Task A: Effect of data imbalance on linear classifiers

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

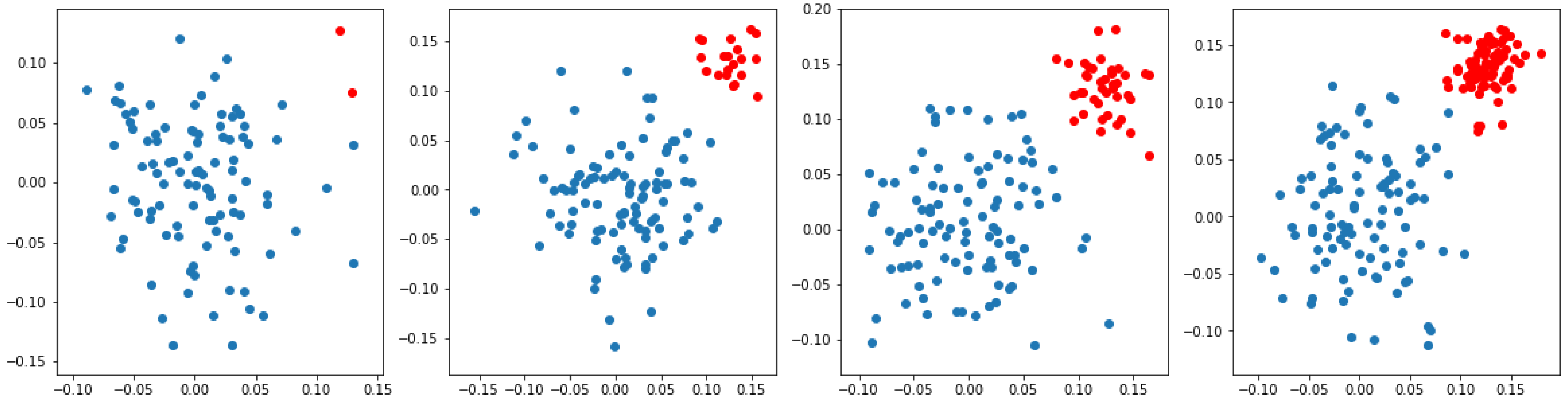
In [19]:
```
def draw_line(coef,intercept, mi, ma):
    # for the separating hyper plane ax+by+c=0, the weights are [a, b] and the intercept is c
    # to draw the hyper plane we are creating two points
    # 1. ((b*min-c)/a, min) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here in place of y we are keeping the mi
    # 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 are keeping the ma
    points=np.array([[((-coef[1]*mi - intercept)/coef[0]), mi],[((-coef[1]*ma - intercept)/coef[0]), ma]])
    plt.plot(points[:,0], points[:,1])
```

What if Data is imbalanced

```
1. As a part of this task you will observe how linear models work in case of data imbalanced
2. observe how hyper plane is changs according to change in your learning rate.
3. below we have created 4 random datasets which are linearly separable and having class imbalance
4. in the first dataset the ratio between positive and negative is 100 : 2, in the 2nd data its
100:20,
in the 3rd data its 100:40 and in 4th one its 100:80
```

In [0]:
```
# here we are creating 2d imbalanced data points for visualization:
```

