SRM VALLIAMMAI ENGINEERINGCOLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur-603203.

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB MANUAL

AD3565 & MACHINE LEARNING LABORATORY

(V semester)

Regulation 2023

Academic Year 2025-2026 (Odd Semester)

Prepared By

Dr.S.Sandhya, Assistant Professor (Sr.G)/IT
Mr.K.Sivakumar, Assistant Professor (Sr.G)/ IT

AD3565 MACHINE LEARNING LABORATORY L T P C

0 0 3 1.5

OBJECTIVES:

- To understand the data sets and apply suitable algorithms for selecting the appropriate features for analysis.
- To learn to implement supervised machine learning algorithms on standard datasets and evaluate the performance.
- To experiment the unsupervised machine learning algorithms on standard datasets and evaluate the performance.
- To build the graph based learning models for standard data sets.

LIST OF EXPERIMENTS

1. For a given set of training data examples stored in a .CSV file, implement and demonstrate the **Candidate-Elimination algorithm** to output a description of the set of all hypotheses consistent with the training examples.
2. Write a program to demonstrate the working of the decision tree based **ID3 algorithm**. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.
3. Build an Artificial Neural Network by implementing the **Backpropagation algorithm** and test the same using appropriate data sets.
4. Write a program to implement the **naïve Bayesian classifier** for a sample training data set stored as a .CSV file and compute the accuracy with a few test data sets.
5. Implement **naïve Bayesian Classifier** model to classify a set of documents and measure the accuracy, precision, and recall.
6. Write a program to construct a **Bayesian network to** diagnose CORONA infection using standard WHO Data Set.
7. Apply **EM algorithm** to cluster a set of data stored in a .CSV file. Use the same data set for clustering using the k-Means **algorithm**. Compare the results of these two algorithms.
8. Write a program to implement ***k*-Nearest Neighbour algorithm** to classify the iris data set. Print both correct and wrong predictions.
9. Implement the non-parametric **Locally Weighted Regression algorithm** in order to fit data points. Select an appropriate data set for your experiment and draw graphs.

SOFTWARE:

1. Windows 7 or higher
2. The programs can be implemented in either Python or R.

TOTAL: 45 PERIODS

OUTCOMES:

At the end of this course, the students should be able to:

- Apply suitable algorithms for selecting the appropriate features for analysis.
- Implement supervised machine learning algorithms on standard datasets and evaluate the performance.
- Apply unsupervised machine learning algorithms on standard datasets and evaluate the performance.
- Build the graph based learning models for standard data sets.
- Assess and compare the performance of different ML algorithms and select the suitable one based on the application.

CO – PO – PSO Mapping

CO	PO												PSO			
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4
1	2	2	2	1	-	-	-	-	-	2	3	3	3	2	-	-
2	2	1	1	3	2	-	-	-	3	2	3	2	3	-	-	-
3	2	2	1	1	2	-	-	-	-	-	-	-	2	3	-	3
4	2	2	3	3	2	-	-	-	-	2	3	-	-	2	-	2
5	2	2	3	1	2	-	-	-	3	-	-	-	2	-	-	2
Avg	**2.0**	**1.8**	**2.0**	**1.4**	**2.0**	**-**	**-**	**-**	**3.0**	**2.0**	**3.0**	**2.5**	**2.5**	**2.3**	**-**	**2.3**

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

1. **PEO1:** To afford the necessary background in the field of Information Technology to deal with Engineering problems to excel as engineering professionals in industries.

2. **PEO2:** To improve the qualities like creativity, leadership, team work and skill thus contributing towards the growth and development of society.
3. **PEO3:** To develop ability among students towards innovation and entrepreneurship that caters to the need of Industry and society.
4. **PEO4:** To inculcate an attitude for life- long learning process through the use of information technology sources.
5. **PEO5:** To prepare them to be innovative and ethical leaders, both in their chosen profession and in other activities.

PROGRAM OUTCOMES (POs) ENGINEERING GRADUATES WILL BE ABLE TO:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **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.
3. **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.
4. **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.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **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.
7. **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.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **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.
11. **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.
12. **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 OBJECTIVES (PSOs)

PSO1: Design secured database applications involving planning, development and maintenance using state of the art methodologies based on ethical values.

PSO2: Design and develop solutions for modern business environments coherent with the advanced technologies and tools.

PSO3: Design, plan and setting up the network that is helpful for contemporary business environments using latest hardware components.

PSO4: Planning and defining test activities by preparing test cases that can predict and correct errors ensuring a socially transformed product catering all technological needs.

COURSE OUTCOMES:
Course Name: AD3566- MACHINE LEARNING LABORATORY

Year of study: 2025 –2026

AD3565.1	Apply suitable algorithms for selecting the appropriate features for analysis.
AD3565.2	Implement supervised machine learning algorithms on standard datasets and evaluate the performance
AD3565.3	Apply unsupervised machine learning algorithms on standard datasets and evaluate the performance.
AD3565.4	Build the graph based learning models for standard data sets
AD3565.5	Assess and compare the performance of different ML algorithms and select the suitable one based on the application.

CO-PO Matrix:

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
AD3565.1	2	2	2	1	-	-	-	-	-	2	3	3
AD3565.2	2	1	1	3	2	-	-	-	3	2	3	2
AD3565.3	2	2	1	1	2	-	-	-	-	-	-	-
AD3565.4	2	2	3	3	2	-	-	-	-	2	3	-
AD3565.5	2	2	3	1	2	-	-	-	3	-	-	-

Justification:

Course Outcome	PO	Value	Justification
AD3565.1	PO1	2	Applying the fundamental Engineering Knowledge to select suitable algorithm for analysis.
	PO2	2	Understanding and analyzing the problems associated in selecting appropriate feature for analysis using suitable algorithm.
	PO3	2	Able to develop solutions using suitable algorithm for problems identified.
	PO4	1	Applying research based knowledge for analysis and interpretation of data and developing solution by selecting suitable algorithm for data analysis.
	PO10	2	Ability to comprehend and write report ,design documentation by selecting suitable machine learning algorithm.
	PO11	3	Ability to manage projects in a multidisciplinary environment by usage of correct algorithm for selecting features for analysis.

	PO12	3	Ability to learn and adapt to the dynamic changes in handling data using appropriate algorithm.
AD3565.2	PO1	2	Applying the fundamental concepts of engineering to implement supervised machine learning algorithm on dataset to evaluate performance.
	PO2	1	Ability to understand and analyze the problems associated with the dataset and perform analysis .
	PO3	1	Learning to design solution to complex engineering problems by implementing supervised machine learning algorithm.
	PO4	3	Applying research based knowledge for interpretation of data by implementing supervised machine learning algorithm.
	PO5	2	Learning to implement supervised machine learning algorithm by selecting and applying appropriate tools for prediction and modelling.
	PO9	3	Understanding the importance of data exchange in multidisciplinary setting and implementation of supervised machine learning algorithm for developing solution.
	PO10	2	Learning to communicate effectively in implementing supervised ML algorithm and write reports and design documentation.
	PO11	3	Learn to apply the engineering knowledge in implementing supervised ML algorithm for project management.
	PO12	2	Understanding the importance of life long learning in evaluating the performance of various supervised ML algorithms.
AD3565.3	PO1	2	Applying appropriate unsupervised ML algorithm for solving engineering problems with machine learning method.
	PO2	2	Identify and analyze the problems associated with data analysis with help of un supervised machine learning techniques
	PO3	1	Design and develop solution for complex engineering problems using unsupervised ML algorithm and evaluate their performance.
	PO4	1	Understanding the importance of data interpretation using research based knowledge and applying the suitable unsupervised ML algorithm for analysis.
	PO5	2	Learning and adapting with performance change of various machine learning methods with appropriate tools during data analysis
AD3565.4	PO1	2	Applying the fundamental knowledge for analysis and developing graph based learning models for different data sets.
	PO2	2	Identify the problem associated in data interpretation with help of graph based learning model for data analysis.
	PO3	3	Select and apply appropriate graph based learning model to develop solutions to complex engineering problems.
	PO4	3	Learning to analyze and interpret data using research based knowledge and through graph based learning model.
	PO5	2	Ability to make use of modern tools available to build graph based learning model for analysis of standard data set.
	PO10	2	Understanding the importance of effective communication among different groups during complex activities and in developing learning models.

	PO11	3	Adapting to the technological changes ,learning it and building graph based learning model for analysis and interpretation.
AD3565.5	PO1	2	Apply the fundamental knowledge to compare and select appropriate ML techniques to analyze Data
	PO2	2	Understand the problem by application of basic principles of engineering and select the appropriate ML algorithm after performance evaluation.
	PO3	3	Finding solutions by evaluating the different ML algorithm and choosing the appropriate technique to analyze Data in solving complex problems.
	PO4	1	Investigating the problem and choosing the appropriate ML algorithm by comparison of performance evaluation .
	PO5	2	Make use of modern tools for comparing and evaluating the various ML algorithms for analyzing the performance.
	PO9	3	Adapting to the advancement in tools usage and techniques used for analysis of data and interpretation.

8. CO-PSO Matrix:

CO	PSO1	PSO2	PSO3	PSO4
AD3566.1	3	2	-	-
AD3566.2	3	-	-	-
AD3566.3	2	3	-	3
AD3566.4	-	2	-	2
AD3566.5	2	-	-	2

Justification:

Course Outcome	Program Specific Outcome	Value	Justification
AD3565.1	PSO1	**3**	Design and develop suitable algorithm for analysis of secured database application with analytical approach.
	PSO2	**2**	Design and develop Intelligent systems for problem solving by applying suitable algorithm for feature selection.
AD3565.2	PSO1	**3**	Design and develop database applications using supervised machine learning models for performance evaluation.
AD3565.3	PSO1	**2**	Understand and develop database application to perform data analysis with unsupervised machine learning method for knowledge extraction.
	PSO2	**3**	Apply unsupervised machine learning model for development of intelligent systems to solve real time problems using tools and advanced technologies.

	PSO3	**3**	Learn to set up a network for contemporary business environment using unsupervised machine learning algorithm for data analysis.
AD3565.4	PSO2	**2**	Apply graph based learning model to develop intelligent system for extraction of knowledge and analysis.
	PSO4	**2**	Plan and use appropriate graph based learning models for developing contemporary business environments by using graphical data analysis and prepare test cases to predict results
AD3565.5	PSO1	**2**	Compare the performance of ML algorithm and choose an appropriate one to design and develop secure database application for the purpose of data analysis.
	PSO4	**2**	Plan, test and find the appropriate ML techniques to develop a product catering all technological needs.

CO-PO and PSO Mapping

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
AD3565	**2.0**	**1.8**	**2.0**	**1.4**	**2.0**	**-**	**-**	**-**	**3.0**	**2.0**	**3.0**	**2.5**	**2.5**	**2.3**	**-**	**2.3**

ASSESSMENT METHOD

MARK SPLIT UP	
AIM & PRE LAB VIVA QUESTIONS	**20**
OBSERVATION	**30**
CONDUCTION & EXECUTION	**30**
OUTPUT & RESULT	**10**
POST LAB VIVA QUESTIONS	**10**
TOTAL	**100**

TABLE OF CONTENTS

Ex. No: 1 **CANDIDATE-ELIMINATION ALGORITHM**

AIM:

1. To implement the Candidate-Elimination algorithm to determine the version space given training examples in a CSV file.
2. To output the set of all hypotheses consistent with the observed data..

Candidate-Elimination Algorithm

1. **Initialize**:
   - S ← the most specific hypothesis in H
   - G ← the most general hypothesis in H
2. **For each training example**:
   - If the example is **positive**:
     - Remove from G any hypothesis inconsistent with the example
     - For each hypothesis s in S that is not consistent with the example:
       - Remove s from S
       - Add to S all minimal generalizations of s that are consistent with the example and more specific than some hypothesis in G
     - Remove from S any hypothesis more general than another hypothesis in S
   - If the example is **negative**:
     - Remove from S any hypothesis inconsistent with the example
     - For each hypothesis g in G that is not consistent with the example:
       - Remove g from G
       - Add to G all minimal specializations of g that are consistent with the example and more general than some hypothesis in S
     - Remove from G any hypothesis less general than another hypothesis in G
3. **Output**:
   - The version space: All hypotheses between S and G (i.e., more general than S, more specific than G)

Source Code

```
import numpy as np
import pandas as pd
# Loading Data from a CSV File
data = pd.DataFrame(data=pd.read_csv('trainingdata.csv'))
print(data)
# Separating concept features from Target
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
# Isolating target into a separate DataFrame
# copying last column to target array
target = np.array(data.iloc[:,-1])
print(target)
def learn(concepts, target):
    '''
    learn() function implements the learning method of the Candidate elimination algorithm.
    Arguments:
        concepts - a data frame with all the features
        target - a data frame with corresponding output values
    '''
    # Initialise S0 with the first instance from concepts
    # .copy() makes sure a new list is created instead of just pointing to the same memory location
    specific_h = concepts[0].copy()
    print("\nInitialization of specific_h and general_h")
    print(specific_h)
    #h=["#" for i in range(0,5)]
    #print(h)
    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_h))]
    print(general_h)
    # The learning iterations
    for i, h in enumerate(concepts):
```

```
        # Checking if the hypothesis has a positive target
        if target[i] == "Yes":
            for x in range(len(specific_h)):
                # Change values in S & G only if values change
                if h[x] != specific_h[x]:
                    specific_h[x] = '?'
                    general_h[x][x] = '?'
        # Checking if the hypothesis has a positive target
        if target[i] == "No":
            for x in range(len(specific_h)):
                # For negative hyposthesis change values only  in G
                if h[x] != specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'
        print("\nSteps of Candidate Elimination Algorithm",i+1)
        print(specific_h)
        print(general_h)


    # find indices where we have empty rows, meaning those that are unchanged
    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
    for i in indices:
        # remove those rows from general_h
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    # Return final values
    return specific_h, general_h
s_final, g_final = learn(concepts, target)
print("\nFinal Specific_h:", s_final, sep="\n")
print("\nFinal General_h:", g_final, sep="\n")
```

Output

```
sky airTemp humidity    wind water forecast enjoySport
0  Sunny    Warm   Normal  Strong  Warm     Same       Yes
```

1 Sunny Warm High Strong Warm Same Yes

2 Rainy Cold High Strong Warm Change No

3 Sunny Warm High Strong Cool Change Yes

[['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']

['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']

['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']

['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]

['Yes' 'Yes' 'No' 'Yes']

Initialization of specific_h and general_h

['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']

[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Steps of Candidate Elimination Algorithm 1

['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']

[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Steps of Candidate Elimination Algorithm 2

['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']

[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Steps of Candidate Elimination Algorithm 3

['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']

[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'Same']]

Steps of Candidate Elimination Algorithm 4

['Sunny' 'Warm' '?' 'Strong' '?' '?']

