```
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")
```

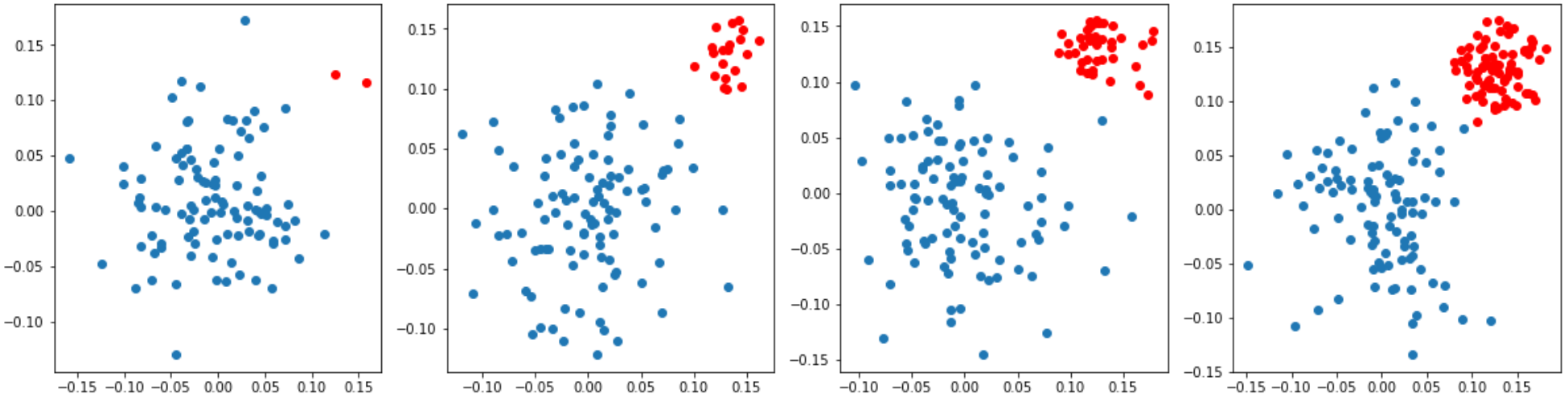
```
def draw_line(coef,intercept, mi, ma):
    # for the separating hyper plane ax+by+c=0, the weights are [a, b] and the intercept is c
    # to draw the hyper plane we are creating two points
    # 1. ((b*min-c)/a, min) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here in place of y we are keeping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

```
# here we are creating 2d imbalanced data points
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
```

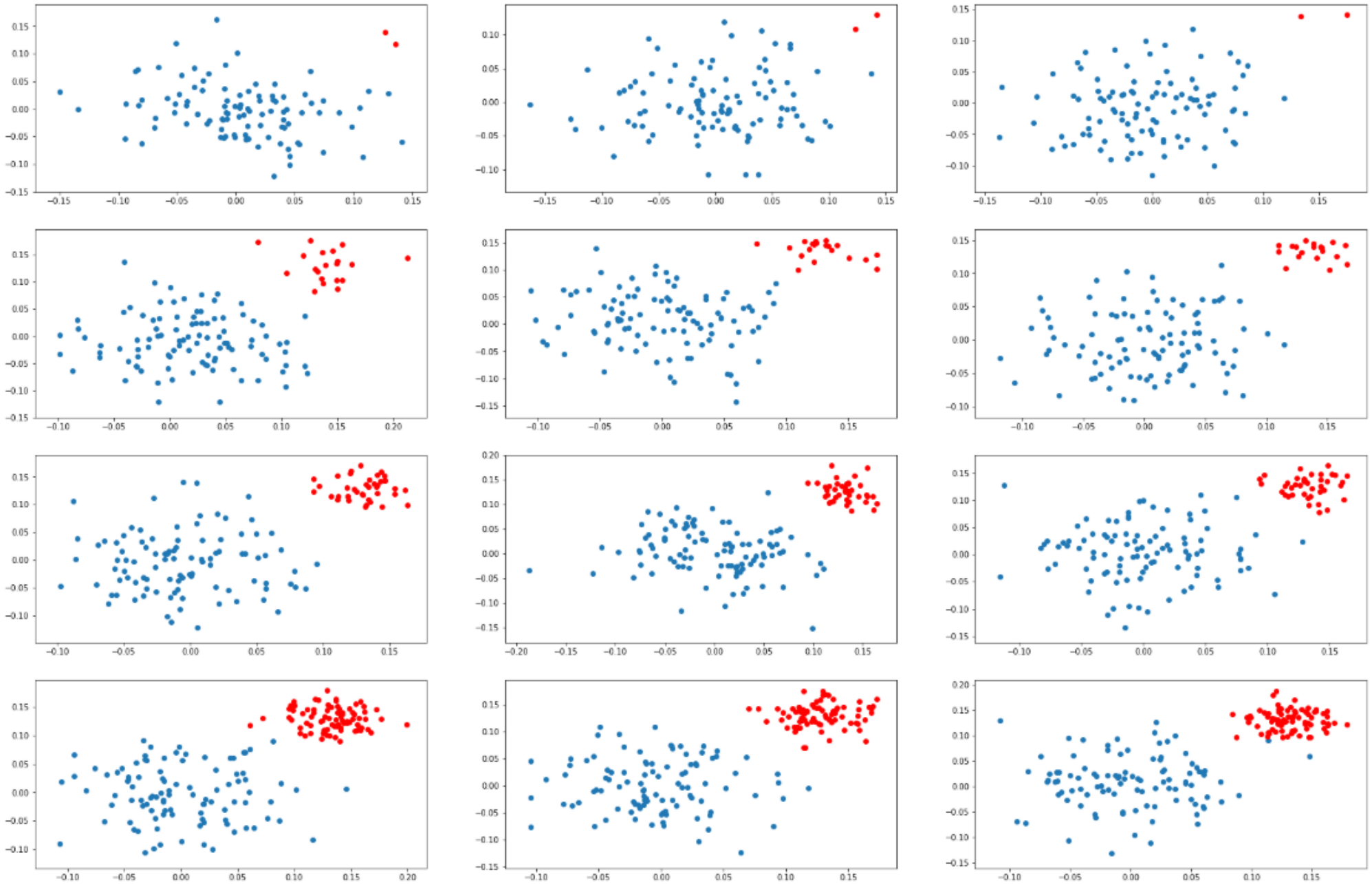
```
plt.figure(figsize=(20,5))
for j,i in enumerate(ratios):
    plt.subplot(1, 4, j+1)
    X_p=np.random.normal(0,0.05,size=(i[0],2))
    X_n=np.random.normal(0.13,0.02,size=(i[1],2))
    y_p=np.array([1]*i[0]).reshape(-1,1)
    y_n=np.array([0]*i[1]).reshape(-1,1)
    X=np.vstack((X_p,X_n))
    y=np.vstack((y_p,y_n))
    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))

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

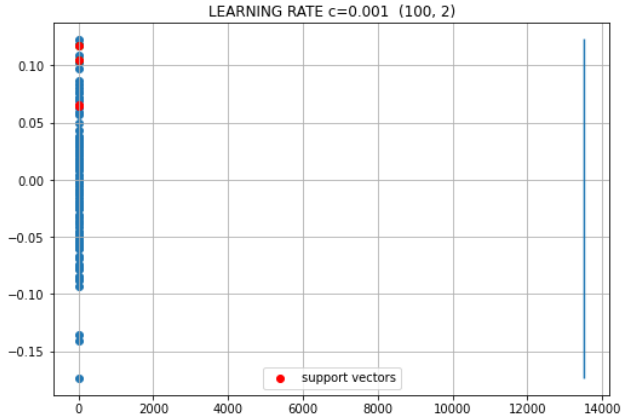
check the optimization problem here https://scikit-learn.org/stable/modules/svm.html#mathematical-formulation

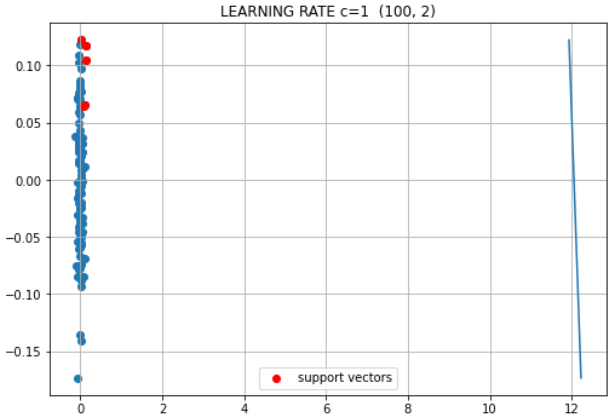
if you can describe your understanding by writing it on a paper
and attach the picture, or record a video upload it in assignment.

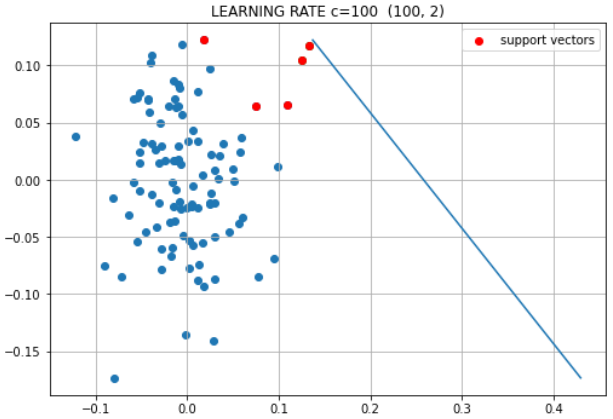
Applying support vector machine

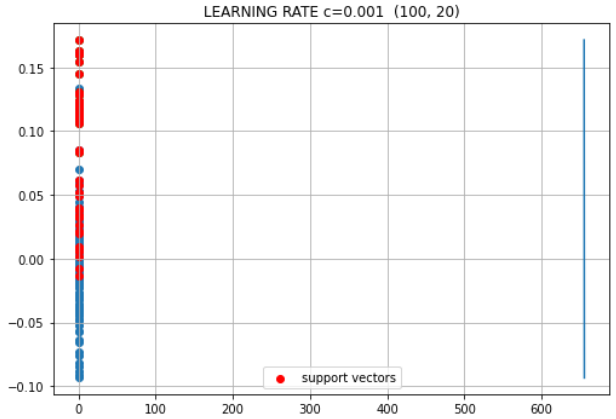
```
np.random.seed(15)
K=0

# dataset size
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]

# LEARNING RATE
rate= [0.001, 1, 100]

# PLOT SIZE
plt.figure(figsize=(28,26))

# NOW ENUMERATE THE BOTH J AND I
for j,i in enumerate(ratios):
    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))

    #  MAKING SAME DIMESNION
    y_p=np.array([1]*i[0]).reshape(-1,1)
    y_n=np.array([0]*i[1]).reshape(-1,1)

    # STACKING
    X=np.vstack((X_p,X_n))
    y=np.vstack((y_p,y_n))
    for k in range(3):
      # INCREAMENTING BY 1
        K=K+1
        plt.subplot(4,3,K)
        plt.title("LEARNING RATE c="+str(rate[k])+"  "+str(i))

        # PLOTTING GRID OF PLOT
        plt.grid()


        # SCATTER PLOT TO SHOW DATAPOINT
        plt.scatter(X_p[:,0],X_p[:,1])
        plt.scatter(X_n[:,0],X_n[:,1],color='green')
```

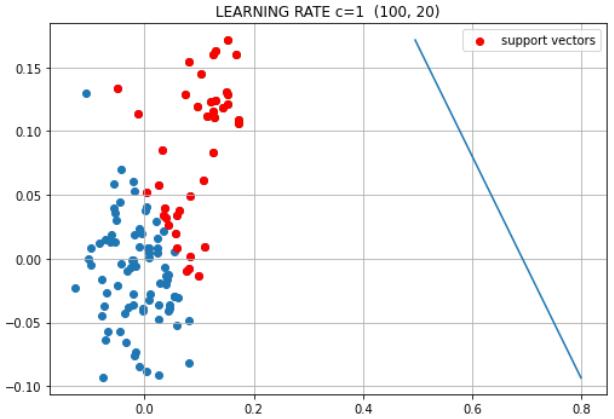
```
    # USING LINEAR SVM WITH RANDOM STATE 15
    clf=SVC(kernel="linear",C=rate[k],random_state=15)
    clf.fit(X,y)  # GETTING THE INTERCEPT AND WEIGHT COEFFICIENT

    # adding the coefficient to data
    weight=clf.coef_
    intercept=clf.intercept_

    sv=clf.support_vectors_

    plt.scatter(sv[:,0],sv[:,1],color="red",label='support vectors')

    # plotting the legend
    plt.legend()
    mi=min(X[:,1])
    mx=max(X[:,1])

    # draw the line with intercept
    draw_line(weight[0],intercept,mi,mx)
```

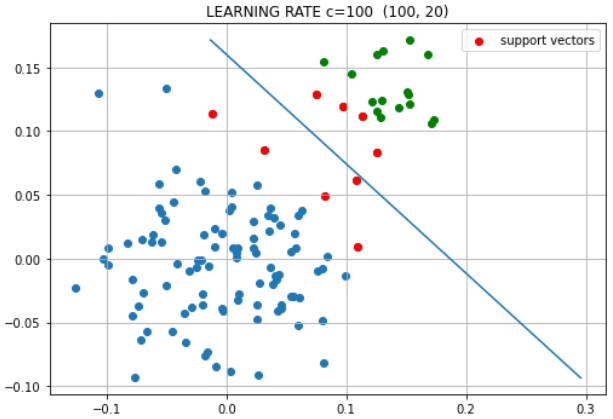

LEARNING RATE c=0.001 (100, 2)
support vectors

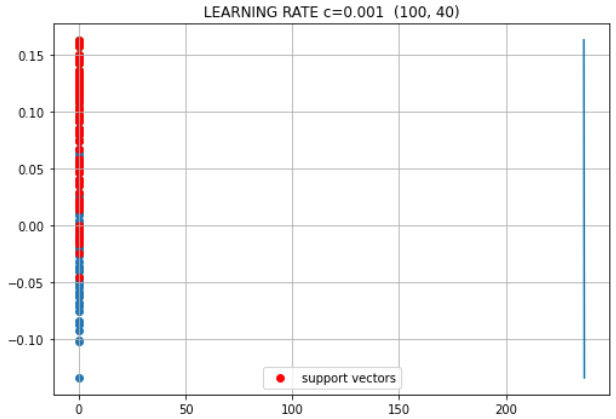

LEARNING RATE c=1 (100, 2)
support vectors

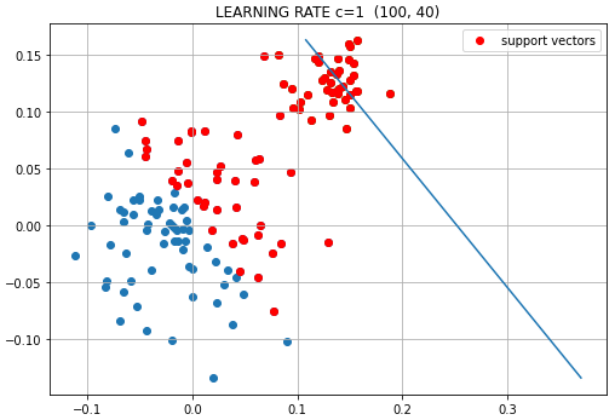

LEARNING RATE c=100 (100, 2)
support vectors

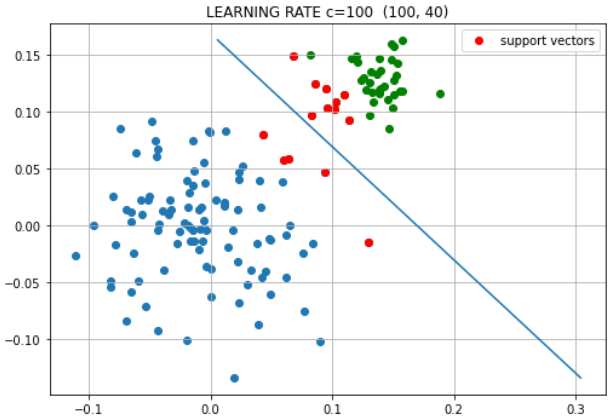

LEARNING RATE c=0.001 (100, 20)
support vectors

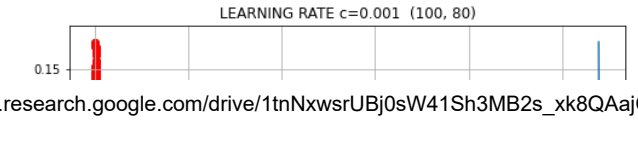

LEARNING RATE c=1 (100, 20)
support vectors

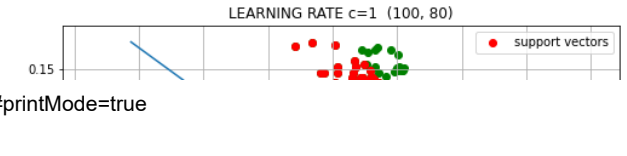

LEARNING RATE c=100 (100, 20)
support vectors

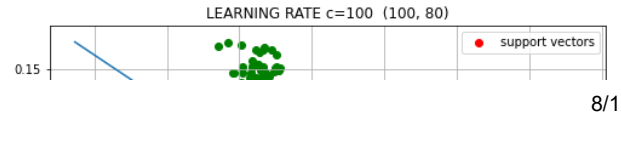

LEARNING RATE c=0.001 (100, 40)
support vectors

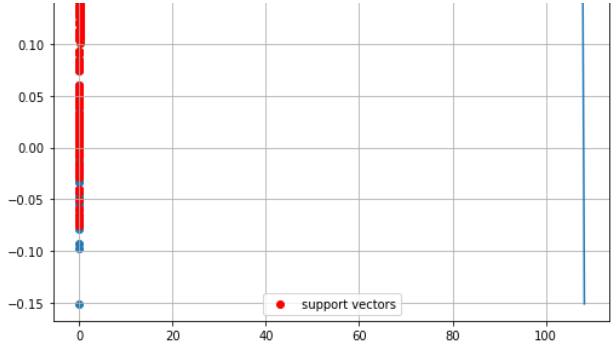

LEARNING RATE c=1 (100, 40)
support vectors

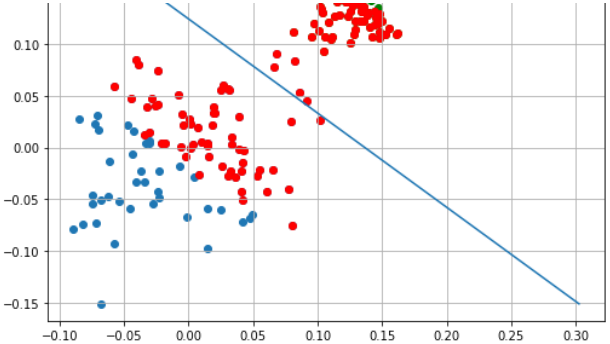

LEARNING RATE c=100 (100, 40)
support vectors

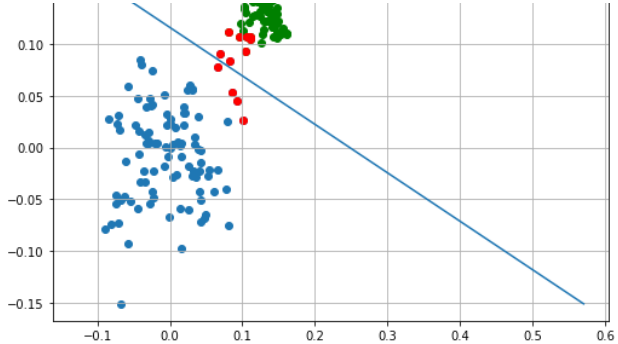

LEARNING RATE c=0.001 (100, 80)


LEARNING RATE c=1 (100, 80)
support vectors


LEARNING RATE c=100 (100, 80)
support vectors



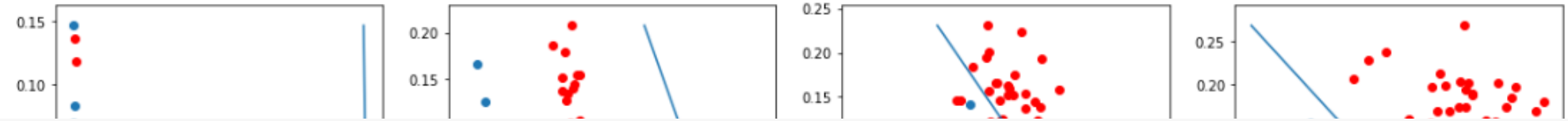




Task 2: Applying LR

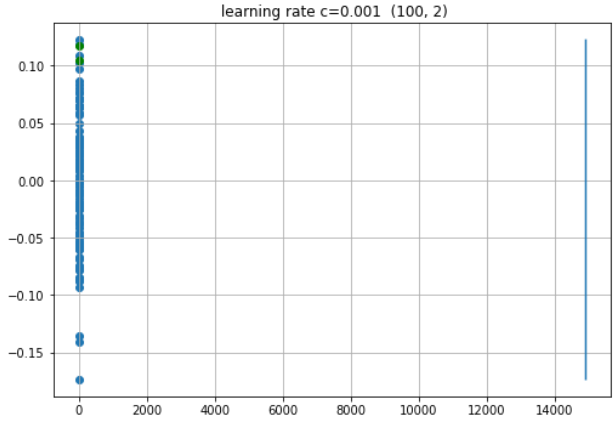
you will do the same thing what you have done in task 1.1, except instead of SVM you apply logistic regression

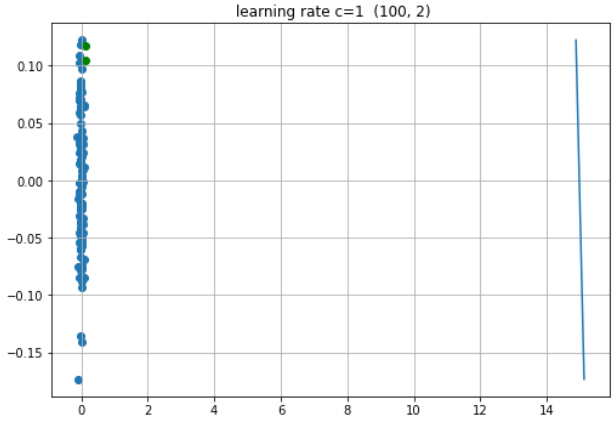
these are results we got when we are experimenting with one of the model

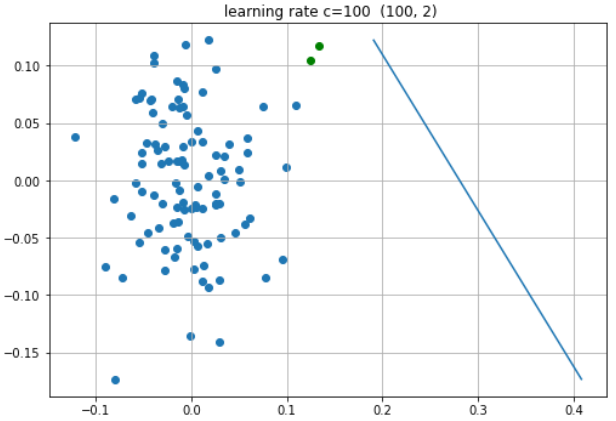


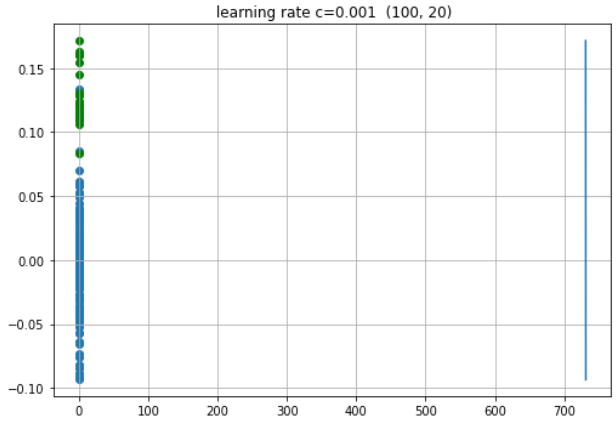

```
np.random.seed(15)
K=0
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]

# learning rate c
rate= [0.001, 1, 100]

# plot size
plt.figure(figsize=(28,26))
for j,i in enumerate(ratios):
    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))

    # make the datapoint same dimension
    y_p=np.array([1]*i[0]).reshape(-1,1)
    y_n=np.array([0]*i[1]).reshape(-1,1)

    # will stck the both fatures
    X=np.vstack((X_p,X_n))
    y=np.vstack((y_p,y_n))
    for k in range(3):

      # incrementing by 1
        K=K+1
        plt.subplot(4,3,K)
        plt.title(" learning rate c="+str(rate[k])+"  "+str(i))

        # plotting the grid of plot
        plt.grid()

        #plotting the scatter plot
        plt.scatter(X_p[:,0],X_p[:,1])
```

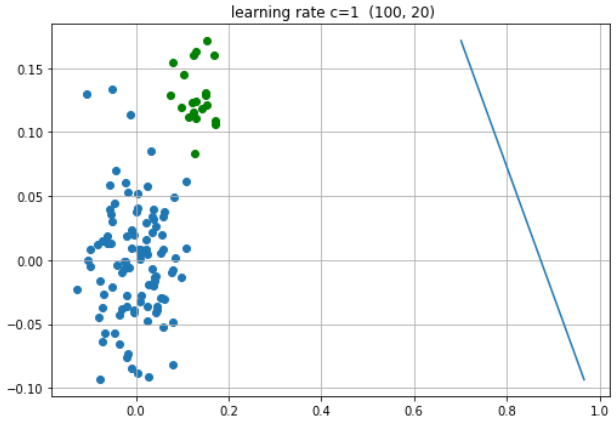
```
        plt.scatter(X_n[:,0],X_n[:,1],color='green')

        # now will use logestic regression model with random state of 15
        clf = LogisticRegression(C=rate[k],random_state=15)

        # fitting the dataset
        clf.fit(X,y)

        # adding the coeficient into classification model
        weight=clf.coef_
        intercept=clf.intercept_
        mi=min(X[:,1])
        mx=max(X[:,1])

        # draing the line
        draw_line(weight[0],intercept,mi,mx)
```

