Adichunchanagiri Institute of Technology

Jyothinagara, Chikkamagaluru – 577102

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

(ACADEMIC YEAR 2024-25)

MACHINE LEARNING LABORATORY MANUAL

SUB CODE: BCSL606

SEMESTER: VI

As per Choice Based Credit System (CBCS)
& Outcome Based Education (OBE)

Prepared By

Mrs. Anjali B V

Lab Instructor

Approved by

Dr. Sampath S

HOD, IS&E, AIT

INSTITUTIONAL VISION and MISSION

Vision

- To develop Adichunchanagiri Institute of Technology as a center of excellence and to strive for continuous improvement of technical education and human resource advancement.

Mission

- To achieve Excellence in Education, Entrepreneurship, and Innovation by producing Engineers with high Ethical Standards, Integrity, and Credibility.

DEPARTMENT VISION and MISSION

Vision

- To be recognized as a center of excellence in information technology and allied area with quality learning and research environment.

Mission

- M1: Provide intellectual & professional leadership in ethical and social areas pertaining to information in contemporary society.
- M2: Advancing the state of knowledge of information studies through research and development.
- M3: Providing platform to discuss cutting edge technologies.

Department of Information Science & Engineering

Program Outcomes (POs)

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

PSO1: Graduates will be able to understand, analyze information technology problems and provide solutions through their problem solving skills

PSO2: Graduates will be able to apply the skills of programming in software development

PSO3:Graduates will be able to work in industries in the areas of web designing, software testing, development and maintenance

PSO4: Should have the capability to comprehend the technological advancements in the usage of modern design tools to analyze and design subsystems/processes for a variety of applications

Course outcomes (Cos)

CO1: Illustrate the principles of multivariate data and apply dimensionality reduction techniques.

CO2: Demonstrate similarity-based learning methods and perform regression analysis.

CO3: Develop decision trees for classification and regression problems, and Bayesian models for probabilistic learning.

CO4: Implement the clustering algorithms to share computing resources.

CO-PO Mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	2	2	2	1							
CO2	1	2	2	2	1							
CO3	2	2	2	2	1							
CO4	1	1	1	1	1							

CO-PSO Mapping

	PSO1	PSO2	PSO3	PSO4
CO1	2	1	1	2
CO2	2	1	1	2
CO3	2	1	1	2
CO4	2	1	1	2

Machine Learning lab		Semester	6
Course Code	BCSL606	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Credits	01	Exam Hours	100
Examination type (SEE)	Practical		

Course objectives:

- To become familiar with data and visualize univariate, bivariate, and multivariate data using statistical techniques and dimensionality reduction.
- To understand various machine learning algorithms such as similarity-based learning, regression, decision trees, and clustering.
- To familiarize with learning theories, probability-based models and developing the skills required for decision- making in dynamic environments.

Sl.NO	Experiments
1	Develop a program to create histograms for all numerical features and analyze the distribution of each feature. Generate box plots for all numerical features and identify any outliers. Use California Housing dataset. **Book 1: Chapter 2**
2	Develop a program to Compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationships between features. Use California Housing dataset. **Book 1: Chapter 2**
3	Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2. **Book 1: Chapter 2**
4	For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples. **Book 1: Chapter 3**
5	Develop a program to implement k-Nearest Neighbour algorithm to classify the randomly generated 100 values of x in the range of [0,1]. Perform the following based on dataset generated. Label the first 50 points $\{x_1,\ldots\ldots,x_{50}\}$ as follows: if (xi ≤ 0.5), then $x_i \in$ Class$_1$, else $x_i \in$ Class$_1$ Classify the remaining points, $x_{51},\ldots\ldots,x_{100}$ using KNN. Perform this for $k=1,2,3,4,5,20,30$ **Book 2: Chapter – 2**
6	Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs **Book 1: Chapter – 4**
7	Develop a program to demonstrate the working of Linear Regression and Polynomial Regression. Use Boston Housing Dataset for Linear Regression and Auto MPG Dataset (for vehicle fuel efficiency prediction) for Polynomial Regression. **Book 1: Chapter – 5**
8	Develop a program to demonstrate the working of the decision tree algorithm. Use Breast Cancer Data set for building the decision tree and apply this knowledge to classify a new sample. **Book 2: Chapter – 3**

9	Develop a program to implement the Naive Bayesian classifier considering Olivetti Face Data set for training. Compute the accuracy of the classifier, considering a few test data sets. **Book 2: Chapter – 4**
10	Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result. **Book 2: Chapter – 4**

Course outcomes (Course Skill Set):

At the end of the course the student will be able to:

- Illustrate the principles of multivariate data and apply dimensionality reduction techniques.
- Demonstrate similarity-based learning methods and perform regression analysis.
- Develop decision trees for classification and regression problems, and Bayesian models for probabilistic learning.
- Implement the clustering algorithms to share computing resources.

Laboratory outcomes: The students should be able to

1. Implement and demonstrate ML algorithms.
2. Evaluate different algorithms.

Conduction of Practical Examination:

- Experiment distribution
  - For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.
  - For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with equal opportunity.
- Students Change of experiment is allowed only once and marks allotted for procedure to be made zero of the changed part only.
- Marks Distribution (*Courseed to change in accoradance with university regulations*)
  a) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks
  b) For laboratories having PART A and PART B

i. Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks

ii. Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

CONTENT LIST

INTRODUCTION

1.1 Machine Learning

Machine Learning is used anywhere from automating mundane tasks to offering intelligent insights, industries in every sector try to benefit from it. You may already be using a device that utilizes it. For example, a wearable fitness tracker like Fitbit, or an intelligent home assistant like Google Home. But there are much more examples of ML in use.

- Prediction: Machine learning can also be used in the prediction systems. Considering the loan example, to compute the probability of a fault, the system will need to classify the available data in groups.

- Image recognition: Machine learning can be used for face detection in an image as well. There is a separate category for each person in a database of several people.

- Speech Recognition: It is the translation of spoken words into the text. It is used in voice searches and more. Voice user interfaces include voice dialing, call routing, and appliance control. It can also be used a simple data entry and the preparation of structured documents.

- Medical diagnoses: ML is trained to recognize cancerous tissues.

- Financial industry: and trading: companies use ML in fraud investigations and credit checks.

Types of Machine Learning?

Machine learning can be classified into 3 types of algorithms

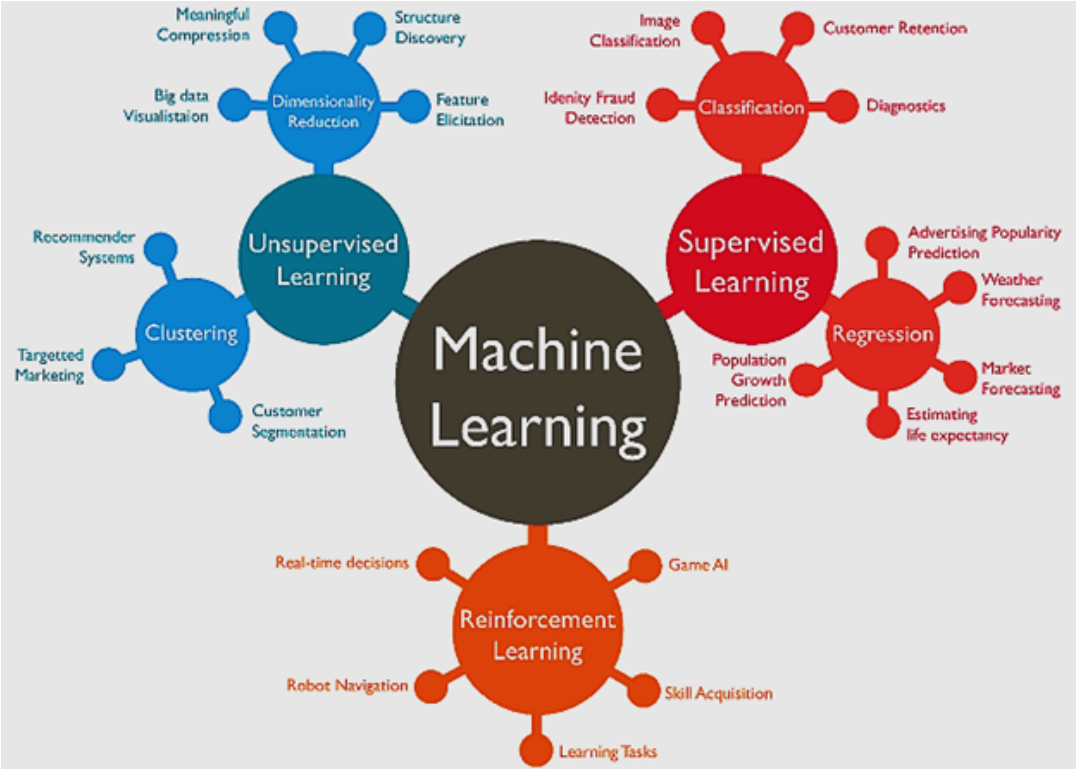
1. Supervised Learning

2. Unsupervised Learning

3. Reinforcement Learning

Overview of Supervised Learning Algorithm

In Supervised learning, an AI system is presented with data which is labeled, which means that each data tagged with the correct label. The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data



Types of Supervised learning

- Classification: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.

- Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”

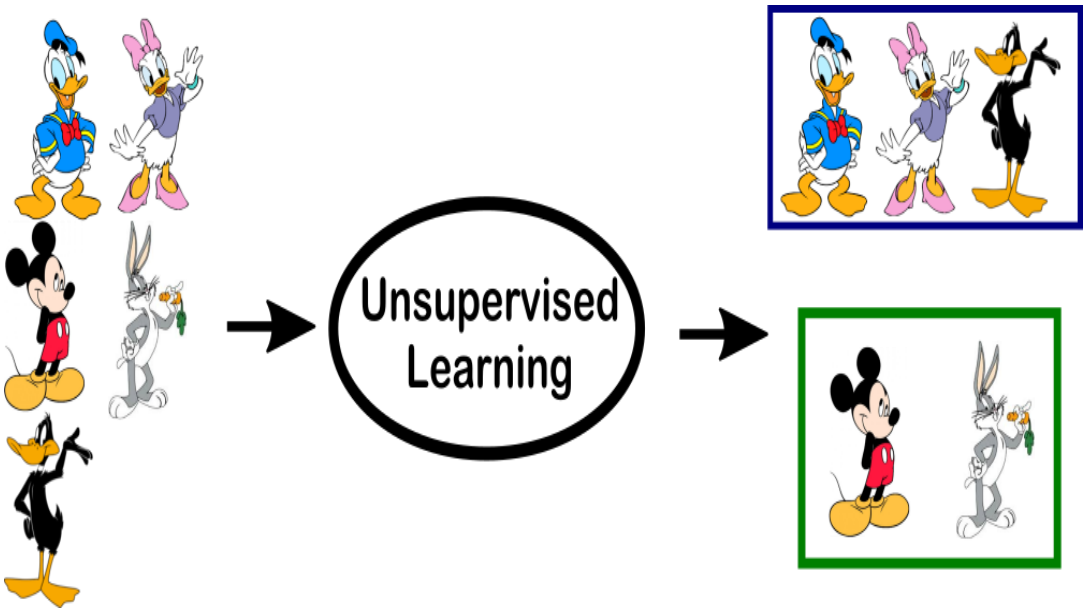
Overview of Unsupervised Learning Algorithm

In unsupervised learning, an AI system is presented with unlabeled, uncategorized data and the system’s algorithms act on the data without prior training. The output is dependent upon the coded algorithms. Subjecting a system to unsupervised learning is one way of testing AI.

Types of Unsupervised learning:

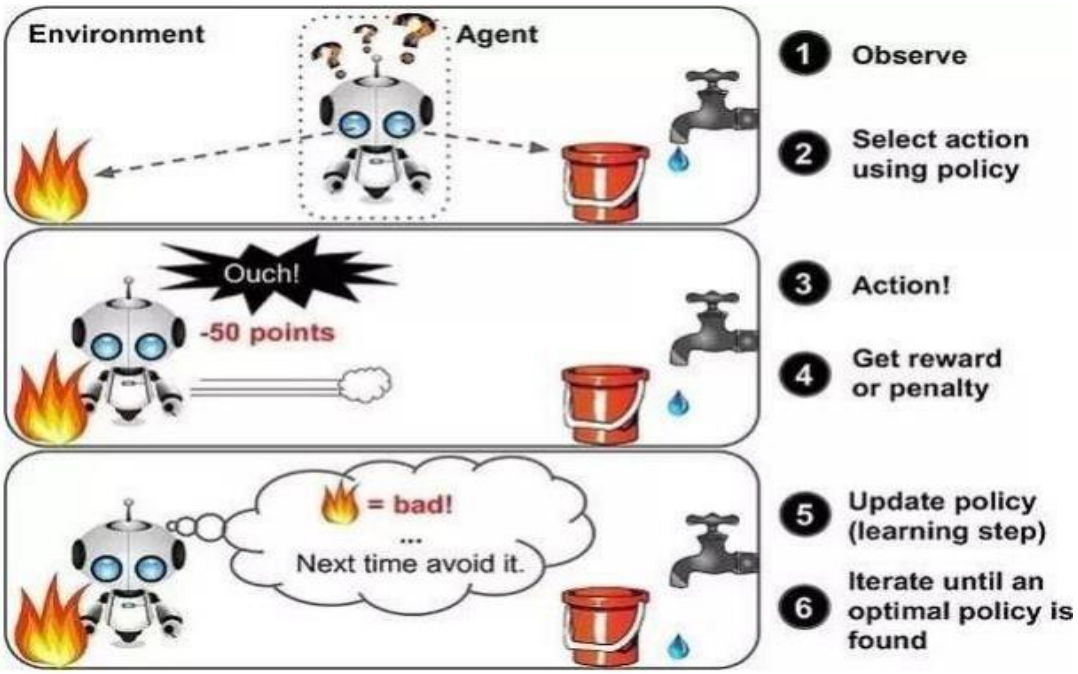
- Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

- Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.



