ISE 5103 Intelligent Data Analytics

Homework 5 - Modeling

Daniel Carpenter & Sonaxy Mohanty

October 2022

Contents

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

```
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"
```

```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

```r
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"
```

```r
# 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 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     2.64807 1.851188 1.61109 1.394719 1.17239 1.10602
## Proportion of Variance 0.22620 0.110550 0.08373 0.062750 0.04434 0.03946
## Cumulative Proportion  0.22620 0.336750 0.42048 0.483230 0.52757 0.56703
##                             PC7     PC8      PC9     PC10
## Standard deviation     1.062042 1.03686 1.007337 1.004871
## Proportion of Variance 0.036380 0.03468 0.032730 0.032570
## Cumulative Proportion  0.603410 0.63809 0.670820 0.703400
```

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 +
##     Foundation + CentralAir + KitchenQual + Functional + PavedDrive,
##     data = df.pcr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67241 -0.05915  0.00483  0.06501  0.31518
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         11.813758   0.053233 221.927  < 2e-16 ***
## PC1                  0.098800   0.002406  41.064  < 2e-16 ***
## PC2                  0.006409   0.003399   1.886 0.059624 .
## PC3                 -0.053215   0.003364 -15.820  < 2e-16 ***
## PC4                 -0.019187   0.003624  -5.294 1.49e-07 ***
## PC5                  0.052859   0.003957  13.358  < 2e-16 ***
## PC7                  0.009032   0.003405   2.653 0.008119 **
## PC8                  0.012671   0.003560   3.560 0.000389 ***
## PC9                  0.004952   0.003487   1.420 0.155889
## PC10                 0.010865   0.003642   2.983 0.002928 **
## MSZoningRH          -0.061475   0.040152  -1.531 0.126091
## MSZoningRL          -0.035991   0.020459  -1.759 0.078862 .
## MSZoningRM          -0.114250   0.022064  -5.178 2.74e-07 ***
## LandContourHLS       0.079256   0.026984   2.937 0.003392 **
## LandContourLow      -0.001530   0.028738  -0.053 0.957547
## LandContourLvl      -0.007821   0.018249  -0.429 0.668316
## LotConfigCulDSac     0.047788   0.015221   3.140 0.001744 **
## LotConfigInside      0.005195   0.009133   0.569 0.569605
## LotConfigother      -0.003358   0.019332  -0.174 0.862122
## Condition1Feedr      0.054981   0.024901   2.208 0.027487 *
## Condition1Norm       0.096075   0.020605   4.663 3.57e-06 ***
## Condition1RR         0.052206   0.029468   1.772 0.076780 .
## Condition1Other      0.027030   0.031514   0.858 0.391271
## BldgType2fmCon       0.025559   0.027575   0.927 0.354228
## BldgTypeDuplex       0.039516   0.027057   1.460 0.144497
## BldgTypeTwnhs       -0.048909   0.021980  -2.225 0.026301 *
## BldgTypeTwnhsE      -0.003510   0.015229  -0.230 0.817755
## HouseStyle1Story    -0.065523   0.015615  -4.196 2.97e-05 ***
## HouseStyle2Story    -0.013115   0.015268  -0.859 0.390562
## HouseStyleSLvl      -0.031828   0.020607  -1.545 0.122800
## HouseStyleOther     -0.054804   0.020250  -2.706 0.006925 **
## RoofStyleHip         0.015142   0.009430   1.606 0.108659
## RoofStyleother       0.102179   0.024865   4.109 4.31e-05 ***
## Exterior1stMetalSd   0.024658   0.012761   1.932 0.053615 .
## Exterior1stVinylSd   0.022464   0.011476   1.957 0.050583 .
## Exterior1stWd Sdng  -0.005930   0.013426  -0.442 0.658843
## Exterior1stOther     0.034101   0.011709   2.912 0.003671 **
## ExterQualAvg        -0.038199   0.011753  -3.250 0.001194 **
## ExterQualBelowAvg   -0.128353   0.045781  -2.804 0.005156 **
## FoundationCBlock     0.004510   0.014286   0.316 0.752285
## Foundationother      0.027254   0.025729   1.059 0.289749
## FoundationPConc      0.056297   0.016672   3.377 0.000763 ***
## CentralAirY          0.056031   0.017270   3.244 0.001218 **
## KitchenQualAvg      -0.023099   0.010479  -2.204 0.027743 *
```

```
## KitchenQualBelowAvg -0.044931   0.025053  -1.793 0.073223 .
## FunctionalMaj2      -0.213234   0.062147  -3.431 0.000627 ***
## FunctionalMin1       0.024312   0.039197   0.620 0.535249
## FunctionalMin2       0.017339   0.038158   0.454 0.649644
## FunctionalMod       -0.016132   0.044695  -0.361 0.718220
## FunctionalTyp        0.087396   0.032083   2.724 0.006566 **
## PavedDriveP         -0.008990   0.025490  -0.353 0.724415
## PavedDriveY          0.047632   0.016364   2.911 0.003691 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1048 on 948 degrees of freedom
## Multiple R-squared:  0.921,  Adjusted R-squared:  0.9167
## F-statistic: 216.6 on 51 and 948 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.

Visualizatoin of PCR Model - Predicted vs. Actuals

Function to compare predicted vs. observed values

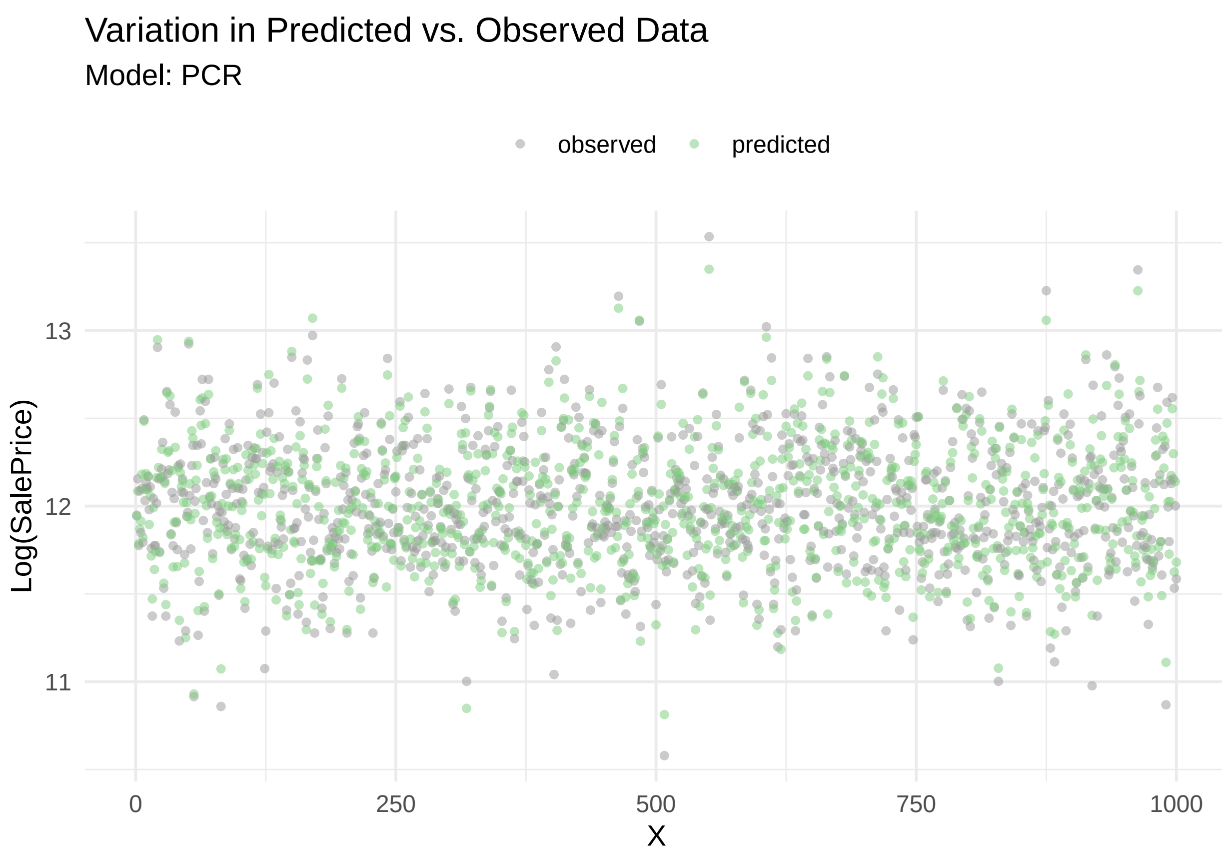
```
# Function to compare predicted vs. actual (observed) regression outputs
predictedVsObserved <- function(predicted, observed, modelName, outcomeName = 'Log(SalePrice)') {

  ## Create dataset for predicted vs. actuals
  comparison <- data.frame(observed  = observed,
                           predicted = predicted) %>%

    # Row index
    mutate(ID = row_number()) %>%

    # Put in single column
    pivot_longer(cols      = c('observed', 'predicted'),
                 names_to  = 'metric',
                 values_to = 'value')


  # Plot --- Observed vs. Actuals across all variables in data
  variationScatter <- comparison %>%
    ggplot(aes(x     = ID,
               y     = value,
               color = metric
               )
           ) +
    geom_point(alpha = 0.5, size = 1) +

    labs(title = 'Variation in Predicted vs. Observed Data',
         subtitle = paste('Model:', modelName),
         x = 'X', y = outcomeName) +
    theme_minimal() + theme(legend.title = element_blank(),
                            legend.position = 'top') +
    scale_color_manual(values = c('grey60', 'palegreen3'))


  print(variationScatter)

  # Limit for x and y axis for scatter of predicted vs. observed
  axisLim = c( min(c(predicted, observed)), max(c(predicted, observed)) )


  # Simple comparison of data
  plot(x = observed,
       y = predicted,
       main = 'PCR Model - Actual (Observed) vs. Predicted\n',
       xlab = paste('Observed Values -', outcomeName),
       ylab = paste('Predicted Values -', outcomeName),
       pch  = 16,
       cex  = 0.75,
       col  = alpha('steelblue3', 1/4),
       xlim = axisLim,
       ylim = axisLim
```

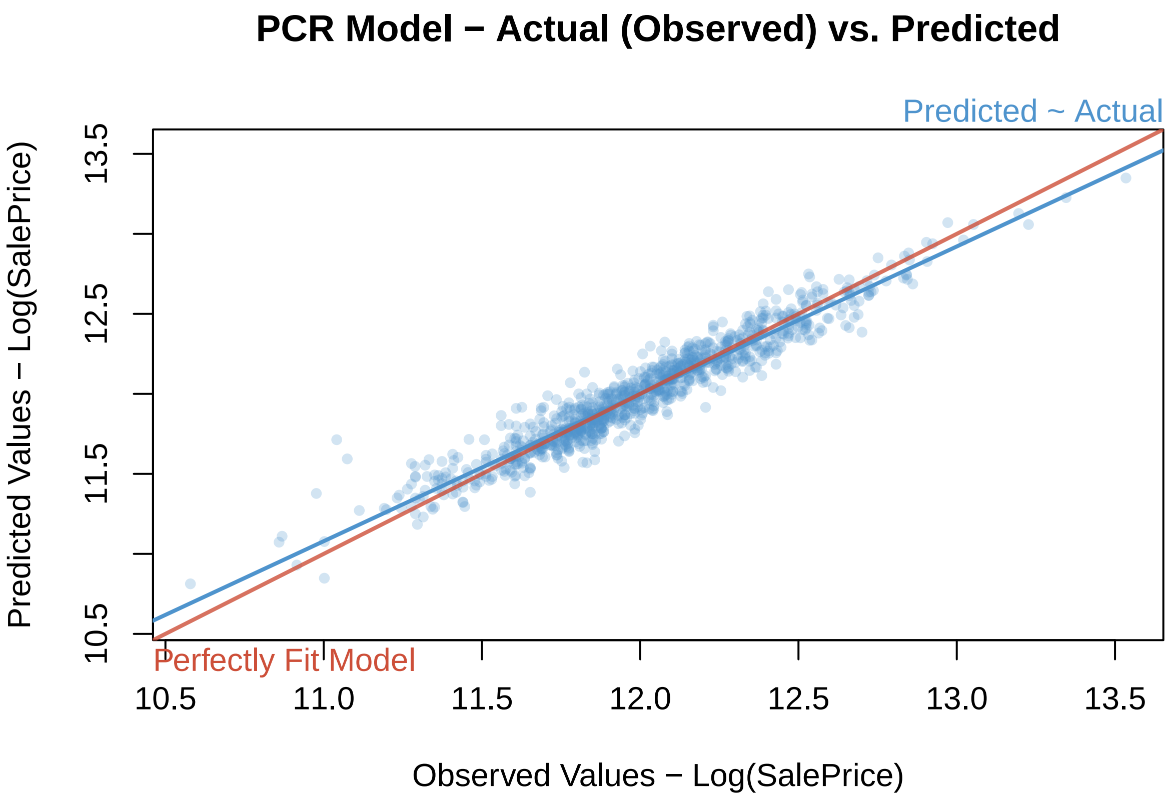
```
  )

  # Add the Predicted vs. actual line
  abline(lm(predicted ~ observed), col = 'steelblue3', lwd = 2)
  mtext('Predicted ~ Actual', side = 3, adj = 1, col = 'steelblue3')

  # Add line for perfectly fit model
  abline(0,1, col = alpha('tomato3', 0.8), lwd = 2)
  mtext('Perfectly Fit Model', side = 1, adj = 0, col = 'tomato3')
}
```

View results of the PCR Model

- See that the variation in the data is very closely resembled actual by changes in independent variables
- Implication? This model fits its own data well, but what is not know if it can predict out of sample data.
- Note that it the data (blue) deviates slightly from perfect line model (red), indicating that the model is slightly skewed from predicted and actual data.

```
# How do the two compare?
predictedVsObserved(observed  = log(df.pcr$SalePrice),
                    predicted = predict(fit.pcrReduced),
                    modelName = 'PCR')
```




PCR Model − Actual (Observed) vs. Predicted
Predicted ~ Actual
Perfectly Fit Model
Predicted Values − Log(SalePrice)
10.5
11.5
12.5
13.5
10.5
11.0
11.5
12.0
12.5
13.0
13.5
Observed Values − Log(SalePrice)

1 (d, ii) - SVR Model

`1 (d, iii)` - MARS Model