[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Final Specific_h:

['Sunny' 'Warm' '?' 'Strong' '?' '?']

Final General_h:

[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]

VIVA QUESTIONS

1. What is the version space?
2. What are the S and G boundaries?
3. How does the Candidate-Elimination algorithm work?
4. What assumptions does Candidate-Elimination make?
5. How does it handle noisy data?
6. Can it be used for continuous attributes?
7. What is the role of generalization and specialization?
8. What happens if no hypothesis is consistent with the training data?
9. Can this algorithm work on continuous data?
10. What is the output of this algorithm?

RESULT:

Thus the Candidate –Elimination algorithm for the given dataset have done and output is verified.

Ex. No: 2 **DECISION TREE BASED ID3 ALGORITHM**

AIM:

1. To implement the ID3 algorithm for constructing a decision tree using a given dataset.
2. To classify a new sample based on the constructed decision tree.

ID3 (Iterative Dichotomiser 3) Algorithm

1. **Input**:
   - A dataset D with attributes and class labels.
2. **If**:
   - All examples in D belong to the same class → return a leaf node with that class.
   - The attribute list is empty → return a leaf node with the majority class in D.
3. **Otherwise**:
   - Select the best attribute A using **Information Gain**.
   - Create a decision node that splits on A.
4. **For each value** of attribute A:
   - Partition the dataset D into subsets where A = value.
   - Recursively apply ID3 to each subset.
5. **Output**:
   - A decision tree that classifies new instances by following attribute tests from root to leaf.

Source Code

```
import numpy as np
import math
import csv
def read_data(filename):
    with open(filename, 'r') as csvfile:
        datareader = csv.reader(csvfile, delimiter=',')
        headers = next(datareader)
        metadata = []
        traindata = []
        for name in headers:
            metadata.append(name)
```

```
        for row in datareader:
            traindata.append(row)

    return (metadata, traindata)

class Node:
    def __init__(self, attribute):
        self.attribute = attribute
        self.children = []
        self.answer = ""

    def __str__(self):
        return self.attribute

def subtables(data, col, delete):
    dict = {}
    items = np.unique(data[:, col])
    count = np.zeros((items.shape[0], 1), dtype=np.int32)

    for x in range(items.shape[0]):
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                count[x] += 1

    for x in range(items.shape[0]):
        dict[items[x]] = np.empty((int(count[x]), data.shape[1]), dtype="|S32")
        pos = 0
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                dict[items[x]][pos] = data[y]
                pos += 1
```

```
        if delete:
            dict[items[x]] = np.delete(dict[items[x]], col, 1)

    return items, dict


def entropy(S):
    items = np.unique(S)

    if items.size == 1:
        return 0

    counts = np.zeros((items.shape[0], 1))
    sums = 0

    for x in range(items.shape[0]):
        counts[x] = sum(S == items[x]) / (S.size * 1.0)

    for count in counts:
        sums += -1 * count * math.log(count, 2)
    return sums


def gain_ratio(data, col):
    items, dict = subtables(data, col, delete=False)

    total_size = data.shape[0]
    entropies = np.zeros((items.shape[0], 1))
    intrinsic = np.zeros((items.shape[0], 1))

    for x in range(items.shape[0]):
        ratio = dict[items[x]].shape[0]/(total_size * 1.0)
        entropies[x] = ratio * entropy(dict[items[x]][:, -1])
```

```
        intrinsic[x] = ratio * math.log(ratio, 2)

    total_entropy = entropy(data[:, -1])
    iv = -1 * sum(intrinsic)

    for x in range(entropies.shape[0]):
        total_entropy -= entropies[x]

    return total_entropy / iv

def create_node(data, metadata):
    if (np.unique(data[:, -1])).shape[0] == 1:
        node = Node("")
        node.answer = np.unique(data[:, -1])[0]
        return node

    gains = np.zeros((data.shape[1] - 1, 1))

    for col in range(data.shape[1] - 1):
        gains[col] = gain_ratio(data, col)

    split = np.argmax(gains)

    node = Node(metadata[split])
    metadata = np.delete(metadata, split, 0)

    items, dict = subtables(data, split, delete=True)

    for x in range(items.shape[0]):
        child = create_node(dict[items[x]], metadata)
        node.children.append((items[x], child))
```

```
    return node


def empty(size):
    s = ""
    for x in range(size):
        s += "   "
    return s


def print_tree(node, level):
    if node.answer != "":
        print(empty(level), node.answer)
        return
    print(empty(level), node.attribute)
    for value, n in node.children:
        print(empty(level + 1), value)
        print_tree(n, level + 2)


metadata, traindata = read_data("tennisdata.csv")
data = np.array(traindata)
node = create_node(data, metadata)
print_tree(node, 0)
```

Output

```
Outlook
    Overcast
      b'Yes'
    Rainy
      Windy
        b'False'
          b'Yes'
        b'True'
```

b'No'

Sunny

Humidity

b'High'

b'No'

b'Normal'

b'Yes'

VIVA QUESTIONS

1. What is entropy?
2. What is information gain?
3. How does the ID3 algorithm choose attributes?
4. What kind of data does ID3 handle?
5. What are leaf and internal nodes in a decision tree?
6. What is the stopping condition for building a tree?
7. How do you handle missing values in ID3?
8. How is pruning used in decision trees?
9. What is the stopping criterion for ID3?
10. How is the final tree used to classify new data

RESULT:

Thus the **ID3 (Iterative Dichotomiser 3) Algorithm** for the given dataset have done and output is verified.

Ex. No: 3 ANN BY IMPLEMENTING THE BACKPROPAGATION ALGORITHM

Aim

To implement an Artificial Neural Network (ANN) using the Backpropagation algorithm. To train the ANN on a dataset and test its performance on new inputs.

Backpropagation Algorithm

1. **Initialize weights and biases** randomly (usually small values).
2. **For each training example**:
    - **Forward Pass**:
        - Compute activations of all neurons layer by layer from input to output using:

$$a = f(w \cdot x + b)$$

where f is the activation function (e.g., sigmoid, ReLU).

- **Compute Error**:
    - Calculate the difference between actual output and predicted output.
    - Use a loss function like Mean Squared Error (MSE):

$$E = \frac{1}{2}(y_{\text{true}} - y_{\text{pred}})^2$$

3. **Backward Pass (Backpropagation)**:
    - Compute **error gradients** at output layer.
    - Propagate the error backwards through the network using the chain rule.
    - Compute gradients for hidden layers.
4. **Update Weights and Biases**:
    - Adjust weights and biases using gradient descent:

$$w = w - \eta \frac{\partial E}{\partial w}$$

where η is the learning rate.

