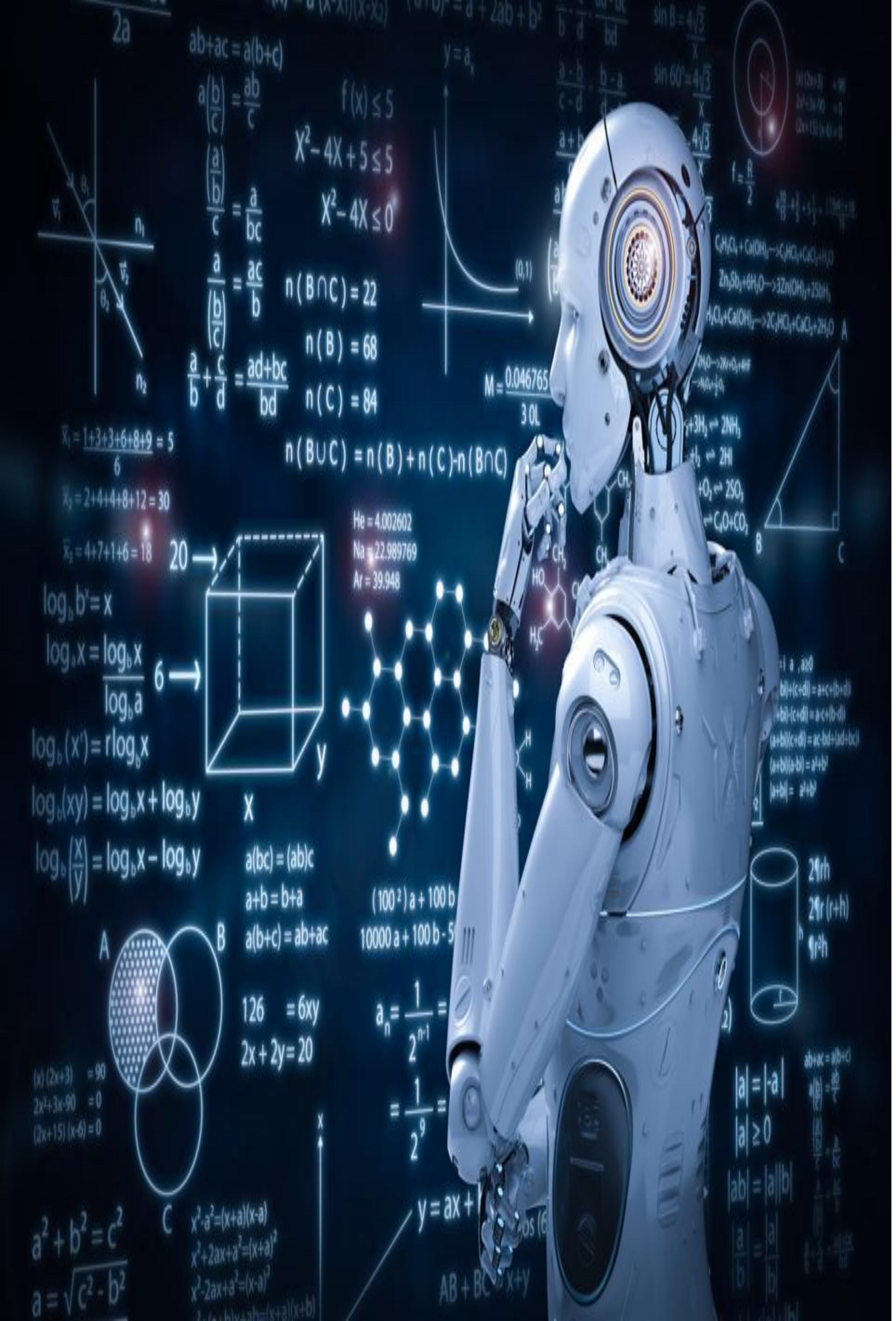


Practical Machine Learning

Day 6: Sep22 DBDA

Kiran Waghmare

Agenda

- Ridge, Lasso &ElasticNet
- Preprocessing Techniques
- Classification

jupyter KPML Day5 Regression

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo

Who can see what you share here? Recording On

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3 (ipykernel)

Run Code Commit

In [37]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train, y_train)
```

Out[37]:
```
LinearRegression()
```

Prediction

In [38]:
```
y_pred=reg.predict(X_test)
y_pred
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_tes
