In [ ]:

```
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 [ ]:

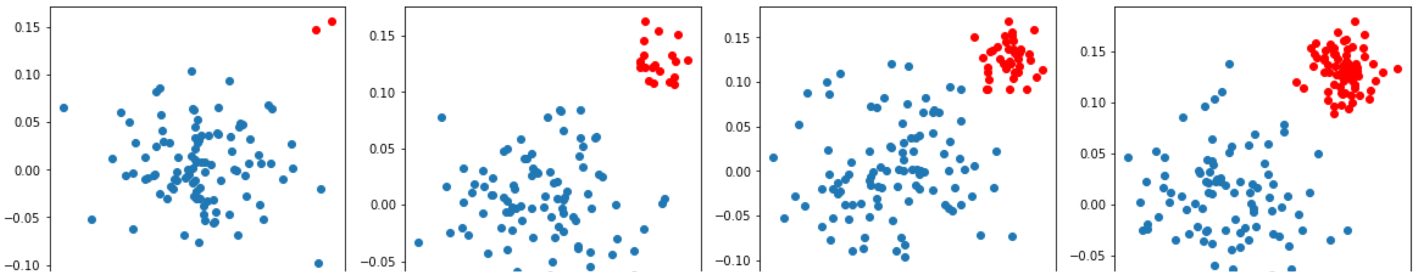
```
def draw_line(coef,intercept, mi, ma):
    # for the separating hyper plane ax+by+c=0, the weights are [a, b] and the intercept
is c
    # to draw the hyper plane we are creating two points
    # 1. ((b*min-c)/a, min) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here in plac
e of y we are keeping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

