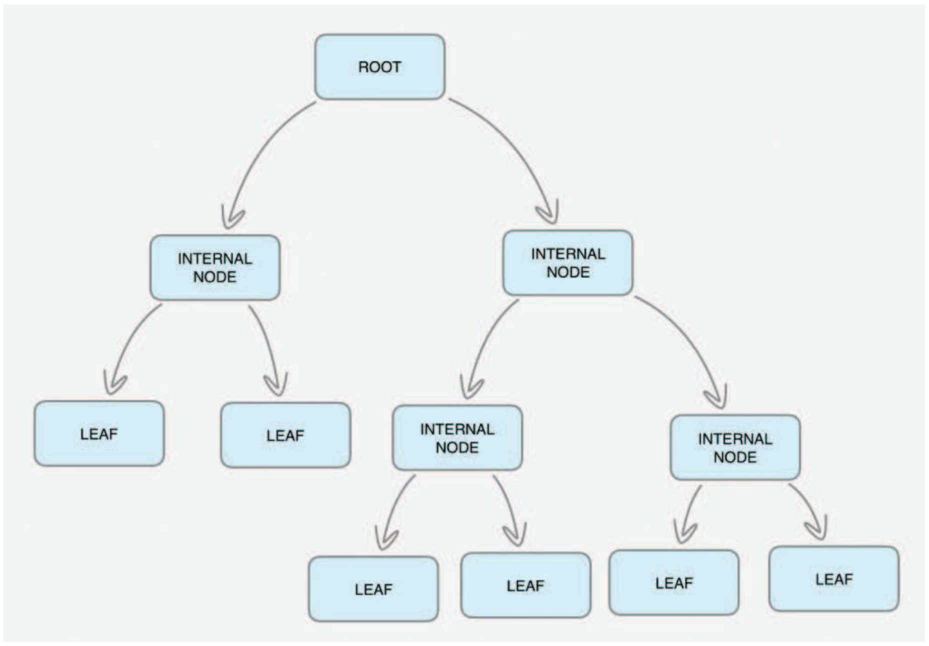
10. DECISION TREES

- Decision tree is a an example of a supervised machine learning algorithm used both for classification and regression. When a decision tree classifies data into categories, it is called a *Classification Tree* (predicting if an email is a spam or not) and when it predicts numerical values, it is called a *Regression Tree* (predicting the effectiveness of a medication based on its dosage).
- Decision trees work well with both quantitative and qualitative features and don't require feature scaling or centering.
- Sklearn uses the CART algorithm which produces only binary trees (nonleaf nodes have only two children). Other algorithms, such as ID3 can produce identification decision trees with nodes that have more than two children. These algorithms work by *greedily* constructing a partition of the input space into distinct and non-overlapping regions, where the splits are *axis aligned*, and by fitting a *constant* model in each region.
- The decision tree contains a root node, internal nodes, leaves, and edges.
  - Root: The node where the first split takes place.
  - Edges: They direct the outcome of a split to the next nodes. If we have two edges and if the statement in the node is true, we go to the left, and if the statement is false, we go to the right.
  - Internal Nodes: The nodes where the tree splits according to the value of some attribute/feature of the dataset.
  - Leaves: The terminal nodes that predict the outcome.



https://blog.quantinsti.com/decision-tree/

1 Regression Trees

Consider a training data set $\{(x^{(1)}, y^{(1)}) \dots, (x^{(n)}, y^{(n)})\}$, consisting of features $x = (x_1, \dots, x_d)$ which can be categorical or numerical, and a target variable $y \in \mathbb{R}$.

We build a regression tree by

- dividing the input space $\mathbb{R}^d$ into M distinct and non-overlapping regions $R_1, \dots, R_M$ with boundaries being axis aligned, and
- for every observation that falls into the region R_m we make the same prediction which is the average of the target values for the training data in R_m

$$\hat{y}_m = \frac{1}{|R_m|} \sum_{x^{(i)} \in R_m} y^{(i)}$$

The goal is to find partition $R_1, \dots, R_M$ that minimizes the squared error

$$\boxed{\sum_{m=1}^{M} \sum_{x^{(i)} \in R_m} (y^{(i)} - \hat{y}_m)^2}$$

This problem is known to be NP-complete, and we take the *top-down greedy approach* that is known as recursive binary splitting. It is greedy because at each step the best split is made at that particular step rather than looking ahead and choosing a split that will produce a better tree in a long run.

Given the data, the splitting condition is determined in two steps.

1. We measure the impurity of a node by finding its MSE (or variance)

$$\boxed{Var(\text{node}) = \frac{1}{k} \cdot \sum_{i=1}^{k} (y^{(i)} - \bar{y})^2} \quad \text{where} \quad \bar{y} = \frac{1}{k} \sum_{i=1}^{k} y^{(i)}$$

   Note that higher variance implies higher impurity.

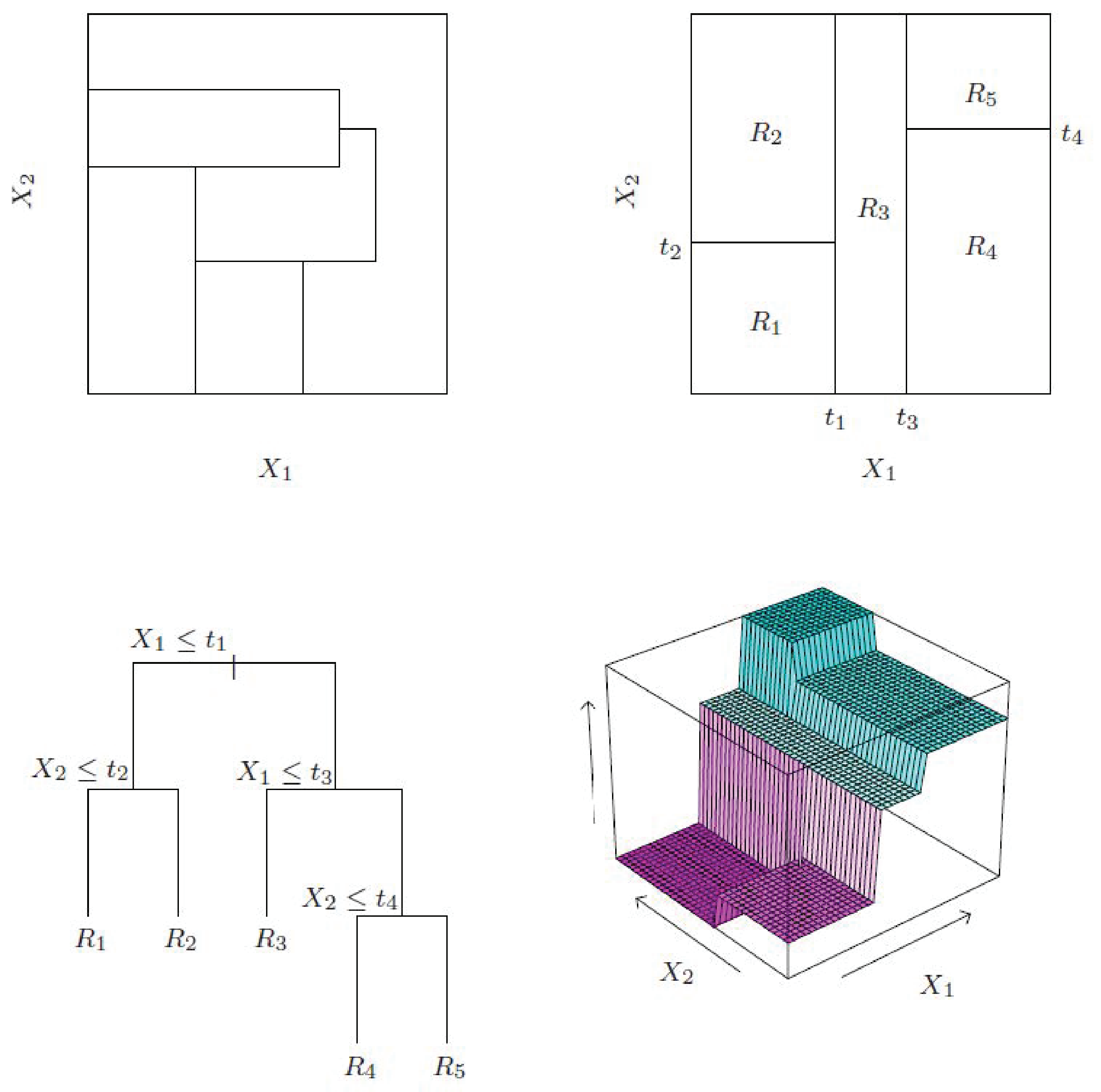
2. We compute the *Variance Reduction* from a parent node to its two children nodes

$$\boxed{VarRed = Var(\text{parent}) - w_1 \cdot Var(\text{child}_1) - w_2 \cdot Var(\text{child}_2)}$$

   and we pick a split that **maximizes VarRed**. Here, w_1 and w_2 are relative sizes of children nodes to the parent node.

This process continues until a stopping criterion is reached.

