DATA 590 | Homework 3 | Praveen Kumar Neelappa

**See how well you can predict the presence of coronary heart disease (CHD)**

**inputs: all predictors, independent variables,**

**output: CHD yes/no**

1. **Using KPD and kNN methods, see how well you can predict CHD based on all the quantitative inputs. Be sure to use confusionMatrix() to help you interpret.**
2. **Scale the data for kNN, and use at least 4 different ks (1,5,11,19).**

> sa <-read.csv('http://math.mercyhurst.edu/~sousley/STAT\_139/data/sahdd.csv', as.is = T);

> head(sa)

row sbp tobacco ldl adiposity famhist typea obesity alcohol age chd

1 1 160 12.00 5.73 23.11 Present 49 25.30 97.20 52 1

2 2 144 0.01 4.41 28.61 Absent 55 28.87 2.06 63 1

3 3 118 0.08 3.48 32.28 Present 52 29.14 3.81 46 0

4 4 170 7.50 6.41 38.03 Present 51 31.99 24.26 58 1

5 5 134 13.60 3.50 27.78 Present 60 25.99 57.34 49 1

6 6 132 6.20 6.47 36.21 Present 62 30.77 14.14 45 0

> attach(sa)

The following objects are masked from sa (pos = 3):

adiposity, age, alcohol, chd, famhist, ldl, obesity, row, sbp, tobacco, typea

> x = na.omit(sa[c(2, 3, 4,5,7,8,9,10)])

> y = sa$chd

**#using knnmethod with k =1**

>

> kcres1 <- knn.cv(x, y, k = 1, prob = TRUE)

> round(100\*prop.table(table(kcres1,y),1),1)

y

kcres1 0 1

0 66.2 33.8

1 63.4 36.6

>

> confusionMatrix(kcres1,y)

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 217 111

1 85 49

Accuracy : 0.5758

95% CI : (0.5292, 0.6213)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.99978

Kappa : 0.0258

Mcnemar's Test P-Value : 0.07415

Sensitivity : 0.7185

Specificity : 0.3063

Pos Pred Value : 0.6616

Neg Pred Value : 0.3657

Prevalence : 0.6537

Detection Rate : 0.4697

Detection Prevalence : 0.7100

Balanced Accuracy : 0.5124

'Positive' Class : 0

**#using knnmethod with k =5**

>

> kcres5 <- knn.cv(x, y, k = 5, prob = TRUE)

> round(100\*prop.table(table(kcres5,y),1),1)

y

kcres5 0 1

0 66.2 33.8

1 62.7 37.3

>

> confusionMatrix(kcres5,y)

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 233 119

1 69 41

Accuracy : 0.5931

95% CI : (0.5467, 0.6382)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.997085

Kappa : 0.03

Mcnemar's Test P-Value : 0.000352

Sensitivity : 0.7715

Specificity : 0.2562

Pos Pred Value : 0.6619

Neg Pred Value : 0.3727

Prevalence : 0.6537

Detection Rate : 0.5043

Detection Prevalence : 0.7619

Balanced Accuracy : 0.5139

'Positive' Class : 0

**#using knnmethod with k =11**

>

> kcres11 <- knn.cv(x, y, k = 11, prob = TRUE)

> round(100\*prop.table(table(kcres11,y),1),1)

y

kcres11 0 1

0 69.7 30.3

1 49.5 50.5

>

> confusionMatrix(kcres11,y)

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 253 110

1 49 50

Accuracy : 0.6558

95% CI : (0.6105, 0.6991)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.4825

Kappa : 0.165

Mcnemar's Test P-Value : 1.952e-06

Sensitivity : 0.8377

Specificity : 0.3125

Pos Pred Value : 0.6970

Neg Pred Value : 0.5051

Prevalence : 0.6537

Detection Rate : 0.5476

Detection Prevalence : 0.7857

Balanced Accuracy : 0.5751

'Positive' Class : 0

**#using knnmethod with k =19**

>

> kcres19 <- knn.cv(x, y, k = 19, prob = TRUE)

