Name - Sagar Suman

Roll No. 2019197

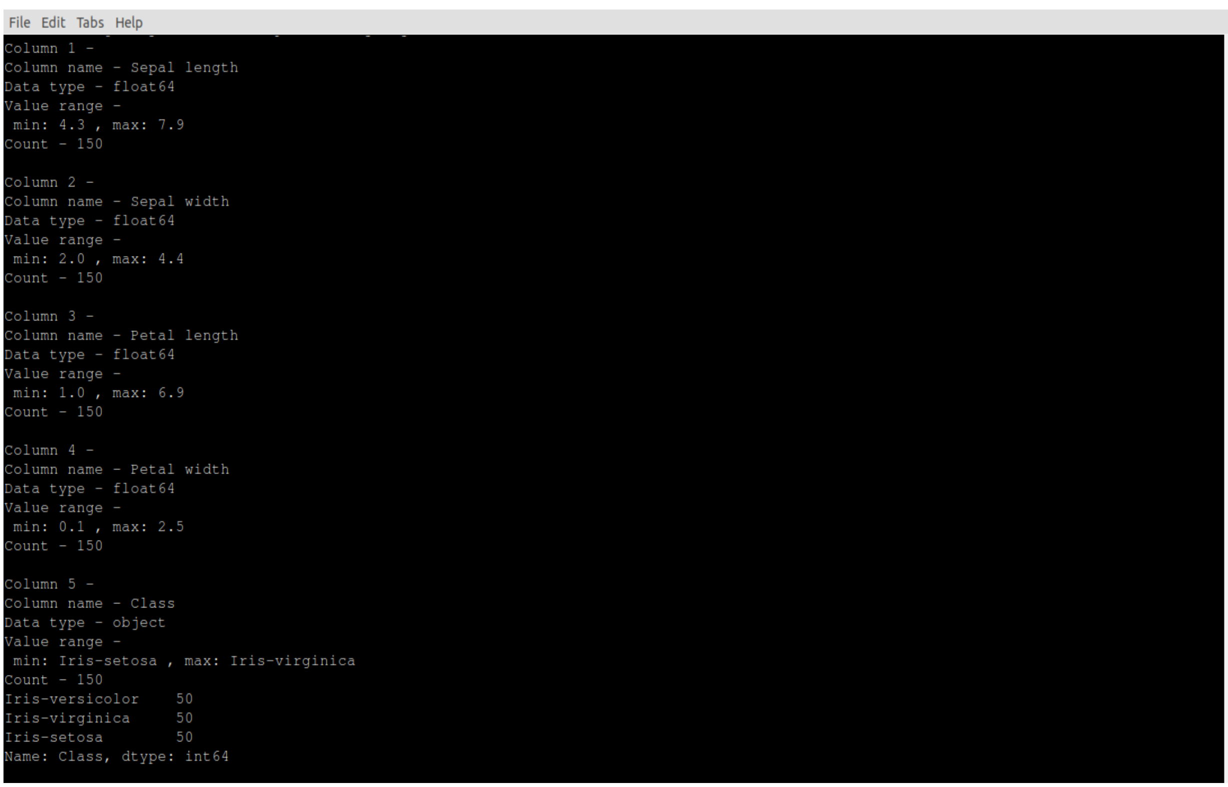
ML Assignment 1 Report

Programming Questions –

Answer 1 - Basic operation + Data Visualization

Part 1 - In this part we were asked to download the IRIS dataset. There are 5 columns in dataset. We have added the respective column names in the beginning of iris.data file. We have used columns names as given in iris.names file. Then we loaded the dataset with help of pandas library. For inspecting columns information we have used different functions of pandas library and printed the result.

We got following information for each columns -



```
File  Edit  Tabs  Help
Column 1 -
Column name - Sepal length
Data type - float64
Value range -
 min: 4.3 , max: 7.9
Count - 150

Column 2 -
Column name - Sepal width
Data type - float64
Value range -
 min: 2.0 , max: 4.4
Count - 150

Column 3 -
Column name - Petal length
Data type - float64
Value range -
 min: 1.0 , max: 6.9
Count - 150

Column 4 -
Column name - Petal width
Data type - float64
Value range -
 min: 0.1 , max: 2.5
Count - 150

Column 5 -
Column name - Class
Data type - object
Value range -
 min: Iris-setosa , max: Iris-virginica
Count - 150
Iris-versicolor    50
Iris-virginica     50
Iris-setosa        50
Name: Class, dtype: int64
```

Fig 1. Column informations

Analysis -

From this, we can observe the following information about 5 columns.

There are 150 samples in this dataset.

There are no null values in any columns, each column have 150 samples (i.e. each have count = 150).

First 4 columns are of floating data type. Range of each of these four columns are also presented above.

Last column is of type string. It have three classes - Iris-setosa, Iris-virginica and Iris-versicolor.

Each class of last column is present in equal proportion in the dataset (i.e. each have count = 50).

Now plotting the histograms for continuous valued output we get following result.

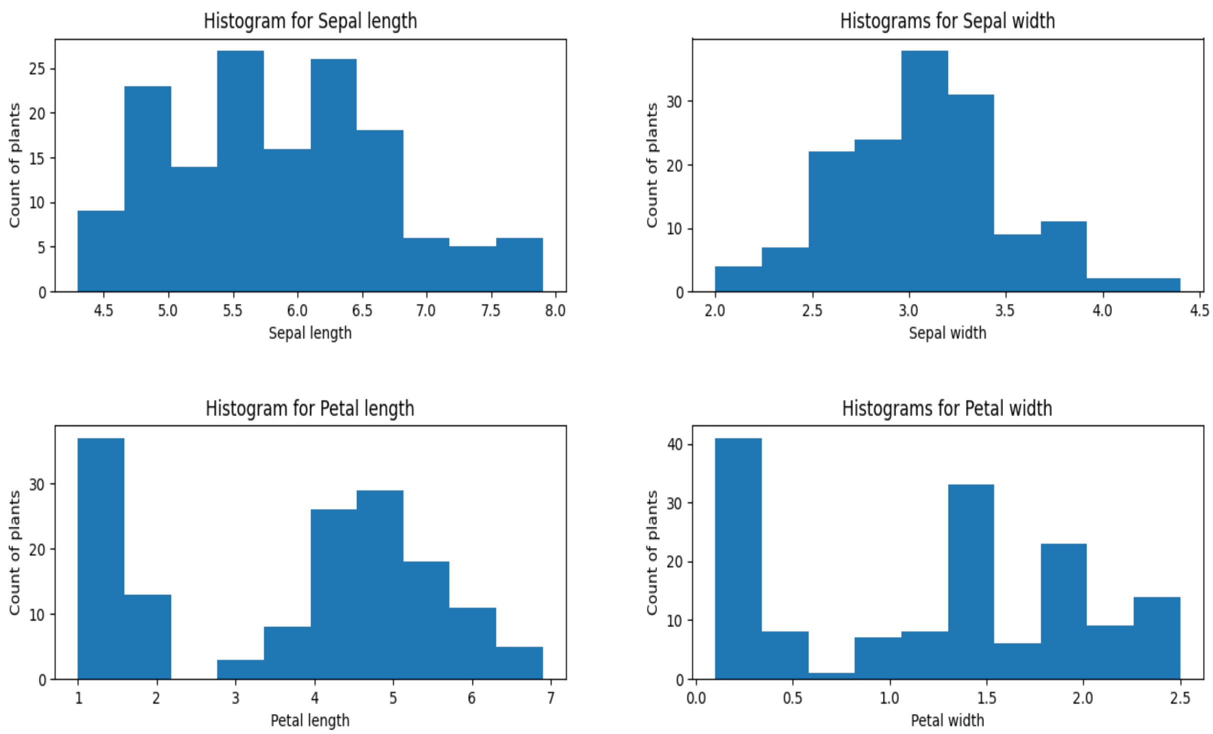


Fig 2. Histograms of continuous values

And plotting the bar graph for last column, we get the following result.

