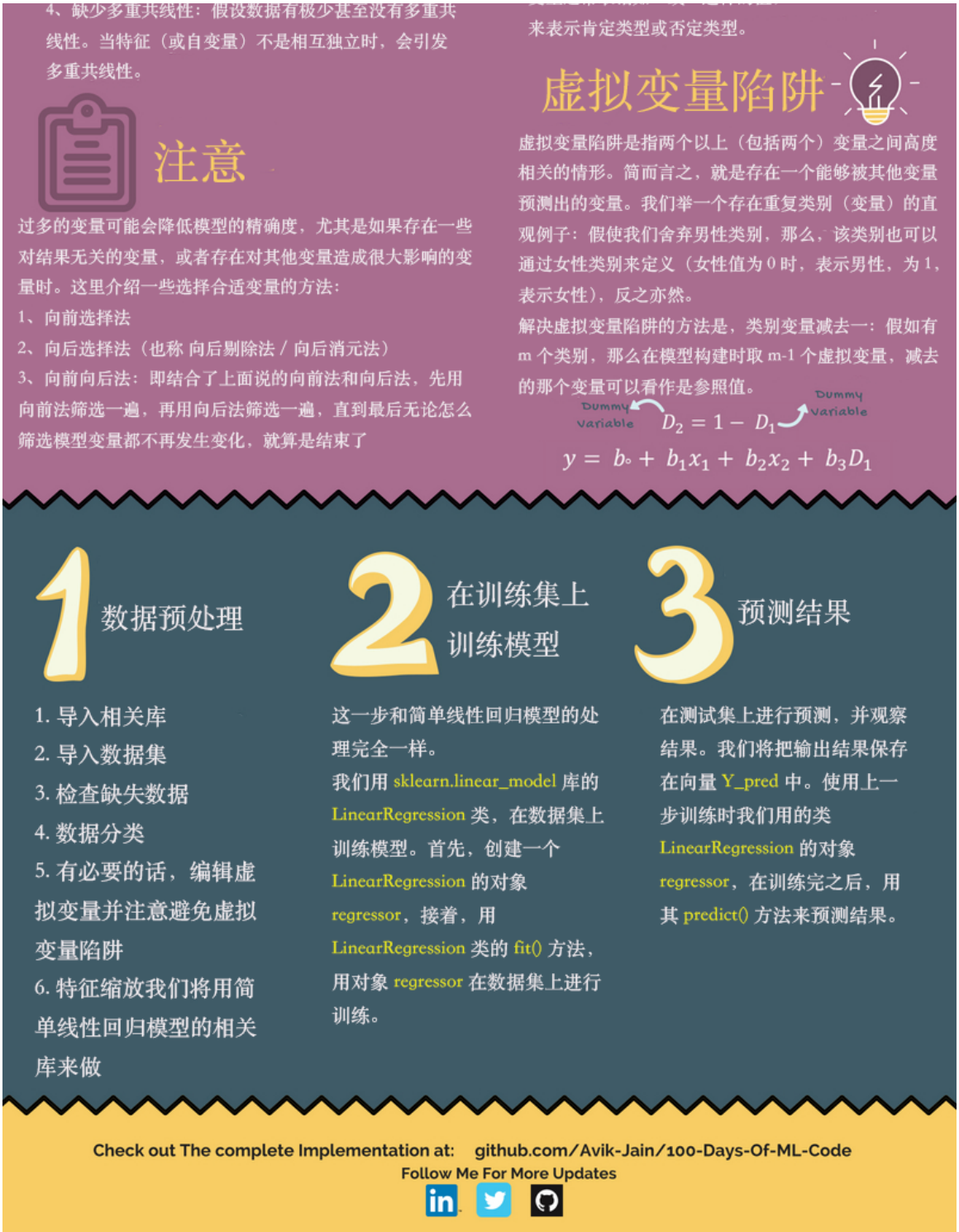
day3多元线性回归

笔记本:	100days		
创建时间:	2019/1/10 22:10	**更新时间:**	2019/1/11 17:37
URL:	http://localhost:8888/notebooks/Day3/day3%E5%A4%9A%E5%85%83%E7%BA%BF%E6%80%A7%E5%9B...		

Multiple Linear Regression(多元线性回归)¶



Step 1: Data Preprocessing(数据预处理)¶

Importing the libraries(导入库)¶

In [1]:

x

```
import pandas as pd
```

```
import numpy as np
```

. . .

