HW 2 Solutions

Author:
Fateme Noorzad

StudentID:
810198271

Machine Learning
Fall 2020

Contents

Chapter 1

Question 1

In order to determine the decision boundary between the two classes, on the grounds of the fact that these two have a Guassian distribution, their covariance matrix and expected values needs to be calcaulated. Based on the fact that we are representing each data in a two dimentional environment:

$$\underline{X} = [x \;\; y]^T$$

Now in order to calculate the expected value we have:
*For red class:

$$\begin{gathered}\underline{\mu}_{red} \;=\; [\mu_{red_1} \;\; \mu_{red_2}]^T \;=\; [E\{x_{red_1}\} \;\; E\{x_{red_2}\}]^T \\ \rightarrow \begin{cases} \mu_{red_1} \;\approx\; \frac{1}{n_1}\sum_{i=1}^{n_1} x_{red_{1,i}} \;=\frac{1}{6}[1*2 \;+\; 2*3 \;+\; 4] = 2 \\ \mu_{red_2} \;\approx\; \frac{1}{n_1}\sum_{i=1}^{n_1} x_{red_{2,i}} \;=\frac{1}{6}[1*2 \;+\; 2*2 \;+\; 3*2] = 2 \end{cases} \\ \rightarrow \underline{\mu}_{red} \;=\; [2 \;\; 2]^T\end{gathered}$$

*For blue class:

$$\begin{gathered}\underline{\mu}_{blue} \;=\; [\mu_{blue_1} \;\; \mu_{blue_2}]^T \;=\; [E\{x_{blue_1}\} \;\; E\{x_{blue_2}\}]^T \\ \rightarrow \begin{cases} \mu_{blue_1} \;\approx\; \frac{1}{n_2}\sum_{i=1}^{n_2} x_{blue_{1,i}} \;=\frac{1}{5}[(-1)*2 \;+\; (-2) \;+\; (-3)*2] = -2 \\ \mu_{blue_2} \;\approx\; \frac{1}{n_2}\sum_{i=1}^{n_2} x_{blue_{2,i}} \;=\frac{1}{5}[1 \;+\; (-1)*2 \;+\; (-2)*2] = -1 \end{cases} \\ \rightarrow \underline{\mu}_{blue} \;=\; [-2 \;\; -1]^T\end{gathered}$$

In order to calculate covariance matrix for each of these classes we know that covariance matrix is calculated as:

$$Cov(\underline{X}) \;=\; \frac{1}{n-1}\sum_{i=1}^{n}(x_i \;-\; \mu)^2$$

By employing numpy covarinace matrix calculator(the code can be found under the name "Q1.py" in the uploaded file) the resulting matrix for both two classes are:
*For red class:

$$\Sigma_{red} \;=\; \begin{bmatrix} 1.2 & 0.6 \\ 0.6 & 0.8 \end{bmatrix}$$

*For blue class:

$$\Sigma_{blue} \;=\; \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1.5 \end{bmatrix}$$

On the grounds of the fact that covariance matrix of these two features are arbitraty, case 3 occurs. Meaning the below formulation holds for discriminative function of each feature:

$$g_i(\underline{X}) \;=\; \underline{X}^T W_i \underline{X} \;+\; \underline{w}_i^T \underline{X} \;+\; w_{i_0}$$

where:

$$\begin{aligned} W_i \;&=\; -\frac{1}{2}\Sigma_i^{-1} \\ \underline{w}_i \;&=\; \Sigma_i^{-1}\underline{\mu}_i \\ w_{i_0} \;&=\; -\frac{1}{2}\underline{\mu}_i^T \Sigma_i^{-1} \underline{\mu}_i \;-\; \frac{1}{2} ln\{\Sigma_i\} \;+\; ln\{p(w_i)\} \end{aligned}$$

By employing the above calculated values in the equations, as well as considering the fact that $p(w_{red}) = \frac{6}{11}$ for red color, we get: (the calculations are done by python and can be found in "Q1.py" in the uploaded file)

$$g_{red}(\underline{X}) \;=\; \underline{X}^T W_{red} \underline{X} \;+\; \underline{w}_{red}^T \underline{X} \;+\; w_{red_0}$$

where:

$$\begin{aligned} W_{red} \;&=\; -\frac{1}{2}\Sigma_{red}^{-1} = \begin{bmatrix} -0.667 & 0.5 \\ 0.5 & -1 \end{bmatrix} \\ \underline{w}_{red} \;&=\; \Sigma_{red}^{-1}\underline{\mu}_{red} = \begin{bmatrix} 0.667 \\ 2 \end{bmatrix} \\ w_{red_0} \;&=\; -\frac{1}{2}\underline{\mu}_{red}^T \Sigma_{red}^{-1} \underline{\mu}_{red} \;-\; \frac{1}{2} ln\{\Sigma_{red}\} \;+\; ln\{p(w_{red})\} = -3.017 \\ \rightarrow g_{red}(\underline{X}) \;&=\; \underline{X}^T \begin{bmatrix} -0.667 & 0.5 \\ 0.5 & -1 \end{bmatrix} \underline{X} \;+\; \begin{bmatrix} 0.667 & 2 \end{bmatrix} \underline{X} \;-\; 3.017 \\ &=\; -0.667x^2 + 0.667x + xy - y^2 + 2y - 3.017 \end{aligned}$$

Carry out the same procedure for color blue while $p(w_{red}) = \frac{5}{11}$, we get:

$$g_{blue}(\underline{X}) \;=\; \underline{X}^T W_{blue} \underline{X} \;+\; \underline{w}_{blue}^T \underline{X} \;+\; w_{blue_0}$$

where:

$$\begin{aligned} W_{blue} \;&=\; -\frac{1}{2}\Sigma_{blue}^{-1} = \begin{bmatrix} -0.6 & -0.2 \\ -0.2 & -0.4 \end{bmatrix} \\ \underline{w}_{blue} \;&=\; \Sigma_{blue}^{-1}\underline{\mu}_{blue} = \begin{bmatrix} -2.8 \\ -1.6 \end{bmatrix} \\ w_{blue_0} \;&=\; -\frac{1}{2}\underline{\mu}_{blue}^T \Sigma_{blue}^{-1} \underline{\mu}_{blue} \;-\; \frac{1}{2} ln\{\Sigma_{blue}\} \;+\; ln\{p(w_{blue})\} = -4.5 \\ \rightarrow g_{blue}(\underline{X}) \;&=\; \underline{X}^T \begin{bmatrix} -0.6 & -0.2 \\ -0.2 & -0.4 \end{bmatrix} \underline{X} \;+\; \begin{bmatrix} -2.8 & -1.6 \end{bmatrix} \underline{X} \;-\; 4.5 \\ &=\; -0.6x^2 - 2.8x - 0.4xy - 1.6y - 0.4y^2 - 4.5 \end{aligned}$$

Now, to determine the line representing the decision boundary we are required to solve the below equation for $\underline{X}$.

$$\begin{gathered} g_{blue}(\underline{X}) \;=\; g_{blue}(\underline{X}) \\ \rightarrow -0.067x^2 \;+\; 3.467x \;+\; 1.4xy \;+\; 3.6y \;-\; 0.6y^2 \;=\; 1.483 \end{gathered}$$

The plot of the decison boundary as well as the classes is as below:

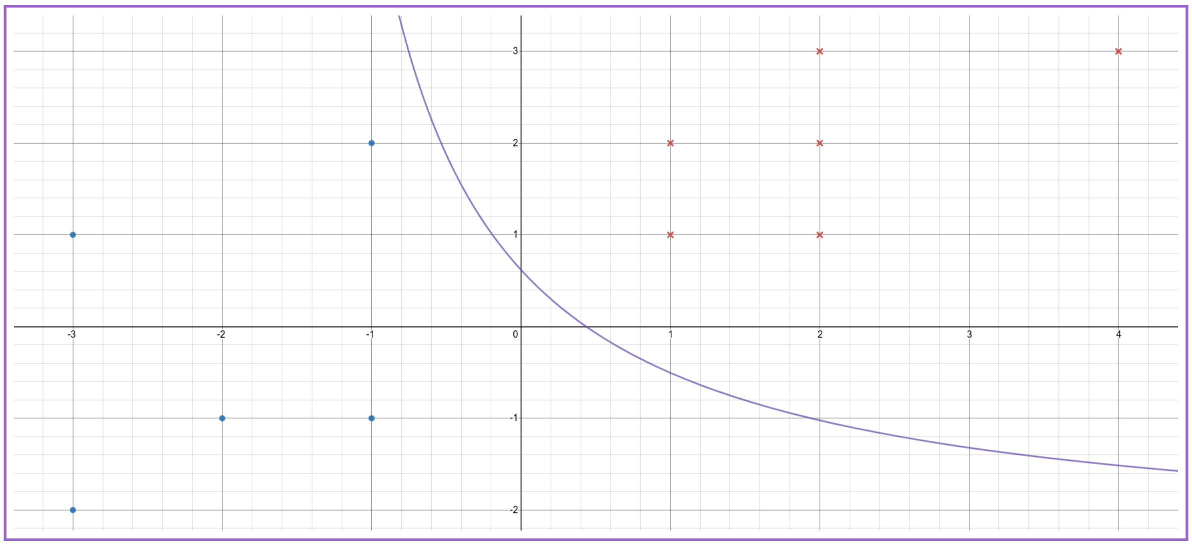


FIGURE 1.1: Two classes and decision boundary

Chapter 2

Question 2

2.1 Part A

Generally, the probability of a random variable with a continuous distribution having the same value as a vetor is zero. However, the value of the PDF of a random varialbe in a certain point can be evaluated.
General multivariate normal density function in d dimentions can be represented as below:

$$p(\underline{X}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} exp\{-\frac{1}{2}(\underline{X}-\underline{\mu})^T\Sigma^{-1}(\underline{X}-\underline{\mu})\}$$

Therefore, in order to calculated the value of the given point using Numpy we get: (Its code can be found under the name "Q2.np")

$$\Sigma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 8 & 2 \\ 0 & 2 & 5 \end{bmatrix} \rightarrow |\Sigma| = 36,\ \Sigma^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.1389 & -0.0556 \\ 0 & -0.0556 & 0.2222 \end{bmatrix}$$
$$\rightarrow f([2\ \ 0.5\ \ 3]^T) = 0.00297969$$

2.2 Part B

Whitening transformation can be done using the below transformation matriz:

$$A_w = \Phi\Lambda^{\frac{-1}{2}}$$

where:

Φ : is the matrix whose columns are the orthogonal eigenvectors of Σ.

Λ : is the diagonal matrix of corrresponding eigenvalue of the stated vectors.

Therefore, in order to determine A_w we need to calculate the eigenvalues and eigenvectors of Σ. To do so, using the command "eig" in "linalg" under "numpy", helps determine both of these requirements. The resulting values and vectors are: (Its code can be found under the name "Q2.np")

Eigenvalues are: $\lambda_1 = 4,\ \lambda_2 = 9,\ \lambda_3 = 1 \rightarrow \Lambda = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Eigenvectors are (corresponding to the computed eigenvalues):

$$\underline{\varphi}_1 = \begin{bmatrix} 0 \\ 0.4472 \\ -0.8944 \end{bmatrix},\ \underline{\varphi}_2 = \begin{bmatrix} 0 \\ -0.8944 \\ -0.4472 \end{bmatrix},\ \underline{\varphi}_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\rightarrow \Phi = \begin{bmatrix} 0 & 0 & 1 \\ 0.4472 & -0.8944 & 0 \\ -0.8944 & -0.4472 & 0 \end{bmatrix}$$

$$\rightarrow A_w = \begin{bmatrix} 0 & 0 & 1 \\ 0.2236 & -0.2981 & 0 \\ -0.4472 & -0.1491 & 0 \end{bmatrix}$$

2.3 Part C

In general, linear transformations have an effect as conveyed below:

$$if\ \underline{X} \sim N(\underline{\mu}, \Sigma),\ \underline{Y} = A^T \underline{X} \rightarrow \underline{Y} \sim N(A^T \underline{\mu}, A^T \Sigma A)$$

In the case where A is a whitening transformation as computed in last section, $\underline{Y}$ will be:

$$\underline{Y} \sim N(A_w^T \underline{\mu}, I)$$

In order to reach the goal this problem is seeking through, all we need to do is to transform $\underline{X}$ into a zero-mean random variable. Therefore, the suitable transformation is:

$$\underline{Y} = A^T(\underline{X} - \underline{\mu}) = A^T \underline{X} - A^T \underline{\mu}$$

As it is shown below, now $\underline{Y}$ is a standard normal random variable:

$$\begin{aligned} E\{\underline{Y}\} &= E\{A^T \underline{X} - A^T \underline{\mu}\} = A^T E\{\underline{X}\} - A^T \underline{\mu} = A^T \underline{\mu} - A^T \underline{\mu} = 0 \\ Cov\{\underline{Y}\} &= E\{(\underline{Y} - E\{\underline{Y}\})(\underline{Y} - E\{\underline{Y}\})^T\} = E\{(A^T \underline{X} - A^T \underline{\mu})(A^T \underline{X} - A^T \underline{\mu})^T\} \\ &= A^T\ E\{\underline{X}\underline{X}^T\} A = A^T \Sigma A = \Lambda^{\frac{-1}{2}} \Phi^T \Phi \Lambda \Phi^{-1} \Phi \Lambda^{\frac{-1}{2}} = I \end{aligned}$$

where we used the fact that $\Phi^{-1} = \Phi^T$.

Now that we are sure the given transformation is suitable for us to achieve our goal, the given $\underline{x}$ vector is tranformed into:

$$\underline{y} = \begin{bmatrix} 0 & 0 & 1 \\ 0.2236 & -0.2981 & 0 \\ -0.4472 & -0.1491 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 0.5 \\ 3 \end{bmatrix} = \begin{bmatrix} -1.23 \\ 0.149 \\ 1 \end{bmatrix}$$

2.4 Part D

The Mahalanobis distance from $\underline{X}$ to $\underline{\mu}$ can be calculated as below: (It is calculated in the same file)

$$r^2 = (\underline{X} - \underline{\mu})^T \Sigma^{-1} (\underline{X} - \underline{\mu}) = 2.5347 \rightarrow r = 1.5921$$

The euclidean distance can be calculated as below: (It is calculated in the same file)

$$euclidean\ distance \;=\; \sqrt{y_1^2 + y_2^2 + y_3^2} \;=\; 1.5921$$

2.5 Part E

By using the defenition of the Mahalanobis distance from $\underline{X}$ to $\underline{\mu}$ we get:

$$\begin{aligned} r^2 \;&=\; (\underline{X} - \underline{\mu})^T \Sigma^{-1} (\underline{X} - \underline{\mu}) \;=\; (\underline{X} - \underline{\mu})^T (\Phi \Lambda \Phi^{-1})^{-1} (\underline{X} - \underline{\mu}) \;=\; (\underline{X} - \underline{\mu})^T (\Phi \Lambda^{-1} \Phi^{-1}) (\underline{X} - \underline{\mu}) \\ &=\; (\underline{X} - \underline{\mu})^T (\Phi \Lambda^{-\frac{1}{2}} \Lambda^{-\frac{1}{2}} \Phi^{-1}) (\underline{X} - \underline{\mu}) \;=\; (\Lambda^{\frac{-1}{2}} \Phi^T (\underline{X} - \underline{\mu}))^T (\Lambda^{\frac{-1}{2}} \Phi^T (\underline{X} - \underline{\mu})) \\ &=\; (A^T (\underline{X} - \underline{\mu}))^T (A^T (\underline{X} - \underline{\mu})) \;=\; \underline{y}^T \underline{y} \;=\; ||\underline{y}||^2 \end{aligned}$$

where $||\underline{y}||$ is the euclidean distance in the second space.

Chapter 3

Question 3

3.1 Part A

Based on the definition of conditional risk, in order to select a class, its risk must be less than that of other available classes. Therefore, in order to select w_i over w_j, the below equation needs to hold:

$$R(\alpha_i|\underline{X}) \leq R(\alpha_j|\underline{X}) \;\; \forall j \in \{1, 2, ..., c\}$$

In addition, in this problem, an extra condition is employed as well. To pick the class, its conditional risk needs to be less than ignoring it. This fact also provides another condition that is required to be satisfied while picking a class. To formulate it, we have:

$$R(\alpha_i|\underline{X}) \leq R(\alpha_{c+1}|\underline{X})$$

Now we simplify the above equations to reach formulations based on the conditional probability of class:

$$R(\alpha_i|\underline{X}) \leq R(\alpha_j|\underline{X}) \;\; \forall j \in 1, 2, ..., c, \;\; R(\alpha_i|\underline{X}) \; = \; \sum_{j=1}^{c} \lambda(\alpha_i|w_j)p(w_j|\underline{X})$$

$$\rightarrow \sum_{k=1}^{c} \lambda(\alpha_i|w_k)p(w_k|\underline{X}) \leq \sum_{k=1}^{c} \lambda(\alpha_j|w_k)p(w_k|\underline{X}), \text{ (based on the given cost)}$$

$$\rightarrow \lambda_s \sum_{k=1,\, k\neq i}^{c} p(w_k|\underline{X}) \leq \lambda_s \sum_{k=1,\, k\neq j}^{c} p(w_k|\underline{X}), \text{ (assuming } \lambda_s > 0)$$

$$\rightarrow \sum_{k=1,\, k\neq i}^{c} p(w_k|\underline{X}) \leq \sum_{k=1,\, k\neq j}^{c} p(w_k|\underline{X}) \rightarrow 1 - p(w_i|\underline{X}) \leq 1 - p(w_j|\underline{X})$$

$$\rightarrow p(w_j|\underline{X}) \leq p(w_i|\underline{X}) \;\; \forall j \in \{1, 2, ..., c\} \quad \blacksquare$$

$$R(\alpha_i|\underline{X}) \leq R(\alpha_{c+1}|\underline{X}) \rightarrow R(\alpha_i|\underline{X}) \leq \lambda_r \sum_{j=1}^{c} p(w_j|\underline{X}) \rightarrow R(\alpha_i|\underline{X}) \leq \lambda_r, \text{ (the above results)}$$

$$\rightarrow \lambda_s[1 - p(w_i|\underline{X})] \leq \lambda_r, \text{ (assuming } \lambda_s > 0)$$

$$\rightarrow 1 - \frac{\lambda_r}{\lambda_s} \leq p(w_i|\underline{X}) \quad \blacksquare$$

3.2 Part B

It is crystal clear, and also have been discussed that our measurement for choosing one action over another is the conditional risk. Based on the defined costs in this problem, and as was also concluded in the last part, the conditional risk of an action $\alpha_i \ \ \forall i \in \{1, 2, ..., c\}$ can be simplified as below:

$$R(\alpha_i|\underline{X}) \ = \ \sum_{j=1}^{c} \lambda(\alpha_i|w_j)p(w_j|\underline{X}) \ = \ \lambda_s \sum_{j=1,\, j\neq i}^{c} p(w_j|\underline{X}) \ = \ \lambda_s[1 - p(w_i|\underline{X})]$$

In addition, the conditional risk of the action α_{c+1} can be simplified as below:

$$R(\alpha_{c+1}|\underline{X}) \ = \ \lambda_r \sum_{j=1}^{c} p(w_j|\underline{X}) \ = \ \lambda_r$$

Based on the grounds of the fact that for a classifier minimizing the conditional risk, $g_i(\underline{X}) = -R(\alpha_i|\underline{X})$ holds for the various choices of i, and also based on the above equations, $g_i(\underline{X})$ can be defined as below:

- if $i \in \{1, ..., c\}$:

  $$-\lambda_s[1 - p(w_i|\underline{X})]$$

  However, since these functions are a tool for comparision, the terms independent of i can be eliminated to achieve a more simplified version:

  $$g_i(\underline{X}) \ = \ p(w_i|\underline{X}) \ = \ p(\underline{X}|w_i)p(w_i)$$

- if $i = c + 1$:

  $$g_{c+1}(\underline{X}) \ = \ -\lambda_r$$

  However, since these functions are a tool for comparision, the terms independent of i can be added to achieve a version similar to the wanted one in the problem:

$$\begin{aligned} g_{c+1}(\underline{X}) \ &= \ -\lambda_r \ = \ -\frac{\lambda_r}{\lambda_s} \ = \ 1 - \frac{\lambda_r}{\lambda_s} \ = \ \frac{\lambda_s - \lambda_r}{\lambda_s} * 1 \ = \ \frac{\lambda_s - \lambda_r}{\lambda_s} \sum_{j=1}^{c} p(w_j|\underline{X}) \\ &= \ \frac{\lambda_s - \lambda_r}{\lambda_s} \sum_{j=1}^{c} p(\underline{X}|w_j)p(w_j) \end{aligned}$$

$$\rightarrow g_i(\underline{X}) \ = \ \begin{cases} p(\underline{X}|w_i)p(w_i), & i = 1, ..., c \\ \\ \frac{\lambda_s - \lambda_r}{\lambda_s} \sum_{j=1}^{c} p(\underline{X}|w_j)p(w_j), & i = c + 1 \end{cases}$$

3.3 Part C

In order to answer this problem, first of all, we discuss the two extream cases.

- $\frac{\lambda_r}{\lambda_s} = 0$:
  This fraction is zero when $\lambda_r = 0$ or $\lambda_r \lll \lambda_s$. In the first case, refusing to choose a class has the same cost as choosing the correct class. However, not selecting any classes has the benefit of unambiguity. In cases where $\lambda_r \lll \lambda_s$ takes place, as it is crystal clear, the cost of not making a decision is way smaller than being inaccurate. As a result, in either cases, refusing to include the incoming data in a class is the best strategy, since it has the least cost. As a

matter of fact, this means that none of the classes are selected which is anothor word for not classifing. This fact can also be shown in mathemathical terms:

$$1 - \frac{\lambda_r}{\lambda_s} \leq p(w_i|\underline{X}),\ \frac{\lambda_r}{\lambda_s} = 0,\ p(w_i|\underline{X}) \leq 1$$

$$\to p(w_i|\underline{X}) = 1 : \text{which is only true when there is only one class in hand } \equiv \text{ no classification}$$

Therefore, in this case, the agent is too risk averse to take any choices that may cause errors, and tends to not make a decision.

- $\frac{\lambda_r}{\lambda_s} = 1$:
  This fraction is one when $\lambda_r = \lambda_s$. This suggest that making a desicion is the best strategy. In other word, if the agent chooses a class, in cases where the choice was accurate, the agent faces no cost and in other cases, it faces no harm more than not selecting a class. This fact can also be shown in mathemathical terms:

$$1 - \frac{\lambda_r}{\lambda_s} \leq p(w_i|\underline{X}),\ \frac{\lambda_r}{\lambda_s} = 1,\ 0 \leq p(w_i|\underline{X}) \leq 1$$

$$\to 0 \leq p(w_i|\underline{X}) \leq 1 : \text{which is true for any choices of } i.$$

  Therefore, the agent being a risk seeker, tends to choose no matter what the outcome is.

To sum up, based on the strategies mentioned in the two extream cases, as $\frac{\lambda_r}{\lambda_s}$ increases from zero to one, the agent's tendency for clssifying increases and becomes more willing to take risks.

Chapter 4

Question 4

Generally, to solve a problem as below:

$$\begin{cases} min\ f_0(x) \\ s.t.\ f_i(x) \le 0, \quad for\ i = 1, 2,\ ..., n \\ \qquad h_k(x) = 0, \quad for\ k = 1, 2,\ ..., p \end{cases}$$

A lagrangian function can be created as below:

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{n} \lambda_i f_i(X) + \sum_{k=1}^{p} \nu_k h_k(x)$$

By derivating with respect to x, λ, ν, the optimal point which satisfies the sufficient conditions are found.

For this case based on the given information, there are not inequality, therefore the KKT conditions are not needed. Therefore, the problem can be stated as below:

$$\begin{cases} min\ f(\mathbf{x}) \\ s.t.\ g_l(\mathbf{x}) = \mathbf{C}_l, \quad for\ l = 1, 2,\ ..., k \end{cases}$$

A lagrangian function can be created as below:

$$L(x, \lambda, \mathbf{x}) = f(\mathbf{x}) + \sum_{l=1}^{k} \lambda_l [g_l(\mathbf{x}) - \mathbf{C}_l]$$

A bordered Hessian is used for the second derivative test in certain constrained optimization problems. Given the above problem, the Hessian of the lagrange function is defined as below:

$$\mathbf{H}(L) = \begin{bmatrix} \frac{\partial^2 L}{\partial \lambda^2} & \frac{\partial^2 L}{\partial \lambda \partial x} \\ \frac{(\partial^2 L}{\partial \lambda \partial x)^T} & \frac{\partial^2 L}{\partial x^2} \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 & -(g_1)_{x_1} & \cdots & -(g_1)_{x_n} \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & -(g_k)_{x_1} & \cdots & -(g_k)_{x_n} \\ -(g_1)_{x_1} & \cdots & -(g_k)_{x_1} & L_{x_1 x_2} & \cdots & L_{x_1 x_n} \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ -(g_1)_{x_n} & \cdots & -(g_k)_{x_n} & L_{x_n x_1} & \cdots & L_{x_n x_n} \end{bmatrix}$$

Based on the fact that the above matrix is p.d or n.d, it is detected that the found optimal point is minimum or maximum.

4.1 Distance from a Plane

In order to minimize the distance of x_0 point from the given plane by employing the lagrangian coefficients, we need to define the cost function and the constrains.

- **Distance**: Distance of a point from a plane: $||x - x_0||^2$
- **Constraint**: Plane's equation: $\underline{w}^T\underline{x} + b = 0$. However, as can be seen there is no inequality contraint.

Based on the above explanations, the problem can be written as: (the $\frac{1}{2}$ is used for normalization)

$$\begin{cases} min\ \frac{1}{2}||\underline{x} - \underline{x}_0||^2 \\ s.t.\ \underline{w}^T\underline{x} + b = 0 \end{cases} \rightarrow L(\underline{x}, \lambda) = \frac{1}{2}||\underline{x} - \underline{x}_0||^2 + \lambda(\underline{w}^T\underline{x} + b)$$

$$\rightarrow \frac{\partial L}{\partial \underline{x}} = \underline{x} - \underline{x}_0 + \lambda\underline{w} = 0 \rightarrow \underline{x} = \underline{x}_0 - \lambda\underline{w}$$

- **Method 1**:

$$\rightarrow h(\lambda) = inf_x L(\underline{x}, \lambda) = \frac{1}{2}||\lambda\underline{w}||^2 + \lambda(\underline{w}^T\underline{x}_0 - \underline{w}^T\lambda\underline{w} + b)$$

$$\rightarrow \frac{\partial h}{\partial \lambda} = \lambda\underline{w}^T\underline{w} + \underline{w}^T\underline{x}_0 - \lambda\underline{w}^T\underline{w} + b + -\lambda\underline{w}^T\underline{w} = 0$$

$$\rightarrow \lambda = \frac{\underline{w}^T\underline{x}_0 + b}{\underline{w}^T\underline{w}} \rightarrow \boxed{\underline{x}^* = \underline{x}_0 - \frac{\underline{w}^T\underline{x}_0 + b}{||\underline{w}||^2}\underline{w}}$$

- **Method 2**:

$$\rightarrow \frac{\partial L}{\partial \lambda} = 0 \rightarrow g(x) = 0 \rightarrow \underline{w}^T\underline{x} + b = 0,\ \underline{x} = \underline{x}_0 - \lambda\underline{w}$$

$$\rightarrow \lambda = \frac{\underline{w}^T\underline{x}_0 + b}{\underline{w}^T\underline{w}} \rightarrow \boxed{\underline{x}^* = \underline{x}_0 - \frac{\underline{w}^T\underline{x}_0 + b}{||\underline{w}||^2}\underline{w}}$$

Now what we need to do is to check whether the sufficient condition for the evaluated answer holds and what are the constrains for it to be true. The condition that needs to be added is that $w \neq 0$.

