THE CHINESE UNIVERSITY OF HONG KONG, SHENZHEN

DDA3020

MACHINE LEARNING

Assignment 1 Report

Auther: *Student Number:*

Zhu Yuanhao *121090885*

Feb 26, 2021

Contents

1 Written Questions

1.1 Question 1

Since $X \in \mathbb{R}^{h\times d}$ and $y \in \mathbb{R}^{h\times 1}$, $w \in \mathbb{R}^{d\times 1}$, thus $y^T X w \in \mathbb{R}^{1\times 1}$. Let's assume

$$y^T = [y_1 \quad ... \quad y_h], \quad w = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad X = \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{h1} & \cdots & x_{hd} \end{bmatrix} \tag{1}$$

Then

$$\begin{aligned} y^T X w &= \sum_{k=1}^{h}\sum_{j=1}^{d} w_j y_k x_{kj} \\ &= w_1 \sum_{k=1}^{h} y_k x_{k1} + \cdots + w_d \sum_{k=1}^{h} y_k x_{kd} \end{aligned} \tag{2}$$

$$\frac{d(y^T X w)}{dw} = \begin{bmatrix} \frac{\partial \sum_{k=1}^{h}\sum_{j=1}^{d} w_j y_k x_{kj}}{\partial w_1} \\ \vdots \\ \frac{\partial \sum_{k=1}^{h}\sum_{j=1}^{d} w_j y_k x_{kj}}{\partial w_d} \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{h} y_k x_{k1} \\ \vdots \\ \sum_{k=1}^{h} y_k x_{kd} \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{h1} \\ \vdots & \ddots & \vdots \\ x_{1d} & \cdots & x_{hd} \end{bmatrix} \times \begin{bmatrix} y_1 \\ \vdots \\ y_h \end{bmatrix} = X^T y \tag{3}$$

Also we have

$$w^T w = \sum_{k=1}^{d} w_k^2 \tag{4}$$

Thus

$$\frac{d(w^T w)}{dw} = \begin{bmatrix} \frac{\partial \sum_{k=1}^{d} w_k^2}{\partial w_1} \\ \vdots \\ \frac{\partial \sum_{k=1}^{d} w_k^2}{\partial w_d} \end{bmatrix} = \begin{bmatrix} 2w_1 \\ \vdots \\ 2w_d \end{bmatrix} = 2w \tag{5}$$

If $X \in \mathbb{R}^{d \times d}$ and $w \in \mathbb{R}^{d \times 1}$, then

$$\begin{aligned} w^T X w &= \sum_{k=1}^{d} \sum_{j=1}^{d} w_j w_k x_{kj} \\ &= w_1 \sum_{k=1}^{d} w_k x_{k1} + \cdots + w_d \sum_{k=1}^{d} w_k x_{kd} \end{aligned} \tag{6}$$

We have

$$\begin{aligned} \frac{d(w^T X w)}{dw} &= \begin{bmatrix} \frac{\partial \sum_{k=1}^{d} \sum_{j=1}^{d} w_j w_k x_{kj}}{\partial w_1]} \\ \vdots \\ \frac{\partial \sum_{k=1}^{d} \sum_{j=1}^{d} w_j w_k x_{kj}}{\partial w_d]} \end{bmatrix} = \begin{bmatrix} 2w_1 x_{11} + \sum_{k=2}^{d} w_k x_{k1} + \sum_{k=2}^{d} w_k x_{1k} \\ \vdots \\ 2w_d x_{dd} + \sum_{k=1}^{d-1} w_k x_{kd} + \sum_{k=1}^{d-1} w_k x_{dk} \end{bmatrix} \\ &= \begin{bmatrix} 2x_{11} w_1 + w_2(x_{21} + x_{12}) + \cdots + w_d(x_{d1} + x1d) \\ \vdots \\ 2x_{dd} w_d + w_1(x_{d1} + x_{1d}) + \cdots + w_{d-1}(x_{d(d-1)} + x(d-1)d) \end{bmatrix} \\ &= \begin{bmatrix} 2x_{11} & x_{12} + x_{21} & \cdots & x_{1d} + x_{d1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1d} + x_{d1} & x_{2d} + x_{d2} & \cdots & 2x_{dd} \end{bmatrix} \times \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} = (X + X^T) w \end{aligned} \tag{7}$$

1.2 Question 2

(1) Let

$$X = \begin{bmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_N \end{bmatrix}^T, w = [b \quad w_1 \quad \cdots \quad w_d]^T, y = [y_1 \quad \cdots \quad y_N]$$

$$\begin{aligned} &\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2 + \lambda \bar{w}^T \bar{w} \\ &= (Xw - y)^T (Xw - y) + \lambda w^T \hat{I}_d \hat{I}_d w \\ &= (w^T X^T - y^T)(Xw - y) + \lambda w^T \hat{I}_d w \\ &= w^T X^T X w - w^T X^T y - y^T X w + y^T y + \lambda w^T \hat{I}_d w \end{aligned} \tag{8}$$

$$\begin{aligned} &\frac{d(\sum_{i=1}^{N}(f_{w,b}(x_i) - y_i)^2 + \lambda \bar{w}^T \bar{w})}{dw} \\ &= \frac{d(w^T (X^T X)^T w)}{dw} - \frac{d(w^T X^T y)}{dw} - \frac{d(y^T X w)}{dw} + \lambda \frac{d(w^T \hat{I}_d w)}{dw} \\ &= 2X^T X w - X^T y - X^T y + 2\lambda \hat{I}_d w \\ &= 2(X^T X + \lambda \hat{I}_d) w - 2X^T y \end{aligned} \tag{9}$$

Since it is the closed-form solution, we need the first derivative to be 0.

$$2(X^T X + \lambda \hat{I}_d) w - 2X^T y = 0 \implies w = (X^T X + \lambda \hat{I}_d)^{-1} X^T y \tag{10}$$

Where $(X^T X + \lambda \hat{I}_d)$ is invertible, and we have the second order derivative

$$X^T X + \lambda \hat{I}_d \succ 0 \tag{11}$$

(2) Use gradient descent

$$w \leftarrow w - \alpha \frac{\partial J(w)}{\partial w} \tag{12}$$

Where

$$\frac{\partial J(w)}{\partial w} = 2(X^T X + \lambda \hat{I}_d)w - 2X^T y \tag{13}$$

1.3 Question 3

(1) $f(x) = x^2 \in C^2$ and

$$f^{''}(x) = 2 > 0 \tag{14}$$

For any $x \in \mathbb{R}$, thus $f(x)$ is convex.

(2) For any $\theta \in (0,1), x \in \mathbb{R}, y \in \mathbb{R}$

$$\begin{aligned} f(\theta x + (1-\theta)y) &= a(\theta x + (1-\theta)y) + b \\ &= \theta(ax+b) + (1-\theta)(ay+b) \\ &= \theta f(x) + (1-\theta) f(y) \\ &\leq \theta f(x) + (1-\theta) f(y) \end{aligned} \tag{15}$$

Thus $f(x)$ is convex, but not strictly convex since the equal sign can hold.

(3) For any $\theta \in (0,1), x \in \mathbb{R}, y \in \mathbb{R}$

$$\begin{aligned} f(\theta x + (1-\theta)y) &= |\theta x + (1-\theta)y| \\ &\leq \theta|x| + (1-\theta)|y| \\ &= \theta f(x) + (1-\theta) f(y) \end{aligned} \tag{16}$$

Thus $f(x)$ is convex, however, if we pick $x \in \mathbb{R}_+, y \in \mathbb{R}_+$,

$$\begin{aligned} f(\theta x + (1-\theta)y) &= |\theta x + (1-\theta)y| \\ &= \theta x + (1-\theta)y \\ &= \theta|x| + (1-\theta)|y| \\ &= \theta f(x) + (1-\theta) f(y) \end{aligned} \tag{17}$$

Thus not strictly convex.

1.4 Question 4

The probability density function of $Laplace(\mu, b)$ is

$$f(x) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}, b > 0 \tag{18}$$

To maximize the likelihood function

$$L(x_1, x_2, ..., x_n | \mu, b) = \frac{1}{(2b)^n} e^{-\frac{1}{b}\sum_{k=1}^{n} |x_k - \mu|} \tag{19}$$

Take logarithm, we get the problem equivalent to

$$\min_{\mu,b} \quad n\log(2b) + \frac{1}{b}\sum_{k=1}^{n}|x_k - \mu| \tag{20}$$

By observing the term $\sum_{k=1}^{n}|x_k - \mu|$, we know that it equals to the total distance between x_i and μ, thus $\mu = \hat{\mu}$, where $\hat{\mu}$ shall be the median of the $x_i's$.

