IEOR E4650 Business Analytics

Session 10: K-Nearest Neighbors and Decision Trees

Spring 2018

Prof. Adam Elmachtoub

Parametric vs Non-parametric Methods

Parametric Models:

- Examples like linear and logistic regression
- Easy to fit
- Small number of interpretable parameters/coefficients
- Allow for extrapolation
- May force the data into a wrong form, leading to wrong conclusions

Non-parametric Methods

- Examples like K-NN and Decision Trees
- Do not require any assumption on the data, do not force the data into a particular form
- Are better at prediction than at describing the data, often offer no interpretation
- Require a lot of data to estimate, less efficient than a correct parametric model

K-Nearest Neighbors (KNN)

The K-Nearest Neighbors is a non-parametric method that can be used both for regression and classification. It uses a metric of closeness, defined on the vectors of covariates.

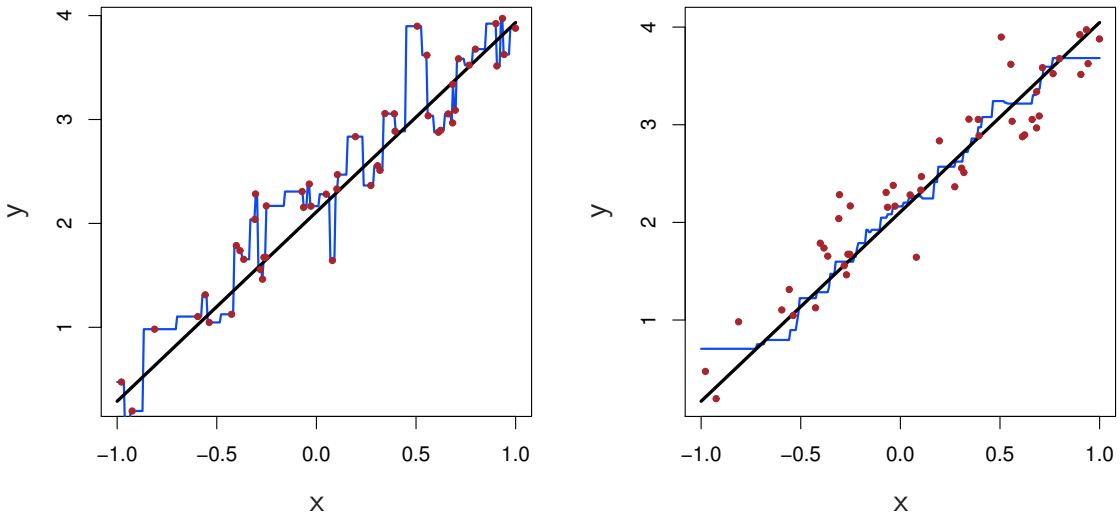
- Given a test observation x_0, identify the K points in the training data that are closest to x_0 by Euclidean distance.
- Let $\mathcal{N}(x_0)$ be the set of indexes of these K closest points.
- In the case of classification, estimate the conditional probability $P(Y = j|X = x_0)$ by

$$\frac{1}{K} \sum_{i \in \mathcal{N}(x_0)} I(y_i = j).$$

- In the case of regression, the estimated value of $E[Y|X = x_0]$ is

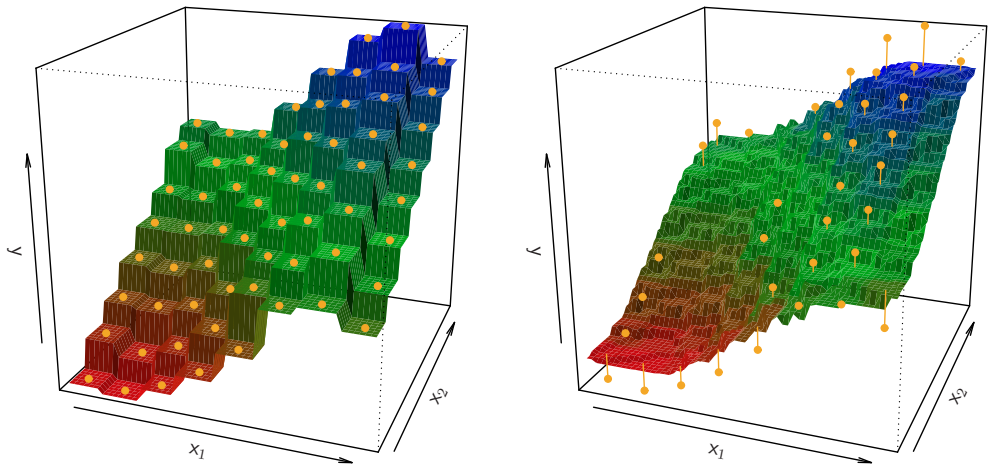
$$\frac{1}{K} \sum_{i \in \mathcal{N}(x_0)} y_i.$$

Linear Regression vs. K-Nearest Neighbors



The plot shows the KNN curve for one predictor X and 100 observations (x_i, y_i). The points were drawn by adding noise to the value of the solid line. The blue line on the left is the 1-NN estimated curve. The blue line on the right is the 9-NN estimated curve.

K-Nearest Neighbors with 2 predictors



The plot shows the KNN curve for two predictors X_1, X_2 and 64 observations (x_i, y_i). The curve on the left is the 1-NN estimated curve. The curve on the right is the 9-NN estimated curve.

Limitations of KNN

- Like all non-parametric methods, KNN won't be able to predict the outcome for x_0 well if there aren't any points close to it in our data
- KNN requires a distance measure such that if x_1 and x_2 are close y_1 and y_2 should be similar
- KNN does not perform well when we add irrelevant covariates
- The previous two comments are related to the idea that one could weight each feature like we described last time. Choosing all p weights is very hard and instead we typically normalize the data so the mean of each column X_i is 0 and the standard deviation is 1.
- Can use scale() command in R to normalize columns of features

K-Nearest Neighbors in R

The `knn()` function (in the `class` library) can be used to perform K-Nearest Neighbors classification. It simultaneously fits model and makes predictions.

The function `knn()` requires four inputs:

- A matrix containing the training data predictors
- A matrix containing the testing data predictors, for which we wish to make predictions
- A categorical vector of outcomes for the training observations
- A value for K, the number of nearest neighbors to be used by the classifier

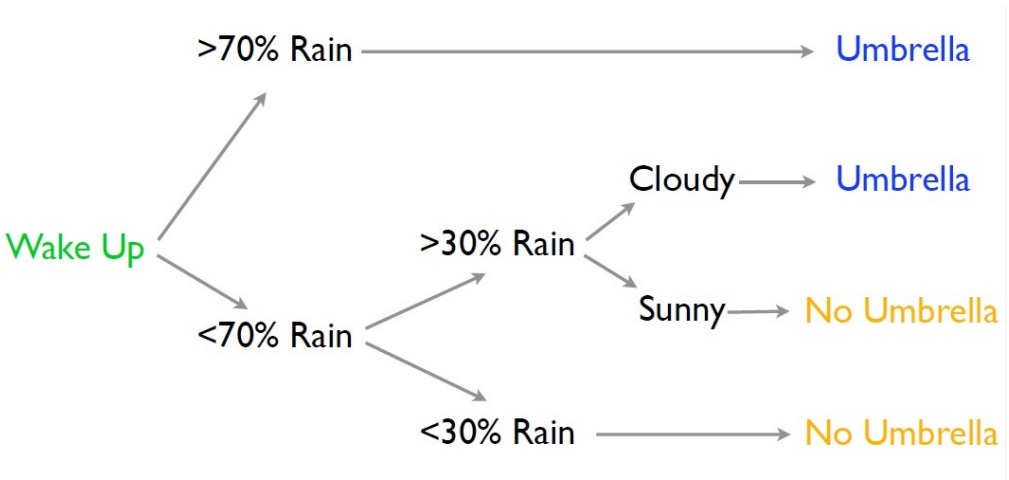
K-Nearest Neighbors: Example

- Sample Code

```
>library(class)
>xTrain = train[,-outcomeColNum]
>xTest = test[,-outcomeColNum]
>yTrain = train[,outcomeColNum]
>yTest = test[,outcomeColNum]
>KNNpred = knn(xTrain,xTest,yTrain, k = 5)
>print(mean(KNNpred == yTest))
[1] 0.877551
```

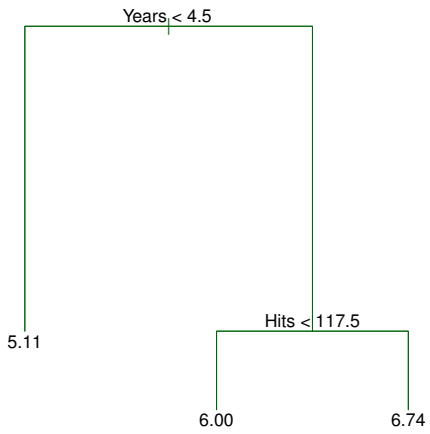
- Note that ties are broken randomly, making the results somewhat random.
- We can apply Model Selection ideas to chose the best k before doing Model Assessment

Decision Tree



Tree-logic uses a series of simple questions to come to a conclusion. Each question is a binary split, and final nodes, or leafs, correspond to an answer.

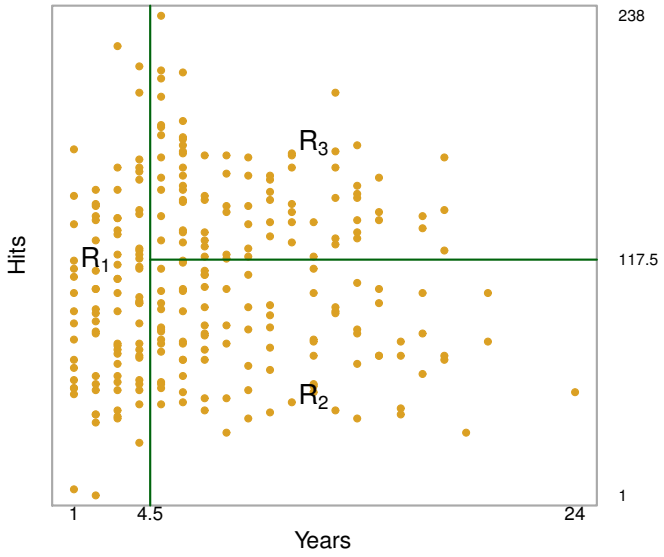
Decision Tree Partitions the Covariate Space



A simple tree to predict the *log-salary* of baseball players. The tree uses number of years played and number of hits made in the previous year. At any node, follow left if the condition is satisfied.

A Decision Tree for Regression

Below is a scatter plot of the data along the covariates.
These three regions can be written as $R_1 = \{X|\texttt{Years} < 4.5\}$,
$R_2 = \{X|\texttt{Years} >= 4.5, \texttt{Hits} < 117.5\}$,
$R_3 = \{X|\texttt{Years} >= 4.5, \texttt{Hits} >= 117.5\}$.



Decision Trees

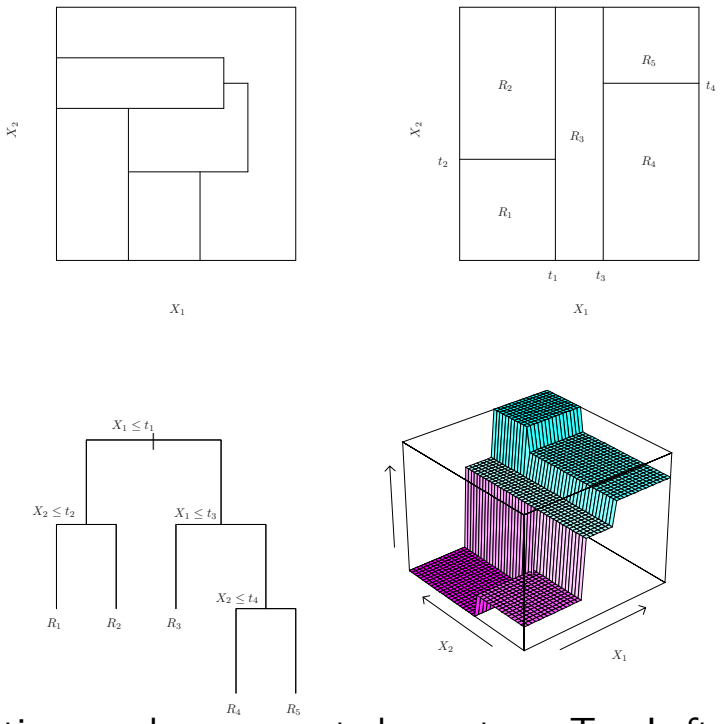
We can use decision trees for prediction, both for regression and classification. A tree is defined by its decision nodes and values for each leaf.

- The tree splits are on one feature X_j at a time
- For every set of features x_i, it maps to one specific leaf in the tree by following the rules starting at the root
- At the leaf, we simply predict using the training data belonging to the leaf.
  - Classification trees have class probabilities at each leaf (visually can show the best one)
  - Regression trees have mean response at each leaf

Decision Trees

- We can measure how well the decision tree fits the data using the same measures we used before:
  - RSS (MSE) (RMSE) for regression: $\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$
  - Deviance for classification: $-\sum_{i=1}^{n}\sum_{l=1}^{K} I(y_i = l)\log(\widehat{p}(y_i = l|x_i))$
- Selecting the best value to put at each leaf is simple
- How can we select the best trees?

Decision Tree for prediction



Not every partition can be represented as a tree. Top-Left: the true partition. Rest: representations of the estimated tree.

Classification Trees vs Linear Classification

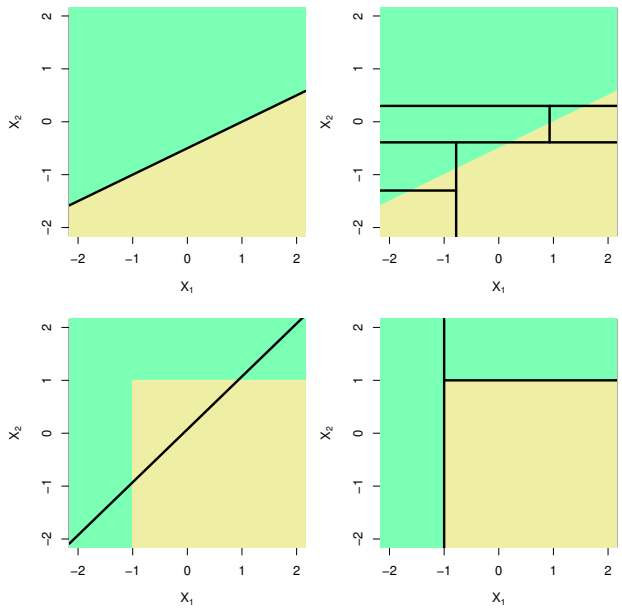


Figure: Classification using a linear regression model will do better in the top panel, whereas a regression tree will do better in the lower panel

Estimation of Regression Trees

To build a regression tree:

- Divide the predictor space, i.e., the set of possible values for $X_1, X_2, \ldots, X_p$ into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
- For every observation that falls into the region R_j, make the same prediction, which is simply the mean of the response values for the training observations in R_j.

The goal is to find boxes $R_1, R_2, \ldots, R_J$ that minimize the RSS, given by

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

$\widehat{y}_{R_j}$ is the mean response for the training observations within the j-th box.

Estimation of Classification Trees

To build a classification tree:

- Divide the predictor space, i.e., the set of possible values for $X_1, X_2, \ldots, X_p$ into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
- For every observation that falls into the region R_j, make the same prediction, which is simply the mean of the response values for the training observations in R_j.

The goal is to find boxes $R_1, R_2, \ldots, R_J$ that minimize the deviance, given by

$$-\sum_{j=1}^{J}\sum_{i\in R_j}\sum_{l=1}^{K} I(y_i = l)\log(\widehat{p}(Y = l|R_j)),$$

where $\widehat{p}(Y = l|R_j)$ is the fraction of points in R_j with label l

Finding the Best Decision Tree

- There is a *huge* set of possible tree configurations.
- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes.
- For this reason, we take a top-down, greedy approach that is known as recursive binary splitting.

Binary Splitting: CART algorithm

We can use a recursive (greedy) method:

- Split the data into two different decisions using a rule of the form $X_j < \gamma$.
- Take each new partition and split again.

Find the split in $\mathbf{X}$ that minimizes RSS or deviance.

- Find the predictor and the split (γ) that minimizes the RSS or deviance.

You then grow the tree at this point

- Each new child node contains a subset of the data.
- Each subset has its own prediction values

View each child as a new dataset, and try to grow again.

- Stop splitting when the number of observations in each leaf node is too small, or the improvement in RSS or deviance is small.

Use the tree library for CART in R

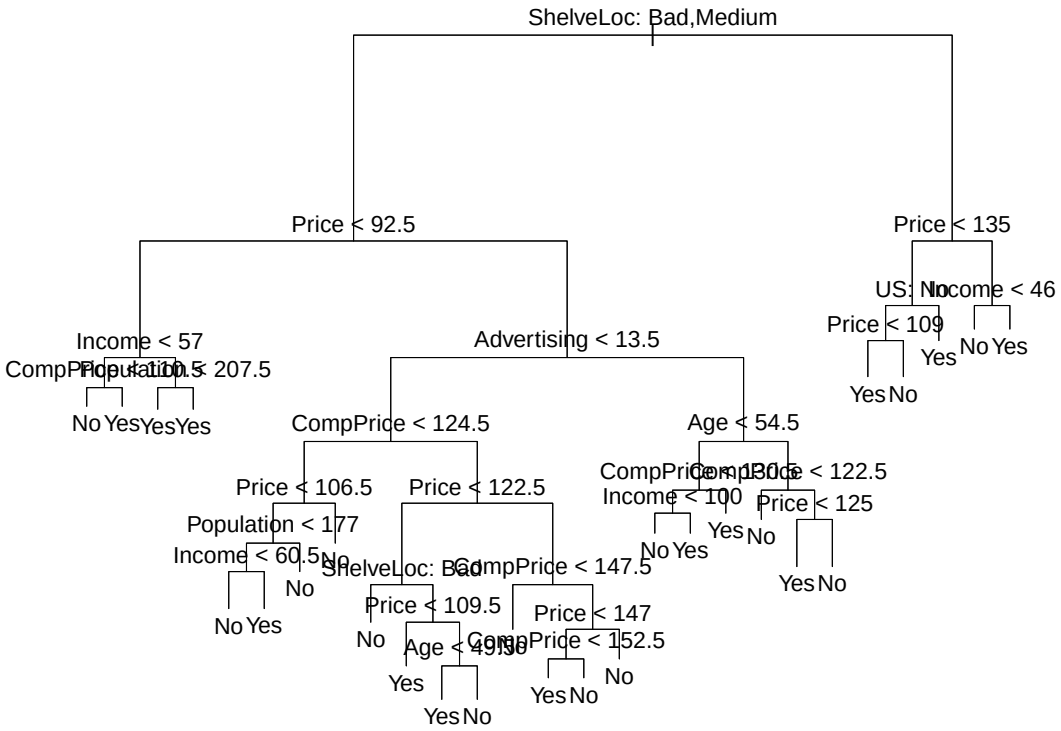
- The syntax is essentially the same as for `lm`:
  `mytree = tree(y ~ x1 + x2 + x3 ..., data=mydata)`
- Other arguments:
  - `mincut` is the minimum size for a new child.
  - `mindev` is the minimum (proportion) deviance improvement for proceeding with a new split.
- Defaults: `mincut=5, mindev=0.01`.
- As usual, you can print, summarize, and plot the tree.

R examples: Carseats data set

- Library `tree`
- Example with car seats data

```
>library(tree)
>library(ISLR)
>attach(Carseats)
>HighSales=ifelse(Sales <= 8, "No", "Yes")
>
>CarseatsData = data.frame(Carseats, HighSales)
>tree_carseats_HighSales = tree(HighSales~.-Sales, CarseatsData)
>summary(tree_carseats_HighSales)
>
>plot(tree_carseats_HighSales)
>text(tree_carseats_HighSales,pretty=0)
>tree_carseats_HighSales
```


Example: Carseats dataset



Cost Complexity Pruning

The CART method will find a tree that fits the training data, but the challenge is to avoid overfitting. We take an approach similar to LASSO and impose a cost for using more nodes. We choose a cost complexity value α, and find a tree that minimizes

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

- At $\alpha = 0$, we get a full tree. As α increases we prefer small trees. the tree is pruned in a predictable fashion.
- We start from a full tree, and prune to yield candidate trees. Prune by removing the splits that help the least for deviance reduction.
- Use cross validation (Model Selection) to select α!

Example: Carseats dataset

The function `cv.tree()` performs cross-valication in order to determine optimal level of tree

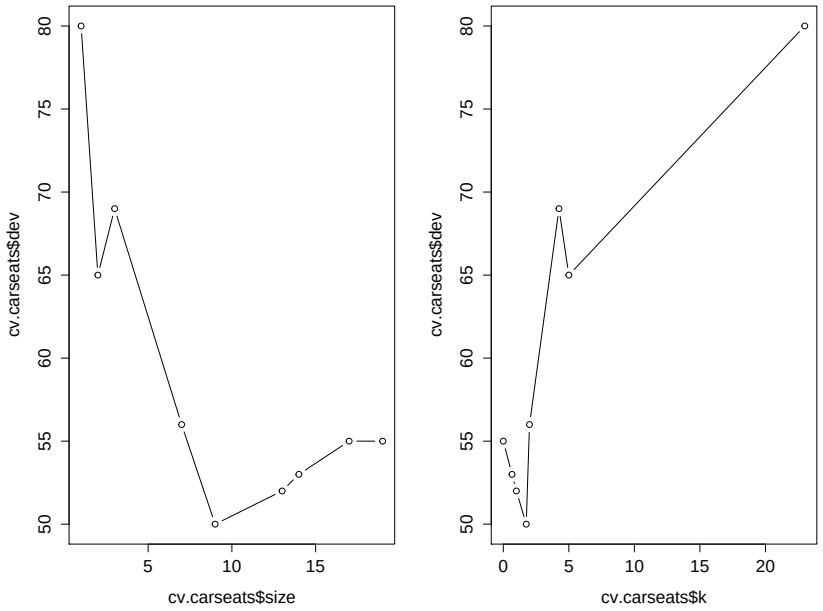
- Argument `FUN=prune.misclass` indicates that the classification error rate will guide cross-validation and pruning (default is deviance).
- In R the α corresponds to "k"

Cross validation on car seats data:

```
> set.seed(3)
> cv.carseats = cv.tree(tree_carseats_HighSales, FUN=prune.misclass)
> names(cv.carseats)
[1] "size"   "dev"    "k"      "method"
> par(mfrow=c(1,2))
> plot(cv.carseats$size,cv.carseats$dev,type="b")
> plot(cv.carseats$k,cv.carseats$dev,type="b")
> prune.carseats = prune.misclass(tree_carseats_HighSales,best=9)
> plot(prune.carseats)
> text(prune.carseats,pretty=0)
```

Example: Carseats dataset - Pruned Tree

The tree with 9 terminal nodes results in the lowest error rate.



Example: Carseats dataset - Pruned Tree

The tree with 9 terminal nodes results in the lowest error rate.