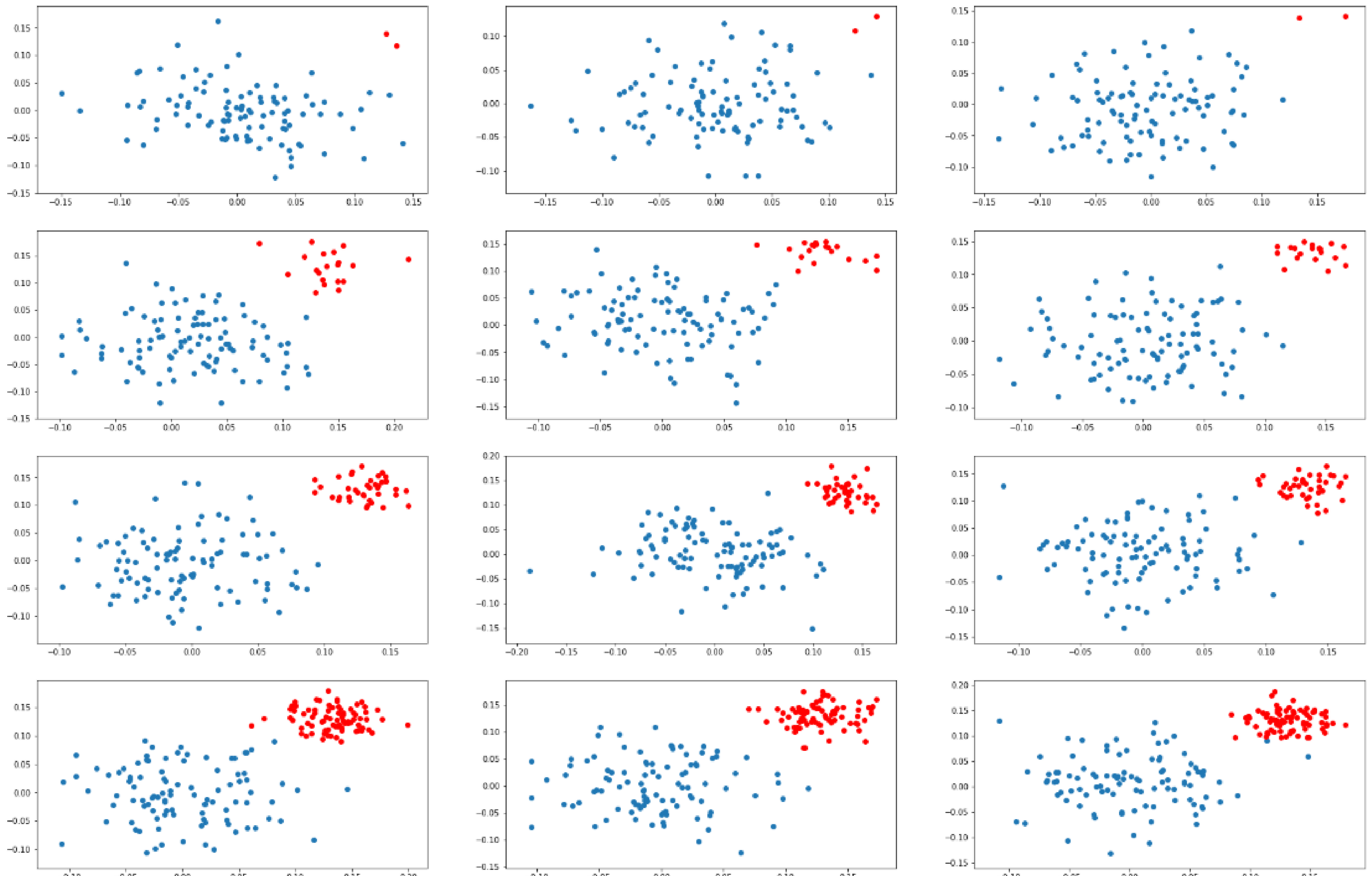
```
ratios = [(100,2), (100, 20), (100, 40), (100, 80)] # ratio of positive & negative points
plt.figure(figsize=(20,5))
for j,i in enumerate(ratios):
    plt.subplot(1, 4, j+1)  # create subplots
    X_p=np.random.normal(0,0.05,size=(i[0],2))
    X_n=np.random.normal(0.13,0.02,size=(i[1],2))
    y_p=np.array([1]*i[0]).reshape(-1,1)
    y_n=np.array([0]*i[1]).reshape(-1,1)
    X=np.vstack((X_p,X_n))
    y=np.vstack((y_p,y_n))
    plt.scatter(X_p[:,0],X_p[:,1])
    plt.scatter(X_n[:,0],X_n[:,1],color='red')

plt.show()
```



> your task is to apply SVM (sklearn.svm.SVC) and LR (sklearn.linear_model.LogisticRegression) with different regularization strength [0.001, 1, 100]

Task 1: Applying SVM

```
1. you need to create a grid of plots like this
```



in each of the cell[i][j] you will be drawing the hyper plane that you get after applying SVM on ith dataset and
        jth learnig rate

i.e

Plane(SVM().fit(D1, C=0.001))	Plane(SVM().fit(D1, C=1))	Plane(SVM().fit(D1, C=100))
Plane(SVM().fit(D2, C=0.001))	Plane(SVM().fit(D2, C=1))	Plane(SVM().fit(D2, C=100))
Plane(SVM().fit(D3, C=0.001))	Plane(SVM().fit(D3, C=1))	Plane(SVM().fit(D3, C=100))
Plane(SVM().fit(D4, C=0.001))	Plane(SVM().fit(D4, C=1))	Plane(SVM().fit(D4, C=100))

if you can do, you can represent the support vectors in different colors,
which will help us understand the position of hyper plane

