

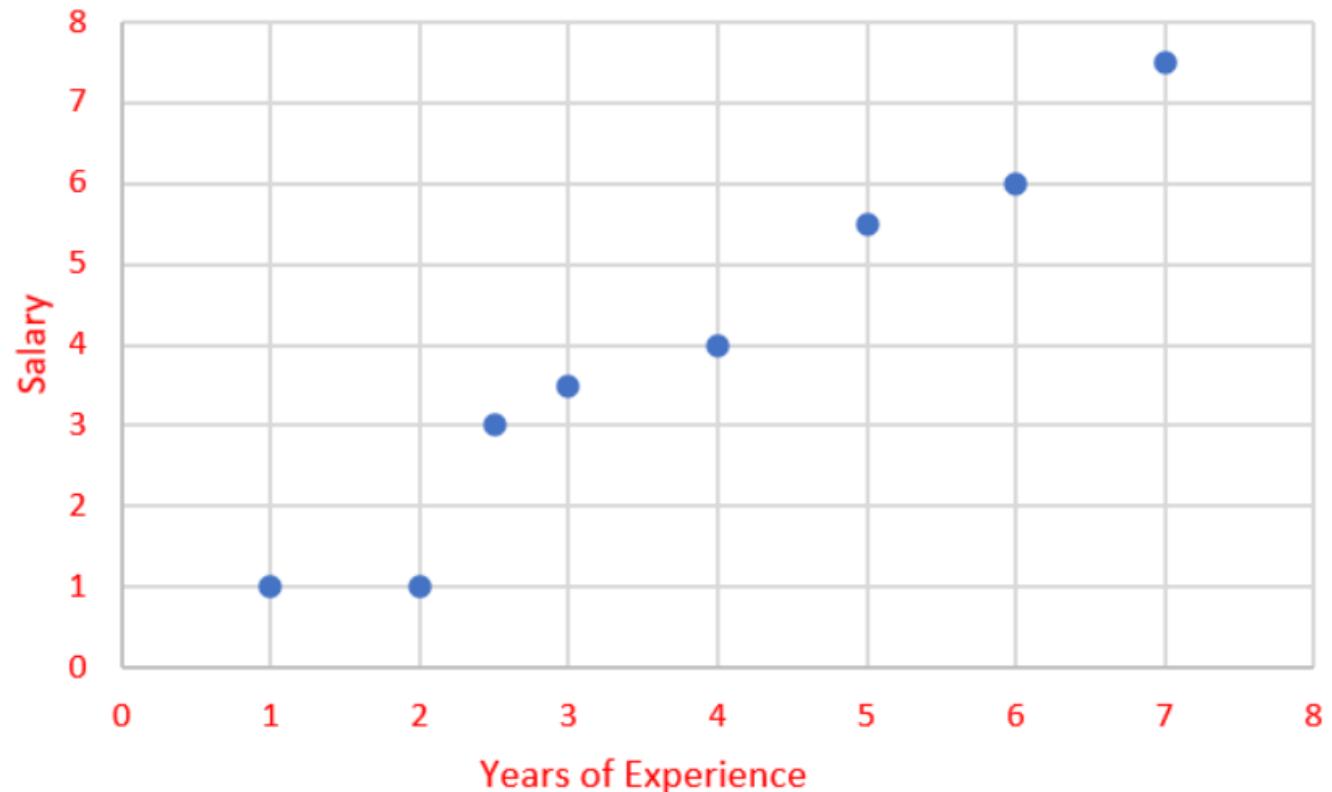
Regression

What is Regression in Terms of ML?

Let us consider we have a data of 150 employee's years of experience and their salaries.

Write a model which can help us to predict the salary of an employee based on his years of experience by learning from the data that we have in hand (150 employee's data).

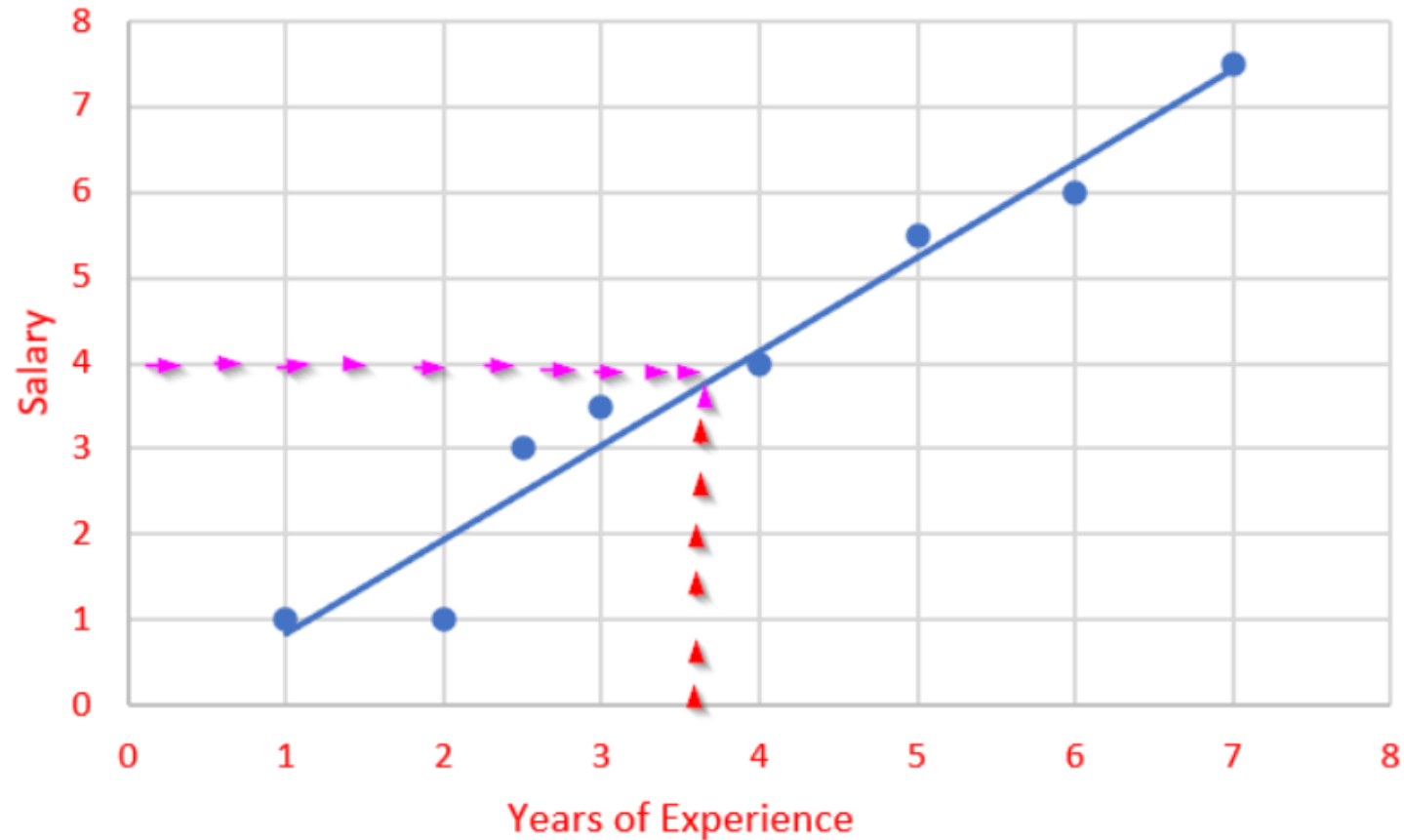
There are 2 variables —
Years of experience
Salary



There is a trend in data.

When Years of Experience increase, the Salary also increased.

By using this trend, we can draw a trend-line in the graph as below.



Observations:

Now we know the trend of the data.

We can somewhat mind map and predict what will be the salary for a particular experience.

Consider that now we want **to predict the salary of a new employee, whose experience is 3.5.**

Note : Drawn a red dotted and pink dotted lines in the above graph which mentions that we can predict the salary for 3.5 years using the trend line. 4!

This way of prediction falls under **Regression**.

Regression is a problem of predict the output values based on input values by using the historical data.

How the prediction happened?

We had some advantages in the given data. What were they?

Advantage Number 1:

There was a trend between Experience and Salary. So, we made a trend line and predicted the Salary.

Even though all the points are not in trend line, we can be sure for some good confidence level.

This way of predicting output based on a straight-line of trend is called Linear Regression.

Linear Regression is a linear approach to model the relationship between a two or more variables by fitting a straight line i.e. linear, to predict the output for the given input data.

As the trend is a Straight line, we will use the equation of the straight line to find the Output (y) value.

Straight Line Equation: $y = mx + c$

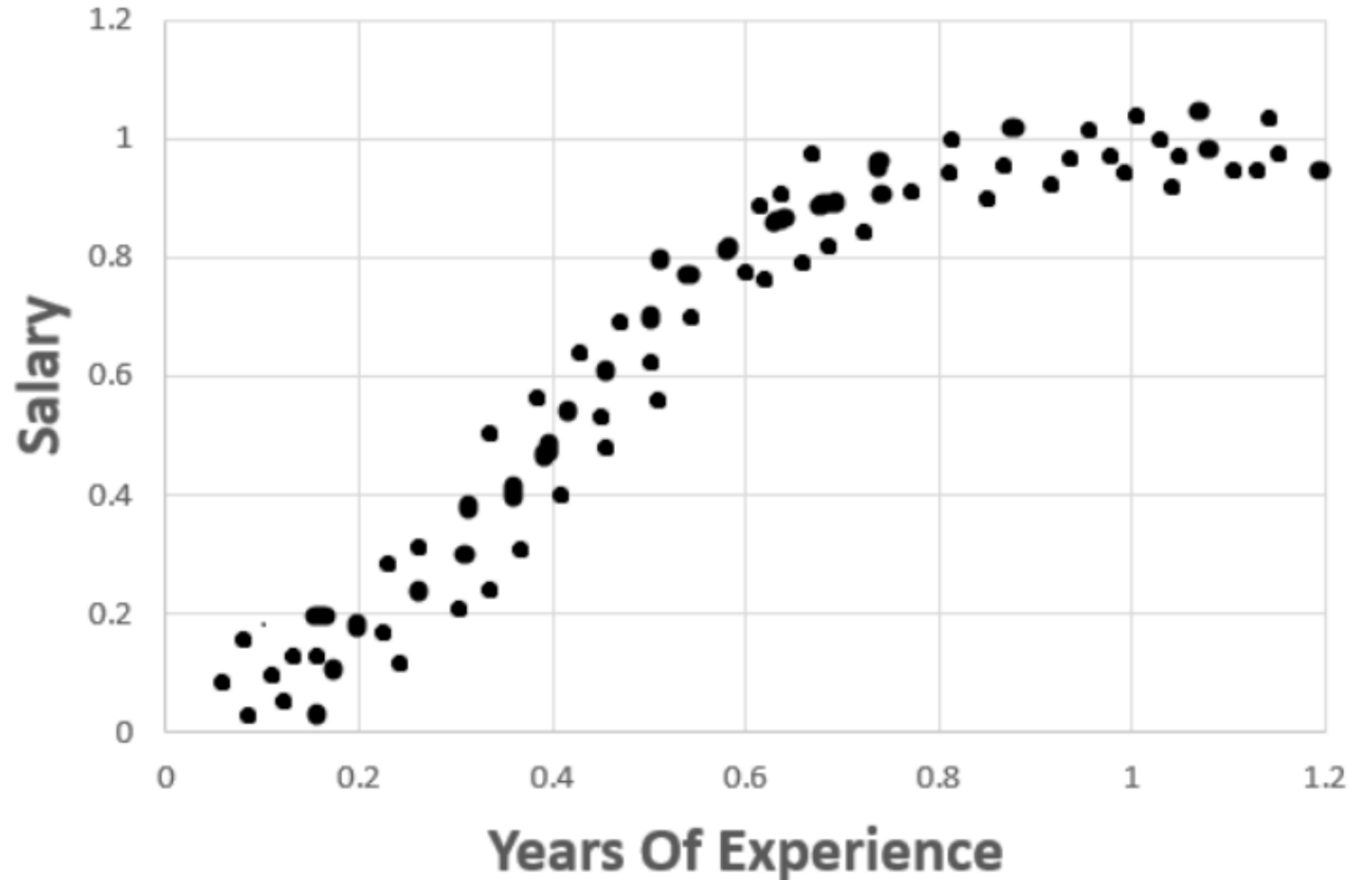
What if there is a different trend?

Consider the below graph.

In this data the trend is different and it seems, a curve fits very well than a line. But still there is a trend.

This Regression falls under **Non-Linear Regression**.

For different types of Non-Linear curves, the equations change.



Non-Linear Regression is a regression analysis in which the model is fitted by a function which is a nonlinear combination of the model parameters.

Advantage Number 2: Correlation with Causation

What if there is no trend?

- When there is no trend then that means there is no relationship between the two variables. So, we cannot predict the output based on input variables.
- This relationship can be determined by the Correlation Coefficient between two variables X & Y .
- For highest correlation coefficient, the output is more accurate.

What if we get a data which says good correlation coefficient between two variables but that cannot be true in real world?

What if we get a data which says good correlation coefficient between two variables but that cannot be true in real world?

Consider the problem that we need to predict the house price based on the age of the house owner. Also, unfortunately, the data says positive good correlation of 0.9.

But we know that this cannot be true.

In this case, if we make a model with these 2 variables, the result will be a failure.

We all know about the housing prices and it mostly determined by the size of the house, locality, facilities and age of the house etc. But why should people care about the age of the house owner?

Size of the house, locality, facilities available in the property will make some percentage of cause in the pricing.

We can apply Regression analysis only when there is a Correlation and Causation exist.

