

Session 2

Basic preprocessing

Preprocessing decides what the algorithms see. This session removes attributes, applies filters and records the effect on three small datasets.

Objectives

✕ Do not copy. Read for understanding and the viva

- Complete questions 5 to 6 of the manual: basic preprocessing
- Prepare the deliverable before the lab and finish it during the session
- Be ready to explain every step in the viva

Questions Covered

✕ Do not copy. Read for understanding and the viva

Question	Requirement	Status
Q5	Perform the basic pre-processing operations on data relation such as removing an...	Complete
Q6	Demonstrate the preprocessing mechanism on the following datasets	Complete

Preparation

✕ Do not copy. Read for understanding and the viva

- Remove an attribute with the checkbox and Remove button, or with the `Remove` unsupervised attribute filter (which is repeatable).
- Useful filters: `ReplaceMissingValues`, `Normalize`, `Standardize`, `Discretize`, `NominalToBinary`, `Resample`.

- Before and after each filter, note the attribute count, instance count and any changed value ranges.

Question 5

Problem Statement

■ Write in lab record

Perform the basic pre-processing operations on data relation such as removing an attribute and filter attribute bank data.

Solution

■ Write in lab record

The bank data is `bank-data.csv`: 600 bank customers, 12 attributes (`id`, `age`, `sex`, `region`, `income`, `married`, `children`, `car`, `save_act`, `current_act`, `mortgage`, `pep`). `pep` (bought a Personal Equity Plan) is the class. The file is distributed with the lab; it is not in WEKA's data folder.

Steps

1. Explorer, Open file..., Files of type CSV data files, open `bank-data.csv`. Click Save..., type Arff data files, name `Bank.arff`. From now on open the ARFF.
2. Remove `id` with the button: tick the checkbox next to `id` in the Attributes list, click Remove at the bottom of the list. The Current relation box now says Attributes: 11. Click Undo if you removed the wrong one.
3. Remove with a filter (repeatable, and it can be saved in the relation name): Filter, Choose, `weka.filters.unsupervised.attribute.Remove`. Click the filter name to open its options, set `attributeIndices` to `1`, leave `invertSelection` False, OK, then Apply. Both routes give the same result; use one.
4. Discretize `age` (now attribute 1): Choose `weka.filters.unsupervised.attribute.Discretize`, set `attributeIndices` to `1`, `bins` to `3`, keep `useEqualFrequency` False, OK, Apply. Click `age`: Type has changed from Numeric to Nominal with three labels.
5. Discretize `income` (attribute 4): same filter, `attributeIndices` `4`, `bins` `3`, Apply.

6. Convert `children` (attribute 6, integers 0 to 3) with `weka.filters.unsupervised.attribute.NumericToNominal`, `attributeIndices` `6`, Apply. Values 0 to 3 become four labels.
7. Save as `bank-nominal.arff`. This all-nominal file is what Apriori needs in Session 4.

Output

Before and after, as the Preprocess panel reports it. The bin boundaries are what equal-width `Discretize` with 3 bins computes from age 18 to 67 and income 5014.21 to 63130.1:

Attribute	Before	After	Filter
id	Numeric, 600 distinct, 600 unique	removed	Remove, attributeIndices 1
age	Numeric, min 18, max 67	Nominal: '(-inf- 34.333333]', '(34.333333- 50.666667]', '(50.666667- inf)'	Discretize, bins 3
sex	Nominal FEMALE 300, MALE 300	unchanged	none
region	Nominal INNER_CITY, TOWN, RURAL, SUBURBAN	unchanged	none
income	Numeric, min 5014.21, max 63130.1	Nominal: '(-inf- 24386.173333]', '(24386.173333- 43758.136667]', '(43758.136667-inf)'	Discretize, bins 3
married, car, save_act, current_act, mortgage	Nominal YES, NO	unchanged	none
children	Numeric 0 to 3, 4 distinct	Nominal 0, 1, 2, 3	NumericToNominal
pep	Nominal YES 274, NO 326 (class)	unchanged	none

Instances stay at 600 through every step; attributes go from 12 to 11 after Remove and stay at 11. The relation name in the panel grows with each filter, for example:

```
1 Relation: bank-data-weka.filters.unsupervised.attribute.Remove-R1-weka.filte
2 Instances: 600      Attributes: 11
```

The width of an age bin is $\frac{67-18}{3} = 16.333$, so the cut points are $18 + 16.333 = 34.333$ and $18 + 32.667 = 50.667$; the same arithmetic on income gives 24386.17 and 43758.14.

