Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

3) i) Problem statement :-

Develop a program to implement principal component analysis (PCA) for reducing the dimensionality of the iris dataset from 4 features to 2

ii) Objective :-

The purpose of this program is to reduce the dimensionality of the Iris dataset using PCA from 4 features to 2, making it easier to visualize and analyze the data while retaining most of the important information

iii) Algorithm :-

a) Import necessary libraries :- import numpy, pandas, matplotlib and sklearn libraries

b) Load Dataset : load the iris dataset using the load_iris() function

c) Prepare data : convert the dataset into a pandas DataFrame and add labels

d) Apply PCA : Reduce the dimensionality from 4 features to 2 using PCA

e) Create DataFrame : create a DataFrame for the reduced data with labels

f) Visualize Data : Plot the reduced data points with different colors for each class

g) Display plot : Show the plot with labels, grid and plot

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

iv) Source / Program Code

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

iris = load_iris()
data = iris.data
labels = iris.target
label_names = iris.target_names

iris_df = pd.DataFrame(data = iris.data, column = iris.
                                    - feature_names)

pca = PCA(n_components = 2)

data_reduced = pca.fit_transform(data)

reduced_df = pd.DataFrame(data = data_reduced,
                        - columns = ['PC1', 'PC2'])
reduced_df['labels'] = labels

plt.figure(figsize = (10,6))
colors = ['r', 'g', 'b']

for i, label in enumerate(np.unique(labels)):
        plt.scatter(
```

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

```
        reduced_df[reduced_df['labels'] == label]
                                        ['PC1'],
        reduced_df[reduced_df['labels'] == label]
                                        ['PC2'],

        c = colors[i],
        label = label_names[label]
    )
plt.title('PCA on ~~Iris~~ IRIS Dataset')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend()
plt.grid()
plt.show()
```

v) Compilation and Execution steps

* save the program in python file (eg. iris_pca.py)
* open a terminal or command prompt
* Navigate to the directory containing the python file
* Run the program using the command

```
python iris_pca.py
```

vi) Sample input and output

Sample input:

The program uses the built-in Iris dataset, so no external input is required.

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

Sample output:

A scatter plot showing the reduced Iris dataset with 2 principal components (PC1 and PC2) plotted against each other. Different colors represent different classes.

vii) Explaination of Output:

The plot shows the Iris dataset reduced to two principal components using PCA. Each class (setosa, Versicolor, Virginica) is represented by a different color. The plot helps in understanding how well the PCA has seperated the different classes.

viii) Observation and Analysis:

* PCA reduced the dataset from 4 dimension to 2 dimension effectively
* Setosa is well seperated from the other two classes
* Versicolor and Virginica have some overlapping points, indicating similarities between them

ix) Conclusion:

The program successfully implemented PCA to reduce the dimensionality of the Iris dataset. Visualization of the reduced data helps in identifying the seperation between different classes, which can be further used for classification tasks.

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

4) i) Problem Statement

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

ii) Objective :

The purpose of this program is to implement the Find-S algorithm for finding the most specific hypothesis from a given set of training data stored in a .csv file

iii) Algorithm :

* Import necessary libraries : Import the pandas library for handling the .csv file
* Define Find-S Algorithm Function : Implement the Find-S Algorithm as a function that reads data from a .csv file
* Initialize Hypothesis : Start with the most general hypothesis
* Update Hypothesis : For each positive example, refine the hypothesis by comparing attribute values
* Return Final Hypothesis : Output the most specific hypothesis consistent with all positive training example.

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

iv) Source / Program Code

```
import pandas as pd
def find_s_algorithm(file_path):
    data = pd.read_csv(file_path)

    print("Training data:")
    print(data)
    attributes = data.columns[:-1]
    class_label = data.columns[-1]
    hypothesis = ['?' for _ in attributes]

    for index, row in data.iterrows():
        if row[class_label] == 'Yes':
            for i, value in enumerate(row
                [attributes]):
                if hypothesis[i] == "?" or
                    hypothesis[i] == value:
                    hypothesis[i] = value
                else:
                    hypothesis[i] = '?'
    return hypothesis
file_path = 'training_data.csv'
hypothesis = find_s_algorithm(file_path)
print("In The final hypothesis is:", hypothesis)
```

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

v) Compilation and Execution steps

* prepare a CSV file (e.g training-data.csv) with training data examples
* Save the program in a python file (e.g find-s.py)
* Open a terminal or command prompt
* Navigate to the directory containing the python file and CSV file
* Run the program using the command

```
python find-s.py
```

vi) Sample Input and Output

Sample Input:
A .csv file containing training data examples with various attributes and a class label (Yes / NO)

Sample output:
The final hypothesis generated by the find-S algorithm, displayed as a list of attribute values or '?' indicating generalization

vii) Explaination of Output
The output shows the most specific hypothesis that covers all positive training examples. If a value differs between positive examples, the hypothesis will have a '?' at that position, indicating generalization.

Experiment No. : ..... Date: ..... Page No.: .....

Name of the Experiment : .....

viii) Observation and Analysis:

* The Find-S algorithm produces a hypothesis this is as specific as possible, covering only positive example
* It cannot handle noisy data or negative examples properly

xi) Conclusion:

The find-S algorithm was successfully implemented and tested with a .CSV file. the algorithm provides the most specific hypothesis that fits all positive training examples. However, it is sensitive to noise and incomplete data.