Task A

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, Normalizer
from sklearn.svm import SVC
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
def draw_line(coef,intercept, mi, ma):
    # for the separating hyper plane ax+by+c=0, the weights are [a, b] and the intercept is c
    # to draw the hyper plane we are creating two points
    # 1. ((b*min-c)/a, min) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here in place of y we a
re keeping the minimum value of y
    # 2. ((b*max-c)/a, max) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here in place of y we a
re keeping the maximum value of y

    points=np.array([[((-coef[1]*mi - intercept)/coef[0]), mi],[((-coef[1]*ma - intercept)/coef[0]),
ma]])

    plt.plot(points[:,0], points[:,1], label='Decision Boundary')
```

In [4]:
```
def draw_support_vectors(support_vectors):

    #Support Vectors
    p_support_vectors = support_vectors[:, 0]
    n_support_vectors = support_vectors[:, 1]

    #https://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html
    plt.scatter(p_support_vectors, n_support_vectors, s=100, linewidth=1, facecolors='none', edgecol
ors='k')
```

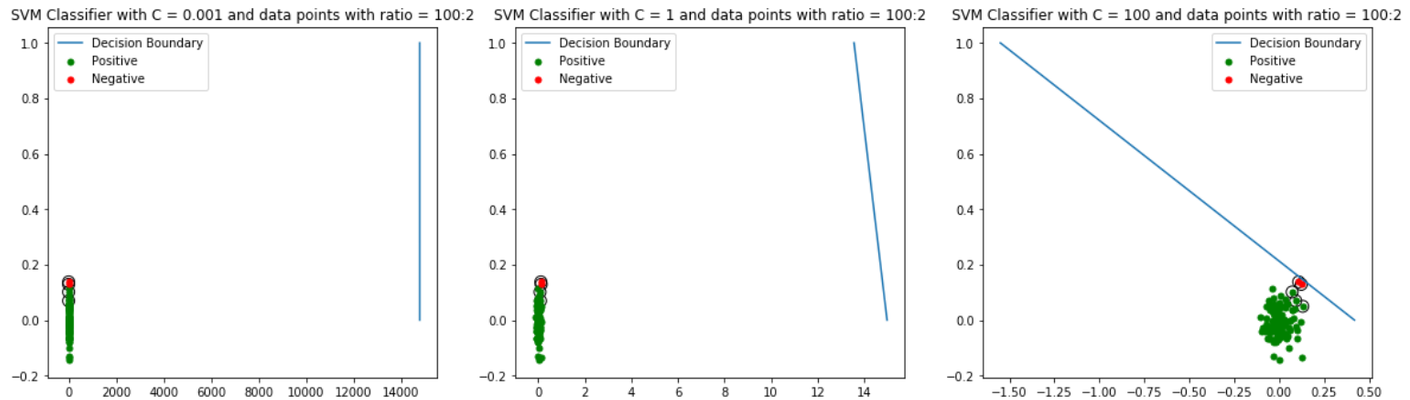
In [13]:
```
def classify(classifier_name):

    ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
    C_Range = [0.001, 1, 100]
```

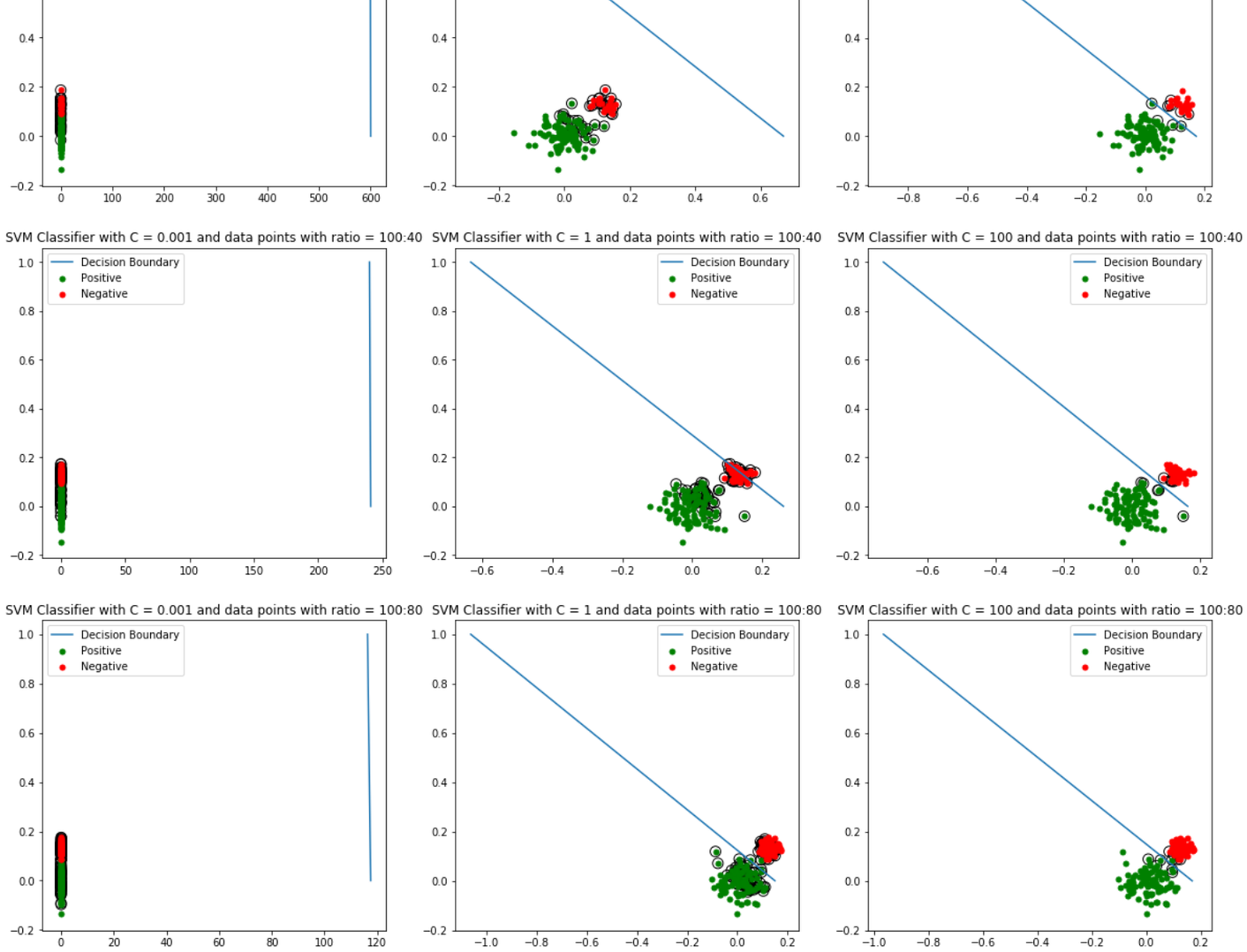
```
plot_ctr = 0
plt.figure(figsize=(20,25))

for j,i in enumerate(ratios):

    X_p=np.random.normal(0,0.05,size=(i[0],2))
    X_n=np.random.normal(0.13,0.02,size=(i[1],2))

    y_p=np.array([1]*i[0]).reshape(-1,1)
    y_n=np.array([0]*i[1]).reshape(-1,1)

    X=np.vstack((X_p,X_n))
    y=np.vstack((y_p,y_n))

    #Finding min and max y
    min_y = np.min(y)
    max_y = np.max(y)


    #Foreach loop of C range
    for a, c in enumerate(C_Range):

        plot_ctr  += 1

        #Initializing the classifier
        if classifier_name == 'SVM':
            clf = SVC(C= c, kernel='linear')
        else:
            clf = LogisticRegression(C= c)


        #Fitting data to prepare model
        clf.fit(X, y)

        #Weight and Bias
        weight = clf.coef_[0]
        bias = clf.intercept_

        #Preparing Subplots for plotting each graph
        plt.subplot(len(ratios), len(C_Range), plot_ctr)


        #Drawing the hyperplane/Line
        draw_line(weight, bias, min_y, max_y)

        #Plotting support vectors
        if classifier_name == 'SVM':
```

```
                draw_support_vectors(clf.support_vectors_)

            #plt.axis([-0.2, 2, -0.2, 2])

