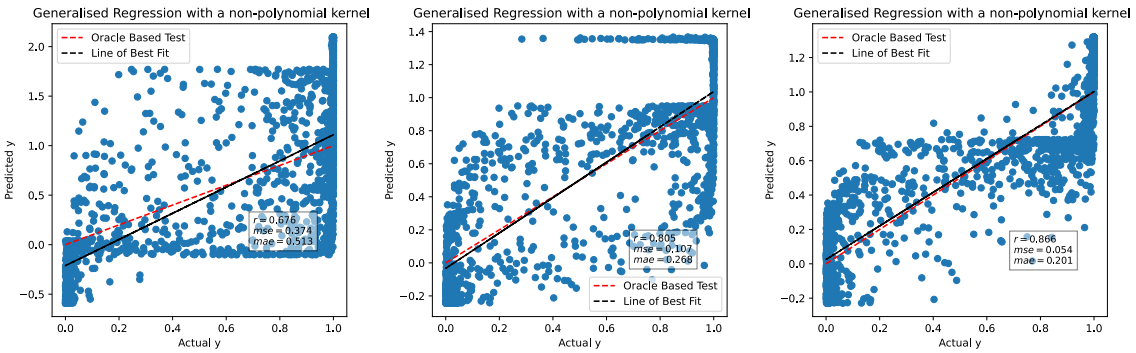
where, W_i is the Weight Vector corresponding to i^{th} dimension of target(Y) : y

The following Non-Polynomial Kernels are used:

$$\phi_1(X_j) = \begin{pmatrix} x_j \\ e^{-x_j^2} \\ 1 \end{pmatrix} \quad \phi_2(X_j) = \begin{pmatrix} x_j \\ \frac{1}{1+e^{-x_j}} \\ 1 \end{pmatrix} \quad \phi_3(X_j) = \begin{pmatrix} x_j \\ e^{x_j} \\ sin(x_j) \\ 1 \end{pmatrix}$$

where $x_j = \begin{pmatrix} x_j^1 & x_j^2 & x_j^3 & x_j^4 & x_j^5 \end{pmatrix}^T$

For $\phi_1(X_j)$, $\phi_2(X_j)$ and $\phi_3(X_j)$ the Correlation Plots given in Figure. 5:



(a) ϕ_1 (b) ϕ_2 (c) ϕ_3

Figure 5: Correlation Plots for different Non-Polynomial Kernel

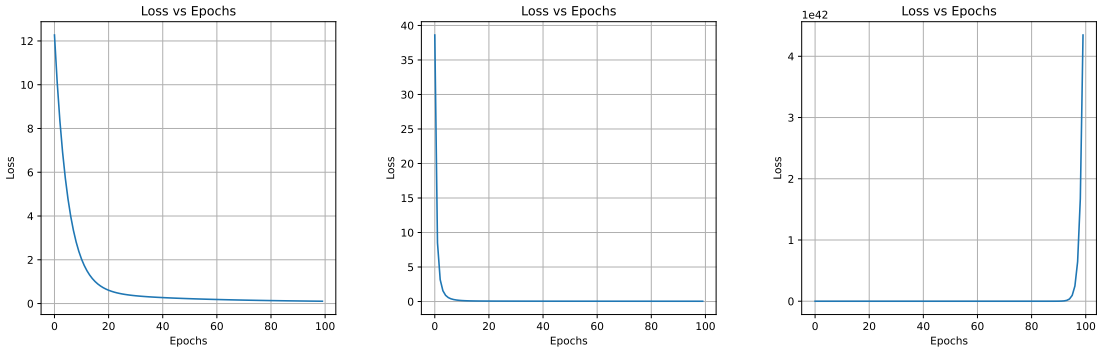
Observation - ϕ_3 performs better than ϕ_2, which in turn performs better than ϕ_1

Here are some possible reasons :

- ϕ_3 has a higher-dimensional feature space compared to ϕ_2 and ϕ_1. This increased complexity allows the model to capture more intricate patterns and relationships in the data, leading to potentially better performance.

- ϕ_3 includes additional nonlinear features such as e^{x_j} and $\sin(x_j)$, which can better represent the underlying data distribution compared to the simpler features in ϕ_2 and ϕ_1.
- ϕ_3 includes a diverse set of features, including exponential and trigonometric functions, which offer a wider range of transformations on the input data. This diversity allows the model to capture a broader range of data patterns and variations, potentially leading to improved generalization performance.

Now, we fix the kernel as ϕ_3 and vary the learning rate. We choose the following learning rates - 0.01 and 0.05. The loss plots are given in Figure.6



(a) rate = 0.01 (b) rate = 0.05 (c) rate = 0.1

Figure 6: Loss Plots for different learning rates

Observation - From the plots we can see the a higher learning rate (0.05) is proving to be better. Also, we can observe from the loss vs epochs curves that due to higher learning rate, the loss is decreasing faster. In all problems, number of epochs is chosen as 100.

Note: Choosing a very high value of learning rate blows up the optimization and Loss starts increasing instead of decreasing. In this problem choosing a learning rate >= 0.1 showed such a phenomenon.

2 Classification Analysis

Q4. Binary Classification

Bayes Classifier (Parametric Density Estimation - Normal)

In this Classification problem we assume each class conditional density to be a Gaussian Densitiy and we estiamte the parameters of the Gaussian Density from the given Dataset.

Calculating Parameters for Gaussian Density: Mean: The mean of a set of vectors $x_1, x_2, \ldots, x_N$ is calculated as:

$$\text{mean} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Covariance Matrix: The covariance matrix of a set of vectors $x_1, x_2, \ldots, x_N$ is calculated as:

$$\text{covariance matrix} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \text{mean})(x_i - \text{mean})^T$$

In these expressions:

- N is the number of vectors.
- x_i is the i-th vector.
- mean is the mean vector.
- $(x_i - \text{mean})$ is the difference between the i-th vector and the mean vector.
- $(x_i - \text{mean})^T$ is the transpose of this difference.

The outer product of these two vectors gives a matrix, and the sum of these matrices divided by N gives the covariance matrix.

The probability density function (PDF) of a multivariate Gaussian (or normal) distribution is given by:

$$f(x|\mu, \Sigma) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

Here's what each symbol represents:

- $f(x|\mu, \Sigma)$: This is the PDF of the multivariate Gaussian distribution. Given a vector x, and parameters μ and Σ, it gives the probability density of x.
- x: This is a vector in $\mathbb{R}^k$ (a k-dimensional real vector). Each element of x is one variable in the multivariate distribution.
- μ: This is the mean vector of the distribution. It is also a vector in $\mathbb{R}^k$. Each element of μ is the mean of the corresponding variable in x.
- Σ: This is the covariance matrix of the distribution. It is a $k \times k$ matrix. The element at the i-th row and j-th column of Σ is the covariance between the ith and jth variables in x.

Observations:

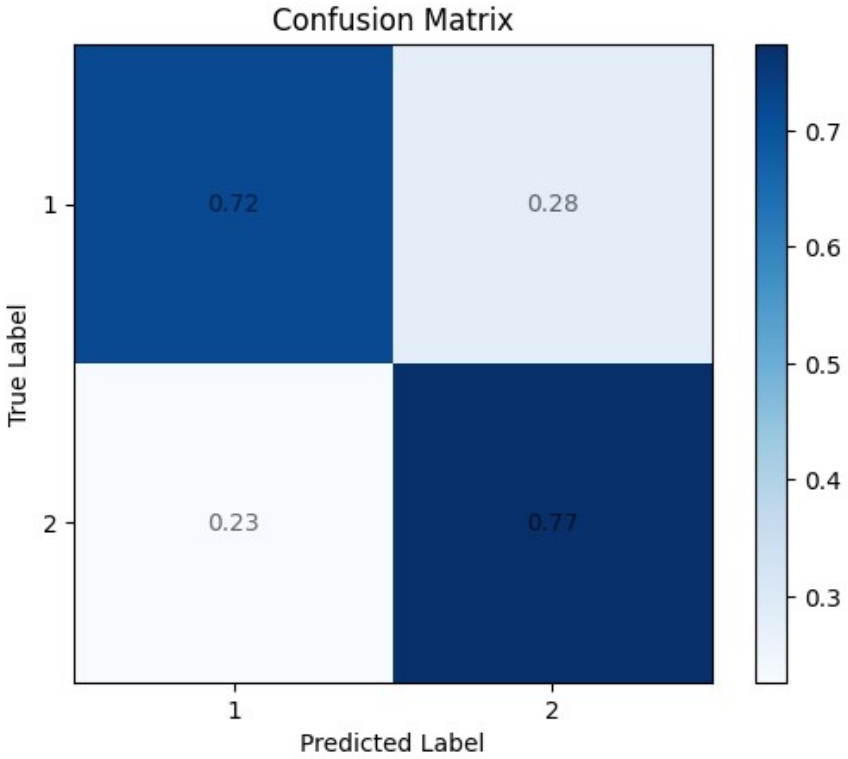


Figure 7: Confusion Matrix for Bayes Classifier for Binary Classification

- **The classifier has a moderate performance** with a classification accuracy of **0.7471** and an F1 score of **0.7621**.
- The **True Positive Rate (sensitivity)** is 0.72 and the **True Negative Rate (specificity)** is 0.77, indicating a good balance between identifying positive and negative instances.
- The **ROC curve** is significantly above the line of no discrimination, suggesting that the classifier is effective at distinguishing between the two classes.
- However, there's still room for improvement as indicated by the **False Positive Rate** of 0.23 and the **False Negative Rate** of 0.28.

