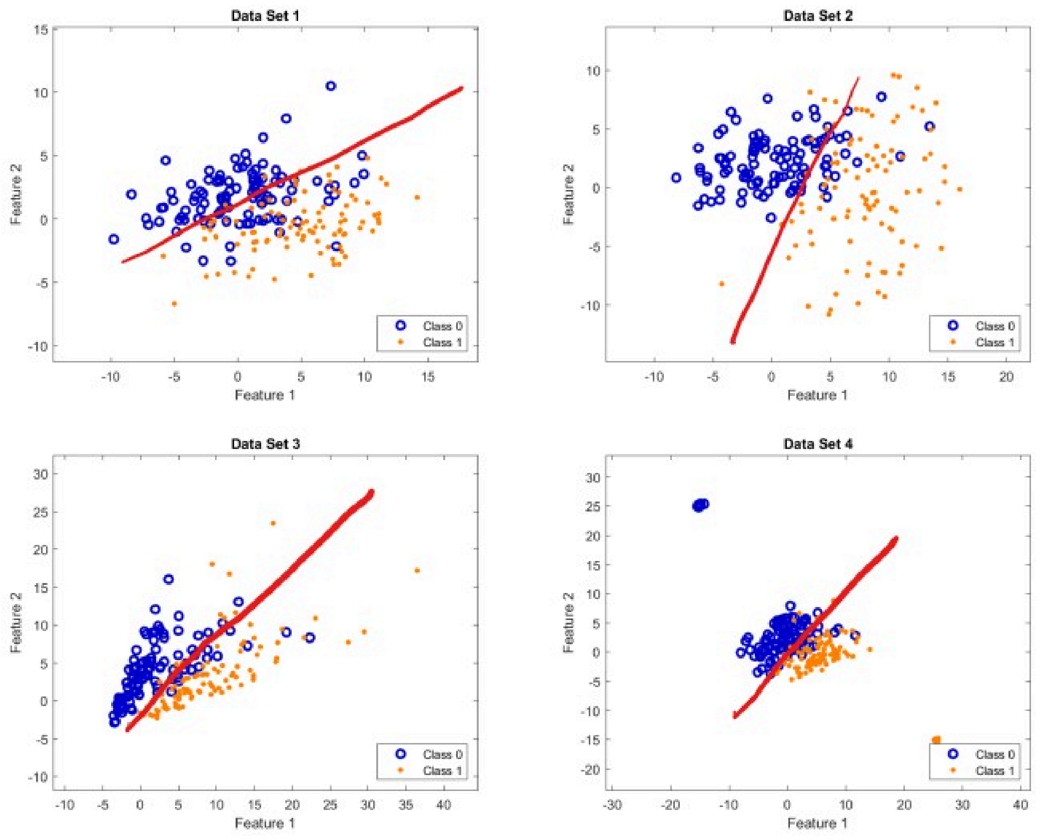
Comparing Linear SVM, Linear Discriminant, and Naive Bayes

1)



```
In [ ]: import numpy as np

        data1 = np.genfromtxt('dataSet1.csv',delimiter=',')
        data2 = np.genfromtxt('dataSet2.csv',delimiter=',')
        data3 = np.genfromtxt('dataSet3.csv',delimiter=',')
        data4 = np.genfromtxt('dataSet4.csv',delimiter=',')
```

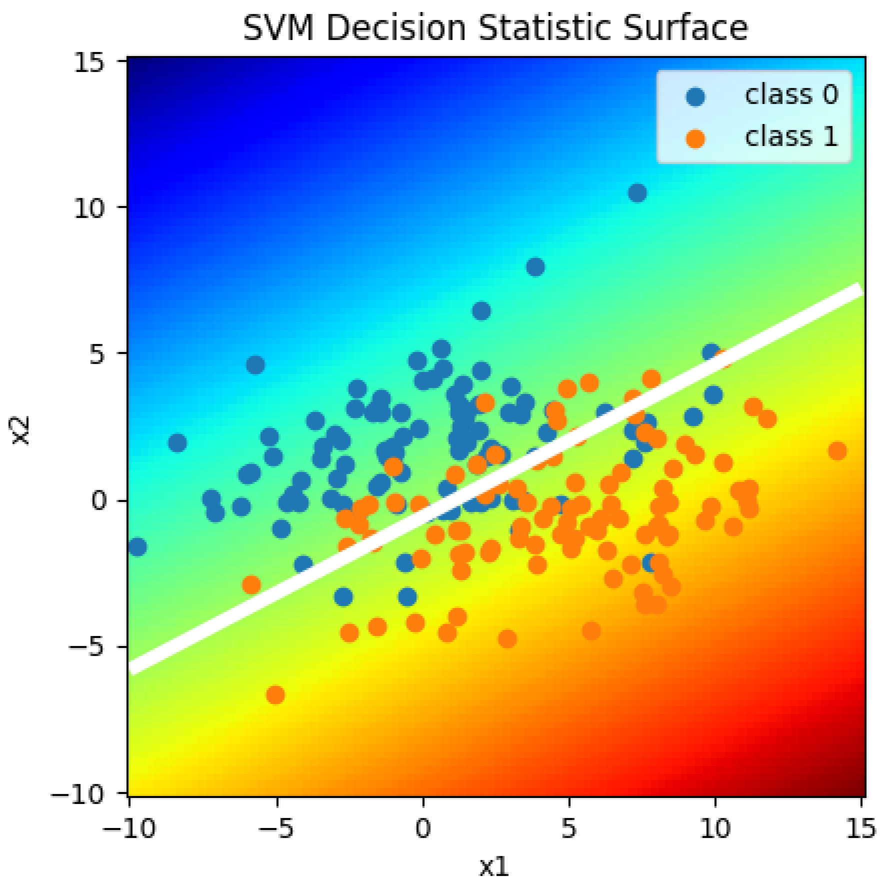
```
In [ ]: import matplotlib.pyplot as plt
        from matplotlib.lines import Line2D
        from sklearn.svm import SVC
        from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
        from sklearn.decomposition import PCA
        from sklearn.naive_bayes import GaussianNB




        class SVM_LDA_PCA_NB():
            def __init__(self, dataset):
```

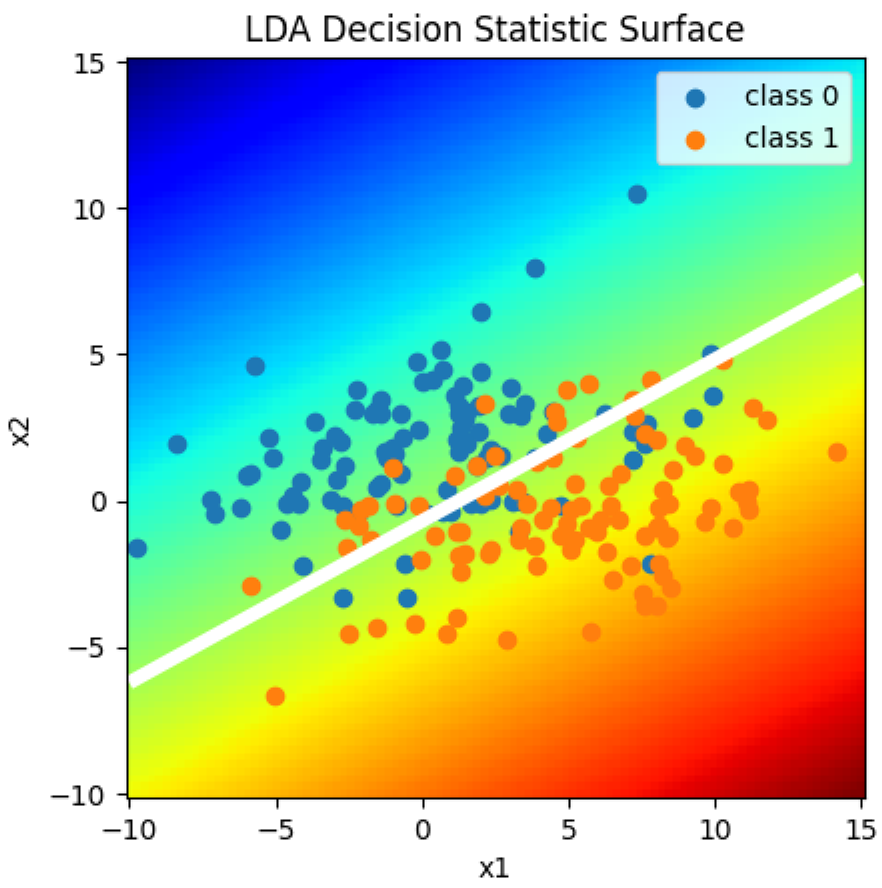
```
        minmin = np.floor(min(min(dataset[:,1]),min(dataset[:,2]))).astype(int)
        maxmax = np.ceil(max(max(dataset[:,1]),max(dataset[:,2]))).astype(int)
        data_x_range = np.linspace(minmin,maxmax,100)

        self.dataset = dataset
        self.xx, self.yy = np.meshgrid(data_x_range,data_x_range)
        self.data_grid = np.c_[self.xx.ravel(), self.yy.ravel()]
        self.data_c0 = dataset[dataset[:,0]==0, : ]
        self.data_c1 = dataset[dataset[:,0]==1, : ]


    def plot_decision_stat_surface(self, pred, title):
        plt.pcolormesh(self.xx, self.yy, pred.reshape(self.xx.shape), cmap="jet")
        plt.contour(self.xx, self.yy, pred.reshape(self.xx.shape), levels=[0], colo
        plt.scatter(self.data_c0[:,1],self.data_c0[:,2],label="class 0")
        plt.scatter(self.data_c1[:,1],self.data_c1[:,2],label="class 1")
        plt.title(title)
        plt.xlabel("x1")
        plt.ylabel("x2")
        plt.legend()
        plt.gca().set_aspect('equal')
        plt.show()


    def svm(self):
        svm = SVC(kernel='linear')
        svm.fit(self.dataset[:,1:3], self.dataset[:,0])
        self.svm_pred = svm.decision_function(self.data_grid)
        self.plot_decision_stat_surface(self.svm_pred,"SVM Decision Statistic Surfa

    def lda(self):
        lda = LinearDiscriminantAnalysis()
        lda.fit(self.dataset[:,1:3], self.dataset[:,0])
        self.lda_pred = lda.decision_function(self.data_grid)
        self.plot_decision_stat_surface(self.lda_pred,"LDA Decision Statistic Surfa

    # pca nb
    def pca_nb(self):
        pca = PCA(n_components=2)
        newx = pca.fit_transform(self.dataset[:,1:3])
        gnb = GaussianNB()
        gnb.fit(newx, self.dataset[:,0])

        pca_naiveBayes_pred = gnb.predict_proba(pca.transform(self.data_grid))
        self.pca_naiveBayes_loglikelihood_ratio = np.log(pca_naiveBayes_pred[:,1] /
        self.plot_decision_stat_surface(self.pca_naiveBayes_loglikelihood_ratio,"PC

    def three_decision_boundaries(self):
        plt.contour(self.xx, self.yy, self.svm_pred.reshape(self.xx.shape), levels=
        plt.contour(self.xx, self.yy, self.lda_pred.reshape(self.xx.shape), levels=
        plt.contour(self.xx, self.yy, self.pca_naiveBayes_loglikelihood_ratio.resha
        plt.scatter(self.data_c0[:,1],self.data_c0[:,2],label="class 0")
        plt.scatter(self.data_c1[:,1],self.data_c1[:,2],label="class 1")

        contour_legend_elements = [
            Line2D([0], [0], color='blue', linewidth=2, label='SVM'),
            Line2D([0], [0], color='green', linewidth=2, label='LDA'),
            Line2D([0], [0], color='red', linewidth=2, label='PCA + NB')
        ]
        scatter_legend_elements = [
            Line2D([0], [0], marker='o', color='blue', linestyle='', label='Class 0
            Line2D([0], [0], marker='o', color='orange', linestyle='', label='Class
        ]
        legend_elements = contour_legend_elements + scatter_legend_elements

        plt.title("SVM vs LDA vs PCA+NB")
        plt.xlabel("x1")
        plt.ylabel("x2")
        plt.legend(handles=legend_elements)
        ax = plt.gca()
        ax.set_aspect('equal')
        plt.show()
```

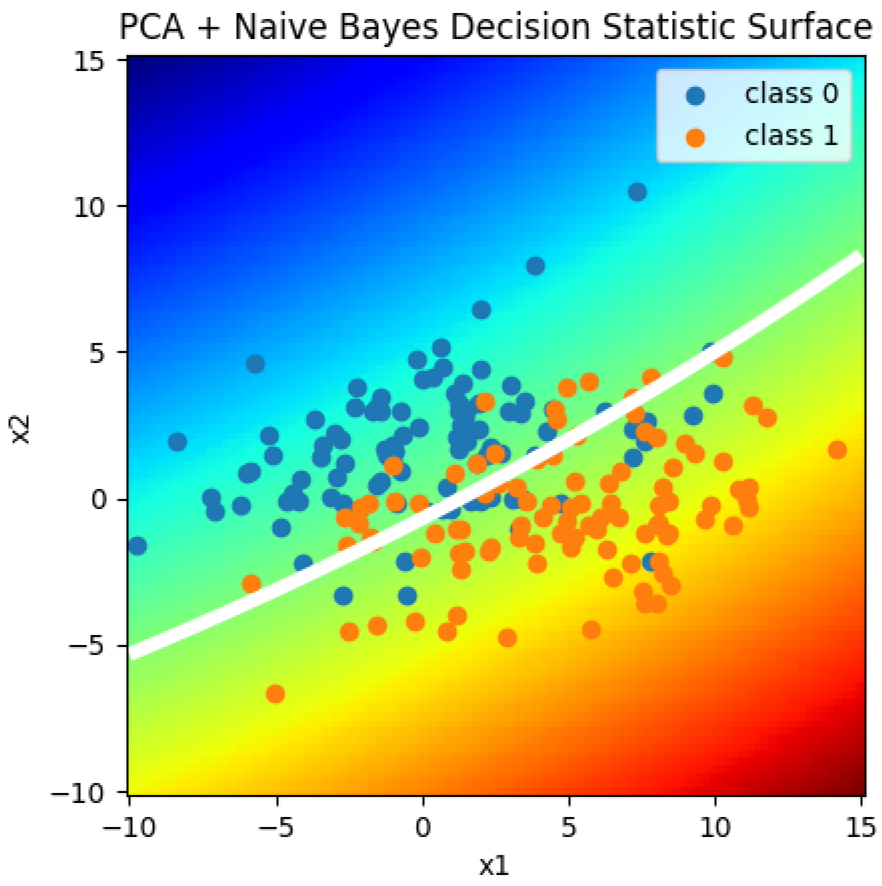
2)

2a)

In [ ]:
```
data1_models = SVM_LDA_PCA_NB(data1)
data1_models.svm()
data1_models.lda()
data1_models.pca_nb()
```

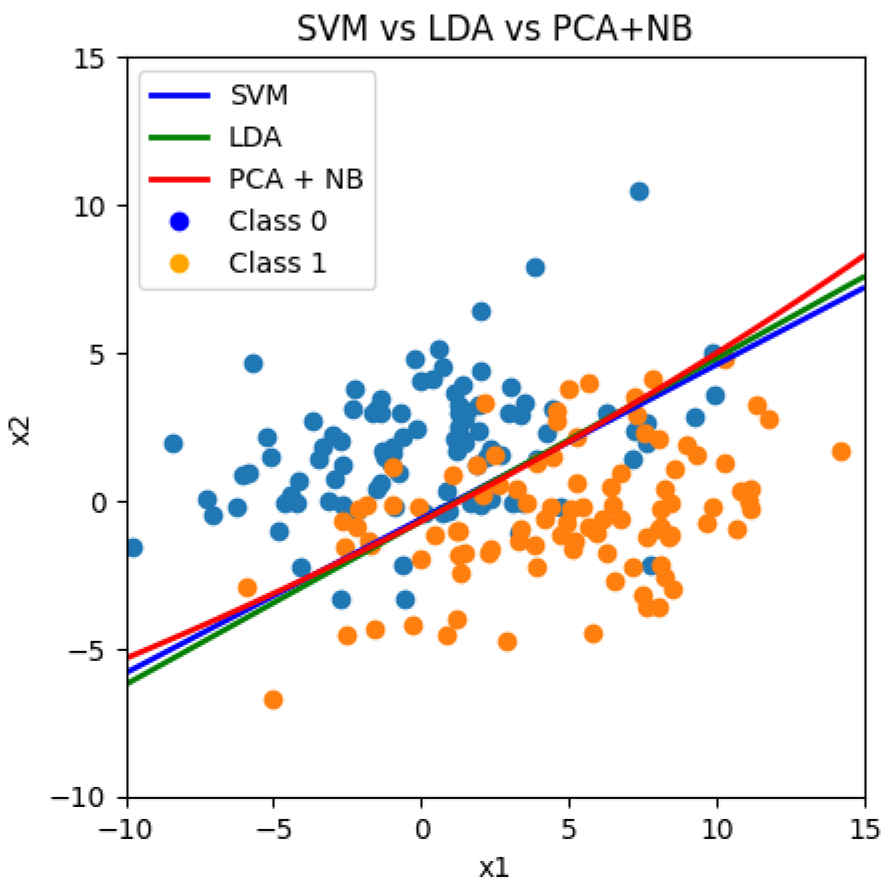



LDA Decision Statistic Surface
class 0
class 1
x2
x1
15
10
5
0
−5
−10
−10
−5
0
5
10
15


PCA + Naive Bayes Decision Statistic Surface
class 0
class 1
x2
x1
15
10
5
0
−5
−10
−10
−5
0
5
10
15

2b)

```
In [ ]: data1_models.three_decision_boundaries()
```



2c)

All 3 decision boundaries were similar to my sketch. The decision boundary that was most different was the PCA + NB one since it was the only non linear boundary and my boundary was linear. Nevertheless, the non linear boundary produced wasn't too far from being linear, so I'd say that the boundary still closely resembles my sketch.

The SVM classifier resembles my sketch because the model assumes that my data is linearly separable. This means that the SVM classifier is looking for some linear vector that can separate the two classes. Since my sketch is also a linear boundary between the two blobs of data, the SVM classifier resembles my sketch.

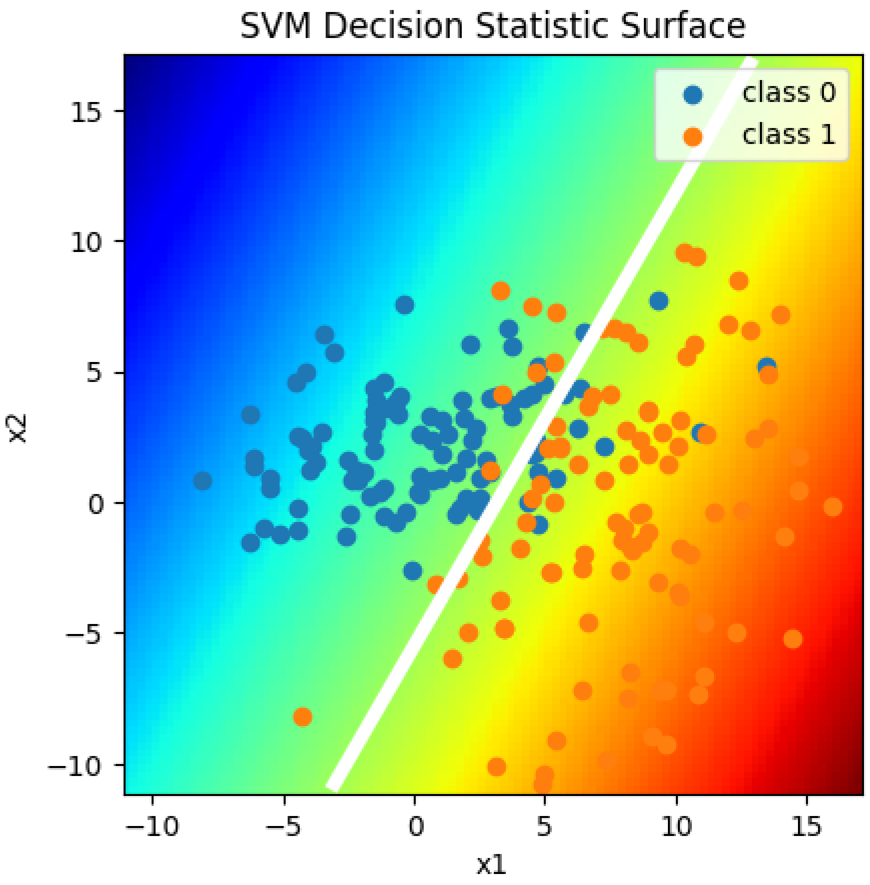
The LDA classifier resembles my sketch because the model assumes that the data is normally distributed with identical covariances. This assumption about the data is true so the boundary created by LDA classifier is a line between the two blobs of data, which is similar to my sketch.

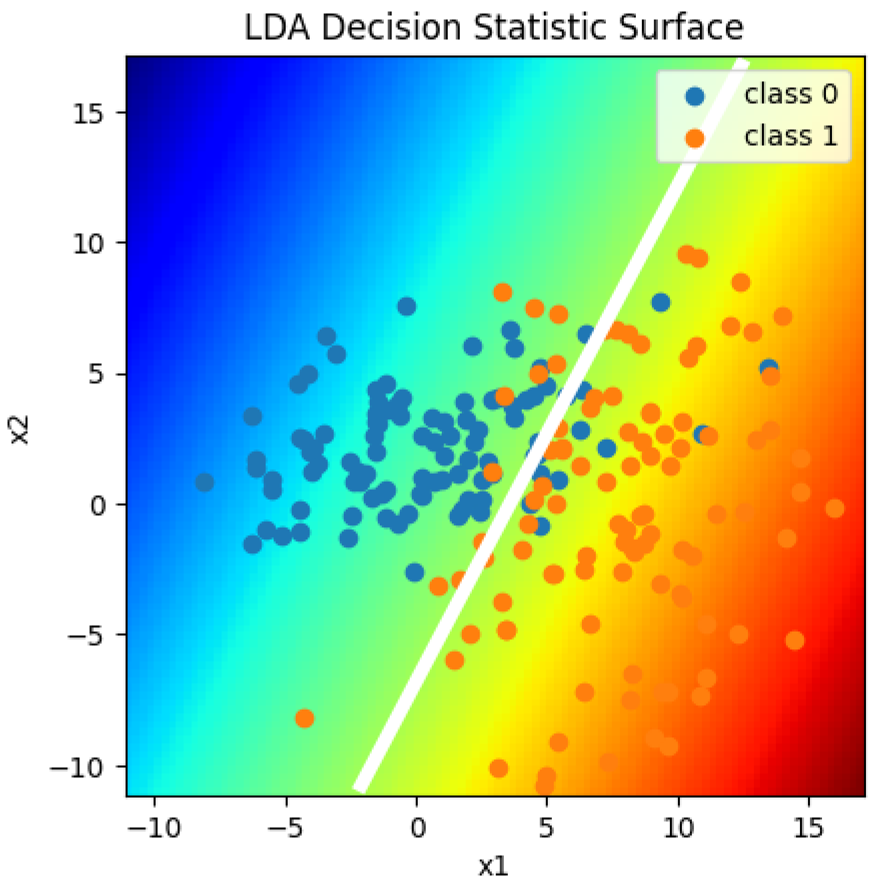
The PCA + NB classifier resembles my sketch because the model first finds two vectors that when the data is projected onto them, the variance is maximized. Since the data has identical covariances and distinct means, the approximated PDFs of the classes along one of the two PCA vectors has little overlap. Thus, the boundary created by the PCA + NB classifier resembles mine since the gaussian PDFs along the second PCA dimension has little overlap.

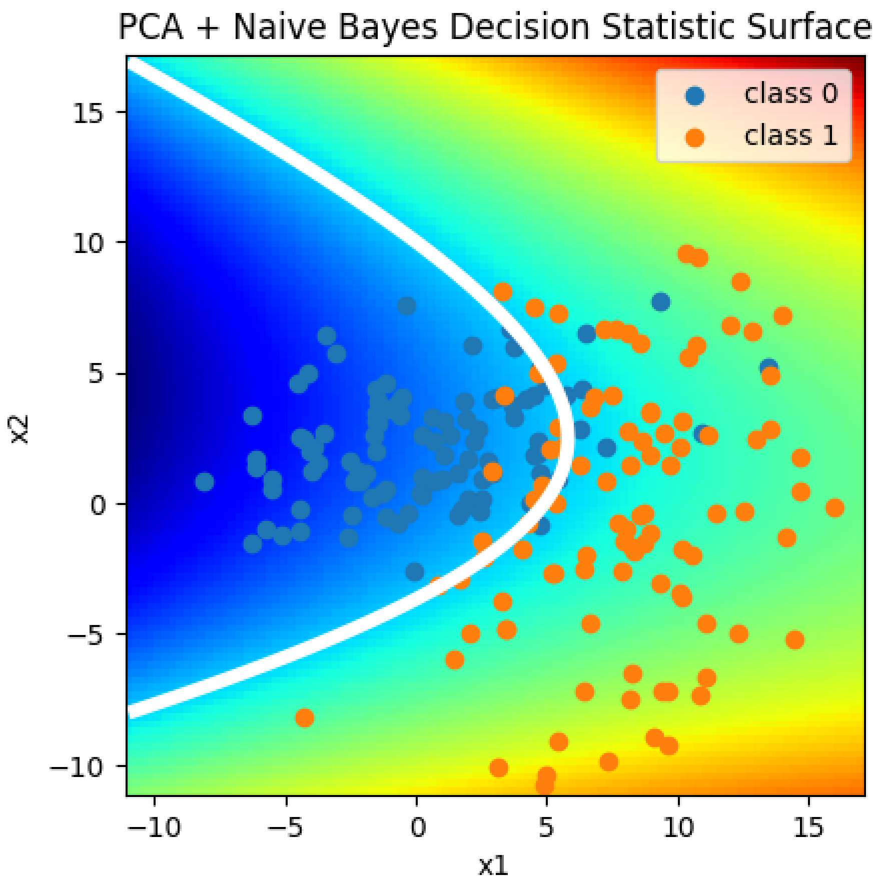
3)

3a)

```
In [ ]: data2_models = SVM_LDA_PCA_NB(data2)
        data2_models.svm()
        data2_models.lda()
        data2_models.pca_nb()
```

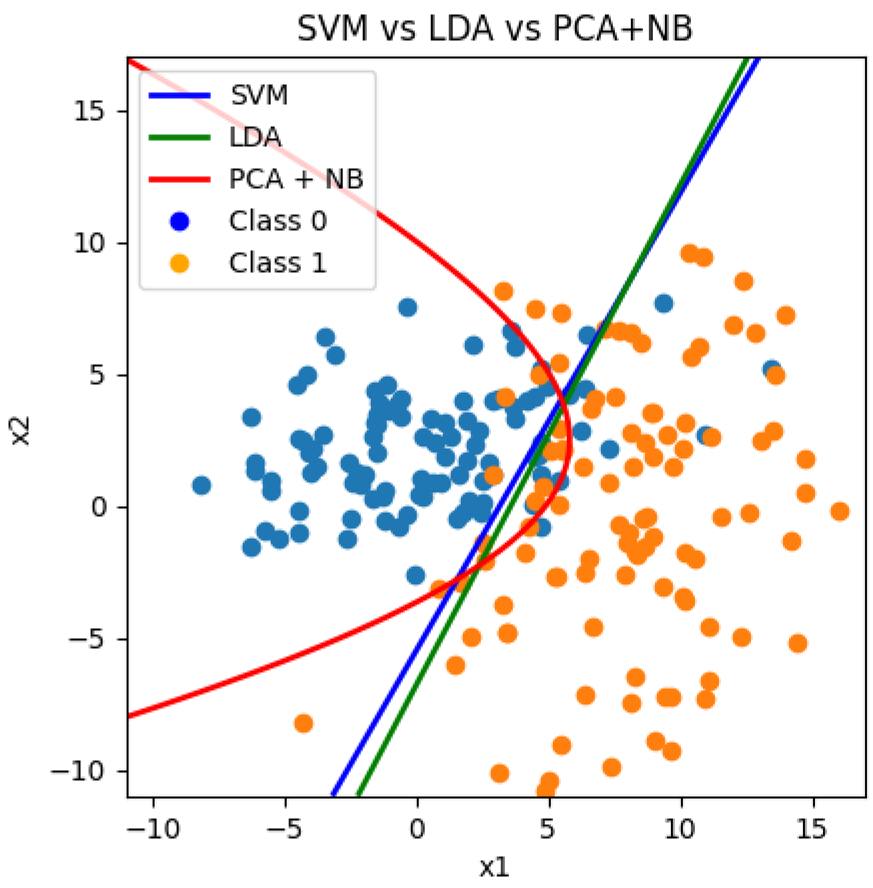



LDA Decision Statistic Surface
class 0
class 1
x2
x1
15
10
5
0
−5
−10
−10
−5
0
5
10
15


PCA + Naive Bayes Decision Statistic Surface
class 0
class 1
x2
x1
15
10
5
0
−5
−10
−10
−5
0
5
10
15

3b)

```
In [ ]: data2_models.three_decision_boundaries()
```



3c)

The SVM and LDA decision boundaries were similar to my sketch. The boundary produced by PCA + NB was wildly different as it produced a curve around the blue (class 0) data points.

The SVM classifier resembles my sketch because the model assumes that my data is linearly separable. This means that the SVM classifier is looking for some linear vector that can separate the two classes. Since my sketch is also a linear boundary between the two blobs of data, the SVM classifier resembles my sketch.

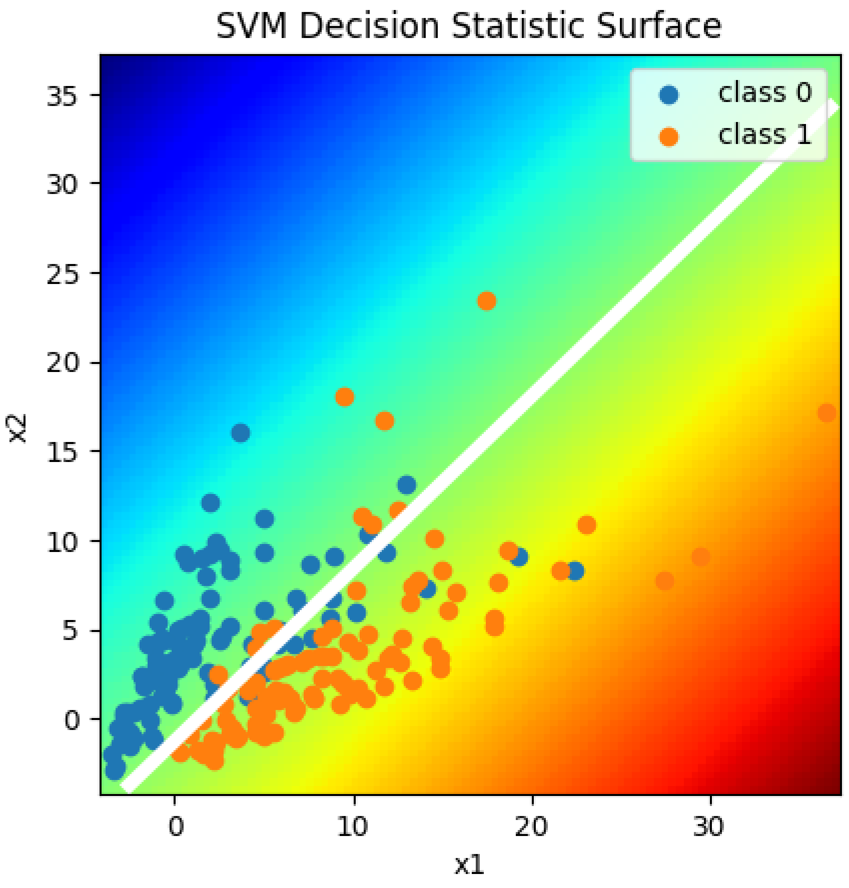
The LDA classifier resembles my sketch because the model assumes that the data is normally distributed with identical covariances. This assumption about the data is partially true as the data is gaussian but the covariances are different. However, since the direction of the covariances is similar, the LDA classifier does a fine job, as it is able to identify a vector that when the data is projected onto it, there is a lot of discrimination between the two classes.

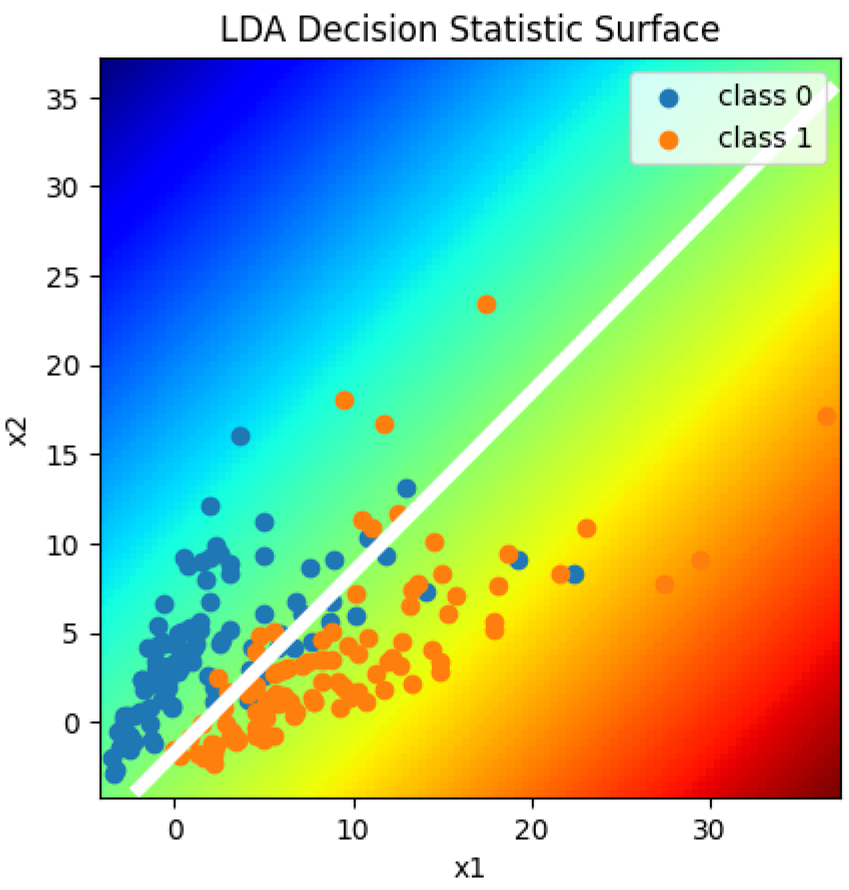
The PCA + NB classifier does not look like my sketch. The curve towards the left happens because the data has different covariances. Since the orange (class 1) data has has a higher covariance, the data is more spread than the blue (class 0) data. This results in the model being trained to believe that in spaces where there is no definitive orange/blue data, the model will assume that orange is more likely since it has a higher spread.

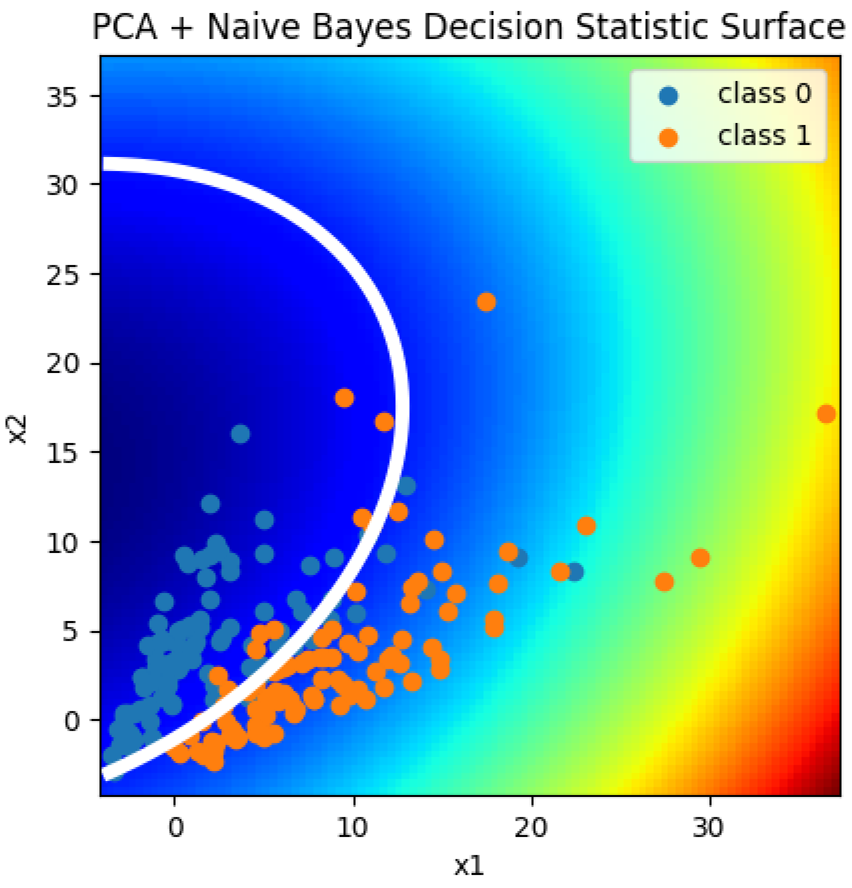
4)

4a)

```
In [ ]: data3_models = SVM_LDA_PCA_NB(data3)
        data3_models.svm()
        data3_models.lda()
        data3_models.pca_nb()
```

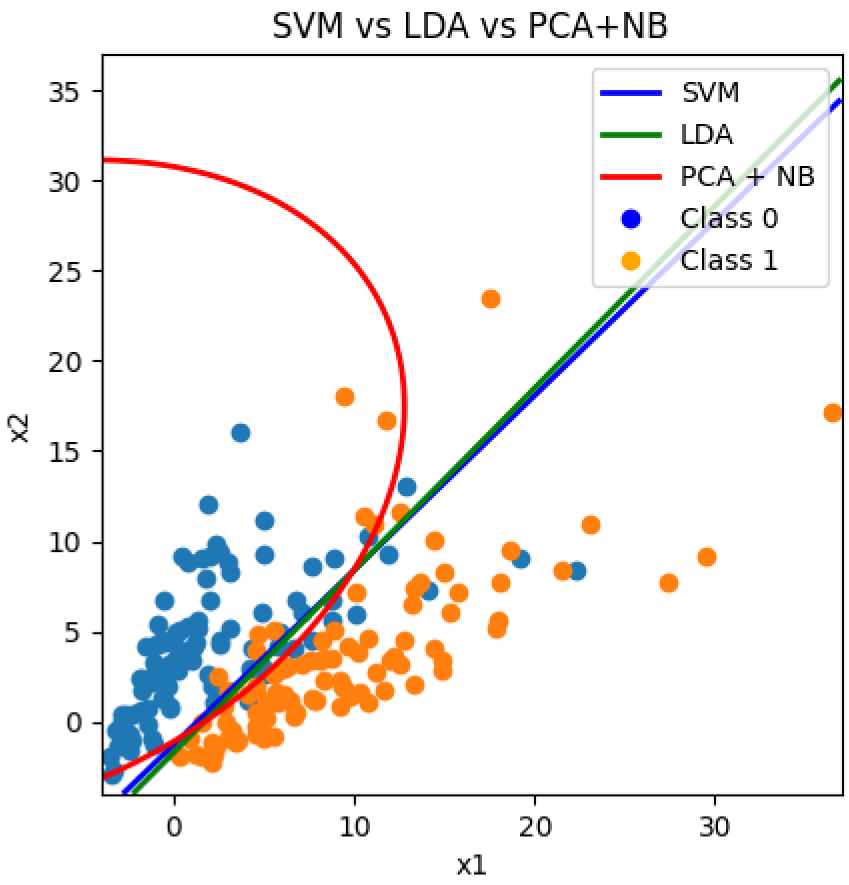



LDA Decision Statistic Surface
class 0
class 1
x1
x2
0
10
20
30
-5
0
5
10
15
20
25
30
35


PCA + Naive Bayes Decision Statistic Surface
class 0
class 1
x1
x2
0
10
20
30
0
5
10
15
20
25
30
35

4b)

```
In [ ]: data3_models.three_decision_boundaries()
```



4c)

The SVM and LDA decision boundaries were similar to my sketch. The boundary produced by PCA + NB was wildly different as it produced a curve around the blue (class 0) data points.

The SVM classifier resembles my sketch because the model assumes that my data is linearly separable. This means that the SVM classifier is looking for some linear vector that can separate the two classes. Since my sketch is also a linear boundary between the two blobs of data, the SVM classifier resembles my sketch.

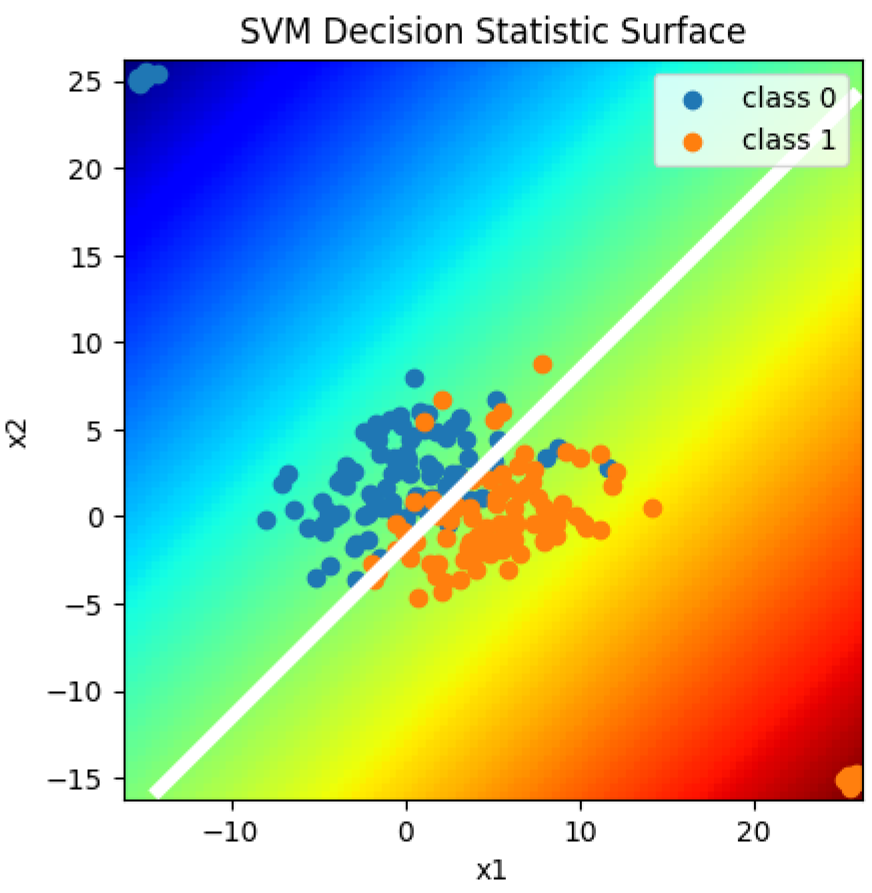
The LDA classifier resembles my sketch. Although, the model assumes that the data is normally distributed with identical covariances, his assumption about the data is not true as this data is log normal. However, since direction of the covariances is similiar, the LDA classifier does a fine job, as it is able to identify a vector that when the data is projected onto it, there is a lot of discrimination between the two classes.

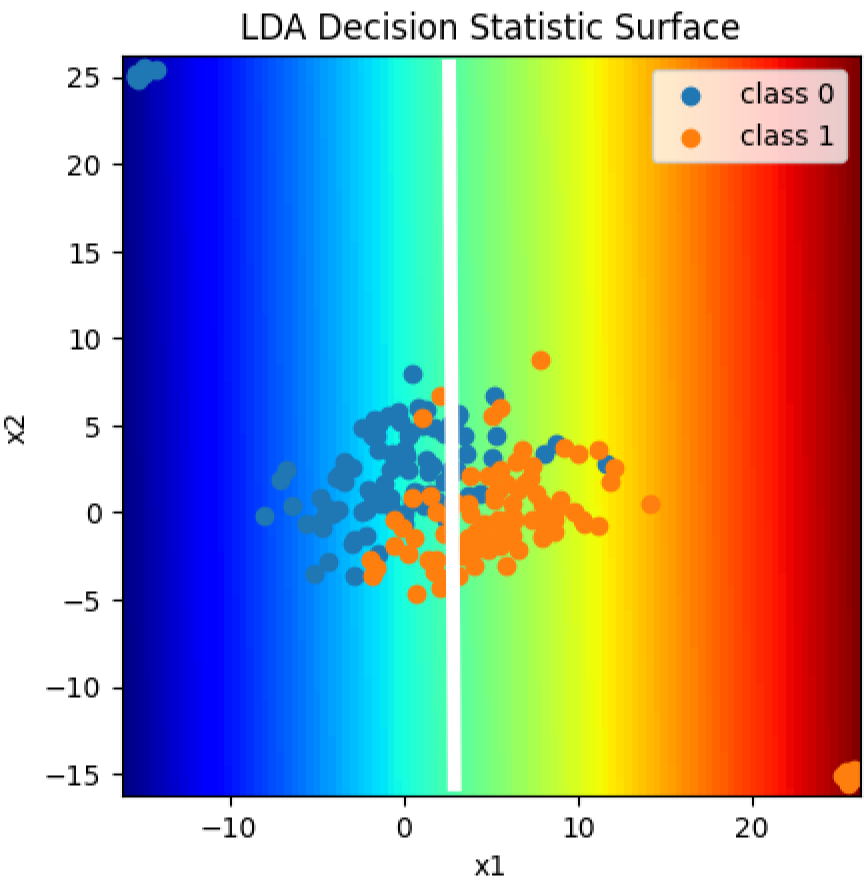
The PCA + NB classifier does not look like my sketch. NB classifiers assume that data is normally distributed, but I think the curve to the left is caused by the orange (class 1) data having more spread especially for the data points that are not in the main clump. Since NB classifiers work by identifying the probability of a class given the data, the increased variance of orange data makes it so that the NB classifier will be more likely to choose class 1 when considering points near the SVM decision boundary line.

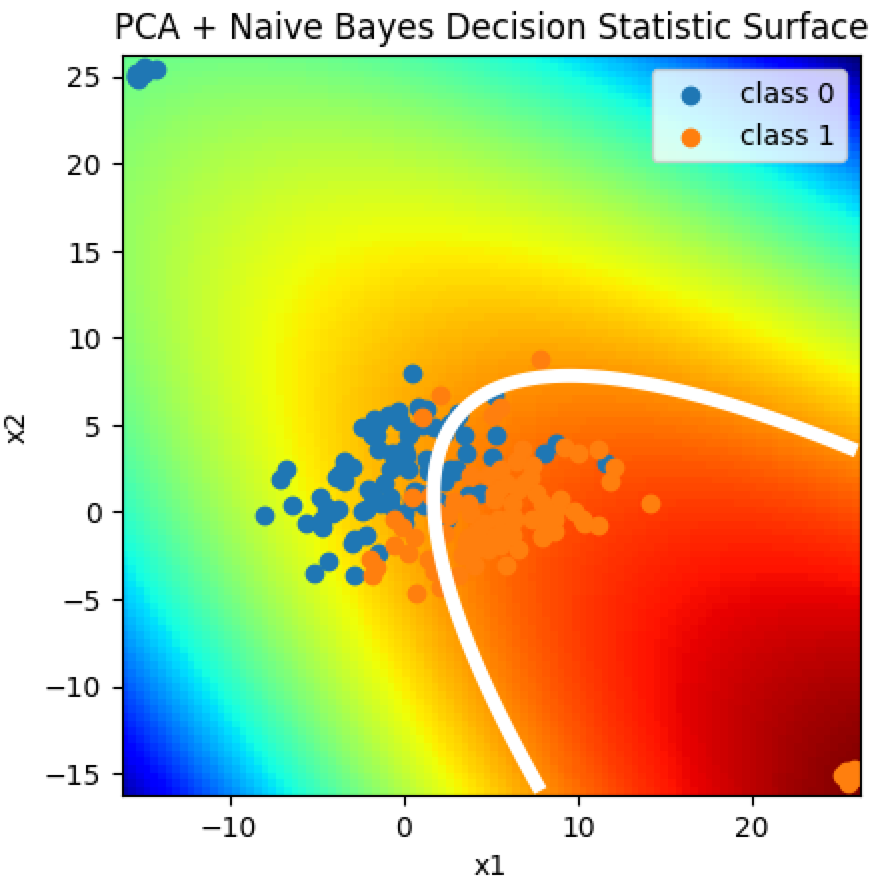
5)

5a)

```
In [ ]: data4_models = SVM_LDA_PCA_NB(data4)
        data4_models.svm()
        data4_models.lda()
        data4_models.pca_nb()
```

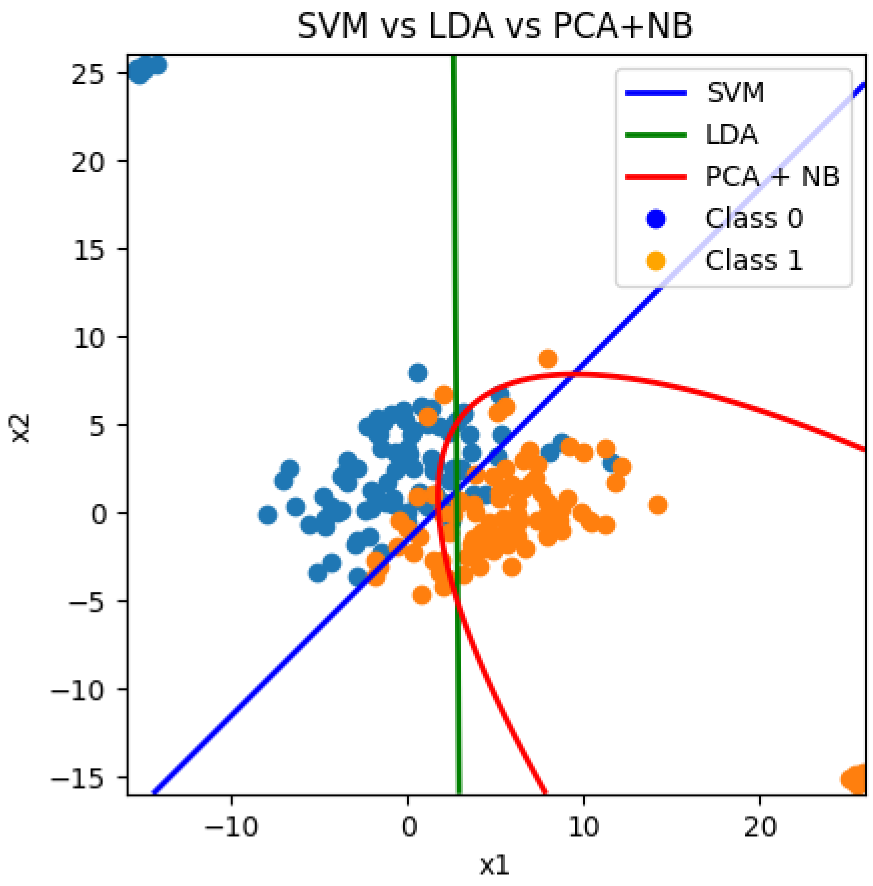



LDA Decision Statistic Surface
class 0
class 1
x2
x1
25
20
15
10
5
0
−5
−10
−15
−10
0
10
20


PCA + Naive Bayes Decision Statistic Surface
class 0
class 1
x2
x1
25
20
15
10
5
0
−5
−10
−15
−10
0
10
20

5b)

```
In [ ]: data4_models.three_decision_boundaries()
```



5c)

The SVM boundary was the only one similar to my sketch. The boundary produced by LDA was a vertical line down the middle of the main clump of data. The boundary produced by PCA + NB was wildly different as it produced a curve around the orange (class 1) data points.

The SVM classifier resembles my sketch because the model assumes that my data is linearly separable. This means that the SVM classifier is looking for some linear vector that can separate the two classes. Since my sketch is also a linear boundary between the two blobs of data, the SVM classifier resembles my sketch.

The LDA classifier does not resemble my sketch because the model assumes that the data is normally distributed with identical covariances. This assumption about the data is not true as this data has outliers making it not really gaussian. Dispite the data being separable linearly separable, the LDA classifier fails. I think this is because the outliers change changes the direction of the covariance from one diagonal to the other diagonal. This causes the direction of the LDA vector to be wrong, resulting in a vertical line.

The PCA + NB classifier does not look like my sketch. The curve towards the right happens because the data has outliers. I think the outliers make it so that the conditional probability of class 1 given the data gets skewed, causing a curve towards one of the outlier clumps.

6)

From the results of the four datasets, we can deduce the following:

Linear SVM classifiers are pretty robust to most types of data, but this is highly reliant on the fact that the data is linearly separable. If any of the 4 datasets were distributed in a way that wasn't linearly separable, this classifier will probably not be very effective.

LDA classifiers should only be used if there are no outliers. I think this is the biggest factor as seen by these four datasets. Although LDA assumes gaussian distributed data with identical covariance, with dataset 3, the data wasn't gaussian and LDA still worked. I think it's also important the the direction of the covariances of the two classes be in the direction of the boundary. As seen with dataset 4, when the covariance direction was nearly perpendicular to the SVM boundary, the LDA classifier did not perform well.

PCA + NB classifier should be used only if the data is gaussian distributed. This classifier produced an inaccurate model only for datasets 3 and 4, which were the only ones that didn't have gaussian distributed data.