

Session 7

Classification: logistic regression, decision tree, Naive Bayes, k-NN, SVM

Five classifiers on the same three datasets make the trade-offs visible: trees are readable, Naive Bayes is fast and needs little data, k-NN needs normalised attributes, SVM handles many attributes.

Objectives

✗ Do not copy. Read for understanding and the viva

- Complete questions 15 to 16 of the manual: classification: logistic regression, decision tree, naive bayes, k-nn, svm
- 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
Q15	Demonstrate the classification rule process on the student.arff, employee.arff and...	Complete
Q16	Demonstrate the classification rule process on the student.arff, employee.arff and...	Complete

Preparation

✗ Do not copy. Read for understanding and the viva

- WEKA names: `functions.Logistic`, `trees.J48`, `bayes.NaiveBayes`, `lazy.IBk` (k-NN, set K), `functions.SMO` (SVM).

- Use 10-fold cross-validation for every run so results are comparable; note accuracy, kappa, and the confusion matrix for each.
- Build student.arff and employee.arff with a nominal class attribute and at least 30 rows each, or the classifiers will not have enough to learn.

Question 15

Problem Statement

■ Write in lab record

Demonstrate the classification rule process on the student.arff, employee.arff and labor.arff datasets using the following algorithms:

1. Logistic Regression
2. Decision Tree
3. Naive Bayes

Solution

■ Write in lab record

Steps

Every number on this page and in Sessions 8 to 10 comes from the two 30-row data sets defined here (`public/code/mcs1-223/section-2/session-7/`): three numeric attributes, one nominal attribute and a two-valued class each, with no missing values, so the Python check script can reproduce WEKA's evaluation. The Session 1 files (`session-1/student.arff` , `session-1/employee.arff`) have a richer layout with an identifier column, more attributes and a few `?` cells; the WEKA procedure is identical on them, but first remove `sid` or `eid` on the Preprocess tab (an identifier makes every classifier overfit) and expect different numbers. `labor.arff` ships with WEKA (57 instances, 16 attributes plus the class `class` with values `bad` and `good` , many `?` missing values).

1. **Explorer, Preprocess, Open file...**, `student.arff` . Check the status line: 30 instances, 5 attributes, and click `result` in the attribute list to confirm the class distribution (19 pass, 11 fail). Repeat later for `employee.arff` (16 yes, 14 no) and `labor.arff` (37 good, 20 bad).

2. **Classify** tab. Under Test options select **Cross-validation**, Folds 10. Leave the class chooser at the last attribute.
3. **Choose**, `functions.Logistic` (defaults: `ridge` `1.0E-8`, `maxIts` `-1`). **Start**. Record from the output: the coefficient table under "Classifier model", then under "Stratified cross-validation" the Correctly Classified Instances line, the Kappa statistic and the Confusion Matrix.
4. **Choose**, `trees.J48` (defaults: `confidenceFactor` `0.25`, `minNumObj` `2`). **Start**. Right-click the entry in the Result list and pick **Visualize tree** to see the tree drawn; the same tree is printed as text with leaf counts such as `pass (16.0)` or `pass (4.0/1.0)` (instances reaching the leaf, and how many of them are misclassified).
5. **Choose**, `bayes.NaiveBayes` (defaults). **Start**. The model block lists, per class, the mean and standard deviation of every numeric attribute and the counts of every nominal value.
6. Repeat steps 3 to 5 for `employee.arff` and `labor.arff`; nine runs in all. Keep the Result list entries: the next question adds two more classifiers to the same list for a side-by-side comparison.
7. To check the numbers without WEKA run `python3 classify_cv.py student.arff employee.arff`. The script reimplements the five classifiers of this session and the next in about 250 lines of standard-library Python and prints the same summary and confusion matrix layout as WEKA.

Program

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

student.arff

student.arff

```
1 % student.arff : 30 students, class = result
2 % pass needs internal  $\geq$  12 and attendance  $\geq$  60 (one noisy row on purpose)
3 @relation student
4
5 @attribute hours numeric
6 @attribute attendance numeric
7 @attribute internal numeric
8 @attribute stream {science,commerce,arts}
9 @attribute result {pass,fail}
10
11 @data
12 12,85,24,science,pass
13 15,92,27,science,pass
14 8,70,15,commerce,pass
15 10,78,19,arts,pass
16 18,95,29,science,pass
17 6,65,13,commerce,pass
18 14,88,22,arts,pass
19 9,72,16,science,pass
20 11,80,20,commerce,pass
21 16,90,26,arts,pass
22 7,68,14,science,pass
23 13,84,23,commerce,pass
24 10,75,18,arts,pass
25 17,93,28,science,pass
26 9,74,17,commerce,pass
27 12,82,21,arts,pass
28 8,66,12,science,pass
29 15,89,25,commerce,pass
30 11,77,19,arts,pass
31 3,45,6,science,fail
32 4,50,8,commerce,fail
33 2,40,5,arts,fail
34 5,55,9,science,fail
35 6,58,11,commerce,fail
36 3,48,7,arts,fail
37 4,52,10,science,fail
38 7,60,11,commerce,fail
39 10,70,14,arts,fail
40 12,58,20,science,fail
41 6,62,10,commerce,fail
```

Output

Decision tree on `student.arff` (`classify_cv.py` , run for real; J48's pruning step would merge the two small leaves under `internal ≤ 14.5` into one leaf `pass (4.0/1.0)` , everything else is the same tree):

```

1  ≡≡≡ Tree on the full training set ≡≡≡
2  attendance ≤ 63.5: fail (10.0)
3  attendance > 63.5
4  |   internal ≤ 14.5
5  |   |   hours ≤ 7.5: pass (2.0)
6  |   |   hours > 7.5: pass (2.0/1.0)
7  |   internal > 14.5: pass (16.0)
8
9  ≡≡≡ Tree (ID3/J48, unpruned) : 10-fold cross-validation ≡≡≡
10 Correctly Classified Instances      29    96.6667 %
11 Incorrectly Classified Instances     1     3.3333 %
12 Kappa statistic                      0.9268
13 ROC Area (pass)                     0.9450
14 ≡≡≡ Confusion Matrix ≡≡≡
15      a   b   ← classified as
16    19   0 | a = pass
17     1  10 | b = fail

```

Decision tree on `employee.arff` :

```

1  ≡≡≡ Tree on the full training set ≡≡≡
2  experience ≤ 5.5: no (12.0)
3  experience > 5.5
4  |   experience ≤ 9.5
5  |   |   age ≤ 32.5
6  |   |   |   experience ≤ 6.5: yes (2.0/1.0)
7  |   |   |   experience > 6.5: yes (2.0)
8  |   |   age > 32.5: yes (2.0/1.0)
9  |   experience > 9.5: yes (12.0)

```

Results for the three classifiers of this question, 10-fold cross-validation (student and employee computed by the script; labor values are the ones WEKA 3.8 prints for its bundled file with the default seed, so paste the block from your own run):

Data set	Classifier	Accuracy	Kappa	Confusion matrix (rows = actual)
student	Logistic	86.67 % (26/30)	0.7129	pass: 17 2; fail: 2 9
student	J48	96.67 % (29/30)	0.9268	pass: 19 0; fail: 1 10
student	NaiveBayes	83.33 % (25/30)	0.6479	pass: 16 3; fail: 2 9
employee	Logistic	83.33 % (25/30)	0.6696	yes: 12 4; no: 1 13
employee	J48	86.67 % (26/30)	0.7297	yes: 15 1; no: 3 11
employee	NaiveBayes	86.67 % (26/30)	0.7345	yes: 13 3; no: 1 13
labor	Logistic	about 88 %	record	record
labor	J48	73.68 % (42/57)	0.4415	bad: 14 6; good: 9 28
labor	NaiveBayes	89.47 % (51/57)	record	record

The J48 block WEKA prints for `Labor.arff` :

```

1  == Stratified cross-validation ==
2  == Summary ==
3
4  Correctly Classified Instances      42           73.6842 %
5  Incorrectly Classified Instances    15           26.3158 %
6  Kappa statistic                     0.4415
7  Total Number of Instances          57
8
9  == Confusion Matrix ==
10
11   a   b   ← classified as
12  14   6 |   a = bad
13   9  28 |   b = good

```

Explanation

- **Logistic regression** fits one weight per attribute (nominal attributes become 0/1 indicator columns) and passes the weighted sum through the logistic function to get a probability of the

first class. Its decision boundary is a straight line in attribute space, which suits `employee` (promotion rises with experience and salary together) and `student` reasonably well. The `ridge` value shrinks the weights slightly so the solution exists even when a class is perfectly separable.

- **J48** picks the attribute and threshold with the highest information gain at each node (formula sheet), so the student tree splits first on `attendance ≤ 63.5`, which alone sends all 10 low-attendance students to `fail` with no error, then on `internal`. It scores highest on `student` because the data was generated by exactly such a rule. On `employee` the two noisy rows (experience 6 and 9, not promoted) force the small impure leaves and cost accuracy.
- **Naive Bayes** multiplies the class prior by one Gaussian likelihood per numeric attribute and one frequency ratio per nominal attribute. It has no thresholds, so it is slightly worse on the rule-shaped `student` data and best on `employee`, where the class is a smooth function of two correlated numbers.
- **Kappa** removes the agreement expected by chance. On `student`, chance agreement is $p_e = (19 \cdot 20 + 11 \cdot 10) / 30^2 = 0.544$ for the J48 matrix, so $\kappa = (0.9667 - 0.544) / (1 - 0.544) = 0.927$, exactly the printed value.
- **labor** has only 57 rows and many missing values, so the pruned tree is unstable across folds (73.7 %) while Naive Bayes, which simply skips a missing attribute in the product, is the most accurate of the three.

Question 16

Problem Statement

■ Write in lab record

Demonstrate the classification rule process on the `student.arff`, `employee.arff` and `labor.arff` datasets using the following algorithms:

1. K-Nearest Neighbour
2. SVM

Solution

■ Write in lab record

Steps

1. With `student.arff` still loaded and 10-fold cross-validation selected, **Choose**, `lazy.IBk`. Click the name to open the dialog: `KNN` 1 (default). **Start**. Then set `KNN` to 3 and `distanceWeighting` to `Weight by 1/distance` and **Start** again to see the effect of a larger neighbourhood.
2. **Choose**, `functions.SMO`. Defaults: `c` 1.0, `kernel` PolyKernel with exponent 1.0 (a linear SVM), `filterType` Normalize training data. **Start**. The model block prints one weight per (normalised) attribute and the bias, in the form `-1.2 * (normalized) hours + ... + 0.8`.
3. Repeat both for `employee.arff` and `labor.arff`.
4. For a side-by-side view of all five classifiers, open the **Experimenter** (New, Add new... data set, Add new... algorithm for each of the five, Run, then Analyse with Percent_correct as the comparison field). The Explorer is enough for the record book.

Output

`classify_cv.py` for the two classifiers of this question (run for real; $k = 1$ for IBk, linear kernel and $C = 1$ for the SVM, which is trained with the Pegasos sub-gradient method instead of SMO, so WEKA's numbers can differ by a row or two):

```

1  === IBk (k=1) : 10-fold cross-validation ===          student.arff
2  Correctly Classified Instances      27   90.0000 %
3  Kappa statistic                     0.7805
4  === Confusion Matrix ===
5      a    b    ← classified as
6      18    1 | a = pass
7       2    9 | b = fail
8
9  === SVM (linear, C=1) : 10-fold cross-validation ===  student.arff
10 Correctly Classified Instances      22   73.3333 %
11 Kappa statistic                     0.4030
12 === Confusion Matrix ===
13      a    b    ← classified as
14      16    3 | a = pass
15       5    6 | b = fail
16
17 === IBk (k=1) : 10-fold cross-validation ===          employee.arff
18 Correctly Classified Instances      23   76.6667 %
19 Kappa statistic                     0.5374
20 === Confusion Matrix ===
21      a    b    ← classified as
22      11    5 | a = yes
23       2   12 | b = no
24
25 === SVM (linear, C=1) : 10-fold cross-validation ===  employee.arff
26 Correctly Classified Instances      22   73.3333 %
27 Kappa statistic                     0.4690
28 === Confusion Matrix ===
29      a    b    ← classified as
30      11    5 | a = yes
31       3   11 | b = no

```

All five classifiers together (the summary table the script prints at the end of each data set):

```

1 student.arff
2 classifier          accuracy %   kappa   ROC area
3 NaiveBayes          83.33    0.6479   0.9187
4 IBk (k=1)           90.00    0.7805   0.8828
5 Tree (ID3/J48, unpruned) 96.67    0.9268   0.9450
6 Logistic            86.67    0.7129   0.9139
7 SVM (linear, C=1)   73.33    0.4030   0.8804
8
9 employee.arff
10 classifier          accuracy %   kappa   ROC area
11 NaiveBayes          86.67    0.7345   0.9688
12 IBk (k=1)           76.67    0.5374   0.7723
13 Tree (ID3/J48, unpruned) 86.67    0.7297   0.8683
14 Logistic            83.33    0.6696   0.8973
15 SVM (linear, C=1)   73.33    0.4690   0.8705

```

On `labor.arff` WEKA prints about 82 % for IBk with $k = 1$ and 89.47 % (51/57) for SMO; paste the kappa and confusion matrix from your run.

Explanation

- **k-NN** stores the training rows and, for each test row, takes the majority class of the k closest rows under Euclidean distance on attributes scaled to 0 to 1 (nominal attributes count 1 when different). It needs the scaling because `attendance` ranges over 58 units and `hours` over 16; without it `attendance` would decide everything. With $k = 1$ one noisy neighbour flips a prediction, which is why the `employee` result (76.7 %) is the weakest of the five there: the noisy rows 26 and 27 sit inside the promoted region. $k = 3$ with distance weighting smooths this out.
- **SVM** finds the line (hyperplane) with the largest margin between the classes; `c` sets how many margin violations are tolerated. On `student` the true boundary is an L-shaped rule (`attendance` and `internal` both matter, but as thresholds, not as a weighted sum), so a linear kernel cannot follow it and the SVM comes last at 73.3 %. Switching `kernel` to `RBFKernel` or raising the PolyKernel exponent to 2 lets it bend and recovers most of the loss. On `labor`, whose attributes are mostly ordinal wage figures, a linear boundary is right and SMO is joint best.
- The ROC area column comes from ranking the test rows by the predicted probability of the first class; $k = 1$ gives only 0 or 1 scores, so its ROC area is little more than its accuracy, whereas Naive Bayes and Logistic produce graded scores and rank rows well even when their hard predictions are wrong. Session 8 uses this column.

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 $\frac{1}{|A||B|} \sum d(a, b)$.

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 10-fold cross-validation and not the training set? **A:** Training-set accuracy rewards memorising; cross-validation tests each row with a model that never saw it, so it estimates performance on new data.

Q: What does `pass (4.0/1.0)` on a J48 leaf mean? **A:** Four training instances reach that leaf and one of them is not `pass`.

Q: Why does IBk normalise attributes? **A:** Euclidean distance adds squared differences; an attribute with a large range would swamp the others unless all are scaled to 0 to 1.

Q: Why is Naive Bayes called naive? **A:** It assumes the attributes are conditionally independent given the class so the joint likelihood is a plain product.

Q: What is the `c` parameter of SMO? **A:** The penalty for points on the wrong side of the margin; a large `c` fits the training data more tightly.

Q: What does a kappa of 0 mean? **A:** The classifier agrees with the true labels no more often than random assignment with the same class proportions would.

Q: Which of the five classifiers has no training phase? **A:** IBk; it stores the data and does all the work at prediction time.

Q: How does Logistic regression handle a nominal attribute such as `stream`? **A:** It expands it into one 0/1 indicator column per value and learns a weight for each.

Common Mistakes

✗ Do not copy. Read for understanding and the viva

- Leaving Test options at "Use training set" and reporting 100 percent for J48 and IBk.
- Building `student.arff` with a numeric class such as marks and then wondering why the classifiers are greyed out.
- Using the class attribute as an input by picking the wrong attribute in the class chooser below the Test options.
- Comparing classifiers run with different fold counts or seeds; keep every run at 10 folds, seed 1.
- Reading the confusion matrix with rows and columns swapped; rows are the actual class, columns the predicted one.
- Reporting IBk with $k = 1$ as "the k-NN result" without trying a larger k ; one neighbour is the noisiest setting.

Session Summary

■ Write in lab record

- Question 15: `student.arff` and `employee.arff` (30 rows each, defined here) and `labor.arff` classified with Logistic, J48 and NaiveBayes under 10-fold cross-validation; accuracy, kappa and confusion matrix tabulated, J48 trees recorded.
- Question 16: the same three data sets with IBk (k = 1 and 3) and SMO; five-classifier comparison table per data set from `classify_cv.py` .

 [Edit this page](#)

[< Previous](#)
Session 6

Session 8 [Next >](#)