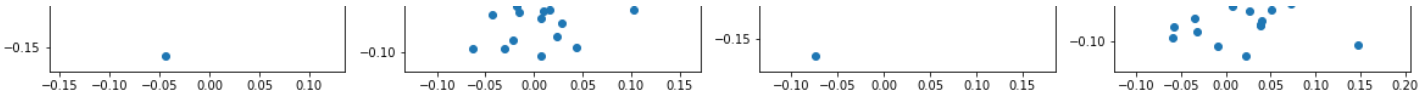
What if Data is imabalanced

```
1. As a part of this task you will observe how linear models work in case of data i
mbalanced
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 [ ]:

```
# 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()
```

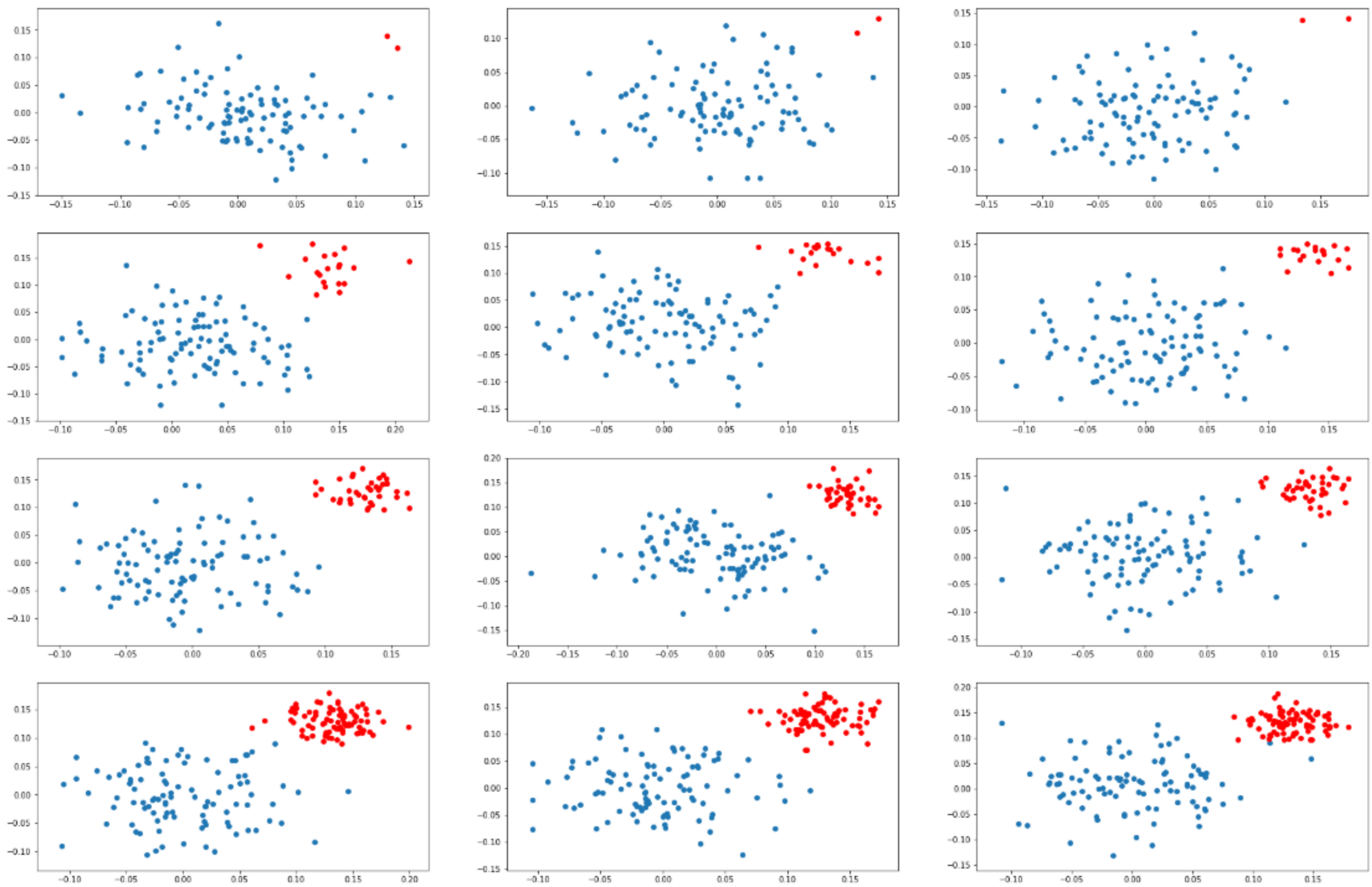




> **your task is to apply SVM ([sklearn.svm.SVC](#)) and LR ([sklearn.linear_model.LogisticRegression](#)) with different regularization strength [0.001, 1, 100]**

