Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 01

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Installation of Python and Setting of Envirionment.

INTRODUCTION:

Python is a high-level, general-purpose, and very popular programming language used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry. It is being used by almost all tech-giant companies like Google, Amazon, Facebook, Instagram, Dropbox, Uber,etc.
The biggest strength of Python is huge collection of standard library which can be used for the following:
Machine Learning, GUI Applications (like Kivy, Tkinter, PyQt etc.), Web frameworks like Django (used by YouTube, Instagram, Dropbox), Image processing (like OpenCV, Pillow), Web scraping (like Scrapy, BeautifulSoup, Selenium), Test frameworks, Multimedia, Scientific computing,Text processing and many more.

Steps of Python Installation

1.Download the Python installer: Go to the official Python download page (https://www.python.org/downloads/) and download the latest version of Python for your operating system.
2.Install Python: Run the downloaded installer and follow the on-screen instructions to install Python.
3. Run the installer: Ensuring you check the "Add Python 3.x to PATH" option during installation. This makes Python accessible from any command prompt.
4.Verify the installation: Open a terminal or command prompt and type the following command:

```
python –version
```

Setup Environment for Python in VS Code

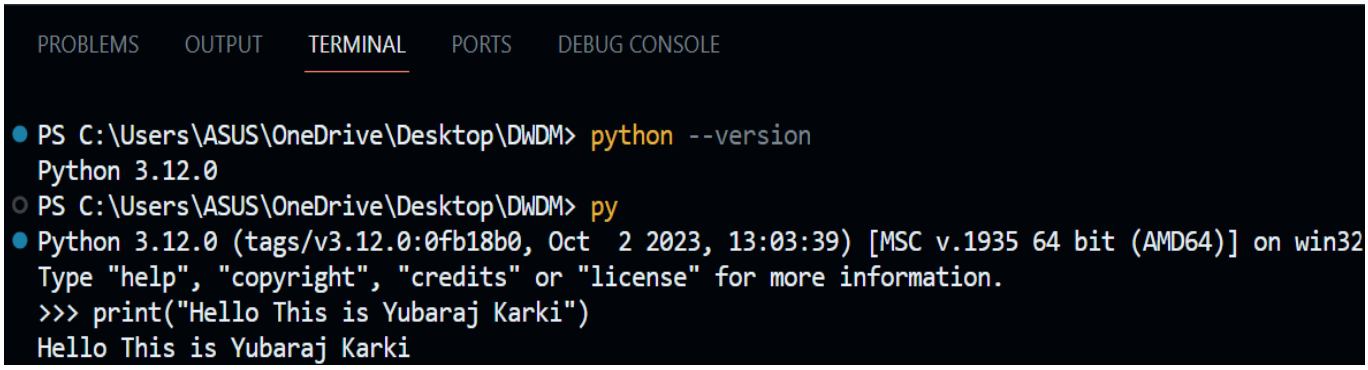
1.Install the Python extension for VS Code: Open VS Code and go to the Extensions tab. Search for "Python" and install the official Python extension by Microsoft.
2.Configure the Python interpreter: Open the Command Palette (Ctrl+Shift+P on Windows/Linux, Cmd+Shift+P on macOS) and type "Python: Select Interpreter". Select the Python interpreter that you want to use with VS Code.

Writing First Python Program in VS Code:

1.Create a new Python file: Open VS Code and create a new file with the .py extension

2.Write your Python code.

3.Run your Python program: To run your Python program, press F5 or go to the Run menu and select "Run Python File in Terminal" or We can execute a python program with the command ‘python filename’.



Pandas

Pandas is a powerful Python library for data manipulation and analysis. It provides data structures and operations for manipulating numerical tables and time series. Pandas is widely used in data science, machine learning, and financial analysis.

Jupyter Notebooks

It is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and explanatory text. It is a versatile and powerful tool for data science, machine learning, and scientific computing. It is easy to use, powerful, and extensible.

It is used for Data exploration and analysis, Machine learning model development, Scientific computing, Education and training, Data visualization.

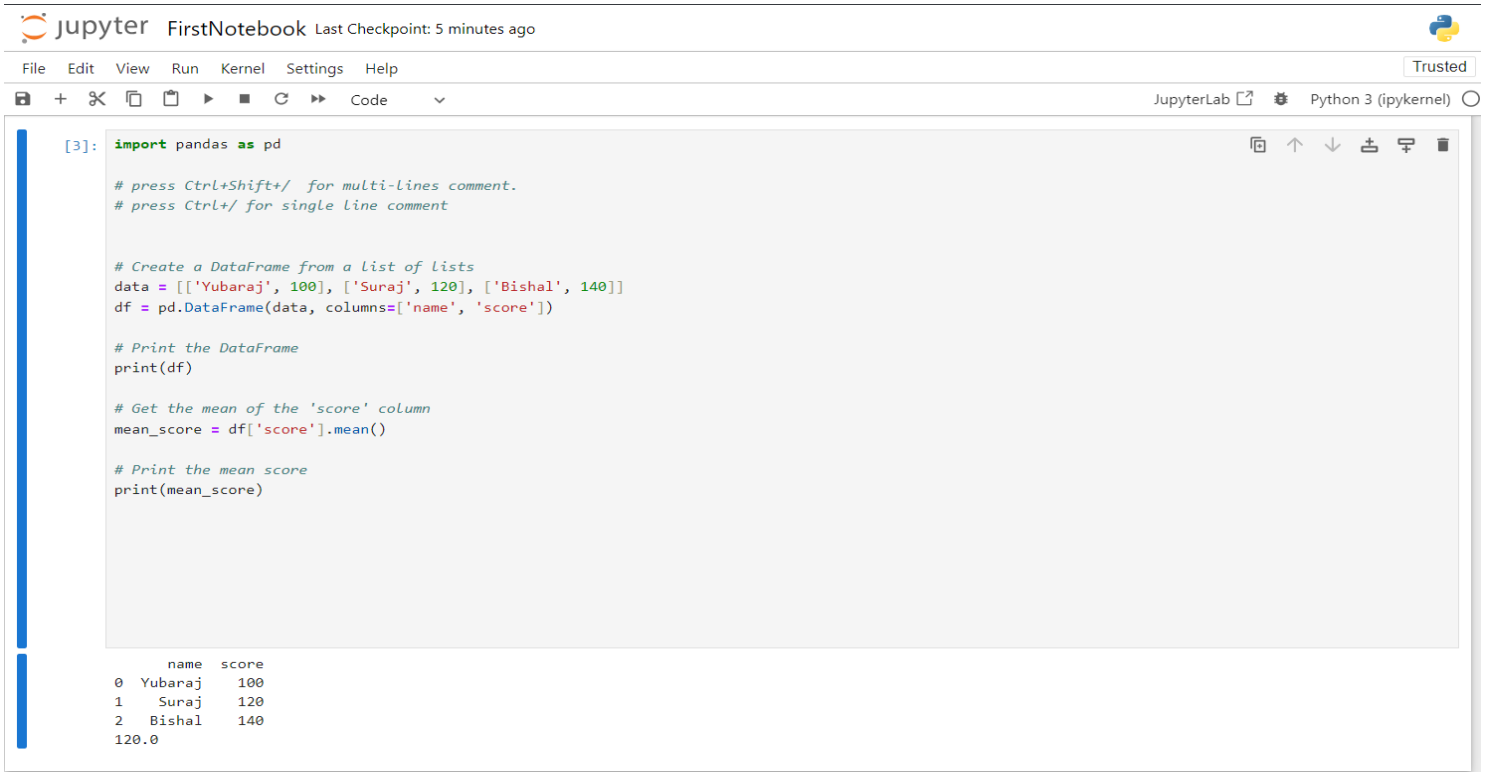
Steps to Install and Use

1. Install Python.
2. Install pandas using the terminal by typing pip install pandas.

3. Install pandas using the terminal by typing pip install jupyter.
4. Start Jupyter Notebooks: Open a terminal or command prompt and type the following command: jupyter notebook
5. Create a new notebook: Click on the "New" button in the Jupyter Notebooks interface and select "Python 3".
6. Write your first program: In the new notebook, type the following code: print("Hello, Yubaraj Karki!")
7. Run your program: Click on the "Run" button in the toolbar to run the program.

Example of how to use Jupyter Notebooks to create a DataFrame and perform some basic operations:

```
import pandas as pd
# press Ctrl+Shift+/  for multi-lines comment.
# press Ctrl+/ for single line comment

# Create a DataFrame from a list of lists
data = [[Yubaraj, 100], [Suraj, 120], [Bishal, 140]]
df = pd.DataFrame(data, columns=['name', 'score'])
# Print the DataFrame
print(df)
# Get the mean of the 'score' column
mean_score = df['score'].mean()
# Print the mean score
print(mean_score)
```



Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 02

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement Data Integration Technique.

INTRODUCTION

Data Integration is the process of combining data from multiple sources into a single, consistent data store. It gives a unified view. It is a technique of combining data from multiple sources, removing duplicate records, removing data conflicts, etc. There are a number of challenges associated with data integration such as data heterogeneity, data redundancy and data inconsistency. It is used to improve data quality, increase data accessibility, enhanced decision making, reduced cost and enhanced collaborations .

Methods of Data Integration

1. **Extracting, transforming, and loading (ETL):** This is a traditional method of data integration that involves extracting data from source systems, transforming it into a common format, and then loading it into a target system.

2. **Enterprise information integration (EII):** This is a more modern approach that uses middleware to connect different systems and applications. EII can be used to integrate data in real time or near real time.

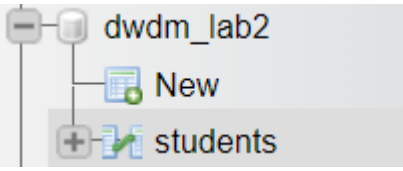
3. **Data virtualization:** This approach allows users to access data from multiple sources as if it were all stored in a single location. Data virtualization does not actually move or copy data, but rather provides a unified view of data from disparate sources.

Steps

1. Create a database with dummy data. Specify Table name with numbers of Column and column name.
2. Breakdown the table into 3 different part to export the data as CSV, Excel and SQL file.
3. Create an new Database.
4. Import the files that were exported previously in the new database.

Before Exporting

Databse: dwdm_lab2



Table

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

Name	Class	Section	RollNo	Address	PhoneNo
Aayush	10	A	1	Kathmandu	9801234561
Aarav	9	B	2	Bhaktapur	9801234562
Abhinav	8	C	3	Lalitpur	9801234563
Aditya	7	A	4	Kavre	9801234564
Ajay	6	B	5	Dhanusha	9801234565
Akash	10	C	6	Mahottarai	9801234566
Amit	9	A	7	Darchula	9801234567
Ankit	8	B	8	Mustang	9801234568
Bikram	7	C	9	Pyuthan	9801234569
Binod	6	A	10	Kaski	9801234570
Bishal	10	B	11	Kathmandu	9801234571
Chandra	9	C	12	Bhaktapur	9801234572
Deepak	8	A	13	Lalitpur	9801234573
Dipesh	7	B	14	Kavre	9801234574
Ganesh	6	C	15	Dhanusha	9801234575
Hari	10	A	16	Mahottarai	9801234576
Indra	9	B	17	Darchula	9801234577
Janak	8	C	18	Mustang	9801234578
Kamal	7	A	19	Pyuthan	9801234579
Krishna	6	B	20	Kaski	9801234580

After Integrations

Database: integrateddb

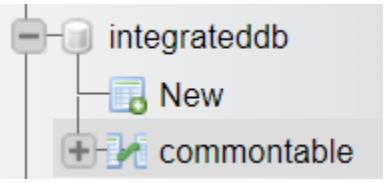


Table: commontable

Name	Class	Section	RollNo	Address	PhoneNo
Ajay	6	B	5	Dhanusha	9801234565
Akash	10	C	6	Mahottarai	9801234566
Amit	9	A	7	Darchula	9801234567
Ankit	8	B	8	Mustang	9801234568
Bikram	7	C	9	Pyuthan	9801234569
Binod	6	A	10	Kaski	9801234570
Bishal	10	B	11	Kathmandu	9801234571
Chandra	9	C	12	Bhaktapur	9801234572
Deepak	8	A	13	Lalitpur	9801234573
Dipesh	7	B	14	Kavre	9801234574
Ganesh	6	C	15	Dhanusha	9801234575
Hari	10	A	16	Mahottarai	9801234576
Indra	9	B	17	Darchula	9801234577
Janak	8	C	18	Mustang	9801234578
Kamal	7	A	19	Pyuthan	9801234579
Krishna	6	B	20	Kaski	9801234580
Laxman	10	C	21	Kathmandu	9801234581
Mahesh	9	A	22	Bhaktapur	9801234582
Mohan	8	B	23	Lalitpur	9801234583
Nabin	7	C	24	Kavre	9801234584
Prakash	6	A	25	Dhanusha	9801234585
Ram	10	B	26	Mahottarai	9801234586

Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 03

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement Data Cleaning.

INTRODUCTION:

Data cleaning is the process of removing errors and inconsistencies from data. Data cleaning is an essential step in the data mining process, often referred to as "data cleansing" or "data preparation." It involves identifying and correcting errors, inconsistencies, and missing values in your dataset before using it for analysis.

Uses: Improves data Accuracy, Enhances Performance and Efficiency, Reveals Hidden Patterns, Improves data quality and standardize data, Enhance data reliability, Resolves data conflict.

Common Data Cleaning Techniques:
Identifying and removing duplicates, Handling missing values, Correcting formatting errors, Standardizing data, Validating data, Identifying and removing outliers.

Tools and Techniques for Data Cleaning:
Programming languages: Python, R, and Java offer libraries like Pandas and scikit-learn for data cleaning.
Data visualization tools: Tools like Tableau and Power BI can help visualize inconsistencies and outliers.
Data profiling tools: These tools analyze your data and provide insights into its characteristics and potential issues.

Python Code

```
import pandas as pd

df = pd.read_csv("DataCleaning.csv")
print("Data without any cleaning")
print(df)
print("================================================")
```

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE
PS C:\Users\ASUS\OneDrive\Desktop\DWDM\DataClean> python DataClean.py
Data without any cleaning
    Duration         Date  Pulse  Maxpulse  Calories
0         60  2023/12/01'    110       130     409.1
1         60  2023/12/02'    117       145     479.0
2         60  2023/12/03'    103       135     340.0
3         45  2023/12/04'    109       175     282.4
4         45  2023/12/05'    117       148     406.0
5         60  2023/12/06'    102       127     300.0
6         60  2023/12/07'    110       136     374.0
7        450  2023/12/08'    104       134     253.3
8         30  2023/12/09'    109       133     195.1
9         60  2023/12/10'     98       124     269.0
10        60  2023/12/11'    103       147     329.3
11        60  2023/12/12'    100       120     250.7
12        60  2023/12/12'    100       120     250.7
13        60  2023/12/13'    106       128     345.3
14        60  2023/12/14'    104       132     379.3
15        60  2023/12/15'     98       123     275.0
16        60  2023/12/16'     98       120     215.2
17        60  2023/12/17'    100       120     300.0
18        45  2023/12/18'     90       112       NaN
19        60  2023/12/19'    103       123     323.0
20        45  2023/12/20'     97       125     243.0
21        60  2023/12/21'    108       131     364.2
22        45          NaN    100       119     282.0
23        60  2023/12/23'    130       101     300.0
24        45  2023/12/24'    105       132     246.0
25        60  2023/12/25'    102       126     334.5
26        60     20231226    100       120     250.0
27        60  2023/12/27'     92       118     241.0
28        60  2023/12/28'    103       132       NaN
29        60  2023/12/29'    100       132     280.0
30        60  2023/12/30'    102       129     380.3
31        60  2023/12/31'     92       115     243.0
================================================
```

```
# Drop column
new_df = df.copy()
new_df.dropna(inplace=True)
print("Dropping NaN rows")
print(new_df)
print("================================================")
```

```
Dropping NaN rows
    Duration          Date  Pulse  Maxpulse  Calories
0         60  2023/12/01'    110       130     409.1
1         60  2023/12/02'    117       145     479.0
2         60  2023/12/03'    103       135     340.0
3         45  2023/12/04'    109       175     282.4
4         45  2023/12/05'    117       148     406.0
5         60  2023/12/06'    102       127     300.0
6         60  2023/12/07'    110       136     374.0
7        450  2023/12/08'    104       134     253.3
8         30  2023/12/09'    109       133     195.1
9         60  2023/12/10'     98       124     269.0
10        60  2023/12/11'    103       147     329.3
11        60  2023/12/12'    100       120     250.7
12        60  2023/12/12'    100       120     250.7
13        60  2023/12/13'    106       128     345.3
14        60  2023/12/14'    104       132     379.3
15        60  2023/12/15'     98       123     275.0
16        60  2023/12/16'     98       120     215.2
17        60  2023/12/17'    100       120     300.0
19        60  2023/12/19'    103       123     323.0
20        45  2023/12/20'     97       125     243.0
21        60  2023/12/21'    108       131     364.2
23        60  2023/12/23'    130       101     300.0
24        45  2023/12/24'    105       132     246.0
25        60  2023/12/25'    102       126     334.5
26        60     20231226    100       120     250.0
27        60  2023/12/27'     92       118     241.0
29        60  2023/12/29'    100       132     280.0
30        60  2023/12/30'    102       129     380.3
31        60  2023/12/31'     92       115     243.0
================================================
```

```
# Replace values
new_df2 = df.copy()
# print(new_df2)
new_df2.fillna(222, inplace=True)
print("Filling missing values with 222")
print(new_df2)
print("================================================")
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

```
Filling missing values with 222
    Duration          Date  Pulse  Maxpulse  Calories
0         60  2023/12/01'     110       130     409.1
1         60  2023/12/02'     117       145     479.0
2         60  2023/12/03'     103       135     340.0
3         45  2023/12/04'     109       175     282.4
4         45  2023/12/05'     117       148     406.0
5         60  2023/12/06'     102       127     300.0
6         60  2023/12/07'     110       136     374.0
7        450  2023/12/08'     104       134     253.3
8         30  2023/12/09'     109       133     195.1
9         60  2023/12/10'      98       124     269.0
10        60  2023/12/11'     103       147     329.3
11        60  2023/12/12'     100       120     250.7
12        60  2023/12/12'     100       120     250.7
13        60  2023/12/13'     106       128     345.3
14        60  2023/12/14'     104       132     379.3
15        60  2023/12/15'      98       123     275.0
16        60  2023/12/16'      98       120     215.2
17        60  2023/12/17'     100       120     300.0
18        45  2023/12/18'      90       112     222.0
19        60  2023/12/19'     103       123     323.0
20        45  2023/12/20'      97       125     243.0
21        60  2023/12/21'     108       131     364.2
22        45          222     100       119     282.0
23        60  2023/12/23'     130       101     300.0
24        45  2023/12/24'     105       132     246.0
25        60  2023/12/25'     102       126     334.5
26        60     20231226     100       120     250.0
27        60  2023/12/27'      92       118     241.0
28        60  2023/12/28'     103       132     222.0
29        60  2023/12/29'     100       132     280.0
30        60  2023/12/30'     102       129     380.3
31        60  2023/12/31'      92       115     243.0
================================================
```

```
# Replace Specific Column only
new_df3 = df.copy()
# print(new_df3)
print("Replacing specific column only")
new_df3["Date"].fillna(5,inplace=True)
print(new_df3)
print("================================================")
```

```
Replacing specific column only
    Duration          Date  Pulse  Maxpulse  Calories
0         60  2023/12/01'     110       130     409.1
1         60  2023/12/02'     117       145     479.0
2         60  2023/12/03'     103       135     340.0
3         45  2023/12/04'     109       175     282.4
4         45  2023/12/05'     117       148     406.0
5         60  2023/12/06'     102       127     300.0
6         60  2023/12/07'     110       136     374.0
7        450  2023/12/08'     104       134     253.3
8         30  2023/12/09'     109       133     195.1
9         60  2023/12/10'      98       124     269.0
10        60  2023/12/11'     103       147     329.3
11        60  2023/12/12'     100       120     250.7
12        60  2023/12/12'     100       120     250.7
13        60  2023/12/13'     106       128     345.3
14        60  2023/12/14'     104       132     379.3
15        60  2023/12/15'      98       123     275.0
16        60  2023/12/16'      98       120     215.2
17        60  2023/12/17'     100       120     300.0
18        45  2023/12/18'      90       112       NaN
19        60  2023/12/19'     103       123     323.0
20        45  2023/12/20'      97       125     243.0
21        60  2023/12/21'     108       131     364.2
22        45            5     100       119     282.0
23        60  2023/12/23'     130       101     300.0
24        45  2023/12/24'     105       132     246.0
25        60  2023/12/25'     102       126     334.5
26        60     20231226     100       120     250.0
27        60  2023/12/27'      92       118     241.0
28        60  2023/12/28'     103       132       NaN
29        60  2023/12/29'     100       132     280.0
30        60  2023/12/30'     102       129     380.3
31        60  2023/12/31'      92       115     243.0
================================================
```

```
#Replace missing value of a column with a mean value
new_df4 = df.copy()
# print(new_df4)
x = new_df4["Calories"].mean()
print("Mean of the calories :",x)
print("Replacing missing value with mean")
new_df4["Calories"].fillna(x,inplace=True)
print(new_df4)
print("================================================")
```

```
Mean of the calories : 304.68
Replacing missing value with mean
    Duration          Date  Pulse  Maxpulse  Calories
0         60  2023/12/01'     110       130    409.10
1         60  2023/12/02'     117       145    479.00
2         60  2023/12/03'     103       135    340.00
3         45  2023/12/04'     109       175    282.40
4         45  2023/12/05'     117       148    406.00
5         60  2023/12/06'     102       127    300.00
6         60  2023/12/07'     110       136    374.00
7        450  2023/12/08'     104       134    253.30
8         30  2023/12/09'     109       133    195.10
9         60  2023/12/10'      98       124    269.00
10        60  2023/12/11'     103       147    329.30
11        60  2023/12/12'     100       120    250.70
12        60  2023/12/12'     100       120    250.70
13        60  2023/12/13'     106       128    345.30
14        60  2023/12/14'     104       132    379.30
15        60  2023/12/15'      98       123    275.00
16        60  2023/12/16'      98       120    215.20
17        60  2023/12/17'     100       120    300.00
18        45  2023/12/18'      90       112    304.68
19        60  2023/12/19'     103       123    323.00
20        45  2023/12/20'      97       125    243.00
21        60  2023/12/21'     108       131    364.20
22        45          NaN     100       119    282.00
23        60  2023/12/23'     130       101    300.00
24        45  2023/12/24'     105       132    246.00
25        60  2023/12/25'     102       126    334.50
26        60     20231226     100       120    250.00
27        60  2023/12/27'      92       118    241.00
28        60  2023/12/28'     103       132    304.68
29        60  2023/12/29'     100       132    280.00
30        60  2023/12/30'     102       129    380.30
31        60  2023/12/31'      92       115    243.00
================================================
```

```
#Cleaning wrong data
new_df5 = df.copy()
# print(new_df5)
new_df5["Date"] = new_df5["Date"].apply(pd.to_datetime)
print("Cleaning wrong data")
print(new_df5)
print("================================================")
```

```
Cleaning wrong data
    Duration       Date  Pulse  Maxpulse  Calories
0         60 2023-12-01    110       130     409.1
1         60 2023-12-02    117       145     479.0
2         60 2023-12-03    103       135     340.0
3         45 2023-12-04    109       175     282.4
4         45 2023-12-05    117       148     406.0
5         60 2023-12-06    102       127     300.0
6         60 2023-12-07    110       136     374.0
7        450 2023-12-08    104       134     253.3
8         30 2023-12-09    109       133     195.1
9         60 2023-12-10     98       124     269.0
10        60 2023-12-11    103       147     329.3
11        60 2023-12-12    100       120     250.7
12        60 2023-12-12    100       120     250.7
13        60 2023-12-13    106       128     345.3
14        60 2023-12-14    104       132     379.3
15        60 2023-12-15     98       123     275.0
16        60 2023-12-16     98       120     215.2
17        60 2023-12-17    100       120     300.0
18        45 2023-12-18     90       112       NaN
19        60 2023-12-19    103       123     323.0
20        45 2023-12-20     97       125     243.0
21        60 2023-12-21    108       131     364.2
22        45        NaT    100       119     282.0
23        60 2023-12-23    130       101     300.0
24        45 2023-12-24    105       132     246.0
25        60 2023-12-25    102       126     334.5
26        60 2023-12-26    100       120     250.0
27        60 2023-12-27     92       118     241.0
28        60 2023-12-28    103       132       NaN
29        60 2023-12-29    100       132     280.0
30        60 2023-12-30    102       129     380.3
31        60 2023-12-31     92       115     243.0
================================================
```

```
#Remove the unidentified Datetime
new_df5.dropna(subset=['Date'],inplace=True)
print("Remove unidentified datetime")
print(new_df5)
print("================================================")
```

```
Remove unidentified datetime
    Duration       Date  Pulse  Maxpulse  Calories
0         60 2023-12-01    110       130     409.1
1         60 2023-12-02    117       145     479.0
2         60 2023-12-03    103       135     340.0
3         45 2023-12-04    109       175     282.4
4         45 2023-12-05    117       148     406.0
5         60 2023-12-06    102       127     300.0
6         60 2023-12-07    110       136     374.0
7        450 2023-12-08    104       134     253.3
8         30 2023-12-09    109       133     195.1
9         60 2023-12-10     98       124     269.0
10        60 2023-12-11    103       147     329.3
11        60 2023-12-12    100       120     250.7
12        60 2023-12-12    100       120     250.7
13        60 2023-12-13    106       128     345.3
14        60 2023-12-14    104       132     379.3
15        60 2023-12-15     98       123     275.0
16        60 2023-12-16     98       120     215.2
17        60 2023-12-17    100       120     300.0
18        45 2023-12-18     90       112       NaN
19        60 2023-12-19    103       123     323.0
20        45 2023-12-20     97       125     243.0
21        60 2023-12-21    108       131     364.2
23        60 2023-12-23    130       101     300.0
24        45 2023-12-24    105       132     246.0
25        60 2023-12-25    102       126     334.5
26        60 2023-12-26    100       120     250.0
27        60 2023-12-27     92       118     241.0
28        60 2023-12-28    103       132       NaN
29        60 2023-12-29    100       132     280.0
30        60 2023-12-30    102       129     380.3
31        60 2023-12-31     92       115     243.0
================================================
```

```
# cleaning outlier data
new_df6 = df.copy()
# print(new_df6)
#Replace all values higher than 120 with 80
for i in new_df6.index:
 if new_df6.loc[i,"Duration"]>120:
 new_df6.loc[i,"Duration"]=80
print("Replacing value higher than 120 with 80")
print(new_df6)
print("=============================================")
```

```
Replacing value higher than 120 with 80
    Duration          Date  Pulse  Maxpulse  Calories
0         60  2023/12/01'     110       130     409.1
1         60  2023/12/02'     117       145     479.0
2         60  2023/12/03'     103       135     340.0
3         45  2023/12/04'     109       175     282.4
4         45  2023/12/05'     117       148     406.0
5         60  2023/12/06'     102       127     300.0
6         60  2023/12/07'     110       136     374.0
7         80  2023/12/08'     104       134     253.3
8         30  2023/12/09'     109       133     195.1
9         60  2023/12/10'      98       124     269.0
10        60  2023/12/11'     103       147     329.3
11        60  2023/12/12'     100       120     250.7
12        60  2023/12/12'     100       120     250.7
13        60  2023/12/13'     106       128     345.3
14        60  2023/12/14'     104       132     379.3
15        60  2023/12/15'      98       123     275.0
16        60  2023/12/16'      98       120     215.2
17        60  2023/12/17'     100       120     300.0
18        45  2023/12/18'      90       112       NaN
19        60  2023/12/19'     103       123     323.0
20        45  2023/12/20'      97       125     243.0
21        60  2023/12/21'     108       131     364.2
22        45          NaN     100       119     282.0
23        60  2023/12/23'     130       101     300.0
24        45  2023/12/24'     105       132     246.0
25        60  2023/12/25'     102       126     334.5
26        60     20231226     100       120     250.0
27        60  2023/12/27'      92       118     241.0
28        60  2023/12/28'     103       132       NaN
29        60  2023/12/29'     100       132     280.0
30        60  2023/12/30'     102       129     380.3
31        60  2023/12/31'      92       115     243.0
=============================================
```

```
#Duplicate values
new_df7 = df.copy()
print("print duplicated values")
print(new_df7.duplicated())
print("================================================")
```

```
print duplicated values
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12      True
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28     False
29     False
30     False
31     False
dtype: bool
================================================
```

```
print("Drop dublicate values")
new_df7.drop_duplicates(inplace=True)

print(new_df7)
```

```
Drop dublicate values
     Duration          Date  Pulse  Maxpulse  Calories
0          60  2023/12/01'    110       130     409.1
1          60  2023/12/02'    117       145     479.0
2          60  2023/12/03'    103       135     340.0
Run and Debug (Ctrl+Shift+D)  04'    109       175     282.4
4          45  2023/12/05'    117       148     406.0
5          60  2023/12/06'    102       127     300.0
6          60  2023/12/07'    110       136     374.0
7         450  2023/12/08'    104       134     253.3
8          30  2023/12/09'    109       133     195.1
9          60  2023/12/10'     98       124     269.0
10         60  2023/12/11'    103       147     329.3
11         60  2023/12/12'    100       120     250.7
13         60  2023/12/13'    106       128     345.3
14         60  2023/12/14'    104       132     379.3
15         60  2023/12/15'     98       123     275.0
16         60  2023/12/16'     98       120     215.2
17         60  2023/12/17'    100       120     300.0
18         45  2023/12/18'     90       112       NaN
19         60  2023/12/19'    103       123     323.0
20         45  2023/12/20'     97       125     243.0
21         60  2023/12/21'    108       131     364.2
22         45          NaN    100       119     282.0
23         60  2023/12/23'    130       101     300.0
24         45  2023/12/24'    105       132     246.0
25         60  2023/12/25'    102       126     334.5
26         60     20231226    100       120     250.0
27         60  2023/12/27'     92       118     241.0
28         60  2023/12/28'    103       132       NaN
29         60  2023/12/29'    100       132     280.0
30         60  2023/12/30'    102       129     380.3
31         60  2023/12/31'     92       115     243.0
PS C:\Users\ASUS\OneDrive\Desktop\DWDM\DataClean>
```

Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 04

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement Data Cubes.

INTRODUCTION:

Data cubes are multi-dimensional data structures used in data warehousing and online analytical processing (OLAP) to efficiently analyze large amounts of data from various perspectives. A data cube is a collection of data that is organized into a lattice of dimensions and measures and cells.
Each dimension represents a different aspect of the data, such as time, product, or customer. Each dimension has its own set of attributes (e.g., subcategories, demographics, months).
Each measure represents a quantitative value that is associated with specific combinations of dimensions (e.g., sales amount, average order value, customer count).
Each cell represents the intersection of a particular combination of dimensions and contains a measure value.

Benefits: Fast data retrieval and analysis, multidimensional analysis, drill-down and roll-up, flexible exploration, analyze data from multiple perspectives, trend and pattern insights for decision making.
Applications: Sales Analysis, Customer Analysis, Financial Analysis.

```
CREATE TABLE employee (
  emp_id INT,
  name VARCHAR(50),
  salary DECIMAL(10, 2),
  dept_id VARCHAR(10),
  job_id VARCHAR(10),
  manager_id VARCHAR(10)
);

INSERT INTO employee (emp_id, name, salary, dept_id, job_id, manager_id)
VALUES
 (76800, 'Bishal Karki', 30000.00, 'dept700', 'jb811', 'mi600'),
 (76801, 'Sita Bhattarai', 35000.00, 'dept700', 'jb806', 'mi600'),
 (76802, 'Hari Paudel', 40000.00, 'dept707', 'jb809', 'mi600'),
 (76803, 'Ramesh Dahal', 38000.00, 'dept707', 'jb809', 'mi600'),
 (76804, 'Nisha Khadka', 45000.00, 'dept777', 'jb811', 'mi619'),
 (76805, 'Rajesh Karki', 42000.00, 'dept777', 'jb806', 'mi619'),
 (76806, 'Sarita Bhattarai', 35000.00, 'dept700', 'jb809', 'mi602'),
 (76807, 'Sanjay Paudel', 40000.00, 'dept707', 'jb809', 'mi602'),
 (76808, 'Manish Dahal', 43000.00, 'dept777', 'jb809', 'mi602'),
 (76809, 'Pratima Khadka', 32000.00, 'dept700', 'jb811', 'mi603'),
 (76810, 'Nabin Karki', 40000.00, 'dept707', 'jb811', 'mi603'),
 (76811, 'Anita Bhattarai', 42000.00, 'dept777', 'jb811', 'mi603'),
 (76812, 'Rina Paudel', 40000.00, 'dept700', 'jb811', 'mi604');
```

emp_id	name	salary	dept_id	job_id	manager_id
76800	Bishal Karki	30000.00	dept700	jb811	mi600
76801	Sita Bhattarai	35000.00	dept700	jb806	mi600
76802	Hari Paudel	40000.00	dept707	jb809	mi600
76803	Ramesh Dahal	38000.00	dept707	jb809	mi600
76804	Nisha Khadka	45000.00	dept777	jb811	mi619
76805	Rajesh Karki	42000.00	dept777	jb806	mi619
76806	Sarita Bhattarai	35000.00	dept700	jb809	mi602
76807	Sanjay Paudel	40000.00	dept707	jb809	mi602
76808	Manish Dahal	43000.00	dept777	jb809	mi602
76809	Pratima Khadka	32000.00	dept700	jb811	mi603
76810	Nabin Karki	40000.00	dept707	jb811	mi603
76811	Anita Bhattarai	42000.00	dept777	jb811	mi603
76812	Rina Paudel	40000.00	dept700	jb811	mi604

```
CREATE TABLE dept_cube AS
SELECT dept_id, COUNT(*) AS noofemp, SUM(salary) AS sumsal
FROM employee
GROUP BY dept_id;
```

```
SELECT * FROM dept_cube;
```

dept_id	noofemp	sumsal
dept700	5	172000.00
dept707	4	158000.00
dept777	4	172000.00

```
CREATE TABLE deptjob_cube AS
SELECT dept_id, job_id, COUNT(*) AS noofemp, SUM(salary) AS sumsal
FROM employee
GROUP BY dept_id, job_id;
```

```
SELECT * FROM deptjob_cube;
```

dept_id	noofemp	sumsal
dept700	5	172000.00
dept707	4	158000.00
dept777	4	172000.00

```
CREATE TABLE deptJobManager_cube AS
SELECT dept_id, job_id, manager_id, COUNT(*) AS noofemp, SUM(salary) AS sumsal
FROM employee
GROUP BY dept_id, job_id, manager_id;
```

```
SELECT * FROM deptJobManager_cube;
```

dept_id	job_id	manager_id	noofemp	sumsal
dept700	jb806	mi600	1	35000.00
dept700	jb809	mi602	1	35000.00
dept700	jb811	mi600	1	30000.00
dept700	jb811	mi603	1	32000.00
dept700	jb811	mi604	1	40000.00
dept707	jb809	mi600	2	78000.00
dept707	jb809	mi602	1	40000.00
dept707	jb811	mi603	1	40000.00
dept777	jb806	mi619	1	42000.00
dept777	jb809	mi602	1	43000.00
dept777	jb811	mi603	1	42000.00
dept777	jb811	mi619	1	45000.00

```
SELECT dept_id, SUM(noofemp) AS noofemp, SUM(sumsal) AS sumsal
FROM deptJobManager_cube

GROUP BY dept_id;
```

dept_id	noofemp	sumsal
dept700	5	172000.00
dept707	4	158000.00
dept777	4	172000.00

Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 05

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement K-Means Clustering Algorithm.

Objectives

To generate 1000 2D data points in the range 0-100 randomly and divide data points into 3 clusters.

Introduction

k-means is one of the simplest partitioning based clusteringalgorithm. The procedure follows a simple and easy way to classify a given data set into a certain number of clusters (assume k clusters) fixed apriori.
The main idea is to define k centers, one for each cluster. These centers should be selected cleverly becauseof different location causes different result. So, the better choice is to place them as much as possible far away from each other.

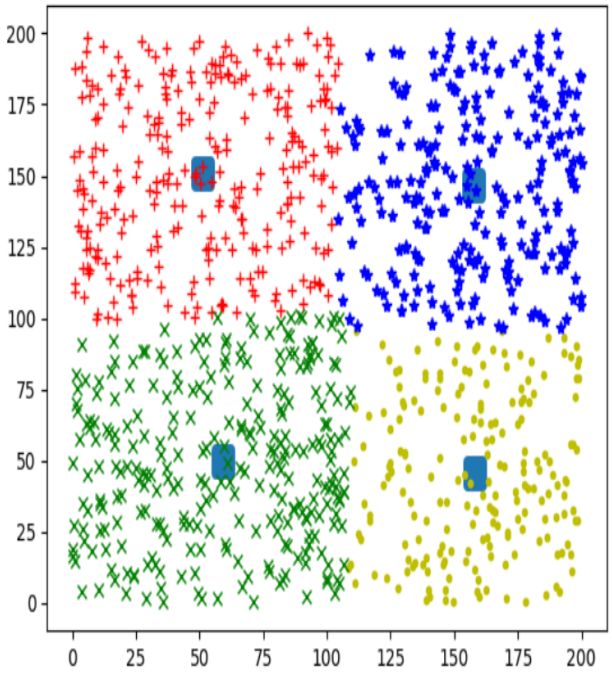
Algorithm

Let X = {x1,x2,x3,……..,xn} be the set of data points and C ={c1,c2,…….,ck} be the set of cluster centers.
1. Randomly select k cluster centers.
2. Calculate the distance between each data point and cluster centers.
3. Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.
4. If
   No data is reassigned then terminate
5. Else
• Recalculate the new cluster center using centroid.
• Recalculate the distance between each data point and new cluster centers.
• Go to step 3

Python Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import Kmeans

# Generating random 1000 data
data = np.random.rand(1000, 2) * 200

# Creating an instance of the KMeans algorithm
km = KMeans(n_clusters=4, init="k-means++")

# Training the algorithm on the data
km.fit(data)

# Retrieving the cluster centers and labels
centers = km.cluster_centers_
labels = km.labels_

# Printing the cluster centers
print("Cluster centers: ", *centers)

# Defining colors and markers for data visualization
colors = ["r", "g", "b", "y"]
markers = ["+", "x", "*", "."]

# Plotting the data points with different colors based on cluster labels
for i in range(len(data)):
    plt.plot(data[i][0], data[i][1], color=colors[labels[i]], marker=markers[labels[i]])
# Plotting the cluster centers
plt.scatter(centers[:, 0], centers[:, 1], marker="s", s=100, linewidths=5)
# Displaying the plot
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
Cluster centers:  [ 51.14379612 150.90414584] [58.94604207 49.71739722] [157.48114955 146.58767461] [158.23181892  45.49745728]
```



Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 06

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement K-Means++ Clustering Algorithm.

Objectives

To generate 1000 2D data points in the range 0-200 randomly and divide data points into 4 clusters.

Introduction

Randomization of picking k cluster centers in K-means algorithm results in the problem of initialization sensitivity. This problem tends to affect the final formed clusters. The final formed clusters depend on how initial cluster centers werepicked. To overcome the above-mentioned drawback we use Kmeans++. This algorithm ensures a smarter initialization of the cluster centers and improves the quality of the clustering.

Algorithm

Initialization Algorithm

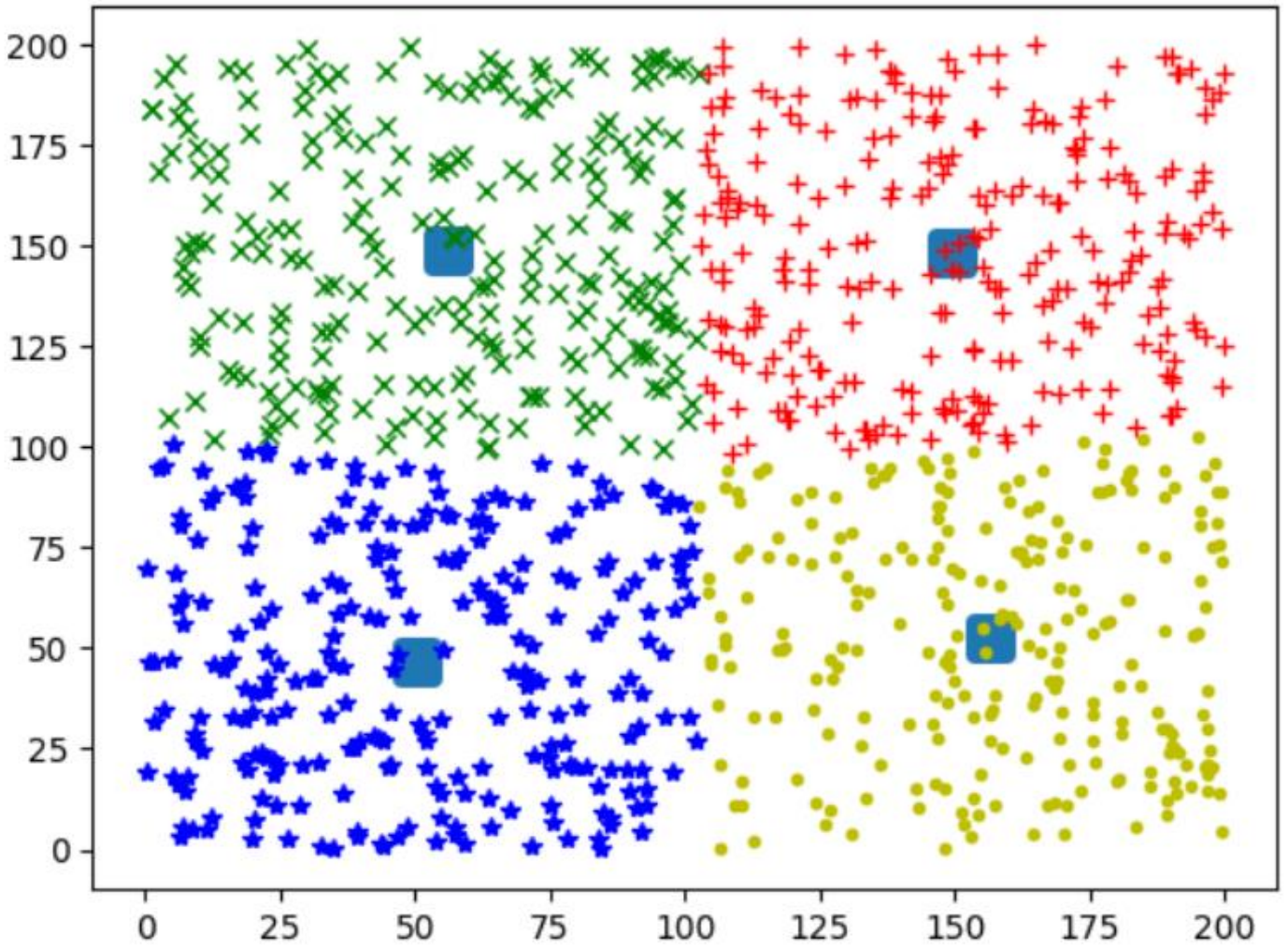
1. Randomly select the first cluster center from the data points.
2. For each data point compute its distance from the nearest, previously chosen cluster center.
3. Select the next cluster from the data points such that the probability of choosing a point as cluster center is directly proportional to its distance from the nearest, previously chosen cluster center.
4. Repeat steps 2 and 3 until K cluster centers have been sampled

Python Code

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import Kmeans

# Generate random data
data = np.random.rand(1000, 2) * 200

# Create an instance of the KMeans algorithm
km = KMeans(n_clusters=4, init="k-means++")
# Train the algorithm on the data
km.fit(data)
# Retrieve the cluster centers and labels
centers = km.cluster_centers_
labels = km.labels_

# Print the cluster centers
print("Cluster centers: ", *centers)

# Print the cluster labels
print("Cluster Labels: ", *labels)
# Define colors and markers for data visualization
colors = ["r", "g", "b", "y"]
markers = ["+", "x", "*", "."]

# Plot the data points with different colors based on cluster labels
for i in range(len(data)):
    plt.plot(data[i][0], data[i][1], color=colors[labels[i]], marker=markers[labels[i]])
# Plot the cluster centers
plt.scatter(centers[:, 0], centers[:, 1], marker="s", s=100, linewidths=5)
# Display the plot
plt.show()
```

Output:

Cluster centers: [149.28636193 148.4277196] [56.04790514 149.09697859] [50.17893485 46.95611314] [156.45178906 52.55743717]


200
175
150
125
100
75
50
25
0
0
25
50
75
100
125
150
175
200

Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 07

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement K-Mediods Clustering Algorithm.

Objectives

To find clusters of Iris Dataset using KMedoids Clustering Algorithm.

Introduction

K-Medoids is also a portioning based clustering algorithm. It is also called as Partitioning Around Medoid (PAM) algorithm.
A medoid can be defined as the point in the cluster, whose dissimilarities with all the other points in the cluster is minimum.
It majorly differs from the K-Means algorithm in terms of the way it selects the cluster centers.
It selects the average of a cluster's points as its center whereas the K-Medoid algorithm always picks the actual data points from the clusters as their centers.

Algorithm

1. Select k random points out of the n data points as the medoids.
2. Associate each data point to the closest medoid.
3. Repeat while the cost decreases:
4. For each medoid m
• For each non-medoid point o
• Swap m and o
• Associate each data point to the closest medoid
• Re-compute the cost
5. If the total cost is more than that in the previous step
• undo the swap.

Python Code

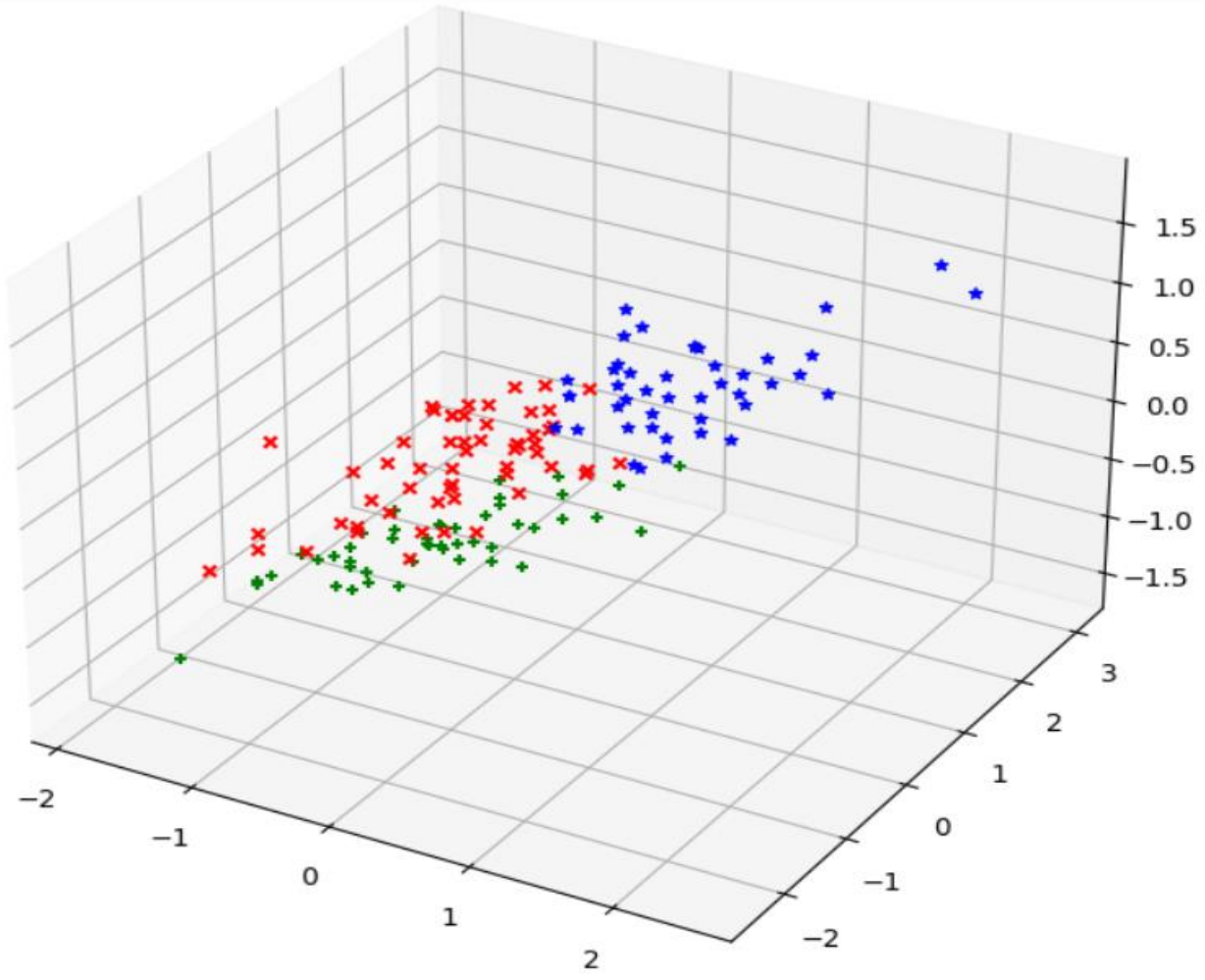
```
!pip install scikit-learn-extra

from sklearn.datasets import load_iris  # Load Iris dataset
from sklearn.preprocessing import StandardScaler  # Scale data
from sklearn_extra.cluster import KMedoids  # Import KMedoids algorithm
from sklearn import metrics  # For evaluating clustering performance
import matplotlib.pyplot as plt  # For visualization

# Load Iris dataset
iris_data = load_iris()
# Extract features and target labels
x = iris_data.data  # Features
y = iris_data.target  # True labels
# Normalize features using StandardScaler
sc = StandardScaler().fit(x)
sx = sc.transform(x)
# Create a KMedoids model with 3 clusters
km = KMedoids(n_clusters=3)
# Fit the model to the scaled data
km.fit(sx)

# Predict cluster labels for each data point
py = km.fit_predict(sx)

# Visualize clusters in a 3D scatter plot
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection="3d")
colors = ["g", "r", "b"]  # Colors for clusters
markers = ["+", "x", "*"]  # Markers for clusters
for i in range(len(sx)):
   ax.scatter(sx[i][0], sx[i][1], sx[i][2],
              color=colors[py[i]], marker=markers[py[i]])
plt.show()

# Evaluate clustering performance
ri = metrics.adjusted_rand_score(y, py)  # Rand Index
print("Rand Index:", ri)
hs = metrics.homogeneity_score(y, py)  # Homogeneity Score
print("Homogeniety Score:", hs)
cs = metrics.completeness_score(y, py)  # Completeness Score
print("Completeness Score:", cs)
sc = metrics.silhouette_score(sx, py, metric="euclidean")  # Silhouette Coefficient
print("Silhouette Coefficient:", sc)print("Homogeniety Score:", hs)
cs = metrics.completeness_score(y, py)  # Completeness Score
print("Completeness Score:", cs)
```

```
sc = metrics.silhouette_score(sx, py, metric="euclidean")  # Silhouette Coefficient
print("Silhouette Coefficient:", sc)
```

Output

```
Requirement already satisfied: scikit-learn-extra in /usr/local/lib/python3.10/dist-packages (0.3.0)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn-extra) (1.25.
Requirement already satisfied: scipy>=0.19.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn-extra) (1.11.
Requirement already satisfied: scikit-learn>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn-extra)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.23.0->scik
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.23.
```

15s




```
Rand Index: 0.631158086738433
Homogeniety Score: 0.6672491406379297
Completeness Score: 0.6701843437329579
Silhouette Coefficient: 0.4590416105554613
```

Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 08

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement K-Agglomerative Clustering Algorithm.

Objectives

To find clusters of Iris Dataset using Agglomerative Clustering Algorithm and compare them in terms of different performance measures.

Introduction

This is a bottom up approach. Initially,each observation is considered in separate cluster and pairs of clusters are merged as one moves up the hierarchy.
This process continues until the single cluster or required number of clusters are formed.
Distance matrix is used for deciding which clusters to merge

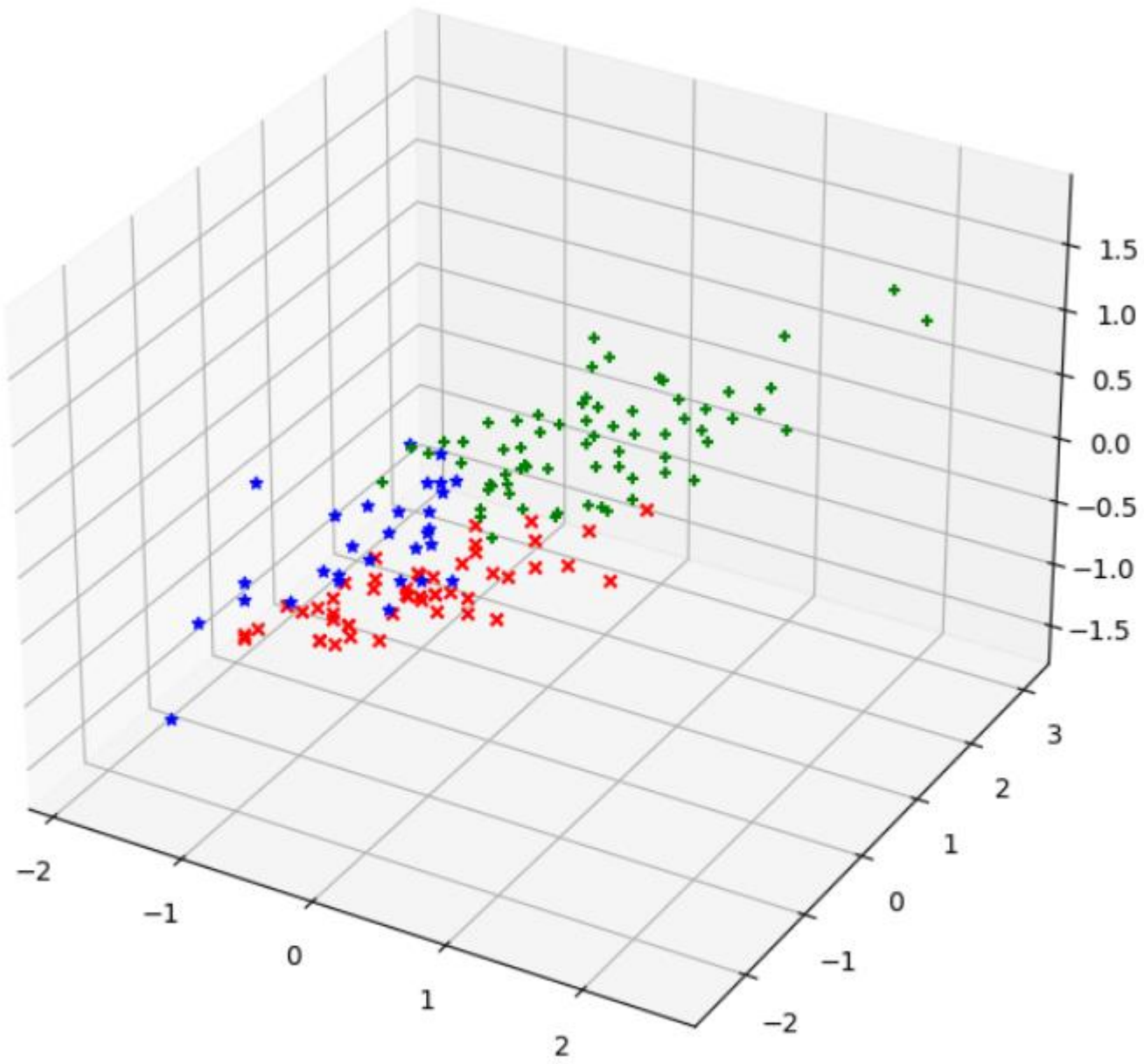
Algorithm

• Compute the distance matrix between the input data points
• Let each data point be a cluster
• Repeat Merge the two closest clusters and Update the distance matrix Until only K clusters remains