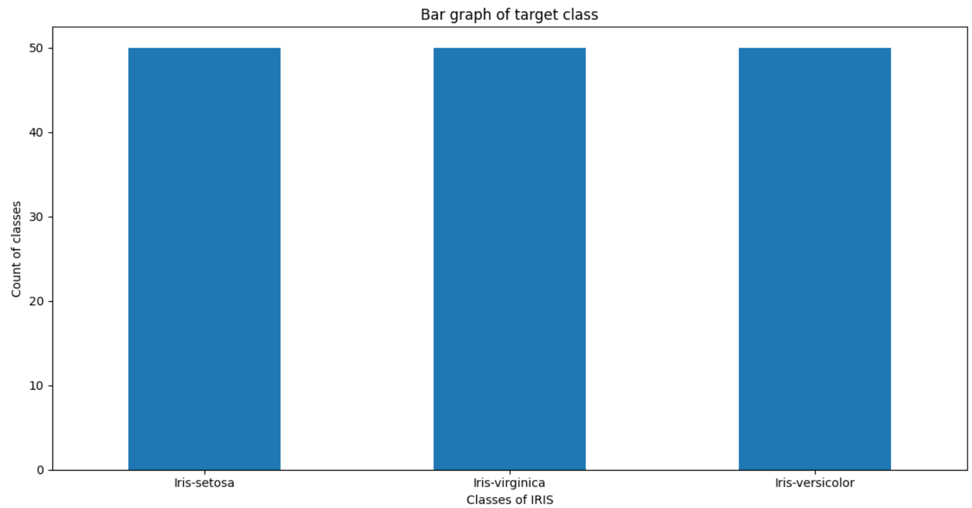


Fig 3. Bar graph of target variable

Above tells us that data is uniformly distributed between these three classes.

For running the python code -

File named Q1p1.py can be executed for this part.

Data file - iris.data should be modified as per above instruction, and should be placed in same folder as code.

Part 2 -

In this part we were asked to download MNIST dataset. After downloading, we extracted the four files. In the code, we have used convert_from_file() function of idx2numpy library to load the data in form of numpy arrays. We then have four numpy arrays as - X_train, y_train, X_test, y_test. From the dataset, we found that we have 60,000 samples for training and 10,000 samples for testing. Each sample of X_test and X_train is of dimension 28 X 28, representing one digit. Y_test and y_train represent corresponding label of digit.

In the first part, we have to visualize the two random images. From the training set, we have two random images are as follows -

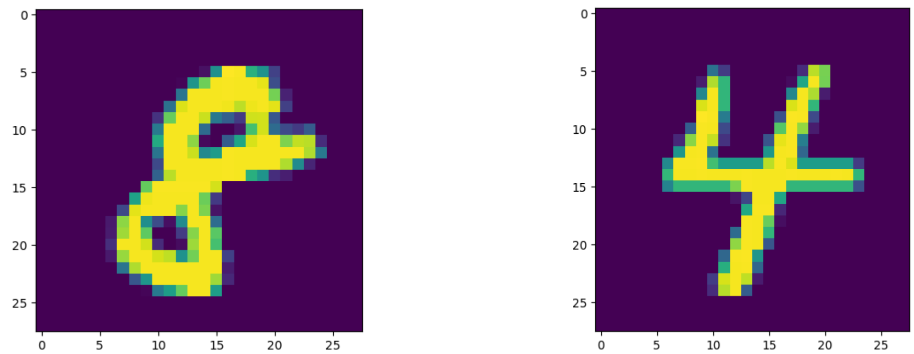
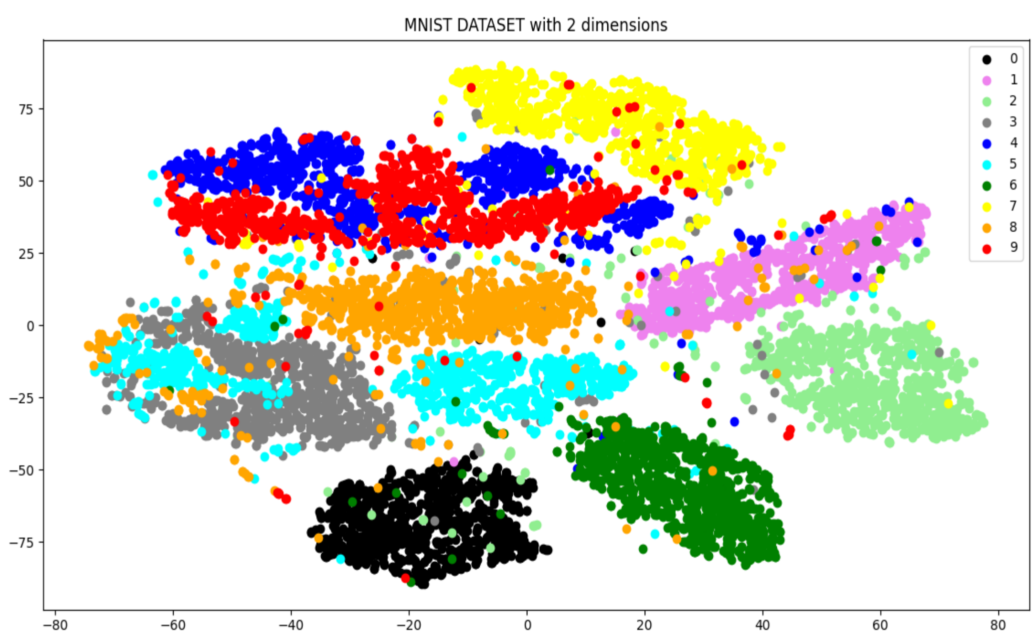


Fig 4. Two Random images from dataset

Next, we have reduce the data dimension of training data to 2 using TSNE (t-distributed stochastic neighbour embedding). X_train have 60,000 samples in which each sample is of 28 x 28. It can be viewed as 60,000 samples in which each sample is of dimension 784 x 1. It can be achieved by using reshape function from numpy library. Now our task is to reduce this 784 features to 2 features. Using TSNE we can achieve this. As TSNE, takes lot of time to execute, we have reduced the training size from 60.000 to10,000 samples.

Out of these 10,000 samples there are 1000 samples from each of 10 digits (i.e 10 x 1000 = 10,000 samples). These 10,000 samples are chosen randomly from the dataset. Then with the help of sklearn's TSNE we have reduced the dimension to 2.

Scatter plot result of TSNE out is as follows -



Comment on separability of resulting data -

From this we can see that TSNE is very nicely capturing the distinction between samples of each digit. Majority of digits are separable like 0,6,2 etc. There are also some overlaps in samples of digits like 4 and 9, 3 and 5. But overall wise TSNE can be considered to be good dimensionality reduction technique because even coming from 784 dimensions to 2, we can easily separate out 6 - 7 digit.

