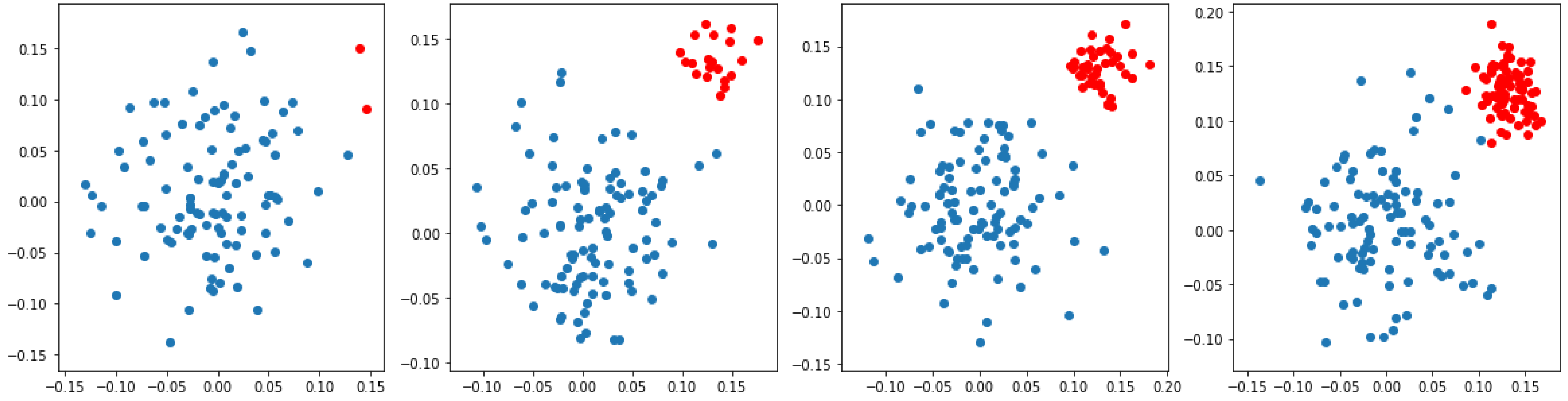
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, Normalizer
import matplotlib.pyplot as plt
from sklearn.svm import SVC
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
def draw_line(coef,intercept, mi, ma):
    # 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 minim
    # 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 maxim
    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 imabalanced

```
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
```

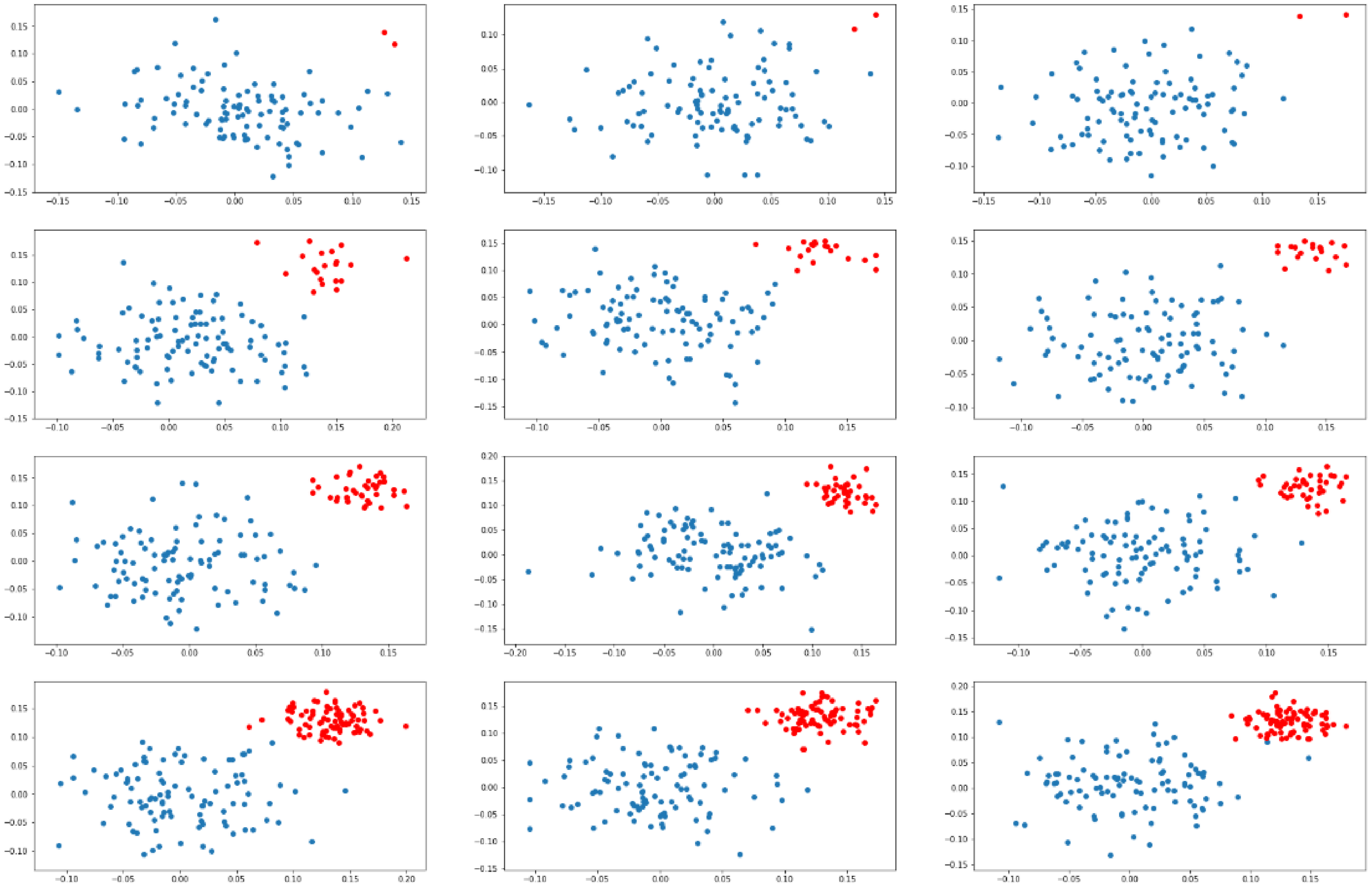
In [3]:
```python
# here we are creating 2d imbalanced data points
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