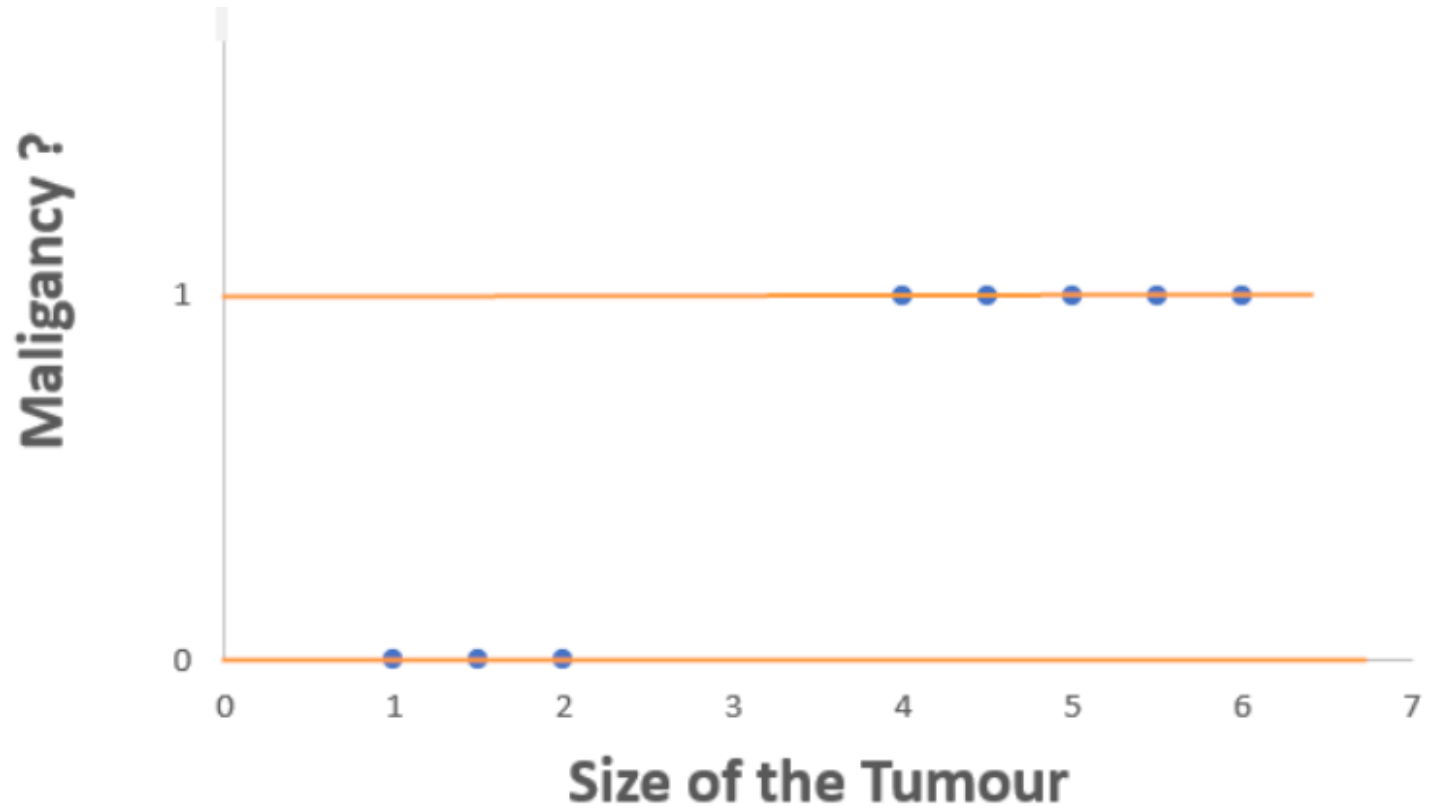
Advantage Number 3: Data was continuous.

Example: Salary is a continuous data.

Consider the we have a data of Tumor detected patients and the result of the examination is malignancy or not.

In this data, if we plot it in a scatter plot it will become like the above. Because if you see the dependent variable (or Y axis values), it is not continuous. **It is Discrete — values are either ‘Yes’ or ‘No’.**

With this data we cannot have a graph with a line or a curve. This type of analysis falls under **classification** as we just classify our output into one of two categories.



Classification is a problem of predicting output values which is **Discrete (Categorical)**.

Regression is a problem of predicting output values which is **Continuous (Numerical/Quantitative)**.

Conclusion:

Regression is a process of predicting the output values based on input values by using the historical data.

To perform Regression:

1. The data should be continuous.
2. The Input and Output variables should have Correlation as well as Causation.

Classification is a process of predicting output values which is Discrete (Categorical).

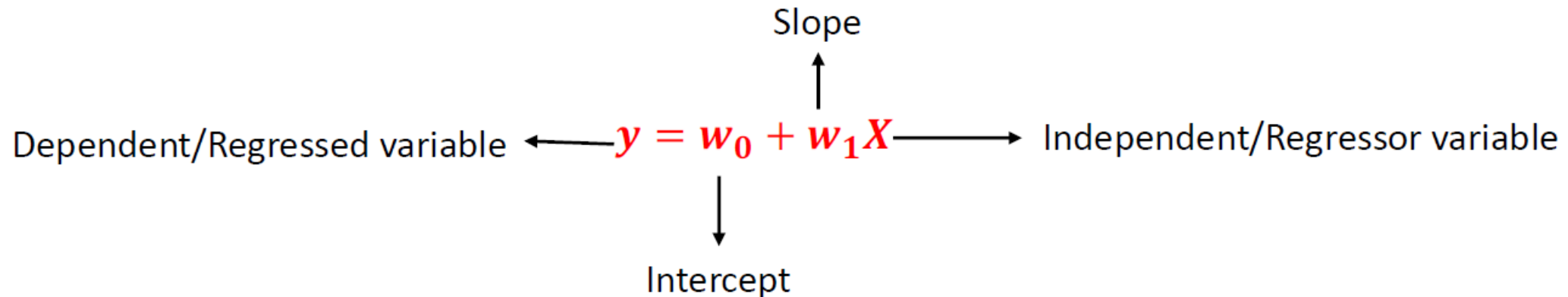
Linear Regression

- ❖ Linear regression is a way to identify a relationship between two or more variables and use these relationships to predict values of one variable for given value(s) of other variable(s).
- ❖ Linear regression assume the relationship between variables can be modelled through linear equation or an equation of line

$$y = w_0 + w_1 X$$

Diagram illustrating the components of the linear regression equation $y = w_0 + w_1 X$:

- Dependent/Regressed variable**: y
- Independent/Regressor variable**: X
- Slope**: w_1
- Intercept**: w_0



Multiple Regression

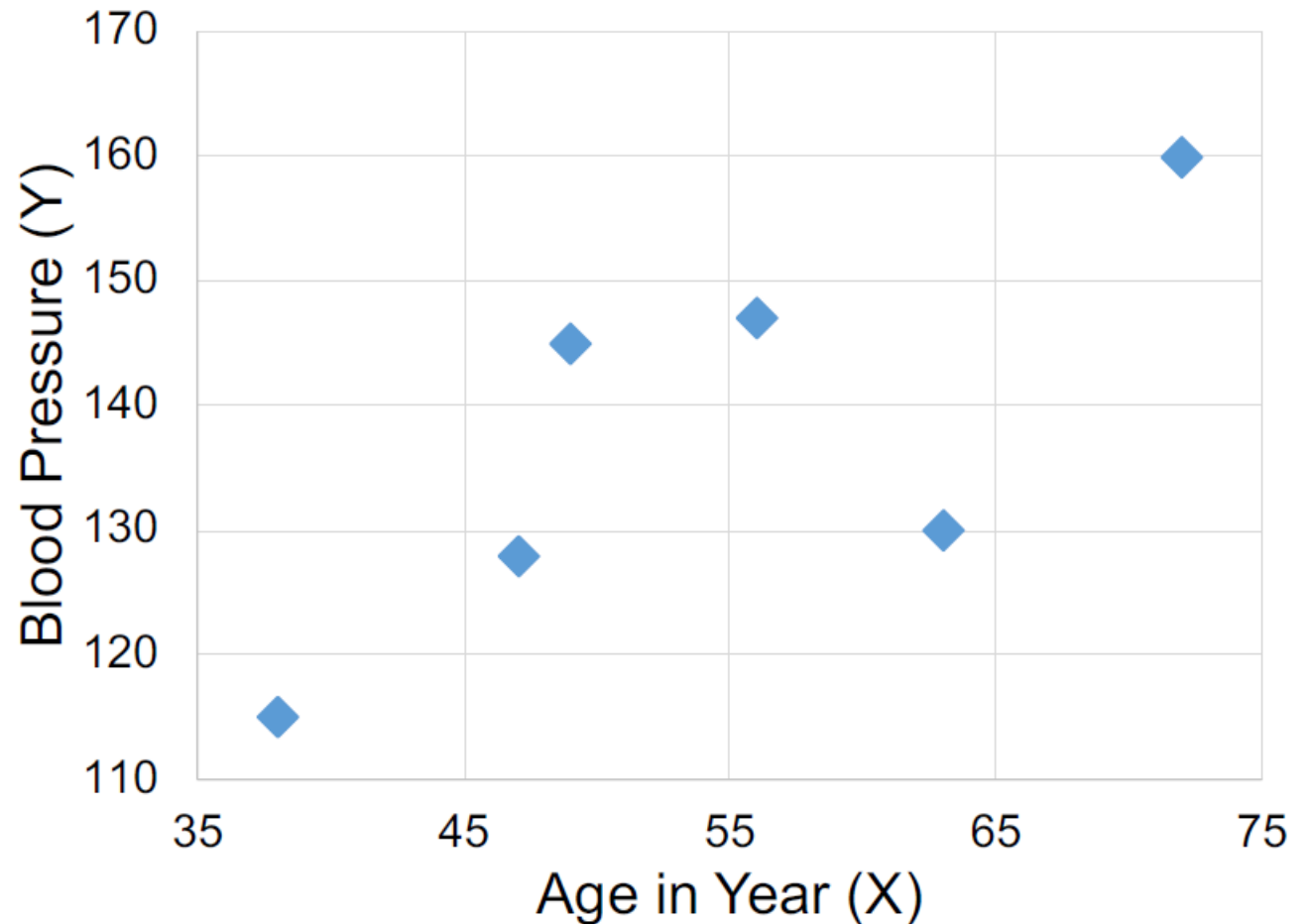
- ❖ Last slide showed the linear regression model with one independent and one dependent variable.
- ❖ In Real world a data point has various important attributes and they need to be catered to while developing a regression model. (Many independent variables and one dependent variable)

$$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3. \dots \dots \dots W_nX_n$$

Regression –Problem Formulation

Let you have given with a data:

Age in Years (X)	Blood Pressure (Y)
56	147
49	145
72	160
38	115
63	130
47	128



Linear Regression

❖ For given example the Linear Regression is modeled as:

$$\text{BloodPressure}(y) = w_0 + w_1 \text{AgeinYear}(X)$$

OR

$$y = w_0 + w_1 X - \text{Equation of line}$$

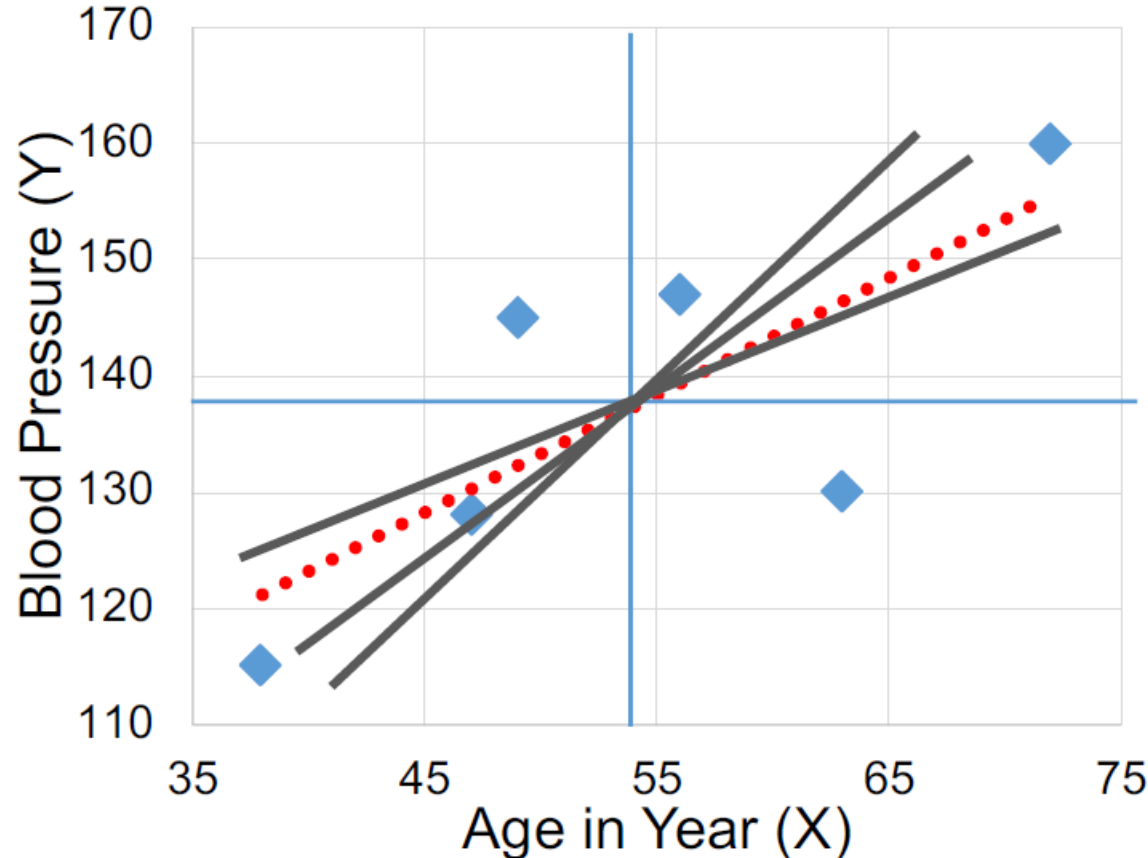
with w_0 is intercept on Y -axis and w_1 is slope of line