Take derivative respect to b, and let it be 0

$$\begin{aligned} &\frac{n}{b} - \frac{1}{b^2}\sum_{k=1}^{n}|x_k - \hat{\mu}| = 0 \\ \implies &b = \frac{1}{n}\sum_{k=1}^{n}|x_k - \hat{\mu}| \end{aligned} \tag{21}$$

2 Programming Question

2.1 Model Structure

In this assignment, we implement a multivariable linear regression. We use several predictors (will show in 2.1.4) to fit and predict the dependent variable.

2.1.1 Hyperparameter Settings

There are several parameters to be set, they are step size, initial regression parameters and iteration times.

Step size measures the step size in every round of gradient descent to modify the regression parameters, which is set to 0.001 as default.

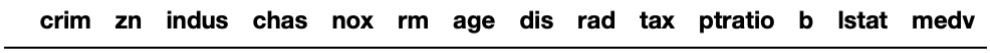
Initial regression parameters is set by own judgement, and the true iteration times and accuracy will depend on it. It is all set to 0.1 as default.

Iteration times measures how many times we use the gradient descent method, the more iteration time is, the more accuracy the result is (generally). It is set to 100 as default.

2.1.2 Data Loading

Use pandas library and read_csv() function to load the data of Boston Housing Dataset. We can see there are 14 columns and 506 rows.

First check whether there exists blank, i.e. NaN or Null, use the function of isnull() and DataFrame[DataFrame.isnull().values==True] function to print all the rows that contain NA data. 2.1.2 shows that there is no such kind of data.



Figur 1: Rows that contain NA data

Second, we check the data type and summarise the data features of each column, include the counts, mean, std, min, max, 25%, 50%, 75% percentiles, and get the result 2.1.2. We firstly guess that RM is the most relevant attribute to MEDV since the average number of rooms per dwelling directly affect the price.

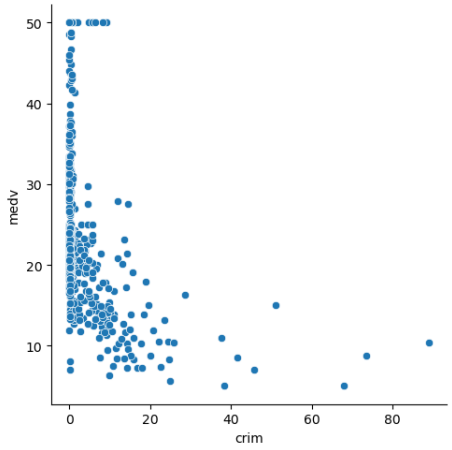
	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000

Figur 2: Summary statistics of the attributes (partial)

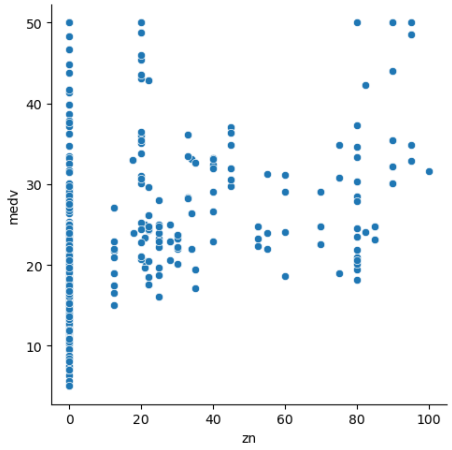
2.1.3 Visualization

We plot MEDV over each attribute to take a brief look.

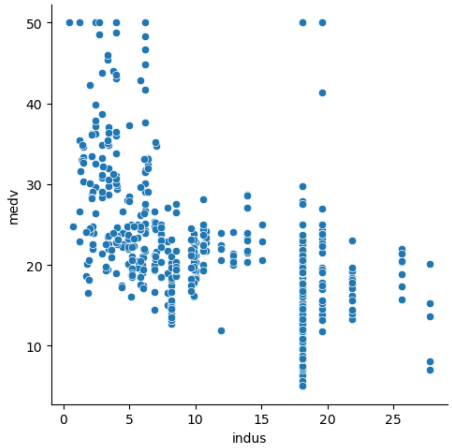
We use the seaborn library to plot the data and get the results below. We can easily see that, the price go high with higher RM, and go down with LSTAT. Now LSTAT maybe the most relevant attribute since it varies the most widely and have more clear relationship.



