Program 1 : 17/7/25 **Practice 1: Loading and Exploring Data in WEKA**

Step 1: Launch WEKA

- Open the **WEKA application**.
- From the **WEKA GUI Chooser**, select **“Explorer”**.

🔷 Step 2: Open a Dataset

- You’ll land in the **Preprocess** tab by default.
- Click the **“Open file...”** button.
- Select a dataset from your system:
    - .arff (Attribute-Relation File Format) – preferred by WEKA.
    - .csv – Comma-separated file (make sure it’s properly formatted).
- Example datasets (bundled with WEKA):
    - weather.arff
    - iris.arff
    - contact-lenses.arff

📌 **Tip**: Sample datasets are usually found in:
C:\Program Files\Weka-3-8\data

🔷 Step 3: View Dataset Summary

Once loaded:

- The **relation name** is shown at the top.
- The **attributes (features)** are listed on the left.
- For each selected attribute, you can see:
    - **Type**: Nominal or Numeric
    - **Distinct Values**
    - **Missing Values**
    - **Histogram** showing distribution.

🔷 Step 4: Explore Individual Attributes

- Click on an attribute in the left panel.
- You'll see a **histogram** or **bar chart** for that attribute.
    - **Nominal attributes**: Show frequency of each category.
    - **Numeric attributes**: Show distribution in ranges.
- Observe class distribution using the color legend.

🔷 Step 5: Basic Data Operations

Use these options for preprocessing:

- **Remove**: Delete unwanted attributes.
- **Rename**: Change attribute names.
- **Edit**: Launches a spreadsheet-style editor to change individual records.

🔷 Step 6: Apply Filters (Optional)

- Use **filters** to preprocess your data.
    - Click on **"Choose"** under the Filter section.
    - Examples:
        - unsupervised.attribute.Remove: Remove specific attributes.
        - supervised.attribute.Discretize: Convert numeric to nominal.
        - unsupervised.instance.RemoveWithValues: Remove rows with specific values.

After choosing a filter, click **Apply** to process the data.

🔷 Step 7: Save the Processed Data (Optional)

- Click **"Save"** to store the cleaned or edited data.
- You can save it in:
    - .arff (default)
    - .csv (optional)

Pg 2: 24/7/25 Practice 2:Data Preprocessing in WEKA

✅ Step 1: Open WEKA Explorer

- Launch **WEKA GUI Chooser**
- Click **"Explorer"**

✅ Step 2: Load the Dataset

- Go to the **Preprocess** tab
- Click **"Open file..."**
- Load an .arff or .csv file
  - Example: weather.arff or iris.arff (found in the data/ folder of WEKA)

✅ Step 3: Explore the Data

- See **Relation name** and **Number of instances**
- View the list of **attributes (features)**
- Click an attribute to see:
  - Type (Nominal/Numeric)
  - wDistinct values
  - Mean, StdDev (for numeric)
  - Histogram (data distribution)

✅ Step 4: Apply Filters (Preprocessing Tasks)

🔷 A. Remove an Attribute

1. Click **"Choose"** → unsupervised.attribute.Remove
2. Click the filter name to configure it
3. Enter index of attribute to remove (e.g., 2)
4. Click **"Apply"**

🔷 B. Replace Missing Values

1. Click **"Choose"** → unsupervised.attribute.ReplaceMissingValues
2. Click **"Apply"**

🔷 C. Normalize Data (0–1 range)

1. Click **"Choose"** → unsupervised.attribute.Normalize
2. Apply to scale all numeric attributes

🔷 D. Discretize Numeric Attributes

(Useful when converting numeric data to nominal classes)

1. Choose unsupervised.attribute.Discretize
2. Click **"Apply"**

🔷 E. Convert String to Nominal

1. Choose unsupervised.attribute.StringToNominal
2. Set attribute index (e.g., first or 1)
3. Click **"Apply"**

✅ Step 5: Save the Preprocessed Data

- Click **“Save”**
- Choose .arff or .csv format
- Example: weather_cleaned.arff

✅ Step 6: Optional – Visualize Data

- Click **"Visualize All"** button
- Check scatter plots of attribute relationships

Additional:

Objective: Clean and prepare weather.arff dataset.

Steps:

1. Open weather.arff
2. Remove the humidity attribute
3. Replace any missing values
4. Normalize numeric data
5. Save the final dataset as weather_preprocessed.arff

📝 **Output File**

- Contains cleaned, normalized data ready for modeling.
- Can now be used in the **Classify** or **Cluster** tabs.

Pg 3: 31/7/25 : Data Visualization in WEKA

Step 1: Open WEKA

- Launch **WEKA** from your system.
- Choose **"Explorer"** from the GUI Chooser window.

Step 2: Load a Dataset

- Click on the **"Open file…"** button under the **Preprocess** tab.
- Select a dataset (e.g., weather.arff, iris.arff, or any .csv/.arff file).
- Once loaded, you will see the attributes listed on the left and summary statistics on the right.

Step 3: Explore Attribute-Wise Visualization

- Click on any attribute name from the list on the left.
- WEKA displays a **histogram** for that attribute.
    - **Nominal attributes**: Bar chart showing class distribution.
    - **Numeric attributes**: Distribution graph.
- Class labels are usually color-coded.

Step 4: Use the "Visualize All" Option

- At the bottom of the Preprocess tab, click the **"Visualize All"** button.
- A new window opens showing **scatter plots** for each attribute pair.
    - Each point is an instance.
    - Colors represent different class values.

Step 5: Customize Visualization

- In the Visualize panel:
    - Click on **any scatter plot** to enlarge it.

- You can set:
    - **X-axis and Y-axis** attributes.
    - **Coloring by class**.
- Use zoom and drag for better inspection.
- Right-click to save the plot as an image.

Step 6: Use Filters for Better Visualization (Optional)

- Back in the Preprocess tab, click **Choose** (in the Filter section).
- Try filters like:
    - **unsupervised.attribute.Remove** – remove unwanted attributes.
    - **supervised.attribute.Discretize** – convert numeric to nominal for better visual clarity.
- Apply the filter and recheck visualization.

Step 7: Save Modified Data (Optional)

- After filtering or preprocessing, click **"Save"** to store the modified dataset for future analysis.

Additional

Task: Load the iris.arff dataset and visualize sepal and petal dimensions.

Instructions:

1. Load iris.arff.
2. Go to **Visualize All**.
3. Select scatter plot with:
    - X-axis: petalwidth
    - Y-axis: sepallength
    - Coloring: by class
4. Observe cluster formation.

Pg 4:8/8/25

Association Rule Mining in WEKA – Step-by-Step Procedure

◈ Step 1: Launch WEKA

- Open WEKA.
- From the **GUI Chooser**, click **Explorer**.

◈ Step 2: Load a Dataset

- Go to the **Preprocess** tab.
- Click **“Open file...”**.
- Select an **ARFF** or **CSV** file suitable for association mining.
  - Use sample datasets like:
    - weather.nominal.arff
    - supermarket.arff
    - Or create your own market basket .arff file with **nominal** values.

◈ Step 3: Go to Associate Tab

- Click the **"Associate"** tab.
- This tab is used for discovering **association rules** from data.

◈ Step 4: Choose Algorithm

- Click on the **“Choose”** button.
- Select **weka.associations.Apriori** algorithm.

◈ Step 5: Configure Apriori Parameters

Click on **Apriori** to modify settings such as:

- LowerBoundMinSupport: Minimum support (default: 0.1)
- minMetric: Minimum confidence (default: 0.9)
- numRules: Number of rules to find (default: 10)

◈ **Example:** To find more rules, change numRules to 20.

Click **OK** after configuring.

◈ Step 6: Start Rule Mining

- Click the **“Start”** button.
- WEKA will generate association rules based on the dataset.

◈ Step 7: View Output

In the **Result List**, you will see output like:

Best rules found:

1. outlook=sunny humidity=high ==> play=no   conf:(0.9)

2. outlook=overcast ==> play=yes   conf:(1.0)

...

Each rule shows:

- **Antecedent** (IF part)
- **Consequent** (THEN part)
- **Confidence** value

◈ Step 8: Interpret the Results

Understand key metrics:

- **Support**: Frequency of itemset in the dataset.
- **Confidence**: How often the rule is correct.
- **Lift**: Measures how much more often the rule occurs than expected if the items were independent.

✅ Also Practise

Objective: Perform association rule mining on a transactional dataset using the Apriori algorithm.

Steps:

1. Load supermarket.arff.
2. Go to **Associate** tab.
3. Choose **Apriori**.
4. Set numRules = 15.
5. Click **Start**.
6. Analyze the top 5 rules and note support/confidence.

Points to Note:

- Make sure your dataset contains **nominal attributes only**.
- Convert numeric to nominal using filters if needed.
- For custom datasets, save in **ARFF format** for compatibility.
- Use **“Save”** to store the rules or result output.

Pg 6

Step 1: Launch WEKA

- Open **WEKA GUI Chooser**
- Click on **"Explorer"**

� Step 2: Load the Dataset

1. In the **"Preprocess"** tab, click **"Open file"**
2. Load the dataset weather.nominal.arff
   (Found in WEKA’s **data** folder)

� Step 3: Go to Association Rule Tab

- Click on the **“Associate”** tab

◈ Step 4: Choose Apriori Algorithm

- Click the **"Choose"** button
- Select:
  weka.associations.Apriori

◈ Step 5: Apply Constraints

1. Click on the text **"Apriori"** next to Choose to open parameter settings
2. Set the following parameters for constraint-based rule mining:

mathematica

CopyEdit

-N 10 -C 0.9 -M 0.1 -c last -R

☑ Explanation:

  - -N 10: Generate 10 rules
  - -C 0.9: Minimum confidence = 90%
  - -M 0.1: Minimum support = 10%
  - -c last: Use last attribute (play) as class (consequent)
  - -R: Remove rules that do not include the class

3. Click **OK**

◈ Step 6: Run the Algorithm

- Click **"Start"**
- WEKA will generate and display the top 10 association rules

---

◈ Step 7: Analyze the Output

In the **Result output**, you will see:

text

CopyEdit

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

Apriori

Best rules found:

1. humidity=normal 6 ==> play=yes 6   conf:(1.0)

2. outlook=overcast 4 ==> play=yes 4   conf:(1.0)

3. windy=FALSE 8 ==> play=yes 6   conf:(0.75)

...

☑ These rules satisfy:

- play=yes as the **consequent**
- At least **10% support**
- At least **90% confidence**

📌 **Result / Observation:**

- Association rules involving play=yes were successfully mined.
- Constraints on **confidence**, **support**, and **target attribute** were respected.

☑ **Conclusion:**

This practical demonstrates how **Apriori algorithm** can be used for **Constraint-Based Association Rule Mining** in WEKA by setting **user-defined parameters** such as support, confidence, and target class.

Pg 7: Implement Decision Tree Classifier using J48 (C4.5) in WEKA

Implement Decision Tree (J48/C4.5) in WEKA

1) Aim

Build and evaluate a J48 decision tree classifier on a given dataset in WEKA.

2) Requirements

- WEKA 3.8+ (Explorer GUI)
- Dataset: use any .arff or .csv (e.g., iris.arff, weather.nominal.arff in WEKA's data/)

---

3) Theory (very short)

- J48 = WEKA's implementation of C4.5 decision trees.
- Handles nominal features, tolerates missing values, supports pruning.

- Key parameters: C (confidence factor) for pruning, M (minNumObj) for minimum leaf size.

---

4) Step-by-Step in WEKA (Explorer GUI)

A. Load & Inspect Data (Preprocess tab)

1. Open WEKA → click Explorer.
2. Go to Preprocess tab → Open file… → choose your dataset (.arff or .csv).
3. Confirm at top-right: #Instances and #Attributes.
4. Set Class attribute (bottom right). If not set, use the Class: dropdown to pick your target column.
5. (Optional but recommended) Clean data:
    - Remove IDs: Filter → Unsupervised → Attribute → Remove → set index (e.g., 1) → Apply.
    - Handle missing values: Filter → Unsupervised → Attribute → ReplaceMissingValues → Apply.
    - If your class is numeric but you need classification: Filter → Unsupervised → Attribute → NumericToNominal (select the class index) → Apply.
    - If class is string: use StringToNominal on that attribute.

B. Choose Algorithm (Classify tab)

6. Go to Classify tab.
7. Choose → trees → J48.
8. Click J48 to set Options (common ones):
    - C (confidence factor): default 0.25 (lower ⇒ more pruning).
    - M (minNumObj): default 2 (higher ⇒ simpler tree).
    - -U (unpruned): builds unpruned tree (usually avoid for generalization).
    - -A (Laplace smoothing): improves probability estimates.

C. Pick Evaluation Method (right panel)

9. Test options (select one):
    - Cross-validation → 10-fold (good default).
    - Percentage split → e.g., 70% train / 30% test.
    - Supplied test set → click Set… and load test file.
    - (Use training set only for quick checks; it's optimistic.)

D. Train & View Results

10. Click Start.
11. Read Classifier output (center):
    - Correctly/Incorrectly Classified Instances (% Accuracy).
    - Kappa statistic.
    - Mean absolute error (MAE).
    - Confusion Matrix.
    - Per-class Precision/Recall/F-Measure, ROC Area (if generated).
    - Tree size and #Leaves.
12. In Result list (left), right-click your run:
    - Visualize tree (inspect splits/leaves).
    - Visualize classifier errors (spot misclassifications).
    - Visualize threshold curve (ROC) for a class.

E. Save Model / Outputs

13. Save model: right-click run → Save model (e.g., j48.model).
14. Save predictions: right-click run → Re-evaluate model on current test set → Save.
15. Export tree: in Visualize tree, use Save (image) or copy the textual tree from the output.

---

5) Experiment & Compare (Short Tuning Loop)

Run three experiments and record metrics:

- Run 1 (Baseline): defaults C=0.25, M=2, 10-fold CV.
- Run 2 (More pruning): C=0.15, M=5.
- Run 3 (Simpler leaves): C=0.25, M=10.

For each run, note: Accuracy, F1 (macro), #Leaves, Tree size, and confusion matrix. Compare complexity vs. performance.

---

6) Optional: Supplied Test / Percentage Split

- Supplied test set: Train on train.arff, test on test.arff to simulate real deployment.
- Percentage split: Try 70/30 or 80/20 and compare to 10-fold CV.

---

7) Optional: Command Line (Quick Reproducibility)

Open a terminal in WEKA directory (where weka.jar is):

```
# 10-fold cross-validation on iris
java -cp weka.jar weka.classifiers.trees.J48 -C 0.25 -M 2 -t data/iris.arff -x 10 -i


# Train on train.arff, test on test.arff, save model
java -cp weka.jar weka.classifiers.trees.J48 -C 0.15 -M 5 -t train.arff -T test.arff -i -k -d j48.model


# Unpruned (for comparison)
java -cp weka.jar weka.classifiers.trees.J48 -U -t data/iris.arff -x 10 -i
```

---

8) Optional: KnowledgeFlow (Pipeline)

1. KnowledgeFlow → New layout.
2. Add: ArffLoader → ClassAssigner → J48 → ClassifierPerformanceEvaluator → TextViewer.
3. Configure ClassAssigner to your class attribute.
4. On J48, set C, M as desired.
5. Connect components, then click Run → open TextViewer for metrics.

---

9) Interpreting Key Outputs (What to write in Observation)

- Accuracy: overall correctness; compare across runs.
- Confusion Matrix: where errors happen (which classes confuse).
- Precision/Recall/F1: especially when classes are imbalanced.
- Kappa: agreement beyond chance (≈0.6–0.8 good).
- Tree Size / Leaves: simpler trees are easier to explain; watch for overfitting.

---

10) Troubleshooting (Common Issues)

- Class not set / wrong target → Set Class: in Preprocess or use ClassAssigner.
- Class is numeric (regression) → convert to nominal for classification: NumericToNominal (select class index).

- Class is string → StringToNominal on the class attribute.
- Too many unique IDs → remove ID columns; they harm splits.
- Imbalanced classes → try Stratified CV (default in 10-fold) or CostSensitiveClassifier wrapper.
- Overfitting (huge tree, poor test) → decrease C (e.g., 0.15), increase M (e.g., 5–20), or try reduced-error pruning (-R).

---

11) Submission Template (fill this in your record)

- Dataset: ____________________________
- Instances / Attributes: ____________ / ____________
- Class Attribute: ___________________
- Preprocessing (filters used): ____
- Test Option: ☐ 10-fold CV ☐ % Split (/) ☐ Supplied Test
- J48 Options: C=____, M=____, other: _________________________
- Results:
  - Accuracy: __________ %
  - Kappa: __________
  - Precision / Recall / F1 (macro): __________ / __________ / __________
  - #Leaves: __________, Tree size: __________
  - Confusion Matrix: ____
- Observations (compare runs): ____

Pg 7:

Open WEKA Explorer

- Launch WEKA → click "Explorer" in the GUI Chooser. If you want a sample dataset, WEKA ships many under the data/ folder (e.g., iris.arff, weather.nominal.arff). MachineLearningMastery.comGeeksforGeeks

2. Load your data

- Go to the Preprocess tab → Open file… → choose your .arff or .csv file. (WEKA accepts CSV too.) MachineLearningMastery.com

3. Set the class (target) attribute

- In Preprocess, use the Class dropdown (near the top/right in some versions) to choose your label/target column (often the last attribute). EECIS

4. Go to the Classify tab and select J48

- Click Choose → trees → J48 (this is WEKA's C4.5 implementation). Sabancı University PeopleGeeksforGeeks

5. (Optional) Tweak key J48 options
   Click the text that says J48 to open its settings. Common, useful ones:

- -C Confidence factor for pruning (default 0.25; lower = more pruning).
- -M Minimum instances per leaf (default 2).
- -U Use an unpruned tree.
- -R Reduced-error pruning; -N folds for it (default 3).
- -B Binary splits on nominal attributes.
- -S Disable subtree raising.
- -A Laplace smoothing for probabilities.
  (You can leave defaults for a first run.) Weka

6. Pick how you want to evaluate (left side: Test options)

- Cross-validation (default) with 10 folds is a solid starting choice.
- Or choose Use training set, Supplied test set (load a separate file), or Percentage split. Then click Start to train and evaluate. MachineLearningMastery.comSabancı University People

7. Read the results

- The right pane shows accuracy, confusion matrix, number of leaves, tree size, etc. A new entry appears in the Result list (bottom-left). GeeksforGeeks

8. Visualize the tree (super handy!)

- In Result list, right-click your J48 run → Visualize tree to see the graph. You can also right-click for Visualize classifier errors. MachineLearningMastery.comfacweb.cs.depaul.edu

9. Save or reuse your trained model

- Right-click the entry in Result list → Save model.
- Later, load it via Right-click → Load model, set Supplied test set, then Re-evaluate model on current test set to get predictions for new data. (Use More options… → Output predictions if you want a CSV of predictions.) Waikato

10. That's it — you've trained a J48 decision tree!

Pg 8. k-NN (IBk) in WEKA — simple step-by-step

1. Open WEKA Explorer
   Launch WEKA → click Explorer. Medium

2. Load your dataset
   In Preprocess → Open file… → pick your .arff or .csv file. Make sure the class (target) attribute is set (usually the last attribute). Medium

3. (Important) Preprocess — scale numeric attributes
   k-NN uses distances, so scale numeric features first. In Preprocess → Choose → weka.filters.unsupervised.attribute.Normalize (or Standardize) → Apply. Normalization rescales numeric attributes (default to [0,1]) and prevents any single feature from dominating distance calculations. WekaMachineLearningMastery.com

Quick note: when you evaluate with cross-validation, avoid normalizing the *entire* dataset beforehand (that leaks info). Instead use a meta-classifier (FilteredClassifier) so normalization happens inside each CV fold — see step 5. Waikato

4. Pick the classifier (IBk = k-NN)
   Switch to the Classify tab → click Choose → navigate to classifiers → lazy → IBk. That is WEKA's k-NN implementation. WekaStack Overflow

5. (Best practice) Put Normalize inside a FilteredClassifier for correct CV
   For honest cross-validation: in Classify → Choose → meta → FilteredClassifier.

   - Click the Filter field → select unsupervised.attribute.Normalize.

   - Click the Classifier field inside FilteredClassifier → choose lazy → IBk.
     This ensures scaling is learned from each training fold and applied to its test fold (no data leakage). Waikato

6. Set IBk options (click the classifier name to edit)

   - K (number of neighbors) — common starters: 3 or 5; you can let WEKA search for a good k (see next bullet).

   - Distance weighting — options include "no weighting" (majority vote) or weight by distance (e.g., 1/d) so nearer neighbors count more.

   - crossValidate (inside IBk) — IBk can search for a best k by internal CV (turn this on if you want automatic selection).

