Regression

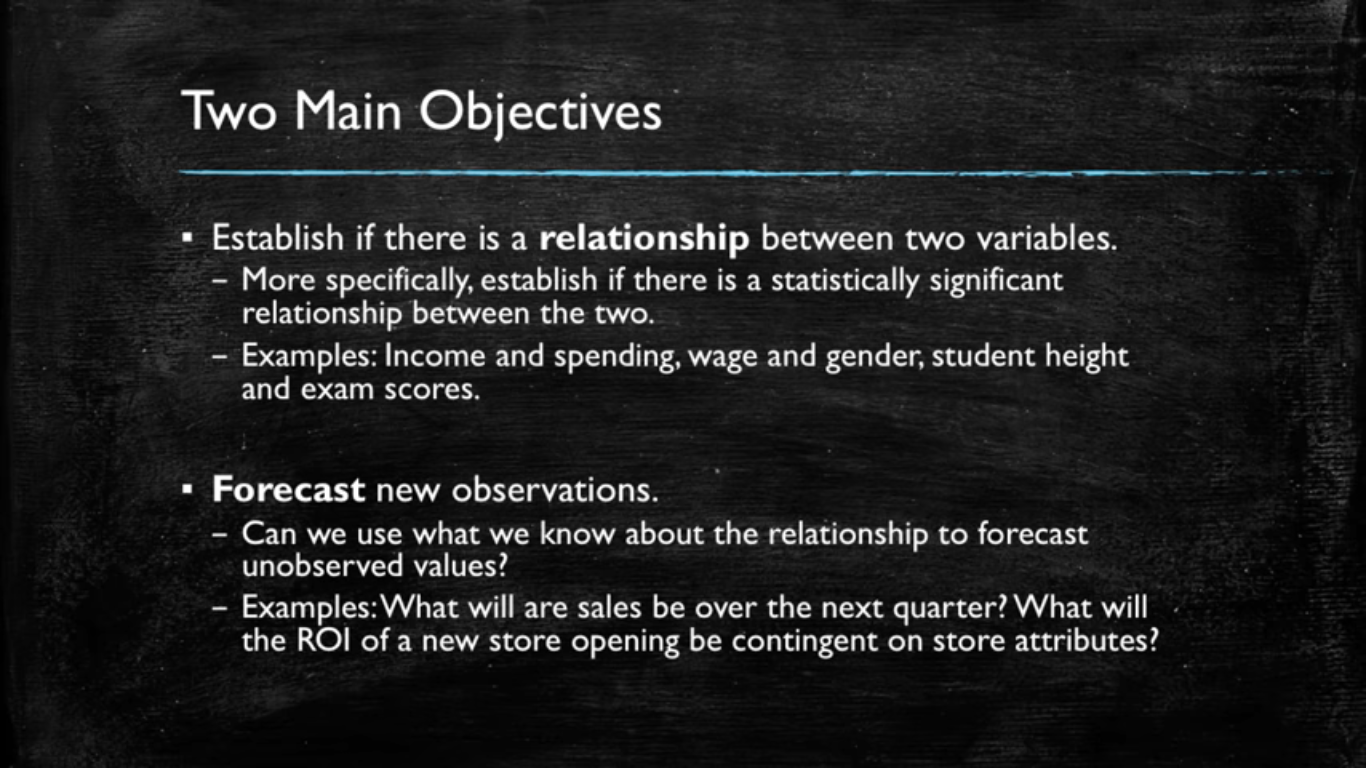
What is a regression algorithm?

Regression algorithms predict the output values based on input features from the data fed in the system. The go-to methodology is the algorithm builds a model on the features of training data and using the model to predict the value for new data.

There are 3 types of regression,

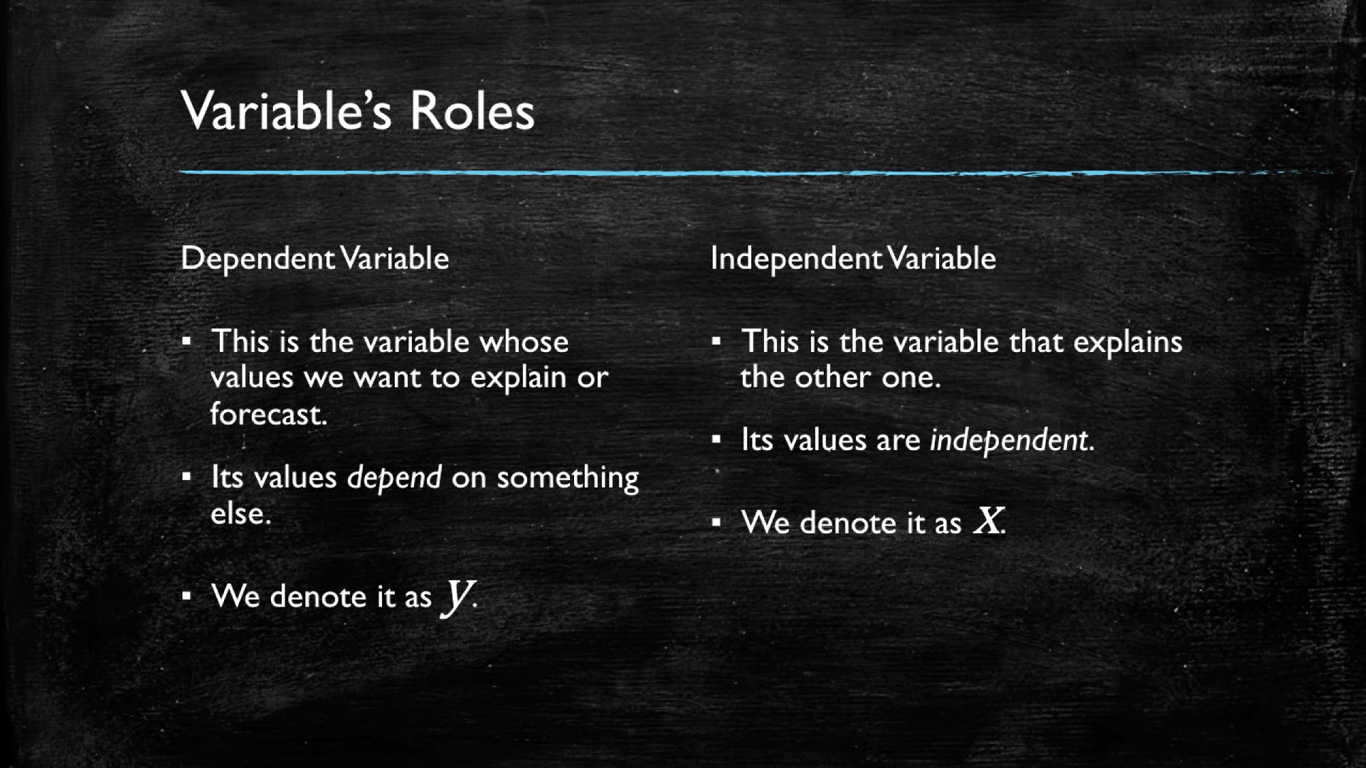
1. Linear Regression
2. Logistic Regression
3. Poisson Regression

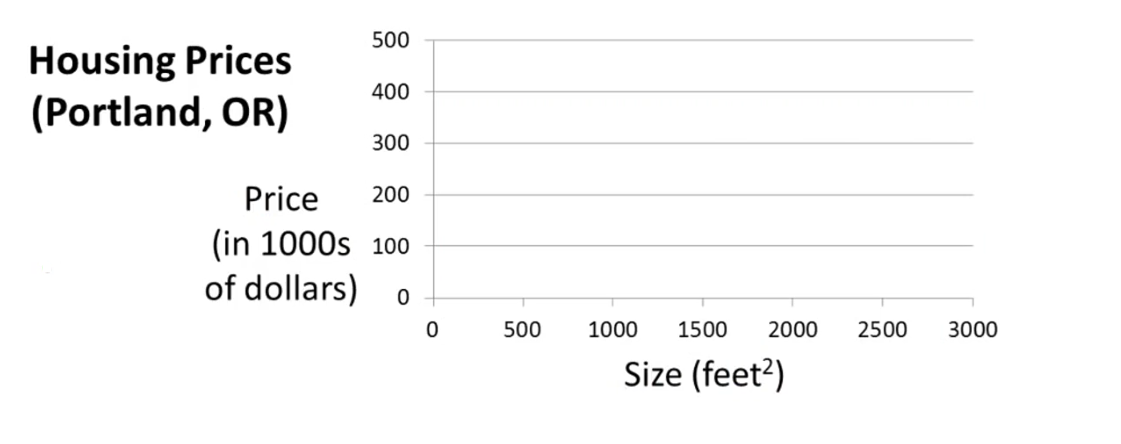
Slide Reference: https://www.youtube.com/watch?v=owI7zxCqNY0



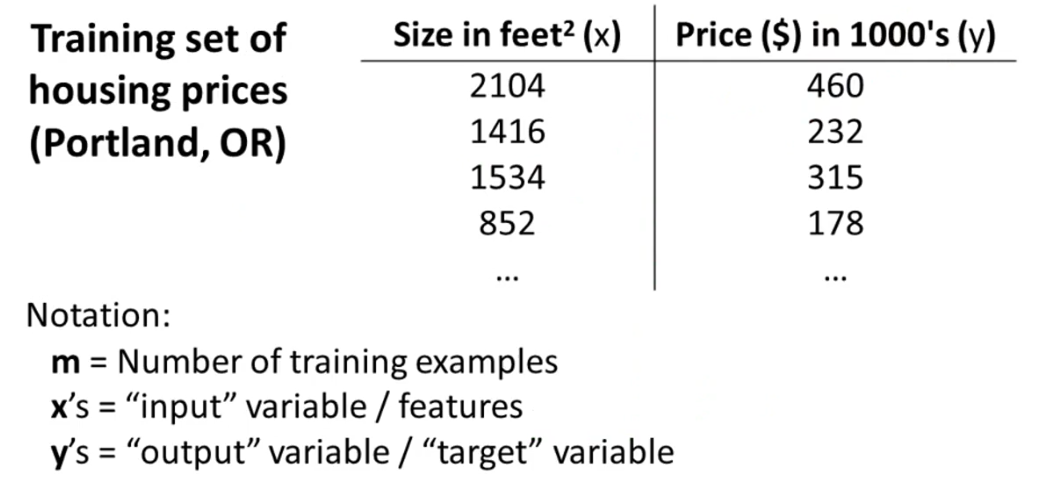
Dependent or the variable to be found is always taken on the Y-axis.

Independent variables are taken on the X-axis.





Here the dependent variable is price and size in square ft is independent variable. We will be given with the size and we will be asked to find the price on that size.



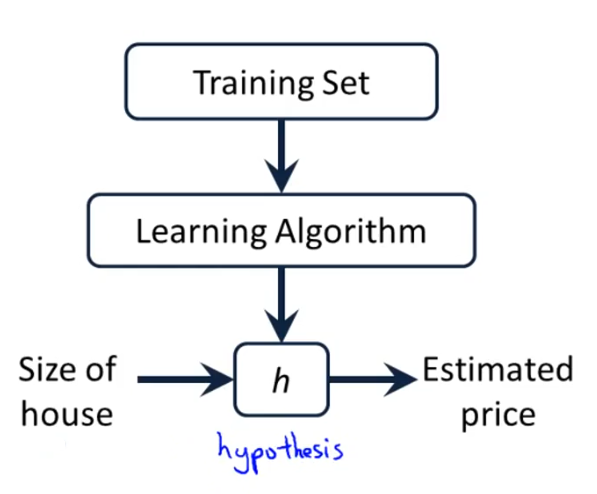
m -> Total data already given (rows).

x’s -> feature (In machine learning a feature is any input. For example, an image of cat)

Example: 2104,1416 etc.

y’s -> target (In machine learning a target value is any value that is corresponding to any feature)

Example: 460,232 etc.



Where hypothesis is a function obtained from learning algorithm that takes x’s as input and gives y’s as output. Here, the training set means the data fed to the machine or a learning algorithm to predict new or upcoming results.

**Hypothesis 🡪 Regression predicting line 🡪 y = mx + c** 🡪 **y = b0 + b1x.**

In linear equations the slope ‘**m**’ OR ‘**b1**’ means that how much value of y or the dependent variable has in terms of independent variable or variable **x**.

For Example,

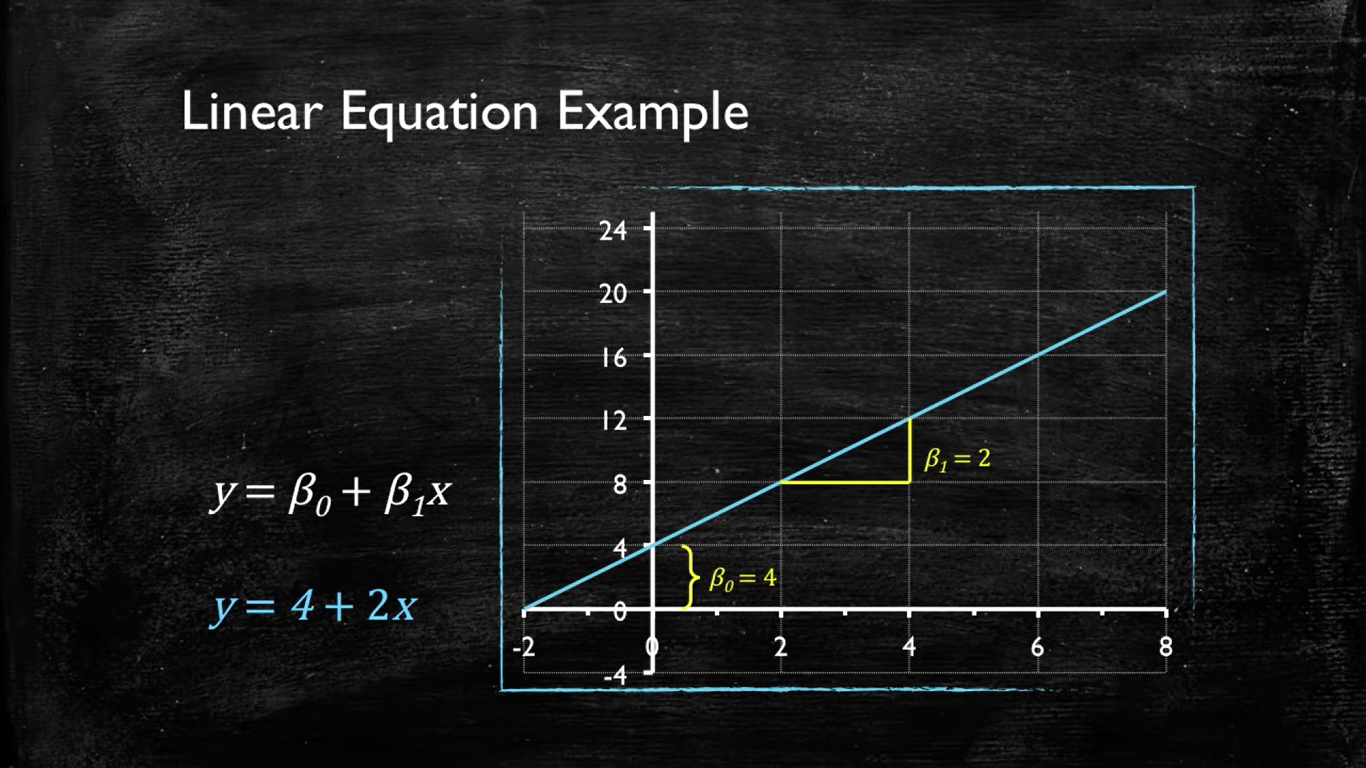
In the example below, b1 = 2 which means y is twice of x that is why when we move

2🡪4 in the X-axis, we move x2 times in Y-axis means (8🡪12).

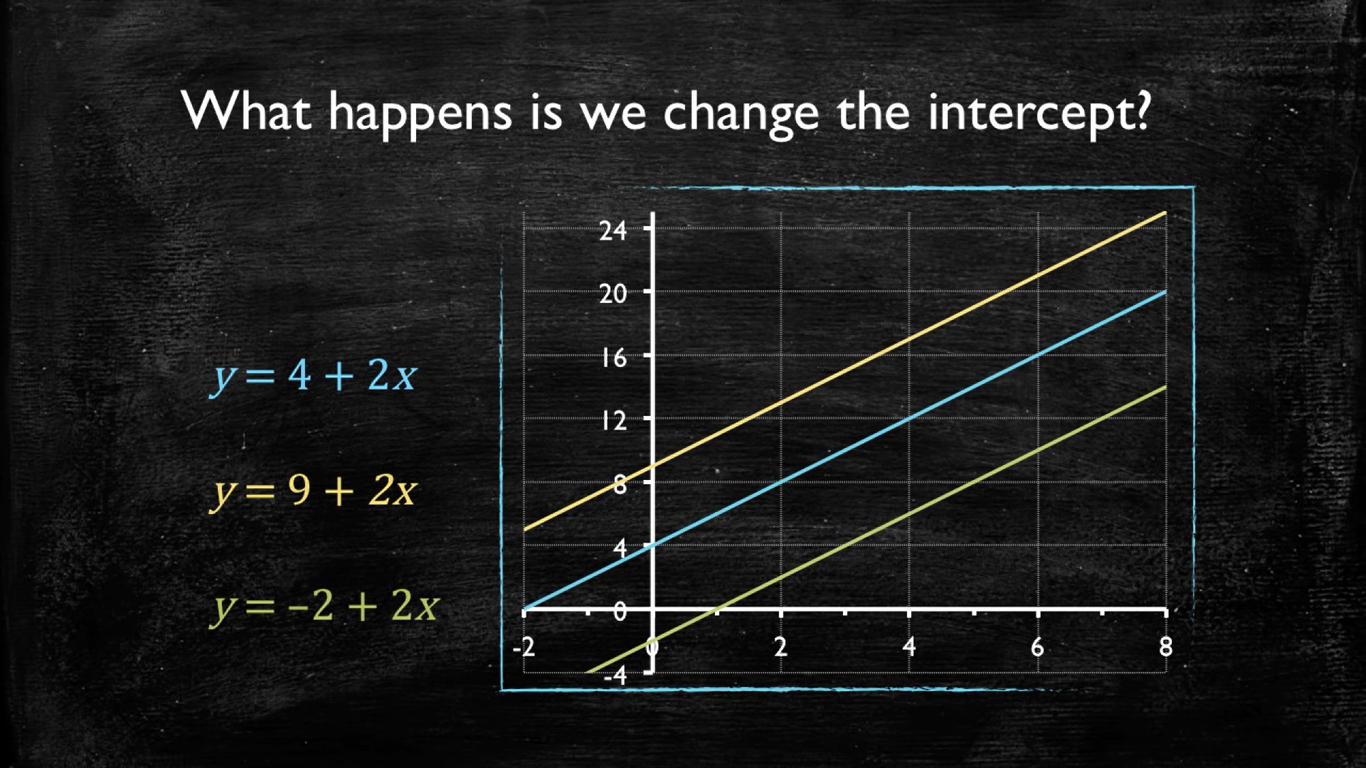
If slope was 3 then increase in y 🡪 x3 of X

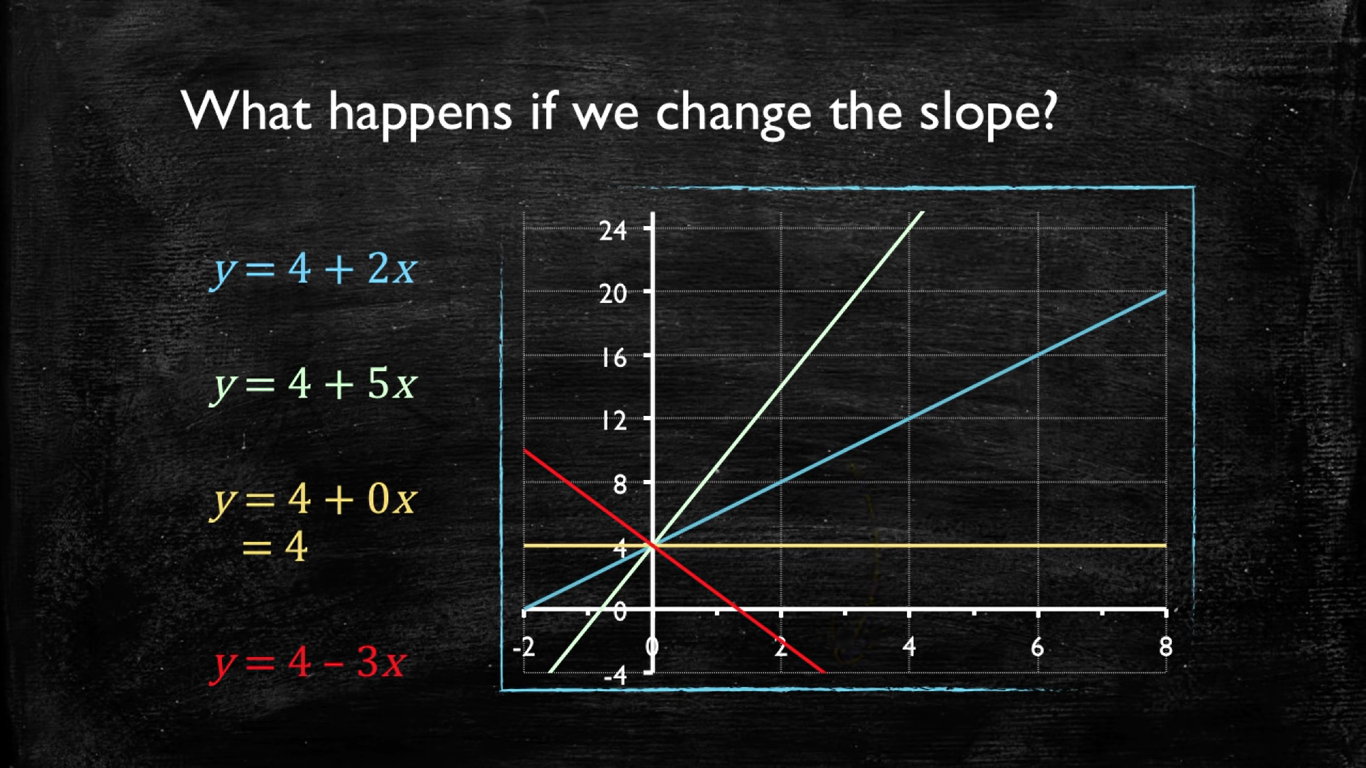
If slope was [] then increase in y 🡪 x[] of X

Where ‘**c**’ OR ‘**b0**’ is the **Y-intercept** (A point where Y-axis is intersected from the given line equation).

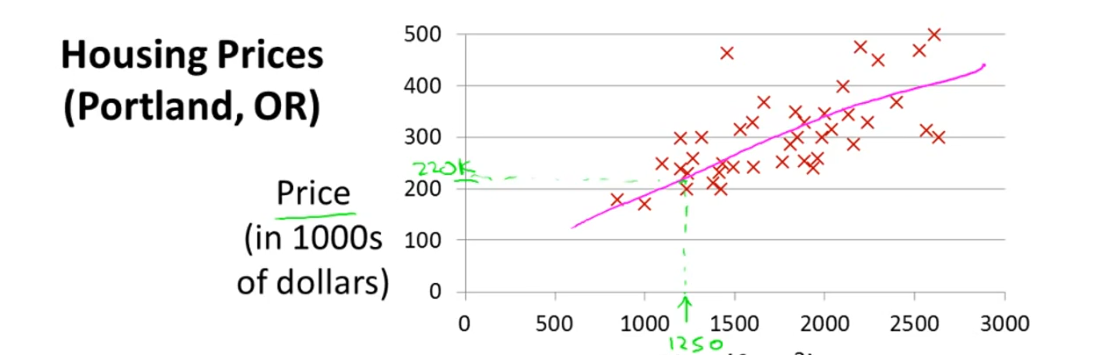


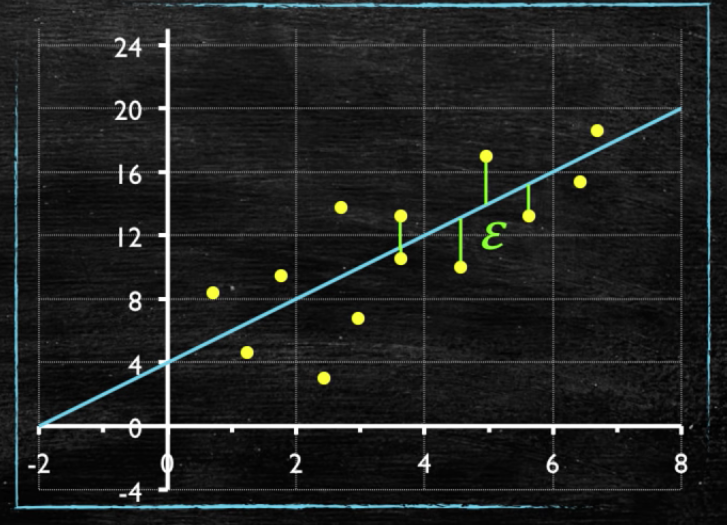
**Change in Y-intercept ‘c’ OR ‘b0’**



**Change in Slope ‘m’ OR ‘b1’**

Linear Regression with 1 variable

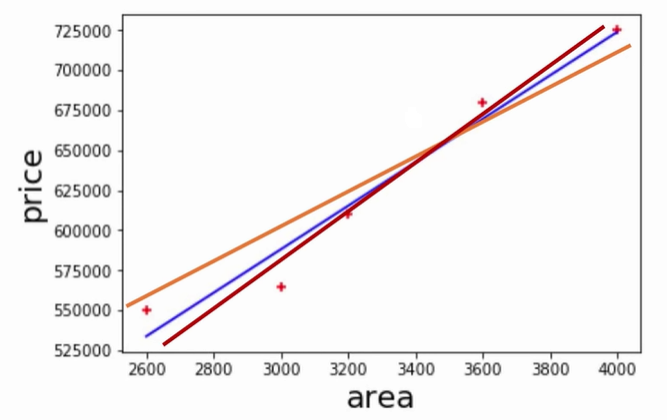


Suppose we want to find the price of the house having square feet 1250sqft now regression will help us to predict the appropriate price for our requirement because we already have some data and based on that data it will help us predicting the price. We will map/bring the given data points into a predicting line from which we will predict our result ‘1250’ but there will be error cost if we bring the actual/real value into our predicting line that error cost is founded by a cost function.  


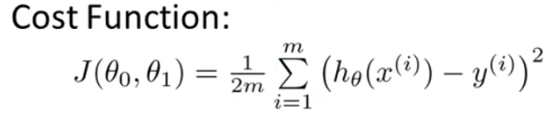
Yellow dots 🡪 Real/actual data points

Green lines 🡪 error cost

There can be ‘n’ prediction lines or hypothesis between our data and out of those we will be selecting only the only that best fits. The cost function and some methods will help us to find the best fit line. There are many cost functions available and we will be using mean squared error.



Mean Square Cost Function for Simple Regression,



The cost function **J (θ0, θ1)** takes both Y-intercept**(θ0)** and the slope**(θ1)** as argument in simple linear regression.

‘i’ here represents i th point.

h(x) 🡪 Hypothesis function **OR** Regression prediction line (y = mx + c)

Y🡪 Real/actual value.

Sigma or Summation represents the distance from Regression prediction line(y) to all the given or real value data points(y’).

1/2m is representing the average so that we can find the average/mean error cost for all real data points whose distance was calculated through sigma. The 1/2m is only for averaging purpose.

(h(x)-Y) ^2 squaring is done for making negatives 🡪 positives **OR** for making absolute values of error distance.

**What is the purpose of Cost function?**

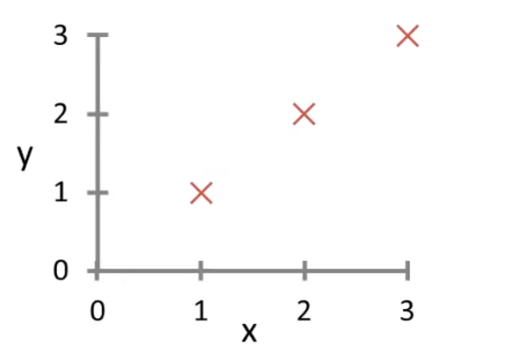
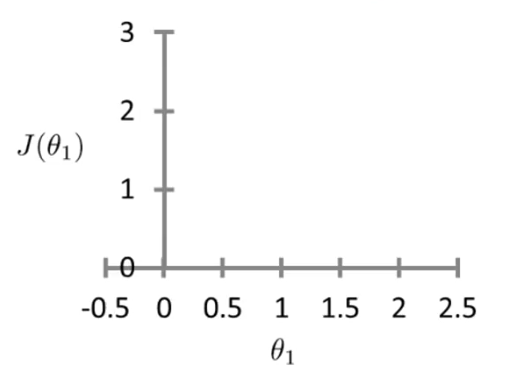
The mean squared error or the cost function tells you how close a regression line is to a set of real data points.

**“We give cost function a predicting regression line as an input an it tells us the cost of it which means, is our line is good enough to give minimal/good cost or not? (either it is a best fit line or not).”**

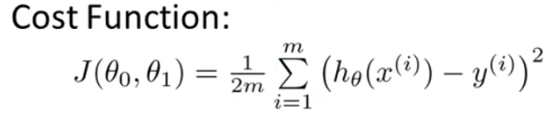
In ML, cost functions are used to estimate how badly models are performing. Put simply, a cost function is a measure of how wrong the model is in terms of its ability to estimate the relationship between X and Y. This is typically expressed as a difference or distance between the predicted value and the actual value.

**Hypothesis / Regression Line Function Graph Cost Function Graph**



The graph on the left hand side is the graph of our hypothesis or the regression predicting line and the graph on the right hand side is the graph of our cost function. The given data points are (1,1), (2,2), and (3,3) and we will now take some hypothesis functions / regression prediction lines for these 3 given data points and will draw the prediction line on the left hand side graph and on the right hand side graph we will be marking cost for each prediction line in terms of the slope of each prediction line. The general hypothesis function is,



But in this example, in order to make things simpler and for understanding purpose we are assuming that **θ0 = 0** which means our prediction line will be passing from the origin. Therefore, the cost function will now look like,

In this example, we will work on the second cost function in which we have assumed that θ0=0. Therefore, hypothesis becomes h(x) = 0 + θx => **θx**

**Cost function 🡪 J(θ1) = 1/2m ∑ (θ1x – y) ^2**

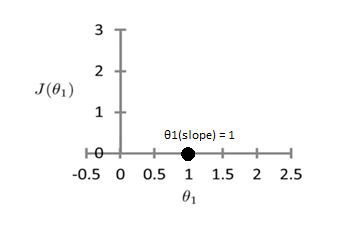
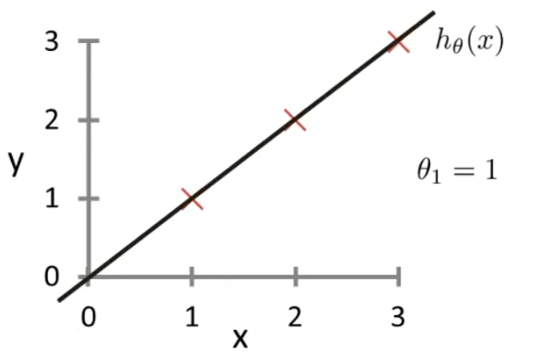
**Graphs Notations**

Black line 🡪 hypothesis function **OR** prediction line.

Black dots 🡪 cost when slope or **θ1** was ‘\_’.

**Cost for first hypothesis or first prediction line**





Calculation:

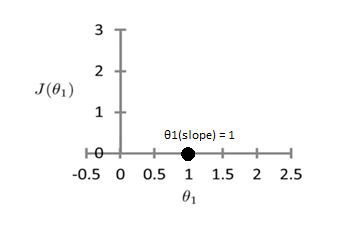
Applying cost function

**Cost function 🡪 J(θ1) = 1/2m ∑ (θ1x – y) ^2**

**Slope θ1 = 1 (Assuming first prediction line with slope ‘1’)**

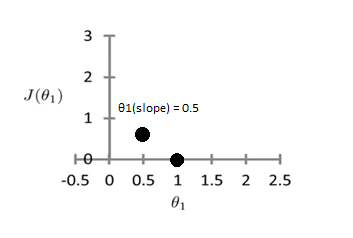
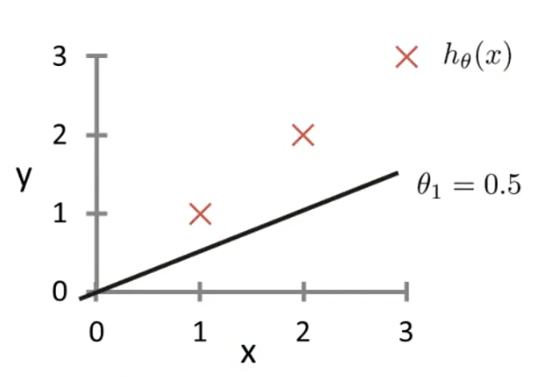
* J(θ1) = 1/2m ∑(h(x) – y) ^2
* J (1) = 1/2(3) [((θ1) \*(x1)-y1) ² + ((θ1) \*(x2)-y2) ² + ((θ1) \*(x3)-y3) ²]
* J (1) = 1/6 [((1) \*(1)-1) ² + ((1) \*(2)-2) ² + ((1) \*(3)-3) ²]
* J (1) = 1/6 [0+0+0]
* J (1) = 0

J(θ1) 🡪 J (1) = 0 this is the cost that we have got for the first prediction line with slope ‘1’. This will now be plotted on the right-hand side graph (cost function graph).



**Cost for second hypothesis or second prediction line**





Calculation:

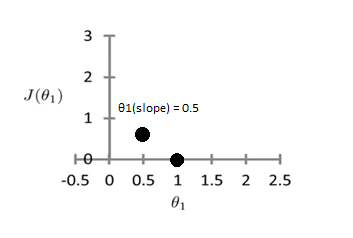
Applying cost function

**Cost function 🡪 J(θ1) = 1/2m ∑ (θ1x – y) ^2**

**Slope θ1 = 0.5 (Assuming second prediction line with slope ‘0.5’)**

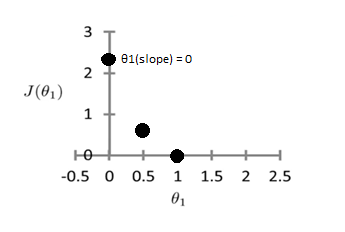
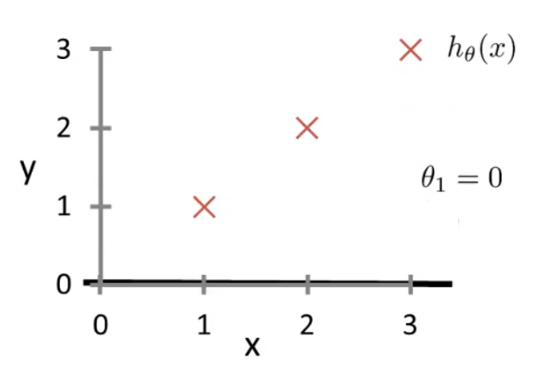
* J(θ1) = 1/2m ∑(h(x) – y) ^2
* J (0.5) = 1/2(3) [((θ1) \*(x1)-y1) ² + ((θ1) \*(x2)-y2) ² + ((θ1) \*(x3)-y3) ²]
* J (0.5) = 1/6 [((0.5) \*(1)-1) ² + ((0.5) \*(2)-2) ² + ((0.5) \*(3)-3) ²]
* J (0.5) = 0.58

J(θ1) 🡪 J (0.5) = 0.68 this is the cost that we have got for the second prediction line with slope ‘0.5’. This will now be plotted on the right-hand side graph (cost function graph).



**Cost for third hypothesis or third prediction line**





Calculation:

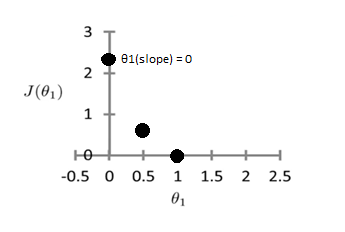
Applying cost function

**Cost function 🡪 J(θ1) = 1/2m ∑ (θ1x – y) ^2**

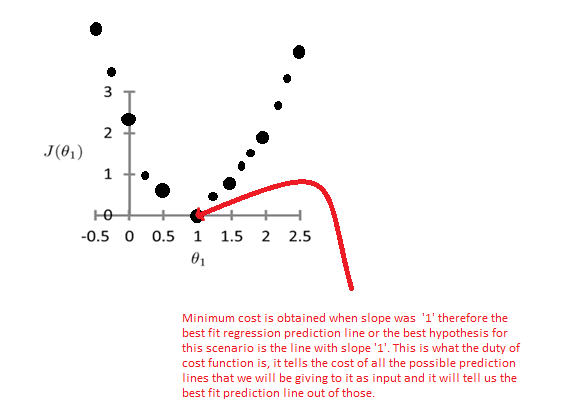
**Slope θ1 = 0 (Assuming second prediction line with slope ‘0’)**

* J(θ1) = 1/2m ∑(h(x) – y) ^2
* J (0) = 1/2(3) [((θ1) \*(x1)-y1) ² + ((θ1) \*(x2)-y2) ² + ((θ1) \*(x3)-y3) ²]
* J (0) = 1/6 [((0) \*(1)-1) ² + ((0) \*(2)-2) ² + ((0) \*(3)-3) ²]
* J (0) = 2.3

J(θ1) 🡪 J (0) = 2.3 this is the cost that we have got for the third prediction line with slope ‘0’. This will now be plotted on the right-hand side graph (cost function graph).



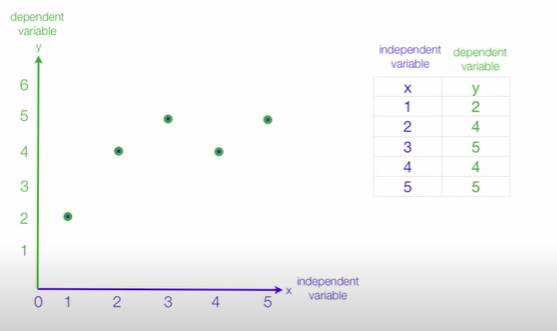
Similarly, we could have more prediction lines / hypothesis functions and out of those we should opt the one with the least most cost. **The job of a cost function is to find the best hypothesis function or the best fit regression prediction line.**



**Another way to get best fit predicting regression line**

Better or less cost can be gained if we avoid taking the predicting line randomly and take the predicting line by following the rules described in the questions given below,

**Example Question # 1**

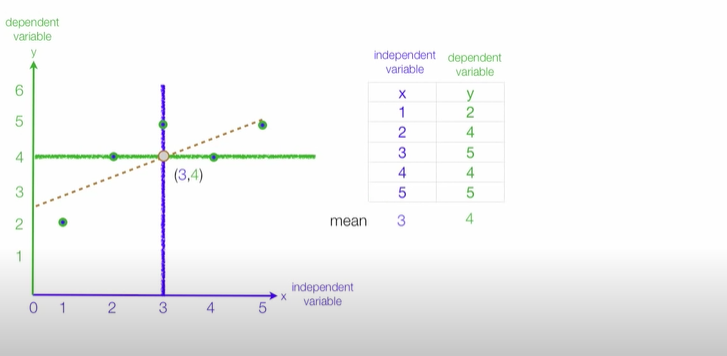


**Step # 1**

Find the mean of independent and dependent variables both from the given data points and then draw straight lines for both the mean values and make a new point (3,4) in this case which is the intersecting point of both of these mean lines.

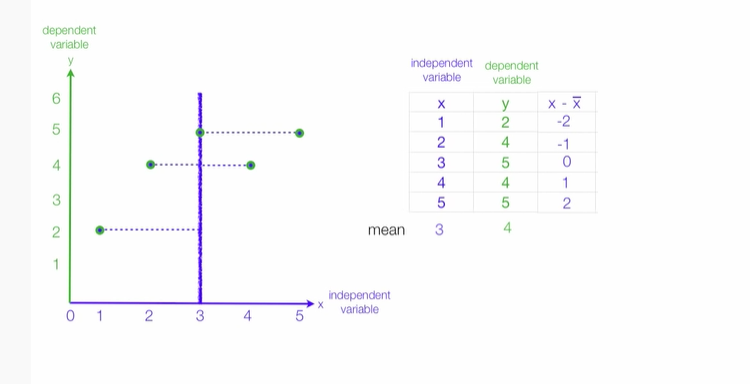
Example,

Green line 🡪 Y-mean, Blue line 🡪 X-mean, (3,4) 🡪 Intersecting point.



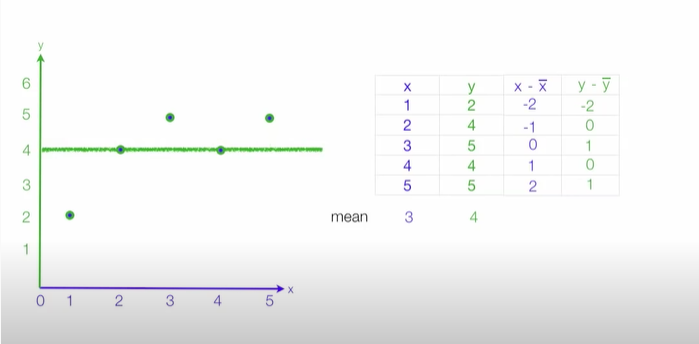
**Step # 2**

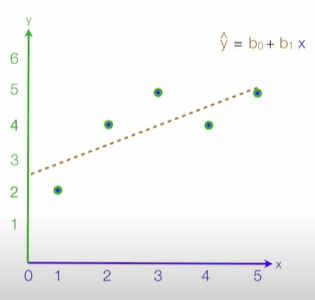
Subtract mean of X from each X co-ordinate to bring the points onto the predicting line with respect to X co-ordinates. Blue line 🡪 x = 3 and subtracting 3(mean) from each point will actually bring the points above the predicting line.

****

Repeat Step # 2 but with respect to Y co-ordinate of each point.

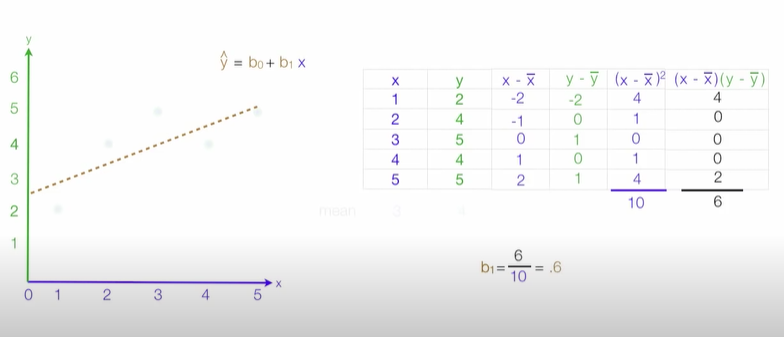
Green line 🡪 y = 4 subtracting 4(mean) from each point will actually bring the points above the predicting line.



We did this for both X and Y values of the points and for both the mean lines (X-mean and Y-mean line) because the prediction line is neither Y=4 nor X=3 however it is a line in between them n cross direction which is y= b0 + b1 and can be seen in dotted notation from the image given below,  


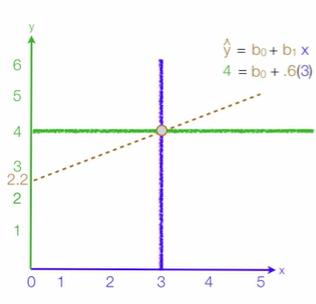
**Step # 3**

Since our line slope is always with the dependent variable therefore, we will find the summation of (x-x’) ^2 which is 10 in our case and then will find the summation of product of (x-x’) and (y-y’) because it will help us in calculating the slope(b1) of our predicting line. We divided 6 by 10 because our equation was y = b0 + b1x and since b1 is our slope and slope b1 is along ‘x’ therefore we divided the product summation by 10.



**Step # 4**

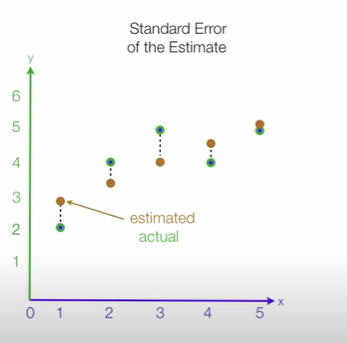
Calculating the Y intercept of our required prediction line by putting Y = 4 and X = 3 because our predicting line always contains the intersection point of the X-mean and the Y-mean lines which is (3,4) and slope b1 is already calculated in Step # 3.



Y-intercept = b0 = 2.2

Therefore, we now have got our prediction line / hypothesis which looks much better than the randomly selected prediction lines. If we follow this method, we will be getting less errors and our model would be more efficient.



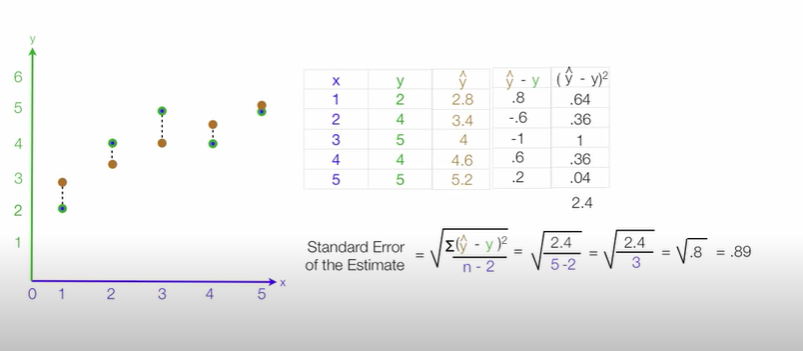


Estimated 🡪 Data points in Regression predicting line.

Actual 🡪 Data points in actual.

Standard Estimated Error **OR** Output of cost function 🡪 Distance between estimated and actual.

**Example for calculation of Standard Estimated Error**

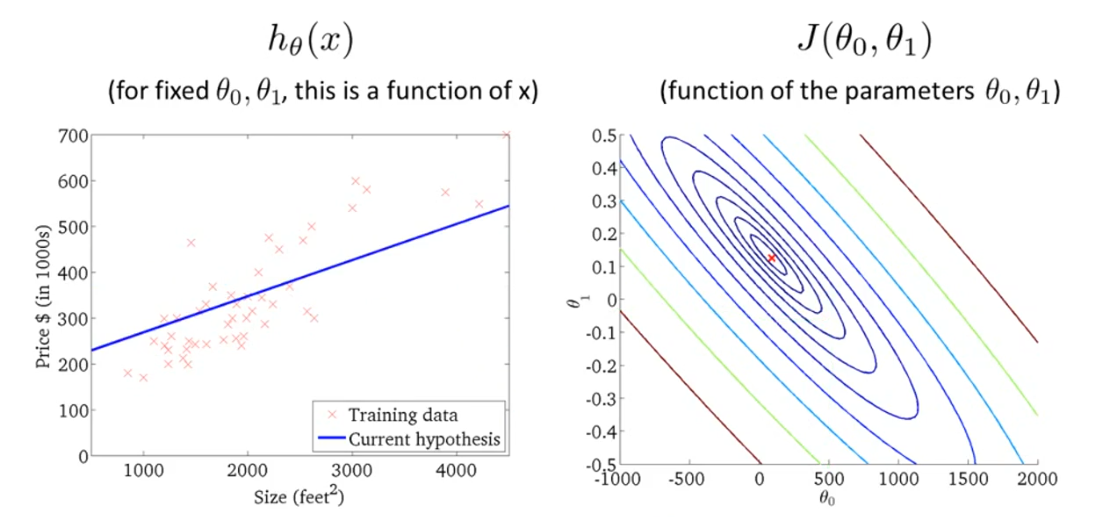


This Standard Estimated Error tells that the best possible fit line still has this much amount of error.

So far what we have done is, we took a cost function and we assumed that θ0 = 0 which means all our possible hypothesis or the prediction lines will be passing from the origin. We took these assumptions to make a simple 2D cost function graph to find the minimal cost hypothesis. However, in real world, we don’t work with just 2 variable cost function but our cost function could be of thousands and millions of variables or dimensions like ‘θ0, θ01, θ02 ……., θ0n’ and it is not easy to find a best fit or minimal cost hypothesis when we will be having millions of variables and dimensions. We will now be working with 3 variables or a 3-dimensional cost function and similarly the method will be same for >3 variables or >3 dimensions.

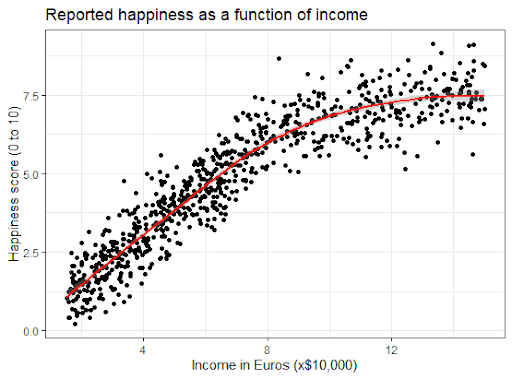
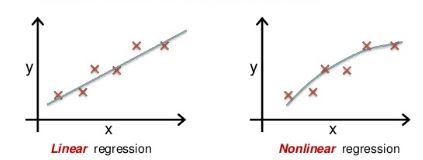
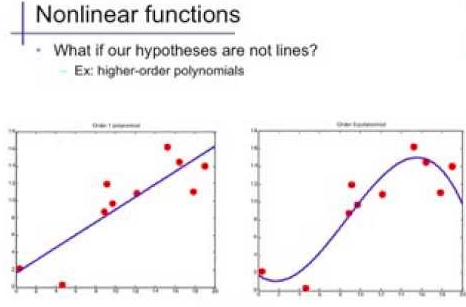
With 3 variables J (θ0, θ1), θ0 and θ1 and the cost function graph looks like a bowl,





Treating Y-intercept constant as variable in our cost function graph because the predicting lines or hypothesis function can have any Y-intercept value it is not fixed or it is not necessary that all our predicting lines or hypothesis must pass from the origin or some fixed constant of Y-intercept and that is the reason we are treating it as variable instead of a constant. We have 3 variables in this graph,

we were working with linear regression and that is why our hypothesis function was a straight line. However, a hypothesis function can be of any type and not a straight line every time it can be any curve.



**Gradient Descent Algorithm**

