योगः 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)

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

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

S.No	EXP.	Allotment Date	Submission Date	Marks			Remark	Updated Marks	Faculty Signature
				Laboratory Assessment (15 Marks)	Class Participation (5 Marks)	Viva (5 Marks)			

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

S.No	EXP.	Allotment Date	Submission Date	Marks			Remark	Updated Marks	Faculty Signature
				Laboratory Assessment (15 Marks)	Class Participation (5 Marks)	Viva (5 Marks)			

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())
```

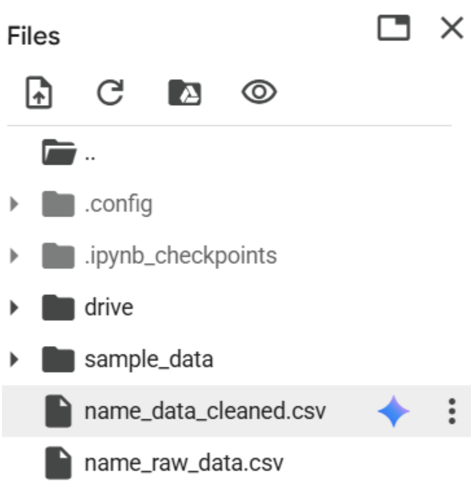
```
           Raw_name
0      RaHuL ShArMa
1     rahul   SHARma
2    RAHUL sharma
3      PrIyA KaPoOr
4  priya    KAPOOR
```

```
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:



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)
   ---------------------------------------- 159.9/159.9 kB 3.2 MB/s eta 0:00:00
Downloading rapidfuzz-3.14.1-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl (3.2 MB)
   ---------------------------------------- 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)
```

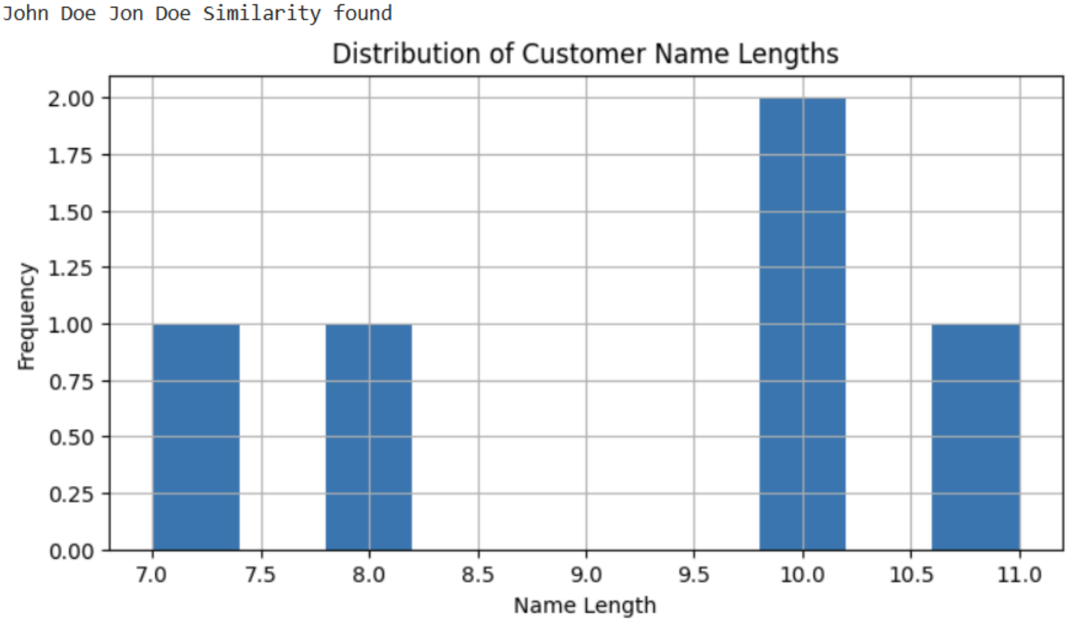
```
   Customer_ID        Name              Email       Phone          Address
0          101    John Doe   john@example.com  1234567890       123 Elm St
1          102  Jane Smith   jane@example.com  2345678901       456 Oak St
2          103    john doe   john@example.com  1234567890   123 Elm Street
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  alice@example.com  3456789012   Ally Brown
```

```python
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:

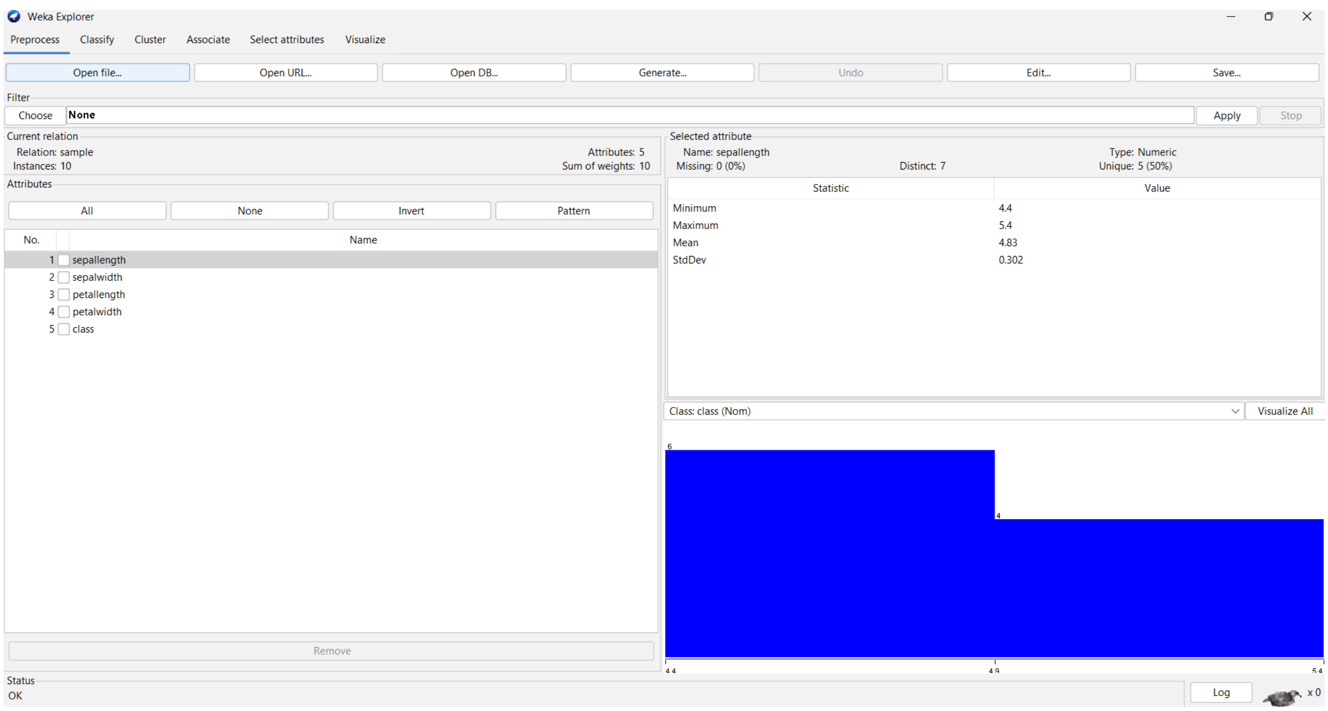
- Preprocessing (cleaning, normalization)
- Classification (J48, Naive Bayes, SVM)
- Clustering (K-Means, EM)
- Association (Apriori)
- Visualization (scatter plots, histograms)
  Supports **ARFF, CSV** formats.
  Used widely for academic ML experiments.

Dataset:

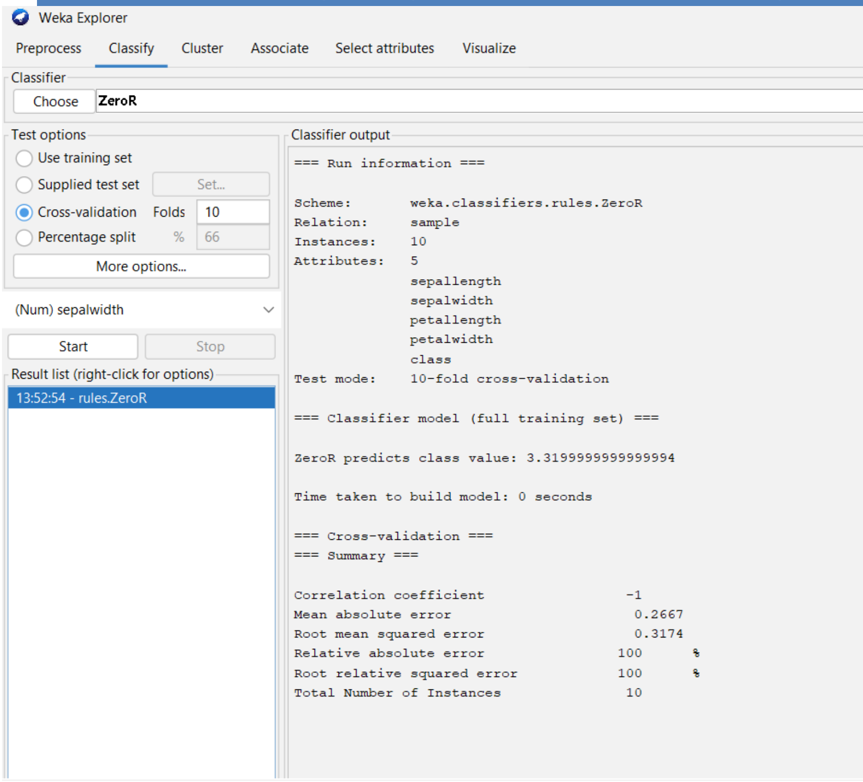
```
@relation weather

@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
```

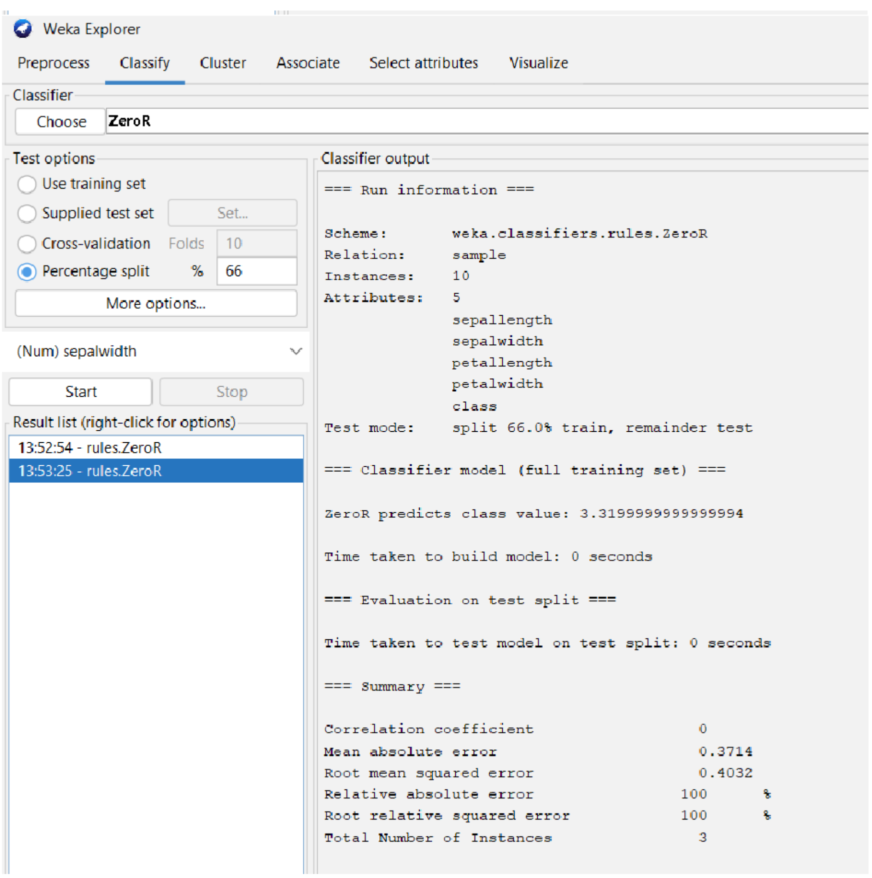
TIGMANSHU SHAH BAHRI CSE-B 35917702722

WEKA:

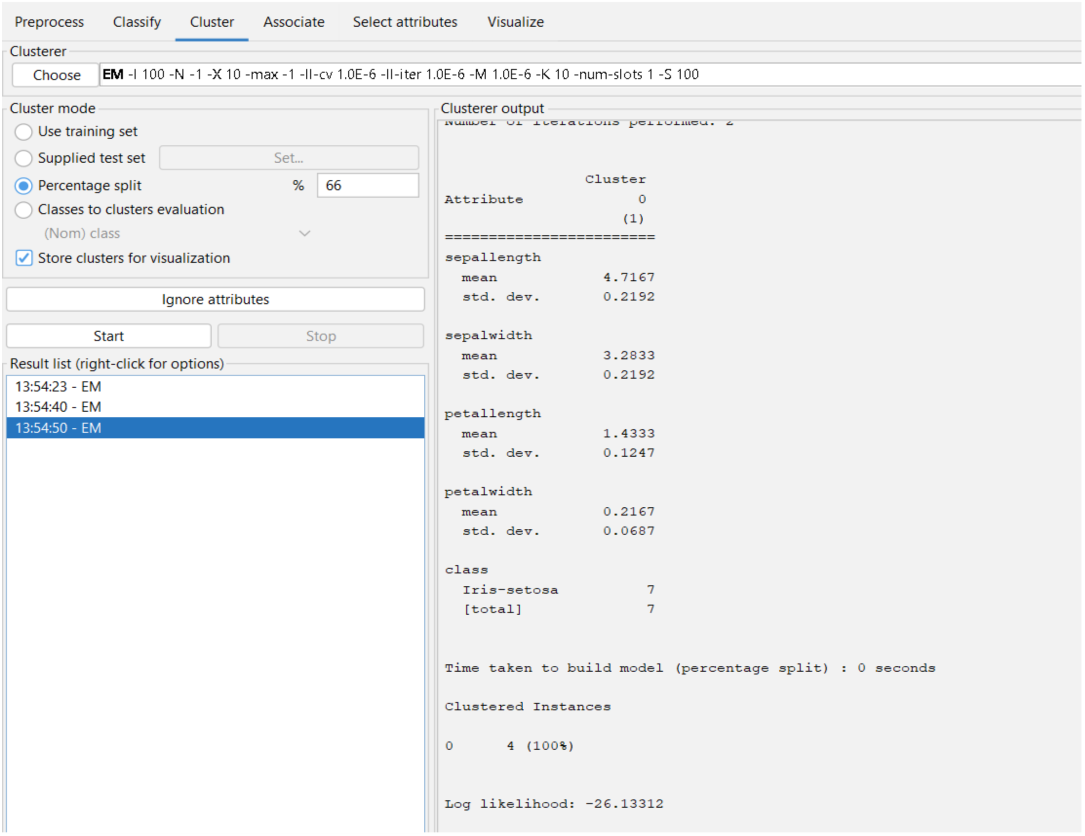


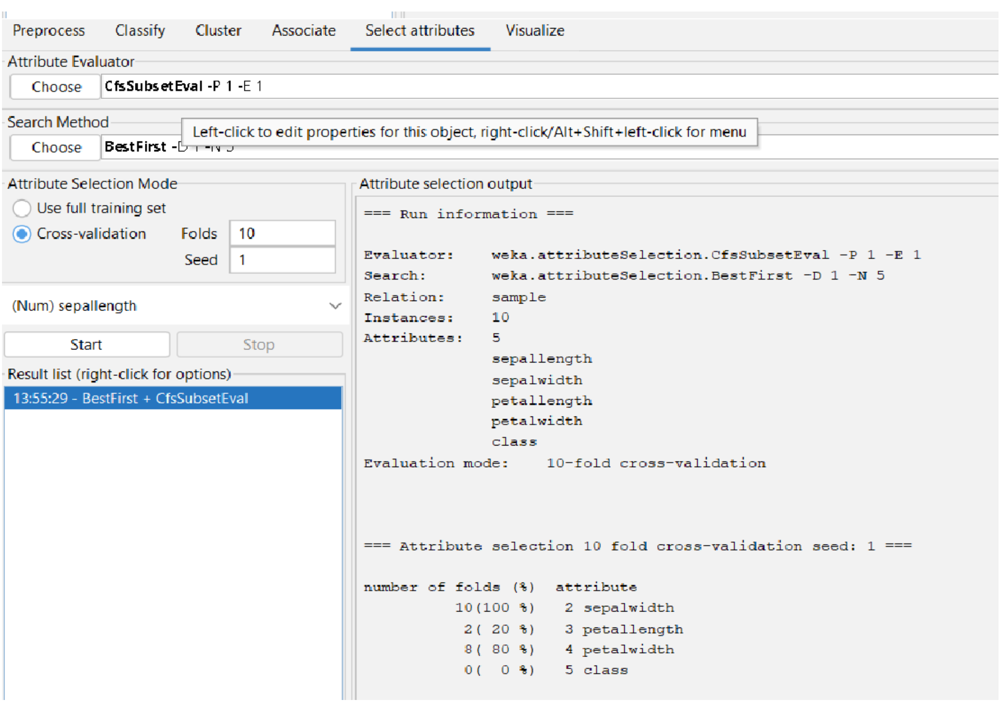
TIGMANSHU SHAH BAHRI CSE-B 35917702722

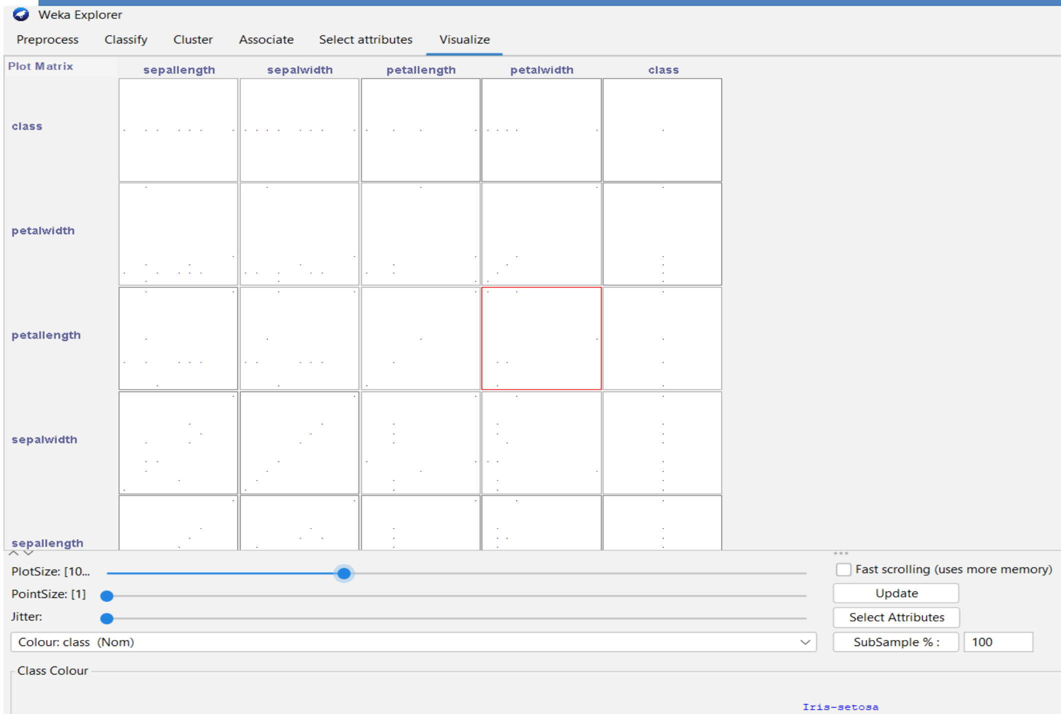




TIGMANSHU SHAH BAHRI CSE-B 35917702722







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:

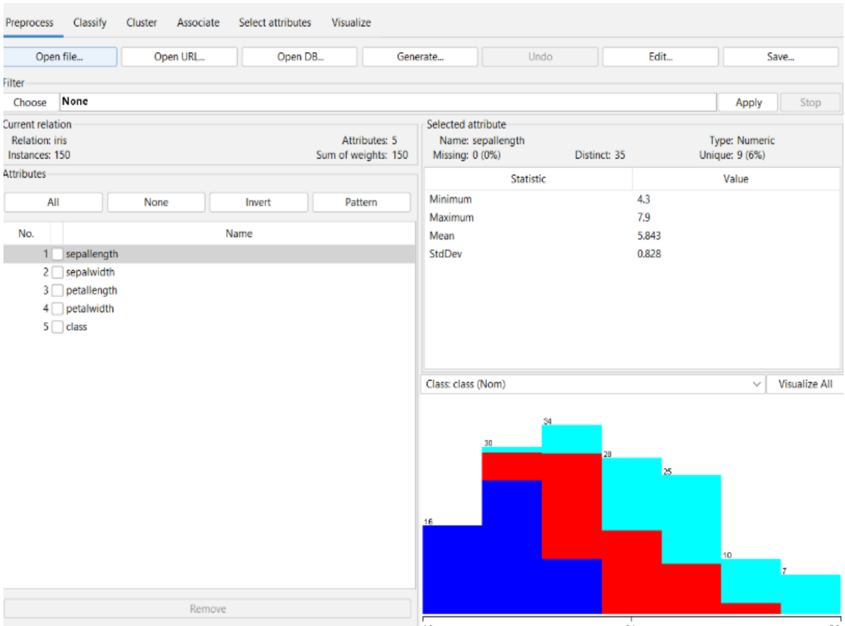
1. Load ARFF
2. Preprocess
3. Choose classifier
4. Train/Test split
5. Evaluate accuracy, precision, confusion matrix

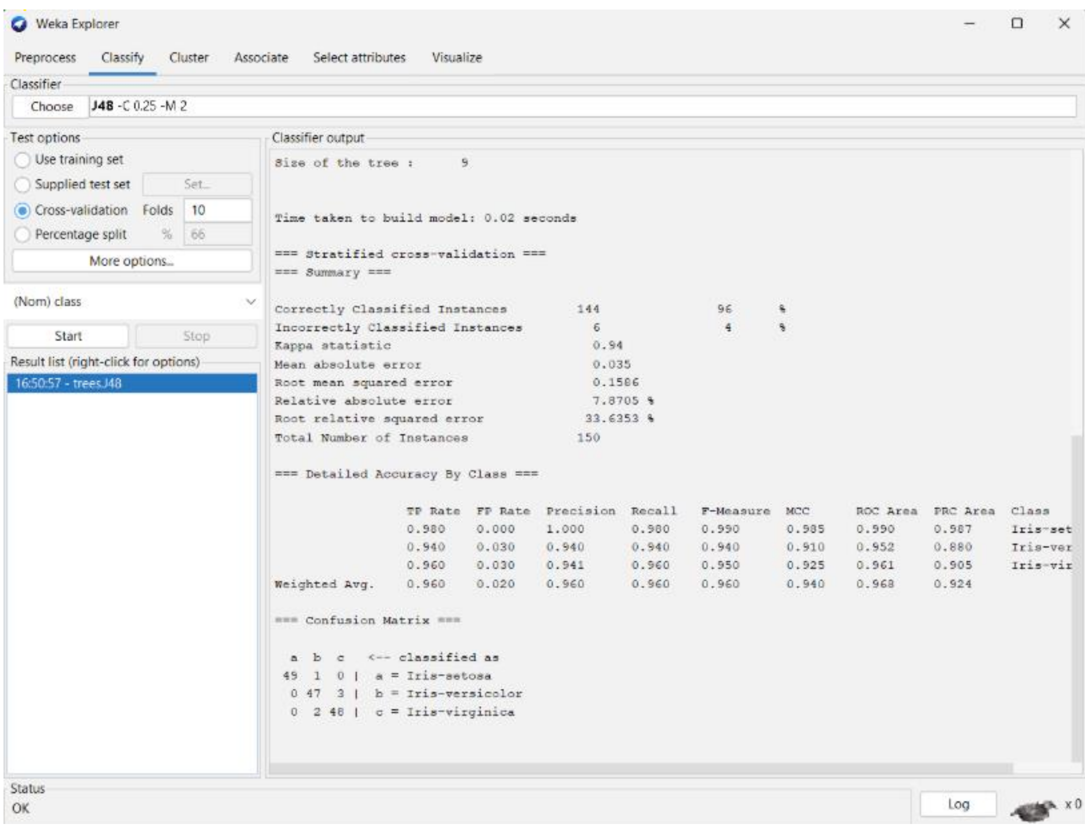
Dataset:

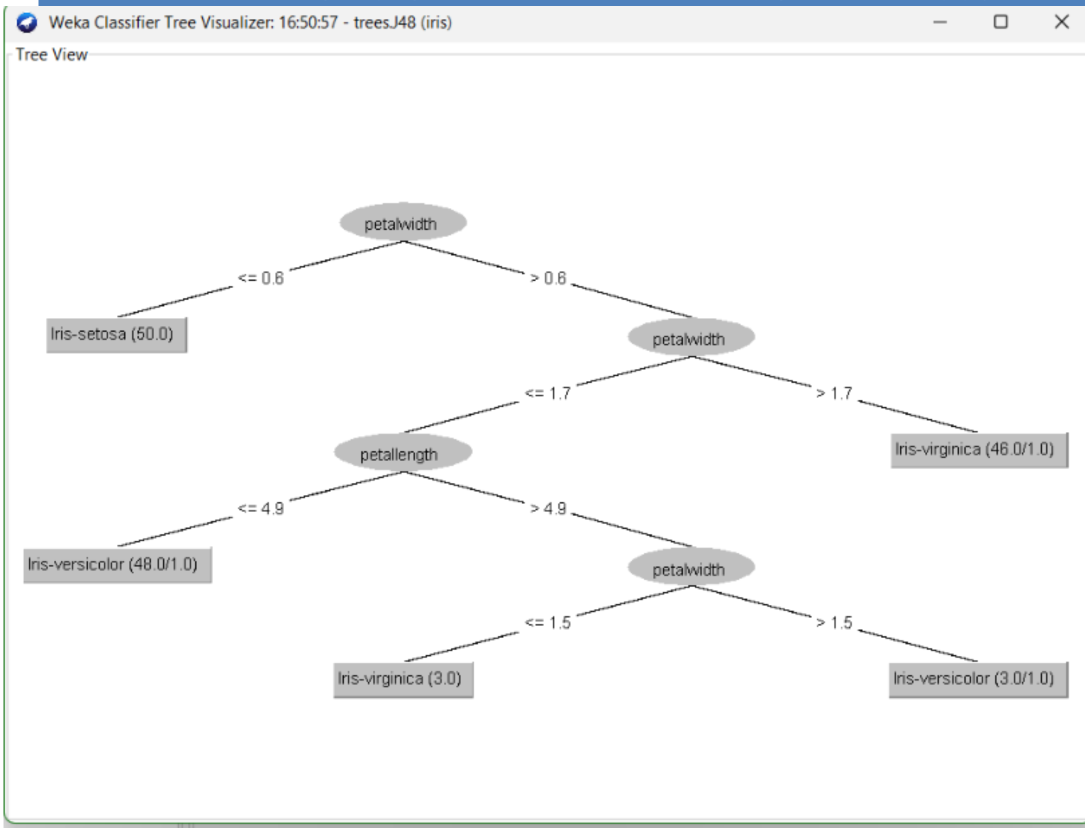
```
@relation iris

@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
```

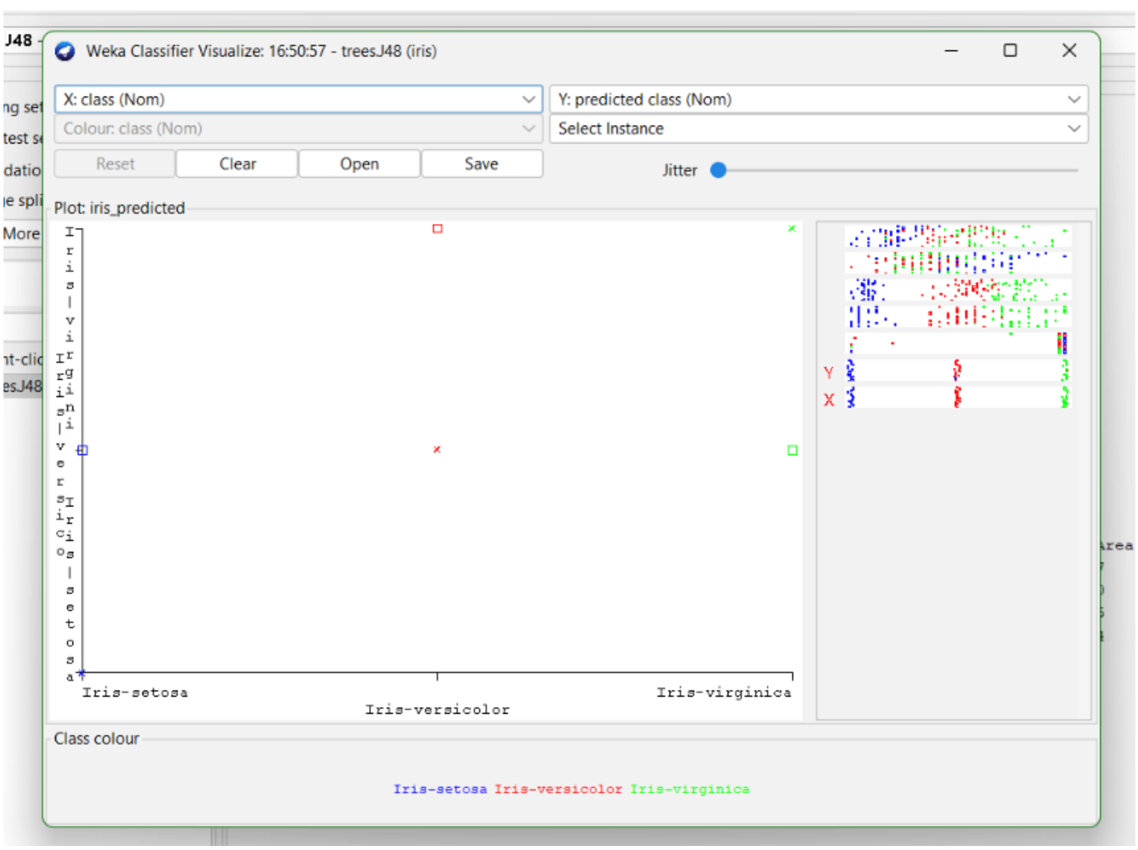
TIGMANSHU SHAH BAHRI CSE-B 35917702722

WEKA:









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:

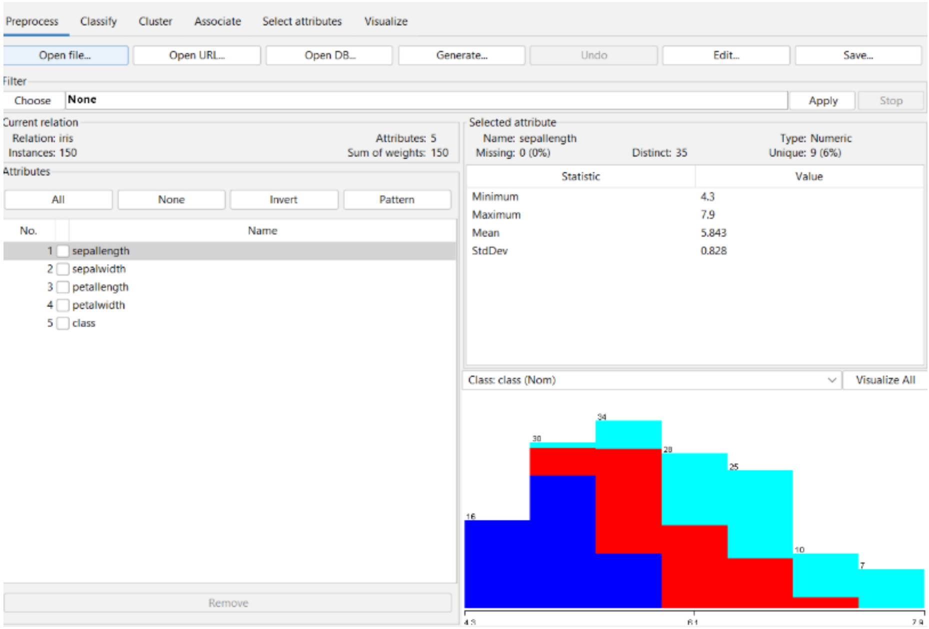
- **K-Means**
- **EM**
- **Hierarchical Clustering**
  Steps:

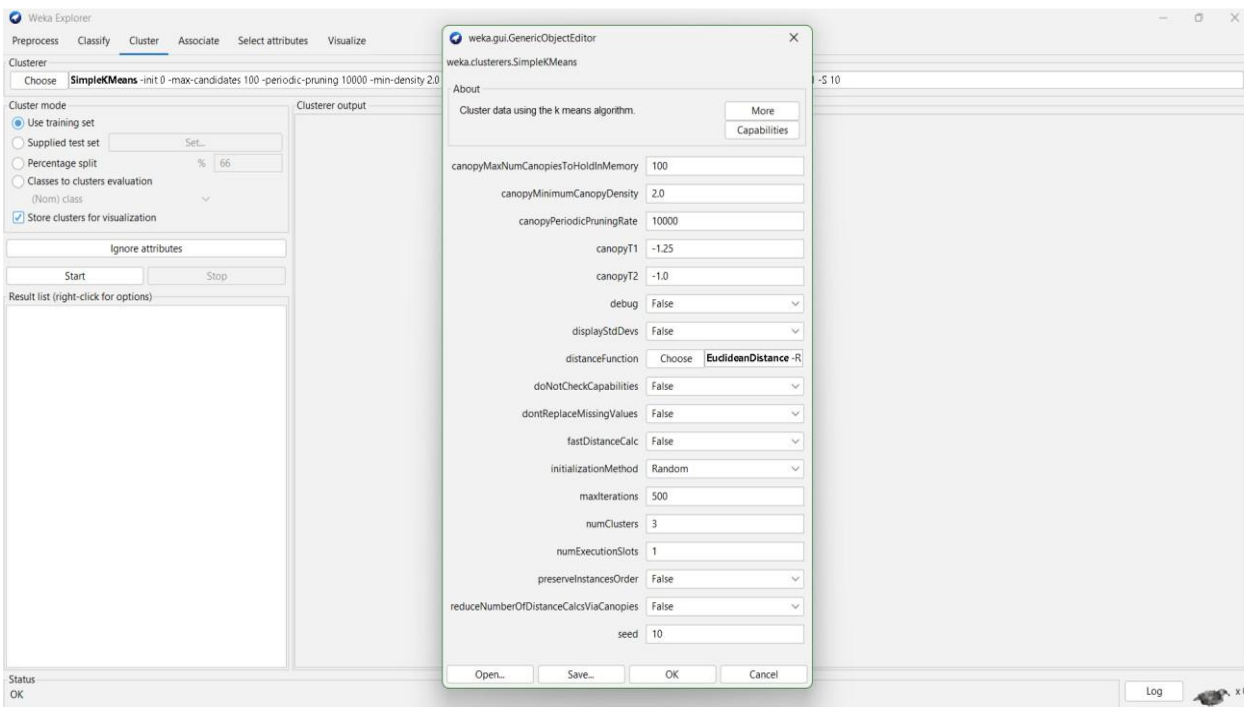
1. Load ARFF
2. Remove class label (unsupervised)
3. Apply K-Means
4. Visualize cluster assignments

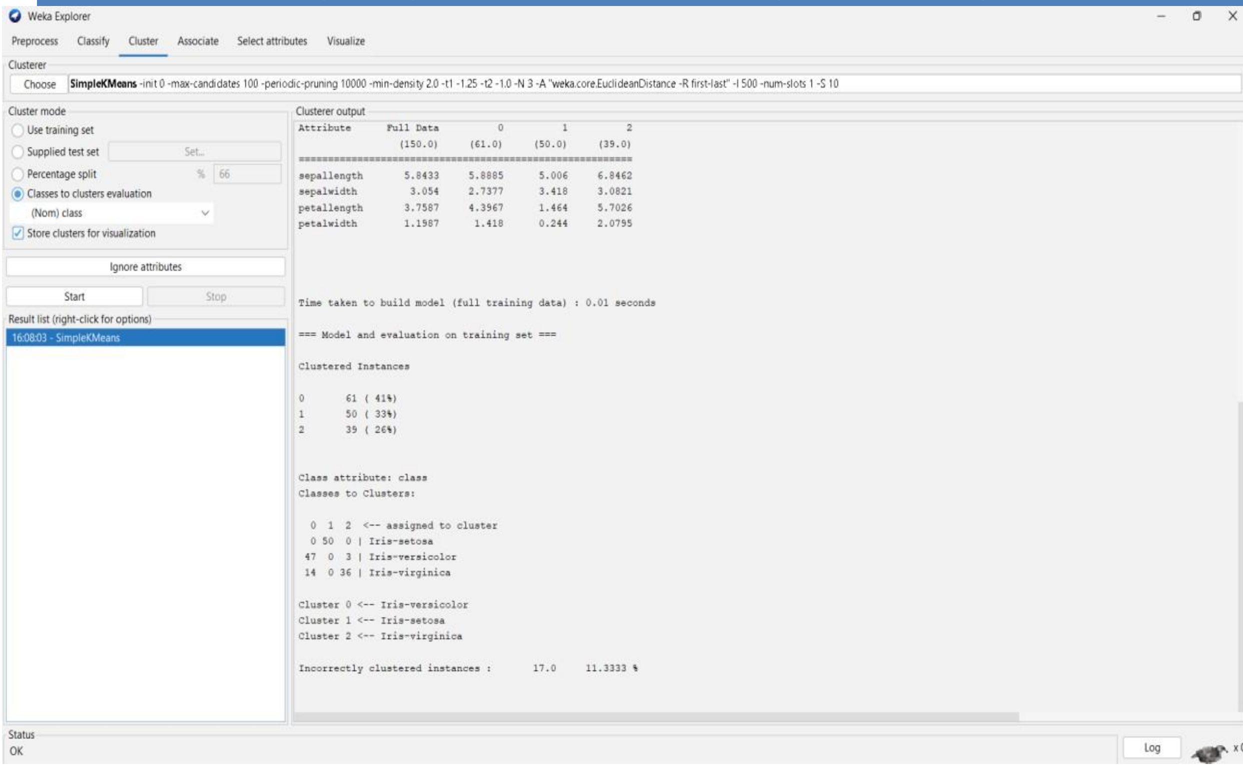
Dataset:

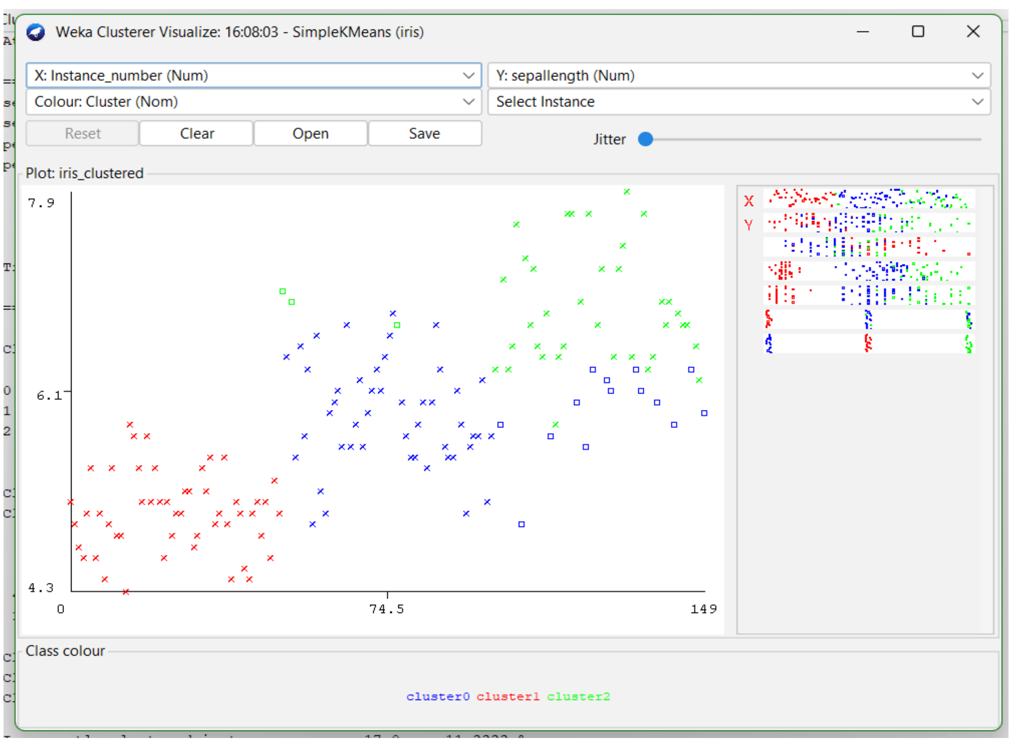
```
@relation customers

@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: sepallength
Missing: 0 (0%)
Distinct: 35
Type: Numeric
Unique: 9 (6%)
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
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
Clusterer
Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0
-S 10
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
Status
OK
Log
x 0
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 EuclideanDistance -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





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}`

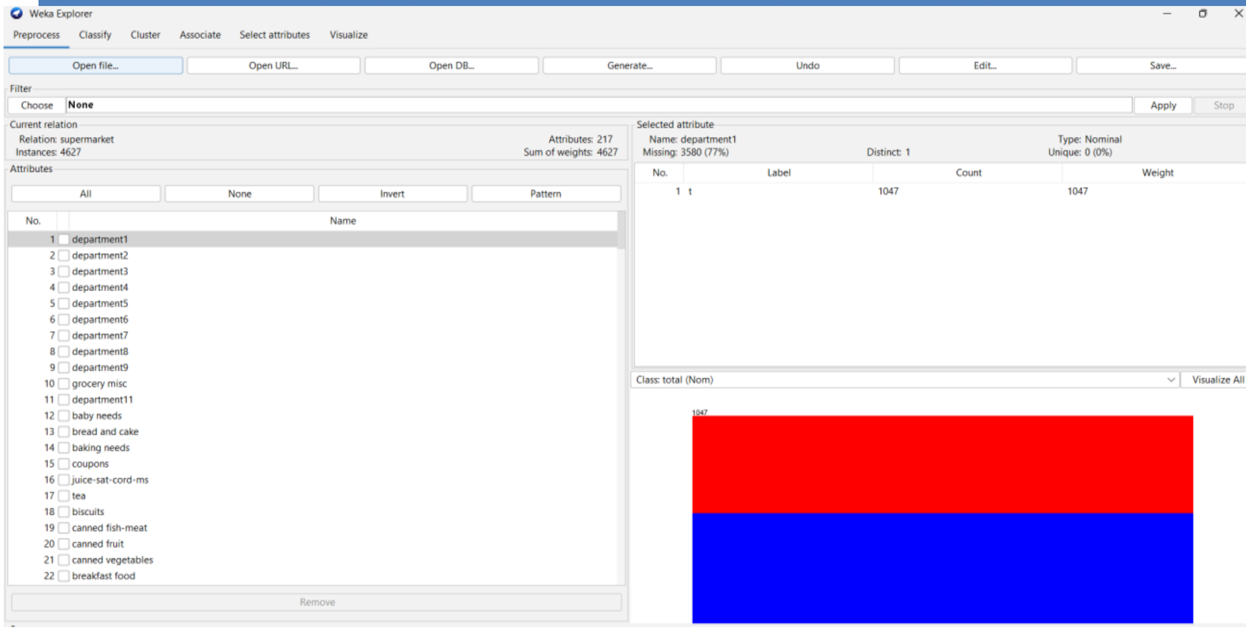
Used in market basket analysis.

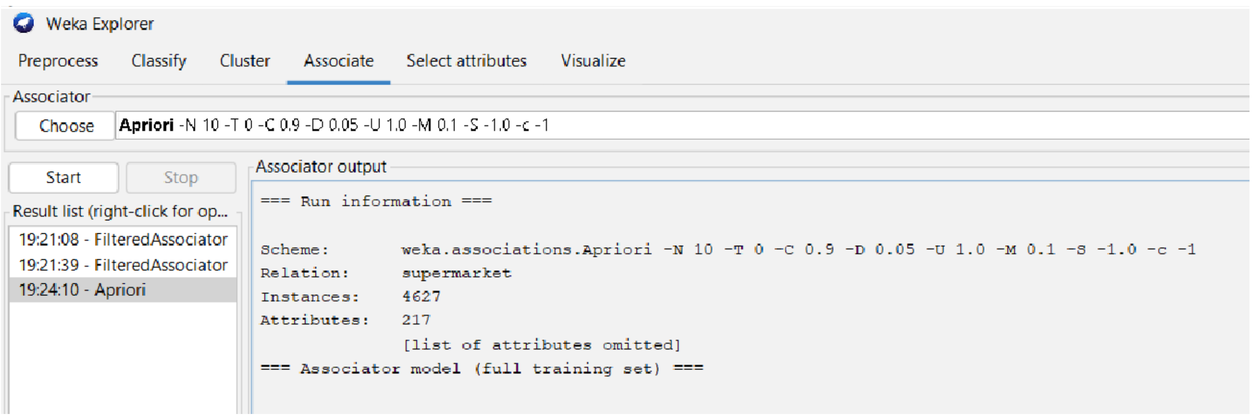
Dataset:

```
@relation market

@attribute items string

@data
"Milk,Bread"
"Bread,Butter"
"Milk,Butter,Biscuits"
"Biscuits,Chips"
"Laptop,Mouse"
```

WEKA:

TIGMANSHU SHAH BAHRI CSE-B 35917702722






```
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:

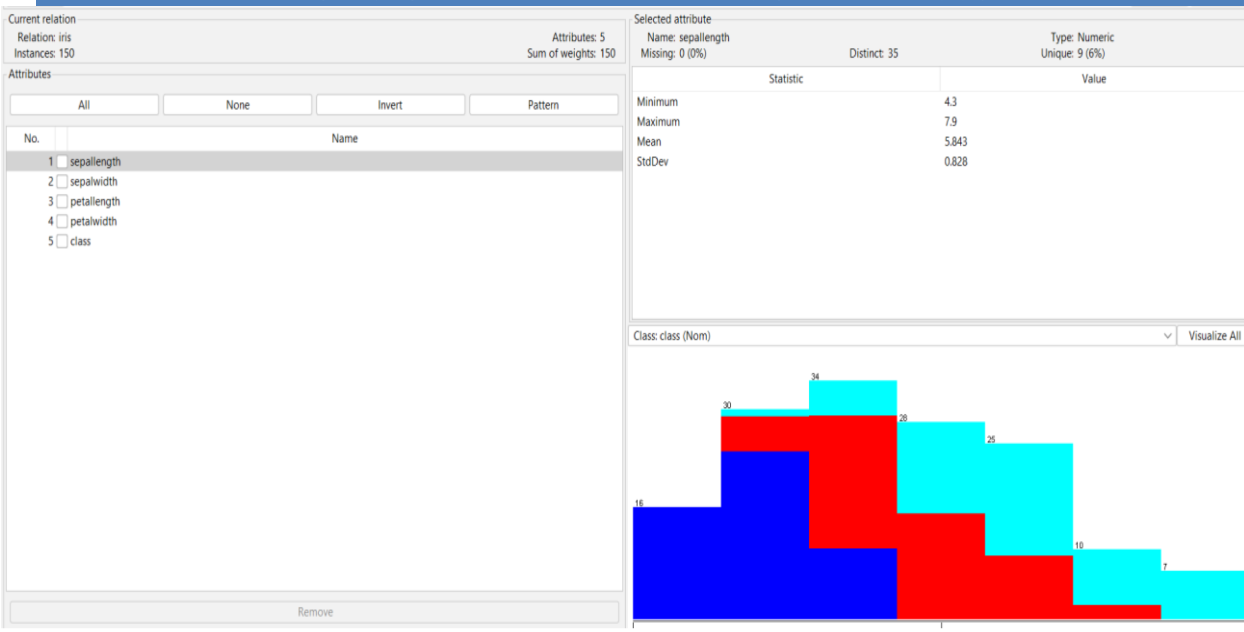
- Scatter Plots
- Attribute Histograms
- Class Color Visualization
- Cluster Visualization
- Decision Tree Visualizer

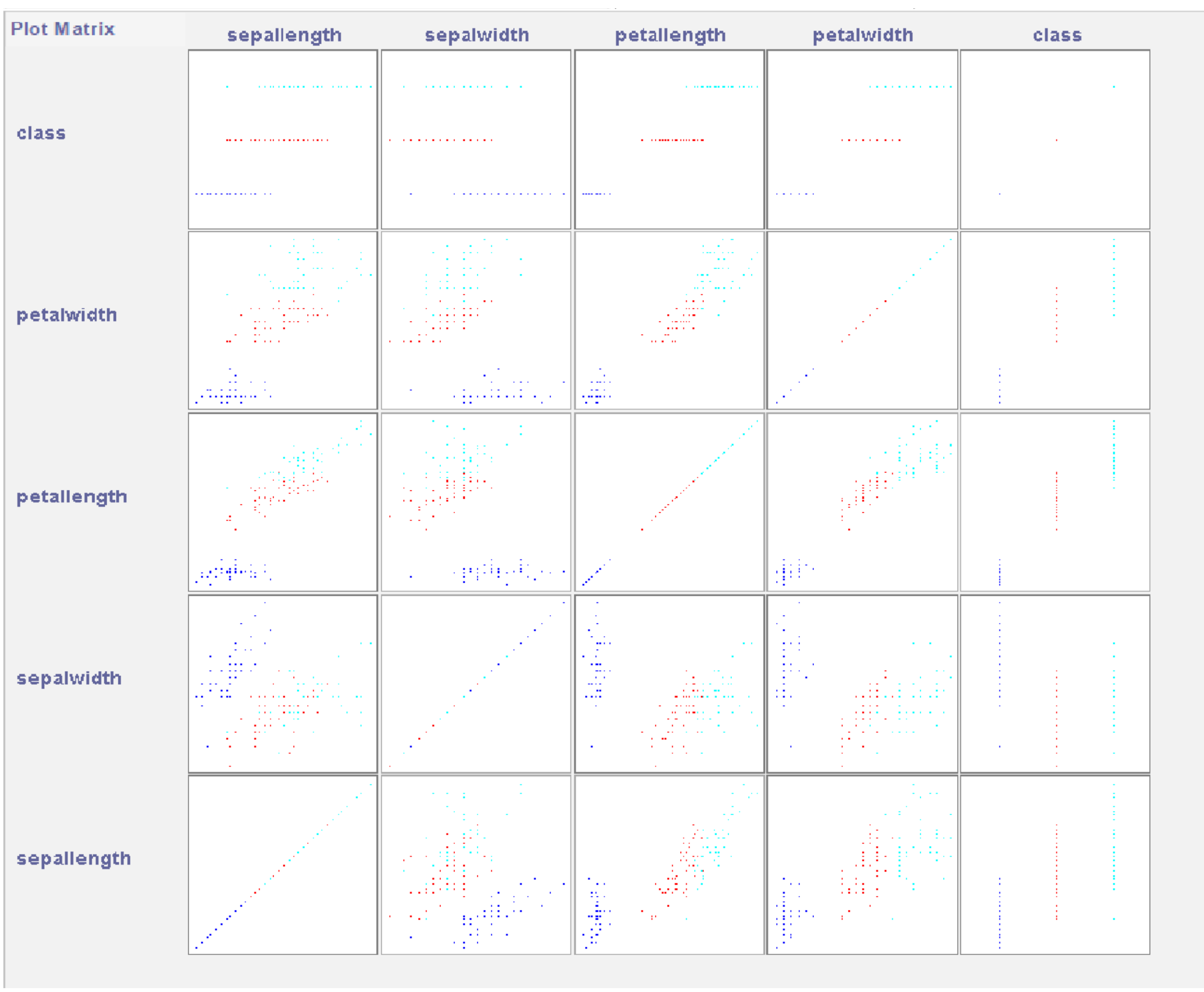
Visualization helps understand data distributions, class separability, and find outliers.

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: