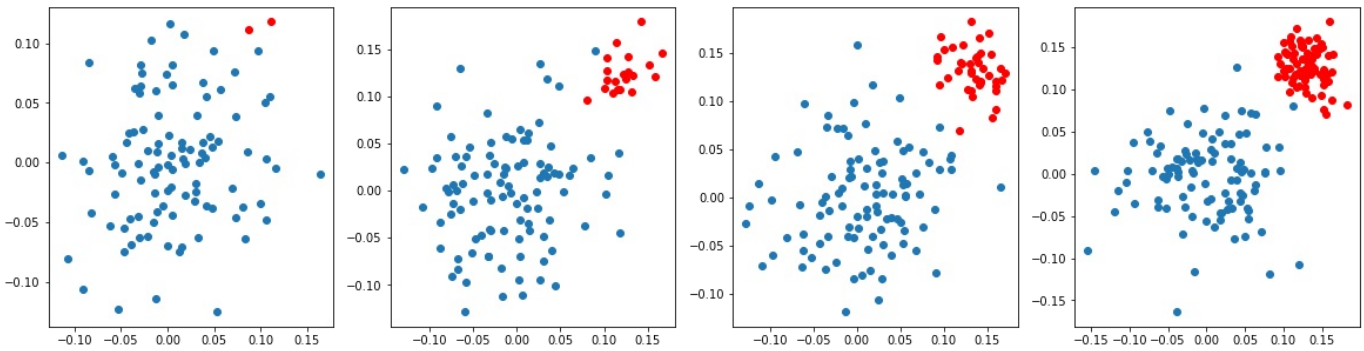
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, Normalizer
import matplotlib.pyplot as plt
from sklearn.svm 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 th
    # 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 th
    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

