

1.10 APPLICATIONS OF NumPy ARRAYS

Broadly, statistics is concerned with collecting and analyzing data. It seeks to describe methods for collecting data (samples), for analysing the data, and for inferring conclusions from the data. NumPy is a popular library of Python that provides useful built-in tools and standard statistical functions to compute for finding minimum, maximum, percentile standard deviation and variance, correlation, etc., from the given elements in the array.

In this section, we are going to discuss three important applications of NumPy Arrays: in calculating covariance, correlation and linear regression.

1.10.1 Covariance

The three terms which are most extensively used in statistics and analysis of a dataset and are inter-related are variance, covariance and correlation. However, there are a few differences between them which are as follows:

- Variance is a measure of variability from the mean.
- Covariance is a measure of relationship between the variability (the variance) of 2 variables.
- Correlation/Correlation coefficient defines statistical relationships between two datasets and their interdependence.

➤ Variance:

Before we get started, we shall take a quick look at the difference between covariance and variance. Variance measures the variation of a single random variable (like the height of a person in a population), whereas covariance is a measure of how much two random variables vary together (like the height of a person and the weight of a person in a population). The formula for variance is given by:

$$\text{Variance} = (x_n - \bar{x})^2/n$$

where, x_n = value point, \bar{x} = mean, and n = number of observations.

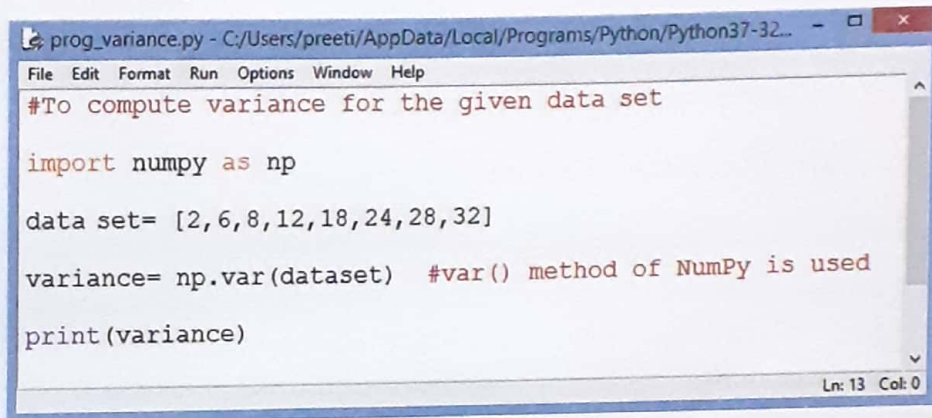
To compute the variance, we use the numpy module. Variance measures how far a set of (random) numbers are spread out from their average value. In Python, we can calculate the variance using the numpy module. With NumPy, the var() function calculates the variance for a given dataset.

In the code below, we show how to calculate the variance for a dataset.

Practical Implementation-19

To compute the variance for the given dataset using NumPy.

[2, 6, 8, 12, 18, 24, 28, 32]



```
prog_variance.py - C:/Users/preeti/AppData/Local/Programs/Python/Python37-32...
File Edit Format Run Options Window Help
#To compute variance for the given data set

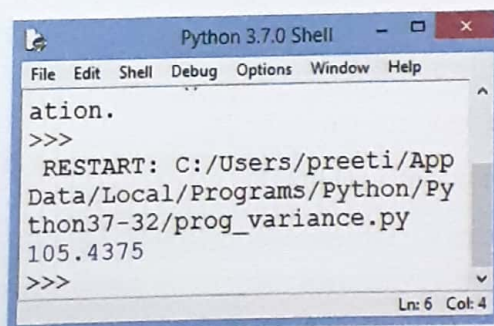
import numpy as np

data set= [2,6,8,12,18,24,28,32]

variance= np.var(dataset) #var() method of NumPy is used

print(variance)

Ln: 13 Col: 0
```



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
ation.
>>>
RESTART: C:/Users/preeti/App
Data/Local/Programs/Python/Py
thon37-32/prog_variance.py
105.4375
>>>

Ln: 6 Col: 4
```

The variance calculated in the above code through NumPy var() functions can be described in terms of the following steps:

- We import the NumPy module as np. This means that we reference the numpy module with the keyword, np.
- We then create a variable, dataset, which is equal to [2,6,8,12,18,24,28,32]
- We then get the variance of this dataset by using the np.var (dataset) function.
- We then display the variance using print() statement, which in this case is 105.4375.

So let's go over the formula for variance to see if the value calculated is correct.

The formula for variance is $(x_n - \bar{x})^2/n$

This means that in order to calculate the standard deviation, we must first calculate the mean of the dataset. The mean in this case is $(2+6+8+12+18+24+28+32)/8 = 130/8 = 16.25$.

We now take each x value and minus 16.25 from it.

This gives us, $(2-16.25) = -14.25$; $(6-16.25) = -10.25$; $(8-16.25) = -8.25$; $(12-16.25) = -4.25$; $(18-16.25) = 1.75$; $(24-16.25) = 7.75$; $(28-16.25) = 11.75$; $(32-16.25) = 15.75$.

We then square all these numbers to get $-14.25^2 + -10.25^2 + -8.25^2 + -4.25^2 + 1.75^2 + 7.75^2 + 11.75^2 + 15.75^2 = 203.0625 + 105.0625 + 68.0625 + 18.0625 + 3.0625 + 60.0625 + 138.0625 + 248.0625 = 843.5$

We now take this value and divide it by n.

This gives us $843.5/8 = 105.4375$

This value of 105.4375 is the variance. And this is how we compute the variance of a dataset in Python using the NumPy module.

Now, coming to the concept of covariance, **Covariance** is a statistical measure that shows whether two variables are related by measuring how the variables change in relation to each other. It's similar to variance but where variance tells you how a single variable varies, covariance tells you how two variables vary together.

The formula for calculating covariance is:

$$\text{Cov}(X,Y) = \sum_{i,j=1}^n \frac{(X_i - \bar{X})(Y_j - \bar{Y})}{n-1}$$

where,

X = the independent variable

Y = the dependent variable

n = number of data points in the sample

\bar{X} (bar) = the mean of the independent variable X

\bar{Y} (bar) = the mean of the dependent variable Y

A positive covariance means the variables are positively related or we say that they are very strongly similar, while a negative covariance means the variables are inversely related, *i.e.*, they are very dissimilar.

CTM: Covariance is a measure of relationship between 2 variables. It measures the degree of change in the variables, *i.e.*, when one variable changes, there will be same/a similar change in the other variable. A positive covariance indicates that when one variable increases, the second also increases and when one variable decreases, the other also decreases. On the other hand, if the covariance is negative, it means that an increase in one will cause a decrease in the other.

NumPy does not have a function to calculate the covariance between two variables directly. Instead, it has a function for calculating a covariance matrix called `cov()` that we can use to retrieve the covariance. By default, the `cov()` function will calculate the unbiased or sample covariance between the provided random variables.

Practical Implementation-20

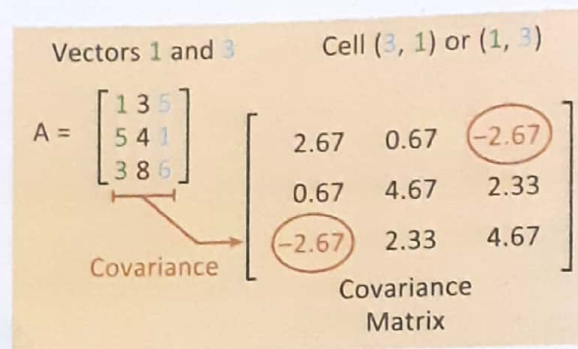
Calculate covariance between two given arrays and display the output both for covariance matrix as well as covariance value.

In case of three datasets or arrays as arguments, the resultant covariance matrix shall be represented as:

$$C = \begin{bmatrix} \text{var}(x) & \text{cov}(y,x) & \text{cov}(z,x) \\ \text{cov}(x,y) & \text{var}(y) & \text{cov}(z,y) \\ \text{cov}(x,z) & \text{cov}(y,z) & \text{var}(z) \end{bmatrix}$$

In other words, if the three variables are termed as x , y , and z , then the covariance matrix element C_{ij} is the covariance between x_i and y_j .

For instance, the covariance between the first and the third column is located in the covariance matrix as column 1 and row 3 (or column 3 and row 1).



The position in the covariance matrix. Column corresponds to the first variable and row to the second (or the opposite). The covariance between the first and the third column of matrix 'A' is the element in column 1 and row 3 (or the opposite = same value).

Consider the matrix of 5 observations each of 3 variables, x_0 , x_1 and x_2 whose observed values are held in the three rows of the array X:

```
X = np.array([ [0.1, 0.3, 0.4, 0.8, 0.9],
               [3.2, 2.4, 2.4, 0.1, 5.5],
               [10., 8.2, 4.3, 2.6, 0.9]
             ])
```

The covariance matrix is a 3×3 array of values,

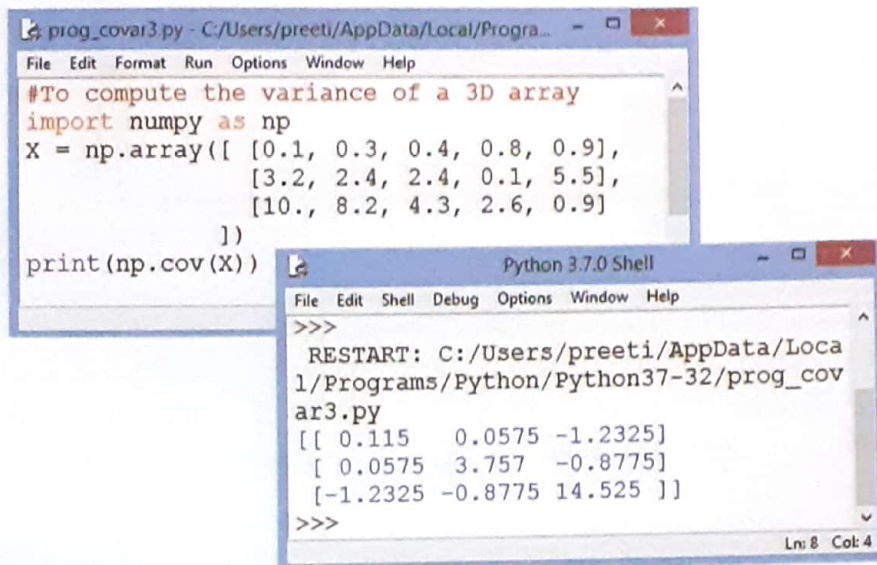
```
In [x]: print(np.cov(X))
```

```
[ [ 0.115,  0.0575, -1.2325]
  [ 0.0575,  3.757,  -0.8775]
  [-1.2325, -0.8775, 14.525]]
```

The diagonal elements $C_{ij}C_{ij}$ are the variances in the variables x_i , where x is $n-1$ as shown in Practical Implementation-21.

Practical Implementation-21

To compute variance of a 3D array 'X'.



```
prog_covar3.py - C:/Users/preeti/AppData/Local/Progra...
File Edit Format Run Options Window Help
#To compute the variance of a 3D array
import numpy as np
X = np.array([ [0.1, 0.3, 0.4, 0.8, 0.9],
               [3.2, 2.4, 2.4, 0.1, 5.5],
               [10., 8.2, 4.3, 2.6, 0.9]
             ])
print(np.cov(X))

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
>>>
RESTART: C:/Users/preeti/AppData/Local/Programs/Python/Python37-32/prog_covar3.py
[[ 0.115  0.0575 -1.2325]
 [ 0.0575  3.757  -0.8775]
 [-1.2325 -0.8775 14.525 ]]
>>>
```

Practical Implementation-22

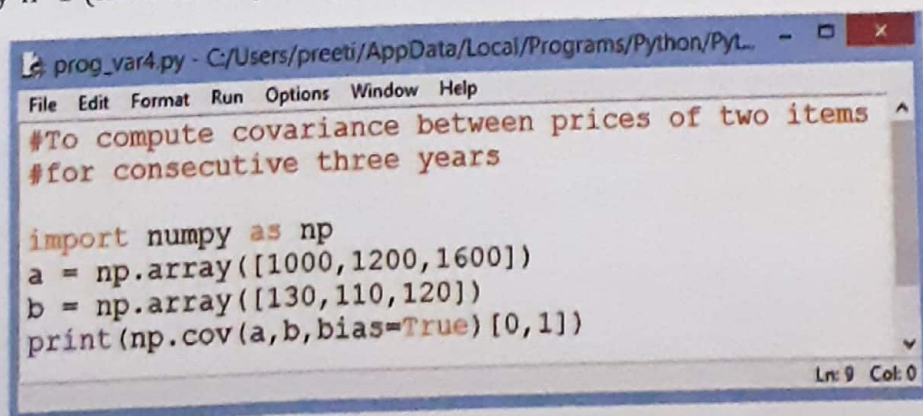
To compute variance of prices of two given items for three successive years.

YEAR	ITEM 1 PRICE	ITEM 2 PRICE
2015	1000	130
2016	1200	110
2017	1600	120

The covariance for the above data can be calculated as:

YEAR	ITEM 1 PRICE	ITEM 2 PRICE	VARIANCE A	VARIANCE B	A × B
2015	1000	130	-266.667	10	-2666.67
2016	1200	110	-66.6667	-10	666.6667
2017	1600	120	333.3333	10	3333.333
	1266.666667	120			1333.333
				covariance =	666.6667

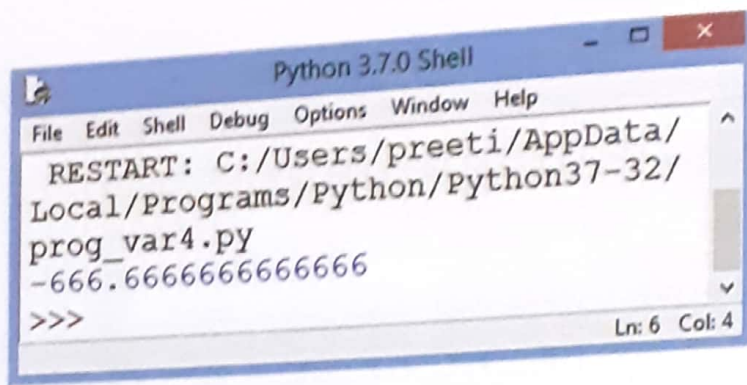
In the above calculation, value of A is calculated by subtracting item1 price from item1 mean price. Same is to be done for B and for all its values. Covariance is then calculated by dividing sum of AXB by n-1 (number of years). This can be implemented using NumPy as:



```
prog_var4.py - C:/Users/preeti/AppData/Local/Programs/Python/PyL...
File Edit Format Run Options Window Help
#To compute covariance between prices of two items
#for consecutive three years

import numpy as np
a = np.array([1000,1200,1600])
b = np.array([130,110,120])
print(np.cov(a,b,bias=True)[0,1])

Ln: 9 Col: 0
```

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
RESTART: C:/Users/preeti/AppData/
Local/Programs/Python/Python37-32/
prog_var4.py
-666.66666666666666
>>>
Ln: 6 Col: 4
```

1.10.2 Correlation

With reference to Practical Implementation-21, we had obtained negative value for covariance as -7.5. This covariance can be normalized to a score between -1 and 1 to make the magnitude interpretable by dividing it by the standard deviation of X and Y. The result is called the correlation of the variables, also called the Pearson correlation coefficient, named after the developer of the method.

Therefore, the term “correlation” refers to a mutual relationship or association between quantities. In almost any business, it is useful to express one quantity in terms of its relationship with others. *For example*, sales might increase when the marketing department spends more on TV advertisements, or a customer’s average purchase amount on an e-commerce website might depend on a number of factors related to that customer. Often, correlation is the first step to understanding these relationships and subsequently building better business and statistical models. The significance of correlation can be defined as:

- Correlation can help in predicting one quantity from another.
- Correlation can indicate the presence of a casual relationship.
- Correlation is used as a basic quantity and foundation for many other modelling techniques.

Correlation is the scaled measure of covariance. Besides, it is dimensionless. In other words, the correlation coefficient is always a pure value and not measured in any units. The formula for calculating correlation is:

$$\rho(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma_x \sigma_y}$$

where,

Cov(X,Y)—the covariance between the variables X and Y

σ_x —the standard deviation of the X-variable

σ_y —the standard deviation of the Y-variable

NumPy provides the **corrcoef(<array1>,<array2>)** function for calculating the correlation matrix between two variables directly. Like cov(), it returns a matrix, in this case a correlation matrix.

Practical Implementation-23

To compute correlation between the prices of two items given in Practical Implementation-22.

```
prog_correlt1.py - C:/Users/preeti/AppData/Local/Programs/...
File Edit Format Run Options Window Help
#To compute correlation among the price
#of three items

import numpy as np
a = np.array([1000,1200,1600])
b = np.array([130,110,120])
print(np.corrcoef(a, b))

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
>>>
RESTART: C:/Users/preeti/AppData/Local/Programs/Python/Python37-32/prog_corre
lt1.py
[[ 1.          -0.32732684]
 [-0.32732684  1.          ]]
>>>
Ln: 7 Col: 4
```

CTM: Correlation can be better described as normalized covariance.

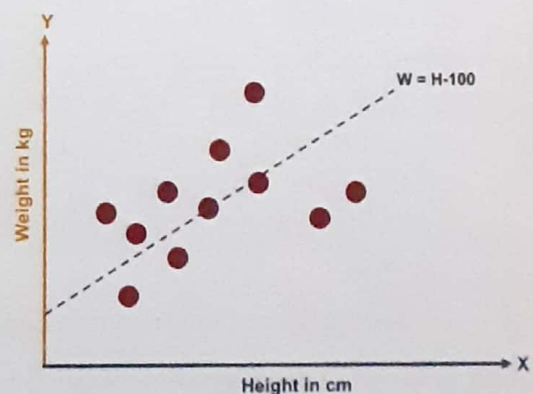
A correlation coefficient will always be between -1 and 1. The closer the value is to -1 or 1, the stronger the relationship, is, the closer to 0, the weaker it is. If the correlation coefficient value is positive, it means as one variable increases, so does the other, and if the correlation coefficient value is negative, it means as one variable increases, the other decreases.

1.10.3 Linear Regression

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable. Linear regression is a method used to find a relationship between a dependent variable and independent variable(s).

The best way to understand linear regression is to relive this experience of childhood.

Let us say, you ask a child in fifth grade to arrange children in his class in increasing order of weight, without asking them their weight! What do you think the child will do? He/she would likely look (visually analyze) at the height and build of children and arrange them using a combination of these visible parameters. This is linear regression in real life. The child has actually figured out that height and build would be correlated to the weight by a relationship, which looks like the equation below.



$$Y = aX + b$$

where,

- Y - Dependent variable
- a - Slope
- X - Independent variable
- b - Intercept

These coefficients a and b are derived based on minimizing the sum of squared difference of distance between data points and regression line.

CTM: Linear Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent and independent variable using a straight line.

Linear Regression can be broadly classified into two types:

1. **Single Linear Regression:** In single linear regression, there is only one independent variable, *e.g.*, the price of the house depends only on one field that is the size of the plot.
2. **Multiple Linear Regression:** In multiple linear regression, there is more than one independent variable, *e.g.*, the price of the house depends on more than one field that is the size of the plot and the number of rooms.

➤ Where is Linear Regression used?

Now the question arises where we use linear regression. Linear regression is used for predicting whether it is in scientific, business, or technical applications, which are described as follows:

1. **Trend Forecasting—Evaluating Trends and Sales Estimates:** Linear regressions can be used in business to evaluate trends and make estimates or forecasts. *For example*, if a company's sales have increased steadily every month in the past few years, conducting a linear analysis on the sales data with monthly sales on the y-axis and time on the x-axis would produce a line that depicts the upward trend in sales.

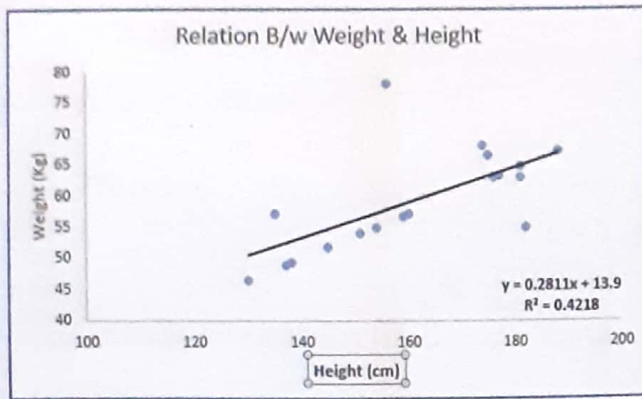


2. **Forecasting an Effect—Analyzing the Impact of Price Changes:** Linear regression can also be used to analyze the effect of pricing on consumer behaviour. *For example*, if a company changes the price of a certain product several times, it can record the quantity it sells at each price level and then perform a linear regression with quantity sold as the dependent variable and price as the explanatory variable. The result would be a line that depicts the extent to which consumers reduce their consumption of the product as prices increase, which could help guide future pricing decisions.



3. **Assessing Risk:** Linear regression can be used to analyze risk.

Look at the example given below. Here, we have identified the best fit line having linear equation $y=0.2811x+13.9$. Now, using this equation, we can find the weight, knowing the height of a person.



x	y
1	3
2	4
3	2
4	4
5	5

Let us discuss Best Fit Line method for linear regression in detail.

➤ Best Fit Line Method for Linear Regression

Linear regression is used with a continuous variable and used to predict the value of the variable as the output. Consider a linear expression for a straight line represented as:

$$y = mx + c$$

where y is a dependent variable

x is an independent variable

m is the slope of the line and

c is the coefficient of linear expression to be calculated

When we plot a graph between dependent and independent variable, a line is obtained which is termed as Regression line as shown in Fig. 1.6(a).

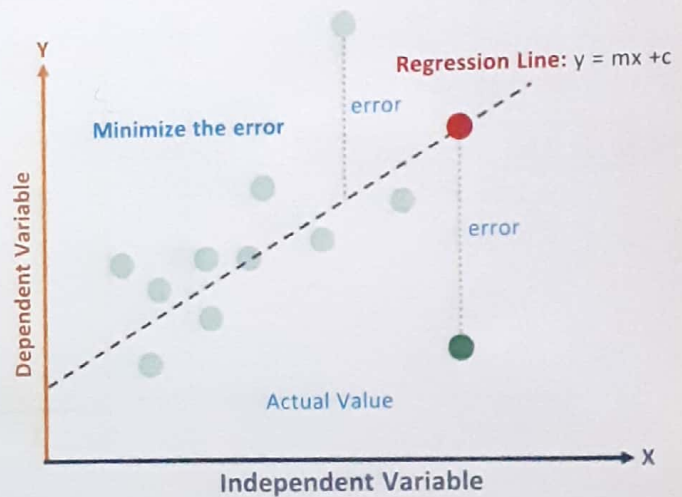


Fig. 1.6(a): Regression Line, $y = mx + c$

Let's make some data points which are also termed as observations on the plot area as shown in Fig. 1.6(b).

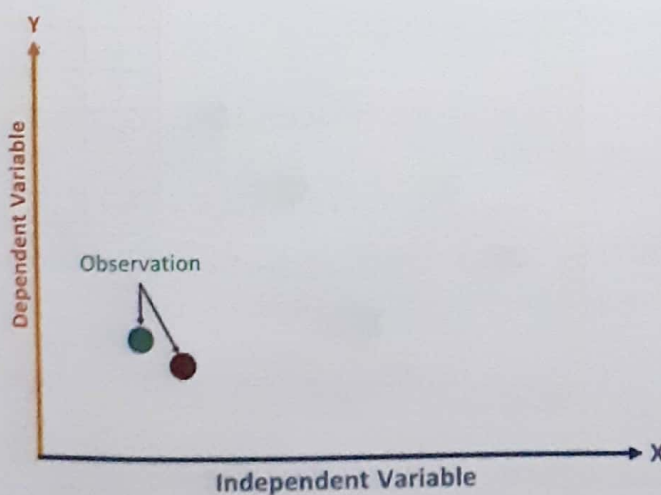


Fig. 1.6(b): Data Points or Observations

After all data points are plotted, *i.e.*, the line is drawn, the next step involves prediction to be made. With reference to Fig. 1.6(a), there is an estimated value or predicted value, and there is an actual value. The distance between the estimated value and the actual value is the error. So, our objective is to reduce this error, *i.e.*, we have to reduce this distance between the estimated value and the actual value. This is what linear regression is all about.

Thus, the best fit line will be the one having least error or least difference between the actual and estimated value. Now, we will start plotting the points on the graph, as shown in Fig. 1.6(c).

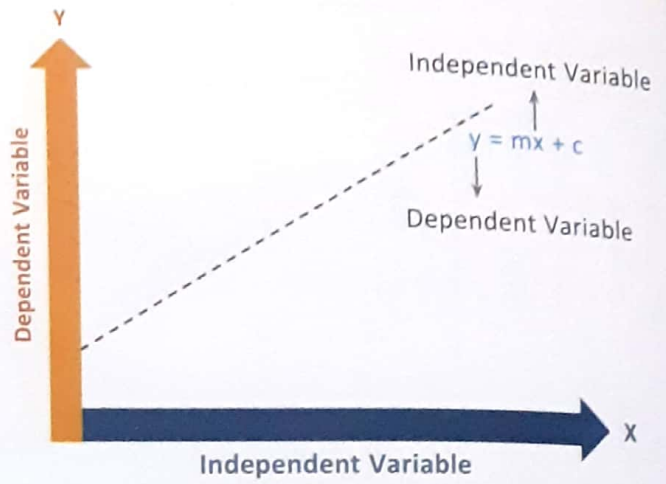
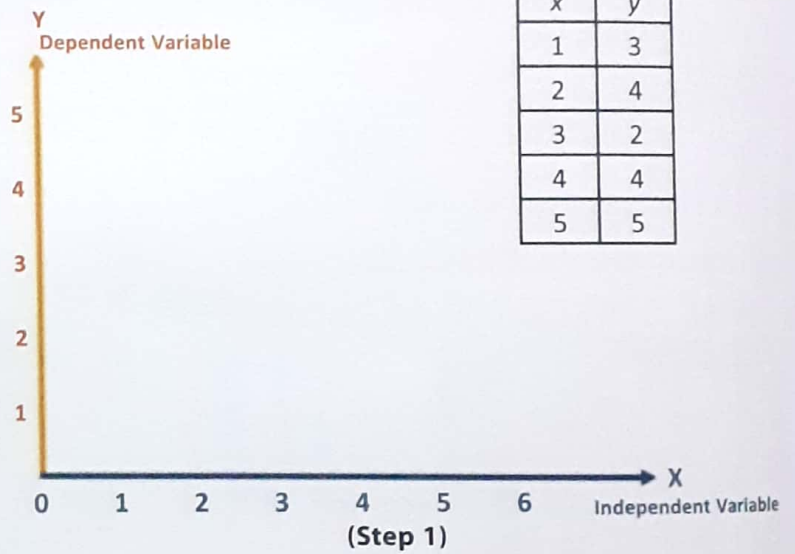
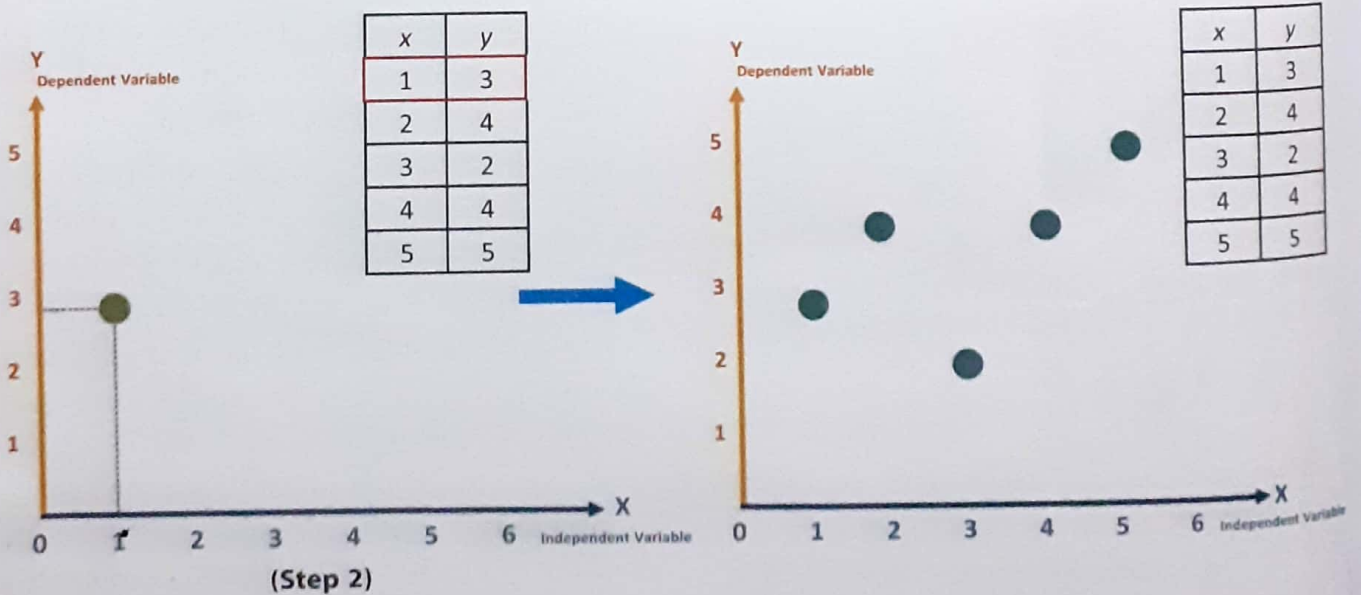


Fig. 1.6(c): Dependent vs Independent Variables

Step 1: The first step is to take x-axis and y-axis as per the datasets.



Step 2: Next step is to plot the first point, viz. (1,3) as shown.

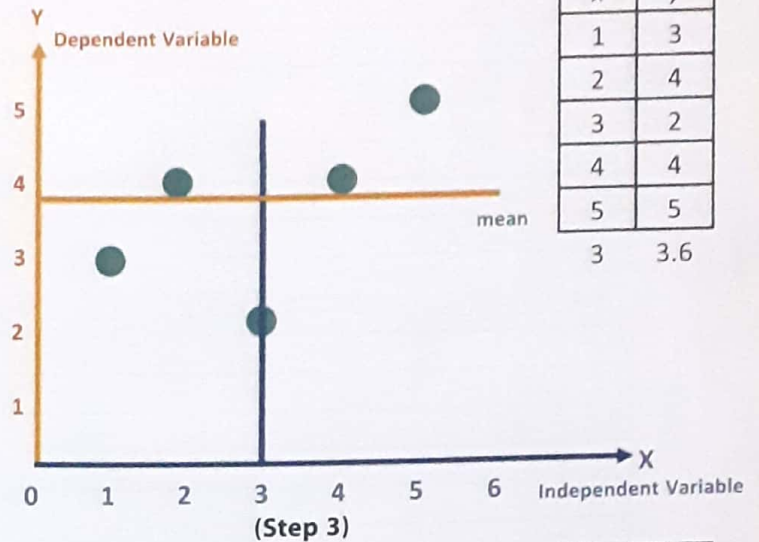


This is how we will plot rest of the points.

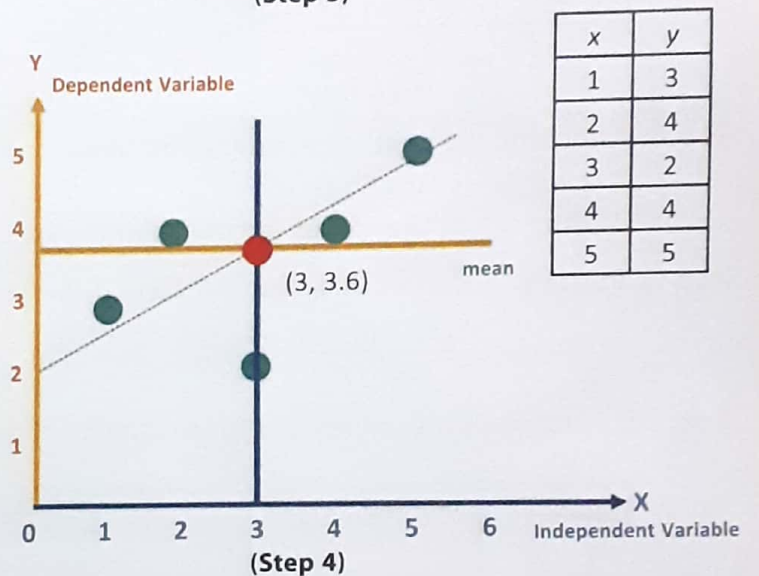
Step 3: Our next step is to calculate mean value for 'x' and 'y' respectively. The mean value for x will be:

$$x' \text{ or } \bar{x} (x\text{-bar}) = 1+2+3+4+5/5 = 15/5 = 3, \text{ similarly for mean of y,}$$

$$y' \text{ or } \bar{y} (y\text{-bar}) = 3+4+2+4+5/5 = 18/5 = 3.6$$



Step 4: Now we will plot this point holding mean of x and y on the graph, (3, 3.6), indicated by red point in the figure.



After we have plotted the mean value, what we have to do is to find line of regression, $y = mx + c$. For this we have to find the value of 'm', i.e., slope of the line given by the formula

$$m = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$

Step 5: As per the formula, $(x - x')$ is nothing but the distance of all the points to the line $y = 3$ and $(y - y')$ indicate distance of all the points to the line $x = 3.6$. Let us calculate $(x - x')$ and $(y - y')$

For $(x - x')$, the first value will be -2 (1-3) where $x = 1$ and x' (mean) = 3, we have already created in the previous step.

x	y	$x - \bar{x}$
1	3	-2
2	4	-1
3	2	0
4	4	1
5	5	2

(Step 5)

Step 6: Similarly $(y - y')$ is calculated as:

$(3 - 3.6)$ which will be equal to -0.6.

This will be done for all the values of y.

x	y	$x - \bar{x}$	$y - \bar{y}$
1	3	-2	-0.6
2	4	-1	-0.4
3	2	0	-1.6
4	4	1	0.4
5	5	2	1.4

mean 3 3.6 mean 3 3.6

(Step 6)

Step-7: As per the formula for calculating 'm', we have to calculate $(x - \bar{x})^2$

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$
1	3	-2	-0.6	4
2	4	-1	0.4	1
3	2	0	-1.6	0
4	4	1	0.4	1
5	5	2	1.4	4

Then we have to calculate $(x - \bar{x}) * (y - \bar{y})$

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x}) (y - \bar{y})$
1	3	-2	-0.6	4	1.2
2	4	-1	0.4	1	-0.4
3	2	0	-1.6	0	0
4	4	1	0.4	1	0.4
5	5	2	1.4	4	2.8
				$\Sigma = 10$	$\Sigma = 4$

(Step 7)

Step 8: After this the next step is to calculate

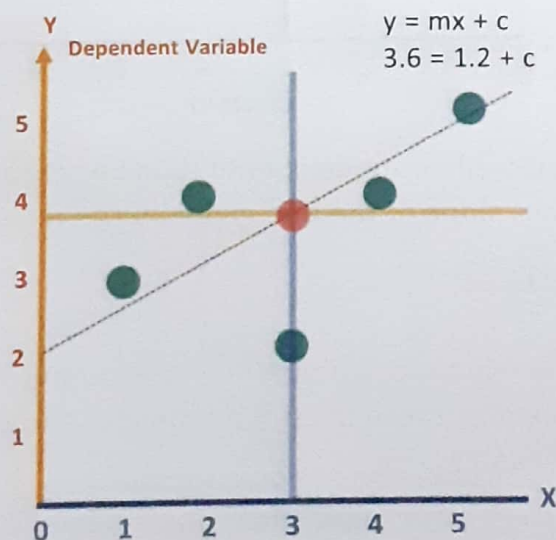
$\Sigma (x - \bar{x}) (y - \bar{y})$ and $\Sigma (x - \bar{x})^2$

Adding up all the values for the columns $(x - \bar{x})^2$ and $(x - \bar{x})(y - \bar{y})$

Step 9: Putting all these calculated values into the formula,

$$m = \frac{\Sigma(x - \bar{x})(y - \bar{y})}{\Sigma(x - \bar{x})^2} = \frac{4}{10} \text{ the value of } m \text{ comes out to be } 0.4.$$

Step 10: Putting all the values, we get the line of regression as $3.6 = 1.2 + c$.



Therefore, we obtain line of regression as:

$$m = 0.4$$

$$c = 2.4$$

$$y = 0.4x + 2.4$$

For the given values of $m = 0.4$ and $c = 2.4$, let's predict values for y'' with respect to $x = \{1, 2, 3, 4, 5\}$

$$y = 0.4 \times 1 + 2.4 = 2.8$$

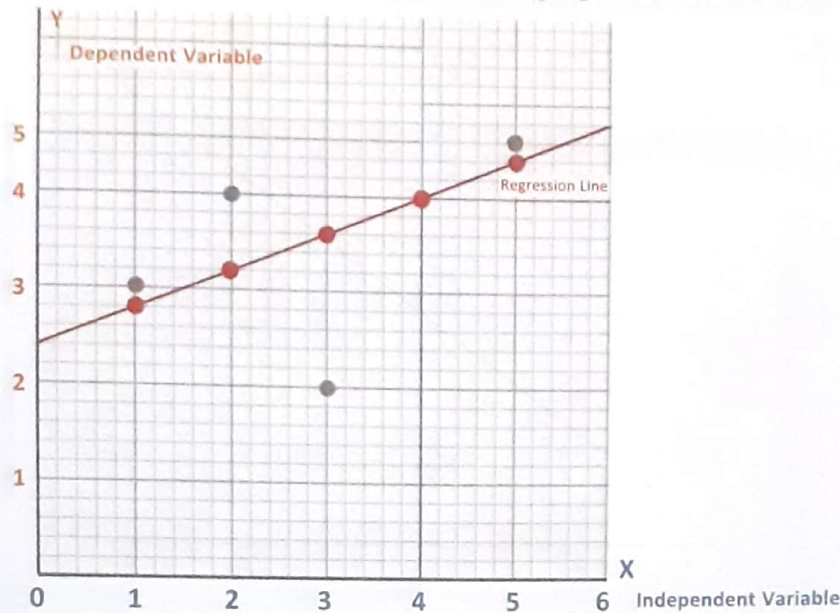
$$y = 0.4 \times 2 + 2.4 = 3.2$$

$$y = 0.4 \times 3 + 2.4 = 3.6$$

$$y = 0.4 \times 4 + 2.4 = 4.0$$

$$y = 0.4 \times 5 + 2.4 = 4.4$$

So, after predicting the values, (x,y) are plotted on the graph to obtain the line of Regression.



After understanding the concept of linear regression using best fit line method, we will now implement it through NumPy.

Practical Implementation-24

To plot linear regression for a linear equation given as

$$\text{Linear Equation: } Y = aX + b$$

Here,

a: Slope of the line

b: Constant (Y-intercept, where X=0)

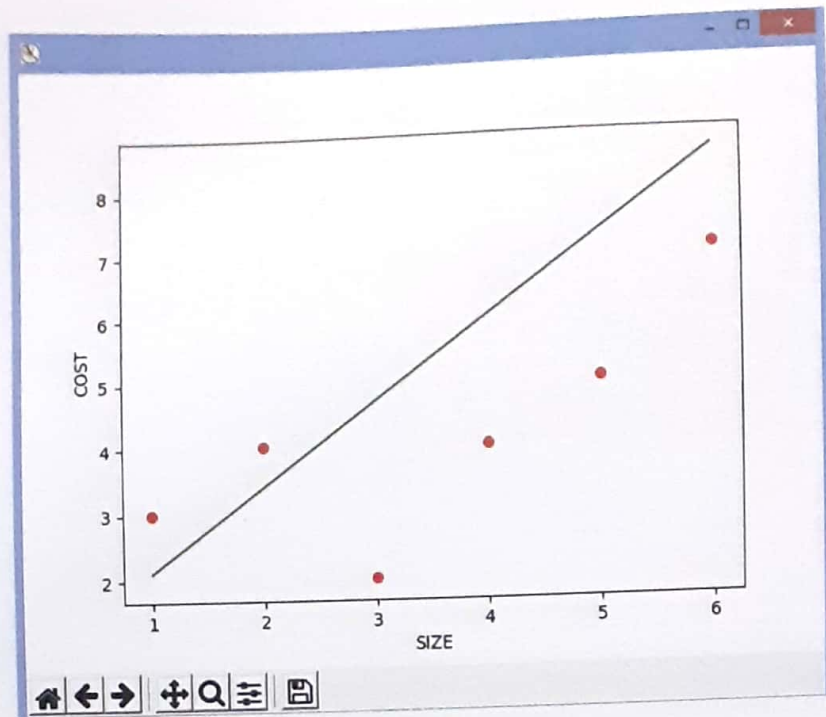
X: Independent variable

Y: Dependent variable

```
prog_alternte_linreg.py - C:\Users\preeti\AppData\Local\Programs\Python\Python37-32\prog_alternte_linreg.py (3.7.0)
File Edit Format Run Options Window Help
#Program for linear regression using BEST FIT LINE method

import numpy as np
import matplotlib.pyplot as plt
def estcoefficients(x,y):
    n=np.size(x)
    meanx,meany=np.mean(x),np.mean(y)
    sy=np.sum(y*x-n*meany*meanx)
    sx=np.sum(x*x-n*meanx*meanx)
    a=sx/sy
    b=meany-a*meanx
    return(a,b)
def plotregline(x,y,b):
    plt.scatter(x,y,color="r",marker="o",s=30)
    ypred=b[0]+b[1]*x
    plt.plot(x,ypred,color="g")
    plt.xlabel('SIZE')
    plt.ylabel('COST')
    plt.show()
x=np.array([1,2,3,4,5,6])#independentvariable
y=np.array([3,4,2,4,5,7])#dependentvariable
b=estcoefficients(x,y)
plotregline(x,y,b)
```

Ln 26 Col 0



Plot for the line:

$$Y = aX + b$$

It can be observed from the output window that performing linear regression is the same as fitting a scatter plot to a line.

➤ **polyfit() Method for Linear Regression**

We have already seen how to fit a given set of points minimizing an error function. Now we will see how to find a fitting polynomial for the data using the function `polyfit` provided by NumPy. For finding the linear regression, you can also use an alternative method of `polyfit()`, which produces least squares polynomial fit. This method accepts the data set and a polynomial function of any degree (specified by the user), and returns an array of coefficients that minimizes the squared error. For simple linear regression, one can choose degree 1. If you want to fit a model of higher degree, you can construct polynomial features out of the linear feature data and fit to the model too.

The **polyfit()** method fits a polynomial $p(x) = p[0] * x^{deg} + \dots + p[deg]$ of degree 'deg' to points (x,y). The syntax for `polyfit()` is:

```
numpy.polyfit(x, y, deg)
```

where,

- x is an array with shape like 'M', containing x-coordinates of the M sample points $(x[i], y[i])$.
- y is an array having the same shape as x and contains the y-coordinates of the sample points.
- deg int; it specifies degree of the fitting polynomial, which by default is 3.

The `polyfit()` method returns a vector of coefficients 'p' that minimizes the squared error in the order deg, deg-1, ..., 0. This is an array of size 2 with the values $\{p(0), p(1)\}$.

Several data sets of sample points having the same x-coordinates can be fitted at once by passing in a 2D-array that contains one data set per column. Let us implement this method using NumPy.

Practical Implementation-25

To plot linear regression for a linear equation using polyfit() method to generate vector of coefficients for given x and y arrays.

```
prog_poly1.py - C:\Users\preeti\AppData\Local\Programs\Python\Python37-32\prog_p...
File Edit Format Run Options Window Help
#To implement linear regression using polyfit() method
import numpy as np
import matplotlib.pyplot as plt

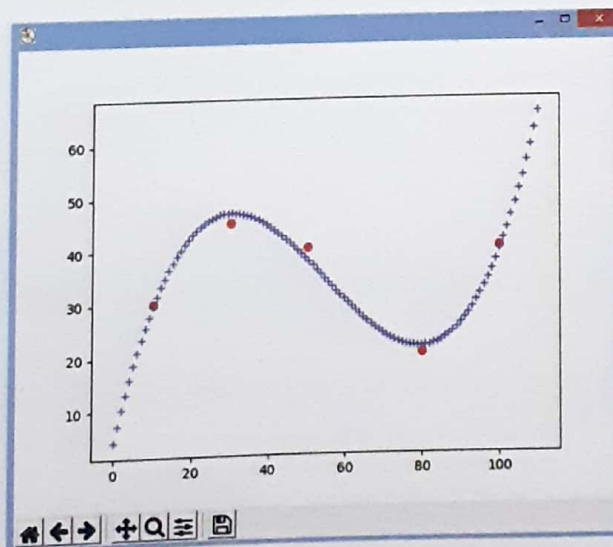
x = np.array([10, 30, 50, 80, 100])
y = np.array([30, 45, 40, 20, 40])
for x1, y1 in zip(x, y):
    plt.plot(x1, y1, 'ro')

z = np.polyfit(x, y, 3)
f = np.polyld(z)

for x1 in np.linspace(0, 110, 110):
    plt.plot(x1, f(x1), 'b+')

print(f(110)) #Passing the value of x coordinate
print(f(60)) #Passing the value of y coordinate
plt.show()
```

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\preeti\AppData\Local\Programs\Python\Python37-32\prog_poly1.py
65.57142857142918
29.98496240601496
Ln: 5 Col: 0
```



Note: For detailed description of Pyplot and the methodology to generate different charts/plots, refer to Chapter 2 (Data Visualization Using Pyplot)



MEMORY BYTES

- A NumPy array is simply a grid that contains values of the same/homogeneous type.
- NumPy arrays are also referred to as ndarrays.
- NumPy (Numerical Python) is a linear algebra library in Python.
- NumPy is very useful for performing mathematical and logical operations on arrays.