

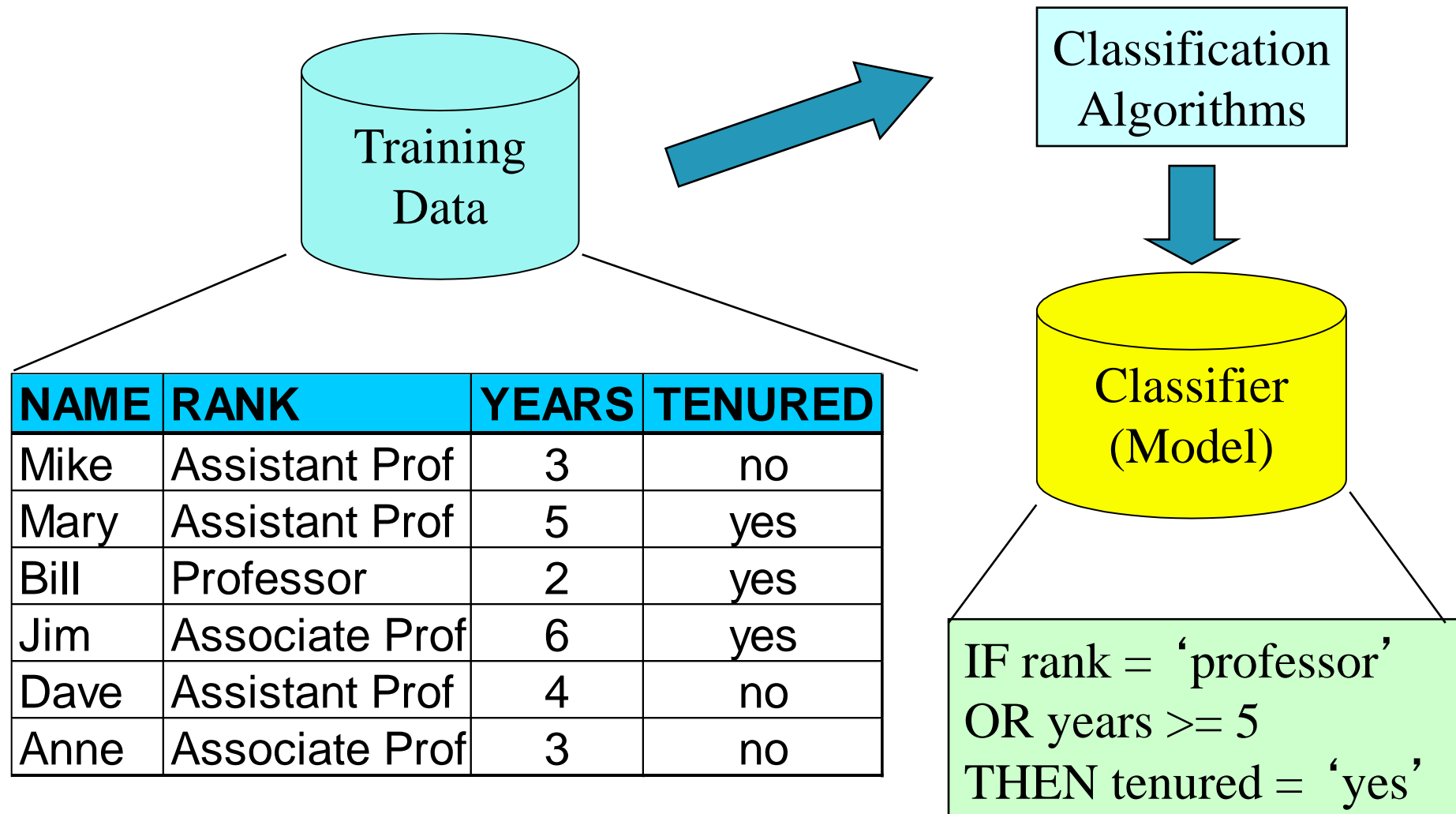
# SE488 Ai Driven Software Development

**Supervised Learning  
Model Evaluation/Prediction**

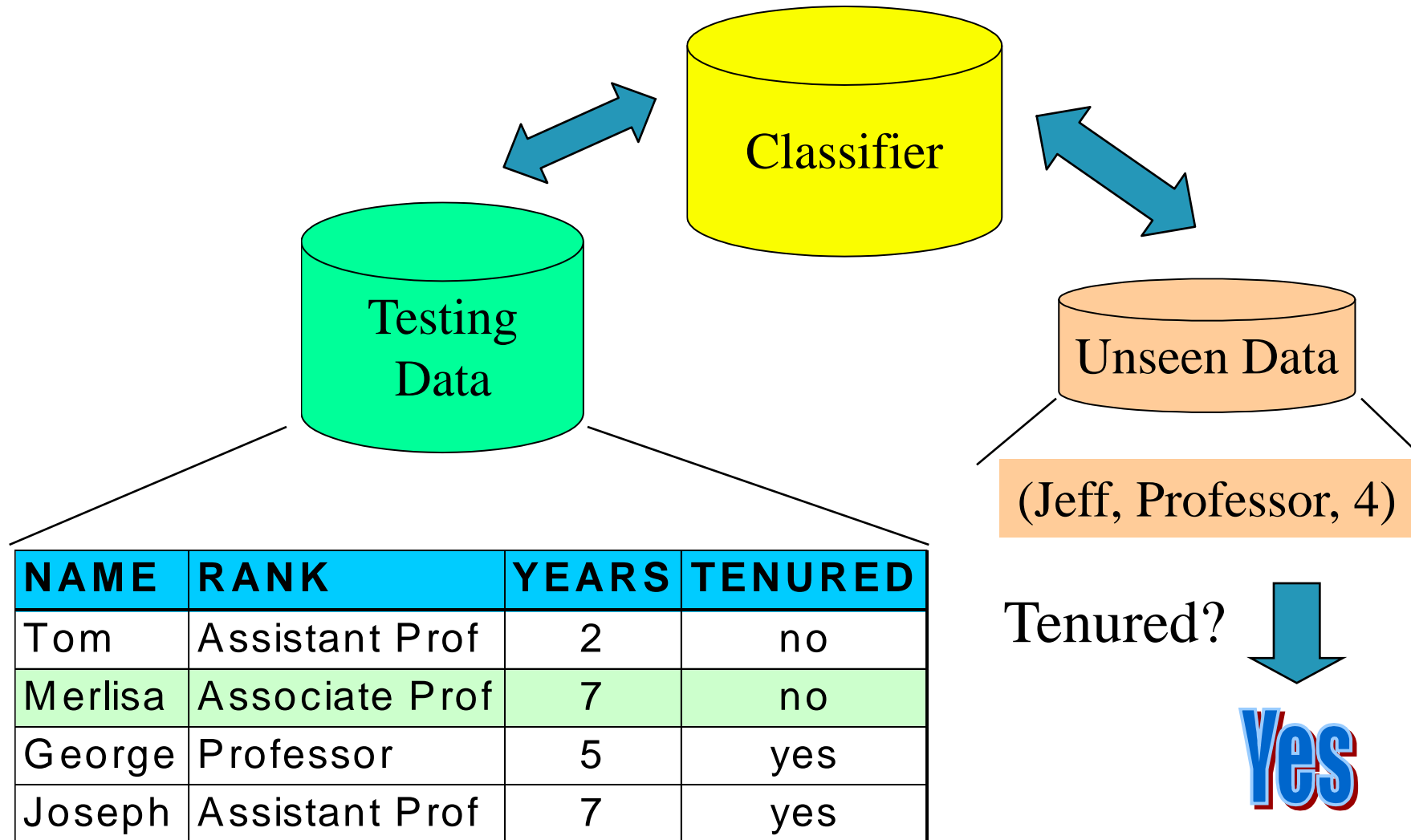
# Classification—A Two-Step Process

- **Model construction:** describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class label attribute**
  - The set of tuples used for model construction is called the **training set**
  - The model is represented as classification rules, decision trees, or mathematical formulae
- **Model usage:** for classifying future or unknown objects
  - Estimate **accuracy** of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to **classify data** tuples whose class labels are not known

