REGRESSION ANALYSIS (ASSESSMENT - 1)

CSE4020(MACHINE LEARNING)LAB:L49-L50

JANUARY 22, 2022

ANIRUDH VADERA
20BCE2940

QUESTION:

1. Linear Regression:

Read the students' height and weight from the csv file. Understand the relationship between the features using Linear Regression. Predict the value of weight for the given height of a student. Find the performance of the model using appropriate metrics. Visualize the result in scatter plot. Divide the dataset into training and testing. Predict the value for the test set.

Expected Output

----

1. **Initial scatter plot**
2. **Scatter plot along with regression line**
3. **List the x value and the predicted y value**
4. **Display the error**

2. Multi Linear Regression

Multi Linear Regression Predict the value of house price given the features about the house. Use Multi Linear Regression to predict the value. Divide the data set into train and test with 65% and 35% respectively. Visualize the result in scatter plot and display the error

Expected Output

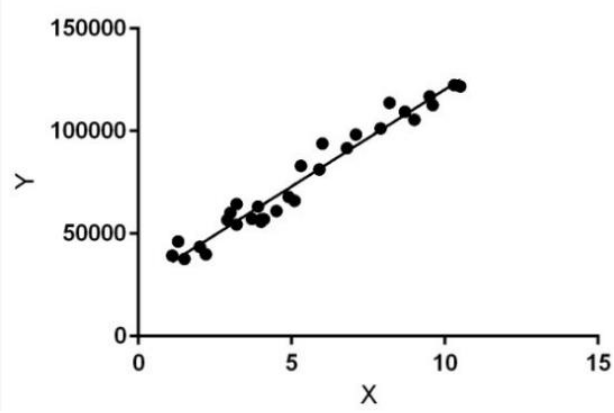
----

1) **Scatter plot with regression line**
2) **List the x value and the predicted y value**
3) **Display the error**

➔ Linear Regression

Description:

Linear regression is an approach to model the relationship between a single **dependent variable** (target variable) and one (simple regression) or more (multiple regression) **independent variables**. The linear regression model assumes a linear relationship between the input and output variables. If this relationship is present, we can estimate the coefficients required by the model to make predictions on new data.



Formula Used:

Hypothesis function for Linear Regression:

Y(Pred) = b0 + b1*X

b0 = Intercept

b1 = Coefficient

$$\text{Error} = \sum_{i=1}^{n}(actual\_output - predicted\_output) ** 2$$

Figure 2: Error Calculation

$$b_0 = \bar{y} - b_1\bar{x}$$

Figure 3: Intercept Calculation

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Figure 4: Co-efficient Formula

Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_absolute_error

from sklearn.metrics import mean_squared_error

from sklearn import metrics


# Importing the dataset

df=pd.read_csv("C:/Users/Anirudh/OneDrive/Desktop/weight-height.csv")

print("The dataset is as following : [10000 rows x 3 columns]")

print(df)

print("\n")


# Check for missing values

print("Checking for missing values :")

print(df.isnull().sum())

print("\n")


# Printing the header of the dataset

print("Dataset Header : ")

print(df.head())

print("\n")


# Information regarding the columns

print("Information regarding the columns : ")

print("Height : inches")

print("Weight : pounds")

print(df.info())

print("\n")


# Information related to the dataset

print("Dataset Details : ")

print(df.describe())
```

```
print("\n")


# Plot Height vs Weight...........

# Blue points - Male

# Magenta points - Female

ax1 = df[df['Gender'] == 'Male'].plot(kind='scatter', x='Height', y='Weight', color='blue', alpha=0.5,
figsize=(10, 7))

df[df['Gender'] == 'Female'].plot(kind='scatter', x='Height', y='Weight', color='magenta', alpha=0.5,
figsize=(10 ,7), ax=ax1)

plt.legend(labels=['Males', 'Females'])

plt.title('Relationship between Height and Weight', size=24)

plt.xlabel('Height (inches)', size=18)

plt.ylabel('Weight (pounds)', size=18)


# Fitting a Regression Model

# 20% of dataset - Testing

# 80% of dataset - Training

# Random state = 123

X = df[["Height"]]

Y = df[["Weight"]]

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=123)

lin_reg = LinearRegression()

lin_reg.fit(X_train, Y_train)


# Predicting the test set values

lin_pred = lin_reg.predict(X_test)


print("The X Values are : ")

print(X_test)

print("\n")


print("The predicted Y_Values are : ")
```

```
print(lin_pred)
print("\n")

# Visualizing the result in form of a scatterplot (Training Set)
plt.scatter(X_train, Y_train, color = 'Magenta')
plt.plot(X_train, lin_reg.predict(X_train), color = 'blue')
plt.legend(labels=['Height vs Weight plot(Training Set)', 'Regresion Line'])
plt.title('Relationship between Height and Weight using Regression Line(Training Set)', size=24)
plt.xlabel('Height (inches)', size=18)
plt.ylabel('Weight (pounds)', size=18)

# Visualizing the result in form of a scatterplot (Testing Set)
plt.scatter(X_test, Y_test, color = 'Magenta')
plt.plot(X_train, lin_reg.predict(X_train), color = 'blue')
plt.legend(labels=['Height vs Weight plot(Test Set)', 'Regresion Line'])
plt.title('Relationship between Height and Weight using Regression Line(Test Set)', size=24)
plt.xlabel('Height (inches)', size=18)
plt.ylabel('Weight (pounds)', size=18)

# Checking the accuracy of our model
print("Accuracy of the model : ")
print("\n")
print('Mean Absolute Error:', mean_absolute_error(Y_test, lin_pred))
print('Mean Squared Error:', mean_squared_error(Y_test, lin_pred))
print('Mean Root Squared Error:', np.sqrt(mean_squared_error(Y_test, lin_pred)))
print("\n")
print('Variance score: %.2f' % lin_reg.score(X_test, Y_test))
print('Coefficients: ', lin_reg.coef_)
print('R square = ',metrics.r2_score(Y_test, lin_pred)*100)