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 ap
plying 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, an

d
what do you think about the position of the hyper plane

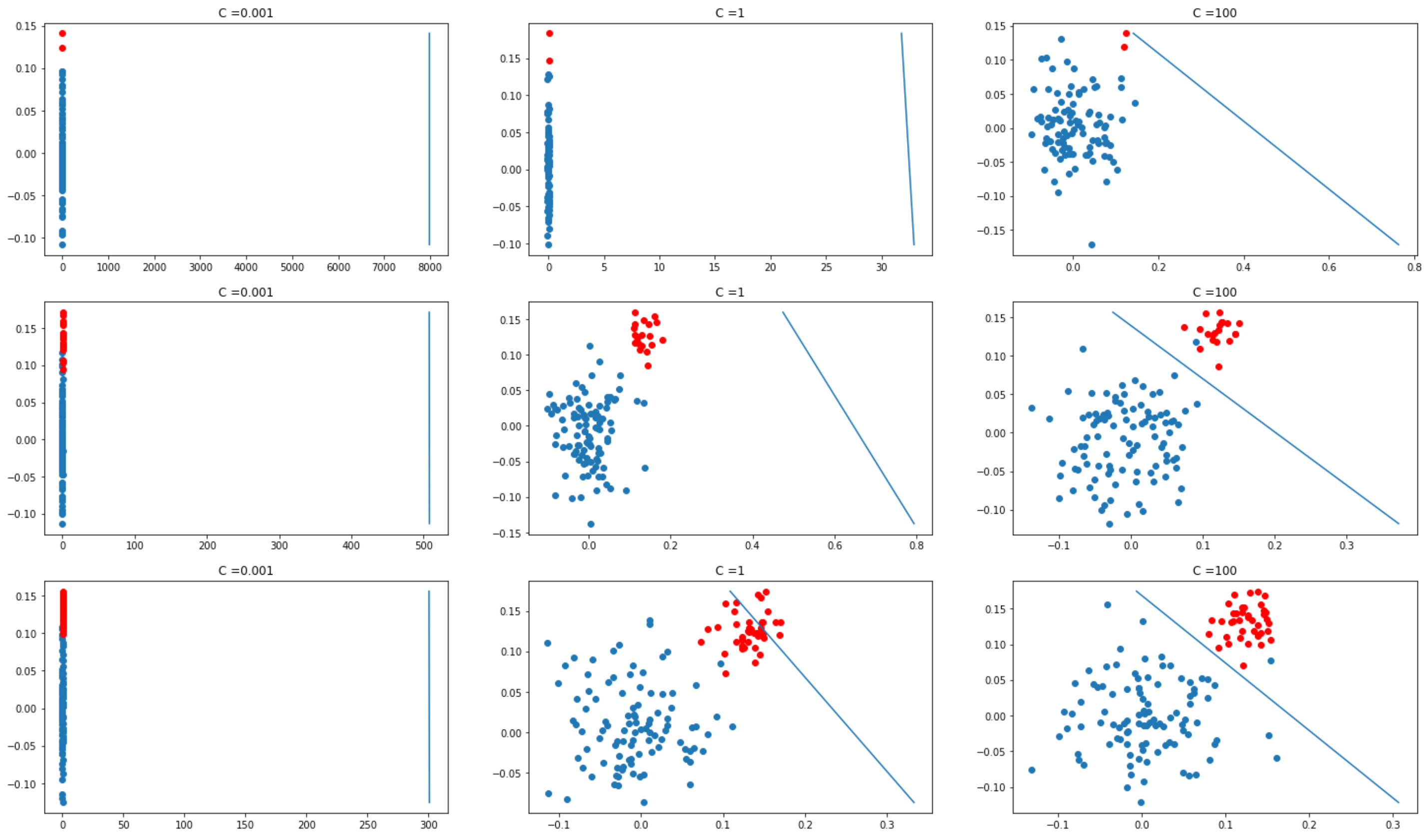
check the optimization problem here https://scikit-learn.org/stable/modules/svm.html#mathematical-formulation

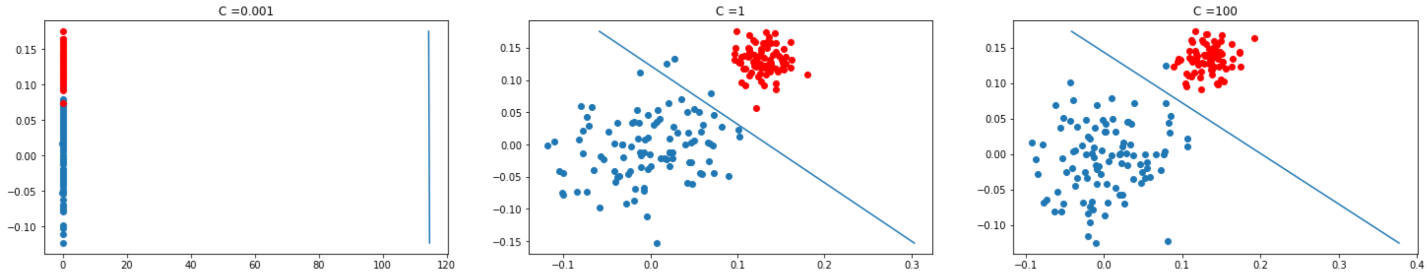
if you can describe your understanding by writing it on a paper
and attach the picture, or record a video upload it in assignment.

In [ ]:

```
from sklearn import svm
import sklearn
from sklearn.svm import SVC
plt.figure(figsize = (25,20))
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
C=[0.001, 1, 100]
k=0
for j,i in enumerate(ratios):
  for c in C:
    plt.subplot(4, 3, k+1)
    k=k+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))

    clf=sklearn.svm.SVC(C=c, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
    clf.fit(X,Y)
    #print('w = ',clf.coef_)
    #print('b = ',clf.intercept_)
    plt.title('C ='+ str(c))
    draw_line(coef=clf.coef_,intercept=clf.intercept_,ma=max(X[:,1]), mi=min(X[:,1]))
    plt.scatter(X_p[:,0],X_p[:,1])
    plt.scatter(X_n[:,0],X_n[:,1],color='red')
plt.show()
```





Observation

1.when C=0.001 , the seperating plane did not do well in seperating positive and negative points ,even when the data set is balanced.

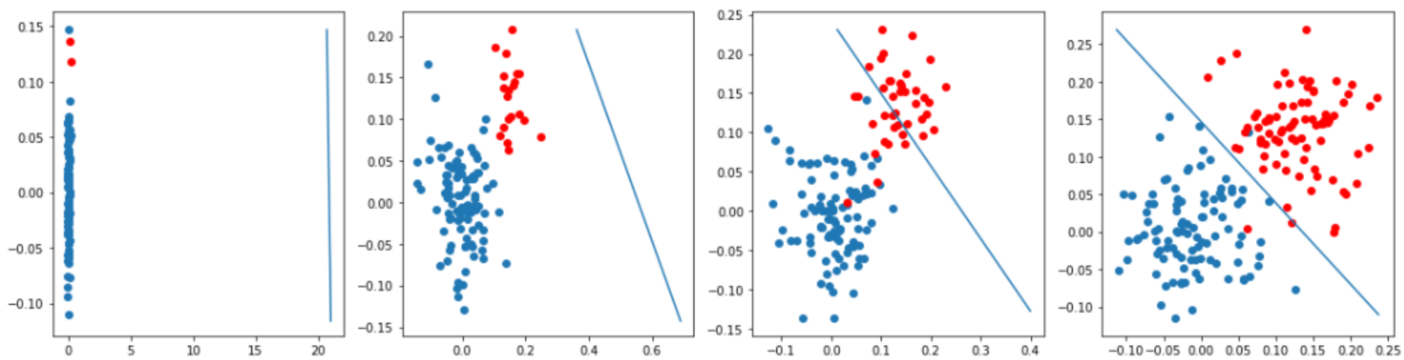
2.when C=1, seperator plane separates data much better when data in balanced ,and didnt do well when we have imbalances dataset.

3.when c=100, its the best seperator plane ,works well even if the data is not balanced, and works like magic with balanced data set

Task 2: Applying LR

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

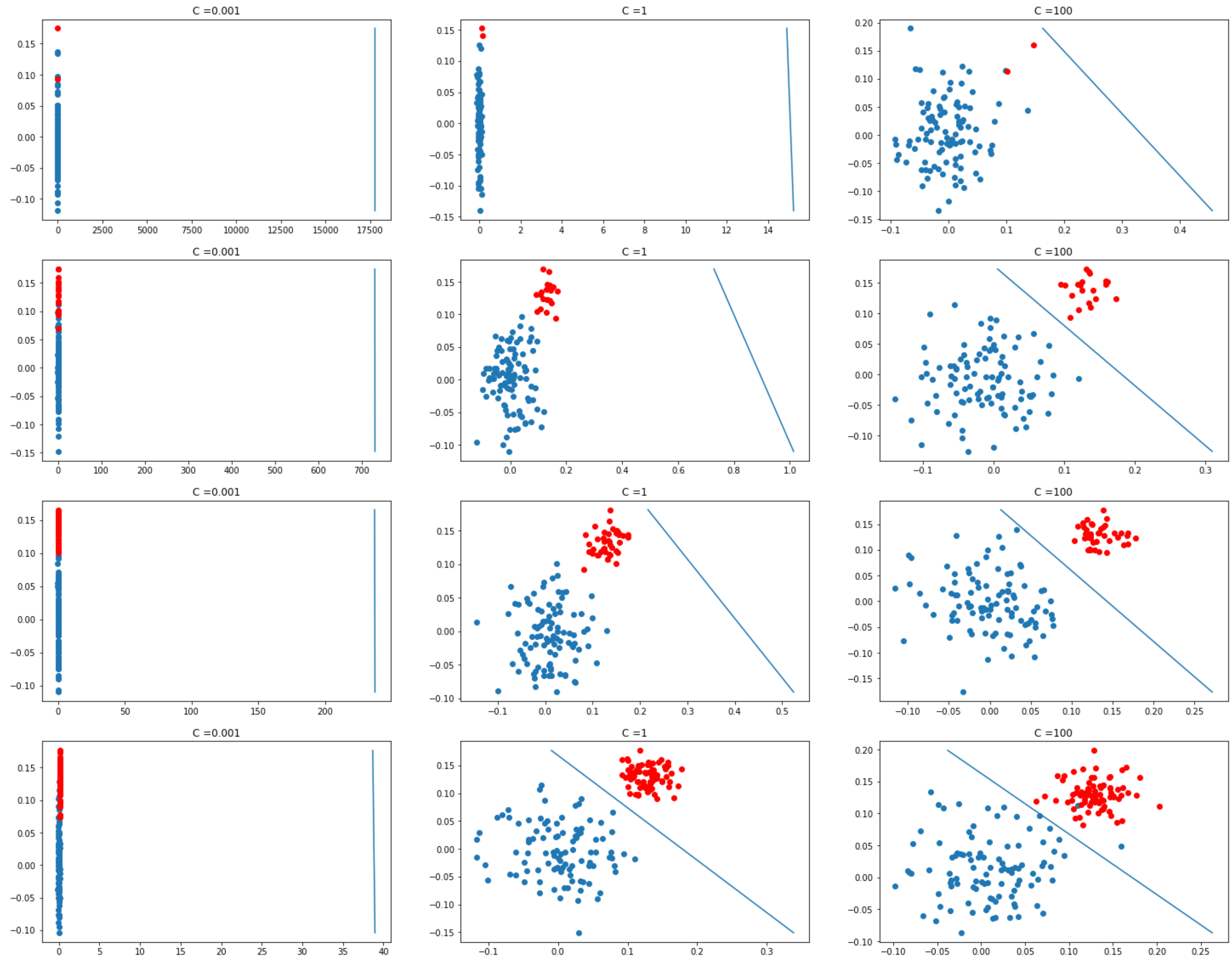
```
these are results we got when we are experimenting with one of the model
```



In [ ]:

```
#you can start writing code here.
import sklearn
from sklearn.linear_model import LogisticRegression
plt.figure(figsize = (25,20))
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
C=[0.001, 1, 100]
k=0
for j,i in enumerate(ratios):
  for c in C:
    plt.subplot(4, 3, k+1)
    k=k+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))
    clf=LogisticRegression (penalty='l2',dual=False,tol=0.0001,C=c,fit_intercept=True,int
ercept_scaling='l2',
             class_weight=None,random_state=None,solver='lbfgs',max_iter=100,
             multi_class='auto',verbose=0,warm_start=False,n_jobs=None,l1_ratio=None)
    #print('w = ',clf.coef_)
    #print('b = ',clf.intercept_)
    plt.title('C ='+ str(c))
    draw_line(coef=clf.coef_,intercept=clf.intercept_,ma=max(X[:,1]), mi=min(X[:,1]))
    plt.scatter(X_p[:,0],X_p[:,1])
    plt.scatter(X_n[:,0],X_n[:,1],color='red')
plt.show()
```



OBSERVATION

1.when C=0.001, model does not perform well , not able to seperate data points using seperator plane eventhough dataset is

2.when C=1,only performs well when dataset is balanced fully.

3.when C=100 , works well even when data is slightly imbalanced. and works very well with balanced data