R ile Regresyon Analizleri

Dr. Öğr. Gözde Ertürk Zararsız

Mayıs 15, 2022

Contents

Kurs Öncesi Gereksinimler:

- R 4.2.0 sürümünün kullanılması önerilir.
- RStudio GUI Desktop
- MAC Kullanıcılar için XQuartz güncel sürümü.
- İhtiyaç duyulan R kütüphaneleri: `caret`, `ggplot2`, `dplyr`, `lattice`, `doParallel`, `magrittr`, `pls`, `grid`

Gerekli kütüphaneler yüklendikten sonra aşağıdaki şekilde çağırılır. Ayrıca bu uygulama için gerekli diğer fonksiyonların çağırılmasına ihtiyaç vardır.

```
library(caret)
library(magrittr)
library(dplyr)
library(lattice)

source("../functions/findOutliers.R")
source("../functions/diagnosticPlots.R")
source("../functions/multiplot.R")
source("../functions/rmse.R")
```

Giriş

`caret` paketi regresyon (Regression) ve sınıflama (Classification) amacı ile kullanılabilen çok sayıda yöntemi içinde barındırmaktadır. Bu yöntemler yalnızca ***sınıflama*** (discriminant analysis, logistic regression, vb.), yalnızca ***regresyon*** (linear regression, principal component analysis, ridge regression, vb.) veya her iki amaç için kullanılabilmektedir (genel linear modeller, support vector machines, random forests, vb.). `caret` içerisinde yer alan modellerin listesine https://topepo.github.io/caret/index.html üzerinden erişilebilir.

Bu bölümde yalnızca ***regresyon*** tabanlı modeller üzerinde durulacak ve farklı veri setleri üzerinde uygulamalar yapılarak yöntemler arası karşılaştırmalar yapılacaktır.

SOCR (Statistics Online Computational Resource) Body Fat Data

`SOCR Body Fat Data` Kaliforniya Üniversitesi, Los Angeles. Veri seti 252 gözlem ve 15 değişkenden oluşmaktadır. İnsanların vücut yağ yüzdelerinin kestirilebilmesi amacı ile toplanmış bir dizi değişkeni içerisinde barındırmaktadır.

- **UnderwaterDensity** Su altı tartımı ile belirlenen yoğunluk
- **BodyFatSiriEqu** Siri (1956) eşitliği ile hesaplanan yağ yüzdesi
- **Age** Yaş (yıl)
- **Weight** Ağırlık (kg)
- **Height** Boy (cm)
- **NeckCircumf** Boyun çevresi (cm)
- **ChestCircumf** Göğüs çevresi (cm)
- **Abdomen2Circumf**Karın çevresi (cm)
- **HipCircumf** Kalça çevresi (cm)
- **ThighCircumf** Uyluk çevresi (cm)
- **KneeCircumf** Diz çevresi (cm)
- **AnkleCircumf** Ayak bileği çevresi (cm)
- **ExtendBicepsCircumf** Pazı çevresi (cm)
- **ForearmCircumf** Ön kol çevresi (cm)
- **WristCircumf** El bileği çevresi (cm)

Veri Önişleme

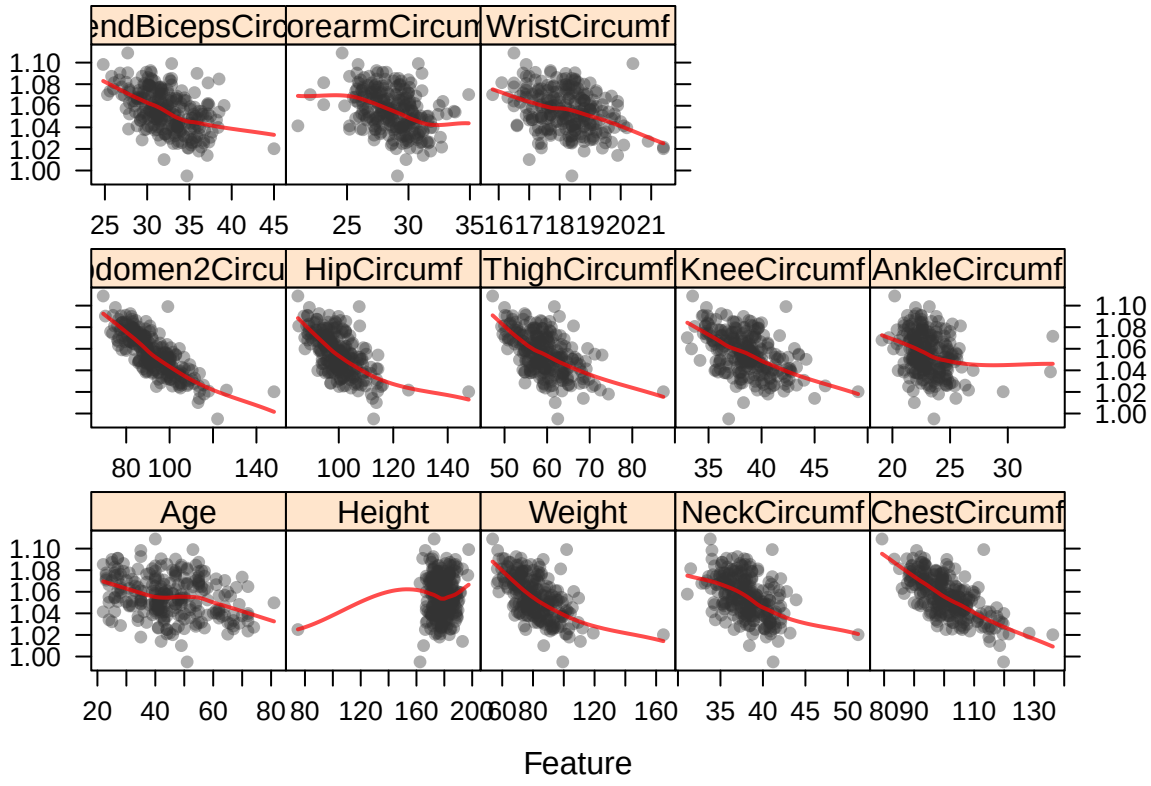
```
bfd <- read.table("../data/bodyfat.txt", header = TRUE)
bfd <- bfd[,-2]
head(bfd)
```

```
##   UnderwaterDensity Age  Height   Weight NeckCircumf ChestCircumf
## 1            1.0708  23 172.085 69.96662        36.2         93.1
## 2            1.0853  22 183.515 78.58488        38.5         93.6
## 3            1.0414  22 168.275 69.85322        34.0         95.8
## 4            1.0751  26 183.515 83.80119        37.4        101.8
## 5            1.0340  24 180.975 83.57439        34.4         97.3
## 6            1.0502  24 189.865 95.36780        39.0        104.5
##   Abdomen2Circumf HipCircumf ThighCircumf KneeCircumf AnkleCircumf
## 1            85.2       94.5         59.0        37.3         21.9
## 2            83.0       98.7         58.7        37.3         23.4
## 3            87.9       99.2         59.6        38.9         24.0
## 4            86.4      101.2         60.1        37.3         22.8
## 5           100.0      101.9         63.2        42.2         24.0
## 6            94.4      107.8         66.0        42.0         25.6
##   ExtendBicepsCircumf ForearmCircumf WristCircumf
## 1                32.0           27.4         17.1
## 2                30.5           28.9         18.2
## 3                28.8           25.2         16.6
## 4                32.4           29.4         18.2
## 5                32.2           27.7         17.7
## 6                35.7           30.6         18.8
```

Yanıt değişkeni ile bağımsız değişkenler arası ilişkiler (Saçılım Grafiği)

