ECE 580 Homework 5

Libo Zhang (lz200)

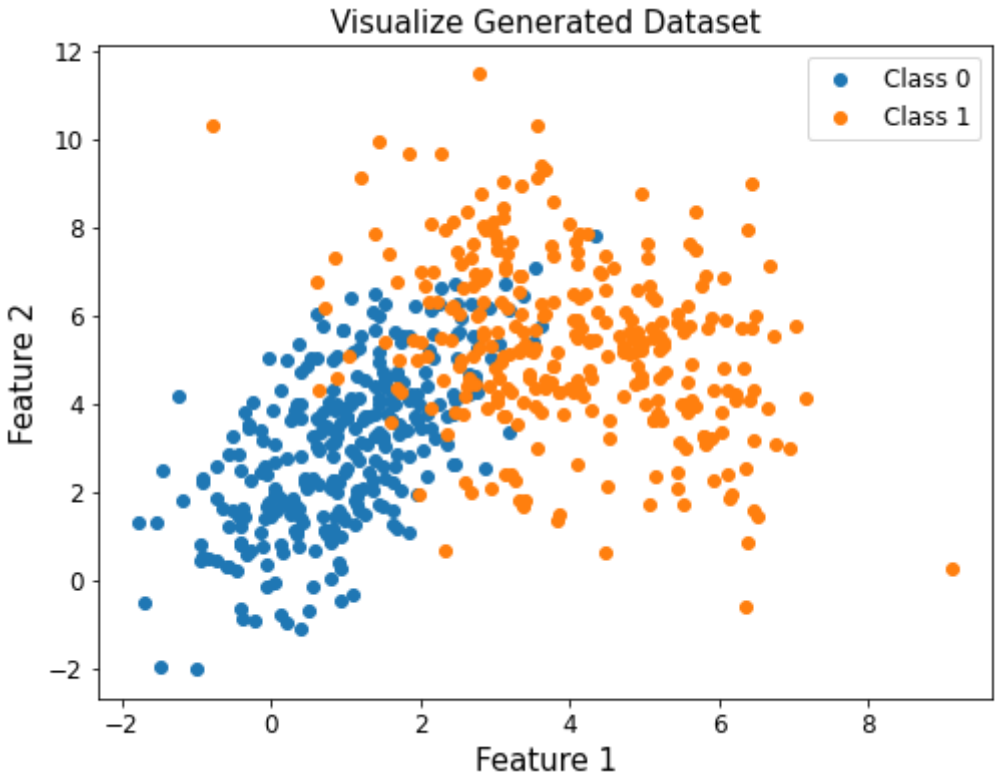
Exploring Bayes Classifiers Section

Question 1 Solution:

For Class 0 data points, $\mu_0 = [1, 3]^T$, $\Sigma_0 = \begin{bmatrix} 1 & +1 \\ +1 & 3 \end{bmatrix}$, Σ_0 has a positive correlation and $\sigma_0^2 \neq \sigma_1^2$.

For Class 1 data points, $\mu_1 = [4, 5]^T$, $\Sigma_1 = \begin{bmatrix} 2 & -1 \\ -1 & 4 \end{bmatrix}$, Σ_1 has a negative correlation and $\sigma_0^2 \neq \sigma_1^2$.

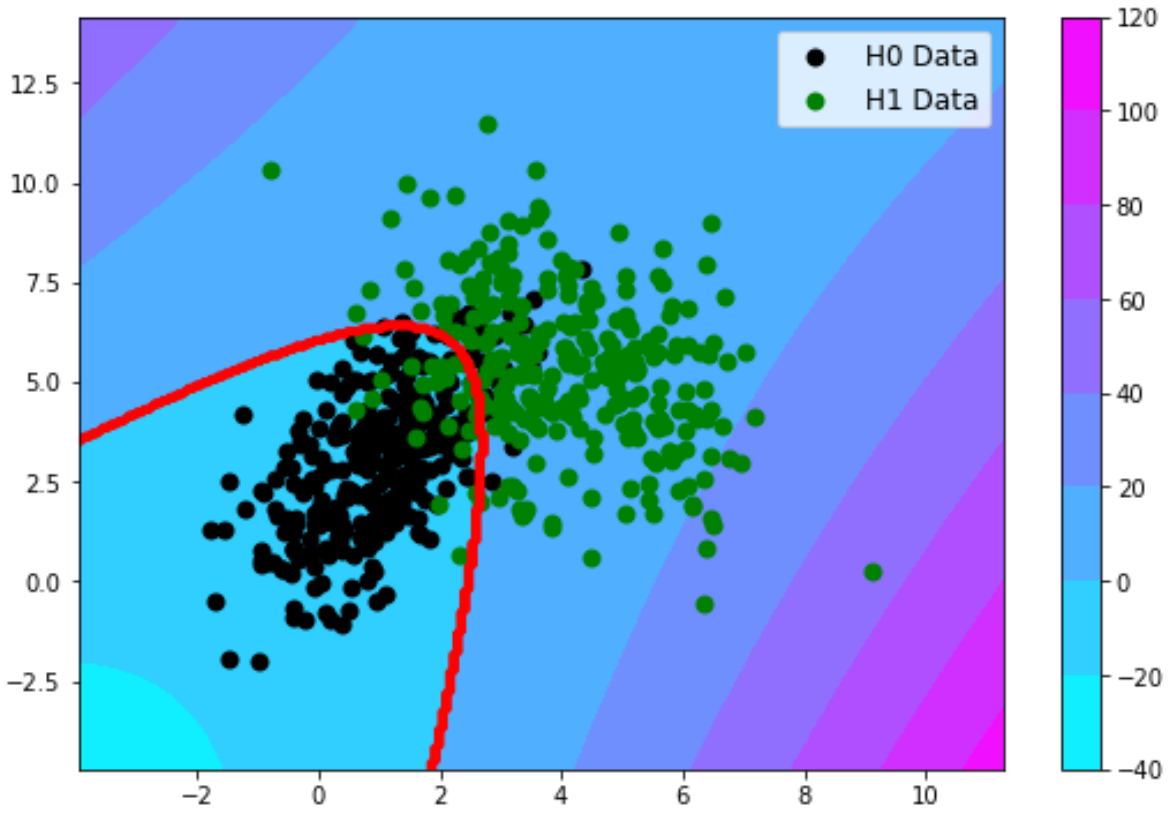
The scatter plot of data points is shown below, each class has 300 data points respectively.



We can see that the distributions indeed overlap noticeably.

Question 2 Solution:

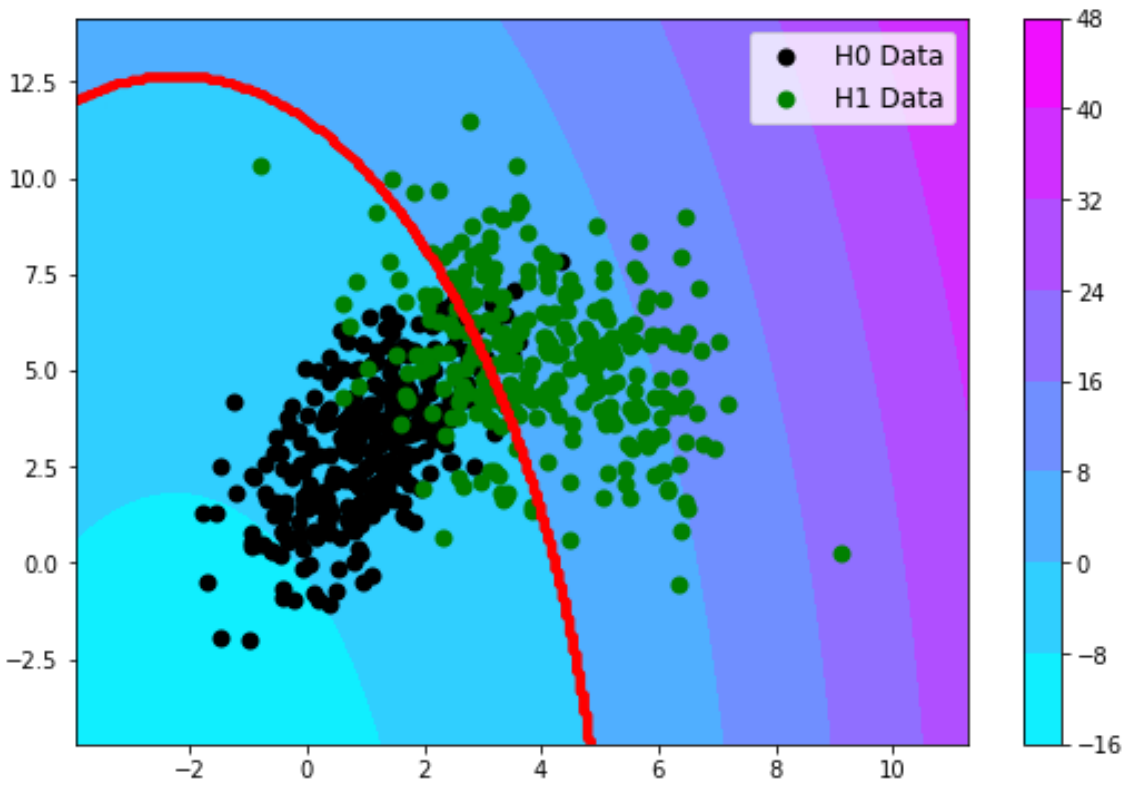
Question 2 (a) – Making no simplifying assumptions regarding the covariance structure, the decision statistic surface plot is shown below. The red line is the decision boundary.



Question 2 (b) – We assume the covariance matrices under each class are unique, which means that we cannot eliminate the quadratic term $x^T(\frac{-1}{2}\Sigma_i^{-1})x$ in the discriminant function $g(x)$, so we have the quadratic decision boundary curve here. In addition, the shape of the decision boundary looks like part of an oblique ellipse. The tilt or oblique ellipse shape is due to our assumption that these two features may be dependent.

