What if Data is imabalanced

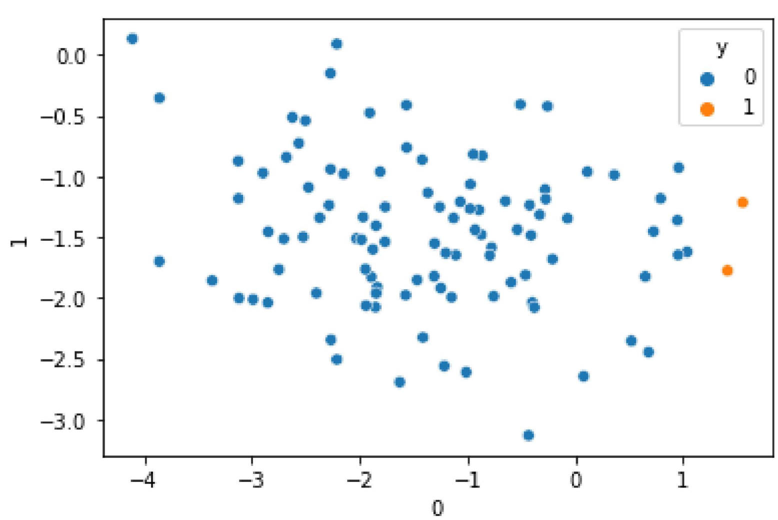
1. As a part of this task you will observe how linear models work in case of data imbalanced
2. observe how hyper plane is changs according to change in your learning rate.
3. below we have created 4 random datasets which are linearly separable and having class imbalance
4. in the first dataset the ratio 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

In [325]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

from sklearn.datasets import make_classification

from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
```

In [121]:
```
data1 = make_classification(n_samples=102,n_features=2,n_informative=2,n_redundant=0,n_
df1 = pd.DataFrame(data1[0])
df1['y'] = data1[1]
```

In [140]:
```
sns.scatterplot(x=df1[0],y=df1[1],hue=df1['y'])
```

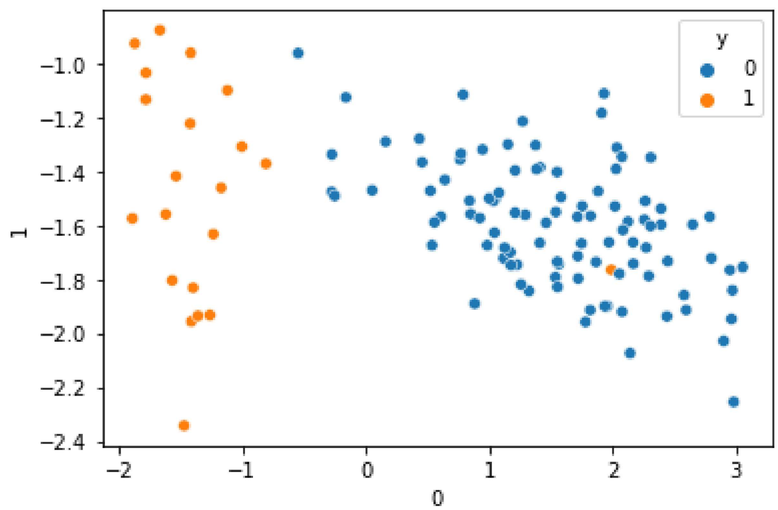
Out[140]: <AxesSubplot:xlabel='0', ylabel='1'>



In [138]:
```python
data = make_classification(n_samples=120,n_features=2,n_informative=2,n_redundant=0,n_
df = pd.DataFrame(data[0])
df['y'] = data[1]
```

In [139]:
```python
sns.scatterplot(x=df[0],y=df[1],hue=df['y'])
```

Out[139]: <AxesSubplot:xlabel='0', ylabel='1'>

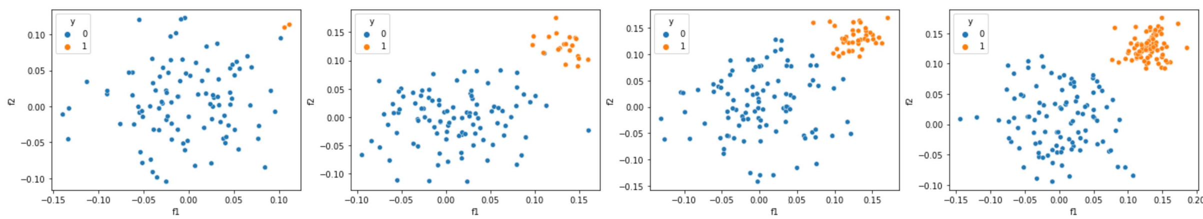


I the above case we can create an dataset of two classes but, fail to provide consistent format, as we can see the class 2 in the first data set is on the east side but in the second case it is on west side. So basically this method will not work.

In [192]:
```python
np.random.seed(9)
def sample_generator(one,m1,v1,two,m2,v2):
    x1 = np.random.normal(m1,v1,(one,2))
    y1 = np.zeros((one,1))
    data1 = np.concatenate((x1,y1),axis=1)

    x2 = np.random.normal(m2,v2,(two,2))
    y2 = np.ones((two,1))
    data2 = np.concatenate((x2,y2),axis=1)

    data = np.concatenate((data1,data2),axis=0)
    df = pd.DataFrame(data,columns=['f1','f2','y']).astype({'y': 'int32'})
    return df
```

In [193]:

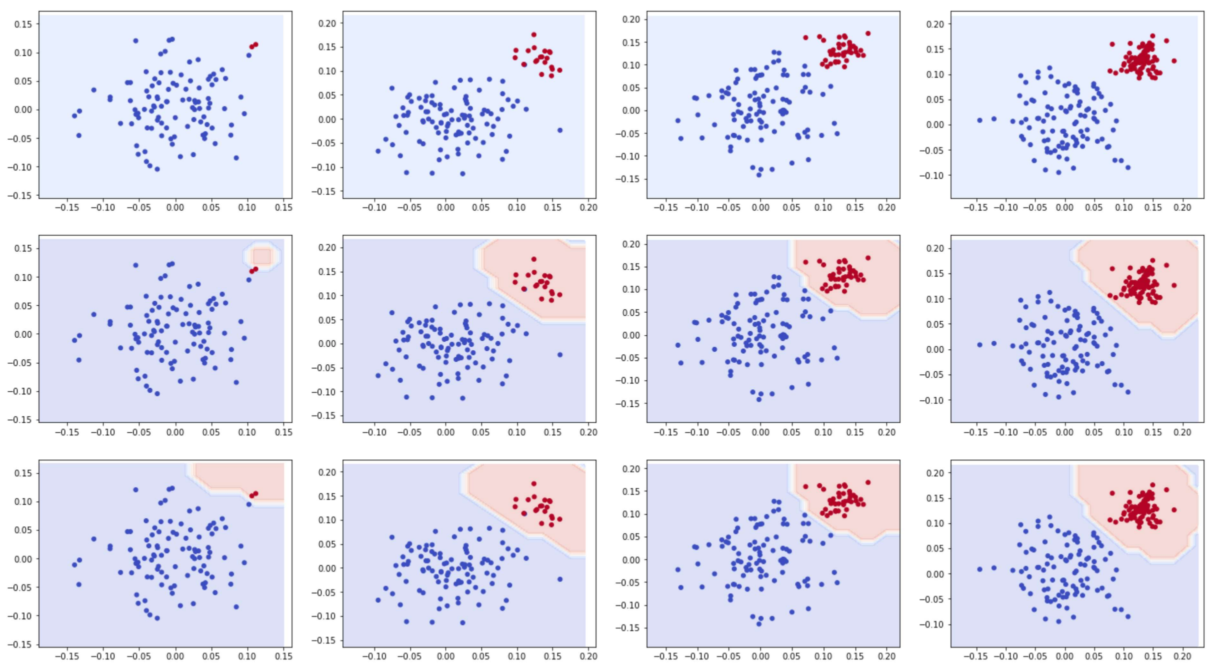
```
plt.rcParams["figure.figsize"] = (25,4)
df1 = sample_generator(100,0,0.05,2,0.13,0.02)
plt.subplot(141)
sns.scatterplot(data=df1,x='f1',y='f2',hue='y')

df2 = sample_generator(100,0,0.05,20,0.13,0.02)
plt.subplot(142)
sns.scatterplot(data=df2,x='f1',y='f2',hue='y')

df3 = sample_generator(100,0,0.05,40,0.13,0.02)
plt.subplot(143)
sns.scatterplot(data=df3,x='f1',y='f2',hue='y')

df4 = sample_generator(100,0,0.05,80,0.13,0.02)
plt.subplot(144)
sns.scatterplot(data=df4,x='f1',y='f2',hue='y')

plt.show()
```



Task 1: Applying SVM

RBF

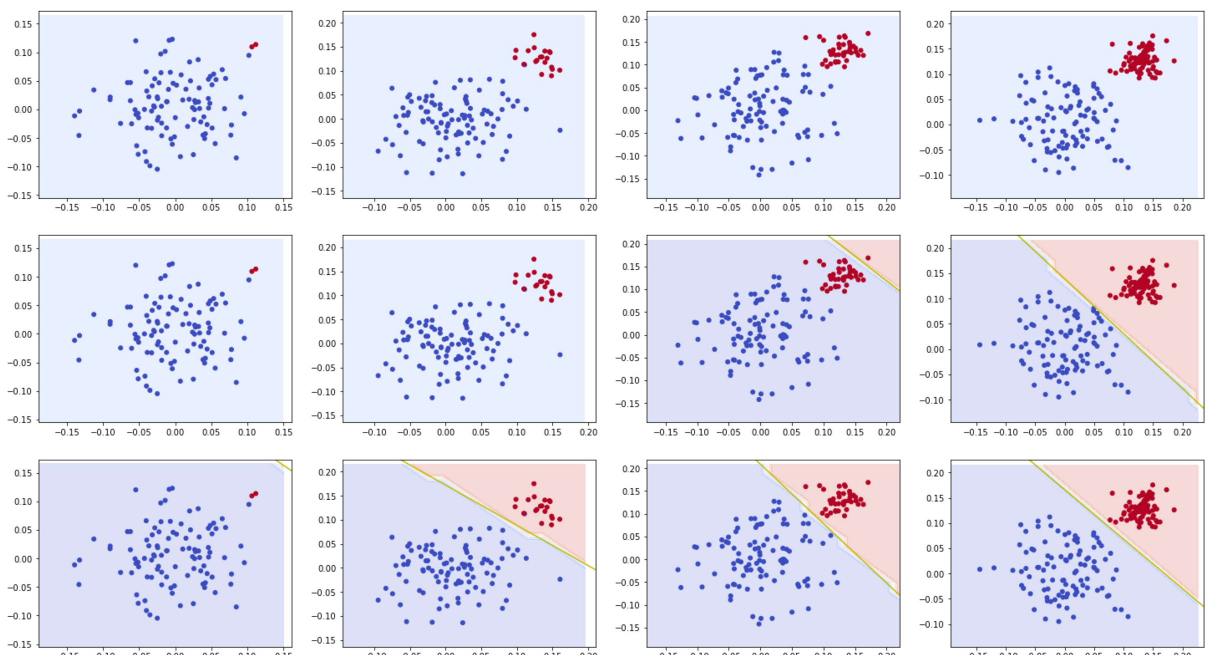
In [302]:
```python
plt.rcParams["figure.figsize"] = (25,14)
f,axs = plt.subplots(3,4)
i=0
for c in [0.001, 1, 100]:
    j=0
    for df in [df1,df2,df3,df4]:
        svc = SVC(C=c,kernel='rbf').fit(df[['f1','f2']],df['y'])

        # create a mesh to plot in
        x_min, x_max = df.f1.min() - 0.05, df.f1.max() + 0.05
        y_min, y_max = df.f2.min() - 0.05, df.f2.max() + 0.05
        xx2, yy2 = np.meshgrid(np.arange(x_min, x_max, .02),np.arange(y_min, y_max, .02))
        Z = svc.predict(np.c_[xx2.ravel(), yy2.ravel()])
        Z = Z.reshape(xx2.shape)
        ax = axs[i][j]
        ax.contourf(xx2, yy2, Z, cmap=plt.cm.coolwarm, alpha=0.2)
        ax.scatter(df.f1, df.f2, c=df.y, cmap=plt.cm.coolwarm, s=25)
        ax.axis([x_min, x_max,y_min, y_max])
        j=j+1
    i=i+1
plt.show()
```



Linear SVM

with axes bounded

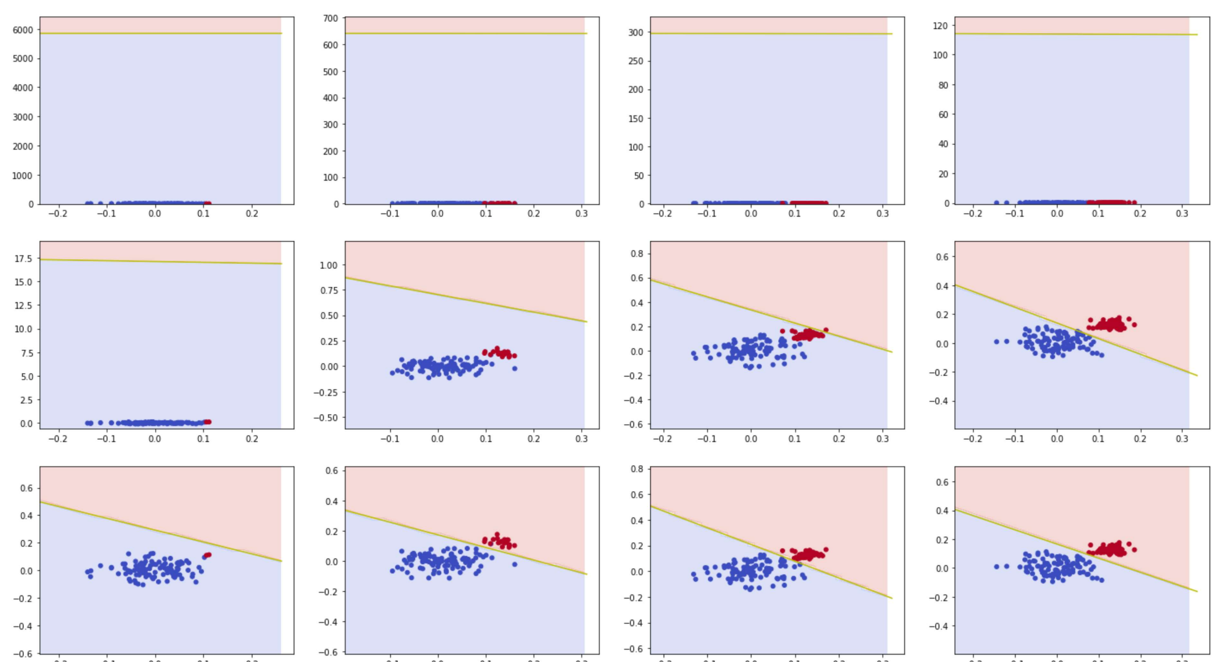
In [318]:
```
plt.rcParams["figure.figsize"] = (25,14)
f,axs = plt.subplots(3,4)
i=0
for c in [0.001, 1, 100]:
    j=0
    for df in [df1,df2,df3,df4]:
        svc = SVC(C=c,kernel='linear').fit(df[['f1','f2']],df['y'])

        # create a mesh to plot in
        x_min, x_max = df.f1.min() - 0.05, df.f1.max() + 0.05
        y_min, y_max = df.f2.min() - 0.05, df.f2.max() + 0.05
        xx2, yy2 = np.meshgrid(np.arange(x_min, x_max, .02),np.arange(y_min, y_max, .02))
        Z = svc.predict(np.c_[xx2.ravel(), yy2.ravel()])
        Z = Z.reshape(xx2.shape)

        #Plotting Line
        w = svc.coef_[0]
        a = -w[0] / w[1]
        xx = np.linspace(x_min, x_max)
        yy = a * xx - (svc.intercept_[0]) /w[1]

        #Plotting
        ax = axs[i][j]
        ax.contourf(xx2, yy2, Z, cmap=plt.cm.coolwarm, alpha=0.2)
        ax.scatter(df.f1, df.f2, c=df.y, cmap=plt.cm.coolwarm, s=25)
        ax.axis([x_min, x_max,y_min, y_max])
        ax.plot(xx,yy,c='y')
        j=j+1
    i=i+1
plt.show()
```



Linear SVM

without bounded axis

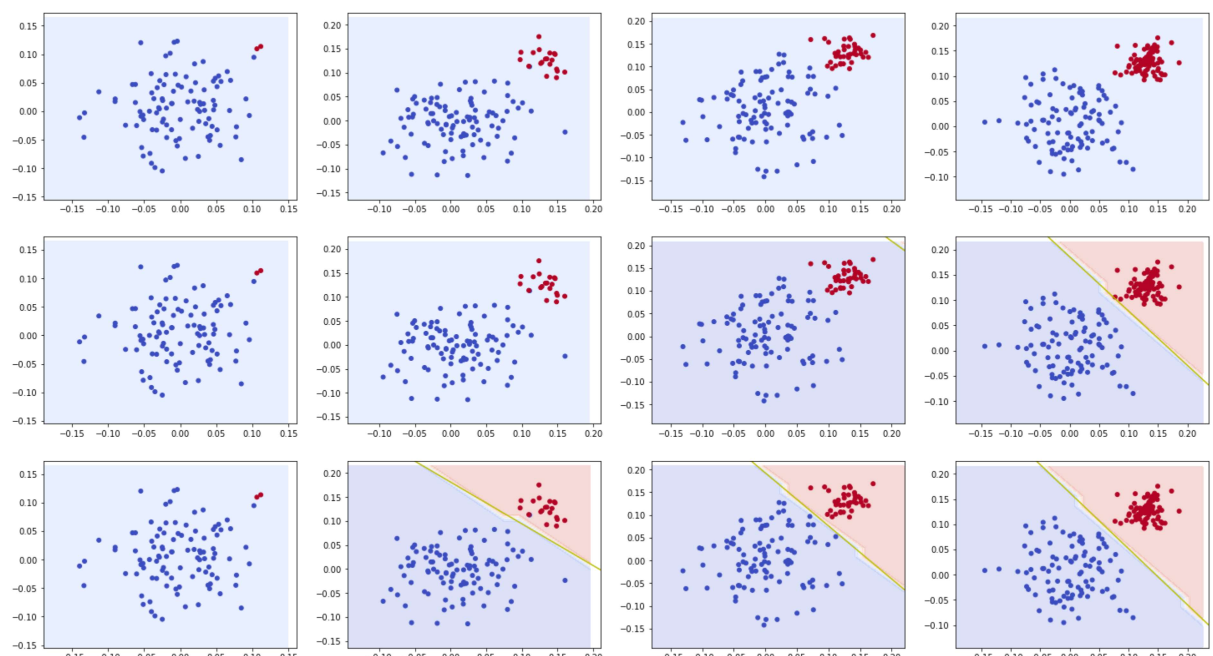
In [326]:
```
plt.rcParams["figure.figsize"] = (25,14)
f,axs = plt.subplots(3,4)
i=0
for c in [0.001, 1, 100]:
    j=0
    for df in [df1,df2,df3,df4]:
        svc = SVC(C=c,kernel='linear').fit(df[['f1','f2']],df['y'])

        # create a mesh to plot in
        x_min, x_max = df.f1.min() - 0.1, df.f1.max() + 0.15
        y_min, y_max = df.f2.min() - 0.5, df.f2.max() + 0.1

        #Plotting Line
        w = svc.coef_[0]
        a = -w[0] / w[1]
        xx = np.linspace(x_min, x_max)
        yy = a * xx - (svc.intercept_[0]) /w[1]

        #prediction
        xx2, yy2 = np.meshgrid(np.arange(x_min, x_max, .02),np.arange(y_min, y_max+(yy.max()
        Z = svc.predict(np.c_[xx2.ravel(), yy2.ravel()])
        Z = Z.reshape(xx2.shape)

        #Plotting
        ax = axs[i][j]
        ax.contourf(xx2, yy2, Z, cmap=plt.cm.coolwarm, alpha=0.2)
        ax.scatter(df.f1, df.f2, c=df.y, cmap=plt.cm.coolwarm, s=25)
        # ax.axis([x_min, x_max,y_min, y_max])
        ax.plot(xx,yy,c='y')
        j=j+1
    i=i+1
plt.show()
```



Task 2: Applying LR

with bounded axis

In [327]:
```
plt.rcParams["figure.figsize"] = (25,14)
f,axs = plt.subplots(3,4)
i=0
for c in [0.001, 1, 100]:
    j=0
    for df in [df1,df2,df3,df4]:
        logiclf = LogisticRegression(C=c,fit_intercept=True,random_state=99,n_jobs=-1).fit(df

        # create a mesh to plot in
        x_min, x_max = df.f1.min() - 0.05, df.f1.max() + 0.05
        y_min, y_max = df.f2.min() - 0.05, df.f2.max() + 0.05
        xx2, yy2 = np.meshgrid(np.arange(x_min, x_max, .02),np.arange(y_min, y_max, .02))
        Z = logiclf.predict(np.c_[xx2.ravel(), yy2.ravel()])
        Z = Z.reshape(xx2.shape)

        #Plotting Line
        w = logiclf.coef_[0]
        a = -w[0] / w[1]
        xx = np.linspace(x_min, x_max)
        yy = a * xx - (logiclf.intercept_[0]) /w[1]

        #Plotting
        ax = axs[i][j]
        ax.contourf(xx2, yy2, Z, cmap=plt.cm.coolwarm, alpha=0.2)
        ax.scatter(df.f1, df.f2, c=df.y, cmap=plt.cm.coolwarm, s=25)
        ax.axis([x_min, x_max,y_min, y_max])
        ax.plot(xx,yy,c='y')
        j=j+1
    i=i+1
plt.show()
```

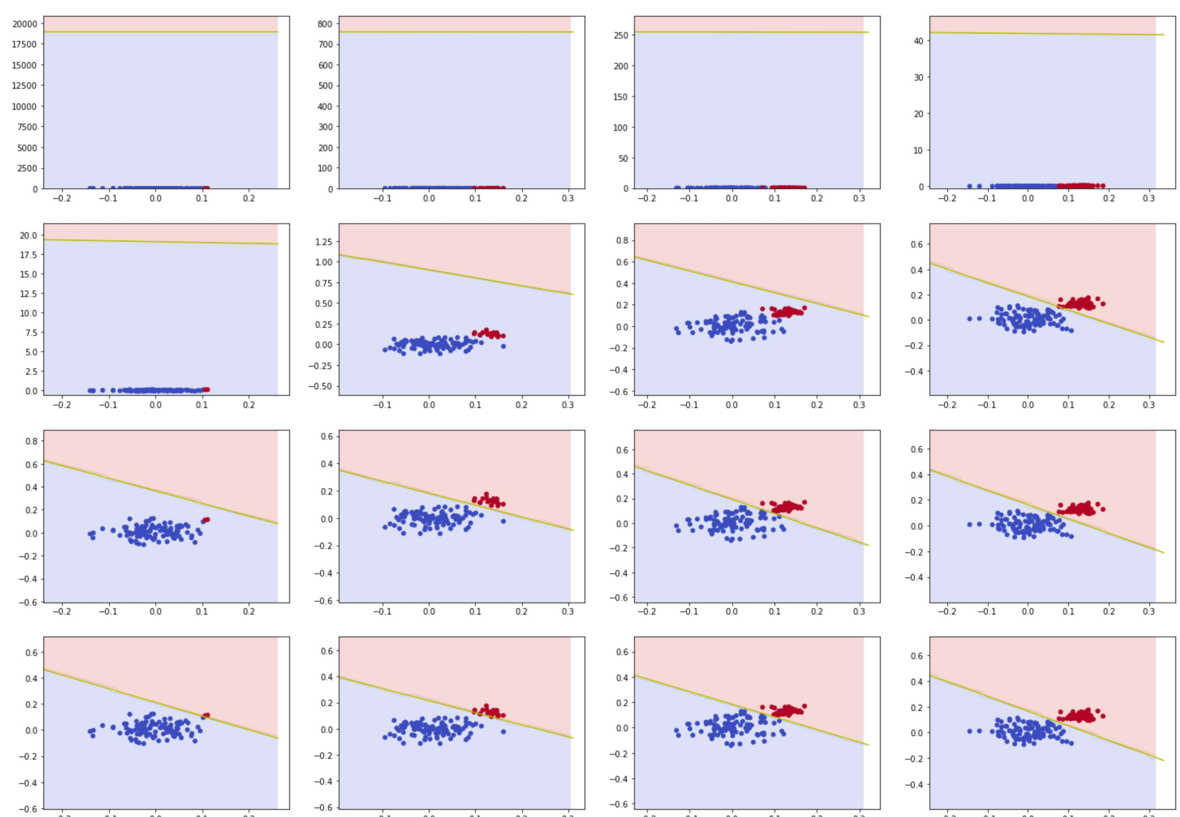


LR

without bounded axis

In [333]:

```
plt.rcParams["figure.figsize"] = (25,18)
f,axs = plt.subplots(4,4)
i=0
for c in [0.001, 1,100, 100000000]: # the last causes the lambda to almost become zero hence
    j=0
    for df in [df1,df2,df3,df4]:
        logiclf = LogisticRegression(C=c,fit_intercept=True,random_state=99,n_jobs=-1).fit(df

        # create a mesh to plot in
        x_min, x_max = df.f1.min() - 0.1, df.f1.max() + 0.15
        y_min, y_max = df.f2.min() - 0.5, df.f2.max() + 0.1

        #Plotting Line
        w = logiclf.coef_[0]
        a = -w[0] / w[1]
        xx = np.linspace(x_min, x_max)
        yy = a * xx - (logiclf.intercept_[0]) /w[1]

        #prediction
        xx2, yy2 = np.meshgrid(np.arange(x_min, x_max, .02),np.arange(y_min, y_max+(yy.max()
        Z = logiclf.predict(np.c_[xx2.ravel(), yy2.ravel()])
        Z = Z.reshape(xx2.shape)

        #Plotting
        ax = axs[i][j]
        ax.contourf(xx2, yy2, Z, cmap=plt.cm.coolwarm, alpha=0.2)
        ax.scatter(df.f1, df.f2, c=df.y, cmap=plt.cm.coolwarm, s=25)
        # ax.axis([x_min, x_max,y_min, y_max])
        ax.plot(xx,yy,c='y')
        j=j+1
    i=i+1
plt.show()
```



In [ ]:
```
#### ` We can see that, as C is 1/Lambda for lower C values the seperation line is almost h

Now for an constant C, as the data becomes more balanced even with high regularization the
This happens as the loss function has two parts one-the error from points and two-the regul
1> High unbalanced/ balanced data & high regularization : The coeff (solpe of the line) tends
2> Data becomes balanced : As we make the data balance, the significance of the error funct
3> Lower regularization: At lower lambdas the error function has considerable weitage and h
`
```