Linear Regression Lab Manual

Ahsan Saadat, PhD

1 Introduction

I have created this document to summarize my expectations from students by providing detailed aspects of how to perform the lab tasks. The purpose of this lab is to provide you with a practical learning experience, enabling you to gain firsthand knowledge of different artificial intelligence techniques and accomplish various objectives.

- Gain knowledge about various techniques and strategies for implementing logistic regression and its cost function.
- Read data from files and use it to train model.
- Develop a hands-on understanding of the inner working of logistic regression by building it from scratch.
- Understand how its cost function works.
- Implement its cost function from scratch.

2 Lab Setup

In this lab, the implementation will be done using Python language. Different IDEs can be used for implementing and executing the code. Following are some options.

- Google Colab (https://colab.research.google.com/) (Preferable)
- Jupyter Notebook (https://jupyter.org/)
- Visual Code Studio (https://code.visualstudio.com/)

2.1 Starting with Google Colab

This section is a guide for getting started with Google Colab.

- Sign into your Google account and open the Google Colab website.
- Create a new notebook from the drop-down menu after clicking the files in the menu bar.
- After creating a new notebook, write code in a cell and execute it by clicking the run button at the left side of the cell or by pressing the shift+enter keys.
- Rename your file to the name of the lab and download it as .ipynb file for future use.

3 Preliminary Concept

Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the independent variables and the dependent variable.

The goal of linear regression is to find the best-fit line that minimizes the difference between the predicted values and the actual values of the dependent variable. This best-fit line is determined by estimating the coefficients (slope and intercept) of the linear equation.

The equation for simple linear regression with one independent variable is:

$$y = b0 + b1 \times x \tag{1}$$

The cost function, also known as the loss function or the mean squared error (MSE), is used to measure the difference between the predicted values and the actual values. The goal is to minimize this difference. The cost function for linear regression is calculated as the average squared difference between the predicted and actual values:

$$Cost(b0, b1) = \frac{1}{n} \times \sum (y\_pred - y\_actual)^2 \tag{2}$$

4 Tasks

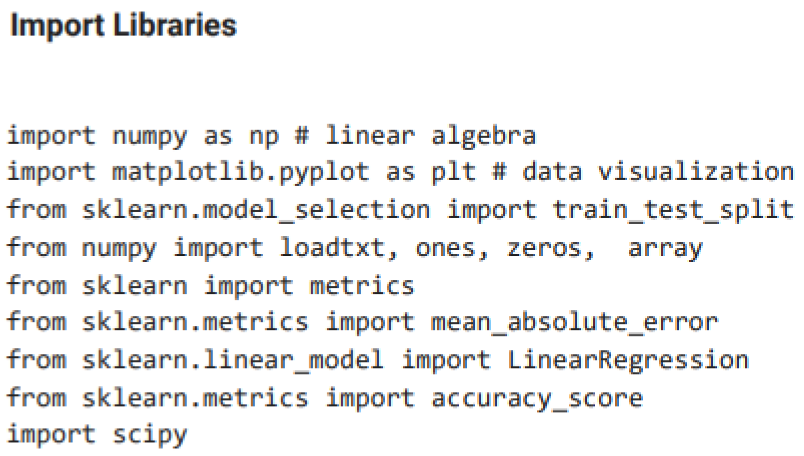
The following tasks shall be performed for achieving the goals of this lab:

- Implement binary linear regression model using libraries.
  - Use input data file and use it to train and test your model.
  - Divide the data set in training and testing sets - 60% train and 40% test.
  - Build linear regression model.

5 Libraries to Use

- import pandas as pd
- import numpy as np
- from mpl_toolkits import mplot3d
- import matplotlib.pyplot as plt

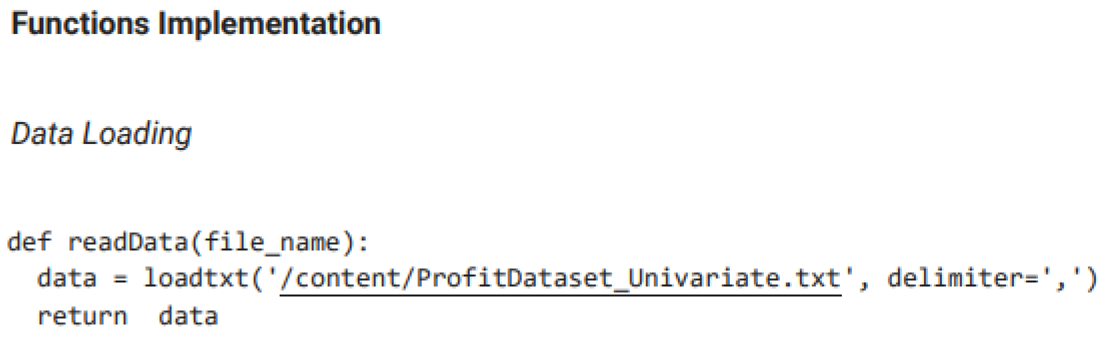
Figure 1: We are importing all the libraries that will be required for performing various tasks such as linear regression, data split, reading files, etc.