# Process (1): Model Construction

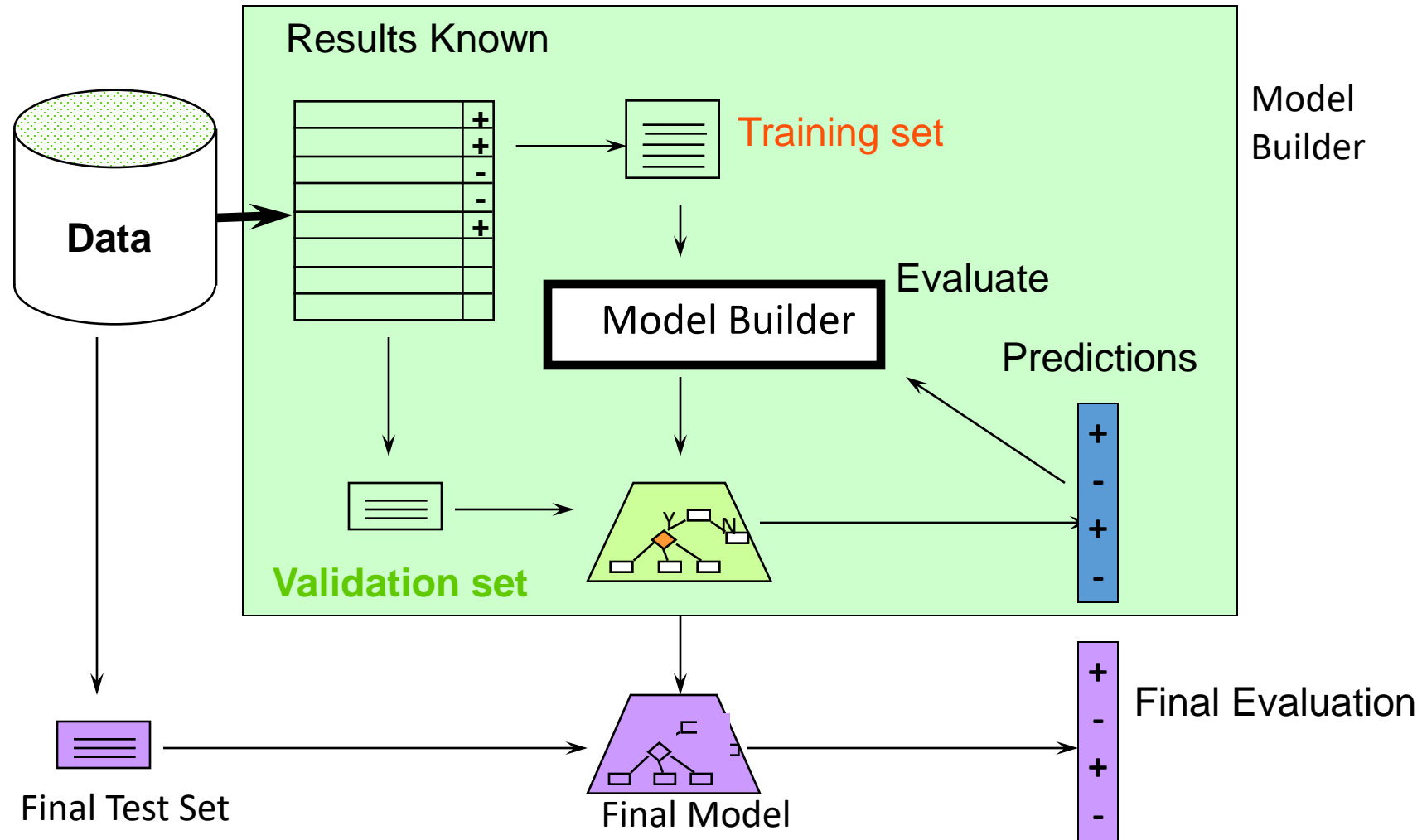


## Process (2): Using the Model in Prediction



- How predictive is the model we learned?
- Low error rates on the training data do not necessarily mean the model will perform well on new data
  - *Q: Why?*
    - A: Because new data will probably not be **exactly** the same as the training data!
- Overfitting – fitting the training data too precisely - usually leads to poor results on new data

# Classification: Train, Validation, Test split



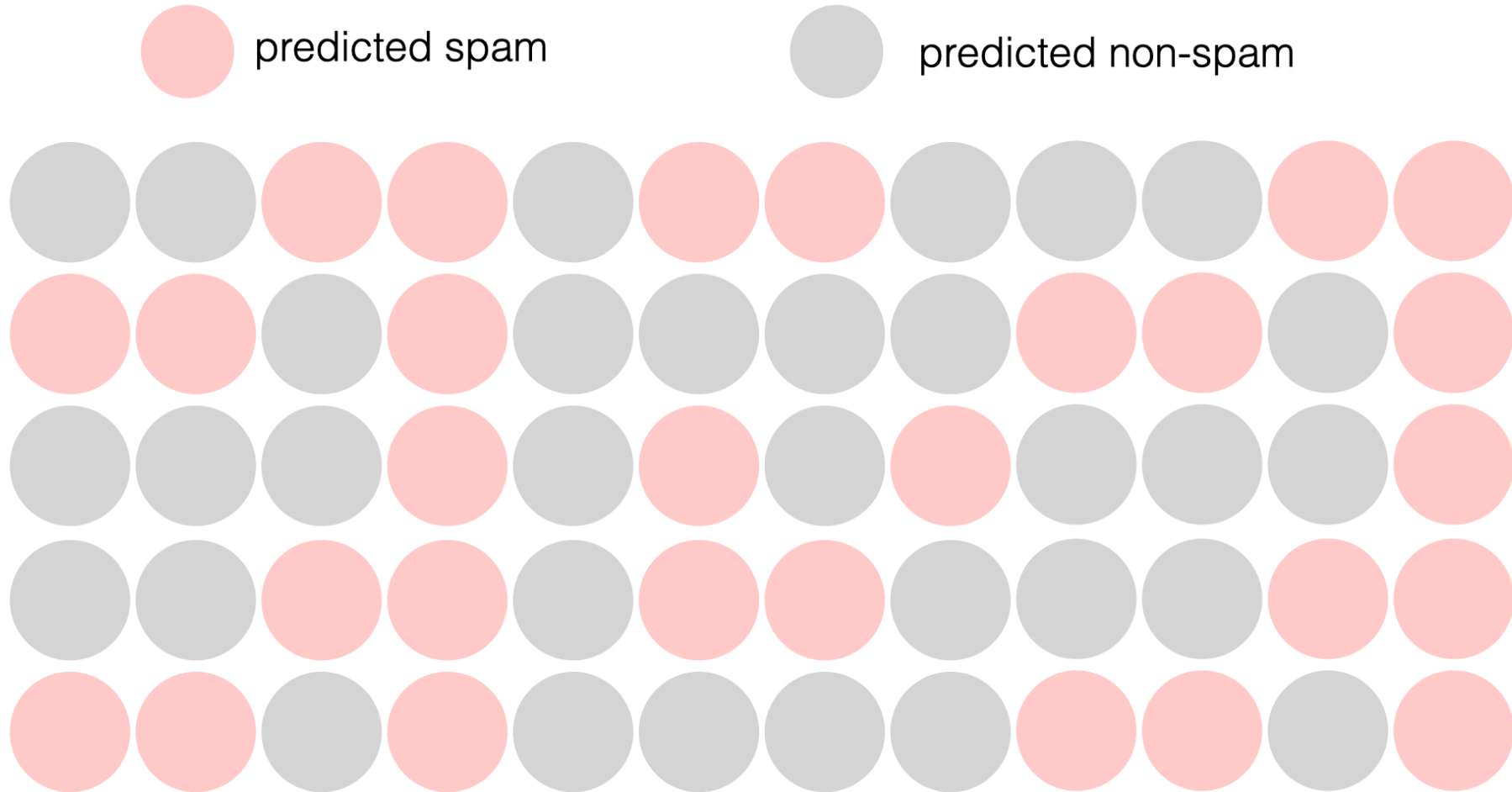
- Evaluation of classifiers

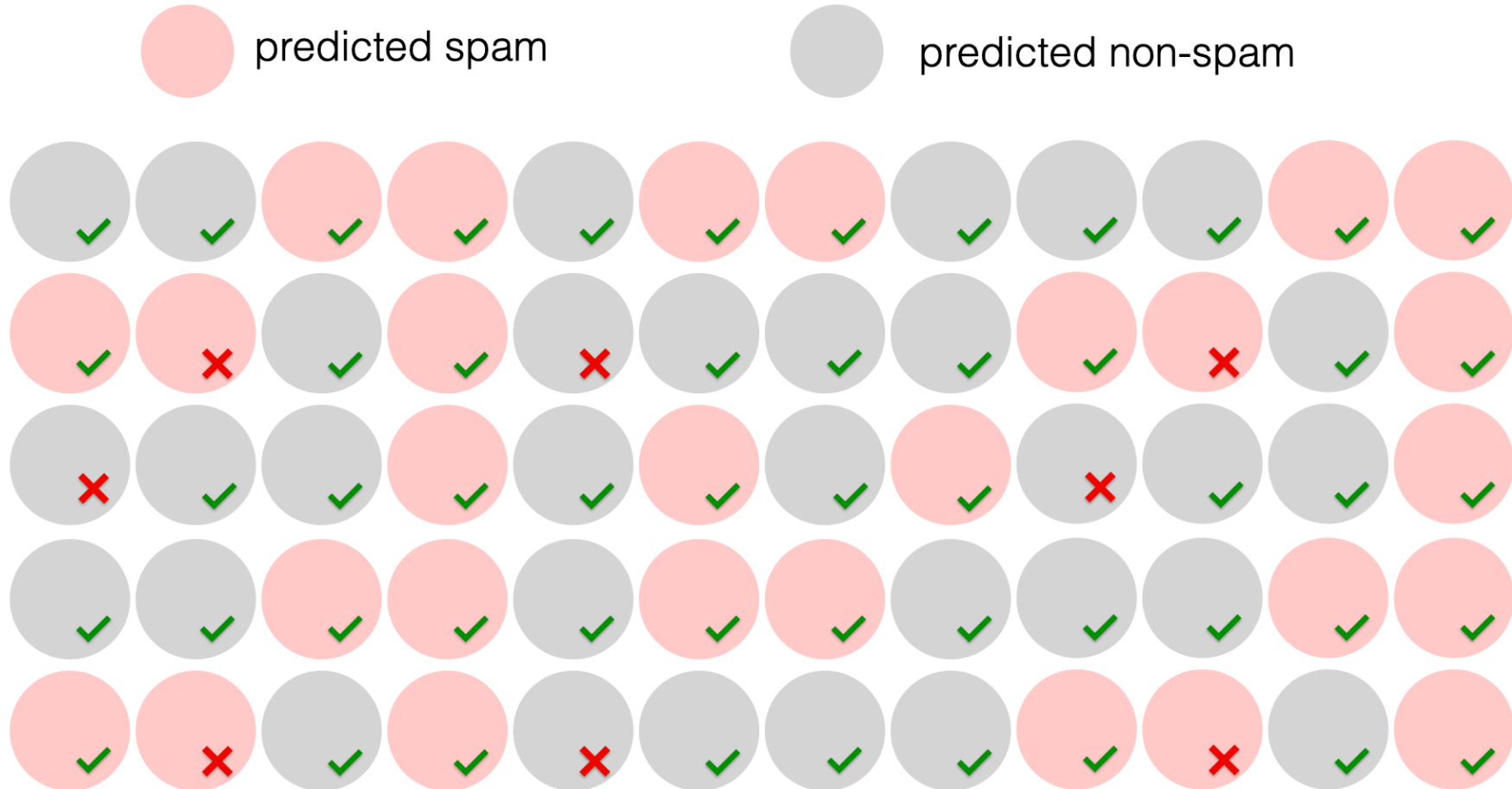
- Accuracy, precision, and recall help evaluate the quality of classification models in machine learning. Each metric reflects a different aspect of the model quality, and depending on the use case, you might prefer one or another.

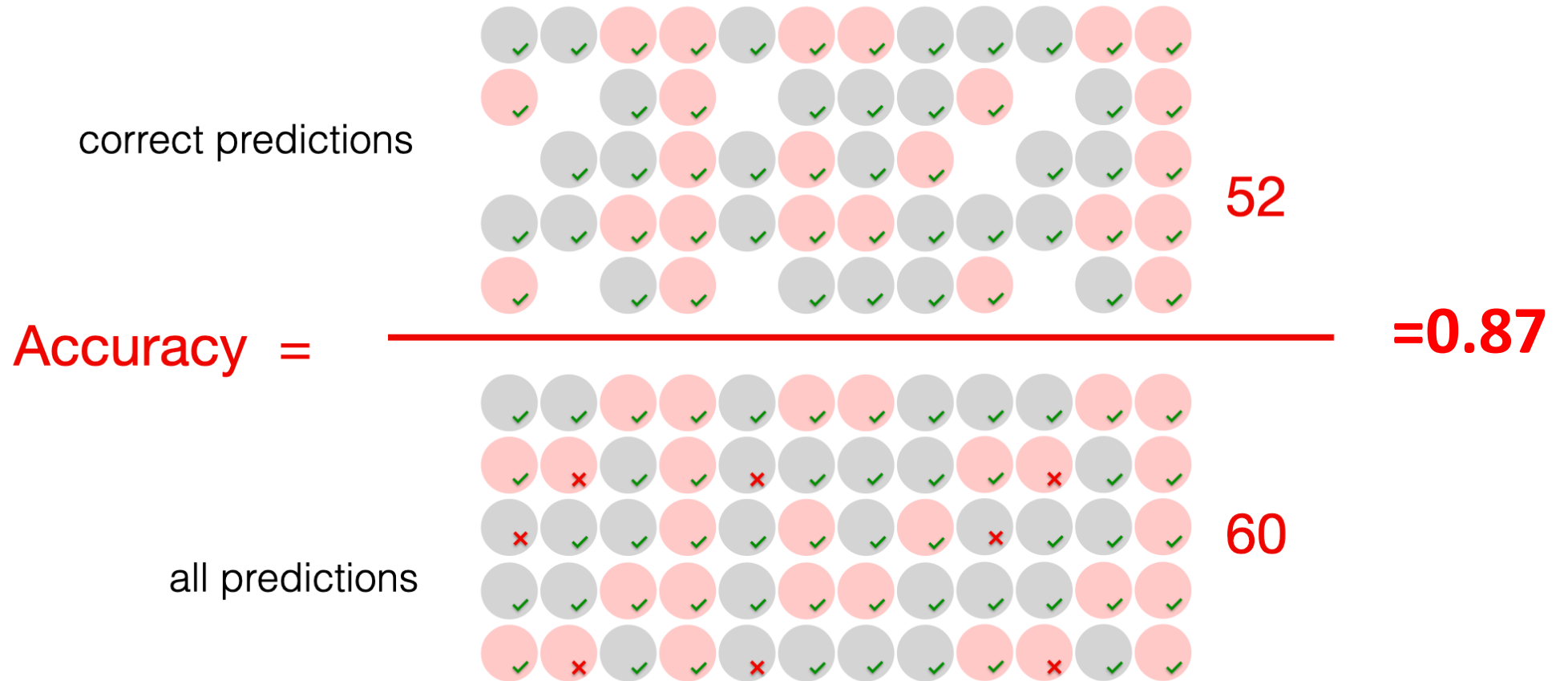


- **Accuracy** shows how often a classification ML model is correct **overall**.
- How often a machine learning model correctly predicts the outcome
- calculate accuracy by dividing the number of correct predictions by the total number of predictions.

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}}$$



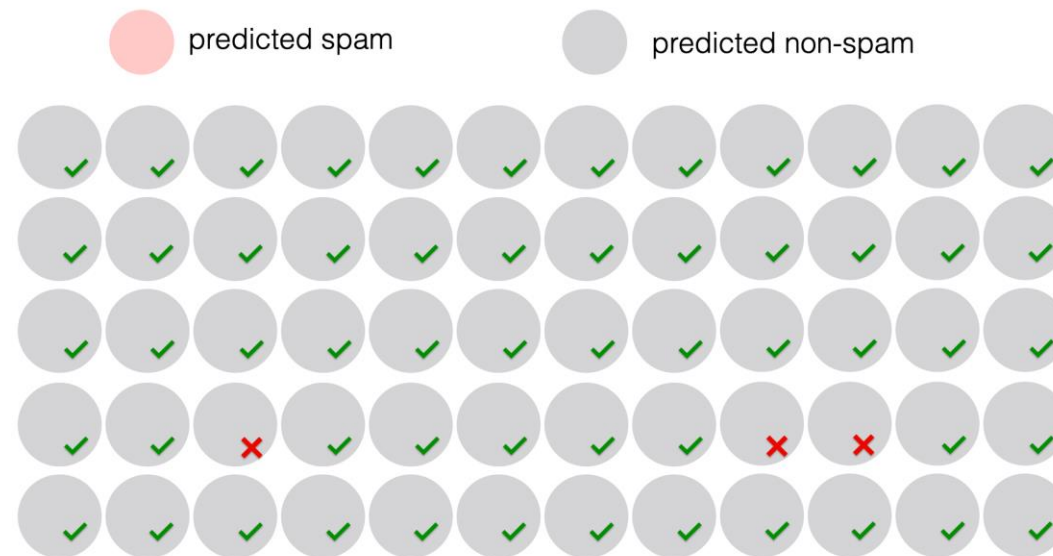
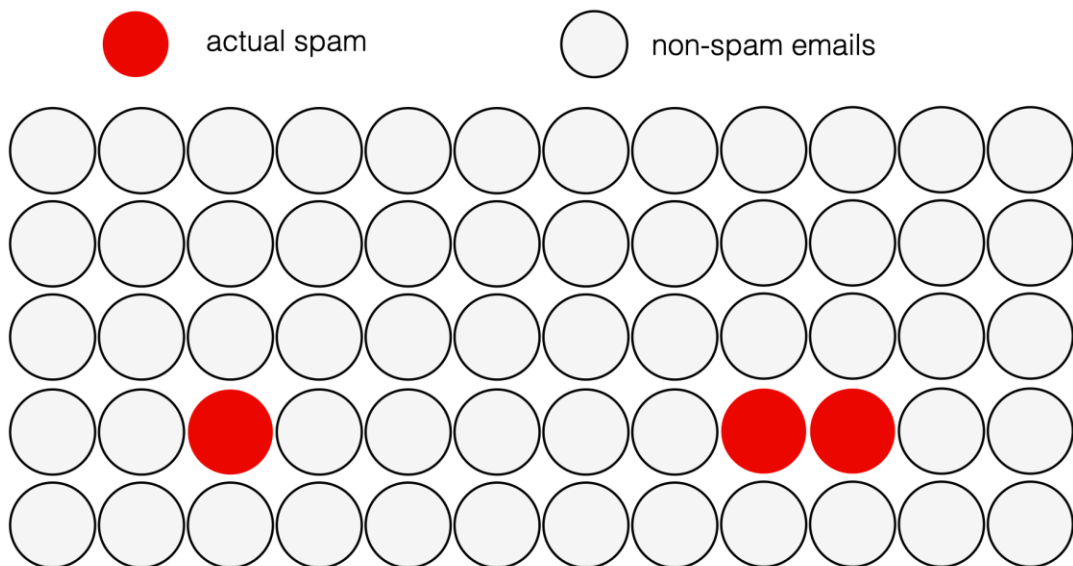




# Accuracy downside

- Accuracy treats all classes as equally important and looks at all correct predictions.
- However, many real-world applications have a **high imbalance of classes**. These are the cases when one category has significantly more frequent occurrences than the other.
- Problem when : you are interested in predicting the **events that rarely occur**.

# Accuracy downside

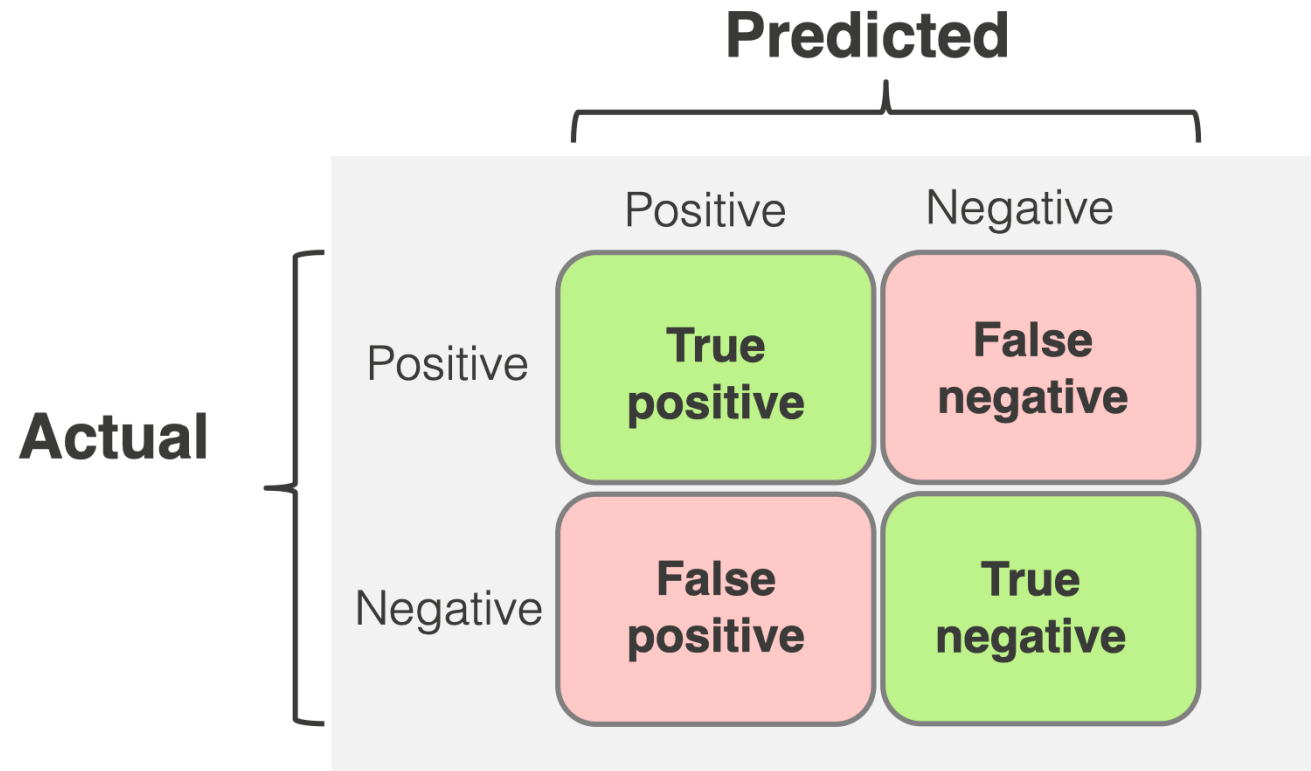


- In this specific example, the accuracy is 95%: yes, the model missed every spam email, but it was still right in 57 cases out of 60.
- However, this accuracy is now meaningless. The model does not serve the primary goal and does not help identify the target event.

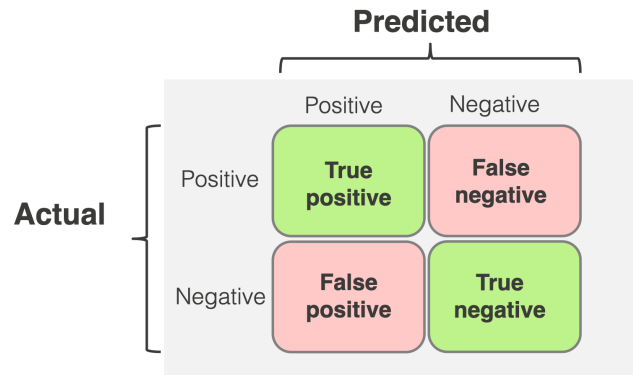
# Confusion matrix

- In binary classification, there are two possible target classes, which are typically labeled as "positive" and "negative" or "1" and "0". In our spam example, the target (positive class) is "spam," and the negative class is "not spam."
- When evaluating the accuracy, we looked at correct and wrong predictions disregarding the class label. However, in binary classification, we can be "correct" and "wrong" in two different ways.

# Confusion matrix







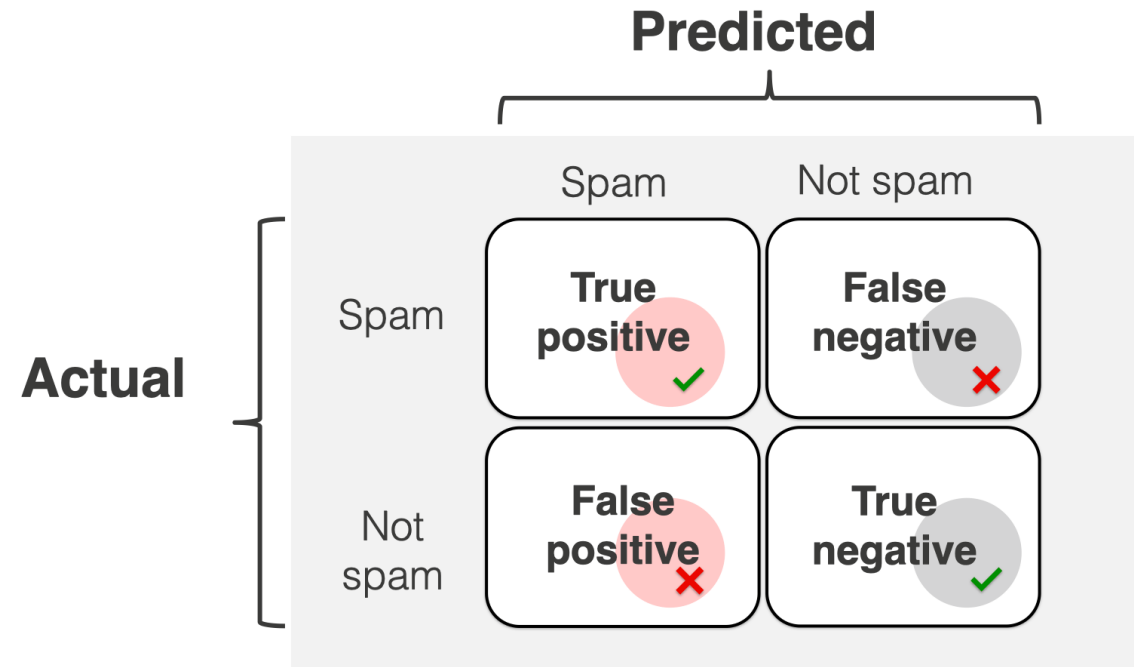
|    | Predicted | Actual    |                |
|----|-----------|-----------|----------------|
| 1. | Not fraud | Not fraud |                |
| 2. | Not fraud | Not fraud |                |
| 3. | Not fraud | Fraud     | False Negative |
| 4. | Fraud     | Fraud     | False Positive |
| n. | Fraud     | Not fraud |                |



|    | Predicted | Actual    |               |
|----|-----------|-----------|---------------|
| 1. | Not fraud | Not fraud | True Negative |
| 2. | Not fraud | Not fraud |               |
| 3. | Not fraud | Fraud     |               |
| 4. | Fraud     | Fraud     | True Positive |
| n. | Fraud     | Not fraud |               |

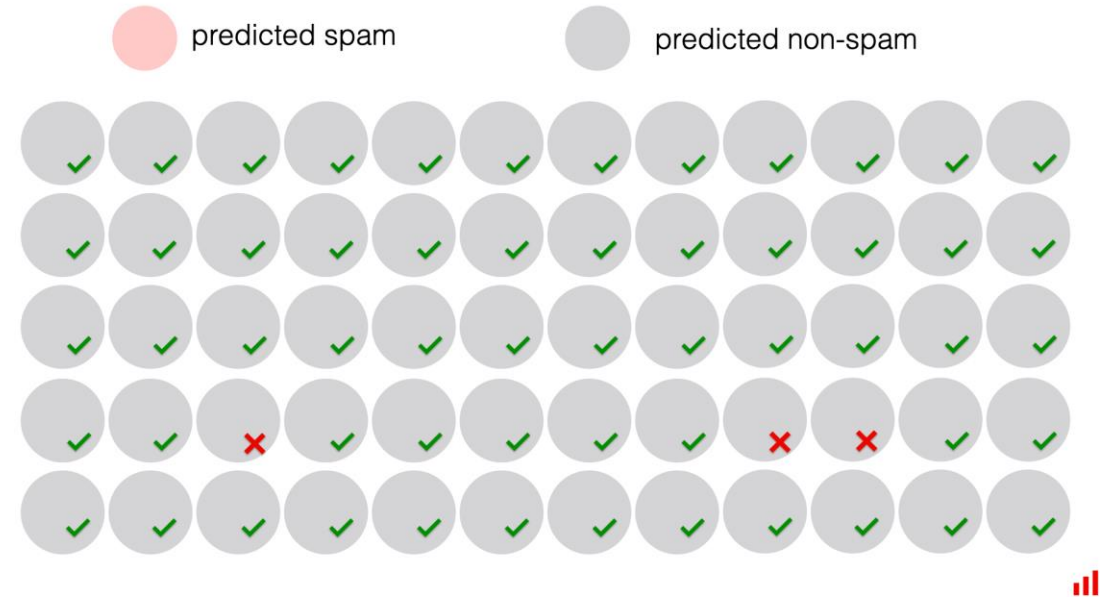
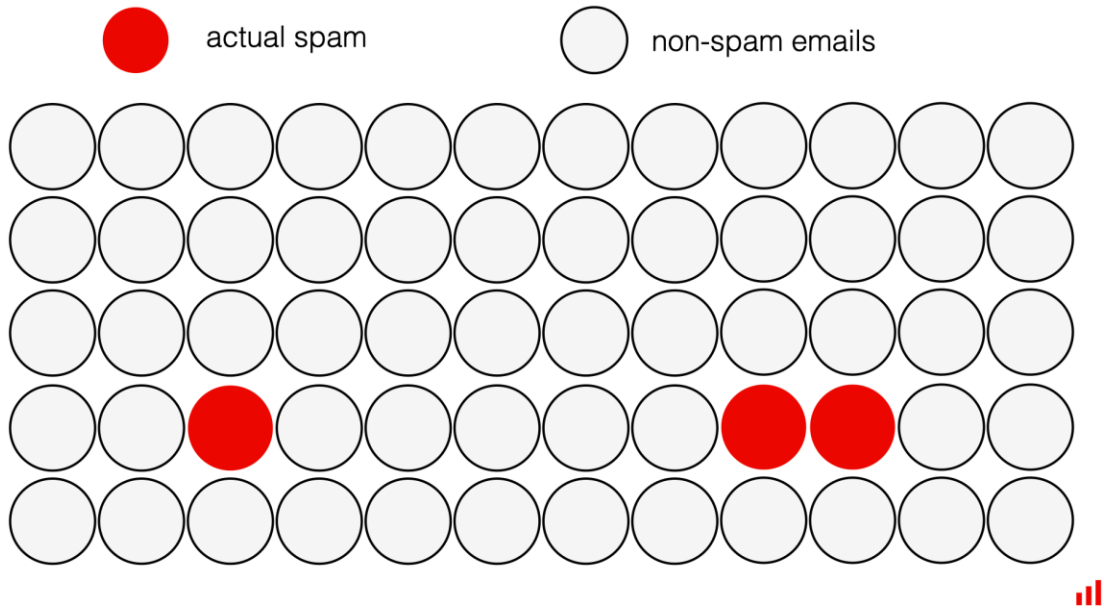


- **Correct predictions** include so-called true positives and true negatives. This is how it unpacks for our spam use case example:
  - **True positive (TP):** An email that is actually spam and is correctly classified by the model as spam.
  - **True negative (TN):** An email that is actually not spam and is correctly classified by the model as not spam.
- **Model errors** include so-called false positives and false negatives. In our example:
  - **False Positive (FP):** An email that is actually not spam but is incorrectly classified by the model as spam (a "false alarm").
  - **False Negative (FN):** An email that is actually spam but is incorrectly classified by the model as not spam (a "missed spam").



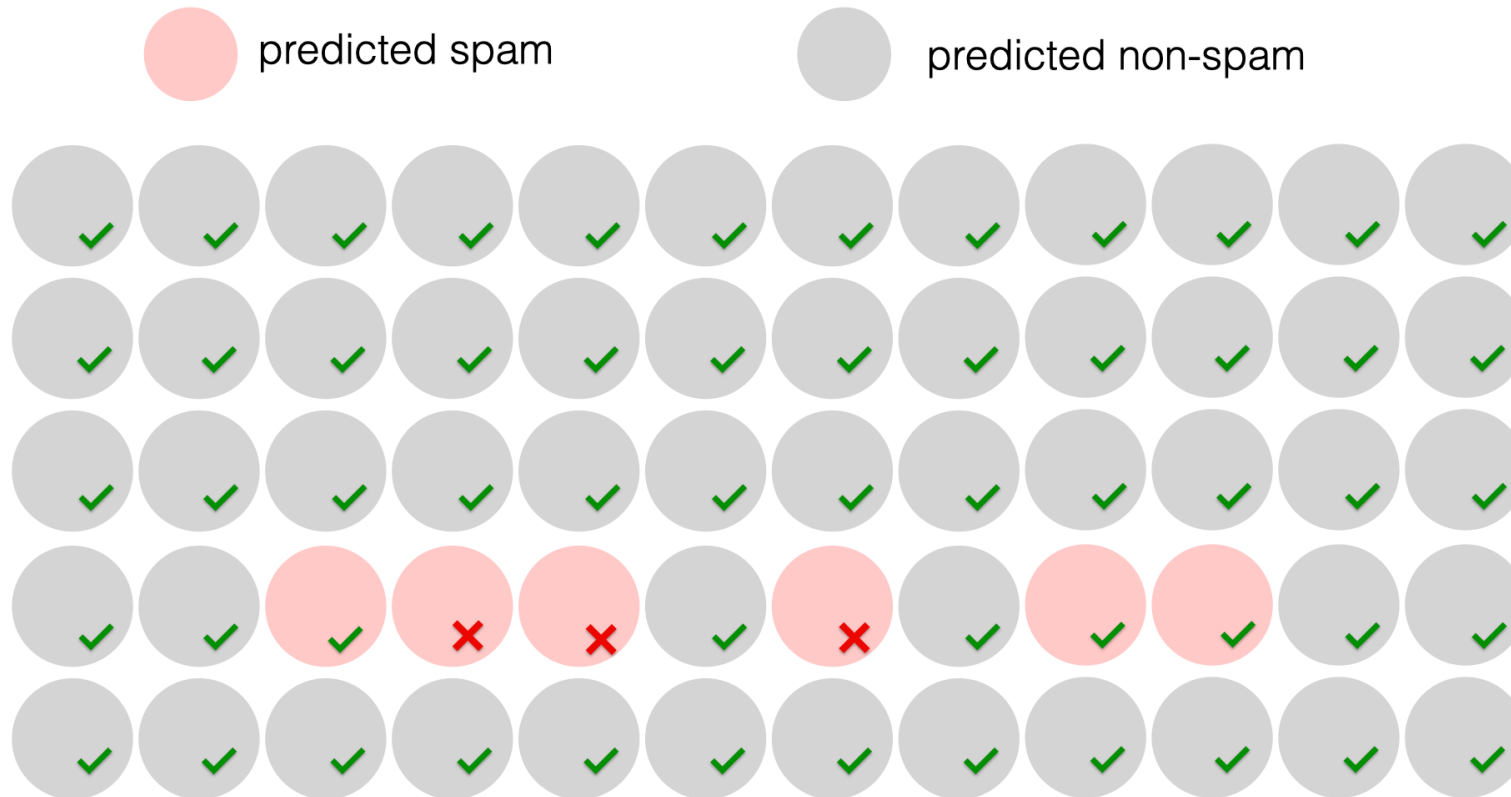
- Precision : how often a machine learning model correctly predicts the positive class

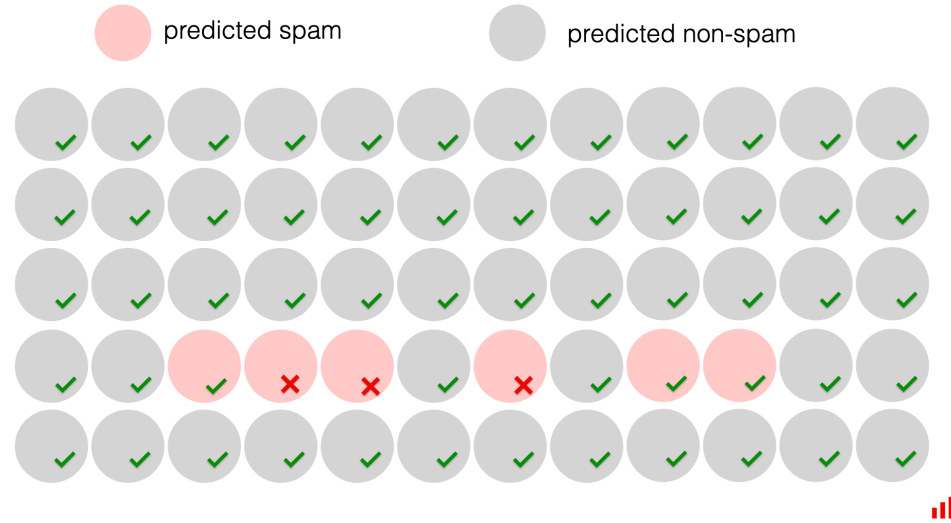
$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$



- **accuracy** is 95% (the model is right in 57 cases out of 60), but the **precision** is 0!

- Another Machine Learning Model





correct positive predictions ● ● ● 3

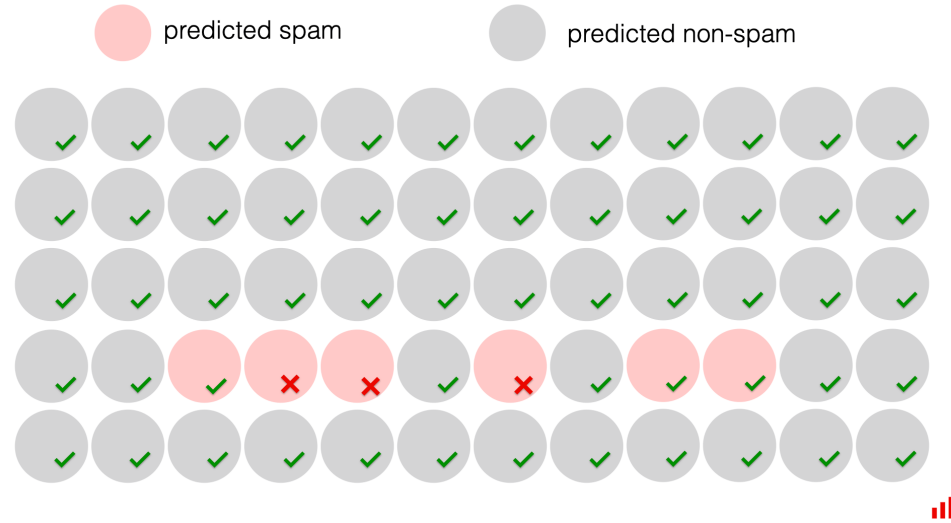
Precision = ————— = 50%

all positive predictions ● ● ● ● ● ● 6

- Recall (Also called sensitivity): how often a machine learning model correctly identifies positive instances (true positives) from all the actual positive samples in the dataset.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$





correct positive  
predictions



3

Recall =

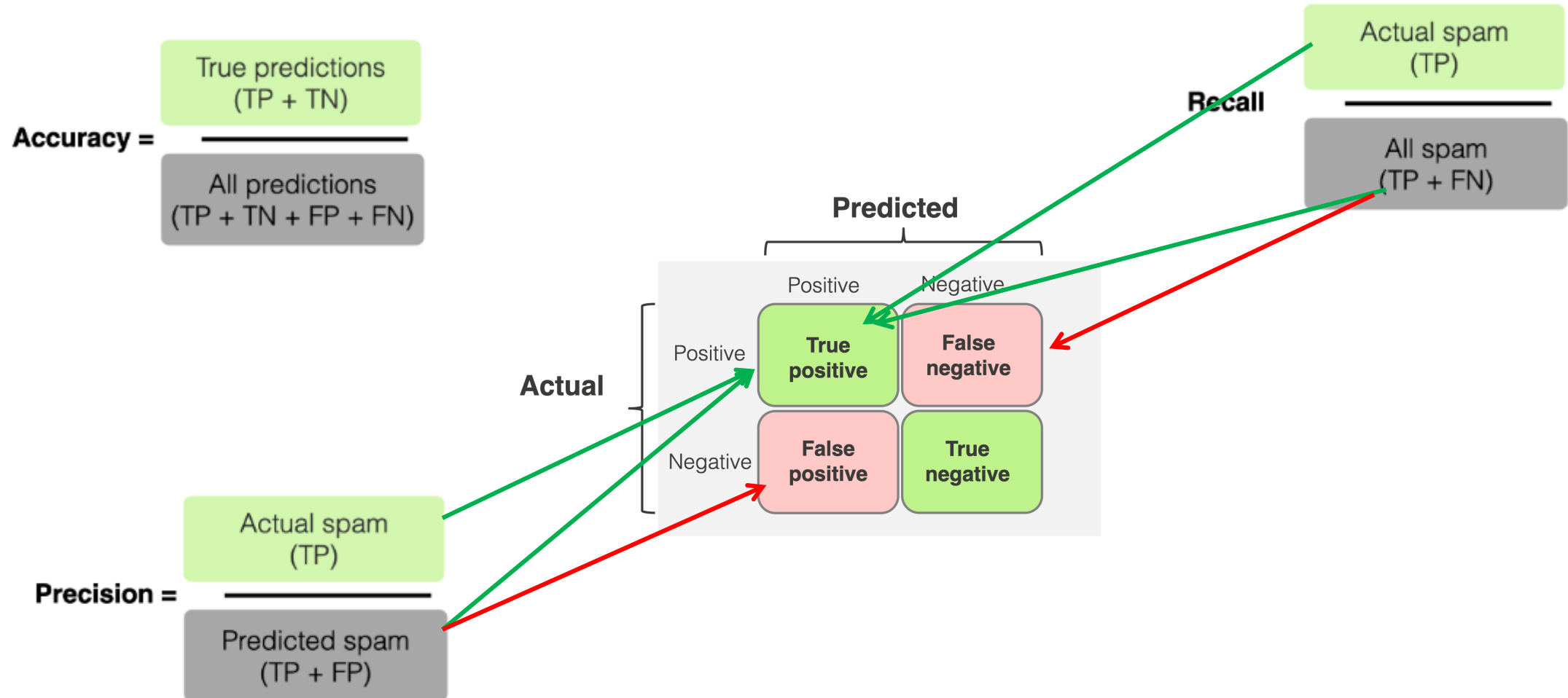
all positive instances



3

**=100%**

# Summary



# Cross-Validation

- Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds.

# Cross-validation

- In cross-validation the original sample is split into two parts. One part is called the training (or *derivation*) sample, and the other part is called the *validation (or validation + testing)* sample.

- **What portion of the sample should be in each part?**

If sample size is very large, it is often best to split the sample in half. For smaller samples, it is more conventional to split the sample such that  $\frac{2}{3}$  of the observations are in the derivation sample and  $\frac{1}{3}$  are in the validation sample.

# CROSS VALIDATION – THE IDEAL PROCEDURE

1. Divide data into three sets, training, validation and test sets



2. Find the optimal model on the training set, and use the test set to check its predictive capability



3. See how well the model can predict the test set



4. The validation error gives an unbiased estimate of the predictive power of a model

# K-Fold Cross-validation

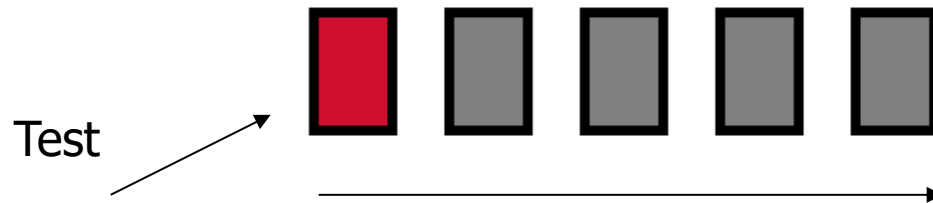
- *Cross-validation* avoids overlapping test sets
  - First step: data is split into  $k$  subsets of equal size
  - Second step: each subset in turn is used for testing and the remainder for training
    - for  $i = 1:10$  (here  $k = 10$ )
      - train on 90% of data,
      - $\text{Acc}(i)$  = accuracy on other 10%
    - end
  - Cross-Validation-Accuracy =  $1/k \sum_i \text{Acc}(i)$ 
    - choose the method with the highest cross-validation accuracy
    - common values for  $k$  are 5 and 10
- This is called *k-fold cross-validation*

# Example

—*Break up data into groups of the same size*



—*Hold aside one group for testing and use the rest to build model*

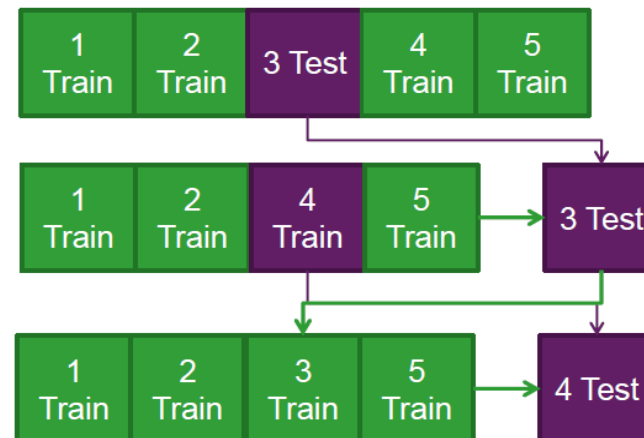


—*Repeat*



# 5-fold Cross Validation Example

1. Split the data into 5 samples
2. Fit a model to the training samples and use the test sample to calculate a Cross-Validation (CV) metric.
3. Repeat the process for the next sample, until all samples have been used to either train or test the model





# Introduction to WEKA

---

- Weka Tool
- **W**aikato **E**nvironment for **K**nowledge **A**nalysis
  - It's a data mining/machine learning tool developed by Department of Computer Science, University of Waikato, New Zealand.
  - Weka is a collection of machine learning algorithms for data mining tasks.
  - Weka is open source software issued under the GNU General Public License.

# Download and Install WEKA

---

- Website
  - <https://ml.cms.waikato.ac.nz/weka>
- Support multiple platforms (written in java):
  - Windows, Mac OS X and Linux
- Documentation
  - <https://waikato.github.io/weka-wiki/documentation/>

# Weka: Main Features

---

- 49 data preprocessing tools
- 76 classification/regression algorithms
- 8 clustering algorithms
- 3 algorithms for finding association rules
- 15 attribute/subset evaluators + 10 search algorithms for feature selection

# Datasets in Weka

- Each entry in a dataset is an instance of the java class:
  - `weka.core.Instance`
- Each instance consists of a number of attributes
  - *Nominal*: one of a predefined list of values
    - e.g. red, green, blue
  - *Numeric*: A real or integer number
  - *String*: Enclosed in “double quotes”
  - *Date*
  - *Relational*

# ARFF Files

- Weka wants its input data in ARFF format.
  - A dataset has to start with a declaration of its name:
    - @relation name
  - @attribute attribute\_name specification
    - If an attribute is nominal, specification contains a list of the possible attribute values in curly brackets:
      - @attribute nominal\_attribute {first\_value, second\_value, third\_value}
    - If an attribute is numeric, specification is replaced by the keyword numeric: (Integer values are treated as real numbers in WEKA.)
      - @attribute numeric\_attribute numeric
  - After the attribute declarations, the actual data is introduced by a tag:
    - @data
- The external representation of an Instances class Consists of:
  - A header: Describes the attribute types
  - Data section: Comma separated list of data

# ARFF File

---

```
@relation weather
@attribute outlook { sunny, overcast, rainy }
@attribute temperature numeric
@attribute humidity numeric
@attribute windy { TRUE, FALSE }
@attribute play { yes, no }
@data
sunny, 85, 85, FALSE, no
sunny, 80, 90, TRUE, no
overcast, 83, 86, FALSE, yes
rainy, 70, 96, FALSE, yes
rainy, 68, 80, FALSE, yes
rainy, 65, 70, TRUE, no
overcast, 64, 65, TRUE, yes
sunny, 72, 95, FALSE, no
sunny, 69, 70, FALSE, yes
rainy, 75, 80, FALSE, yes
sunny, 75, 70, TRUE, yes
overcast, 72, 90, TRUE, yes
overcast, 81, 75, FALSE, yes
rainy, 71, 91, TRUE, no
```

**Weka Explorer**

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

Filter: Choose None Apply

Current relation: Attributes: 36

Selected attribute: Name: date, Missing: 1 (0%), Distinct: 7, Type: Nominal, Unique: 0 (0%)

Attributes: All None Invert

| No. | Name                                     |
|-----|--|
| 1   | <input checked="" type="checkbox"/> date |
| 2   | <input type="checkbox"/> plant-stand     |
| 3   | <input type="checkbox"/> precip          |
| 4   | <input type="checkbox"/> temp            |
| 5   | <input type="checkbox"/> hail            |
| 6   | <input type="checkbox"/> crop-hist       |
| 7   | <input type="checkbox"/> area-damaged    |
| 8   | <input type="checkbox"/> severity        |
| 9   | <input type="checkbox"/> seed-tmt        |
| 10  | <input type="checkbox"/> germination     |

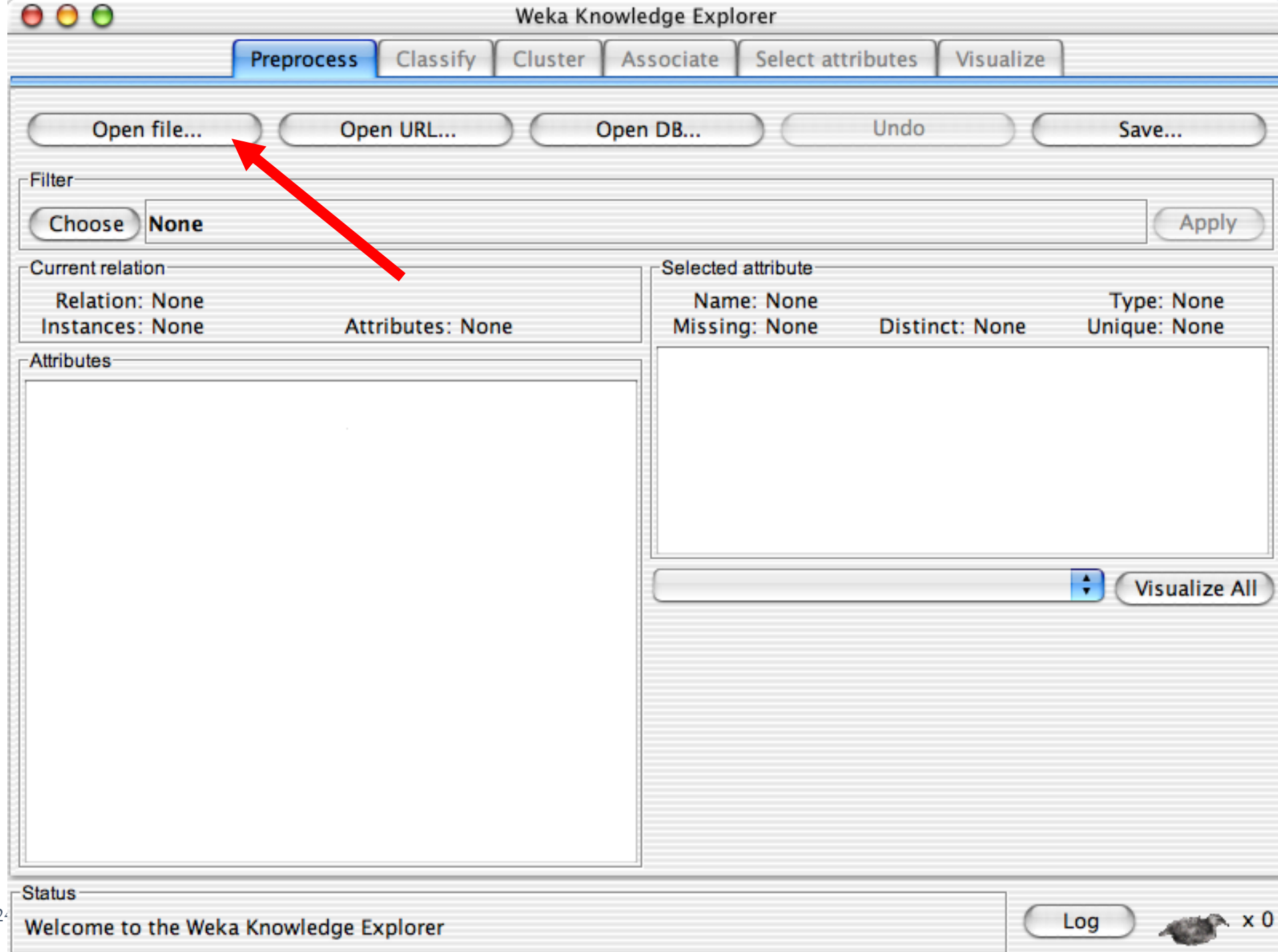
Remove

| No. | Label     | Count |
|-----|-----------|-------|
| 1   | april     | 26    |
| 2   | may       | 75    |
| 3   | june      | 93    |
| 4   | july      | 118   |
| 5   | august    | 131   |
| 6   | september | 149   |

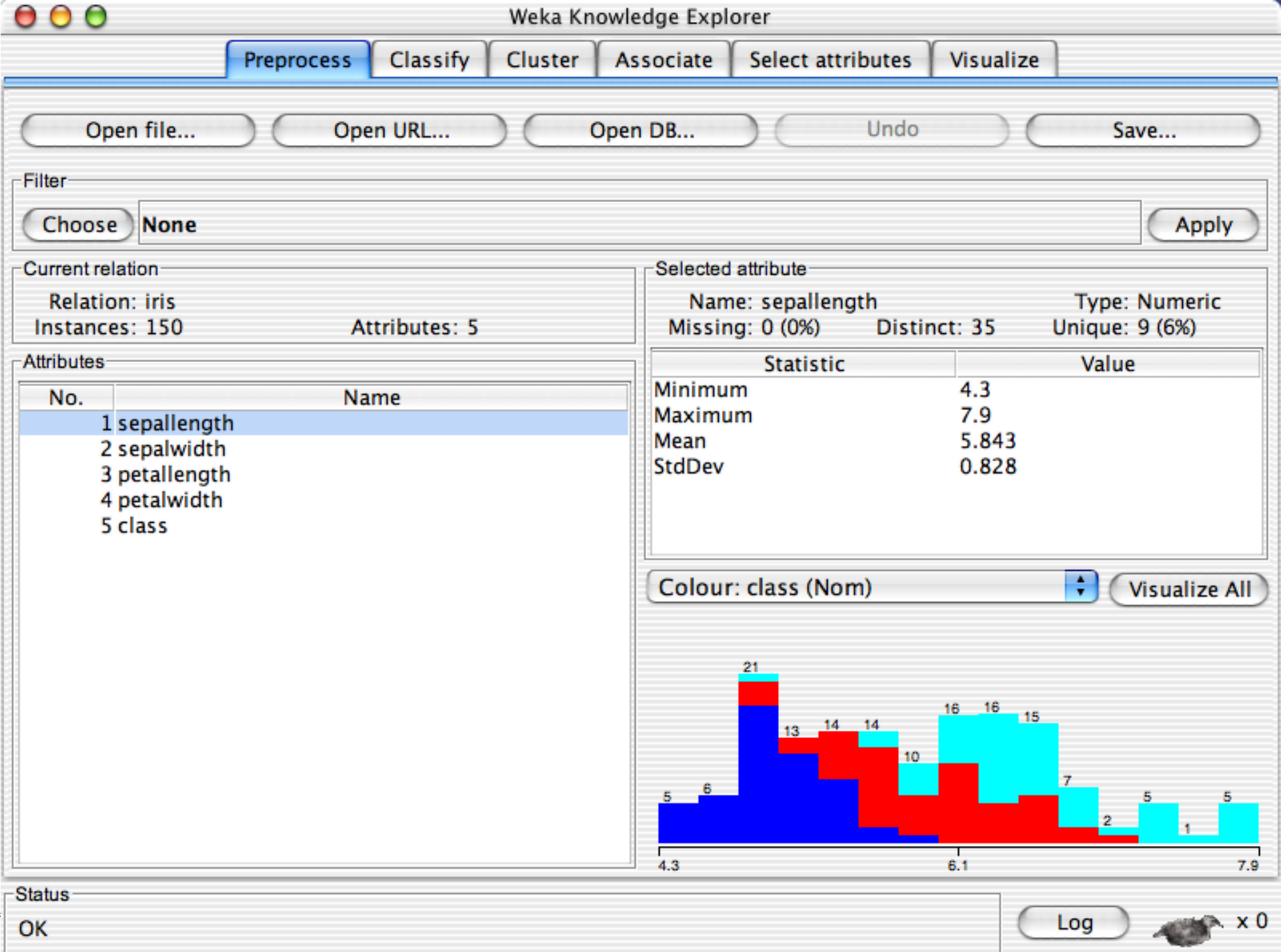
Class: class (Nom) Visualize All

Status: OK Log x 0

**filter** **load** **analyze**







PreprocessClassifyClusterAssociateSelect attributesVisualize

Open file...Open URL...Open DB...UndoSave...

Filter

ChooseNoneApply

Current relation

Relation: iris  
Instances: 150Attributes: 5

Attributes

| No. | Name        |
|-----|-------------|
| 1   | sepalength  |
| 2   | sepalwidth  |
| 3   | petallength |
| 4   | petalwidth  |
| 5   | class       |

Selected attribute

Name: class  
Missing: 0 (0%)Distinct: 3Type: Nominal  
Unique: 0 (0%)

| Label           | Count |
|-----------------|-------|
| Iris-setosa     | 50    |
| Iris-versicolor | 50    |
| Iris-virginica  | 50    |

Colour: class (Nom)Visualize All


50

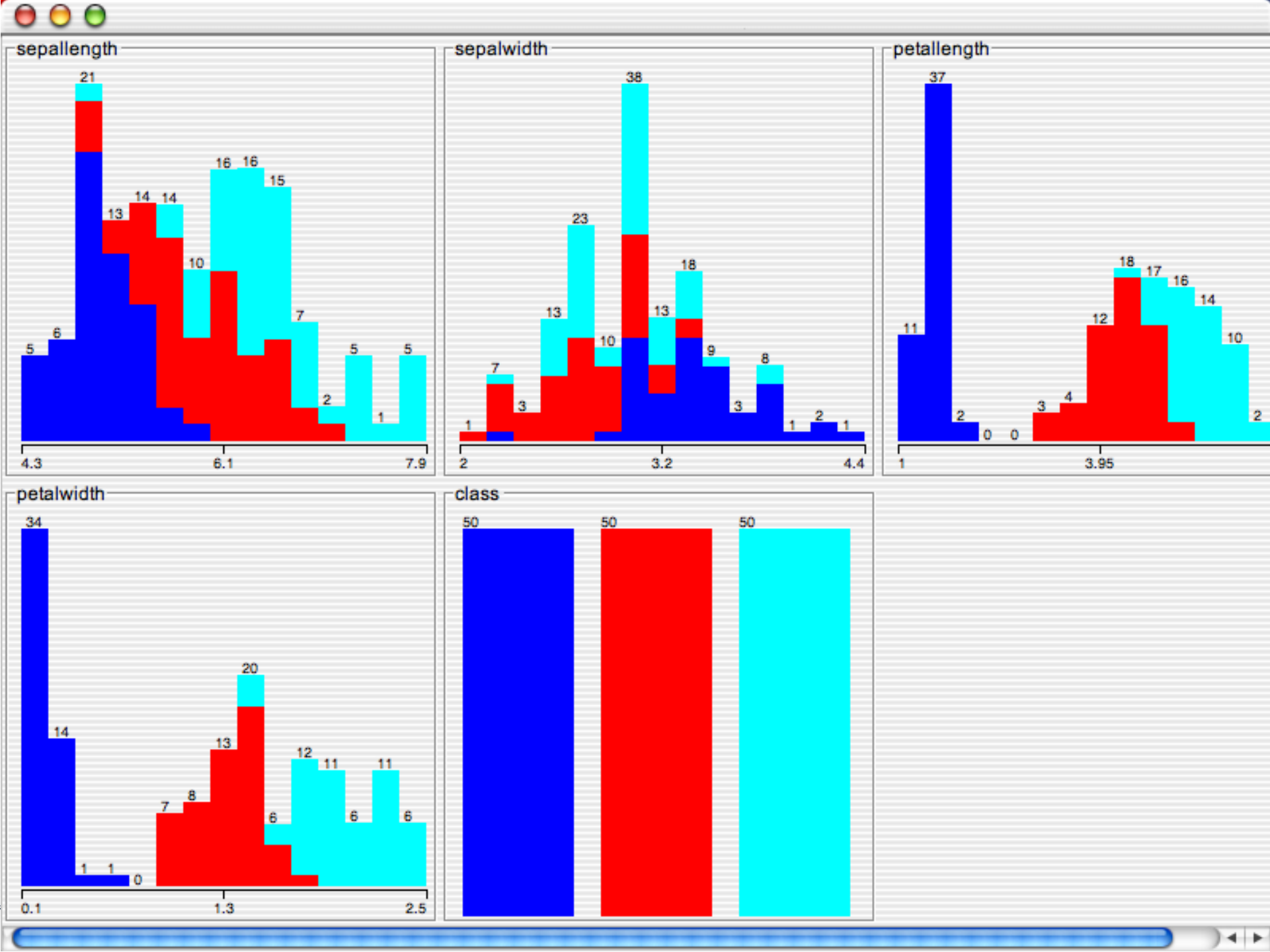
50

50

Status

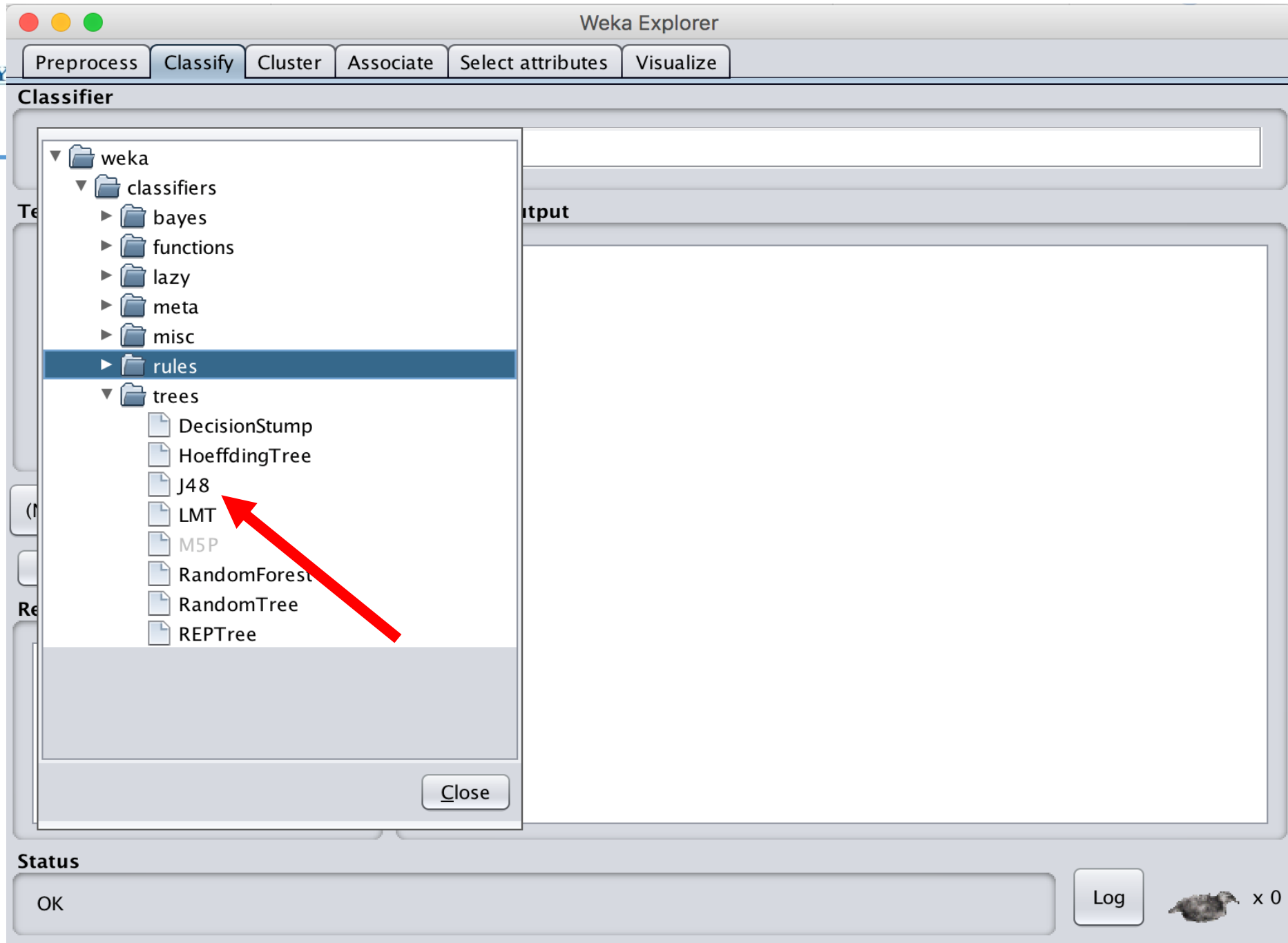
OK

Log x 0



# Explorer: building “classifiers”

- Classifiers in WEKA are models for predicting nominal or numeric quantities
- Implemented learning schemes include:
  - **Decision trees** and lists, instance-based classifiers, support vector machines, multi-layer perceptrons, logistic regression, Bayes' nets, ...



Weka Knowledge Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier: Choose **J48 -C 0.25 -M 2**

Test options

☐ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 10

☒ Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

11:49:05 - trees.j48.J48

Classifier output

Time taken to build model: 0.24 seconds

=== Evaluation on test split ===

=== Summary ===

|                                  |           |           |
|----------------------------------|-----------|-----------|
| Correctly Classified Instances   | 49        | 96.0784 % |
| Incorrectly Classified Instances | 2         | 3.9216 %  |
| Kappa statistic                  | 0.9408    |           |
| Mean absolute error              | 0.0396    |           |
| Root mean squared error          | 0.1579    |           |
| Relative absolute error          | 8.8979 %  |           |
| Root relative squared error      | 33.4091 % |           |
| Total Number of Instances        | 51        |           |

=== Detailed Accuracy By Class ===

| TP Rate | FP Rate | Precision | Recall | F-Measure | Class           |
|---------|---------|-----------|--------|-----------|-----------------|
| 1       | 0       | 1         | 1      | 1         | Iris-setosa     |
| 1       | 0.063   | 0.905     | 1      | 0.95      | Iris-versicolor |
| 0.882   | 0       | 1         | 0.882  | 0.938     | Iris-virginica  |

=== Confusion Matrix ===

| a  | b  | c  | <-- classified as   |
|----|----|----|---------------------|
| 15 | 0  | 0  | a = Iris-setosa     |
| 0  | 19 | 0  | b = Iris-versicolor |
| 0  | 2  | 15 | c = Iris-virginica  |

Status: OK

Log x 0

Weka Knowledge Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier: Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 10

☒ Percentage split % 66

More options...

(Nom) class

Start Stop

Classifier output

Time taken to build model: 0.24 seconds

=== Evaluation on test split ===

=== Summary ===

|                                  |           |           |
|----------------------------------|-----------|-----------|
| Correctly Classified Instances   | 49        | 96.0784 % |
| Incorrectly Classified Instances | 2         | 3.9216 %  |
| Kappa statistic                  | 0.9408    |           |
| Mean absolute error              | 0.0396    |           |
| Root mean squared error          | 0.1579    |           |
| Relative absolute error          | 8.8979 %  |           |
| Root relative squared error      | 33.4091 % |           |
| Total Number of Instances        | 51        |           |

=== Detailed Accuracy By Class ===

| Recall | F-Measure | Class           |
|--------|-----------|-----------------|
| 1      | 1         | Iris-setosa     |
| 1      | 0.95      | Iris-versicolor |
| 0.882  | 0.938     | Iris-virginica  |

Result list (right-click for options)

11:49:05 - trees.j48.J48

- View in main window
- View in separate window
- Save result buffer
- Load model
- Save model
- Re-evaluate model on current test set
- Visualize classifier errors
- Visualize tree**
- Visualize margin curve
- Visualize threshold curve
- Visualize cost curve

Status: OK

Log x 0



Weka Knowledge Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier: Choose J48 -C 0.25 -M 2

Test options:

- ☐ Use training set
- ☐ Supplied test set
- ☐ Cross-validation
- ☒ Percentage split

More options

(Nom) class

Start

Result list (right-click for details): 11:49:05 - trees.j48.J

Tree View

```

graph TD
    A(petalwidth) -- "<= 0.6" --> B[Iris-setosa (50.0)]
    A -- "> 0.6" --> C(petalwidth)
    C -- "<= 1.7" --> D(petalwidth)
    C -- "> 1.7" --> E[Iris-virginica (46.0/1.0)]
    D -- "<= 4.9" --> F[Iris-versicolor (48.0/1.0)]
    D -- "> 4.9" --> G(petalwidth)
    G -- "<= 1.5" --> H[Iris-virginica (3.0)]
    G -- "> 1.5" --> I[Iris-versicolor (3.0/1.0)]
  
```

96.0784 %  
3.9216 %

ass  
is-setosa  
is-versicolor  
is-virginica

15 0 0 | a = Iris-setosa  
0 19 0 | b = Iris-versicolor  
0 2 15 | c = Iris-virginica

Status: OK

Log x 0



# ARFF File

- Inclass Demo