Importing the dataset(导入数据集)¶

In [7]:

1

```
dataset = pd.read_csv('50_Startups.csv')
```

```
dataset
```

Out[7]:

R&D SPEND	ADMINISTRATION	MARKETING SPEND	STATE	PROFIT	
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96

10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84
33	55493.95	103057.49	214634.81	Florida	96778.92
34	46426.07	157693.92	210797.67	California	96712.80

35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83
42	23640.93	96189.63	148001.11	California	71498.49
43	15505.73	127382.30	35534.17	New York	69758.98
44	22177.74	154806.14	28334.72	California	65200.33
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0.00	116983.80	45173.06	California	14681.40

. . .

In [10]:

1

```
X = dataset.iloc[:,:-1].values
```

```
Y = dataset.iloc[:,4].values
```

```
X,Y
```

Out[10]:

(array([[165349.2, 136897.8, 471784.1, 'New York'],
[162597.7, 151377.59, 443898.53, 'California'],
[153441.51, 101145.55, 407934.54, 'Florida'],
[144372.41, 118671.85, 383199.62, 'New York'],
[142107.34, 91391.77, 366168.42, 'Florida'],
[131876.9, 99814.71, 362861.36, 'New York'],
[134615.46, 147198.87, 127716.82, 'California'],
[130298.13, 145530.06, 323876.68, 'Florida'],
[120542.52, 148718.95, 311613.29, 'New York'],
[123334.88, 108679.17, 304981.62, 'California'],
[101913.08, 110594.11, 229160.95, 'Florida'],
[100671.96, 91790.61, 249744.55, 'California'],
[93863.75, 127320.38, 249839.44, 'Florida'],
[91992.39, 135495.07, 252664.93, 'California'],
[119943.24, 156547.42, 256512.92, 'Florida'],
[114523.61, 122616.84, 261776.23, 'New York'],
[78013.11, 121597.55, 264346.06, 'California'],
[94657.16, 145077.58, 282574.31, 'New York'],
[91749.16, 114175.79, 294919.57, 'Florida'],
[86419.7, 153514.11, 0.0, 'New York'],
[76253.86, 113867.3, 298664.47, 'California'],
[78389.47, 153773.43, 299737.29, 'New York'],
[73994.56, 122782.75, 303319.26, 'Florida'],
[67532.53, 105751.03, 304768.73, 'Florida'],
[77044.01, 99281.34, 140574.81, 'New York'],
[64664.71, 139553.16, 137962.62, 'California'],
[75328.87, 144135.98, 134050.07, 'Florida'],
[72107.6, 127864.55, 353183.81, 'New York'],
[66051.52, 182645.56, 118148.2, 'Florida'],
[65605.48, 153032.06, 107138.38, 'New York'],
[61994.48, 115641.28, 91131.24, 'Florida'],
[61136.38, 152701.92, 88218.23, 'New York'],
[63408.86, 129219.61, 46085.25, 'California'],
[55493.95, 103057.49, 214634.81, 'Florida'],
[46426.07, 157693.92, 210797.67, 'California'],
[46014.02, 85047.44, 205517.64, 'New York'],
[28663.76, 127056.21, 201126.82, 'Florida'],
[44069.95, 51283.14, 197029.42, 'California'],
[20229.59, 65947.93, 185265.1, 'New York'],
[38558.51, 82982.09, 174999.3, 'California'],
[28754.33, 118546.05, 172795.67, 'California'],
[27892.92, 84710.77, 164470.71, 'Florida'],
[23640.93, 96189.63, 148001.11, 'California'],
[15505.73, 127382.3, 35534.17, 'New York'],
[22177.74, 154806.14, 28334.72, 'California'],
[1000.23, 124153.04, 1903.93, 'New York'],
[1315.46, 115816.21, 297114.46, 'Florida'],

```
[0.0, 135426.92, 0.0, 'California'],
[542.05, 51743.15, 0.0, 'New York'],
[0.0, 116983.8, 45173.06, 'California']], dtype=object),
array([192261.83, 191792.06, 191050.39, 182901.99, 166187.94, 156991.12,
156122.51, 155752.6 , 152211.77, 149759.96, 146121.95, 144259.4 ,
141585.52, 134307.35, 132602.65, 129917.04, 126992.93, 125370.37,
124266.9 , 122776.86, 118474.03, 111313.02, 110352.25, 108733.99,
108552.04, 107404.34, 105733.54, 105008.31, 103282.38, 101004.64,
99937.59, 97483.56, 97427.84, 96778.92, 96712.8 , 96479.51,
90708.19, 89949.14, 81229.06, 81005.76, 78239.91, 77798.83,
71498.49, 69758.98, 65200.33, 64926.08, 49490.75, 42559.73,
35673.41, 14681.4 ]))
```

. . .

