Deep Learning 2019

Assignment # 1

Report

Topic: Linear Regression

➢ Question 1:

Data pre-processing and visualization:

NOTE: After preprocessing I created a csv **'A01_afterPreProcessing_googleplaystore.csv'** and submitted it on google classroom along with code.

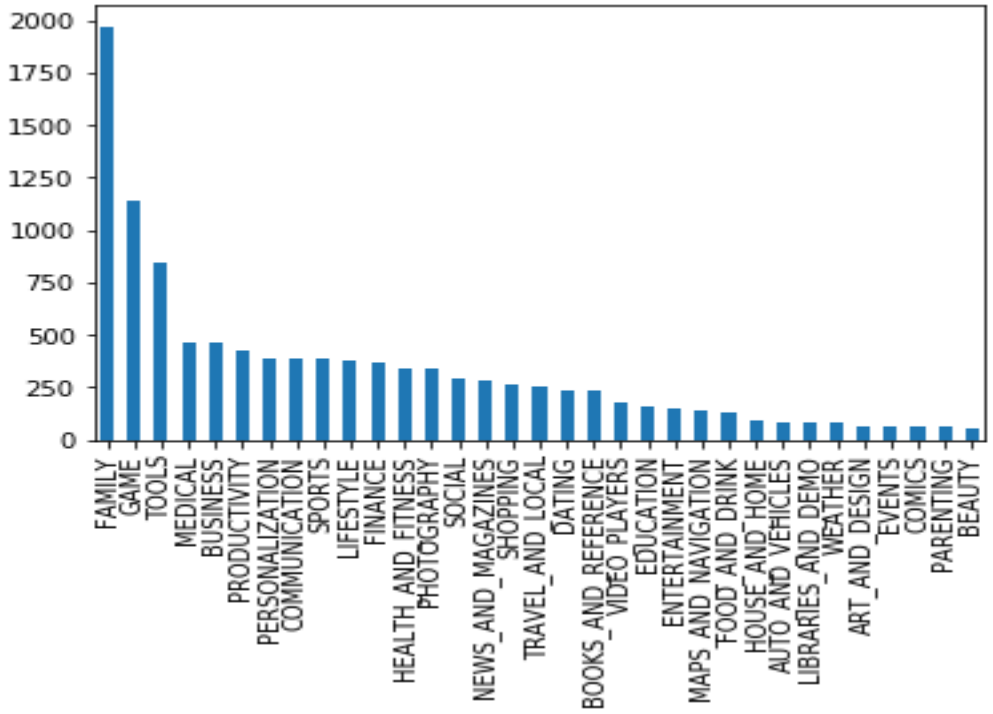
a. Column names:

```
['App' 'Category' 'Rating' 'Reviews' 'Size' 'Installs' 'Type' 'Price'
 'Content Rating' 'Genres' 'Last Updated' 'Current Ver' 'Android Ver']
```

b. Profit calculation for android version 4.3 and below:
I calculated profits and added a new column with name **'Profit_for4.3nBelow'** in the data frame. You can see it in final csv file.

c. Fill the missing values of ratings:
I've create a new csv file after doing all the pre-processing, the missing values in the original csv file have been filled in the new csv file.
Assumption: If more than one consecutive values are missing, to fill the rating of ith entry I've filled the $(i-1)^{st}$ and $(i+1)^{st}$ value by the rating value $(i-2)^{nd}$ entry.

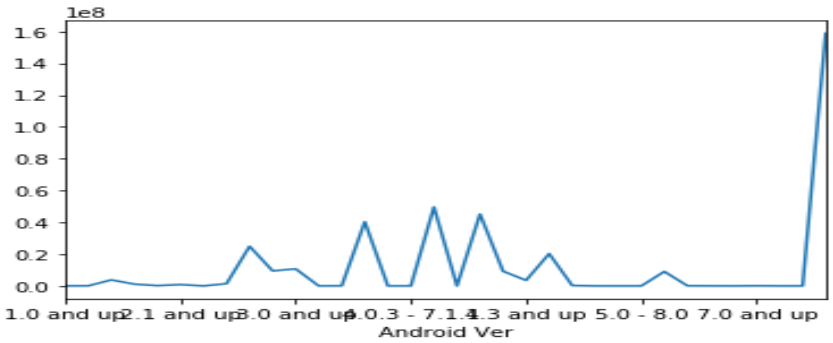
d. Histogram graph of total count of each category in the dataset:



e. A graph of profit generated by each android version:
I calculated profit and added a new column **'Profit_byAndroidVersion'** in the data frame and used these values to plot graph. You can see it in final csv file.

Android Ver	**Profit**
1.0 and up	0.000000e+00

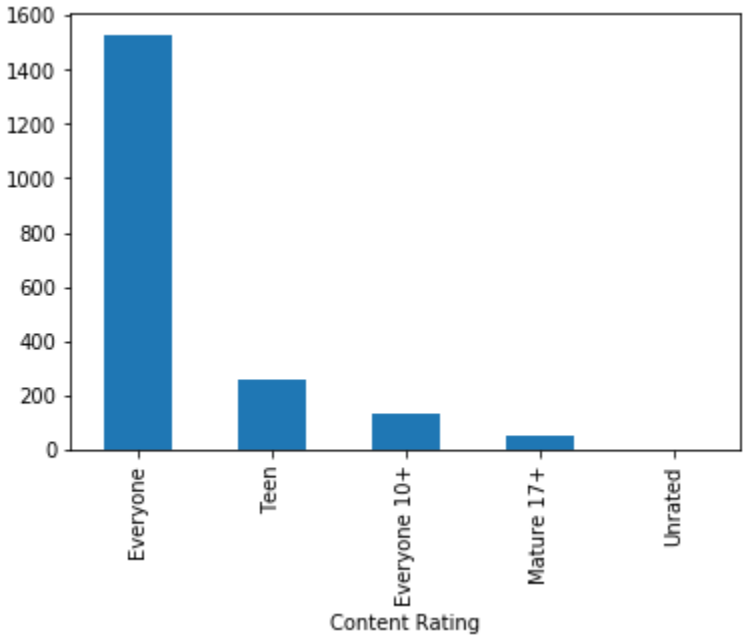
```
1.5 and up              6.182350e+04
1.6 and up              3.768736e+06
2.0 and up              1.109009e+06
2.0.1 and up            1.894244e+05
2.1 and up              8.324022e+05
2.2 - 7.1.1             0.000000e+00
2.2 and up              1.398355e+06
2.3 and up              2.488614e+07
2.3.3 and up            9.442051e+06
3.0 and up              1.070706e+07
3.1 and up              4.950000e+00
3.2 and up              6.139385e+04
4.0 and up              4.028438e+07
4.0.0                   1.490000e+03
4.0.3 - 7.1.1           0.000000e+00
4.0.3 and up            4.959422e+07
4.1 - 7.1.1             0.000000e+00
4.1 and up              4.513077e+07
4.2 and up              9.224184e+06
4.3 and up              3.492658e+06
4.4 and up              2.024084e+07
4.4W and up             2.497900e+05
5.0 - 6.0               0.000000e+00
5.0 - 7.1.1             0.000000e+00
5.0 - 8.0               0.000000e+00
5.0 and up              9.015109e+06
5.1 and up              7.494900e+04
6.0 and up              2.017529e+04
7.0 - 7.1.1             0.000000e+00
7.0 and up              7.884400e+04
7.1 and up              0.000000e+00
8.0 and up              1.490000e+02
Varies with device      1.587655e+08
Name: Profit_byAndroidVersion, dtype: float64
```



f. Graph of Number of installation which belongs to family category based on content rating:

Content Rating	Installations
Everyone	1529
Teen	261

```
Everyone 10+     131
Mature 17+        50
Unrated            1
Name: Content Rating, dtype: int64
```



➢ Question 2:

Part 2.a:

Linear Regression (without gradient descent):

In this problem we have to perform linear regression i.e. we have to fit a line on the given datasets (which is divided into training and testing portion). We are required to measure the performance of our system using mean squared error.

Least Square Method:

Our goal is to fit a line:

$$y = m * x + c$$

m (slope of regression line) and c (intercept of regression line) are our parameters that we need to find using the given datasets. I found these by the formulas given on the link (mentioned in the problem statement, by you).

$$m = \frac{Cov\ (x, y)}{Var\ (x)}$$

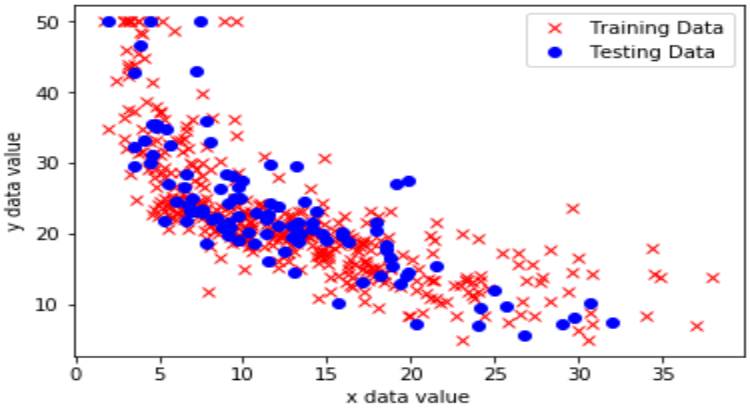
$$c = \bar{y} - m * \bar{x}$$

Here x is the training data point, y is its label.
x = x values of all the training data points
y = y values of all the training data points.
$\bar{x}$ = mean of x values of all the training data points
$\bar{y}$ = mean of y values of all the training data points.

Results:

Dataset 1:

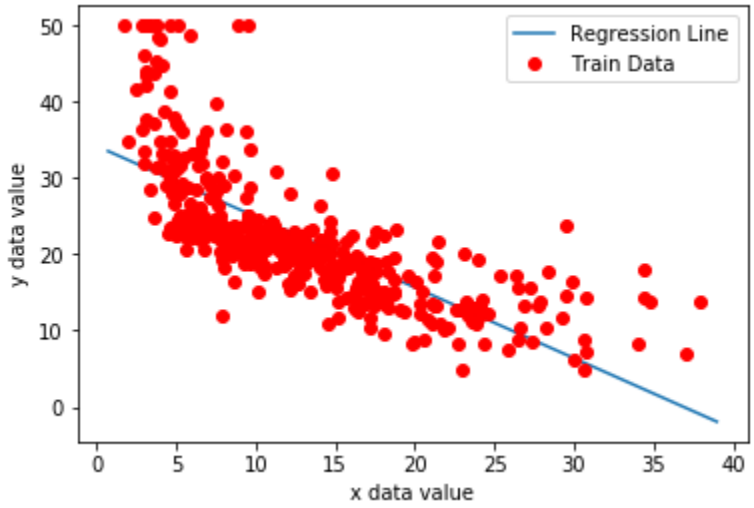
Plot of Training & Testing data:



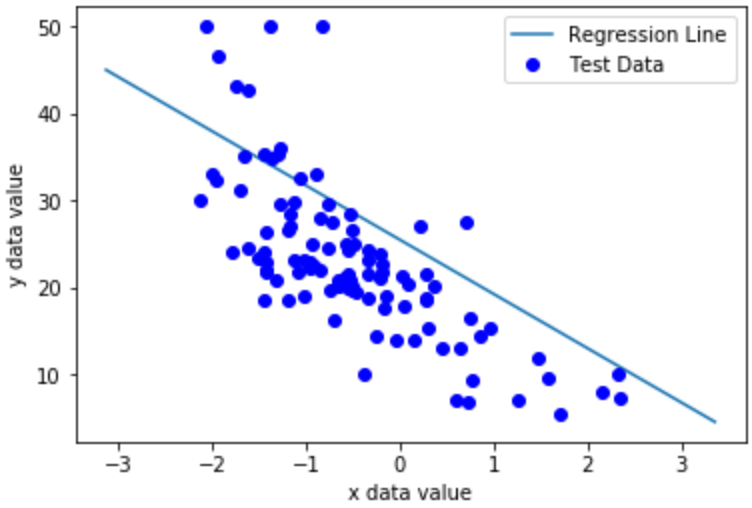
M	c
-0.9278216902292562	34.216255712624125

Training Error	Testing Error
39.429056363858884	34.87619673683118

Plot of Regression Line and Training Data:

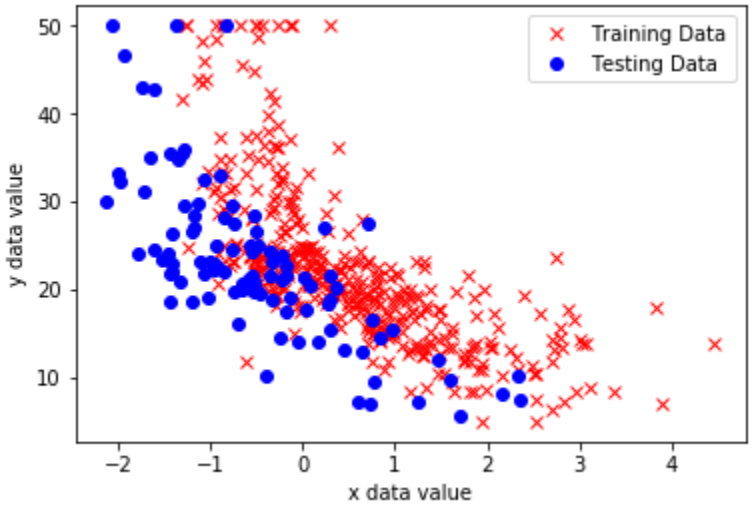


Plot of Regression Line and Testing Data:



Dataset 2:

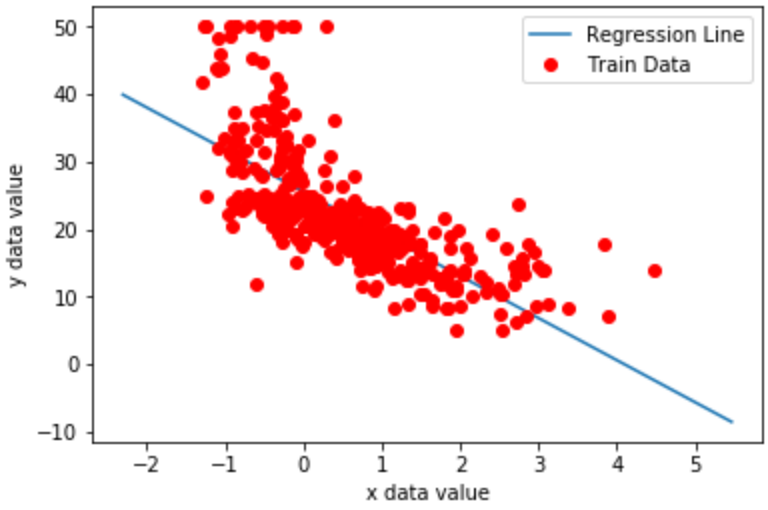
Plot of Training & Testing data:



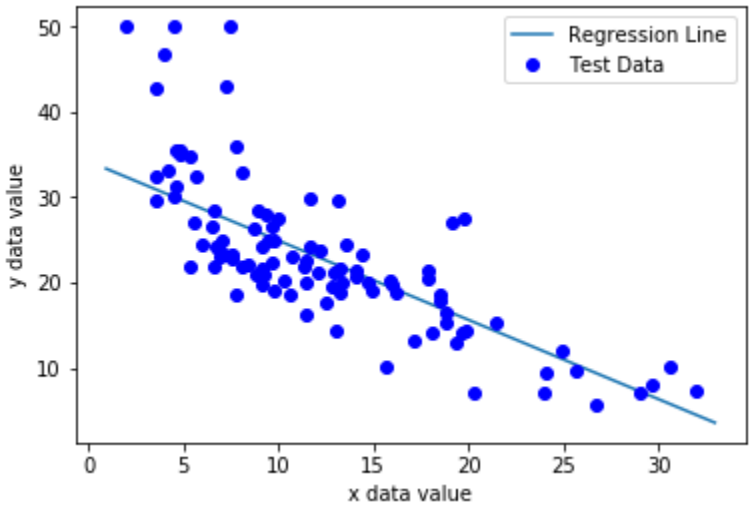
M	c
-6.238131127132846	25.5263621729055

Training Error	Testing Error
42.84019807969878	70.80834523792943

Plot of Regression Line and Training Data:



Plot of Regression Line and Testing Data:



Part 2.b:

Linear Regression (with gradient descent):

In this part, we are required to implement linear regression using (stochastic) gradient descent. Again we have to fit a line on the given datasets:

$$y = m * x + c$$

But now we'll find the values of m and c using gradient descent. Gradient descent updates the values of parameters by differentiating cost function with respect to the parameters and equating to zero i.e. minimizing the cost function:

$$J\,(m, c) = \frac{1}{2 * n}\sum_{i=1}^{n}\left(\left(m * x^i + c\right) - y^i\right)^2$$

To update m and c we'll differentiate cost function with respect to m and c one by one. This gives gradient of cost function with respect to m and c respectively. In case of **Stochastic Gradient Descent,** we update values of m and c on every sample. So, for every sample, gradients of m and c are given as:

$$\frac{\partial \mathrm{J}}{\partial \mathrm{m}} = \left(\left(m * x^i + c\right) - y^i\right) * \; x^i$$

$$\frac{\partial \mathrm{J}}{\partial \mathrm{c}} = \left(\left(m * x^i + c\right) - y^i\right)$$

And then at every sample we'll update m and c as follows:

$$m = m - alpha * \frac{\partial \mathrm{J}}{\partial \mathrm{m}}$$

$$c = c - alpha * \frac{\partial \mathrm{J}}{\partial \mathrm{m}}$$

Here alpha is the learning rate which governs how slowly or quickly we move along the gradient.

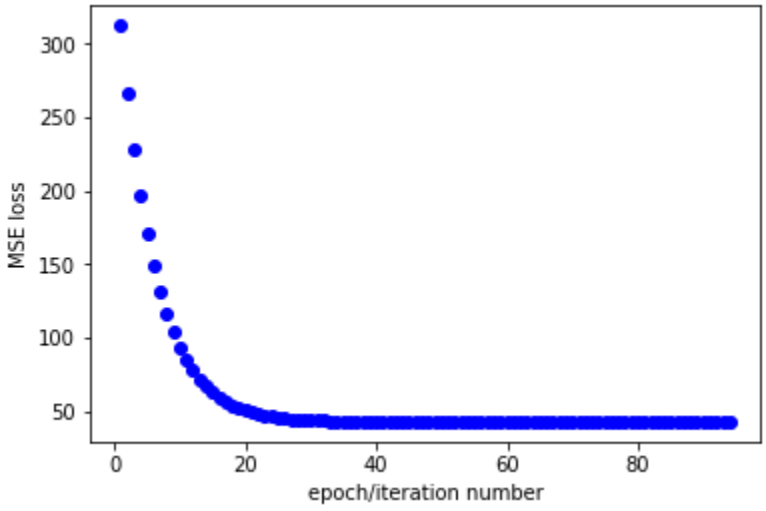
Results:

Dataset 1:

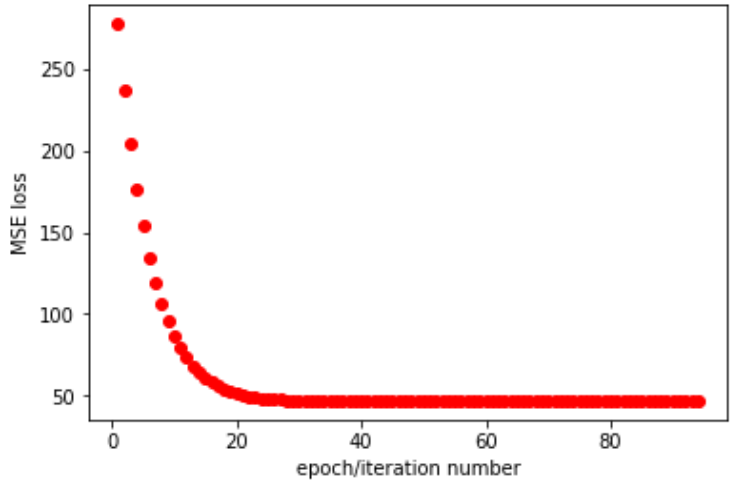
alpha = 0.001
m_sgd = 0
c_sgd = 0
epochs = 500

The loss converged at 90th epoch for a difference value of '0.0001' at a loss value of 43.2. I've plotted two types of MSE loss here.

1) **Plot of mean square error in each iteration:**
   During one epoch, I calculated MSE loss for each sample (while updating the **m** and **c** parameters for each sample) and kept on adding the values and then divided by the number of samples at the end.



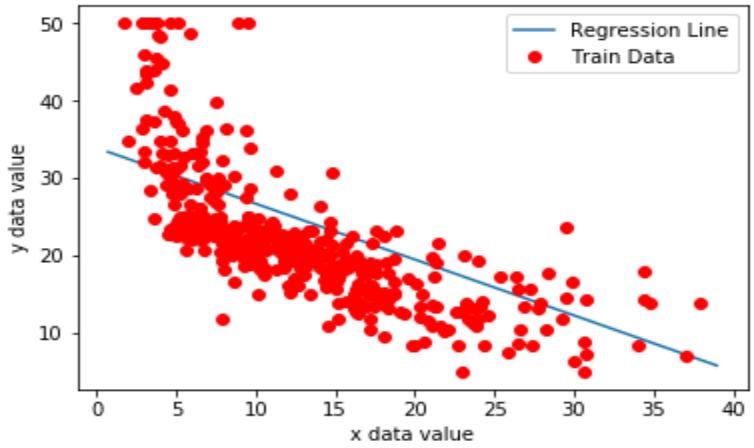
2) **Plot of mean square error in each iteration:**
   At the end of the epoch, when all samples have been viewed once, then I calculated MSE loss of entire training set with one value of **m** and **c** parameters.



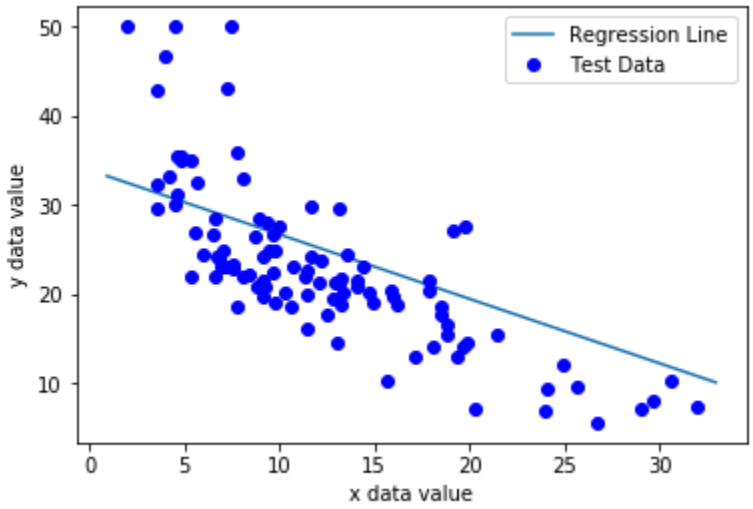
m	c
-0.7214360638168078	33.911381327506305

Training Error	Testing Error
47.069200592910576	42.788472086549035

Plot of Regression Line and Training Data:



Plot of Regression Line and Testing Data:

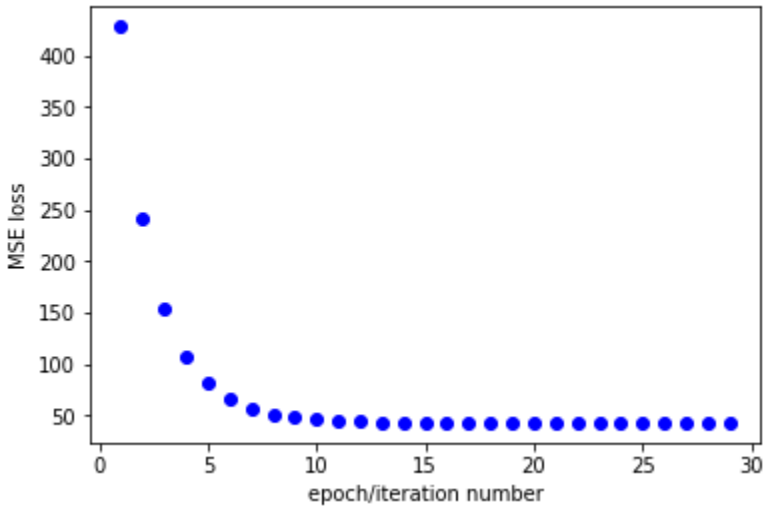


Dataset 2:

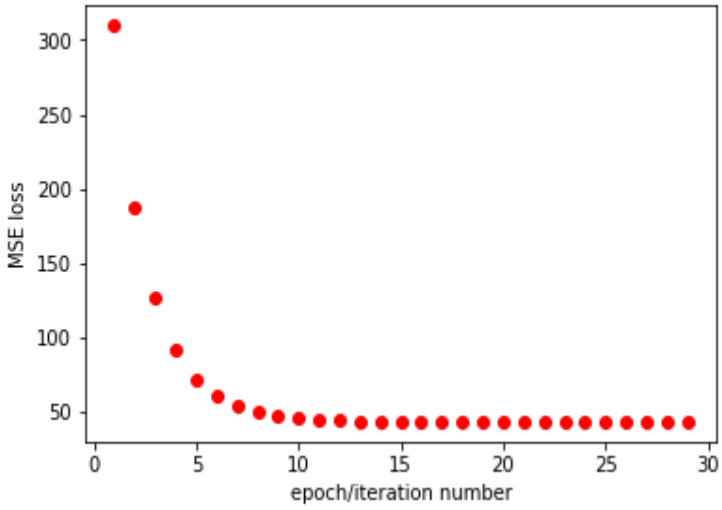
alpha = 0.001
m_sgd = 0
c_sgd = 0
epochs = 500

The loss converged at 28th epoch for a difference value of '0.0001' at a loss value of 42.9. I've plotted two types of MSE loss here.

1) **Plot of mean square error in each iteration:**
   During one epoch, I calculated MSE loss for each sample (while updating the **m** and **c** parameters for each sample) and kept on adding the values and then divided by the number of samples at the end.



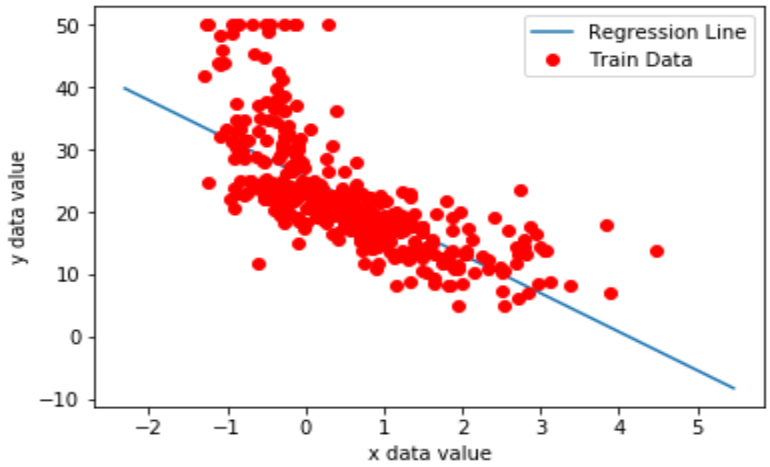
2) **Plot of mean square error in each iteration:**
   At the end of the epoch, when all samples have been viewed once, then I calculated MSE loss of entire training set with one value of **m** and **c** parameters.



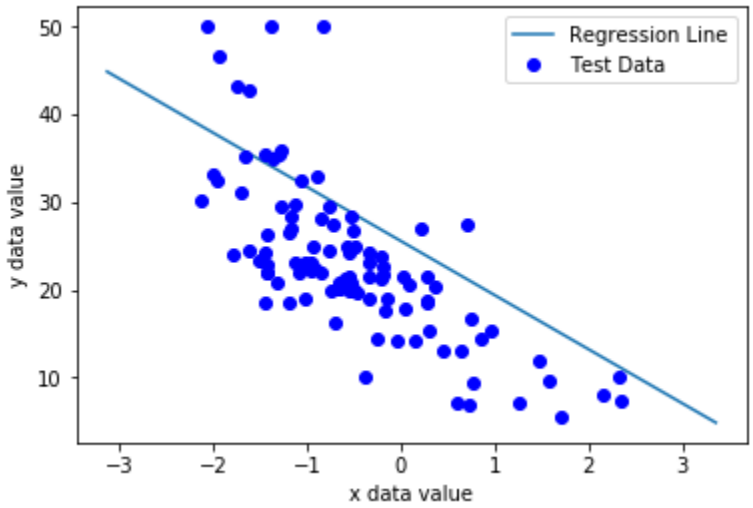
m	c
-6.177648686808649	25.568209674595693

Training Error	Testing Error
42.84933970760121	71.01054571958036

Plot of Regression Line and Training Data:



Plot of Regression Line and Testing Data:



➢Question 3:

For a 10 dimensional dataset, what is the minimum number of points that are required to fit a hyperplane?

The hyperplane of 2-D space is in 1-D. In 1-D we have line, which requires (minimum) two points to fit. These two points shouldn't be same.

The hyperplane of 3-D space is in 2-D. In 2-D we have plane, which requires (minimum) three points to fit. These three points shouldn't be collinear.

The hyperplane of 4-D space is in 3-D. We'll need (minimum) four points to fit this hyperplane. These three points shouldn't be coplanar.

.

.

.

The hyperplane of n-D space is in (n-1)-D and it'll require (minimum) n points to be defined.

Following the above argument, the hyperplane of a 10-D space will be in 9-D. (minimum) 10 points will be required to define this hyperplane.

What is the difference between deep and shallow learning? Explain with concrete example(s) when shallow learning is suitable as compared to deep learning and vice versa.

Shallow learning:

A shallow neural network is the one having only one hidden layer. Shallow learning is the machine learning in which hand crafted features are used (mostly) and major focus is on training the prediction model. It is less cheap (as no requirement of costly hardware) and requires less computational time. Shallow learning algorithms are easier to understand.

When to use:

1. Should be used when data is less.
2. When problem is specific and can be solved with hand-crafted features with high accuracy e.g. gender detection from audio. It can be easily solved by extracting features such as pitch, fundamental frequency (fundamental frequency range of males and females is different).

Deep Learning:

A deep neural network is the one having more than one hidden layer. Deep Learning learns representations directly from raw data mostly. Deep learning is the type of machine learning in which there is a major focus on learning representation of data, i.e. it focuses on learning the features on its own using complex and deep (layer-wise) functions, hence also called representational or hierarchical learning. It requires lots of data. In many image and speech related task, background noise, variation in orientation, position, illumination change the data and hand crafted features may not be useful.

When to use:

1. When lots of data is available.
2. When hand-crafted features give poor performance.
3. When there are complex relations between features, a deep network unfolds/learns these relations in many layers. E.g. To identify the images of White Wolf and Samoyed (a white dog breed) in different poses and environments, simple image features are not enough as it is very difficult to differentiate a White Wolf and Samoyed in a same pose. So in such a problem deep learning should be used to extract more discriminative features.

Write summary of the paper "Deep Learning".

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

(LeCun, 2015), has given a brief review of Deep Learning and how it has revolutionized the fields of speech, image, medicine etc. Deep Learning networks learn the complex structures present in the data using back propagation algorithm. Deep Networks have multiple hidden layers which make it possible to learn the most discriminative hidden features from raw data, thus surpassing, in performance, the traditional machine learning, success of whom is dependent on hand crafted features. Fixed size input to output mapping is done through feed forward network is used to do fixed sized input to output mapping using the non-linear activation function ReLU which is comparatively faster. Hidden layers change the input non-linearly so that output layer can separate the categories linearly.

Convolutional Neural Networks (CNNs) comprise of convolutional and pooling layers which detect features and reduce feature space by merging similar features, respectively. CNNs have achieved great success in image related tasks such as object detection, face recognition, segmentation.
Recurrent Neural Networks (RNNs) have been proved very successful for sequential data such as speech and language. A variation of RNN called LSTM has special hidden units and can learn long-term dependencies of a sequence.
So far major success of deep networks is in supervised learning but in future focus will shift to unsupervised learning. Combination of deep learning with reinforcement learning and simple reasoning is expected to further revolutionize the field of Artificial Intelligence.

➢References:

https://www.coursera.org/lecture/machine-learning/gradient-descent-8SpIM

https://www.quora.com/2-points-define-a-line-3-points-a-plane-What-do-4-points-define

http://www.deeplearningbook.org/contents/ml.html