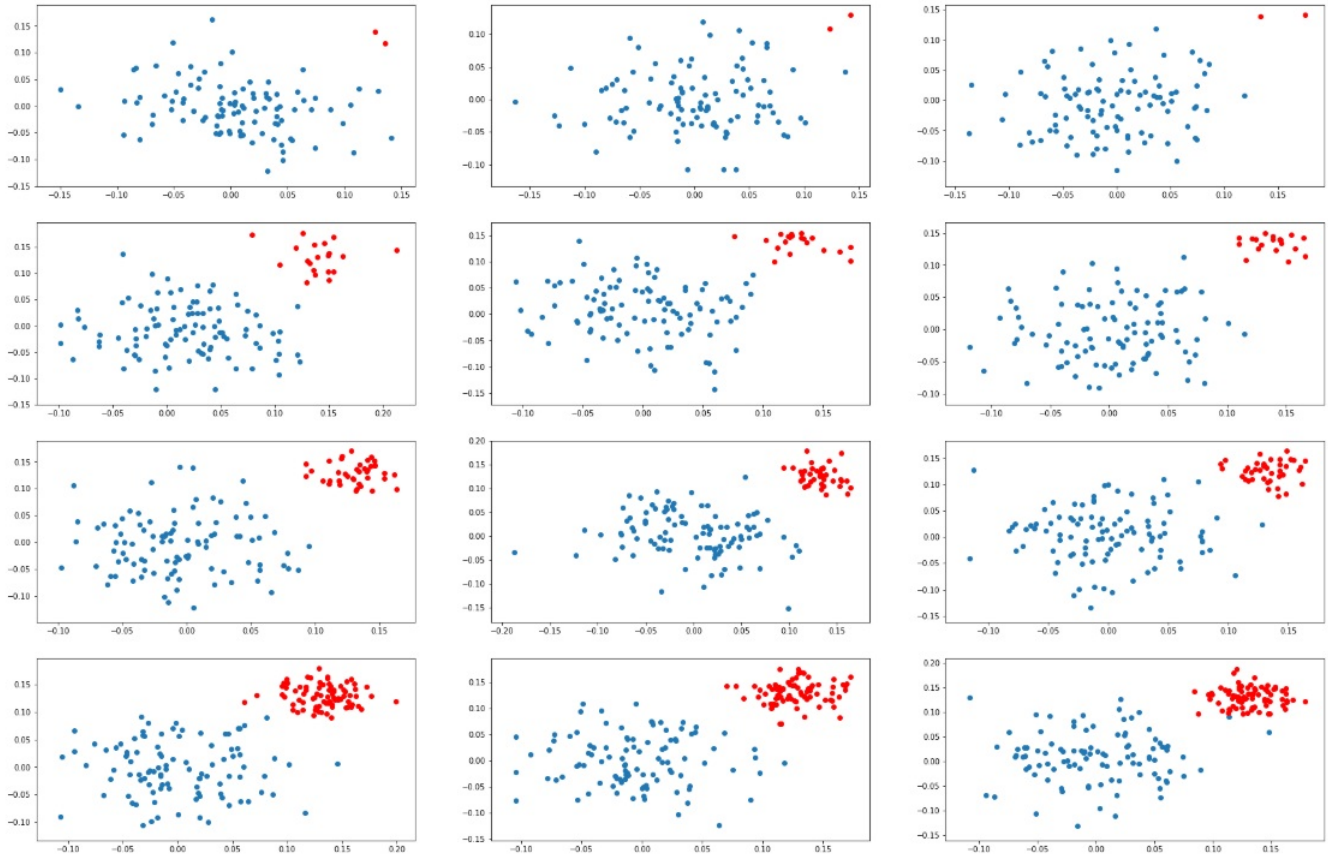
In [3]:
```
# here we are creating 2d imbalanced data points
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
plt.figure(figsize=(20,5))
for j,i in enumerate(ratios):
    plt.subplot(1, 4, j+1)
    X_p=np.random.normal(0,0.05,size=(i[0],2))
    X_n=np.random.normal(0.13,0.02,size=(i[1],2))
    y_p=np.array([1]*i[0]).reshape(-1,1)
    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) 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



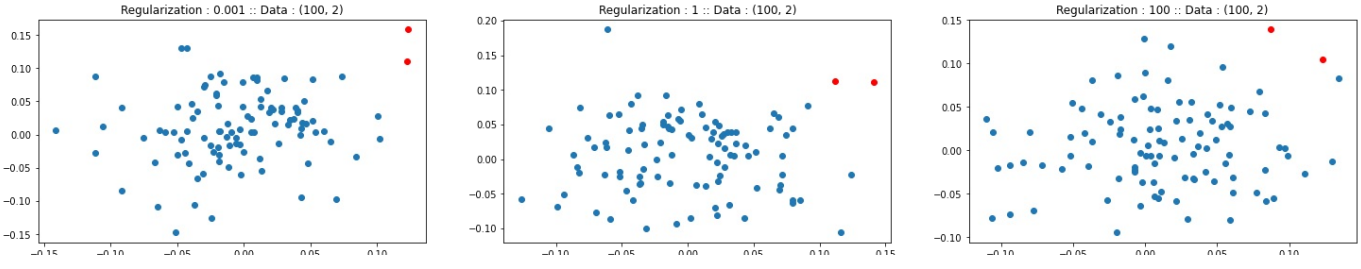
