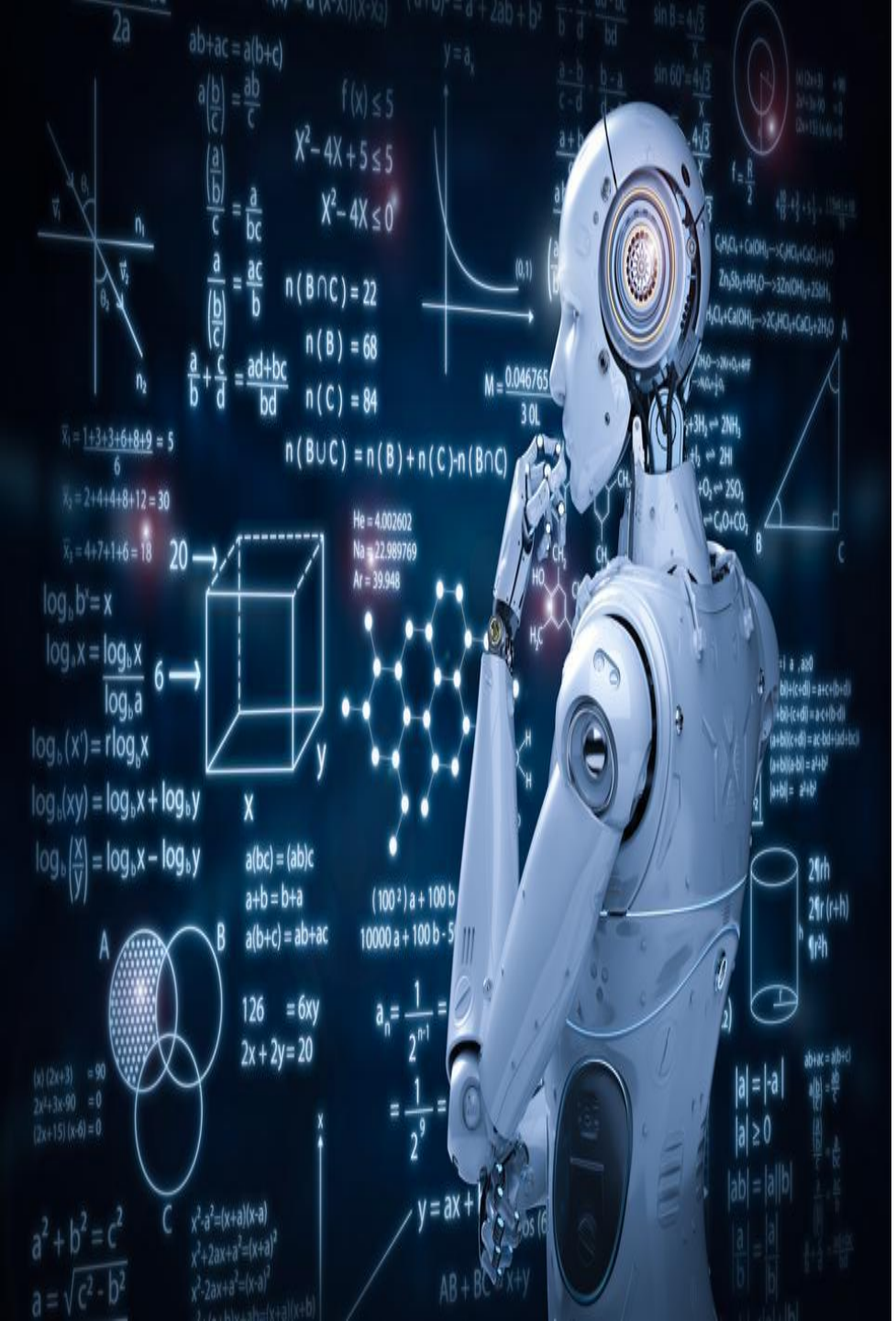


Practical Machine Learning

Day 5: Sep22 DBDA

Kiran Waghmare

Agenda

- Regression
- Types of Regression

Who can see what you share here? Recording On

4.Data Preprocessing
    -Encoders
    -Scaling

5.Splitting the dataset into Training set and Testing set.(.20)

6. Apply the modelling algorithm
    -LinearRegression()

7.Prediction for the testing datset

8.Visualising the training and testing dataset

9.Metrics

10.summary for model.

jupyter Untitled (autosaved)

Logout

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

Run Code Commit

In [12]:
```
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
```

In [13]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
```

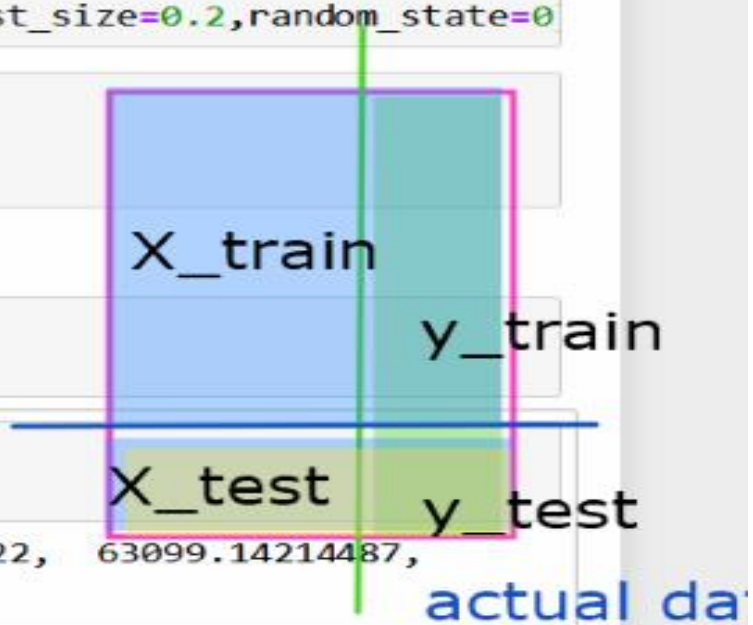
Out[13]:
```
LinearRegression()
```

In [14]:
```
#Prediction
y_pred=model.predict(X_test)
```

predicted data

In [15]:
```
#Predicted values
y_pred
```

Out[15]:
```
array([ 40748.96184072, 122699.62295594,  64961.65717022,  63099.14214487,
       115249.56285456, 107799.50275317])
```



In [18]:
```
#Training data
plt.scatter(X_train,y_train,color='red')
plt.plot(X_train,model.predict(X_train), color='blue')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.title('Distribution of salary vs experience')
```

Out[18]:
```
Text(0.5, 1.0, 'Distribution of salary vs experience')
```

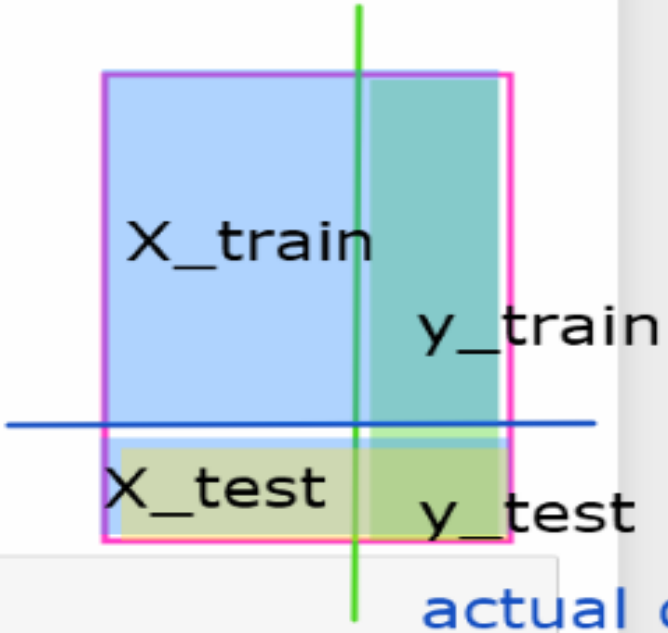
jupyter Untitled (unsaved changes)

Logout

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

Run Code Commit

y=mx+c




```
In [20]: #Testing data
         plt.scatter(X_test,y_test,color='red')
         plt.plot(X_test,model.predict(X_test), color='blue')
         plt.xlabel('Experience')
         plt.ylabel('Salary')
         plt.title('Distribution of salary vs experience')
```

```
Out[20]: Text(0.5, 1.0, 'Distribution of salary vs experience')
```

jupyter Untitled (autosaved)

Logout

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

Run Code Commit

Out[18]: Text(0.5, 1.0, 'Distribution of salary vs experience')

In [20]:
```
#Testing data
plt.scatter(X_test,y_test,color='red')
plt.plot(X_test,model.predict(X_test), color='blue')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.title('Distribution of salary vs experience')
```

Out[20]: Text(0.5, 1.0, 'Distribution of salary vs experience')

$$y = m X + C$$

Dependent variable | Y intercept | Slope | Error

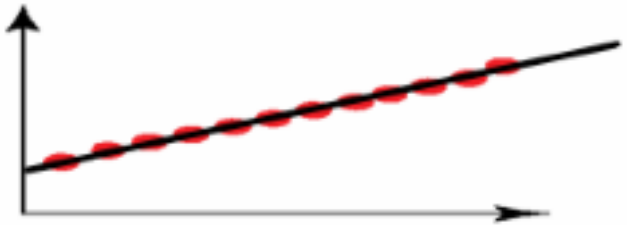
$$Y = \beta_0 + \beta_1 X + \varepsilon$$

Random Error Componenet

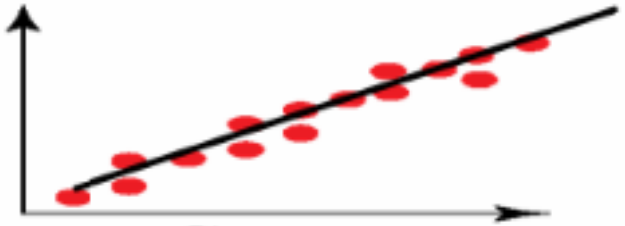
First order Linear Equation

b >0; +ve: x,y(incre)
b <0; -ve: x(in),y(dec)
b =0; no effect

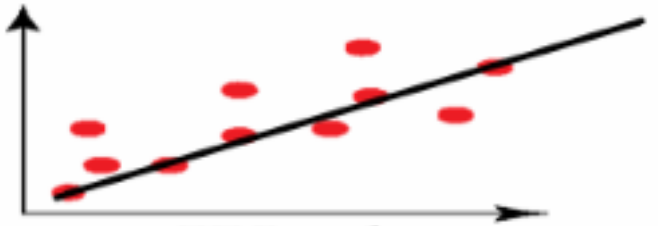
r>0



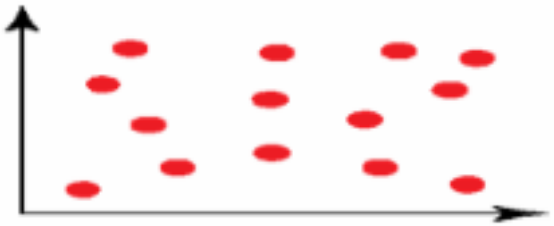
Perfect Positive Correlation



Strong Positive Correlation

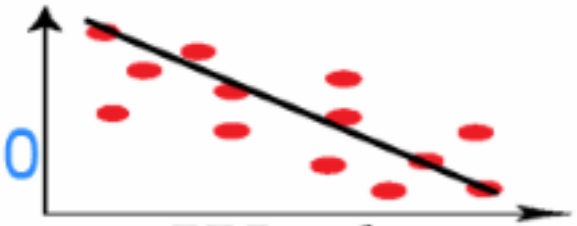


Weak Positive Correlation

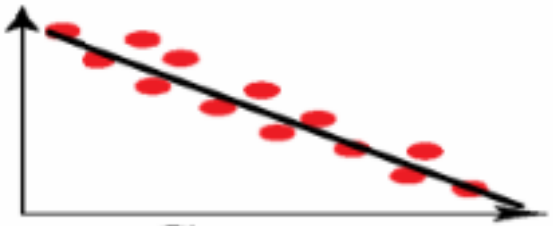


No Correlation

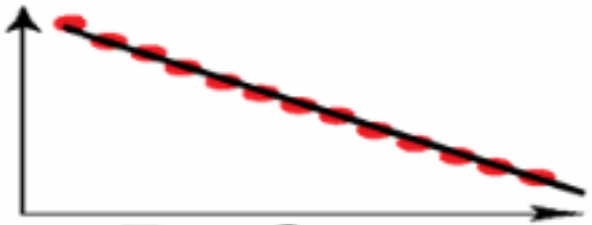
r<0



Weak Negative Correlation



Strong Negative Correlation

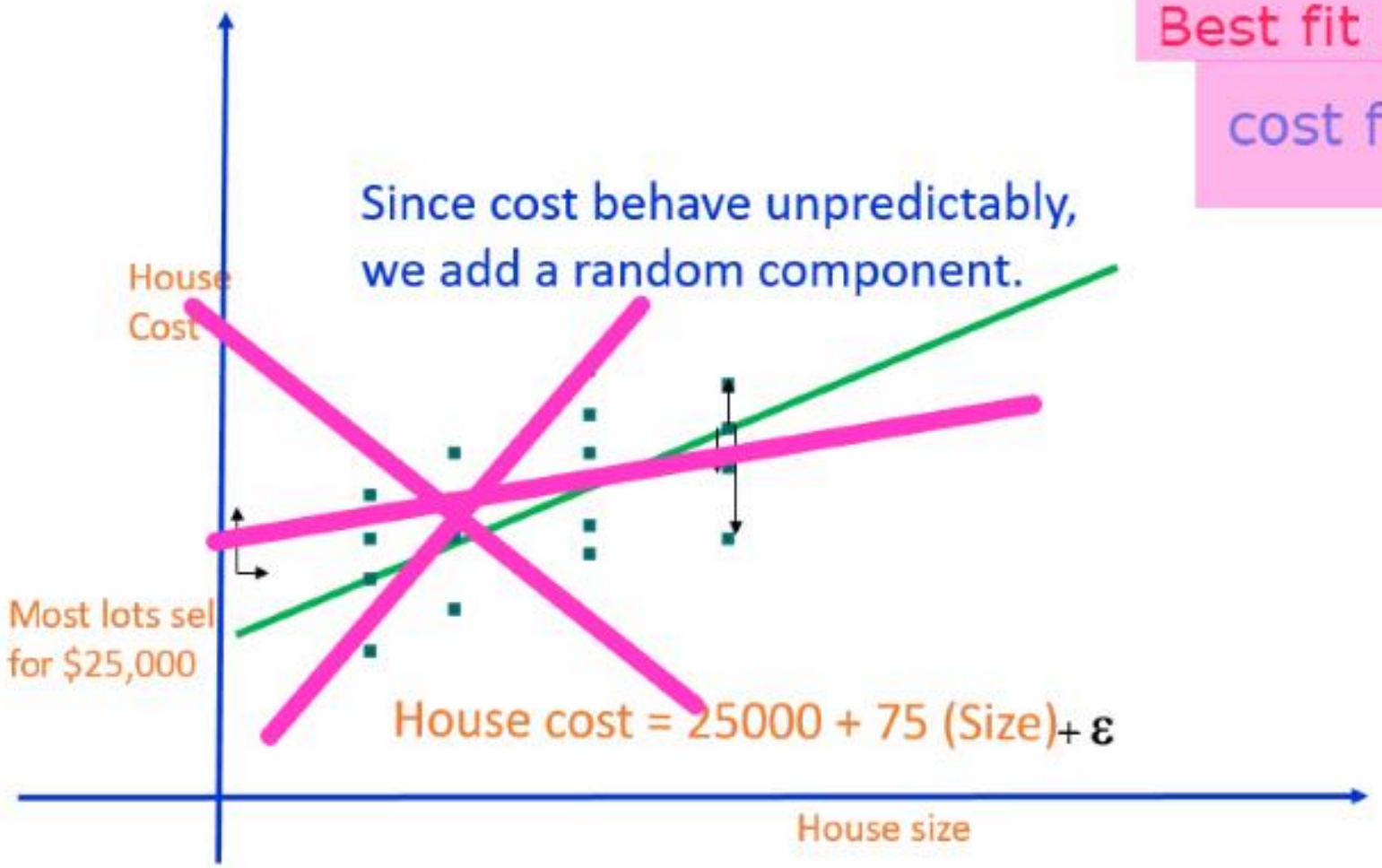


Perfect Negative Correlation

However, house cost vary even among same size houses!

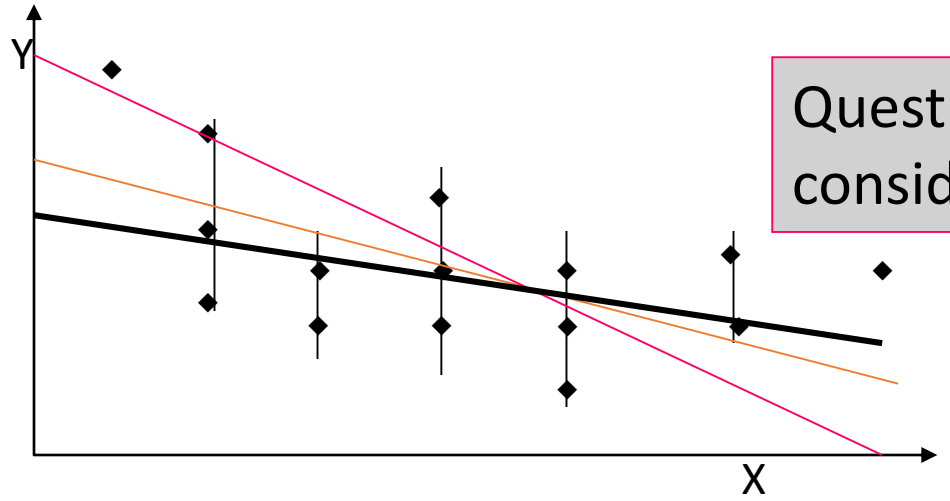
Best fit line

cost function



Estimating the Coefficients

- The estimates are determined by
  - drawing a sample from the population of interest,
  - calculating sample statistics.
  - producing a straight line that cuts into the data.



Question: What should be considered a good line?

- $MeanSquaredError(mse) = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2}$
- $MeanAbsoluteError(mae) = (\frac{1}{n})\sum_{i=1}^{n}|y_i - x_i|$

Actual value

predicted value

The Estimated Coefficients

To calculate the estimates of the line coefficients, that minimize the differences between the data points and the line, use the formulas:

$$b_1 = \frac{\text{cov}(X,Y)}{s_X^2}\left(= \frac{s_{XY}}{s_X^2}\right)$$

$$b_0 = \overline{Y} - b_1\overline{X}$$

The regression equation that estimates the equation of the first order linear model is:

$$\hat{Y} = b_0 + b_1 X$$

The Least Squares (Regression) Line

A good line is one that **minimizes the sum of squared** differences between the points and the line.

Sum of Squares for Errors

- This is the sum of differences between the points and the regression line.
- It can serve as a measure of how well the line fits the data.

  SSE is defined by

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

  - A shortcut formula

$$\text{SSE} = (n-1)s_Y^2 - \frac{[\text{cov}(X,Y)]^2}{s_X^2}$$

Types of Linear Regression

- Linear regression can be further divided into two types of the algorithm:
- **Simple Linear Regression:**

  If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.
- **Multiple Linear regression:**

  If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Simple Linear Regression

$$y = b_0 + b_1 x_1$$

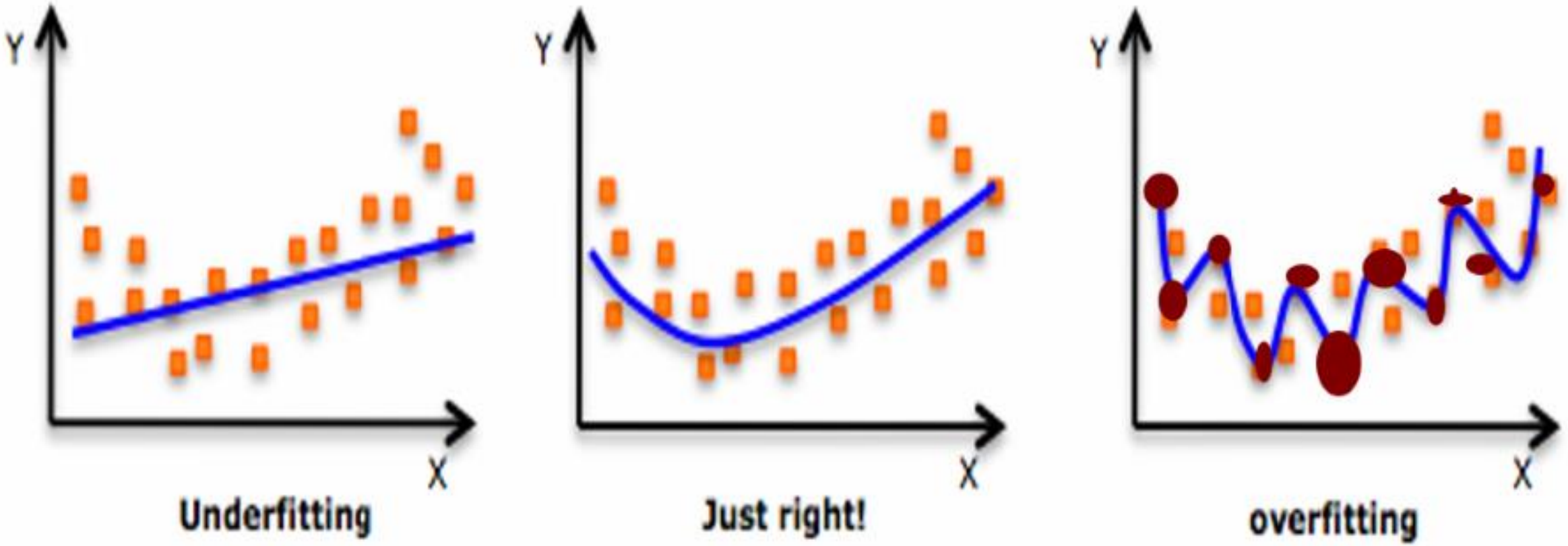
Multiple Linear Regression

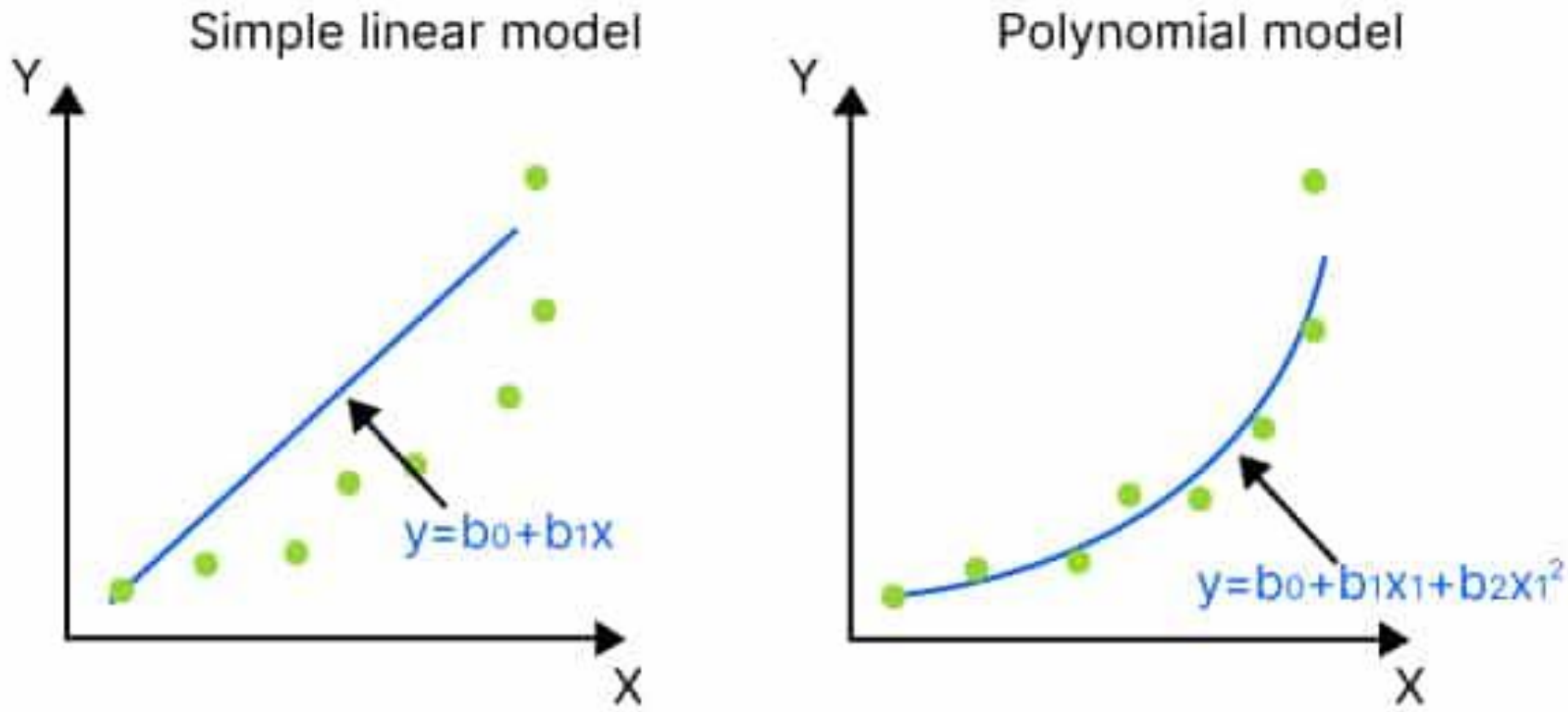
$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

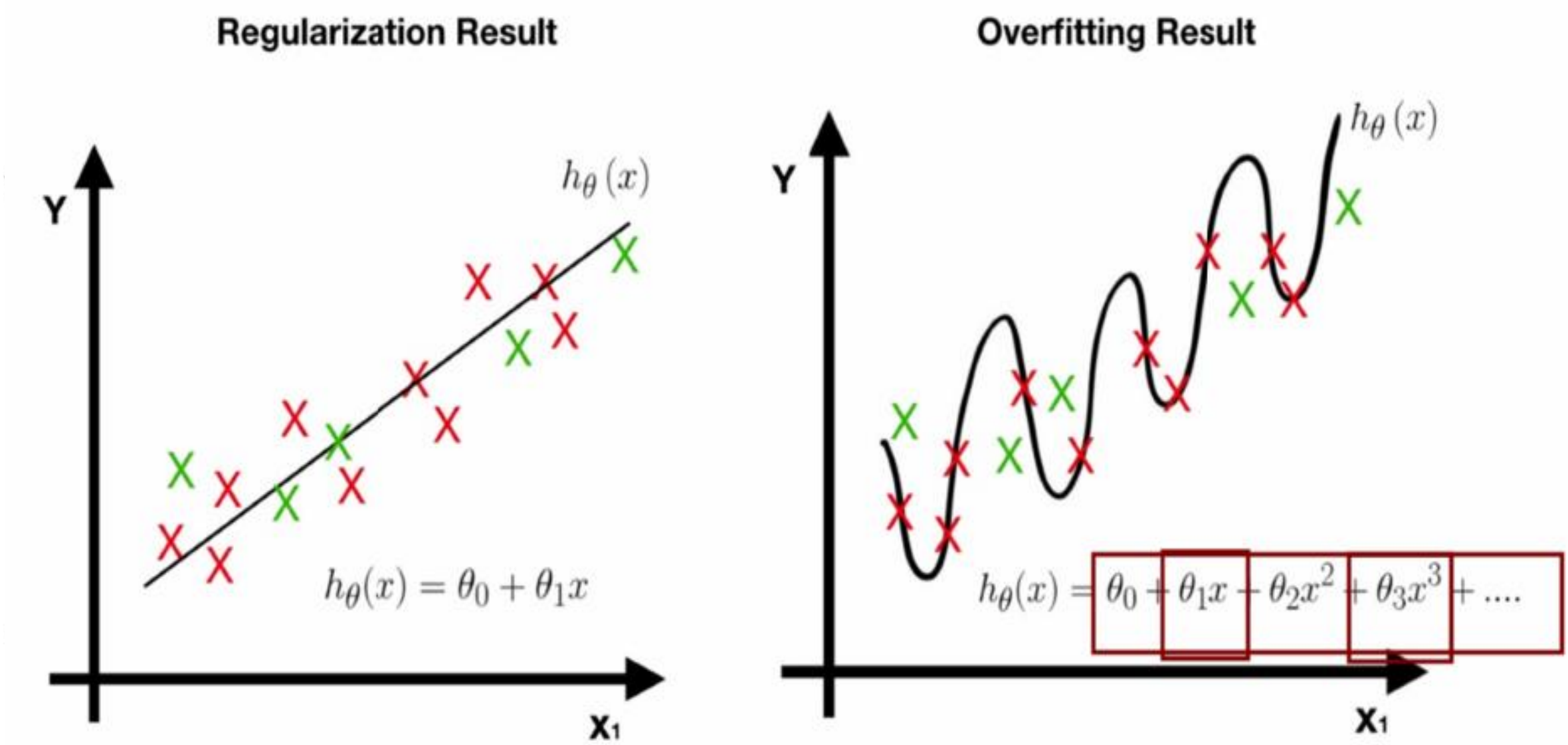
Polynomial Linear Regression

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

1, 1
2, 4
3, 9


Y
X
Underfitting
Y
X
Just right!
Y
X
overfitting


Simple linear model
Y
X
y=b0+b1x
Polynomial model
Y
X
y=b0+b1x1+b2x1^2


Regularization Result
Y
$h_\theta(x)$
$h_\theta(x) = \theta_0 + \theta_1 x$
X1
Overfitting Result
Y
$h_\theta(x)$
$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + ...$
X1

Multiple Linear Regression

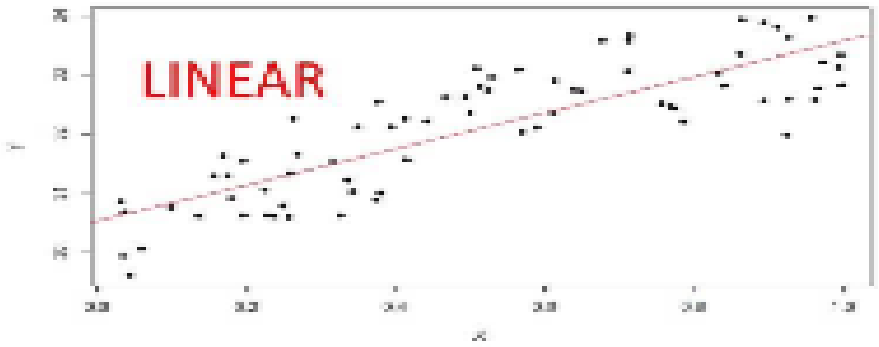
In [36]:
```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [40]:
```
dataset=pd.read_csv('D:/Test/50_Startups.csv')
dataset
```

Out[40]:

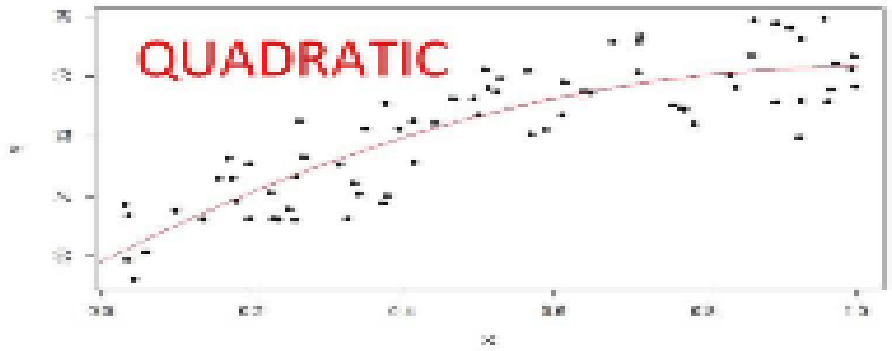
	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
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

Multicollinearity



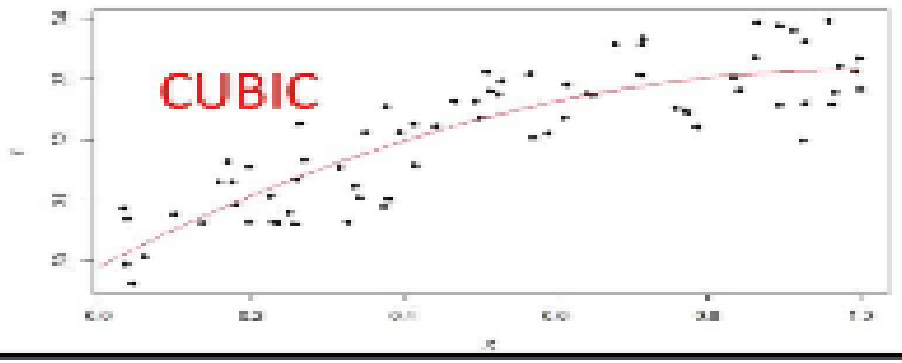
Multiple R-squared: 0.7044

$Y = 30.53 + 3.05*X$



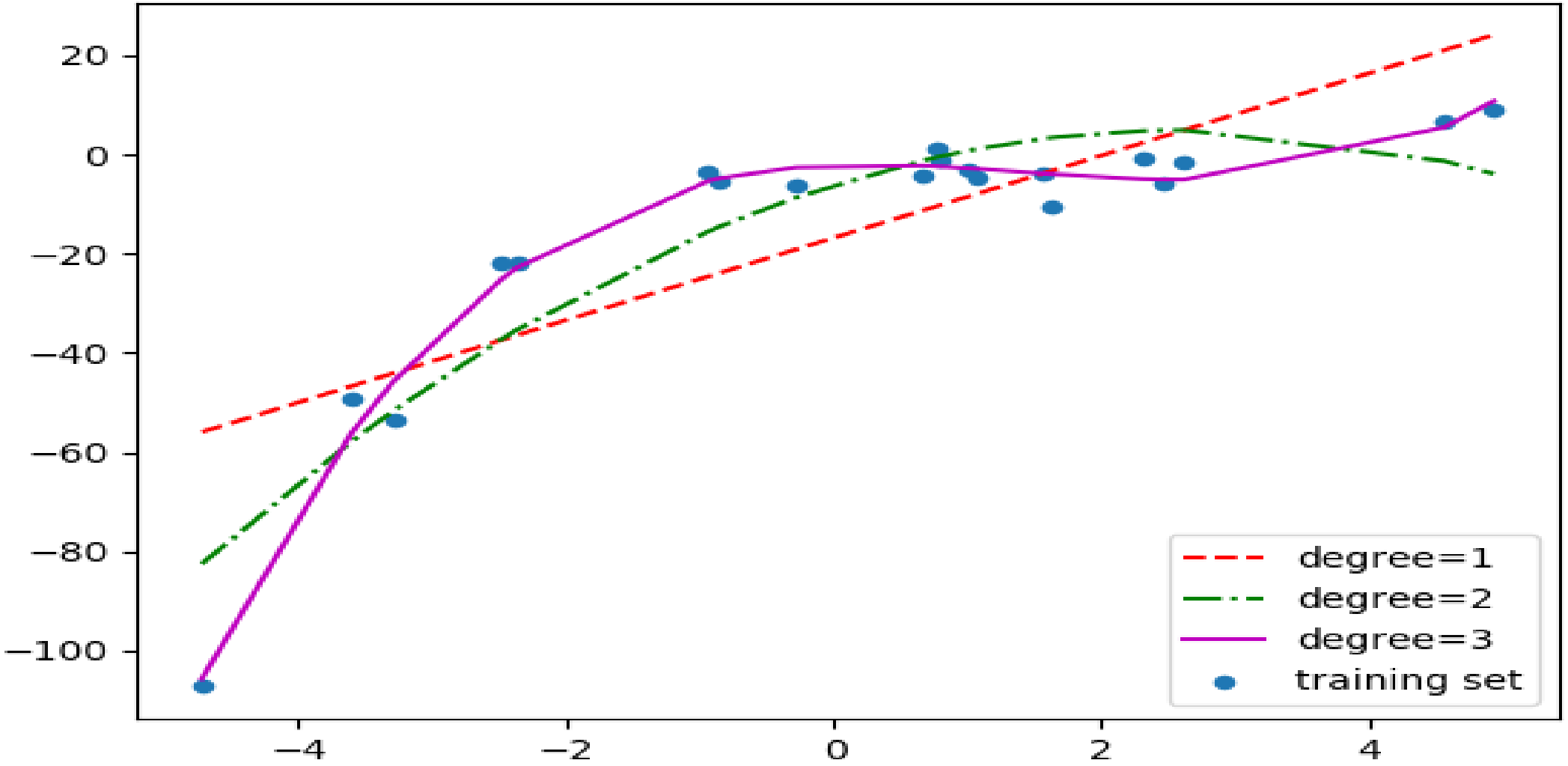
Multiple R-squared: 0.7559

$Y = 29.90 + 6.48*X - 3.22*X^2$



Multiple R-squared: 0.7623

$Y = 30.17 + 3.61*X + 3.71*X^2 - 4.48*X^3$


20
0
−20
−40
−60
−80
−100
−4
−2
0
2
4
degree=1
degree=2
degree=3
training set