Therefore, the distance can be calculated as below:

$$d = ||\frac{\underline{w}^T\underline{x}_0 + b}{||\underline{w}||^2}\underline{w}||$$

4.2 Distance from a Ellipsoid

Same as the last part, first of all the optimization problem is defined as below:

$$\begin{cases} min\ ||\underline{x} - \underline{x}_0||^2 \\ s.t.\ \underline{x}^T A\underline{x} = 1 \end{cases}$$

Therefore, if we create the lagrangian function we get:

$$L(\underline{x}, \lambda) = ||\underline{x} - \underline{x}_0||^2 + \lambda(\underline{x}^T A\underline{x} - 1)$$

Now in the same procedure as the method 2 is taken, with keeping in mind that A is a p.d. matrix:

$$\frac{\partial L}{\partial \underline{x}} = 2(\underline{x} - \underline{x}_0) + \lambda(2A\underline{x}) = 0 \rightarrow \underline{x} = (\lambda A + I)^{-1}x_0,\ \underline{x}^T A\underline{x} = 1$$

In order to find the optimal point in these two cases, there are two path we can take:

- **Method 1**:
  The cost can be defined as below:

  $$J(\lambda) = 2\ ln x^T Ax - 1, x = (I + \lambda A)^{-1}\underline{x}_0$$

  However, in order to define such function, we need to make sure $x^T Ax - 1$ it is positive. The gradient of the cost function is defined and λ is updated iteratively by gradient descent method.

  $$\lambda_t := \lambda_t + \alpha \nabla_\lambda J(\lambda),$$
  $$\nabla J(\lambda) = \frac{-2x_t^T A(\lambda_t A + I)^{-1} A(\lambda_t A + I)^{-1}\underline{x}_0}{(I + \lambda_t A)^{-1}\underline{x}_0},$$
  $$x_t = (I + \lambda_t A)^{-1}\underline{x}_0$$

  Afterward, the optimal value of $\underline{x}$ is calculated using λ.

- **Method 2**:
  Using the fact that an ellipsiod's representor matrix is symmetric and p.d:

  $$x = (I + \lambda A)^{-1}\underline{x}_0 \rightarrow x_i = \frac{x_{0i}}{1 + \lambda\nu_i},\ \underline{x}^T A\underline{x} = \sum_i \nu_i (x_i)^2 = 1$$
  $$\rightarrow \sum_i \nu_i \frac{\nu_i (x_i)^2}{(1 + \lambda\nu_i)} = 1$$

  To find the optimal choice of λ Newton method is suggested. Afterward, the optimal value of $\underline{x}$ is calculated using λ.

4.3 Examples

Since we are asked not to employ numerical method in this part, the general equations are used in both cases. These equations are:

$$\begin{cases} min\ \frac{1}{2}||\underline{x} - \underline{x}_0||^2 \\ s.t.\ \underline{x}^T A\underline{x} = 1 \end{cases} \rightarrow L(\underline{x}, \lambda) = \frac{1}{2}||\underline{x} - \underline{x}_0||^2 + \lambda(\underline{x}^T A\underline{x} - 1)$$

$$\rightarrow \frac{\partial L}{\partial \underline{x}} = \underline{x} - \underline{x}_0 + \lambda((A + A^T)\underline{x}) = 0 \rightarrow \boxed{\underline{x} = (\lambda(A + A^T) + I)^{-1}x_0},\ \boxed{\underline{x}^T A\underline{x} = 1}$$

4.3.1 Example 1

Since $A = I$, and assuming $\underline{x} = [x_1\ x_2]^T$, we get:

$$\begin{cases} \underline{x} = (2\lambda A + I)^{-1} x_0 = \frac{2}{\lambda+1} I \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{2}{\lambda+1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \rightarrow x_1 = x_2 = \frac{2}{\lambda+1} \\ \underline{x}^T A \underline{x} = 1 \rightarrow \underline{x}^T \underline{x} = 1 \rightarrow x_1^2 + x_2^2 = 1 \end{cases}$$

$\rightarrow \lambda = -1 \pm 2\sqrt{2}$, Based on KKT condition it must be positive. $\rightarrow \lambda = -1 + 2\sqrt{2}$

$$\rightarrow \boxed{x_1 = x_2 = \frac{1}{\sqrt{2}}} \rightarrow \boxed{distance = 0.293}$$

4.3.2 Example 2

$$\underline{x} = (\begin{bmatrix} 4\lambda + 1 & 4\lambda \\ 4\lambda & 2\lambda + 1 \end{bmatrix})^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{-8\lambda^2 + 6\lambda + 1} \begin{bmatrix} 2\lambda + 1 & -4\lambda \\ -4\lambda & 4\lambda + 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$= \frac{1}{-8\lambda^2 + 6\lambda + 1} \begin{bmatrix} -2\lambda + 1 \\ 1 \end{bmatrix} \rightarrow x_1 = \frac{-2\lambda + 1}{-8\lambda^2 + 6\lambda + 1},\ x_2 = \frac{1}{-8\lambda^2 + 6\lambda + 1}$$

$$\underline{x}^T A \underline{x} = 1 \rightarrow 2x_1^2 + 4x_1 x_2 + x_2^2 = 1$$

$$\rightarrow 2[\frac{-2\lambda + 1}{-8\lambda^2 + 6\lambda + 1}]^2 + 4[\frac{-2\lambda + 1}{-8\lambda^2 + 6\lambda + 1}][\frac{1}{-8\lambda^2 + 6\lambda + 1}] + [\frac{1}{-8\lambda^2 + 6\lambda + 1}]^2 = 1$$

$$\lambda = -0.51101,\ 0.22601,\ 0.8925 \pm 0.123j,$$

Based on KKT conditions it must be positive and based on the domain of $\underline{x}$ it must also be real.

$$\rightarrow \lambda = 0.22601$$

$$\rightarrow \boxed{x_1 = 0.2814,\ x_2 = 0.5135} \rightarrow \boxed{distance = 0.6136}$$

Chapter 5

Question 5

5.1 Part A

For a two category classification, the interval where each choice is taken can be represented as below:

$$\begin{cases} w_1 & \frac{p(\underline{X}|w_2)}{p(\underline{X}|w_1)} > \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}} \frac{p(w_2)}{p(w_1)} \\ \\ w_2 & O.W. \end{cases}$$

In binary case, where $\lambda_{11} = \lambda_{22} = 0$ holds, the $\frac{\lambda_{12}}{\lambda_{21}}$ plays a prominent role regarding the alternations in the decision intervals of these two classes. As the above relation shows, as $\frac{\lambda_{12}}{\lambda_{21}}$ increases, the interval where w_1 is chosen shrinks. Therefore, we rather select w_2, not w_1 in most of the cases. In fact, this matter is crystal clear without the above formlation as well. The increament in the stated ratio, means that selecting w_1 over w_2 costs more than its opposite case. As a matter of fact, for decreasing the risk, w_1 is chosen less often. However, as this ratio decreases, the attractiveness of w_2 decreases, while that of w_1 increases. This fact occurs due to the fact that w_1 seems like a choice with lower risk. This matter is depicted in the below plot:

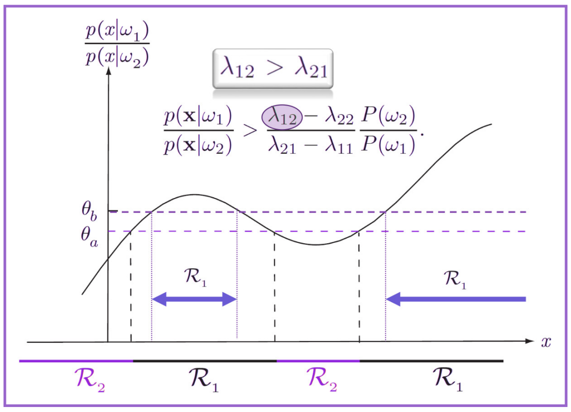

$\frac{p(x|\omega_1)}{p(x|\omega_2)}$
$\lambda_{12} > \lambda_{21}$
$\frac{p(\mathbf{x}|\omega_1)}{p(\mathbf{x}|\omega_2)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{P(\omega_2)}{P(\omega_1)}$.
θ_b
θ_a
$\mathcal{R}_1$
$\mathcal{R}_1$
x
$\mathcal{R}_2$
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_1$

FIGURE 5.1: Changes in the interval with respect to $\frac{\lambda_{12}}{\lambda_{21}}$

5.2 Part B

In favor of explaining the reducible error, let's assume the world is deterministic. Any outcome y is an unknown function of $\mathbf{X}$, which is the representor of *everything*. Although we obviously do not have the power to observe *everything*, we can observe some of the critical things. In this case, we hope the ones we do not have the ability to observe, are among the non-essnetial ones. Therefore, it is crystal clear that we cannot determine the deterministic relationship between y and $\mathbf{X}$, and only an estimation is in hand. Also, these is a noise term corresponding to the things we did not have the chance to measure. This fact can be represented mathematically as well:

$$\text{reality: } y = f(\mathbf{X}) + \epsilon, \quad \text{what we predict: } \hat{y} = \hat{f}(\mathbf{X})$$

As the above explanations and the equations suggest, our prediction is imperfect. In fact, there is a difference between what we predict and what we have observed, called *the prediction error*. It is obvious that our goal in any prediction is to reduce this error. To do so we need to know that this error consists of two parts: The reducible and irreducible error.

The reducible error arises from the mismatch between $\hat{f}$ and f. By employing various methods, we can reduce or even eliminate this error to improve our predictions. (This error is also represented by MSE) The below plot for a two classifier problem, depicts where reducible error is.

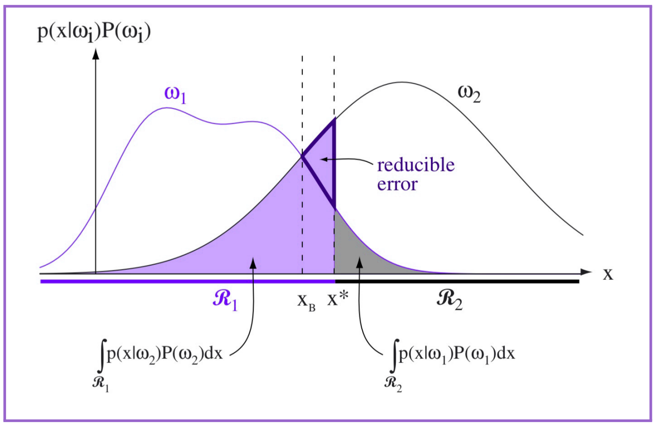


FIGURE 5.2: Distribution of two classes

This plot suggests that if we choose the decision boundary in an accurate way, we can eliminate the reducible error. Therefore, x_B is the optimum decision boundary while x^*. Although x_B is the optimum decision boundary, it is essential to mention that this threshold cannot eliminate the irreducible errors which are also highlighted in the above plot.

5.3 Part C

The classifier employs a threshold x^* for determining whether the external pulse is present or not. However, our goal is to find some measure of discriminating in a form independent of the choice of x^*. Such a measurement is *discriminability*, decribing the inherent and unchangable properties due to noise and the strength of the external signal, but not based on the decision strategy. In the case of comparing two normally distributed random variables, discriminability can be formulated as:

$$d' = \frac{|\mu_2 - \mu_1|}{\sigma}$$

Based on the definition given above, it is obvious that higher d' is what we desire. Meaning, we would like these two to have expected values far from one to another, as well as small standard devation, so that most of the data be near the separated means.

In addition, it is clear that expected values cannot determine this matter alone. In fact, when two distributions' mean are far from one another, while their variance is large as well, we cannot distinguish these two from one another, since data is scattered in a large interval and overlapping is huge.

It is also essential to note that in cases where we do not have the knowledge of μ_1, μ_2, and σ, but have the samples of both classes in hand,(in some cases even the decision strategy is clear as well) gaining some insights about d' is also possible by ROC or AUC curves.

5.4 Part D

Generally, we have the samples of both classes, meaning we have the knowledge about the states of both classes. In some cases, we are also familiar with the decision strategy. However, we do not have the knowledge of μ_1, μ_2, and σ. What we seek to do is to determine or gain an insight of d'. In these cases ROC curve is used.

If we have a large number of trials, determining the *hit* and *false alarm* is done experimentally. The hit and false alarm for the below distributions are defined:

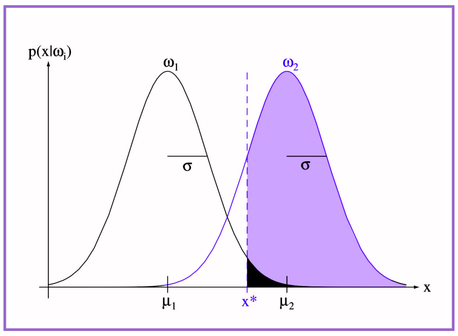


FIGURE 5.3: Two normal distributions

- *hit*: $P(x \in w_2 | x \in w_2) \rightarrow$ the probability that the internal signal is above x^* given that the external signal is present.
- *false alarm*: $P(x \in w_2 | x \in w_1) \rightarrow$ the probability that the internal signal is above x^* given that there is no external signal present.

In fact, for each fixed value of d', decision boundary x^* is varied. Each representor point is depicted smoothly in a graph with respect to hit and false alarm afterwards. By plotting this curve for different values, we can tell how much model is capable of distinguishing between classes. The x-axis can be interpreted as a measure of liberalism of the classifier represented by its false alarm. The y-axis represents how well it is at detecting w_2, depicting the classifier's hit.