Explanation

`id` carries no pattern, only identity, and any learner that sees it can memorise the data, so it goes first. Apriori and most rule learners accept only nominal attributes, so the numeric ones are binned. Equal-width bins are easy to explain but can leave one bin nearly empty when the data is skewed (income here); `useEqualFrequency` True gives bins with 200 rows each and is worth comparing. `NumericToNominal` is the right tool when the numbers are already categories, because it keeps every distinct value as a label instead of grouping them.

Question 6

Problem Statement

■ Write in lab record

Demonstrate the preprocessing mechanism on the following datasets:

1. `student.arff`
2. `labor.arff`
3. `contactlenses.arff`

Solution

■ Write in lab record

Steps

`student.arff` (30 rows, written in Session 1, two missing cells):

1. Open file `student.arff`. Click `attendance` : Missing 1 (3%). Click `internal` : Missing 1 (3%).
2. Filter, Choose, unsupervised, attribute, `ReplaceMissingValues`, Apply. Both Missing fields now read 0. The mean of the other 29 values was written into the empty cell.

3. Choose **Normalize** (defaults **scale** 1.0, **translation** 0.0), Apply. Every numeric attribute now runs from 0 to 1; nominal attributes are untouched.
4. Undo, then Choose **Discretize**, **attributeIndices** 5 (attendance), **bins** 3, Apply. Read the three labels and counts.
5. Undo, then Choose unsupervised, instance, **Resample**, set **sampleSizePercent** 50, **noReplacement** True, **randomSeed** 1, Apply. Instances drop from 30 to 15.
6. Run **preprocess.py** on the same file to check every number.

labor.arff (57 rows, 17 attributes, many missing values):

1. Open file **labor.arff**. Click through the attributes and note Missing: **standby-pay** and **wage-increase-third-year** are missing in most rows, **duration** in almost none. Class **class** is **bad** 20, **good** 37.
2. Choose **Remove**, **attributeIndices** 8,4 (standby-pay and wage-increase-third-year, the two mostly-empty columns), Apply. Attributes 17 to 15.
3. Choose **ReplaceMissingValues**, Apply. Every Missing field reads 0; numeric gaps got the attribute mean, nominal gaps got the most frequent label.
4. Choose **Normalize**, Apply. **duration** (1 to 3), **working-hours** (27 to 40) and the wage attributes now share the 0 to 1 range, so a distance-based learner will not be dominated by working-hours.
5. Choose **Discretize** with **attributeIndices** **first-last** and **bins** 3, Apply: an all-nominal labor file for rule mining.

contact-lenses.arff (24 rows, all nominal, no missing values):

1. Open file **contact-lenses.arff**. Every Type is Nominal, every Missing is 0, so **ReplaceMissingValues**, **Normalize** and **Discretize** change nothing; apply **Discretize** once to see that the relation name changes but no attribute does.
2. Choose **Remove**, **attributeIndices** 1, Apply: **age** gone, 4 attributes left. Undo.
3. Choose **NominalToBinary** with **transformAllValues** True, Apply: 5 attributes become 12 binary ones (**age=young**, **age=pre-presbyopic**, ...). This is the form FP-Growth wants in Session 4. Undo.
4. Choose instance, **Resample**, **sampleSizePercent** 50, **noReplacement** True, Apply: 12 instances. Click **contact-lenses** and compare the class counts with the original soft 5, hard 4, none 15.

Program

preprocess.py

```

1  #!/usr/bin/env python3
2  """Reproduce WEKA's Preprocess panel numbers and four filters on student.arf
3
4  Standard library only. Prints, in order:
5      1. the per-attribute summary the Preprocess panel shows (counts, min, max,
6          mean, sample standard deviation, missing, distinct)
7      2. ReplaceMissingValues (numeric → mean, nominal → mode)
8      3. Normalize (x' = (x - min) / (max - min) on every numeric at
9      4. Discretize (equal-width bins with WEKA's labels)
10     5. Resample (50 % without replacement; class counts before an
11
12 Usage: python3 preprocess.py student.arff
13 """
14 import random
15 import re
16 import statistics
17 import sys
18 from collections import Counter
19
20
21 def read_arff(path):
22     attrs, rows, in_data = [], [], False
23     for line in open(path):
24         line = line.strip()
25         if not line or line.startswith("%"):
26             continue
27         if in_data:
28             rows.append([v.strip() for v in line.split(",")])
29         elif line.lower().startswith("@attribute"):
30             m = re.match(r"@attribute\s+(\S+)\s+(.*)", line, re.I)
31             typ = m.group(2).strip()
32             attrs.append((m.group(1), [v.strip() for v in typ.strip("{}").split(" ")]))
33         elif line.lower().startswith("@data"):
34             in_data = True
35     return attrs, rows
36
37
38 def fmt(x, places=3):
39     return f"{x:.{places}f}".rstrip("0").rstrip(".")
40
41
42 def summary(attrs, rows):
43     print(f"Instances: {len(rows)}    Attributes: {len(attrs)}\n")

