hw1 - writing.

杨景兰 12190699

**1.1** Given the following denominator layout derivatives, (13 points)

- **Differentiation of a scalar function *w.r.t.* a vector**: If $f(\mathbf{w})$ is a scalar function of $d$ variables, $\mathbf{w}$ is a $d \times 1$ vector, then differentiation of $f(\mathbf{w})$ *w.r.t.* $\mathbf{w}$ results in a $d \times 1$ vector

$$\frac{df(\mathbf{w})}{d\mathbf{w}} = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}$$

- **Differentiation of a vector function *w.r.t.* a vector**: If $\mathbf{f(w)}$ is a vector function of size $h \times 1$ and $\mathbf{w}$ is a $d \times 1$ vector, then differentiation of $\mathbf{f}(\mathbf{w})$ *w.r.t.* $\mathbf{w}$ results in a $d \times h$ matrix

$$\frac{d\mathbf{f}(\mathbf{w})}{d\mathbf{w}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_h}{\partial w_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial w_d} & \cdots & \frac{\partial f_h}{\partial w_d} \end{bmatrix}$$

**Please prove the following derivatives:**
Consider $\mathbf{X} \in \mathbb{R}^{h \times d}$ and $\mathbf{y} \in \mathbb{R}^{h \times 1}$, which are not functions of $\mathbf{w}$:

$$\frac{d(\mathbf{y}^\top \mathbf{X}\mathbf{w})}{d\mathbf{w}} = \mathbf{X}^\top \mathbf{y}, \quad (4 \text{ points})$$

$$\frac{d(\mathbf{w}^\top \mathbf{w})}{d\mathbf{w}} = 2\mathbf{w}, \quad (4 \text{ points})$$

Consider $\mathbf{X} \in \mathbb{R}^{d \times d}$ and $\mathbf{w} \in \mathbb{R}^{d \times 1}$ (5 points):

$$\frac{d(\mathbf{w}^\top \mathbf{X}\mathbf{w})}{d\mathbf{w}} = (\mathbf{X} + \mathbf{X}^\top)\mathbf{w}$$

proof: ① $y \in \mathbb{R}^{n\times 1}$, $X \in \mathbb{R}^{n\times d}$, $w \in \mathbb{R}^{d\times 1}$

$$y^T X = [y_1, \cdots, y_n] \cdot \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{bmatrix} = [y_1 x_{1,1} + \cdots + y_n x_{n,1}, \cdots, y_1 x_{1,d} + \cdots + y_n x_{n,d}] \in \mathbb{R}^{1\times d}$$

$$(y^T X) w = [y_1 x_{1,1} + \cdots + y_n x_{n,1}, \cdots, y_1 x_{1,d} + \cdots + y_n x_{n,d}] \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} = (y_1 x_{1,1} + \cdots + y_n x_{n,1}) \cdot w_1 + \cdots + (y_1 x_{1,d} + \cdots + y_n x_{n,d}) w_d \in \mathbb{R}^{1\times 1}$$

⇒ differentiation of a scalar function w.r.t. a vector:

$$\frac{d(y^T X w)}{dw} = \begin{bmatrix} \frac{\partial[(y_1 x_{1,1} + \cdots + y_n x_{n,1}) \cdot w_1 + \cdots + (y_1 x_{1,d} + \cdots + y_n x_{n,d}) w_d]}{\partial w_1} \\ \vdots \\ \frac{\partial[(y_1 x_{1,1} + \cdots + y_n x_{n,1}) \cdot w_1 + \cdots + (y_1 x_{1,d} + \cdots + y_n x_{n,d}) w_d]}{\partial w_d} \end{bmatrix} = \begin{bmatrix} y_1 x_{1,1} + \cdots + y_n x_{n,1} \\ \vdots \\ y_1 x_{1,d} + \cdots + y_n x_{n,d} \end{bmatrix}$$

$$X^T y = \begin{bmatrix} x_{1,1} & \cdots & x_{n,1} \\ \vdots & \ddots & \vdots \\ x_{1,d} & \cdots & x_{n,d} \end{bmatrix} \times \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} y_1 x_{1,1} + \cdots + y_n x_{n,1} \\ \vdots \\ y_1 x_{1,d} + \cdots + y_n x_{n,d} \end{bmatrix} = \frac{d(y^T X w)}{dw}$$

