Megan Cusey

Unit 2 Homework

Summer 2019

Pages 83-119

3.2 In real-world data, tuples with *missing values* for some attributes are a common occurrence. Describe various methods for handling this problem.

Methods of handling missing values for attributes in a tuple include the following:

- Ignoring the tuple,

- Filling in the missing value manually

- Using a global constant to fill in the missing value

- Using a measure of central tendency to fill the missing value

- Using a measure of central tendency of the same class as the tuple

- Use the most probably value to fill in the missing value.

Of these methods, Ignoring the tuple, filling in the missing value manually, and using a globally constant to fill in the missing value are typically undesirable methods.

If there are many missing values for attributes in a tuple, ignoring the tuple may not be a bad choice. Otherwise, remove the tuple also means throwing out other pieces of information that provide useful observations.

Filling in the missing value manually is likely unrealistic in a large data set. Even if there are a fraction of a data set that have missing values, the task of manually finding the values for them can be time consuming.

Using a global constant to fill in the missing value can provide confusing results in a model due to many tuples having the same value. This can be mistaken as a significant find.

The remaining three methods are more desirable methods for filling in missing values. Using a measure of central tendency, either from the whole data set or from within a class the tuple has been identified in, can be an OK approximation of what the missing value would have been. However, going the extra step to calculate what the most probable value would be to fill in the missing value with is going to help maintain relationships between attributes. Calculating the most probable value is most likely more time consuming and more error prone then calculating a measure of central tendency (mean if normal data, median if the data is skewed)

3.6 (a-c only) Use these methods to *normalize* the following group of data:

200, 300, 400, 600, 1000

1. min-max normalization by setting min=0 and max=1

[1] 0.000 0.125 0.250 0.500 1.000

1. z-score normalization

[1] -0.9486833 -0.6324555 -0.3162278 0.3162278 1.5811388

1. z-score normalization using the mean absolute deviation instead of standard deviation

[1] -1.2500000 -0.8333333 -0.4166667 0.4166667 2.0833333

R Script

exercise3.6 <- c(200,300,400,600,1000)

normalize <- function(x) {

return ((x-min(x))/(max(x)-min(x)))

}

normalize(exercise3.6)

## Returns

## [1] 0.000 0.125 0.250 0.500 1.000

mean <- mean(exercise3.6)

mean

sd <- sd(exercise3.6, na.rm=FALSE)

zscorenorm <- function(x) {

return ((x-mean)/sd)

}

zscorenorm(exercise3.6)

##RETURNS

##[1] -0.9486833 -0.6324555 -0.3162278 0.3162278

##[5] 1.5811388

library(DescTools)

meanAD <- MeanAD(exercise3.6, FUN=mean, na.rm=FALSE)

zscorenormMAD <- function(x) {

return ((x-mean)/meanAD)

}

zscorenormMAD(exercise3.6)

##RETURNS [1] -1.2500000 -0.8333333 -0.4166667 0.4166667

##[5] 2.0833333

rm(list = ls()) #clear entire workspace