Behaviour of Linear Models

```
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")
```

Double-click (or enter) to edit

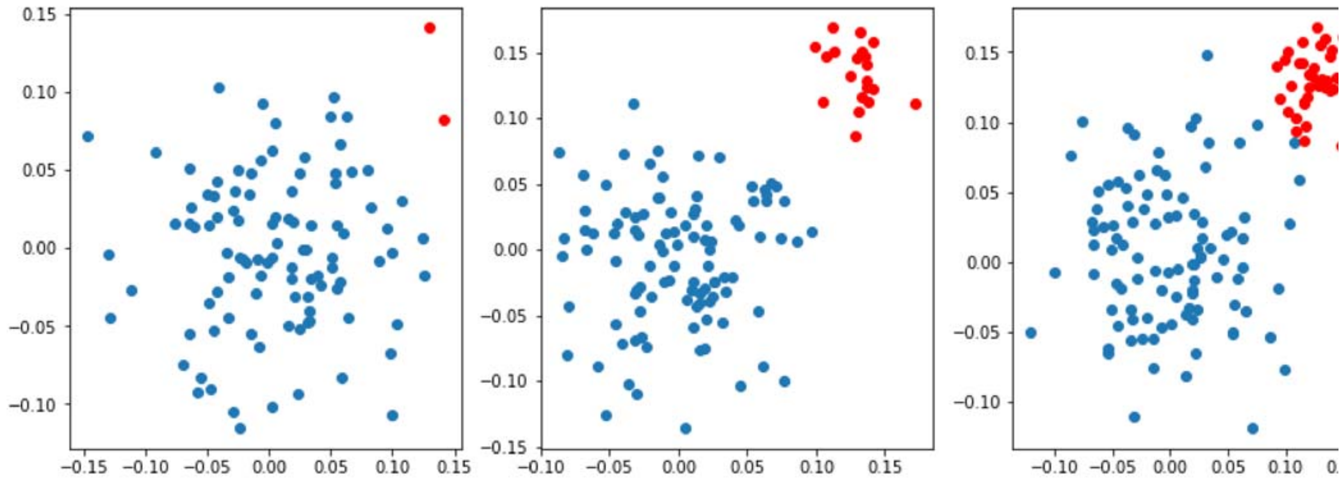
```
def draw_line(coef,intercept, mi, ma):
    # for the separating hyper plane ax+by+c=0, the weights are [a, b] and the intercept is c
    # to draw the hyper plane we are creating two points
    # 1. ((b*min-c)/a, min) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here in place of
    # 2. ((b*max-c)/a, max) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here in place of
    points=np.array([[((-coef[1]*mi - intercept)/coef[0]), mi],[((-coef[1]*ma - intercept)/co
    plt.plot(points[:,0], points[:,1],'black',linestyle='--',alpha=0.7)
```

What if Data is imabalanced

```
1. As a part of this task you will observe how linear models work in case of data imbalance
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 im
4. in the first dataset the ratio between positive and negative is 100 : 2, in the 2nd data
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)
```

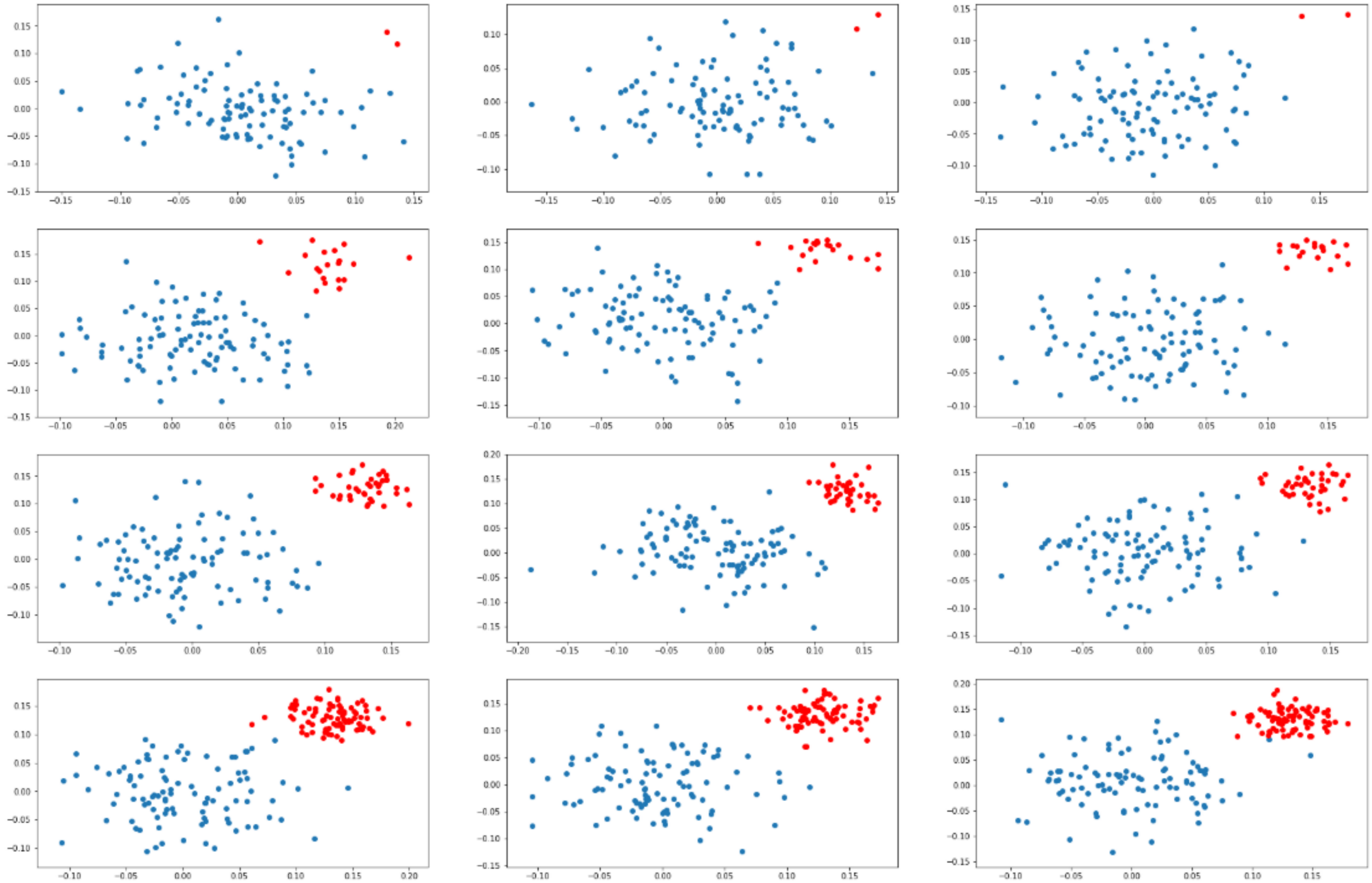
```
        X=np.vstack((X_p,X_n))
        y=np.vstack((y_p,y_n))
        plt.scatter(X_p[:,0],X_p[:,1])
        plt.scatter(X_n[:,0],X_n[:,1],color='red')
    plt.show()
```



> 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



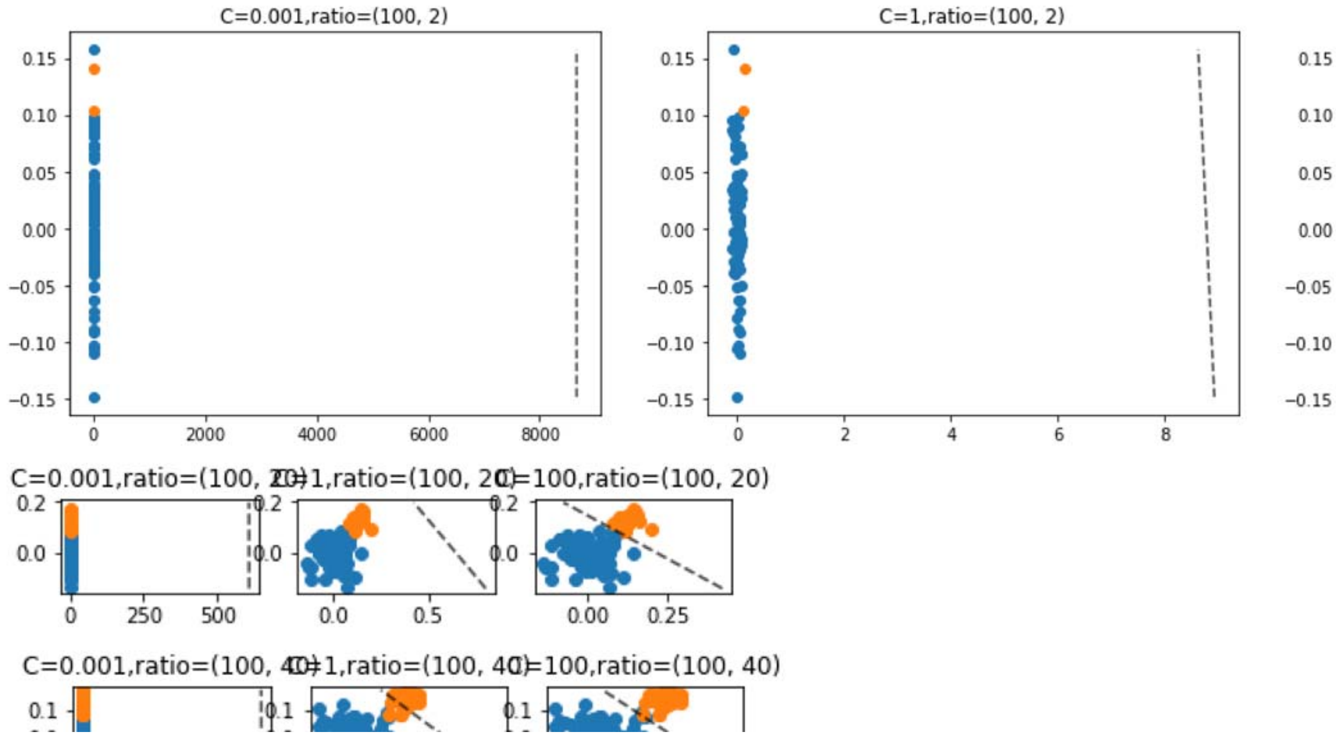
```
in each of the cell[i][j] you will be drawing the hyper plane that you get after applying S
        jth learnig rate

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#mathem

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

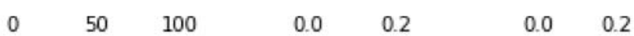
Double-click (or enter) to edit

```
ratios=[(100,2),(100,20),(100,40),(100,80)]
C=[0.001,1,100]

index=1
plt.figure(figsize=(20,20))

for ratio in ratios:
  positives=np.random.normal(0,0.05,size=(ratio[0],2))
  negatives=np.random.normal(0.13,0.02,size=(ratio[1],2))
  pos_labels=np.array([1]*ratio[0]).reshape(-1,1)
  neg_labels=np.array([0]*ratio[1]).reshape(-1,1)
  X=np.vstack((positives,negatives))
  y=np.vstack((pos_labels,neg_labels)).ravel()
  for c in C:
    plt.subplot(len(ratios),len(C),index)
    index+=1

    plt.scatter(positives[:,0],positives[:,1])
    plt.scatter(negatives[:,0],negatives[:,1])
    plt.title(f'C={c},ratio={ratio}')

    svc=SVC(C=c,kernel='linear')
    svc.fit(X,y)
    coef,intercept=svc.coef_,svc.intercept_
    mi,ma=X.min(),X.max()
    draw_line(*coef,*intercept,mi,ma)

  plt.show()
```



when C is small which is inverse of regularization Hyper parameter shows extreme bias in the Model irrespective of ratio of classes

As the c is getting closer to classes 1 less the bias it has



when c is allowed to increase the regularization strength decreases and the model is able to accurately fit the data

When class ratios are similar better C yields better results

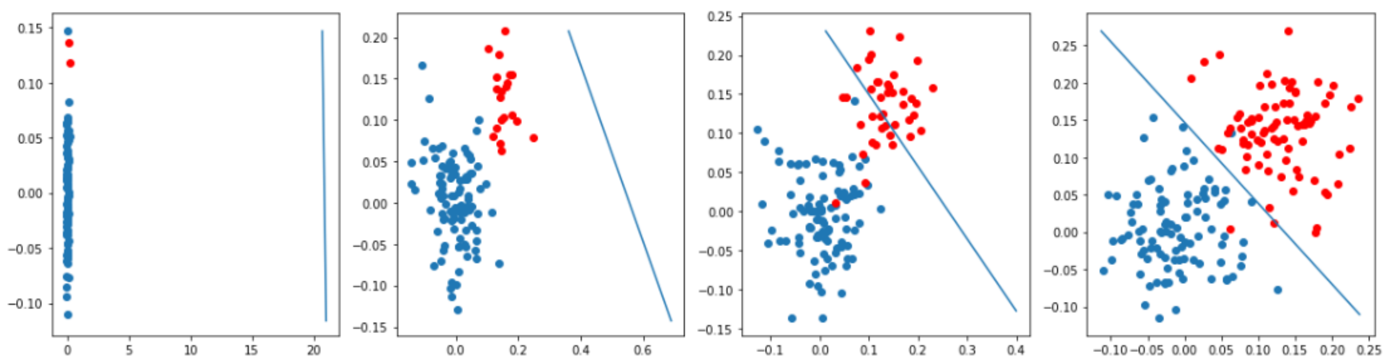
the best models have high c and class ratio approaching 1

Double-click (or enter) to edit

Task 2: Applying LR

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

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



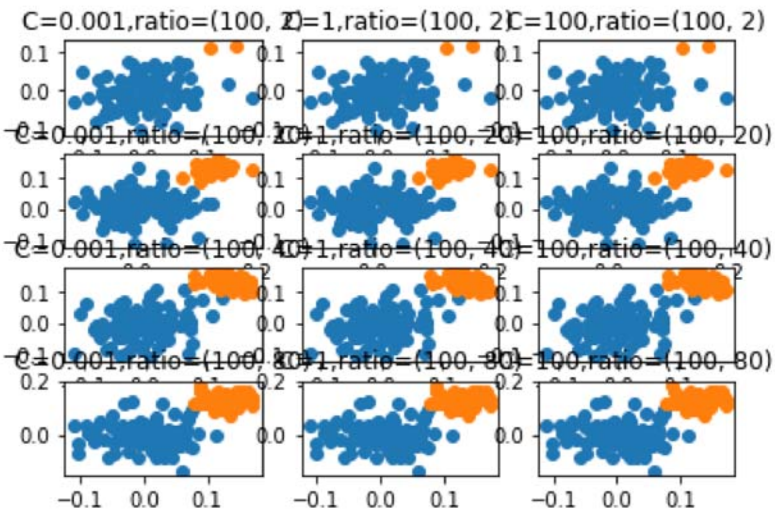
```
#you can start writing code here.
```

Double-click (or enter) to edit

```
index=1
plt.figure(figsize=(20,20))
```

```
<Figure size 1440x1440 with 0 Axes>
<Figure size 1440x1440 with 0 Axes>
```

```
for ratio in ratios:
  positives=np.random.normal(0,0.05,size=(ratio[0],2))
  negatives=np.random.normal(0.13,0.02,size=(ratio[1],2))
  pos_labels=np.array([1]*ratio[0]).reshape(-1,1)
  neg_labels=np.array([0]*ratio[1]).reshape(-1,1)
  X=np.vstack((positives,negatives))
  y=np.vstack((pos_labels,neg_labels)).ravel()

  for c in C:
    plt.subplot(len(ratios),len(C),index)
    index+=1

    plt.scatter(positives[:,0],positives[:,1])
    plt.scatter(negatives[:,0],negatives[:,1])
    plt.title(f'C={c},ratio={ratio}')

    log_reg=LogisticRegression(C=c)
    log_reg.fit(X,y)
    coef,intercept=log_reg.coef_,log_reg.intercept_
    mi,ma=X.min(),X.max()
    draw_line=(*coef,*intercept,mi,ma)

plt.show()
```



As we see SVM is showing better performance than Logistic Regression in this case

and SVM induces less bias compared to Logistic Regression

1s completed at 12:18 PM