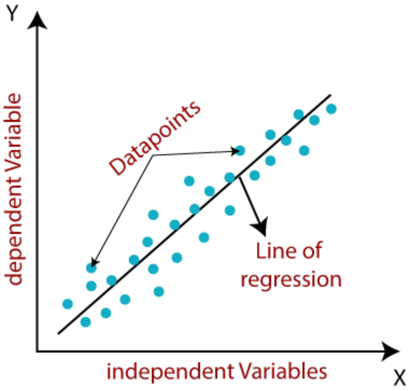
Practical No. 2

Aim : For a given set of training data examples stored in a .CSV file implement Linear Regression algorithm.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable. Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

he linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



Mathematically, we can represent a linear regression as:

$y= a_0+a_1x+ \varepsilon$

Here,

Y= Dependent Variable (Target Variable)
X= Independent Variable (predictor Variable)
a0= intercept of the line (Gives an additional degree of freedom)
a1 = Linear regression coefficient (scale factor to each input value).
ε = random error

The values for x and y variables are training datasets for Linear Regression model representation.

Types of Linear Regression

Linear Regression can be broadly classified into two types of algorithms:

1.Simple Linear Regression

A simple straight-line equation involving slope (dy/dx) and intercept (an integer/continuous value) is utilized in simple Linear Regression. Here a simple form is:

y=mx+c

where y denotes the output, x is the independent variable, and c is the intercept when x=0. With this equation, the algorithm trains the model of machine learning and gives the most accurate output

2. Multiple Linear Regression

When a number of independent variables are more than one, the governing linear equation applicable to regression takes a different form like:

y= c+m1x1+m2x2… mnxn where represents the coefficient responsible for impact of different independent variables x1, x2 etc. This machine learning algorithm, when applied, finds the values of coefficients m1, m2, etc., and gives the best fitting line.

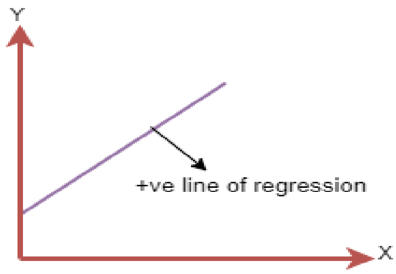
3. Non-Linear Regression

When the best fitting line is not a straight line but a curve, it is referred to as Non-Linear Regression.

Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a regression line. A regression line can show two types of relationship:

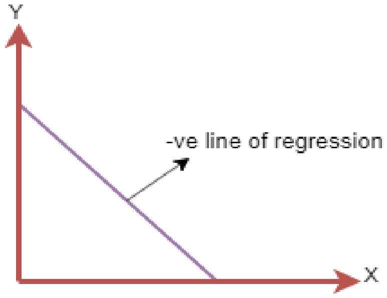
Positive Linear Relationship:



The line equation will be: **$Y= a_0+a_1x$**

If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.

Negative Linear Relationship:



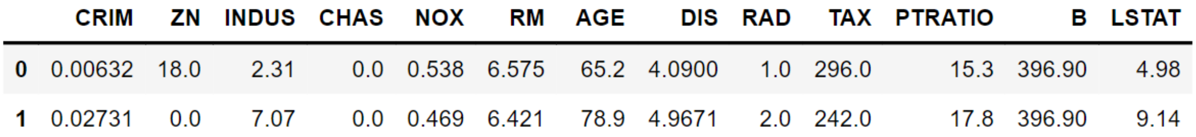
The line of equation will be: $Y= -a_0+a_1x$

If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.

