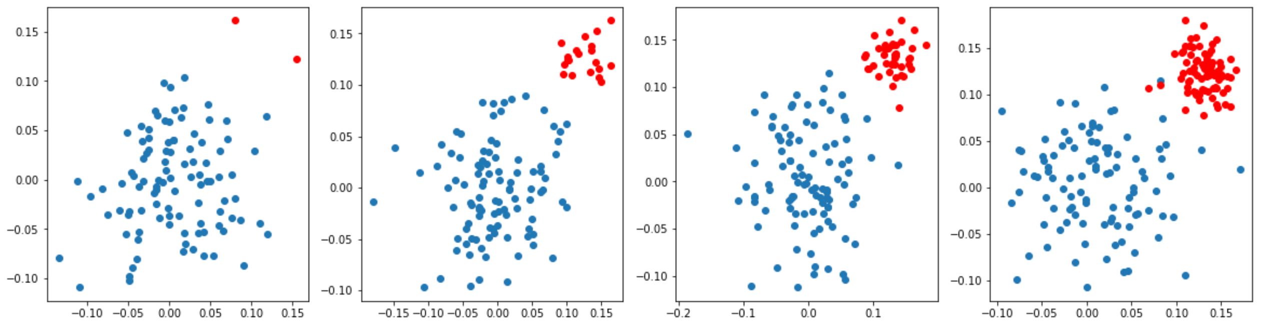
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, Normalizer
import matplotlib.pyplot as plt
from sklearn.svm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

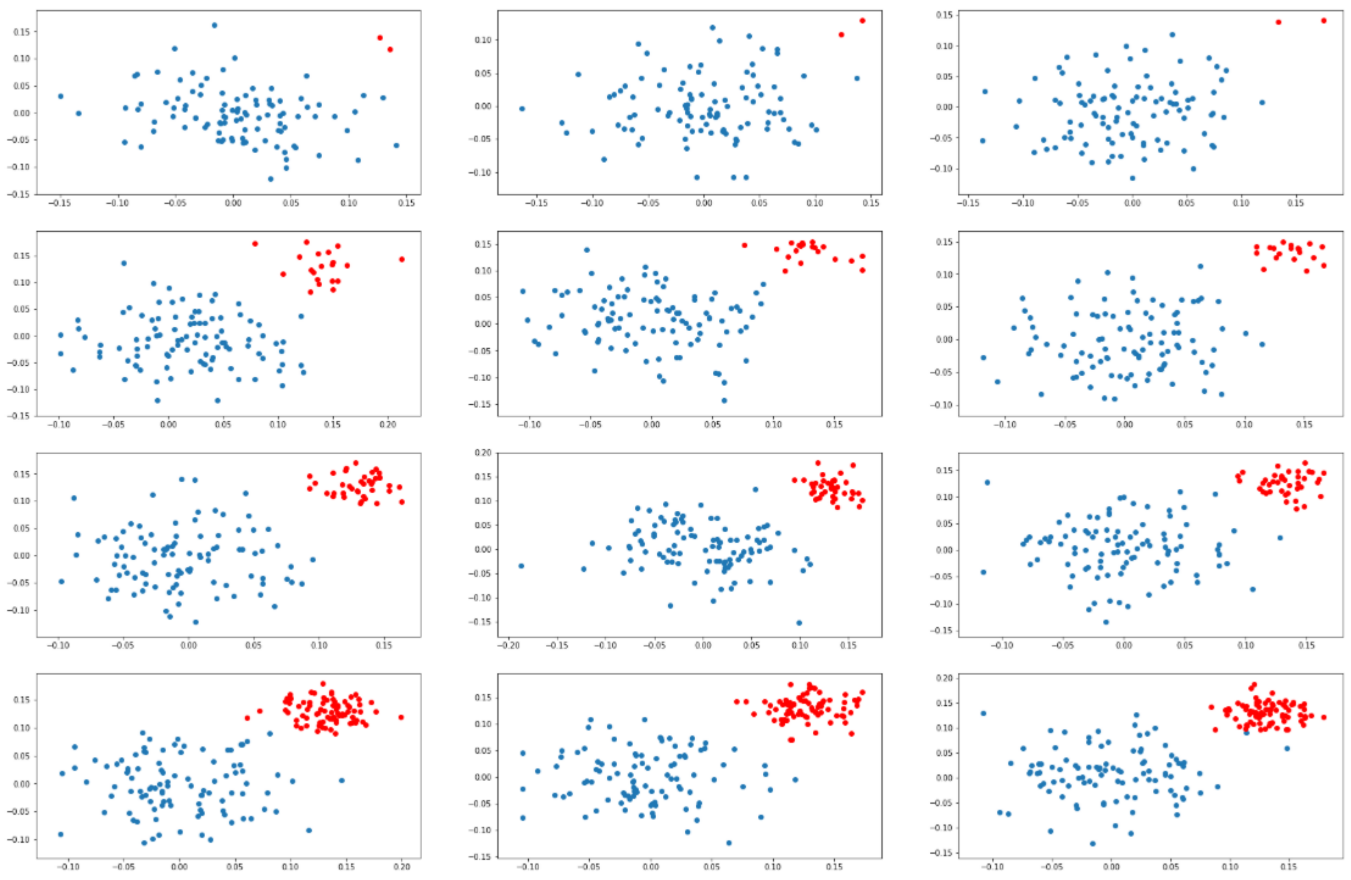
```
# here we are creating 2d imbalanced data points
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
plt.figure(figsize=(20,5))
for j,i in enumerate(ratios):
    plt.subplot(1, 4, j+1)
    X_p=np.random.normal(0,0.05,size=(i[0],2))
    X_n=np.random.normal(0.13,0.02,size=(i[1],2))
    y_p=np.array([1]*i[0]).reshape(-1,1)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

```
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
```

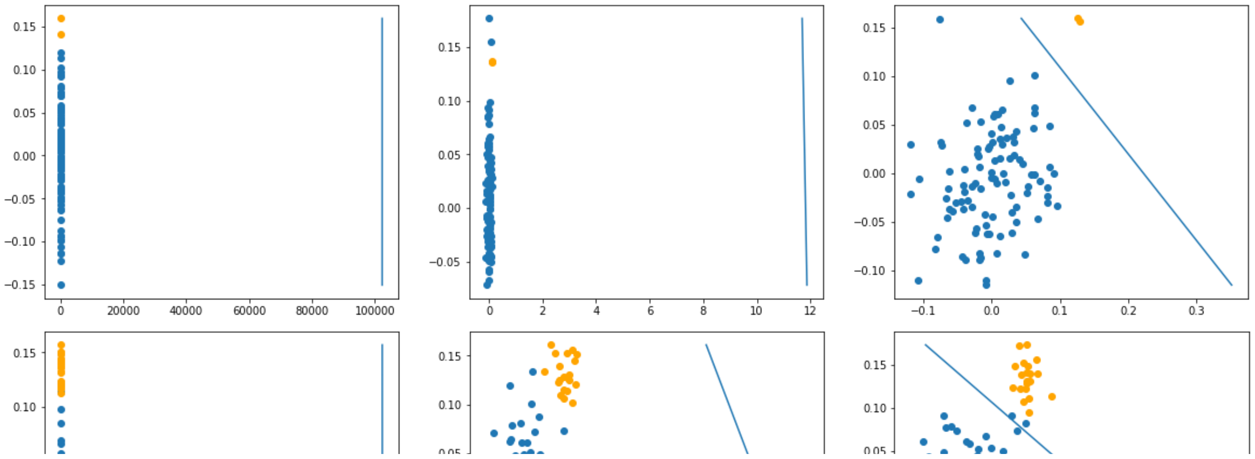
```
i.e
       Plane(SVM().fit(D1, C=0.001))  Plane(SVM().fit(D1, C=1))  Plane(SVM().fit(D1, C=100))
       Plane(SVM().fit(D2, C=0.001))  Plane(SVM().fit(D2, C=1))  Plane(SVM().fit(D2, C=100))
       Plane(SVM().fit(D3, C=0.001))  Plane(SVM().fit(D3, C=1))  Plane(SVM().fit(D3, C=100))
       Plane(SVM().fit(D4, 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.
```

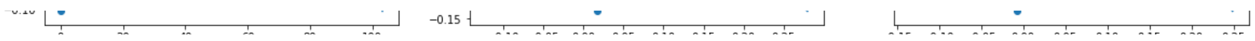
```
# here we are creating 2d imbalanced data points
from sklearn.svm import SVC
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
Learning_Rate_Parameters = [0.001, 1, 100]

for j,i in enumerate(ratios):
  plt.figure(figsize=(20,5))

  for LRP in range(len(Learning_Rate_Parameters)):
    plt.subplot(1, 3, LRP+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 = SVC(kernel = 'linear', C = Learning_Rate_Parameters[LRP]) #Applying SVM Algorith
    clf.fit(X,y)
    intercept = clf.intercept_
    coef = clf.coef_[0]
    mi = np.min(X[:,1])
    ma = np.max(X[:,1])
    draw_line(coef,intercept, mi, ma)
    plt.scatter(X_p[:,0],X_p[:,1])
    plt.scatter(X_n[:,0],X_n[:,1],color='orange')
    plt.scatter(X_n[:,0],X_n[:,1],color='orange')
  plt.show()
```



***Observations on SVM:***

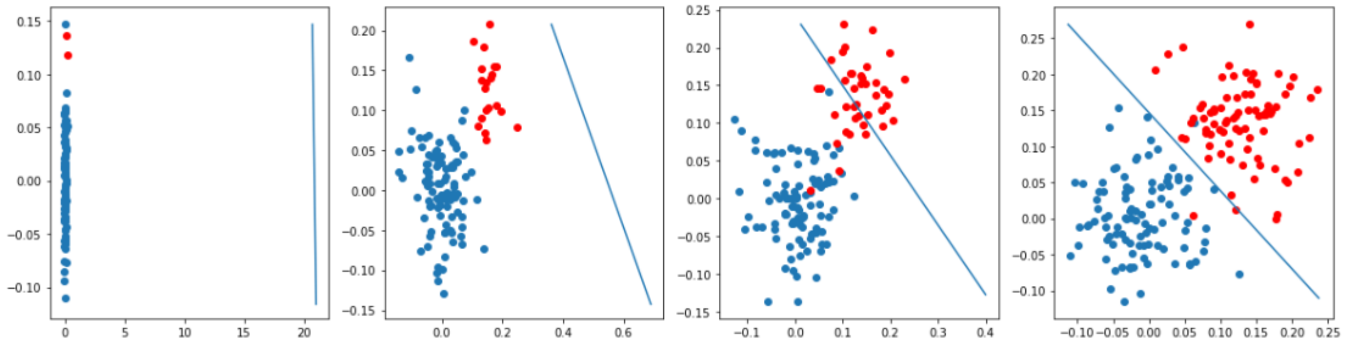
1. When the dataset is extremely imbalanced then the change in learning rate does not classify data.
2. When the dataset is highly imbalanced, then the high change in learning rate classfies the data very well.
3. When the dataset is imbalanced, change in learning rate tries to classify the datapoints significantly well.
4. When the dataset is highly balanced, then change in learning rate eventually classifies the datapoints very well, it includes some misclassified data-points.

## Task 2: Applying LR

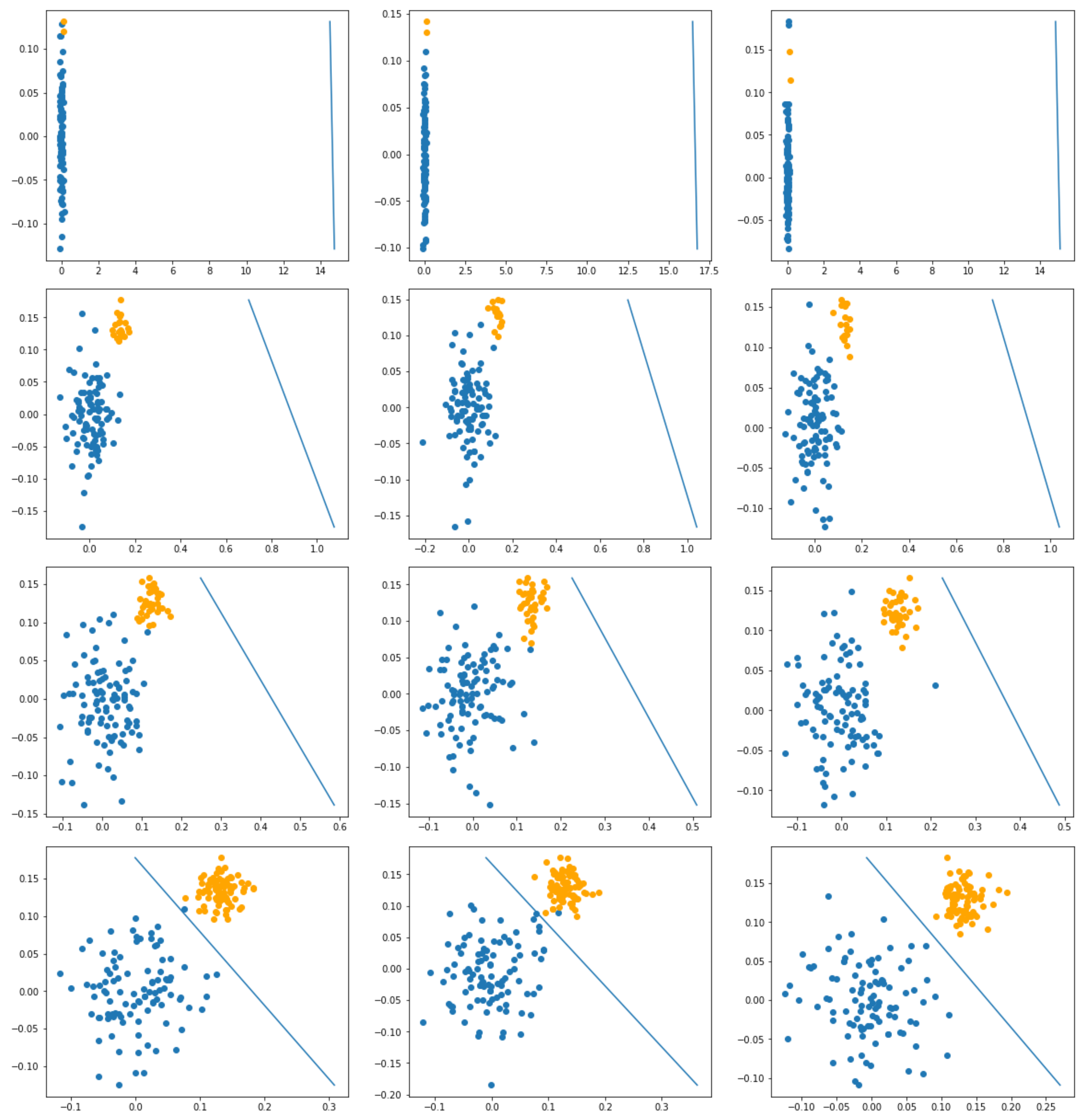


```
you will do the same thing what you have done in task 1.1, except instead of SVM you ap
```

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



```
#you can start writing code here.
from sklearn.linear_model import LogisticRegression
# here we are creating 2d imbalanced data points
from sklearn.svm import SVC
ratios = [(100,2), (100, 20), (100, 40), (100, 80)]
Learning_Rate_Parameters = [0.001, 1, 100]

for j,i in enumerate(ratios):
  plt.figure(figsize=(20,5))

  for LRP in range(len(Learning_Rate_Parameters)):
    plt.subplot(1, 3, LRP+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() #Applying Logistic Regression Algorithm
    clf.fit(X,y)
    intercept = clf.intercept_
    coef = clf.coef_[0]
    mi = np.min(X[:,1])
    ma = np.max(X[:,1])
    draw_line(coef,intercept, mi, ma)
    plt.scatter(X_p[:,0],X_p[:,1])
    plt.scatter(X_n[:,0],X_n[:,1],color='orange')
  plt.show()
```



***Observations on Logistic Regression:***

1. Highly imbalanced dataset with change in learning parameters does not classify the datapoints very effectively.
2. Whereas, highly balanced dataset with change in learning parameters classify the the datapoints in very balanced or effective manner.

# ***Thank you!!***