As figure 5.2 suggets, through any pair of hit and false alarm rates passes one and only one ROC curve; thus so long as neither rate is exactly one or zero, we can determine the discriminability from these rates. In addition, in the below plot, this fact can be seen easier.

A perfect classifier's ROC curve passes through (0,1), meaning it can classify all w_2 correctly without a single error. This results in an area under the curve of exactly 1. Intuitively, a more conservative classifier (which labels less stuff as w_2) has

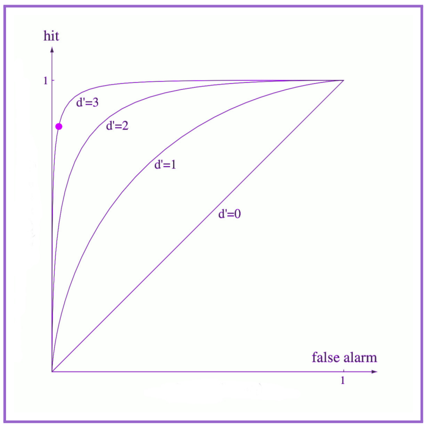


FIGURE 5.4: ROC curve for different values of d'

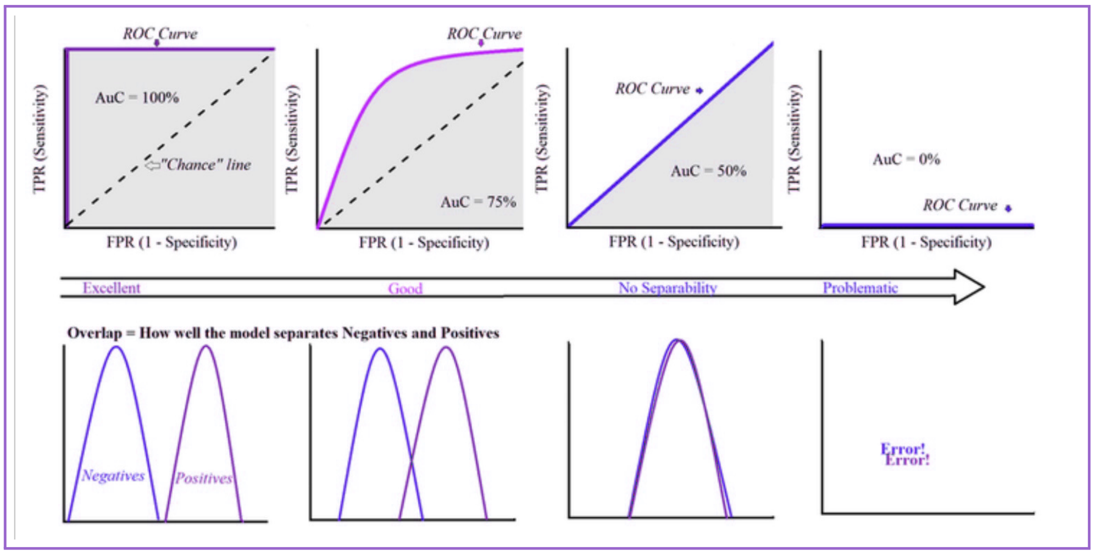


FIGURE 5.5: ROC curve for different cases of distributions d'

higher precision and lower sensitivity than a more liberal one. When the threshold for positive prediction decreases (e.g. the required positive confidence score decreases), both the false alarm and hit rise monotonically. On the grounds of this fact, as well as what the above plots suggest, ROC curve always increases monotonically.

5.5 Part E

In classification, the goal is to find a mapping from inputs $\underline{X}$ (features represented by vector X consisting $\underline{x_1}$, $\underline{x_2}$, ... , $\underline{x_n}$) to outputs w (class represented by vector Ω consisting of $\underline{w_1}$, $\underline{w_2}$, ... , $\underline{w_c}$) given a labeled set of input-ouptput pairs.

- **Generative Approach**: This approach creates a joint model of feature vectors and classes, represented by $p(\underline{X}, w)$. By employing this joint model, $p(w_j|k)$ can be driven, in order to form the classification. It is essential to note that finding $p(\underline{X}, w)$ is an arduous task because it needs a lot of samples. Therefore, we may face the curse of dimentionality problem.
- **Discriminative Approach**: This approach creates a model of the form of $p(w_j|k)$ directly. Therefore, it requires fewer samples.

As the above descriptions suggest, their difference is in what they evaluate first.

Chapter 6

Question 6

6.1 Part A

6.1.1 Chebyshev Distance:

As is formulated in the problem, Chebyshev distance for a two dimentional space can be conveyed as below:

$$d = max\{|y_2 - y_1|,\ |x_2 - x_1|\}$$

Therefore, first of all, the distance between each labeled point and the target point with respect to each dimention is evaluated. Secondly, for each point, the maximum value between the two numbers are selected as the Chebyshev distance. Afterward, the minimum value between these numbers is selected. By what color it has, the class our target point belongs to is found.
If all of the above discriptions are carried out, the result shows that our point belongs to the **red** class. (The python code for simulating is in "Q6.py")

6.1.2 Manhattan Distance:

As is formulated in the problem, for calculating the Manhattan distance the below formulation is used:

$$d = |y_2 - y_1|\ +\ |x_2 - x_1|$$

Therefore, first of all, the distance between each labeled point and the target point with respect to each dimention is evaluated. Secondly, for each point, these values are added up so that the Manhattan distance is calculated. Afterward, the minimum value between these numbers is selected. By what color it has, the class our target point belongs to is found.
If all of the above discriptions are carried out, the result shows that our point belongs to the **blue** class. (The python code for simulating is in "Q6.py")

6.1.3 Euclidean Distance:

As is formulated in the problem, for calculating the Euclidean distance the below formulation is used:

$$d = |y_2 - y_1|^2\ +\ |x_2 - x_1|^2$$

Therefore, first of all, the distance between each labeled point and the target point with respect to each dimention is evaluated. Secondly, for each point, these values are squared and then added up so that the Euclidean distance is calculated. Afterward,

the minimum value between these numbers is selected. By what color it has, the class our target point belongs to is found.
If all of the above discriptions are carried out, the result shows that our point belongs to the **red** class. (The python code for simulating is in "Q6.py")

6.2 Part B

The Mahalanobis distance from $\underline{X}$ to $\underline{\mu}$ can be calculated as below: (It is calculated in the same file)

$$r^2 = (\underline{X} - \underline{\mu})^T \Sigma^{-1} (\underline{X} - \underline{\mu})$$

- $\underline{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \ \Sigma = \begin{bmatrix} 3 & -3 \\ -3 & 3.5 \end{bmatrix}$:

$$r = \sqrt{(\begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix})^T \begin{bmatrix} 3 & -3 \\ -3 & 3.5 \end{bmatrix}^{-1} (\begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix})} = 4.330127$$

- $\underline{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \ \Sigma = \begin{bmatrix} 3 & 3 \\ 3 & 3.5 \end{bmatrix}$:

$$r = \sqrt{(\begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix})^T \begin{bmatrix} 3 & 3 \\ 3 & 3.5 \end{bmatrix}^{-1} (\begin{bmatrix} 1.5 \\ 1.5 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix})} = 0.866025$$

6.3 Part C

For a matrix to be a valid covariance matrix, it needs to satisfy the below conditions:

- It needs to hermitian. In other words: $A = A^H$. For the given matrix, this property holds if α is a **real** number.
- It needs to be a none-negative definite (semi-positive definite) matrix. In order to have the property of being a n.n.d matrix, matrix A needs to satisfy the below condition:

  $$\forall \underline{x} \neq 0 : \ \underline{x}^H A \underline{x} \geq 0$$

  However, since a covariance matrix needs to be hermitain as well, if its eigenvalues are non-negative, then we can say it is n.n.d too. Therefore, the given matrix's eigenvalues are calculated:

  $$det(\Sigma - \lambda I) = 0 \rightarrow \begin{vmatrix} 5-\lambda & \alpha \\ \alpha & 4-\lambda \end{vmatrix} = 0 \rightarrow (5-\lambda)(4-\lambda) - \alpha^2 = 0$$

  $$\rightarrow \lambda^2 - 9\lambda + (20 - \alpha^2) = 0, \text{ eigenvalues need to be non-negative} \rightarrow 20 - \alpha^2 \geq 0$$
  $$\rightarrow -\sqrt{20} \leq \alpha \leq \sqrt{20}$$

Therefore, if $\alpha \in \mathbb{R}$ and $-\sqrt{20} \leq \alpha \leq \sqrt{20}$, the given matrix can be a valid one for being a covariance matrix.

6.4 Part D

Condional risk of a decision α_i is defined as:

$$R(\alpha_i|\underline{X}) = \sum_{j=1}^{c} \lambda(\alpha_i|w_j)p(w_j|\underline{X})$$

Based on the given values, the conditional risk for both classes can be calculated with the above formulation:

$$R(\alpha_1|\underline{X}) = \lambda_{11}p(w_1|\underline{X}) + \lambda_{12}p(w_2|\underline{X})$$
$$R(\alpha_2|\underline{X}) = \lambda_{21}p(w_1|\underline{X}) + \lambda_{22}p(w_2|\underline{X})$$

The first choice takes place where the risk of choosing decision one is less than that of decision two:

$$R(\alpha_1|\underline{X}) < R(\alpha_2|\underline{X}) \rightarrow \frac{p(\underline{X}|w_1)}{p(\underline{X}|w_2)} > \frac{\lambda_{21} - \lambda_{22}}{\lambda_{21} - \lambda_{11}}.\frac{p(w_2)}{p(w_1)}, \text{ (employing the given information)}$$

$$\rightarrow \frac{x + 0.5}{\frac{3}{4}x^2 + \frac{3}{4}} > \frac{2-1}{3-1}.\frac{\frac{1}{4}}{\frac{3}{4}} \rightarrow x^2 - 8x - 3 < 0 \rightarrow 4 - \sqrt{19} < x < 4 + \sqrt{19} \rightarrow -0.3589 < x < 8.3589$$

Owing to the fact that the calculated interval covers the valid interval for x, we can coclude that in this interval, decision one is selected all the time.

Chapter 7

Question 7

Generalized Linear Models:

In general, models can handel more complicated situations and analyze the simultaneous effects of multiple variabels, including mixtures of categorical and continuous variables. On the grounds of the fact that its parameters provide measures of the strength of associations between these variabels, the focus is on estimating them.

The term ***generalized linear model*** refers to a large class of models where the responce variable represented by y_i is assumed to follow an exponential family distribution with mean μ_i, which is assumed to be (often nonlinear) function of $x_i^T\beta$. Mathematically speaking:

$$p(y : \eta) = b(y)exp\{\eta^T T(y) - a(\eta)\}$$

where:

- η: Natual/cononical parameter.
- $T(y)$: The sufficient statistics.
- $a(\eta)$: Log partition function.
- $exp(-a(\eta))$: Plays the role of a normalization constant.

However, it is essential to note that this method is available for the cases where the data generating process can be linearized (is the case for the exponential family)

Logestic Regression:

Logistic regression is basically a supervised classification algorithm, where the target value (responce variable) represented as y, can only take discrete values for a given set of features. In fact, the model builds a regression model to predict the probability that a given data entry belongs to the category numbered as '1'. These models try to do so by modeling the data with the sigmoid function represented as below:

$$g(z) = \frac{1}{1 + e^{-z}}$$

It is essential to note that logistic regression becomes a classification technique only when a decision threshold is brought into the picture.

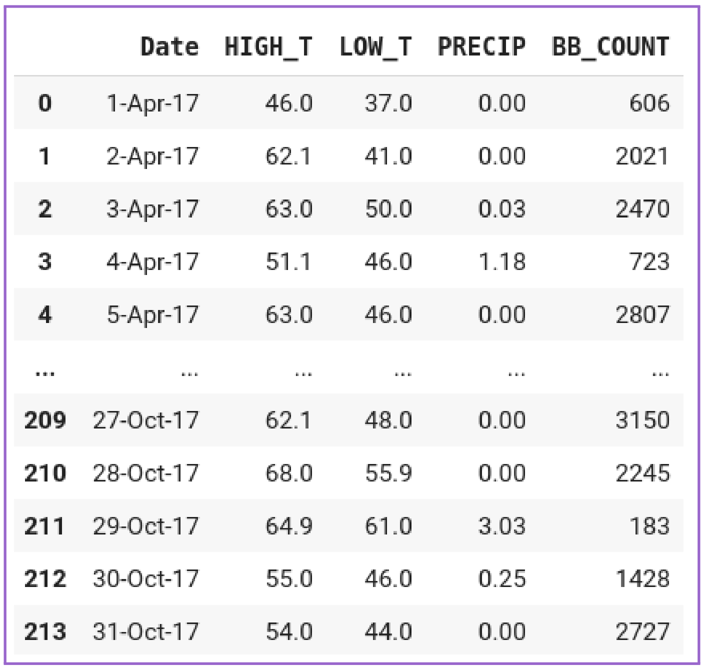
Linear Regression:

In linear regression the goal is to achieve a linear model to represent the relationship between y and X.

Therefore, in ***Linear Regression***, it is assumed that the model is linear, in ***Logistic Regression***, the output is binary, and in ***Generalized Linear Regression*** the output is discrete or bounded while having the exponential family distribution. As its name suggests it extends the linear models in ***Linear Regression*** by employing a link function as in ***Logistic Regression***, as well as the exponential family contribution. In addition, it is crystal clear that if the data is binary, the generalized model is actually logisic.