5. **Repeat** steps 2–4 for a number of epochs or until the error is minimized.
6. **Test** the network on unseen data to evaluate accuracy.

Source Code

```
import numpy as np
X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)     # X = (hours sleeping, hours studying)
y = np.array(([92], [86], [89]), dtype=float)          # y = score on test
# scale units
X = X/np.amax(X, axis=0)       # maximum of X array
y = y/100                      # max test score is 100
class Neural_Network(object):
   def __init__(self):
                        # Parameters
      self.inputSize = 2
      self.outputSize = 1
      self.hiddenSize = 3
                        # Weights
      self.W1 = np.random.randn(self.inputSize, self.hiddenSize)       # (3x2) weight
matrix from input to hidden layer
      self.W2 = np.random.randn(self.hiddenSize, self.outputSize)      # (3x1) weight
matrix from hidden to output layer
   def forward(self, X):
                        #forward propagation through our network
      self.z = np.dot(X, self.W1)             # dot product of X (input) and first set of 3x2
weights
      self.z2 = self.sigmoid(self.z)          # activation function
      self.z3 = np.dot(self.z2, self.W2)      # dot product of hidden layer (z2) and second
set of 3x1 weights
      o = self.sigmoid(self.z3)               # final activation function
      return o

   def sigmoid(self, s):
      return 1/(1+np.exp(-s))     # activation function

   def sigmoidPrime(self, s):
      return s * (1 - s)          # derivative of sigmoid
```

```
    def backward(self, X, y, o):
                        # backward propgate through the network
        self.o_error = y - o        # error in output
        self.o_delta = self.o_error*self.sigmoidPrime(o) # applying derivative of sigmoid to
        self.z2_error = self.o_delta.dot(self.W2.T)    # z2 error: how much our hidden layer
weights contributed to output error
        self.z2_delta = self.z2_error*self.sigmoidPrime(self.z2) # applying derivative of
sigmoid to z2 error
        self.W1 += X.T.dot(self.z2_delta)       # adjusting first set (input --> hidden) weights
        self.W2 += self.z2.T.dot(self.o_delta)  # adjusting second set (hidden --> output)
weights

    def train (self, X, y):
        o = self.forward(X)
        self.backward(X, y, o)

NN = Neural_Network()
print ("\nInput: \n" + str(X))
print ("\nActual Output: \n" + str(y))
print ("\nPredicted Output: \n" + str(NN.forward(X)))
print ("\nLoss: \n" + str(np.mean(np.square(y - NN.forward(X))))     # mean sum squared
loss)
NN.train(X, y)
```

Output

```
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]

Actual Output:
[[0.92]
 [0.86]
 [0.89]]
```

Predicted Output:

[[0.67045559]

[0.66412418]

[0.64589341]]

Loss:

0.05340925983160877

VIVA QUESTIONS

1. What is a perceptron?
2. What is forward propagation?
3. What is the backpropagation algorithm?
4. What is the loss function used in backpropagation?
5. How do you update weights in a neural network?
6. What are activation functions? Name a few.
7. What is the difference between gradient descent and stochastic gradient descent?
8. What is an epoch?
9. What is vanishing gradient problem?
10. What are weights and biases?

RESULT:

Thus the **Backpropagation Neural Network** for the given dataset have done and output is verified.

Ex. No: 4 **NAÏVE BAYESIAN CLASSIFIER ALGORITHM**

Aim

To implement the Naïve Bayes classifier using a sample training dataset from a CSV file. To classify test data and compute the accuracy of the model.

Naïve Bayes Algorithm

1. **Input**:
    - A training dataset with feature values and class labels.
2. **Preprocessing**:
    - Convert categorical data to numerical form (if needed).
    - Split the data into **training** and **test** sets.
3. **Training (Model Building)**:
    - Calculate the **prior probability** for each class:

$$P(C) = \frac{\text{Number of samples in class } C}{\text{Total number of samples}}$$

    - For each class and feature, calculate the **likelihood** (probability of feature value given the class), assuming features are independent:

$$P(X_i|C) = \frac{\text{Count of } X_i \text{ in class } C}{\text{Total samples in class } C}$$

4. **Prediction (Classification)**:
    - For a given test instance, calculate:

$$P(C|X) \propto P(C) \cdot \prod_i P(X_i|C)$$

    - Choose the class with the highest posterior probability.
5. **Accuracy Calculation**:
    - Compare predicted labels with actual test labels.

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{Total test samples}} \times 100\%$$

Source Code

```
# import necessary libarities
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

# load data from CSV
data = pd.read_csv('tennisdata.csv')
print("THe first 5 values of data is :\n",data.head())

# obtain Train data and Train output
X = data.iloc[:,:-1]
print("\nThe First 5 values of train data is\n",X.head())

y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is\n",y.head())

# Convert then in numbers
le_outlook = LabelEncoder()
X.Outlook = le_outlook.fit_transform(X.Outlook)

le_Temperature = LabelEncoder()
X.Temperature = le_Temperature.fit_transform(X.Temperature)

le_Humidity = LabelEncoder()
X.Humidity = le_Humidity.fit_transform(X.Humidity)

le_Windy = LabelEncoder()
X.Windy = le_Windy.fit_transform(X.Windy)

print("\nNow the Train data is :\n",X.head())

le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
```

```
print("\nNow the Train output is\n",y)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20)

classifier = GaussianNB()
classifier.fit(X_train,y_train)

from sklearn.metrics import accuracy_score
print("Accuracy is:",accuracy_score(classifier.predict(X_test),y_test))
```

Output

THe first 5 values of data is :

	Outlook	Temperature	Humidity	Windy	PlayTennis
0	Sunny	Hot	High	False	No
1	Sunny	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Rainy	Mild	High	False	Yes
4	Rainy	Cool	Normal	False	Yes

The First 5 values of train data is

	Outlook	Temperature	Humidity	Windy
0	Sunny	Hot	High	False
1	Sunny	Hot	High	True
2	Overcast	Hot	High	False
3	Rainy	Mild	High	False
4	Rainy	Cool	Normal	False

The first 5 values of Train output is

0 No

1 No

2 Yes

3 Yes

4 Yes

Name: PlayTennis, dtype: object

Now the Train data is :

	Outlook	Temperature	Humidity	Windy
0	2	1	0	0
1	2	1	0	1
2	0	1	0	0
3	1	2	0	0
4	1	0	1	0

Now the Train output is

[0 0 1 1 1 0 1 0 1 1 1 1 1 0]

Accuracy is: 0.6666666666666666

VIVA QUESTIONS

1. What is Bayes' Theorem?
2. What assumptions are made in Naïve Bayes?
3. Why is it called "naïve"?
4. How does Naïve Bayes work with categorical vs. continuous data?
5. How do you calculate the posterior probability?
6. What are the limitations of Naïve Bayes?
7. Can Naïve Bayes handle multi-class classification?
8. What are the strengths of Naïve Bayes?
9. What is a confusion matrix?
10. What is the accuracy of Naïve Bayes based on?