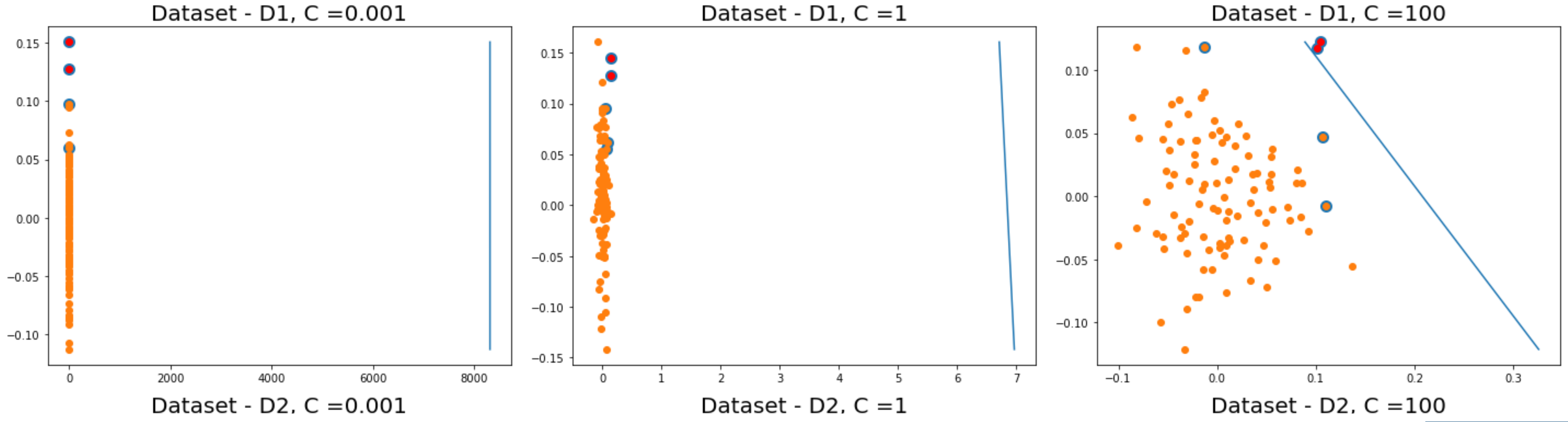
Write in your own words, the observations from the above plots, and what do you think about the position of the hyper plane

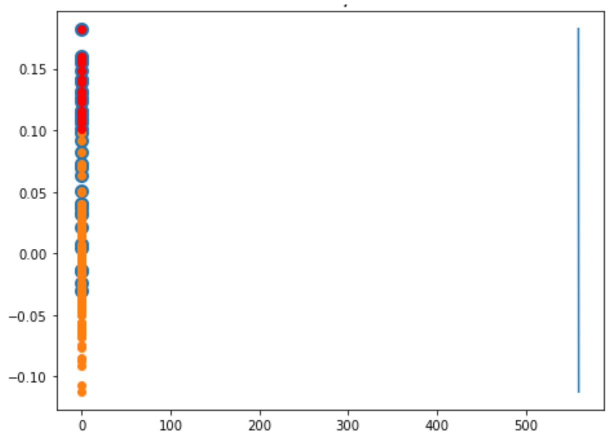
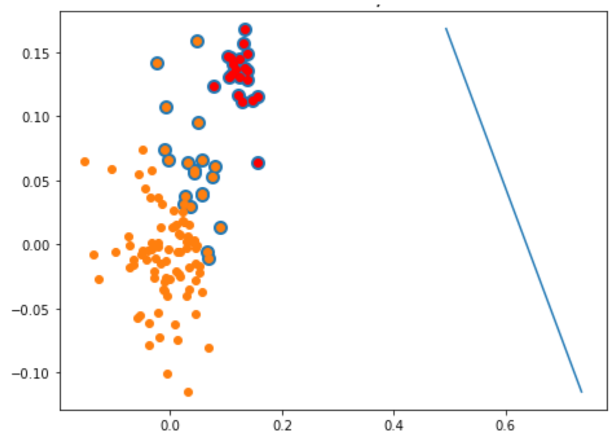
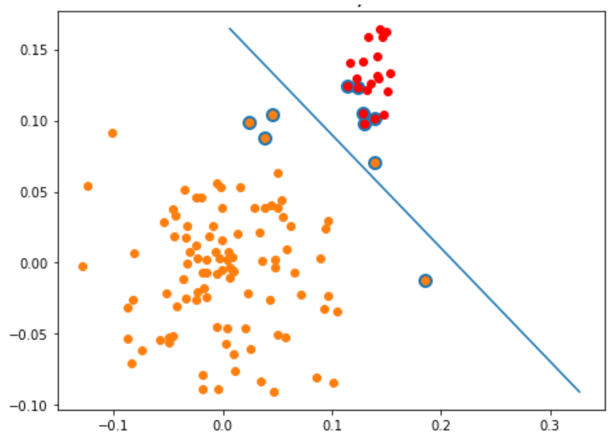
check the optimization problem here https://scikit-learn.org/stable/modules/svm.html#mathematical-formulation

