In [2]:

```
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 [3]:

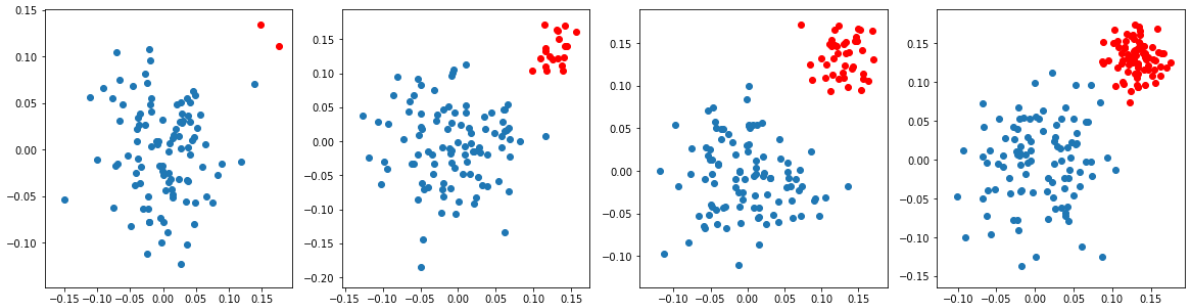
```
def draw_line(coef,intercept, mi, ma):
    # for the separating hyper plane ax+by+c=0, the weights are [a, b] and the inter
    # to draw the hyper plane we are creating two points
    # 1. ((b*min-c)/a, min) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here ir
    # 2. ((b*max-c)/a, max) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here ir
    points=np.array([[((-coef[1]*mi - intercept)/coef[0]), mi],[((-coef[1]*ma - inte
    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 ra te.
3. below we have created 4 random datasets which are linearly separable and having class imbalance
4. in the first dataset the ration 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 10 0:80

In [4]:

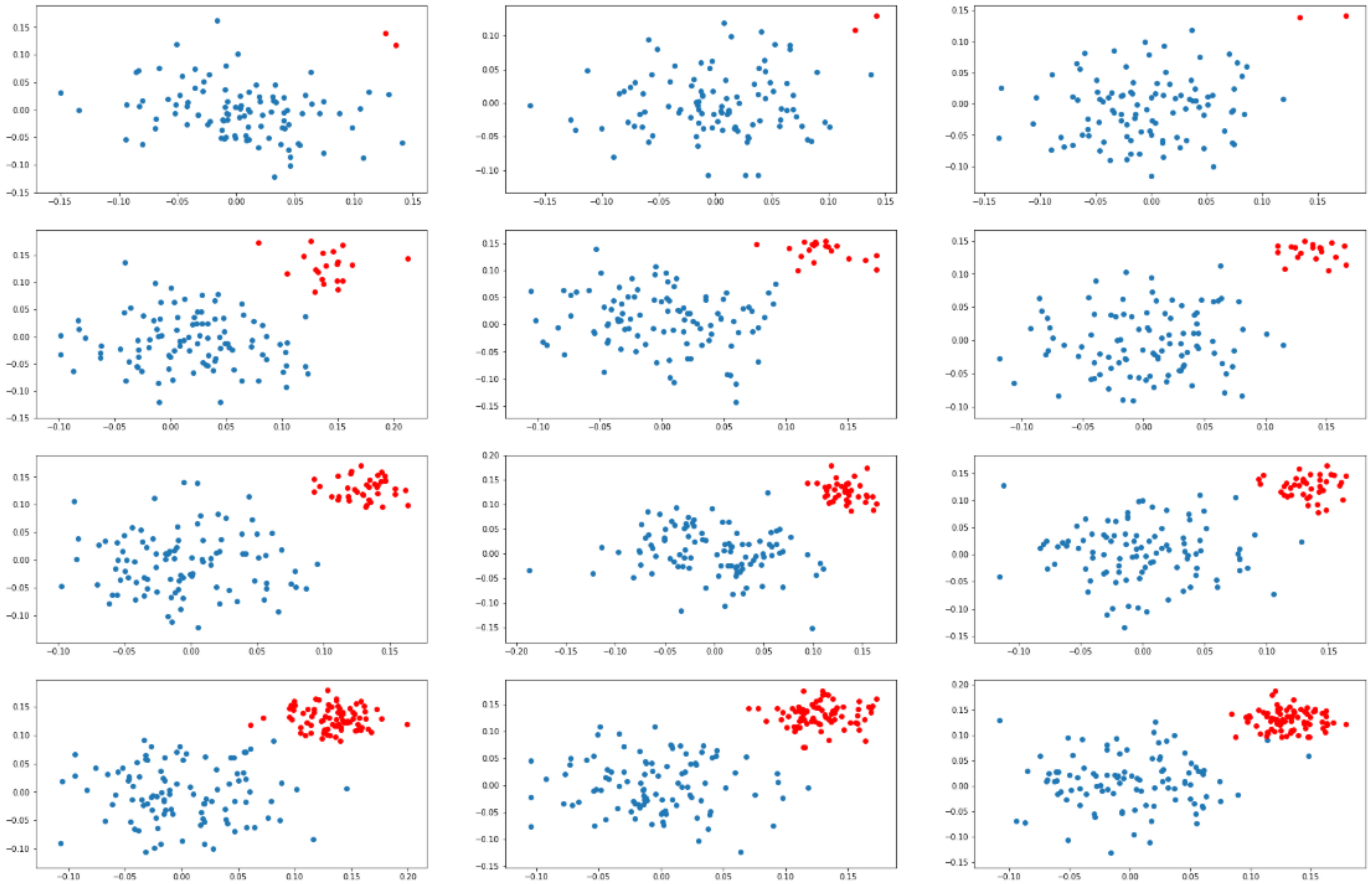
```
# here we are creating 2d imbalanced data points
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
plt.figure(figsize=(20,5))
for j,i in enumerate(ratios):
    plt.subplot(1, 4, j+1)
    X_p=np.random.normal(0,0.05,size=(i[0],2))
    X_n=np.random.normal(0.13,0.02,size=(i[1],2))
    y_p=np.array([1]*i[0]).reshape(-1,1)
    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)) 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 a fter 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, wh ich will help us understand the position of hyper plane

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

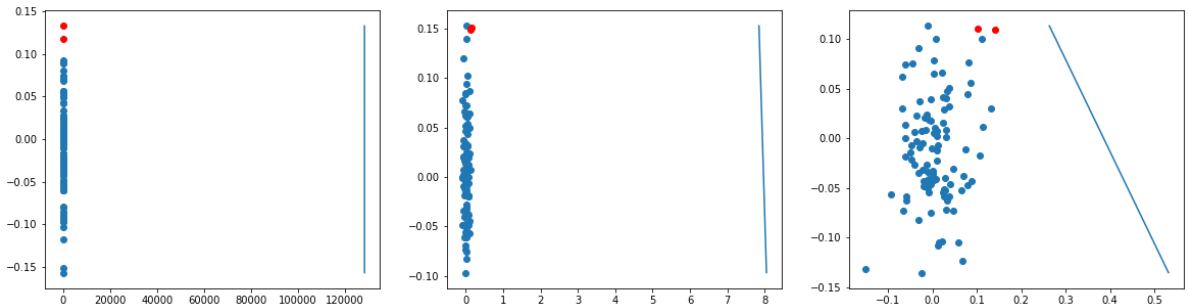
check the optimization problem here https://scikit-learn.org/stable/modules/ svm.html#mathematical-formulation

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

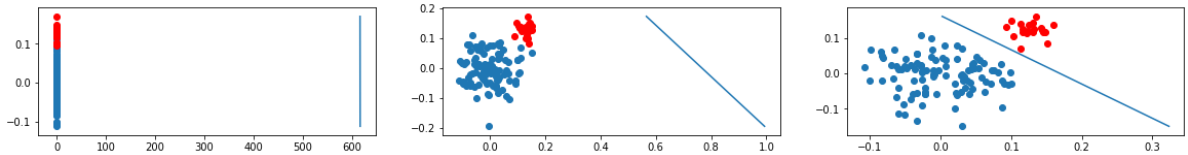
In [6]:

```
#you can start writing code here.
from sklearn.svm import SVC

# here we are creating 2d imbalanced data points

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

for j,i in enumerate(ratios):
    plt.figure(figsize=(20,5))
    for indx, regulizer in enumerate(regularization_strength):
        plt.subplot(j+1, 3, indx+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 = SVC(kernel='linear',C=regulizer)
        model.fit(X,y)
        print(model.coef_)
        coef = model.coef_[0]
        intercept = model.intercept_[0]
        mi = X[:,1].min()
        ma = X[:,1].max()
        draw_line(coef,intercept, mi, ma)
        plt.scatter(X_p[:,0],X_p[:,1])
        plt.scatter(X_n[:,0],X_n[:,1],color='red')
    plt.show()
```

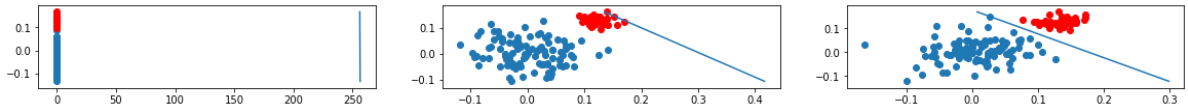
```
[[-7.78546779e-06 -1.17143595e-04]]
[[-0.12765879 -0.1028781 ]]
[[-4.49318249 -4.84663705]]
```



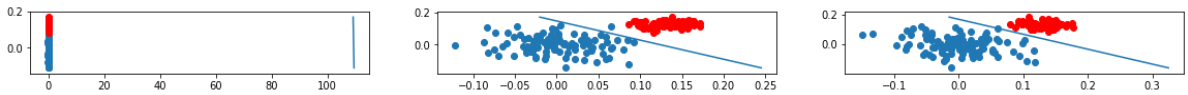
```
[[-0.00162159 -0.00140255]]
[[-1.42651684 -1.66128962]]
[[-14.88512439 -15.34811662]]
```



