Pranshu Parate
202211063

Previous Assignment: - Generate the datasets A and B in R 2 with each of them consisting 2000 data points from normal distribution. The dataset A and B has been drawn from the N (μ1, Σ1) and N(μ2, Σ2) . Let us fix the μ1 = [-1,1] and μ2 = [2,1]. Separate the 250 data points from each classes as testing set. Plot the optimal Bayesian decision boundary.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import math
import sklearn.metrics
import scipy.io as sio
import seaborn as sb
```

In [2]:
```
u1 = np.array([-1,1])
u2 = np.array([2,2])
cov2 = np.array([[0.7,0],[0,0.3]])

A1,A2 = np.random.multivariate_normal(u1,cov2,2000).T
B1,B2 = np.random.multivariate_normal(u2,cov2,2000).T

df1 = pd.DataFrame([A1,A2, np.zeros(len(A1))])
df1 = df1.T
df1.columns = ['x1', 'x2', 'y']
df2 = pd.DataFrame([B1,B2, np.ones(len(A1))])
df2 = df2.T
df2.columns = ['x1', 'x2', 'y']
df = pd.concat([df1, df2])
train_x, test_x, train_y, test_y = train_test_split(df[['x1', 'x2']], df['y'],test_siz
a_x, ax_t, a_y, ay_t = train_test_split(A1, A2, test_size=0.125,random_state=42)
b_x, bx_t, b_y, by_t = train_test_split(B1, B2, test_size=0.125,random_state=42)
```

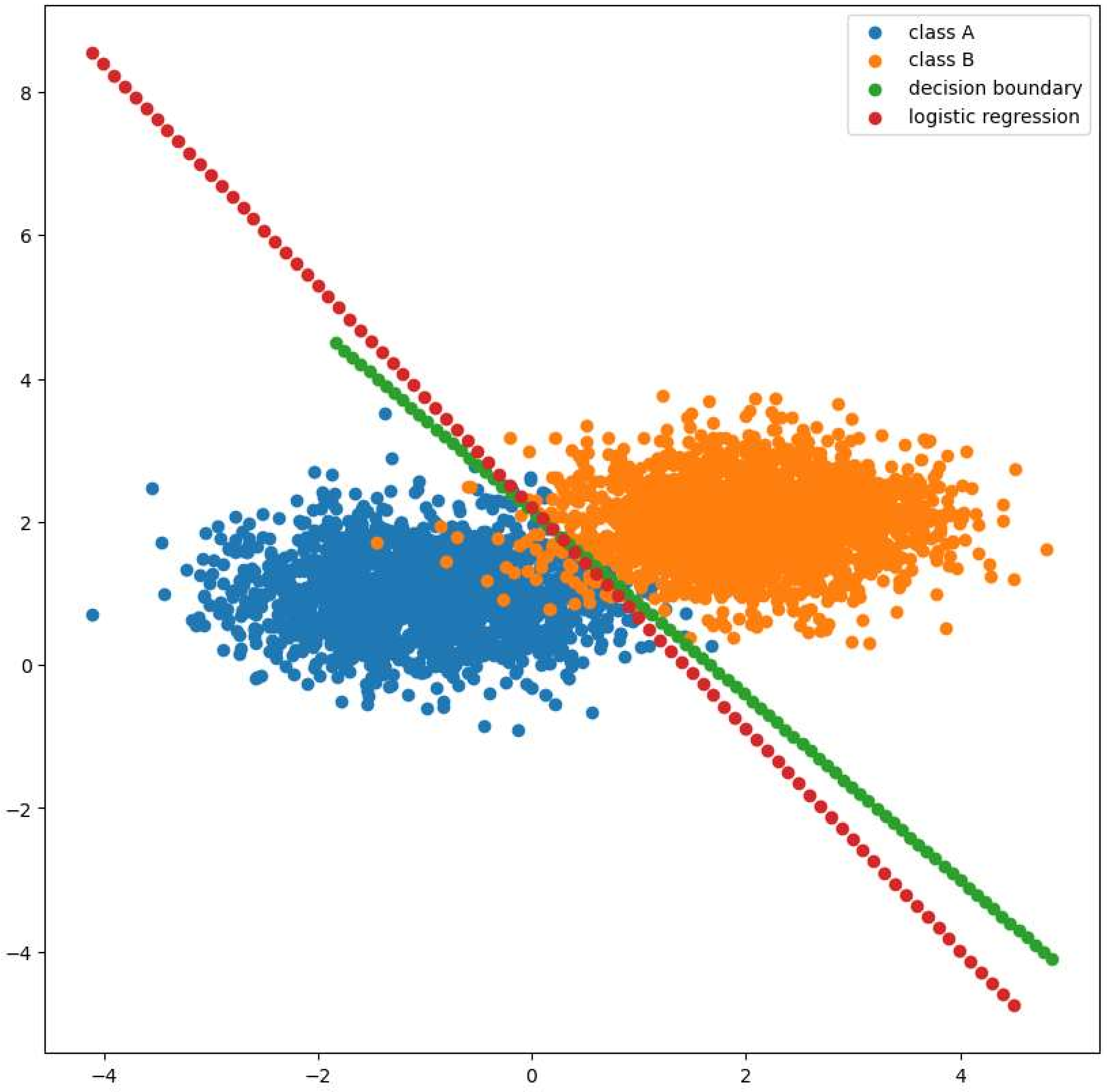
Q1. Write a function implementing the logistic regression model using the gradient descent method. Obtain the best accuracy on the test set by tuning the value of the parameter λ . Plot the decision boundary obtained by the logistic regression. Compare it with the Bayesian decision boundary.

In [5]:
```
w = np.dot(np.linalg.inv(cov2), u1 - u2)
c = np.dot(w , (1/2) * (u1 + u2))
print(w)
print(c)
```

```
[-4.28571429 -3.33333333]
-7.142857142857142
```

In [4]:
```
x2 = np.arange(np.append(a_x, b_x).min(), np.append(a_x, b_x).max(), 0.1)
x1 = (c - w[1] * x2) / w[0]

alpha = 0.0001
lmda = 2**-8
theta = np.array([0,0,0])
combined_x = train_x[['x1', 'x2']]
combined_x['ones'] = np.ones(len(train_x))
for i in range(2000):
  temp1 = np.dot(combined_x, theta)
  temp2 = 1 / (1 + math.e**( (-1) * temp1))
  temp3 = temp2 - train_y
  temp4 = np.dot(temp3, combined_x) + lmda * theta
  theta = theta - alpha * temp4

x1_lr = np.arange(np.append(a_x, b_x).min(), np.append(a_x, b_x).max(), 0.1)
x2_lr = (-theta[0]*x1_lr - theta[2]) / theta[1]

combined_x_test = test_x[['x1', 'x2']]
combined_x_test['ones'] = np.ones(len(test_x))

y_hat_test = np.dot(combined_x_test, theta)

for i in range(len(y_hat_test)):
  if y_hat_test[i] > 0:
    y_hat_test[i] = 1
  else:
    y_hat_test[i] = 0

print(f'Accuracy: {sklearn.metrics.accuracy_score(test_y, y_hat_test)}')

plt.figure(figsize = (10,10))
plt.scatter(A1, A2, label = 'class A')
plt.scatter(B1, B2, label = 'class B')
plt.scatter(x1,x2, label = 'decision boundary')
plt.scatter(x1_lr,x2_lr, label = 'logistic regression')
plt.legend()
plt.show()
```

```
Accuracy: 0.99
```



Q2. Consider the Iris dataset. The dataset contains three types of flower described by the four features. Consider only the data points with label 1 and 2. Divide the dataset into training, testing and validation in the ration 8:1:1. Use the training set to train the logistic regression model. Use the validation set to tune the parameter value λ. Finally obtain the accuracy on the test set.

In [8]:
```
data_dict = sio.loadmat("C:/Users/Dell/Downloads/iris.mat")
Y = data_dict['iris'][:, 0]
X = data_dict['iris'][:, 1:]

Y_new, X_new = [], []
for i,j in enumerate(Y):
  if j != 3:
    if j == 1:
      Y_new.append(0)
    else:
      Y_new.append(1)
    X_new.append(X[i])
x_train, x_test, y_train, y_test = train_test_split(X_new, Y_new, test_size=0.1, randc
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.1, rar
```

In [9]:
```
alpha = 0.001
lmda = 2**-8
theta = np.array([0,0,0,0,0])

combined_x = pd.DataFrame(x_train)
combined_x['ones'] = np.ones(len(x_train))

for i in range(2000):
  temp1 = np.dot(combined_x, theta)
  temp2 = 1 / (1 + math.e**( (-1) * temp1))
  temp3 = temp2 - y_train
  temp4 = np.dot(temp3, combined_x) + lmda * theta
  theta = theta - alpha * temp4

combined_x_test = pd.DataFrame(x_test)
combined_x_test['ones'] = np.ones(len(x_test))

y_hat_test = np.dot(combined_x_test, theta)

for i in range(len(y_hat_test)):
  if y_hat_test[i] > 0:
    y_hat_test[i] = 1
  else:
    y_hat_test[i] = 0

