



योग: कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE
An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

B.Tech Programme: Computer Science & Engineering

Course Title: Data Warehousing Data Mining Lab

Course Code: CIE-405P

Semester: 7th

Submitted By

Name: Shubham Gupta

Enrollment No: 20617702722

Branch & Section: CSE-A (G2)



An ISO 9001:2015 Certified Institution
SCHOOL OF ENGINEERING & TECHNOLOGY

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS
Grade A++ Accredited Institution by NAAC
NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

VISION OF INSTITUTE

To be an educational institute that empowers the field of engineering to build a sustainable future by providing quality education with innovative practices that supports people, planet and profit.

MISSION OF INSTITUTE

To groom the future engineers by providing value-based education and awakening students' curiosity, nurturing creativity and building capabilities to enable them to make significant contributions to the world.



योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

An ISO 9001:2015 Certified Institution
SCHOOL OF ENGINEERING & TECHNOLOGY

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE


INDEX

[illegible]

[illegible]



योग: कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

			 <p>योग: कर्मसु कौशलम् IN PURSUIT OF PERFECTION</p>					

EXPERIMENT 1

Aim: Study of ETL process and its tools.

Theory: ETL Process (Extract, Transform, Load)

The ETL process is a key part of data warehousing that helps in moving and preparing data for analysis. In this process, data is first extracted from multiple sources, transformed into a clean and consistent format, and then loaded into a data warehouse. This ensures that organizations have accurate, well-structured data available for reporting, analytics, and decision-making.

- Extract: Collect data from multiple sources like databases, files, and applications.
- Transform: Clean, format, and convert data into a common structure; handle errors and missing values.
- Load: Store the processed data into a data warehouse for reporting and analysis.

Dataset:

Purpose: A small retail sales log for Jan–Jun 2025. Each row = one order.

Columns:

- order_id: Unique ID (2001–2025).
- date: Order date (DD-MM-2025).
- customer_name: Buyer's name.
- product: Item bought (Laptop, Mouse, Keyboard, Monitor, plus a few like Tablet, Printer, Desk Lamp, Office Chair).
- quantity: Units purchased (mostly 1–3).
- price: Per-unit price in the same currency.
- region: Customer/market region (North, South, East, West).

Program:

```
import pandas as pd
sales_data=pd.read_csv('sales_data.csv')
print(sales_data.head())
```

	order_id	date	customer_name	product	quantity	price	region
0	1001	2025-01-15	Alice Johnson	Laptop	1.0	1299.99	North
1	1002	2025-01-17	Bob Williams	Mouse	2.0	24.50	South
2	1003	2025-01-20	Charlie Brown	Keyboard	1.0	75.00	East
3	1004	2025-01-22	David Smith	Keyboard	1.0	NaN	East
4	1005	2025-02-05	Eva Green	Monitor	2.0	300.50	West

```

sales_data_cleaned = sales_data.fillna(0)

sales_data_cleaned['total_amount'] = sales_data_cleaned['quantity'] * sales_data_cleaned['price']

display(sales_data_cleaned['total_amount'].head())

display(sales_data_cleaned.head())

```



	total_amount
0	1299.99
1	49.00
2	75.00
3	0.00
4	601.00

dtype: float64

	order_id	date	customer_name	product	quantity	price	region	total_amount
0	1001	2025-01-15	Alice Johnson	Laptop	1.0	1299.99	North	1299.99
1	1002	2025-01-17	Bob Williams	Mouse	2.0	24.50	South	49.00
2	1003	2025-01-20	Charlie Brown	Keyboard	1.0	75.00	East	75.00
3	1004	2025-01-22	David Smith	Keyboard	1.0	0.00	East	0.00
4	1005	2025-02-05	Eva Green	Monitor	2.0	300.50	West	601.00

```
[ ] sales_data_cleaned.to_csv('sales_data_cleaned.csv', index=False)
```

```

cleaned_file=pd.read_csv('sales_data_cleaned.csv')
display(cleaned_file.head())

```



	order_id	date	customer_name	product	quantity	price	region	total_amount
0	1001	2025-01-15	Alice Johnson	Laptop	1.0	1299.99	North	1299.99
1	1002	2025-01-17	Bob Williams	Mouse	2.0	24.50	South	49.00
2	1003	2025-01-20	Charlie Brown	Keyboard	1.0	75.00	East	75.00
3	1004	2025-01-22	David Smith	Keyboard	1.0	0.00	East	0.00
4	1005	2025-02-05	Eva Green	Monitor	2.0	300.50	West	601.00

Learning Outcomes:

EXPERIMENT 2

Aim: Program of Data warehouse cleansing to input names from users (inconsistent) and format them.

Theory: Data warehouse cleansing is part of the Transform stage in the ETL (Extract, Transform, Load) process. It improves data quality by making it accurate, complete, and consistent before storing it in the warehouse.

When users input names, inconsistencies like wrong case, extra spaces, or typos may occur. Cleansing ensures a uniform format, e.g., " john DOE " → "John Doe".

In Python, common string functions used for cleansing names are:

- `strip()` – Removes extra spaces from start and end.
- `title()` – Converts text to Title Case.
- `lower()` / `upper()` – Converts text to lower or upper case if needed.
- `replace()` – Replaces unwanted characters or patterns.

Clean data ensures accurate analytics and better decision-making.

Dataset:

The column `Raw_name` contains user-entered names in inconsistent formats.


It includes common formatting issues such as:

- Leading and trailing spaces
- Multiple spaces between first and last names
- Random mix of uppercase and lowercase letters
- Same names entered in different formatting styles

This dataset is intentionally kept “dirty” so that it can be used for practicing data cleansing and standardization tasks.