> round(100\*prop.table(table(kcres19,y),1),1)

y

kcres19 0 1

0 70.2 29.8

1 45.6 54.4

>

> confusionMatrix(kcres19,y)

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 261 111

1 41 49

Accuracy : 0.671

95% CI : (0.6261, 0.7137)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.2324

Kappa : 0.19

Mcnemar's Test P-Value : 2.185e-08

Sensitivity : 0.8642

Specificity : 0.3063

Pos Pred Value : 0.7016

Neg Pred Value : 0.5444

Prevalence : 0.6537

Detection Rate : 0.5649

Detection Prevalence : 0.8052

Balanced Accuracy : 0.5852

'Positive' Class : 0

1. **Use the scaled data for KPD with Euclidean and Minkowski (k >= 3), and if Mahalanobis works, use it too. It may not. How will you tell? It should be obvious if it does not. Use the kernel type (weight) assigned for each distance method.**

> sa <-read.csv('http://math.mercyhurst.edu/~sousley/STAT\_139/data/sahdd.csv', as.is = T);

> head(sa)

row sbp tobacco ldl adiposity famhist typea obesity alcohol age chd

1 1 160 12.00 5.73 23.11 Present 49 25.30 97.20 52 1

2 2 144 0.01 4.41 28.61 Absent 55 28.87 2.06 63 1

3 3 118 0.08 3.48 32.28 Present 52 29.14 3.81 46 0

4 4 170 7.50 6.41 38.03 Present 51 31.99 24.26 58 1

5 5 134 13.60 3.50 27.78 Present 60 25.99 57.34 49 1

6 6 132 6.20 6.47 36.21 Present 62 30.77 14.14 45 0

> attach(sa)

>

> sa <- as.data.frame(lapply(sa[c(2, 3, 4,5,7,8,9,10, 11)], normalize))

>

> x = na.omit(sa[1:8])

> y = sa$chd

> levels(as.factor(y))

[1] "0" "1"

>

> levels(as.factor(y))[1]

[1] "0"

>

> y[which( y == levels(as.factor(y))[2])] <- '2'

>

> y[which( y == levels(as.factor(y))[1])] <- '1'

>

> y <- as.numeric(y)

**#using Euclidean & logistic**

>

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 3, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 232 97

2 70 63

Accuracy : 0.6385

95% CI : (0.5929, 0.6824)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.76902

Kappa : 0.1687

Mcnemar's Test P-Value : 0.04423

Sensitivity : 0.7682

Specificity : 0.3937

Pos Pred Value : 0.7052

Neg Pred Value : 0.4737

Prevalence : 0.6537

Detection Rate : 0.5022

Detection Prevalence : 0.7121

Balanced Accuracy : 0.5810

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 5, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 244 98

2 58 62

Accuracy : 0.6623

95% CI : (0.6172, 0.7054)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.367777

Kappa : 0.2077

Mcnemar's Test P-Value : 0.001793

Sensitivity : 0.8079

Specificity : 0.3875

Pos Pred Value : 0.7135

Neg Pred Value : 0.5167

Prevalence : 0.6537

Detection Rate : 0.5281

Detection Prevalence : 0.7403

Balanced Accuracy : 0.5977

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 9, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 252 103

2 50 57

Accuracy : 0.6688

95% CI : (0.6239, 0.7116)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.2635

Kappa : 0.2068

Mcnemar's Test P-Value : 2.623e-05

Sensitivity : 0.8344

Specificity : 0.3563

Pos Pred Value : 0.7099

Neg Pred Value : 0.5327

Prevalence : 0.6537

Detection Rate : 0.5455

Detection Prevalence : 0.7684

Balanced Accuracy : 0.5953

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 11, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 254 98

2 48 62

Accuracy : 0.684

95% CI : (0.6394, 0.7262)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.09274

Kappa : 0.2467

Mcnemar's Test P-Value : 5.008e-05

Sensitivity : 0.8411

Specificity : 0.3875

Pos Pred Value : 0.7216

Neg Pred Value : 0.5636

Prevalence : 0.6537

Detection Rate : 0.5498

Detection Prevalence : 0.7619

Balanced Accuracy : 0.6143

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 15, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 256 111

2 46 49

Accuracy : 0.6602

95% CI : (0.615, 0.7033)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.4053

Kappa : 0.1702

Mcnemar's Test P-Value : 3.26e-07

Sensitivity : 0.8477

Specificity : 0.3063

Pos Pred Value : 0.6975

Neg Pred Value : 0.5158

Prevalence : 0.6537

Detection Rate : 0.5541

Detection Prevalence : 0.7944

Balanced Accuracy : 0.5770

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 19, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 264 111

2 38 49

Accuracy : 0.6775

95% CI : (0.6327, 0.7199)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.1522

Kappa : 0.2021

Mcnemar's Test P-Value : 3.669e-09

Sensitivity : 0.8742

Specificity : 0.3063

Pos Pred Value : 0.7040

Neg Pred Value : 0.5632

Prevalence : 0.6537

Detection Rate : 0.5714

Detection Prevalence : 0.8117

Balanced Accuracy : 0.5902

'Positive' Class : 1

**#using minkowski & Logisitic**

>

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 3, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 230 96

2 72 64

Accuracy : 0.6364

95% CI : (0.5907, 0.6803)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.79748

Kappa : 0.1675

Mcnemar's Test P-Value : 0.07598

Sensitivity : 0.7616

Specificity : 0.4000

Pos Pred Value : 0.7055

Neg Pred Value : 0.4706

Prevalence : 0.6537

Detection Rate : 0.4978

Detection Prevalence : 0.7056

Balanced Accuracy : 0.5808

'Positive' Class : 1

>

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 5, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 238 102

2 64 58

Accuracy : 0.6407

95% CI : (0.5951, 0.6845)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.738427

Kappa : 0.1595

Mcnemar's Test P-Value : 0.004082

Sensitivity : 0.7881

Specificity : 0.3625

Pos Pred Value : 0.7000

Neg Pred Value : 0.4754

Prevalence : 0.6537

Detection Rate : 0.5152

Detection Prevalence : 0.7359

Balanced Accuracy : 0.5753

'Positive' Class : 1

>

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 9, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 254 101

2 48 59

Accuracy : 0.6775

95% CI : (0.6327, 0.7199)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.1522

Kappa : 0.2275

Mcnemar's Test P-Value : 2.044e-05

Sensitivity : 0.8411

Specificity : 0.3688

Pos Pred Value : 0.7155

Neg Pred Value : 0.5514

Prevalence : 0.6537

Detection Rate : 0.5498

Detection Prevalence : 0.7684

Balanced Accuracy : 0.6049

'Positive' Class : 1

>

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 11, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 258 101

2 44 59

Accuracy : 0.6861

95% CI : (0.6417, 0.7282)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.07732

Kappa : 0.2434

Mcnemar's Test P-Value : 3.311e-06

Sensitivity : 0.8543

Specificity : 0.3688

Pos Pred Value : 0.7187

Neg Pred Value : 0.5728

Prevalence : 0.6537

Detection Rate : 0.5584

Detection Prevalence : 0.7771

Balanced Accuracy : 0.6115

'Positive' Class : 1

>

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 15, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 265 105

2 37 55

Accuracy : 0.6926

95% CI : (0.6483, 0.7344)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.04256

Kappa : 0.2458

Mcnemar's Test P-Value : 1.882e-08

Sensitivity : 0.8775

Specificity : 0.3438

Pos Pred Value : 0.7162

Neg Pred Value : 0.5978

Prevalence : 0.6537

Detection Rate : 0.5736

Detection Prevalence : 0.8009

Balanced Accuracy : 0.6106

'Positive' Class : 1

**#using minkowski & Logisitic**

>

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 19, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 266 109

2 36 51

Accuracy : 0.6861

95% CI : (0.6417, 0.7282)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.07732

Kappa : 0.2235

Mcnemar's Test P-Value : 2.241e-09

Sensitivity : 0.8808

Specificity : 0.3187

Pos Pred Value : 0.7093

Neg Pred Value : 0.5862

Prevalence : 0.6537

Detection Rate : 0.5758

Detection Prevalence : 0.8117

Balanced Accuracy : 0.5998

'Positive' Class : 1

**#using mahalanobis & Logisitic**

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 3, regression = FALSE,

+ Levels = unique(y), method = 'mahalanobis', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 221 93

2 81 67

Accuracy : 0.6234

95% CI : (0.5774, 0.6677)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.9212

Kappa : 0.1532

Mcnemar's Test P-Value : 0.4043

Sensitivity : 0.7318

Specificity : 0.4188

Pos Pred Value : 0.7038

Neg Pred Value : 0.4527

Prevalence : 0.6537

Detection Rate : 0.4784

Detection Prevalence : 0.6797

Balanced Accuracy : 0.5753

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 5, regression = FALSE,

+ Levels = unique(y), method = 'mahalanobis', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 245 104

2 57 56

Accuracy : 0.6515

95% CI : (0.6061, 0.695)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.5602290

Kappa : 0.1732

Mcnemar's Test P-Value : 0.0002886

Sensitivity : 0.8113

Specificity : 0.3500

Pos Pred Value : 0.7020

Neg Pred Value : 0.4956

Prevalence : 0.6537

Detection Rate : 0.5303

Detection Prevalence : 0.7554

Balanced Accuracy : 0.5806

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 9, regression = FALSE,

+ Levels = unique(y), method = 'mahalanobis', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 250 99

2 52 61

Accuracy : 0.6732

95% CI : (0.6283, 0.7158)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.2033940

Kappa : 0.2246

Mcnemar's Test P-Value : 0.0001815

Sensitivity : 0.8278

Specificity : 0.3812

Pos Pred Value : 0.7163

Neg Pred Value : 0.5398

Prevalence : 0.6537

Detection Rate : 0.5411

Detection Prevalence : 0.7554

Balanced Accuracy : 0.6045

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 11, regression = FALSE,

+ Levels = unique(y), method = 'mahalanobis', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 256 103

2 46 57

Accuracy : 0.6775

95% CI : (0.6327, 0.7199)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.1522

Kappa : 0.2226

Mcnemar's Test P-Value : 4.482e-06

Sensitivity : 0.8477

Specificity : 0.3563

Pos Pred Value : 0.7131

Neg Pred Value : 0.5534

Prevalence : 0.6537

Detection Rate : 0.5541

Detection Prevalence : 0.7771

Balanced Accuracy : 0.6020

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 15, regression = FALSE,

+ Levels = unique(y), method = 'mahalanobis', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 260 111

2 42 49

Accuracy : 0.6688

95% CI : (0.6239, 0.7116)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.2635

Kappa : 0.186

Mcnemar's Test P-Value : 3.853e-08

Sensitivity : 0.8609

Specificity : 0.3063

Pos Pred Value : 0.7008

Neg Pred Value : 0.5385

Prevalence : 0.6537

Detection Rate : 0.5628

Detection Prevalence : 0.8030

Balanced Accuracy : 0.5836

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 19, regression = FALSE,

+ Levels = unique(y), method = 'mahalanobis', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 263 117

2 39 43

Accuracy : 0.6623

95% CI : (0.6172, 0.7054)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.3678

Kappa : 0.1577

Mcnemar's Test P-Value : 7.051e-10

Sensitivity : 0.8709

Specificity : 0.2687

Pos Pred Value : 0.6921

Neg Pred Value : 0.5244

Prevalence : 0.6537

Detection Rate : 0.5693

Detection Prevalence : 0.8225

Balanced Accuracy : 0.5698

'Positive' Class : 1

1. **Use the UNscaled data for KPD with Euclidean and Minkowski (k >= 3), and if Mahalanobis works, use it too. Use the kernel type (weight) assigned for each distance method. (see next page)**

> sa <-read.csv('http://math.mercyhurst.edu/~sousley/STAT\_139/data/sahdd.csv', as.is = T);

> head(sa)

row sbp tobacco ldl adiposity famhist typea obesity alcohol age chd

1 1 160 12.00 5.73 23.11 Present 49 25.30 97.20 52 1

2 2 144 0.01 4.41 28.61 Absent 55 28.87 2.06 63 1

3 3 118 0.08 3.48 32.28 Present 52 29.14 3.81 46 0

4 4 170 7.50 6.41 38.03 Present 51 31.99 24.26 58 1

5 5 134 13.60 3.50 27.78 Present 60 25.99 57.34 49 1

6 6 132 6.20 6.47 36.21 Present 62 30.77 14.14 45 0

> attach(sa)

>

> x = na.omit(sa[c(2, 3, 4,5,7,8,9,10)])

> y = sa$chd

>

> levels(as.factor(y))

[1] "0" "1"

>

> levels(as.factor(y))[1]

[1] "0"

>

> y[which( y == levels(as.factor(y))[2])] <- '2'

>

> y[which( y == levels(as.factor(y))[1])] <- '1'

>

> y <- as.numeric(y)

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 3, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 213 113

2 89 47

Accuracy : 0.5628

95% CI : (0.5162, 0.6086)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 1.0000

Kappa : -0.001

Mcnemar's Test P-Value : 0.1056

Sensitivity : 0.7053

Specificity : 0.2938

Pos Pred Value : 0.6534

Neg Pred Value : 0.3456

Prevalence : 0.6537

Detection Rate : 0.4610

Detection Prevalence : 0.7056

Balanced Accuracy : 0.4995

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 5, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 233 119

2 69 41

Accuracy : 0.5931

95% CI : (0.5467, 0.6382)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.997085

Kappa : 0.03

Mcnemar's Test P-Value : 0.000352

Sensitivity : 0.7715

Specificity : 0.2562

Pos Pred Value : 0.6619

Neg Pred Value : 0.3727

Prevalence : 0.6537

Detection Rate : 0.5043

Detection Prevalence : 0.7619

Balanced Accuracy : 0.5139

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 9, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 248 109

2 54 51

Accuracy : 0.6472

95% CI : (0.6017, 0.6908)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.6356

Kappa : 0.1522

Mcnemar's Test P-Value : 2.341e-05

Sensitivity : 0.8212

Specificity : 0.3187

Pos Pred Value : 0.6947

Neg Pred Value : 0.4857

Prevalence : 0.6537

Detection Rate : 0.5368

Detection Prevalence : 0.7727

Balanced Accuracy : 0.5700

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 11, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 253 110

2 49 50

Accuracy : 0.6558

95% CI : (0.6105, 0.6991)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.4825

Kappa : 0.165

Mcnemar's Test P-Value : 1.952e-06

Sensitivity : 0.8377

Specificity : 0.3125

Pos Pred Value : 0.6970

Neg Pred Value : 0.5051

Prevalence : 0.6537

Detection Rate : 0.5476

Detection Prevalence : 0.7857

Balanced Accuracy : 0.5751

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 15, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 258 113

2 44 47

Accuracy : 0.6602

95% CI : (0.615, 0.7033)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.4053

Kappa : 0.1648

Mcnemar's Test P-Value : 5.731e-08

Sensitivity : 0.8543

Specificity : 0.2938

Pos Pred Value : 0.6954

Neg Pred Value : 0.5165

Prevalence : 0.6537

Detection Rate : 0.5584

Detection Prevalence : 0.8030

Balanced Accuracy : 0.5740

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 19, regression = FALSE,

+ Levels = unique(y), method = 'euclidean', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 261 111

2 41 49

Accuracy : 0.671

95% CI : (0.6261, 0.7137)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.2324

Kappa : 0.19

Mcnemar's Test P-Value : 2.185e-08

Sensitivity : 0.8642

Specificity : 0.3063

Pos Pred Value : 0.7016

Neg Pred Value : 0.5444

Prevalence : 0.6537

Detection Rate : 0.5649

Detection Prevalence : 0.8052

Balanced Accuracy : 0.5852

'Positive' Class : 1

#using minkowski

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 3, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 211 114

2 91 46

Accuracy : 0.5563

95% CI : (0.5097, 0.6022)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 1.0000

Kappa : -0.0143

Mcnemar's Test P-Value : 0.1244

Sensitivity : 0.6987

Specificity : 0.2875

Pos Pred Value : 0.6492

Neg Pred Value : 0.3358

Prevalence : 0.6537

Detection Rate : 0.4567

Detection Prevalence : 0.7035

Balanced Accuracy : 0.4931

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 5, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 227 117

2 75 43

Accuracy : 0.5844

95% CI : (0.538, 0.6298)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.999152

Kappa : 0.0217

Mcnemar's Test P-Value : 0.003087

Sensitivity : 0.7517

Specificity : 0.2687

Pos Pred Value : 0.6599

Neg Pred Value : 0.3644

Prevalence : 0.6537

Detection Rate : 0.4913

Detection Prevalence : 0.7446

Balanced Accuracy : 0.5102

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 9, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 245 112

2 57 48

Accuracy : 0.6342

95% CI : (0.5884, 0.6782)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.8237

Kappa : 0.121

Mcnemar's Test P-Value : 3.269e-05

Sensitivity : 0.8113

Specificity : 0.3000

Pos Pred Value : 0.6863

Neg Pred Value : 0.4571

Prevalence : 0.6537

Detection Rate : 0.5303

Detection Prevalence : 0.7727

Balanced Accuracy : 0.5556

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 11, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 254 111

2 48 49

Accuracy : 0.6558

95% CI : (0.6105, 0.6991)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.4825

Kappa : 0.1623

Mcnemar's Test P-Value : 8.792e-07

Sensitivity : 0.8411

Specificity : 0.3063

Pos Pred Value : 0.6959

Neg Pred Value : 0.5052

Prevalence : 0.6537

Detection Rate : 0.5498

Detection Prevalence : 0.7900

Balanced Accuracy : 0.5737

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 15, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 262 112

2 40 48

Accuracy : 0.671

95% CI : (0.6261, 0.7137)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.2324

Kappa : 0.1874

Mcnemar's Test P-Value : 8.468e-09

Sensitivity : 0.8675

Specificity : 0.3000

Pos Pred Value : 0.7005

Neg Pred Value : 0.5455

Prevalence : 0.6537

Detection Rate : 0.5671

Detection Prevalence : 0.8095

Balanced Accuracy : 0.5838

'Positive' Class : 1

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 19, regression = FALSE,

+ Levels = unique(y), method = 'minkowski', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 265 120

2 37 40

Accuracy : 0.6602

95% CI : (0.615, 0.7033)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.4053

Kappa : 0.1452

Mcnemar's Test P-Value : 5.977e-11

Sensitivity : 0.8775

Specificity : 0.2500

Pos Pred Value : 0.6883

Neg Pred Value : 0.5195

Prevalence : 0.6537

Detection Rate : 0.5736

Detection Prevalence : 0.8333

Balanced Accuracy : 0.5637

'Positive' Class : 1

#using Mahalonabis

> outKNN <- KernelKnn(x, TEST\_data = NULL, as.numeric(y), k = 3, regression = FALSE,

+ Levels = unique(y), method = 'mahalanobis', weights\_function = 'logistic')

>

> Classinto <- c(0,0,0)

> for (i in seq(length(y)))

+ {

+ Classinto[i] <- match(1,match(outKNN[i,], max(outKNN[i,])))

+ }

>

> confusionMatrix(Classinto,y)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 302 160

2 0 0

Accuracy : 0.6537

95% CI : (0.6083, 0.697)

No Information Rate : 0.6537

P-Value [Acc > NIR] : 0.5215

Kappa : 0

Mcnemar's Test P-Value : <2e-16

Sensitivity : 1.0000

Specificity : 0.0000

Pos Pred Value : 0.6537

Neg Pred Value : NaN

Prevalence : 0.6537

Detection Rate : 0.6537

Detection Prevalence : 1.0000

Balanced Accuracy : 0.5000

'Positive' Class : 1

Warning message:

In confusionMatrix.default(Classinto, y) :

Levels are not in the same order for reference and data. Refactoring data to match.

1. Can you figure out if inputs as have a positive or negative effect on wellness (no CHD)?

> install.packages("popbio")

Installing package into ‘C:/Users/pneelappa/Documents/R/win-library/3.3’

(as ‘lib’ is unspecified)

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.3/popbio\_2.4.3.zip'

Content type 'application/zip' length 242133 bytes (236 KB)

downloaded 236 KB

package ‘popbio’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\pneelappa\AppData\Local\Temp\Rtmpc5rTGf\downloaded\_packages

> library("popbio")

Attaching package: ‘popbio’

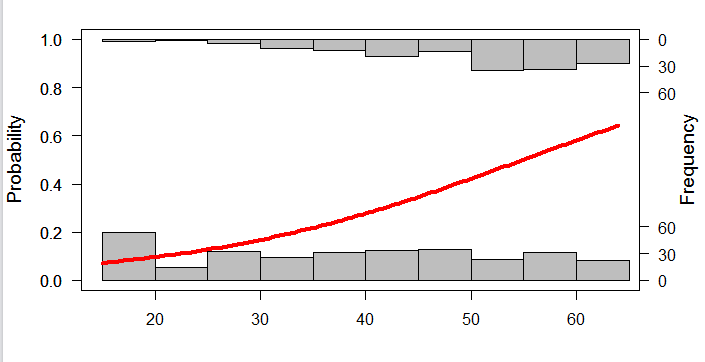
The following object is masked from ‘package:caret’:

sensitivity

Warning message:

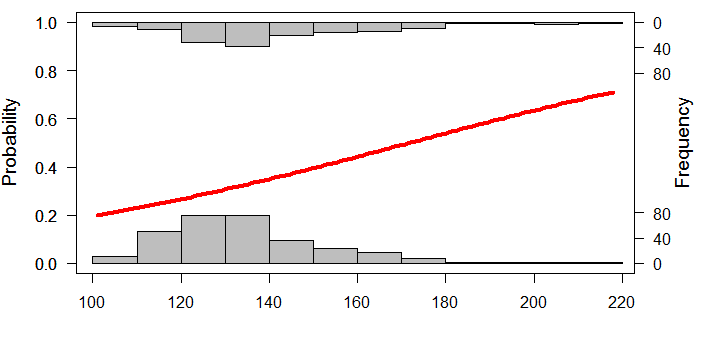
package ‘popbio’ was built under R version 3.3.2

> logi.hist.plot(sa$age,sa$chd,boxp=FALSE,type="hist",col="gray")



The below graph shows that increase in age will increase the changes of CHD

> logi.hist.plot(sa$sbp,sa$chd,boxp=FALSE,type="hist",col="gray")



The below graph shows that increase in sbp will increase the changes of CHD

2. Write a short summary of your results - what worked, what didn't, what happened?

We see that using Knn we got a good accuracy with k =19 with accuracy of 0.671.

But value shows that this method is classifying false positive (wrongly classified as having chd) more than true positive (actually having chd).

Also it classify more false negative (wrongly classified as not having chd) than true negative (actually not having chd).

For kpd with scaled data Eucledian method gets an accuracy of 0.684 with k=11, Minkowski methods get an accuracy of 0.6926 with k=15 and mahalanobis methods get an accuracy of 0.671 with k=11

For kpd with unscaled data Eucledian method gets an accuracy of 0.67 with k=19, Minkowski methods get an accuracy of 0.671 with k=15 and mahalanobis methods doesn not work as it just classify the data into true positives or false positives.

3. Compare to LDFA in predicting CHD based on all the quantitative inputs (Homework 2).

In this case LDFA prediction was 0.7294, which was higher than any of the method using kpd and knn. Both True positive and True negative were predicted better in LDFA.