plt.figure(figsize=(20,5))
for j,i in enumerate(ratios):
    plt.subplot(1, 4, j+1)
    X_p=np.random.normal(0,0.05,size=(i[0],2))
    X_n=np.random.normal(0.13,0.02,size=(i[1],2))
    y_p=np.array([1]*i[0]).reshape(-1,1)
    y_n=np.array([0]*i[1]).reshape(-1,1)
    X=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 (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC)](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC)) and LR (sklearn.linear_model.LogisticRegression (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)) with different regularization strength [0.001, 1, 100]

Task 1: Applying SVM

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

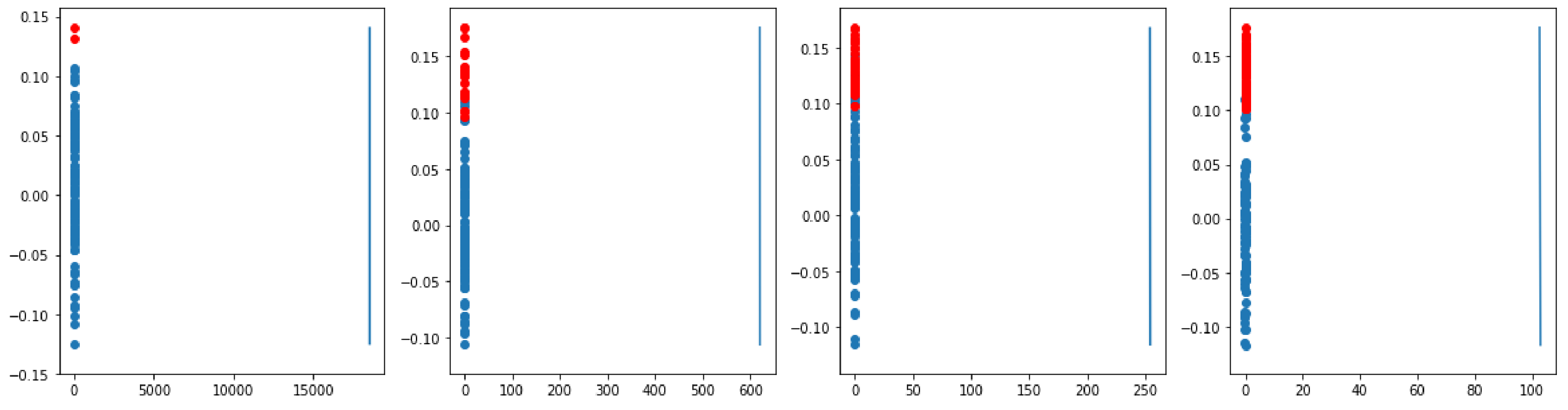


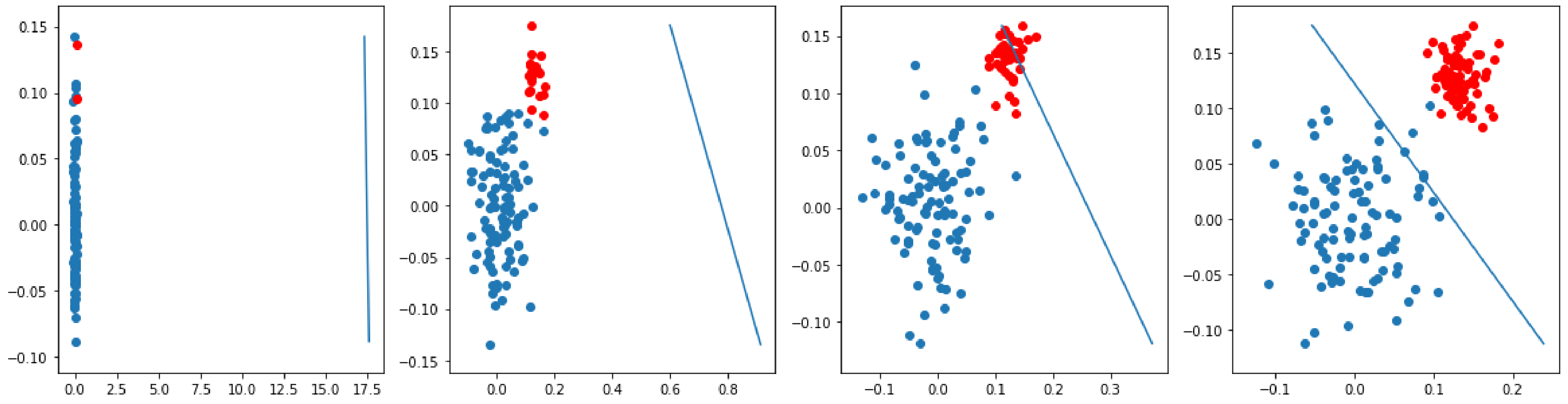
```
in each of the cell[i][j] you will be drawing the hyper plane that you get after applying SVM (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html) on ith dataset and
        jth learnig rate

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
```

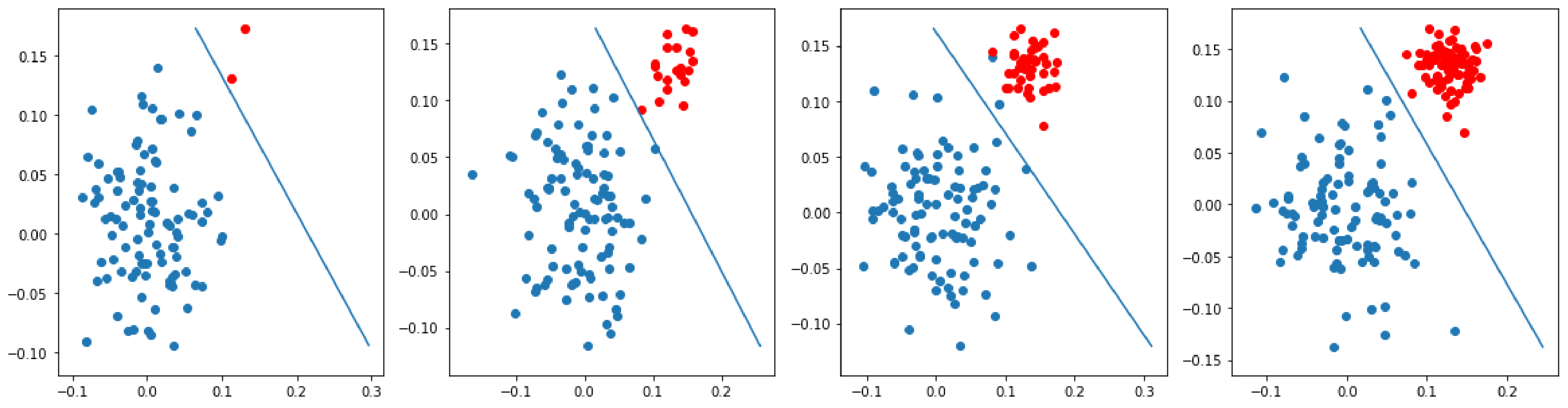
Write in your own words, the observations from the above plots, and what do you think about the position of the hyper plane

```
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.
