ISE 5103 Intelligent Data Analytics

Homework 5 - Modeling

Daniel Carpenter & Sonaxy Mohanty

October 2022

Contents

Packages

```r
# Data Wrangling
library(tidyverse)

# Modeling
library(outliers)  # grubbs.test for outlier detection

# Aesthetics
library(knitr)
library(cowplot)  # multiple ggplots on one plot with plot_grid()
library(scales)
library(kableExtra)
```

General Data Prep

Read Data

```r
housingData <- read.csv('housingData.csv')
```

Impute Missing Values with `PMM`

Make dataset of `numeric` variables

```r
housingNumeric <- housingData %>%

  #selecting all the numeric data
  dplyr::select_if(is.numeric) %>%

  #converting the dataframe to tibble
  as_tibble()
```

Make dataset of `character` variables

```r
housingFactor <- housingData %>%

  #selecting all the numeric data
  dplyr::select_if(is.character) %>%

  #converting the dataframe to tibble
  as_tibble()
```

For each column with missing data, impute missing values with `PMM`

- Done with function `imputeWithPMM()` function
- Applys function via `dplyr` logic

- Note `seeImputation()` function to visualize the imputation from prior homework 4, not shown for simplicity in viewing

Create function to impute via `PMM`

```
imputeWithPMM <- function(colWithMissingData) {

  # Using the mice package
  suppressMessages(library(mice))

  # Discover the missing rows
  isMissing <- is.na(colWithMissingData)

  # Create data frame to pass to PMM inputation function from mic package
  df <- data.frame(x       = rexp(length(colWithMissingData)), # meaningless x to help show varation
                   y       = colWithMissingData,
                   missing = isMissing)

  # imputation by PMM
  df[isMissing, "y"] <- mice.impute.pmm( df$y,
                                        !df$missing,
                                         df$x)

  return(df$y)
}
```

Apply `PMM` function to numeric data containing null values

```
# Data to store imputed values with PMM method
housingDataImputed <- housingData

# Which columns has NA's?
colNamesWithNulls <- colnames(housingNumeric[ , colSums(is.na(housingNumeric)) != 0])
colNamesWithNulls
```

```
## [1] "LotFrontage" "MasVnrArea"  "GarageYrBlt"
```

```
numberOfColsWithNulls = length(colNamesWithNulls)


# For each of the numeric columns with null values
for (colWithNullsNum in 1:numberOfColsWithNulls) {

  # The name of the column with null values
  nameOfThisColumn <- colNamesWithNulls[colWithNullsNum]

  # Get the actual data of the column with nulls
  colWithNulls <- housingData[, nameOfThisColumn]

  # Impute the missing values with PMM
  imputedValues <- imputeWithPMM(colWithNulls)

  # Now store the data in the original new frame
```

```
  housingDataImputed[, nameOfThisColumn] <- imputedValues

  # Save a visualization of the imputation
  pmmVisual <- seeImputation(data.frame(y = colWithNulls),
                             data.frame(y = imputedValues),
                             nameOfThisColumn )

  fileToSave = paste0('OutputPMM/Imputation_With_PMM_', nameOfThisColumn, '.pdf')
  print(paste0('For imputation results of ', nameOfThisColumn, ', see ', fileToSave))
  ggsave(pmmVisual, filename = fileToSave,
         height = 11, width = 8.5)
}
```

```
## [1] "For imputation results of LotFrontage, see OutputPMM/Imputation_With_PMM_LotFrontage.pdf"

## [1] "For imputation results of MasVnrArea, see OutputPMM/Imputation_With_PMM_MasVnrArea.pdf"

## [1] "For imputation results of GarageYrBlt, see OutputPMM/Imputation_With_PMM_GarageYrBlt.pdf"
```

Factor Level Collapse - Create `Other` Bin for Columns over 4 Unique Values

```
housingDataCleaned <- housingDataImputed # For final cleaned data

# Get list of factors and the number of unique values
factorCols <- as.data.frame(t(housingFactor %>% summarise_all(n_distinct)))

# We are going to factor collapse factor columns with more than 4 columns
# So there will be 4 of the original, and 1 containing 'other'
# This is the threshold
factorThreshold = 4

# Get a list of the factors we are going to collapse
colsWithManyFactors <- rownames(factorCols %>% filter(V1 > factorThreshold))

# Show a summary of how many factors will be collapsed
numberOfColsWithManyFactors = length(colsWithManyFactors)
paste('Before cleaning, there are', numberOfColsWithManyFactors, 'factor columns with more than',
      factorThreshold, 'unique values')
```

```
## [1] "Before cleaning, there are 14 factor columns with more than 4 unique values"
```

```
# Collapse the affected factors in the original data (the one that already has imputation)

## for each factor column that we are about to collapse
for (collapsedColNum in 1:numberOfColsWithManyFactors) {

  # The name of the column with null values
  nameOfThisColumn <- colsWithManyFactors[collapsedColNum]

  # Get the actual data of the column with nulls
  colWithManyFactors <- housingData[, nameOfThisColumn]

  # lumps all levels except for the n most frequent
  housingDataCleaned[, nameOfThisColumn] <- fct_lump_n(colWithManyFactors,
                                                       n=factorThreshold)
}

# Check to see if the factor lumping worked
factorColsCleaned <- t(housingDataCleaned %>%
                       select_if(is.character) %>%
                       summarise_all(n_distinct))
paste('After cleaning, there are', sum(factorColsCleaned > factorThreshold, na.rm = TRUE),
      "columns with more than", factorThreshold, "unique values (omitting NA's)")
```

```
## [1] "After cleaning, there are 0 columns with more than 4 unique values (omitting NA's)"
```

Remove Outliers from Numeric Data

- Since there are so many outliers, we are only going to remove some outliers
- If you count the number of outliers by column, the 75% of columns contain less than 50 outliers.
- However, some contain up to 200. Since remove ALL outliers would reduce the size of the data to less than 300 observations, we are removing up to 50 per column.

```r
housingDataCleanedNoOutliers <- housingDataCleaned

# Remove up to 75% of the outliers in the dataset
# this is the 3rd quartile of number of outliers.
k_outliers = 50
numOutliers = data.frame() # to store the number of outliers per column

theColNames <- colnames(housingDataCleaned)

for (colNum in 1:ncol(housingDataCleaned)) {

  theCol <- housingDataCleaned[, colNum]
  nrowBefore = length(theCol)
  colName <- theColNames[colNum]


  # Only consider numeric
  if (is.numeric(theCol)) {

        # Identify the outliers in the column
        # Source: https://www.geeksforgeeks.org/remove-outliers-from-data-set-in-r/
        columnOutliers <- boxplot.stats(housingDataCleaned[, colNum])$out
        numOutliers <- rbind(numOutliers, length(columnOutliers))

        # Now remove k outliers from the column
        if (length(columnOutliers) < k_outliers) {

          housingDataCleaned  <- housingDataCleaned %>%

            # If this syntax looks weird, it is just referencing a column in the
            # dataset using dplyr piping. See below for more info:
            # https://stackoverflow.com/questions/48062213/dplyr-using-column-names-as-function-argumen
            # https://stackoverflow.com/questions/72673381/column-names-as-variables-in-dplyr-select-v-
            filter( !( get({{colName}}) %in% columnOutliers ) )
        }
  }
}
paste0('Of the columns with outliers, removed up to 75th percentile of num. outliers.')
```

```
## [1] "Of the columns with outliers, removed up to 75th percentile of num. outliers."
```

```r
paste0('See that the 75th percentile of columns with outliers contain ',
       paste0(summary(numOutliers$X0L)[5]), ' outliers')
```

```
## [1] "See that the 75th percentile of columns with outliers contain 49.5 outliers"
```

1 (a) - OLS Model

1 (b) - PLS Model

1 (c) - LASSO Model

`1 (d)` - Model Variants

`1 (d, i)` - PCR Model

Perform PCA analysis to see how Principal components explain variance

Now, Apply predictions with `PCR`

`1 (d, ii)` - SVR Model

1 (d, iii) - MARS Model