Program:

```
[12] import pandas as pd
      name_data=pd.read_csv('name_raw_data.csv')
      print(name_data.head())
```



	Raw_name
0	RaHuL ShArMa
1	rahul SHARma
2	RAHUL sharma
3	PrIyA KaPoOr
4	priya KAPOOR


```
name_data['cleaned_name'] = name_data['Raw_name'].str.strip()
name_data['cleaned_name'] = name_data['cleaned_name'].str.title()
name_data['cleaned_name'] = name_data['cleaned_name'].apply(lambda x: ' '.join(x.split()))

name_data = name_data.drop_duplicates(subset=['cleaned_name'])

display(name_data.head())
```



	Raw_name	cleaned_name
0	RaHuL ShArMa	Rahul Sharma
3	PriyA KaPoOr	Priya Kapoor
6	aMiT KuMaR	Amit Kumar
9	SuNiTa sInGh	Sunita Singh
12	MoHiT VeRmA	Mohit Verma





```
[17] name_data.to_csv('name_data_cleaned.csv', index=False)
```

Output:

Files



- ..
- .config
- .ipynb_checkpoints
- drive
- sample_data
- name_data_cleaned.csv  
- name_raw_data.csv

Learning Outcomes:

EXPERIMENT 3

Aim: Program of Data warehouse cleansing to remove redundancy in data.

Theory:

Redundancy occurs when **duplicate records** or **repeated attributes** exist in data.

DW cleansing removes:

- Exact duplicates (same row)
 - Near duplicates (same name with different spellings)
 - Repeated entries due to system errors
- Redundancy increases storage cost and leads to incorrect analytics.

Techniques include:

- Primary key-based deduplication
- Hashing on rows
- Similarity-based duplicate detection (Levenshtein)
- Canonical formatting (lowercasing, trimming)

Dataset:

customer_id	customer_name	email
101	John Doe	john@gmail.com
102	john doe	john@gmail.com
103	Priya Sharma	priya@gmail.com
104	PRIYA SHARMA	priya@gmail.com
105	Rohit Verma	rohit@gmail.com

Goal → remove duplicates & standardize names.

Program:

```
!pip install fuzzywuzzy
!pip install python-Levenshtein
```

```

Collecting fuzzywuzzy
  Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl.metadata (4.9 kB)
  Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)
  Installing collected packages: fuzzywuzzy
  Successfully installed fuzzywuzzy-0.18.0
Collecting python-Levenshtein
  Downloading python_levenshtein-0.27.1-py3-none-any.whl.metadata (3.7 kB)
  Collecting Levenshtein==0.27.1 (from python-Levenshtein)
  Downloading levenshtein-0.27.1-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.6 kB)
  Collecting rapidfuzz<4.0.0,>=3.9.0 (from Levenshtein==0.27.1->python-Levenshtein)
  Downloading rapidfuzz-3.14.1-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (12 kB)
  Downloading python_levenshtein-0.27.1-py3-none-any.whl (9.4 kB)
  Downloading levenshtein-0.27.1-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (159 kB)
  Downloading rapidfuzz-3.14.1-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (3.2 MB)
  159.9/159.9 kB 3.2 MB/s eta 0:00:00
  3.2/3.2 MB 39.3 MB/s eta 0:00:00
  Installing collected packages: rapidfuzz, Levenshtein, python-Levenshtein
  Successfully installed Levenshtein-0.27.1 python-Levenshtein-0.27.1 rapidfuzz-3.14.1

```

```

import pandas as pd
from fuzzywuzzy import fuzz
from fuzzywuzzy import process
import matplotlib.pyplot as plt

customer_data=pd.read_csv('customer_data.csv')
print(customer_data.head())

customer_data['Name']=customer_data['Name'].str.title()
customer_data['Address']=customer_data['Name'].str.strip().str.title()

print("After Standardization\n", customer_data)

```

```

Customer_ID      Name      Email      Phone      Address
0      101      John Doe      john@example.com      1234567890      123 Elm St
1      102      Jane Smith      jane@example.com      2345678901      456 Oak St
2      103      john doe      john@example.com      1234567890      123 Elm Street
3      104      Alice Brown      alice@example.com      3456789012      789 Pine St
4      105      Bob Martin      bob@example.com      4567890123      321 Maple Ave
After Standardization
Customer_ID      Name      Email      Phone      Address
0      101      John Doe      john@example.com      1234567890      John Doe
1      102      Jane Smith      jane@example.com      2345678901      Jane Smith
2      103      John Doe      john@example.com      1234567890      John Doe
3      104      Alice Brown      alice@example.com      3456789012      Alice Brown
4      105      Bob Martin      bob@example.com      4567890123      Bob Martin
5      106      Jane Smith      jane@example.com      2345678901      Jane Smith
6      107      Jon Doe      jon@example.com      1234567890      Jon Doe
7      108      Ally Brown      ally@example.com      3456789012      Ally Brown

```

```

customer_data_wo_duplicates=customer_data.drop_duplicates(subset=['Name', 'Address'])
print("After removing duplicates\n", customer_data_wo_duplicates)

customer_data=customer_data_wo_duplicates.drop_duplicates(subset=['Email'], keep='first')
print("After removing duplicates\n", customer_data)

names=customer_data['Name'].tolist()
print('Potential Near Duplicates')

for i in range(len(names)):
    for j in range(i+1, len(names)):
        if fuzz.ratio(names[i], names[j]) >= 80:
            print(names[i], names[j], "Similarity found")

plt.figure(figsize=(8, 4))
customer_data['Name'].str.len().hist()
plt.xlabel("Name Length")
plt.ylabel("Frequency")
plt.title("Distribution of Customer Name Lengths")
plt.show()

```

Output:

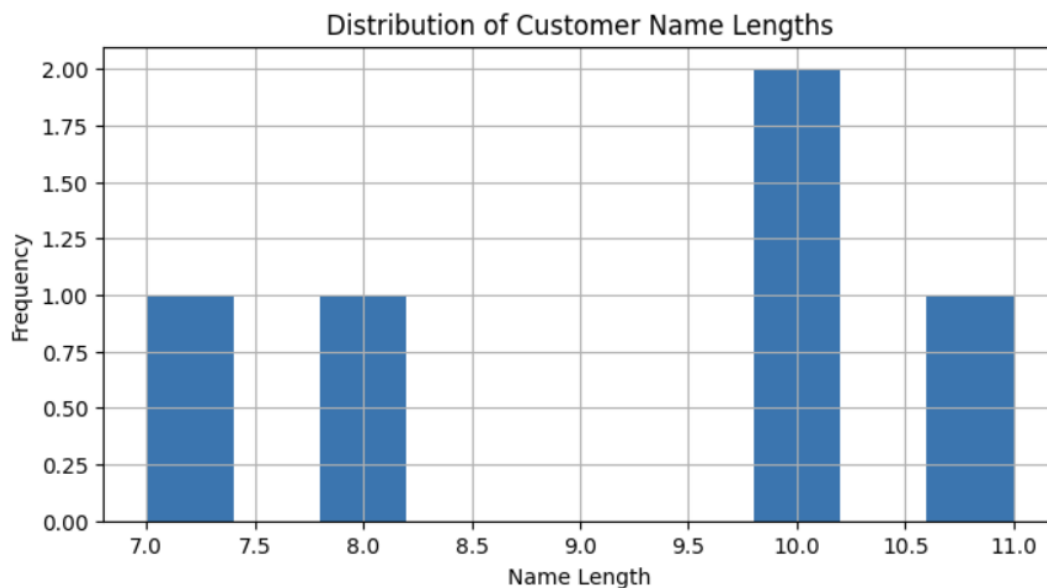
➡ After removing duplicates

	Customer_ID	Name	Email	Phone	Address
0	101	John Doe	john@example.com	1234567890	John Doe
1	102	Jane Smith	jane@example.com	2345678901	Jane Smith
3	104	Alice Brown	alice@example.com	3456789012	Alice Brown
4	105	Bob Martin	bob@example.com	4567890123	Bob Martin
6	107	Jon Doe	jon@example.com	1234567890	Jon Doe
7	108	Ally Brown	alice@example.com	3456789012	Ally Brown

After removing duplicates

	Customer_ID	Name	Email	Phone	Address
0	101	John Doe	john@example.com	1234567890	John Doe
1	102	Jane Smith	jane@example.com	2345678901	Jane Smith
3	104	Alice Brown	alice@example.com	3456789012	Alice Brown
4	105	Bob Martin	bob@example.com	4567890123	Bob Martin
6	107	Jon Doe	jon@example.com	1234567890	Jon Doe

Potential Near Duplicates
John Doe Jon Doe Similarity found



Learning Outcomes:

EXPERIMENT 4

Aim: Introduction to WEKA tool.

Theory:

WEKA (Waikato Environment for Knowledge Analysis) is an open-source data mining tool containing ML algorithms for **classification, clustering, association mining, visualization**.

Functions:

- Preprocessing (cleaning, normalization)
- Classification (J48, Naive Bayes, SVM)
- Clustering (K-Means, EM)
- Association (Apriori)
- Visualization (scatter plots, histograms)
Supports **ARFF, CSV** formats.
Used widely for academic ML experiments.

Dataset:

@relation weather

@attribute outlook {sunny, overcast, rainy}

@attribute temperature numeric

@attribute humidity numeric

@attribute windy {true, false}

@attribute play {yes, no}

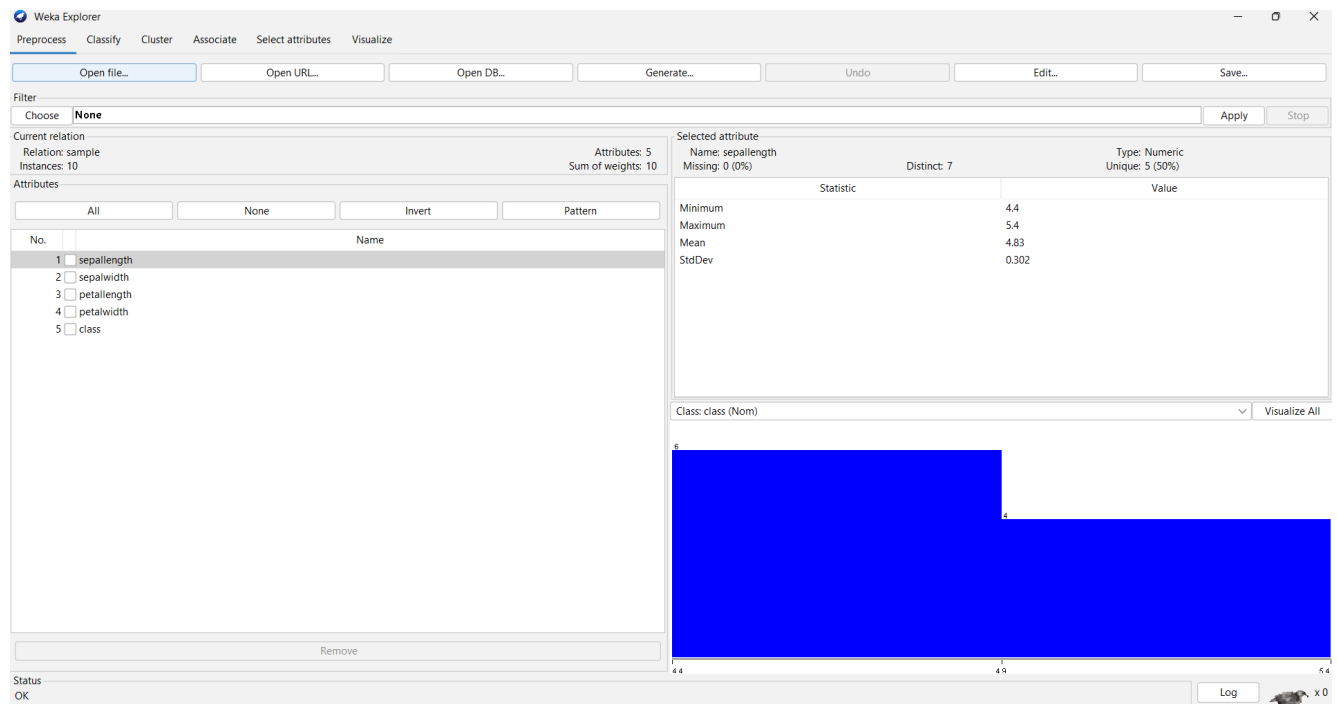
@data

sunny,85,85,false,no

sunny,80,90,true,no

overcast,83,78,false,yes

rainy,70,96,false,yes

WEKA:The image shows the WEKA Explorer interface. The top menu bar includes "Preprocess", "Classify", "Cluster", "Associate", "Select attributes", and "Visualize". Below the menu bar are buttons for "Open file...", "Open URL...", "Open DB...", "Generate...", "Undo", "Edit...", and "Save...". The "Filter" section shows "Choose: None" with "Apply" and "Stop" buttons. The "Current relation" section shows "Relation: sample" and "Instances: 10". The "Attributes" section shows a list of attributes: "1 [x] sepalwidth", "2 [x] sepalwidth", "3 [x] petalwidth", "4 [x] petalwidth", and "5 [x] class". The "Selected attribute" section shows "Name: sepalwidth" and "Missing: 0 (0%)". The "Statistic" table shows the following values: Minimum: 4.4, Maximum: 5.4, Mean: 4.83, and StdDev: 0.302. The "Class: class (Nom)" section shows a bar chart with a blue bar. The status bar at the bottom shows "Status: OK" and a "Log" button.

Weka Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier

Choose **ZeroR**

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 10

☐ Percentage split % 66

More options...

(Num) sepalwidth

Start Stop

Result list (right-click for options)

13:52:54 - rules.ZeroR

Classifier output

```
=== Run information ===

Scheme:      weka.classifiers.rules.ZeroR
Relation:    sample
Instances:   10
Attributes:  5
    sepalength
    sepalwidth
    petallength
    petalwidth
    class
Test mode:   10-fold cross-validation

=== Classifier model (full training set) ===

ZeroR predicts class value: 3.3199999999999994

Time taken to build model: 0 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient      -1
Mean absolute error         0.2667
Root mean squared error     0.3174
Relative absolute error      100 %
Root relative squared error  100 %
Total Number of Instances   10
```

Weka Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier

Choose **ZeroR**

Test options

☐ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 10

☒ Percentage split % 66

More options...

(Num) sepalwidth

Start Stop

Result list (right-click for options)

13:52:54 - rules.ZeroR

13:53:25 - rules.ZeroR

Classifier output

```
=== Run information ===

Scheme:      weka.classifiers.rules.ZeroR
Relation:    sample
Instances:   10
Attributes:  5
    sepalength
    sepalwidth
    petallength
    petalwidth
    class
Test mode:   split 66.0% train, remainder test

=== Classifier model (full training set) ===

ZeroR predicts class value: 3.3199999999999994

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correlation coefficient      0
Mean absolute error         0.3714
Root mean squared error     0.4032
Relative absolute error      100 %
Root relative squared error  100 %
Total Number of Instances   3
```


Preprocess Classify **Cluster** Associate Select attributes Visualize

Clusterer
Choose **EM** -I 100 -N -1 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E-6 -K 10 -num-slots 1 -S 100

Cluster mode
☐ Use training set
☐ Supplied test set Set...
☒ Percentage split % **66**
☐ Classes to clusters evaluation (Nom) class v
☒ Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)
 13:54:23 - EM
 13:54:40 - EM
 13:54:50 - EM

Cluster output
 number of iterations performed: 2

Attribute	Cluster
	0
	(1)
=====	
sepalength	
mean	4.7167
std. dev.	0.2192
sepalwidth	
mean	3.2833
std. dev.	0.2192
petallength	
mean	1.4333
std. dev.	0.1247
petalwidth	
mean	0.2167
std. dev.	0.0687
class	
Iris-setosa	7
[total]	7

Time taken to build model (percentage split) : 0 seconds

Clustered Instances

Cluster	Instances	Percentage
0	4	100%

Log likelihood: -26.13312

Preprocess Classify Cluster Associate **Select attributes** Visualize

Attribute Evaluator
Choose **CfsSubsetEval** -P 1 -E 1

Search Method
Choose **BestFirst** -D 1 -N 5

Attribute Selection Mode
☐ Use full training set
☒ Cross-validation Folds **10** Seed **1**

(Num) sepalength v

Start Stop

Result list (right-click for options)
 13:55:29 - BestFirst + CfsSubsetEval

Attribute selection output

```

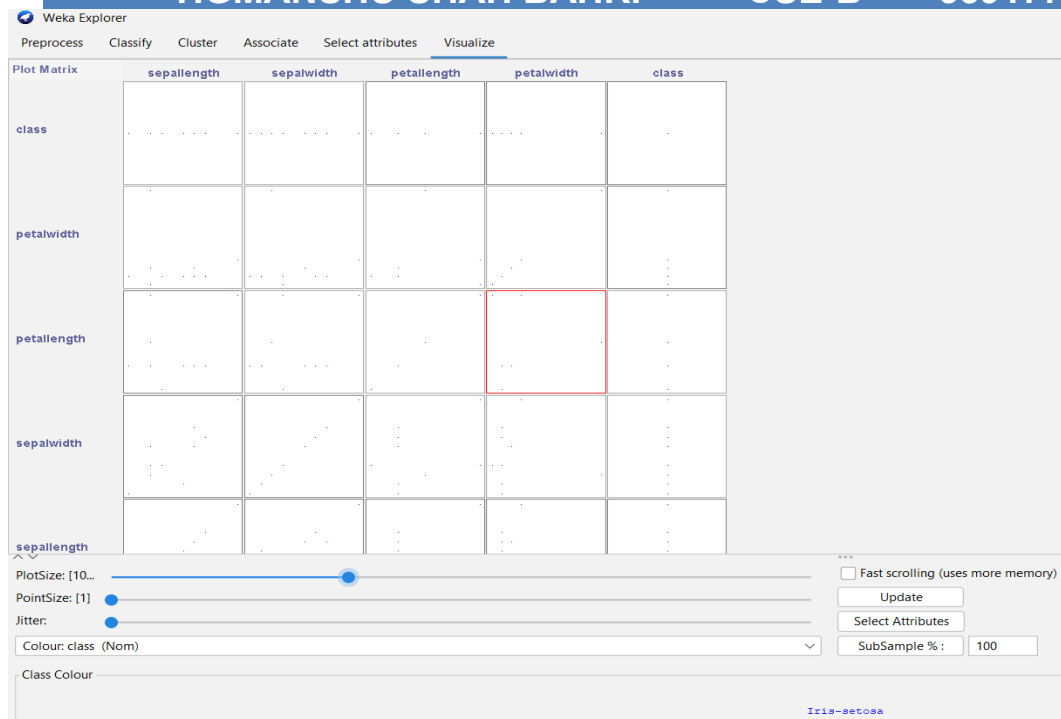
=== Run information ===

Evaluator:   weka.attributeSelection.CfsSubsetEval -P 1 -E 1
Search:     weka.attributeSelection.BestFirst -D 1 -N 5
Relation:   sample
Instances:  10
Attributes:  5
             sepalength
             sepalwidth
             petallength
             petalwidth
             class

Evaluation mode: 10-fold cross-validation

=== Attribute selection 10 fold cross-validation seed: 1 ===

number of folds (%)  attribute
10 (100 %)          2 sepalwidth
2 ( 20 %)           3 petallength
8 ( 80 %)            4 petalwidth
0 ( 0 %)             5 class
  
```



Learning Outcomes:

EXPERIMENT 5

Aim: Implementation of Classification technique on ARFF files using WEKA.

