PLS & CART

Marjete Vucinaj

2024-12-11

```
library(readr)
library(pls)
```

```
##
## Attaching package: 'pls'

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

```
library(magrittr)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     1.0.2
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::extract()   masks magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x purrr::set_names() masks magrittr::set_names()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
##
## The following object is masked from 'package:pls':
##
##     R2
```

```
library(rpart)
```

Response Variable.: pH

```
imputed_data <- read_csv("imputed_test_data.csv")
```

```
## Rows: 2447 Columns: 36
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (36): Carb.Volume, Fill.Ounces, PC.Volume, Carb.Pressure, Carb.Temp, PSC...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(imputed_data)
```

```
## # A tibble: 6 x 36
##   Carb.Volume Fill.Ounces PC.Volume Carb.Pressure Carb.Temp   PSC PSC.Fill
##         <dbl>       <dbl>     <dbl>         <dbl>     <dbl> <dbl>    <dbl>
## 1        5.34        24.0     0.263          68.2      141. 0.104     0.26
## 2        5.43        24.0     0.239          68.4      140. 0.124     0.22
## 3        5.29        24.1     0.263          70.8      145. 0.09      0.34
## 4        5.44        24.0     0.293          63        133. 0.129     0.42
## 5        5.49        24.3     0.111          67.2      137. 0.026     0.16
## 6        5.38        23.9     0.269          66.6      138. 0.09      0.24
## # i 29 more variables: PSC.CO2 <dbl>, Mnf.Flow <dbl>, Carb.Pressure1 <dbl>,
## #   Fill.Pressure <dbl>, Hyd.Pressure1 <dbl>, Hyd.Pressure2 <dbl>,
## #   Hyd.Pressure3 <dbl>, Hyd.Pressure4 <dbl>, Filler.Level <dbl>,
## #   Filler.Speed <dbl>, Temperature <dbl>, Usage.cont <dbl>, Carb.Flow <dbl>,
## #   Density <dbl>, MFR <dbl>, Balling <dbl>, Pressure.Vacuum <dbl>, PH <dbl>,
## #   Oxygen.Filler <dbl>, Bowl.Setpoint <dbl>, Pressure.Setpoint <dbl>,
## #   Air.Pressurer <dbl>, Alch.Rel <dbl>, Carb.Rel <dbl>, Balling.Lvl <dbl>, ...
```

```
# Split

train_index <- createDataPartition(imputed_data$PH, p = 0.75, list = FALSE)
train_data <- imputed_data[train_index, ]
test_data <- imputed_data[-train_index, ]
```

```
# pls can handle multicollinearity but just checking
correlation <- cor(train_data$PH, train_data[, -which(names(train_data) == "PH")])
print(correlation)
```

```
##      Carb.Volume Fill.Ounces PC.Volume Carb.Pressure  Carb.Temp         PSC
## [1,]  0.07218182 -0.09250504 0.0264231    0.05558605 0.01134131 -0.08631458
##        PSC.Fill     PSC.CO2   Mnf.Flow Carb.Pressure1 Fill.Pressure
## [1,] -0.0242662 -0.09720644 -0.4418839    -0.06036832    -0.2197768
##      Hyd.Pressure1 Hyd.Pressure2 Hyd.Pressure3 Hyd.Pressure4 Filler.Level
## [1,]   -0.08367108    -0.1987897    -0.2379096    -0.1315832    0.3180917
##      Filler.Speed Temperature Usage.cont Carb.Flow    Density        MFR
## [1,]  -0.06206353  -0.1488801 -0.3119293 0.1366277 0.08483373 -0.0723247
```

```
##        Balling Pressure.Vacuum Oxygen.Filler Bowl.Setpoint Pressure.Setpoint
## [1,] 0.0657022        0.219004     0.1726586     0.3389863        -0.3155336
##      Air.Pressurer  Alch.Rel  Carb.Rel Balling.Lvl Brand.CodeA Brand.CodeB
## [1,]   -0.01561297 0.1511893 0.1583297   0.1024894 -0.08977359   0.1001531
##      Brand.CodeC Brand.CodeD
## [1,]  -0.2940141   0.1818833
```

```
trainX <- train_data[, -which(names(train_data) == "PH")]
trainY <- train_data$PH

trainX <- as.data.frame(trainX)
trainY <- as.numeric(trainY)
```

```
plsTune_model <- train(
  x = trainX,
  y = trainY,
  method = "pls",
  tuneLength = 10,
  trControl = trainControl(method = "cv"),
)
# removed - preProc = c("center", "scale"); was already applied in imputation code
print(plsTune_model)
```

```
## Partial Least Squares
##
## 1837 samples
##   35 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1654, 1654, 1654, 1654, 1653, 1654, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE       Rsquared    MAE
##    1     0.1698915  0.03889131  0.1360835
##    2     0.1682757  0.05418183  0.1351559
##    3     0.1549418  0.19924680  0.1217946
##    4     0.1512552  0.23834257  0.1176840
##    5     0.1495613  0.25227642  0.1165574
##    6     0.1481872  0.26633955  0.1156638
##    7     0.1461027  0.28651904  0.1137971
##    8     0.1455007  0.29293936  0.1130008
##    9     0.1444273  0.30358775  0.1123713
##   10     0.1418244  0.32805595  0.1101039
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 10.
```

```
#suppress or address warning?
```

```
optimal_components <- 7
pls_final_model <- plsr(PH ~ ., data = train_data, ncomp = optimal_components)
```

```r
predictions <- predict(pls_final_model, newdata = test_data, ncomp = optimal_components)
rmse <- sqrt(mean((test_data$PH - predictions)^2))
r_squared <- cor(test_data$PH, predictions)^2
mae <- mean(abs(test_data$PH - predictions))

cat("Test RMSE:", rmse, "\n")
```

```
## Test RMSE: 0.1446069
```

```r
cat("Test R²:", r_squared, "\n")
```

```
## Test R²: 0.2718792
```

```r
cat("Test MAE:", mae, "\n")
```

```
## Test MAE: 0.1145993
```

CART: Regression: response variable is numerical and continuous.

```r
cart_model <- train(
  PH ~ .,
  data = train_data,
  method = "rpart",
  trControl = trainControl(method = "cv", number = 3, verboseIter = TRUE),
  tuneLength = 10
)
```

```
## + Fold1: cp=0.01515
## - Fold1: cp=0.01515
## + Fold2: cp=0.01515
## - Fold2: cp=0.01515
## + Fold3: cp=0.01515
## - Fold3: cp=0.01515
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
## Aggregating results
## Selecting tuning parameters
## Fitting cp = 0.0159 on full training set
```

```r
print(cart_model)
```

```
## CART
##
## 1837 samples
##   35 predictor
##
## No pre-processing
```

```
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 1226, 1224, 1224
## Resampling results across tuning parameters:
##
##   cp          RMSE       Rsquared   MAE
##   0.01515350  0.1352945  0.3921165  0.1041521
##   0.01587580  0.1352070  0.3921712  0.1042564
##   0.01644801  0.1359497  0.3853779  0.1053982
##   0.01727906  0.1366106  0.3790389  0.1055609
##   0.01930733  0.1373128  0.3726317  0.1058595
##   0.02246855  0.1405136  0.3424462  0.1090925
##   0.03362470  0.1424540  0.3234390  0.1117618
##   0.04199066  0.1466342  0.2843337  0.1149674
##   0.06305480  0.1523962  0.2253521  0.1191724
##   0.21056463  0.1613791  0.1875065  0.1271038
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.0158758.
```

```
print(cart_model$bestTune)
```

```
##          cp
## 2 0.0158758
```

```
best_cp <- cart_model$bestTune$cp

cart_final_model <- rpart(PH ~ ., data = train_data, method = "anova", control = rpart.control(cp = bes

cart_predictions <- predict(cart_final_model, newdata = test_data)

cart_rmse <- sqrt(mean((test_data$PH - cart_predictions)^2))
cart_r_squared <- cor(test_data$PH, cart_predictions)^2
cart_mae <- mean(abs(test_data$PH - cart_predictions))

cat("CART Test RMSE:", cart_rmse, "\n")
```

```
## CART Test RMSE: 0.1323511
```

```
cat("CART Test R²:", cart_r_squared, "\n")
```

```
## CART Test R²: 0.3903304
```

```
cat("CART Test MAE:", cart_mae, "\n")
```

```
## CART Test MAE: 0.1055604
```

Question: We wait to predict on the best model?