if you can describe your understanding by writing it on a paper
and attach the picture, or record a video upload it in assignment.

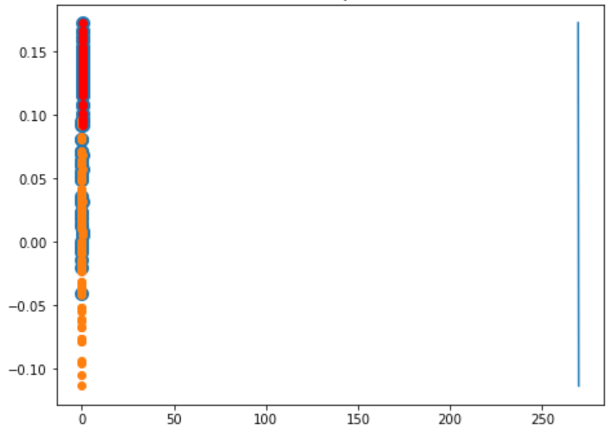
Kindly Refer below cells:

In [47]:
```
#lets create datasets D1,D2,D3 and D4:
ratio = [(100,2),(100,20),(100,40),(100,80)]

index = 0;
plt.figure(figsize=(20,20))   # figure size

for j,i in enumerate(ratio):
    reg_param = [0.001,1,100]    # diff hyperparameters
    for c in range(0,len(ratio)-1):
        plt.subplot(4,3,index+1)     # subplots creation
        index += 1
        # extracting positive & negative data points according to the ratios initialized
        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)) # obtain input data points
        y=np.vstack((y_p,y_n)) # obtain target varible

        clf = SVC(C = reg_param[c],kernel = 'linear').fit(X,y) # fit linear SVM

        #lets plot the seperating hyperplane along with support vectors:
        plt.title("Dataset - D{0}, C ={1}".format(j+1,reg_param[c]),fontsize = 20)
        plt.scatter(clf.support_vectors_[:,0],clf.support_vectors_[:,1],lw = 5)
        plt.scatter(X_p[:,0],X_p[:,1])
        plt.scatter(X_n[:,0],X_n[:,1],color = 'red')
        plt.tight_layout()

        #draw_line()
        coef = clf.coef_;intercept = clf.intercept_
        draw_line(coef[0],intercept[0],min(X[:,1]),max(X[:,1]))

plt.show()
```