print(f'Accuracy {sklearn.metrics.accuracy_score(y_test, y_hat_test)}')
```

```
Accuracy 1.0
```

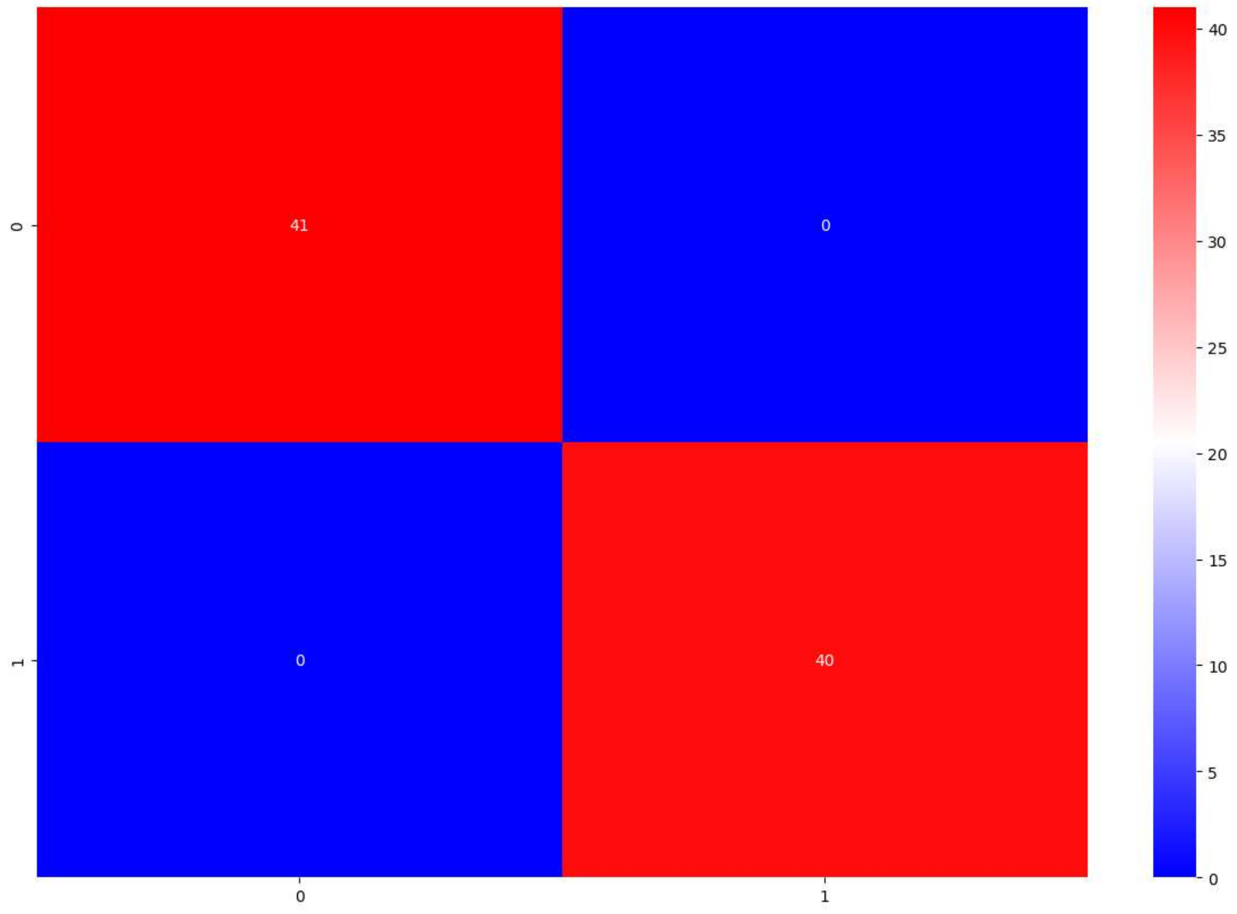
In [10]:
```
combined_x_test = pd.DataFrame(x_train)
combined_x_test['ones'] = np.ones(len(x_train))
y_hat_test = np.dot(combined_x_test, theta)
for i in range(len(y_hat_test)):
  if y_hat_test[i] > 0:
    y_hat_test[i] = 1
  else:
    y_hat_test[i] = 0
print(f'Accuracy {sklearn.metrics.accuracy_score(y_train, y_hat_test)}')
```

```
Accuracy 1.0
```

Q3. Learn about the different evaluation measure of a classification model namely Accuracy, Precision, Recall, Confusion matrix, F-Score and their significance. Write a brief note about them and submit it (Not less than 200 words). Compute all of them for problem 3.

In [13]:
```
print(f'Precision: {sklearn.metrics.precision_score(y_train,y_hat_test)}')
print(f'Recall: {sklearn.metrics.recall_score(y_train, y_hat_test)}')
print(f'F1 score:{sklearn.metrics.f1_score(y_train, y_hat_test)}')
plt.figure(figsize = (15,10))
sb.heatmap(sklearn.metrics.confusion_matrix(y_train,y_hat_test),annot = True,cmap = 'b
plt.show()
print("\n")
```

```
Precision: 1.0
Recall: 1.0
F1 score:1.0
```



Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions.

The precision is calculated as the ratio between the number of Positive samples correctly classified to the total number of samples classified as Positive (either correctly or incorrectly). The precision measures the model's accuracy in classifying a sample as positive.

The recall is calculated as the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect Positive samples. The higher the recall, the more positive samples detected.

Confusion matrix is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.

The traditional F measure is calculated as follows:

F-Measure = (2 *Precision* Recall) / (Precision + Recall) This is the harmonic mean of the two fractions. This is sometimes called the F-Score or the F1-Score and might be the most common metric used on imbalanced classification problems.