For running python code -

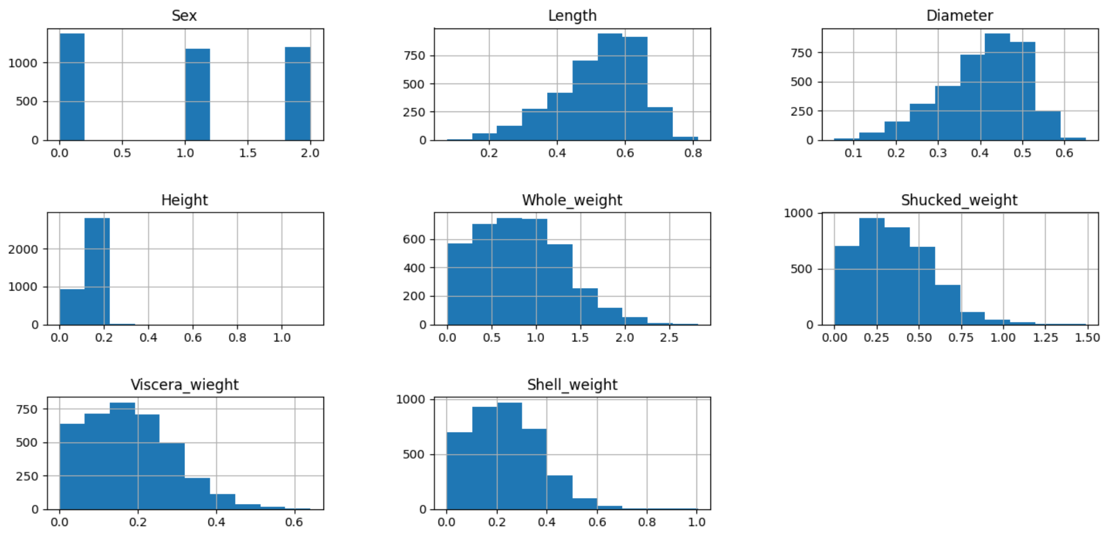
File named Q1p2.py can be executed for this part.

Data file - files from MNIST site should be downloaded and extracted then should be placed in same folder as code.

Answer 2 - Linear Regression -

In this question, we were asked to download Abalone Dataset. There are 9 columns in dataset. We firstly add the column names in Dataset.data file. We used columns names mentioned in Dataset.spec file. Then using pandas, we have read the dataset. We are using first 8 columns as input variables (X) to predict the last column (y). Out of 8 input features, 1st feature is of type string and rest 7 features are of floating data type. 1st column represents sex, which have only three classes {M,F,I}. We converted this column to float by mapping - M to 0, F to 1 and I to 2.

After this, we have splitted our data set into 90% (for training + validation)[X_train , y_train] and 10% (for testing)[X_test, y_test] using scikit-learn. We then visualize various attributes of X_train and we get the following graphs -



We can see some of attributes are not scaled properly, so it requires normalization.

<u>Gradient descent -</u>

We have defined function for gradient descent in which we have implemented its functionality from scratch. We have also customized this function to take into account for L1 and L2 regularization.

<u>Part a)</u> -

Now, let’s move to first part -

1. Firstly, we have used KFold implementation of scikit-learn to do 5 splits.

2. For each of 5 splits, we are using 4 splits for training and 1 spilt as validation set.

3. Now, we will perform following for each of val set -

    3.1. In training set we are performing normalization, using formula: X -min/max

    3.2. We then store the min, max value of above.

    3.3. We initialize our parameters as 9x1 vector equal to 1.(We have also added x0 = 1 in X_train).

    3.4. We decide some learning rate and no. of iterations and perform gradient descent to minimize parameter.

    3.5. We get some parameters from gradient descent.

    3.6. We normalize validation set using values of step 3.2. and formula 3.1.

    3.7. We calculate the RMSE on validation set and see the result.

We use step 3 to tune the hyperparametes, to get minimum avg RMSE on val sets.

Finally, we come to conclusion that

learning rate = 0.2

and iterations = 200.

For above iteration vs RMSE graph is as follows -

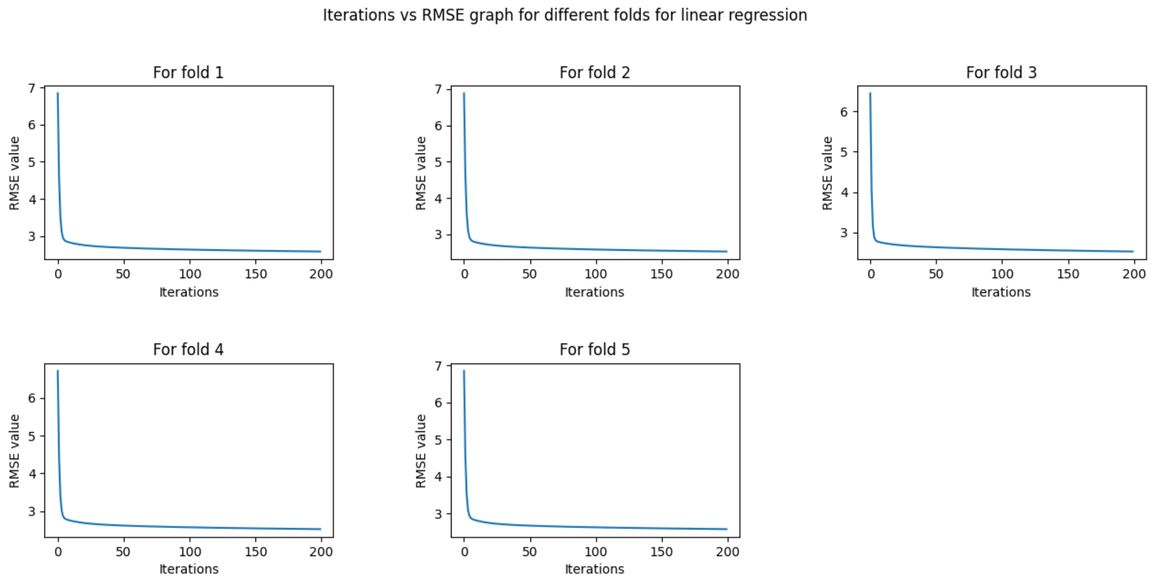
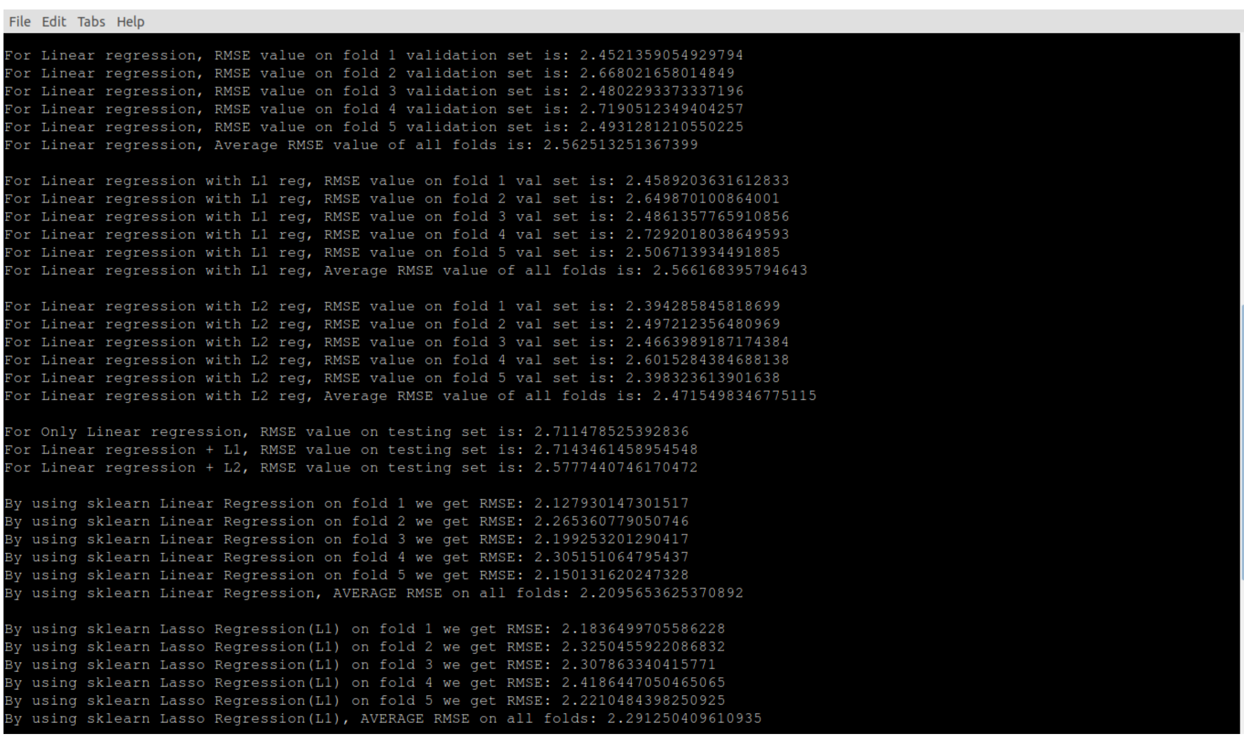


Fig. Iteration vs RMSE graph for linear regression for different folds

RMSE values on validation set is as follows -



```
File  Edit  Tabs  Help
For Linear regression, RMSE value on fold 1 validation set is: 2.4521359054929794
For Linear regression, RMSE value on fold 2 validation set is: 2.668021658014849
For Linear regression, RMSE value on fold 3 validation set is: 2.4802293373337196
For Linear regression, RMSE value on fold 4 validation set is: 2.7190512349404257
For Linear regression, RMSE value on fold 5 validation set is: 2.4931281210550225
For Linear regression, Average RMSE value of all folds is: 2.562513251367399

For Linear regression with L1 reg, RMSE value on fold 1 val set is: 2.4589203631612833
For Linear regression with L1 reg, RMSE value on fold 2 val set is: 2.649870100864001
For Linear regression with L1 reg, RMSE value on fold 3 val set is: 2.4861357765910856
For Linear regression with L1 reg, RMSE value on fold 4 val set is: 2.7292018038649593
For Linear regression with L1 reg, RMSE value on fold 5 val set is: 2.506713934491885
For Linear regression with L1 reg, Average RMSE value of all folds is: 2.566168395794643

For Linear regression with L2 reg, RMSE value on fold 1 val set is: 2.394285845818699
For Linear regression with L2 reg, RMSE value on fold 2 val set is: 2.497212356480969
For Linear regression with L2 reg, RMSE value on fold 3 val set is: 2.4663989187174384
For Linear regression with L2 reg, RMSE value on fold 4 val set is: 2.6015284384688138
For Linear regression with L2 reg, RMSE value on fold 5 val set is: 2.398323613901638
For Linear regression with L2 reg, Average RMSE value of all folds is: 2.4715498346775115

For Only Linear regression, RMSE value on testing set is: 2.711478525392836
For Linear regression + L1, RMSE value on testing set is: 2.7143461458954548
For Linear regression + L2, RMSE value on testing set is: 2.5777440746170472

By using sklearn Linear Regression on fold 1 we get RMSE: 2.127930147301517
By using sklearn Linear Regression on fold 2 we get RMSE: 2.265360779050746
By using sklearn Linear Regression on fold 3 we get RMSE: 2.199253201290417
By using sklearn Linear Regression on fold 4 we get RMSE: 2.305151064795437
By using sklearn Linear Regression on fold 5 we get RMSE: 2.150131620247328
By using sklearn Linear Regression, AVERAGE RMSE on all folds: 2.2095653625370892

By using sklearn Lasso Regression(L1) on fold 1 we get RMSE: 2.1836499705586228
By using sklearn Lasso Regression(L1) on fold 2 we get RMSE: 2.3250455922086832
By using sklearn Lasso Regression(L1) on fold 3 we get RMSE: 2.307863340415771
By using sklearn Lasso Regression(L1) on fold 4 we get RMSE: 2.4186447050465065
By using sklearn Lasso Regression(L1) on fold 5 we get RMSE: 2.2210484398250925
By using sklearn Lasso Regression(L1), AVERAGE RMSE on all folds: 2.291250409610935
```

Fig. Follow first five lines, which corresponds to Linear regression

Part b) -

In this we follow the same procedure as part a. In step 3.4 use gradient descent of L1 and L2 respectively.

At the end of tuning, we come to conclusion that

For L1 -

learning rate = 0.2

iterations = 200

regularization parameter(lambda) = 0.01

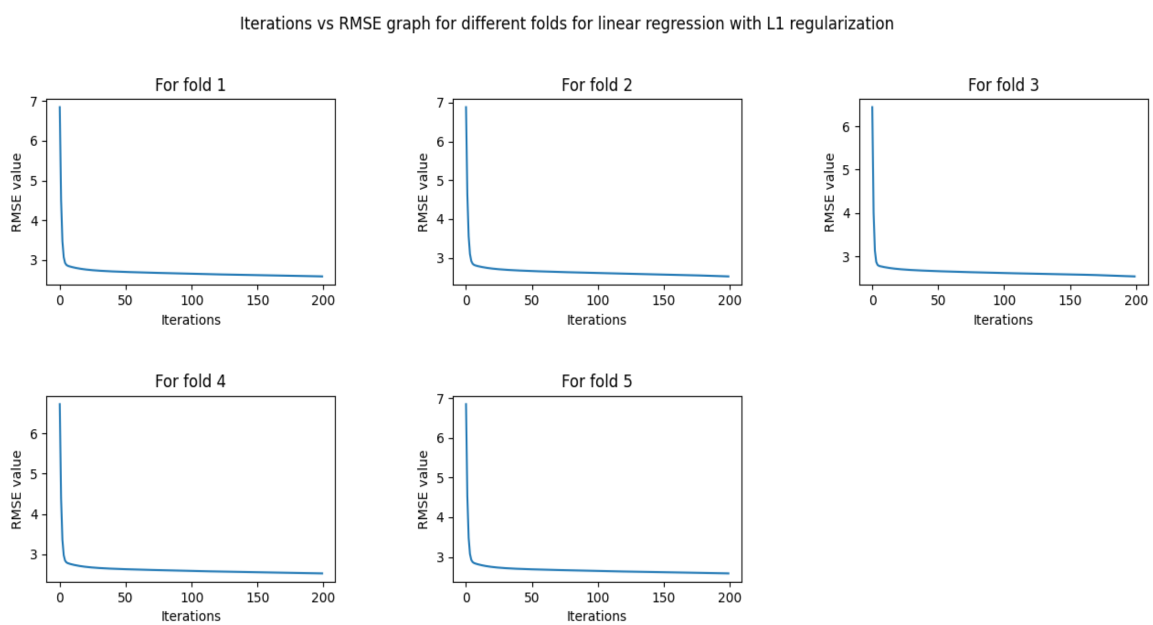
For L2 -

learning rate = 0.2

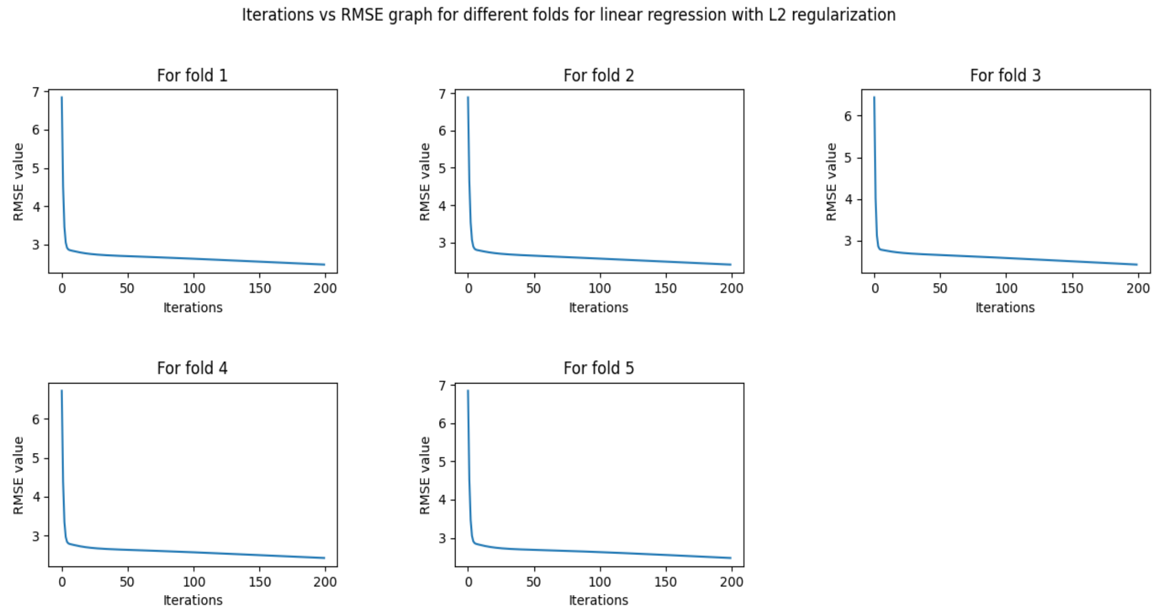
iteration = 200

regularization parameter(lambda) = 0.005

For L1 iteration vs RMSE graph is as follows -



For L2 iteration vs RMSE graph is as follows -



RMSE values on validation set is as follows -

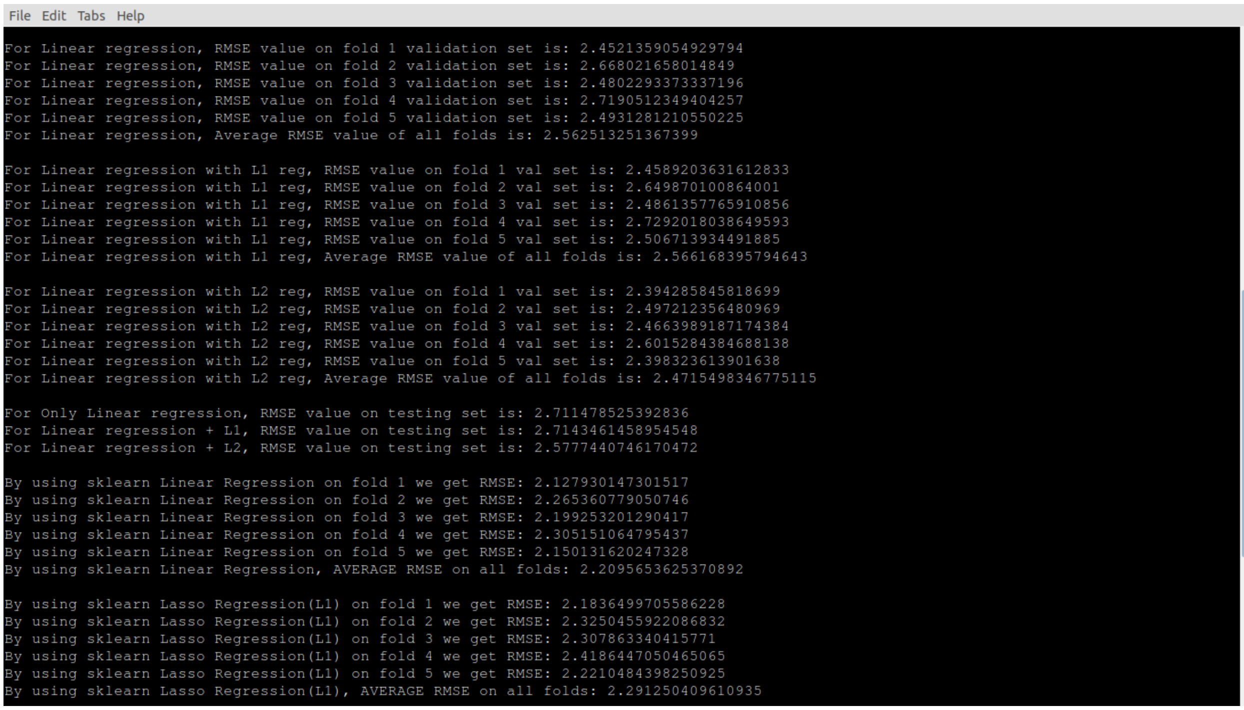
```
File  Edit  Tabs  Help
For Linear regression, RMSE value on fold 1 validation set is: 2.4521359054929794
For Linear regression, RMSE value on fold 2 validation set is: 2.668021658014849
For Linear regression, RMSE value on fold 3 validation set is: 2.4802293373337196
For Linear regression, RMSE value on fold 4 validation set is: 2.7190512349404257
For Linear regression, RMSE value on fold 5 validation set is: 2.4931281210550225
For Linear regression, Average RMSE value of all folds is: 2.562513251367399

For Linear regression with L1 reg, RMSE value on fold 1 val set is: 2.4589203631612833
For Linear regression with L1 reg, RMSE value on fold 2 val set is: 2.649870100864001
For Linear regression with L1 reg, RMSE value on fold 3 val set is: 2.4861357765910856
For Linear regression with L1 reg, RMSE value on fold 4 val set is: 2.7292018038649593
For Linear regression with L1 reg, RMSE value on fold 5 val set is: 2.506713934491885
For Linear regression with L1 reg, Average RMSE value of all folds is: 2.566168395794643

For Linear regression with L2 reg, RMSE value on fold 1 val set is: 2.394285845818699
For Linear regression with L2 reg, RMSE value on fold 2 val set is: 2.497212356480969
For Linear regression with L2 reg, RMSE value on fold 3 val set is: 2.4663989187174384
For Linear regression with L2 reg, RMSE value on fold 4 val set is: 2.6015284384688138
For Linear regression with L2 reg, RMSE value on fold 5 val set is: 2.398323613901638
For Linear regression with L2 reg, Average RMSE value of all folds is: 2.4715498346775115

For Only Linear regression, RMSE value on testing set is: 2.711478525392836
For Linear regression + L1, RMSE value on testing set is: 2.7143461458954548
For Linear regression + L2, RMSE value on testing set is: 2.5777440746170472

By using sklearn Linear Regression on fold 1 we get RMSE: 2.127930147301517
By using sklearn Linear Regression on fold 2 we get RMSE: 2.265360779050746
By using sklearn Linear Regression on fold 3 we get RMSE: 2.199253201290417
By using sklearn Linear Regression on fold 4 we get RMSE: 2.305151064795437
By using sklearn Linear Regression on fold 5 we get RMSE: 2.150131620247328
By using sklearn Linear Regression, AVERAGE RMSE on all folds: 2.2095653625370892

By using sklearn Lasso Regression(L1) on fold 1 we get RMSE: 2.1836499705586228
By using sklearn Lasso Regression(L1) on fold 2 we get RMSE: 2.3250455922086832
By using sklearn Lasso Regression(L1) on fold 3 we get RMSE: 2.307863340415771
By using sklearn Lasso Regression(L1) on fold 4 we get RMSE: 2.4186447050465065
By using sklearn Lasso Regression(L1) on fold 5 we get RMSE: 2.2210484398250925
By using sklearn Lasso Regression(L1), AVERAGE RMSE on all folds: 2.291250409610935
```

Fig. Follow linear regression with L1 and L2 on validation set

Part c)-

Using parameters founded out in part(a) and part(b), we then trained the models on 90% data (train+val) and test them on testing set.