Given a new data instance with features, we run these features through the tree, and see in which leaf we end up to make the prediction.


X_2
X_1
R_2
R_5
t_4
R_3
R_4
t_2
R_1
t_1
t_3
X_2
X_1
$X_1 \le t_1$
$X_2 \le t_2$
$X_1 \le t_3$
$X_2 \le t_4$
R_1
R_2
R_3
R_4
R_5
X_2
X_1

Remark: We can keep splitting the tree until each node contains only one data instance. In that case the squared error on the training data will be 0; however, this tree will not work well on the test data. This is an example of over-fitting and there are generally two methods to deal with this in case of decision trees.

- *tuning the hyper-parameters using cross-validation*
  - limiting the depth of the tree
  - requiring that the leaves need to contain a certain minimum number of data points, etc.
- *pruning the tree* – we grow a large tree and we prune it to obtain a subtree

  We grow a large tree and to find the best subtree, we can estimate the test error for each subtree using cross-validation, however, since the number of subtrees can be extremely high, this is not computationally feasible.

  Another approach is to use *cost complexity pruning* or *weakest link pruning*.

  1. Use recursive binary splitting to grow a large tree T_0 on the training data stopping only before each terminal node has fewer than some minimum number of observations set in advance.
  2. For each nonegative hyper-parameter α, find a subtree $T \subset T_0$ that minimizes

     $$\sum_{m=1}^{|T|} \sum_{x^{(i)} \in R_m} (y^{(i)} - \hat{y}_m)^2 + \alpha\, |T|$$

     where $|T|$ is the number of terminal nodes, R_m is the region in the input space corresponding to the mth terminal node and $\hat{y}_m$ is the predicted response associated with R_m (average of target values in regression). The hyper-parameter α is a trade-off between the subtree's fit to the training data and its complexity. If $\alpha = 0$, then $T = T_0$ and when α increases, there is a price to pay for having a tree with many terminal nodes. Note the similarity of this cost function to lasso regression.
  3. Use K-fold cross-validation to select α. For each $k = 1, \ldots, K$
     - repeat steps 1 and 2 on all but the kth fold of the training data
     - evaluate the MSE error on the data in the left-out kth fold as a function of α

     Average the results for each α and pick α to minimize the average error.
  4. Return the subtree from step 2 for the chosen value of α.

2 Classification Trees

Example: Assume a doctor's office is collecting records on patients to predict if someone will have a chance of a heart attack or not. The features include information whether a patient exercises or not, whether they have a high blood pressure or not, and their age.

Patient	Exercises	High Blood Pressure	Age	Had a Heart Attack
P1	Yes	Yes	7	No
P2	Yes	No	12	No
P3	No	Yes	18	Yes
P4	No	Yes	35	Yes
P5	Yes	Yes	38	Yes
P6	Yes	No	50	No
P7	No	No	55	No

When a new patient comes to the office with

$$\text{Exercises} = \text{"Yes"} \quad \text{High Blood Pressure} = \text{"Yes"} \quad \text{Age} = \text{"51"}$$

we want to use the given data to predict whether this patient is at risk of having a heart attack.

How do we build a Classification Tree?

Given a training data set $\{(x^{(1)}, y^{(1)}) \ldots, (x^{(n)}, y^{(n)})\}$ consisting of features $x = (x_1, \ldots, x_d)$ which can be categorical or numerical, and categorical target variable y, we first decide which feature and splitting condition to use in the root node by determining which feature predicts y the best. We utilize a technique known as the Attribute Selection Measure (ASM). There are two commonly used ASM techniques to measure impurity of a node: entropy and gini index.

- **Entropy** signifies the randomness (or uncertainty or impurity) in the dataset and, for a given node, it is defined by:

  $$\boxed{E(\text{node}) = -p_1 \cdot \log_2 p_1 - p_2 \cdot \log_2 p_2} \qquad \text{binary classification with 2 classes}$$

  $$\boxed{E(\text{node}) = -\sum_{i=1}^{k} p_i \cdot \log_2 p_i} \qquad \text{multi-class classification with } k \text{ classes}$$

  where p_i is the probability of the ith class. In general, p_i's are fractions that add up to 1 and represent the percentage of each class present in the node. We will assume $0 \cdot \log_2 0 = 0$.

  For simplicity, consider binary classification and note:

  – if a node is pure (all of its data points are in the same class), its entropy is

  $$E = -1 \cdot \log_2 1 - 0 \cdot \log_2 0 = 0.$$

  – if we have a mixed node with 50% of data values in each of the two classes, its entropy is

  $$E = -0.5 \cdot \log_2 0.5 - 0.5 \cdot \log_2 0.5 = 1.$$

- **Gini index** is defined as the probability of misclassifying an observation and is defined by

  $$\boxed{\text{Gini(node)} = 1 - p_1^2 - p_2^2} \qquad \text{binary classification with 2 classes}$$

  $$\boxed{\text{Gini(node)} = 1 - \sum_{i=1}^{k} p_i^2} \qquad \text{multi-class classification with } k \text{ classes}$$

  For simplicity, consider binary classification and note

  - if a node is pure, its gini impurity is

    $$\text{Gini} = 1 - 1^2 - 0^2 = 0$$

  - if we have a node with $50 - 50$ split, its gini impurity is

    $$\text{Gini} = 1 - 0.5^2 - 0.5^2 = 0.5.$$

- If the variable is categorical, finding the entropy or gini impurity for a decision node is straightforward.

  If the variable is numerical, we sort its values from the lowest to highest, find the average between each two adjacent values, and calculate the entropy/gini for each of the average values. The decision with the smallest entropy/gini wins.

- Gini is less computationally expensive compared to entropy because entropy requires calculating logarithms, but entropy sometimes gives slightly more accurate results. Even though the trees might look different whether we use entropy or gini (gini tends to isolate the most frequent class in its own branch of the tree while entropy tends to produce slightly more balanced trees), the decisions they produce are very similar.

The process of deciding which attribute and splitting condition should be used at a parent node involves finding the **information gain**

$$\boxed{IG = I(\text{parent}) - w_1 \cdot I(\text{child}_1) - w_2 \cdot I(\text{child}_2)}$$

where I can be entropy or gini index and w_i is percentage of data points in the ith child node relative to the parent node. We pick a split that **maximizes IG** since IG tells how important a given attribute is.

IG is based on the concept of entropy and information content from information theory. IG is the measurement of changes in entropy after the splitting of the dataset based on a feature and it tells how much information that feature provides. Decision tree always tries to maximize the value of the information gain, and a node/feature having the highest value of the information gain is being split first.

We continue splitting until we reach pure nodes and they are called leaves because there is no reason to split them further or if a stopping criterion is reached. The output of the leaf is the majority category it contains.

Given a new data instance with features, we run these features through the tree, and see in which leaf we end up to make the prediction.

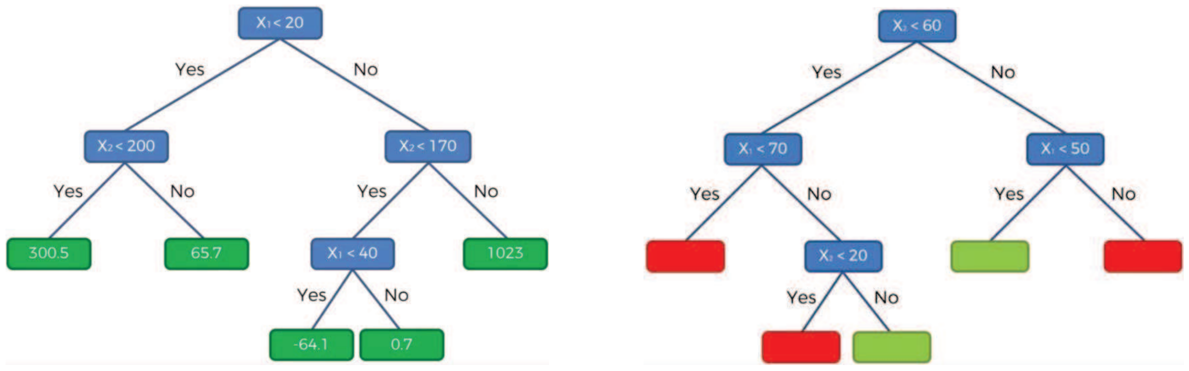
The classification tree for the above example, using gini index, is presented in the APPENDIX at the end of this handout.

Remark: The appropriate error in pruning the classification tree is the misclassification error (instead of MSE in the regression tree)

$$\frac{|\{i \,|\, x^{(i)} \in R_m \text{ and } y^{(i)} \neq \hat{y}_m\}|}{|R_m|}$$

where $|R_m|$ is the number of data points in the region R_m and $\hat{y}_m$ is the predicted label for points in R_m (majority class).

3 Summary of Decision Trees



https://medium.com/swlh/decision-tree-regression-and-its-mathematical-implementation-58c6e9c5e88e

- Decision tree is a non-parameteric model that works by greedily constructing a split. It does not check whether that split will lead to the lowest possible impurity several levels down. A greedy algorithm produces a solution that is good, but not always the best. Finding the best tree is an NP-complete problem.
- The splits are axis aligned; however, there are variations where non-axis-parallel splits are used (for example, run a perceptron to get a split).
- In case of regression, a constant is fit in each leaf; however, it is possible to fit linear regression or other regression method in each leaf.

Advantages:

- Easily interpretable.
- Fast to train.

  For balanced trees, traversing requires going through $O(\log_2 n)$ nodes, where n in the number of data points, and since we need to check the value of only one feature in each node, the prediction complexity is $O(\log_2 n)$.

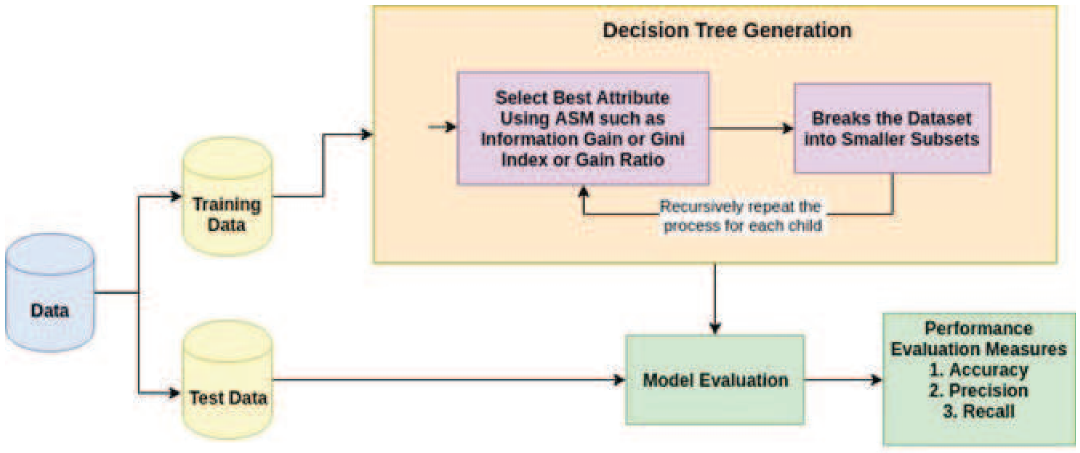
Since training requires comparing all features of all training data instances at each node, the training complexity is of $O(d \times n \times \log_2 n)$, where d is the number of features.

- There is very little need for data cleaning in decision trees compared to other ML algorithms. In fact, they don't require feature scaling or centering at all.
- Handles both numerical as well as categorical values.

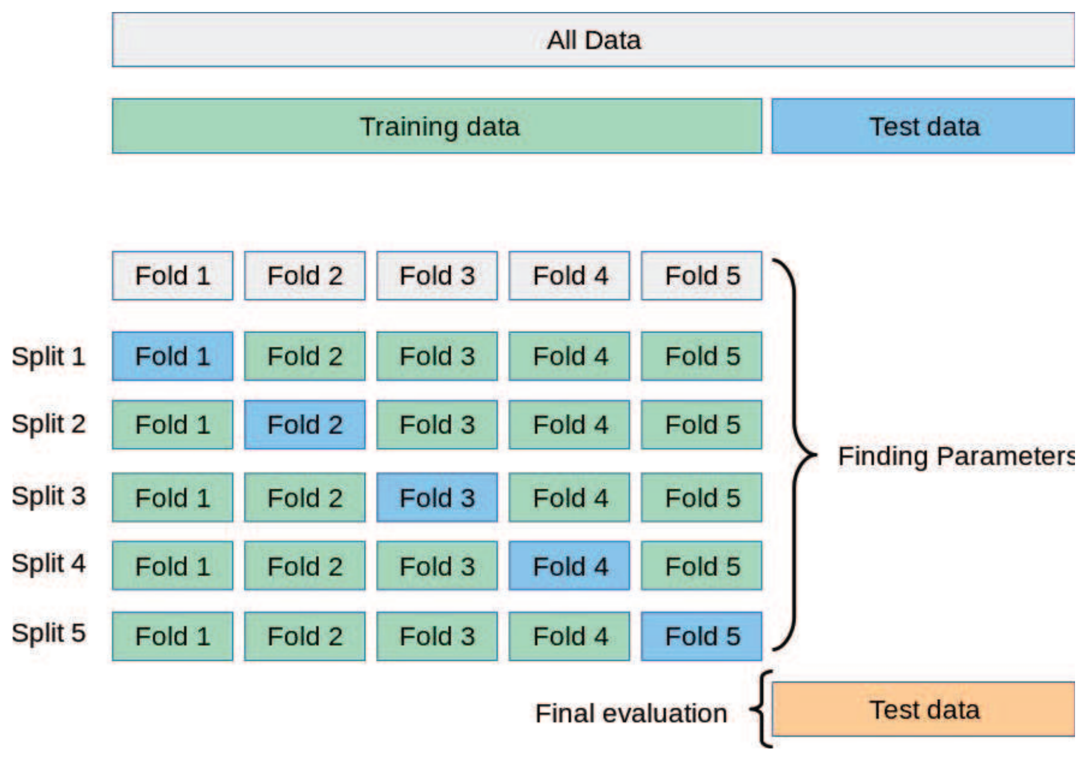
Disadvantages:

- High estimation error (or variance): small changes in data can result in a very different tree.
- It may result in over-fitting.

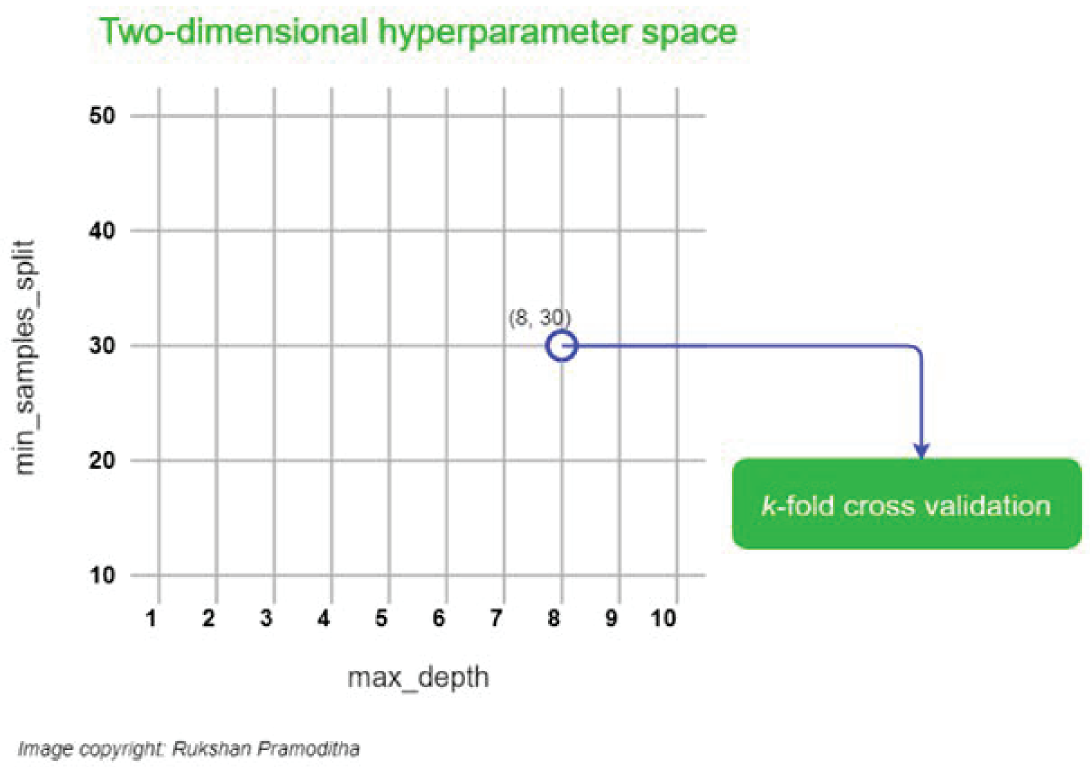
Note: training, validation, and test data; gridsearch for optimal hyper-parameters


Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53



https://scikit-learn.org/stable/modules/cross_validation.html



Python code: Lecture_9_DTC.ipynb and Lecture_9_DTR.ipynb

Homework 6:

- Create a python code for performing the regression tree algorithm on the `cali_housing.csv` data set.

  1. Import the data set into pandas data frame and inspect it. This data set provides information on the following variables.
     - MedInc: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
     - HouseAge: House age; a lower number is a newer building
     - AveRooms: Average number of rooms in houses within a block
     - AveBedrms: Average number of bedrooms in houses within a block
     - Population: Total number of people residing within a block
     - AveOccup: Average occupancy in houses within a block
     - Latitude: A measure of how far north a house is; a higher value is farther north
     - Longitude: A measure of how far west a house is; a higher value is farther west
     - MedHouseVal: Median house value for households within a block (measured in hundreds of thousands of US Dollars)

     The goal of this assignment is to build a regression tree and use some or all of the explanatory variables to predict the median house value.
  2. Select one or more explanatory variables you would like to use by looking at the scatter plot how each of those explanatory variables affects the target variable.
  3. Split the data into training and testing sets.
  4. Figure out if there are any missing values in the explanatory variables you want to use. If there are any missing values, you can either delete those data instances from the data set or fill in the missing values. If a numerical variable has missing values, you might fill those in with the average or median of that variable. If a categorical variable has missing values, you might fill those in using the most common value.
  5. Build a regression tree using the training data.
  6. Inspect the evaluation measures such as MAE, MSE, or RMSE.
  7. Modify the tree hyper-parameters (such as `criterion`, `max_depth`, `min_samples_split`, and `min_samples_leaf`) of your model to increase the quality of the prediction. You might plot graphs showing error for different values of these parameters.
  8. Use `GridSearchCV()`, `RandomizedSearchCV()`, or pruning to find the best model.
  9. Compute the MAE, MSE, or RMSE for your model.
  10. Determine which features are the most important in your model.
  11. Take some values for the explanatory variables and use your model to predict the median house value.

APPENDIX

Example: Build the decision tree using gini index.

Patient	Exercises	High Blood Pressure	Age	Had a Heart Attack (LABEL)
P1	Yes	Yes	7	No
P2	Yes	No	12	No
P3	No	Yes	18	Yes
P4	No	Yes	35	Yes
P5	Yes	Yes	38	Yes
P6	Yes	No	50	No
P7	No	No	55	No

We start by thinking which of the three attributes to use in the root note. In the root node we have 7 patients with labels: 3 yes and 4 no, meaning 3 patients had the heart attack and 4 did not have the heart attack. The gini impurity for this node is:

$$\text{Gini (parent)} = 1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2 = \boxed{0.49}.$$

- **Step 1:** If we use the attribute "exercises", we will split the data into two child nodes. If the person exercises, we place him/her in the left child node and if the person does not exercise we place him/her in the right child node.
  - The child on the left contains 4 patients of which 1 has the heart attack and 3 did not have the heart attack. The gini impurity for this node is

    $$\text{Gini}(\text{child}_1) = 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0.375$$

  - The child on the right contains 3 patients of which 2 had heart attack and 1 did not have the heart attack. Its gini impurity is

    $$\text{Gini}(\text{child}_2) = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 0.444$$

  Now, we calculate the information gain resulting from this split (which measures the difference in gini impurity before and after the split):

  $$\text{IG(exercises)} = 0.49 - \frac{4}{7} \cdot 0.375 - \frac{3}{7} \cdot 0.444 = \boxed{0.085} \tag{1}$$

- **Step 2:** If we use the attribute "high blood pressure", we will split the data into two child nodes. If the person has a high blood pressure, we place him/her in the left child node and if the person does not have a high blood pressure we place him/her in the right child node.
  - The child on the left has 4 patients (3 had the heart attack and 1 did not have the heart attack). The gini impurity for this node is

    $$\text{Gini}(\text{child}_1) = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$$

  - The child on the right has 3 patients (0 who had heart attack and 3 who did not have the heart attack). Its gini impurity is

$$\text{Gini}(\text{child}_2) = 1 - \left(\frac{0}{3}\right)^2 - \left(\frac{3}{3}\right)^2 = 0$$

Now, we calculate the information gain resulting from this split:

$$\text{IG}(\text{high blood pressure}) = 0.49 - \frac{4}{7} \cdot 0.375 - \frac{3}{7} \cdot 0 = \boxed{0.276} \qquad (2)$$

- **Step 3:** Lastly, we consider the attribute "age". However, this variable is numerical and calculating the information gain is a little bit more involved. First, extract only the columns "age" and "label" and sort the two columns from the lowest to the highest age. Next, find the average between each two adjacent value. For example, the average of 7 and 12 is 9.5. You will get 9.5, 15, 26.5, 36.5, 44, and 52.5. Now we look for each of these average values and calculate its information gain.

  - For the condition "age $<$ 9.5", we split the data (all 7 patients) into two child nodes. The child node on the left contains patients of age less than 9.5 and the child node on the right contains patients of age above 9.5.

    * The child node on the left contains only 1 patient. This patient did not have the heart attack, so in this node we have 0 patients who had the heart attack and 1 patient who did not have the heart attack. The gini impurity is

$$\text{Gini}(\text{child}_1) = 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2 = 0$$

    * The child node on the right contains 6 patients of which 3 had the heart attack and 3 did not have the heart attack. Its gini impurity is

$$\text{Gini}(\text{child}_2) = 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5$$

    The information gain for this split is

$$\text{IG (age} < 9.5) = 0.49 - \frac{1}{7} \cdot 0 - \frac{6}{7} \cdot 0.5 = 0.061$$

  - Similarly, we find IG for the remaining conditions "age $<$ 15", "age $<$ 26.5", "age $<$ 36.5", "age $<$ 44", and "age $<$ 52.5". The results are given in the table on the next page.

    The condition with the highest information gain wins. We have two such conditions "age $<$ 15" and "age $<$ 44", so we can pick either one. Let's say we pick the first one. Therefore, the best splitting condition for the attribute "age" is

$$\text{IG}(\text{age} < 15) = \boxed{0.147} \qquad (3)$$

Now we compare the information gain for each of the three attributes (see lines labeled (1),(2), and (3)) and we pick the attribute with the highest information gain. It is the "high blood pressure". Now, our tree has in the root all 7 patients and it is split according to the high blood pressure into

Condition	Information Gain
age < 9.5	0.061
age < 15	0.147
age < 26.5	0.014
age < 36.5	0.014
age < 44	0.147
age < 52.5	0.061

Patient	Exercises	High Blood Pressure	Age	Had a Heart Attack (LABEL)
P1	Yes	Yes	7	No
P3	No	Yes	18	Yes
P4	No	Yes	35	Yes
P5	Yes	Yes	38	Yes

two child nodes.

Note that the node on the right is pure (all patients in that node did not have the heart attack), so we don't split it further. However, we continue splitting the first child node. This node contains only 4 patients (P1, P3, P4, and P5) and we focus only on those patients in our table.
We start the same procedure by thinking of this node as a parent and we want to see if we should split it according to "exercises" or "age". In this node we have 3 patients who had heart attack and 1 who did not. Its gini impurity is

$$\text{Gini (parent)} = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = \boxed{0.375}$$

Now we need to decide whether "exercises" or "age" should be used for the split.

- **Step 1:** If we use "exercises" to split the data in this node, the patients who exercised will be placed in the left child node and the patients who did not exercise will be placed on the right. The left child node contains 2 patients and 1 of them had heart attack and 1 did not have the heart attack. The gini impurity for that node is

  $$\text{Gini}(\text{child}_1) = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$$

  The right child node contains 2 patients of which both had the heart attack. So in that node we have 2 patients who had the heart attack and 0 patients who did not have the heart attack. The gini impurity is

  $$\text{Gini}(\text{child}_1) = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0$$

  Therefore, the information gain for this split is

  $$\text{IG (exercises)} = 0.375 - \frac{2}{4} \cdot 0.5 - \frac{2}{4} \cdot 0 = \boxed{0.125} \tag{4}$$

- **Step 2:** Now we look at variable "age". We compute averages for each pair of adjacent values. For example, the average of 7 and 18 is 12.5. All the averages are 12.5, 26.5, and 36.5.

We consider each of the three conditions "age < 12.5", "age < 26.5", and "age < 36.5", to determine which condition has the highest information gain.

Let's start with the condition "age < 12.5". This condition splits the 4 patients in two nodes. The child node on the left contains patients of age less than 12.5 (there is 1 such patient; 0 had the heart attack and 1 did not have the heart attack) and the child node on the right contains patients of age over 12.5 (there are 3 such patients; 3 had the heart attack and 0 did not have the heart attack). We compute the gini impurity for each child node and the information gain for this splitting condition:

$$\text{Gini}(\text{child}_1) = 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2 = 0$$

$$\text{Gini}(\text{child}_2) = 1 - \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2 = 0$$

Note that both child nodes are pure which means that this will be the best split when we consider variable "age" and we do not need to look at the conditions "age < 26.5" and "age < 36.5". We also compute

$$\text{IG (age} < 12.5) = 0.375 - \frac{1}{4} \cdot 0 - \frac{3}{4} \cdot 0 = \boxed{0.375} \tag{5}$$

To decide whether we use "exercises" or "age < 12.5" as the splitting condition, we look at lines (4) and (5) and notice that IG is higher for the second condition.

Since all the nodes are pure, we are done. All we need to do is determine the label for each leaf. This is done by looking which class is present in the leaf.

References and Reading Material:

[1] *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, by Geron, Chapter 6

[2] MIT notes, Chapter 14 on Non-parametric methods

[3] To code a decision tree from scratch, see: https://betterdatascience.com/mml-decision-trees/

[4] Prof. Patrick Winston's lecture (MIT, Artificial Intelligence)
Youtube video on identification trees #11

[5] *Introduction to Statistical Learning*, James et al., Section 8.1

[6] *Elements of Statistical Learning*, Hastie et al., Section 9.2