Week 10:
Decision Trees

CMPS 320 : Machine Learning

Decision Trees

- In todays lecture, we will describe decision trees for **regression** and **classification**.
- These involve stratifying or segmenting the predictor space into a number of simple regions.
- To make a prediction for a given observation, we typically use the mean or the mode of the training observations in the region to which it belongs.
- The set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as **decision tree methods**.

Decision Trees (cont.)

- Decision Trees algorithms are capable of fitting complex datasets.
- They are the fundamental components of Random Forests which are among the most powerful Machine Learning algorithms available today.
- In today's lectures, we will discuss how to train, visualize, and make predictions with Decision Trees.

Decision Trees (cont.)

- Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression.
- The objective is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- Decision Trees are intuitive, and their decisions are easy to interpret.
  - Such models are often called **white box models**.
- In contrast, as we will see, Random Forests or neural networks are generally considered **black box models**.
  - They make great predictions, and you can easily check the calculations that they performed to make these predictions;
  - However, it is usually hard to explain in simple terms why the predictions were made.

Decision Trees–Pros and Cons

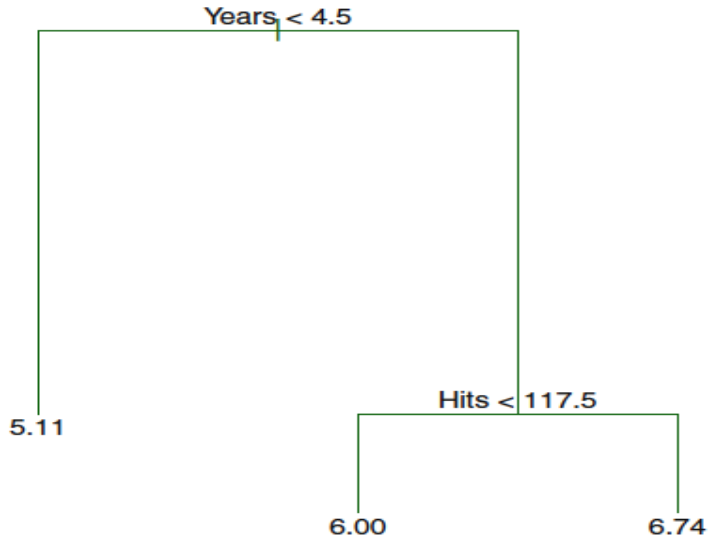
- Advantages:
  - Simple to understand and to interpret.
  - Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed.
  - Able to handle multi-output problems.
- Disadvantages:
  - Decision-tree learners can create over-complex trees that do not generalize the data well.
  - Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.
  - Decision tree learners create biased trees if some classes dominate.

Regression Trees: Predicting Baseball Players' Salaries

- To motivate **regression trees**, we begin with a simple example.
- We use the Hitters data set to predict a baseball player's Salary based on
  - Years (the number of years that he has played in the major leagues) and
  - Hits (the number of hits that he made in the previous year).

Regression Trees

- A regression tree for predicting the log salary of a baseball player:



- The top split assigns observations having Years < 4.5 to the left branch.
- The predicted salary for these players is given by the mean response value for the players in the data set with Years < 4.5.
- Players with Years ≥ 4.5 are assigned to the right branch, and then that group is further subdivided by Hits.

Regression Trees

- Overall, the tree stratifies or segments the players into three regions of predictor space:
  - players who have played for four or fewer years,
  - players who have played for five or more years and who made fewer than 118 hits last year, and
    - ★ players who have played for five or more years and who made at least 118 hits last year

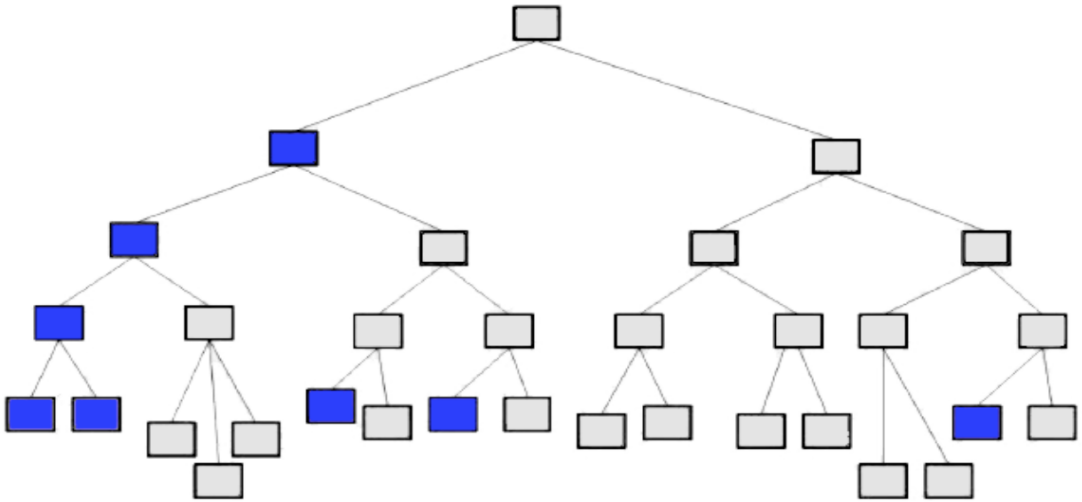
Regression Trees

- These three regions can be written as

$$R_1 = \{X | Years < 4.5\}$$
$$R_2 = \{X | Years \geq 4.5, Hits < 117.5\}$$
$$R_3 = \{X | Years \geq 4.5, Hits \geq 117.5\}$$

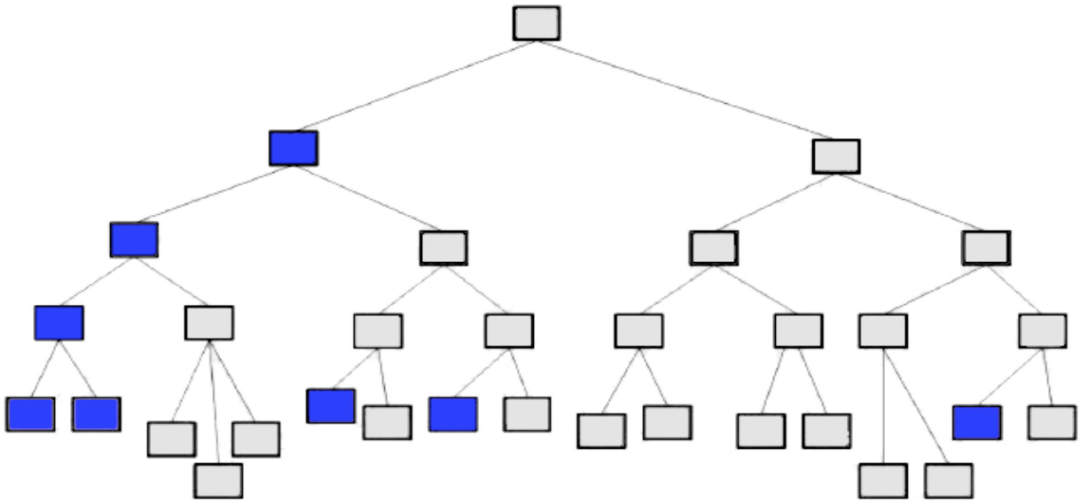
- The regions R_1, R_2, and R_3 are known as **terminal nodes** or **leaves** of the tree.
- The points along the tree where the predictor space is split are referred to as **internal nodes**.
  - The two internal nodes are indicated by the text Years < 4.5 and Hits < 117.5
- The segments of the trees that connect the nodes are known as **branches**.
- The number in each leaf is the mean of the response for the observations that fall there.

Regression Trees



- Question: what could go wrong with this approach?

Regression Trees



- Question: what could go wrong with this approach?
- Answer: the resulting tree might be too complex, leading to poor test

Growing (and pruning) trees

- Big idea: build a big tree, then **cut off ("prune")** the branches that aren't improving performance
- Question: why not just build a smaller tree to begin with?

Growing (and pruning) trees

- Big idea: build a big tree, then **cut off ("prune")** the branches that aren't improving performance
- Question: why not just build a smaller tree to begin with?
- Answer: this is the same issue we had with our linear model selection methods:
  - a branch that doesn't seem useful early on may give rise to a better branch further down - if we stop too soon.

Classification Trees

- A **classification tree** is very similar to a regression tree, except that it is classification used to **predict a qualitative response** rather than a quantitative one.
- For a regression tree, the predicted response for an observation is given by the mean response of the training observations that belong to the same terminal node.
- For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.
- In interpreting the results of a classification tree, we are often interested in:
  - the class prediction corresponding to a particular terminal node region and
  - proportions among the training observations that fall into that region.

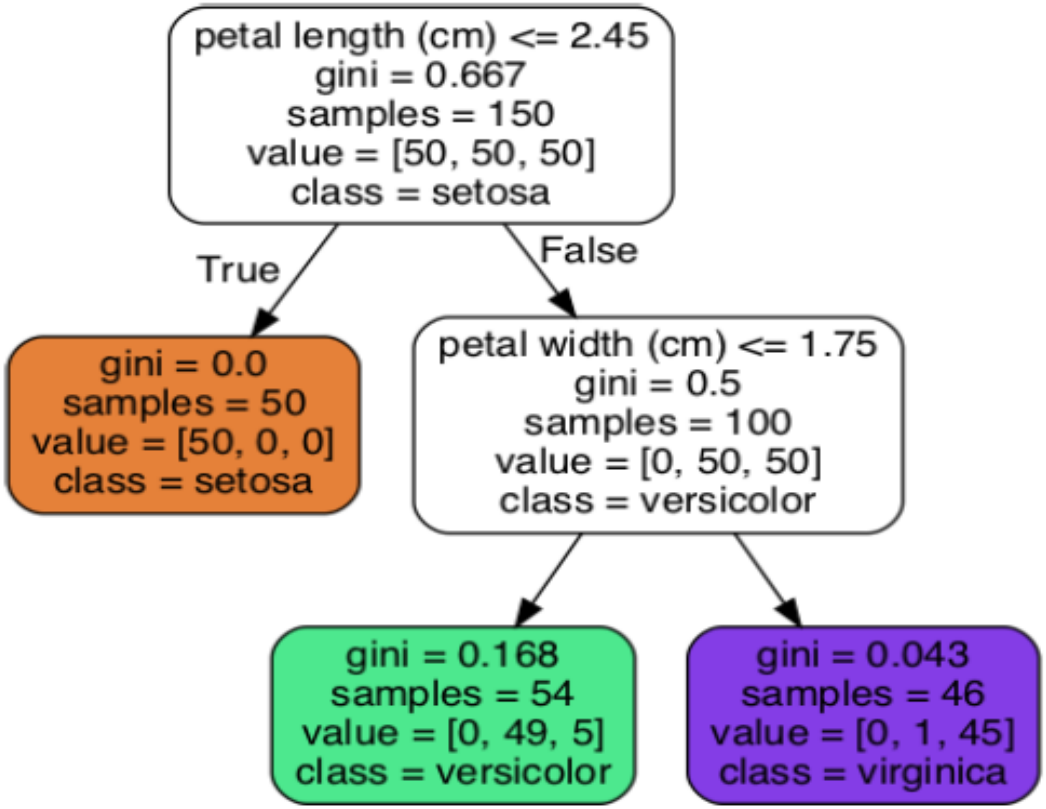
Making Predictions

- To understand classification of decision trees, we build one and take a look at how it makes predictions.
- Suppose you find an iris flower and you want to classify it.
- You start at the **root node** (depth 0, at the top): this node asks whether the flower's **petal length** is smaller than 2.45 cm.
  - If it is, then you move down to the root's **left child node** (depth 1, left).
  - In this case, it is a **leaf node** (i.e., it does not have any child nodes), so it does not ask any questions:
    - ★ It looks at the predicted class for that node, and the Decision Tree predicts that your flower is an **Iris setosa** (class=setosa).

Making Predictions (cont.)

- Now suppose you find another flower, and this time the **petal length** is greater than 2.45 cm.
  - We must move down to the root's **right child node** (depth 1, right), which is not a leaf node, so the node asks another question:
  - Is the **petal width** smaller than 1.75 cm?
    - ★ If it is, then your flower is most likely an **Iris versicolor** (depth 2, left).
    - ★ If not, it is likely an **Iris virginica** (depth 2, right).

Decision Tree- Diagram



Making Predictions (cont.)

- A **node's samples** attribute counts how many training instances it applies to.
  - For example, 100 training instances have a petal length greater than 2.45 cm (depth 1, right), and of those 100:
    - ★ 54 have a petal width smaller than 1.75 cm (depth 2, left)
    - ★ 46 have a petal width larger than 1.75 cm (depth 2, right)
- A **node's value** attribute tells you how many training instances of each class this node applies to:
  - For example, the bottom-right node applies to 0 Iris setosa, 1 Iris versicolor, and 45 Iris virginica.
- A **node's gini** attribute measures its impurity: a node is "pure" (gini=0) if all training instances it applies to belong to the same class.
  - For example, since the depth-1 left node applies only to Iris setosa training instances, it is pure and its gini score is 0.

Gini impurity score

- The **Gini impurity score** is defined as:

$$G_i = 1 - \sum_{k=1}^{n} P_{i,k}^2$$

  where $P_{i,k}$ is the ratio of class k instances among the training instances in the ith node.

- As an example, the depth-2 left node has a gini score equal to

$$G = 1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0.168.$$

Estimating Class Probabilities

- A Decision Tree can also estimate the probability that an instance belongs to a particular class k.
- First it traverses the tree to find the leaf node for this instance, and then it returns the ratio of training instances of class k in this node.
- For example, suppose we have found a flower whose petals are 5 cm long and 1.5 cm wide.
- The corresponding leaf node is the depth-2 left node, so the Decision Tree should output the following probabilities:
  - 0% for Iris setosa (0/54)
  - 90.7% for Iris versicolor (49/54) and
  - 9.3% for Iris virginica (5/54).
- It would predict Iris versicolor (class 1) because it has the highest probability.

The CART Training Algorithm

- Scikit-Learn uses the **Classification and Regression Tree (CART) algorithm** to train Decision Trees (also called **"growing"** trees).
- The algorithm works by first splitting the training set into two subsets using a single feature k and a threshold t_k (e.g., "petal length $\leq$ 2.45 cm").
- Next it splits the subsets using the same logic, then the sub-subsets, and so on, recursively.
- It stops recursing once it reaches the maximum depth (defined by the **max_depth** hyperparameter), or if it cannot find a split that will reduce impurity.

Equation 6-2. CART cost function for classification

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

where $\begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$

Computational Complexity

- Making predictions requires traversing the Decision Tree from the root to a leaf.
- Decision Trees generally are approximately balanced, so traversing the decision tree requires going through roughly $O(\log_2(m))$ nodes
- Since each node only requires checking the value of one feature, the overall prediction complexity is $O(\log_2(m))$, independent of the number of features.
- So predictions are very fast, even when dealing with large training sets.

Regularization Hyperparameters

- Decision Trees make very few assumptions about the training data (as opposed to linear models, which assume that the data is linear, for example).
- If left unconstrained, the tree structure will adapt itself to the training data, fitting it very closely-indeed, most likely **overfitting** it.
- To avoid overfitting the training data, you need to restrict the Decision Tree's freedom (regularization) during training.

Regularization Hyperparameters (cont.)

- In Scikit-Learn, this is controlled by the max_depth hyperparameter (the default value is None, which means unlimited).
  - Reducing max_depth will regularize the model and thus reduce the risk of overfitting.
- Other parameters that can be used to restrict the shape of the decision tree are:
  - min_samples_split (the minimum number of samples a node must have before it can be split),
  - min_samples_leaf (the minimum number of samples a leaf node must have) etc.