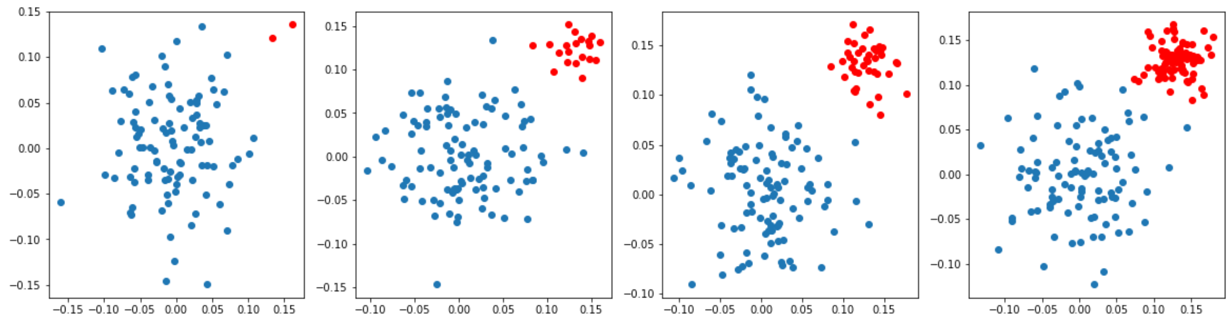
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, Normalizer
import matplotlib.pyplot as plt
from sklearn.svm 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 in
    # 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
    # 2. ((b*max-c)/a, max) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here
    points=np.array([[((-coef[1]*mi - intercept)/coef[0]), mi],[((-coef[1]*ma - i
    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 a
nd 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
```

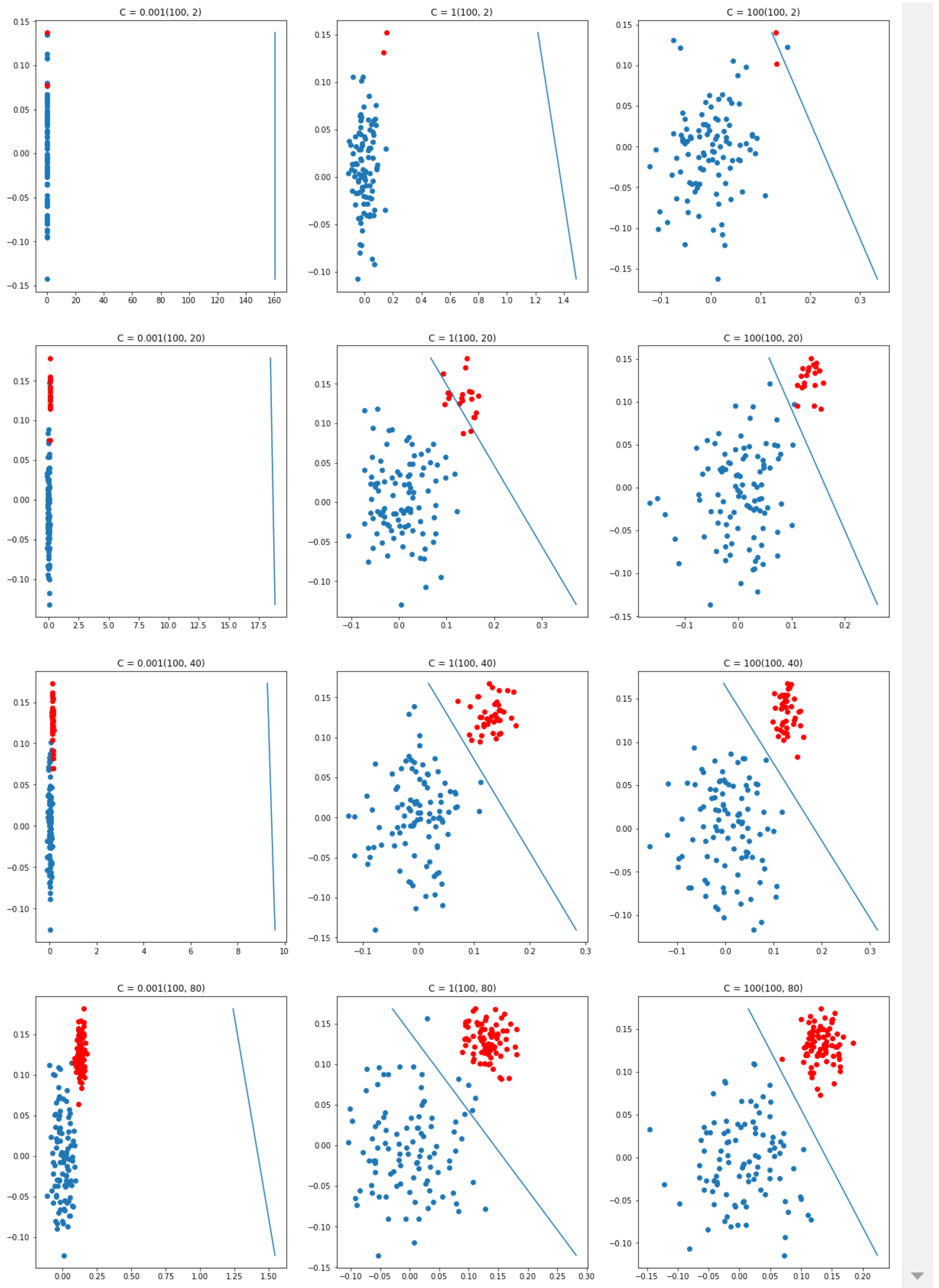
In [3]:
```
# here we are creating 2d imbalanced data points
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
plt.figure(figsize=(20,5))
for j,i in enumerate(ratios):
    plt.subplot(1, 4, j+1)
    X_p=np.random.normal(0,0.05,size=(i[0],2))
    X_n=np.random.normal(0.13,0.02,size=(i[1],2))
    y_p=np.array([1]*i[0]).reshape(-1,1)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

In [4]:
```
from sklearn.svm import LinearSVC, SVC
c = [0.001,1,100]
plt.figure(figsize = (20,30))
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
num=1
for j,i in enumerate(ratios):
 for k in range(0, 3):
   model=LinearSVC(C=c[k])
   plt.subplot(4, 3, num)
   num=num+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)
   plt.scatter(X_p[:,0],X_p[:,1])
   plt.scatter(X_n[:,0],X_n[:,1],color="red")
   plt.title("C = "+ str(c[k])+str(i))
   draw_line(coef=model.coef_[0],intercept=model.intercept_,ma=max(X[:,1]), mi= m
plt.show()
```



Observation

For c=0.001

1. 100:2 --> The dataset is underfitted.

2. 100:20 --> The dataset is more underfitted than previous dataset(100:2)
3. 100:40 --> The dataset is more underfitted than the previous dataset.
4. 100:80 --> The dataset is most underfitted.

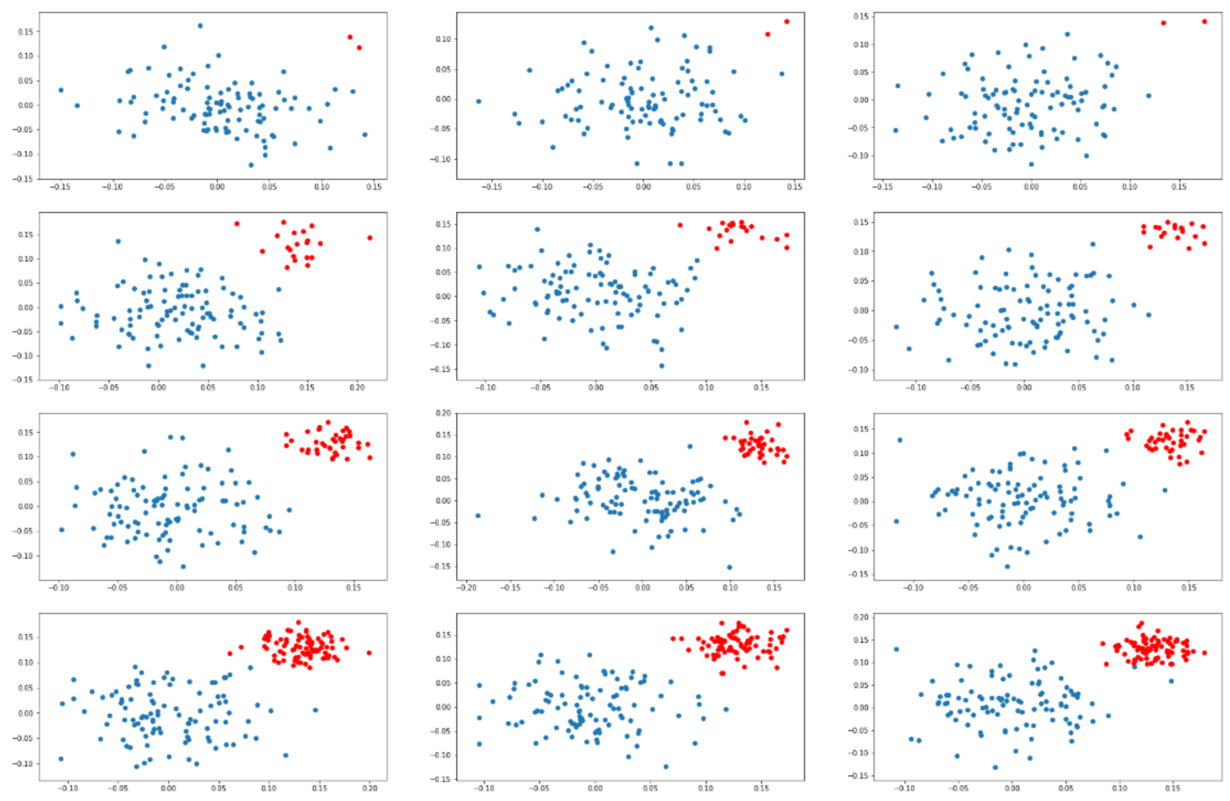
For c=1

1. 100:2 --> The dataset is underfitted.
2. 100:20 --> The dataset is less underfitted than previous dataset(100:2)
