```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

#pandas : It used for data manipulation and analysis.
#numPy : It is a powerful Python library for numerical computing.
#seaborn : It provides a high-level interface for creating attractive and informative statistical graphics.
#matplotlib.pyplot : It provides a MATLAB-like interface for creating basic plots and visualizations.
```

```
#read_file
df=pd.read_csv("student.csv")
print(df)                    #print student data
```

```
    id          name  class  mark  gender
0    1      John Deo   Four    75  female
1    2      Max Ruin  Three    85    male
2    3        Arnold  Three    55    male
3    4    Krish Star   Four    60  female
4    5     John Mike   Four    60  female
5    6     Alex John   Four    55    male
6    7   My John Rob  Fifth    78    male
7    8        Asruid   Five    85    male
8    9       Tes Qry    Six    78    male
9   10      Big John   Four    55  female
10  11        Ronald    Six    89  female
11  12         Recky    Six    94  female
12  13           Kty  Seven    88  female
13  14          Bigy  Seven    88  female
14  15      Tade Row   Four    88    male
15  16         Gimmy   Four    88    male
16  17         Tumyu    Six    54    male
17  18         Honny   Five    75    male
18  19         Tinny   Nine    18    male
19  20        Jackly   Nine    65  female
20  21    Babby John   Four    69  female
21  22        Reggid  Seven    55  female
22  23         Herod  Eight    79    male
23  24     Tiddy Now  Seven    78    male
24  25      Giff Tow  Seven    88    male
25  26        Crelea  Seven    79    male
26  27      Big Nose  Three    81  female
27  28     Rojj Base  Seven    86  female
28  29   Tess Played  Seven    55    male
29  30     Reppy Red    Six    79  female
30  31   Marry Toeey   Four    88    male
31  32     Binn Rott  Seven    90  female
32  33     Kenn Rein    Six    96  female
33  34      Gain Toe  Seven    69    male
34  35    Rows Noump    Six    88  female
```

```
df = pd.read_csv('student.csv', usecols=['id'])
df.columns = df.columns.str.strip()
```

```
print(df.head())#Print first five rows
```

```
   id        name  class  mark  gender
0   1    John Deo   Four    75  female
1   2    Max Ruin  Three    85    male
2   3      Arnold  Three    55    male
3   4  Krish Star   Four    60  female
4   5   John Mike   Four    60  female
```

```
print(df.tail(5))   #print last 5 rows
```

```
    id         name  class  mark  gender
30  31  Marry Toeey   Four    88    male
31  32    Binn Rott  Seven    90  female
32  33    Kenn Rein    Six    96  female
33  34     Gain Toe  Seven    69    male
34  35   Rows Noump    Six    88  female
```

```
print(df.sample())#print random five rows
```

```
   id       name class  mark gender
5   6  Alex John  Four    55   male
```

```
print(df.info())#print information about dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      35 non-null     int64 
 1   name    35 non-null     object
 2   class   35 non-null     object
 3   mark    35 non-null     int64 
 4   gender  35 non-null     object
dtypes: int64(2), object(3)
memory usage: 1.5+ KB
None
```

```
print(df.describe())#print descriptive statistics that generate statistics that summarize central tendency of data
```

```
              id       mark
count  35.000000  35.000000
mean   18.000000  74.657143
std    10.246951  16.401117
min     1.000000  18.000000
25%     9.500000  62.500000
50%    18.000000  79.000000
75%    26.500000  88.000000
max    35.000000  96.000000
```

```
df.dtypes#printing data types of the respective data
```

```
id         int64
name      object
class     object
mark       int64
gender    object
dtype: object
```

```
print(df.info()) #print information about dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      35 non-null     int64 
 1   name    35 non-null     object
 2   class   35 non-null     object
 3   mark    35 non-null     int64 
 4   gender  35 non-null     object
dtypes: int64(2), object(3)
memory usage: 1.5+ KB
None
```

```
print(df.isnull().sum())#print number of missing values in the given dataset
```

```
id        0
name      0
class     0
mark      0
gender    0
dtype: int64
```

```
df=df.dropna()
# drop on the rows missing values
```

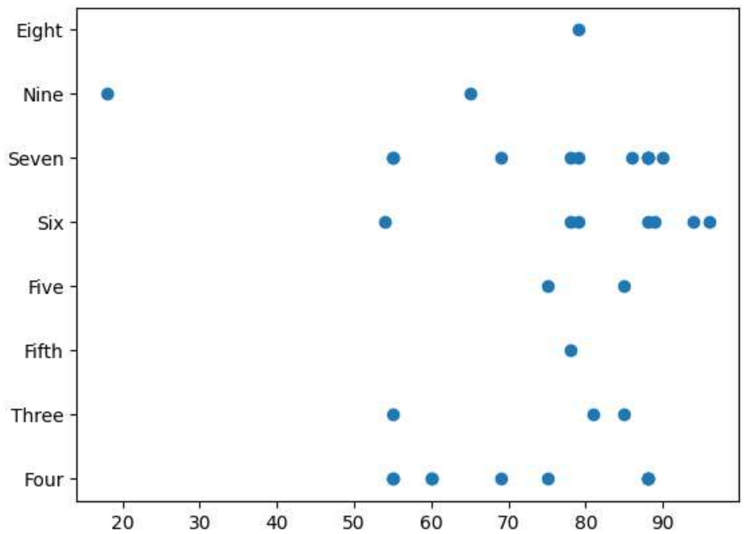
```
df.count()#count no of rows in a particular time
```

```
id        35
name      35
class     35
mark      35
gender    35
dtype: int64
```

```
#visualization
```

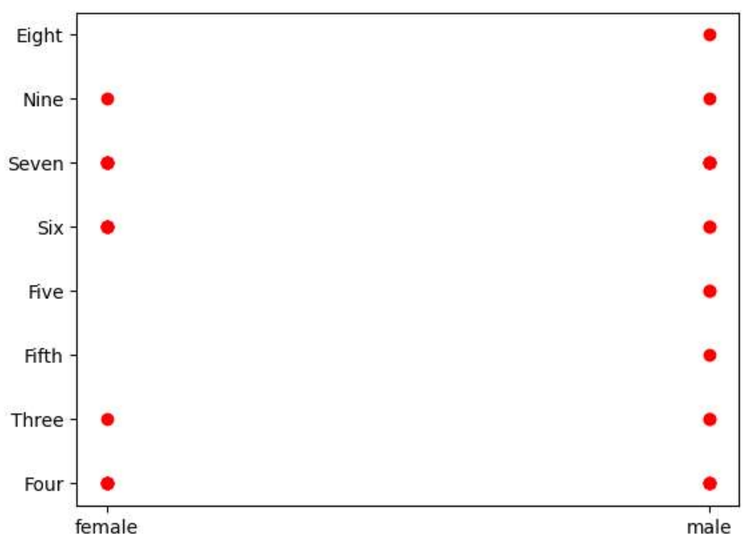
```
plt.scatter(df['mark'],df['class'] )# scatter plot for marks and class
```

<matplotlib.collections.PathCollection at 0x7e7be7abd960>




```
plt.scatter (df['gender'],df['class'],color='red' )#scatter plot for gender and class ratio
```

<matplotlib.collections.PathCollection at 0x7e7be5897400>

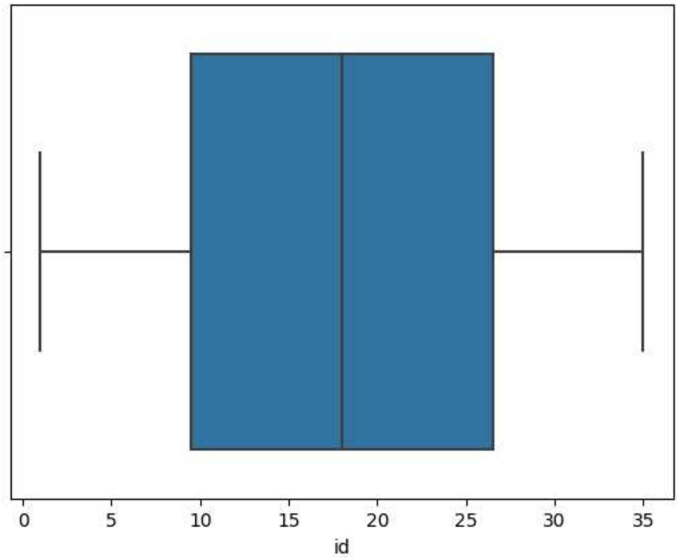



```
plt.scatter (df['name'],df['class'],color='BLACK' )#scatter plot for gender and class ratio
```

```
<matplotlib.collections.PathCollection at 0x7e7be55f3790>
```

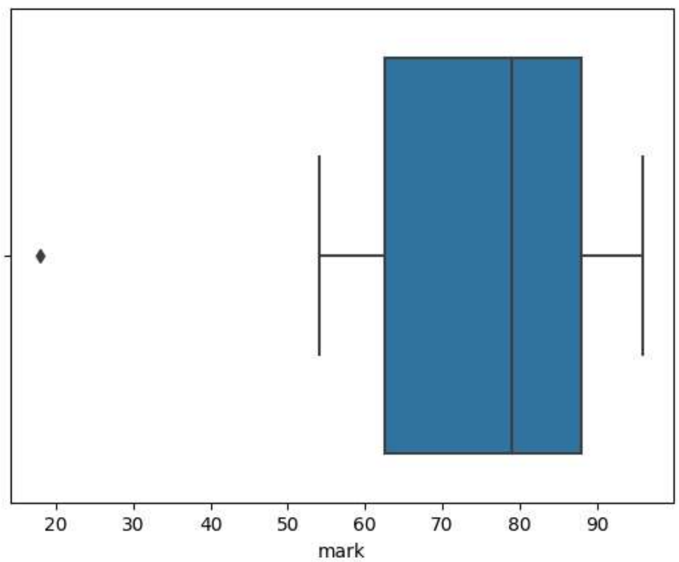
```
sns.boxplot(x=df['id'])#boxplot of id column
```

```
<Axes: xlabel='id'>
```




```
sns.boxplot(x=df['mark'])#boxplot of mark column
```

```
<Axes: xlabel='mark'>
```



Double-click (or enter) to edit

This Calculates Interquartile Range (IQR) for each column in 25 % and 75 % and then it will substract to get the 50% of Interquartile Range (IQR)

```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
id      17.0
mark    25.5
dtype: float64
<ipython-input-11-d7397e803310>:1: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a futu
  Q1 = df.quantile(0.25)
<ipython-input-11-d7397e803310>:2: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a futu
  Q3 = df.quantile(0.75)
```

(df < (Q1 - 1.5 * IQR)): This part of the expression checks if any value in the DataFrame is less than the lower bound (Q1 - 1.5 * IQR). (df > (Q3 + 1.5 * IQR)): This part of the expression checks if any value in the DataFrame is greater than the upper bound (Q3 + 1.5 * IQR). The | operator is used to combine these two conditions with an OR operation.

```
df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
df.shape
```

```
<ipython-input-12-f4e1682787c4>:1: FutureWarning: Automatic reindexing on DataFrame vs Series comparisons is deprecated and will ra
  df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
(34, 5)
```

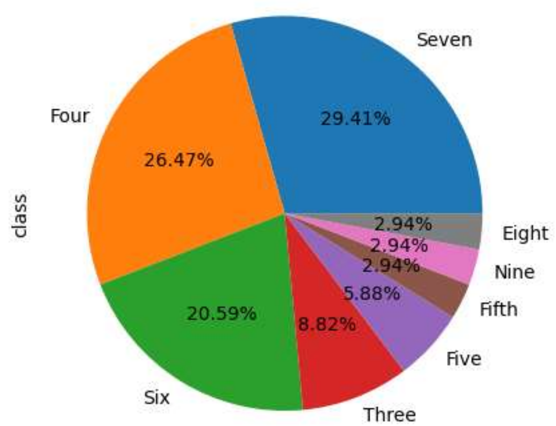
Pie Chart It is univarient analysis , as single variable is used. In this region and gas can show the region in the pie format. It is called a "pie chart" because the chart resembles a pie that is divided into slices, with each slice representing a particular category or data point. The size of each slice corresponds to the proportion or percentage of the whole that each category represents.

```
pie=df["class"].value_counts()
pie
```

```
Seven    10
Four      9
Six       7
Three     3
Five      2
Fifth     1
Nine      1
Eight     1
Name: class, dtype: int64
```

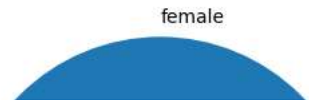
```
pie.plot(kind="pie",autopct="%.2f%%")
```

```
<Axes: ylabel='class'>
```




```
df["gender"].value_counts().plot(kind="pie",autopct="%.4f%%")
```

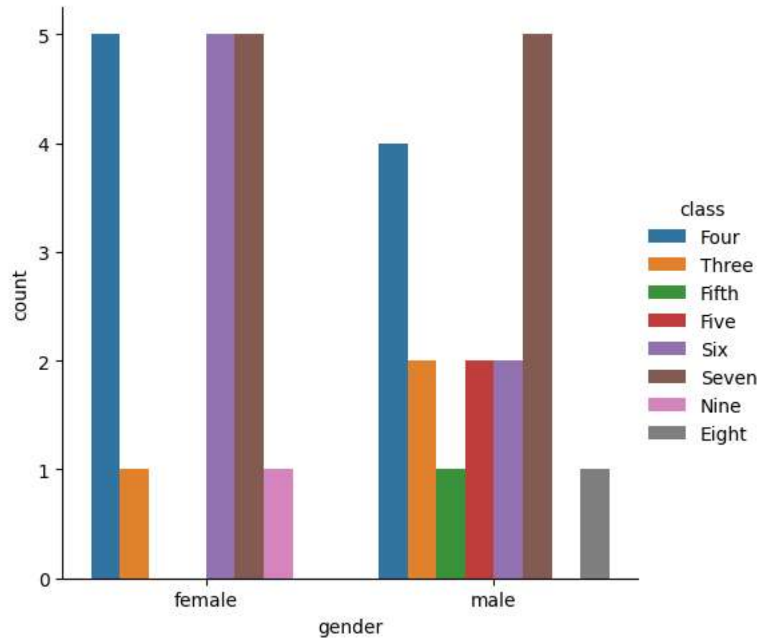
```
<Axes: ylabel='gender'>
```



Count Plot In the Countplots height of each bar represents the number of occurrences of each category in the dataset. Countplots are particularly useful for visualizing the frequency of different categories and identifying the most common or least common categories in the data. In this the number of the gases released in each year.

```
sns.catplot(x ='gender', hue ='class',kind ='count', data = df)
```

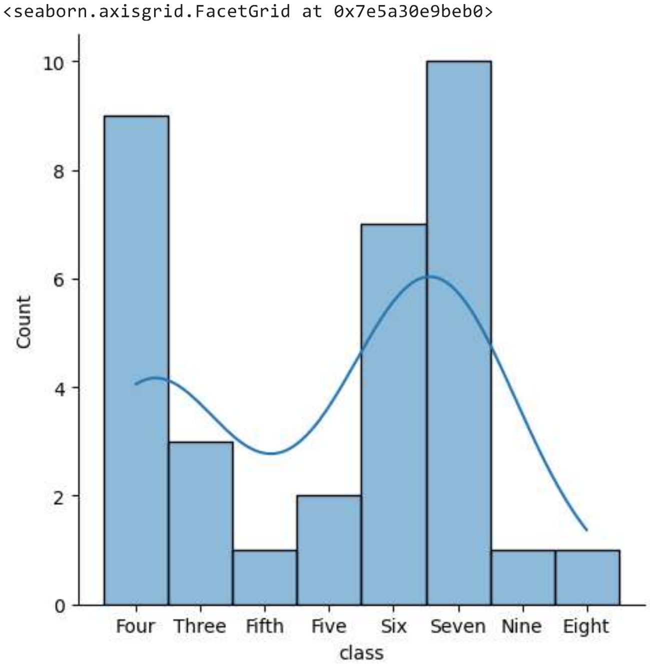
```
<seaborn.axisgrid.FacetGrid at 0x7e5a3393b940>
```



displot is used to create a histogram to visualize the distribution of a numerical variable. It is used to create a KDE plot to visualize the estimated probability density function of a numerical variable. KDE plots show the smoothed continuous representation of the data distribution.

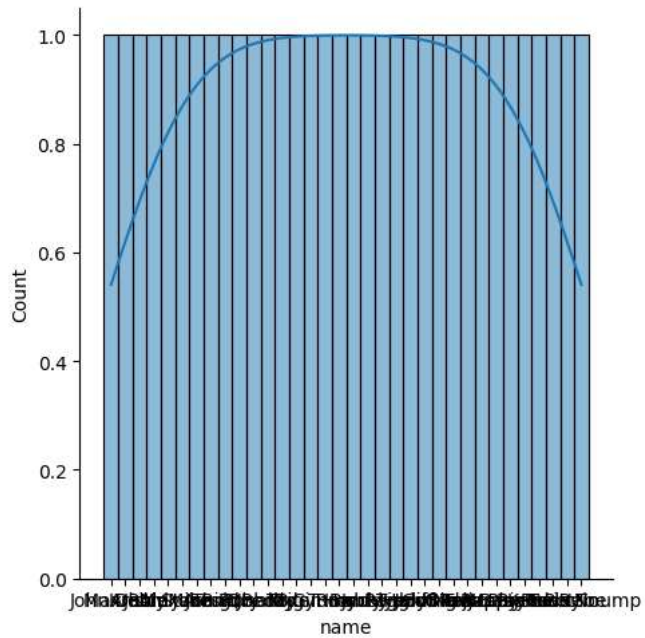
```
sns.displot(df['class'],kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7e5a30e9beb0>
```



```
sns.displot(df['name'],kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7e5a30e9b1f0>
```

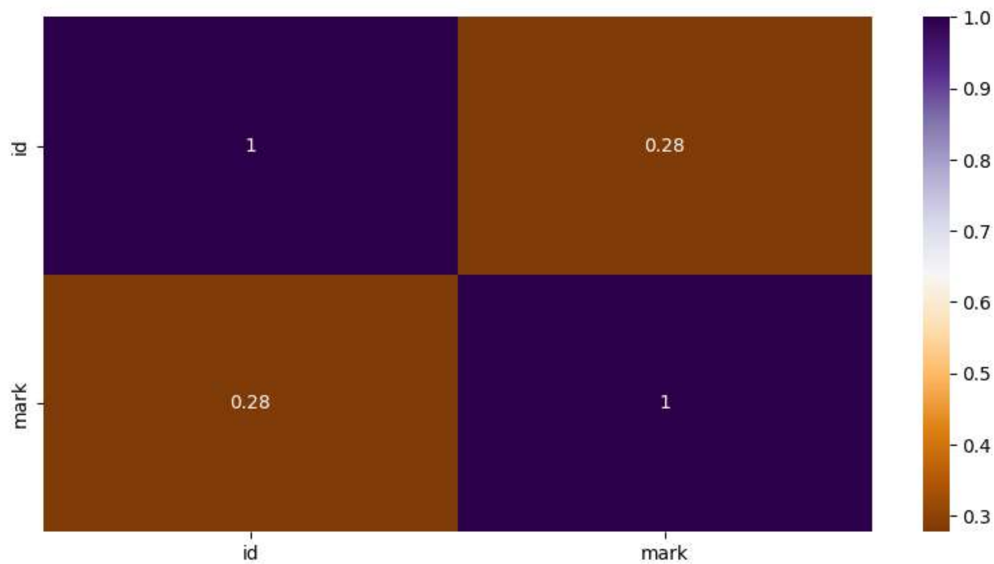


Heatmaps The colors on a heatmap are used to represent the magnitude or density of the data points at different locations, with warmer colors typically indicating higher values, and cooler colors indicating lower values.

```
plt.figure(figsize=(10,5))
c= df.corr()
sns.heatmap(c,cmap="PuOr",annot=True)
c
```

```
<ipython-input-28-38bf0624e2ab>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future v
  c= df.corr()
```

	id	mark
id	1.000000	0.277559
mark	0.277559	1.000000



DATA PREPROCESSING

```
x=df.iloc[:,2:3]#In x axis we r including all the columns except the targeted row which will be the y axis
x
```

	class	mark	gender
0	Four	75	female
1	Three	85	male
2	Three	55	male
3	Four	60	female
4	Four	60	female
5	Four	55	male
6	Fifth	78	male
7	Five	85	male
8	Six	78	male
9	Four	55	female
10	Six	89	female
11	Six	94	female
12	Seven	88	female
13	Seven	88	female
14	Four	88	male
15	Four	88	male
16	Six	54	male
17	Five	75	male
19	Nine	65	female
20	Four	69	female
21	Seven	55	female
22	Eight	79	male
23	Seven	78	male
24	Seven	88	male
25	Seven	79	male
26	Three	81	female
27	Seven	86	female
28	Seven	55	male
29	Six	79	female
30	Four	88	male
31	Seven	90	female
32	Six	96	female
33	Seven	69	male
34	Six	88	female

```
y=df.iloc[:,5:5]#In y axis we r including the targeted row only
y
```

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
19
20
21
22
23
24
25
26

.....ENCODING In Encoding we will convert all the object datatype of the column to the integer for the Machine Learning Process

30

```
from sklearn.preprocessing import OrdinalEncoder
```

OriginalEncoder –> It is usedfor encoding categorical features into ordinal integers region anzsic_descriptor gas magnitude year

```
oe=OrdinalEncoder()
x[["class","gender"]]=oe.fit_transform(x[["class","gender"]])
x
#Fit : to perform calculations on data
#Transform : apply that calculation
#Converting the Object data type of columns to the integer using fit_transform with OriginalEncoder of X-axis
```

	class	mark	gender
0	3.0	75	0.0
1	7.0	85	1.0
2	7.0	55	1.0
3	3.0	60	0.0
4	3.0	60	0.0
5	3.0	55	1.0
6	1.0	78	1.0
7	2.0	85	1.0
8	6.0	78	1.0
9	3.0	55	0.0
10	6.0	89	0.0
11	6.0	94	0.0
12	5.0	88	0.0
13	5.0	88	0.0
14	3.0	88	1.0
15	3.0	88	1.0
16	6.0	54	1.0
17	2.0	75	1.0
19	4.0	65	0.0
20	3.0	69	0.0
21	5.0	55	0.0
22	0.0	79	1.0
23	5.0	78	1.0
24	5.0	88	1.0
25	5.0	79	1.0
26	7.0	81	0.0
28	5.0	55	1.0

```
from sklearn.model_selection import train_test_split
```

```
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.2, random_state=1)
```

LogisticRegression --> It is used to perform logistic regression

classification_report --> It provides a comprehensive report with metrics such as precision, recall, F1-score, and support for both classes (positive and negative). It helps you understand the performance of the model for each class and overall accuracy.

confusion_matrix --> It provides a matrix that compares the predicted class labels against the actual class labels. It helps to visualize the number of true positives, false positives, true negatives, and false negatives, which is useful for understanding the model's performance and identifying potential areas for improvement.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix
```

```
#step1 -: import the model
from sklearn.linear_model import LinearRegression
```

```
#step2 -: initalize the model
linreg = LinearRegression()
```

```
#step3 -: train the model -> m & c
linreg.fit(xtrain, ytrain)
```

```
▾ LinearRegression
LinearRegression()
```

```
#step4 -: make prediction
ypred = linreg.predict(xtest)
```

```
#Accuracy : It is used to see how much accurate the data it is providing by comparing the ypred and ytest
from sklearn.metrics import r2_score
print(f"Accuracy : {r2_score(ytest, ypred)}")
```

```
Accuracy : 0.7104365736488776
```

```
#linreg.intercept_ : It provides the value of the intercept
linreg.intercept_
```

```
33.98824601307956
```

```
#linreg.coef_ : It represents the coefficients (or weights) associated
#with each input feature in the linear regression equation.
linreg.coef_
```

```
array([0.11888612, 3.53655888])
```

```
from sklearn.linear_model import LinearRegression
```

object of linear regression class training of model testing of model ypred stores the predicted value of y store hai

```
model=LinearRegression()
model.fit(x,y)
ypred=model.predict(x)
```

```
ypred
```

```
array([ 70.29835968,  80.64856285, 108.09519021,  73.99469048,
        60.40989978, 104.95295132,  74.27173643,  91.46050928,
       101.81071244,  84.8066369 ,  47.28685234,  57.6370555 ,
        50.89083448, 102.2724557 ,  85.26838016,  95.61858333,
        82.03379263,  65.02971709,  37.76778704,  54.95655988,
        41.37176918,  65.3991117 ,  99.68430873,  69.00309384,
        82.77258184,  65.7685063 , 100.05370334,  89.88819748,
        86.56126129,  69.55718575,  83.32667375,  79.99973757,
        45.89923784,  90.44228939,  59.76107449])
```

```
#The sklearn. metrics module implements several loss,
# score, and utility functions to measure classification performance.
from sklearn import metrics
r2_score(y,ypred)
```

```
0.8615689111808863
```

```
#shows the point where the estimated regression line crosses the y axis
model.intercept_
```

```
36.1055113016053
```

```
#It is used to estimate the coefficients for the linear regression problem.
model.coef_
```

```
array([0.09234865, 3.41928484])
```

● ×