```

```
[[103015.2   103282.38]
 [132582.28  144259.4 ]
 [132447.74  146121.95]
 [ 71976.1    77798.83]
 [178537.48  191050.39]
 [116161.24  105008.31]
 [ 67851.69   81229.06]
 [ 98791.73   97483.56]
 [113969.44  110352.25]
```

Simple Linear Regression

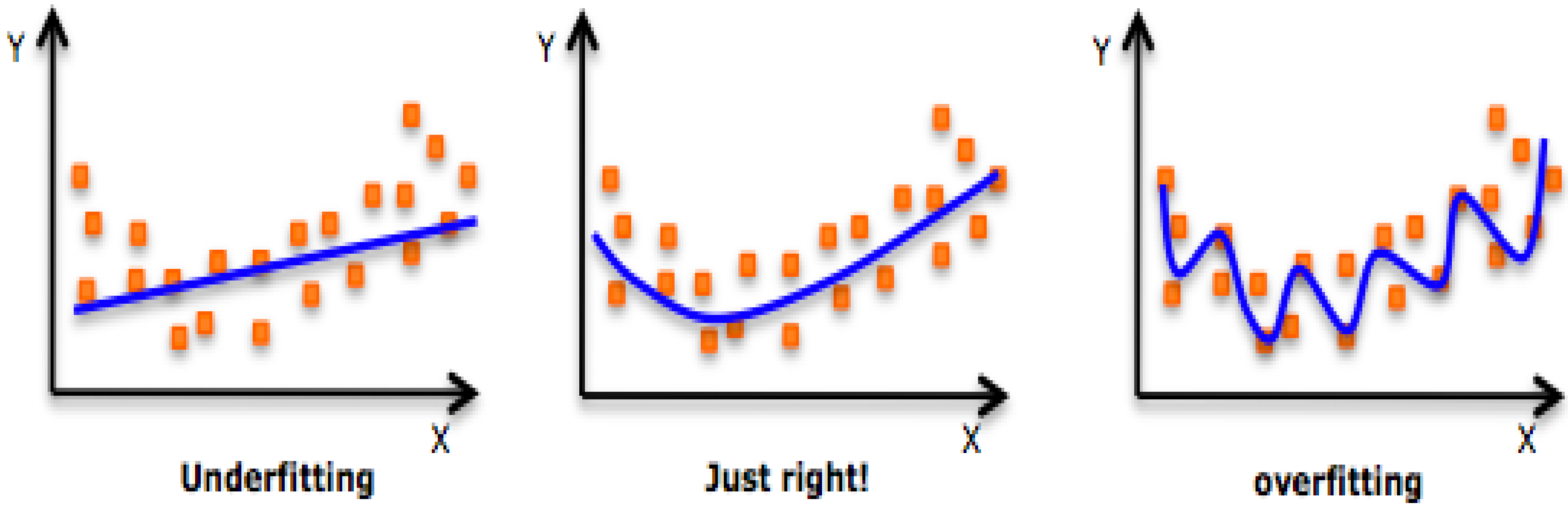
$$y = b_0 + b_1 x_1$$

Multiple Linear Regression

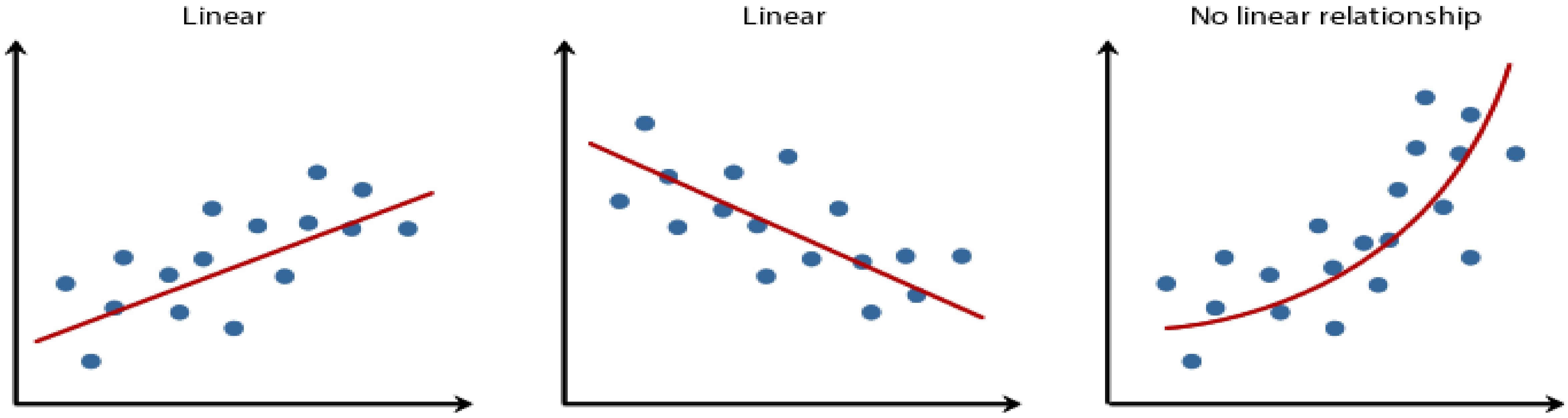
$$y = b_0 + b_1 x_1 + b_2 x_2 + ... + b_n x_n$$

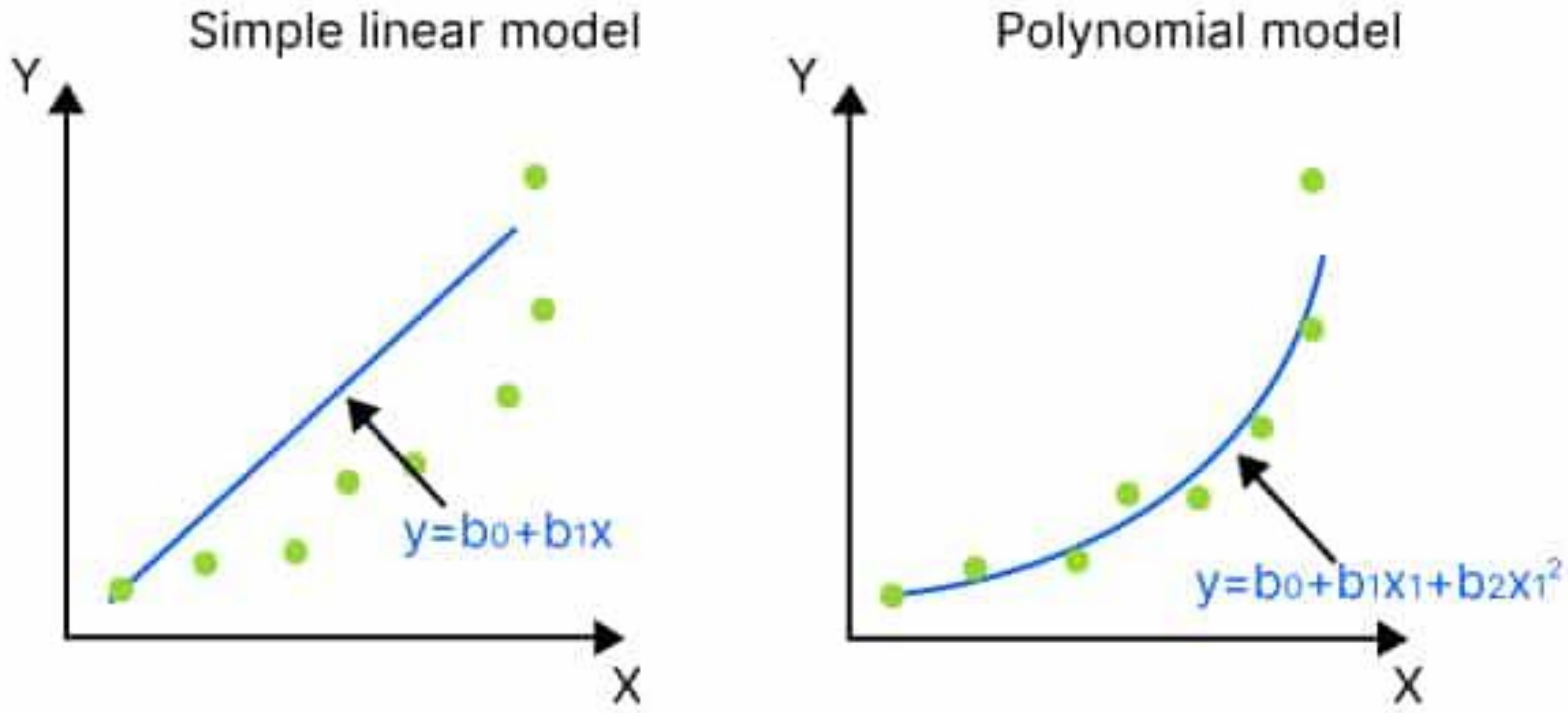
Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + ... + b_n x_1^n$$



Y
X
Underfitting
Y
X
Just right!