in each of the cell[i][j] you will be drawing the hyper plane that you get after applying SVM on ith dataset and
jth learnig rate

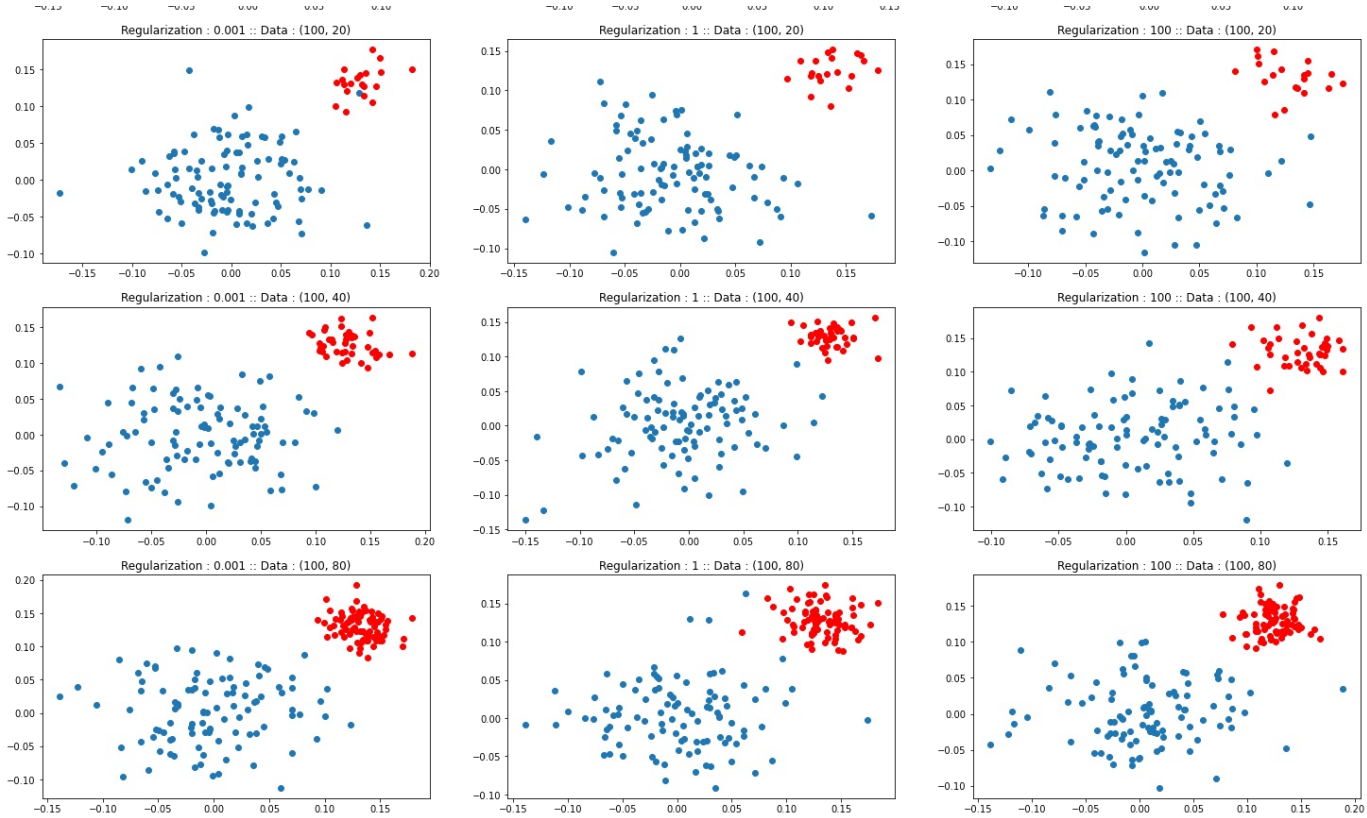
i.e

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))

In [4]:

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

reg_strength =[0.001, 1, 100]
plt.figure(figsize = (25,20))
img_index = 1

for index,i in enumerate(ratios):
    for j in range(0, 3):
        plt.subplot(4, 3, img_index)
        img_index +=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.title(f'Regularization : {reg_strength[j]} :: Data : {i}')

plt.show()
```