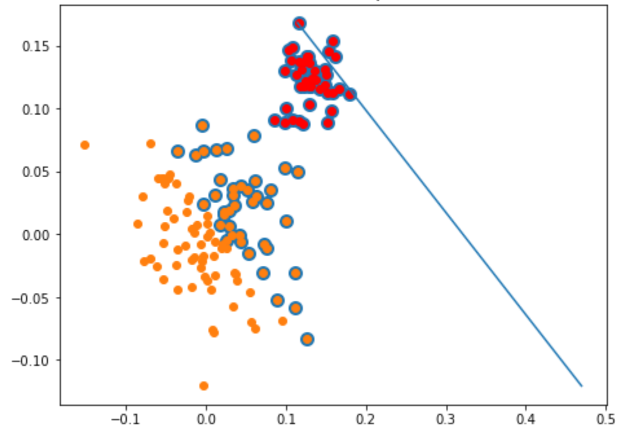


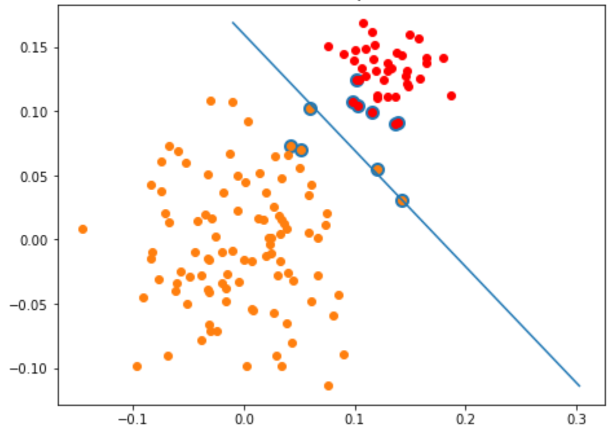
Dataset - D3, C =0.001



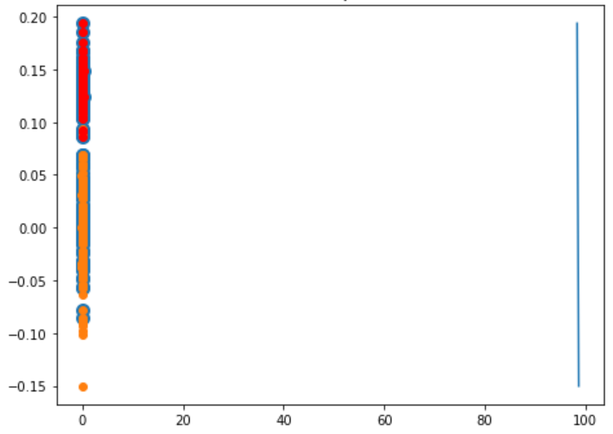
Dataset - D3, C =1



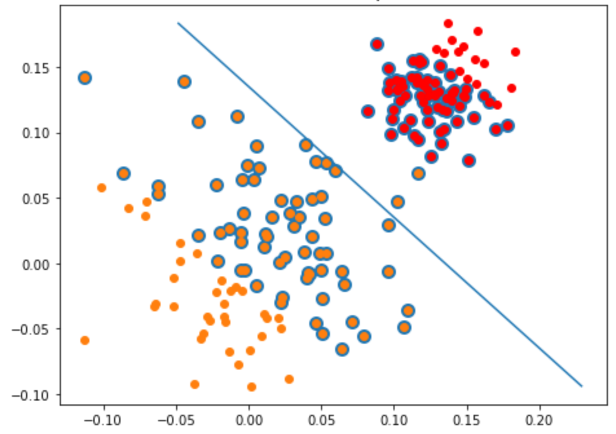
Dataset - D3, C =100



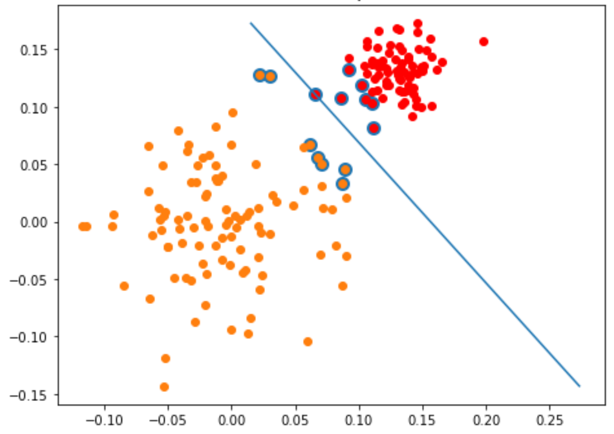
Dataset - D4, C =0.001



Dataset - D4, C =1



Dataset - D4, C =100



OBSERVATION :

Here from the above plots we can observe that,

1) For dataset 1, As the ratio of imbalance is (100 : 2) , we can clearly infer that the linear separating hyperplane underfits to the data for small value of 'C'(regularization parameter) & as 'C' value increases the model tends to give more importance to the error term in the loss function which eventually makes less error & thus, for C= 100 we are able to seperate positive and negative points perfectly.

2) For dataset 2, As the ratio of imbalance is (100 : 20) , still we can observe severe imbalance in the dataset because of which the linear hyperplane changes drastically as'C' increases. In this case as the number of negative points is quite large compared to previous case we are able to observe that the model is trying to fit the data as 'C' increases slightly. So, the optimal hyperparameter would lie somewhere near '100'.

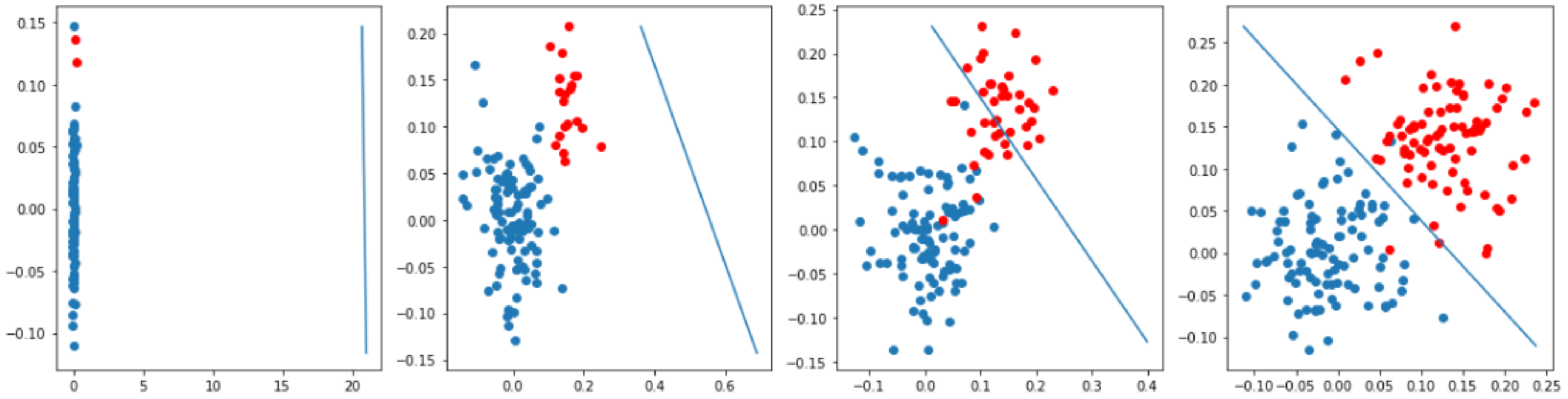
3) For dataset 3, As the ratio of imbalance is (100 : 40), where the number of minority points has increased significantly compared to previous case. So, here we can observe that the hyperplane tends to occupy its position in the space with the minimal impact of majority points.

4) For dataset 4, As the ratio of imbalance is (100 : 80), we can conclude that the data is somewhat balanced and hence, small value of 'C' triggers the hyperplane to underfit and large 'C' tends to slightly overfit . Therefore,optimal hyperparameter can be chosen to be values closer to '1'.

Task 2: Applying LR

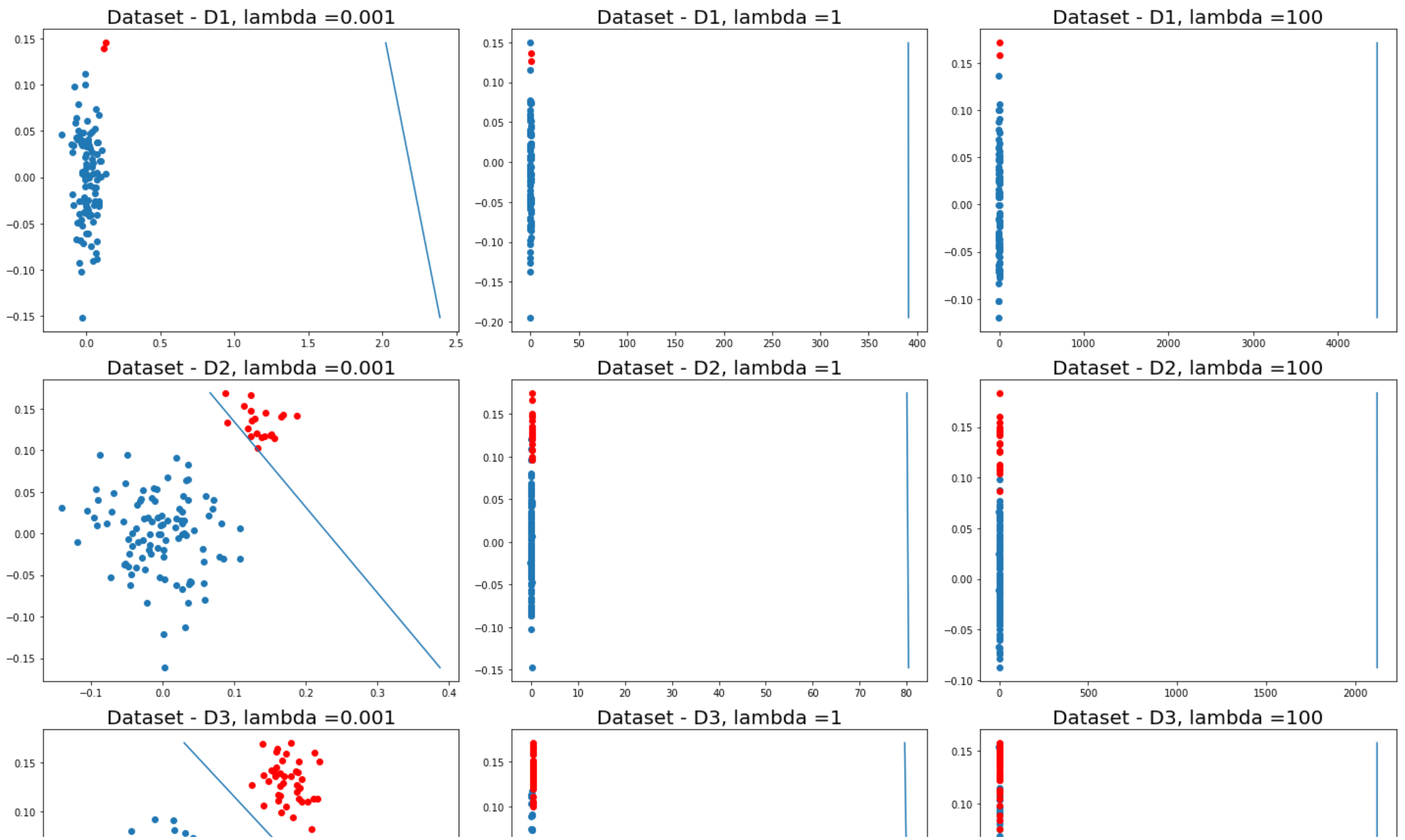
```
 you will do the same thing what you have done in task 1.1, except instead of SVM you apply logistic
regression
```

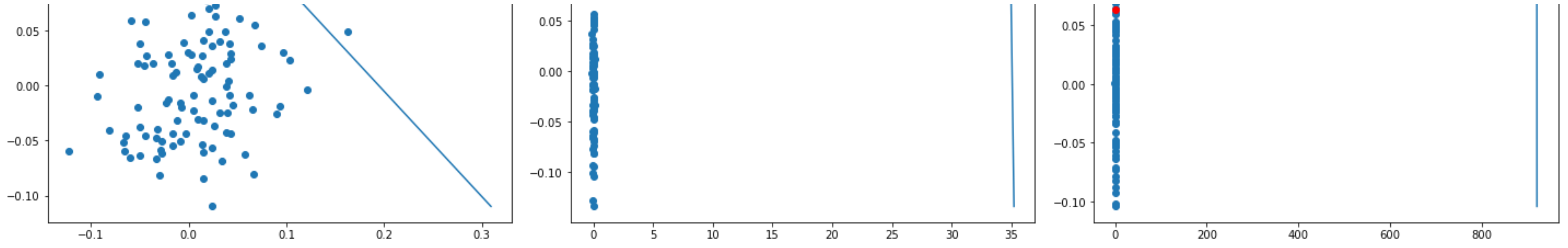
```
these are results we got when we are experimenting with one of the model
```