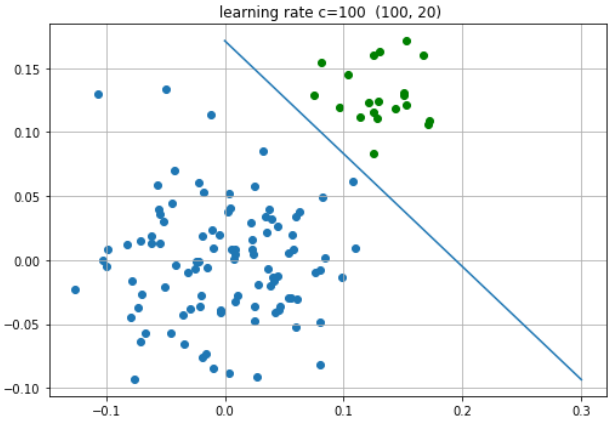

learning rate c=0.001 (100, 2)

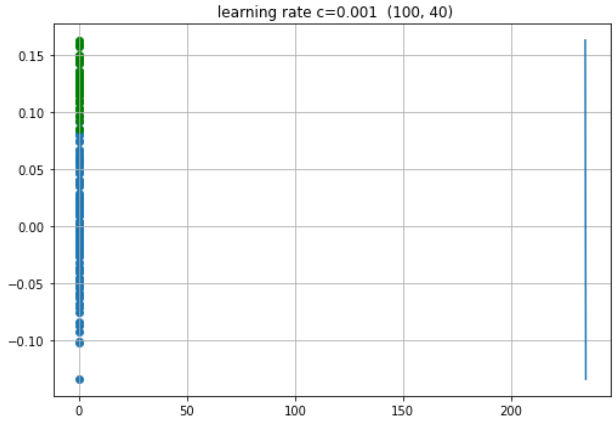

learning rate c=1 (100, 2)

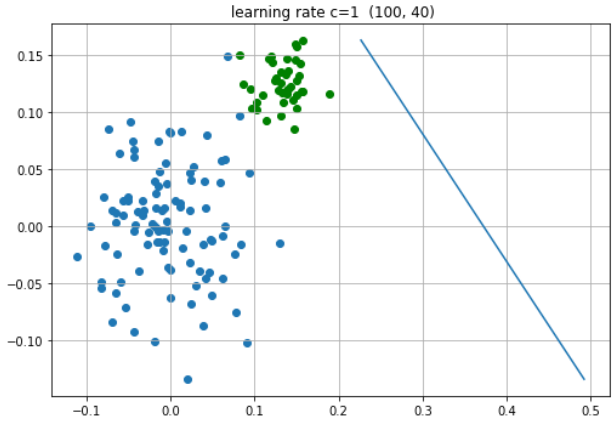

learning rate c=100 (100, 2)

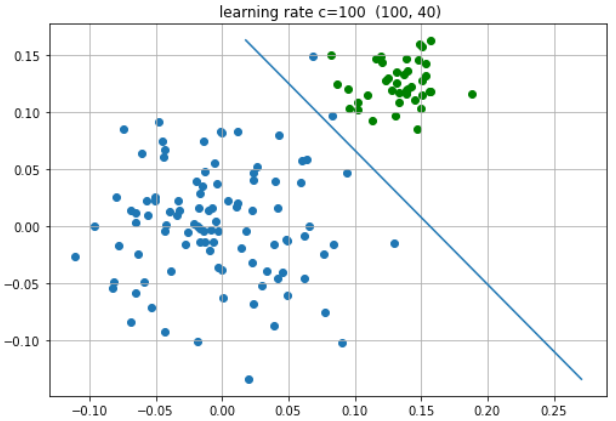

learning rate c=0.001 (100, 20)

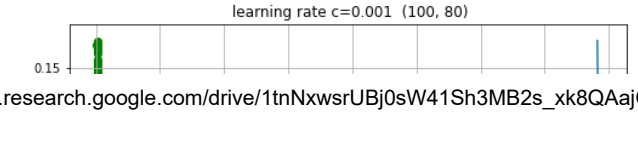

learning rate c=1 (100, 20)

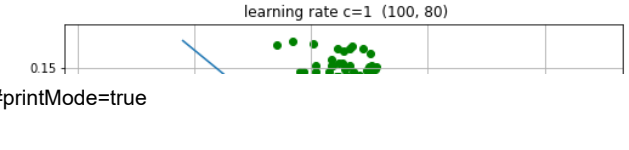

learning rate c=100 (100, 20)

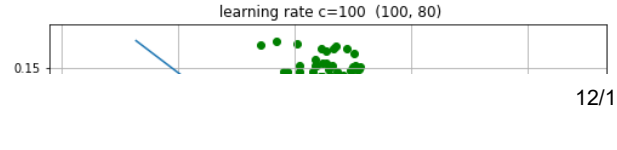

learning rate c=0.001 (100, 40)

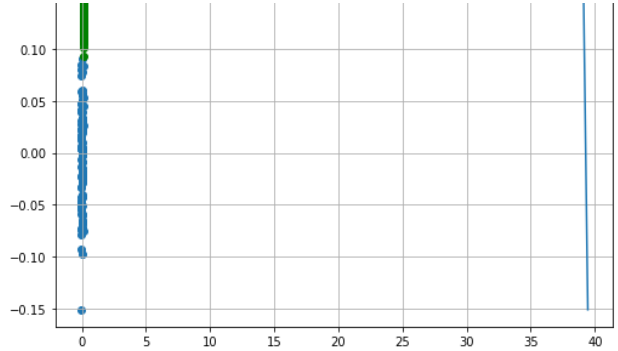

learning rate c=1 (100, 40)

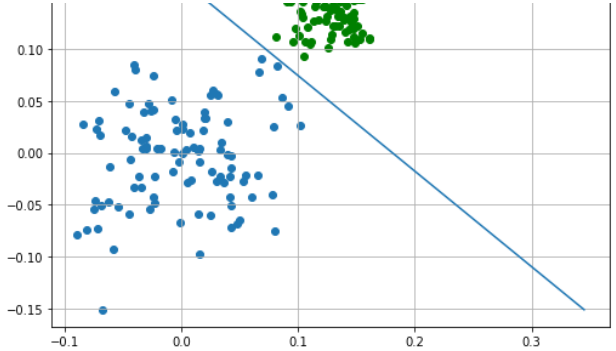

learning rate c=100 (100, 40)

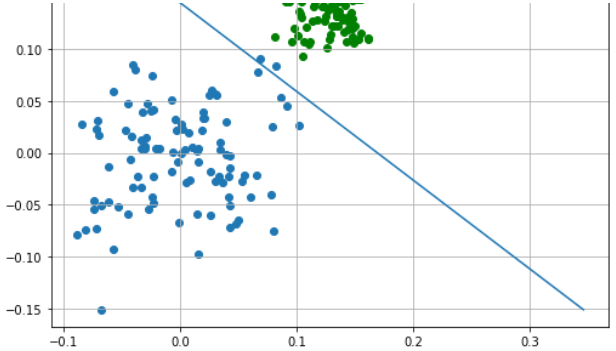

learning rate c=0.001 (100, 80)

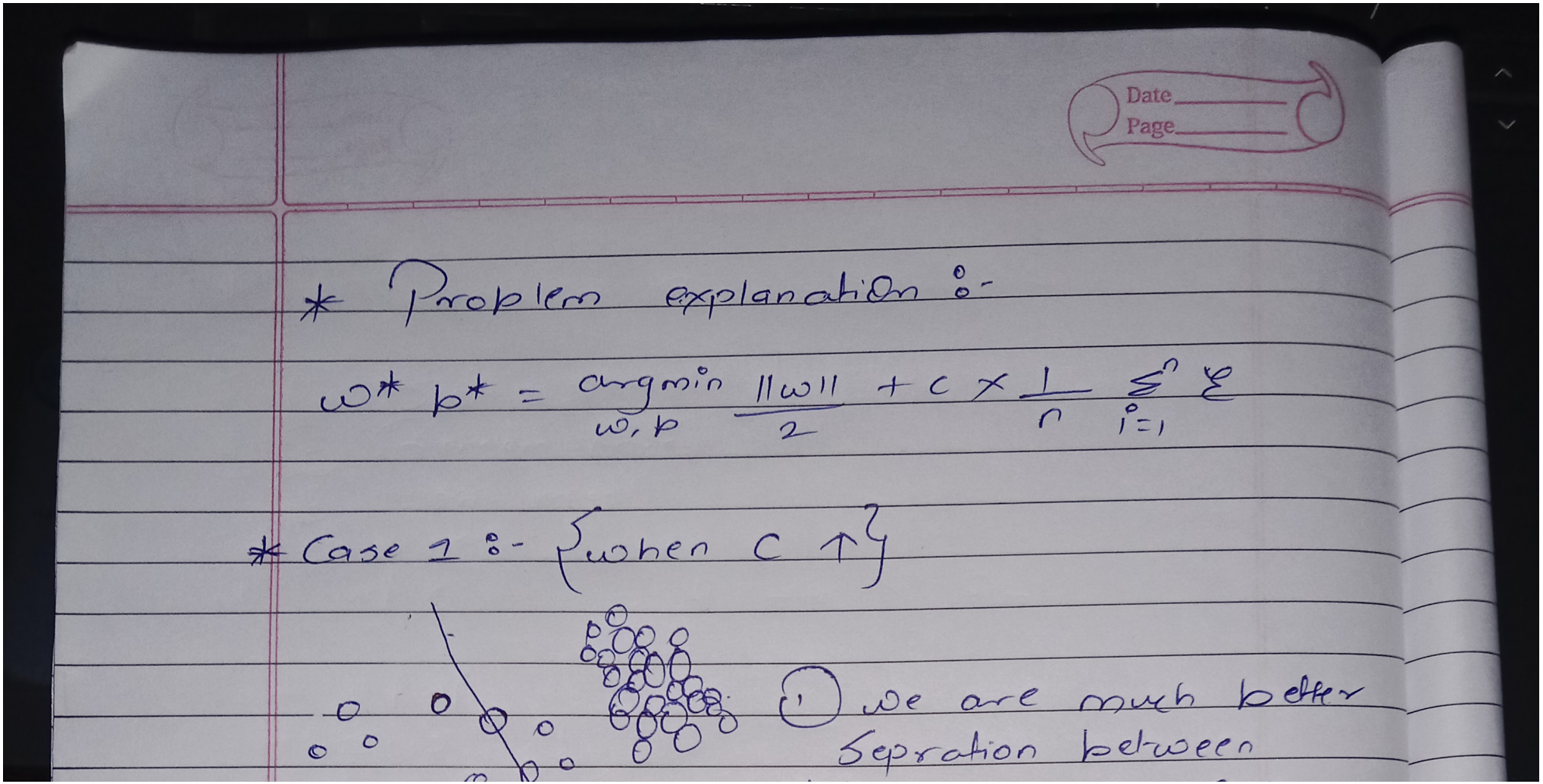

learning rate c=1 (100, 80)

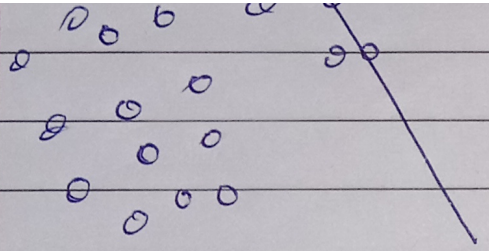

learning rate c=100 (100, 80)


0.10
0.05
0.00
−0.05
−0.10
−0.15
0
5
10
15
20
25
30
35
40


0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.1
0.0
0.1
0.2
0.3


0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.1
0.0
0.1
0.2
0.3


Date
Page
* Problem explanation :-
$w^* b^* = \underset{w,b}{\operatorname{argmin}} \frac{\|w\|}{2} + C \times \frac{1}{n} \sum_{i=1}^{n} \xi$
* Case 1 :- {when C ↑}
(1) we are much better
Sepration between



datapoints when C ↑

(2) But still data not getting seperating well

(3) dataset Highly imbalanced and overlaping so its hard to find best Hyperplane.

* Case 2°:- (when C ↓)

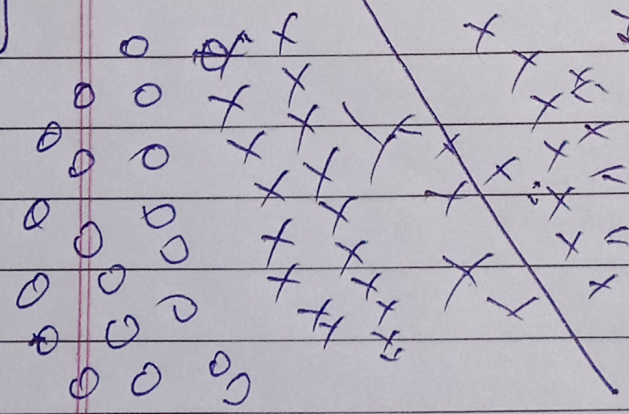
* oBservation :-

(1) when C is 1 Some overlap bet datapts.

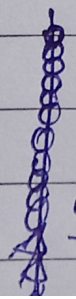
(2) still not getting best hyperplane.

(3) Hyperplane can not seprate datapts in linear Svm when data is highly imbalanced.

Plot 1]

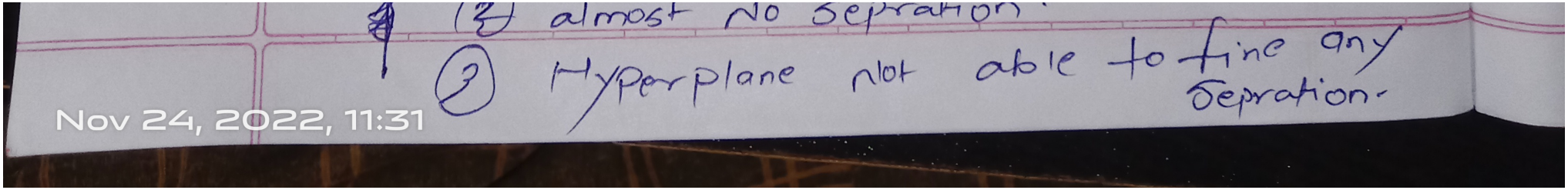


plot 2].



when C = 0.001

oBservation :- (1) Data is Highly overlapped.



OBSERVATION

1)DATASET IS HIGHLY IMBALANCED AND AND IVERLAPPED

2).IN LINEAR SVM AND LOGESTIC REGRESSION BOTH ARE NOT PERFORMING WELL WHEN DATASET IS HIGHLY IMBALANCD

3).WHEN WE HAVE STARTED MAKE DATA LITTLE MORE BALANCED AND INCREASING THE VALUE OF C THEN WE CAN SEE SOME SENSIBLE HYPERPLANCE TO SEPERATE HYPERPLANE

4).I THINK ITS WE CAN DO MORE EMPARIMENT WITH MAKE THE DATA MORE BALANCED OR WE CAN PERFORM RBF KERNAL WITH SVM SO MAY BE BETTER SEPRATION BETWEEN DATA

5).ONE MORE BIG INSIGHTS WHEN WE INCREASES THE VALUE OF C THEN PLOT LOOK MORE SENSIBLE WHEN C VALUE LOW THE PLOTS MAKE NO SENSE