- 1.1 Linear Regression
- 1.2.what is Regression
- 1.3 What is best fit line
- 1.4.Mathematical model of Linear Regression
- 1.5.Performing step by step mathematical problem of Linear Regression
- 1.6 Create and predict model using Sklearn library, Most commonly used in ML.

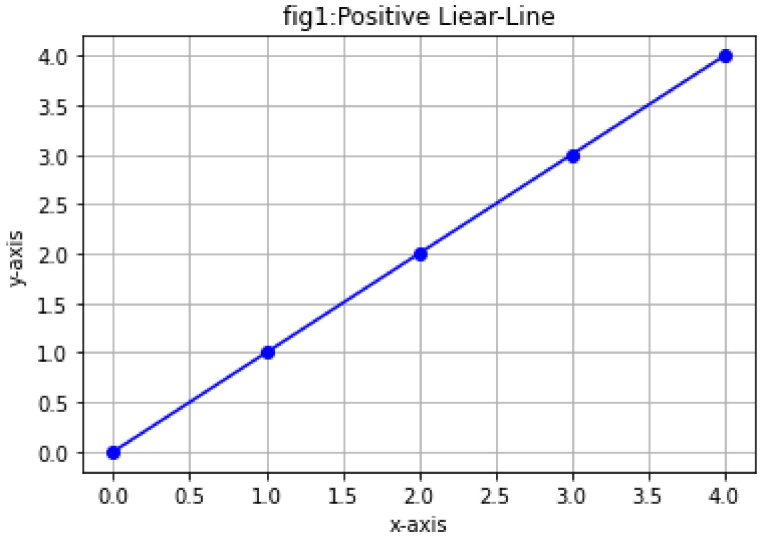
1.1 Linear Regression:

1. what is Linear or linearity:

- Linearity is the property of mathametics relationship that can be graphically represented as a straight line.

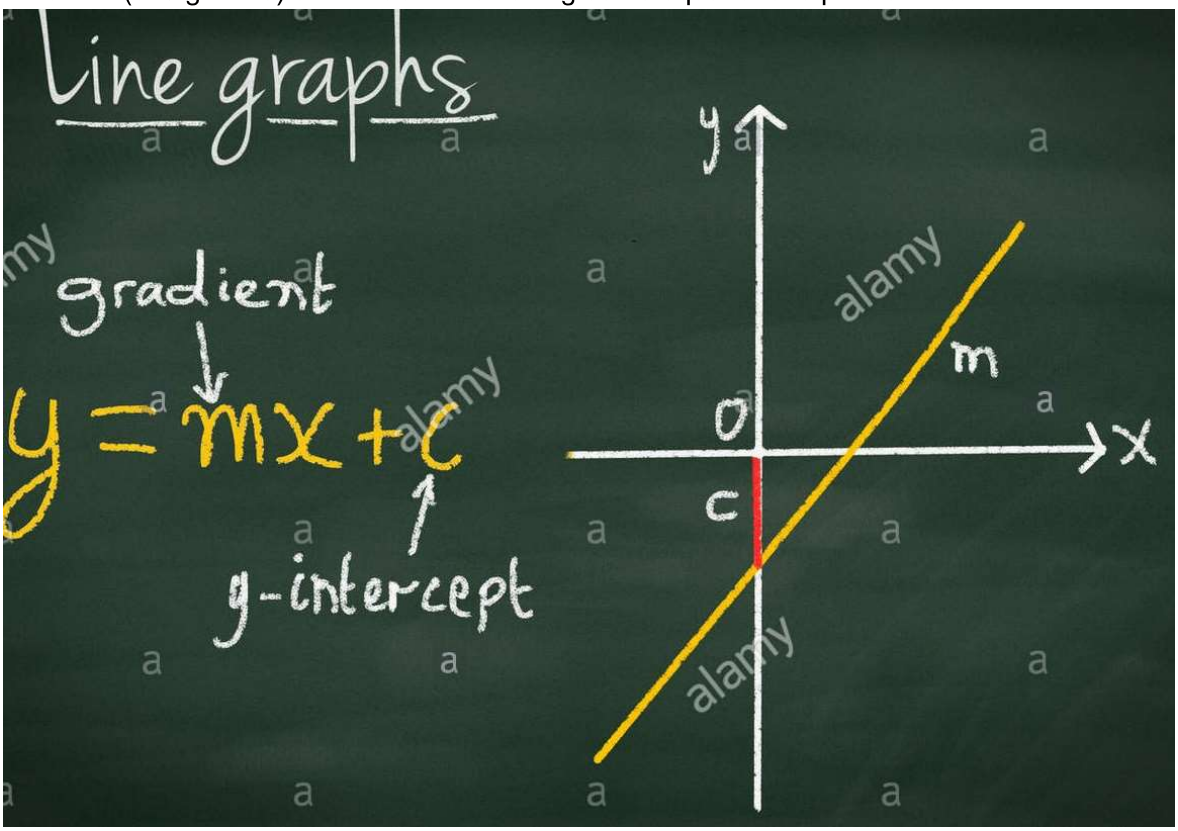
In [1]:
```
mporting some necessary modules for data handling,analysis and visualization requ
port numpy as np                    # data analysis and handling
port pandas as pd                   #data analysis and handling
port matplotlib.pyplot as plt       #visualization
port seaborn as sns                 #visualization
port warnings                       #to avoid warnings
rnings.filterwarnings('ignore')
```