Question 3 Solution:

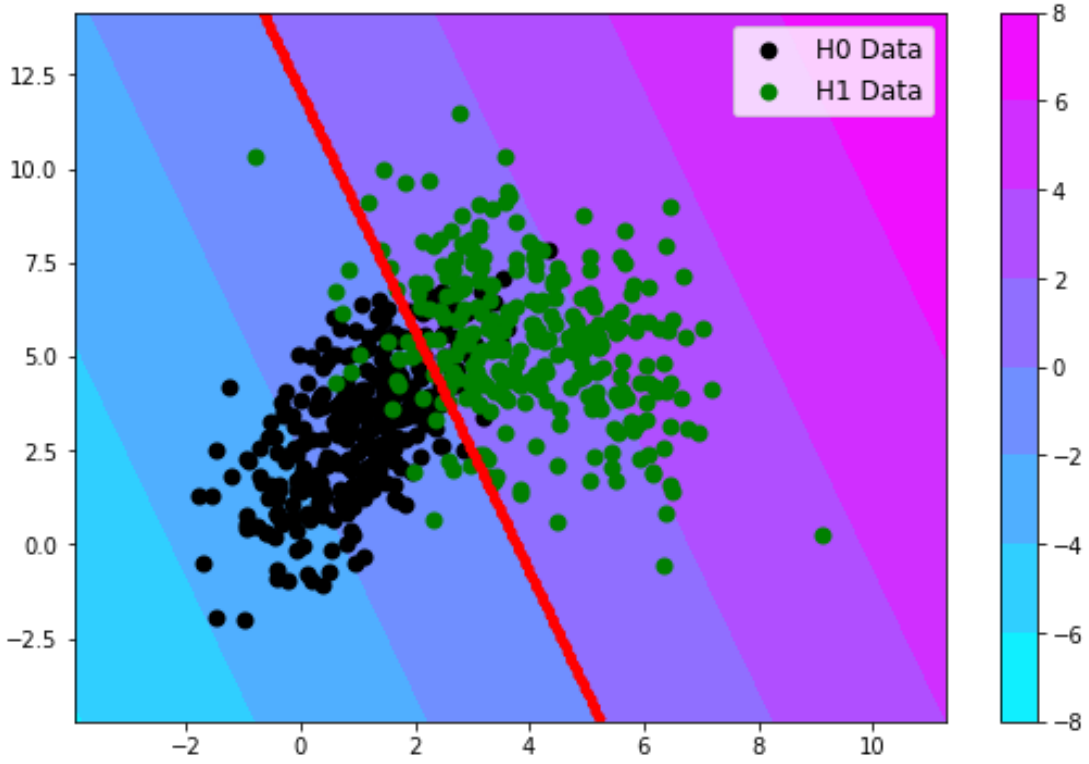
Question 3 (a) – Under the simplifying assumption that features are independent but the covariance matrices under each class are unique, the decision statistic surface with both the training data and the decision boundary superimposed on top plot is shown below. The red line is the decision boundary.



Question 3 (b) – We assume the covariance matrices under each class are unique, so that we still cannot eliminate the quadratic term $x^T(\frac{-1}{2}\Sigma_i^{-1})x$ in the discriminant function $g(x)$, which explains why we still get a quadratic decision boundary curve here. However, since we assume that these two features are independent, now the shape of the decision boundary looks like part of a standard ellipse (no tilt).

Question 4 Solution:

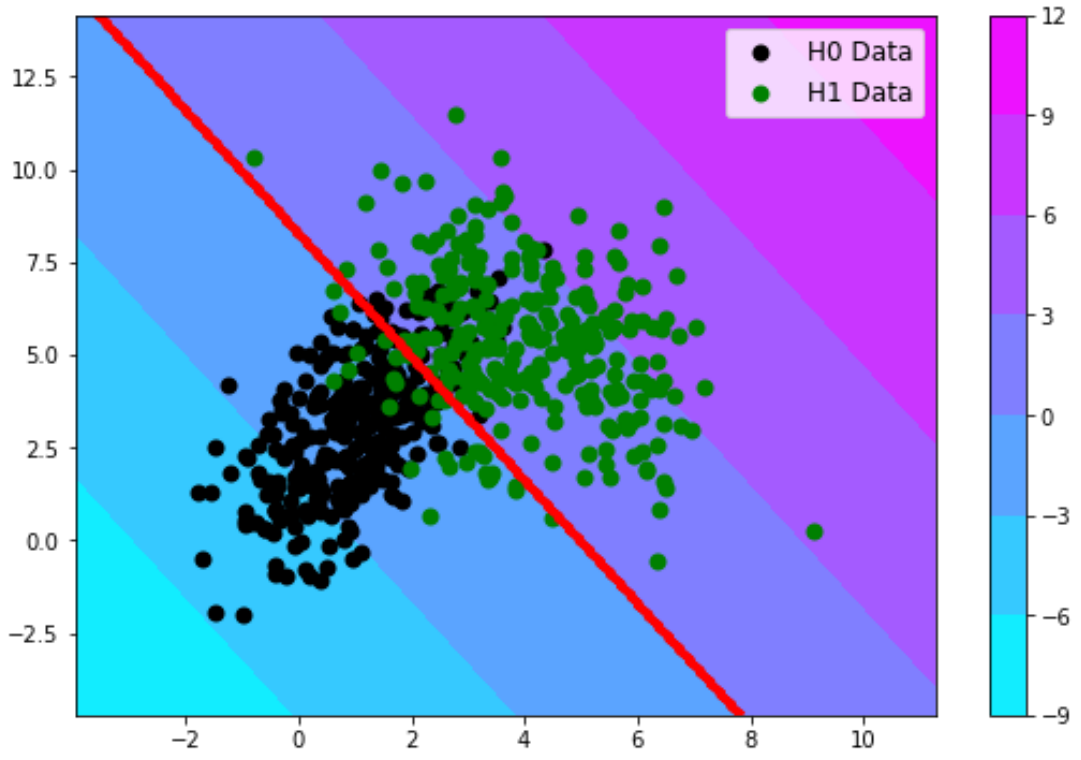
Question 4 (a) – Under the simplifying assumption that covariance matrices under both classes are the same but the features may be dependent, the decision statistic surface plot with both the training data and the decision boundary superimposed on top is shown below. The red line is the decision boundary.



Question 4 (b) – We assume the covariance matrices under both classes are the same, so that we can eliminate the quadratic term $x^T(\frac{-1}{2}\Sigma_i^{-1})x$ in the discriminant function $g(x)$, and get a linear decision boundary here. Although we assume the same covariance matrix under both classes, Class 0 and Class 1 data points still have different mean values ($\mu_0 \neq \mu_1$) as a prerequisite when we generate this dataset, which helps discriminate different classes based on $g(x)$.

Question 5 Solution:

Question 5 (a) – Under the simplifying assumptions that the covariance matrices under both classes are the same and the features are independent, the decision statistic surface with both the training data and the decision boundary plot is shown below. The red line is the decision boundary.



Question 5 (b) – We assume the covariance matrices under both classes are the same, so that we can eliminate the quadratic term $x^T(\frac{-1}{2}\Sigma_i^{-1})x$ in the discriminant function $g(x)$, and get a linear decision boundary here. Although we assume the same diagonal covariance matrix under both classes, Class 0 and Class 1 data points still have different mean values $(\mu_0 \neq \mu_1)$ as a prerequisite when we generate this dataset, which helps discriminate different classes based on $g(x)$.

Question 6 Solution:

When deciding how to model the data for a Bayes classifier, I would consider factors including the total number of features (dimensions) of the dataset, the total number of data points of the dataset, the covariance (variance & correlation) under each class, and the mean under each class. I specifically list the covariance and mean under each class as important factors because when I generate the dataset in Question 1, I realize that it is very important to have a basic understanding of whether the data points have some positive or negative correlation among different features, and to what extent the data points from different classes are overlapped.

To explain how I expect my classifier design choices to affect the resulting classifier. If the training data points from different classes do not overlap or scatter or correlate among features noticeably, then I prefer to make the simplifying assumptions that covariance matrices under both classes are the same and the features are independent, because I expect this simple assumption can still perform well in classification these two classes. If the training data points from different classes overlap noticeably, scatter significantly, and correlate among features explicitly, then I prefer to make no simplifying assumptions regarding the covariance structure, because I expect the classification task could be very difficult, and we might only get satisfying classification performance under the full covariance assumption that the features may be dependent, and the covariance matrices are unique.

Question 7 Solution:

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
%matplotlib inline
```

Reference the Packages/Toolboxes

I use numpy to process arraies and matrices.
I use pandas to read and process the original datasets.
I use matplotlib.pyplot to plot decision statistic surfaces.
To plot decision statistic surfaces, I also referenced the ECE580 course materials and the website below.
https://hackernoon.com/how-to-plot-a-decision-boundary-for-machine-learning-algorithms-in-python-3o1n3w07

[2]:
```
N0 = 300
N1 = 300
mean0 = np.array([1, 3])
mean1 = np.array([4, 5])
covariance0 = np.array([[1, 1], [1, 3]])
covariance1 = np.array([[2, -1], [-1, 4]])
```

[3]:
```
rng = np.random.default_rng()
X0 = rng.multivariate_normal(mean = mean0, cov = covariance0, size = N0)
X1 = rng.multivariate_normal(mean = mean1, cov = covariance1, size = N1)
X = np.concatenate((X0, X1))
y0 = np.zeros(N0)
y1 = np.ones(N1)
y = np.concatenate((y0, y1))
data = np.hstack((y.reshape((N0+N1, 1)), X))
print(data.shape)
print(data[0:3])
print(X0.shape)
print(X1.shape)
print(X.shape)
print(y0.shape)
print(y1.shape)
print(y.shape)
```

```
(600, 3)
[[ 0.          0.83371455  1.45174303]
 [ 0.          0.2785657   3.6704976 ]
 [ 0.         -0.74545888  0.45292756]]
(300, 2)
(300, 2)
(600, 2)
(300,)
(300,)
(600,)
```

[4]:
```
# print(np.mean(X0, axis = 0))
# print(np.mean(X1, axis = 0))
# print(np.mean(X, axis = 0))
# print(np.cov(X0, rowvar = False))
# print(np.cov(X1, rowvar = False))
# print(np.cov(X, rowvar = False))
# print(np.linalg.det(np.cov(X0, rowvar = False)))
```

[5]:
```
figure, axis = plt.subplots()
axis.scatter(X0[:,0], X0[:,1], label = "Class 0")
axis.scatter(X1[:,0], X1[:,1], label = "Class 1")
axis.set_xlabel("Feature 1", fontsize = 15)
axis.set_ylabel("Feature 2", fontsize = 15)
axis.set_title("Visualize Generated Dataset", fontsize = 15)
figure.set_size_inches(8, 6)
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.legend(fontsize = 12)
plt.show()
```

[6]:
```
# a = np.array([[1, 0, 0], [0, 2, 0], [0, 0, 3]])
# print(a)
# print(np.linalg.inv(a))
```

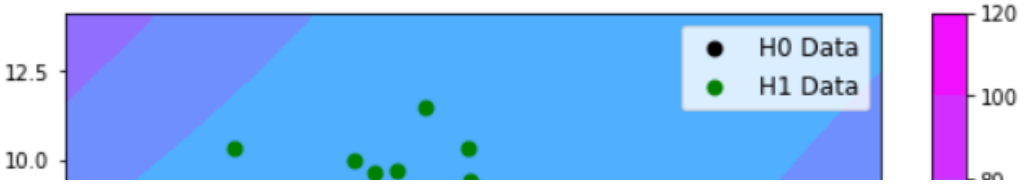
[7]:
```
def compute_weights (X, Xi, case) :
    d = X.shape[1]
    wi2 = np.zeros((d, d))
    wi1 = np.zeros(d)
    wi0 = 0