RESULT:

Thus the **Naïve Bayes Classifier** for the given dataset have done and output is verified.

Ex. No: 5 NAÏVE BAYES FOR DOCUMENT CLASSIFICATION

Aim:

To implement N**aïve Bayesian Classifier** model to classify a set of documents and measure the accuracy, precision, and recall.

Algorithm:

1. Load Dataset:

- Read document.csv with columns message and label.

2. Preprocess:

- Map labels pos → 1, neg → 0.
- Split data into X (messages) and y (labels).

3. Split Dataset:

- Divide into training and testing sets: Xtrain, Xtest, ytrain, ytest.

4. Vectorize Text:

- Convert text into numerical format using CountVectorizer.

5. Train Model:

- Fit a MultinomialNB model on training data.

6. Predict:

- Predict labels for the test set.

7. Evaluate:

- Calculate accuracy, precision, recall, and confusion matrix.

8. Output:

- Display performance metrics and predictions.

Source Code

```
import pandas as pd
msg = pd.read_csv('document.csv', names=['message', 'label'])
print("Total Instances of Dataset: ", msg.shape[0])
msg['labelnum'] = msg.label.map({'pos': 1, 'neg': 0})

X = msg.message
y = msg.labelnum
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y)
```

```
from sklearn.feature_extraction.text import CountVectorizer

count_v = CountVectorizer()
Xtrain_dm = count_v.fit_transform(Xtrain)
Xtest_dm = count_v.transform(Xtest)

df = pd.DataFrame(Xtrain_dm.toarray(),columns=count_v.get_feature_names())
print(df[0:5])

from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(Xtrain_dm, ytrain)
pred = clf.predict(Xtest_dm)

for doc, p in zip(Xtrain, pred):
    p = 'pos' if p == 1 else 'neg'
    print("%s -> %s" % (doc, p))

from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score
print('Accuracy Metrics: \n')
print('Accuracy: ', accuracy_score(ytest, pred))
print('Recall: ', recall_score(ytest, pred))
print('Precision: ', precision_score(ytest, pred))
print('Confusion Matrix: \n', confusion_matrix(ytest, pred))
```

Output

```
Accuracy Metrics:
Accuracy:  0.8
Recall:  1.0
Precision:  0.5
Confusion Matrix:
 [[3 1]
 [0 1]]
```

VIVA QUESTIONS

1. How are text documents preprocessed?
2. What is a Bag-of-Words model?
3. What is TF-IDF?
4. What are the steps to classify text using Naïve Bayes?
5. How are precision and recall calculated in text classification?
6. What is n-gram representation?
7. What is stopword removal?
8. What is tokenization?
9. What is the difference between multinomial and Bernoulli Naïve Bayes?
10. What preprocessing steps are applied to text?

RESULT:

Thus the **Naïve Bayes for Document Classification for** the given dataset have done and output is verified.

Ex. No: 6 BAYESIAN NETWORK TO DIAGNOSE DISEASE

Aim:

To write a program to construct a **Bayesian network to** diagnose CORONA infection using standard WHO Data Set.

Algorithm

- **Define the Problem**:
  - Identify symptoms, risk factors, and the target condition (e.g., COVID-19).
- **Simulate or Obtain Dataset**:
  - Create or obtain a dataset containing attributes:
    - Symptoms (e.g., Fever, Cough, Fatigue).
    - Risk factors (e.g., Travel History, Age, Pre-existing Conditions).
    - Target condition (e.g., Disease Presence: Yes/No).
  - If simulating, define basic rules for relationships between attributes and the target.
- **Preprocess Data**:
  - Format the dataset appropriately for training:
    - Encode categorical variables into numerical values if needed.
    - Normalize or scale data if required.
- **Define the Bayesian Network**:
  - Identify relationships between variables.
  - Create directed edges to represent conditional dependencies:
    - For example, symptoms and risk factors pointing to the disease.
- **Train the Bayesian Network**:
  - Use a library like `pgmpy` to create the Bayesian network.
  - Fit the model parameters using the dataset with:
    - Maximum Likelihood Estimation (MLE).
    - Bayesian Estimation (if prior probabilities are known).
- **Perform Inference**:
  - Use an inference algorithm (e.g., Variable Elimination) to:
    - Query the probability of the disease given specific evidence (e.g., symptoms observed).
    - Compute marginal probabilities for diagnostic decisions.
- **Evaluate and Validate**:
  - Test the model with known cases to check its predictions.
  - Calculate metrics such as accuracy, precision, and recall if ground truth is available.

- **Output Results**:
  - Display the structure of the Bayesian network.
  - Show probabilities for the target condition based on provided evidence.
  - Save or visualize the dataset and network structure.

Source Code

```
import pandas as pd
import numpy as np
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination

# Simulating a dataset based on WHO COVID-19 guidelines
np.random.seed(42)  # For reproducibility
data_size = 1000
data = pd.DataFrame({
    "Fever": np.random.choice([0, 1], size=data_size, p=[0.3, 0.7]),
    "Cough": np.random.choice([0, 1], size=data_size, p=[0.4, 0.6]),
    "Fatigue": np.random.choice([0, 1], size=data_size, p=[0.5, 0.5]),
    "Travel_History": np.random.choice([0, 1], size=data_size, p=[0.8, 0.2]),
    "COVID19": None,  # Placeholder
})

# Defining conditional probabilities
# Using simple rules for simulated data
data["COVID19"] = np.where(
    (data["Fever"] & data["Cough"] & (data["Travel_History"] | data["Fatigue"])),
    1,
    0,
)

# Save dataset for inspection
data.to_csv("covid19_dataset.csv", index=False)
```

```
# Bayesian Network
model = BayesianNetwork([
    ("Fever", "COVID19"),
    ("Cough", "COVID19"),
    ("Travel_History", "COVID19"),
    ("Fatigue", "COVID19"),
])

# Fit the model
model.fit(data, estimator=MaximumLikelihoodEstimator)

# Print the structure
print("Bayesian Network Structure:")
print(model.edges())

# Perform inference
inference = VariableElimination(model)

# Example: Querying the network for COVID-19 probabilities given evidence
query_result = inference.query(variables=["COVID19"], evidence={"Fever": 1, "Cough": 1, "Travel_History": 1})
print("\nCOVID-19 Diagnosis Probability:")
print(query_result)
```

Output

```
Bayesian Network Structure:
[('Fever', 'COVID19'), ('Cough', 'COVID19'), ('Travel_History', 'COVID19'), ('Fatigue', 'COVID19')]
COVID-19 Diagnosis Probability:
+-----------+-------------+
| COVID19   |   phi(COVID19) |
+===========+=============+
```

| COVID19(0)| 0.2526 |

+----------+----------+

| COVID19(1)| 0.7474 |

+----------+----------+

VIVA QUESTIONS

1. What is a Bayesian Network?
2. What is a Conditional Probability Table (CPT)?
3. How is inference done in Bayesian Networks?
4. Why are Bayesian Networks useful in medical diagnosis?
5. What is d-separation?
6. Can you learn Bayesian networks from data?
7. What is the difference between Bayesian Networks and Naïve Bayes?
8. How do you construct a Bayesian network?
9. What are real-world applications of Bayesian Networks?
10. Define nodes and edges in a Bayesian Network

RESULT:

Thus the **Bayesian Network to diagnose disease** for the given dataset have done and output is verified.

