EX.NO: 01	DATA MIGRATION USING INFORMATICA
DATE:	

AIM:

To study the Concept of Data Migration using Informatica.

DATA MIGRATION:

Data migration is the process of moving data from one location to another, one format to another, or one application to another. Generally, this is the result of introducing a new system or location for the data.

TYPES OF DATA MIGRATION:

There are numerous business advantages to upgrading systems or extending a data centre into the cloud.

a. **Storage migration:** The process of moving data off existing arrays into more modern ones that enable other systems to access it. Storage migrations occur when data is moved from one storage medium to another. Companies need to migrate their data to upgrade to newer technology or infrastructure offering significantly faster performance and more cost-effective scaling while enabling expected data management features such as cloning, snapshots, and backup and disaster recovery.

b. **Database migration:** Databases are data storage media where data is structured in an organized way. Databases are managed through database management systems (DBMSs). Hence, database migration involves moving from one DBMS to another – or upgrading from the current version of a DBMS to the latest version of the same DBMS. The former is more challenging if the source system and the target system use different data structures.

c. **Cloud migration:** Cloud data migration is the process of moving applications and data storage into the cloud. A migration initiative may involve consolidating on-premises data warehouses in the cloud or building new cloud data warehouses and data lakes. Whether you're moving existing data or building a new cloud data warehouse or cloud data lake from scratch, migrating to the cloud leverages the inherent benefits of cloud computing: fast provisioning, infinite scalability, per-use pricing, reduced infrastructure, and IT costs, seamless upgrades, and rapid technology innovation.

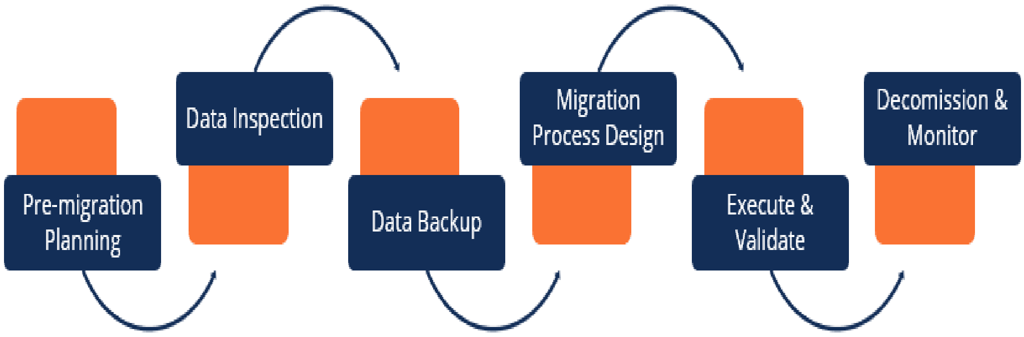
d. **Data centre migration:** Data centre migration relates to the migration of data centre infrastructure to a new physical location or the movement of data from the old data centre infrastructure to new infrastructure equipment at the exact physical location. A data centre houses the data storage infrastructure, which maintains the organization's critical applications. It consists of servers, network routers, switches, computers, storage devices, and related data equipment.

e. **Business process migration:** Business process migration requires the movement of business applications and data on business processes and metrics to a new environment. This type of migration is driven by mergers and acquisitions, business optimization, or reorganization to address competitive challenges or enter new markets. All these changes may require transferring business applications and databases with data on customers, products, and operations to the new environment.

f. **Application migration:** Application migration occurs when an organization goes through a change in application software or changes an application vendor. This migration requires moving data from one computing environment to another. For example, a company may be implementing a new HR system or switching from one CRM to another. Companies migrating applications must make sure their data can be synchronized between the two applications. Each application may have a unique data model, so you need to pay attention to formatting that data. After all, an application is only as good as the data within it. Companies have a variety of ways to migrate applications. For example, they may use middleware to bridge technology gaps or use scripts to migrate data automatically, or they could also use APIs to protect data integrity.

DATA MIGRATION PROCESS:

It involves the following steps in the planning, migration, and post-migration



phases:

STEPS:

Data migration involves 3 basic steps:

a. Extract data
b. Transform data
c. Load data

PROCESS:

It is 7 phase process.

- **Pre-migration planning:** Evaluate the data being moved for stability.
- **Project initiation:** Identify and brief key stakeholders.
- **Landscape analysis:** Establish a robust data quality rules management process and brief thebusiness on the goals of the project, including shutting down legacy systems.
- **Solution design:** Determine what data to move, and the quality of that data before and after themove.
- **Build & test:** Code the migration logic and test the migration with a mirror of the productionenvironment.
- **Execute & validate:** Demonstrate that the migration has complied with requirements and thatthe data moved is viable for business use.
- **Decommission & monitor:** Shut down and dispose of old systems.

CHARACTERISTICS OF DATA MIGRATION:

a. Complexity
b. Risk
c. Planning
d. Testing
e. Validation
f. Maintenance
g. Time-consuming

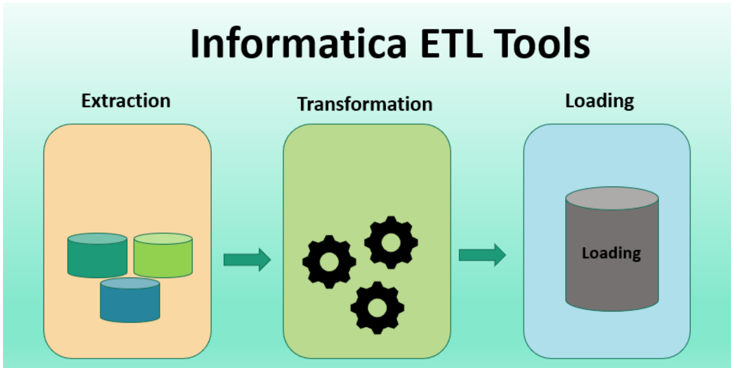
BENEFITS:

- Increased agility and flexibility.
- Ability to innovate faster.
- Easing of increasing resource demands.
- Better managing of increased customer expectations.
- Reduction in costs.
- Shift to everything as-a-service.

INFORMATICA:

Informatica Power Center is a popular data integration tool that provides a unified platform for accessing, discovering, and integrating data from various sources. It offers a range of features and functionalities for data migration, including:

1. **Extraction:** This involves extracting data from the source systems, which could be structured or unstructured data sources like databases, flat files, or XML files.
2. **Transformation:** This involves applying transformations to the extracted data to convert it into a format that can be easily processed and loaded into the target system.
3. **Loading:** This involves loading the transformed data into the target system, which could be adata warehouse or a data lake.



ROLES:

- To handle proper data storage for a company or organization.
- To Design and maintain data storage systems.

BENEFITS OF INFORMATICA ETL:

a. Integration.
b. High Performance.
c. Supports Different Databases.
d. Easy Maintenance.
e. Error Handling.

DATA MIGRATION USING INFORMATICA:

Data migration is a process of transferring data from one system to another system. In the context of Informatica, data migration involves transferring data from source systems to target systems using Informatica PowerCenter.

The process of data migration in Informatica typically involves the following steps:

a. **Analyzing the source data:** This involves analyzing the source data to identify the data types, formats, and structures.
b. **Designing the migration plan**: This involves designing the migration plan, which includes defining the scope of the migration, selecting the appropriate migration approach, and identifying the migration tools and technologies.
c. **Extracting the source data:** This involves extracting the data from the source systems using Informatica PowerCenter.
d. **Transforming the data:** This involves applying transformations to the extracted data to ensure that it is in a format that can be easily processed and loaded into the target system.
e. **Loading the data:** This involves loading the transformed data into the target system using Informatica PowerCenter.
f. **Verifying the data:** This involves verifying the data in the target system to ensure that it has been migrated correctly and is consistent with the source data.
g. **Testing the migration:** This involves testing the migration to ensure that it meets the performance, quality, and security requirements.

RESULT:

Thus the Concept of Data Migration using Informatica was studied successfully.

EX.NO: 02	IDENTIFICATION AND RETRIEVAL OF DATASET
DATE:	

AIM:

To study the Concept of Identification and Retrieval of Dataset Using Kaggle and UCI Repository.

DATASET:

1. A Dataset is a collection of related data that has been organized or structured in a particular wayto facilitate analysis and processing.
2. Datasets can come in various forms, including text files, spreadsheets, databases, and evenunstructured data like images or audio files.
3. In DWDM, datasets are typically preprocessed and transformed into a more structured formatbefore they can be analyzed.
4. A Dataset can be considered as a unit of data that represents a complete set of observations ormeasurements for a specific problem or analysis.
5. It may contain one or more tables, each of which is composed of rows (representing individual data points) and columns (representing different features or attributes).

IDENTIFICATION AND RETRIEVAL OF DATASET:

It involves finding relevant datasets for a specific purpose or analysis. Here are the steps involved in the process:

1. **Identify relevant sources:** Once the research question is defined, the next step is to identify relevant sources that may contain datasets related to the analysis. These sources may include data repositories, libraries, research institutions, or online search engines.
2. **Search for datasets:** After identifying relevant sources, the next step is to search for datasets thatmatch the specific requirements of the analysis. This may involve using keywords or specific search terms related to the research question.
3. **Evaluate the quality of datasets:** Once datasets are found, it's important to evaluate their quality by checking for completeness, accuracy, relevance, and timeliness.
4. **Retrieve datasets:** After evaluating the quality of datasets, the final step is to retrieve the selecteddatasets and store them in a format that is suitable for analysis.
5. It's important to note that identification and retrieval of datasets can be an iterative process that may require going back and forth between the steps until suitable datasets are found.
6. Additionally, it's crucial to comply with legal and ethical guidelines when accessing and using datasets.

KAGGLE TOOL:

Kaggle offers a community of over 4 million members who share datasets, compete in machine learning competitions, and collaborate on data science projects. Some of the key features of Kaggle include:

1. **Competitions:** Kaggle hosts machine learning competitions where users can compete to create the most accurate predictive models for a given problem. These competitions often offer cash prizes, recognition, and the opportunity to work on real-world problems.
2. **Datasets:** Kaggle provides a repository of over 20,000 public datasets, ranging from climate change data to medical imaging data. Users can also upload their own datasets to share with the community.
3. **Notebooks:** Kaggle offers an online platform for creating and sharing Jupyter notebooks, which allow users to write code, visualize data, and share insights with others.
4. **Discussion forums:** Kaggle has a discussion forum where users can ask and answer questions related to data science and machine learning.
5. Kaggle has become a popular platform for data scientists and machine learning practitioners to develop their skills, collaborate with others, and work on real-world problems.

IDENTIFICATION AND RETRIEVAL OF DATASET USING KAGGLE:

To identify and retrieve a dataset using Kaggle, you can follow these steps:

- Go to the Kaggle website at http://www.kaggle.com and sign in or create an account if you don't already have one.
- Once you are logged in, you can use the search bar at the top of the page to search for a specific dataset or browse through the available datasets by clicking on the "Datasets" option in the main menu.
- When you find a dataset that you are interested in, click on its title to open its details page. This page will provide you with information about the dataset, such as its description, format, size, and any associated files or scripts.
- If you decide to download the dataset, click on the "Download" button on the right side of the page. Depending on the size of the dataset, you may need to sign up for a Kaggle subscription or wait for a few minutes while the download completes.
- Once the download is complete, you can open the dataset files using your preferred data analysis tool or programming language.
- Note that some datasets on Kaggle may be subject to specific licenses or terms of use, so be sure to read and understand these before downloading and using the data.

UCI REPOSITORY:

1. The UCI (University of California, Irvine) Machine Learning Repository is a collection of databases, domain theories, and data generators used by the machine learning community for research and education purposes.
2. The datasets are preprocessed and formatted to facilitate their use in machine learning research and experimentation.
3. The repository also includes datasets for regression, time series, and image classification tasks.

- In addition to the datasets, the UCI Repository also provides information about the data sources, as well as metadata and documentation for each dataset.
- The UCI Repository is a valuable resource for researchers, educators, and students in the field of machine learning, providing access to a large collection of datasets for experimentation and analysis.

IDENTIFICATION AND RETRIEVAL OF DATASET USING UCI REPOSITORY:

To identify and retrieve a dataset from the UCI Machine Learning Repository, you can follow these steps:

1. Go to the UCI Machine Learning Repository website at https://archive.ics.uci.edu/ml/index.php.
2. On the homepage, you will find a list of datasets categorized by their characteristics. You can also use the search bar at the top of the page to search for a specific dataset by keywords or dataset name.
3. Click on the dataset title to open its details page, which will provide you with information about the dataset, such as its description, format, size, and any associated files or scripts.
4. On the details page, you will find a link to download the dataset in its original format. Click on the link to download the dataset.
5. Once the download is complete, you can open the dataset files using your preferred data analysis tool or programming language.

Note that some datasets on the UCI Machine Learning Repository may be subject to specific licenses or terms of use, so be sure to read and understand these before downloading and using the data.

RESULT:

Thus the Concept of Identification and Retrieval of Dataset Using Kaggle and UCI Repository was studied successfully.

EX.NO:03(a)	IMPLEMENTATION OF DATA PREPROCESSING TECHNIQUES IN WEKA TOOL
DATE:	

AIM:

To explore and implement various options in WEKA tool for pre-processing data.

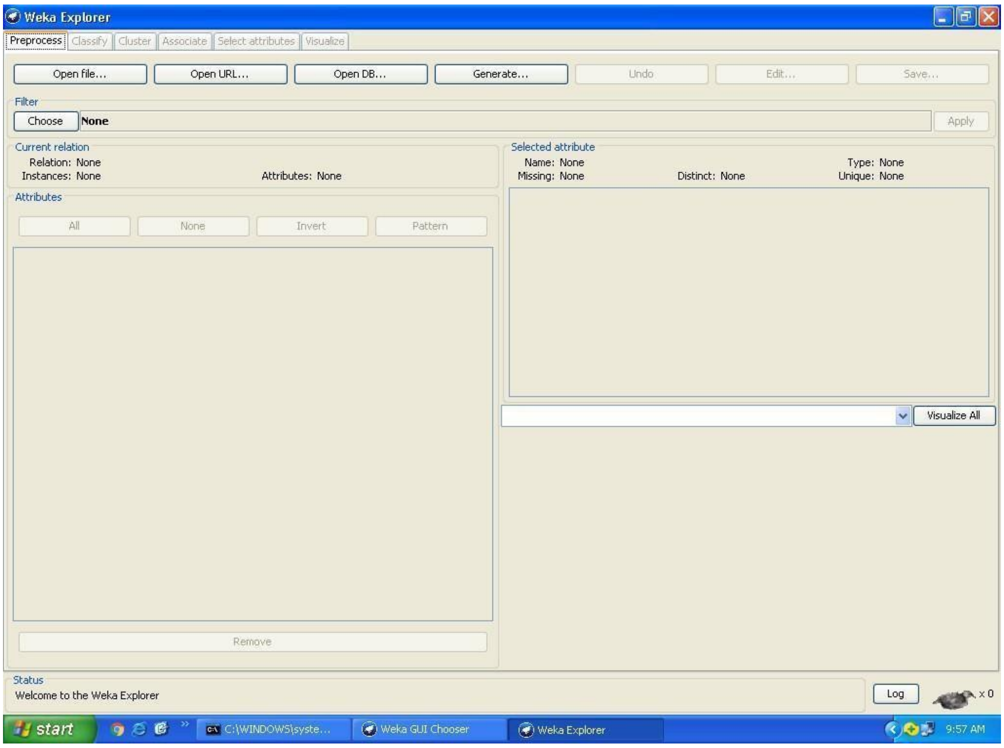
DESCRIPTION:

PRE-PROCESS TAB

1. **Loading Data:** The first four buttons at the top of the preprocess section enable you to load data into WEKA
2. **Open file:** Brings up a dialog box allowing you to browse for the data file on the local filesystem.
3. **Open URL:** Asks for a Uniform Resource Locator address for where the data is stored.
4. **Open DB:** Reads data from a database. (Note that to make this work you might have to edit the file in weka/experiment/DatabaseUtils.props.)
5. **Generate:** Enables you to generate artificial data from a variety of Data Generators. Using the Open file button you can read files in a variety of formats: WEKA's ARFF format, CSV format, C4.5 format, or serialized Instances format. ARFF files typically have an .arff extension, CSV files have a .csv extension, C4.5 files have a .data and .names extension, and serialized Instances objects have a .bsi extension.

Current Relation: Once some data has been loaded, the Preprocess panel shows a variety of information. The Current relation box (the "current relation" is the currently loaded data, which can be interpreted as a single relational table in database terminology) has three entries:

1. **Relation:**The name of the relation, as given in the file it was loaded from. Filters (describedbelow) modify the name of a relation.
2. **Instances:**The number of instances (data points/records) in the data.
3. **Attributes:**The number of attributes (features) in the data.



WORKING WITH ATTRIBUTES:

Below the Current relation box is a box titled Attributes. There are four buttons, and beneath themis a list of the attributes in the current relation. The list has three columns:

1. **No:** A number that identifies the attribute in the order they are specified in the data file.
2. **Selection tick boxes:** These allow you select which attributes are present in the relation.
3. **Name:** The name of the attribute, as it was declared in the data file. When you click on different rows in the list of attributes, the fields change in the box to the right titled selectedattribute. This box displays the characteristics ofthe currently highlighted attribute in the list:

1. **Name:** The name of the attribute, the same as that given in the attribute list.
2. **Type:** The type of attribute, most commonly Nominal or Numeric.
3. **Missing:** The number (and percentage) of instances in the data for which this attribute ismissing (unspecified).
4. **Distinct:** The number of different values that the data contains for this attribute.
5. **Unique:** The number (and percentage) of instances in the data having a value for this attribute that no other instances have.

Below these statistics is a list showing more information about the values stored in this attribute, which differ depending on its type.

- If the attribute is nominal, the list consists of each possible value for the attribute alongwith the number of instances that have that value.
- If the attribute is numeric, the list gives four statistics describing the distribution of valuesin the data— the minimum, maximum, mean and standard deviation.
- And below these statistics there is a coloured histogram, colour-coded according to the attribute chosen as the Class using the box above the histogram. (This box will bring up adrop-down list of available selections when clicked).
- Note that only nominal Class attributes will result in a colour-coding.
- Finally, after pressing the Visualize All button, histograms for all the attributes in the dataare shown in a separate window.
- Returning to the attribute list, to begin with all the tick boxes are unpicked.

PREPROCESSING

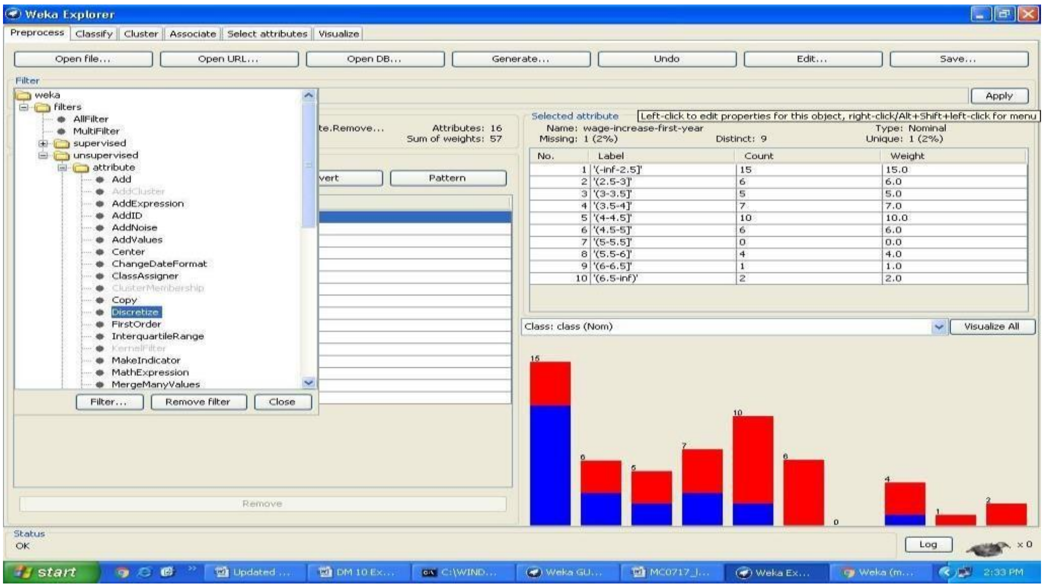
1. **All.** All boxes are ticked.
2. **None.** All boxes are cleared (unpicked).
3. **Invert.** Boxes that are ticked become unpicked and vice versa.
4. **Pattern.** Enables the user to select attributes based on a Perl 5 Regular Expression.

E.g., .* id selects all attributes which name ends with id.

Once the desired attributes have been selected, they can be removed by clicking the Remove button below the list of attributes. Note that this can be undone by clicking the Undo button, which is located next to the Edit button in the top-right corner of the Preprocess panel.

WORKING WITH FILTERS:

1. The preprocess section allows filters to be defined that transform the data in various ways.
2. The Filter box is used to set up the filters that are required.
3. At the left of the Filter box is a Choose button.
4. By clicking this button it is possible to select one of the filters in WEKA.
5. Once a filter has been selected, its name and options are shown in the field next to the Choose button. Clicking on this box with the left mouse button brings up a GenericObjectEditor dialogbox.
6. A click with the right mouse button (or Alt+Shift+left click) brings up a menu where you can choose, either to display the properties in a GenericObjectEditor dialog box, or to copy the current setup string to the clipboard.



THE GENERICOBJECTEDITOR DIALOG BOX:

1. The GenericObjectEditor dialog box lets you configure a filter.
2. The same kind of dialog box is used to configure other objects, such as classifiers and clusters.
3. The fields in the window reflect the available options.
4. Right-clicking (or Alt+Shift+Left-Click) on such a field will bring up a popup menu, listing thefollowing options:
    1. **Show properties:** has the same effect as left-clicking on the field, i.e., a dialog appears allowing you to alter the settings.
    2. **Copy configuration:** to clipboard copies the currently displayed configuration stringto the system's clipboard and therefore can be used anywhere else in WEKA or in the console. This israther handy if you have to setup complicated, nested schemes.
    3. **Enter configuration:** got copied to the clipboard earlier on. In this dialog you can enter a class name followed by options (if the class supports these). This also allows you to transfer a filter setting from the Preprocess panel to a Filtered Classifier used in the Classify panel.
5. Left-Clicking on any of these gives an opportunity to alter the filters settings.
6. For example, the setting may take a text string, in which case you type the string into thetext field provided.
7. Orit may give a drop-down box listing several states to choose from. Or it may do something else, depending on the information required.
8. Information on the options is provided in a tool tip if you let the mouse pointer hover ofthe corresponding field.
9. More information on the filter and its options can be obtained by clicking on the Morebutton in the about panel at the top of the GenericObjectEditor window.

STEPS FOR RUN PREPROCESSING TAB IN WEKA

Step 1: Open WEKA Tool.

Step 2: Click on WEKA Explorer.

Step 3: Click on Preprocessing tab button.

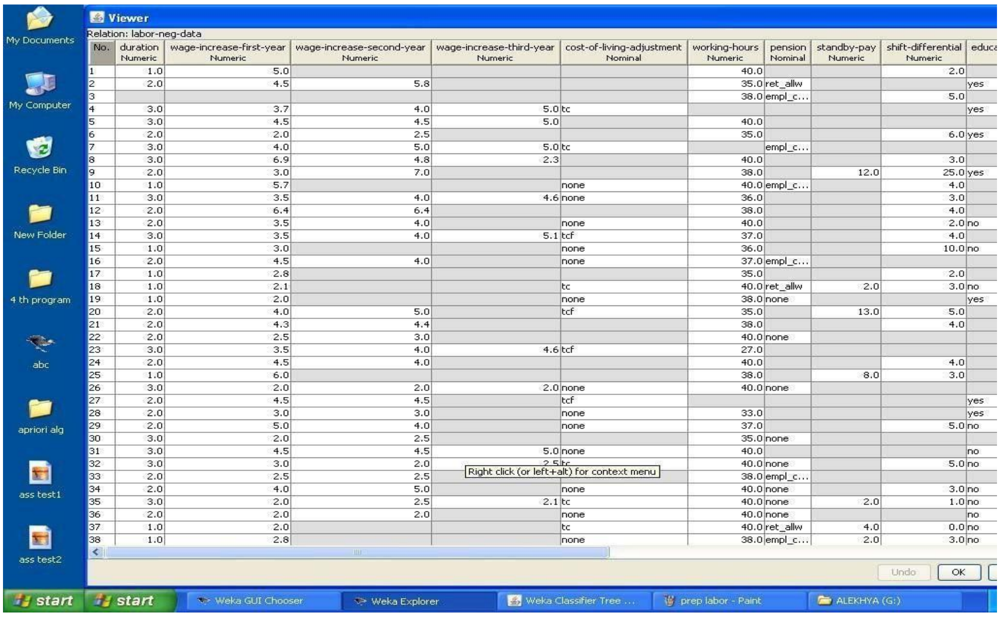
Step 4: Click on open file button.

Step 5: Choose WEKA folder in C drive.

