ISE 5103 Intelligent Data Analytics

Homework 5 - Modeling

Daniel Carpenter & Sonaxy Mohanty

October 2022

Contents

Packages

```
# Data Wrangling
library(tidyverse)

# Modeling
library(MASS)
library(caret) # Modeling variants like SVM
library(earth) # Modeling with Mars
library(pls)  #Modeling with PLS
library(glmnet) #Modeling with LASSO

# Aesthetics
library(knitr)
library(cowplot)  # multiple ggplots on one plot with plot_grid()
library(scales)
library(kableExtra)
library(ggplot2)

#Hold-out Validation
library(caTools)

#Data Correlation
library(GGally)
library(regclass)

#RMSE Calculation
library(Metrics)

#p-value for OLS model
library(broom)

#ncvTest
library(car)
```

General Data Prep

Read Data

```
# Convert all character data to factor
hd <- read.csv('housingData.csv', stringsAsFactors = TRUE) %>%

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

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


#str(hd)
```

Impute Missing Values with `PMM`

Make data set of `numeric` variables

Make data set of `factor` variables

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

- Done with function `imputeWithPMM()` function

- Applies 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`

Apply `PMM` function to numeric data containing null values

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

```
## [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

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

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

```
## [1] "See that the 75th percentile of columns with outliers contain 51.75 outliers"
```

Exploratory Data Analysis

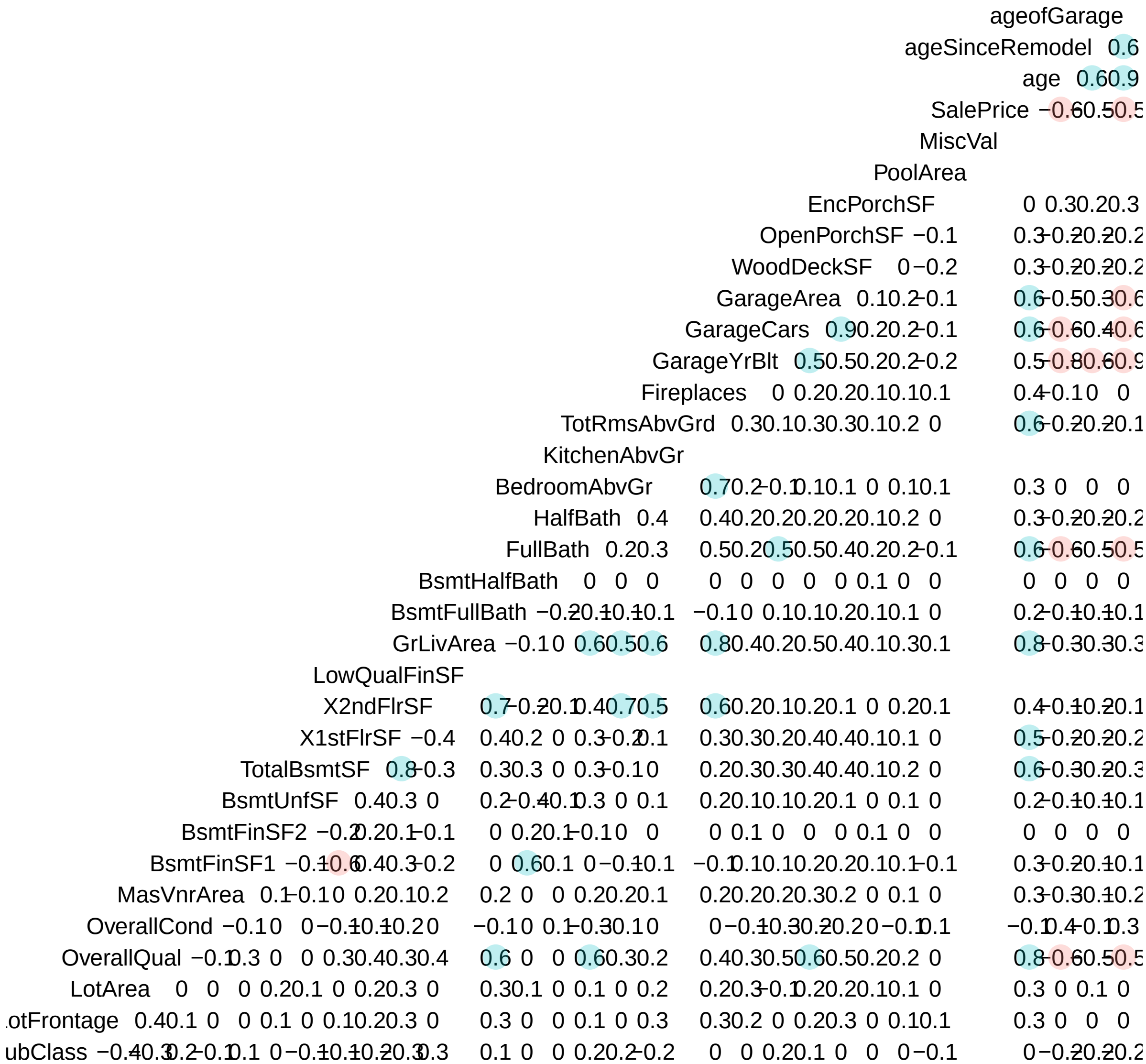
Checking the distribution of Sale Price of houses

```r
hist(hd.CleanedNoOutliers$SalePrice,
     col = 'skyblue4',
     main = 'Distribution of Sale Price of houses',
     xlab = 'House Price')
```

- After removing the desired outliers, we can see that the distribution of Sale Price looks like a normal distribution with few outliers on the right tail.

Correlation between features in the dataset

```r
ggcorr(hd.CleanedNoOutliers, geom='blank', label=T, label_size=3, hjust=1,
       size=3, layout.exp=2) +
  geom_point(size = 4, aes(color = coefficient > 0, alpha = abs(coefficient) >= 0.5)) +
  scale_alpha_manual(values = c("TRUE" = 0.25, "FALSE" = 0)) +
  guides(color = F, alpha = F)
```



- We can see that `SalePrice` has strong correlations with `GarageArea`, `GarageCars`, `TotRmsAbvGrd`, `FullBath`, `GrLivArea`, `X1stFlrSF`, `TotalBsmtSF`, `OverallQual`.

1 (a) - OLS Model

i.

Hold-out validation set

- Since, we have deleted some of the outlier values during data pre-processing, using 10% of the data as test and remaining 90% as train

```
idx <- sample(nrow(hd.CleanedNoOutliers), nrow(hd.CleanedNoOutliers)*0.1)
test <- hd.CleanedNoOutliers[idx,]
train <- hd.CleanedNoOutliers[-idx,]
```

Fit the OLS Model

`Model 1:`

- Linear model containing:
  - *Independent variables:* `GarageArea + GarageCars + TotRmsAbvGrd + FullBath + GrLivArea + X1stFlrSF + TotalBsmtSF + OverallQual`
  - *Predicted variable:* `SalePrice`

```
ols.mdl1 <- lm(SalePrice ~ GarageArea + GarageCars + TotRmsAbvGrd
              + FullBath + GrLivArea + X1stFlrSF + TotalBsmtSF + OverallQual, data=train)
```

- **For Model 1**: Adjusted R-squared is `0.821`, AIC is `16695.31` and BIC is `16741.28` and RMSE is `20541.22`.

- Still trying to improve the existing model.

- No multicollinearity detected.

`Model 2:`

- This model created is based on `Principal Component Analysis`.
  - Uses `numeric` data for Principal Component Analysis

  - Then appends the `factor` data to the data *without `NULL` values*

  - Finally, uses `stepAIC()` to best model data

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%

```
## [1] "There are 9 principal components that explain up to 75% of the variation in the data"
```

Fit the Model

- Linear model containing:
  - Principal components explaining 75% of variation in `numeric` data
  - Non-null `factor` data

  - *Predicted variable:* `SalePrice`
- Then use `stepAIC()` to identify which variables are actually important for model

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

```r
# Reduce to only important variables
ols.mdl2 <- stepAIC(fit.ols, direction="both")
```

- Reporting `all the variables` of the best model (`Model 2`):

Coefficient estimates:

```
##                         Estimate Std. Error      t value      Pr(>|t|)
## (Intercept)         132084.03696 10595.8674  12.46561817  2.991747e-32
## PC1                  12562.26171   475.1182  26.44028795 2.507707e-106
## PC3                  -6602.94706   730.4852  -9.03912523  1.644606e-18
## PC4                  -3506.79243   521.5626  -6.72362733  3.774185e-11
## PC5                   1642.77240   660.5434   2.48700148  1.312263e-02
## PC6                  -7099.91044   708.8347 -10.01631302  4.110301e-22
## PC7                  -4157.93100   641.9053  -6.47748309  1.795685e-10
## PC8                  -1289.27140   630.8678  -2.04364756  4.137549e-02
## PC9                   2488.81664   648.7043   3.83659662  1.364335e-04
## MSZoningRH          -16613.40293  8836.3403  -1.88012258  6.052115e-02
## MSZoningRL           -5258.83266  3824.4485  -1.37505648  1.695695e-01
## MSZoningRM          -13506.46541  4230.2960  -3.19279438  1.474376e-03
## LandContourHLS       13972.61581  5334.1867   2.61944633  9.004838e-03
## LandContourLow        9540.10875  5735.3207   1.66339587  9.669671e-02
## LandContourLvl          64.73445  3467.5304   0.01866875  9.851109e-01
## LotConfigCulDSac      4274.74389  2844.1948   1.50297153  1.333134e-01
## LotConfigInside      -1838.95995  1765.6924  -1.04149507  2.980182e-01
## LotConfigother       -5876.77666  3650.4997  -1.60985542  1.078963e-01
## NeighborhoodNAmes    -7654.95043  2813.1977  -2.72108512  6.674244e-03
## NeighborhoodOldTown  -3948.31147  3920.4195  -1.00711454  3.142401e-01
## Neighborhoodother    -2980.34849  3420.4132  -0.87134165  3.838770e-01
## NeighborhoodOther     -797.10902  2218.9501  -0.35922801  7.195367e-01
## Condition1Feedr       2878.23898  4982.7025   0.57764616  5.636954e-01
## Condition1Norm       10734.21247  4130.6829   2.59865325  9.563062e-03
## Condition1RR          -322.85461  5575.1128  -0.05790997  9.538375e-01
## Condition1Other       1953.48688  6811.7895   0.28678028  7.743684e-01
## BldgType2fmCon      -14195.57637  7464.0298  -1.90186491  5.761364e-02
## BldgTypeDuplex       -6658.56135  8781.2982  -0.75826617  4.485559e-01
## BldgTypeTwnhs       -13769.29172  4896.2727  -2.81219871  5.063128e-03
## BldgTypeTwnhsE       -3545.85572  3923.0796  -0.90384495  3.663996e-01
## HouseStyle1Story     -4779.78828  2627.0620  -1.81944250  6.928615e-02
## HouseStyle2Story      4416.44039  2895.5935   1.52522804  1.276699e-01
## HouseStyleSLvl       -3546.26247  3901.8213  -0.90887361  3.637408e-01
## HouseStyleOther      -4007.69457  3776.4927  -1.06122131  2.889683e-01
## RoofStyleHip          2937.37319  1790.3806   1.64064180  1.013370e-01
## RoofStyleother       24330.40171  5534.2609   4.39632361  1.278419e-05
## Exterior1stMetalSd    8454.65291  7172.9232   1.17869000  2.389364e-01
## Exterior1stVinylSd   -2276.67820  7385.5030  -0.30826312  7.579771e-01
## Exterior1stWd Sdng   -6159.99005  5194.6354  -1.18583685  2.361034e-01
## Exterior1stOther      9225.20645  3671.2218   2.51284366  1.220784e-02
## Exterior2ndMetalSd    -914.74166  7290.6898  -0.12546709  9.001911e-01
## Exterior2ndVinylSd    9503.71922  7542.2462   1.26006483  2.080808e-01
## Exterior2ndWd Sdng   11037.76569  5309.4984   2.07887164  3.800568e-02
## Exterior2ndOther     -3442.93039  3631.0054  -0.94820305  3.433649e-01
## ExterQualAvg        -10817.79756  2236.4785  -4.83697814  1.633866e-06
## ExterQualBelowAvg    27451.15724 11846.0210   2.31733148  2.078350e-02
## ExterCondAvg          4706.41525  2178.2433   2.16064721  3.107323e-02
```

```
## ExterCondBelowAvg      683.82765  6425.4413   0.10642501  9.152767e-01
## KitchenQualAvg       -4410.58396  1906.6264  -2.31329216  2.100578e-02
## KitchenQualBelowAvg   1613.53530  4774.9596   0.33791601  7.355314e-01
## FunctionalMaj2      -19264.90742 14110.3202  -1.36530618  1.726110e-01
## FunctionalMin1       15263.09959  7587.4961   2.01161218  4.465720e-02
## FunctionalMin2       18176.27161  7650.9839   2.37567768  1.779456e-02
## FunctionalMod        27704.34340 12113.3971   2.28708291  2.249916e-02
## FunctionalTyp        28660.70584  6454.1324   4.44067525  1.047319e-05
## PavedDriveP          -1166.21480  5071.0772  -0.22997378  8.181817e-01
## PavedDriveY           7233.65295  3292.1203   2.19726267  2.834024e-02
```

p-values:

```
##        value
## 2.12313e-290
```

Adjusted R-squared:

```
## [1] 0.8849292
```

AIC:

```
## [1] 16381.04
```

BIC:

```
## [1] 16647.67
```

VIF:

```
##                     GVIF Df GVIF^(1/(2*Df))
## PC1             4.080965  1        2.020140
## PC3             4.346983  1        2.084942
## PC4             1.469448  1        1.212208
## PC5             1.582655  1        1.258036
## PC6             1.684946  1        1.298055
## PC7             1.237646  1        1.112495
## PC8             1.148415  1        1.071641
## PC9             1.158146  1        1.076172
## MSZoning        3.645275  3        1.240571
## LandContour     1.560030  3        1.076933
## LotConfig       1.352988  3        1.051677
## Neighborhood    5.886935  4        1.248062
## Condition1      1.644309  4        1.064138
## BldgType        6.380121  4        1.260676
## HouseStyle      5.536189  4        1.238515
## RoofStyle       1.427511  2        1.093062
## Exterior1st  6632.913003  4        3.004091
## Exterior2nd  6515.077892  4        2.997367
## ExterQual       4.218565  2        1.433148
## ExterCond       1.614037  2        1.127141
## KitchenQual     2.825011  2        1.296448
## Functional      2.105329  5        1.077288
## PavedDrive      1.600230  2        1.124723
```

RMSE:

