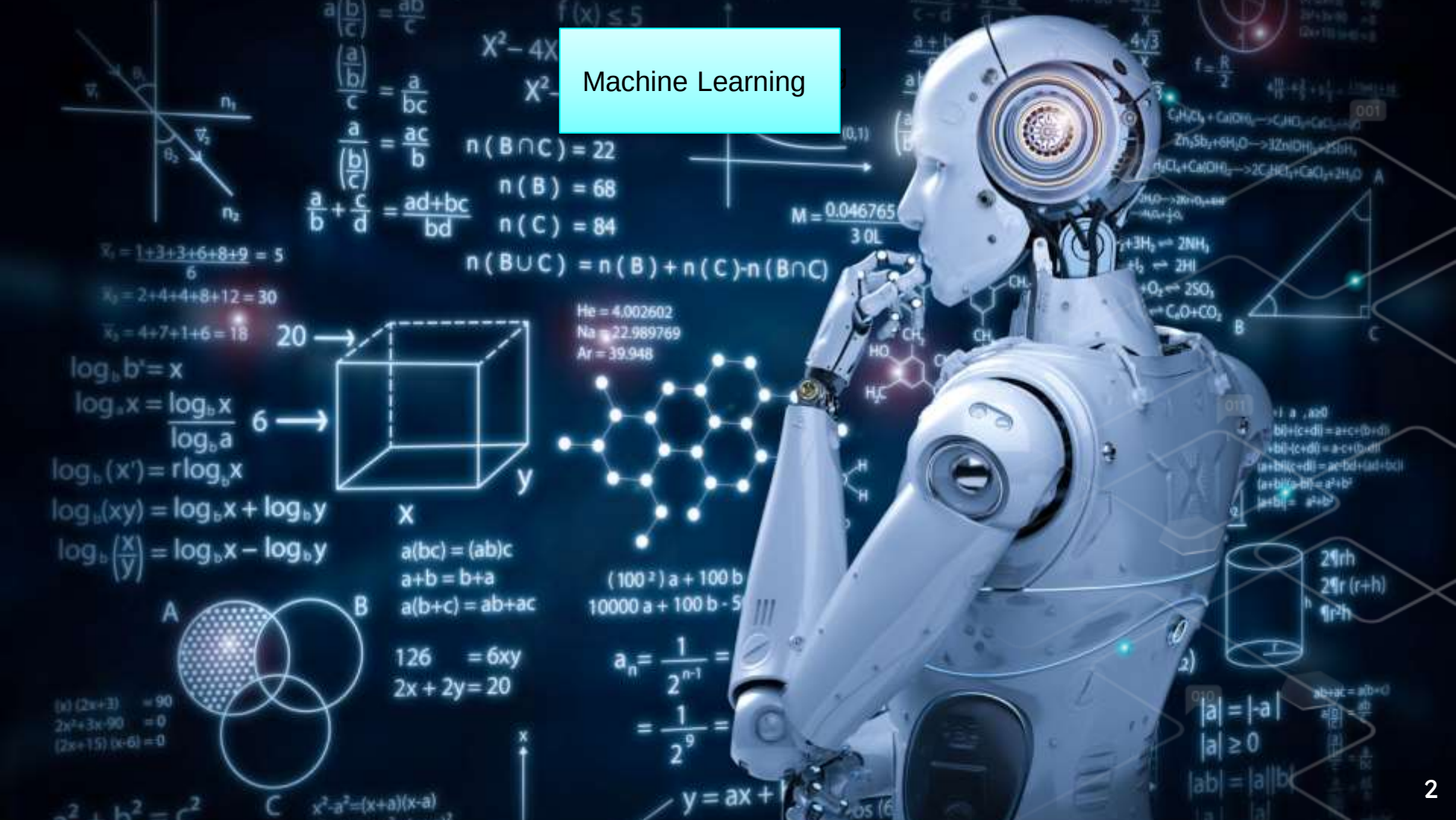
PROJECT BASED LEARNING-1

Group-1
Project-12


Machine Learning

Topic

Multi 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.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

where, for $i = n$ observations:

y_i = dependent variable

x_i = expanatory variables

β_0 = y-intercept (constant term)

