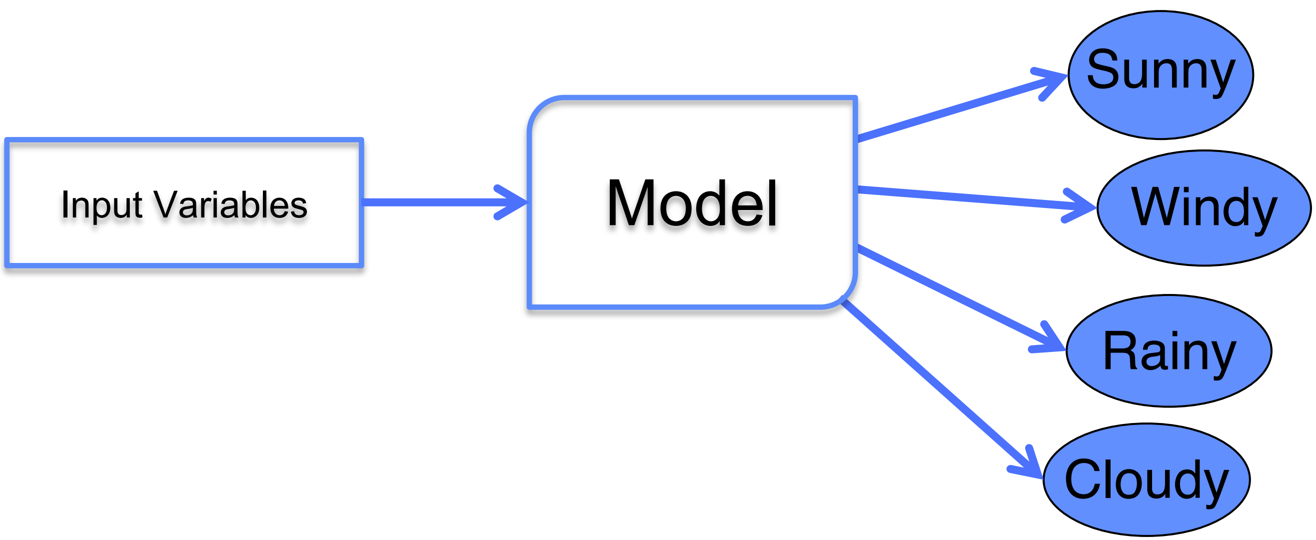
Classification

Mai H. Nguyen, Ph.D.

What is Classification?

- **Given input variables, predict target variable.**
  - Model needs to learn relationship between input data and target
  - Target variable is *categorical*
  - Other names for ‘target’
    - label, class, class variable, output



Data for Classification

Input Variables

Target Variable

Temperature	Humidity	Wind Speed	Weather
79	48	2.7	Sunny
60	80	3.8	Rainy
68	45	17.9	Windy
57	77	4.2	Cloudy

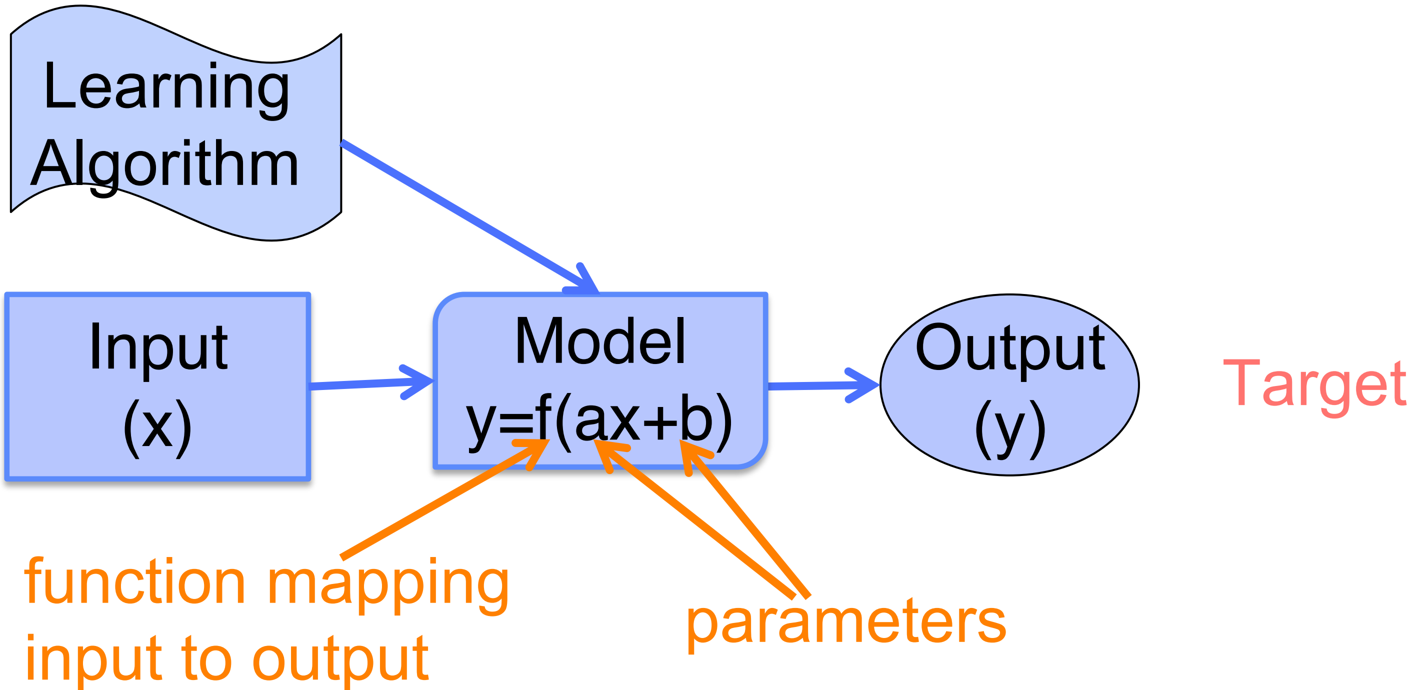
Classification is **supervised** learning.