Import Libraries

```
import numpy as np # linear algebra
import matplotlib.pyplot as plt # data visualization
from sklearn.model_selection import train_test_split
from numpy import loadtxt, ones, zeros,  array
from sklearn import metrics
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
import scipy
```

Figure 2: A function for reading the txt file is implemented.

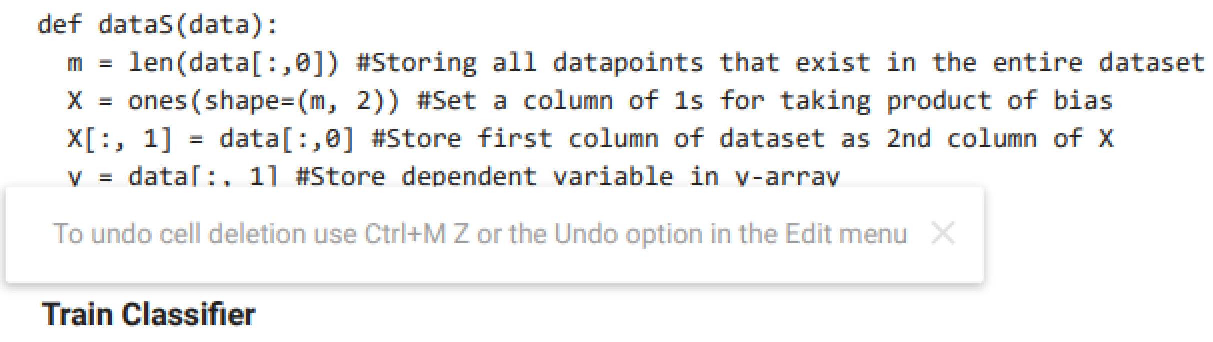


Functions Implementation

Data Loading

```
def readData(file_name):
  data = loadtxt('/content/ProfitDataset_Univariate.txt', delimiter=',')
  return  data
```

Figure 3: A function for adding the bias column in the dataset is implemented.

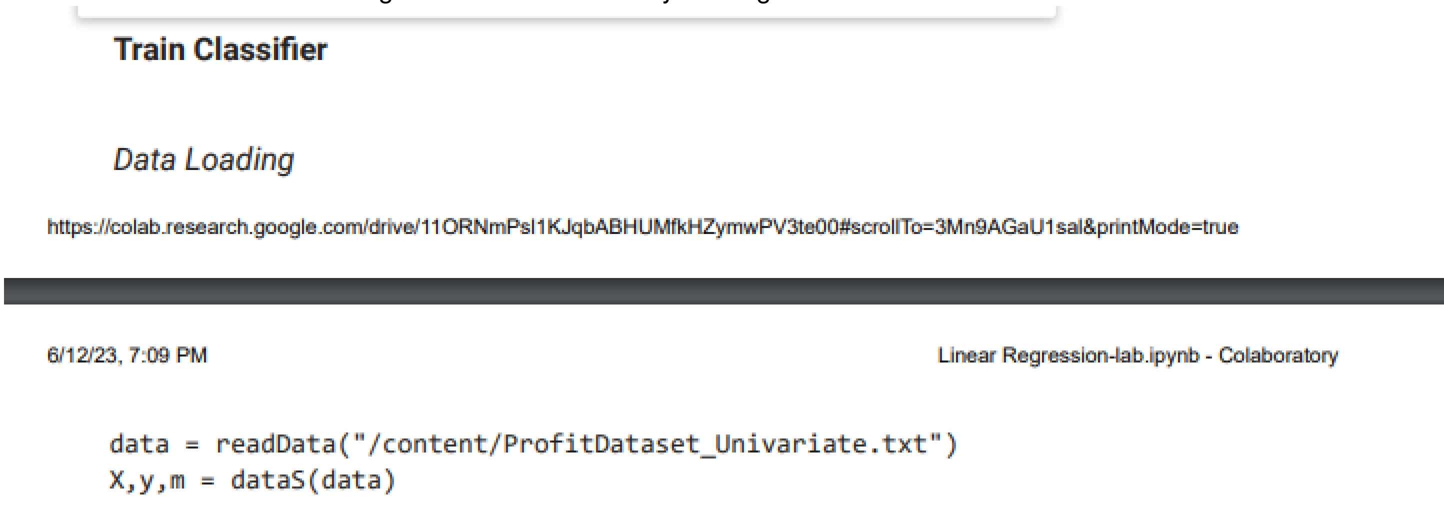


```
def dataS(data):
  m = len(data[:,0]) #Storing all datapoints that exist in the entire dataset
  X = ones(shape=(m, 2)) #Set a column of 1s for taking product of bias
  X[:, 1] = data[:,0] #Store first column of dataset as 2nd column of X
  y = data[:, 1] #Store dependent variable in y-array
```

To undo cell deletion use Ctrl+M Z or the Undo option in the Edit menu

Train Classifier

Figure 4: Data is loaded by reading the txt file.



Train Classifier

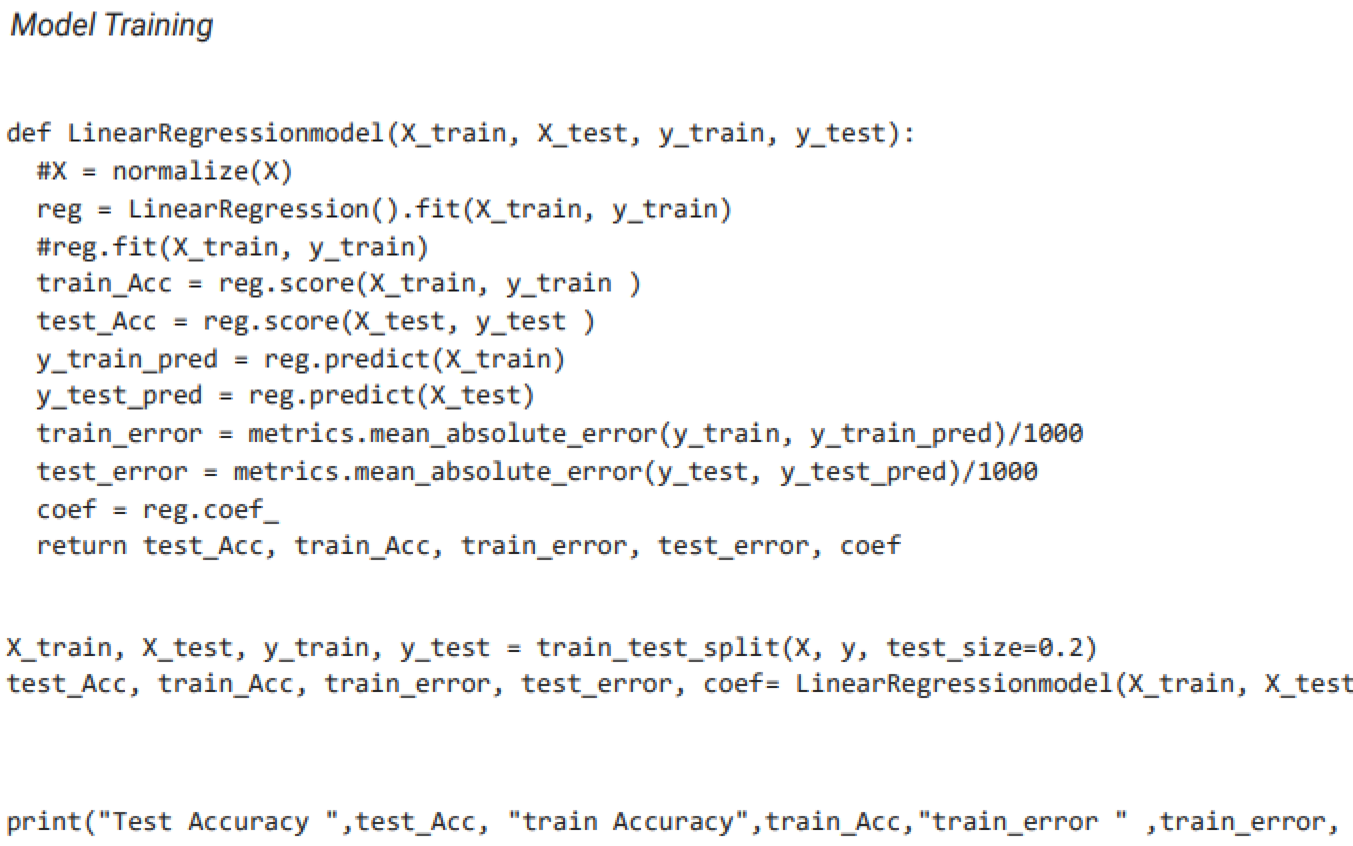
Data Loading

https://colab.research.google.com/drive/11ORNmPsl1KJqbABHUMfkHZymwPV3te00#scrollTo=3Mn9AGaU1sal&printMode=true

6/12/23, 7:09 PM Linear Regression-lab.ipynb - Colaboratory

```
data = readData("/content/ProfitDataset_Univariate.txt")
X,y,m = dataS(data)
```

Figure 5: The linear regression model is created for training and testing the data.



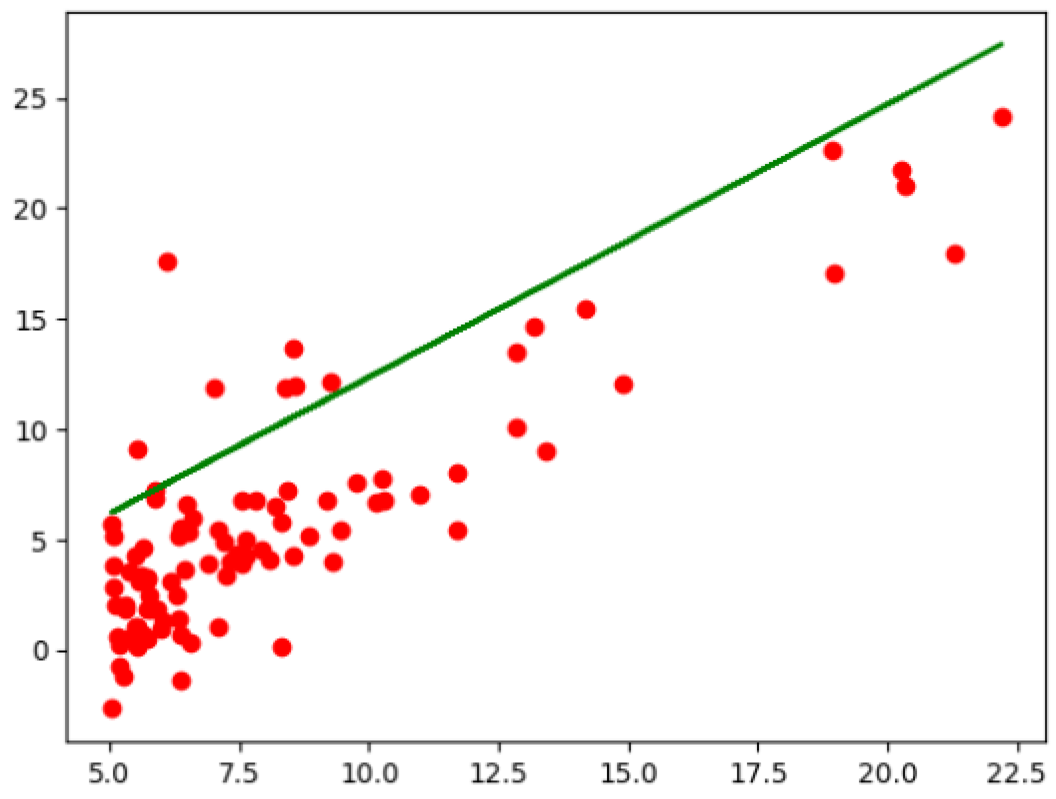
Model Training

```
def LinearRegressionmodel(X_train, X_test, y_train, y_test):
  #X = normalize(X)
  reg = LinearRegression().fit(X_train, y_train)
  #reg.fit(X_train, y_train)
  train_Acc = reg.score(X_train, y_train )
  test_Acc = reg.score(X_test, y_test )
  y_train_pred = reg.predict(X_train)
  y_test_pred = reg.predict(X_test)
  train_error = metrics.mean_absolute_error(y_train, y_train_pred)/1000
  test_error = metrics.mean_absolute_error(y_test, y_test_pred)/1000
  coef = reg.coef_
  return test_Acc, train_Acc, train_error, test_error, coef


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
test_Acc, train_Acc, train_error, test_error, coef= LinearRegressionmodel(X_train, X_test



print("Test Accuracy ",test_Acc, "train Accuracy",train_Acc,"train_error " ,train_error,
```

Figure 6: The results are plotted by plotting a linear relation between the data points. using matplot library.

Train Scores and Test Scores

```
result = X.dot(coef).flatten()
plt.plot(X[:, 1], result, c = 'g')
plt.scatter(X[:,1], y, c = 'r')
plt.show()
```



25
20
15
10
5
0
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5