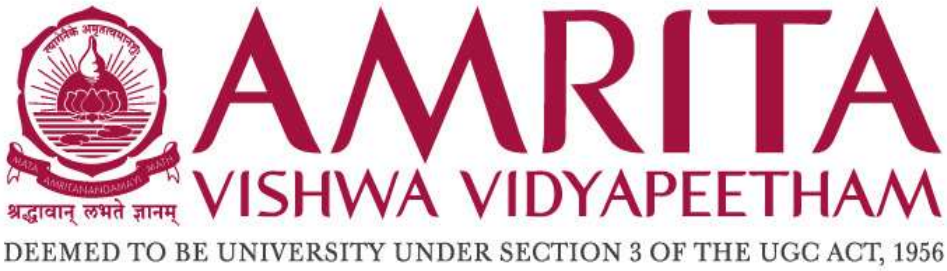
Bachelor of Technology (B. Tech.)

Computer and Communication Engineering (CCE)

Amrita School of Engineering, Coimbatore – 641112



19CCE213 Machine Learning and Artificial Intelligence

Fourth Semester, Third Year

Name: Santosh

Roll Number: CB.EN.U4CCE20053

Period: April 2022 – August 2022

Academic Year: 2021-22

Ms Suguna G

Assistant Professor

TABLE OF CONTENTS

EXPERIMENT 1 – INTRODUCTION TO PYTHON PROGRAMMING

AIM: A brief introduction to Python with the implementation of basic commands.

SOFTWARE REQUIRED:

Spyder IDE 5.1.5

Anaconda3 2021.11 (Python 3.9.7 64-bit)

Anaconda Inc., 2021.11

NUMPY: (https://www.w3schools.com/python/numpy/default.asp)

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

```
import numpy as np # Support for large, multi-dimensional arrays and matrices
```

1. **NumPy Creating Arrays:** – NumPy is used to work with arrays. The array object in NumPy is called ndarray. We can create a NumPy ndarray object by using the array() function. For example,

```
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
```

(i) 0-D Arrays – 0-D arrays, or Scalars, are the elements in an array. Each value in an array is a 0-D array. For example, create a 0-D array with the value 42.

```
arr = np.array(42)
print(arr)
```

(ii) 1-D Arrays – An array that has 0-D arrays as its elements are called a uni-dimensional or 1-D array. These are the most common and basic arrays. For example, create a 1-D array containing the values 1,2,3,4,5.

```
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

(iii) 2-D Arrays – An array that has 1-D arrays as its elements is called a 2-D array. These are often used to represent matrix or 2nd order tensors. For example, create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6.

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr)
```

2. **NumPy Array Indexing:** –

(i) Access Array Elements – Array indexing is the same as accessing an array element. You can access an array element by referring to its index number. The indexes in NumPy arrays start with 0, meaning that the first element has index 0, the second has index 1, etc. For example,

```
arr = np.array([1, 2, 3, 4])
print(arr[0]) # Get the first element from the array
print(arr[1]) # Get the second element from the array
print(arr[2] + arr[3]) # Get third and fourth elements from the
array and add them
```

(ii) Access 2-D Arrays – To access elements from 2-D arrays we can use comma-separated integers representing the dimension and the index of the element. Think of 2-D arrays like a table with rows and columns, where the row represents the dimension and the index represents the column. For example,

```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print("2nd element on 1st row: ", arr[0, 1]) # Access the element
on the 1st row, 2nd column
print("5th element on 2nd row: ", arr[1, 4]) # Access the element
on the 2nd row, 5th column
```

(iii) Access 3-D Arrays – To access elements from 3-D arrays we can use comma-separated integers representing the dimensions and the index of the element. For example, access the third element of the second array of the first array.

```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11,
12]]])
print(arr[0, 1, 2])
```

3. **NumPy Array Slicing:** – Slicing in python means taking elements from one given index to another given index. We pass a slice instead of an index like this: [start:end].

We can also define the step, like this: [start:end:step]. If we don't pass the start it's considered 0. If we don't pass the end it's considered the length of the array in that dimension. If we don't pass a step it's considered 1. For example,

```
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[1:5]) # Slice elements from index 1 to index 5 from the array
print(arr[4:]) # Slice elements from index 4 to the end of the array
print(arr[:4]) # Slice elements from the beginning to index 4 (not
included)
```

(i) Negative Slicing – Use the minus operator to refer to an index from the end. For example, slice from index 3 from the end to index 1 from the end.

```
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[-3:-1])
```

(ii) Step – Use the step value to determine the step of the slicing. For example,

```
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[1:5:2]) # Return every other element from index 1 to
index 5
print(arr[::2]) # Return every other element from the entire
array
```

MATPLOTLIB: (https://www.w3schools.com/python/matplotlib_intro.asp)

Matplotlib is a low-level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, and a few segments are written in C, Objective-C and Javascript for Platform compatibility. Most of the Matplotlib utilities lie under the pyplot submodule, and are usually imported under the plt alias:

```
import matplotlib.pyplot as plt # Provides an implicit way of plotting
```

1. **Matplotlib Pyplot:** – Now the Pyplot package can be referred to as plt. For example, draw a line in a diagram from position (0,0) to position (6,250).

```
xpoints = np.array([0, 6])
ypoints = np.array([0, 250])
plt.plot(xpoints, ypoints)
plt.show()
```

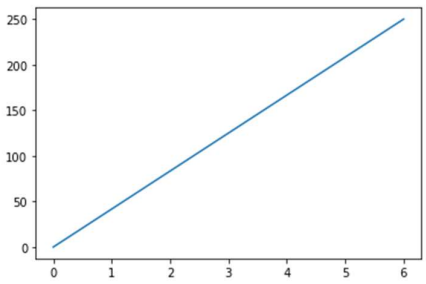


Figure 1 – Matplotplib Pyplot

2. **Matplotlib Plotting:** –
   (i) Plotting x and y points – The plot() function is used to draw points (markers) in a diagram. By default, the plot()

function draws a line from point to point. The function takes parameters for specifying points in the diagram.

Parameter 1 is an array containing the points on the x-axis. Parameter 2 is an array containing the points on the y-axis. If we need to plot a line from (1, 3) to (8, 10), we have to pass two arrays [1, 8] and [3, 10] to the plot function. For example, draw a line in a diagram from position (1, 3) to position (8, 10).

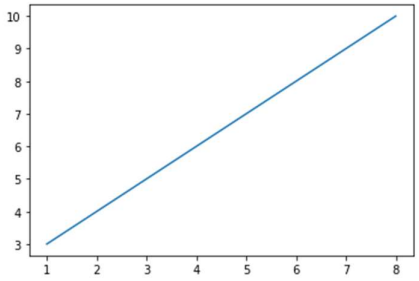


Figure 2 – Matplotlib Plotting

```
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints)
plt.show()
```

(ii) Plotting Without Line – To plot only the markers, you can use shortcut string notation parameter 'o', which means 'rings'. For example, draw two points in the diagram, one at position (1, 3) and one in position (8, 10).

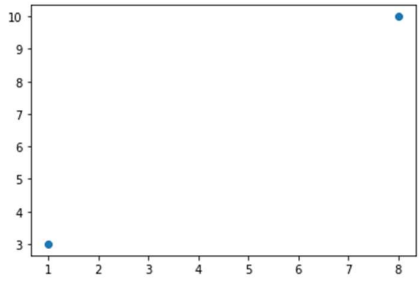


Figure 3 – Plotting Without Line

```
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints,
'o')
plt.show()
```

(iii) Multiple Points – You can plot as many points as you like, just make sure you have the same number of points on both axis. For example, draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10).

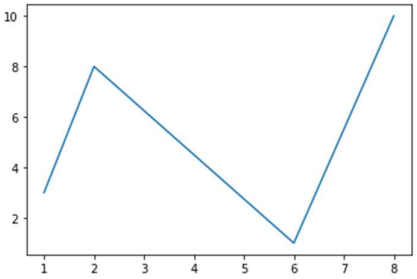


Figure 4 – Multiple Points

```
xpoints = np.array([1, 2, 6,
8])
ypoints = np.array([3, 8, 1,
10])
plt.plot(xpoints, ypoints)
plt.show()
```

3. **Matplotlib Markers:** –

(i) Markers – You can use the keyword argument marker to emphasize each point with a specified marker. For example, mark each point with a circle.

```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'o')
plt.show()
```

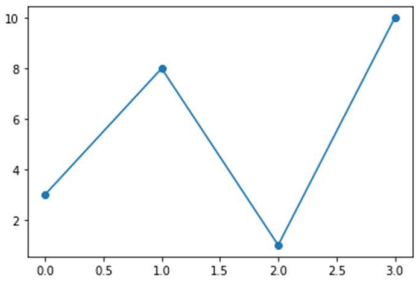


Figure 5 - Markers

(ii) Format Strings (fmt) – You can use also use the shortcut string notation parameter to specify the marker. This parameter is also called fmt, and is written with this syntax: marker|line|color. For example, mark each point with a circle.

```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, 'o:r')
plt.show()
```

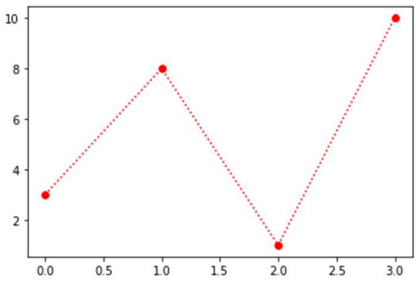


Figure 6 – Format Strings (fmt)

(iii) Marker Size – You can use the keyword argument marker size or the shorter version, ms to set the size of the markers. For example, set the size of the markers to 20.

```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'o', ms = 20)
plt.show()
```

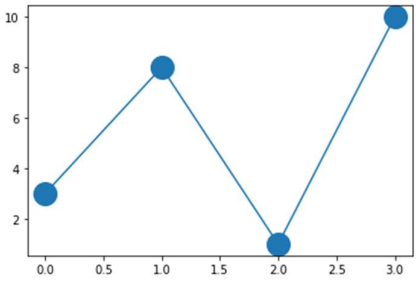


Figure 7 – Marker Size

SEABORN: (https://www.tutorialspoint.com/seaborn/index.htm)

It is an open-source, BSD-licensed Python library providing a high-level API for visualizing the data using Python programming language.

```
import seaborn as sb # Provides high-level API to visualize data
```

1. **Importing Data as Pandas DataFrame:** – In this section, we will import a dataset. This dataset loads as Pandas DataFrame by default. If there is any function in the Pandas

DataFrame, it works on this DataFrame. The following line of code will help you import the dataset,

```
df = sb.load_dataset('flights')
print (df.head())
```

2. **Histogram:** – Histograms represent the data distribution by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin. For example,

```
df = sb.load_dataset('iris')
sb.histplot(df['petal_length'], kde = False) # Here, kde flag is set to False. As a result, the representation of the kernel estimation plot will be removed and only the histogram is plotted.
plt.show()
```

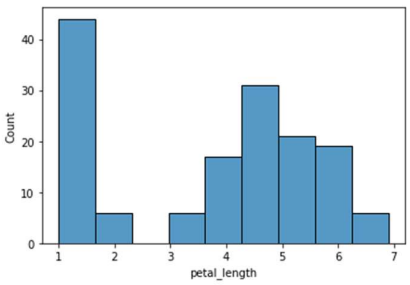


Figure 8 - Histogram

3. **Categorical Scatter Plots, stripplot():** – It is used when one of the variables under study is categorical. It represents the data in sorted order along any one of the axis. For example,

```
df = sb.load_dataset('iris')
sb.stripplot(x = "species", y = "petal_length", data = df)
plt.show()
```

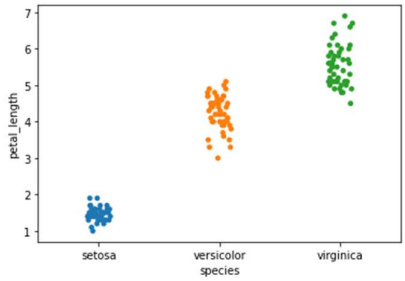


Figure 9 – Categorical Scatter Plots, stripplot()

4. **Statistical Estimation:** – In most situations, we deal with estimations of the whole distribution of the data. But when it comes to central tendency estimation, we need a specific way to summarize the distribution. Mean and median are the very often used techniques to estimate the central tendency of the distribution.

(i) Bar Plot – The barplot() shows the relation between a categorical variable and a continuous variable. The data is represented in rectangular bars where the length of the bar represents the proportion of the data in that category. The bar plot represents the estimate of central tendency. Let us use the 'titanic' dataset to learn bar plots.

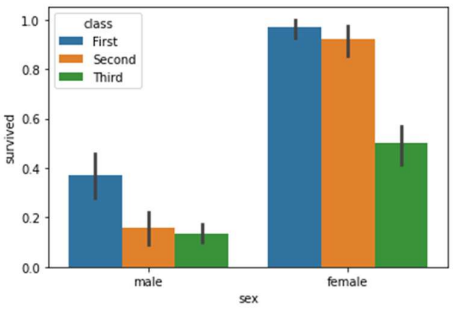


Figure 10 – Bar Plot

```
df = sb.load_dataset('titanic')
sb.barplot(x = "sex", y = "survived", hue = "class", data = df)
plt.show()
```

(ii) Point Plots – Point plots serve the same as bar plots but in a different style. Rather than the full bar, the value of the estimate is represented by the point at a certain height on the other axis. For example,

```
df = sb.load_dataset('titanic')
sb.pointplot(x = "sex", y = "survived", hue = "class", data = df)
plt.show()
```

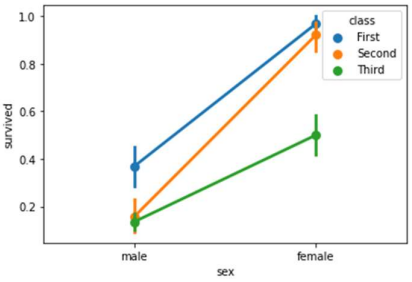


Figure 11 – Point Plots

5. **Plotting Wide Form Data:** – It is always preferable to use 'long-from' or 'tidy' datasets. But at times when we are left with no option rather but to use a 'wide-form' dataset, the same functions can also be applied to "wide-form" data in a variety of formats, including Pandas Data Frames or two-dimensional NumPy arrays. These objects should be passed directly to the data parameter the x and y variables must be specified as strings. For example,

```
df = sb.load_dataset('iris')
sb.boxplot(data = df, orient = "h")
plt.show()
```

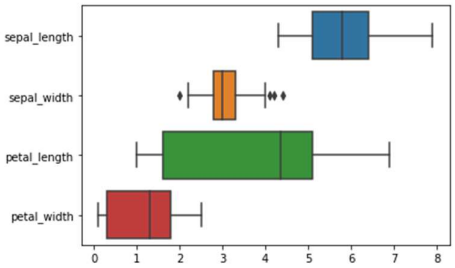


Figure 12 – Plotting Wide Form Data

PANDAS: (https://www.w3schools.com/python/pandas/default.asp)

It is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

```
import pandas as pd # Library for working with data sets
```

1. **Series:** – It is like a column in a table. It is a one-dimensional array holding data of any type. For example, create a simple Pandas Series from a list.

```
a = [1, 7, 2]
myvar = pd.Series(a)
print(myvar)
```

```
print(myvar[0]) # Return the first value of the series
```

(i) Create Labels – With the index argument, you can name your labels. For example,

```
myvar = pd.Series(a, index = ["x", "y", "z"])
print(myvar)
print(myvar["y"]) # Return the value of "y"
```

(ii) Key/Value Objects as Series – You can also use a key/value object, like a dictionary, when creating a Series. For example, create a simple Pandas Series from a dictionary.

```
calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories)
print(myvar)
```

(iii) DataFrames – Data sets in Pandas are usually multi-dimensional tables, called DataFrames. Series is like a column, a DataFrame is a whole table. For example, create a DataFrame from two Series.

```
data = {
"calories": [420, 380, 390],
"duration": [50, 40, 45]
}
myvar = pd.DataFrame(data)
print(myvar)
```

2. **Read CSV Files:** – A simple way to store big data sets is to use CSV files (comma-separated files). CSV files contain plain text and are a well know format that can be read by everyone including Pandas. For example, load the CSV into a DataFrame.

```
df = pd.read_csv('data.csv')
print(df.to_string())
```

3. **Analyzing DataFrames:** –

(i) Viewing the Data – One of the most used methods for getting a quick overview of the DataFrame, is the head() method. The head() method returns the headers and a specified number of rows, starting from the top. For example, get a quick overview by printing the first 10 rows of the DataFrame.

```
df = pd.read_csv('data.csv')
print(df.head(10))
```

(ii) Info About the Data – The DataFrames object has a method called info(), that gives you more information about the data set. For example,

```
df = pd.read_csv('data.csv')
print(df.info())
```

SCIPY: (https://www.w3schools.com/python/scipy/index.php)

SciPy is a scientific computation library that uses NumPy underneath. SciPy stands for Scientific Python. It provides more utility functions for optimization, stats and signal processing. Like NumPy, SciPy is open source so we can use it freely. SciPy was created by NumPy's creator Travis Oliphant.

1. **Constants in SciPy:** – As SciPy is more focused on scientific implementations, it provides many built-in scientific constants. These constants can be helpful when you are working with Data Science. For example, print the constant value of PI.

```
from scipy import constants
print("\nConstants in SciPy:")
print(constants.pi)
```

2. **Roots of an Equation**: – NumPy is capable of finding roots for polynomials and linear equations, but it can not find roots for non-linear equations, like this one:

$$x + \cos(x)$$

For that, you can use SciPy's optimize.root function. This function takes two required arguments:

(i) fun - a function representing an equation,
(ii) x0 - an initial guess for the root.

