# MACHINE LEARNING USING R

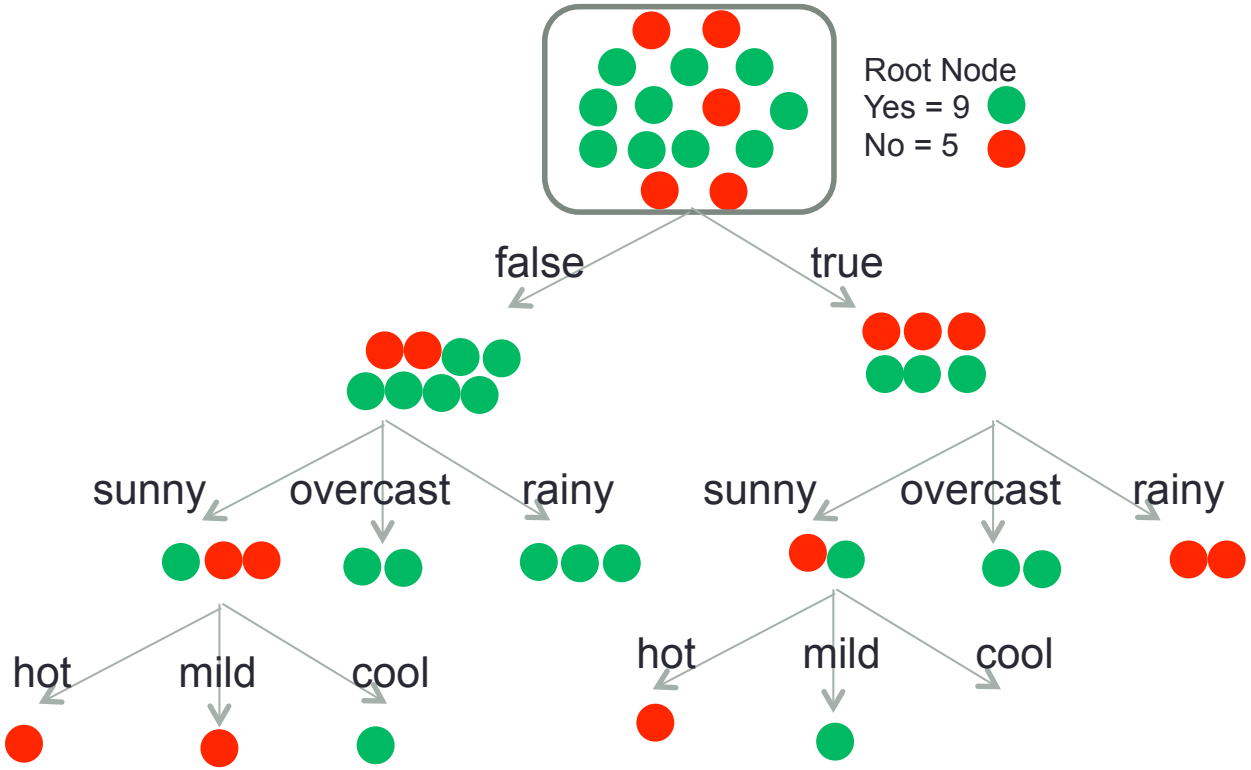
Class 4

# Decision Trees

| Outlook | Temperature Numeric | Temperature Nominal | Humidity Numeric | Humidity Nominal | Windy | Play |
|---|---|---|---|---|---|---|
| overcast | 83 | hot | 86 | high | FALSE | **yes** |
| overcast | 64 | cool | 65 | normal | TRUE | **yes** |
| overcast | 72 | mild | 90 | high | TRUE | **yes** |
| overcast | 81 | hot | 75 | normal | FALSE | **yes** |
| rainy | 70 | mild | 96 | high | FALSE | **yes** |
| rainy | 68 | cool | 80 | normal | FALSE | **yes** |
| rainy | 65 | cool | 70 | normal | TRUE | **no** |
| rainy | 75 | mild | 80 | normal | FALSE | **yes** |
| rainy | 71 | mild | 91 | high | TRUE | **no** |
| sunny | 85 | hot | 85 | high | FALSE | **no** |
| sunny | 80 | hot | 90 | high | TRUE | **no** |
| sunny | 72 | mild | 95 | high | FALSE | **no** |
| sunny | 69 | cool | 70 | normal | FALSE | **yes** |
| sunny | 75 | mild | 70 | normal | TRUE | **yes** |