    covX = np.cov(X, rowvar = False)
    covX_simple = np.array([[covX[0,0], 0], [0, covX[1, 1]]])

    covXi = np.cov(Xi, rowvar = False)
    covXi_simple = np.array([[covXi[0,0], 0], [0, covXi[1,1]]])

    inv_covX = np.linalg.inv(covX)
    inv_covX_simple = np.linalg.inv(covX_simple)

    inv_covXi = np.linalg.inv(covXi)
    inv_covXi_simple = np.linalg.inv(covXi_simple)

    det_covX = np.linalg.det(covX)
    det_covXi = np.linalg.det(covXi)
    det_covXi_simple = np.linalg.det(covXi_simple)

    meanXi = np.mean(Xi, axis = 0)
    log_pri_wi = np.log(float(Xi.shape[0] / X.shape[0]))

    if case == "Q2" :
        # Xi
        wi2 = float(-1/2) * inv_covXi
        wi1 = np.dot(inv_covXi, meanXi)
        wi0 = float(-1/2) * np.dot(meanXi.T, np.dot(inv_covXi, meanXi)) - float(1/2) * np.log(det_covXi) + log_pri_wi
    if case == "Q3" :
        # Xi
        wi2 = float(-1/2) * inv_covXi_simple
        wi1 = np.dot(inv_covXi_simple, meanXi)
        wi0 = float(-1/2) * np.dot(meanXi.T, np.dot(inv_covXi_simple, meanXi)) - float(1/2) * det_covXi_simple + log_pri_wi
    if case == "Q4" :
        # X
        wi2 = np.zeros((d, d))
        wi1 = np.dot(inv_covX, meanXi)
        wi0 = float(-1/2) * np.dot(meanXi.T, np.dot(inv_covX, meanXi)) + log_pri_wi
    if case == "Q5" :
        # X
        wi2 = np.zeros((d, d))
        wi1 = np.dot(inv_covX_simple, meanXi)
        wi0 = float(-1/2) * np.dot(meanXi.T, np.dot(inv_covX_simple, meanXi)) + log_pri_wi
    return wi2, wi1, wi0
```

[8]:
```
def compute_gix (x, wi2, wi1, wi0) :
    gix = np.dot(x.T, np.dot(wi2, x)) + np.dot(wi1.T, x) + wi0
    return gix
```

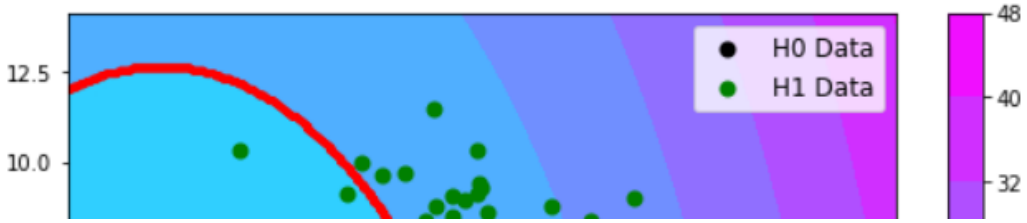
[9]:
```
def Bayes_Classifier (X, grid, case) :
    # Training
    N = X.shape[0]
    X0 = np.copy(X[0:int(N/2), :])
    X1 = np.copy(X[int(N/2):int(N-1), :])
    w02, w01, w00 = compute_weights(X, X0, case)
    w12, w11, w10 = compute_weights(X, X1, case)

    # Testing for decision statistic surfaces
    testN = grid.shape[0]
    y_proba = np.zeros(testN)
    for i in range(testN) :
        g0x = compute_gix(grid[i,:], w02, w01, w00)
        g1x = compute_gix(grid[i,:], w12, w11, w10)
        gx = g1x - g0x
        y_proba[i] = gx
    return y_proba
```

[10]:
```
def visualize_dss (data, case) :
    x1max = np.max(data[:, 1])
    x1min = np.min(data[:, 1])
    x2max = np.max(data[:, 2])
    x2min = np.min(data[:, 2])
    x1Range = x1max - x1min
    x2Range = x2max - x2min
    x1 = np.linspace((x1min - (0.2 * x1Range)), (x1max + (0.2 * x1Range)), 251)
    x2 = np.linspace((x2min - (0.2 * x2Range)), (x2max + (0.2 * x2Range)), 251)
    xx, yy = np.meshgrid(x1, x2)
    r1, r2 = xx.flatten(), yy.flatten()
    r1, r2 = r1.reshape((len(r1), 1)), r2.reshape((len(r2), 1))
    grid = np.hstack((r1, r2))

    yhat = Bayes_Classifier(data[:, 1:], grid, case)
    yhat_H1 = np.copy(yhat)

    zz = yhat_H1.reshape(xx.shape)
    figure, axis = plt.subplots()
    c1 = axis.contourf(xx, yy, zz, cmap = mpl.cm.cool)
    plt.colorbar(c1)

