In [1]:

```
import numpy as np
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 [2]:

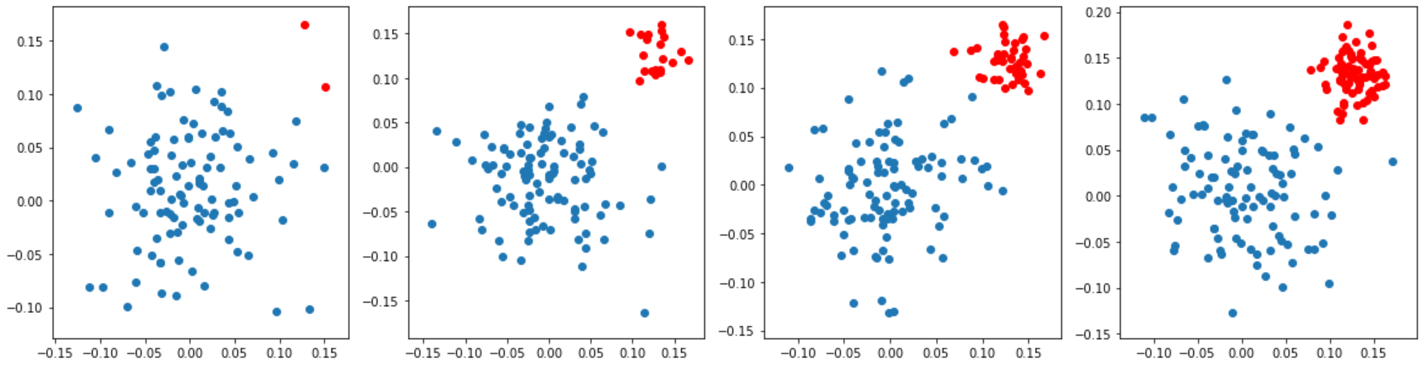
```
def draw_line(coef,intercept, mi, ma):    #both of them are y values
    # for the separating hyper plane ax+by+c=0, the weights are [a, b] and the intercept
is c
    # to draw the hyper plane we are creating two points
    # 1. ((b*min-c)/a, min) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a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

What if Data is imabalanced

```
1. As a part of this task you will observe how linear models work in case of data i
mbalanced
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 [3]:

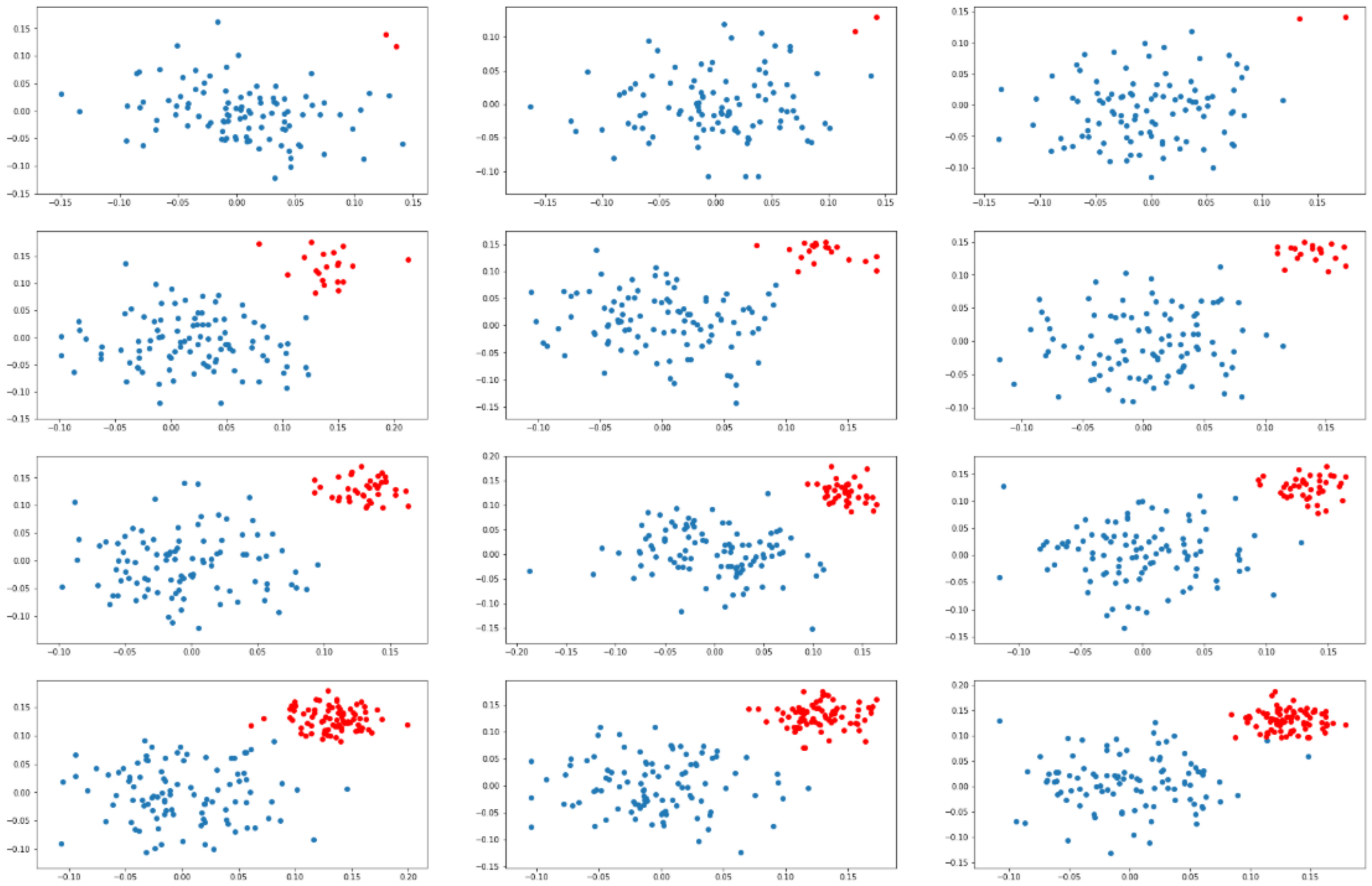
```
# here we are creating 2d imbalanced data points
data_X = list()
data_y = list()
X_mp = list()
X_mn = list()

ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
plt.figure(figsize=(20,5))
for j,i in enumerate(ratios):
    plt.subplot(1, 4, j+1)
    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_mp.append(X_p)
    X_mn.append(X_n)

    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')
    data_X.append(X)
    data_y.append(y)
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 ap
plying 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))
```

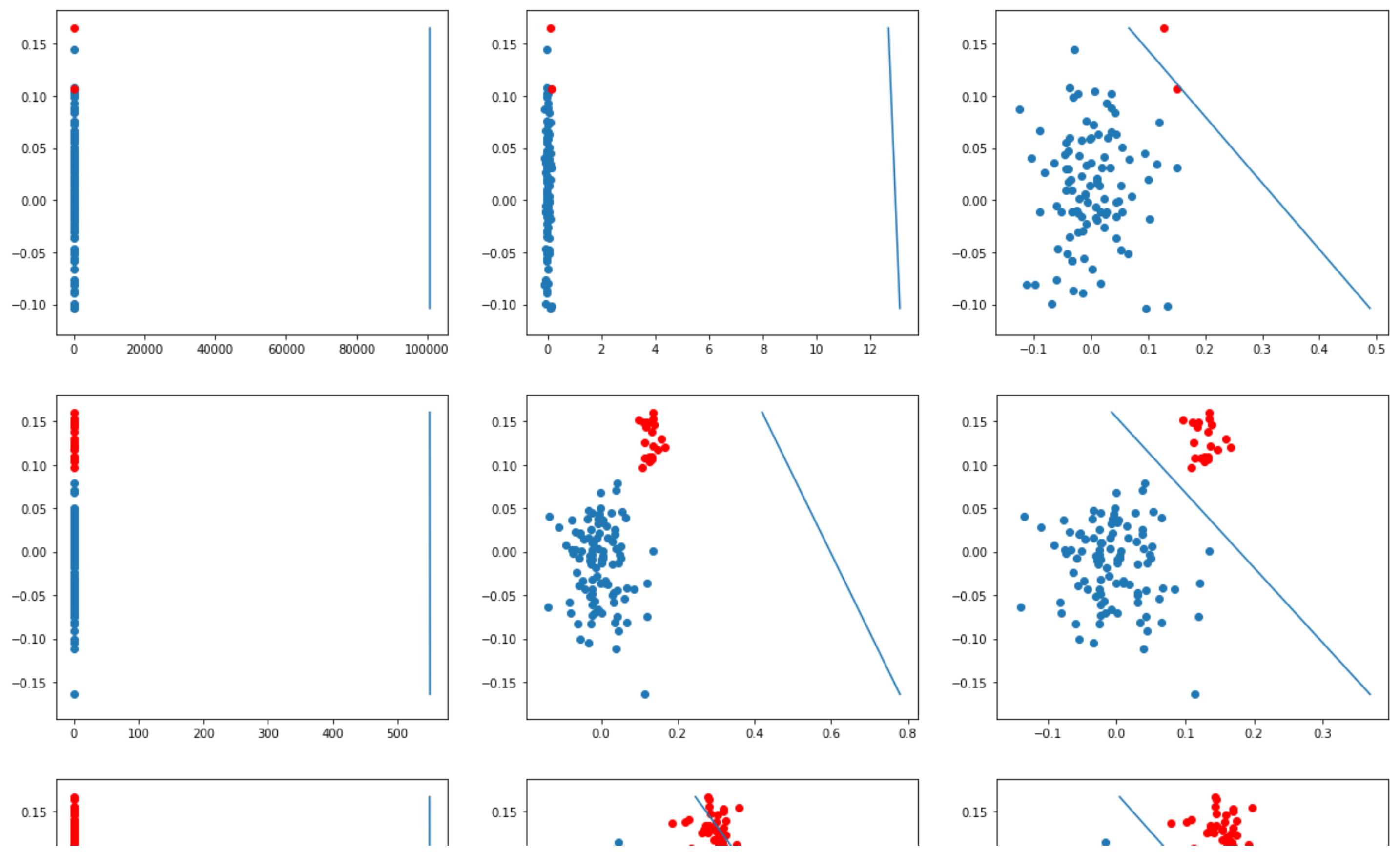
```
if you can do, you can represent the support vectors in different colors,
which will help us understand the position of hyper plane
```

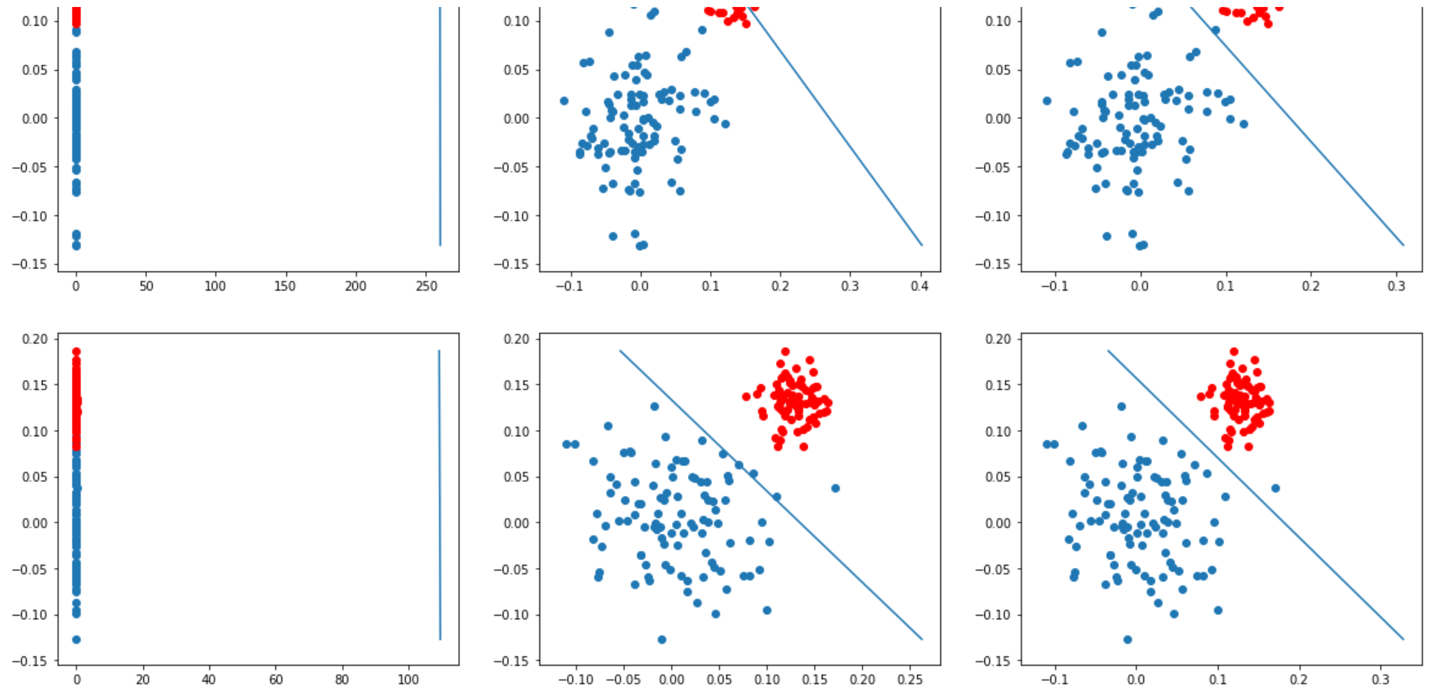
```
Write in your own words, the observations from the above plots, an
d
what do you think about the position of the hyper plane
```

```
check the optimization problem here https://scikit-learn.org/stable/modules/svm.htm
l#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.
```

In [4]:

```python
C = [0.001, 1, 100]



for i in range(len(data_X)):
    plt.figure(figsize=(20,5))
    for k,l in enumerate(C):
        plt.subplot(1,3,k+1)
        clf = SVC(kernel = 'linear',C = l)
        clf.fit(data_X[i],data_y[i])
        coef = clf.coef_
        intercept = clf.intercept_
        plt.scatter(X_mp[i][:,0],X_mp[i][:,1])
        plt.scatter(X_mn[i][:,0],X_mn[i][:,1],color='red')
        mi = min(list(np.transpose(data_X[i])[1]))
        ma = max(list(np.transpose(data_X[i])[1]))
        draw_line(coef[0],intercept[0],mi,ma)
```





In [6]:

```
print(clf.coef_)
```

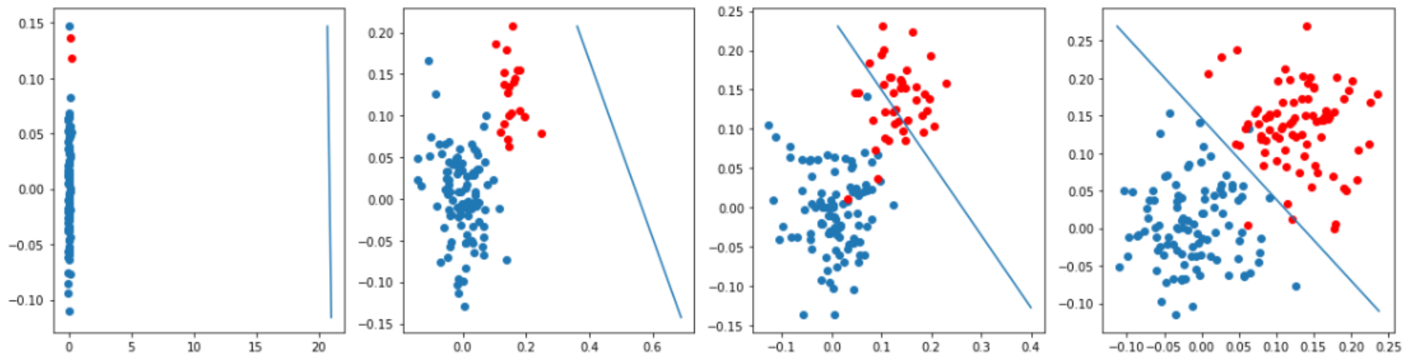
```
[[-27.56322545 -32.9432763 ]]
```

The above is the plot of decision surface of linear-SVM OBSERVATIONS 1. When the data is heavily imbalanced(100:2) the decision surface tries to match but not perfectly matching in case of C being 100. It does not classify in any other cases 2. When data imbalance ratio is (100:20) the decision surface does a good job with the C value 100 The decision surface tends to classify when the C value is 1 but does not do the classification 3. When data imbalance ratio is (100:40) The decision surface almost seperates the data points when C value is 1 and it does a pretty good job when C is 100 4. When data imbalance ration is (100:80) The decision surface mis-classifies almost two points when C value is 1 and it does classify properly with single misclassification when C value is 100. CONCLUSION Even with a tough imbalance ratio,a properly choosed hyperparameter could easily generate a decision surface to classify.

Task 2: Applying LR

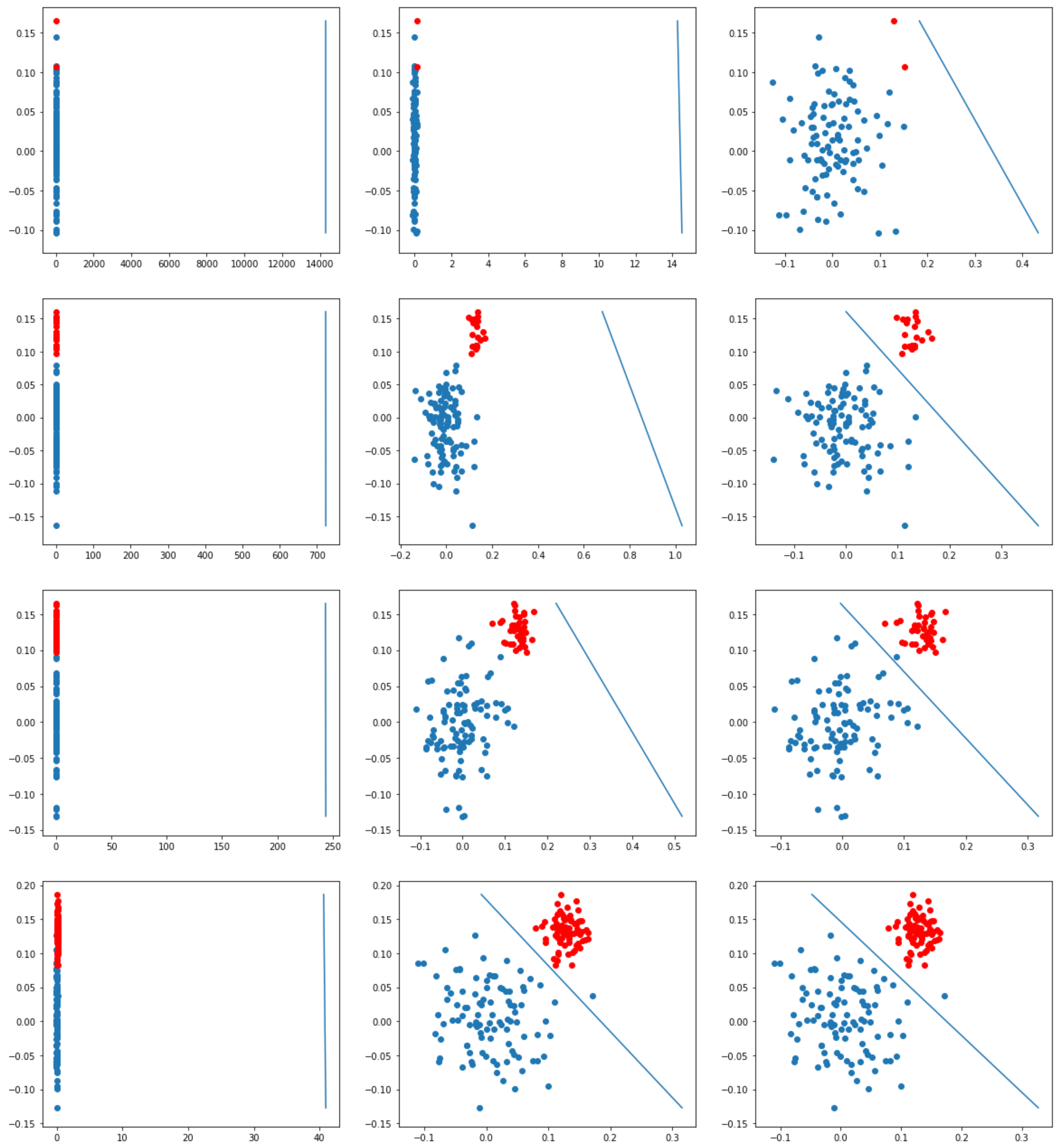
```
 you will do the same thing what you have done in task 1.1, except instead of SVM y
ou apply logistic regression 
```

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



In [5]:

```
#you can start writing code here.

for i in range(len(data_X)):
    plt.figure(figsize=(20,5))
    for k,l in enumerate(C):
        plt.subplot(1,3,k+1)
        clf = LogisticRegression(C = l)
        clf.fit(data_X[i],data_y[i])
        coef = clf.coef_
        intercept = clf.intercept_
        plt.scatter(X_mp[i][:,0],X_mp[i][:,1])
        plt.scatter(X_mn[i][:,0],X_mn[i][:,1],color='red')
        mi = min(list(np.transpose(data_X[i])[1]))
        ma = max(list(np.transpose(data_X[i])[1]))
        draw_line(coef[0],intercept[0],mi,ma)
```



The above is the plot of decision surface of logistic Regression OBSERVATIONS 1. As we can see when imbalance ratio is (100:2) logistic Regression does not do a good job as when compared to linear-svc. 2. None of the plots have properly classified when C = 1 unless there is good amount data (100:80) CONCLUSION Even though we choose the right regularization parameter, it is important to not have highly imbalanced data when we use logistic regression

as a classifier.