Code :

```
import pandas as pd

import numpy as np

import seaborn as sns

from sklearn.datasets import load_boston

df=load_boston()

dataset=pd.DataFrame(df.data)

dataset.columns=df.feature_names

dataset.head(2) # These are the independent features
```

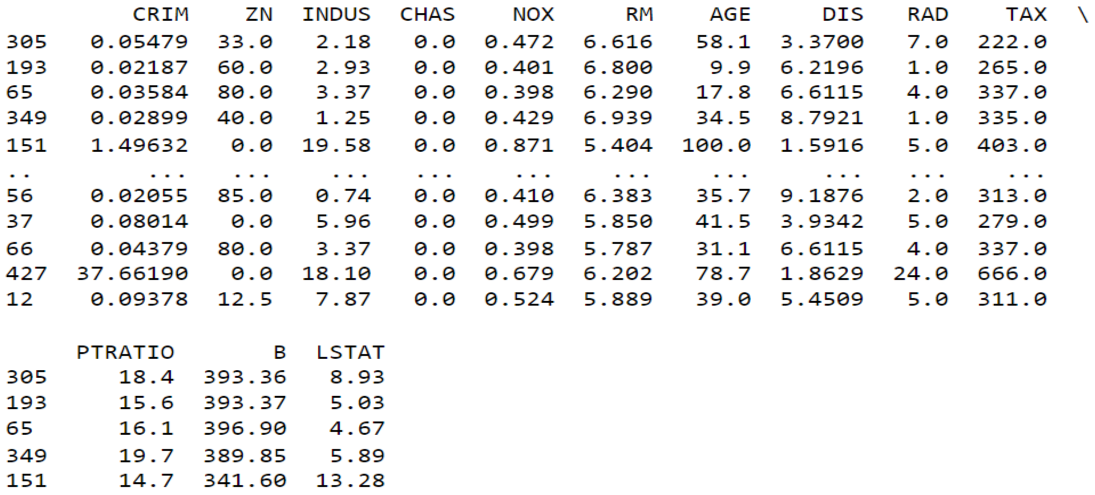
Output :



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14

```
#independent and dependent

x=dataset

y=df.target

#train test split

from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.30,random_state=10)

print(X_test)
```

Output :

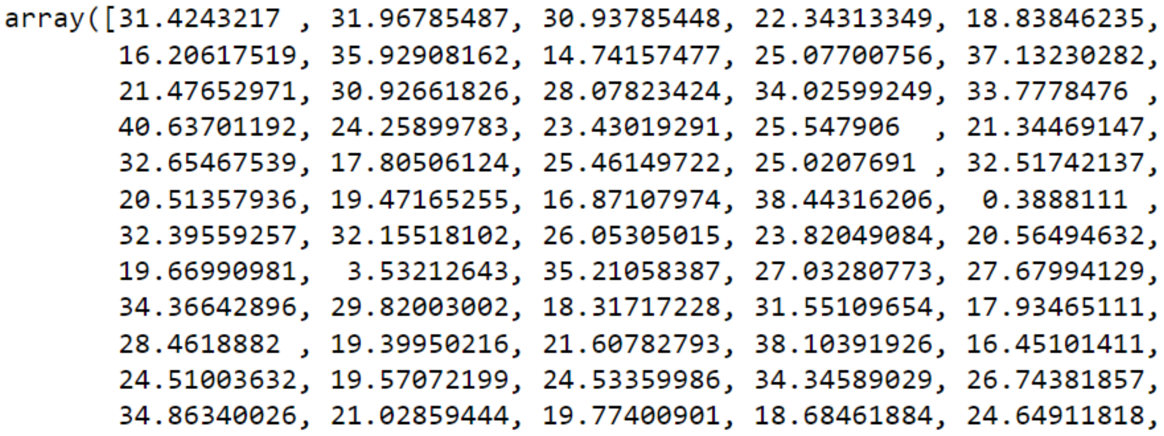


```
         CRIM    ZN  INDUS  CHAS    NOX     RM    AGE     DIS   RAD    TAX  \
305   0.05479  33.0   2.18   0.0  0.472  6.616   58.1  3.3700   7.0  222.0
193   0.02187  60.0   2.93   0.0  0.401  6.800    9.9  6.2196   1.0  265.0
65    0.03584  80.0   3.37   0.0  0.398  6.290   17.8  6.6115   4.0  337.0
349   0.02899  40.0   1.25   0.0  0.429  6.939   34.5  8.7921   1.0  335.0
151   1.49632   0.0  19.58   0.0  0.871  5.404  100.0  1.5916   5.0  403.0
..        ...   ...    ...   ...    ...    ...    ...     ...   ...    ...
56    0.02055  85.0   0.74   0.0  0.410  6.383   35.7  9.1876   2.0  313.0
37    0.08014   0.0   5.96   0.0  0.499  5.850   41.5  3.9342   5.0  279.0
66    0.04379  80.0   3.37   0.0  0.398  5.787   31.1  6.6115   4.0  337.0
427  37.66190   0.0  18.10   0.0  0.679  6.202   78.7  1.8629  24.0  666.0
12    0.09378  12.5   7.87   0.0  0.524  5.889   39.0  5.4509   5.0  311.0

     PTRATIO       B  LSTAT
305     18.4  393.36   8.93
193     15.6  393.37   5.03
65      16.1  396.90   4.67
349     19.7  389.85   5.89
151     14.7  341.60  13.28
```

```
from sklearn.linear_model import LinearRegression

model=LinearRegression()

model.fit(X_train,Y_train)

print('length of X_train ',len(X_train))

print('length of X_test  ',len(X_test))

model.predict(X_test)
```

```
length of X_train  354

length of X_test   152
```