# Making new predictions based on given height
```

given_height = int(input("Enter the height of the student whose weight you want to predict (in inches) : "));

print("\n")

predict_weight = lin_reg.predict([[given_height]])

print("The predicted weight is (in pounds) : ", predict_weight)

 # The Linear Regression Equation

print("The Coefficient is : " , lin_reg.coef_)

print("The Intercept is : " , lin_reg.intercept_)

print("The Linear regression line is : Y = " , lin_reg.intercept_[0] , "+" , lin_reg.coef_[0][0] , "*X")

 # Residual Ananlysis

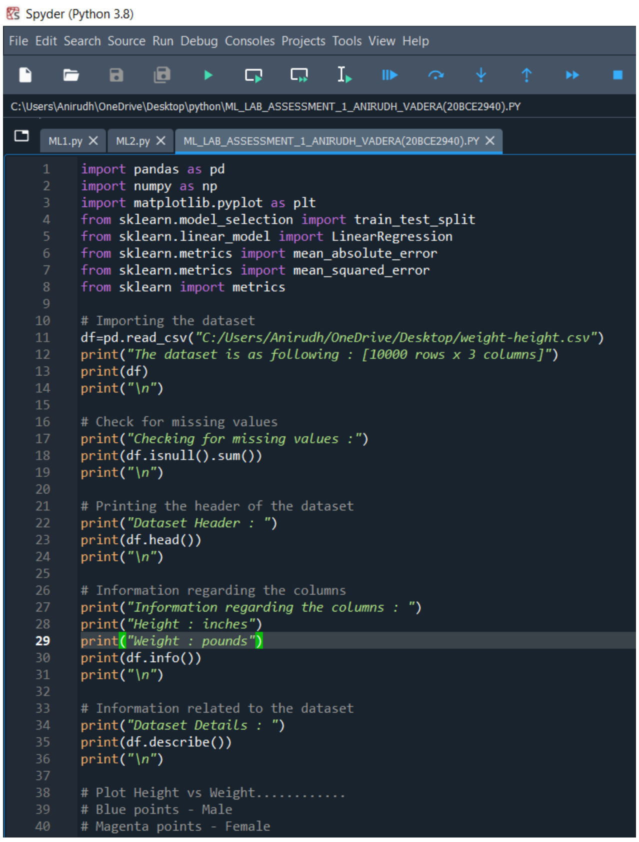
plt.scatter(Y_test,lin_pred,color="Magenta")

plt.title('Checking difference between the predicted Y_Values and the original Y_Values', size=24)

plt.xlabel('Y_test set', size=18)

plt.ylabel('Y Predicted', size=18)

Code Snippets:



Output and Results:

Dataset Details:

Dataset:

```
Console 1/A

Python 3.8.12 (default, Oct 12 2021, 03:01:40) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Anirudh/OneDrive/Desktop/python/ML_LAB_ASSESSMENT_1_ANIRUDH_VADERA(20BCE2940).PY',
OneDrive/Desktop/python')
The dataset is as following : [10000 rows x 3 columns]
      Gender     Height      Weight
0       Male  73.847017  241.893563
1       Male  68.781904  162.310473
2       Male  74.110105  212.740856
3       Male  71.730978  220.042470
4       Male  69.881796  206.349801
...      ...        ...         ...
9995  Female  66.172652  136.777454
9996  Female  67.067155  170.867906
9997  Female  63.867992  128.475319
9998  Female  69.034243  163.852461
9999  Female  61.944246  113.649103

[10000 rows x 3 columns]


Checking for missing values :
Gender    0
Height    0
Weight    0
dtype: int64
```

As the missing values is none we can proceed further:

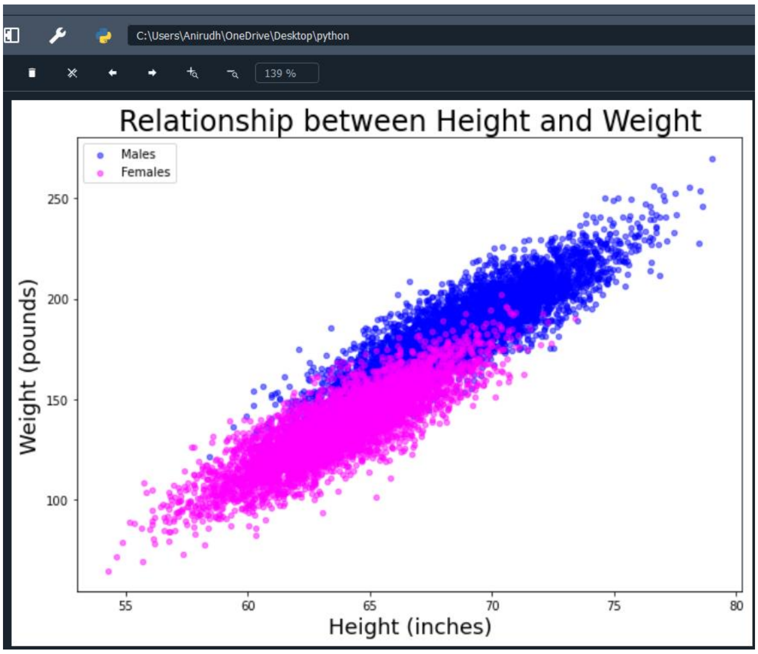
Dataset Details:

```
Dataset Header :
  Gender     Height      Weight
0   Male  73.847017  241.893563
1   Male  68.781904  162.310473
2   Male  74.110105  212.740856
3   Male  71.730978  220.042470
4   Male  69.881796  206.349801


Information regarding the columns :
Height : inches
Weight : pounds
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Gender  10000 non-null  object
 1   Height  10000 non-null  float64
 2   Weight  10000 non-null  float64
dtypes: float64(2), object(1)
memory usage: 234.5+ KB
None
```

