

Session 6

Apriori on the zoo dataset and regression

The zoo dataset shows why attribute selection matters for association rules, and how support and the number of rules interact. The session closes with numeric prediction using regression.

Objectives

✕ Do not copy. Read for understanding and the viva

- Complete questions 13 to 14 of the manual: apriori on the zoo dataset and regression
- 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
Q13	Perform the following	Complete
Q14	Demonstrate to predict the Numerical Values in the given Data Set is using Regression...	Complete

Preparation

✕ Do not copy. Read for understanding and the viva

- Deselect `animal` (an identifier) and `legs` (numeric) or discretise `legs`; Apriori will refuse numeric attributes.
- Increase `numRules` in steps (20, 30, 50) and record at which count a rule with `type=mammal` first appears; then lower `upperBoundMinSupport` until it is the top rule.
- For regression use `cpu.arff` or your own numeric dataset with `functions.LinearRegression` and `functions.SimpleLinearRegression`; explain the coefficients with the regression formulas.

Question 13

Problem Statement

■ Write in lab record

Perform the following:

- Use zoo.arff dataset and load it into WEKA. Examine the attributes and make sure you understand their meaning. Are all attributes nominal?
- In the preprocess area, deselect the animal and legs attributes. The animal attribute is the name of the animal, and is not useful for mining. The legs attribute is numeric and cannot be used directly with Apriori. Alternatively, you can try to use the Discretize Filter to discretize the legs attribute.
- After deselecting the attributes, use the Apply Filters button to generate a working relation that removes those attributes. Notice how the working relation changes, and has fewer attributes than the base relation.
- First, try using the Apriori algorithm with the default parameters. Record the generated rules.
- Vary the number of rules generated (click on the command that you are running). Try 20, 30, and so on. Record how many rules you have to generate before generating a rule containing type=mammal.
- Vary the maximum support until a rule containing type=mammal is the top rule generated. Record the maximum support needed.
- Select one generated rule that was interesting to you. Why was it interesting? What does it mean? Check its confidence and support: are they high enough?
- Suggest one improvement to the Apriori implementation in WEKA that would have made this data mining lab easier to accomplish.

Solution

■ Write in lab record

Steps

1. Start WEKA, click **Explorer**, and on the **Preprocess** tab click **Open file....** Pick `data/zoo.arff` inside the WEKA install folder. The status line shows Relation: zoo, Instances: 101, Attributes: 18.
2. Click each attribute name in the Attributes list and read the Type field in the Selected attribute panel. Fifteen attributes (`hair` to `catsize`, minus `legs`) are Nominal with the two labels `false` and `true`; `type` is Nominal with seven labels (mammal, bird, reptile, fish, amphibian, insect, invertebrate); `animal` is Nominal with 100 labels, one per animal (frog appears twice); `legs` is Numeric with the values 0, 2, 4, 5, 6 and 8. So the answer to "are all attributes nominal" is no: `legs` is numeric.
3. Tick the boxes for `animal` (attribute 1) and `legs` (attribute 14) and click **Remove**. In WEKA 3.8 this button replaces the older "Apply Filters" step; the working relation name changes to `zoo-weka.filters.unsupervised.attribute.Remove-R1,14` and Attributes drops to 16. The alternative for `legs` is **Choose**, `filters.unsupervised.attribute.Discretize`, `attributeIndices` 14, `bins` 3, **Apply**, which turns it into three nominal ranges.
4. Open the **Associate** tab, click **Choose**, pick `weka.associations.Apriori` and click **Start**. The text box shows the defaults `-N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1`: 10 rules, confidence metric, minimum confidence 0.9, delta 0.05, upper bound 1.0, lower bound 0.1.
5. Click the text box to open the parameter dialog, set `numRules` to 20, click **OK** and **Start**. Repeat for 30, 50, 100, 200, 500 and 1000, each time reading the "Minimum support" line and scanning the rule list for `type=mammal`.
6. Set `numRules` back to 10 and set `upperBoundMinSupport` to 0.7, 0.6, 0.55, 0.5, 0.45 and 0.4, running each and noting rule 1.
7. The same procedure can be checked outside WEKA with `apriori.py` below, which follows WEKA's algorithm step by step (support starts at the upper bound minus delta, drops by delta per cycle, itemsets above the upper bound are dropped, rules are ranked by confidence then support). On the manual's weather data it reproduces the manual's block exactly: support 0.15, 17 cycles, L(1) to L(4) of 12, 47, 39 and 6, and the same first nine rules.