Hence, $\frac{d(y^T X w)}{dw} = X^T y$

② $w \in \mathbb{R}^{d\times 1}$:

$$w^T w = [w_1, \cdots, w_d] \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} = w_1^2 + \cdots + w_d^2 \in \mathbb{R}^{1\times 1}$$

⇒ Differentiation of a scalar function w.r.t. a vector:

$$\frac{d(w^T w)}{dw} = \begin{bmatrix} \frac{\partial(w_1^2 + \cdots + w_d^2)}{\partial w_1} \\ \vdots \\ \frac{\partial(w_1^2 + \cdots + w_d^2)}{\partial w_d} \end{bmatrix} = \begin{bmatrix} 2w_1 \\ \vdots \\ 2w_d \end{bmatrix} = 2w$$

Hence, $\frac{d(w^T w)}{dw} = 2w$

③ $w \in \mathbb{R}^{d\times 1}$, $X \in \mathbb{R}^{d\times d}$:

$$w^T X = [w_1, \cdots, w_d] \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{d,1} & \cdots & x_{d,d} \end{bmatrix} = [w_1 x_{1,1} + \cdots + w_d x_{d,1}, \cdots, w_1 x_{1,d} + \cdots + w_d x_{d,d}] \in \mathbb{R}^{1\times d}$$

$$(w^T X) w = [w_1 x_{1,1} + \cdots + w_d x_{d,1}, \cdots, w_1 x_{1,d} + \cdots + w_d x_{d,d}] \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} = [w_1 (w_1 x_{1,1} + \cdots + w_d x_{d,1}) + \cdots + w_d (w_1 x_{1,d} + \cdots + w_d x_{d,d})] \in \mathbb{R}^{1\times 1}$$

⇒ Differentiation of a scalar function w.r.t. a vector:

$$\frac{d(w^T X w)}{dw} = \begin{bmatrix} \frac{\partial[w_1 (w_1 x_{1,1} + \cdots + w_d x_{d,1}) + \cdots + w_d (w_1 x_{1,d} + \cdots + w_d x_{d,d})]}{\partial w_1} \\ \vdots \\ \frac{\partial[w_1 (w_1 x_{1,1} + \cdots + w_d x_{d,1}) + \cdots + w_d (w_1 x_{1,d} + \cdots + w_d x_{d,d})]}{\partial w_d} \end{bmatrix} = \begin{bmatrix} (w_1 x_{1,1} + \cdots + w_d x_{d,1}) + (w_1 + \cdots + w_d) x_{1,d} \\ \vdots \\ (w_1 x_{1,d} + \cdots + w_d x_{d,d}) + (w_1 + \cdots + w_d) x_{d,d} \end{bmatrix} = \begin{bmatrix} 2w_1 x_{1,1} + \cdots + (x_{1,d} + x_{d,1}) w_d \\ \vdots \\ (x_{1,d} + x_{d,1}) w_1 + \cdots + 2 x_{d,d} w_d \end{bmatrix}$$

$$(X+X^T)W = \left( \begin{bmatrix} X_{1,1} & \cdots & X_{1,d} \\ \vdots & \ddots & \vdots \\ X_{d,1} & \cdots & X_{d,d} \end{bmatrix} + \begin{bmatrix} X_{1,1} & \cdots & X_{d,1} \\ \vdots & \ddots & \vdots \\ X_{1,d} & \cdots & X_{d,d} \end{bmatrix} \right) \cdot \begin{bmatrix} W_1 \\ \vdots \\ W_d \end{bmatrix} = \begin{bmatrix} 2X_{1,1} & \cdots & X_{1,d}+X_{d,1} \\ \vdots & \ddots & \vdots \\ X_{1,d}+X_{d,1} & \cdots & 2X_{d,d} \end{bmatrix} \cdot \begin{bmatrix} W_1 \\ \vdots \\ W_d \end{bmatrix} = \begin{bmatrix} 2X_{1,1}W_1 + \cdots + (X_{1,d}+X_{d,1})W_d \\ (X_{2,1}+X_{1,2})W_1 + \cdots + (X_{2,d}+X_{d,2})W_d \\ \vdots \\ (X_{1,d}+X_{d,1})W_1 + \cdots + 2X_{d,d}W_d \end{bmatrix} = \frac{d(W^TXW)}{dW}$$