Output :



```
array([31.4243217 , 31.96785487, 30.93785448, 22.34313349, 18.83846235,
       16.20617519, 35.92908162, 14.74157477, 25.07700756, 37.13230282,
       21.47652971, 30.92661826, 28.07823424, 34.02599249, 33.7778476 ,
       40.63701192, 24.25899783, 23.43019291, 25.547906  , 21.34469147,
       32.65467539, 17.80506124, 25.46149722, 25.0207691 , 32.51742137,
       20.51357936, 19.47165255, 16.87107974, 38.44316206,  0.3888111 ,
       32.39559257, 32.15518102, 26.05305015, 23.82049084, 20.56494632,
       19.66990981,  3.53212643, 35.21058387, 27.03280773, 27.67994129,
       34.36642896, 29.82003002, 18.31717228, 31.55109654, 17.93465111,
       28.4618882 , 19.39950216, 21.60782793, 38.10391926, 16.45101411,
       24.51003632, 19.57072199, 24.53359986, 34.34589029, 26.74381857,
       34.86340026, 21.02859444, 19.77400901, 18.68461884, 24.64911818,
```

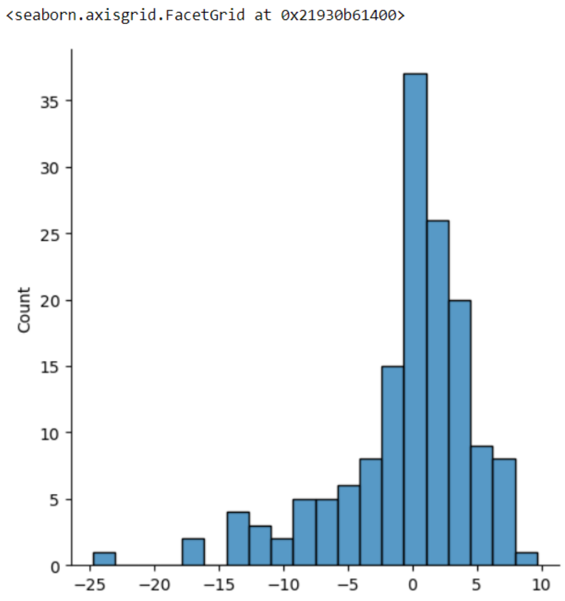
```
model.score(X_test,Y_test)
```

Output :

0.6996255772983113

```
#Prediction
y_pred=model.predict(X_test)
sns.displot(y_pred-Y_test)
```

Output :



```
<seaborn.axisgrid.FacetGrid at 0x21930b61400>
```