```

```

44     for j, (name, values) in enumerate(attrs):
45         col = [r[j] for r in rows]
46         missing = col.count("?")
47         present = [v for v in col if v != "?"]
48         if values is None:
49             nums = [float(v) for v in present]
50             print(f"{name:12s} Numeric Missing: {missing} ({100 * missing /
51                 f" Min: {fmt(min(nums))} Max: {fmt(max(nums))} Mean: {f
52                 f" StdDev: {fmt(statistics.stdev(nums))}")
53         else:
54             c = Counter(present)
55             print(f"{name:12s} Nominal Missing: {missing} ({100 * missing /
56                 " ".join(f"{v}: {c[v]}" for v in values))
57
58
59 def main():
60     attrs, rows = read_arff(sys.argv[1])
61     print("=== 1. Preprocess panel, as loaded ===")
62     summary(attrs, rows)
63
64     print("\n=== 2. ReplaceMissingValues ===")
65     for j, (name, values) in enumerate(attrs):
66         present = [r[j] for r in rows if r[j] != "?"]
67         fill = fmt(statistics.mean(float(v) for v in present)) if values is
68         for i, r in enumerate(rows):
69             if r[j] == "?":
70                 print(f"row {i + 1}: {name} ? → {fill}")
71                 r[j] = fill
72
73     print("\n=== 3. Normalize (first 5 rows, numeric attributes only) ===")
74     numeric = [j for j, (_, v) in enumerate(attrs) if v is None]
75     lo = {j: min(float(r[j]) for r in rows) for j in numeric}
76     hi = {j: max(float(r[j]) for r in rows) for j in numeric}
77     print("before:", *[" ".join(r[j] for j in numeric) for r in rows[:5]], s
78     norm = [[fmt((float(r[j]) - lo[j]) / (hi[j] - lo[j])) for j in numeric]
79     print("after: ", *[" ".join(r) for r in norm], sep="\n ")
80
81     print("\n=== 4. Discretize attendance, 3 equal-width bins ===")
82     j = [n for n, _ in attrs].index("attendance")
83     vals = [float(r[j]) for r in rows]
84     width = (max(vals) - min(vals)) / 3
85     cuts = [min(vals) + width, min(vals) + 2 * width]
86     labels = [f"'(-inf-{fmt(cuts[0], 6)}]'", f"'({fmt(cuts[0], 6)}-{fmt(cuts
87     bins = Counter(labels[0 if v ≤ cuts[0] else 1 if v ≤ cuts[1] else 2] f

```

```
88     for lab in labels:
89         print(f"{lab} {bins[lab]}")
90
91     print("\n≡ 5. Resample 50% without replacement (seed 1) ≡")
92     cls = len(attrs) - 1
93     print("before:", len(rows), dict(Counter(r[cls] for r in rows)))
94     sample = random.Random(1).sample(rows, len(rows) // 2)
95     print("after: ", len(sample), dict(Counter(r[cls] for r in sample)))
96
97
98 if __name__ == "__main__":
99     main()
```