   - NearestNeighbourSearch — you can pick LinearNNSearch (default) or faster structures like KDTree for large numeric datasets.
     (Open IBk's option dialog to change these.)
     MachineLearningMastery.comStack Overflow

7. Choose evaluation method & run
   In the Test options area choose one (typical):

   - Cross-validation → 10 folds (default) — good for small/medium datasets.

- Or Supplied test set to evaluate on separate test data.
  Click Start to run. Medium

8. Inspect results & save model
   After run: read accuracy, confusion matrix, class-by-class stats in the Classifier output. To save the trained model: right-click the result in Result list → Save model.

Pg 9

 Open WEKA Explorer
Start WEKA and click Explorer.

 Load your dataset
In Preprocess → Open file… choose your .arff or .csv. Make sure the class (target) attribute is set (usually the last attribute).

 Scale numeric attributes (very important)
SVMs are distance/scale-sensitive — normalize or standardize numeric attributes first. Two ways:

- Quick/manual: Preprocess → Filter → unsupervised.attribute.Normalize (or Standardize) → Apply.
- *Best practice for cross-validation:* wrap normalization inside a FilteredClassifier so scaling is learned from each training fold (prevents data leakage). In Classify → Choose → meta → FilteredClassifier, set Filter = Normalize and Classifier = SMO. This makes the scaling happen per fold. StudocuMachineLearningMastery.com

 Choose the classifier (SMO)
Go to Classify → Choose → functions → SMO. Click the SMO text to open option dialog. (SMO implements SVM with kernels like RBF and Polynomial.) Weka

 Pick a kernel
Click the kernel field inside SMO and choose one:

- RBFKernel — good default for many problems (has -G gamma option).
- PolyKernel — polynomial kernel (-E exponent).
- PUK — Pearson VII (another flexible kernel).
  You can start with RBFKernel and the default gamma (WEKA default is 0.01 for RBFKernel) then tune if needed. Weka+1

 Set SMO hyperparameters (basic ones to know)

- -C (complexity / regularization): larger C → less regularization (fits training harder).
- Kernel parameters: RBF -G (gamma), Poly -E (degree).
- Tolerance / epsilon and caching settings exist but defaults are fine for first runs.
  Tune C and kernel parameters with cross-validation (see step 8). Weka+1

 Choose evaluation method & run
In Test options pick:

- Cross-validation → 10 folds (typical), or
- Supplied test set if you have a held-out test file.
  Click Start to train & evaluate.

 Tune C and kernel params (if you want better performance)

- Use a grid search or Weka's meta-classifiers (e.g., CVParameterSelection) to search C × gamma (for RBF). Weka offers meta-tools to automate parameter search. Be careful: grid search with RBF on large datasets can be very slow — try a small sample first to find a promising region. WaikatoStack Overflow

 Inspect results & save model
The Classifier output shows accuracy, per-class metrics, confusion matrix. Right-click the result in Result list → Save model to reuse later.

 (Optional) Use LibSVM/LibLinear for large datasets
If SMO with RBF is too slow on big data, install LibSVM or LibLinear from WEKA's Package Manager — they're often faster/scalable for certain problems. Stack Overflow

Pg 10:

 Open WEKA Explorer
Launch WEKA and click Explorer.

 Load your data
Preprocess → Open file… → select your .arff or .csv. Remove any ID columns (they'll bias distance calculations).

 Preprocess: handle missing values & scale numeric features

- Replace missing values: Filters → unsupervised → attribute → ReplaceMissingValues (apply if needed).
- Normalize or Standardize numeric attributes (k-means uses distances; scale matters). Use Filters → unsupervised → attribute → Normalize or Standardize. Normalizing avoids one feature dominating the distance calculation. MachineLearningMastery.com

 Go to the Cluster tab and choose SimpleKMeans
Cluster → Choose → clusterers → SimpleKMeans. This is WEKA's k-means implementation; use the text box at the right to open its options. DePaul University Faculty Website

 Important SimpleKMeans options (click the classifier text to edit)

- Number of clusters (-N): set k (default = 2).
- Initialization method (-init): random (0), k-means++ (1), canopy (2), farthest-first (3). k-means++ often gives better starts. SciJava JavadocWeka

- Max iterations (-I) and Seed (-S) for reproducibility.
- Distance function (-A): default is Euclidean. Weka
- Display std devs (-V), Replace missing (-M), fast mode (faster distance calc) — explore if needed.