Ex. No: 7 **EM ALGORITHM**

Aim

To apply **EM algorithm** to cluster a set of data stored in a .CSV file and use the same data set for clustering using the k-Means **algorithm**.

Algorithm

1. **Input**:
   - Load the dataset (e.g., Iris dataset).
   - Dataset includes features (XXX) and true class labels (yyy).

2. **Data Preparation**:
   - Convert the dataset into a DataFrame for easy manipulation.
   - Assign meaningful column names for features.

3. **Visualization Setup**:
   - Define a colormap to represent classes or clusters.
   - Prepare plots for comparison:
     - Real classification (ground truth).
     - KMeans clustering results.
     - GMM clustering results.

4. **Ground Truth Visualization**:
   - Plot true class labels using two selected features (e.g., `Petal_Length` vs. `Petal_Width`) for visualization.

5. **KMeans Clustering**:
   - Initialize the KMeans model with k=3k=3k=3 (number of clusters).
   - Train the model on the dataset.
   - Obtain cluster assignments for each data point.
   - Reassign predicted cluster labels to align with true class labels (optional using `np.choose`).
   - Plot the results with cluster colors.

6. **GMM Clustering**:
   - Standardize the dataset using `StandardScaler` for better GMM performance.
   - Initialize and fit the Gaussian Mixture Model with n=3n=3n=3.
   - Predict cluster assignments for each data point.
   - Plot the results with cluster colors.

7. **Comparison and Analysis**:
   - Compare the visualizations of KMeans and GMM clustering results with the ground truth.
   - Analyze differences in clustering behavior:
     - KMeans forms spherical clusters.
     - GMM allows for elliptical cluster shapes, making it more flexible.

8. **Output**:
   - Three subplots:

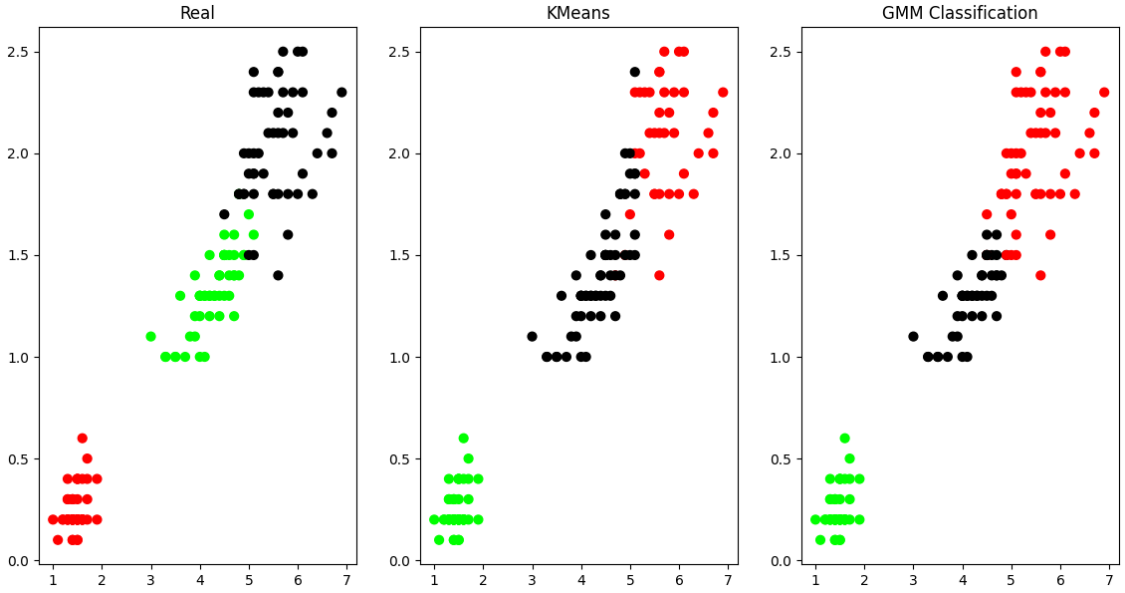
1. Ground truth labels.
2. KMeans clustering results.
3. GMM clustering results.

Source Code

```
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.mixture import GaussianMixture
from sklearn.datasets import load_iris
import sklearn.metrics as sm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset=load_iris()
# print(dataset)
X=pd.DataFrame(dataset.data)
X.columns=['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y=pd.DataFrame(dataset.target)
y.columns=['Targets']
# print(X)
plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])
# REAL PLOT
plt.subplot(1,3,1)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y.Targets],s=40)
plt.title('Real')
# K-PLOT
plt.subplot(1,3,2)
model=KMeans(n_clusters=3)
model.fit(X)
predY=np.choose(model.labels_,[0,1,2]).astype(np.int64)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[predY],s=40)
plt.title('KMeans')
```

```
# GMM PLOT
scaler=preprocessing.StandardScaler()
scaler.fit(X)
xsa=scaler.transform(X)
xs=pd.DataFrame(xsa,columns=X.columns)
gmm=GaussianMixture(n_components=3)
gmm.fit(xs)

y_cluster_gmm=gmm.predict(xs)
plt.subplot(1,3,3)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm],s=40)
plt.title('GMM Classification')
```

Output



VIVA QUESTIONS

1. What is the difference between EM and K-Means?
2. What are the steps of the EM algorithm?
3. How is a cluster evaluated?
4. What is a Gaussian Mixture Model?
5. How does K-Means initialize centroids?
6. What are the convergence criteria for both?
7. What is the Expectation step?
8. What is the Maximization step?
9. What assumptions does EM make?
10. How do you choose number of clusters (K)?

RESULT:

Thus the **Naïve Bayes for Document Classification** for the given dataset have done and output is verified.

Ex. No: 8 ***K*-NEAREST NEIGHBOUR ALGORITHM**

Aim

To write a program to implement ***k*-Nearest Neighbour Algorithm** to classify the iris data set.

Algorithm

- **Load Dataset**:
  - Import the dataset suitable for classification (e.g., Iris dataset in this case).
- **Split Dataset**:
  - Divide the dataset into training and testing sets:
    - Feature data (XXX): Input features of the dataset.
    - Target labels (yyy): Corresponding class labels.
- **Initialize the Model**:
  - Choose the kkk-value for the KNN classifier (e.g., k=1k=1k=1).
- **Train the Model**:
  - Fit the KNN model on the training data (XtrainX_{\text{train}}Xtrain, ytrainy_{\text{train}}ytrain).
- **Make Predictions**:
  - For each test instance (XtestX_{\text{test}}Xtest):
    - Extract the instance as input (xxx).
    - Reshape into the correct format (if needed).
    - Predict the class label using the trained model.
- **Compare Predictions**:
  - Compare the predicted label with the actual label for each test instance.
  - Optionally, map numerical class labels to class names for interpretability.
- **Evaluate the Model**:
  - Compute the model's accuracy on the test data:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

- **Output:**
  - Display:
    - Actual vs. predicted class labels for each test instance.
    - Overall model accuracy.

