Linear Regression

1. Linear regression is one of the easiest and most popular Machine Learning algorithms.
2. 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.
3. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) 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.
4. Mathematically, we can represent a linear regression as: y= a0+a1x+ ε
5. Here,

i)Y= Dependent Variable (Target Variable)

ii)X= Independent Variable (predictor Variable)

iii)a0= intercept of the line (Gives an additional degree of freedom)

iv)a1 = Linear regression coefficient (scale factor to each input value).

v)ε = random error

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

In [9]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [21]:
```
column_names = ['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad
       'ptratio', 'black', 'lstat','medv']
df = pd.read_csv('./housing.csv', header=None, delimiter=r"\s+", names=column_
```

In [22]:
```
df
```

Out[22]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	m
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	2
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	2
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	3
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	3
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	2
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	2
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	2
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	2
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	1

506 rows × 14 columns

In [23]:
```
df.shape
```

Out[23]:
```
(506, 14)
```

df.describe()

Here we seperate the first 13 columns in X and the last column to be predicted as Y.

In [25]:
```
X = df[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax'
       'ptratio', 'black', 'lstat']]
Y = df['medv']
```

Splitting the data into training and testing data using train_test_split

In [26]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.3, rand
```

Standardizing the data

In [27]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Importing and loading the Linear Regression model on the data.

In [28]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

Printing our predicted values

In [29]:
```
print(y_pred)
```

```
[30.77834822 38.66581541 16.96705276 26.79035818 20.36068008 24.89386356
 18.98948959 15.44554414 24.50205701 22.12455513 27.16709892 20.60218969
 -4.91312377 23.45520373 20.44300739 27.97454642 21.82481134  6.80793603
 43.13876668 19.08804672 28.93737925 31.91943808 12.05148619 25.41493356
 19.60614164 17.23543021 24.61263361 16.907528   24.25623469 20.73183868
 23.98650325 26.66317526 27.45468242 19.57475267 18.15660164 19.94422395
 33.04896681 21.26475982 25.83570427 26.36436267 15.22545112 33.61610719
 45.24385042 18.83620774 28.91584593 18.49470905 14.91054896 27.65199578
 21.75498681 32.36964545 23.36168054 36.5955736  16.82098601 27.59874838
 42.39261672 24.39945725 20.42785358 34.46608842 26.52068304 14.0304141
 24.03170468 32.40960974 33.59067192 17.33583061 22.46180876 18.28308242
 22.01801285 27.57486429 32.54366102 13.95067422 21.96577449 29.21079363
 12.65101384 17.15704754 25.56951945  6.99478235 22.73334313 43.93983367
 20.03571324 10.919632   22.50309934 14.46771561 23.07376266 10.52688537
 24.71136556 34.38262811 21.02382869 27.04400189 30.94450114 21.54068028
 27.48732329  7.31708375 21.56254771 16.57285469 14.43338806 22.26113509
 26.42907376  1.28686278 15.98179144 18.13930581 23.62097091 26.3208465
 12.12867204 20.66733158 25.32719152 14.23008516 19.48076211 26.9289352
 22.02461937 26.11181168  9.83654844 21.30376854 23.13155127 28.99045653
 34.46072055 17.31422502 36.81803955 14.0639201  22.38719669 30.47720506
 16.96724187 26.23079837  6.30719991 25.6891189  27.44337177 24.41723783
 26.72754089 35.52652858 23.8342233  41.30725508 15.51580837 27.19720138
 18.99437335 22.50080685 11.95557604 23.13463621 23.4954008  34.27666272
 33.53948706 16.55984176 17.98511069 31.00757978 26.51337445 18.4765867
  7.99167823 28.32517448 26.1665569  19.032183   14.47502635 42.88762567
 18.69947012 19.82530166]
```

Root Mean Squared Error:

In [30]:
```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mead squared Error is:")
print(rmse)
```

```
Root Mead squared Error is:
4.912717301969202
```

Training Accuracy:

In [31]:
```
print("Training accuracy is:")
lr.score(X_train, y_train)
```

```
Training accuracy is:
```

Out[31]:
```
0.7434997532004697
```

Testing Accuracy:

In [33]:
```
print("Testing accuracy is:")
lr.score(X_test, y_test)
```

```
Testing accuracy is:
```

Out[33]:
```
0.6761000049033605
```

In [ ]:
```

```