```
## [1] 15929.13
```

- So, we can say that using PCA followed by stepAIC the OLS regression model is better as compared to the other OLS model built.
- There is also no multicollinearity found in the model as the VIF values are less than 10.

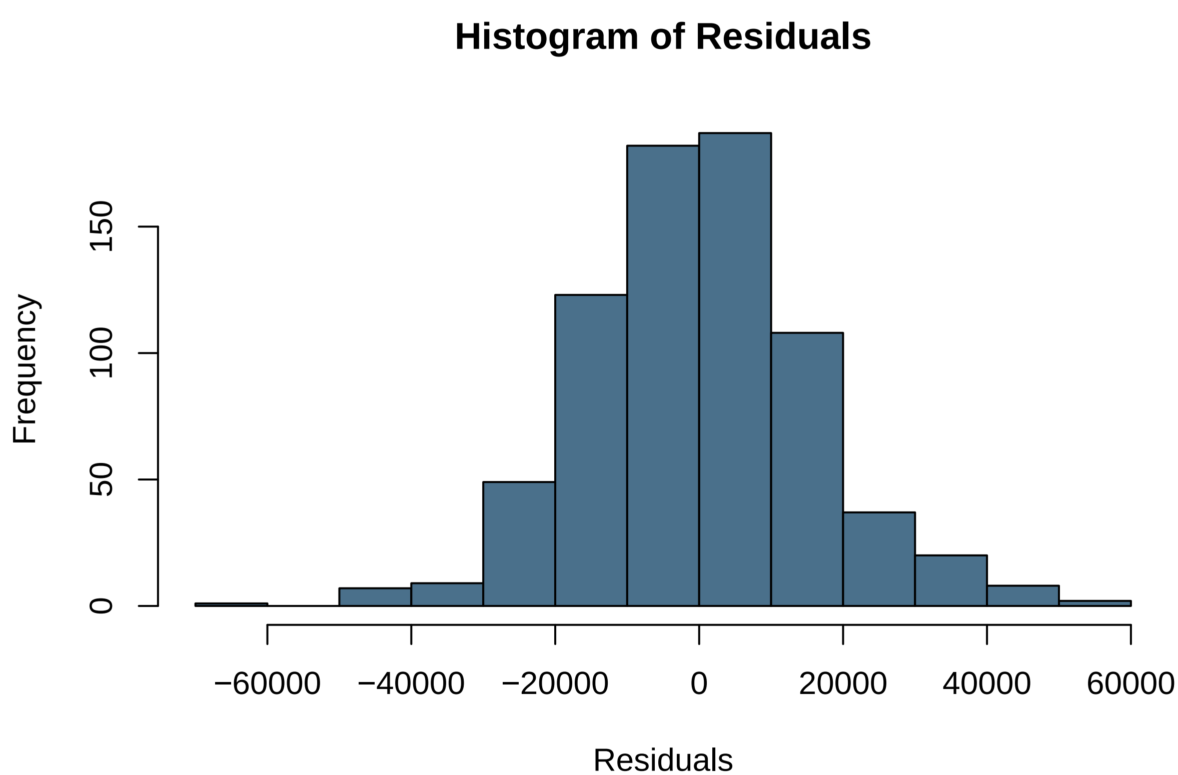
Model	Notes	Hyperparameters	RMSE	Rsquared
OLS	lm + 2-way interactions	N/A	15929.13	0.8849292

ii. Complete analysis of the residuals

A linear regression model is considered fit if the below assumptions are met:

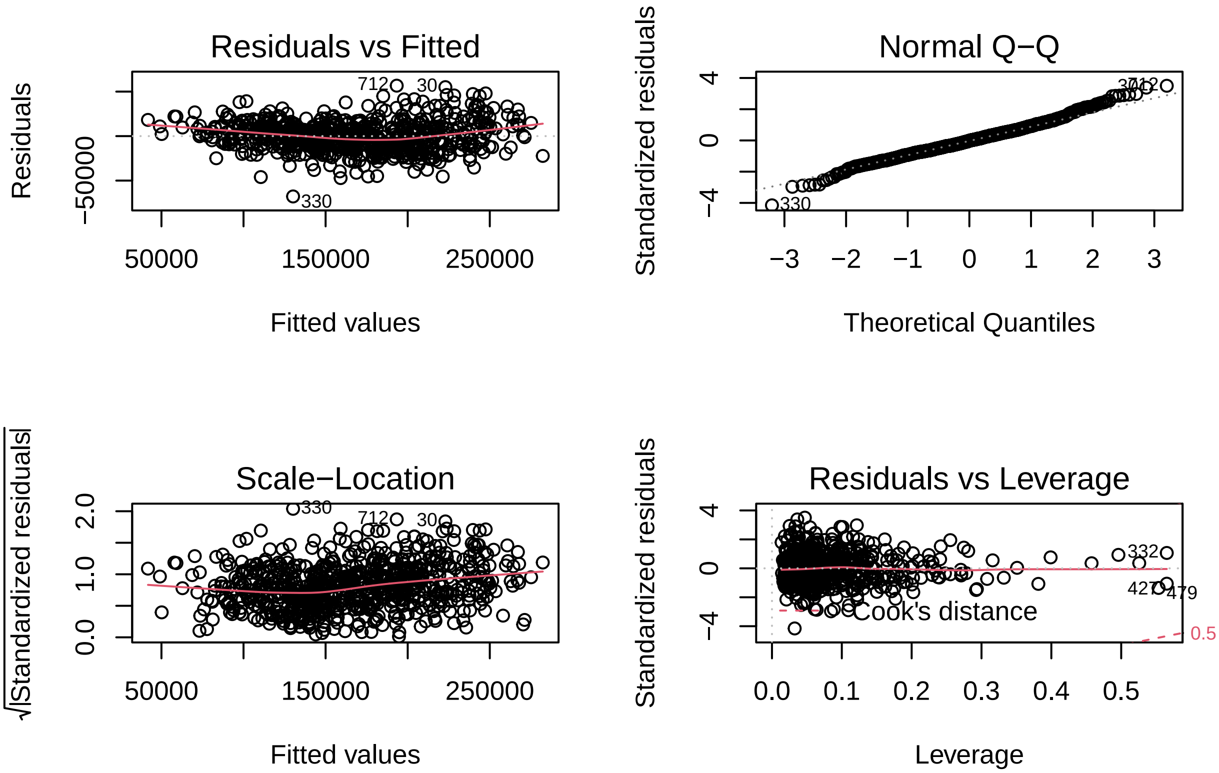
- **Residuals should follow normal distribution**
- **There should be no heteroscedasticity**
- **There should be no multicollinearity**

```
hist(ols.mdl2$residuals,
     col = 'skyblue4',
     main = 'Histogram of Residuals',
     xlab = 'Residuals')
```



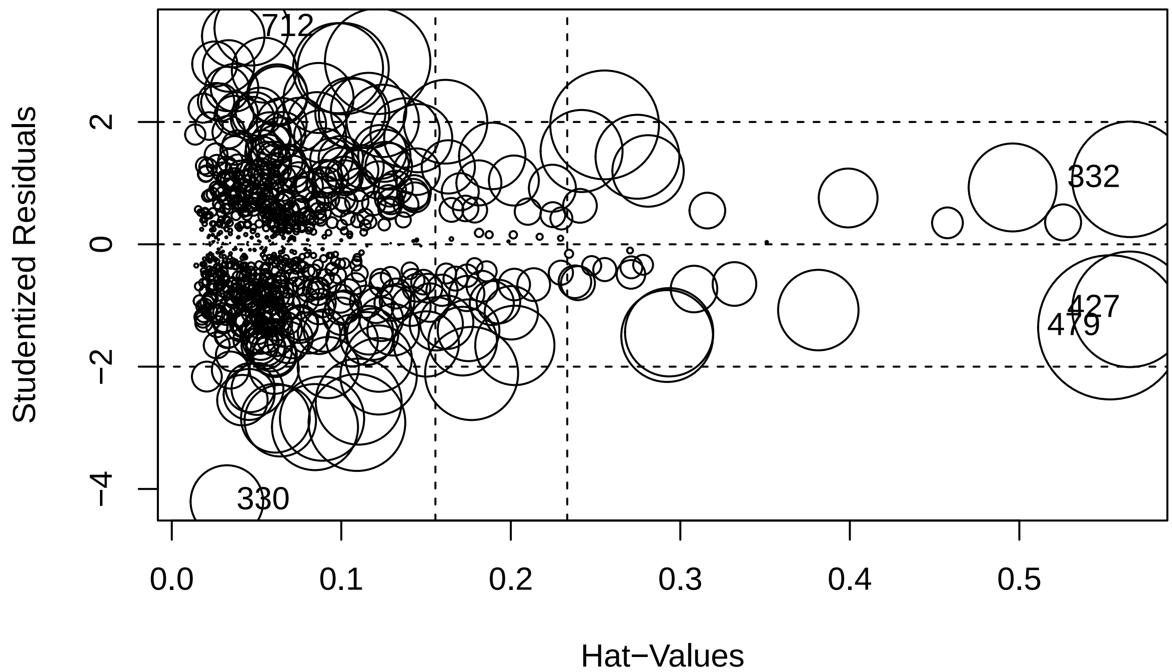
We can see that the residuals are `normally distributed`.

```
par(mfrow=c(2,2)) #combining multiple plots together
plot(ols.mdl2)
```



- From the *Residuals vs Fitted* plot, we can see there are points above and below the 0 line.
- There is also a pattern seen like a `slight curvature pattern` which indicates that there maybe a systematic lack of fit.
- From the *Normal Q-Q* plot, we can see that most of the points are `very close to the dotted line`, indicating that the residuals follow a normal distribution, except some points which might be outliers which maybe affecting the regression line fit of data.
- Here the *Scale-Location* plot suggests that the red line is roughly horizontal across the plot and the spread of magnitude looks unequal, at some fitted values there are more residuals as compared to other like the ones in between 120000 and 210000, indicating some heteroskedasticity.
- From the *Residuals vs Leverage* plot, we can see that there are no influential points in our regression model. We need to check `influencePlot` to see if we are missing any leverage.

```
influencePlot(ols.mdl2)
```




```
##        StudRes        Hat      CookD
## 330 -4.206583 0.03250006 0.01017704
## 332  1.062619 0.56525366 0.02575165
## 427 -1.062619 0.56525366 0.02575165
## 479 -1.357941 0.55353100 0.04005859
## 712  3.530266 0.04707287 0.01062059
```

- We can now see some high influential points for the fitted values.

```
#ncv Test
ncvTest(ols.mdl2)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 45.09124, Df = 1, p = 1.8806e-11
```

Since `p-value` is less than significance level (α) of `0.05`, that means we `reject the null hypothesis` of constant error variance which indicates heteroscedasticity.

```
VIF(ols.mdl2)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## PC1             4.080965  1        2.020140
## PC3             4.346983  1        2.084942
## PC4             1.469448  1        1.212208
## PC5             1.582655  1        1.258036
## PC6             1.684946  1        1.298055
## PC7             1.237646  1        1.112495
## PC8             1.148415  1        1.071641
```

```
## PC9              1.158146  1         1.076172
## MSZoning         3.645275  3         1.240571
## LandContour      1.560030  3         1.076933
## LotConfig        1.352988  3         1.051677
## Neighborhood     5.886935  4         1.248062
## Condition1       1.644309  4         1.064138
## BldgType         6.380121  4         1.260676
## HouseStyle       5.536189  4         1.238515
## RoofStyle        1.427511  2         1.093062
## Exterior1st   6632.913003  4         3.004091
## Exterior2nd   6515.077892  4         2.997367
## ExterQual        4.218565  2         1.433148
## ExterCond        1.614037  2         1.127141
## KitchenQual      2.825011  2         1.296448
## Functional       2.105329  5         1.077288
## PavedDrive       1.600230  2         1.124723
```

Generally, VIF values which are greater than 5 or 7 are the cause of multicollinearity which we do not see in our model.

Improving the current model:
* To improve our model, we need to remove some influential observations from our model and then fit the regression model to the data.
* We can re-build the model with new predictors.
* We can also perform variable transformation such as Box-Cox or use better evolved models like SVR, PCR etc., and see how it works.

1 (b) - PLS Model

Model Setup

- Using the whole data set after PMM imputation and factor level collapsing without omitting any outliers
- Using the predictors - `GarageArea`, `GarageCars`, `TotRmsAbvGrd`, `FullBath`, `GrLivArea`, `X1stFlrSF`, `TotalBsmtSF`, `OverallQual` which has strong correlations with response variable - `SalePrice`

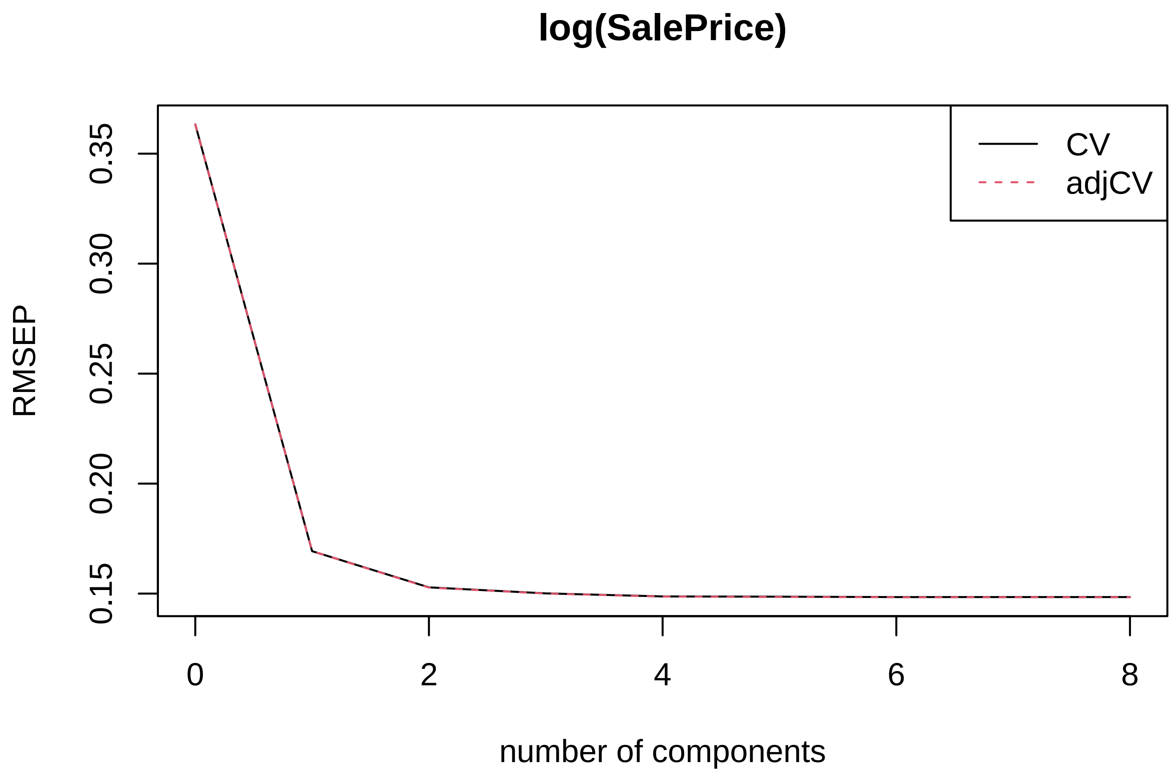
```
#creating a PLS model to predict the log of the sale price
#using 5-fold CV

pls.model <- plsr(log(SalePrice) ~ GarageArea + GarageCars + TotRmsAbvGrd
              + FullBath + GrLivArea + X1stFlrSF + TotalBsmtSF + OverallQual,
              data=hd.Cleaned, scale=TRUE, validation='CV', k=5)
```

- Hyperparameter tuning to determine the number of PLS components with RMSE as the error metric

```
#report chart
summary(pls.model)
```

```
## Data:    X dimension: 1000 8
##  Y dimension: 1000 1
## Fit method: kernelpls
## Number of components considered: 8
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.3633   0.1693   0.1528   0.1501   0.1487   0.1486   0.1484
## adjCV       0.3633   0.1693   0.1527   0.1500   0.1487   0.1485   0.1483
##        7 comps  8 comps
## CV      0.1484   0.1484
## adjCV   0.1484   0.1484
##
## TRAINING: % variance explained
##                 1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X                 54.34    62.66    74.93    79.61    83.32    95.82    97.73
## log(SalePrice)    78.36    82.58    83.18    83.49    83.60    83.60    83.60
##                 8 comps
## X                 100.0
## log(SalePrice)     83.6
```

```
plot(RMSEP(pls.model),legendpos="topright")
```

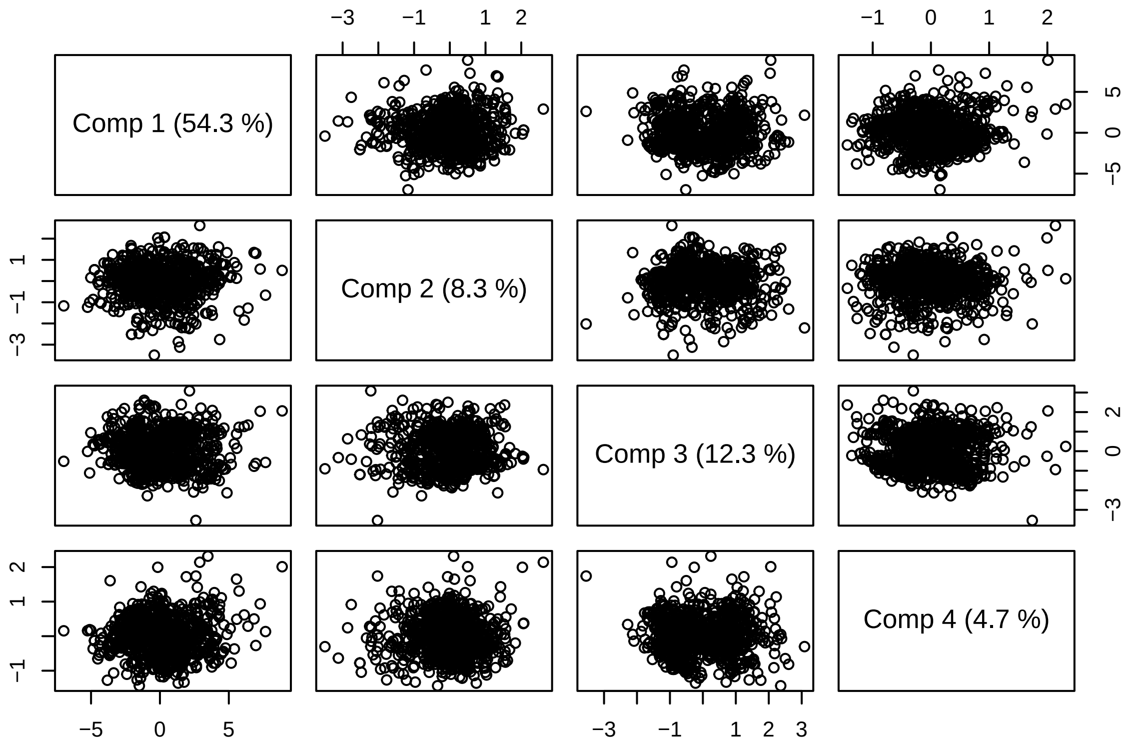


- From the table, we can see that if we use `6` PLS components only in our model, the RMSE drops to `0.1486` and after that even if we keep adding components the RMSE still is the same.

- Though we are eyeballing the CV component, but from the plot we can see that fitting `4` PLS components is enough because even if we are adding 2 more components there is not much difference in the CV component.

- Using the final model with `four PLS components` to make predictions

```
final.pls <- plsr(log(SalePrice) ~ GarageArea + GarageCars + TotRmsAbvGrd
              + FullBath + GrLivArea + X1stFlrSF + TotalBsmtSF + OverallQual,4,
              data=hd.Cleaned, scale=TRUE, validation='CV', k=5)

plot(final.pls, plottype = "scores", comps = 1:4)
```



- From the above plot, we can see that by using only four PLS components we can describe about 80% of the variation in the response variable.
- Metric Calculations:

Model	Notes	Hyperparameters	RMSE	Rsquared
PLS	pls	ncomp = 4	0.1474771	0.0218368

- If we now compare between our preferred OLS model and PLS model on basis of RMSE values, we can see that PLS model's efficiency is much higher.
- RMSE for chosen OLS model was `ols.mdl2.rsme` whereas for PLS model is `0.1475`.

1 (c) - LASSO Model

Model Setup

- We first setup our cross-validation strategy

- Then create a dataframe with PMM imputed values, and only whole columns without NA. Does not omit outliers

- Then we train the model using `glmnet` which actually fits the elastic net

```
ctrl <- trainControl(method  = "repeatedcv",
                     number  = 5, # 5 fold cross validation
                     repeats = 2  # 2 repeats
                     )

# The data (PMM imputed values, and only whole columns without NA. Does not omit outliers)
df.lasso <- cbind(SalePrice = hd.numericClean$SalePrice,
                hd.numericClean, hd.factorClean)
```

Fit the Model

```
# Train and tune the SVM
fit.lasso <- train(data = df.lasso,
                 log(SalePrice) ~ .,
                 method     = "glmnet",          # Elastic net
                 preProc    = c("center","scale"), # Center and scale data
                 tuneLength = 10,  #10 values of alpha and 10 lamda values for each
                 trControl  = ctrl)
```

- The variables with non-zero coefficients of the final model:

```
lasso.coeff <- drop(coef(fit.lasso$finalModel, fit.lasso$bestTune$lambda))

lasso.coeff[lasso.coeff != 0]
```

```
##         (Intercept)          MSSubClass         LotFrontage             LotArea
##        1.200247e+01       -7.152110e-05        3.440567e-03        2.420300e-02
##         OverallQual         OverallCond          MasVnrArea          BsmtFinSF1
##        7.994212e-02        4.961582e-02        2.014865e-03        3.054406e-02
##          BsmtFinSF2         TotalBsmtSF        LowQualFinSF           GrLivArea
##        3.951459e-03        4.266567e-02       -1.226020e-03        1.330731e-01
##        BsmtFullBath            FullBath            HalfBath        BedroomAbvGr
##        1.013981e-02        6.445650e-05        4.026834e-04       -6.668434e-03
##        KitchenAbvGr        TotRmsAbvGrd          Fireplaces          GarageCars
##       -7.830963e-03        7.270762e-04        2.321698e-02        2.346675e-02
##          GarageArea          WoodDeckSF         OpenPorchSF         EncPorchSF
##        1.983305e-02        6.241643e-03        8.166466e-03        1.317097e-02
##            PoolArea             MiscVal                 age     ageSinceRemodel
##        1.941268e-03       -1.111083e-04       -5.187661e-02       -1.096697e-02
##          MSZoningRH          MSZoningRM         LotShapeIR3         LotShapeReg
##       -3.307184e-03       -2.456559e-02       -2.167316e-03       -1.256580e-03
##      LandContourHLS     LotConfigCulDSac        LandSlopeMod         LandSlopeSev
##        5.603433e-03        5.360943e-03        3.218830e-03       -1.209156e-03
## NeighborhoodOldTown   Neighborhoodother   NeighborhoodOther      Condition1Norm
##       -6.718497e-03       -3.094314e-03        2.095832e-03        1.557727e-02
```

```
##       BldgTypeDuplex        BldgTypeTwnhs       BldgTypeTwnhsE       HouseStyleSLvl
##        -2.030359e-03        -1.061458e-02        -3.343478e-03         9.630373e-04
##      HouseStyleOther         RoofStyleHip       RoofStyleother  Exterior1stMetalSd
##        -4.150667e-03         7.859146e-04         9.913929e-03         3.892198e-03
##   Exterior1stWd Sdng     Exterior1stOther  Exterior2ndVinylSd  Exterior2ndWd Sdng
##        -2.882589e-03         5.848763e-03         6.424823e-03         4.191821e-03
##     Exterior2ndOther         ExterQualAvg   ExterQualBelowAvg         ExterCondAvg
##        -4.336144e-04        -4.844774e-03        -6.207730e-03         3.657840e-03
##      Foundationother      FoundationPConc        Heatingother        HeatingQCAvg
##        -3.155674e-04         1.777523e-02         3.117311e-03        -6.889703e-03
##    HeatingQCBelowAvg          CentralAirY      KitchenQualAvg KitchenQualBelowAvg
##        -2.721806e-03         1.042547e-02        -7.274004e-03        -3.839291e-03
##       FunctionalMaj2       FunctionalMin2        FunctionalMod        FunctionalTyp
##        -1.235929e-02         2.701013e-03        -1.480586e-04         1.487099e-02
##          PavedDriveP          PavedDriveY
##        -1.641397e-05         6.425672e-03
```

Model	Notes	Hyperparameters	RMSE	Rsquared
Lasso	caret and elasticnet	Alpha = 1 , Lambda = 0.00167070437878296	0.1010816	0.922609

1 (d) - Model Variants

1 (d, i) - PCR Model

Model Setup

- Uses `numeric` data for Principal Component Analysis
- 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`

Perform PCA

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%

```
## [1] "There are 12 principal components that explain up to 75% of the variation in the data"
```

Join on the factor data

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

Fit the Model

- 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")
```

Model	Notes	Hyperparameters	RMSE	Rsquared
PCR	lm	N/A	0.1014132	0.9178239

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

```
##
## Call:
## lm(formula = log(SalePrice) ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 +
##     PC7 + PC9 + PC12 + MSZoning + LandContour + LotConfig + Condition1 +
##     HouseStyle + RoofStyle + Exterior1st + ExterQual + ExterCond +
##     Foundation + Heating + CentralAir + KitchenQual + Functional +
##     PavedDrive, data = df.pcr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.69673 -0.05875  0.00159  0.06701  0.31402
```

```
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)    
## (Intercept)         11.7686397  0.0550213 213.892  < 2e-16 ***
## PC1                  0.0986702  0.0023721  41.597  < 2e-16 ***
## PC2                 -0.0054867  0.0033611  -1.632 0.102923    
## PC3                 -0.0511019  0.0033405 -15.298  < 2e-16 ***
## PC4                 -0.0235630  0.0033654  -7.002 4.78e-12 ***
## PC5                 -0.0379678  0.0033373 -11.377  < 2e-16 ***
## PC6                 -0.0087772  0.0031369  -2.798 0.005245 ** 
## PC7                  0.0349521  0.0035230   9.921  < 2e-16 ***
## PC9                  0.0081982  0.0034207   2.397 0.016739 *  
## PC12                 0.0181949  0.0036259   5.018 6.23e-07 ***
## MSZoningRH          -0.0659683  0.0398357  -1.656 0.098051 .  
## MSZoningRL          -0.0321814  0.0201756  -1.595 0.111031    
## MSZoningRM          -0.1125247  0.0218390  -5.152 3.13e-07 ***
## LandContourHLS       0.0744653  0.0266931   2.790 0.005382 ** 
## LandContourLow       0.0010358  0.0284758   0.036 0.970992    
## LandContourLvl      -0.0156112  0.0181350  -0.861 0.389548    
## LotConfigCulDSac     0.0439742  0.0151677   2.899 0.003827 ** 
## LotConfigInside      0.0032318  0.0091396   0.354 0.723713    
## LotConfigother      -0.0071985  0.0192764  -0.373 0.708908    
## Condition1Feedr      0.0486306  0.0246766   1.971 0.049047 *  
## Condition1Norm       0.0926746  0.0203588   4.552 6.00e-06 ***
## Condition1RR         0.0520491  0.0291863   1.783 0.074851 .  
## Condition1Other      0.0244003  0.0311172   0.784 0.433153    
## HouseStyle1Story    -0.0668074  0.0157227  -4.249 2.36e-05 ***
## HouseStyle2Story    -0.0146966  0.0153565  -0.957 0.338797    
## HouseStyleSLvl      -0.0270715  0.0203274  -1.332 0.183255    
## HouseStyleOther     -0.0509964  0.0199468  -2.557 0.010724 *  
## RoofStyleHip         0.0152849  0.0093476   1.635 0.102345    
## RoofStyleother       0.0970582  0.0246717   3.934 8.96e-05 ***
## Exterior1stMetalSd   0.0290609  0.0125908   2.308 0.021207 *  
## Exterior1stVinylSd   0.0256903  0.0113709   2.259 0.024091 *  
## Exterior1stWd Sdng  -0.0034236  0.0133337  -0.257 0.797416    
## Exterior1stOther     0.0314277  0.0115424   2.723 0.006592 ** 
## ExterQualAvg        -0.0383087  0.0116412  -3.291 0.001036 ** 
## ExterQualBelowAvg   -0.1032930  0.0466318  -2.215 0.026991 *  
## ExterCondAvg         0.0222015  0.0116446   1.907 0.056874 .  
## ExterCondBelowAvg    0.0060538  0.0322546   0.188 0.851162    
## FoundationCBlock     0.0066855  0.0143041   0.467 0.640335    
## Foundationother      0.0261987  0.0253505   1.033 0.301651    
## FoundationPConc      0.0535857  0.0166839   3.212 0.001363 ** 
## Heatingother         0.0356275  0.0252882   1.409 0.159204    
## CentralAirY          0.0648420  0.0184339   3.518 0.000456 ***
## KitchenQualAvg      -0.0212534  0.0104036  -2.043 0.041339 *  
## KitchenQualBelowAvg -0.0412470  0.0257426  -1.602 0.109426    
## FunctionalMaj2      -0.2007091  0.0617032  -3.253 0.001183 ** 
## FunctionalMin1       0.0411613  0.0387696   1.062 0.288646    
## FunctionalMin2       0.0371397  0.0377803   0.983 0.325835    
## FunctionalMod        0.0145972  0.0446559   0.327 0.743829    
## FunctionalTyp        0.1083890  0.0317133   3.418 0.000658 ***
## PavedDriveP         -0.0009913  0.0252404  -0.039 0.968680    
## PavedDriveY          0.0503331  0.0161331   3.120 0.001864 ** 
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1041 on 949 degrees of freedom
## Multiple R-squared:  0.9219, Adjusted R-squared:  0.9178
## F-statistic: 224.2 on 50 and 949 DF,  p-value: < 2.2e-16
```

View and Interpret Results

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.

View Predicted vs. Actuals

Function to compare predicted vs. observed values

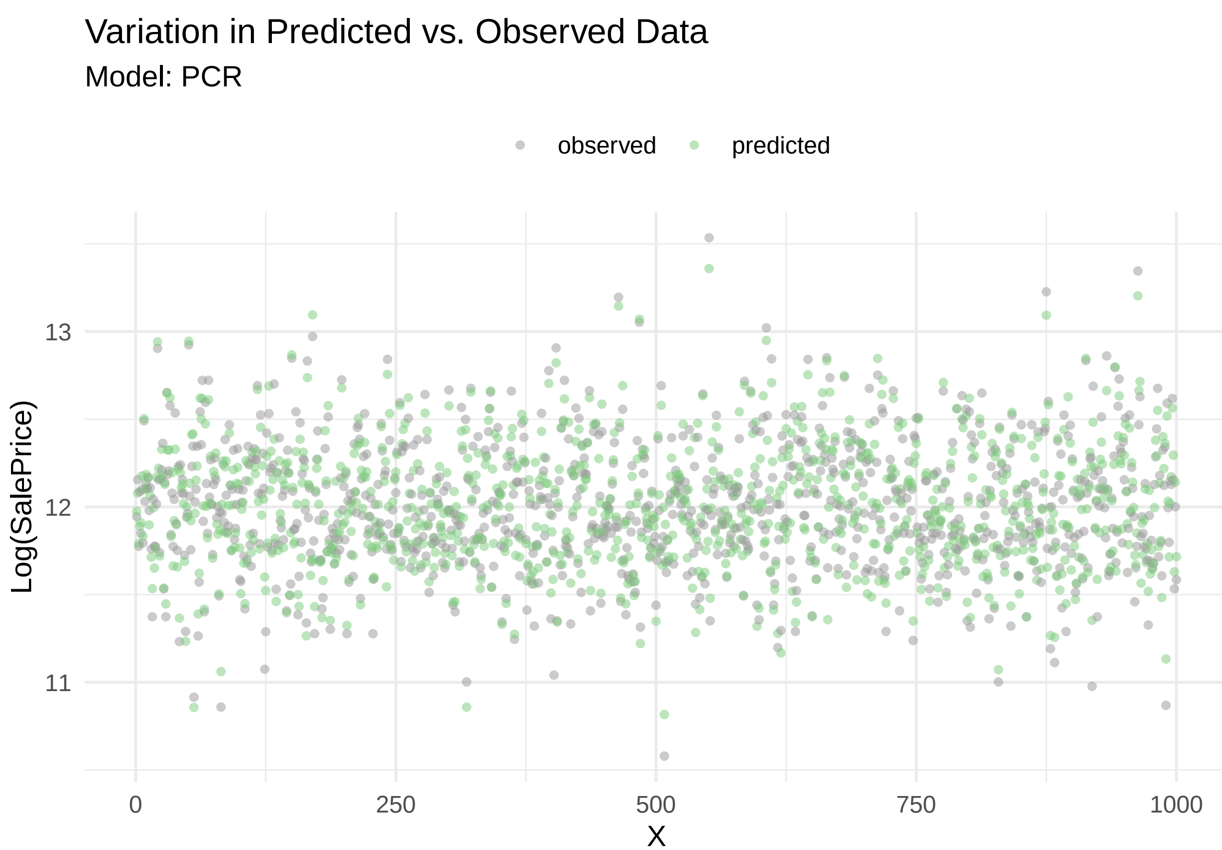
```r
# Function to compare predicted vs. actual (observed) regression outputs
predictedVsObserved <- function(predicted, observed, modelName, outcomeName = 'Log(SalePrice)') {

  ## Create data set 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 = paste(modelName, '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
  )
```

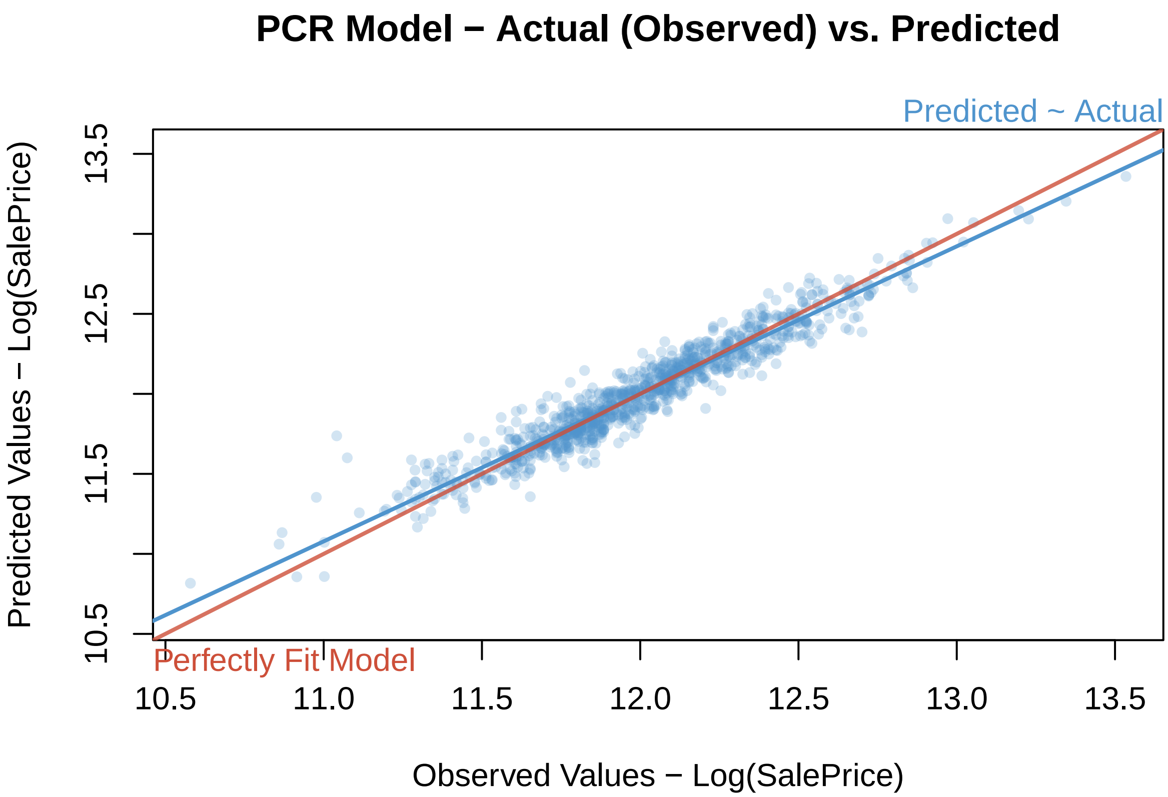
```
  # Add the Predicted vs. actual line
  abline(lm(predicted ~ observed), col = 'steelblue3', lwd = 2)
  mtext('Predicted ~ Actual', side = 3, adj = 1, col = 'steelblue3')

  # 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 predicted vs. Actuals 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

Model Setup

```
ctrl <- trainControl(method  = "repeatedcv",
                     number  = 5, # 5 fold cross validation
                     repeats = 2  # 2 repeats
                     )

# The data (PMM imputed values, and only whole columns without NA. Does not omit outliers)
df.svm <- cbind(SalePrice = hd.numericClean$SalePrice,
                hd.numericClean, hd.factorClean)
```

Fit the Model

```
# Train and tune the SVM
fit.svm <- train(data = df.svm,
                 log(SalePrice) ~ .,
                 method     = "svmRadial",          # Radial kernel
                 tuneLength = 9,                    # 9 values of the cost function
                 preProc    = c("center","scale"), # Center and scale data
                 trControl  = ctrl)
```

View and Interpret Results

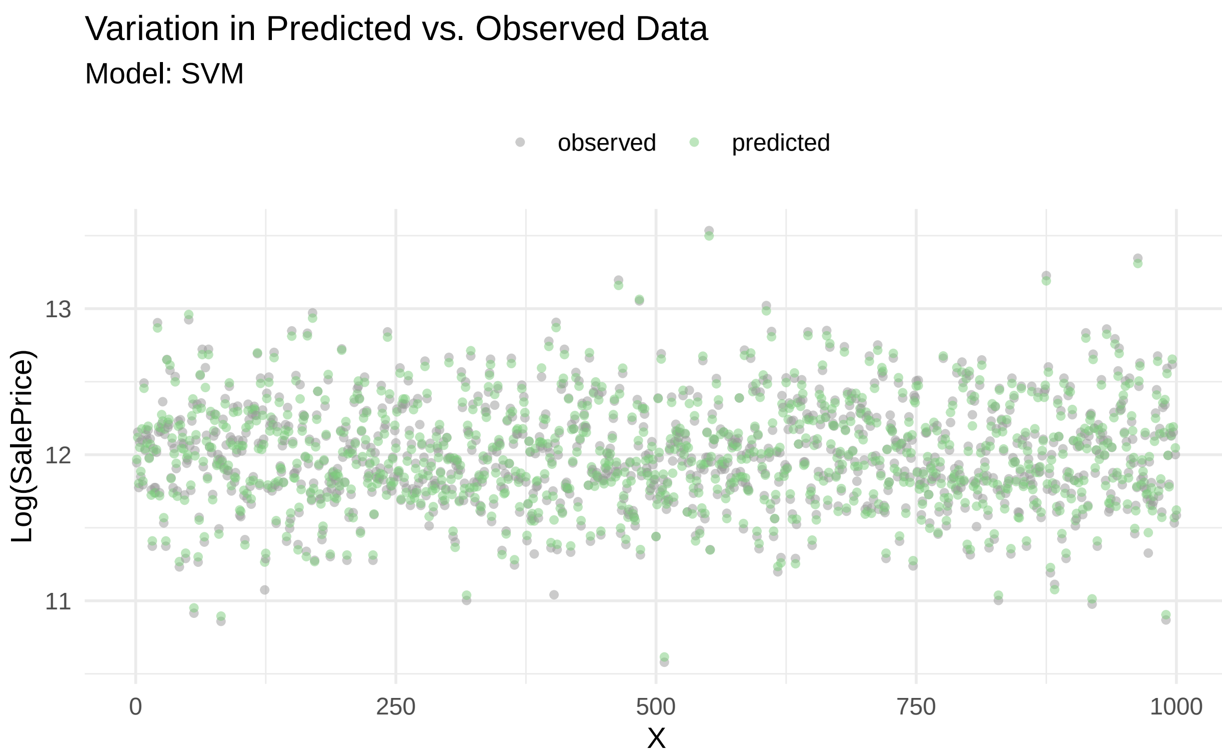
- Note all numbers mentioned below are approximate
- See that the R Squared of the model is around 0.86, and RMSE is 0.14
- See that the model predicts the data well.
- Also, note that the model predicts the data with less error than the linear model. See this from the RMSE or scatter plot of predicted values.

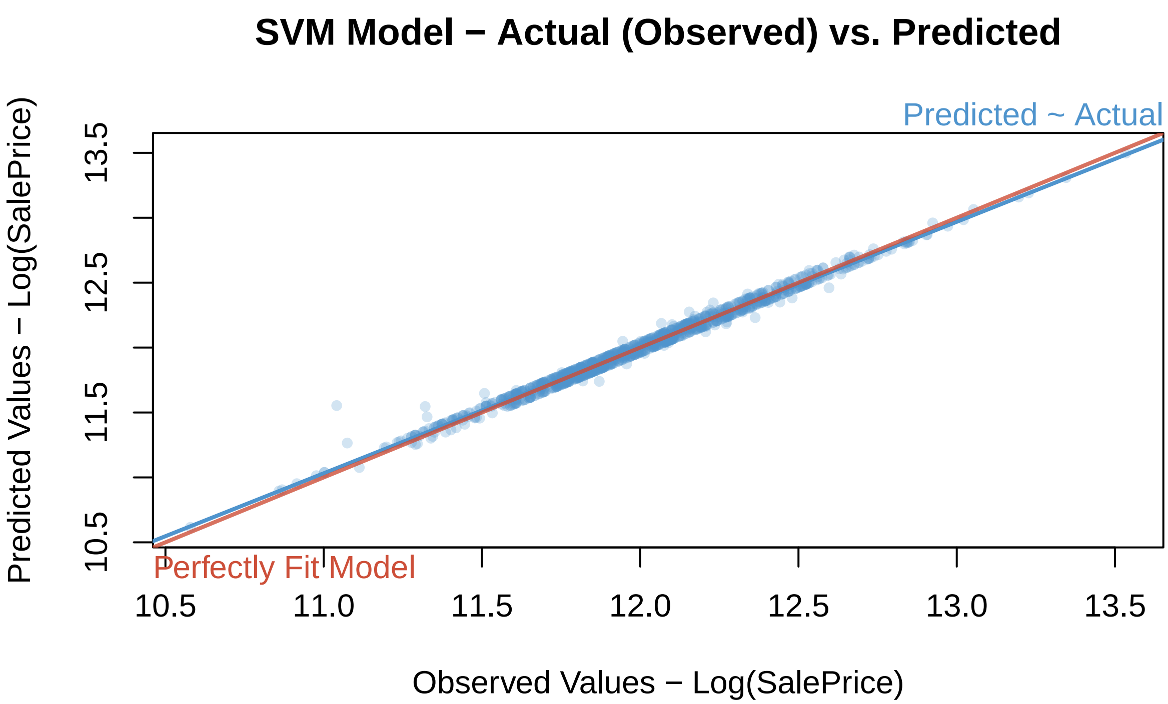
Model	Notes	Hyperparameters	RMSE	Rsquared
SVM	caret and svmRadial	C = 4 , Epsilon = 0.1	0.1418915	0.8517242

```
# Final model?
fit.svm$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 4
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.00752928895571278
##
## Number of Support Vectors : 671
##
## Objective Function Value : -162.4365
## Training error : 0.012145
```

```
# How do the predicted vs. Actuals Compare?
predictedVsObserved(observed  = log(df.svm$SalePrice),
                    predicted = predict(fit.svm, df.svm),
                    modelName = 'SVM')
```


Variation in Predicted vs. Observed Data
Model: SVM
observed
predicted
Log(SalePrice)
13
12
11
0
250
500
750
1000
X


SVM 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, iii) - MARS Model

Fit the Model

```
# Train and tune the MARS model
fit.mars <- train(data = df.svm, # note this is fine since data is the same for this model
                  log(SalePrice) ~ .,
                  method     = "earth",             # Radial kernel
                  tuneLength = 9,                   # 9 values of the cost function
                  preProc    = c("center","scale"), # Center and scale data
                  trControl  = ctrl
                 )
```

Model	Notes	Hyperparameters	RMSE	Rsquared
MARS	caret and earth	Degree = 1 , nprune = 17	0.1101934	0.9085685

View and Interpret Results

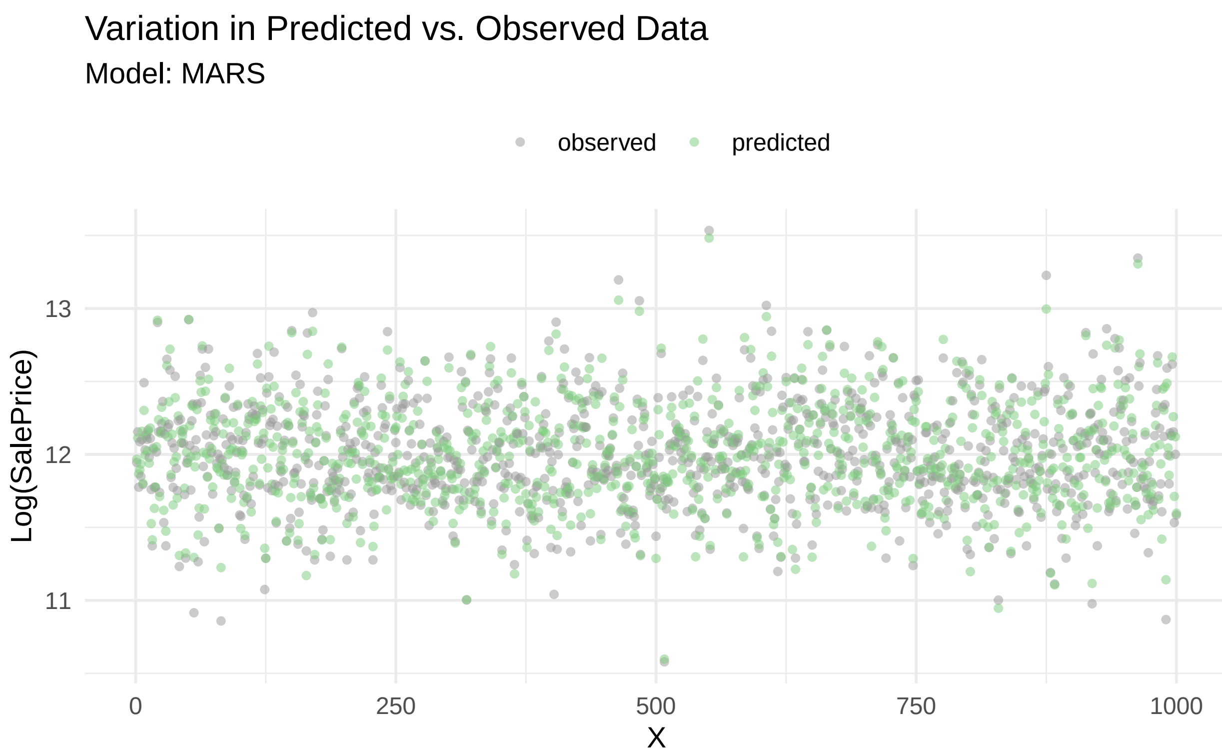
- See that the model overall performs very well, and in fact performs similarly to the PCR model (in terms of RMSE and Adjusted R Squared).

- Again, unsure if the model would do well for prediction of out of sample data, but fits this data extremely well.

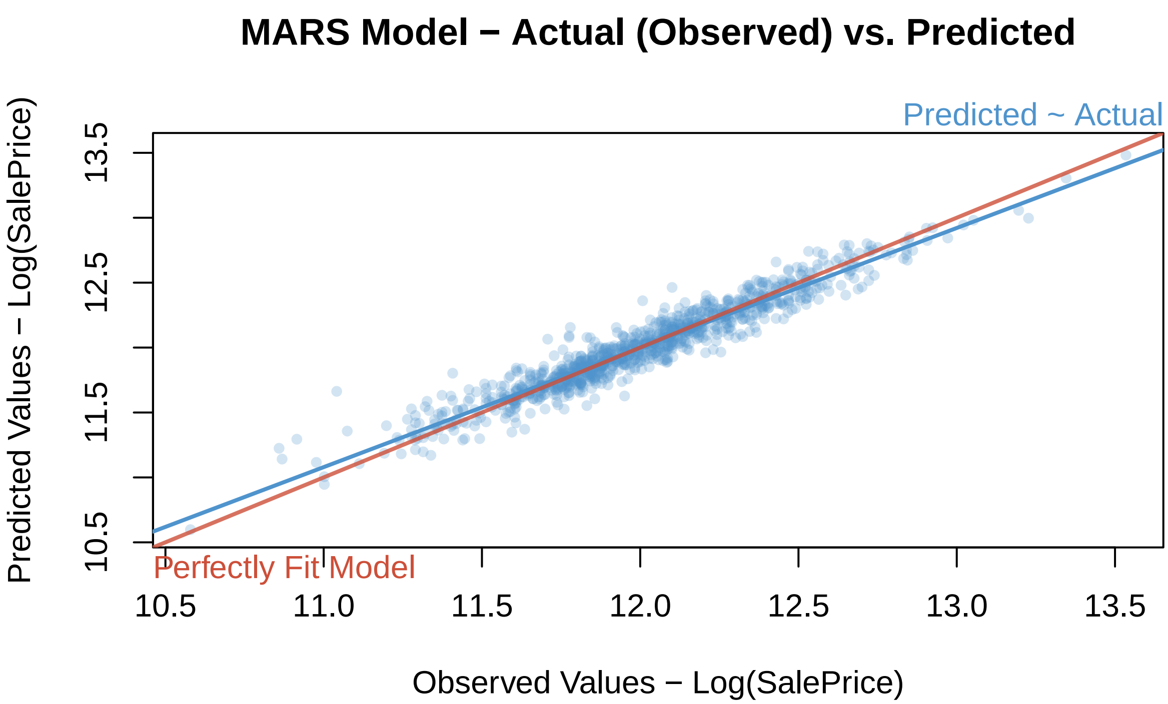
```
# Final model?
fit.mars$finalModel
```

```
## Selected 17 of 21 terms, and 10 of 94 predictors (nprune=17)
## Termination condition: RSq changed by less than 0.001 at 21 terms
## Importance: GrLivArea, age, OverallQual, TotalBsmtSF, OverallCond, LotArea, ...
## Number of terms at each degree of interaction: 1 16 (additive model)
## GCV 0.011145    RSS 10.42157    GRSq 0.9155756    RSq 0.9208976
```

```
# How do the predicted vs. Actuals Compare?
predicted.mars = fit.mars[["finalModel"]][["fitted.values"]]
colnames(predicted.mars) <- 'predicted'

predictedVsObserved(observed  = log(df.svm$SalePrice),
                    predicted = predicted.mars,
                    modelName = 'MARS')
```



Variation in Predicted vs. Observed Data
Model: MARS
observed
predicted
Log(SalePrice)
13
12
11
0
250
500
750
1000
X



MARS 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)

Summary Table of Model Performance

Model	Notes	Hyperparameters	RMSE	Rsquared
OLS	lm	N/A	20948.4222	0.8142
OLS	lm + 2-way interactions	N/A	15929.1310	0.8849
PLS	pls	ncomp = 4	0.1475	0.0218
Lasso	caret and elasticnet	Alpha = 1 , Lambda = 0.00167070437878296	0.1011	0.9226
PCR	lm	N/A	0.1014	0.9178
SVM	caret and svmRadial	C = 4 , Epsilon = 0.1	0.1419	0.8517
MARS	caret and earth	Degree = 1 , nprune = 17	0.1102	0.9086

References

1. https://rpubs.com/staneaurelius/house_price_prediction
2. https://www.statology.org/partial-least-squares-in-r/
3. https://daviddalpiaz.github.io/r4sl/elastic-net.html