Aim: To implement Linear Regression.

1.) Implement **Linear Regression** and calculate sum of residual error on the following datasets.

- Y=4X + 13 +s(0,1)
- Y=10sin(X1)+15sin(X2)+s(0,1)
- Boston Housing Rate Dataset

Where X is random variable with values [0,50] and s(0,1) Guassian white noise of zero mean and unit variance.

```
import numpy as np

# Generate random data with noise
np.random.seed(0)
X = np.random.rand(100) * 50
noise = np.random.normal(0, 1, 100)
Y = 4 * X + 13 + noise

# Formulate the linear regression problem using the normal equation
X_b = np.c_[np.ones((100, 1)), X]
theta = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(Y)

# Print the computed weights
print("Weights (Analytic Matrix Formulation):", theta)
# Define the LMS loss function
def lms_loss(theta, X, Y):
    error = X.dot(theta) - Y
    return np.mean(error**2)

# Gradient Descent - Full Batch
def gradient_descent_full_batch(X, Y, learning_rate, n_iterations):
    theta = np.random.randn(2)
    for iteration in range(n_iterations):
        gradient = 2/X.shape[0] * X.T.dot(X.dot(theta) - Y)
        theta -= learning_rate * gradient
    return theta

# Gradient Descent - Stochastic
def gradient_descent_stochastic(X, Y, learning_rate, n_iterations):
    theta = np.random.randn(2)
    for iteration in range(n_iterations):
        random_index = np.random.randint(X.shape[0])
        xi = X[random_index:random_index+1]
        yi = Y[random_index:random_index+1]
        gradient = 2 * xi.T.dot(xi.dot(theta) - yi)
        theta -= learning_rate * gradient
    return theta
```

```python
# Gradient Descent - Mini-Batch
def gradient_descent_mini_batch(X, Y, learning_rate, n_iterations, batch_size):
    theta = np.random.randn(2)
    for iteration in range(n_iterations):
        random_indices = np.random.choice(X.shape[0], batch_size)
        xi = X[random_indices]
        yi = Y[random_indices]
        gradient = 2/batch_size * xi.T.dot(xi.dot(theta) - yi)
        theta -= learning_rate * gradient
    return theta

# Example usage and comparison
learning_rate = 0.01
n_iterations = 1000
batch_size = 32

theta_full_batch = gradient_descent_full_batch(X_b, Y, learning_rate, n_iterations)
theta_stochastic = gradient_descent_stochastic(X_b, Y, learning_rate, n_iterations)
theta_mini_batch = gradient_descent_mini_batch(X_b, Y, learning_rate, n_iterations, batch_size)

print("Weights (Full Batch):", theta_full_batch)
print("Weights (Stochastic):", theta_stochastic)
print("Weights (Mini-Batch):", theta_mini_batch)
```

Output:-

```
PS C:\Users\HP\Desktop\Python> python -u "c:\Users\HP\Desktop\Python\exp_5.py"
Weights (Analytic Matrix Formulation): [13.22215108  3.9987387 ]
```

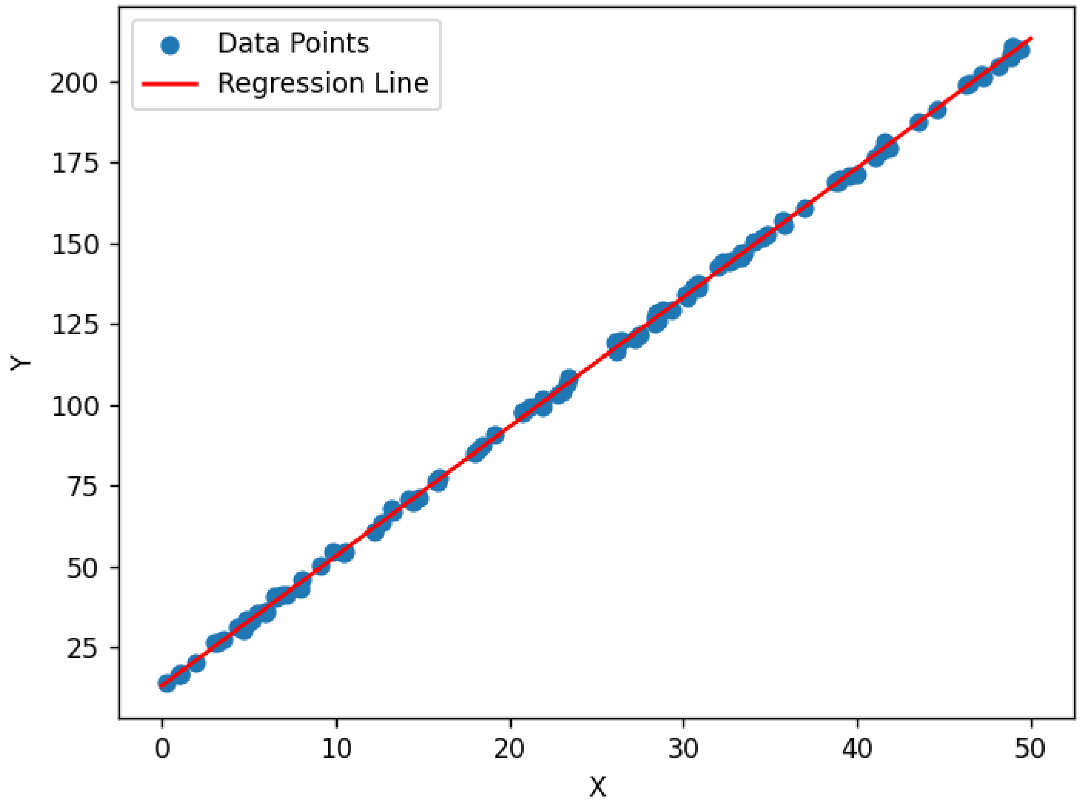
```python
# Decaying Learning Rate
def gradient_descent_decaying_learning_rate(X, Y, initial_learning_rate, n_iterations):
    theta = np.random.randn(2)
    for iteration in range(n_iterations):
        learning_rate = initial_learning_rate / (iteration + 1)  # Decaying learning rate
        gradient = 2/X.shape[0] * X.T.dot(X.dot(theta) - Y)
        theta -= learning_rate * gradient
    return theta

# Example usage with decaying learning rate
theta_decaying_lr = gradient_descent_decaying_learning_rate(X_b, Y, initial_learning_rate=0.1, n_iterations=n_iterations)
print("Weights (Decaying Learning Rate):", theta_decaying_lr)
```

Output:-

```
c:\Users\HP\Desktop\Python\exp_5.py:25: RuntimeWarning: invalid value encountered in subtract
  theta -= learning_rate * gradient
c:\Users\HP\Desktop\Python\exp_5.py:36: RuntimeWarning: invalid value encountered in subtract
  theta -= learning_rate * gradient
c:\Users\HP\Desktop\Python\exp_5.py:47: RuntimeWarning: invalid value encountered in subtract
  theta -= learning_rate * gradient
Weights (Full Batch): [nan nan]
Weights (Stochastic): [nan nan]
Weights (Mini-Batch): [nan nan]
Weights (Decaying Learning Rate): [ 3.67663635e+28 -1.13412350e+27]
```

```
import matplotlib.pyplot as plt

# Plot data points
plt.scatter(X, Y, label='Data Points')

# Plot regression line
X_range = np.linspace(0, 50, 100)
Y_pred = theta[0] + theta[1] * X_range
plt.plot(X_range, Y_pred, color='red', label='Regression Line')

# Add labels and legend
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()

# Show the plot
plt.show()
```

Output:-



<u>Boston Housing Rate Dataset:</u>

[1]
```
import pandas as pd
```

[3]
```
df=pd.read_csv("HousingData.csv")
df
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	NaN	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	NaN	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	NaN	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

[6]
```
from sklearn.datasets import load_boston
```

[7]
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[8]
```
df= load_boston()
```

[9]
```
df
```

Output:-

```
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
         4.9800e+00],
        [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
         9.1400e+00],
        [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
         4.0300e+00],
        ...,
        [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
         5.6400e+00],
        [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
         6.4800e+00],
        [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
         7.8800e+00]]),
 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
        18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
        15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
        13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
        21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
        35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
        19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
        20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
        23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
        33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
        21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
        20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
...
        23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9]),
 'feature_names': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
        'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'),
```

```python
dataset=pd.DataFrame(df.data)
```

```python
dataset
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

506 rows × 13 columns

```python
dataset.columns=df.feature_names
```

```python
dataset.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

```python
X=dataset
Y=df.target
```

```
X
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

506 rows × 13 columns

```
Y
```

```
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
       18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
       15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
       13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
       21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
       35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
       19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
       20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
       23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
       33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
       21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
       20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
       23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
       15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
       17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
       25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
       23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
       32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
       34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
       20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
       26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
       31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
       22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
       42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
       36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
...
       15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
       19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
       29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
       20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
       23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9])
```

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test= train_test_split(
    X, Y, test_size=0.30, random_state=42)
```

```python
X_train
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21
116	0.13158	0.0	10.01	0.0	0.547	6.176	72.5	2.7301	6.0	432.0	17.8	393.30	12.04
45	0.17142	0.0	6.91	0.0	0.448	5.682	33.8	5.1004	3.0	233.0	17.9	396.90	10.21
16	1.05393	0.0	8.14	0.0	0.538	5.935	29.3	4.4986	4.0	307.0	21.0	386.85	6.58
468	15.57570	0.0	18.10	0.0	0.580	5.926	71.0	2.9084	24.0	666.0	20.2	368.74	18.13
...	...	...	...	...	...	...	...	...	...	...	...	...	...
106	0.17120	0.0	8.56	0.0	0.520	5.836	91.9	2.2110	5.0	384.0	20.9	395.67	18.66
270	0.29916	20.0	6.96	0.0	0.464	5.856	42.1	4.4290	3.0	223.0	18.6	388.65	13.00
348	0.01501	80.0	2.01	0.0	0.435	6.635	29.7	8.3440	4.0	280.0	17.0	390.94	5.99
435	11.16040	0.0	18.10	0.0	0.740	6.629	94.6	2.1247	24.0	666.0	20.2	109.85	23.27
102	0.22876	0.0	8.56	0.0	0.520	6.405	85.4	2.7147	5.0	384.0	20.9	70.80	10.63

354 rows × 13 columns

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```python
X_train=scaler.fit_transform(X_train)
```

```python
X_train
```

```
array([[-0.41425879, -0.50512499, -1.29214218, ...,  0.18727079,
         0.39651419, -1.01531611],
       [-0.40200818, -0.50512499, -0.16208345, ..., -0.21208981,
         0.3870674 , -0.05366252],
       [-0.39721053, -0.50512499, -0.60948856, ..., -0.16771641,
         0.42854113, -0.31132373],
       ...,
       [-0.41604586,  3.03838247, -1.3166773 , ..., -0.56707702,
         0.35987906, -0.90549329],
       [ 0.92611293, -0.50512499,  1.00549958, ...,  0.8528718 ,
        -2.87841346,  1.52750437],
       [-0.39030549, -0.50512499, -0.37135358, ...,  1.16348561,
        -3.32828832, -0.25218837]])
```

[38]
```python
X_test=scaler.transform(X_test)
```

[40]
```python
y_train
```

```
array([28.7, 21.2, 19.3, 23.1, 19.1, 25. , 33.4,  5. , 29.6, 18.7, 21.7,
       23.1, 22.8, 21. , 48.8, 14.6, 16.6, 27.1, 20.1, 19.8, 21. , 41.3,
       23.2, 20.4, 18.5, 29.4, 36.4, 24.4, 11.8, 13.8, 12.3, 17.8, 33.1,
       26.7, 13.4, 14.4, 50. , 22. , 19.9, 23.8, 17.5, 12.7,  5.6, 31.1,
       26.2, 19.4, 16.7, 13.8, 22.9, 15.3, 27.5, 36.1, 22.9, 24.5, 25. ,
       50. , 34.9, 31.7, 24.1, 22.1, 14.1, 42.8, 19.3, 32.2, 26.4, 21.8,
       21.7,  8.3, 46.7, 43.1, 31.5, 10.5, 16.7, 20. , 33.3, 17.8, 50. ,
       20.5, 23.2, 13.1, 19.6, 22.8, 28.7, 30.7, 22.9, 21.9, 23.9, 32.7,
       24.3, 21.5, 24.6,  8.5, 26.4, 23.1, 15. ,  8.8, 19.3, 23.9, 24.7,
       19.8, 23.8, 13.3, 29. , 27.1, 34.6, 13.3, 15.6, 12.5, 14.6, 11. ,
       24.8, 17.3,  8.1, 21.4, 15.6, 23.3, 32. , 38.7, 30.1, 20.5, 32.5,
       42.3, 24.3, 20.6, 22. , 18.2, 15. ,  6.3, 20.1, 21.4, 28.4, 30.1,
       20.8, 23. , 14.3, 11.7, 37.3, 17.1, 10.4, 23. , 22.7, 20.3, 21.7,
       50. ,  8.4, 18.8, 37.2, 16.1, 16.5, 22.2, 20.6, 13.5, 48.3, 23.8,
       22.7, 17.4, 30.3, 36. , 41.7, 18.3, 22. , 18.6, 44.8, 11.9, 18.7,
       16.2, 22. ,  7.2, 20.4, 13.8, 13. , 18.4, 23.1, 21.2, 23.1, 23.5,
       50. , 26.6, 22.2, 50. ,  8.3, 23.3, 21.7, 18.9, 18.4, 17.4, 13.4,
       12.1, 26.6, 21.7, 28.4, 20.5, 22. , 13.9, 11.3, 29.9, 26.6, 10.5,
       23.2, 24.4, 46. , 21.9,  7.5, 36.2, 44. , 17.8, 27.5, 37.6, 14.1,
       28.1, 10.2, 19.1, 43.8, 27.9, 25. , 16. , 16.6, 13.2, 50. , 22.2,
       32.9, 15.2, 14.8, 13.8, 24.3, 33.8, 22.3, 50. ,  9.5, 13.3, 22.2,
       18.1, 18. , 25. , 16.5, 23. , 20.1, 33. , 24.8, 18.2, 13.1, 34.9,
       10.2, 19.9, 27.9, 23.3, 35.1, 12.8, 22. , 18.5, 25.1, 22.5, 22.4,
```

[41]
```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
```

[42]
```python
regression=LinearRegression()
regression.fit(X_train,y_train)
```

```
LinearRegression()
```

[45]
```python
cross_val_score(regression,X_train,y_train,scoring='neg_mean_squared_error',cv=5)
```

```
array([-24.85792467, -32.34889563, -29.41534458, -18.46226827,
       -24.80445401])
```

[46]
```python
mse=cross_val_score(regression,X_train,y_train,scoring='neg_mean_squared_error', cv=10)
```

[47]
```python
np.mean(mse)
```

```
-25.55066079166079
```

```
reg_pred=regression.predict(X_test)
```

[48]

```
reg_pred
```

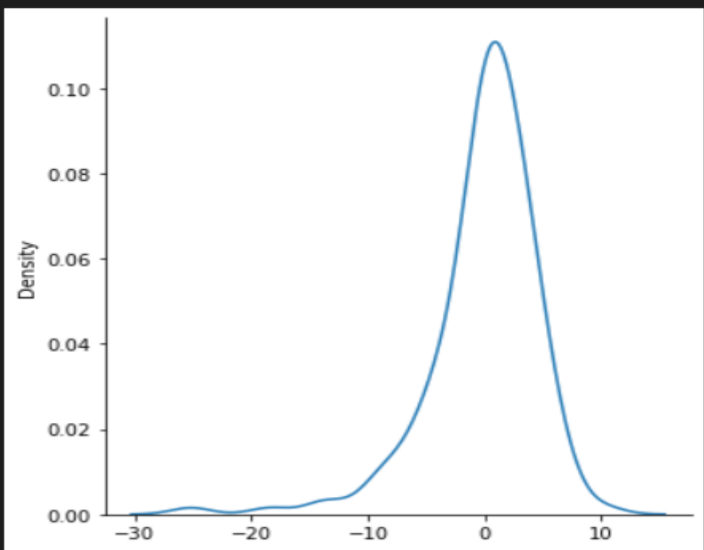
[49]

```
array([28.64896005, 36.49501384, 15.4111932 , 25.40321303, 18.85527988,
       23.14668944, 17.3921241 , 14.07859899, 23.03692679, 20.59943345,
       24.82286159, 18.53057049, -6.86543527, 21.80172334, 19.22571177,
       26.19191985, 20.27733882,  5.61596432, 40.44887974, 17.57695918,
       27.44319095, 30.1715964 , 10.94055823, 24.02083139, 18.07693812,
       15.934748  , 23.12614028, 14.56052142, 22.33482544, 19.3257627 ,
       22.16564973, 25.19476081, 25.31372473, 18.51345025, 16.6223286 ,
       17.50268505, 30.94992991, 20.19201752, 23.90440431, 24.86975466,
       13.93767876, 31.82504715, 42.56978796, 17.62323805, 27.01963242,
       17.19006621, 13.80594006, 26.10356557, 20.31516118, 30.08649576,
       21.3124053 , 34.15739602, 15.60444981, 26.11247588, 39.31613646,
       22.99282065, 18.95764781, 33.05555669, 24.85114223, 12.91729352,
       22.68101452, 30.80336295, 31.63522027, 16.29833689, 21.07379993,
       16.57699669, 20.36362023, 26.15615896, 31.06833034, 11.98679953,
       20.42550472, 27.55676301, 10.94316981, 16.82660609, 23.92909733,
        5.28065815, 21.43504661, 41.33684993, 18.22211675,  9.48269245,
       21.19857446, 12.95001331, 21.64822797,  9.3845568 , 23.06060014,
       31.95762512, 19.16662892, 25.59942257, 29.35043558, 20.13138581,
       25.57297369,  5.42970803, 20.23169356, 15.1949595 , 14.03241742,
       20.91078077, 24.82249135, -0.47712079, 13.70520524, 15.69525576,
       22.06972676, 24.64152943, 10.7382866 , 19.68622564, 23.63678009,
       12.07974981, 18.47894211, 25.52713393, 20.93461307, 24.6955941 ,
        7.59054562, 19.01046053, 21.9444339 , 27.22319977, 32.18608828,
       15.27826455, 34.39190421, 12.96314168, 21.01681316, 28.57880911,
       15.86300844, 24.85124135,  3.37937111, 23.90465773, 25.81792146,
...
```

```
import seaborn as sns
sns.displot(reg_pred-y_test,kind='kde')
```

[50]

```
<seaborn.axisgrid.FacetGrid at 0x1969f256130>
```



[51]
```
from sklearn.metrics import r2_score
```

[56]
```
score=r2_score(reg_pred,y_test)
```

[57]
```
score
```

```
0.6693702691495591
```

Submitted By:
Name- Shankar Singh Mahanty
Reg.No.-2101020758
Group-3
Sem-5th
Branch- CSE