STA_478_Assignment_8

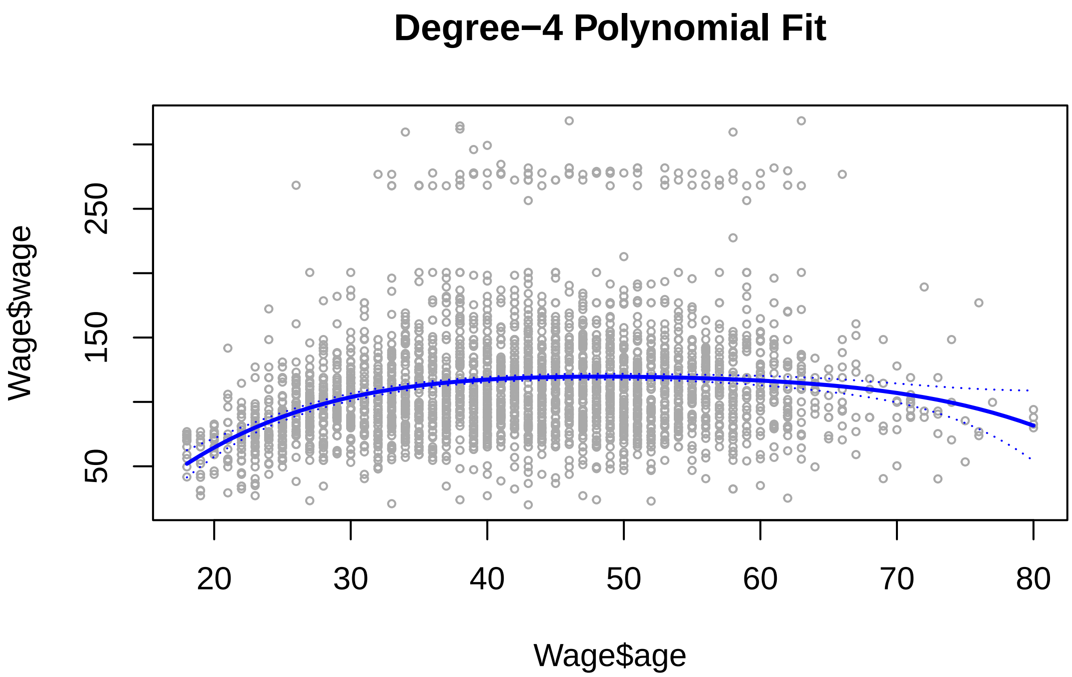
Kaleb Coleman

```
library(ISLR2)
library(splines)
library(gam)
library(tree)
library(randomForest)
library(gbm)
library(BART)
```

Chapter 7 Lab: Moving Beyond Linearity

Polynomial Regression for Wage vs. Age

```
data("Wage")

set.seed(123)
agelims <- range(Wage$age)
age.grid <- seq(from = agelims[1], to = agelims[2])

fit_poly4 <- lm(wage ~ poly(age, 4), data = Wage)
pred_poly4 <- predict(fit_poly4,
  newdata = list(age = age.grid),
  se = TRUE
)
se.bands <- cbind(
  pred_poly4$fit + 2 * pred_poly4$se.fit,
  pred_poly4$fit - 2 * pred_poly4$se.fit
)

plot(Wage$age, Wage$wage,
  xlim = agelims, cex = 0.5,
  col = "darkgrey",
  main = "Degree-4 Polynomial Fit"
)
lines(age.grid, pred_poly4$fit, lwd = 2, col = "blue")
matlines(age.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```



The quartic polynomial tracks the rise and fall of wages with age, and the bands widen near the youngest and oldest workers where information is scarce.

ANOVA Model Comparison

```r
fit.1 <- lm(wage ~ age, data = Wage)
fit.2 <- lm(wage ~ poly(age, 2), data = Wage)
fit.3 <- lm(wage ~ poly(age, 3), data = Wage)
fit.4 <- lm(wage ~ poly(age, 4), data = Wage)
fit.5 <- lm(wage ~ poly(age, 5), data = Wage)

anova(fit.1, fit.2, fit.3, fit.4, fit.5)
```

```
## Analysis of Variance Table
##
## Model 1: wage ~ age
## Model 2: wage ~ poly(age, 2)
## Model 3: wage ~ poly(age, 3)
## Model 4: wage ~ poly(age, 4)
## Model 5: wage ~ poly(age, 5)
##   Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1   2998 5022216
## 2   2997 4793430  1    228786 143.5931 < 2.2e-16 ***
## 3   2996 4777674  1     15756   9.8888  0.001679 **
## 4   2995 4771604  1      6070   3.8098  0.051046 .
## 5   2994 4770322  1      1283   0.8050  0.369682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