β_p = slope coefficients for each explanatory variable

ϵ = the model's error term (also known as the residuals)

STEPS

- Import the Packages
- Loading the Dataset
- Data Preprocessing
- Train and Test Data Split
- Modelling
- Predicting
- Data Visualizing

Import the Pakages

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

Load the Dataset

There are 5 different functions to load the dataset. They are:

1. Manual Function
2. loadtxt Function
3. genfromtxt Function
4. read_csv Function
5. Pickle

I have used read_csv function to load the data.

```
In [2]: df = pd.read_csv(r"E:\csv files\50_Startups.csv")
```

Required :

- Packages
  - NumPy
  - Pandas
  - Sklearn
  - Matplotlib
- Softwares
  - Python
  - Jupyter Notebook

Data Preprocessing

- df.head()
- df.info()
- df.describe()
- df.isnull().sum()
- df.corr()

df.head( )

This function **returns the first 5 rows of the dataframe.** To override the default, you may insert a value between the parenthesis to change the number of rows returned.

Example: df.head(2) → this prints the first 2 rows of the dataframe

```
In [3]: df.head()
```

Out[3]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

df.info( )

This function is **used to print a concise summary of a DataFrame**. This method prints information about a DataFrame including the index dtype and column dtypes, non-null values and memory usage.

```
In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object 
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

df.describe( )

This function is **used for calculating some statistical data like percentile, mean and std** of the numerical values of the Series or DataFrame. It analyzes both numeric and object series and also the DataFrame column sets of mixed data types.

```
In [4]: df.describe()
```

Out[4]:

	R&D Spend	Administration	Marketing Spend	Profit
count	50.000000	50.000000	50.000000	50.000000
mean	73721.615600	121344.639600	211025.097800	112012.639200
std	45902.256482	28017.802755	122290.310726	40306.180338
min	0.000000	51283.140000	0.000000	14681.400000
25%	39936.370000	103730.875000	129300.132500	90138.902500
50%	73051.080000	122699.795000	212716.240000	107978.190000
75%	101602.800000	144842.180000	299469.085000	139765.977500
max	165349.200000	182645.560000	471784.100000	192261.830000

df.isnull( ).sum( )

This function **returns the number of missing values in the data set**. A simple way to deal with data containing missing values is to skip rows with missing values in the dataset.

```
In [5]: df.isnull().sum()

Out[5]: R&D Spend          0
        Administration     0
        Marketing Spend    0
        State              0
        Profit             0
        dtype: int64
```

df.corr( )

This function is **used to find the pairwise correlation of all columns in the dataframe.** Any na values are automatically excluded. For any non-numeric data type columns in the dataframe it is ignored.

