Linear Regression Model: $Y = X\beta + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ **OLS**: $\hat{\beta} = (X^\top X)^{-1} X^\top Y$ **Var**: $\widehat{\mathrm{Var}}(\hat{\beta}) = \hat{\sigma}^2 (X^\top X)^{-1}$, $\hat{\sigma}^2 = \frac{\|Y - X\hat{\beta}\|^2}{n-p}$, $R^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}}$ **SE**: Small SE $\Rightarrow \hat{\beta}$ stable, across different data the coefficient estimate won't fluctuate much. Large SE $\Rightarrow \hat{\beta}$ noisy, unstable estimate and you can't know if predictor really matters. SE of $\hat{\beta}_j$ coefficients is relevant for T-tests and CI's.
LS Choice if variance of errors not constant (heteroscedasticity, each obs. i has own variance σ_i^2) $\Rightarrow$ WLS (give each point a weight $w_i = 1/\sigma_i^2$. if errors are correlated (dependence across observations, e.g. in time series) $\Rightarrow$ GLS
Model Complexity As number of predictor variables (q) increases, the model becomes more flexible and complex, allowing it to capture relationships better and decrease bias. However, increasing complexity leads to model being more sensitive to the specific training data that was used. $\uparrow$ q $\Rightarrow \downarrow$ Bias $\uparrow$ Var
Bias-Variance $E[(Y - \hat{f}(x))^2] = \sigma^2 + \mathrm{Bias}^2 + \mathrm{Var}$; $\uparrow$ df $\Rightarrow \downarrow$ Bias $\uparrow$ Var ; $\uparrow$ Smoothing $\Rightarrow \uparrow$ Bias $\downarrow$ Var ; $\uparrow Bias \Rightarrow \uparrow$ MSE
Kernel Density **KNN** $\hat{y} = \frac{1}{k} \sum_{i \in N_k(x)} y_i$; small k: low bias/high var; large k: high bias/low var. Locally constant assumption; poor in high-dim; complexity $\uparrow$ when $k \downarrow$. **KDE** $\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$ **Role of Bandwidth** h large: $\hat{f}_h(x)$ smooth and slowly varying, h small: $\hat{f}_h(x)$ more wiggly. **Bias–Variance Trade-off**: As $h \uparrow$, $|\mathrm{Bias}| \uparrow$ and Var $\downarrow$.
MSE:
$\mathrm{MSE}(x) = E[(\hat{f}(x) - f(x))^2] = (E[\hat{f}(x)] - f(x))^2 + \mathrm{Var}(\hat{f}(x)) = \mathrm{Bias}^2 + \mathrm{Var}$
MISE: $\int MSE(x)\, dx$, rate $O(n^{-4/5})$,estimated by empirical MSE; $\widehat{\mathrm{MSE}} = \frac{1}{n_{\text{test}}} \sum_{i \in \text{test}} (y_i - \hat{y}_i)^2$ **AMISE:** Approximation of MISE for large n, used to derive optimal bandwith h.
Non-parametric Regression Model: $Y_i = m(x_i) + \epsilon_i$, where $\epsilon_i = Y_i - m(x_i)$ satisfies $E(\epsilon_i \mid X_i) = 0$
Local Avg: $\hat{m}(x) = \frac{\sum K(\frac{x - x_i}{h}) Y_i}{\sum K(\frac{x - x_i}{h})}$ **Role of h**: as $h \to \infty$ we see $\hat{m}(x) \to \bar{y}$, as $h \to 0$ we see $\hat{m}(x_j) \to y_j$ (overfit) NW is example of Local Avg and performs a locally constant fit. **Asymptotically optimal local bandwidth** $h_{\text{opt}}(x) = n^{-1/5} \left(\frac{\sigma_\epsilon^2 \int K^2(z)\, dz}{\{m''(x) \int z^2 K(z)\, dz\}^2} \right)^{1/5}$
Effects of h_{opt}: $h_{\text{opt}} \propto n^{-1/5}$ $(\uparrow data \Rightarrow \downarrow h)$, $\uparrow \sigma_\varepsilon^2 \Rightarrow \uparrow h_{\text{opt}}$, $\uparrow |m''(x)| \Rightarrow \downarrow h_{\text{opt}}$, kernel affects constants only, balances bias
Local modeling: Fit simple model (e.g. linear) in nbhd of x, weights $K_h(x - x_i) \Rightarrow$ local linear regression. **Global modeling**: Fit smooth f on all data (e.g. polynomial, spline basis) with parameters shared globally.
Penalized modeling: Min $\sum_i (y_i - f(x_i))^2 + \lambda J(f)$, $J(f)$ penalizes roughness (e.g. $\int [f''(t)]^2 dt$). **Local Polynomial Regression (LOESS)**: Is generalization of kernel regression. Extends NW to locally polynomial fits, if we use degree $d = 0$, it's the same as kernel regression (NW).
Hat/Smoother Matrix: $\hat{y} = Sy$, S depends on smoother, e.g. LM: $H = X(X^\top X)^{-1} X^\top$; Splines: $S = X(X^\top X + \lambda\Omega)^{-1} X^\top$. **Deg. freedom**: df $= \mathrm{tr}(S)$, measures model complexity for non-parametric fit. S_{ij} is weight assigned to y_j when estimating $\hat{y}_i$ (how much influence/contribution). Diagonal elements S_{ii} are the leverages; how much observation i influences its own fitted value.
Smoothing Splines $\min_f \sum_i (y_i - m(x_i))^2 + \lambda \int [f''(t)]^2 dt$ $\lambda = 0$: solution interpolates data (overfit risk); $\lambda \to \infty$: solution approaches lin. func. (underfit risk). df=n-p (incl. β_0)
Model Evaluation/Selection **Model Selection** Best subset: fit all $\binom{p}{k}$, choose smallest RSS; RSS $\downarrow$ as $k \uparrow$, penalize complexity. Stepwise: Forward (start null, add vars), Backward (start full, remove vars), efficient but may miss global optimum. **Error Decomposition** Training error: $\mathrm{err} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$ Measures fit to training data, optimistic since same data is used. Test set error estimate: $\hat{\mathrm{Err}}_{\hat{f}} = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{f}(x_i))^2 \to_{m \to \infty} E[(y - \hat{f}(x))^2 \mid D_n]$ Unbiased estimate of conditional error and if we assume independence of model and test set then also for generalization error. Conditional error (extra-sample): $E[(y - \hat{f}(x))^2 \mid D_n]$ Expected prediction error on new unseen data given fixed D_n. Generalization error: $E[(y - f(x))^2]$ Best achievable error using true regression f, lower bound on performance. Estimated using Hold-out method, CV, etc. Mallows' C_p: $C_p = \mathrm{err} + \frac{2}{n} \hat{\sigma}^2 \,\mathrm{df}(\hat{f})$, highlightis an unbiased estimate of Err_{in}. AIC: $\mathrm{AIC} = n \log(\mathrm{err}) + 2\, \mathrm{df}(\hat{f})$ highlightBIC: $\mathrm{BIC} = n \log(\mathrm{err}) + \log(n)\, \mathrm{df}(\hat{f})$ Interpretation: AIC selects more complex models, BIC penalizes complexity more, best model minimizes AIC/BIC, only compare models on same data, only meaningful in differences, only applicable for likelihood-based models.
Cross-validation $\mathrm{CV}_K = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i \in S_k} L(y_i, \hat{f}^{(-k)}(x_i))$
Holdout (one split): depends on one random split; test/train proportion arbitrary; both bias and var can be poor; ok in clear-cut cases; fast.
LOOCV: approx. unbiased for GE; slight bias since train size is $n-1$; high variance due to strong correlation across folds.
K-fold CV: larger bias than LOOCV; unclear vs LOOCV for variance (common choices $K = 5, 10$).
Linear smoother setup: $\hat{\boldsymbol{m}} = S\boldsymbol{Y}$ (e.g., splines, least squares).
LOOCV error (shortcut): $\mathrm{CV} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{Y_i - \hat{m}(x_i)}{1 - S_{ii}} \right)^2$ (fit once on full data; need only S_{ii}).
Bootstrap Allows us to obtain CIs for an estimator, is consistent if the limiting distribution of the estimator $(\hat{\theta}_n)$ is Normal and if the data $(Z_1, \ldots, Z_n)$ are i.i.d. Normal CI: assumes estimator approx. normal, is symmetric simple and fast; poor if distr. skewed or small n Percentile CI: $[\hat{\theta}_{(\alpha/2)}, \hat{\theta}_{(1-\alpha/2)}]$ accounts for skewedness; doesn't account for bias, may undercover Basic CI (Reverse) Symmetric around $\hat{\theta}$ Adjusts for bias in bootstrap est.; assumes approx. symmetry; may distort under skewedness Studentized CI Adjusts for skewedness, heteroscedasticity and bias; often best coverage; computationally heavy **Studentization** Improve accuracy of bootstrap inference by standardizing estimator using an estimate of its standard error. Achieves second-order accuracy for sample mean, i.e. the coverage error is of order $O(1/n)$ instead of $O(1/\sqrt{n})$. Improves CI accuracy, particularly for small samples or when distribution of $\hat{\theta}$ is skewed or heteroscedastic. **Double BS** Improve coverage accuracy of BS CIs, can achieve second-order accuracy for CIs,
Classification Metrics $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$ (recall, sensitivity), $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$ (1 - specifity), $\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$ (specificity), $\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$ (precision), $\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}}$, $\mathrm{F1} = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV}+\mathrm{TPR}}$.
$\mathrm{MisclassificationRate} = \frac{\mathrm{FP}+\mathrm{FN}}{\mathrm{Total}} = \frac{\mathrm{FPR} * \mathrm{N}_{\mathrm{neg}} + (1 - \mathrm{TPR}) * N_{pos}}{N_{pos} + N_{neg}}$
ROC TPR(y) vs FPR(x) for all possible decision thresholds **AUC** Area under the ROC curve, measures overall predictive accuracy of model $AUC = 0.5$: random, $AUC \geq 0.95$: very good
LDA Finds linear decision boundary separating classes, low variance $\hat{p}_k = \frac{1}{n} \sum_{i=1}^{n} I(y_i = k)$ (class prior), $\hat{\mu}_k = \frac{\sum_{i=1}^{n} x_i I(y_i = k)}{\sum_{i=1}^{n} I(y_i = k)}$ (class mean), $\hat{\Sigma} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i=1}^{n} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^\top I(y_i = k)$ (shared cov). LDA Prediction $\hat{y} = \arg\max_k \log \Pr(Y = k \mid X = x)$, $\log \Pr(Y = k \mid x) = -\frac{1}{2}(x - \hat{\mu}_k)^\top \hat{\Sigma}^{-1} (x - \hat{\mu}_k) + \log \hat{p}_k + \text{const}$ Number of Parameters: For a predictors and b groups we have $a \times b$ for mean, $a(a+1)/2$ for cov. matrix and $b - 1$ for priors $\Rightarrow$ $ab + 1/2a(a+1) + b - 1$ total parameters.
QDA No longer assume equal cov. matrices for all categories, $\hat{y} = \arg\max_j \left(-\frac{1}{2} \log |\hat{\Sigma}_j| - \frac{1}{2}(x - \hat{\mu}_j)^\top \hat{\Sigma}_j^{-1} (x - \hat{\mu}_j) + \log \hat{p}_j \right)$ Number of Parameters: Now $ab(a+1)/2$ for cov. matrices $\Rightarrow$ $ab + ab(a+1)/2 + b - 1$ parameters.
SVM Does not make distributional assumptions, unlike LDA (models $P(Y, X)$ and Log. Reg. (models $P(Y \mid X)$). SVM finds maximum margin hyperplane. Hard-Margin SVM is infeasible when data not linearly separable, in which case we use Soft-Margin SVM with regularization parameter C. $\uparrow C \Rightarrow$ Narrower Margin and $\downarrow$ Bias, $\uparrow$ Var $\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C \sum \xi_i$ s.t. $y_i(w^\top \phi(x_i) + b) \geq 1 - \xi_i$, $\xi_i \geq 0$
GAM Makes it possible to estimate a regression function $m(x) = E[Y \mid X = x]$ or classification probabilities $\pi_j(x) = P[Y = j \mid X = x]$ fully non-parametrically without structure. GAMs is nonparametric and flexible model. $g(x) = \mu + \sum_{j=1}^{p} g_j(x_j)$, $\sum_i f_j(x_{ij}) = 0$ Generalizes linear models using flexible $g_j(.)$, uses backfitting algorithm to estimate component functions $g_j(x_j)$
Trees & Ensembles Single trees have poor predictive performance and high variance. Bagging reduces variance, bias stays similar over single tree. Boosting reduces bias.
Formulas Size of a binary tree is 2^{n-1} (n: leaves), for depth d, maximum number of nodes is $2^d - 1$
CART split: $\max_s \Delta I(s)$, Bagging: $\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*(b)}(x)$
Tukey-Anscombe Residuals on y-axis, fitted values on x-axis. A reason to use the fitted values is that the sample correlation between them and the residuals is always zero. The points in the plot should "fluctuate randomly" around the horizontal line through zero. if errors are uncorrelated; if error has zero mean; if error is normally distributed; if error has constant variance
Normal Q-Q Theoretical quantiles of standard normal distr. on x-axis, ordered standardized residuals on y-axis. If residuals are approx. normal distr. then points should lie close to 45° reference line. if error is normally distributed; if error has constant variance; if errors are uncorrelated; if error has zero mean
Location-Scale SQRT of abs. standardized residuals on y-axis, fitted values on x-axis. Random scatter of points means constant variance; a funnel shape suggests heteroscedasticity. if error has constant variance; if error is normally distributed; if errors are uncorrelated; if error has zero mean
Cook's Distance Plots Cook's distance for each observation, combines leverage and residual size to measure influence. Larger values $(D_i > 1)$ indicate more influential observations. if there are influential observations; if error is normally distributed; if error has constant variance; if error has zero mean; if errors are uncorrelated
Code Reconstruct summary table of fitted model based on 30 observations, call: lm(y $\sim$ x1 + x2 + x3 + x4)

```
n = 30; p = 5; # n: sample size, p: predictors
t_value = estimate/std_error
std_error = estimate/t
df = n - p; # 30 - 5 = 25
p_value = 2 * (1 - pt(abs(t_value), df));
```

Compute kernel density estimate using gaussian kernel. What is kernel density estimate at x = 2?

```
z <- sample(c(0,1), size = 20, replace = TRUE)
x <- z*rnorm(20) + (1-z)*(4+rnorm(20))
density(x, from = -2, to = 2, n = 3)$y[3] #0.08503856
density(x, from = -2, to = 2, n = 100)$y[100] #0.08503856
```

Analyze bias and variance (by simulation) of two non-parametric regression models; (gaussian) kernel regression and smoothing splines.

```
m <- function(x) cos(x);
nrep <- 200; # "200 different datasets"
x <- seq(0, 1, length.out = 100); # "100 design points"
est.nw <- est.ss <- matrix(0, nrow = length(x), ncol = nrep)
for(i in 1:nrep) {
  ## Simulate y-values
  y <- m(x) + rnorm(length(x))
  ## Get estimates for the mean function m(x)
  ## in the points given by the vector x
  est.nw[,i] <- ksmooth(x,y,bandwidth=0.1,kernel="normal")$y
  est.ss[,i] <- predict(smooth.spline(x,y, df=4), x = x)$y
}
# Estimator variance at each point x (len 100)
var.nw <- apply(est.nw, 1, var)
var.ss <- apply(est.ss, 1, var)
# Bias and MSE at each point x
bias.nw <- rowMeans(est.nw) - m(x)
mse.nw.vec <- var.nw + bias.nw^2
bias.ss <- rowMeans(est.ss) - m(x)
```

```r
mse.ss.vec <- var.ss + bias.ss^2
# rowMeans is shortcut for apply(..., mean)
# Estimation of MSE at x = 0 for kernel regression estimator
mse.nw = var.nw[1] + (mean(est.nw[1, ] - m(0)))^2
mse.nw.vec[1] # Alternative
# Estimation of Variance at x = 0 for smoothing splines
var.ss1 = var.ss[1] # or: var(est.ss[1, ])
# Does SS estimator have lower var than NW on all points?
all(var.nw > var.ss) # TRUE -> Yes
# At grid ends the SS has a smaller squared empirical bias than the NW?
bias.ss[1]^2 < bias.nw[1]^2 # TRUE
bias.ss[100]^2 < bias.nw[100]^2; # Also TRUE -> Yes
```

Compute NW kernel estimator $\hat{m}_{h_0}$ using kssmooth, use normal kernel and bandwidth=qnorm(0.75)/0.25. (1) What is $\hat{m}_{h_0}(1)$? (2) Since $\hat{y} = Sy$ for some smoother matrix S, what is $S_{1,1}$?

```r
x <- seq(-3,3,by=0.2);
y <- cos(x/2) + 0.1*rnorm(length(x))
# (1)
bw <- qnorm(0.75)/0.25
ksmooth(x,y,kernel = "normal",bandwidth=bw, x.points=1)$y
# (2) Approach A: Calculate first row and then S[1] gives S11
S = (dnorm(x[1] - x)/sum(dnorm(x[1] - x)))
S11 = S[1]
# (2) Approach B: Directly calculate S11
S11 = dnorm(0)/sum(dnorm(x[1]-x))
```

Prediction Functions + Build Hat Matrices (NW, LOESS, SS)

```r
library(sfsmisc) # Alternative Data:
# data(iris); X <- iris$Petal.Length; y <- iris$Sepal.Length
# Need to order x values for ksmooth & ss
# ix = order(X); X = X[ix]; y = y[ix]
X <- seq(-3, 3, by = 0.2); y=cos(X/2) + 0.1*rnorm(length(X))
bw <- qnorm(0.75)/0.25

pred.lm <- function(x, y, new_x) {
  fit <- lm(y ~ x)
  predict(fit, newdata = data.frame(x = new_x))
}
pred.ksmooth <- function(x, y, new_x) {
  ksmooth(x, y, kernel = "normal", bandwidth = bw, x.points = new_x)$y
}
pred.loess <- function(x, y, new_x) {
  fit <- loess(y ~ x)
  predict(fit, newdata = data.frame(x = new_x))
}
pred.ss <- function(x, y, new_x) {
  fit <- smooth.spline(x, y, spar = 0.6, all.knots = TRUE)
  predict(fit, new_x)$y
}
# For building hat matrices, we predict on same x (new_x=X)
S.nw <- hatMat(X, pred.sm = pred.ksmooth, new_x=X) # S11 = S[1][1]
S.loess <- hatMat(X, pred.sm = pred.loess, new_x=X)
S.ss <- hatMat(X, pred.sm = pred.ss, new_x=X)
```

Perform non-parametric regression using local polynomial fit (LOESS) and calculate LOOCV

```r
diabetes = read.table("...",header = TRUE)
dataset <- diabetes[, c("Age", "C.Peptide")]
names(dataset) <- c("x", "y")
dataset <- dataset[sort.list(dataset$x), ] # Sort for LOESS
loocv = function(data, model_fn) {
  # Compute LOO prediction for observation i
  loo_pred = function(i, data, model_fn) {
    # Fit model on all data except i, then return prediction on i
    return(model_fn(data$x[-i], data$y[-i], data$x[i]))
  }
  n = nrow(data)
  # array where preds[i] is LOO prediction for i
  preds = sapply(1:n, loo_pred, data, model_fn)
  mean((data$y - preds)^2) # LOOCV estimate of mean squared pred. error
}
# Local polynomial regression function (LOESS)
loess_predict = function(x, y, new_x) {
  fit = loess(y ~ x, surface = "direct")
  predict(fit, new_x)
}
# LOOCV estimate using helper
(cv_loocv = loocv(dataset, loess_predict)) # 0.3849...
# Calculate Hat Matrix (manually)
n = nrow(dataset)
I_n = diag(n)
S = matrix(0, n, n)
for (j in 1:n) {
  S[, j] = loess_predict(dataset$x, I_n[, j], dataset$x)
}
y_hat = loess_predict(dataset$x, dataset$y, dataset$x)
# Shortcut LOOCV error
(cv_hatMat = mean(((dataset$y-y_hat)/(1-diag(S)))^2)) # 0.3849...
```

Perform 10-fold cross-validated SS fitting with spar=0.4. Use the "faithful" dataset. Then calculate the mean squared errors to evaluate the performance. Determine the degrees of freedom.

```r
data("faithful"); x <- faithful$waiting; y <- faithful$eruptions
n <- nrow(faithful) # 272 observations
K <- 10 # 10 folds
folds <- sample(cut(seq(1, n),breaks = K, labels = FALSE), replace=F)
spar <- 0.4
# Store the CV MSE for each fold
cv <- matrix(NA, 1, K, dimnames = list(NULL, 1:K))
for (k in 1:K) {
    # Observation indices not in fold k (train)
    ind <- which(folds != k)
    # fit on train
    ss <- smooth.spline(x[ind],y[ind],spar=spar)
    # predict y for test set
    yhat <- predict(ss, x = x[-ind])$y
     # compute MSE on test fold
    cv[1, k] <- mean((yhat - y[-ind])^2)
}
# Average of 10 MSEs -> CV Error Estimate
cv.m <- apply(cv, 1, mean) # 0.151421
df <- ss$df # degrees of freedom

# Task B: Repeat but vary spar from 0 to 1 in steps 1/40.
spars <- (0:40)/40 # 0.000, 0.025, ..., 1.00
ns <- length(spars) # 41
# 41 x 10 matrix, cv.spars[i,]: fold errors for i-th spar
cv.spars <-matrix(NA,ns,K,dimnames=list(spars,1:K))
i <- 0;
for (spar in spars) {
  i <- i + 1
  for(k in 1:K){
    # ... (same as above)
    cv.spars[i,k] <- mean((yhat - y[-ind])^2)
  }
}
cv.m.spars = apply(cv.spars, 1, mean)
# Give CV MSE for spar = 0.4
cv.m.spars[spars==0.4] # 0.151421
# Which of these spar values minimizes CV MSE?
(spars[which.min(cv.m.spars)]) # 0.65
```

Bootstrap and CIs

```r
rData <- function(n)rgamma(n,shape=1.2)
true.param <- IQR(rData(100000000))# 1.2490...
sample40 <- rData(n = 40)
(th.hat <- IQR(sample40))# 1.117164
# IQR of resampled indices
tIQR <- function(x, ind) IQR(x[ind])
require("boot")
res.boot <- boot(data = sample40, statistic = tIQR, R = 10000)
bci <- boot.ci(res.boot,conf=0.95,type=c("basic","norm","perc"))
```

LDA, QDA, Multinomial Regression and Misclassification Rates

```r
library(MASS);library(nnet);library(ROCR);
# --- Iris: LDA, QDA, multinomial ---
Iris <- iris[,c("Petal.Length","Petal.Width","Species")]
lda.fit   <- lda(Species ~ ., Iris)
qda.fit   <- qda(Species ~ ., Iris)
multi.fit <- multinom(Species ~ ., Iris)
mean(predict(lda.fit, Iris)$class != Iris$Species) # MR
mean(predict(qda.fit, Iris)$class != Iris$Species) # MR
mean(predict(multi.fit, Iris) != Iris$Species) # MR
# --- Binary logistic regression ---
d.baby <- read.table("...", header=TRUE)
fit <- glm(Survival ~ ., data=d.baby, family=binomial)
pred <- prediction(fit$fitted.values, d.baby$Survival)
perf.roc  <- performance(pred, "tpr","fpr")
perf.cost <- performance(pred, "cost")
prob <- predict(fit, type="response")
yhat <- ifelse(prob > 0.5, 1, 0)
mean(yhat != d.baby$Survival) # MR
```

Generate classification tree using cp = 0 and minsplit = 30, then prune it using cost-complexity criterion.

```r
require(rpart)
rp.veh <- rpart(Class~.,data=data,control=rpart.control(cp =
↪  0.0,minsplit=30))
preds = predict(rp.veh, newdata = data, type = "class")
table(data$Class, preds)
require(rpart.plot) # Plot Tree
prp(rp.veh,extra=1,type=1,box.col=c("pink","red","blue","green")[rp.ve⌋
↪  h$frame$yval])
## PRUNING: Find smallest cp value that is below line (plot) in table
plotcp(rp.veh); printcp(rp.veh);
cp.opt <- rp.veh$cptable[7, "CP"] # Optimal cp value is 0.00875
```

Bootstrap Resampling for means, CIs and SE/Bias

```r
x <- subset(iris, Species == "setosa")$Petal.Length
n <- length(x)
B <- 1000
# Pre-Generate bootstrap resampling indices
index <- matrix(sample.int(n, n*B, replace=TRUE), nrow=n, ncol=B)
boot_means <- numeric(B)
# FOR OOB: oob_errors = rep(0,B)
for (i in 1:B) {
  ind <- index[, i]
  # FOR OOB: do predict(... newdata=x[-ind]) then store in oob_errors
  boot_means[i] <- mean(x[ind])
}
# Alternative Approach using replicate()
# boot_means <- replicate(B, mean(sample(x, size=n, replace=TRUE)))

# Full-sample estimate
theta_hat <- mean(x)
# Bootstrap SE
se <- sd(boot_means)
# Bootstrap Bias
bias <- mean(boot_means) - theta_hat
# 95% Percentile CI for mean
ci <- quantile(boot_means, c(0.025, 0.975))
```

SVM Example

```r
library(e1071)
tuned <- tune.svm(Species ~ ., data = train, kernel = "radial",
                  cost = c(0.01,0.1,1,10,100),gamma = c(0.1,1,10)
summary(tuned)
best <- tuned$best.model
pred <- predict(best, test)
# Mallow's Cp
sigma <- summary(fit)$sigma
Cp <- function(object,sigma){ # object: fit of a model
    res <- residuals(object)
    n <- length(res)
    p <- n - object$df.residual
    SSE <- sum(res^2)
    SSE / sigma^2 - n + 2 * p
} # pick model with smallest Cp
```

Model Selection

```r
mortal.full <- lm(Mortality ~ . , data=mortality)
mortal.empty <- lm(Mortality ~ 1, data = mortality)
mortal.bw <- step(mortal.full, direction = "backward")
mortal.fw <- step(mortal.empty, direction = "forward",
scope = list(upper=mortal.full, lower=mortal.empty))
```

Mamahuevo Misc

```r
x <- 1:10; y <- 2*x + rnorm(10, 0, 2); fit <- lm(y ~ x);
y_hat <- fitted(fit) # Compute manually
rss = sum((y-y_hat)^2); tss = sum((y-mean(y))^2); r2  = 1 - rss/tss;
generalization_error_est[i] = mean(predict(fit_Q, Iris[-ind,
↪  c("Petal.Length","Petal.Width") ])$class != Iris[-ind, "Species"])
# Log Reg for binary classification
fit = glm(Y~., data=data, family="binomial")
mean((predict(fit, type="response") > 0.5) == data$Y)
confint(lm_fit) # 95% CI for fit coefficients
fit <- lm(y ~ x, data=df, subset=ind); # df = data.frame(x,y)
preds <- predict(fit, newdata=df[-ind,]);
# Alternative way to calculate estimated prediction error
fit = glm(y~x, data=df) cv.err = cv.glm(df, fit, K=5)$delta; cv.err[1]
```