    c2 = axis.contour(xx, yy, zz >= 0.0, colors = "r", linewidths = 3)
    axis.scatter(data[data[:,0]==0., 1], data[data[:,0]==0., 2], label = "H0 Data", linewidths = 2, c = "k")
    axis.scatter(data[data[:,0]==1., 1], data[data[:,0]==1., 2], label = "H1 Data", linewidths = 2, c = "g")
    figure.set_size_inches((9, 6))
    plt.xticks(fontsize = 10)
    plt.yticks(fontsize = 10)
    plt.legend(fontsize = 12)
    plt.show()
    return None
```

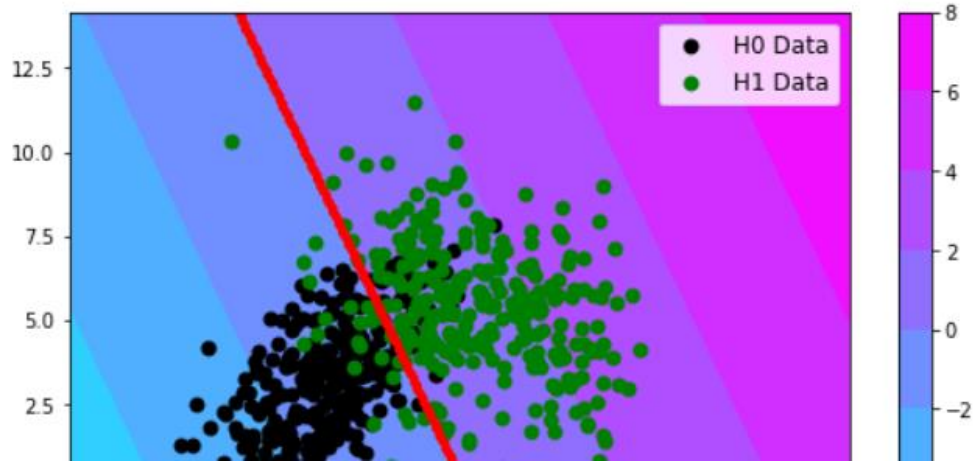
[11]:
```
visualize_dss(data, case = "Q2")
```



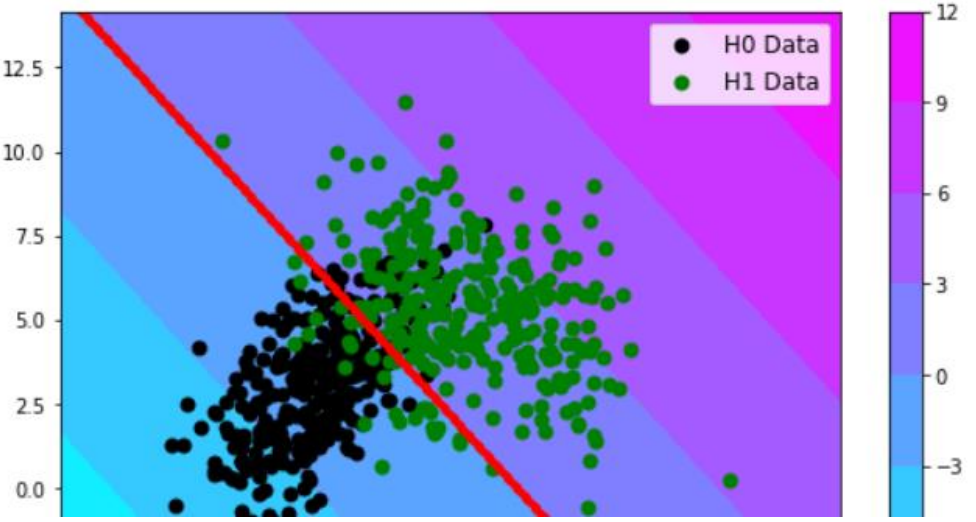
[12]:
```
visualize_dss(data, case = "Q3")
```



[13]:
```
visualize_dss(data, case = "Q4")
```



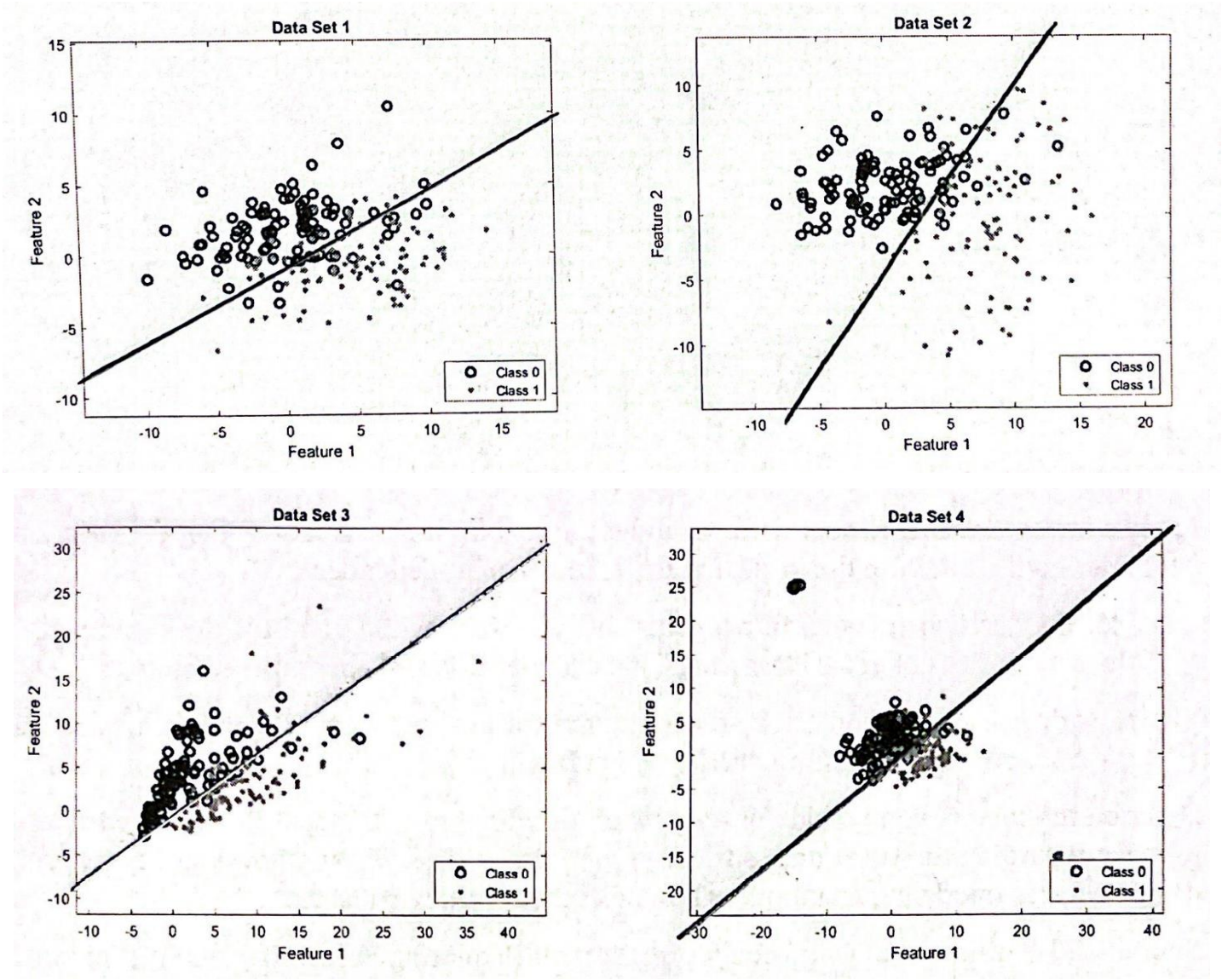
[14]:
```
visualize_dss(data, case = "Q5")
```



Comparing Linear Discriminant and Logistic Discriminant (and Bayes) Section

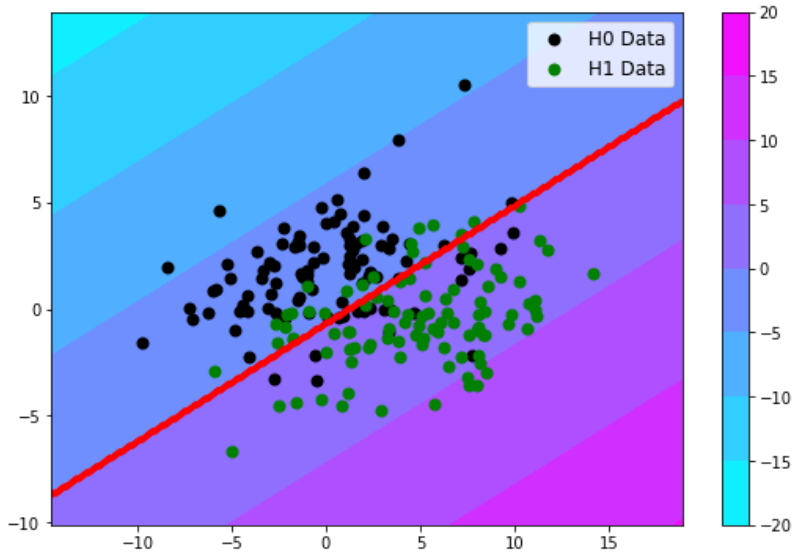
Question 8 Solution:

The qualitatively sketch of a "good" linear decision boundary for these 4 datasets is shown below.

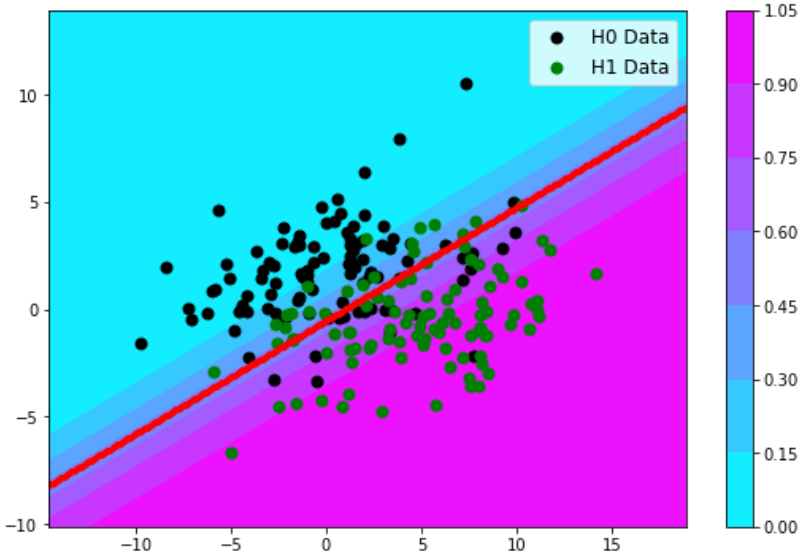


Question 9 Solution:

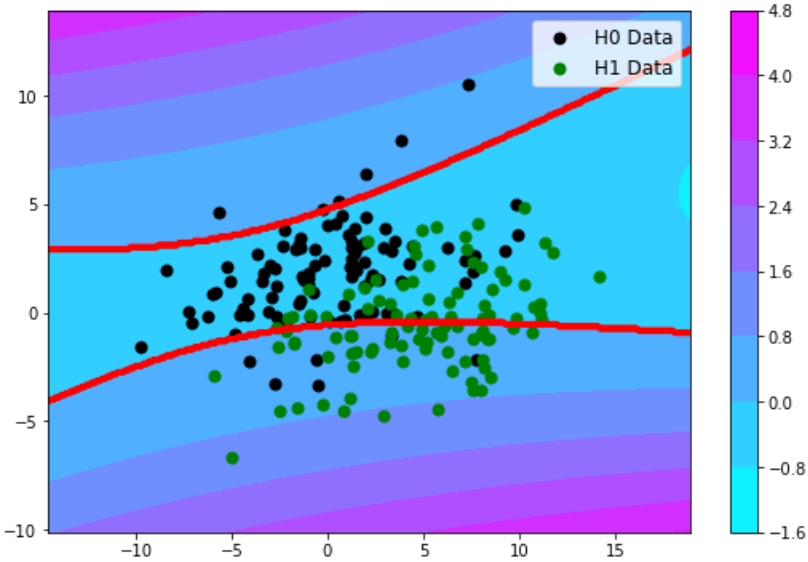
Question 9 (a): Linear Discriminant, Dataset 1. The red line is the decision boundary $\lambda(x) = 0$.



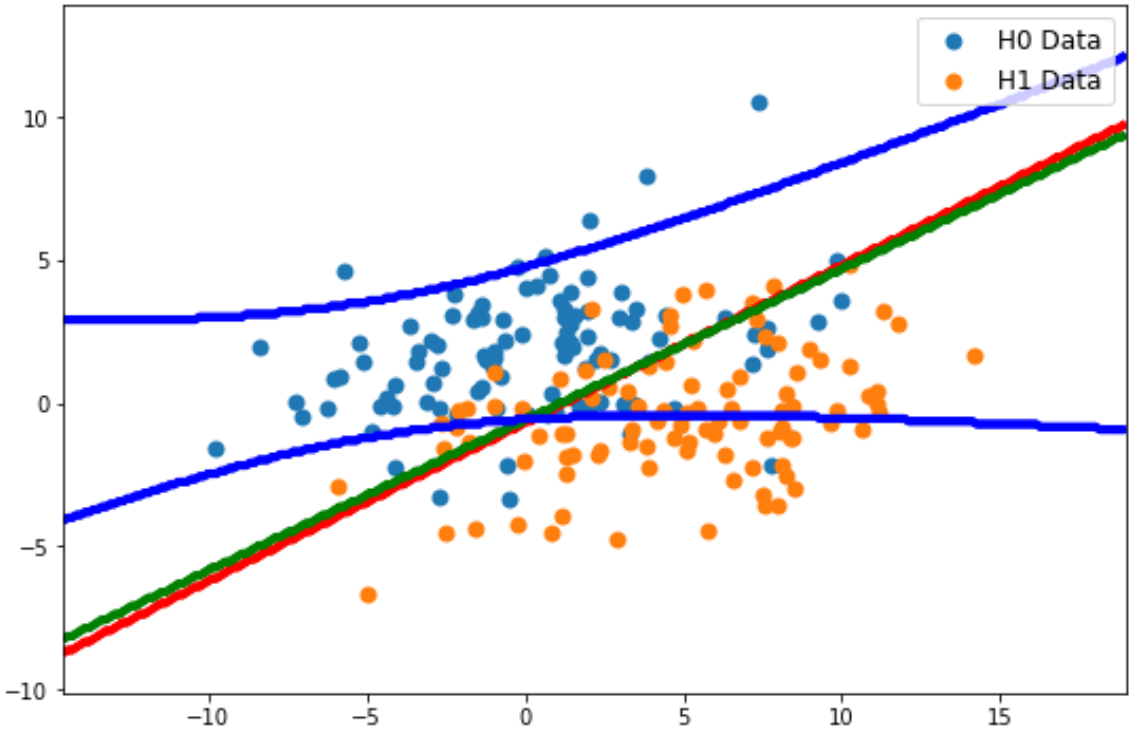
Question 9 (b): Logistic Discriminant, Dataset 1. The red line is the decision boundary $\lambda(x) = 0.5$.



Question 9 (c): Bayes Classifier, Dataset 1. The red line is the decision boundary $ln\lambda(x) = 0$.



Question 9 (d): Compare the three decision boundaries. Linear Discriminant has the red decision boundary, Logistic Regression has the green decision boundary, and Bayes Classifier has the blue decision boundary.

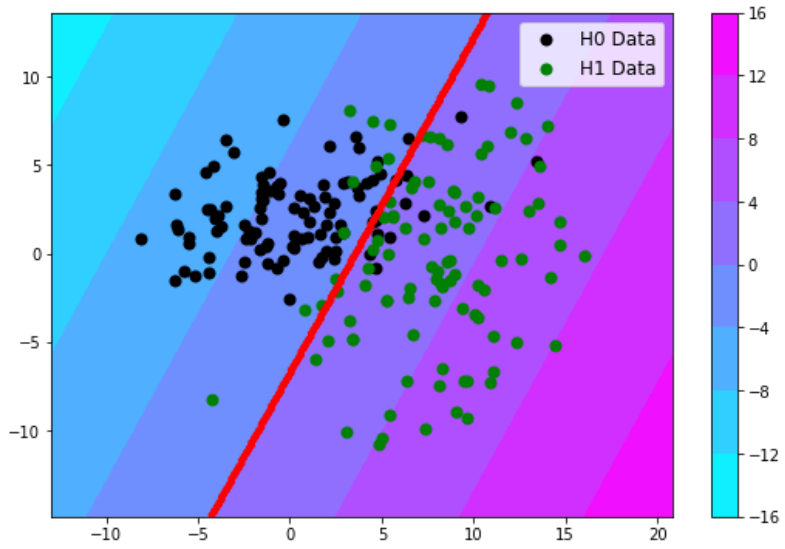


Question 9 (e): I think linear discriminant and logistic discriminant have very similar decision boundaries compared to my sketch, and Bayes classifier has very different decision boundary from my sketch.

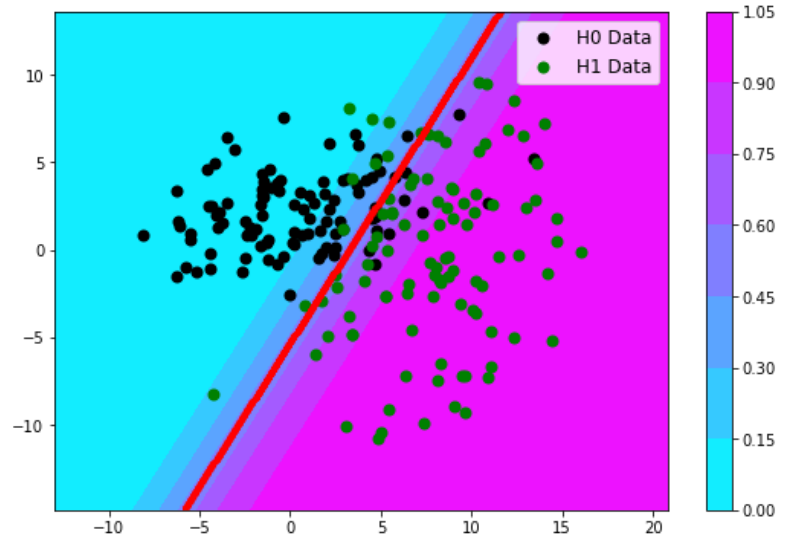
To explain why, the data is Gaussian with means for the two classes that are different as well as identical covariances, so dataset 1 is consistent with the assumptions underlying linear discriminant, and logistic discriminant compares posteriors to make a decision. These factors explain why linear discriminant and logistic discriminant have similar decision boundaries. We assume the features may be dependent and the covariance matrices for the two classes are distinct when applying the Bayes classifier, so that we cannot eliminate the quadratic term $x^T(\frac{-1}{2}\Sigma_i^{-1})x$ in the discriminant function $g(x)$. This is why we get a quadratic decision boundary curve (an oblique ellipse) for Bayes, which is very different from my sketch.

Question 10 Solution:

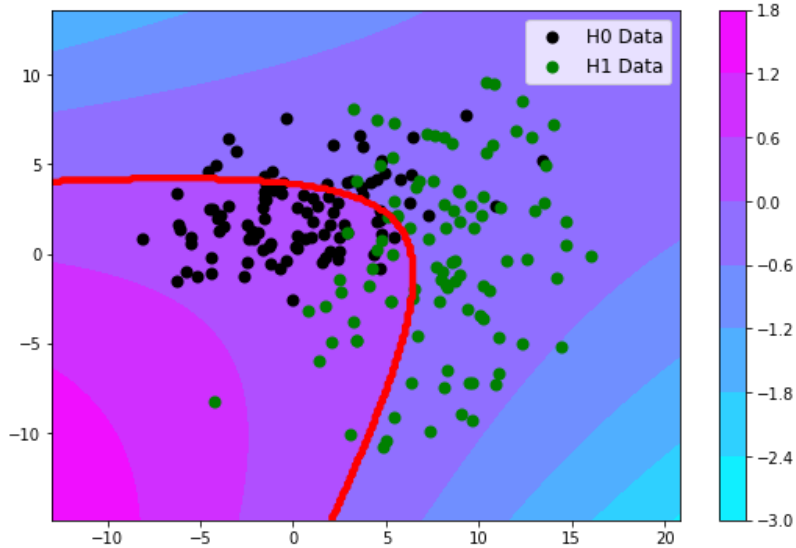
Question 10 (a): Linear Discriminant, Dataset 2. The red line is the decision boundary $\lambda(x) = 0$.



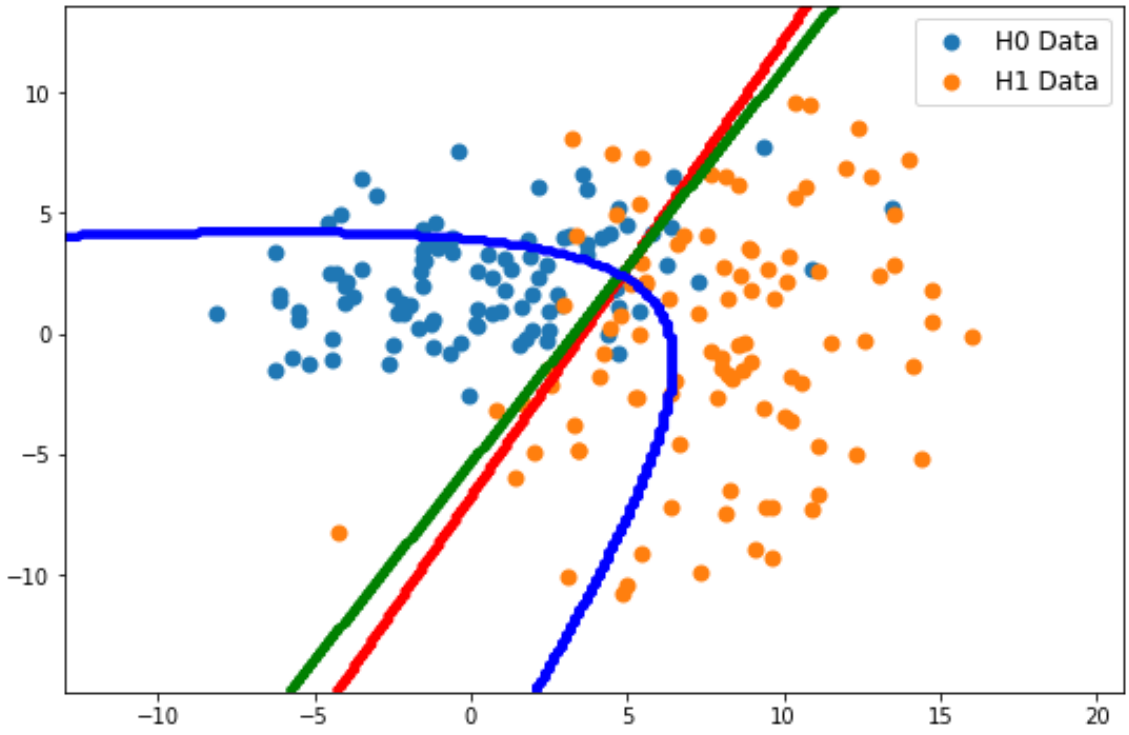
Question 10 (b): Logistic Discriminant, Dataset 2. The red line is the decision boundary $\lambda(x) = 0.5$.



Question 10 (c): Bayes Classifier, Dataset 2. The red line is the decision boundary $ln\lambda(x) = 0$.



Question 10 (d): Compare the three decision boundaries. Linear Discriminant has the red decision boundary, Logistic Discriminant has the green decision boundary, and Bayes Classifier has the blue decision boundary.

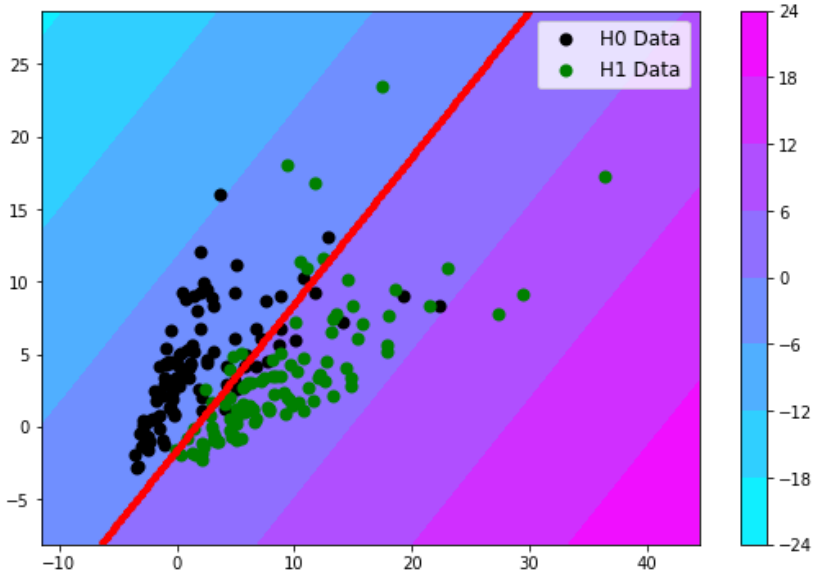


Question 10 (e): I think linear discriminant and logistic discriminant have very similar decision boundaries compared to my sketch, and Bayes classifier has very different decision boundary from my sketch.

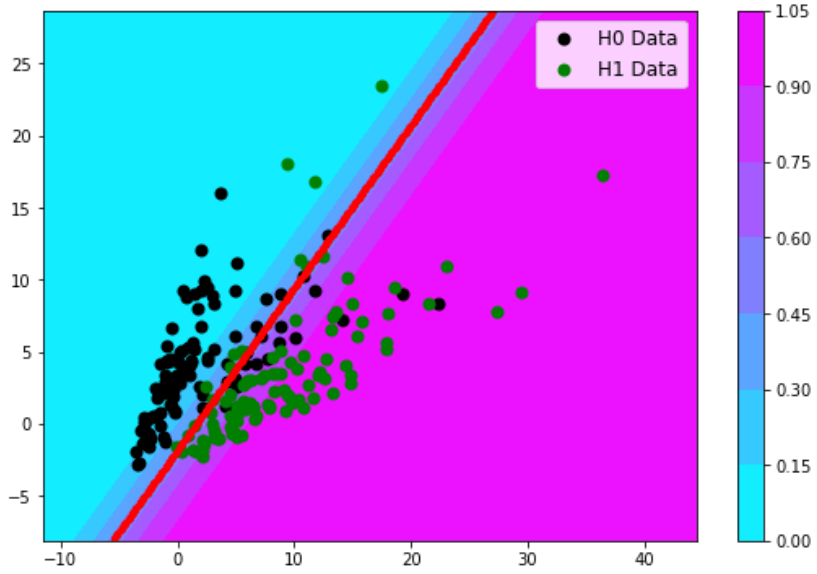
To explain why, although dataset 2 is not consistent with the assumptions underlying linear discriminant, the data is still Gaussian and has different means for the two classes, and these factors help linear discriminant and logistic discriminant (logistic discriminant compares posteriors to make a decision) make relatively good prediction. We assume the features may be dependent and the covariance matrices for the two classes are distinct when applying the Bayes classifier, so that we cannot eliminate the quadratic term $x^T(\frac{-1}{2}\Sigma_i^{-1})x$ in the discriminant function $g(x)$. This is why we get a quadratic decision boundary curve (an oblique ellipse) for Bayes, which is very different from my sketch.

Question 11 Solution:

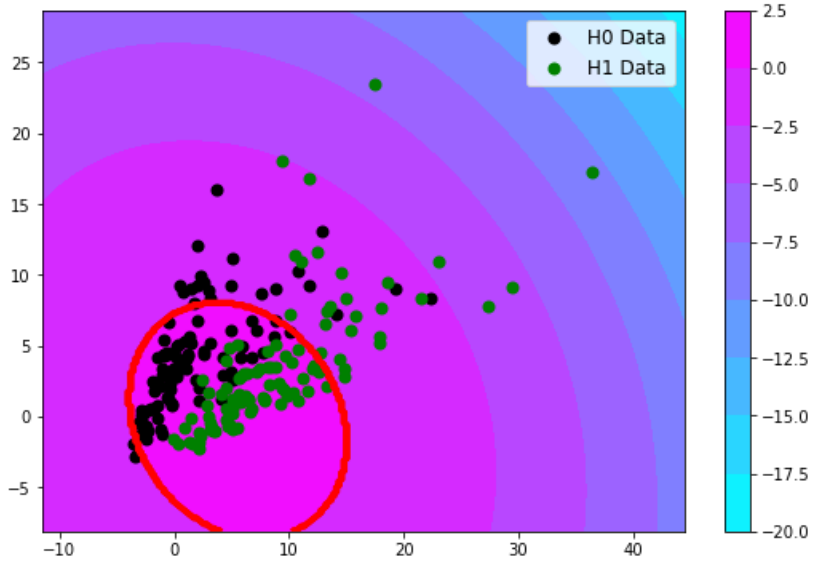
Question 11 (a): Linear Discriminant, Dataset 3. The red line is the decision boundary $\lambda(x) = 0$.



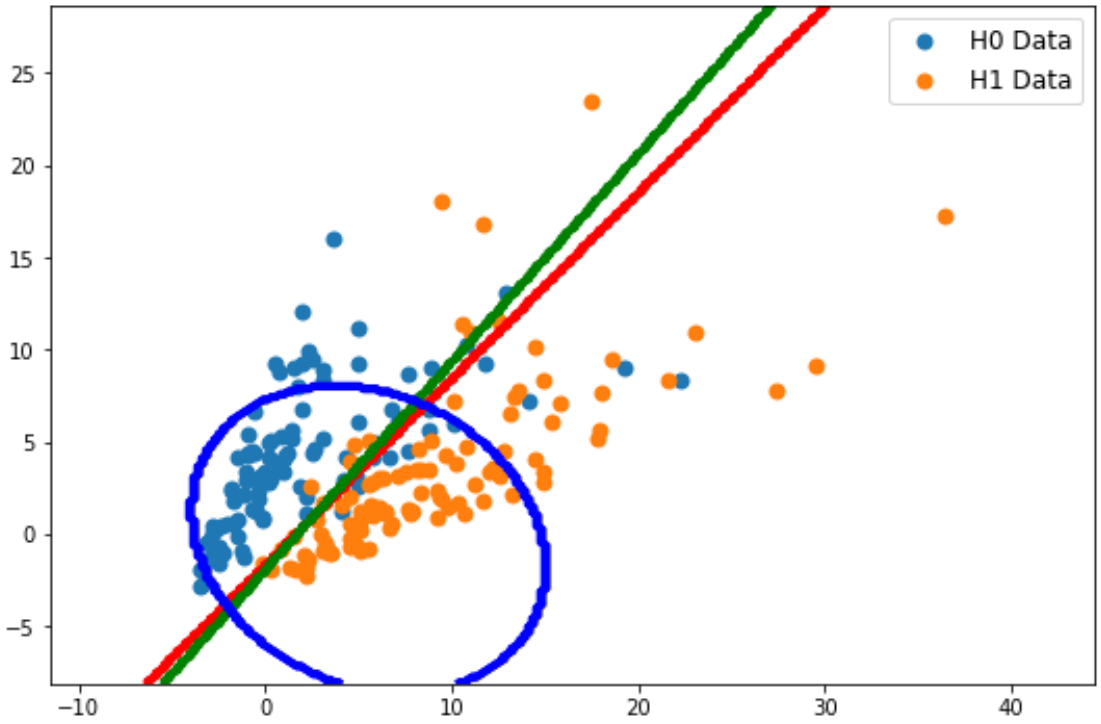
Question 11 (b): Logistic Discriminant, Dataset 3. The red line is the decision boundary $\lambda(x) = 0.5$.



Question 11 (c): Bayes Classifier, Dataset 3. The red line is the decision boundary $ln\lambda(x) = 0$.



Question 11 (d): Compare the three decision boundaries. Linear Discriminant has the red decision boundary, Logistic Discriminant has the green decision boundary, and Bayes Classifier has the blue decision boundary.

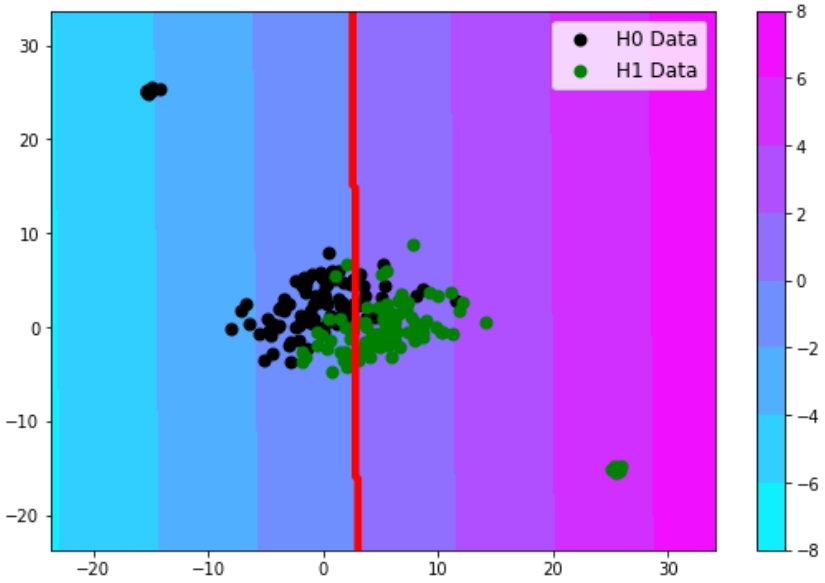


Question 11 (e): I think linear discriminant and logistic discriminant have very similar decision boundaries compared to my sketch, and Bayes classifier has very different decision boundary from my sketch.

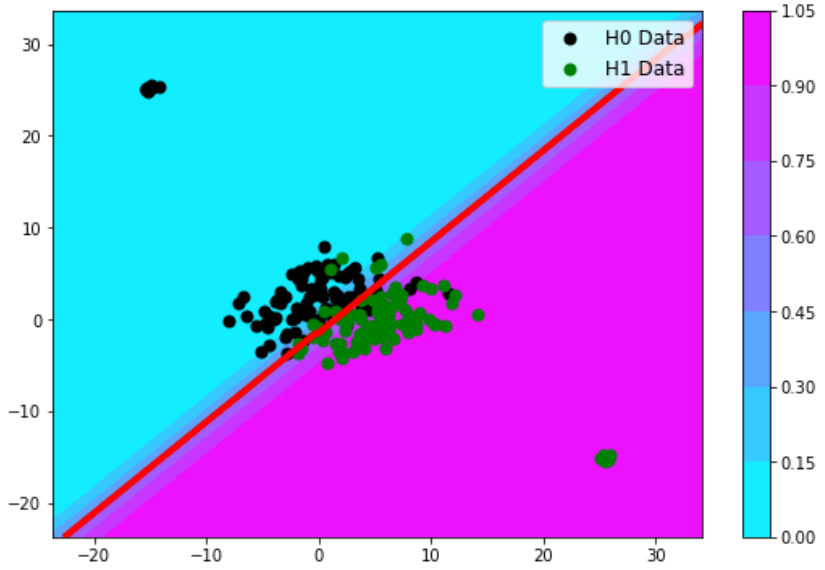
To explain why, although dataset 3 is not consistent with the assumptions underlying linear discriminant and the data is not Gaussian (log-normal instead), the data still has different means for the two classes, which could help linear discriminant and logistic discriminant (logistic discriminant compares posteriors to make a decision) make relatively good prediction. We assume the features may be dependent and the covariance matrices for the two classes are distinct when applying the Bayes classifier, so that we cannot eliminate the quadratic term $x^T(\frac{-1}{2}\Sigma_i^{-1})x$ in the discriminant function $g(x)$. This is why we get a quadratic decision boundary curve (an oblique ellipse) for Bayes, which is very different from my sketch.

Question 12 Solution:

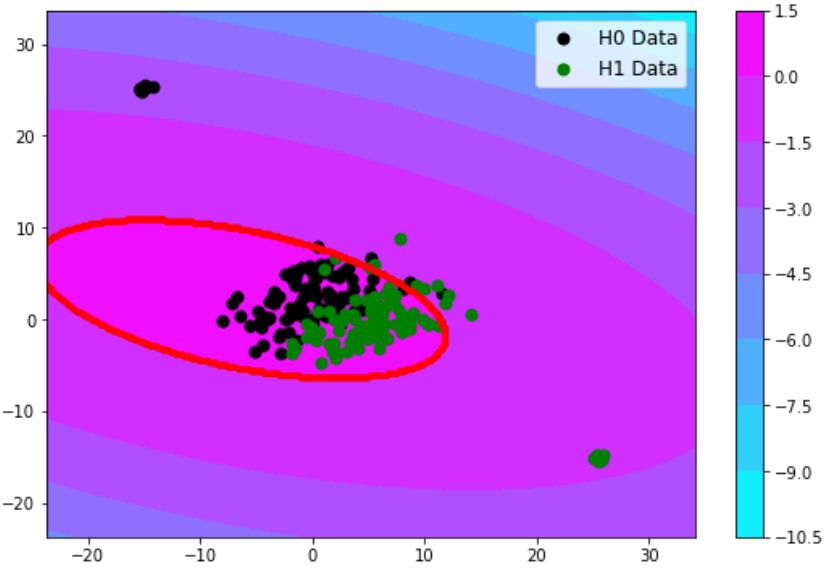
Question 12 (a): Linear Discriminant, Dataset 4. The red line is the decision boundary $\lambda(x) = 0$.



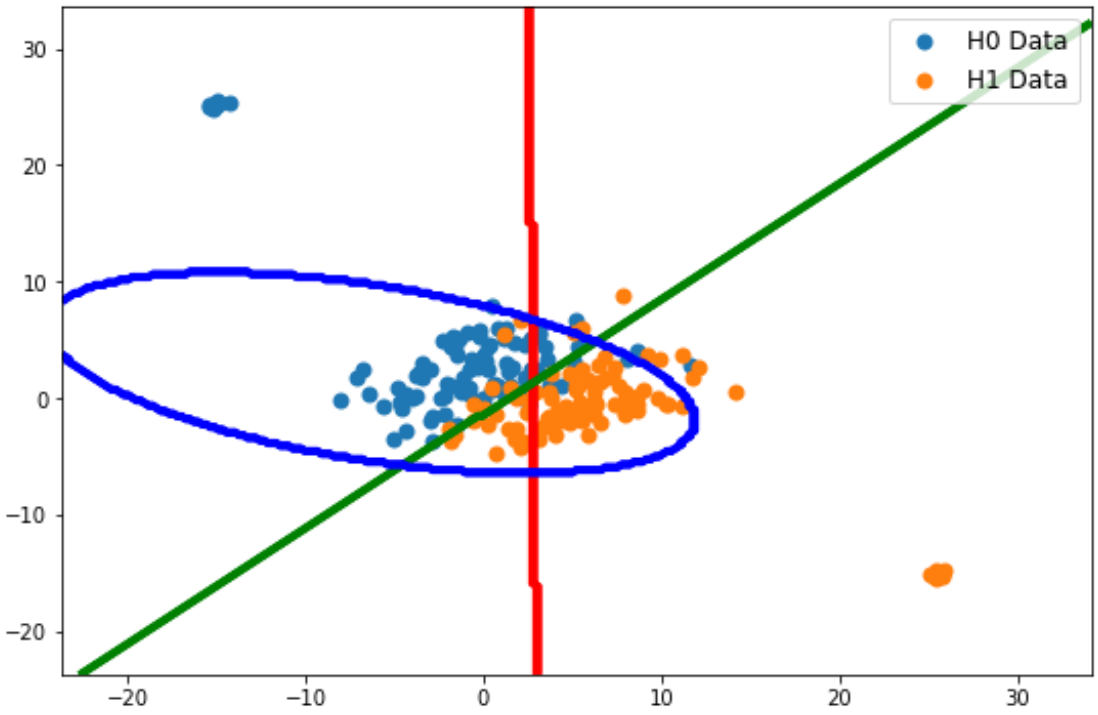
Question 12 (b): Logistic Discriminant, Dataset 4. The red line is the decision boundary $\lambda(x) = 0.5$.



Question 12 (c): Bayes Classifier, Dataset 4. The red line is the decision boundary $ln\lambda(x) = 0$.



Question 12 (d): Compare the three decision boundaries. Linear Discriminant has the red decision boundary, Logistic Discriminant has the green decision boundary, and Bayes Classifier has the blue decision boundary.



Question 12 (e): I think only logistic discriminant has a very similar decision boundary compared to my sketch. Both Linear Discriminant and Bayes Classifier have very different decision boundaries from my sketch.

To explain why, dataset 4 has some outliers for both class 0 and class 1, these outliers will distort the means and variances (or covariances including variances). Since means and variances are very important components for Linear Discriminant, if the means and variances for each class are distorted, then the Linear Discriminant classification performance will also be distorted. Logistic Discriminant compares posteriors to make a decision, so it will be less influenced by some outliers, which explains why it can still achieve relatively good classification performance. We assume the features may be dependent and the covariance matrices for the two classes are distinct when applying the Bayes classifier, so that we cannot eliminate the quadratic term $x^T(\frac{-1}{2}\Sigma_i^{-1})x$ in the discriminant function $g(x)$. This is why we get a quadratic decision boundary curve (an oblique ellipse) for Bayes, which is very different from my sketch. Besides, the means and variances (or covariances including variances) for each class are also very important components in Bayes discriminant function. If the means and variances are distorted by some outliers in both classes, then Bayes Classification will also be distorted, even with the quadratic decision boundary curve.

Question 13 Solution:

Firstly, I would visualize the data to check whether there are some outliers and whether the data points for the 2 classes seem to be linearly separable. If there are some outliers, then I would prefer Logistic Discriminant than Linear Discriminant which is heavily and negatively affected by outliers. If the data seems to be linearly separable, then I prefer Linear Discriminant or Logistic Discriminant than Bayes, because if we assume dependent features and distinct covariance matrices for the two classes for Bayes, we get a quadratic decision boundary curve (an oblique ellipse), which is inappropriate for linearly separable data.

Secondly, I would check the means and variances for each class data points. If the mean of class 0 and the mean of class 1 is very different, and the variances for both classes are small, then I can choose Linear Discriminant with confidence because Linear Discriminant optimizes the projection vector so that the difference in the means is large relative to the spread of each class, which can give me very good classification performance. Otherwise, if the mean for each class is close to each other, and the variance for each class is large, then I would prefer Logistic Discriminant which compares posteriors to make a classification decision. We have seen that Logistic Discriminant can maintain relatively good classification performance for all 4 datasets, so it is reasonable that we would prefer Logistic Discriminant than Logistic Discriminant and Bayes Classifier.

Question 14 Solution:

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
%matplotlib inline
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
```