Target Variable in Classification

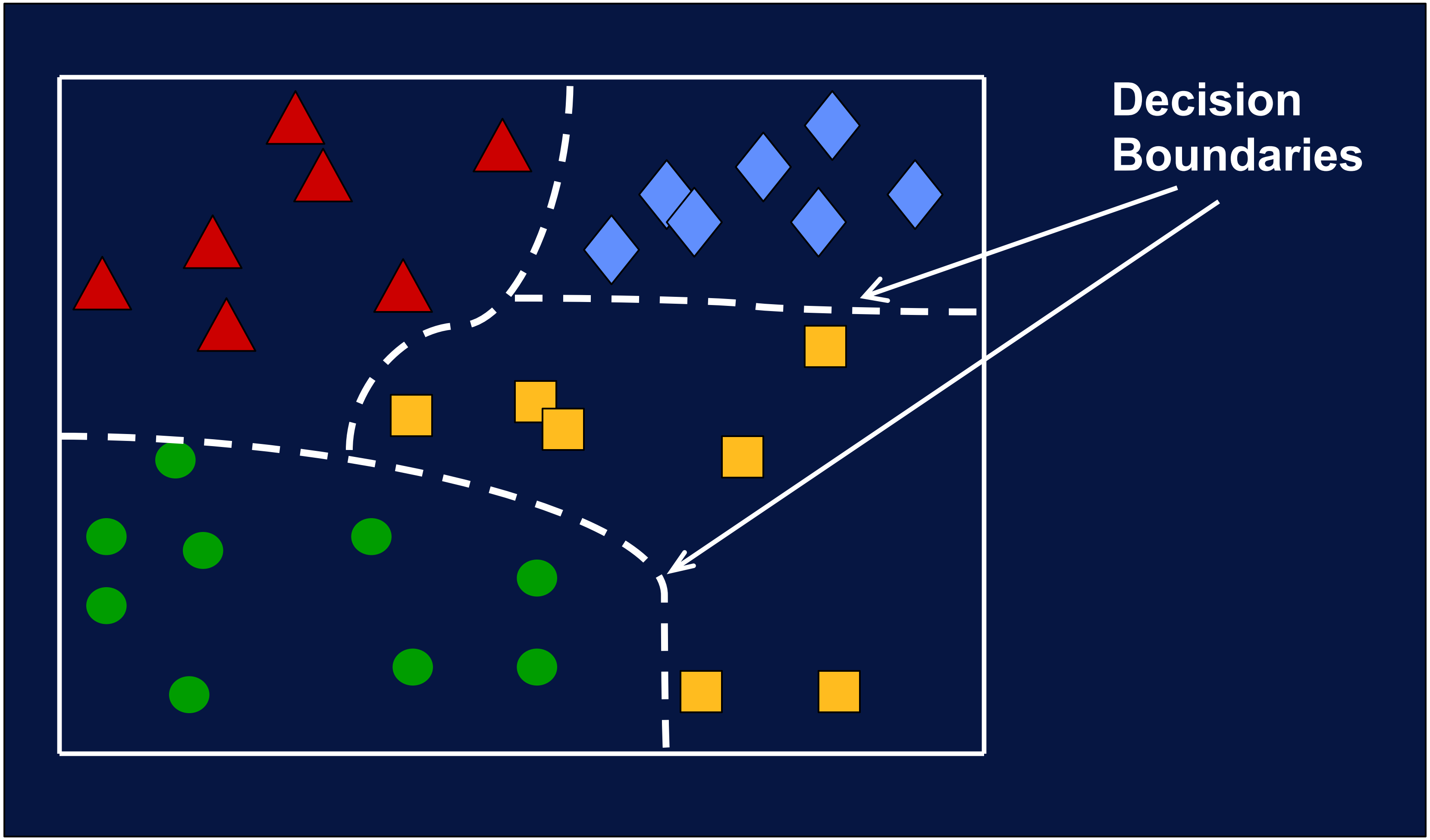
- **In classification, the target variable is always of *categorical* type**
- **Binary Classification**
  - One of *two* classes for target
- **Multiclass or Multinomial Classification**
  - One of *many* classes for target
  - “many” = more than 2
- **Multilabel Classification**
  - One *or more* classes for target

Building Classification Model

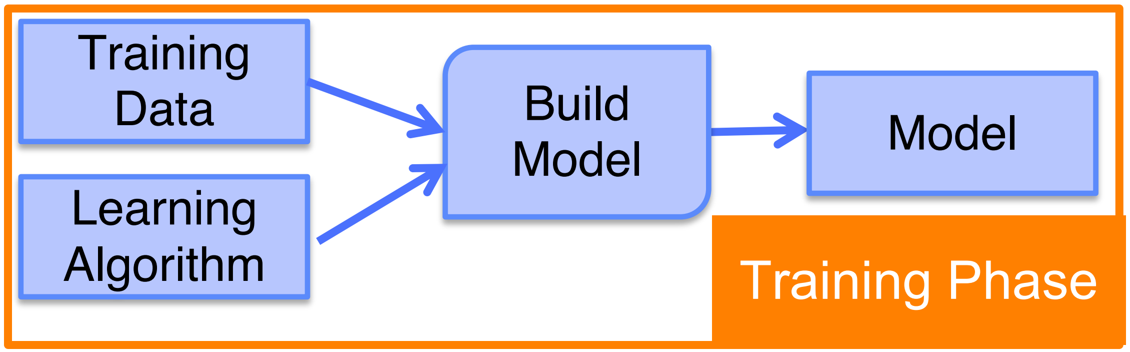
- Goal: Get model outputs to match targets
- Learning algorithm used to adjust model parameters to minimize difference between outputs and targets
- Parameters are learned or estimated from data
  - "fitting the model", "training the model", "build model"



Building Classification Model

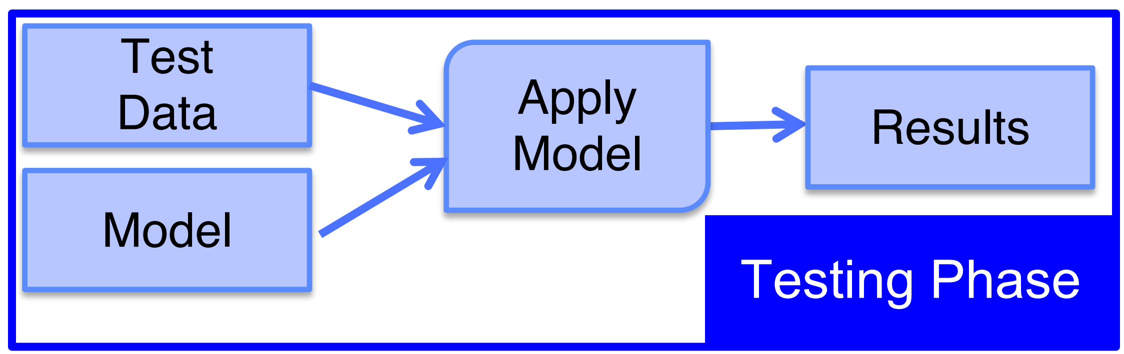


Building vs. Applying Model



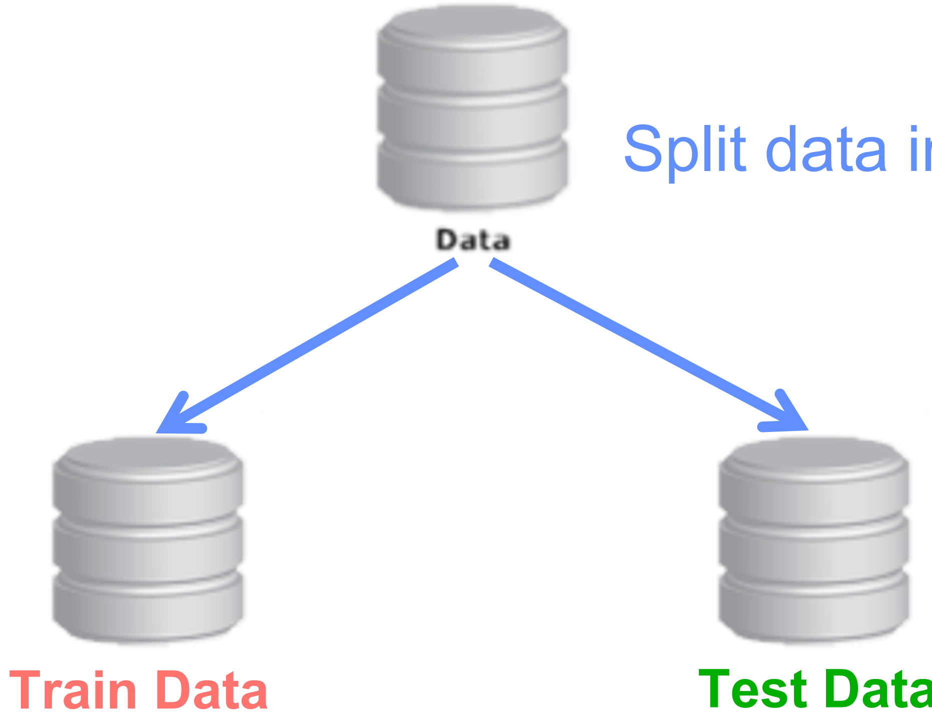
Adjust model parameters

Test model on new data



Generalization

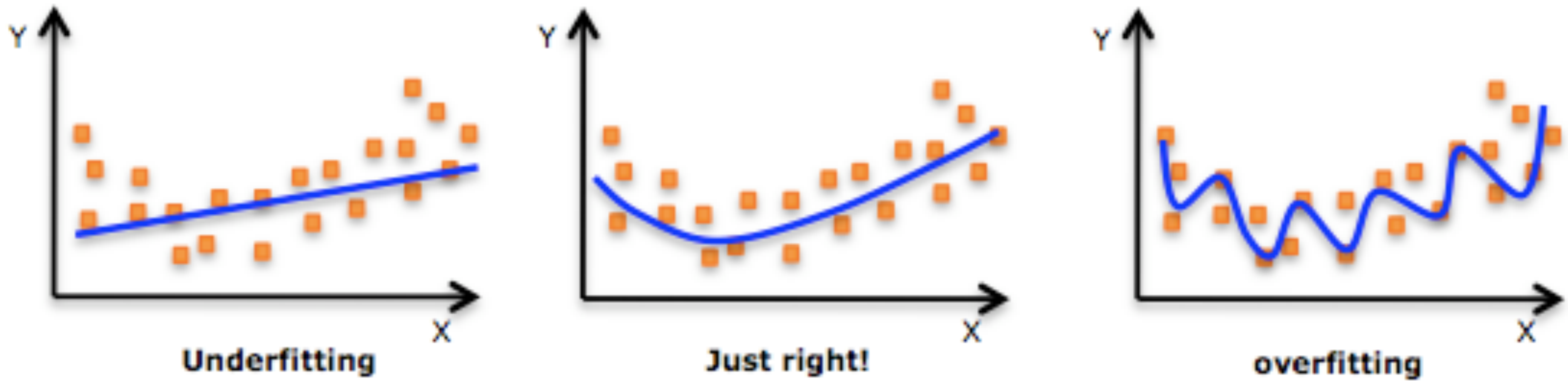
Split data into two sets



Build model on train data

Test model on test data

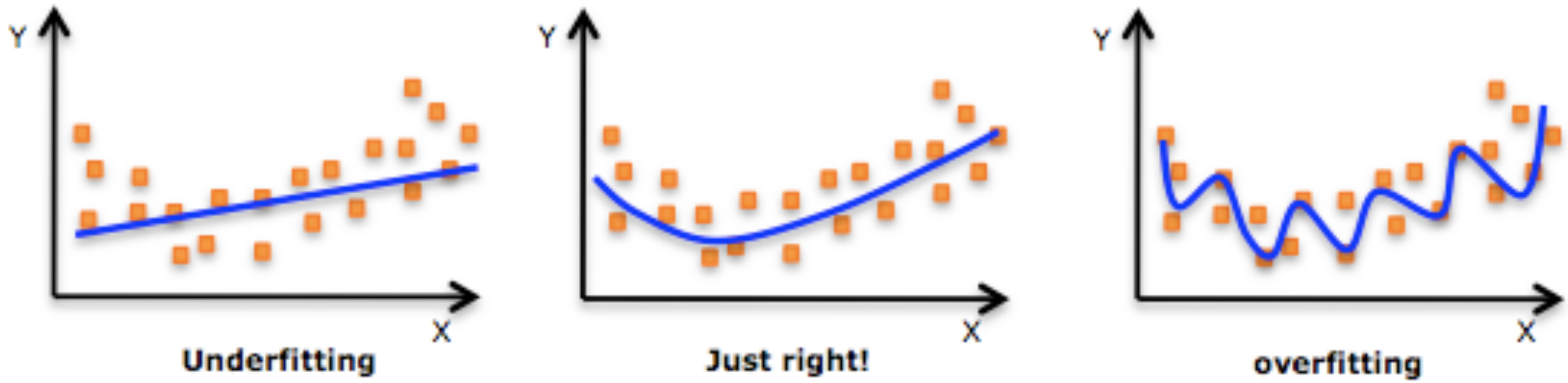
Overfitting



http://stats.stackexchange.com/questions/192007/what-measures-you-look-at-the-determine-over-fitting-in-linear-regression

Underfitting	Just Right	Overfitting
Model has not learned structure of data	Model has learned distribution of data	Model is fitting to noise in data

Overfitting



Underfitting	Just Right	Overfitting
High training error	Low training error	Low training error
High test error	Low test error	High test error

Overfitting & Generalization

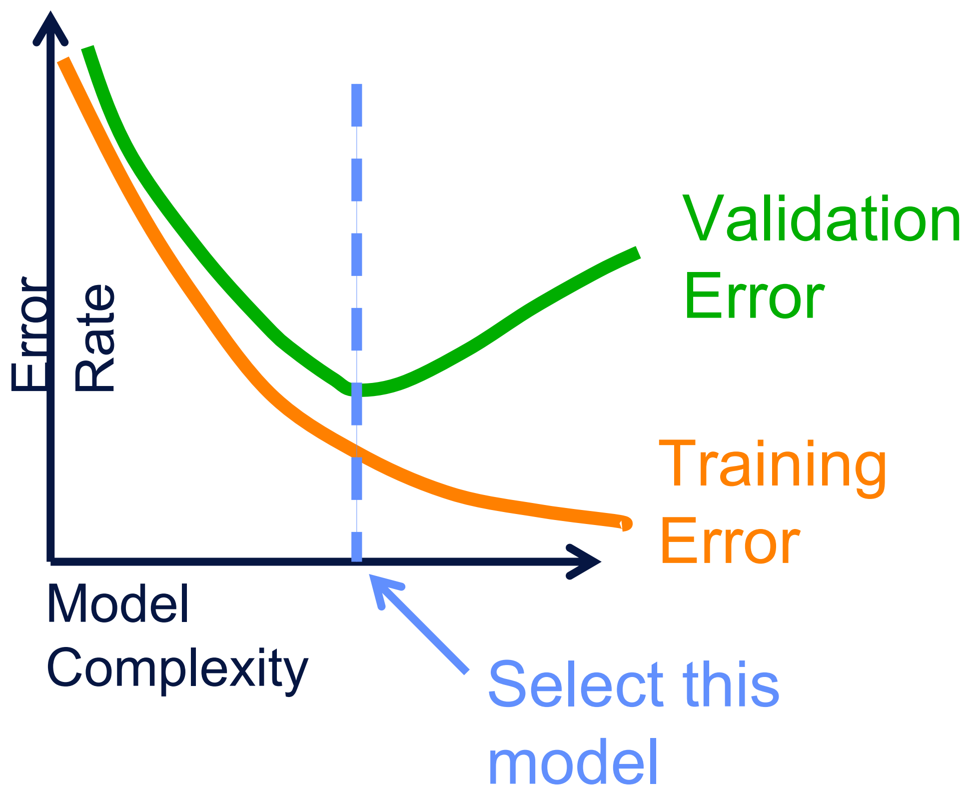
- **Reasons for overfitting**
  - Training set is too small
  - Model is too complex, i.e., has too many parameters
- **Overfitting leads to poor generalization**
  - Model that overfits will not generalize well to new data

Overfitting & Generalization

- **Validation set**
  - One way to address overfitting and estimate generalization performance
- **Idea:**
  - Divide training set into multiple datasets
    - Training set: Used to fit model parameters to data
    - Validation set: Used to validate model performance
  - Monitor performance on training and validation sets to determine when to stop training.

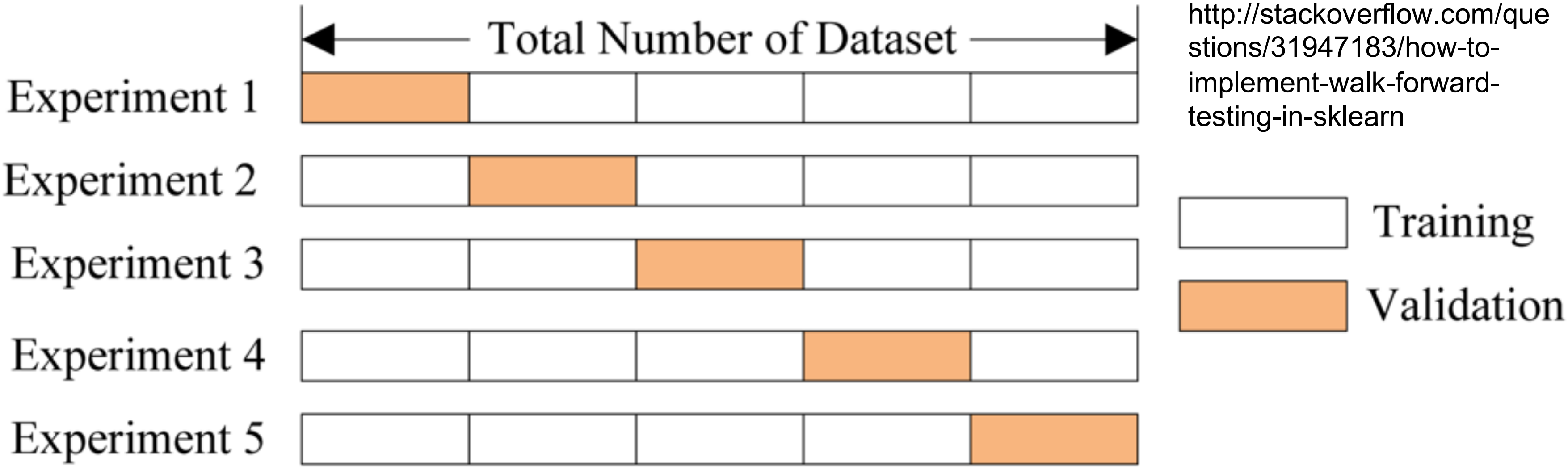
Validation with Holdout Set

- Overfitting is occurring if training error decreases while validation error increases.
- Model with best generalization performance is one with lowest validation error.



Cross-Validation

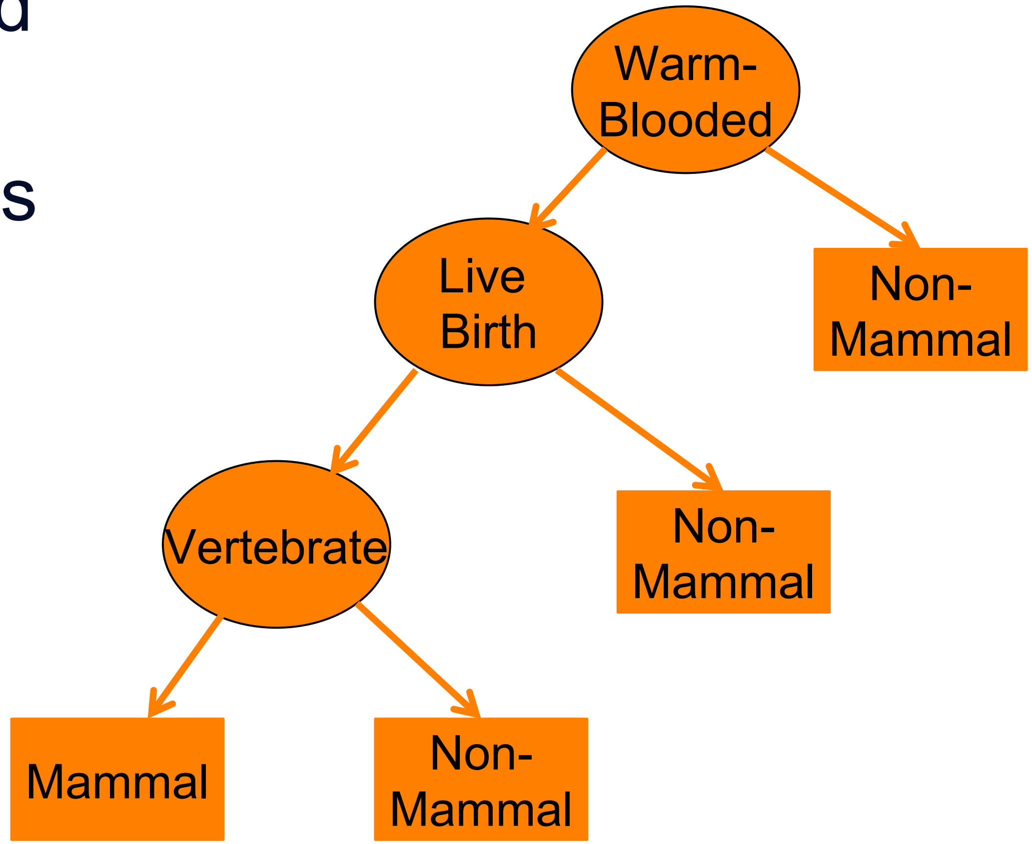
- **K-fold cross-validation**
  - Partition data into k disjoint datasets
  - For each iteration, use 1 partition for validation and the rest for training.
  - Repeat process k times. Each partition is used for validation exactly once.
  - Overall error estimate (generalization performance) is average of error rates for k iterations.



Source: http://stackoverflow.com/questions/31947183/how-to-implement-walk-forward-testing-in-sklearn

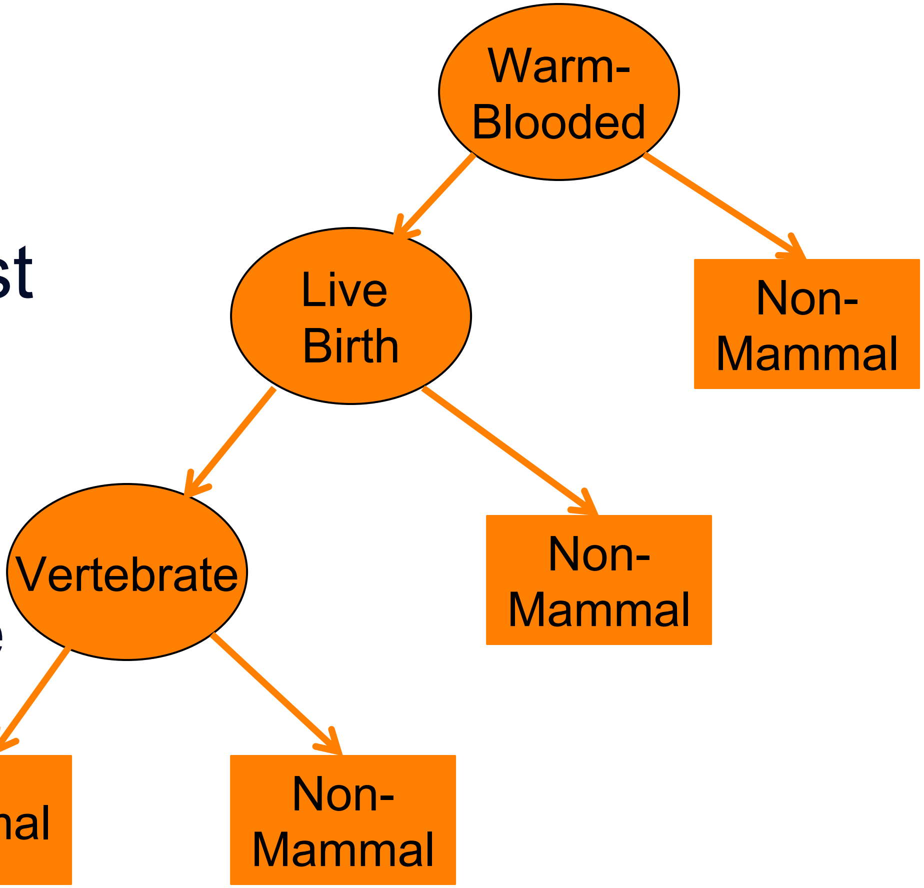
Decision Tree

- Hierarchical structure with nodes and directed edges
- Root and internal nodes have test conditions
- Leaf nodes have class labels
- Paths from root to leaf represent classification rules



Classification Using Decision Tree

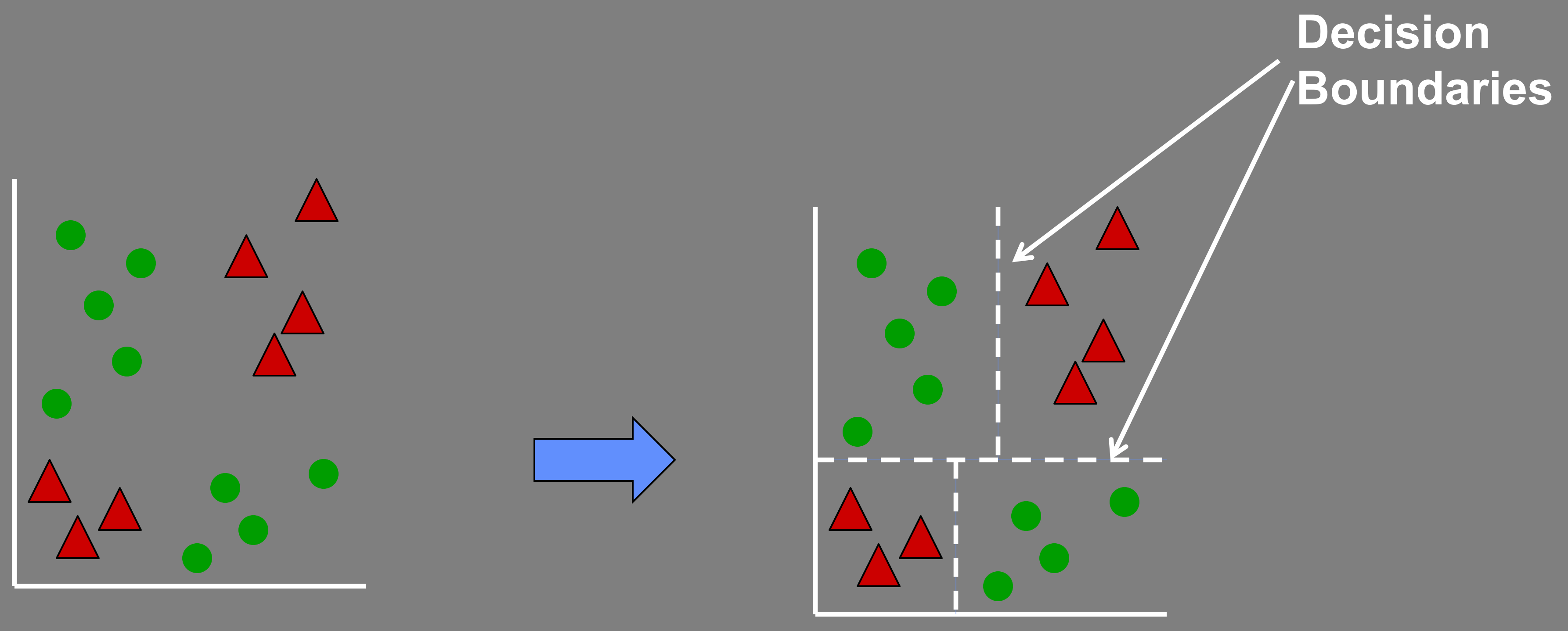
- Traverse tree from root to leaf
- At each node, answer to test condition determines child node to move to
- Category at leaf node determines label for sample



Warm-Blooded	Live Birth	Verte-brate	Target Label
Yes	Yes	Yes	Mammal

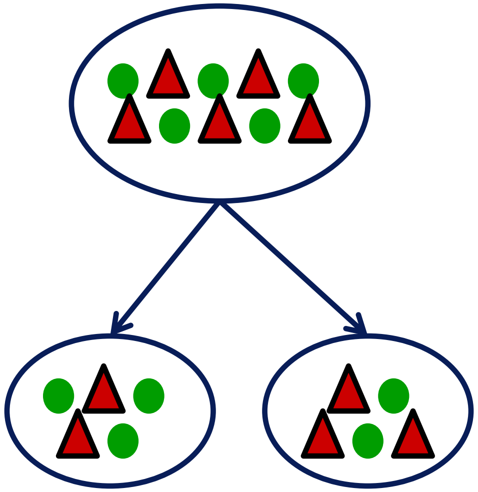
Constructing Decision Tree

- Split data into “pure” regions
- Classification decision based on these regions



Constructing Decision Tree

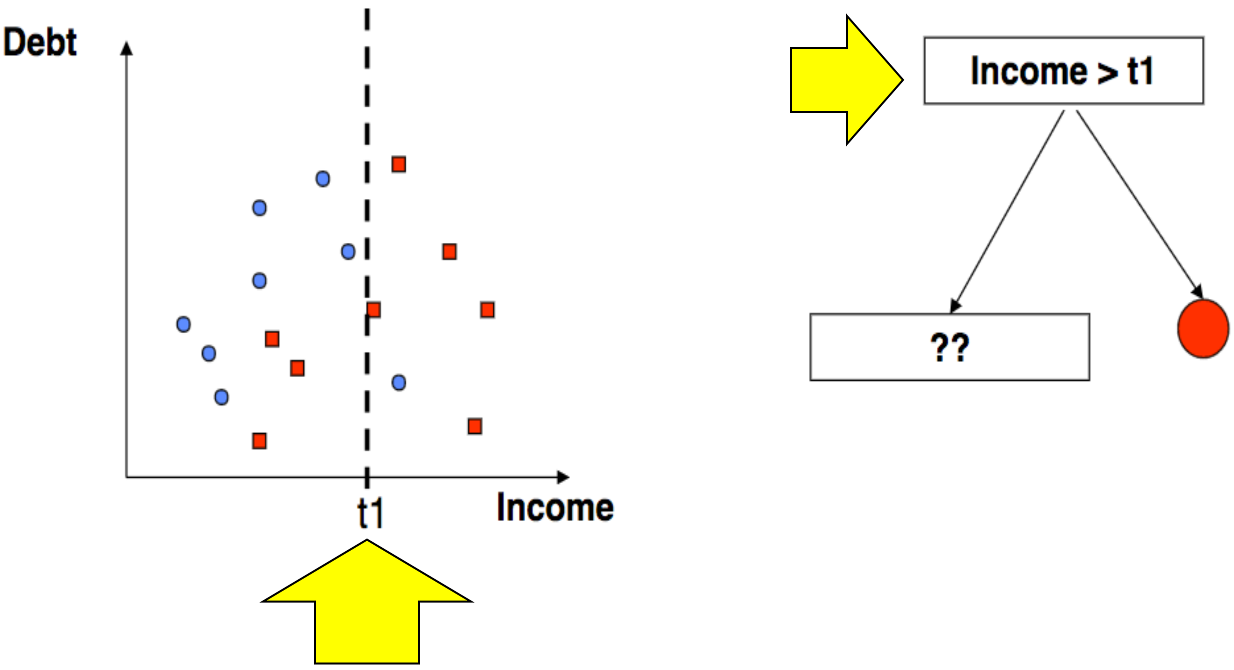
- Start with all samples at a node
- Partition samples based on input to create purest subsets
- Repeat to partition data into successively purer subsets
- Also referred to as ‘tree induction’



Tree Induction

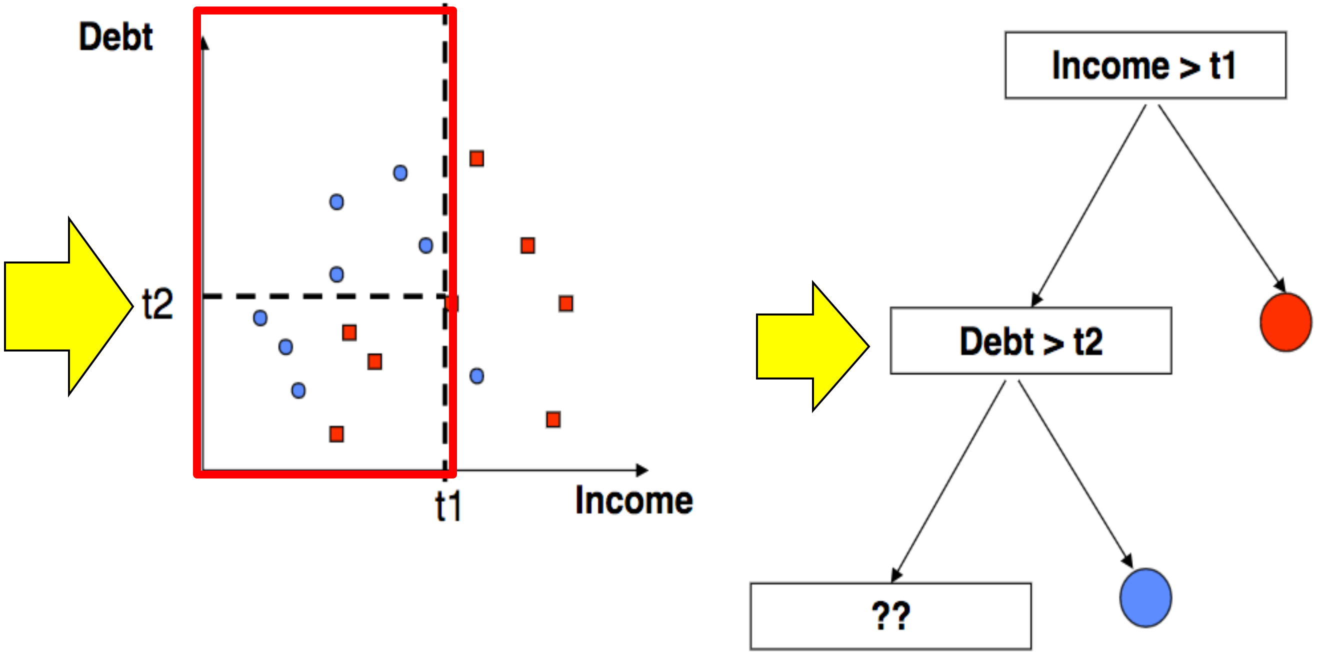
Tree Induction Example

- Split 1



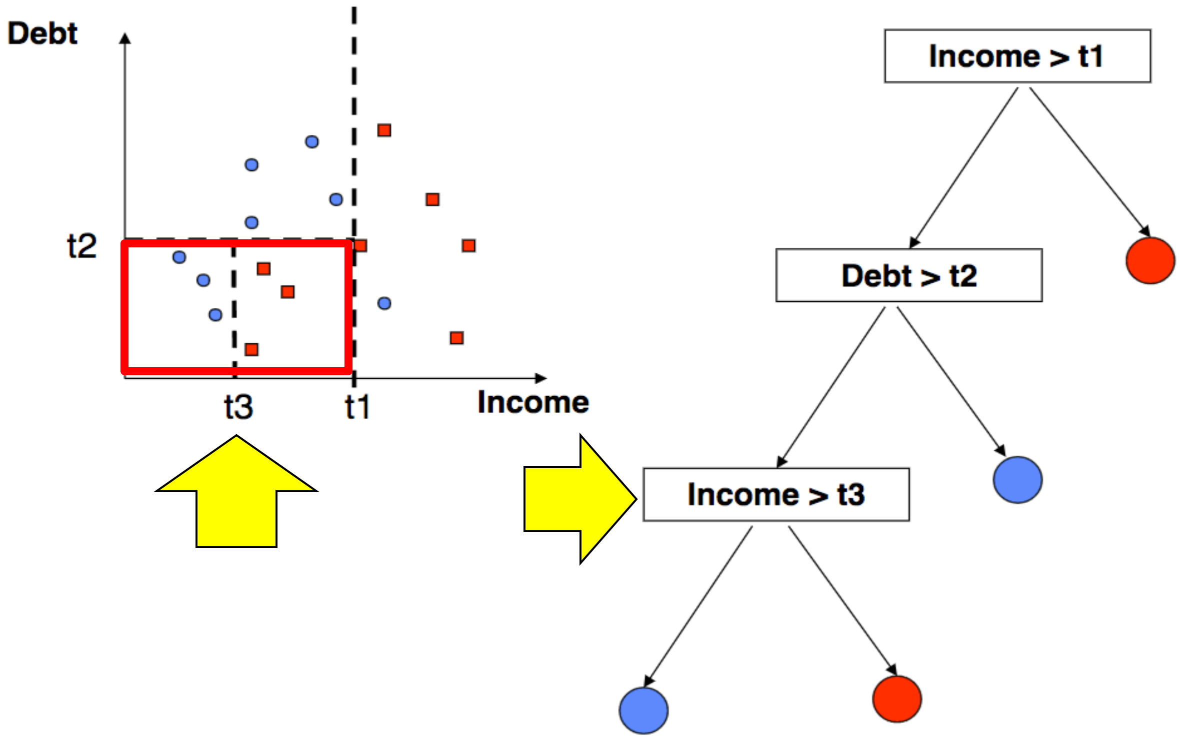
Tree Induction Example

- Split 2



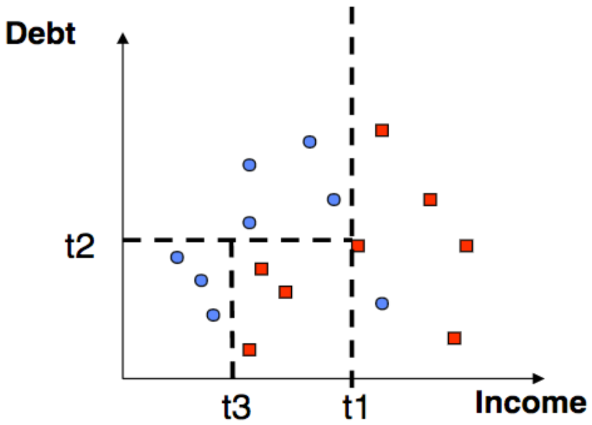
Tree Induction Example

- Split 3

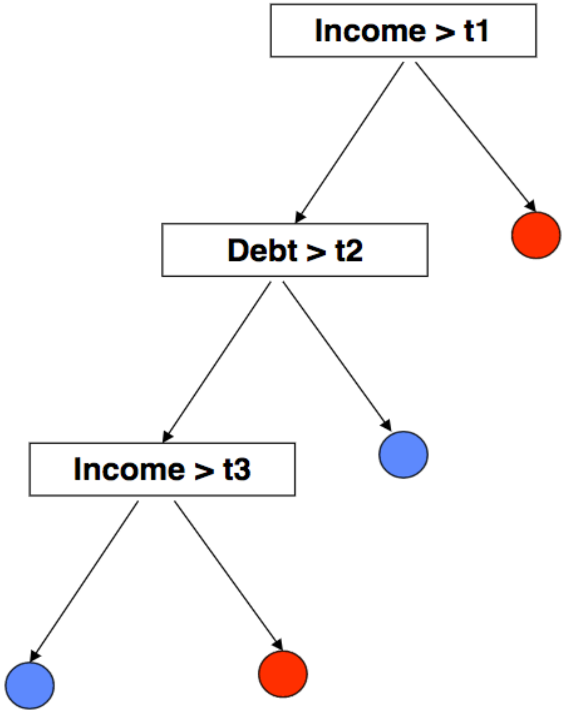


Tree Induction Example

Decision Boundaries



Resulting model



Decision Tree Classification

- **Pros**
  - Resulting tree often simple to understand and interpret
  - Tree induction algorithms relatively computationally inexpensive

- **Cons**
  - Induction algorithms make locally optimal decisions. No guarantee of globally optimal model
  - Small variants in data can generate different trees

Ensemble Methods

- **“ensemble”:**
  - a group producing a single effect (from Merriam-Webster)
- **Idea:**
  - Combine several simple models into more complex one
- **Approach:**
  - Construct a set of models from training data
  - Prediction is made by combining outputs of the multiple models
    - Classification: Combine votes of classifiers

Ensemble Methods

- **Advantage**
  - Ensemble learning generates more robust model with is less susceptible to overfitting and generalizes better
- **Rationale**
  - Ensemble with majority voting
    - Base classifiers may make mistakes, but ensemble will misclassify a pattern only if over half of base classifiers are incorrect.
  - Intuitively, combining decisions from multiple “experts” may be more reliable than relying on a single “expert”
- **Approaches**
  - Bagging
  - Boosting

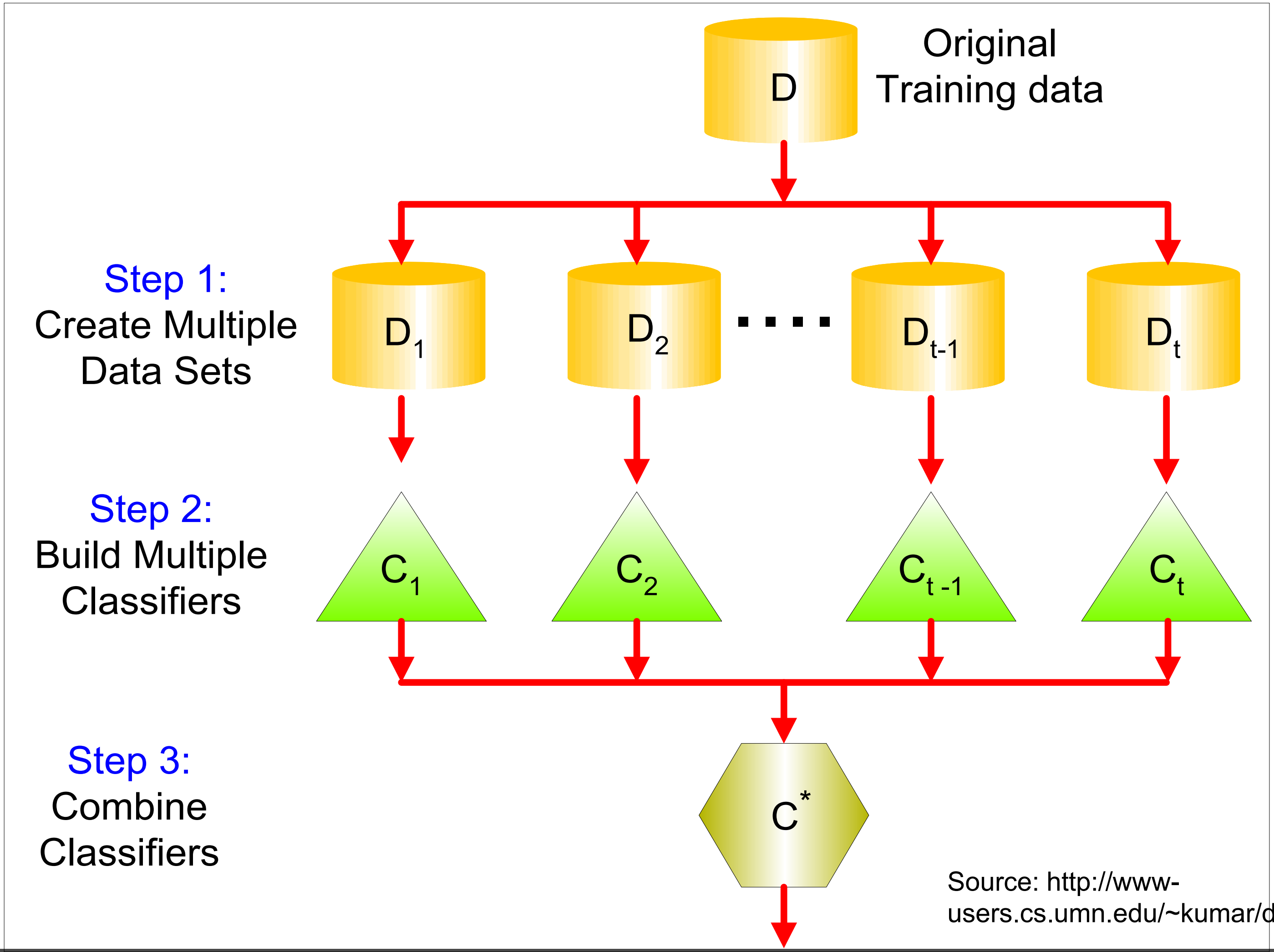
Ensemble Method: Bagging

- **Bagging stands for “bootstrap aggregation”**
- **Approach:**
  - Sample training data set with replacement to construct bootstrap samples
  - Build separate classifier on each bootstrap sample
  - Each classifier predicts class label for unknown record
  - Bagged classifier takes majority vote
- **Generalization can be improved since variance of individual base classifiers is reduced**

Random Forest

- **“forest”**
  - Ensemble method
    - Model is composed of set of decision trees => forest!
- **“random”**
  - For each tree, only subset of variables used to determine best split.
  - Subset of attributes is determined randomly.
- **Idea:**
  - To improve generalization over single decision tree

Random Forest – Illustration



Source: http://www-users.cs.umn.edu/~kumar/dmbook/index.php

Randomness in Random Forest

- **Randomness can be incorporated in several ways:**
  - Forest-RI
    - Random attribute selection: At each node, randomly select subset of input attributes to consider in splitting node.
  - Forest-RC
    - Random combinations of attributes: At each node, randomly select subset of input attributes to be linearly combined. These new attributes are considered in splitting node.
  - Randomly select best split:
    - At each node, randomly select one of F best splits.

Random Forest

- **Ensemble of decision tree classifiers**
  - Bagging is used to generate bootstrap samples for base decision trees
  - Base decision tree built by using only subset of attributes to determine split at each node
    - Subset of attributes is determined randomly
  - Final classification is based on the majority vote

Boosting Methods

- **Ensemble approach**
  - Uses multiple models for prediction
- **Boosting**
  - Combine set of weak learners to create final strong learner
  - Base models ("weak learners") added iteratively until no further improvements can be made or max number of models have been added
  - Weighted aggregation of base models' outputs used as final prediction

AdaBoost

- **Adaptive Boosting**
  - Adaptive: New models are built based on errors from previous ones.
- **Main ideas**
  - Misclassified samples are weighted more
  - New models focus more on samples that are difficult for existing models
  - Models are weighted relative to their predictive performance
  - Final prediction is weighted average of base models

XGBoost

- **Gradient Boosting**
  - New models are trained to minimize residuals (i.e., errors) of existing models
  - Loss function combines error and penalty term for model complexity
  - Gradient descent used to minimize loss when adding new models
- **eXtreme Gradient Boosting**
  - Implementation of gradient boosted trees
  - Optimized for execution speed and model performance
  - Uses parallelization and distributed computing to speed computation

Ensemble Models

- **In practice, often results in improved performance due to lower variance**
- **Training takes longer**
- **Ensembles more difficult to understand than single models.**
    - But can provide feature importance

Classification – Key Points

- **Classification task**
  - Predict categorical variable from input variables
- **Overfitting & Generalization**
  - Avoid overfitting to have model generalize to new data
  - Techniques: use validation set (e.g., cross-validation)
- **Algorithms**
  - Decision Trees, Random Forest, Boosting Algorithms
  - Others: k-nearest-neighbor, naïve Bayes, logistic regression, neural networks, …

Sources

- A. Liaw et al. Package 'randomForest'. Retrieved from https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
- P. Tan, M. Steinbach, & V. Kumar. *Introduction to Data Mining*. Pearson: 2005.
- T. Therneau, B. Atkinson, & B. Ripley. Package 'rpart'. Retrieved from https://cran.r-project.org/web/packages/rpart/index.html
- XGBoost: https://xgboost.readthedocs.io/en/latest/

Questions?

Classification Hands-On

- **Predict whether it will rain tomorrow**
  - Decision tree
  - Random forest
- **Using same Australian weather dataset from EDA hands-on**