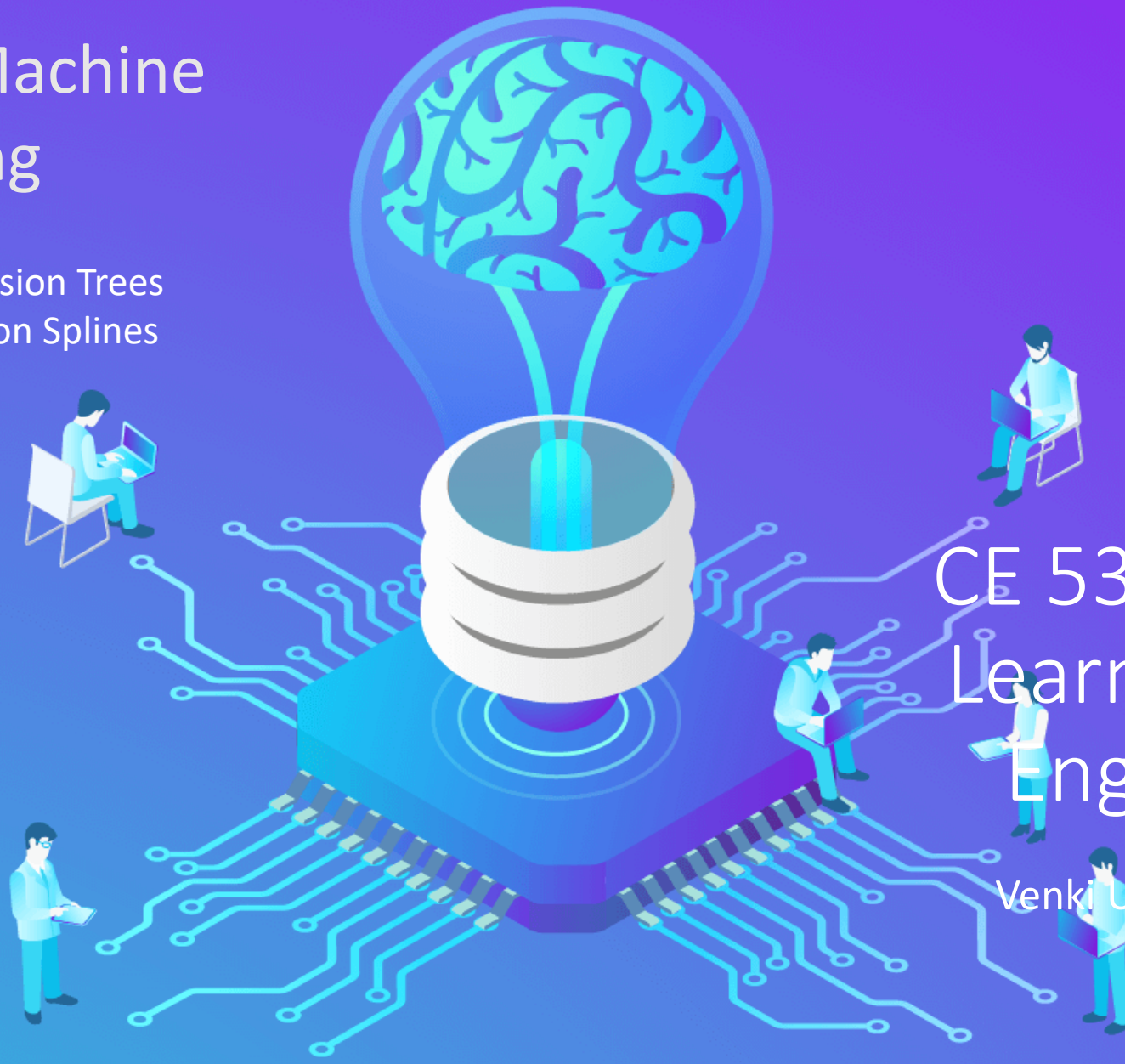
Python for Machine Learning

Classification and Regression Trees
Multi-Adaptive Regression Splines

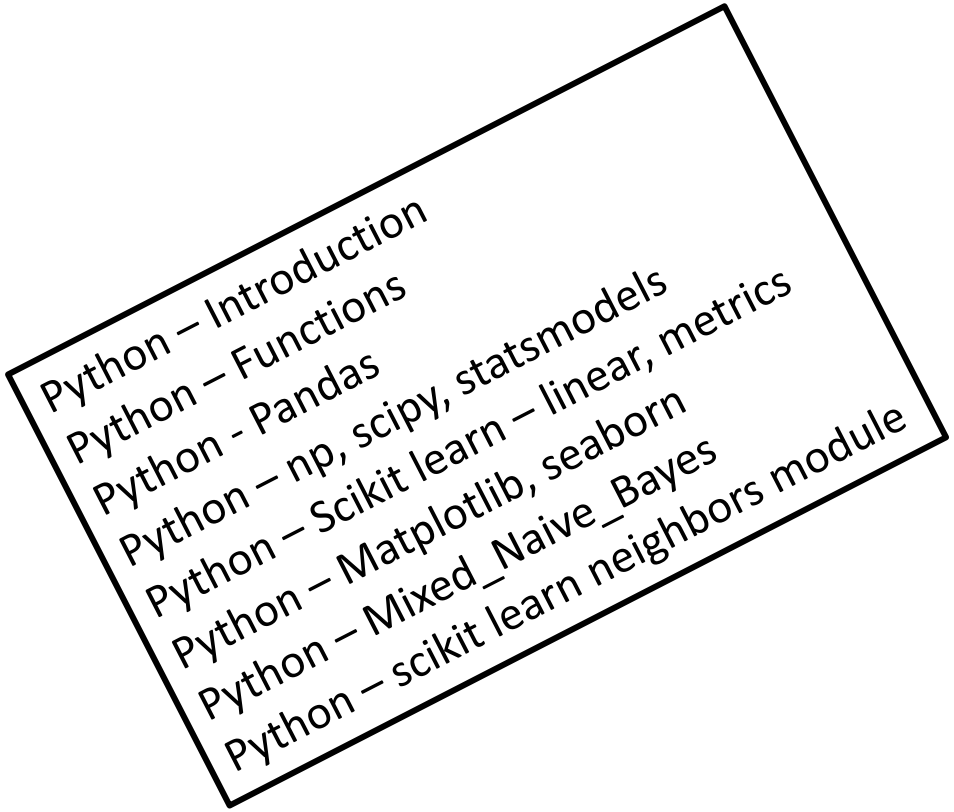


CE 5331 Machine Learning for Civil Engineers

Venki Uddameri, Ph.D. , P.E.

Recap

- What is Machine Learning
- How is it useful for Civil Engineers
- Overview of Machine Learning Methods
- Linear Regression
  - Bivariate
  - Regression interpretation
  - Multivariate
- Logistic Regression
  - Maximum likelihood estimation
  - Regularization (introduction)
- Naïve Bayesian Classifier
  - What is it
  - What makes it naïve
  - Bayes theorem
  - Prior, likelihood and posterior
- K-Nearest Neighbor
  - How does the algorithm work
  - Why is it a lazy learner
  - How to do regression and classification
- Introduction to Decision Trees
  - Fundamentals
  - Information Gain, Entropy and Gini Index
  - ID3 algorithm