Hence, $\frac{d(W^TXW)}{dW} = (X+X^T)W$

**1.2** Suppose we have training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}^d, i = 1, 2, \ldots, N$. Consider $f_{w,b}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b$. (12 points)

(1) Find the closed-form solution of the following problem

$$\min_{\mathbf{w},b} \sum_{i=1}^{N} (f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2 + \lambda \bar{\mathbf{w}}^\top \bar{\mathbf{w}}, \tag{1}$$

where $\bar{\mathbf{w}} = [b, w_1, w_2, \ldots, w_d]^\top$. (6 points)

($\bar{\mathbf{w}} \in \mathbb{R}^{(d+1)\times(d+1)}$)

1) $f_{w,b}(x) = x^TW + b = XW = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} W = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,d} \end{bmatrix} \begin{bmatrix} W_{0,1} & \cdots & W_{0,d} \\ W_{1,1} & \cdots & W_{1,d} \\ \vdots & \ddots & \vdots \\ W_{d,1} & \cdots & W_{d,d} \end{bmatrix} = \begin{bmatrix} f_{1,1} & \cdots & f_{1,n} \\ \vdots & & \\ f_{N,1} & \cdots & f_{N,d} \end{bmatrix}$

($X \in \mathbb{R}^{N\times(1+d)}$, $W \in \mathbb{R}^{(1+d)\times d}$; sample 1's output; sample N's output.)

$E = XW - Y$

$J(W) = \text{trace}(E^TE)$

$= \text{trace}[(XW-Y)^T(XW-Y)]$

$\min_{W,b} \sum_{i=1}^{N} (f_{w,b}(x_i) - y_i)^2 + \lambda \bar{w}^T\bar{w}$

$= \min_{W} (XW-Y)^T(XW-Y) + \lambda \bar{w}^TW$

$\frac{\partial}{\partial W}(XW-Y)^T(XW-Y) + \lambda \bar{w}^TW = 0$

$\Rightarrow 2X^TXW - 2X^TY + 2\lambda \hat{I}_d W = 0$

$\Rightarrow (X^TX + \lambda \hat{I}_d)W = X^TY$

$\Rightarrow W = (X^TX + \lambda \hat{I}_d)^{-1}X^TY$

Note that $(X^TX + \lambda \hat{I}_d)^{-1}$ is guaranteed to be invertible given $\lambda > 0$.

(2) Show how to use gradient descent to solve the problem. (6 points)

Assume $f(W) = \sum_{i=1}^{N} (f_{w,b}(x_i) - y_i)^2 + \lambda \bar{w}^T\bar{w}$

$= (XW-Y)^T(XW-Y) + \lambda \bar{w}^TW$

start from $W_0$, set $\varepsilon > 0$, $k = 0$.

① If $\| 2X^TXW_k - 2X^TY + 2\lambda \hat{I}_d W_k \| < \varepsilon$, output $W_k$.

Otherwise, continue.

② compute search direction $d^k = -(2X^TXW_k - 2X^TY + 2\lambda \hat{I}_d W_k)$

③ choose step size $\alpha_k$ via exact line search or backtracking line search.

④ set $W_{k+1} = W_k + \alpha_k d^k$, $k = k+1$

go back to step 1.

**1.3** Prove that:

(1) $f(x) = x^2$ is convex. (4 points)

(2) every affine function $f(x) = ax + b$ is convex, but not strictly convex; (4 points)

(3) $f(x) = |x|$ is convex, but not strictly convex; (5 points)

Proof: (1) $f(x) = x^2$, $x \in R$.

$\Rightarrow f'(x) = 2x$

$\Rightarrow f''(x) = 2 > 0$

$\forall x \in R$, $f''(x) > 0$. Therefore, $f(x)$ is convex.

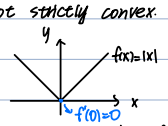
(2) $f(x) = ax + b$

$\Rightarrow f'(x) = a$

$\Rightarrow f''(x) = 0$

$\forall x \in R$, $f''(x)=0$. Therefore, $f(x)$ is convex but not strictly convex.

(3) $f(x)=|x|$




Suppose $\forall x,y \in R$, $\forall 0 \le \alpha \le 1$.

$f[\alpha x+(1-\alpha)y] = |\alpha x+(1-\alpha)y| \le |\alpha x|+|(1-\alpha)y| = \alpha|x|+(1-\alpha)|y| = \alpha f(x)+(1-\alpha)|y|$

When $xy \ge 0$, the equality holds.

Therefore, $f(x)$ is convex but not strictly convex

**1.4** Suppose $x_1, x_2, \ldots, x_N$ are drawn from $Laplace(\mu, b)$. Calculate the MLE (maximum likelihood estimation) of $\mu$ and $b$. Hint: Use the logarithmic trick to process multiple exponential items. (12 points)

$P(x) = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)$

$L(x) = \prod_{i=1}^{N} P(x) = \left(\frac{1}{2b}\right)^n \exp\left(-\frac{\sum_{i=1}^{N}|x_i-\mu|}{b}\right)$

$l(x) = -n\log(2b) - \frac{1}{b}\sum_{i=1}^{N}|x_i-\mu|$

$\frac{\partial l(x)}{\partial b} = -\frac{n}{b} + \frac{1}{b^2}\sum_{i=1}^{N}|x_i-\mu| = 0 \Rightarrow \hat{b} = \frac{1}{N}\sum_{i=1}^{N}|x_i-\mu|$

$\frac{\partial l(x)}{\partial \mu} = -\frac{1}{b}\sum_{i=1}^{N}\frac{\partial|x_i-\mu|}{\partial \mu}$ $\quad \frac{\partial|x|}{\partial x} = \frac{\partial\sqrt{x^2}}{\partial x} = x(x^2)^{-\frac{1}{2}} = \frac{x}{|x|} = sgn(x)$.

$= -\frac{1}{b}\sum_{i=1}^{N} sgn(x_i-\mu)$

$-\frac{1}{b}\sum_{i=1}^{N} sgn(x_i-\mu) = 0$

① N is odd. We choose $\hat{\mu} = median(x_1, \ldots, x_N)$

② N is even. Ranking the observations as $x_1 \le x_2 \le \ldots, x_N$ and then choose $\hat{\mu} = x_{N/2}$ or $\hat{\mu} = x_{(N+1)/2}$

In summary, $\hat{\mu} = median(x_1, \ldots, x_N)$ is the MLE of $\mu$, and $\hat{b} = \frac{1}{N}\sum_{i=1}^{N}|x_i-\mu|$ is the MLE of $b$.

DDA3020 Assignment 1 report (coding part)

2.Programming Question

2.1 Load data and summarize characteristics

Use pd.read_csv() function to read the data file boston.csv. Notice that the loading of the data is using relative path instead of absolute path. Hence, when executing the code of the model, one should place the train data txt file and test data test file under the same directory of the model python file.

Below is the general content of the boston data. (generated by df.head())

| | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | b | lstat | medv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 396.90 | 4.98 | 24.0 |
| **1** | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.90 | 9.14 | 21.6 |
| **2** | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 | 4.03 | 34.7 |
| **3** | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 | 2.94 | 33.4 |
| **4** | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.90 | 5.33 | 36.2 |

We guess that 'medv' and 'rm' have a strong relationship, since as 'rm' becomes larger, 'medv' also becomes larger.

I use 3 methods to test whether there exists 'NaN' or 'Null' in this dataset.:

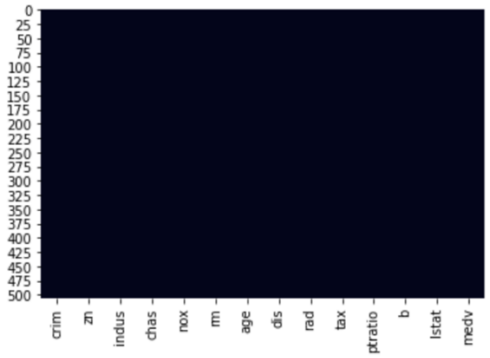
1. df.isnull()

| | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | b | lstat | medv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **4** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **501** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **502** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **503** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **504** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **505** | False | False | False | False | False | False | False | False | False | False | False | False | False | False |

506 rows × 14 columns

NaN (Null) corresponds to True, and otherwise.