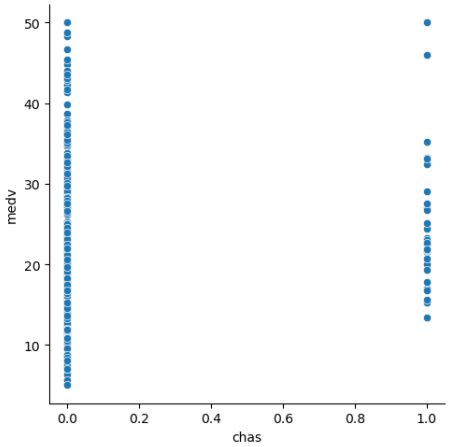
Figur 3: crim



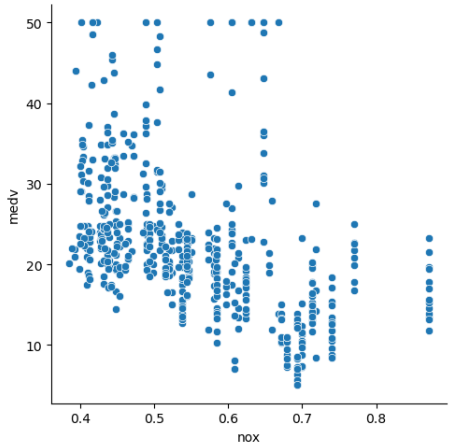
Figur 4: zn



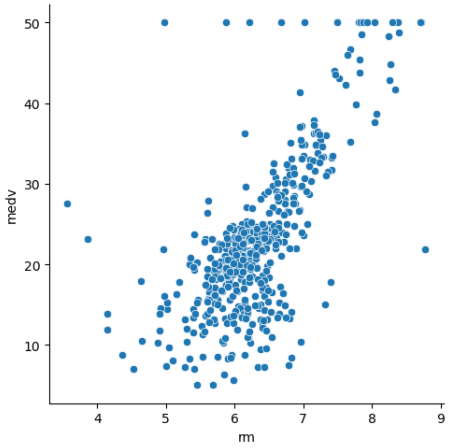
Figur 5: indus



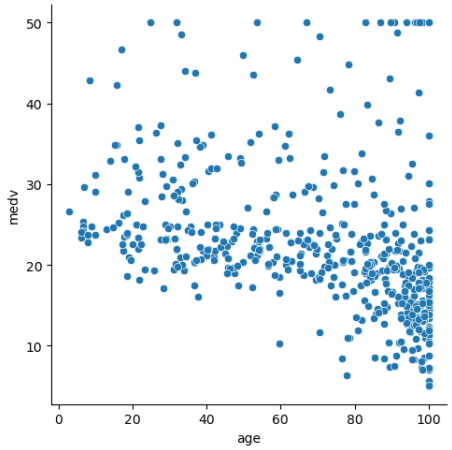
Figur 6: chas



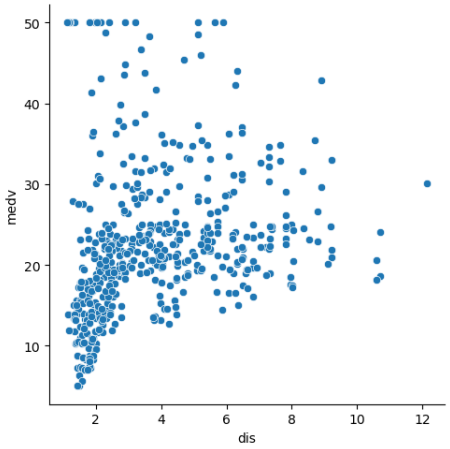
Figur 7: nox



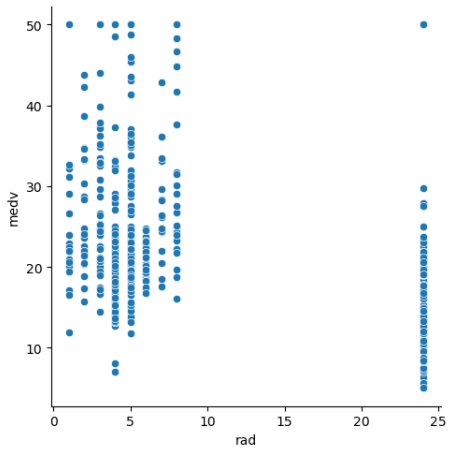
Figur 8: age



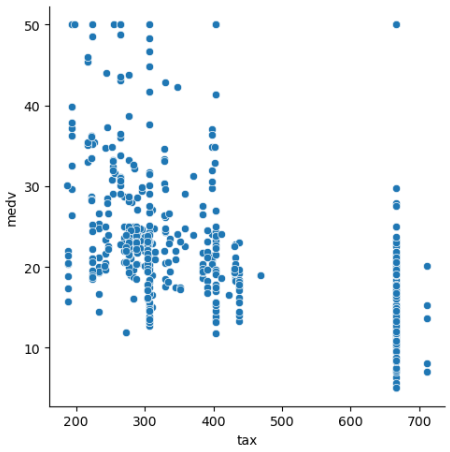
Figur 9: rad



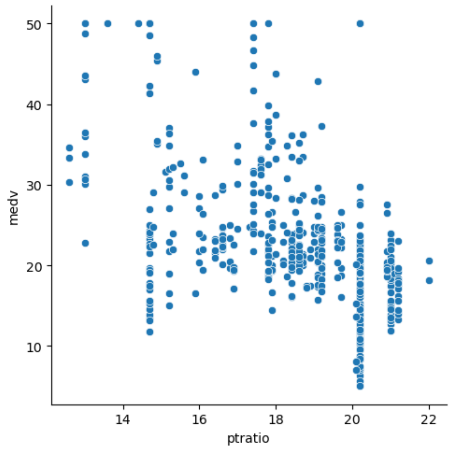
Figur 10: rm



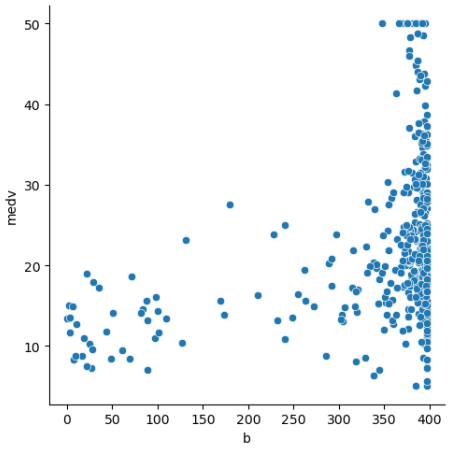
Figur 11: tax



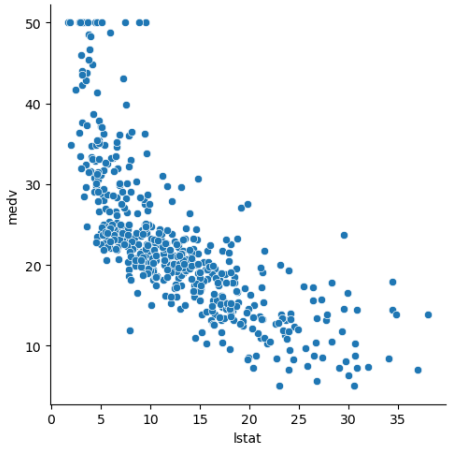
Figur 12: dis



Figur 13: ptratio



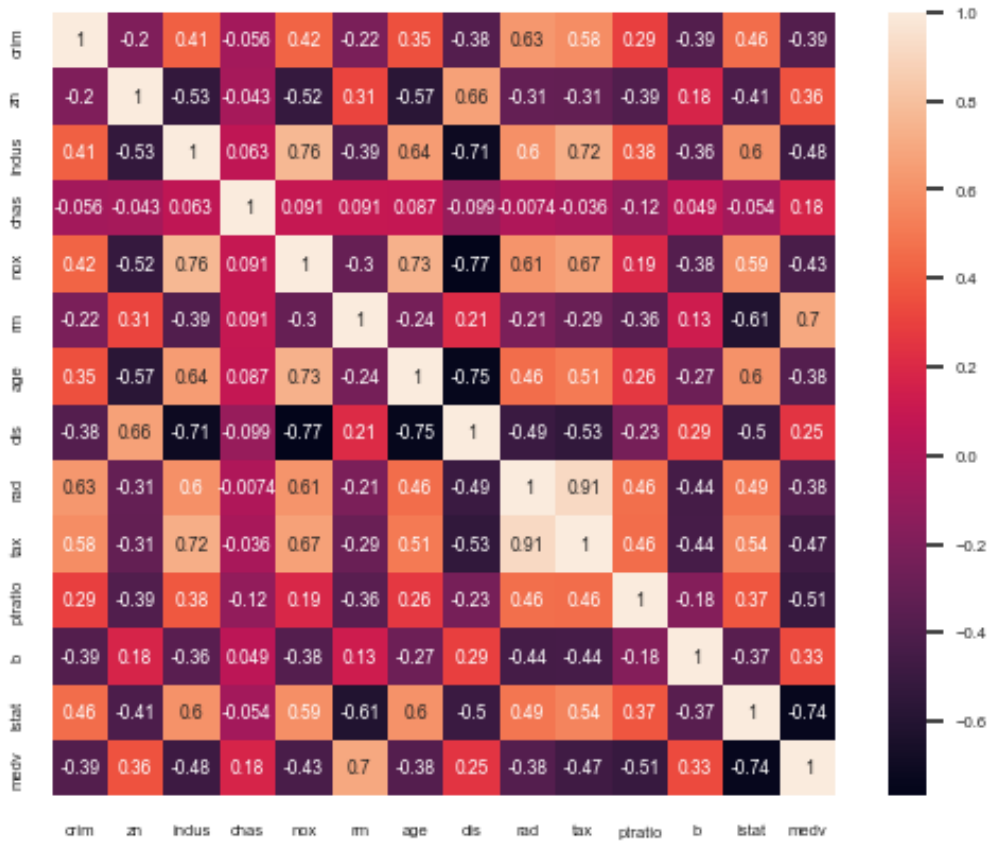
Figur 14: b



Figur 15: lstat

2.1.4 Heatmap and Predictors selection

In order to select suitable predictors, we can take a look at the heatmap. Use seaborn.heatmap function to draw the correlation coefficient heatmap of all the attributes, including the dependent variable, and get the result 2.1.4. We simply select the attributes that have correlation coefficient larger than 0.4 with MEDV to be the predictors in our model, i.e. choose INDUS, NOX, RM, TAX, PTRATIO and LSTAT to be the predictors.

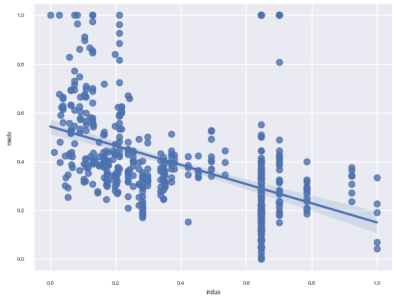


Figur 16: Heatmap

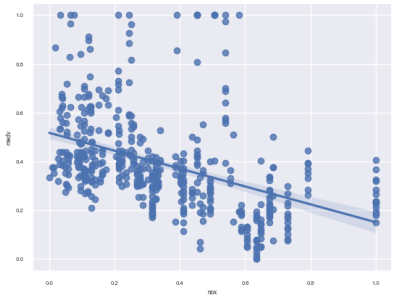
2.1.5 Preprocessing and Relevance Plot

Since the scale of the data are different, so one way is to use the MinMaxScaler function to scale the data to the range of $[0, 1]$, to better do regression later.

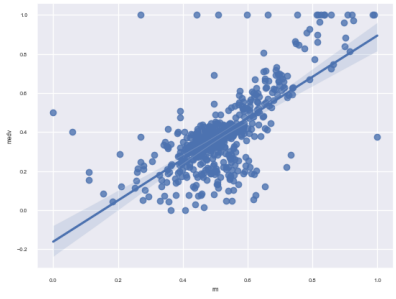
After scaling those data, we use seaborn.regplot to plot the relevance of these columns against MEDV with 95% confidence interval.



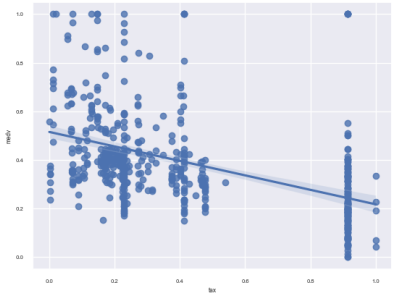
Figur 17: indus



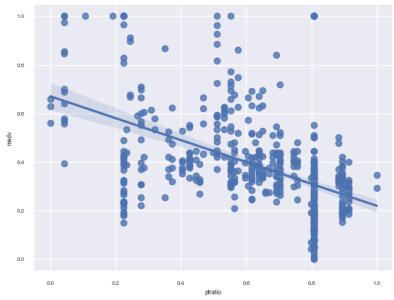
Figur 18: nox



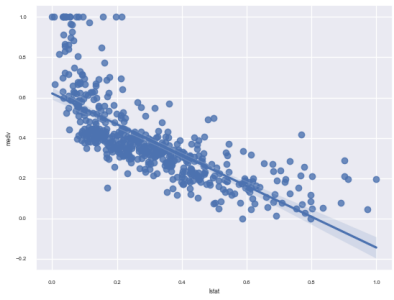
Figur 19: rm



Figur 20: tax



Figur 21: ptratio



Figur 22: lstat

2.1.6 Model Training

Use random.sample function to randomly split the data into two parts, one contains 80% of the samples and the other contains 20% of the samples, i.e, since there are 506 rows, we choose 405 rows of the data as the training data, and rest as the testing data. Use the first part of the data as the training data to train the linear regression model which is

$$f_{w,b}(x) = x^T w + b \tag{22}$$

We use the gradient descent medthod to update the parameters as the step of

$$w \leftarrow w - \alpha(X^T(Xw - y)) \tag{23}$$

where w and α are set as default.

2.2 Loss Analysis

2.2.1 Loss and RMSE

We can use the squared error loss to be our loss function, which is

$$J(w) = (Xw - y)^T(Xw - y) \tag{24}$$

And also we can see the RMSE, the Root mean squared error,

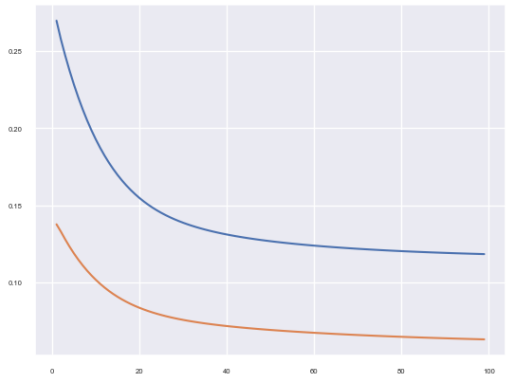
$$RMSE = \sqrt{\frac{SSE}{n}} = \sqrt{\frac{(Xw - y)^T(Xw - y)}{n}} \tag{25}$$

Where n is the size of the data.

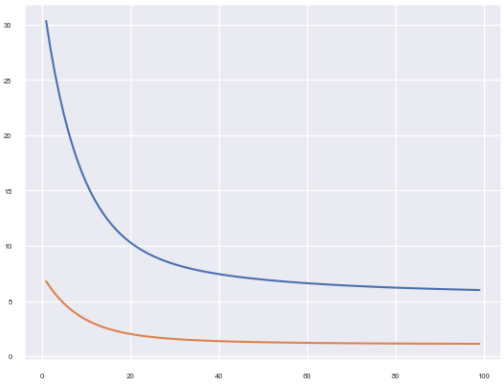
2.2.2 Training and Testing Error

We draw the loss function and RMSE of the training data as curves over the iteration time, which are shown as blue curves below. In addition, we add a different orange line, which measures the error of the testing procedure.

Finally, we get the RMSE as 0.06317266293025348 and total loss as 1.6162680633882656.



Figur 23: RMSE



Figur 24: Total Loss Function

2.3 Parameters Analysis

RMSE will be influenced by different parameters such as step size, iteration steps, etc. So here we repeat the splitting, training, and testing for 10 times.

Set the initial step size as 0.0000001 and iteration time as 100.

We first use 10 loops to see the effect of the step size. And each time enlarge it to 10 times than before, and can get the result below 2.3. We can see that when the step

size is very small, RMSE will be lower as the step size grow, however, RMSE will be bigger even blow up when the step size reaches out some bound and keeps growing.

```
0 0.13773078936801894
1 0.1376905013530665 7
2 0.13728856487778818
3 0.1333231539380151
4 0.10182282690143452
5 1628104.4642545835
6 4105965834784454.5
7 4.4813392871759575e+24
8 4.5205038477908594e+33
9 4.5244369872811174e+42
```

Figur 25: RMSE with different step size

Secondly we see the effect of the iteration time. We use 5 loops to see the result (since 10 times will be too time consuming). Each time enlarge it to 1o times than before, and we get the result below. Obviously, RMSE becomes smaller as the iteration time grows.

```
0 0.10182282690143452
1 0.0631726629302534 8
2 0.0585770111044610 5
3 0.0585817968879286 25
4 0.0585817968879301 86
```

Figur 26: RMSE with different iteration times