Applied Machine Learning

Linear Regression

Salahadin Seid

School of Data Science
Emerland University

August 10, 2024

ኤመራልድ ኢንተርናሽናል ኮሌጅ
EMERALD INTERNATIONAL COLLEGE

Outline

- **What is Linear Regression?**
- **How does it work?**
- **Estimating the coefficients**
- **Assessing the accuracy of the coefficient estimates**
- **Assessing the accuracy of the model**
-

What is Linear Regression?

- **Linear regression** is a simple supervised statistical learning technique used for predicting quantitative a response target.
- It is one of the **simplest** and **oldest** statistical learning techniques
- It is still very useful and widely used to this day for many reasons.
  - A good starting point for the newer and more sophisticated machine-learning techniques, many of which may be seen as generalizations of linear regresssion,
  - Having a good understanding of linear regression proves invaluable in studying these **newer** and more **sophisticated** methods.

What is Regression?

X: Independent variable

Y: Dependent variable

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

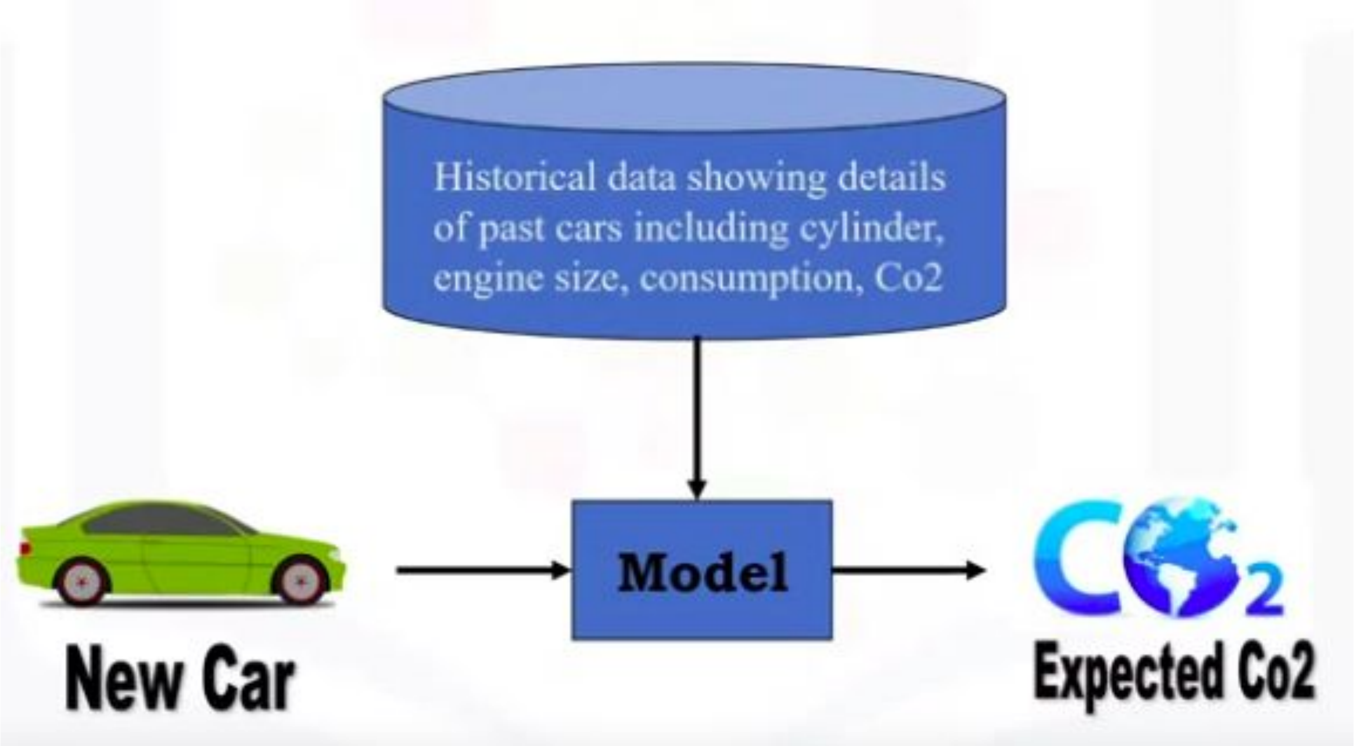
- **Regression is the process of predicting a continuous value**

What is Regression?

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

- **Is that possible to predict the co2 emission of vehicle 9 before production given engine size and cylinder size?**

What is Regression?



