# Project Report

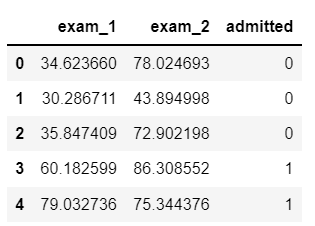
In this project, the data of student from their two examination is given, the dataset have a column called ‘admitted’ where the student has passed the exams and being promoted or not, I used Logistic Regression to predict that either the student is admitted or not. I have discussed each and everything along with the coding below. I have used Jupyter Notebook and Pandas for this project.

First I uploaded the dataset using pandas library.

df = pd.read\_csv('exams(1).csv')

df.head()

In the above code, I have made a Data frame called df, in this way our data from ‘exams(1).csv’ will be shown in tabular form as shown below.



After uploading the data into pandas, the next step is to make a graph as asked, from the code below, I have plotted a scatter plot.

X = df1['exam\_1']

Y = df1['exam\_2']

X1 = df2['exam\_1']

Y1 = df2['exam\_2']

plt.scatter(X,Y, c = "b", label = 'Passed')

plt.scatter(X1,Y1, c = "r", label = 'Failed')

plt.title('First Exam vs Second Exam')

plt.xlabel('First Exam Score')

plt.ylabel('Second Exam Score')

plt.legend()

plt.show()

I have copied the data from exam\_1 and exam\_2 columns to variable X and Y where the students are being admitted and variables X1 and Y1 contain all those students who are not admitted.

Then I plotted a scatter plot of both by giving the colors, blue dots are those students who are admitted and red are those who are not admitted.

# Logistic Regression from scratch

The next step was to perform Logistic Regression from scratch on the given data to predict either the student is admitted or not based on their scores from exam\_1 and exam\_1.

# Importing libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import warnings

warnings.filterwarnings( "ignore" )

First, I imported all the required libraries, pandas for importing the data, NumPy for arrays, model\_selection us to split the dataset.

# to compare our model's accuracy with sklearn model

from sklearn.linear\_model import LogisticRegression

# Logistic Regression

class LogitRegression() :

def \_\_init\_\_( self, learning\_rate, iterations ) :

self.learning\_rate = learning\_rate

self.iterations = iterations

# Function for model training

def fit( self, X, Y ) :

# no\_of\_training\_examples, no\_of\_features

self.m, self.n = X.shape

# weight initialization

self.W = np.zeros( self.n )

self.b = 0

self.X = X

self.Y = Y

# gradient descent learning

for i in range( self.iterations ) :

self.update\_weights()

return self

Then I made a class called LogisticRegression in which I defined some functions like the ‘fit’ function, this function is used to fit the data into the regression model to train on.

# Helper function to update weights in gradient descent

def update\_weights( self ) :

A = 1 / ( 1 + np.exp( - ( self.X.dot( self.W ) + self.b ) ) )

# calculate gradients

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

# update weights

self.W = self.W - self.learning\_rate \* dW

self.b = self.b - self.learning\_rate \* db

return self

# Hypothetical function h( x )

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

The weights are update after one iteration of every batch of data. For example, if you have 1000 samples and you set a batch size of 200, then the neural network's weights gets updated after every 200 samples. For this purpose, I designed another class called ‘update\_weights’ which will update the weight of the dataset.

# Driver code

def main() :

# Importing dataset

df = pd.read\_csv( "exams(1).csv" )

X = df.iloc[:,:-1].values

Y = df.iloc[:,-1:].values

# Splitting dataset into train and test set

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

X, Y, test\_size = 1/3, random\_state = 0 )

# Model training

model = LogitRegression( learning\_rate = 0.01, iterations = 1000 )

model.fit( X\_train, Y\_train )

model1 = LogisticRegression()

model1.fit( X\_train, Y\_train)

# Prediction on test set

Y\_pred = model.predict( X\_test )

Y\_pred1 = model1.predict( X\_test )

# measure performance

correctly\_classified = 0

correctly\_classified1 = 0

# counter

count = 0

for count in range( np.size( Y\_pred ) ) :

if Y\_test[count] == Y\_pred[count] :

correctly\_classified = correctly\_classified + 1

if Y\_test[count] == Y\_pred1[count] :

correctly\_classified1 = correctly\_classified1 + 1

count = count + 1

print( "Accuracy on test set by our model : ", (

correctly\_classified / count ) \* 100 )

print( "Accuracy on test set by sklearn model : ", (

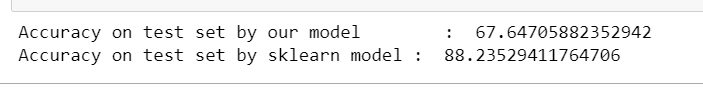
correctly\_classified1 / count ) \* 100 )

if \_\_name\_\_ == "\_\_main\_\_" :

main()

Now the main thing is going to happen, the Main class, I have build main class to call all the above function to perform the real prediction. I have split the dataset into test and train manually, then called the ‘fit’ function to train the model on the data, in the same way, I am training another model from sklearn library to see the comparison. Then I have called the predict function to do the predictions.

After predicting the data through models, I have printed the accuracy of the dataset. I got the following accuracy.



# Logistic Regression through Sklearn library

Now I will train the Logistic Regression model by using Sklearn library.

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

I have imported Logistic Regression from Sklearn library and imported the metrics.

X = df.drop(labels='admitted', axis=1)

Y = df['admitted']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,Y, test\_size=0.3, random\_state=0)

logreg = LogisticRegression()

logreg.fit(X\_train, y\_train)

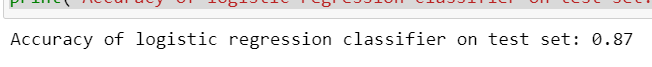
After importing the model, the next step is to split the data into test and train set, the train set will be used to train the model and test set will be used to test the model.

Then I called the fit function to fit the data into the model.

y\_pred = logreg.predict(X\_test)

print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X\_test, y\_test)))

Now by calling the predict function, I have done the predictions and printed the accuracy, I got 87% accuracy.



from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred))

After doing the training and testing and predictions, the last thing is to make a classification report, I have imported the classification report from sklearn metrics, then by using the print function, I have printed the classification report of the model.

