Pre-Processing Data

Felliks F Tampinongkol, Sahid A Hudjimartsu, Lilik B Prasetyo dan Yudi Setiawan

Jumat 14/04-2020

## Pre-Processing Data LiDAR dan Landsat 8 OLI using Support Vector Regression (SVR)

Data Canopy Cover yang digunakan dapat didownload pada Folder “…” Berikut merupakan package yang digunakan dalam pre-processing data menggunakan RStudio:

# install.packages(c("packages\_name"))  
# library(dbscan, readxl, dplyr, e1071, Boruta, dismo, caret, raster, openxlsx)  
  
rmse <- function(error)  
# Fungsi untuk menghitung Nilai root mean squared error (RMSE)  
{  
 sqrt(mean(error^2))  
}

## Langkah 1 - Set lokasi penyimpanan dan load file excel

library(openxlsx)  
  
setwd('C:/Users/Felix/Dropbox/FORESTS2020/00AllData/')  
load\_data <- read.xlsx("Data Canopy Cover.xlsx")  
head(load\_data)

## FID Shape Class kategori frci Band\_1 Band\_2 Band\_3 Band\_4  
## 1 544 Point 1 Sangat Rendah 0 0.119028 0.096263 0.088627 0.067901  
## 2 684 Point 1 Sangat Rendah 0 0.113155 0.089109 0.072956 0.050277  
## 3 686 Point 1 Sangat Rendah 0 0.124964 0.104814 0.095479 0.100075  
## 4 723 Point 1 Sangat Rendah 0 0.120681 0.099382 0.089137 0.079335  
## 5 739 Point 1 Sangat Rendah 0 0.113995 0.090231 0.077047 0.051841  
## 6 742 Point 1 Sangat Rendah 0 0.114693 0.091302 0.083048 0.052883  
## Band\_5 Band\_6 Band\_7 Band\_9  
## 1 0.356388 0.246095 0.124561 0.001402  
## 2 0.299869 0.164176 0.080432 0.001548  
## 3 0.237620 0.215206 0.135871 0.001697  
## 4 0.288038 0.236145 0.133857 0.001737  
## 5 0.352246 0.195241 0.088700 0.001420  
## 6 0.381807 0.207427 0.092613 0.001679

summary(load\_data)

## FID Shape Class kategori   
## Min. : 0.0 Length:1047 Min. : 1.00 Length:1047   
## 1st Qu.: 261.5 Class :character 1st Qu.: 4.00 Class :character   
## Median : 523.0 Mode :character Median : 7.00 Mode :character   
## Mean : 523.0 Mean : 6.18   
## 3rd Qu.: 784.5 3rd Qu.: 9.00   
## Max. :1046.0 Max. :10.00   
## frci Band\_1 Band\_2 Band\_3   
## Min. :0.0000 Min. :0.1051 Min. :0.08066 Min. :0.06139   
## 1st Qu.:0.3355 1st Qu.:0.1112 1st Qu.:0.08642 1st Qu.:0.06958   
## Median :0.6408 Median :0.1143 Median :0.08960 Median :0.07287   
## Mean :0.5680 Mean :0.1139 Mean :0.08948 Mean :0.07382   
## 3rd Qu.:0.8128 3rd Qu.:0.1162 3rd Qu.:0.09168 3rd Qu.:0.07679   
## Max. :1.0000 Max. :0.1360 Max. :0.11716 Max. :0.09990   
## Band\_4 Band\_5 Band\_6 Band\_7   
## Min. :0.03673 Min. :0.1778 Min. :0.08948 Min. :0.03179   
## 1st Qu.:0.04273 1st Qu.:0.2993 1st Qu.:0.13215 1st Qu.:0.05226   
## Median :0.04547 Median :0.3253 Median :0.15375 Median :0.06280   
## Mean :0.04873 Mean :0.3236 Mean :0.15736 Mean :0.06931   
## 3rd Qu.:0.05039 3rd Qu.:0.3508 3rd Qu.:0.17137 3rd Qu.:0.07383   
## Max. :0.10473 Max. :0.4551 Max. :0.29533 Max. :0.21130   
## Band\_9   
## Min. :0.000918   
## 1st Qu.:0.001532   
## Median :0.001736   
## Mean :0.001762   
## 3rd Qu.:0.001960   
## Max. :0.002959

## Langkah 2 - Sleksi dan Balencing Data

Setelah data file berhasil diload selanjutnya data dipilih sesuai dengan kebutuhan. Dalam kasus ini atribut data yang akan digunakan adalah (“Class, frci dan Band\_2 sampai Band\_7”). Kemudian data yang telah dipilih berdasarkan atribut dilakukan balancing berdasarkan jumlah Class yang memiliki nilai frekuensi terkecil. Tahapan seleksi dan balancing data sebagai berikut:

# Tahapan seleksi dan Balancing Data  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

balancing\_class <- function(data){  
 number <- data %>% group\_by(Class) %>% summarize(n())  
 sample <- data %>% group\_by(Class) %>% sample\_n(min(number$`n()`))  
}  
  
data <- load\_data[, c("Class", "frci", "Band\_2", "Band\_3",  
 "Band\_4", "Band\_5", "Band\_6", "Band\_7")]  
  
sample <- balancing\_class(data)  
sample <- sample[-1] ## Hapus column Class  
head(sample)

## # A tibble: 6 x 7  
## frci Band\_2 Band\_3 Band\_4 Band\_5 Band\_6 Band\_7  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0.0953 0.0927 0.0865 0.0574 0.420 0.229 0.106   
## 2 0.00394 0.0962 0.0903 0.0715 0.365 0.250 0.126   
## 3 0.0565 0.0913 0.0807 0.0534 0.361 0.210 0.0945  
## 4 0.0248 0.101 0.0849 0.0875 0.179 0.233 0.211   
## 5 0.0693 0.0890 0.0771 0.0512 0.362 0.181 0.0800  
## 6 0.00313 0.0908 0.0794 0.0527 0.324 0.187 0.0895

## Langkah 3 - Menghapus Data Pencilan using DBSCAN

Data yang telah diproses dalam Langkah 2, selanjutnya dilakukan pembulatan 3 angka dibelakang koma dan proses membuang outlier atau data pencilan menggunakan algoritma DBSCAN. Proses dapat dilihat seperti dibawah ini:

1. Proses pembulatan 3 Angka dibelakang koma
2. Menentukan nilai epsilon untuk menghapus data pencilan

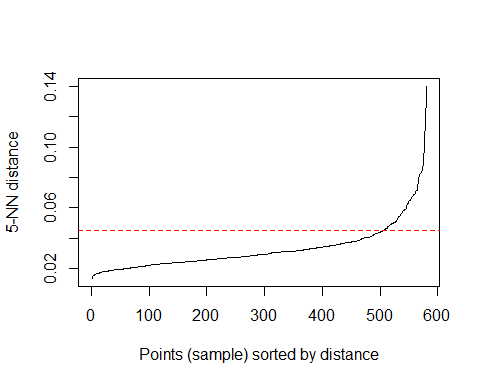
library(dbscan)  
lst <- as.data.frame(lapply(sample, function(x) round(x, 3)))  
head(lst)

## frci Band\_2 Band\_3 Band\_4 Band\_5 Band\_6 Band\_7  
## 1 0.095 0.093 0.086 0.057 0.420 0.229 0.106  
## 2 0.004 0.096 0.090 0.071 0.365 0.250 0.126  
## 3 0.057 0.091 0.081 0.053 0.361 0.210 0.094  
## 4 0.025 0.101 0.085 0.087 0.179 0.233 0.211  
## 5 0.069 0.089 0.077 0.051 0.362 0.181 0.080  
## 6 0.003 0.091 0.079 0.053 0.324 0.187 0.090

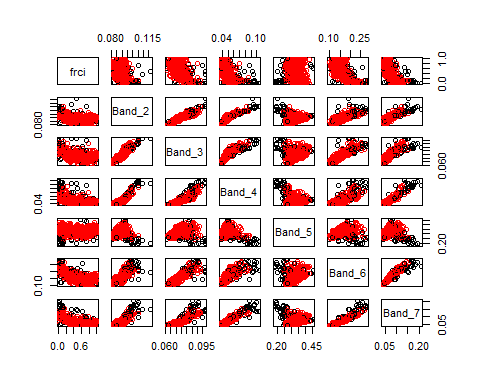
dataSample <- lst  
head(dataSample)

## frci Band\_2 Band\_3 Band\_4 Band\_5 Band\_6 Band\_7  
## 1 0.095 0.093 0.086 0.057 0.420 0.229 0.106  
## 2 0.004 0.096 0.090 0.071 0.365 0.250 0.126  
## 3 0.057 0.091 0.081 0.053 0.361 0.210 0.094  
## 4 0.025 0.101 0.085 0.087 0.179 0.233 0.211  
## 5 0.069 0.089 0.077 0.051 0.362 0.181 0.080  
## 6 0.003 0.091 0.079 0.053 0.324 0.187 0.090

# Nilai Epsilon yang digunakan 0.045  
kNNdistplot(dataSample, k = 5)  
change\_data <- 0.045  
abline(h = change\_data, col = "red", lty = 2)

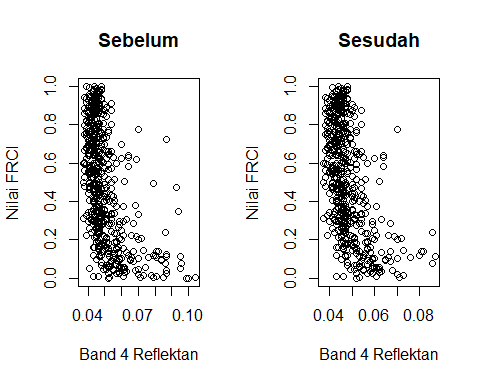


res <- dbscan(dataSample, eps = change\_data, minPts = 5)  
# Ploting sebaran data FRCI terhadap nilai Band Reflektan  
pairs(dataSample, col = res$cluster + 1L)



Dalam proses ini kita telah memperoleh new data frame (“cleanall”) yang kita angap bebas dari outlier atau data pencilan, sehingga data frame inilah yang akan digunakan untuk proses membuat model menggunakan SVR.

dataSample$cluster <- res$cluster  
cleanall <- dataSample %>% filter(cluster > 0)  
  
# Ploting data sebelum dan sesudah dihapus outlier  
par(mfrow=c(1,2))  
plot(dataSample$Band\_4, dataSample$frci, xlab = 'Band 4 Reflektan',  
 ylab = 'Nilai FRCI', main = 'Sebelum')  
plot(cleanall$Band\_4, cleanall$frci, xlab = 'Band 4 Reflektan',  
 ylab = 'Nilai FRCI', main = 'Sesudah')



## Langkah 4 - Boruta dan Support Vector Regression (SVR)

Dalam Langkah ke 4 - pertama yang kita lakukan adalah melihat importance variable menggunakan Boruta algorithm, bertujuan untuk mengetahui variabel-variabel apa saja yang sangat berpengaruh dalam pembuatan model menggunakan Machine Learning. Tapi sebelum itu, kita pastikan bahwa data frame yang akan digunakan untuk membuat model telah sesuai dengan kebutuhan kita.

head(cleanall)

## frci Band\_2 Band\_3 Band\_4 Band\_5 Band\_6 Band\_7 cluster  
## 1 0.095 0.093 0.086 0.057 0.420 0.229 0.106 1  
## 2 0.004 0.096 0.090 0.071 0.365 0.250 0.126 1  
## 3 0.057 0.091 0.081 0.053 0.361 0.210 0.094 1  
## 4 0.069 0.089 0.077 0.051 0.362 0.181 0.080 1  
## 5 0.003 0.091 0.079 0.053 0.324 0.187 0.090 1  
## 6 0.055 0.094 0.084 0.062 0.321 0.235 0.118 1

svrdata <- cleanall[, -c(8)]  
# svrData merupakan data frame baru dari 'cleanall' yang column cluster hasil DBSCAN telah dihilangkan  
head(svrdata)

## frci Band\_2 Band\_3 Band\_4 Band\_5 Band\_6 Band\_7  
## 1 0.095 0.093 0.086 0.057 0.420 0.229 0.106  
## 2 0.004 0.096 0.090 0.071 0.365 0.250 0.126  
## 3 0.057 0.091 0.081 0.053 0.361 0.210 0.094  
## 4 0.069 0.089 0.077 0.051 0.362 0.181 0.080  
## 5 0.003 0.091 0.079 0.053 0.324 0.187 0.090  
## 6 0.055 0.094 0.084 0.062 0.321 0.235 0.118

# Cara untuk menjalankan banyak package sekaligus dalam RStudio  
x <- c("Boruta", "caret", "e1071")  
lapply(x, FUN = function(X){ do.call("require", list(X)) })

## Loading required package: Boruta

## Loading required package: ranger

## Loading required package: caret

## Loading required package: lattice

## Loading required package: ggplot2

## Loading required package: e1071

## [[1]]  
## [1] TRUE  
##   
## [[2]]  
## [1] TRUE  
##   
## [[3]]  
## [1] TRUE

boruta\_output <- Boruta(frci ~ ., data=na.omit(svrdata), doTrace=2)

## 1. run of importance source...

## 2. run of importance source...

## 3. run of importance source...

## 4. run of importance source...

## 5. run of importance source...

## 6. run of importance source...

## 7. run of importance source...

## 8. run of importance source...

## 9. run of importance source...

## 10. run of importance source...

## After 10 iterations, +0.82 secs:

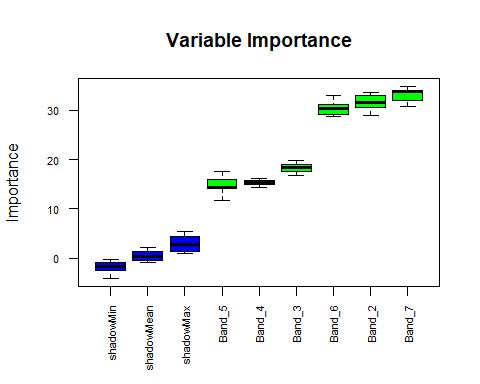
## confirmed 6 attributes: Band\_2, Band\_3, Band\_4, Band\_5, Band\_6 and 1 more;

## no more attributes left.

boruta\_signif <- names(boruta\_output$finalDecision[boruta\_output$finalDecision %in% ("Confirmed")])  
print(boruta\_signif)

## [1] "Band\_2" "Band\_3" "Band\_4" "Band\_5" "Band\_6" "Band\_7"

plot(boruta\_output, cex.axis=.7, las=2, xlab="", main="Variable Importance")



## Support Vector Regression (SVR)

Setelah kita mendapati variable yang berpengaruh dalam proses pembuatan model yang akan kita bangun, selanjutnya kita membagi data kita menjadi dua, yang pertama training data untuk membuat model dan testing data untuk melakukan test pada model yang telah kita bangun menggunakan training data. Dalam kasus ini kita membagi 80% training dan 20% testing.

set.seed(3033)  
intrain <- createDataPartition(y = svrdata$frci, p= 0.8, list = FALSE)  
training <- svrdata[intrain,]  
testing <- svrdata[-intrain,]  
dim(training)

## [1] 432 7

dim(testing)

## [1] 107 7

anyNA(svrdata)

## [1] FALSE

Kernel yang digunakan dalam kasus ini adalah RBF (radial basis function)

model <- svm(frci ~ . , data = training, kernel = "radial")  
plot(model, svrdata)  
predictedY <- predict(model, testing)  
error <- testing$frci - predictedY  
svrPredictionRMSE <- rmse(error)  
  
print(model)

##   
## Call:  
## svm(formula = frci ~ ., data = training, kernel = "radial")  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 0.1666667   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 368

paste("RMSE value:", svrPredictionRMSE)

## [1] "RMSE value: 0.178430850273019"

Setelah kita memperoleh model prediksi selanjutnya kita membuat data frame berdasarkan model yang telah kita peroleh pada proses sebelumnya. Data frame ini bertujuan untuk menghitung nilai coefficient of determination dan dapat dipakai untuk membuat model prediksi pada proses selanjutnya.

df <- data.frame(testing$frci, predictedY)  
average\_actual <- mean(df$testing.frci)  
sumofSquares\_total <- sum((df$testing.frci - average\_actual)^2)  
sumofSquares\_regression <- sum((df$predictedY - average\_actual)^2)  
sumofSquares\_residuals <- sum((df$testing.frci - df$predictedY)^2)  
RSquare <- 1 - sumofSquares\_residuals / sumofSquares\_total  
paste("RSquare value:", RSquare)

## [1] "RSquare value: 0.594581826682559"

Selanjutnya kita simpan data frame yang dihasilkan berdasarkan nilai RSquare dan RMSE yang di peroleh. setwd(“directory”) untuk menset lokasi penyimpanan

setwd("C:/Users/Felix/Dropbox/FORESTS2020/00AllData/")  
write.xlsx(cleanall, file = "New\_DataCC\_SVR")

## Note: zip::zip() is deprecated, please use zip::zipr() instead