Theory:

Classification predicts a **categorical class** using supervised learning.

Algorithms used in WEKA:

- **J48 (Decision Tree)**
- **Naive Bayes**
- **Random Forest**
- **SVM (SMO)**
Process:

1. Load ARFF
2. Preprocess
3. Choose classifier
4. Train/Test split
5. Evaluate accuracy, precision, confusion matrix

Dataset:

@relation iris

@attribute sepallength numeric

@attribute sepalwidth numeric

@attribute petallength numeric

@attribute petalwidth numeric

@attribute class {setosa, versicolor, virginica}

@data

5.1,3.5,1.4,0.2,setosa

6.2,2.2,4.5,1.5,versicolor

6.5,3.0,5.2,2.0,virginica

WEKA:

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

Filter Choose **None** Apply Stop

Current relation
Relation: iris
Instances: 150
Attributes: 5
Sum of weights: 150

Selected attribute
Name: sepalength
Missing: 0 (0%)
Distinct: 35
Type: Numeric
Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Class: class (Nom) Visualize All

Remove

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier Choose **J48 -C 0.25 -M 2**

Test options
☐ Use training set
☐ Supplied test set
☒ Cross-validation Folds **10**
☐ Percentage split % **66**
 More options...

(Nom) class
 Start Stop

Result list (right-click for options)
16:50:57 - trees.J48

Classifier output

Size of the tree : 9

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
 === Summary ===

Metric	Value
Correctly Classified Instances	144 96 %
Incorrectly Classified Instances	6 4 %
Kappa statistic	0.94
Mean absolute error	0.035
Root mean squared error	0.1566
Relative absolute error	7.8705 %
Root relative squared error	33.6253 %
Total Number of Instances	150

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.980	0.000	1.000	0.980	0.990	0.995	0.990	0.987	Iris-set
	0.940	0.030	0.940	0.940	0.940	0.910	0.952	0.880	Iris-ver
	0.960	0.030	0.941	0.960	0.950	0.925	0.961	0.905	Iris-vir
Weighted Avg.	0.960	0.020	0.960	0.960	0.960	0.940	0.968	0.924	

=== Confusion Matrix ===

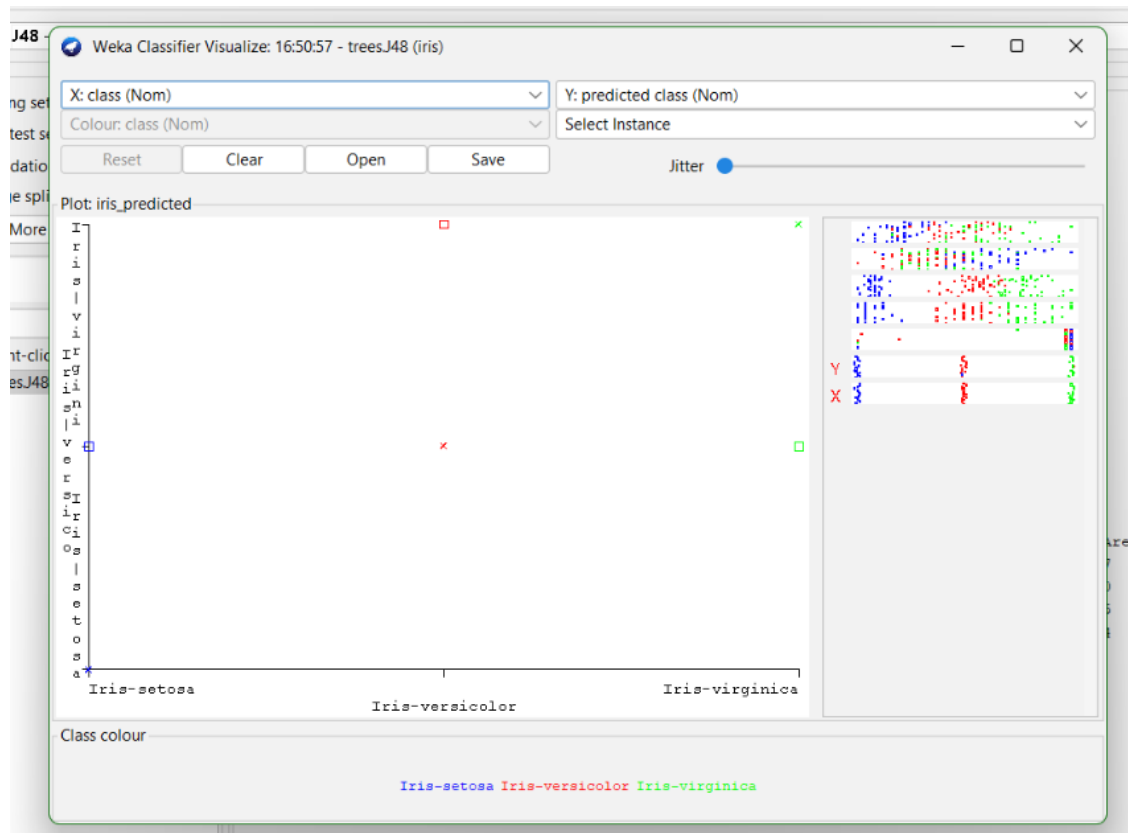
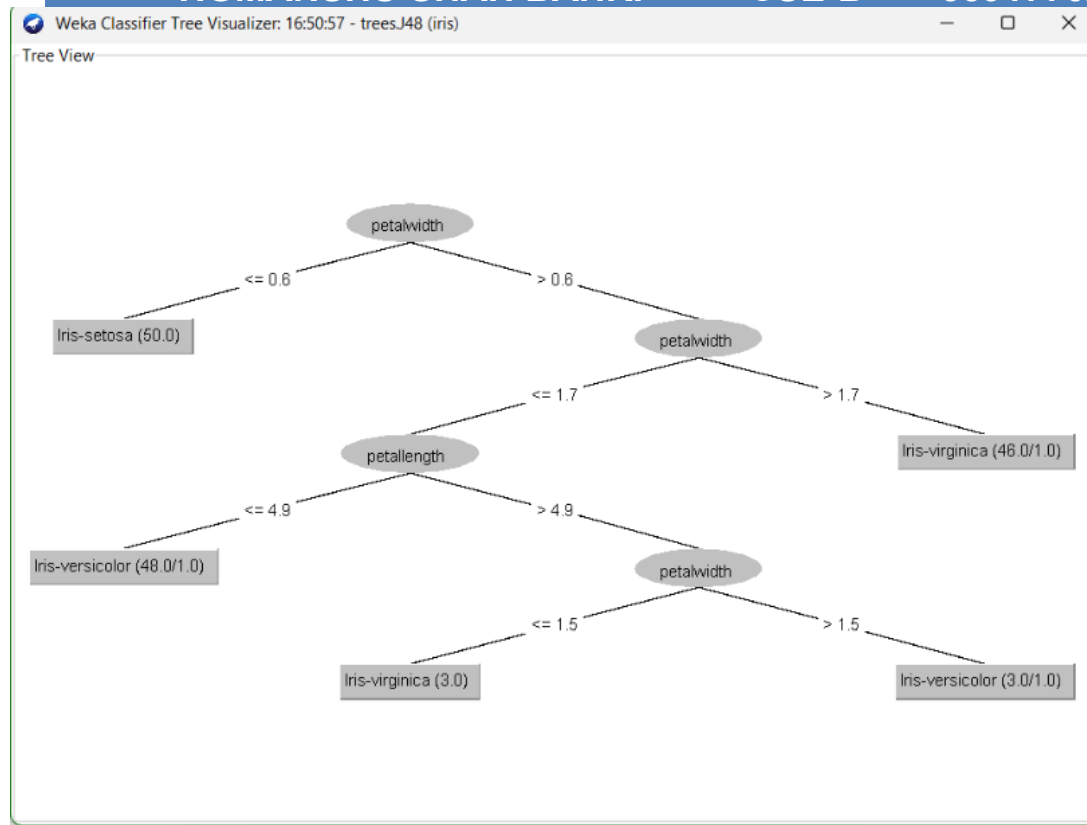
```

a b c <-- classified as
45 1 0 | a = Iris-setosa
0 47 3 | b = Iris-versicolor
0 2 46 | c = Iris-virginica

```

Status
OK

Log x 0



Learning Outcomes:

EXPERIMENT 6

Aim: Implementation of Clustering technique on ARFF files using WEKA.

Theory:

Clustering is **unsupervised learning** that groups similar data points.

WEKA supports:

- **K-Means**
- **EM**
- **Hierarchical Clustering**
Steps:

1. Load ARFF
2. Remove class label (unsupervised)
3. Apply K-Means
4. Visualize cluster assignments

Dataset:

@relation customers

@attribute age numeric

@attribute annual_income numeric

@attribute spending_score numeric

@data

25,30000,39

45,60000,70

20,15000,20

33,45000,55

52,80000,85

WEKA:

Preprocess **Classify** Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

Filter: Choose **None** Apply Stop

Current relation
Relation: iris
Instances: 150
Attributes: 5 Sum of weights: 150

Selected attribute
Name: sepal.length
Missing: 0 (0%) Distinct: 35 Type: Numeric
Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Class: class (Nom) Visualize All

Remove

Weka Explorer

Preprocess **Cluster** Associate Select attributes Visualize

Clusterer: Choose **SimpleKMeans** -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0

Cluster mode
☒ Use training set
☐ Supplied test set Set...
☐ Percentage split % 66
☐ Classes to clusters evaluation
 (Nom) class
☒ Store clusters for visualization

Ignore attributes
 Start Stop

Result list (right-click for options)

Clusterer output

weka.gui.GenericObjectEditor

weka.clusterers.SimpleKMeans

About
 Cluster data using the k means algorithm. More
 Capabilities

canopyMaxNumCanopiesToHoldInMemory 100
 canopyMinimumCanopyDensity 2.0
 canopyPeriodicPruningRate 10000
 canopyT1 -1.25
 canopyT2 -1.0
 debug False
 displayStdDevs False
 distanceFunction Choose **EudclideanDistance** -R
 doNotCheckCapabilities False
 dontReplaceMissingValues False
 fastDistanceCalc False
 initializationMethod Random
 maxIterations 500
 numClusters 3
 numExecutionSlots 1
 preserveInstancesOrder False
 reduceNumberOfDistanceCalcsViaCanopies False
 seed 10

Open... Save... OK Cancel

Status
OK

Log x 0

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Clusterer

Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -125 -t2 -10 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Cluster mode

☐ Use training set

☐ Supplied test set

☐ Percentage split

☒ Classes to clusters evaluation

(Nom) class

☒ Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

16:08:03 - SimpleKMeans

Clusterer output

Attribute	Full Data	0	1	2
	(150.0)	(61.0)	(50.0)	(39.0)
sepalength	5.8433	5.8885	5.006	6.8462
sepalwidth	3.054	2.7377	3.418	3.0821
petallength	3.7587	4.3567	1.464	5.7026
petalwidth	1.1987	1.418	0.244	2.0795

Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0	61 (41%)
1	50 (33%)
2	39 (26%)

Class attribute: class

Classes to Clusters:

0	1	2	<-- assigned to cluster
0	50	0	Iris-setosa
47	0	3	Iris-versicolor
14	0	36	Iris-virginica

Cluster 0 <-- Iris-versicolor

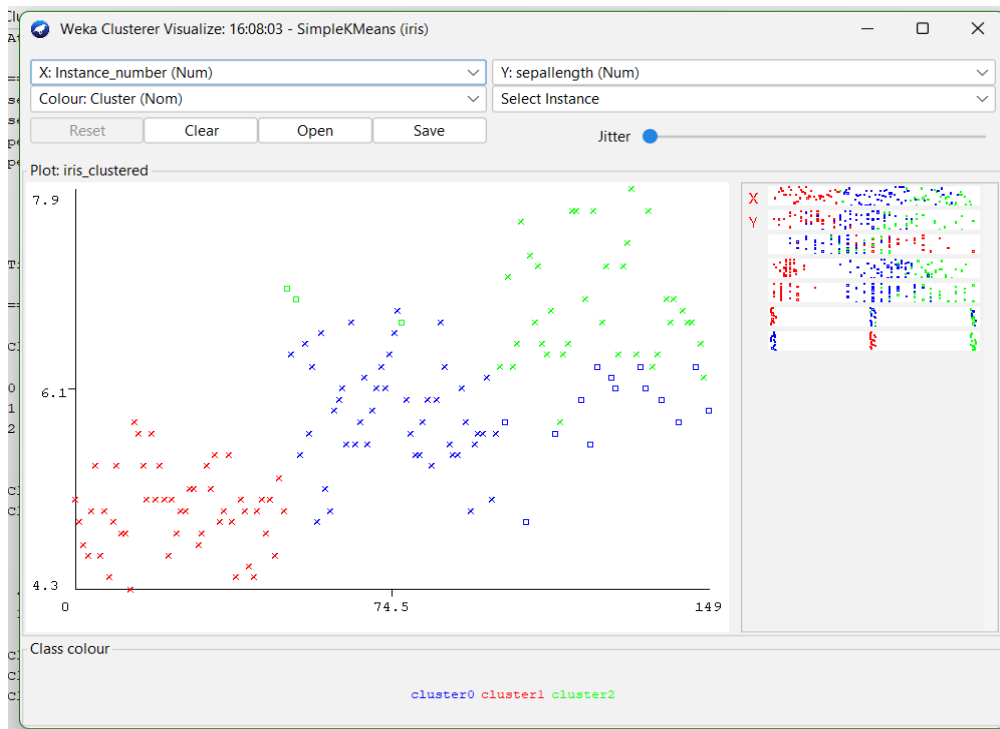
Cluster 1 <-- Iris-setosa

Cluster 2 <-- Iris-virginica

Incorrectly clustered instances : 17.0 11.3333 %

Status OK

Log



Learning Outcomes:

EXPERIMENT 7

Aim: Implementation of Association Rule technique on ARFF files using WEKA.

Theory:

Association mining finds **co-occurring patterns** like “bread → butter”.

Apriori algorithm steps:

1. Generate frequent itemsets using minimum support
2. Generate rules using minimum confidence
3. Output rules like:
 - {Milk} → {Bread}
 - {Laptop, Mouse} → {Bag}

Used in market basket analysis.

Dataset:

@relation market

@attribute items string

@data

"Milk,Bread"

"Bread,Butter"

"Milk,Butter,Biscuits"

"Biscuits,Chips"

"Laptop,Mouse"

WEKA:

Weka Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

Filter: Choose **None** Apply Stop

Current relation: supermarket
Instances: 4627

Attributes: 217
Sum of weights: 4627

Attributes: All None Invert Pattern

No.	Name
1	<input type="checkbox"/> department1
2	<input type="checkbox"/> department2
3	<input type="checkbox"/> department3
4	<input type="checkbox"/> department4
5	<input type="checkbox"/> department5
6	<input type="checkbox"/> department6
7	<input type="checkbox"/> department7
8	<input type="checkbox"/> department8
9	<input type="checkbox"/> department9
10	<input type="checkbox"/> grocery misc
11	<input type="checkbox"/> department11
12	<input type="checkbox"/> baby needs
13	<input type="checkbox"/> bread and cake
14	<input type="checkbox"/> baking needs
15	<input type="checkbox"/> coupons
16	<input type="checkbox"/> juice-sat-cord-ms
17	<input type="checkbox"/> tea
18	<input type="checkbox"/> biscuits
19	<input type="checkbox"/> canned fish-meat
20	<input type="checkbox"/> canned fruit
21	<input type="checkbox"/> canned vegetables
22	<input type="checkbox"/> breakfast food

Remove

Selected attribute:
Name: department1
Missing: 3580 (77%)
Distinct: 1
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	t	1047	1047

Class: total (Nom) Visualize All

Weka Explorer

Preprocess Classify Cluster **Associate** Select attributes Visualize

Associator: Choose **Apriori** -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

Start Stop

Result list (right-click for op...)

- 19:21:08 - FilteredAssociator
- 19:21:39 - FilteredAssociator
- 19:24:10 - Apriori

Associator 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:    supermarket
Instances:   4627
Attributes:  217
              [list of attributes omitted]

