DSE6211 Module 03 Lab 03

Joseph Annand

2024-02-04

# Code for Data Pre-Processing

## Load Libraries

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

library(caret)

## Warning: package 'caret' was built under R version 4.3.2

## Loading required package: ggplot2

## Loading required package: lattice

## Load data

# Separate lab 3 data set into test and training set  
data <- read.csv("lab\_3\_data/lab\_3\_data.csv")  
training\_ind <- createDataPartition(data$lodgepole\_pine,  
 p = 0.75,  
 list = F,  
 times = 1)  
  
training\_set <- data[training\_ind, ]  
test\_set <- data[-training\_ind, ]  
  
unique(training\_set$wilderness\_area)

## [1] "wilderness\_area\_1" "wilderness\_area\_3" "wilderness\_area\_4"  
## [4] "wilderness\_area\_2"

unique(training\_set$soil\_type)

## [1] "soil\_type\_18" "soil\_type\_30" "soil\_type\_12" "soil\_type\_29" "soil\_type\_20"  
## [6] "soil\_type\_23" "soil\_type\_24" "soil\_type\_22" "soil\_type\_32" "soil\_type\_10"  
## [11] "soil\_type\_5" "soil\_type\_33" "soil\_type\_17" "soil\_type\_39" "soil\_type\_13"  
## [16] "soil\_type\_2" "soil\_type\_31" "soil\_type\_14" "soil\_type\_3" "soil\_type\_6"   
## [21] "soil\_type\_16" "soil\_type\_11" "soil\_type\_40" "soil\_type\_38" "soil\_type\_25"  
## [26] "soil\_type\_4" "soil\_type\_19" "soil\_type\_1" "soil\_type\_9" "soil\_type\_28"  
## [31] "soil\_type\_34" "soil\_type\_37" "soil\_type\_35" "soil\_type\_21" "soil\_type\_36"  
## [36] "soil\_type\_26" "soil\_type\_27"

## Prepare categorical features

# Create a table with the 20 most common soil types from the training set  
top\_20\_soil\_types <- training\_set %>%  
 group\_by(soil\_type) %>%  
 summarise(count = n()) %>%  
 arrange(desc(count)) %>%  
 select(soil\_type) %>%  
 top\_n(20)

## Selecting by soil\_type

# Convert the value of 'soil\_type' feature to "other" for all observations whose soil type is not in the 20 most common  
training\_set$soil\_type <- ifelse(training\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,  
 training\_set$soil\_type,  
 "other")  
  
# Convert wilderness area and soil type features to factor class  
training\_set$wilderness\_area <- factor(training\_set$wilderness\_area)  
training\_set$soil\_type <- factor(training\_set$soil\_type)  
  
class(training\_set$wilderness\_area)

## [1] "factor"

class(training\_set$soil\_type)

## [1] "factor"

levels(training\_set$wilderness\_area)

## [1] "wilderness\_area\_1" "wilderness\_area\_2" "wilderness\_area\_3"  
## [4] "wilderness\_area\_4"

levels(training\_set$soil\_type)

## [1] "other" "soil\_type\_26" "soil\_type\_27" "soil\_type\_28" "soil\_type\_29"  
## [6] "soil\_type\_3" "soil\_type\_30" "soil\_type\_31" "soil\_type\_32" "soil\_type\_33"  
## [11] "soil\_type\_34" "soil\_type\_35" "soil\_type\_36" "soil\_type\_37" "soil\_type\_38"  
## [16] "soil\_type\_39" "soil\_type\_4" "soil\_type\_40" "soil\_type\_5" "soil\_type\_6"   
## [21] "soil\_type\_9"

## One Hot Encoding Training Set

# Create one hot encoder for categorical variables  
onehot\_encoder <- dummyVars(~ wilderness\_area + soil\_type,  
 training\_set[, c("wilderness\_area", "soil\_type")],  
 levelsOnly = T,  
 fullRank = T)  
  
# Use one hot encoder to encode categorical variables  
onehot\_enc\_training <- predict(onehot\_encoder,  
 training\_set[, c("wilderness\_area", "soil\_type")])  
  
# Combine with one hot encoded data with training set  
training\_set <- cbind(training\_set, onehot\_enc\_training)

## One Hot Encoding Test Set

# Repeat previous steps on the test data set  
test\_set$soil\_type <- ifelse(test\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,  
 test\_set$soil\_type,  
 "other")  
  
test\_set$wilderness\_area <- factor(test\_set$wilderness\_area)  
test\_set$soil\_type <- factor(test\_set$soil\_type)  
  
onehot\_enc\_test <- predict(onehot\_encoder, test\_set[, c("wilderness\_area", "soil\_type")])  
test\_set <- cbind(test\_set, onehot\_enc\_test)

## Scaling Numerical features

# Scale test and training set numerical features  
test\_set[, -c(11:13)] <- scale(test\_set[, -c(11:13)],  
 center = apply(training\_set[, -c(11:13)], 2, mean),  
 scale = apply(training\_set[, -c(11:13)], 2, sd))  
training\_set[, -c(11:13)] <- scale(training\_set[, -c(11:13)])

## Create R arrays with pre-processed training and test data

# Convert training and test data into tensors for neural network  
training\_features <- array(data = unlist(training\_set[, -c(11:13)]),  
 dim = c(nrow(training\_set), 33))  
training\_labels <- array(data = unlist(training\_set[, 13]),  
 dim = c(nrow(training\_set)))  
  
test\_features <- array(data = unlist(training\_set[, -c(11:13)]),  
 dim = c(nrow(test\_set), 33))  
test\_labels <- array(data = unlist(training\_set[, 13]),  
 dim = c(nrow(test\_set)))

# Exercises

## Question 1

head(training\_features)

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] -1.6384537 -0.5558474 -0.9561590 -0.410598483 -0.4847989 -1.0538806  
## [2,] -0.2905485 -0.6098401 -0.2739933 -1.133432458 -0.7424029 1.2895821  
## [3,] -0.8275189 -0.1868972 0.2717392 -0.191846095 0.5627907 -0.4961275  
## [4,] -0.3709114 -0.2588875 -0.8197258 -0.001626628 -0.6737085 0.5949610  
## [5,] -0.5060672 -0.7358231 0.8174718 -0.848103257 -0.4504517 0.3385745  
## [6,] -0.7873374 -0.2858839 0.6810386 -1.133432458 -0.7767501 0.2441163  
## [,7] [,8] [,9] [,10] [,11] [,12]  
## [1,] 0.6967673 0.29077514 -0.2882851 2.6131580 -0.2215892 -0.8906707  
## [2,] 0.9256788 -0.12257641 -0.7603459 2.9206975 -0.2215892 -0.8906707  
## [3,] 1.1927423 0.39411303 -0.8127972 -0.8324791 -0.2215892 -0.8906707  
## [4,] 0.7730711 0.60078881 -0.2096083 0.7768783 -0.2215892 -0.8906707  
## [5,] 0.9638307 -1.31096212 -1.5733397 0.1894181 -0.2215892 -0.8906707  
## [6,] 1.3453500 -0.07090746 -1.2586324 0.1782212 -0.2215892 -0.8906707  
## [,13] [,14] [,15] [,16] [,17] [,18]  
## [1,] -0.2532355 -0.07122517 -0.04463561 -0.03500164 -0.5044034 -0.08778692  
## [2,] -0.2532355 -0.07122517 -0.04463561 -0.03500164 -0.5044034 -0.08778692  
## [3,] -0.2532355 -0.07122517 -0.04463561 -0.03500164 -0.5044034 -0.08778692  
## [4,] -0.2532355 -0.07122517 -0.04463561 -0.03500164 -0.5044034 -0.08778692  
## [5,] -0.2532355 -0.07122517 -0.04463561 -0.03500164 1.9822369 -0.08778692  
## [6,] -0.2532355 -0.07122517 -0.04463561 -0.03500164 1.9822369 -0.08778692  
## [,19] [,20] [,21] [,22] [,23] [,24]  
## [1,] -0.2338519 -0.2215892 -0.3207851 -0.2911808 -0.04632411 -0.06069905  
## [2,] 4.2755572 -0.2215892 -0.3207851 -0.2911808 -0.04632411 -0.06069905  
## [3,] 4.2755572 -0.2215892 -0.3207851 -0.2911808 -0.04632411 -0.06069905  
## [4,] -0.2338519 -0.2215892 -0.3207851 -0.2911808 -0.04632411 -0.06069905  
## [5,] -0.2338519 -0.2215892 -0.3207851 -0.2911808 -0.04632411 -0.06069905  
## [6,] -0.2338519 -0.2215892 -0.3207851 -0.2911808 -0.04632411 -0.06069905  
## [,25] [,26] [,27] [,28] [,29] [,30]  
## [1,] -0.01749279 -0.01749279 -0.1568579 -0.144653 -0.1390423 -0.1220747  
## [2,] -0.01749279 -0.01749279 -0.1568579 -0.144653 -0.1390423 -0.1220747  
## [3,] -0.01749279 -0.01749279 -0.1568579 -0.144653 -0.1390423 -0.1220747  
## [4,] -0.01749279 -0.01749279 -0.1568579 -0.144653 -0.1390423 -0.1220747  
## [5,] -0.01749279 -0.01749279 -0.1568579 -0.144653 -0.1390423 -0.1220747  
## [6,] -0.01749279 -0.01749279 -0.1568579 -0.144653 -0.1390423 -0.1220747  
## [,31] [,32] [,33]  
## [1,] -0.04953015 -0.1062619 -0.04288122  
## [2,] -0.04953015 -0.1062619 -0.04288122  
## [3,] -0.04953015 -0.1062619 -0.04288122  
## [4,] -0.04953015 -0.1062619 -0.04288122  
## [5,] -0.04953015 -0.1062619 -0.04288122  
## [6,] -0.04953015 -0.1062619 -0.04288122

head(test\_features)

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] -1.6384537 0.83453232 0.7614751 -0.5558474 -0.6998280 -0.3848705  
## [2,] -0.2905485 0.23911619 0.3487020 -0.6098401 -1.0147855 1.2259121  
## [3,] -0.8275189 0.03455604 0.9185481 -0.1868972 -0.3668729 0.8659607  
## [4,] -0.3709114 1.21442973 1.0171753 -0.2588875 -0.9607928 0.7039825  
## [5,] -0.5060672 0.79800372 1.3349741 -0.7358231 -0.7088268 0.4880117  
## [6,] -0.7873374 0.97699385 1.4189899 -0.2858839 1.7838368 0.4610153  
## [,7] [,8] [,9] [,10] [,11] [,12]  
## [1,] -0.9561590 2.318236250 -1.092592102 -0.410598483 -0.95747945 -0.70543866  
## [2,] -0.2739933 -0.001127029 -1.092592102 -1.133432458 0.31699098 0.15054895  
## [3,] 0.2717392 -1.638324638 -1.229025236 -0.191846095 1.44879681 1.72461504  
## [4,] -0.8197258 0.408172373 -0.001127029 -0.001626628 0.24090319 -1.13343246  
## [5,] 0.8174718 1.363204312 0.135306105 -0.848103257 -0.95747945 -0.04918149  
## [6,] 0.6810386 -0.956158968 -0.410426431 -1.133432458 0.04117275 2.70424529  
## [,13] [,14] [,15] [,16] [,17] [,18]  
## [1,] -0.4847989 -0.3302365 -0.7424029 -1.0538806 1.5935189 -0.05982060  
## [2,] -0.7424029 0.1849715 0.1677979 1.2895821 -0.6297826 -0.44343649  
## [3,] 0.5627907 0.1334507 -0.1756741 -0.4961275 -0.9484986 -1.07765574  
## [4,] -0.6737085 1.1810403 -0.8454445 0.5949610 1.9282993 0.06355336  
## [5,] -0.4504517 -0.5363197 0.1849715 0.3385745 1.5896635 0.73696954  
## [6,] -0.7767501 -0.7767501 1.7134220 0.2441163 2.0593841 0.35335364  
## [,19] [,20] [,21] [,22] [,23] [,24]  
## [1,] 0.6967673 1.0401346 0.65861534 0.29077514 -2.8610304 0.5491199  
## [2,] 0.9256788 0.3152480 -0.33333472 -0.12257641 -0.7942727 0.8074646  
## [3,] 1.1927423 0.3533999 -0.14257509 0.39411303 0.7041267 1.0141404  
## [4,] 0.7730711 0.4297038 -0.63855012 0.60078881 -1.1559553 1.4791609  
## [5,] 0.9638307 1.0782865 -0.18072702 -1.31096212 -1.8276516 1.5308298  
## [6,] 1.3453500 -0.1807270 0.01003261 -0.07090746 0.2907751 1.3758230  
## [,25] [,26] [,27] [,28] [,29] [,30]  
## [1,] -0.2882851 -2.7797175 -0.1309314 2.6131580 -0.42043582 -0.8533798  
## [2,] -0.7603459 -0.6029923 0.8394159 2.9206975 0.26779584 -0.2972707  
## [3,] -0.8127972 0.2362270 0.7869647 -0.8324791 0.00429717 -0.8265074  
## [4,] -0.2096083 -0.9176996 1.2852512 0.7768783 -0.52419308 -1.1123998  
## [5,] -1.5733397 -2.0454006 0.8918671 0.1894181 -0.43835074 -1.1743555  
## [6,] -1.2586324 0.4198063 0.7607391 0.1782212 -0.23158266 -0.6018244  
## [,31] [,32] [,33]  
## [1,] -0.2215892 -0.2215892 -0.2215892  
## [2,] -0.2215892 -0.2215892 -0.2215892  
## [3,] -0.2215892 -0.2215892 -0.2215892  
## [4,] -0.2215892 -0.2215892 -0.2215892  
## [5,] -0.2215892 -0.2215892 -0.2215892  
## [6,] -0.2215892 -0.2215892 -0.2215892

## Question 2

The rank of ‘training\_features’ is 2. The shape is (6537, 33). There are 33 dimension along the second axis.

## Question 3

One situation where scaling numerical variables is important is a neural network that uses gradient descent in its optimization method. If the ranges of each feature is largely different, then the steps in the gradient descent will not update at the same rate for all features, which will cause issues when the optimizer searches for the minimum. Scaling all features resolves this issue and makes the optimization method faster. A second situation where scaling is important is principal components analysis. The features that vary more simply because they have larger scales will be dominant in the creation of the principal components, resulting in a biased model.