In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('Students.csv')
df
```

Out[2]:

	Background	Machine Learning	Python	Remark
0	Tech	78.0	98.0	Pass
1	Tech	NaN	10.0	Fail
2	Non-Tech	69.0	NaN	Pass
3	Non-Tech	76.0	56.0	Pass
4	Tech	NaN	NaN	Pass
5	Tech	20.0	40.0	Fail
6	Tech	NaN	10.0	Fail
7	Non-Tech	54.0	20.0	Pass
8	Tech	87.0	98.0	Pass
9	Non-Tech	NaN	NaN	Fail

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Background        10 non-null     object 
 1   Machine Learning  6 non-null      float64
 2   Python            7 non-null      float64
 3   Remark            10 non-null     object 
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

In [4]:
```
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

In [5]:
```
df
```

Out[5]:

	Background	Machine Learning	Python	Remark
0	Tech	78.0	98.0	Pass
1	Tech	NaN	10.0	Fail
2	Non-Tech	69.0	NaN	Pass
3	Non-Tech	76.0	56.0	Pass
4	Tech	NaN	NaN	Pass
5	Tech	20.0	40.0	Fail
6	Tech	NaN	10.0	Fail
7	Non-Tech	54.0	20.0	Pass
8	Tech	87.0	98.0	Pass
9	Non-Tech	NaN	NaN	Fail

In [6]:
```
from sklearn.impute import SimpleImputer
si = SimpleImputer(missing_values=np.nan,strategy="mean")
df.iloc[:,1:3]=si.fit_transform(df.iloc[:,1:3])
```

In [7]:
```
df
```

Out[7]:

	Background	Machine Learning	Python	Remark
0	Tech	78.0	98.000000	Pass
1	Tech	64.0	10.000000	Fail
2	Non-Tech	69.0	47.428571	Pass
3	Non-Tech	76.0	56.000000	Pass
4	Tech	64.0	47.428571	Pass
5	Tech	20.0	40.000000	Fail
6	Tech	64.0	10.000000	Fail
7	Non-Tech	54.0	20.000000	Pass
8	Tech	87.0	98.000000	Pass
9	Non-Tech	64.0	47.428571	Fail

- Use OneHotEncoder to Features

In [8]:
```
background = pd.get_dummies(x['Background'])
```

In [9]:
```
df = pd.concat([background,df],axis=1)
```

In [10]:
```
df
```

Out[10]:

	Non-Tech	Tech	Background	Machine Learning	Python	Remark
0	0	1	Tech	78.0	98.000000	Pass
1	0	1	Tech	64.0	10.000000	Fail
2	1	0	Non-Tech	69.0	47.428571	Pass
3	1	0	Non-Tech	76.0	56.000000	Pass
4	0	1	Tech	64.0	47.428571	Pass
5	0	1	Tech	20.0	40.000000	Fail
6	0	1	Tech	64.0	10.000000	Fail
7	1	0	Non-Tech	54.0	20.000000	Pass
8	0	1	Tech	87.0	98.000000	Pass
9	1	0	Non-Tech	64.0	47.428571	Fail

In [11]:
```
df.drop('Background',axis=1,inplace=True)
```

In [12]:
```
df
```

Out[12]:

	Non-Tech	Tech	Machine Learning	Python	Remark
0	0	1	78.0	98.000000	Pass
1	0	1	64.0	10.000000	Fail
2	1	0	69.0	47.428571	Pass
3	1	0	76.0	56.000000	Pass
4	0	1	64.0	47.428571	Pass
5	0	1	20.0	40.000000	Fail
6	0	1	64.0	10.000000	Fail
7	1	0	54.0	20.000000	Pass
8	0	1	87.0	98.000000	Pass
9	1	0	64.0	47.428571	Fail

- Use LabelEncoder to Target

In [13]:
```
y = df[["Remark"]]
```

In [14]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y[["Remarks"]]= le.fit_transform(y[["Remark"]])
```

In [15]:
```
df = pd.concat([df,y],axis=1)
```

In [16]:
```
df
```

Out[16]:

	Non-Tech	Tech	Machine Learning	Python	Remark	Remark	Remarks
0	0	1	78.0	98.000000	Pass	Pass	1
1	0	1	64.0	10.000000	Fail	Fail	0
2	1	0	69.0	47.428571	Pass	Pass	1
3	1	0	76.0	56.000000	Pass	Pass	1
4	0	1	64.0	47.428571	Pass	Pass	1
5	0	1	20.0	40.000000	Fail	Fail	0
6	0	1	64.0	10.000000	Fail	Fail	0
7	1	0	54.0	20.000000	Pass	Pass	1
8	0	1	87.0	98.000000	Pass	Pass	1
9	1	0	64.0	47.428571	Fail	Fail	0

In [17]:
```
df.drop('Remark',axis=1,inplace=True)
```

In [18]:
```
df
```

Out[18]:

	Non-Tech	Tech	Machine Learning	Python	Remarks
0	0	1	78.0	98.000000	1
1	0	1	64.0	10.000000	0
2	1	0	69.0	47.428571	1
3	1	0	76.0	56.000000	1
4	0	1	64.0	47.428571	1
5	0	1	20.0	40.000000	0
6	0	1	64.0	10.000000	0
7	1	0	54.0	20.000000	1
8	0	1	87.0	98.000000	1
9	1	0	64.0	47.428571	0

Do Feature Scaling with StandardScaler

In [19]:
```
from sklearn.preprocessing import StandardScaler
for col in df:
    ss = StandardScaler()
    df[col]=ss.fit_transform(df[[col]])
```

In [20]:
```
df
```

Out[20]:

	Non-Tech	Tech	Machine Learning	Python	Remarks
0	-0.816497	0.816497	0.817889	1.704984	0.816497
1	-0.816497	0.816497	0.000000	-1.261881	-1.224745
2	1.224745	-1.224745	0.292103	0.000000	0.816497
3	1.224745	-1.224745	0.701047	0.288980	0.816497
4	-0.816497	0.816497	0.000000	0.000000	0.816497
5	-0.816497	0.816497	-2.570507	-0.250450	-1.224745
6	-0.816497	0.816497	0.000000	-1.261881	-1.224745
7	1.224745	-1.224745	-0.584206	-0.924737	0.816497
8	-0.816497	0.816497	1.343674	1.704984	0.816497
9	1.224745	-1.224745	0.000000	0.000000	-1.224745

In [21]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Non-Tech          10 non-null     float64
 1   Tech              10 non-null     float64
 2   Machine Learning  10 non-null     float64
 3   Python            10 non-null     float64
 4   Remarks           10 non-null     float64
dtypes: float64(5)
memory usage: 528.0 bytes
```

In [25]:
```
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

- Split the Training and Testing set

In [26]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.3,random_state=1)
```

In [27]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(xtrain,ytrain)
ypred = linreg.predict(xtest)
```

Checking Accuracy

In [28]:
```
from sklearn.metrics import mean_absolute_error as mae , mean_squared_error as mse , r2_score
```

In [29]:
```
print(f"MAE :- {mae(ytest,ypred)}")
print(f"MSE :- {mse(ytest,ypred)}")
print(f"RMSE :- {mse(ytest,ypred)**0.5}")
print(f"Accuracy :- {r2_score(ytest,ypred)}")
```

```
MAE :- 0.9670826569088663
MSE :- 1.6691311803567157
RMSE :- 1.2919485981867153
Accuracy :- -0.8026616747852522
```

In [ ]:
```

```