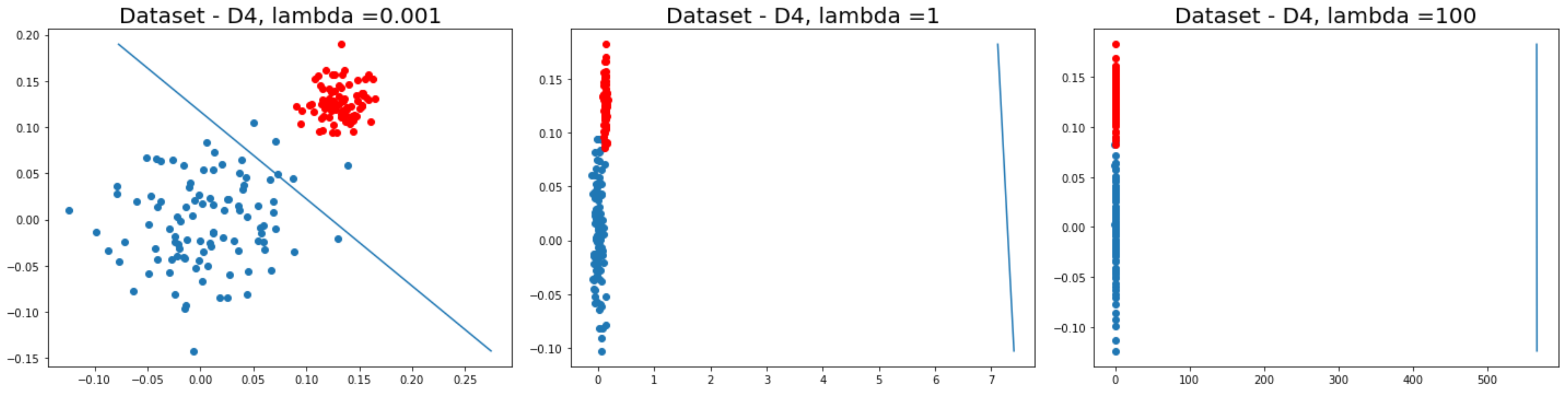


```
In [58]:  #Apply Log regression
          ratio = [(100,2),(100,20),(100,40),(100,80)]
          index = 0;
          plt.figure(figsize=(20,20))   # figure size

          for j,i in enumerate(ratio):
              reg_param = [0.001,1,100]  # diff hyperparamters

              for c in range(0,len(ratio)-1):
                  plt.subplot(4,3,index+1)       # subplot creation
                  index += 1
                  # extracting positive & negative data points according to the ratios initialized
                  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))# obtain input data points
                  y=np.vstack((y_p,y_n))# obtain target data

                  clf = SGDClassifier(alpha = reg_param[c],loss = 'log').fit(X,y) # fit the model(LR)

                  #lets plot the seperating hyperplane along with support vectors:
                  plt.title("Dataset - D{0}, lambda ={1}".format(j+1,reg_param[c]),fontsize = 20)
                  plt.scatter(X_p[:,0],X_p[:,1])
```

```
        plt.scatter(X_n[:,0],X_n[:,1],color = 'red')
        plt.tight_layout()

        #draw_line()
        coef = clf.coef_;intercept = clf.intercept_
        draw_line(coef[0],intercept[0],min(X[:,1]),max(X[:,1]))
plt.show()
```







OBSERVATION :

Here from the above plots we can observe that,

1) For dataset 1, As the ratio of imbalance is (100 : 2) , we can clearly infer that the linear separating hyperplane cannot seperate the positive & negative points for any value of 'lamda' and thus, this is the severe impact of majority points present in the data.

2) For dataset 2, As the ratio of imbalance is (100 : 20) , still we can observe severe imbalance in the dataset because of which the linear hyperplane changes drastically as 'lambda' increases. Also ,In this case as the number of negative points is quite large compared to previous case and with the apt hyperparameter tuning logistic regression is able to outperform Svm.

3) For dataset 3, As the ratio of imbalance is (100 : 40), where the number of minority points has increased significantly compared to previous case. So, here for small value of 'lambda = 0.001' we can observe that the hyperplane tends to

occupy its position in the space with the minimal impact of majority points and is able to fairly seperate positive & negative points.

4) For dataset 4, As the ratio of imbalance is (100 : 80), we can conclude that the data is somewhat balanced and here we can also infer that the hyperplane is moving away from the minority points compared to previous cases which is a good sign for us. hence, very large value of 'lambda' triggers the hyperplane to underfit and very small 'lambda' tends to slightly overfit . Therefore,optimal hyperparameter can be chosen to be values lying in [0.001/0.01/0.1].