Simulation:

First of all, in order to gain insight of what the dataset looks like, it is represented as below:



	Date	HIGH_T	LOW_T	PRECIP	BB_COUNT
0	1-Apr-17	46.0	37.0	0.00	606
1	2-Apr-17	62.1	41.0	0.00	2021
2	3-Apr-17	63.0	50.0	0.03	2470
3	4-Apr-17	51.1	46.0	1.18	723
4	5-Apr-17	63.0	46.0	0.00	2807
...	...	...	...	...	...
209	27-Oct-17	62.1	48.0	0.00	3150
210	28-Oct-17	68.0	55.9	0.00	2245
211	29-Oct-17	64.9	61.0	3.03	183
212	30-Oct-17	55.0	46.0	0.25	1428
213	31-Oct-17	54.0	44.0	0.00	2727

FIGURE 7.1: NYC Cyclers Contests Dataset

After that, the $HIGH-T$, $LOW-T$, and $PRECIP$ are extracted in order to predict $BB-COUNT$ column. Meaning, we are assuming that the first three columns are features and the last column acts as their class. In addition, since we want to represent the data in the end based on their dates, by employing zip in python, a tuple of each data with its $BB-COUNT$ is created.

The next step is to separate 80% of data as train dataset, while the remaining 20% is the test dataset.

To perform GLM, $TweedieRegressor$ from *sklearn.linear* $-$ *model* is selected, in which if the *power* parameter is selected as 1, the condition on the poisson disctibution is satisfied as well.

As was discussed, Generalized Linear Models (GLM) extend linear models in two ways. First, the predicted values $\hat{}y$ are linked to a linear combination of the input variables $\mathbf{X}$ via an inverse link function h as:

$$\hat{y}(w, \mathbf{X}) = h(\mathbf{X}w)$$

Secondly, the squared loss function is replaced by the unite deviance of a distribution of the exponential family.

The minimization problem them becomes as below:

$$min_w \frac{1}{2n_{samples}} \sum_i d(y_i, \hat{y}_i) + \frac{\alpha}{2}||w||_2$$

where α is the l2 regulazation penalty. When sample weights are provided, the average becomes a weighted average.

Therefore, by choosing an appropriate value for α, we can acheive a proper extimation. By employing and considering all the conditions, the resulting estimation is as below:

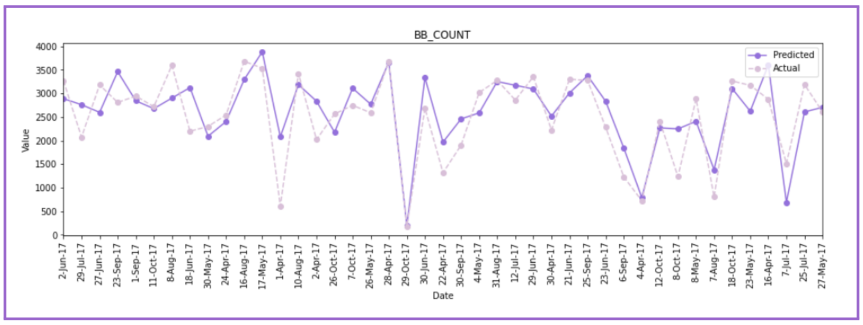


FIGURE 7.2: Predicted and Actual Values of $BB - Count$

In addition, since we used *zip*, we have assecc to dates for each value of $BB - Count$. Therefore, plotting is a piece of cake!

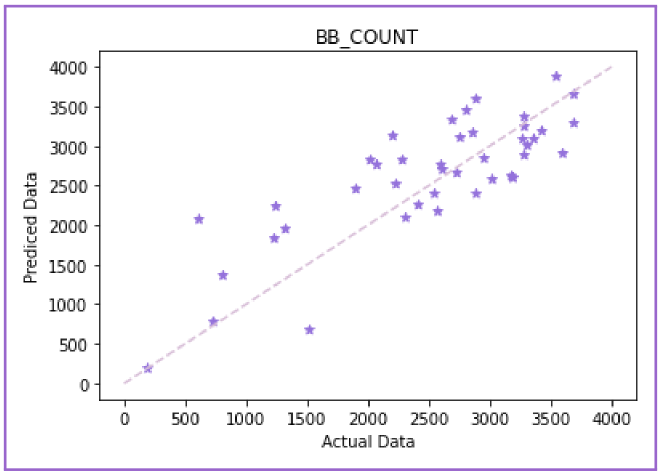


FIGURE 7.3: Scatter Plot of Predicted and Actual Values of $BB - Count$

In order to show that the predicted labels are close to the actual labels, the scattering plot is depicted as well. The data in this scattering plot are almost scattered around $x = y$ line. This means that x axis data is almost the same as the data of y axis. Thus, the prediction is performed well.

Chapter 8

Question 8

8.1 Part A

In this part, the scattering plot of data, based on two different features are plotted as below:

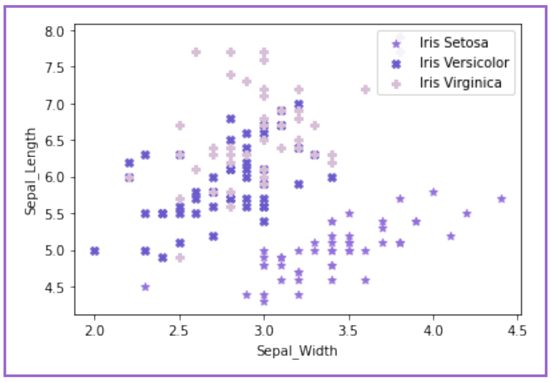


FIGURE 8.1: Plot of Iris Dataset Based on Sepal Width and Sepal Length

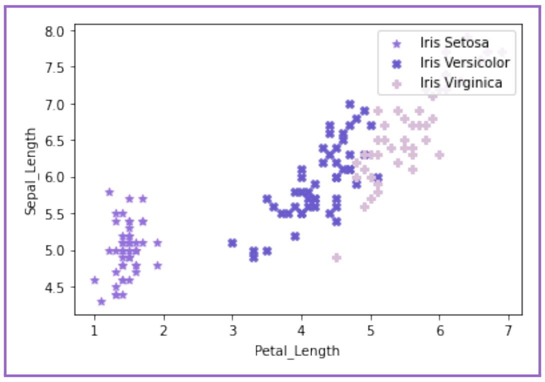


FIGURE 8.2: Plot of Iris Dataset Based on Petal Length and Sepal Length

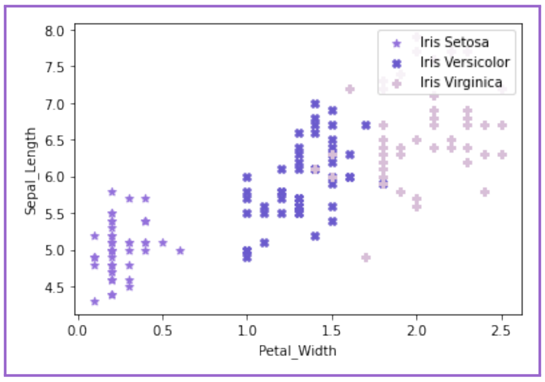


FIGURE 8.3: Plot of Iris Dataset Based on Petal Width and Sepal Length

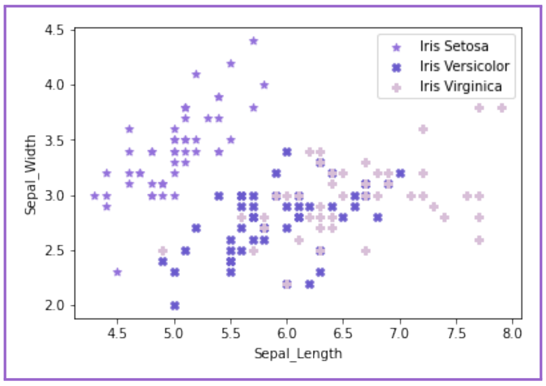


FIGURE 8.4: Plot of Iris Dataset Based on Sepal Length and Sepal Width

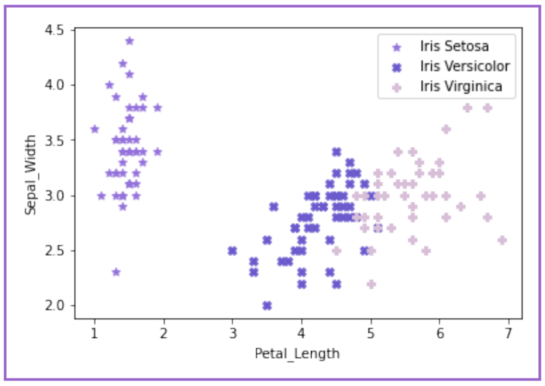


FIGURE 8.5: Plot of Iris Dataset Based on Petal Length and Sepal Width

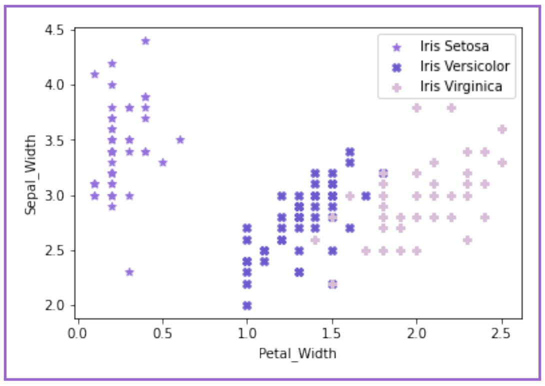


FIGURE 8.6: Plot of Iris Dataset Based on Petal Width and Sepal Width

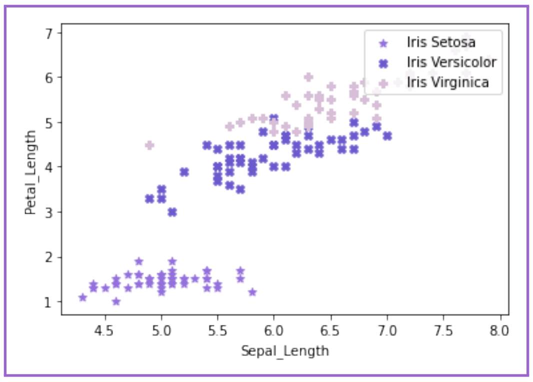


FIGURE 8.7: Plot of Iris Dataset Based on Sepal Length and Petal Length

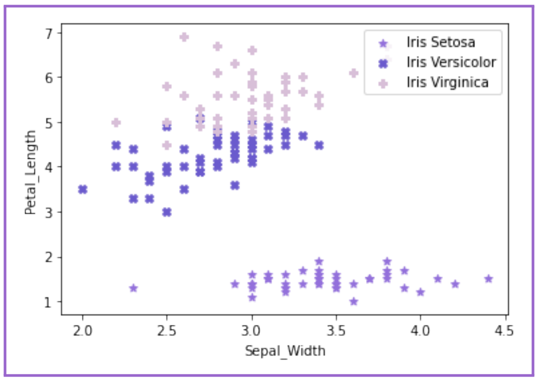


FIGURE 8.8: Plot of Iris Dataset Based on Sepal Width and Petal Length

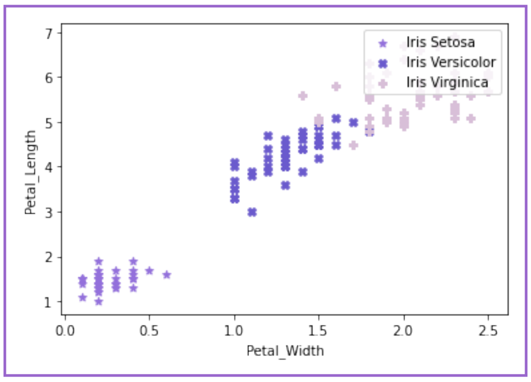


FIGURE 8.9: Plot of Iris Dataset Based on Petal Width and Petal Length

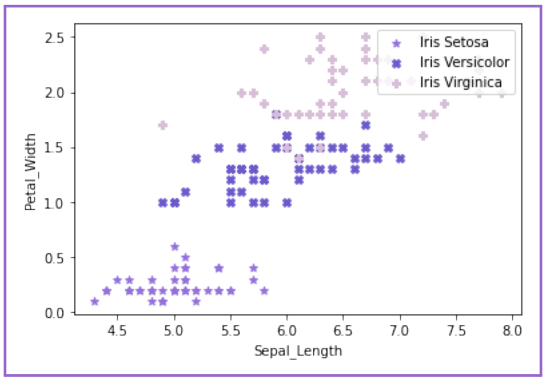


FIGURE 8.10: Plot of Iris Dataset Based on Sepal Length and Petal Width

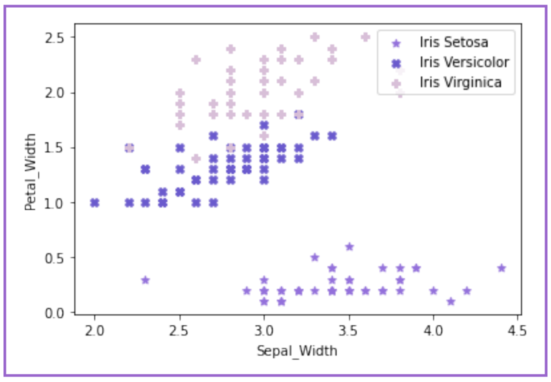


FIGURE 8.11: Plot of Iris Dataset Based on Sepal Width and Petal Width

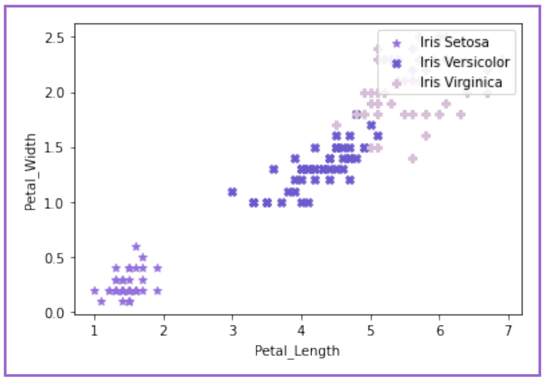


FIGURE 8.12: Plot of Iris Dataset Based on Petal Length and Petal Width

It is crystal clear that if in each above figure the x and y axis are swaped, the plot is mirrored based on the $x = y$ line. However, in order to represent in all cases, they are depicted as well. In addition, in the case where the x and y axis are the same, the scattering plot is a line. Since this case was obvious, and drawing it does not give extra information regarding data behavior, it is not depicted.

In addition, based on the above plots, it is clear that using *Petal Width* and *Petal Length* features, help us separate these features linearly. (The same result was also cocluded in hands on exercise as well).

8.2 Part B

In this section, what we first are required to do, is to implement a fucntion that splits the dataset into train data, train labels, test data, and test labels. To do this task, based on the fact that 80% of data is used as train and the remaining 20% for test, 20 random indexes are generated and the splitting is carried out.

As the next step, a nearest neighbor classifier is created. To simulate this classifier, the euclidan distance is implimented. Meaning, each test data's euclidean distance is calcualted with respect to each train data. The train data which is nearest to the test data, share the same class. The same procedure is carried out for all test data so that their labels (classes) are found.

In this step, two scenarios are considered.

- *Not Normalized Data*: In this case, the separed data are given to the classifer, and the evaluated accuracy is 95%.
- *Normalized Data*: In order to normalize data, the mean value of each feature is removed from it (the result is biased free), then the result is divided to each feature's standard deviation. As a result, each feature is normalized, meaning its mean is zero and its standard deviation is one. In the next step, the normalized data is given to the classifier. However, in this case the accuracy decreases to 90%.

The reason responsible for this decreament lies on the way KNN classifier works. Suppose a test data is the point we are trying to classify into either of the three classes. For the first case, let us assume we have not normalized any of the data. For instance, we assume that it is closer to *Iris-virginica* class more than the two others.

Let us assume it in fact was the accurate label as well.

Now, to discuss normalization. Normalization is a way of taking data that is slightly dissimilar but giving it a common state. Assume in the above example that you normalize the test data's features, and therefore the output y's value becomes less. This would place the particular test data be discussed below it's current position and surrounded by more of another class, for instance, *Iris-versicolor* than its true label. Therefore, our algorithm would label it as Iris-versicolor which is inaccurate.

This matter may take place due to the fact that in reality some features are more essential in detecting a flower's type rather than others. (This matter was already discussed in hands on section) Therefore, by normalization, we neglect this importance and end-up to an inaccurate label.

Howoever, there is no general rule as normalization is a useless task an its demerits outweights its merits. Sometimes, normalizing data removes important feature differences, (our case) causing accuracy to go down. Other times, it helps to eliminate the noise in features which cause incorrect classifications. Also, just because accuracy goes up for the dataset we are currently working with, doesn't mean we will get the same results with a different data set.

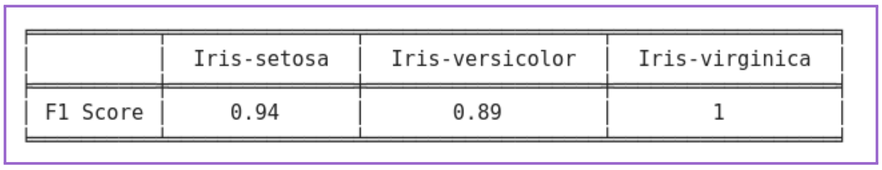
Long story short, instead of trying to label normalization as good/bad, we need to consider the feature inputs we are using for classification, and determine which ones are important to our model, and make sure differences in those features are reflected accurately in your classification model.

8.3 Part C

For this part of problem, confusion matrix, and f_1 *score* are implemented. (Note that accuracy was calculated in the last part for both normalized and non-normalized case.) It's value for the nomalized case was 90%. In order to calculate the f_1 *score* the below equations is used.

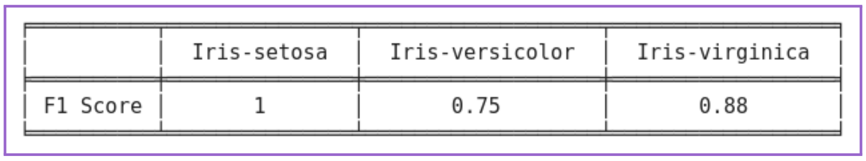
$$f_1\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$

The results of these calculations can be depicted as below:



	Iris-setosa	Iris-versicolor	Iris-virginica
F1 Score	0.94	0.89	1

FIGURE 8.13: f_1 *score* of Non-Normalized Data



	Iris-setosa	Iris-versicolor	Iris-virginica
F1 Score	1	0.75	0.88

FIGURE 8.14: f_1 *score* of Normalized Data

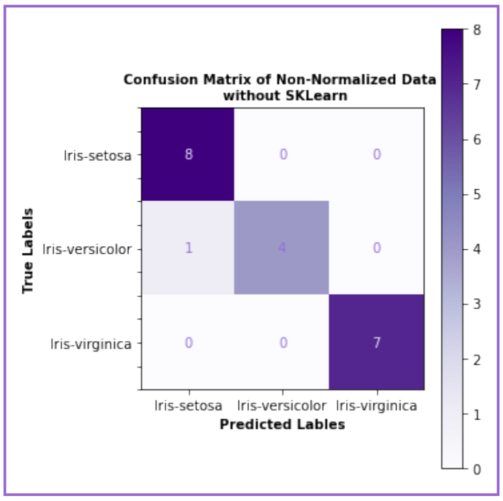


FIGURE 8.15: Confusion Matrix of Non-Normalized Data

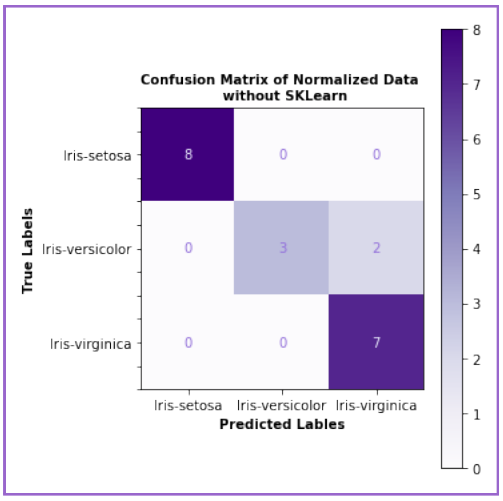
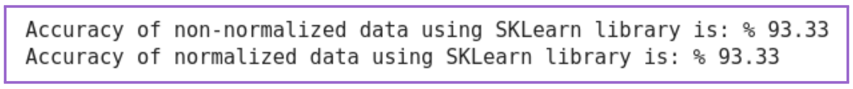


FIGURE 8.16: Confusion Matrix of Normalized Data

The above confusion matrix suggests that although normalization helped us not to detect *Iris-versicolor* as *Iris-setosa*, *Iris-versicolor* is now being detected as *Iris-virginica* more often. Thus, the decreased accuracy can be noticed in the alternations in confusion matrix as well.

8.4 Part D

The same procedure is carried out with implimented libraries. Note that the results do differ with the last part, due to the fact that the spliting randomization alters between the two implimentations, and thus the changes in accuracy, confusion matrix and f_1 *score* is expected.



```
Accuracy of non-normalized data using SKLearn library is: % 93.33
Accuracy of normalized data using SKLearn library is: % 93.33
```

FIGURE 8.17: Accuracy of Two Cases

The above result for accuracy proves the made point regarding the fact that there is no general rule about whether normalization is beneficial or not.

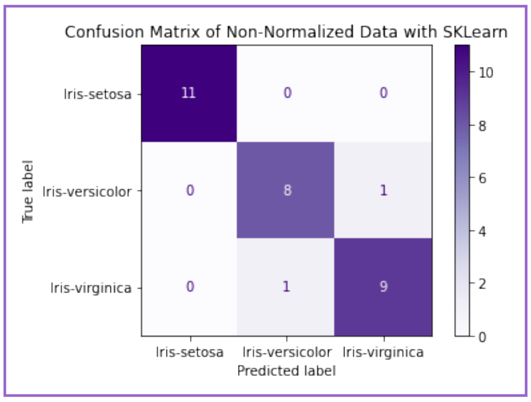


FIGURE 8.18: Confusion Matrix of Non-Normalized Data

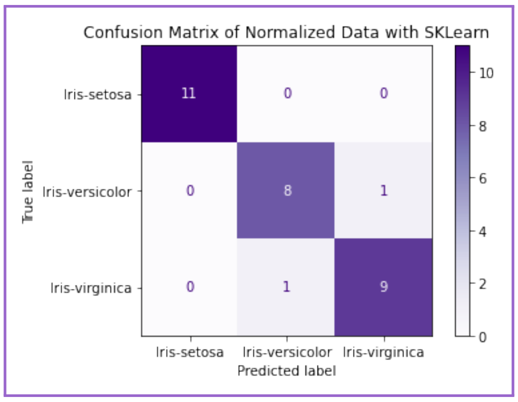
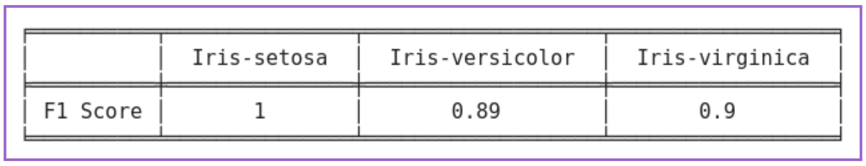
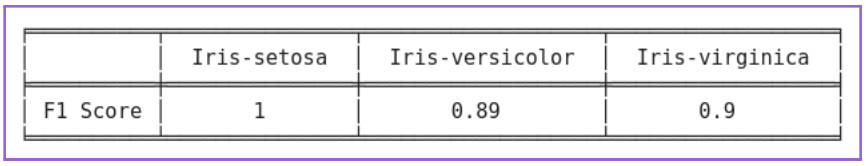


FIGURE 8.19: Confusion Matrix of Normalized Data



	Iris-setosa	Iris-versicolor	Iris-virginica
F1 Score	1	0.89	0.9

FIGURE 8.20: f_1 *score* of Non-Normalized Data



	Iris-setosa	Iris-versicolor	Iris-virginica
F1 Score	1	0.89	0.9

FIGURE 8.21: f_1 *score* of Normalized Data

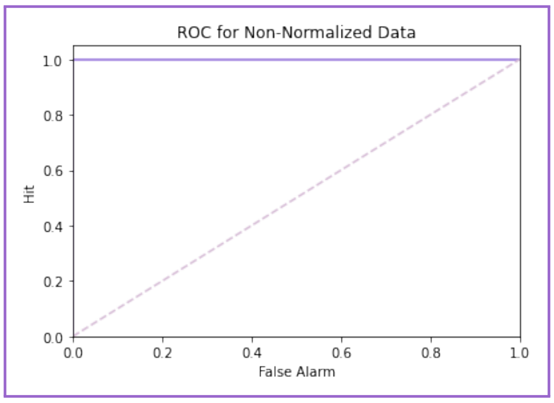


FIGURE 8.22: ROC of Class *Iris-setosa* vs Others for Non-Normalized Data

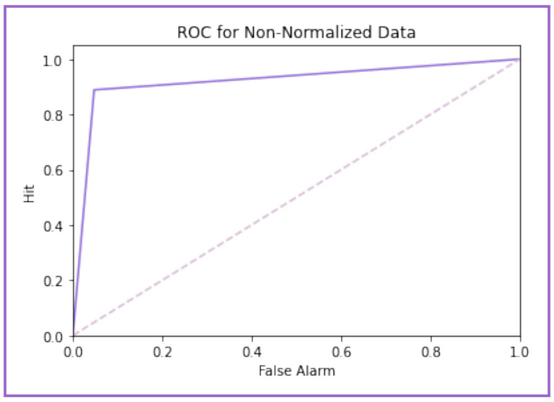


FIGURE 8.23: ROC of Class *Iris-versicolor* vs Others for Non-Normalized Data

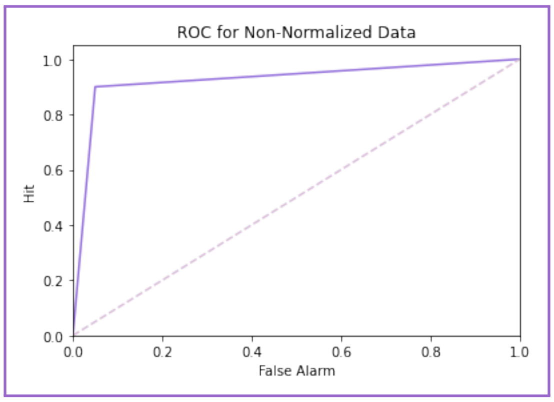
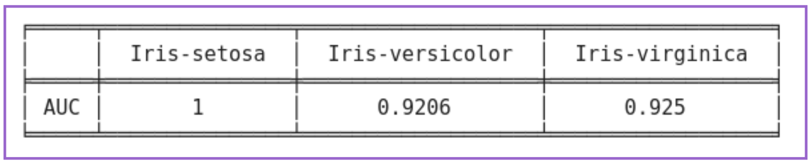