            #Plotting dataset
            plt.scatter(X_p[:,0],X_p[:,1], label='Positive', s=25, color='green')
            plt.scatter(X_n[:,0],X_n[:,1], label='Negative', s=25, color='red')

            plt.legend()

            if classifier_name == 'SVM':
                plt.title('SVM Classifier with C = ' + str(c) + ' and data points with ratio = ' + s
tr(i[0]) + ':' + str(i[1]))
            else:
                plt.title('LR with C = ' + str(c) + ' and data points with ratio = ' + str(i[0]) +
':' + str(i[1]))




    plt.show()
```

Task A.1

In [8]:
```
classify('SVM')
```





Observation:

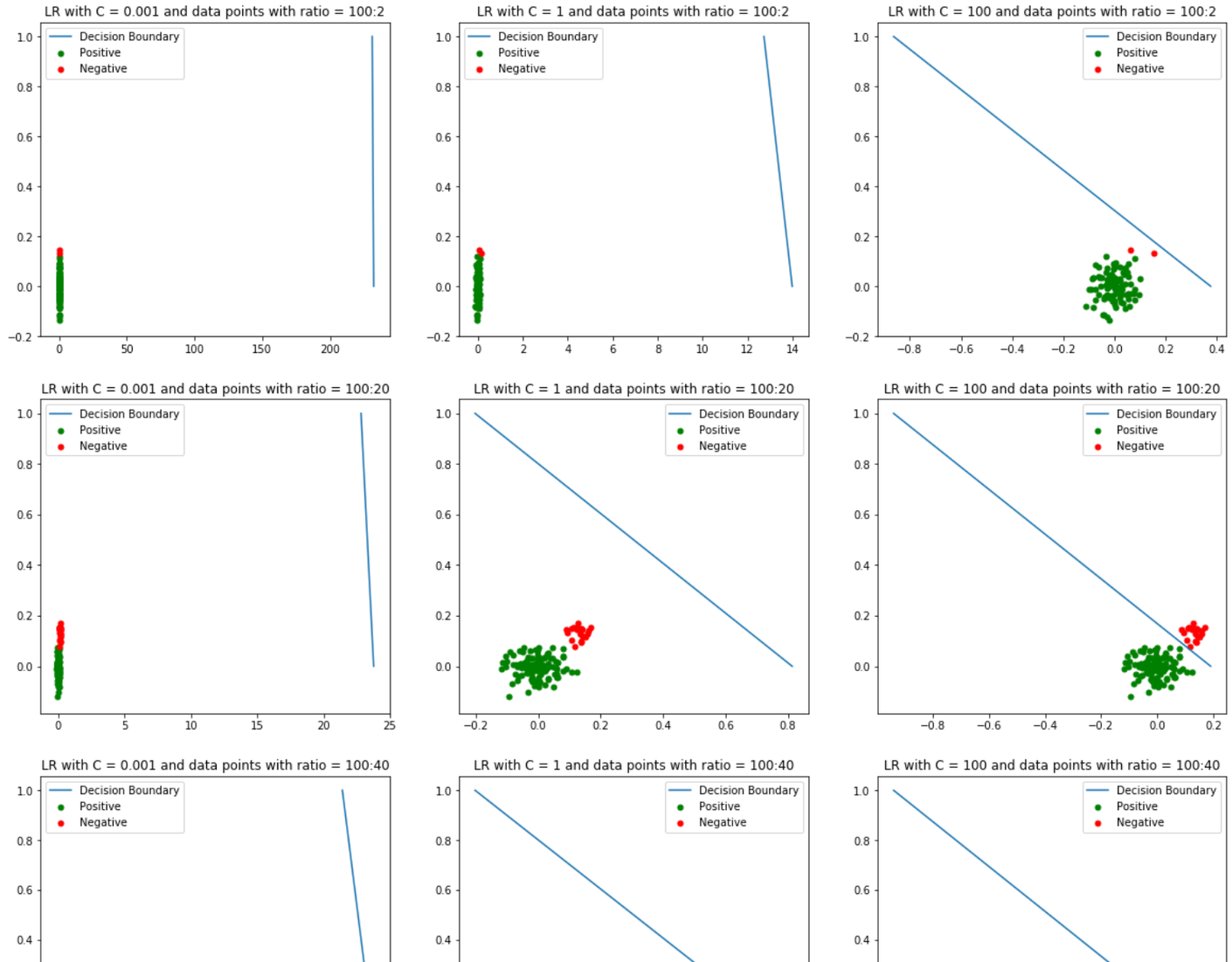
1. As we can see in the plots above that as the learning rate is increase the plane/line is coming near to the datapoints(check the plots from left to right).
2. One more thing we should notice that as the number of negative points increases(data is getting balanced) the plane/line is getting more support vectors which are helping to plane/line to classify positive and negative points clearly.
3. Data with 100:2 ratio(imbalance data) is not properly classified, and as mentioned above w

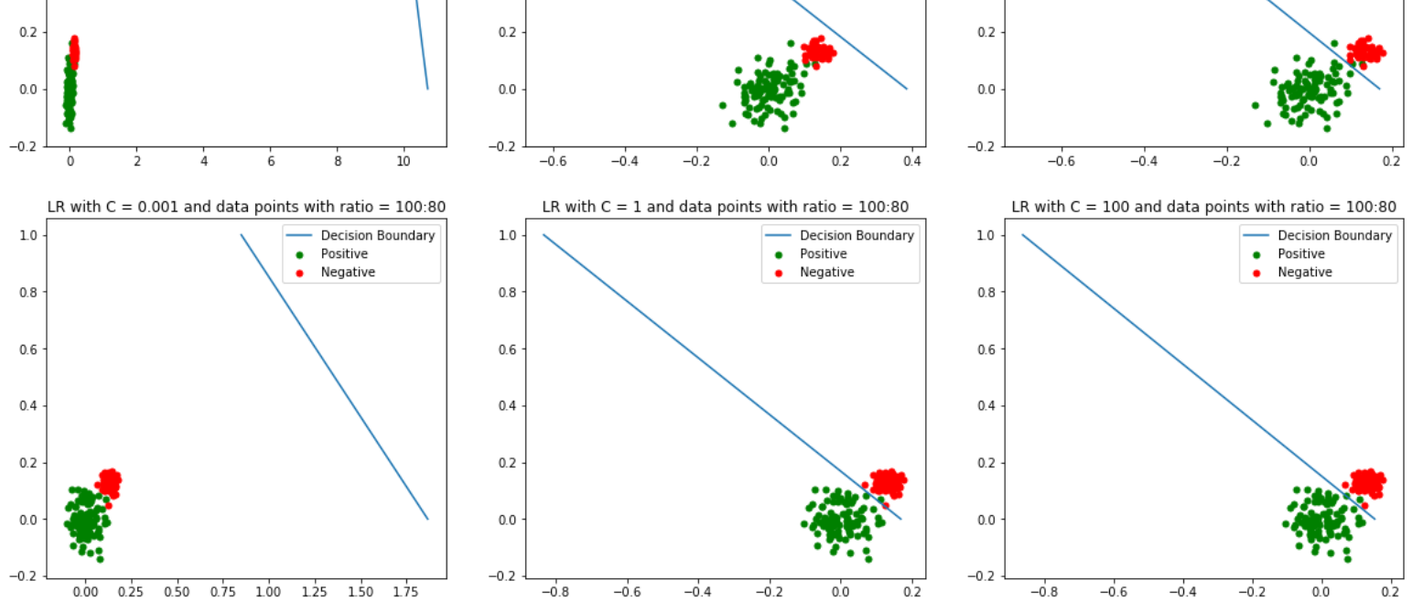
ith balanced data set we get the best plane/line which can be seen in (C=100, ratio=100:20), (C=100, ratio=100:40) and (C=100, ratio=100:80)

Conclusion: We can say that C=100 is the best hyperparameter in case of SVM

Task A.2

In [14]:
```
classify('LR')
```





Observation:

1. Just like the case of SVM here also increasing learning rate is helping to get the best plane/line
(check the plots from left to right).
2. Number of increasing negative datapoints balancing th data which is improving the model
(check the plots from top to bottom).
3. When we see the plot with ratio of 100:80, we can see the C=1 and C=100 both have best plane/line which classifies positve and neative points with 1 or 2 outliers, but still I would say that C=100 is managing to be the best in most of the ratios(100:20, 100:40, 100:80).

Conclusion: We can say that C=100 is the best hyperparameter in case of LR as well.