Data Science | 30 Days of Machine Learning | Day - 26

Educator Name: Nishant Dhote
Support Team: **+91-7880-113-112**

----Today Topics | Day 26----

Multiple Linear Regression

- Multiple Linear Regression
- Equation 3D Plane
- Equation 4D Hyperplane

Dataset Link GitHub: https://github.com/TheiScale/30_Days_Machine_Learning/

- **Linear Regression**

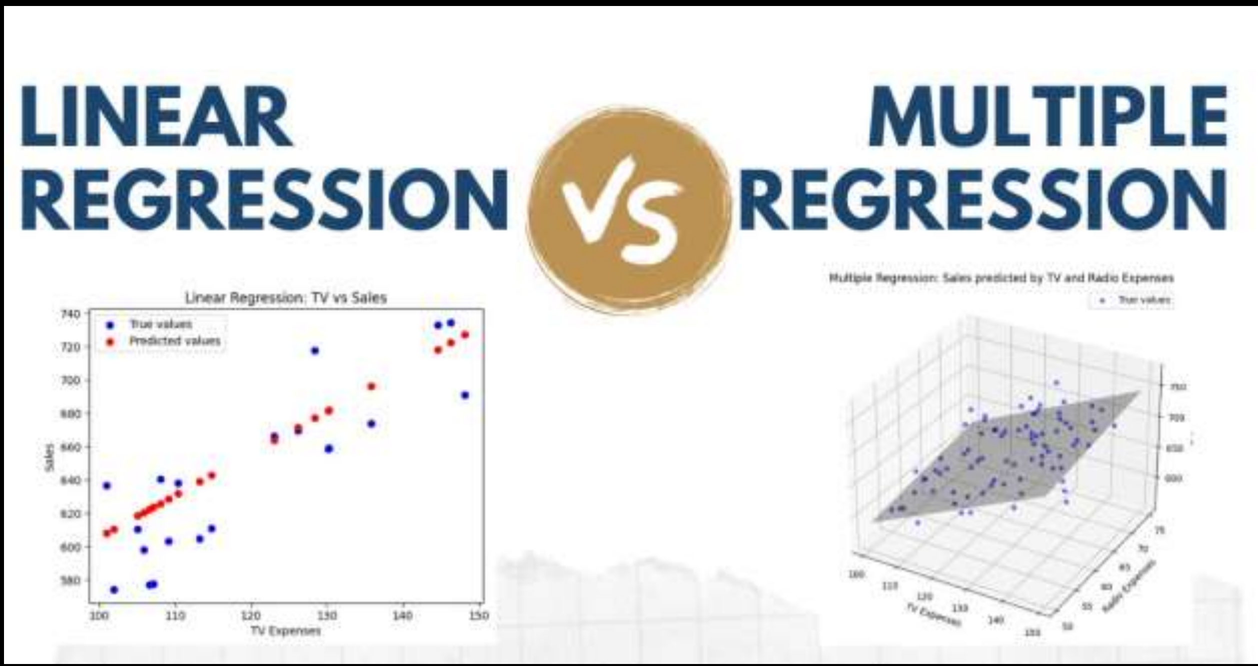
It is the basic and commonly used type for predictive analysis. It is a statistical approach to modelling the relationship between a dependent variable and a given set of independent variables.

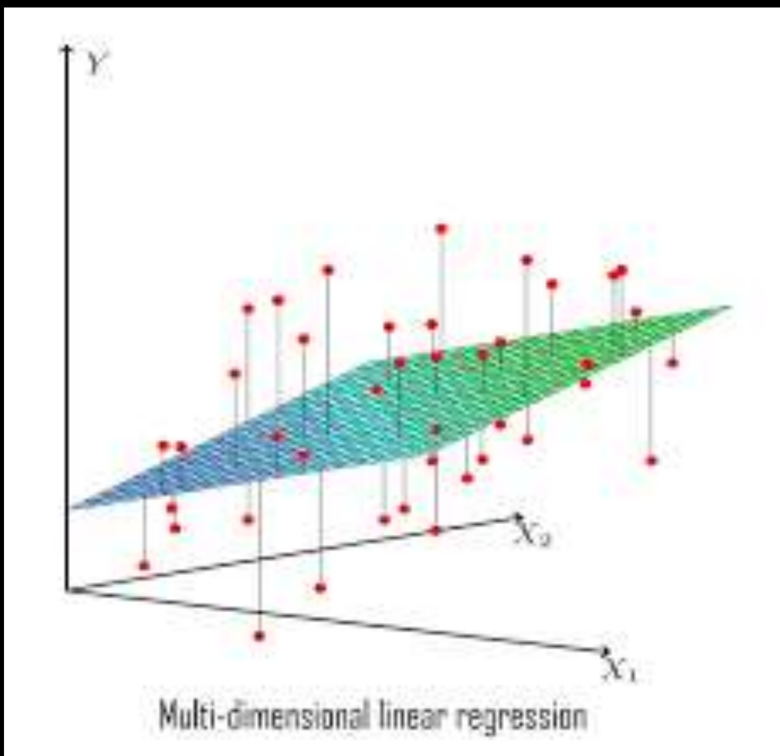
These are of two types:

1. Simple linear Regression
2. Multiple Linear Regression

- **Multiple Linear Regression**

Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.





Multi-dimensional linear regression

Difference Between LR and MLR

Difference Between Linear Regression and Multiple Regression: Linear Regression vs Multiple Regression

Parameter	Linear (Simple) Regression	Multiple Regression
Definition	Models the relationship between one dependent and one independent variable.	Models the relationship between one dependent and two or more independent variables.
Equation	$Y = C_0 + C_1X + e$	$Y = C_0 + C_1X_1 + C_2X_2 + C_3X_3 + \ldots\ldots + C_nX_n + e$
Complexity	Simpler dealing with one relationship.	More complex due to multiple relationships.
Use Cases	Suitable when there is one clear predictor.	Suitable when multiple factors affect the outcome.
Assumptions	Linearity, Independence, Homoscedasticity, Normality	Same as linear regression, with the added concern of multicollinearity.
Visualization	Typically visualized with a 2D scatter plot and a line of best fit.	Requires 3D or multi-dimensional space, often represented using partial regression plots.
Risk of Overfitting	Lower, as it deals with only one predictor.	Higher, especially if too many predictors are used without adequate data.
Multicollinearity Concern	Not applicable, as there's only one predictor.	A primary concern; having correlated predictors can affect the model's accuracy and interpretation.
Applications	Basic research, simple predictions, understanding a singular relationship.	Complex research, multifactorial predictions, studying interrelated systems.

Multiple Linear Regression

LR

1 Simple Linear R

2 Multiple LR

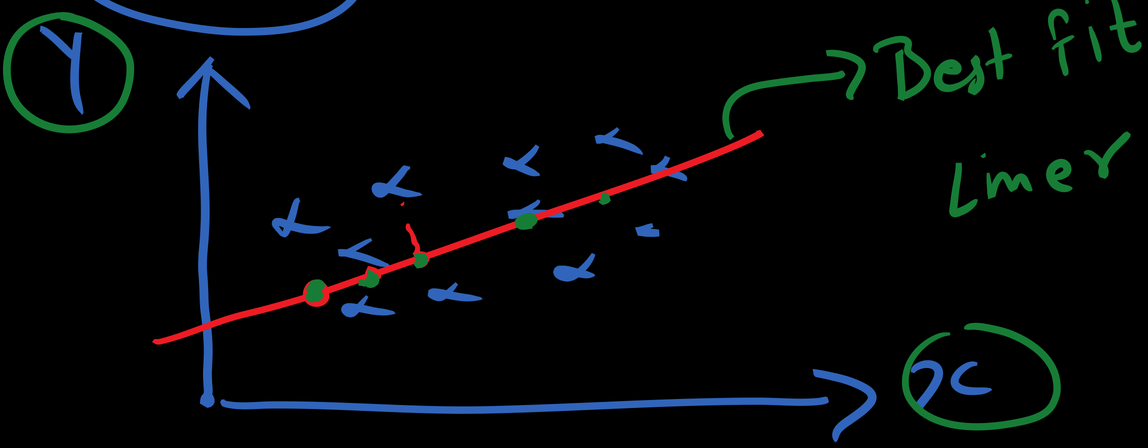
3 Polynomial LR

① Simple L R

x | Y

Input

output



$Y = mx + b$

(m, b) ?

② Multiple L R

$x_1 \mid x_2 \mid x_3 \mid Y$

Multiple I/P columns

Independent columns

dependent columns

Example:- 2D — Simple LR

CgPA	LPA
x	y

3D MLR

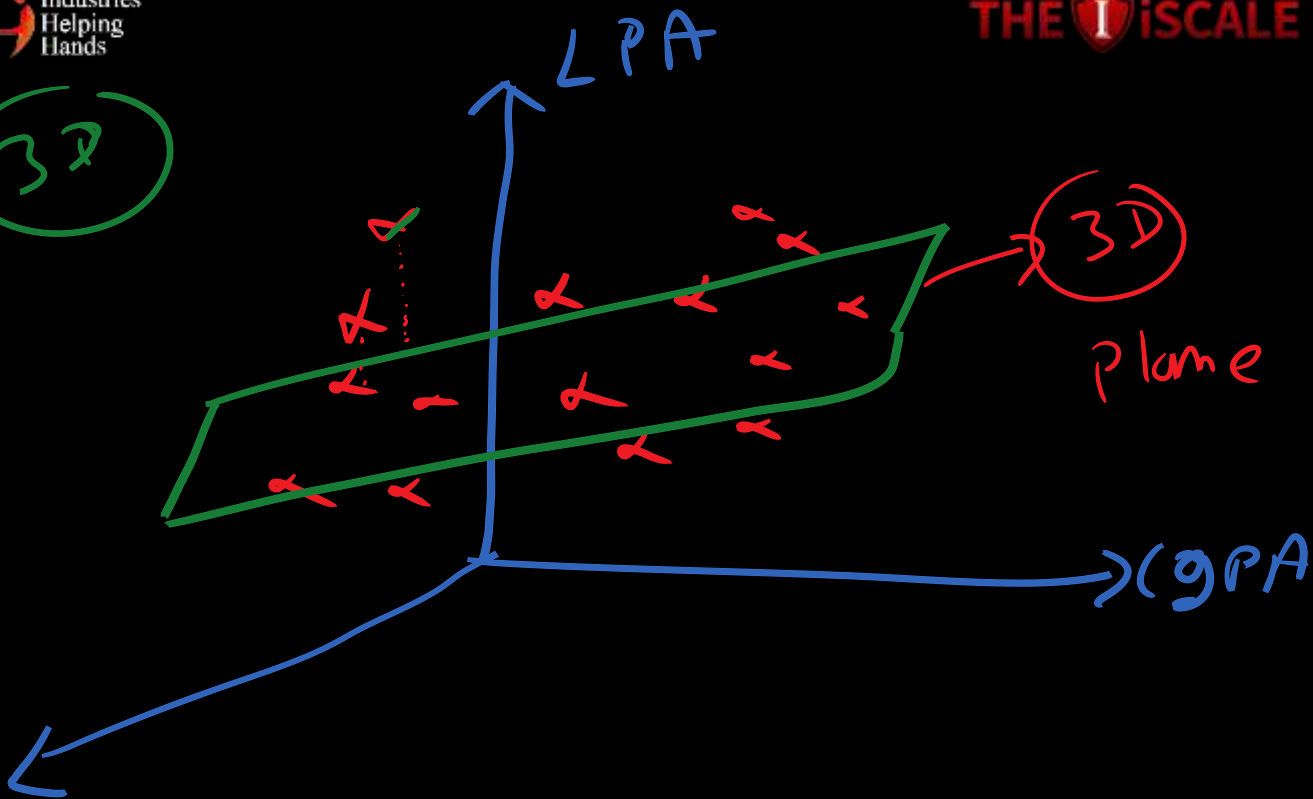
CgPA	%	~~gender~~	LPA
x_1	x_2	~~x_3~~	y

I/P

3D plane

CgPA	%	LPA
x_1	x_2	y

3D



4D → Hyper plane

2D → Equation

$Y = mx + b$ | or $Y = \beta_0 + \beta_1 x_1$

3D — Eqn →

$$\boxed{Y = mx_1 + mx_2 + b}$$

or

$$\boxed{Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2}$$

3) D $\rightarrow$ $\boxed{\beta_0, \beta_1, \beta_2}$?

4D Eqn

$$\boxed{\beta_0, \beta_1, \beta_2, \beta_3}$$

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

$\boxed{n^{th} \text{ Dim Eq}^n}$

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

$$\boxed{Y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i}$$

3D

$$Y = B_0 + B_1 X_1 + B_2 X_2$$

$x_1 \to$ cgpA | $x_2 = \%$

$Y = LPA$

$$LPA = B_0 + B_1 (cgpA) + B_2 (\%)$$

$\downarrow$

Cofficiant = weigths

$B_2 \downarrow$ $B_1 \uparrow$

$B_2 > B_1$

$\boxed{\text{Closest Point Now to HyperPlane}}$

<Start Coding>

```
#Import Library


from sklearn.datasets import make_regression
import pandas as pd
import numpy as np

import plotly.express as px
import plotly.graph_objects as go

from sklearn.metrics import
mean_absolute_error,mean_squared_error,r2_score

#Function Called Make Regression

X,y = make_regression(n_samples=100, n_features=2,
n_informative=2, n_targets=1, noise=50)


#Define Data Frame

df =
pd.DataFrame({'feature1':X[:,0],'feature2':X[:,1],'t
arget':y})

----
df.shape

----
df.head()
----
```

```
#Data Scatter Plot with the help of Plotly

fig = px.scatter_3d(df, x='feature1', y='feature2',
z='target')

fig.show()

#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=0.2,random_state=3)

#Import LR Class

from sklearn.linear_model import LinearRegression

----
lr = LinearRegression()
----
lr.fit(X_train,y_train)
----
y_pred = lr.predict(X_test)


#Print Values
print("MAE",mean_absolute_error(y_test,y_pred))
print("MSE",mean_squared_error(y_test,y_pred))
print("R2 score",r2_score(y_test,y_pred))
```

```
#Draw 3D Plane

# Generate x and y grids
x = np.linspace(-5, 5, 10)
y = np.linspace(-5, 5, 10)
xGrid, yGrid = np.meshgrid(y, x)

# Construct 'final' before using it in predictions
final = np.vstack((xGrid.ravel(), yGrid.ravel())).T

# Assuming lr is your linear regression model
# You need to have lr defined and trained before
this step

# Predict using the model and reshape the result
z_final = lr.predict(final).reshape(10, 10)

# Assign z_final to z if needed
z = z_final
```

```
#Data Scatter plot

fig = px.scatter_3d(df, x='feature1', y='feature2',
z='target')

fig.add_trace(go.Surface(x = x, y = y, z =z ))

fig.show()

#Apply coefficient for Beta values
lr.coef_
----
lr.intercept_
```

Keep Learning

Thank You