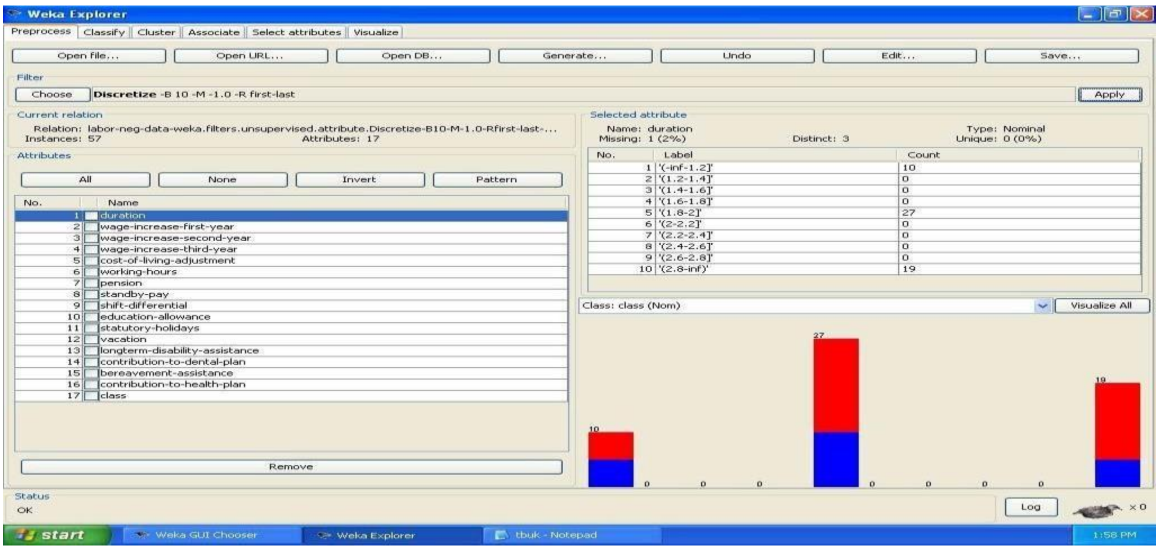
Step 6: Select and Click on data option button.

Step 7: Choose labor data set and open file.

Step 8: Choose filter button and select the Unsupervised-Discretize option and apply Dataset **“labor.arff”**



The following screenshot shows the effect of discretization.



LOAD EACH DATASET INTO WEKA AND RUN APRIORI ALGORITHM WITH DIFFERENT VALUES:

Steps for run Apriori algorithm in WEKA.

Step 1: Open WEKA Tool.

Step 2: Click on WEKA Explorer.

Step 3: Click on Preprocessing tab button.

Step 4: Click on open file button.

Step 5: Choose WEKA folder in C drive.

Step 6: Select and Click on data option button.

Step 7: Choose Weather data set and open file.

Step 8: Click on Associate tab and Choose Apriori algorithm

Step 9: Click on start button.

OUTPUT:

=== Run information ===

Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c • 1
Relation: weather.symbolic
Instances: 14

Attributes: 5
outlook
temperature
humidity windy
play

=== Associator model (full training set) ===Apriori =======

Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

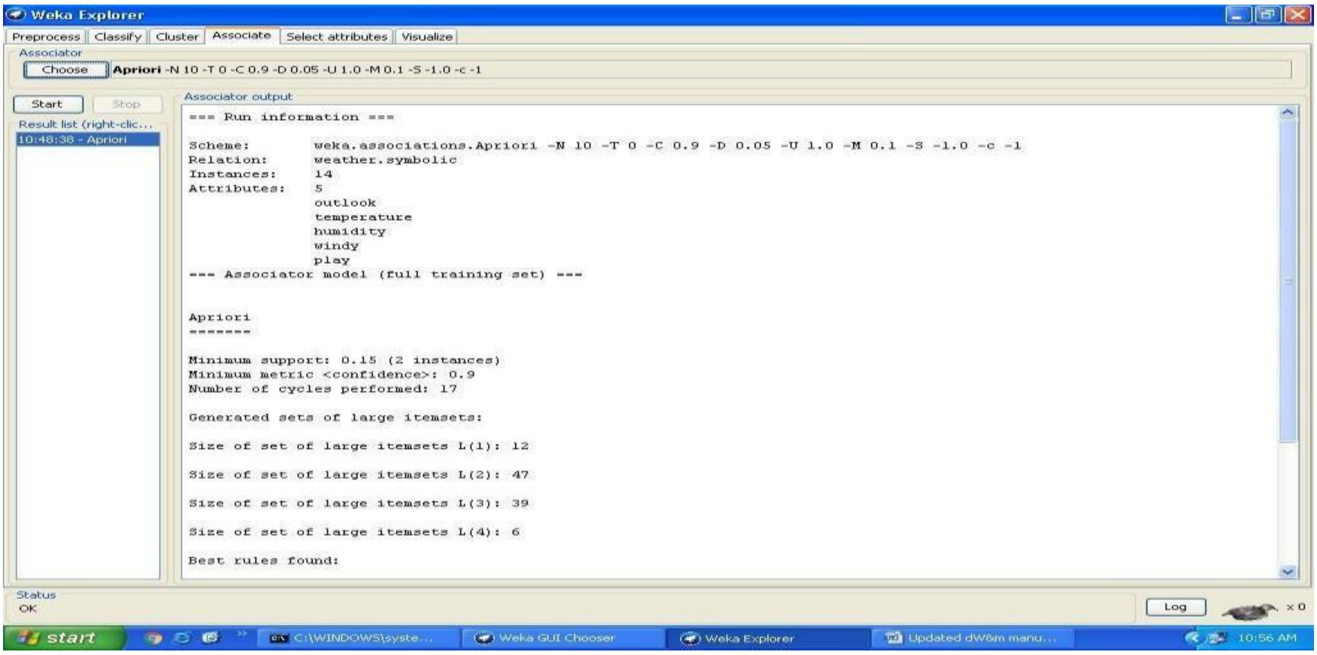
Size of set of large itemsets L(2): 47Size of set of large itemsets L(3): 39

Size of set of large itemsets L(4): 6

Best rules found:

1. outlook=overcast 4 ==> play=yes 4 conf:(1)
2. temperature=cool 4 ==> humidity=normal 4 conf:(1)
3. humidity=normal windy=FALSE 4 ==> play=yes 4 conf:(1)

4. outlook=sunny play=no 3 ==> humidity=high 3 conf:(1)
5. outlook=sunny humidity=high 3 ==> play=no 3 conf:(1)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 conf:(1)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3 conf:(1)
8. temperature=cool play=yes 3 ==> humidity=normal 3 conf:(1)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2 conf:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2 conf:(1)



APPLY DIFFERENT DISCRETIZATION FILTERS ON NUMERICAL ATTRIBUTES AND RUN THE APRIORI ASSOCIATION RULE ALGORITHM.

Step 1: Open WEKA Tool.

Step 2: Click on WEKA Explorer.

Step 3: Click on Preprocessing tab button.

Step 4: Click on open file button.

Step 5: Choose WEKA folder in C drive.

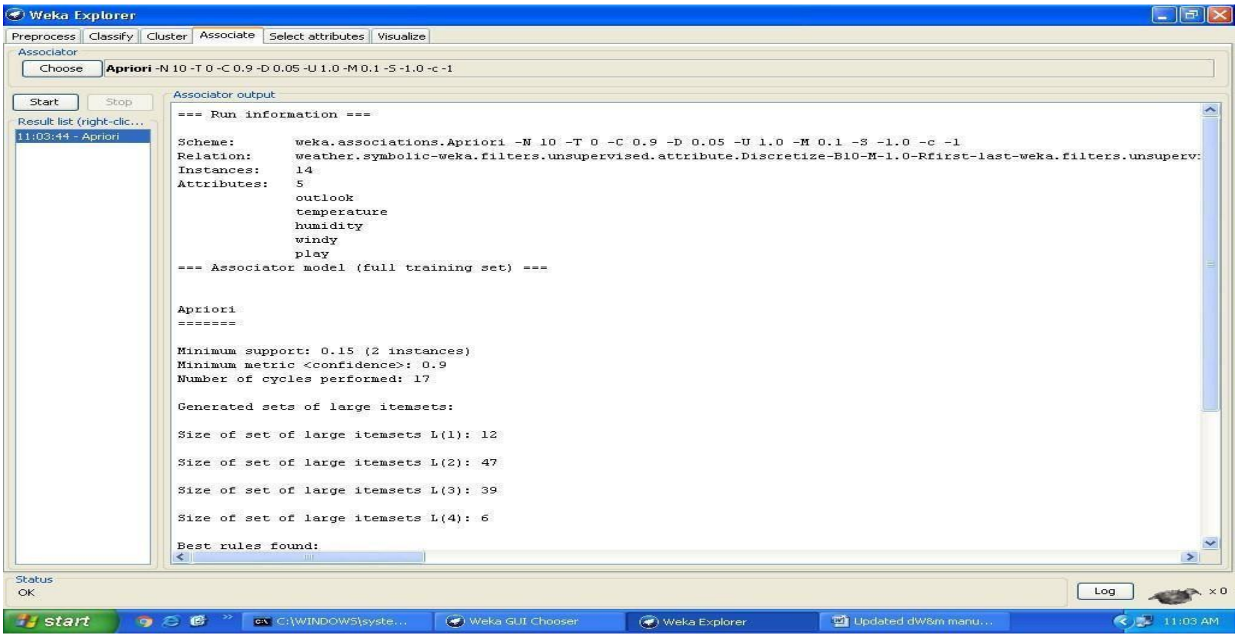
Step 6: Select and Click on data option button.

Step 7: Choose Weather data set and open file.

Step 8: Choose filter button and select the Unsupervised-Discritize option and apply

Step 9: Click on Associate tab and Choose Aprior algorithm

Step 10: Click on start button.



OUTPUT:

=== Run information ===

Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c•1
Relation: weather.symbolic
Instances: 14
Attributes: 5outlook temperature
humidity windy
play

=== Associator model (full training set) ===Apriori =======

Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17
Generated sets of large itemsets:
Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 47Size of
set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6

Best rules found:

1. outlook=overcast 4 ==> play=yes 4 conf:(1)
2. temperature=cool 4 ==> humidity=normal 4 conf:(1)
3. humidity=normal windy=FALSE 4 ==> play=yes 4 conf:(1)
4. outlook=sunny play=no 3 ==> humidity=high 3 conf:(1)
5. outlook=sunny humidity=high 3 ==> play=no 3 conf:(1)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 conf:(1)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3 conf:(1)

RESULT:

Thus the implementation of various options in WEKA tool for pre-processing data has been successfully executed.

EX.NO:03(b)	IMPLEMENTATION OF DATA PREPROCESSING ON TRAINING DATA SET
DATE:	

AIM:

To apply Pre-Processing techniques to the training data set of Employee Table.

DESCRIPTION:

- Real world databases are highly influenced to noise, missing and inconsistency due to their queue size so the data can be pre-processed to improve the quality of data and missing results and it also improves the efficiency.
- There are 3 pre-processing techniques they are:
  - ✓ Add
  - ✓ Remove
  - ✓ Normalization

CREATION OF EMPLOYEE TABLE:

PROCEDURE:

1. Open Start a Programs a Accessories a Notepad
2. Type the following training data set with the help of Notepad for Employee Table.

   @relation employee

   @attribute name {x,y,z,a,b}

   @attribute id numeric

   @attribute salary {low.medium high}

   @attribute exp numeric

   @attribute gender {male female}

   @attribute phone numeric

   @data

   x,101,low,2,male,250311

   y,102,high,3,female,251665

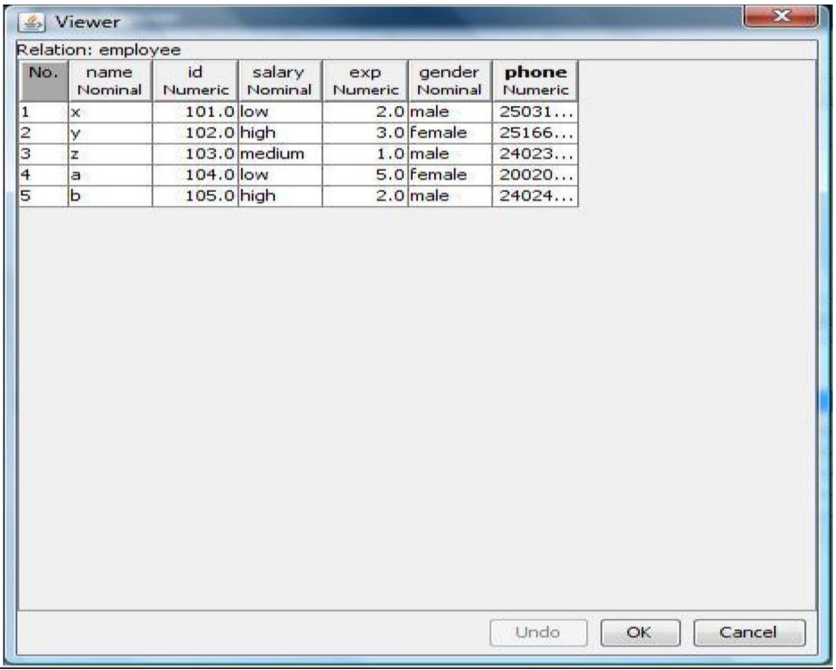
   z,103,medium,1,male,240238

   a,104,low,5,female,200200

   b,105,high,2,male,240240
3. After that the file is saved with employee.arff file format.
4. Minimize the arff file and then open Start à Programs a weka-3-4.
5. Click on weka-3-4, then Weka dialog box is displayed on the screen.

6. In that dialog box there are four modes, click on explorer.
7. Explorer shows many options. In that click on 'open file' and select the arff file.
8. Click on edit button which shows employee table on Weka.

TRAINING DATA SET AN EMPLOYEE TABLE:

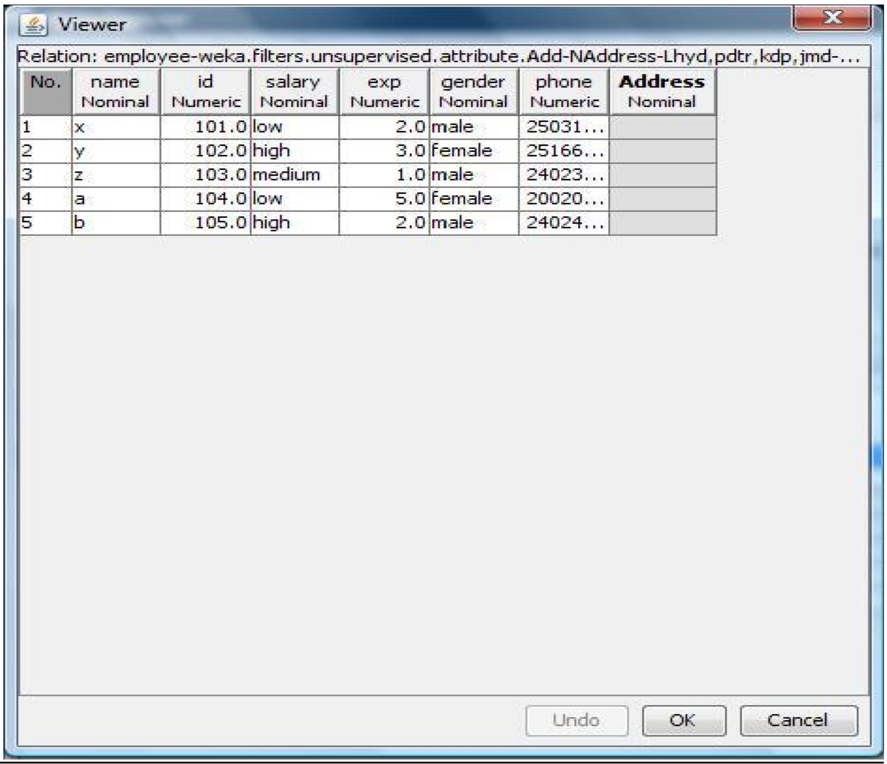


No.	name Nominal	id Numeric	salary Nominal	exp Numeric	gender Nominal	phone Numeric
1	x	101.0	low	2.0	male	25031...
2	y	102.0	high	3.0	female	25166...
3	z	103.0	medium	1.0	male	24023...
4	a	104.0	low	5.0	female	20020...
5	b	105.0	high	2.0	male	24024...

PROCEDURE:

1. Start à Program Weka-3.4.
2. Click on **explorer.**
3. Click on **open file.**
4. Select **employee.arff** file and click on open.
5. Click on **Choose button** and select the **Filters option**.
6. In Filters, we have **Supervised** and **Unsupervised data**.
7. Click on **Unsupervised data**.
8. Select the attribute **Add**.
9. A new window is opened.
10. In that we enter attribute index, type, data format, nominal label values for **Address**.
11. Click on **OK.**
12. Press the **Apply button**, then a new attribute is added to the Employee Table.
13. **Save** the file.
14. Click on the **Edit button**, it shows a new Employee Table on Weka.

EMPLOYEE TABLE AFTER ADDING NEW ATTRIBUTES ADDRESS:

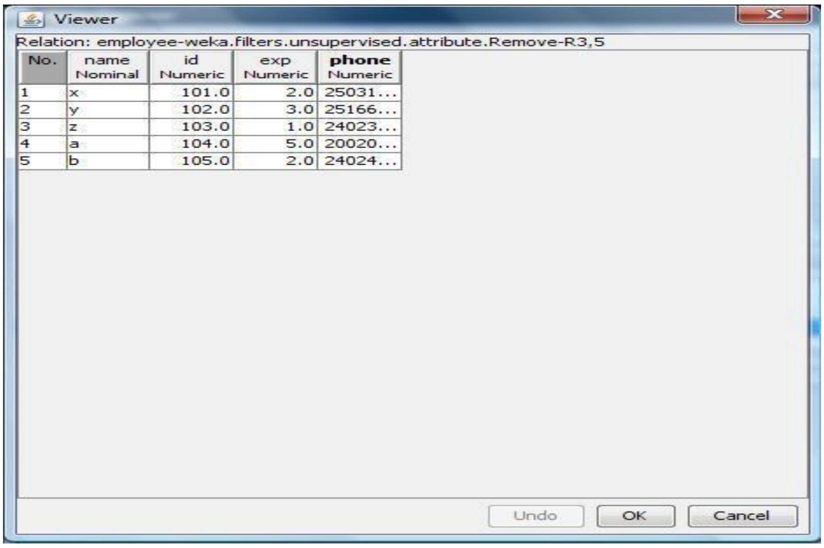


No.	name Nominal	id Numeric	salary Nominal	exp Numeric	gender Nominal	phone Numeric	Address Nominal
1	x	101.0	low	2.0	male	25031...	
2	y	102.0	high	3.0	female	25166...	
3	z	103.0	medium	1.0	male	24023...	
4	a	104.0	low	5.0	female	20020...	
5	b	105.0	high	2.0	male	24024...	

PROCEDURE TO REMOVE A PRE-PROCESSING TECHNIQUE:

1. Start a Programs Weka-3.4.
2. Click on **explorer.**
3. Click on **open file.**
4. Select **employee.arff** file and click on open.
5. Click on **Choose button** and select the **Filters option**.
6. In Filters, we have **Supervised** and **Unsupervised data**.
7. Click on **Unsupervised data**.
8. Select the attribute **Remove**.
9. Select the attributes **salary, gender** to Remove.
10. Click **Remove button** and then **Save**.
11. Click on the **Edit button**, it shows a new Employee Table on Weka.

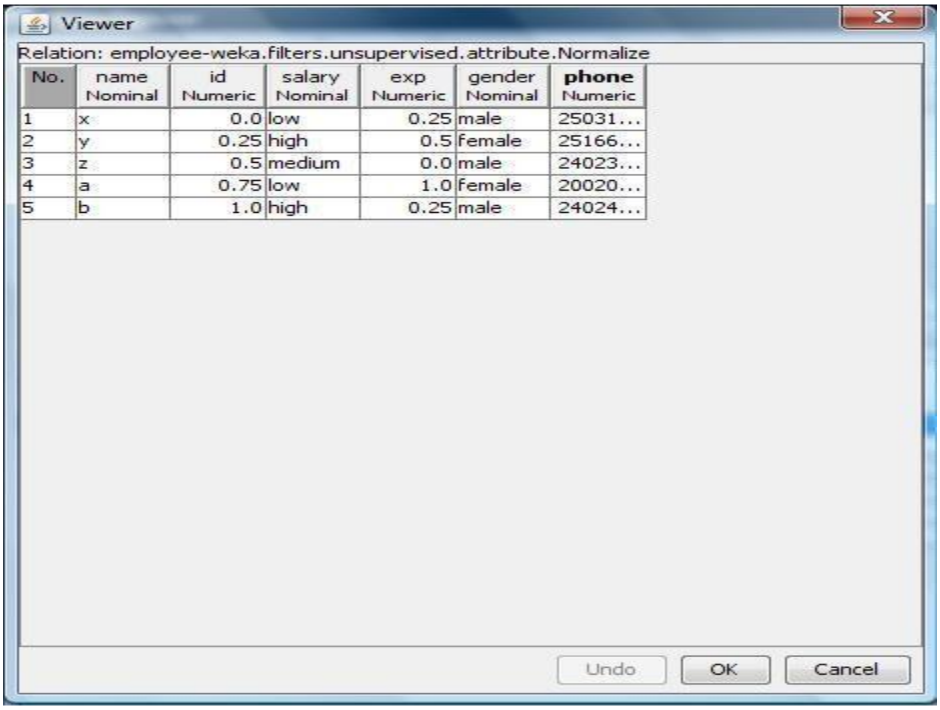
EMPLOYEE TABLE AFTER REMOVING ATTRIBUTES SALARY, GENDER:



PROCEDURE TO NORMALIZE A **PRE-PROCESSING TECHNIQUE:**

1. Start a Programs Weka-3.4.
2. Click on **explorer.**
3. Click on **open file.**
4. Select **employee.arff** file and click on open.
5. Click on **Choose button** and select the **Filters option**.
6. In Filters, we have **Supervised** and **Unsupervised data**.
7. Click on **Unsupervised data**.
8. Select the attribute **Normalize**.
9. Select the attributes **id, experience, phone** to Normalize.
10. Click on **Apply button** and then **save**.
11. Click on the **Edit button**, it shows a new Employee Table with normalized values on Weka.

EMPLOYEE TABLE AFTER NORMALIZING ID, EXP, PHONE:



No.	name Nominal	id Numeric	salary Nominal	exp Numeric	gender Nominal	phone Numeric
1	x	0.0	low	0.25	male	25031...
2	y	0.25	high	0.5	female	25166...
3	z	0.5	medium	0.0	male	24023...
4	a	0.75	low	1.0	female	20020...
5	b	1.0	high	0.25	male	24024...

RESULT:

Thus the implementation of Data Preprocessing on training Dataset has been successfully executed.

EX.NO:04(a)	IMPLEMENTATION OF RULE BASED CLASSIFICATION
DATE:	

AIM:

To write a Python program for the implementation of Rule Based classification Algorithm.

ALGORITHM:

Step 1: A simple rule based classifier will be developed.

Step 2: By using a series of if-else statements the input data can be classified based on the NDVI value.

Step 3: The script will output two images.

Step 4: The first contains an integer value associated with each class while the second contains an image coloured according to the output class that can be used for visualization.

Step 5: To start the script you need the same outline as the previous script with some minor modification.

Step 6: Firstly, the number of output images has been increased to two and the function which creates the output datasets now takes in a variable for the number of output bands in the datasets being created.