Blood Pressure - Dependent Variable

Age in Year - Independent Variable

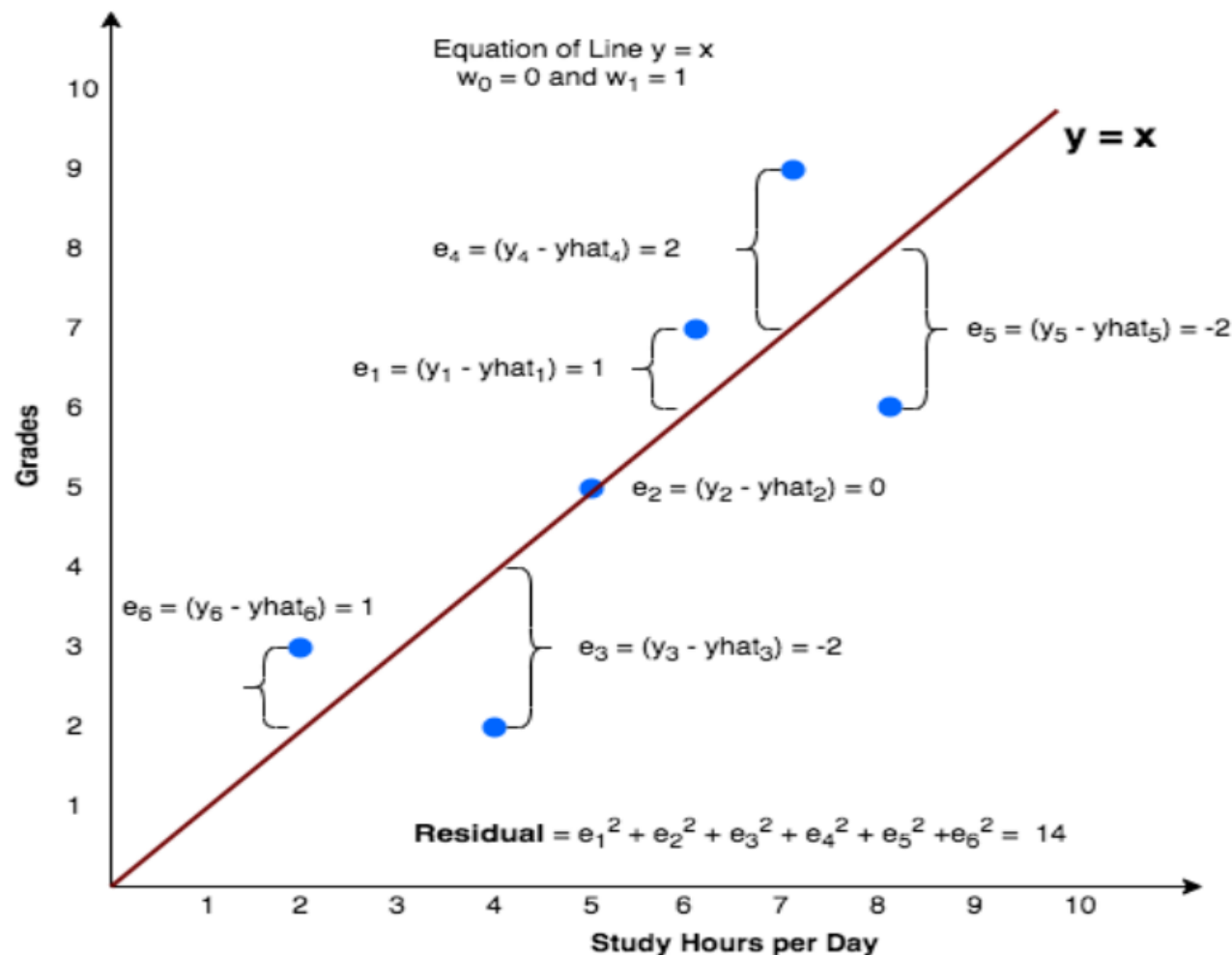
Linear Regression- Best Fit Line

- ❖ Regression uses line to show the trend of distribution.
- ❖ There can be many lines that try to fit the data points in scatter diagram
- ❖ The aim is to find **Best fit** Line



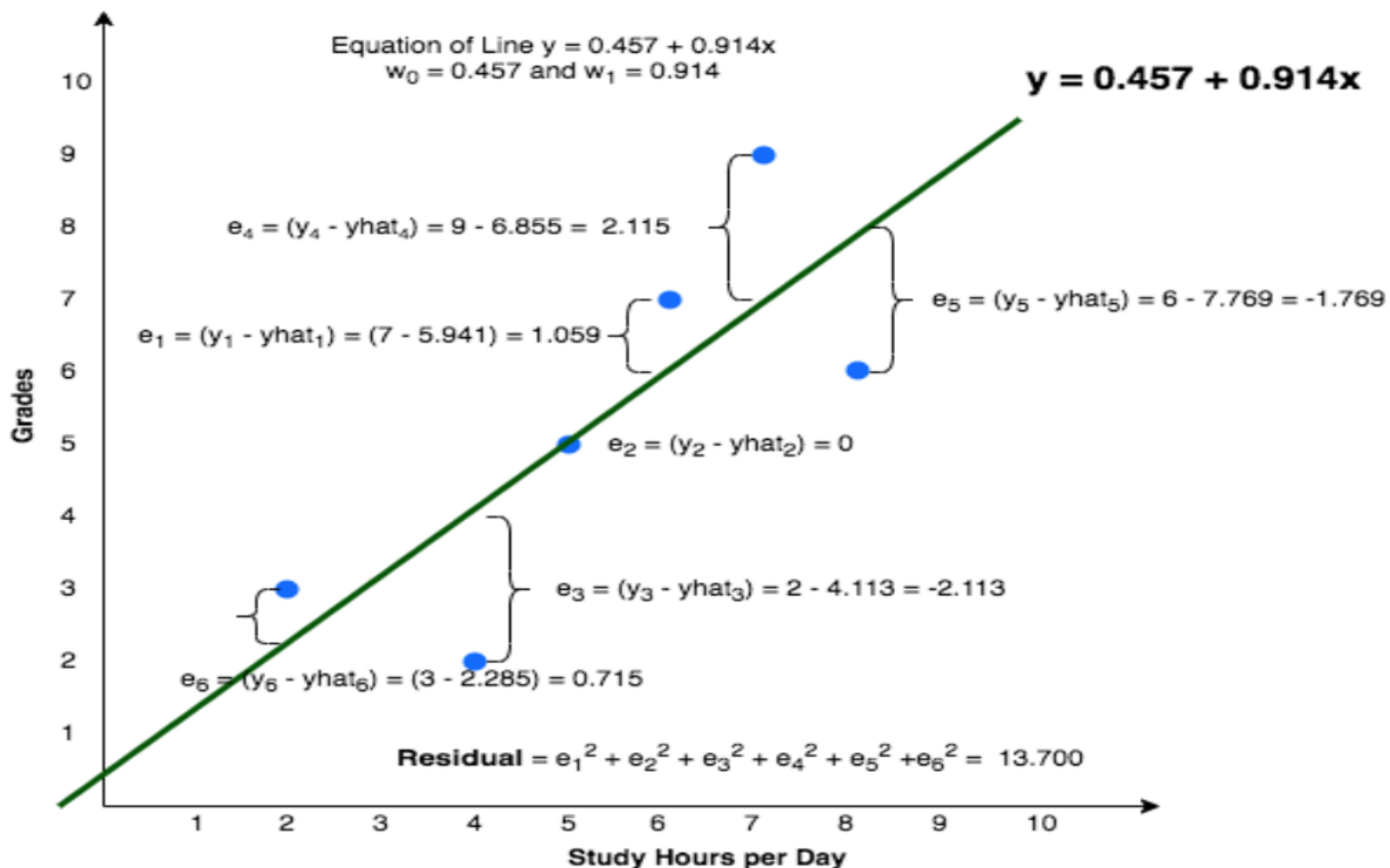
What is Best Fit Line

- ❖ Best fit line tries to explain the variance in given data. (minimize the total residual/error)



What is Best Fit Line

- ❖ Best fit line tries to explain the variance in given data. (minimize the total residual/error)



Linear Regression- Methods to Get Best

❖ Least Square

❖ Gradient Descent

Gradient descent is an optimization algorithm used to minimize a cost function (i.e. Error) parameterized by a model.

Gradient means the slope of a surface or a line. This algorithm involves calculations with slope.

To understand about Gradient Descent, we must know what is a cost function.

Cost function(J) of Linear Regression is the Root Mean Squared Error (RMSE) between predicted y value (predicted) and true y value (y).

$$\text{minimize } \frac{1}{n} \sum_{i=1}^n (\text{pred}_i - y_i)^2$$

$$J = \frac{1}{n} \sum_{i=1}^n (\text{pred}_i - y_i)^2$$

For a Linear Regression model, our ultimate goal is to get the minimum value of a Cost Function.

Linear Regression- Gradient Descent

Model: $Y = w_0 + w_1X$

Task: Estimate *the value of* w_0 and w_1

Define the cost function,

$$cost(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Objective of gradient Descent

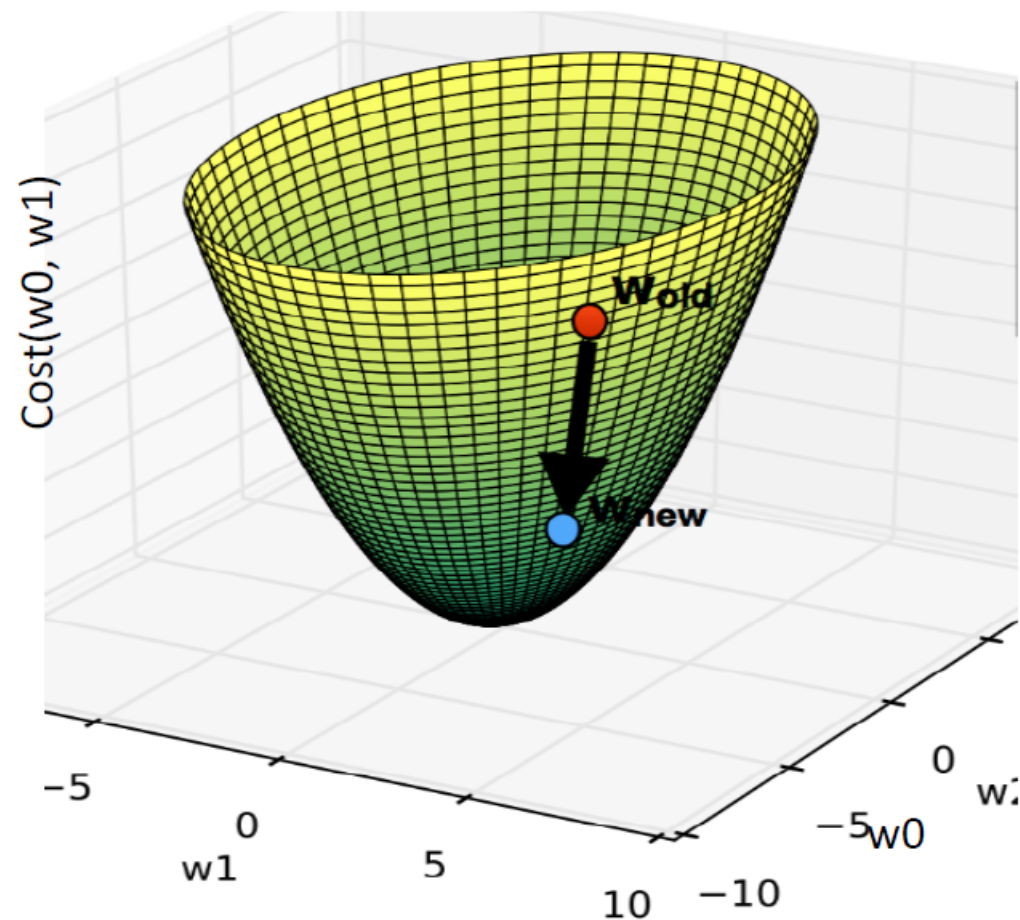
$$\min_{w_0, w_1} cost(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (y_i - (w_0 + w_1 x_i))^2$$

Linear Regression- Gradient Descent

Model: $Y = w_0 + w_1X$

Task: Estimate *the value of* w_0 and w_1
the objective,

$$\min_{w_0, w_1} \text{cost}(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (y_i - (w_0 + w_1 x_i))^2$$



Linear Regression- Gradient Descent

❖ Gradient descent works if following steps:

1. Initialize the parameters to some random variable
2. Calculate the gradient of cost function w. r. t. to parameters
3. Update the parameters using gradient in opposite direction.
4. Repeat step-2 and step-3 for some number of times or till it reaches to minimum cost value.

Gradient Descent is an iterative algorithm, that fit various lines to find the best fit line iteratively.

Each time we get an error value Sum of Squares Error (SSE).

