Import Library

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

Using Logistic Regression Algorithm from scrath

Using the Gradient Descent Algorithm for loss function

In [2]:

```
#Build The Class
class Logistic_Regression() :

    #Initiation
    def __init__(self) :
        self.learning_rate = 0.01
        self.epoch = 2000

    #Using Gradient Descent
    def update_weights(self) :
        A = 1/(1+np.exp(-(self.X.dot(self.W) + self.b)))

        tmp = (A-self.Y.T)
        tmp = np.reshape(tmp,self.m)
        dW = np.dot(self.X.T,tmp)/self.m
        db = np.sum(tmp)/self.m

        self.W = self.W - self.learning_rate * dW
        self.b = self.b - self.learning_rate * db

        return self

    def fit(self, X, Y) :

        self.m, self.n = X.shape

        self.W = np.zeros(self.n)
        self.b = 0
        self.X = X
        self.Y = Y

        for i in range(self.epoch) :
            self.update_weights()
        return self

    def predict(self, X) :
        Z = 1 / ( 1 + np.exp( - ( X.dot( self.W ) + self.b ) ) )
        Y = np.where( Z > 0.5, 1, 0 )
        return Y

#Import Dataset
ds = pd.read_excel("Dataset.xlsx")

#Feature Selection & Cleaning Dataset
ds = ds[['Rating','Total Reviews','Distance','Prediction']]
ds = ds.dropna()

#Define Feature and Target
feature = ds.iloc[:,:-1]
```

```
target = ds.iloc[:,-1:]

#Split Validation
X_train,X_test,y_train,y_test=train_test_split(feature,target,test_size=0.2,random_state
=42)

#Reshape Array
y_train = np.array(y_train)
y_train = y_train.reshape(-1)
y_test = np.array(y_test)
y_test = y_test.reshape(-1)

#Fit the model
model = Logistic_Regression()
model.fit(X_train, y_train)

#Predict validation
train_prediction = model.predict(X_train)
validation_prediction = model.predict(X_test)

#Get training accuracy function
def train_score(X):
    correct = 0
    count = 0
    for count in range(np.size(X)):

        if y_train[count] == X[count] :
            correct = correct + 1
        count = count + 1

    print( "Training Accuracy :", (correct / count) * 100 )

#Get testing accuracy function
def validation_score(X):
    correct = 0
    count = 0
    for count in range(np.size(X)):

        if y_test[count] == X[count] :
            correct = correct + 1
        count = count + 1

    print( "Validation Accuracy :", (correct / count) * 100 )

#Call the function
train_score(train_prediction)
validation_score(validation_prediction)
```

```
Training Accuracy : 90.85714285714286
Validation Accuracy : 90.9090909090909
```

Here we got accuracy above 90% both in training and testing set which able to classify models without problems such as overfit or underfit.