FIGURE 8.24: ROC of Class *Iris-virginica* vs Others for Non-Normalized Data



	Iris-setosa	Iris-versicolor	Iris-virginica
AUC	1	0.9206	0.925

FIGURE 8.25: AUC of Non-Normalized Data

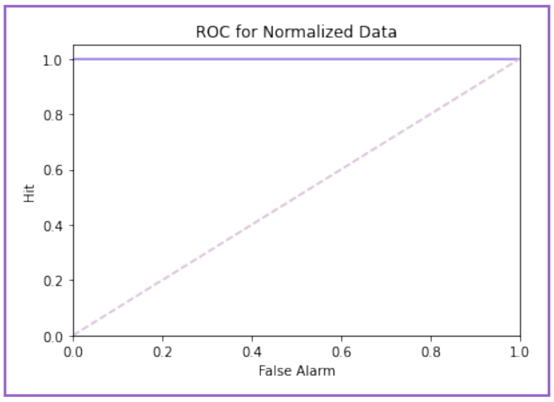


FIGURE 8.26: ROC of Class *Iris-setosa* vs Others for Normalized Data

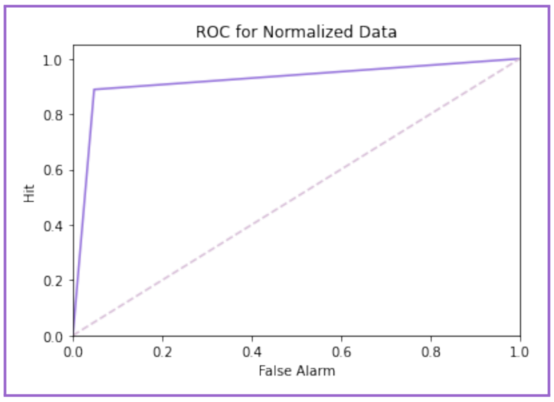


FIGURE 8.27: ROC of Class *Iris-versicolor* vs Others for Normalized Data

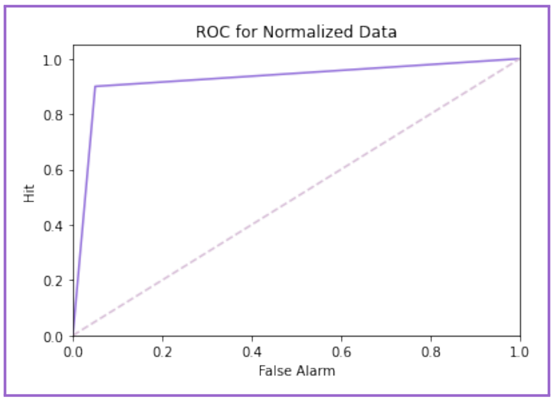
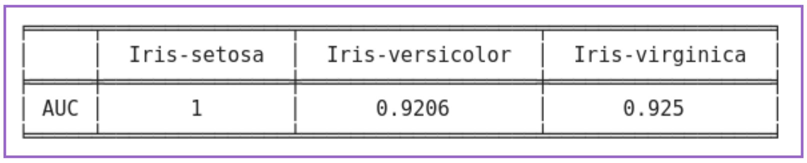


FIGURE 8.28: ROC of Class *Iris-virginica* vs Others for Normalized Data



	Iris-setosa	Iris-versicolor	Iris-virginica
AUC	1	0.9206	0.925

FIGURE 8.29: AUC of Normalized Data

Chapter 9

Question 9

9.1 Part A

9.1.1 Description

- **Naive Bayesian Classifier**:

  They are a collection of classifiacation algorithms based on Bayes' Theorem. Bayes' Theorem finds the probability of an event occuring given the probability of another event that has already occured. Mathematically speakinga: (Another way of representing this theorem is discussed in Optimal Bayes Classifier section.)

  $$P(y|\mathbf{X}) = \frac{P(\mathbf{X}|y)P(y)}{P(\mathbf{X})}$$

  where y is class variable and $\mathbf{X}$ is a feature vector of size n, i.e. $\mathbf{X} = (x_1, x_2, ..., x_n)$. The fundamental Naive Bayese assumption is that each feature makes an independent and equal contribution to the outcome. Owing to the fact that such assumptions are not generally true in real world, one may think less of them, while it is not true.

  Keeping the stated assumptions in mind, we get:

  $$P(y|x_1, ...x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

  On the grounds of the fact that the denominator is the same for all, and has a constant value, it can be eliminated. Therefore, we get:

  $$P(y|x_1, ...x_n) \alpha P(y) \prod_{i=1}^{n} P(x_i|y)$$

  Given various values of y, the one that maximizes the above equation is our goal. In mathemathical form:

  $$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

  Therefore, based on the distribution of the conditional probability ($P(x_i|y)$), various kind of these classifiers are created. What we are asked to assume in this problem, is that $P(x_i|y)$ are normally distributed, or:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2})$$

Based on the given description, the Gaussian Naive Bayesian Classifier is implimented.

- **Optimal(Non-Naive) Bayesian Classifier**:
  It is a probabilistic model that makes the most probable prediction for a new example, given the training dataset. Meaning, in general, the most probable classification of the new instance is obtained by combining the predictions of all hypotheses, weighted by their posterior probabilities. Mathematically speaking, for a new instance x_j that requires to be classified:

  $$P(x_j|D) = \sum_{h \in H} P(x_j|h_i)P(h_i|D)$$

  (Note that the above formulation is another form of representing Bayes' Theorem and does not differ from the one stated before.)where H is the set of hypothesis for classifing any incoming instance, h_i is a given hypothesis, $P(x_j|h_i)$ is the posterior probability of x_j given hypothesis h_i, and $P(h_i|D)$ is the posterior probability of hypothesis h_i given the data D.

  Any model classifing each instance by the above equation is a Bayes optimal classifier and no other model can outperform this algorithm, on average. (That is why this algorithm carries on "optimal" as its bold attribute.)

  As the above explanation suggest, this algorithm can have collossal computational cost. Therefore, some assumptions are made to simplify this method, such as independece of features, represented in Naive Bayes, discribed above. In other words, Naive Bayes classifier is a subset of Optimal Bayes classifier.

9.1.2 Implimentation

- **Naive Bayesian Classifier**:
  To implement this algorithm, first of all, a variance smoothing parameter is set, which its role is explained furthur.

  The second step is to fit a model to the train data and its labels. To do so, after the initalization of train data and its labels, the classes are separeted based on their label and each of their variance and expected value is calculated and stored.

  Afterward, a classifier needs to be created. To do so, for each lable, the prior probability is calculated. Note that one of the parameters of the *GaussianNB* implemented by "sklearn" library, is the prior probabilites which are calculated by the train data as well. Now, the likelihood function is calculated. The variance smoothing's role is critical in this part. Owing to the fact that for one data, varicance is zero, to prevent likelihood to be infinity, a variace smoothig is added to varince. The value of it can be set at the time of initialization of the class, same as what is implimented in *GaussianNB*.

  The last part is to find the maximum posterior for each label, and the prediction is done.

- **Optimal(Non-Naive) Bayesian Classifier**:
  To implement this algorithm, first of all, a variance smoothing parameter

and the method of fixing a singular matrix is set. The prominant role of both of these procedures are explained furthur.

The second step, same as the last algorithm, to fit a model to the train data and its labels. To do so, after the initalization of train data and its labels, the mean and covariance of classes combined to one another is calculated. Note that in contrast to the last algorithm, since the features are not independent, they cannot be separated.

Afterward, a classifier needs to be created. To do so, for each lable, the prior probability is calculated. Now, the likelihood function is calculated. The variance smoothing and method's role are critical in this part. As is known, in order to calculate the likelihood for the mulivariate Gaussian case, we need to find the inverse of the covariance matrix, as well as its determinant. However, not all matrices are invertible. They are non-invertible due to the fact that their determinant is zeros. In order to fix this matter, two methods are implemented.

– **Employing Eigen Values:**
  In this procedure, the eigen values of covariance matrix are calculated. Afterward, the ones which are more than a threshold are selected and their product is counted as determinant of the covariance matrix. For its inverse, pseudo inverse is calculated as well.

– **Employing a Smoothing Value:**
  Same as what was done in Naive Bayes algorithm, when varianve was zero, in this procedure a values is added to the covariance matrix to fix it. Afterward, its inverse and determinant and pseudo inverse is calculated.

Both of these methods have merits and dimerits, therefore, both are implimented. The last part is to find the maximum posterior for each label, and the prediction is done.

9.2 Part B

In order to gain insight of the dataset we are dealing with, one of its handwritings are depicted as below:

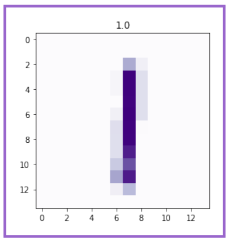
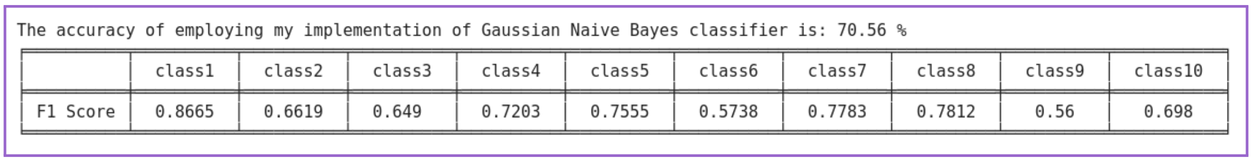


FIGURE 9.1: Plot of One Data and Its Label

Now both of the implemented classifiers are employed on this data set. As was described in the last section, in order to prevent likelihood function to be infinity, a smoothing parameter is added in both of the algorithms. Without any other process on data, the result is:

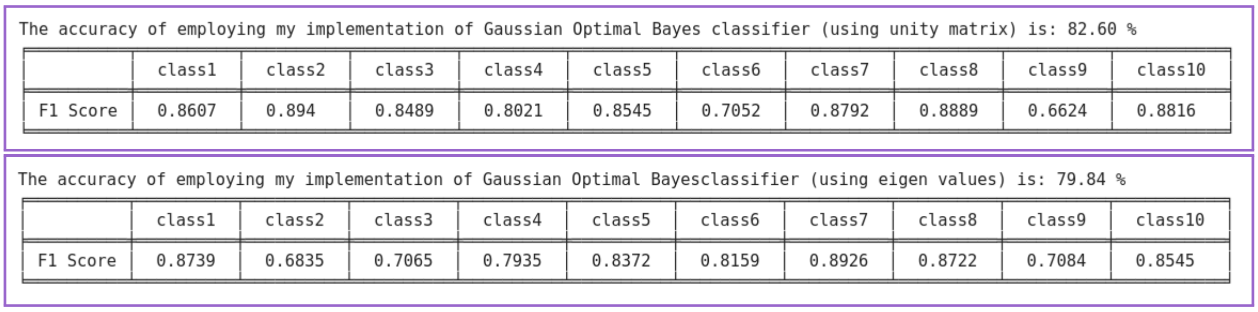
- **Naive Bayesian Classifier**:



```
The accuracy of employing my implementation of Gaussian Naive Bayes classifier is: 70.56 %
|          | class1 | class2 | class3 | class4 | class5 | class6 | class7 | class8 | class9 | class10 |
| F1 Score | 0.8665 | 0.6619 | 0.649  | 0.7203 | 0.7555 | 0.5738 | 0.7783 | 0.7812 | 0.56   | 0.698   |
```

Figure 9.2: Results of Simulation

- **Optimal(Non-Naive) Bayesian Classifier**:



```
The accuracy of employing my implementation of Gaussian Optimal Bayes classifier (using unity matrix) is: 82.60 %
|          | class1 | class2 | class3 | class4 | class5 | class6 | class7 | class8 | class9 | class10 |
| F1 Score | 0.8607 | 0.894  | 0.8489 | 0.8021 | 0.8545 | 0.7052 | 0.8792 | 0.8889 | 0.6624 | 0.8816  |
```

```
The accuracy of employing my implementation of Gaussian Optimal Bayesclassifier (using eigen values) is: 79.84 %
|          | class1 | class2 | class3 | class4 | class5 | class6 | class7 | class8 | class9 | class10 |
| F1 Score | 0.8739 | 0.6835 | 0.7065 | 0.7935 | 0.8372 | 0.8159 | 0.8926 | 0.8722 | 0.7084 | 0.8545  |
```

Figure 9.3: Results of Simulation

As the above results suggets, the accuracy in the *Optimal(Non-Naive) Bayesian* classifier is higher. As was discussed on the last section, in the *Naive Bayesian* classifier, we assume the features are independent to one another, and this assumption is not accurate in all cases. Therefore, the classification can have flaws. Although the accuracy in *Optimal(Non-Naive) Bayesian* classifier is higher, we should not neglect the fact that its computation cost is higher too. Therefore, in cases were we have limitations on this matter, *Naive Bayesian* classifier can be a wiser choice.

9.3 Part C

Same as the last part, first of all the dataset is depicted to give insights about what we are dealing with.