 Pick k (practical tips)

- If you don't know k, run SimpleKMeans for several k values and compare the sum of squared errors (SSE) / cluster interpretability (i.e., the "elbow" method). Repeat runs with different seeds or use k-means++ to reduce bad starts. (WEKA shows centroid means, counts and SSE in output.) Waikato

 Run clustering
With options set, click Start. The output pane will show centroids (mean vectors), cluster sizes, and stats.

 Visualize results & save cluster assignments

- Right-click the run in the Result list → Visualize cluster assignments to inspect clusters on scatterplots.
- In the visualize window click Save to export an ARFF that includes a cluster attribute (cluster assignment for each instance). This is how to get cluster labels back into a file. DePaul University Faculty WebsiteCS CCSU

 Save the clusterer (optional)
Right-click the result → Save model to reuse the trained clusterer on new data (or use the CLI / Java API to apply it).

 Command-line example (optional)

Pg 11:

DBSCAN in WEKA — simple step-by-step

Short summary: DBSCAN in WEKA is provided by the optics_dbScan package. Install the package (if not present), load & preprocess your data (remove IDs, handle missing values, scale numeric features), open the Cluster tab → choose DBSCAN, set epsilon (E) and minPoints (M) and run. Below are the exact clicks and practical tips.

---

1) (Only if DBSCAN is not available) Install the package

- From the WEKA GUI Chooser (the program start window) click Tools → Package Manager.
- Search for optics_dbScan (OPTICS and DBSCAN) and click Install.
- After installation restart WEKA so the new clusterer shows up. Weka

2) Open Explorer and load your data

- Launch Explorer → Preprocess → Open file… → pick your .arff or .csv.

- Remove any ID or index columns (they will distort distance calculations). Use Remove filter if needed.

3) Preprocess (important)

- Handle missing values (e.g., Filters → unsupervised → attribute → ReplaceMissingValues).
- Scale numeric attributes unless you have domain distances — DBSCAN uses a distance function (Euclidean by default) so features on different scales will break the neighborhood test. The OPTICS/DBSCAN package supports WEKA distance functions and normalization can be toggled, but in practice normalizing numeric attributes first is recommended. WekaWikipedia

4) Select DBSCAN

- Click the Cluster tab → click Choose → find clusterers → DBSCAN (provided by the optics_dbScan package). Click the DBSCAN text to edit options. Weka+1

5) Important DBSCAN options (what they mean)

- E (epsilon) — radius for neighborhood (ε). Default in many WEKA GUIs: 0.9 (but this is dataset dependent). NodePit
- M (minPoints) — minimum number of points required to form a dense region (minPts). Default often 6. NodePit
- Distance function / index — choose Euclidean (default) or another distance if appropriate.
(You'll usually only change E and M for a first run.) NodePit

6) How to pick good minPts and eps (practical rules)

- minPts (M): rule-of-thumbs: minPts ≥ D+1 (D = number of features). A common starting choice is minPts = 2 * D for noisy/high-dim data. Increase minPts for large / noisy datasets. WikipediaDataCamp
- eps (E): use a k-distance (k = minPts−1) plot — compute each point's distance to its k-th nearest neighbor, sort these distances ascending and look for the "elbow". That elbow is a good starting ε. If ε too small → most points labeled noise; too large → clusters merge. WikipediaDataCamp

Note: WEKA doesn't provide an automatic k-distance plot inside the DBSCAN dialog — you can compute a k-distance plot in Python/R or compute k-NN distances in WEKA/RWeka and inspect them to select ε.

7) Run DBSCAN

- With E and M set, click Start. The output shows cluster counts, noise points, and centroids/statistics (if applicable). Weka

8) Visualize & export results

- In the Result list (bottom-left) right-click the DBSCAN run → Visualize cluster assignments to inspect clusters on scatterplots (choose attributes for X/Y).
- Save cluster assignments: from the visualize or result dialog use the Save functions (this writes an ARFF with a cluster attribute or lets you save the model/output). WekaCourse Hero

9) Important caveats / tips

- DBSCAN does not require k (number of clusters) but is sensitive to ε and minPts — try a small grid of values and inspect cluster/ noise tradeoff. Wikipedia
- WEKA's DBSCAN implementation (from the package) may not support clustering new instances in some versions — check the package docs if you plan to apply a trained DBSCAN to new data. Weka
- Defaults (like E = 0.9, M = 6) are only starting points — they rarely fit raw real-world data without scaling/tuning. NodePit