Source Code

```
from sklearn.datasets import load_iris

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split
```

```
import numpy as np

dataset=load_iris()
#print(dataset)
X_train,X_test,y_train,y_test=train_test_split(dataset["data"],dataset["target"],random_state=
0)

kn=KNeighborsClassifier(n_neighbors=1)
kn.fit(X_train,y_train)

for i in range(len(X_test)):
    x=X_test[i]
    x_new=np.array([x])
    prediction=kn.predict(x_new)
    print("TARGET=",y_test[i],dataset["target_names"][y_test[i]],"PREDICTED=",prediction
,dataset["target_names"][prediction])
print(kn.score(X_test,y_test))
```

Output

```
TARGET= 2 virginica PREDICTED= [2] ['virginica']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 2 virginica PREDICTED= [2] ['virginica']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 2 virginica PREDICTED= [2] ['virginica']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 2 virginica PREDICTED= [2] ['virginica']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 2 virginica PREDICTED= [2] ['virginica']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 2 virginica PREDICTED= [2] ['virginica']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
```

```
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 2 virginica PREDICTED= [2] ['virginica']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 2 virginica PREDICTED= [2] ['virginica']
TARGET= 2 virginica PREDICTED= [2] ['virginica']
TARGET= 1 versicolor PREDICTED= [1] ['versicolor']
TARGET= 0 setosa PREDICTED= [0] ['setosa']
TARGET= 1 versicolor PREDICTED= [2] ['virginica']
0.9736842105263158
```

VIVA QUESTIONS

1. How does k-NN work?
2. What is the role of distance metrics in k-NN?
3. What is the effect of choosing different values of **k**?
4. What distance metrics are commonly used?
5. Is k-NN a lazy learner?
6. Can k-NN be used for regression?
7. What is the decision boundary in k-NN?
8. What are the drawbacks of k-NN?
9. What happens when k is too small or too large?
10. How do you evaluate k-NN performance?

RESULT:

Thus the **k-Nearest Neighbors Algorithm** for the given dataset have done and output is verified.

Ex. No:9 **LOCALLY WEIGHTED REGRESSION**

Aim

To implement the non-parametric **Locally Weighted Regression Algorithm** in order to fit data points.

Algorithm

1. **Input**:
   - x: Independent variable values.
   - y: Dependent variable values.
   - f: Smoothing parameter (fraction of data points considered for local fitting).
   - iterations: Number of robustness iterations.

2. **Calculate Local Bandwidth**:
   - Compute the local neighborhood size $(r = \lceil f \times n \rceil)$.
   - For each x_i calculate h_i , the distance to the ***r-th*** nearest point.

3. **Weight Calculation**:
   - Compute weights w_{ij} for each pair of points:
     - Based on the distance between x_i and x_j , normalized by h_i .
     - Use the tricube weighting function:

$$w_{ij} = (1 - \left(\frac{|x_j - x_i|}{h_i}\right)^3)^3 \text{ for } |x_j - x_i| \leq h_i$$

4. **Initial Fit**:
   - For each x_i , solve a weighted linear regression problem to estimate $\hat{y}_i$:
     - Solve for coefficients β=[β0,β1] where:

$$A = \begin{bmatrix} \sum w_{ij} & \sum w_{ij}x_j \\ \sum w_{ij}x_j & \sum w_{ij}x_j^2 \end{bmatrix}, \quad b = \begin{bmatrix} \sum w_{ij}y_j \\ \sum w_{ij}y_j x_j \end{bmatrix}$$

5. **Robustness Iterations**:
   - Compute residuals $r_i = y_i - \hat{y}_i$
   - Calculate a robust weight δi using the median absolute deviation:

$$\delta_i = \left(1 - \left(\frac{r_i}{6s}\right)^2\right)^2, \quad s = \text{median}(|r_i|)$$

   - Update weights as $w_{ij} \times \delta_i$ and repeat the fitting process.

6. **Output**:

- Smoothed values $\hat{y}$

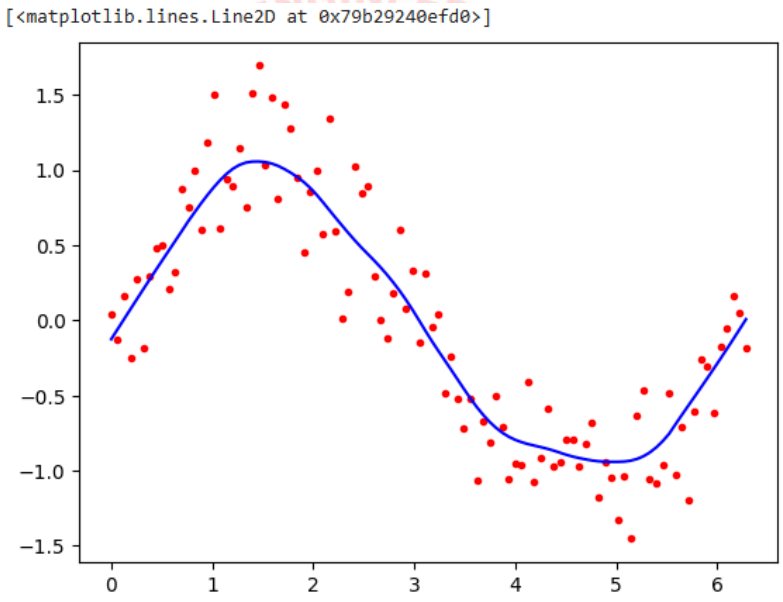
7. **Visualization (Optional)**:
   - Plot the original data points **(x, y)** and the smoothed curve **(x, $\hat{y}$)**

Source Code

```
from math import ceil
import numpy as np
from scipy import linalg

def lowess(x, y, f, iterations):
   n = len(x)
   r = int(ceil(f * n))
   h = [np.sort(np.abs(x - x[i]))[r] for i in range(n)]
   w = np.clip(np.abs((x[:, None] - x[None, :]) / h), 0.0, 1.0)
   w = (1 - w ** 3) ** 3
   yest = np.zeros(n)
   delta = np.ones(n)
   for iteration in range(iterations):
      for i in range(n):
         weights = delta * w[:, i]
         b = np.array([np.sum(weights * y), np.sum(weights * y * x)])
         A = np.array([[np.sum(weights), np.sum(weights * x)],[np.sum(weights * x),
np.sum(weights * x * x)]])
         beta = linalg.solve(A, b)
         yest[i] = beta[0] + beta[1] * x[i]

      residuals = y - yest
      s = np.median(np.abs(residuals))
      delta = np.clip(residuals / (6.0 * s), -1, 1)
      delta = (1 - delta ** 2) ** 2

   return yest

import math
n = 100
```

```
x = np.linspace(0, 2 * math.pi, n)
y = np.sin(x) + 0.3 * np.random.randn(n)
f =0.25
iterations=3
yest = lowess(x, y, f, iterations)

import matplotlib.pyplot as plt
plt.plot(x,y,"r.")
plt.plot(x,yest,"b-")
```

Output



VIVA QUESTIONS

1. What is Locally Weighted Regression (LWR)?
2. How does LWR differ from linear regression?
3. What is the role of the kernel/bandwidth?
4. Why is it called "non-parametric"?
5. What are the advantages of LWR?
6. What kernel functions are used?
7. What does "locally" mean in this context?
8. How is the weight for each point calculated?
9. What is the bandwidth parameter?
10. How does LWR handle non-linearity?

RESULT:

Thus the **Locally Weighted Regression** for the given dataset have done and output is verified.

Ex. No:10

PRINCIPAL COMPONENT ANALYSIS (PCA) FOR DIMENSIONALITY REDUCTION

Aim:

To implement Principal Component Analysis (PCA) for dimensionality reduction, visualize the reduced data, and evaluate its effectiveness using a classification algorithm.

Algorithm:

1. **Input**:
   - Dataset with multiple features (XXX) and labels (yyy).
2. **Preprocessing**:
   - Standardize the dataset to ensure all features have equal importance.
3. **Compute Covariance Matrix**:
   - Calculate the covariance matrix of the standardized data.
4. **Eigen Decomposition**:
   - Compute eigenvalues and eigenvectors of the covariance matrix.
   - Select the top kkk eigenvectors corresponding to the largest eigenvalues.
5. **Project Data**:
   - Transform the original data to the lower-dimensional space using the selected eigenvectors.
6. **Visualization**:
   - Plot the transformed data in 2D or 3D for visualization.
7. **Classification**:
   - Use a classification algorithm (e.g., Logistic Regression or k-Nearest Neighbors) on the reduced data to evaluate PCA's effectiveness.

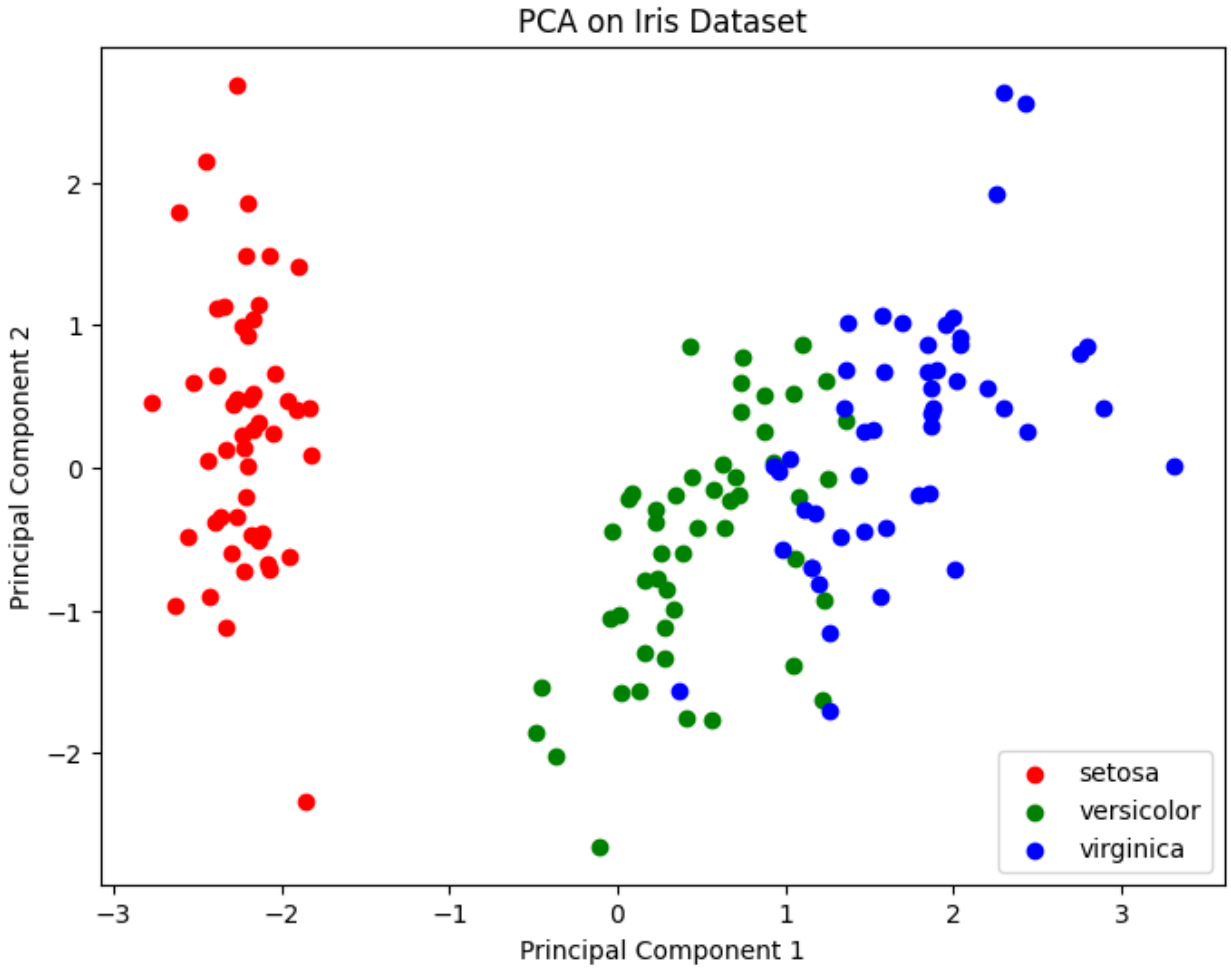
Source Code

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score



# Load Iris Dataset

from sklearn.datasets import load_iris
```

```
data = load_iris()

X = pd.DataFrame(data.data, columns=data.feature_names)

y = pd.DataFrame(data.target, columns=["Target"])

# Standardize the Data

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

# Apply PCA

pca = PCA(n_components=2)

X_pca = pca.fit_transform(X_scaled)

# Visualize Reduced Data

plt.figure(figsize=(8, 6))

for target, color in zip([0, 1, 2], ['red', 'green', 'blue']):

    plt.scatter(X_pca[y["Target"] == target, 0], X_pca[y["Target"] == target, 1], c=color,
label=data.target_names[target])

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('PCA on Iris Dataset')

plt.legend()

plt.show()

# Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(X_pca, y["Target"], test_size=0.3,
random_state=42)

# Apply k-Nearest Neighbors

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)

# Predictions and Accuracy

y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
```

```
print(f"Accuracy of k-NN on PCA-reduced data: {accuracy * 100:.2f}%")
```

Output



Viva Questions:

1. What is PCA, and why is it used?
2. What are eigenvalues and eigenvectors, and how do they relate to PCA?
3. Why do we standardize data before applying PCA?
4. What does the explained variance ratio indicate in PCA?
5. How is PCA different from Linear Discriminant Analysis (LDA)?
6. Can PCA be used for supervised learning? Why or why not?
7. What is the role of covariance in PCA?
8. What are the limitations of PCA?
9. How would you determine the optimal number of components in PCA?
10. How does PCA handle correlated features in a dataset?

RESULT:

Thus the **Principal Component Analysis (PCA) for dimensionality reduction** for the given dataset have done and output is verified.