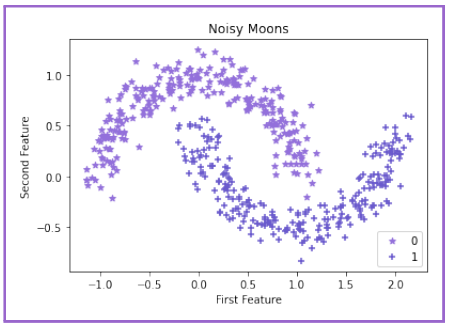
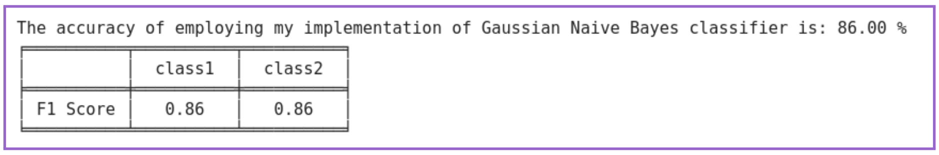


Figure 9.4: Plot of Noisy Moon Dataset

Now both of the implemented classifiers are employed on this data set. As was described in the last section, in order to prevent likelihood function to be infinity, a smoothing parameter is added in both of the algorithms. Without any other process on data, the result is:

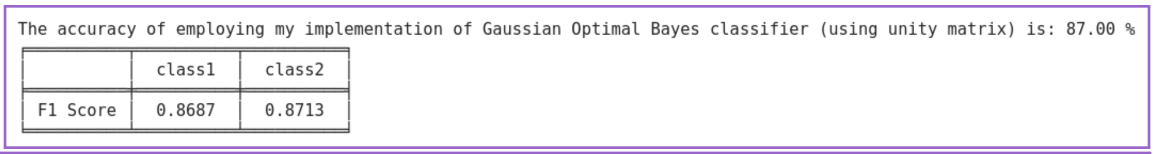
- **Naive Bayesian Classifier**



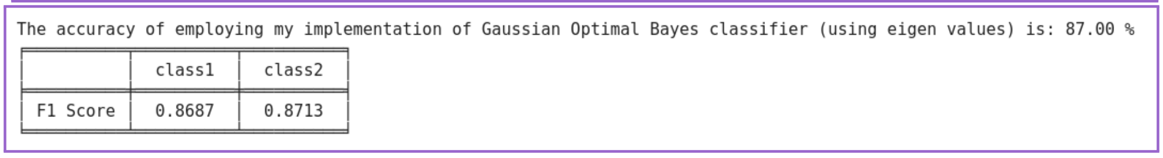
```
The accuracy of employing my implementation of Gaussian Naive Bayes classifier is: 86.00 %
|          | class1 | class2 |
| F1 Score |  0.86  |  0.86  |
```

FIGURE 9.5: Results of Simulation

- **Optimal(Non-Naive) Bayesian Classifier**



```
The accuracy of employing my implementation of Gaussian Optimal Bayes classifier (using unity matrix) is: 87.00 %
|          | class1 | class2 |
| F1 Score | 0.8687 | 0.8713 |
```



```
The accuracy of employing my implementation of Gaussian Optimal Bayes classifier (using eigen values) is: 87.00 %
|          | class1 | class2 |
| F1 Score | 0.8687 | 0.8713 |
```

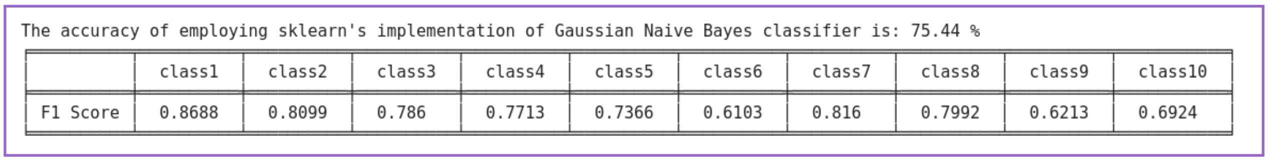
FIGURE 9.6: Results of Simulation

The same conclusion on the usage of which classifier holds for this dataset as well.

9.4 Part D

In this part "sklearn" library is employed to calculate the results. They are depicted as below. Note that in order to have comparable results, the variance smoothing in the *GaussianNB* is set same as the implimented algorithms.

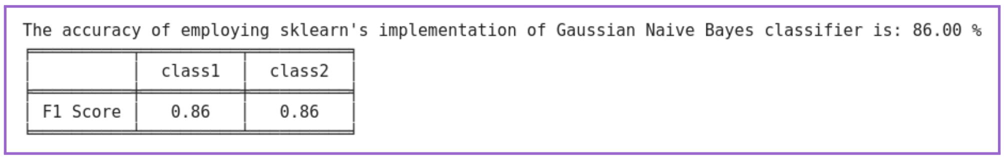
- **Tiny MNIST**



```
The accuracy of employing sklearn's implementation of Gaussian Naive Bayes classifier is: 75.44 %
|          | class1 | class2 | class3 | class4 | class5 | class6 | class7 | class8 | class9 | class10 |
| F1 Score | 0.8688 | 0.8099 | 0.786  | 0.7713 | 0.7366 | 0.6103 | 0.816  | 0.7992 | 0.6213 | 0.6924  |
```

FIGURE 9.7: Results of Simulation

- **Noisy Moons**



```
The accuracy of employing sklearn's implementation of Gaussian Naive Bayes classifier is: 86.00 %
╒══════════╤══════════╤══════════╕
│          │  class1  │  class2  │
╞══════════╪══════════╪══════════╡
│ F1 Score │   0.86   │   0.86   │
╘══════════╧══════════╧══════════╛
```

FIGURE 9.8: Results of Simulation

The same conclusion on the usage of which classifier holds for this dataset as well.

Chapter 10

Question 10

One Vs Rest:

As its name suggets, by employing this algorithm on any problem, the problem is transformed into a binary classificaion one. One vs. rest strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. (We can label the target class as one while the other ones are labeled as zero). Its pseudocode is as below:

```
Inputs:
  • $L$: a learner (training algorithm for binary classifiers)
  • samples $X$
  • labels $y$ where $y_i \in \{1, \dots K\}$ is the label for the sample $X_i$
Output:
  • a list of classifiers $f_k$ for $k \in \{1, \dots K\}$
Procedure:
  • for each $k$ in $\{1, \dots K\}$
      – Construct a new label vector $z$ where $z_i = y_i$ if $y_i = k$ and $z_i = 0$ otherwise
      – Apply $L$ to $X$, $z$ to obtain $f_k$
```

Now that the logic behind one vs rest strategy is clear, we need to gain some insights of the dataset we want to work on.

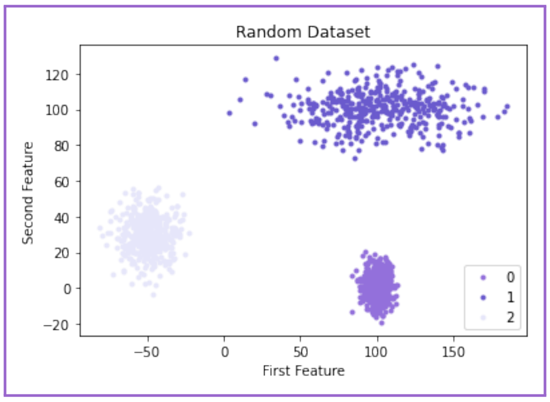


Figure 10.1: Plot of Random Dataset

As we know, logistic regression is a method in which a classification problem is seen through the lens of a regression one, while ignoring the fact that the labels are discrete. Mathemathically speaking:

$$y \in \{0,1\},\ h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \rightarrow P\{y=1|x;\theta\} = h_\theta(x),\ P\{y=0|x;\theta\} = 1 - h_\theta(x)$$

Therefore, the log-likelihood function is: (cross entropy)

$$log\{L(\theta)\} = \sum_{i=1}^{n} y^{(i)} log\{h(x^{(i)})\} + (1-y^{(i)}) log\{1 - h(x^{(i)})\}$$

Hence, in order to update weights, the gradient of the log-likelihood needs to be evaluated with respect to θ:

$$\theta_j := \theta_j + \alpha[y^{(i)} - h_\theta(x^{(i)})]x_j^{(i)}$$

As the above equations suggest, we employ gradient ascent to maximize the log-likelihood function which is analogous to minimizing the cost function.

If the above equations and algorithms are employed, the result is:

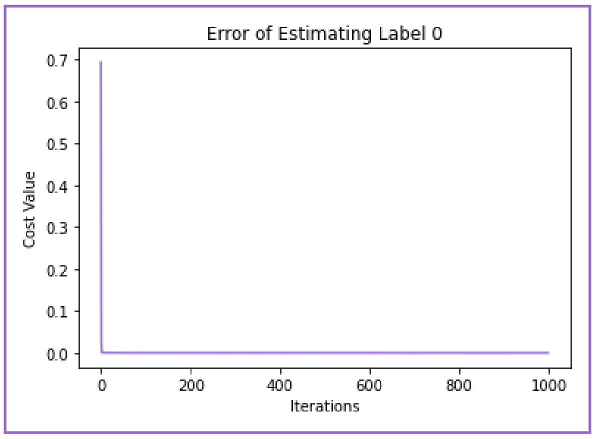


FIGURE 10.2: Plot of Cost Function During Trials

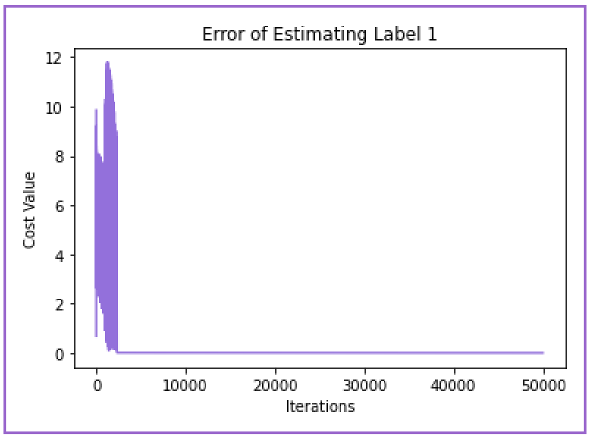


FIGURE 10.3: Plot of Cost Function During Trials

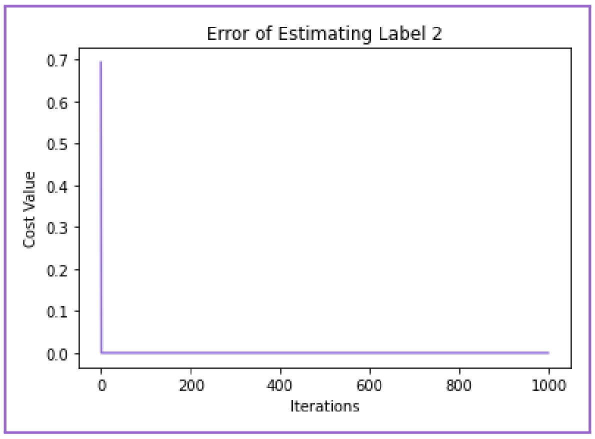


FIGURE 10.4: Plot of Cost Function During Trials

By employing the above algorithms, weights and bias are calculated. Therefore, we can depict lines that separate these classes by imploying them. If we do so, the result is:

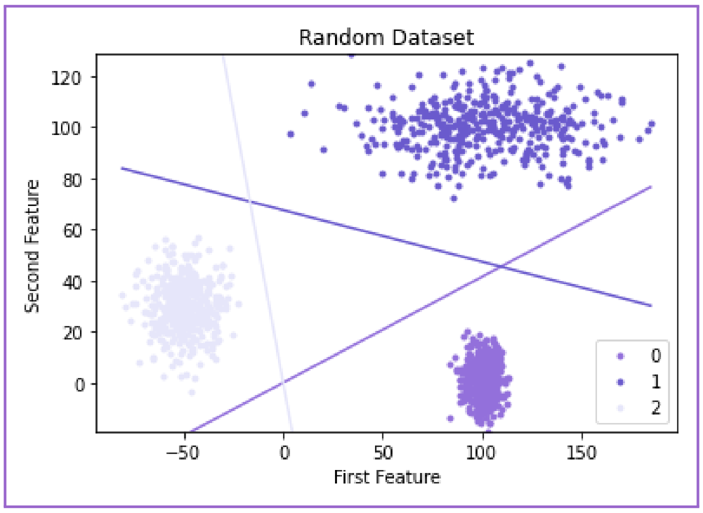


FIGURE 10.5: Plot of Random Dataset and Separating Lines

Althoguh the blank space created between the lines show that one vs. rest has some flaws!

Chapter 11

Question 11

Before doing anything, data is plotted to gain an insight of what we are dealing with.

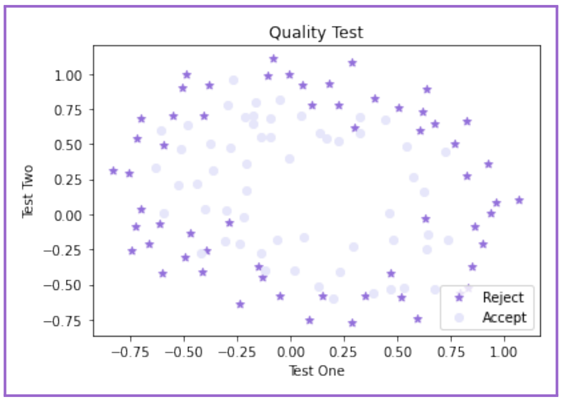
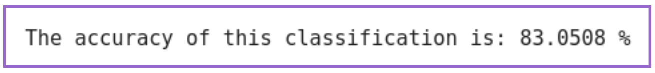


FIGURE 11.1: Plot of Quality Test Dataset

Secondly, the increament in dimension takes place. In order to do so, given the implimented functions, $f(X)$ is created. By increasing their dimension, we are trying to increase our level of freedom so that we are able to linearly separate these two from one another.

As the next step, logistic regression is emplyed on this multi-dimensional dataset, and the new labels are predicted. A comparision between the predicted labels, and the actual ones is represented by its accuracy.

Before doing anything, data is plotted to gain an insight of what we are dealing with.



```
The accuracy of this classification is: 83.0508 %
```

FIGURE 11.2: How Accurate the Implimented Model Is

As the next step, what we need to do, is to use the classifier's parameters to gain insights of the coefficients and bias of the created model. By knowing these values, we can achieve our initial goal. After the extraction of these values, the next step is to create a curve using data and these coefficients. However, we need the dimensions to follow the same pattern as our data in the last step.(based on the given formulation for $f(X)$) To do so, the same function is used, and then the result is scaled based on the coefficients of the created model. (These values include bias and the slops) After plotting the created model we get:

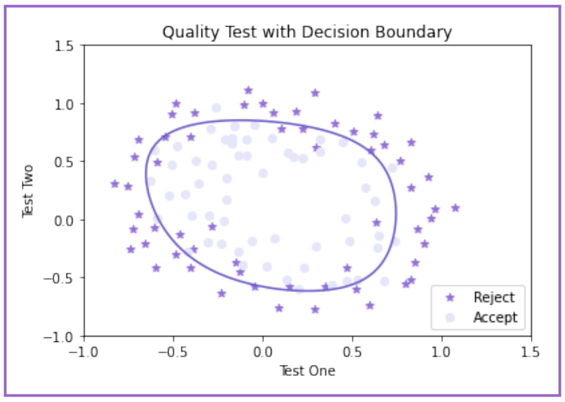


FIGURE 11.3: Plot of Quality Test Dataset and Decision Boundary

As can be seen, the two classes are separated by a boundary. However, there are some errors which also proves why the accuracy is not 100%.