Pg 12

1. Open WEKA GUI Chooser.
2. Menu **Tools → Package Manager**.
3. In the package list find **localOutlierFactor** (or "Local Outlier Factor") and click **Install**.
4. Restart WEKA if prompted.
   (WEKA packages are installed/managed from the Package Manager). waikato.github.ioweka.sourceforge.io

4) Create / save a tiny example dataset (ARFF)

Save the following as simple_lof.arff and open it in WEKA Explorer → Preprocess.

```
@RELATION simple_lof

@ATTRIBUTE age    NUMERIC
@ATTRIBUTE salary NUMERIC
@ATTRIBUTE dept   {IT,HR,Sales}
@ATTRIBUTE Outlier {normal,outlier}

@DATA
25,35000,IT,normal
27,37000,IT,normal
29,36000,IT,normal
```

45,120000,Sales,outlier

31,40000,HR,normal

30,38000,IT,normal

23,33000,HR,normal

60,200000,Sales,outlier

Notes:

- LOF's WEKA implementation expects the class attribute (if present) to be *unary or binary*, with **the first value = normal, second = outlier** (so use {normal,outlier} with **normal first** if you want to evaluate detection performance). weka.sourceforge.io

5) Preprocess in Explorer

1. Open the ARFF in **Explorer** → **Preprocess**.
2. Make sure attribute types are correct (numeric vs nominal). If necessary apply filters like NumericToNominal, StringToNominal, or remove irrelevant attributes. If attributes are on very different scales, consider **Normalize** or **Standardize** (LOF uses distances). MachineLearningMastery.comStack Overflow

6) Run LOF (as a Classifier — to get scores & evaluate)

1. Go to the **Classify** tab.
2. Click **Choose** → navigate to weka.classifiers.misc → select **LOF**. (The package includes a LOF classifier and a LOF filter.) weka.sourceforge.io+1
3. Click the name to open options: you'll see options like -min (lower bound for k) and -max (upper bound for k); default is -min 10 -max 40. Adjust if your dataset is small (e.g., use -min 3 -max 10 for small samples). weka.sourceforge.io
4. Set **Test options**: if you have a labeled class (Outlier), choose **Cross-validation (10-fold)** to evaluate detection; otherwise use **Supplied test set** or **Use training set** to just compute scores.
5. Click **Start**.

Interpreting output: LOF will produce an output in the result pane. The implementation returns 1 − normalized outlier score in the first element of the distribution; if you used a binary class the second element holds the normalized outlier score. So larger normalized LOF → more anomalous. weka.sourceforge.io

7) Save / inspect predictions (view top flagged instances)

- After the run, right-click the result in the result list → **Visualize classifier errors** (or **Save result buffer / Save predictions**) — you can save the predictions as an ARFF (for example LOF_Results.arff). Opening that ARFF will show new attributes (prediction margin / predicted outlier) so you can sort and inspect predicted outliers. This is a common workflow in LOF labs. Course Hero

8) Run LOF as a filter (optional; unsupervised scoring)

- The package includes a **filter** that computes a per-instance LOF score and adds it as an attribute. Use **Preprocess → Choose filter** and look for the localOutlierFactor filter (installed with the package). Then **Apply** and the dataset will be augmented with the LOF score — useful when you want to export scores or combine with other methods. weka.sourceforge.io+1

9) Command-line example (quick)

If you prefer CLI, after installing package or having the package classes on classpath you can run:

java -cp weka.jar weka.classifiers.misc.LOF -t simple_lof.arff -min 3 -max 10 -num-slots 2

This runs LOF over simple_lof.arff with min_k=3, max_k=10 and 2 parallel slots. (The LOF class supports -min, -max, -num-slots etc.) weka.sourceforge.io

10) Troubleshooting tips

- If LOF is not visible after install, restart WEKA & confirm the package is **loaded** in Package Manager. waikato.github.io
- If you cannot run LOF on your dataset, check the **Capabilities** of LOF and compare against your dataset (string attributes, missing values, etc.). Convert string attributes or remove non-numeric features if needed. Stack Overflow
- Scale/normalize numeric attributes first if attributes are on different scales — LOF is distance-based.