In [2]:
```
#Just visualizing how a simple linear line looks like
x=np.arange(5)
plt.plot(x,'-bo')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('fig1:Positive Liear-Line')
plt.grid(True)
```



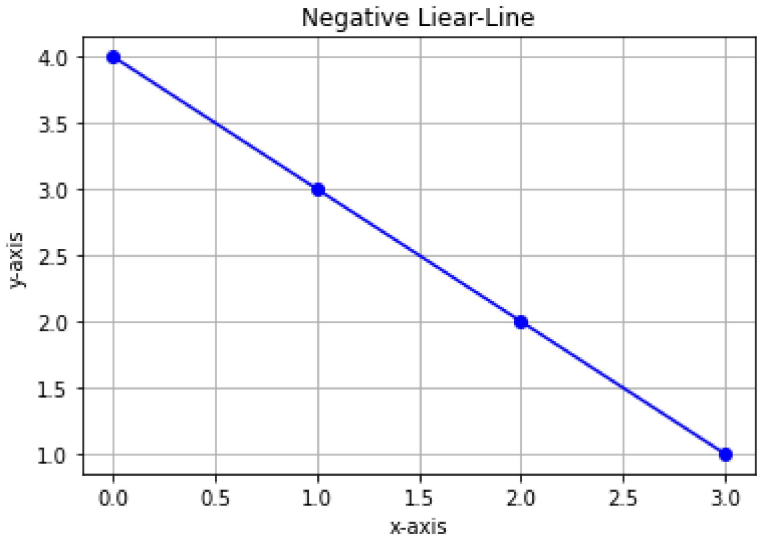
line equation:

- linear line(straight line) shown above is having some equations to predict Y- axis



- y=mx+c
- m=gradient(slope)
- gradient= rate of change of y w.r.t x
- c=intercept
- intercept= value of y when x is zero.

In [3]:
```
#this is also an example of linear line
x = range(4,0,-1)
plt.plot(x,'-bo')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Negative Liear-Line')
plt.grid(True)
```



1.2.what is Regression:

- A measure of the relation between the mean value of one variable (e.g. output) and corresponding value of other variables(eg.time and cost).

Regression Analysis and why we use it:

- Regression analysis is form of predictive modelling technique.
- It investigates relationship between dependent(target) and independent(predictor) variables.

why we use:

- Regression analysis is an important tool for modelling and analyzing data, here we fit a curve/line to data points, in such a manner that differences between distances of data points from curve or line is minimized.

Benefits of using Regression analysis?

- It indicates significant relationship between dependent and independent variables.
- it indicates the strength of impact of multiple independent variables on a dependent variables.

Regression analysis types:

- Linear Regression.(one independent variable and single predictor)
- Mulitple Linear Regression.(multiple independent variable and multiple predictor)
- Polynomial regression(Shape of regression line)

