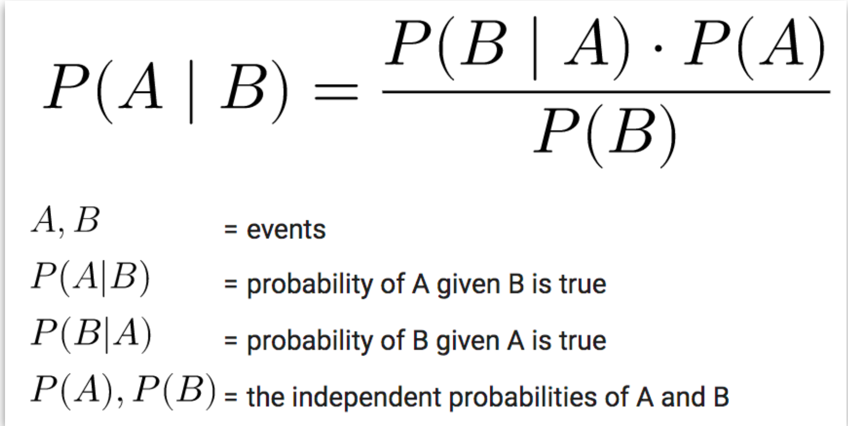
Advanced Data Analysis

DATA 71200

Class 9: Naive Bayes and Decision Trees

Naive Bayes



$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

A, B	= events
$P(A\|B)$	= probability of A given B is true
$P(B\|A)$	= probability of B given A is true
$P(A), P(B)$	= the independent probabilities of A and B

- **Uses Bayes rule to predict the probability of an example belonging to a given class**
- **"Naive" because it examines each feature individually**
- **Calculate per-class statistics for each feature**
  - Bernoulli (binary data) - number of times a feature is non-zero for each class
  - Multinomial (count data) - average of how many times a feature occurs for each class
  - Gaussian (continuous data) - mean and standard deviation of the value of the feature for each class

Naive Bayes

- **Assumptions**
  - Independence between features
  - Gaussian version assumes normally distributed data with the same variance in each class

- **Best Practices**
  - Bernoulli and Multinomial best used on sparse data
  - Gaussian version best used on high-dimensional data

Naive Bayes

- **Parameters**
  - Alpha (Bernoulli and Multinomial) - model complexity
    - The larger the alpha, the more virtual data points that are added for smoothing
- **Strengths**
  - Faster to train and predict than linear models
  - Understandable training procedure
  - Works well on large datasets
    - More efficient on them than linear models
- **Weaknesses**
  - Coefficients not easily interpreted
  - Does not generalize as well as linear models

Decision Trees

- **Learned hierarchy of if/else questions**

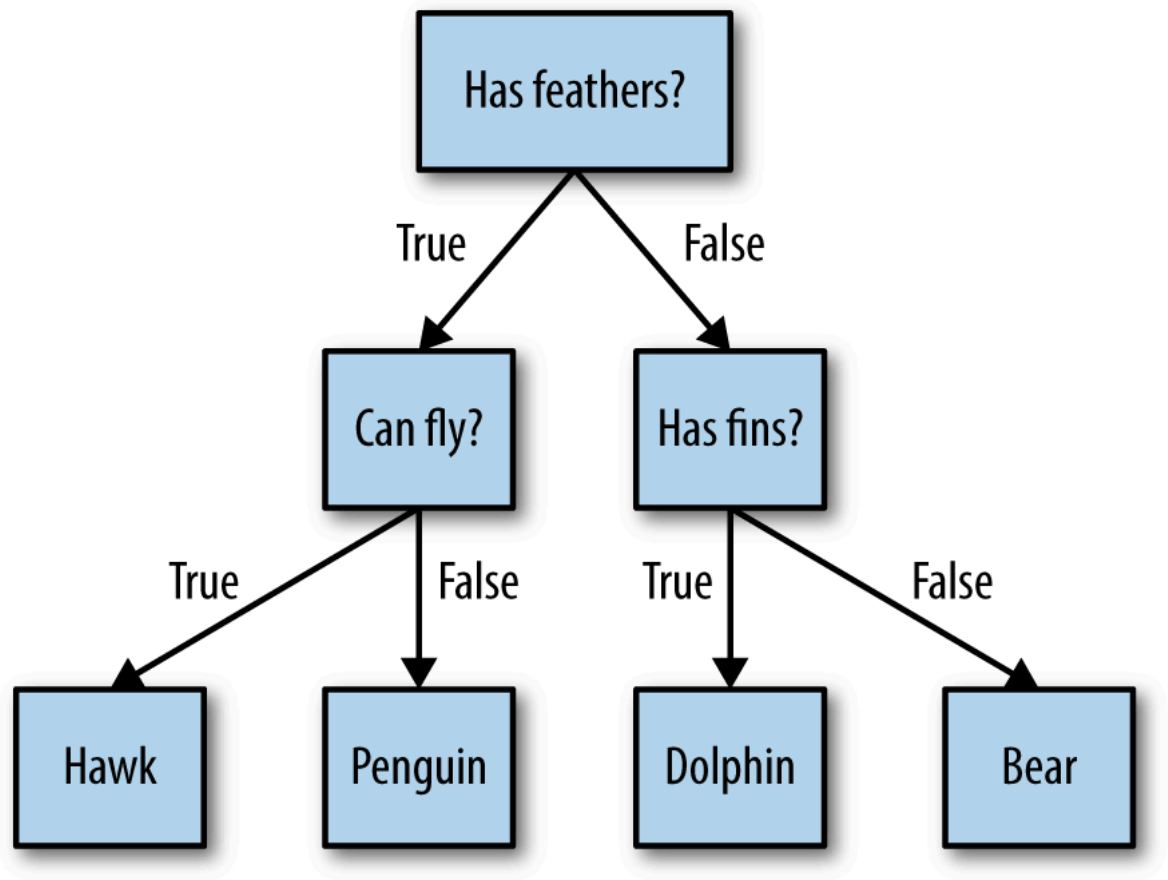


Figure 2-22. A decision tree to distinguish among several animals

Sometimes referred to as CART (Classification and Regression Trees)

Jupyter Notebook 02-supervised-learning .ipynb [55]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Decision Trees

- **Algorithm searches through all possible tests to separate the data into classes**

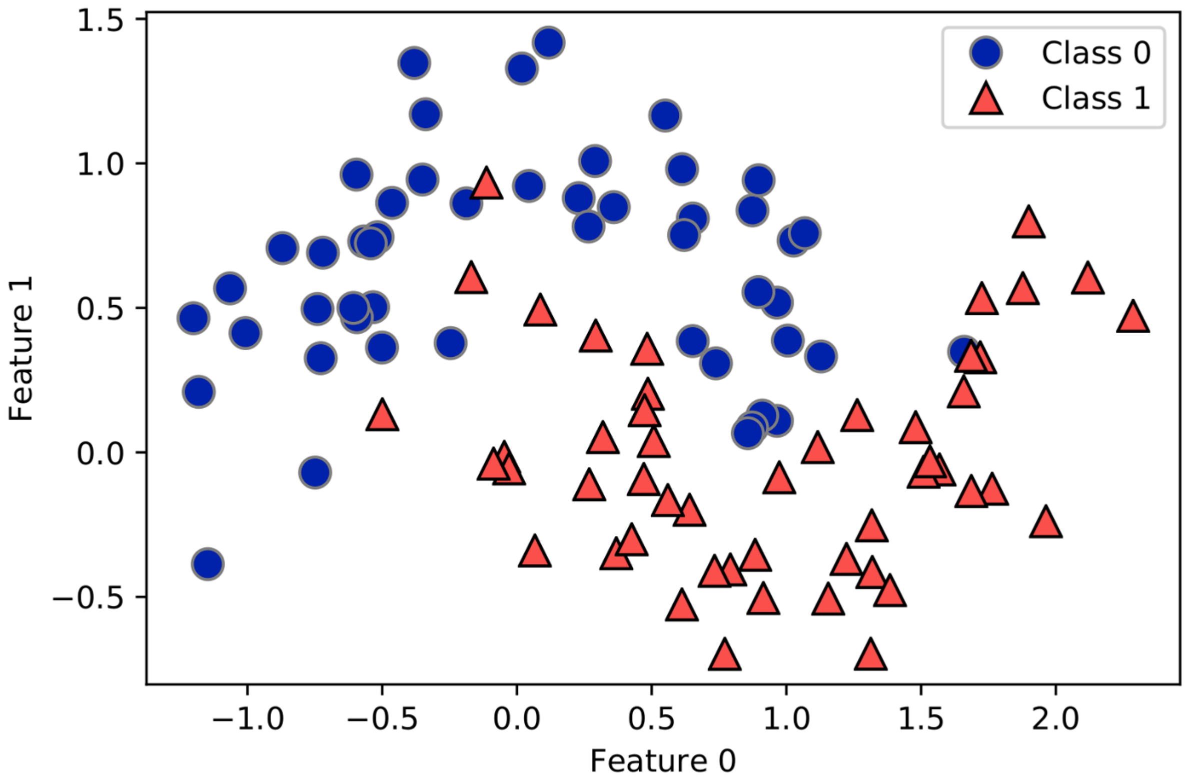


Figure 2-23. Two-moons dataset on which the decision tree will be built

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Decision Trees

- **At each step the algorithm selects the test that provides the best separation of the classes**

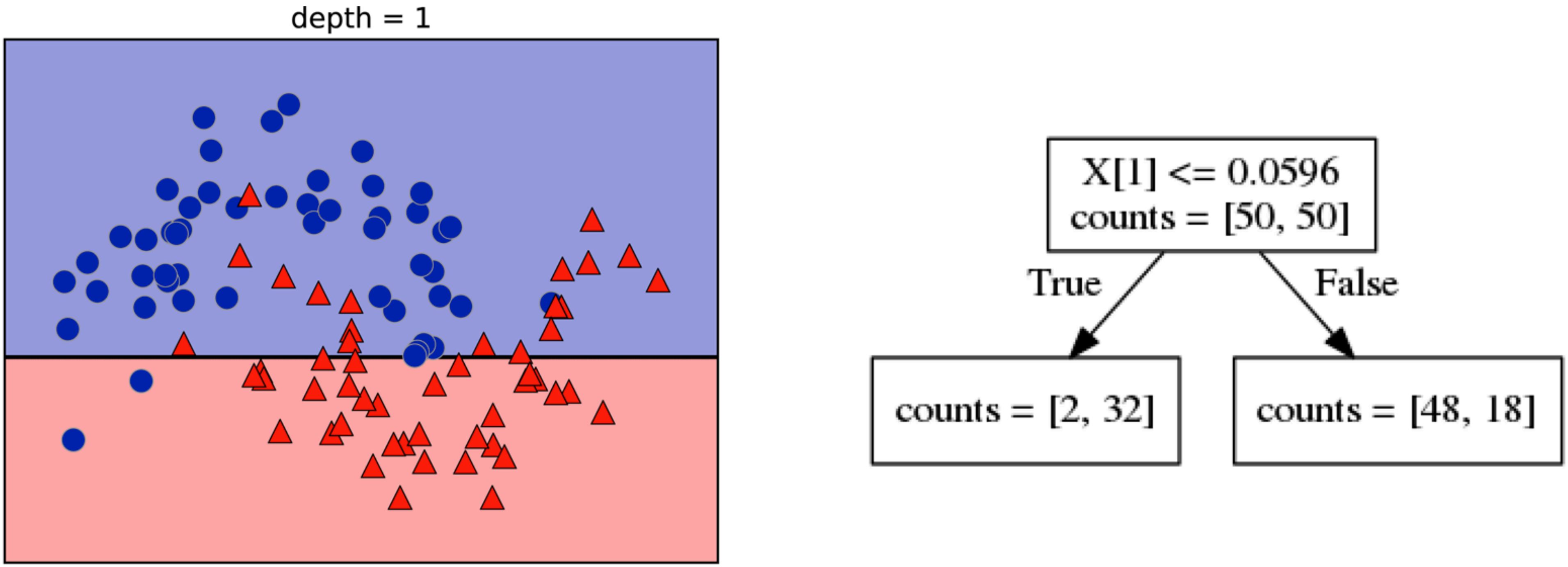


Figure 2-24. Decision boundary of tree with depth 1 (left) and corresponding tree (right)

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Decision Trees

- **At each step the algorithm selects the test that provides the best separation of the classes**

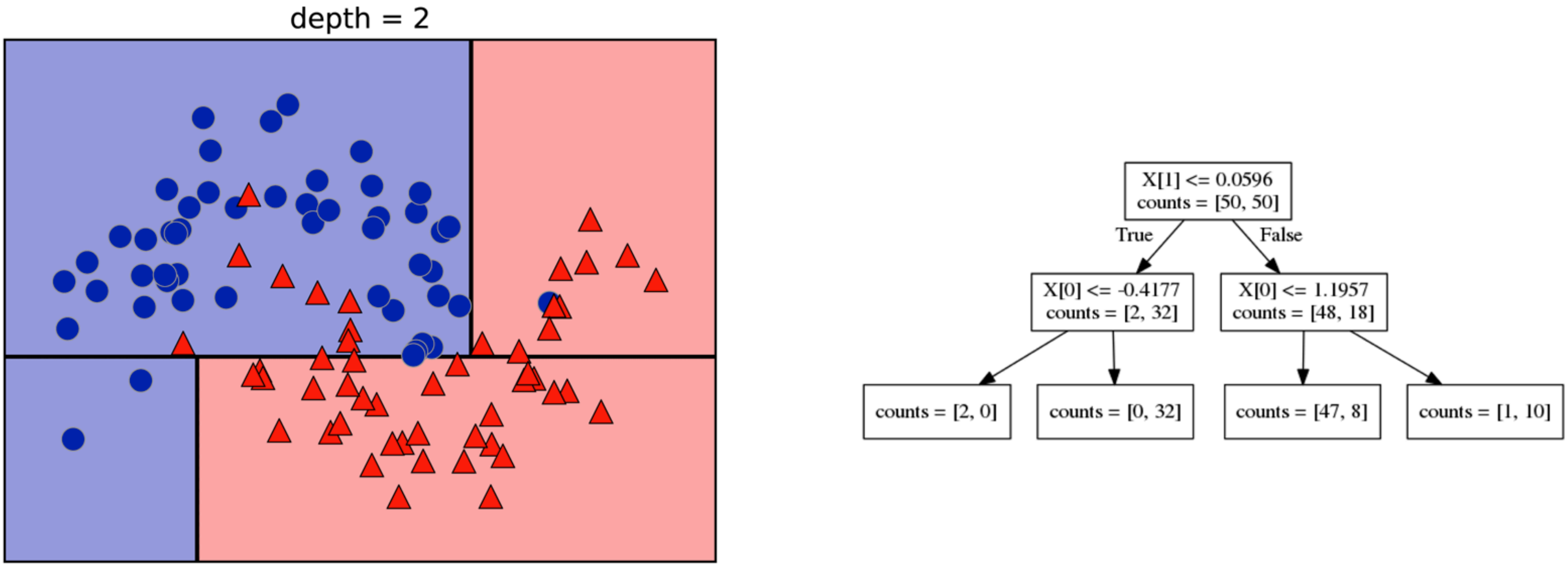


Figure 2-25. Decision boundary of tree with depth 2 (left) and corresponding decision tree (right)

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Decision Trees

- **Partitioning is repeated until each “leaf” contains only a single target (class or regression value) - referred to as pure**

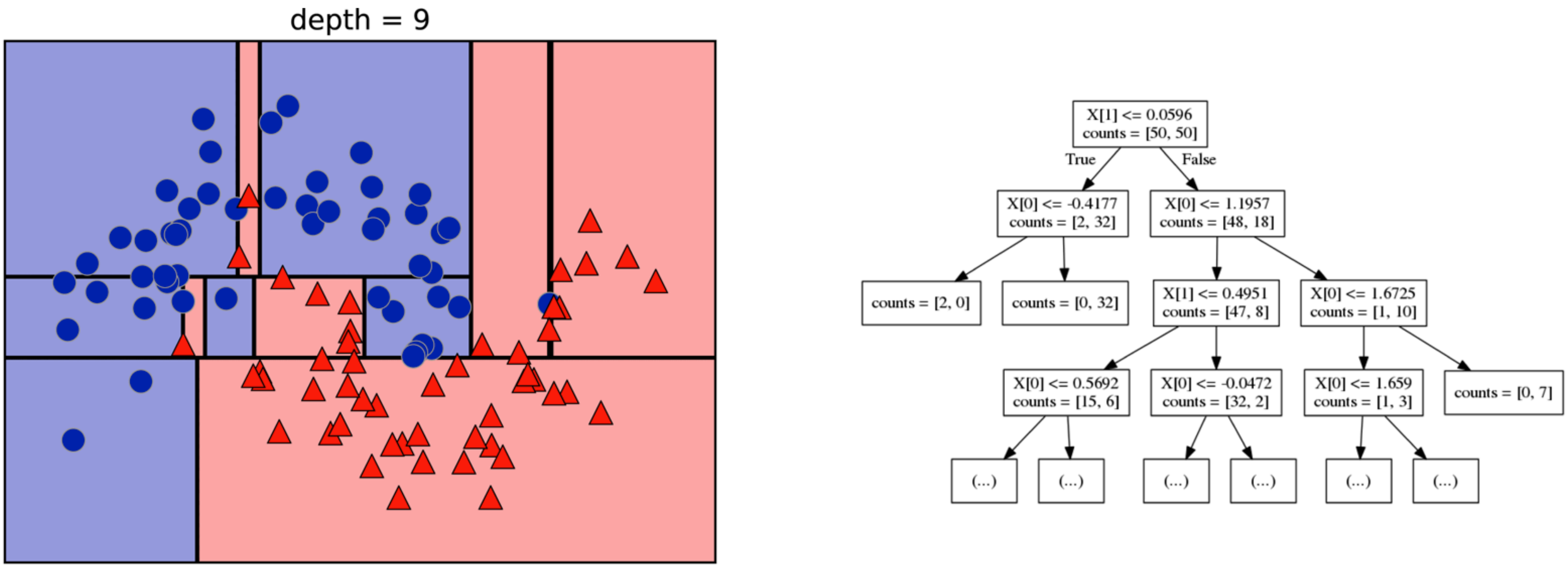


Figure 2-26. Decision boundary of tree with depth 9 (left) and part of the corresponding tree (right); the full tree is quite large and hard to visualize

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O’Reilly Media, Inc.

Decision Trees

- **Comprehensive tree building leads to overfitting the training data, this can be minimized by**
  - Pre-pruning - stopping the tree building early
    - Can be achieved by limiting the depth of the tree, number of leaves, or only splitting those with a certain number of points
  - Post-pruning/pruning - removing nodes that don’t contain much information

Jupyter Notebook 02-supervised-learning .ipynb [56-57]

Decision Trees

- **Visualizing the tree can provide information about how the algorithm makes decisions**

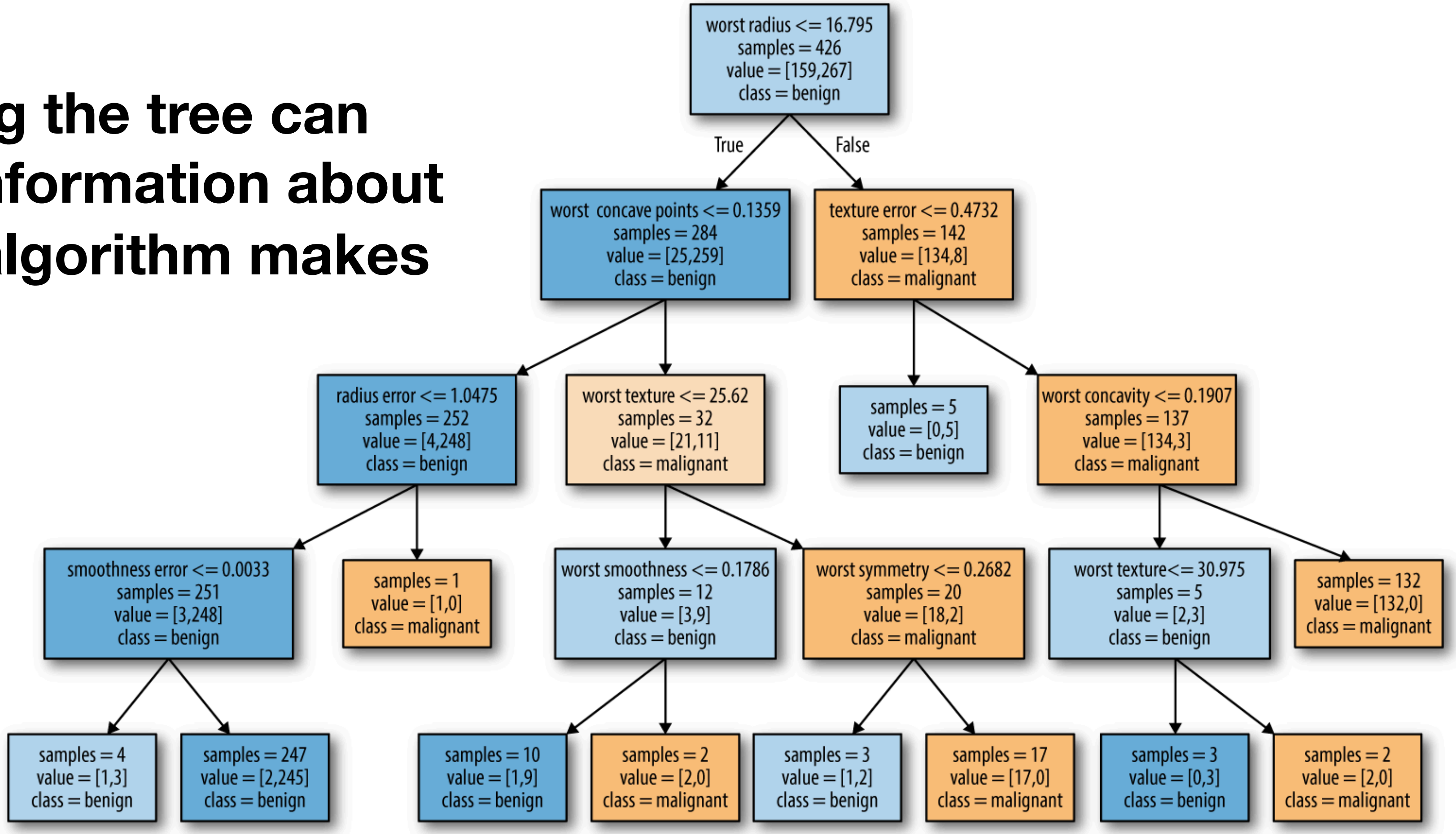


Figure 2-27. Visualization of the decision tree built on the Breast Cancer dataset

Jupyter Notebook 02-supervised-learning .ipynb [58-59]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Decision Trees

- **Feature importance is a summary of how important each feature is in the tree's decision making**
- **Feature importance isn't class specific**

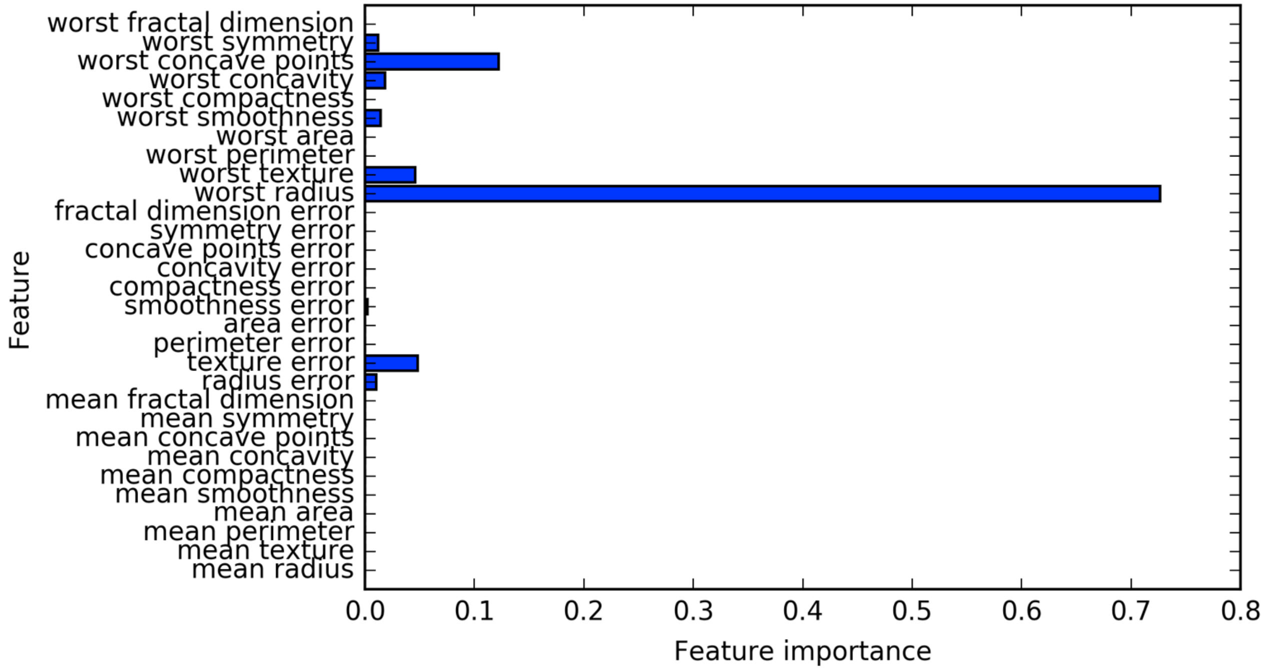
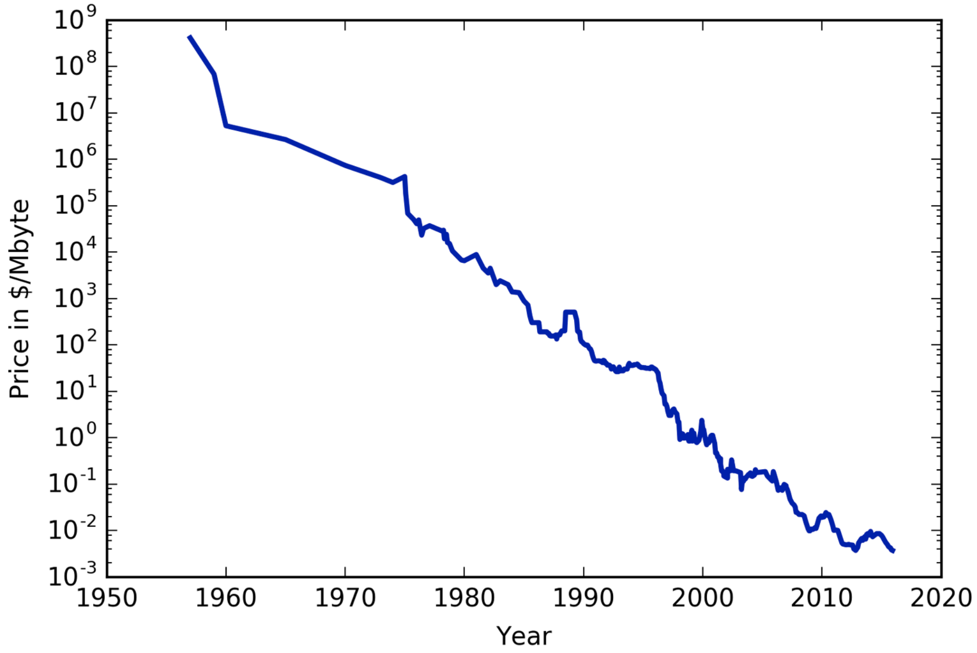


Figure 2-28. Feature importances computed from a decision tree learned on the Breast Cancer dataset

Jupyter Notebook 02-supervised-learning .ipynb [60-61]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Decision Trees Regression



- Not able to make predictions outside of the range of the training data (extrapolate)

Figure 2-31. Historical development of the price of RAM, plotted on a log scale

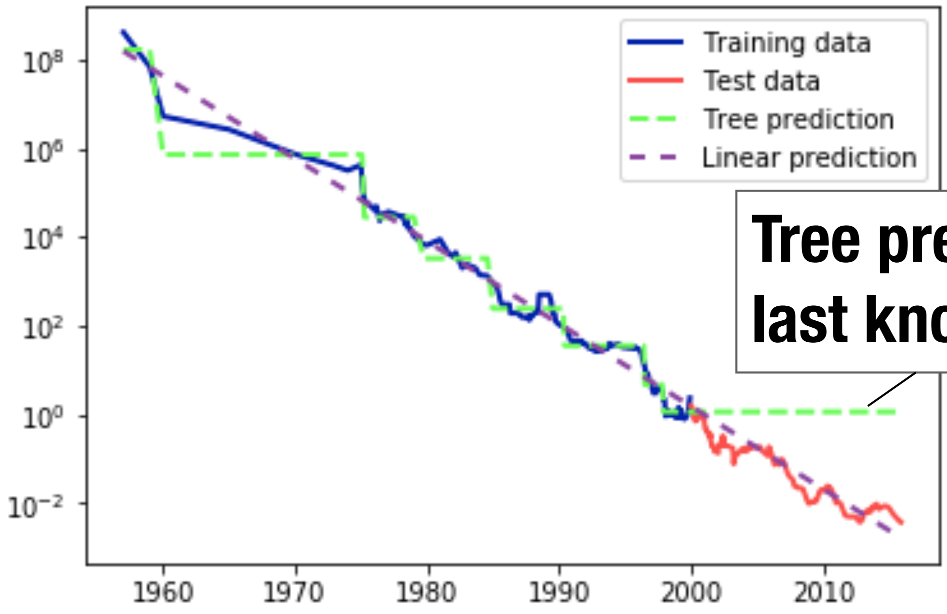


Figure 2-32. Comparison of predictions made by a linear model and predictions made by a regression tree on the RAM price data

Tree predicts the last known point

Jupyter Notebook 02-supervised-learning .ipynb [63-65]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Decision Trees

- **Parameters**
  - Maximum depth - for pre-pruning
- **Assumptions**
  - No assumptions about the distribution of the data
- **Best Practices**
  - Use of pre-pruning prevents overfitting
- **Strengths**
  - Can be visualized, which aids in interpretation
  - Invariant to scaling data
  - Works well with mixed data (e.g., binary and continuous features)
- **Weaknesses**
  - Tend to overfit (even with pre-prunning) and thus don’t always generalize
  - Doesn’t work well on high-dimensional sparse data

Random Forests

- **Ensemble of slightly different decision trees**
  - Averaging the predictions of the trees minimizing overfitting
  - Difference between trees is achieved by randomizing which data points or features are used

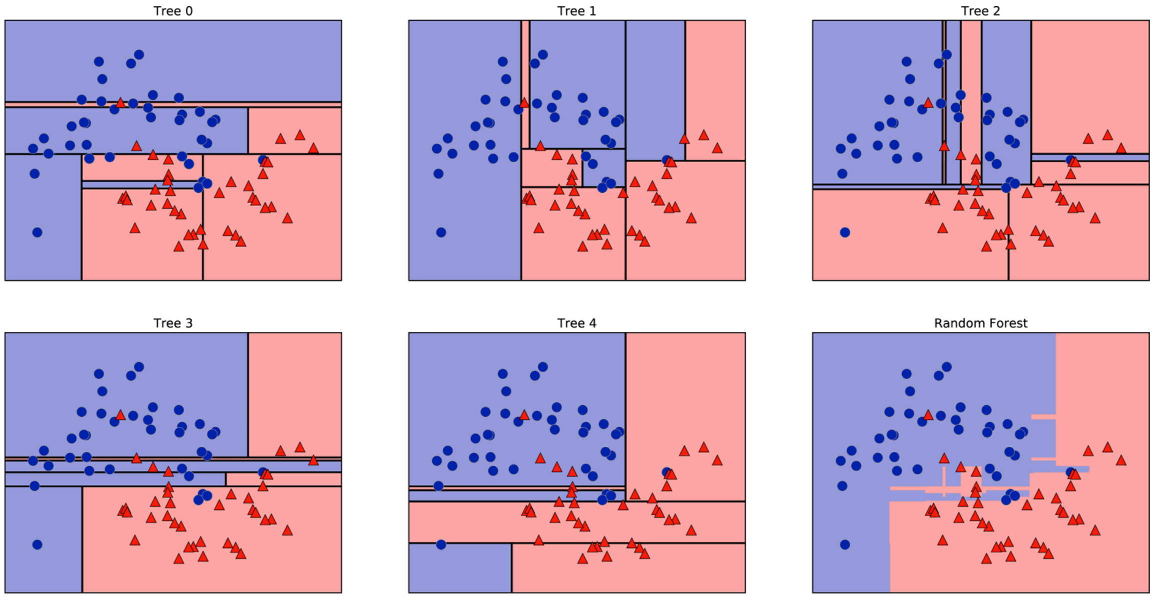


Figure 2-33. Decision boundaries found by five randomized decision trees and the decision boundary obtained by averaging their predicted probabilities

Jupyter Notebook 02-supervised-learning .ipynb [66-67]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Random Forests

- **Feature importance is aggregated across the ensemble of decision trees**
  - Typically more informative for random forests than for individual trees

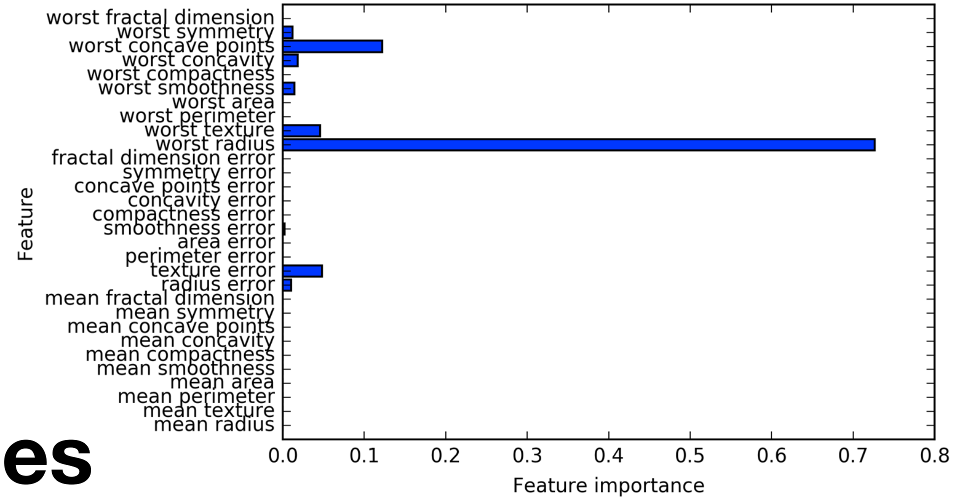


Figure 2-28. Feature importances computed from a decision tree learned on the Breast Cancer dataset

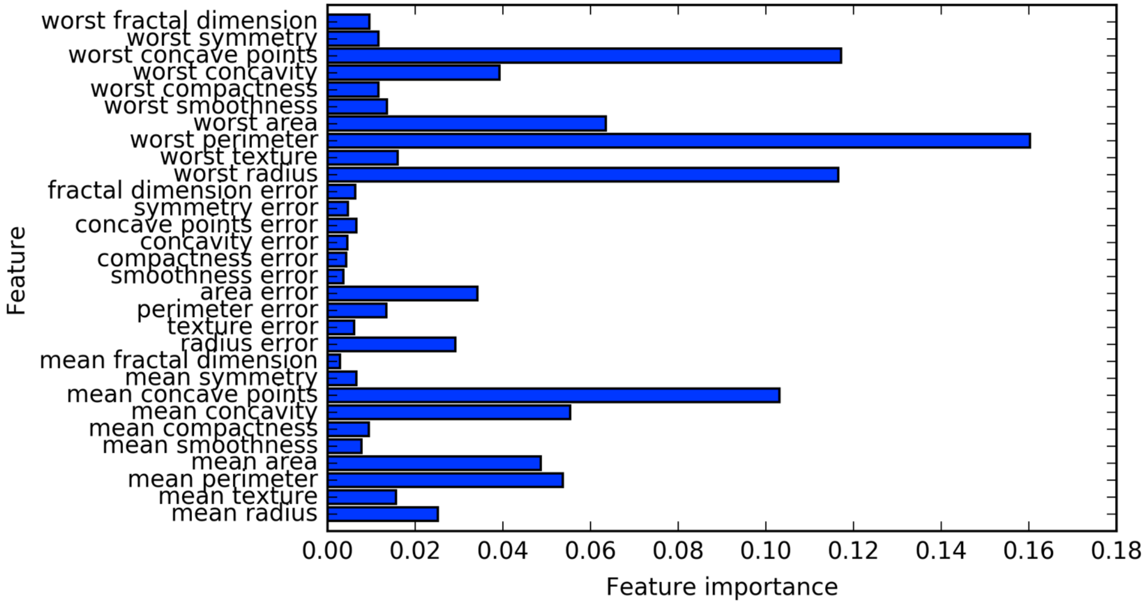


Figure 2-34. Feature importances computed from a random forest that was fit to the Breast Cancer dataset

Jupyter Notebook 02-supervised-learning .ipynb [68-69]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Random Forests

- **Parameters**
  - Number of estimators - larger is always better
  - Number of data points - number of samples drawn from training dataset
  - Maximum features - amount of randomness (smaller reduces overfitting)
  - Maximum depth - for pre-prunning
- **Strength**
  - Invariant to scaling data
  - Works well with mixed data (e.g., binary and continuous features)
  - Generalizes better than decision trees
- **Weaknesses**
  - Harder to interpret than decision trees
  - Can be time consuming to train
  - Random process can make reproducibility difficult

Gradient Boosted Random Forests

- Series of decision trees where each new tree tries to correct the mistakes of the previous trees using pre-pruning (rather than randomness)

- Trees are typically shallow (weak learners)

- Feature importances tend to be sparser than random forests

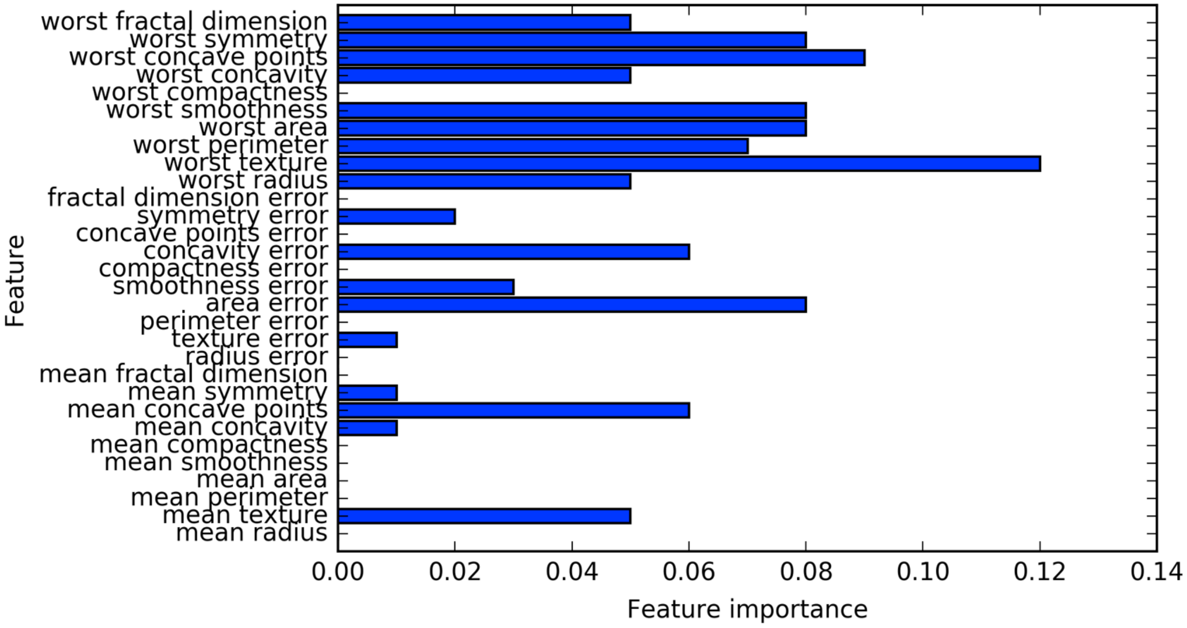


Figure 2-35. Feature importances computed from a gradient boosting classifier that was fit to the Breast Cancer dataset

Jupyter Notebook 02-supervised-learning .ipynb [70-73]

Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

Gradient Boosted Random Forests

- **Best Practices**
  - Random forests are a good place to start
    - Gradient-boosting can improve accuracy
- **Parameters**
  - Number of estimators - larger is always better
  - Maximum depth - for pre-prunning
  - Learning rate - how much a tree tries to correct the previous mistakes
- **Strengths**
  - Typically perform very well
  - Invariant to scaling data
  - Works well with mixed data (e.g., binary and continuous features)
- **Weaknesses**
  - Can take a long time to train
  - Doesn’t work well on high-dimensional sparse data