[weather dataset]

# Decision Tree # 1



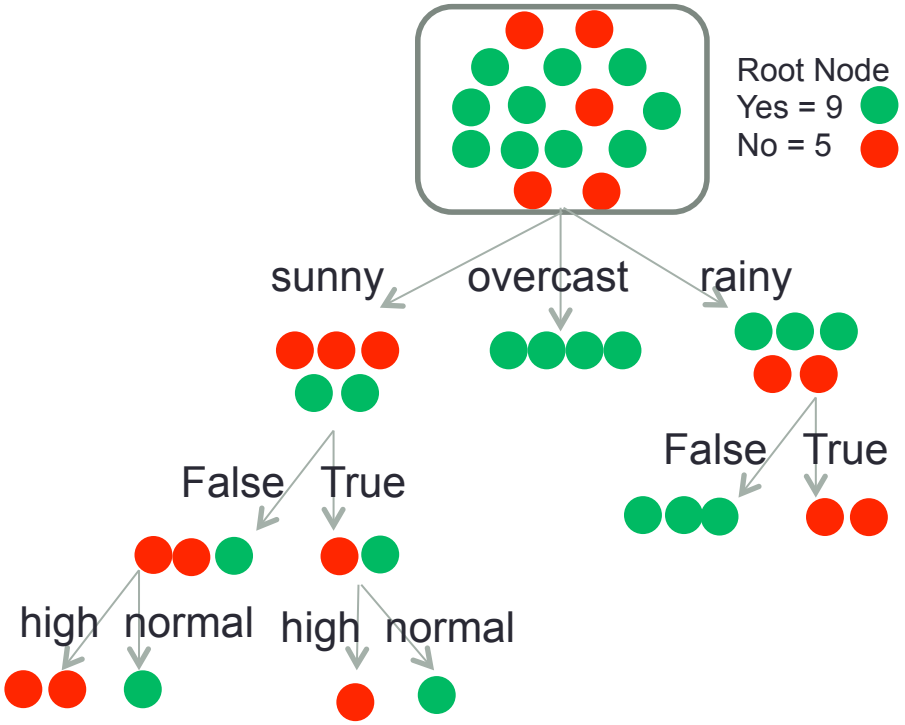

Windy

Outlook

Temperature

Humidity

# Decision Tree # 2




Outlook

Windy

Humidity

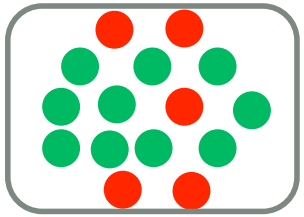
Temperature

# Choosing the best split

- Entropy is the measure of purity in a segment of data.

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

- Entropy for “play”



*Entropy(Play) = 0.940286*

Entropy (play) = - 9/14 log2 9/14 – 5/14 log 2 5/14

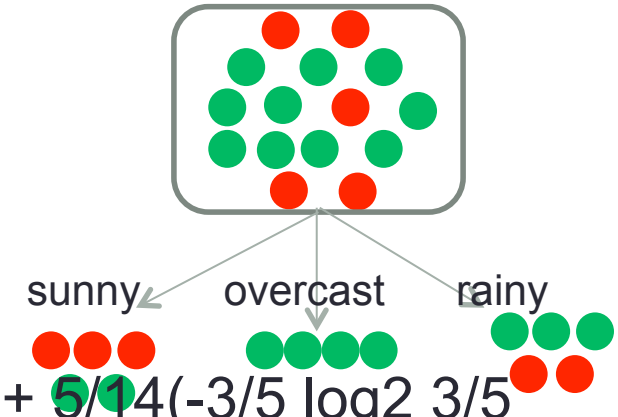
= *0.940286*

# Choosing the best split

- Entropy using frequency of two attributes:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

- Entropy for “play” based on “outlook”




*Entropy (play, outlook)*

= 5/14(-3/5 log2 3/5 – 2/5 log2 2/5) + 4/14(0) + 5/14(-3/5 log2 3/5 – 2/5 log2 2/5

