Regression

Puzzle

Var1(X)	Output(Y)
2	4
1	2
5	10
3	6
7	?

$\boxed{y = 2x}$

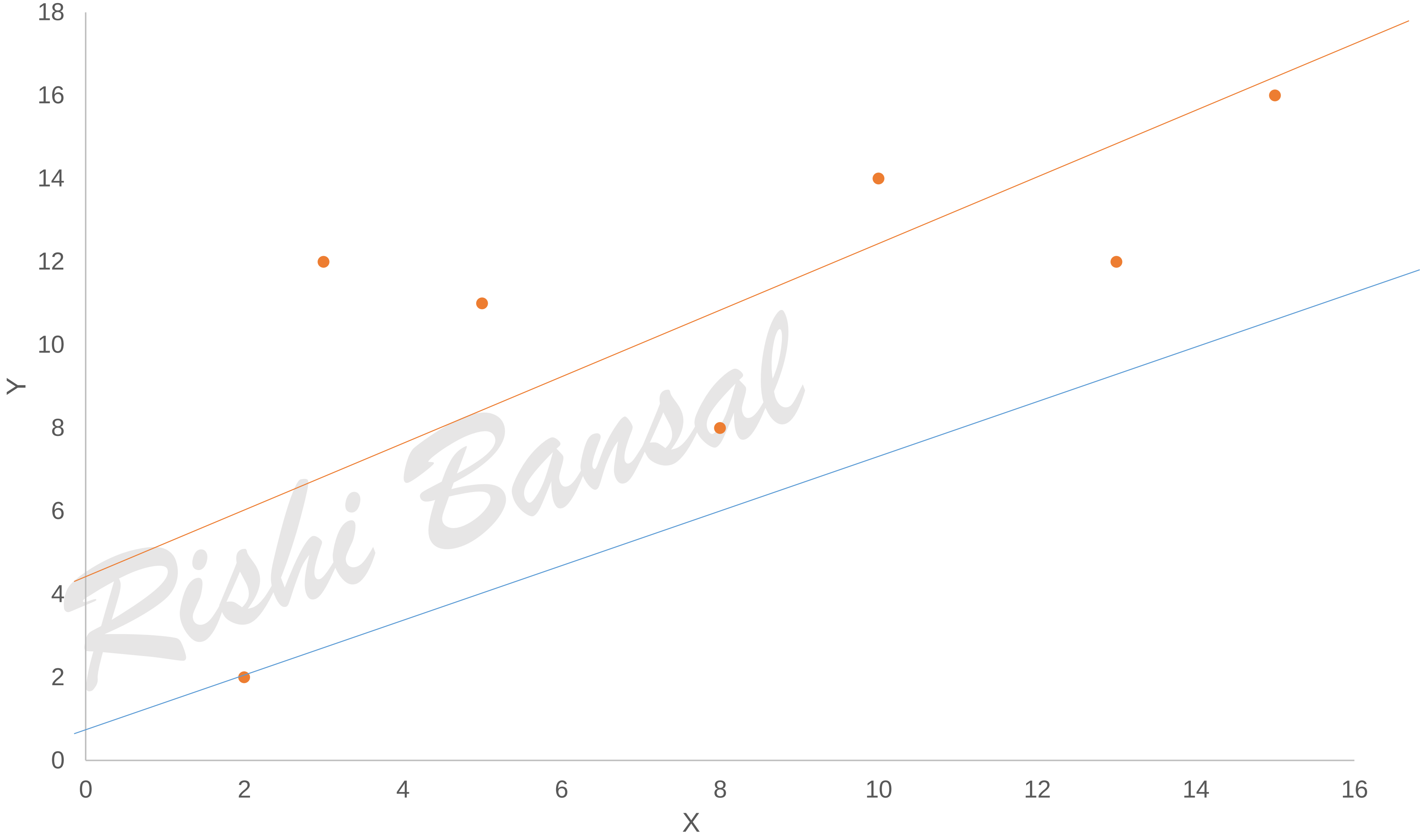
Var1	Var2	Output(Y)
2	1	5
1	1	3
5	1	11
3	1	7
7	1	?