CODING:

```
# Import required libraries from Python
import sys, os, struct
# Import GDAL
from osgeo import gdal
```

```
# Define the GDALRuleClassifier class
class GDALRuleClassifier(object):
  # A function to create the output image with a given number of image bands.
  def createOutputImage(self, outFilename, inDataset,numOutBands):
    # Define the image driver to be used
    # This defines the output file format (e.g., GeoTiff)
    driver = gdal.GetDriverByName("GTiff")
    # Check that this driver can create a new file.
    metadata = driver.GetMetadata()
    if gdal.DCAP_CREATE in metadata and metadata[gdal.DCAP_CREATE] == 'YES':
```

```
            print('Driver GTiff supports Create() method.')
        else:
            print('Driver GTIFF does not support Create()')
            sys.exit(-1)
        # Get the spatial information from the input file
        geoTransform = inDataset.GetGeoTransform()
        geoProjection = inDataset.GetProjection()
        # Create an output file of the same size as the inputted
        # image but with numOutBands output image bands.
        newDataset = driver.Create(outFilename, inDataset.RasterXSize, \
                inDataset.RasterYSize, numOutBands, gdal.GDT_Float32)
        # Define the spatial information for the new image.
        newDataset.SetGeoTransform(geoTransform)
        newDataset.SetProjection(geoProjection)
        return newDataset
```

```
# The function which runs the classification.
    def classifyImage(self, filePath, outFilePathQKL, outFilePathSpatial):
        # Open the inputted dataset
        dataset = gdal.Open(filePath, gdal.GA_ReadOnly)
        # Check the dataset was successfully opened
        if dataset is None:
            print("The dataset could not be opened")
            sys.exit(-1)
```

```
        # Create the output dataset (Colored Image)
        outDatasetQKL = self.createOutputImage(outFilePathQKL, dataset, 3)
        # Check the datasets were successfully created.
        if outDatasetQKL is None:
            print('Could not create quicklook output image')
            sys.exit(-1)
```

```
        # Create the output dataset (Single band Image)
        outDataset = self.createOutputImage(outFilePathSpatial, dataset, 1)
        # Check the datasets were successfully created.
        if outDataset is None:
            print('Could not create output image')
            sys.exit(-1)
```

```
# Open the NDVI image band
    ndvi_band = dataset.GetRasterBand(1)  # NDVI BAND
    numLines = ndvi_band.YSize
    # Loop through the image lines
    for line in range(numLines):
      outputLine = ''
      outputLineR = ''
      outputLineG = ''
      outputLineB = ''
      # Read in data for the current line from the
      # image band representing the NDVI
      ndvi_scanline = ndvi_band.ReadRaster(0,line, ndvi_band.XSize, 1, \
                                    ndvi_band.XSize, 1, gdal.GDT_Float32)
      # Unpack the line of data to be read as floating point data
      ndvi_tuple = struct.unpack('f' * ndvi_band.XSize,ndvi_scanline)
```

```
# Loop through the row and assess each pixel.
      for i in range(len(ndvi_tuple)):
        # Initialize default class and color values for each pixel
        outClass = 0  # Output class
        red = 0  # Quantity of Red
        green = 0  # Quantity of Green
        blue = 0  # Quantity of Blue
        # Add the current pixel values to the output lines
        outputLine += struct.pack('f', outClass).decode('utf-8')
        outputLineR += struct.pack('B', red).decode('utf-8')
        outputLineG += struct.pack('B', green).decode('utf-8')
        outputLineB += struct.pack('B', blue).decode('utf-8')
```

```
# Write the completed lines to the output images
      outDataset.GetRasterBand(1).WriteRaster(0, line, ndvi_band.XSize, 1, \
                                    outputLine, buf_xsize=ndvi_band.XSize, \
                                    buf_ysize=1, buf_type=gdal.GDT_Float32)
      outDatasetQKL.GetRasterBand(1).WriteRaster(0, line, ndvi_band.XSize, 1, \
                                      outputLineR, buf_xsize=ndvi_band.XSize, \
                                      buf_ysize=1, buf_type=gdal.GDT_Byte)
      outDatasetQKL.GetRasterBand(2).WriteRaster(0, line, ndvi_band.XSize, 1, \
                                      outputLineG, buf_xsize=ndvi_band.XSize, \
                                      buf_ysize=1, buf_type=gdal.GDT_Byte)
```

```
        outDatasetQKL.GetRasterBand(3).WriteRaster(0, line, ndvi_band.XSize, 1, \
                                     outputLineB, buf_xsize=ndvi_band.XSize, \
                                     buf_ysize=1, buf_type=gdal.GDT_Byte)
      print('Classification Completed. Outputted to File')
```

```
# The function from which the script runs.
  def run(self):
    from google.colab import files
    # Upload the file
    uploaded = files.upload()
    # After uploading, you can access the uploaded files
    for filename in uploaded.keys():
      print('Uploaded file:', filename)
    # Define the input and output images
    filePath = list(uploaded.keys())[0]
    outFilePathQKL = "orthol7_20423xs100999_NDVI_classQK.tif"
    outFilePathSpatial = "orthol7_20423xs100999_NDVI_class.tif"
    # Check if the input file exists
    if os.path.exists(filePath):
      # Run the classifyImage function
      self.classifyImage(filePath, outFilePathQKL, outFilePathSpatial)
    else:
      print('The file does not exist.')
# Start the script by instantiating the GDALRuleClassifier class and calling the run function.
if __name__ == '__main__':
  obj = GDALRuleClassifier()
  obj.run()
```

```
    obj.run()
```

```
Choose Files Landsat_QK_204_23.jpg
• Landsat_QK_204_23.jpg(image/jpeg) - 81022 bytes, last modified: 4/20/2024 - 100% done
Saving Landsat_QK_204_23.jpg to Landsat_QK_204_23.jpg
Uploaded file: Landsat_QK_204_23.jpg
Driver GTiff supports Create() method.
Driver GTiff supports Create() method.
Classification Completed. Outputted to File
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
{x}
    Mounted at /content/drive
```

```
# The function which runs the classification.
def classifyImage(self, filePath, outFilePathQKL, outFilePathSpatial):
    # Open the inputted dataset
    dataset = gdal.Open(filePath, gdal.GA_ReadOnly)
    # Check the dataset was successfully opened
    if dataset is None:
        print("The dataset could not be opened")
        sys.exit(-1)
    # Create the output dataset (Coloured Image)
    outDatasetQKL = self.createOutputImage(outFilePathQKL, dataset, 3)
    # Check the dataset was successfully created
    if outDatasetQKL is None:
        print('Could not create quicklook output image')
        sys.exit(-1)
    # Create the output dataset (Single band Image)
    outDataset = self.createOutputImage(outFilePathSpatial, dataset, 1)
    # Check the dataset was successfully created
    if outDataset is None:
        print('Could not create output image')
        sys.exit(-1)
    # Open the NDVI image band
    ndvi_band = dataset.GetRasterBand(1)  # NDVI BAND
    numLines = ndvi_band.YSize
    # Define variables for pixel output
    outClass = 0
    red = 0
    green = 0
    blue = 0
    # Loop through the image lines
    for line in range(numLines):
        outputLine = ''
        outputLineR = ''
        outputLineG = ''
        outputLineB = ''
        # Read in data for the current line from the image band representing the NDVI
        ndvi_scanline = ndvi_band.ReadRaster(0, line, ndvi_band.XSize, 1,
                                   ndvi_band.XSize, 1, gdal.GDT_Float32)
```

```
# Unpack the line of data to be read as floating point data
ndvi_tuple = struct.unpack('f' * ndvi_band.XSize, ndvi_scanline)
# Loop through the row and assess each pixel.
for i in range(len(ndvi_tuple)):
  # If statements are used to encode the rules.
  if ndvi_tuple[i] < 0:
    outClass = 0  # Output class
    red = 200  # Quantity of Red
    green = 200  # Quantity of Green
    blue = 200  # Quantity of Blue
  elif 0 <= ndvi_tuple[i] < 0.3:
    outClass = 1
    red = 127
    green = 255
    blue = 212
  elif 0.3 <= ndvi_tuple[i] < 0.4:
    outClass = 2
    red = 0
    green = 145
    blue = 255
  # Add more elif statements for other classes
  # ...
  # Add the current pixel values to the output lines
  outputLine += struct.pack('f', outClass)
  outputLineR += struct.pack('B', red)
  outputLineG += struct.pack('B', green)
  outputLineB += struct.pack('B', blue)
# Write the completed lines to the output images
outDataset.GetRasterBand(1).WriteRaster(0, line, ndvi_band.XSize, 1 ,
                    outputLine, buf_xsize=ndvi_band.XSize,
                    buf_ysize=1, buf_type=gdal.GDT_Float32)
# Write other bands if necessary
# ...
outDatasetQKL.GetRasterBand(1).WriteRaster(0, line, ndvi_band.XSize, 1,
                      outputLineR, buf_xsize=ndvi_band.XSize,
                      buf_ysize=1, buf_type=gdal.GDT_Byte)
# Write other bands if necessary
# ...
```

```
        outDatasetQKL.GetRasterBand(2).WriteRaster(0, line, ndvi_band.XSize, 1,
                                    outputLineG, buf_xsize=ndvi_band.XSize,
                                    buf_ysize=1, buf_type=gdal.GDT_Byte)
        # Write other bands if necessary
        # ...
        outDatasetQKL.GetRasterBand(3).WriteRaster(0, line, ndvi_band.XSize, 1,
                                    outputLineB, buf_xsize=ndvi_band.XSize,
                                    buf_ysize=1, buf_type=gdal.GDT_Byte)
        # Write other bands if necessary
        # ...
        # Delete the output lines following write
        del outputLine
        del outputLineR
        del outputLineG
        del outputLineB
    print('Classification Completed Outputted to File')
```

```
    # The function which runs the classification.
    def ClassifyImage(self, filePath, outFilePathQKL, outFilePathSpatial):
        # Open the inputted dataset
        dataset = gdal.Open(filePath, gdal.GA_ReadOnly)
        # Check the dataset was successfully opened
        if dataset is None:
            print("The dataset could not be opened")
            sys.exit(-1)
```

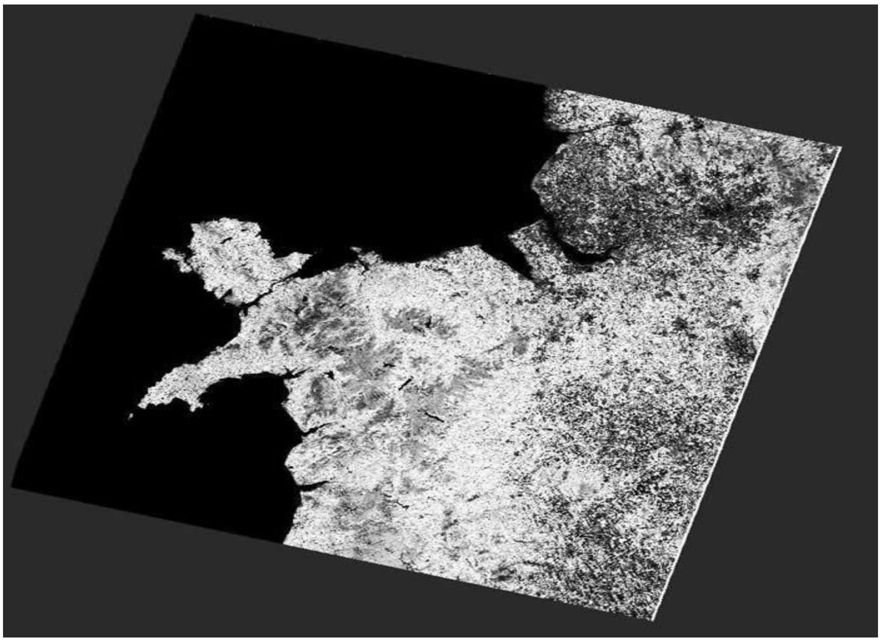
```
    # Create the output dataset (Coloured Image)
        outDatasetQKL = self.createOutputImage(outFilePathQKL, dataset, 3)
        # Check the dataset was successfully created
        if outDatasetQKL is None:
            print('Could not create quicklook output image')
            sys.exit(-1)
        # Create the output dataset (Single band Image)
        outDataset = self.createOutputImage(outFilePathSpatial, dataset, 1)
        # Check the dataset was successfully created
        if outDataset is None:
            print('Could not create output image')
            sys.exit(-1)
```

```
# Open the NDVI image band
ndvi_band = dataset.GetRasterBand(1)  # NDVI BAND
numLines = ndvi_band.YSize
# Define variables for pixel output
outClass = 0
red = 0
green = 0
blue = 0
# Loop through the image lines
for line in range(numLines):
   outputLine = ''
   outputLineR = ''
   outputLineG = ''
   outputLineB = ''
   # Read in data for the current line from the image band representing the NDVI
   ndvi_scanline = ndvi_band.ReadRaster(0, line, ndvi_band.XSize, 1,
                              ndvi_band.XSize, 1, gdal.GDT_Float32)
   # Unpack the line of data to be read as floating point data
   ndvi_tuple = struct.unpack('f' * ndvi_band.XSize, ndvi_scanline)

   # Loop through the row and assess each pixel.
   for i in range(len(ndvi_tuple)):
      # If statements are used to encode the rules.
      if ndvi_tuple[i] < 0:
         outClass = 0  # Output class
         red = 200  # Quantity of Red
         green = 200  # Quantity of Green
         blue = 200  # Quantity of Blue
      elif 0 <= ndvi_tuple[i] < 0.3:
         outClass = 1
         red = 127
         green = 255
         blue = 212
      elif 0.3 <= ndvi_tuple[i] < 0.4:
         outClass = 2
         red = 0
         green = 145
         blue = 255
```

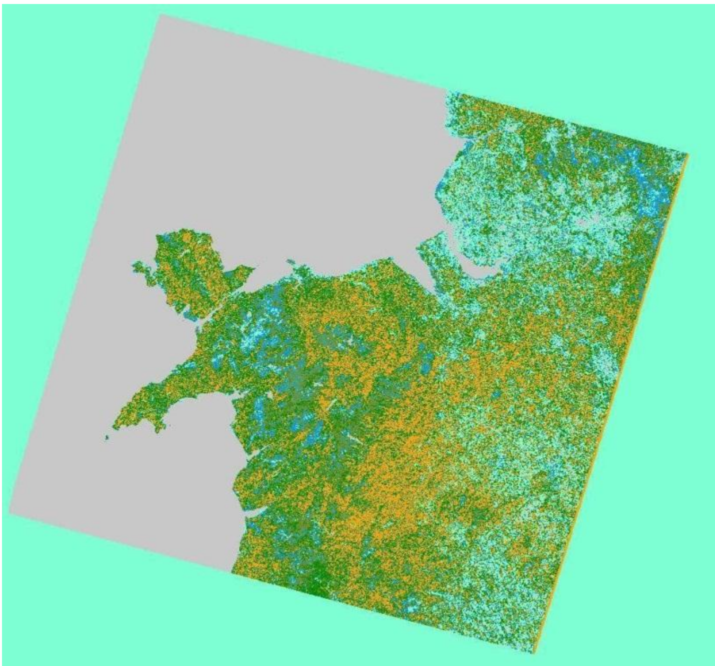
```
            # Add more elif statements for other classes
            # ...
            # Add the current pixel values to the output lines
            outputLine += struct.pack('f', outClass)
            outputLineR += struct.pack('B', red)
            outputLineG += struct.pack('B', green)
            outputLineB += struct.pack('B', blue)
        # Write the completed lines to the output images
        outDataset.GetRasterBand(1).WriteRaster(0, line, ndvi_band.XSize, 1,
                                                outputLine, buf_xsize=ndvi_band.XSize,
                                                buf_ysize=1, buf_type=gdal.GDT_Float32)
        # Write other bands if necessary
        # ...
        outDatasetQKL.GetRasterBand(1).WriteRaster(0, line, ndvi_band.XSize, 1,
                                                   outputLineR, buf_xsize=ndvi_band.XSize,
                                                   buf_ysize=1, buf_type=gdal.GDT_Byte)
        # Write other bands if necessary
        # ...
        outDatasetQKL.GetRasterBand(2).WriteRaster(0, line, ndvi_band.XSize, 1,
                                                   outputLineG, buf_xsize=ndvi_band.XSize,
                                                   buf_ysize=1, buf_type=gdal.GDT_Byte)
        # Write other bands if necessary
        # ...
        outDatasetQKL.GetRasterBand(3).WriteRaster(0, line, ndvi_band.XSize, 1,
                                                   outputLineB, buf_xsize=ndvi_band.XSize,
                                                   buf_ysize=1, buf_type=gdal.GDT_Byte)
        # Write other bands if necessary
        # ...
        # Delete the output lines following write
        del outputLine
        del outputLineR
        del outputLineG
        del outputLineB
    print('Classification Completed Outputted to File')
```

OUTPUT:

The class image from the rule based classification of the NDVI



The coloured image from the rule based classification of the NDVI



RESULT:

Thus the Python program for the implementation of Rule based classification was done and executed successfully.

EX.NO:04(b)	DECISION TREE IMPLEMENTATION
DATE:	

AIM:

To write a python program for the implementation of Decision tree algorithm.

ALGORITHM:

Step 1: Initialize the decision tree classifier object.

Step 2: Train the decision tree classifier on the training dataset using the fit() method.

Step 3: Make predictions on the testing dataset using the predict() method.

Step 4: Calculate the accuracy of the model on the testing dataset using the score() method.

Step 5: Return the prediction accuracy.

CODING:

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

```
# Function importing Dataset
def importdata():
    balance_data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/balance-scale/balance-scale.data', sep=',', header=None)
    print("Dataset Length: ", len(balance_data))
    print("Dataset Shape: ", balance_data.shape)
    print("Dataset: ", balance_data.head())
    return balance_data
```

```
# Function to split the dataset
def splitdataset(balance_data):
    X = balance_data.values[:, 1:5]
    Y = balance_data.values[:, 0]
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=100)
```

```
    return X, Y, X_train, X_test, y_train, y_test

# Function to perform training with giniIndex
def train_using_gini(X_train, X_test, y_train):
    clf_gini = DecisionTreeClassifier(criterion="gini", random_state=100, max_depth=3,
min_samples_leaf=5)
    clf_gini.fit(X_train, y_train)
    return clf_gini

# Function to perform training with entropy
def train_using_entropy(X_train, X_test, y_train):
    clf_entropy = DecisionTreeClassifier(criterion="entropy", random_state=100, max_depth=3,
min_samples_leaf=5)
    clf_entropy.fit(X_train, y_train)
    return clf_entropy

# Function to make predictions
def prediction(X_test, clf_object):
    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
    return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):
    print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
    print("Accuracy : ", accuracy_score(y_test,y_pred)*100)
    print("Report : ", classification_report(y_test, y_pred))

# Driver code
def main():
    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = train_using_entropy(X_train, X_test, y_train)

    print("Results Using Gini Index:")
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)

    print("Results Using Entropy:")
    y_pred_entropy = prediction(X_test, clf_entropy)
    cal_accuracy(y_test, y_pred_entropy)

# Calling main function
```

```
if __name__ == "__main__":
    main()
```

OUTPUT:

DATA INFORMATION :

Dataset Length: 625

Dataset Shape: (625, 5)

Dataset: 0 1 2 3 4

0 B	1	1	1	1
1 R	1	1	1	2
2 R	1	1	1	3
3 R	1	1	1	4
4 R	1	1	1	5

RESULTS USING GINI INDEX :

Predicted values:

['R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'L'
'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L'
'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'L' 'R'
'R' 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'R'
'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'R'
'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L'
'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
'L' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R'
'L' 'R' 'R' 'L' 'L' 'R' 'R' 'R']

CONFUSION MATRIX :

[[0 6 7]
[0 67 18]
[0 19 71]]

ACCURACY : 73.4042553191

REPORT :

	PRECISION	RECALL	F1-SCORE	SUPPORT
B	0.00	0.00	0.00	13
L	0.73	0.79	0.76	85
R	0.74	0.79	0.76	90
avg/total	0.68	0.73	0.71	188

RESULTS USING ENTROPY:

Predicted values:

['R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L'
'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L'
'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L'
'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R'
'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'R'
'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L'
'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'L' 'R'
'R' 'R' 'L' 'L' 'L' 'R' 'R' 'R']

CONFUSION MATRIX:

[[0 6 7]
[0 63 22]
[0 20 70]]

ACCURACY : 70.7446808511

REPORT :

	PRECISION	**RECALL**	**F1-SCORE**	**SUPPORT**
B	0.00	0.00	0.00	13
L	0.71	0.74	0.72	85
R	0.71	0.78	0.74	90
avg/total	0.66	0.71	0.68	188

RESULT:

Thus the Python program for the Implementation of Python decision tree is executed successfully.

EX.NO:04(c)	IMPLEMENTATION OF NAIVE BAYES CLASSIFICATION
DATE:	

AIM:

To write a Python program for the implementation of Naïve Bayes classification.

ALGORITHM:

STEP 1: Initialize the Naive Bayes classifier object.

STEP 2: Train the Naive Bayes classifier on the training dataset using the fit() method.

STEP 3: Make predictions on the testing dataset using the predict() method.

STEP 4: Calculate the accuracy of the model on the testing dataset using the score() method.

STEP 5: Return the prediction accuracy.

INTRODUCTION TO NAIVE BAYES

- Naive Bayes is among one of the very simple and powerful algorithms for classification based on Bayes Theorem with an assumption of independence among the predictors.
- The Naive Bayes classifier assumes that the presence of a feature in a class is not related to any other feature.
- Naive Bayes is a classification algorithm for binary and multi-class classification problems.

BAYES THEOREM

- Based on prior knowledge of conditions that may be related to an event, Bayes theorem describes the probability of the event
- conditional probability can be found this way
- Assume we have a Hypothesis(H) and evidence(E),
- According to Bayes theorem, the relationship between the probability of Hypothesis before getting the evidence represented as P(H) and the probability of the hypothesis after getting the evidence represented as P(H|E) is:

$$P(H|E) = P(E|H)*P(H)/P(E)$$

i. Prior probability = P(H) is the probability before getting the evidence

ii. Posterior probability = P(H|E) is the probability after getting evidence

In general,

P(class|data) = (P(data|class) * P(class)) / P(data)

BAYES THEOREM EXAMPLE

1. Assume we have to find the probability of the randomly picked card to be king given that it is aface card.
2. There are 4 Kings in a Deck of Cards which implies that P(King) = 4 as all the Kings are faceCards so P(Face|King) = 1
3. There are 3 Face Cards in a Suit of 13 cards and there are 4 Suits in total so,

P(Face)= 12/52

Therefore,

P(King|face) = P(face|king)*P(king)/P(face) = 1/3

CODING:

```
# @title Default title text
def encode_class(mydata):
    classes = [ ]
    for i in range(len(mydata)):
        if mydata[i][-1] not in classes:
            classes.append(mydata[i][-1])
    for i in range(len(classes)):
        for j in range(len(mydata)):
            if mydata[j][-1] == classes[i]:
                mydata[j][-1] = i
    return mydata
```

```
import math
import random
import pandas as pd
import numpy as np
```

```
def splitting(mydata, ratio):
    train_num = int(len(mydata) * ratio)
    train = []
    test = list(mydata)
    while len(train) < train_num:
        index = random.randrange(len(test))
        train.append(test.pop(index))
    return train, test
```

```
def groupUnderClass(mydata):
    data_dict = {}
    for i in range(len(mydata)):
        if mydata[i][-1] not in data_dict:
```

```
            data_dict[mydata[i][-1]] = []
        data_dict[mydata[i][-1]].append(mydata[i])
    return data_dict
```

```
def MeanAndStdDev(numbers):
    avg = np.mean(numbers)
    stddev = np.std(numbers)
    return avg, stddev

def MeanAndStdDevForClass(mydata):
    info = {}
    data_dict = groupUnderClass(mydata)
    for classValue, instances in data_dict.items():
        info[classValue] = [MeanAndStdDev(attribute) for attribute in zip(*instances)]
    return info
```

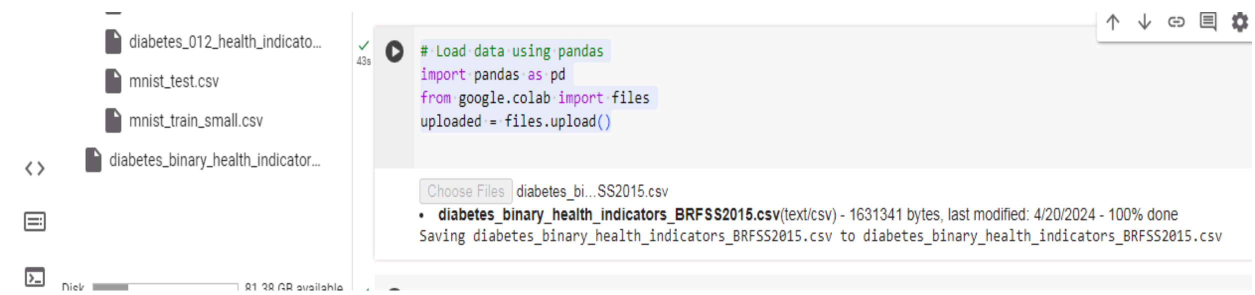
```
def calculateGaussianProbability(x, mean, stdev):
    epsilon = 1e-10
    expo = math.exp(-(math.pow(x - mean, 2) / (2 * math.pow(stdev + epsilon, 2))))
    return (1 / (math.sqrt(2 * math.pi) * (stdev + epsilon))) * expo

def calculateClassProbabilities(info, test):
    probabilities = {}
    for classValue, classSummaries in info.items():
        probabilities[classValue] = 1
        for i in range(len(classSummaries)):
            mean, std_dev = classSummaries[i]
            x = test[i]
            probabilities[classValue] *= calculateGaussianProbability(x, mean, std_dev)
    return probabilities
```

```
def predict(info, test):
    probabilities = calculateClassProbabilities(info, test)
    bestLabel = max(probabilities, key=probabilities.get)
    return bestLabel

def getPredictions(info, test):
    predictions = [predict(info, instance) for instance in test]
    return predictions
```

```
def accuracy_rate(test, predictions):
    correct = sum(1 for i in range(len(test)) if test[i][-1] == predictions[i])
```

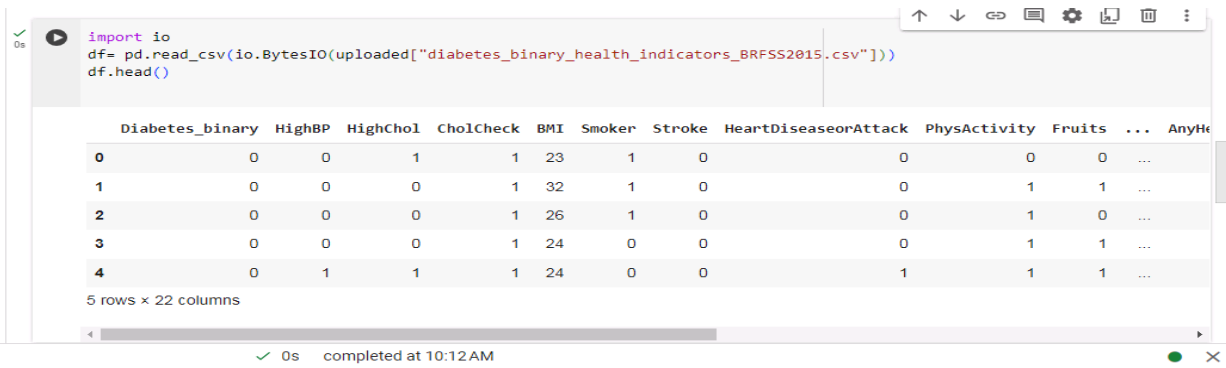
```
    return (correct / float(len(test))) * 100.0
```

```
# Load data using pandas
import pandas as pd
from google.colab import files
uploaded = files.upload()
```




```
import pandas as pd
```

```
import io
df= pd.read_csv(io.BytesIO(uploaded["diabetes_binary_health_indicators_BRFSS2015.csv"]))
df.head()
```



	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...
0	0	0	1	1	23	1	0	0	0	0	...
1	0	0	0	1	32	1	0	0	1	1	...
2	0	0	0	1	26	1	0	0	1	0	...
3	0	0	0	1	24	0	0	0	1	1	...
4	0	1	1	1	24	0	0	1	1	1	...

