

Name:- Divyaprada G

USN:- 1SI24AD401

DATA MINING AND VISUALIZATION LABORATORY-4

- 4. Set up SQL database and insert sample data consisting of customer data. Integrate SQL database with Weka.**
- a). Perform data preprocessing, classification, clustering tasks on customer data.**
 - b). Apply filters and interpret the results**
 - c). Connect WEKA to a relational database using JDBC.**
 - d). Retrieve customer data directly using SQL queries.**
 - e). Load data into WEKA's Explorer interface.**
 - f). Apply classification algorithms (e.g., J48, Naive Bayes) to predict customer spending behavior.**
 - g). Use clustering (e.g., k-means) for market segmentation.**
 - h). Generate evaluation metrics (accuracy, precision, recall)**
 - i). Optionally automate data refresh using scripts or the WEKA KnowledgeFlow interface. Validate and analyse the result.**

Additional: Derive Fact Table, Star schema, Snowflake schema and do the comparison

Step-1:- Download and Install Java 8 binaries from Eclipse Adoptium:

https://adoptium.net/temurin/releases/?version=8&utm_source

open CMD :-

➤ Java – version

Step 2:- Download MYSQL:

<https://dev.mysql.com/downloads/file/?id=546163>

Substeps

- ↳ Open MySQL Workbench
 - ↳ Create database customerdb;
 - ↳ Create table customers;
 - ↳ Insert records to customers table.

Step 3:- Download MYSQL jar file (if don't have jar file):-

<https://dev.mysql.com/downloads/connector/j/>

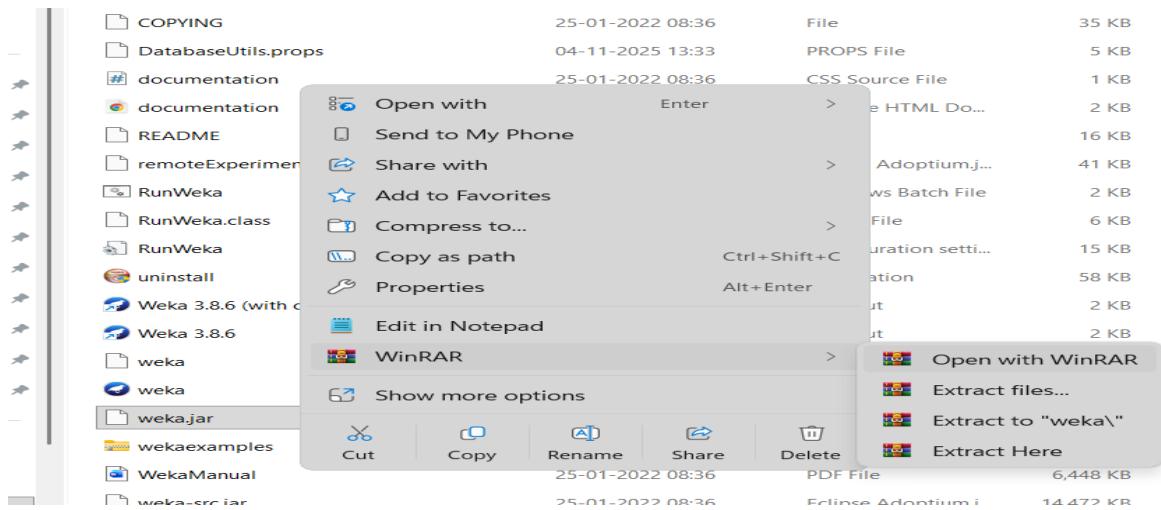
Extract Folder

- ↳ Copy MySQL “jar” file
 - ↳ Paste into inside WEKA folder

Step 4:-

File Explorer

- ↳ This PC
 - ↳ Local Disk (C:) or (D:)
 - ↳ Program Files
 - ↳ weka-3-8-6
 - Open
 - ↳ weka.java in “WinRAR”
 - ↳ weka
 - ↳ experiments
 - ↳ DatabaseUtils.props.mysql



Name	Size	Packed	Type	Modified	CRC32
..			File folder		
com	3,625,605	1,516,429	File folder	25-01-2022 16:...	
java_cup	59,093	26,887	File folder	25-01-2022 16:...	
javax	263,022	120,701	File folder	25-01-2022 16:...	
META-INF	138	124	File folder	25-01-2022 16:...	
org	1,993,636	892,305	File folder	25-01-2022 16:...	
weka	16,567,444	9,006,280	File folder	25-01-2022 16:...	
arpack_combined.jar	1,204,349	1,104,937	Eclipse Adoptium.j...	25-01-2022 16:...	6308395F
core.jar	164,422	159,141	Eclipse Adoptium.j...	25-01-2022 16:...	C6ACEECD
mtj.jar	271,988	246,989	Eclipse Adoptium.j...	25-01-2022 16:...	A13687BC

attributeSelection	268,191	130,919	File folder	25-01-2022 16:...
classifiers	2,626,182	1,287,925	File folder	25-01-2022 16:...
clusterers	258,736	127,914	File folder	25-01-2022 16:...
core	2,871,128	1,331,606	File folder	25-01-2022 16:...
datagenerators	171,651	81,349	File folder	25-01-2022 16:...
estimators	171,105	90,236	File folder	25-01-2022 16:...
experiment	455,656	221,052	File folder	25-01-2022 16:...
filters	859,181	419,595	File folder	25-01-2022 16:...
gui	7,988,497	4,885,874	File folder	25-01-2022 16:...
knowledgeflow	688,679	331,530	File folder	25-01-2022 16:...
PluginManager.props	511	189	PROPS File	25-01-2022 16:... BFB02033
Run\$SchemeType.class	1,845	1,027	CLASS File	25-01-2022 16:... B4B53C2D
Run.class	8,157	4,370	CLASS File	25-01-2022 16:... 6AEA4256

AveragingResultProducer.class	16,514	7,895	CLASS File	25-01-2022 16...	DDC1012D
ClassifierSplitEvaluator.class	24,347	11,366	CLASS File	25-01-2022 16...	0AA1B7F1
Compute.class	352	217	CLASS File	25-01-2022 16...	A50EE85E
CostSensitiveClassifierSplitEvaluator.class	12,078	5,788	CLASS File	25-01-2022 16...	D793DE29
CrossValidationResultProducer.class	14,251	6,728	CLASS File	25-01-2022 16...	02F251CC
CrossValidationSplitResultProducer.class	5,056	2,595	CLASS File	25-01-2022 16...	37E22EF3
CSVResultListener.class	6,128	3,100	CLASS File	25-01-2022 16...	35CA8989
DatabaseResultListener.class	7,718	3,885	CLASS File	25-01-2022 16...	E59EDB84
DatabaseResultProducer.class	9,615	4,353	CLASS File	25-01-2022 16...	0A5960AA
DatabaseUtils.class	25,202	12,442	CLASS File	25-01-2022 16...	4C733614
DatabaseUtils.props	2,345	1,142	PROPS File	25-01-2022 16...	8163D11D
DatabaseUtils.props.hsql	3,296	1,499	HSQL File	25-01-2022 16...	4984E03F
DatabaseUtils.props.msaccess	4,021	1,791	MSACCESS File	25-01-2022 16...	14597964
DatabaseUtils.props.mssqlserver	3,714	1,697	MSSQLSERVER File	25-01-2022 16...	9A6A4E43
DatabaseUtils.props.mssqlserver2005	3,781	1,702	MSSQLSERVER200...	25-01-2022 16...	469968F2
DatabaseUtils.props.mysql	4,364	1,865	MYSQL File	25-01-2022 16...	B2B9FC18
DatabaseUtils.props.odbc	1,639	844	ODBC File	25-01-2022 16...	7E3800F7
DatabaseUtils.props.oracle	2,817	1,368	ORACLE File	25-01-2022 16...	850E9C50
DatabaseUtils.props.postgresql	8,569	3,120	POSTGRESQL File	25-01-2022 16...	E6264A0D

Selected 1 file, 4,364 bytes Total 1 folder, 60 files, 455,656 bytes

DMV weka 25-01-2022 08:36 ICO File 351 KB

Name	Date modified	Type	Size
changelogs	19-09-2025 09:11	File folder	
data	19-09-2025 09:11	File folder	
doc	19-09-2025 09:11	File folder	
jre	19-09-2025 09:11	File folder	
COPYING	25-01-2022 08:36	File	35 KB
DatabaseUtils.props	04-11-2025 13:33	PROPS File	5 KB
documentation	25-01-2022 08:36	CSS Source File	1 KB
documentation	25-01-2022 08:36	Chrome HTML Do...	2 KB
README	25-01-2022 08:36	File	16 KB
remoteExperimentServer.jar	25-01-2022 08:36	Eclipse Adoptium.j...	41 KB
RunWeka	25-01-2022 08:37	Windows Batch File	2 KB
RunWeka.class	25-01-2022 08:37	CLASS File	6 KB
RunWeka	25-01-2022 08:37	Configuration setti...	15 KB
Uninstall	19-09-2025 09:11	Application	58 KB

Rename the file –

Remove "mysql" from the end of the file name.

Example:

- Original: DatabaseUtils.props.mysql
- New name: DatabaseUtils.props

Then **save** the file.

Open the DatabaseUtils.props file –

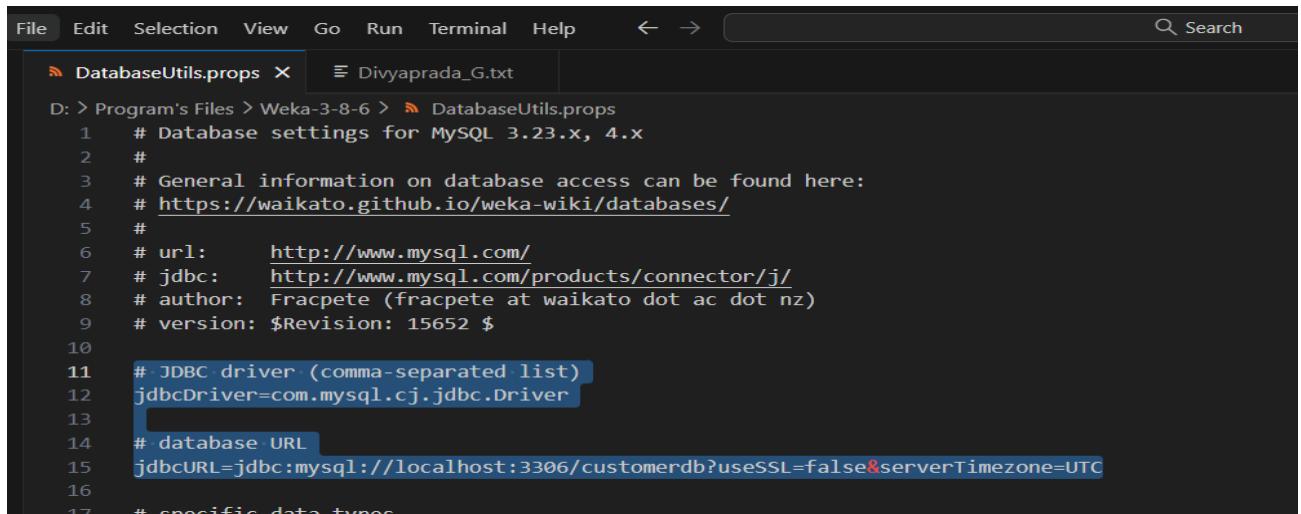
Edit the following entries:

- **Database URL:** Set your MySQL connection string.

Example:- jdbcURL=jdbc:mysql://localhost:3306/your_database_name

- **Database Driver:** Set your MySQL JDBC driver.

Example: jdbcDriver=com.mysql.cj.jdbc.Driver



A screenshot of a code editor window titled "DatabaseUtils.props". The window has a menu bar with "File", "Edit", "Selection", "View", "Go", "Run", "Terminal", and "Help". A search bar is at the top right. The main area contains the following text:

```
D: > Program's Files > Weka-3-8-6 > DatabaseUtils.props
1 # Database settings for MySQL 3.23.x, 4.x
2 #
3 # General information on database access can be found here:
4 # https://waikato.github.io/weka-wiki/databases/
5 #
6 # url:      http://www.mysql.com/
7 # jdbc:     http://www.mysql.com/products/connector/j/
8 # author:   Fracpete (fracpete at waikato dot ac dot nz)
9 # version: $Revision: 15652 $
10 #
11 # JDBC driver (comma-separated list)
12 jdbcDriver=com.mysql.cj.jdbc.Driver
13 #
14 # database URL
15 jdbcURL=jdbc:mysql://localhost:3306/customerdb?useSSL=false&serverTimezone=UTC
16 #
17 # specific data types
```

- Remove tags in front of the data type lines, then save the file.

```
# database URL
jdbcURL=jdbc:mysql://localhost:3306/customerdb?useSSL=false&serverTimezone=UTC

# specific data types
STRING, getString() = 0;          --> nominal
BOOLEAN, getBoolean() = 1;         --> nominal
DOUBLE, getDouble() = 2;           --> numeric
BYTE, getByte() = 3;              --> numeric
SHORT, getShort() = 4;             --> numeric
INT, getInt() = 5;                --> numeric
INTEGER, getInt() = 5;             --> numeric
BIGINT, getLong() = 6;             --> numeric
FLOAT, getFloat() = 7;             --> numeric
DECIMAL, getDouble() = 2;           --> numeric
NUMERIC, getDouble() = 2;           --> numeric
DATE, getDate() = 8;               --> date
TEXT, getString() = 9;              --> string
TIME, getTime() = 10;              --> date
TIMESTAMP, getTimestamp() = 11;    --> date

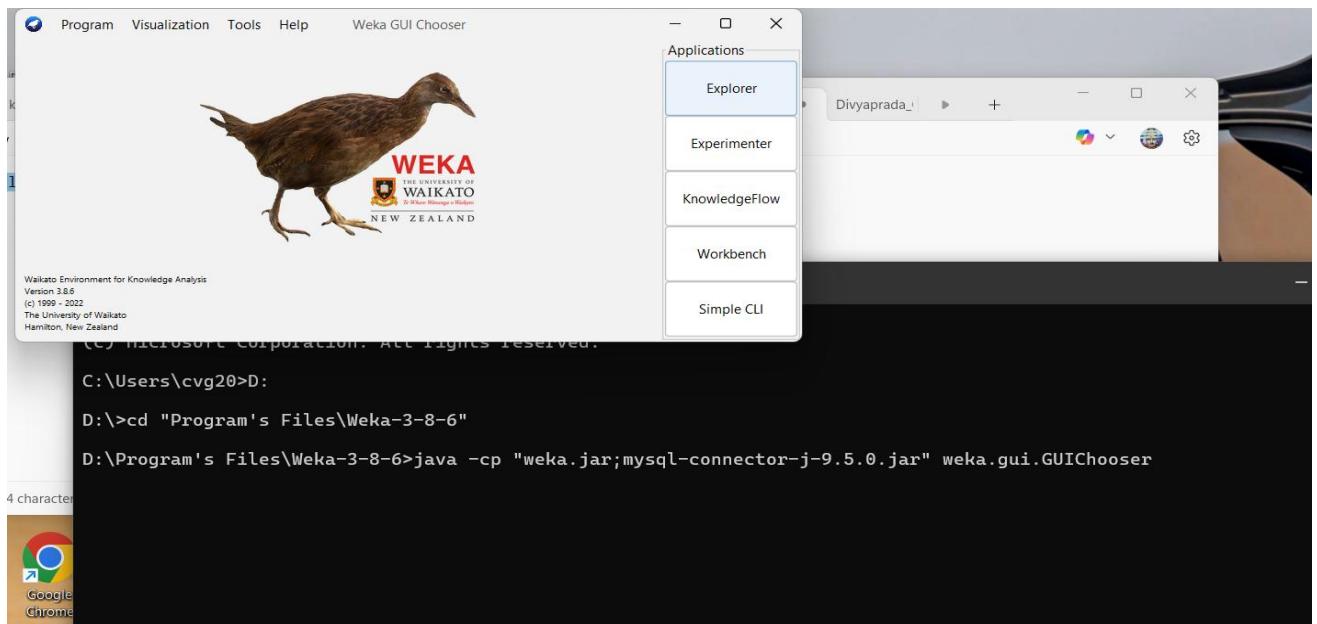
# other options
```

c). Connect WEKA to a relational database using JDBC.

Step 5:- Run Weka in CMD.

```
> D: cd "Program's Files\Weka-3-8-6"
```

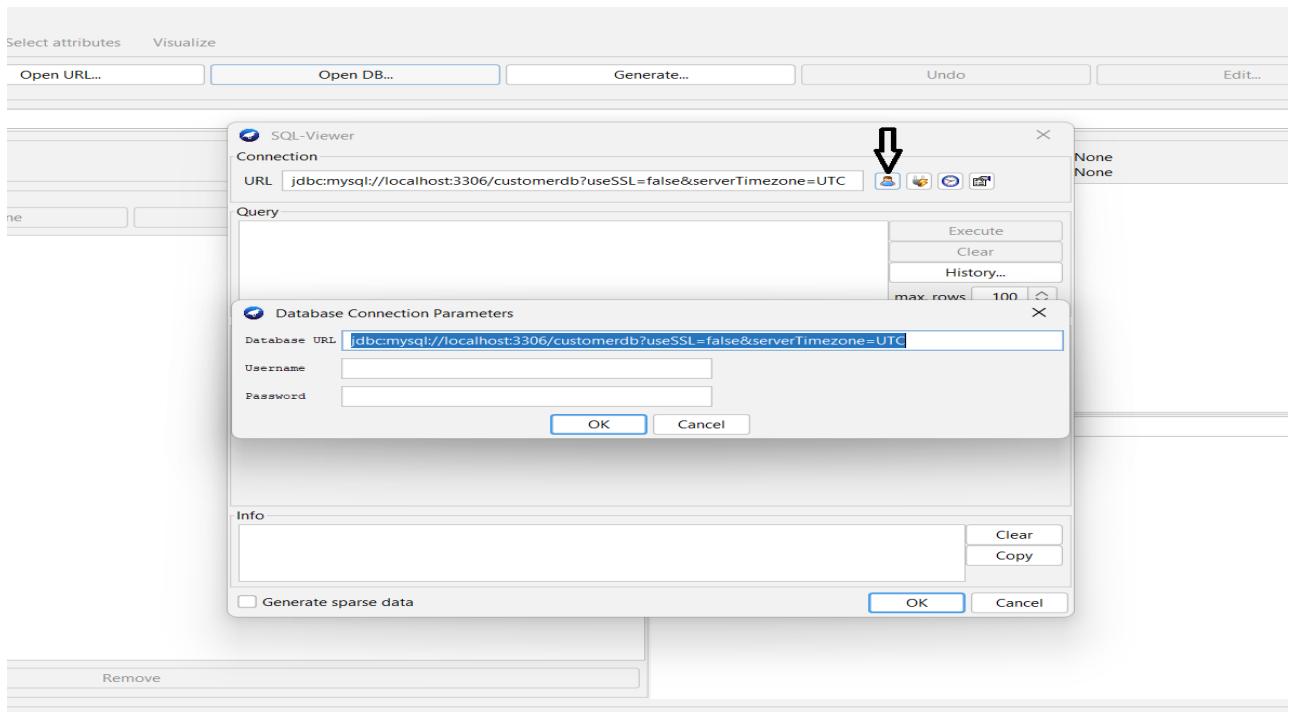
```
java -cp "weka.jar;mysql-connector-j-9.5.0.jar" weka.gui.GUIChooser
```



Explorer

↳ Preprocess

↳ Open DB



Enter the JDBC URL in props file

↳ Type Username

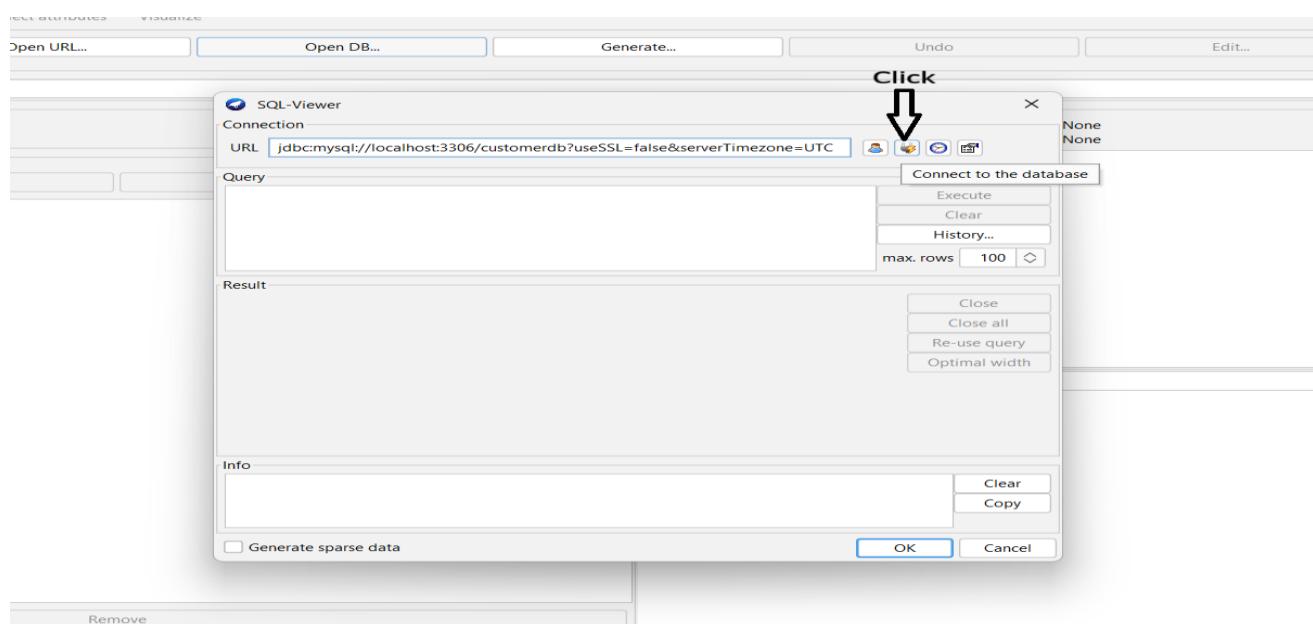
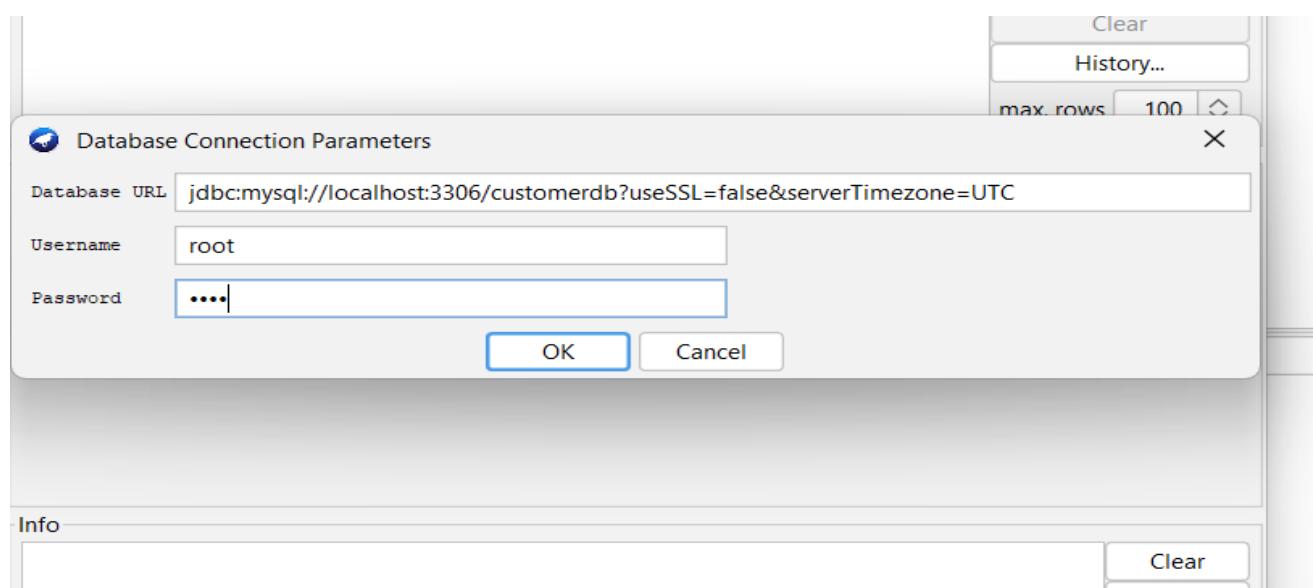
↳ Type Password

↳ Click OK

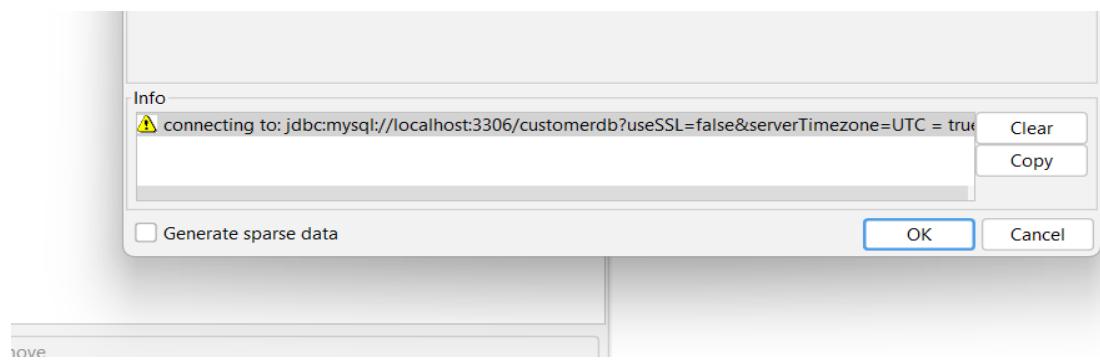
```
# JDBC driver (comma-separated list)
jdbcDriver=com.mysql.cj.jdbc.Driver

# database URL
jdbcURL=jdbc:mysql://localhost:3306/customerdb?useSSL=false&serverTimezone=UTC

# specific data types
STRING, getString() = 0;      --> nominal
```



- Check Server Connection.



d). Retrieve customer data directly using SQL queries.

Write the SQL Query

↳ Select * from <table_name>;

↳ Click Execute

↳ Click OK

The screenshot shows the SQL-Viewer application interface. In the top left, there's a "Connection" section with a URL field containing "jdbc:mysql://localhost:3306/customerdb?useSSL=false&serverTimezone=UTC". To the right of the URL are several icons: a user profile, a magnifying glass, a refresh, and a print. Below the connection section is a "Query" panel with the SQL command "select * from customers;". To the right of the query are three buttons: "Execute" (highlighted in blue), "Clear", and "History...". Below the query is a "max. rows" input field set to 100. In the main area, there's a "Result" table with the following data:

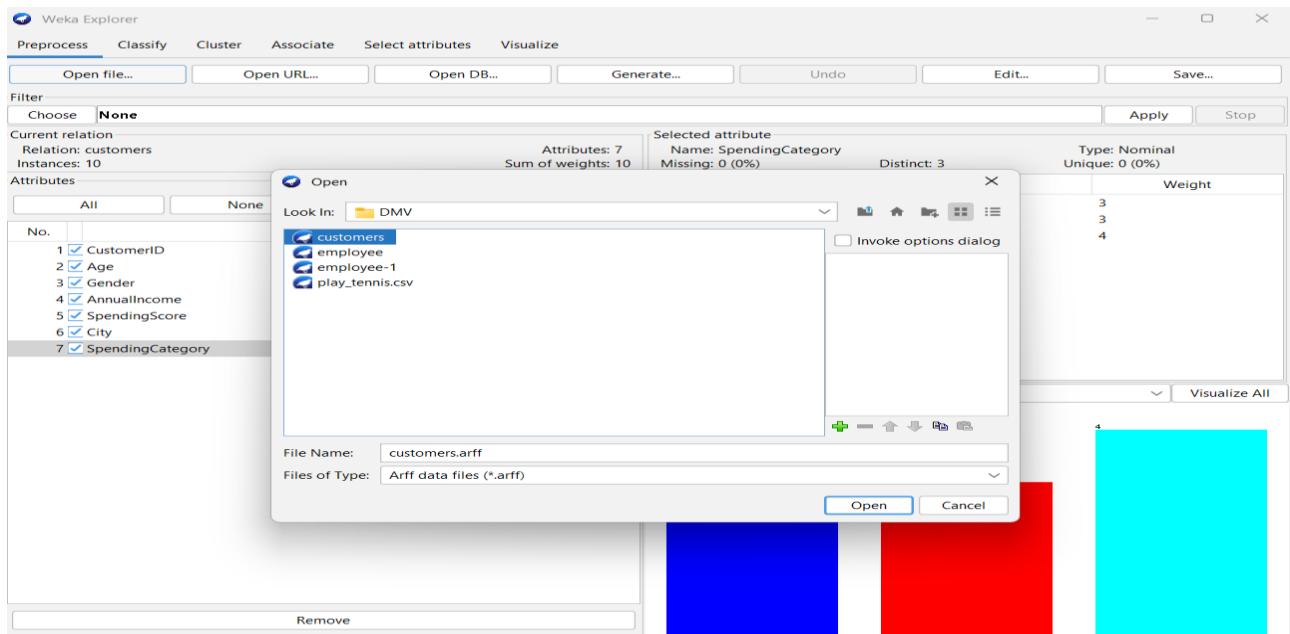
Row	customer_id	first_name	last_name	email	phone	city	country
1	1	Aarav	Sharma	aara...	9876...	De...	India
2	2	Priya	Patel	priya...	9876...	M...	India
3	3	Raj	Verma	rajv...	9876...	Ba...	India
4	4	Sneha	Iyer	sneh...	9876...	Ch...	India
5	5	Arjun	Nair	arjun...	9876...	Ko...	India
6	6	Kavya	Reddy	kavy...	9876...	Hy...	India

Below the table is a "Query1" label. In the bottom left of the main area is an "Info" panel with the following messages:

- ⚠ connecting to: jdbc:mysql://localhost:3306/customerdb?useSSL=false&serverTimezone=UTC = true
- ⚠ Query: select * from customers;
- ⚠ 8 rows selected.

At the very bottom of the application window are two checkboxes: "Generate sparse data" and "Current query: select * from customers;". To the right of these are "OK" (highlighted in blue) and "Cancel" buttons.

e). Load data into WEKA's Explorer interface.

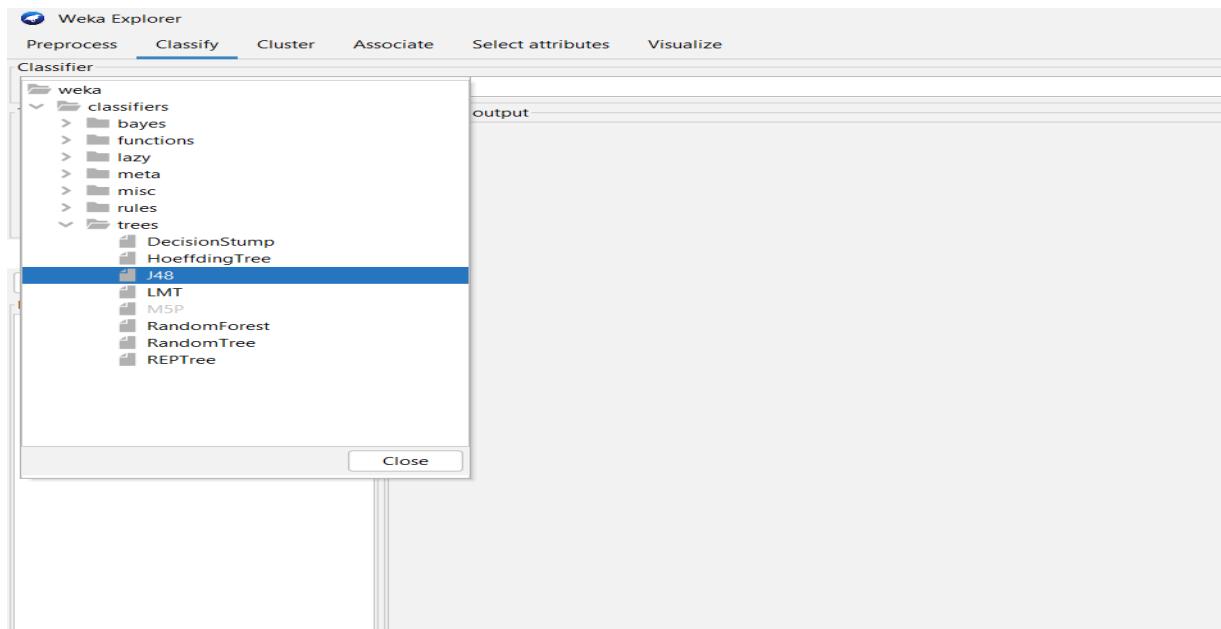


f). Apply classification algorithms (e.g., J48, Naive Bayes) to predict customer spending behavior.

Note:-

- **Training data** → loaded from your MySQL database (via query `SELECT * FROM customers;`)
- **Test data** → uploaded as an ARFF/CSV file (via “Supplied test set → Set...”).

1. Go to “Classify” tab.
 - You’ll see “Choose” button next to “Classifier”.
2. Choose an algorithm:
 - Click Choose → trees → J48 (Decision Tree algorithm).
 - Or try Choose → bayes → NaiveBayes.

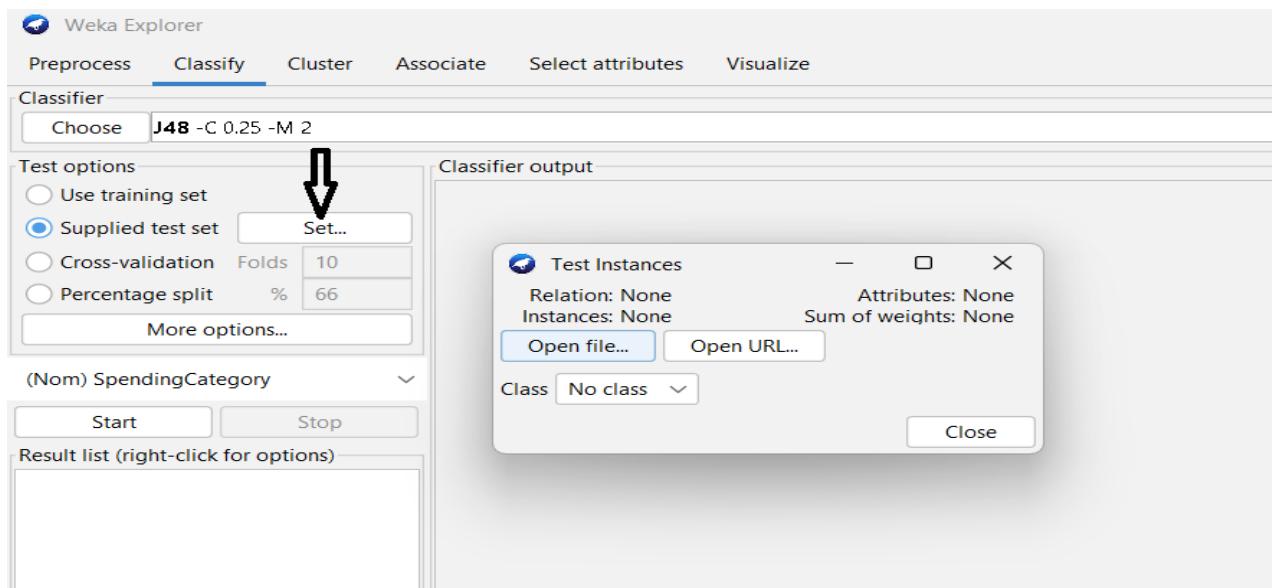


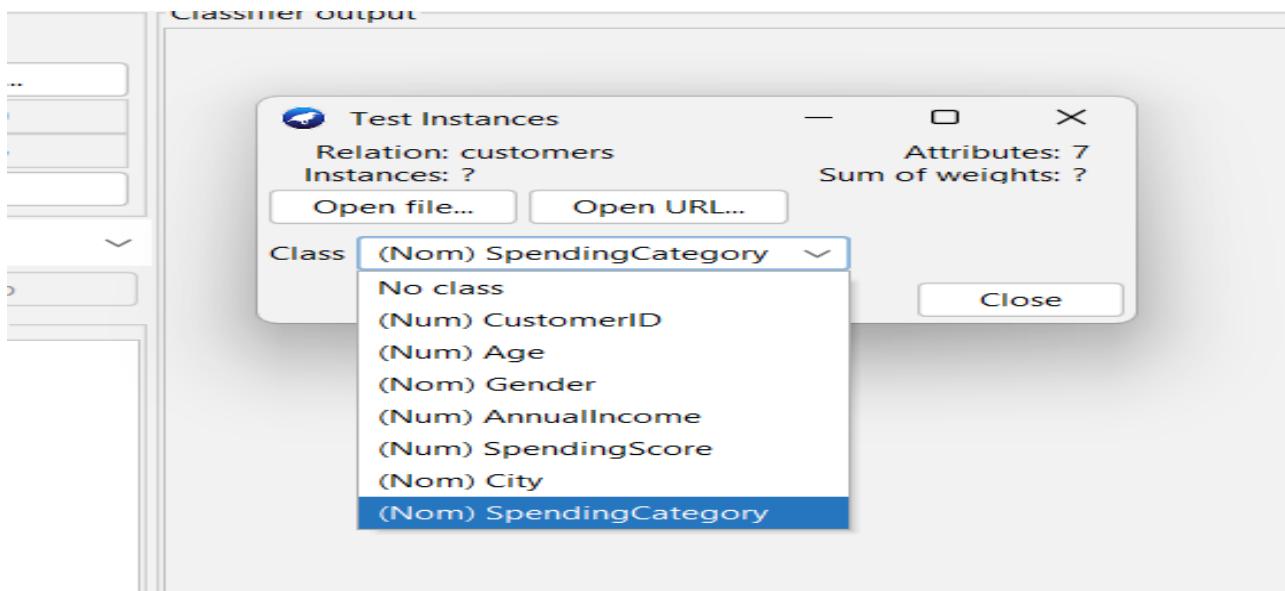
3. Select your class (target) attribute:

- At the bottom of the window, click “Set Class” and choose the column you want to predict (e.g., spend_category or purchase_behavior).

4. Evaluation method:

Choose Cross-validation (10 folds) — this is easiest.



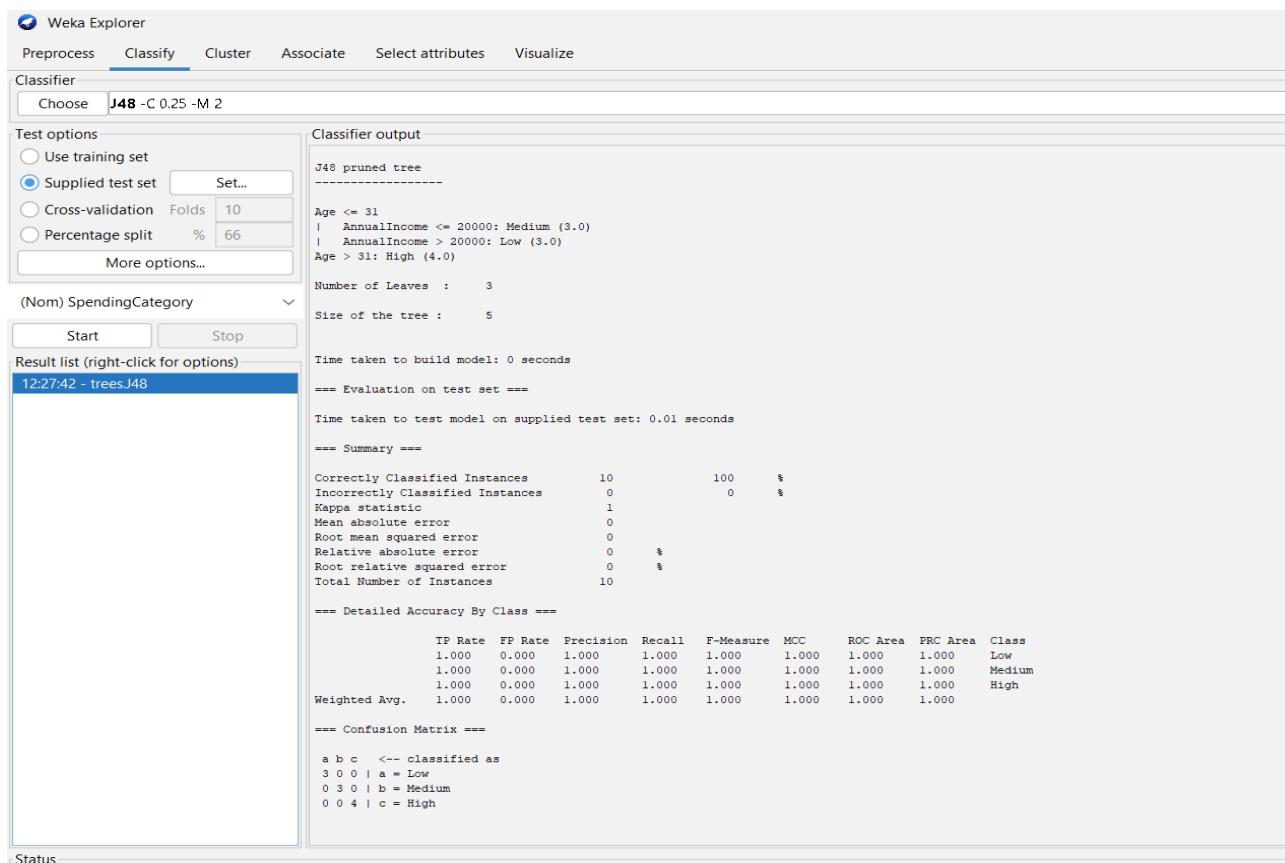


5. Click Start.

- Wait a few seconds → results will appear on the right side.

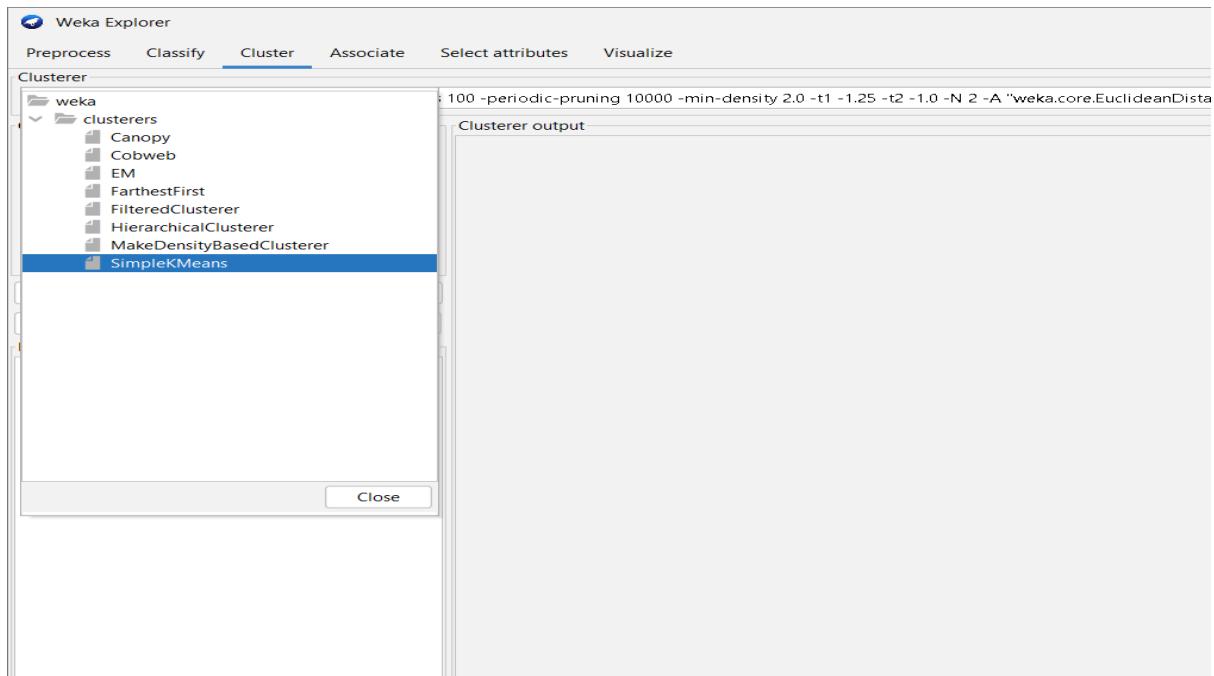
6. Read your results:

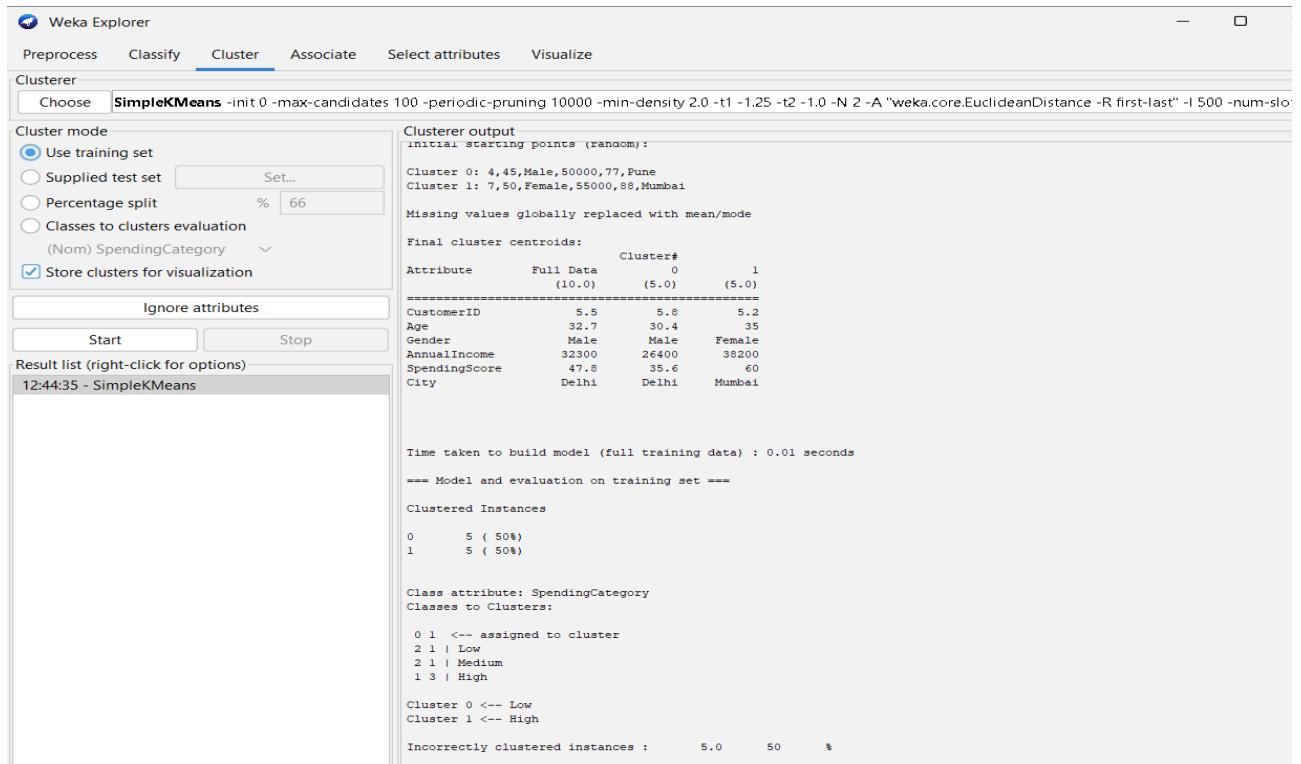
- You'll see Accuracy (%), Precision, Recall, F-Measure.
- You'll also see a Confusion Matrix (shows correct vs wrong predictions).



g). Use clustering (e.g., k-means) for market segmentation.

1. Click on “Cluster” tab (next to Classify).
2. Choose algorithm:
 - Click Choose → SimpleKMeans.
3. Configure:
 - Click on the name SimpleKMeans → change number of clusters (like 3 or 4)
→ Click OK.
4. Click Start.
 - WEKA will group your customers into clusters.
 - Output will show:
 - How many clusters (e.g., 3)
 - What each cluster’s average values are (like average income, age, etc.)
 - Which cluster each customer belongs to.





h). Generate evaluation metrics (accuracy, precision, recall).

You already get metrics when you run classification (Step o):

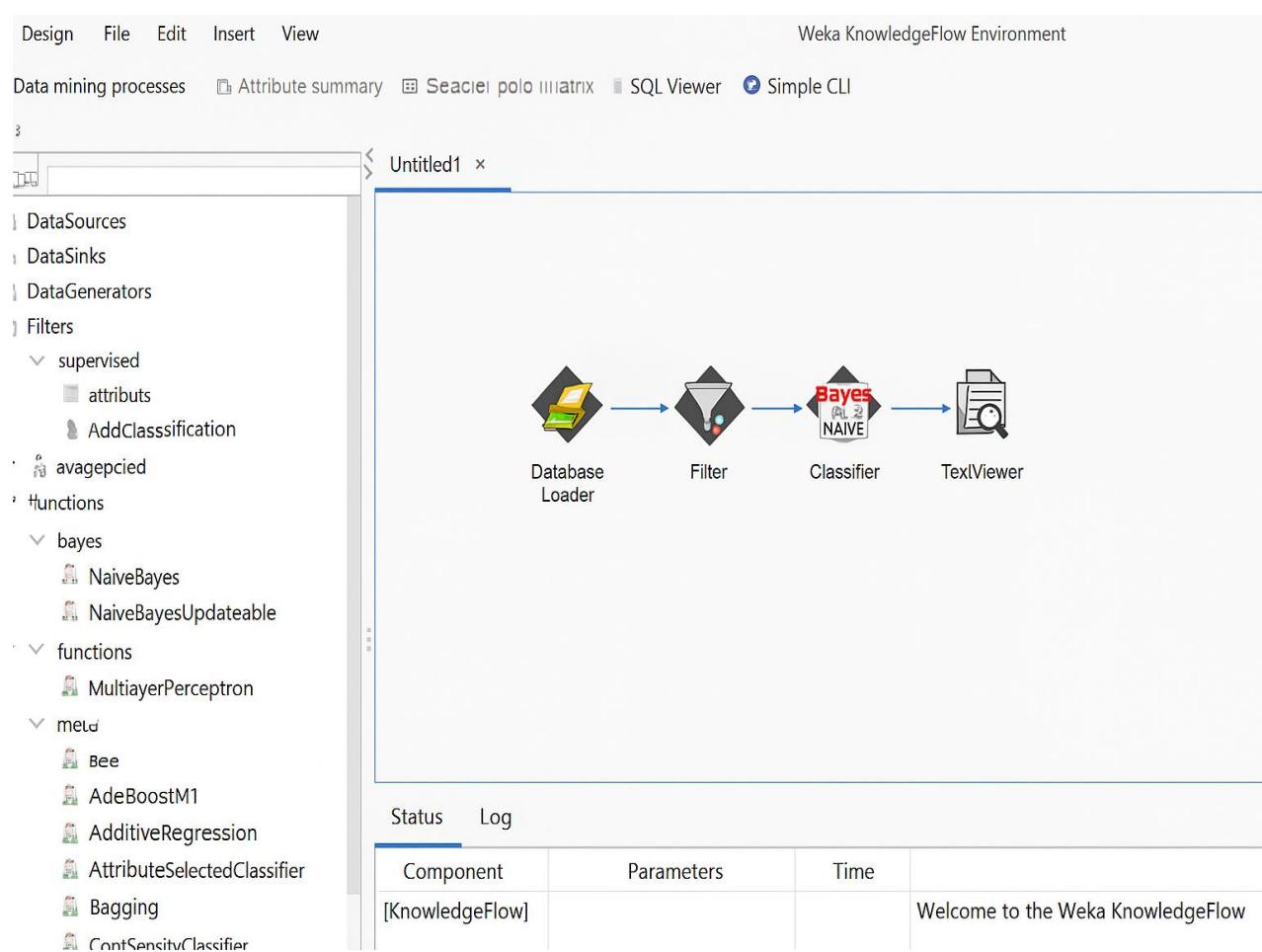
- Accuracy
- Precision
- Recall
- F-Measure
- Confusion Matrix

To save them:

- Right-click the result in the “Result list”.
- Click “Save result buffer” → save as .txt file.

i). Optionally automate data refresh using scripts or the WEKA KnowledgeFlow interface. Validate and analyse the result.

- Open WEKA → KnowledgeFlow (instead of Explorer).
- Drag these blocks:
 - DatabaseLoader → Filter → Classifier → TextViewer.
 - Connect them with arrows.
 - Right-click → Configure each block (like set SQL query, choose J48).
 - Click Run → It will do everything automatically (load → train → show result).
 - Save the flow file (.kfml) to reuse.



```

Classifier output
Scheme:      weka.classifiers.rules.ZeroR
Relation:    customers
Instances:   10
Attributes:  7
              CustomerID
              Age
              Gender
              AnnualIncome
              SpendingScore
              City
              SpendingCategory
Test mode:   10-fold cross-validation

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

ZeroR predicts class value: High

Time taken to build model: 0 seconds

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

Correctly Classified Instances          0           0   %
Incorrectly Classified Instances       10          100  %
Kappa statistic                      -0.5625
Mean absolute error                  0.4778
Root mean squared error              0.5083
Relative absolute error              100          %
Root relative squared error         100          %
Total Number of Instances            10

==== Detailed Accuracy By Class ====

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC     ROC Area  PRC Area  Class
      0.000    0.571    0.000     0.000    0.000     -0.535   0.000    0.300    Low
      0.000    0.000    ?         0.000    ?         ?        0.000    0.300    Medium
      0.000    1.000    0.000     0.000    0.000     -1.000   0.000    0.400    High
Weighted Avg.   0.000    0.571    ?         0.000    ?         ?        0.000    0.340

==== Confusion Matrix ====

a b c  <-- classified as
0 0 3 | a = Low
0 0 3 | b = Medium
4 0 0 | c = High

```