Reed Ballesteros
MSDS-410-DL, Summer 2022
Dr. Mickelson
7/24/2022

Modeling Assignment 6: Finalizing the Model – Variable Selection Procedures and Validation

Preparatory Work

Similar to what was previously done in Modeling Assignment 4, we want to perform both data cleanup and waterfall dropdown.

Cleanup:

- Correct GarageCars with <NA> values to 0
- Correct MasVnrArea with <NA> values to 0
- Correct TotalBsmtSF with <NA> values to 0
- Correct TotRmsAbvGrd with <NA> values to 0
- Correct FullBath with <NA> values to 0
- Correct BsmtFullBathwith <NA> values to 0
- Correct BsmtHalfBathwith <NA> values to 0
- Correct BsmtFinSF1with <NA> values to 0
- Correct BsmtFinSF2with <NA> values to 0
- Correct BsmtUnfSFwith <NA> values to 0
- Delete GarageYrBlt: contains <NA> values and will not interpret them in this study; numerical model selection will interpret this value as a numerical value and <NA> values will result in errors for automatic model selection
- Delete SubClass column as its column is numeric but is interpreted as a categorical code for various types of homes; we will not define a particular SubClass as the control group for this study.
- LotFrontage: <NA> values are based on the average LotFrontage of their given Neighborhood
- Delete the SID and PID columns as our numerical model selection will interpret these variables as quantitative values instead of simply identifiers.

Waterfall dropdown:

- Narrow population to only single-family homes (BldgType = ‘1Fam’)

New Data/Transformations:

- QualityIndex: OverallQual* OverallCond
- TotalSqftCalc: BsmtFinSF1 + BsmtFinSF2 + GrLivArea
- PriceSqft: SalePrice/TotalSqftCalc
- TotalFullBath: FullBath + BsmtFullBath
- TotalHalfBath: HalfBath + BsmtHalfBath

Dummy Variables:

- CentralAir replaced by CentralAirY and CentrailAirN, with CentralAirY as the control variable
- CentralAir replaced by CentralAirY and CentrailAirN, with CentralAirY as the control variable
- FullBath and BsmtFullBath replaced by TotalFullBath1, TotalFullBath1, TotalFullBath1, TotalFullBath1, with TotalFullBath1 as the control variable
- HalfBath and Bsmt HalfBath replaced by TotalHalfBath1, TotalHalfBath2, TotalHalfBath3 with TotalFullBath1 as the control variable

(1) The Predictive Modeling Framework

Our 70/30 training/test split is the most basic form of cross-validation. We will 'train' each model by estimating the models on the 70% of the data identified as the training data set, and we will 'test' each model by examining the predictive accuracy on the 30% of the data. In R will estimate our models using the lm() function, and we will be able to apply those linear models using the R function predict(). You will want to read the R help page for the R function predict(). In particular, pay attention to the newdata argument. Your test data set is your new data.

Show a table of observation counts for your train/test data partition in your data section.

We have the counts for our 70/30 split for our training and test data:

	Count
mydata	2425
train.df	1707
test.df	718
train.df + test.df	2425

With the preparatory work and train/test split performed, our ‘clean’ training and test data set for the variable auto-selection process is filtered to only numerical columns, shown here:

```
num_cols_train <- unlist(lapply(train.df, is.numeric))
num_cols_train
train.clean <- train.df[ , num_cols_train]
names(train.clean)

num_cols_test <- unlist(lapply(test.df, is.numeric))
num_cols_test
test.clean <- test.df[ , num_cols_test]
```

```
> names(train.clean)
 [1] "LotFrontage"     "LotArea"         "OverallQual"     "OverallCond"     "YearBuilt"       "YearRemodel"     "MasVnrArea"
 [8] "BsmtFinSF1"      "BsmtFinSF2"      "BsmtUnfSF"       "TotalBsmtSF"     "FirstFlrSF"      "SecondFlrSF"     "LowQualFinSF"
[15] "GrLivArea"       "BedroomAbvGr"    "KitchenAbvGr"    "TotRmsAbvGrd"    "Fireplaces"      "GarageCars"      "GarageArea"
[22] "WoodDeckSF"      "OpenPorchSF"     "EnclosedPorch"   "ThreeSsnPorch"   "ScreenPorch"     "PoolArea"        "MiscVal"
[29] "MoSold"          "YrSold"          "SalePrice"       "TotalSqftCalc"   "PriceSqft"       "CentralAirN"     "TotalFullBath2"
[36] "TotalFullBath3"  "TotalFullBath4"  "TotalHalfBath1"  "TotalHalfBath2"  "TotalHalfBath3"
```

(2) Model Identification by Automated Variable Selection

Compute the VIF values for the variable selection models. If the models selected highly correlated pairs of predictors that you do not like, then go back, add them to your drop list, and re-perform the variable selection before you go on with the assignment. The VIF values do not need to be ideal, but if you have a very large VIF value (like 20, 30, 50 etc.), then you should consider removing a variable so that your variable selection models are not junk too.

```
> sort(vif(forward.lm),decreasing=TRUE)
  QualityIndex    OverallQual    OverallCond  TotalSqftCalc    TotalBsmtSF      BsmtUnfSF      PriceSqft   TotRmsAbvGrd      YearBuilt
     36.310987      28.841144      17.448386      12.075102       6.180767       5.232822       3.626124       3.477752       2.625959
    GarageCars TotalFullBath3     MasVnrArea    LotFrontage    OpenPorchSF     WoodDeckSF       PoolArea TotalHalfBath2        MiscVal
      2.270348       1.619202       1.582756       1.396181       1.231822       1.202020       1.109638       1.069030       1.052113
   ScreenPorch
      1.047910
> sort(vif(backward.lm),decreasing=TRUE)
    FirstFlrSF     BsmtFinSF1   QualityIndex      BsmtUnfSF    SecondFlrSF    OverallCond   TotRmsAbvGrd      PriceSqft      YearBuilt
      5.096830       4.879808       4.802488       3.897953       3.807888       3.782602       3.490826       3.477243       2.525822
    GarageCars TotalFullBath3     MasVnrArea     BsmtFinSF2    LotFrontage    OpenPorchSF     WoodDeckSF       PoolArea   LowQualFinSF
      2.272131       1.652085       1.589498       1.554553       1.450566       1.233143       1.215880       1.115438       1.094659
TotalHalfBath2        MiscVal    ScreenPorch
      1.070409       1.057400       1.044817
> sort(vif(stepwise.lm),decreasing=TRUE)
 TotalSqftCalc    TotalBsmtSF      BsmtUnfSF   QualityIndex    OverallCond   TotRmsAbvGrd      PriceSqft      YearBuilt     GarageCars
     12.053737       6.106835       5.206510       4.768849       3.769310       3.452965       3.452604       2.454701       2.266291
TotalFullBath3     MasVnrArea    LotFrontage    OpenPorchSF     WoodDeckSF       PoolArea TotalHalfBath2        MiscVal    ScreenPorch
      1.618430       1.566920       1.383286       1.231410       1.199912       1.109418       1.066646       1.051203       1.043291
> sort(vif(junk.lm),decreasing=TRUE)
 QualityIndex   OverallQual   OverallCond     GrLivArea TotalSqftCalc
    33.186105     21.564057     16.433013      3.474054      3.029836
```

Should we be concerned with VIF values for indicator variables? Why or why not?

We should be concerned with very high VIF values in variables because it indicates very has rates of collinearity within a model. In this case, QualityIndex has a very high VIF value of over 36 in the forward.lm model. Because of that, we will remove QualityIntex from our training and test dataset and re-run the respective variable auto-selection process.

Did the different variable selection procedures select the same model or different models?

With identifying QualityIndex with a very high VIF value, we deleted the column from our training and test data. Our automatic variable selection process gave us the following forward, backward and stepwise models:

forward.lm with QualityIndex in the data:

```
> summary(forward.lm)

Call:
lm(formula = SalePrice ~ OverallQual + TotalSqftCalc + PriceSqft +
    MiscVal + TotalFullBath3 + TotalBsmtSF + MasVnrArea + PoolArea +
    LotFrontage + OpenPorchSF + GarageCars + ScreenPorch + OverallCond +
    QualityIndex + WoodDeckSF + BsmtUnfSF + TotRmsAbvGrd + YearBuilt +
    TotalHalfBath2, data = train.clean)

Residuals:
    Min      1Q  Median      3Q     Max
-516423  -10042     644   10043  207596

Coefficients:
                  Estimate  Std. Error t value            Pr(>|t|)
(Intercept)     29030.3635  62814.9139   0.462            0.644028
OverallQual     -2605.8187   2338.3078  -1.114            0.265265
TotalSqftCalc      75.8435      2.6613  28.499 < 0.0000000000000002 ***
PriceSqft        1907.3510     46.4309  41.079 < 0.0000000000000002 ***
MiscVal            -9.1456      0.9323  -9.809 < 0.0000000000000002 ***
TotalFullBath3  10161.9019   1928.9027   5.268        0.0000001555 ***
TotalBsmtSF        -4.5468      3.5045  -1.297            0.194669
MasVnrArea         23.0808      4.2484   5.433        0.0000000636 ***
PoolArea          -71.2758     16.1446  -4.415        0.0000107499 ***
LotFrontage      -179.1024     37.0086  -4.839        0.0000014206 ***
OpenPorchSF       -30.1323      9.8766  -3.051            0.002317 **
GarageCars       4265.5227   1247.2089   3.420            0.000641 ***
ScreenPorch        28.3743     10.1733   2.789            0.005345 **
OverallCond    -10252.4087   2254.5150  -4.548        0.0000058156 ***
QualityIndex     1435.1489    402.2997   3.567            0.000371 ***
WoodDeckSF         13.5582      5.0578   2.681            0.007419 **
BsmtUnfSF         -13.9013      3.3928  -4.097        0.0000437863 ***
TotRmsAbvGrd     2299.9992    775.3033   2.967            0.003054 **
YearBuilt         -76.7580     33.2203  -2.311            0.020977 *
TotalHalfBath2 -10269.6390   4977.5404  -2.063            0.039247 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 25700 on 1687 degrees of freedom
Multiple R-squared:  0.9089,    Adjusted R-squared:  0.9079
F-statistic: 886.2 on 19 and 1687 DF,  p-value: < 0.00000000000000022
```

After removing QualityIndex, the forward variable selection process replaced it with ThreeSsnPorch:

```
> summary(forward.lm)

Call:
lm(formula = SalePrice ~ OverallQual + TotalSqftCalc + PriceSqft +
    MiscVal + TotalFullBath3 + TotalBsmtSF + MasVnrArea + PoolArea +
    LotFrontage + OpenPorchSF + GarageCars + ScreenPorch + OverallCond
    WoodDeckSF + YearBuilt + BsmtUnfSF + TotRmsAbvGrd + TotalHalfBath2
    ThreeSsnPorch, data = train.clean)

Residuals:
    Min      1Q  Median      3Q     Max
-515813   -9799     571    9528  206742

Coefficients:
                 Estimate Std. Error t value              Pr(>|t|)
(Intercept)    35346.8768 63018.1061   0.561              0.574940
OverallQual     5189.1602   850.1891   6.104          0.00000000128 ***
TotalSqftCalc     76.3286     2.6657  28.633 < 0.0000000000000002 ***
PriceSqft       1895.5059    46.4448  40.812 < 0.0000000000000002 ***
MiscVal           -9.0719     0.9351  -9.701 < 0.0000000000000002 ***
TotalFullBath3 10527.5928  1931.7665   5.450          0.00000005790 ***
TotalBsmtSF       -5.9917     3.4891  -1.717              0.086116 .
MasVnrArea        21.9973     4.2497   5.176          0.00000025361 ***
PoolArea         -73.5379    16.1809  -4.545          0.00000589183 ***
LotFrontage     -172.1872    37.0642  -4.646          0.00000365299 ***
OpenPorchSF      -28.6392     9.9065  -2.891              0.003890 **
GarageCars      4212.2080  1251.0839   3.367              0.000777 ***
ScreenPorch       30.8668    10.1954   3.028              0.002503 **
OverallCond    -2590.6885   632.8507  -4.094          0.00004446501 ***
WoodDeckSF        14.3761     5.0714   2.835              0.004641 **
YearBuilt       -100.5492    32.7079  -3.074              0.002145 **
BsmtUnfSF        -12.8143     3.3873  -3.783              0.000160 ***
TotRmsAbvGrd    2186.5974   777.1107   2.814              0.004953 **
TotalHalfBath2 -9669.6751  4989.9644  -1.938              0.052811 .
ThreeSsnPorch     38.9110    27.0411   1.439              0.150348
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 25780 on 1687 degrees of freedom
Multiple R-squared:  0.9084,	Adjusted R-squared:  0.9073
F-statistic: 880.1 on 19 and 1687 DF,  p-value: < 0.00000000000000022
```

The R2 is slightly lower than the original model by 0.05%.

