General Instructions to students:

1. There are 5 types of cells in this notebook. The cell type will be indicated within the cell.
    A. Markdown cells with problem written in it. (DO NOT TOUCH THESE CELLS) (**Cell type: TextRead**)
    B. Python cells with setup code for further evaluations. (DO NOT TOUCH THESE CELLS) (**Cell type: CodeRead**)
    C. Python code cells with some template code or empty cell. (FILL CODE IN THESE CELLS BASED ON INSTRUCTIONS IN CURRENT AND PREVIOUS CELLS) (**Cell type: CodeWrite**)
    D. Markdown cells where a written reasoning or conclusion is expected. (WRITE SENTENCES IN THESE CELLS) (**Cell type: TextWrite**)
    E. Temporary code cells for convenience. (YOU MAY DO WHAT YOU WILL WITH THESE CELLS, WE WILL REPLACE WHATEVER YOU WRITE HERE WITH OFFICIAL EVALUATION CODE) (**Cell type: Convenience**)
2. You are not allowed to insert new cells in the submitted notebook.
3. You are not allowed to import any extra packages.
4. In CodeWrite Cells, the only outputs to be given are plots asked in the question. Nothing else to be output/print.
5. If TextWrite cells ask you to give accuracy/error/other numbers you can print them on the code cells, but remove the print statements before submitting.
6. The convenience code can be used to check the expected syntax of the functions. At a minimum, your entire notebook must run with "run all" with the convenience cells as it is. Any runtime failures on the submitted notebook as it is will get zero marks.
7. All evaluation datasets will be given as .npz files, and will contain data in 4 numpy arrays :"X_train, Y_train, X_test, Y_test". In that order. The meaning of the 4 arrays can be easily inferred from their names.
8. All plots must be labelled properly.
9. Plotting the data and prediction is highly encouraged for debugging. But remove debugging/understanding code before submitting.
10. You need to submit a notebook named Team_{TeamNumber}.ipynb with all the CodeRead and CodeWrite cells along with TextWrite markdown cells. Your notebook NEEDS to run out of the box otherwise you will NOT be evaluated.
11. If you are unable to copy paste cells (specially the markdown cells) from the given pdf, you may write your own markdown cells that resemble the ones given in the pdf.

12. In addition to the notebook you shall also be required to make a presentation on how you approached the problems, your observations, etc. You DON'T need to submit the PPT just the notebook. The presentation will happen later, i.e. you will have more time (of the order of days) after you submit the notebook to prepare for the presentation.

13. The case study is divided into 2 broad sections:

    - Binary Classification: In this section you first have to write binary classifiers (that you think will be optimal) for 3 cases of data distributions. You have to then test out your models using data that you will simulate to follow the specified distributions.
    - Multiclass Classification: The 2 subparts for this section are similar to the previous section, the difference would be that in this section there will be more than 2 classes.

Classification Overview

Variables can be characterized as either *quantitative* or *qualitative* (also known as *categorical*). Quantitative variables take on numerical values. Examples include a person's age, height, or income, the value of a house, and the price of a stock. In contrast, qualitative variables take on values in one of K different *classes*, or categories. Examples of qualitative variables include a person's gender (male or female), the brand of product purchased (brand A, B, or C), whether a person defaults on a debt (yes or no), or a cancer diagnosis (Acute Myelogenous Leukemia, Acute Lymphoblastic Leukemia, or No Leukemia).

For this case study we shall be building models to predict a categorical response (output) variable using quantitative predictors (input).

Measure of Accuracy

Suppose that we seek to estimate $f : R^p \rightarrow \{C_1, C_2, \ldots, C_k\}$ where C_i's are the class labels, on the basis of training observations $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where x_i is a p-dimensional input data point and y_i is one of k class labels. The most common approach for quantifying the accuracy of our estimate $\hat{f}$ is the training *error rate*, i.e. the proportion of mistakes that are made if we apply our estimate $\hat{f}$ to the training observations:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) \tag{1}$$

Here $\hat{y}_i$ is the predicted class label for the ith observation using $\hat{f}$, and $I(y_i \neq \hat{y}_i)$ is an *indicator variable* that equals 1 iff $y_i \neq \hat{y}_i$ and 0 otherwise. Therefore, the equation above gives the proportion of incorrect classifications. The above is the training error, i.e. misclassifications in the training data, a "good" classifier for which the test error, given by $Ave(I(y_0 \neq \hat{y}_0))$, is also small.

The Bayes Classifier

It is possible to show that the test error rate is minimized, *on average*, by a very simple classifier that *assigns each observation to the most likely class, given its predictor (input) values*. In other words, we should simple assign a test observation with predictor x_0 to class j that maximizes $P(Y = j|X = x_0)$. This ver simple classifier is called the Bayes Classifier. The Bayes Classifer produces the lowest possible (average) test error rate out of ALL possible classifiers, this error rate is called the Bayes error Rate. Since it will always choose the class for which $P(Y = j|X = x_0)$ is largest, the error rates at $X = x_0$ will be $1 - \max_j P(Y = j|X = x_0)$. In general, the overall Bayes error rate is given by $1 - E[\max_j P(Y = j|X)]$, where the the expectation is over all input data points.

Using Bayes' Theorem for Classification

Suppose that we wish to classify into one of K classes. Let π_k denote the prior (unconditional) probability that a randomly chosen observation comes from the kth class, $\pi_k = P(Y = k)$. Let $f_k(x) = P(X = s|Y = k)$ (if X is continuous random variable then f_k will represent the probability that X falls in a small region dx around x) denote the the density function of X for an observation that comes from the kth class. Then according to Bayes's Theorem we can say:

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \tag{2}$$

The above is called the posterior probability, i.e. the probability of the response belonging to a certain class conditioned on the predictor value being $X = x.\backslash$ In general, estimating π_k is simple if we have a random sample of the response variable then it can just be estimated as the proportion of each class in the random sample. To estimate $f_k(x)$ we usually estimate some functional form for the density and then try to estimate the associated parameters. For this case study we will be assuming that the observations inside each class follow gaussian distributions.

Linear Discriminant Analysis (LDA)

Given n data points on p predictors $[x_ij]_{n\times p}$ and their true labels $Y = (y_1, y_2, \ldots, y_n)$ we have to build a classifier. In LDA we shall assume that all the $f_k(x)$ follow a multivariate gaussian distribution with class-specific mean vectors and common covariance matrix. Therefore $X \sim N(\mu_k, \Sigma)$ if $X \in$ kth class. Formally, the multivariate gaussian density function for p variates is given by:

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \cdot \exp(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)) \tag{3}$$

Plugging the density function for the kth class into Bayes' theorem and doing some algebra (taking the log and simplifying) reveals that the Bayes classifier assigns an observation $X = x$ to the class for which

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu^T \Sigma^{-1} \mu_k + \log \pi_k \tag{4}$$

is maximized. Note that, π_k's, μ_k's, and Σ are unknown parameters that would need to be estimated. The functions δ_k are called the discriminant functions and since, under the assumptions of LDA, the discriminant functions are linear in x the technique is called *Linear* Discriminant Analysis.

Quadratic Discriminant Analysis (QDA)

Given n data points on p predictors $[x_ij]_{n\times p}$ and their true labels $Y = (y_1, y_2, \ldots, y_n)$ we have to build a classifier. In QDA we shall assume that all the $f_k(x)$ follow a multivariate gaussian distribution with class-specific mean vectors and class specific covariances. Plugging the above assumptions into the Bayes' theorem and doing some algebraic manipulations tells us that the Bayes classifier assigns an observation $X = x$ to the class for which

$$\delta_k(x) = -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\log|\Sigma_k| + \log\pi_k \tag{5}$$

is largest. Note that, π_k's, μ_k's, and Σ_k's are unknown parameters that would need to be estimated. Notice that the discriminant functions are quadratic in x this time round hence the name of the technique.

In [ ]:
```
# Cell type : CodeRead

import numpy as np
import matplotlib.pyplot as plt
```

Cell type : TextRead

Problem 1: Binary Classifiers

Problem 1.1: Build Binary Classifiers

Derive classifiers under assumptions below, and use ML estimators to compute and return the results on a test set.

1a) Assume $X|Y = -1 \sim N(\mu_-, I)$ and $X|Y = 1 \sim N(\mu_+, I)$. *(Same known covariance)*

1b) Assume $X|Y = -1 \sim N(\mu_-, \Sigma)$ and $X|Y = 1 \sim N(\mu_+, \Sigma)$ *(Same unknown covariance)*

1c) Assume $X|Y = -1 \sim N(\mu_-, \Sigma_-)$ and $X|Y = 1 \sim N(\mu_+, \Sigma_+)$ *(different unknown covariance)*

In [ ]:
```
# Cell type : CodeWrite
def Bayes1a(X_train, Y_train, X_test):
    """ Give prediction for test instance using assumption 1a.

    Arguments:
    X_train: numpy array of shape (n,2)
    Y_train: +1/-1 numpy array of shape (n,)
    X_test : numpy array of shape (m,2)
```

```
    Returns:
    Y_test_pred : +1/-1 numpy array of shape (m,)

    """
    # Write your code here

    return Y_test_pred

def Bayes1b(X_train, Y_train, X_test):
    """ Give prediction for test instance using assumption 1b.

    Arguments:
    X_train: numpy array of shape (n,2)
    Y_train: +1/-1 numpy array of shape (n,)
    X_test : numpy array of shape (m,2)

    Returns:
    Y_test_pred : +1/-1 numpy array of shape (m,)

    """
    # Write your code here

    return Y_test_pred

def Bayes1c(X_train, Y_train, X_test):

    """ Give prediction for test instance using assumption 1c.

    Arguments:
    X_train: numpy array of shape (n,2)
    Y_train: +1/-1 numpy array of shape (n,)
    X_test : numpy array of shape (m,2)

    Returns:
    Y_test_pred : +1/-1 numpy array of shape (m,)

    """
    # Write your code here

    return Y_test_pred
```

In [ ]:

```
# Cell type : Convenience

# Testing the functions above

# You may use the example here for testing syntax issues
# with your functions, and also as a sanity check. But the final evaluation
# will be done for different inputs to the functions. (So you can't just
# solve the problem for this one example given below.)


X_train_pos = np.random.randn(1000,2)+np.array([[1.,2.]])
X_train_neg = np.random.randn(1000,2)+np.array([[2.,4.]])
X_train = np.concatenate((X_train_pos, X_train_neg), axis=0)
Y_train = np.concatenate(( np.ones(1000), -1*np.ones(1000) ))
```

```
X_test_pos = np.random.randn(1000,2)+np.array([[1.,2.]])
X_test_neg = np.random.randn(1000,2)+np.array([[2.,4.]])
X_test = np.concatenate((X_test_pos, X_test_neg), axis=0)
Y_test = np.concatenate(( np.ones(1000), -1*np.ones(1000) ))

Y_pred_test_1a = Bayes1a(X_train, Y_train, X_test)
Y_pred_test_1b = Bayes1b(X_train, Y_train, X_test)
Y_pred_test_1c = Bayes1c(X_train, Y_train, X_test)
```

Cell type : TextRead

Problem 1.2

For this part you have to demonstrate, using data that you will simulate as specified in Problem 1.1, that Bayes1a performs best for data that follow distribution 1a), Bayes1b performs best for data that follow distribution 1b), and Bayes1c performs best for data that follow distribution 1c).

In the next CodeWrite cell, plot all the classifiers (3 classification algos on 3 datasets = 9 plots) on a 2d plot (color the positively classified area light green, and negatively classified area light red). Add the training data points also on the plot. Plots to be organised into 3 plots as follows: One plot for each dataset, with three subplots in each for the three classifiers. Label the 9 plots appropriately.

In the next Textwrite cell, summarise (use the plots of the data and the assumptions in the problem to explain) your observations regarding the nine learnt classifiers, and also give the error rate of the three classifiers on the three datasets as 3x3 table, with appropriately named rows and columns.

In [ ]:
```
# Cell type : CodeWrite
# Write the code for loading the data, running the three algos, and plotting here.
# Assume that the 3 datasets are named dataset_1a.npz, dataset_1b.npz, dataset_1c.npz r
# (Use the functions written previously.)
```

Cell type : TextWrite

	Dataset_1a	Dataset_1b	Dataset_1b
Bayes1a	x	x	x
Bayes1b	x	x	x
Bayes1c	x	x	x

Observations:

(1) For dataset_1a: ...
(2) For dataset_1b: ...
(3) For dataset_1c: ...

Cell type : TextRead

Problem 2 : Multiclass Classifiers

Problem 2.1: Build Multiclass Classifiers

Derive classifiers under assumptions below, and use ML estimators to compute and return the results on a test set.

2a) Assume $\mathrm{X}|\mathrm{Y} = \mathrm{a}$ is distributed as Normal with mean μ_a and variance I.

2b) Assume $\mathrm{X}|\mathrm{Y} = \mathrm{a}$ is distributed as Normal with mean μ_a and variance Σ.

2c) Assume $\mathrm{X}|\mathrm{Y} = \mathrm{a}$ is distributed as Normal with mean μ_a and variance Σ_a.

In [ ]:

```
# Cell type : CodeWrite
# Fill in functions in this cell


def Bayes2a(X_train, Y_train, X_test):
    """ Give Bayes classifier prediction for test instances
    using assumption 2a.

    Arguments:
    X_train: numpy array of shape (n,2)
    Y_train: {1,2,3,4} numpy array of shape (n,)
    X_test : numpy array of shape (m,2)

    Returns:
    Y_test_pred : {1,2,3,4} numpy array of shape (m,)

    """
    # Write code here

    return Y_test_pred

def Bayes2b(X_train, Y_train, X_test):
    """ Give Bayes classifier prediction for test instances
    using assumption 2b.

    Arguments:
    X_train: numpy array of shape (n,2)
    Y_train: {1,2,3,4} numpy array of shape (n,)
    X_test : numpy array of shape (m,2)

    Returns:
    Y_test_pred : {1,2,3,4} numpy array of shape (m,)

    """
    # Write Code here

    return Y_test_pred


def Bayes2c(X_train, Y_train, X_test):
    """ Give Bayes classifier prediction for test instances
    using assumption 2c.

    Arguments:
```

```
    X_train: numpy array of shape (n,2)
    Y_train: {1,2,3,4} numpy array of shape (n,)
    X_test : numpy array of shape (m,2)

    Returns:
    Y_test_pred : {1,2,3,4} numpy array of shape (m,)

    """
    # Write code here

    return Y_test_pred
```

In [ ]:

```
# Cell type : Convenience

# Testing the functions above

mat1=np.array([[1.,0.],[0.,1.]])
mat2=np.array([[1.,0.],[0.,1.]])
mat3=np.array([[1.,0.],[0.,1.]])
mat4=np.array([[1.,0.],[0.,1.]])

X_train_1 = np.dot(np.random.randn(1000,2), mat1)+np.array([[0.,0.]])
X_train_2 = np.dot(np.random.randn(1000,2), mat2)+np.array([[0.,2.]])
X_train_3 = np.dot(np.random.randn(1000,2), mat3)+np.array([[2.,0.]])
X_train_4 = np.dot(np.random.randn(1000,2), mat4)+np.array([[2.,2.]])

X_train = np.concatenate((X_train_1, X_train_2, X_train_3, X_train_4), axis=0)
Y_train = np.concatenate(( np.ones(1000), 2*np.ones(1000), 3*np.ones(1000), 4*np.ones(1


X_test_1 = np.dot(np.random.randn(1000,2), mat1)+np.array([[0.,0.]])
X_test_2 = np.dot(np.random.randn(1000,2), mat2)+np.array([[0.,2.]])
X_test_3 = np.dot(np.random.randn(1000,2), mat3)+np.array([[2.,0.]])
X_test_4 = np.dot(np.random.randn(1000,2), mat4)+np.array([[2.,2.]])

X_test = np.concatenate((X_test_1, X_test_2, X_test_3, X_test_4), axis=0)
Y_test = np.concatenate(( np.ones(1000), 2*np.ones(1000), 3*np.ones(1000), 4*np.ones(10


Y_pred_test_2a = Bayes2a(X_train, Y_train, X_test)
Y_pred_test_2b = Bayes2b(X_train, Y_train, X_test)
Y_pred_test_2c = Bayes2c(X_train, Y_train, X_test)
```

Cell type : TextRead

Problem 2.2

For this part you have to demonstrate, using data that you will simulate as specified in Problem 2 description (assuming there are 4 classes), that Bayes2a performs best for data that follow distribution 2a), Bayes2b performs best for data that follow distribution 2b), and Bayes2c performs best for data that follow distribution 2c).

In the next CodeWrite cell, plot all the classifiers (3 classification algos on 3 datasets = 9 plots) on a 2d plot (color the regions corresponding to each class with colors of your choice, coloring should be

consistent across all plots, and there should be corresponding legend). Add the training data points also on the plot. Plots to be organised into 3 plots follows: One plot for each dataset, with three subplots in each for the three classifiers. Label the 9 plots appropriately.

In the next Textwrite cell, summarise (use the plots of the data and the assumptions in the problem to explain) your observations regarding the nine learnt classifiers, and also give the error rate of the three classifiers on the three datasets as 3x3 table, with appropriately named rows and columns.

In [ ]:
```
# Cell type : CodeWrite
# Write the code for loading the data, running the three algos, and plotting here.
# Assume that the datasets are called dataset_2a.npz, dataset_2b.npz, dataset_2c.npz re
# (Use the functions written previously.)
```

Cell type : TextWrite (Write your observations and table of errors here)

Error rate:

	Dataset_2a	Dataset_2b	Dataset_2c
Bayes2a	x	x	x
Bayes2b	x	x	x
Bayes2c	x	x	x

Obsevation:
(1) In dataset_2a: ...
(2) In dataset_2b: ...
(3) In dataset_2c: ...

Below is a subjective question. You are encouraged to gain understanding of the machine learning data pipeline from sources of your choice. You will be quizzed based on your answer for this question during your presentation.

Problem 3

Provide a comprehensive explanation of the entire machine learning data pipeline for a classification problem. Please detail each step involved in the process and discuss the specific tasks that are carried out at each stage. This should include the initial data collection, data cleaning and preprocessing, model selection and training, validation and testing, and finally, model deployment and monitoring. Also discuss the importance of each step in the overall machine learning process and how they contribute to the development of an effective and accurate model.

In [ ]:
```
```