CS521 Assignment 1: Regression Analysis

Ashwin Goyal
2018meb1214@iitrpr.ac.in

Indian Institute of Technology
Ropar, Punjab

Abstract

The purpose of this assignment is to help us understand the various versions of Linear Regression through its implementation on the **Boston house prices dataset**. We are required to use the data to predict house prices for new data/unseen data points. The dataset has features like residential land zoned, proportion of non-retail business acres per town, average number of rooms per dwelling, etc. and has the prices corresponding to each data point. In total there are 13 features and 506 data points. The data was drawn from the Boston Standard Metropolitan Statistical Area in 1970.

1 Introduction

In this assignment, I've run 3 different linear regression models and plotted the results for better understanding. This has helped me understand Linear Regression better and comparatively analyse the 3 techniques. Also, good results have been obtained through the models. I have also plotted the residuals and analysed the mean error of these approaches.

2 Importing libraries, loading the dataset, Data preprocessing and train/test split

The libraries imported by me are: *Numpy, Pandas, Seaborn, and Matplotlib, SKLearn*. After that, I've imported the Boston house prices dataset using the datasets package in sklearn. The I've created a Pandas dataframe wherein, I have set the columns as the feature names and the last column as the labels. *(the dataframe is shown in Figure 1 below)*

After this, I've preprocessed the dataset through scaling the features. This is done by dividing by the standard deviation and subtracting the mean for each data point. This is done to standardise the features. I've even checked if some values in the dataset are missing, using df.isnull().sum(), and noted that there is no missing value.

The dataset has then been split by using the train_test_split functionality of sklearn. I have taken the training set size to be 70% of the dataset and the testing set is 30% of the dataset.

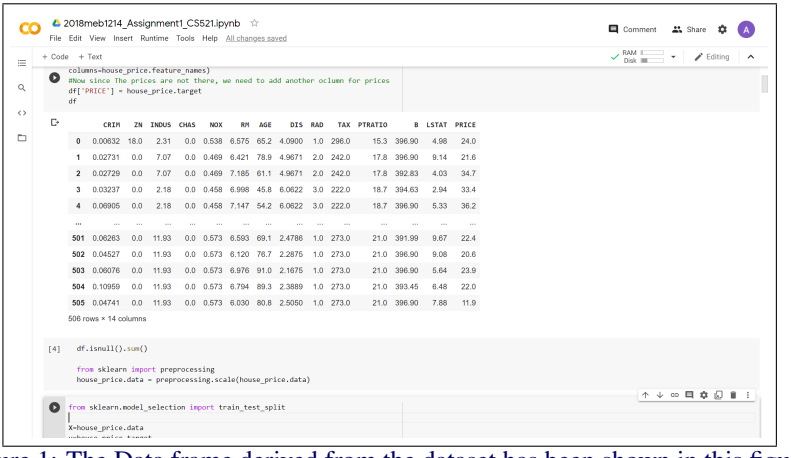


Figure 1: The Data frame derived from the dataset has been shown in this figure.

3 Regression Analysis

3.1 Task 1: OLS Regression

Note: Train/test split (mentioned above) is also a part of task 1. After splitting into train/test sets, I have created an object of linear regression and then fitted it on the training data. The predictions on the training data are quite accurate as seen from the scatter plot obtained, *shown in figure 2 below.*

Next, I have plotted the values of the regression coefficients for the different predictor variables using a bar graph. The graph is shown below in *figure 3*. For this graph, I created a new dataframe which holds only the regression coefficients obtained for different feature variables.

Also, as required in task 4, I've plotted the residual plots along with the OLS Regression model in the code. These are mentioned in Task 4 in detail.

Next, I have done something interesting. I have plotted a heatmap using Seaborn which shows the correlation of different features with each other and with the price. As we can see from this heatmap, RM, and LSTAT features have strong absolute correlation with PRICE. Hence, now I have trained our dataset using only these 2 features. The scatter plot obtained from the predicted test data is shown in the code and also the regression coefficients obtained for RM and LSTAT have been plotted on a bar graph. Also, the residual plot has been plotted again, this time using only these 2 features. The results are quite good.

3.2 Task 2: Ridge Regression

After doing a train/test split, I have created an object of Ridge Regression where I've initialised my model using alpha=0 (alpha is lambda, the regularization parameter). Then I created a dataframe with the feature names and with the regression coefficients obtained through using Ridge Regression. After this, I've trained the model on different regularization coefficients (lambda values) from 1 to 200 and created a data frame with the coefficient estimates for these regularization values and with the feature names as the columns. Using this new data frame, I've **plotted the coefficient estimates with ridge regression for the**

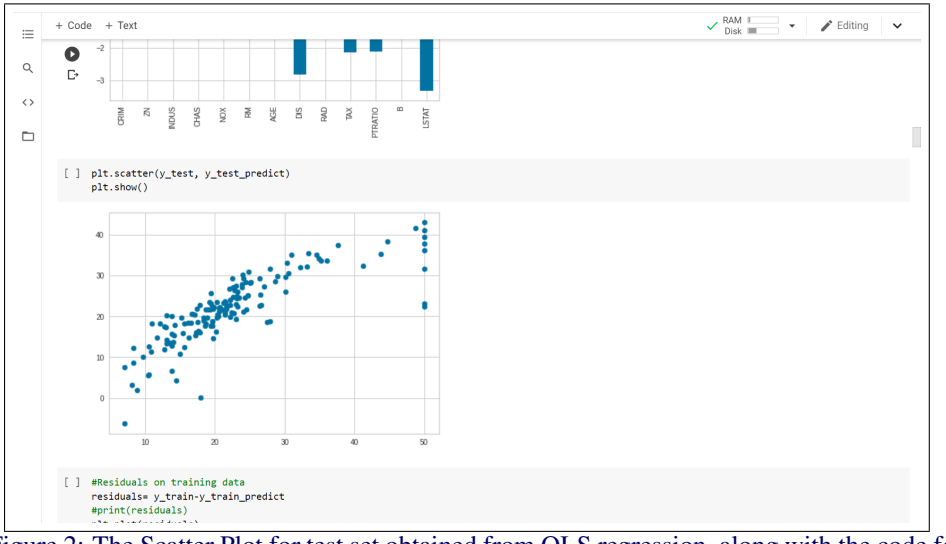


Figure 2: The Scatter Plot for test set obtained from OLS regression, along with the code for plotting it.

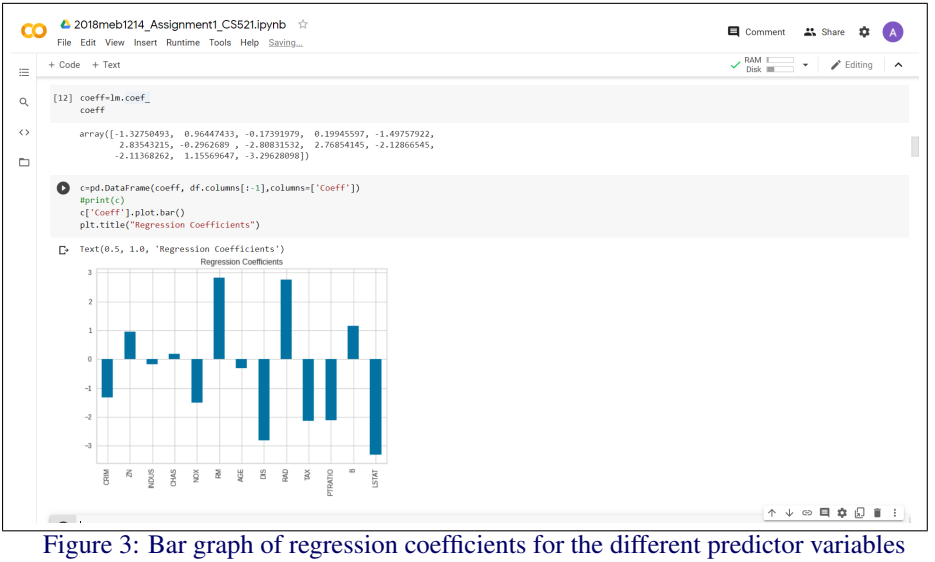


Figure 3: Bar graph of regression coefficients for the different predictor variables

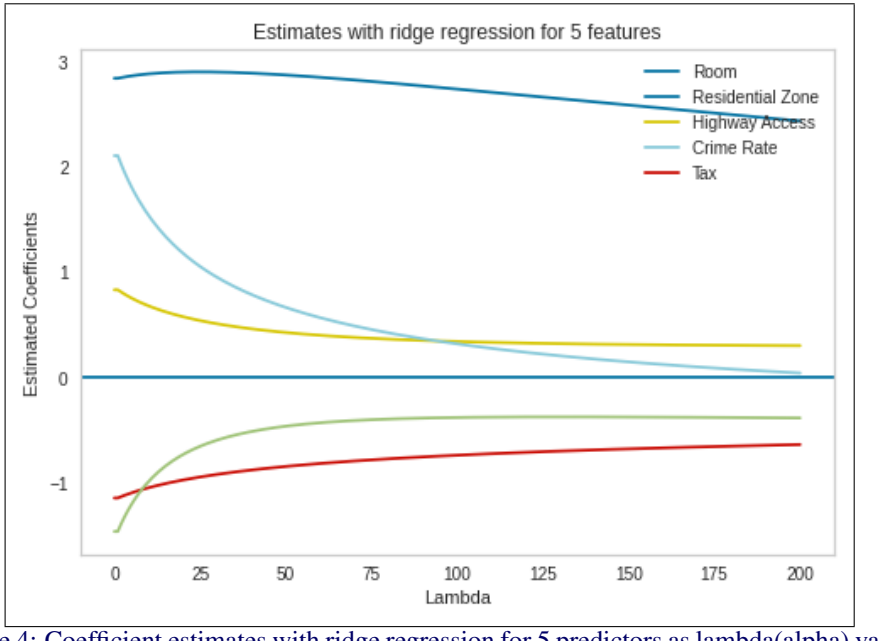


Figure 4: Coefficient estimates with ridge regression for 5 predictors as lambda(alpha) varies from 0-200

following 5 predictors: room, residential zone, highway access, crime rate and tax as lambda(alpha) varies from 0-200. The plot above*(figure 4)* shows the resulting figure. **As we can see, the lines are converging towards 0 as lambda increases. The feature Room (shown with blue line) decreases very less while features like Crime Rate and Tax rush towards zero quickly.**

After this, I have drawn a bar graph with the coefficient values for different features as I had done in OLS regression. See figure 5.

Also, I've plotted the residuals as I had done in OLS Regression. (More on this in task 4)

3.3 Task 3: Lasso Regression

To begin with Lasso Regression, I've done a train/test split in 70/30 ratio. After that, I've initialised my model using alpha=0 (alpha is lambda, the regularization parameter). Then I've trained the model for different lambdas from 1-200.

A dataframe is then created which stores the coefficients as well as the predictions using Lasso regression on the training set and the test set with columns as the feature names. The dataframe created is shown in figure 6. ***Note that many values are 0, which is expected because lasso regression drops many features and uses the most useful ones only. Hence, Lasso Regression helps in selecting useful features as well.***

Using this new data frame, I've plotted the coefficient estimates with lasso regression for the following 5 predictors: room, residential zone, highway access, crime rate and tax as lambda(alpha) varies from 0-200. The plot below (figure 7) shows the resulting figure.

As we can see, the lines are converging very fast towards 0 as lambda increases. The

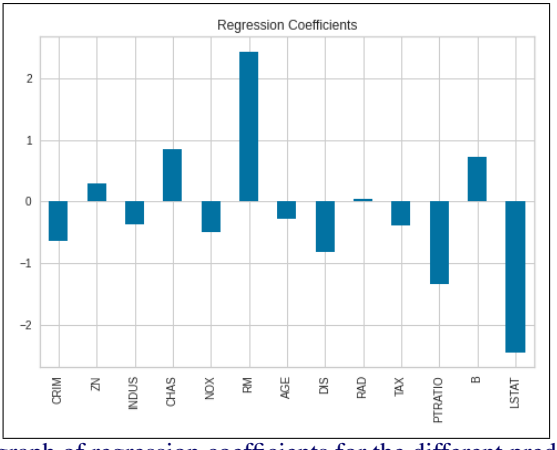
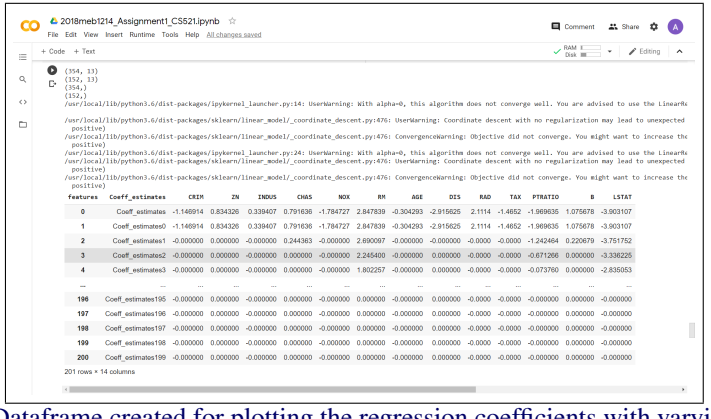


Figure 5: Bar graph of regression coefficients for the different predictor variables



features	Coeff_estimates	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	Coeff_estimates	-1.146914	0.834326	0.339407	0.791636	-1.784727	2.847839	-0.304293	-2.915625	2.1114	-1.4652	-1.989635	1.075678	-3.903107
1	Coeff_estimates0	-1.146914	0.834326	0.339407	0.791636	-1.784727	2.847839	-0.304293	-2.915625	2.1114	-1.4652	-1.989635	1.075678	-3.903107
2	Coeff_estimates1	-0.000000	0.000000	-0.000000	0.244363	-0.000000	2.690097	-0.000000	-0.000000	-0.0000	-0.0000	-1.242464	0.220679	-3.751752
3	Coeff_estimates2	-0.000000	0.000000	-0.000000	0.000000	-0.000000	2.245400	-0.000000	0.000000	-0.0000	-0.0000	-0.671266	0.000000	-3.336225
4	Coeff_estimates3	-0.000000	0.000000	-0.000000	0.000000	-0.000000	1.802257	-0.000000	0.000000	-0.0000	-0.0000	-0.073760	0.000000	-2.835053
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
196	Coeff_estimates195	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.0000	-0.0000	-0.000000	0.000000	-0.000000
197	Coeff_estimates196	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.0000	-0.0000	-0.000000	0.000000	-0.000000
198	Coeff_estimates197	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.0000	-0.0000	-0.000000	0.000000	-0.000000
199	Coeff_estimates198	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.0000	-0.0000	-0.000000	0.000000	-0.000000
200	Coeff_estimates199	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.0000	-0.0000	-0.000000	0.000000	-0.000000

Figure 6: Dataframe created for plotting the regression coefficients with varying lambda

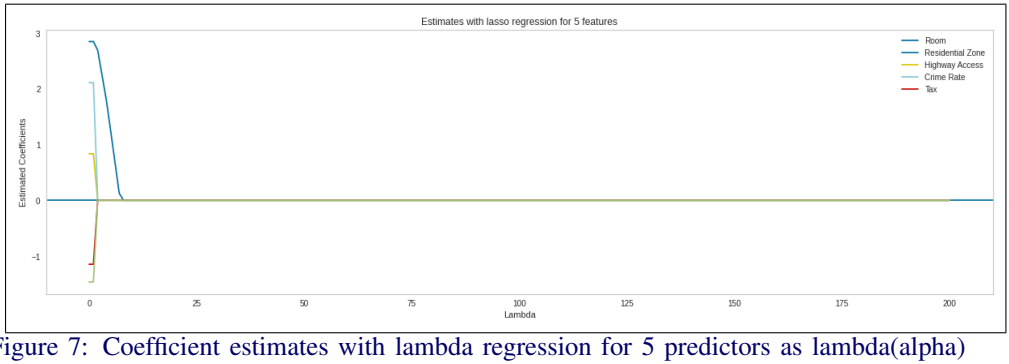


Figure 7: Coefficient estimates with lambda regression for 5 predictors as lambda(alpha) varies from 0-200

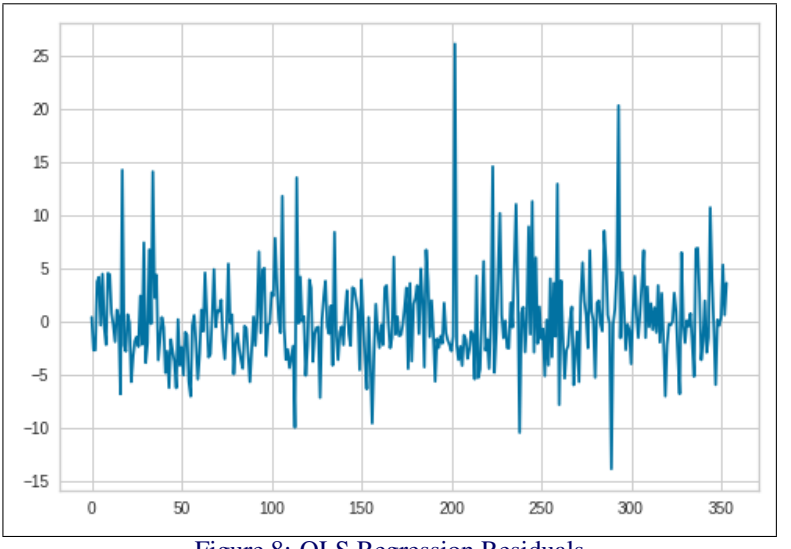


Figure 8: OLS Regression Residuals

feature Room (blue line) reaches 0 later as compared to other features while features like Crime Rate and Tax rush towards zero quickly.

Lastly, I've plotted the residuals as I had done in OLS Regression. (More on this in task 4).

3.4 Task 4: Plotting residuals obtained for the training data

OLS Regression: The residual plots have been plotted in the code along with OLS regression itself. The figures below show the graphs. residuals= y_train-y_train_predict

Ridge Regression: I have plotted the residuals in the code where I have trained ridge regression. They are also shown here in figure 10. These residuals are for 4 different values of lambda: 1,5,100,150. As we can see from here, they are almost similar as in ridge

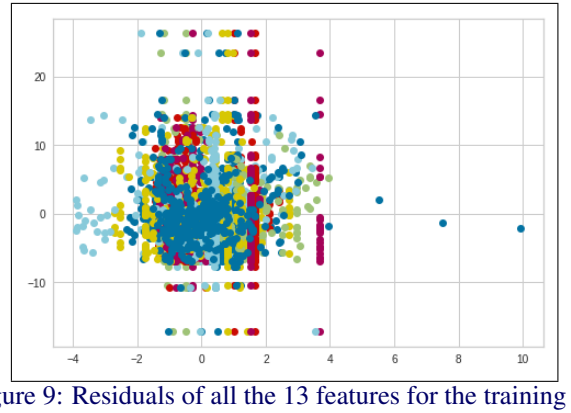


Figure 9: Residuals of all the 13 features for the training set

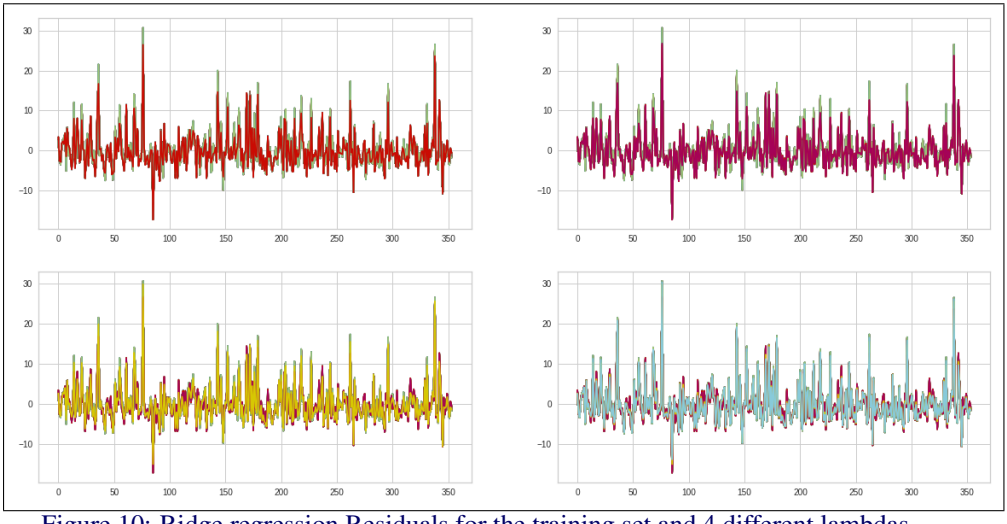


Figure 10: Ridge regression Residuals for the training set and 4 different lambdas

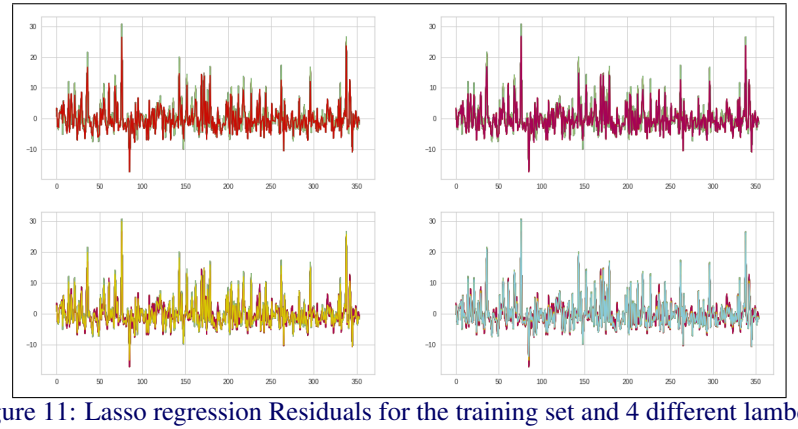


Figure 11: Lasso regression Residuals for the training set and 4 different lambdas

regression, lambda affects the model very less and coefficients converge to 0 very slowly.

Lasso Regression: The graphs have been plotted in the code along with Lasso regression itself. *Figure 11* shows the graph.

3.5 Task 5: Mean training and test errors for each of the above models.

1. OLS Regression: Mean Square error for the train set: 19.068341850927226 **Mean Square error for the test set:** 30.697037704088576 (Refer Table 1)

This is taken as a benchmark for comparing ridge and lasso regression whose test errors are expected to be better than the test error of OLS Regression.

2. Ridge Regression: The errors obtained through ridge regression are as follows: **Mean Square error for the train set:** 26.1336520487822 **Mean Square error for the test set:** 25.185886164823717 (Refer Table 2)

Part of Dataset	MSE
Training set	19.068341850927226
Test set	30.697037704088576

Table 1: OLS Regression

Part of Dataset	MSE
Training set	26.1336520487822
Test set	25.185886164823717

Table 2: Ridge Regression

Clearly, the error is higher on train set and lower on test set. Also, the test error is lower than OLS regression test error as expected. This is expected because ridge regression gives higher importance to more useful features (while not dropping unimportant features like Lasso regression) and also has a regularisation term which tries to prevent the model from overfitting. **A higher lambda(regularization) means less overfitting as it tries to smoothen out the predictions. A very high lambda means more underfitting.** Hence lambda tries to balance out the model bias and variance in order to give better predictions.

3. Lasso Regression:

The errors are: **Mean Square error for the train set:** 87.89652941364231 **Mean Square error for the test set:** 77.09851667615845 (Refer Table 3)

As we can see that the mean square error in Lasso regression is high. This is because, Lasso regression is such a model that the features are adjusted in a manner that the predictions come close to the actual values very fast, by adjusting the lambda. Hence, for higher lambdas we will find that errors are high, but the **errors will decrease very fast for lower lambdas**. This is also shown in the figure 12(MSE for Lasso Regression) for easy understanding and visualisation.

4 Conclusion

Most of the results and inferences are summarised in the respective sections above. In general, the ridge and lasso regression are better estimates for our data. **Ridge regression reduces the standard errors through introducing a regularisation parameter which tends to smoothen out the prediction function. It is different from OLS due to the fact that it has a penalty term as well.**

OLS Regression finds unbiased coefficients. This means that very large values may affect linear regression, hence we need to consider the bias as well. For this, ridge regression comes handy. There has to be a proper balance between bias and variance and this is where the difference arise between the various regression techniques. **Also, ridge regression gives**

Part of Dataset	MSE
Training set	87.89652941364231
Test set	77.09851667615845

Table 3: Lasso Regression

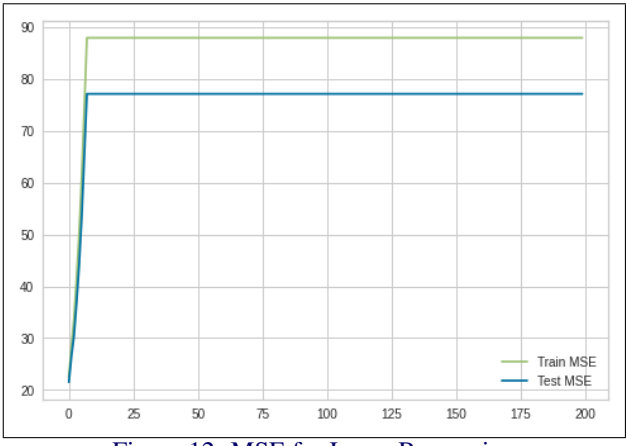


Figure 12: MSE for Lasso Regression

small and well distributed weights whereas lasso regression gives sparse weights with many being zeros. ***I realised that they are different in the sense that whether the model is penalised for its weights or not.***

Through my analysis and plots, I understood the differences between OLS, ridge and lasso regression and visualised my findings through useful plots and graphs.

5 References

[1] Kaggle dataset: Boston House Prices https://www.kaggle.com/vikrishnan/boston-
[2] Towards Data Science (Medium): https://towardsdatascience.com/ridge-regressio
[3] Geeks for Geeks: https://www.geeksforgeeks.org/ml-implementing-l1-and-l2-r
[4] Scikit Learn Residuals: https://www.scikit-yb.org/en/latest/api/regressor/residuals.html

[5] Regularization Medium Blog: https://towardsdatascience.com/how-to-use-residua
[6] Stack Exchange

[7] Documentation of Python Scikit Learn library