Overview of Reinforcement Learning

A reinforcement learning algorithm, or agent, learns by interacting with its environment. The agent receives rewards by performing correctly and penalties for performing incorrectly. The agent learns without intervention from a human by maximizing its reward and minimizing its penalty. It is a type of dynamic programming that trains algorithms using a system of reward and punishment. Some more examples of tasks that are best solved by using a learning algorithm



in the above example, we can see that the agent is given 2 options i.e. a path with water or a path with fire. A reinforcement algorithm works on reward a system i.e. if the agent uses

the fire path then the rewards are subtracted and agent tries to learn that it should avoid the fire path. If it had chosen the water path or the safe path then some points would have been added to the reward points, the agent then would try to learn what path is safe and what path isn't. It is basically leveraging the rewards obtained; the agent improves its environment knowledge to select the next action.

Machine learning Approaches

Decision tree learning: Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value. Association rule learning Association rule learning is a method for discovering interesting relations between variables in large databases.

Artificial neural networks

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is vaguely inspired by biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

Deep learning

Falling hardware prices and the development of GPUs for personal use in the last few years have contributed to the development of the concept of deep learning which consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.

Inductive logic programming

Inductive logic programming (ILP) is an approach to rule learning using logic programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming languages for representing hypotheses (and not only logic programming), such as functional programs.

Clustering Cluster

analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to some pre designated criterion or criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated for example by internal compactness (similarity

between members of the same cluster) and separation between different clusters. Other methods are based on estimated density and graph connectivity. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.

Bayesian networks

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning.

1.2 Numpy

Let's start with NumPy. Among other things, NumPy contains:

A powerful N-dimensional array object.

Sophisticated (broadcasting/universal) 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.

The key to NumPy is the ndarray object, an n-dimensional array of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

NumPy arrays have a fixed size. Modifying the size means creating a new array.

NumPy arrays must be of the same data type, but this can include Python objects.

More efficient mathematical operations than built-in sequence types.

Numpy datatypes

To begin, NumPy supports a wider variety of data types than are built-in to the Python language by default. They are defined by the numpy.dtype class and include:

intc (same as a C integer) and intp (used for indexing)

int8, int16, int32, int64

uint8, uint16, uint32, uint64

float16, float32, float64

complex64, complex128

bool_, int_, float_, complex_ are shorthand for defaults.

These can be used as functions to cast literals or sequence types, as well as arguments to numpy functions that accept the dtype keyword argument.

To begin, NumPy supports a wider variety of data types than are built-in to the Python language by default. They are defined by the numpy.dtype class and include:

intc (same as a C integer) and intp (used for indexing)

int8, int16, int32, int64

uint8, uint16, uint32, uint64

float16, float32, float64

complex64, complex128

Some examples:

```
>>> import numpy as np
>>> x = np.float32(1.0)
>>> x
1.0
>>> y = np.int_([1,2,4])
>>> y
array([1, 2, 4])
>>> z = np.arange(3, dtype=np.uint8)
>>> z
array([0, 1, 2], dtype=uint8)
>>> z.dtype
dtype('uint8')
```

Numpy arrays

There are a couple of mechanisms for creating arrays in NumPy:

- Conversion from other Python structures (e.g., lists, tuples).
- Built-in NumPy array creation (e.g., arange, ones, zeros, etc.).
- Reading arrays from disk, either from standard or custom formats (e.g. reading in from a CSV file).
- and others …

In general, any numerical data that is stored in an array-like container can be converted to an ndarray through use of the array() function. The most obvious examples are sequence types like lists and tuples.

```
>>> x = np.array([2,3,1,0])
>>> x = np.array([2, 3, 1, 0])
>>> x = np.array([[1,2.0],[0,0],(1+1j,3.)])
>>> x = np.array([[ 1.+0.j, 2.+0.j], [ 0.+0.j, 0.+0.j], [ 1.+1.j, 3.+0.j]])
```

There are a couple of built-in NumPy functions which will create arrays from scratch.

- zeros(shape) -- creates an array filled with 0 values with the specified shape. The default dtype is float64.

```
>>>np.zeros((2,3))

array([[ 0., 0., 0.], [ 0., 0., 0.]])
```

- ones(shape) -- creates an array filled with 1 values.
- arange() -- creates arrays with regularly incrementing values.

```
>>>np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>>np.arange(2,10,dtype=np.float)
array([2.,3.,4.,5.,6.,7.,8.,9.])
>>>np.arange(2,3,0.1)
array([ 2. , 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9])
```

- linspace() -- creates arrays with a specified number of elements, and spaced equally between the specified beginning and end values.

```
>>>np.linspace(1.,4.,6)
array([ 1. , 1.6, 2.2, 2.8, 3.4, 4. ])
```

- random.random(shape) – creates arrays with random floats over the interval [0,1).

```
>>>np.random.random((2,3))
array([[0.75688597,0.41759916,0.35007419],[ 0.77164187, 0.05869089, 0.98792864]])
```

Printing an array can be done with the print statement.

```
>>> import numpy as np
>>> a = np.arange(3)
>>> print a
[0 1 2]
>>> a
array([0, 1, 2])
>>> b = np.arange(9).reshape(3,3)
>>> print b
[[0 1 2]
 [3 4 5]
 [6 7 8]]
>>> c = np.arange(8).reshape(2,2,2)
>>> print c
[[[0 1]
  [2 3]]
```

```
[[4 5]
 [6 7]]]
```

Indexing

Single-dimension indexing is accomplished as usual.

```
>>> x = np.arange(10)
>>> x[2]
2 >>> x[-2]
8
```

```
[ 0  1  2  3  4  5  6  7  8  9]
```

Multi-dimensional arrays support multi-dimensional indexing.

```
>>> x.shape = (2,5) # now x is 2-dimensional
>>> x[1,3]
8
>>> x[1,-1]
9
```

```
[ 0  1  2  3  4
  5  6  7  8  9]
```

Using fewer dimensions to index will result in a subarray.

```
>>> x[0]
array([0, 1, 2, 3, 4])
```

This means that x[i, j] == x[i][j] but the second method is less efficient.

Slicing is possible just as it is for typical Python sequences.

```
>>> x = np.arange(10)
>>> x[2:5]
array([2, 3, 4])
>>> x[:-7]
array([0, 1, 2])
>>> x[1:7:2]
array([1, 3, 5])
>>> y = np.arange(35).reshape(5,7)
>>> y[1:5:2,::3]
array([[ 7, 10, 13], [21, 24, 27]])
```

Array operations

```
>>> a = np.arange(5)
>>> b = np.arange(5)
>>> a+b
array([0, 2, 4, 6, 8])
```

```
>>> a-b
array([0, 0, 0, 0, 0])
>>> a**2
array([ 0,  1,  4,  9, 16])
>>> a>3
array([False, False, False, False,  True], dtype=bool)
>>> 10*np.sin(a)
array([ 0., 8.41470985, 9.09297427, 1.41120008, -7.56802495])
>>> a*b
array([ 0,  1,  4,  9, 16])
```

Basic operations apply element-wise. The result is a new array with the resultant elements. Operations like *= and += will modify the existing array.

Since multiplication is done element-wise, you need to specifically perform a dot product to perform matrix multiplication.

```
>>> a = np.zeros(4).reshape(2,2)
>>> a
array([[ 0.,  0.],
       [ 0.,  0.]])
>>> a[0,0] = 1
>>> a[1,1] = 1
>>> b = np.arange(4).reshape(2,2)
>>> b
array([[0, 1],
       [2, 3]])
>>> a*b
array([[ 0.,  0.],
       [ 0.,  3.]])
>>> np.dot(a,b)
array([[ 0.,  1.],
       [ 2.,  3.]])
```

There are also some built-in methods of ndarray objects. Universal functions which may also be applied include exp, sqrt, add, sin, cos, etc…

```
>>> a = np.random.random((2,3))
>>> a
array([[ 0.68166391, 0.98943098, 0.69361582],
       [ 0.78888081, 0.62197125, 0.40517936]])
>>> a.sum()
4.1807421388722164
>>> a.min()
0.4051793610379143
>>> a.max(axis=0)
array([ 0.78888081, 0.98943098, 0.69361582])
>>> a.min(axis=1)
array([ 0.68166391, 0.40517936])
```

Pandas

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient Data Frame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and sub setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Standard Python distribution doesn't come bundled with Pandas module. A lightweight alternative is to install NumPy using popular Python package installer, pip.

pip install pandas

If you install Anaconda Python package, Pandas will be installed by default with the following

CSV File

Python has a vast library of modules that are included with its distribution. The csv module gives the Python programmer the ability to parse CSV (Comma Separated Values) files. A CSV file is a human readable text file where each line has a number of fields, separated by commas or some other delimiter. You can think of each line as a row and each field as a column. The CSV format has no standard, but they are similar enough that the csv module will be able to read the vast majority of CSV files. You can also write CSV files using the csv module.

```
import csv
#----------------------------------------------
def csv_reader(file_obj):
    """
    Read a csv file
    """
    reader = csv.reader(file_obj)
    for row in reader:
        print(" ".join(row))
#----------------------------------------------
if __name__ == "__main__":
    csv_path = "TB_data_dictionary_2014-02-26.csv"
    with open(csv_path, "rb") as f_obj:
        csv_reader(f_obj)
```

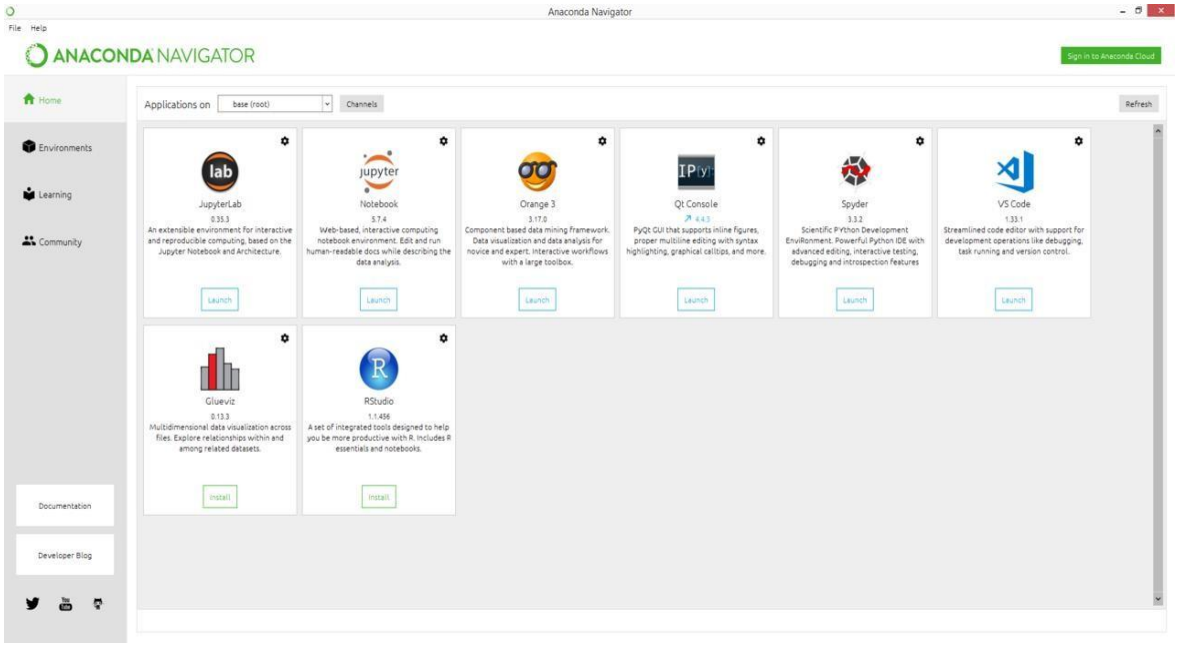
Writing a CSV File

The csv module also has two methods that you can use to write a CSV file. You can use the writer function or the DictWriter class. We'll look at both of these as well. We will be with the writer function. Let's look at a simple example:

```
import csv
#----------------------------------------------
def csv_writer(data, path):
    """
    Write data to a CSV file path
    """
    with open(path, "wb") as csv_file:
        writer = csv.writer(csv_file, delimiter=',')
        for line in data:
            writer.writerow(line)
#----------------------------------------------
if __name__ == "__main__":
    data = ["first_name,last_name,city".split(","),
            "Tyrese,Hirthe,Strackeport".split(","),
            "Jules,Dicki,Lake Nickolasville".split(","),
            "Dedric,Medhurst,Stiedemannberg".split(",")
            ]
    path = "output.csv"
    csv_writer(data, path)
```

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.



The following applications are available by default in Navigator:

JupyterLab, Jupyter Notebook, QtConsole[19], Spyder, Glue, Orange, RStudio, Visual Studio Code

Conda

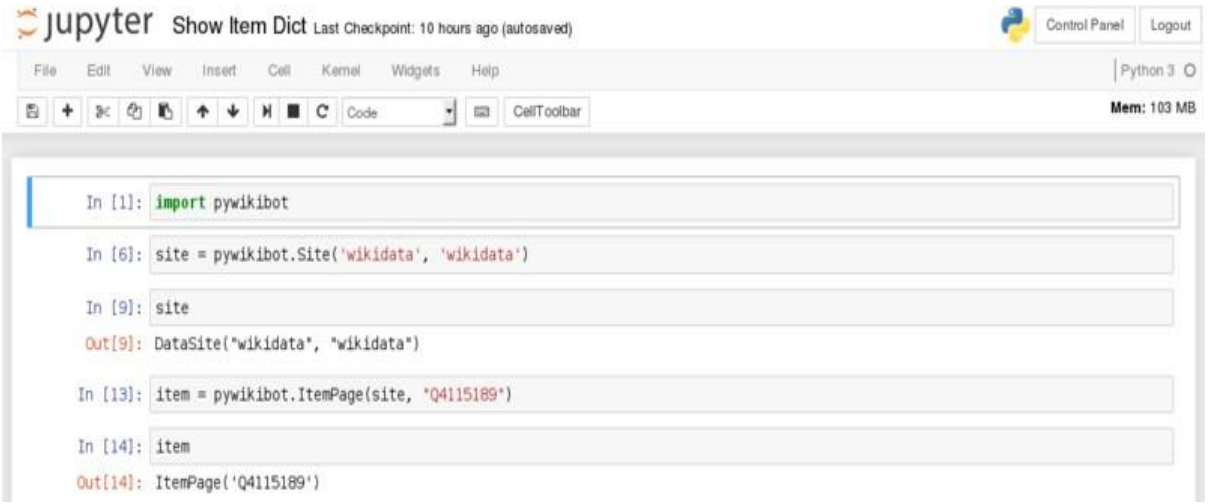
Conda is an open source cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Jupyter Notebook

Jupyter Notebook can colloquially refer to two different concepts, either the user-facing application to edit code and text, or the underlying file format which is interoperable across many implementations.

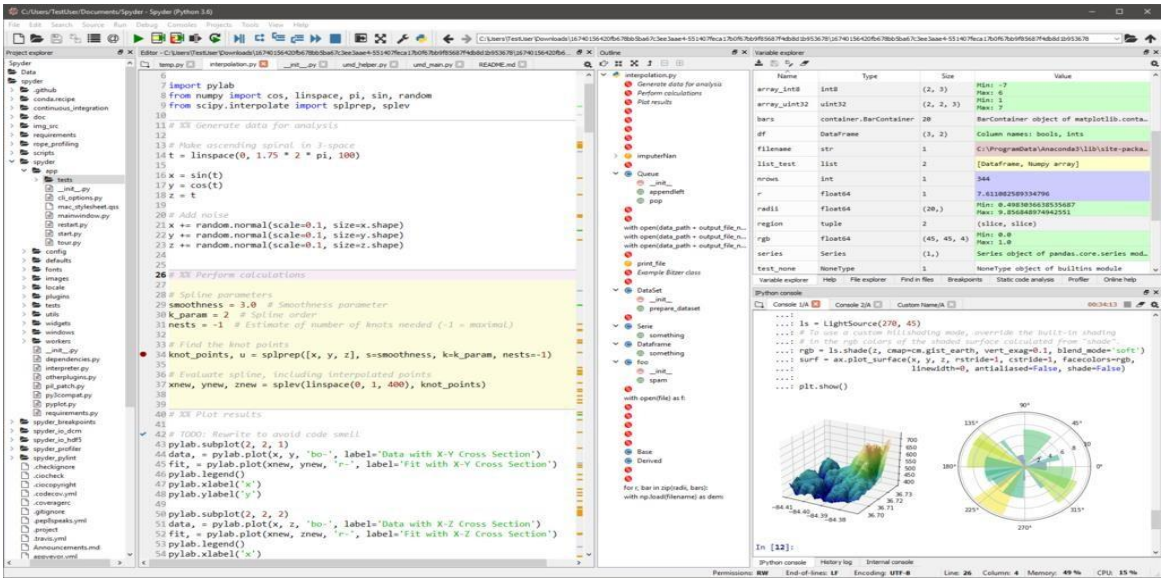
Jupyter Notebook (formerly IPython Notebook) is a web-based interactive computational environment for creating notebook documents. Jupyter Notebook is built using several open-

source libraries, including IPython, ZeroMQ, Tornado, jQuery, Bootstrap, and MathJax. A Jupyter Notebook application is a browser-based REPL containing an ordered list of input/output cells which can contain code, text (using Github Flavored Markdown), mathematics, plots and rich media.
Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s.



Spyder (software)

Spyder is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open-source software.



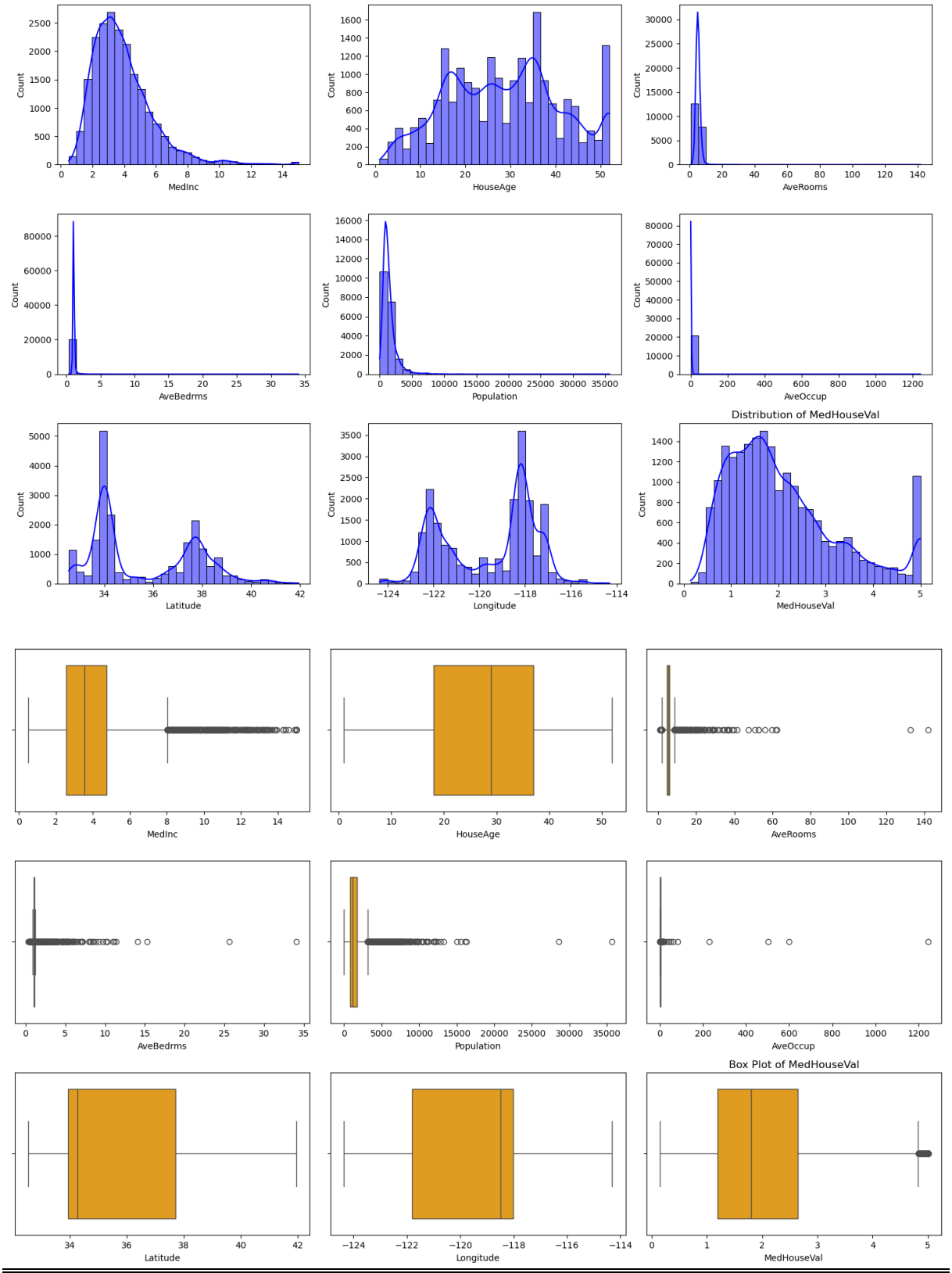
Experiment 1

1. **Develop a program to create histograms for all numerical features and analyze the distribution of each feature. Generate box plots for all numerical features and identify any outliers. Use California Housing dataset.**

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

from sklearn.datasets import fetch_california_housing
# Step 1: Load the California Housing dataset
data = fetch_california_housing(as_frame=True)
housing_df = data.frame
# Step 2: Create histograms for numerical features
numerical_features = housing_df.select_dtypes(include=[np.number]).columns
# Plot histograms
plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features):
    plt.subplot(3, 3, i + 1)
    sns.histplot(housing_df[feature], kde=True, bins=30, color='blue')
plt.title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()
# Step 3: Generate box plots for numerical features
plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features):
    plt.subplot(3, 3, i + 1)
    sns.boxplot(x=housing_df[feature], color='orange')
```

```
plt.title(f'Box Plot of {feature}')
plt.tight_layout()
plt.show()
# Step 4: Identify outliers using the IQR method
print("Outliers Detection:")
outliers_summary = {}
for feature in numerical_features:
    Q1 = housing_df[feature].quantile(0.25)
    Q3 = housing_df[feature].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = housing_df[(housing_df[feature] < lower_bound) |
(housing_df[feature] > upper_bound)]
    outliers_summary[feature] = len(outliers)
print(f"{feature}: {len(outliers)} outliers")
# Optional: Print a summary of the dataset
print("\nDataset Summary:")
print(housing_df.describe())
```

Output:


Count
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
Distribution of MedHouseVal
MedHouseVal
Box Plot of MedHouseVal

Outliers Detection:

MedHouseVal: 1071 outliers

Dataset Summary:

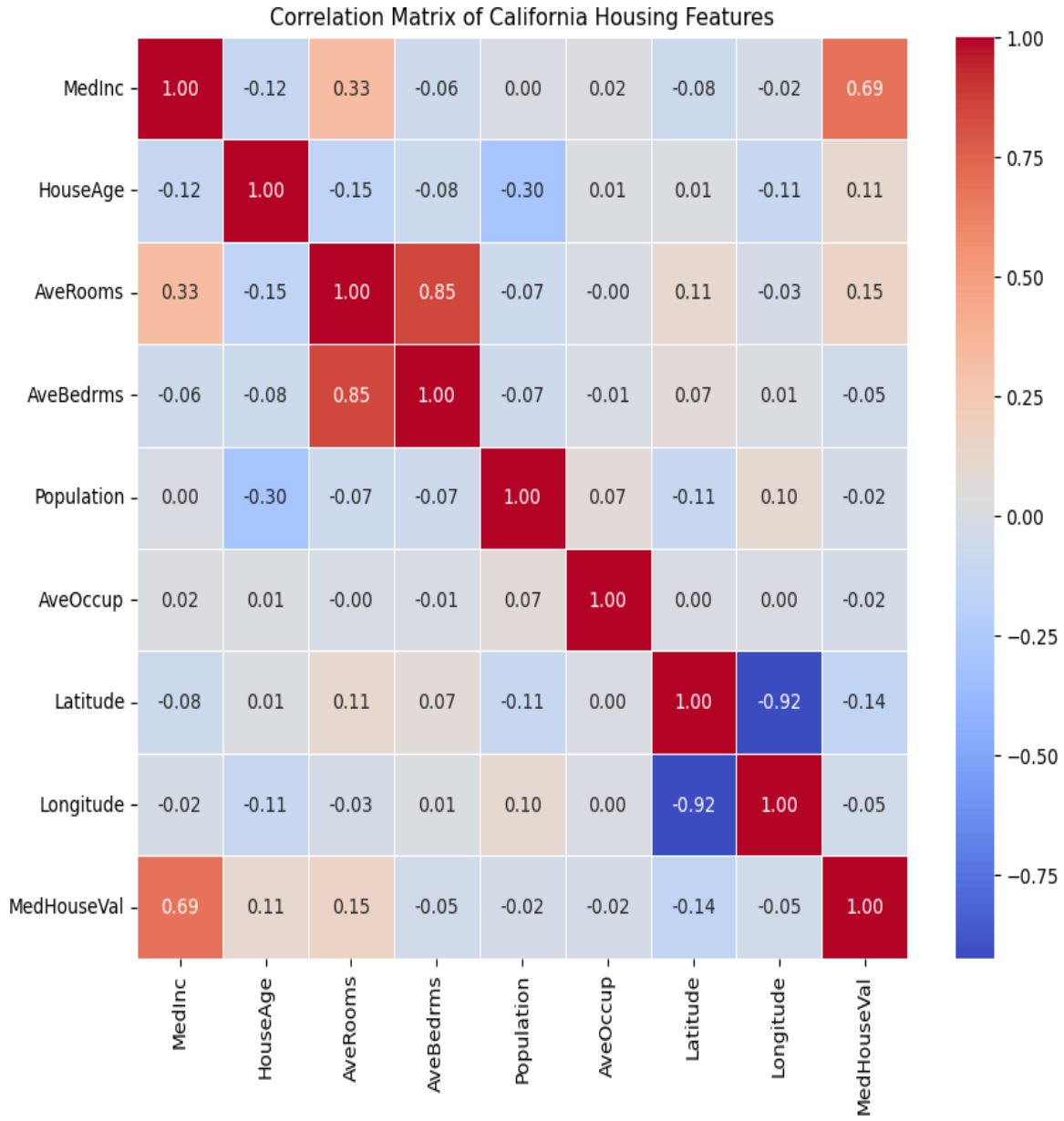
	MedInc	HouseAge	AveRooms	AveBedrms	Population	\
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	
std	1.899822	12.585558	2.474173	0.473911	1132.462122	
min	0.499900	1.000000	0.846154	0.333333	3.000000	
25%	2.563400	18.000000	4.440716	1.006079	787.000000	
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	
max	15.000100	52.000000	141.909091	34.066667	35682.000000	

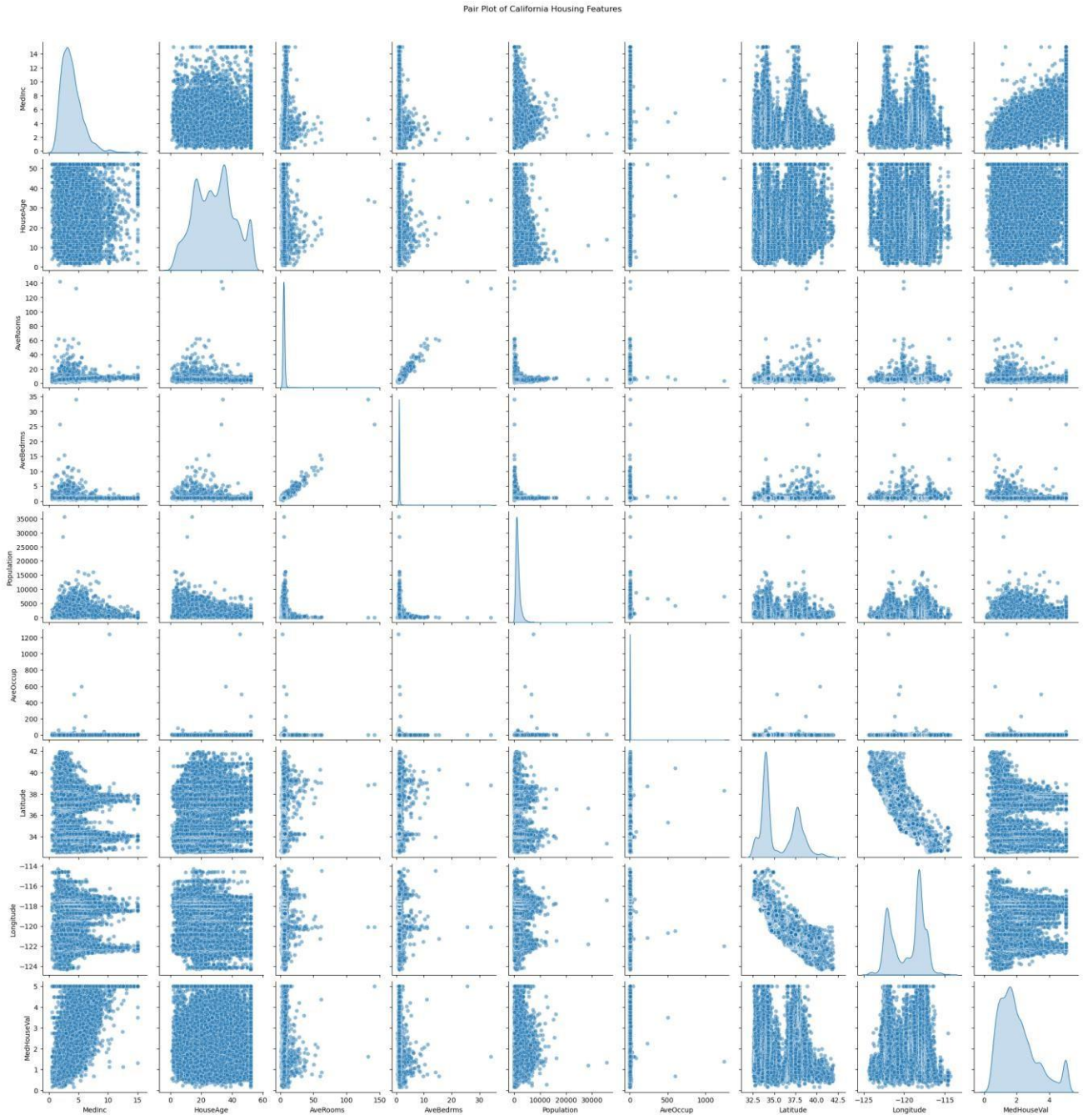
	AveOccup	Latitude	Longitude	MedHouseVal
count	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.070655	35.631861	-119.569704	2.068558
std	10.386050	2.135952	2.003532	1.153956
min	0.692308	32.540000	-124.350000	0.149990
25%	2.429741	33.930000	-121.800000	1.196000
50%	2.818116	34.260000	-118.490000	1.797000
75%	3.282261	37.710000	-118.010000	2.647250
max	1243.333333	41.950000	-114.310000	5.000010

2. Develop a program to Compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationships between features. Use California Housing dataset.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
# Step 1: Load the California Housing Dataset
california_data = fetch_california_housing(as_frame=True)
data = california_data.frame
# Step 2: Compute the correlation matrix
correlation_matrix = data.corr()
# Step 3: Visualize the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix of California Housing Features')
plt.show()
# Step 4: Create a pair plot to visualize pairwise relationships
sns.pairplot(data, diag_kind='kde', plot_kws={'alpha': 0.5})
plt.suptitle('Pair Plot of California Housing Features', y=1.02)
plt.show()
```

Output:


Correlation Matrix of California Housing Features
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal
1.00 -0.12 0.33 -0.06 0.00 0.02 -0.08 -0.02 0.69
-0.12 1.00 -0.15 -0.08 -0.30 0.01 0.01 -0.11 0.11
0.33 -0.15 1.00 0.85 -0.07 -0.00 0.11 -0.03 0.15
-0.06 -0.08 0.85 1.00 -0.07 -0.01 0.07 0.01 -0.05
0.00 -0.30 -0.07 -0.07 1.00 0.07 -0.11 0.10 -0.02
0.02 0.01 -0.00 -0.01 0.07 1.00 0.00 0.00 -0.02
-0.08 0.01 0.11 0.07 -0.11 0.00 1.00 -0.92 -0.14
-0.02 -0.11 -0.03 0.01 0.10 0.00 -0.92 1.00 -0.05
0.69 0.11 0.15 -0.05 -0.02 -0.02 -0.14 -0.05 1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

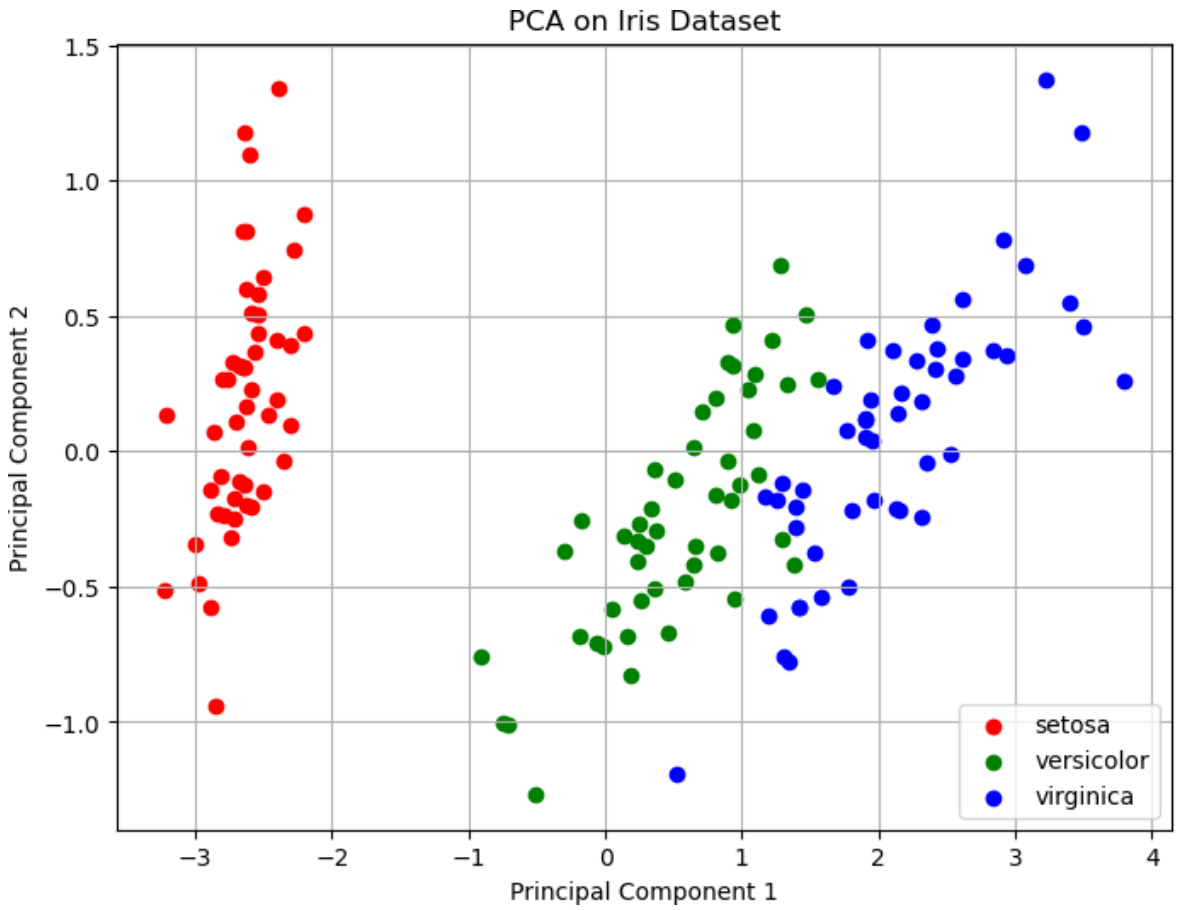

Pair Plot of California Housing Features
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal

3. Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2.

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
# Load the Iris dataset
iris = load_iris()
data = iris.data
labels = iris.target
label_names = iris.target_names
# Convert to a DataFrame for better visualization
iris_df = pd.DataFrame(data, columns=iris.feature_names)
# Perform PCA to reduce dimensionality to 2
pca = PCA(n_components=2)
data_reduced = pca.fit_transform(data)
# Create a DataFrame for the reduced data
reduced_df = pd.DataFrame(data_reduced, columns=['Principal Component 1',
'Principal Component 2'])
reduced_df['Label'] = labels
# Plot the reduced data
plt.figure(figsize=(8, 6))
colors = ['r', 'g', 'b']
for i, label in enumerate(np.unique(labels)):
    plt.scatter(
        reduced_df[reduced_df['Label'] == label]['Principal Component 1'],
```

```
        reduced_df[reduced_df['Label'] == label]['Principal Component 2'],
        label=label_names[label],
        color=colors[i]
    )
plt.title('PCA on Iris Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.grid()
plt.show()
```

Output:



4. For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S algorithm to output a description of the set of all hypotheses consistent with the training examples.

```
import pandas as pd

# Function to implement the Find-S Algorithm

def find_s_algorithm(file_path):

    # Read the dataset

    data = pd.read_csv(file_path)

    print("Training data:")

    print(data)


    # Separate attributes and class label

    attributes = data.columns[:-1]  # All columns except the last one

    class_label = data.columns[-1]  # Last column is the class label


    # Initialize hypothesis with '?' for each attribute

    hypothesis = ['?' for _ in attributes]


    # Iterate through the dataset to find the specific hypothesis

    for index, row in data.iterrows():

        # Only consider positive examples (class label == 'Yes')

        if row[class_label] == 'Yes':

            for i, value in enumerate(row[attributes]):

                # If hypothesis is '?' or matches the current value, update hypothesis

                if hypothesis[i] == '?' or hypothesis[i] == value:

                    hypothesis[i] = value
```

```
            else:
                # If the value is different, set '?' for that attribute
                hypothesis[i] = '?'
    return hypothesis

# Define file path
file_path = r"C:\Users\atme\Desktop\example ml\training_data.csv"

# Call the Find-S algorithm
hypothesis = find_s_algorithm(file_path)

# Print the final hypothesis
print("\nThe final hypothesis is:", hypothesis)
```

Output:

```
Training data:Outlook Temperature Humidity Windy PlayTennis
0     Sunny    Hot    High False   No
1     Sunny    Hot    High True    No
2    Overcast   Hot    High False    Yes
3      Rain   Cold    High False   Yes
4      Rain   Cold    High True    No
5    Overcast   Hot    High True     Yes
6     Sunny    Hot    High False   No

The final hypothesis is: ['Overcast', 'Hot', 'High', '?']
```

5. Develop a program to implement k-Nearest Neighbour algorithm to classify the randomly generated 100 values of x in the range of [0,1]. Perform the following based on dataset generated. a. Label the first 50 points {x1,……,x50} as follows:

if (xi ≤ 0.5), then xi ϵ Class1, else xi ϵ Class1 b. Classify the remaining points, x51,……,x100 using KNN. Perform this for k=1,2,3,4,5,20,30

```
import numpy as np

import matplotlib.pyplot as plt

from collections import Counter

# Generate random data

data = np.random.rand(100)

labels = ["Class1" if x <= 0.5 else "Class2" for x in data[:50]]

# Function to compute Euclidean distance

def euclidean_distance(x1, x2):

    return abs(x1 - x2)


# k-NN classifier function

def knn_classifier(train_data, train_labels, test_point, k):

    distances = [(euclidean_distance(test_point, train_data[i]), train_labels[i]) for i in range(len(train_data))]

    distances.sort(key=lambda x: x[0])  # Sort by distance


    k_nearest_neighbors = distances[:k]  # Get k nearest neighbors

    k_nearest_labels = [label for _, label in k_nearest_neighbors]  # Extract the labels

    return Counter(k_nearest_labels).most_common(1)[0][0]     # Return the most common label
```