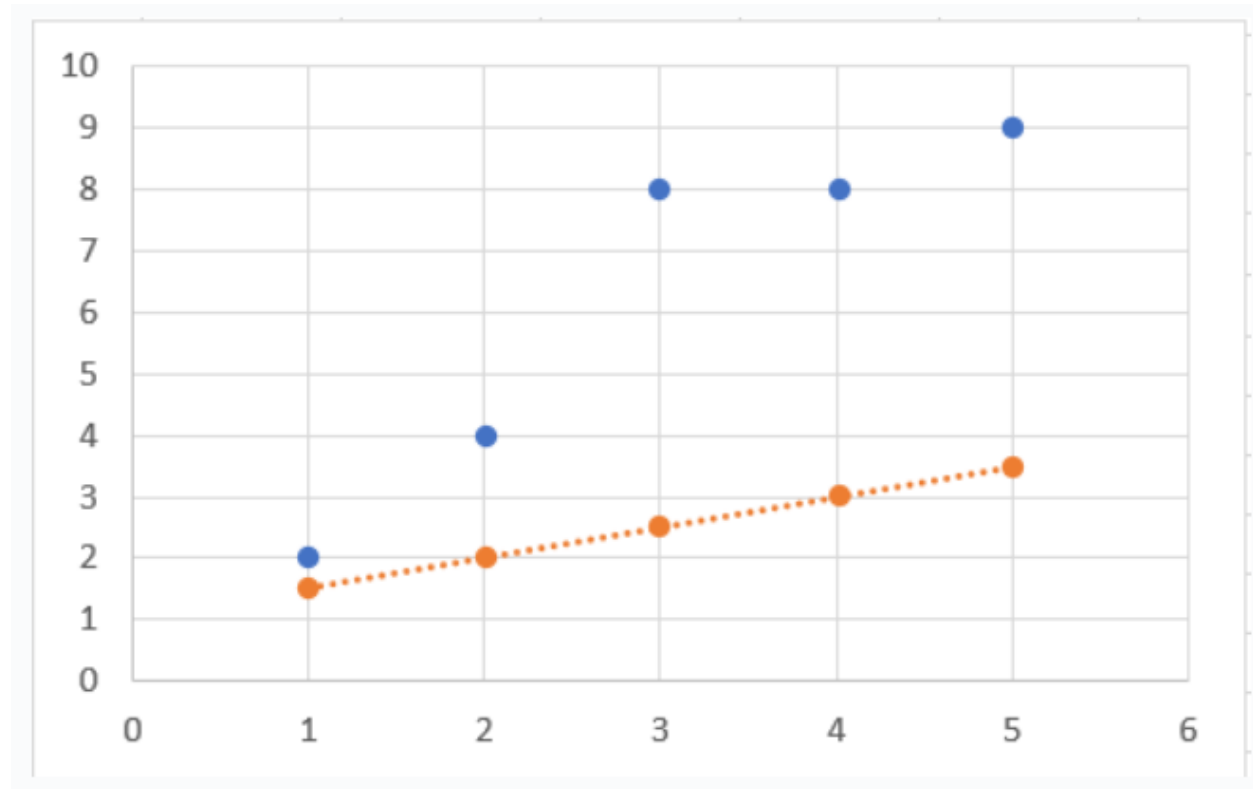
If we fit all the error values in a graph, it will become a parabola.

What is the Relationship between Slope, Intercept and SSE? Why Gradient Descent is a Parabola?

For $m = 0.45$ $b = 0.75$; line $Y_{\text{Predicted}} = 0.45x + 0.75$				
X	Y	$Y_P = 0.45x + 0.75$	$(Y - Y_P)^2$	$SSE = 1/2(Y - Y_P)^2$
1	2	1.2	0.64	0.04
2	4	1.65	5.5225	0.34515625
3	8	2.1	34.81	2.175625
4	8	2.55	29.7025	1.85640625
5	9	3	36	2.25
			Total SSE	6.6671875

Let us take random value of $m = 0.45$ & $c = 0.75$.

For this slope and intercept we have come up with a new Y predicted values for a regression line.



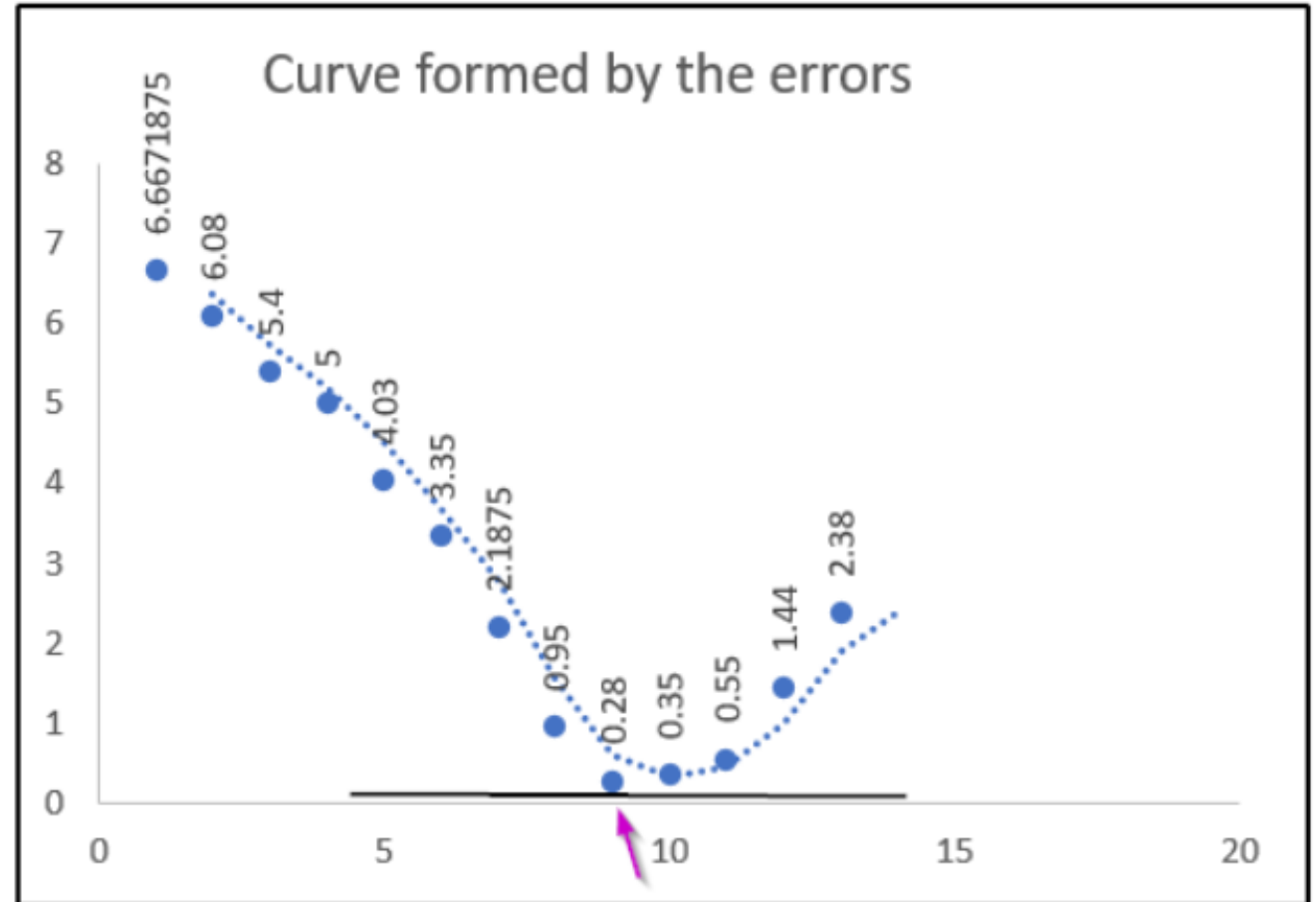
In the above graph, blue color dots are Original Y values & Orange color dots are Y predicted values that we just found with $m = 0.45$ & $c = 0.75$.

We can clearly say that this line is not a best fit line for this data.

The total SSE for this iteration is 6.66 approximately.

Then let's do the iterations by decreasing the m & c values randomly and the results are given below:

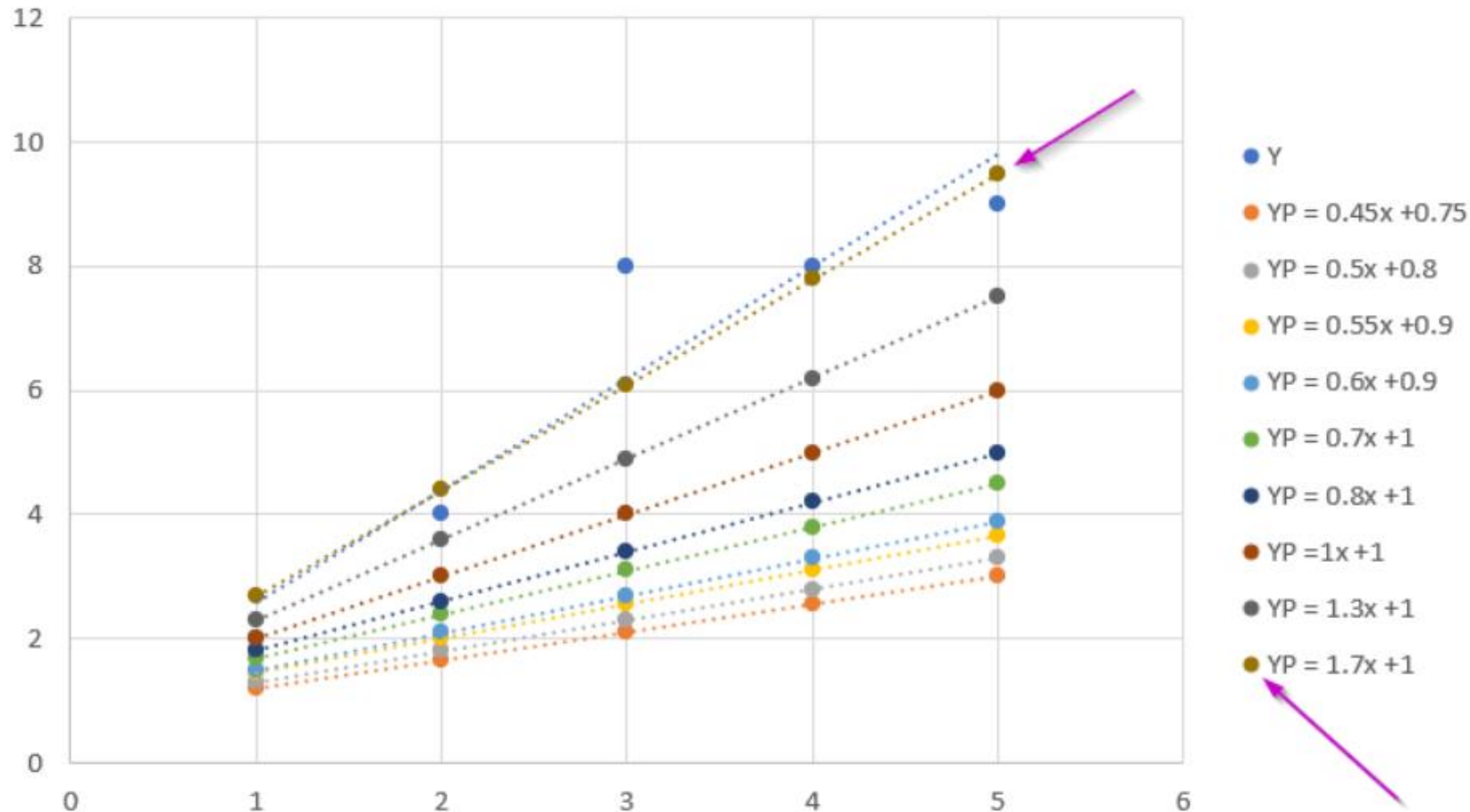
m	c	Error	Line Equation
0.45	0.75	6.6671875	YP = 0.45x + 0.
0.5	0.8	6.08	YP = 0.5x + 0.8
0.55	0.9	5.4	YP = 0.55x + 0.
0.6	0.9	5	YP = 0.6x + 0.9
0.7	1	4.03	YP = 0.7x + 1
0.8	1	3.35	YP = 0.8x + 1
1	1	2.1875	YP = 1x + 1
1.3	1	0.95	YP = 1.3x + 1
1.7	1	0.28	YP = 1.7x + 1
1.9	1	0.35	YP = 1.9x + 1
2	1.1	0.55	YP = 2x + 1.1
2.3	1.1	1.44	YP = 2.3x + 1.1
2.5	1.1	2.38	YP = 2.5x + 1.1



If we plot a graph for all the error values alone then that will look like a curve as shown in the above graph

In the above table, we can see that for different values of m & c the error value decreases gradually and from a point it starts increasing again.

Plot all the lines that formed with above m & c values in our original graph



We can see pink marked line is formed with m as 1.7 & c as 1 which produces the minimum error of all other m & c values in the m, c & Error table.

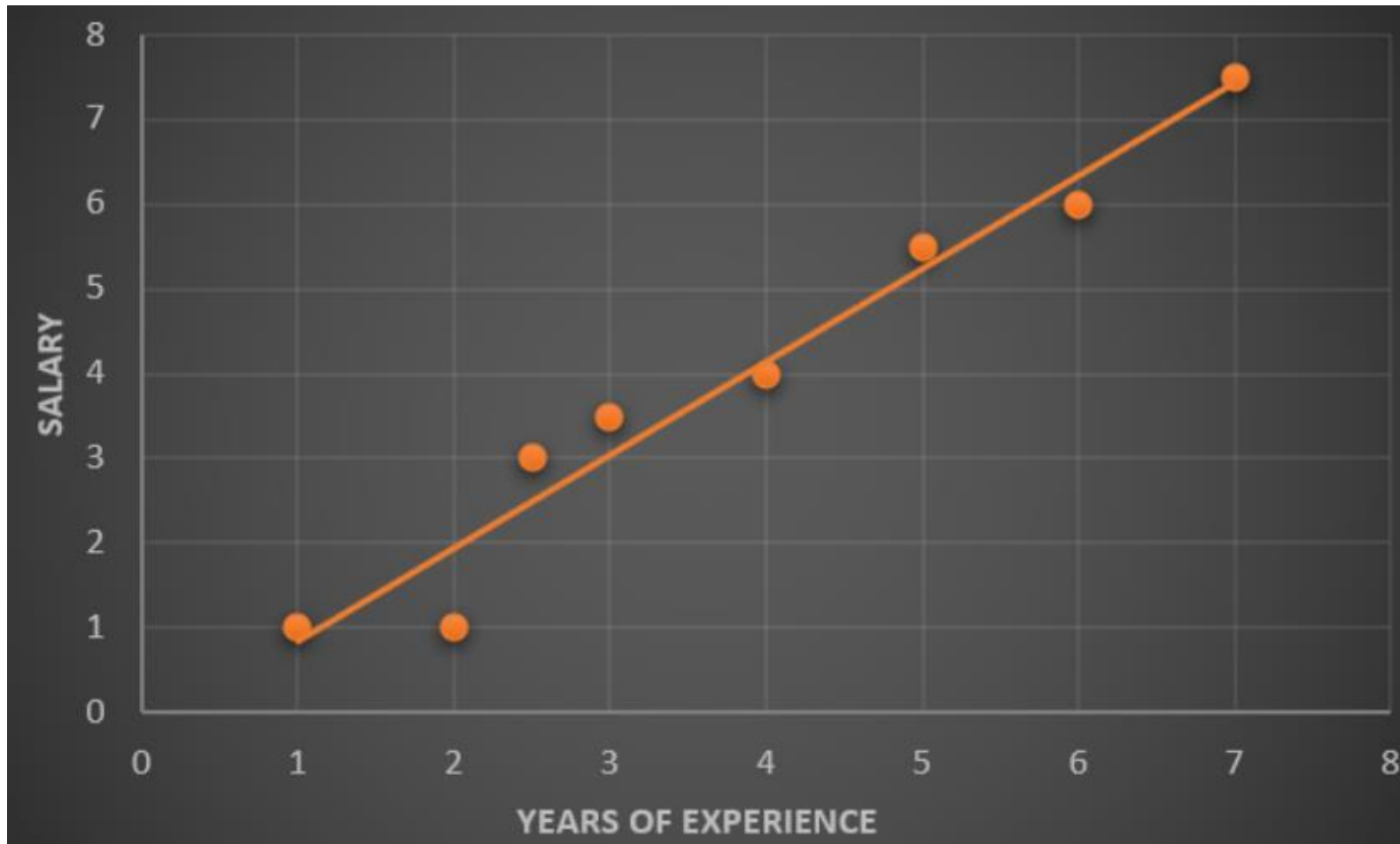
The blue color line is the best fit line produced by excel itself. Our line **$YP = 1.7x + 1$** is closest to the best fit line.

If we go into the iterations seriously with minimum change in m/c values, we will be hitting the best fit line.

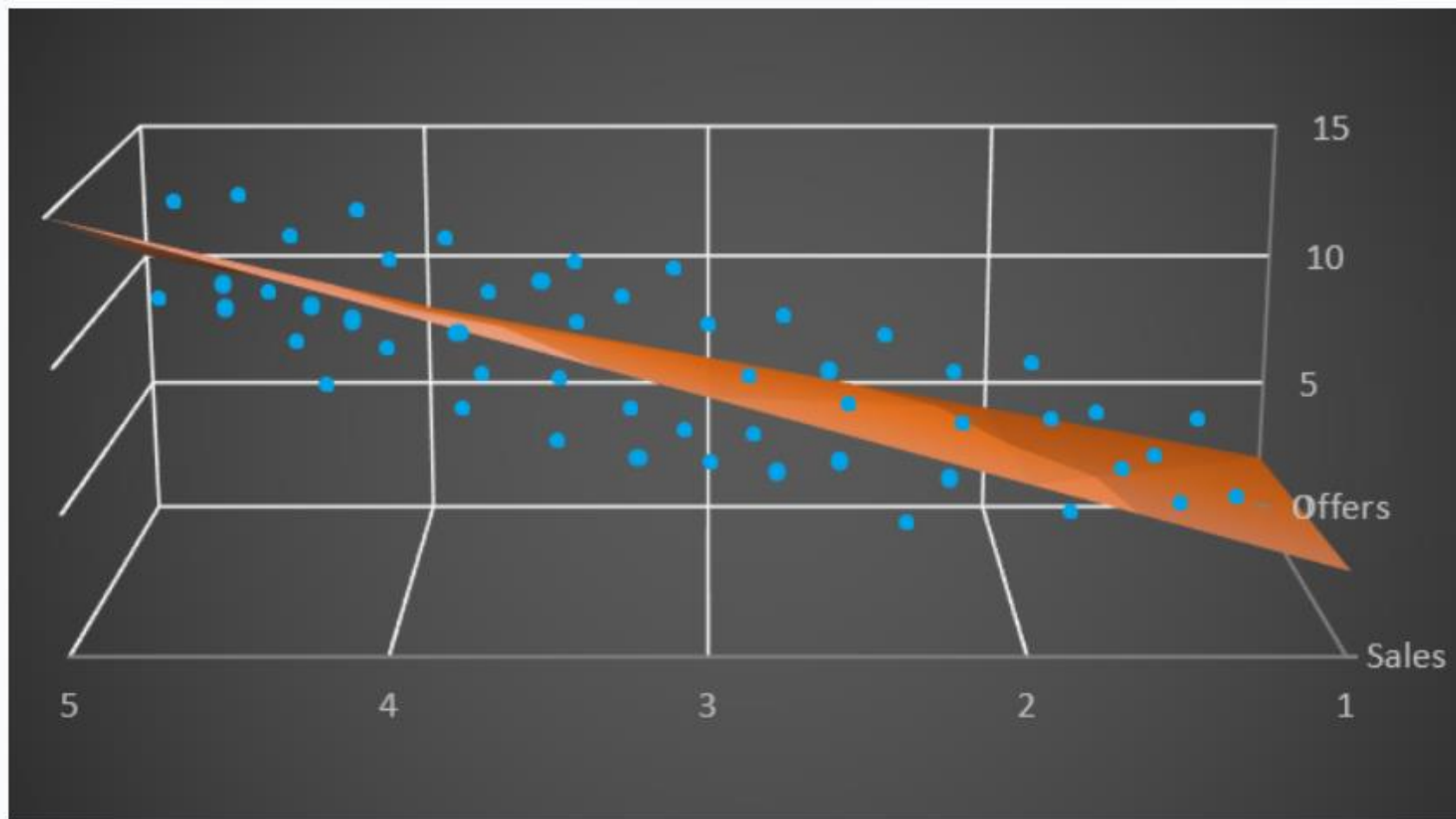
In the parabola graph, we can identify that the error 0.28 is the minimum and it was produced by the line equation **$YP = 1.7x + 1$** .

More than 1 Independent Variables:

Where there are only one Independent Variable X, the Linear Regression graph will be 2-Dimensional and the best fit will be a line.

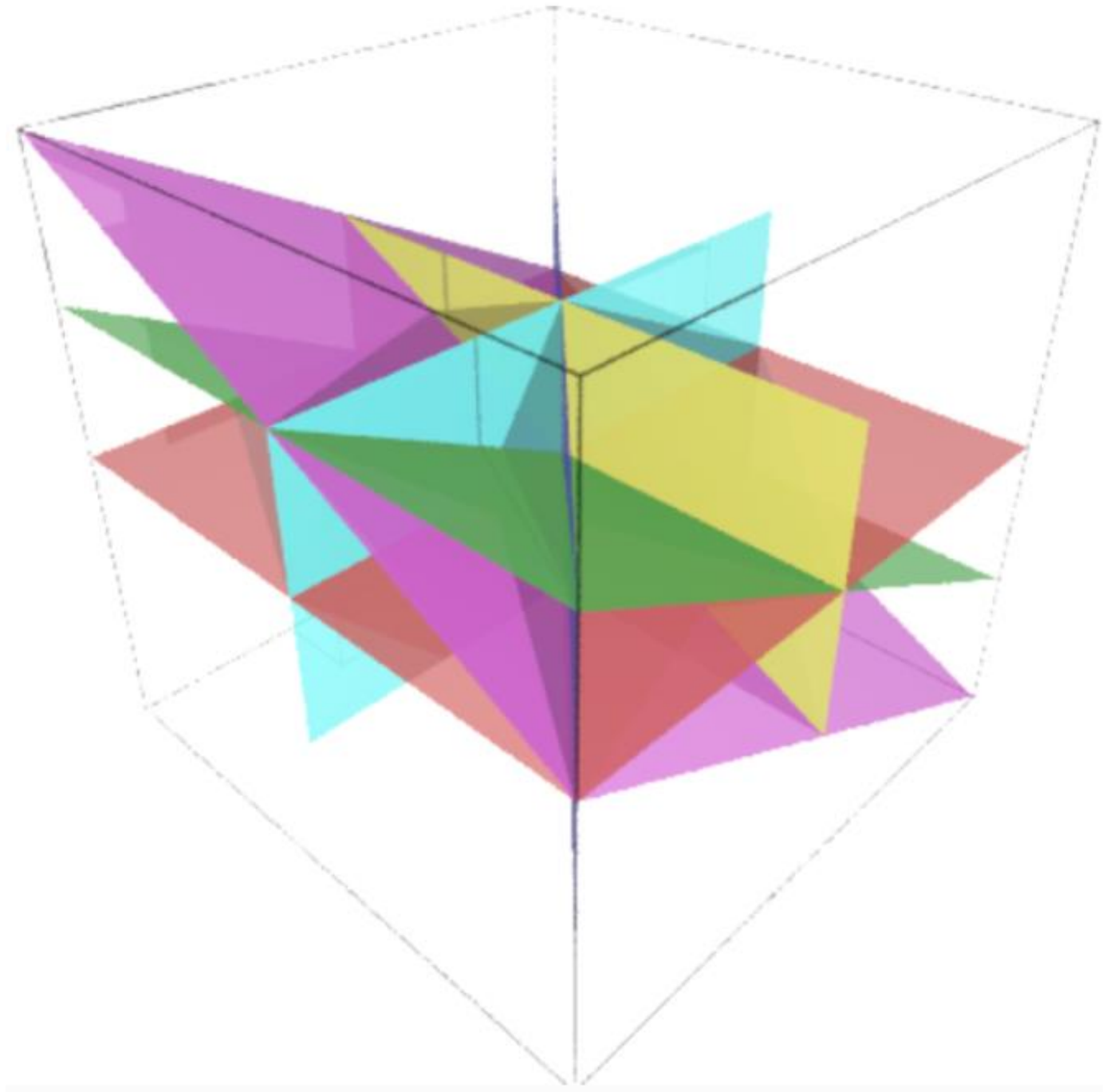


When there are 2 independent variables, the 2-D area becomes a surface/space of 3 Dimensional and the best fit become a Plane.

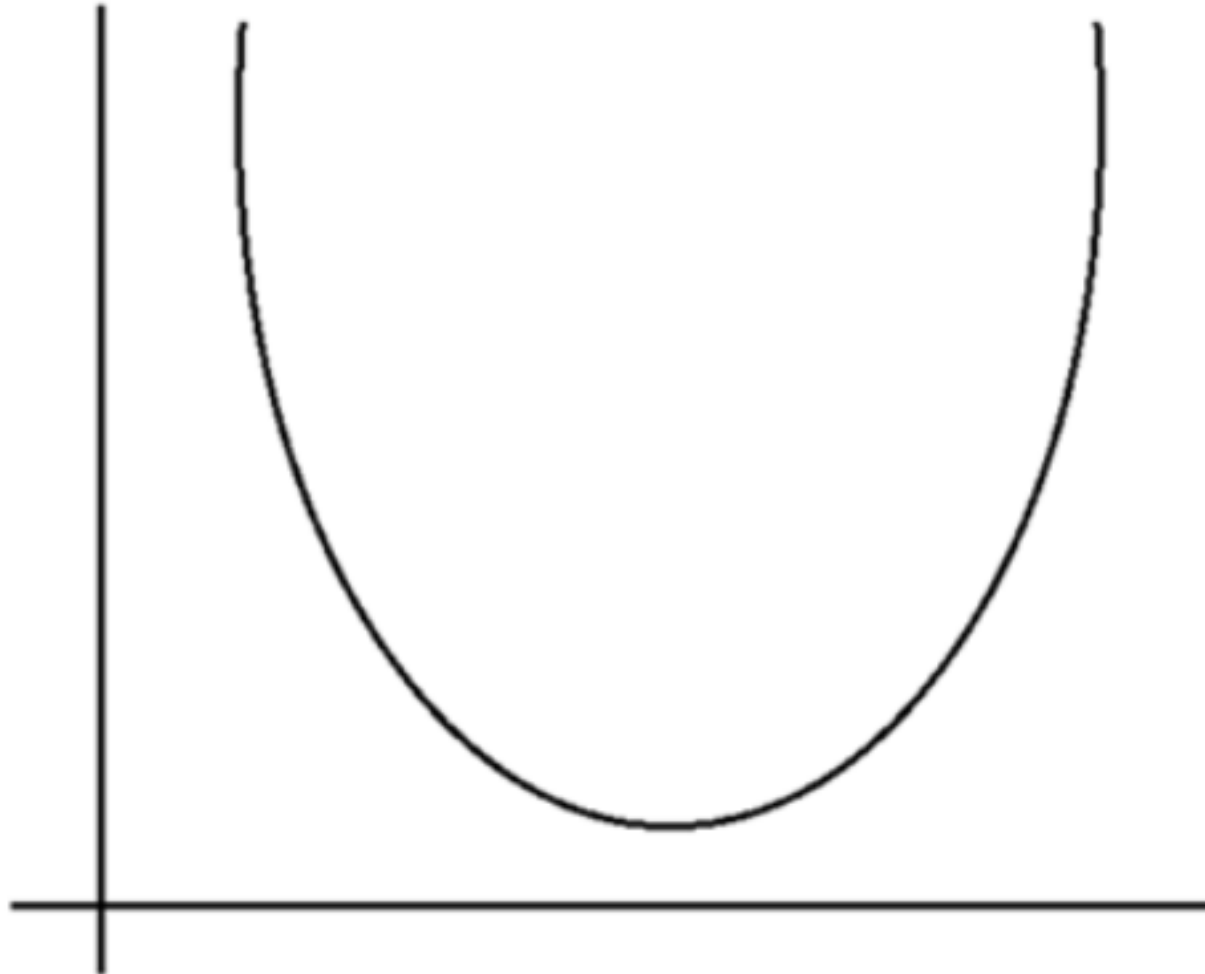


When n variables ($n > 2$) exists, then the n -Dimension area becomes a n -Dimensional space and the best fit becomes a Hyperplane.

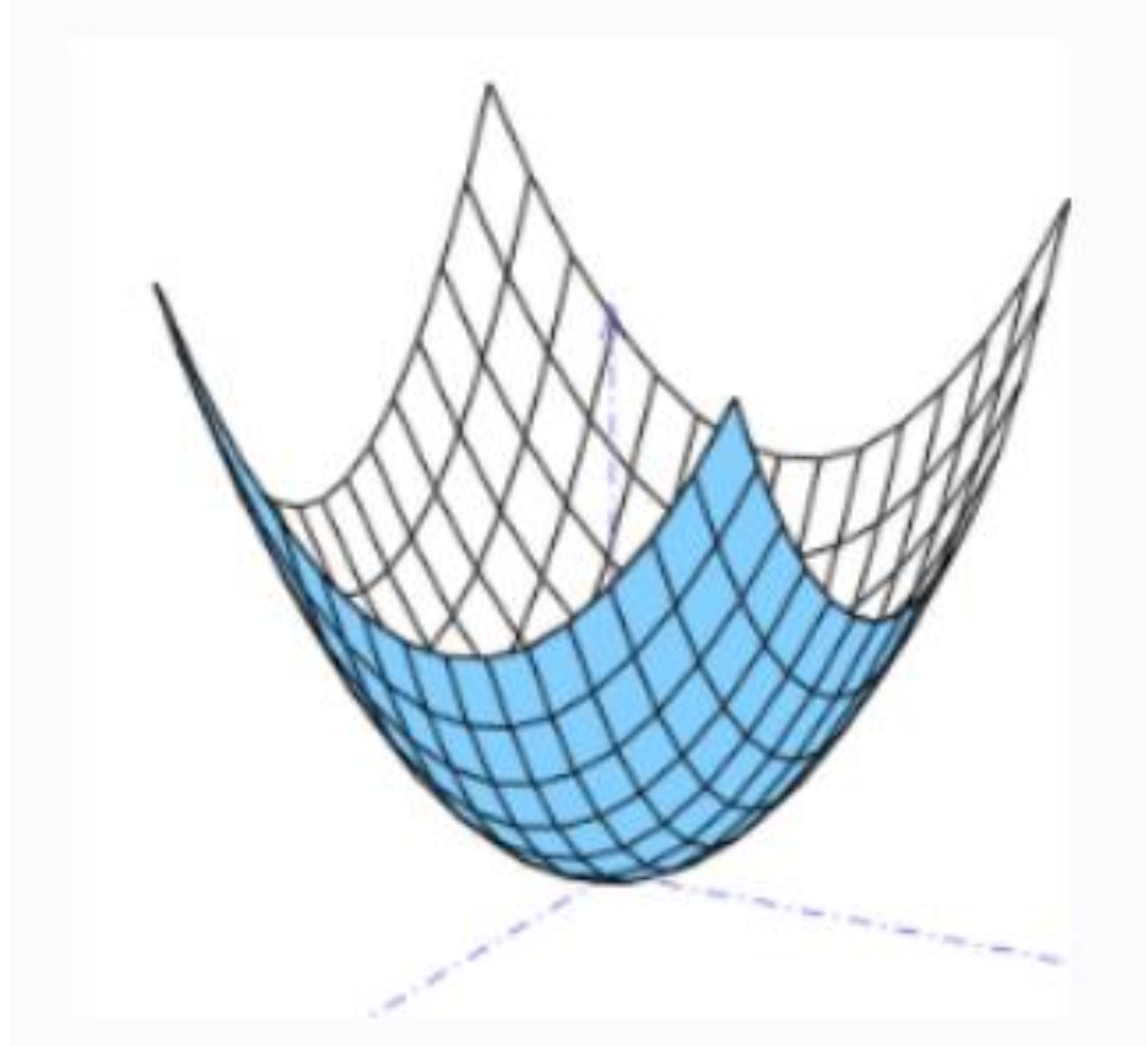
Practically it is more complex to plot the n dimensional graph.



When Linear Regression is in 2-Dimensional, the Gradient Descent Error Graph will be formed as a Parabola.



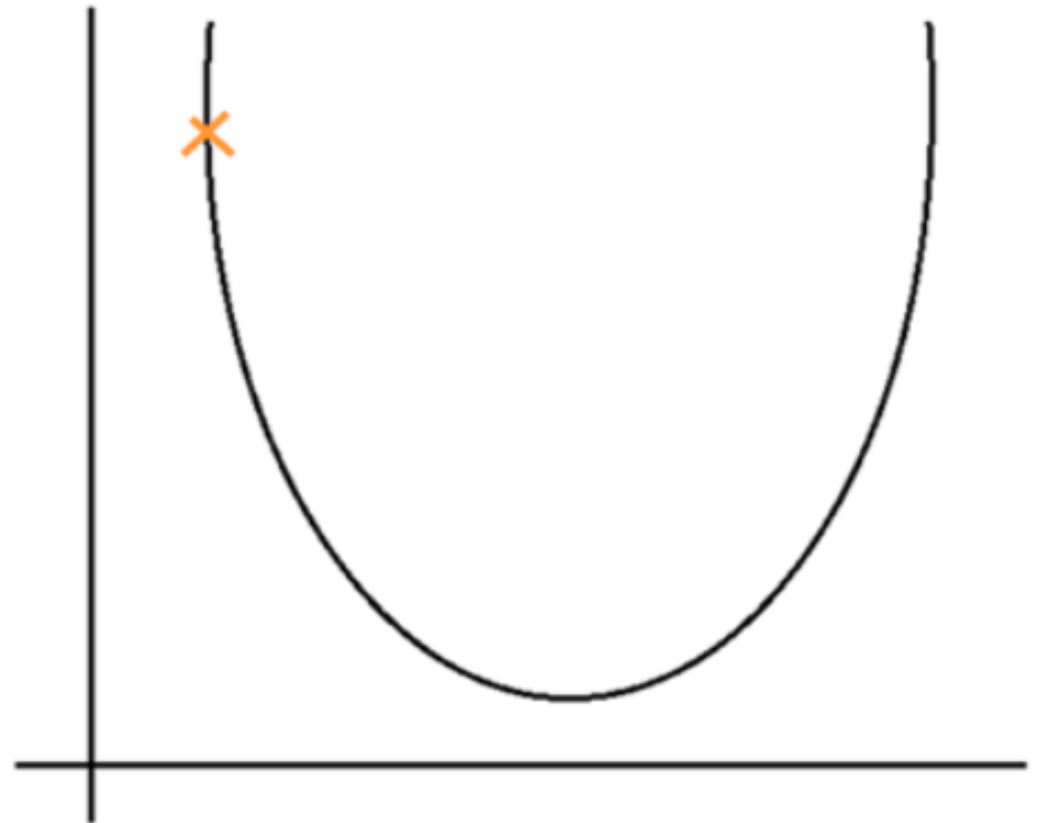
When the Linear Regression is in 3 Dimensional, the Gradient Descent will become elliptic paraboloid.



How to find the minimum cost?

We found that all the possible error values form a parabola (or paraboloid in 3D) and the most minimum lies in the deepest of the curve.

Step #1: Take a random value for m & c . For any random values of m & c , the error value will not necessarily be minimum. Let's say it is somewhere

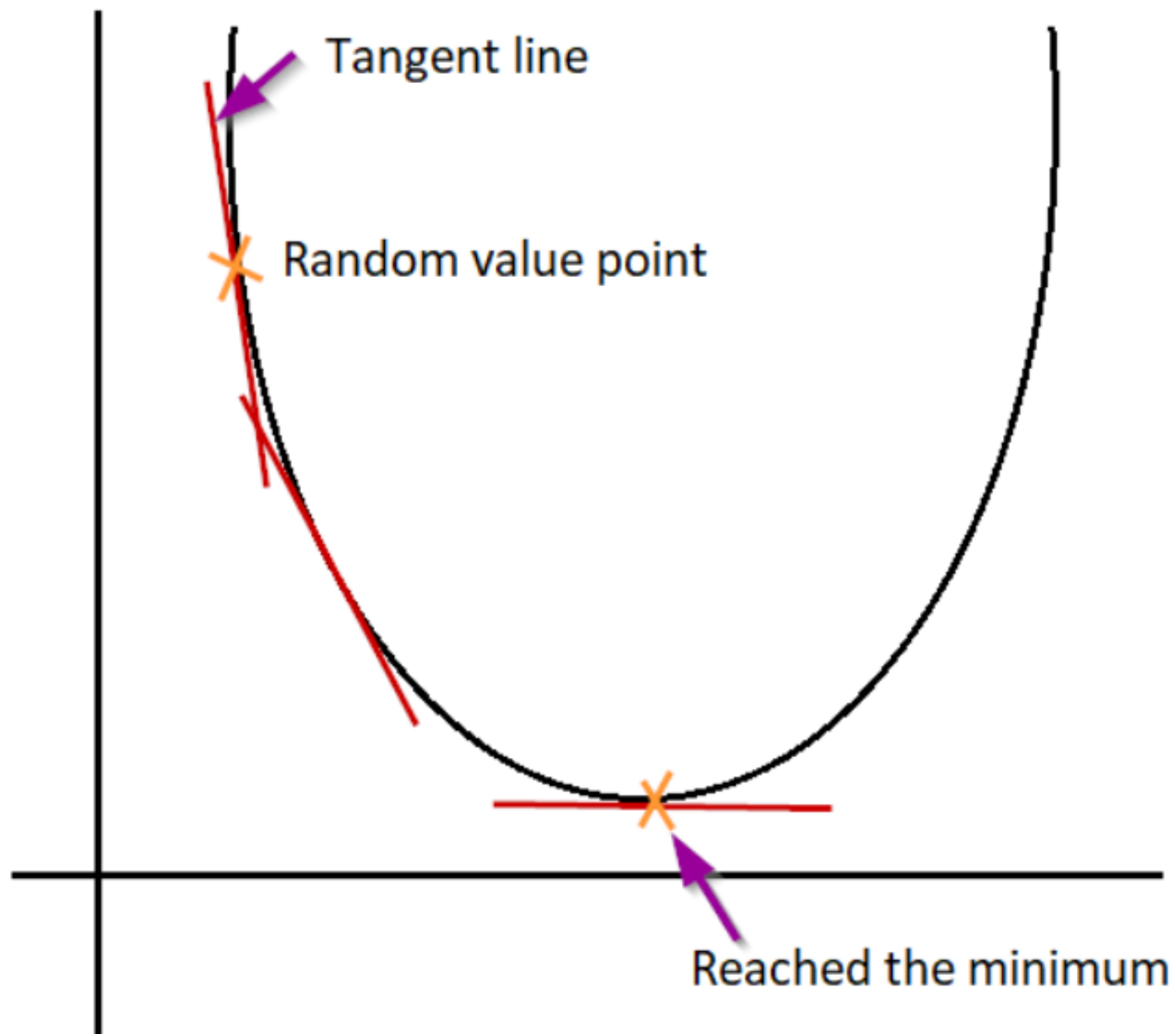


Step #2: Now we need to find, in which direction the slope is reducing so that we can reach the deep of the parabola.

This is when partial derivatives come to play.

We can compare descending from top of a parabola/paraboloid, we have to reduce the slope of the curve to reach its bottom.

If we draw a tangent line in the point which you took as a random value, it is the slope of the curve in that point. Reducing the slope of the line will move down in the curve.



In which rate we have reduce the value is reducing the slope of the line.

Basically, to reduce the slope, we can use Partial Derivative in first order to get the reduced slope.

Below is the update rule for m & c using partial derivative.

Update Rule:

$$m := m - \alpha \frac{\partial f(m,c)}{\partial m}$$

$$c := c - \alpha \frac{\partial f(m,c)}{\partial c}$$

α – Learning rate. usually a constant of 0.01

Step #3: We need to change the m & c values using the update rule and know the new SSE value.

Iterate this process until we reach the global minimum.

Points to remember:

1. m & c should be updated simultaneously. We should not update m first then apply the value of new m to update c .
2. Learning rate is usually 0.01 but not necessarily to be 0.01 for all models. Highest learning rate will reduce the m & c values rapidly and it may skip the global minimum (a fancy word to mention the deepest point in parabola) value.
3. Even for a fixed value of learning rate, once we are near to the global minimum, your steps will be automatically smaller. We do not need to reduce the learning rate often.

