1

The Bayes Theorem

2

The Bayes Classifier

3

Maximum Likelihood Estimation (MLE)

4

Linear Discriminant Analysis (LDA)

5

Quadratic Discriminant Analysis (QDA)

The 5-step approach to our solution case

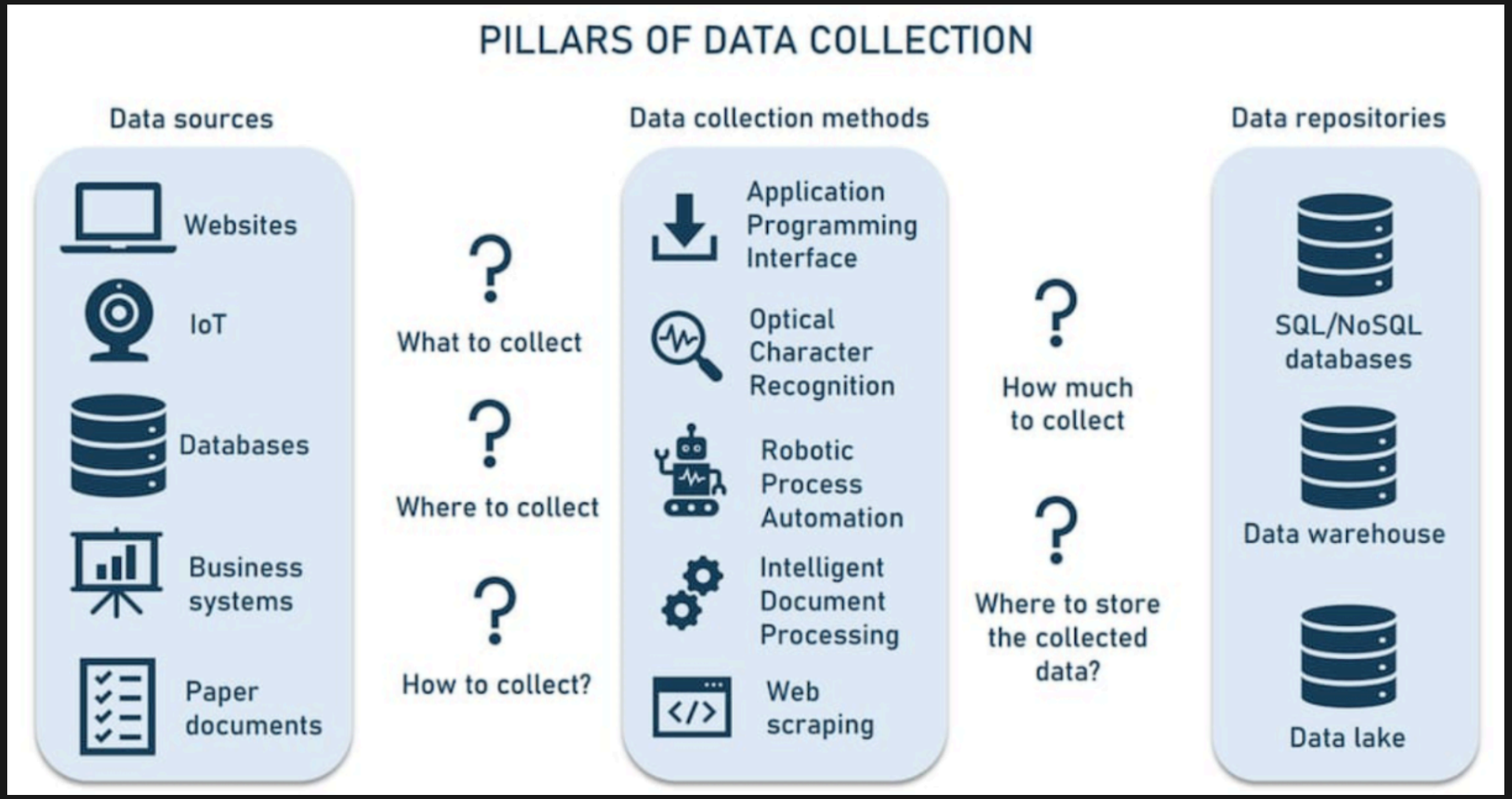
JPMC QMP DATA SCIENCE CASE STUDY

Group Number 17 - Anupriya (QMP 36), Saiuditi (QMP 34)

METHODOLOGY IN MACHINE LEARNING

Data Collection:	Data collection is a methodical practice to acquire meaningful information to build a consistent and complete dataset.	**Model Training:**	Training a model involves fitting the model to the data such that it learns the underlying patterns and relationships.
Data Cleaning and Preprocessing:	This step involves cleaning the collected data to handle missing values, outliers, and inconsistencies.	**Validation:**	The trained model is evaluated on a separate validation dataset to assess its performance and generalization ability.
Feature Selection:	Feature selection is a process in machine learning where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested.	**Testing:**	Performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC are calculated to measure the model's effectiveness.
Model Selection:	A suitable classification algorithm is chosen based on the problem requirements, data characteristics, and computational resources.	**Model Deployment:**	Once the model has been trained and validated, it is deployed into production to make predictions on new, unseen data.

DATA COLLECTION:



Articulating exactly what you want to know is the starting point for a data collection journey. From here, we will have to take the next steps.

- Define what information we need to collect.
- Find sources of relevant data.
- Choose data collection methods and tools.
- Decide on a sufficient data amount.
- Set up data storage technology.

DATA SIMULATION IN THE CASE STUDY

```python
# simulating data for 2c and testing MLE
# saving data in the next 3 cells

n_samples = 1000
num_classes = 4

mu_1_2c = np.array([2, 2])
mu_2_2c = np.array([2, -2])
mu_3_2c = np.array([-2, -2])
mu_4_2c = np.array([-2, 2])

sigma_1_2c = np.array([[5, 2], [2, 4]])
sigma_2_2c = np.array([[1, -1.5], [-1.5, 3]])
sigma_3_2c = np.array([[6, -3], [-3, 3]])
sigma_4_2c = np.array([[4, 0], [0, 9]])


X_1_2c = np.random.multivariate_normal(mu_1_2c, sigma_1_2c, n_samples)
X_2_2c = np.random.multivariate_normal(mu_2_2c, sigma_2_2c, n_samples)
X_3_2c = np.random.multivariate_normal(mu_3_2c, sigma_3_2c, n_samples)
X_4_2c = np.random.multivariate_normal(mu_4_2c, sigma_4_2c, n_samples)


Y_1_2c = np.ones(n_samples)
Y_2_2c = 2* np.ones(n_samples)
Y_3_2c = 3* np.ones(n_samples)
Y_4_2c = 4* np.ones(n_samples)

X_2c = np.concatenate((X_1_2c, X_2_2c, X_3_2c, X_4_2c))
Y_2c = np.concatenate((Y_1_2c, Y_2_2c, Y_3_2c, Y_4_2c))

X_1_2c_test = np.random.multivariate_normal(mu_1_2c, sigma_1_2c, n_samples)
X_2_2c_test = np.random.multivariate_normal(mu_2_2c, sigma_2_2c, n_samples)
X_3_2c_test = np.random.multivariate_normal(mu_3_2c, sigma_3_2c, n_samples)
```

```python
np.savez("dataset_2c", X_train= X_train, Y_train=Y_train, X_test=X_test, Y_test=Y_test)

mu_2c = []

for k in range(1, num_classes+1):
  mu_k = np.zeros(np.shape(X_2c[Y_2c == k][0]))

  for x in X_2c[Y_2c == k]:
    mu_k += x

  mu_k/= np.shape(X_2c[Y_2c == k])[0]
  mu_2c.append(mu_k)

print(mu_2c)

pi_2c = []

for k in range(1,num_classes+1):
  pi_k = np.shape(X_2c[Y_2c == k])[0]/np.shape(X_2c)[0]
  if(pi_k ==0):
    pi_k = 0.00001
  pi_2c.append(pi_k)


cov_2c = []
for k in range(1, num_classes+1):
  cov_k = np.zeros(np.shape(np.outer(X_2c[0] - mu_2c[k-1], (X_2c[0] - mu_2c[k-1]).T)))

  for x in X_2c[Y_2c == k]:
    cov_k+= np.outer((x - mu_2c[k-1]), (x - mu_2c[k-1]).T)
  cov_k/= np.shape(X_2c[Y_2c == k])[0]
  cov_2c.append(cov_k)

print(cov_2c)
```

DATA CLEANING AND PREPROCESSING:

Common Data Cleaning Techniques

Handling Missing Values

Imputation

Deletion

Removing Duplicates

Data Type Conversion

Outlier Detection

Common Data Preprocessing Techniques

Data Scaling

Encoding Categorical Variables

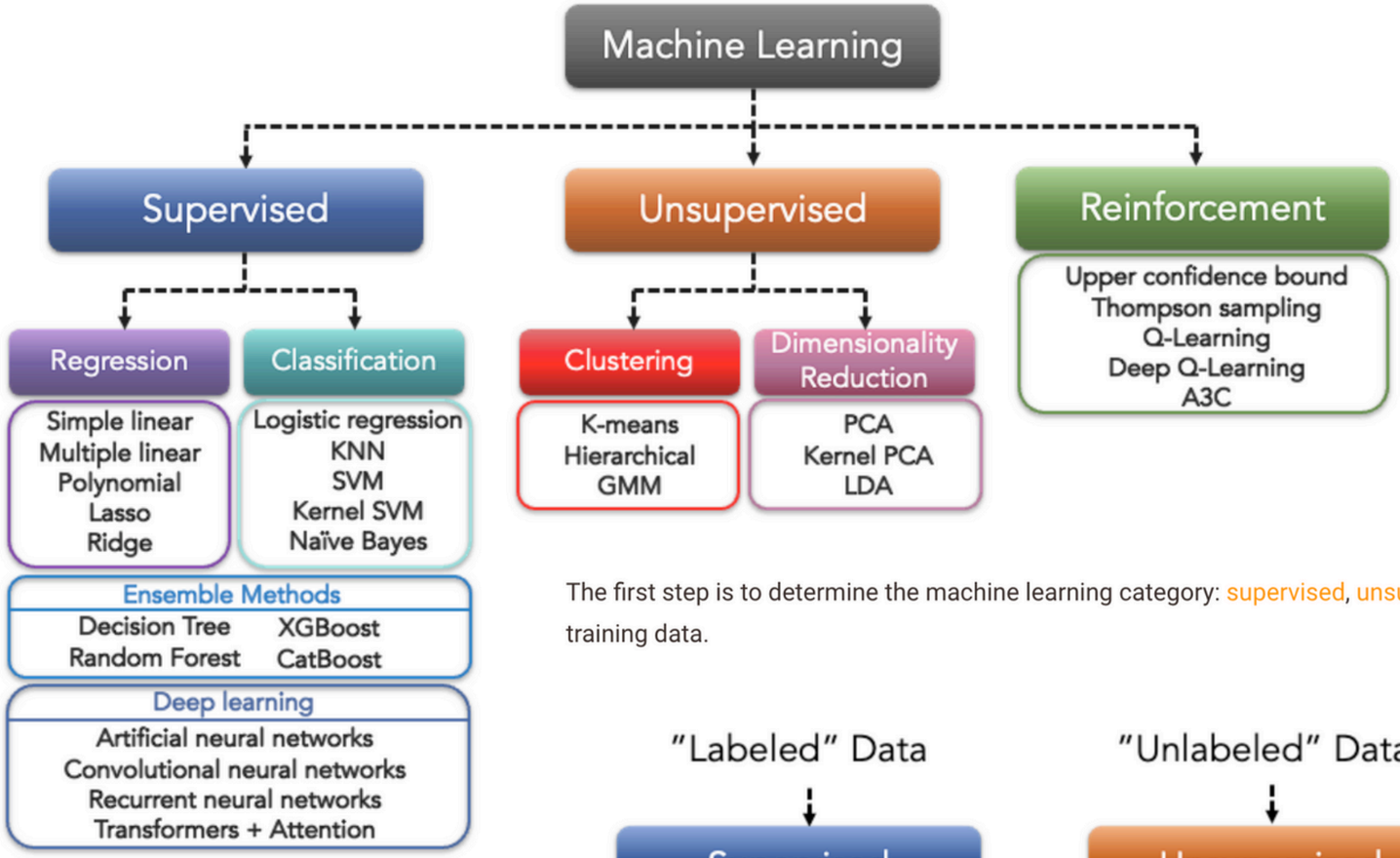
Data Splitting

Handling Missing Values

Data Cleaning Process

MODEL SELECTION:

2 Next, we determine the sub-task, which could be classification or regression for supervised learning, or clustering & dimensionality reduction for unsupervised.



1

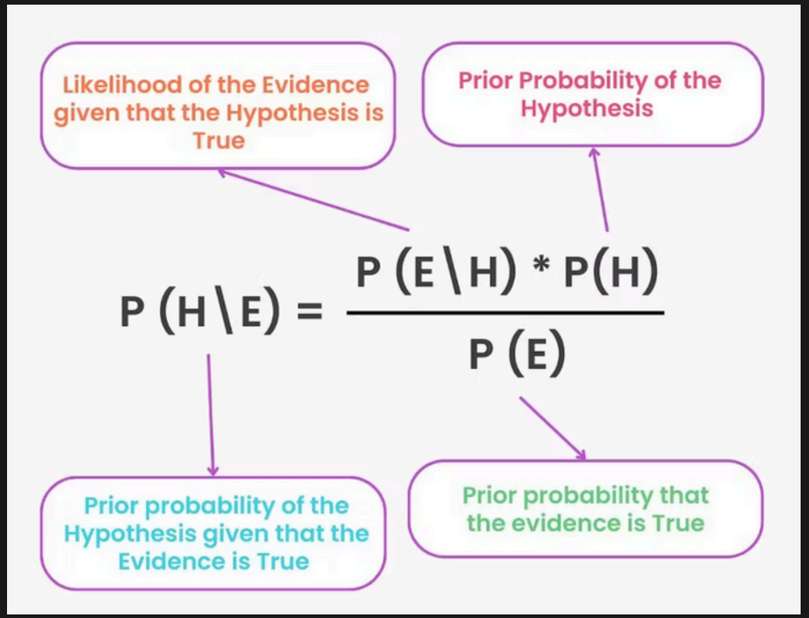
The first step is to determine the machine learning category: supervised, unsupervised, or reinforcement learning, based on our training data.

WHAT IS GAUSSIAN BAYES CLASSIFIER?

An Intuitive Approach

Gaussian Bayes is a machine learning classification technique based on a probablistic approach that assumes each class follows a normal distribution. It assumes each parameter has an independent capacity of predicting the output variable. It is able to predict the probability of a dependent variable to be classified in each group.
The combination of the prediction for all parameters is the final prediction that returns a probability of the dependent variable to be classified in each group. The final classification is assigned to the group with the higher probability.

$$\text{posterior probability} = \frac{(\text{conditional probability})(\text{prior probability})}{\text{evidence (a. k. a. “stabilizer”)}}$$



WHAT IS GAUSSIAN BAYES CLASSIFIER?

Given a features vector X=(x1,x2,...,xn) and a class variable y, Bayes Theorem states that:

We're interested in calculating the posterior probability P(y | X) from the likelihood P(X | y) and prior probabilities P(y),P(X).

We're interested in calculating the posterior probability P(y | X) from the likelihood P(X | y) and prior probabilities P(y),P(X).

Using the chain rule, the likelihood P(X | y) can be decomposed as:

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

$$P(X|y) = P(x_1|y) * P(x_2|y) \ldots . P(x_n|y)$$

but because of the Naive's conditional independence assumption, the conditional probabilities are independent of each other.

Thus, by conditional independence, we have:

$$P(X|y) = P(x_1, x_2, \ldots, x_n|y)$$

$$= P(x_1|x_2, \ldots, x_n, y) * P(x_2|x_3, \ldots, x_n, y) \ldots . P(x_n|y)$$

WHAT IS GAUSSIAN BAYES CLASSIFIER?

$$P(y|X) = \frac{P(x_1|y)*P(x_2|y) \ldots\ldots P(x_n|y) * P(y)}{P(x_1) * P(x_2) \ldots . P(x_n)}$$

And as denominator remains constant for all values, the posterior probability can then be:

$$P(y|x_1, x_2, \ldots\ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

The Naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that's most probable; this is known as the maximum a posteriori or MAP decision rule.

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

WHAT IS MAXIMUM LIKELIHOOD ESTIMATION (MLE)?

Let us try to understand the intuition and mathematics behind MLE and the principal of maximum likelihood. We used CS229 notes to understand MLE and learned how to apply it to our problem.

When faced with a regression problem, why might linear regression, and specifically why might the least-squares cost function J, be a reasonable choice? In this section, we will give a set of probabilistic assumptions, under which least-squares regression is derived as a very natural algorithm.

Let us assume that the target variables and the inputs are related via the equation

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)},$$

MAXIMUM LIKELIHOOD ESTIMATION (MLE)

where $\epsilon^{(i)}$ is an error term that captures either unmodeled effects (such as if there are some features very pertinent to predicting housing price, but that we'd left out of the regression), or random noise. Let us further assume that the $\epsilon^{(i)}$ are distributed IID (independently and identically distributed) according to a Gaussian distribution (also called a Normal distribution) with mean zero and some variance σ^2. We can write this assumption as "$\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$." I.e., the density of $\epsilon^{(i)}$ is given by

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right).$$

This implies that

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right).$$

The notation "$p(y^{(i)}|x^{(i)};\theta)$" indicates that this is the distribution of $y^{(i)}$ given $x^{(i)}$ and parameterized by θ. Note that we should not condition on θ ("$p(y^{(i)}|x^{(i)},\theta)$"), since θ is not a random variable. We can also write the distribution of $y^{(i)}$ as $y^{(i)} \mid x^{(i)};\theta \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$.

MAXIMUM LIKELIHOOD ESTIMATION (MLE)

Given X (the design matrix, which contains all the $x^{(i)}$'s) and θ, what is the distribution of the $y^{(i)}$'s? The probability of the data is given by $p(\vec{y}|X;\theta)$. This quantity is typically viewed a function of $\vec{y}$ (and perhaps X), for a fixed value of θ. When we wish to explicitly view this as a function of θ, we will instead call it the **likelihood** function:

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X;\theta).$$

Note that by the independence assumption on the $\epsilon^{(i)}$'s (and hence also the $y^{(i)}$'s given the $x^{(i)}$'s), this can also be written

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right). \end{aligned}$$

Now, given this probabilistic model relating the $y^{(i)}$'s and the $x^{(i)}$'s, what is a reasonable way of choosing our best guess of the parameters θ? The principal of **maximum likelihood** says that we should choose θ so as to make the data as high probability as possible. I.e., we should choose θ to maximize $L(\theta)$.

MAXIMUM LIKELIHOOD ESTIMATION (MLE)

Instead of maximizing $L(\theta)$, we can also maximize any strictly increasing function of $L(\theta)$. In particular, the derivations will be a bit simpler if we instead maximize the **log likelihood** $\ell(\theta)$:

$$\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2.
\end{aligned}$$

Hence, maximizing $\ell(\theta)$ gives the same answer as minimizing

$$\frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2,$$

which we recognize to be $J(\theta)$, our original least-squares cost function.

TESTING MLE ON OUR DATASETS

HOW CAN WE APPLY MLE TO OUR PROBLEM?

4.1.2 The Gaussian discriminant analysis model

When we have a classification problem in which the input features x are continuous-valued random variables, we can then use the Gaussian Discriminant Analysis (GDA) model, which models $p(x|y)$ using a multivariate normal distribution. The model is:

$$\begin{aligned} y &\sim \text{Bernoulli}(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu_1, \Sigma) \end{aligned}$$

Writing out the distributions, this is:

$$\begin{aligned} p(y) &= \phi^y(1-\phi)^{1-y} \\ p(x|y=0) &= \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right) \\ p(x|y=1) &= \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) \end{aligned}$$

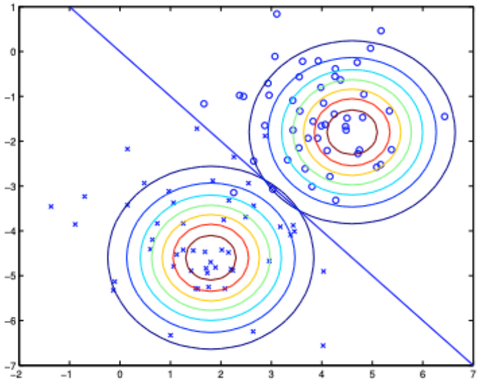
Here, the parameters of our model are ϕ, Σ, μ_0 and μ_1. (Note that while there're two different mean vectors μ_0 and μ_1, this model is usually applied using only one covariance matrix Σ.) The log-likelihood of the data is given by

$$\begin{aligned} \ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{n} p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\ &= \log \prod_{i=1}^{n} p(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi). \end{aligned}$$

By maximizing ℓ with respect to the parameters, we find the maximum likelihood estimate of the parameters (see problem set 1) to be:

$$\begin{aligned} \phi &= \frac{1}{n}\sum_{i=1}^{n} 1\{y^{(i)}=1\} \\ \mu_0 &= \frac{\sum_{i=1}^{n} 1\{y^{(i)}=0\}x^{(i)}}{\sum_{i=1}^{n} 1\{y^{(i)}=0\}} \\ \mu_1 &= \frac{\sum_{i=1}^{n} 1\{y^{(i)}=1\}x^{(i)}}{\sum_{i=1}^{n} 1\{y^{(i)}=1\}} \\ \Sigma &= \frac{1}{n}\sum_{i=1}^{n} (x^{(i)}-\mu_{y^{(i)}})(x^{(i)}-\mu_{y^{(i)}})^T. \end{aligned}$$

Pictorially, what the algorithm is doing can be seen in as follows:



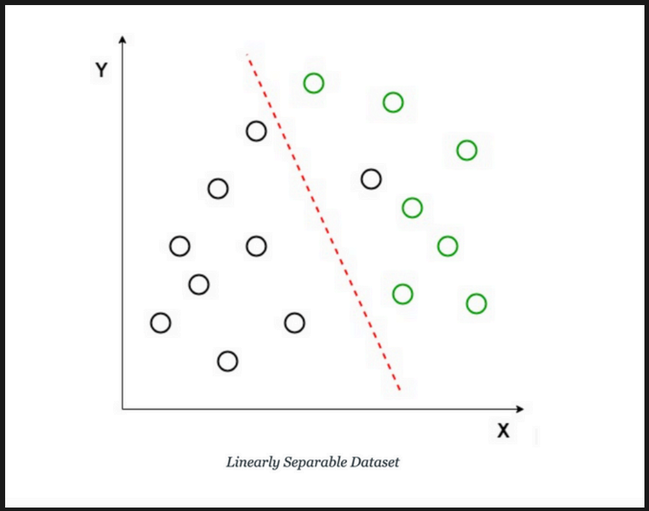
Shown in the figure are the training set, as well as the contours of the two Gaussian distributions that have been fit to the data in each of the two classes. Note that the two Gaussians have contours that are the same shape and orientation, since they share a covariance matrix Σ, but they have different means μ_0 and μ_1. Also shown in the figure is the straight line giving the decision boundary at which $p(y=1|x)=0.5$. On one side of the boundary, we'll predict $y=1$ to be the most likely outcome, and on the other side, we'll predict $y=0$.

WHAT IS LINEAR DISCRIMANT ANALYSIS?

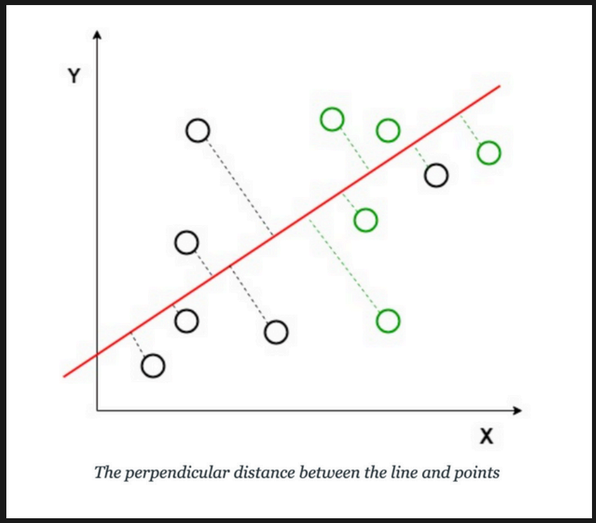
An Intuitive and Mathematical Approach

features from a higher-dimensional space into a lower-dimensional one. LDA assumes that the data has a Gaussian distribution and that the covariance matrices of the different classes are equal. It also assumes that the data is linearly separable, meaning that a linear decision boundary can accurately classify the different classes.