Application of Regression

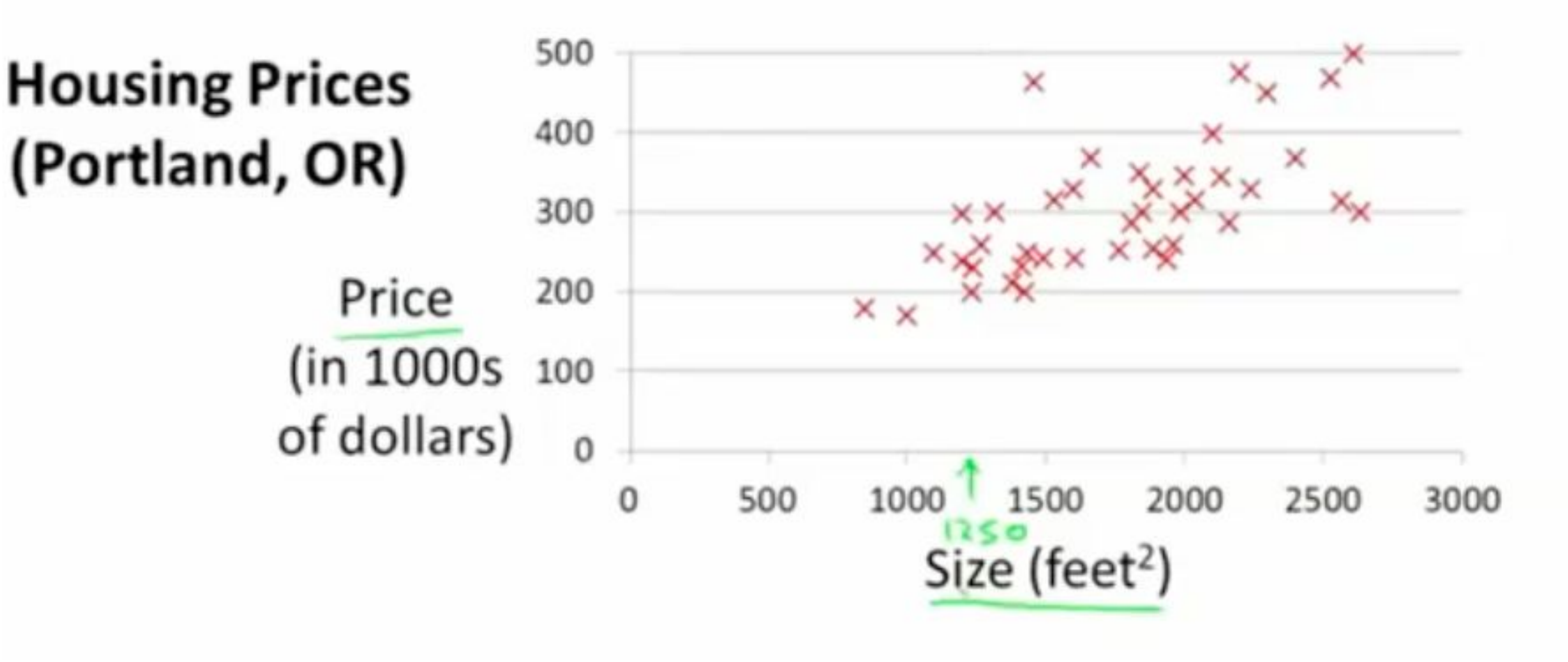
- **Sales forecasting**
- **Satisfaction analysis**
- **Price estimation**
- **Employment Income**

Linear Regression with one variable

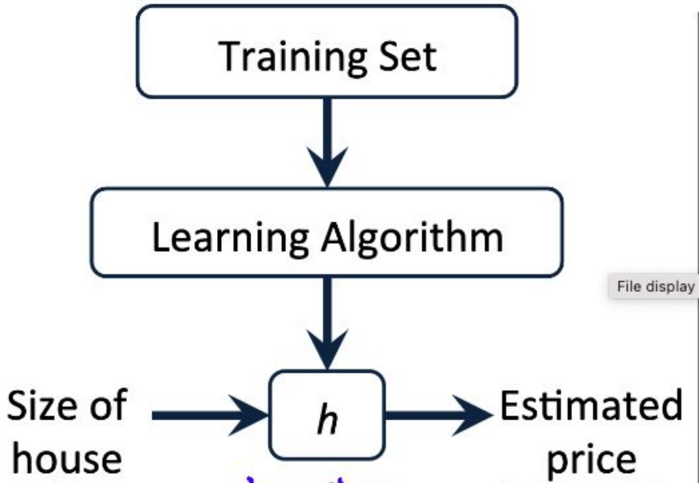
Model Representation

Linear regression with one variable

We want to tell the price of the house given the size of the house?



Linear regression with one variable



x

hypothesis

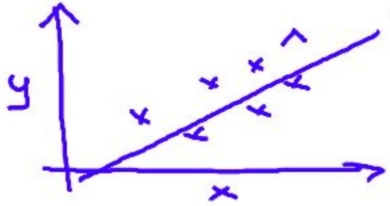
(estimated value of y)

h maps from x's to y's.

How do we represent *h* ?

$\underline{h_\theta(x) = \theta_0 + \theta_1 x}$

Shorthand: $h(x)$



$h(x) = \theta_0 + \theta_1 x$

Linear regression with one variable. (x)

Univariate linear regression.