The function returns an object with information regarding the solution. The actual solution is given under attribute x of the returned object. For example, find root of the equation x + cos(x).

```
from scipy.optimize import root
from math import cos
def eqn(x):
      return x + cos(x)
myroot = root(eqn, 0)
print(myroot.x) # Find root of the equation x + cos(x)
print("\n")
print(myroot) # Print all information about the solution (not just x
which is the root)
```

SCIKIT-LEARN: (https://www.tutorialspoint.com/scikit_learn/index.htm)

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistency interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

1. **Dataset Loading:** – A collection of data is called a dataset. It is having the following two components,

(i) Features – The variables of data are called their features. They are also known as predictors, inputs or attributes.
- Feature Matrix – It is the collection of features, in case there is more than one.
- Feature Names – It is the list of all the names of the features.

(ii) Response – It is the output variable that depends upon the feature variables. They are also known as target, label or output.
- Response Vector – It is used to represent the response column. Generally, we have just one response column.
- Target Names – These represent the possible values taken by a response vector.

Scikit-learn has a few example datasets like iris and digits for classification and the Boston house prices for regression. Following is an example to load an iris dataset,

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
print("\nFirst 10 rows of X:\n", X[:10])
```

2. **Splitting the Dataset**: – To check the accuracy of our model, we can split the dataset into two pieces-a training set and a testing set. Use the training set to train the model and the testing set to test the model. After that, we can evaluate how well our model did.

   The following example will split the data into a 70:30 ratio, i.e. 70% of data will be used as training data and 30% will be used as testing data. The dataset is the iris dataset as in the above example.

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 1)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

RESULT:

A brief introduction to Python is documented along with the syntaxes and examples from basic concepts and commands and all simulation results were verified successfully.

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

Restarting kernel...
```

```
In [1]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 1 - Introduction to
Python Programming/Expt_1_Code_Matplotlib.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/
Subject Materials/Semester IV/19CCE213 - Machine Learning and Artificial Intelligence/Lab/
Experiment 1 - Introduction to Python Programming'

Pyplot:

Plotting x and y points:

Plotting Without Line:

Multiple Points:

Markers:

Format Strings (fmt):

Marker Size:

In [2]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 1 - Introduction to
Python Programming/Expt_1_Code_NumPy.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/
Subject Materials/Semester IV/19CCE213 - Machine Learning and Artificial Intelligence/Lab/
Experiment 1 - Introduction to Python Programming'


[1 2 3 4 5]
<class 'numpy.ndarray'>


42


[1 2 3 4 5]


[[1 2 3]
 [4 5 6]]


1
2
7
```

```
2nd element on 1st row:  2
5th element on 2nd row:  10


6


[2 3 4 5]
[5 6 7]
[1 2 3 4]


[5 6]


[2 4]
[1 3 5 7]
```

In [3]:
```
'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 1 - Introduction to Python Programming/Expt_1_Code_Pandas.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 1 - Introduction to Python Programming'

Pandas Series:
0    1
1    7
2    2
dtype: int64
1

Create Labels:
x    1
y    7
z    2
dtype: int64
7

Key/Value Objects as Series:
day1    420
day2    380
day3    390
dtype: int64

DataFrames:
   calories  duration
0       420        50
1       380        40
2       390        45

Read CSV Files:
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
```

3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.0
6	60	110	136	374.0
7	45	104	134	253.3
8	30	109	133	195.1
9	60	98	124	269.0
10	60	103	147	329.3
11	60	100	120	250.7
12	60	106	128	345.3
13	60	104	132	379.3
14	60	98	123	275.0
15	60	98	120	215.2
16	60	100	120	300.0
17	45	90	112	NaN
18	60	103	123	323.0
19	45	97	125	243.0
20	60	108	131	364.2
21	45	100	119	282.0
22	60	130	101	300.0
23	45	105	132	246.0
24	60	102	126	334.5
25	60	100	120	250.0
26	60	92	118	241.0
27	60	103	132	NaN
28	60	100	132	280.0
29	60	102	129	380.3
30	60	92	115	243.0
31	45	90	112	180.1
32	60	101	124	299.0
33	60	93	113	223.0
34	60	107	136	361.0
35	60	114	140	415.0
36	60	102	127	300.0
37	60	100	120	300.0
38	60	100	120	300.0
39	45	104	129	266.0
40	45	90	112	180.1
41	60	98	126	286.0
42	60	100	122	329.4
43	60	111	138	400.0
44	60	111	131	397.0
45	60	99	119	273.0
46	60	109	153	387.6
47	45	111	136	300.0
48	45	108	129	298.0
49	60	111	139	397.6
50	60	107	136	380.2
51	80	123	146	643.1
52	60	106	130	263.0
53	60	118	151	486.0
54	30	136	175	238.0
55	60	121	146	450.7
56	60	118	121	413.0
57	45	115	144	305.0

58	20	153	172	226.4
59	45	123	152	321.0
60	210	108	160	1376.0
61	160	110	137	1034.4
62	160	109	135	853.0
63	45	118	141	341.0
64	20	110	130	131.4
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
68	20	106	136	110.4
69	300	108	143	1500.2
70	150	97	129	1115.0
71	60	109	153	387.6
72	90	100	127	700.0
73	150	97	127	953.2
74	45	114	146	304.0
75	90	98	125	563.2
76	45	105	134	251.0
77	45	110	141	300.0
78	120	100	130	500.4
79	270	100	131	1729.0
80	30	159	182	319.2
81	45	149	169	344.0
82	30	103	139	151.1
83	120	100	130	500.0
84	45	100	120	225.3
85	30	151	170	300.0
86	45	102	136	234.0
87	120	100	157	1000.1
88	45	129	103	242.0
89	20	83	107	50.3
90	180	101	127	600.1
91	45	107	137	NaN
92	30	90	107	105.3
93	15	80	100	50.5
94	20	150	171	127.4
95	20	151	168	229.4
96	30	95	128	128.2
97	25	152	168	244.2
98	30	109	131	188.2
99	90	93	124	604.1
100	20	95	112	77.7
101	90	90	110	500.0
102	90	90	100	500.0
103	90	90	100	500.4
104	30	92	108	92.7
105	30	93	128	124.0
106	180	90	120	800.3
107	30	90	120	86.2
108	90	90	120	500.3
109	210	137	184	1860.4
110	60	102	124	325.2
111	45	107	124	275.0
112	15	124	139	124.2

113	45	100	120	225.3
114	60	108	131	367.6
115	60	108	151	351.7
116	60	116	141	443.0
117	60	97	122	277.4
118	60	105	125	NaN
119	60	103	124	332.7
120	30	112	137	193.9
121	45	100	120	100.7
122	60	119	169	336.7
123	60	107	127	344.9
124	60	111	151	368.5
125	60	98	122	271.0
126	60	97	124	275.3
127	60	109	127	382.0
128	90	99	125	466.4
129	60	114	151	384.0
130	60	104	134	342.5
131	60	107	138	357.5
132	60	103	133	335.0
133	60	106	132	327.5
134	60	103	136	339.0
135	20	136	156	189.0
136	45	117	143	317.7
137	45	115	137	318.0
138	45	113	138	308.0
139	20	141	162	222.4
140	60	108	135	390.0
141	60	97	127	NaN
142	45	100	120	250.4
143	45	122	149	335.4
144	60	136	170	470.2
145	45	106	126	270.8
146	60	107	136	400.0
147	60	112	146	361.9
148	30	103	127	185.0
149	60	110	150	409.4
150	60	106	134	343.0
151	60	109	129	353.2
152	60	109	138	374.0
153	30	150	167	275.8
154	60	105	128	328.0
155	60	111	151	368.5
156	60	97	131	270.4
157	60	100	120	270.4
158	60	114	150	382.8
159	30	80	120	240.9
160	30	85	120	250.4
161	45	90	130	260.4
162	45	95	130	270.0
163	45	100	140	280.9
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4

```
168        75    125       150     330.4

Viewing the Data:
   Duration  Pulse  Maxpulse  Calories
0        60    110       130     409.1
1        60    117       145     479.0
2        60    103       135     340.0
3        45    109       175     282.4
4        45    117       148     406.0
5        60    102       127     300.0
6        60    110       136     374.0
7        45    104       134     253.3
8        30    109       133     195.1
9        60     98       124     269.0

Info About the Data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  169 non-null    int64
 1   Pulse     169 non-null    int64
 2   Maxpulse  169 non-null    int64
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None

In [4]:          'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 1 - Introduction to
Python Programming/Expt_1_Code_SciPy_Sklearn.py'      ='E:/Plan B/Amrita Vishwa
Vidyapeetham/Subject Materials/Semester IV/19CCE213 - Machine Learning and Artificial
Intelligence/Lab/Experiment 1 - Introduction to Python Programming'

Constants in SciPy:
3.141592653589793

Roots of an Equation:
[-0.73908513]


    fjac: array([[-1.]])
     fun: array([0.])
 message: 'The solution converged.'
    nfev: 9
     qtf: array([-2.66786593e-13])
       r: array([-1.67361202])
  status: 1
 success: True
       x: array([-0.73908513])

Dataset Loading:
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
```

```
Target names: ['setosa' 'versicolor' 'virginica']

First 10 rows of X:
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]

Splitting the Dataset:
(105, 4)
(45, 4)
(105,)
(45,)

In [5]:          'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 1 - Introduction to
Python Programming/Expt_1_Code_Seaborn.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/
Subject Materials/Semester IV/19CCE213 - Machine Learning and Artificial Intelligence/Lab/
Experiment 1 - Introduction to Python Programming'

Importing Datasets and Libraries (Importing Data as Pandas DataFrame):
   year month  passengers
0  1949   Jan         112
1  1949   Feb         118
2  1949   Mar         132
3  1949   Apr         129
4  1949   May         121

Histogram:

Plotting Categorical Data (stripplot()):

Statistical Estimation (Bar Plot):

Statistical Estimation (Point Plots):

Plotting Wide Form Data:

In [6]:
```

EXPERIMENT 2 – MEASURE OF CENTRAL TENDENCY AND DATA DISPERSION

AIM: To introduce typical measures for basic statistical data description and overview data visualization techniques for various kinds of data.

SOFTWARE REQUIRED:

Spyder IDE 5.1.5

Anaconda3 2021.11 (Python 3.9.7 64-bit)

Anaconda Inc., 2021.11

DATA SET: Real Estate Data Set

PYTHON CODE:

```
import matplotlib.pyplot as plt # Provides an implicit way of plotting
import numpy as np # Support for large, multi-dimensional arrays and matrices
import pandas as pd # Library for working with data sets
import seaborn as sb # Provides high-level API to visualize data

# The mean() function returns the values' mean for the requested axis.
def mean():

    print("\nMean:")

    mean_X2 = df['X2 house age'].mean()
    mean_X3 = df['X3 distance to the nearest MRT station'].mean()
    mean_X4 = df['X4 number of convenience stores'].mean()
    mean_Y = df['Y house price of unit area'].mean()

    print("X2. The age of house in years: " + str(round(mean_X2, 1)))
    print("X3. The distance to nearest MRT station in meters: " +
str(round(mean_X3, 5)))
    print("X4. The number of convenience stores within walking distance: " +
str(round(mean_X4)))
    print("Y. House price per local unit area: " + str(round(mean_Y, 1)))

# The median() method returns a series with the median value of each column.
def median():

    print("\nMedian:")
```

```
    median_X2 = df['X2 house age'].median()
    median_X3 = df['X3 distance to the nearest MRT station'].median()
    median_X4 = df['X4 number of convenience stores'].median()
    median_Y = df['Y house price of unit area'].median()

    print("X2. The age of house in years: " + str(median_X2))
    print("X3. The distance to nearest MRT station in meters: " +
str(median_X3))
    print("X4. The number of convenience stores within walking distance: " +
str(int(median_X4)))
    print("Y. House price per local unit area: " + str(median_Y))

# Get each element’s mode(s) along the selected axis.
def mode():

    print("\nMode:")

    mode_X2 = df['X2 house age'].mode()[0]
    mode_X4 = df['X4 number of convenience stores'].mode()[0]

    """
    You can also use mode() to calculate the mode of the sequence, but this
returns a list of numbers, so you'll have to use mode()[0] to get the first
one.
    """

    print("X2. The age of house in years: " + str(int(mode_X2)))
    print("X4. The number of convenience stores within walking distance: " +
str(mode_X4))

    if (int(mode_X2) == 0 or int(mode_X4) == 0):
        print("(Please note that a null value in database is used when the
value in a column is unknown. By default, missing values are not
considered.)")

#  Compute the qth quantile of the given data (array elements) along the
specified axis.
def print_five_number_summary_IQR_outlier(minimum, Q1, median, Q3, maximum):

    print("Minimum = ", minimum)
    print("Q1 quantile = ", Q1)
    print("Median =", median)
    print("Q3 quantile = ", Q3)
    print("Maximum =", maximum)

    IQR = Q3 - Q1
    print("Inter-Quartile Range (IQR) = ", IQR)
    outlier = 1.5 * IQR
```

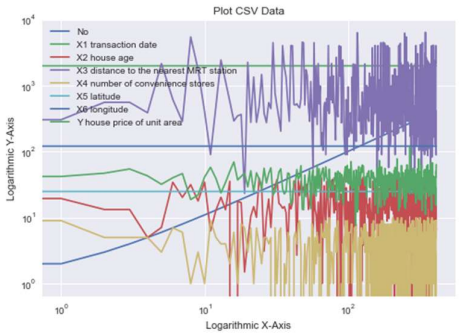
```
    print("Outlier (1.5 X IQR) = ", outlier)

def calc_five_number_summary_variance_standard_deviation():

    print("\nX2. The age of house in years -")

    min_X2 = df['X2 house age'].min()
    Q1_X2 = np.quantile(df['X2 house age'], .25)
    median_X2 = df['X2 house age'].median()
    Q3_X2 = np.quantile(df['X2 house age'], .75)
    max_X2 = df['X2 house age'].max()
    print_five_number_summary_IQR_outlier(min_X2, Q1_X2, median_X2, Q3_X2,
max_X2)

    var_X2 = df['X2 house age'].var()
    print("Variance = ", var_X2)
    std_X2 = df['X2 house age'].std()
    print("Standard Deviation = ", std_X2)

    print("\nX3. The distance to nearest MRT station in meters -")

    min_X3 = df['X3 distance to the nearest MRT station'].min()
    Q1_X3 = np.quantile(df['X3 distance to the nearest MRT station'], .25)
    median_X3 = df['X3 distance to the nearest MRT station'].median()
    Q3_X3 = np.quantile(df['X3 distance to the nearest MRT station'], .75)
    max_X3 = df['X3 distance to the nearest MRT station'].max()
    print_five_number_summary_IQR_outlier(min_X3, Q1_X3, median_X3, Q3_X3,
max_X3)

    var_X3 = df['X3 distance to the nearest MRT station'].var()
    print("Variance = ", var_X3)
    std_X3 = df['X3 distance to the nearest MRT station'].std()
    print("Standard Deviation = ", std_X3)

    print("\nX4. The number of convenience stores within walking distance -")

    min_X4 = df['X4 number of convenience stores'].min()
    Q1_X4 = np.quantile(df['X4 number of convenience stores'], .25)
    median_X4 = df['X4 number of convenience stores'].median()
    Q3_X4 = np.quantile(df['X4 number of convenience stores'], .75)
    max_X4 = df['X4 number of convenience stores'].max()
    print_five_number_summary_IQR_outlier(min_X4, Q1_X4, median_X4, Q3_X4,
max_X4)

    var_X4 = df['X4 number of convenience stores'].var()
    print("Variance = ", var_X4)
    std_X4 = df['X4 number of convenience stores'].std()
    print("Standard Deviation = ", std_X4)
```

```
    print("\nY. House price per local unit area -")

    min_Y = df['Y house price of unit area'].min()
    Q1_Y = np.quantile(df['Y house price of unit area'], .25)
    median_Y = df['Y house price of unit area'].median()
    Q3_Y = np.quantile(df['Y house price of unit area'], .75)
    max_Y = df['Y house price of unit area'].max()
    print_five_number_summary_IQR_outlier(min_Y, Q1_Y, median_Y, Q3_Y, max_Y)

    var_Y = df['Y house price of unit area'].var()
    print("Variance = ", var_Y)
    std_Y = df['Y house price of unit area'].std()
    print("Standard Deviation = ", std_Y)

def plot_data():

    print("\nPlot CSV Data:")

    """
    The method yscale() or xscale() takes a single value as a parameter which
is the type of conversion of the scale, to convert axes to a logarithmic scale
we pass the “log” keyword or the matplotlib.scale
    """

    plt.xscale('log'); plt.xlabel("Logarithmic X-Axis")
    plt.yscale('log'); plt.ylabel("Logarithmic Y-Axis")
    plt.title("Plot CSV Data")
    plt.plot(df)
    plt.legend(df)
    plt.show()

def boxplot():
    print("\nBoxplot: Graphic display of five-number summary.")
    sb.boxplot(data = df, orient = 'h')
    plt.show()

def histogram():

    print("\nHistogram: X-axis are values and Y-axis represent frequencies.")

    """
    Here, the kde flag is set to False. As a result, the representation of the
kernel estimation plot will be removed and only the histogram is plotted.
    """

    plt.show()
    sb.histplot(df['X2 house age'], kde = False)
```

```
    plt.show()
    sb.histplot(df['X3 distance to the nearest MRT station'], kde = False)
    plt.show()
    sb.histplot(df['X4 number of convenience stores'], kde = False)
    plt.show()
    sb.histplot(df['Y house price of unit area'], kde = False)
    plt.show()

def scatter_plot():

    print("\nScatter Plot: Each pair of values is a pair of coordinates and
plotted as points in the plane.")
    house_age = df['X2 house age']
    house_price = df['Y house price of unit area']

    plt.xlabel("X2: The age of house in years")
    plt.ylabel("Y: House price per local unit area")
    plt.title("Relationship Between House Price and House Age")
    plt.scatter(house_age, house_price, cmap="Blues", s=100, alpha=0.6,
edgecolor='black', linewidth=1)

    cbar = plt.colorbar()
    cbar.set_label('Intensity Ratio')
    plt.tight_layout()
    plt.show()

# Driver Code: main() ; Execution starts here.

"""
There is a Unicode character '\u0332', COMBINING LOW LINE*, which acts as an
underline on the character that precedes it in a string. The centre () method
will centre align the string, using a specified character (space is the
default) as the fill character.
"""

print("\n")
heading = "Identification of Response Variable & Regressor Variables"
print('{:s}'.format('\u0332'.join(heading.center(100))))

print("\nThere are six regressor variables (from X1 to X6) and one response
variable (namely, y):")
print("No - Serial Number")
print("X1 - Transaction Date")
print("X2 - Age of House in year(s)")
print("X3 - Distance to Nearest MRT station in meter(s)")
print("X4 - Number of Convenience Stores within Walking Distance")
print("X5 - Latitude Coordinates")
print("X6 - Longitude Coordinates")
```

```
print("Y - House Price per Local Unit Area")

print("\n")
heading = "Read CSV File"
print('{:s}'.format('\u0332'.join(heading.center(100))))
df = pd.read_csv("Real Estate Data Set.csv"); print(df)

print("\n")
heading = "Measuring the Central Tendency"
print('{:s}'.format('\u0332'.join(heading.center(100))))
mean(); median(); mode()

print("\n")
heading = "Measuring the Dispersion of Data"
print('{:s}'.format('\u0332'.join(heading.center(100))))
calc_five_number_summary_variance_standard_deviation()

print("\n")
heading = "Graphic Displays of Basic Statistical Descriptions"
print('{:s}'.format('\u0332'.join(heading.center(100))))
plt.style.use('seaborn') # To get seaborn type plot
plot_data(); boxplot(); histogram(); scatter_plot()
```

PLOTS:



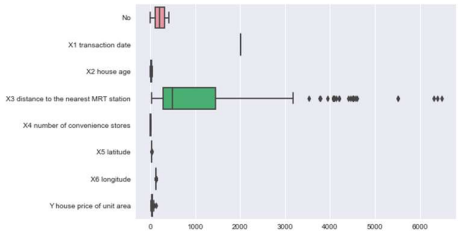


Figure 2. Boxplot

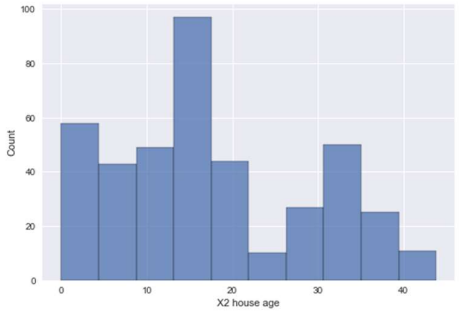


Figure 3. Histogram - X2 house age

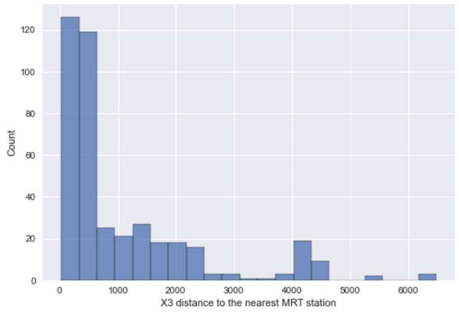


Figure 4. Histogram - X3 distance to the nearest MRT station

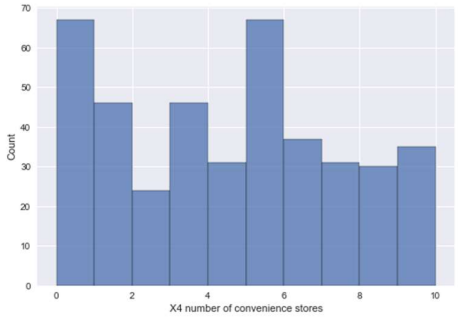


Figure 5. Histogram - X4 number of convenience stores

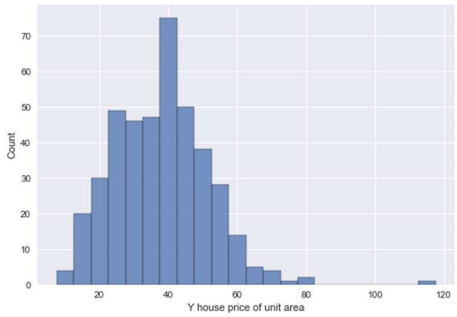


Figure 6. Histogram - Y house price of unit area

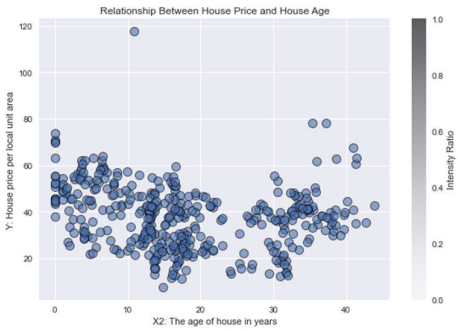


Figure 7. Scatter Plot

RESULT:

Familiarized with typical measures for basic statistical data description and reviewed the data visualization techniques for various kinds of data. All the simulation results were verified successfully.

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

Restarting kernel...
```

In [1]:
```
'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 2 - Measure of Central Tendency and Data Dispersion/Expt_2_Code.py' ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 2 - Measure of Central Tendency and Data Dispersion'
```

Identification of Response Variable & Regressor Variables

There are six regressor variables (from X1 to X6) and one response variable (namely, y):
No - Serial Number
X1 - Transaction Date
X2 - Age of House in year(s)
X3 - Distance to Nearest MRT station in meter(s)
X4 - Number of Convenience Stores within Walking Distance
X5 - Latitude Coordinates
X6 - Longitude Coordinates
Y - House Price per Local Unit Area

Read CSV File

	No	X1 transaction date	...	X6 longitude	Y house price of unit area
0	1	2012.917	...	121.54024	37.9
1	2	2012.917	...	121.53951	42.2
2	3	2013.583	...	121.54391	47.3
3	4	2013.500	...	121.54391	54.8
4	5	2012.833	...	121.54245	43.1
..	...	...	...	...	...
409	410	2013.000	...	121.50381	15.4
410	411	2012.667	...	121.54310	50.0
411	412	2013.250	...	121.53986	40.6
412	413	2013.000	...	121.54067	52.5
413	414	2013.500	...	121.54310	63.9

[414 rows x 8 columns]

Measuring the Central Tendency

Mean:
X2. The age of house in years: 17.7
X3. The distance to nearest MRT station in meters: 1083.88569
X4. The number of convenience stores within walking distance: 4
Y. House price per local unit area: 38.0

Median:

X2. The age of house in years: 16.1
X3. The distance to nearest MRT station in meters: 492.2313
X4. The number of convenience stores within walking distance: 4
Y. House price per local unit area: 38.45

Mode:
X2. The age of house in years: 0
X4. The number of convenience stores within walking distance: 0
(Please note that a null value in database is used when the value in a column is unknown. By default, missing values are not considered.)

Measuring the Dispersion of Data

X2. The age of house in years -
Minimum = 0.0
Q1 quantile = 9.025
Median = 16.1
Q3 quantile = 28.15
Maximum = 43.8
Inter-Quartile Range (IQR) = 19.125
Outlier (1.5 X IQR) = 28.6875
Variance = 129.7887038401704
Standard Deviation = 11.392484533242536

X3. The distance to nearest MRT station in meters -
Minimum = 23.38284
Q1 quantile = 289.3248
Median = 492.2313
Q3 quantile = 1454.279
Maximum = 6488.021
Inter-Quartile Range (IQR) = 1164.9542000000001
Outlier (1.5 X IQR) = 1747.4313000000002
Variance = 1592920.6308205703
Standard Deviation = 1262.1095954078514

X4. The number of convenience stores within walking distance -
Minimum = 0
Q1 quantile = 1.0
Median = 4.0
Q3 quantile = 6.0
Maximum = 10
Inter-Quartile Range (IQR) = 5.0
Outlier (1.5 X IQR) = 7.5
Variance = 8.676334350984305
Standard Deviation = 2.945561805663617

Y. House price per local unit area -
Minimum = 7.6
Q1 quantile = 27.7
Median = 38.45
Q3 quantile = 46.6
Maximum = 117.5
Inter-Quartile Range (IQR) = 18.900000000000002
Outlier (1.5 X IQR) = 28.35

Variance = 185.13650746862245
Standard Deviation = 13.606487697735314

Graphic Displays of Basic Statistical Descriptions

Plot CSV Data:

Boxplot: Graphic display of five-number summary.

Histogram: X-axis are values and Y-axis represent frequencies.

Scatter Plot: Each pair of values is a pair of coordinates and plotted as points in the plane.

In [2]:

EXPERIMENT 3 – DATA CLEANING

AIM: Study basic methods of data cleaning, look at ways of handling missing values, explain data smoothing techniques and discuss approaches to data cleaning as a process.

SOFTWARE REQUIRED:

Spyder IDE 5.1.5

Anaconda3 2021.11 (Python 3.9.7 64-bit)

Anaconda Inc., 2021.11

DATA SET: Real Estate Data Set

PYTHON CODE:

```
import numpy as np # Support for large, multi-dimensional arrays and matrices
import pandas as pd # Library for working with data sets

"""
There is a Unicode character '\u0332', COMBINING LOW LINE*, which acts as an
underline on the character that precedes it in a string. The center() method
will center align the string, using a specified character (space is the
default) as the fill character.
"""

def print_csv_file(df):
    print("\n")
    heading = "Read CSV File"
    print('{:s}'.format('\u0332'.join(heading.center(100))))
    print(df.to_string())

def clean_empty_cells(): # Clean Empty Cells

    def remove_rows(): # Remove Rows

        print("\nRemove Rows - This is usually OK, since data sets can be very
big, and removing a few rows will not have a big impact on the result.")

        change = str(input("Do you want to change the original DataFrame?
(y/n): "))

        if (change == 'N' or change == 'n'): # Return a new Data Frame with no
empty cells
```

```
            new_df = df.dropna() # By default, the dropna() method returns a
new DataFrame, and will not change the original.
            print_csv_file(new_df)

        elif (change == 'Y' or change == 'y'): # Remove all rows with NULL
values
            df.dropna(inplace = True) # Use the inplace = True argument
            print_csv_file(df)
            # Now, the dropna(inplace = True) will NOT return a new DataFrame,
but it will remove all rows containing NULL values from the original
DataFrame.

        else:
            print("Error: Invalid Input! Please try again.")

    def replace_empty_values(): # Replace Empty Values

        print("\nReplace Empty Values - This way you do not have to delete
entire rows just because of some empty cells. The fillna() method allows us to
replace empty cells with a value.")

        value = input("Enter the value with which you want to replace the
empty cell: ")
        df.fillna(value, inplace = True) # Replace NULL values with the number
"value"
        print_csv_file(df)

    def replace_only_specified_columns(): # Replace Only For Specified Columns

        print("\nReplace Only For Specified Columns - To only replace empty
values for one column, specify the column name for the DataFrame.")

        value = input("Enter the value with which you want to replace the
empty cell in the column 'X2. The age of house in years': ")

        df['X2 house age'].fillna(value, inplace = True)
        print_csv_file(df)

    def replace_using_mean(): # Replace Using Mean
        print("\nMean - The average value (the sum of all values divided by
the number of values).")
        mean = df['X2 house age'].mean()
        df['X2 house age'].fillna(mean, inplace = True)
        print_csv_file(df)

    def replace_using_median(): # Replace Using Median
        print("\nMedian - The value in the middle, after you have sorted all
values ascending.")
```

```
        median = df['X2 house age'].median()
        df['X2 house age'].fillna(median, inplace = True)
        print_csv_file(df)

    while True:  # This simulates a Do Loop

        print("\n")
        heading = "CLEAN EMPTY CELLS - MENU"
        print('{:s}'.format('\u0332'.join(heading.center(100))))

        choice = input(
            "   1. Remove Rows\n   2. Replace Empty Values\n   3. Replace Only
For Specified Columns\n   4. Replace Using Mean\n   5. Replace Using
Median\n   6. Exit\nEnter the number corresponding to the menu to implement
the choice: ") # Menu Driven Implementation

        # str() returns the string version of the variable "choice"
        if choice == str(1):
            remove_rows() # Remove Rows
        elif choice == str(2):
            replace_empty_values() # Replace Empty Values
        elif choice == str(3):
            replace_only_specified_columns() # Replace Only For Specified
Columns
        elif choice == str(4):
            replace_using_mean() # Replace Using Mean
        elif choice == str(5):
            replace_using_median() # Replace Using Median
        elif choice == str(6):
            break  # Exit loop
        else:
            print("Error: Invalid Input! Please try again.")

def clean_data_wrong_format(): # Clean Data of Wrong Format

    print("\nIn our Data Frame, we have two cells with the wrong format. Check
out rows 8 and 14, the 'X1 transaction date' column should be a string that
represents a date.")

    def convert_into_correct_format(): # Convert Into a Correct Format

        df['X1 transaction date'] = pd.to_datetime(df['X1 transaction date'])
        print_csv_file(df)

    def remove_rows(): # Remove Rows - The result from the converting in the
example above gave us a NaT value, which can be handled as a NULL value, and
we can remove the row by using the dropna() method.
```

```
        df.dropna(subset=['X1 transaction date'], inplace = True)
        print_csv_file(df)

    while True:  # This simulates a Do Loop

        print("\n")
        heading = "CLEAN DATA OF WRONG FORMAT - MENU"
        print('{:s}'.format('\u0332'.join(heading.center(100))))

        choice = input(
            "   1. Convert Into a Correct Format\n   2. Remove Rows\n   3.
Exit\nEnter the number corresponding to the menu to implement the choice: ") #
Menu Driven Implementation

        # str() returns the string version of the variable "choice".
        if choice == str(1):

            print("\nLet's try to convert all cells in the 'X1. Transaction
Date' column into dates.")
            convert_into_correct_format() # Convert Into a Correct
Format

        elif choice == str(2):

            print("\nAs you can see from the result, the date in row 14 was
fixed, but the empty date in row 8 got a NaT (Not a Time) value, in other
words, an empty value. One way to deal with empty values is simply removing
the entire row.")
            remove_rows() # Remove Rows

        elif choice == str(3):
            break  # Exit loop

        else:
            print("Error: Invalid Input! Please try again.")

def fix_wrong_data(): # Fix Wrong Data

    def convert_into_correct_format(): # Convert Into a Correct Format
        df.loc[4, 'X4 number of convenience stores'] = 10
        print_csv_file(df)

    # Compute the qth quantile of the given data (array elements) along the
specified axis.
    def print_five_number_summary_IQR_outlier(minimum, Q1, median, Q3,
maximum):

        print("Minimum = ", minimum)
```

```
        print("Q1 quantile = ", Q1)
        print("Median =", median)
        print("Q3 quantile = ", Q3)
        print("Maximum =", maximum)

        IQR = Q3 - Q1
        print("Inter-Quartile Range (IQR) = ", IQR)
        outlier = 1.5 * IQR
        print("Outlier (1.5 X IQR) = ", outlier)
        df.loc[4, 'X4 number of convenience stores'] = outlier

    def calc_five_number_summary_variance_standard_deviation():

        min_X4 = df['X4 number of convenience stores'].min()
        Q1_X4 = np.quantile(df['X4 number of convenience stores'], .25)
        median_X4 = df['X4 number of convenience stores'].median()
        Q3_X4 = np.quantile(df['X4 number of convenience stores'], .75)
        max_X4 = df['X4 number of convenience stores'].max()
        print_five_number_summary_IQR_outlier(min_X4, Q1_X4, median_X4, Q3_X4,
max_X4)

        var_X4 = df['X4 number of convenience stores'].var()
        print("Variance = ", var_X4)
        std_X4 = df['X4 number of convenience stores'].std()
        print("Standard Deviation = ", std_X4)

    def remove_rows(): # Remove Rows

        max_value = input("\nEnter the value above which the row should be
deleted: ")
        for i in df.index: # Delete rows where "X4 number of convenience
stores" is higher than "max_value"
            if df.loc[i, 'X4 number of convenience stores'] > int(max_value):
                df.drop(i, inplace = True)
        print_csv_file(df)

    while True:  # This simulates a Do Loop

        print("\n")
        heading = "FIX WRONG - MENU"
        print('{:s}'.format('\u0332'.join(heading.center(100))))

        choice = input(
            "   1. Replace Values\n   2. Remove Rows\n   3. Exit\nEnter the
number corresponding to the menu to implement the choice: ") # Menu Driven
Implementation

        # str() returns the string version of the variable "choice"
```

```
        if choice == str(1):

            print("\nIn our example, it is most likely a typo, and the value
should be '10' instead of '100', and we could just insert '10' in row 5.")
            convert_into_correct_format() # Convert Into a Correct Format

            print("\nFor small data sets, you might be able to replace the
wrong data one by one, but not for big data sets. To replace wrong data for
larger data sets you can create some rules, e.g. outliers.")
            calc_five_number_summary_variance_standard_deviation()
            print_csv_file(df)

        elif choice == str(2):

            print("\nRemove Rows - This way you do not have to find out what
to replace them with, and there is a good chance you do not need them to do
your analyses.")
            remove_rows() # Remove Rows

        elif choice == str(3):
            break  # Exit loop
        else:
            print("Error: Invalid Input! Please try again.")

def remove_duplicates(): # Remove Duplicates

    print("\nTo discover duplicates, we can use the duplicated() method. The
duplicated() method returns a Boolean value for each row:")
    print(df.duplicated()) # Returns True for every row that is a duplicate,
otherwise False.

    print("\nTo remove duplicates, use the drop_duplicates() method.")
    df.drop_duplicates(inplace = True) # Remove all duplicates
    # The (inplace = True) will make sure that the method does NOT return a
new DataFrame, but it will remove all duplicates from the original
DataFrame.
    print_csv_file(df)

# Driver Code: main() ; Execution starts here.

df = pd.read_csv("Real Estate Data Set.csv")
print_csv_file(df)

print("\n")
heading = "Identification of Response Variable & Regressor Variables"
print('{:s}'.format('\u0332'.join(heading.center(100))))

print("\nThere are three regressor variables (X1, X2, X4), namely:")
```

```
print("X1 - Transaction Date")
print("X2 - Age of House in Year(s)")
print("X4 - Number of Convenience Stores within Walking Distance")

print("\n")
heading = "Our Data Set"
print('{:s}'.format('\u0332'.join(heading.center(100))))

print("1. The data set contains some empty cells ('X1 transaction date' in row
9, and 'X2 house age' in rows 7 and 10).")
print("2. The data set contains wrong format ('X1 transaction date' in row
15).")
print("3. The data set contains wrong data ('X4 number of convenience stores'
in row 5).")
print("4. The data set contains duplicates (row 3 and 4).")

while True:  # This simulates a Do Loop

    print("\n")
    heading = "MAIN MENU"
    print('{:s}'.format('\u0332'.join(heading.center(100))))

    choice = input(
        "   1. Clean Empty Cells\n   2. Clean Data of Wrong Format\n   3. Fix
Wrong Data\n   4. Remove Duplicates\n   5. Exit\nEnter the number
corresponding to the menu to implement the choice: ") # Menu Driven
Implementation

    # str() returns the string version of the variable "choice"
    if choice == str(1):
        print("Clean Empty Cells - Empty cells can potentially give you a
wrong result when you analyze data.")
        clean_empty_cells() # Clean Empty Cells

    elif choice == str(2):
        print("Clean Data of Wrong Format - Cells with data of the wrong
format can make it difficult, or even impossible, to analyze data. To fix it,
you have two options:-")
        clean_data_wrong_format() # Clean Data of Wrong Format

    elif choice == str(3):
        fix_wrong_data() # Fix Wrong Data

    elif choice == str(4):
        print("Duplicate rows are rows that have been registered more than one
time.")
        remove_duplicates() # Remove Duplicates
```

```
    elif choice == str(5):
        break # Exit loop

    else:
        print("Error: Invalid Input! Please try again.")
```

RESULT:

Data cleaning routines attempted to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data. All the simulation results were verified successfully.

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

Restarting kernel...
```

```
In [1]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning/
Expt_3_Code.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning'
```

```
                                                Read CSV File
```

	X1 transaction date	X2 house age	X4 number of convenience stores
0	17-09-2012	32.0	10
1	17-09-2012	19.5	9
2	03-08-2013	13.3	5
3	03-08-2013	13.3	5
4	03-03-2012	5.0	100
5	07-06-2012	7.1	3
6	07-06-2012	NaN	7
7	17-04-2013	20.3	6
8	NaN	31.7	1
9	17-04-2013	NaN	3
10	03-08-2013	34.8	1
11	13-03-2013	6.3	9
12	17-09-2012	13.0	5
13	07-06-2012	20.4	4
14	20130506	13.2	4

```
                    Identification of Response Variable & Regressor Variables

There are three regressor variables (X1, X2, X4), namely:
X1 - Transaction Date
X2 - Age of House in Year(s)
X4 - Number of Convenience Stores within Walking Distance


                                                Our Data Set
1. The data set contains some empty cells ('X1 transaction date' in row 9, and 'X2 house
age' in row 7 and 10).
2. The data set contains wrong format ('X1 transaction date' in row 15).
3. The data set contains wrong data ('X4 number of convenience stores' in row 5).
4. The data set contains duplicates (row 3 and 4).


                                                MAIN MENU

   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
```

```
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 1
Clean Empty Cells - Empty cells can potentially give you a wrong result when you analyze
data.


_____________________________________CLEAN EMPTY CELLS - MENU

   1. Remove Rows
   2. Replace Empty Values
   3. Replace Only For Specified Columns
   4. Replace Using Mean
   5. Replace Using Median
   6. Exit
Enter the number corresponding to the menu to implement the choice: 1

Remove Rows - This is usually OK, since data sets can be very big, and removing a few rows
will not have a big impact on the result.

Do you want to change the original DataFrame? (y/n): y


_____________________________________________Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
0           17-09-2012          32.0                               10
1           17-09-2012          19.5                                9
2           03-08-2013          13.3                                5
3           03-08-2013          13.3                                5
4           03-03-2012           5.0                              100
5           07-06-2012           7.1                                3
7           17-04-2013          20.3                                6
10          03-08-2013          34.8                                1
11          13-03-2013           6.3                                9
12          17-09-2012          13.0                                5
13          07-06-2012          20.4                                4
14            20130506          13.2                                4


_____________________________________CLEAN EMPTY CELLS - MENU

   1. Remove Rows
   2. Replace Empty Values
   3. Replace Only For Specified Columns
   4. Replace Using Mean
   5. Replace Using Median
   6. Exit
Enter the number corresponding to the menu to implement the choice: 7
Error: Invalid Input! Please try again.


_____________________________________CLEAN EMPTY CELLS - MENU

   1. Remove Rows
   2. Replace Empty Values
```

```
   3. Replace Only For Specified Columns
   4. Replace Using Mean
   5. Replace Using Median
   6. Exit
Enter the number corresponding to the menu to implement the choice: 6


_____________________________________________MAIN MENU

   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 2
Clean Data of Wrong Format - Cells with data of wrong format can make it difficult, or
even impossible, to analyze data. To fix it, you have two options:-

In our Data Frame, we have two cells with the wrong format. Check out row 8 and 14, the
'X1 transaction date' column should be a string that represents a date.


_____________________________CLEAN DATA OF WRONG FORMAT - MENU

   1. Convert Into a Correct Format
   2. Remove Rows
   3. Exit
Enter the number corresponding to the menu to implement the choice: 1

Let's try to convert all cells in the 'X1. Transaction Date' column into dates.


_____________________________________Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
0           2012-09-17          32.0                              10
1           2012-09-17          19.5                               9
2           2013-03-08          13.3                               5
3           2013-03-08          13.3                               5
4           2012-03-03           5.0                             100
5           2012-07-06           7.1                               3
7           2013-04-17          20.3                               6
10          2013-03-08          34.8                               1
11          2013-03-13           6.3                               9
12          2012-09-17          13.0                               5
13          2012-07-06          20.4                               4
14          2013-05-06          13.2                               4


_____________________________CLEAN DATA OF WRONG FORMAT - MENU

   1. Convert Into a Correct Format
   2. Remove Rows
   3. Exit
Enter the number corresponding to the menu to implement the choice: 4
Error: Invalid Input! Please try again.
```

CLEAN DATA OF WRONG FORMAT - MENU

1. Convert Into a Correct Format
2. Remove Rows
3. Exit

Enter the number corresponding to the menu to implement the choice: 3

MAIN MENU

1. Clean Empty Cells
2. Clean Data of Wrong Format
3. Fix Wrong Data
4. Remove Duplicates
5. Exit

Enter the number corresponding to the menu to implement the choice: 3

FIX WRONG - MENU

1. Replace Values
2. Remove Rows
3. Exit

Enter the number corresponding to the menu to implement the choice: 1

In our example, it is most likely a typo, and the value should be '10' instead of '100', and we could just insert '10' in row 5.

Read CSV File

	X1 transaction date	X2 house age	X4 number of convenience stores
0	2012-09-17	32.0	10
1	2012-09-17	19.5	9
2	2013-03-08	13.3	5
3	2013-03-08	13.3	5
4	2012-03-03	5.0	10
5	2012-07-06	7.1	3
7	2013-04-17	20.3	6
10	2013-03-08	34.8	1
11	2013-03-13	6.3	9
12	2012-09-17	13.0	5
13	2012-07-06	20.4	4
14	2013-05-06	13.2	4

For small data sets, you might be able to replace the wrong data one by one, but not for big data sets. To replace wrong data for larger data sets you can create some rules, e.g. outliers.

Minimum = 1
Q1 quantile = 4.0
Median = 5.0
Q3 quantile = 9.0
Maximum = 10
Inter-Quartile Range (IQR) = 5.0

```
Outlier (1.5 X IQR) =  7.5
Variance =  7.2935606060606055
Standard Deviation =  2.700659290999256


_____________________________________________Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
0           2012-09-17          32.0                              10.0
1           2012-09-17          19.5                               9.0
2           2013-03-08          13.3                               5.0
3           2013-03-08          13.3                               5.0
4           2012-03-03           5.0                               7.5
5           2012-07-06           7.1                               3.0
7           2013-04-17          20.3                               6.0
10          2013-03-08          34.8                               1.0
11          2013-03-13           6.3                               9.0
12          2012-09-17          13.0                               5.0
13          2012-07-06          20.4                               4.0
14          2013-05-06          13.2                               4.0


_____________________________________________FIX WRONG - MENU

   1. Replace Values
   2. Remove Rows
   3. Exit
Enter the number corresponding to the menu to implement the choice: 4
Error: Invalid Input! Please try again.


_____________________________________________FIX WRONG - MENU

   1. Replace Values
   2. Remove Rows
   3. Exit
Enter the number corresponding to the menu to implement the choice: 3


_____________________________________________MAIN MENU

   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 4
Duplicate rows are rows that have been registered more than one time.

To discover duplicates, we can use the duplicated() method. The duplicated() method
returns a Boolean value for each row:
0     False
1     False
2     False
3      True
4     False
```

```
5     False
7     False
10    False
11    False
12    False
13    False
14    False
dtype: bool

To remove duplicates, use the drop_duplicates() method.


                                          Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
0           2012-09-17          32.0                              10.0
1           2012-09-17          19.5                               9.0
2           2013-03-08          13.3                               5.0
4           2012-03-03           5.0                               7.5
5           2012-07-06           7.1                               3.0
7           2013-04-17          20.3                               6.0
10          2013-03-08          34.8                               1.0
11          2013-03-13           6.3                               9.0
12          2012-09-17          13.0                               5.0
13          2012-07-06          20.4                               4.0
14          2013-05-06          13.2                               4.0


                                          MAIN MENU

   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 6
Error: Invalid Input! Please try again.


                                          MAIN MENU

   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 5

In [2]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning/
Expt_3_Code.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning'


                                          Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
```

```
0           17-09-2012          32.0                          10
1           17-09-2012          19.5                           9
2           03-08-2013          13.3                           5
3           03-08-2013          13.3                           5
4           03-03-2012           5.0                         100
5           07-06-2012           7.1                           3
6           07-06-2012           NaN                           7
7           17-04-2013          20.3                           6
8                  NaN          31.7                           1
9           17-04-2013           NaN                           3
10          03-08-2013          34.8                           1
11          13-03-2013           6.3                           9
12          17-09-2012          13.0                           5
13          07-06-2012          20.4                           4
14            20130506          13.2                           4
```

Identification of Response Variable & Regressor Variables

There are three regressor variables (X1, X2, X4), namely:
X1 - Transaction Date
X2 - Age of House in Year(s)
X4 - Number of Convenience Stores within Walking Distance

Our Data Set

1. The data set contains some empty cells ('X1 transaction date' in row 9, and 'X2 house age' in row 7 and 10).
2. The data set contains wrong format ('X1 transaction date' in row 15).
3. The data set contains wrong data ('X4 number of convenience stores' in row 5).
4. The data set contains duplicates (row 3 and 4).

MAIN MENU

1. Clean Empty Cells
2. Clean Data of Wrong Format
3. Fix Wrong Data
4. Remove Duplicates
5. Exit

Enter the number corresponding to the menu to implement the choice: 1
Clean Empty Cells - Empty cells can potentially give you a wrong result when you analyze data.

CLEAN EMPTY CELLS - MENU

1. Remove Rows
2. Replace Empty Values
3. Replace Only For Specified Columns
4. Replace Using Mean
5. Replace Using Median
6. Exit

Enter the number corresponding to the menu to implement the choice: 1

Remove Rows - This is usually OK, since data sets can be very big, and removing a few rows will not have a big impact on the result.

Do you want to change the original DataFrame? (y/n): n

Read CSV File

	X1 transaction date	X2 house age	X4 number of convenience stores
0	17-09-2012	32.0	10
1	17-09-2012	19.5	9
2	03-08-2013	13.3	5
3	03-08-2013	13.3	5
4	03-03-2012	5.0	100
5	07-06-2012	7.1	3
7	17-04-2013	20.3	6
10	03-08-2013	34.8	1
11	13-03-2013	6.3	9
12	17-09-2012	13.0	5
13	07-06-2012	20.4	4
14	20130506	13.2	4

CLEAN EMPTY CELLS - MENU

1. Remove Rows
2. Replace Empty Values
3. Replace Only For Specified Columns
4. Replace Using Mean
5. Replace Using Median
6. Exit

Enter the number corresponding to the menu to implement the choice: 6

MAIN MENU

1. Clean Empty Cells
2. Clean Data of Wrong Format
3. Fix Wrong Data
4. Remove Duplicates
5. Exit

Enter the number corresponding to the menu to implement the choice: 2
Clean Data of Wrong Format - Cells with data of wrong format can make it difficult, or even impossible, to analyze data. To fix it, you have two options:-

In our Data Frame, we have two cells with the wrong format. Check out row 8 and 14, the 'X1 transaction date' column should be a string that represents a date.

CLEAN DATA OF WRONG FORMAT - MENU

1. Convert Into a Correct Format
2. Remove Rows
3. Exit

Enter the number corresponding to the menu to implement the choice: 2

As you can see from the result, the date in row 14 was fixed, but the empty date in row 8 got a NaT (Not a Time) value, in other words, an empty value. One way to deal with empty values is simply removing the entire row.

```
                                                Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
0           17-09-2012          32.0                                10
1           17-09-2012          19.5                                 9
2           03-08-2013          13.3                                 5
3           03-08-2013          13.3                                 5
4           03-03-2012           5.0                               100
5           07-06-2012           7.1                                 3
6           07-06-2012           NaN                                 7
7           17-04-2013          20.3                                 6
9           17-04-2013           NaN                                 3
10          03-08-2013          34.8                                 1
11          13-03-2013           6.3                                 9
12          17-09-2012          13.0                                 5
13          07-06-2012          20.4                                 4
14            20130506          13.2                                 4
```

```
                                    CLEAN DATA OF WRONG FORMAT - MENU

   1. Convert Into a Correct Format
   2. Remove Rows
   3. Exit
Enter the number corresponding to the menu to implement the choice: 3
```

```
                                                MAIN MENU

   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 3
```

```
                                            FIX WRONG - MENU

   1. Replace Values
   2. Remove Rows
   3. Exit
Enter the number corresponding to the menu to implement the choice:
```

```
2           03-08-2013          13.3                                5
3           03-08-2013          13.3                                5
4           03-03-2012           5.0                              100
5           07-06-2012           7.1                                3
6           07-06-2012           NaN                                7
7           17-04-2013          20.3                                6
9           17-04-2013           NaN                                3
10          03-08-2013          34.8                                1
11          13-03-2013           6.3                                9
12          17-09-2012          13.0                                5
13          07-06-2012          20.4                                4
14            20130506          13.2                                4
```

CLEAN DATA OF WRONG FORMAT - MENU

```
   1. Convert Into a Correct Format
   2. Remove Rows
   3. Exit
Enter the number corresponding to the menu to implement the choice: 3
```

MAIN MENU

```
   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 3
```

FIX WRONG - MENU

```
   1. Replace Values
   2. Remove Rows
   3. Exit
Enter the number corresponding to the menu to implement the choice: 2

Remove Rows - This way you do not have to find out what to replace them with, and there is
a good chance you do not need them to do your analyses.


Enter the value above which the row should be deleted: 10
```

Read CSV File

```
   X1 transaction date  X2 house age  X4 number of convenience stores
0           17-09-2012          32.0                               10
1           17-09-2012          19.5                                9
2           03-08-2013          13.3                                5
3           03-08-2013          13.3                                5
5           07-06-2012           7.1                                3
6           07-06-2012           NaN                                7
7           17-04-2013          20.3                                6
```

```
9           17-04-2013          NaN                                3
10          03-08-2013         34.8                                1
11          13-03-2013          6.3                                9
12          17-09-2012         13.0                                5
13          07-06-2012         20.4                                4
14            20130506         13.2                                4
```

FIX WRONG - MENU

```
   1. Replace Values
   2. Remove Rows
   3. Exit
Enter the number corresponding to the menu to implement the choice: 3
```

MAIN MENU

```
   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 5
```

```
In [3]:          'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning/
Expt_3_Code.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning'
```

Read CSV File

	X1 transaction date	X2 house age	X4 number of convenience stores
0	17-09-2012	32.0	10
1	17-09-2012	19.5	9
2	03-08-2013	13.3	5
3	03-08-2013	13.3	5
4	03-03-2012	5.0	100
5	07-06-2012	7.1	3
6	07-06-2012	NaN	7
7	17-04-2013	20.3	6
8	NaN	31.7	1
9	17-04-2013	NaN	3
10	03-08-2013	34.8	1
11	13-03-2013	6.3	9
12	17-09-2012	13.0	5
13	07-06-2012	20.4	4
14	20130506	13.2	4

Identification of Response Variable & Regressor Variables

There are three regressor variables (X1, X2, X4), namely:
X1 - Transaction Date
X2 - Age of House in Year(s)

X4 - Number of Convenience Stores within Walking Distance

Our Data Set

1. The data set contains some empty cells ('X1 transaction date' in row 9, and 'X2 house age' in row 7 and 10).
2. The data set contains wrong format ('X1 transaction date' in row 15).
3. The data set contains wrong data ('X4 number of convenience stores' in row 5).
4. The data set contains duplicates (row 3 and 4).