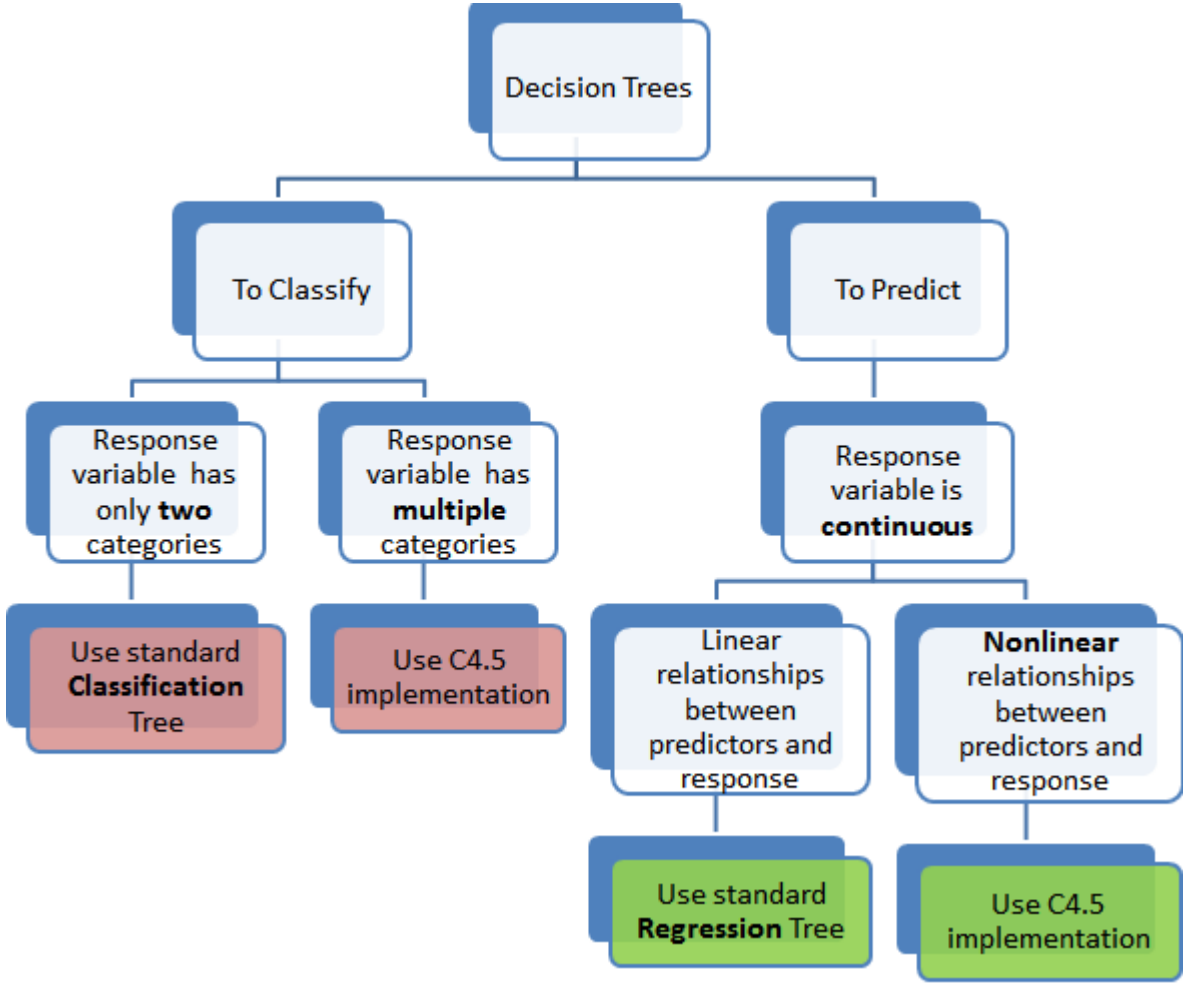
In this module we shall look at Classification and Regression Tree (CART) Algorithm

Classification or Regression?

- There are some newer versions of decision trees
  - Classification and Regression Trees (CART)
  - Multiadaptive Regression Splines (MARS)

CART and MARS are now considered de facto algorithms for building single tree models

C.5 is another algorithm that is popular as well



C 4.5/5.0 Algorithm

- C4.5 algorithm was developed by Quinlan based on the ID3 algorithm
- ID3 only works on discrete data
  - You have to turn continuous data to discrete
- C4.5 algorithm works on both continuous and discrete data
  - The algorithm looks for optimal splits from the records of continuous data
- C4.5 algorithm works on divide and conquer strategy
  - Records are assigned weights to account for other attributes not considered
  - Can handle missing values
- Pruning in C4.5 algorithm
  - Sub-tree replacement → sub-tree is replaced by a leaf

C5.0 Algorithm idea is very similar to C4.5 but improves computational backend for necessary calculations

C4.5/C5 algorithm

The attribute with the highest informational gain is computed using the following formulas:

Entropy: $E(S) = \sum_{i=1}^{n} -Pr(C_i) * log_2 Pr(C_i)$

Gain: $G(S, A) = E(S) - \sum_{i=1}^{m} Pr(A_i) E(S_{Ai})$

E(S) – information entropy of S

G(S,A) – gain of S after a split on attribute A

n – nr of classes in S

$Pr(C_i)$ – frequency of class C_i in S

m – nr of values of attribute A in S

$Pr(A_i)$ – frequency of cases that have A_i value in S

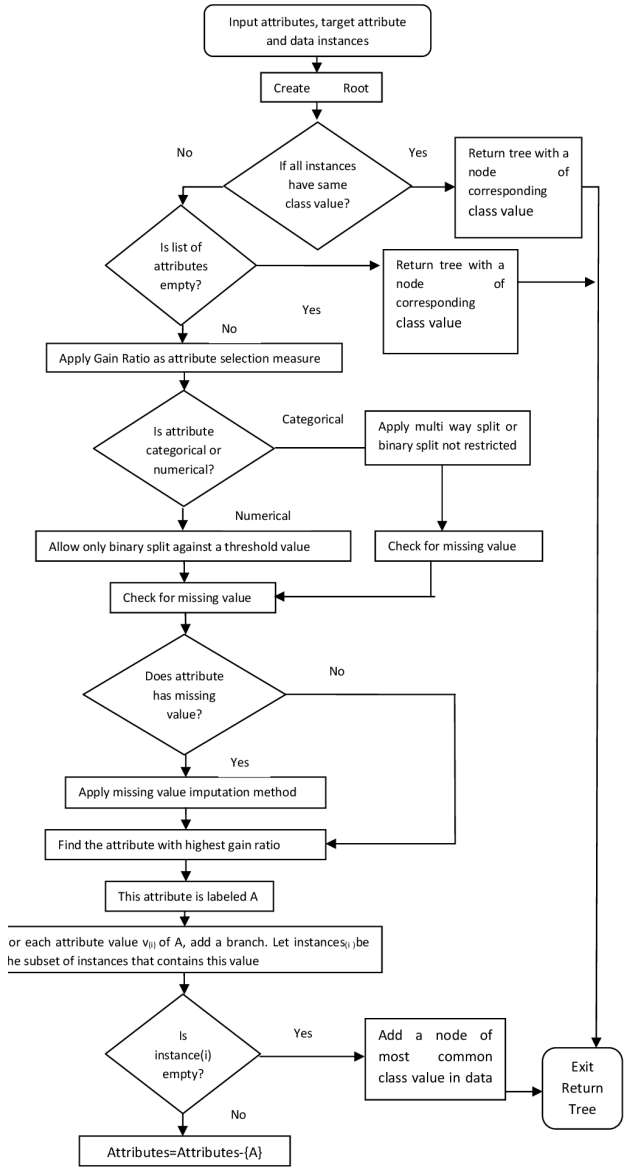
$E(S_{Ai})$ – subset of S with items that have A_i value

Missing Values are imputed
(Mean or Group Mean)

Gain Ratio is used for Feature Selection

Data is sorted to obtain numerical splits

$A \leq A_L$						$A > A_R$							
64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No



Gain Ratio

- Gain ratio is a modification of information gain that may reduce its bias
- Gain ratio takes into account the number and size of branches when choosing an attribute
  - Corrects information gain by taking into account the intrinsic information of a split into account
  - Intrinsic Information – Information of the class that is disregarded
- Gain ratio can also be problematic
  - May choose a variable just because its intrinsic information is very low
  - Fix this by choosing only variables whose values are greater than average gain across all variables

CART

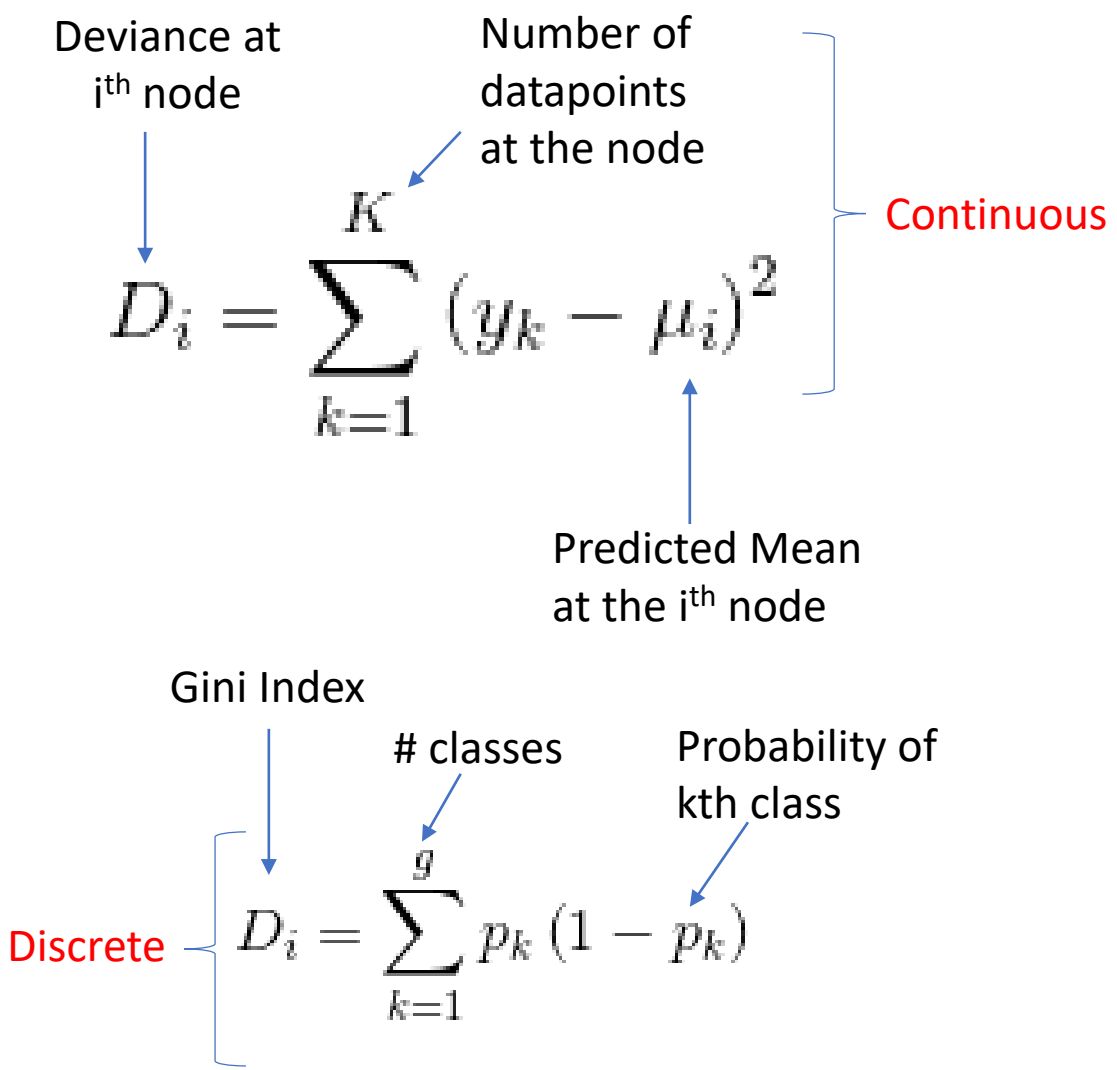
- Classification and Regression Trees (CART)
  - Proposed by Breiman et al., 1984
  - Default approach in Scikit learn for building a single tree
  - Very similar to C4.5 or C5.0 algorithm
  - Greedy search technique
    - Optimize at each node instead of overall optimization
  - Partition in a way that items in a group are as similar as they can be and groups are as distinct as they can be
  - The splitting is done such at – Go Left if condition is true otherwise go right
  - Missing value handing is automatic and occurs at every node
  - Can handle imbalanced datasets
    - Use of prior probabilities

CART Building Trees

- There are 4 basic steps to building classification and regression trees
  - Specifying the criteria for predictive accuracy
  - Selecting splits
  - Determining when to stop splitting
  - Selecting the "right-sized" tree
    - Pruning
- Different criteria are used for discrete (Classification) and continuous (Regression) problems

Accuracy

- Residual sum of squares error is used for continuous variable
- Gini Index is typically used for discrete variables
  - Entropy is also made use of sometimes



Splitting

- A split is done such that there is greatest predictive accuracy
- The algorithm is greedy
  - Local maximima at the node rather than global maxima
- For discrete variables the splits are tried at various classes
  - There has to be at least one class on either side of the split
- For continuous variable the data are ordered and splits are tried sequentially
  - There has to be at least one value on either side of the split
- Splits result in
  - Greatest homogeneity within each side of the split
  - Highest heterogeneity between the spilt

Decision to stop splitting

1) Till there are no more splits
2) Specified number of splits
3) Minimum number of samples

Complexity parameter (cp) is used to specify the growth of the tree
Smaller the complexity parameter the longer the tree

Tree Growth does not continue until it can improve the fit by a factor of Cp

Pruning of the Tree

- Cross-validation is used to compute the errors associated with each split
  - The data is split into k folds.
  - K-1 folds are used to build the model and the Kth fold is used to test
  - Mean error (xerror) and standard deviation (xstd) of the error of the Kth fold are determined
  - The process is repeated K times
- Select the tree that has a cp value or a size that corresponds to minimum xerror
  - Sometime an even smaller tree but within 1 std. error (1sd) of the minimum error is selected to further reduce the complexity

CART Example

- Fit the Ogallala MCL data using a decision tree
- Change the complexity parameter to evaluate how the tree complexity changes
- use (printcp/plotcp) command to find the tree with the optimal complexity
- Develop the pruned tree and plot it
- Evaluate the contingency table for this tree

```
# CART Model Predictions
b1 <- rpart(MCL ~ DWT + Clay + Recharge + Slope + SOM + SHG + Texture +
                                    Dist_MSW + Ndep + WellDepth + ORP + Mn + Fe + Ca + Nload + DO,
                  method = "class", data=dat.train, control = rpart.control(minsplit =3,cp = 0.0001))
```

RCode

```
# load libraries
library(rpart)
library(corrplot)
library(RWeka)
library(party)

#Read the data
a <- read.csv('Ogallaladata.csv')
head(a)
```

```
# Split the dataset into training and
testing
a.trn <- subset(a,Train=='Training')
a.tst <- subset(a,Train=='Testing')

# Explore the data
summary(a.trn)
summary(a.tst)
```

```
# Remove externeous variables
a.trn <- a.trn[,c(-1,-2,-3,-21,-22,-
23)]
a.tst <- a.tst[,c(-1,-2,-3,-21,-22,-
23)]

# Look at correlations
trn.cor <- cor(a.trn)
corrplot(trn.cor,type='upper',diag
=F)
```

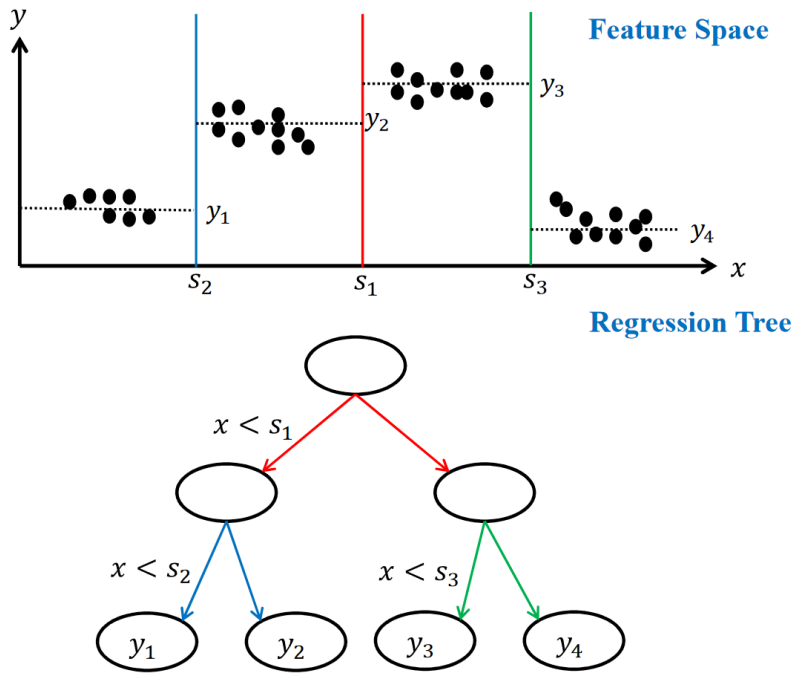
```
#Built the CART Tree
fit <- rpart(MCL ~., data =
a.trn)
plot(fit)
text(fit, use.n = TRUE)
```

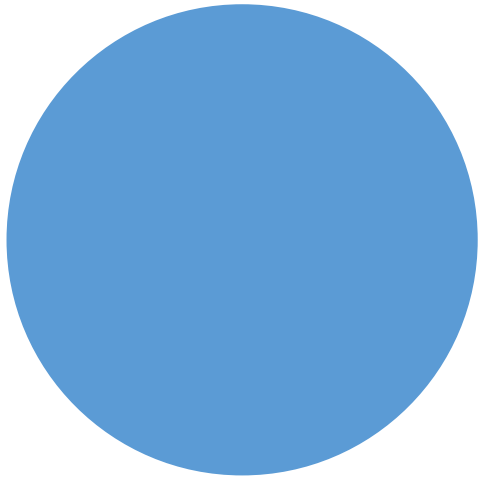
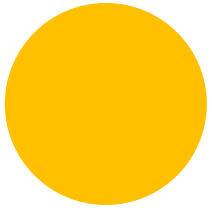
```
# Make predictions using CART
out <- predict(fit)
MCL.trn <- ifelse(out<=0.5,0,1)
MCL.obs <- a.trn$MCL
table(MCL.trn,MCL.obs)

# Make the plot
plot(fit)
```

Regression Trees

- Regression trees are constructed to predict continuous variables
  - Need a value instead of a classification (or probability of classification)
- The idea is similar to classification trees
  - Split the datasets as usual
  - Find a regression model at the leaf that has minimum variance
  - Can also be envisioned as a piecewise regression model
    - The regression parameters however change from node to node



Multi-Adaptive Regression Splies (MARS)

MultiAdaptive Regression Splines (MARS)

- Similar to CART
  - But largely used to fit continuous dependent variables
  - Introduced by Freidman et al., (1991)
- It is similar in spirit to piece-wise regression
  - But makes no assumption with regards to the underlyng functional form of the model
    - Input-output relationship is not global
    - Nonparametric method
- It is a greedy search algorithm
  - Locally fits a regression model
  - Divides the data space into several sub-regions
    - Each region has its own model

MARS

- Basis Functions are key to the formulation of MARS model

Coefficient

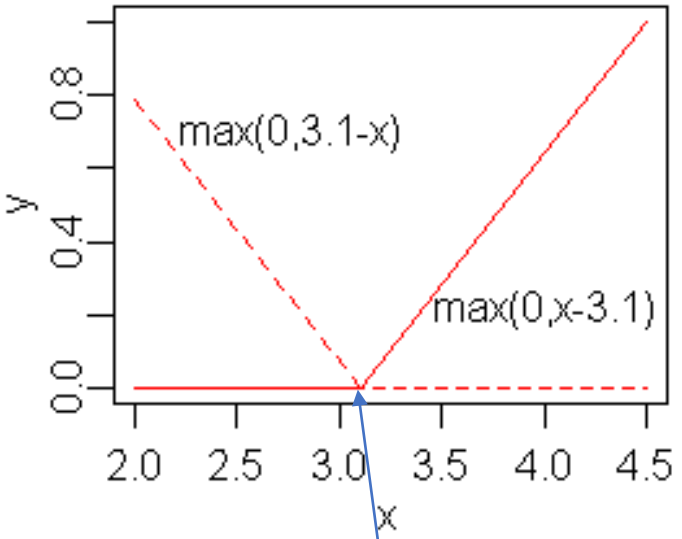
$$y = f(x) = \beta_o + \sum_{m=1}^{M} \beta_m h_m(X)$$

Intercept

Basis Function

Interactions are not shown but can be added

$$H_{(x,km)} = \prod_{k=1}^{K} h_{km}$$



Knot of a basis function

Knots are determined as part of the model calibration process

MARS

- Model is calibrated using Generalized Cross-Validation approach
- Algorithm Steps:
- Start with the simplest model
  - Involving only the constant basis function.
- Search the space of basis functions, for each variable and for all possible knots,
  - Add those which maximize a certain measure of goodness of fit (minimize prediction error).
- Step 2 is recursively applied
  - Until a model of pre-determined maximum complexity is derived.
- Finally, in the last stage, a pruning procedure is applied
  - Those basis functions are removed that contribute least to the overall (least squares) goodness of fit.

$$y = f(x) = \beta_o + \sum_{m=1}^{M} \beta_m h_m(X)$$

$$GCV = \frac{\sum_{i=1}^{N} (y_i - f(x_i))^2}{\left(1 - \frac{C}{N}\right)^2}$$

Regularization Term

Penalty for adding a basis function

$$C = 1 + cd$$

Ideally d is between 2 and 3
To get optimal C

Number of degrees of freedom
(number of basis functions)

MARS

- MARS is generally used for continuous dependent variables
  - Independent variables can be continuous or discrete
- MARS can also be used with discrete predictors
  - However the basis splits need not coincide with the classes
  - MARS is combined with GLM (generalized linear models) to predict dichotomous variables
- The term “MARS” is trademarked and licensed exclusively to Salford Systems http://www.salfordsystems.com.
  - We can use MARS as an abbreviation; however, it cannot be used for competing software solutions.
  - This is why the R package uses the name **earth**
  - Python has **py-earth** class in Sci-kit learn package
    - https://contrib.scikit-learn.org/py-earth/content.html

R Code

```
# CART model for MCL prediction
# MCL = 0 safe to drink; MCL=1 unsafe to drink
# Venki Uddameri, TTU 02/19/2020

# load libraries
library(rpart)
library(earth)
library(rpart.plot)
library(RColorBrewer)

# Set working directory
setwd('D:\\Dropbox\\000CE5333Machine Learning\\Classwork')

# Read the data
a <- read.csv('Ogallaladata.csv')
a.new <- a[,c(-1,-2,-3,-22,-23)] # remove externeous values

#Split the data into training and testing
train <- subset(a.new, Train == 'Training')
test <- subset(a.new,Train == 'Testing')
```

```
# Fit the CART model
cart.f <- rpart(MCL~.,data=train,control=c(cp=0.000001,minsplit=3))
cart.f
summary(cart.f)
plotcp(cart.f)
rpart.plot(cart.f)
train.mclp <- predict(cart.f)
train.MCL <- ifelse(train.mclp <=0.5,0,1)
table(train.MCL,train$MCL)

test.mclp <- predict(cart.f,test)
test.MCL <- ifelse(test.mclp <=0.5,0,1)
table(test.MCL,test$MCL)

cart.f1 <- rpart(MCL~Ca+DWT+WellDepth+SOM+Ndep,data=train,control=c(cp=0.2))
```

```
# Fit the earth (MARS) model
earth.model <- earth(MCL ~ ., glm =list(family=binomial),
data=train)
plot(earth.model)
earth.imp <- evimp(earth.model)
plot(earth.imp)
```

MARS

- The earth package in R provides functions to carryout MARS analysis

```
# MARS Model predictions
MCL <- train[,20]
dat.train <- cbind(MCL,inp)
earth.model <- earth(MCL ~ ., glm =list(family=binomial),
data=dat.train)
plot(earth.model)
earth.imp <- evimp(earth.model)
plot(earth.imp)
```

You Should Know

- Idea behind C4.5 and C5 algorithms
- Idea behind classification and regression trees (CART)
  - How are the variables selected at nodes
  - How does the split takes place
  - How does the tree grow (spit)
  - How to prune the tree
- Idea behind the multiadaptive Regression Splines (MARS)
  - Basis function
  - Generalized cross validation