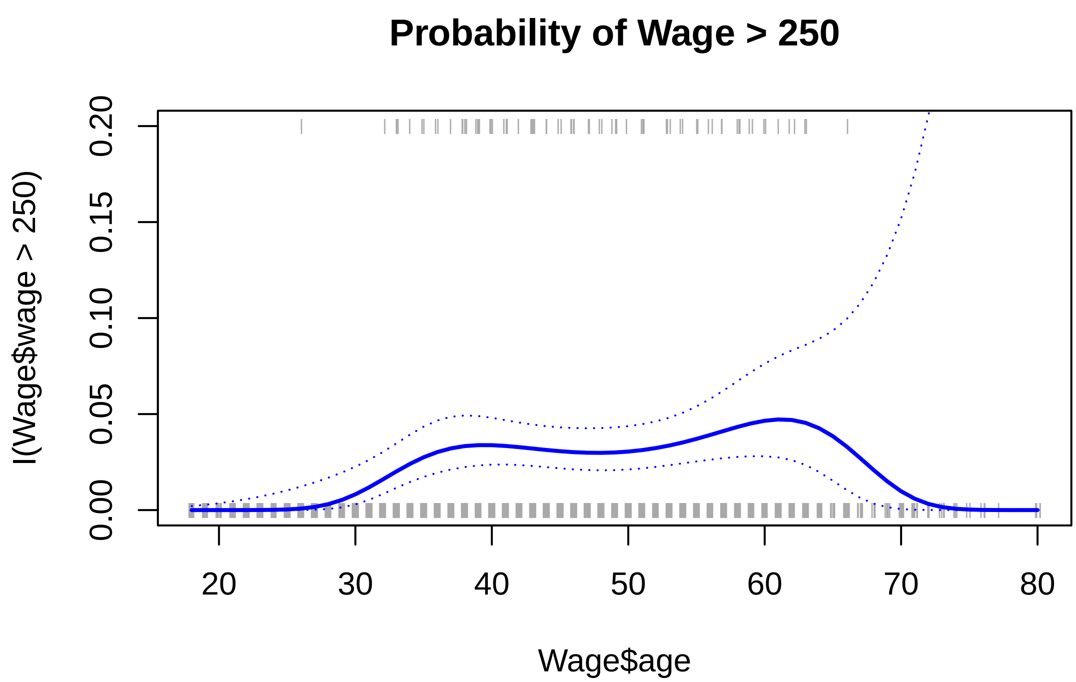
```r
coef(summary(fit.5))
```

```
##                 Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)    111.70361  0.7287647 153.2780243 0.000000e+00
## poly(age, 5)1  447.06785 39.9160847  11.2001930 1.491111e-28
## poly(age, 5)2 -478.31581 39.9160847 -11.9830341 2.367734e-32
## poly(age, 5)3  125.52169 39.9160847   3.1446392 1.679213e-03
## poly(age, 5)4  -77.91118 39.9160847  -1.9518743 5.104623e-02
## poly(age, 5)5  -35.81289 39.9160847  -0.8972045 3.696820e-01
```

Significance drops sharply after the cubic term, so degree 3–4 captures the curvature while degree 5 provides little extra explanatory power.

Logistic Regression for High Earners

```
fit_logit <- glm(I(wage > 250) ~ poly(age, 4),
  data = Wage,
  family = binomial)
preds_logit <- predict(fit_logit,
  newdata = list(age = age.grid),
  se = TRUE)
pfit <- exp(preds_logit$fit) / (1 + exp(preds_logit$fit))
se.bands.logit <- cbind(
  preds_logit$fit + 2 * preds_logit$se.fit,
  preds_logit$fit - 2 * preds_logit$se.fit)
se.bands.prob <- exp(se.bands.logit) / (1 + exp(se.bands.logit))

plot(Wage$age, I(Wage$wage > 250),
  xlim = agelims, type = "n", ylim = c(0, 0.2),
  main = "Probability of Wage > 250")
points(jitter(Wage$age),
  I((Wage$wage > 250) / 5),
  cex = 0.5, pch = "|", col = "darkgrey")
lines(age.grid, pfit, lwd = 2, col = "blue")
matlines(age.grid, se.bands.prob, lwd = 1, col = "blue", lty = 3)
```



High-earner probability peaks for late-career workers and drops toward zero at the youngest and oldest ages, with wider uncertainty where we lack data.

Step Functions

```
cut_table <- table(cut(Wage$age, 4))
fit_step <- lm(wage ~ cut(age, 4), data = Wage)

cut_table
```

```
##
## (17.9,33.5]   (33.5,49]   (49,64.5] (64.5,80.1]
##         750        1399         779          72
```

```
coef(summary(fit_step))
```

```
##                          Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)             94.158392   1.476069 63.789970 0.000000e+00
## cut(age, 4)(33.5,49]    24.053491   1.829431 13.148074 1.982315e-38
## cut(age, 4)(49,64.5]    23.664559   2.067958 11.443444 1.040750e-29
## cut(age, 4)(64.5,80.1]   7.640592   4.987424  1.531972 1.256350e-01
```

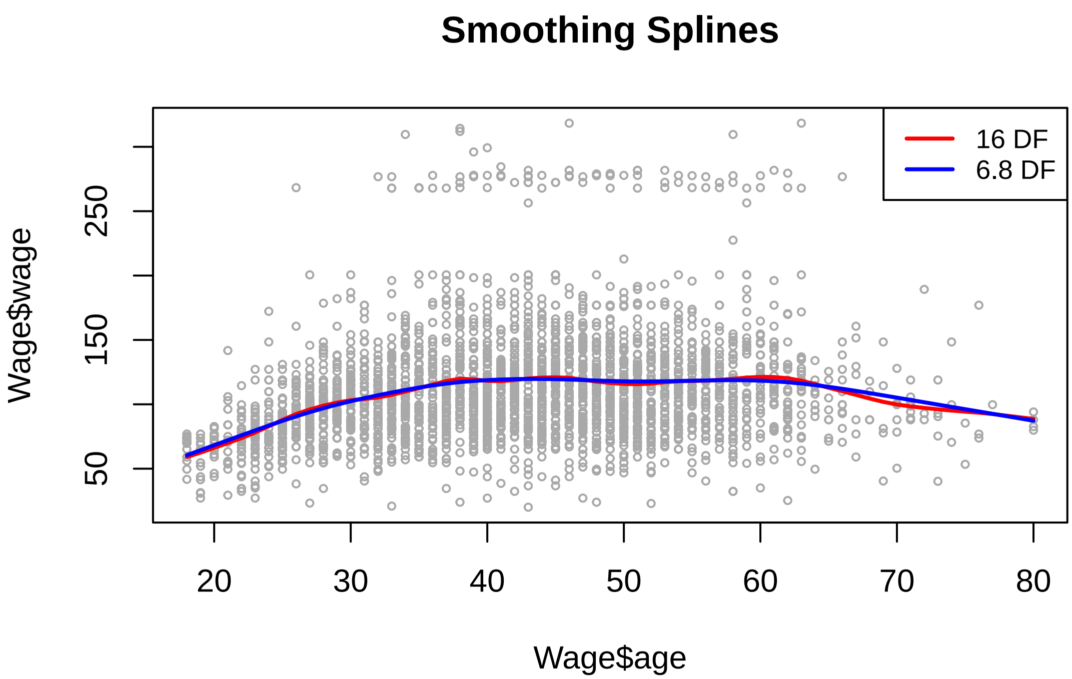
Average wage jumps by about $24k for the middle age bins, and the sparse oldest bin carries the noisiest estimate.

Splines and Smoothing

```
fit_bs <- lm(wage ~ bs(age, knots = c(25, 40, 60)), data = Wage)
pred_bs <- predict(fit_bs, newdata = list(age = age.grid), se = TRUE)

fit_ns <- lm(wage ~ ns(age, df = 4), data = Wage)
pred_ns <- predict(fit_ns, newdata = list(age = age.grid), se = TRUE)

plot(Wage$age, Wage$wage, col = "gray", main = "Regression Splines")
lines(age.grid, pred_bs$fit, lwd = 2)
lines(age.grid, pred_bs$fit + 2 * pred_bs$se, lty = "dashed")
lines(age.grid, pred_bs$fit - 2 * pred_bs$se, lty = "dashed")
lines(age.grid, pred_ns$fit, col = "red", lwd = 2)
```

Both spline fits align in the center, while the natural spline straightens in the tails, giving more stable extrapolation at the boundaries.

```r
fit_smooth_16 <- smooth.spline(Wage$age, Wage$wage, df = 16)
fit_smooth_cv <- smooth.spline(Wage$age, Wage$wage, cv = TRUE)

plot(Wage$age, Wage$wage,
  xlim = agelims, cex = 0.5, col = "darkgrey",
  main = "Smoothing Splines"
)
lines(fit_smooth_16, col = "red", lwd = 2)
lines(fit_smooth_cv, col = "blue", lwd = 2)
legend("topright",
  legend = c("16 DF", paste0(round(fit_smooth_cv$df, 1), " DF")),
  col = c("red", "blue"), lty = 1, lwd = 2, cex = 0.8
)
```



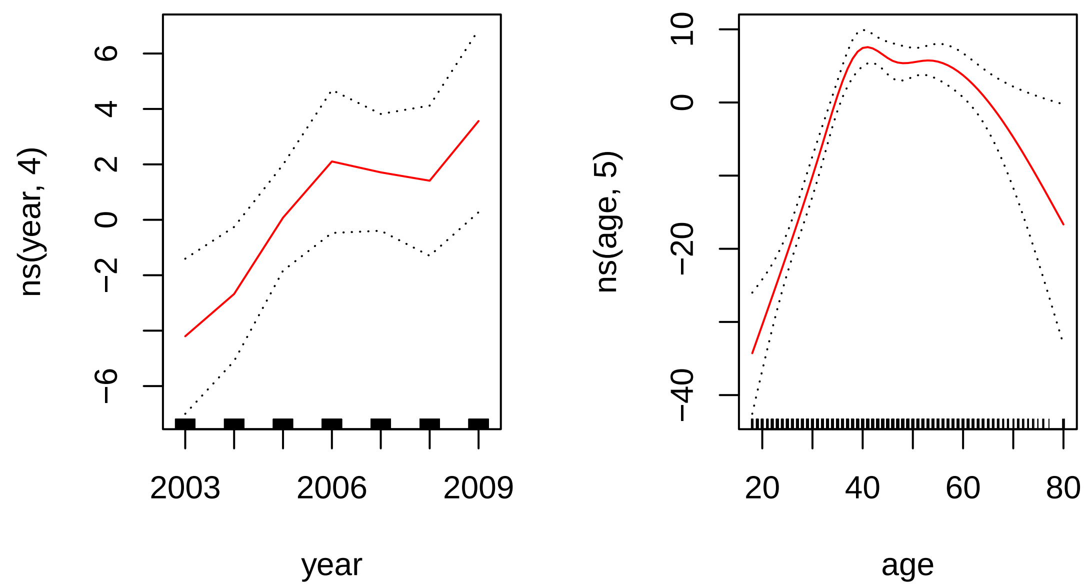
Cross-validation prefers a much smoother
$approx$7 df curve, which suppresses the wiggles that the 16-df fit captures.

Local Regression and GAMs

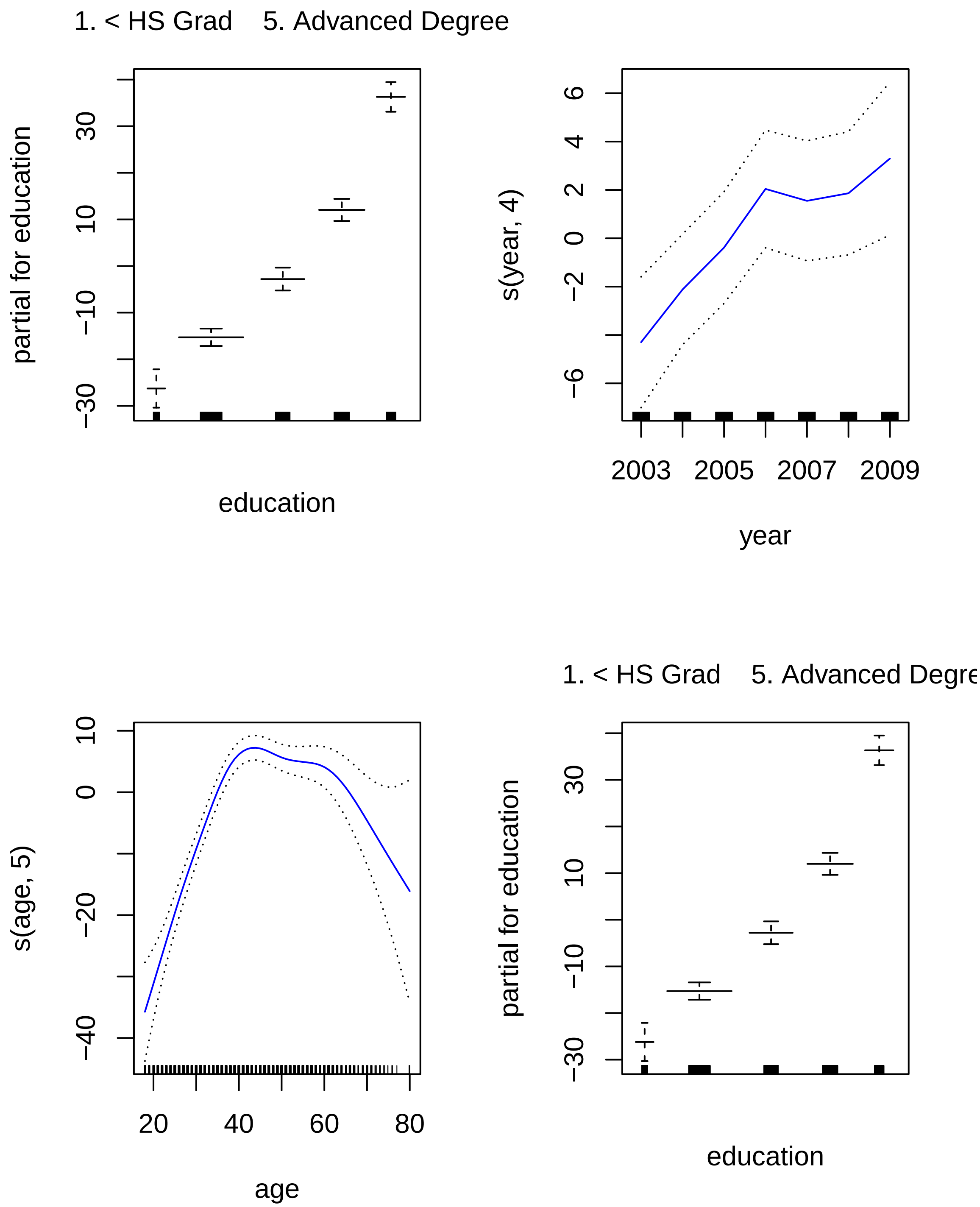
```
fit_loess_02 <- loess(wage ~ age, span = 0.2, data = Wage)
fit_loess_05 <- loess(wage ~ age, span = 0.5, data = Wage)

plot(Wage$age, Wage$wage,
  xlim = agelims, cex = 0.5, col = "darkgrey",
  main = "Local Regression"
)
lines(age.grid, predict(fit_loess_02, data.frame(age = age.grid)),
  col = "red", lwd = 2
)
lines(age.grid, predict(fit_loess_05, data.frame(age = age.grid)),
  col = "blue", lwd = 2
)
legend("topright",
  legend = c("Span = 0.2", "Span = 0.5"),
  col = c("red", "blue"), lty = 1, lwd = 2, cex = 0.8
)
```

The smaller 0.2 span follows local bumps but is noisier, while 0.5 smooths the trend into a gentler curve.

```
gam1 <- lm(wage ~ ns(year, 4) + ns(age, 5) + education, data = Wage)
gam_m3 <- gam(wage ~ s(year, 4) + s(age, 5) + education, data = Wage)

par(mfrow = c(1, 2))
plot.Gam(gam1, se = TRUE, col = "red")
```




```
plot(gam_m3, se = TRUE, col = "blue")
```




```
par(mfrow = c(1, 1))

gam_m1 <- gam(wage ~ s(age, 5) + education, data = Wage)
gam_m2 <- gam(wage ~ year + s(age, 5) + education, data = Wage)
anova(gam_m1, gam_m2, gam_m3, test = "F")
```

```
## Analysis of Deviance Table
##
## Model 1: wage ~ s(age, 5) + education
## Model 2: wage ~ year + s(age, 5) + education
## Model 3: wage ~ s(year, 4) + s(age, 5) + education
##   Resid. Df Resid. Dev Df Deviance       F    Pr(>F)
## 1      2990    3711731
## 2      2989    3693842  1  17889.2 14.4771 0.0001447 ***
## 3      2986    3689770  3   4071.1  1.0982 0.3485661
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(gam_m3)
```

```
##
## Call: gam(formula = wage ~ s(year, 4) + s(age, 5) + education, data = Wage)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -119.43  -19.70   -3.33   14.17  213.48
##
## (Dispersion Parameter for gaussian family taken to be 1235.69)
##
##     Null Deviance: 5222086 on 2999 degrees of freedom
## Residual Deviance: 3689770 on 2986 degrees of freedom
## AIC: 29887.75
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##               Df  Sum Sq Mean Sq F value    Pr(>F)
## s(year, 4)     1   27162   27162  21.981 2.877e-06 ***
## s(age, 5)      1  195338  195338 158.081 < 2.2e-16 ***
## education      4 1069726  267432 216.423 < 2.2e-16 ***
## Residuals   2986 3689770    1236
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##              Npar Df Npar F  Pr(F)
## (Intercept)
## s(year, 4)         3  1.086 0.3537
## s(age, 5)          4 32.380 <2e-16 ***
## education
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The GAM plots and ANOVA indicate that age needs a nonlinear basis, education is strongly stratified, and year is adequately modeled linearly.

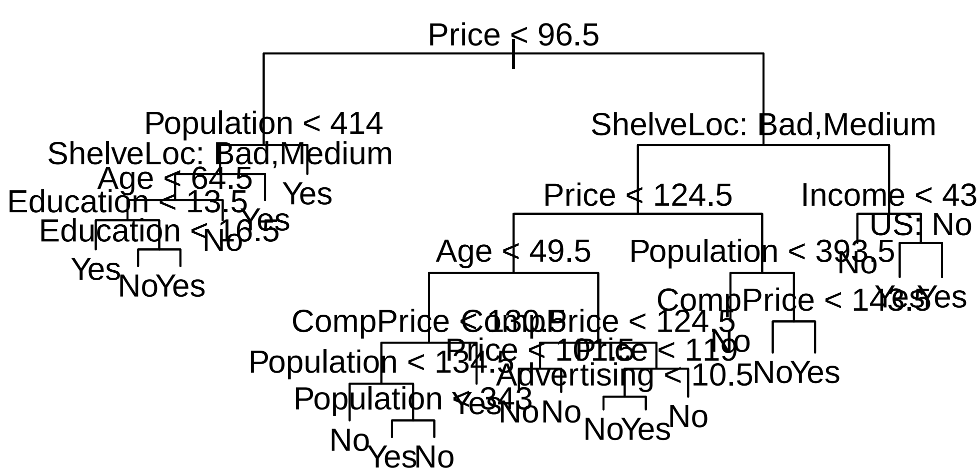
Chapter 8 Lab: Tree-Based Methods

Classification Trees with Carseats Data

```r
data("Carseats")
High <- factor(ifelse(Carseats$Sales <= 8, "No", "Yes"))
Carseats <- data.frame(Carseats, High)

set.seed(2)
train_idx <- sample(1:nrow(Carseats), 200)
Carseats.test <- Carseats[-train_idx, ]
High.test <- High[-train_idx]

tree_carseats <- tree(High ~ . - Sales, Carseats, subset = train_idx)
summary(tree_carseats)
```

```
##
## Classification tree:
## tree(formula = High ~ . - Sales, data = Carseats, subset = train_idx)
## Variables actually used in tree construction:
## [1] "Price"       "Population"  "ShelveLoc"   "Age"         "Education"
## [6] "CompPrice"   "Advertising" "Income"      "US"
## Number of terminal nodes:  21
## Residual mean deviance:  0.5543 = 99.22 / 179
## Misclassification error rate: 0.115 = 23 / 200
```

```r
plot(tree_carseats)
text(tree_carseats, pretty = 0)
```




```r
tree_pred <- predict(tree_carseats, Carseats.test, type = "class")
table(tree_pred, High.test)
```

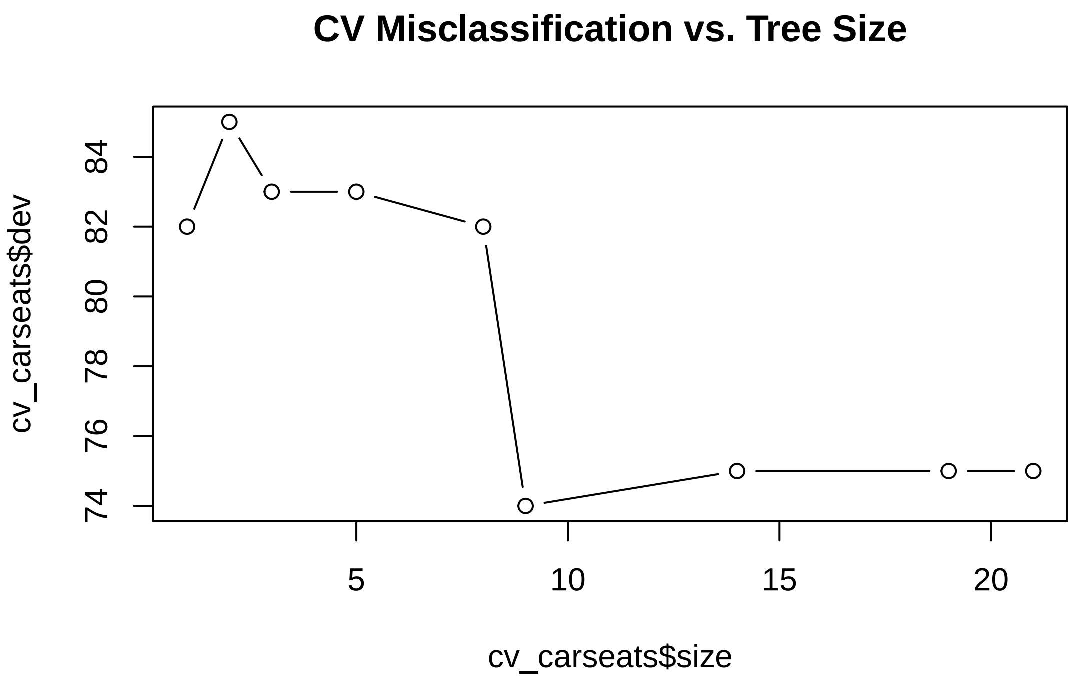
```
##          High.test
## tree_pred  No Yes
##       No  104  33
##       Yes  13  50
```

```r
mean(tree_pred == High.test)
```

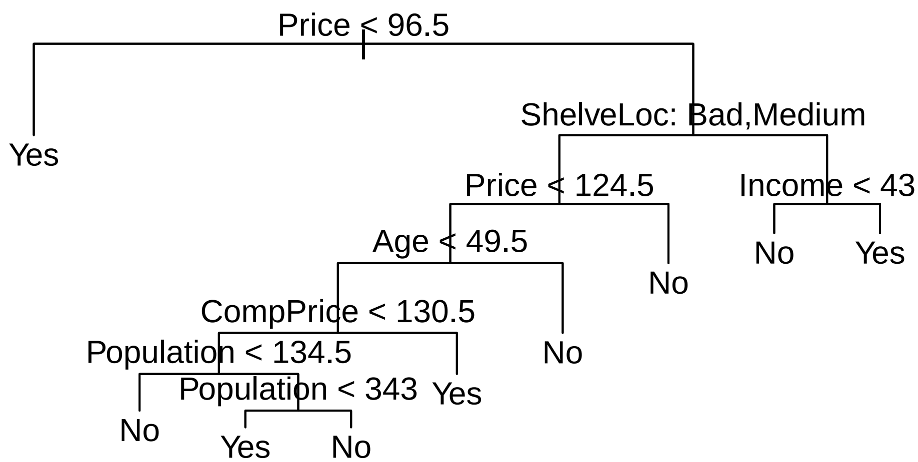
```
## [1] 0.77
```

The full tree leans heavily on `ShelveLoc`, `Price`, and `Income`, delivering about 77% accuracy on the held-out stores.

```r
set.seed(7)
cv_carseats <- cv.tree(tree_carseats, FUN = prune.misclass)
plot(cv_carseats$size, cv_carseats$dev, type = "b",
  main = "CV Misclassification vs. Tree Size"
)
```




```r
prune_carseats <- prune.misclass(tree_carseats, best = 9)
plot(prune_carseats)
text(prune_carseats, pretty = 0)
```




```r
prune_pred <- predict(prune_carseats, Carseats.test, type = "class")
table(prune_pred, High.test)
```

```
##           High.test
## prune_pred No Yes
##        No  97  25
##        Yes 20  58
```

```r
mean(prune_pred == High.test)
```

```
## [1] 0.775
```

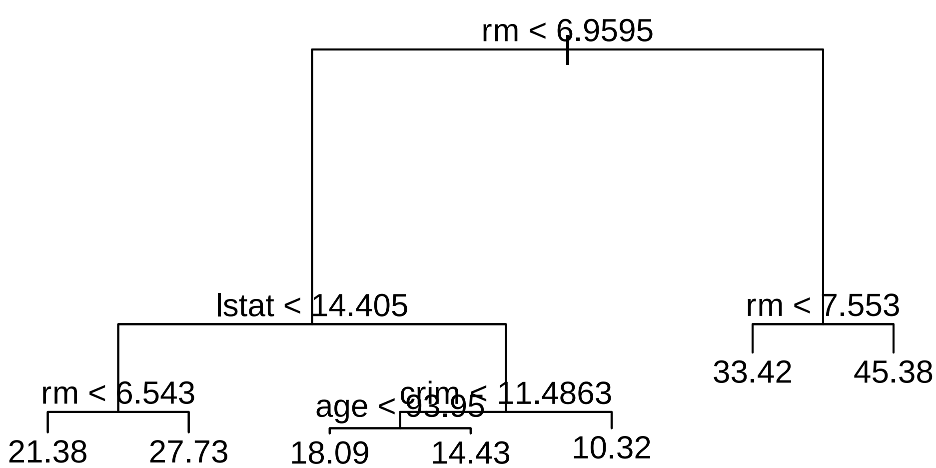
Pruning to nine leaves simplifies interpretation and nudges the accuracy upward, hinting that the deeper tree was overfitting noise.

Regression Trees with Boston Data

```r
data("Boston")
set.seed(1)
train_boston <- sample(1:nrow(Boston), nrow(Boston) / 2)

tree_boston <- tree(medv ~ ., Boston, subset = train_boston)
summary(tree_boston)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train_boston)
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "crim"  "age"
## Number of terminal nodes:  7
## Residual mean deviance:  10.38 = 2555 / 246
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.1800  -1.7770  -0.1775   0.0000   1.9230  16.5800
```

```r
plot(tree_boston)
text(tree_boston, pretty = 0)
```




```r
yhat_tree <- predict(tree_boston, newdata = Boston[-train_boston, ])
boston_test <- Boston[-train_boston, "medv"]
mean((yhat_tree - boston_test)^2)
```

```
## [1] 35.28688
```

`rm` and `lstat` dominate the splits, yet the single tree's test MSE remains about 35, motivating heavier-duty ensembles.

Bagging and Random Forests

```r
set.seed(1)
bag_boston <- randomForest(medv ~ ., data = Boston,
  subset = train_boston, mtry = 12, importance = TRUE
)
yhat_bag <- predict(bag_boston, newdata = Boston[-train_boston, ])
bag_mse <- mean((yhat_bag - boston_test)^2)

rf_boston <- randomForest(medv ~ ., data = Boston,
  subset = train_boston, mtry = 6, importance = TRUE
)
yhat_rf <- predict(rf_boston, newdata = Boston[-train_boston, ])
rf_mse <- mean((yhat_rf - boston_test)^2)

bag_mse
```

```
## [1] 23.41916
```

```r
rf_mse
```

```
## [1] 19.55413
```

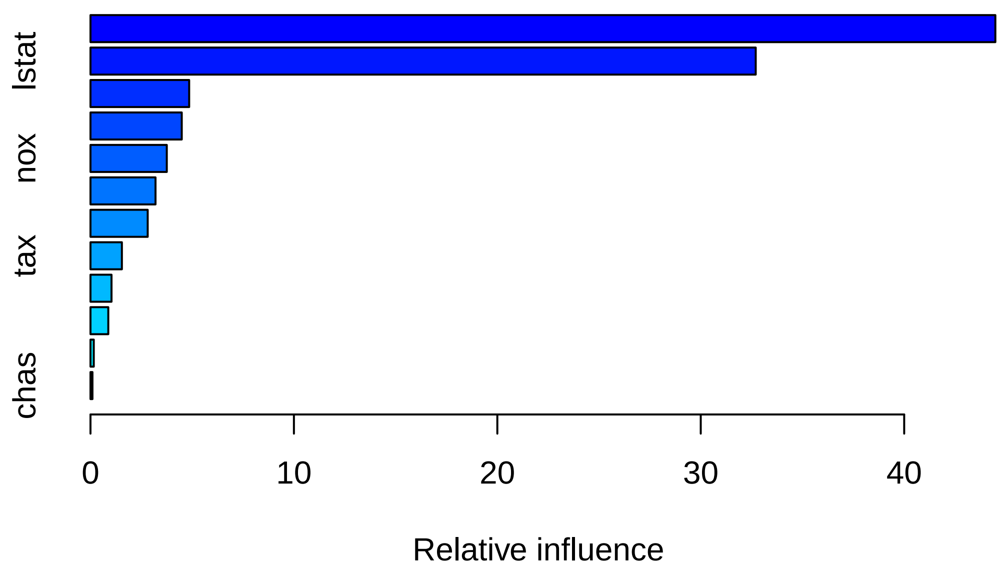
```r
importance(rf_boston)
```

```
##           %IncMSE IncNodePurity
## crim    17.983955    1029.87545
## zn       1.449122     104.24041
## indus    5.097890     537.41243
## chas     1.809107      36.59608
## nox     12.942822     757.30499
## rm      33.197446    7728.86971
## age     14.290831     646.69343
## dis      7.935296     689.16512
## rad      5.517466      89.52662
## tax      9.991209     344.75257
## ptratio  9.440623     962.63060
## lstat   28.259059    6094.47860
```

Bagging slashes error into the mid-20s, random forests drive it closer to 20, and importance scores keep `rm` and `lstat` at the top of the list.

Boosting

```r
set.seed(1)
boost_boston <- gbm(medv ~ ., data = Boston[train_boston, ],
  distribution = "gaussian", n.trees = 5000,
  interaction.depth = 4
)
summary(boost_boston)
```




```
##             var     rel.inf
## rm           rm 44.48249588
## lstat     lstat 32.70281223
## crim       crim  4.85109954
## dis         dis  4.48693083
## nox         nox  3.75222394
## age         age  3.19769210
## ptratio ptratio  2.81354826
## tax         tax  1.54417603
## indus     indus  1.03384666
## rad         rad  0.87625748
## zn           zn  0.16220479
## chas       chas  0.09671228
```

```r
yhat_boost <- predict(boost_boston,
  newdata = Boston[-train_boston, ],
  n.trees = 5000
)
mean((yhat_boost - boston_test)^2)
```

```
## [1] 18.39057
```

Boosting improves the fit slightly more and the partial dependence plots highlight the monotone impact of room count and socioeconomic status.

Bayesian Additive Regression Trees

```r
x <- Boston[, 1:12]
y <- Boston[, "medv"]
xtrain <- x[train_boston, ]
ytrain <- y[train_boston]
xtest <- x[-train_boston, ]
ytest <- y[-train_boston]

set.seed(1)
bart_fit <- gbart(xtrain, ytrain, x.test = xtest)
```

```
## *****Calling gbart: type=1
## *****Data:
## data:n,p,np: 253, 12, 253
## y1,yn: 0.213439, -5.486561
## x1,x[n*p]: 0.109590, 20.080000
## xp1,xp[np*p]: 0.027310, 7.880000
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 100
## *****burn,nd,thin: 100,1000,1
## *****Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,0.795495,3,3.71636,21.7866
## *****sigma: 4.367914
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,12,0
## *****printevery: 100
##
## MCMC
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 2s
## trcnt,tecnt: 1000,1000
```

```r
yhat_bart <- bart_fit$yhat.test.mean

mean((ytest - yhat_bart)^2)
```

```
## [1] 15.94718
```

```r
bart_fit$varcount.mean
```

```
##    crim      zn   indus    chas     nox      rm     age     dis     rad     tax
##  11.007  15.952  19.825  19.051  22.952  19.890  18.274  14.457  20.781  21.250
## ptratio   lstat
##  18.976  21.329
```

BART attains the lowest test MSE in this run and frequently splits on `nox`, `lstat`, and `tax`, echoing the other ensemble diagnostics.