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

```

Apriori

=====

Minimum support: 0.15 (694 instances)

Minimum metric <confidence>: 0.9

Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44

Size of set of large itemsets L(2): 380

Size of set of large itemsets L(3): 910

Size of set of large itemsets L(4): 633

Size of set of large itemsets L(5): 105

Size of set of large itemsets L(6): 1

Best rules found:

1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723 <conf:(0.92)> lift:(1.27) lev:(0.03) [155] conv:(3.35)
2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696 <conf:(0.92)> lift:(1.27) lev:(0.03) [149] conv:(3.28)
3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705 <conf:(0.92)> lift:(1.27) lev:(0.03) [150] conv:(3.27)
4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746 <conf:(0.92)> lift:(1.27) lev:(0.03) [159] conv:(3.26)
5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779 <conf:(0.91)> lift:(1.27) lev:(0.04) [164] conv:(3.15)
6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725 <conf:(0.91)> lift:(1.26) lev:(0.03) [151] conv:(3.06)
7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701 <conf:(0.91)> lift:(1.26) lev:(0.03) [145] conv:(3.01)
8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866 <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(3)
9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757 <conf:(0.91)> lift:(1.26) lev:(0.03) [156] conv:(3)
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877 <conf:(0.91)> lift:(1.26) lev:(0.04) [179] conv:(2.92)

EXPERIMENT 8

Aim: Implementation of Visualization technique on ARFF files using WEKA.

Theory:

WEKA provides multiple visualization tools:

- Scatter Plots
- Attribute Histograms
- Class Color Visualization
- Cluster Visualization
- Decision Tree Visualizer

Visualization helps understand data distributions, class separability, and find outliers.

Dataset:

@relation students

@attribute hours_studied numeric

@attribute attendance numeric

@attribute marks numeric

@data

2,60,45

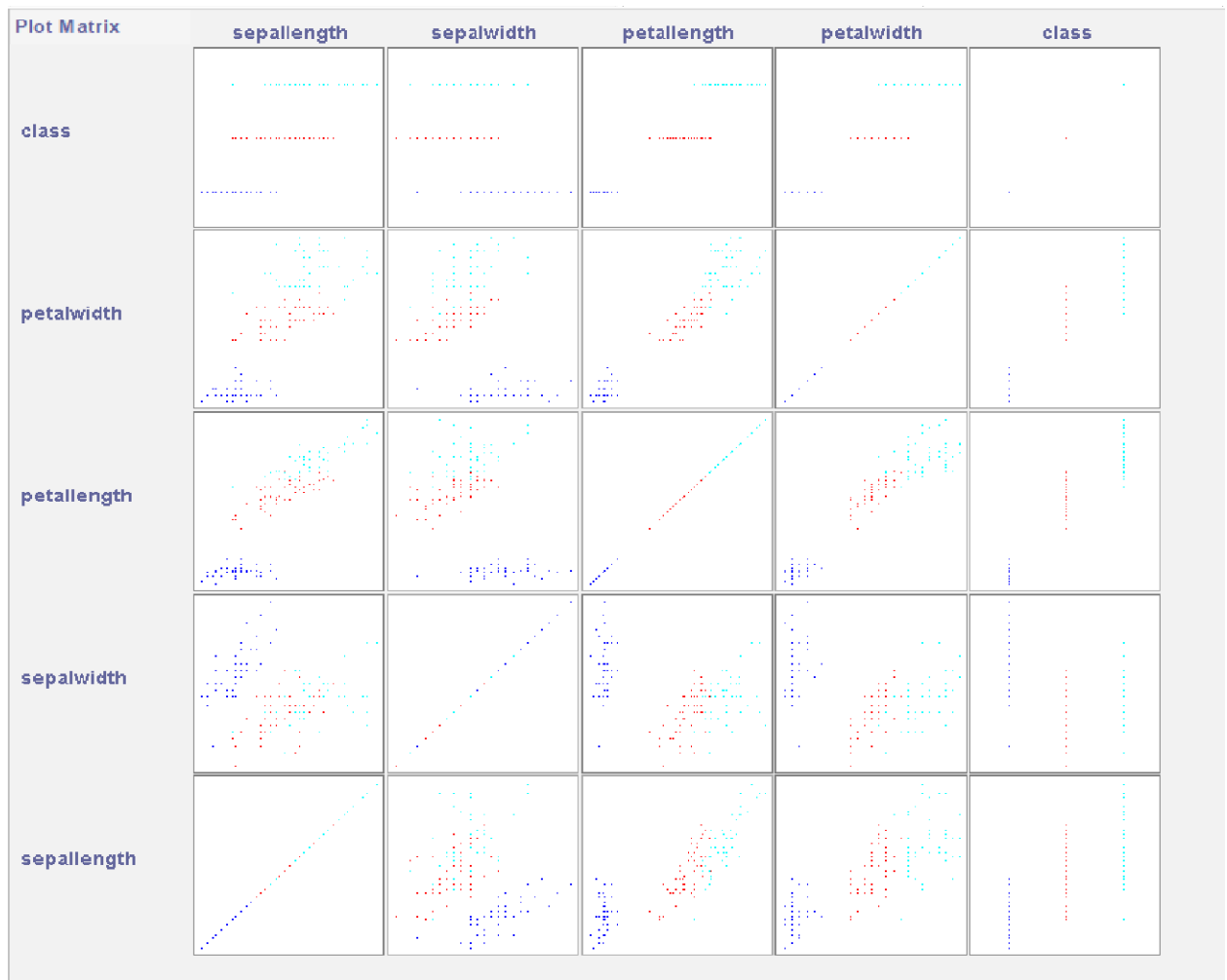
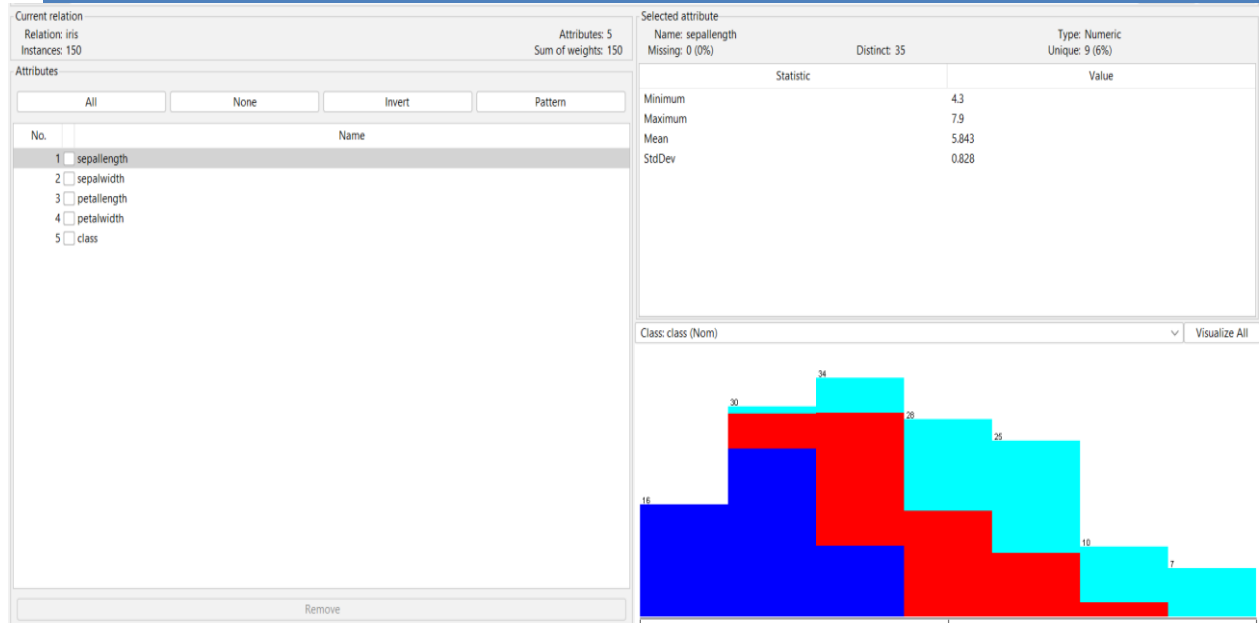
5,75,65

8,85,80

3,70,50

6,90,78

WEKA:



Learning Outcomes: