Chapter 18: Regression Models in Machine Learning

Ram Gopal, Dan Philps, and Tillman Weyde

2022

Contents

Load packages

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

Get the data and pre-process

Read data

```
dataset <- read.csv('diabetes.csv', header = TRUE)
```

Partition Data

```
N = nrow(dataset)
cut1 = floor(0.6*N)
cut2 = floor(0.8*N)
index = sample(1:N)
train_index = index[1:cut1]
val_index = index[(cut1+1):cut2]
test_index = index[(cut2+1):N]
df_train = dataset[train_index,]
df_val = dataset[val_index,]
df_test = dataset[test_index,]
sapply(list(df_train,df_val,df_test),nrow)
```

```
## [1] 265  88  89
```

Regression Model and predict

```
df_train[-11] = scale(df_train[-11])
lr = caret::train(target ~ ., method='lm',data = df_train)
lr$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
```

```
## Coefficients:
## (Intercept)         age         sex         bmi          bp          s1
##      155.37        1.44       -9.05       26.22       14.15      -39.36
##          s2          s3          s4          s5          s6
##       23.39        7.00       11.43       30.25        4.46
```

```
train_pred = predict(lr,newdata = df_train)
val_pred = predict(lr,newdata = scale(df_val[-11]))
test_pred = predict(lr,newdata = scale(df_test[-11]))
```

Compute functions for Residual Mean,MSE, RMSE and R_2

```
rm <- function(actual,pred) {
  return(mean(abs(actual-pred)))
}

mse <- function(actual,pred) {
  return(mean((pred-actual)^2))
}

rmse <- function(actual,pred) {
  return(mse(pred,actual)^0.5)
}

R_2 <- function(actual,pred){
  SST = sum((actual-mean(actual))^2)
  SSE = sum((actual-pred)^2)
  return(1-(SSE/SST))
}
```

```
res = data.frame()
w = rm(df_train$target,train_pred)
x = mse(df_train$target,train_pred)
y = rmse(df_train$target,train_pred)
z = R_2(df_train$target,train_pred)
res = rbind(res,c("Train",w,x,y,z))
w = rm(df_val$target,val_pred)
x = mse(df_val$target,val_pred)
y = rmse(df_val$target,val_pred)
z = R_2(df_val$target,val_pred)
res = rbind(res,c("Validation",w,x,y,z))
w = rm(df_test$target,test_pred)
x = mse(df_test$target,test_pred)
y = rmse(df_test$target,test_pred)
z = R_2(df_test$target,test_pred)
res = rbind(res,c("Test",w,x,y,z))
colnames(res) = c("Data","Residual Mean","MSE","RMSE","R_2")
# Following converts appropriate columns to numeric type
res[-1] = lapply(res[-1], FUN = function(y){as.numeric(y)})
# Following rounds numeric values to 4 digits
```

```
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
res
```

```
##         Data Residual Mean  MSE  RMSE    R_2
## 1      Train         43.84 2909 53.94 0.4894
## 2 Validation         46.56 3395 58.27 0.5146
## 3       Test         41.87 2421 49.20 0.5587
```

Polynomial

- Write formula for intercept, raw features, squared features,and interactions

```
# Degree 2 polynomial feature generation function

pf2_transform <- function(df, target_name='target') {
  formula_pf2 <- as.formula(paste(target_name, '~ .^2 +',
                                  paste('poly(',
                                        colnames(df)[-c(1)],
                                       ',2, raw=TRUE)[, 2]',
                                        collapse = ' + ')
                                  )
                            )
   output <- model.matrix(formula_pf2, data = df)
  # Rewrite column names for readability
   colnames_pf2 <- c("1",
                    colnames(df)[-1],              # exclude target
                    paste0(colnames(df)[-1],"^2"), # include squares
                    colnames(output)[-(1:(length(df)*2-1))]) # include interactions
    colnames(output) <- colnames_pf2
  # Convert to dataframe
  output_df <- data.frame(output)
  # Exclude intercept column
  output_df[,1] <- NULL
  return(output_df)
}
```

Create data sets with polynomials for training, validation and test sets.

- Create the training data set

```
train_sc_pf2 <- pf2_transform(df_train,target_name = "target")
train_sc_pf2$target= df_train$target
train_sc_pf2 = train_sc_pf2[-20]
print(colnames(train_sc_pf2))
```

```
##  [1] "sex"     "bmi"     "bp"      "s1"      "s2"      "s3"      "s4"
##  [8] "s5"      "s6"      "target"  "sex.2"   "bmi.2"   "bp.2"    "s1.2"
## [15] "s2.2"    "s3.2"    "s4.2"    "s5.2"    "s6.2"    "age.sex" "age.bmi"
## [22] "age.bp"  "age.s1"  "age.s2"  "age.s3"  "age.s4"  "age.s5"  "age.s6"
## [29] "sex.bmi" "sex.bp"  "sex.s1"  "sex.s2"  "sex.s3"  "sex.s4"  "sex.s5"
```

```
## [36] "sex.s6"   "bmi.bp"   "bmi.s1"   "bmi.s2"   "bmi.s3"   "bmi.s4"   "bmi.s5"
## [43] "bmi.s6"   "bp.s1"    "bp.s2"    "bp.s3"    "bp.s4"    "bp.s5"    "bp.s6"
## [50] "s1.s2"    "s1.s3"    "s1.s4"    "s1.s5"    "s1.s6"    "s2.s3"    "s2.s4"
## [57] "s2.s5"    "s2.s6"    "s3.s4"    "s3.s5"    "s3.s6"    "s4.s5"    "s4.s6"
## [64] "s5.s6"
```

- Prepare the validation and test sets

```
df_val[-11]= scale(df_val[-11])
val_sc_pf2 <- pf2_transform(df_val,target_name = "target")
val_sc_pf2$target = df_val$target
val_sc_pf2 = val_sc_pf2[-20]
print(colnames(val_sc_pf2))
```

```
##  [1] "sex"      "bmi"      "bp"       "s1"       "s2"       "s3"       "s4"
##  [8] "s5"       "s6"       "target"   "sex.2"    "bmi.2"    "bp.2"     "s1.2"
## [15] "s2.2"     "s3.2"     "s4.2"     "s5.2"     "s6.2"     "age.sex"  "age.bmi"
## [22] "age.bp"   "age.s1"   "age.s2"   "age.s3"   "age.s4"   "age.s5"   "age.s6"
## [29] "sex.bmi"  "sex.bp"   "sex.s1"   "sex.s2"   "sex.s3"   "sex.s4"   "sex.s5"
## [36] "sex.s6"   "bmi.bp"   "bmi.s1"   "bmi.s2"   "bmi.s3"   "bmi.s4"   "bmi.s5"
## [43] "bmi.s6"   "bp.s1"    "bp.s2"    "bp.s3"    "bp.s4"    "bp.s5"    "bp.s6"
## [50] "s1.s2"    "s1.s3"    "s1.s4"    "s1.s5"    "s1.s6"    "s2.s3"    "s2.s4"
## [57] "s2.s5"    "s2.s6"    "s3.s4"    "s3.s5"    "s3.s6"    "s4.s5"    "s4.s6"
## [64] "s5.s6"
```

```
df_test[-11]= scale(df_test[-11])
test_sc_pf2 <- pf2_transform(df_test,target_name = "target")
test_sc_pf2$target = df_test$target
test_sc_pf2 = test_sc_pf2[-20]
print(colnames(test_sc_pf2))
```

```
##  [1] "sex"      "bmi"      "bp"       "s1"       "s2"       "s3"       "s4"
##  [8] "s5"       "s6"       "target"   "sex.2"    "bmi.2"    "bp.2"     "s1.2"
## [15] "s2.2"     "s3.2"     "s4.2"     "s5.2"     "s6.2"     "age.sex"  "age.bmi"
## [22] "age.bp"   "age.s1"   "age.s2"   "age.s3"   "age.s4"   "age.s5"   "age.s6"
## [29] "sex.bmi"  "sex.bp"   "sex.s1"   "sex.s2"   "sex.s3"   "sex.s4"   "sex.s5"
## [36] "sex.s6"   "bmi.bp"   "bmi.s1"   "bmi.s2"   "bmi.s3"   "bmi.s4"   "bmi.s5"
## [43] "bmi.s6"   "bp.s1"    "bp.s2"    "bp.s3"    "bp.s4"    "bp.s5"    "bp.s6"
## [50] "s1.s2"    "s1.s3"    "s1.s4"    "s1.s5"    "s1.s6"    "s2.s3"    "s2.s4"
## [57] "s2.s5"    "s2.s6"    "s3.s4"    "s3.s5"    "s3.s6"    "s4.s5"    "s4.s6"
## [64] "s5.s6"
```

- run the regression models

```
lr = caret::train(target ~ ., method='lm',data = train_sc_pf2)
train_pred = predict(lr,newdata = train_sc_pf2)
val_pred = predict(lr,newdata = val_sc_pf2)
test_pred = predict(lr,newdata = test_sc_pf2)
```

- Evaluate the results

```r
res = data.frame()
w = rm(df_train$target,train_pred)
x = mse(df_train$target,train_pred)
y = rmse(df_train$target,train_pred)
z = R_2(df_train$target,train_pred)
res = rbind(res,c("Train",w,x,y,z))
w = rm(df_val$target,val_pred)
x = mse(df_val$target,val_pred)
y = rmse(df_val$target,val_pred)
z = R_2(df_val$target,val_pred)
res = rbind(res,c("Validation",w,x,y,z))
w = rm(df_test$target,test_pred)
x = mse(df_test$target,test_pred)
y = rmse(df_test$target,test_pred)
z = R_2(df_test$target,test_pred)
res = rbind(res,c("Test",w,x,y,z))
colnames(res) = c("Data","Residual Mean","MSE","RMSE","R_2")
# Following converts appropriate columns to numeric type
res[-1] = lapply(res[-1], FUN = function(y){as.numeric(y)})
# Following rounds numeric values to 4 digits
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
res
```

```
##         Data Residual Mean   MSE  RMSE     R_2
## 1      Train          37.6  2219  47.1  0.6107
## 2 Validation         109.5 17122 130.9 -1.4480
## 3       Test         136.1 30422 174.4 -4.5463
```

```r
str(train_sc_pf2)
```

```
## 'data.frame':    265 obs. of  64 variables:
##  $ sex    : num  -0.7984 -0.0889 0.1475 0.2264 1.4089 ...
##  $ bmi    : num  1.009 -0.987 -0.987 1.009 1.009 ...
##  $ bp     : num  -1.246 -1.557 -0.695 -0.456 -0.144 ...
##  $ s1     : num  -0.382 -0.3102 0.4076 -0.0949 1.2689 ...
##  $ s2     : num  -0.1714 0.0286 0.2 0.8285 -0.857 ...
##  $ s3     : num  -0.164 -0.189 0.179 1.097 -1.98 ...
##  $ s4     : num  0.9036 1.6069 0.2004 -0.0341 1.2162 ...
##  $ s5     : num  -0.869 -0.869 -0.128 0.612 -1.609 ...
##  $ s6     : num  -1.108 -1.315 0.187 -0.116 1.224 ...
##  $ target : num  39 48 154 111 283 103 116 138 293 144 ...
##  $ sex.2  : num  1.019 0.974 0.974 1.019 1.019 ...
##  $ bmi.2  : num  1.5521 2.4246 0.4832 0.2076 0.0208 ...
##  $ bp.2   : num  0.1459 0.0962 0.1661 0.009 1.6101 ...
##  $ s1.2   : num  0.029379 0.000816 0.039988 0.686331 0.73448 ...
##  $ s2.2   : num  0.0267 0.0359 0.0321 1.2039 3.9217 ...
##  $ s3.2   : num  0.81654 2.58209 0.04015 0.00116 1.47911 ...
##  $ s4.2   : num  0.7547 0.7547 0.0164 0.375 2.5899 ...
##  $ s5.2   : num  1.2283 1.7289 0.0348 0.0135 1.4978 ...
##  $ s6.2   : num  0.2441 0.576 0.0199 0.0903 1.0482 ...
##  $ age.sex: num  -0.806 0.0878 -0.1456 0.2285 1.4222 ...
##  $ age.bmi: num  0.995 0.138 -0.103 -0.103 -0.203 ...
##  $ age.bp : num  0.305 0.0276 0.0601 -0.0215 1.7877 ...
```

```
##  $ age.s1 : num  0.13685 -0.00254 0.02951 0.18755 -1.20742 ...
##  $ age.s2 : num  0.1306 0.0168 0.0264 0.2484 -2.79 ...
##  $ age.s3 : num  -0.72148 -0.14293 0.02956 -0.00771 1.71343 ...
##  $ age.s4 : num  0.6936 0.0773 -0.0189 0.1386 -2.2673 ...
##  $ age.s5 : num  0.8849 0.117 0.0275 -0.0263 1.7242 ...
##  $ age.s6 : num  -0.3945 -0.0675 0.0208 -0.068 1.4424 ...
##  $ sex.bmi: num  -1.258 1.537 0.686 -0.46 -0.146 ...
##  $ sex.bp : num  -0.3856 0.3062 -0.4022 -0.0958 1.2809 ...
##  $ sex.s1 : num  -0.173 -0.0282 -0.1973 0.8363 -0.8651 ...
##  $ sex.s2 : num  -0.165 0.187 -0.177 1.108 -1.999 ...
##  $ sex.s3 : num  0.9122 -1.5858 -0.1977 -0.0344 1.2277 ...
##  $ sex.s4 : num  -0.877 0.857 0.126 0.618 -1.625 ...
##  $ sex.s5 : num  -1.119 1.298 -0.184 -0.117 1.235 ...
##  $ sex.s6 : num  0.499 -0.749 -0.139 -0.303 1.033 ...
##  $ bmi.bp : num  0.4759 0.4831 -0.2833 0.0432 -0.1832 ...
##  $ bmi.s1 : num  0.2135 -0.0445 -0.139 -0.3775 0.1237 ...
##  $ bmi.s2 : num  0.204 0.295 -0.125 -0.5 0.286 ...
##  $ bmi.s3 : num  -1.1258 -2.5021 -0.1393 0.0155 -0.1756 ...
##  $ bmi.s4 : num  1.0823 1.3527 0.0891 -0.2791 0.2324 ...
##  $ bmi.s5 : num  1.3808 2.0474 -0.1297 0.0529 -0.1767 ...
##  $ bmi.s6 : num  -0.616 -1.182 -0.098 0.137 -0.148 ...
##  $ bp.s1  : num  0.06548 -0.00886 0.0815 -0.07861 -1.08748 ...
##  $ bp.s2  : num  0.0625 0.0588 0.073 -0.1041 -2.5129 ...
##  $ bp.s3  : num  -0.34519 -0.49851 0.08166 0.00323 1.54323 ...
##  $ bp.s4  : num  0.3319 0.2695 -0.0522 -0.0581 -2.0421 ...
##  $ bp.s5  : num  0.423 0.408 0.076 0.011 1.553 ...
##  $ bp.s6  : num  -0.1887 -0.2354 0.0574 0.0285 1.2991 ...
##  $ s1.s2  : num  0.02803 -0.00541 0.03582 0.90898 1.69718 ...
##  $ s1.s3  : num  -0.1549 0.0459 0.0401 -0.0282 -1.0423 ...
##  $ s1.s4  : num  0.1489 -0.0248 -0.0256 0.5074 1.3792 ...
##  $ s1.s5  : num  0.19 -0.0376 0.0373 -0.0962 -1.0489 ...
##  $ s1.s6  : num  -0.0847 0.0217 0.0282 -0.249 -0.8774 ...
##  $ s2.s3  : num  -0.1478 -0.3044 0.0359 -0.0374 -2.4084 ...
##  $ s2.s4  : num  0.142 0.165 -0.023 0.672 3.187 ...
##  $ s2.s5  : num  0.1813 0.249 0.0334 -0.1274 -2.4236 ...
##  $ s2.s6  : num  -0.0808 -0.1438 0.0252 -0.3297 -2.0274 ...
##  $ s3.s4  : num  -0.785 -1.396 -0.0257 -0.0209 -1.9572 ...
##  $ s3.s5  : num  -1.00148 -2.11283 0.03737 0.00395 1.48843 ...
##  $ s3.s6  : num  0.4465 1.2195 0.0282 0.0102 1.2451 ...
##  $ s4.s5  : num  0.9628 1.1423 -0.0239 -0.0711 -1.9695 ...
##  $ s4.s6  : num  -0.4292 -0.6593 -0.0181 -0.184 -1.6476 ...
##  $ s5.s6  : num  -0.5476 -0.9979 0.0263 0.0349 1.253 ...
```

Parameter Shrinkage with Ridge Regression

```
ridge <- caret::train(y = train_sc_pf2$target,x = train_sc_pf2[-10],
                 method = 'glmnet',
                 tuneGrid = expand.grid(alpha = 0, lambda = 1)
               )
train_pred = predict(ridge,newdata = train_sc_pf2)
val_pred = predict(ridge,newdata = val_sc_pf2)
```

```
test_pred = predict(ridge,newdata = test_sc_pf2)
res = data.frame()
w = rm(df_train$target,train_pred)
x = mse(df_train$target,train_pred)
y = rmse(df_train$target,train_pred)
z = R_2(df_train$target,train_pred)
res = rbind(res,c("Train",w,x,y,z))
w = rm(df_val$target,val_pred)
x = mse(df_val$target,val_pred)
y = rmse(df_val$target,val_pred)
z = R_2(df_val$target,val_pred)
res = rbind(res,c("Validation",w,x,y,z))
w = rm(df_test$target,test_pred)
x = mse(df_test$target,test_pred)
y = rmse(df_test$target,test_pred)
z = R_2(df_test$target,test_pred)
res = rbind(res,c("Test",w,x,y,z))
colnames(res) = c("Data","Residual Mean","MSE","RMSE","R_2")
# Following converts appropriate columns to numeric type
res[-1] = lapply(res[-1], FUN = function(y){as.numeric(y)})
# Following rounds numeric values to 4 digits
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
res
```

```
##         Data Residual Mean  MSE  RMSE     R_2
## 1      Train         39.91 2410 49.10  0.5770
## 2 Validation         49.45 3727 61.05  0.4672
## 3       Test         69.86 6837 82.69 -0.2465
```

- Best lambda

hyper-parameter tuning

```
parameters <- c(seq(0.1, 2, by =0.1) ,  seq(2, 5, 0.5) , seq(5, 25, 1))

ridge<-caret::train(y = train_sc_pf2$target,
                 x = train_sc_pf2[-10],
                 method = 'glmnet',
                 tuneGrid = expand.grid(alpha = 0, lambda = parameters) ,
                 metric =  "Rsquared"
               )

lasso<-caret::train(y = train_sc_pf2$target,
                 x = train_sc_pf2[-10],
                 method = 'glmnet',
                 tuneGrid = expand.grid(alpha = 1, lambda = parameters) ,
                 metric =  "Rsquared"
               )
paste(" Ridge Best lambda = ",ridge$finalModel$lambdaOpt)
```

```
## [1] " Ridge Best lambda =  25"
```

```r
paste(" Lasso Best lambda = ",lasso$finalModel$lambdaOpt)
```

```
## [1] " Lasso Best lambda =  7"
```

Run best ridge regression

```r
ridge_best<-caret::train(y = train_sc_pf2$target,
      x = train_sc_pf2[-10],
      method = 'glmnet',
      tuneGrid = expand.grid(alpha = 0, lambda = ridge$finalModel$lambdaOpt))
train_pred = predict(ridge_best,newdata = train_sc_pf2)
val_pred = predict(ridge_best,newdata = val_sc_pf2)
test_pred = predict(ridge_best,newdata = test_sc_pf2)
res = data.frame()
w = rm(df_train$target,train_pred)
x = mse(df_train$target,train_pred)
y = rmse(df_train$target,train_pred)
z = R_2(df_train$target,train_pred)
res = rbind(res,c("Train",w,x,y,z))
w = rm(df_val$target,val_pred)
x = mse(df_val$target,val_pred)
y = rmse(df_val$target,val_pred)
z = R_2(df_val$target,val_pred)
res = rbind(res,c("Validation",w,x,y,z))
w = rm(df_test$target,test_pred)
x = mse(df_test$target,test_pred)
y = rmse(df_test$target,test_pred)
z = R_2(df_test$target,test_pred)
res = rbind(res,c("Test",w,x,y,z))
colnames(res) = c("Data","Residual Mean","MSE","RMSE","R_2")
# Following converts appropriate columns to numeric type
res[-1] = lapply(res[-1], FUN = function(y){as.numeric(y)})
# Following rounds numeric values to 4 digits
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
res
```

```
##         Data Residual Mean  MSE  RMSE    R_2
## 1      Train         42.35 2617 51.16 0.5407
## 2 Validation         49.25 3598 59.98 0.4856
## 3       Test         47.48 3680 60.66 0.3291
```

Run best Lasso regression

```r
lasso_best<-caret::train(y = train_sc_pf2$target,
      x = train_sc_pf2[-10],
      method = 'glmnet',
      tuneGrid = expand.grid(alpha = 1, lambda = lasso$finalModel$lambdaOpt))
```

```r
train_pred = predict(lasso_best,newdata = train_sc_pf2)
val_pred = predict(lasso_best,newdata = val_sc_pf2)
test_pred = predict(lasso_best,newdata = test_sc_pf2)
res = data.frame()
w = rm(df_train$target,train_pred)
x = mse(df_train$target,train_pred)
y = rmse(df_train$target,train_pred)
z = R_2(df_train$target,train_pred)
res = rbind(res,c("Train",w,x,y,z))
w = rm(df_val$target,val_pred)
x = mse(df_val$target,val_pred)
y = rmse(df_val$target,val_pred)
z = R_2(df_val$target,val_pred)
res = rbind(res,c("Validation",w,x,y,z))
w = rm(df_test$target,test_pred)
x = mse(df_test$target,test_pred)
y = rmse(df_test$target,test_pred)
z = R_2(df_test$target,test_pred)
res = rbind(res,c("Test",w,x,y,z))
colnames(res) = c("Data","Residual Mean","MSE","RMSE","R_2")
# Following converts appropriate columns to numeric type
res[-1] = lapply(res[-1], FUN = function(y){as.numeric(y)})
# Following rounds numeric values to 4 digits
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
res
```

```
##         Data Residual Mean  MSE  RMSE    R_2
## 1      Train         46.15 3063 55.35 0.4625
## 2 Validation         50.24 3692 60.77 0.4721
## 3       Test         46.30 2871 53.58 0.4765
```

Lasso non zero coefficients

```r
lasso_coef = data.frame(
  lasso = as.data.frame.matrix(coef(lasso$finalModel, lasso$finalModel$lambdaOpt))
)
subset(lasso_coef, s1>0)
```

```
##                   s1
## (Intercept) 154.2859
## bp           24.1292
## s1            8.7508
## s6           15.8463
## age.s5        0.5735
## bmi.bp        2.4652
```

Neural Network

- Large network

```r
mlp <- caret::train(target ~ .,data = train_sc_pf2,method='mlp',size=1000)

train_pred = predict(mlp,newdata = train_sc_pf2)
val_pred = predict(mlp,newdata = val_sc_pf2)
test_pred = predict(mlp,newdata = test_sc_pf2)
res = data.frame()
y = rmse(df_train$target,train_pred)
res = rbind(res,c("Train",y))
y = rmse(df_val$target,val_pred)
res = rbind(res,c("Validation",y))
y = rmse(df_test$target,test_pred)
res = rbind(res,c("Test",y))
colnames(res) = c("Data","RMSE")
# Following converts appropriate columns to numeric type
res[-1] = lapply(res[-1], FUN = function(y){as.numeric(y)})
# Following rounds numeric values to 4 digits
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
res
```

```
##         Data  RMSE
## 1      Train 75.83
## 2 Validation 84.85
## 3       Test 75.25
```

- Smaller network

```r
mlp <- caret::train(target ~ .,data = train_sc_pf2,method='mlp',size=50)
```

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

```r
train_pred = predict(mlp,newdata = train_sc_pf2)
val_pred = predict(mlp,newdata = val_sc_pf2)
test_pred = predict(mlp,newdata = test_sc_pf2)
res = data.frame()
y = rmse(df_train$target,train_pred)
res = rbind(res,c("Train",y))
y = rmse(df_val$target,val_pred)
res = rbind(res,c("Validation",y))
y = rmse(df_test$target,test_pred)
res = rbind(res,c("Test",y))
colnames(res) = c("Data","RMSE")
# Following converts appropriate columns to numeric type
res[-1] = lapply(res[-1], FUN = function(y){as.numeric(y)})
# Following rounds numeric values to 4 digits
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
res
```

```
##         Data  RMSE
## 1      Train 75.50
## 2 Validation 83.80
## 3       Test 74.47
```

Regression Trees

```
dtr <- caret::train(target ~ .,data = train_sc_pf2,method='rpart')

train_pred = predict(dtr,newdata = train_sc_pf2)
val_pred = predict(dtr,newdata = val_sc_pf2)
test_pred = predict(dtr,newdata = test_sc_pf2)
res = data.frame()
y = rmse(df_train$target,train_pred)
res = rbind(res,c("Train",y))
y = rmse(df_val$target,val_pred)
res = rbind(res,c("Validation",y))
y = rmse(df_test$target,test_pred)
res = rbind(res,c("Test",y))
colnames(res) = c("Data","RMSE")
# Following converts appropriate columns to numeric type
res[-1] = lapply(res[-1], FUN = function(y){as.numeric(y)})
# Following rounds numeric values to 4 digits
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
res
```

```
##         Data  RMSE
## 1      Train 58.84
## 2 Validation 69.18
## 3       Test 67.80
```

- with a maximum depth of 2

```
dtr <- caret::train(target ~ .,data = train_sc_pf2,method='rpart',
                    control = list(max_depth=2))
```

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

```
train_pred = predict(dtr,newdata = train_sc_pf2)
val_pred = predict(dtr,newdata = val_sc_pf2)
test_pred = predict(dtr,newdata = test_sc_pf2)
res = data.frame()
y = rmse(df_train$target,train_pred)
res = rbind(res,c("Train",y))
y = rmse(df_val$target,val_pred)
res = rbind(res,c("Validation",y))
y = rmse(df_test$target,test_pred)
res = rbind(res,c("Test",y))
colnames(res) = c("Data","RMSE")
# Following converts appropriate columns to numeric type
res[-1] = lapply(res[-1], FUN = function(y){as.numeric(y)})
# Following rounds numeric values to 4 digits
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
res
```

```
##         Data  RMSE
## 1      Train 58.84
## 2 Validation 69.18
## 3       Test 67.80
```

Harder problem

- Read data

```
df <- read.csv("ENB2012_data.csv")
```

- Data preparation

```
dataset = df[-c(11,12)]
N = nrow(dataset)
cut1 = floor(0.6*N)
cut2 = floor(0.8*N)
index = sample(1:N)
train_index = index[1:cut1]
val_index = index[(cut1+1):cut2]
test_index = index[(cut2+1):N]
df_train = dataset[train_index,]
df_val = dataset[val_index,]
df_test = dataset[test_index,]
sapply(list(df_train,df_val,df_test),nrow)
```

```
## [1] 460 154 154
```

```
df_train[-10] = scale(df_train[-10])
```

- Build models

```
res = data.frame()
# Linear Regression
lr = caret::train(Y2 ~ ., method='lm',data = df_train)
train_pred = predict(lr,newdata = df_train)
val_pred = predict(lr,newdata = scale(df_val[-10]))
test_pred = predict(lr,newdata = scale(df_test[-10]))
y1 = rmse(df_train$Y2,train_pred)
y2 = rmse(df_val$Y2,val_pred)
y3 = rmse(df_test$Y2,test_pred)
res = rbind(res,c("Linear Regression",y1,y2,y3))
# Ridge Regression
ridge <- caret::train(y = df_train$Y2,x = df_train[-10],
                 method = 'glmnet',
                 tuneGrid = expand.grid(alpha = 0, lambda = 1)
               )
train_pred = predict(ridge,newdata = df_train)
val_pred = predict(ridge,newdata = scale(df_val[-10]))
test_pred = predict(ridge,newdata = scale(df_test[-10]))
```

```r
y1 = rmse(df_train$Y2,train_pred)
y2 = rmse(df_val$Y2,val_pred)
y3 = rmse(df_test$Y2,test_pred)
res = rbind(res,c("Ridge Regression",y1,y2,y3))
# Lasso Regression
lasso <- caret::train(y = df_train$Y2,x = df_train[-10],
                 method = 'glmnet',
                 tuneGrid = expand.grid(alpha = 1, lambda = 0)
               )
train_pred = predict(lasso,newdata = df_train)
val_pred = predict(lasso,newdata = scale(df_val[-10]))
test_pred = predict(lasso,newdata = scale(df_test[-10]))
y1 = rmse(df_train$Y2,train_pred)
y2 = rmse(df_val$Y2,val_pred)
y3 = rmse(df_test$Y2,test_pred)
res = rbind(res,c("Lasso Regression",y1,y2,y3))
# Neural Net
nn <- caret::train(y = df_train$Y2,x = df_train[-10],
                 method = 'mlp')
train_pred = predict(nn,newdata = df_train)
val_pred = predict(nn,newdata = scale(df_val[-10]))
test_pred = predict(nn,newdata = scale(df_test[-10]))
y1 = rmse(df_train$Y2,train_pred)
y2 = rmse(df_val$Y2,val_pred)
y3 = rmse(df_test$Y2,test_pred)
res = rbind(res,c("Neural Net",y1,y2,y3))
# Regression Tree
dt <- caret::train(y = df_train$Y2,x = df_train[-10],
                 method = 'rpart')
train_pred = predict(dt,newdata = df_train)
val_pred = predict(dt,newdata = scale(df_val[-10]))
test_pred = predict(dt,newdata = scale(df_test[-10]))
y1 = rmse(df_train$Y2,train_pred)
y2 = rmse(df_val$Y2,val_pred)
y3 = rmse(df_test$Y2,test_pred)
res = rbind(res,c("Regression Tree",y1,y2,y3))

colnames(res) = c("Model","Train","Validation","Test")
# Following converts appropriate columns to numeric type
res[-1] = lapply(res[-1], FUN = function(y){as.numeric(y)})
# Following rounds numeric values to 4 digits
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
res
```

```
##                Model Train Validation  Test
## 1 Linear Regression 1.826      2.225 2.179
## 2  Ridge Regression 2.154      2.691 2.557
## 3  Lasso Regression 1.828      2.229 2.183
## 4        Neural Net 3.065      3.619 3.276
## 5   Regression Tree 2.939      3.555 3.113
```