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

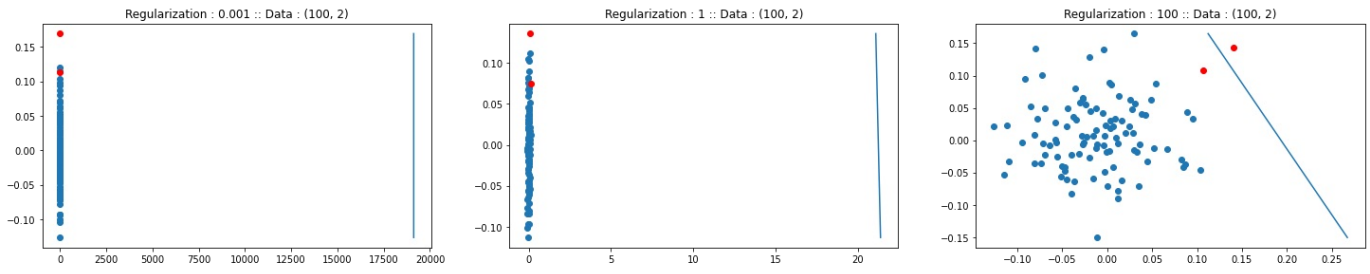
Write in your own words, the observations from the above plots, and what do you think about the position of the hyper plane

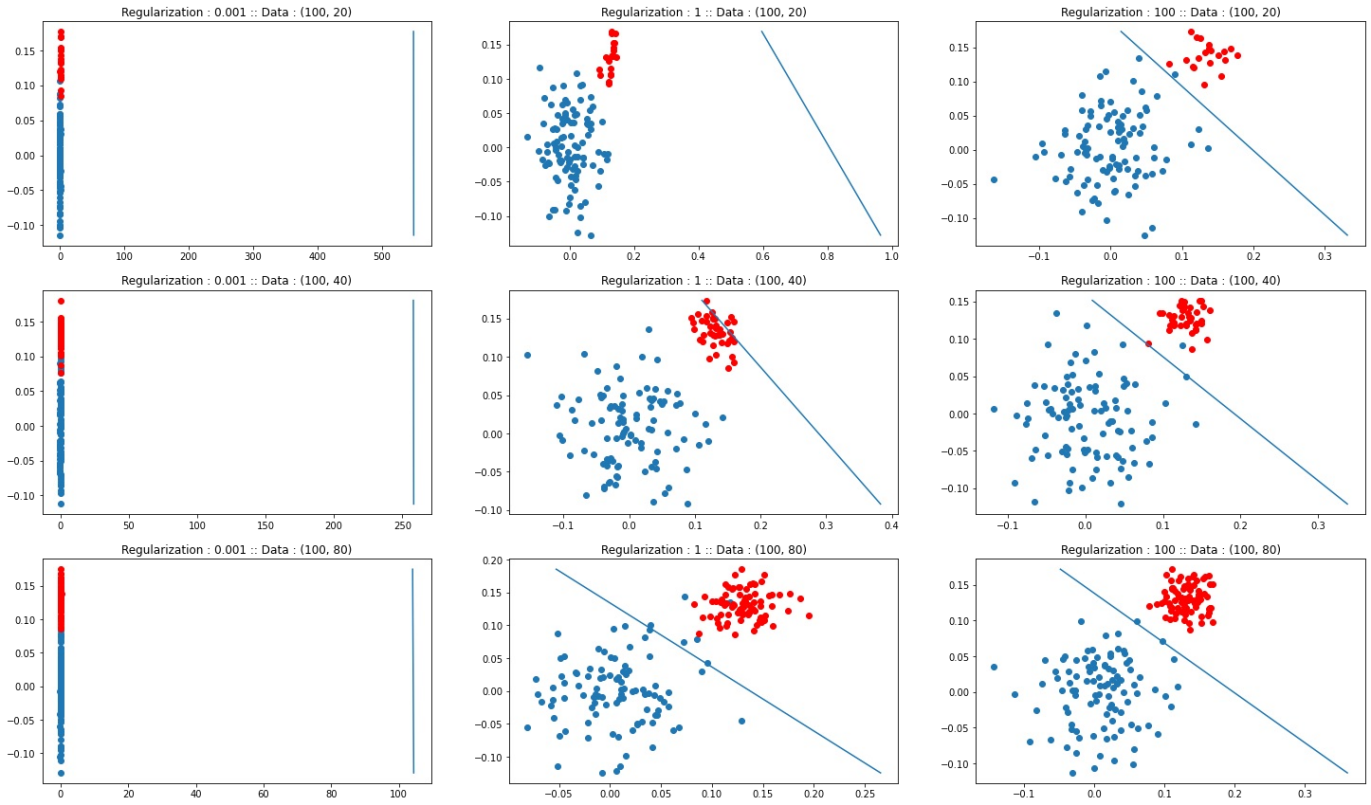
check the optimization problem here https://scikit-learn.org/stable/modules/svm.html#mathematical-formulation

