一、Logistic regression function.

```
def logistic_regression(train,w,b,learningrate):
    G_w =np.zeros(57)
    G_b = 0.0
    s_loss = 0.0

    for i in range(0,len(train)):
        tem = train.iloc[i,range(1,58)].get_values().tolist()
        f = prediction(tem,w,b)
        if f == 0.0:
            f = 0.00001
        if f == 1.0:
            f = 0.99999
        real = train.iloc[i,58]
        s_loss -= (real*math.log(f)+(1-real)*math.log(1-f))
        G_w0 = np.dot((real - f),tem)
        G_w = np.array(G_w)-np.array(G_w0)
        G_b -= (real - f)

    w0 = np.dot(learningrate,G_w)
    w = np.array(w)-np.array(w0)
    b -= learningrate*G_b
    loss = s_loss/len(train)
    print loss
    return [w,b]

f = prediction(tem1,w,b)
        if f > 0:
            label = 1
        else:
            label = 0
```

為了使 f 帶入 log 裡後，數值不會爆掉。

Loss function:C(f(x),y)

Gradient of the loss function

更新 weight

預測值>0 : label=1

預測值<0 : label=0

二、Describe your another method, and which one is best.

使用 Naive Bayes Classifiers

```
#%%
def p(x,everage0,everage1,cov_matrix):

f=(gausian(x,everage0,cov_matrix)*pc0)/(gausian(x,everage0,cov_matrix)*pc0+gausi
an(x,everage1,cov_matrix)*pc1)
    return f

everage = train.groupby('label', as_index=False, group_keys=False,
sort=False).apply(everage)
everage0 = np.array(everage[0])
everage1 = np.array(everage[1])

cov_matrix0 = cov_matrix(train[(train.label==0)],everage0)
cov_matrix1 = cov_matrix(train[(train.label==1)],everage1)
cov_matrix = (2447*cov_matrix0 + 1554*cov_matrix1)/4001

pc0=2447/4001
pc1=1554/4001

f = p(x,everage0,everage1,cov_matrix)
    if f > 0.5:
        label = 0
    else:
        label = 1
```

→ Maximum likehood

計算相關參數
Mean, covariance matrix
P(c1),p(c2)

預測機率>0.5: label 0
預測機率<0.5: label 1

The performance of logistic regression is much better than naive bayes classifiers.

三、

Naive bayes classifiers doesn’t have the updating procedure to revise the decision boundaries. It only simulated the sample points as probability models, and predict the testing data sample through this probability model. On the other hand, logistic regression function make the update by calculate the gradient of the loss function. It gives a better decision in the end.

We need to be very careful to set initial value of the function. If we start with a big weight or bias, it will cause the output of the sigmoid function to be either near 1 or -1, and finally when we use the cross entropy to calculate the error, It will get infinite when we have ln(0).

In addition to setting the initial weight very small, I also processed the last few statistics of training data so that it could have the same order of magnitude as the other statistics , making it easier to design the initial weight and bias.