Encoding Categorical data(将类别数据数字化)¶

In [18]:

6

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

```
labelencoder = LabelEncoder()
```

```
X[:,3]=labelencoder.fit_transform(X[:,3])
```

```
onehotencoder=OneHotEncoder(categorical_features=[3])
```

```
X=onehotencoder.fit_transform(X).toarray()
```

```
X
```

```
C:\Users\张帅\AppData\Roaming\Python\Python36\site-
packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer data
will change in version 0.22. Currently, the categories are determined based on the range [0,
max(values)], while in the future they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers,
then you can now use the OneHotEncoder directly.
warnings.warn(msg, FutureWarning)
```

```
C:\Users\张帅\AppData\Roaming\Python\Python36\site-
packages\sklearn\preprocessing\_encoders.py:390: DeprecationWarning: The
'categorical_features' keyword is deprecated in version 0.20 and will be removed in 0.22. You can
use the ColumnTransformer instead.
"use the ColumnTransformer instead.", DeprecationWarning)
```

Out[18]:

```
array([[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
1.3689780e+05, 4.7178410e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.5137759e+05, 4.4389853e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.0114555e+05, 4.0793454e+05],
...,
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.3542692e+05, 0.0000000e+00],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
5.1743150e+04, 0.0000000e+00],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.1698380e+05, 4.5173060e+04]])
```

. . .

1

```
### Avoiding Dummy Variable Trap(躲避虚拟变量陷阱)
```

In [22]:

1

```
X = X[:, 1:]
```

```
X
```

Out[22]:

```
array([[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
1.3689780e+05, 4.7178410e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.5137759e+05, 4.4389853e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.0114555e+05, 4.0793454e+05],
```

```
...,
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.3542692e+05, 0.0000000e+00],
[0.0000000e+00, 1.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
5.1743150e+04, 0.0000000e+00],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.1698380e+05, 4.5173060e+04]])
```

. . .

Splitting the dataset into the Training set and Test set(拆分数据集为训练集和测试集¶

)

In [24]:

3

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

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
```

```
X_train, X_test, Y_train, Y_test
```

Out[24]:

```
(array([[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.0305749e+05, 2.1463481e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
8.5047440e+04, 2.0551764e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.4413598e+05, 1.3405007e+05],
...,
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
1.3689780e+05, 4.7178410e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.3542692e+05, 0.0000000e+00],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, ..., 0.0000000e+00,
1.5480614e+05, 2.8334720e+04]]),
array([[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
```

0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0000000e+00,
0.0000000e+00, 1.8264556e+05, 1.1814820e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 9.1790610e+04, 2.4974455e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0000000e+00,
0.0000000e+00, 1.1059411e+05, 2.2916095e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,

```
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0000000e+00,
0.0000000e+00, 8.4710770e+04, 1.6447071e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0000000e+00,
0.0000000e+00, 1.0114555e+05, 4.0793454e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
1.0000000e+00, 1.2786455e+05, 3.5318381e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
```

```
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
1.0000000e+00, 6.5947930e+04, 1.8526510e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
1.0000000e+00, 1.5270192e+05, 8.8218230e+04],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0000000e+00,
0.0000000e+00, 1.2278275e+05, 3.0331926e+05],
[1.0000000e+00, 0.0000000e+00, 1.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 0.0000000e+00,
0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0000000e+00,
0.0000000e+00, 9.1391770e+04, 3.6616842e+05]]),
array([ 96778.92, 96479.51, 105733.54, 96712.8 , 124266.9 , 155752.6 ,
132602.65, 64926.08, 35673.41, 101004.64, 129917.04, 99937.59,
```

```
97427.84, 126992.93, 71498.49, 118474.03, 69758.98, 152211.77,
134307.35, 107404.34, 156991.12, 125370.37, 78239.91, 14681.4 ,
191792.06, 141585.52, 89949.14, 108552.04, 156122.51, 108733.99,
90708.19, 111313.02, 122776.86, 149759.96, 81005.76, 49490.75,
182901.99, 192261.83, 42559.73, 65200.33]),
array([103282.38, 144259.4 , 146121.95, 77798.83, 191050.39, 105008.31,
81229.06, 97483.56, 110352.25, 166187.94]))
```

. . .

Step 2: Fitting Multiple Linear Regression to the Training set(在训练集上训练多元线性回归模型)¶

In [29]:

3

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

```
regressor = LinearRegression()
```

```
regressor.fit(X_train,Y_train)
```

Out[29]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

. . .

Step 3: Predicting the Test set results(在测试集上预测结果)¶

In [30]:

2

```
y_pred=regressor.predict(X_test)
```

```
y_pred
```

Out[30]:

```
array([199284.29616179, 58975.97231506, 84818.06472524, 34825.52114832,
83740.4976782 , 136541.34536323, 15834.62401121, 156765.35558325,
112067.27328005, 63426.14627526])
```

. . .

In [ ]:

1

. . .