lr

2023-02-20

library(dplyr) # is used for data manipulation

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2) # for visualization  
library(caTools) # for train/test split  
library(caret) # for cross-validation, etc.

## Loading required package: lattice

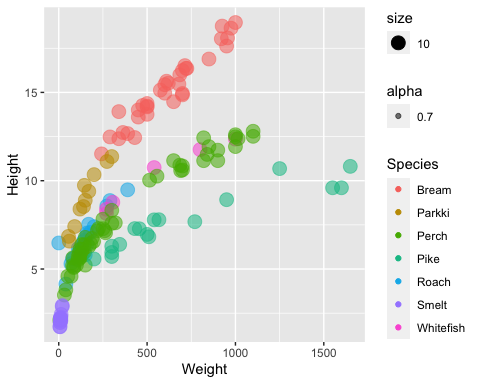
setwd('/Users/qinyiliu/Downloads/ml course')  
df <- read.csv('Fish.csv')  
head(df)

## Species Weight Length1 Length2 Length3 Height Width  
## 1 Bream 242 23.2 25.4 30.0 11.5200 4.0200  
## 2 Bream 290 24.0 26.3 31.2 12.4800 4.3056  
## 3 Bream 340 23.9 26.5 31.1 12.3778 4.6961  
## 4 Bream 363 26.3 29.0 33.5 12.7300 4.4555  
## 5 Bream 430 26.5 29.0 34.0 12.4440 5.1340  
## 6 Bream 450 26.8 29.7 34.7 13.6024 4.9274

any(is.na(df))

## [1] FALSE

ggplot(data=df, aes(x=Weight, y=Height)) +  
 geom\_point(aes(color=Species, size=10, alpha=0.7))

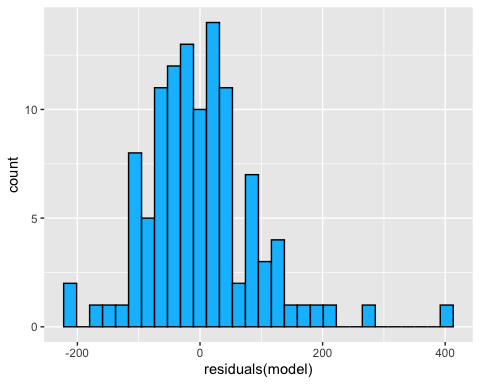


set.seed(42) # set random seed  
  
sampleSplit <- sample.split(Y=df$Weight, SplitRatio=0.7)  
trainSet <- subset(x=df, sampleSplit==TRUE)  
testSet <- subset(x=df, sampleSplit==FALSE)  
  
model <- lm(formula=Weight ~ ., data=trainSet)  
  
summary(model)

##   
## Call:  
## lm(formula = Weight ~ ., data = trainSet)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -215.89 -54.61 -5.17 43.84 398.39   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -906.264 163.753 -5.534 2.56e-07 \*\*\*  
## SpeciesParkki 119.896 92.780 1.292 0.1993   
## SpeciesPerch 163.519 152.037 1.076 0.2848   
## SpeciesPike -186.470 177.873 -1.048 0.2970   
## SpeciesRoach 103.074 118.739 0.868 0.3875   
## SpeciesSmelt 421.896 156.123 2.702 0.0081 \*\*   
## SpeciesWhitefish 126.796 120.745 1.050 0.2962   
## Length1 -57.474 44.056 -1.305 0.1951   
## Length2 57.483 54.485 1.055 0.2940   
## Length3 34.205 36.225 0.944 0.3473   
## Height 8.176 16.539 0.494 0.6222   
## Width -14.080 31.333 -0.449 0.6541   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 95 on 99 degrees of freedom  
## Multiple R-squared: 0.9383, Adjusted R-squared: 0.9315   
## F-statistic: 136.9 on 11 and 99 DF, p-value: < 2.2e-16

modelResiduals <- as.data.frame(residuals(model))   
  
ggplot(modelResiduals, aes(residuals(model))) +  
 geom\_histogram(fill='deepskyblue', color='black')

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



preds <- predict(model, testSet)  
  
modelEval <- cbind(testSet$Weight, preds)  
colnames(modelEval) <- c('Actual', 'Predicted')  
modelEval <- as.data.frame(modelEval)  
  
modelEval

## Actual Predicted  
## 1 242.0 284.147047  
## 2 290.0 334.776325  
## 4 363.0 436.395644  
## 7 500.0 482.364798  
## 13 500.0 528.017728  
## 16 600.0 563.541494  
## 17 700.0 600.601825  
## 21 575.0 645.596147  
## 23 620.0 676.981866  
## 24 680.0 706.847974  
## 28 714.0 755.890707  
## 30 1000.0 814.388249  
## 31 920.0 862.364460  
## 32 955.0 859.530406  
## 36 40.0 -177.997339  
## 39 87.0 49.523195  
## 44 150.0 128.428325  
## 46 160.0 181.065171  
## 47 140.0 145.088428  
## 49 169.0 249.954848  
## 54 272.0 361.517041  
## 56 270.0 349.102363  
## 62 55.0 -129.654511  
## 63 60.0 -100.899448  
## 65 120.0 56.110744  
## 68 170.0 133.885962  
## 85 125.0 136.910697  
## 94 145.0 197.734347  
## 95 150.0 202.047994  
## 99 188.0 264.996434  
## 105 265.0 363.343219  
## 106 250.0 361.978107  
## 107 250.0 388.577785  
## 112 840.0 661.270858  
## 113 685.0 727.585824  
## 120 850.0 880.131066  
## 127 1000.0 1009.156897  
## 128 1000.0 1013.443758  
## 131 300.0 353.733642  
## 134 345.0 450.005803  
## 135 456.0 606.327883  
## 137 540.0 632.356318  
## 139 567.0 729.535367  
## 144 1550.0 1318.813189  
## 146 6.7 -86.660378  
## 148 7.0 -60.793364  
## 153 9.9 -5.751781  
## 156 13.4 19.745701

mse <- mean((modelEval$Actual - modelEval$Predicted)^2)  
mse

## [1] 9198.647

rmse <- sqrt(mse)  
rmse

## [1] 95.90958

#10-fold cross-validation  
library(caret)  
set.seed(1560)  
ctrl <- trainControl(method = "cv", number = 10)  
cv.fit <- train(Weight~.,  
 data = trainSet,  
 preProcess = "range",  
 method = "lm",  
 metric = "RMSE",  
 control = ctrl)

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded  
  
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'control' will be disregarded

cv.fit$results

## intercept RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 TRUE 107.1045 0.9152069 78.82044 16.82156 0.02018916 8.865054

# Using 10-fold cross-validation, estimate a lasso regression. Which features survive the lasso.   
# How does it perform relative to the regression you ran previously?  
set.seed(1560)  
ctrl <- trainControl(method = "cv", number = 10)  
grid <- expand.grid(alpha = 1, lambda = seq(0,25,0.1))  
lasso.fit <- train(Weight~.,  
 data = trainSet,  
 preProcess = "range",  
 method = "glmnet",  
 metric = "RMSE",  
 control = ctrl,  
 tuneGrid = grid)  
lasso.fit$bestTune

## alpha lambda  
## 24 1 2.3

lasso.fit$results[24,]

## alpha lambda RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 24 1 2.3 104.764 0.9199777 75.93927 16.36208 0.01757888 9.25055

# Run a principal component regression and show the results  
library(pls)

##   
## Attaching package: 'pls'

## The following object is masked from 'package:caret':  
##   
## R2

## The following object is masked from 'package:stats':  
##   
## loadings

set.seed(6180)  
pcr.fit <- pcr(Weight~.,  
 data = trainSet,  
 scale = TRUE,  
 validation = "CV",  
 segments = 10)  
summary(pcr.fit)

## Data: X dimension: 111 11   
## Y dimension: 111 1  
## Fit method: svdpc  
## Number of components considered: 11  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 364.5 147.5 149.4 150.6 150.2 147.4 127.7  
## adjCV 364.5 147.2 149.0 150.2 149.6 146.8 127.2  
## 7 comps 8 comps 9 comps 10 comps 11 comps  
## CV 105.2 103.0 104.6 105.0 105.0  
## adjCV 104.7 102.3 103.9 104.2 104.1  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 45.18 58.46 70.32 80.46 90.16 97.10 99.68 99.93  
## Weight 84.33 84.34 84.49 84.96 85.67 89.23 93.03 93.64  
## 9 comps 10 comps 11 comps  
## X 99.99 100.00 100.00  
## Weight 93.65 93.79 93.83

validationplot(pcr.fit)