```
# Prepare the training and test data
train_data = data[:50]  # First 50 points for training
train_labels = labels  # Corresponding labels for training
test_data = data[50:]  # Remaining 50 points for testing

# Values of k to test
k_values = [1, 2, 3, 4, 5, 20, 30]
print("--- k-Nearest Neighbors Classification ---")
print("Training dataset: First 50 points labeled based on the rule (x <= 0.5 -> Class1, x > 0.5 -> Class2)")
print("Testing dataset: Remaining 50 points to be classified\n")

# Store the results for each k value
results = {}
# Loop over different values of k
for k in k_values:
    print(f"Results for k = {k}:")
    classified_labels = [knn_classifier(train_data, train_labels, test_point, k) for test_point in test_data]
    results[k] = classified_labels

    # Output the classification results
    for i, label in enumerate(classified_labels, start=51): # Start index at 51 for test points
        print(f"Point x{i} (value: {test_data[i - 51]:.4f}) is classified as {label}")
    print("\n")
print("Classification complete.\n")
```

```
# Visualize the classification results for each k value

for k in k_values:

    classified_labels = results[k]

    class1_points = [test_data[i] for i in range(len(test_data)) if classified_labels[i] == "Class1"]

    class2_points = [test_data[i] for i in range(len(test_data)) if classified_labels[i] == "Class2"]



    plt.figure(figsize=(10, 6))

    plt.scatter(train_data, [0] * len(train_data), c=["blue" if label == "Class1" else "red" for label in train_labels],

                label="Training Data", marker="o")

    plt.scatter(class1_points, [1] * len(class1_points), c="blue", label="Class1 (Test)", marker="x")

    plt.scatter(class2_points, [1] * len(class2_points), c="red", label="Class2 (Test)", marker="x")

    plt.title(f"k-NN Classification Results for k = {k}")

    plt.xlabel("Data Points")

    plt.ylabel("Classification Level")

    plt.legend()

    plt.grid(True)

    plt.show()
```

Output:

--- k-Nearest Neighbors Classification ---

Training dataset: First 50 points labeled based on the rule (x <= 0.5 -> Class1, x > 0.5 -> Class2)

Testing dataset: Remaining 50 points to be classified

Results for k = 1:
Point x51 (value: 0.4701) is classified as Class1
Point x52 (value: 0.2775) is classified as Class1
Point x53 (value: 0.2544) is classified as Class1
Point x54 (value: 0.4693) is classified as Class1
Point x55 (value: 0.6401) is classified as Class2
Point x56 (value: 0.8233) is classified as Class2
Point x57 (value: 0.6015) is classified as Class2
Point x58 (value: 0.7991) is classified as Class2
Point x59 (value: 0.0437) is classified as Class1
Point x60 (value: 0.2407) is classified as Class1
Point x61 (value: 0.5801) is classified as Class2
Point x62 (value: 0.4369) is classified as Class1
Point x63 (value: 0.3841) is classified as Class1
Point x64 (value: 0.3875) is classified as Class1
Point x65 (value: 0.3774) is classified as Class1
Point x66 (value: 0.7355) is classified as Class2
Point x67 (value: 0.9212) is classified as Class2
Point x68 (value: 0.9475) is classified as Class2
Point x69 (value: 0.1168) is classified as Class1
Point x70 (value: 0.1124) is classified as Class1
Point x71 (value: 0.0971) is classified as Class1
Point x72 (value: 0.4307) is classified as Class1
Point x73 (value: 0.9223) is classified as Class2
Point x74 (value: 0.7348) is classified as Class2
Point x75 (value: 0.3533) is classified as Class1
Point x76 (value: 0.8451) is classified as Class2
Point x77 (value: 0.5237) is classified as Class2
Point x78 (value: 0.7778) is classified as Class2
Point x79 (value: 0.4949) is classified as Class1
Point x80 (value: 0.1261) is classified as Class1
Point x81 (value: 0.5396) is classified as Class2
Point x82 (value: 0.2353) is classified as Class1
Point x83 (value: 0.5715) is classified as Class2
Point x84 (value: 0.5105) is classified as Class2
Point x85 (value: 0.5449) is classified as Class2
Point x86 (value: 0.8197) is classified as Class2
Point x87 (value: 0.2319) is classified as Class1
Point x88 (value: 0.5876) is classified as Class2
Point x89 (value: 0.8649) is classified as Class2
Point x90 (value: 0.3587) is classified as Class1
Point x91 (value: 0.0785) is classified as Class1
Point x92 (value: 0.8560) is classified as Class2
Point x93 (value: 0.8341) is classified as Class2
Point x94 (value: 0.0014) is classified as Class1
Point x95 (value: 0.0512) is classified as Class1
Point x96 (value: 0.4411) is classified as Class1
Point x97 (value: 0.7493) is classified as Class2
Point x98 (value: 0.6286) is classified as Class2
Point x99 (value: 0.0223) is classified as Class1
Point x100 (value: 0.3796) is classified as Class1

Results for k = 2:
Point x51 (value: 0.4701) is classified as Class1
Point x52 (value: 0.2775) is classified as Class1
Point x53 (value: 0.2544) is classified as Class1
Point x54 (value: 0.4693) is classified as Class1
Point x55 (value: 0.6401) is classified as Class2
Point x56 (value: 0.8233) is classified as Class2
Point x57 (value: 0.6015) is classified as Class2
Point x58 (value: 0.7991) is classified as Class2
Point x59 (value: 0.0437) is classified as Class1
Point x60 (value: 0.2407) is classified as Class1
Point x61 (value: 0.5801) is classified as Class2
Point x62 (value: 0.4369) is classified as Class1
Point x63 (value: 0.3841) is classified as Class1
Point x64 (value: 0.3875) is classified as Class1
Point x65 (value: 0.3774) is classified as Class1
Point x66 (value: 0.7355) is classified as Class2
Point x67 (value: 0.9212) is classified as Class2
Point x68 (value: 0.9475) is classified as Class2
Point x69 (value: 0.1168) is classified as Class1
Point x70 (value: 0.1124) is classified as Class1
Point x71 (value: 0.0971) is classified as Class1
Point x72 (value: 0.4307) is classified as Class1
Point x73 (value: 0.9223) is classified as Class2
Point x74 (value: 0.7348) is classified as Class2
Point x75 (value: 0.3533) is classified as Class1
Point x76 (value: 0.8451) is classified as Class2
Point x77 (value: 0.5237) is classified as Class2
Point x78 (value: 0.7778) is classified as Class2
Point x79 (value: 0.4949) is classified as Class1
Point x80 (value: 0.1261) is classified as Class1
Point x81 (value: 0.5396) is classified as Class2
Point x82 (value: 0.2353) is classified as Class1
Point x83 (value: 0.5715) is classified as Class2
Point x84 (value: 0.5105) is classified as Class2
Point x85 (value: 0.5449) is classified as Class2
Point x86 (value: 0.8197) is classified as Class2
Point x87 (value: 0.2319) is classified as Class1
Point x88 (value: 0.5876) is classified as Class2
Point x89 (value: 0.8649) is classified as Class2
Point x90 (value: 0.3587) is classified as Class1
Point x91 (value: 0.0785) is classified as Class1
Point x92 (value: 0.8560) is classified as Class2
Point x93 (value: 0.8341) is classified as Class2
Point x94 (value: 0.0014) is classified as Class1
Point x95 (value: 0.0512) is classified as Class1
Point x96 (value: 0.4411) is classified as Class1
Point x97 (value: 0.7493) is classified as Class2
Point x98 (value: 0.6286) is classified as Class2
Point x99 (value: 0.0223) is classified as Class1

Results for k = 3:

Point x51 (value: 0.4701) is classified as Class1
Point x52 (value: 0.2775) is classified as Class1
Point x53 (value: 0.2544) is classified as Class1
Point x54 (value: 0.4693) is classified as Class1
Point x55 (value: 0.6401) is classified as Class2
Point x56 (value: 0.8233) is classified as Class2
Point x57 (value: 0.6015) is classified as Class2
Point x58 (value: 0.7991) is classified as Class2
Point x59 (value: 0.0437) is classified as Class1
Point x60 (value: 0.2407) is classified as Class1
Point x61 (value: 0.5801) is classified as Class2
Point x62 (value: 0.4369) is classified as Class1
Point x63 (value: 0.3841) is classified as Class1
Point x64 (value: 0.3875) is classified as Class1
Point x65 (value: 0.3774) is classified as Class1
Point x66 (value: 0.7355) is classified as Class2
Point x67 (value: 0.9212) is classified as Class2
Point x68 (value: 0.9475) is classified as Class2
Point x69 (value: 0.1168) is classified as Class1
Point x70 (value: 0.1124) is classified as Class1
Point x71 (value: 0.0971) is classified as Class1
Point x72 (value: 0.4307) is classified as Class1
Point x73 (value: 0.9223) is classified as Class2
Point x74 (value: 0.7348) is classified as Class2
Point x75 (value: 0.3533) is classified as Class1
Point x76 (value: 0.8451) is classified as Class2
Point x77 (value: 0.5237) is classified as Class2
Point x78 (value: 0.7778) is classified as Class2
Point x79 (value: 0.4949) is classified as Class1
Point x80 (value: 0.1261) is classified as Class1
Point x81 (value: 0.5396) is classified as Class2
Point x82 (value: 0.2353) is classified as Class1
Point x83 (value: 0.5715) is classified as Class2
Point x84 (value: 0.5105) is classified as Class1
Point x85 (value: 0.5449) is classified as Class2
Point x86 (value: 0.8197) is classified as Class2
Point x87 (value: 0.2319) is classified as Class1
Point x88 (value: 0.5876) is classified as Class2
Point x89 (value: 0.8649) is classified as Class2
Point x90 (value: 0.3587) is classified as Class1
Point x91 (value: 0.0785) is classified as Class1
Point x92 (value: 0.8560) is classified as Class2
Point x93 (value: 0.8341) is classified as Class2
Point x94 (value: 0.0014) is classified as Class1
Point x95 (value: 0.0512) is classified as Class1
Point x96 (value: 0.4411) is classified as Class1
Point x97 (value: 0.7493) is classified as Class2
Point x98 (value: 0.6286) is classified as Class2
Point x99 (value: 0.0223) is classified as Class1
Point x100 (value: 0.3796) is classified as Class1

Results for k = 5:

Point x51 (value: 0.4701) is classified as Class1
Point x52 (value: 0.2775) is classified as Class1
Point x53 (value: 0.2544) is classified as Class1
Point x54 (value: 0.4693) is classified as Class1
Point x55 (value: 0.6401) is classified as Class2
Point x56 (value: 0.8233) is classified as Class2
Point x57 (value: 0.6015) is classified as Class2
Point x58 (value: 0.7991) is classified as Class2
Point x59 (value: 0.0437) is classified as Class1
Point x60 (value: 0.2407) is classified as Class1
Point x61 (value: 0.5801) is classified as Class2
Point x62 (value: 0.4369) is classified as Class1
Point x63 (value: 0.3841) is classified as Class1
Point x64 (value: 0.3875) is classified as Class1
Point x65 (value: 0.3774) is classified as Class1
Point x66 (value: 0.7355) is classified as Class2
Point x67 (value: 0.9212) is classified as Class2
Point x68 (value: 0.9475) is classified as Class2
Point x69 (value: 0.1168) is classified as Class1
Point x70 (value: 0.1124) is classified as Class1
Point x71 (value: 0.0971) is classified as Class1
Point x72 (value: 0.4307) is classified as Class1
Point x73 (value: 0.9223) is classified as Class2
Point x74 (value: 0.7348) is classified as Class2
Point x75 (value: 0.3533) is classified as Class1
Point x76 (value: 0.8451) is classified as Class2
Point x77 (value: 0.5237) is classified as Class2
Point x78 (value: 0.7778) is classified as Class2
Point x79 (value: 0.4949) is classified as Class1
Point x80 (value: 0.1261) is classified as Class1
Point x81 (value: 0.5396) is classified as Class2
Point x82 (value: 0.2353) is classified as Class1
Point x83 (value: 0.5715) is classified as Class2
Point x84 (value: 0.5105) is classified as Class2
Point x85 (value: 0.5449) is classified as Class2
Point x86 (value: 0.8197) is classified as Class2
Point x87 (value: 0.2319) is classified as Class1
Point x88 (value: 0.5876) is classified as Class2
Point x89 (value: 0.8649) is classified as Class2
Point x90 (value: 0.3587) is classified as Class1
Point x91 (value: 0.0785) is classified as Class1
Point x92 (value: 0.8560) is classified as Class2
Point x93 (value: 0.8341) is classified as Class2
Point x94 (value: 0.0014) is classified as Class1
Point x95 (value: 0.0512) is classified as Class1
Point x96 (value: 0.4411) is classified as Class1
Point x97 (value: 0.7493) is classified as Class2
Point x98 (value: 0.6286) is classified as Class2
Point x99 (value: 0.0223) is classified as Class1
Point x100 (value: 0.3796) is classified as Class1

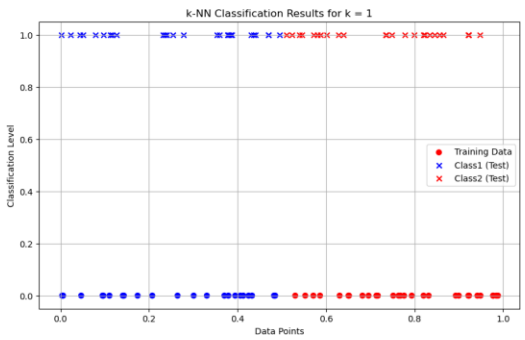
Results for k = 20:

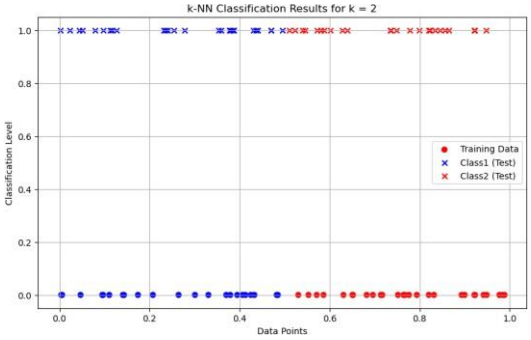
Point x51 (value: 0.4701) is classified as Class1
Point x52 (value: 0.2775) is classified as Class1
Point x53 (value: 0.2544) is classified as Class1
Point x54 (value: 0.4693) is classified as Class1
Point x55 (value: 0.6401) is classified as Class2
Point x56 (value: 0.8233) is classified as Class2
Point x57 (value: 0.6015) is classified as Class2
Point x58 (value: 0.7991) is classified as Class2
Point x59 (value: 0.0437) is classified as Class1
Point x60 (value: 0.2407) is classified as Class1
Point x61 (value: 0.5801) is classified as Class2
Point x62 (value: 0.4369) is classified as Class1
Point x63 (value: 0.3841) is classified as Class1
Point x64 (value: 0.3875) is classified as Class1
Point x65 (value: 0.3774) is classified as Class1
Point x66 (value: 0.7355) is classified as Class2
Point x67 (value: 0.9212) is classified as Class2
Point x68 (value: 0.9475) is classified as Class2
Point x69 (value: 0.1168) is classified as Class1
Point x70 (value: 0.1124) is classified as Class1
Point x71 (value: 0.0971) is classified as Class1
Point x72 (value: 0.4307) is classified as Class1
Point x73 (value: 0.9223) is classified as Class2
Point x74 (value: 0.7348) is classified as Class2
Point x75 (value: 0.3533) is classified as Class1
Point x76 (value: 0.8451) is classified as Class2
Point x77 (value: 0.5237) is classified as Class2
Point x78 (value: 0.7778) is classified as Class2
Point x79 (value: 0.4949) is classified as Class1
Point x80 (value: 0.1261) is classified as Class1
Point x81 (value: 0.5396) is classified as Class2
Point x82 (value: 0.2353) is classified as Class1
Point x83 (value: 0.5715) is classified as Class2
Point x84 (value: 0.5105) is classified as Class2
Point x85 (value: 0.5449) is classified as Class2
Point x86 (value: 0.8197) is classified as Class2
Point x87 (value: 0.2319) is classified as Class1
Point x88 (value: 0.5876) is classified as Class2
Point x89 (value: 0.8649) is classified as Class2
Point x90 (value: 0.3587) is classified as Class1
Point x91 (value: 0.0785) is classified as Class1
Point x92 (value: 0.8560) is classified as Class2
Point x93 (value: 0.8341) is classified as Class2
Point x94 (value: 0.0014) is classified as Class1
Point x95 (value: 0.0512) is classified as Class1
Point x96 (value: 0.4411) is classified as Class1
Point x97 (value: 0.7493) is classified as Class2
Point x98 (value: 0.6286) is classified as Class2
Point x99 (value: 0.0223) is classified as Class1
Point x100 (value: 0.3796) is classified as Class1

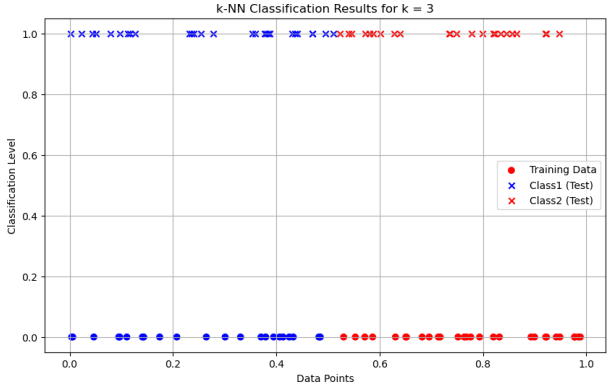
Results for k = 30:

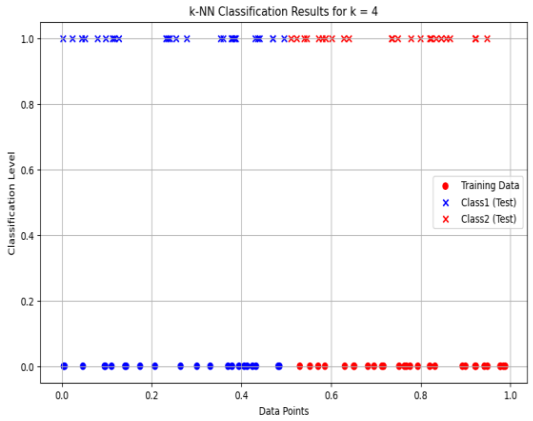
Point x51 (value: 0.4701) is classified as Class2
Point x52 (value: 0.2775) is classified as Class1
Point x53 (value: 0.2544) is classified as Class1
Point x54 (value: 0.4693) is classified as Class2
Point x55 (value: 0.6401) is classified as Class2
Point x56 (value: 0.8233) is classified as Class2
Point x57 (value: 0.6015) is classified as Class2
Point x58 (value: 0.7991) is classified as Class2
Point x59 (value: 0.0437) is classified as Class1
Point x60 (value: 0.2407) is classified as Class1
Point x61 (value: 0.5801) is classified as Class2
Point x62 (value: 0.4369) is classified as Class1
Point x63 (value: 0.3841) is classified as Class1
Point x64 (value: 0.3875) is classified as Class1
Point x65 (value: 0.3774) is classified as Class1
Point x66 (value: 0.7355) is classified as Class2
Point x67 (value: 0.9212) is classified as Class2
Point x68 (value: 0.9475) is classified as Class2
Point x69 (value: 0.1168) is classified as Class1
Point x70 (value: 0.1124) is classified as Class1
Point x71 (value: 0.0971) is classified as Class1
Point x72 (value: 0.4307) is classified as Class1
Point x73 (value: 0.9223) is classified as Class2
Point x74 (value: 0.7348) is classified as Class2
Point x75 (value: 0.3533) is classified as Class1
Point x76 (value: 0.8451) is classified as Class2
Point x77 (value: 0.5237) is classified as Class2
Point x78 (value: 0.7778) is classified as Class2
Point x79 (value: 0.4949) is classified as Class2
Point x80 (value: 0.1261) is classified as Class1
Point x81 (value: 0.5396) is classified as Class2
Point x82 (value: 0.2353) is classified as Class1
Point x83 (value: 0.5715) is classified as Class2
Point x84 (value: 0.5105) is classified as Class2
Point x85 (value: 0.5449) is classified as Class2
Point x86 (value: 0.8197) is classified as Class2
Point x87 (value: 0.2319) is classified as Class1
Point x88 (value: 0.5876) is classified as Class2
Point x89 (value: 0.8649) is classified as Class2
Point x90 (value: 0.3587) is classified as Class1
Point x91 (value: 0.0785) is classified as Class1
Point x92 (value: 0.8560) is classified as Class2
Point x93 (value: 0.8341) is classified as Class2
Point x94 (value: 0.0014) is classified as Class1
Point x95 (value: 0.0512) is classified as Class1
Point x96 (value: 0.4411) is classified as Class1
Point x97 (value: 0.7493) is classified as Class2
Point x98 (value: 0.6286) is classified as Class2
Point x99 (value: 0.0223) is classified as Class1
Point x100 (value: 0.3796) is classified as Class1

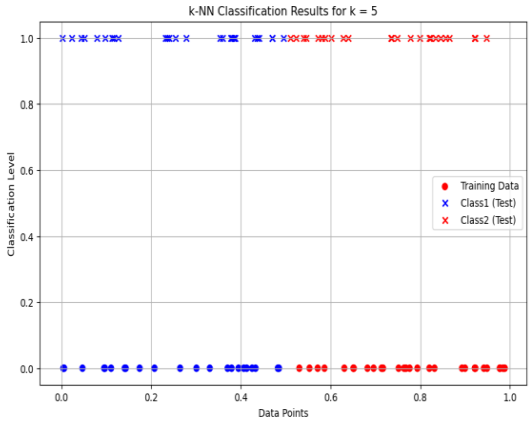
Classification complete.

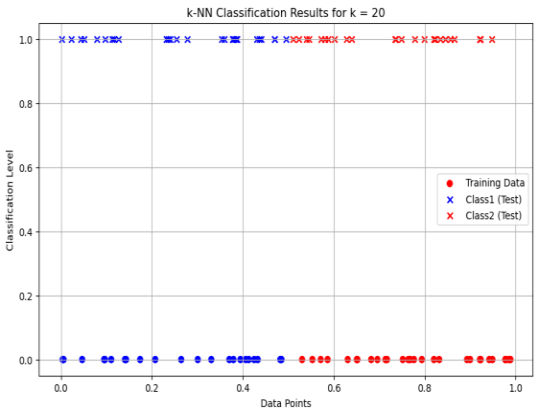


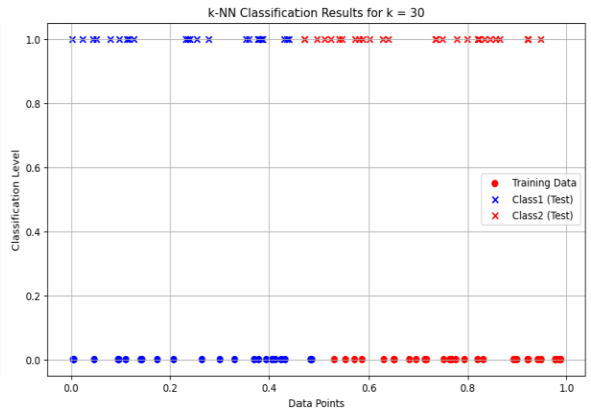










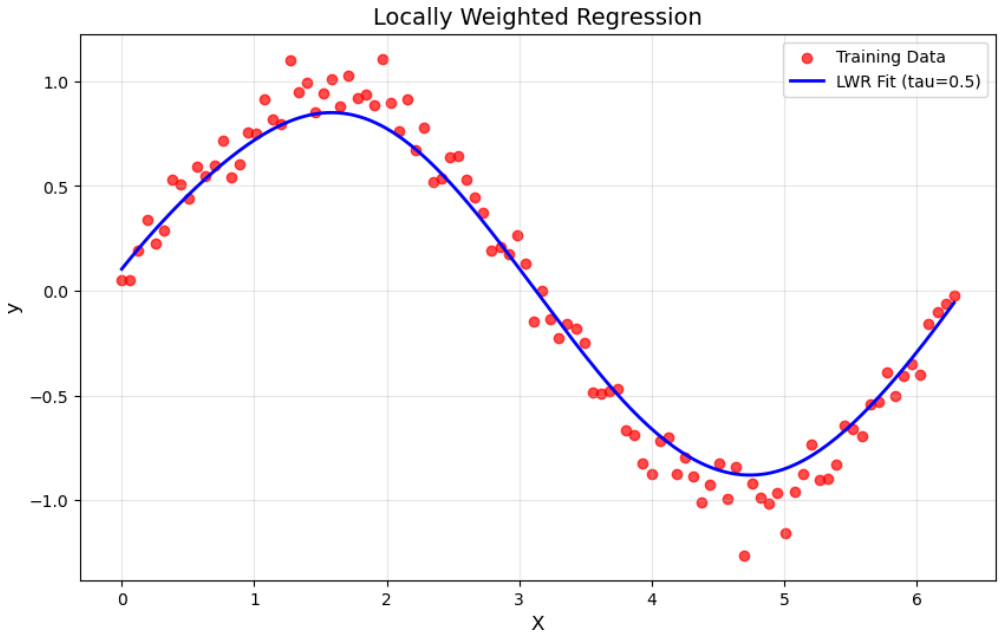


6. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs

```
import numpy as np
import matplotlib.pyplot as plt
def gaussian_kernel(x, xi, tau):
  return np.exp(-np.sum((x - xi) ** 2) / (2 * tau ** 2))
  def locally_weighted_regression(x, X, y, tau):
  m = X.shape[0]
  weights = np.array([gaussian_kernel(x, X[i], tau) for i in range(m)])
  W = np.diag(weights)
  X_transpose_W = X.T @ W
  theta = np.linalg.inv(X_transpose_W @ X) @ X_transpose_W @ y
  return x @ theta

np.random.seed(42)
X = np.linspace(0, 2 * np.pi, 100)
y = np.sin(X) + 0.1 * np.random.randn(100)
X_bias = np.c_[np.ones(X.shape), X]
x_test = np.linspace(0, 2 * np.pi, 200)
x_test_bias = np.c_[np.ones(x_test.shape), x_test]
tau = 0.5
y_pred = np.array([locally_weighted_regression(xi, X_bias, y, tau) for xi in
x_test_bias])
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='red', label='Training Data', alpha=0.7)
plt.plot(x_test, y_pred, color='blue', label=f'LWR Fit (tau={tau})', linewidth=2)
```

```
plt.xlabel('X', fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title('Locally Weighted Regression', fontsize=14)
plt.legend(fontsize=10)
plt.grid(alpha=0.3)
plt.show()
```

Output:



7. Develop a program to demonstrate the working of Linear Regression and Polynomial Regression. Use Boston Housing Dataset for Linear Regression and Auto MPG Dataset (for vehicle fuel efficiency prediction) for Polynomial Regression.

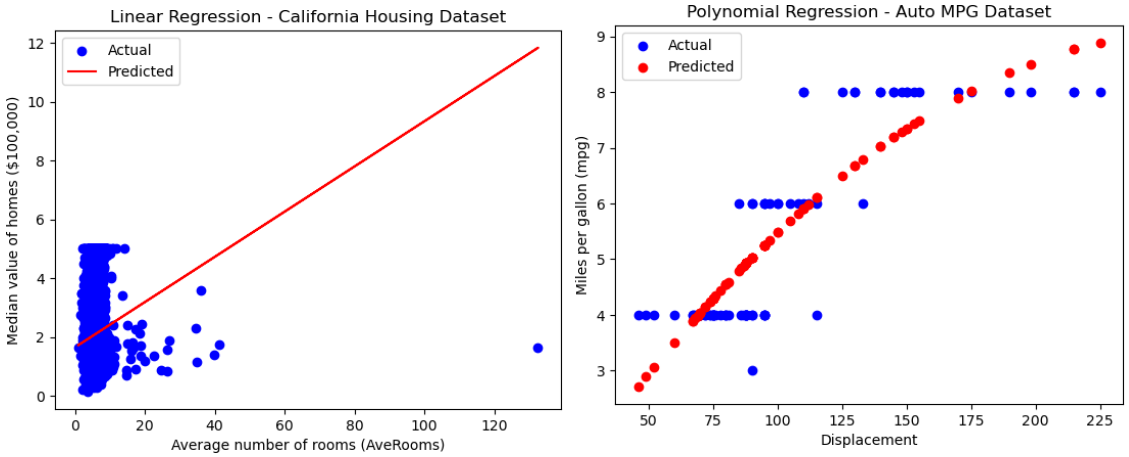
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, r2_score


# Linear Regression for California Housing Dataset
def linear_regression_california():
    housing = fetch_california_housing(as_frame=True)
    X = housing.data[["AveRooms"]]  # Using only AveRooms as feature
    y = housing.target  # Median value of homes as target
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


    # Linear Regression model
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)



    # Plotting results
```

```
    plt.scatter(X_test, y_test, color="blue", label="Actual")
    plt.plot(X_test, y_pred, color="red", label="Predicted")
    plt.xlabel("Average number of rooms (AveRooms)")
    plt.ylabel("Median value of homes ($100,000)")
    plt.title("Linear Regression - California Housing Dataset")
    plt.legend()
    plt.show()

    # Print performance metrics
    print("Linear Regression - California Housing Dataset")
    print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
    print("R^2 Score:", r2_score(y_test, y_pred))

# Polynomial Regression for Auto MPG Dataset
def polynomial_regression_auto_mpg():
    # Load the Auto MPG dataset
    url = "https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
    column_names = ["mpg", "cylinders", "displacement", "horsepower", "weight", "acceleration", "model_year", "origin"]
    data = pd.read_csv(url, sep='\s+', names=column_names, na_values="?")
    data = data.dropna()  # Drop rows with missing values
    X = data["displacement"].values.reshape(-1, 1)  # Feature: displacement
    y = data["mpg"].values  # Target: mpg

    # Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Polynomial Regression model (degree 2)

poly_model = make_pipeline(PolynomialFeatures(degree=2), StandardScaler(), LinearRegression())

poly_model.fit(X_train, y_train)

y_pred = poly_model.predict(X_test)


# Plotting results

plt.scatter(X_test, y_test, color="blue", label="Actual")

plt.scatter(X_test, y_pred, color="red", label="Predicted")

plt.xlabel("Displacement")

plt.ylabel("Miles per gallon (mpg)")

plt.title("Polynomial Regression - Auto MPG Dataset")

plt.legend()

plt.show()
```

```
    # Print performance metrics
    print("Polynomial Regression - Auto MPG Dataset")
    print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
    print("R^2 Score:", r2_score(y_test, y_pred))
if __name__== "__main__":
    print("Demonstrating Linear Regression and Polynomial Regression\n")
    linear_regression_california()  # Call the linear regression function
    polynomial_regression_auto_mpg()  # Call the polynomial regression function
```

Output:



Linear Regression - California Housing Dataset

Mean Squared Error: 1.2923314440807299

R^2 Score: 0.013795337532284901

Polynomial Regression - Auto MPG Dataset

Mean Squared Error: 0.7431490557205862

R^2 Score: 0.7505650609469626

8. Develop a program to demonstrate the working of the decision tree algorithm. Use Breast Cancer Data set for building the decision tree and apply this knowledge to classify a new sample.

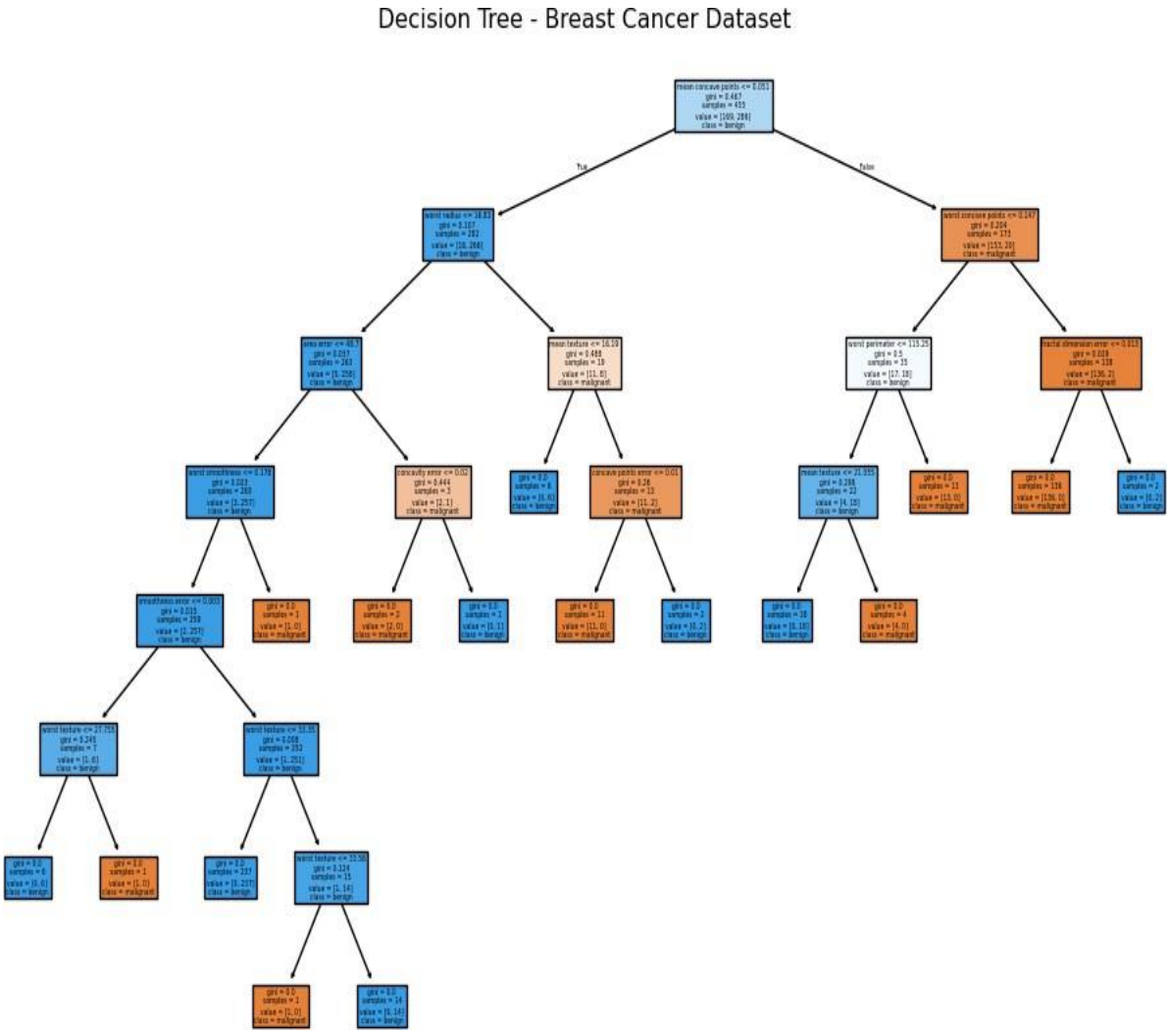
```
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
data = load_breast_cancer()
X = data.data
y = data.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")
new_sample = np.array([X_test[0]])
prediction = clf.predict(new_sample)
prediction_class = "Benign" if prediction == 1 else "Malignant"
print(f"Predicted Class for the new sample: {prediction_class}")
plt.figure(figsize=(12,8))
```

```
tree.plot_tree(clf, filled=True, feature_names=data.feature_names,
class_names=data.target_names)
plt.title("Decision Tree - Breast Cancer Dataset")
plt.show()
```

Output:

Model Accuracy: 94.74%

Predicted Class for the new sample: Benign



9. Develop a program to implement the Naive Bayesian classifier considering Olivetti Face Data set for training. Compute the accuracy of the classifier, considering a few test data sets.

```
import numpy as np

from sklearn.datasets import fetch_olivetti_faces

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
data = fetch_olivetti_faces(shuffle=True, random_state=42)

X = data.data
y = data.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
print("\nClassification Report:")
print(classification_report(y_test, y_pred, zero_division=1))
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
cross_val_accuracy = cross_val_score(gnb, X, y, cv=5, scoring='accuracy')
print(f'\nCross-validation accuracy: {cross_val_accuracy.mean() * 100:.2f}%')
fig, axes = plt.subplots(3, 5, figsize=(12, 8))
for ax, image, label, prediction in zip(axes.ravel(), X_test, y_test, y_pred):
    ax.imshow(image.reshape(64, 64), cmap=plt.cm.gray)
    ax.set_title(f"True: {label}, Pred: {prediction}")
    ax.axis('off')
plt.show()
```

Output:

Accuracy: 80.83%

Classification Report:

	precision	recall	f1-score	support
0	0.67	1.00	0.80	2
1	1.00	1.00	1.00	2
2	0.33	0.67	0.44	3
3	1.00	0.00	0.00	5
4	1.00	0.50	0.67	4
5	1.00	1.00	1.00	2
7	1.00	0.75	0.86	4
8	1.00	0.67	0.80	3
9	1.00	0.75	0.86	4
10	1.00	1.00	1.00	3
11	1.00	1.00	1.00	1
12	0.40	1.00	0.57	4
13	1.00	0.80	0.89	5
14	1.00	0.40	0.57	5
15	0.67	1.00	0.80	2
16	1.00	0.67	0.80	3
17	1.00	1.00	1.00	3
18	1.00	1.00	1.00	3
19	0.67	1.00	0.80	2
20	1.00	1.00	1.00	3

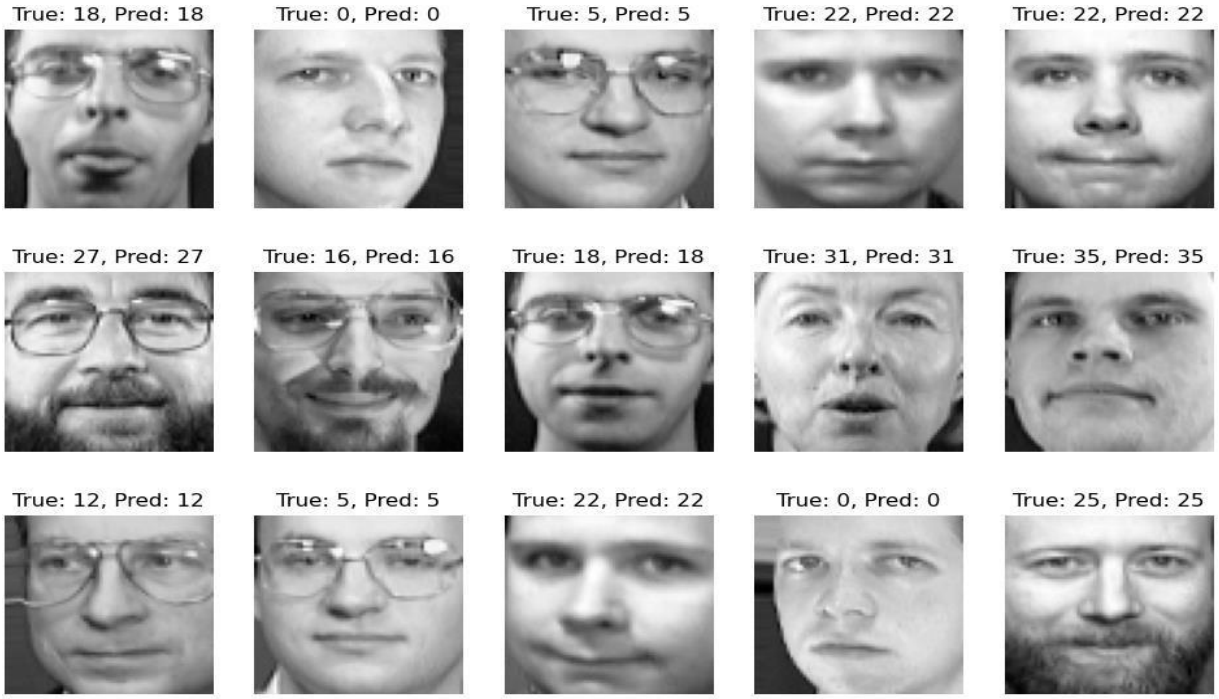
...

[0 0 0 ... 0 3 0]

[0 0 0 ... 0 0 5]]

Cross-validation accuracy: 87.25%

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



10. Develop a program to implement k-means clustering using Wisconsin Breast Cancer data set and visualize the clustering result.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_breast_cancer
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix, classification_report
data = load_breast_cancer()
X = data.data
y = data.target
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
kmeans = KMeans(n_clusters=2, random_state=42)
y_kmeans = kmeans.fit_predict(X_scaled)
print("Confusion Matrix:")
print(confusion_matrix(y, y_kmeans))
print("\nClassification Report:")
print(classification_report(y, y_kmeans))
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
df = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])
df['Cluster'] = y_kmeans
df['True Label'] = y
plt.figure(figsize=(8, 6))
```

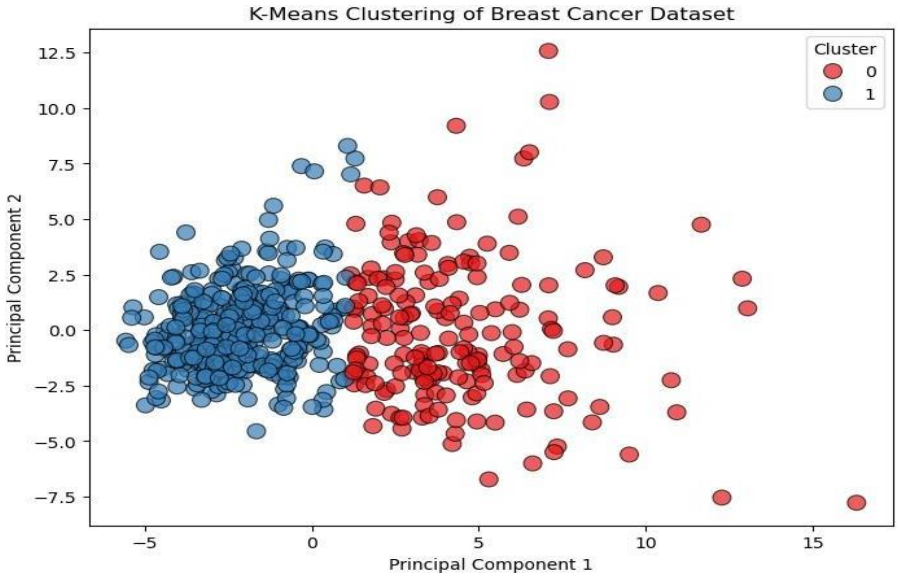
```
sns.scatterplot(data=df, x='PC1', y='PC2', hue='Cluster', palette='Set1', s=100,
edgecolor='black', alpha=0.7)
plt.title('K-Means Clustering of Breast Cancer Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="Cluster")
plt.show()
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='True Label', palette='coolwarm',
s=100, edgecolor='black', alpha=0.7)
plt.title('True Labels of Breast Cancer Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="True Label")
plt.show()
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='Cluster', palette='Set1', s=100,
edgecolor='black', alpha=0.7)
centers = pca.transform(kmeans.cluster_centers_)
plt.scatter(centers[:, 0], centers[:, 1], s=200, c='red', marker='X', label='Centroids')
plt.title('K-Means Clustering with Centroids')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="Cluster")
plt.show()
```

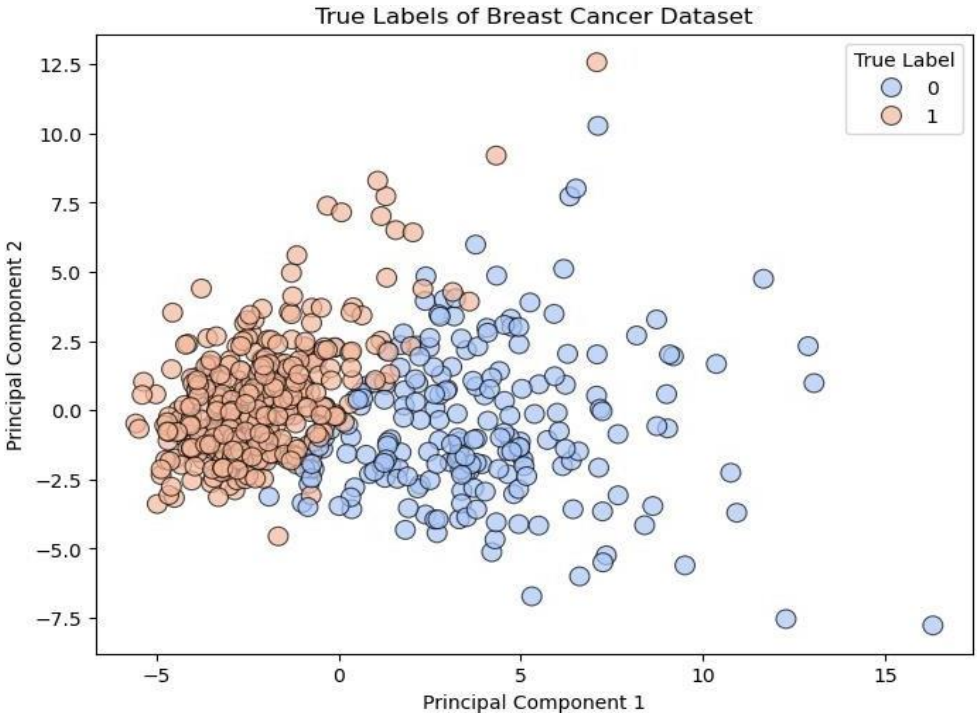
Output:

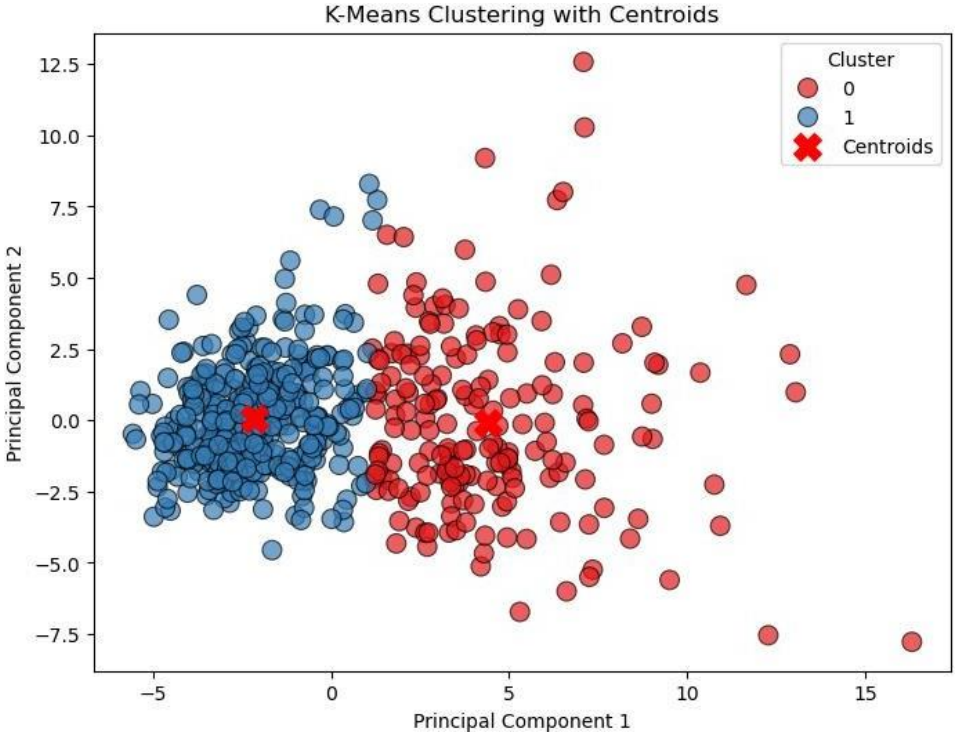
Confusion Matrix:
[[175 37]
[13 344]]

Classification Report:

	precision	recall	f1-score	support
0	0.93	0.83	0.88	212
1	0.90	0.96	0.93	357
accuracy			0.91	569
macro avg	0.92	0.89	0.90	569
weighted avg	0.91	0.91	0.91	569




True Labels of Breast Cancer Dataset
True Label
0
1
Principal Component 2
Principal Component 1
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
15


K-Means Clustering with Centroids
Cluster
0
1
Centroids
Principal Component 2
Principal Component 1
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−5
0
5
10
15

VIVA QUESTIONS

1. What is the difference between supervised and unsupervised machine learning?

A Supervised learning is a process where it requires training labeled data. When it comes to Unsupervised learning it doesn't require data labeling.

2. How is KNN different from K-means clustering?

KNN stands for K- Nearest Neighbours, it is classified as a supervised algorithm. K-means is an unsupervised cluster algorithm.

4. How to handle or missing data in a dataset?

An individual can easily find missing or corrupted data in a data set either by dropping the rows or columns. On contrary, they can decide to replace the data with another value. In Pandas they are two ways to identify the missing data, these two methods are very useful. isnull() and dropna().

5. What is the difference between an array and Linked list?

Deep An array is an ordered fashion of collection of objects. A linked list is a series of objects that are processed in a sequential order.

6. Explain why Navie Bayes is so Naive?

It is based on an assumption that all of the features in the data set are important, equal and independent.

7. Please state few popular Machine Learning algorithms?

Nearest Neighbour
Neural Networks
Decision Trees etc
Support vector machines

8. What are the different types of algorithm techniques available in machine learning?

Some of them are :
Supervised learning
Unsupervised learning
Semi-supervised learning
Transduction
Learning to learn

9. What are the three stages to build the model in machine learning?

1. Model building
2. Model testing
3. Applying the model

10. Name a few libraries in Python used for Data Analysis and Scientific computations

NumPy, SciPy, Pandas, SciKit, Matplotlib, Seaborn

11. Which is the standard data missing marker used in Pandas?

NaN

12. Write the code to sort an array in NumPy by the nth column?

Using argsort () function this can be achieved. If there is an array X and you would like to sort the nth column then code for this will be x[x [: n-1].argsort ()]

13. What is pylab?

A package that combines NumPy, SciPy and Matplotlib into a single namespace.

14. Is all the memory freed when Python exits?

No it is not, because the objects that are referenced from global namespaces of Python modules are not always de-allocated when Python exits.

15. How can you randomize the items of a list in place in Python?

Shuffle (lst) can be used for randomizing the items of a list in Python

16. Which tool in Python will be used to find bugs if any?

Pylint and Pychecker. Pylint verifies that a module satisfies all the coding standards or not. Pychecker is a static analysis tool that helps find out bugs in the course code.

17. What are the supported data types in Python?

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

19. What are the supported sequence types in Python?

Python supports 7 sequence types. They are str, list, tuple, unicode, byte array, xrange, and buffer. where xrange is deprecated in python 3.5.X.

20. What is pylab?

A package that combines NumPy, SciPy and Matplotlib into a single namespace.

21. What are the supported data types in Python?

SciKit-Learn

23. Is Python a case-sensitive programming language?

Yes, it is a case-sensitive language.

24. What is the difference between a tuple and a list?

The basic difference between a tuple and a list is that the former is immutable and the latter is mutable.

25. What is the difference between Xrange() and range()?

Range() returns a list and xrange() returns an xrange object, which is kind of like an iterator and generates the numbers on demand.

26. Optimize the below python code-

```
word = 'word'
print word.__len__()
```

print 'word'._len_ ()

27. What is PEP 8?

PEP 8 is a coding convention that lets us write more readable code. In other words, it is a set of recommendations.

28. What are Python decorators?

A Python decorator is a specific change that we make in Python syntax to alter functions easily.

29. What is Dict and List comprehensions are?

They are syntax constructions to ease the creation of a Dictionary or List based on existing iterable.

30. What is lambda in Python?

It is a single expression anonymous function often used as inline function.

31. What is pass in Python?

Pass means, no-operation Python statement, or in other words it is a place holder in compound statement, where there should be a blank left and nothing has to be written there.

32. In Python what are iterators?

In Python, iterators are used to iterate a group of elements, containers like list.

33. In Python what is slicing?

A mechanism to select a range of items from sequence types like list, tuple, strings etc. is known as slicing.

34. What is docstring in Python?

A Python documentation string is known as docstring, it is a way of documenting Python functions, modules and classes.

35. How can you copy an object in Python?

To copy an object in Python, you can try copy.copy () or copy.deepcopy() for the general case. You cannot copy all objects but most of them.

36. Explain how to delete a file in Python?

By using a command os.remove (filename) or os.unlink(filename)

37. Is It Mandatory For A Python Function To Return A Value?

It is not at all necessary for a function to return any value. However, if needed, we can use None as a return value.

38. What Is Whitespace In Python?

Whitespace represents the characters that we use for spacing and separation. They possess an "empty" representation. In Python, it could be a tab or space.

39. What Is Isalpha() In Python?

Python provides this built-in isalpha() function for the string handling purpose. It returns True if all characters in the string are of alphabet type, else it returns False.

40. What Does The Join Method Do In Python?

Python provides the join() method which works on strings, lists, and tuples. It combines them and returns a united value.

41. What Makes The CPython Different From Python?

CPython has its core developed in C. The prefix 'C' represents this fact. It runs an interpreter loop used for translating the Python-ish code to C language.

42. What Is A Tuple In Python?

A tuple is a collection type data structure in Python which is immutable. They are similar to sequences, just like the lists. However, there are some differences between a tuple and list; the former doesn't allow modifications whereas the list does.

43. What is Artificial Intelligence?

Artificial Intelligence is an area of computer science that emphasizes the creation of intelligent machine that work and reacts like humans.

44. What are the various areas where AI (Artificial Intelligence) can be used?

Artificial Intelligence can be used in many areas like Computing, Speech recognition, Bio-informatics, Humanoid robot, Computer software, Space and Aeronautics' etc.