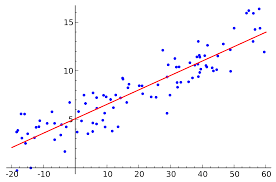
**Linear Regression**

**Simple Linear Regression:**

**Linear regression** is a technique used to model the relationships between observed variables. The idea behind simple linear regression is to "fit" the observations of two variables into a linear relationship between them. Graphically, the task is to draw the line that is "best-fitting" or "closest" to the points (x\_i,y\_i),(*xi*​,*yi*​), where x\_i*xi*​ and y\_i*yi*​ are observations of the two variables which are expected to depend linearly on each other.



*The best-fitting linear relationship between the variables xx and yy.*[*[1]*](https://brilliant.org/wiki/linear-regression/#citation-1)

Regression is a common process used in many applications of statistics in the real world. There are two main types of applications:

* **Predictions:** After a series of observations of variables, regression analysis gives a statistical model for the relationship between the variables. This model can be used to generate predictions: given two variables x*x* and y,*y*, the model can predict values of y*y* given future observations of x.*x*. This idea is used to predict variables in countless situations, e.g. the outcome of political elections, the behavior of the stock market, or the performance of a professional athlete.
* **Correlation:** The model given by a regression analysis will often fit some kinds of data better than others. This can be used to analyze correlations between variables and to refine a statistical model to incorporate further inputs: if the model describes certain subsets of the data points very well, but is a poor predictor for other data points, it can be instructive to examine the differences between the different types of data points for a possible explanation. This type of application is common in scientific tests, e.g. of the effects of a proposed drug on the patients in a controlled study.

**General Formula:**

  y= *mx*+*b*

where y=instance, output, x= predictor, input

m=slope of the regression line

b=intercept of the line

*m= n(Σxy) - (Σx)(Σy) /n(Σx^2) - (Σx)^2*

*b=(Σy)(Σx^2) - (Σx)(Σxy)/ n(Σx^2) - (Σx)^2*

**Reference links:**

For more details click on the below link

<https://www.wallstreetmojo.com/regression-formula/>

From the example of the above link: we discuss the training and testing the data.

**the regression line Y = 0.52 + 1.20 \* X**

**intercept = 0.52**

**slope = 1.20**

So, for n=5, we will consider this as training data=5

And for the testing data or actual data we will take it indexing number 6 which is 259.95

So, here we manually predicted the number 6 by applying x value in the Y

***Y= 0.52+1.20\*259.95***

***Which is 312.46***

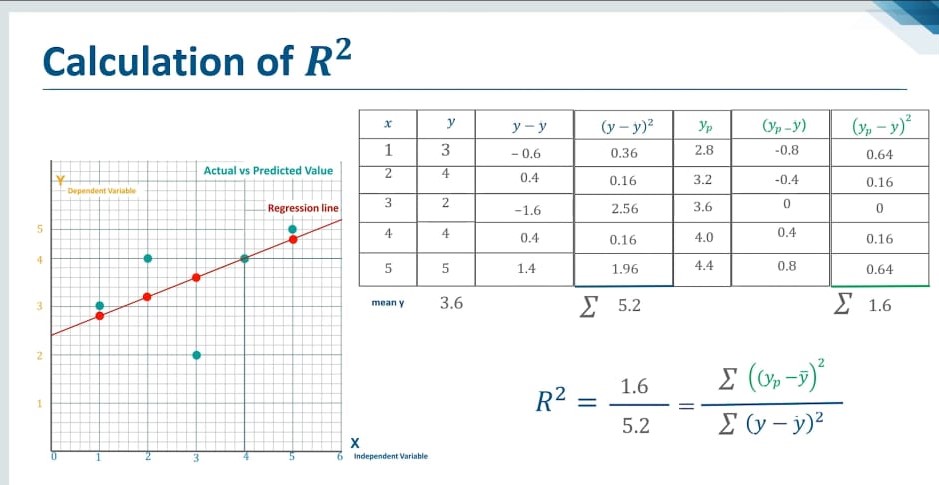
***But our Actual Value or testing value is 314.17***

**R-Squared:**

R-squared value is a statistical measure of how close the data are to the fitted regression line.

It is also known as **coefficient of determination,** or the **coefficient of multiple determination.**

**Example:**



**Linear Regression in Machine Learning in R**

**Our Goal:**

All we are trying to do when we calculate our regression line is draw a line that’s as close to every do as possible.

For a classic linear regression, or “Least Square Method”, you only measure the closeness in the “up and down” direction.

Now wouldn’t it be great if we could apply this same concept to a graph with more than just two data points?

By doing this, we could take multiple men and their son’s heights and do things like tell a man how tall we expect his son to be. Before he even has a son!

Our goal with linear regression is to **minimize the vertical distance** between all the data points and our line.

So, in determining **the best line**, we attempting to minimize the distance between **all** the points and their distance to our line.

There are lots of different ways to minimize this, (sum of squared errors, sum of the absolute errors, etc.), but all these methods have a general goal of minimizing this distance.

**Formulas:**

Formulas in R take the form (y ~ x). To add more predictor variables, use the sign. i.e (y~ x+z)

**Model<- lm(log(PINCP,base=10) ~ AGEP+SEX+COW+SCHL,data=dtrain)**

Mode= R object to save result in

Lm= Linear regression modelling command

Formula

log(PINCP,base=10)=Quantity we want to predict

AGEP+SEX+COW+SCHL=variables available to make prediction

data=dtrain= Dataframe to use for training

**dtest$predLogPINCP <- predict( model,newdata=dtrain)**

**dtest$predLogPINCP= Store the prediction as a new column named “predLogPINCP”**

**predict= Prediction command**

**model= linear regression model**

**dtest= Data to use in prediction**

**dtrain$predLogPINCP<- predict(model,newdata=dtrain)**

**Estimating the coefficients**

*β*0 and *β*1 are unknown. So before we can use (3.1) to make

predictions, we must use data to estimate the coefficients. Let

(*x*1*, y*1)*,* (*x*2*, y*2)*, . . . ,* (*xn, yn*)

represent *n* observation pairs, each of which consists of a measurement

of *X* and a measurement of *Y*

In other words, we want to find an intercept ˆ β0 and a slope ˆ β1 such that the resulting line is as close as possible to the n.

There are number of ways of measuring closness… the most common approach involves minimizing the **Least square method**.

mse<-mean((results$real- results$pred)^2)

**R-Square root of MSE**

(mse^0.5)

**R^2**

*Square error=SSE, total sum error=SST*

sse<-sum( (results$pred-results$real)^2)--🡪 sse=sum( Ẏ -Y )^2

sst<-sum( (mean(df$G3)-results$real)^2)-🡪 sst=sum( mean( Y )-Ytest)^2

r2<-1-sse/sst