Reference the Packages/Toolboxes

I use numpy to process arraies and matrices.
I use pandas to read and process the original datasets.
I use matplotlib.pyplot to plot decision statistic surfaces.
To plot decision statistic surfaces, I also referenced the ECE580 course materials and the website below.
https://hackernoon.com/how-to-plot-a-decision-boundary-for-machine-learning-algorithms-in-python-3o1n3w07
I use sklearn to implement Linear Discriminant Analysis.
I use sklearn to implement Logistic Regression.

[2]:
```
filename1 = "dataSet1.csv"
filename2 = "dataSet2.csv"
filename3 = "dataSet3.csv"
filename4 = "dataSet4.csv"
dataframe1 = pd.read_csv(filename1, header = None)
dataframe2 = pd.read_csv(filename2, header = None)
dataframe3 = pd.read_csv(filename3, header = None)
dataframe4 = pd.read_csv(filename4, header = None)
data1 = dataframe1.to_numpy()
data2 = dataframe2.to_numpy()
data3 = dataframe3.to_numpy()
data4 = dataframe4.to_numpy()
X1 = np.copy(data1[:, 1:])
X2 = np.copy(data2[:, 1:])
X3 = np.copy(data3[:, 1:])
X4 = np.copy(data4[:, 1:])
y1 = np.copy(data1[:, 0])
y2 = np.copy(data2[:, 0])
y3 = np.copy(data3[:, 0])
y4 = np.copy(data4[:, 0])
```

[3]:
```
def compute_weights (X, Xi, case) :
    d = X.shape[1]
    wi2 = np.zeros((d, d))
    wi1 = np.zeros(d)
    wi0 = 0
    covXi = np.cov(Xi, rowvar = False)
    inv_covXi = np.linalg.inv(covXi)
    det_covXi = np.linalg.det(covXi)
    meanXi = np.mean(Xi, axis = 0)
    log_pri_wi = np.log(float(Xi.shape[0] / X.shape[0]))
    if case == "Q2" :
        wi2 = float(-1/2) * inv_covXi
        wi1 = np.dot(inv_covXi, meanXi)
        wi0 = float(-1/2) * np.dot(meanXi.T, np.dot(inv_covXi, meanXi)) - float(1/2) * np.log(det_covXi) + log_pri_wi
    return wi2, wi1, wi0
```

[4]:
```
def compute_gix (x, wi2, wi1, wi0) :
    gix = np.dot(x.T, np.dot(wi2, x)) + np.dot(wi1.T, x) + wi0
    return gix

def Bayes_Classifier (X, grid, case) :
    # Training
    N = X.shape[0]
    X0 = np.copy(X[0:int(N/2), :])
    X1 = np.copy(X[int(N/2):int(N-1), :])
    w02, w01, w00 = compute_weights(X, X0, case)
    w12, w11, w10 = compute_weights(X, X1, case)

    # Testing for decision statistic surfaces
    testN = grid.shape[0]
    y_proba = np.zeros(testN)
    for i in range(testN) :
        g0x = compute_gix(grid[i,:], w02, w01, w00)
        g1x = compute_gix(grid[i,:], w12, w11, w10)
        gx = g1x - g0x
        y_proba[i] = gx
    return y_proba
```

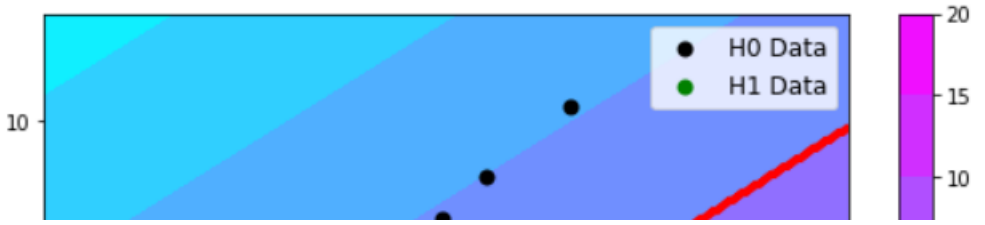
[5]:
```
def compare_boundaries (data) :
    x1max = np.max(data[:, 1])
    x1min = np.min(data[:, 1])
    x2max = np.max(data[:, 2])
    x2min = np.min(data[:, 2])
    x1Range = x1max - x1min
    x2Range = x2max - x2min
    x1 = np.linspace((x1min - (0.2 * x1Range)), (x1max + (0.2 * x1Range)), 251)
    x2 = np.linspace((x2min - (0.2 * x2Range)), (x2max + (0.2 * x2Range)), 251)
    xx, yy = np.meshgrid(x1, x2)
    r1, r2 = xx.flatten(), yy.flatten()
    r1, r2 = r1.reshape((len(r1), 1)), r2.reshape((len(r2), 1))
    grid = np.hstack((r1, r2))

    X = data[:, 1:]
    y = data[:, 0]
    model_1 = LinearDiscriminantAnalysis()
    model_1.fit(X, y)
    yhat_1 = model_1.decision_function(grid)
    yhat_H1_1 = np.copy(yhat_1)
    zz_1 = yhat_H1_1.reshape(xx.shape)
    threshold_1 = 0.0

    model_2 = LogisticRegression()
    model_2.fit(X, y)
    yhat_2 = model_2.predict_proba(grid)
    yhat_H1_2 = np.copy(yhat_2[:, 1])
    zz_2 = yhat_H1_2.reshape(xx.shape)
    threshold_2 = 0.5

    yhat_3 = Bayes_Classifier(data[:, 1:], grid, case = "Q2")
    yhat_H1_3 = np.copy(yhat_3)
    zz_3 = yhat_H1_3.reshape(xx.shape)
    threshold_3 = 0.0
```

```
    figure, axis = plt.subplots()
    # c1 = axis.contourf(xx, yy, zz, cmap = mpl.cm.cool)
    # plt.colorbar(c1)
    c1 = axis.contour(xx, yy, zz_1 >= threshold_1, colors = "r", linewidths = 3)
    c2 = axis.contour(xx, yy, zz_2 >= threshold_2, colors = "g", linewidths = 3)
    c3 = axis.contour(xx, yy, zz_3 >= threshold_3, colors = "b", linewidths = 3)
    axis.scatter(data[data[:,0]==0., 1], data[data[:,0]==0., 2], label = "H0 Data", linewidths = 2)
    axis.scatter(data[data[:,0]==1., 1], data[data[:,0]==1., 2], label = "H1 Data", linewidths = 2)
    figure.set_size_inches((9, 6))
    plt.xticks(fontsize = 10)
    plt.yticks(fontsize = 10)
    plt.legend(fontsize = 12)
    plt.show()
    return None
```

[6]:
```
def visualize_dss (data, mode) :
    x1max = np.max(data[:, 1])
    x1min = np.min(data[:, 1])
    x2max = np.max(data[:, 2])
    x2min = np.min(data[:, 2])
    x1Range = x1max - x1min
    x2Range = x2max - x2min
    x1 = np.linspace((x1min - (0.2 * x1Range)), (x1max + (0.2 * x1Range)), 251)
    x2 = np.linspace((x2min - (0.2 * x2Range)), (x2max + (0.2 * x2Range)), 251)
    xx, yy = np.meshgrid(x1, x2)
    r1, r2 = xx.flatten(), yy.flatten()
    r1, r2 = r1.reshape((len(r1), 1)), r2.reshape((len(r2), 1))
    grid = np.hstack((r1, r2))

    X = data[:, 1:]
    y = data[:, 0]

    if mode == "Linear" :
        model = LinearDiscriminantAnalysis()
        model.fit(X, y)
        yhat = model.decision_function(grid)
        yhat_H1 = np.copy(yhat)
        threshold = 0.0
    if mode == "Logistic" :
        model = LogisticRegression()
        model.fit(X, y)
        yhat = model.predict_proba(grid)
        yhat_H1 = np.copy(yhat[:, 1])
        threshold = 0.5
    if mode == "Bayes" :
        yhat = Bayes_Classifier(data[:, 1:], grid, case = "Q2")
        yhat_H1 = np.copy(yhat)
        threshold = 0.0
```

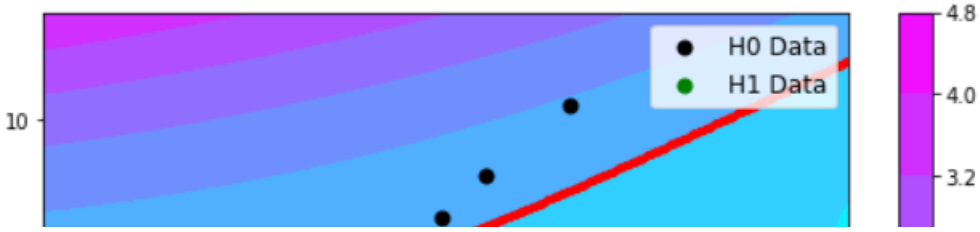
```
zz = yhat_H1.reshape(xx.shape)
figure, axis = plt.subplots()
c1 = axis.contourf(xx, yy, zz, cmap = mpl.cm.cool)
plt.colorbar(c1)
c2 = axis.contour(xx, yy, zz >= threshold, colors = "r", linewidths = 3)
axis.scatter(data[data[:,0]==0., 1], data[data[:,0]==0., 2], label = "H0 Data", linewidths = 2, c = "k")
axis.scatter(data[data[:,0]==1., 1], data[data[:,0]==1., 2], label = "H1 Data", linewidths = 2, c = "g")
figure.set_size_inches((9, 6))
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)
plt.legend(fontsize = 12)
plt.show()
return None
```

[7]:
```
visualize_dss(data1, mode = "Linear")
```

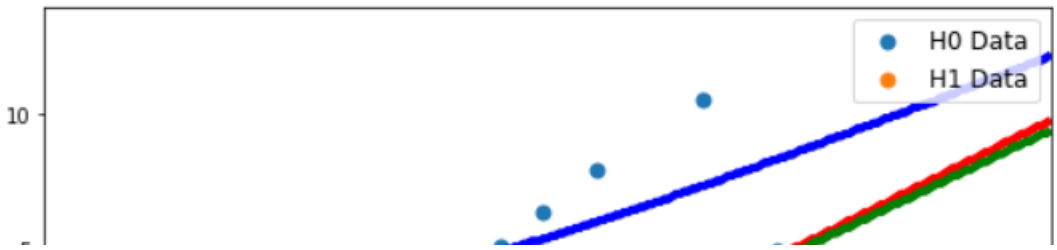


[8]:
```
visualize_dss(data1, mode = "Logistic")
```

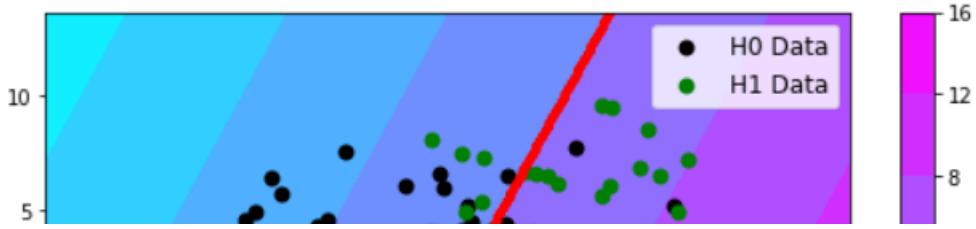
[9]:
```
visualize_dss(data1, mode = "Bayes")
```



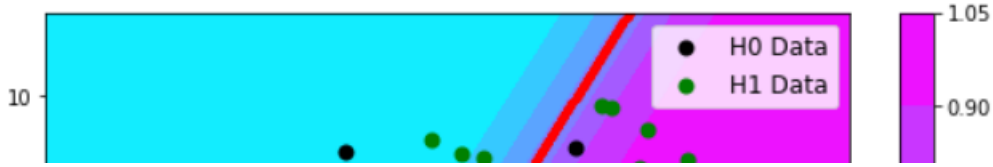
[10]:
```
# Compare all for data1
compare_boundaries(data1)
```



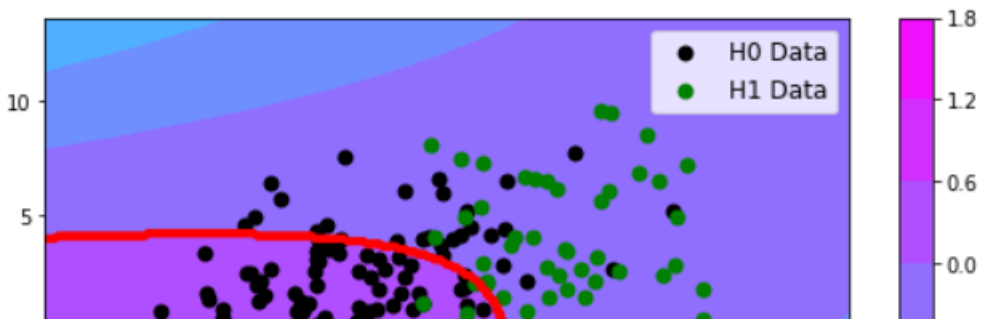
[11]:
```
visualize_dss(data2, mode = "Linear")
```



[12]:
```
visualize_dss(data2, mode = "Logistic")
```

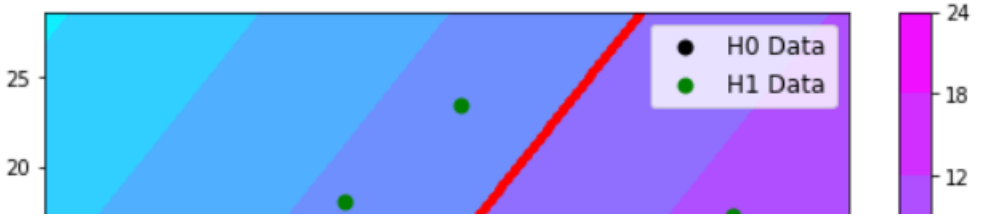


[13]:
```
visualize_dss(data2, mode = "Bayes")
```

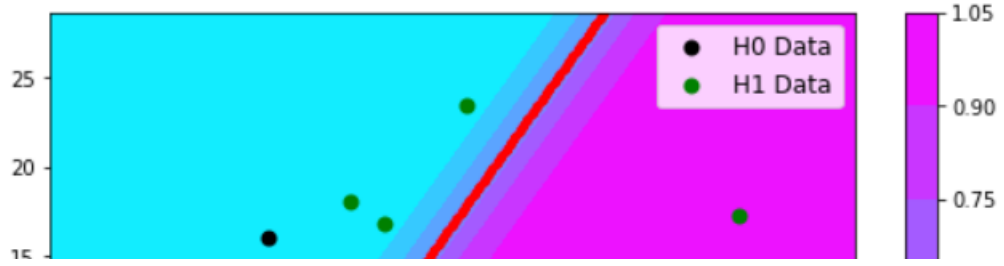


[14]:
```
# Compare all for data2
compare_boundaries(data2)
```

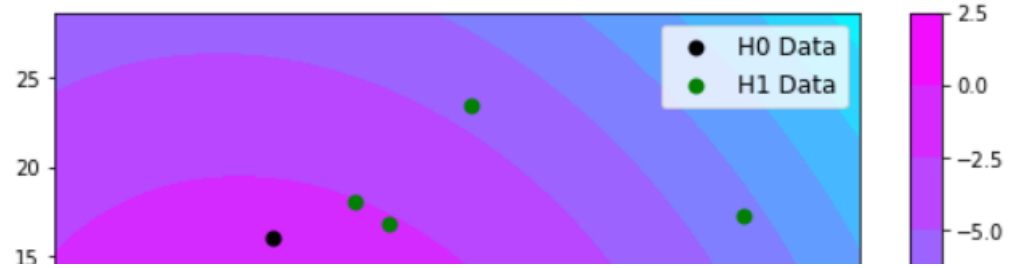
[15]:
```
visualize_dss(data3, mode = "Linear")
```



[16]:
```
visualize_dss(data3, mode = "Logistic")
```

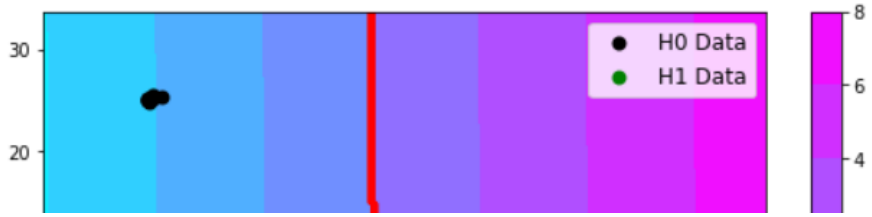


[17]:
```
visualize_dss(data3, mode = "Bayes")
```



[18]:
```
# Compare all for data3
compare_boundaries(data3)
```

[19]:
```
visualize_dss(data4, mode = "Linear")
```



[20]:
```
visualize_dss(data4, mode = "Logistic")
```

[21]:
```
visualize_dss(data4, mode = "Bayes")
```

[22]:
```
# Compare all for data4
compare_boundaries(data4)
```