Lab 3

Adeline Casali

2024-02-01

Data Pre-Processing for Keras and TensorFlow

# Load libraries and 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

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

data <- read.csv("lab\_3\_data.csv")  
  
# Create testing and training sets  
training\_ind <- createDataPartition(data$lodgepole\_pine,   
 p = 0.75,   
 list = FALSE,   
 times = 1)  
training\_set <- data[training\_ind, ]  
test\_set <- data[-training\_ind, ]  
  
# Assessing, grouping, and factoring categorical variables  
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\_12" "soil\_type\_29" "soil\_type\_30" "soil\_type\_23" "soil\_type\_24"  
## [6] "soil\_type\_22" "soil\_type\_10" "soil\_type\_11" "soil\_type\_33" "soil\_type\_17"  
## [11] "soil\_type\_39" "soil\_type\_13" "soil\_type\_31" "soil\_type\_2" "soil\_type\_1"   
## [16] "soil\_type\_32" "soil\_type\_3" "soil\_type\_6" "soil\_type\_14" "soil\_type\_40"  
## [21] "soil\_type\_35" "soil\_type\_4" "soil\_type\_38" "soil\_type\_27" "soil\_type\_25"  
## [26] "soil\_type\_20" "soil\_type\_16" "soil\_type\_18" "soil\_type\_19" "soil\_type\_8"   
## [31] "soil\_type\_9" "soil\_type\_5" "soil\_type\_28" "soil\_type\_34" "soil\_type\_37"  
## [36] "soil\_type\_21" "soil\_type\_36" "soil\_type\_26"

top\_20\_soil\_types <- training\_set %>%   
 group\_by(soil\_type) %>%   
 summarize(count = n()) %>%   
 arrange(desc(count)) %>%  
 select(soil\_type) %>%   
 top\_n(20)

## Selecting by soil\_type

training\_set$soil\_type <- ifelse(training\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,   
 training\_set$soil\_type,   
 "other")  
  
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\_27" "soil\_type\_28" "soil\_type\_29" "soil\_type\_3"   
## [6] "soil\_type\_30" "soil\_type\_31" "soil\_type\_32" "soil\_type\_33" "soil\_type\_34"  
## [11] "soil\_type\_35" "soil\_type\_36" "soil\_type\_37" "soil\_type\_38" "soil\_type\_39"  
## [16] "soil\_type\_4" "soil\_type\_40" "soil\_type\_5" "soil\_type\_6" "soil\_type\_8"   
## [21] "soil\_type\_9"

# One-hot encoding the training set  
onehot\_encoder <- dummyVars(~ wilderness\_area + soil\_type,   
 training\_set[, c("wilderness\_area", "soil\_type")],   
 levelsOnly = TRUE,   
 fullRank = TRUE)  
  
onehot\_enc\_training <- predict(onehot\_encoder,   
 training\_set[, c("wilderness\_area", "soil\_type")])  
training\_set <- cbind(training\_set, onehot\_enc\_training)  
  
# One-hot encoding the test 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 test and training sets  
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)])  
  
# Convert data sets to tensors  
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(test\_set[, -c(11:13)]),   
 dim = c(nrow(test\_set), 33))  
test\_labels <- array(data = unlist(test\_set[, 13]),   
 dim = c(nrow(test\_set)))

1. Displaying the training and test features.

head(training\_features)

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] -0.37087715 -0.25792821 -0.8282961 0.003839668 -0.67115386 0.6038816  
## [2,] -0.50605898 -0.73436074 0.8148068 -0.835969376 -0.44710371 0.3466254  
## [3,] -0.78738332 -0.28489609 0.6778815 -1.119051076 -0.77456162 0.2518468  
## [4,] -0.11147419 -0.98606095 0.2671058 1.527762821 0.32845450 0.2234777  
## [5,] -0.02013512 0.05669704 -0.1436699 0.579439125 0.08716972 0.4259299  
## [6,] -0.15897051 0.66796896 -0.2805952 -0.944484027 -0.67115386 1.6915786  
## [,7] [,8] [,9] [,10] [,11] [,12]  
## [1,] 0.7796804 0.60277523 -0.2100318 0.80006035 -0.2234819 -0.8942664  
## [2,] 0.9711918 -1.29255564 -1.5822446 0.20247157 -0.2234819 -0.8942664  
## [3,] 1.3542147 -0.06315184 -1.2655801 0.19108169 -0.2234819 -0.8942664  
## [4,] 0.3583552 -0.98520469 -0.7905834 -0.05266163 -0.2234819 -0.8942664  
## [5,] 0.7796804 0.96135134 -0.1308657 -0.28273711 -0.2234819 -0.8942664  
## [6,] -0.4076906 1.42237776 1.1357922 -0.35942894 -0.2234819 -0.8942664  
## [,13] [,14] [,15] [,16] [,17] [,18]  
## [1,] -0.252208 -0.04795369 -0.02474232 -0.506073 -0.09210721 -0.2279725  
## [2,] -0.252208 -0.04795369 -0.02474232 1.975697 -0.09210721 -0.2279725  
## [3,] -0.252208 -0.04795369 -0.02474232 1.975697 -0.09210721 -0.2279725  
## [4,] -0.252208 -0.04795369 -0.02474232 1.975697 -0.09210721 -0.2279725  
## [5,] -0.252208 -0.04795369 -0.02474232 -0.506073 -0.09210721 -0.2279725  
## [6,] -0.252208 -0.04795369 -0.02474232 -0.506073 -0.09210721 4.3858234  
## [,19] [,20] [,21] [,22] [,23] [,24]  
## [1,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [2,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [3,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [4,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [5,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [6,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [,25] [,26] [,27] [,28] [,29] [,30]  
## [1,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [2,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [3,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [4,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [5,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [6,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [,31] [,32] [,33]  
## [1,] -0.1077244 -0.01236832 -0.03913902  
## [2,] -0.1077244 -0.01236832 -0.03913902  
## [3,] -0.1077244 -0.01236832 -0.03913902  
## [4,] -0.1077244 -0.01236832 -0.03913902  
## [5,] -0.1077244 -0.01236832 -0.03913902  
## [6,] -0.1077244 -0.01236832 -0.03913902

head(test\_features)

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] -1.6386635 -0.5545749 -0.9652214 -0.40191077 -0.48157297 -1.0505528  
## [2,] -0.2904988 -0.6085106 -0.2805952 -1.11905108 -0.74009237 1.3008588  
## [3,] -0.8275725 -0.1860139 0.2671058 -0.18488147 0.56973928 -0.4909078  
## [4,] -1.3390713 -1.3186648 -0.8282961 -0.85955952 -0.87796939 -1.3000719  
## [5,] -1.3208035 1.5579090 -0.9652214 -1.11905108 -0.79179625 -1.1408181  
## [6,] 1.0284374 0.4342473 -0.2805952 0.04630192 0.06993509 0.7863465  
## [,7] [,8] [,9] [,10] [,11] [,12]  
## [1,] 0.70307581 0.29542427 -0.2891979 2.6679999 -0.2234819 -0.8942664  
## [2,] 0.93288954 -0.11437699 -0.7641947 2.9808418 -0.2234819 -0.8942664  
## [3,] 1.20100556 0.39787459 -0.8169721 -0.8370444 -0.2234819 -0.8942664  
## [4,] -0.06296997 0.09052364 0.2385762 -0.5325550 -0.2234819 -0.8942664  
## [5,] -0.33108599 0.50032491 0.6871842 -0.3784121 -0.2234819 -0.8942664  
## [6,] 0.05193690 1.42237776 0.7135729 0.9291456 -0.2234819 -0.8942664  
## [,13] [,14] [,15] [,16] [,17] [,18]  
## [1,] -0.252208 -0.04795369 -0.02474232 -0.506073 -0.09210721 -0.2279725  
## [2,] -0.252208 -0.04795369 -0.02474232 -0.506073 -0.09210721 4.3858234  
## [3,] -0.252208 -0.04795369 -0.02474232 -0.506073 -0.09210721 4.3858234  
## [4,] -0.252208 -0.04795369 -0.02474232 1.975697 -0.09210721 -0.2279725  
## [5,] -0.252208 -0.04795369 -0.02474232 -0.506073 -0.09210721 -0.2279725  
## [6,] -0.252208 -0.04795369 -0.02474232 -0.506073 -0.09210721 4.3858234  
## [,19] [,20] [,21] [,22] [,23] [,24]  
## [1,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [2,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [3,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [4,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [5,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [6,] -0.2150528 -0.3137766 -0.2933388 -0.04795369 -0.06318727 -0.01236832  
## [,25] [,26] [,27] [,28] [,29] [,30]  
## [1,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [2,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [3,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [4,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [5,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [6,] -0.01749279 -0.1633894 -0.1516614 -0.143547 -0.1207777 -0.04795369  
## [,31] [,32] [,33]  
## [1,] -0.1077244 -0.01236832 -0.03913902  
## [2,] -0.1077244 -0.01236832 -0.03913902  
## [3,] -0.1077244 -0.01236832 -0.03913902  
## [4,] -0.1077244 -0.01236832 -0.03913902  
## [5,] -0.1077244 -0.01236832 -0.03913902  
## [6,] -0.1077244 -0.01236832 -0.03913902

1. The rank of the tensor “training\_features” is 2. The shape is 6537 x 33 (6537 observations and 33 features). It has 33 dimensions along the second axis.

rank\_of\_tensor <- length(dim(training\_features))  
shape\_of\_tensor <- dim(training\_features)  
dimensions\_second\_axis <- dim(training\_features)[2]  
  
print(paste("Rank of the tensor:", rank\_of\_tensor))

## [1] "Rank of the tensor: 2"

print(paste("Shape of the tensor:", paste(shape\_of\_tensor, collapse = "x")))

## [1] "Shape of the tensor: 6537x33"

print(paste("Dimensions along the second axis:", dimensions\_second\_axis))

## [1] "Dimensions along the second axis: 33"

1. Scaling numerical variables is important when using machine learning methods such as neural networks that rely on gradient descent for optimization. It is also important when a distance is associated with the variables so that one numerical variable doesn’t dominate others simply because of its units.