5 rows × 22 columns

```
mydata = df.values.tolist()

# Encode classes and convert attributes to float
mydata = encode_class(mydata)
for i in range(len(mydata)):
    for j in range(len(mydata[i]) - 1):
        mydata[i][j] = float(mydata[i][j])
```

```
# Split the data into training and testing sets
ratio = 0.7
train_data, test_data = splitting(mydata, ratio)

print('Total number of examples:', len(mydata))
print('Training examples:', len(train_data))
print('Test examples:', len(test_data))
```

Uploaded CSV dataset File: "diabetes_binary_health_indicators_BRFSS2015.csv" From Kaggle.

OUTPUT:

Total number of examples: 34999

Training examples: 24499

Test examples: 10500

```
# Train the model
info = MeanAndStdDevForClass(train_data)

# Test the model
predictions = getPredictions(info, test_data)
accuracy = accuracy_rate(test_data, predictions)
print('Accuracy of the model:', accuracy)
```

OUTPUT:

Accuracy of the model: 100.0

RESULT:

Thus the Python program for the Implementation of Naive Bayes classification was executed successfully.

EX.NO:05(a)	IMPLEMENTATION OF AGGLOMERATIVE CLASSIFICATION
DATE:	

AIM:

To write a Python program for the Implementation of Agglomerative Clustering.

ALGORITHM:

STEP 1: Import the required libraries: NumPy, Scikit-learn, and Matplotlib.

STEP 2: Generate a synthetic dataset using make_classification() function from Scikit-learn.

STEP 3: Define the AgglomerativeClustering() model with the desired number of clusters.

STEP 4: Fit the model to the dataset using fit_predict() method.

STEP 5: Predict the clusters for the dataset using the previously trained model.

STEP 6: Retrieve the unique cluster labels from the predicted clusters.

STEP 7: For each cluster in the unique clusters, get the row indexes of samples with that cluster using where() function.

STEP 8: Create a scatter plot of the samples with the same cluster label using pyplot.scatter() function.

STEP 9: Display the scatter plot using pyplot.show() function.

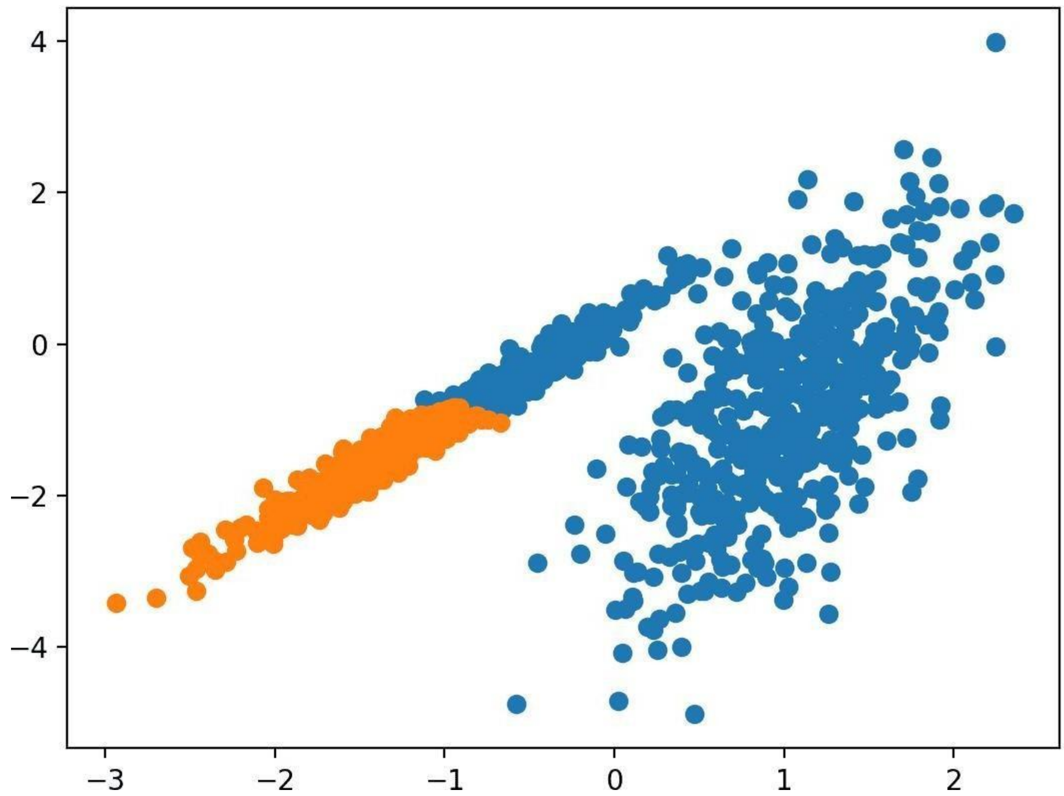
CODING:

```
from sklearn.datasets import make_classification
from sklearn.cluster import AgglomerativeClustering
import matplotlib.pyplot as plt

# Define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0,
                    n_clusters_per_class=1, random_state=4)

# Define the model
model = AgglomerativeClustering(n_clusters=2)  # You can change the number of clusters
here

# Fit and predict
yhat = model.fit_predict(X)

import numpy as np
```

```
# Retrieve unique clusters
clusters = np.unique(yhat)
```

```
import matplotlib.pyplot as plt
import numpy as np

# Plotting the clusters
for cluster in clusters:
    # Get row indexes for samples with this cluster
    row_ix = np.where(yhat == cluster)[0]
    # Plot the data points
    plt.scatter(X[row_ix, 0], X[row_ix, 1])

# Show the plot
plt.show()
```

OUTPUT:



RESULT:

Thus the Python program for the Implementation of Agglomerative Clustering was executed successfully.

EX.NO: 05(b)	IMPLEMENTATION OF BIRCH CLUSTERING
DATE:	

AIM:

To write a Python program for the Implementation of BIRCH clustering.

ALGORITHM:

STEP 1: First, it imports the required libraries: numpy, scikit-learn, and matplotlib.

STEP 2: Then, it generates a synthetic dataset using the make_classification() function from scikit-learn.

STEP 3: Next, it creates an instance of the Birch model with the specified hyper parameters threshold=0.01 and n_cluster=2.

STEP 4: It fits the model on the dataset using the fit() method.

STEP 5: It assigns a cluster label to each sample using the predict() method.

STEP 6: It retrieves the unique cluster labels using the unique() function from numpy.

STEP 7: It creates a scatter plot for each cluster, where each sample is colored according to its predicted cluster label using the scatter() function from matplotlib.

CODING:

```
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import Birch
from matplotlib import pyplot
```

```
# define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2,
n_redundant=0,
n_clusters_per_class=1, random_state=4)
```

```
# define the model
model = Birch(threshold=0.01, n_clusters=2)
```

```
# fit the model
model.fit(X)
# assign a cluster to each example
```

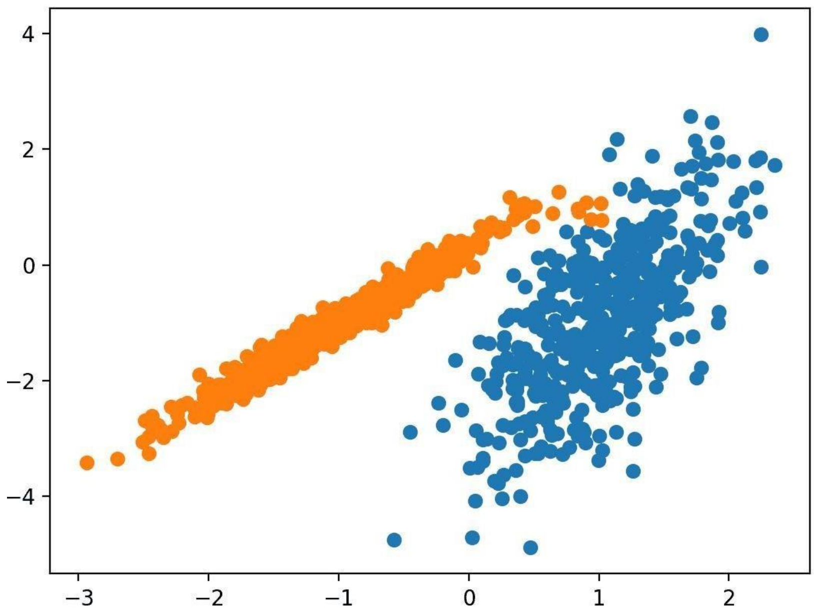
```
Birch
Birch(n_clusters=2, threshold=0.01)
```

```
# Assign a cluster to each example
yhat = model.predict(X)
```

```
# Retrieve unique clusters
clusters = unique(yhat)
```

```
import matplotlib.pyplot as plt
# Plotting the clusters
for cluster in clusters:
   # Get row indexes for samples with this cluster
   row_ix = where(yhat == cluster)[0]
   # Create scatter of these samples
   plt.scatter(X[row_ix, 0], X[row_ix, 1])
```

OUTPUT:



RESULT:

Thus the Python program for the Implementation of BIRCH clustering was executed successfully.

EX.NO: 05(c)	IMPLEMENTATION OF DBSCAN CLUSTERING
DATE:	

AIM:

To write a Python program for the Implementation of DBSCAN Clustering.

ALGORITHM:

STEP 1: First, it imports the required libraries: numpy, scikit-learn, and matplotlib.

STEP 2: Then, it generates a synthetic dataset using the make_classification() function from scikit-learn.

STEP 3: Next, it creates an instance of the DBSCAN model with the specified Hyper parameters: eps=0.30 and min_samples=9.

STEP 4: It fits the model on the dataset and predicts the cluster labels for each sample using the fit_predict() method.

STEP 5: It retrieves the unique cluster labels using the unique() function from numpy.

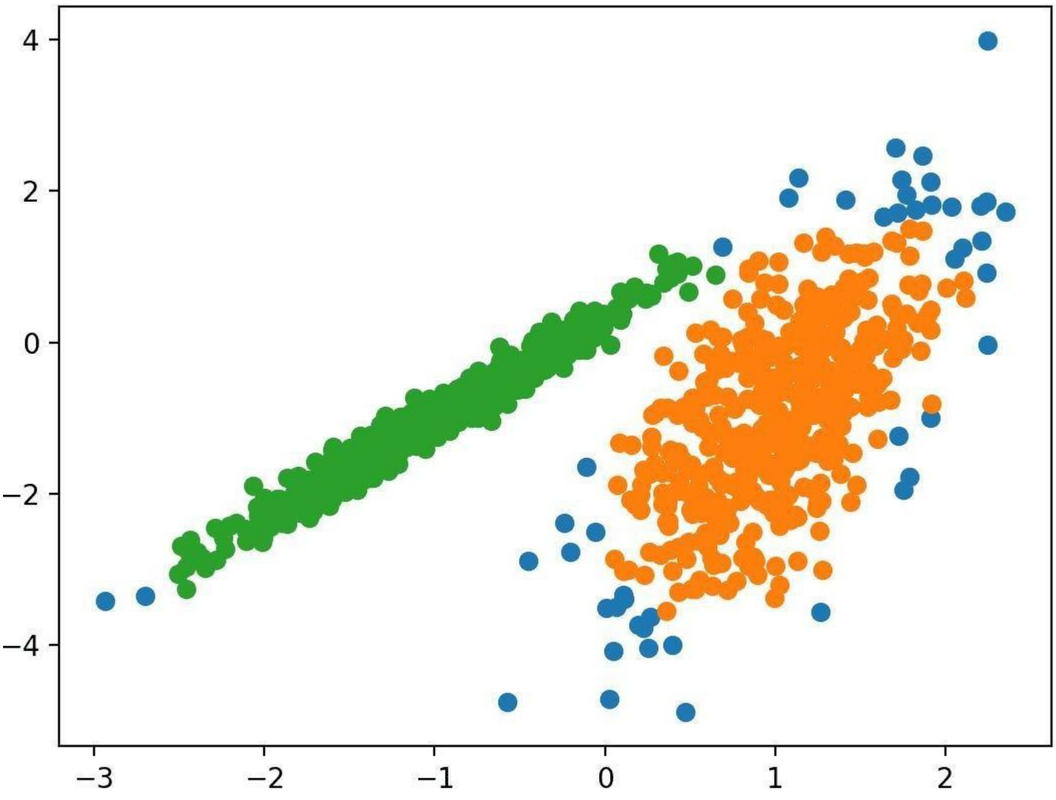
STEP 6: It creates a scatter plot for each cluster, where each sample is colored according to its predicted cluster label using the scatter() function from matplotlib.

CODING:

```
from numpy import unique, where
from sklearn.datasets import make_classification
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt

# Define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0,
        n_clusters_per_class=1, random_state=4)

# Define the model
model = DBSCAN(eps=0.30, min_samples=9)

# Fit model and predict clusters
yhat = model.fit_predict(X)

# Retrieve unique clusters
clusters = unique(yhat)
```

```
# Create scatter plot for samples from each cluster
for cluster in clusters:
    # Get row indexes for samples with this cluster
    row_ix = where(yhat == cluster)[0]
    # Create scatter of these samples
    plt.scatter(X[row_ix, 0], X[row_ix, 1])
```

OUTPUT:



RESULT:

Thus the Python program for the Implementation of DBSCAN clustering was executed successfully.

EX.NO:05(d)	IMPLEMENTATION OF K-MEANS CLUSTERING
DATE:	

AIM:

To write a Python program for the Implementation of K-MEANS clustering.

ALGORITHM:

STEP 1: First, it imports the required libraries: numpy, scikit-learn, and matplotlib.

STEP 2: Then, it generates a synthetic dataset using the make_classification() function from scikit-learn.

STEP 3: Next, it creates an instance of the KMeans model with the specified hyper parameter: n_clusters=3.

STEP 4: It fits the model on the dataset using the fit() method.

STEP 5: It assigns a cluster label to each sample using the predict() method.

STEP 6: It retrieves the unique cluster labels using the unique() function from numpy.

STEP 7: It creates a scatter plot for each cluster, where each sample is colored according to its predicted cluster label using the scatter() function from matplotlib.

CODING:

```
# k-means clustering
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import KMeans
from matplotlib import pyplot
```

```
# define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0,
n_clusters_per_class=1, random_state=4)
```

```
# define the model
model = KMeans(n_clusters=3)
```

```
# fit the model
model.fit(X)
```

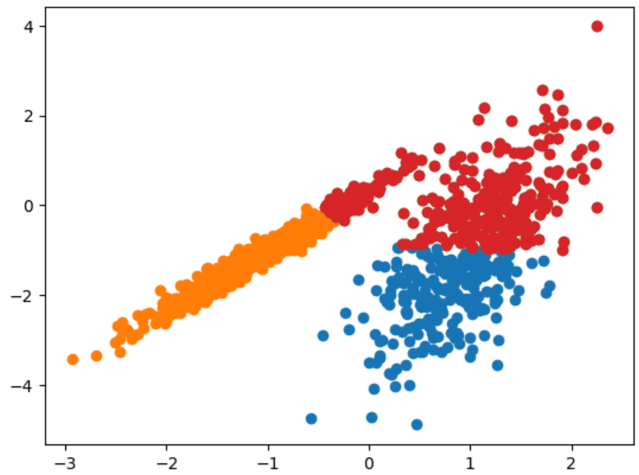
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. S
  warnings.warn(
      KMeans
KMeans(n_clusters=3)
```

```
# assign a cluster to each example
yhat = model.predict(X)
```

```
# retrieve unique clusters
clusters = unique(yhat)
```

```
from matplotlib import pyplot
# create scatter plot for samples from each cluster for cluster in clusters:
# get row indexes for samples with this cluster
# Create scatter plot for samples from each cluster
for cluster in clusters:
  # Get row indexes for samples with this cluster
  row_ix = where(yhat == cluster)[0]
  # Create scatter of these samples
  pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
  # create scatter of these samples
pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```

OUTPUT:



RESULT:

Thus the Python program for the Implementation of K-MEANS clustering was executed successfully.

EX.NO:05(e)	IMPLEMENTATION OF MINI BATCH K-MEANS CLUSTERING
DATE:	

AIM:

To write a Python program for the Implementation of MINI BATCH K-MEANS clustering.

ALGORITHM:

STEP 1: First, it imports the required libraries: numpy, scikit-learn, and matplotlib.

STEP 2: Then, it generates a synthetic dataset using the make_classification() function from scikit-learn.

STEP 3: Next, it creates an instance of the MiniBatchKMeans model with the specified hyperparameter: n_clusters=2.

STEP 4: It fits the model on the dataset using the fit() method.

STEP 5: It assigns a cluster label to each sample using the predict() method.

STEP 6: It retrieves the unique cluster labels using the unique() function from numpy.

STEP 7: It creates a scatter plot for each cluster, where each sample is colored according to its predicted cluster label using the scatter() function from matplotlib.

CODING:

```
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import MiniBatchKMeans
from matplotlib import pyplot
```

```
# define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2,
n_redundant=0,
        n_clusters_per_class=1, random_state=4)
```

```
# define the model
model = MiniBatchKMeans(n_clusters=2) # fit the model
model.fit(X)
```

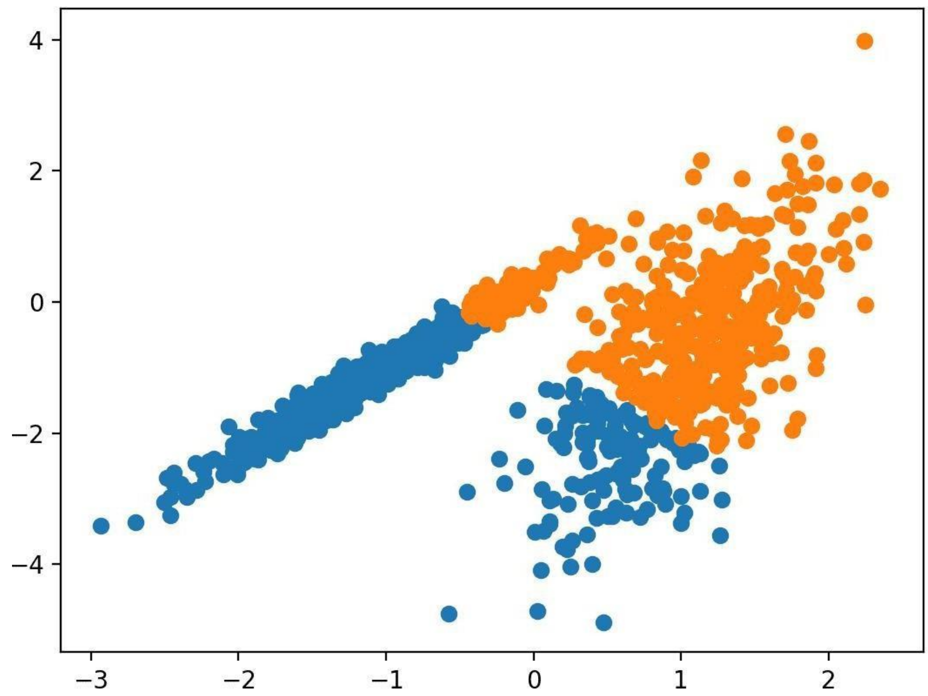
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 3 to 'auto' in 1.4. Set
  warnings.warn(
▾      MiniBatchKMeans
MiniBatchKMeans(n_clusters=2)
```

```
# assign a cluster to each example
yhat = model.predict(X)
```

```
# retrieve unique clusters
clusters = unique(yhat)
```

```
from matplotlib import pyplot
# create scatter plot for samples from each cluster for cluster in clusters:
# get row indexes for samples with this cluster
for cluster in clusters:
   # Get row indexes for samples with this cluster
   row_ix = where(yhat == cluster)
   # Create scatter of these samples
   pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# Show the plot
pyplot.show()
```

OUTPUT:



RESULT:

Thus the Python program for the Implementation of MINI BATCH K-MEANS clustering was executed successfully.

EX.NO: 05(f)	IMPLEMENTATION OF HIERARCHICAL CLUSTERING
DATE:	

AIM:

To write a Python program for the Implementation of Hierarchical clustering.

ALGORITHM:

STEP 1: First, it imports the required libraries: numpy, scikit-learn, and matplotlib.

STEP 2: Then, it generates a synthetic dataset using the linkage () function from scikit-learn.

STEP 3: Next, it creates an instance of the Hierarchical model with the specified Hyper parameter.

STEP 4: It fits the model on the dataset using the dendrogram() method.

STEP 5: It assigns a cluster label to each sample using the linkage method.

STEP 6: It retrieves the unique cluster labels using the function from numpy.

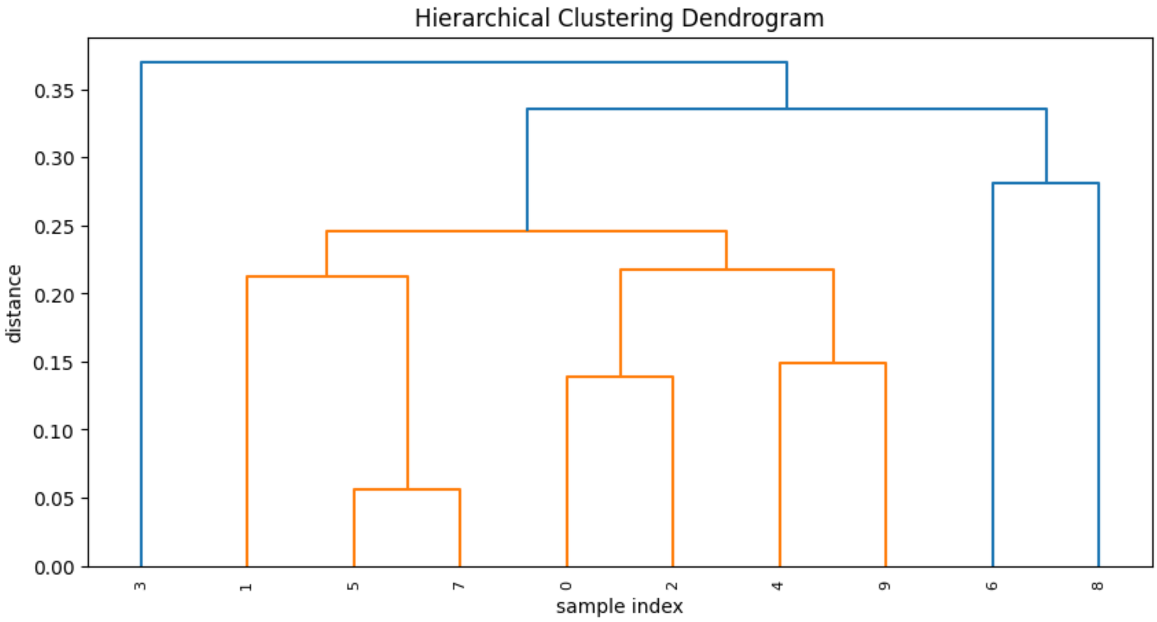
STEP 7: It creates a Dendrogram for each cluster, where each sample is colored according to its predicted cluster label using the dendrogram() function from matplotlib.

CODING:

```
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt

# Generating random data for demonstration
np.random.seed(123)
X = np.random.random((10, 2))  # 10 samples, 2 features

# Agglomerative clustering
Z = linkage(X, 'single')  # You can change the linkage method here

# Plotting the dendrogram
plt.figure(figsize=(10, 5))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('sample index')
plt.ylabel('distance')
dendrogram(
    Z,
```

```
    leaf_rotation=90.,  # rotates the x axis labels
    leaf_font_size=8.,  # font size for the x axis labels
)
plt.show()
```

OUTPUT:



RESULT:

Thus the Python program for the Implementation of Hierarchical Clustering was executed successfully.

EX.NO: 06	**STATISTICAL DESCRIPTION OF DATA**
DATE:	

AIM:

To study about the Statistical descriptions of data on the IRIS dataset.

PROCEDURE:

- Exploratory Data Analysis (EDA) is a technique to analyze data using some visual Techniques.
- With this technique, we can get detailed information about the statistical summary of the data.
- We will also be able to deal with the duplicates values, outliers, and also see some trends orpatterns present in the dataset.

IRIS DATASET:

- If you are from a data science background you all must be familiar with the Iris Dataset.
- If you are not then don't worry we will discuss this here.
- Iris Dataset is considered as the Hello World for data science.
- It contains five columns namely – Petal Length, Petal Width, Sepal Length, Sepal Width, andSpecies Type.
- Iris is a flowering plant, the researchers have measured various features of the different irisflowers and recorded them digitally.

EXAMPLE:

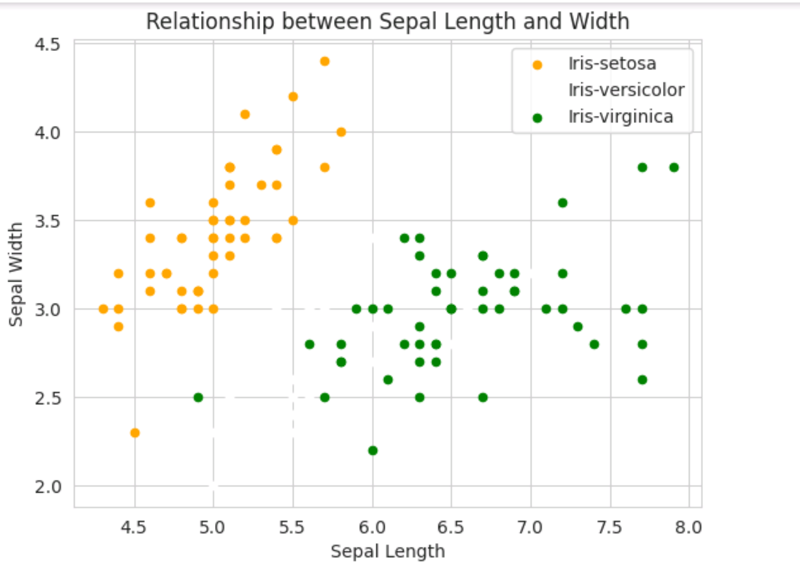
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import io
import ipywidgets as widgets

# Function to handle file upload
def handle_file_upload(change):
    uploaded_file = list(upload_button.value.values())[0]
```

```
    try:
        content = uploaded_file['content']
        # Read the uploaded CSV file
        global iris_df
        iris_df = pd.read_csv(io.StringIO(content.decode('utf-8')))
        print("File uploaded successfully.")
    except Exception as e:
        print(f"Error: {e}")

# Create a file upload widget
upload_button = widgets.FileUpload(accept='.csv')

# Display the upload button
display(upload_button)

# Register the file upload handler
upload_button.observe(handle_file_upload, names='value')

# Scatter plot function
def scatter_plot(data):
    sns.set_style("whitegrid")
    plt.figure(figsize=(10,6))
    ax = None
    for species, color in zip(data['Species'].unique(), ['orange', 'white', 'green']):
        ax = data[data.Species == species].plot.scatter(x='SepalLengthCm',
y='SepalWidthCm', color=color, label=species, ax=ax)
    ax.set_xlabel("Sepal Length")
    ax.set_ylabel("Sepal Width")
    ax.set_title("Relationship between Sepal Length and Width")
    plt.show()

# Main code
if 'iris_df' not in globals():
    print("Please upload a dataset file.")
else:
    # Descriptive Statistics
    print(iris_df.describe())

    # The different categories of Species
```

```
print(iris_df['Species'].unique())

# Number of Records per species
iris = iris_df.groupby('Species', as_index=False)["Id"].count()
print(iris)

# Generate scatter plot
scatter_plot(iris_df)
```

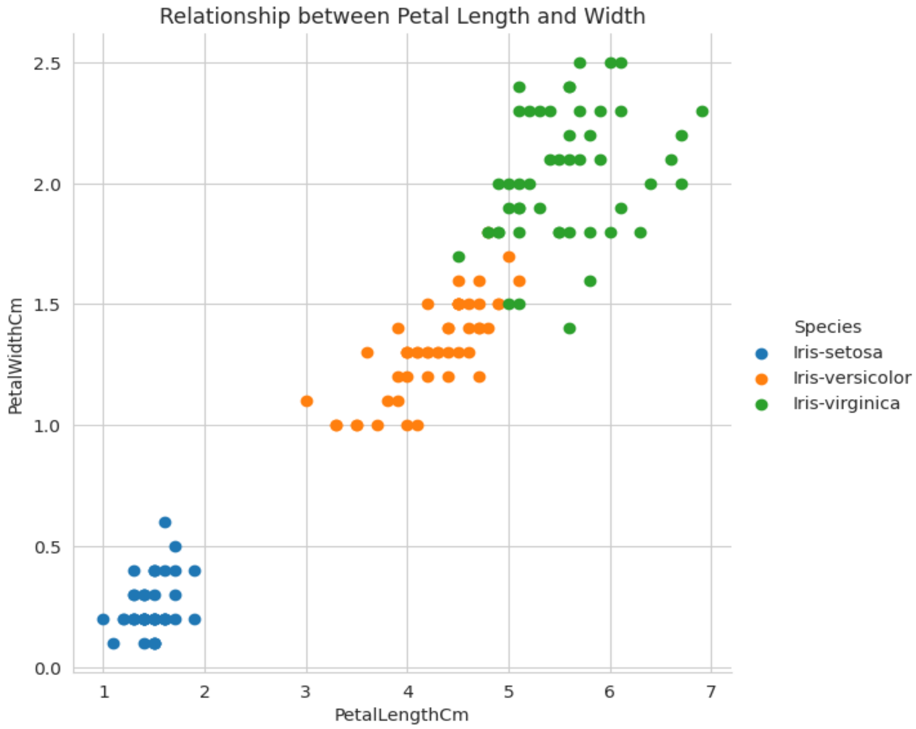
Upload (0)

```
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
           Species  Id
0      Iris-setosa  50
1  Iris-versicolor  50
2   Iris-virginica  50
<Figure size 1000x600 with 0 Axes>
```




```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style("whitegrid")
# Create a FacetGrid
g = sns.FacetGrid(iris_df, hue="Species",height=6)

# Map scatter plot to the FacetGrid
g.map(plt.scatter, "PetalLengthCm", "PetalWidthCm")
```

```
# Add legend
g.add_legend()

# Set title
plt.title("Relationship between Petal Length and Width")

# Show the plot
plt.show()
```




```
from sklearn.datasets import load_iris
import pandas as pd

# Load Iris dataset
iris = load_iris()

# Create a DataFrame from the dataset
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['species'] = iris.target

# Display the first few rows of the DataFrame
print(iris_df.head())
print(iris_df.columns)
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   species
0        0
1        0
2        0
3        0
4        0
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'species'],
      dtype='object')
```

print(iris_df.shape)

(150, 5)

iris_df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    int64
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

iris_df.describe()

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

iris_df.isnull().sum()

```
sepal length (cm)   0
sepal width (cm)    0
petal length (cm)   0
petal width (cm)    0
species             0
dtype: int64
```

```
data = iris_df.drop_duplicates(subset="species")
print(data)
```

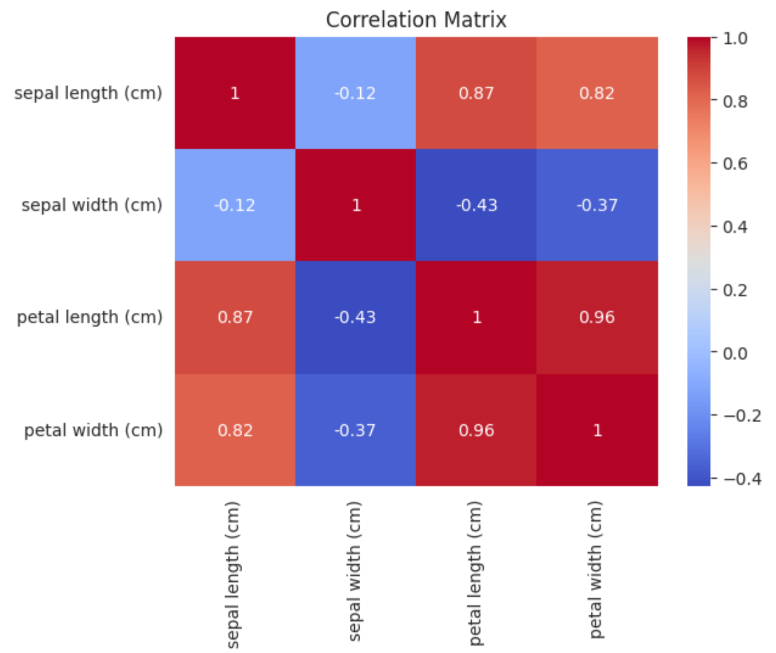
```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                  5.1               3.5                1.4               0.2
50                 7.0               3.2                4.7               1.4
100                6.3               3.3                6.0               2.5

     species
0          0
50         1
100        2
```

```
iris_df['species'].value_counts()
```

```
species
0    50
1    50
2    50
Name: count, dtype: int64
```

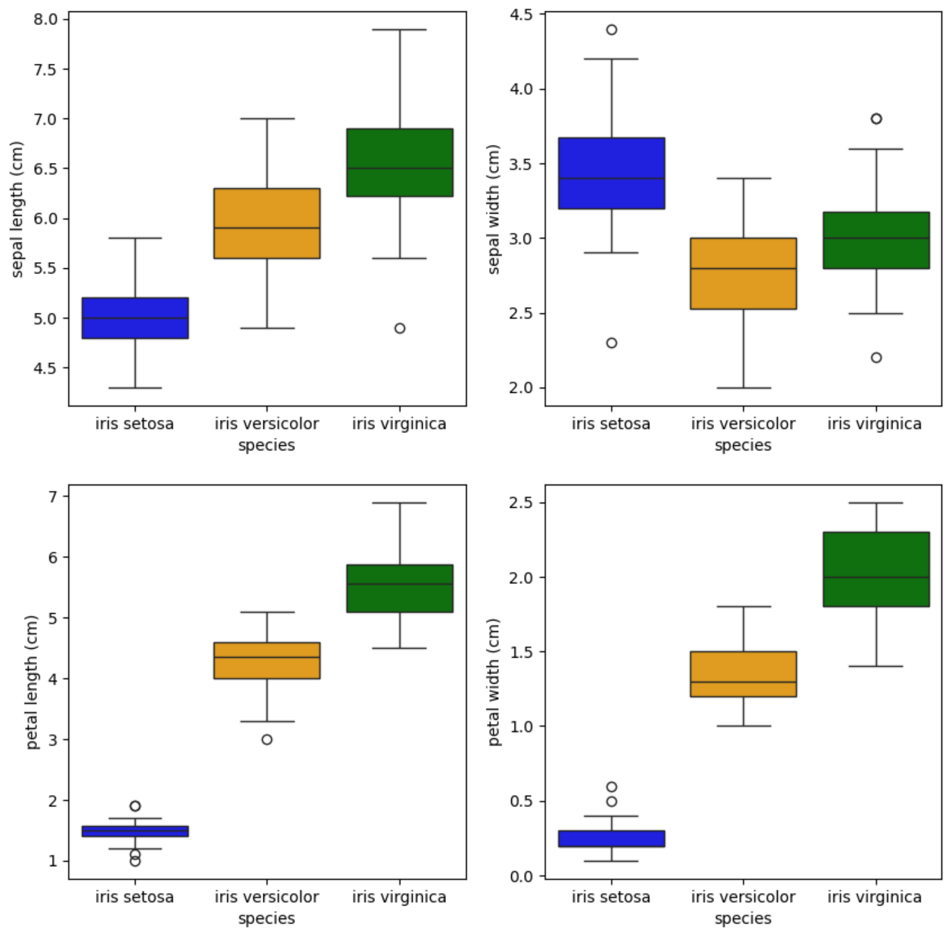
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Drop 'Id' column before calculating correlation
print(iris_df.columns)
iris_df.drop('species', axis=1, inplace=True)

# Generate heatmap
sns.heatmap(iris_df.corr(method='pearson'), annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')
plt.show()
```



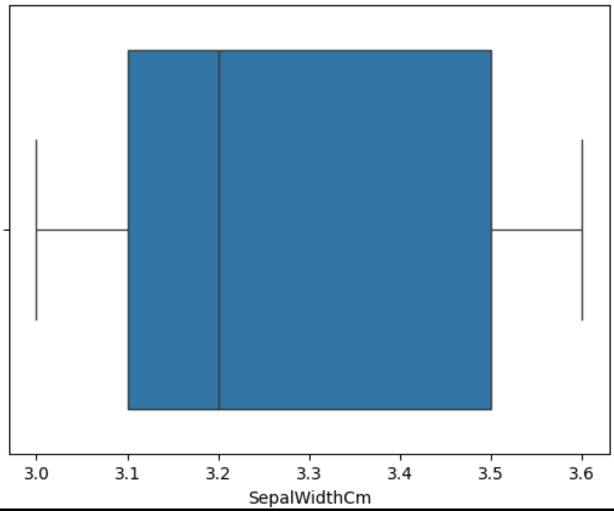

```
# Importing necessary packages
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets  import load_iris

# Load the Iris dataset
iris = load_iris()
```

```
# Create a DataFrame from the dataset
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['species'] = iris.target

def graph(y, df):
    sns.boxplot(x="species", y=y, data=df, palette=["blue", "orange", "green"])

# Define the species names
species_names = ['iris setosa', 'iris versicolor', 'iris virginica']

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

# Assuming you want to plot the features against the species column in the iris_df DataFrame
plt.subplot(221)
graph('sepal length (cm)', iris_df)
plt.xticks(ticks=[0, 1, 2], labels=species_names)

plt.subplot(222)
graph('sepal width (cm)', iris_df)
plt.xticks(ticks=[0, 1, 2], labels=species_names)

plt.subplot(223)
graph('petal length (cm)', iris_df)
plt.xticks(ticks=[0, 1, 2], labels=species_names)

plt.subplot(224)
graph('petal width (cm)', iris_df)
plt.xticks(ticks=[0, 1, 2], labels=species_names)

plt.show()
```




```
# Import necessary packages
```

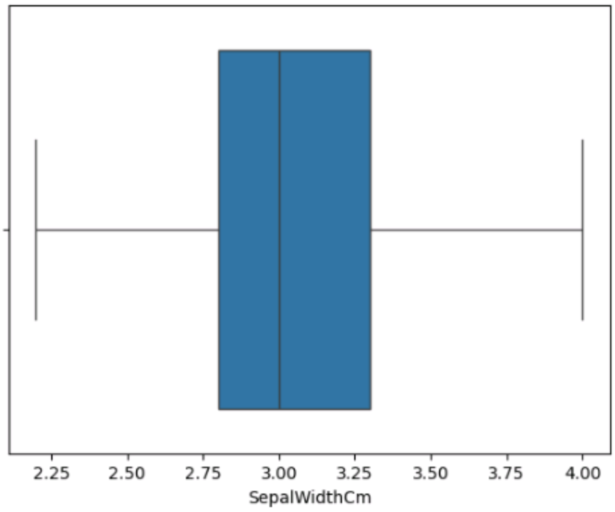
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset from a URL or your Google Drive
# For example, if the CSV file is hosted on the web
# df = pd.read_csv("https://example.com/yourfile.csv")

# Or if the CSV file is in your Google Drive
# from google.colab import drive
# drive.mount('/content/drive')
# file_path = "/content/drive/MyDrive/yourfile.csv"
# df = pd.read_csv(file_path)

# For demonstration, let's create a DataFrame with sample data
# You should replace this with your actual data loading code
data = {
   'SepalWidthCm': [3.5, 3.0, 3.2, 3.1, 3.6],
}
df = pd.DataFrame(data)

# Create the boxplot
sns.boxplot(x='SepalWidthCm', data=df)
plt.show()
```




```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import files
import io

# Ask the user to upload the CSV file
uploaded = files.upload()

# Load the dataset
df = pd.read_csv(io.StringIO(uploaded['Iris.csv'].decode('utf-8')))

# IQR calculation
Q1 = np.percentile(df['SepalWidthCm'], 25, interpolation='midpoint')
Q3 = np.percentile(df['SepalWidthCm'], 75, interpolation='midpoint')
IQR = Q3 - Q1
print("Old Shape:", df.shape)
```

```
# Upper bound
upper = np.where(df['SepalWidthCm'] >= (Q3 + 1.5 * IQR))
# Lower bound
lower = np.where(df['SepalWidthCm'] <= (Q1 - 1.5 * IQR))

# Removing the Outliers
df.drop(upper[0],inplace=True)
df.drop(lower[0], inplace=True)
print("New Shape:", df.shape)

# Create the boxplot
sns.boxplot(x='SepalWidthCm', data=df)
plt.show()
```

```
Choose Files  No file chosen        Upload widget is only available when the cell has been exec
Saving Iris.csv to Iris.csv
Old Shape: (150, 6)
New Shape: (146, 6)
```



RESULT:

Thus the Statistical Descriptions of data on the IRIS data set has been done successfully.

EX.NO: 07	IMPLEMENTATION OF ASSOCIATION RULES
DATE:	

AIM:

To Finding Association Rules for buying data.

DESCRIPTION:

- In data mining, **association rule learning** is a popular and well researched method for discovering interesting relations between variables in large databases.
- It can be described as analyzing and presenting strong rules discovered in databases using different measures of interestingness.
- In market basket analysis association rules are used and they are also employed in many application areas including Web usage mining, intrusion detection and bioinformatics.

➢ **CREATION OF BUYING TABLE:**

PROCEDURES:

1. Open Start Programs Accessories Notepad.
2. Type the following training data set with the help of Notepad for Buying Table.

```
@relation buying
@attribute age {L20,20-40,G40}
@attribute income {high,medium,low}
@attribute stud {yes,no}
@attribute creditrate {fair,excellent}
@attribute buyscomp {yes,no}
@data
L20,high,no,fair,yes
20-40,low,yes,fair,yes
G40,medium,yes,fair,yes
L20,low,no,fair,no
G40,high,no,excellent,yes
L20,low,yes,fair,yes
20-40,high,yes,excellent,no
G40,low,no,fair,yes
L20,high,yes,excellent,yes
G40,high,no,fair,yes
```

L20,low,yes,excellent,no

G40,high,yes,excellent,no

20-40,medium,yes,excellent,yes

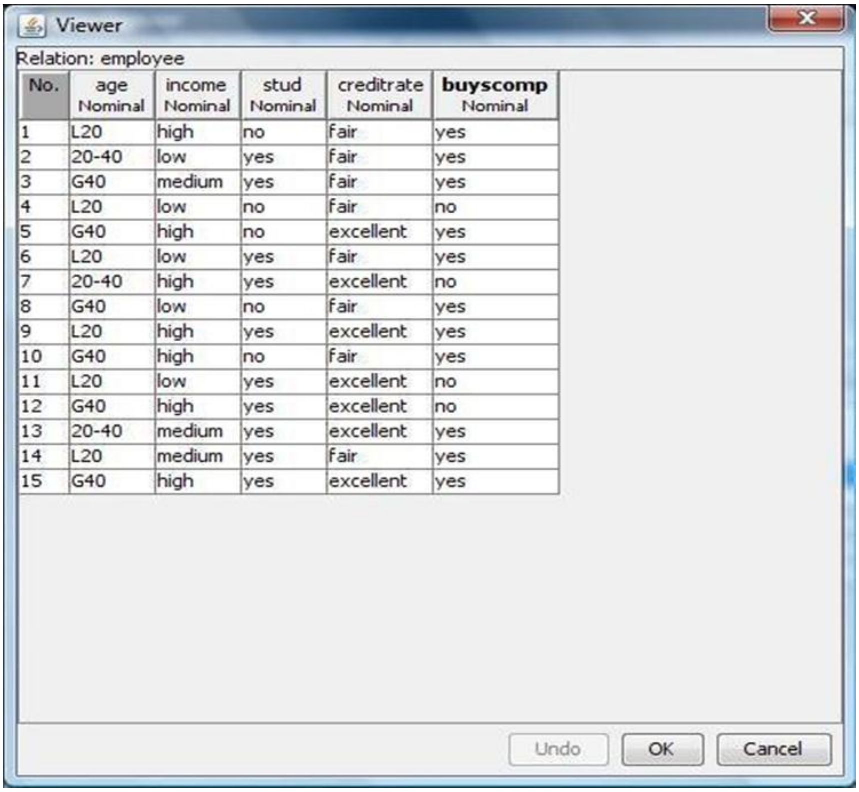
L20,medium,yes,fair,yes

G40,high,yes,excellent,yes

3. After that the file is saved with **buying.arff** file format.
4. Minimize the arff file and then open Start Programs weka-3-4.
5. Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6. In that dialog box there are four modes, click on **explorer**.
7. Explorer shows many options. In that click on **'open file'** and select the arff file
8. Click on **edit button** which shows buying table on weka.

OUTPUT:

TRAINING DATA SET BUYING TABLE:

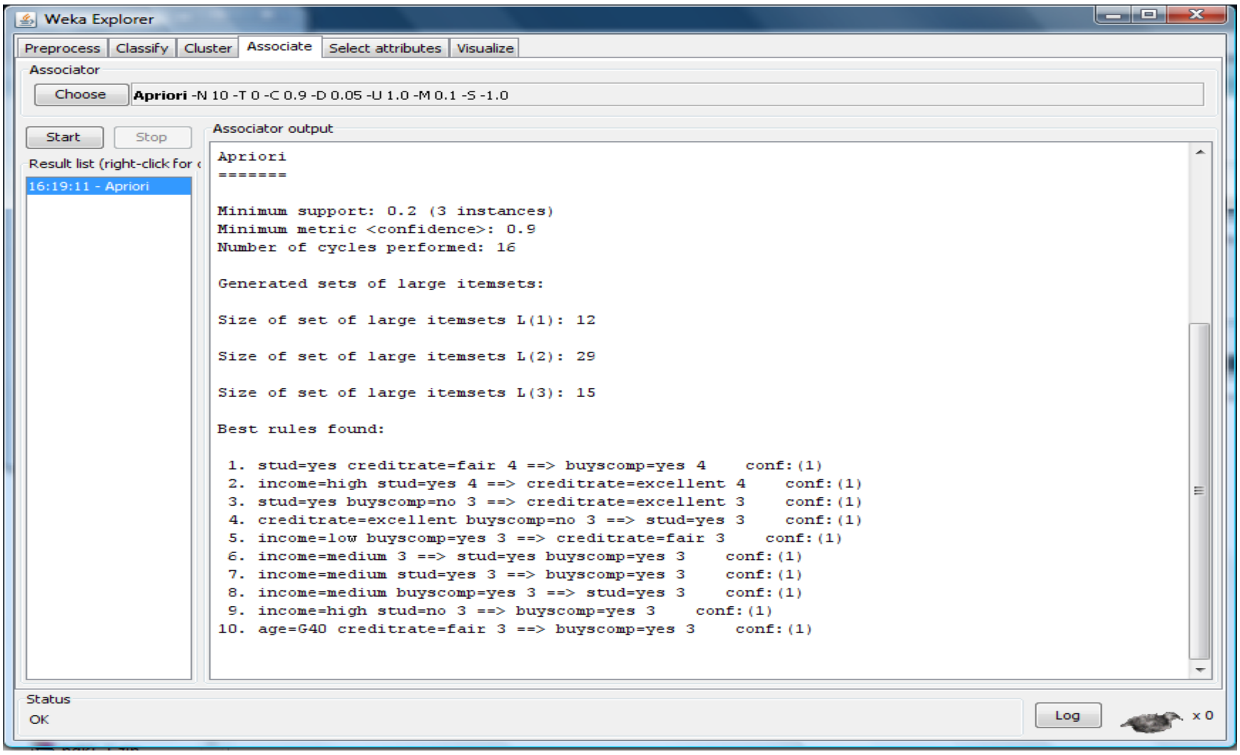


No.	age Nominal	income Nominal	stud Nominal	creditrate Nominal	buyscomp Nominal
1	L20	high	no	fair	yes
2	20-40	low	yes	fair	yes
3	G40	medium	yes	fair	yes
4	L20	low	no	fair	no
5	G40	high	no	excellent	yes
6	L20	low	yes	fair	yes
7	20-40	high	yes	excellent	no
8	G40	low	no	fair	yes
9	L20	high	yes	excellent	yes
10	G40	high	no	fair	yes
11	L20	low	yes	excellent	no
12	G40	high	yes	excellent	no
13	20-40	medium	yes	excellent	yes
14	L20	medium	yes	fair	yes
15	G40	high	yes	excellent	yes

PROCEDURE:

1. Open Start Programs Weka-3-4.
2. Open **explorer**.
3. Click on **open file** and select **buying.arff**
4. Select **Associate option** on the top of the Menu bar.

5. Select **Choose button** and then click on **Apriori Algorithm**.
6. Click on **Start button** and output will be displayed on the **right side** of the window.



- **CREATION OF BANKING TABLE:**

PROCEDURE:

1. Open Start Programs Accessories Notepad.
2. Type the following training data set with the help of Notepad for BankingTable.

@relation bank

@attribute cust {male,female}

@attribute accno {0101,0102,0103,0104,0105,0106,0107,0108,0109,0110,0111,0112, 0113, 0114,0115}

@attribute bankname {sbi,hdfc,sbh,ab,rbi}

@attribute location {hyd,jmd,antp,pdtr,kdp}

@attribute deposit {yes,no}

@data

male,0104,ab,pdtr,yes

female,0105,sbi,jmd,no

male,0106,ab,hyd,yes

female,0107,rbi,jmd,yes

female,0108,hdfc,kdp,no

male,0109,sbh,kdp,yes

male,0110,ab,jmd,no

female,0111,rbi,kdp,yes

male,0112,sbi,jmd,yes

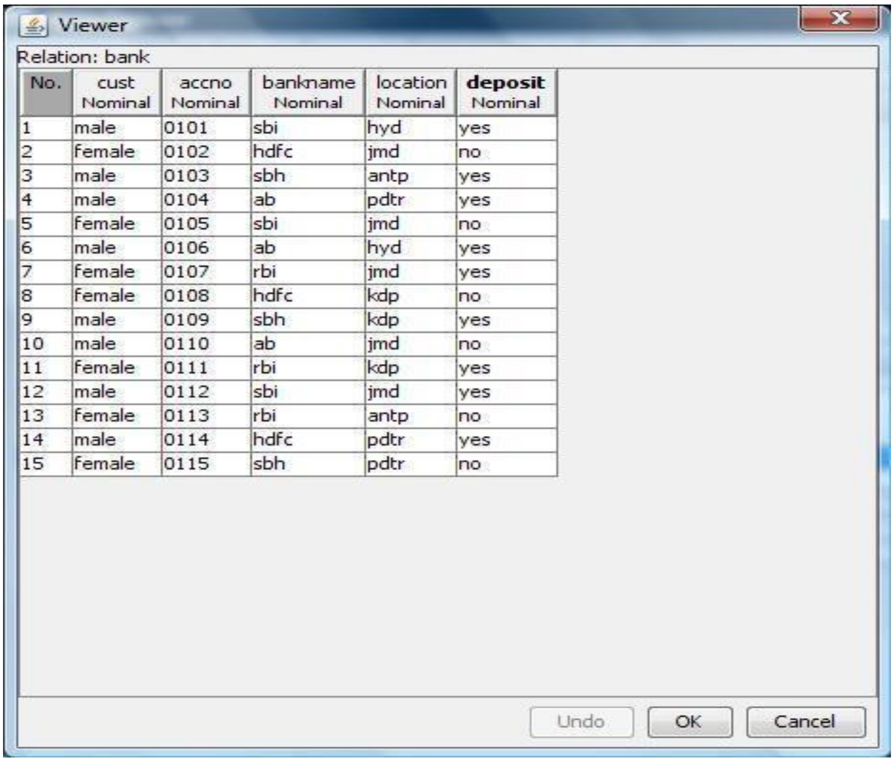
female,0113,rbi,antp,no

male,0114,hdfc,pdtr,yes

female,0115,sbh,pdtr,no

3. After that the file is saved with **bank.arff** file format.
4. Minimize the arff file and then open Start Programs weka-3-4.
5. Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6. In that dialog box there are four modes, click on **explorer**.
7. Explorer shows many options. In that click on **'open file'** and select the arff file
8. Click on **edit button** which shows banking table on weka.

TRAINING DATA SET BANKING TABLE:


Viewer

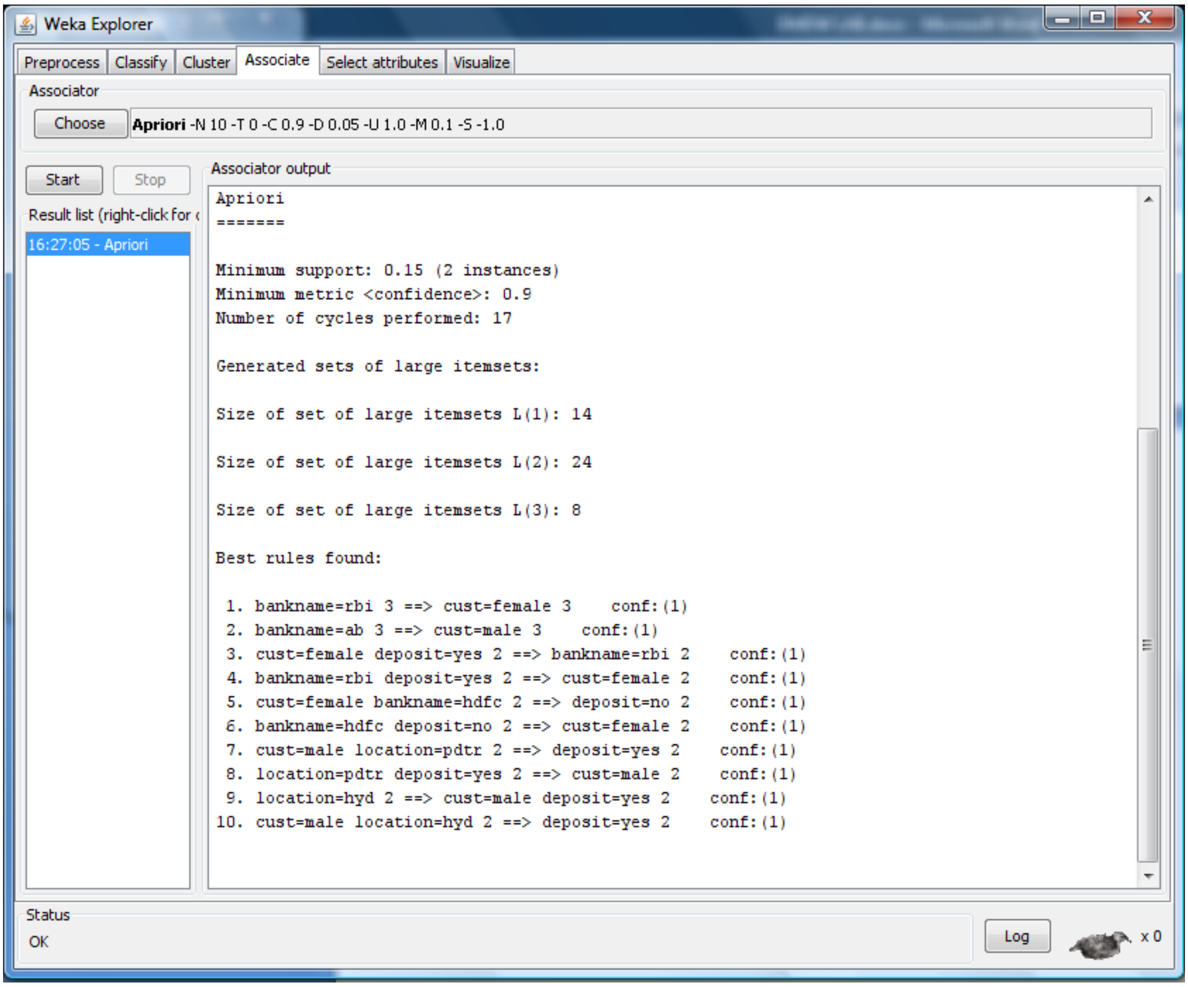
Relation: bank

No.	cust Nominal	accno Nominal	bankname Nominal	location Nominal	deposit Nominal
1	male	0101	sbi	hyd	yes
2	female	0102	hdfc	jmd	no
3	male	0103	sbh	antp	yes
4	male	0104	ab	pdtr	yes
5	female	0105	sbi	jmd	no
6	male	0106	ab	hyd	yes
7	female	0107	rbi	jmd	yes
8	female	0108	hdfc	kdp	no
9	male	0109	sbh	kdp	yes
10	male	0110	ab	jmd	no
11	female	0111	rbi	kdp	yes
12	male	0112	sbi	jmd	yes
13	female	0113	rbi	antp	no
14	male	0114	hdfc	pdtr	yes
15	female	0115	sbh	pdtr	no

PROCEDURE FOR ASSOCIATION RULES:

a. Open Start Programs Weka-3-4.
b. Open **explorer**.
c. Click on **open file** and select **bank.arff**
d. Select **Associate option** on the top of the Menu bar.
e. Select **Choose button** and then click on **Apriori Algorithm**.
f. Click on **Start button** and output will be displayed on the **right side** of the window.

OUTPUT:



- **CREATION OF EMPLOYEE TABLE :**

PROCEDURE:

1. Open Start Programs Accessories Notepad.
2. Type the following training data set with the help of Notepad for EmployeeTable.

@relation employee-1

@attribute age {youth, middle, senior}

@attribute income {high, medium, low}

@attribute class {A, B, C}

@data

youth, high, A

youth,medium,B

youth, low,

C middle, low,

C middle, medium,

C middle, high,

A senior, low,

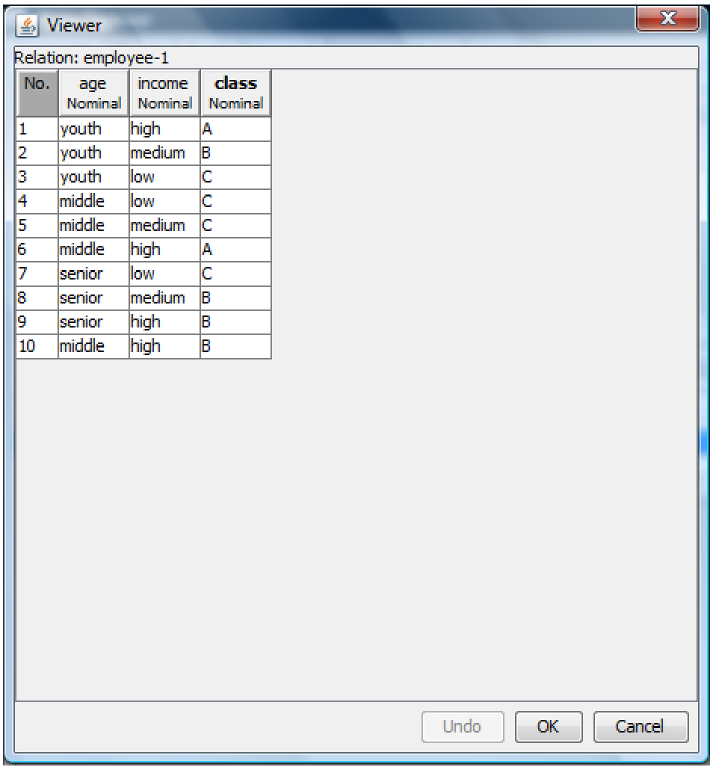
C senior, medium,

B senior, high,

B middle, high, B

3. After that the file is saved with **employee-1.arff** file format.
4. Minimize the arff file and then open Start Programs weka-3-4.
5. Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6. In that dialog box there are four modes, click on **explorer**.
7. Explorer shows many options. In that click on **'open file'** and select the arff file
8. Click on **edit button** which shows employee table on weka.

TRAINING DATA SET EMPLOYEE TABLE:

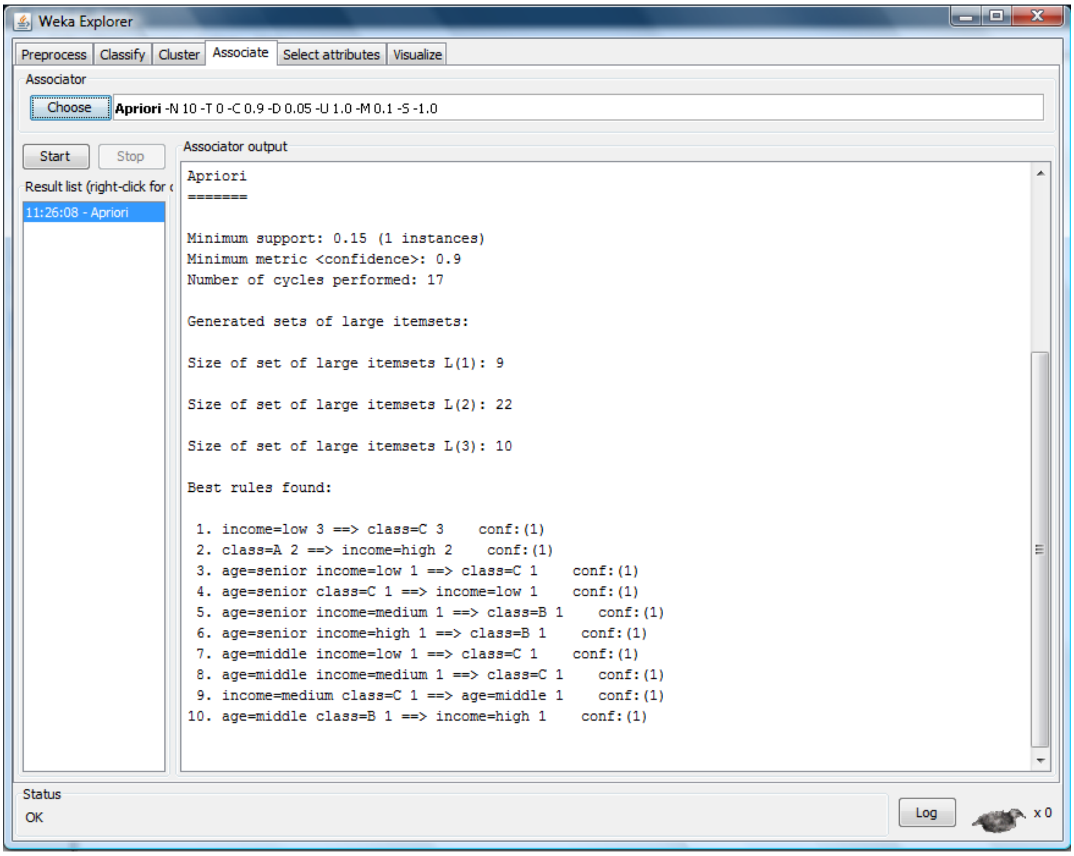


No.	age Nominal	income Nominal	class Nominal
1	youth	high	A
2	youth	medium	B
3	youth	low	C
4	middle	low	C
5	middle	medium	C
6	middle	high	A
7	senior	low	C
8	senior	medium	B
9	senior	high	B
10	middle	high	B

PROCEDURE FOR ASSOCIATION RULES:

- Open Start Programs Weka-3-4.
- Open **explorer**.
- Click on **open file** and select **employee-1.arff**
- Select **Associate option** on the top of the Menu bar.
- Select **Choose button** and then click on **Apriori Algorithm**.
- Click on **Start button** and output will be displayed on the **right side** of the window.

OUTPUT:



RESULT:

Thus the Association Rules for buying data has been successfully executed.

EX.NO: 8(a)	**COBWEB ALGORITHM**
DATE:	

AIM:

To write a procedure for Clustering Buying data using Cobweb Algorithm.

DESCRIPTION:

- Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters.
- Clustering is a main task of explorative data mining, and a common technique for statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

CREATION OF EMPLOYEE TABLE:

PROCEDURE:

1. Open Start Programs Accessories Notepad.
2. Type the following training data set with the help of Notepad for Employee-2 Table.

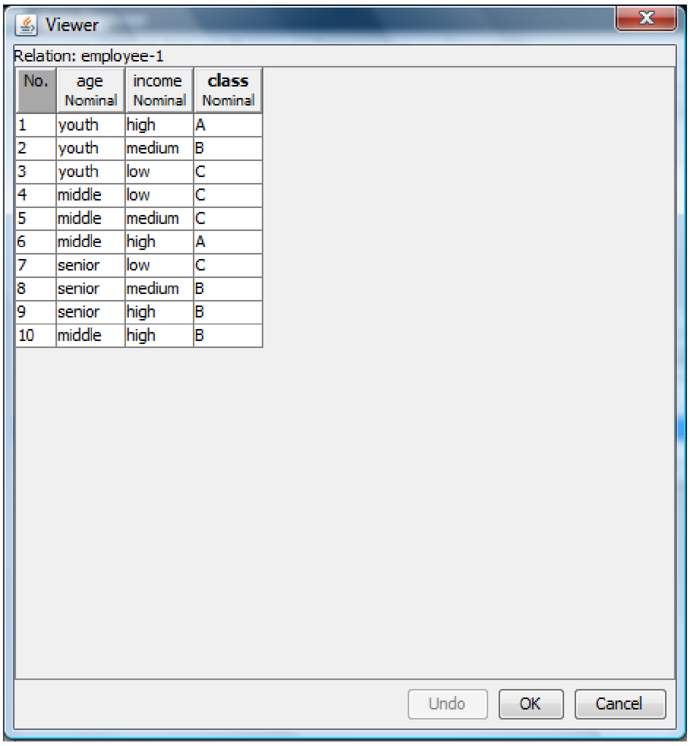
```
@relation buying
@attribute age {L20,20-40,G40}
@attribute income {high,medium,low}
@attribute stud {yes,no}
@attribute creditrate {fair,excellent}
@attribute buyscomp {yes,no}
@data
L20,high,no,fair,yes
20-40,low,yes,fair,yes
G40,medium,yes,fair,yes
L20,low,no,fair,no
G40,high,no,excellent,yes
L20,low,yes,fair,yes
20-40,high,yes,excellent,no
G40,low,no,fair,yes
L20,high,yes,excellent,yes
G40,high,no,fair,yes
L20,low,yes,excellent,no
G40,high,yes,excellent,no
20-40,medium,yes,excellent,yes
```

L20,medium,yes,fair,yes

G40,high,yes,excellent,yes

3. After that the file is saved with employee-1**.arff** file format.
4. Minimize the arff file and then open Start Programs weka-3-4.
5. Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6. In that dialog box there are four modes, click on **explorer**.
7. Explorer shows many options.
8. In that click on **'open file'** and select the arff file.
9. Click on **edit button** which shows buying table on weka.

TRAINING DATA SET EMPLOYEE TABLE:

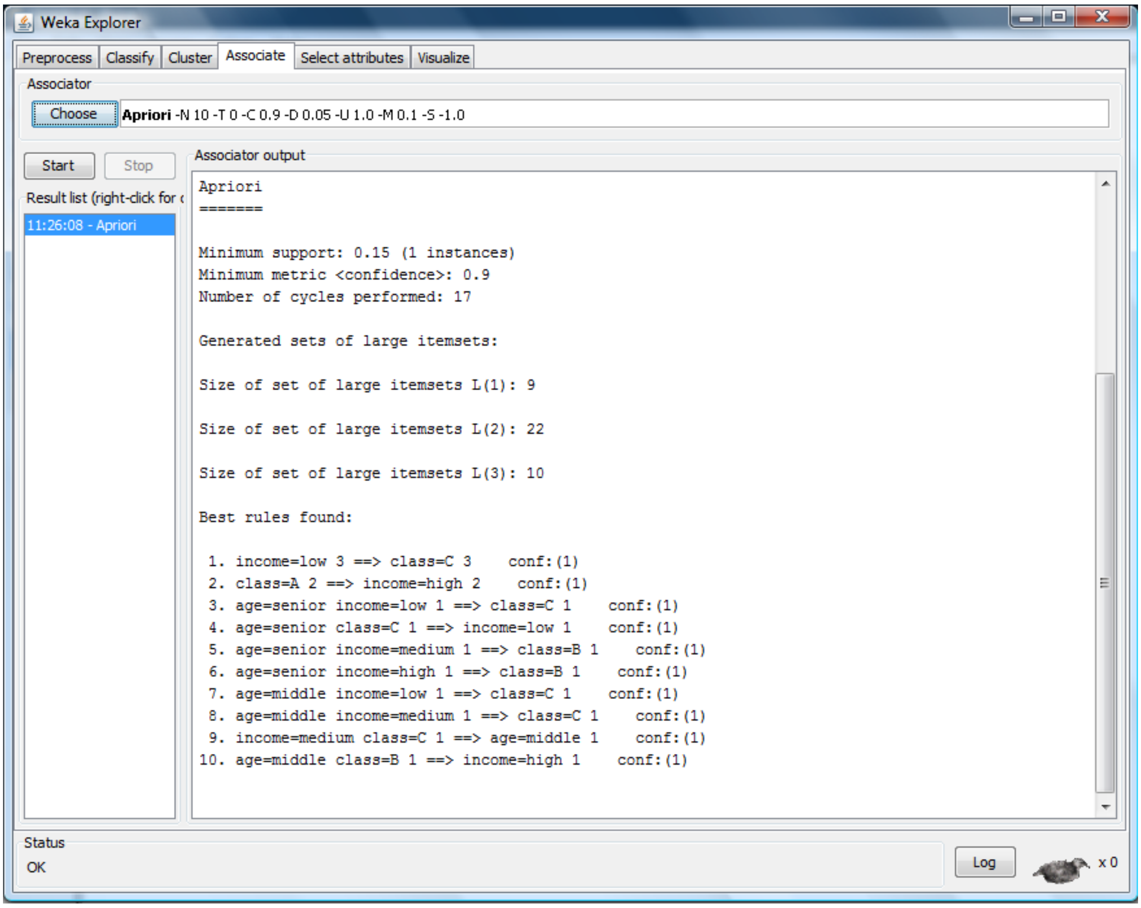


No.	age Nominal	income Nominal	class Nominal
1	youth	high	A
2	youth	medium	B
3	youth	low	C
4	middle	low	C
5	middle	medium	C
6	middle	high	A
7	senior	low	C
8	senior	medium	B
9	senior	high	B
10	middle	high	B

PROCEDURE:

1. Click **Start** -> **Programs** -> **Weka 3.4**
2. Click on **Explorer**.
3. Click on **open file** & then select **employee-1.arff** file.
4. Click on **Cluster menu**. In this there are different algorithms are there.
5. Click on **Choose button** and then select **cobweb** algorithm.
6. Click on **Start button** and then **output** will be displayed on the screen.

OUTPUT:



RESULT:

Thus a procedure for Clustering Buying data using Cobweb Algorithm has been successfully executed.

EX.NO:8(b)	EM ALGORITHM
DATE:	

AIM:

To write a procedure for Clustering Weather data using EM Algorithm.

DESCRIPTION:

- Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) toeach other than to those in other clusters.
- Clustering is a main task of explorative data mining, and a common technique for statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

CREATION OF WEATHER TABLE:

PROCEDURE:

1. Open Start Programs Accessories Notepad.
2. Type the following training data set with the help of Notepad for WeatherTable.

   @relation weather

   @attribute outlook {sunny, rainy, overcast}

   @attribute temperature numeric

   @attribute humidity numeric

   @attribute windy {TRUE, FALSE}

   @attribute play {yes, no}

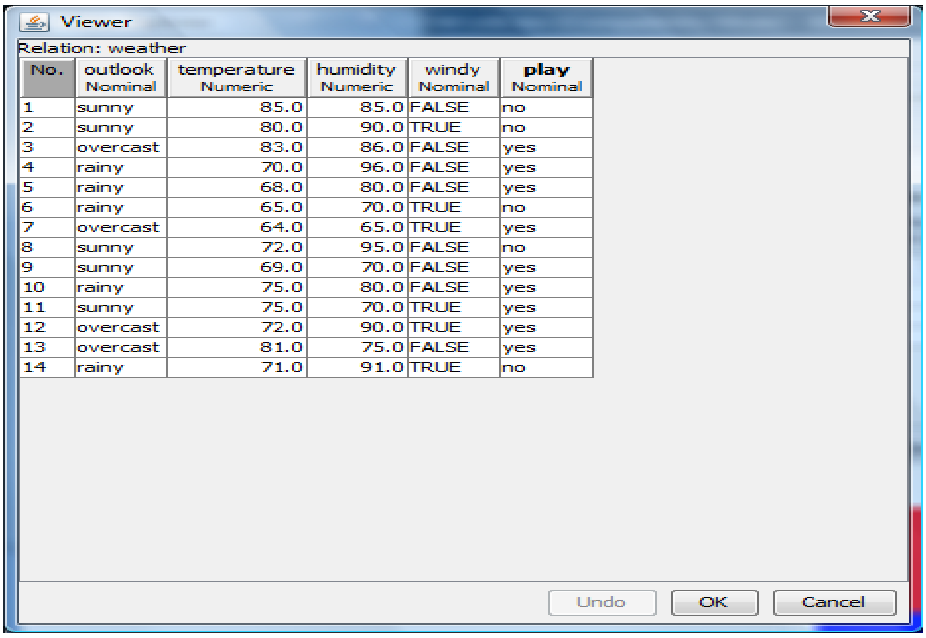
   @data

sunny,85,85,FALSE,no	sunny,72,95,FALSE,no
sunny,80,90,TRUE,no	sunny,69,70,FALSE,yes
overcast,83,86,FALSE,yes	rainy,75,80,FALSE,yes
rainy,70,96,FALSE,yes	sunny,75,70,TRUE,yes
rainy,68,80,FALSE,yes	overcast,72,90,TRUE,yes
rainy,65,70,TRUE,no	overcast,81,75,FALSE,yes
overcast,64,65,TRUE,yes	rainy,71,91,TRUE,no

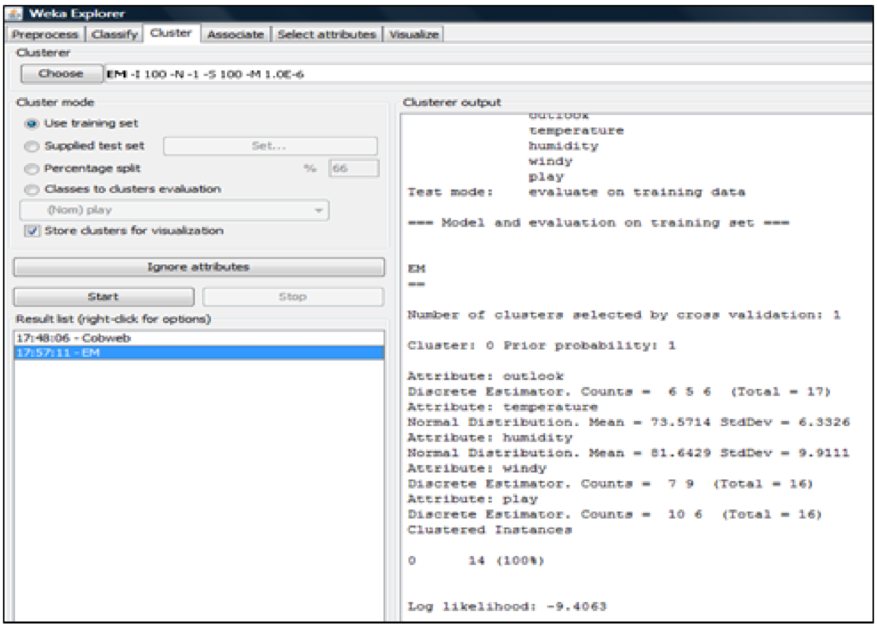
3. After that the file is saved with **weather.arff** file format.
4. Minimize the arff file and then open Start Programs weka-3-4.
5. Click on **weka-3-4**, then Weka dialog box is displayed on the screen.

6. In that dialog box there are four modes, click on **explorer**.
7. Explorer shows many options. In that click on **'open file'** and select the arff file.
8. Click on **edit button** which shows weather table on weka.

TRAINING DATA SETWEATHER TABLE:



No.	outlook Nominal	temperature Numeric	humidity Numeric	windy Nominal	play Nominal
1	sunny	85.0	85.0	FALSE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no



RESULT:

Thus the Clustering Weather data using EM Algorithm has been successfully executed.

EX.NO:8(c)	FARTHEST FIRST ALGORITHM
DATE:	

AIM:

To write a procedure for Banking data using Farthest First Algorithm.

DESCRIPTION:

- Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) toeach other than to those in other clusters.
- Clustering is a main task of explorative data mining, and a common technique for statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

CREATION OF BANKING TABLE:

PROCEDURE:

1. Open Start Programs Accessories Notepad.
2. Type the following training data set with the help of Notepad for Banking Table.

@relation bank

@attribute cust {male,female}

@attribute accno {0101,0102,0103,0104,0105,0106,0107,0108,0109,0110,0111,0112,0113, 0114, 0115}

@attribute bankname {sbi,hdfc,sbh,ab,rbi}

@attribute location {hyd,jmd,antp,pdtr,kdp}

@attribute deposit {yes,no}

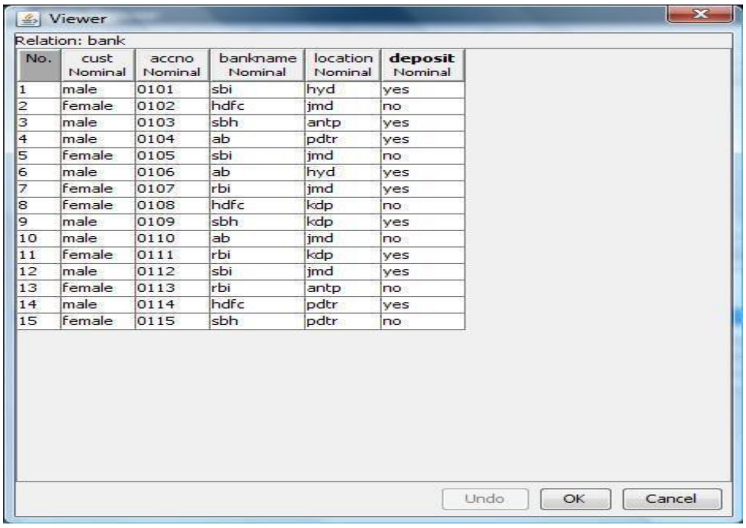
@data

male,0101,sbi,hyd,yes	female,0108,hdfc,kdp,no
female,0102,hdfc,jmd,no	male,0109,sbh,kdp,yes
male,0103,sbh,antp,yes	male,0110,ab,jmd,no
male,0104,ab,pdtr,yes	female,0111,rbi,kdp,yes
female,0105,sbi,jmd,no	male,0112,sbi,jmd,yes
male,0106,ab,hyd,yes	female,0113,rbi,antp,no
female,0107,rbi,jmd,yes	male,0114,hdfc,pdtr,yes
	female,0115,sbh,pdtr,no

3. After that the file is saved with **bank.arff** file format.
4. Minimize the arff file and then open Start Programs weka-3-4.
5. Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6. In that dialog box there are four modes, click on **explorer**.

7. Explorer shows many options. In that click on **'open file'** and select the arff file
8. Click on **edit button** which shows banking table on weka.

TRAINING DATA SET BANKING TABLE:



Viewer

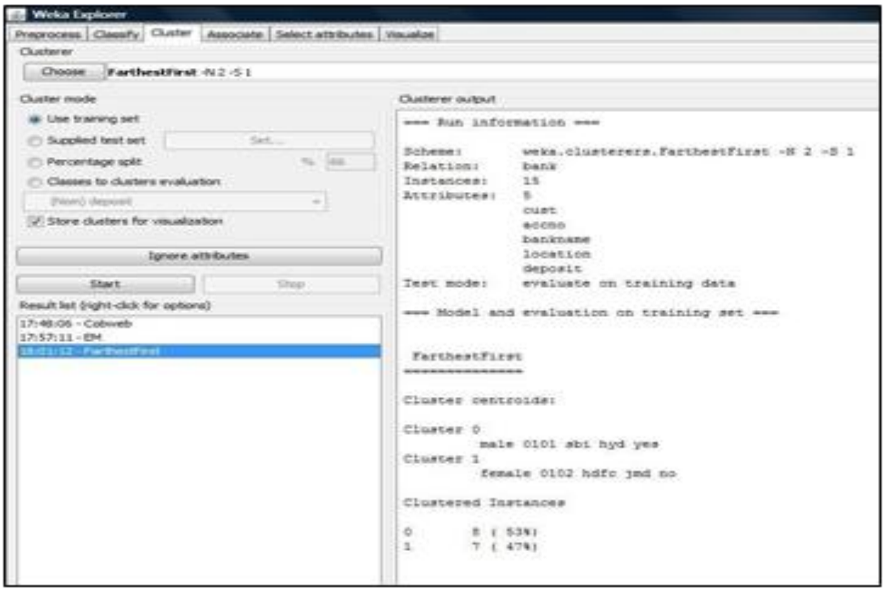
Relation: bank

No.	cust Nominal	accno Nominal	bankname Nominal	location Nominal	deposit Nominal
1	male	0101	sbi	hyd	yes
2	female	0102	hdfc	jmd	no
3	male	0103	sbh	antp	yes
4	male	0104	ab	pdtr	yes
5	female	0105	sbi	jmd	no
6	male	0106	ab	hyd	yes
7	female	0107	rbi	jmd	yes
8	female	0108	hdfc	kdp	no
9	male	0109	sbh	kdp	yes
10	male	0110	ab	jmd	no
11	female	0111	rbi	kdp	yes
12	male	0112	sbi	jmd	yes
13	female	0113	rbi	antp	no
14	male	0114	hdfc	pdtr	yes
15	female	0115	sbh	pdtr	no

Undo OK Cancel

PROCEDURE:

1. Click **Start** -> **Programs** -> **Weka 3.4**
2. Click on **Explorer**.
3. Click on **open file** & then select **Banking.arff** file.
4. Click on **Cluster menu**. In this there are different algorithms are there.
5. Click on **Choose button** and then select **FarthestFirst** algorithm.
6. Click on **Start button** and then **output** will be displayed on the screen.



RESULT:

Thus the Banking data using Farthest First Algorithm has been successfully executed.

EX.NO:8(d)	DENSITY BASED CLUSTER ALGORITHM
DATE:	

AIM:

To write a procedure for Employee data using Density Based Cluster Algorithm.

DESCRIPTION:

- Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters.
- Clustering is a main task of explorative data mining, and a common technique for statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

CREATION OF EMPLOYEE TABLE:

PROCEDURE:

1. Open Start Programs Accessories Notepad.
2. Type the following training data set with the help of Notepad for Employee Table.

@relation employee

@attribute eid numeric

@attribute ename {raj,ramu,anil,sunil,rajiv,sunitha,kavitha,suresh,ravi, ramana, ram,kavya,navya}

@attribute salary numeric

@attribute exp numeric

@attribute address {pdtr,kdp,nlr,gtr}

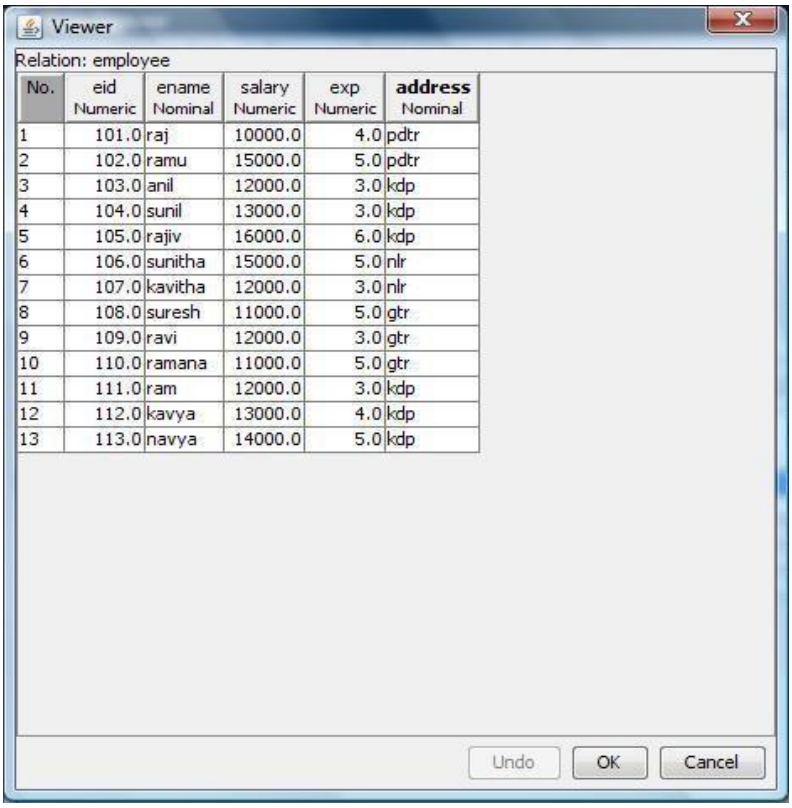
@data

101,raj,10000,4,pdtr	108,suresh,11000,5,gtr
102,ramu,15000,5,pdtr	109,ravi,12000,3,gtr
103,anil,12000,3,kdp	110,ramana,11000,5,gtr
104,sunil,13000,3,kdp	111,ram,12000,3,kdp
105,rajiv,16000,6,kdp	112,kavya,13000,4,kdp
106,sunitha,15000,5,nlr	113,navya,14000,5,kdp
107,kavitha,12000,3,nlr	

3. After that the file is saved with **employee.arff** file format.
4. Minimize the arff file and then open Star tPrograms weka-3-4.
5. Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6. In that dialog box there are four modes, click on **explorer**.

7. Explorer shows many options. In that click on **'open file'** and select the arff file
8. Click on **edit button** which shows employee table on weka.

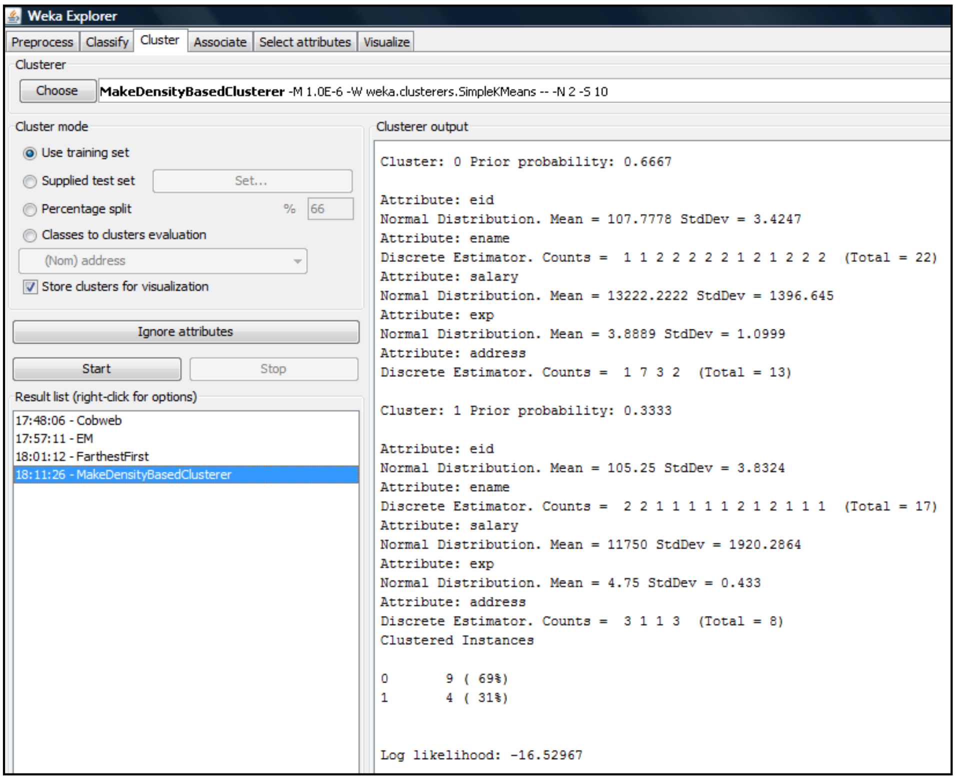
TRAINING DATA SETEMPLOYEE TABLE:



No.	eid Numeric	ename Nominal	salary Numeric	exp Numeric	address Nominal
1	101.0	raj	10000.0	4.0	pdtr
2	102.0	ramu	15000.0	5.0	pdtr
3	103.0	anil	12000.0	3.0	kdp
4	104.0	sunil	13000.0	3.0	kdp
5	105.0	rajiv	16000.0	6.0	kdp
6	106.0	sunitha	15000.0	5.0	nlr
7	107.0	kavitha	12000.0	3.0	nlr
8	108.0	suresh	11000.0	5.0	gtr
9	109.0	ravi	12000.0	3.0	gtr
10	110.0	ramana	11000.0	5.0	gtr
11	111.0	ram	12000.0	3.0	kdp
12	112.0	kavya	13000.0	4.0	kdp
13	113.0	navya	14000.0	5.0	kdp

PROCEDURE:

1. Click Start -> Programs -> Weka 3.4
2. Click on Explorer.
3. Click on open file & then select Employee.arff file.
4. Click on Cluster menu. In this there are different algorithms are there.
5. Click on Choose button and then select MakeDensityBasedClusterer algorithm.
6. Click on Start button and then output will be displayed on the screen.



RESULT:

Thus the Employee data using Density Based Cluster Algorithm has been successfully executed.

EX.NO:8(e)	K-MEANS ALGORITHM
DATE:	

AIM:

To write a procedure for Clustering Customer data using Simple KMeans Algorithm.

DESCRIPTION:

- Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters.
- Clustering is a main task of explorative data mining, and a common technique for statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

CREATION OF CUSTOMER TABLE:

PROCEDURE:

1. Open Start Programs Accessories Notepad.
2. Type the following training data set with the help of Notepad for Customer Table.

   @relation customer

   @attribute name {x,y,z,u,v,l,w,q,r,n}

   @attribute age {youth,middle,senior}

   @attribute income {high,medium,low}

   @attribute class {A,B}

   @data

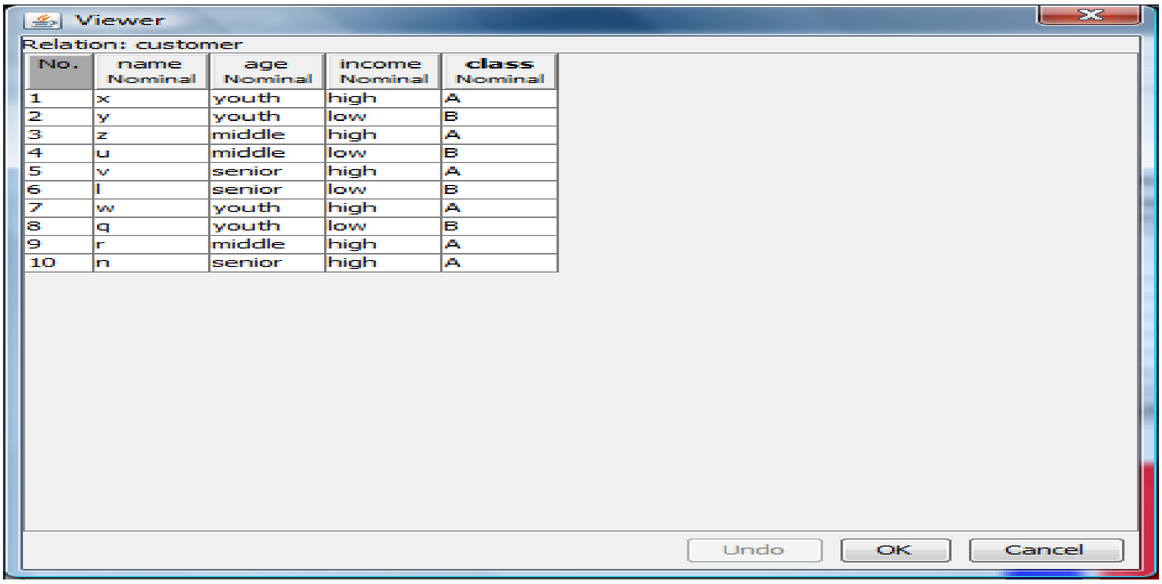
   @data

x,youth,high,A y,youth,low,B z,middle,high,A u,middle,low,B v,senior,high,A l,senior,low,B	w,youth,high,A q,youth,low,B r,middle,high,A n,senior,high,A

3. After that the file is saved with **customer.arff** file format.
4. Minimize the arff file and then open Start Programs weka-3-4.
5. Click on **weka-3-4**, then Weka dialog box is displayed on the screen.
6. In that dialog box there are four modes, click on **explorer**.
7. Explorer shows many options. In that click on **'open file'** and select the arff file

8. Click on **edit button** which shows buying table on weka.

TRAINING DATA SET CUSTOMER TABLE:

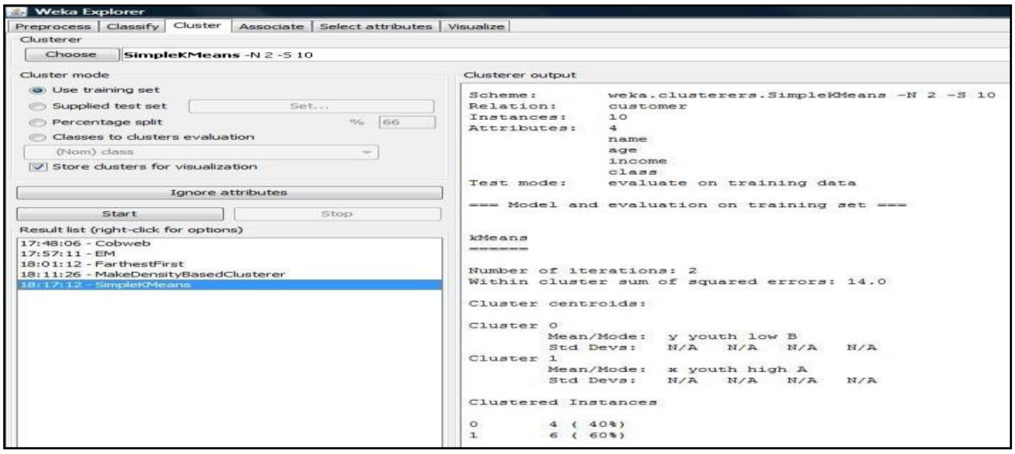


No.	name Nominal	age Nominal	income Nominal	class Nominal
1	x	youth	high	A
2	y	youth	low	B
3	z	middle	high	A
4	u	middle	low	B
5	v	senior	high	A
6	l	senior	low	B
7	w	youth	high	A
8	q	youth	low	B
9	r	middle	high	A
10	n	senior	high	A

PROCEDURE:

1. Click Start -> Programs -> Weka 3.4
2. Click on Explorer.
3. Click on open file & then select Customer.arff file.
4. Click on Cluster menu. In this there are different algorithms are there.
5. Click on Choose button and then select SimpleKMeans algorithm.
6. Click on Start button and then output will be displayed on the screen.

OUTPUT:



RESULT:

Thus the Clustering Customer data using Simple KMeans Algorithm has been successfully executed.

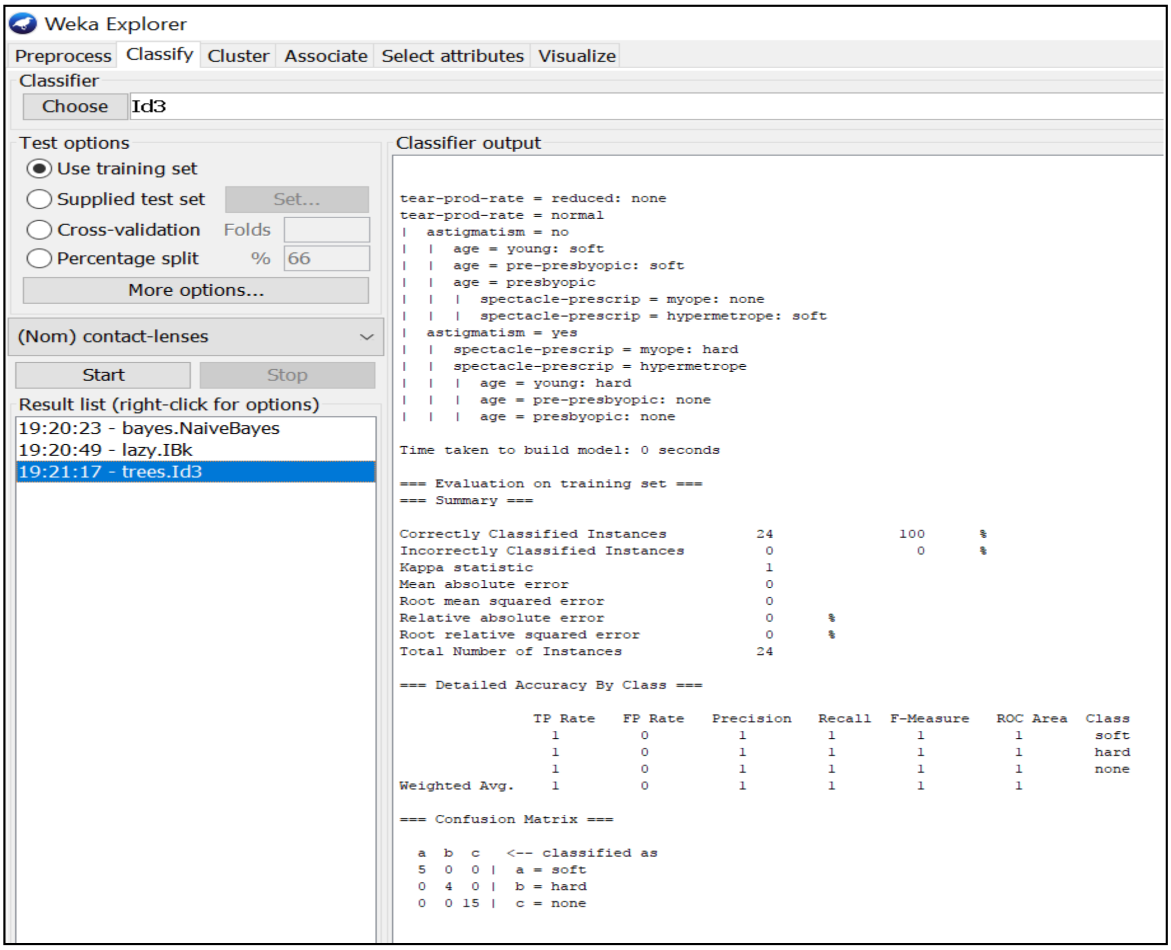
EX.NO: 09	CLASSIFIER MODELS
DATE:	

AIM:

Compare classification results of ID3, J48, Naive-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.

