CS7602 - MACHINE LEARNING

ASSIGNMENT 1

## SUBMITTED BY

JAYASREE LAKSHMI NARAYAN 2016103033

AKHILA G P 2016103503

DATE : 28-01-2019

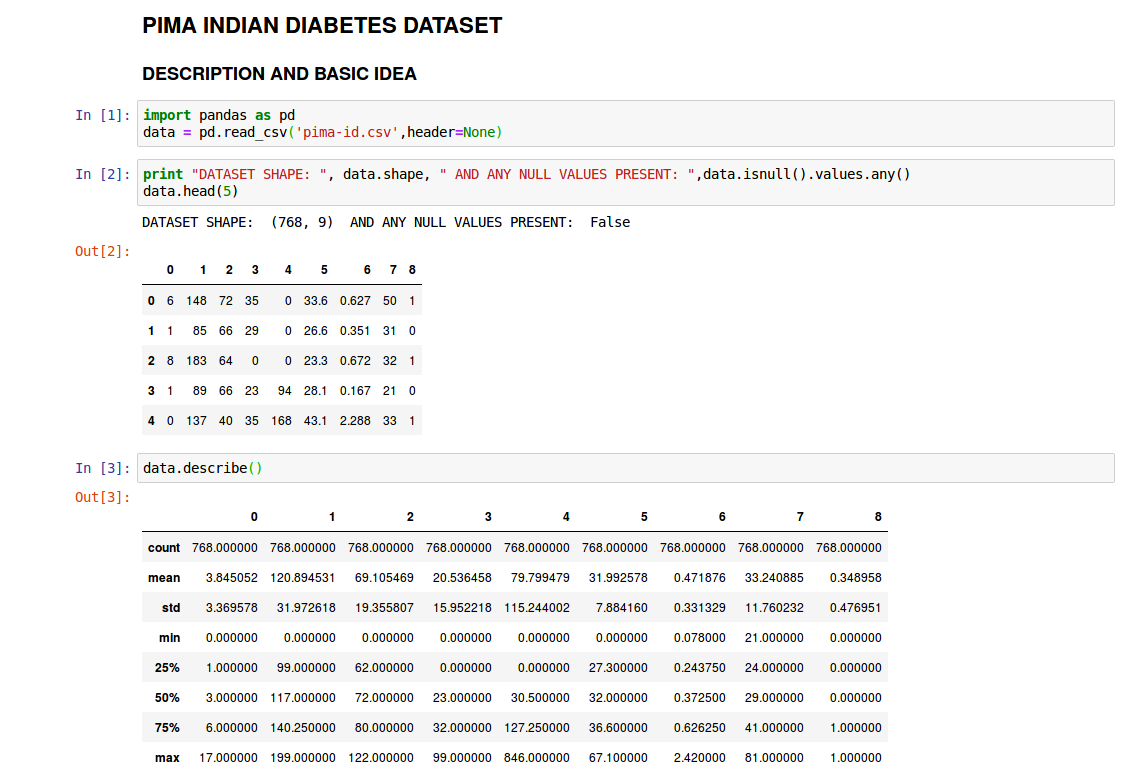
### CONTENTS

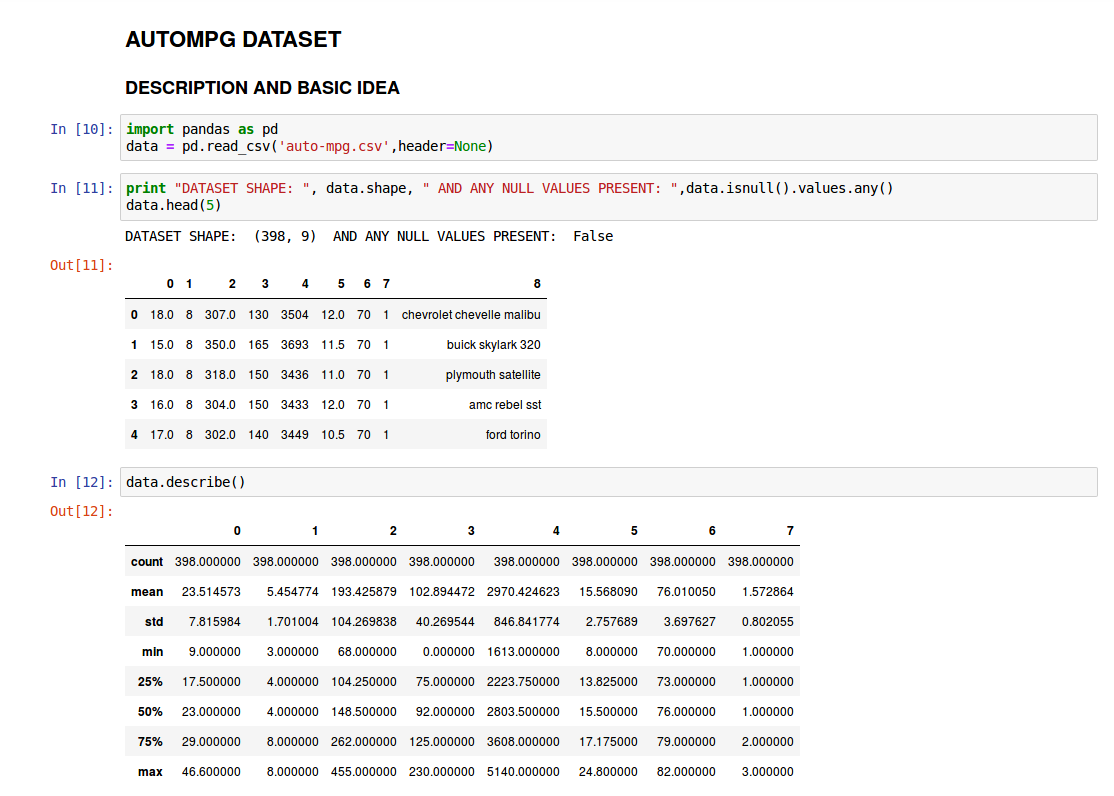
1. SINGLE PERCEPTRON
2. MULTI LAYER PERCEPTRON
3. LINEAR REGRESSION (WITH SINGLE VARIABLE)

### DATASETS USED

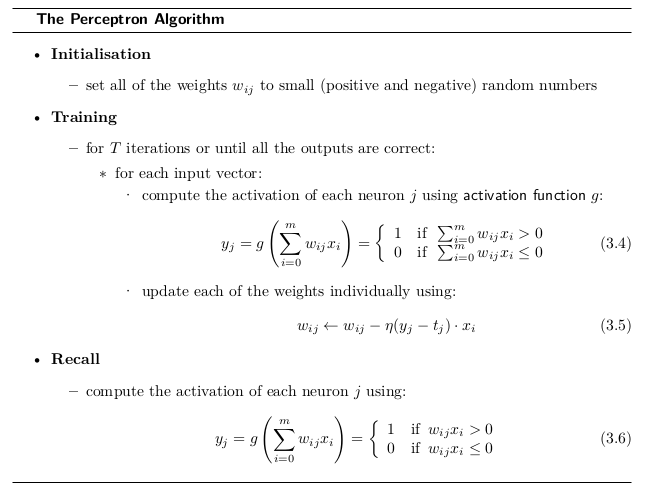
1. PIMA INDIAN DIABETES DATASET (CLASSIFICATION)
2. AUTO-MPG DATASET (REGRESSION)

# A DESCRIPTION ON THE DATASET UNDER STUDY





1. **PERCEPTRON**



The jupyter notebook with the code is uploaded in Github and the link for the document is <https://github.com/Akhilagp/ML_Assignment>.

**PROCEDURE:**

* The perceptron is based on activation and threshold concept.
* A neuron fires when the output of the activation function is above the threshold set.
* It has a single layer of neurons with random weights attached to it.
* PARAMETERS VARIED For Understanding

1. Learning rate
2. Number of Iterations

**INFERENCE:**

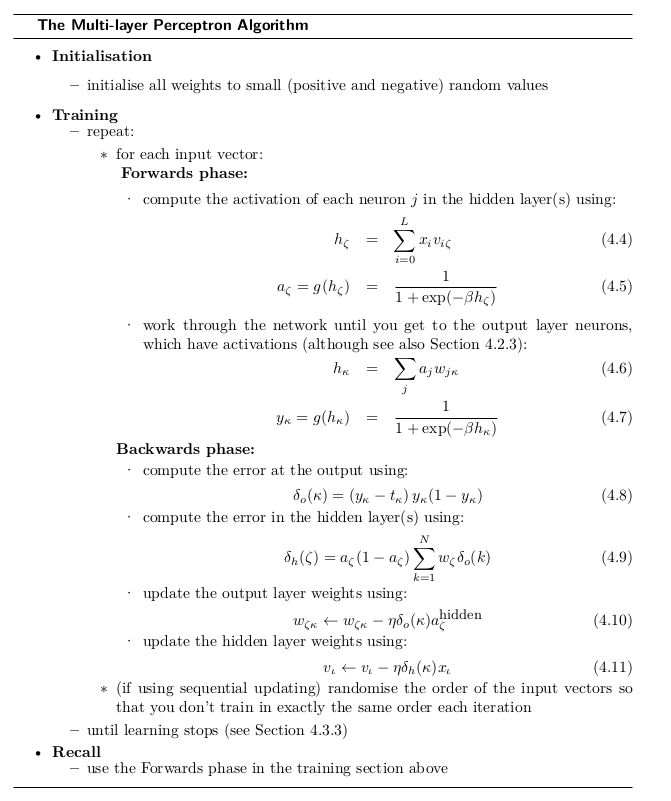
* The perceptron does well on the training set of the pima dataset, when the number of iterations are higher for a particular learning rate.
* A nominal learning rate produces a good result on the preprocessed set.

**OUTPUT:**

|  |  |  |
| --- | --- | --- |
| Learning rate | Number of Iterations | Accuracy |
| 0.01 | 100 | 0.6197916667 |
| 500 | 0.7057291667 |
| 1000 | 0.703125 |
| 2000 | 0.6979166667 |
| 0.03 | 100 | 0.6276041667 |
| 500 | 0.7083333333 |
| 1000 | 0.703125 |
| 2000 | 0.7083333333 |
| 0.1 | 100 | 0.6380208333 |
| 500 | 0.6770833333 |
| 1000 | 0.671875 |
| 2000 | 0.6770833333 |
| 0.25 | 100 | 0.6432291667 |
| 500 | 0.7213541667 |
| 1000 | 0.6979166667 |
| 2000 | 0.7083333333 |
| 0.3 | 100 | 0.7213541667 |
| 500 | 0.7135416667 |
| 1000 | 0.7083333333 |
| 2000 | 0.671875 |

A learning rate of 0.25 and 500 iterations was the highest recorded accuracy for the particular run. By testing the algorithm, an accuracy of 78% was achieved.

2. **MULTI LAYER PERCEPTRON**



The jupyter notebook with the code is uploaded in Github and the link for the document is <https://github.com/Akhilagp/ML_Assignment>.

**PROCEDURE :**

* Ten nodes were used in the hidden layer.
* Running a logistic function, on the training data, ouputs were obtained and tabulated.
* The dataset was split into training set (50%), validation set (20%) and test set (30%).
* PARAMETERS VARIED For a Deeper Insight

1. Learning rate (eta)
2. Number of Iterations

**INFERENCE:**

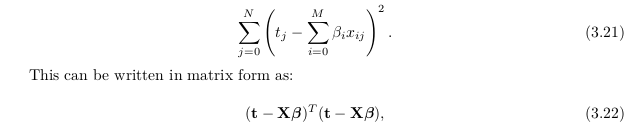
* Higher the learning rate, converging of the descent is not proper and the error seems to increase or stay stable.
* With lower learning rate(<0.1), accuracy is high and loss is minimized.
* Increasing the hidden nodes from 5 to 10 seem to increase the accuracy of the classifier.

**OUTPUT:**

To support the inferences made, the algorithm was run for different learning rates (0.001 < eta < 0.9) for different iterations ( 1000 < it < 9000) . The accuracy and loss for each variation is tabulated below

|  |  |  |  |
| --- | --- | --- | --- |
| Learning rate | Number of Iterations | Accuracy | Error |
| 0.001 | 1000 | 88.5416666667 | 18.6579579272 |
| 2500 | 88.5416666667 | 17.8883626083 |
| 5000 | 89.84375 | 16.8562355692 |
| 0.003 | 1000 | 90.625 | 16.2648364351 |
| 2500 | 90.8854166667 | 15.0400030387 |
| 5000 | 92.96875 | 13.2560325229 |
| 0.01 | 1000 | 94.2708333333 | 12.0870142879 |
| 2500 | 95.0520833333 | 10.6999256481 |
| 5000 | 95.3125 | 9.2779888677 |
| 0.03 | 1000 | 94.53125 | 10.5823039961 |
| 2500 | 92.4479166667 | 12.8231271803 |
| 5000 | 95.33 | 8.9968897062 |
| 0.1 | 1000 | 77.6041666667 | 39.5647531552 |
| 2500 | 83.8541666667 | 29.0762649929 |
| 5000 | 80.9895833333 | 29.8457619597 |
| 0.3 | 1000 | 74.21875 | 47.2384863936 |
| 2500 | 68.78 | 59.9290039822 |
| 5000 | 68.75 | 59.8750835422 |

The row corresponding to learning rate 0.03 and 5000 iteration shows minimum error and maximum accuracy. As the learning rate increases, the dataset gets over-fitted leading to a increasing value of error. The algorithm on test set produced an accuracy of 71-75%.



3. **LINEAR REGRESSION**

**Linear regression** is a **linear** approach to modeling the relationship between a dependent variable and one or more independent variables.

The Error in a linear regression is calculated as follows

The Code is uploaded in Github and the link is <https://github.com/Akhilagp/ML_Assignment>.

The weights can be adjusted by the following formula

weights = (XTX)-1XTy

**PROCEDURE:**

* The Auto-mpg dataset is split into training(80%) and test sets(20%) and the regression is carried out on the input features.
* The features considered were

1. Dependent variable: miles per gallon (mpg)
2. Independent variables: cylinders, displacement and horsepower

* The data is normalized and split.
* The gradient and the intercept for the calculation of the decision boundary line is obtained from stats module

**gradient, intercept, r\_value, p\_value, std\_err = stats.linregress(xtrain,ytrain)**

* The gradient turns out to be negative implying the negative co-relation between the variables taken.
* PARAMETERS VARIED For Insight:

1. Split size of Training and testing
2. Independent variables taken for Linear Regression

**INFERENCES:**

The value of cost function/ error is computed and is found to be in powers of -26. The theta / weights matrix returned will be a column vector.

|  |  |
| --- | --- |
|  |  |
| **Training**  Cylinders vs mpg | **Testing**  Cylinders vs mpg |
|  |  |
| **Training**  Displacement vs mpg | **Testing**  Displacement vs mpg |
|  |  |
| **Training**  Horsepower vs mpg | **Testing**  Horsepower vs mpg |