The Confusion Matrix is given in Figure. 7
ROC Curve is shown in Figure. 8

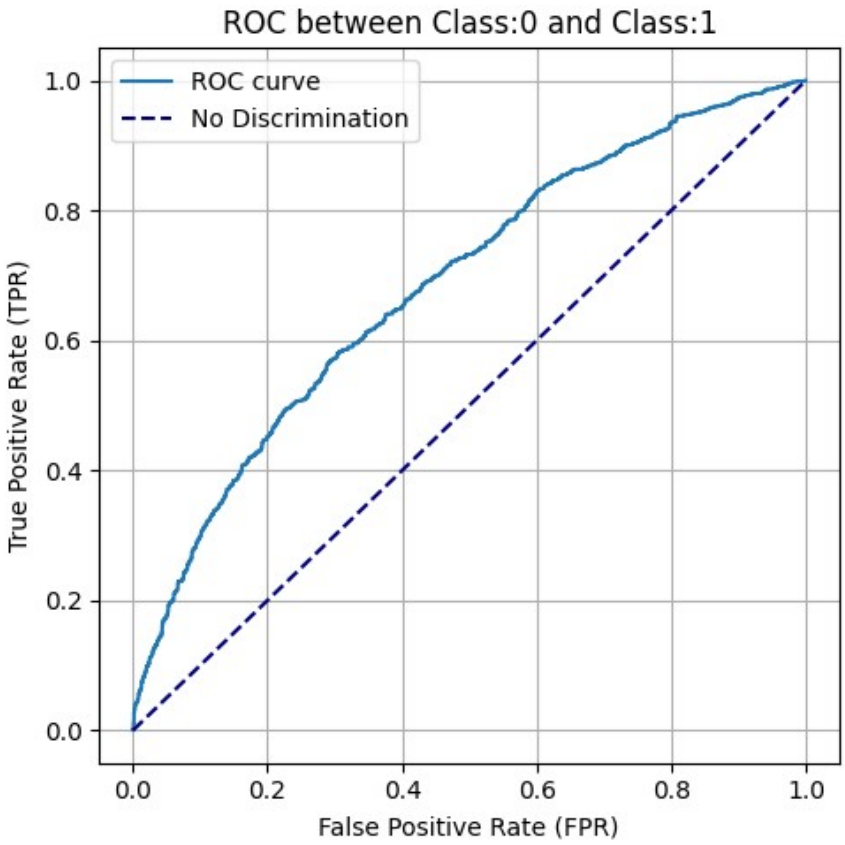


Figure 8: ROC Curve

Gaussian Mixture Models

In this method we assume that the Class Conditional Densities are a mixture of Gaussian Densities

$$P(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$
$$\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right)$$

where,

- $P(x)$: Probability density function (PDF) of the GMM.
- K: Number of Gaussian components in the mixture model.
- π_k: Mixing coefficient for the k-th Gaussian component, representing the prior probability of selecting that component.
- $N(x|\mu_k, \Sigma_k)$: Multivariate Gaussian distribution with mean μ_k and covariance matrix Σ_k. This term represents the likelihood of observing data point x given the parameters μ_k and Σ_k of the k-th Gaussian component.
- μ_k: Mean vector of the k-th Gaussian component, representing the center of the Gaussian distribution.
- Σ_k: Covariance matrix of the k-th Gaussian component, representing the shape and orientation of the Gaussian distribution.

After that the same Bayesian Decision Rule is implemented in which we compute the posterior based on the class-conditional densities and the priors to classify each datapoint. The Number of Gaussians is varied from K = 1 to 7 and the performance is listed in Table. 2

Num_of_Gaussians(K)	**Accuracy**	**F1 Score**
1	0.7482	0.7571
2	0.7596	0.7606
3	0.7725	0.7732
4	0.7889	0.7906
5	0.7839	0.7858
6	0.7882	0.7910
7	0.7904	0.7924

Table 2: Performance vs Number of Gaussian (K)

The variation of accuracy with number of gaussians in mixture.
Obervation - Increasing the number of Gaussians improved model performance (accuracy and F1-score) until 4 Gaussians. Beyond 4, the gains plateaued, suggesting 4 Gaussians may be a good balance between complexity and performance for this dataset.

The Confusion Matrices observed for K = 3 to 7 is given in Figure.10

ROC curve between two classes is given in Figure. 11

The marginal pdfs for Class 0, dim - 2 are given in the Figure.12

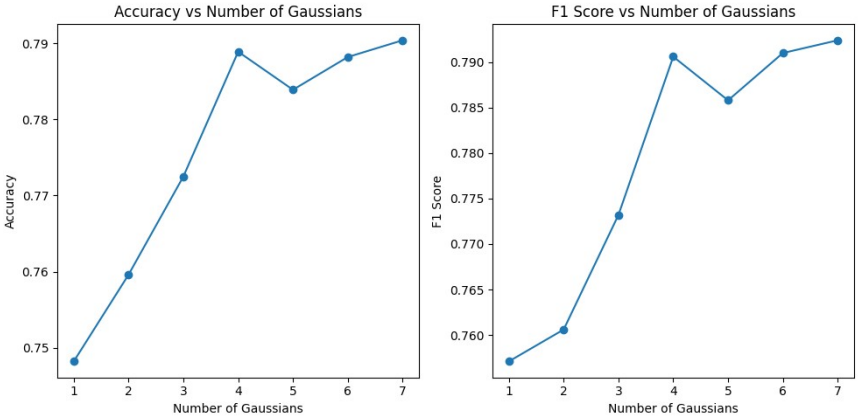
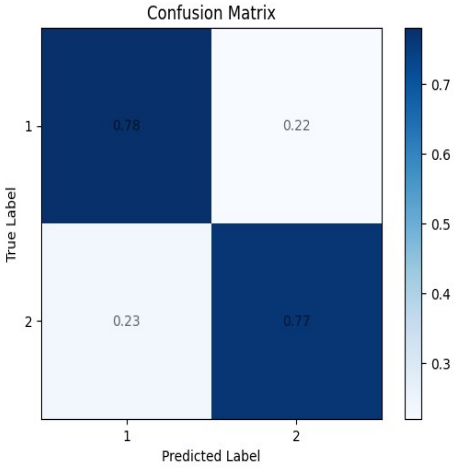
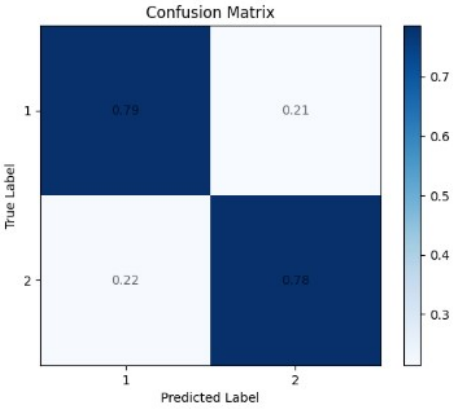


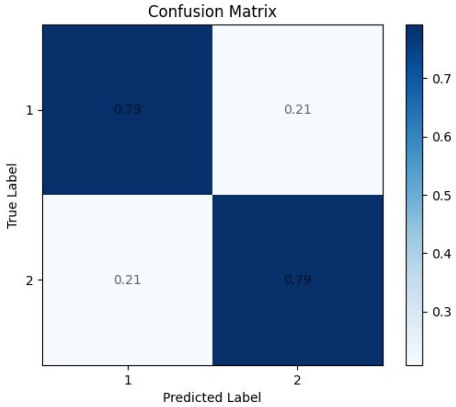
Figure 9: Accuracy and F1 Score vs Number of Gaussians



(a) K = 3

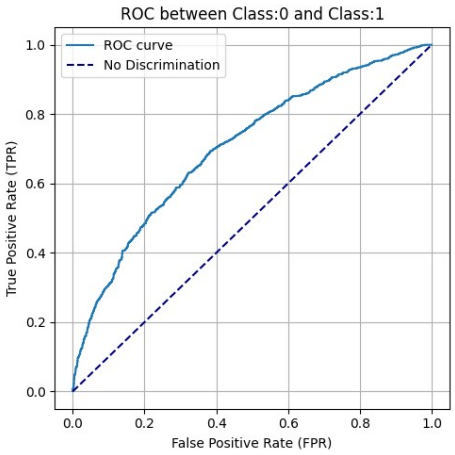


(b) K = 5

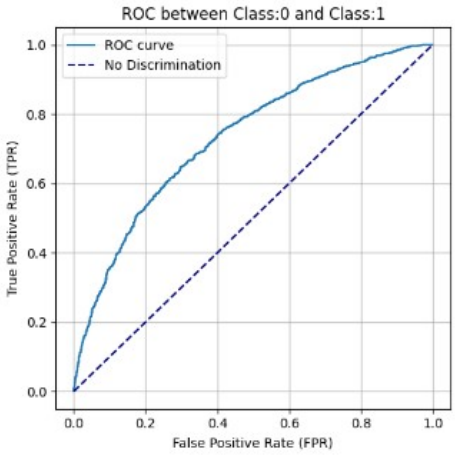


(c) K = 7

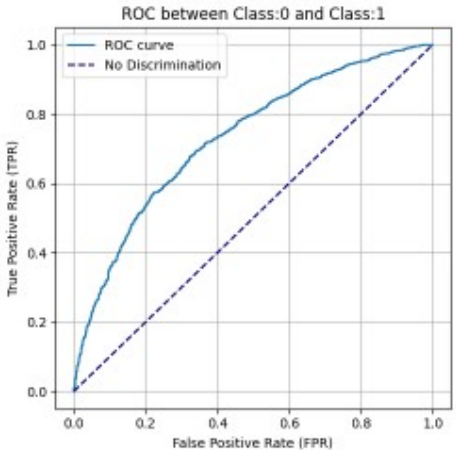
Figure 10: Confusion Matrices for GMM for Binary Classification



(a) K = 3

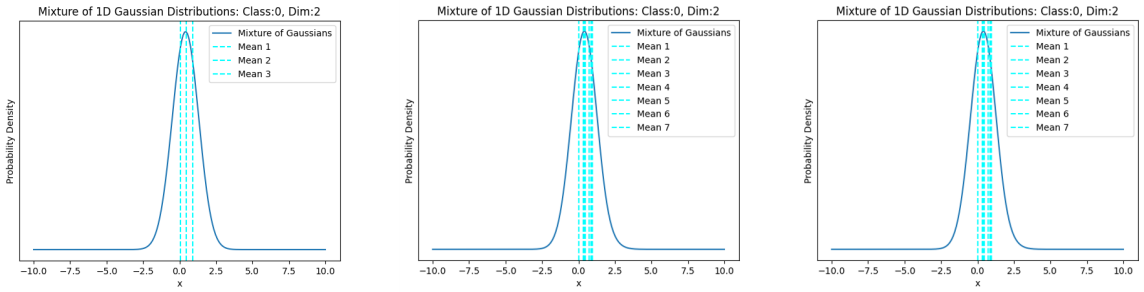


(b) K = 5



(c) K = 7

Figure 11: Confusion Matrices for GMM for Binary Classification



(a) $k = 3$l (b) $k = 5$ (c) $k = 7$

Figure 12: Marginal Distributions for Class 0 dimension = 2 with varying number of Gaussians

Parzen Window (Non-Parametric Density Estimation)

Window Size	Kernel	Accuracy	F-1 Score
7	Gaussian	0.716	0.695
7	Uniform	0.680	0.735
7	Epanechnikov	0.695	0.732
9	Gaussian	0.651	0.525
9	Uniform	0.647	0.670
9	Epanechnikov	0.710	0.720
11	Gaussian	0.541	0.180
11	Uniform	0.522	0.209
11	Epanechnikov	0.705	0.672

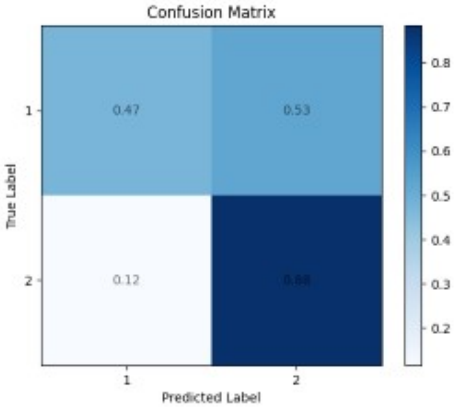
Observations:

- **Epanechnikov Kernel:** Resilient to window size changes, maintains a relatively high F1 score even as window size increases, suggesting good capture of underlying trends.
- **Gaussian Kernel:** Performance drops significantly with increasing window size, indicating possible overfitting to local patterns and noise.
- **Uniform Kernel:** F1 score decreases with larger windows, but less dramatically than the Gaussian kernel, indicating it's less prone to overfitting but may still capture some irrelevant data.

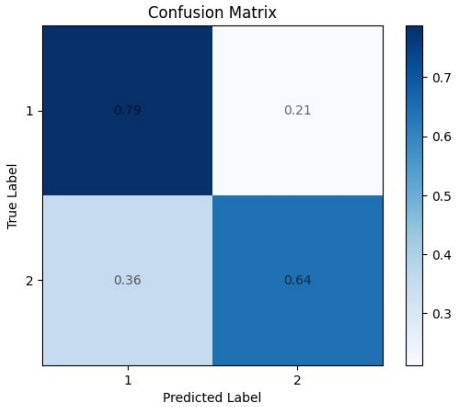
In general, when the window size increases for kernel-based methods:

- **Bias-Variance Trade-off:** Increasing the window size in kernel-based methods can lead to a trade-off between bias and variance. Smaller windows might overfit as they capture less data, while larger windows might underfit.
- **Smoothing Effect:** Larger windows average over more data points, leading to a smoother output. This can be beneficial when fine-grained details are not crucial.
- **Sensitivity to Local Patterns:** Smaller windows are more sensitive to local patterns, useful for capturing rapid changes or specific details. However, this can lead to overfitting if local patterns are not representative of the overall trend.

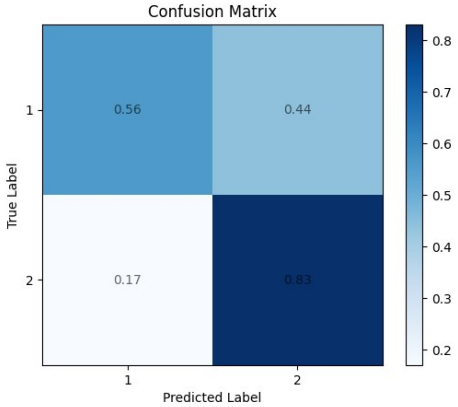
The Confusion Matrices and ROCs are given in Figure. 13, Figure. 14, Figure. 15 and Figure. 16



(a) Uniform Window

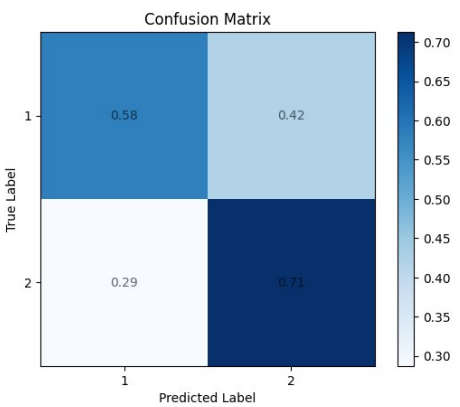


(b) Gaussian Window

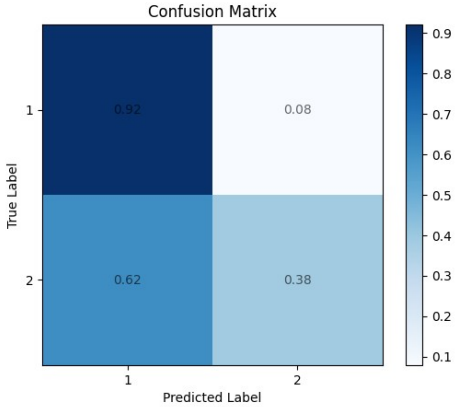


(c) Epanechnikov Window

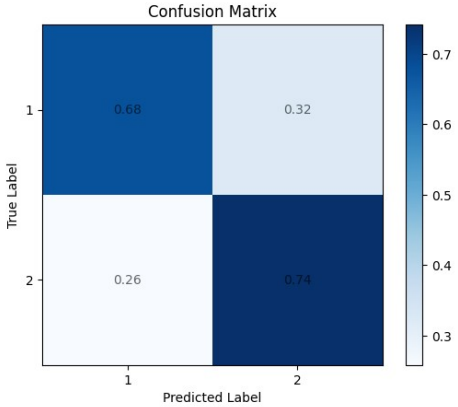
Figure 13: Confusion Matrices for Parzen Window (window size = 7) Binary Classification



(a) Uniform Window

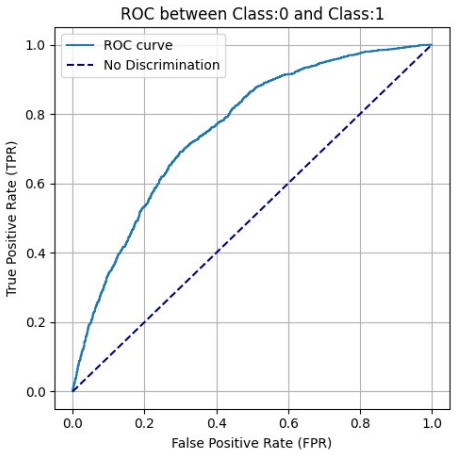


(b) Gaussian Window

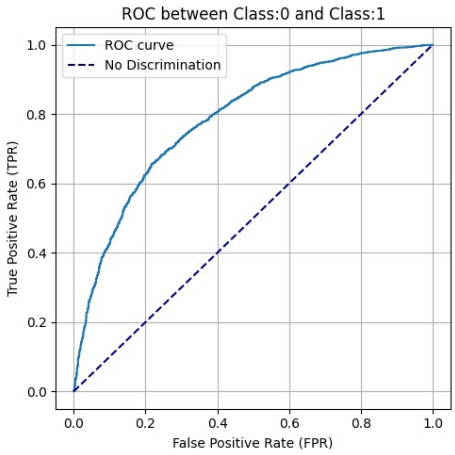


(c) Epanechnikov Window

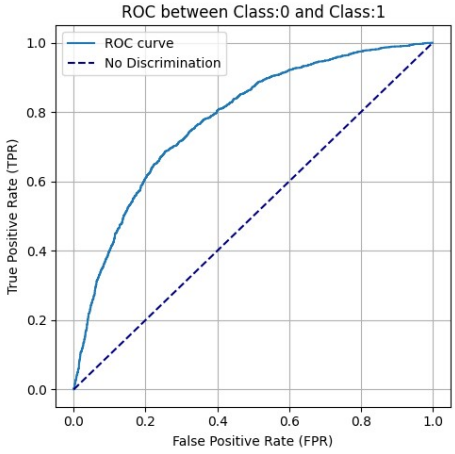
Figure 14: Confusion Matrices for Parzen Window (window size = 9) Binary Classification



(a) Uniform Window

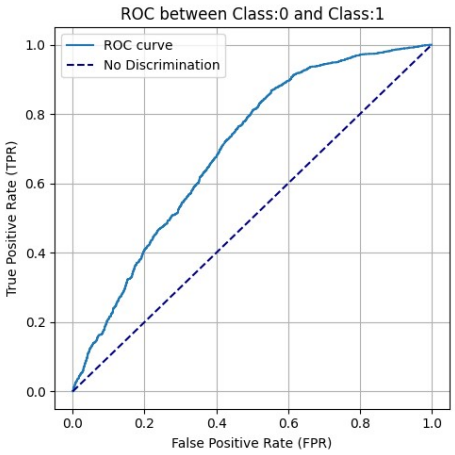


(b) Gaussian Window

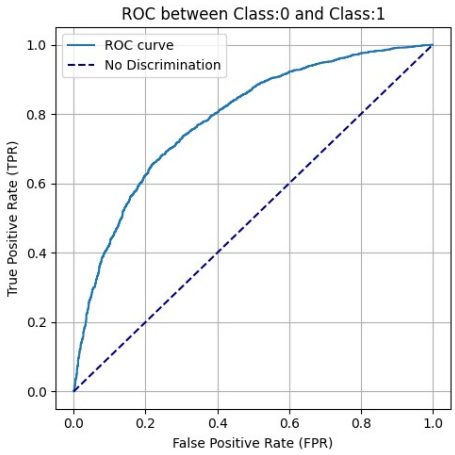


(c) Epanechnikov Window

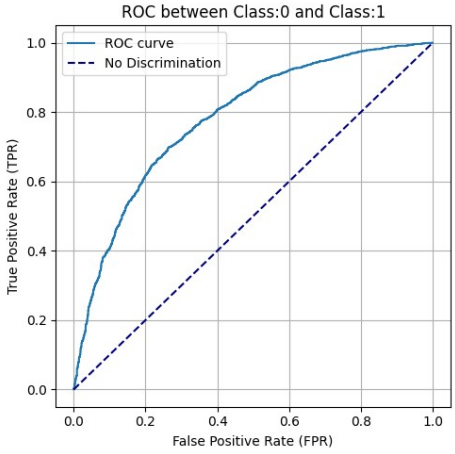
Figure 15: ROCss for Parzen Window (window size = 7) Binary Classification



(a) Uniform Window



(b) Gaussian Window



(c) Epanechnikov Window

Figure 16: ROCss for Parzen Window (window size = 9) Binary Classification

k-Nearest Neighbour

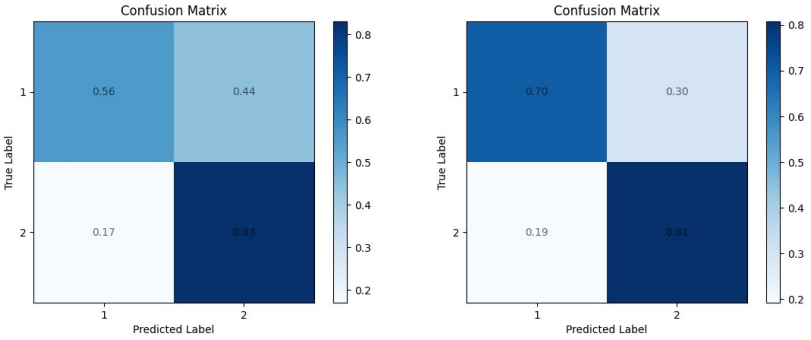
The Results of the KNN Classifier is given in Table.**??**

K	Accuracy (Euclidean)	Accuracy (Cosine)	F1 Score (Euclidean)	F1 Score (Cosine)
7	0.7357	0.7454	0.7498	0.7537
9	0.7450	0.7414	0.7596	0.7517
15	0.7532	0.7500	0.7679	0.7621
19	0.7536	0.7539	0.7686	0.7643

Observation:

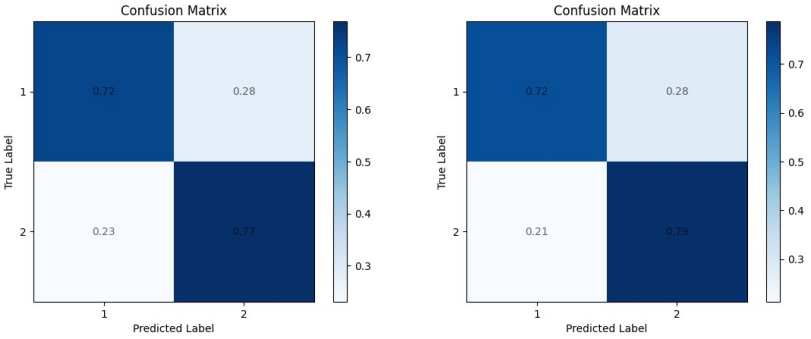
- Increasing K generally improves performance: As the value of K increases from 7 to 19, both the accuracy and F1 score generally increase for both distance metrics. This suggests that a larger K value, which considers more neighbors, may provide a more reliable classification in your case.
- Euclidean vs Cosine Distance: The choice between Euclidean and Cosine distance does not consistently favor one over the other across different K values. For K=7, the Cosine distance metric performs slightly better, while for K=9, the Euclidean distance metric is superior. For K=15 and K=19, the performance is almost identical. This indicates that the choice of distance metric may not significantly impact the performance of the KNN classifier for your specific dataset.
- Trade-off between K value and computational cost: While a larger K value may improve the model's performance, it also increases the computational cost as more distances need to be computed. Therefore, it's important to consider this trade-off when choosing the optimal K value.
- F1 Score vs Accuracy: The F1 score, which considers both precision and recall, is generally higher than the accuracy. This suggests that the classifier has a balanced performance in terms of both false positives and false negatives.

The Accuracy vs K is for both distance metrics is given in Figure 33
Some of the Confusion Matrices are given in Figure. 17 and Figure. 18
Some of the ROCs are given in Figure. 19 and Figure. 20



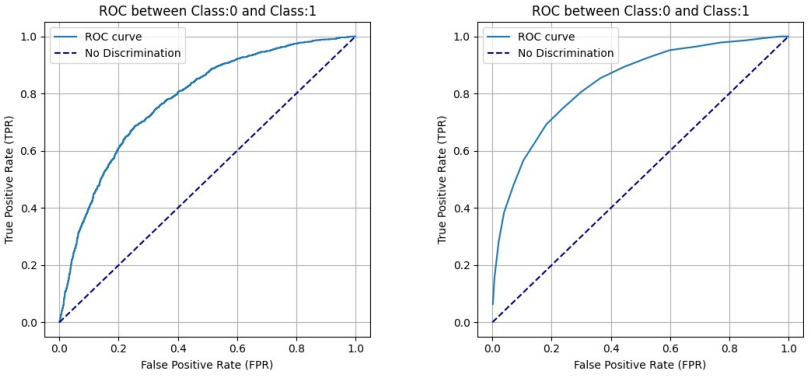
(a) K = 7 (b) K = 19

Figure 17: Confusion Matrices for Euclidean Distance



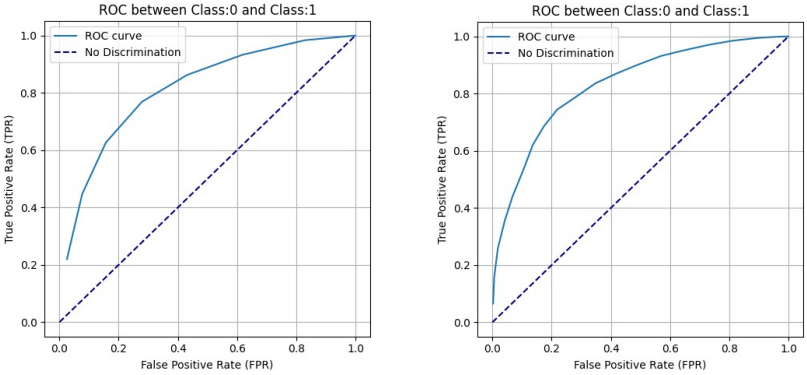
(a) K = 7 (b) K = 19

Figure 18: Confusion Matrices for Euclidean Distance



(a) K = 7 (b) K = 19

Figure 19: ROCs for Euclidean Distance



(a) K = 7 (b) K = 19

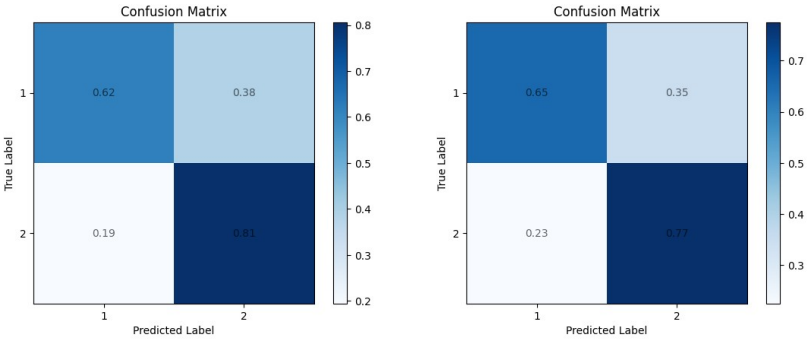
Figure 20: ROCs for Cosine Distance

Linear Classifier

Learning Rate	Empirical Risk (Train)	Empirical Risk (Test)	Accuracy	F-1 Score
1	0.5511	0.5593	0.7196	0.5198
0.1	0.5569	0.5624	0.7193	0.5246
0.01	0.5554	0.5558	0.7168	0.5463
0.001	0.3492	0.3472	0.7154	0.5612

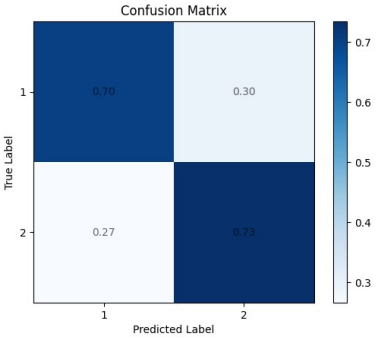
- **Learning Rate and Empirical Risk:** As the learning rate decreases from 1 to 0.001, both the empirical risk on the train dataset and the test dataset decrease. This suggests that a smaller learning rate might be helping the model to better learn from the data and generalize to unseen data.
- **Learning Rate and Accuracy:** The accuracy of the model slightly decreases as the learning rate decreases. This might suggest that while the model is better at minimizing the empirical risk with a smaller learning rate, it's not necessarily translating to better classification accuracy.
- **Learning Rate and F-1 Score:** The F-1 score, which is a measure of a test's accuracy and considers both the precision and the recall of the test, increases as the learning rate decreases. This suggests that the model's balance between precision and recall is improving with a smaller learning rate.
- **Best Learning Rate:** From the given data, a learning rate of 0.001 seems to provide the best balance between empirical risk and F-1 score. However, it's worth noting that the accuracy is slightly lower compared to higher learning rates.

The Confusion Matrices and ROCs for different values of learning rates are given in Figure. 21 and Figure. 22 respectively.

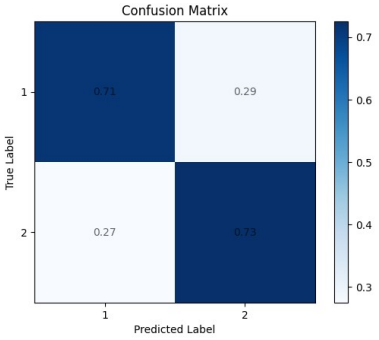


(a) learning rate = 0.001

(b) learning rate = 0.01

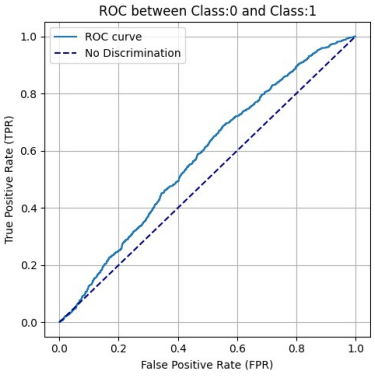


(c) learning rate = 0.1

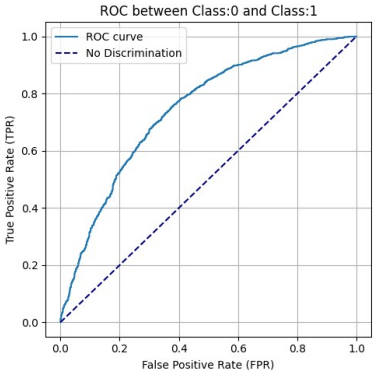


(d) learning rate = 1

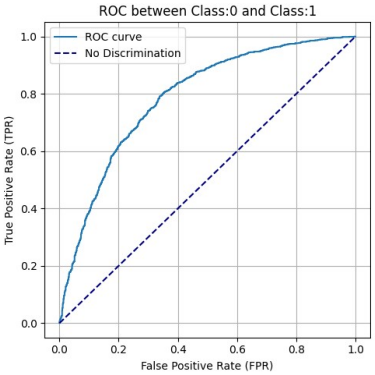
Figure 21: Confusion Matrices for Linear Classifier (Binary)



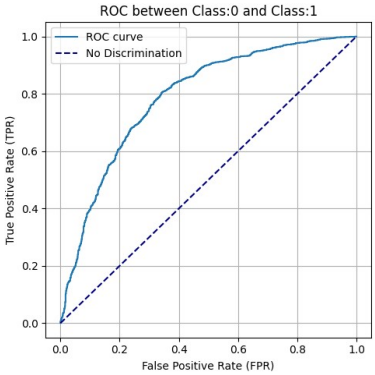
(a) learning rate = 0.001



(b) learning rate = 0.01



(c) learning rate = 0.1



(d) learning rate = 1

Figure 22: ROCs for Linear Classifier (Binary)

Q5. Multi Class Classification *(10 Classes)*

Bayes Classifier (Parametric Density Estimation - Normal)

In this method of classification, we assume the class conditional densities to be a Gaussian Density. We estimate the parameters of the density from given dataset and then we apply Bayesian Decision Rule for Classification

The Same Formulation from Binary Classification is followed

The Classification Accuracy observed by Bayes' classifier with 0-1 loss assuming Normal Distribution of Data is **0.4981** and F1 Score is **0.4950**

The Confusion Matrix is given in Figure23

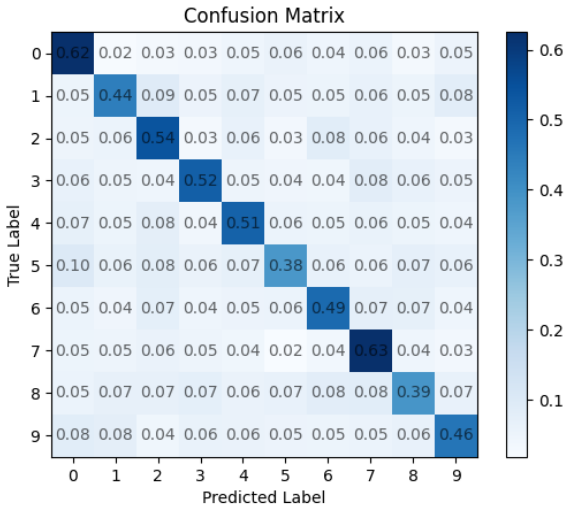
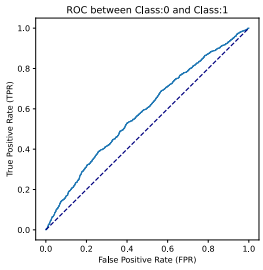
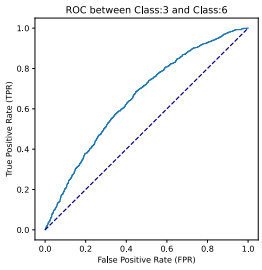


Figure 23: Confusion Matrix for Bayes Classifier for MultiClass

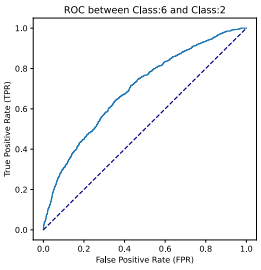
Some ROC Curves between pairs of Classes is given in Figure. 24



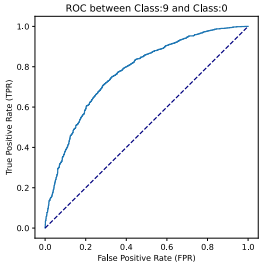
(a) Class 0 vs Class 1



(b) Class 3 vs Class 6



(c) Class 6 vs Class 2



(d) Class 9 vs Class 0

Figure 24: ROC Curves between different Classes for Bayes Classifier (MultiClass)

Gaussian Mixture Models

In this method we assume that the Class Conditional Densitites are a mixture of Gaussian Densities

$$P(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$
$$\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right)$$

where,

- $P(x)$: Probability density function (PDF) of the GMM.
- K: Number of Gaussian components in the mixture model.
- π_k: Mixing coefficient for the k-th Gaussian component, representing the prior probability of selecting that component.
- $N(x|\mu_k, \Sigma_k)$: Multivariate Gaussian distribution with mean μ_k and covariance matrix Σ_k. This term represents the likelihood of observing data point x given the parameters μ_k and Σ_k of the k-th Gaussian component.
- μ_k: Mean vector of the k-th Gaussian component, representing the center of the Gaussian distribution.
- Σ_k: Covariance matrix of the k-th Gaussian component, representing the shape and orientation of the Gaussian distribution.

The Number of Gaussians per mixture is varied from K = 1 to 9 and the performance is listed in Table.3

Table 3: Accuracy and F1 Score vs Number of Gaussians in a GMM

Number of Gaussians (K)	**Accuracy**	**F1 Score**
1	0.4981	0.4950
2	0.6083	0.6080
3	0.6834	0.6832
4	0.7396	0.7393
5	0.7966	0.7962
6	0.8220	0.8216
7	0.8428	0.8425
8	0.8507	0.8505
9	0.8421	0.8420

The variation of Accuracy with K is shown in Figure 25
The Confusion Matrices observed for K = 3 and K = 5 is given in Figure.26
ROC curve between two classes is given in Figure. 27
The marginal distributions have been given in Figure. 28

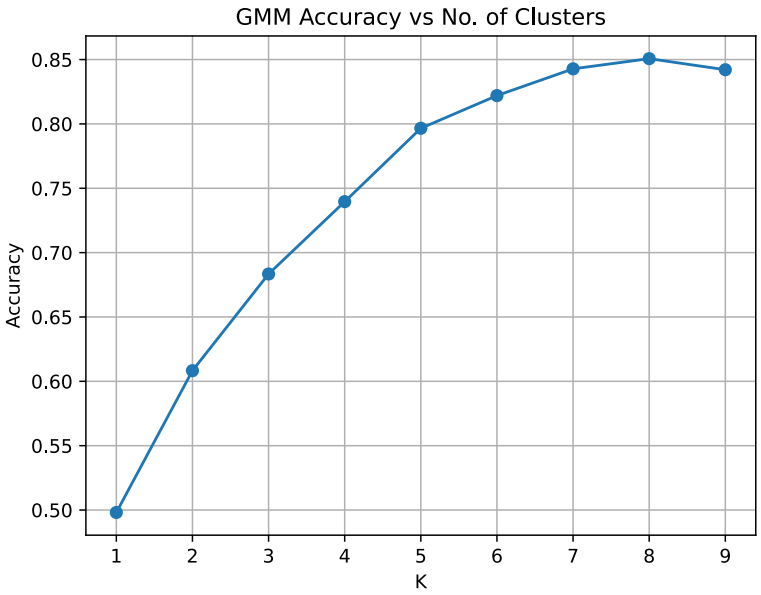
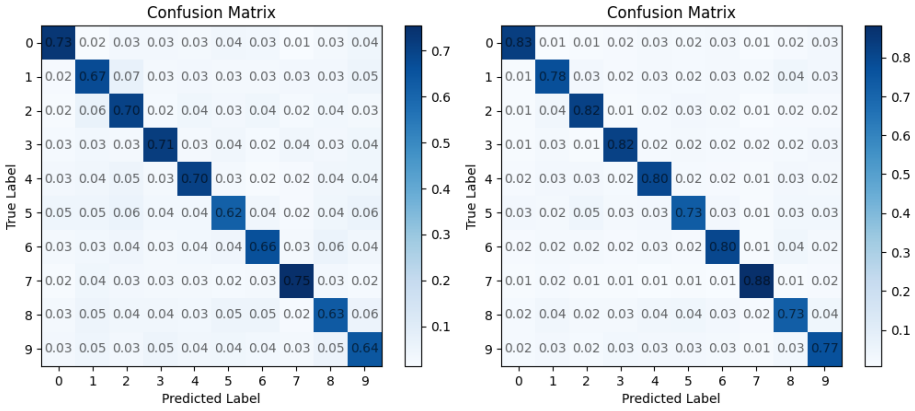


Figure 25: Accuracy vs No. of Clusters



(a) K = 3 (b) K = 5

Figure 26: Confusion Matrices for GMM

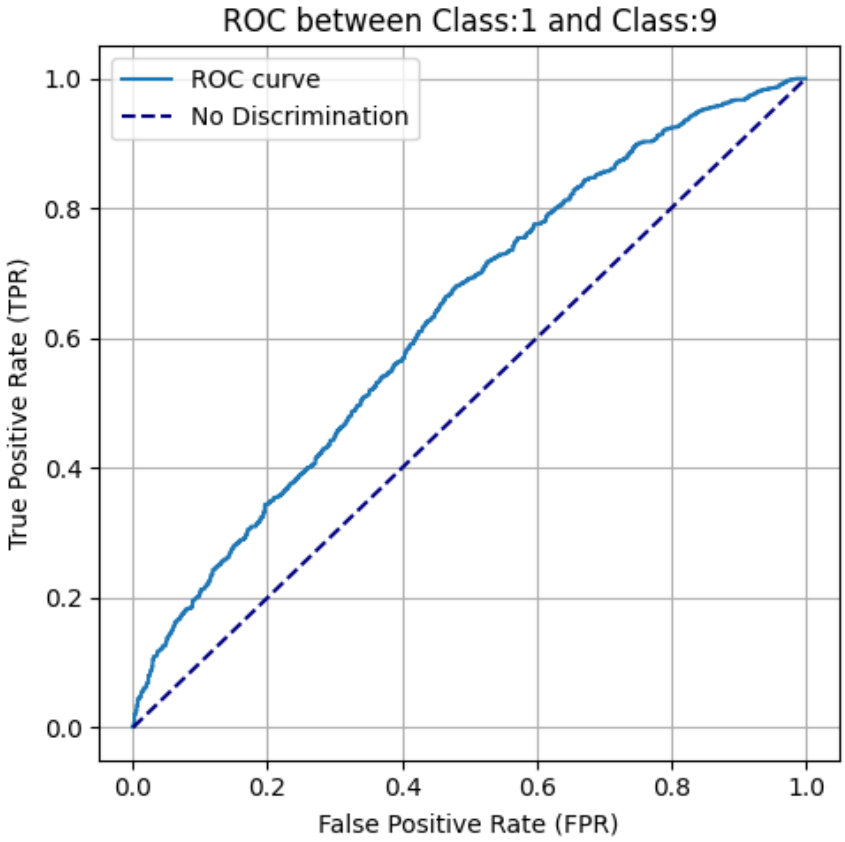
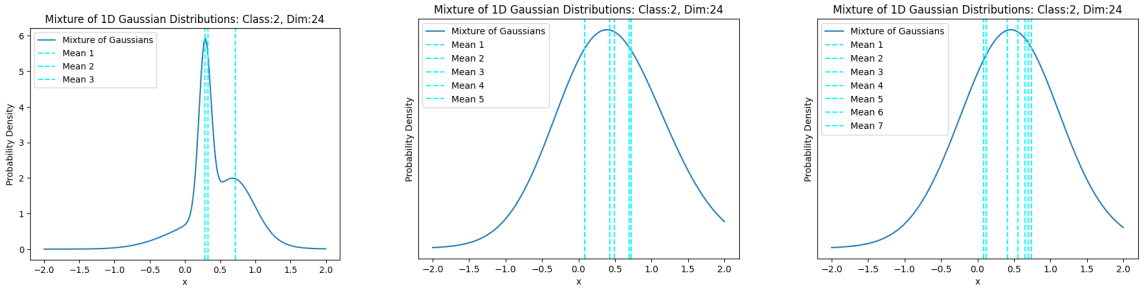


Figure 27: ROC between Class 1 and Class 9



(a) $k = 3$l (b) $k = 5$ (c) $k = 7$

Figure 28: Marginal Distributions for Class 2 dimension = 24 with varying number of Gaussians

Parzen Window (Non-Parametric Density Estimation)

Window Size	**Accuracies**		
	Gaussian Kernel	*Uniform Kernel*	*Epanechnikov Kernel*
0.05	0.1	0.098	0.098
0.1	0.374	0.098	0.098
0.2	0.374	0.098	0.098
0.5	0.382	0.098	0.098
0.8	0.502	0.098	0.098
5	0.224	0.22	0.31
10	0.19	0.244	0.228
20	0.104	0.104	0.124

Observations:

1. *Window Size and Accuracy Relationship*: The relationship between window size and accuracy is not straightforward and appears to differ between the kernel functions.
   - For Gaussian and Uniform kernels, the accuracy generally increases with increasing window size until a certain point (around 0.5 for Gaussian and 5 for Uniform). After that point, the accuracy either plateaus or slightly decreases.
   - Epanechnikov kernel exhibits the opposite trend, with accuracy starting relatively high and decreasing consistently as the window size increases.
2. *Performance Comparison*:
   - Gaussian kernel consistently achieves the highest accuracy across all window sizes, except for the last two (5 and 10), where Epanechnikov kernel has higher values.
   - Uniform kernel generally performs the poorest, having the lowest accuracy in most cases.

Possible Reasons :

- Parzen window estimation involves a trade-off between bias and variance. Smaller window sizes lead to lower bias (more flexibility to fit the data) but higher variance (more susceptible to noise). Larger windows reduce variance (smoother estimates) but can introduce bias (underfitting the data).
  - *Gaussian Kernel* is smooth and bell-shaped nature helps balance bias and variance, leading to generally good performance across a wider range of window sizes.
  - *Uniform Kernel* is rectangular in shape and offers less flexibility than Gaussian, potentially explaining its lower performance overall.
  - *Epanechnikov Kernel* offers a compromise between Gaussian and Uniform, potentially explaining its initial high accuracy (due to flexibility) followed by a decrease (due to bias) as the window size increases.
- The specific characteristics of the underlying data might influence the optimal window size and kernel selection. For example, data with high variability might benefit from larger window sizes to reduce variance, while smoother data might perform better with smaller windows due to the bias-variance trade-off.

A comparison among the different kernels have been shown in 29.

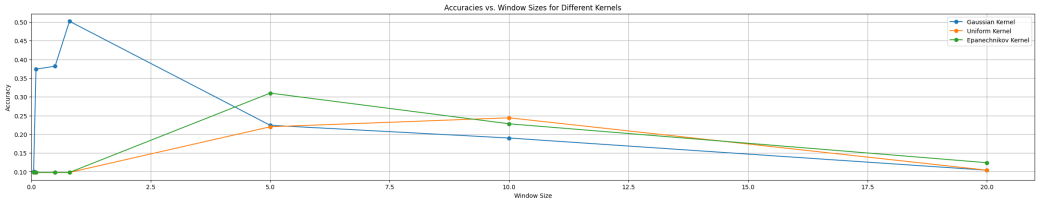
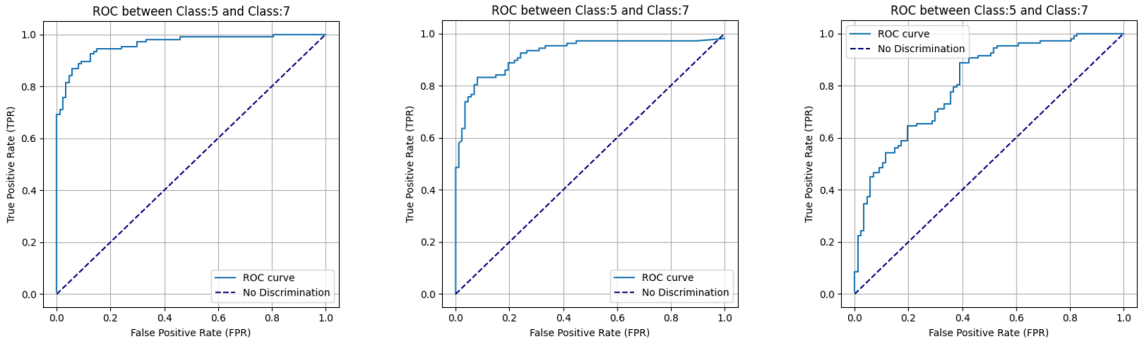


Figure 29: Comparison of the accuracies using Parzen Window algorithm for different kernels

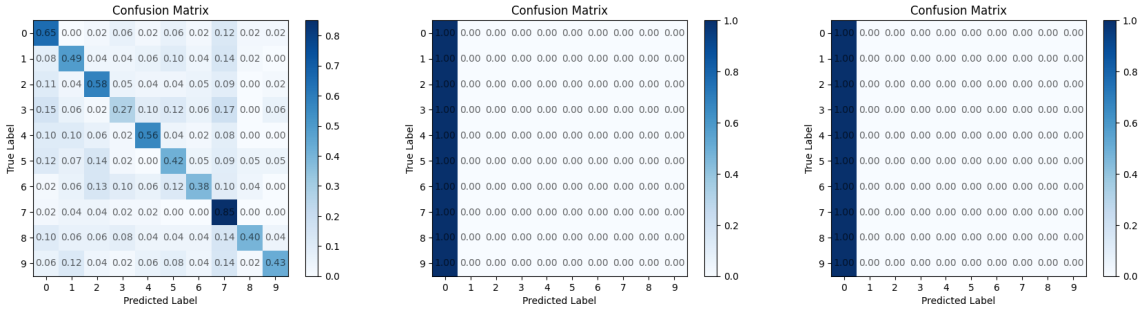
The ROCs for the Parzen Window algorithm are given in 30



(a) Gaussian Kernel, window = 0.8 (b) Uniform Kernel, window = 7 (c) Epanechnikov Kernel, window = 7

Figure 30: ROC for different kernels

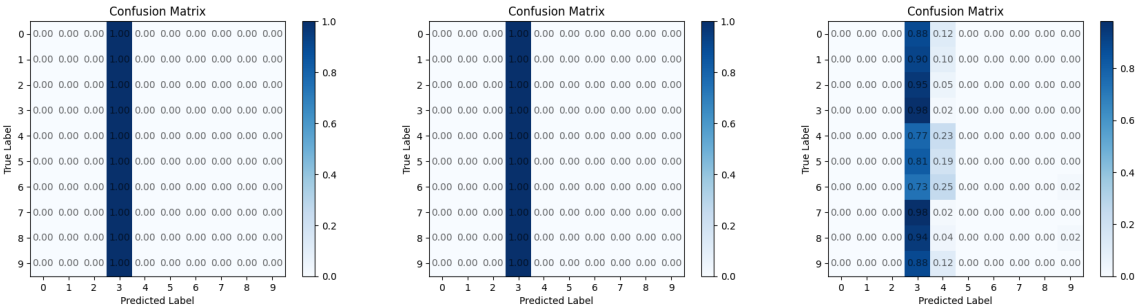
Confusion matrices for a window size of 0.8 have been shown in 31:



(a) Gaussian Kernel (b) Uniform Kernel (c) Epanechnikov Kernel

Figure 31: Confusion Matrices for window size = 0.8

Confusion matrices for a window size of 20 have been shown in 32:



(a) Gaussian Kernel (b) Uniform Kernel (c) Epanechnikov Kernel

Figure 32: Confusion Matrices for window size = 20

k-Nearest Neighbour

k values	Euclidian Distance		Cosine Distance	
	Classification Accuracy	*F1 Score*	*Classification Accuracy*	*F1 Score*
$k = 3$	0.3910	0.3783	0.4400	0.4297
$k = 5$	0.4360	0.4239	0.4700	0.4572
$k = 7$	0.4450	0.4341	0.5000	0.4927
$k = 10$	0.4750	0.4624	0.5000	0.4873
$k = 15$	0.4850	0.4724	0.4700	0.4551
$k = 20$	0.4990	0.4883	0.4900	0.4894
$k = 25$	0.5070	0.4941	0.4700	0.4590

The Accuracy vs K is for both distance metrics is given in Figure 33

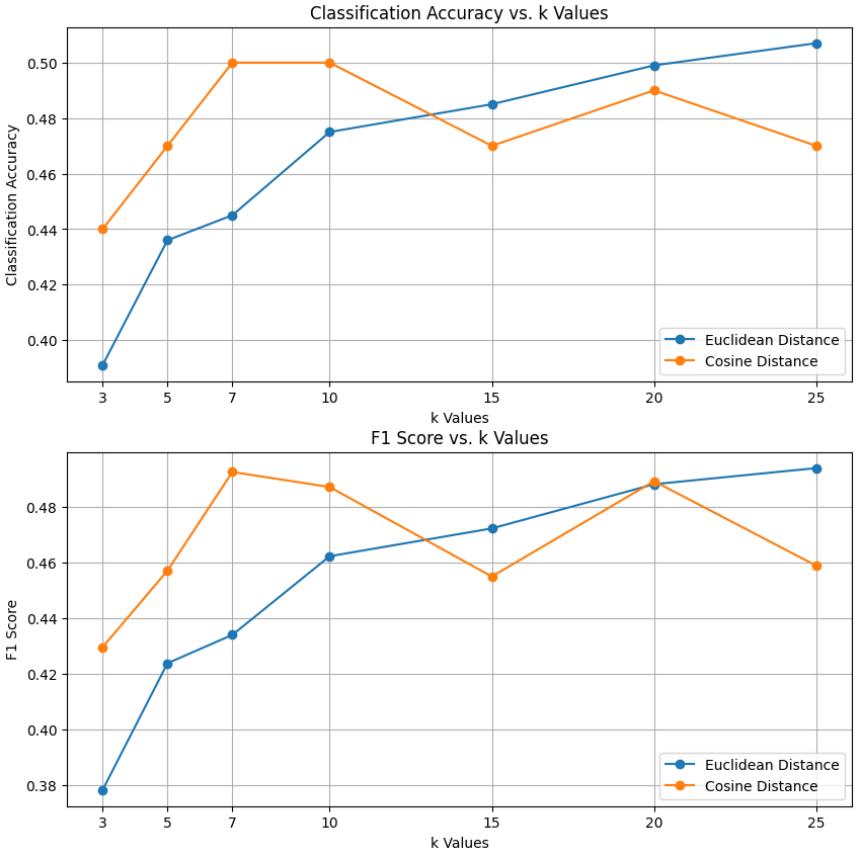


Figure 33: Accuracy and F1 Score vs K

Observations:

1. As the number of clusters (k values) increases, both classification accuracy and F1 score tend

to improve for both Euclidean and cosine distance metrics.

2. The performance improvement is more pronounced for the cosine distance metric compared to the Euclidean distance metric.

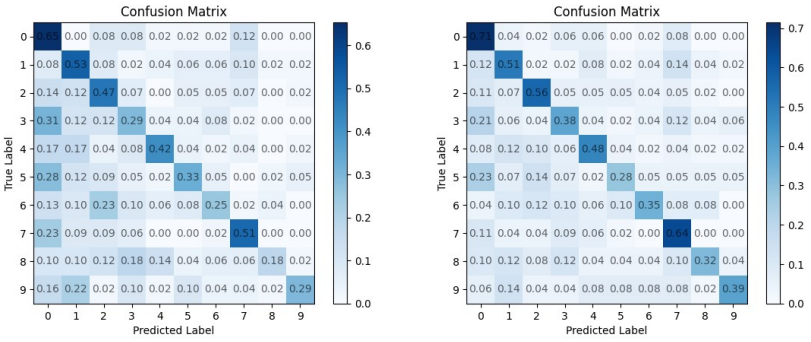
3. For the Euclidean distance metric, the classification accuracy ranges from 0.3910 (for $k = 3$) to 0.5070 (for $k = 15$), while the F1 score ranges from 0.3783 to 0.4941 for the same range of k values.

4. For the cosine distance metric, the classification accuracy ranges from 0.4400 (for $k = 3$) to 0.5000 (for $k = 10$ and $k = 15$), while the F1 score ranges from 0.4297 to 0.4927 for the same range of k values.

Possible Reasons:

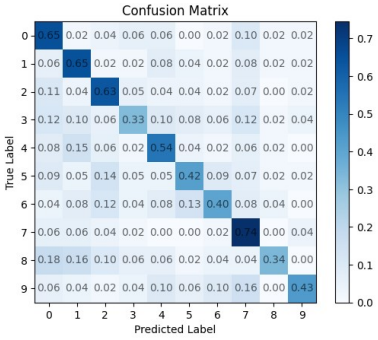
- Increasing k improves cluster granularity: With a higher number of clusters, the model can capture finer details in the data, leading to better discrimination between different classes.
- *Euclidean vs. cosine distance*: Cosine distance is particularly useful when the magnitude of vectors doesn't matter, only the direction. In some cases, cosine distance can better represent the similarity between data points, which might explain the more significant improvement in performance observed with cosine distance.
- *Diminishing returns*: While increasing the number of clusters generally improves performance, there's a point where further increasing k might not significantly enhance performance. This is evident in the fluctuation of performance metrics for certain k values (e.g., $k = 15$).
- *Data characteristics*: The effectiveness of clustering algorithms can be influenced by the characteristics of the dataset, such as its dimensionality, distribution, and separability. The observed trends in performance could be influenced by these factors.

Some of the Confusion Matrices and ROCs are given in Figure. 34, Figure. 35 ,Figure. 36 and Figure. 37 respectively.



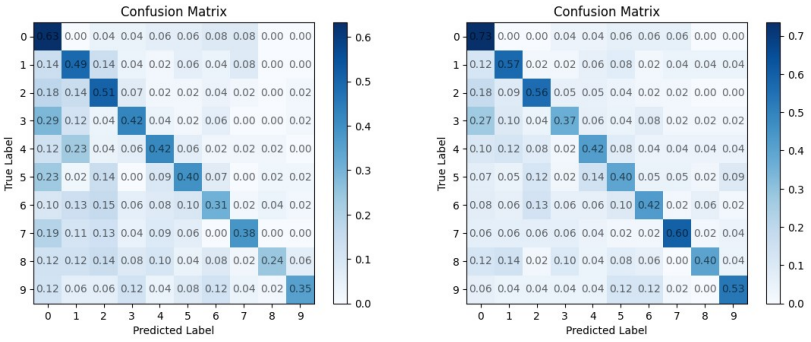
(a) K = 3

(b) K = 10



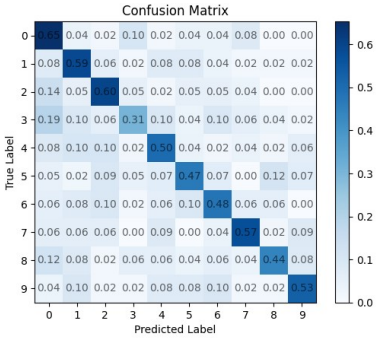
(c) K = 25

Figure 34: Confusion Matrices for Euclidean Distance



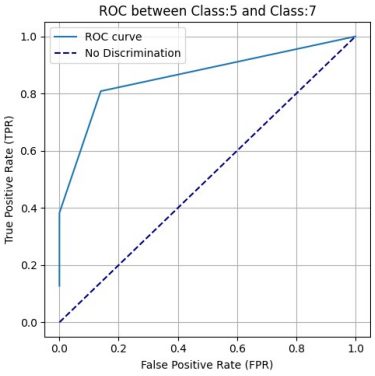
(a) K = 3

(b) K = 10

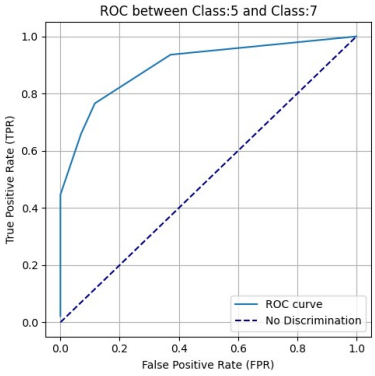


(c) K = 25

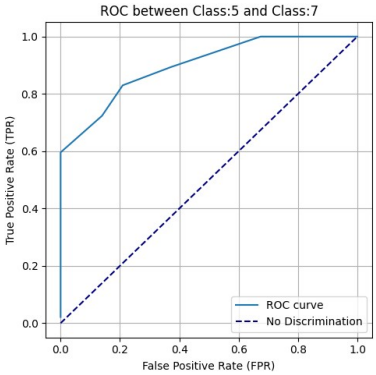
Figure 35: Confusion Matrices for KNN (Cosine Distance) MultiClass



(a) K = 3

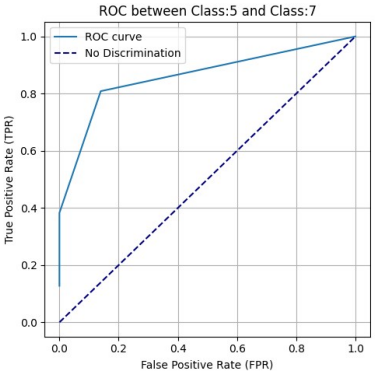


(b) K = 10

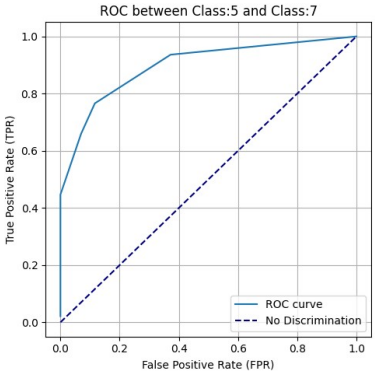


(c) K = 25

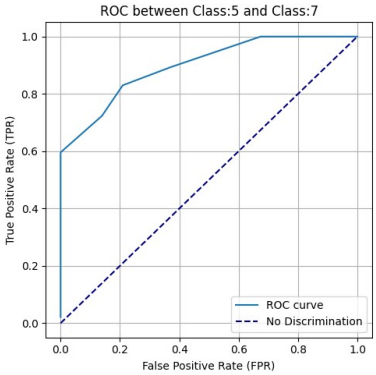
Figure 36: ROCs for KNN (Euclidean Distance) MultiClass



(a) K = 3



(b) K = 10



(c) K = 25

Figure 37: ROCs for KNN (Euclidean Distance) MultiClass

Linear Classifier

A Linear Classifier is fit on the given dataset and the results observed on Testing are given in Table 4

Table 4: Performance Metrics of Linear Classifier (One vs Rest Approach)

Learning Rate	**Accuracy**	**F1 Score**	**Empirical Risk(Test)**
0.001	0.262	0.240	0.578
0.01	0.285	0.274	2.022
0.1	0.296	0.291	2.000
1	0.303	0.298	1.988

Empirical Risk on Training

Learning Rate	**Empirical Risk (Train)**
0.001	0.578
0.01	2.020
0.1	1.993
1	1.983

Observations:

- *Effect of Learning Rate on Accuracy and F1 Score*: Both accuracy and F1 score generally increase with higher learning rates, indicating that a higher learning rate helps the model achieve better performance on the test data.
- *Effect of Learning Rate on Empirical Loss*: The empirical loss on both training and testing data initially decreases with higher learning rates up to a certain point, after which it starts to increase. This suggests that there's an optimal learning rate that balances between underfitting and overfitting.
- *Comparison between Training and Testing Loss*: The empirical loss on the training data is similar to that on the testing data, indicating that the model is not significantly overfitting or underfitting the training data.
- *Overall Performance*: The model achieves relatively low accuracy and F1 score, indicating that it may struggle with effectively discriminating between the classes.

The confusion matrix using the best learning rate is given in Figure 38
ROC between two classes is given in Figure 39

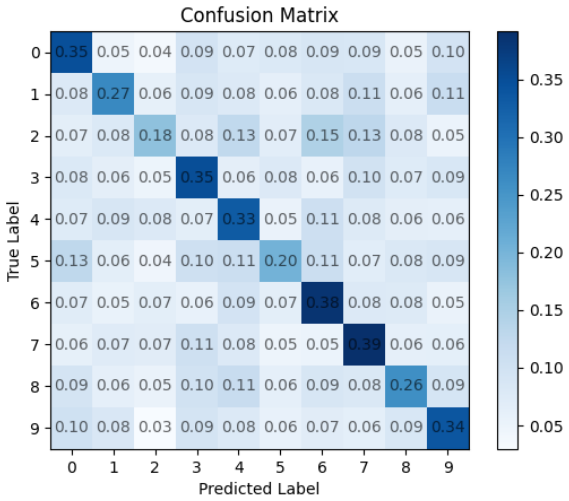


Figure 38: Confusion Matrix for Linear Classifier (O vs R)

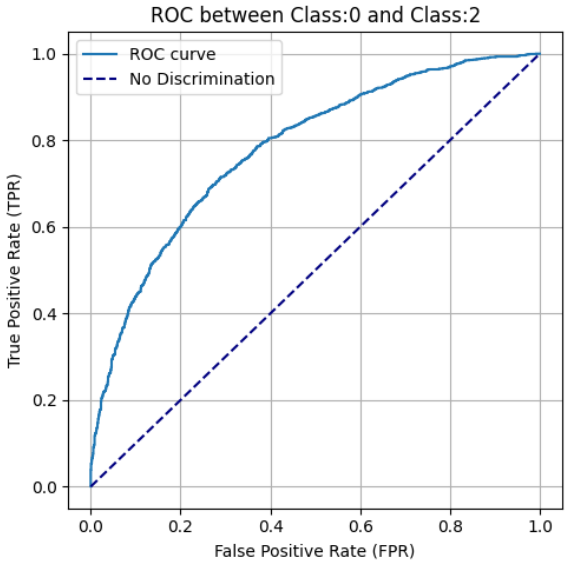


Figure 39: ROC curve for Linear Classifier