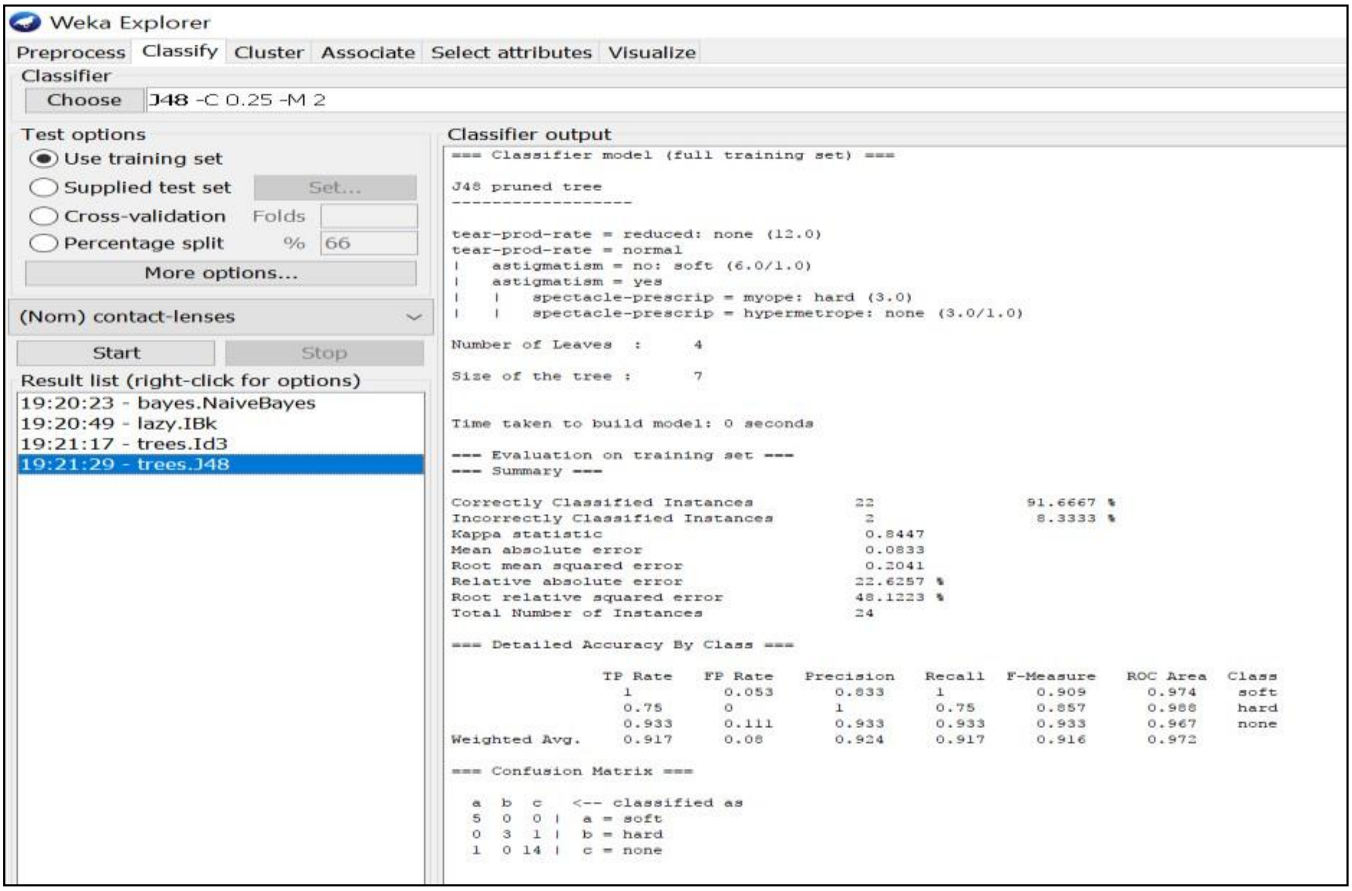
PROCEDURE FOR ID3:

1. Load the dataset (Contact-Lenses. arff) into weka tool.
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under trees section.
3. In which we selected ID3 algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision,Recall values& Confusion Matrix as represented below.



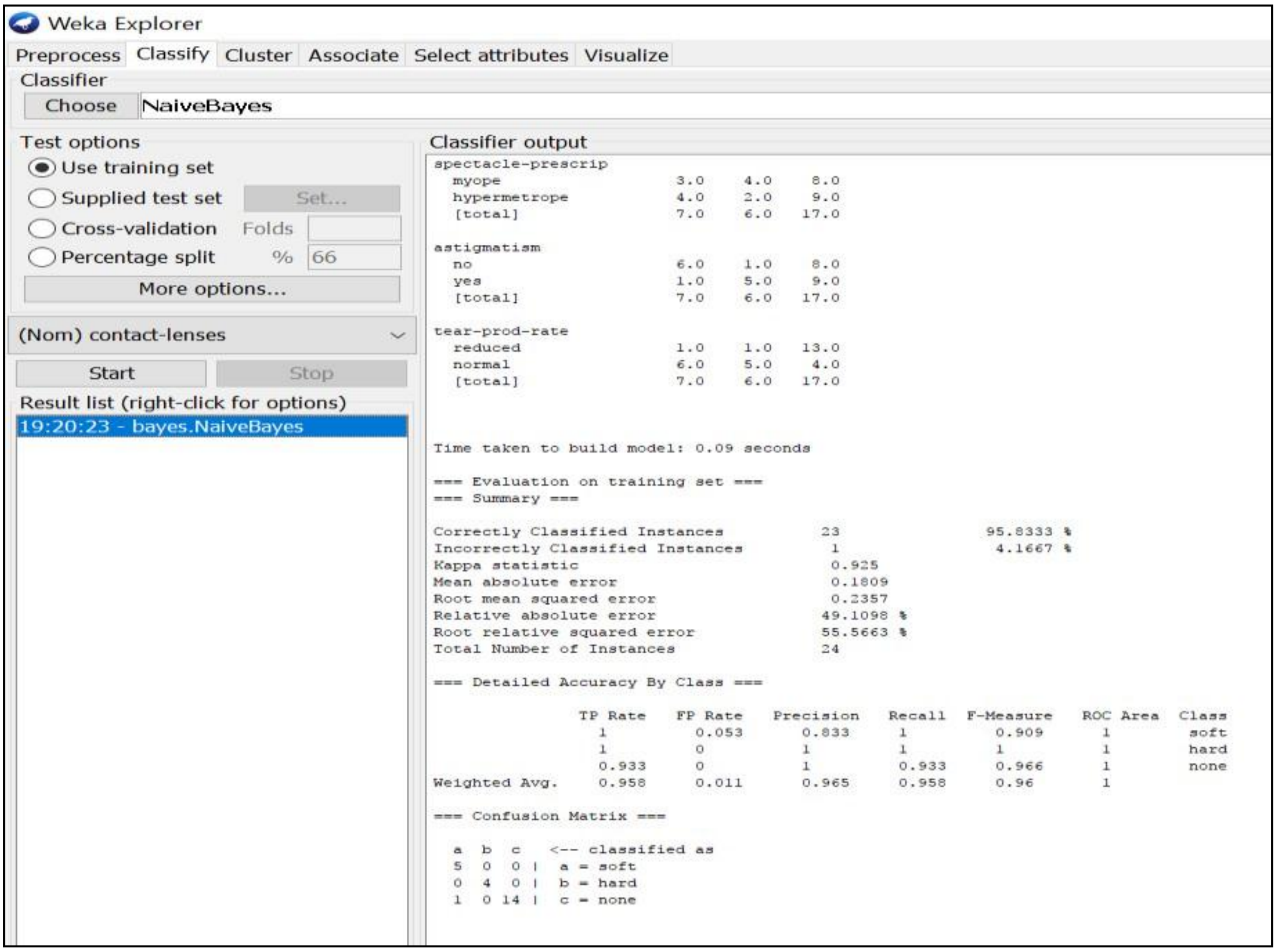
PROCEDURE FOR J48:

1. Load the dataset (Contact-Lenses. arff) into weka tool.
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under trees section.
3. In which we selected J48 algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.



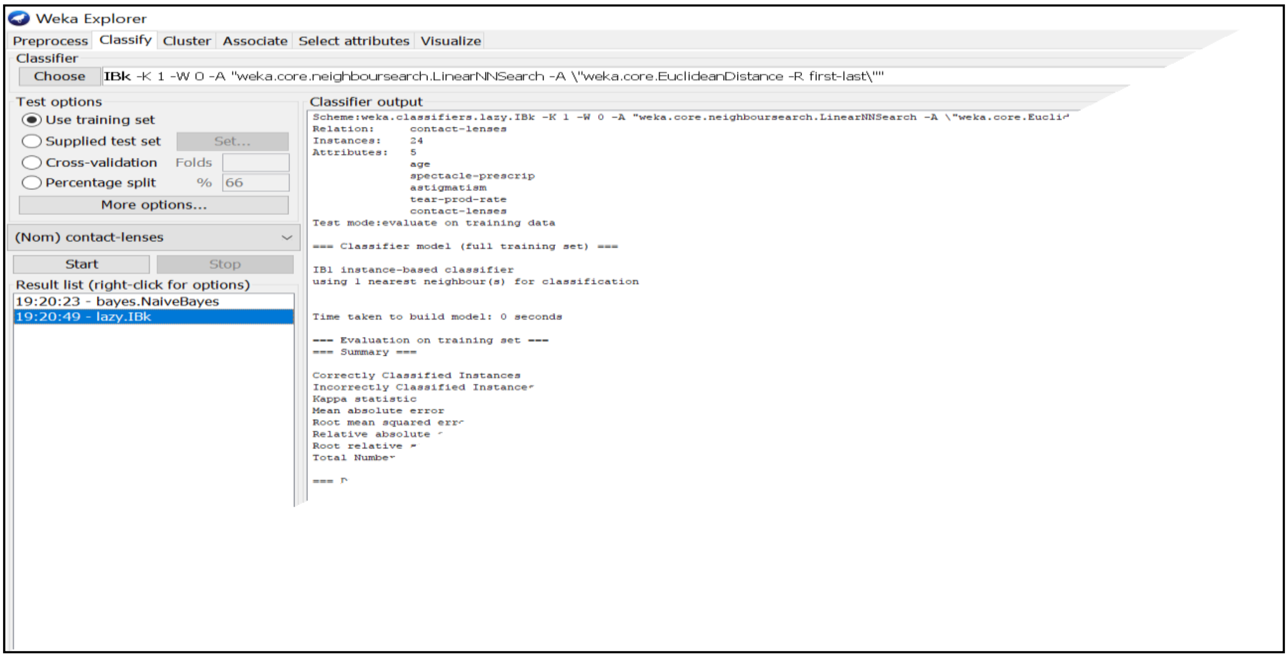
PROCEDURE FOR NAIVE-BAYES:

1. Load the dataset (Contact-Lenses. arff) into weka tool.
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under trees section.
3. In which we selected Naïve-Bayes algorithm & click on start option with ―use training set‖ testoption enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.



PROCEDURE FOR K-NEAREST NEIGHBOUR (IBK):

1. Load the dataset (Contact-Lenses. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under trees section.
3. In which we selected K-Nearest Neighbor (IBK) algorithm & click on start option with usetraining set|| test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.



RESULT:

By observing all these algorithms (ID3, K-NN, J48 & Naïve Bayes) results, we will conclude that ID3 Algorithm's accuracy & performance is best than J48 Algorithm.

EX.NO: 10	DATA MINING APPLICATION
DATE:	

AIM:

To study about data mining applications for real time problems.

ORACLE DATA MINING:

- Oracle Data Mining provides a powerful, state-of-the-art data mining capability within OracleDatabase.
- You can use Oracle Data Mining to build and deploy predictive and descriptive data mining applications, to add intelligent capabilities to existing applications, and to generate predictivequeries for data exploration.
- Oracle Data Mining offers a comprehensive set of in-database algorithms for performing a variety of mining tasks, such as classification, regression, anomaly detection, feature extraction,clustering, and market basket analysis.
- The algorithms can work on standard case data, transactional data, star schemas, and text and other forms of unstructured data. Oracle Data Mining is uniquely suited to the mining of verylarge data sets.
- Oracle Data Mining is one of the two components of the **Oracle Advanced Analytics Option** ofOracle Database Enterprise Edition.
- The other component is Oracle R Enterprise, which integrates R, the open-source statistical environment, with Oracle Database.
- Together, Oracle Data Mining and Oracle R Enterprise constitute a comprehensive advancedanalytics platform for big data analytics.

DATA MINING IN THE DATABASE KERNEL:

- Oracle Data Mining is implemented in the Oracle Database kernel. Data Mining models are firstclass database objects.
- Oracle Data Mining processes use built-in features of Oracle Database to maximize scalabilityand make efficient use of system resources.

DATA MINING WITHIN ORACLE DATABASE OFFERS MANY ADVANTAGES:

- No Data Movement: Some data mining products require that the data be exported from acorporate database and converted to a specialized format for mining.
- With Oracle Data Mining, no data movement or conversion is needed.
- This makes the entire mining process less complex, time-consuming, and error-prone, and itallows for the mining of very large data sets.

- **Security:** Your data is protected by the extensive security mechanisms of Oracle Database. Moreover, specific database privileges are needed for different data mining activities. Only users with the appropriate privileges can define, manipulate, or apply mining model objects.
- **Data Preparation and Administration:** Most data must be cleansed, filtered, normalized, sampled, and transformed in various ways before it can be mined. Up to 80% of the effort in a data mining project is often devoted to data preparation. Oracle Data Mining can automatically manage key stepsin the data preparation process. Additionally, Oracle Database provides extensive administrative tools for preparing and managing data.
- **Ease of Data Refresh: Mining** processes within Oracle Database have ready access to refreshed data. Oracle Data Mining can easily deliver mining results based on current data, thereby maximizing its timeliness and relevance.
- **Oracle Database Analytics:** Oracle Database offers many features for advanced analytics and business intelligence. Oracle Data Mining can easily be integrated with other analytical features ofthe database, such as statistical analysis and OLAP.
- **Oracle Technology Stack:** You can take advantage of all aspects of Oracle's technology stack to integrate data mining within a larger framework for business intelligence or scientific inquiry.
- **Domain Environment:** Data mining models have to be built, tested, validated, managed, and deployed in their appropriate application domain environments. Data mining results may need to be post-processed as part of domain specific computations (for example, calculating estimated risks and response probabilities) and then stored into permanent repositories or data warehouses. With Oracle Data Mining, the pre- and post-mining activities can all be accomplished within the same environment.
- **Application Programming Interfaces**: The PL/SQL API and SQL language operators providedirect access to Oracle Data Mining functionality in Oracle Database.

DATA MINING IN ORACLE EXADATA:

1. Scoring refers to the process of applying a data mining model to data to generate predictions. The scoring process may require significant system resources.
2. Vast amounts of data may be involved, and algorithmic processing may be very complex.
3. With Oracle Data Mining, scoring can be off-loaded to intelligent Oracle Exadata Storage Servers where processing is extremely performant.
4. Oracle Exadata Storage Servers combine Oracle's smart storage software and Oracle's industry- standard Sun hardware to deliver the industry's highest database storage performance.
5. For more information about Oracle Exadata, visit the Oracle Technology Network.

ABOUT PARTITIONED MODEL:

1. Oracle Data Mining supports building of a persistent Oracle Data Mining partitioned model.

2. A partitioned model organizes and represents multiple models as partitions in a single model entity, enabling a user to easily build and manage models tailored to independent slices of data.
3. Persistent means that the partitioned model has an on-disk representation.
4. The product manages the organization of the partitioned model and simplifies the process ofscoring the partitioned model.
5. You must include the partition columns as part of the USING clause when scoring.
6. The partition names, key values, and the structure of the partitioned model are visible inthe ALL_MINING_MODEL_PARTITIONS view.

INTERFACES TO ORACLE DATA MINING:

1. The programmatic interfaces to Oracle Data Mining are PL/SQL for building and maintaining models and a family of SQL functions for scoring.
2. Oracle Data Mining also supports a graphical user interface, which is implemented as anextension to Oracle SQL Developer.
3. Oracle Predictive Analytics, a set of simplified data mining routines, is built on top of OracleData Mining and is implemented as a PL/SQL package.

PL/SQL API:

1. The Oracle Data Mining PL/SQL API is implemented in the DBMS_DATA_MINING PL/SQL package, which contains routines for building, testing, and maintaining data mining models.
2. A batch apply operation is also included in this package.
3. The following example shows part of a simple PL/SQL script for creating an SVM classificationmodel called SVMC_SH_Clas_sample.
4. The model build uses weights, specified in a weights table, and settings, specified in a settings table.
5. The weights influence the weighting of target classes.
6. The settings override default behavior.
7. The model uses Automatic Data Preparation (prep_auto_on setting).
8. The model is trained on the data in mining_data_build_v.

EXAMPLE 1-1 CREATING A CLASSIFICATION MODEL:

COPY:

```
COPY :
----------------------- CREATE AND POPULATE A CLASS WEIGHTS TABLE ------------
CREATE TABLE svmc_sh_sample_class_wt (
 target_value NUMBER,
 class_weight NUMBER);
INSERT INTO svmc_sh_sample_class_wt VALUES (0,0.35);
INSERT INTO svmc_sh_sample_class_wt VALUES (1,0.65);
COMMIT :
----------------------- CREATE AND POPULATE A SETTINGS TABLE ------------------
CREATE TABLE svmc_sh_sample_settings (
 setting_name  VARCHAR2(30),
 setting_value VARCHAR2(4000));


  BEGIN :
  INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
   (dbms_data_mining.algo_name, dbms_data_mining.algo_support_vector_machines);
  INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
   (dbms_data_mining.svms_kernel_function, dbms_data_mining.svms_linear);



 INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
  (dbms_data_mining.clas_weights_table_name, 'svmc_sh_sample_class_wt');
 INSERT INTO svmc_sh_sample_settings (setting_name, setting_value) VALUES
  (dbms_data_mining.prep_auto, dbms_data_mining.prep_auto_on);
 END;
 /
 ----------------------- CREATE THE MODEL ------------------------------------
 BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
   model_name          => 'SVMC_SH_Clas_sample',
   mining_function     => dbms_data_mining.classification,
   data_table_name     => 'mining_data_build_v',
   case_id_column_name => 'cust_id',
   target_column_name  => 'affinity_card',
   settings_table_name => 'svmc_sh_sample_settings');
 END;
```

SQL FUNCTIONS:

1. The Data Mining SQL functions perform prediction, clustering, and feature extraction.
2. The functions score data by applying a mining model object or by executing an analytic clausethat performs dynamic scoring.
3. The following example shows a query that applies the classification model svmc_sh_clas_sample to the data in the view mining_data_apply_v.
4. The query returns the average age of customers who are likely to use an affinity card.
5. The results are broken out by gender.

EXAMPLE 1-2 THE PREDICTION FUNCTION:

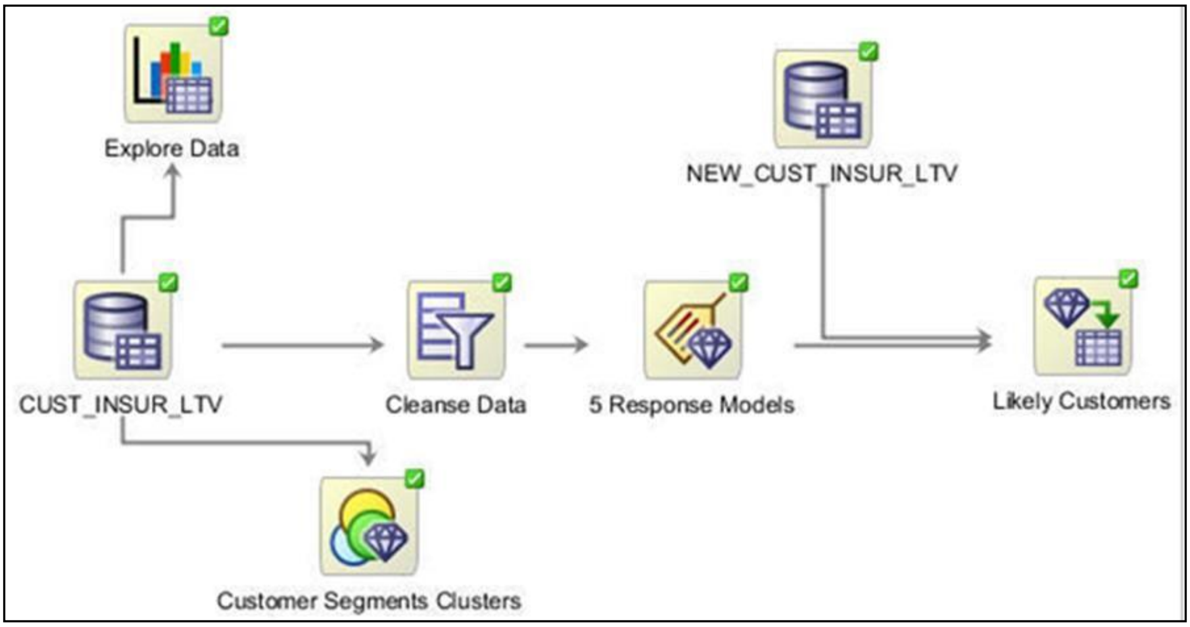
COPY:

```
SELECT cust_gender,COUNT(*) AS cnt, ROUND(AVG(age)) AS avg_age FROM
mining_data_apply_v WHERE PREDICTION(svmc_sh_clas_sample USING *) = 1 GROUP
BY cust_gender ORDER BY cust_gender;
```

C	CNT	AVG_AGE
F	19	41
M	409	45

ORACLE DATA MINER:

1. Oracle Data Miner is a graphical interface to Oracle Data Mining.
2. Oracle Data Miner is an extension to Oracle SQL Developer, which is available for downloadfree of charge on the Oracle Technology Network.
3. Oracle Data Miner uses a work flow paradigm to capture, document, and automate the process ofbuilding, evaluating, and applying data mining models.
4. Within a work flow, you can specify data transformations, build and evaluate multiple models, and score multiple data sets.
5. You can then save work flows and share them with other users.



PREDICTIVE ANALYTICS:

1. Predictive analytics is a technology that captures data mining processes in simple routines.
2. Sometimes called "one-click data mining," predictive analytics simplifies and automates the datamining process.
3. Predictive analytics uses data mining technology, but knowledge of data mining is not needed touse predictive analytics.
4. You can use predictive analytics simply by specifying an operation to perform on your data.
5. You do not need to create or use mining models or understand the mining functions andalgorithms summarized in "Oracle Data Mining Basics ".

Oracle Data Mining predictive analytics operations are described in the following table:

Table 1-1 Oracle Predictive Analytics Operations

Operation	Description
EXPLAIN	Explains how individual predictors (columns) affect the variation of values in a target column
PREDICT	For each case (row), predicts the values in a target column
PROFILE	Creates a set of rules for cases (rows) that imply the same target value

1. The Oracle predictive analytics operations are implemented in the DBMS_PREDICTIVE_ANALYTICS PL/SQL package.
2. They are also available in Oracle Data Miner.

OVERVIEW OF DATABASE ANALYTICS:

1. Oracle Database supports an array of native analytical features that are independent of the OracleAdvanced Analytics Option.
2. Since all these features are part of a common server it is possible to combine them efficiently.
3. The results of analytical processing can be integrated with Oracle Business Intelligence Suite Enterprise Edition and other BI tools and applications.
4. The possibilities for combining different analytics are virtually limitless.
5. Example 1-3 shows data mining and text processing within a single SQL query.
6. The query selects all customers who have a high propensity to attrite (> 80% chance), are valuable customers (customer value rating > 90), and have had a recent conversation withcustomer services regarding a Checking Plus account.
7. The propensity to attrite information is computed using a Data Mining model called tree_model.
8. The query uses the Oracle Text CONTAINS operator to search call center notes for references toChecking Plus accounts.

Some of the native analytics supported by Oracle Database are described in the following table:Table 1-2 Oracle Database Native Analytics

Analytical Feature	Description	Documented In...
Complex data transformations	Data transformation is a key aspect of analytical applications and ETL (extract, transform, and load). You can use SQL expressions to implement data transformations, or you can use the DBMS_DATA_MINING_TRANSFORMpackage. DBMS_DATA_MINING_TRANSFORM is a flexible data transformation package that includes a variety of missing value and outlier treatments, as well as binningand normalization capabilities.	Oracle Database PL/SQL Packages and Types reference.
Statistical functions	Oracle Database provides a long list of SQL statistical functions with support for: hypothesis testing (such as t-test, F-test), correlation computation (such as pearsoncorrelation), cross-tab statistics, and descriptive statistics (such as median and mode). The DBMS_STAT_FUNCS package adds distribution fitting procedures and a summary	Oracle Database SQL Language Reference and OracleDatabase PL/SQL Packages and Types

	procedure that returns descriptive statistics for a column.	Reference
Window and analytic SQL functions	Oracle Database supports analytic and windowing functions for computing cumulative, moving, and centered aggregates. With windowing aggregate functions, you can calculate moving and cumulative versions of SUM, AVERAGE, COUNT, MAX, MIN, and many more functions.	Oracle Database DataWarehousing Guide
Linear algebra	The UTL_NLA package exposes a subset of the popular BLAS and LAPACK (Version 3.0) libraries for operations on vectors and matrices represented as VARRAYs. This package includes procedures to solve systems of linear equations, invert matrices, and compute eigenvalues and eigenvectors.	Oracle Database PL/SQL Packages andTypes Reference
OLAP	Oracle OLAP supports multidimensional analysis and can be used to improve performance of multidimensional queries. Oracle OLAP provides functionality previously found only in specialized OLAP databases. Moving beyond drill-downs and roll- ups, Oracle OLAP also supports time-series analysis, modeling, and forecasting.	Oracle OLAP User's Guide
Spatial analytics	Oracle Spatial provides advanced spatial features to support high-end GIS and LBS solutions. Oracle Spatial's analysis and mining capabilities include functions for binning, detection of regional patterns, spatial correlation, colocation mining, and spatial clustering. Oracle Spatial also includes support for topology and network data models and analytics. The topology data model of Oracle Spatial allows one to work with data about nodes, edges, and faces in a topology. It includesnetwork analysis functions for computing shortest path, minimum cost spanning tree, nearest-neighbors analysis, and traveling salesman problem, among others.	Oracle Spatial and Graph Developer's Guide
Text Mining	Oracle Text uses standard SQL to index, search, and analyze text and documents stored in the Oracle database, in files, and on the web. Oracle Text also supports automatic classification and clustering of document collections. Many of the analytical features of Oracle Text are layered on top of Oracle Data Mining functionality.	Oracle Text Application Developer's Guide

Example 1-3 SQL Query Combining Oracle Data Mining and Oracle Text

COPY :

```
SELECT A.cust_name, A.contact_info FROM customers A WHERE
PREDICTION_PROBABILITY(tree_model,'attrite' USING A.*) > 0.8 AND A.cust_value > 90

AND A.cust_id IN(SELECT B.cust_id FROM call_center B WHERE B.call_date BETWEEN

'01-Jan-2005' AND '30-Jun-2005'   AND CONTAINS(B.notes, 'Checking Plus', 1) > 0);
```

RESULT:

Thus the study about data mining applications for real time problems has been successfully executed.