└ one variable

Linear regression with one variable

Training Set

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

$m = 47$

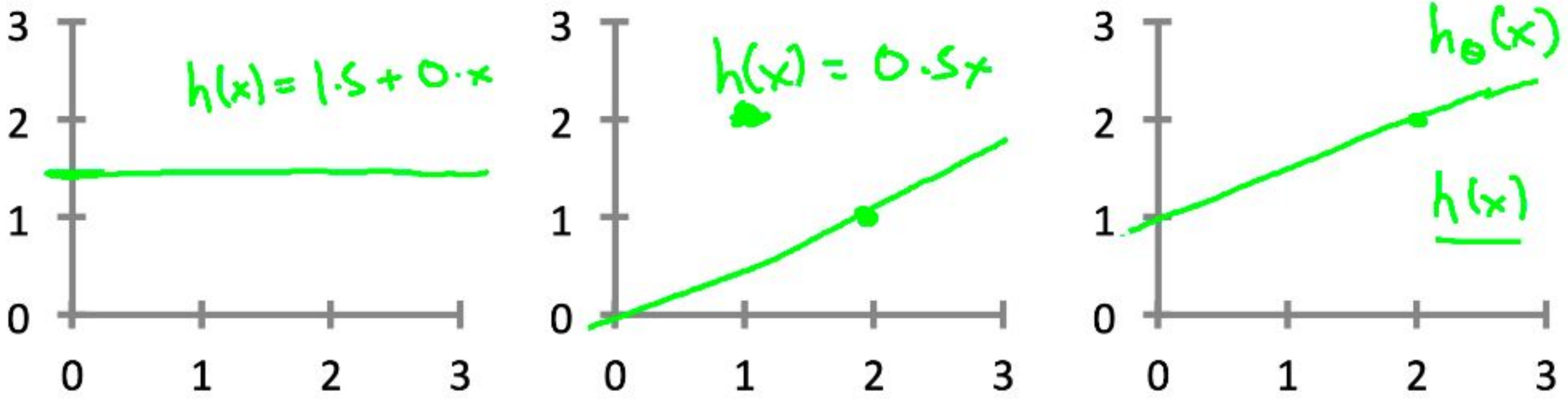
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i's: Parameters

How to choose θ_i's ?

Linear regression with one variable

$$h_\theta(x) = \theta_0 + \theta_1 x$$

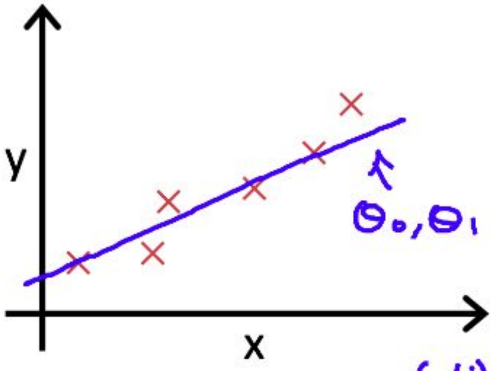


$\theta_0 = 1.5$
$\theta_1 = 0$

$\theta_0 = 0$
$\theta_1 = 0.5$

$\theta_0 = 1$
$\theta_1 = 0.5$

Linear regression with one variable



θ_0, θ_1

$(x^{(i)}, y^{(i)})$

training examples

$$\underset{\theta_0 \theta_1}{\boxed{\text{minimize}}} \ \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

dea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

x, y

$$\underset{\theta_0, \theta_1}{\text{minimize}} \ \underbrace{J(\theta_0, \theta_1)}_{\text{Cost function}}$$

Squared error function

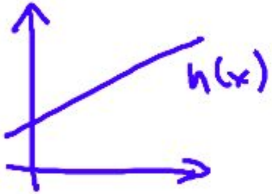
Linear regression with one variable

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$



Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

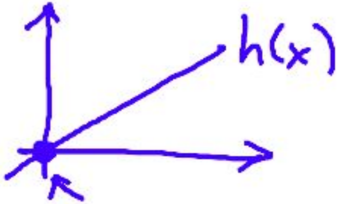
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Simplified

$$h_\theta(x) = \theta_1 x$$

$\theta_0 = 0$

$$\theta_1$$



$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

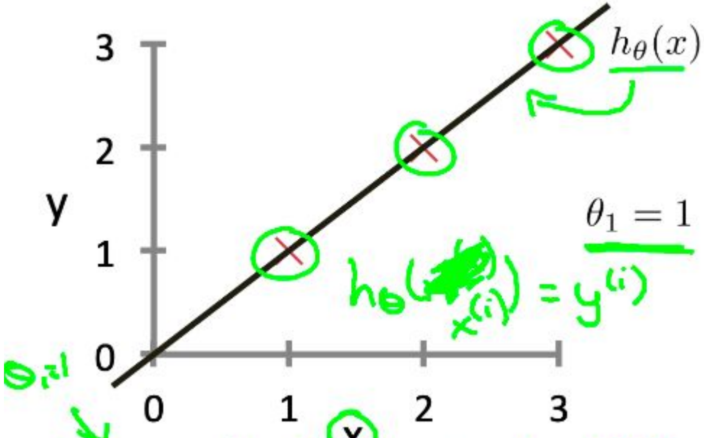
$\theta, x^{(i)}$

$$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$$

Linear regression with one variable

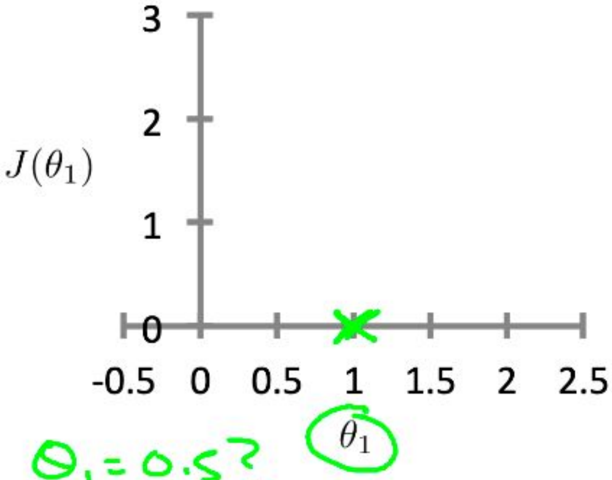
$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$J(\theta_1)$

(function of the parameter θ_1)



$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_1 x^{(i)} - y^{(i)}\right)^2 = \frac{1}{2m}\left(0^2 + 0^2 + 0^2\right) = 0^2$$