Program

apriori.py

```

1  #!/usr/bin/env python3
2  """Apriori the way WEKA's weka.associations.Apriori runs it with the confidence metric.
3
4  Support starts at upperBoundMinSupport - delta and drops by delta each cycle until at least
5  numRules rules with confidence  $\geq$  minMetric exist (or lowerBoundMinSupport is reached).
6  Itemsets whose support is above the upper bound are dropped, as in WEKA. Rules are ranked
7  by confidence, ties by support. Output mimics WEKA's "Associator model" block.
8  Run: python3 apriori.py zoo.arff --remove animal,legs --numRules 10 --upper 1.0
9  """
10 import argparse
11 import itertools
12 from collections import Counter
13
14
15 def load_arff(path, remove=()):
16     names, rows, data = [], [], False
17     for line in open(path):
18         line = line.strip()
19         if not line or line.startswith('%') or line.lower().startswith('@relation'):
20             continue
21         if line.lower().startswith('@attribute'):
22             names.append(line.split(None, 2)[1])
23         elif line.lower() == '@data':
24             data = True
25         elif data:
26             rows.append([v.strip() for v in line.split(',')])
27     keep = [j for j, n in enumerate(names) if n not in remove]
28     return [names[j] for j in keep], [[r[j] for j in keep] for r in rows]
29
30
31 def large_itemsets(rows, nec, nec_max):
32     """Levels L1, L2, ... of itemsets with  $nec \leq \text{support} \leq nec\_max$ . An item is (attr, value)."""
33     counts = Counter((j, v) for r in rows for j, v in enumerate(r))
34     level = {(it,): c for it, c in counts.items() if nec  $\leq$  c  $\leq$  nec_max}
35     levels = []
36     while level:
37         levels.append(level)
38         keys = sorted(level)
39         candsets = set()
40         for a, b in itertools.combinations(keys, 2):
41             if a[:-1] == b[:-1] and a[-1][0]  $\neq$  b[-1][0]: # same prefix, last items on different
42                 c = a + (b[-1],)
43                 if all(c[:i] + c[i + 1:] in level for i in range(len(c))): # every subset frequent
44                     candsets.add(c)
45         cnt = Counter()
46         for r in rows:
47             for c in candsets:
48                 if all(r[j] == v for j, v in c):
49                     cnt[c] += 1
50         level = {c: n for c, n in cnt.items() if nec  $\leq$  n  $\leq$  nec_max}
51     return levels
52
53
54 def rules_from(levels, min_conf):

```

```

55     support = {s: c for lv in levels for s, c in lv.items()}
56     out = []
57     for lv in levels[1:]:
58         for s, c in lv.items():
59             for r in range(1, len(s)):
60                 for cons in itertools.combinations(s, r):
61                     prem = tuple(x for x in s if x not in cons)
62                     conf = c / support[prem]
63                     if conf ≥ min_conf:
64                         out.append((conf, c, prem, support[prem], cons))
65     out.sort(key=lambda t: (-t[0], -t[1]))
66     return out
67
68
69 def run(names, rows, num_rules=10, min_conf=0.9, delta=0.05, upper=1.0, lower=0.1):
70     n = len(rows)
71     nec_max = int(upper * n + 0.5)
72     min_sup = upper - delta          # same double arithmetic as WEKA, so the printed supports m
73     cycles = 0
74     while True:
75         nec = int(min_sup * n + 0.5)
76         levels = large_itemsets(rows, nec, nec_max)
77         rules = rules_from(levels, min_conf)
78         cycles += 1
79         if len(rules) ≥ num_rules or min_sup - delta < lower - 1e-12:
80             break
81         min_sup -= delta
82     return min_sup, nec, cycles, levels, rules
83
84
85 def fmt(items, names):
86     return ' '.join(f"{names[j]}={v}" for j, v in items)
87
88
89 def main():
90     ap = argparse.ArgumentParser()
91     ap.add_argument('arff')
92     ap.add_argument('--remove', default='', help='comma separated attribute names to drop')
93     ap.add_argument('--numRules', type=int, default=10)
94     ap.add_argument('--minMetric', type=float, default=0.9)
95     ap.add_argument('--delta', type=float, default=0.05)
96     ap.add_argument('--upper', type=float, default=1.0)
97     ap.add_argument('--lower', type=float, default=0.1)
98     ap.add_argument('--find', default='', help='report the rank of the first rule mentioning thi
99     a = ap.parse_args()
100
101     names, rows = load_arff(a.arff, set(filter(None, a.remove.split(','))))
102
103     min_sup, nec, cycles, levels, rules = run(names, rows, a.numRules, a.minMetric, a.delta, a.u
104
105     print(f"Scheme:          weka.associations.Apriori -N {a.numRules} -T 0 -C {a.minMetric} -D {a.
106
107     print(f"Instances:      {len(rows)}\nAttributes:   {len(names)}\n")

```

```

10
4     print("Apriori\n=====\n")
10
5     print(f"Minimum support: {min_sup:.2f} ({nec} instances)")
10
6     print(f"Minimum metric <confidence>: {a.minMetric}")
10
7     print(f"Number of cycles performed: {cycles}\n")
10
8     print("Generated sets of large itemsets:\n")
10
9     for i, lv in enumerate(levels, 1):
11
10         print(f"Size of set of large itemsets L({i}): {len(lv)}")
11
1     print(f"\nBest rules found:\n")
11
2     for i, (conf, c, prem, ps, cons) in enumerate(rules[:a.numRules], 1):
11
3         print(f"{i:2d}. {fmt(prem, names)} {ps} ==> {fmt(cons, names)} {c}    conf:({conf:.2g})"
11
4     if a.find:
11
5         j, v = a.find.split('=')
11
6         hit = next((i for i, r in enumerate(rules, 1) if (names.index(j), v) in r[2] + r[4]), No
11
7         print(f"\nrules with confidence ≥ {a.minMetric} at this support: {len(rules)}")
11
8         print(f"first rule mentioning {a.find}: rank {hit}" if hit else f"no rule mentions {a.fi
11
9
12
0
12
1     if __name__ == '__main__':
12
2         main()

```

The data file is the 101-animal zoo data as shipped with WEKA; the column totals (43 hair, 20 feathers, 59 eggs, 41 milk, 83 backbone, 41 mammals, 20 birds and so on) were checked against the UCI summary before use.

zoo.arff (first rows)

```

1 @relation zoo
2 @attribute animal {aardvark,antelope,bass,...}
3 @attribute hair {false,true}
4 ...
5 @attribute legs numeric
6 ...
7 @attribute type {mammal,bird,reptile,fish,amphibian,insect,invertebrate}
8 @data
9 aardvark,true,false,false,true,false,false,true,true,true,true,false,false,4,false,false,true,ma
10 antelope,true,false,false,true,false,false,false,true,true,true,false,false,4,true,false,true,ma
11 bass,false,false,true,false,false,true,true,true,true,false,false,true,0,true,false,false,fish

```

Run: `python3 apriori.py zoo.arff --remove animal,legs --find type=mammal` (the `--find` flag reports the rank of the first rule mentioning the item).

Output

Default parameters (this is the block WEKA prints under "Associator model (full training set)"; every number below was computed by the script on the same data):

```

1 Scheme:      weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
2 Instances:   101
3 Attributes:  16
4
5 Apriori
6 =====
7
8 Minimum support: 0.70 (71 instances)
9 Minimum metric <confidence>: 0.9
10 Number of cycles performed: 6
11
12 Generated sets of large itemsets:
13
14 Size of set of large itemsets L(1): 8
15 Size of set of large itemsets L(2): 13
16 Size of set of large itemsets L(3): 2
17
18 Best rules found:
19
20 1. venomous=false tail=true 71  $\implies$  backbone=true 71    conf:(1)
21 2. tail=true 75  $\implies$  backbone=true 74    conf:(0.99)
22 3. backbone=true tail=true 74  $\implies$  venomous=false 71    conf:(0.96)
23 4. backbone=true 83  $\implies$  venomous=false 79    conf:(0.95)
24 5. breathes=true 80  $\implies$  fins=false 76    conf:(0.95)
25 6. airborne=false 77  $\implies$  feathers=false 73    conf:(0.95)
26 7. tail=true 75  $\implies$  venomous=false 71    conf:(0.95)
27 8. breathes=true venomous=false 75  $\implies$  fins=false 71    conf:(0.95)
28 9. tail=true 75  $\implies$  backbone=true venomous=false 71    conf:(0.95)
29 10. breathes=true 80  $\implies$  venomous=false 75    conf:(0.94)

```

Varying `numRules` (upper bound 1.0):

numRules	Minimum support reached	Cycles	Rules available with confidence 0.9 or more	Rule containing type=mammal
10	0.70 (71 instances)	6	18	none
20	0.65 (66)	7	24	none
30	0.60 (61)	8	63	none
50	0.60 (61)	8	63	none
100	0.55 (56)	9	159	none
200	0.50 (50)	10	365	none
500	0.45 (45)	11	753	none
1000	0.40 (40)	12	5669	first at rank 217: <code>milk=true</code> <code>41 ⇒ type=mammal 41</code> <code>conf:(1)</code>

Recorded answer: 1000 rules (any value above 753 works, because that is what forces support down to 0.40).

Varying `upperBoundMinSupport` (numRules back to 10):

upperBoundMinSupport	Minimum support reached	Rule 1
1.0	0.70	<code>venomous=false tail=true 71 ⇒ backbone=true 71</code>
0.7	0.50	<code>hair=false eggs=true 54 ⇒ milk=false 54</code>
0.6	0.50	<code>hair=false eggs=true 54 ⇒ milk=false 54</code>
0.55	0.40	<code>type=mammal 41 ⇒ milk=true 41</code>
0.5	0.40	<code>type=mammal 41 ⇒ milk=true 41</code>
0.45	0.40	<code>type=mammal 41 ⇒ milk=true 41</code>
0.4	0.20	<code>type=bird 20 ⇒ toothed=false 20</code> (mammals gone: 41 instances is above the cap of 40)

Recorded answer: an upper bound of 0.55 is the largest value that makes a `type=mammal` rule the top rule. The run at 0.55:

```

1 Minimum support: 0.40 (40 instances)
2 Minimum metric <confidence>: 0.9
3 Number of cycles performed: 3
4
5 Generated sets of large itemsets:
6
7 Size of set of large itemsets L(1): 8
8 Size of set of large itemsets L(2): 3
9 Size of set of large itemsets L(3): 1
10
11 Best rules found:
12
13 1. type=mammal 41  $\Rightarrow$  milk=true 41    conf:(1)
14 2. milk=true 41  $\Rightarrow$  type=mammal 41    conf:(1)
15 3. eggs=false type=mammal 40  $\Rightarrow$  milk=true 40    conf:(1)
16 4. eggs=false milk=true 40  $\Rightarrow$  type=mammal 40    conf:(1)
17 5. milk=true 41  $\Rightarrow$  eggs=false 40    conf:(0.98)
18 6. type=mammal 41  $\Rightarrow$  eggs=false 40    conf:(0.98)
19 7. milk=true type=mammal 41  $\Rightarrow$  eggs=false 40    conf:(0.98)
20 8. type=mammal 41  $\Rightarrow$  eggs=false milk=true 40    conf:(0.98)
21 9. milk=true 41  $\Rightarrow$  eggs=false type=mammal 40    conf:(0.98)
22 10. eggs=false 42  $\Rightarrow$  milk=true 40    conf:(0.95)

```

WEKA orders rules of equal confidence and equal support in the order it generated them, so rules that tie on both numbers (for example rules 1 and 2 above) may swap places in your run; everything else matches.

Explanation

- Why the default run never mentions mammals.** Apriori starts at support 0.95 and lowers it by 0.05 per cycle until it has 10 rules. On zoo it stops after 6 cycles at 0.70, because the very common values (`venomous=false` 93 animals, `backbone=true` 83, `breathes=true` 80, `tail=true` 75) already give 18 rules with confidence at least 0.9. Mammals are 41 of 101 animals, support 0.41, so no itemset containing `type=mammal` can be large until the threshold reaches 0.40. Getting there needs more than 753 rules, and even then the 41-instance mammal rules rank behind 216 rules with equal confidence and higher support. The `numRules` knob is a bad tool for this question; the support cap is the right one.
- What the upper bound really does.** WEKA drops every item whose support is above `upperBoundMinSupport`. At 0.55 (cap 56 instances) the frequent boring values disappear and the largest surviving block is the 41 mammals, so `type=mammal 41  $\Rightarrow$  milk=true 41` with confidence 1 becomes rule 1. At 0.4 the cap is 40 instances, which is below 41, so mammals vanish and the 20 birds take over.
- Interesting rule:** `eggs=false 42  $\Rightarrow$  milk=true 40 conf:(0.95)`, support $40/101 = 0.40$. It reads “animals that do not lay eggs give milk”, the biological definition of a mammal. The two exceptions are the scorpion and the seasnake (live young, no milk). The reverse rule `milk=true 41  $\Rightarrow$  eggs=false 40` is 0.98 because of the platypus, the one egg-laying mammal in the data. Confidence 0.95 and support 0.40 are both high for a 16-attribute data set, and the rule is more informative than the confidence-1 rules above it, which only restate that most animals are not venomous.
- One improvement.** Apriori has no way to say “only show rules whose right-hand side is `type`” apart from the `car` (class association rules) switch, and no way to exclude an attribute from rules without removing it from the data. A “must contain attribute” filter, plus a search box in the output panel, would have replaced the eight `numRules` runs with one. A note in the parameter tooltip that `upperBoundMinSupport` also removes items above it would have made the second part obvious.

Question 14

Problem Statement

■ Write in lab record

Demonstrate to predict the Numerical Values in the given Data Set is using Regression Methods.

Solution

■ Write in lab record

Steps

1. **Preprocess, Open file...**, `data/cpu.arff` : 209 machines, 7 numeric attributes (`MYCT` , `MMIN` , `MMAx` , `CACH` , `CHMIN` , `CHMAX`) and the numeric class `class` (published relative performance). A numeric class is what makes this a regression problem: the Classify tab greys out every classifier that needs a nominal class.
2. **Classify** tab, **Choose**, `functions.LinearRegression` . Leave `attributeSelectionMethod` at M5 method and `ridge` at $1.0E-8$. Under Test options pick **Cross-validation, Folds 10**, click **Start**.
3. **Choose**, `functions.SimpleLinearRegression` , **Start**. This model uses only the one attribute that gives the lowest squared error.
4. To see the model on the training data rather than cross-validated, pick **Use training set** and **Start** again; the model text is the same, the error figures are lower.
5. Repeat steps 2 and 3 on your own two-column file `study_marks.arff` below, then compute the slope and intercept by hand with the formula sheet and compare. Right-click the result, **Visualize classifier errors**, to see the residuals as crosses around the fitted line.

Program

■ Lab record: every tab is one file of the answer. Write all of them.

`study_marks.arff`

study_marks.arff

```

1 % study_marks.arff : 10 students, hours studied per week vs marks out of 100
2 @relation study_marks
3
4 @attribute hours numeric
5 @attribute marks numeric
6
7 @data
8 2,35
9 4,48
10 5,50
11 6,58
12 8,66
13 9,70
14 10,74
15 12,80
16 14,89
17 16,94

```

Output

LinearRegression on cpu.arff prints the model WEKA fits on all 209 rows (CHMIN is dropped by the M5 attribute selection) followed by the cross-validation summary. The model text is what WEKA prints for this data set; record the error figures from your own run.

```

1 == Classifier model (full training set) ==
2
3 Linear Regression Model
4
5 class =
6
7      0.0491 * MYCT +
8      0.0152 * MMIN +
9      0.0056 * MMAX +
10     0.6298 * CACH +
11     1.4599 * CHMAX +
12    -56.075
13
14 == Cross-validation ==
15 == Summary ==
16
17 Correlation coefficient           0.9012
18 Mean absolute error             41.0886
19 Root mean squared error        69.556
20 Relative absolute error         42.6943 %
21 Root relative squared error     43.2421 %
22 Total Number of Instances      209

```

SimpleLinearRegression on cpu.arff prints Linear regression on MMAX and a single line class = b1 * MMAX + b0 : it picked maximum main memory as the one attribute that explains performance best, at the cost of a higher error than the

six-attribute model.

Hand computation on `study_marks.arff` (`python3 regression.py study_marks.arff` , run for real):

```

1  x    y  x-xbar  y-ybar  product  (x-xbar)^2
2  2    35  -6.60  -31.40  207.24   43.56
3  4    48  -4.60  -18.40   84.64   21.16
4  5    50  -3.60  -16.40   59.04   12.96
5  6    58  -2.60  -8.40   21.84    6.76
6  8    66  -0.60  -0.40    0.24    0.36
7  9    70   0.40   3.60    1.44    0.16
8 10    74   1.40   7.60   10.64    1.96
9 12    80   3.40  13.60   46.24   11.56
10 14    89   5.40  22.60  122.04   29.16
11 16    94   7.40  27.60  204.24   54.76
12
13 n = 10   x-bar = 8.60   y-bar = 66.40
14 Sxy = 757.60   Sxx = 182.40   Syy = 3192.40
15 slope      b1 = Sxy / Sxx      = 4.1535
16 intercept b0 = y-bar - b1 x-bar = 30.6798
17 model: y = 4.1535 * x + 30.6798
18
19 Correlation coefficient    0.9928
20 Mean absolute error       1.8184
21 Root mean squared error   2.1378
22
23   x actual predicted  error
24   2     35     38.99  -3.99
25   4     48     47.29   0.71
26   5     50     51.45  -1.45
27   6     58     55.60   2.40
28   8     66     63.91   2.09
29   9     70     68.06   1.94
30  10     74     72.21   1.79
31  12     80     80.52  -0.52
32  14     89     88.83   0.17
33  16     94     97.14  -3.14
34 prediction for x = 11: 76.37

```

WEKA's `SimpleLinearRegression` on the same file, evaluated on the training set, prints `marks = 4.15 * hours + 30.68` , correlation 0.9928, MAE 1.8184 and RMSE 2.1378: the same numbers, because the model has a closed form and there is nothing random to differ.

Explanation

- The formula sheet gives $\beta_1 = \sum (x_i - \bar{x}) (y_i - \bar{y}) / \sum (x_i - \bar{x})^2$. The script prints the two sums, 757.60 and 182.40, so the slope is 4.1535 marks per extra hour of study. The intercept $\beta_0 = \bar{y} - \beta_1 \bar{x} = 66.40 - 4.1535 \times 8.60 = 30.68$ is the predicted mark at zero hours.
- Least squares chooses the line that minimises the sum of squared residuals; a side effect is that the residuals sum to zero, which the script asserts.

- The correlation coefficient 0.9928 is $S_{xy}/\sqrt{S_{xx}S_{yy}}$; its square, 0.986, is the fraction of the variance in marks the line explains.
- `LinearRegression` does the same thing with several attributes at once: it solves the normal equations for one coefficient per attribute (the ridge value keeps the matrix invertible) and the M5 method drops attributes that do not reduce the error, which is why `CHMIN` is missing from the cpu model.
- The error figures under cross-validation are larger than on the training set because each fold is predicted by a model that never saw it. Compare RMSE 69.6 with the class standard deviation of about 161 on cpu: the model removes roughly 57 percent of the spread.

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.

Viva Questions

✗ Do not copy. Read for understanding and the viva

Q: Why must `animal` be removed before Apriori? **A:** It has a different value on every row, so every itemset containing it has support 1/101 and nothing useful can be frequent; it only wastes candidate generation.

Q: Why cannot Apriori use the numeric `legs` attribute? **A:** Apriori counts exact value matches; a numeric attribute has no finite set of labels to count. Discretizing it into ranges makes it nominal.

Q: What does "Minimum support: 0.70 (71 instances)" mean? **A:** The last support threshold Apriori reached before it had enough rules: an itemset needs at least 71 of the 101 rows to be large.

Q: What are the numbers before and after the arrow in a rule? **A:** The count of rows matching the left-hand side and the count matching both sides; their ratio is the confidence.

Q: Why did lowering `upperBoundMinSupport` bring `type=mammal` to the top? **A:** WEKA drops items with support above the upper bound, so the very common values that were filling the rule list disappear and the 41-row mammal block becomes the largest one left.

Q: What is the difference between `LinearRegression` and `SimpleLinearRegression`? **A:** The first fits one coefficient per attribute; the second picks the single attribute with the lowest squared error and fits a line to it.

Q: Why is the correlation coefficient reported for regression instead of accuracy? **A:** Predictions are numbers, not labels, so “correct or wrong” does not apply; correlation, MAE and RMSE measure how close the numbers are.

Q: What happens to a linear regression when an attribute is a linear combination of others? **A:** The normal equations become singular; WEKA’s ridge parameter adds a tiny constant to the diagonal so a solution still exists.

Common Mistakes

✖ Do not copy. Read for understanding and the viva

- Running Apriori with `animal` still present and reporting rules with 1 instance of support.
- Reading the “Minimum support” line as a parameter you set; it is the level the algorithm reached.
- Increasing `numRules` to hundreds and not noticing that the mammal rule is ranked in the 200s, then reporting that it never appeared.
- Setting `upperBoundMinSupport` below 0.41 and concluding that there is no mammal rule at all.
- Evaluating a regression model on the training set only and reporting the (optimistic) errors as if they were cross-validated.
- Writing the slope formula with the sums swapped; the denominator is the sum of squares of x only.

Session Summary

■ Write in lab record

- Question 13: zoo.arff loaded, `animal` and `legs` removed, Apriori with defaults (support 0.70, 10 rules, no mammals), `numRules` swept to 1000 (mammal rule first at rank 217 once support reaches 0.40), `upperBoundMinSupport` lowered to 0.55 (mammal rule at rank 1), one rule interpreted, one improvement proposed; `apriori.py` and `zoo.arff` recorded.
- Question 14: `LinearRegression` and `SimpleLinearRegression` on cpu.arff, and the slope and intercept of `study_marks.arff` computed by hand with `regression.py` (marks = 4.1535 hours + 30.68, correlation 0.9928).

✎ Edit this page

< Previous
Session 5

Next >
Session 7