Conclusion:

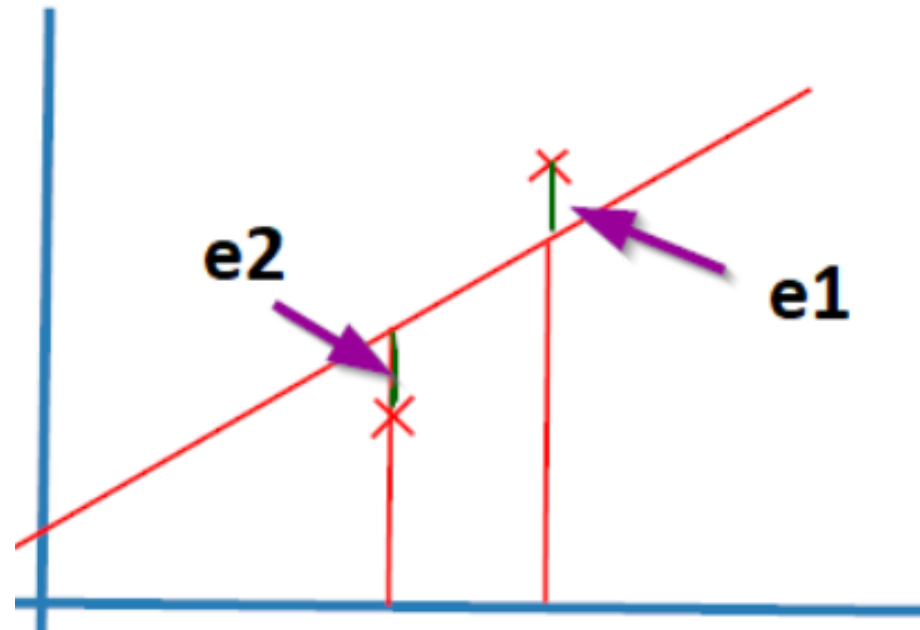
- 1.Gradient Descent is an iterative method, we will be taking a random value for slope m & intercept c of a best fit line.
- 2.The resultant SSE value (J or Cost) value may not be the minimum.
- 3.Identify the direction of most negative slope and step in to that direction. This is where partial derivatives enter the play.
- 4.Then we will do this iteratively with a learning rate until we get the minimum of all errors.
- 5.For the final minimum SSE, the used m & c value are the final outcome. The line set with final m & c is the best fit line.

R Squared

- R Squared is one of the metrics by which we can find the accuracy of a model that we create.
- R squared metrics works only if the **regression model is linear**.

SSE — Sum of Squares of Residuals (Errors)

SSR is the sum of all the difference between the original and predicted values.

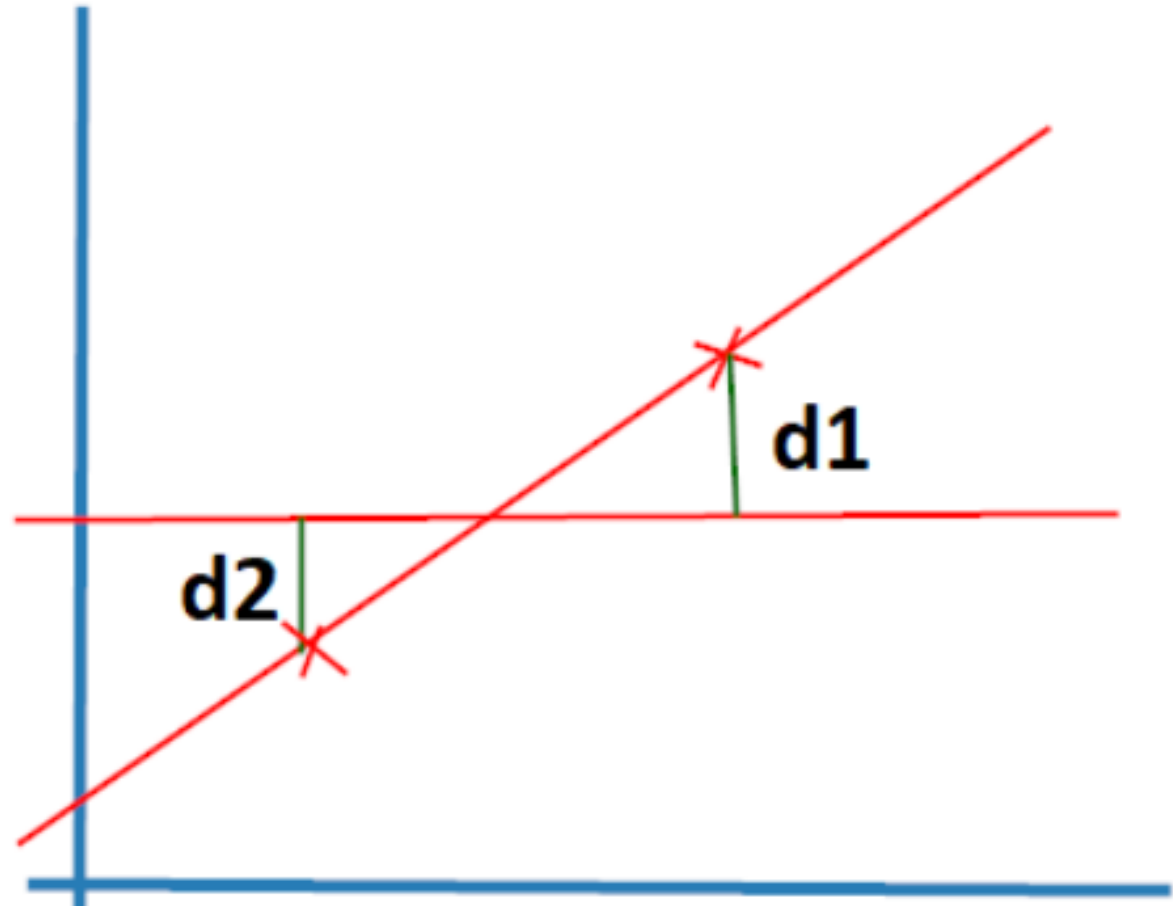


$$R = 1 - \frac{SSE}{SST}$$

Here $SSR = e_1 + e_2 + \dots + e_n$

SST — Sum of Squares of Total

SST is the sum of all the distances between the Y predicted values and the Y mean value.



$$SST = d1 + d2 + + dn$$

SSR is the numerator and SST is the denominator.

If the SSR value is less than SST, then the SSR/SST value will be less than 1.

So, as the SSR value decreases, the SSR/SST value will also move closer to 0.

As R^2 is $1 - (SSR/SST)$, the R^2 value (accuracy) will be high, as much as the SSR/SST value is low.

When the residuals (SSR) are less, the accuracy will be high.

Conclusion:

R Squared is an error metrics of a Linear Regression model which is the measure of accuracy of the model.

Accuracy will be high as much as the residuals are low.

**Training and testing the Linear Regression model and prediction
for chance of admission as well.**

Reading Data:

Import necessary libraries to read the data.

```
1 | import pandas as pd
```

```
1 | df = pd.read_csv("Admission_Predict_Ver1_1.csv")  
2 | df.head()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

Data Pre-Processing:

Our next step is Pre-Processing the data to get more accurate results.

```
1 | df.isnull().sum()
```

```
Serial No.      0
GRE Score       0
TOEFL Score     0
University Rating 0
SOP             0
LOR             0
CGPA            0
Research        0
Chance of Admit 0
dtype: int64
```

df.corr()

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
Serial No.	1.000000	-0.103839	-0.141696	-0.067641	-0.137352	-0.003694	-0.074289	-0.005332	0.008505
GRE Score	-0.103839	1.000000	0.827200	0.635376	0.613498	0.524679	0.825878	0.563398	0.810351
TOEFL Score	-0.141696	0.827200	1.000000	0.649799	0.644410	0.541563	0.810574	0.467012	0.792228
University Rating	-0.067641	0.635376	0.649799	1.000000	0.728024	0.608651	0.705254	0.427047	0.690132
SOP	-0.137352	0.613498	0.644410	0.728024	1.000000	0.663707	0.712154	0.408116	0.684137
LOR	-0.003694	0.524679	0.541563	0.608651	0.663707	1.000000	0.637469	0.372526	0.645365
CGPA	-0.074289	0.825878	0.810574	0.705254	0.712154	0.637469	1.000000	0.501311	0.882413
Research	-0.005332	0.563398	0.467012	0.427047	0.408116	0.372526	0.501311	1.000000	0.545871
Chance of Admit	0.008505	0.810351	0.792228	0.690132	0.684137	0.645365	0.882413	0.545871	1.000000

```
df.columnsIndex(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',  
'LOR ', 'CGPA', 'Research', 'Chance of Admit '], dtype='object')
```

- Out of all the columns, S.No is not related to the Chance of Admit. So remove it.
-
- The value that is going to be predicted is 'Chance of Admit'. Having this column in X does not make sense as change in this column will be exact equal to the change in Y which is the same column for which we are going to predict values.
- Removed 'Serial. No' & 'Chance of Admit' from X and have only 'Chance of Admit' in Y.

```
1 | X = df.drop(['Serial No.', 'Chance of Admit '], axis = 1)
2 | y = df['Chance of Admit ']
```

Split the entire data frame into X & Y parameters. Basically the X parameters are the Independent Variables and Y parameter is the Dependent Variable that we need to predict. Lets look at the X & Y variables.

```
1 | X.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	337	118	4	4.5	4.5	9.65	1
1	324	107	4	4.0	4.5	8.87	1
2	316	104	3	3.0	3.5	8.00	1
3	322	110	3	3.5	2.5	8.67	1
4	314	103	2	2.0	3.0	8.21	0

```
1 | y.head()
```

```
0    0.92
1    0.76
2    0.72
3    0.80
4    0.65
Name: Chance of Admit , dtype: float64
```


Test Train Split:

- Now we need to split the data into train & test sets. We train our model with train set and verify the model with test set.
- Usually the train set should be higher than test set so that we get more data to for learning. Also the accuracy will be different for different split.
Example splits are 60–40, 80–20 or even 73–27.
- We can mention the test size so that the train set will be automatically the remaining.
But how to select which rows for test?
- This split should be done randomly.
- sklearn has package `train_test_split` which allows us to split the data into train and test in a random manner.
- We also control the randomness using the parameter **random_state**.
- **test_size** parameter takes value ≤ 1 i.e. 0.33, 0.4 to mention the percentage of data for test set.

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.linear_model import LinearRegression
3 from sklearn.metrics import r2_score
4
5 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

Lets see how many rows train & test data set has:

```
1 | X_train.shape
```

Output: (300, 7)

```
1 | X_test.shape
```

Output: (200, 7)

Creating Linear Regression Model:

Lets create a Linear Regression model to predict the Chance of Admit of a new student.

sklearn provides a lots of packages for various algorithms.

Here we are using LinearRegression() which creates a new object for this algorithm. with this object, we can fit our data and use this object to get the parameters of our model.

```
1 | lr = LinearRegression() #Object Created
2 | lr.fit(X_train, y_train) #Data given into the LinearReg. object
```

```
1 | lr.coef_
```

Output:

```
array([0.00145221, 0.00302388, 0.00809642, 0.00672185, 0.01318406,  
       0.12002891, 0.02477235])
```

- These are the Coefficients (slopes) of the X Variables — GRE Score, TOEFL Score, University Rating, SOP, LOR, CGPA, Research.
- So $0.00174541 \text{ GRE} + 0.00280216 \text{ TOEFL} + 0.00675831 \text{ University} + 0.0061299 \text{ SOP} + 0.01492133 \text{ LOR} + 0.11902878 \text{ CGPA} + 0.01943883 * \text{Research}$

```
1 | lr.intercept_
```

Output:

```
-1.200250569689947
```

This value is the Y Intercept. So our Y line equation becomes,

$$\text{ChanceOfAdmit} = -1.2567425309612157 + 0.00174541 \text{ GRE} + 0.00280216 \text{ TOEFL} + 0.00675831 \text{ University} + 0.0061299 \text{ SOP} + 0.01492133 \text{ LOR} + 0.11902878 \text{ CGPA} + 0.01943883 * \text{Research}$$

Lets test our model.

Using the predict method of the *lr* object, by sending in the x_test values, we can predict the y_test values.

As we already have the original y_test values, we can now validate our model.

```
1 | predicted = lr.predict(X_test)
2 | r2_score(y_test, predicted)
```

Output:

```
0.8251757711467193
```

We have predicted Y values for test set and compared it with the true values using the `r1_score` function from the package `sklearn.metrics`.

Model's accuracy is 0.8251757711467193. That means our predictions will be 82% correct.

```
1 | y_test.head(10)
```

Output:

18	0.63
361	0.93
104	0.74
4	0.65
156	0.70
350	0.74
32	0.91
205	0.57
81	0.96
414	0.72

Name: Chance of Admit , dtype: float64

```
1 | print(predicted[0:10])
```

Output:

```
[0.74116361 0.91082451 0.81024774 0.62183042 0.64643854 0.69311918  
0.91233928 0.51828756 0.95333009 0.73419915]
```


Here we have predicted for the test data we had.

Now you can see that most of our predictions are very close.

Lets predict for new data.

```
1 | new_y = lr.predict([[347, 120, 4.5, 4.7, 4.7, 9.8, 1]])  
2 | print(new_y)
```

Output:

```
[0.99758071]
```

Another test:

```
1 | new_y = lr.predict([[250, 90, 2, 2.5, 4.7, 7, 1]])  
2 | print(new_y)
```

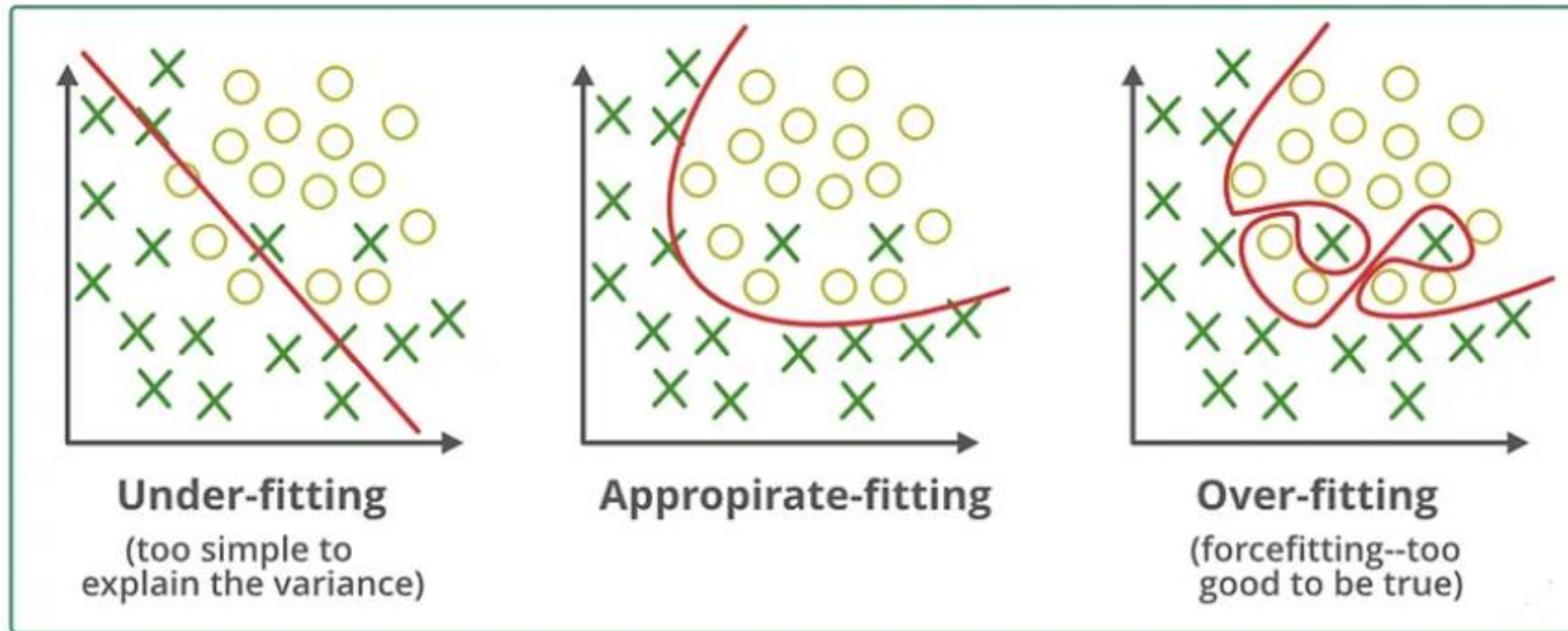
Output:

Now we have got a model that can predict Chance of admission for any scores.

Overfitting and Underfitting

Overfitting occurs when the model is very complex and fits the training data very closely. This will result in poor generalization of the model. This means the model performs well on training data but it will not be able to predict accurate outcomes for new, unseen data.

Underfitting occurs when a model is too simple and is unable to properly capture the patterns and relationships in the data. This means the model will perform poorly on both the training and the test data.



A model is said to be a good machine learning model if it generalizes any new input data from the problem domain in a proper way.

The aim of the machine-learning model should be to get good training and good test accuracy.

What causes Overfitting and Underfitting?

- Overfitting is often caused by using a model with too many parameters or if the model is too powerful for the given dataset.
- On the other hand, underfitting is often caused by the model with too few parameters or by using a model that is not powerful enough for the given dataset.

Bias and Variance

- Bias and Variance are two errors that can severely impact the performance of the machine-learning model.
- If a model has a very good training accuracy it means the model has low variance but if the training accuracy is bad then the model has high variance.
- If the model has bad test accuracy then it has a high variance but if the test accuracy is good this means the model has low variance.

Bias and Variance in Machine Learning

Bias:

- Bias refers to the error due to overly simplistic assumptions in the learning algorithm.
- These assumptions make the model easier to comprehend and learn but might not capture the underlying complexities of the data.
- It is the error due to the model's inability to represent the true relationship between input and output accurately.
- When a model has poor performance both on the training and testing data means high bias because of the simple model, indicating underfitting.

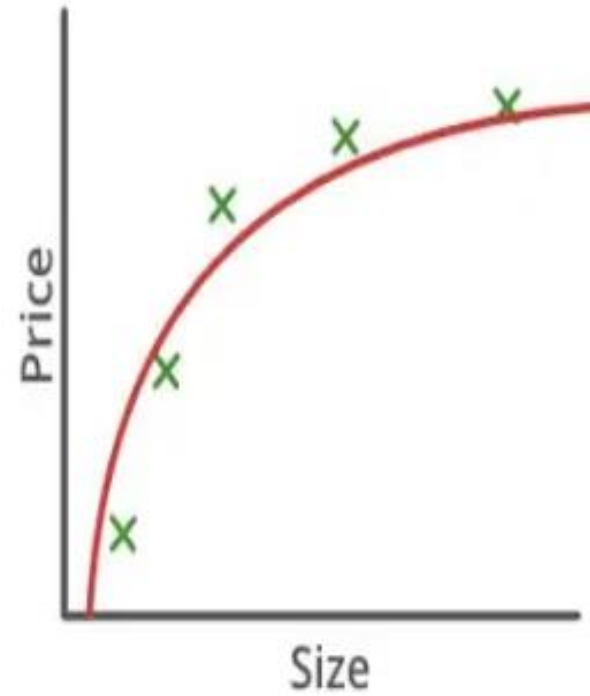
Variance:

- Variance, is the error due to the model's sensitivity to fluctuations in the training data.
- It's the variability of the model's predictions for different instances of training data.
- High variance occurs when a model learns the training data's noise and random fluctuations rather than the underlying pattern.
- As a result, the model performs well on the training data but poorly on the testing data, indicating overfitting.



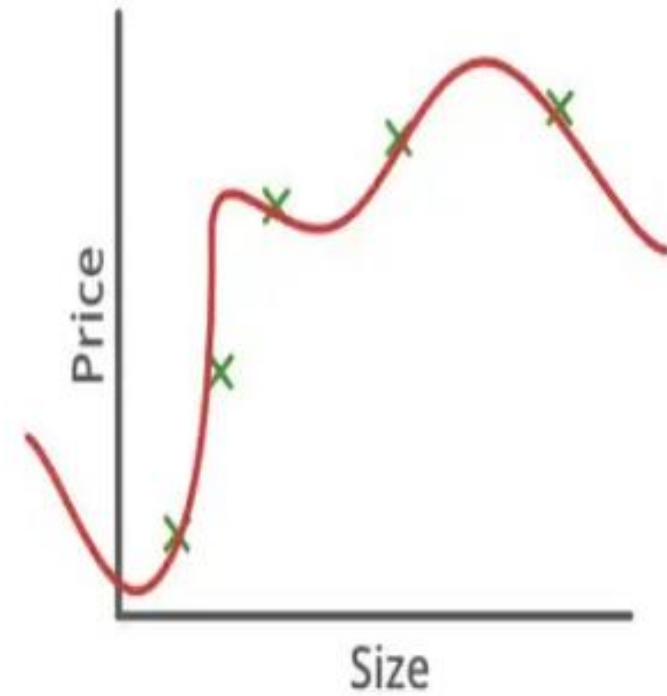
$$\theta_0 + \theta_1 x$$

High Bias
(Underfitting)



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

Low Bias, Low Variance
(Goodfitting)



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

High Variance
(Overfitting)

Underfitting in Machine Learning

- A machine learning algorithm is said to have underfitting when a model is too simple to capture data complexities.
- It represents the inability of the model to learn the training data effectively result in poor performance both on the training and testing data.
- In simple terms, an underfit model's are inaccurate, especially when applied to new, unseen examples.
- It mainly happens when we uses very simple model with overly simplified assumptions.
- To address underfitting problem of the model, we need to use more complex models, with enhanced feature representation, and less regularization.

Note: The underfitting model has High bias and low variance.

Reasons for Underfitting

- 1.The model is too simple, So it may be not capable to represent the complexities in the data.
- 2.The input features which is used to train the model is not the adequate representations of underlying factors influencing the target variable.
- 3.The size of the training dataset used is not enough.
- 4.Excessive regularization are used to prevent the overfitting, which constraint the model to capture the data well.
- 5.Features are not scaled.

Techniques to Reduce Underfitting

- 1.Increase model complexity.
- 2.Increase the number of features, performing [feature engineering](#).
- 3.Remove noise from the data.
- 4.Increase the number of [epochs](#) or increase the duration of training to get better results.

Overfitting in Machine Learning

- A Machine Learning model is said to be overfitted when the model does not make accurate predictions on testing data.
- When a model gets trained with so much data, it starts learning from the noise and inaccurate data entries in our data set. And when testing with test data results in High variance. Then the model does not categorize the data correctly, because of too many details and noise.
- The causes of overfitting are the non-parametric and non-linear methods because these types of machine learning algorithms have more freedom in building the model based on the dataset and therefore they can really build unrealistic models.
- A solution to avoid overfitting is using a linear algorithm if we have linear data or using the parameters like the maximal depth if we are using decision trees.

In a nutshell, [Overfitting](#) is a problem where the evaluation of machine learning algorithms on training data is different from unseen data.

Reasons for Overfitting:

1. High variance and low bias.
2. The model is too complex.
3. The size of the training data.

Techniques to Reduce Overfitting

1. Increase training data.
2. Reduce model complexity.
3. [Early stopping](#) during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
4. [Ridge Regularization](#) and [Lasso Regularization](#).
5. Use [dropout](#) for [neural networks](#) to tackle overfitting.

TRAIN DATA → Very Good Accuracy → Low Bias

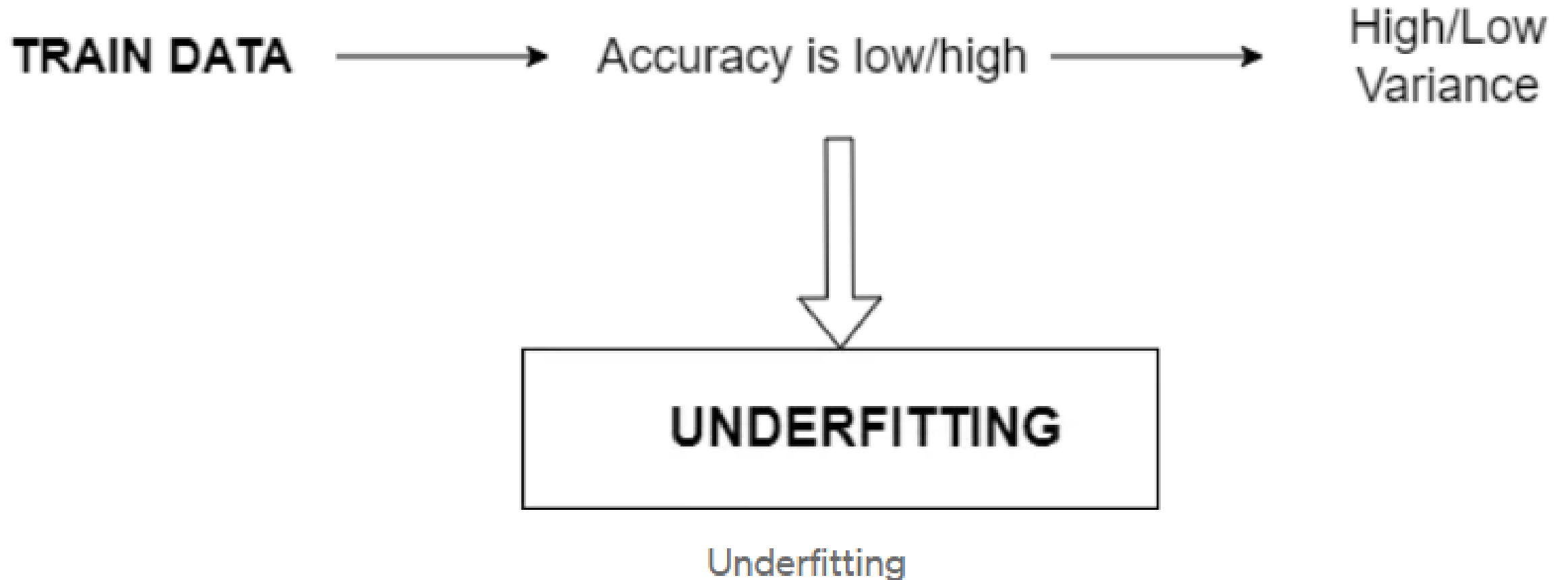
TRAIN DATA → Bad Accuracy → High Variance



OVERFITTING

Overfitting

A model with high variance and little bias will overfit the target.



Note: The underfitting model has High bias and low variance.

How to avoid Overfitting and Underfitting?

- To solve the issue of overfitting and underfitting, it is important to choose an appropriate model for the given dataset.
- Hyper-performance tuning can also be performed.
- For overfitting reducing the model complexity can help similarly for underfitting the model complexity can be increased.
- As overfitting is caused due to too many features in the dataset and underfitting is caused by too few features so during feature engineering the numbers of features can be decreased and increased to avoid overfitting and underfitting respectively.

Conclusion

- Overfitting and Underfitting are two very common issues in machine learning.
- Both overfitting and underfitting can impact the model's performance.
- Overfitting occurs when the model is complex and fits the data closely while underfitting occurs when the model is too simple and is unable to find relationships and patterns accurately.
- It is very important to recognise both these issues while building the model and deal with them to improve its performance of the model.