Output

`python3 preprocess.py student.arff` (run here; the same values appear in the Preprocess panel, which also uses the sample standard deviation):

```

1  ≡≡ 1. Preprocess panel, as loaded ≡≡
2  Instances: 30   Attributes: 9
3
4  sid           Numeric  Missing: 0 (0%)  Distinct: 30  Min: 1  Max: 30  Mean:
5  age           Numeric  Missing: 0 (0%)  Distinct: 5   Min: 19  Max: 23  Mean:
6  gender        Nominal  Missing: 0 (0%)  M: 15  F: 15
7  stream        Nominal  Missing: 0 (0%)  science: 11  commerce: 10  arts: 9
8  attendance     Numeric  Missing: 1 (3%)  Distinct: 29  Min: 40  Max: 95  Mean:
9  internal       Numeric  Missing: 1 (3%)  Distinct: 21  Min: 9   Max: 29  Mean:
10 sem_marks     Numeric  Missing: 0 (0%)  Distinct: 30  Min: 22  Max: 68  Mean:
11 hours_study   Numeric  Missing: 0 (0%)  Distinct: 10  Min: 0.5  Max: 5   Mean:
12 result        Nominal  Missing: 0 (0%)  pass: 18  fail: 12
13
14 ≡≡ 2. ReplaceMissingValues ≡≡
15 row 18: attendance ? → 70.379
16 row 9: internal ? → 18.69
17
18 ≡≡ 3. Normalize (first 5 rows, numeric attributes only) ≡≡
19 before:
20   1 20 85 24 58 3.5
21   2 21 72 18 42 2.0
22   3 19 55 12 28 1.0
23   4 22 90 27 65 4.0
24   5 20 60 15 35 1.5
25 after:
26   0 0.25 0.818 0.75 0.783 0.667
27   0.034 0.5 0.582 0.45 0.435 0.333
28   0.069 0 0.273 0.15 0.13 0.111
29   0.103 0.75 0.909 0.9 0.935 0.778
30   0.138 0.25 0.364 0.3 0.283 0.222
31
32 ≡≡ 4. Discretize attendance, 3 equal-width bins ≡≡
33 '(-inf-58.333333]' 8
34 '(58.333333-76.666667]' 10
35 '(76.666667-inf)' 12
36
37 ≡≡ 5. Resample 50% without replacement (seed 1) ≡≡
38 before: 30 {'pass': 18, 'fail': 12}
39 after: 15 {'fail': 8, 'pass': 7}

```

Before and after summary for the three files:

Dataset	Filter	Before	After
student	ReplaceMissingValues	attendance missing 1, internal missing 1	0 missing; row 18 attendance = 70.379, row 9 internal = 18.69
student	Normalize	attendance 40 to 95, sem_marks 22 to 68	all numeric 0 to 1; row 1 attendance 85 becomes 0.818
student	Discretize attendance, 3 bins	numeric, 29 distinct	3 labels with counts 8, 10, 12
student	Resample 50 percent	30 rows, pass 18 fail 12	15 rows; class mix depends on the seed (7 pass, 8 fail with seed 1 here)
labor	Remove 8,4 then ReplaceMissingValues	17 attributes, many missing	15 attributes, 0 missing, 57 rows
labor	Normalize	working-hours 27 to 40, duration 1 to 3	both 0 to 1
contact-lenses	Remove 1	5 attributes	4 attributes, 24 rows
contact-lenses	NominalToBinary, transformAllValues	5 nominal	12 binary attributes
contact-lenses	Resample 50 percent	24 rows, none 15 soft 5 hard 4	12 rows

Explanation

Each filter answers one question. `ReplaceMissingValues` keeps the row instead of throwing it away, at the cost of pulling the value toward the mean, which is why the attendance mean is unchanged after the fill. `Normalize` uses the formula-sheet map $x' = (x - x_{\min}) / (x_{\max} - x_{\min})$:

attendance 85 becomes $\frac{85-40}{95-40} = 0.818$. **Discretize** with equal width divides the range 40 to 95 into three bins of width 18.333, giving the cut points 58.333 and 76.667 that appear in the labels. **Resample** is a random subsample, so WEKA's seed 1 and Python's seed 1 give different rows; the count (15) and the approximate class mix are what to compare, not the exact rows. On contact-lenses nothing numeric exists, so the only meaningful preprocessing is removing or binarising attributes and sampling rows.

Viva Questions

✗ Do not copy. Read for understanding and the viva

Q: What is the difference between removing an attribute with the button and with the **Remove** filter? **A:** Same result; the filter can be reapplied on another file and is recorded in the relation name, the button is a one-off.

Q: What does **ReplaceMissingValues** put into a missing numeric cell? Into a nominal one? **A:** The mean of the present values; the most frequent label (the mode).

Q: Why normalise before k-nearest neighbour but not before J48? **A:** k-NN uses Euclidean distance, so an attribute with a large range dominates; J48 compares one attribute against a threshold, so scale does not matter.

Q: How does equal-width **Discretize** choose its cut points? **A:** It splits the range from minimum to maximum into **bins** equal intervals; the labels show the cut points.

Q: What does **sampleSizePercent** 50 with **noReplacement** True do? **A:** Draws half of the rows at random, each row at most once.

Q: Why are there no filters to apply on contact-lenses for missing values or scaling? **A:** All attributes are nominal and complete; only Remove, NominalToBinary and Resample change anything.