2. sns.heatmap(df.isnull(), cbar=False)




The picture is all black, no white space in it→no NaN (Null) in this dataset.

3. df.isna().sum()

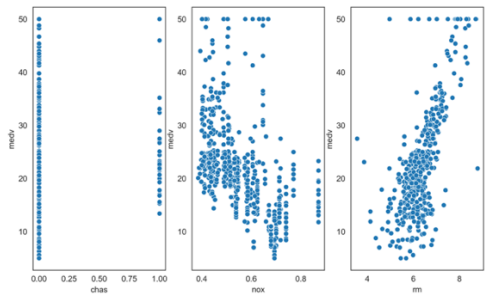
```
crim        0
zn          0
indus       0
chas        0
nox         0
rm          0
age         0
dis         0
rad         0
tax         0
ptratio     0
b           0
lstat       0
medv        0
dtype: int64
```

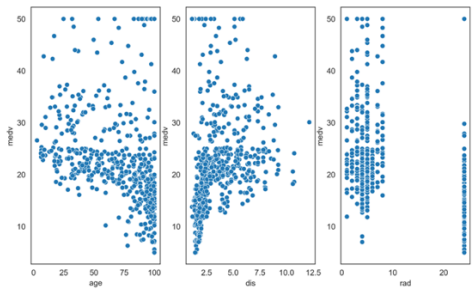
As the function name said, count the sum of the number of NaN or Null in every column of this dataset, respectively.

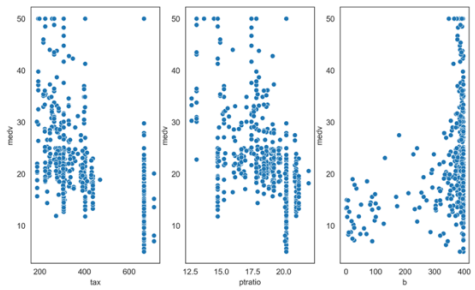
Hence, we conclude that there is no Nan or Null in this boston dataset.

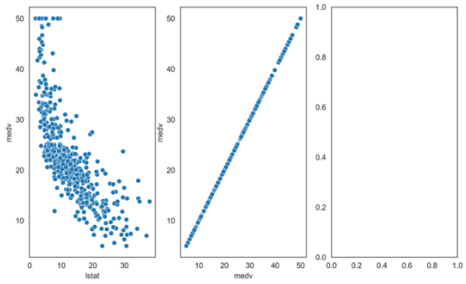
2.2 visualize the dataset

Use sns.scatterplot() to plot the MEDV distributions over each attribute. And below are the figures we get.






From the figures above, we insist the assumption for the most relevant attribute for MEDV is RM.

2.3 Pairwise correlation on data

Use sns.heatmap(df.corr(),annot=True) to draw the covariance figures.


Assume if covariance >0.4 or <-0.4, then the attribute is good predictor

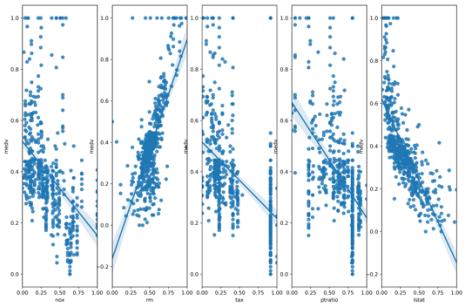
The covariances between medv and nox,rm,tax,ptratio,lstat are -0.43,0.7,-0.47,-0.51,and -0.74, respectively.

Good predictors: nox,rm,tax,ptratio,and lstat

2.4 scale the columns and draw the relevance

Use MinMaxScaler() to scale the attributes.

Then use seaborn.regplot() to plot the relevance of these columns against MEDV with 95% confidence interval.




2.5 train linear regression model and test on it

(1). linear model description

$$f_{w,b} = x^T w + b$$

The set of feature vector $x$ is the columns of "nox,rm,tax,ptratio,and lstat" in the boston

dataset.
The target output $y$ is the column of "medv" in the boston dataset.