```
In [6]: df.corr()
```

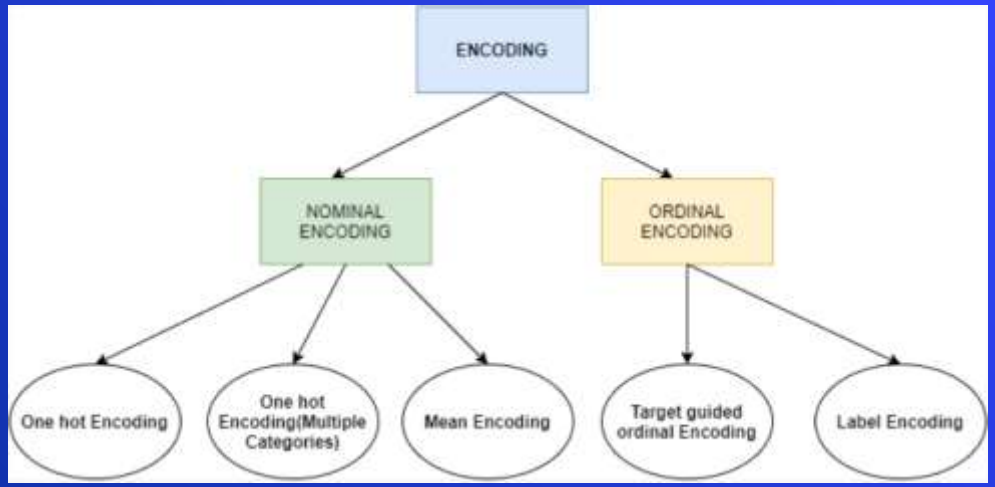
Out[6]:

	R&D Spend	Administration	Marketing Spend	Profit
R&D Spend	1.000000	0.241955	0.724248	0.972900
Administration	0.241955	1.000000	-0.032154	0.200717
Marketing Spend	0.724248	-0.032154	1.000000	0.747766
Profit	0.972900	0.200717	0.747766	1.000000

Encoding the data

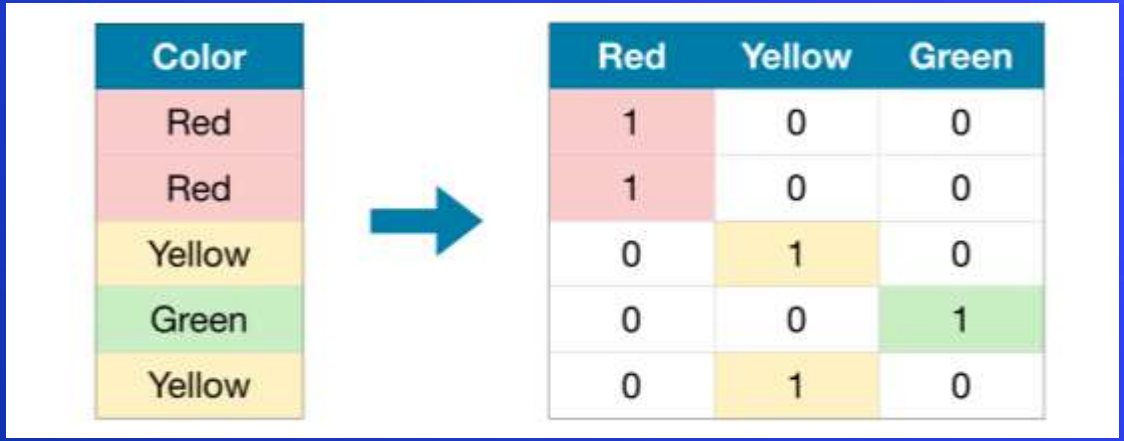
Encoding is **a technique of converting categorical variables into numerical values** so that it could be easily fitted to a machine learning model. We have different types of encoding techniques. We mainly use 2 types of encoding techniques. They are

1. Label Encoder
2. One-Hot Encoder



One-hot Encoder

One hot encoding is one method of converting data to prepare it for an algorithm and get a better prediction. With one-hot, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns. Each integer value is represented as a binary vector. All the values are zero, and the index is marked with a 1.

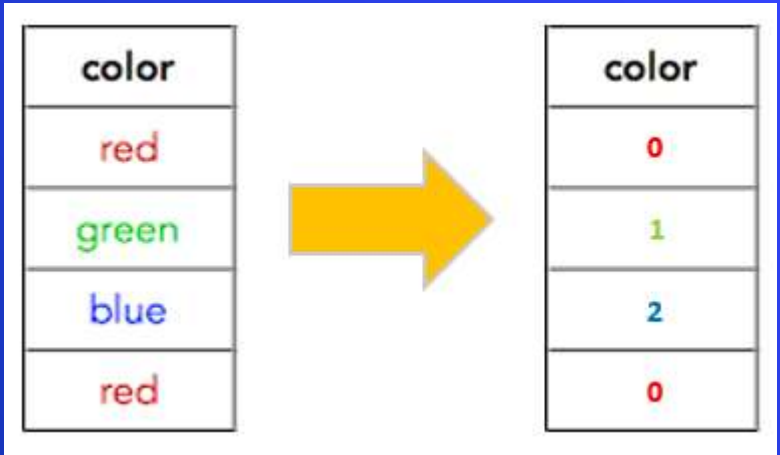


Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

Label Encoder

Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.



Label Encoder

We have used label Encoding technique in this Project to convert the categorical data.

