NYC DATA SCIENCE ACADEMY

Python Machine Learning Class:

Tree Based Models

Decision Trees and Random Forests

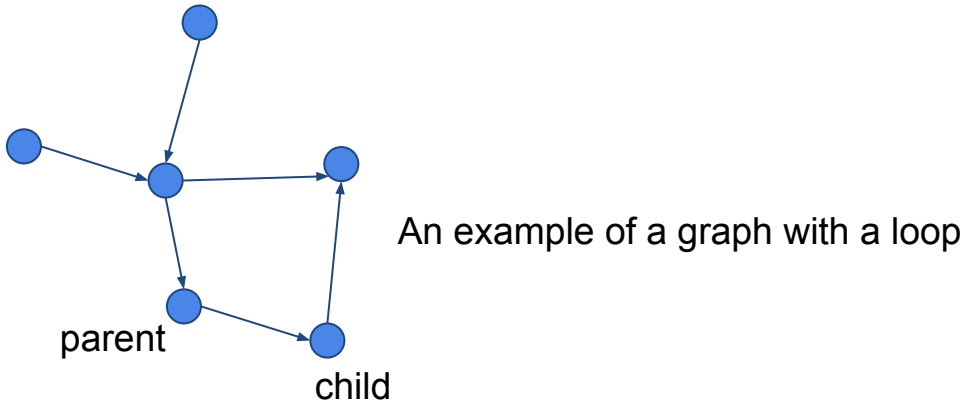
NYC Data Science Academy

Outline

- **Tree-Based Methods**
  - **The Intuition of Tree Models**
  - **Decision Trees, A Geometric Perspective**
  - **Bagging and Random Forests**

What is a graph?

- Recall that a (directed) graph is a collection of nodes along with some one dimensional oriented edges among these nodes

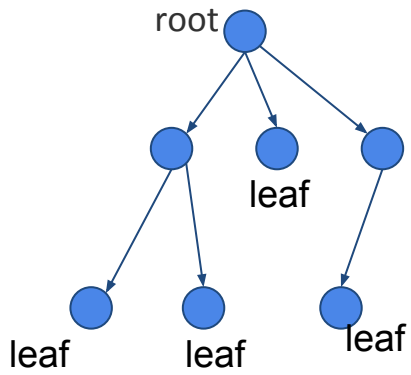


An example of a graph with a loop

- Given a node, it and its child/successor node is connected by an arrow pointing toward the child
- The original node is called a parent node
- The concept of graphs is widely used in machine learning and deep learning

What is a tree?

- A tree is a special type of directed graph which satisfies:
  - The graph is connected
  - Each node has at most one parent node
  - The graph has no loops at all



An example of a tree

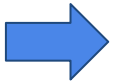
- The unique node which has no parent is often called the root node
- A node in a tree may have no child, 1 child or many children nodes
- The nodes which have no children are called leaf (terminal) nodes

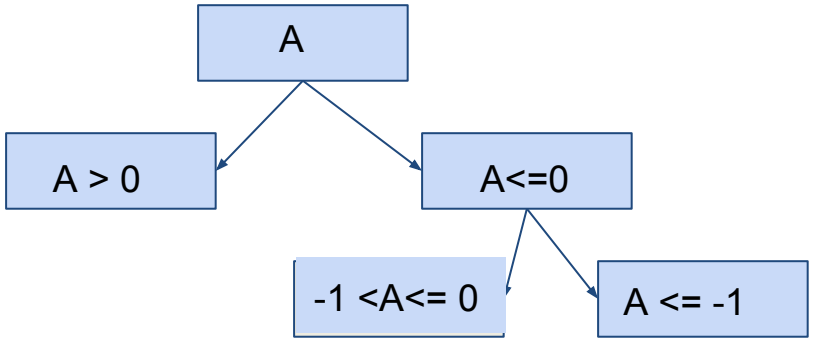
Why Is the Concept of Tree Relevant?

- The concept of tree is used everywhere in our life
  - The maternal ancestry of a person forms a tree
  - The directory structure (without considering links) in a file system forms a tree
  - In python, the singly inherited classes from 'object' (the root class) forms a tree
  - In biology the taxonomic classification of life from domain, kingdom, …., up to species form a tree
  - In python, the compound 'if, else, elif' statements can be coded into a tree

For example,

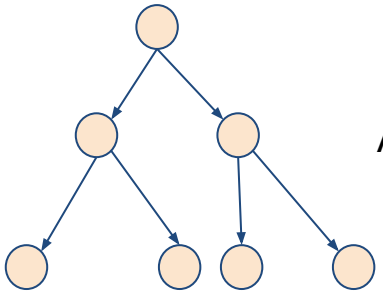
```
If A > 0:
        Some statement
elif A > -1:
        Some other statement
else:
        Another statement
```





Binary Tree and Decision Tree

- In summary, trees help us to lay out the given data structure in a coherent way, which encode classification or a nested logical structure
- We are particularly interested in binary trees, where each node has either zero or exactly two child nodes
- Such binary trees can be used to encode the iterative logical decision processes, or nested binary classification problems, which will be the focal point of our tree based supervised machine learning model



An example of binary tree

- Binary trees are also used in the hierarchical clustering model

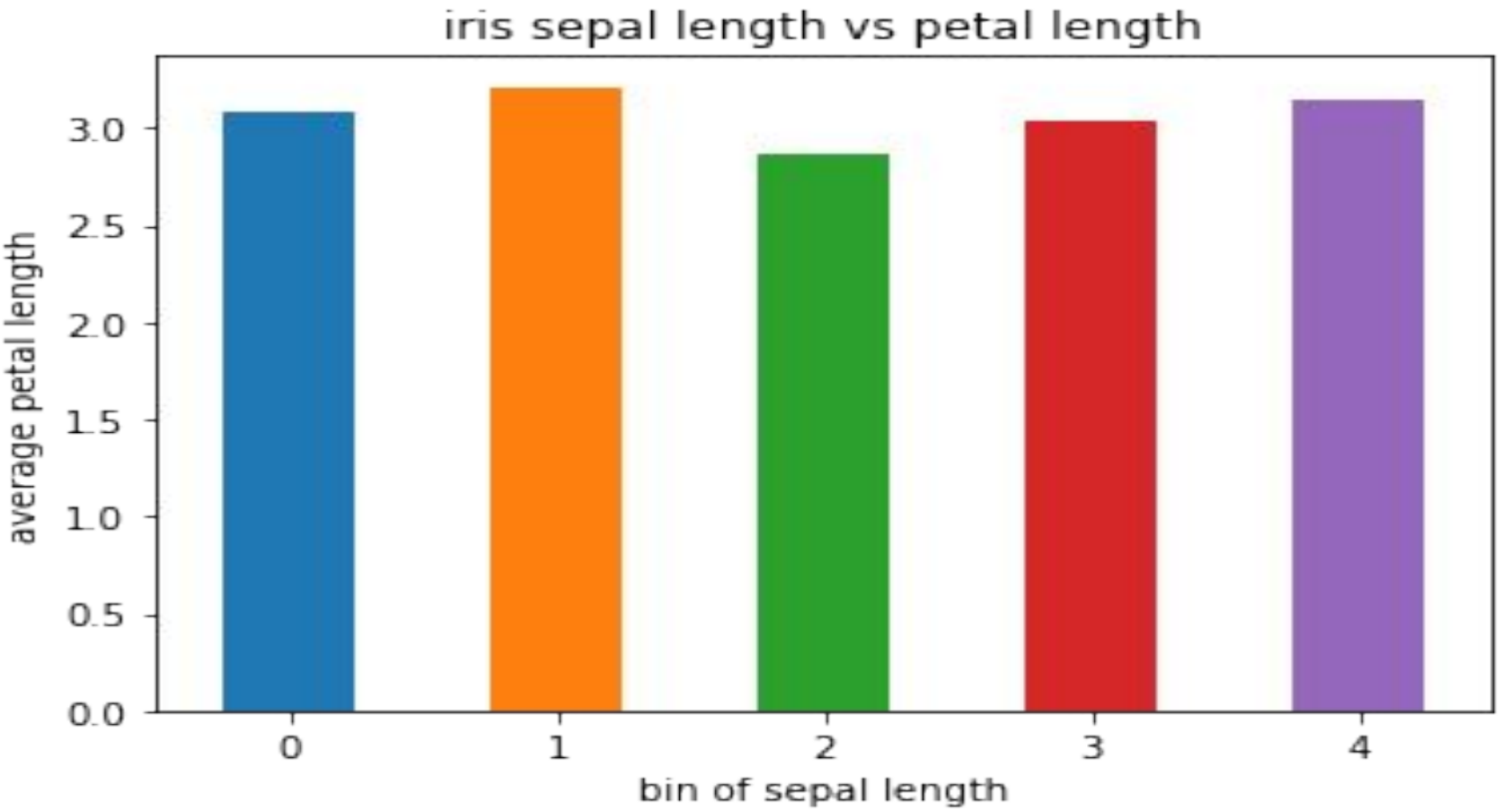
Finding Patterns Between Features and Targets

- In data analysis or machine learning, we often relabel the values of a discrete (i.e. qualitative) feature by integers, effectively treating them as lattice points on the real line
- In this way we can always view the total feature space as an n dimensional Euclidean space
- As our observed feature data points form a scatter plot within the Euclidean space, our goal is to find patterns between the locations of the features and the corresponding target values, by inductive method

From Bar Charts to the Idea of Decision Trees

- In fact this is not the first time we encounter this method
- Recall that when we need to figure out the relationship between two features, we often adapt a bar plot/bar chart to visualize their potential relationship
- For example when we study the relationship between sepal length and petal length of the iris data set, we can:
  - Bin the sepal lengths into 5 different equal-distance bins
  - Replace the quantitative sepal lengths by the bin numbers
  - Groupby the bin numbers and compute the average petal length in each bin
  - Display the barplot of the sepal length bin number vs the petal length mean

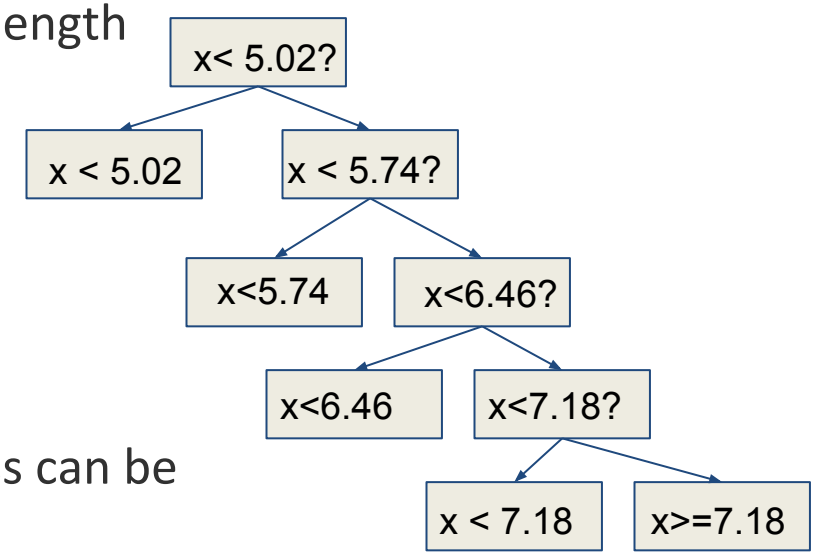
The Bar Plot of Sepal Length vs Petal Length



The bins (4.296, 5.02) < (5.02, 5.74) < (5.74, 6.46) < (6.46, 7.18) < (7.18, 7.9)

The Decision Tree Behind the Analysis

- In the above example of binning the iris sepal lengths, the labelling by the bin numbers can be thought as a sequence of decision-making
- Let x denote the value of sepal length
  - If $x < 5.02 \rightarrow$ bin 0
  - elif $x < 5.74 \rightarrow$ bin 1
  - elif $x < 6.46 \rightarrow$ bin 2
  - elif $x < 7.18 \rightarrow$ bin 3
  - else $\rightarrow$ bin 4
- Graphically, the decision process can be coded by the graph on the right
- Such a graph is called a **decision tree**



Tree-Based Machine Learning Methods

- Tree based methods partition the feature space into a set of rectangular regions parallel to the coordinate axes and then fit a simple model (usually the mean in that region) in each subregion.
- Methods we'll talk about include
  - Decision Trees
  - Bagging and Random Forests

How to Handle Multiple Features?

- It is apparent the above computation can be carried out for any bivariate feature analysis, not specific to iris data set
- The key question is: Does the above bivariate analysis contain special insights that we can carry over to multiple features (high dimensional scenario?)

- The key observation to expand the above bivariate analysis is to allow testing the feature inequalities among many different features, instead of the single one (sepal length in our case)

The Intuition

- Such an idea can be used for both nonlinear regressions and classifications
- In the regression task, the logical decision rule encoded by the decision tree allows us to categorize the data points by the feature inequality constraints they satisfy. Then estimate the predicted value of any such point by the average target value of its peers (i.e. all the other samples satisfying exactly the same constraints)
- For classifications, we may classify the given sample point into the dominant class label among all its peers (again, this refers to all the other samples satisfying the same constraints imposed by the decision tree)
- The task of data science is transforming our intuition into quantitative rules that a machine can follow

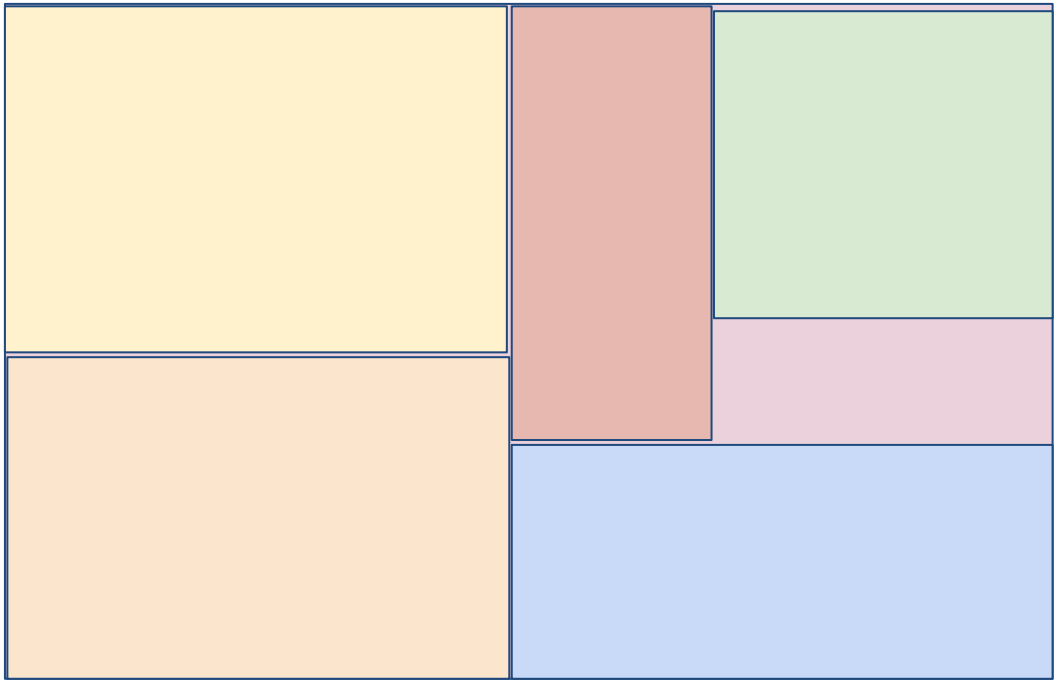
Outline

- Tree-Based Methods
  - The Intuition Behind Tree Models
  - **Decision Trees, A Geometric Perspective**
  - Bagging and Random Forests

Decision Trees

- **Decision trees** partition the feature space into a set of simple rectangular regions.
- **Decision trees** can be used for both regression and classification problems.
- We'll first consider an example of regression problem and then focus on classification.

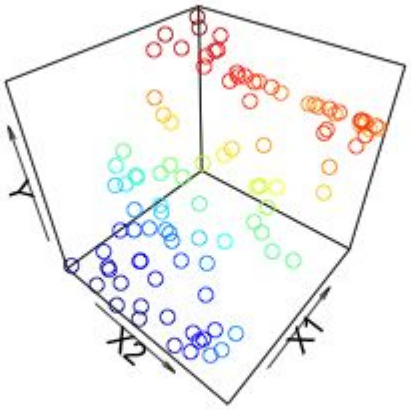
An Example Partition of the Rectangle (by Rectangles)



- A set of feature inequality constraints coded as a decision tree gives rise to a partition of the feature space and vice versa

Decision Trees - Regression

- Let's consider a regression problem with predictors X_1 and X_2 and continuous response Y.



- To fit a decision tree, we divide the feature space into non-overlapping regions and in each region R_j we model the response by its sample mean $\hat{y}_{Rj}$.
- The goal is to find rectangular regions (aka partition) that minimize the aggregated RSS:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Decision Tree - Regression

- Recall that **RSS** has been used in linear regression to gauge the performance of the model
- The task of the machine learning model is to provide an effective algorithm to improve the performance, i.e. reducing the value of the **RSS** loss function
- The same **RSS** works for nonlinear regression task as well
- To simplify the process, we consider a recursive binary splitting as our partition

Decision Trees - Regression

- Starting with the Null model (ANOVA) with the trivial partition, all the sample points are inside some large rectangular box
- We first select the feature X_j and the specific cut-point that splits the dataset to the greatest reduction in ***RSS***
- This choice of (feature, cut-point) pair induces a coarse partition of the feature space along with a slightly reduced **RSS**

Induction Process

- Each time we look at the rectangular boxes which partitions the original domain
- We notice that an individual rectangular box is like our original domain, the only difference is that it is a smaller subset of the original big rectangular domain
- Thus the original wisdom still applies and we can re-select another feature and cut-point pair to further reduce **RSS**
- We apply the above process recursively
- We observe that it cannot continue indefinitely (why not?)
- In reality, a stopping criterion is often used to terminate the process. By the stopping criterion, we mean that the process terminates if **RSS** reduction is not significant

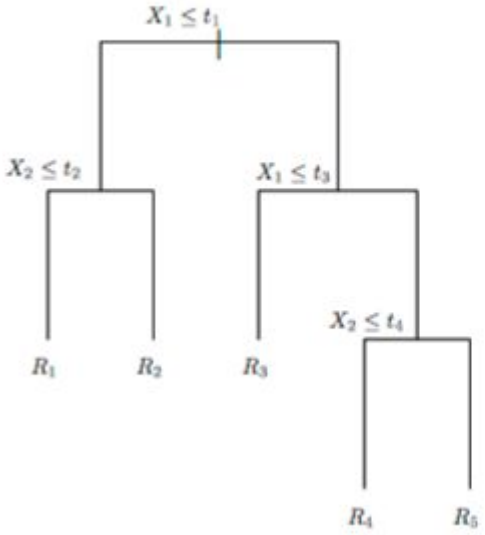
After the termination

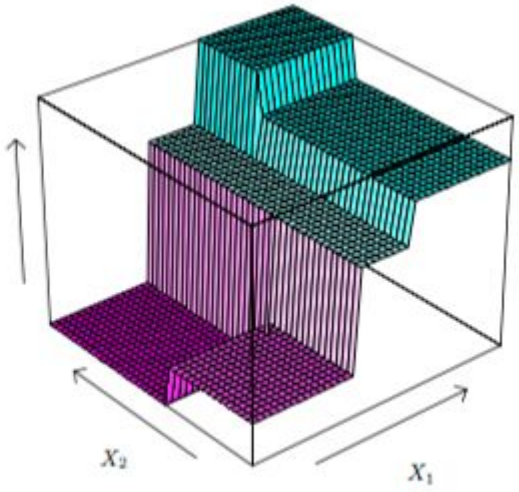
- Finally the ending partition (a collection of refined rectangular boxes) covers the original feature domain such that the aggregated **RSS** is small enough
- The process of box-splitting can be coded as a decision tree and vice versa
- The predicted value of the model in each refined rectangular box is the mean response of all the samples within the box

- A naive implementation of decision tree is computationally very expensive
- Often some greedy search heuristic will be implemented

Decision Trees - Regression

- A five-region splitting is achieved.
  1. split the overall dataset at $X_1 = t_1$;
  2. split the region $X_1 < t_1$ at $X_2 = t_2$ and the region $X_1 > t_1$ at $X_1 = t_3$;
  3. split the region $X_1 > t_3$ at $X_2 = t_4$.





The Weak Learner

- A regression tree is known as a **weak learner**, which performs poorly for unseen test datasets
- For a train dataset which the tree model is built on, the recursive refinement of the partitions reduces the number of training samples in each box. If we grow the tree to the extent that there is unique sample in the box, **RSS/box** would drop to zero. This would make the aggregated **RSS** artificially small for the training set
- But an out-of-sample prediction on unseen test sample can produce very high **RSS**, indicating a poor model fit

Remedy the Weak Learner

- How do we mitigate the overfitting issue on a regression tree?
  - Restrict the growth of the tree, avoiding large trees with few samples per leaf
  - Adapt **tree pruning**
  - Use the collective wisdom of many weak learners (in this context, regression trees) to form a strong learner, known as **ensembling** technique
- The last approach deserves a lot of attention, as it introduces a new family of sophisticated high performance machine learning model with high model complexity

Penalize the Tree Complexity

- Tree complexity refers to several parameters
  - Tree depth (the height of the tree)
  - The number of samples associated with a node (with the corresponding box)
  - …
- Penalizing refers to either introducing ad-hoc early stopping criterion or adding a tunable multiple of the depth of the tree to the loss function
- In either case, it tends to grow a smaller tree, reducing the potential of overfitting
- **But**, it can also lead to an underfit model

Pruning

- Pruning complements early stopping by growing an overly large tree which overfits and then it trims back the tree to optimize the cross-validation result (there are several variations of pruning)
- Why do we grow an overfitting tree at the first place?
- If the tree growing process stops too early, it has a great potential to stop prematurely, way before the optimal is reached
- Tree-pruning makes sure the tree grown starts to be too large and trim it down based on test performance

- Pruning improves overfitting of a decision tree, but it does not make it a strong learner. In particular sklearn **decision tree** does not implement tree pruning

Decision Trees - Classification

- Similar to regression, we use the same procedure to fit a **classification tree**.
- For regression we used the ***RSS*** to determine the splitting, what should we use for classification? Two commonly used metrics are:
  - **Gini impurity**: the probability a randomly chosen item from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset.

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i)$$

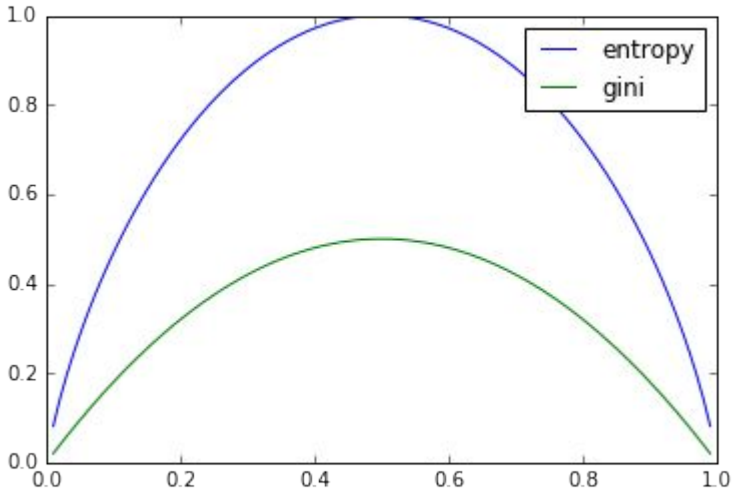
  - **Information Entropy** also works in a similar way:

$$I_E(f) = -\sum_{i=1}^{m} f_i \log_2(f_i)$$

where f_i is the fraction of items labeled with i in the set , known as empirical probabilities, and $\sum f_i = 1$.

Decision Trees - Gini impurity and Information Gain

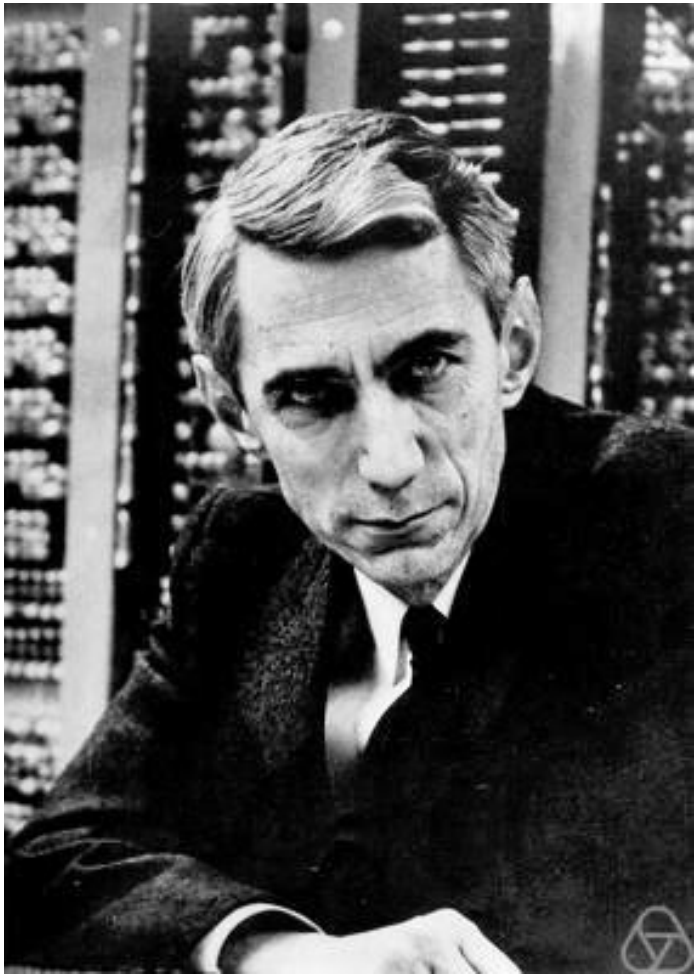
- For a binary classification problem, the gini impurity and information entropy can be visualized as:



where the horizontal axis refers to the proportion of one of the classes.

Claude Shannon--The Founder of Information Theory

- Shannon's information theory, centered around his **information entropy**, has a profound impact on communication theory, computer science, machine learning, NLP, Information criterion (AIC, BIC, …..).



Decision Trees - Gini impurity and Information Gain

- **Gini impurity** is a measure of node purity.
- When we are using **gini impurity** as the metric for splitting, our goal is to have two subregions as pure as possible by reducing the weighted sum of gini impurities.
- Similarly, when using **information entropy** we want the information gain as great as possible after the splitting.
- In practice, the behaviors of the two metrics are very similar.

CART

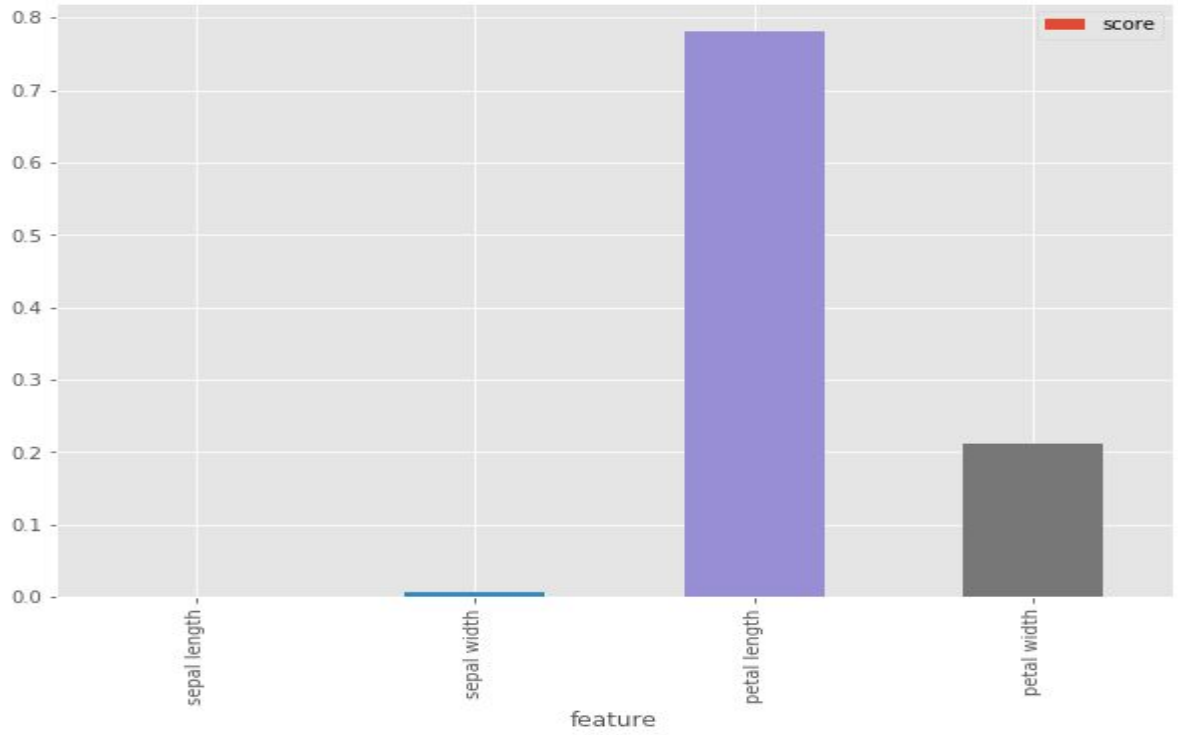
- The **decision tree** model is often given a name '**CART**', the abbreviation of 'classification and regression tree'
- The overfitting issue with a **regression tree** also occurs on a **classification tree**, with the same remedy applied
- Under the hood, both **regression trees** and **classification trees** try to approximate a function by locally constant functions:
  - The former tries to approximate the outcome values as a function on the feature domain
  - The latter (for **binary classification**) tries to approximate the probability function that the given sample belongs to a given class
- As the probability only takes values within [0,1], the use of **Gini** or **information entropy** ensures the approximated functions can still be interpreted as probability functions

Feature Importance

- Recall that the slopes of penalized linear regression models shrink to zero gradually. This allows the penalized linear models to output the relative importance ranking of the input features
- One important bonus of decision trees and its derivatives (random forests, boosting trees) is its output of feature importance as percentage scores
- For each feature, the tree outputs a score in percentage (i.e. between 0 and 1), with properties:
  - The total sum of all the feature scores is 1
  - The higher the score, more important the feature is
- The score can be interpreted as the percentage of contribution of the particular feature to the given tree model's success

Barchart of Decision Tree Feature Importance on Iris

- Train a simple decision tree on iris data set using default setting, results in the following barchart on the feature importance, with petal length being the most important, sepal length least important



The Meaning of Feature Importance

- For decision tree regressors and classifiers, the definitions of feature importance are slightly different
- For regression trees, one traces the total drop of **RSS** during the total node-splits and attributes it to the splits due to different features. Then the importance score is the percentage contribution of each feature to the total **RSS** drop
- For classification trees, one traces the total drop of **Gini** or **Information Entropy** (it depends on which criterion has been used)**,** and trace the percentage contribution of each feature to the total drop
- If a given feature never appears as the node-splitting feature in the given tree, its importance score is zero
- The above discussion can be extended to an ensemble of trees (bagged trees, random forests, boosting trees)

Decision Trees and Categorical Features

- We have taught that we need to dummify the categorical features in dealing with multiple linear regression
- The same observation works for **SVM**/**SVR** or neural networks
- Dummification produces a lot of new binary features, which exhausts computation resource
- For tree-based machine learning models, the dummification is often not necessary, unless the number of possible categorical values is huge
- Because the locally constant function built in a tree is naturally discontinuous, it can handle discrete data readily
- If the possible categorical values are limited compared to the number of tree nodes, the tree can split its nodes following different discrete values. We only need to relabel the categorical values by integers

Decision Trees Pros and Cons

- Pros:
  - Interpretability: easier to interpret than most other regression methods.
  - Easy to handle qualitative/categorical predictors.
  - Can be displayed graphically.
- Cons:
  - Instability: a small change in the data may result in very different splits.
  - Predictive accuracy usually not as good as other approaches.
- By aggregating (**ensembling**) many decision trees, the predictive performance can be improved substantially.

Hands-on Session

- Please go to the **"Decision Tree in Scikit Learn"** in the lecture code.

Outline

- Tree-Based Methods
  - The Intuition Behind the Tree Models
  - Decision Trees, A Geometric Perspective
  - **Bagging and Random Forests**

Why Ensembling?

- **Ensembling** is a statistical technique which enhances a collection of weak learners into a strong model with high complexity
- Don't we say 'three heads are greater than one'?
- There are two favors of **ensembling** in machine learning
  - **Parallel**: all the weak learners perform independently
  - **Sequential**: the latter in the sequence picks up the left-over from its predecessor. The procedure is known as **boosting**

- When a single weak learner is in-action, it produces a model deviating from the grand-truth (the hidden true model no one knows) a lot
- Ensembling is a methodology which helps to reduce the model variance, making the model estimation more accurate

Ensembling with Trees

- In this portion of the lecture, we discuss **parallel ensembling** using decision trees. Resulting in bagging trees and the more sophisticated random forests
- Roughly speaking, a parallel ensemble is like a committee where the final prediction is based on aggregating the individual members' outcome
- For regression problem, we can simply take the (weighted) average of all trees' outcomes
- For classification problem, we can take the majority vote (or other variants) among the team members

How Do We Make the Trees Different?

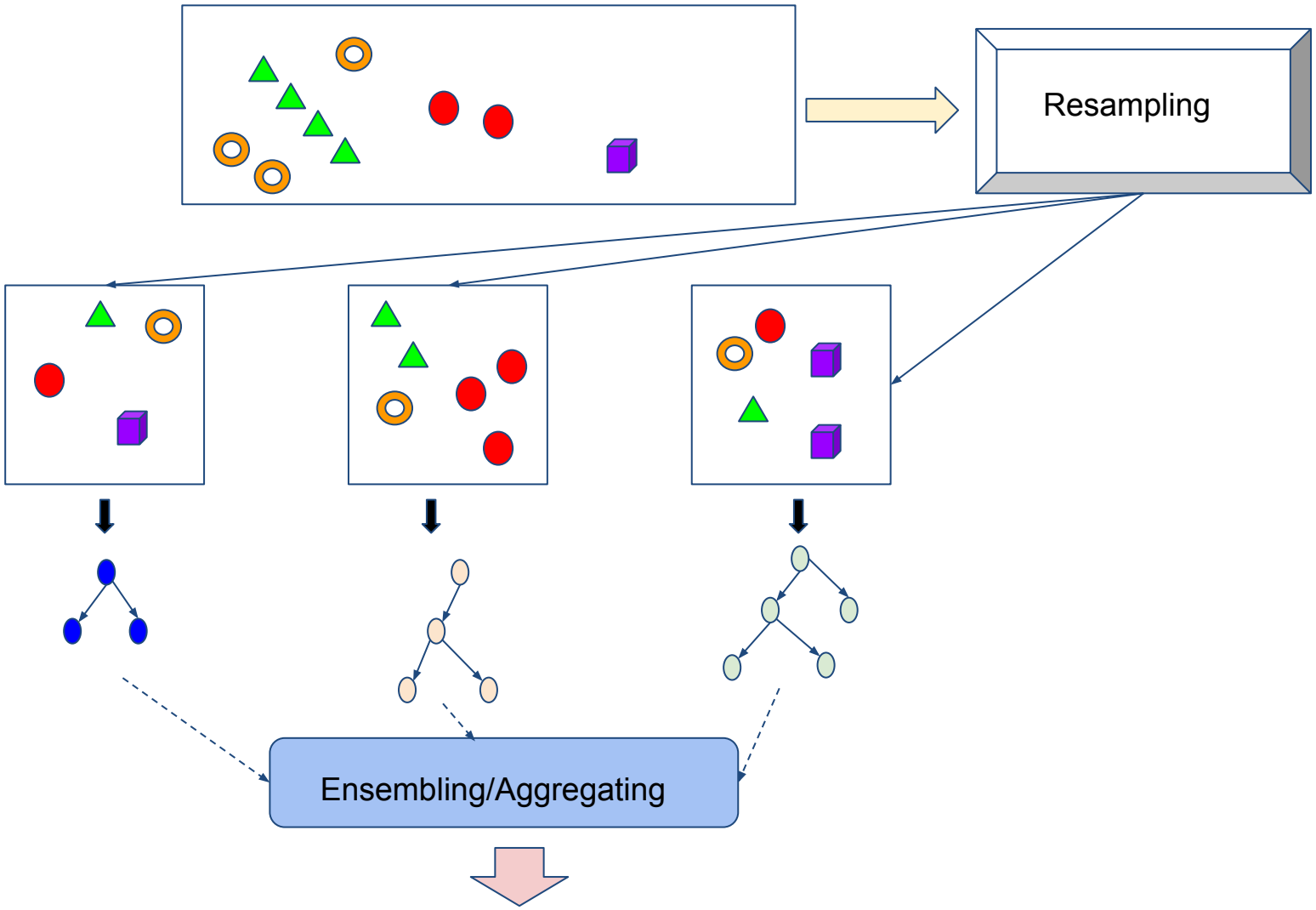
- The idea of **ensembling** looks good, only if we can make all the weak learners different from each other
- The idea of **bagging** is a major milestone in this direction
- **Bagging** means **bootstrap aggregating**
- Bootstrapping is a traditional statistical technique of creating many synthesized copies of data from the original data by **resampling with replacement**
- The synthesization procedure introduces additional distortions to the data which can be used to our advantage
- In the linear regression setting, the fitting of a linear model on the original data set produces a single set of estimated model parameter with wide confidence intervals (i.e. the true model parameters can be far away from our estimations)

Bagging Trees

- By bootstrapping, we fit linear models on all these synthesized data, producing a large family of coefficient estimations:
  - The average estimation based on all the biased estimation can offer a better model estimation
  - Bootstrapping can be used to estimate confidence interval without resorting to the exact formula
- The same idea can be carried out for bagging trees as well
- **Repeatedly** Resampling with replacement allows the tree model to probe beyond the original samples to enhance the predictive performance

- Traditional bagging resamples upon the full data set, but for larger data set, one can randomly select, say 70%, of the samples and then resample with replacement from the restricted subsets

Bagging Tree Illustration



Random Forest

Bias and Variance -- The Enemies of Predictive Models

- Theoretically, there are two main sources of predictive errors in a machine learning predictive task
  - Bias
  - Model Variance
- Living in a finite world, the machine learning algorithm tries to estimate the grand-truth based on partial knowledge on data sets of finite sample sizes
- Bias error refers to the given model's systematically under/over-estimating the true answer
- Model variance refers to the model's certainty on its estimation, or equivalently its sensitivity on the training sets
- Both bias square and model variance contribute to the final performance error

A Toy Example

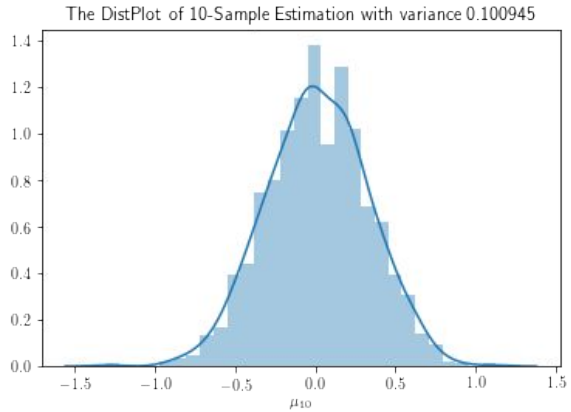
- Let us give a toy example of bias and model variance to illustrate the idea. Suppose that there is a hidden normal distribution with mean = 0, std = 1
- The probability distribution is completely hidden from the outside world that the 'model' can only estimate the true mean value based on a finite i.i.d. samples from the distribution

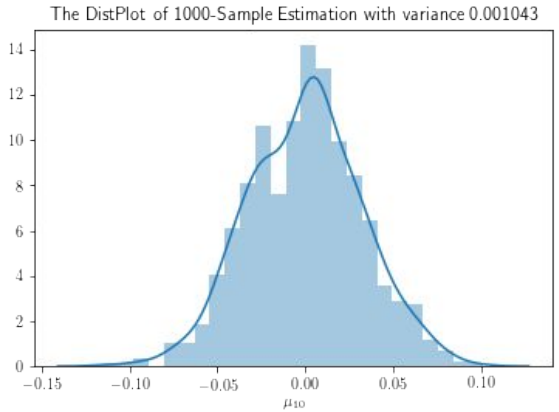
$$\mu_N = \frac{\sum_{i \leq N} \epsilon_i}{N}$$

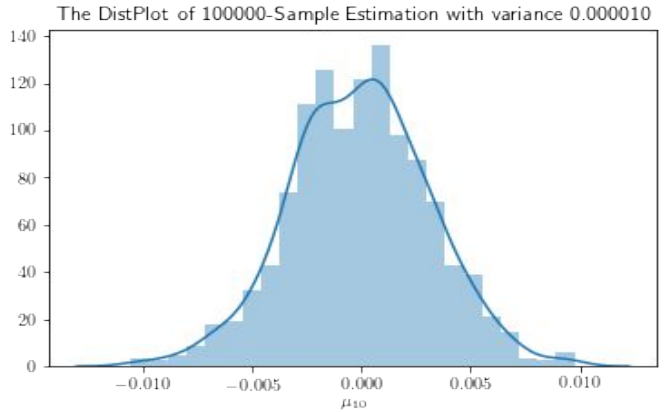
- Fixing an N, a single estimation never gives us the true answer 0. Yet we say the methodology of taking the sample mean (this is our model!) is an **unbiased** estimation: sometimes its estimation is positive, sometimes its estimation is negative, but there is no systematic tendency for it to overestimate or underestimate

What About Model Variance?

- In this example, the model variance refers to the range of possible μ_N with different i.i.d N-samples
- The following histograms illustrate their dependences on N







- We view the sample averages with a fixed number of samples N as from a single model. Evidently the model variances decreases w.r.t. Increasing N
  - A special case of the classical **central limit theorem** asserts that asymptotically for large N the sample mean μ_N always converges to the true mean (0 in our case) controls by the vanishing variance

Bias-Variance Trade-Off

- High model variance can cause overfitting, a discrepancy between the training and test sets predictions
- High model bias can contribute to underfitting, the inability of the model to capture the hidden relationship
- Suppose that the features and target values satisfy a hidden linear relationship (up to Gaussian noises). Then multiple linear regression model provides an unbiased estimation on the true coefficients, with the estimation error coming from model variance (confidence intervals)
- Penalized linear model (Ridge and Lasso, etc) introduces bias in exchange for a reduction of model variance
- The sweet spot of the trade-off is found out by doing a grid search on the penalty/regularization scale hyperparameter

Random Forests and Bias Reduction

- Through the lens of bias-model variance, decision tree, as a weak learner, has both high biases and model variances
- The idea for bagging of decision trees is to average many noisy trees and hence to reduce the model variance
- However, since each tree generated in bagging is based on identical features, the expectation value of the averages is the same as the expectation of any one of them. This means the bias will not be much improved
- Random forests is a enhancement of bagging that builds a large collection of de-correlated trees, and then averages them

Random Feature Selections

- **Decision trees**, as weak learners, have relative low complexity to capture a complex nonlinear phenomena. This leaves a lot of unexplained relationship not captured by a single tree
- To build a high performance machine learning model from DT, we need to make sure different trees within the same **ensemble** capture different facets of the nonlinear pattern
- **Random forests** do so by introducing randomness in the individual tree generation process -- at each tree node, we seek the loss function minimization only among a randomly chosen subset of features
  - This allows this particular node split to learn the less prominent relationship which may be ignored by the other trees/nodes

Out of Bag Errors

- The sample **bootstrapping** procedure resamples the data set allowing replacements
- Often this will introduce duplicated samples in the bootstrapped data set, which also means some data points will be left unused in that particular resampled data set
- The number of unused samples depend on the random resampling, which makes them a random variable
- In a bagged tree or a random forest, the same data point can be out-of-bag for some bootstrapped instances, but often not all of them
- As the sample is not explicitly used in the trainings of these trees which avoid using this particular data point, the prediction error of these trees on this data point is particular valuable to gauge the model performance
- We define **oob** error as the mean error running through all the samples using **ONLY** the trees which avoid the sample in training

OOB Error and Test Error

- Even though **oob error** can differ from the test set error, often it provides a good estimation of the test set error when the sample size of the data set/number of trees is large enough

Random Forest in Scikit-Learn

- **Random forests** in sklearn inherit the basic hyperparameters from **decision trees**
- It is subdivided into separated models:
  - **RandomForestRegressor**
  - **RandomForestClassifier**
- The hyperparameter **n_estimators** controls the number of trees in the ensemble
- The hyperparameter **max_features** controls the maximum number of features randomly selected at each node

- The best **n_estimators** and **max_features** can be found out by grid-search
- Based on early experiments, the default max_features is chosen to be the square root of the total number of features

Boosting Tree - XGBoost

- In the past few years, the newer generation boosting tree model, **XGBoost**, has become very popular after winning a few Kaggle competitions.
- It offers much faster training time, higher accuracy and scalability for large data sets.
- There exist R and scikit-learn wrappers for **XGBoost** as well.

Multiclass Classification

- Among the classification algorithms we have learned, logistic regression, LDA, QDA, SVM are naturally binary classifiers.

Question: **How do we handle multi-class classification?**

- Some machine learning technique, like logistic regression, neural network, tree based models…., can be extended naturally into multi-class classifiers.
- The other popular direction is to reduce the original multi-class classification problem into multiple binary classification problems.
- This is achieved through the concept of **ensembling**.
- Among them the easiest and most widely used ones are the:
- **One vs One** Classification Scheme
- One vs The Rest (**One vs All**) Classification Scheme

One vs One

- For a K class multi-class classification scheme, we construct K(K-1)/2 binary classifiers specialized in discerning class i and j for distinct i, j in {1,2,...K}.

- For example, for K = 4, we need to train **6** classifiers to classify the pairs of classes **(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)** on sub-samples of these class pairs.
- Then one either takes the majority votes or uses the pairwise discriminant functions to compute an aggregate score, and chooses the class with the maximum score value.

One vs One

- **Con**: The number of class pairs grow quadratically with respect to K
- **Pro**: For each binary classifier in the list, the sample size can be much smaller.

- **SVMs**' computation time is very sensitive to the sample size (and the number of features), so the opportunity to cut down sample size outweighs the growth of total classifiers used.

One vs the Rest (One vs All)

- **One vs the Rest** trains **K** binary classifiers, each is focused on whether the given sample is in class k or **NOT** in class k.
- Conceptually this can be viewed as dummifying the **K** valued categorical outputs into **K** columns of binary output values.
- Then one predicts the class value based on the highest discriminant function value (e.g. probability value)

- Pro: The number of classifiers needed grows linearly with respect to **K**
- Con: Each classifier trains on the full samples.
- Logistic Regression, Discriminant Analysis often use this approach.

Summary

- There exist other schemes which reduce a multi-class classification into multiple binary classifications.

- Coding 1, 2,...,K into binary representation
- Coding 1, 2,...,K into the leaves of a binary decision tree, etc.
- The discussion is beyond the scope of our study.

Hands-on Session

- Please go to the **"Random Forest in Scikit Learn"** in the lecture code.