3. 100:40 --> The dataset is overfitted on the dataset.
4. 100:80 --> The dataset is fitted well.

For c=100

1. 100:2 --> The dataset is slightly underfitted but is better than the previous value of c.
2. 100:20 --> The dataset is fitted well.
3. 100:40 --> The dataset is overfitted on the dataset.
4. 100:80 --> The dataset is overfitted on the dataset.

```
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 ge
t after applying SVM (https://scikit-learn.org/stable/modules/generated/s
klearn.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))

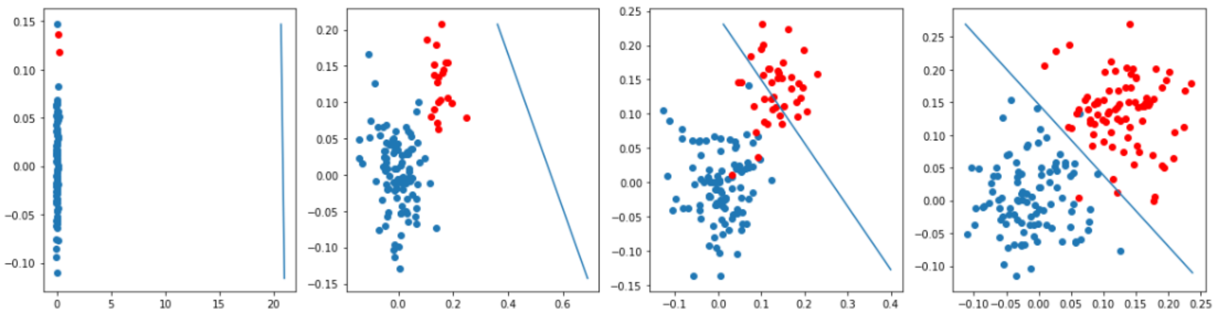
```
if you can do, you can represent the support vectors in different colors,
which will help us understand the position of hyper plane
```

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

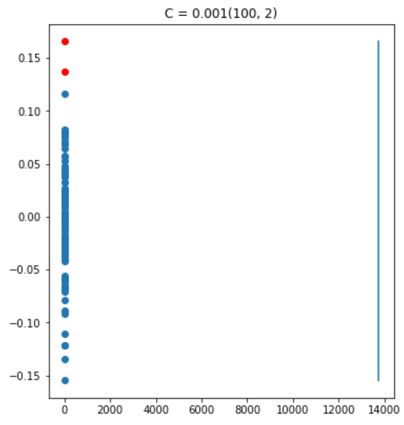
```
check the optimization problem here https://scikit-learn.org/stable/modul
es/svm.html#mathematical-formulation

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

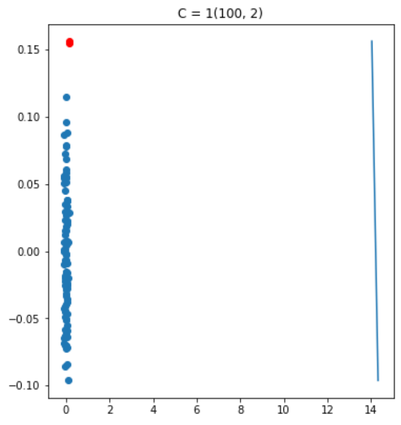
Task 2: Applying LR

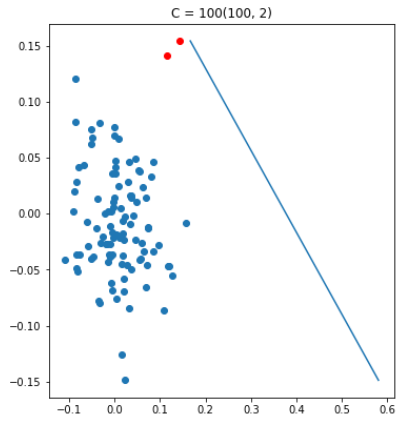
```
 you will do the same thing what you have done in task 1.1, except instea
d of SVM you apply logistic regression  (https://scikit-learn.org/stable/
modules/generated/sklearn.linear_model.LogisticRegression.html)



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

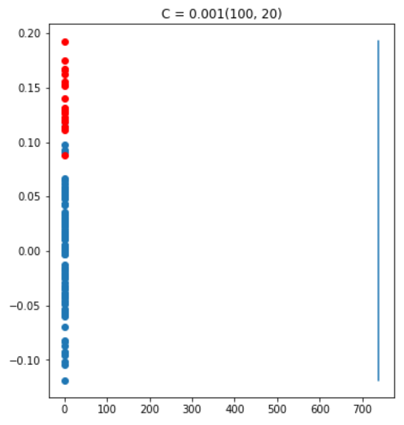

0.15
0.10
0.05
0.00
−0.05
−0.10
0
5
10
15
20

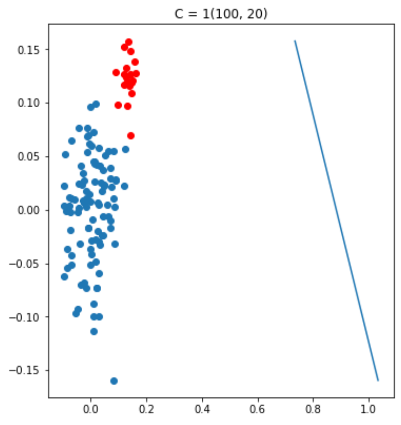
In [5]:
```
from sklearn.svm import LinearSVC, SVC
from sklearn.linear_model import LogisticRegression
c = [0.001,1,100]
plt.figure(figsize = (20,30))
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
num=1
for j,i in enumerate(ratios):
 for k in range(0, 3):
   model=LogisticRegression(C=c[k])
   plt.subplot(4, 3, num)
   num=num+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)
   plt.scatter(X_p[:,0],X_p[:,1])
   plt.scatter(X_n[:,0],X_n[:,1],color="red")
   plt.title("C = "+ str(c[k])+str(i))
   draw_line(coef=model.coef_[0],intercept=model.intercept_,ma=max(X[:,1]), mi= m
plt.show()
```

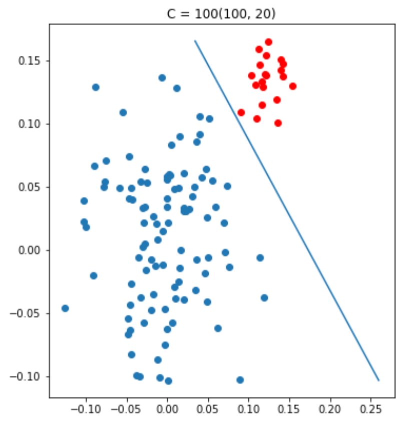

C = 0.001(100, 2)

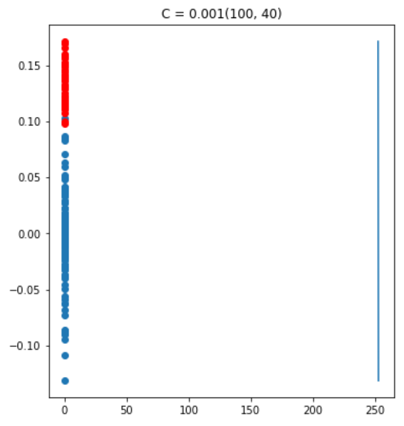

C = 1(100, 2)

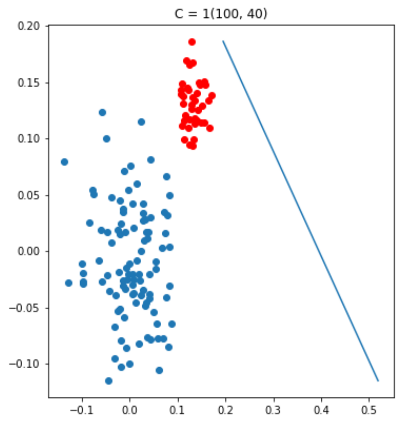

C = 100(100, 2)

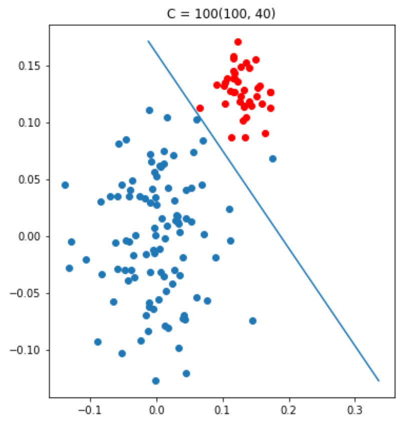

C = 0.001(100, 20)

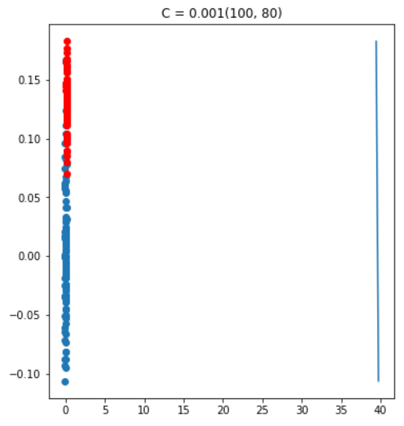

C = 1(100, 20)

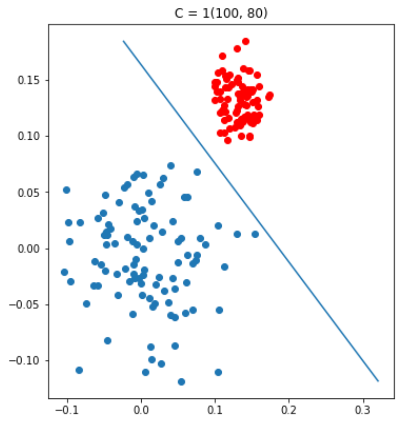

C = 100(100, 20)

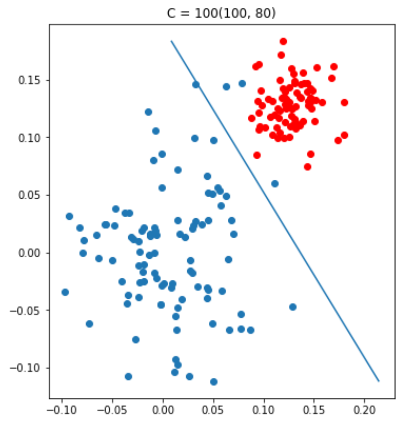

C = 0.001(100, 40)


C = 1(100, 40)


C = 100(100, 40)


C = 0.001(100, 80)


C = 1(100, 80)


C = 100(100, 80)

Observation

C=0.001

1. The data set is underfitted.
2. The dataset is more underfitted tha the previous one.
3. The dataset is more underfitted than the previous one.
4. The dataset is highly underfitted.

C=1

1. The data set is underfitted.
2. The dataset is more underfitted tha the previous one.
3. The dataset is more underfitted than the previous one.
4. The dataset is fitted well.

C=100

1. The data set is underfitted.
2. The dataset is overfitted.
3. The dataset is fitted well.
4. The dataset is more fitted well.