MAIN MENU

1. Clean Empty Cells
2. Clean Data of Wrong Format
3. Fix Wrong Data
4. Remove Duplicates
5. Exit

Enter the number corresponding to the menu to implement the choice: 1
Clean Empty Cells - Empty cells can potentially give you a wrong result when you analyze data.

CLEAN EMPTY CELLS - MENU

1. Remove Rows
2. Replace Empty Values
3. Replace Only For Specified Columns
4. Replace Using Mean
5. Replace Using Median
6. Exit

Enter the number corresponding to the menu to implement the choice: 1

Remove Rows - This is usually OK, since data sets can be very big, and removing a few rows will not have a big impact on the result.

Do you want to change the original DataFrame? (y/n): s
Error: Invalid Input! Please try again.

CLEAN EMPTY CELLS - MENU

1. Remove Rows
2. Replace Empty Values
3. Replace Only For Specified Columns
4. Replace Using Mean
5. Replace Using Median
6. Exit

Enter the number corresponding to the menu to implement the choice: 2

Replace Empty Values - This way you do not have to delete entire rows just because of some empty cells. The fillna() method allows us to replace empty cells with a value.

Enter the value with which you want to replace the empty cell: 7

```
                                                Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
0           17-09-2012         32.0                               10
1           17-09-2012         19.5                                9
2           03-08-2013         13.3                                5
3           03-08-2013         13.3                                5
4           03-03-2012          5.0                              100
5           07-06-2012          7.1                                3
6           07-06-2012            7                                7
7           17-04-2013         20.3                                6
8                    7         31.7                                1
9           17-04-2013            7                                3
10          03-08-2013         34.8                                1
11          13-03-2013          6.3                                9
12          17-09-2012         13.0                                5
13          07-06-2012         20.4                                4
14            20130506         13.2                                4


                                          CLEAN EMPTY CELLS - MENU

   1. Remove Rows
   2. Replace Empty Values
   3. Replace Only For Specified Columns
   4. Replace Using Mean
   5. Replace Using Median
   6. Exit
Enter the number corresponding to the menu to implement the choice: 6


                                                MAIN MENU

   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 5

In [4]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning/
Expt_3_Code.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning'


                                                Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
0           17-09-2012         32.0                               10
1           17-09-2012         19.5                                9
2           03-08-2013         13.3                                5
3           03-08-2013         13.3                                5
4           03-03-2012          5.0                              100
5           07-06-2012          7.1                                3
6           07-06-2012          NaN                                7
```

```
7           17-04-2013          20.3                            6
8                  NaN          31.7                            1
9           17-04-2013           NaN                            3
10          03-08-2013          34.8                            1
11          13-03-2013           6.3                            9
12          17-09-2012          13.0                            5
13          07-06-2012          20.4                            4
14            20130506          13.2                            4
```

Identification of Response Variable & Regressor Variables

There are three regressor variables (X1, X2, X4), namely:
X1 - Transaction Date
X2 - Age of House in Year(s)
X4 - Number of Convenience Stores within Walking Distance

Our Data Set

1. The data set contains some empty cells ('X1 transaction date' in row 9, and 'X2 house age' in row 7 and 10).
2. The data set contains wrong format ('X1 transaction date' in row 15).
3. The data set contains wrong data ('X4 number of convenience stores' in row 5).
4. The data set contains duplicates (row 3 and 4).

MAIN MENU

1. Clean Empty Cells
2. Clean Data of Wrong Format
3. Fix Wrong Data
4. Remove Duplicates
5. Exit

```
Enter the number corresponding to the menu to implement the choice: 1
```

Clean Empty Cells - Empty cells can potentially give you a wrong result when you analyze data.

CLEAN EMPTY CELLS - MENU

1. Remove Rows
2. Replace Empty Values
3. Replace Only For Specified Columns
4. Replace Using Mean
5. Replace Using Median
6. Exit

```
Enter the number corresponding to the menu to implement the choice: 3
```

Replace Only For Specified Columns - To only replace empty values for one column, specify the column name for the DataFrame.

```
Enter the value with which you want to replace the empty cell in the column 'X2. The age of house in years': 5
```

```
                                        Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
0           17-09-2012         32.0                               10
1           17-09-2012         19.5                                9
2           03-08-2013         13.3                                5
3           03-08-2013         13.3                                5
4           03-03-2012          5.0                              100
5           07-06-2012          7.1                                3
6           07-06-2012            5                                7
7           17-04-2013         20.3                                6
8                  NaN         31.7                                1
9           17-04-2013            5                                3
10          03-08-2013         34.8                                1
11          13-03-2013          6.3                                9
12          17-09-2012         13.0                                5
13          07-06-2012         20.4                                4
14            20130506         13.2                                4
```

```
                                    CLEAN EMPTY CELLS - MENU

   1. Remove Rows
   2. Replace Empty Values
   3. Replace Only For Specified Columns
   4. Replace Using Mean
   5. Replace Using Median
   6. Exit
Enter the number corresponding to the menu to implement the choice: 6
```

```
                                        MAIN MENU

   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 5
```

```
In [5]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning/
Expt_3_Code.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning'
```

```
                                        Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
0           17-09-2012         32.0                               10
1           17-09-2012         19.5                                9
2           03-08-2013         13.3                                5
3           03-08-2013         13.3                                5
4           03-03-2012          5.0                              100
5           07-06-2012          7.1                                3
6           07-06-2012          NaN                                7
7           17-04-2013         20.3                                6
```

```
8                 NaN         31.7                                1
9          17-04-2013          NaN                                3
10         03-08-2013         34.8                                1
11         13-03-2013          6.3                                9
12         17-09-2012         13.0                                5
13         07-06-2012         20.4                                4
14           20130506         13.2                                4
```

Identification of Response Variable & Regressor Variables

There are three regressor variables (X1, X2, X4), namely:
X1 - Transaction Date
X2 - Age of House in Year(s)
X4 - Number of Convenience Stores within Walking Distance

Our Data Set

1. The data set contains some empty cells ('X1 transaction date' in row 9, and 'X2 house age' in row 7 and 10).
2. The data set contains wrong format ('X1 transaction date' in row 15).
3. The data set contains wrong data ('X4 number of convenience stores' in row 5).
4. The data set contains duplicates (row 3 and 4).

MAIN MENU

```
   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 1
```

Clean Empty Cells - Empty cells can potentially give you a wrong result when you analyze data.

CLEAN EMPTY CELLS - MENU

```
   1. Remove Rows
   2. Replace Empty Values
   3. Replace Only For Specified Columns
   4. Replace Using Mean
   5. Replace Using Median
   6. Exit
Enter the number corresponding to the menu to implement the choice: 4
```

Mean - The average value (the sum of all values divided by number of values).

Read CSV File

```
   X1 transaction date  X2 house age  X4 number of convenience stores
0           17-09-2012     32.000000                                10
1           17-09-2012     19.500000                                 9
2           03-08-2013     13.300000                                 5
```

```
3           03-08-2013     13.300000                                5
4           03-03-2012      5.000000                              100
5           07-06-2012      7.100000                                3
6           07-06-2012     17.684615                                7
7           17-04-2013     20.300000                                6
8                  NaN     31.700000                                1
9           17-04-2013     17.684615                                3
10          03-08-2013     34.800000                                1
11          13-03-2013      6.300000                                9
12          17-09-2012     13.000000                                5
13          07-06-2012     20.400000                                4
14            20130506     13.200000                                4
```

```
_____________________________________CLEAN EMPTY CELLS - MENU

   1. Remove Rows
   2. Replace Empty Values
   3. Replace Only For Specified Columns
   4. Replace Using Mean
   5. Replace Using Median
   6. Exit
Enter the number corresponding to the menu to implement the choice: 6
```

```
_____________________________________________MAIN MENU

   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 5
```

```
In [6]:          'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning/
Expt_3_Code.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 3 - Data Cleaning'
```

```
_____________________________________Read CSV File
   X1 transaction date  X2 house age  X4 number of convenience stores
0           17-09-2012          32.0                                10
1           17-09-2012          19.5                                 9
2           03-08-2013          13.3                                 5
3           03-08-2013          13.3                                 5
4           03-03-2012           5.0                               100
5           07-06-2012           7.1                                 3
6           07-06-2012           NaN                                 7
7           17-04-2013          20.3                                 6
8                  NaN          31.7                                 1
9           17-04-2013           NaN                                 3
10          03-08-2013          34.8                                 1
11          13-03-2013           6.3                                 9
12          17-09-2012          13.0                                 5
```

```
13          07-06-2012          20.4                                4
14            20130506          13.2                                4
```

Identification of Response Variable & Regressor Variables

There are three regressor variables (X1, X2, X4), namely:
X1 - Transaction Date
X2 - Age of House in Year(s)
X4 - Number of Convenience Stores within Walking Distance

Our Data Set

1. The data set contains some empty cells ('X1 transaction date' in row 9, and 'X2 house age' in row 7 and 10).
2. The data set contains wrong format ('X1 transaction date' in row 15).
3. The data set contains wrong data ('X4 number of convenience stores' in row 5).
4. The data set contains duplicates (row 3 and 4).

MAIN MENU

1. Clean Empty Cells
2. Clean Data of Wrong Format
3. Fix Wrong Data
4. Remove Duplicates
5. Exit

```
Enter the number corresponding to the menu to implement the choice: 1
Clean Empty Cells - Empty cells can potentially give you a wrong result when you analyze
data.
```

CLEAN EMPTY CELLS - MENU

1. Remove Rows
2. Replace Empty Values
3. Replace Only For Specified Columns
4. Replace Using Mean
5. Replace Using Median
6. Exit

```
Enter the number corresponding to the menu to implement the choice: 5
```

Median - The value in the middle, after you have sorted all values ascending.

Read CSV File

```
   X1 transaction date  X2 house age  X4 number of convenience stores
0           17-09-2012          32.0                                10
1           17-09-2012          19.5                                 9
2           03-08-2013          13.3                                 5
3           03-08-2013          13.3                                 5
4           03-03-2012           5.0                               100
5           07-06-2012           7.1                                 3
6           07-06-2012          13.3                                 7
7           17-04-2013          20.3                                 6
```

```
8                NaN         31.7                                1
9          17-04-2013         13.3                                3
10         03-08-2013         34.8                                1
11         13-03-2013          6.3                                9
12         17-09-2012         13.0                                5
13         07-06-2012         20.4                                4
14           20130506         13.2                                4


 ____________________________________CLEAN EMPTY CELLS - MENU

   1. Remove Rows
   2. Replace Empty Values
   3. Replace Only For Specified Columns
   4. Replace Using Mean
   5. Replace Using Median
   6. Exit
Enter the number corresponding to the menu to implement the choice: 6


 ____________________________________________MAIN MENU

   1. Clean Empty Cells
   2. Clean Data of Wrong Format
   3. Fix Wrong Data
   4. Remove Duplicates
   5. Exit
Enter the number corresponding to the menu to implement the choice: 5

In [7]:
```

EXPERIMENT 4 – DATA REDUCTION

AIM: Perform dimensionality reduction to obtain a reduced or "compressed" representation of the original data using principal components analysis.

SOFTWARE REQUIRED:

Spyder IDE 5.1.5

Anaconda3 2021.11 (Python 3.9.7 64-bit)

Anaconda Inc., 2021.11

DATA SET: Real Estate Data Set

PYTHON CODE:

```
import matplotlib.pyplot as plt # Provides an implicit way of plotting
import numpy as np # Support for large, multi-dimensional arrays and matrices
from numpy.linalg import eig # Compute the eigenvalues and right eigenvectors
of a square array
import pandas as pd  # Library for working with data sets
import seaborn as sns # Provides high-level API to visualize data

"""
There is a Unicode character '\u0332', COMBINING LOW LINE*, which acts as an
underline on the character that precedes it in a string. The center() method
will center align the string, using a specified character (space is the
default) as the fill character.
"""

def print_csv_file(df, heading):
    print("\n")
    print('{:s}'.format('\u0332'.join(heading.center(100))))
    print(df)

def standardize_the_data_set():

    print("\nMean:")

    mean_X2 = df['X2'].mean()
    mean_X3 = df['X3'].mean()
    mean_X4 = df['X4'].mean()
    mean_Y = df['Y'].mean()
```

```
    print("X2. The age of house in years: " + str(mean_X2))
    print("X3. The distance to nearest MRT station in meters: " +
str(mean_X3))
    print("X4. The number of convenience stores within walking distance: " +
str(mean_X4))
    print("Y. House price per local unit area: " + str(mean_Y))

    print("\nStandard Deviation:")

    std_X2 = df['X2'].std()
    std_X3 = df['X3'].std()
    std_X4 = df['X4'].std()
    std_Y = df['Y'].std()

    print("X2. The age of house in years: " + str(std_X2))
    print("X3. The distance to nearest MRT station in meters: " + str(std_X3))
    print("X4. The number of convenience stores within walking distance: " +
str(std_X4))
    print("Y. House price per local unit area: " + str(std_Y))

    heading = "Step 1 - Standardize the dataset (Z-Score Normalization)"
    for i in df.index:
        df.loc[i, 'X2'] = ((df.loc[i, 'X2']) - mean_X2)/std_X2
        df.loc[i, 'X3'] = ((df.loc[i, 'X3']) - mean_X3)/std_X3
        df.loc[i, 'X4'] = ((df.loc[i, 'X4']) - mean_X4)/std_X4
        df.loc[i, 'Y'] = ((df.loc[i, 'Y']) - mean_Y)/std_Y
    print_csv_file(df, heading)

    standardized_dataset = df
    return standardized_dataset

# Calculate the covariance matrix for the whole dataset:-
def covariance_matrix():

    size = df.shape[1] # Get the shape of the DataFrame, which is a tuple
where the first element is the number of rows and the second is the number of
columns.
    covariance_matrix_1D = [0] * (size * size)

    sum = 0
    for i in df.index:
        sum = sum + df.loc[i, 'X2'] * df.loc[i, 'X2']
    covariance_matrix_1D[0] = sum / len(df)  # var(f1)
    sum = 0
    for i in df.index:
        sum = sum + df.loc[i, 'X2'] * df.loc[i, 'X3']
    covariance_matrix_1D[1] = sum / len(df)  # cov(f1,f2)
    sum = 0
```

```
for i in df.index:
    sum = sum + df.loc[i, 'X2'] * df.loc[i, 'X4']
covariance_matrix_1D[2] = sum / len(df)  # cov(f1,f3)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X2'] * df.loc[i, 'Y']
covariance_matrix_1D[3] = sum / len(df)  # cov(f1,f4)

sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X3'] * df.loc[i, 'X2']
covariance_matrix_1D[4] = sum / len(df)  # cov(f2,f1)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X3'] * df.loc[i, 'X3']
covariance_matrix_1D[5] = sum / len(df)  # var(f2)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X3'] * df.loc[i, 'X4']
covariance_matrix_1D[6] = sum / len(df)  # cov(f2,f3)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X3'] * df.loc[i, 'Y']
covariance_matrix_1D[7] = sum / len(df)  # cov(f2,f4)

sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X4'] * df.loc[i, 'X2']
covariance_matrix_1D[8] = sum / len(df)  # cov(f3,f1)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X4'] * df.loc[i, 'X3']
covariance_matrix_1D[9] = sum / len(df)  # cov(f3,f2)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X4'] * df.loc[i, 'X4']
covariance_matrix_1D[10] = sum / len(df)  # var(f3)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X4'] * df.loc[i, 'Y']
covariance_matrix_1D[11] = sum / len(df)  # cov(f3,f4)

sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'Y'] * df.loc[i, 'X2']
covariance_matrix_1D[12] = sum / len(df)  # cov(f4,f1)
sum = 0
for i in df.index:
```

```
        sum = sum + df.loc[i, 'Y'] * df.loc[i, 'X3']
    covariance_matrix_1D[13] = sum / len(df)  # cov(f4,f2)
    sum = 0
    for i in df.index:
        sum = sum + df.loc[i, 'Y'] * df.loc[i, 'X4']
    covariance_matrix_1D[14] = sum / len(df)  # cov(f4,f3)
    sum = 0
    for i in df.index:
        sum = sum + df.loc[i, 'Y'] * df.loc[i, 'Y']
    covariance_matrix_1D[15] = sum / len(df)  # var(f4)

    covariance_matrix_multidim = np.reshape(covariance_matrix_1D, (size,
size))
    print("\nThe covariance matrix is as follows:\n{}".format(
        covariance_matrix_multidim))
    return covariance_matrix_multidim

# Determine explained variance:-
def determine_explained_variance(eigenvalues):

    total_eigenvalues = sum(eigenvalues)
    explained_variance = [(i/total_eigenvalues) for i in sorted(eigenvalues,
reverse=True)]
    cumulative_explained_variance = np.cumsum(explained_variance) # Return the
cumulative sum of the elements along a given axis

    plt.bar(range(0,len(explained_variance)), explained_variance, alpha=0.5,
align='center', label='Individual Explained Variance')
    plt.step(range(0,len(cumulative_explained_variance)),
cumulative_explained_variance, where='mid',label='Cumulative Explained
Variance')

    plt.xlabel('Principal Component Index')
    plt.ylabel('Explained Variance Ratio')
    plt.legend(loc='best')
    plt.grid(True)
    plt.tight_layout()
    plt.show()

# Convert CSV data into multidimensional array:-
def convert_csv_array():
    size = df.shape[1]
    single_dimensional_array = []
    for i in df.index:
        single_dimensional_array.append(df.loc[i, 'X2'])
        single_dimensional_array.append(df.loc[i, 'X3'])
        single_dimensional_array.append(df.loc[i, 'X4'])
        single_dimensional_array.append(df.loc[i, 'Y'])
```

```
    multi_dimensional_array = np.reshape(single_dimensional_array, (len(df),
size))
    return multi_dimensional_array

# Driver Code: main() ; Execution starts here.

print("\nThere are three regressor variables and one response variable
(namely, y):")
print("X2 - Age of House in year(s)")
print("X3 - Distance to Nearest MRT station in meter(s)")
print("X4 - Number of Convenience Stores within Walking Distance")
print("Y - House Price per Local Unit Area")

heading = "Original Data Set"
df = pd.read_csv("Real Estate Data Set.csv")
print_csv_file(df, heading)

# Step 1 - Standardize the dataset:-
standardized_dataset = standardize_the_data_set()
print("\nSince we have standardized the dataset, so the mean for each feature
is 0 and the standard deviation is 1.\n")

heading = "Step 2 - Calculate the covariance matrix for the features in the
dataset"
print('{:s}'.format('\u0332'.join(heading.center(100))))
covariance_matrix_multidim = covariance_matrix()

print("\nNow, we will check the co-relation between our scaled dataset using a
heat map.  The correlation between various features is given by the corr()
function and then the heat map is plotted by the heatmap() function. The
colour scale on the side of the heatmap helps determine the magnitude of the
co-relation.")

print("\nIn our example, we can see that a darker shade represents less co-
relation while a lighter shade represents more co-relation. The diagonal of
the heatmap represents the co-relation of a feature with itself – which is
always 1.0, thus, the diagonal of the heatmap is of the highest shade.")

sns.heatmap(standardized_dataset.corr())
plt.title("Co-relation between features before Principal Component Analysis
(PCA)")
plt.grid(True)
plt.tight_layout()
plt.show()

heading = "Step 3 - Calculate and sort the eigenvalues and eigenvectors for
the covariance matrix"
print("\n")
```

```
print('{:s}'.format('\u0332'.join(heading.center(100))))
eigenvalues, eigen_vectors = eig(covariance_matrix_multidim)

index = eigenvalues.argsort()[::-1] # Returns the indices that would sort an
array
eigenvalues = eigenvalues[index]
eigen_vectors = eigen_vectors[:, index]

print("\nSorted Eigenvalues: ", eigenvalues)
print("\nCorresponding Eigen Vectors:\n{}".format(eigen_vectors))
determine_explained_variance(eigenvalues)

heading = "Step 4 - Pick k eigenvalues and form a matrix of eigenvectors"
print("\n")
print('{:s}'.format('\u0332'.join(heading.center(100))))

eigenvectors_1D = eigen_vectors.flatten() # Return a copy of the array
(matrix) collapsed into one dimension.

while True:
    top = int(input("Enter the number of top eigen vectors to transform the
original matrix: "))
    if top > df.shape[1]:
        print("Invalid Input! Please try again.")
    else:
        break

top_eigenvectors_1D = []  # NULL Single Dimensional Array Declaration
for i in range(int(len(eigenvectors_1D)/len(eigen_vectors)) * top):
    top_eigenvectors_1D.append(eigenvectors_1D[i])
top_eigenvectors_multidim = np.reshape(top_eigenvectors_1D,
(len(eigen_vectors), top))
print("\nIf we choose the top " + str(top) + " eigenvectors, the matrix will
look like this:\n{}".format(top_eigenvectors_multidim))

heading = "Step 5 - Tranform the Original Matrix"
print("\n")
print('{:s}'.format('\u0332'.join(heading.center(100))))

feature_matrix = convert_csv_array()
transformed_data = np.dot(feature_matrix, top_eigenvectors_multidim) # Matrix
Multiplication

print("\nThe transformed data is as follows:\n")
for i in transformed_data:
    print(i)
```

```
print("\nNow that we have applied PCA and obtained the reduced feature set, we
will check the co-relation between various Principal Components, again by
using a heatmap.")

new_dataframe = pd.DataFrame(transformed_data)
sns.heatmap(new_dataframe.corr())
plt.title("Co-relation between features after Principal Component Analysis
(PCA)")
plt.grid(True)
plt.tight_layout()
plt.show()
```

PLOTS:

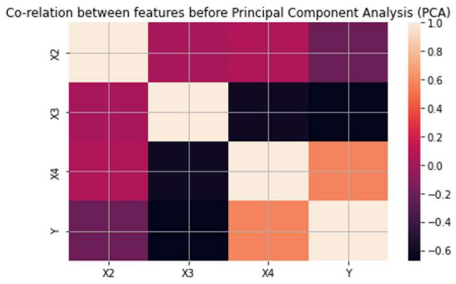


Figure 1. Co-relation between features before Principal Component Analysis (PCA)

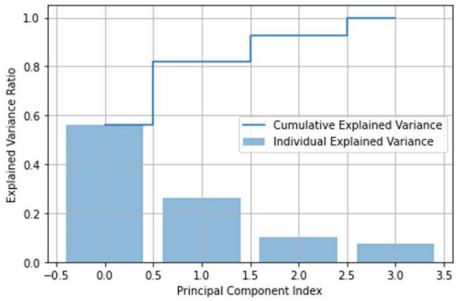


Figure 2. Determine explained variance

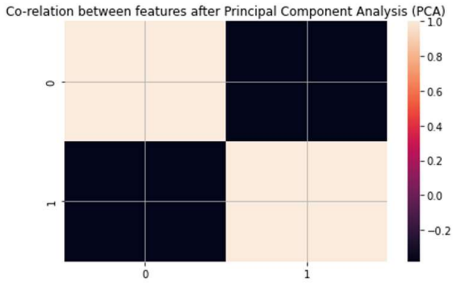


Figure 3. Co-relation between features after Principal Component Analysis (PCA)

RESULT:

Thus, demonstrated dimensionality reduction by reducing the number of random variables or attributes under consideration using principal component analysis, which transforms or projects the original data onto a smaller space. All simulation results were verified successfully.

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

Restarting kernel...
```

```
In [1]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 4 - Principal
Component Analysis/Expt_4_Code.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject
Materials/Semester IV/19CCE213 - Machine Learning and Artificial Intelligence/Lab/
Experiment 4 - Principal Component Analysis'

There are three regressor variables and one response variable (namely, y):
X2 - Age of House in year(s)
X3 - Distance to Nearest MRT station in meter(s)
X4 - Number of Convenience Stores within Walking Distance
Y - House Price per Local Unit Area


                                        Original Data Set
       X2          X3  X4     Y
0    32.0    84.87882  10  37.9
1    19.5   306.59470   9  42.2
2    13.3   561.98450   5  47.3
3    13.3   561.98450   5  54.8
4     5.0   390.56840   5  43.1
..    ...         ...  ..   ...
409  13.7  4082.01500   0  15.4
410   5.6    90.45606   9  50.0
411  18.8   390.96960   7  40.6
412   8.1   104.81010   5  52.5
413   6.5    90.45606   9  63.9

[414 rows x 4 columns]

Mean:
X2. The age of house in years: 17.71256038647343
X3. The distance to nearest MRT station in meters: 1083.8856889130436
X4. The number of convenience stores within walking distance: 4.094202898550725
Y. House price per local unit area: 37.98019323671498

Standard Deviation:
X2. The age of house in years: 11.392484533242536
X3. The distance to nearest MRT station in meters: 1262.1095954078514
X4. The number of convenience stores within walking distance: 2.945561805663617
Y. House price per local unit area: 13.606487697735314


                    Step 1 - Standardize the dataset (Z-Score Normalization)
          X2        X3        X4         Y
0    1.254111 -0.791537  2.004982 -0.005894
1    0.156896 -0.615866  1.665488  0.310132
```

```
2   -0.387322 -0.413515  0.307513  0.684953
3   -0.387322 -0.413515  0.307513  1.236161
4   -1.115873 -0.549332  0.307513  0.376277
..        ...       ...       ...       ...
409 -0.352211  2.375490 -1.389957 -1.659517
410 -1.063206 -0.787118  1.665488  0.883388
411  0.095452 -0.549014  0.986500  0.192541
412 -0.843763 -0.775745  0.307513  1.067124
413 -0.984207 -0.787118  1.665488  1.904959

[414 rows x 4 columns]
```

Since we have standardized the dataset, so the mean for each feature is 0 and the standard deviation is 1.

Step 2 - Calculate the covariance matrix for the features in the dataset

```
The covariance matrix is as follows:
[[ 0.99758454  0.02556016  0.04947272 -0.21005843]
 [ 0.02556016  0.99758454 -0.60106378 -0.67198577]
 [ 0.04947272 -0.60106378  0.99758454  0.56962567]
 [-0.21005843 -0.67198577  0.56962567  0.99758454]]
```

Now, we will check the co-relation between our scaled dataset using a heat map. The correlation between various features is given by the corr() function and then the heat map is plotted by the heatmap() function. The colour scale on the side of the heatmap helps determine the magnitude of the co-relation.

In our example, we can clearly see that a darker shade represents less co-relation while a lighter shade represents more co-relation. The diagonal of the heatmap represents the co-relation of a feature with itself – which is always 1.0, thus, the diagonal of the heatmap is of the highest shade.

Step 3 - Calculate and sort the eigenvalues and eigenvectors for the covariance matrix

```
Sorted Eigenvalues:  [2.23691337 1.04279849 0.41043771 0.3001886 ]

Corresponding Eigen Vectors:
[[ 0.08954605  0.955056   -0.20221334 -0.19738113]
 [ 0.58700983 -0.09294975  0.46288628 -0.65765956]
 [-0.55073343  0.22961701  0.80147479  0.04008594]
 [-0.58659497 -0.16280176 -0.32013157 -0.72589098]]
```

Step 4 - Pick k eigenvalues and form a matrix of eigenvectors

```
Enter the number of top eigen vectors to transform the original matrix:
```

lighter shade represents more co-relation. The diagonal of the heatmap represents the co-relation of a feature with itself – which is always 1.0, thus, the diagonal of the heatmap is of the highest shade.

Step 3 - Calculate and sort the eigenvalues and eigenvectors for the covariance matrix

Sorted Eigenvalues: [2.23691337 1.04279849 0.41043771 0.3001886]

Corresponding Eigen Vectors:
```
[[ 0.08954605  0.955056   -0.20221334 -0.19738113]
 [ 0.58700983 -0.09294975  0.46288628 -0.65765956]
 [-0.55073343  0.22961701  0.80147479  0.04008594]
 [-0.58659497 -0.16280176 -0.32013157 -0.72589098]]
```

Step 4 - Pick k eigenvalues and form a matrix of eigenvectors

Enter the number of top eigen vectors to transform the original matrix: 5
Invalid Input! Please try again.

Enter the number of top eigen vectors to transform the original matrix: 2

If we choose the top 2 eigenvectors, the matrix will look like this:
```
[[ 0.08954605  0.955056  ]
 [-0.20221334 -0.19738113]
 [ 0.58700983 -0.09294975]
 [ 0.46288628 -0.65765956]]
```

Step 5 - Tranform the Original Matrix

The transformed data is as follows:

```
[1.44657583 1.17149427]
[ 1.25979938 -0.08736267]
[ 0.54650337 -0.7673434 ]
[ 0.80164982 -1.12985038]
[ 0.36584657 -1.23333825]
[-0.67633824 -0.74157388]
[0.86372155 1.27550922]
[ 0.82435962 -0.14016339]
[-1.86866245  1.50477925]
[-0.86886282  0.70843689]
[-0.25724691  1.47095995]
[ 1.73158598 -1.9286594 ]
[ 0.28316487 -0.39491023]
[-0.70207758  0.69693689]
[-0.192411   -0.21010549]
[0.23081387 1.04780015]
[ 1.45999214 -2.97382072]
[-0.51902291  0.2392696 ]
[ 1.03361877 -0.28822819]
[ 0.95222839 -1.75477588]
```

```
[-0.80818673 -0.83997287]
[ 1.11466317 -1.22879345]
[-1.13975971  0.44860953]
[ 0.98564688 -1.08348952]
[0.27779542 1.89255059]
[-0.76453376  1.5050247 ]
[ 0.79764262 -2.02474936]
[ 0.10338964 -0.30362225]
[ 0.38410791 -0.22597305]
[ 0.84890484 -1.74345953]
[-1.84228024  1.04579897]
[0.28132757 1.5814241 ]
[-0.4911769   2.06580964]
[ 0.87716574 -0.59003273]
[ 1.28407102 -0.9756436 ]
[-1.68916015 -0.14267043]
[-1.09041153  0.40931808]
[-1.13716831  0.18842885]
[ 0.67666457 -1.67581912]
[ 0.57556131 -0.42842139]
[-2.07975688  0.38278439]
[-1.9738878   0.54229112]
[0.30388051 1.75969053]
[0.47046222 1.61566448]
[ 0.49299656 -1.93895561]
[1.03304894 1.5377278 ]
[1.24507448 0.08212413]
[0.79602852 0.49171102]
[-2.16540054  1.31033372]
[-1.91676249  1.7392897 ]
[0.31570199 0.12596965]
[-1.20576799  1.96644124]
[-0.53534227  1.71590704]
[ 0.27191509 -0.35042689]
[ 0.7618827  -0.70264305]
[-1.54427238  2.46335558]
[1.15077581 1.13061485]
[ 1.15995999 -1.87262555]
[-1.58990485  1.36039652]
[ 0.41600606 -0.49517215]
[-1.17331599  0.1770687 ]
[ 1.27225714 -2.19079055]
[-0.74677287  0.31766871]
[ 0.52963203 -2.00050651]
[-1.23484297  0.73804721]
[1.32316845 1.59822884]
[ 0.82380126 -1.93675667]
[ 0.90520821 -1.55741559]
[0.5182437  1.18642902]
[ 0.35991179 -0.57823749]
[ 1.76456158 -1.94701061]
[0.08868027 1.45869743]
[0.94308155 1.30073665]
[-1.9387048   0.20138033]
[ 1.76261057 -1.78467497]
```

```
[-0.99192827  0.01057458]
[-0.04035429  1.68937365]
[-0.79375773  0.69436986]
[-0.44945085  2.26208642]
[-1.05794792  0.62486618]
[ 0.10185788 -0.51873744]
[0.55564673 1.20447767]
[ 1.03736318 -0.81322173]
[-1.1084599   1.39692788]
[ 0.86538534 -0.47760649]
[ 1.39390861 -2.14282629]
[-1.17483446 -0.763787  ]
[-1.96969984  0.5458409 ]
[-0.59599423 -1.14432204]
[-1.6646079   0.73743132]
[-0.37367971 -1.61921628]
[-0.75700873 -0.89486085]
[-0.76854462  0.81037184]
[-1.45876189  2.36635661]
[0.61230694 1.91260656]
[ 0.73118053 -1.35766032]
[ 1.77999933 -1.98794416]
[-0.17431493  1.16680526]
[ 0.74042706 -0.64365943]
[ 1.87185205 -2.11844667]
[ 0.00612145 -0.00683925]
[-0.68245617  0.06587643]
[ 0.95045952 -2.10721024]
[ 0.64347133 -1.78123297]
[0.35391064 1.66597338]
[ 1.49060971 -3.01732156]
[ 1.22788847 -0.46714895]
[-1.09137091  0.14236073]
[-0.36303447  1.47792523]
[-0.82214611 -0.81829183]
[ 0.72516263 -1.33961026]
[0.89206208 1.28516746]
[-0.94887336 -0.06048123]
[-0.56596282  1.27210586]
[1.55145361 0.36367272]
[-0.0527816  -0.04471449]
[-2.24115184  1.61843276]
[-2.19689215  0.50491669]
[-0.48957524  0.94914646]
[ 1.03456701 -1.04256767]
[0.01336669 0.01691351]
[ 0.58385082 -0.76511906]
[0.01033999 1.52829902]
[-0.55537103 -1.57864008]
[ 1.30726037 -1.55943145]
[ 0.75314626 -1.82687151]
[1.03789641 0.59319367]
[ 0.76232125 -1.90806577]
[1.49174841 0.96906196]
[0.98412691 1.64059807]
```

```
[0.94332919 0.98226584]
[-0.7439012  -0.92946064]
[0.33034484 0.83368963]
[ 0.94616787 -0.06149914]
[0.79036245 1.19127954]
[-1.37908639  1.07080828]
[ 0.54202345 -0.87564954]
[ 0.79046883 -0.74059382]
[-0.66610043 -0.76204334]
[ 0.39124135 -0.55796309]
[ 0.75246285 -0.67975956]
[-0.702354   -0.65835695]
[0.25160584 0.23894531]
[ 0.31509656 -0.38327838]
[-1.03249379  0.03020943]
[ 0.41497783 -1.60194262]
[-0.3274402  -1.90247965]
[ 0.93704706 -1.49926872]
[-1.01608454 -0.74670639]
[1.1168802  1.37592687]
[1.22543746 1.05895425]
[ 1.21708464 -1.44484243]
[-1.01469801  0.0144255 ]
[ 0.50438671 -1.03055895]
[-1.88805297  0.42983867]
[-2.08839072  0.41405109]
[-1.41232108  2.14090599]
[-0.06949003  0.02473853]
[ 0.29185097 -0.50120956]
[-0.01297094 -0.1125624 ]
[ 1.50662176 -2.11095324]
[-0.07636325  0.1540415 ]
[-2.20470476  0.79423157]
[ 0.86098283 -1.49458104]
[-0.22538162 -2.04748244]
[-0.92370365  0.09409772]
[ 1.57906048 -3.14299065]
[1.16595582 0.61185771]
[0.32961926 0.91729531]
[-1.32322971 -0.12058823]
[-2.12044606  1.25744114]
[ 0.89002344 -2.10850239]
[ 1.73394401 -1.90350977]
[0.17791328 2.22622812]
[ 0.82371364 -1.46311992]
[-0.07225505  1.24858638]
[-2.04392411  0.17152406]
[1.37921712 1.0736674 ]
[-0.10828137 -0.33130027]
[-0.38530294 -0.33600964]
[-2.04766644  1.45965019]
[ 1.48126093 -1.36388573]
[-0.91076891  0.21011766]
[-1.95437591  0.5276442 ]
[-1.64014314  0.36888183]
```

```
[-0.72934677  1.92170132]
[-0.78721174  0.72306928]
[-1.74834662 -0.14907529]
[1.27088804 1.1435506 ]
[-1.89957294  0.38920626]
[1.13844033 1.21542954]
[-0.40546161 -0.25129824]
[1.10913238 2.02764406]
[ 0.60875765 -1.14383388]
[-1.56163209 -0.08226374]
[0.14555287 0.02155981]
[-0.36407269  0.61808509]
[1.41056368 0.93141526]
[0.89360874 1.36445129]
[0.66644881 0.32487074]
[-1.22882507  0.71999715]
[ 0.49730401 -0.71644185]
[-0.20600699  2.14262955]
[ 0.5674433  -0.47388722]
[-1.05454597  0.62003275]
[-0.87933684  0.36687657]
[1.529853   0.00901584]
[-0.11616555  2.02539579]
[-1.11048079  0.10301196]
[1.15753036 1.31014064]
[ 0.67699629 -1.65641363]
[-0.43831681 -1.52738118]
[-0.76793382  0.27631832]
[ 1.72739801 -1.93220919]
[-0.90811423  0.78320443]
[ 1.24651459 -0.3781891 ]
[-0.32125492  1.77558046]
[0.67054512 0.77721947]
[ 0.44096881 -0.56211515]
[1.03374513 0.82751292]
[ 2.64626783 -0.32962125]
[-0.91768652 -0.67607302]
[1.74070581 0.48763886]
[-0.78762631 -0.85136001]
[0.90617688 1.07824703]
[ 0.76675407 -1.84620521]
[-2.16242313  0.77573372]
[1.10052387 1.12879139]
[-0.90281079 -1.10117003]
[-1.36879483  2.14916147]
[-0.65204847 -1.05996315]
[-2.09897782  0.65988735]
[-1.77632233  1.35873465]
[1.08089243 2.07553465]
[-0.75560028 -0.30784289]
[ 0.71966011 -0.42862134]
[ 1.99352094 -2.41455036]
[-0.76202402 -0.26026139]
[-0.7091648  -0.35436263]
[-1.05512132  0.60039033]
```

```
[-0.88314977 -0.15125537]
[ 0.7973892 -0.4630576]
[-0.65655562 -1.21668227]
[1.3855799  0.78517276]
[-0.94965263 -0.33549988]
[ 0.26734926 -0.95153829]
[ 0.56980868 -0.18825568]
[-1.28617633  1.18720171]
[-1.32727109  1.00669009]
[-2.23285193  0.41575804]
[0.5821079  2.19844708]
[-1.54068169  2.45870638]
[ 1.54473649 -1.70118722]
[-0.83299959  1.61166572]
[ 0.64087031 -1.97131533]
[-1.9178618   1.55568079]
[-0.91233991  0.50805119]
[-0.31598552 -0.1230243 ]
[ 1.22865935 -2.64514773]
[-1.08888293  0.61035833]
[-0.33632435  0.23237462]
[-0.88486335  0.37580306]
[ 0.80453606 -0.78224401]
[-0.71066768 -0.98617812]
[0.87979217 1.12202986]
[ 0.01241475 -0.12962997]
[-0.81521758  0.62271886]
[0.51648194 1.33872636]
[ 0.35968119 -0.06558374]
[-1.2869822   0.88231384]
[ 2.1674511  -4.19537813]
[0.48901917 0.52945943]
[ 0.32398831 -0.45291135]
[-0.80099642  0.28179583]
[0.8692776  0.69073504]
[ 1.02026744 -1.85144441]
[0.15579553 0.3775056 ]
[-0.71681466  0.65155038]
[ 0.58563813 -1.69906472]
[-0.64626775 -0.97340041]
[ 0.6551986  -1.57015606]
[ 1.21891457 -1.46644228]
[-1.22088978 -0.67935261]
[-0.92922543  1.94953287]
[ 0.54821773 -0.03648105]
[0.5270932  0.29302847]
[ 1.66720679 -1.87519057]
[0.11916006 0.4390565 ]
[ 0.93577915 -0.66382343]
[ 0.49964988 -0.53906699]
[-0.59706434  1.94498604]
[ 1.18979155 -1.9245096 ]
[-0.16948353  0.71278183]
[ 0.80491167 -0.62918565]
[0.57205985 0.77939219]
```

```
[-0.94022287  0.6860923 ]
[-0.2014482  -0.25593045]
[-0.12338099  1.98288033]
[-2.02817495  0.60399652]
[1.72906444 0.86998928]
[-0.02646787 -1.0750443 ]
[-0.63435589  2.03743667]
[-0.92332876  0.4589538 ]
[-0.53660842  1.93553599]
[-0.57884421  0.55621257]
[ 0.87493315 -0.8537629 ]
[-0.08053237 -0.62776851]
[-1.64576437 -0.16248091]
[ 0.80846611 -0.74032796]
[-1.38824949  2.06870573]
[-0.96910891  0.5104202 ]
[0.24047363 0.18515674]
[ 2.60076189 -0.48666521]
[ 0.42736281 -0.89750232]
[ 0.47386151 -1.33068084]
[-0.90503682  0.30002713]
[ 0.81465688 -0.52559101]
[-0.39723526 -0.0360226 ]
[ 1.60847878 -1.53718772]
[0.06627732 2.04954404]
[-2.00716881  0.2258616 ]
[-0.01560438 -0.03297139]
[-0.67686058  0.0702621 ]
[0.76121119 0.77279541]
[-0.78709681 -0.15472608]
[1.02624504 1.54739465]
[ 1.45724398 -2.23800095]
[-0.32100419 -1.0439944 ]
[-0.78781662  0.51291542]
[-1.98597108  0.20524426]
[-1.57191428  2.53260893]
[-2.0657652   1.33482205]
[-0.1139418   1.91233346]
[ 0.27374684 -0.72674762]
[-0.22802708  1.74625469]
[0.52437035 0.93185151]
[-0.72296151 -0.99871583]
[0.12383625 1.52823668]
[1.16116132 1.21136541]
[ 0.59795903 -1.8533481 ]
[-0.39341476  1.35042279]
[-0.43424925 -0.22939725]
[ 1.57038009 -1.75662105]
[1.27911386 0.86502813]
[-0.89428774  1.07629824]
[-0.81391944 -1.21129968]
[-0.7978185  -0.06349326]
[-2.39598082  0.52068667]
[ 0.94356956 -1.79029752]
[ 0.56631478 -1.14242228]
```

```
[ 0.38679548 -0.5231466 ]
[-0.82060843 -0.83309907]
[-0.88534231  0.4368841 ]
[-0.72270347 -0.95007799]
[-0.97230256 -0.02680919]
[ 0.95623256 -2.18357107]
[-0.28609824 -0.74112728]
[ 1.1931934  -1.85765413]
[ 0.63407792 -1.65770159]
[-1.5752167  -0.45157034]
[1.7662413  0.80188895]
[1.95452925 0.7642341 ]
[ 0.06378791 -0.12778827]
[1.78851494 0.70454766]
[0.62656447 1.6920413 ]
[-0.499173    0.19411445]
[-1.20760712  0.32799536]
[-1.13899884  0.54782147]
[-0.32118156  0.00568035]
[-0.89137044  0.80455617]
[ 0.43372322 -0.25540054]
[ 0.66378493 -1.71231469]
[0.9744696  1.24006585]
[-0.14234692 -1.94788927]
[ 0.58671558 -1.5091447 ]
[-1.27885252  1.13727193]
[-0.79696497  0.07604189]
[ 1.47574627 -2.02908613]
[0.99206084 1.62441782]
[ 1.44638432 -2.95448702]
[ 0.80059374 -0.94764208]
[ 1.37079352 -1.27071735]
[-1.51416006  0.04784968]
[0.24222145 0.93583714]
[-2.15969336  0.73978024]
[ 1.63518159 -0.39720212]
[-0.22197967 -2.05231586]
[-0.81448665  0.34664899]
[-0.62672953 -0.27650695]
[2.12573473 0.53448874]
[1.23045514 1.19046721]
[-1.07150398 -0.67844808]
[0.58757716 2.26425843]
[ 0.07242175 -0.15905497]
[-1.44044978  1.51021198]
[0.25393874 0.16622184]
[-0.64461396  2.04265527]
[-0.26463802 -0.11698675]
[-1.05211142  0.43854887]
[-0.53277025 -0.14679433]
[0.26387781 0.94712458]
[-0.82209411 -0.48698706]
[-0.83492243  0.27583329]
[1.28753172 1.01178124]
[ 0.40703569 -0.16998365]
```

```
[0.54748788 0.56985071]
[ 0.6545547 -1.4277782]
[-1.66000761 -0.36910095]
[-0.72292562  0.40733755]
[-2.09598063  0.41533474]
[ 1.45052574 -1.59583434]
[ 0.78777524 -0.01879397]
[ 0.75578021 -1.3831111 ]
[ 1.93047125 -2.19223171]
```

Now that we have applied PCA and obtained the reduced feature set, we will check the co-relation between various Principal Components, again by using a heatmap.

In [2]:

EXPERIMENT 5 – CLUSTERING

AIM: Introduce the basic concepts and methods for data clustering, including an overview of basic cluster analysis and hierarchical methods. It also introduces methods for the evaluation of clustering.

SOFTWARE REQUIRED:

Spyder IDE 5.1.5

Anaconda3 2021.11 (Python 3.9.7 64-bit)

Anaconda Inc., 2021.11

DATA SET: Real Estate Data Set

PYTHON CODE:

```
# Importing Libraries:-
import matplotlib.pyplot as plt # Provides an implicit way of plotting
import pandas as pd  # Library for working with data sets
import scipy.cluster.hierarchy as shc # These functions cut hierarchical
clusterings into flat clusterings or find the roots of the forest formed by a
cut by providing the flat cluster ids of each observation.
from sklearn.cluster import AgglomerativeClustering # Recursively merges pair
of clusters of sample data; uses linkage distance.
from sklearn.cluster import KMeans # K-Means clustering

import warnings
warnings.filterwarnings('ignore') # Never print matching warnings

def print_csv_file(df, heading):
    print("\n")
    print('{:s}'.format('\u0332'.join(heading.center(100))))
    print(df)

# The K-Means Clustering Method:-
def k_means_clustering():

    # Finding the optimal number of clusters using the elbow method:-
    wcss_list= [] # Initialize the list for the values of WCSS

    for i in range(1, 11):
        kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42) # K-
Means clustering
```

```
        kmeans.fit(x) # Compute k-means clustering
        wcss_list.append(kmeans.inertia_) # Sum of squared distances of
samples to their closest cluster centre, weighted by the sample weights if
provided.

    plt.xlabel("Number of clusters(k)")
    plt.ylabel("wcss_list")
    plt.title("The Elbow Method Graph")
    plt.plot(range(1, 11), wcss_list)
    plt.grid(True)
    plt.show()

    print("\nFrom the above plot, we can see the elbow point is at 3. Hence,
the number of clusters here will be 3.")

    # Training the K-means algorithm on the training dataset:-
    kmeans = KMeans(n_clusters=3, init='k-means++', random_state= 42) # K-
Means clustering
    y_predict = kmeans.fit_predict(x) # Compute cluster centers and predict
cluster index for each sample

    # Visualizing the clusters:-
    plt.xlabel("Age of House in Year(s)")
    plt.ylabel("House Price per Local Unit Area")
    plt.title("Clusters of Customers")

    plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c =
'blue', label = 'Cluster 1') # For first cluster
    plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c =
'green', label = 'Cluster 2') # For second cluster
    plt.scatter(x[y_predict == 2, 0], x[y_predict == 2, 1], s = 100, c =
'red', label = 'Cluster 3') # For third cluster
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
s = 300, c = 'yellow', label = 'Centroid')

    plt.legend()
    plt.show()

# Hierarchical Clustering - Dendrogram:-
def hierarchial_clustering_dendrogram():

    # Finding the optimal number of clusters using the Dendrogram:-
    plt.xlabel("Customers")
    plt.ylabel("Euclidean Distances") # It is a metric used to compute the
linkage.
    plt.title("Dendrogram Plot")
```

```
    shc.dendrogram(shc.linkage(x, method="ward")) # This method is the popular
linkage method that we have already used for creating the Dendrogram. It
reduces the variance in each cluster.
    plt.grid(True)
    plt.show()

    print("\nUsing this Dendrogram, we will now determine the optimal number
of clusters for our model. For this, we will find the maximum vertical
distance that does not cut any horizontal bar. Accordingly, the number of
clusters will be 3.")

    # Training the hierarchical clustering model:-
    hc = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
linkage='ward') # Recursively merges pair of clusters of sample data; uses
linkage distance.
    y_predict = hc.fit_predict(x) # Compute cluster centers and predict
cluster index for each sample

    # Visualizing the clusters:-
    plt.xlabel("Age of House in Year(s)")
    plt.ylabel("House Price per Local Unit Area")
    plt.title("Clusters of Customers")

    plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c =
'blue', label = 'Cluster 1') # For first cluster
    plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c =
'green', label = 'Cluster 2') # For second cluster
    plt.scatter(x[y_predict == 2, 0], x[y_predict == 2, 1], s = 100, c =
'red', label = 'Cluster 3') # For third cluster

    plt.legend()
    plt.show()

# Driver Code: main() ; Execution starts here.

print("No - Serial Number")
print("X2 - Age of House in Year(s)")
print("X3 - Distance to Nearest MRT Station in Meter(s)")
print("X4 - Number of Convenience Stores Within Walking Distance")
print("X5 - Latitude Coordinates")
print("X5 - Longitude Coordinates")
print("Y - House Price per Local Unit Area")

# Importing the data set:-
heading = "Original Data Set"
df = pd.read_csv("Real Estate Data Set.csv")
print_csv_file(df, heading)
```

```
x = df.iloc[:, [1, 6]].values # Extracting Independent Variables; X2 and Y

print("\n"); heading = "The K-Means Clustering Method"
print('{:s}'.format('\u0332'.join(heading.center(100))))
k_means_clustering()

print("\n"); heading = "Hierarchical Clustering - Dendrogram"
print('{:s}'.format('\u0332'.join(heading.center(100))))
hierarchial_clustering_dendrogram()
```

PLOTS:

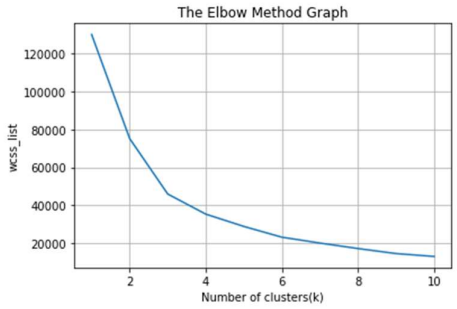


Figure 1. The Elbow Method Graph

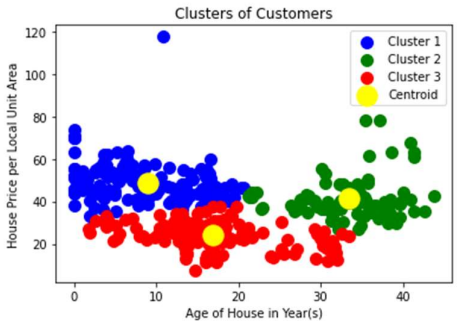


Figure 2. K-Means Clusters of Customers

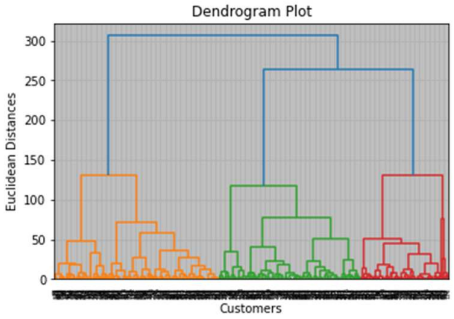


Figure 3. Dendrogram Plot

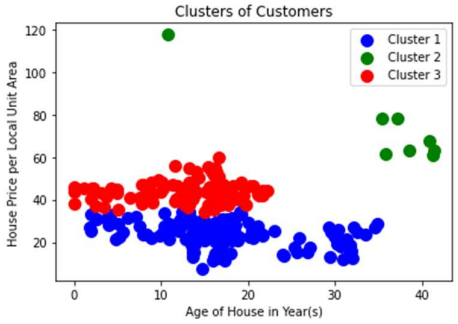


Figure 4. Hierarchical Clusters of Customers

RESULT:

Thus, presented the basic concepts and methods of cluster analysis. Learnt several basic clustering techniques and briefly discussed how to evaluate clustering methods. All the simulation results were verified successfully.

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

Restarting kernel...
```

In [1]:
```
'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 5 - Clustering/
Expt_5_Code.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 5 - Clustering'
```

```
There are six regressor variables and one response variable (namely, y):
No - Serial Number
X2 - Age of House in Year(s)
X3 - Distance to Nearest MRT Station in Meter(s)
X4 - Number of Convenience Stores Within Walking Distance
X5 - Latitude Coordinates
X5 - Longitude Coordinates
Y - House Price per Local Unit Area


                                        Original Data Set
      No  X2 house age  ...  X6 longitude  Y house price of unit area
0      1          32.0  ...     121.54024                        37.9
1      2          19.5  ...     121.53951                        42.2
2      3          13.3  ...     121.54391                        47.3
3      4          13.3  ...     121.54391                        54.8
4      5           5.0  ...     121.54245                        43.1
..   ...           ...  ...           ...                         ...
409  410          13.7  ...     121.50381                        15.4
410  411           5.6  ...     121.54310                        50.0
411  412          18.8  ...     121.53986                        40.6
412  413           8.1  ...     121.54067                        52.5
413  414           6.5  ...     121.54310                        63.9

[414 rows x 7 columns]


                                    The K-Means Clustering Method

From the above plot, we can see the elbow point is at 3. Hence, the number of clusters
here will be 3.


                                Hierarchical Clustering - Dendrogram

Using this Dendrogram, we will now determine the optimal number of clusters for our model.
For this, we will find the maximum vertical distance that does not cut any horizontal bar.
Accordingly, the number of clusters will be 3.
```

In [2]:

EXPERIMENT 6 – CLASSIFICATION

AIM: Introduce basic concepts and methods for classification, including decision tree induction, Bayes classification, and rule-based classification. Also discuss the model evaluation and selection methods for improving classification accuracy, including ensemble methods and how to handle imbalanced data.

SOFTWARE REQUIRED:

Spyder IDE 5.1.5

Anaconda3 2021.11 (Python 3.9.7 64-bit)

Anaconda Inc., 2021.11

DATA SET: Glaucoma Eye Net Data Set

PYTHON CODE:

```
# Importing Necessary Libraries:-
import matplotlib.pyplot as plt # Provides an implicit way of plotting
import numpy as np # Support for large, multi-dimensional arrays and matrices
from sklearn.metrics import accuracy_score # Accuracy classification score
from sklearn.metrics import classification_report # Build a text report
showing the main classification metrics
from sklearn.metrics import confusion_matrix # Compute confusion matrix to
evaluate the accuracy of a classification
from sklearn.metrics import roc_auc_score # Compute Area Under the Receiver
Operating Characteristic Curve (ROC AUC) from prediction scores
from sklearn.metrics import roc_curve # Compute Receiver operating
characteristic (ROC); This implementation is restricted to the binary
classification task
from sklearn.naive_bayes import GaussianNB # Gaussian Naive Bayes (GaussianNB)
from sklearn.neighbors import KNeighborsClassifier # Classifier implementing
the k-nearest neighbors vote
from sklearn.preprocessing import StandardScaler # Standardize features by
removing the mean and scaling to unit variance
from sklearn.tree import DecisionTreeClassifier # A decision tree classifier
from sklearn.svm import SVC # C-Support Vector Classification

import warnings
warnings.filterwarnings('ignore') # Never print matching warnings

def load_dataset_feature_scaling():
```

```
    X_train = np.load("eyenet_features256svm.npy")
    X_test = np.load("eyenet_test_features256svm.npy")
    y_train = np.load("Ytrain256.npy")
    y_test = np.load("Ytest256.npy")

    st_x = StandardScaler() # Standardize features by removing the mean and
scaling to unit variance
    X_train = st_x.fit_transform(X_train) # Fit to data, then transform it
    X_test = st_x.transform(X_test) # Perform standardization by centering and
scaling

    return X_train, X_test, y_train, y_test

def k_nearest_neighbor():

    X_train, X_test, y_train, y_test = load_dataset_feature_scaling()
    neighbors = np.arange(1, 11) # Return evenly spaced values within a given
interval

    # Return a new array of given shape, without initializing entries:-
    train_accuracy = np.empty(len(neighbors))
    test_accuracy = np.empty(len(neighbors))

    # Loop over K values:-
    for i, k in enumerate(neighbors):
        knn = KNeighborsClassifier(n_neighbors=k) # Classifier implementing
the k-nearest neighbors vote
        knn.fit(X_train, y_train) # Fit the k-nearest neighbors classifier
from the training dataset

        # Compute training and test data accuracy - Return the mean accuracy
on the given test data and labels:-
        train_accuracy[i] = knn.score(X_train, y_train)
        test_accuracy[i] = knn.score(X_test, y_test)

    # Generate plot:-
    plt.xlabel("n_neighbors"); plt.ylabel("Accuracy")
    plt.title("K-Nearest Neighbor (KNN) - Model Accuracy")
    plt.plot(neighbors, test_accuracy, label = 'Testing Dataset Accuracy')
    plt.plot(neighbors, train_accuracy, label = 'Training Dataset Accuracy')
    plt.legend(); plt.grid(True); plt.show()

# Function to make predictions:-
def prediction(X_test, clf_object):

    # Predicton on test with gini index:-
    y_pred = clf_object.predict(X_test)
    print("Predicted Values -"); print(y_pred)
```

```
    return y_pred

# Function to calculate accuracy:-
def cal_accuracy(y_test, y_pred):

    print("\nConfusion Matrix -\n", confusion_matrix(y_test, y_pred)) #
Compute confusion matrix to evaluate the accuracy of a classification

    print ("\nAccuracy - ", accuracy_score(y_test, y_pred)*100) # Accuracy
classification score

    print("Report -\n", classification_report(y_test, y_pred)) # Build a text
report showing the main classification metrics

def naive_bayes_classifier():

    X_train, X_test, y_train, y_test = load_dataset_feature_scaling()

    # Fitting Naive Bayes to the training set:-
    classifier = GaussianNB() # Gaussian Naive Bayes (GaussianNB)
    classifier.fit(X_train, y_train[:, 0]) # Fit Gaussian Naive Bayes
according to X, y
    y_pred= classifier.predict(X_test) # Predict the test set result
    cal_accuracy(y_test[:, 0], y_pred) # Function to calculate accuracy

def support_vector_machine():

    X_train, X_test, y_train, y_test = load_dataset_feature_scaling()

    # Fitting the SVM classifier to the training set:-
    classifier = SVC(kernel='linear', random_state=0) # C-Support Vector
Classification
    classifier.fit(X_train, y_train[:, 0]) # Fit the SVM model according to
the given training data
    y_pred = classifier.predict(X_test) # Predict the test set result
    cal_accuracy(y_test[:, 0], y_pred) # Function to calculate accuracy

def decision_tree():

    def train_using_gini(X_train, X_test, y_train):

        clf_gini = DecisionTreeClassifier(criterion = "gini", random_state =
100, max_depth = 3, min_samples_leaf = 5) # Create the classifier object
        clf_gini.fit(X_train, y_train[:, 0]) # Perform training
        return clf_gini

    def train_using_entropy(X_train, X_test, y_train):
```

```
        clf_entropy = DecisionTreeClassifier(criterion = "entropy",
random_state = 100,    max_depth = 3, min_samples_leaf = 5) # Decision tree
with entropy
        clf_entropy.fit(X_train, y_train[:, 0]) # Perform training
        return clf_entropy

    X_train, X_test, y_train, y_test = load_dataset_feature_scaling()
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = train_using_entropy(X_train, X_test, y_train)

    print("\n\t\t\t\t\t\t\t\t(i) Results Using Gini Index:-\n")
    y_pred_gini = prediction(X_test, clf_gini) # Function to make predictions
    cal_accuracy(y_test[:, 0], y_pred_gini) # Function to calculate accuracy

    print("\n\t\t\t\t\t\t\t\t(ii) Results Using Entropy:-\n")
    y_pred_entropy = prediction(X_test, clf_entropy) # Function to make
predictions
    cal_accuracy(y_test[:, 0], y_pred_entropy) # Function to calculate
accuracy

def auc_roc_curve():

    model1 = KNeighborsClassifier(n_neighbors=3) # Classifier implementing the
k-nearest neighbors vote
    model2 = GaussianNB() # Gaussian Naive Bayes (GaussianNB)
    model3 = SVC(kernel='linear', random_state=0, probability=True) # C-
Support Vector Classification
    model4 = DecisionTreeClassifier(criterion = "gini", random_state = 100,
max_depth = 3, min_samples_leaf = 5) # A decision tree classifier
    model5 = DecisionTreeClassifier(criterion = "entropy", random_state =
100,    max_depth = 3, min_samples_leaf = 5) # A decision tree classifier

    X_train, X_test, y_train, y_test = load_dataset_feature_scaling()

    # Fit Model:-
    model1.fit(X_train, y_train[:,0])
    model2.fit(X_train, y_train[:,0])
    model3.fit(X_train, y_train[:,0])
    model4.fit(X_train, y_train[:,0])
    model5.fit(X_train, y_train[:,0])

    # Predict Probabilities:-
    pred_prob1 = model1.predict_proba(X_test)
    pred_prob2 = model2.predict_proba(X_test)
    pred_prob3 = model3.predict_proba(X_test)
    pred_prob4 = model4.predict_proba(X_test)
    pred_prob5 = model5.predict_proba(X_test)
```

```
    # ROC Curve For Models:-
    fpr1, tpr1, thresh1 = roc_curve(y_test[:,0], pred_prob1[:,1], pos_label=1)
    fpr2, tpr2, thresh2 = roc_curve(y_test[:,0], pred_prob2[:,1], pos_label=1)
    fpr3, tpr3, thresh3 = roc_curve(y_test[:,0], pred_prob3[:,1], pos_label=1)
    fpr4, tpr4, thresh4 = roc_curve(y_test[:,0], pred_prob4[:,1], pos_label=1)
    fpr5, tpr5, thresh5 = roc_curve(y_test[:,0], pred_prob5[:,1], pos_label=1)

    # ROC Curve for tpr = fpr:-
    random_probs = [0 for i in range(len(y_test))]
    p_fpr, p_tpr, _ = roc_curve(y_test[:,0], random_probs, pos_label=1)

    # AUC Scores:-
    auc_score1 = roc_auc_score(y_test[:,0], pred_prob1[:,1])
    auc_score2 = roc_auc_score(y_test[:,0], pred_prob2[:,1])
    auc_score3 = roc_auc_score(y_test[:,0], pred_prob3[:,1])
    auc_score4 = roc_auc_score(y_test[:,0], pred_prob4[:,1])
    auc_score5 = roc_auc_score(y_test[:,0], pred_prob5[:,1])

    print("\nAUC Scores-")
    print("(i) K-Nearest Neighbor (KNN): ", auc_score1)
    print("(ii) Naive Bayes Classifier: ", auc_score2)
    print("(iii) Support Vector Machine (SVM): ", auc_score3)
    print("(iv) Decision Tree Using Gini Index: ", auc_score4)
    print("(v) Decision Tree Using Entropy: ", auc_score5)

    # Plot ROC Curves:-
    plt.style.use('seaborn')

    plt.plot(fpr1, tpr1, linestyle='--', color='red', label='K-Nearest
Neighbor (KNN): ' + str(round(auc_score1*100, 2)) + '%')
    plt.plot(fpr2, tpr2, linestyle='--', color='blue', label='Naive Bayes
Classifier: ' + str(round(auc_score2*100, 2)) + '%')
    plt.plot(fpr3, tpr3, linestyle='--', color='green', label='Support Vector
Machine (SVM): ' + str(round(auc_score3*100, 2)) + '%')
    plt.plot(fpr4, tpr4, linestyle='--', color='magenta', label='Decision Tree
Using Gini Index: ' + str(round(auc_score4*100, 2)) + '%')
    plt.plot(fpr5, tpr5, linestyle='--', color='brown', label='Decision Tree
Using Entropy: ' + str(round(auc_score5*100, 2)) + '%')
    plt.plot(p_fpr, p_tpr, linestyle='--', color='yellow')

    plt.xlabel("False Positive Rate"); plt.ylabel("True Positive rate")
    plt.title("ROC Curve"); plt.legend(loc='best'); plt.show()

# Driver Code: main():-
def main():

    print("\n"); heading = "K-Nearest Neighbor (KNN)"
    print('{:s}'.format('\u0332'.join(heading.center(100))))
```

```
    k_nearest_neighbor()

    print("\n"); heading = "Naive Bayes Classifier"
    print('{:s}'.format('\u0332'.join(heading.center(100))))
    naive_bayes_classifier()

    print("\n"); heading = "Decision Tree"
    print('{:s}'.format('\u0332'.join(heading.center(100))))
    decision_tree()

    print("\n"); heading = "Support Vector Machine (SVM)"
    print('{:s}'.format('\u0332'.join(heading.center(100))))
    support_vector_machine()

    print("\n"); heading = "AUC-ROC Curve"
    print('{:s}'.format('\u0332'.join(heading.center(100))))
    auc_roc_curve()

# Call main function ; Execution starts here.
if __name__=="__main__":
    main()
```

CLASSIFIERS' OUTCOMES SUMMARIZED:

Confusion Matrix	Naive Bayes Classifier	Decision Tree Using Gini Index	Decision Tree Using Entropy	Support Vector Machine (SVM)
	26 14 15 36	27 13 12 39	28 12 12 39	34 6 5 46
Accuracy	68.13186813	72.52747253	73.62637363	87.91208791
Report				

Report	Precision	Recall	F1-Score	Support
0	0.63	0.65	0.64	40
1	0.72	0.71	0.71	51
Accuracy			0.68	91
Macro Average	0.68	0.68	0.68	91
Weighted Average	0.68	0.68	0.68	91

Report	Precision	Recall	F1-Score	Support
0	0.69	0.68	0.68	40
1	0.75	0.76	0.76	51
Accuracy			0.73	91
Macro Average	0.72	0.72	0.72	91
Weighted Average	0.72	0.73	0.72	91

Report	Precision	Recall	F1-Score	Support
0	0.7	0.7	0.7	40
1	0.76	0.76	0.76	51
Accuracy			0.74	91
Macro Average	0.73	0.73	0.73	91
Weighted Average	0.74	0.74	0.74	91

Report	Precision	Recall	F1-Score	Support
0	0.87	0.85	0.86	40
1	0.88	0.9	0.89	51
Accuracy			0.88	91
Macro Average	0.88	0.88	0.88	91
Weighted Average	0.88	0.88	0.88	91

AUC Score	0.67941176	0.75833333	0.79387549	0.93578431

PLOTS:

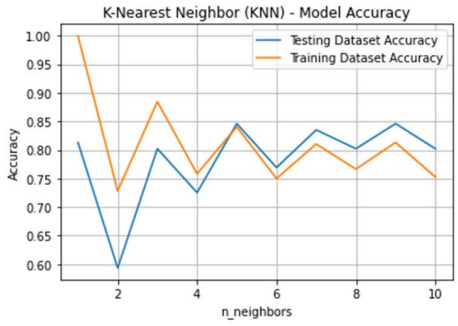


Figure 1. K-Nearest Neighbour (KNN) - Model Accuracy

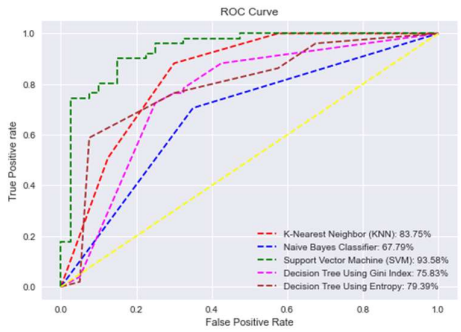


Figure 2. ROC Curve

RESULT:

Thus, described methods for data classification and prediction, including decision tree induction, Bayesian classification, support vector machine and k-nearest classifiers. Issues regarding accuracy and how to choose best classifier or predictor are discussed. All the simulation results were verified successfully.

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

Restarting kernel...
```

```
In [1]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 6 - Classification/
Expt_6_Code_Glaucoma.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/
Semester IV/19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 6 -
Classification'
```

K-Nearest Neighbor (KNN)

Naive Bayes Classifier

```
Confusion Matrix -
 [[26 14]
 [15 36]]

Accuracy -  68.13186813186813
Report -
               precision    recall  f1-score   support

         0.0       0.63      0.65      0.64        40
         1.0       0.72      0.71      0.71        51

    accuracy                           0.68        91
   macro avg       0.68      0.68      0.68        91
weighted avg       0.68      0.68      0.68        91
```

Decision Tree

(i) Results Using Gini Index:-

```
Predicted Values -
[1. 1. 0. 1. 1. 0. 1. 1. 0. 0. 1. 1. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 0. 1.
 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1.
 1. 0. 0. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 1. 1. 0. 1. 1. 0. 1.
 1. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1.]

Confusion Matrix -
 [[27 13]
 [12 39]]

Accuracy -  72.52747252747253
Report -
               precision    recall  f1-score   support
```

```
         0.0       0.69      0.68      0.68        40
         1.0       0.75      0.76      0.76        51

    accuracy                           0.73        91
   macro avg       0.72      0.72      0.72        91
weighted avg       0.72      0.73      0.72        91
```

(ii) Results Using Entropy:-

```
Predicted Values -
[1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 1. 1. 0. 1.
 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 1. 0. 1. 0. 0. 1. 1. 1. 1.
 1. 1. 0. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 1. 1. 0. 1. 1. 0. 0.
 1. 1. 0. 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1.]

Confusion Matrix -
 [[28 12]
 [12 39]]

Accuracy -  73.62637362637363
Report -
               precision    recall  f1-score   support

         0.0       0.70      0.70      0.70        40
         1.0       0.76      0.76      0.76        51

    accuracy                           0.74        91
   macro avg       0.73      0.73      0.73        91
weighted avg       0.74      0.74      0.74        91
```

Support Vector Machine (SVM)

```
Confusion Matrix -
 [[34  6]
 [ 5 46]]

Accuracy -  87.91208791208791
Report -
               precision    recall  f1-score   support

         0.0       0.87      0.85      0.86        40
         1.0       0.88      0.90      0.89        51

    accuracy                           0.88        91
   macro avg       0.88      0.88      0.88        91
weighted avg       0.88      0.88      0.88        91
```

AUC-ROC Curve

```
AUC Scores-
(i) K-Nearest Neighbor (KNN):  0.8375
(ii) Naive Bayes Classifier:  0.6779411764705883
(iii) Support Vector Machine (SVM):  0.9357843137254902
(iv) Decision Tree Using Gini Index:  0.7583333333333333
(v) Decision Tree Using Entropy:  0.7938725490196079
```

In [2]: