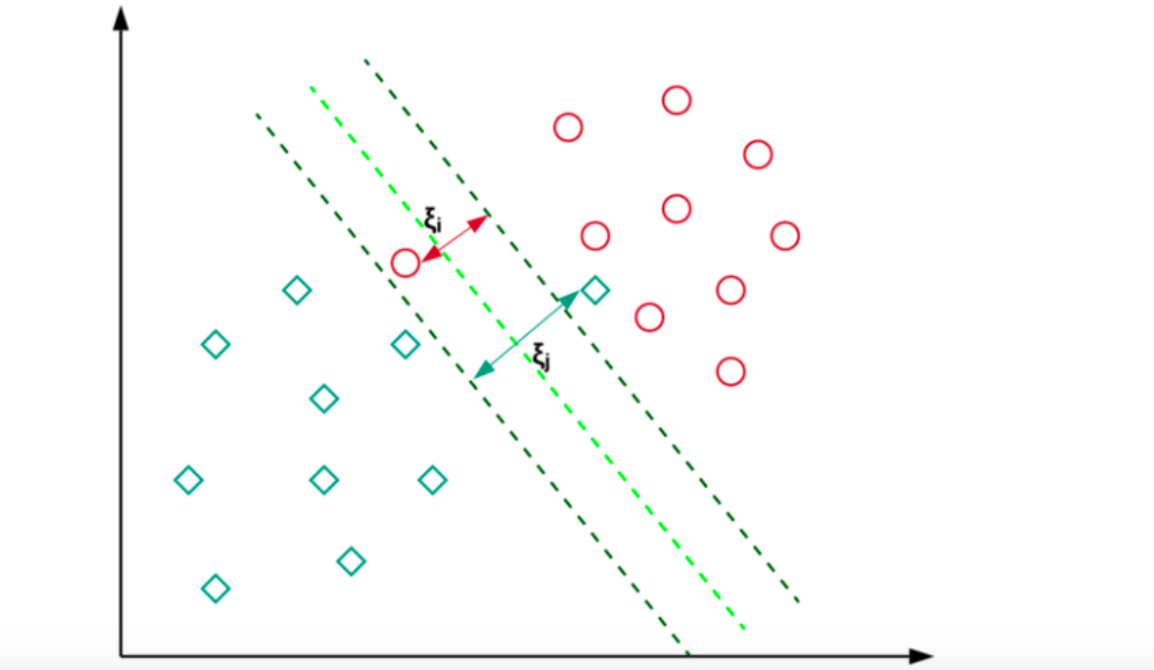
CIS-5930: Project Report
Ridhima Pahwa

1a and 1b)

Support Vector Machines(SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces

Support Vector Machines (SVMs) and related kernel methods have become increasingly popular tools for data mining tasks such as classification, regression, and novelty detection. The goal of this project is to provide an intuitive explanation of SVMs from a geometric perspective. We used Sequential Minimal Optimization Algorithm for implementation.

Basically, In supervised machine learning, we can create models that classifies each sample into one of two classes. For this programming assignment, its either Class A or Class B. Class A corresponds to +1 and Class B corresponds to -1. The models classifies each sample into one of the two classes and then assign one of two classes to a new sample, based on samples from the past that instruct it to do so.
We need to classify new samples according to a preexisting set of decision criteria. Class A is +1 and Class B is -1.
In the supervised machine learning we can mimic the decision-making ability by allowing to create a classifier. Using data from the past, it attempts to learn a decision boundary between the samples from the different classes. The end result will be, if a machine learning model is trained well, it can be used to decide automatically what class should be assigned once it is fed a new sample. As we have two classes, that is why it's a binary classifier.



So, a simple way to classify observations is to classify observations into two classes is to draw a linear boundary between them. However, even if the data is perfectly separable in this way, there are many possible linear boundaries that can be used. How to know which one is the best? One approach is to use some kind of statistical distribution fit. This means that even the points that are far from the boundary have an influence on where the boundary is located. Intuitively it seems like a better approach for this kind of the problem would be to put the boundary as far as possible from any of the observations.This is the basic idea behind support vector machine classification. If the data is actually linearly separable, this results in a simple optimization problem. Choose the coefficients of the linear boundary to maximize the margin, that is the distance between the boundary and the nearest observations. Subject to the constraint, that all observations must be on the correct side of the boundary. In the end, the optimal solution determined only by the observations nearest to the boundary. These observations are referred to as the Support Vectors. The real noisy data may not be linearly separable, that is there's no linear boundary that can correctly classify every observation. In this case, we can modify the optimization problem to maximize the margin but with the penalty term for the misclassified observations. The observations are correctly classified only if they lie on the correct side of the margin.So the penalty term prevents the solution by having a huge margin.
The SVM solution then is the one that gives the best possible separation between the classes that is the widest margin without unnecessary classifications.

Now, for different values of C(the trade-off parameter) between the misclassification error and the margin, decision boundary was plotted for different values of C.(Below are the Screenshots)

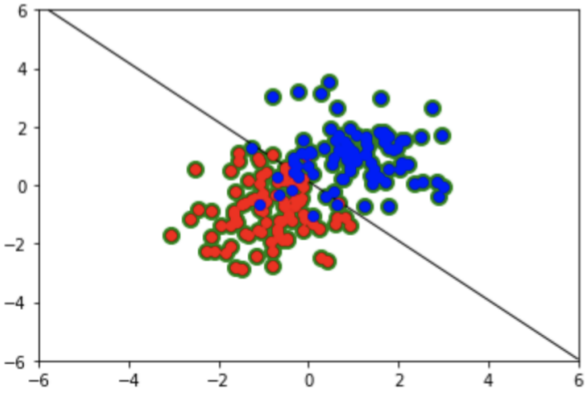
C=0.0001

```
def test_soft():
    X1, y1, X2, y2 = gen_lin_separable_overlap_data()
    X_train, y_train = split_train(X1, y1, X2, y2)
    X_test, y_test = split_test(X1, y1, X2, y2)

    clf = SVM(C=0.0001)
    clf.fit(X_train, y_train)
    y_predict = clf.predict(X_test)
    correct = np.sum(y_predict == y_test)
    print('{0} out of {1} predictions correct'.format(correct, len(y_predict)))
    plot_contour(X_train[y_train==1], X_train[y_train==-1], clf)


test_soft()
```

```
     pcost       dcost       gap    pres   dres
0: -3.0539e+01 -1.6212e-01  1e+03  3e+01  3e-15
1: -4.8562e-01 -1.6003e-01  1e+01  4e-01  3e-15
2: -1.4032e-01 -8.1248e-02  1e+00  4e-02  6e-16
3: -1.2088e-02 -4.4047e-02  4e-02  3e-04  1e-15
4: -1.3803e-02 -2.2753e-02  1e-02  7e-05  5e-16
5: -1.6960e-02 -1.8692e-02  2e-03  7e-06  4e-16
6: -1.7709e-02 -1.7730e-02  2e-05  8e-08  3e-16
7: -1.7719e-02 -1.7719e-02  2e-07  8e-10  3e-16
8: -1.7719e-02 -1.7719e-02  2e-09  8e-12  3e-16
ptimal solution found.
80 support vectors out of 180 points
03 out of 220 predictions correct
usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:73: UserWarning: No contour levels were found
usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:74: UserWarning: No contour levels were found
```

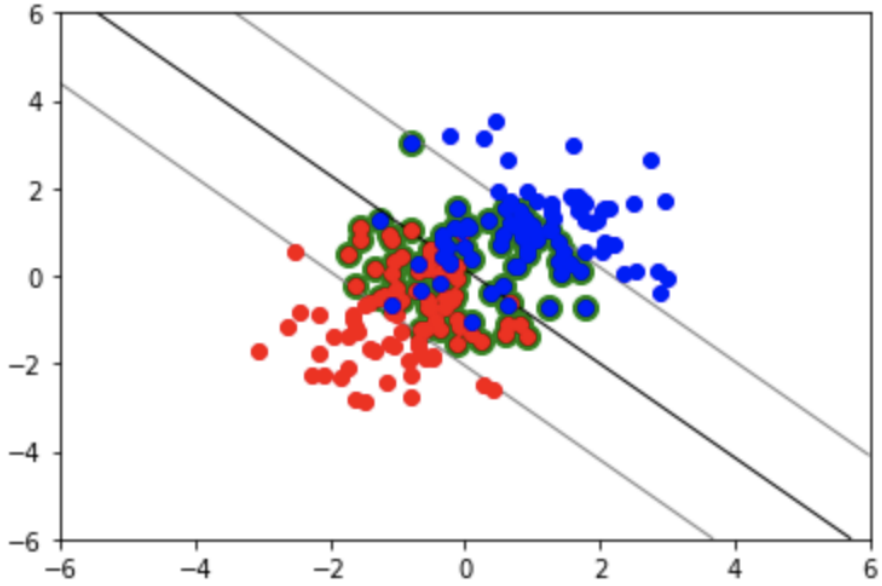


C=0.01

```
    clf = SVM(C=0.01)
    clf.fit(X_train, y_train)
    y_predict = clf.predict(X_test)
    correct = np.sum(y_predict == y_test)
    print('{0} out of {1} predictions correct'.format(correct, len(y_predict)))
    plot_contour(X_train[y_train==1], X_train[y_train==-1], clf)


test_soft()
```

```
     pcost       dcost       gap    pres   dres
 0: -3.0840e+01 -3.7398e+00  1e+03  3e+01  3e-15
 1: -9.3076e-01 -3.7189e+00  2e+01  4e-01  3e-15
 2: -5.7571e-01 -2.5740e+00  2e+00  7e-04  4e-16
 3: -6.7536e-01 -9.1919e-01  2e-01  9e-05  4e-16
 4: -7.7609e-01 -8.1573e-01  4e-02  1e-05  3e-16
 5: -7.9235e-01 -8.0038e-01  8e-03  1e-06  3e-16
 6: -7.9587e-01 -7.9689e-01  1e-03  1e-07  3e-16
 7: -7.9636e-01 -7.9639e-01  3e-05  2e-09  3e-16
 8: -7.9638e-01 -7.9638e-01  1e-06  6e-11  3e-16
 9: -7.9638e-01 -7.9638e-01  4e-08  2e-12  3e-16
Optimal solution found.
102 support vectors out of 180 points
203 out of 220 predictions correct
```



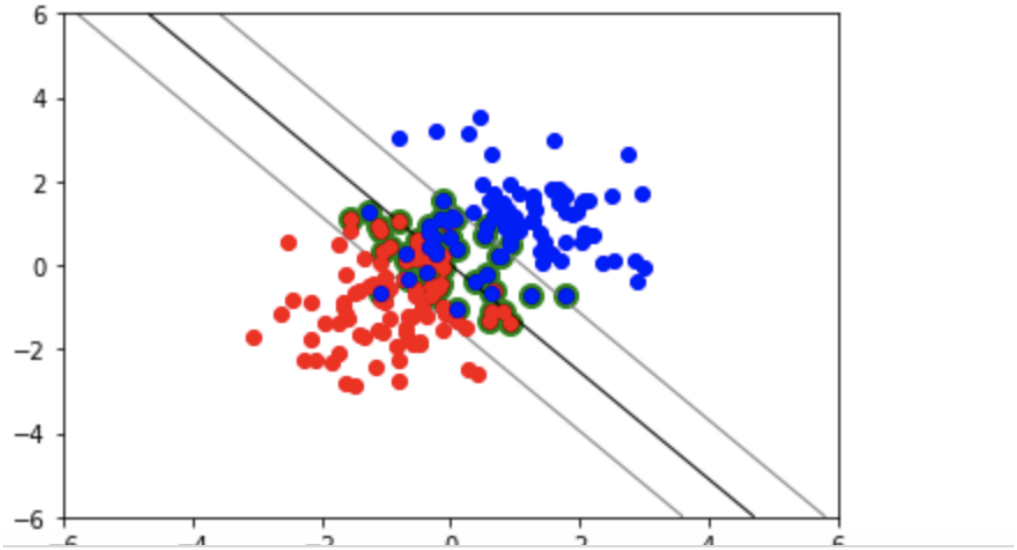
C=0.1

```
def test_soft():
    X1, y1, X2, y2 = gen_lin_separable_overlap_data()
    X_train, y_train = split_train(X1, y1, X2, y2)
    X_test, y_test = split_test(X1, y1, X2, y2)

    clf = SVM(C=0.1)
    clf.fit(X_train, y_train)
    y_predict = clf.predict(X_test)
    correct = np.sum(y_predict == y_test)
    print('{0} out of {1} predictions correct'.format(correct, len(y_predict)))
    plot_contour(X_train[y_train==1], X_train[y_train==-1], clf)


test_soft()
```

```
     pcost       dcost       gap    pres   dres
 0: -3.3576e+01 -3.6275e+01  1e+03  3e+01  4e-15
 1: -5.8770e+00 -3.3739e+01  8e+01  1e+00  3e-15
 2: -3.8808e+00 -1.5305e+01  1e+01  1e-01  1e-15
 3: -4.1341e+00 -5.8543e+00  2e+00  1e-02  2e-15
 4: -4.5714e+00 -5.0855e+00  6e-01  3e-03  7e-16
 5: -4.7135e+00 -4.8525e+00  1e-01  4e-04  6e-16
 6: -4.7645e+00 -4.7732e+00  9e-03  6e-06  6e-16
 7: -4.7681e+00 -4.7703e+00  2e-03  9e-07  5e-16
 8: -4.7691e+00 -4.7693e+00  3e-04  3e-08  5e-16
 9: -4.7692e+00 -4.7692e+00  6e-06  7e-10  6e-16
10: -4.7692e+00 -4.7692e+00  8e-08  8e-12  5e-16
Optimal solution found.
57 support vectors out of 180 points
202 out of 220 predictions correct
```



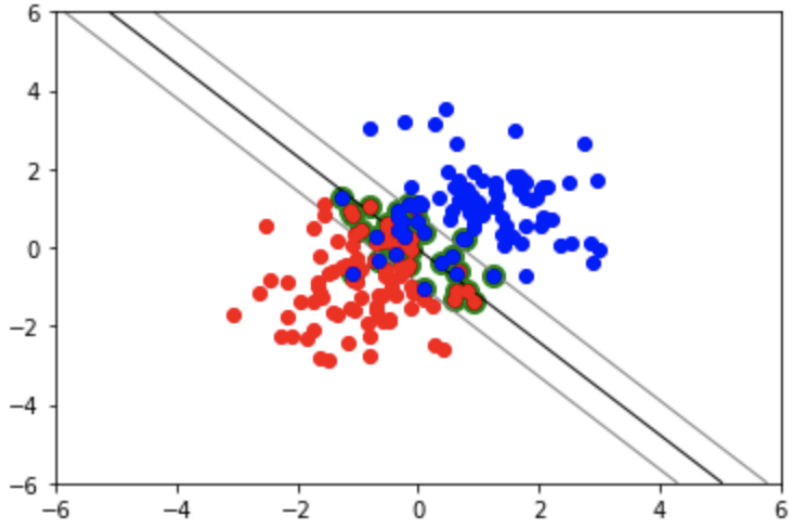
C=0.5

```
def test_soft():
    X1, y1, X2, y2 = gen_lin_separable_overlap_data()
    X_train, y_train = split_train(X1, y1, X2, y2)
    X_test, y_test = split_test(X1, y1, X2, y2)

    clf = SVM(C=0.5)
    clf.fit(X_train, y_train)
    y_predict = clf.predict(X_test)
    correct = np.sum(y_predict == y_test)
    print('{0} out of {1} predictions correct'.format(correct, len(y_predict)))
    plot_contour(X_train[y_train==1], X_train[y_train==-1], clf)


test_soft()
```

```
     pcost       dcost       gap    pres   dres
 0: -4.5709e+01 -1.8112e+02  1e+03  5e+00  4e-15
 1: -2.3722e+01 -1.3357e+02  2e+02  6e-01  3e-15
 2: -1.7643e+01 -4.4278e+01  4e+01  6e-02  3e-15
 3: -1.8289e+01 -2.2797e+01  6e+00  9e-03  3e-15
 4: -1.9073e+01 -2.0531e+01  2e+00  2e-03  1e-15
 5: -1.9378e+01 -1.9885e+01  6e-01  6e-04  1e-15
 6: -1.9544e+01 -1.9620e+01  8e-02  6e-05  1e-15
 7: -1.9572e+01 -1.9580e+01  8e-03  5e-06  1e-15
 8: -1.9575e+01 -1.9576e+01  2e-04  8e-08  1e-15
 9: -1.9575e+01 -1.9575e+01  2e-06  1e-09  1e-15
Optimal solution found.
44 support vectors out of 180 points
203 out of 220 predictions correct
```



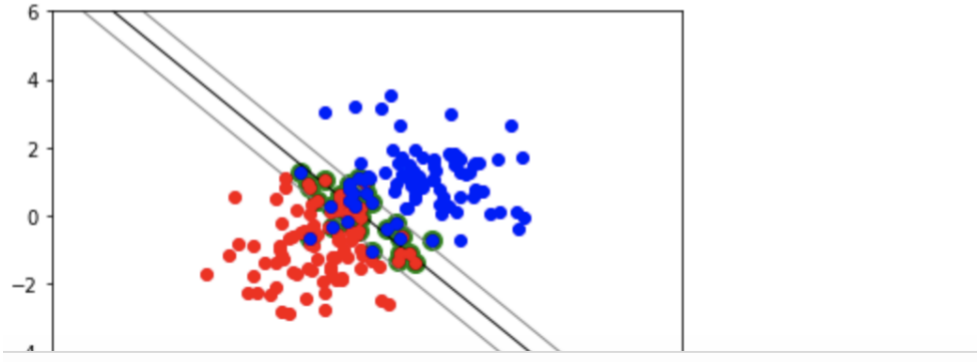
C=100

```
def test_soft():
    X1, y1, X2, y2 = gen_lin_separable_overlap_data()
    X_train, y_train = split_train(X1, y1, X2, y2)
    X_test, y_test = split_test(X1, y1, X2, y2)

    clf = SVM(C=100)
    clf.fit(X_train, y_train)
    y_predict = clf.predict(X_test)
    correct = np.sum(y_predict == y_test)
    print('{0} out of {1} predictions correct'.format(correct, len(y_predict)))
    plot_contour(X_train[y_train==1], X_train[y_train==-1], clf)


test_soft()
```

```
     pcost       dcost       gap    pres   dres
 0: -1.8117e+03 -1.1136e+06  3e+06  6e-01  3e-13
 1:  1.0123e+03 -2.6245e+05  3e+05  4e-02  3e-13
 2: -8.8125e+02 -4.4445e+04  5e+04  4e-03  2e-13
 3: -2.0979e+03 -9.0206e+03  7e+03  6e-04  1e-13
 4: -2.5504e+03 -6.5633e+03  4e+03  3e-04  1e-13
 5: -2.7897e+03 -5.6860e+03  3e+03  2e-04  1e-13
 6: -2.9606e+03 -5.1851e+03  2e+03  1e-04  1e-13
 7: -3.0440e+03 -4.6024e+03  2e+03  4e-05  1e-13
 8: -3.1643e+03 -4.2960e+03  1e+03  2e-05  1e-13
 9: -3.3673e+03 -3.8303e+03  5e+02  2e-06  1e-13
10: -3.4878e+03 -3.6462e+03  2e+02  5e-07  1e-13
11: -3.5241e+03 -3.5933e+03  7e+01  2e-07  1e-13
12: -3.5315e+03 -3.5765e+03  4e+01  8e-08  2e-13
13: -3.5361e+03 -3.5669e+03  3e+01  3e-08  1e-13
14: -3.5382e+03 -3.5648e+03  3e+01  2e-08  1e-13
15: -3.5474e+03 -3.5542e+03  7e+00  3e-09  2e-13
16: -3.5506e+03 -3.5507e+03  9e-02  4e-11  1e-13
17: -3.5506e+03 -3.5506e+03  9e-04  7e-13  1e-13
Optimal solution found.
41 support vectors out of 180 points
202 out of 220 predictions correct
```



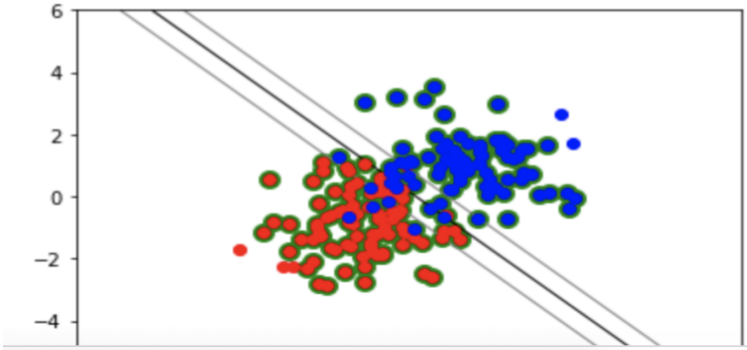
C=1000

```
def test_soft():
    X1, y1, X2, y2 = gen_lin_separable_overlap_data()
    X_train, y_train = split_train(X1, y1, X2, y2)
    X_test, y_test = split_test(X1, y1, X2, y2)

    clf = SVM(C=1000)
    clf.fit(X_train, y_train)
    y_predict = clf.predict(X_test)
    correct = np.sum(y_predict == y_test)
    print('{0} out of {1} predictions correct'.format(correct, len(y_predict)))
    plot_contour(X_train[y_train==1], X_train[y_train==-1], clf)


test_soft()
```

```
 2:  8.6470e+04 -3.9573e+06  4e+06  4e-03  3e-12
 3: -1.8812e+04 -2.1174e+05  2e+05  7e-05  1e-12
 4: -2.3194e+04 -1.0250e+05  8e+04  3e-05  1e-12
 5: -2.5047e+04 -7.8785e+04  5e+04  2e-05  1e-12
 6: -2.7749e+04 -5.0401e+04  2e+04  6e-06  1e-12
 7: -2.8975e+04 -4.8686e+04  2e+04  4e-06  1e-12
 8: -2.9801e+04 -4.4974e+04  2e+04  2e-06  1e-12
 9: -3.1969e+04 -4.1419e+04  9e+03  8e-07  1e-12
10: -3.3032e+04 -3.9103e+04  6e+03  3e-07  2e-12
11: -3.4659e+04 -3.6984e+04  2e+03  3e-08  2e-12
12: -3.5230e+04 -3.5991e+04  8e+02  1e-08  1e-12
13: -3.5230e+04 -3.5886e+04  7e+02  7e-09  1e-12
14: -3.5405e+04 -3.5604e+04  2e+02  1e-09  1e-12
15: -3.5408e+04 -3.5577e+04  2e+02  6e-10  1e-12
16: -3.5412e+04 -3.5573e+04  2e+02  5e-10  1e-12
17: -3.5468e+04 -3.5505e+04  4e+01  1e-11  1e-12
18: -3.5483e+04 -3.5485e+04  3e+00  4e-12  1e-12
19: -3.5484e+04 -3.5484e+04  3e-02  3e-12  1e-12
Optimal solution found.
175 support vectors out of 180 points
201 out of 220 predictions correct
```



2.Linear Regression Implementation

The source code file is named Linear_regression2.

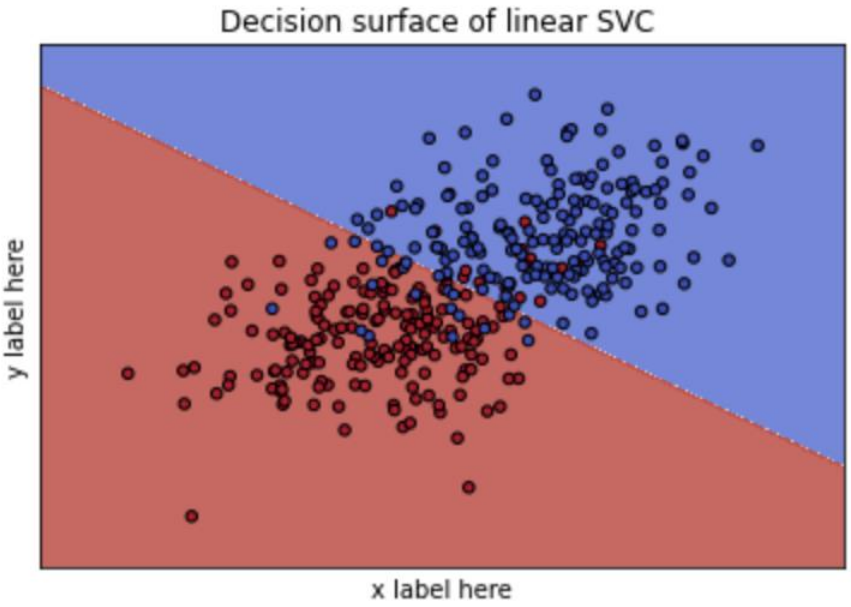
In this question of the programming project, we were required to Implement a Linear Regression.and also create a classifier by thresholding the output at 0.

The data from question 1b. needs to be used in this question to implement the Linear regression. Running the implementation on the random data generated in the following way:

For Class A which corresponds to +1, generate 200 points, where according to a two-dimensional gaussian with mean (-1,-1) and covariance matrix of (1,0.25) and (0.25,1)

For Class B, That corresponds to -1, generate 200 points using same distribution. The mean in this case will be (1,1) .Seed the random number generator with the last five digits of FSU library card number: 47405

The linear regression was successfully implemented and a decision boundary of the classifier was plotted along with the training points.



To evaluate the performance of the model on a dataset, we need to measure how well the predictions made by the model match the observed data. One

commonly used method for doing this is known as Leave-one-out-Crossvalidation(LOOCV).