We got the following RMSE values –



```
File  Edit  Tabs  Help
For Linear regression, RMSE value on fold 1 validation set is: 2.4521359054929794
For Linear regression, RMSE value on fold 2 validation set is: 2.668021658014849
For Linear regression, RMSE value on fold 3 validation set is: 2.4802293373337196
For Linear regression, RMSE value on fold 4 validation set is: 2.7190512349404257
For Linear regression, RMSE value on fold 5 validation set is: 2.4931281210550225
For Linear regression, Average RMSE value of all folds is: 2.562513251367399

For Linear regression with L1 reg, RMSE value on fold 1 val set is: 2.4589203631612833
For Linear regression with L1 reg, RMSE value on fold 2 val set is: 2.649870100864001
For Linear regression with L1 reg, RMSE value on fold 3 val set is: 2.4861357765910856
For Linear regression with L1 reg, RMSE value on fold 4 val set is: 2.7292018038649593
For Linear regression with L1 reg, RMSE value on fold 5 val set is: 2.506713934491885
For Linear regression with L1 reg, Average RMSE value of all folds is: 2.566168395794643

For Linear regression with L2 reg, RMSE value on fold 1 val set is: 2.394285845818699
For Linear regression with L2 reg, RMSE value on fold 2 val set is: 2.497212356480969
For Linear regression with L2 reg, RMSE value on fold 3 val set is: 2.4663989187174384
For Linear regression with L2 reg, RMSE value on fold 4 val set is: 2.6015284384688138
For Linear regression with L2 reg, RMSE value on fold 5 val set is: 2.398323613901638
For Linear regression with L2 reg, Average RMSE value of all folds is: 2.4715498346775115

For Only Linear regression, RMSE value on testing set is: 2.711478525392836
For Linear regression + L1, RMSE value on testing set is: 2.7143461458954548
For Linear regression + L2, RMSE value on testing set is: 2.5777440746170472

By using sklearn Linear Regression on fold 1 we get RMSE: 2.127930147301517
By using sklearn Linear Regression on fold 2 we get RMSE: 2.265360779050746
By using sklearn Linear Regression on fold 3 we get RMSE: 2.199253201290417
By using sklearn Linear Regression on fold 4 we get RMSE: 2.305151064795437
By using sklearn Linear Regression on fold 5 we get RMSE: 2.150131620247328
By using sklearn Linear Regression, AVERAGE RMSE on all folds: 2.2095653625370892

By using sklearn Lasso Regression(L1) on fold 1 we get RMSE: 2.1836499705586228
By using sklearn Lasso Regression(L1) on fold 2 we get RMSE: 2.3250455922086832
By using sklearn Lasso Regression(L1) on fold 3 we get RMSE: 2.307863340415771
By using sklearn Lasso Regression(L1) on fold 4 we get RMSE: 2.4186447050465065
By using sklearn Lasso Regression(L1) on fold 5 we get RMSE: 2.2210484398250925
By using sklearn Lasso Regression(L1), AVERAGE RMSE on all folds: 2.291250409610935
```

Fig. Follow RMSE values on testing set for Linear regression, Linear regression +L1, Linear regression+L2

We can see that linear regression + L2 is performing slightly better with these set of parameters.

Part d) -

In this part, we perform the same steps as part(a) and part(b), but here we will use inbuilt libraries to train the models.

Inbuilt sklearn linear regression does not require any parameters.

By testing on validation set, we set

Parameter for lasso regression as 0.01 and

Parameter for ridge regression as 0.05.

This is result of three models on validation set -

```
File  Edit  Tabs  Help
For Only Linear regression, RMSE value on testing set is: 2.711478525392836
For Linear regression + L1, RMSE value on testing set is: 2.7143461458954548
For Linear regression + L2, RMSE value on testing set is: 2.5777440746170472

By using sklearn Linear Regression on fold 1 we get RMSE: 2.127930147301517
By using sklearn Linear Regression on fold 2 we get RMSE: 2.265360779050746
By using sklearn Linear Regression on fold 3 we get RMSE: 2.199253201290417
By using sklearn Linear Regression on fold 4 we get RMSE: 2.305151064795437
By using sklearn Linear Regression on fold 5 we get RMSE: 2.150131620247328
By using sklearn Linear Regression, AVERAGE RMSE on all folds: 2.2095653625370892

By using sklearn Lasso Regression(L1) on fold 1 we get RMSE: 2.1836499705586228
By using sklearn Lasso Regression(L1) on fold 2 we get RMSE: 2.3250455922086832
By using sklearn Lasso Regression(L1) on fold 3 we get RMSE: 2.307863340415771
By using sklearn Lasso Regression(L1) on fold 4 we get RMSE: 2.4186447050465065
By using sklearn Lasso Regression(L1) on fold 5 we get RMSE: 2.2210484398250925
By using sklearn Lasso Regression(L1), AVERAGE RMSE on all folds: 2.291250409610935

By using sklearn Ridge Regression(L2) on fold 1 we get RMSE: 2.1277722149491645
By using sklearn Ridge Regression(L2) on fold 2 we get RMSE: 2.264699471871522
By using sklearn Ridge Regression(L2) on fold 3 we get RMSE: 2.203654839208121
By using sklearn Ridge Regression(L2) on fold 4 we get RMSE: 2.3085378961098098
By using sklearn Ridge Regression(L2) on fold 5 we get RMSE: 2.146513599470076
By using sklearn Ridge Regression(L2), AVERAGE RMSE on all folds: 2.2102356043217384

By using sklearn Linear Regression on testing set,we get RMSE: 2.344177532338043
By using sklearn Lasso Regression(L1) on testing set,we get RMSE: 2.410392685881329
By using sklearn Ridge Regression(L2) on testing set,we get RMSE: 2.3432424259101


For Linear regression in closed form, RMSE value on fold 1 val set is: 2.1279301473015133
For Linear regression in closed form, RMSE value on fold 2 val set is: 2.2653607790507473
For Linear regression in closed form, RMSE value on fold 3 val set is: 2.199253201290348
For Linear regression in closed form, RMSE value on fold 4 val set is: 2.3051510647955364
For Linear regression in closed form, RMSE value on fold 5 val set is: 2.1501316202473313
For Linear regression in closed form, Average RMSE value of all folds is: 2.2095653625370955


Press ENTER or type command to continue
```

Fig. Follow Sklearn Linear regression, Sklearn Ridge Regression and Sklearn Lasso Regression on validation set

With parameters describe earlier, we then trained the model on validation set and tested it on testing set.

We got following RMSE values -


```
File  Edit  Tabs  Help
For Only Linear regression, RMSE value on testing set is: 2.711478525392836
For Linear regression + L1, RMSE value on testing set is: 2.7143461458954548
For Linear regression + L2, RMSE value on testing set is: 2.5777440746170472

By using sklearn Linear Regression on fold 1 we get RMSE: 2.127930147301517
By using sklearn Linear Regression on fold 2 we get RMSE: 2.265360779050746
By using sklearn Linear Regression on fold 3 we get RMSE: 2.199253201290417
By using sklearn Linear Regression on fold 4 we get RMSE: 2.305151064795437
By using sklearn Linear Regression on fold 5 we get RMSE: 2.150131620247328
By using sklearn Linear Regression, AVERAGE RMSE on all folds: 2.2095653625370892

By using sklearn Lasso Regression(L1) on fold 1 we get RMSE: 2.1836499705586228
By using sklearn Lasso Regression(L1) on fold 2 we get RMSE: 2.3250455922086832
By using sklearn Lasso Regression(L1) on fold 3 we get RMSE: 2.307863340415771
By using sklearn Lasso Regression(L1) on fold 4 we get RMSE: 2.4186447050465065
By using sklearn Lasso Regression(L1) on fold 5 we get RMSE: 2.2210484398250925
By using sklearn Lasso Regression(L1), AVERAGE RMSE on all folds: 2.291250409610935

By using sklearn Ridge Regression(L2) on fold 1 we get RMSE: 2.1277722149491645
By using sklearn Ridge Regression(L2) on fold 2 we get RMSE: 2.264699471871522
By using sklearn Ridge Regression(L2) on fold 3 we get RMSE: 2.203654839208121
By using sklearn Ridge Regression(L2) on fold 4 we get RMSE: 2.3085378961098098
By using sklearn Ridge Regression(L2) on fold 5 we get RMSE: 2.146513599470076
By using sklearn Ridge Regression(L2), AVERAGE RMSE on all folds: 2.2102356043217384

By using sklearn Linear Regression on testing set,we get RMSE: 2.344177532338043
By using sklearn Lasso Regression(L1) on testing set,we get RMSE: 2.410392685881329
By using sklearn Ridge Regression(L2) on testing set,we get RMSE: 2.3432424259101


For Linear regression in closed form, RMSE value on fold 1 val set is: 2.1279301473015133
For Linear regression in closed form, RMSE value on fold 2 val set is: 2.2653607790507473
For Linear regression in closed form, RMSE value on fold 3 val set is: 2.199253201290348
For Linear regression in closed form, RMSE value on fold 4 val set is: 2.3051510647955364
For Linear regression in closed form, RMSE value on fold 5 val set is: 2.1501316202473313
For Linear regression in closed form, Average RMSE value of all folds is: 2.2095653625370955


Press ENTER or type command to continue
```


Fig. Follow Testing result of Sklearn Linear regression, Sklearn Ridge Regression and Sklearn Lasso Regression

We can see that only linear regression and linear regression+L2 is almost giving same RMSE values.

