> rm(list-ls())

Error in rm(list - ls()) : ... must contain names or character strings

> library(ISLR)

> library(boot)

Attaching package: ‘boot’

The following object is masked from ‘package:lattice’:

melanoma

>

> # Exercise 1.

> # Cross-validation can also be used to estimate the test error for a classification problem.

> # Run a logit model with the Smarket data. The dependent variable is Direction

> # glm.fit <- glm(Direction~Lag1+Lag2, family=binomial, data=Smarket)

> # summary(glm.fit)

> set.seed(17)

>

> cv.error.1 <- rep(NA, 10)

> glm.fit.1 <- glm(Direction~Lag1+Lag2, family=binomial, data=Smarket)

> cv.error.1 <- cv.glm(Smarket, glm.fit.1, K=10)$delta[1]

> cv.error.1

[1] 0.2503548

>

> cv.error.2 <- rep(NA, 10)

> glm.fit.2 <- glm(Direction~Lag1+Lag2+Lag3, family=binomial, data=Smarket)

> cv.error.2 <- cv.glm(Smarket, glm.fit.2, K=10)$delta[1]

> cv.error.2

[1] 0.2507165

>

> cv.error.3 <- rep(NA, 10)

> glm.fit.3 <- glm(Direction~Lag1+Lag2+Lag3+Lag4, family=binomial, data=Smarket)

> cv.error.3 <- cv.glm(Smarket, glm.fit.3, K=10)$delta[1]

> cv.error.3

[1] 0.2512181

>

> cv.error.4 <- rep(NA, 10)

> glm.fit.4 <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5, family=binomial, data=Smarket)

> cv.error.4 <- cv.glm(Smarket, glm.fit.4, K=10)$delta[1]

> cv.error.4

[1] 0.2518706

>

> # Compare this model with the following models using K fold cross-validation with K=10.

> # Direction~Lag1+Lag2+Lag3, Direction~Lag1+Lag2+Lag3+Lag4, Direction~Lag1+Lag2+Lag3+Lag4+Lag5

>

>

>

> # Exercise 2.

> # Consider KNN estimation to predict direction using Lag1 and Lag2. To choose the optimal number of neighbors,

> # use the K=10, k fold cross validation. Use only 2004 and 2005 year data.

>

> library(class)

> set.seed(1)

>

> n <- nrow(Smarket)

> ind <- 1:n

>

>

> for (i in 1:10) {

+ testSplit <- sample(ind, size=n/10, replace=FALSE)

+ train <- Smarket[-testSplit, ]

+ test <- Smarket[testSplit, ]

+ train.x <- train[, c("Lag1", "Lag2")]

+ test.x <- test[, c("Lag1", "Lag2")]

+ train.y <- train$Direction

+ test.y <- test$Direction

+

+ ind <- ind[-testSplit]

+

+ Acc <- rep(NA, 100)

+

+ for (j in 1:100) {

+ knnPred <- knn(train.x, test.x, train.y, k=j)

+ Acc[j] <- mean(test.y == knnPred)

+ }

+ optKs <- which.max(Acc)

+ }

>

> optKs

[1] 38

> BestK <- mean(optKs)

>

> df <- Smarket

> # install.packages("caret")

> # install.packages("e1071")

> library(caret)

> trControl = trainControl(method="cv", number=10)

> fit <- train(Direction~Lag1+Lag2, method="knn", tuneGrid=expand.grid(k=1:50), trControl=trControl, metric="Accuracy", data=Smarket)

> fit

k-Nearest Neighbors

1250 samples

2 predictor

2 classes: 'Down', 'Up'

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 1125, 1126, 1124, 1125, 1125, 1125, ...

Resampling results across tuning parameters:

k Accuracy Kappa

1 0.5072638 0.0132380954

2 0.5160583 0.0289832740

3 0.5015873 -0.0003944142

4 0.4927737 -0.0182139944

5 0.5007548 -0.0018738457

6 0.5016059 -0.0001213371

7 0.4767865 -0.0513281017

8 0.5031937 0.0023702911

9 0.4799418 -0.0443287005

10 0.4895677 -0.0262621970

11 0.4703671 -0.0640120298

12 0.4759800 -0.0534156150

13 0.4743608 -0.0555445385

14 0.4863674 -0.0316709820

15 0.4768436 -0.0520842892

16 0.4719670 -0.0615664643

17 0.4743797 -0.0565129867

18 0.4839863 -0.0377674587

19 0.4824182 -0.0416710249

20 0.4768631 -0.0532545633

21 0.4864249 -0.0340436263

22 0.4968768 -0.0132340165

23 0.4960828 -0.0165984575

24 0.4952894 -0.0175375036

25 0.5088386 0.0095674488

26 0.5064062 0.0045968763

27 0.5055934 0.0029935301

28 0.5216262 0.0360215252

29 0.5143811 0.0208710102

30 0.5135874 0.0189634721

31 0.5048066 0.0008426084

32 0.5032382 -0.0025323664

33 0.5160260 0.0233507109

34 0.5087934 0.0098321276

35 0.5192006 0.0305570598

36 0.5112069 0.0144234213

37 0.5096257 0.0116479355

38 0.5224324 0.0368939806

39 0.5200003 0.0320222456

40 0.5047868 0.0011085906

41 0.5144964 0.0206922788

42 0.5088639 0.0089595316

43 0.5128193 0.0167203646

44 0.5080127 0.0076119944

45 0.5048124 0.0005737322

46 0.5007871 -0.0085801494

47 0.5024060 -0.0044392503

48 0.5143680 0.0197529813

49 0.4999675 -0.0100442663

50 0.5072127 0.0049307672

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was k = 38.