Y
X
overfitting


Linear
Linear
No linear relationship

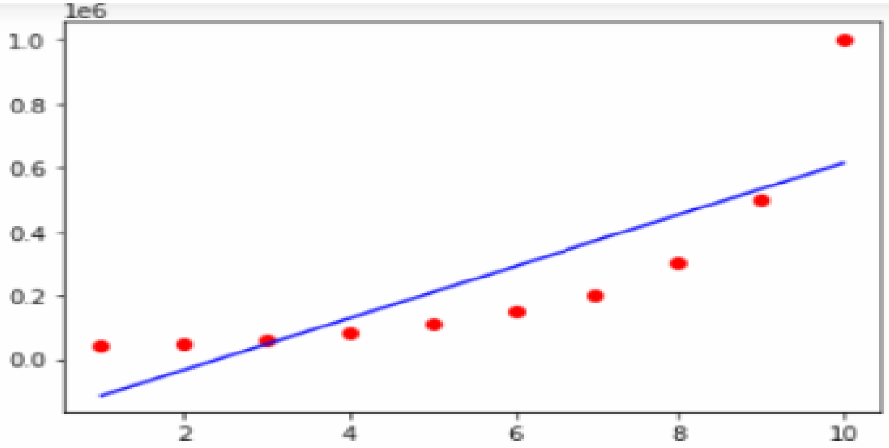


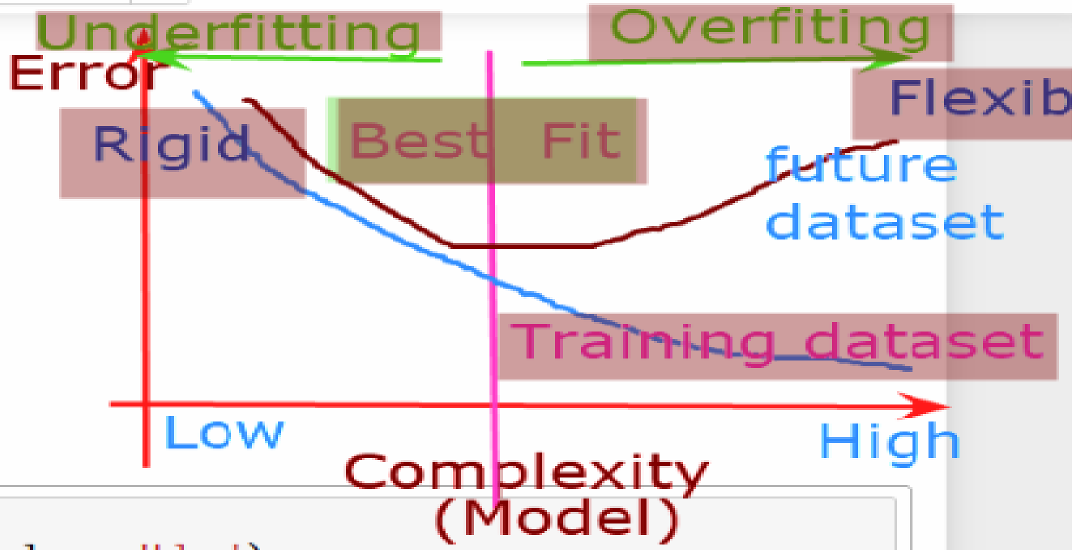
Simple linear model
Y
X
y=b0+b1x
Polynomial model
Y
X
y=b0+b1x1+b2x1²


THE BEST WAY TO
EXPLAIN OVERFITTING

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code Commit



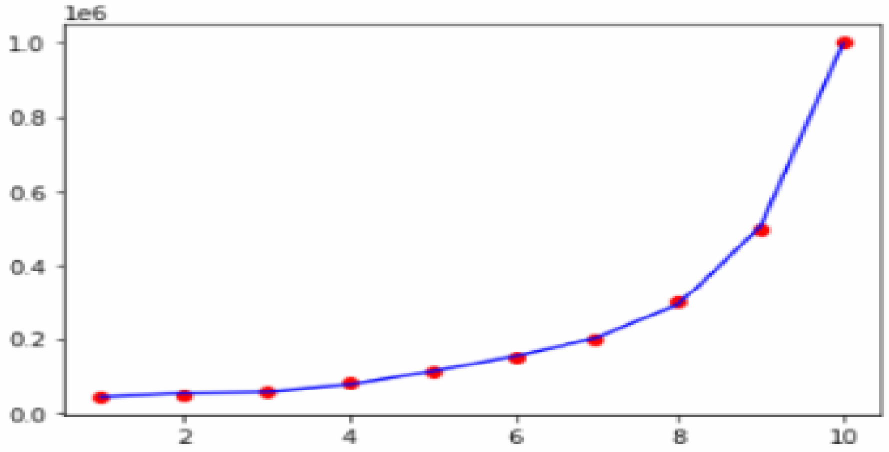


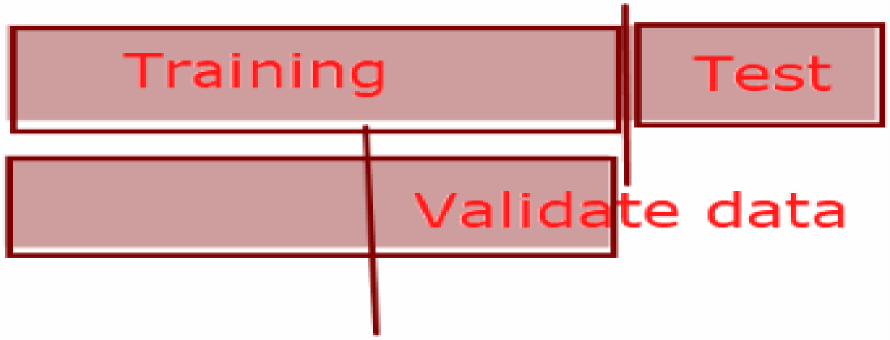
In [39]:

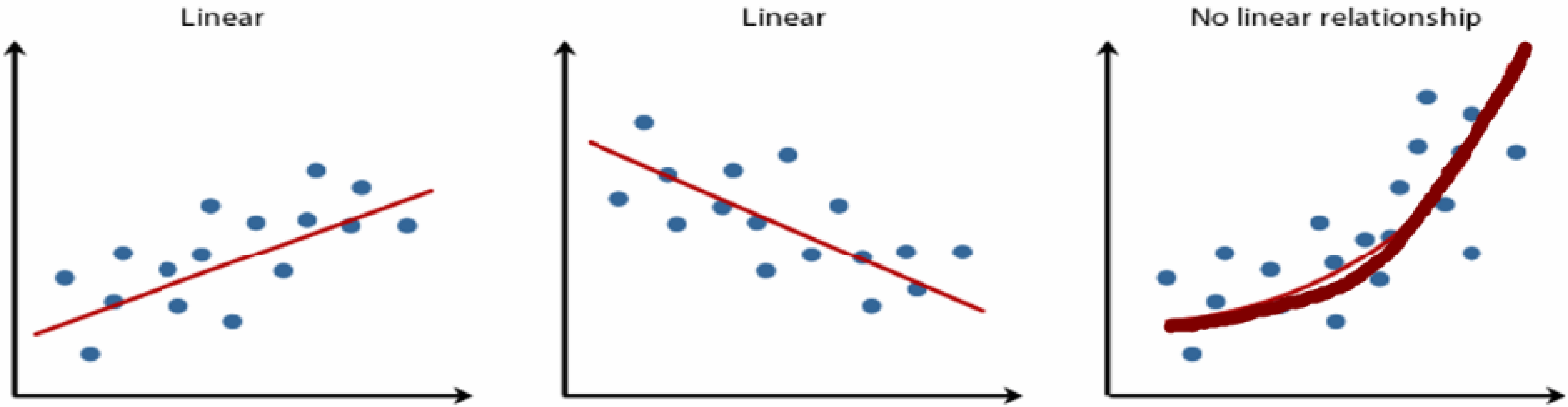
```
plt.scatter(X,y, color = 'red')
plt.plot(X,model1.predict(poly_reg.fit_transform(X)), color = 'blue')
```

Out[39]:

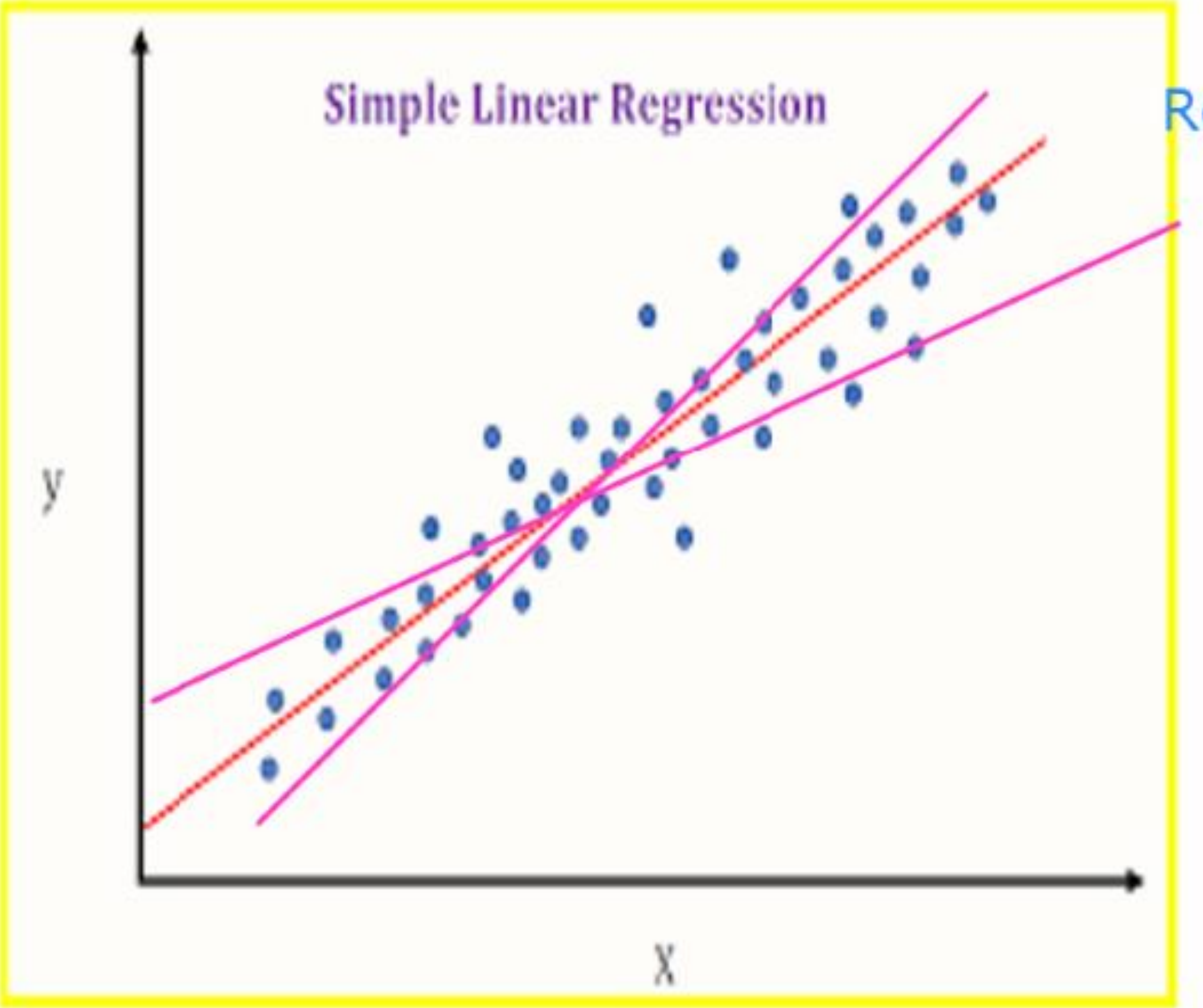
```
[<matplotlib.lines.Line2D at 0x2b561d8e160>]
```



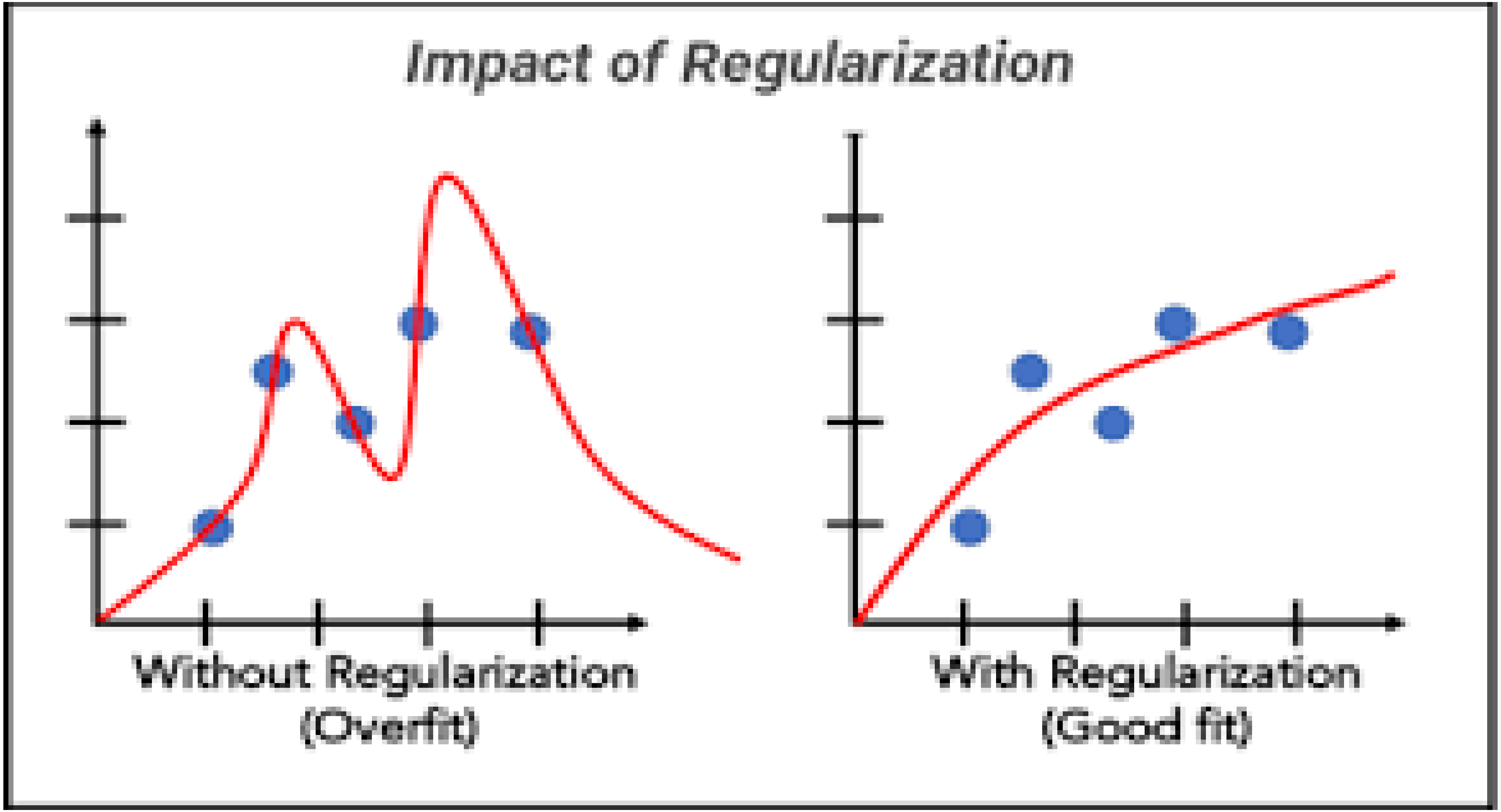




Overfitting ---> Good fit
(Regularization)



Regression line(best fit)


Impact of Regularization
Without Regularization
(Overfit)
With Regularization
(Good fit)

Regularization

Regularization Term

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 + \boxed{\boxed{\lambda} \sum_{j=1}^{n} {\theta_j}^2}$$

start at θ_1

Regularization Parameter

Regularization

x1	x2	x3	x4	y
		✓	✓	

Penalty

Regularization Term

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^{n} \theta_j^{\,2}$$

start at θ_1

Regularization Parameter

x1:1
x2:1.5
x3:90
x4:85

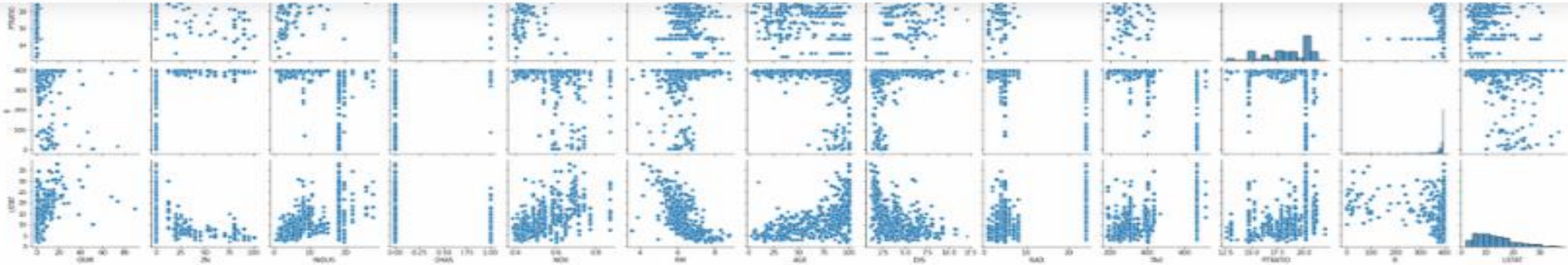
Loss function + Regularized term

Ridge
Lasso
ElasticNet

Who can see what you share here? Recording On

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

Run Code Commit



In [12]:
```python
new_data['Houseprice'] =data.target
new_data.head()
```

Out[12]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Houseprice	House_price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2	36.2

In [ ]:

4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

In [23]:
```
new_data.shape
```
Out[23]:
```
(506, 14)
```

In [24]:
```
X=new_data.iloc[:,0:13]
y=new_data.iloc[:,13]
```

In [25]:
```
X
```
Out[25]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

506 rows × 13 columns

Regularization

Regression:

-Ridge

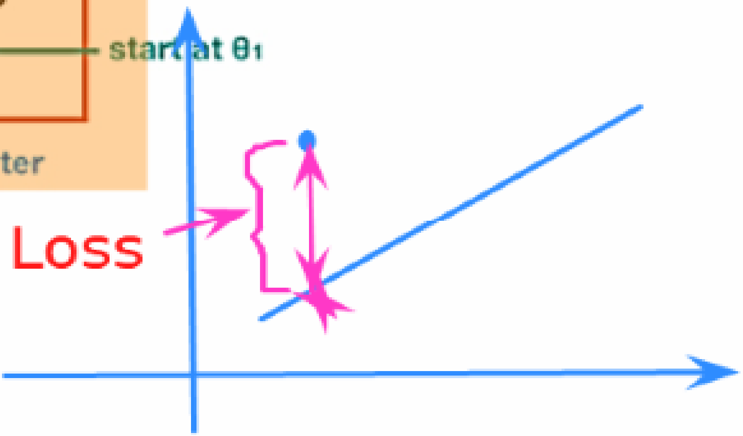
-Lasso

Regularization Term

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \boxed{\lambda}\sum_{j=1}^{n}\theta_j^2$$

start at θ_1

Regularization Parameter



Ridge Regression

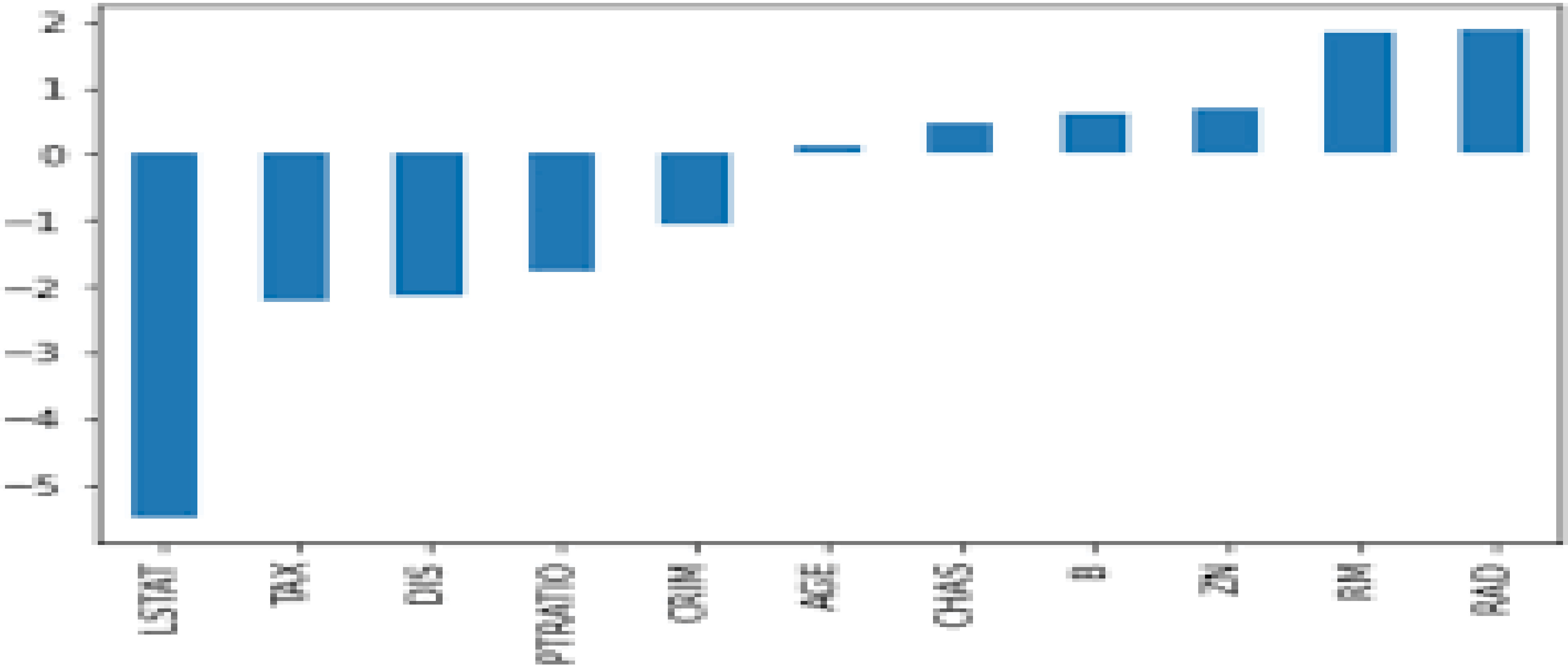
$$Regularization\ (L2) = Loss\ Function + \lambda \sum_{i=1}^{m} w_i^2$$

Ridge Regression

$$Regularization\ (L2) = Loss\ Function + \lambda \sum_{i=1}^{m} w_i^2$$

Lasso Regression

Lasso Regression $= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Penalty}}$


2
1
0
−1
−2
−3
−4
−5
LSTAT
TAX
DIS
PTRATIO
CRIM
AGE
CHAS
B
ZN
RM
RAD

Regularization

y=0.8+0.6x1+04x2+$\boxed{1232434}$x3

cost
time
computation

Regularization Term

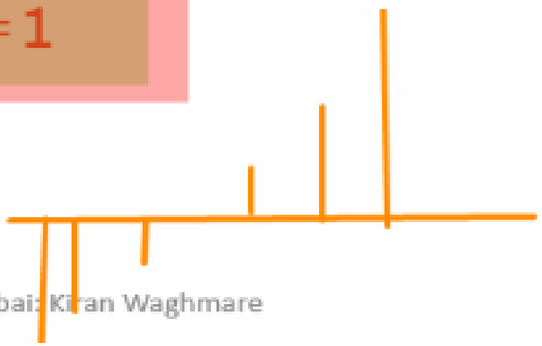
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \boxed{\lambda}\sum_{j=1}^{n}\theta_j^{\,2}$$

start at θ_1

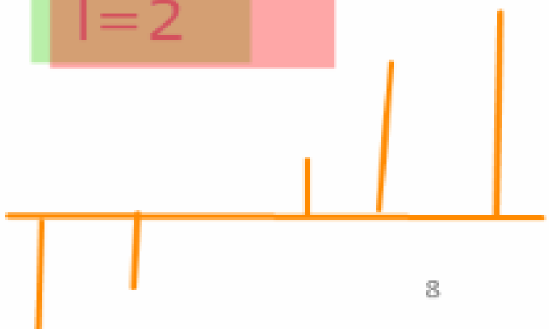
Regularization Parameter

l=0.01

l=1



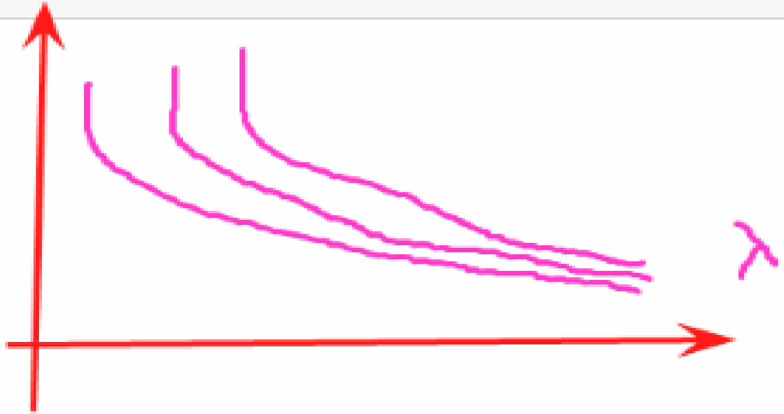
l=2



Out[21]: Ridge()

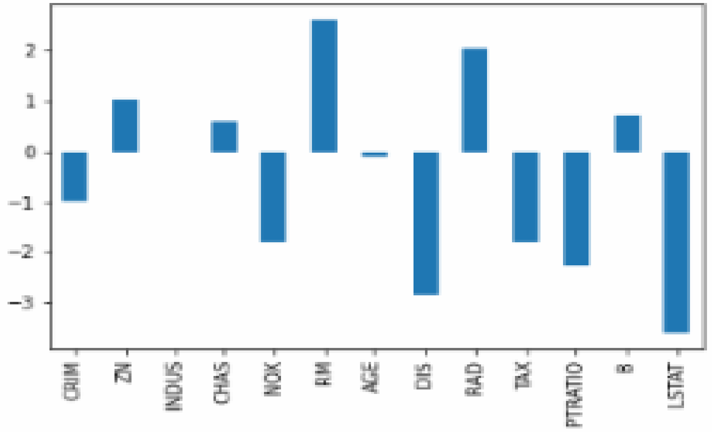
In [22]:
```
rid_coff=pd.Series(rid.coef_,index=data.feature_names)
rid_coff
```

Out[22]:
```
CRIM       -0.962257
ZN          1.040872
INDUS       0.011680
CHAS        0.598719
NOX        -1.820134
RM          2.583786
AGE        -0.095188
DIS        -2.848263
RAD         2.036231
TAX        -1.806092
PTRATIO    -2.283191
B           0.718310
LSTAT      -3.576073
dtype: float64
```



In [23]:
```
rid_coff.plot(kind='bar')
```

Out[23]: <AxesSubplot:>



In [ ]:
```
#Lasso Algorithm
from sklearn.linear_model import Lasso
```

In [25]:
```
la_coff=pd.Series(la.coef_,index=data.feature_names)
la_coff
```

Out[25]:
```
CRIM      -0.000000
ZN         0.000000
INDUS     -0.000000
CHAS       0.000000
NOX       -0.000000
RM         2.540098
AGE       -0.000000
DIS       -0.000000
RAD       -0.000000
TAX       -0.171527
PTRATIO   -1.784796
B          0.110959
LSTAT     -3.585324
dtype: float64
```

α / λ

Feature Selection

Regularization

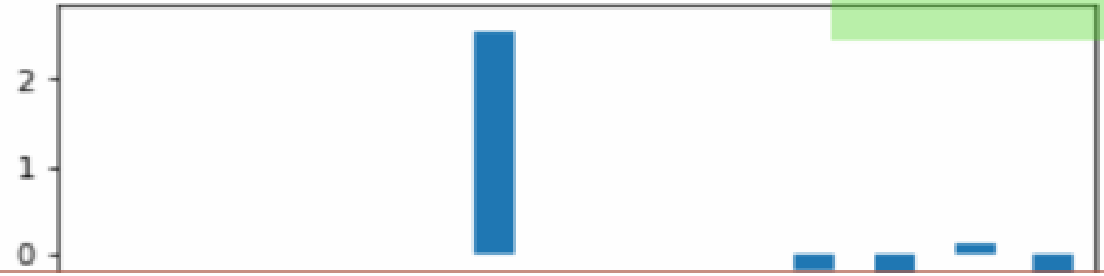
Lasso + Ridge

Elasticnet

Multi feature correlation

In [26]:
```
la_coff.plot(kind='bar')
```

Out[26]: <AxesSubplot:>

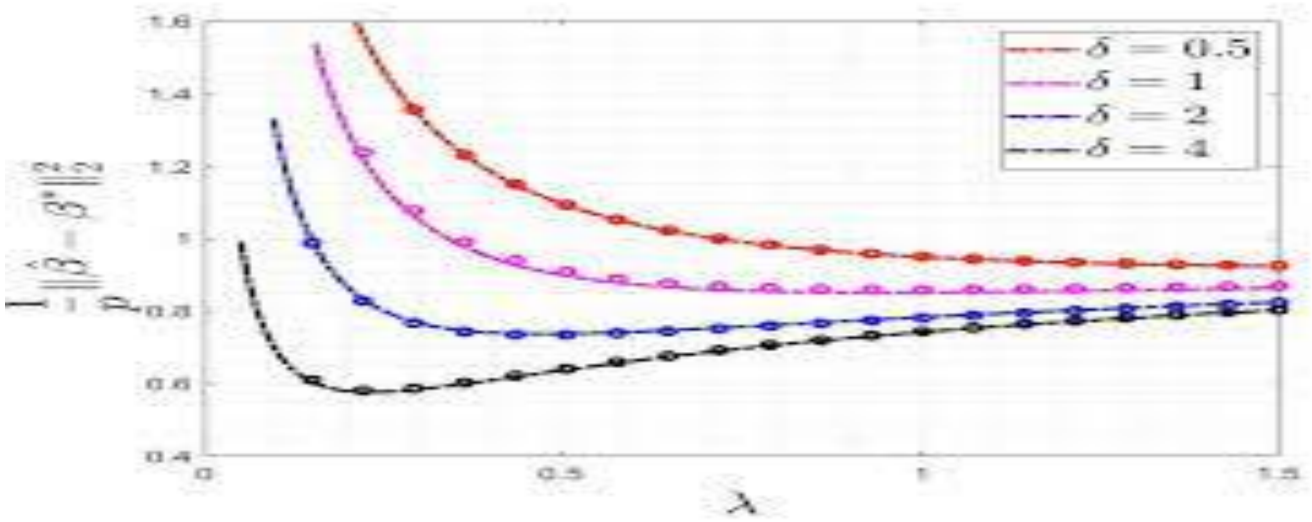


Transforming the Loss function into Lasso Regression

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \Rightarrow \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} |w_j|$$

Loss function → **Loss function + Regularized term**

Designed by Author (Shanthababu)



Ridge Regression

Ridge regression uses the mean squared error loss function and applies L2 Regularization. Its cost function *J(θ)* is given as

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y - \hat{y})^2 + \lambda \sum_{j=1}^{n} w_j^2$$

where,

$\frac{1}{m}\sum_{i=1}^{m}(y - \hat{y})^2$ is the Mean Squared error (loss function)

$\lambda \sum_{j=1}^{n} w_j^2$ is the penalty (L2 Regularization)

Now, substitute $\hat{y}$ as $wx_i + b$.

Lasso Regression

Lasso regression uses the same mean squared error loss function and this applies L1 Regularization and will repeat the same steps as Ridge. The cost function of Lasso Regression *J(θ)* is given as

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y - \hat{y})^2 + \lambda\sum_{j=1}^{n}\left|w_j\right|$$

where

$\lambda \sum_{j=1}^{n}\left|w_j\right|$ is the penalty (L1 Regularization).