Öncelikle veri setinin görsel olarak ön incelemesi gerçekleştirilmelidir. Değişkenler arasında ilişki olup olmadığı, varsa bu ilişkinin doğrusal olup olmadığı araştırılmalıdır. Ayrıca, veride herhangi bir aşırı ya da aykırı değere sahip gözlem varsa görsel olarak da tespit edilebilir.

```
theme1 <- trellis.par.get()
theme1$plot.symbol$col = rgb(.2, .2, .2, .4)
theme1$plot.symbol$pch = 16
theme1$plot.line$col = rgb(1, 0, 0, .7)
theme1$plot.line$lwd <- 2
trellis.par.set(theme1)

featurePlot(x = bfd[, -1], y = bfd[, 1], plot = "scatter", type = c("p", "smooth"), span = 0.7)
```



Aşırı Değerlerin Tespit Edilmesi ve Veriden Çıkarılması

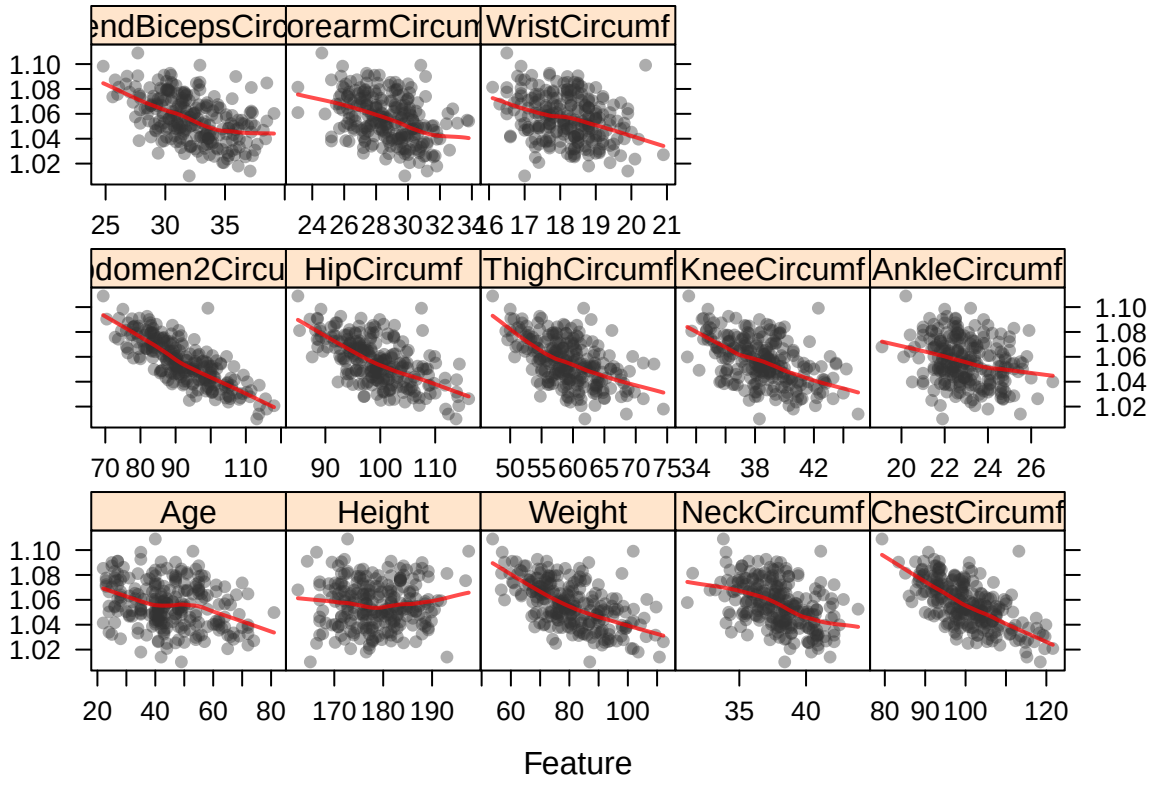
Veride aşırı değerlerin olduğu görülmektedir. Bu değerler `findOutliers(...)` fonksiyonu yardımıyla tespit edilebilir. Aşırı değerlerin tespiti ve veriden çıkarılması aşağıdaki kodlar yardımıyla gerçekleştirilebilir.

```
tmp <- lapply(bfd, findOutliers, abs.cutoff.z = 3)
out.idx <- unique(unlist(tmp)) ## Aykırı gözlemlerin sıra numaraları.
if (!is.null(out.idx)){
  bfd <- bfd[-out.idx, ]
}
out.idx
```

```
##  [1] 216  42  39  41 244  31  86 159 175 226
```

Aşırı değerler çıkarıldıktan sonra veri tekrar görsel olarak incelenir.

```
theme1 <- trellis.par.get()
theme1$plot.symbol$col = rgb(.2, .2, .2, .4)
theme1$plot.symbol$pch = 16
theme1$plot.line$col = rgb(1, 0, 0, .7)
theme1$plot.line$lwd <- 2
trellis.par.set(theme1)

featurePlot(x = bfd[, -1], y = bfd[, 1], plot = "scatter", type = c("p", "smooth"), span = 0.7)
```



Sıfıra yakın varyanslı ve yüksek düzeyde korele olan değişkenlerin araştırılması

Regresyon modelleme aşamasına geçilmeden önce sıfıra yakın varyanslı değişkenlerin ve yüksek korelasyonlu değişkenlerin varlığı araştırılabilir. Öncelikle veri setinde sıfıra yakın varyanslı değişkenlerin olup olmadığı aşağıdaki biçimde incelenebilir:

```
nzv <- nearZeroVar(bfd, saveMetrics= TRUE, freqCut = 10, uniqueCut = 10)
nzv
```

```
##                       freqRatio percentUnique zeroVar   nzv
## UnderwaterDensity      1.333333      87.19008   FALSE FALSE
## Age                    1.307692      21.07438   FALSE FALSE
## Height                 1.166667      19.42149   FALSE FALSE
## Weight                 1.000000      77.68595   FALSE FALSE
## NeckCircumf            1.200000      35.53719   FALSE FALSE
## ChestCircumf           1.000000      68.59504   FALSE FALSE
## Abdomen2Circumf        1.333333      73.55372   FALSE FALSE
## HipCircumf             1.166667      60.33058   FALSE FALSE
## ThighCircumf           1.400000      54.54545   FALSE FALSE
## KneeCircumf            1.285714      35.95041   FALSE FALSE
## AnkleCircumf           1.000000      23.14050   FALSE FALSE
## ExtendBicepsCircumf    1.000000      41.32231   FALSE FALSE
## ForearmCircumf         1.000000      30.16529   FALSE FALSE
## WristCircumf           1.125000      17.35537   FALSE FALSE
```

Görüldüğü üzere veride varyansı sıfıra yakın bir değişkene rastlanmamıştır. `caret` paketinde bulunan `findCorrelation` fonksiyonu kullanılarak istenilen kesim noktasına göre yüksek ilişkili değişkenler tespit edilebilir. Bu fonksiyon, her değişken için ortalama mutlak korelasyonu hesaplar ve en büyük ortalama mutlak korelasyona sahip değişkeni veriden çıkarır.

```
corr <- cor(bfd)
highlyCorr <- findCorrelation(corr, cutoff = 0.95)
highlyCorr
```

```
## integer(0)
```

Veride yüksek düzeyde korele yapıda değişkenlere rastlanmamıştır.

Eğitim ve test setlerinin oluşturulması

Model oluşturmak için öncelikle algoritmanın eğitileceği eğitim (training) seti ve eğitilen modelin performansının test edileceği test seti oluşturulmalıdır. Eğitim ve test setlerinin hangi oranlarda oluşturulacağına ilişikin standart bir oran bulunmamakla birlikte %80 (eğitim) - %20 (test), %70 (eğitim) / %30 (test) sıklıkla kullanılan oranlardır. Bu oranlar veri setinin büyüklüğüne göre belirlenmelidir.

`caret` paketinde bulunan createDataPartition fonksiyonu kullanılarak eğitim ve test setleri belirlenebilir.

```
set.seed(1881)
inTrain <- createDataPartition(y = bfd$UnderwaterDensity, ## sınıf değişkeni
                               p = 0.80, ## eğitim seti oranı
                               list = FALSE,
                               times = 1)

head(inTrain)
```

```
##      Resample1
## [1,]         2
## [2,]         4
## [3,]         5
## [4,]         6
## [5,]         7
## [6,]         8
```

```
train <- bfd[inTrain, ]
test <- bfd[-inTrain, ]
dim(train)
```

```
## [1] 195  14
```

```
dim(test)
```

```
## [1] 47 14
```

Verinin standartlaştırılması

Değişkenlerin standartlaştırılarak aynı birime indirgenmeleri model performansını arttırabilir. `caret` paketinde bulunan `preProcess(...)` fonksiyonu kullanılarak z-standartlaştırması yapılabilir. Burada dikkat edilmesi gereken nokta standartlaştırmanın eğitim veri setine uygulanması ve test setine standartlaştırma uygulanırken eğitim setinden elde edilen ortalama ve standart sapma değerlerinin kullanılmasıdır.

```
preProcValues <- preProcess(train, method = c("center", "scale"))
train.std <- predict(preProcValues, train)
test.std <- predict(preProcValues, test)
```

Önemli değişkenlerin seçilmesi

Wrapper yöntemleri ile iteratif bir süreç ile çoklu model kullanılarak model performansını maksimize eden en uygun değişken kombinasyonu bulunmaya çalışılır. Bu yöntemlerden biri recursive feature elimination yöntemidir. Bu yöntem kullanılarak en önemli değişken kümesi aşağıdaki şekilde belirlenebilir.

```
set.seed(1881)
ctrl <- rfeControl(functions = rfFuncs,
                   method = "repeatedcv",
                   number = 5,
                   repeats = 1,
                   verbose = FALSE)

featureSelect <- rfe(x=train.std[,-1], y=train.std[,1],
                     sizes = c(1:30),
                     metric = "RMSE",
                     rfeControl = ctrl)

featureSelect
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (5 fold, repeated 1 times)
##
## Resampling performance over subset size:
##
##  Variables   RMSE Rsquared    MAE  RMSESD RsquaredSD   MAESD Selected
##          1 0.6399   0.6000 0.5117 0.08562    0.09834 0.07190
##          2 0.5912   0.6534 0.4855 0.11505    0.11545 0.10200
##          3 0.5888   0.6615 0.4846 0.09016    0.06884 0.06591
##          4 0.6070   0.6423 0.4981 0.10000    0.07817 0.07426
##          5 0.6202   0.6261 0.5089 0.10943    0.08634 0.08239
##          6 0.5937   0.6535 0.4848 0.09527    0.07038 0.07247
##          7 0.5954   0.6585 0.4848 0.09814    0.07504 0.07901
##          8 0.6001   0.6508 0.4875 0.09858    0.07916 0.07718
##          9 0.5824   0.6708 0.4737 0.09625    0.07871 0.07688        *
##         10 0.5851   0.6707 0.4748 0.09454    0.07445 0.07333
##         11 0.5878   0.6663 0.4789 0.10248    0.08097 0.07320
##         12 0.5856   0.6685 0.4718 0.10072    0.08168 0.07068
##         13 0.5833   0.6714 0.4713 0.10094    0.08336 0.07198
```

```
##
## The top 5 variables (out of 9):
##    Abdomen2Circumf, ChestCircumf, HipCircumf, Weight, ThighCircumf
```

```
bestSubset <- featureSelect$optVariables
bestSubset
```

```
## [1] "Abdomen2Circumf"     "ChestCircumf"        "HipCircumf"
## [4] "Weight"              "ThighCircumf"        "Height"
## [7] "Age"                 "NeckCircumf"         "ExtendBicepsCircumf"
```

```
vars <- c("UnderwaterDensity", bestSubset)
tr <- train.std[,vars]
ts <- test.std[,vars]
```

Regresyon modellerinin uygulanması

Veri ön işleme, eğitim ve test setlerinin belirlenmesi ve değişken seçimi adımlarının ardından model kurma adımına geçilebilir. `caret` paketinde bulunan train fonksiyonu kullanılarak model eğitme işlemleri gerçekleştirilir. `train` fonksiyonu kullanılarak;

- parametre optimizasyonunun model performansı üzerindeki etkileri değerlendirilebilir,
- optimal model parametreleri seçilebilir,
- eğitim veri seti kullanılarak model performansı kestirilebilir,
- `train` fonksiyonu içerisinde yüzlerce farklı makine öğrenimi algoritması eğitilebilir.

Doğrusal regresyon modelleri (linear regression, LM) algoritmasını kullanarak eğitim setimizi eğitelim:

```
fitLM <- train(x = tr[,-1], y = tr[,1], method = "lm", tuneLength = 10,
               trControl = trainControl(method = "repeatedcv", number = 5, repeats = 3))
fitLM
```

```
## Linear Regression
##
## 195 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 157, 156, 155, 156, 156, 155, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.5338756  0.7205651  0.4392313
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Radyal tabanlı çekirdek fonksiyon ile destek vektör makineleri (support vector machines, SVM) algoritmasını kullanarak eğitim setimizi eğitelim:

```
fitSVM <- train(x = tr[,-1], y = tr[,1], method = "svmRadial", tuneLength = 10,
                trControl = trainControl(method = "repeatedcv", number = 5, repeats = 3))
fitSVM
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 195 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 156, 156, 157, 155, 156, 155, ...
## Resampling results across tuning parameters:
##
##   C       RMSE       Rsquared   MAE
##     0.25  0.6416410  0.6181545  0.5134242
##     0.50  0.6073769  0.6417179  0.4893495
##     1.00  0.5875268  0.6570546  0.4726548
##     2.00  0.5784057  0.6647933  0.4659876
##     4.00  0.5877333  0.6534852  0.4743000
##     8.00  0.6161316  0.6234857  0.5019510
##    16.00  0.6572556  0.5843128  0.5366746
##    32.00  0.6990109  0.5505930  0.5697553
##    64.00  0.7345507  0.5223667  0.5972148
##   128.00  0.7703944  0.5005949  0.6218717
##
## Tuning parameter 'sigma' was held constant at a value of 0.1275977
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were sigma = 0.1275977 and C = 2.
```

Rasgele orman (random forests, RF) algoritmasını kullanarak eğitim setimizi eğitelim:

```
fitRF <- train(x = tr[,-1], y = tr[,1], method = "rf", tuneLength = 10,
               trControl = trainControl(method = "repeatedcv", number = 5, repeats = 3))
```

```
## note: only 8 unique complexity parameters in default grid. Truncating the grid to 8 .
```

```
fitRF
```

```
## Random Forest
##
## 195 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 156, 156, 155, 157, 156, 156, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##   2     0.5895839  0.6592180  0.4767874
##   3     0.5762053  0.6729850  0.4632743
```

```
##   4     0.5683124  0.6818700  0.4563938
##   5     0.5642185  0.6859441  0.4549075
##   6     0.5648080  0.6858117  0.4549246
##   7     0.5644924  0.6861718  0.4556683
##   8     0.5636949  0.6870987  0.4547465
##   9     0.5636284  0.6876906  0.4550704
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 9.
```

Lasso algoritmasını kullanarak eğitim setimizi eğitelim:

```
fitLASSO <- train(x = tr[,-1], y = tr[,1], method = "lasso", tuneLength = 10,
                  trControl = trainControl(method = "repeatedcv", number = 5, repeats = 3))
fitLASSO
```

```
## The lasso
##
## 195 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 155, 156, 155, 158, 156, 156, ...
## Resampling results across tuning parameters:
##
##   fraction   RMSE       Rsquared   MAE
##   0.1000000  0.8263772  0.6932449  0.6795201
##   0.1888889  0.6974949  0.6932449  0.5733995
##   0.2777778  0.6032995  0.6932130  0.4961501
##   0.3666667  0.5588136  0.7004720  0.4600828
##   0.4555556  0.5407866  0.7136975  0.4410272
##   0.5444444  0.5356875  0.7183088  0.4354345
##   0.6333333  0.5353459  0.7186832  0.4362043
##   0.7222222  0.5349932  0.7194006  0.4369227
##   0.8111111  0.5345467  0.7201361  0.4379367
##   0.9000000  0.5342603  0.7208389  0.4387639
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was fraction = 0.9.
```

Model performanslarının değerlendirilmesi ve karşılaştırılması

`caret` paketi içerisinde yer alan `predict` fonksiyonu kullanılarak test verilerinin sınıf değişkenleri kestirilebilir. Elde edilen kestirimler kullanılarak tüm modeller için RMSE ölçüsü aşağıdaki gibi hesaplanabilir:

```
predLM <- predict(fitLM, newdata = ts[,-1])
predSVM <- predict(fitSVM, newdata = ts[,-1])
predRF <- predict(fitRF, newdata = ts[,-1])
predLASSO <- predict(fitLASSO, newdata = ts[,-1])

RMSE <- data.frame(rmseLM = rmse(predLM, ts[,1]), rmseRF = rmse(predRF, ts[,1]),
```

```
                    rmseSVM = rmse(predSVM, ts[,1]), rmseLASSO = rmse(predLASSO, ts[,1]))
print(RMSE)
```

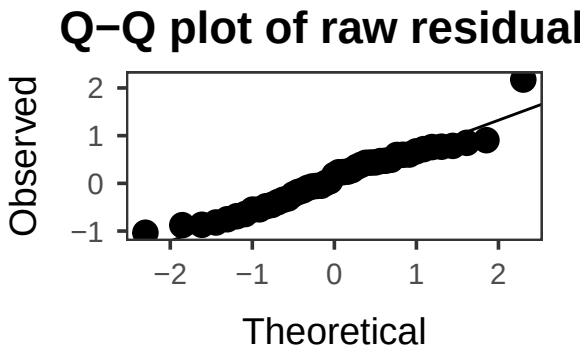
```
##      rmseLM    rmseRF   rmseSVM rmseLASSO
## 1 0.6284304 0.7326029 0.7453147 0.6322011
```

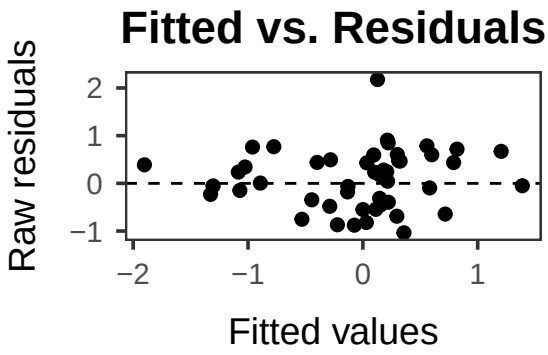
En düşük RMSE istatistiği, dolayısıyle en iyi performans doğrusal regresyon modelleri için elde edilmiştir.

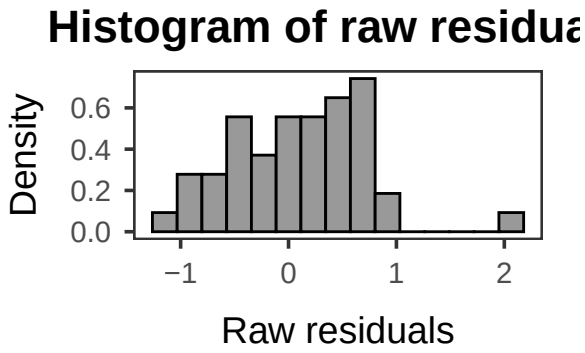
Model yeterliliği ve artıkların analizi

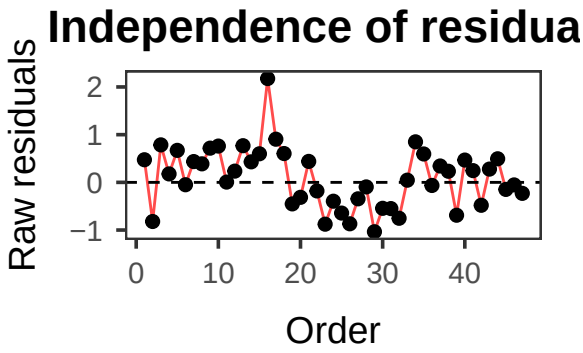
En iyi performansın elde edildiği doğrusal regresyon modelleri için model yeterliliği incelenmeli ve artıklar analiz edilmelidir. Bu amaçla aşağıdaki kodlar kullanılarak model yeterliliği ve artıkların grafikleri elde edilebilir:

```
model <- fitLM
res.train <- data.frame(Actual = ts[,1], Predicted = predLM)

res.train <- res.train %>%
  mutate(Residuals = Actual - Predicted,
         Std.Residuals = Residuals / sd(Residuals))

res.train <- data.frame(ResidID = 1:nrow(res.train), res.train)

plots <- diagnosticPlots(res.train)
multiplot(plotlist = plots, cols = 2)
```










```
sessionInfo()
```

```
## R version 4.2.0 (2022-04-22 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Turkish_Turkey.utf8  LC_CTYPE=Turkish_Turkey.utf8
## [3] LC_MONETARY=Turkish_Turkey.utf8 LC_NUMERIC=C
## [5] LC_TIME=Turkish_Turkey.utf8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] dplyr_1.0.9     magrittr_2.0.3  caret_6.0-92    lattice_0.20-45
## [5] ggplot2_3.3.6
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.8.3         lubridate_1.8.0      listenv_0.8.0
##  [4] class_7.3-20         digest_0.6.29        ipred_0.9-12
```

```
##  [7] foreach_1.5.2        utf8_1.2.2           parallelly_1.31.1
## [10] R6_2.5.1             plyr_1.8.7           lars_1.3
## [13] stats4_4.2.0         hardhat_0.2.0        evaluate_0.15
## [16] pillar_1.7.0         rlang_1.0.2          data.table_1.14.2
## [19] rstudioapi_0.13      kernlab_0.9-30       rpart_4.1.16
## [22] Matrix_1.4-1         rmarkdown_2.14       labeling_0.4.2
## [25] splines_4.2.0        gower_1.0.0          stringr_1.4.0
## [28] munsell_0.5.0        compiler_4.2.0       xfun_0.31
## [31] pkgconfig_2.0.3      globals_0.15.0       htmltools_0.5.2
## [34] nnet_7.3-17          tidyselect_1.1.2     tibble_3.1.7
## [37] prodlim_2019.11.13   codetools_0.2-18     randomForest_4.7-1
## [40] fansi_1.0.3          future_1.25.0        crayon_1.5.1
## [43] withr_2.5.0          MASS_7.3-56          recipes_0.2.0
## [46] ModelMetrics_1.2.2.2 nlme_3.1-157         gtable_0.3.0
## [49] lifecycle_1.0.1      pROC_1.18.0          scales_1.2.0
## [52] future.apply_1.9.0   cli_3.3.0            stringi_1.7.6
## [55] farver_2.1.0         reshape2_1.4.4       timeDate_3043.102
## [58] ellipsis_0.3.2       generics_0.1.2       vctrs_0.4.1
## [61] lava_1.6.10          iterators_1.0.14     tools_4.2.0
## [64] glue_1.6.2           purrr_0.3.4          elasticnet_1.3
## [67] parallel_4.2.0       fastmap_1.1.0        survival_3.3-1
## [70] yaml_2.3.5           colorspace_2.0-3     knitr_1.39
```