```
Dataset Details :
             Height        Weight
count  10000.000000  10000.000000
mean      66.367560    161.440357
std        3.847528     32.108439
min       54.263133     64.700127
25%       63.505620    135.818051
50%       66.318070    161.212928
75%       69.174262    187.169525
max       78.998742    269.989699
```

Initial Scatterplot:

ScatterPlot between weight and height to understand the relationship between the two:



The X_Values and Predicted Y_Values:

Fitting the Regression Model:

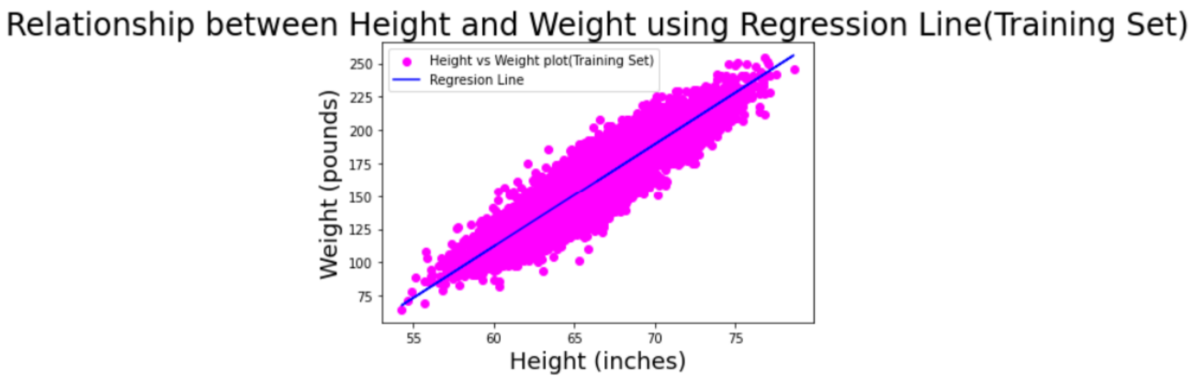
80% data for training and 20% for testing:

The X-Values for which Y-Values to be predicted are:

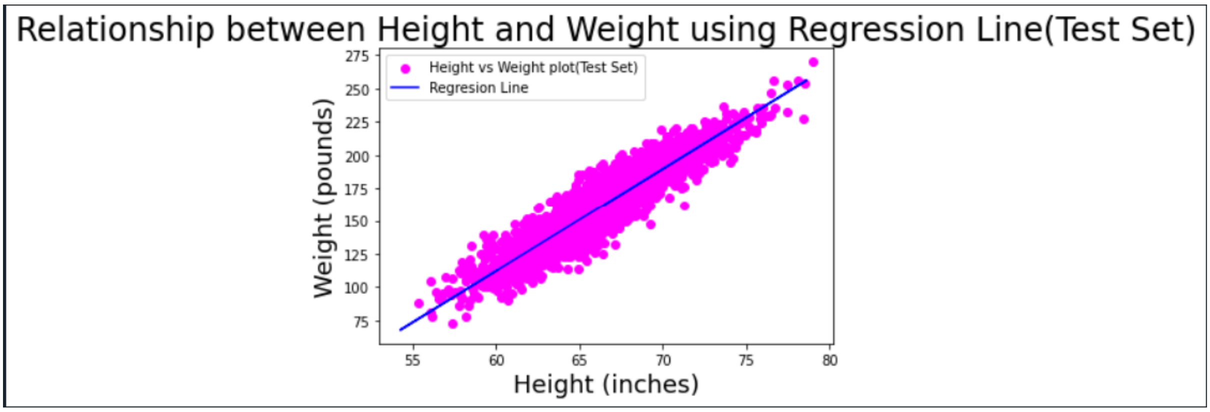
```
The X Values are :
          Height
2656  73.181767
445   71.433376
9505  60.026750
332   66.556587
4168  71.035509
...         ...
8018  63.992565
6463  64.388396
2883  65.625006
7895  61.530110
620   65.892753

[2000 rows x 1 columns]


The predicted Y_Values are :
[[214.02231091]
 [200.50905719]
 [112.34767127]
 ...
 [155.61638129]
 [123.96708917]
 [157.68578703]]
```

Regression Results:

ScatterPlot for training set:



ScatterPlot for testing set:



Accuracy Analysis(Errors):

Checking the Accuracy of the model:

```
Accuracy of the model :


Mean Absolute Error: 9.688834054963015
Mean Squared Error: 143.2255601011165
Mean Root Squared Error: 11.9676881686112


Variance score: 0.86
Coefficients:  [[7.72896259]]
R square =  86.49031737206691
```

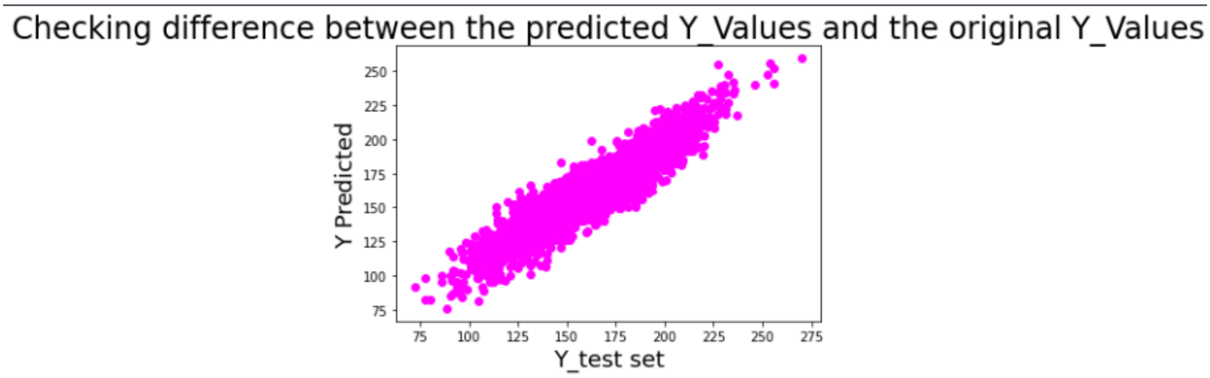
Predicting the weight based on given height:

```
Enter the height of the student whose weight you want to predict (in inches) : 170


The predicted weight is (in pounds) :  [[962.32680767]]
```

The Regression Equation:

```
The Coefficient is :  [[7.72896259]]
The Intercept is :  [-351.59683192]
The Linear regression line is : Y =  -351.59683191643967 + 7.728962585782046 *X
```

The Difference between the predicted and original Y-values:



We get a straight line with positive slope

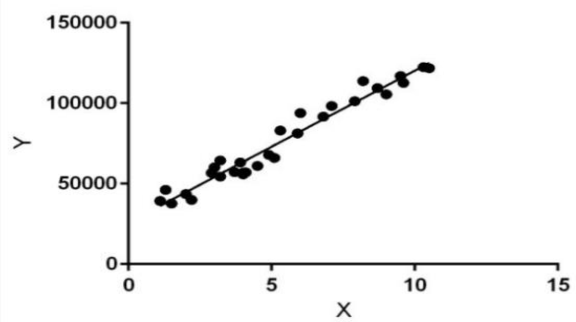
Inference:

1. **Positive Linear Relationship**
   The initial ScatterPlot shows that on increasing the height the weight also increases
2. **We can see from the output below that R square is 0.864 that is 86.4% which is good and mean squared error is a bit high at 143 but manageable.**
3. **The variance in predicted y_values and the actual y_values is not so much which can be seen through the value of mean absolute error which is 9.688**
4. **The Regression Equation is:**
   Y = -351.59683191643967 + 7.728962585782046 *X
   Y = weight, X = height
5. **As we can see in the last scatterplot (the difference between predicted and actual y_values) we get almost a slanting straight line in positive direction (and positive slope) which shows values were well predicted.**

➔Multi Linear Regression

Description:

Linear regression is an approach to model the relationship between a single **dependent variable** (target variable) and one (simple regression) or more (multiple regression) **independent variables**. The linear regression model assumes a linear relationship between the input and output variables. If this relationship is present, we can estimate the coefficients required by the model to make predictions on new data.



Formula Used:

Hypothesis function for Linear Regression:

Y(Pred) = b0 + b1*X1 + b2*X2 …. bn*Xn

b0 = Intercept

b1 = Coefficient of X1

b2 = Coefficient of X2

bn = Coefficient of Xn

$$b_0 = \bar{y} - b_1\bar{x}$$

Figure 3: Intercept Calculation

$$\text{Error} = \sum_{i=1}^{n}(actual\_output - predicted\_output) ** 2$$

Figure 2: Error Calculation

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Figure 4: Co-efficient Formula

Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import mean_absolute_error

from sklearn.metrics import mean_squared_error

from sklearn import metrics

import seaborn as sns


# Importing the dataset

df=pd.read_csv("C:/Users/Anirudh/OneDrive/Desktop/Housing.csv")

print("The dataset is as following : [545 rows x 13 columns]")

print(df)

print("\n")


# Check for missing values

print("Checking for missing values :")

print(df.isnull().sum())

print("\n")


# Printing the header of the dataset

print("Dataset Header : ")

print(df.head())

print("\n")


# Information regarding the columns

print("Information regarding the columns : ")

print("Height : inches")

print("Weight : pounds")

print(df.info())

print("\n")


# Information related to the dataset

print("Dataset Details : ")

print(df.describe())
```

```
print("\n")
```

 # Data Preparation

 # You can see that your dataset has many columns with values as 'Yes' or 'No'.

 # But in order to fit a regression line, we would need numerical values and not string.

 # Hence, we need to convert them to 1s and 0s, where 1 is a 'Yes' and 0 is a 'No'.

 # List of variables to map

```
varlist =  ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']
```

 # Defining the map function

```
def binary_map(x):
    return x.map({'yes': 1, "no": 0})
```

 # Applying the function to the housing list

```
df[varlist] = df[varlist].apply(binary_map)
```

 # Dummy Variables

 # The variable furnishingstatus has three levels. We need to convert these levels into integer as well.

 # For this, we will use something called dummy variables.

 # Get the dummy variables for the feature 'furnishingstatus' and store it in a new variable - 'status'

```
status = pd.get_dummies(df['furnishingstatus'])
```

 # Now, you don't need three columns.

 # You can drop the furnished column, as the type of furnishing can be identified with just the last two columns where —

 # 00 will correspond to furnished

 # 01 will correspond to unfurnished

 # 10 will correspond to semi-furnished

```
# Let's drop the first column from status df using 'drop_first = True'


status = pd.get_dummies(df['furnishingstatus'], drop_first = True)


# Add the results to the original housing dataframe

df = pd.concat([df, status], axis = 1)


# Drop 'furnishingstatus' as we have created the dummies for it

df.drop(['furnishingstatus'], axis = 1, inplace = True)

# Now let's see the head of our dataframe.

print("After Trimming and correcting the dataset looks like follows : ")

print(df.head())


# Fitting a Regression Model

# 35% of dataset - Testing

# 65% of dataset - Training

# Random state = 10

df_train, df_test = train_test_split(df,test_size=0.35,train_size=0.65,random_state=10)



# Let's check the correlation coefficients to see which variables are highly correlated


plt.figure(figsize = (16, 10))

sns.heatmap(df_train.corr(), annot = True, cmap="YlGnBu")

plt.show()


# We can see there are some factors that affects the price of a house to great extent

# These columns are :

# Area

# Bathrooms

# Stories
```

 # Air conditioning

 # This means changing value of even one of these factors will greatly affect the pricing of the house

 # Showing the relationship between the price and the most dominant factor that is area

```
plt.scatter(df["area"], df["price"], color = 'Magenta')

plt.legend(labels=['Area vs Height plot'])

plt.title('Relationship between Price and area of the houses', size=24)

plt.xlabel('Area', size=18)

plt.ylabel('Price', size=18)
```

 # Dividing into X and Y sets for the model building

```
Y_train = df_train.pop('price')

X_train = df_train

# Using the Linear Regression Model

lin_reg = LinearRegression()

lin_reg.fit(X_train, Y_train)



Y_test = df_test.pop("price")

X_test = df_test
```

 # Predicting the test set values

```
lin_pred = lin_reg.predict(X_test)


print("The X Values are : ")

print(X_test)

print("\n")


print("The predicted Y_Values are : ")
```

```
print(lin_pred)
print("\n")

print("The values look something like this : ")
X_test["Price"] = lin_pred
print(X_test)
X_test.drop(["Price"],axis=1,inplace=True)

new_lin_reg = LinearRegression()
new_lin_reg.fit(X_train["area"].values.reshape(-1,1), Y_train)

# Visualizing the result in form of a scatterplot (Training Set)
# We are only visualizing for the area factor as it is the most dominant
plt.scatter(X_train["area"], Y_train, color = 'Magenta')
plt.plot(X_train["area"], new_lin_reg.predict(X_train["area"].values.reshape(-1,1)), color = 'blue')
plt.legend(labels=['Area vs Price plot(Training Set)', 'Regresion Line'])
plt.title('Relationship between Area and Price of Houses using Regression Line(Training Set)',
size=24)
plt.xlabel('Area' , size=18)
plt.ylabel('Price', size=18)

# Visualizing the result in form of a scatterplot (Testing Set)
# We are only visualizing for the area factor as it is the most dominant
plt.scatter(X_test["area"], Y_test, color = 'Magenta')
plt.plot(X_train["area"], new_lin_reg.predict(X_train["area"].values.reshape(-1,1)), color = 'blue')
plt.legend(labels=['Area vs Price plot(Training Set)', 'Regresion Line'])
plt.title('Relationship between Area and Price of Houses using Regression Line(Test Set)', size=24)
plt.xlabel('Area' , size=18)
plt.ylabel('Price', size=18)
```

```
# Checking the accuracy of our model
print("Accuracy of the model : ")
print("\n")
print('Mean Absolute Error:', mean_absolute_error(Y_test, lin_pred))
print('Mean Squared Error:', mean_squared_error(Y_test, lin_pred))
print('Mean Root Squared Error:', np.sqrt(mean_squared_error(Y_test, lin_pred)))
print("\n")
print('Variance score: %.2f' % lin_reg.score(X_test, Y_test))
print('R square = ',metrics.r2_score(Y_test, lin_pred)*100)

# Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((Y_test - lin_pred), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)            # Plot heading
plt.xlabel('Errors', fontsize = 18)                   # X-label

# Making new predictions based on given features of the house
print("Enter the features of the house : ")
area = int(input("Enter the area of the house : "))
bedrooms = int(input("Enter the no of bedrooms in the house : "))
bathrooms = int(input("Enter the no of bathrooms in the house : "))
stories = int(input("Enter the stories in the house : "))
mainroad = int(input("Is there a attached mainroad 'Yes':1  'No':0 "))
guestroom = int(input("Is there a attached guestroom 'Yes':1  'No':0 "))
basement = int(input("Is there a basement 'Yes':1  'No':0 "))
hotwaterheating = int(input("Is there hot water heating 'Yes':1  'No':0 "))
airconditioning = int(input("Is there air conditioning 'Yes':1  'No':0 "))
parking = int(input("Is there parking nearby 'Yes':1  'No':0 "))
prefarea = int(input("Is there prefarea nearby 'Yes':1  'No':0 "))
furnished = int(input("Is the house semi-furnished:10 furnished:00 or unfurnihsed:01 :: "))
if (furnished==0):
```

```
    semi_furnished = 0
    unfurnished = 0
elif(furnished==1):
    semi_furnished = 0
    unfurnished = 1
else:
    semi_furnished = 1
    unfurnished = 0
predict_price = lin_reg.predict([[area,bedrooms,bathrooms,stories,mainroad,guestroom,basement,hotwaterheating,airconditioning,parking,prefarea,semi_furnished,unfurnished]])
print("The predicted price of the house is : ", predict_price)
```

 # The Linear Regression Equation

 # Checking the coefficients

```
print("The Coefficient are : ")
coeff_df = pd.DataFrame(lin_reg.coef_,X_train.columns,columns=["Coefficient"])
print(coeff_df)
print("The Intercept is : " , lin_reg.intercept_)
print("The Linear regression line is : Price = " , lin_reg.intercept_ , "+" , lin_reg.coef_[0] , "*area" , "+" , lin_reg.coef_[1] , "*bedrooms" , "+" , lin_reg.coef_[2] , "*bathrooms" , "+" , lin_reg.coef_[3] , "*stories" , "+" , lin_reg.coef_[4] , "*mainroad" , "+" , lin_reg.coef_[5] , "*guestroom" , "+" , lin_reg.coef_[6] , "*basement" , "+" , lin_reg.coef_[7] , "*hotwaterheating" , "+" , lin_reg.coef_[8] , "*airconditioning" , "+" , lin_reg.coef_[9] , "*parking" , "+" , lin_reg.coef_[10] , "*prefarea")
```

 # Residual Ananlysis

```
plt.scatter(Y_test,lin_pred,color="Magenta")
plt.title('Checking difference between the predicted Y_Values and the original Y_Values', size=24)
plt.xlabel('Y_test set', size=18)
plt.ylabel('Y Predicted', size=18)
```

 # Plot the histogram of the error terms

```
fig = plt.figure()
sns.distplot((Y_test - lin_pred), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)              # Plot heading
plt.xlabel('Errors', fontsize = 18)
```

Code Snippets:

Spyder (Python 3.8)

...\Anirudh\OneDrive\Desktop\python\ML_LAB_ASSESSMENT_1_ANIRUDH_VADERA(20BCE2940)MULTIPLE_REGRESSION.PY

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn import metrics
import seaborn as sns

# Importing the dataset
df=pd.read_csv("C:/Users/Anirudh/OneDrive/Desktop/Housing.csv")
print("The dataset is as following : [545 rows x 13 columns]")
print(df)
print("\n")

# Check for missing values
print("Checking for missing values :")
print(df.isnull().sum())
print("\n")

# Printing the header of the dataset
print("Dataset Header : ")
print(df.head())
print("\n")

# Information regarding the columns
print("Information regarding the columns : ")
print("Height : inches")
print("Weight : pounds")
print(df.info())
print("\n")

# Information related to the dataset
print("Dataset Details : ")
print(df.describe())
print("\n")
```

Spyder (Python 3.8)

...\Anirudh\OneDrive\Desktop\python\ML_LAB_ASSESSMENT_1_ANIRUDH_VADERA(20BCE2940)MULTIPLE_REGRESSION.PY

```python
# Data Preparation
# You can see that your dataset has many columns with values as 'Yes' or 'No'.
# But in order to fit a regression line, we would need numerical values and not string.
# Hence, we need to convert them to 1s and 0s, where 1 is a 'Yes' and 0 is a 'No'.

# List of variables to map
varlist =  ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']

# Defining the map function
def binary_map(x):
    return x.map({'yes': 1, "no": 0})

# Applying the function to the housing list
df[varlist] = df[varlist].apply(binary_map)

# Dummy Variables
# The variable furnishingstatus has three levels. We need to convert these levels into integer as well.
# For this, we will use something called dummy variables.
# Get the dummy variables for the feature 'furnishingstatus' and store it in a new variable - 'status'
status = pd.get_dummies(df['furnishingstatus'])

# Now, you don't need three columns.
# You can drop the furnished column, as the type of furnishing can be identified with just the last two columns where

# 00 will correspond to furnished
# 01 will correspond to unfurnished
# 10 will correspond to semi-furnished

# Let's drop the first column from status df using 'drop_first = True'

status = pd.get_dummies(df['furnishingstatus'], drop_first = True)

# Add the results to the original housing dataframe
df = pd.concat([df, status], axis = 1)

# Drop 'furnishingstatus' as we have created the dummies for it
df.drop(['furnishingstatus'], axis = 1, inplace = True)
# Now let's see the head of our dataframe.
print(df.head())
```

Output and Results:

Dataset Details:

Dataset:

```
In [1]: runfile('C:/Users/Anirudh/OneDrive/Desktop/python/ML_LAB_ASSESSMENT_1_
wdir='C:/Users/Anirudh/OneDrive/Desktop/python')
The dataset is as following : [545 rows x 13 columns]
        price  area  bedrooms  ...  parking  prefarea furnishingstatus
0    13300000  7420         4  ...        2       yes        furnished
1    12250000  8960         4  ...        3        no        furnished
2    12250000  9960         3  ...        2       yes   semi-furnished
3    12215000  7500         4  ...        3       yes        furnished
4    11410000  7420         4  ...        2        no        furnished
..        ...   ...       ...  ...      ...       ...              ...
540   1820000  3000         2  ...        2        no      unfurnished
541   1767150  2400         3  ...        0        no   semi-furnished
542   1750000  3620         2  ...        0        no      unfurnished
543   1750000  2910         3  ...        0        no        furnished
544   1750000  3850         3  ...        0        no      unfurnished

[545 rows x 13 columns]


Checking for missing values :
price               0
area                0
bedrooms            0
bathrooms           0
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
prefarea            0
furnishingstatus    0
dtype: int64
```

As the missing values is none we can proceed further:

Dataset Details:

```
Dataset Header :
      price  area  bedrooms  ...  parking  prefarea furnishingstatus
0  13300000  7420         4  ...        2       yes        furnished
1  12250000  8960         4  ...        3        no        furnished
2  12250000  9960         3  ...        2       yes   semi-furnished
3  12215000  7500         4  ...        3       yes        furnished
4  11410000  7420         4  ...        2        no        furnished

[5 rows x 13 columns]


Information regarding the columns :
Height : inches
Weight : pounds
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   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  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
None
```

```
Dataset Details :
              price          area  ...     stories     parking
count  5.450000e+02    545.000000  ...  545.000000  545.000000
mean   4.766729e+06   5150.541284  ...    1.805505    0.693578
std    1.870440e+06   2170.141023  ...    0.867492    0.861586
min    1.750000e+06   1650.000000  ...    1.000000    0.000000
25%    3.430000e+06   3600.000000  ...    1.000000    0.000000
50%    4.340000e+06   4600.000000  ...    2.000000    0.000000
75%    5.740000e+06   6360.000000  ...    2.000000    1.000000
max    1.330000e+07  16200.000000  ...    4.000000    3.000000
```

 # Data Preparation

 # You can see that your dataset has many columns with values as 'Yes' or 'No'.

 # But in order to fit a regression line, we would need numerical values and not string.

 # Hence, we need to convert them to 1s and 0s, where 1 is a 'Yes' and 0 is a 'No'.

 # Dummy Variables

 # The variable furnishingstatus has three levels. We need to convert these levels into integer as well.

 # For this, we will use something called dummy variables.

 # Get the dummy variables for the feature 'furnishingstatus' and store it in a new variable - 'status'

 # Now, you don't need three columns.

 # You can drop the furnished column, as the type of furnishing can be identified with just the last two columns where —

 # 00 will correspond to furnished

 # 01 will correspond to unfurnished

 # 10 will correspond to semi-furnished

```
After Trimming and correcting the dataset looks like follows :
      price  area  bedrooms  ...  prefarea  semi-furnished  unfurnished
0  13300000  7420         4  ...         1               0            0
1  12250000  8960         4  ...         0               0            0
2  12250000  9960         3  ...         1               1            0
3  12215000  7500         4  ...         1               0            0
4  11410000  7420         4  ...         0               0            0

[5 rows x 14 columns]
```

Initial Scatterplot:

We now draw a heatmap of covariance of each column with one another.

We now get some following observations:

We can see there are some factors that affects the price of a house to great extent

These columns are:

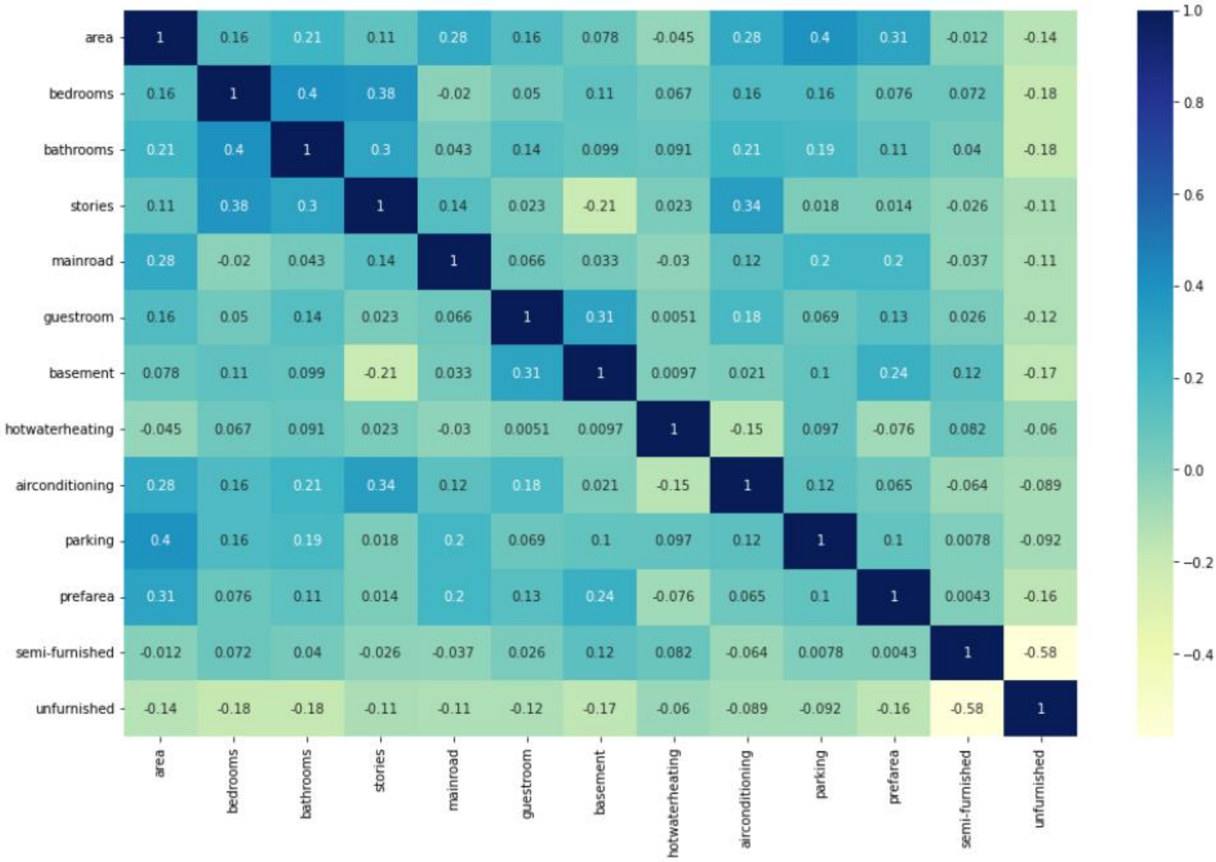
Area

Bathrooms

Stories

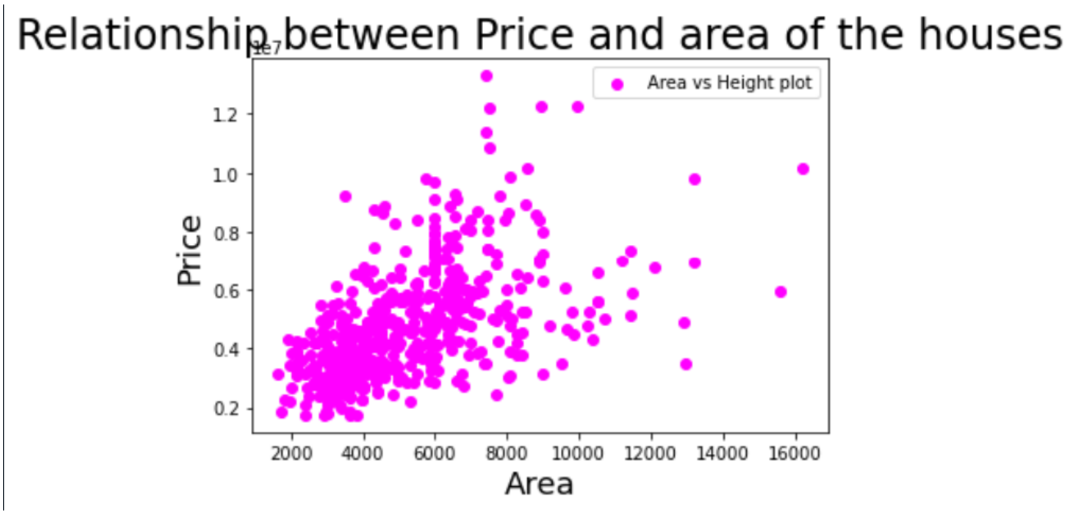
Air conditioning

This means changing value of even one of these factors will greatly affect the pricing of the house



As the area is the most dominant factor in affecting the value of price of the house we will draw scatterplots between area and price of the house

ScatterPlot between Price and Area to understand the relationship between the two:



The X_Values and Predicted Y_Values:

Fitting the Regression Model:

65% data for training and 35% for testing:

The X-Values for which Y-Values to be predicted are:

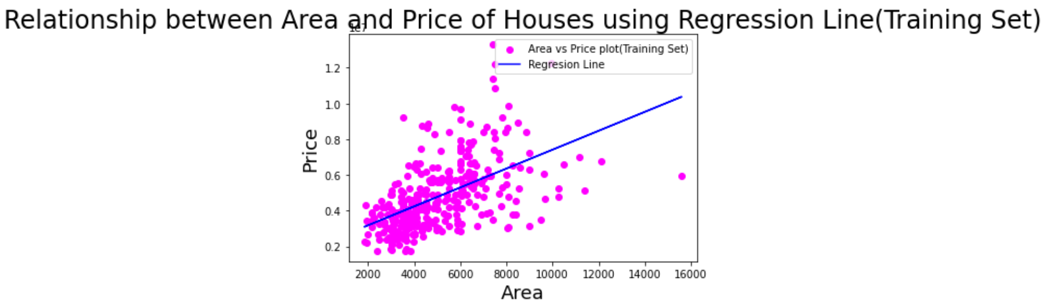
```
The X Values are :
     area  bedrooms  bathrooms  ...  prefarea  semi-furnished  unfurnished
482  3150         3          1  ...         0               0            1
314  4040         2          1  ...         0               1            0
383  4500         4          2  ...         0               0            0
487  5400         4          1  ...         0               0            1
43   6000         4          2  ...         0               1            0
..    ...       ...        ...  ...       ...             ...          ...
12   6550         4          2  ...         1               1            0
97   6400         3          1  ...         1               1            0
130  4800         3          1  ...         0               0            1
33   5960         3          3  ...         0               0            1
465  3800         2          1  ...         0               0            1
```

```
The predicted Y_Values are :
[ 2660628.13495901  3835819.77139557  5820822.60902346  3767930.49734531
  6670701.48650124  6724771.63134602  3545985.06635784  3100031.51306977
  5554503.39233212  3199668.22391934  5047667.16279301  2161341.91419495
  4304172.15652621  4746707.19522931  6730414.60542925  2648241.04780559
  7299096.84535021  6511349.52910361  5182838.03057654  4448705.93984871
  2538223.51269119  1981826.64972116  5923383.65631762  5656007.7591878
  6650998.59277708  6227896.37806669  3819226.88533281  3661500.68911227
  3744646.53737908  3673436.01446116  4320861.53054848  4784148.06135873
  5690613.88966773  5421403.67760399  7422575.60614913  4978800.24562416
  6589798.86011599  2335117.37297983  5139405.48180192  5655270.1751458
  4955655.47660856  5352075.8841257   2853042.3237546   6371473.85858323
  5591239.29939988  7888140.31937221  8179835.08408416  6308371.48751598
  3531006.48625688  5022139.32687507  3422394.49865358  6006108.73703623
  4148960.8385987   4792267.10805539  2815543.64539553  5906169.94684547
  5251100.43535874  3722859.07495565  3843125.0718775   4494856.65418746
  4793195.96418023  4963726.14788397  5303068.43449917  3644597.89959067
  4317494.74840909  6753219.65260247  3047506.81059606  2504775.87533761
  2426800.24508677  7226894.00319624  4146264.70977009  6562335.34613059
  6063259.30206841  3619479.58429875  2780810.44339752  3520494.46129019
```

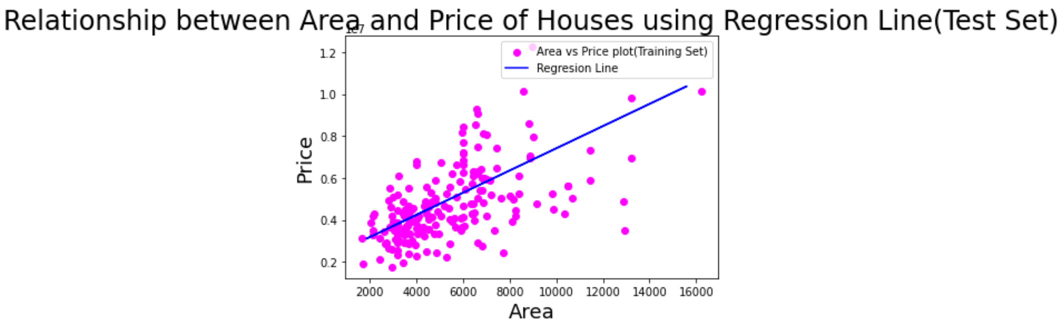
```
The values look something like this :
     area  bedrooms  bathrooms  ...  semi-furnished  unfurnished         Price
482  3150         3          1  ...               0            1  2.660628e+06
314  4040         2          1  ...               1            0  3.835820e+06
383  4500         4          2  ...               0            0  5.820823e+06
487  5400         4          1  ...               0            1  3.767930e+06
43   6000         4          2  ...               1            0  6.670701e+06
..    ...       ...        ...  ...             ...          ...           ...
12   6550         4          2  ...               1            0  7.295587e+06
97   6400         3          1  ...               1            0  6.450643e+06
130  4800         3          1  ...               0            1  3.594404e+06
33   5960         3          3  ...               0            1  6.222597e+06
465  3800         2          1  ...               0            1  2.580317e+06

[191 rows x 14 columns]
```

Regression Results:

ScatterPlot for training set:



ScatterPlot for testing set:



Accuracy Analysis(Errors):

Checking the Accuracy of the model:

```
Accuracy of the model :


Mean Absolute Error: 724444.1496414025
Mean Squared Error: 816243286778.123
Mean Root Squared Error: 903461.834710312


Variance score: 0.74
R square =  73.5848179015316
```

Predicting the price of the house based on given features:

```
Enter the features of the house :

Enter the area of the house : 3000

Enter the no of bedrooms in the house : 2

Enter the no of bathrooms in the house : 3

Enter the stories in the house : 2

Is there a attached mainroad 'Yes':1  'No':0 1

Is there a attached guestroom 'Yes':1  'No':0 1

Is there a basement 'Yes':1  'No':0 1

Is there hot water heating 'Yes':1  'No':0 1

Is there air conditioning 'Yes':1  'No':0 1

Is there parking nearby 'Yes':1  'No':0 1

Is there prefarea nearby 'Yes':1  'No':0 1

Is the house semi-furnished:10 furnished:00 or unfurnihsed:01 :: 00
```

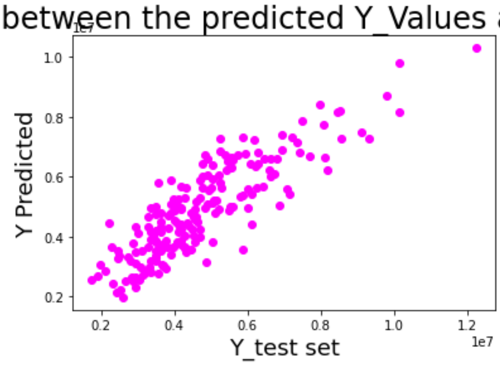
```
The predicted price of the house is :  [8252024.39997437]
```

The Regression Equation:

```
The Coefficient are :
                  Coefficient
area             2.476073e+02
bedrooms         1.720664e+05
bathrooms        7.828399e+05
stories          4.473094e+05
mainroad         3.781196e+05
guestroom        1.627630e+05
basement         4.316503e+05
hotwaterheating  8.178225e+05
airconditioning  1.017280e+06
parking          3.279801e+05
prefarea         6.940205e+05
semi-furnished   1.549957e+04
unfurnished     -4.052882e+05
The Intercept is :  92295.7479408225
The Linear regression line is : Price =  92295.7479408225 + 247.60726134315706 *area + 172066.38956123567 *bedrooms +
782839.8873357589 *bathrooms + 447309.35295425705 *stories + 378119.63480296044 *mainroad + 162762.95492393035 *guestroom +
431650.2903978061 *basement + 817822.5179267152 *hotwaterheating + 1017279.6925102217 *airconditioning + 327980.1105063436
*parking + 694020.5198978384 *prefarea
```

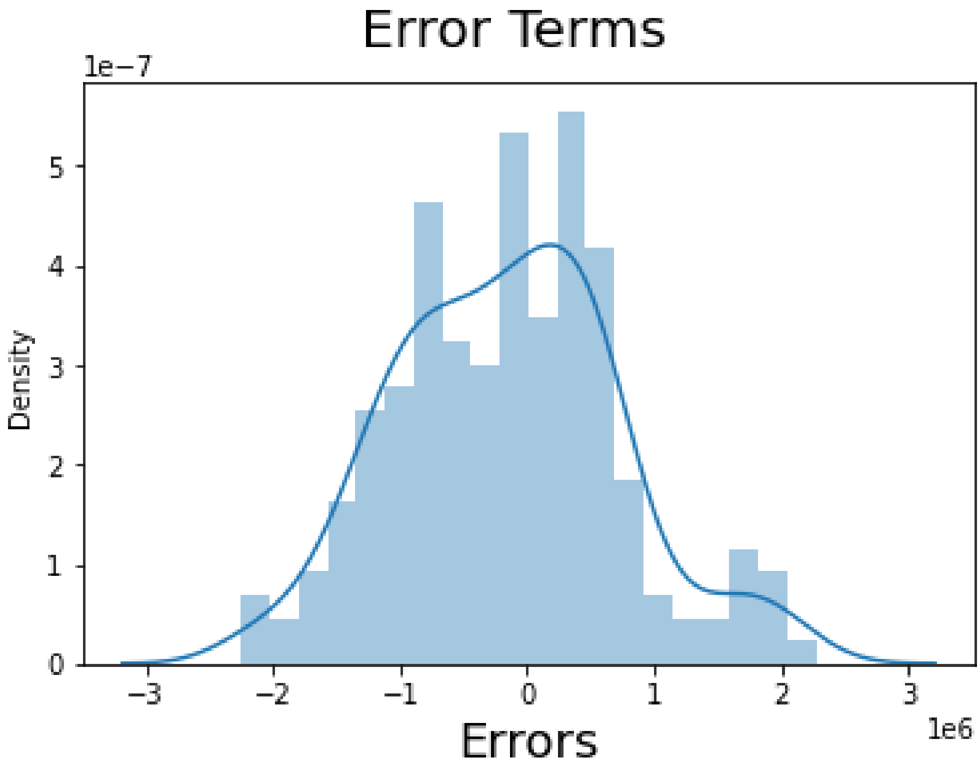
The Difference between the predicted and original Y-values:

Checking difference between the predicted Y_Values and the original Y_Values



We get a straight line with positive slope

Error Terms:



We notice that errors are in acceptable range

Inference:

1. **Positive Linear Relationship**
   The initial ScatterPlot shows that on increasing the area and the other constraints the weight also increases
2. **We can see from the output below that R square is 0.735 that is ~74% which is good.**
3. **The variance score is 0.74 which means the predicted values is much closer to the actual values**
4. **The Regression Equation is:**
   Price = 92295.7479408225 + 247.60726134315706 *area + 172066.38956123567 *bedrooms + 782839.8873357589 *bathrooms + 447309.35295425705 *stories + 378119.63480296044 *mainroad + 162762.95492393035 *guestroom + 431650.2903978061 *basement + 817822.5179267152 *hotwaterheating + 1017279.6925102217 *airconditioning + 327980.1105063436 *parking + 694020.5198978384 *prefarea
5. **As we can see in the last scatterplot (the difference between predicted and actual y_values) we get almost a slanting straight line in positive direction (and positive slope) which shows values were well predicted.**
6. **The error barplot shows that the error is in acceptable range**
7. **The area is the most dominant factor in affecting the value of price of the house**