```

In [40]:
```
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
for c in [0.001,1,100]:
    model = SVC(C=c,kernel='linear')
    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=np.vstack((X_p,X_n))
        y=np.vstack((y_p,y_n))
        model.fit(X,y)
        inter = model.intercept_
        coef = model.coef_
        mi = min(X[:,1])
        ma = max(X[:,1])
        plt.scatter(X_p[:,0],X_p[:,1])
        plt.scatter(X_n[:,0],X_n[:,1],color='red')
        draw_line(coef[0],inter,mi,ma)
plt.show()
```




0.15
0.10
0.05
0.00
−0.05
−0.10
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
0.0 0.2 0.4 0.6 0.8
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.1 0.0 0.1 0.2 0.3
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.1 0.0 0.1 0.2



Observation - classification using SVM

Case 1 - When hyperparameter C = 0.001, model is not useful in classifying the dataset of all ratios. All the datapoints (of both classes) lie on the same side of hyperplane and they are a bit overlapped as well.

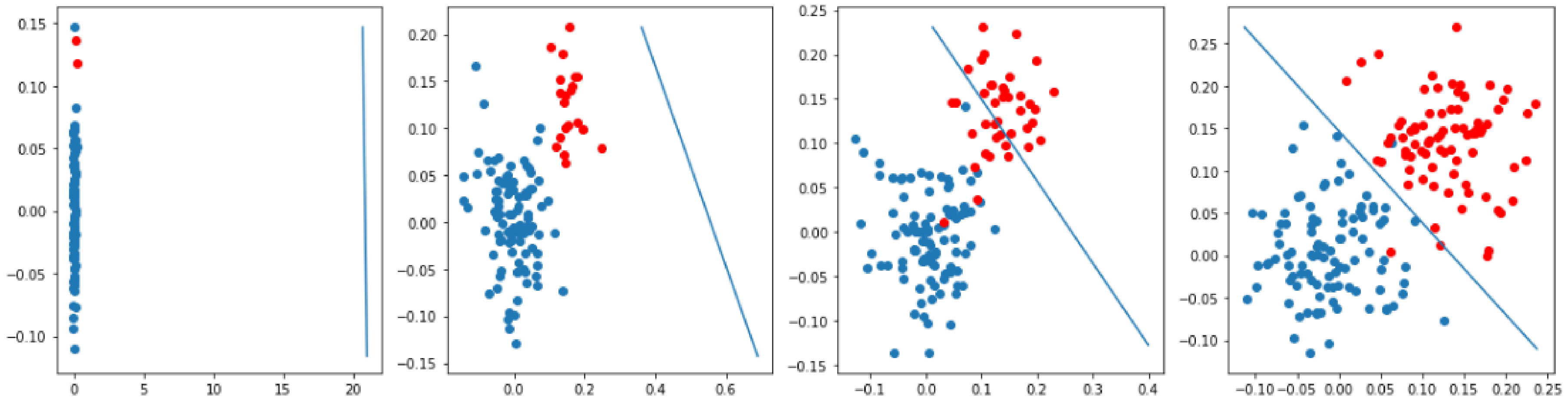
Case 2 - When hyperparameter C = 1, it works comparatively better on dataset with positive and negative classes in the ratio 100:80, only few positive points are misclassified (they are present on the the negative side or other side of hyperplane). For dataset with ratios 100:2 and 100:20, it's same as first case, all datapoints of both classes are present on the same side of hyperplane. For dataset with ratio 100:40, all positive points are correctly classified, but a significant number of negative points are misclassified. (they lie on positive points' side). The model in this case doesn't work well on imbalanced datases

Case 3 - When hyperparameter C = 100, dataset of all ratios are correctly classified. It works fairly well with both balanced and imbalanced datasets.

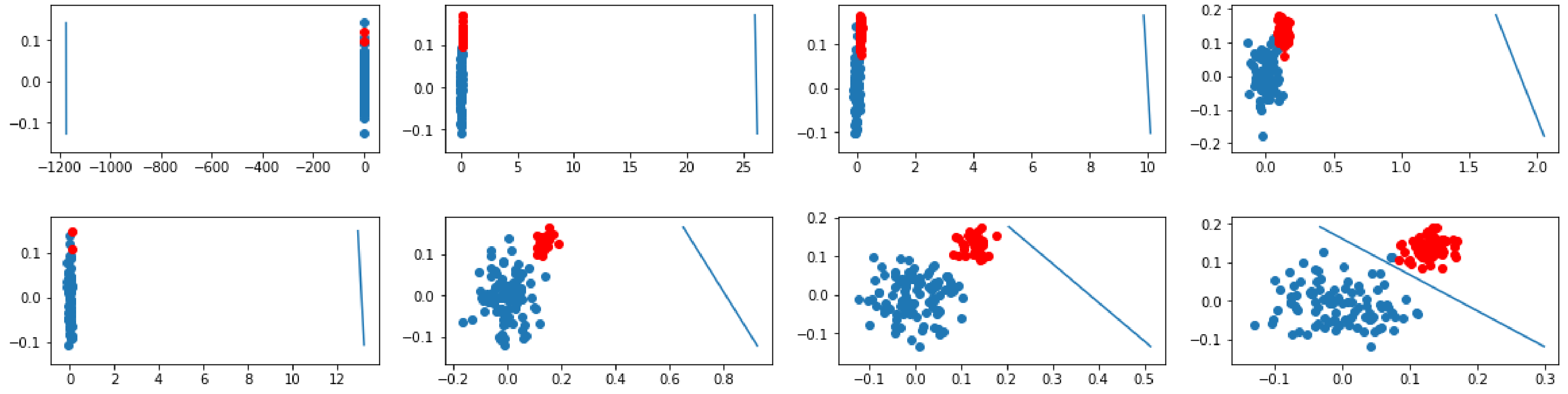
Task 2: Applying LR

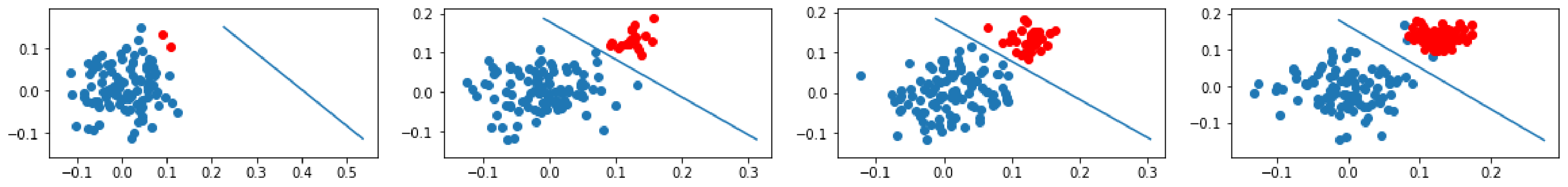
```
you will do the same thing what you have done in task 1.1, except instead of SVM you apply logistic regression (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
```

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



In [46]:
```
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
for c in [0.001,1,100]:
    model = LogisticRegression(C=c,n_jobs=-1)
    plt.figure(figsize=(20,2))
    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=np.vstack((X_p,X_n))
        y=np.vstack((y_p,y_n))
        model.fit(X,y)
        inter = model.intercept_
        coef = model.coef_
        mi = min(X[:,1])
        ma = max(X[:,1])
        plt.scatter(X_p[:,0],X_p[:,1])
        plt.scatter(X_n[:,0],X_n[:,1],color='red')
        draw_line(coef[0],inter,mi,ma)
plt.show()
```





Observation - classification using Logistic Regression

Case 1 - When hyperparameter C = 0.001, model is not useful in classifying the dataset of all ratios. All the datapoints (of both classes) lie on the same side of hyperplane and they are a bit overlapped as well.

Case 2 - When hyperparameter C = 1, it works comparatively better on dataset with positive and negative classes in the ratio 100:80, only one positive point is misclassified. For other ratios, it's same as first case, all datapoints of both classes are present on the same side of hyperplane. For The model in this case doesn't work well when dataset is imbalanced

Case 3 - When hyperparameter C = 100, dataset of all ratios are correctly classified except 100:2. This shows that it doesn't work well when data is highly imbalanced

The working of models SVC and LogisticRegression is similar with few differences.

References

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC)

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic%20regression#sklearn.linear_model.LogisticRegression (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic%20regression#sklearn.linear_model.LogisticRegression)

In [ ]:

```
```