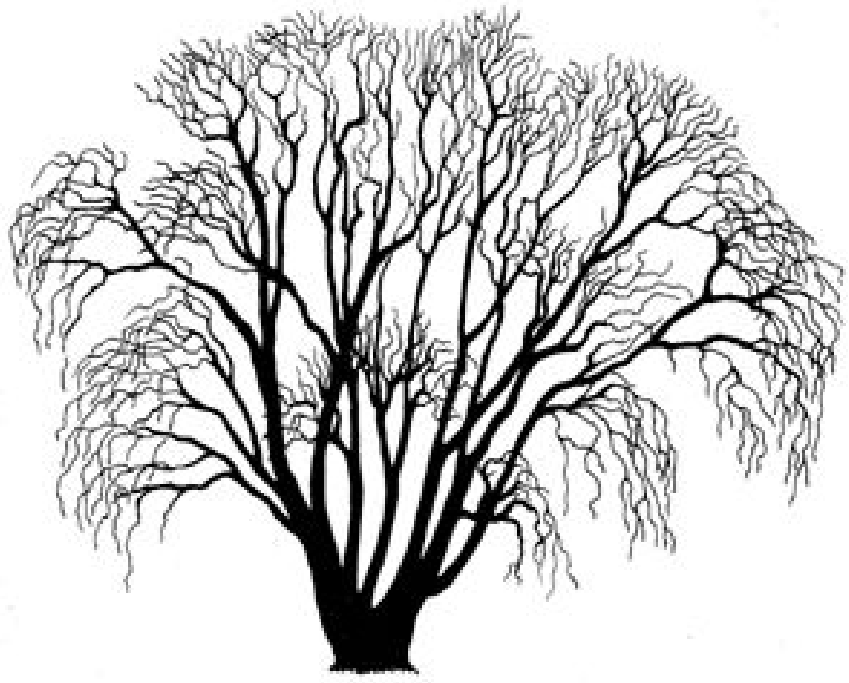
Tree-based Method From Scratch

Keyu Chen, Yiran Zheng, Yusheng Jiang

Catalog



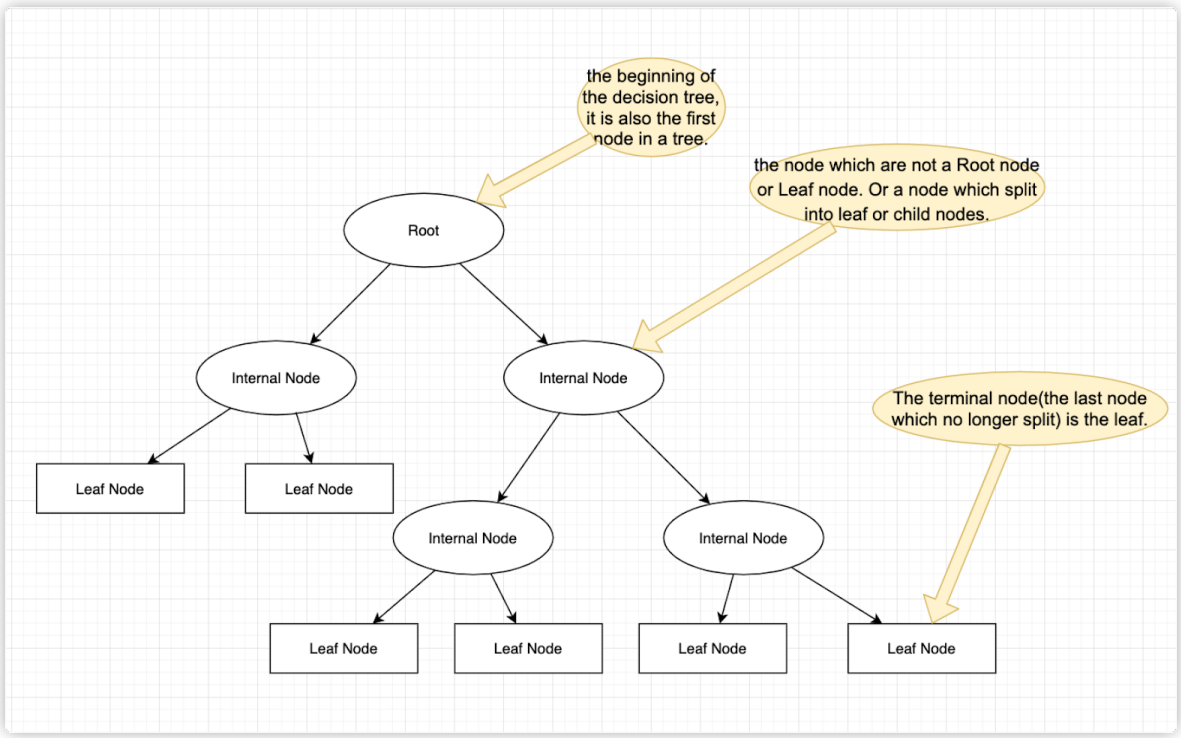
Introduction of Tree-based Methods

Tree-based methods are simple and useful for interpretation and mostly used in supervised learning methods. Tree-based algorithms empower predictive models with high accuracy and stability. They are adaptable at solving both classification and regression problems.

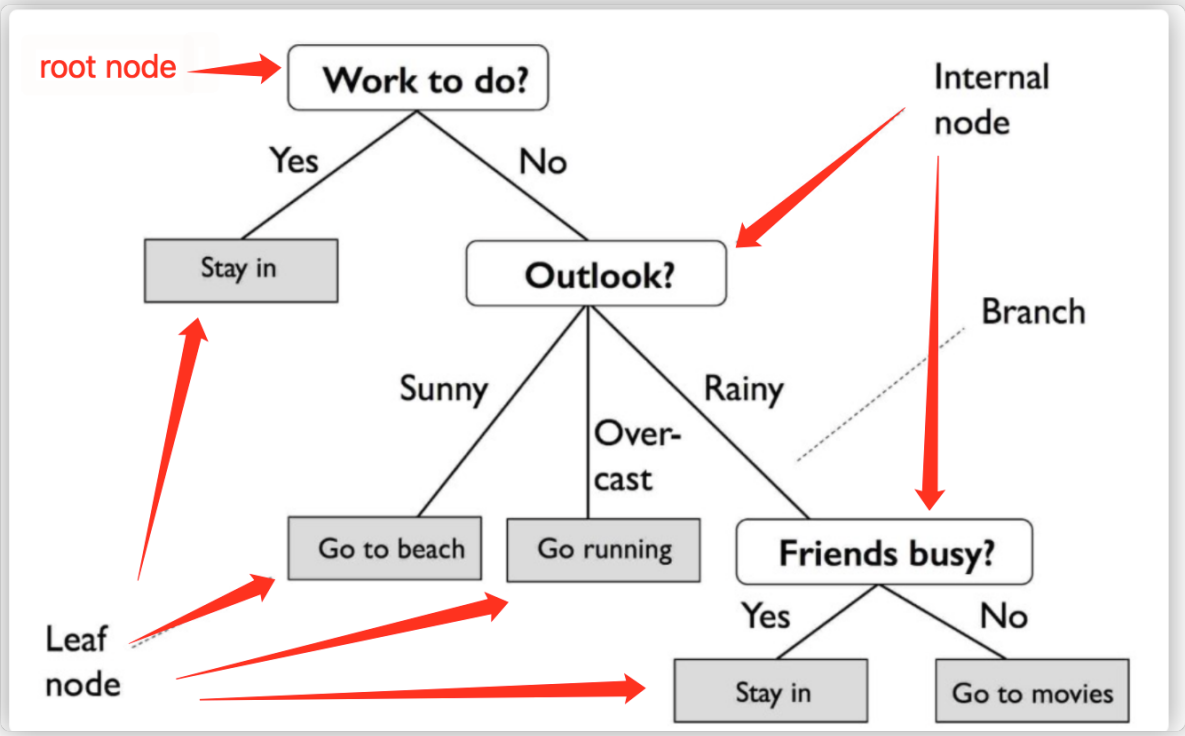
This tutorial is meant to help beginners learn tree based algorithms from scratch. After the successful completion of this tutorial, one is expected to become familiar with the key concept of the tree-based algorithms. In this Tutorial, you will learn the decision tree methods, both regression trees and classification trees, and Random forest trees which involves producing multiple trees which are then combined to yield a single consensus prediction.

Decision trees

Terminology
Root: the beginning of the decision tree (the first node in a tree). The root includes all sets of samples.
Internal node: the node which is not a Root node or a Leaf node or a node which split into leaf or child nodes. Internal node represents different kinds of conditions for each attribute
Leaf: The terminal node(the last node which no longer split) is the leaf. Leaf nodes represent different results of decisions



Review of Decision Trees



Our everyday life is a decision Tree, for example:

When we wake up, the first decision we are facing is the root node, in this case, "work to do?", the root node will split into two child nodes, if the answer is yes for "work to do", the left node in is 'stay in', since there is no further split, in this case 'stay in' is the leaf node. The right child node of the 'work to do' is "Outlook?". In this class it has 3 child nodes, "Go to beach", "Go running" and "Friends busy?". Both "go to Beach" and "Go running" have no further split, so both of them are leaf nodes. While "Friends busy" has further split, so it is an Internal node. "Stay in" and "Go to movies" are last split from Friends busy, so both are leaf nodes.

Key summary:
Root node - the first parent node in the tree.
Internal node - nodes that are beneath root and will further split and become parent nodes of others.

CART training algorithm

CART algorithm is used to train a decision tree. It first splits the training set into two subsets using a single feature k and threshold tk—for example, age>=6.5 or Sex=male, from the previous example. The algorithm searches for the pair (k, tk) that produces the purest subsets. The cost for classification algorithm that needs to minimize is defined by

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

where $m_{left/right}$ *means the number of instance in the left/right subset*

$G_{left/right}$ *measure the impurity in the left/right subset*

The CART algorithm splits the child node recursively until it reaches the max depth and then stops.

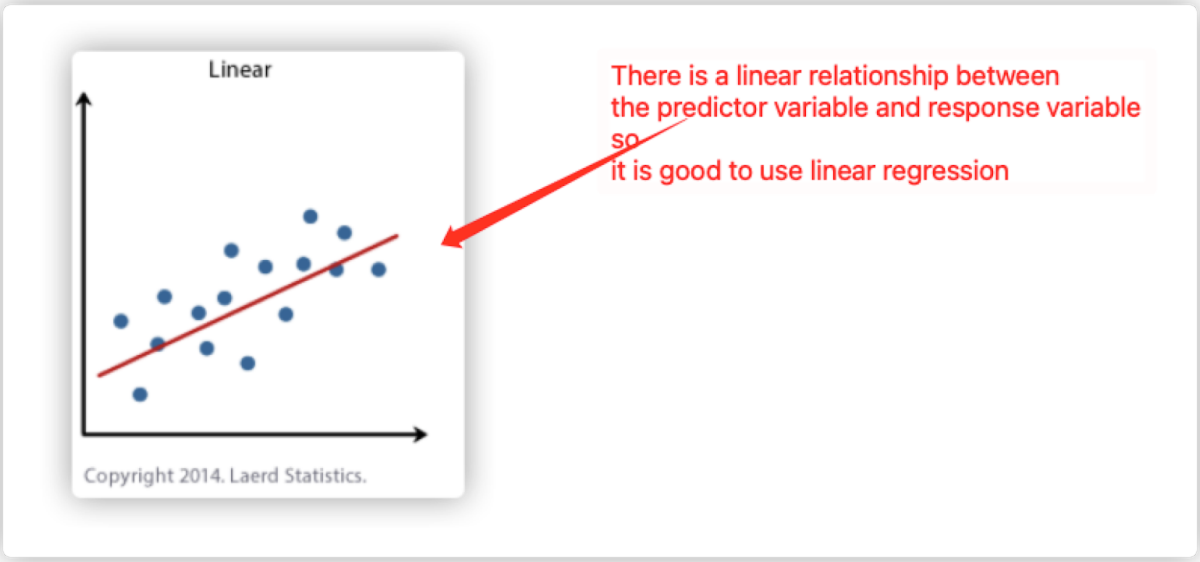
Differences between regression and classification

	Regression Tree	**Classification Tree**
Response variable	Can be used for both numerical or continuous response variables. For example, if we want to predict housing price, housing price in this case is our response variable, and it is a numeric variable.	When the response variable is Yes/No or other discrete categorical value, it could be more than two classifications.
Splitting method	Use Residual Sum of Squares(RSS) as criterion of making splits	Use Classification Error Rate as criterion of making binary splits
Fitting type	For prediction-type problems	For classification-type problems

Regression Tree

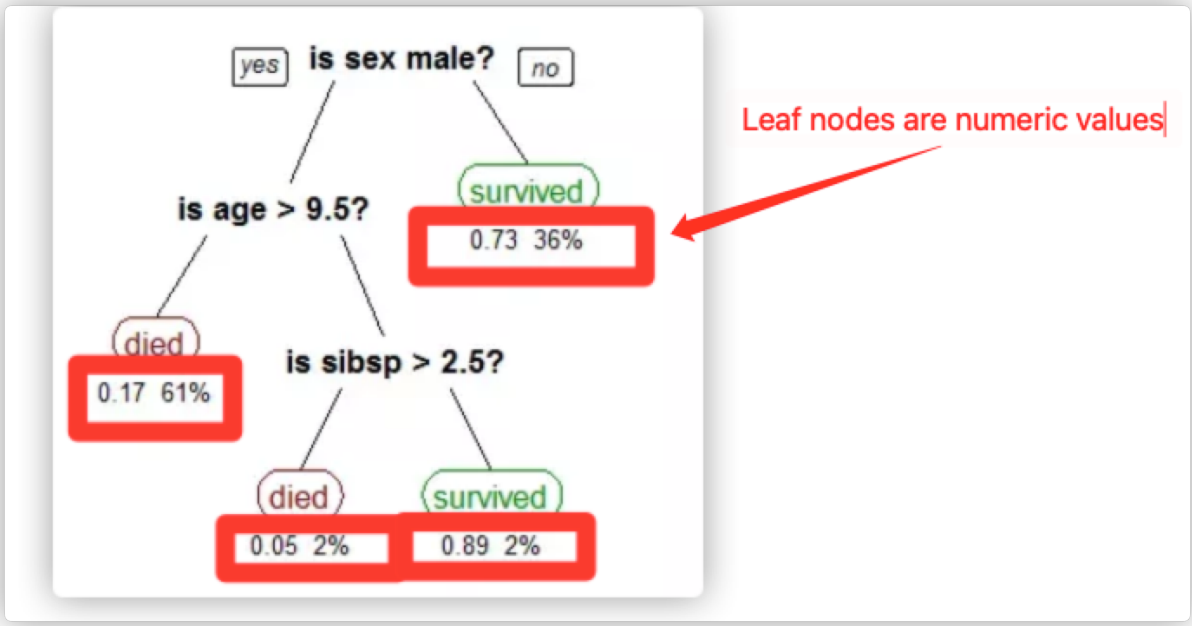
Why do we use Regression Tree?

When a simple linear model or others are not fitting well for the data, Regression Tree model can be a good try.



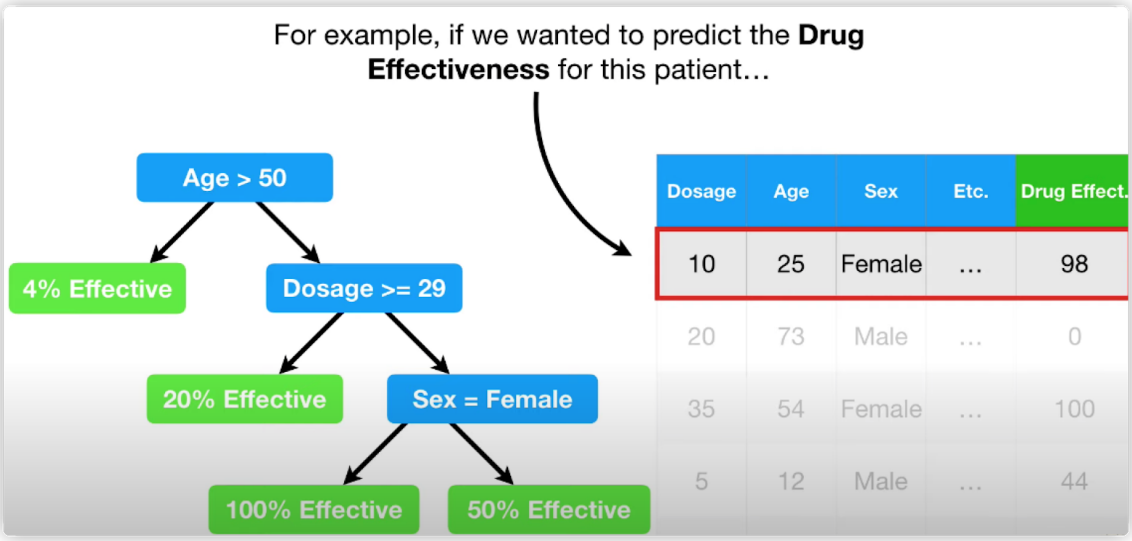
How Regression Trees Work?

- Regression Tree split from top to bottom, with multiple splits
- Each leaf represents a numeric value (the red rectangle).
- Example below



- First step:
  - find the thresholds to split the data into two. (Choose the one with the smallest sum of squared residuals as node, if there are multiple variables, the variable with smallest sum of squared residuals will be the root node)
- Second Step:
  - Set a minimum number of observations for a node, if the node has less than the minimum number of observations, the node will stop to split, and become a leaf.
  - Before the node has less than the minimum number of observations, it will keep using the optimum threshold to split.
  - Every time split the node, if the data meet the requirements, the leaf will return the average value as its prediction.

Regression Tree Example(Screenshot from STATQUEST):



Step by

step walkthrough a Regression Tree

Example dataset: response variable is column "Drug Effectiveness" from 0 - 100, with 3 predictor variables, Dosage, Age, Sex.

Prediction of the Drug Effectiveness example:

- We need to choose a root node first, which we determine how to divide the observations by trying different thresholds, In this case, If the age is older than 50, (50 is our threshold because it give us the smallest SSRs)
    - Yes, follow the left node
        - 4% effective
            - This is the average effectiveness of people who are older than 50
    - No, follow the right node, which is asking if the Dosage is larger or = 29, choose 29 as our threshold because it gives us the smallest SSR within Dosage.
        - If >= 29, 20% effective (average)
        - If < 29, ask if Sex = Female
            - If Sex = Female, 100% effective(average)
            - If Sex not equal female, 50% effective((average)

Common Question Regarding Regression Tree:

How do we find out the optimum threshold to split the data into two groups?
Find the number (the threshold) that gave us the smallest sum of squared residuals. In the Drug Effectiveness example case above, For age variable, 50 is used to split the two groups, because it gave us the smallest Residual Sum of Squares(RSS).

When will the tree stop splitting?
Depending on the minimum observation we set for each leaf. Generally we set it as 20, but if you have a small dataset, you need to adjust and try from 20 to some smaller values, such as 5.

How to find out the root?
Calculate each variable's sum of squared residuals, then select the smallest one as the root.

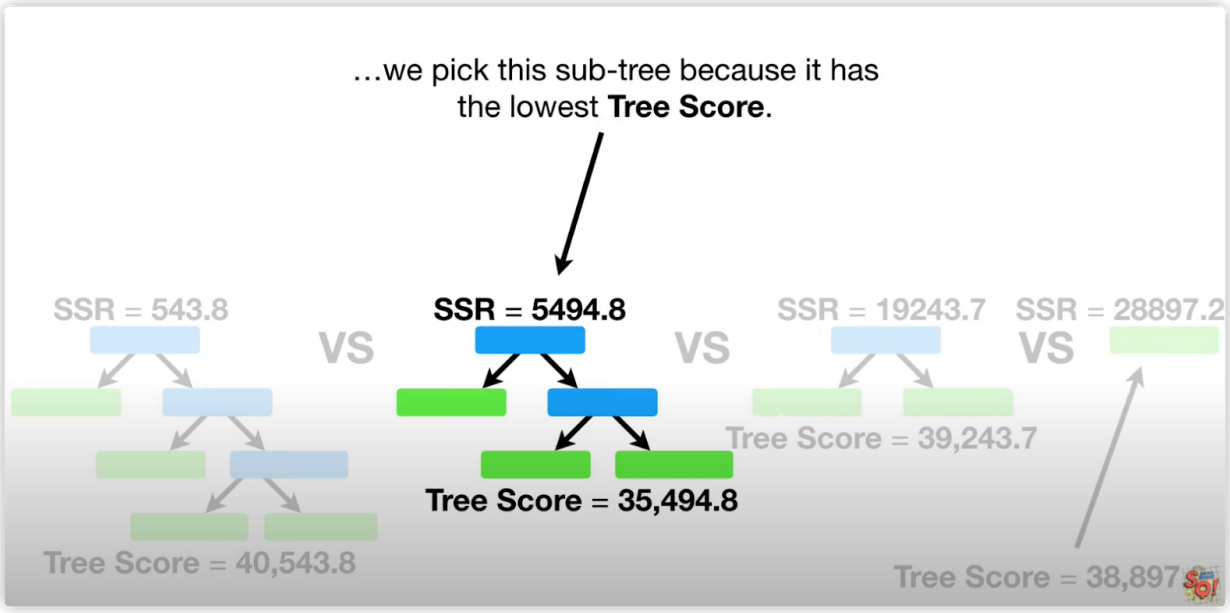
What indicator can we use for the evaluation of the Tree?

- Mean absolute error(MAE)
- Mean Squared Error
- Root mean squared error (RMSE)
- R Squared score (R^2)
- Sum of Squared Residuals
- linear correlation or rank correlation

Tree Pruning(picture from STATQUEST)

- One of the way is to use Cost complexity Pruning (weakest link pruning)
    - Calculate a Tree Score

- Tree score = SSR + αT
- α, tuning parameter
- T, penality



Pruning by Information Gain

One pruning method is to prune out the portions of the tree that introduce least information gaining. The information gaining is defined by following:

$$IG(S,\ a)\ =\ H(S) - H(S\mid a)$$

Where

- IG(S,a) is the information for dataset S for the random variable 'a'
- H(S) is the entropy for the dataset before any changing
- H(S|a) is the conditional entropy for the dataset S given the variable 'a'

This method is based on the information that is already computed when the tree is built from the training data. The process of Information Gain based pruning requires us to identify the "twigs" which are nodes whose children are all leaves. We prune a "twig" by removing all of the leaves that are the children of the twig and making the "twig" a leaf.

The algorithm for information gain based pruning show as follow step by step:

1. Catalog all twigs in the tree
2. Count the total number of leaves in the tree.
3. While the number of leaves in the tree exceeds the desired number:
    1. Find the twig with the least Information Gain
    2. Remove all child nodes of the twig.
    3. Relabel twig as a leaf.
    4. Update the leaf count.

Why pruning the tree?

- To solve the issue of overfitting by replacing the node which doesn't improve the quality of expected prediction.

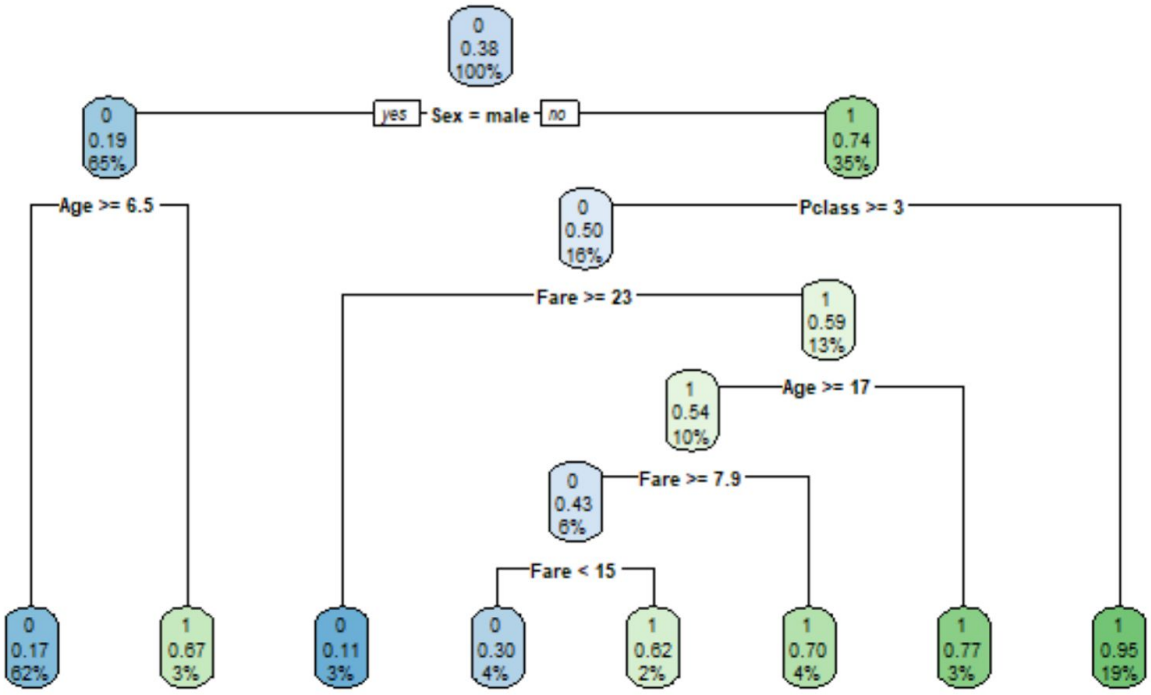
Classification tree

Classification tree is very similar to regression tree, but the classification tree is to predict a qualitative response rather than a quantitative response variable.We predict that
each observation belongs to the most commonly occurring class of training
observations in the region to which it belongs.Just like a regression tree, we use recursive binary split to grow the tree. However, we use the classification error rate as a criterion to make the splitting. The classification error rate is the fraction of the training observations in that region that do not belong to the most common class.

Here is an example of classification tree(Data source: https://www.kaggle.com/c/titanic/data):

Description: the data set provides information on the fate of passengers on the Titanic, summarized according to economic status (class), sex, age and survival.The specific data description for all variables in the dataset can be found in the above link.

The result of classification tree only consider the survival as response variable and limited variables in the dataset, sex, age, class, and fare, as predictor variables.



As you can see from the above figure, based on the training data, our model finds that if the passenger is not a male and the ticket class was third class, she has a half and half possibility of survival. If the

passenger is male and his age was approximately smaller than 6.5 year old, he has a 67% chance of survival. If the passenger is female and got a first or second class ticket, the passenger has a 95% chance of survival when the Titanic is sinking.

- Notice: 0 means death in the sinking of titanic, and 1 means survival.

Two measurement decided how to split the trees

If you ever wondered how decision tree nodes are split, it is by using impurity. Impurity is a measure of the homogeneity of the labels on a node. Sometimes, the classification error is not sensitive to grow the decision trees, there are two other measures that are preferable, and they are Gini Index and Entropy.

Gini Index

The gini index is defined by

$$Gini\ index = 1 - \sum_{i=1}^{n} p_i^2$$

Where p_i denotes the probability of an element being classified for a distinct class.

The gini index is measured by the total variance across the classes. The number could get really small if a node contains predominantly observations from a single class.In another word, Gini Index measures the divergences between the probability distributions of the target attribute's values and splits a node such that it gives the least amount of impurity.

Entropy

Entropy is a measure of randomness given by

$$Entropy = \ -\sum_{i=1}^{n} p_i log p_i$$

Where p_i denotes the probability of an element being classified for a distinct class.

One can show that the entropy will take on a value near zero if the p_i are all near zero or near one. Therefore, like the Gini index, the entropy will take on a small value if the ith node is pure.In fact, the gini index and entropy are numerically similar to each other.

Calculating entropy for a split

1. Calculate the entropy of the parent node.
2. Calculate the entropy of each individual node of the split.
3. Calculate the weighted average of all sub-nodes available on the split.

When building classification decision trees, either of the measurements will be taken to evaluate the quality of the split due to the fact that the two measurements are more sensitive to node purity than the classification error rate.

Regularization of hyperparameters to prevent overfitting/underfitting

Decision trees make few assumptions on the data unlike the linear regression model that assume that the data is linear. The tree as a type of non-parametric model would adapt itself to the training data unconstrained,and further cause overfitting. In order to prevent the overfitting of the training data, we will regularize the hyperparameters to restrict the decision tree's freedom from training.

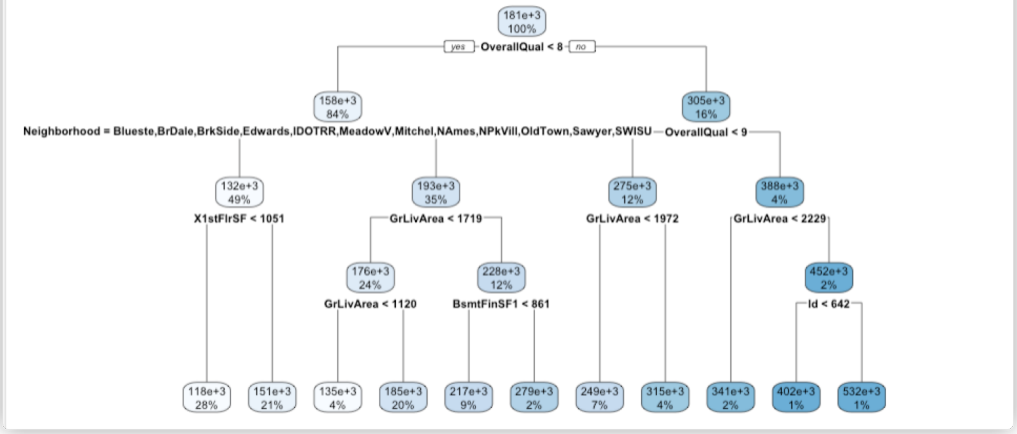
1. Min_samples_split
   Minimum number of samples in a node must have before it can be split. This can help reduce overfitting.
2. Min_samples_leaf
   Minimum number of samples a leaf node must have. (most of the time, the minimum number of observations to allow for a split is 20. But you should adjust the number based on the number of observations you have in your dataset, For example if you only got a few observations you may want to set the minimum observation to be 10 or even smaller.)
3. Max_depth
   Maximum depth of a tree. Usually higher depth allows models to learn the relations very specific to a certain data point.
4. Max_leaf_nodes
   Maximum number of leaf nodes
5. Max_feature:
   Maximum number of features that the tree takes in consideration for splitting at each node.

Advantage of the Trees

- Easy to understand and interpret, perfect for visual representation. Decision trees more closely mirror the human decision-making process.
- Can work with both numerical and categorical features.
- Requires little data preprocessing: no need for one-hot encoding, dummy variables, and so on.
- Do not need assumptions about the shape of the data.
- Feature selection happens automatically: unimportant features will not influence the result. The presence of features that depend on each other (multicollinearity) also doesn't affect the quality.
- Real world example, easy to fit and very fast (it took 3 second to run on the Dataset, House Prices: Advanced Regression Techniques, https://www.kaggle.com/c/house-prices-advanced-regression-techniques)

```{r}
data <- read.csv("train.csv")
tree = rpart(SalePrice~., data=data)   # SalePrice is response variables vs all other predictor variables
#-- Plot Tree
library(rpart.plot)
rpart.plot(tree)

```



Disadvantages of the Trees

- May have issues with highly collinear predictors.
- Trees are very unrobust, meaning that a small change in data may cause a large change in the final estimated trees.
- The problem with finding the globally optimal decision tree is NP-complete.
- It does not always provide a clear informative relationship between the predictors and the response variables.
- Biased toward predictors with more variance or levels.
- Doesn't work very well with low sample sizes(poor prediction accuracy for responses).
- Overfitting of the tree can happen very easily, especially under the circumstances that only few observations in last nodes, this can be fixed by setting a larger minimum number of observations to split, so the model won't overfit and have some variances.
- Trees are inflexible, if we add a new labeled data, we need to retrain the whole tree from scratch on the whole dataset. Therefore, trees are a poor choice when the data need dynamic changing over time.
- It tends to overfit. This usually can be mitigated in one of three ways:
    - Limiting tree depth
    - Setting the minimal number of objects in leaves
    - Tree pruning by deleting unimportant splits moving from the leaves to the root
- For classification, if some classes dominate, it can create biased trees. It is therefore recommended to balance the dataset prior to fitting.

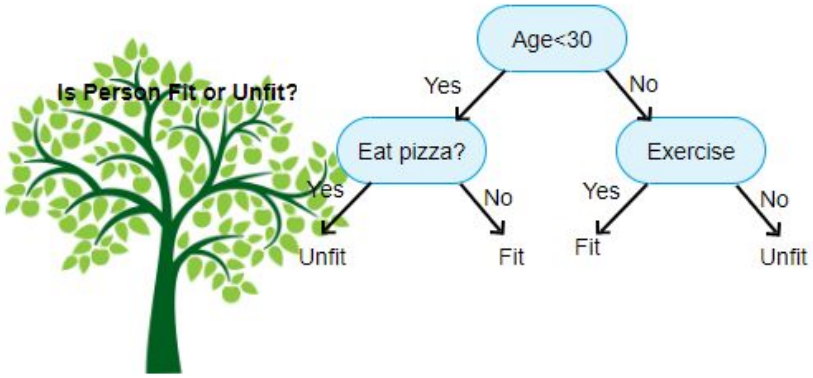
- Decision trees perform greedy search of best splits at each node(Greedy algorithms will not always give the optimal solution, in fact, greedy algorithms tend to always choose the solution best at the moment). This is particularly true for CART based implementation which tests all possible splits.
- Does not work well for all the distribution

Random Forest：A very intuitive introduction

Introduction of Random Forest

When people are talking about Random Forest, in fact, it is a kind of machine learning algorithm or model, which has already been invented for more than 20 years.

Here “Random” means when we take samples from the total dataset, we extract them by using a random way. Besides, the reason why we call it “forest” lies in that it is based on the “tree” model, and we set multiple times of random samples, so that we will create a lot of tree models, which just look like a big forest during a random forest process.



Process of decision tree

When doing the prediction, we use a certain attribute value as a judgment value at internal nodes. Then according to the result of the judgment, we decide which way the current sample should go into, and then we have another judgment until we reach the leaf nodes.

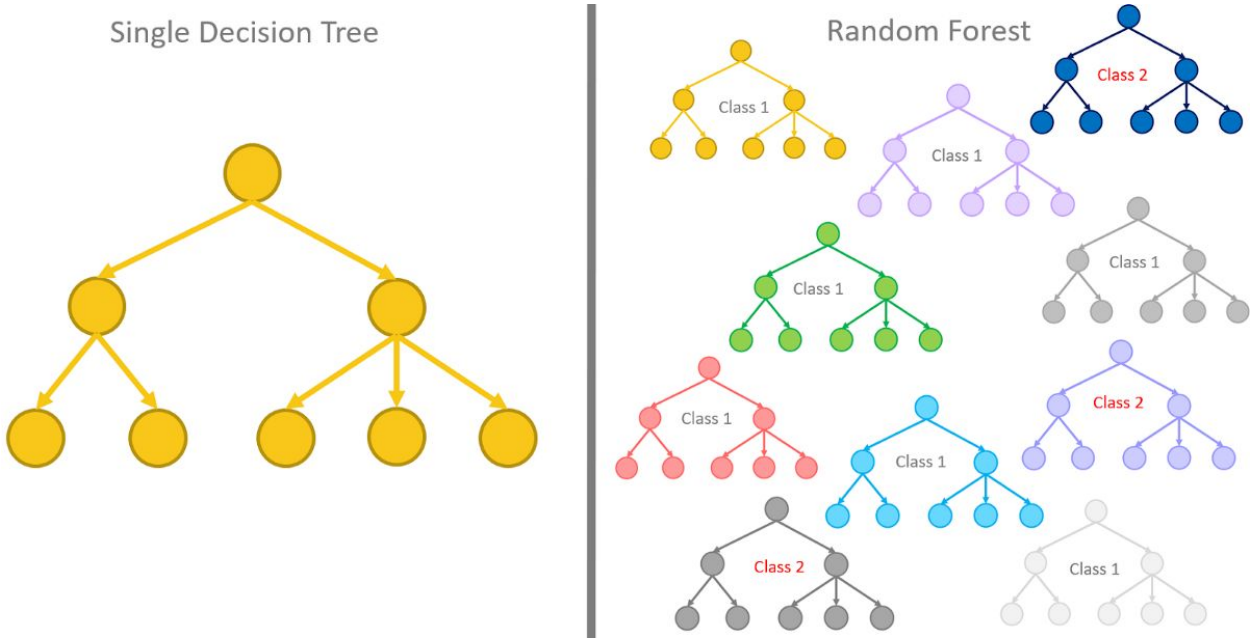
Process of random forest

After introducing the concept and process of the decision tree, let’s move on to talking about Random Forest.

Random forest is composed of a huge number of decision trees, and there is no correlation between each decision tree.

When doing the classification, new input samples get entered, each decision tree in the forest will conduct judging and classifying work separately. Then, each decision tree will get its own classification

result. If the count of a certain classification result from all decision trees is to be the biggest one, then the random forest will take this result as the final result.



Four general steps for Random forest

1. Randomly picking up different sets of samples from all(with replacement)
2. Randomly selecting attributes for nodes splitting
3. Repeating Step 2 for different attributes until reaching leaf nodes
4. Repeating all above steps, building a large number of trees as forest

Addition knowledge:

If you want deep through the details of application of Random Forest, here is the detailed version of 4 steps.

- If we say totally, we have N samples, then we randomly select N samples with replacement(one sample is randomly picked out each time, and then return it back to continue selecting). The N selected samples are used to train a single decision tree, which starts from the root node of it.

- Then suppose each sample has M attributes(columns). When every node of a decision tree needs to be split, then m attributes are randomly selected from total M attributes as long as it satisfies the condition m <<(far less than) M. Next, we use a certain kind of strategy (like information gain) to select 1 attribute from m ones as the best splitting attribute.

- In the process of decision tree growth, each node should be split according to step 2 and all selected attributes should be used until it can't split again. Note that there is no pruning in the whole process of decision tree formation.

- Follow steps 1 to 3 to build a large number of decision trees, which finally are composed of a random forest.

Important hyperparameters

In the random forest package, we have two hyperparameters that can be used for tuning manually. They are mtry and ntree.

ntree:
Number of trees to grow. This parameter should not be set to a number that is too small, because we want to ensure that every input row gets predicted at least a few times.
mtry:
Number of variables randomly sampled as candidates at each split. Note that the default values in the function if not specify are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3)

Pros and cons of Random Forest

Pros:

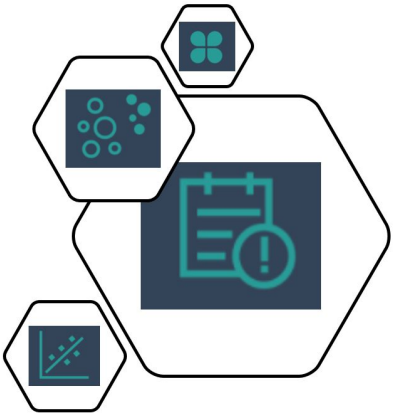
- can produce data with high dimensions (many features), no need for dimension reduction and no need for feature selection
- can judge the significance of different features
- can measure the interaction between different features
- training speed is relatively fast, and is easy to work parallelly
- implementation is relatively simple
- can balance the errors for unbalanced datasets
- the accuracy can be maintained and secured if a large portion of the features are missing

Cons:

- can be overfitting in classification problems with large noise or regression ones
- will lose credibility when the attributes have too many different values

Application Directions:

- **Classification** on discrete values
- **Regression** on continuous values
- **Clustering** of unsupervised learning
- **Outlier** detection



Reference

Titanic: Machine Learning from Disaster. (n.d.). Retrieved December 10, 2020, from https://www.kaggle.com/c/titanic/data

James, G. (n.d.). Retrieved December 10, 2020, from http://faculty.marshall.usc.edu/gareth-james/ISL/

Tyagi, N. (2020, September 30). Understanding the Gini Index and Information Gain in Decision Trees. Retrieved December 10, 2020, from https://medium.com/analytics-steps/understanding-the-gini-index-and-information-gain-in-decisihttps://virginia.zoom.us/j/9371315275?pwd=U2Zna2VZckNWelVWbjVEZTQ2NTZaUT09on-trees-ab4720518ba8

Sosnovshchenko, A. (n.d.). Machine Learning with Swift. Retrieved December 10, 2020, from https://learning.oreilly.com/library/view/machine-learning-with/9781787121515/697c4c5f-1109-4058-8938-d01482389ce3.xhtml

Mwangi., B. (2020, April 10). Understanding Tree-Based Machine Learning Methods. Retrieved December 10, 2020, from https://heartbeat.fritz.ai/understanding-tree-based-machine-learning-methods-5c2206a9d5f9

Starmer, J. (Director). (n.d.). Regression Trees, Clearly Explained!!! [Video file]. Retrieved from https://youtu.be/g9c66TUylZ4

Li, L. (2019, May 16). Classification and Regression Analysis with Decision Trees. Retrieved December 10, 2020, from https://towardsdatascience.com/https-medium-com-lorrli-classification-and-regression-analysis-with-decision-trees-c43cdbc58054

House Prices: Advanced Regression Techniques. (n.d.). Retrieved December 10, 2020, from https://www.kaggle.com/c/house-prices-advanced-regression-techniques

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier¶. (n.d.). Retrieved December 10, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Tree picture [Digital image]. (n.d.). Retrieved from https://images.app.goo.gl/uWsTzfqs7dhT161x9

Gupta, S. (2020, June 23). Pros and cons of various Classification ML algorithms. Retrieved December 10, 2020, from https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6

Random forest picture. (n.d.). Retrieved December 10, 2020, from https://images.app.goo.gl/NMDzbobAu6ngo5tH9

Donges, N. (n.d.). A complete guide to the random forest algorithm. Retrieved December 10, 2020, from https://builtin.com/data-science/random-forest-algorithm

RandomForest. (n.d.). Retrieved December 10, 2020, from https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest

Decision Trees - Carnegie Mellon University. (n.d.). Retrieved December 10, 2020, from https://www.cs.cmu.edu/~bhiksha/courses/10-601/decisiontrees/