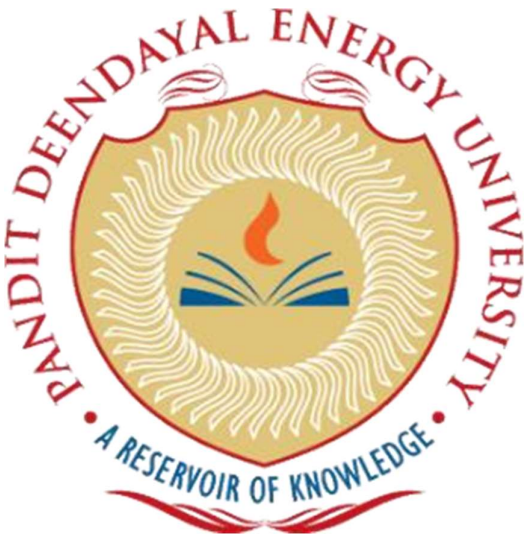
**PANDIT DEENDAYAL ENERGY UNIVERSITY**

**SCHOOL OF TECHNOLOGY**




# Course Name: Industry 4.0 Lab

## Course Code: 20IF201P

## Lab Manual

### B.Tech. (Computer Science and Engineering)

**Name:** Ravi Jamanbhai Makwana

**Roll no:** 21BCP418

**Division:** 6

**Group:** 12

**Semester:** V

**Submitted To:**

Mr. Chandan (VF)

# Index

| Sr. no. | List Of Experiment | Date | Sign |
|---|---|---|---|
| 1 | Implement supervised machine learning algorithm using MATLAB. | 25-08-2023 | |
| 2 | Data Analysis and Data Visualization using Python. | 01-09-2023 | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| 10 | | | |

# EXPERIMENT-1

**Title:** Implement supervised machine learning algorithm using MATLAB.

**Aim:** To implement Linear Regression algorithm and use it to predict the effect of an independent variable on outcome. Particularly, forecast the impact of TV advertising using the available database.

## Introduction:

Linear regression represents a category of machine learning algorithms, specifically falling under supervised machine learning. This method learns patterns from labelled datasets and maps data points to the most optimal linear functions. These functions can subsequently be employed for predictions on new, unprocessed datasets.

In the realm of supervised machine learning, linear regression serves as a foundational algorithm for establishing the linear connection between a dependent variable and one or more independent features. When dealing with a solitary independent feature, it is referred to as Univariate Linear Regression; when there are multiple features involved, it's termed Multivariate Linear Regression. The algorithm's primary objective is to identify the most suitable linear equation for forecasting the value of the dependent variable based on the independent variables. This equation essentially represents a straight line that encapsulates the relationship between the dependent and independent variables. The slope of this line provides insight into how much the dependent variable changes with a unitary shift in the independent variable(s).

Within the framework of supervised learning, regression holds a pivotal role. It entails the acquisition of a function from a dataset containing both X (independent) and Y (dependent) values. This function subsequently aids in predicting Y for a given, unfamiliar X. In the context of regression, the primary focus is on deducing the value of Y, which necessitates the development of a function capable of predicting continuous Y values based on the independent features X.

Here, Y (representing SALES) assumes the role of a dependent or target variable, while X (representing TV ADVERTISING) serves as an independent variable, also known as the predictor of Y. Various types of functions or models can be employed for regression purposes, with a linear function being the most straightforward choice. In this context, X can either encompass a single feature or multiple features, depending on the problem at hand.

**Procedure:**

**Problem Statement**

Imagine you are an owner of a startup; you wish to forecast the sales of your product to plan how much money should be spent on advertisements. This is because the sale of a product is usually proportional to the money spent on advertisements. Our goal is to predict the impact of TV advertising on your product sales by performing simple linear regression analysis using MATLAB.

**Step 1: Analysing the Dataset**

Start by downloading the Advertising-Sales dataset from Kaggle. This dataset includes information about TV, radio, and newspaper ad spending (in thousands of dollars) along with corresponding sales (in thousands of units). The dataset is divided by 1000 for examples.
Dataset link: https://www.kaggle.com/ishaanv/ISLR-Auto

**Step 2: Train-Test Split**

For simple linear regression, focus on the effect of TV ads on sales. TV is the feature variable (independent) and Sales is the target variable (dependent). We need to split the dataset into a training set and test set wherein the training set is used to train a machine learning model and the test set is used to test the accuracy of predictions on the ML model. Divide the dataset into training and test sets. Typically, training has 75% of instances, while the test set has the remainder.

**Step 3: Model Training**

Train the simple regression model on the training data to find the best-fit line y = mx + c. Calculate errors using functions errors_product() and squared_errors(), then derive slope (m) and intercept (c) using formulas. For this perform the following tasks: -

**1.Create following two functions:**
-A function err_prod() that calculates the errors for the feature and target variables i.e. $(xi - \bar{x})(yi - y)$.
Here $\bar{x}$ is the mean of x_train and $\bar{y}$ is the mean of y_train
- A function sq_error() that calculates the squared errors for the feature variable only i.e. $(xi - \bar{x})$

**2.Calculate the slope and intercept values for the best fit line by applying the following formulae:**
$slope \Rightarrow$
$m = \Sigma(xi - \bar{x})(yi - \bar{y}) = errors\_product().\ sum()$
$\Sigma(xi - \bar{x})_2\ squared\_errors().\ sum()$
$intercept \Rightarrow c = \bar{y} - m\bar{x}$

**Step 4: Plotting the Best Fit Line**
In simple linear regression, we consider only one independent variable to predict the value of the dependent variable. In this case, we want to create a simple linear regression model that predicts the Sales (dependent variable) in a city for a certain TV ad (independent variable) recorded. After obtaining the slope and intercept values for the best fit line, plot this line along with the scatter plot to see how well it fits the points.

**Step 5: Model Prediction**
Predict sales for a TV advertising budget of Rs 50,000.

**Step 6: Evaluated the performance of Linear Regression using R_squared value**
Assess the model's accuracy using the R-squared value, indicating the correlation between actual and predicted values in the test set. Higher values (near 1) imply a well-fitted model using Linear Regression.

**Code:**

**Main file:**

```
%%Load dataset
data = readtable("Advertising.csv");

%loading the TV advertisements and Sales of into variable x and y
```

```
%respectively
x = data(:,'TV');
y = data(:,'Sales');

%converting that data into arrays
x = table2array(x);
y = table2array(y);

%making train set and test set for TV advertisements
x_train_set = x(1:150,:);
x_test_set = x(151:200,:);

%making train set and test set for Sales
y_train_set = y(1:150,:);
y_test_set = y(151:200,:);

%calculating number of rows in the train set of TV advertisements
num_rows_x = size(x,1);

%doing a random permutation of number of rows(in a jumbled order)
m = randperm(num_rows_x);

%permuting the values of the array of test set and trainset of TV
%advertisements
x_train_set = x(m(1:150));
x_test_set = x(m(151:end));

%permuting the values of the array of test set and trainset of Sales
y_train_set = y(m(1:150));
y_test_set = y(m(151:end));

%calling the functions to calculate the err_product and square_err
err_prod = errors_product(x_train_set, y_train_set);
sq_err = squared_product(x_train_set);

%calculating the slope of the graph
slope = err_prod/sq_err;
%calculating the intercept
intercept = mean(y_train_set) - slope * mean(x_train_set);

%displaying the value of slope and intercept obtained
disp(['The slope of the line is ', num2str(slope)])
disp(['The intercept of the line is ', num2str(intercept)])

%%Plotting the best fit line
plot(x,y,'bo')
hold on
plot(x, slope*x + intercept, 'k', 'LineWidth', 2)
title('Regression Line')
xlabel('TV Advertisement Expense')
ylabel('Sales')
legend({'Data', 'Best Fit Line'}, 'Location', 'northwest')
hold off
```

```
%%Predict sales for Rs 50,000 spent on TV Ads
spent_value = 50000/1000;
sale = (spent_value*slope + intercept)*1000;
disp (['for tv ad of rs ',num2str(spent_value*1000), ' the predicted sales
is ',num2str(sale)])

%% test the model on test set
y_predicted=(x_test_set*slope+intercept);
plot(x_test_set,y_test_set,'ro');
hold on
plot(x_test_set,y_predicted,'k+');
legend({'Original value','Predicted Value'})
title('Comparision of Original and Predicted sales value')
xlabel('TV Expense')
ylabel('Sales')
hold off

%% check the performance
mean_error=immse(y_test_set,y_predicted);
Rsquared=(corr(y_test_set,y_predicted))^2;
disp(['Root Mean Squared error (RMS): ',num2str(mean_error)])
disp(['r squared is: ',num2str(Rsquared)])
```