Comparing it with our above result part(d), we can see that there is slight difference in decimals. And inbuilt functions are slightly giving better RMSE values. This could be due to implementation of inbuilt functions. As they implement closed form solution and we are training with gradient descent. Also we are only doing 200 iterations in gradient descent.

Part e)-

Now for implementing closed form solution, we used formula derived in theory part of this assignment.

We get following RMSE value on validation sets -

```
File  Edit  Tabs  Help
For Only Linear regression, RMSE value on testing set is: 2.711478525392836
For Linear regression + L1, RMSE value on testing set is: 2.7143461458954548
For Linear regression + L2, RMSE value on testing set is: 2.5777440746170472

By using sklearn Linear Regression on fold 1 we get RMSE: 2.127930147301517
By using sklearn Linear Regression on fold 2 we get RMSE: 2.265360779050746
By using sklearn Linear Regression on fold 3 we get RMSE: 2.199253201290417
By using sklearn Linear Regression on fold 4 we get RMSE: 2.305151064795437
By using sklearn Linear Regression on fold 5 we get RMSE: 2.150131620247328
By using sklearn Linear Regression, AVERAGE RMSE on all folds: 2.2095653625370892

By using sklearn Lasso Regression(L1) on fold 1 we get RMSE: 2.1836499705586228
By using sklearn Lasso Regression(L1) on fold 2 we get RMSE: 2.3250455922086832
By using sklearn Lasso Regression(L1) on fold 3 we get RMSE: 2.307863340415771
By using sklearn Lasso Regression(L1) on fold 4 we get RMSE: 2.4186447050465065
By using sklearn Lasso Regression(L1) on fold 5 we get RMSE: 2.2210484398250925
By using sklearn Lasso Regression(L1), AVERAGE RMSE on all folds: 2.291250409610935

By using sklearn Ridge Regression(L2) on fold 1 we get RMSE: 2.1277722149491645
By using sklearn Ridge Regression(L2) on fold 2 we get RMSE: 2.264699471871522
By using sklearn Ridge Regression(L2) on fold 3 we get RMSE: 2.203654839208121
By using sklearn Ridge Regression(L2) on fold 4 we get RMSE: 2.3085378961098098
By using sklearn Ridge Regression(L2) on fold 5 we get RMSE: 2.146513599470076
By using sklearn Ridge Regression(L2), AVERAGE RMSE on all folds: 2.2102356043217384

By using sklearn Linear Regression on testing set,we get RMSE: 2.344177532338043
By using sklearn Lasso Regression(L1) on testing set,we get RMSE: 2.410392685881329
By using sklearn Ridge Regression(L2) on testing set,we get RMSE: 2.3432424259101


For Linear regression in closed form, RMSE value on fold 1 val set is: 2.1279301473015133
For Linear regression in closed form, RMSE value on fold 2 val set is: 2.2653607790507473
For Linear regression in closed form, RMSE value on fold 3 val set is: 2.199253201290348
For Linear regression in closed form, RMSE value on fold 4 val set is: 2.3051510647955364
For Linear regression in closed form, RMSE value on fold 5 val set is: 2.1501316202473313
For Linear regression in closed form, Average RMSE value of all folds is: 2.2095653625370955


Press ENTER or type command to continue
```

Fig. Follow the last set of results for closed form solution

Note that, result we got almost same result for this as we got by using scikit-learn linear regression.

This is because of the fact that scikit-learn implements closed form solution in its implementations.

For RUNNING python code -

Q2.py file should be executed for this part.

Dataset - should be manipulated as discussed at start of this question and should be placed in same folder as code.

There are functions calls at the end of the file Q2.py, for each part. Uncommenting any part will not run that part.

THEORY PART FOLLOWS -

Name → Sagar Suman
Roll No. 2019197
ML Assignment - 1

Page No.
Date:

THEORY - PART

Q4 :→ We have linear regression model as

$$y = X\theta + \epsilon$$

where

y is $N \times 1$ vector
X is $N \times d$ vector
θ is $d \times 1$ vector

and ϵ is irreducible loss.
We are required to ~~predict~~ estimate parameter θ, and derive closed form solution for it.

Firstly we will add one more feature in X, i.e. $x_0 = 1$ so, that we can have θ_0 parameter for representing constant.
So,

X now will have dimension $(N) \times (d+1)$

$$X = \underbrace{\begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \cdots \\ 1 & x_{21} & x_{22} & x_{23} & \cdots \\ \vdots \\ 1 \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nd} \end{bmatrix}}_{N \times (d+1)}$$

Similarly we will add 1 more feature in θ, i.e. θ_0

So θ will have dimension $(d+1) \times 1$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix}_{(d+1)\times 1}$$

Page No.	
Date:	

Now we will try to model output y as

$\rightarrow \quad \hat{y} = X\theta$

Note that ϵ is irreducible error, and we are finding parameters θ such that it best mimic y.
In other words, we want $\hat{y}$ as close to y.
As $\rightarrow$ $\hat{y}$ is $N\times 1$ and y is also $N\times 1$

So, it makes sense to minimize square of difference between them, to get the measure of how good is $\hat{y}$ is.
i.e,

$$\boxed{\theta = \arg\min_{\theta} \sum_{i=1}^{N} \left(y^{(i)} - \hat{y}^{(i)} \right)^2}$$

$y^{(i)}, \hat{y}^{(i)}$ are ith term of y and $\hat{y}$.

So this is our objective function, i.e. arg min will give that values of θ, which will minimize the sum of square errors.

$\rightarrow$ Note:- We have come up with this function intuitively, we can also come up with same objective function mathematically. i.e. by assuming ϵ to have distribution as $N(0, \sigma^2)$ [zero mean and variance σ^2, gaussian] followed by obtaining maximum likelihood function.

Now, we have

$$\theta = \arg\min_{\theta} \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2$$

let $e = y - \hat{y}$ $\quad$ e :→ is $N \times 1$ vector.

so,

minimizing $\sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)})^2$

is equivalent to $\boxed{\text{minimizing } e^T e}$

$\qquad\qquad 1 \times N \quad N \times 1$

and note that e is only function of θ, as x and y are given.

followed by above, as sum of square error is convex function, so ~~minimum~~ differentiating $(e^T e)$ w.r.t. θ and equating it to 0 will give us minimum loss.

Note that θ is $((d+1) \times 1)$ vector.

Now, with above in mind putting derivative w.r.t θ of $(e^T e) = 0$

$$\nabla_{\theta}(e^T e) = 0$$

$$\nabla_{\theta}\left((y - \hat{y})^T (y - \hat{y})\right) = 0$$

is $\hat{y} = X\theta$

$$\nabla_\theta \left[(y - X\theta)^T (y - X\theta) \right] = 0$$

Page No.

Date:

as $(ab)^T = b^T a^T$

$$\nabla_\theta \left[(y^T - \theta^T X^T)(y - X\theta) \right] = 0$$

multiplying

$$\nabla_\theta \left[y^T y - y^T X\theta - \underline{\underline{\theta^T X^T y}} + \theta^T X^T X \theta \right] = 0$$

$$\underset{1\times(d+1)}{(\theta^T} \;\; \underset{(d+1)\times N}{X^T} \;\; \underset{N\times 1}{y)}$$ => it have dimension of (1×1) so, taking its transpose will not change its value

i.e. $\boxed{(\theta^T X^T y) = (\theta^T X^T y)^T = y^T X \theta}$

Putting it in above equation.

$$\nabla_\theta \left[y^T y - y^T X\theta - y^T X\theta + \theta^T X^T X \theta \right] = 0$$

$$\nabla_\theta \left[y^T y - 2 y^T X\theta + \theta^T X^T X \theta \right] = 0$$

$\Downarrow$

Now we will use following rules from vector calculus → a and b vectors and matrix

Rules

① → $\nabla_a(b) = 0$ [where 'b' does not depend on any 'a' term]

② → $\nabla_a(ba) = b$

③ → $\nabla_a(a^T b a) = 2ab$ [condition => b should be symmetric]

| $a \to m \times 1$ | c : -) | $b \to m \times m$ |

With these rules in mind, lets expand the expression →

Page No.
Date:

$$\underline{\nabla_\theta(yy^T)} - \underline{2\nabla_\theta(y^T X\theta)} + \underline{\nabla_\theta(\theta^T x^T x\theta)} = 0$$

$\nabla_\theta(yy^T)$ → By Rule 1 ⇓ $= 0$.

$2\nabla_\theta(y^T X\theta)$ → By Rule ② $= 2y^T x$

$\nabla_\theta(\theta^T x^T x\theta)$ → As, $(x^T x)$ is symmetric so, By Rule ③ $= \cancel{2(x^T x)\theta}\ 2\theta^T(x^T x)$

So,

$\Rightarrow \cancel{-2y^T x + 2\theta^T(x^T x) = 0}$

$\Rightarrow \cancel{-y^T x + (x^T x)\theta = 0}$

$\Rightarrow \cancel{-y^T x + x^T x\theta = 0}$

$\Rightarrow \cancel{x^T x\theta = y^T x}$

So,

$\Rightarrow -2y^T x + 2\theta^T(x^T x) = 0$

$\Rightarrow -2y^T x + 2\theta^T x^T x = 0$

$\Rightarrow -y^T x + \theta^T x^T x = 0$

$\Rightarrow \theta^T x^T x = y^T x$

⇒ Taking Transpose on both side →

$(\theta^T x^T x)^T = (y^T x)^T$

$$\Rightarrow \boxed{(x^T x\theta) = x^T y}$$

So

$$X^T X \theta = X^T y$$

if $X^T X$ is invertible →

which gives

$$\boxed{\theta = (X^T X)^{-1} X^T y}$$

So,

θ can be directly estimated by computing

$$\theta = \boxed{(X^T X)^{-1} X^T y}$$

closed form solution.

Ans4.2: → Condition for existence of closed form solution is →

We are taking inverse of $X^T X$,

So, $(X^T X)$ should be invertible

↓

and it is invertible when X is full column rank.

So, if X is ~~N×(d+1)~~ $N \times d$ matrix then,

⇒ ① $\boxed{N \geq d}$

⇒ ② Features should be linearly independent.

Above two ensures rank of X to be d.

Ans 4 ③ Closed form solution gives / estimates the parameter in one short

①
But, And we don't have to tune any parameters for it.

But, there is computation constraint. That is, closed form solution is of order $O(n^3)$ complexity. So, for large dataset, lets say when we have 10,000 samples then, $O(n^3)$ will take much time.

In this case, if we use gradient descent, which is of order $O(kn^2)$, then it will be computationally less expensive.

(II)
Second constraint could be on very ~~small~~ dataset, where closed solution itself does not exist. In that cases also, it is better to use gradient descent.

Ans 4.4

we have to prove that for simple linear regression, least squares fit line passes through arithmetic mean of independent and dependent variable.

lets take simple linear regression, which corresponds to (x, y) pairs.

And least square fit line be represented as →

$$\hat{y}^{(i)} = \theta_0 + \theta_1 x^{(i)}$$

where,

~~$\hat{y}^{(i)}$ → ith sample of dependent variable~~

$x^{(i)}$ → i^{th} independent variable sample

θ_0, θ_1 be parameters estimated by least square.

Lets say we have N samples, so →

$$\hat{y}^{(1)} = \theta_0 + \theta_1 x^{(1)}$$
$$\hat{y}^{(2)} = \theta_0 + \theta_1 x^{(2)}$$
$$\hat{y}^{(3)} = \theta_0 + \theta_1 x^{(3)}$$
$$\vdots$$
$$\hat{y}^{(N)} = \theta_0 + \theta_1 x^{(N)}$$

Now, we can write,

$$y^{(i)} = \hat{y}^{(i)} + \epsilon^{(i)}$$

where $y^{(i)}$ is i^{th} sample of dependent variable

$\hat{y}^{(i)}$ is predicted value of $y^{(i)}$

ϵ is irreducible error.

So, for N Terms we can write →

Page No.			
Date:			

$$y^{(1)} = \theta_0 + \theta_1 x^{(1)} + \epsilon_1$$

$$y^{(2)} = \theta_0 + \theta_1 x^{(2)} + \epsilon_2$$

$\vdots$

$$y^{(N)} = \theta_0 + \theta_1 x^{(N)} + \epsilon_N$$

$$\sum_{i=1}^{N} y^{(i)} = \sum_{i=1}^{N} \theta_0 + \sum_{i=1}^{N} \theta_1 x^{(i)} + \sum_{i=1}^{N} \epsilon_i$$ ← Summing over all values.

$$\sum_{i=1}^{N} y^{(i)} = N\theta_0 + \theta_1 \sum_{i=1}^{N} x^{(i)} + \sum_{i=1}^{N} \epsilon_i$$

as $\frac{\epsilon_1 + \epsilon_2 + \epsilon_n}{n} = 0$ we will ~~assume~~ use mean of irreducible error $= 0$

Dividing by N

$$\sum_{i=1}^{N} \frac{y^{(i)}}{N} = \theta_0 + \theta_1 \frac{\sum_{i=1}^{N} x^{(i)}}{N} + \cancel{\frac{\sum_{i=1}^{N} \epsilon_i}{N}}$$

$$\Rightarrow \boxed{\bar{y} = \theta_0 + \theta_1 \bar{x}}$$

where

$$\bar{y} = \sum_{i=1}^{N} \frac{y^{(i)}}{N}$$

$$\bar{x} = \sum_{i=1}^{N} \frac{x^{(i)}}{N}$$

→ arithmetic mean of independent and dependent variable.

Least square line passing through

Ans 4.5 : yes
So,
Answer is <u>not directly.</u>

Page No.	
Date:	

We can not model classification problem <u>directly</u> with LR this is because of the fact that in linear regression, output we get is continuous and in classification problem we have discrete output. Even if we use some thresholding in linear regression it still might not give correct result.

But,

if we can somehow model an invertible function of output which is continuous, then in that case $\boxed{\text{we can use linear regression.}}$

For example,
In case of logistic regression we are modelling ~~the probability of a output being~~ ~~...~~.
<u>log (of odds (of probability of y being 1))</u> as linear model.
logit function

And we are using ~~this~~ ~~logit function~~ linear regression for this logit function.
And since this function is invertible we ~~are~~ getting our probablity back with the help of sigmoid function.
This probability is being used in classification.