We solve this linear regression by gradient descent.
We split the data into two parts: training (80%) and test (20%).

(2). loss function

$$loss\ function: J(\bar{w}) = \frac{1}{2}\sum_{i=1}^{m}(x_i^T w + b - y_i)^2 = \frac{1}{2}(X\bar{w} - y)^2$$

$$\bar{w}^* = arg_{\bar{w}} min J(\bar{w})$$

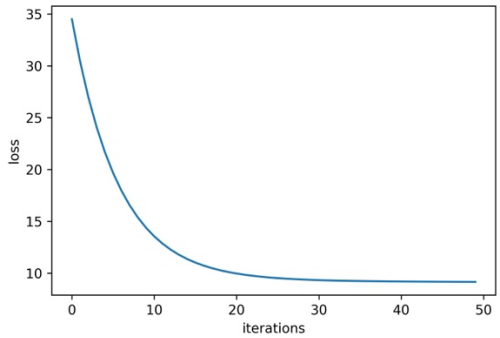
$w\ can\ be\ updated\ by\ gradient\ descent\ algorithm,$

$$w \leftarrow w - \alpha\frac{\partial J(w)}{\partial w}, \frac{\partial J(w)}{\partial w} = X^T(Xw - y)$$

Where $X = [(1, x_1^T); (1, x_2^T); \ldots; (1, x_m^T)] \in \mathcal{R}^{m\times(d+1)}$, and $\bar{w} = [b; w] \in \mathcal{R}^{(d+1)\times 1}$
We transform $x_i^T w + b$ into $X\bar{w}$ by "get_full_sample_matrix" this self-define function.
(3). hyperparameter settings
After transforming into $X\bar{w}$, we only need to train one hyperparameter $\bar{w}$ .
(4). RMSE equation
Mathematic equation of RMSE is:

$$RMSE = \sqrt{\sum_{i=1}^{m}\frac{(\bar{y_i} - y_i)^2}{m}}$$

In python, we use RMSE=np.sqrt(metrics.mean_squared_error($y$,$\bar{y}$)) to compute.
(5). outputs (errors, plots, figures)
Assume step size=0.0001, iteration number=50;
we get that:
W= [[0.19354444],[0.05286568],[0.10555365],[0.05506099],[0.0970676 ],[0.04649039]]
RMSE of train= 0.20864370794310755
RMSE of test= 0.22787361921161076
the loss curve in the training process is as below:




2.6 repeat
We choose step size from list [0.01,0.001,0.0001,0.00001], iteration number from list

[50,100,500].
Repeat 2.5 and get solutions as below:
1. step size= 0.01 iteration steps= 50
RMSE of train= 1.176899655389494e+43
RMSE of test= 1.2011007219694868e+43
2. step size= 0.001 iteration steps= 50
RMSE of train= 0.2058516518776543
RMSE of test= 0.19632119342529225
3. step size= 0.001 iteration steps= 100
RMSE of train= 0.20649137801162187
RMSE of test= 0.1929998937307037
4. step size= 0.001 iteration steps= 500
RMSE of train= 0.19124148666161425
RMSE of test= 0.24220397879116876
5. step size= 0.0001 iteration steps= 50
RMSE of train= 0.2140196628121716
RMSE of test= 0.18928550674499142
6. step size= 0.0001 iteration steps= 100
RMSE of train= 0.2098229912994151
RMSE of test= 0.20121029612621325
7. step size= 0.0001 iteration steps= 500
RMSE of train= 0.20552547567407597
RMSE of test= 0.20527230239717645
8. step size= 1e-05 iteration steps= 50
RMSE of train= 0.3284879924913493
RMSE of test= 0.3325026354759041
9. step size= 1e-05 iteration steps= 100
RMSE of train= 0.2708984994062212
RMSE of test= 0.2611387972606977
10. step size= 1e-05 iteration steps= 500
RMSE of train= 0.2114803182586758
RMSE of test= 0.20545886408108904
Conclusion, as step size increases, it takes less iterations to get close to the optimal solution. But if step size is too big, the result may diverge, and loss may be too large.