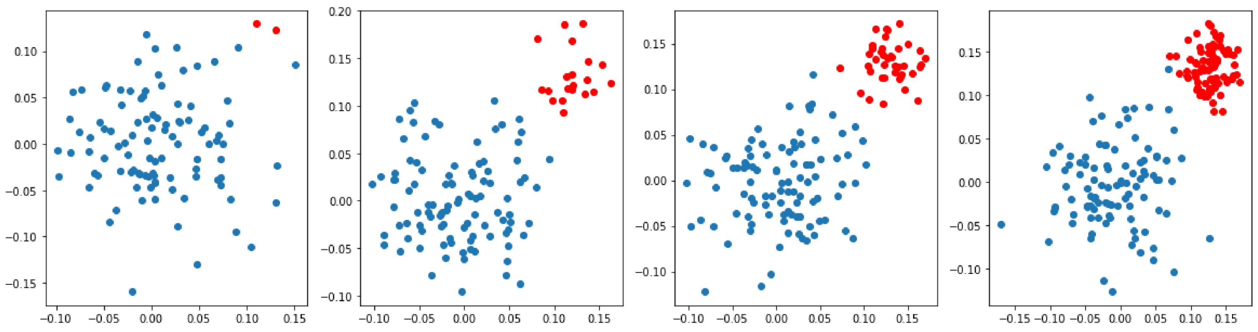
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, Normalizer
import matplotlib.pyplot as plt
from sklearn.svm 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 i
    # 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
    # 2. ((b*max-c)/a, max) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here in place
    points=np.array([[((-coef[1]*mi - intercept)/coef[0]), mi],[((-coef[1]*ma - intercept)
    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 imbala
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
4. in the first dataset the ratio between positive and negative is 100 : 2, in the 2nd d
in the 3rd data its 100:40 and in 4th one its 100:80
```

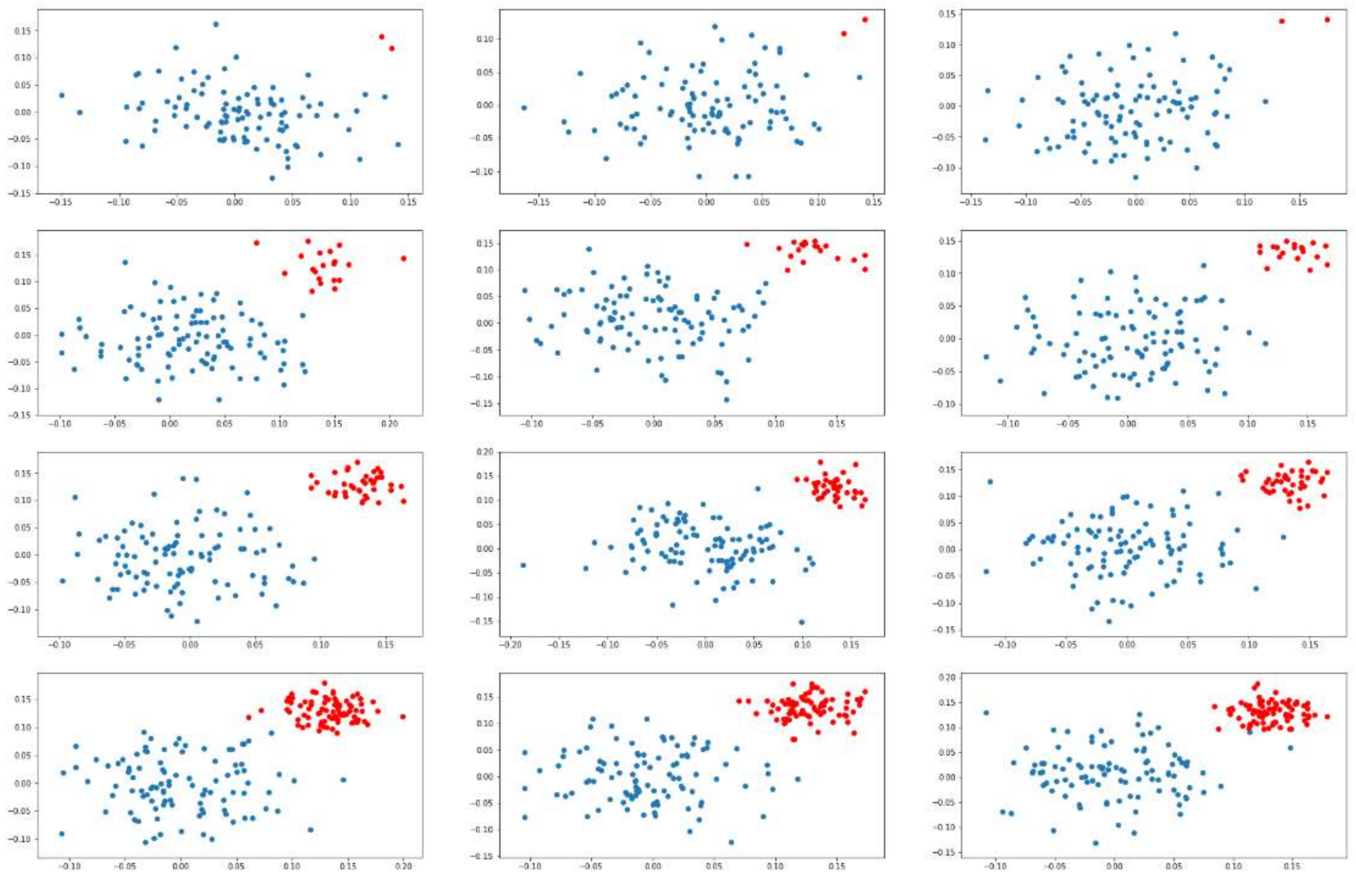
```
# here we are creating 2d imbalanced data points
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
plt.figure(figsize=(20,5))
for j,i in enumerate(ratios):
    plt.subplot(1, 4, j+1)
    X_p=np.random.normal(0,0.05,size=(i[0],2))
    X_n=np.random.normal(0.13,0.02,size=(i[1],2))
    y_p=np.array([1]*i[0]).reshape(-1,1)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

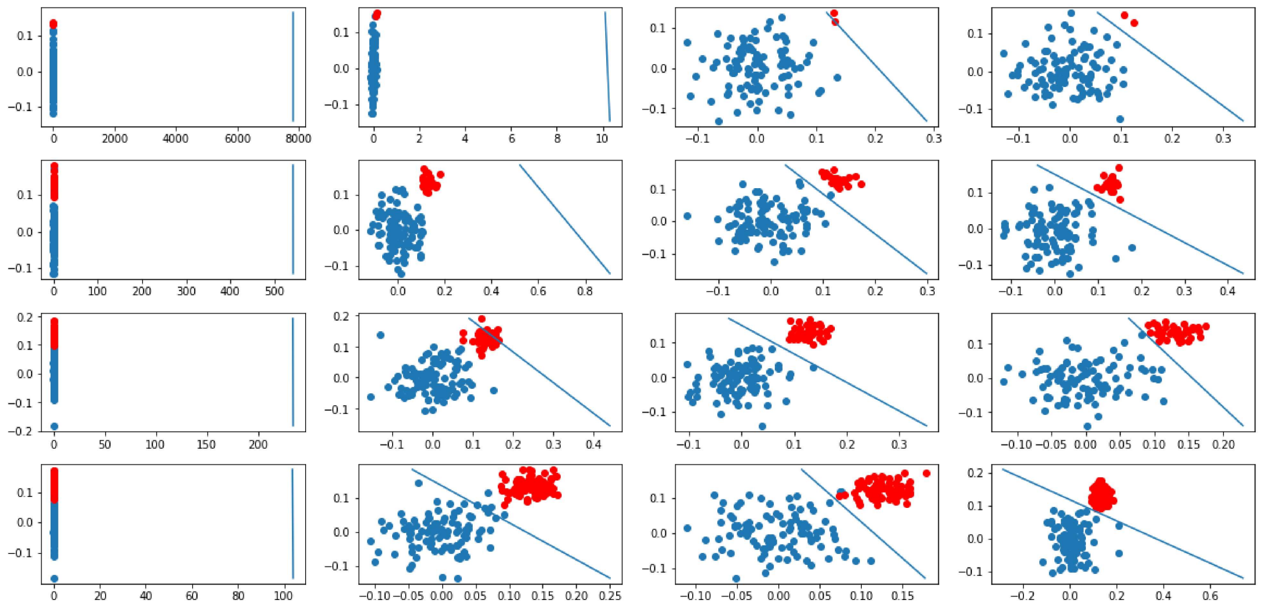


```
in each of the cell[i][j] you will be drawing the hyper plane that you get after applyin
        jth learnig rate


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#mat

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

```python
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
for i in ratios:
  c_val = [0.001, 1, 100, 10000]
  plt.figure(figsize=(20,2))
  for k,val in enumerate(c_val):
    plt.subplot(1, 4, 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))
    plt.scatter(X_p[:,0],X_p[:,1])
    plt.scatter(X_n[:,0],X_n[:,1],color='red')
    clf = SVC(C=val, kernel='linear', degree=3, gamma='scale', coef0=0.0, shrinking=True,
    clf.fit(X,y)
    draw_line(clf.coef_[0],clf.intercept_[0], np.amin(X), np.amax(X))
  plt.show()
```



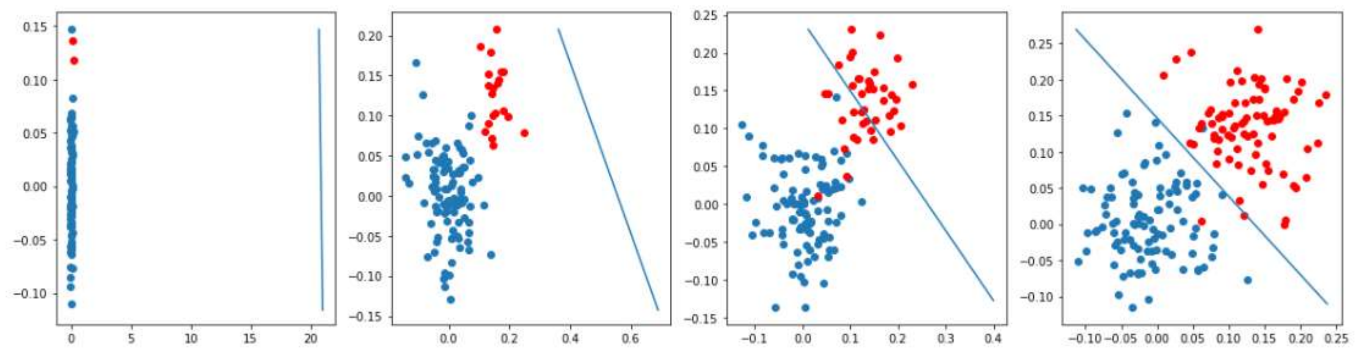
Summary:

1. C with very less value like 0.001 leads to underfit of the the training data points.
2. C with very high value like 10000 leads to overfit of the training data points
3. All ratios of imbalanced data can be seperable linearly when c=100 or more
4. When c=1, No. of support vectors are high

Task 2: Applying LR

```
you will do the same thing what you have done in task 1.1, except instead of SVM you ap
```

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



```
#you can start writing code here.
for i in ratios:
  c_val = [0.001, 1, 100, 10000]
  plt.figure(figsize=(20,2))
  for k,val in enumerate(c_val):
    plt.subplot(1, 4, 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))
    plt.scatter(X_p[:,0],X_p[:,1])
    plt.scatter(X_n[:,0],X_n[:,1],color='red')
    clf = LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=val, fit_intercept=Tr
    clf.fit(X,y)
    draw_line(clf.coef_[0],clf.intercept_[0], np.amin(X), np.amax(X))
  plt.show()
```