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

In [5]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
# https://scikit-learn.org/stable/auto_examples/svm/plot_svm_margin.html

plt.figure(figsize = (25,20))
img_index = 1

for index,i in enumerate(ratios):
    for j in range(0, 3):
        svc_model = SVC(C=reg_strength[j], kernel="linear")
        plt.subplot(4, 3, img_index)
        img_index +=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))
        svc_model.fit(x,y)
        plt.scatter(X_p[:,0],X_p[:,1])
        plt.scatter(X_n[:,0],X_n[:,1],color='red')
        plt.title(f'Regularization : {reg_strength[j]} :: Data : {i}')

# https://stackoverflow.com/a/65815689
        co_eff = svc_model.coef_[0]
        intercept = svc_model.intercept_
        margin = 1 / (np.sqrt(np.sum(svc_model.coef_ ** 2)))
        draw_line(co_eff, intercept, min(x[:,1]), max(x[:,1]))

plt.show()
```





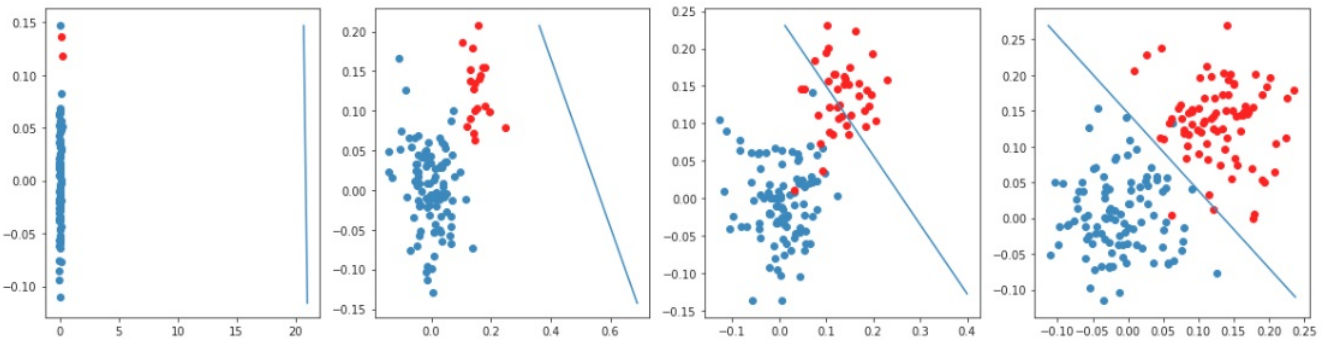
Observation

- While the Regularization value is small the keeps fails to classify the data points despite of the dataset imbalance. (Column-1 is an example).
- The SVM calssifier fails to classify data points in an imbalance dataset with lower regularization value's (Row-1)
- As increasing regularization parameter even with highly imbalance dataset, SVM starts to try to classify better and better (Row-2)
- With increasing regularization parameter and reduced imbalanced data ratio model delivers better classification (Row-3).
- When the dataset imbalance ratio reduced and in increasing regularization values the model classifies the points better and better (Row-4).
- In Row-4 its clear, with higher regularization value the decision boundary changed inorder to reduce the classification error.

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

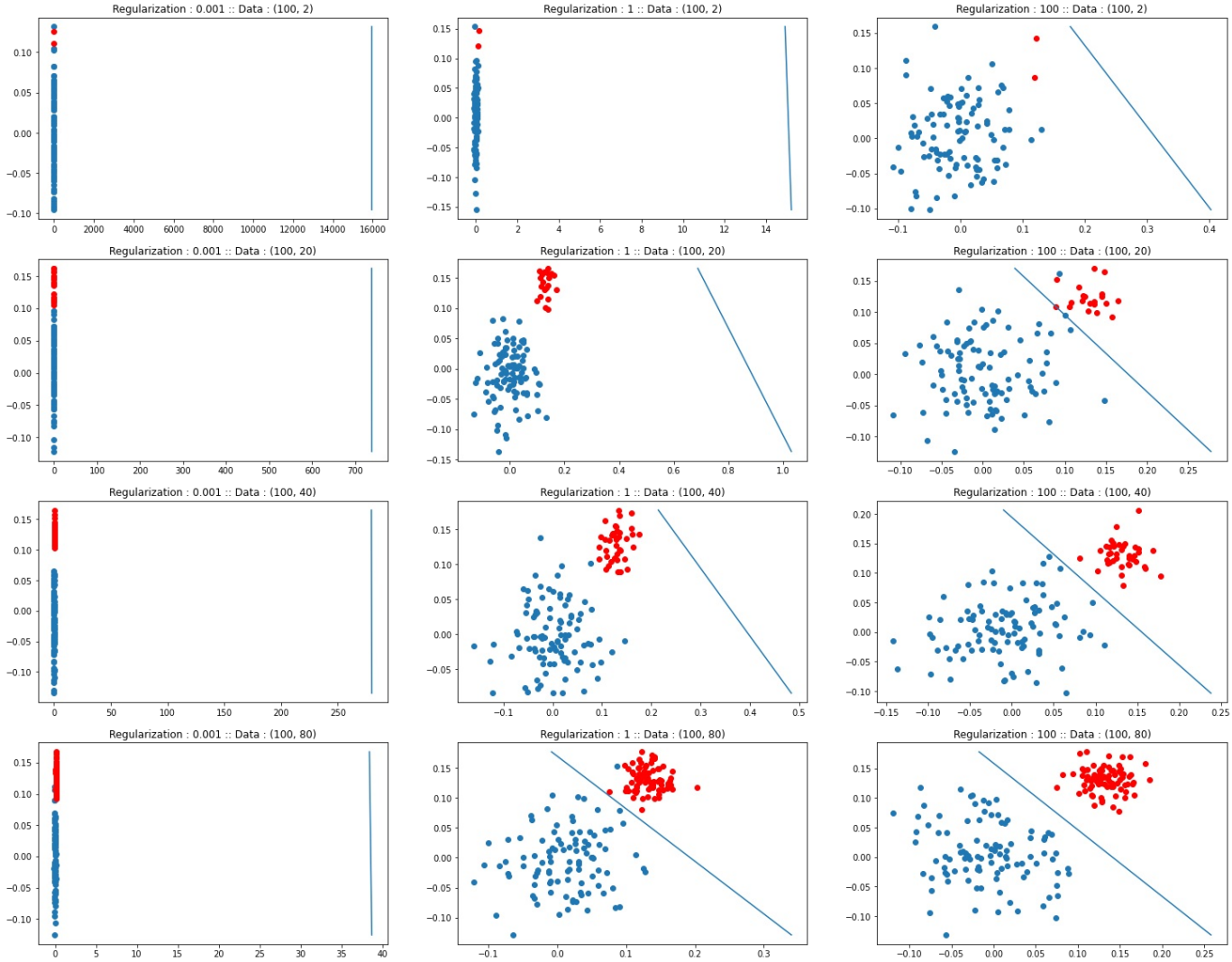
these are results we got when we are experimenting with one of the model



In [6]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

reg_strength =[0.001, 1, 100]
plt.figure(figsize = (25,20))
img_index = 1
```

```
for index,i in enumerate(ratios):
    for j in range(0, 3):
        lr_model = LogisticRegression(C=reg_strength[j])
        plt.subplot(4, 3, img_index)
        img_index +=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))
        lr_model.fit(x,y)
        plt.scatter(X_p[:,0],X_p[:,1])
        plt.scatter(X_n[:,0],X_n[:,1],color='red')
        plt.title(f'Regularization : {reg_strength[j]} :: Data : {i}')

# https://stackoverflow.com/a/65815689
        co_eff = lr_model.coef_[0]
        intercept = lr_model.intercept_
        margin = 1 / (np.sqrt(np.sum(lr_model.coef_ ** 2)))
        draw_line(co_eff, intercept, min(x[:,1]), max(x[:,1]))

plt.show()
```



Observation

- While the regularization value is small the keeps fails to classify the data points despite of the dataset imbalance. (Column-1 is an example).
- The LogisticRegression fails to classify data points in an imbalance dataset with lower regularization values (Row-1)
- As increasing regularization parameter even with highly imbalance dataset, LR starts to try to classify better and better (Row-2)
- With increasing regularization parameter and reduced imbalanced data ratio model delivers better classification (Row-3).
- When the dataset imbalance ratio reduced and in increasing regularization values the model classifies the points better and better (Row-4).
- In Row-4 its clear, with higher regularization value the decision boundary changed inorder to reduce the classification error.