# SmartInternz Externships

## Applied Data Science

Assignment- 3

Name : SAKTHIVEL V

Reg no : 20BCE7137

Campus: VIT-AP

In [1]:

**import** numpy **as** np **import** pandas **as** pd **import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

# 2) Load the dataset

In [2]:

df **=** pd**.**read\_csv("housing.csv")

In [3]:

df**.**head()

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[3]: | **price** | **area** | **bedrooms** | **bathrooms** | **stories** | **mainroad** | **guestroom** | **basement** | **hotwaterheating** | **airconditio** |
|  | **0** 13300000 | 7420 | 4 | 2 | 3 | yes | no | no | no |  |
|  | **1** 12250000 | 8960 | 4 | 4 | 4 | yes | no | no | no |  |
|  | **2** 12250000 | 9960 | 3 | 2 | 2 | yes | no | yes | no |  |
|  | **3** 12215000 | 7500 | 4 | 2 | 2 | yes | no | yes | no |  |
|  | **4** 11410000 | 7420 | 4 | 1 | 2 | yes | yes | yes | no |  |

In [4]:

<class 'pandas.core.frame.DataFrame'> RangeIndex: 545 entries, 0 to 544 Data columns (total 12 columns):

df**.**info()

# Column Non-Null Count Dtype

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 |  | price | 545 | non-null |  | int64 |
| 1 |  | area | 545 | non-null |  | int64 |
| 2 |  | bedrooms | 545 | non-null |  | int64 |
| 3 |  | bathrooms | 545 | non-null |  | int64 |
| 4 |  | stories | 545 | non-null |  | int64 |
| 5 |  | mainroad | 545 | non-null |  | object |
| 6 |  | guestroom | 545 | non-null |  | object |
| 7 |  | basement | 545 | non-null |  | object |
| 8 |  | hotwaterheating | 545 | non-null |  | object |
| 9 |  | airconditioning | 545 | non-null |  | object |
| 10 |  | parking | 545 | non-null |  | int64 |
| 11 |  | furnishingstatus | 545 | non-null |  | object |

dtypes: int64(6), object(6)

# Perform Below Visualizations.

## Univariate Analysis

* Bi - Variate Analysis

## Multi - Variate Analysis

### univariate analysis

In [5]:

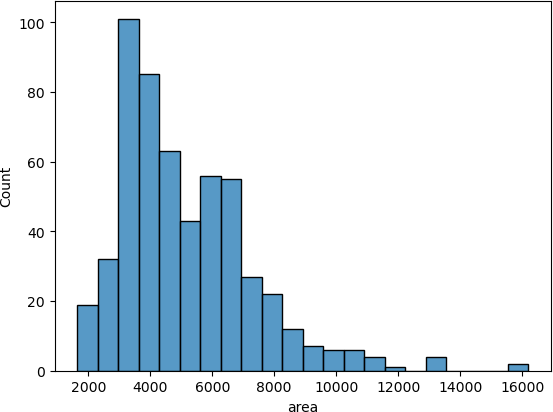
Out[5]:

In [6]:

Out[6]:

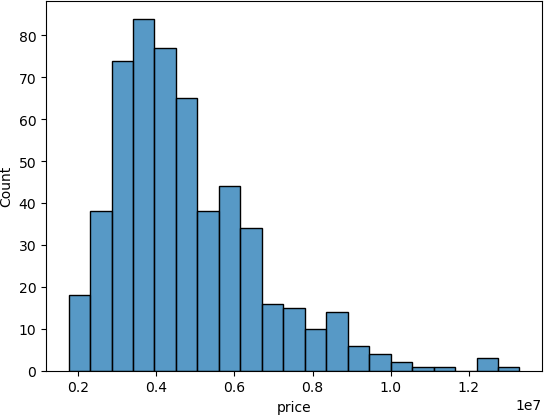
<Axes: xlabel='area', ylabel='Count'>

sns**.**histplot(df['area'])



sns**.**histplot(df['price'])

<Axes: xlabel='price', ylabel='Count'>



In [7]:

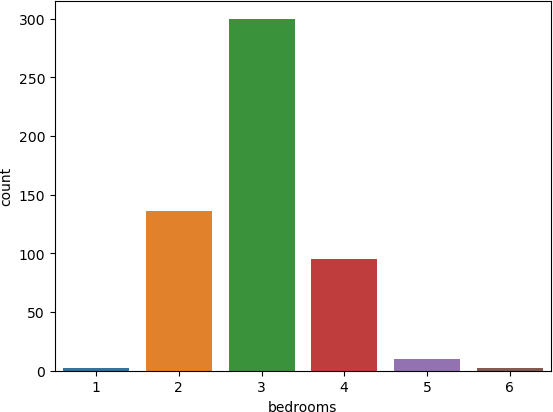
Out[7]:

In [8]:

Out[8]:

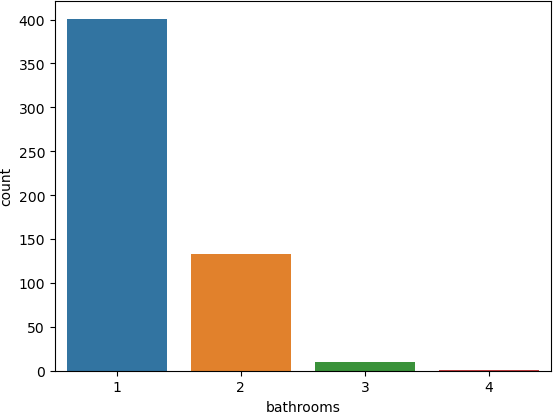
sns**.**countplot(x **=** df['bedrooms'])

<Axes: xlabel='bedrooms', ylabel='count'>



sns**.**countplot(x **=** df['bathrooms'])

<Axes: xlabel='bathrooms', ylabel='count'>



In [9]:

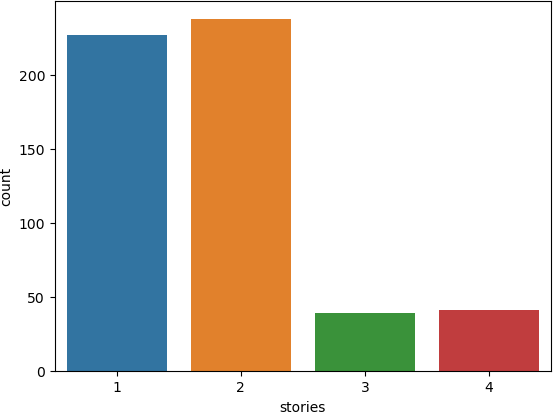
Out[9]:

In [10]:

Out[10]:

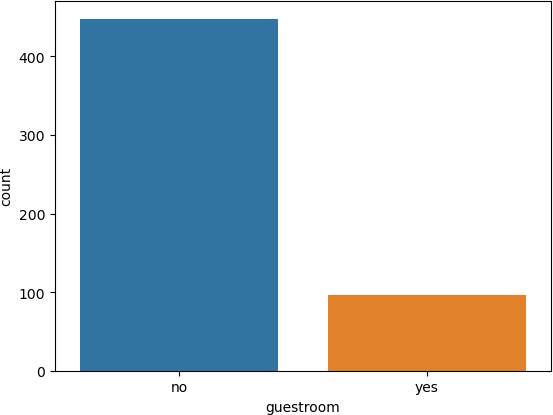
<Axes: xlabel='stories', ylabel='count'>

sns**.**countplot(x **=** df['stories'])



sns**.**countplot(x **=** df['guestroom'])

<Axes: xlabel='guestroom', ylabel='count'>



In [11]:

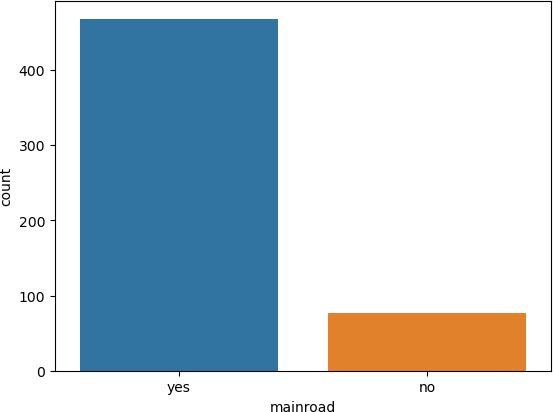
Out[11]:

In [12]:

Out[12]:

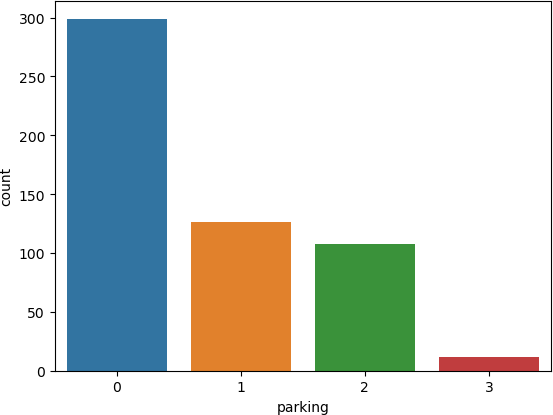
sns**.**countplot(x **=** df['mainroad'])

<Axes: xlabel='mainroad', ylabel='count'>



sns**.**countplot(x **=** df['parking'])

<Axes: xlabel='parking', ylabel='count'>



In [13]:

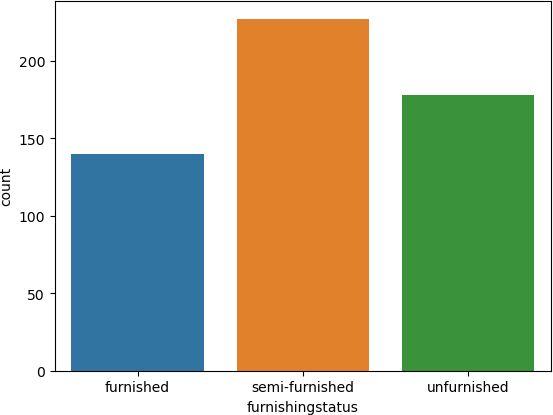
Out[13]:

In [14]:

Out[14]:

sns**.**countplot(x **=** df['furnishingstatus'])

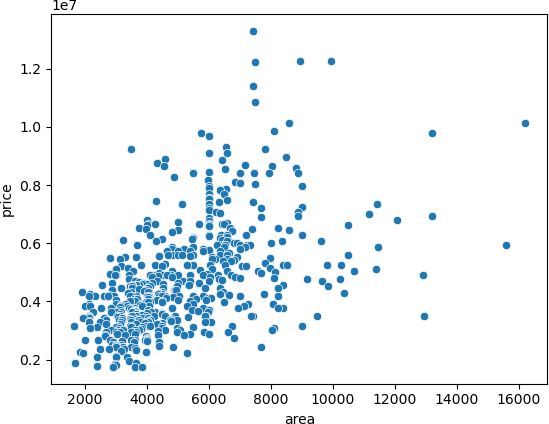
<Axes: xlabel='furnishingstatus', ylabel='count'>



### bivariate analysis

sns**.**scatterplot(data **=** df, x **=** 'area', y **=** 'price')

<Axes: xlabel='area', ylabel='price'>



In [15]:

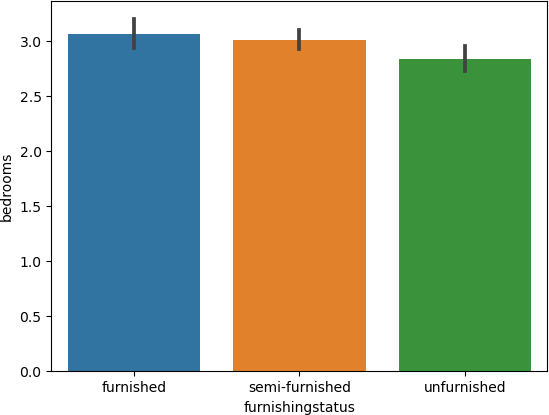
Out[15]:

In [16]:

Out[16]:

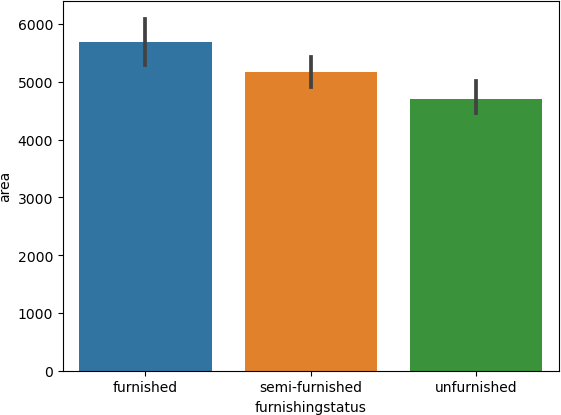
<Axes: xlabel='furnishingstatus', ylabel='bedrooms'>

sns**.**barplot(data **=** df, x **=** 'furnishingstatus', y **=** 'bedrooms')



sns**.**barplot(data **=** df, x **=** 'furnishingstatus', y **=** 'area')

<Axes: xlabel='furnishingstatus', ylabel='area'>



In [17]:

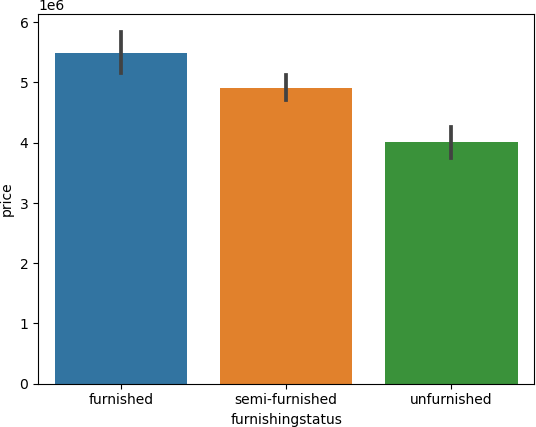
Out[17]:

In [18]:

Out[18]:

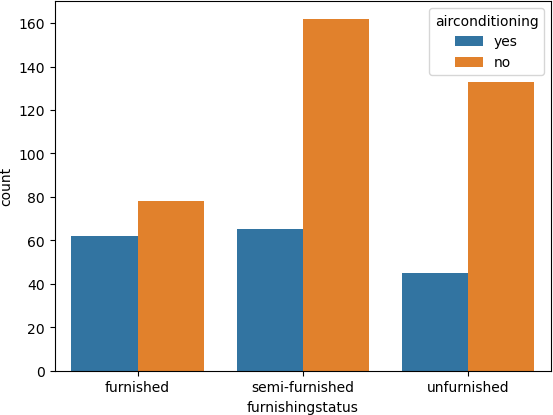
sns**.**barplot(data **=** df, x **=** 'furnishingstatus', y **=** 'price')

<Axes: xlabel='furnishingstatus', ylabel='price'>



sns**.**countplot(x **=** df['furnishingstatus'], hue **=** df['airconditioning'])

<Axes: xlabel='furnishingstatus', ylabel='count'>



In [19]:

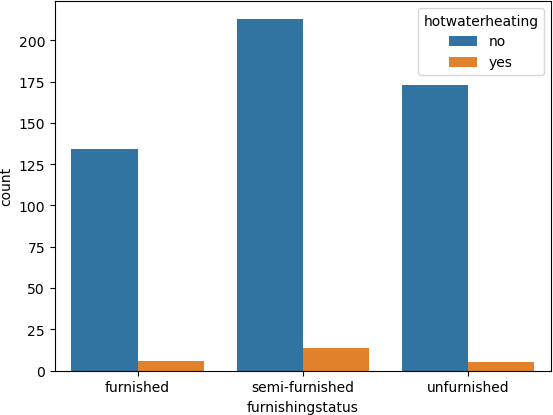
Out[19]:

In [20]:

Out[20]:

sns**.**countplot(x **=** df['furnishingstatus'], hue **=** df['hotwaterheating'])

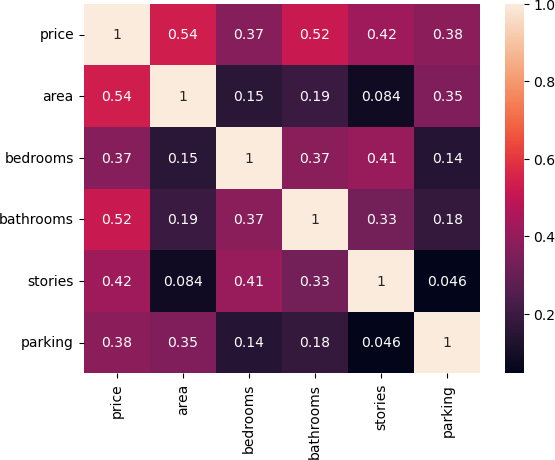
<Axes: xlabel='furnishingstatus', ylabel='count'>



### multivariate analysis

sns**.**heatmap(df**.**corr(numeric\_only**=True**), annot **= True**)

<Axes: >



# Perform descriptive statistics on the dataset.

In [21]:

df**.**describe()

Out[21]:

In [22]:

Out[22]:

**price area bedrooms bathrooms stories parking count** 5.450000e+02 545.000000 545.000000 545.000000 545.000000 545.000000

**mean** 4.766729e+06 5150.541284 2.965138 1.286239 1.805505 0.693578

**std** 1.870440e+06 2170.141023 0.738064 0.502470 0.867492 0.861586

**min** 1.750000e+06 1650.000000 1.000000 1.000000 1.000000 0.000000

**25%** 3.430000e+06 3600.000000 2.000000 1.000000 1.000000 0.000000

**50%** 4.340000e+06 4600.000000 3.000000 1.000000 2.000000 0.000000

**75%** 5.740000e+06 6360.000000 3.000000 2.000000 2.000000 1.000000

**max** 1.330000e+07 16200.000000 6.000000 4.000000 4.000000 3.000000

# Handle the Missing values.

df**.**isnull()**.**sum()

price 0

area 0

bedrooms 0

bathrooms 0

stories 0

mainroad 0

guestroom 0

basement 0

hotwaterheating 0

airconditioning 0

parking 0

furnishingstatus 0

dtype: int64

# Find the outliers and replace the outliers

In [23]:

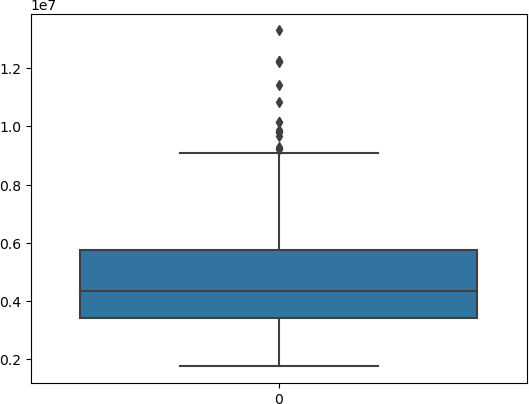
Out[23]:

In [24]:

Out[24]:

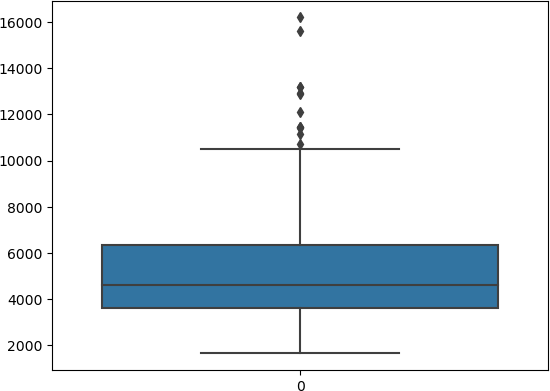
<Axes: >

sns**.**boxplot(df['price'])



sns**.**boxplot(df['area'])

<Axes: >



In [25]:

median\_age **=** df['price']**.**median()

df["price"] **=** np**.**where(df["price"] **>**12000000, median\_age, df['price']) sns**.**boxplot(df['price'])

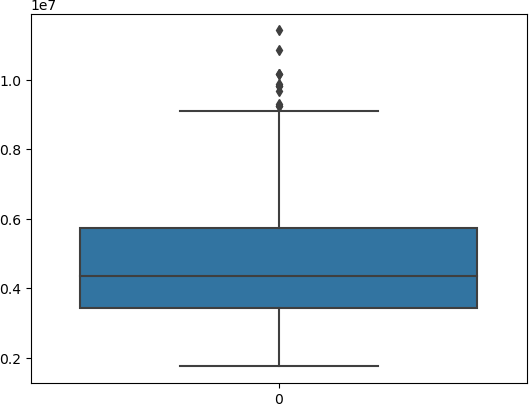
Out[25]:

In [26]:

median\_area **=** df['area']**.**median()

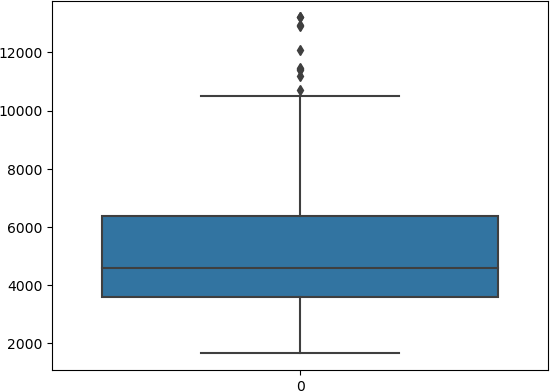
df["area"] **=** np**.**where(df["area"] **>** 14000, median\_area, df['area']) sns**.**boxplot(df['area'])

<Axes: >



Out[26]:

<Axes: >



# Check for Categorical columns and perform encoding.

In [27]:

**from** sklearn.preprocessing **import** OneHotEncoder

In [28]:

encoding **=** pd**.**get\_dummies(df, columns **=** ['mainroad', 'guestroom', 'basement','hotwaterhe

'airconditioning', 'furnishingstatus'])

In [29]:

encoding**.**head()

Out[29]:

In [30]:

Out[30]:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **price** | **area** | **bedrooms** | **bathrooms** | **stories** | **parking** | **mainroad\_no** | **mainroad\_yes** | **guestroom\_no** | **gues** |
| **0** 4340000.0 | 7420.0 | 4 | 2 | 3 | 2 | 0 | 1 | 1 |  |
| **1** 4340000.0 | 8960.0 | 4 | 4 | 4 | 3 | 0 | 1 | 1 |  |
| **2** 4340000.0 | 9960.0 | 3 | 2 | 2 | 2 | 0 | 1 | 1 |  |
| **3** 4340000.0 | 7500.0 | 4 | 2 | 2 | 3 | 0 | 1 | 1 |  |
| **4** 11410000.0 | 7420.0 | 4 | 1 | 2 | 2 | 0 | 1 | 0 |  |

# Split the data into dependent and independent variables

df**.**columns

Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'parking', 'furnishingstatus'],

dtype='object')

In [65]:

*# independent variables*

X **=** encoding**.**drop(['price'], axis **=** 1) X**.**head()