```
In [8]: from sklearn.preprocessing import LabelEncoder
        le = LabelEncoder()
        df["State"] = le.fit_transform(df["State"])
        df["State"].head()

Out[8]: 0    2
        1    0
        2    1
        3    2
        4    1
        Name: State, dtype: int32
```

Train and Test the data

Train/Test is a method to measure the accuracy of your model. It is called Train/Test because you split the the data set into two sets: a training set and a testing set i.e 80% for training, and 20% for testing.

```
In [10]: x = df.drop(columns="Profit")
         y = df["Profit"]

In [11]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test = train_test_split (x,y,test_size=0.4,
                                                           random_state=42)
```

Modelling

Since our project is based on Multi Linear Regression we have to use 2 to 3 models to predict the accuracy. Here we have used Linear Regression model and also Lasso model. We fit and predict the data into the model.

```
In [12]: from sklearn.linear_model import LinearRegression
         lr = LinearRegression()
         lr.fit(x_train,y_train)
         y_pred = lr.predict(x_test)
```

```
In [16]: from sklearn.linear_model import Lasso
         lasso_ = Lasso()
         lasso_.fit(x_train,y_train)
         y_pred2 = lasso_.predict(x_test)
```

Predicting

“Prediction” refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome.

```
In [15]: print("Model's Testing Accuracy:",lr.score(x_test,y_test))
         print("Model's Training Accuracy:",lr.score(x_train,y_train))

         Model's Testing Accuracy: 0.9533550269348674
         Model's Training Accuracy: 0.9445212239677727
```

```
In [19]: print("Model's Testing Accuracy:",lasso_.score(x_test,y_test))
         print("Model's Training Accuracy:",lasso_.score(x_train,y_train))