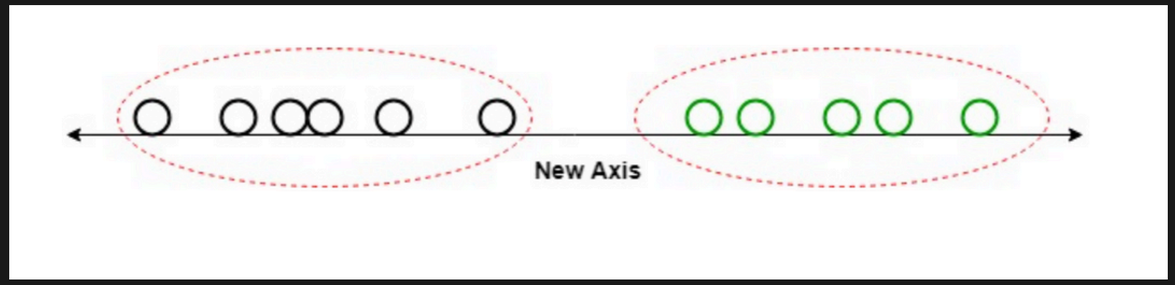
Suppose we have two sets of data points belonging to two different classes that we want to classify. As shown in the given 2D graph, when the data points are plotted on the 2D plane, there's no straight line that can separate the two classes of data points completely. Hence, in this case, LDA (Linear Discriminant Analysis) is used which reduces the 2D graph into a 1D graph in order to maximize the separability between the two classes.



Linearly Separable Dataset



The perpendicular distance between the line and points



WHAT IS QDA?

An Intuitive and Mathematical Approach

Since QDA estimates a covariance matrix for each class, it has a greater number of effective parameters than LDA. We can derive the number of parameters in the following way.

We need K class priors πk. Since $\sum Ki=1 \pi k=1$, we do not need a parameter for one of the priors. Thus, there are $K-1$ free parameters for the priors.

Since there are K centroids, μk, with p entries each, there are Kp parameters relating to the means.

From the covariance matrix, Σk, we only need to consider the diagonal and the upper right triangle. This region of the covariance matrix has $p(p+1)2$ elements. Since K such matrices need to be estimated, there are $Kp(p+1)2$ parameters relating to the covariance matrices.

Thus, the effective number of QDA parameters is $K-1+Kp+Kp(p+1)2$.

Since the number of QDA parameters is quadratic in p, QDA should be used with care when the feature space is large.

QDA, with its ability to embrace non-linear relationships, can capture intricate curves and contours in the data. It models class-conditional distributions using quadratic functions, offering greater flexibility in separating classes.

LDA FOR BINARY CLASSIFICATION

Now let us think of data as *multivariate* data with dimensionality d. The PDF for multivariate Gaussian distribution, $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is:

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\Big(-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{2}\Big), \tag{16}$$

3. Linear Discriminant Analysis for Binary Classification

In Linear Discriminant Analysis (LDA), we assume that the two classes have equal covariance matrices:

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}. \tag{18}$$

Therefore, the Eq. (17) becomes:

$$\begin{aligned}
&\frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\Big(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)}{2}\Big)\, \pi_1 \\
&= \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\Big(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)}{2}\Big)\, \pi_2, \\
&\implies \exp\Big(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)}{2}\Big)\, \pi_1 \\
&\quad = \exp\Big(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)}{2}\Big)\, \pi_2, \\
&\overset{(a)}{\implies} -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \ln(\pi_1) \\
&\quad = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) + \ln(\pi_2),
\end{aligned}$$

where (a) takes natural logarithm from the sides of equation.

We can simplify this term as:

$$\begin{aligned}
&(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) = (\boldsymbol{x}^\top - \boldsymbol{\mu}_1^\top)\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) \\
&= \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 \\
&\overset{(a)}{=} \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - 2\,\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x},
\end{aligned} \tag{19}$$

where (a) is because $\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 = \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x}$ as it is a scalar and $\boldsymbol{\Sigma}^{-1}$ is symmetric so $\boldsymbol{\Sigma}^{-\top} = \boldsymbol{\Sigma}^{-1}$. Thus, we have:

$$\begin{aligned}
&-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \ln(\pi_1) \\
&= -\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \ln(\pi_2).
\end{aligned}$$

Therefore, if we multiply the sides of equation by 2, we have:

$$\begin{aligned}
&2\left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\right)^\top \boldsymbol{x} \\
&\quad + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)) + 2\,\ln(\frac{\pi_2}{\pi_1}) = 0,
\end{aligned} \tag{20}$$

which is the equation of a line in the form of $\boldsymbol{a}^\top \boldsymbol{x} + b = 0$. Therefore, if we consider Gaussian distributions for the two classes where the covariance matrices are assumed to be equal, the decision boundary of classification is a line. Because of linearity of the decision boundary which discriminates the two classes, this method is named *linear discriminant* analysis.

For obtaining Eq. (20), we brought the expressions to the right side which was corresponding to the second class; therefore, if we use $\delta(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ as the left-hand-side expression (function) in Eq. (20):

$$\begin{aligned}
&\delta(\boldsymbol{x}) := 2\left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\right)^\top \boldsymbol{x} \\
&\quad + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{\ -1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)) + 2\,\ln(\frac{\pi_2}{\pi_1}),
\end{aligned} \tag{21}$$

the class of an instance $\boldsymbol{x}$ is estimated as:

$$\widehat{\mathcal{C}}(x) = \begin{cases} 1, & \text{if } \delta(\boldsymbol{x}) < 0, \\ 2, & \text{if } \delta(\boldsymbol{x}) > 0. \end{cases} \tag{22}$$

If the priors of two classes are equal, i.e., $\pi_1 = \pi_2$, the Eq. (20) becomes:

$$\begin{aligned}
&2\left(\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\right)^\top \boldsymbol{x} \\
&\quad + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) = 0,
\end{aligned} \tag{23}$$

whose left-hand-side expression can be considered as $\delta(\boldsymbol{x})$ in Eq. (22).

QDA FOR BINARY CLASSIFICATION

4. Quadratic Discriminant Analysis for Binary Classification

In Quadratic Discriminant Analysis (QDA), we relax the assumption of equality of the covariance matrices:

$$\boldsymbol{\Sigma}_1 \neq \boldsymbol{\Sigma}_2, \tag{24}$$

which means the covariances are not *necessarily* equal (if they are actually equal, the decision boundary will be linear and QDA reduces to LDA).

Therefore, the Eq. (17) becomes:

$$\begin{aligned}
&\frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}_1|}}\exp\Big(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)}{2}\Big)\,\pi_1\\
&=\frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}_2|}}\exp\Big(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)}{2}\Big)\,\pi_2,\\
&\overset{(a)}{\implies} -\frac{d}{2}\ln(2\pi)-\frac{1}{2}\ln(|\boldsymbol{\Sigma}_1|)\\
&\quad-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)+\ln(\pi_1)\\
&\quad=-\frac{d}{2}\ln(2\pi)-\frac{1}{2}\ln(|\boldsymbol{\Sigma}_2|)\\
&\quad-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)+\ln(\pi_2),
\end{aligned}$$

where (a) takes natural logarithm from the sides of equation. According to Eq. (19), we have:

$$\begin{aligned}
&-\frac{1}{2}\ln(|\boldsymbol{\Sigma}_1|)-\frac{1}{2}\boldsymbol{x}^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{x}-\frac{1}{2}\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1\\
&+\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{x}+\ln(\pi_1)\\
&=-\frac{1}{2}\ln(|\boldsymbol{\Sigma}_2|)-\frac{1}{2}\boldsymbol{x}^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{x}-\frac{1}{2}\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2\\
&+\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{x}+\ln(\pi_2).
\end{aligned}$$

Therefore, if we multiply the sides of equation by 2, we have:

$$\begin{aligned}
&\boldsymbol{x}^\top(\boldsymbol{\Sigma}_1-\boldsymbol{\Sigma}_2)^{-1}\boldsymbol{x}+2\,(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2-\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1)^\top\boldsymbol{x}\\
&+(\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2)+\ln\Big(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|}\Big)\\
&+2\,\ln(\frac{\pi_2}{\pi_1})=0,
\end{aligned}\tag{25}$$

which is in the quadratic form $\boldsymbol{x}^\top\boldsymbol{A}\,\boldsymbol{x}+\boldsymbol{b}^\top\boldsymbol{x}+c=0$. Therefore, if we consider Gaussian distributions for the two classes, the decision boundary of classification is quadratic. Because of quadratic decision boundary which discriminates the two classes, this method is named *quadratic discriminant* analysis.

For obtaining Eq. (25), we brought the expressions to the right side which was corresponding to the second class; therefore, if we use $\delta(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ as the left-hand-side expression (function) in Eq. (25):

$$\begin{aligned}
\delta(\boldsymbol{x}) :=\ &\boldsymbol{x}^\top(\boldsymbol{\Sigma}_1-\boldsymbol{\Sigma}_2)^{-1}\boldsymbol{x}+2\,(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2-\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1)^\top\boldsymbol{x}\\
&+(\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2)+\ln\Big(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|}\Big)+2\,\ln(\frac{\pi_2}{\pi_1}),
\end{aligned}\tag{26}$$

the class of an instance $\boldsymbol{x}$ is estimated as the Eq. (22).

If the priors of two classes are equal, i.e., $\pi_1 = \pi_2$, the Eq. (20) becomes:

$$\begin{aligned}
&\boldsymbol{x}^\top(\boldsymbol{\Sigma}_1-\boldsymbol{\Sigma}_2)^{-1}\boldsymbol{x}+2\,(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2-\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1)^\top\boldsymbol{x}\\
&+(\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2)+\ln\Big(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|}\Big)=0,
\end{aligned}\tag{27}$$

whose left-hand-side expression can be considered as $\delta(\boldsymbol{x})$ in Eq. (22).

LDA AND QDA FOR MULTICASS CLASSIFICATION

5. LDA and QDA for Multi-class Classification

Now we consider multiple classes, which can be more than two, indexed by $k \in \{1, \ldots, |\mathcal{C}|\}$. Recall Eq. (12) or (15) where we are using the scaled posterior, i.e., $f_k(\boldsymbol{x})\,\pi_k$. According to Eq. (16), we have:

$$f_k(\boldsymbol{x})\,\pi_k = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}_k|}} \exp\Big(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)}{2}\Big)\pi_k.$$

Taking natural logarithm gives:

$$\ln(f_k(\boldsymbol{x})\,\pi_k) = -\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + \ln(\pi_k).$$

We drop the constant term $-(d/2)\ln(2\pi)$ which is the same for all classes (note that this term is multiplied before taking the logarithm). Thus, the scaled posterior of the k-th class becomes:

$$\delta_k(\boldsymbol{x}) := -\frac{1}{2}\ln(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + \ln(\pi_k). \tag{28}$$

In QDA, the class of the instance $\boldsymbol{x}$ is estimated as:

$$\widehat{\mathcal{C}}(\boldsymbol{x}) = \arg\max_k \ \delta_k(\boldsymbol{x}), \tag{29}$$

because it maximizes the posterior of that class. In this expression, $\delta(\boldsymbol{x})$ is Eq. (28).

In LDA, we assume that the covariance matrices of the k classes are equal:

$$\boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_{|\mathcal{C}|} = \boldsymbol{\Sigma}. \tag{30}$$

Therefore, the Eq. (28) becomes:

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) + \ln(\pi_k) = -\frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \boldsymbol{\mu}_k^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \ln(\pi_k).$$

We drop the constant terms $-(1/2)\ln(|\boldsymbol{\Sigma}|)$ and $-(1/2)\,\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$ which are the same for all classes (note that before taking the logarithm, the term $-(1/2)\ln(|\boldsymbol{\Sigma}|)$ is multiplied and the term $-(1/2)\,\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$ is multiplied as an exponential term). Thus, the scaled posterior of the k-th class becomes:

$$\delta_k(\boldsymbol{x}) := \boldsymbol{\mu}_k^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln(\pi_k). \tag{31}$$

In LDA, the class of the instance $\boldsymbol{x}$ is determined by Eq. (29), where $\delta(\boldsymbol{x})$ is Eq. (31), because it maximizes the posterior of that class.

In conclusion, QDA and LDA deal with maximizing the *posterior* of classes but work with the *likelihoods (class conditional)* and *priors*.

6. Estimation of Parameters in LDA and QDA

mate them. Usually, the prior of the k-th class is estimated according to the sample size of the k-th class:

$$\widehat{\pi}_k = \frac{n_k}{n}, \tag{32}$$

The mean of the k-th class can be estimated using the Maximum Likelihood Estimation (MLE), or Method of Moments (MOM), for the mean of a Gaussian distribution:

$$\mathbb{R}^d \ni \widehat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i=1}^{n} \boldsymbol{x}_i\,\mathbb{I}\big(\mathcal{C}(\boldsymbol{x}_i) = k\big), \tag{33}$$

In QDA, the covariance matrix of the k-th class is estimated using MLE:

$$\mathbb{R}^{d\times d} \ni \widehat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k}\sum_{i=1}^{n}(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)^\top\,\mathbb{I}\big(\mathcal{C}(\boldsymbol{x}_i) = k\big). \tag{34}$$

Or we can use the *unbiased* estimation of the covariance matrix:

$$\mathbb{R}^{d\times d} \ni \widehat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k - 1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)^\top\,\mathbb{I}\big(\mathcal{C}(\boldsymbol{x}_i) = k\big). \tag{35}$$

In LDA, we assume that the covariance matrices of the classes are equal; therefore, we use the weighted average of the estimated covariance matrices as the common covariance matrix in LDA:

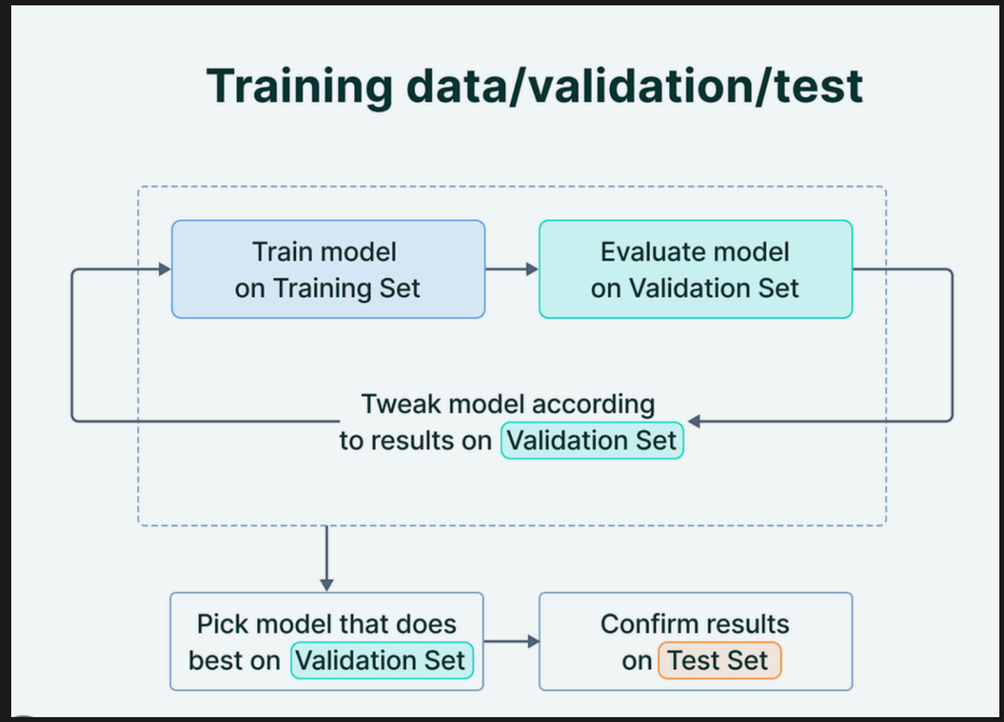
$$\mathbb{R}^{d\times d} \ni \widehat{\boldsymbol{\Sigma}} = \frac{\sum_{k=1}^{|\mathcal{C}|} n_k\,\widehat{\boldsymbol{\Sigma}}_k}{\sum_{r=1}^{|\mathcal{C}|} n_r} = \frac{\sum_{k=1}^{|\mathcal{C}|} n_k\,\widehat{\boldsymbol{\Sigma}}_k}{n}, \tag{36}$$

VALIDATION AND TESTING:

The trained model is evaluated on a separate validation dataset to assess its performance and generalization ability. Performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC are calculated to measure the model's effectiveness. Hyperparameter tuning and model refinement may be performed based on validation results. Finally, the model is tested on an unseen test dataset to provide an unbiased estimate of its performance.

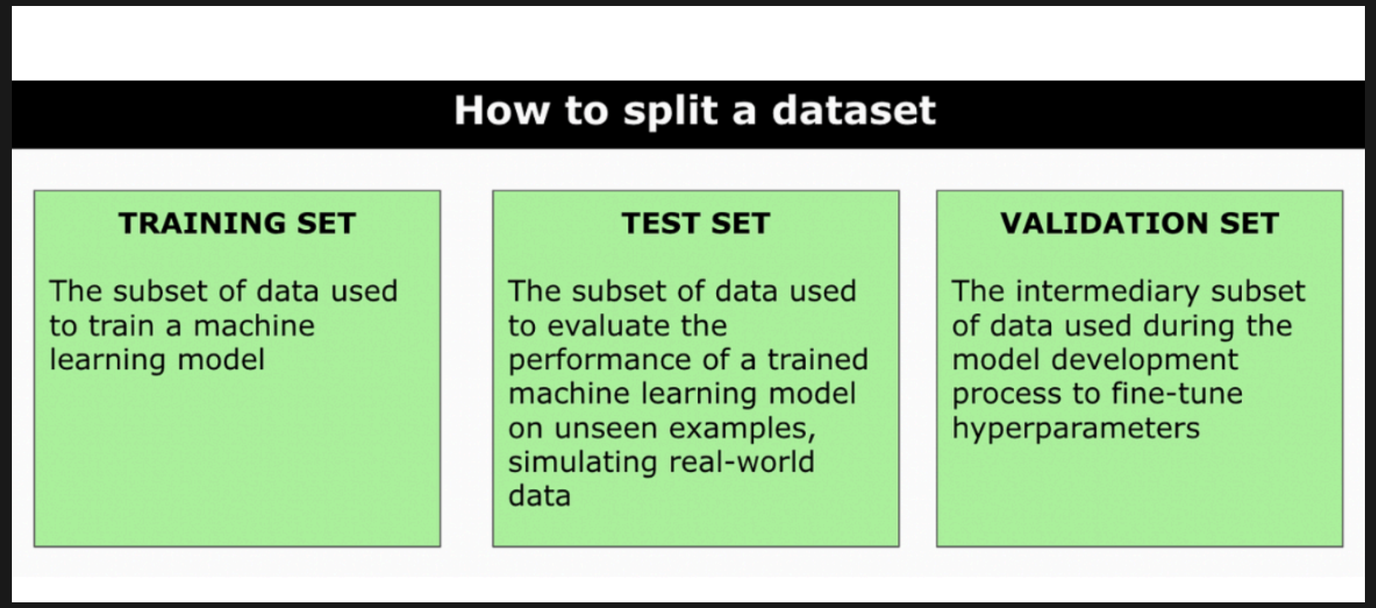
Training Dataset: The sample of data used to fit the model.
The actual dataset that we use to train the model (weights and biases in the case of a Neural Network). The model sees and learns from this data.

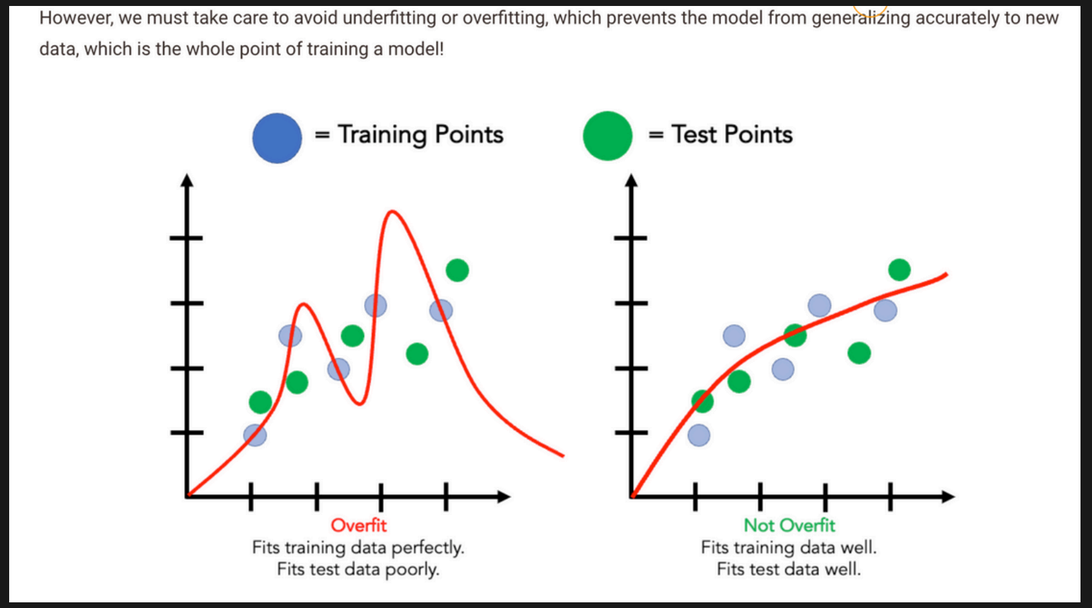
Validation Dataset: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.



Test Dataset: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained(using the train and validation sets). The test set is generally what is used to evaluate competing models (For example on many Kaggle competitions, the validation set is released initially along with the training set and the actual test set is only released when the competition is about to close, and it is the result of the the model on the Test set that decides the winner).

MODEL DEPLOYMENT:





Training Dataset

Training Dataset: The sample of data used to fit the model.

The actual dataset that we use to train the model (weights and biases in the case of a Neural Network). The model sees and learns from this data.

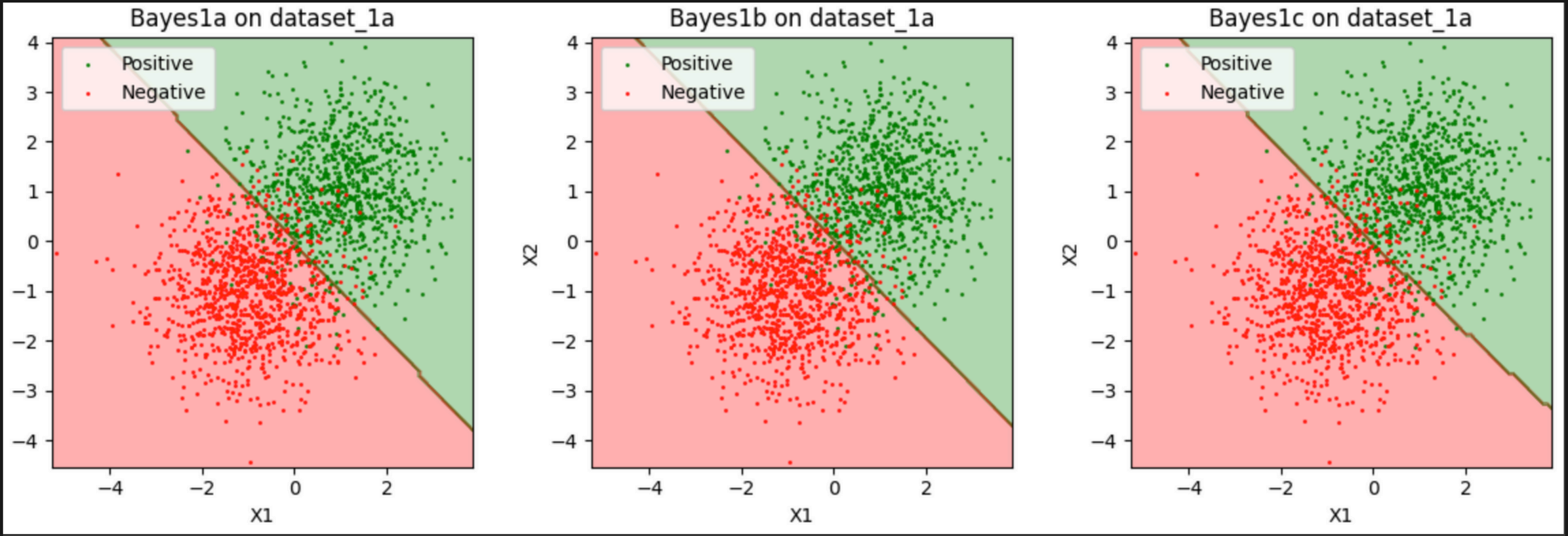
Validation Dataset

Validation Dataset: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

Test Dataset

Test Dataset: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained(using the train and validation sets).

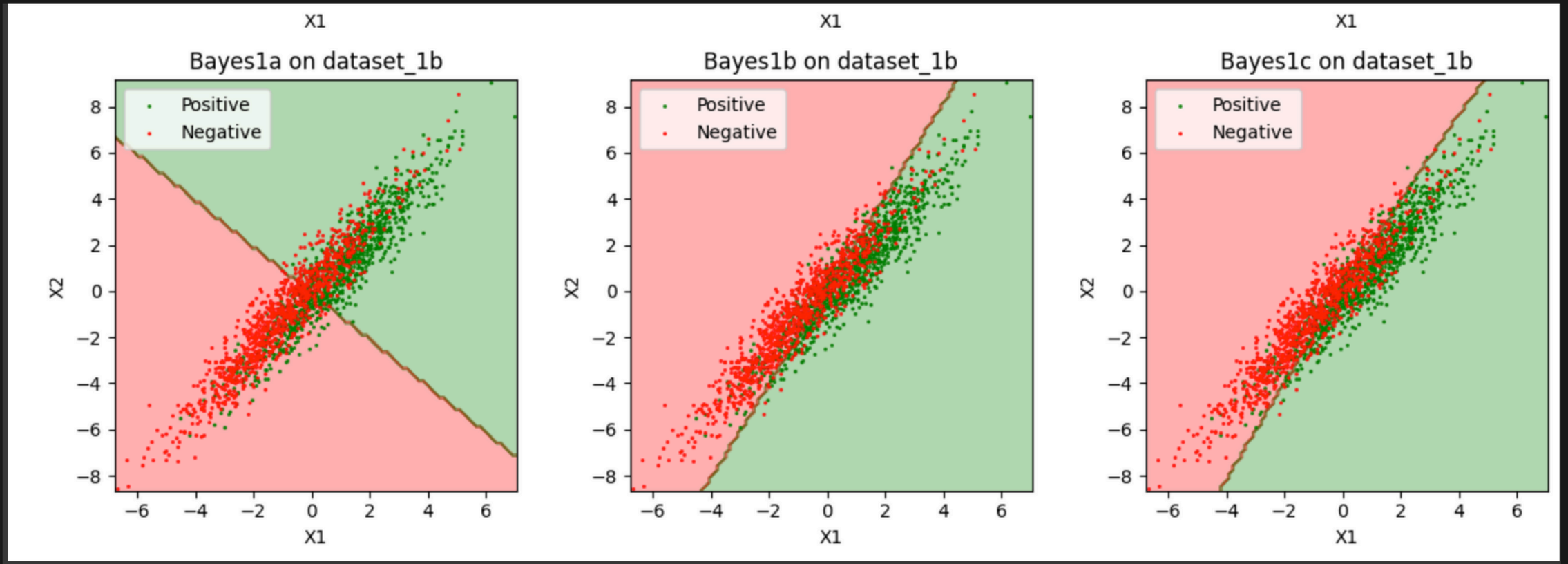
RESULTS OF THE CODE:



The unknown mean is [1,1] and [-1,-1] for the positive and negative class respectively. The known common covariance is [[1,0][0,1]] (the identity matrix).

	Dataset_1a	Dataset_1b	Dataset_1c
Bayes1a	0.0685	0.3005	0.1875
Bayes1b	0.0705	0.228	0.1875
Bayes1c	0.07	0.2285	0.163

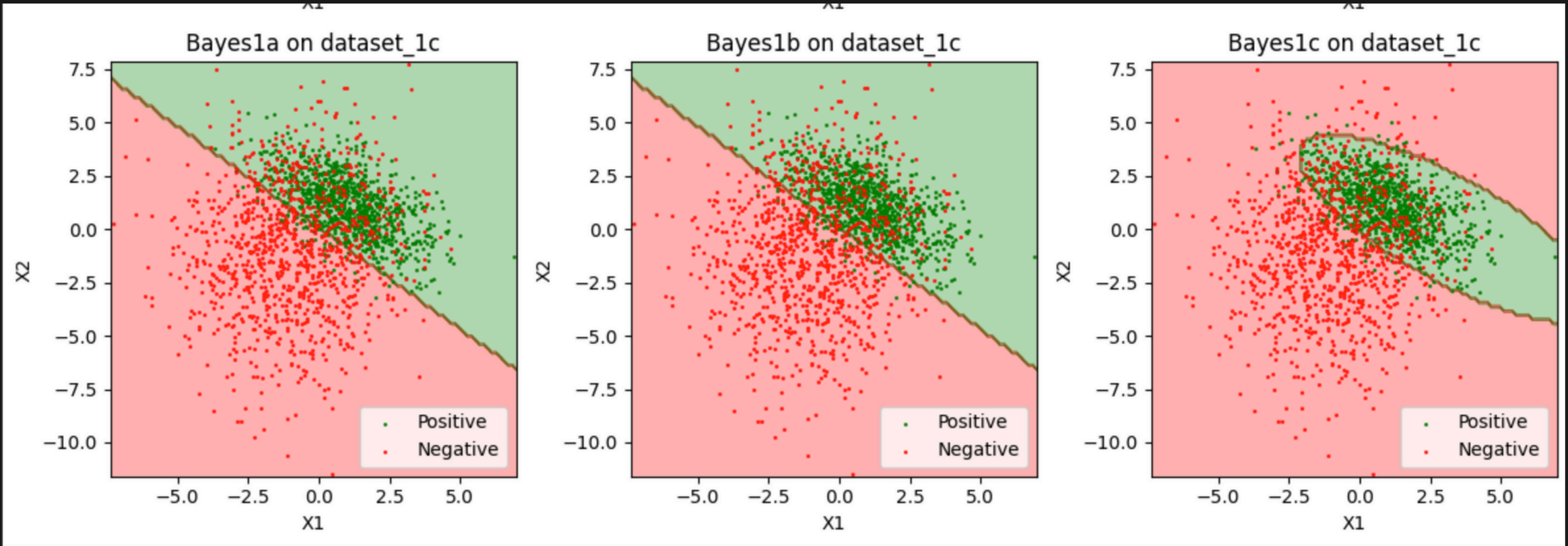
RESULTS OF THE CODE:



The unknown mean is [1,1] and [-1,-1] for the positive and negative class respectively. The known common covariance is [[3, 4][4, 6]], a symmetric, positive-semidefinite matrix.

	Dataset_1a	Dataset_1b	Dataset_1c
Bayes1a	0.0685	0.3005	0.1875
Bayes1b	0.0705	0.228	0.1875
Bayes1c	0.07	0.2285	0.163

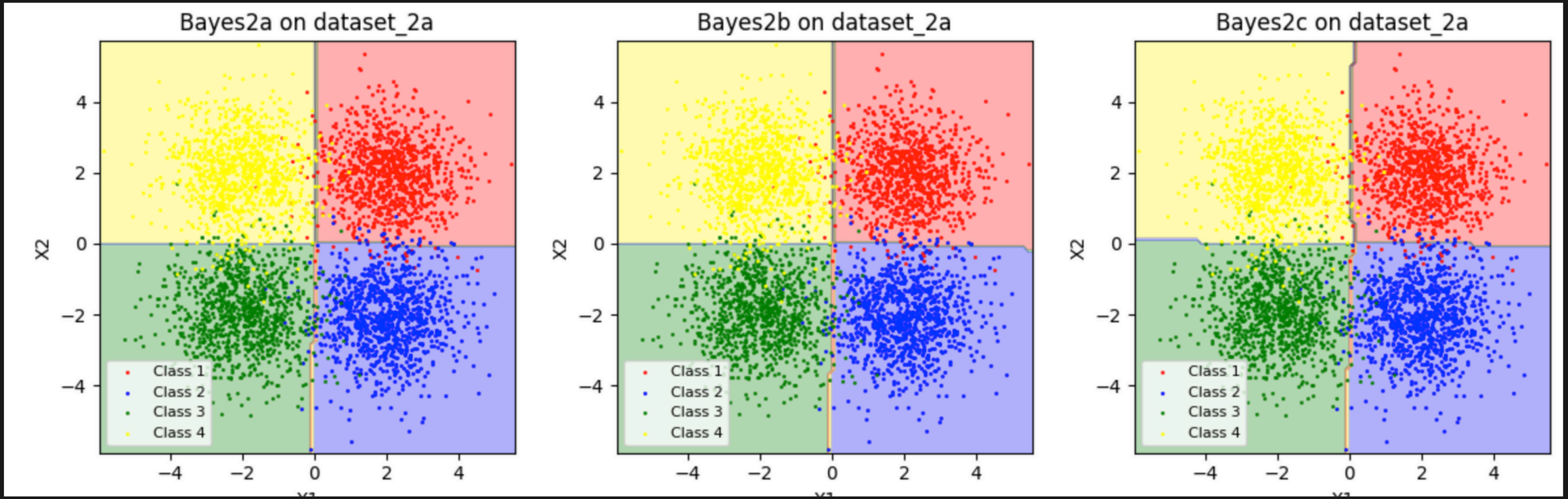
RESULTS OF THE CODE:



The unknown mean is [1,1] and [-1,-1] for the positive and negative class respectively. The unknown different covariances are [[4, 0] [0, 9]] and [[2,-1] [-1, 2]], symmetric, positive-semidefinite matrices.

	Dataset_1a	Dataset_1b	Dataset_1c
Bayes1a	0.0685	0.3005	0.1875
Bayes1b	0.0705	0.228	0.1875
Bayes1c	0.07	0.2285	0.163

RESULTS OF THE CODE:



The unknown means are as follows for each class:

Class 1: [2, 2]

Class 2: [2, -2]

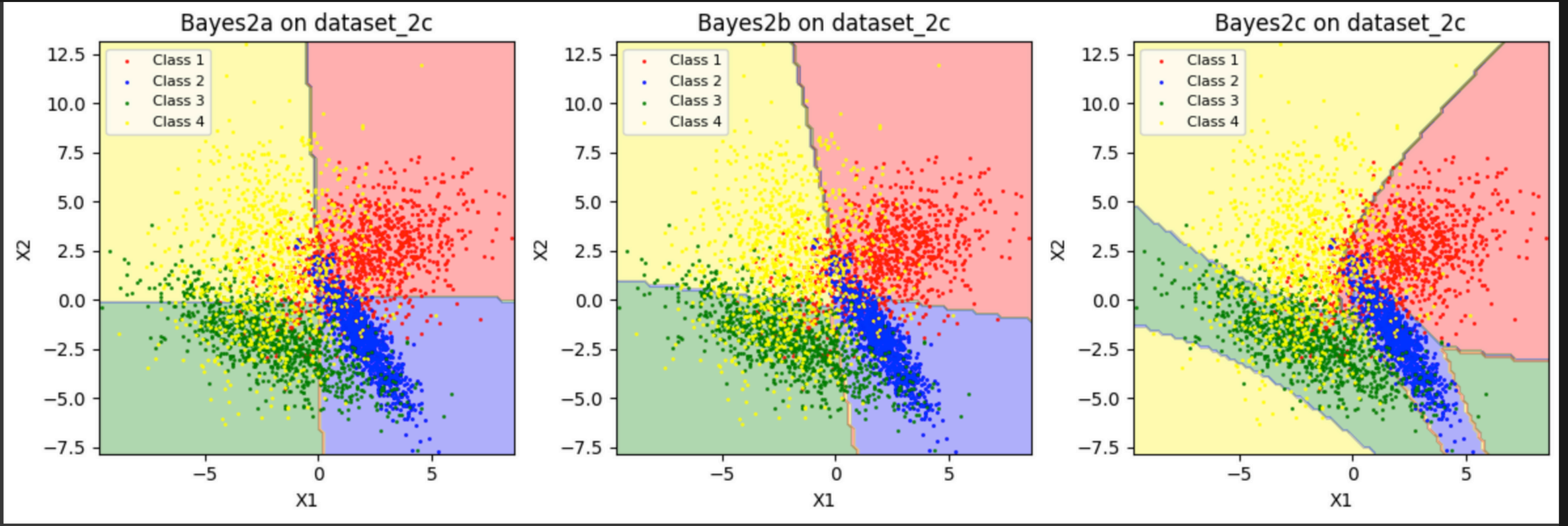
Class 3: [-2, -2]

Class 4: [-2, 2]

The known common covariance for all classes is: [[1, 0], [0, 1]]

	Dataset_2a	Dataset_2b	Dataset_2c
Bayes2a	0.0435	0.30025	0.27475
Bayes2b	0.044	0.20775	0.26425
Bayes2c	0.045	0.20975	0.2035

RESULTS OF THE CODE:



The unknown means are as follows for each class:

Class 1: [2, 2]

Class 2: [2, -2]

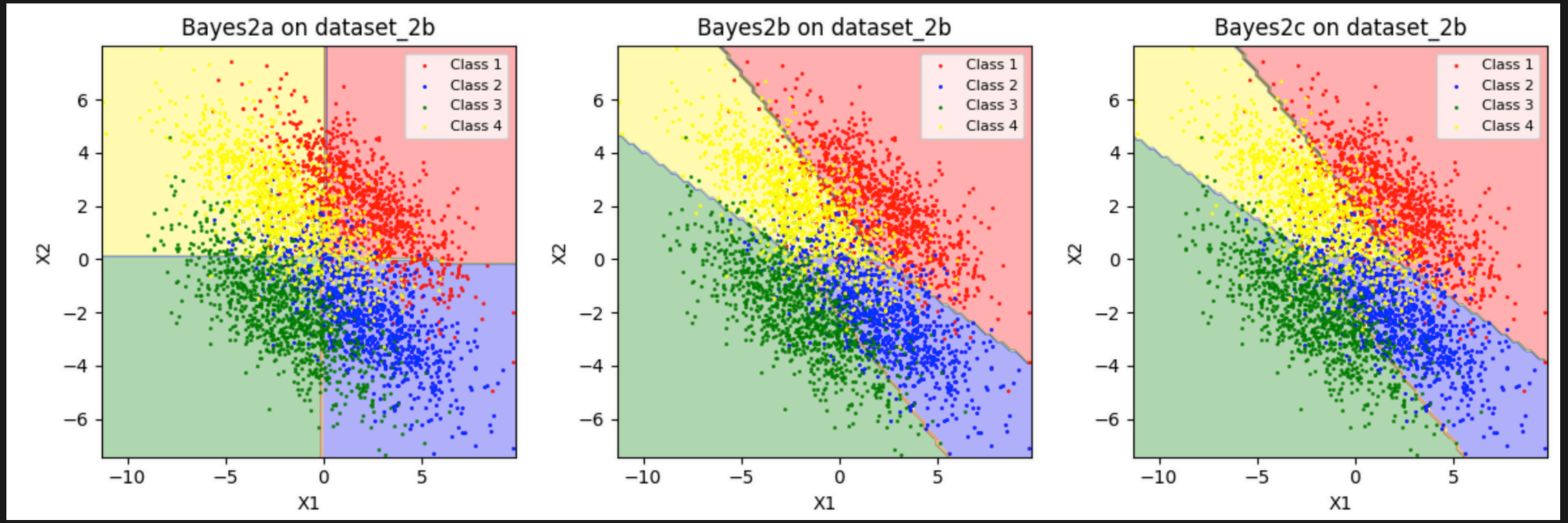
Class 3: [-2, -2]

Class 4: [-2, 2]

The unknown common covariance for all classes is: [[6, -3], [-3, 3]]

	Dataset_2a	Dataset_2b	Dataset_2c
Bayes2a	0.0435	0.30025	0.27475
Bayes2b	0.044	0.20775	0.26425
Bayes2c	0.045	0.20975	0.2035

RESULTS OF THE CODE:



The unknown means are as follows for each class:

Class 1: [2, 2]

Class 2: [2, -2]

Class 3: [-2, -2]

Class 4: [-2, 2]

The unknown covariances are as follows for each class:

Class 1: [[5, 2], [2, 4]]

Class 2: [[1, -1.5], [-1.5, 3]]

Class 3: [[6, -3], [-3, 3]]

Class 4: [[4, 0], [0, 9]]

	Dataset_2a	Dataset_2b	Dataset_2c
Bayes2a	0.0435	0.30025	0.27475
Bayes2b	0.044	0.20775	0.26425
Bayes2c	0.045	0.20975	0.2035