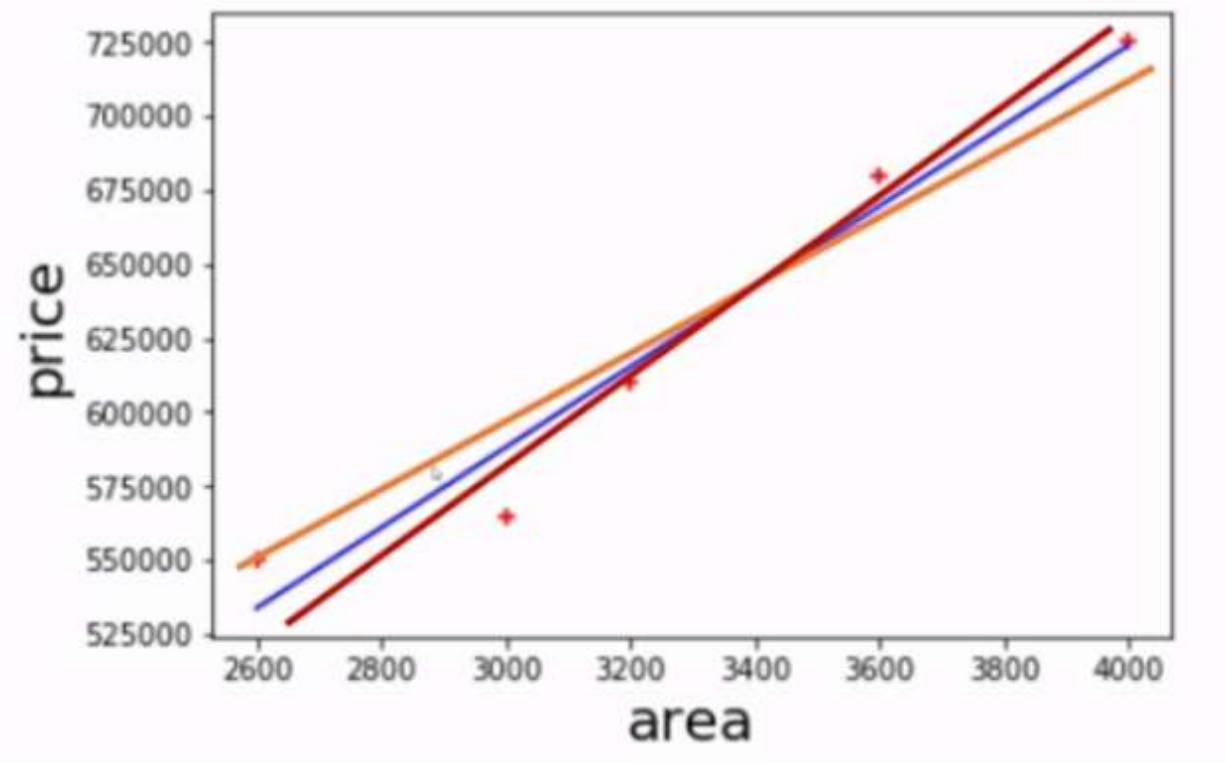
some important points before starting with linear regression:

- 1. X-axis is always a independent axis.
- 2. Y-axis is always a dependent axis.
- 3. Y-axis is always predicted by using given data on xaxis.
- 4. By using Linear line, Y-Axis can be predicted easily.
- 5. As per currently available data in fig1:positive linear line:
  - we can observe: x:1, Y:1; x:2,y:2
  - what about x:7, y:?, at this stage we can predict Y by using Linear Regression.

- If we are predicting Dependent (Y-axis) vairable on the basis of independent (X-axis) variable is known as Linear regression.
- Linear regression is always having one independent variable and one dependent(target variable) variable.

1.3 What is best fit line?

```
- Best fit line is a line touching most of the given data, which is mostly
linear in linear regression modules.
- y=mx+c
- According to this equation by varying slope(m) and intercept we can get many
straight line.
- But this is not satisfied, as we can draw multiple lines and this will give
infinite lines and much more time,but what is best fitting with given data is
not yet found.
- Lines going exactly very close or mostly touching to the given data points is
known as best fit line which is required for each data set.
```



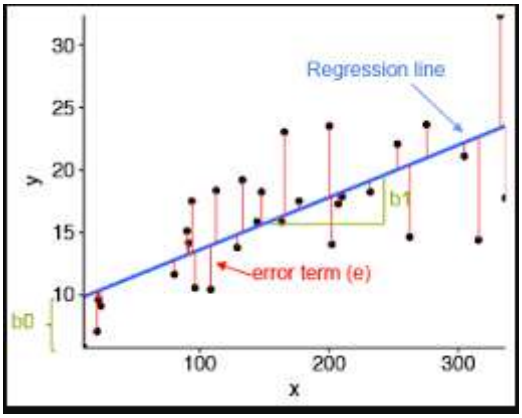
Multiple best fit lines.

how to select best fit line:

- line with a good R2 score and 0(null) error is known as best fit line.
- There are underfiting and overfiting lines too, going to be discussed further.

1. what is error:

- Error= y - y predicted.



How to calculate error:

- 1.Mean squared error(MSE), 2.Root Mean squared error(RMSE), 3.Mean Absolute Error (MAE).