= 0.6935361

# Choosing the best split

- Information gain is the change in homogeneity resulting from a split on each possible feature.
- Calculated as the difference between entropy in segment before the split (S1), and the partitions resulting from the split (S2):

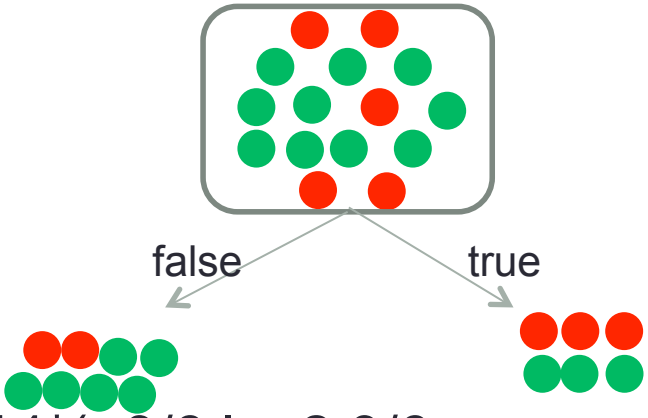
$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

- The higher the information gain, the better a feature is at creating homogeneous groups after a split on that feature.
- If the information gain is zero, there is no reduction in entropy for splitting on this feature.
- The maximum information gain is equal to the entropy prior to the split.

InfoGain = *Entropy(play)* - *Entropy(play, outlook)*
= 0.940286 - 0.6935361
= 0.2467499

# Choosing the best split

- Entropy for “play” based on “windy”




*Entropy (play, windy)*

= 8/14*(-2/8 log2 2/8 – 6/8 log2 6/8) + 6/14*(- 3/6 log2 3/6
– 3/6 log2 3/6)

= 0.8921589

InfoGain(2) = *Entropy (play) – Entropy (play, windy)*
= 0.940286 - 0.8921589
= 0.0481271

Since InfoGain(1) > InfoGain(2), outlook is the chosen split as opposed to windy.

# Decision Trees: Advantages

- An all-purpose classifier that does well on most problems
- Highly-automatic learning process can handle numeric or nominal features, missing data
- Uses only the most important features
- Can be used on data with relatively few training examples or a very large number
- Results in a model that can be interpreted without a mathematical background (for relatively small trees)
- More efficient than other complex models

# Decision Trees: Disadvantages

- Decision tree models are often biased toward splits on features having a large number of levels
- It is easy to overfit or underfit the model
- Can have trouble modeling some relationships due to reliance on axis-parallel splits
- Small changes in training data can result in large changes to decision logic
- Large trees can be difficult to interpret and the decisions they make may seem counterintuitive

# Pruning the Decision tree

## Pre-pruning

- Stop the tree from growing once it reaches a certain number of decisions or if the decision nodes contain only a small number of examples.
- Downside is that there is no way to know whether the tree will miss subtle, but important patterns that it would have learned had it grown to a larger size.

## Post-pruning

- Growing a tree that is too large, then using pruning criteria based on the error rates at the nodes to reduce the size of the tree to a more appropriate level.
- Pruning the tree later on allows the algorithm to be certain that all important data structures were discovered.

# RULE LEARNERS

# Classification Rules

- If – Then rules
  - If there are clouds in the sky, then it will rain.
- Rule learners are generally applied to problems where the features are primarily or entirely nominal.
- They do well at identifying rare events, even if the rare event occurs only for a very specific interaction among features.
- Unlike Trees which must be applied from top-to-bottom, rules are facts that stand alone.
- Trees are based on Divide and Conquer strategy.
- Rules are based on Separate and Conquer strategy.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Classification Rules

| Outlook | Temperature Numeric | Temperature Nominal | Humidity Numeric | Humidity Nominal | Windy | Play |
|---|---|---|---|---|---|---|
| overcast | 83 | hot | 86 | high | FALSE | **yes** |
| overcast | 64 | cool | 65 | normal | TRUE | **yes** |
| overcast | 72 | mild | 90 | high | TRUE | **yes** |
| overcast | 81 | hot | 75 | normal | FALSE | **yes** |
| rainy | 70 | mild | 96 | high | FALSE | **yes** |
| rainy | 68 | cool | 80 | normal | FALSE | **yes** |
| rainy | 65 | cool | 70 | normal | TRUE | **no** |
| rainy | 75 | mild | 80 | normal | FALSE | **yes** |
| rainy | 71 | mild | 91 | high | TRUE | **no** |
| sunny | 85 | hot | 85 | high | FALSE | **no** |
| sunny | 80 | hot | 90 | high | TRUE | **no** |
| sunny | 72 | mild | 95 | high | FALSE | **no** |
| sunny | 69 | cool | 70 | normal | FALSE | **yes** |
| sunny | 75 | mild | 70 | normal | TRUE | **yes** |

[weather dataset]

# Classification Rules

- IF (Humidity = high) and (Outlook = sunny)
  THEN play = no

- IF (Outlook = rainy) and (Windy = TRUE)
  THEN play = no

- OTHERWISE Play = yes

# Classification Rules

| Outlook | Temperature Numeric | Temperature Nominal | Humidity Numeric | Humidity Nominal | Windy | Play |
|---|---|---|---|---|---|---|
| overcast | 83 | hot | 86 | high | FALSE | **yes** |
| overcast | 64 | cool | 65 | normal | TRUE | **yes** |
| overcast | 72 | mild | 90 | high | TRUE | **yes** |
| overcast | 81 | hot | 75 | normal | FALSE | **yes** |
| rainy | 70 | mild | 96 | high | FALSE | **yes** |
| rainy | 68 | cool | 80 | normal | FALSE | **yes** |
| rainy | 65 | cool | 70 | normal | TRUE | **no** |
| rainy | 75 | mild | 80 | normal | FALSE | **yes** |
| rainy | 71 | mild | 91 | high | TRUE | **no** |
| sunny | 85 | hot | 85 | high | FALSE | **no** |
| sunny | 80 | hot | 90 | high | TRUE | **no** |
| sunny | 72 | mild | 95 | high | FALSE | **no** |
| sunny | 69 | cool | 70 | normal | FALSE | **yes** |
| sunny | 75 | mild | 70 | normal | TRUE | **yes** |

[weather dataset]

# Classification Rules

| Outlook | Temperature Numeric | Temperature Nominal | Humidity Numeric | Humidity Nominal | Windy | Play |
|---|---|---|---|---|---|---|
| overcast | 83 | hot | 86 | high | FALSE | **yes** |
| overcast | 64 | cool | 65 | normal | TRUE | **yes** |
| overcast | 72 | mild | 90 | high | TRUE | **yes** |
| overcast | 81 | hot | 75 | normal | FALSE | **yes** |
| rainy | 70 | mild | 96 | high | FALSE | **yes** |
| rainy | 68 | cool | 80 | normal | FALSE | **yes** |
| rainy | 65 | cool | 70 | normal | TRUE | **no** |
| rainy | 75 | mild | 80 | normal | FALSE | **yes** |
| rainy | 71 | mild | 91 | high | TRUE | **no** |
| sunny | 85 | hot | 85 | high | FALSE | **no** |
| sunny | 80 | hot | 90 | high | TRUE | **no** |
| sunny | 72 | mild | 95 | high | FALSE | **no** |
| sunny | 69 | cool | 70 | normal | FALSE | **yes** |
| sunny | 75 | mild | 70 | normal | TRUE | **yes** |

[weather dataset]

# Classification Rules

| Outlook | Temperature Numeric | Temperature Nominal | Humidity Numeric | Humidity Nominal | Windy | Play |
|---|---|---|---|---|---|---|
| overcast | 83 | hot | 86 | high | FALSE | **yes** |
| overcast | 64 | cool | 65 | normal | TRUE | **yes** |
| overcast | 72 | mild | 90 | high | TRUE | **yes** |
| overcast | 81 | hot | 75 | normal | FALSE | **yes** |
| rainy | 70 | mild | 96 | high | FALSE | **yes** |
| rainy | 68 | cool | 80 | normal | FALSE | **yes** |
| | | | | | | |
| rainy | 75 | mild | 80 | normal | FALSE | **yes** |
| | | | | | | |
| | | | | | | |
| sunny | 69 | cool | 70 | normal | FALSE | **yes** |
| sunny | 75 | mild | 70 | normal | TRUE | **yes** |

[weather dataset]

# One Rule Algorithms

- ZeroR - rule learner that learns zero rules. For every unlabeled example it predicts the most common class regardless of the values of its features.

- OneR or 1R - rule learner that learns a single rule.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# One Rule Algorithms

## Advantages

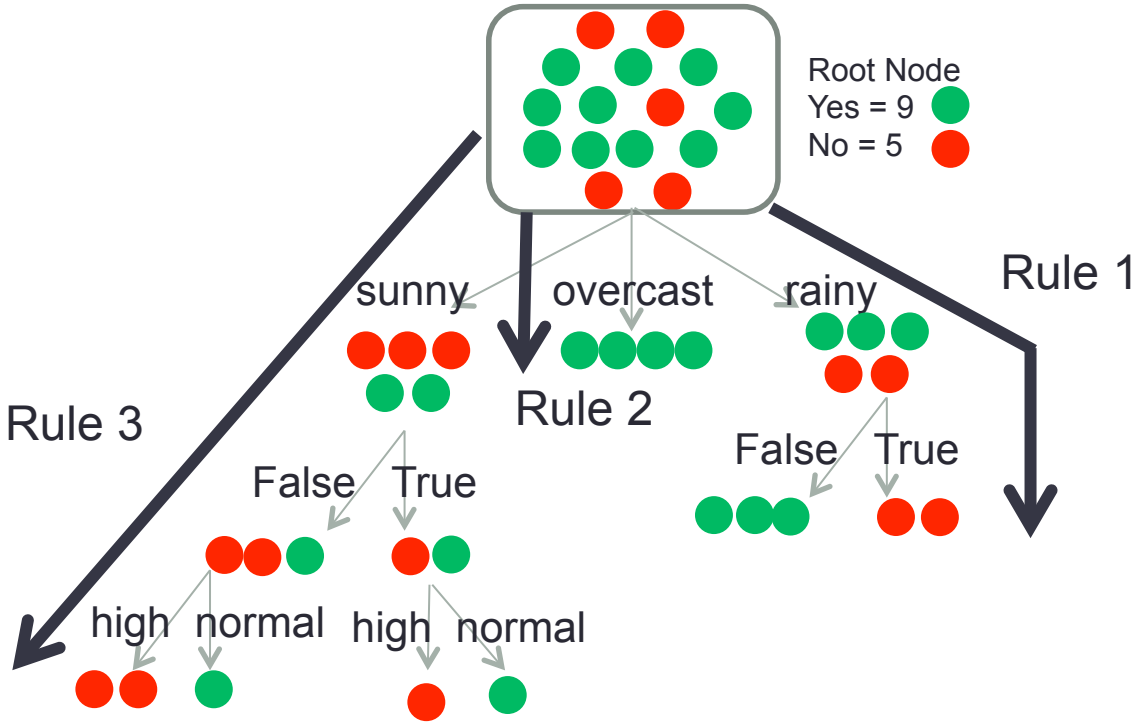
- Generates easy-to-understand, human-readable rules
- Efficient on large and noisy datasets
- Generally produces a simpler model than a comparable decision tree

## Disadvantages

- May result in rules that seem to defy common sense or expert knowledge
- Not ideal for working with numeric data
- Might not perform as well as more complex models

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Classification Rules



# Classification Rules

- The `C5.0() function` will generate a model using classification rules if you specify `rules = TRUE` when training the model.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# EXAMPLE: DECISION TREE RULE LEARNERS

# DECISION TREES: IMPROVING PERFORMANCE

# Improving Model Performance

- In case of Decision Trees using C5.0:
  - we can adjust the `trials` parameter to increase the number of boosting iterations.
  - we can apply a cost matrix to the tree

- This process of adjustment of model fit options is known as Parametric Tuning.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Improving Model Performance

Consider the following questions for Automatic Parametric Tuning:

- What type of machine learning model should be trained on the data?

  *Requires understanding of machine learning models.*

- Which model parameters can be adjusted and how extensively should they be tuned to find the optimal settings?

  *Dictated by the choice of model.*

- What criteria should be used to evaluate the models to find the best candidate?

  *Choice of resampling strategy to create testing and training datasets.*

  *Choice of performance statistics to measure predictive accuracy.*

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Improving Model Performance

- `caret` package is very helpful in automating parametric tuning.
- Decision Trees:
  - By default in `caret`, 3 candidate models will be tested.
  - Tuning a decision tree could result in a comparison of up to 12 different candidate models, possible combinations of `model` (trees vs. rules), `trials`, and `winnow` (true vs. false) settings.
  - Winnowing: When there are numerous alternatives for each test in the tree or ruleset, it is likely that at least one of them will appear to provide valuable predictive information. In applications like these it can be useful to pre-select a subset of the attributes that will be used to construct the decision tree or ruleset. The C5.0 mechanism to do this is called "winnowing" the process for separating useful attributes from unhelpful ones.

[Lantz, Brett. Machine Learning with R. Packt Publishing, https://www.rulequest.com/see5-unix.html#WINNOWING]

# caret: train()

- Fit Predictive Models over Different Tuning Parameters
- This function sets up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure.

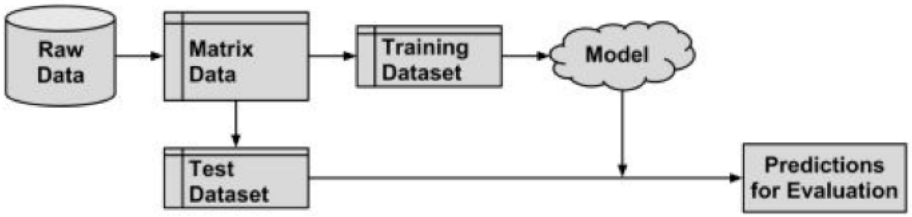
- Optional Reading:

https://topepo.github.io/caret/model-training-and-tuning.html

# RESAMPLING STRATEGY

# Holdout Method

- Procedure used so far to partition the data into training and testing sets.




- For the holdout method to result in a truly accurate estimate of future performance, at no time should results from the test dataset be allowed to influence the model.
- Divide the original data so that in addition to the training and test datasets, a third validation dataset is available. The validation dataset would be used for iterating and refining the model or models chosen, leaving the test dataset to be used only once as a final step to report an estimated error rate for future predictions.
- A typical split between training, test, and validation would be 50 percent, 25 percent, and 25 percent respectively.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Holdout Method

- `> random_ids <- order( runif( 1000))`
- `> credit_train <- credit[ random_ids[ 1: 500],]`
- `> credit_validate <- credit[ random_ids[ 501: 750], ]`
- `> credit_test <- credit[ random_ids[ 751: 1000], ]`

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Holdout Method

- One problem with holdout sampling is that each partition may have a larger or smaller proportion of some classes. In certain cases, particularly those in which a class is a very small proportion of the dataset, this can lead a class to be omitted from the training dataset— a significant problem, because the model cannot then learn this class.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Stratified Random Sampling

- Solution to overcome problem of Holdout method is to use Stratified Random Sampling

- caret package has function createDataPartition that generates stratified test and training sets.

- ```
  > in_train <- createDataPartition( credit $
  default, p = 0.75, list = FALSE)
  ```

- ```
  > credit_train <- credit[ in_train, ]
  ```

- ```
  > credit_test <- credit[-in_train, ]
  ```

[Lantz, Brett. Machine Learning with R. Packt Publishing]
  ```

# Stratified Random Sampling

- Some samples may have too many or too few difficult cases, easy-to-predict cases, or outliers.

- This is especially true for smaller datasets, which may not have a large enough portion of such cases to divide among training and test sets.

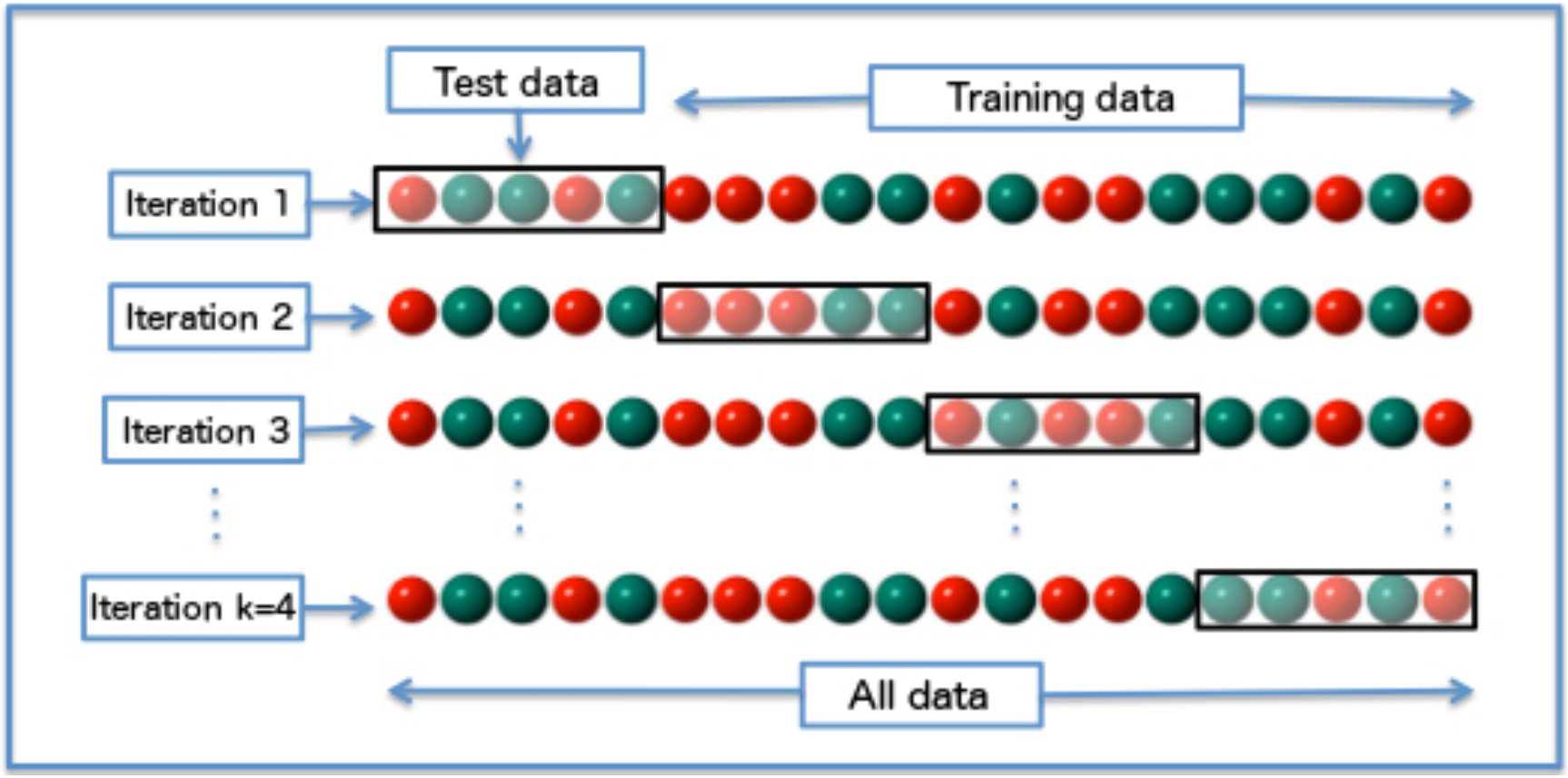
[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Cross-Validation Sampling

- Rather than taking repeated random samples that could potentially use the same record more than once, k-fold CV randomly divides the data into k completely separate random partitions called folds.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Cross Validation




https://upload.wikimedia.org/wikipedia/commons/1/1c/K-fold_cross_validation_EN.jpg

# Cross Validation: Kappa statistics

- Adjusts accuracy by accounting for possibility of a correct prediction by chance alone.

- kappa = [Pr (a) – Pr(e)]/[1-Pr(e)]
  - Pr(a) = Proportion of actual
  - Pr(e) = Proportion of expected

- Kappa can be max. 1
  - Very good agreement = 0.8 to 1
  - Good agreement = 0.6 to 0.8
  - Moderate agreement = 0.4 to 0.6
  - Fair agreement = 0.2 to 0.4
  - Poor agreement = less than 0.2

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Cross Validation: Kappa Statistics

- Pr(a) = 0.864 + 0.110
  = 0.974

- Pr(e) = 0.886 * 0.868
  + 0.114 * 0.132
  = 0.769 + 0.015
  = 0.784

| | sms_test_labels | | |
|---|---|---|---|
| sms_test_pred | ham | spam | Row Total |
| ham | 1201 | 30 | 1231 |
| | 16.317 | 107.620 | |
| | 0.976 | 0.024 | 0.886 |
| | 0.995 | 0.164 | |
| | 0.864 | 0.022 | |
| spam | 6 | 153 | 159 |
| | 126.328 | 833.210 | |
| | 0.038 | 0.962 | 0.114 |
| | 0.005 | 0.836 | |
| | 0.004 | 0.110 | |
| Column Total | 1207 | 183 | 1390 |
| | 0.868 | 0.132 | |

Kappa = (0.974 – 0.784)/(1-0.784)
= 0.879

# Cross Validation

- Although k can be set to any number, by far the most common convention is to use 10-fold cross-validation (10-fold CV).
- 90% of the data is used for Training and 10% of data is used for Testing.
- Let's take a look at some examples.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Leave-one-out method

- An extreme case of k-fold CV is the leave-one-out method, which performs k-fold CV using a fold for each one of the data's examples. This ensures that the greatest amount of data is used for training the model. Although this may seem useful, it is so computationally expensive that it is rarely used in practice.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Bootstrap sampling

- Works by creating several randomly-selected training and test datasets, which are then used to estimate performance statistics.
- The results from the various random datasets are then averaged to obtain a final estimate of future performance.
- In Cross validation partitions each example can appear only once.
- However, Bootstrap allows examples to be selected multiple times through a process of sampling with replacement.
- This means that from the original dataset of n examples, the bootstrap procedure will create one or more new training datasets that also contain n examples, some of which are repeated.
- The corresponding test datasets are then constructed from the set of examples that were not selected for the respective training datasets.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# Bootstrap sampling

- Using sampling with replacement, the probability that any given instance is included in the training dataset is 63.2%.
- The probability of any instance being in the test dataset is 36.8%.
- Only 63.2% of data is represented in the training set some of which are repeated and hence bootstrap sample is less representative of full dataset.
- As a model trained on only 63.2% of the training data is likely to perform worse than a model trained on a larger training set, the bootstrap's performance estimates may be substantially lower than what will be obtained when the model is later trained on the full dataset.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# 0.632 Bootstrap

- Special case of Bootstrap sampling
- calculates the final performance measure as a function of performance on both the training data (which is overly optimistic) and the test data (which is overly pessimistic).
- The final error rate is then estimated as:

$$\text{error} = 0.632 \times \text{error}_{\text{test}} + 0.328 \times \text{error}_{\text{train}}$$

- One advantage of the bootstrap over cross-validation is that it tends to work better with very small datasets.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# trainControl(): method

| Resampling method | Method name | Additional options and default values |
|---|---|---|
| Holdout sampling | `LGOCV` | `p = 0.75` (training data proportion) |
| k-fold cross-validation | `cv` | `number = 10` (number of folds) |
| Repeated k-fold cross-validation | `repeatedcv` | `number = 10` (number of folds)<br>`repeats = 10` (number of iterations) |
| Bootstrap sampling | `boot` | `number = 25` (resampling iterations) |
| 0.632 bootstrap | `boot632` | `number = 25` (resampling iterations) |
| Leave-one-out cross-validation | `LOOCV` | None |

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# PERFORMANCE STATISTICS

# `trainControl()`: `selectionFunction`

- The `trainControl()` parameter `selectionFunction` can be used to choose a function that selects the optimal model among the various candidates.
- The `best` function simply chooses the candidate with the best value on the specified performance measure. This is used by default.
- The `oneSE` function chooses the simplest candidate within one standard error of the best performance.
- The `tolerance` uses the simplest candidate within a user-specified percentage.

[Lantz, Brett. Machine Learning with R. Packt Publishing]

# trainControl()

- ```
  ctrl <- trainControl(method = "cv", number = 10,
        selectionFunction = "oneSE")
  ```

[Lantz, Brett. Machine Learning with R. Packt Publishing]
  ```