For the given dataset, we have 13 features and 506 samples. This denotes that $\boldsymbol{X}$ to be a 506x13 matrix. So that, $\boldsymbol{X^TX}$ will be a 13x13 matrix. The rank of the matrix $\boldsymbol{X^TX}$ is found as 13 by using NumPy library function that is given to us. If a nxn matrix have a rank of n, this will denote that the closed formula Ax = b will have a unique solution and therefore, it will be invertible. So, the result we found denotes that $\boldsymbol{X^TX}$ is invertible and we can have a unique solution for $\beta_{OLS.}$

I have implemented a linear regression model based on the given information as I used only the LSTAT column as X and used all data for training. The results and the plot are shown below.

```
The rank of XTX is  13
MSE of the data is  38.48296722989415
The coefficients of the dataset are B0 = 34.55384087938317   B1 =
-0.9500493537579962
```

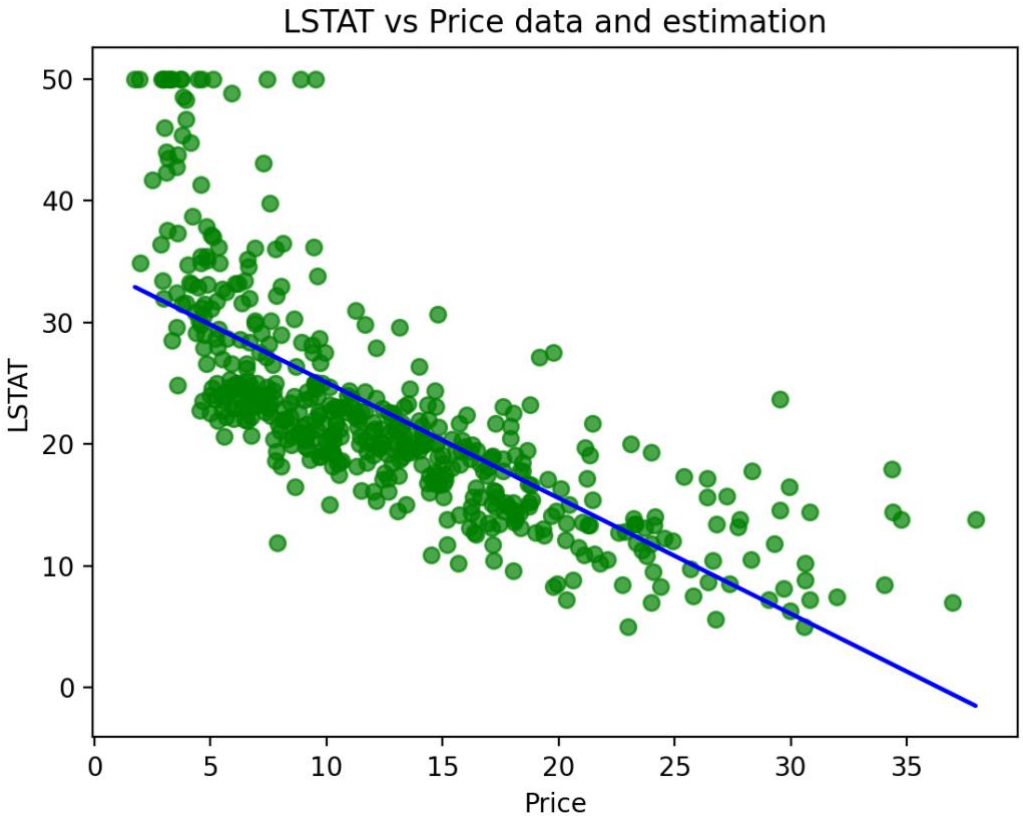
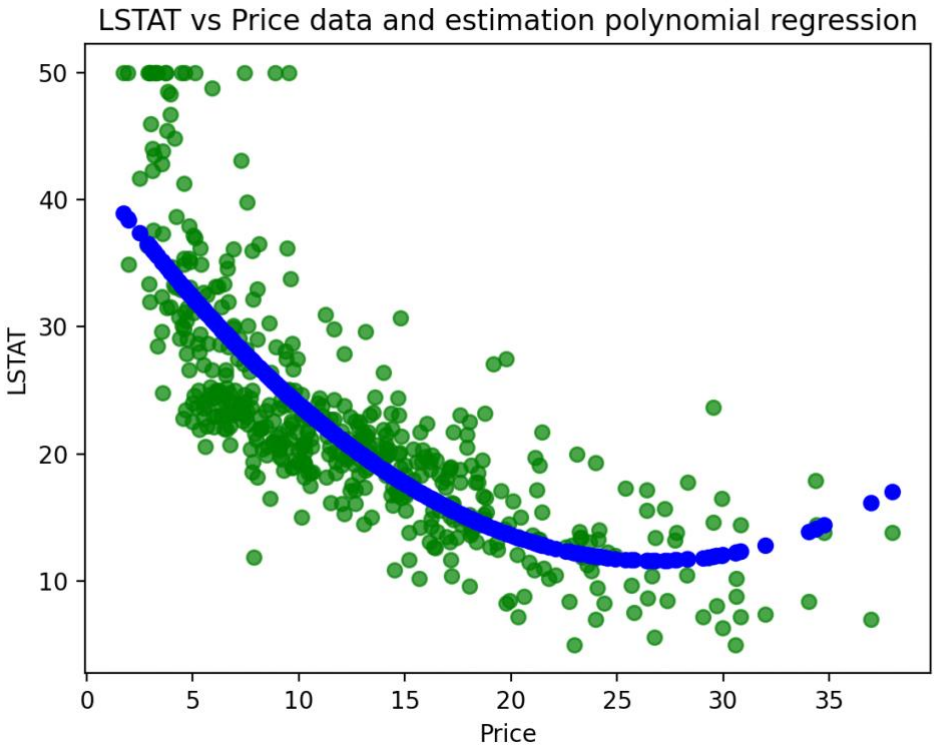


Figure: LSTAT vs Price ordinary linear regression estimation and data points

In this part, I trained a polynomial regression with 2nd degree polynomial. In this part, we will have a 3-dimensional weight different than the first. The weights, MSE and plot is shown below.

```
The coefficients of the dataset are B0 = 34.55384087938317   B1 =
-0.9500493537579962
MSE of the data is  30.330520075853716
The coefficients of the dataset with polynomial regressionare B0 =
[42.86200733]   B1 = [-2.3328211]  B2 =  [0.04354689]
```



In this part, implemented a gradient descent algorithm to the given data set consist of 3 features and binary labels. In the full batch method, the change in weights is calculated for all the samples in the training set and the weighted updates are done afterwards. In the mini batch method, I update the weights after batches and when an epoch ends, I shuffle the data in order to provide the stochastic gradient descent algorithm. I have done it for different learning rates and the results are shown below. There are five images, in every image, first part is for full batch and second part is for minibatch.

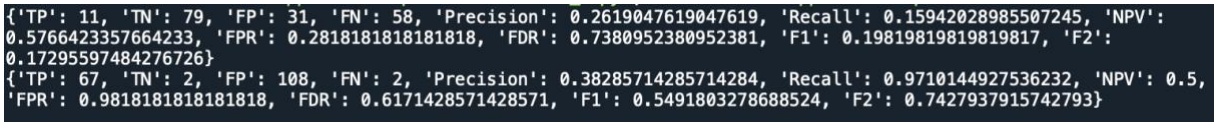
Learning rate = 0.1

```
{'TP': 11, 'TN': 78, 'FP': 32, 'FN': 58, 'Precision': 0.2558139534883721, 'Recall': 0.15942028985507245, 'NPV':
0.5735294117647058, 'FPR': 0.2909090909090909, 'FDR': 0.7441860465116279, 'F1': 0.1864285714285714, 'F2':
0.17241379310344826}
{'TP': 67, 'TN': 2, 'FP': 108, 'FN': 2, 'Precision': 0.38285714285714284, 'Recall': 0.9710144927536232, 'NPV': 0.5,
'FPR': 0.9818181818181818, 'FDR': 0.6171428571428571, 'F1': 0.5491803278688524, 'F2': 0.7427937915742793}
```

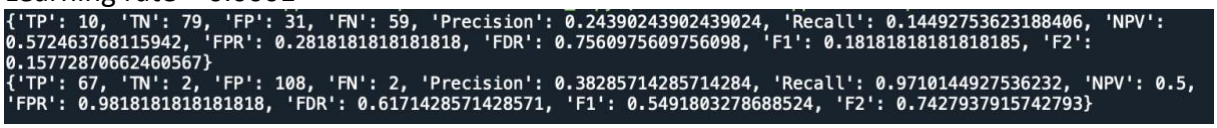
Learning rate = 0.01

```
{'TP': 11, 'TN': 80, 'FP': 30, 'FN': 58, 'Precision': 0.2682926829268293, 'Recall': 0.15942028985507245, 'NPV':
0.5797101449275363, 'FPR': 0.2727272727272727, 'FDR': 0.7317073170731707, 'F1': 0.19999999999999998, 'F2':
0.17350157728706622}
{'TP': 67, 'TN': 2, 'FP': 108, 'FN': 2, 'Precision': 0.38285714285714284, 'Recall': 0.9710144927536232, 'NPV': 0.5,
'FPR': 0.9818181818181818, 'FDR': 0.6171428571428571, 'F1': 0.5491803278688524, 'F2': 0.7427937915742793}
```

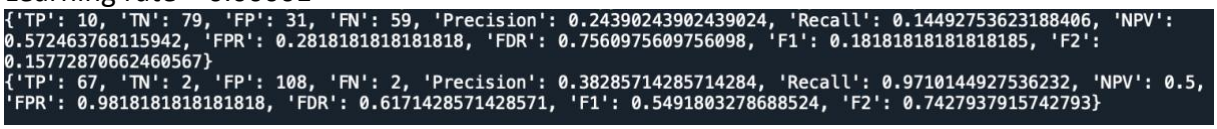
Learning rate = 0.001



Learning rate = 0.0001



Learning rate = 0.00001



Some applications do not tolerate false positive since it will be costly. In these types of applications, we can not only use accuracy and precision as the decisive metrics. The metrics such as NPV, FPR, FDR, F1 and F2 will be more informative when these type of applications are being held off since we can get more information about the performance of the data and we can avoid the costly results.