2.What is R2_score: ¶

- R2_score decides how well is our model performing compared to baseline model.
- R2_score lies between -infinity to 1:
- 1. if r2_score is near to 1 indicates it is close to zero error (good model),
- 2. if r2_score is near to 0 indicates it is close to baseline model,
- 3. if r2_score is -ve means it is worse model.

how to calculate r2_score:

$$R^2 = 1 - \frac{\text{MSE(model)}}{\text{MSE(baseline)}}$$

$$\text{MSE(baseline)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

1.4.Mathematical model of Linear Regression:

```
![title](Linear_reg_math.png)
```

1.5.Performing step by step mathematical problem of Linear Regression:

```
- modules are already imported at start of topic.
- Aim = To find best fit line.
```

In [4]:
```
x=np.arange(1,6)
y=np.array([3,4,2,4,5])
```

In [5]:
```
dic = {'X':x, 'Y':y}
df=pd.DataFrame(dic)
df
```

Out[5]:

	X	Y
0	1	3
1	2	4
2	3	2
3	4	4
4	5	5

m and x are unknown:

- find m(slope) by formula:

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

- where,
- x= given independent variable, x bar= mean of all x, y= given dependent variable, y bar= mean of y.

In [11]:
```
x_mean=x.mean()
y_mean=y.mean()

#xmean and y mean are nothing but x bar and y bar

print(f'mean of x is {x_mean}')
print(f'mean of y is {y_mean}')
```

```
mean of x is 3.0
mean of y is 3.6
```

In [15]:
```
#x,x bar, y, ybar are known so we can find m from above formula:

m = sum((x-x_mean)*(y-y_mean))/ sum((x-x_mean)**2)

print(f'The coefficient of regression (slope(m)) is {m}')
```

```
The coefficient of regression (slope(m)) is 0.4
```

In [17]:
```
# y= mx+c: calculate C

c = y_mean - m * x_mean

print(f'Intercept of line is {c}')
```

```
Intercept of line is 2.4
```

In [34]:
```
def linreg(x):
    return (m*x)+c
```

In [35]:
```
x
```

Out[35]:
```
array([1, 2, 3, 4, 5])
```

In [36]:
```
ypred = np.array(list(map(linreg, x)))
```

In [37]:
```
ypred
```

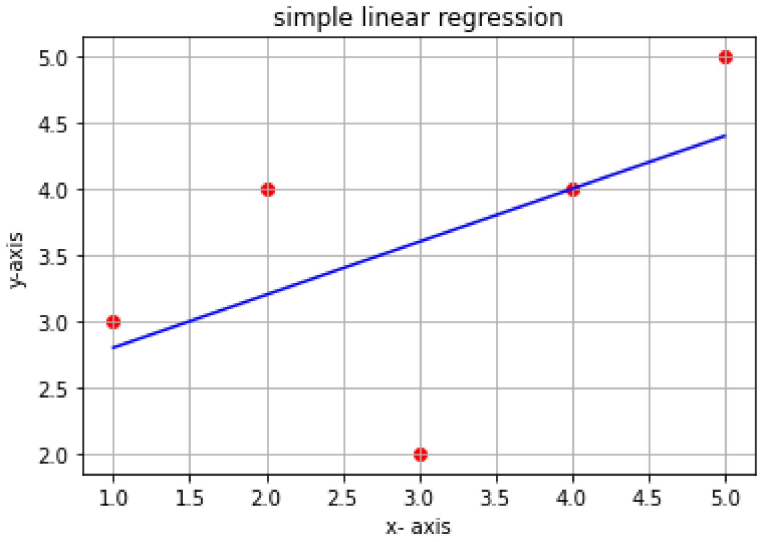
Out[37]:
```
array([2.8, 3.2, 3.6, 4. , 4.4])
```

In [38]:
```
dic={'X':x,'Actual Y': y, 'predicted Y':ypred}
df=pd.DataFrame(dic)
df
```

Out[38]:

	X	Actual Y	predicted Y
0	1	3	2.8
1	2	4	3.2
2	3	2	3.6
3	4	4	4.0
4	5	5	4.4

In [40]:
```
plt.scatter(x,y,color='red') #actual value
plt.plot(x,ypred,color='blue') #predicted value

plt.title('simple linear regression')
plt.xlabel('x- axis')
plt.ylabel('y-axis')
plt.grid(True)
plt.show()
```



In [46]:
```
#now calculating accuracy of model by r2_score
r2 = sum((ypred-y_mean)**2)/sum(y-y_mean)**2
```

In [47]:
```
print(f'The R2 score is -: {r2*100}')
```

```
The R2 score is -: 8.112963841460672e+32
```

- From predicted R2 score,we can justify the predicted model is not good.
- score near 0 indicates that model is near baseline and score near 1 indicates model is highly accurate and -ve score indicates worst model

1.6 Create and predict model using Sklearn library, Most commonly used in ML.

- This is simplest way to create and predict model.
- The google company has made all above calculation in one library Scikit-learn (sklearn) and made it open source for easy and accurate calculations.

In [53]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore') #ignore all future based warnings
```

In [54]:
```
# df = pd.read_csv('xyz.csv')
x = np.arange(1,6)
y = np.array([3,4,2,4,5])
```

In [55]:
```
dict = {'X':x, 'Y':y}

df = pd.DataFrame(dict)
df
```

Out[55]:

	X	Y
0	1	3
1	2	4
2	3	2
3	4	4
4	5	5

In [57]:
```
df.info()
#all the parameters should me numerical and non null, if categorical than encode
```

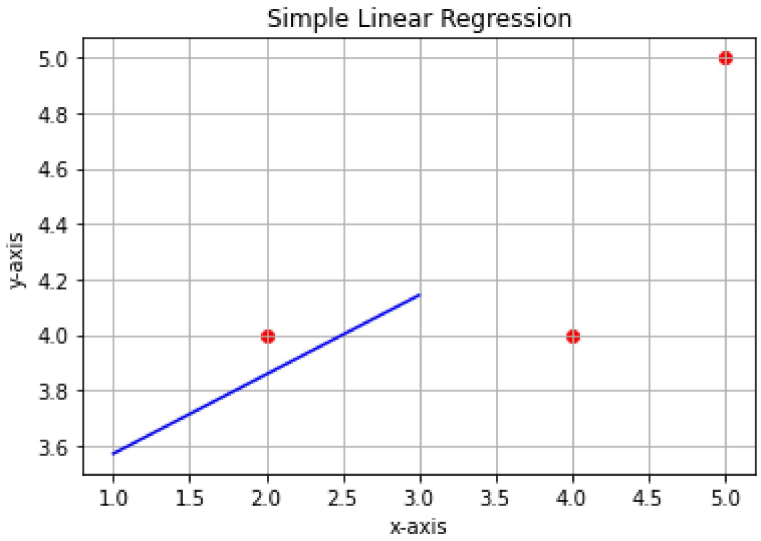
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X       5 non-null      int32
 1   Y       5 non-null      int32
dtypes: int32(2)
memory usage: 168.0 bytes
```

In [59]:
```
x = df.iloc[:, :-1].values
#create 2D array and select 1st column
#x=df.iloc[start row from : end row before, start column from : end column before
#.values converts row into array
y = df.iloc[:,-1].values
```

In [61]:
```
#import library to train our data according to given data

from sklearn.model_selection import train_test_split

xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.25,random_state=0)
```

In [62]:
```
#now creating model

#step1: call the model class from package
from sklearn.linear_model import LinearRegression

#step2: create the object for the model
linreg = LinearRegression()

#step3: train model m and c
linreg.fit(xtrain,ytrain)

#step4: test model
ypred=linreg.predict(xtest)
```

In [66]:
```
plt.scatter(xtrain,ytrain,color='red')
plt.plot(xtest, ypred,color='blue')

plt.title('Simple Linear Regression')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.grid(True)
plt.show()
```



In [67]:
```
dic = {"x": xtest.flatten(), "Actual Y": ytest, "Predicted Y":ypred}

df=pd.DataFrame(dic)
df
```

Out[67]:

	x	Actual Y	Predicted Y
0	3	2	4.142857
1	1	3	3.571429

- from above data actual and predicted y are having large gap indicates bad model

In [77]:
```
print(f'coeffient is-: {linreg.coef_}' )
```

```
coeffient is-: [0.28571429]
```

In [79]:
```
linreg.intercept_
```

Out[79]:
```
3.2857142857142856
```

In [81]:
```
#checking error
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

print(f"Mean Absolute Error -: {mean_absolute_error(ytest, ypred)}")
print(f"Mean Square Error -: {mean_squared_error(ytest, ypred)}")
print(f"Root Mean Square Error -: {np.sqrt(mean_squared_error(ytest, ypred))}")
print(f"R2_score -: {r2_score(ytest, ypred)}")
```

```
Mean Absolute Error -: 1.3571428571428568
Mean Square Error -: 2.4591836734693864
Root Mean Square Error -: 1.5681784571500101
R2_score -: -8.836734693877546
```

- As per analysis model is very bad, as we are having all errors very high, and r2_score is also -ve indicating a model is very worst.

In [ ]:
```

```