         Model's Testing Accuracy: 0.9533550885394169
         Model's Training Accuracy: 0.944521222979573
```

Data Visualization

Data visualization is defined as a graphical representation that contains the information and the data. By using visual elements like charts, graphs, and maps, data visualization techniques provide an accessible way to see and understand trends, outliers, and patterns in data. Data visualization libraries available in python are

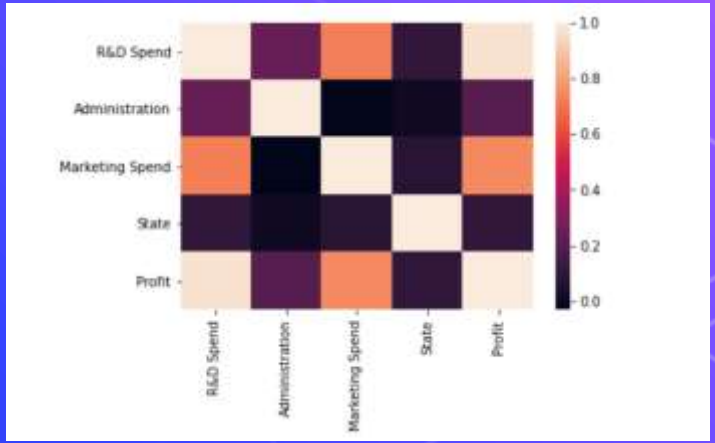
1. Matplotlib
2. Plotly
3. ggplot
4. Seaborn
5. Altair
6. Geoplotlib
7. Bokeh

Types of Visualizations

1. Line Chart
2. Bar Chart
3. Pie Chart
4. Donut Chart
5. Histogram Plot
6. Density Plot
7. Scatter Plot
8. Box Plot
9. Correlation Matrix Plot
10. Scatter Matrix Plot
11. Distribution Plot
12. Violin Plot
13. Heat map

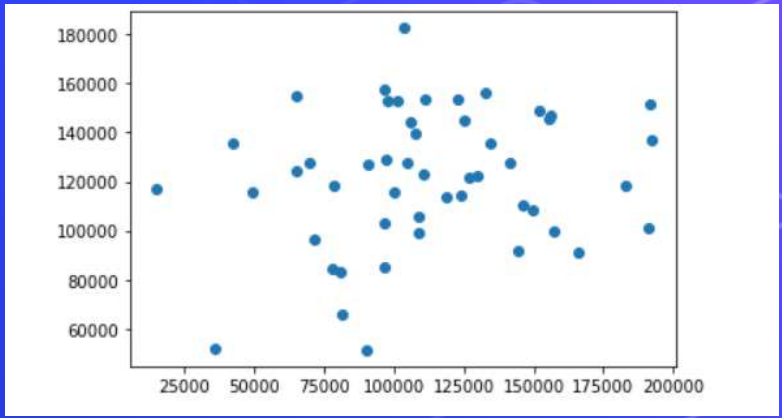
Heat map

Representation of data in the form of a map or diagram in which data values are represented as colours.



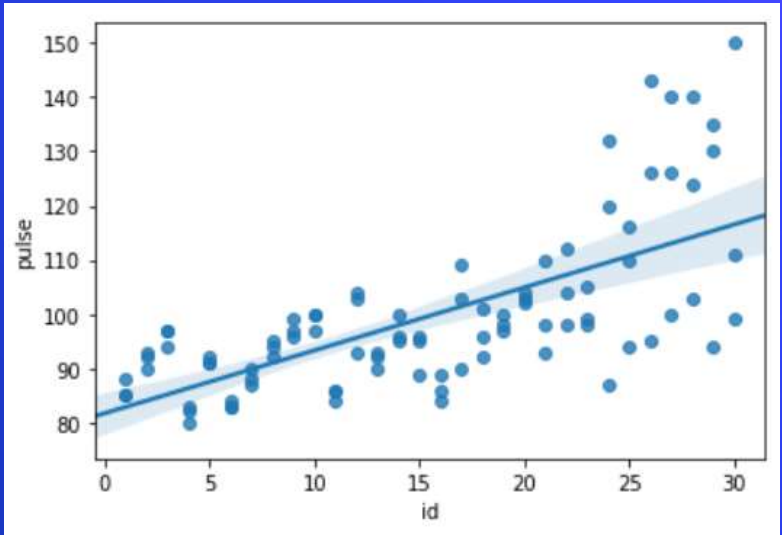
Scatter Plot

A graph in which the values of two variables are plotted along two axes, the pattern of the resulting points revealing any correlation present.



Reg Plot

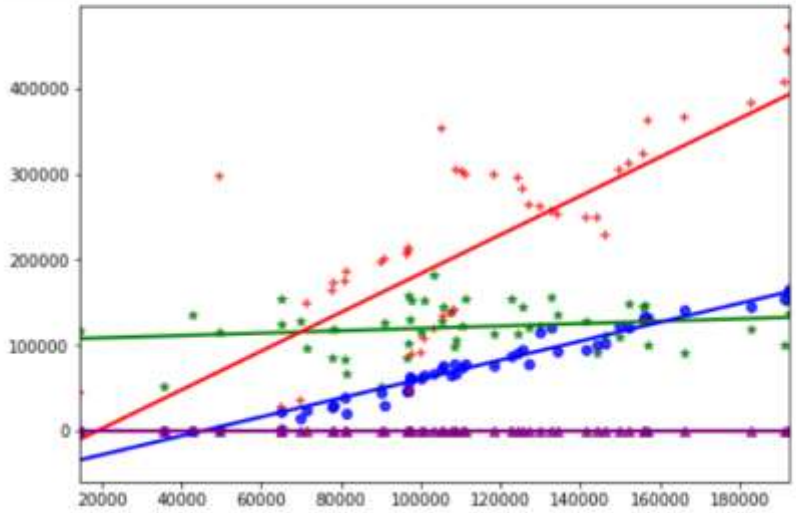
This method is used to plot data and a linear regression model fit. Here we have used **regplot** in this Project.



Example Plot

Reg Plot

```
In [30]: import seaborn as sns
axs = plt.subplots(figsize = (8,5.5))
sns.regplot(x,y,data = df,color='g',marker="*",ci=None)
sns.regplot(x1,y1,data = df,color='b',ci=None)
sns.regplot(x2,y2,data = df,color='red',marker="+",ci=None)
sns.regplot(x3,y3,data = df,color='purple',marker="^",ci=None)
plt.show()
```



Thank You!