Chapter 18 Interactive Notebook for Students

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('../../data/diabetes.csv', header = TRUE)
dim(dataset)
```

```
## [1] 442  11
```

Partition Data

- Note the use of sample() function to create training, validation, and test datasets. Another interesting function is sapply(). This is part of a list of so-called apply() functions in R. They essentially allow you to run a function on multiple different inputs. In the case below, we want to get the number of rows (which is done through a function) for three different dataframes. Instead of writing three different lines of code, sapply() makes the coding simpler and more elegant.

```
set.seed(123456)
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
```

Linear Regression Model

- For this and the subsequent chapter, we will make use of the caret package that provides a uniform approach to work with a variety of predictive models. Also note the use of :: notation. This is used to indicate the package that contains the function that follows. This is helpful as often different packages may use the same function names.

```
means = apply(df_train[-11],2,mean)
sds = apply(df_train[-11],2,sd)
scalefun = function(x){
  return((x-means)/sds)
}
df_train[-11] = data.frame(sapply(df_train[-11],scalefun))
df_val[-11] = data.frame(sapply(df_val[-11],scalefun))
df_test[-11] = data.frame(sapply(df_test[-11],scalefun))
printfun = function(x){
  means = apply(x[-11],2,mean)
  sds = apply(x[-11],2,sd)
  print(paste(mean(means),mean(sds)))
}
printfun(df_train)
```

```
## [1] "0.000554657253475731 1.00044597757519"
```

```
printfun(df_val)
```

```
## [1] "-0.0182503268108164 0.980026441948433"
```

```
printfun(df_test)
```

```
## [1] "0.0533555353079712 0.987301726800513"
```

```
lr = caret::train(target ~ ., method='lm',data = df_train)
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){
  mean_v = rep(mean(actual),length(actual))
  SST = sum((actual-mean_v)^2)
  SSE = sum((actual-pred)^2)
  return(1-(SSE/SST))
}
```

Performance of the Linear Regression Model

- In the code below, we want to assess the performance of the model on training, validation, and test datasets. To make it easier to read and compare, we will put the results in a dataframe. Also note the use of lapply() function - in this case we want to change the data type of all the variables, except the first column, to numeric. Again, this function makes it easier to code.

```
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)
```

```r
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)
knitr::kable(res)
```

Data	Residual Mean	MSE	RMSE	R_2
Train	41.79	2659	51.56	0.5742
Validation	47.58	3497	59.13	0.3238
Test	43.69	3014	54.90	0.4693

Polynomial Regression Model

- Column names of the original data set.

```r
colnames(df_train)
```

```
##  [1] "age"    "sex"    "bmi"    "bp"     "s1"     "s2"     "s3"     "s4"
##  [9] "s5"     "s6"     "target"
```

Generate the dataset

- The strategy is to write a formula for squared and interaction terms, and use the formula to generate a new dataset that contains the polynomial terms.

```r
# 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
```

```r
  output_df[,1] <- NULL
  return(output_df)
}
```

Create training, validation and test sets

- Create the training data set

```r
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"
```

```r
dim(df_train[-10])
```

```
## [1] 265  10
```

```r
dim(train_sc_pf2)
```

```
## [1] 265  64
```

- Prepare the validation and test sets

```r
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]


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]
```

Run the Polynomial Regression

```
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

```
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)
knitr::kable(res)
```

Data	RMSE
Train	44.95
Validation	71.05
Test	77.25

Ridge Regression

Basic Model

```
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()
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")
```

```r
# 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)
knitr::kable(res)
```

Data	RMSE
Train	46.66
Validation	59.19
Test	57.10

Hyper-parameter tuning

```r
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$target,
                 x = train_sc_pf2[-10],
                 method = 'glmnet',
                 tuneGrid = expand.grid(alpha = 0, lambda = parameters) ,
                 metric =  "Rsquared"
               )
paste(" Ridge Best lambda = ",ridge$finalModel$lambdaOpt)
```

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

Run the Optimized 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$lambdaOpt))
train_pred = predict(ridge_best,newdata = train_sc_pf2)
val_pred = predict(ridge_best,newdata = val_sc_pf2)
test_pred = predict(ridge_best,newdata = test_sc_pf2)
res = data.frame()
y = rmse(df_train$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)
knitr::kable(res)
```

Data	RMSE
Train	49.03
Validation	57.48
Test	55.76

Evaluate Performance of the optimized Ridge Regression

```
res = data.frame()
y = R2(train_pred,df_train$target)
res = rbind(res,c("Train",y))
y = R2(val_pred,df_val$target)
res = rbind(res,c("Validation",y))
y = R2(test_pred,df_test$target)
res = rbind(res,c("Test",y))
colnames(res) = c("Data","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)
knitr::kable(res)
```

Data	R_2
Train	0.6275
Validation	0.3794
Test	0.4754

Lasso Regression

Basic Model

```
lasso<-caret::train(y = train_sc_pf2$target,
      x = train_sc_pf2[-10],
      method = 'glmnet',
      tuneGrid = expand.grid(alpha = 1, lambda = 0))
train_pred = predict(lasso,newdata = train_sc_pf2)
val_pred = predict(lasso,newdata = val_sc_pf2)
test_pred = predict(lasso,newdata = test_sc_pf2)
res = data.frame()
y = rmse(df_train$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)
knitr::kable(res)
```

Data	RMSE
Train	45.42
Validation	62.98
Test	65.22

Hyper-parameter tuning

- Find the best hyper parameter lambda

```
parameters <- c(seq(0.1, 2, by =0.1) ,  seq(2, 5, 0.5) , seq(5, 25, 1))
lasso<-caret::train(y = train_sc_pf2$target,
                 x = train_sc_pf2[-10],
                 method = 'glmnet',
                 tuneGrid = expand.grid(alpha = 1, lambda = parameters) ,
                 metric =  "Rsquared"
               )
paste(" Lasso Best lambda = ",lasso$finalModel$lambdaOpt)
```

```
## [1] " Lasso Best lambda =  5"
```

Run the Optimized Lasso Regression

```
lasso_best<-caret::train(y = train_sc_pf2$target,
      x = train_sc_pf2[-10],
      method = 'glmnet',
      tuneGrid = expand.grid(alpha = 1, lambda = lasso$finalModel$lambdaOpt))
train_pred = predict(lasso_best,newdata = train_sc_pf2)
val_pred = predict(lasso_best,newdata = val_sc_pf2)
test_pred = predict(lasso_best,newdata = test_sc_pf2)
res = data.frame()
y = rmse(df_train$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)
knitr::kable(res)
```

Data	RMSE
Train	52.63
Validation	57.74
Test	53.41

Performance of the Optimized Lasso Regression

```
res = data.frame()
y = R2(train_pred,df_train$target)
res = rbind(res,c("Train",y))
y = R2(val_pred,df_val$target)
res = rbind(res,c("Validation",y))
y = R2(test_pred,df_test$target)
res = rbind(res,c("Test",y))
colnames(res) = c("Data","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)
knitr::kable(res)
```

Data	R_2
Train	0.5663
Validation	0.3677
Test	0.5078

Determine non-zero coefficients

```
df = data.frame(
  lasso = as.data.frame.matrix(coef(lasso$finalModel, lasso$finalModel$lambdaOpt))
)
df =subset(df, s1>0.1)
df$var = row.names(df)
knitr::kable(df[order(-df$s1),c(2,1)][2])
```

	s1
(Intercept)	153.1769
bp	25.3203
s6	19.9792
s1	13.1870
bmi.s6	3.7814
sex.s3	2.0960
age.s6	1.1315
s6.2	0.2655
bmi.bp	0.2125

Neural Network

Large network

```
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)
knitr::kable(res)
```

Data	RMSE
Train	102.7
Validation	100.9
Test	107.1

Smaller network

```
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.
```

```
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)})
```

```r
# Following rounds numeric values to 4 digits
res[,sapply(res, is.numeric)] <-round(res[,sapply(res, is.numeric)],4)
knitr::kable(res)
```

Data	RMSE
Train	86.42
Validation	83.83
Test	86.49

Regression Tree

Basic Tree

```r
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)
knitr::kable(res)
```

Data	RMSE
Train	59.71
Validation	66.82
Test	60.15

Tree with a maximum depth of 2

```r
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.
```

```r
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)
knitr::kable(res)
```

Data	RMSE
Train	59.71
Validation	66.82
Test	60.15

Harder problem

Data preparation

- Read data

```r
df <- read.csv("../../data/ENB2012_data.csv")
```

- Create training, validation and test datasets.

```r
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 and Evaluate 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]))
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')
```

```r
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)
knitr::kable(res)
```

Model	Train	Validation	Test
Linear Regression	1.921	2.140	1.992
Ridge Regression	2.308	2.241	2.540
Lasso Regression	1.923	2.139	1.993
Neural Net	3.228	3.547	3.324
Regression Tree	3.006	3.101	3.301