$y = 2*(var1) + (var2)$

$y = mx + c$

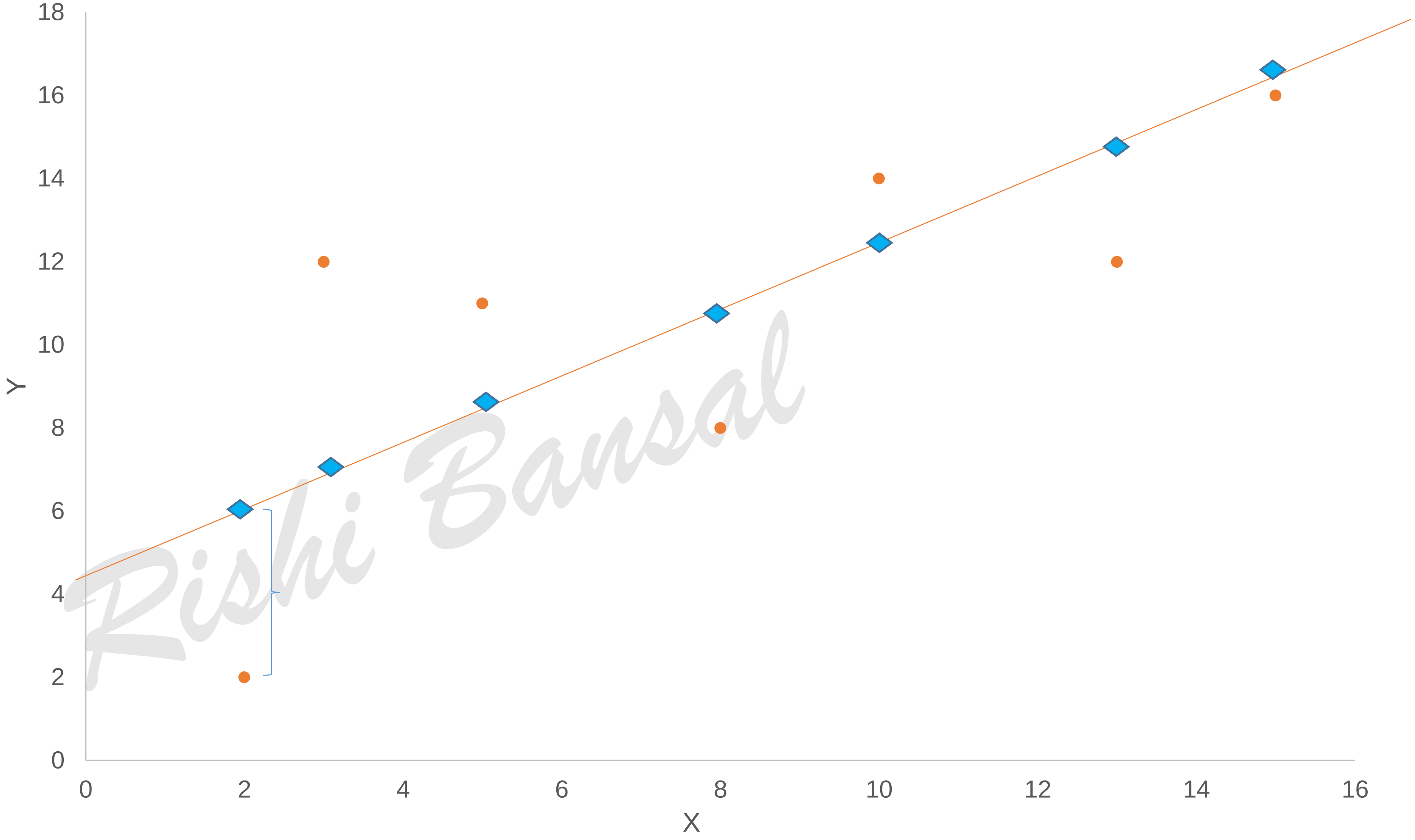
X	Y
2	2
3	12
5	11
8	8
10	14
13	12
15	16

Line Equation:
y = c + mx



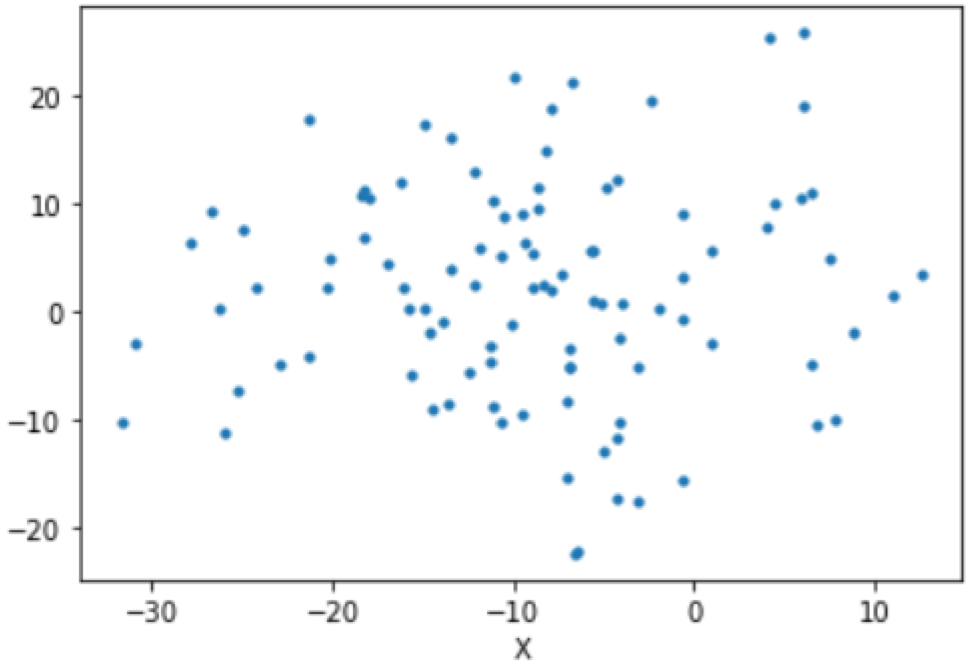
X	Y
2	2
3	12
5	11
8	8
10	14
13	12
15	16

Line Equation:
y = c + mx



Non-Linear Data

- Linear models won't work for such data, there is no linear relationship between X & Y
- Use non-linear models here
- E.g: Support Vector Regressor(rbf kernels), Random Forest Regressor



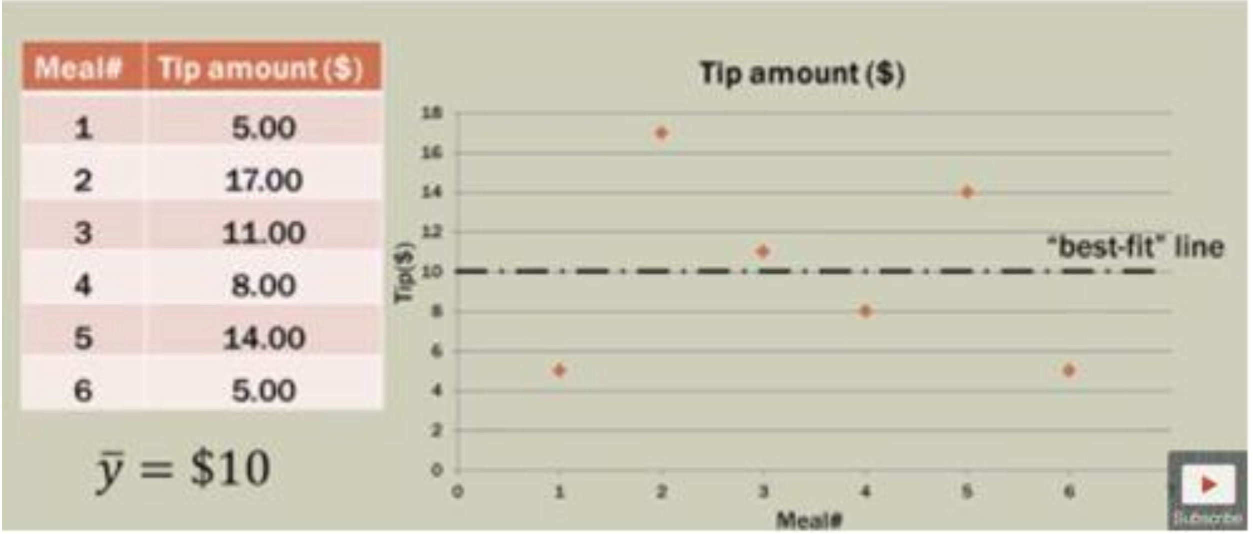
Simple Linear Regression

- **Def Regression**: Using the relationship between variables to find the best fit line or the regression equation that can be used to make predictions.
- If we have only one variable then Mean is the best way to predict

$\hat{y} = b_o + b_1 x$

here $b_1 = 0$

$\hat{y} = b_o = 10$ (because mean is the best fit, i.e, $b_o = 10$)



Meal#	Tip amount ($)
1	5.00
2	17.00
3	11.00
4	8.00
5	14.00
6	5.00

$\bar{y} = \$10$

Bivariate(Two Variable) Statistics

- The Best fit line will always pass from Centroid
- One dependent and one independent variable

$\hat{y} = b_o + b_1$x

$\hat{y}$ = Mean(Expected) value of y for a given value of x

Least Square Criterian: We need to minimize this term $\sum(y_i - \hat{y}_i)^2$

Minimizing the Cost Function

- Error = (y_pred – y_act)^2

- Two Methods:

1. Least Square Criterian (OLS)
2. Gradient Descent

Ordinary Least Square(OLS)

- non-iterative method that fits a model such that the sum-of-squares of differences of observed and predicted values is minimized
- Error = (y_pred – y_act)^2
- Line => y = bo + b1x

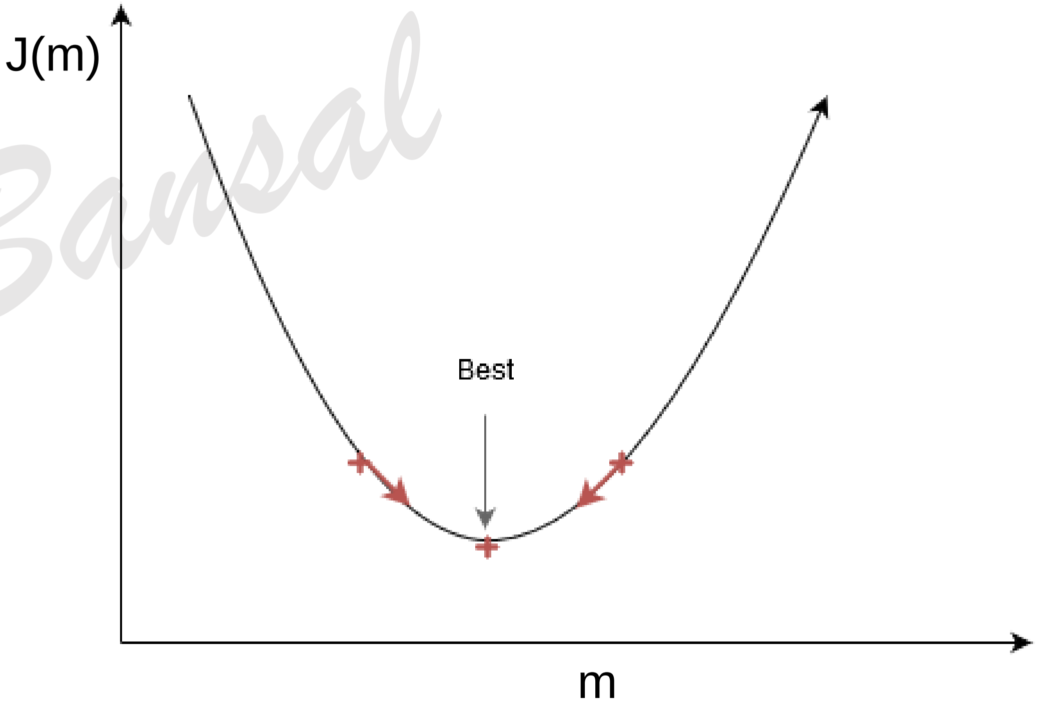
$$b_o = \bar{y} - b_1 \bar{x}$$
$$b_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2}$$

Gradient Descent

- Cost Function, J(m,c) = (y_pred – y_act)^2 / No. of data point
- Hypothesis: y_pred = c + mx
- c = 0

$$J(m) = (mx - y)^2$$

$$\frac{\partial J(m)}{\partial m}$$



Partial Derivative:https://www.mathsisfun.com/calculus/derivatives-partial.html

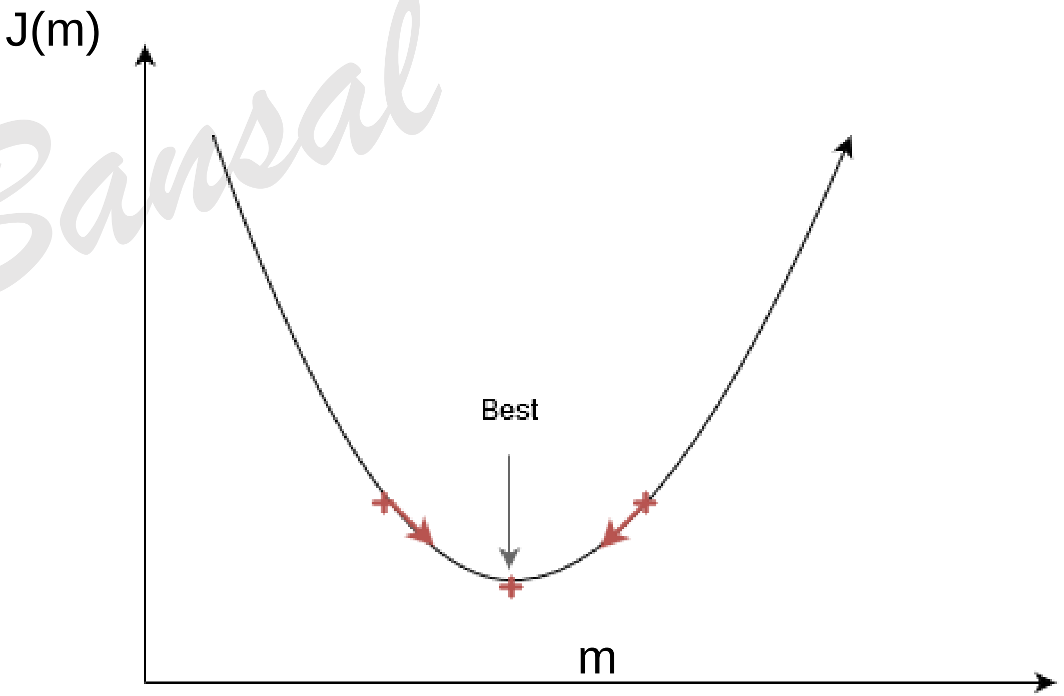
Learning Rate

- As we approach local Minimum, gradient descent will automatically take smaller steps.
- So, no need to decrease alpha over time

$$m = m - \frac{\partial J(m)}{\partial m}$$

- With Learning Rate

$$m = m - \alpha * \frac{\partial J(m)}{\partial m}$$



Algorithm

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

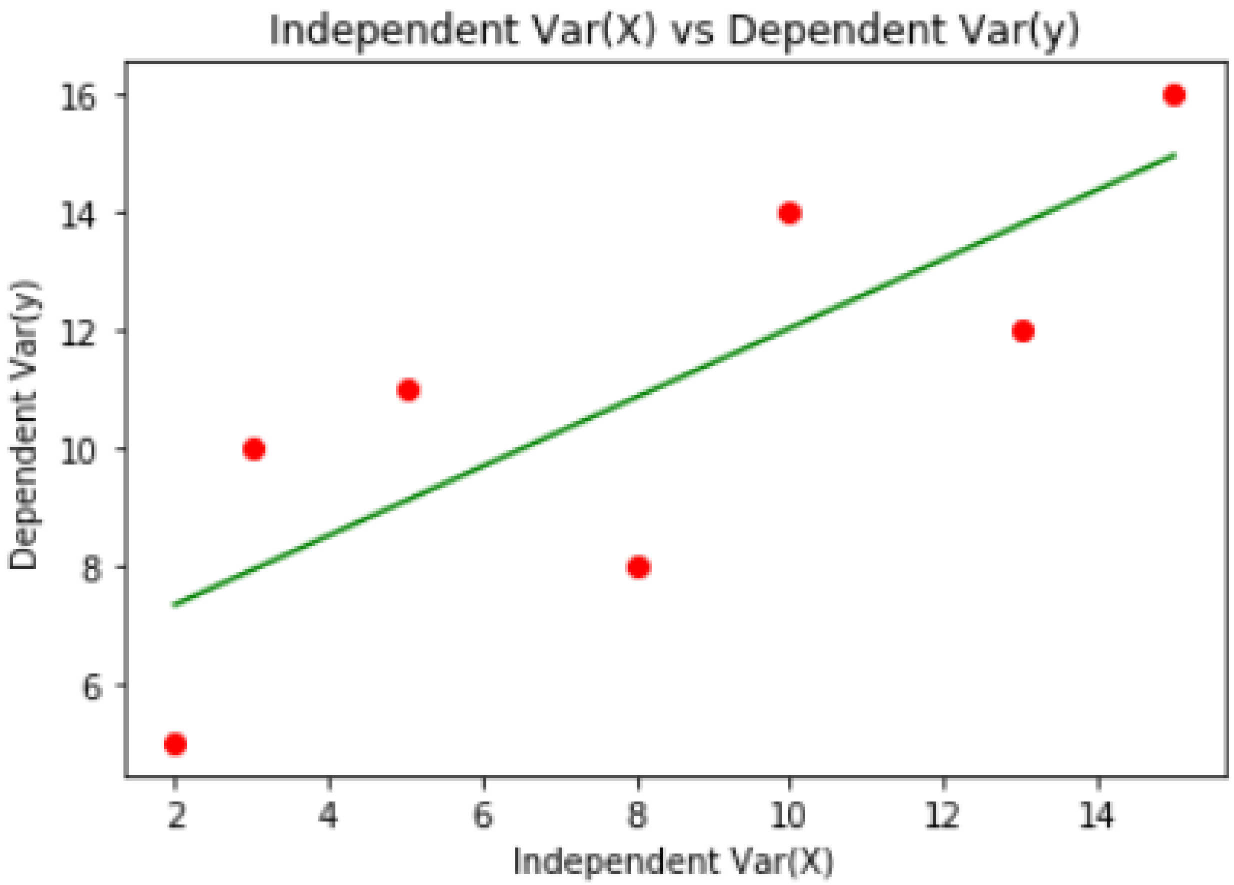
$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Repeat Until Convergence

{

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

}

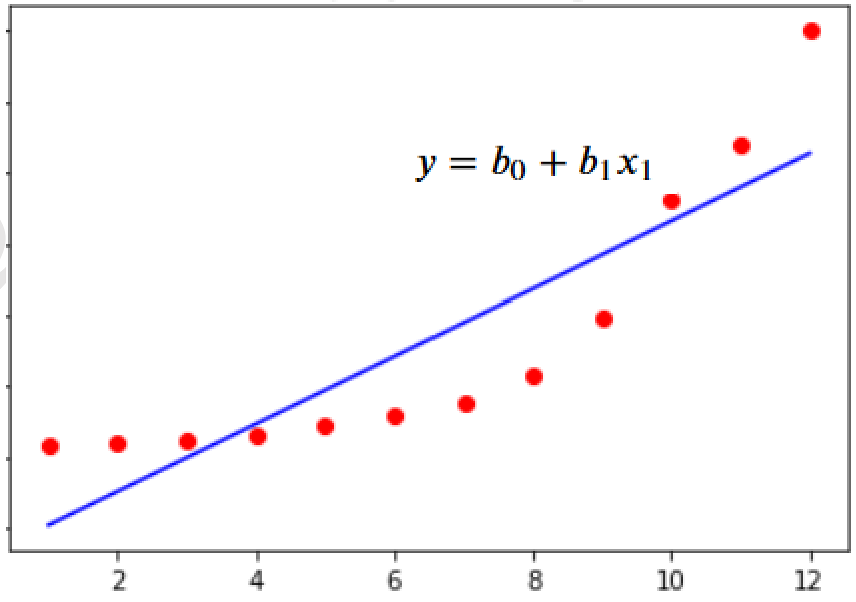

Independent Var(X) vs Dependent Var(y)
16
14
12
10
8
6
2
4
6
8
10
12
14
Independent Var(X)
Dependent Var(y)

Multiple Linear Regression

- Equation: $y = b_0 + b_1 x_1 + b_2 x_2 \ldots + b_n x_n$
- x1,x2,..xn = independent variable
- y = dependent variable

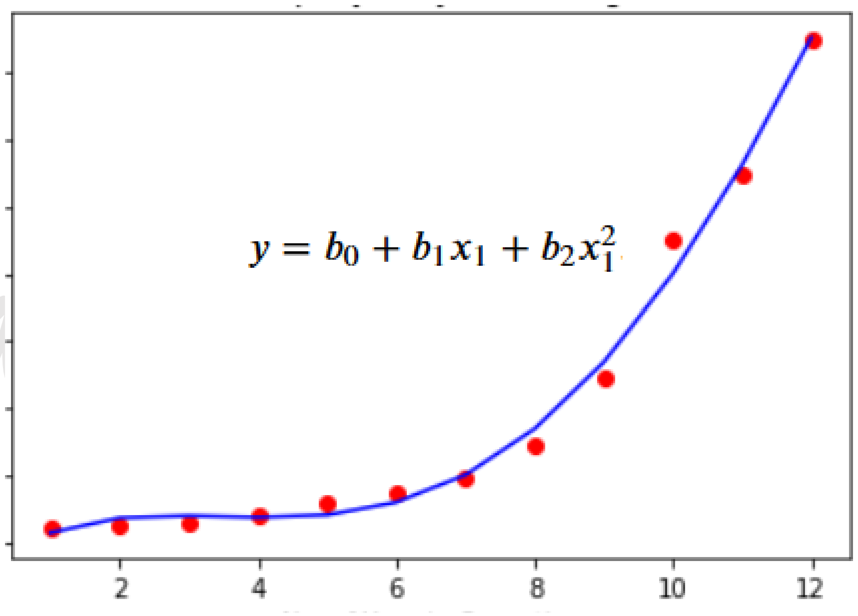
Polynomial Linear Regression

- y = dependent variable



Polynomial Linear Regression Cont….

- If data is not linear, we need polynomial terms to fit it better.



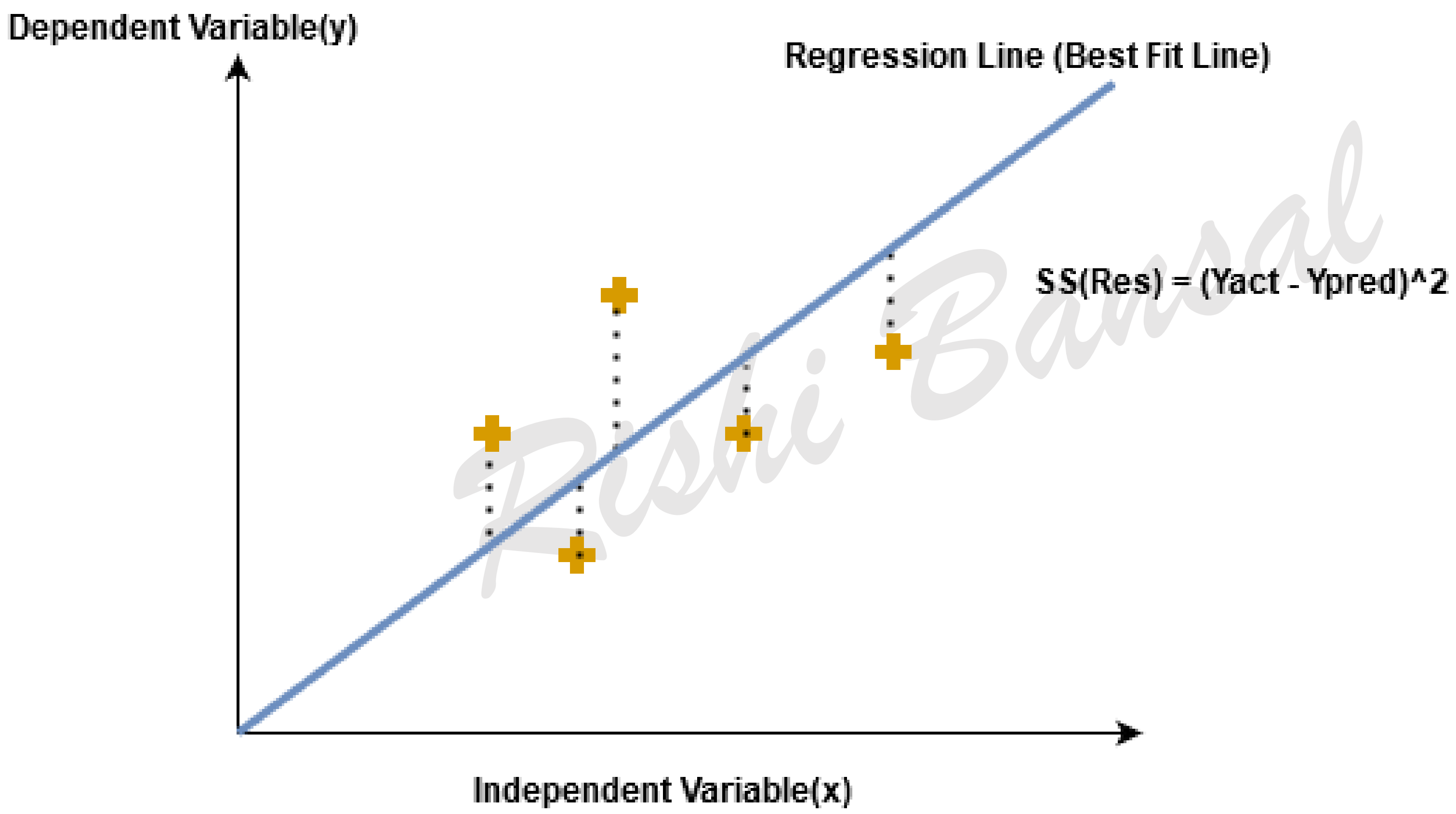
- Equation: $y = b_0 + b_1x_1 + b_2x_1^2 .... + b_nx_1^n$

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

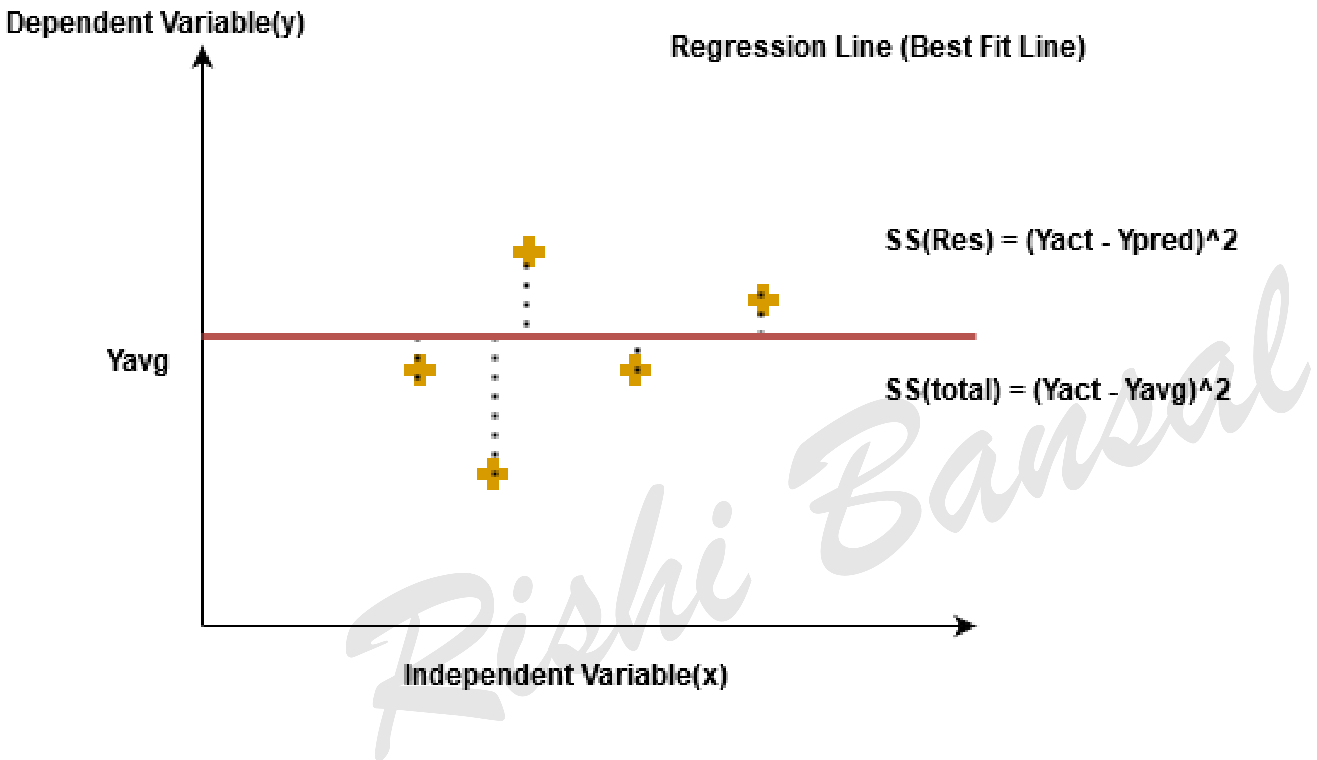
Regression Model Performance(r^2)

- It tells how well regression equation explains the data.



Regression Model Performance

- R^2



$$r^2 = 1 - \frac{Sum of Squares of Errors(SSE)}{Total Sum of Squares(SST)}$$

Regression Model Performance

- A value of $R^2 = 1$ means regression predictions perfectly fit/explains the data.

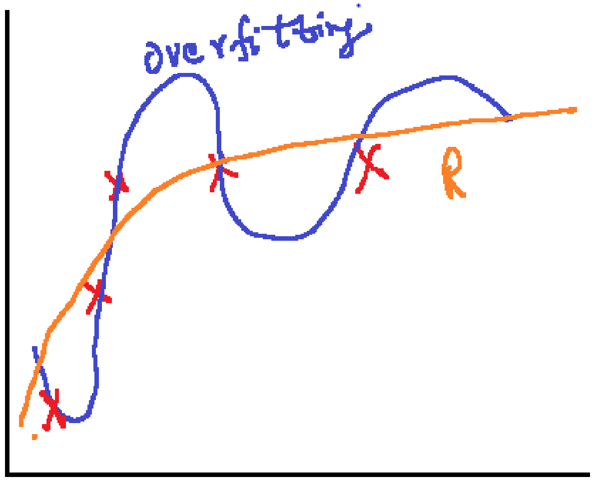
Question: Can $r2$ be negative?

- Ans: When: (Sum of Square Errors(SSE) > {Total Sum of Squares(SST)})
- This means when our predicted model performs worst than average line which is a very rare case.

$$r^2 = 1 - \frac{Sum of Squares of Errors(SSE)}{Total Sum of Squares(SST)}$$

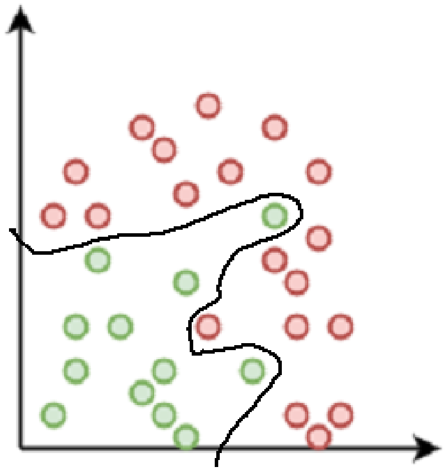
Overfitting

- Complex Decision Boundry
- Good fit of training data
- Poor fit of test Data



Fixing Overfitting

- Regularization Hyperparameter
- Cross Validation
- Bias – Variance trade off



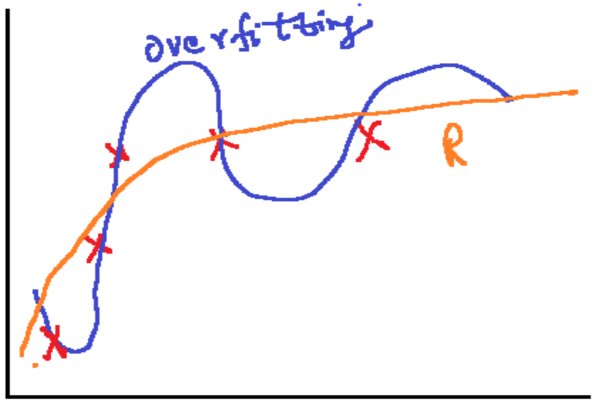
Underfitting

Underfitting Happens when:

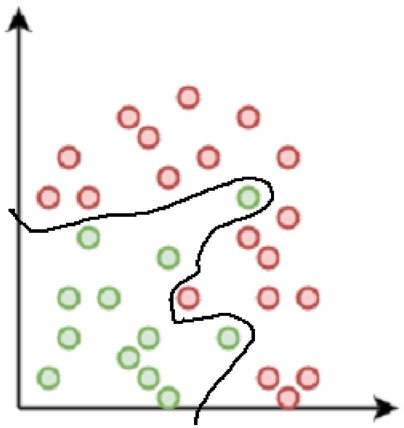
- Less amount of data to train
- Try to build linear model on non-linear data

Overfitting Happens when:

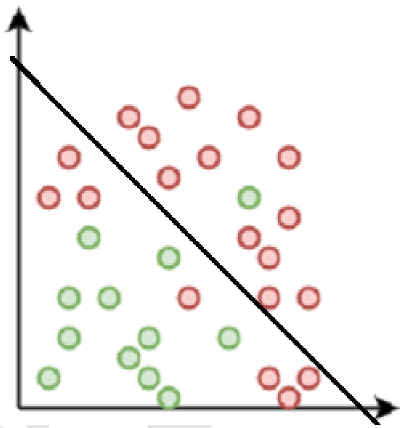
- Our model captures noise along with data's underlying pattern



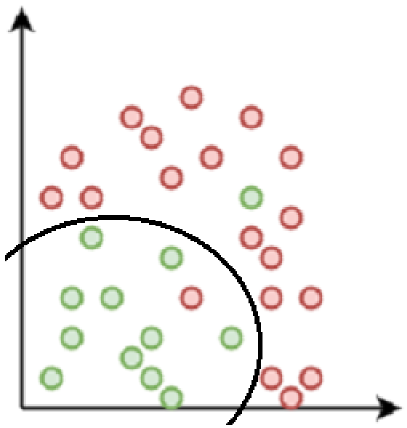
Test Your Understanding



- Overfitting
- High Variance



- Underfitting
- High Bias



- Good Model

Bias & Variance

- **Bias**: Difference between average prediction and the correct value which we are trying to predict
- **Variance**: Its the change in the amount of estimate of the target function on changing the training dataset.

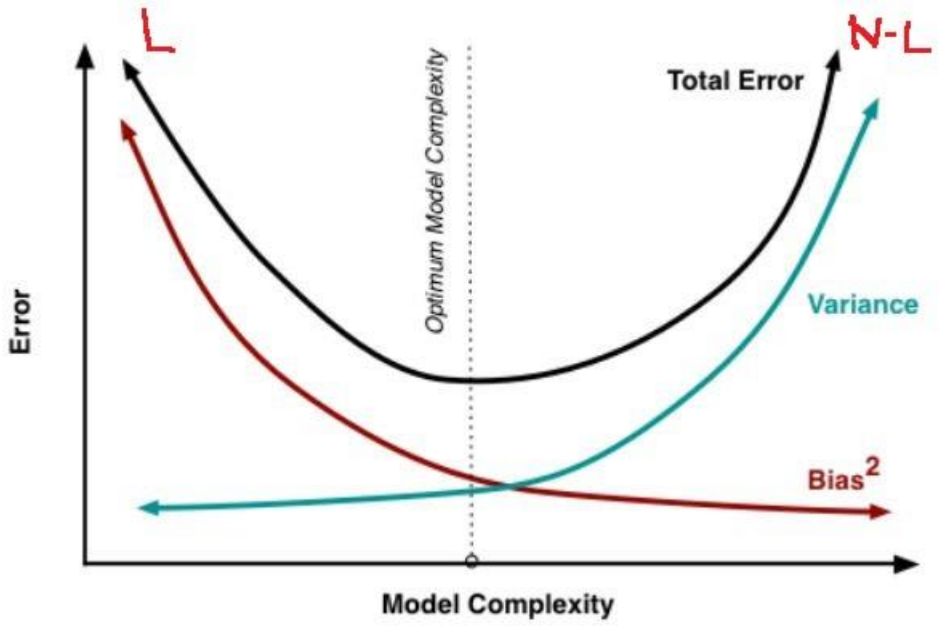
- **High Bias** means model pays very little attention to training data and oversimplifies the model (Underfitting)
- **High Variance**: large change in estimate of the target function on changing training dataset (Overfitting)

- **Low Bias, High Variance**: Decision Trees, Simple Vector Machine and k-Nearest Neighbors
- **High Bias, Low Variance**: Linear Regression, Linear Discriminant Analysis and Logistic Regression

Bias Variance Trade off

- Goal of Supervised Algorithm is have low bias and low variance
- Linear Models - High Bias and Low Variance
- Non Linear Models - Low Bias and High Variance
- Increasing the bias -> decrease the variance
- Increasing the variance -> decrease the bias

- Total Error = bias^2 + variance + Irreducible error



- Parameterization used to balance bias and variance

Regularisation

- Suppose we have a equation: $y = \theta_o + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
- Chances are there that above equation will overfit the training data.
- Intuition:
  If we penalize and make θ_3, θ_4 very small than effectively above equation will behave like $y = \theta_o + \theta_1 x + \theta_2 x^2$ without removing higher polynomial terms.

Cost Function: $J(\theta) = \frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)} + \lambda \sum_{i=1}^{n} \theta_j^2]$

λ is the regularisation parameter. It makes θ reduce after every iteration.

Regularisation

- Keep all the features but reduce the magnitude/values of parameter θ.
- Works well when we have a lot of features, each of which contributes a bit to predicting y.
- Penalize Complex models
- Reduces variance error but increases bias
- E.g: Lasso, Ridge (Modeling Techniques)