Q: Which filter do you apply on **children** in bank-data and why not Discretize? **A:** **NumericToNominal**, because 0 to 3 are already categories; Discretize would merge them into arbitrary ranges.

Common Mistakes

✗ Do not copy. Read for understanding and the viva

- Applying a filter and forgetting to click Apply; the Choose box changes but the data does not.
- Discretising with the wrong `attributeIndices` after an earlier Remove shifted the indices by one.
- Running `Discretize` on the class attribute of a numeric-class file and then wondering why regression is no longer offered.
- Filling missing values with `ReplaceMissingValues` before splitting into train and test sets, which leaks the test mean into training.
- Comparing your `Resample` rows with a classmate's and calling one of them wrong; different seeds give different rows.

Formula Sheet

✗ Do not copy. Read for understanding and the viva

Association rules

For a rule $X \Rightarrow Y$ over N transactions:

$$\text{support}(X \Rightarrow Y) = \frac{|X \cup Y|}{N}, \quad \text{confidence}(X \Rightarrow Y) = \frac{|X \cup Y|}{|X|}, \quad \text{lift}(X \Rightarrow Y) = \frac{\text{confidence}(X \Rightarrow Y)}{\text{support}(Y)}$$

WEKA's Apriori starts at the upper bound of minimum support and lowers it by the delta each pass until the requested number of rules is found or the lower bound is reached.

Entropy, information gain and Gini

For a set S with class proportions $p_1, \dots, p_c$:

$$H(S) = - \sum_{i=1}^c p_i \log_2 p_i, \quad \text{Gain}(S, A) = H(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} H(S_v), \quad \text{Gini}(S) = 1 - \sum_{i=1}^c p_i^2$$

ID3 splits on the attribute with the highest gain; J48 (C4.5) uses the gain ratio

$\text{Gain}(S, A) / \text{SplitInfo}(S, A)$ where $\text{SplitInfo}(S, A) = - \sum_v \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$.

Classifier evaluation

From the confusion matrix with true positives TP , false positives FP , false negatives FN , true negatives TN :

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad \text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Kappa compares observed agreement p_o (accuracy) with the agreement expected by chance p_e :

$$\kappa = \frac{p_o - p_e}{1 - p_e}, \quad p_e = \sum_i \frac{(\text{row}_i \text{ total}) (\text{column}_i \text{ total})}{N^2}$$

The ROC curve plots true positive rate $TP / (TP + FN)$ against false positive rate $FP / (FP + TN)$; the area under it (AUC) is 0.5 for guessing and 1.0 for a perfect classifier.

Naive Bayes and k-nearest neighbour

$$P(C | x_1, \dots, x_n) \propto P(C) \prod_{i=1}^n P(x_i | C)$$

k-NN assigns the majority class among the k nearest training records under Euclidean distance

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$

after normalising each attribute to $[0, 1]$ with $x' = (x - x_{\min}) / (x_{\max} - x_{\min})$.

Linear regression

$$\hat{y} = \beta_0 + \beta_1 x, \quad \beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}, \quad \beta_0 = \bar{y} - \beta_1 \bar{x}$$

WEKA reports the correlation coefficient, mean absolute error and root mean squared error $\sqrt{\frac{1}{N} \sum (y_i - \hat{y}_i)^2}$.

Clustering

k-means minimises the within-cluster sum of squared errors over clusters $C_1, \dots, C_k$ with centroids μ_j :

$$SSE = \sum_{j=1}^k \sum_{\mathbf{x} \in C_j} \|\mathbf{x} - \mu_j\|^2, \quad \mu_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$$

Hierarchical (agglomerative) clustering merges the two closest clusters each step; linkage defines closeness: single $\min d(a, b)$, complete $\max d(a, b)$, average $\frac{1}{|A \cup B|} \sum d(a, b)$.

DBSCAN calls a point a core point when at least `minPts` points lie within radius ϵ ; clusters grow from core points, and points reachable from none are noise.

Session Summary

■ Write in lab record

- Question 5: `Bank.arff` created from `bank-data.csv`, `id` removed, `age` and `income` discretised into 3 bins, `children` converted with `NumericToNominal`, before and after table recorded; `bank-nominal.arff` saved for Session 4
- Question 6: `student.arff`, `labor.arff` and `contact-lenses.arff` preprocessed with `ReplaceMissingValues`, `Normalize`, `Discretize`, `Remove`, `NominalToBinary` and `Resample`; `preprocess.py` output pasted for `student.arff`

[✎ Edit this page](#)

[< Previous](#)
Session 1

Session 3 [Next >](#)