ISE 5103 Intelligent Data Analytics

Homework 5 - Modeling

Daniel Carpenter & Sonaxy Mohanty

October 2022

Contents

Packages . . . 2
General Data Prep . . . 2
Read Data . . . 2
Impute Missing Values with `PMM` . . . 2
Factor Level Collapse - Create `Other` Bin for Columns over `4` Unique Values . . . 5
Remove Outliers from Numeric Data . . . 6

`1 (a)` - OLS Model **7**

`1 (b)` - PLS Model **8**

`1 (c)` - LASSO Model **9**

`1 (d)` - Model Variants **10**
`1 (d, i)` - PCR Model . . . 10
Perform PCA analysis to see how Principal components explain variance . . . 10
Now, Apply predictions with `PCR` . . . 11
Interpretation of PCR Model . . . 13
`1 (d, ii)` - SVR Model . . . 14
`1 (d, iii)` - MARS Model . . . 15

Packages

```r
# 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

```r
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

```r
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

```r
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`

```r
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

```r
# 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"
```

```r
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-filte
        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 50.25 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     2.657147 1.84832 1.609823 1.393088 1.171847 1.107559
## Proportion of Variance 0.227760 0.11020 0.083600 0.062600 0.044300 0.039570
## Cumulative Proportion  0.227760 0.33796 0.421560 0.484160 0.528460 0.568030
##                            PC7      PC8      PC9     PC10
## Standard deviation     1.060806 1.034741 1.008248 1.005577
## Proportion of Variance 0.036300 0.034540 0.032790 0.032620
## Cumulative Proportion  0.604330 0.638870 0.671660 0.704280
```

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 + MSZoning + LandContour + LotConfig + Condition1 +
```

```
##     BldgType + HouseStyle + RoofStyle + Exterior1st + ExterQual +
##     Foundation + CentralAir + KitchenQual + Functional + PavedDrive,
##     data = df.pcr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67073 -0.06094  0.00279  0.06714  0.31587
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         11.816119   0.053210 222.067  < 2e-16 ***
## PC1                  0.098443   0.002399  41.033  < 2e-16 ***
## PC2                  0.004896   0.003404   1.438 0.150693
## PC3                 -0.053829   0.003397 -15.845  < 2e-16 ***
## PC4                 -0.018459   0.003634  -5.079 4.57e-07 ***
## PC5                  0.053802   0.003954  13.606  < 2e-16 ***
## PC7                  0.009082   0.003407   2.665 0.007819 **
## PC8                 -0.013184   0.003562  -3.701 0.000227 ***
## PC9                  0.007889   0.003647   2.163 0.030770 *
## MSZoningRH          -0.055675   0.040153  -1.387 0.165897
## MSZoningRL          -0.036339   0.020456  -1.776 0.075990 .
## MSZoningRM          -0.113188   0.022046  -5.134 3.44e-07 ***
## LandContourHLS       0.082312   0.026970   3.052 0.002336 **
## LandContourLow      -0.004804   0.028840  -0.167 0.867737
## LandContourLvl      -0.005914   0.018233  -0.324 0.745736
## LotConfigCulDSac     0.044585   0.015183   2.937 0.003399 **
## LotConfigInside      0.006491   0.009151   0.709 0.478304
## LotConfigother      -0.002854   0.019360  -0.147 0.882818
## Condition1Feedr      0.052676   0.024908   2.115 0.034706 *
## Condition1Norm       0.093188   0.020614   4.521 6.95e-06 ***
## Condition1RR         0.053171   0.029508   1.802 0.071874 .
## Condition1Other      0.019094   0.031505   0.606 0.544619
## BldgType2fmCon       0.035597   0.027343   1.302 0.193278
## BldgTypeDuplex       0.056117   0.026551   2.114 0.034814 *
## BldgTypeTwnhs       -0.046161   0.022101  -2.089 0.037009 *
## BldgTypeTwnhsE      -0.003571   0.015351  -0.233 0.816109
## HouseStyle1Story    -0.066389   0.015610  -4.253 2.32e-05 ***
## HouseStyle2Story    -0.010274   0.015247  -0.674 0.500584
## HouseStyleSLvl      -0.033754   0.020576  -1.640 0.101247
## HouseStyleOther     -0.055199   0.020248  -2.726 0.006526 **
## RoofStyleHip         0.017808   0.009405   1.893 0.058599 .
## RoofStyleother       0.105921   0.024857   4.261 2.24e-05 ***
## Exterior1stMetalSd   0.024019   0.012770   1.881 0.060287 .
## Exterior1stVinylSd   0.023388   0.011479   2.037 0.041892 *
## Exterior1stWd Sdng  -0.005424   0.013439  -0.404 0.686589
## Exterior1stOther     0.033857   0.011735   2.885 0.004000 **
## ExterQualAvg        -0.037753   0.011765  -3.209 0.001377 **
## ExterQualBelowAvg   -0.135204   0.045739  -2.956 0.003194 **
## FoundationCBlock     0.002849   0.014310   0.199 0.842210
## Foundationother      0.025238   0.025736   0.981 0.327007
## FoundationPConc      0.053678   0.016765   3.202 0.001411 **
## CentralAirY          0.055811   0.017309   3.224 0.001306 **
## KitchenQualAvg      -0.025722   0.010425  -2.467 0.013789 *
## KitchenQualBelowAvg -0.048222   0.025028  -1.927 0.054317 .
```

```
## FunctionalMaj2      -0.223632   0.062105  -3.601 0.000334 ***
## FunctionalMin1       0.024115   0.039218   0.615 0.538782
## FunctionalMin2       0.022843   0.038109   0.599 0.549033
## FunctionalMod       -0.006688   0.044736  -0.149 0.881193
## FunctionalTyp        0.088687   0.032056   2.767 0.005774 **
## PavedDriveP         -0.010921   0.025435  -0.429 0.667746
## PavedDriveY          0.045561   0.016324   2.791 0.005360 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1049 on 949 degrees of freedom
## Multiple R-squared:  0.9207, Adjusted R-squared:  0.9165
## F-statistic: 220.3 on 50 and 949 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