Python Code

```
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from sklearn import metrics
import matplotlib.pyplot as plt

# Load Iris dataset
iris_data = load_iris()

# Extract features and target labels
x = iris_data.data  # Features
y = iris_data.target  # True labels

# Normalize features using StandardScaler
sc = StandardScaler().fit(x)
sx = sc.transform(x)

# Create an instance of AgglomerativeClustering with 3 clusters
ac = AgglomerativeClustering(n_clusters=3)

# Fit the model to the scaled data
ac.fit(sx)

# Predict cluster labels for each data point
py = ac.fit_predict(sx)

# Visualize clusters in a 3D scatter plot
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection="3d")
colors = ["g", "r", "b"]  # Colors for clusters
markers = ["+", "x", "*"]  # Markers for clusters
for i in range(len(sx)):
    ax.scatter(sx[i][0], sx[i][1], sx[i][2], color=colors[py[i]], marker=markers[py[i]])
plt.show()

# Evaluate clustering performance
ri = metrics.rand_score(y, py)  # Rand Index
print("Rand Index:", ri)
hs = metrics.homogeneity_score(y, py)  # Homogeneity Score
print("Homogeniety Score:", hs)
cs = metrics.completeness_score(y, py)  # Completeness Score
print("Completeness Score:", cs)
sc = metrics.silhouette_score(sx, py, metric="euclidean")  # Silhouette Coefficient
print("Silhouette Coefficient:", sc)
```

Output




```
Rand Index: 0.8252348993288591
Homogeniety Score: 0.6578818079976051
Completeness Score: 0.6940248415952218
Silhouette Coefficient: 0.4466890410285909
```

Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 09

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement Naive Bayes Classification.

Objectives

To predict diabeties using Naive Bayes Classification.

Introduction

Bayesian classification is based on Bayes' theorem. It is also called Naïve Bayes

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}$$

Classification or Naïve Bayesian Classification.
Bayes Theorem is given by:

Bayes' theorem is useful in that it provides a way of calculating the posterior probability, P(H|X), from P(H), P(X|H), and P(X).Here P(X) and P(H) are prior probability.

Bayesian Classification

Let X is the set of attributes $\{x_1, x_2, x_3 \ldots\ldots x_n\}$ where attributes are independent of one another. Now the probability $P(X|C_i)$ is given by the equation given below:

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times P(x_2 \mid C_i) \ldots\ldots \times P(x_n \mid C_i)$$

Python Code

```
from google.colab import files
uploaded = files.upload()

import io
# Load the dataset
df = pd.read_csv(io.BytesIO(uploaded['Diabetes.csv']))
print(df)
```

```
     Pragnency  Glucose  Blod Pressure  Skin Thikness  Insulin   BMI     DFP  \
0            1       85             66             29        0  26.6   0.351
1            8      183             64              0        0  23.3   0.672
2            1       89             66             23       94  28.1   0.167
3            0      137             40             35      168  43.1   2.288
4            5      116             74              0        0  25.6   0.201
..         ...      ...            ...            ...      ...   ...     ...
762         10      101             76             48      180  32.9   0.171
763          2      122             70             27        0  36.8   0.340
764          5      121             72             23      112  26.2   0.245
765          1      126             60              0        0  30.1   0.349
766          1       93             70             31        0  30.4   0.315

     Age  Diabetes
0     31         0
1     32         1
2     21         0
3     33         1
4     30         0
..   ...       ...
762   63         0
763   27         0
764   30         0
765   47         1
766   23         0

[767 rows x 9 columns]
```

```
import pandas as pd
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB

# Print the size of the dataset
print("Dataset Size:", len(dataset))

# Split the dataset into train and test sets
split = int(len(dataset) * 0.7)
train, test = dataset.iloc[:split], dataset.iloc[split:]

# Extract the train features and target variable
p = train["Pragnency"].values
g = train["Glucose"].values
bp = train["Blod Pressure"].values
```

```
st = train["Skin Thikness"].values
ins = train["Insulin"].values
bmi = train["BMI"].values
dpf = train["DFP"].values
a = train["Age"].values
d = train["Diabetes"].values
trainfeatures = zip(p, g, bp, st, ins, bmi, dpf, a)
traininput = list(trainfeatures)

# Fit the Gaussian Naive Bayes model to the train data
model = GaussianNB()
model.fit(traininput, d)

# Extract the test features and target variable
p = test["Pragnency"].values
g = test["Glucose"].values
bp = test["Blod Pressure"].values
st = test["Skin Thikness"].values
ins = test["Insulin"].values
bmi = test["BMI"].values
dpf = test["DFP"].values
a = test["Age"].values
d = test["Diabetes"].values
testfeatures = zip(p, g, bp, st, ins, bmi, dpf, a)
testinput = list(testfeatures)

# Predict the target variable for the test data using the trained model
predicted = model.predict(testinput)

# Calculate and print the confusion matrix
print("Confusion Matrix:")
print(metrics.confusion_matrix(d, predicted))

# Calculate and print the classification measures
print("\nClassification Measures:")
print("Accuracy:", metrics.accuracy_score(d, predicted))
print("Recall:", metrics.recall_score(d, predicted))
print("Precision:", metrics.precision_score(d, predicted))
print("F1-score:", metrics.f1_score(d, predicted))
```

Output

+ Code + Text

```
# Calculate and print the confusion matrix
print("Confusion Matrix:")
print(metrics.confusion_matrix(d, predicted))
```

```
Confusion Matrix:
[[128  24]
 [ 30  49]]
```

```


# Calculate and print the classification measures
print("\nClassification Measures:")
print("Accuracy:", metrics.accuracy_score(d, predicted))
print("Recall:", metrics.recall_score(d, predicted))
print("Precision:", metrics.precision_score(d, predicted))
print("F1-score:", metrics.f1_score(d, predicted))
```

```
Classification Measures:
Accuracy: 0.7662337662337663
Recall: 0.620253164556962
Precision: 0.6712328767123288
F1-score: 0.6447368421052632
```

Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 10

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement using ID3 Decision Tree Classifier.

Objectives

To predict diabeties using ID3 Decision Tree Classifier. Compare the performance of both classifiers.

Introduction

Decision tree induction is the learning of decision trees from class labeled training tuples.

Decision tree is a flowchart-like tree structure where internal nodes (non leaf node) denotes a test on an attribute, branches represent outcomes of tests, and Leaf nodes (terminal nodes) hold class labels.

In order to make prediction for a tuple, the attributes of a tuple are tested against the decision tree. A path is traced from the root to a leaf node which determines the predicted class for that tuple.

Algorithms

Algorithm for Constructing Decision Tress

• Constructing a Decision tree uses greedy algorithm. Tree is constructed in a top-down divide-and-conquer manner.

1. At start, all the training tuples are at the root
2. Tuples are partitioned recursively based on selected attributes
3. If all samples for a given node belong to the same class

• Label the class

4. Else if there are no remaining attributes for further partitioning

• Majority voting is employed for assigning class label to the leaf

5. Else

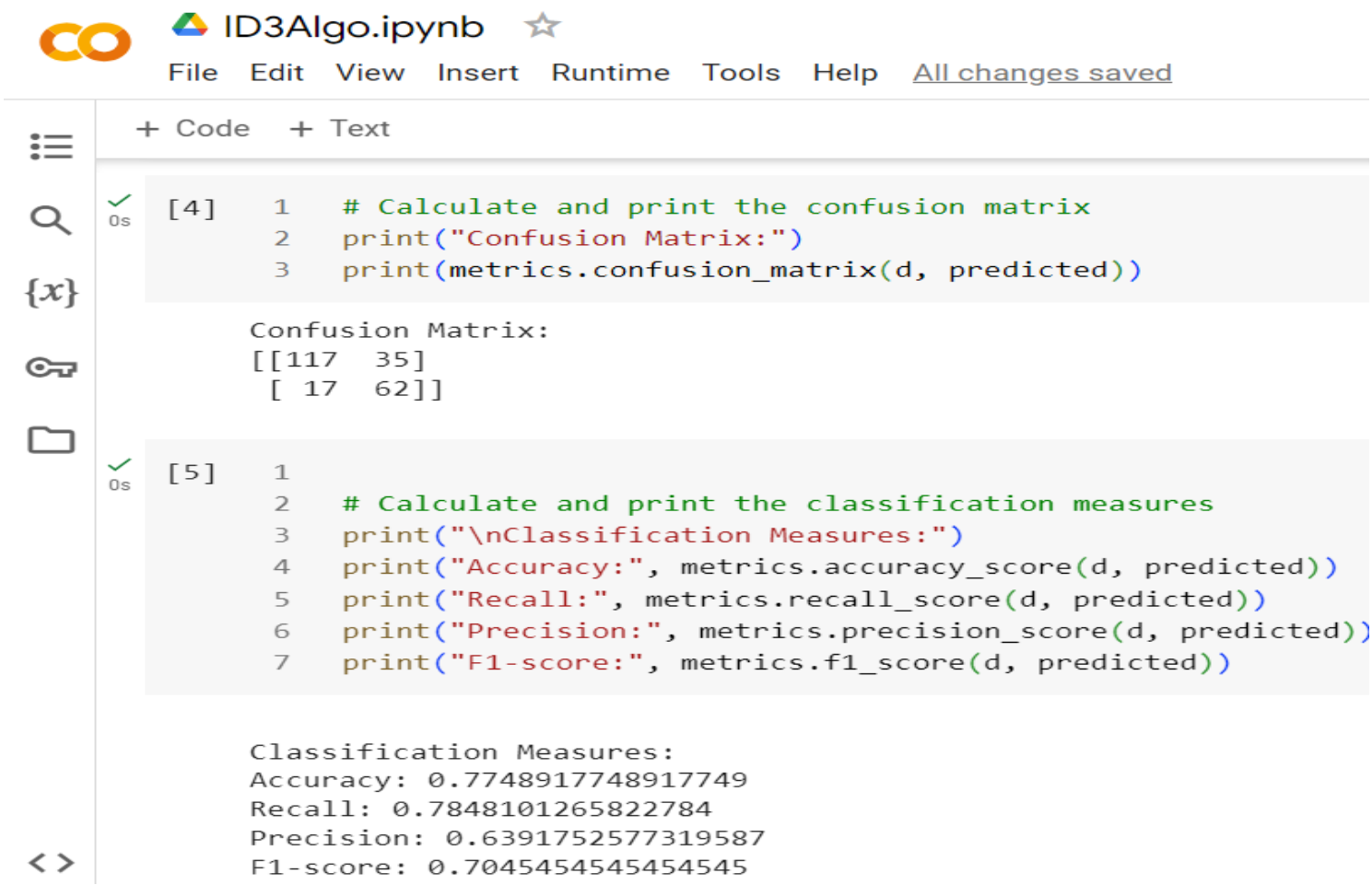
• Got to step 2

Python Code

```
import io
# Load the dataset
df = pd.read_csv(io.BytesIO(uploaded['Diabetes.csv']))
print(df)

import pandas as pd

from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier

# Print the size of the dataset
print("Dataset Size: ", len(dataset))

# Split the dataset into train and test sets
split = int(len(dataset) * 0.7)
train, test = dataset.iloc[:split], dataset.iloc[split:]

# Extract the train features and target variable
p = train["Pragnency"].values
g = train["Glucose"].values
bp = train["Blod Pressure"].values
st = train["Skin Thikness"].values
ins = train["Insulin"].values
bmi = train["BMI"].values
dpf = train["DFP"].values
a = train["Age"].values
d = train["Diabetes"].values
trainfeatures = zip(p, g, bp, st, ins, bmi, dpf, a)
traininput = list(trainfeatures)

# Fit the Decision Tree model to the train data
model = DecisionTreeClassifier(criterion="entropy", max_depth=4)
model.fit(traininput, d)

# Extract the test features and target variable
p = test["Pragnency"].values
g = test["Glucose"].values
bp = test["Blod Pressure"].values
st = test["Skin Thikness"].values
ins = test["Insulin"].values
bmi = test["BMI"].values
dpf = test["DFP"].values
a = test["Age"].values
d = test["Diabetes"].values
testfeatures = zip(p, g, bp, st, ins, bmi, dpf, a)
```

```
testinput = list(testfeatures)

# Predict the target variable for the test data using the trained model
predicted = model.predict(testinput)

# Calculate and print the confusion matrix
print("Confusion Matrix:")
print(metrics.confusion_matrix(d, predicted))

# Calculate and print the classification measures
print("\nClassification Measures:")
print("Accuracy:", metrics.accuracy_score(d, predicted))
print("Recall:", metrics.recall_score(d, predicted))
print("Precision:", metrics.precision_score(d, predicted))
print("F1-score:", metrics.f1_score(d, predicted))
```

Output



Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 11

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement Support Vector Machine(SVM).

Objectives

To classify breast cancer data using support vector machine(SVM).

Introduction

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

It takes input data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the data points into two classes.
This line or hyperplane is the decision boundary: any data points that falls to one side of it is classified in one class and, and the data points that falls to the other of it is classified in another class.
Support vectors are data points that are closest to the hyperplane and influence the position and orientation of the hyperplane.

Python Code

```
from sklearn import datasets
from sklearn.svm import SVC
from sklearn import metrics

# Load the breast cancer dataset
cancer = datasets.load_breast_cancer()
# Extract the features and target variables
x = cancer.data
y = cancer.target

# Print the length of the data
print("Length of Data:", len(cancer.data))

# Split the dataset into train and test sets
split = int(len(x) * 0.7)
trainx, testx = x[:split], x[split:]
trainy, testy = y[:split], y[split:]

# Print the number of features
print("Number of features: ", len(cancer.feature_names))

# Print the number of classes and class labels
print("Number of classes: ", len(cancer.target_names))
print("Class Labels: ", cancer.target_names)

# Create and train the SVM model
model = SVC(kernel="linear")
model.fit(trainx, trainy)

# Predict the target variable for the test data using the trained model
yp = model.predict(testx)

# Print the confusion matrix
print("\nConfusion Matrix:")
print(metrics.confusion_matrix(testy, yp))

# Print classification measures
print("\nClassification Measures:")
print("Accuracy:", metrics.accuracy_score(testy, yp))
print("Recall:", metrics.recall_score(testy, yp))
print("Precision:", metrics.precision_score(testy, yp))
print("F1-score:", metrics.f1_score(testy, yp))
```

Output

+ Code + Text All changes saved

```
[2]  # Print the length of the data
     print("Length of Data:", len(cancer.data))
```

```
Length of Data: 569
```

```
[3]  # Print the number of features
     print("Number of features: ", len(cancer.feature_names))
```

```
Number of features:  30
```

```
     # Print the number of classes and class labels
     print("Number of classes: ", len(cancer.target_names))
     print("Class Labels: ", cancer.target_names)
```

```
Number of classes:  2
Class Labels:  ['malignant' 'benign']
```

```
[5]  # Print the confusion matrix
     print("\nConfusion Matrix:")
     print(metrics.confusion_matrix(testy, yp))
```

```
Confusion Matrix:
[[ 39   0]
 [  9 123]]
```

```
[6]  # Print classification measures
     print("\nClassification Measures:")
     print("Accuracy:", metrics.accuracy_score(testy, yp))
     print("Recall:", metrics.recall_score(testy, yp))
     print("Precision:", metrics.precision_score(testy, yp))
     print("F1-score:", metrics.f1_score(testy, yp))
```

```
Classification Measures:
Accuracy: 0.9473684210526315
Recall: 0.9318181818181818
Precision: 1.0
F1-score: 0.9647058823529412
```

Madan Bhandari Memorial College
Department of Computer Science and Information Technology (B.Sc.CSIT)
Ninayak Nagar, New Baneshwor, Kathmandu

Practical Sheet

Submitted By:- Yubaraj Karki

Program No:- 12

Submitted To:- Raheem Ansari

Lab Date:-

Submission Date:-

T.U.Roll.No. :- 24181

Title: Program to Implement Apriori Algorithm

Introduction

It is a classic algorithm used in data mining for learning association rules.
Mining association rules basically means finding the items that are purchased together more frequently than others.
The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent item set properties.
Apriori employs an iterative approach known as a level-wise search, where frequent k-itemsets are used to explore frequent (k+1)-itemsets.

Algorithm

1. Define minimum support and confidence thresholds.
2. Prepare data in transaction format.
3. Generate 1-itemsets (frequent single items).
4. Iteratively generate candidate itemsets (join frequent itemsets).
5. Prune candidate itemsets (based on Apriori principle).
6. Count support for candidate itemsets.
7. Generate frequent itemsets (based on minimum support).
8. Repeat steps 4-7 until no new frequent itemsets are found.
9. Generate association rules from frequent itemsets.
10. Interpret results: analyze patterns and relationships.

Python Code

```
!pip install apyori
import io

from google.colab import files
uploaded = files.upload()

# Load the store dataset
dataset = pd.read_csv("store_data.csv", header=None)

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori

# Prepare the dataset for association rule mining
records = []
for i in range(0, 7501):
    test = []
    data = dataset.iloc[i]
    data = data.dropna()
    for j in range(0, len(data)):
        test.append(str(dataset.values[i, j]))
    records.append(test)

# Perform association rule mining using apriori algorithm
association_rules = apriori(
    records, min_support=0.005, min_confidence=0.2, min_lift=3, min_length=2
)
association_results = list(association_rules)

# Print the association rules
for item in association_results:
    print(list(item[2][0][0]), '->', list(item[2][0][1]))
```

Output

```
Requirement already satisfied: apyori in /usr/local/lib/python3.10/dist-packages (1.1.2)
['mushroom cream sauce'] -> ['escalope']
['pasta'] -> ['escalope']
['herb & pepper'] -> ['ground beef']
['tomato sauce'] -> ['ground beef']
['whole wheat pasta'] -> ['olive oil']
['pasta'] -> ['shrimp']
['frozen vegetables', 'chocolate'] -> ['shrimp']
['spaghetti', 'frozen vegetables'] -> ['ground beef']
['mineral water', 'shrimp'] -> ['frozen vegetables']
['spaghetti', 'frozen vegetables'] -> ['olive oil']
['spaghetti', 'frozen vegetables'] -> ['shrimp']
['spaghetti', 'frozen vegetables'] -> ['tomatoes']
['grated cheese', 'spaghetti'] -> ['ground beef']
['herb & pepper', 'mineral water'] -> ['ground beef']
['herb & pepper', 'spaghetti'] -> ['ground beef']
['shrimp', 'ground beef'] -> ['spaghetti']
['milk', 'spaghetti'] -> ['olive oil']
['mineral water', 'soup'] -> ['olive oil']
['pancakes', 'spaghetti'] -> ['olive oil']
```