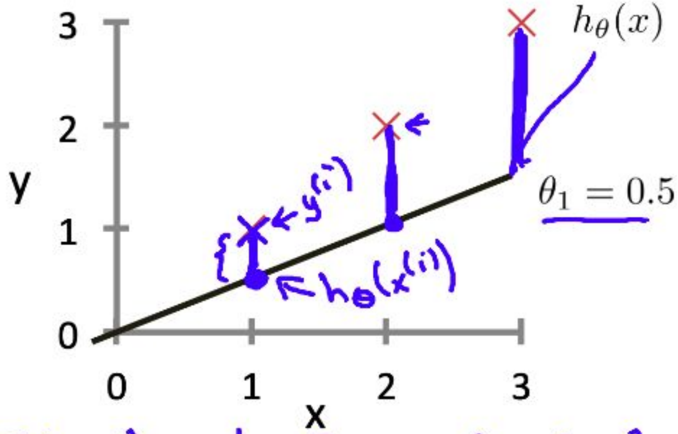
$\theta_1 = 0.5?$

$J(1) = 0$

Linear regression with one variable

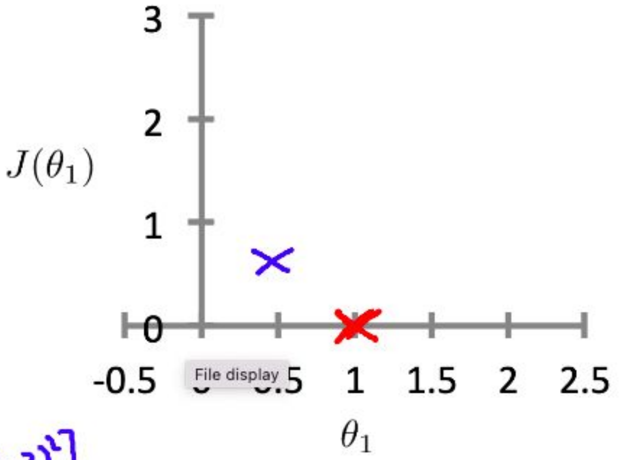
$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$J(\theta_1)$

(function of the parameter θ_1)



$$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$

$$= \frac{1}{2 \times 3}(3.5) = \frac{3.5}{6} \approx 0.58$$

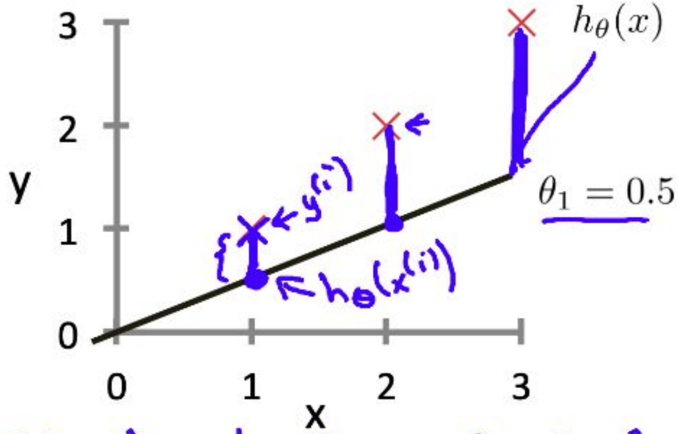
$\theta_1 = 0$?

$J(0) = $?

Linear regression with one variable

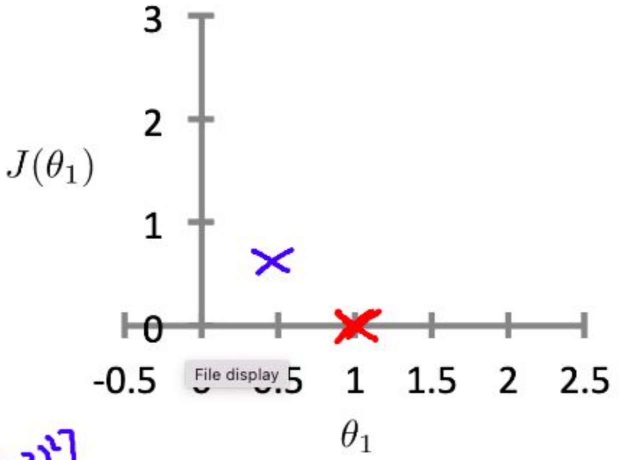
$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$J(\theta_1)$

(function of the parameter θ_1)



$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$

$= \frac{1}{2\times 3}(3.5) = \frac{3.5}{6} \approx 0.58$

$\theta_1 = 0$?

$J(0) = $?

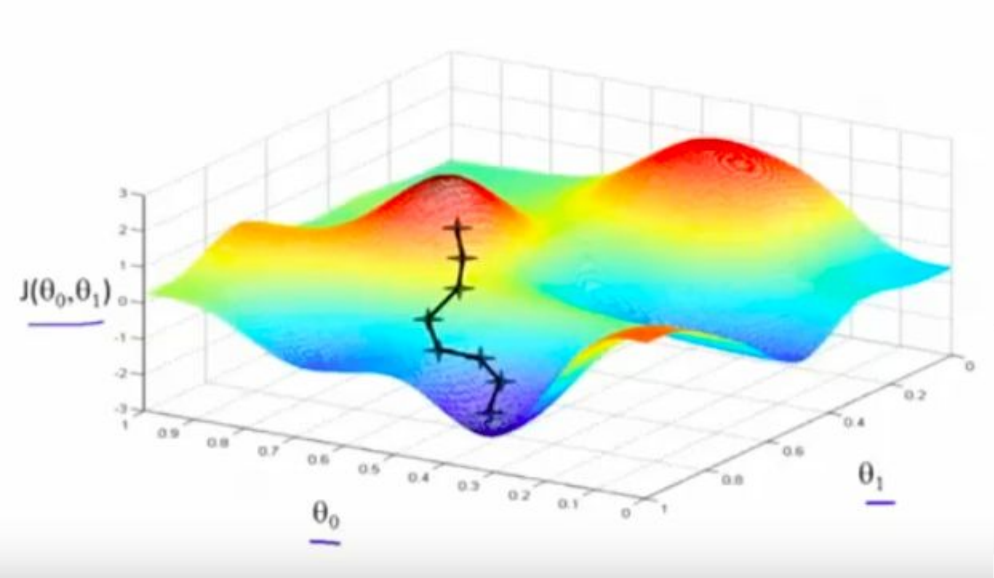
Linear Regression with one variable

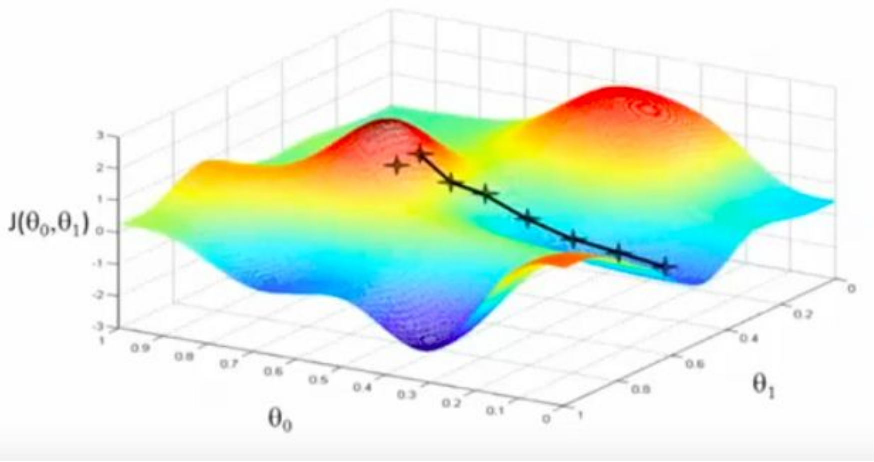
Gradient descent

Gradient Descent in Linear Regression?

- Any ML project our main aim relies on how good our project accuracy is or how much our model prediction differs from the actual data point.
- Based on the difference between model prediction and actual data points we try to find the parameters of the model which give better accuracy on our dataset
- In order to find these parameters we apply gradient descent on the cost function of the machine learning model.

Gradient Descent in Linear Regression?





Gradient Descent in Linear Regression?

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

}

Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Gradient descent

Have some function $J(\theta_0, \theta_1)$

Want $\min\limits_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

Gradient descent

Cost Function

$$J\left(\Theta_0, \Theta_1\right) = \frac{1}{2m}\sum_{i=1}^{m}[\underbrace{h_\Theta(x_i)}_{\text{Predicted Value}} - \underbrace{y_i}_{\text{True Value}}]^2$$

Gradient Descent

$$\boxed{\Theta_j = \Theta_j - \underbrace{\alpha}_{\text{Learning Rate}}\frac{\partial}{\partial \Theta_j} J\left(\Theta_0, \Theta_1\right)}$$

Now,

$$\frac{\partial}{\partial \Theta} J_\Theta = \frac{\partial}{\partial \Theta}\frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y]^2$$

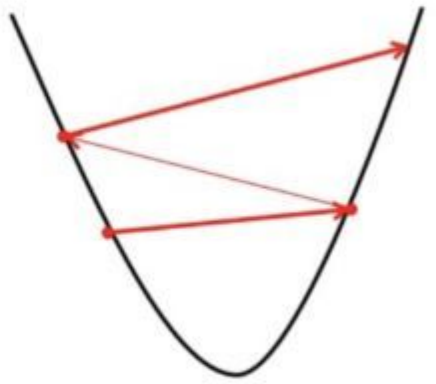
$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x_i) - y)\frac{\partial}{\partial \Theta_j}(\Theta x_i - y)$$

$$= \frac{1}{m}(h_\Theta(x_i) - y)x_i$$

Therefore,

$$\boxed{\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i) - y)x_i]}$$

Gradient descent - learning rate



If we choose **α to be very large**, Gradient Descent can overshoot the minimum. It may fail to converge or even diverge.

If we choose **α** to be **very small**, Gradient Descent will take small steps to reach local minima and will take a longer time to reach minima.

Linear Regression with multiple variables

Multiple features

Single Variable vs multiple variables

Size (feet²) x	Price ($1000) y
2104	460
1416	232
1534	315
852	178
...	...

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Modeling

Hypothesis:

Previously: $\cancel{h_\theta(x) = \theta_0 + \theta_1 x}$

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

E.g. $\underline{h_\theta(x)} = \underline{80} + \underline{0.1} x_1 + \underline{0.01} x_2 + 3 x_3 - 2 x_4$

age

Modeling

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

For convenience of notation, define $\boxed{x_0 = 1.}$ $(x_0^{(i)} = 1)$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$h_\theta(x) = \theta_0 \overset{=1}{x_0} + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$= \boxed{\theta^T x.}$$

$$\underbrace{[\theta_0 \; \theta_1 \cdots \theta_n]}_{\theta^T} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$(n+1) \times 1$ matrix

$\theta^T x$ $\quad x$

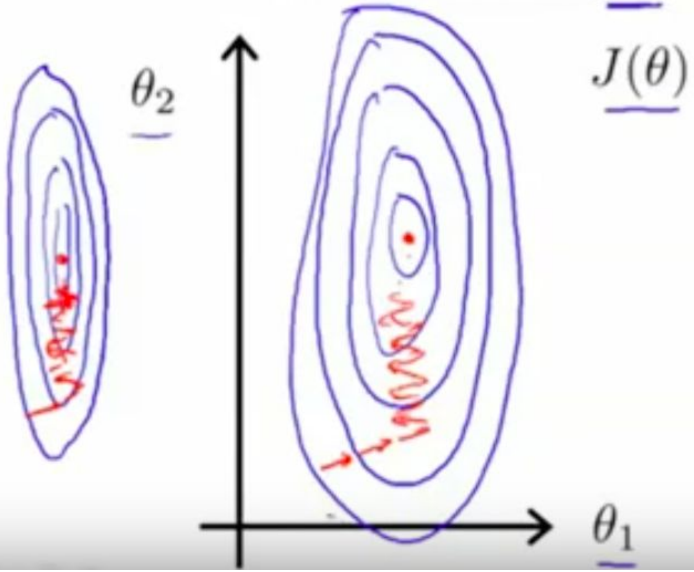
Multivariate linear regression.

Feature Scaling

Idea: Make sure features are on a similar scale.

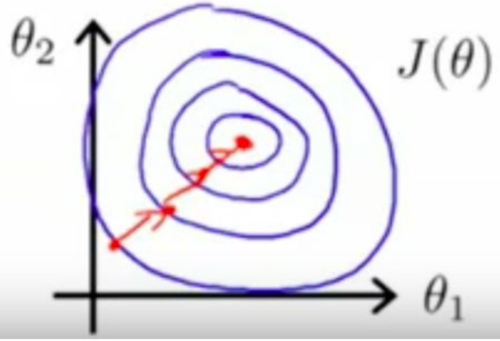
E.g. x_1 = size (0-2000 feet²)

x_2 = number of bedrooms (1-5)



$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$



Feature Scaling

Get every feature into approximately a $-1 \le x_i \le 1$ range.

$x_0 = 1$

$0 \le x_1 \le 3$ ✓

$-2 \le x_2 \le 0.5$ ✓

$-100 \le x_3 \; 100$ ✗

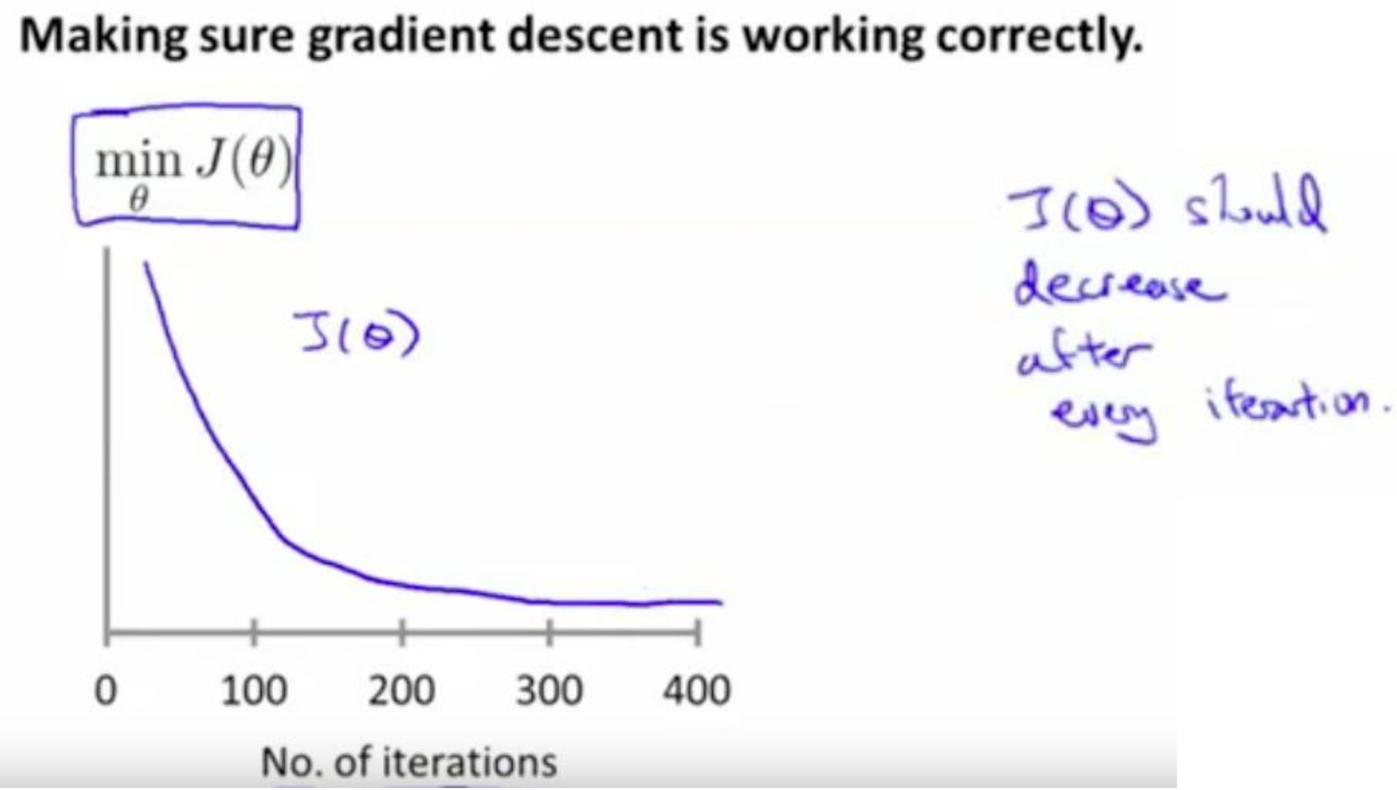
Learning rate

Gradient descent

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

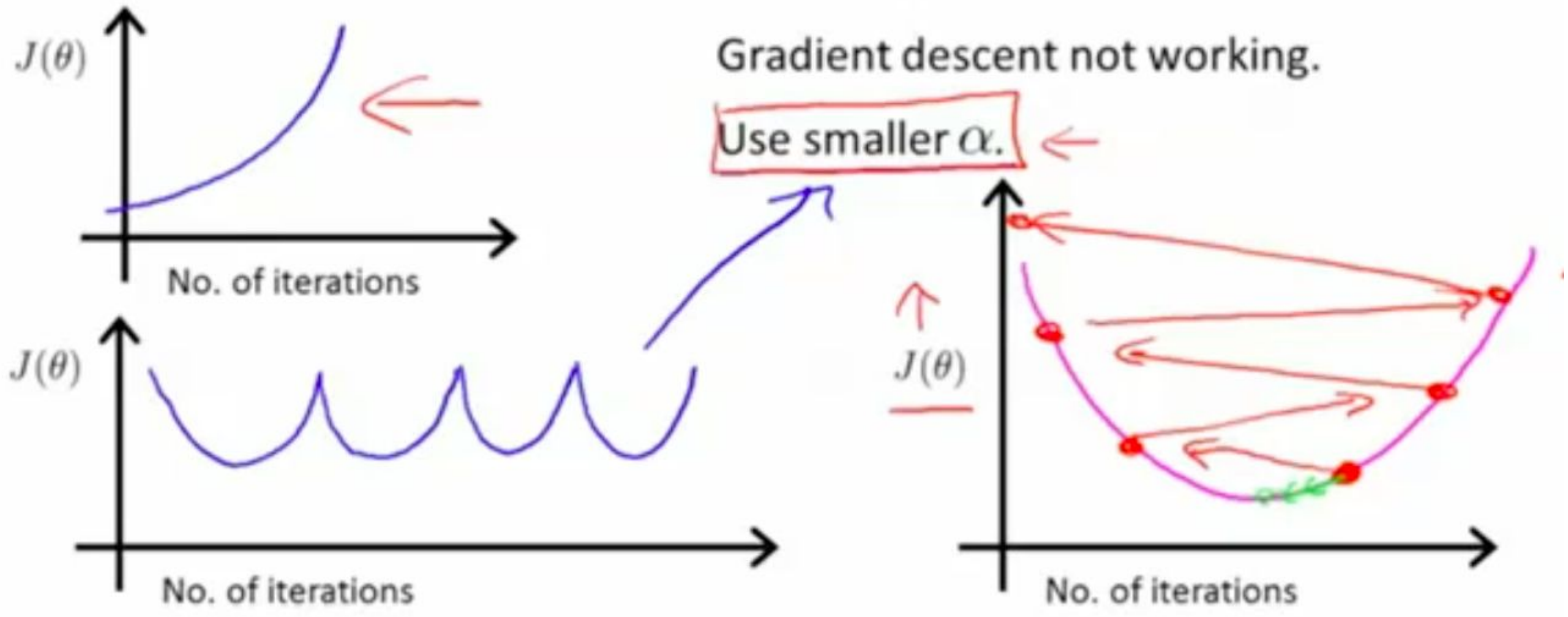
- "Debugging": How to make sure gradient descent is working correctly.
- How to choose learning rate α.

Learning rate



Learning rate

Making sure gradient descent is working correctly.



- For sufficiently small α, $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Linear Regression with Python

Given Boston house prices dataset: Mock Scenario

- Imagine the following scenario: Your team has been contracted by a real-estate company who owns many houses in the Boston metro-area. You are tasked to analyze if the house prices owned by the company are competitive and if they need to be adjusted to optimize the overall profit.

Below are some questions that your team might want to address before making recommendations:

- Is there a relationship between the different **predictors/features** and the house **prices**
- If there is such a relationship, then how strong is it?
- Which features affect the most the house prices?
- How accurately can we estimate the effect of each feature to the house prices?
- How accurately can we predict the house prices based on these features?
- What kind of relationship is there between the features and the house prices? Linear?Non-linear?
- Is there any **interaction** effect among the features?

We will attempt to address all of these questions using linear regression.

Simple linear Regression

Let's go ahead and import some libraries:

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Description of Boston house prices dataset

```
boston=pd.read_excel('Boston_Dataset.xlsx',index_col=[0])
```

```
boston.head()
```

Unnamed: 0	CRIM	CHAS	RM	DIS	RAD	TAX	PTRATIO	LSTAT	Price
0	0.00632	No	6.575	4.0900	1	296	15.3	4.98	24.0
1	0.02731	No	6.421	4.9671	2	242	17.8	9.14	21.6
2	0.02729	No	7.185	4.9671	2	242	17.8	4.03	34.7
3	0.03237	No	6.998	6.0622	3	222	18.7	2.94	33.4
4	0.06905	No	7.147	6.0622	3	222	18.7	5.33	36.2

```
boston['CHAS'].unique()
```

```
array(['No', 'Yes'], dtype=object)
```

Description of Boston house prices dataset

```
#Enter Answer
df=boston.copy()
df['CHAS']=boston['CHAS'].apply(lambda x: 1 if x=='Yes' else 0)
```

```
df.head()
```

	CRIM	CHAS	RM	DIS	RAD	TAX	PTRATIO	LSTAT	Price
Unnamed: 0									
0	0.00632	0	6.575	4.0900	1	296	15.3	4.98	24.0
1	0.02731	0	6.421	4.9671	2	242	17.8	9.14	21.6
2	0.02729	0	7.185	4.9671	2	242	17.8	4.03	34.7
3	0.03237	0	6.998	6.0622	3	222	18.7	2.94	33.4
4	0.06905	0	7.147	6.0622	3	222	18.7	5.33	36.2

Scaling data

- Many of the measurements are in different scale - It is important to scale the data before we fit a linear regression.

```
from sklearn.preprocessing import StandardScaler
```

```
X=df.drop("Price",axis=1)
y=df['Price']
```

```
X.head()
```

	CRIM	CHAS	RM	DIS	RAD	TAX	PTRATIO	LSTAT
Unnamed: 0								
0	0.00632	0	6.575	4.0900	1	296	15.3	4.98
1	0.02731	0	6.421	4.9671	2	242	17.8	9.14
2	0.02729	0	7.185	4.9671	2	242	17.8	4.03
3	0.03237	0	6.998	6.0622	3	222	18.7	2.94
4	0.06905	0	7.147	6.0622	3	222	18.7	5.33

Scaling data

```
scaler=StandardScaler()
```

```
scaled=scaler.fit_transform(X)
```

After scaling the data, create a new dataframe of the scaled data called `X_sc` and check the head

```
X_sc=pd.DataFrame(scaled,columns=X.columns)
```

```
X_sc.head()
```

	CRIM	CHAS	RM	DIS	RAD	TAX	PTRATIO	LSTAT
0	-0.419782	-0.272599	0.413672	0.140214	-0.982843	-0.666608	-1.459000	-1.075562
1	-0.417339	-0.272599	0.194274	0.557160	-0.867883	-0.987329	-0.303094	-0.492439
2	-0.417342	-0.272599	1.282714	0.557160	-0.867883	-0.987329	-0.303094	-1.208727
3	-0.416750	-0.272599	1.016303	1.077737	-0.752922	-1.106115	0.113032	-1.361517
4	-0.412482	-0.272599	1.228577	1.077737	-0.752922	-1.106115	0.113032	-1.026501

Train-Test Split

Split the scaled data into a training set and a test set with a 70-30 split.

Step 1: Import `train_test_split` from `sklearn.model_selection`

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(X_sc,y,test_size=0.3,random_state=12)
```

```
X_train.head()
```

	CRIM	CHAS	RM	DIS	RAD	TAX	PTRATIO	LSTAT
251	-0.395603	-0.272599	0.218494	1.712117	-0.293081	-0.464673	0.297977	-1.270404
465	-0.052359	-0.272599	-0.748850	-0.346327	1.661245	1.530926	0.806576	0.207028
137	-0.379516	-0.272599	0.241288	-0.924708	-0.637962	0.170831	1.268938	0.271508
311	-0.328535	-0.272599	-0.231698	-0.548929	-0.637962	-0.619094	-0.025677	-0.935389
406	1.990294	-0.272599	-3.058221	-1.244014	1.661245	1.530926	0.806576	1.498028

Fitting the Model

First, we need to import Linear Regression model

```
from sklearn.linear_model import LinearRegression
```

Next, we instantiate and fit it to our model

```
lg=LinearRegression()
```

```
lg.fit(X_train,y_train)
```

Now that we have fitted our model, we can actually go ahead and extract the estimated coefficients, including the intercept β_0

```
lg.intercept_
```

```
22.605596398342957
```

```
lg.coef_
```

```
array([-0.63082368,  0.57150473,  2.24918842, -1.09485652,  1.85711767,
-1.88803821, -2.046749  , -4.55974378])
```

Fitting the Model ...

Create a dataframe named `df_coef` that contains one column with the coefficients and the rows are the features of the Boston dataset.

```
#Enter Answer Here
df_coef=pd.DataFrame(lg.coef_,index=X.columns,columns=['Coefficients'])
```

```
df_coef.loc['Intercept','Coefficients']=lg.intercept_
```

```
df_coef
```

How do we interpret these coefficients? What do they mean and what do they tell us?

For example,if the average number of rooms per dwelling, `RM` , increases by one unit, we will observe an increase in house price by roughly $2.2 \times \$10,000 = \$22,000$

On the other hand, if the pupil-teacher ratio by town, `PTRATIO` , increases by one unit, then we will observe a decrease in house price of roughly $2.05 \times \$10,000 = \$20500.$

Assesing Performance of Model

Now that our model is fit, we need to assess the performance of our model by testing it on the test-set.

```
lg_pred=lg.predict(X_test)
```

`g_pred` contains the predicted house prices.

Test the performance

```
from sklearn.metrics import r2_score, mean_squared_error
```

```
lg_r2=r2_score(y_test,lg_pred)
mse=mean_squared_error(y_test,lg_pred)
```

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

```
lg_r2
```

```
mse
```