R Data Science

[**Data Clean**](#_7m27zuqyrm9h) **3**

[**Correlation**](#_qckp70fg5yix) **3**

[**Feature Selection**](#_3fsorq83p07d) **4**

[**Data Split**](#_qdighwm0eqv) **4**

[**Machine Learning**](#_ssnrqy8pgjn) **5**

[**…**](#_slbjw7g25ya1) **5**

# Data Clean

Mapping values in a column

i. e., change the value from smth1 to smth2 - package “plyt”

df$col <- mapvalues(df$col, from=c(smth1), to=c(smth2))

Replace values

replace the pattern by the replacement, in a given column

str\_replace\_all(df$col, pattern, replacement) - all pattern

str\_replace(df$col, pattern, replacement) - first pattern

Duplicate

.

Drop unnecessary rows

the lines that have some element with value NA

df <- na.omit(df)

Normalize the data

df[,cols] <- scale(df[,cols])

Convert column to type factor

df$col <- as.factor(df$col)

# Correlation

Simple correlation

to numeric variables

m = "pearson" | "spearman" | "kendall"

cor(df[,cols],method=m)

Scatterplot Matrix

to numeric variables

package "*psych*"

pairs.panels(df[c("x1","x2", ... ,"xn","y")])

Chi square test

to categorical variables

the lower the probability the greater the correlation

CrossTable(x = df$col, y = df$col, chisq = TRUE) - package "gmodels"

chisq.test(x = df$col, y = df$col)

Contingency table

to categorical variables

package "*gmodels*"

CrossTable(x = df$col, y = df$col)

# Feature Selection

Packet "randomForest"

mod <- randomForest(y~x, data=df, ntree=n, nodesize=n1, importance=T)

varImpPlot(mod) - it’ll plot the most important one

Packet "randomForest" and "caret"

| 1. run.feature.selection <- function(num.iters=20, feature.vars, class.var){  variable.sizes <- 1:10  control <- rfeControl(functions = rfFuncs, method = "cv", verbose = FALSE, returnResamp = "all", number = num.iters)  results.rfe <- rfe(x = feature.vars, y = class.var, sizes = variable.sizes, rfeControl = control)   return(results.rfe)  }  // execute the feature selection rfe.results <- run.feature.selection(feature.vars = x, class.var = y)   // plot the most important columns varImpPlot(mod) |
| --- |

Packet "olsrr"

# Stepwise forward regression -Começa com um modelo vazio e adiciona cada melhor variável sequencialmente.

ols\_step\_forward\_p(modelo)

# Stepwise backward regression-Começa com um modelo saturado (contendo todas as variáveis possíveis) e remove a pior variável sequencialmente.

ols\_step\_backward\_p(modelo)

# Stepwise regression-Começa com um modelo vazio e adicione uma variável por vez. Adicionamos a melhor variável e, em seguida, removemos a pior variável.

ols\_step\_both\_p(modelo)

# Stepwise AIC forward regression

ols\_step\_forward\_aic(modelo)

# Stepwise AIC backward regression

ols\_step\_backward\_aic(modelo)

# Stepwise AIC regression

ols\_step\_both\_aic(modelo)

# Data Split

**1.** Using a function - packet “caret”

index <- createDataPartition(df$FieldY, p=n, list=FALSE)

n is the training data percentage (0.7)

**2.** Using custom variables

i <- 1:nrow(df)

index <- sample(i, trunc(length(i)/2))

Getting the data

training <- df[index,]

testing <- df[-index,]

# Machine Learning

Linear Regression - packet “caret”

mod <- lm(y ~ x, data = df)

bptest(mod) *- Test Heteroskedasticity using Breusch-Pagan Test*

vif(mod) *- Test multicollinearity using Variance Inflation Factor*

GLM w/ Lasso or Elasticnet Reg

glmnet(x, y, alpha) *- alpha=1 is lasso penalty, and, =0 is ridge penalty*

Logistic Regression - packet “caret”

*family* is a description of the error distribution and link func - [others families](https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/family)

mod <- glm(y ~ x, family=binomial(link="logit"), data=training)

Decision Tree - package “party”

mod <- ctree(y ~ x, training)

Random Forest - package "randomForest"

mod <- randomForest(y ~ x, data = training)

KNN - package “class”

mod <- knn(train = training, test = testing, cl = data\_train\_labels, k = n)

SVM - package “*e1071*”

*type* can be: ‘C-classification’, ‘nu-classification’, ‘eps-regression’, ‘nu-regression’

*kernel* can be: ‘radial’, ‘linear’, ‘sigmoid’

mod <- svm(y ~ x, data = training, type = 'C-classification', kernel = 'radial')

Predictions

predict(mod, testing[,-y])

Confusion matrix

table(Predicted = pred, Actual = testing$y)

Residuals

resid(mod)

Model Resume

summary(mod)

Get Coefficient

mod$coefficients

Model Bootstrap - package “*car*”

bootCase(mod, function(x) predict(x,testing[,-y],B=999)

Important Variables

varImp(mod)

# …