## errors_product.m:

```
function prod = error_product(x,y)
prod = (x - mean(x)) .* (y - mean(y));
prod = sum(prod);
end
```

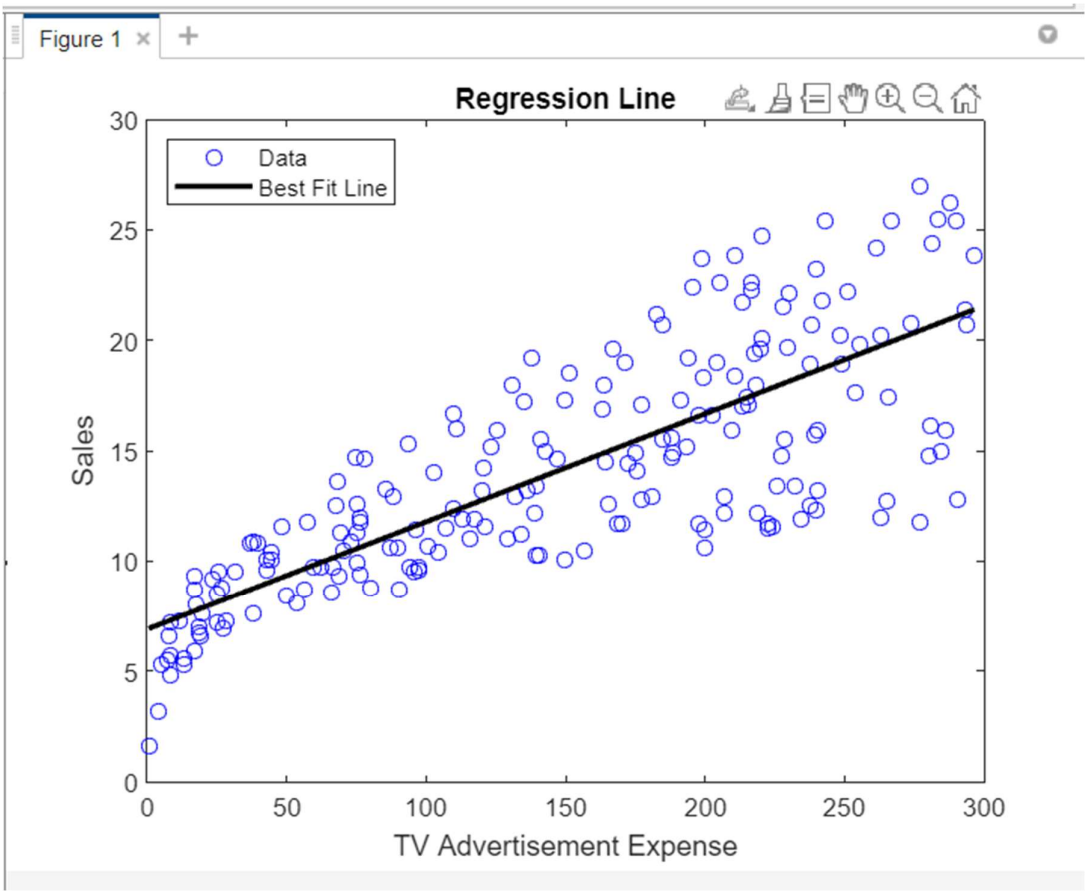
## squared_product.m:

```
function sq_error = squared_errors(x)
sq_error = (x - mean(x)).^2;
sq_error = sum(sq_error);
end
```

## Output:

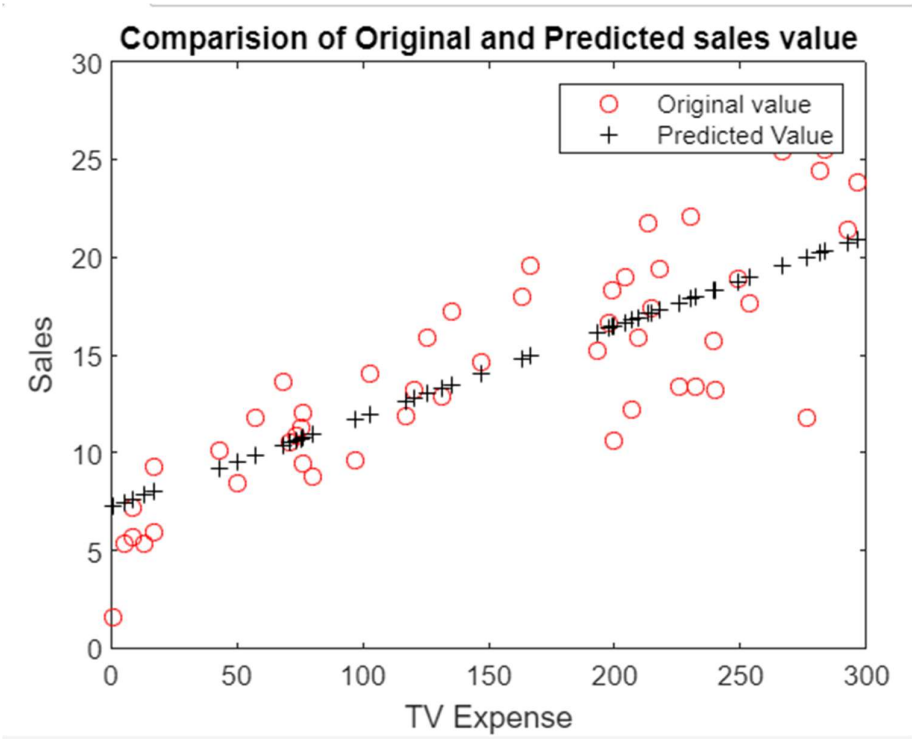
```
>> exp2
The slope of the line is 0.047699
The intercept of the line is 6.9317
for tv ad of rs 50000 the predicted sales is 9316.6313
Root Mean Squared error (RMS): 10.7696
r squared is: 0.57764
```

## Plot:




The above plot is obtained by making a train set out of random 150 values of the TV Advertisements and the corresponding Sales and calculating the slope by using mathematical formulas such as mean, intercept and slope. And after that plotting the equation of line obtained and the original values of data (x and y points) used for generating the plot.

## Comparison:




The above plot shows the comparison of the original and predicted sales value. It is generated using the test set of data (containing data values). The original values are straight away taken from the data set (array) and the predicted sales are calculated using the equation of line obtained using the train set. We can see that the predicted values do follow the general trend of the original data.

## Conclusion:

In this study, we conducted an experiment where we employed a univariate linear regression model within the MATLAB environment. The primary goal was to forecast sales figures based on the expenditures allocated to TV advertising. It is evident from the results that the mean square error exhibits a high value, and the R-Squared coefficient stands at around 50%. These findings imply that relying solely on TV advertising expenses is insufficient for predicting sales values accurately. Hence, the need for a multivariate linear regression approach becomes apparent.

# EXPERIMENT-2

**Title:** Implementation of Linear Regression using Python on Google Colab.

**Aim:** The aim of this experiment is to implement a concept of Linear Regression using Python in Google Colab Notebook to predict the effect of Price on Sq. ft. area using Bengaluru_House_Data.csv dataset.

## Introduction:

Linear regression is a fundamental statistical method employed to comprehend and articulate the relationship among two or more variables. It serves as a cornerstone in machine learning and is geared towards crafting a straight-line equation that best illustrates how alterations in one variable correspond to changes in another. By utilizing the values of one or more independent variables, this method enables us to forecast the value of a dependent variable. Linear regression finds its applicability across diverse domains such as science, social research, economics, and finance. In the realm of data science, it proves especially valuable, offering valuable insights into the interplay between variables and their impacts on one another.

### Slope and Intercept:

The slope within a linear regression model signifies the rate at which both the dependent and independent variables are altering. It indicates the anticipated alteration in the dependent variable for each incremental unit change in the independent variable. The point at which the regression line crosses the vertical axis is referred to as the intercept. It furnishes the initial value of the dependent variable when the independent variable is zero. In the context of a linear regression model, the combination of slope and intercept establishes the linear relationship between the variables.

## Procedure:

**Step 1: Data Exploration and Acquisition**
Begin by obtaining the Bengaluru_House dataset (Bengaluru_House_Data.csv), which contains information on advertising expenses and sales.
Next, initiate a new Google Colab notebook.
Upload the dataset into the notebook.
Import essential libraries like Pandas, Matplotlib, and Numpy.

**Step 2: Data Preprocessing and Cleaning**
Divide the dataset into training and test sets.
Determine which columns should be retained and which should be discarded.
Inspect for any missing data entries.
Replace missing values with the mode value.

**Step 3: Data Visualization**
Slice the data to extract values for the area (in sqft.) and House Prices columns.
Create a Matplotlib plot depicting the relationship between area and prices for visualization purposes.

## Code:

### 1)

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

#making a list of numbers between 0 to 11
l = [i for i in range(1,11)]
#printing the type of l
print("type of l: ",type(l))
#printing list
print("list: ",l)

#convering list into pandas
l_series = pd.Series(l)
print(l_series)
print("type of l_series: ",type(l_series))
print(l_series.dtype)
l_series.astype(float).dtype
```

**Output:**

```
type of l:  <class 'list'>
list:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
0     1
1     2
2     3
3     4
4     5
5     6
6     7
7     8
8     9
9    10
dtype: int64
type of l_series:  <class 'pandas.core.series.Series'>
int64
dtype('float64')
```

## 2)

```
#making list of square of numbers in range (1,11) for numbers greater than 4
lst = [i*i for i in range(1,11) if i > 4]
print(lst)
print()

st = 'ravimakwana'
#printing string
print(st)
print()

#printing consonants in string
print("Printing consonants in string: ")
l1 = [ch for ch in st if ch not in ['a','e','i','o','u']]
print(l1)
print()

#printing number of vowels in string
print("Printing number of vowels in string: ")
count=0
l2 = [i for i in st if i in ['a','e','i','o','u']]
print(l2)
print("\nNumber of vowels in string: ", len(l2))
```

**Output:**

```
[25, 36, 49, 64, 81, 100]

ravimakwana

Printing consonants in string:
['r', 'v', 'm', 'k', 'w', 'n']

Printing number of vowels in string:
['a', 'i', 'a', 'a', 'a']

Number of vowels in string:  5
```

**3)**

```
import os
#listing the files and directories
print(os.listdir())

beng_csv = "Bengaluru_House_Data.csv"
beng_df = pd.read_csv(beng_csv)
#printing first 5 rows of the csv data
print(beng_df.head())
print()
print()

#printing all columns
print()
print(beng_df.columns)
print()

#shape gives number of rows and columns
sz = beng_df.shape
print("Number of rows: ",sz[0])
print("Number of columns: ",sz[1])

#getting the column index of a column using column name
print()
print("Index of 'location' column: ", beng_df.columns.get_loc("location"))
print()

#Percentage of each columns having null values
print("Percentage of each columns having null values: ")
beng_df.isnull().sum()/beng_df.shape[0]*100
```

**Output:**

```
['.config', 'Bengaluru_House_Data.csv', 'sample_data']
              area_type   availability                  location       size  \
0  Super built-up  Area         19-Dec  Electronic City Phase II      2 BHK
1            Plot  Area  Ready To Move          Chikka Tirupathi  4 Bedroom
2        Built-up  Area  Ready To Move               Uttarahalli      3 BHK
3  Super built-up  Area  Ready To Move        Lingadheeranahalli      3 BHK
4  Super built-up  Area  Ready To Move                  Kothanur      2 BHK

   society total_sqft  bath  balcony   price
0  Coomee        1056   2.0      1.0   39.07
1  Theanmp       2600   5.0      3.0  120.00
2      NaN       1440   2.0      3.0   62.00
3  Soiewre       1521   3.0      1.0   95.00
4      NaN       1200   2.0      1.0   51.00



Index(['area_type', 'availability', 'location', 'size', 'society',
       'total_sqft', 'bath', 'balcony', 'price'],
      dtype='object')

Number of rows:  13320
Number of columns:  9

Index of 'location' column:  2

Percentage of each columns having null values:
area_type        0.000000
availability     0.000000
location         0.007508
size             0.120120
society         41.306306
total_sqft       0.000000
bath             0.548048
balcony          4.572072
price            0.000000
dtype: float64
```

## 4)

```
#using iloc to select and print specific rows and columns
print("Printing first 10 rows of society column")
col=beng_df.columns.get_loc('society')
beng_df.iloc[0:10,col]
```

**Output:**

```
Printing first 10 rows of society column
0    Coomee
1    Theanmp
2        NaN
3    Soiewre
4        NaN
5    DuenaTa
6    Jaades
7    Brway G
8        NaN
9        NaN
Name: society, dtype: object
```

## 5)

```
#printing all rows of 'size' column
print("Printing all rows of 'size' column")
beng_df.iloc[:,beng_df.columns.get_loc('size')]
```

**Output:**

```
Printing all rows of 'size' column
0            2 BHK
1        4 Bedroom
2            3 BHK
3            3 BHK
4            2 BHK
              ...
13315    5 Bedroom
13316        4 BHK
13317        2 BHK
13318        4 BHK
13319        1 BHK
Name: size, Length: 13320, dtype: object
```

## 6)

```
#printing all first 10 rows but not the column 'society'
print("Printing first 10 rows of csv but not the columns 'society'")
beng_df.loc[0:9, [col for col in beng_df.columns if col != 'society']]
```

**Output:**

Printing first 10 rows of csv but not the columns 'society'

| | area_type | availability | location | size | total_sqft | bath | balcony | price |
|---|---|---|---|---|---|---|---|---|
| 0 | Super built-up Area | 19-Dec | Electronic City Phase II | 2 BHK | 1056 | 2.0 | 1.0 | 39.07 |
| 1 | Plot Area | Ready To Move | Chikka Tirupathi | 4 Bedroom | 2600 | 5.0 | 3.0 | 120.00 |
| 2 | Built-up Area | Ready To Move | Uttarahalli | 3 BHK | 1440 | 2.0 | 3.0 | 62.00 |
| 3 | Super built-up Area | Ready To Move | Lingadheeranahalli | 3 BHK | 1521 | 3.0 | 1.0 | 95.00 |
| 4 | Super built-up Area | Ready To Move | Kothanur | 2 BHK | 1200 | 2.0 | 1.0 | 51.00 |
| 5 | Super built-up Area | Ready To Move | Whitefield | 2 BHK | 1170 | 2.0 | 1.0 | 38.00 |
| 6 | Super built-up Area | 18-May | Old Airport Road | 4 BHK | 2732 | 4.0 | NaN | 204.00 |
| 7 | Super built-up Area | Ready To Move | Rajaji Nagar | 4 BHK | 3300 | 4.0 | NaN | 600.00 |
| 8 | Super built-up Area | Ready To Move | Marathahalli | 3 BHK | 1310 | 3.0 | 1.0 | 63.25 |
| 9 | Plot Area | Ready To Move | Gandhi Bazar | 6 Bedroom | 1020 | 6.0 | NaN | 370.00 |

## 7)

```
#the mode function that gives the mode value of all the values in that
column
balcony_mode=beng_df['balcony'].mode()[0]
print("The mode of the 'balcony' column is: ",balcony_mode)
print()

#printing indices of all rows where the 'balcony' column in empty [result is
in the form of a list]
balcony_null_indices=beng_df[beng_df.loc[:,'balcony'].isnull() ==
True].index
print(balcony_null_indices)
print()

#filling all the empty values of 'balcony' with the mode value we got above
beng_df.loc[balcony_null_indices,'balcony']=balcony_mode
#printing the first 15 values to see if records are modified or not
print("First 15 records:")
print(beng_df.head(15))
```

**Output:**

```
The mode of the 'balcony' column is:  2.0

Int64Index([    6,     7,     9,    34,    40,    45,    56,    81,   140,
              146,
            ...
            13213, 13217, 13232, 13240, 13247, 13277, 13279, 13306, 13309,
            13316],
           dtype='int64', length=609)

First 15 records:
               area_type   availability                  location       size  \
0   Super built-up  Area         19-Dec  Electronic City Phase II      2 BHK
1             Plot  Area  Ready To Move          Chikka Tirupathi  4 Bedroom
2         Built-up  Area  Ready To Move               Uttarahalli      3 BHK
3   Super built-up  Area  Ready To Move        Lingadheeranahalli      3 BHK
4   Super built-up  Area  Ready To Move                  Kothanur      2 BHK
5   Super built-up  Area  Ready To Move                Whitefield      2 BHK
6   Super built-up  Area         18-May          Old Airport Road      4 BHK
7   Super built-up  Area  Ready To Move              Rajaji Nagar      4 BHK
8   Super built-up  Area  Ready To Move              Marathahalli      3 BHK
9             Plot  Area  Ready To Move              Gandhi Bazar  6 Bedroom
10  Super built-up  Area         18-Feb                Whitefield      3 BHK
11            Plot  Area  Ready To Move                Whitefield  4 Bedroom
12  Super built-up  Area  Ready To Move        7th Phase JP Nagar      2 BHK
13        Built-up  Area  Ready To Move                 Gottigere      2 BHK
14            Plot  Area  Ready To Move                   Sarjapur  3 Bedroom

    society total_sqft  bath  balcony   price
0   Coomee        1056   2.0      1.0   39.07
1   Theanmp       2600   5.0      3.0  120.00
2       NaN       1440   2.0      3.0   62.00
3   Soiewre       1521   3.0      1.0   95.00
4       NaN       1200   2.0      1.0   51.00
5   DuenaTa       1170   2.0      1.0   38.00
6   Jaades        2732   4.0      2.0  204.00
7   Brway G       3300   4.0      2.0  600.00
8       NaN       1310   3.0      1.0   63.25
9       NaN       1020   6.0      2.0  370.00
10      NaN       1800   2.0      2.0   70.00
11  Prrry M       2785   5.0      3.0  295.00
12  Shncyes       1000   2.0      1.0   38.00
13      NaN       1100   2.0      2.0   40.00
14  Skityer       2250   3.0      2.0  148.00
```

**8)**

```
#the median function that gives the median value of all the values in that
column
bath_median=beng_df['bath'].median()
print("Median of all values in 'bath' column is: ", bath_median)
print()

#filling all the empty values of 'bath' with the median value we got above
bath_null_indices=beng_df[beng_df.loc[:,'bath'].isnull() == True].index
```

```
beng_df.loc[bath_null_indices,'bath']=bath_median
beng_df.head(20)
```

**Output:**

```
Median of all values in 'bath' column is:  2.0
```

| | area_type | availability | location | size | society | total_sqft | bath | balcony | price |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Super built-up Area | 19-Dec | Electronic City Phase II | 2 BHK | Coomee | 1056 | 2.0 | 1.0 | 39.07 |
| 1 | Plot Area | Ready To Move | Chikka Tirupathi | 4 Bedroom | Theanmp | 2600 | 5.0 | 3.0 | 120.00 |
| 2 | Built-up Area | Ready To Move | Uttarahalli | 3 BHK | NaN | 1440 | 2.0 | 3.0 | 62.00 |
| 3 | Super built-up Area | Ready To Move | Lingadheeranahalli | 3 BHK | Soiewre | 1521 | 3.0 | 1.0 | 95.00 |
| 4 | Super built-up Area | Ready To Move | Kothanur | 2 BHK | NaN | 1200 | 2.0 | 1.0 | 51.00 |
| 5 | Super built-up Area | Ready To Move | Whitefield | 2 BHK | DuenaTa | 1170 | 2.0 | 1.0 | 38.00 |
| 6 | Super built-up Area | 18-May | Old Airport Road | 4 BHK | Jaades | 2732 | 4.0 | 2.0 | 204.00 |
| 7 | Super built-up Area | Ready To Move | Rajaji Nagar | 4 BHK | Brway G | 3300 | 4.0 | 2.0 | 600.00 |
| 8 | Super built-up Area | Ready To Move | Marathahalli | 3 BHK | NaN | 1310 | 3.0 | 1.0 | 63.25 |
| 9 | Plot Area | Ready To Move | Gandhi Bazar | 6 Bedroom | NaN | 1020 | 6.0 | 2.0 | 370.00 |
| 10 | Super built-up Area | 18-Feb | Whitefield | 3 BHK | NaN | 1800 | 2.0 | 2.0 | 70.00 |
| 11 | Plot Area | Ready To Move | Whitefield | 4 Bedroom | Prrry M | 2785 | 5.0 | 3.0 | 295.00 |
| 12 | Super built-up Area | Ready To Move | 7th Phase JP Nagar | 2 BHK | Shncyes | 1000 | 2.0 | 1.0 | 38.00 |
| 13 | Built-up Area | Ready To Move | Gottigere | 2 BHK | NaN | 1100 | 2.0 | 2.0 | 40.00 |
| 14 | Plot Area | Ready To Move | Sarjapur | 3 Bedroom | Skityer | 2250 | 3.0 | 2.0 | 148.00 |
| 15 | Super built-up Area | Ready To Move | Mysore Road | 2 BHK | PrntaEn | 1175 | 2.0 | 2.0 | 73.50 |
| 16 | Super built-up Area | Ready To Move | Bisuvanahalli | 3 BHK | Prityel | 1180 | 3.0 | 2.0 | 48.00 |
| 17 | Super built-up Area | Ready To Move | Raja Rajeshwari Nagar | 3 BHK | GrrvaGr | 1540 | 3.0 | 3.0 | 60.00 |
| 18 | Super built-up Area | Ready To Move | Ramakrishnappa Layout | 3 BHK | PeBayle | 2770 | 4.0 | 2.0 | 290.00 |
| 19 | Super built-up Area | Ready To Move | Manayata Tech Park | 2 BHK | NaN | 1100 | 2.0 | 2.0 | 48.00 |

## 9)

```
#changing the data types of columns from float to int for using it in
plotting a graph
print("Data type of 'bath' column Before changing: ",beng_df['bath'].dtype)
print("Data type of 'balcony' column Before changing:
",beng_df['balcony'].dtype)
beng_df['bath']=beng_df['bath'].astype('int')
beng_df['balcony']=beng_df['balcony'].astype('int')
print("Data type of 'bath' column After changing: ",beng_df['bath'].dtype)
print("Data type of 'balcony' column After changing:
",beng_df['balcony'].dtype)

x=beng_df.loc[1:10,'total_sqft']
y=beng_df.loc[1:10,'price']

#creating arrrays
x=np.array(x)
y=np.array(y)
```

```
print()

#plotting the graph
plt.title("Plot for Price vs Area of land")
plt.xlabel('Area (in sqft)')
plt.ylabel('Price')
plt.plot(x,y)
```

**Output:**

```
Data type of 'bath' column Before changing:  float64
Data type of 'balcony' column Before changing:  float64
Data type of 'bath' column After changing:  int64
Data type of 'balcony' column After changing:  int64

[<matplotlib.lines.Line2D at 0x7e7a19a42a70>]
```


**Conclusion:**

To sum up, the exploration of Data Analysis and Data Visualization through Python has offered valuable perspectives on the procedures involved in handling data, extracting significant insights, and conveying them visually, by plotting them on graph.