ISE 5103 Intelligent Data Analytics

Homework 5 - Modeling

Daniel Carpenter & Sonaxy Mohanty

October 2022

Contents

Packages

```
# Data Wrangling
library(tidyverse)

# Modeling
library(MASS)

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

General Data Prep

Read Data

```
hd <- read.csv('housingData.csv') %>%

  # creates new variables age, ageSinceRemodel, and ageofGarage, and
  dplyr::mutate(age = YrSold - YearBuilt,
                ageSinceRemodel = YrSold - YearRemodAdd,
                ageofGarage = ifelse(is.na(GarageYrBlt), age, YrSold - GarageYrBlt)) %>%

  # removes the columns used in above the calculations
  dplyr::select(!c(Id,MSSubClass, MiscVal, YrSold ,
                   MoSold, YearBuilt, YearRemodAdd))


# Convert all character data to factor
hd[sapply(hd, is.character)] <-
  lapply(hd[sapply(hd, is.character)], as.factor)
```

Impute Missing Values with `PMM`

Make dataset of `numeric` variables

```
hd.numericRaw <- hd %>%

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

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

Make dataset of `factor` variables

```
hd.factorRaw <- hd %>%

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

  #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
hd.Imputed <- hd

# Which columns has NA's?
colNamesWithNulls <- colnames(hd.numericRaw[ , colSums(is.na(hd.numericRaw)) != 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 <- hd[, nameOfThisColumn]

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

  # Now store the data in the original new frame
  hd.Imputed[, 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

```
hd.Cleaned <- hd.Imputed # For final cleaned data

# Get list of factors and the number of unique values
factorCols <- as.data.frame(t(hd.factorRaw %>% 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 <- hd[, nameOfThisColumn]

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

# Check to see if the factor lumping worked
factorColsCleaned <- t(hd.Cleaned %>%
                       select_if(is.factor) %>%
                       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 14 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.

```
hd.CleanedNoOutliers <- hd.Cleaned

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

theColNames <- colnames(hd.Cleaned)

for (colNum in 1:ncol(hd.Cleaned)) {

  theCol <- hd.Cleaned[, 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(hd.CleanedNoOutliers[, colNum])$out
    numOutliers <- c(numOutliers, length(columnOutliers))

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

      hd.CleanedNoOutliers  <- hd.CleanedNoOutliers %>%

        # 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-arguments
        # https://stackoverflow.com/questions/72673381/column-names-as-variables-in-dplyr-select-v-filt
        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."
```

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

```
## [1] "See that the 75th percentile of columns with outliers contain 51.25 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

- Uses `numeric` data for Principal Component Analyis
- Then appends the `factor` data to the data *without* `NULL` *values*
- Finally, uses `stepAIC()` to best model data
- See interpretation at end

Get cleaned `numeric` and `factor data frames`

```
# After cleaning, two datasets that contain..

## Numeric data ----------------------------------------------------
hd.numericClean <- hd.Cleaned %>% select_if(is.numeric)

## Factors ---------------------------------------------------------
hd.factorClean  <- hd.Cleaned %>% dplyr::select(where(is.factor))

# Removing any columns with NA
removeColsWithNA <- function(df) {
  return( df[ , colSums(is.na(df)) == 0] )
}
hd.factorClean <- removeColsWithNA(hd.factorClean)

paste('Num. factor cols. removed due to null values:',
      ncol(hd.Cleaned %>% dplyr::select(where(is.factor)) ) - ncol(hd.factorClean) )
```

```
## [1] "Num. factor cols. removed due to null values: 16"
```

```
paste(ncol(hd.factorClean), 'factor cols. remain')
```

```
## [1] "22 factor cols. remain"
```

Perform PCA

```
# Principal component analysis on numeric data
pc.house <- prcomp(hd.numericClean %>% dplyr::select(-SalePrice), # do not include response var
                   center = TRUE, # Mean centered
                   scale  = TRUE  # Z-SCore standardized
                   )

# See first 10 cumulative proportions
pc.house.summary <- summary(pc.house)
pc.house.summary$importance[, 1:10]
```

```
##                             PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation     2.648356 1.851874 1.598881 1.393593 1.175993 1.107411
## Proportion of Variance 0.226250 0.110630 0.082470 0.062650 0.044610 0.039560
## Cumulative Proportion  0.226250 0.336880 0.419340 0.481990 0.526600 0.566160
##                             PC7      PC8     PC9     PC10
## Standard deviation     1.062762 1.032981 1.01364 1.005501
## Proportion of Variance 0.036430 0.034420 0.03314 0.032610
## Cumulative Proportion  0.602600 0.637020 0.67016 0.702780
```

Now we choose number of PC's that explain 75% of the variation

- Note this threshold is just a judgement call. No significance behind 75%

```
cumPropThreshold = 0.75 # The threshold

numPCs <- sum(pc.house.summary$importance['Cumulative Proportion', ] < cumPropThreshold)
paste0('There are ', numPCs, ' principal components that explain up to ', cumPropThreshold*100,
       '% of the variation in the data')
```

```
## [1] "There are 11 principal components that explain up to 75% of the variation in the data"
```

```
chosenPCs <- as.data.frame(pc.house$x[, 1:numPCs])
```

Join on the factor data

```
df.pcr <- cbind(SalePrice = hd.numericClean$SalePrice, chosenPCs, hd.factorClean)
```

Now, Apply predictions with `PCR`

- Linear model containing:
  - Principal components explaining 75% of variation in `numeric` data
  - Non-null `factor` data
  - *Predicted variable:* `log(SalePrice)`
- Then use `stepAIC()` to identify which variables are actually important for model

```
# Fit data using PC's, non-null factors
fit.pcr <- lm(log(SalePrice) ~ ., data = df.pcr)

# Reduce to only important variables
fit.pcrReduced <- stepAIC(fit.pcr, direction="both")
```

```
# View results
summary(fit.pcrReduced)
```

```
##
## Call:
## lm(formula = log(SalePrice) ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC7 +
##     PC8 + PC9 + PC10 + MSZoning + LandContour + LotConfig + Condition1 +
```

```
##     BldgType + HouseStyle + RoofStyle + Exterior1st + ExterQual +
##     ExterCond + Foundation + CentralAir + KitchenQual + Functional +
##     PavedDrive, data = df.pcr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67338 -0.06018  0.00342  0.06411  0.31634
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         11.804075   0.054382 217.057  < 2e-16 ***
## PC1                  0.098409   0.002410  40.837  < 2e-16 ***
## PC2                  0.005929   0.003400   1.744 0.081510 .
## PC3                 -0.054548   0.003355 -16.260  < 2e-16 ***
## PC4                 -0.018806   0.003648  -5.156 3.08e-07 ***
## PC5                  0.052928   0.003956  13.378  < 2e-16 ***
## PC7                  0.005361   0.003404   1.575 0.115568
## PC8                 -0.013641   0.003609  -3.780 0.000167 ***
## PC9                  0.012432   0.003624   3.430 0.000629 ***
## PC10                 0.004971   0.003474   1.431 0.152845
## MSZoningRH          -0.062565   0.040045  -1.562 0.118531
## MSZoningRL          -0.038132   0.020405  -1.869 0.061958 .
## MSZoningRM          -0.116062   0.022037  -5.267 1.72e-07 ***
## LandContourHLS       0.083299   0.026918   3.095 0.002029 **
## LandContourLow      -0.002654   0.028695  -0.092 0.926325
## LandContourLvl      -0.006376   0.018229  -0.350 0.726588
## LotConfigCulDSac     0.039119   0.015134   2.585 0.009890 **
## LotConfigInside      0.001539   0.009109   0.169 0.865823
## LotConfigother      -0.010693   0.019271  -0.555 0.579126
## Condition1Feedr      0.055353   0.024851   2.227 0.026155 *
## Condition1Norm       0.095197   0.020572   4.627 4.22e-06 ***
## Condition1RR         0.051127   0.029423   1.738 0.082598 .
## Condition1Other      0.024667   0.031410   0.785 0.432463
## BldgType2fmCon       0.029071   0.027684   1.050 0.293943
## BldgTypeDuplex       0.034616   0.026677   1.298 0.194756
## BldgTypeTwnhs       -0.053884   0.021956  -2.454 0.014300 *
## BldgTypeTwnhsE      -0.007451   0.015168  -0.491 0.623376
## HouseStyle1Story    -0.066839   0.015622  -4.278 2.07e-05 ***
## HouseStyle2Story    -0.012939   0.015251  -0.848 0.396446
## HouseStyleSLvl      -0.035469   0.020623  -1.720 0.085786 .
## HouseStyleOther     -0.054730   0.020212  -2.708 0.006895 **
## RoofStyleHip         0.015557   0.009405   1.654 0.098415 .
## RoofStyleother       0.107711   0.024744   4.353 1.49e-05 ***
## Exterior1stMetalSd   0.025488   0.012772   1.996 0.046261 *
## Exterior1stVinylSd   0.024772   0.011459   2.162 0.030878 *
## Exterior1stWd Sdng  -0.006666   0.013407  -0.497 0.619168
## Exterior1stOther     0.033635   0.011685   2.878 0.004087 **
## ExterQualAvg        -0.040517   0.011716  -3.458 0.000568 ***
## ExterQualBelowAvg   -0.118522   0.046788  -2.533 0.011464 *
## ExterCondAvg         0.019835   0.011695   1.696 0.090209 .
## ExterCondBelowAvg   -0.011675   0.032754  -0.356 0.721594
## FoundationCBlock     0.003687   0.014323   0.257 0.796894
## Foundationother      0.019984   0.025908   0.771 0.440684
## FoundationPConc      0.052362   0.016725   3.131 0.001798 **
```

```
## CentralAirY          0.055715   0.017245   3.231 0.001277 **
## KitchenQualAvg      -0.022087   0.010461  -2.111 0.034996 *
## KitchenQualBelowAvg -0.036908   0.025511  -1.447 0.148296
## FunctionalMaj2      -0.218162   0.062128  -3.511 0.000467 ***
## FunctionalMin1       0.017979   0.039177   0.459 0.646406
## FunctionalMin2       0.016553   0.038373   0.431 0.666297
## FunctionalMod       -0.008600   0.044584  -0.193 0.847081
## FunctionalTyp        0.087328   0.032176   2.714 0.006768 **
## PavedDriveP         -0.006781   0.025436  -0.267 0.789857
## PavedDriveY          0.048822   0.016331   2.989 0.002867 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1045 on 946 degrees of freedom
## Multiple R-squared:  0.9216, Adjusted R-squared:  0.9172
## F-statistic: 209.8 on 53 and 946 DF,  p-value: < 2.2e-16
```

Interpretation of PCR Model

Please note all interpretations below are approximate, given the `stepAIC()` uses stochastic modeling.

Model performance evaluation:

- See that around 28 of the variables cannot be explained by random chance, with a probability of 90% or more (see significance codes above)
- Standard errors range from ± 1-5%, with average around 2%. Larger values may indicate higher uncertainty of the estimated coefficients.
- This model explains around 92% of the variation in the `log(SalePrice)`. See Adjusted R-Squared for reference.
- Note this model may exhibit selection bias, since the data excludes factor data with null values in the variable.
- This model would likely doe well for prediction of `log(SalePrice)`, given the small range of standard errors, high adjusted R squared, and number of significant variables. This model would obviously not do well for inference, given we are using principal components that mask the numeric data.

Practical significance evaluation:

- The principal components contribute positively about 20% of the sale price of the home
- Residential Medium Density (`MSZoningRM`) reduces the home price by around 12%, with a standard error of around 2%.
- If the exterior quality is below average (`ExterQualBelowAvg`), it reduces the home price by around 12%, with a standard error of around 5%.
- If the functionality of the home has 2 major deductions (`FunctionalMaj2`), it reduces the home price by around 20%, with a standard error of around 6%. While having typical functionality (`FunctionalTyp`) increases the home sale price by nearly 10%, with a standard error of 3%.
- See other coefficients of the data for other variables.

`1 (d, ii)` - SVR Model

1 (d, iii) - MARS Model