```
[[-0.00390278 -0.00414664]]
[[-3.54525837 -3.72403055]]
[[-19.57521472 -19.65433624]]
```



```
[[-0.00913205 -0.01013265]]
[[-7.25214102 -6.03258121]]
[[-17.16296396 -16.81149777]]
```



From above plots we can see how the model is performing in the case of imbalance data, when we move on by balancing the class data we can see SVD model is able to divide model with more accurate

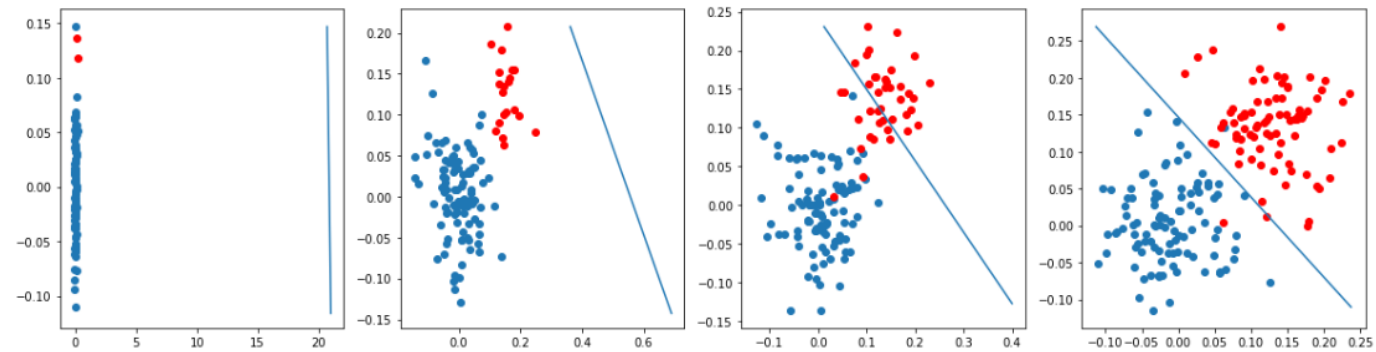
As we shown in the above plots

1. for 100:20 - the plane which divides the datapoints is too far and incorrece
2. for 100:40 - the plane which divides the datapoints is able to divide but with more false postives
3. for 100:80 - the plane which divides the datapoints is good, it is able to divide all most all the points with less false positives

Task 2: Applying LR

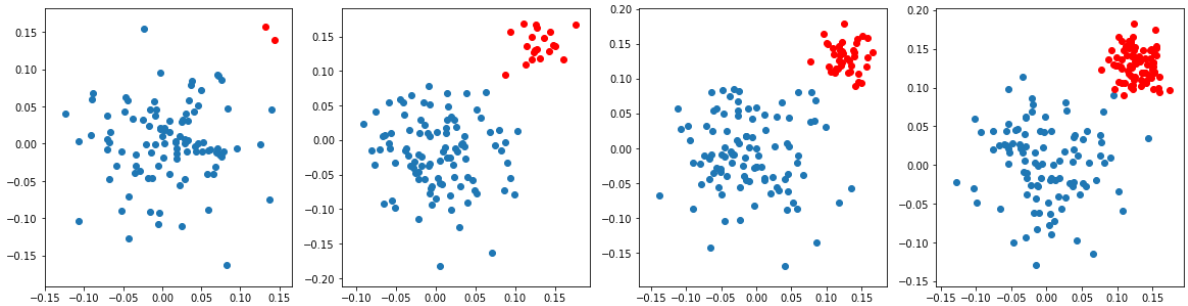
you will do the same thing what you have done in task 1.1, except instead of SVM you apply [logistic regression (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

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



In [7]:

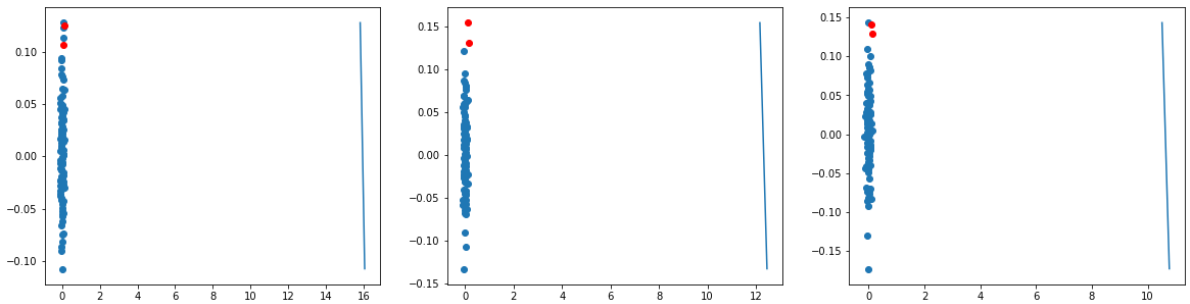
```
#you can start writing code here.
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
# here we are creating 2d imbalanced data points
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
plt.figure(figsize=(20,5))
for j,i in enumerate(ratios):
    plt.subplot(1, 4, j+1)
    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))
    model.fit(X,y)
    coef = model.coef_[0]
    intercept = model.intercept_[0]
    mi = X[:,1].min()
    ma = X[:,1].max()
#     draw_line(coef,intercept, mi, ma)
    plt.scatter(X_p[:,0],X_p[:,1])
    plt.scatter(X_n[:,0],X_n[:,1],color='red')
plt.show()
```



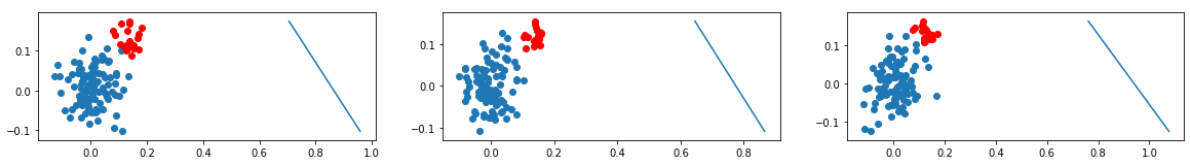
In [8]:

```
#you can start writing code here.
from sklearn.svm import SVC

# here we are creating 2d imbalanced data points

regularization_strength = [0.001, 1, 100]
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
model = LogisticRegression()
for j,i in enumerate(ratios):
    plt.figure(figsize=(20,5))
    for indx, regulizer in enumerate(regularization_strength):
        plt.subplot(j+1, 3, indx+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)
        print(model.coef_)
        coef = model.coef_[0]
        intercept = model.intercept_[0]
        mi = X[:,1].min()
        ma = X[:,1].max()
        draw_line(coef,intercept, mi, ma)
        plt.scatter(X_p[:,0],X_p[:,1])
        plt.scatter(X_n[:,0],X_n[:,1],color='red')
    plt.show()
```

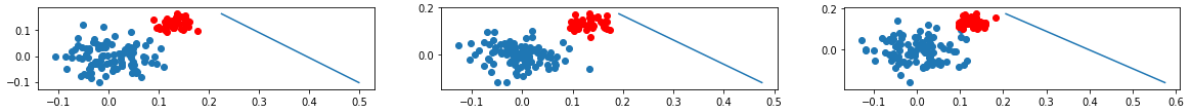
```
[[-0.18661732 -0.18942586]]
[[-0.24123717 -0.24701415]]
[[-0.27937569 -0.23091546]]
```



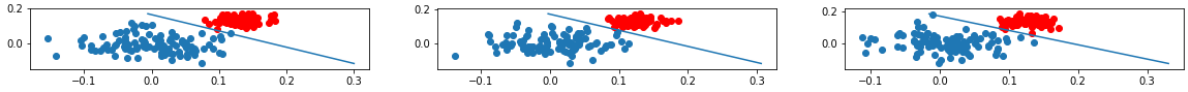
```
[[-1.88459712 -1.72851248]]
[[-2.08074446 -1.72614256]]
[[-1.72042651 -1.88361307]]
```



```
[[-2.84462847 -2.92491166]]
[[-3.10207028 -2.99431268]]
[[-2.821949   -3.10291267]]
```



```
[[-3.96429748 -4.25802197]]
[[-3.76865347 -4.00664201]]
[[-3.76988728 -4.19041696]]
```



From above plots we can see how the model is performing in the case of imbalance data, when we move on by balancing the class data we can see LogisticRegression model is able to divide model with more accurate

##observations

1. for 100:20 - the plane which divides the datapoints is too far and incorrece
2. for 100:40 - the plane which divides the datapoints is able to divide but with more false postives
3. for 100:80 - the plane which divides the datapoints is good, it is able to divide all most all the points with less false positives
4. As similar to of svm logestic regression also able to perform well when there is good data, it show how will model performs based on data