Out[65]:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **area** | **bedrooms** | **bathrooms** | **stories** | **parking** | **mainroad\_no** | **mainroad\_yes** | **guestroom\_no** | **guestroom\_yes** | **b** |
| **0** 7420.0 | 4 | 2 | 3 | 2 | 0 | 1 | 1 | 0 |  |
| **1** 8960.0 | 4 | 4 | 4 | 3 | 0 | 1 | 1 | 0 |  |
| **2** 9960.0 | 3 | 2 | 2 | 2 | 0 | 1 | 1 | 0 |  |
| **3** 7500.0 | 4 | 2 | 2 | 3 | 0 | 1 | 1 | 0 |  |
| **4** 7420.0 | 4 | 1 | 2 | 2 | 0 | 1 | 0 | 1 |  |

In [66]:

*# dependent variables* y **=** df[['price']] y**.**head()

Out[66]:

In [67]:

**from** sklearn.preprocessing **import** StandardScaler scaler **=** StandardScaler()

x\_std **=** scaler**.**fit\_transform(X)

**price**

**0** 4340000.0

**1** 4340000.0

**2** 4340000.0

**3** 4340000.0

**4** 11410000.0

# Scaling the independent variables

In [68]:

x\_std

Out[68]:

In [69]:

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.33, random\_state**=**0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| array([[ 1.11756482, | 1.40341936, | 1.42181174, | ..., | 1.70084013, |
| -0.84488844, | -0.6964292 ], |  |  |  |
| [ 1.8623093 , | 1.40341936, | 5.40580863, | ..., | 1.70084013, |
| -0.84488844, | -0.6964292 ], |  |  |  |
| [ 2.34590961, | 0.04727831, | 1.42181174, | ..., | -0.58794474, |
| 1.18358821, | -0.6964292 ], |  |  |  |
| ..., |  |  |  |  |
| [-0.72011635, | -1.30886273, -0.57018671, | | ..., | -0.58794474, |
| -0.84488844, | 1.43589615], | |  |  |
| [-1.06347257, | 0.04727831, -0.57018671, | | ..., | 1.70084013, |
| -0.84488844, | -0.6964292 ], | |  |  |
| [-0.60888828, | 0.04727831, -0.57018671, | | ..., | -0.58794474, |
| -0.84488844, | 1.43589615]]) | |  |  |

# Split the data into training and testing

1. Build the Model

In [70]:

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** mean\_squared\_error, r2\_score

In [71]:

lr **=** LinearRegression()

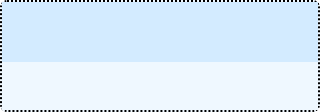
In [ ]:

# Train the Model

In [72]:

lr**.**fit( X\_train, y\_train )

Out[72]:



▾ LinearRegression

LinearRegression()

In [73]:

print("Value of the coefficients: \n", lr**.**coef\_) print(" ")

print("Value of the intercept: \n", lr**.**intercept\_)

Value of the coefficients:

[[ 2.60781675e+02 9.30932038e+04 8.20605321e+05 3.96961106e+05 1.00796216e+05 -3.10469714e+05 3.10469714e+05 -3.32132424e+05

3.32132424e+05 -1.72635846e+05 1.72635846e+05 -7.33905647e+05

7.33905647e+05 -4.83024979e+05 4.83024979e+05 1.04827468e+05

1.38634062e+05 -2.43461530e+05]]

Value of the intercept: [2156139.9017023]

# Test the Model

In [74]:

Y\_pred **=** lr**.**predict(X\_test)

# Measure the performance using Metrics.

In [81]:

**from** sklearn.metrics **import** mean\_squared\_error

**from** sklearn.metrics **import** r2\_score

In [82]:

print("MSE",mean\_squared\_error(y\_test,Y\_pred)) print(" ")

print("RMSE",np**.**sqrt(mean\_squared\_error(y\_test,Y\_pred))) print(" ")

print("R-Square", r2\_score(y\_test,Y\_pred))

MSE 1326521791171.129

RMSE 1151747.2774750236

R-Square 0.5718914765881087

In [ ]: