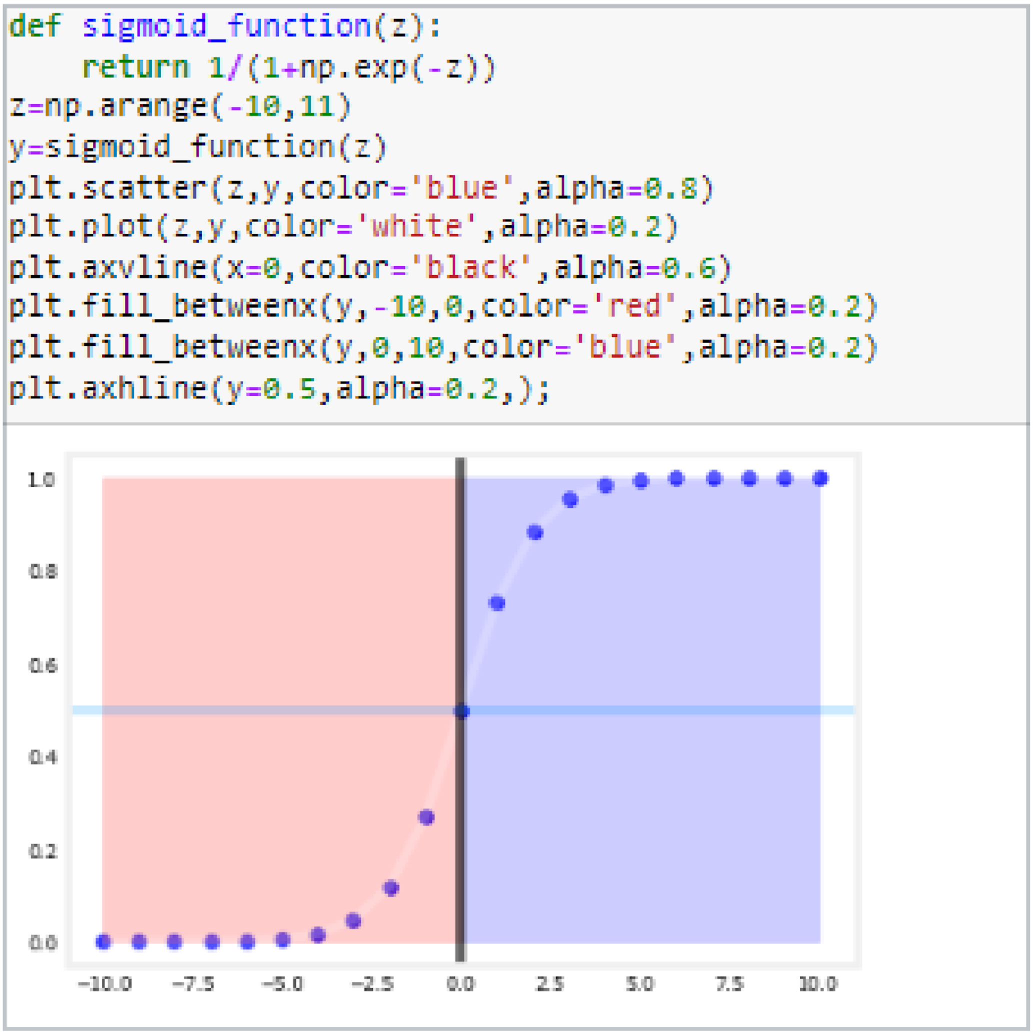
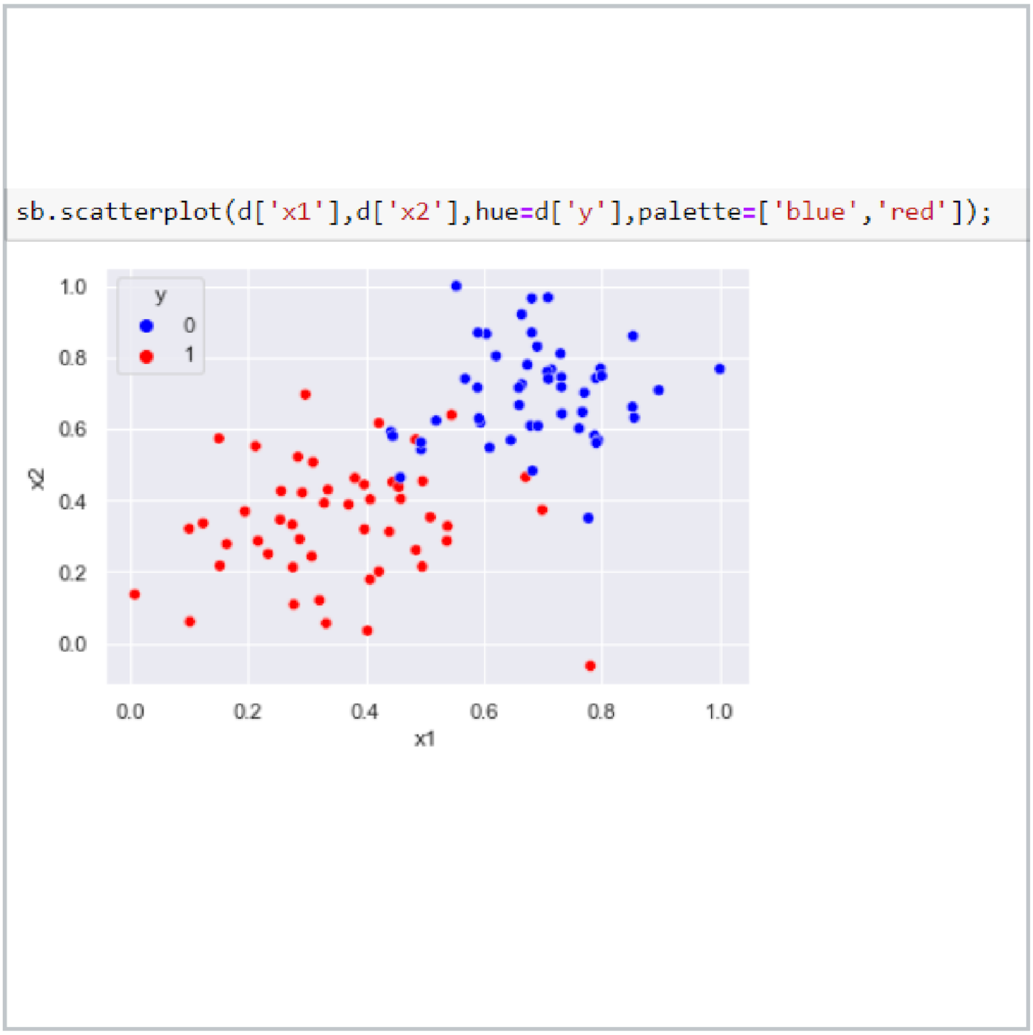
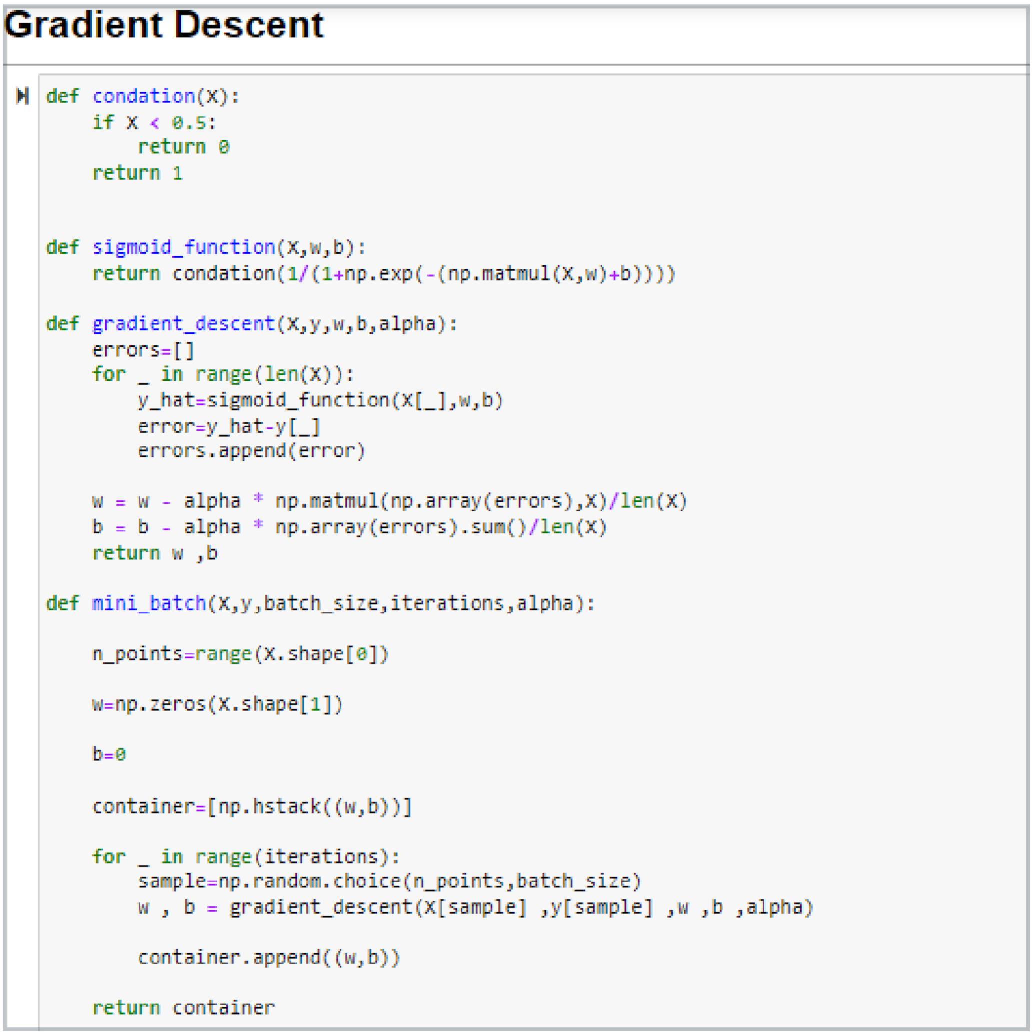
# Logistic Regression From Scratch



```
def sigmoid_function(z):
    return 1/(1+np.exp(-z))
z=np.arange(-10,11)
y=sigmoid_function(z)
plt.scatter(z,y,color='blue',alpha=0.8)
plt.plot(z,y,color='white',alpha=0.2)
plt.axvline(x=0,color='black',alpha=0.6)
plt.fill_betweenx(y,-10,0,color='red',alpha=0.2)
plt.fill_betweenx(y,0,10,color='blue',alpha=0.2)
plt.axhline(y=0.5,alpha=0.2,);
```

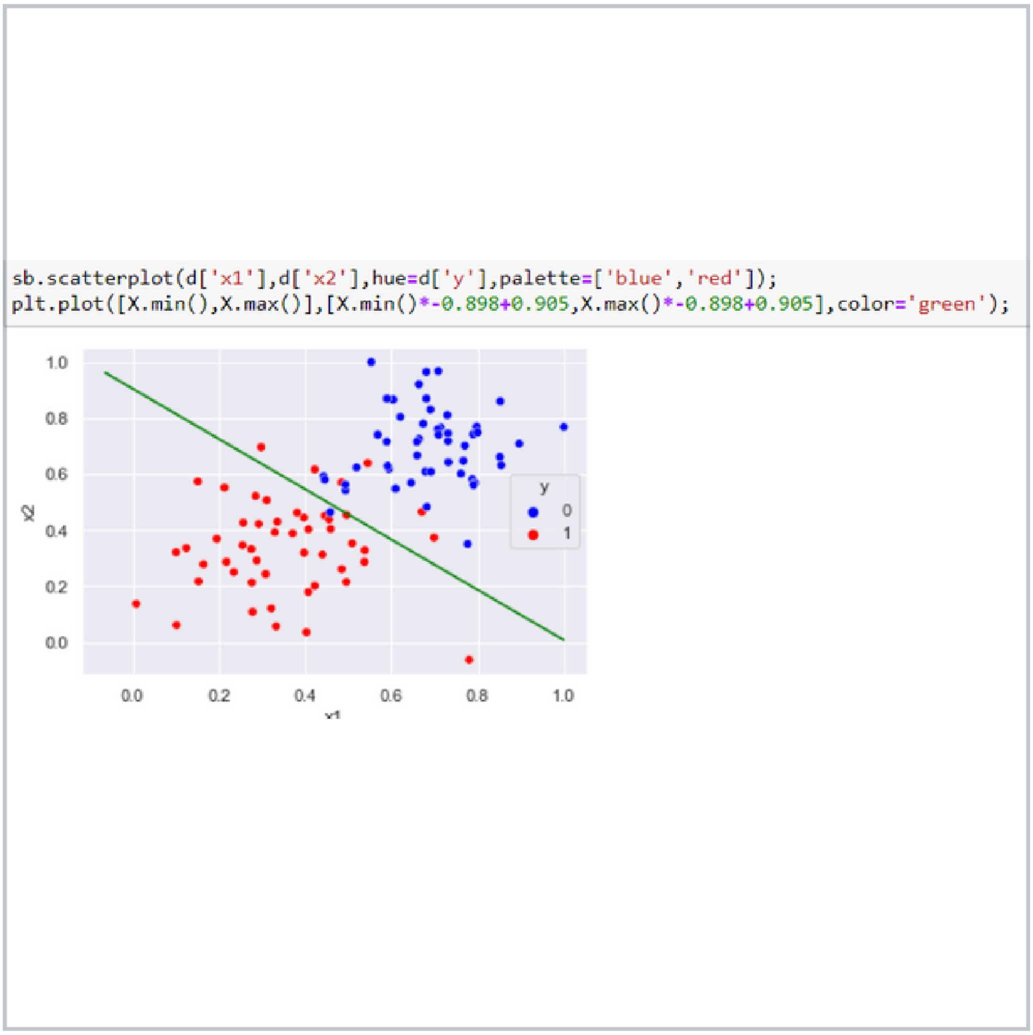


```
sb.scatterplot(d['x1'],d['x2'],hue=d['y'],palette=['blue','red']);
```

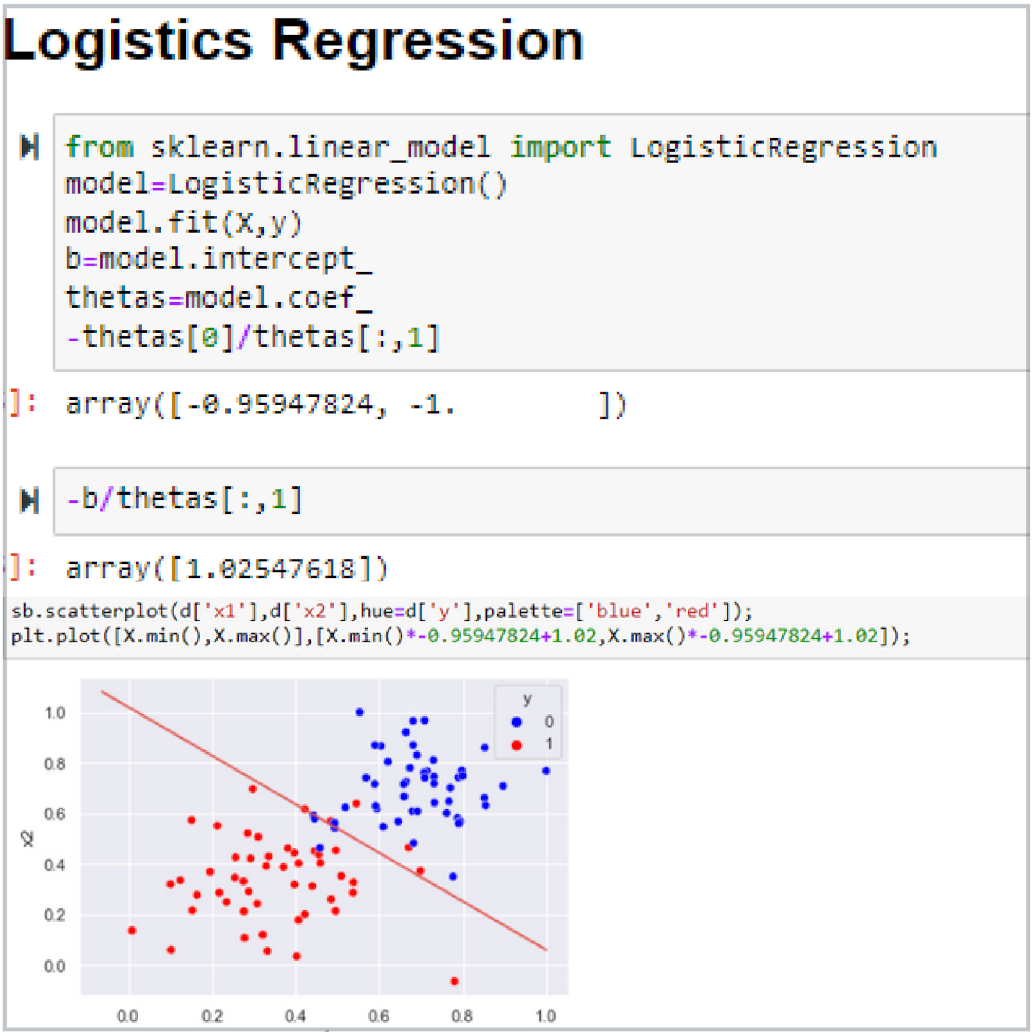


# Gradient Descent

```
def condation(X):
    if X < 0.5:
        return 0
    return 1


def sigmoid_function(X,w,b):
    return condation(1/(1+np.exp(-(np.matmul(X,w)+b))))

def gradient_descent(X,y,w,b,alpha):
    errors=[]
    for _ in range(len(X)):
        y_hat=sigmoid_function(X[_],w,b)
        error=y_hat-y[_]
        errors.append(error)

    w = w - alpha * np.matmul(np.array(errors),X)/len(X)
    b = b - alpha * np.array(errors).sum()/len(X)
    return w ,b

def mini_batch(X,y,batch_size,iterations,alpha):

    n_points=range(X.shape[0])

    w=np.zeros(X.shape[1])

    b=0

    container=[np.hstack((w,b))]

    for _ in range(iterations):
        sample=np.random.choice(n_points,batch_size)
        w , b = gradient_descent(X[sample] ,y[sample] ,w ,b ,alpha)

        container.append((w,b))

    return container
```



```
sb.scatterplot(d['x1'],d['x2'],hue=d['y'],palette=['blue','red']);
plt.plot([X.min(),X.max()],[X.min()*-0.898+0.905,X.max()*-0.898+0.905],color='green');
```



# Logistics Regression

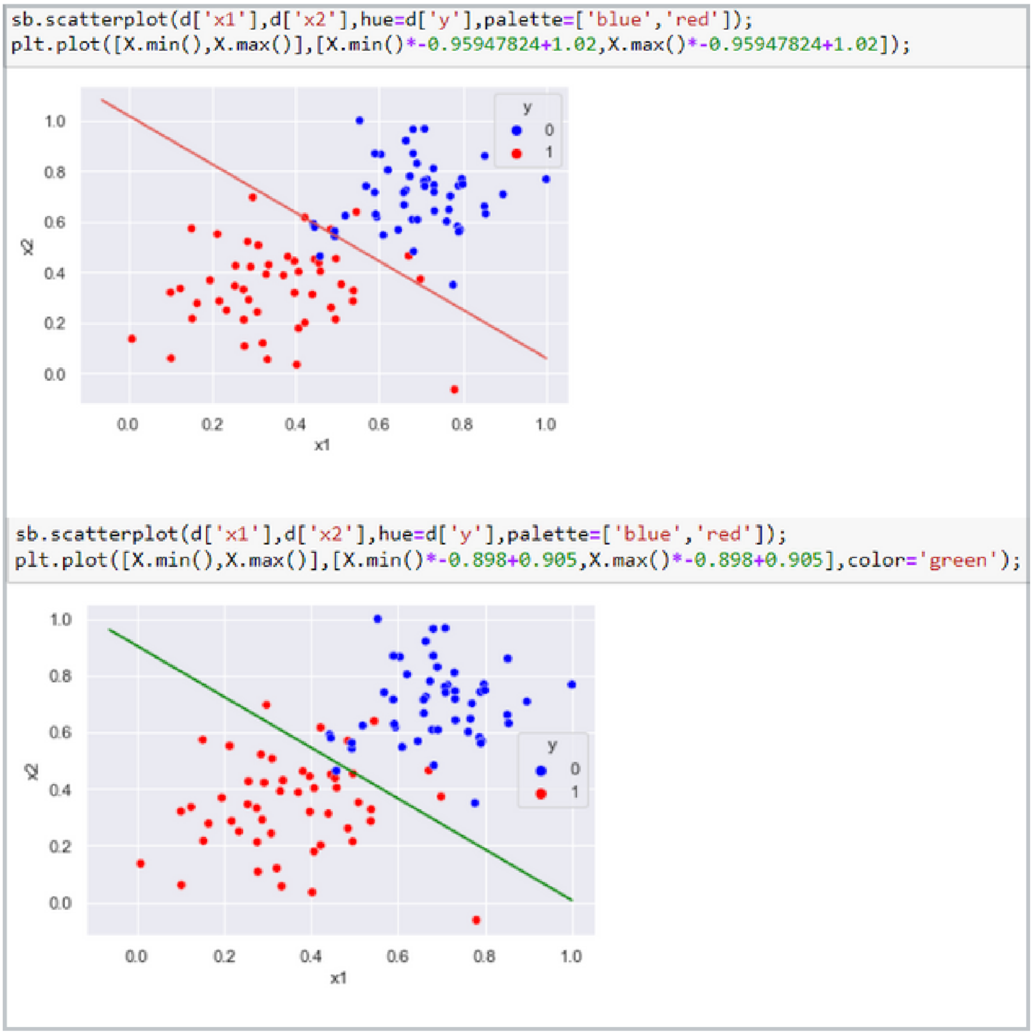
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X,y)
b=model.intercept_
thetas=model.coef_
-thetas[0]/thetas[:,1]
```

```
]:  array([-0.95947824, -1.        ])
```

```
-b/thetas[:,1]
```

```
]:  array([1.02547618])
```

```
sb.scatterplot(d['x1'],d['x2'],hue=d['y'],palette=['blue','red']);
plt.plot([X.min(),X.max()],[X.min()*-0.95947824+1.02,X.max()*-0.95947824+1.02]);
```



```
sb.scatterplot(d['x1'],d['x2'],hue=d['y'],palette=['blue','red']);
plt.plot([X.min(),X.max()],[X.min()*-0.95947824+1.02,X.max()*-0.95947824+1.02]);
```

```
sb.scatterplot(d['x1'],d['x2'],hue=d['y'],palette=['blue','red']);
plt.plot([X.min(),X.max()],[X.min()*-0.898+0.905,X.max()*-0.898+0.905],color='green');
```