backward.lm with QualityIndex in the data:

```
> summary(backward.lm)

Call:
lm(formula = SalePrice ~ LotFrontage + OverallCond + YearBuilt +
    MasVnrArea + BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF + FirstFlrSF +
    SecondFlrSF + LowQualFinSF + TotRmsAbvGrd + GarageCars +
    WoodDeckSF + OpenPorchSF + ScreenPorch + PoolArea + MiscVal +
    QualityIndex + PriceSqft + TotalFullBath3 + TotalHalfBath2,
    data = train.clean)

Residuals:
    Min      1Q  Median      3Q     Max
-514928   -9869     503    9953  207702

Coefficients:
                 Estimate Std. Error t value             Pr(>|t|)
(Intercept)    27982.2904 63538.5651   0.440             0.659705
LotFrontage     -175.4550    37.7487  -4.648   0.0000036128584499 ***
OverallCond    -8017.4841  1050.4426  -7.632   0.0000000000000383 ***
YearBuilt        -82.4621    32.6034  -2.529             0.011521 *
MasVnrArea        22.8890     4.2604   5.373   0.0000000884615654 ***
BsmtFinSF1        70.5090     2.9266  24.093 < 0.0000000000000002 ***
BsmtFinSF2        71.4380     4.5841  15.584 < 0.0000000000000002 ***
BsmtUnfSF        -18.8892     2.9303  -6.446   0.0000000001493945 ***
FirstFlrSF        75.8113     3.5225  21.522 < 0.0000000000000002 ***
SecondFlrSF       75.5304     2.7613  27.353 < 0.0000000000000002 ***
LowQualFinSF      86.9930    12.6056   6.901   0.0000000000072785 ***
TotRmsAbvGrd    2205.4802   777.2987   2.837             0.004603 **
GarageCars      4183.3992  1248.5652   3.351             0.000824 ***
WoodDeckSF        13.5131     5.0904   2.655             0.008014 **
OpenPorchSF      -29.6886     9.8887  -3.002             0.002719 **
ScreenPorch       28.9843    10.1654   2.851             0.004407 **
PoolArea         -71.2366    16.1980  -4.398   0.0000116156227119 ***
MiscVal           -9.1028     0.9353  -9.732 < 0.0000000000000002 ***
QualityIndex    1018.5752   146.4082   6.957   0.0000000000049545 ***
PriceSqft       1898.4419    45.4993  41.725 < 0.0000000000000002 ***
TotalFullBath3 10269.1704  1949.7437   5.267   0.0000001565625208 ***
TotalHalfBath2 -9863.2325  4984.2077  -1.979             0.047990 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 25720 on 1685 degrees of freedom
Multiple R-squared:  0.9089,    Adjusted R-squared:  0.9078
F-statistic: 800.7 on 21 and 1685 DF,  p-value: < 0.00000000000000022
```

With QualityIndex removed from the data, the backward selection process added OverallQual and ThreeSsnPorch to the model:

```
> summary(backward.lm)

Call:
lm(formula = SalePrice ~ LotFrontage + OverallQual + OverallCond +
    YearBuilt + MasVnrArea + BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF +
    FirstFlrSF + SecondFlrSF + LowQualFinSF + TotRmsAbvGrd +
    GarageCars + WoodDeckSF + OpenPorchSF + ThreeSsnPorch + ScreenPorch
    PoolArea + MiscVal + PriceSqft + TotalFullBath3 + TotalHalfBath2,
    data = train.clean)

Residuals:
    Min      1Q  Median      3Q     Max
-514888   -9777     567    9621  207177

Coefficients:
                 Estimate Std. Error t value             Pr(>|t|)
(Intercept)    27910.8990 63943.7816   0.436             0.662536
LotFrontage     -171.8817    37.9339  -4.531     0.00000628140258 ***
OverallQual     5209.6140   854.8918   6.094     0.00000000136300 ***
OverallCond    -2573.0068   633.4585  -4.062     0.00005092462878 ***
YearBuilt        -96.8010    33.1353  -2.921             0.003531 **
MasVnrArea        22.3354     4.2833   5.215     0.00000020702310 ***
BsmtFinSF1        70.1861     2.9830  23.529 < 0.0000000000000002 ***
BsmtFinSF2        71.3286     4.6219  15.433 < 0.0000000000000002 ***
BsmtUnfSF        -18.9956     2.9475  -6.445     0.00000000015084 ***
FirstFlrSF        76.2425     3.5339  21.575 < 0.0000000000000002 ***
SecondFlrSF       76.1577     2.7705  27.489 < 0.0000000000000002 ***
LowQualFinSF      88.9431    12.6309   7.042     0.00000000000276 ***
TotRmsAbvGrd    2155.2735   781.4198   2.758             0.005876 **
GarageCars      4188.4159  1253.5428   3.341             0.000852 ***
WoodDeckSF        14.0605     5.1074   2.753             0.005970 **
OpenPorchSF      -28.3783     9.9187  -2.861             0.004274 **
ThreeSsnPorch     38.9552    27.1016   1.437             0.150796
ScreenPorch       30.6930    10.2083   3.007             0.002680 **
PoolArea         -73.0975    16.2352  -4.502     0.00000718028410 ***
MiscVal           -9.0522     0.9384  -9.646 < 0.0000000000000002 ***
PriceSqft       1898.0267    46.6402  40.695 < 0.0000000000000002 ***
TotalFullBath3 10602.8508  1952.6679   5.430     0.00000006459737 ***
TotalHalfBath2 -9518.8173  5002.3205  -1.903             0.057227 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 25800 on 1684 degrees of freedom
Multiple R-squared:  0.9084,    Adjusted R-squared:  0.9072
F-statistic: 759.3 on 22 and 1684 DF,  p-value: < 0.00000000000000022
```

stepwise.lm with QualityIndex in the data:

```
> summary(stepwise.lm)

Call:
lm(formula = SalePrice ~ TotalSqftCalc + PriceSqft + MiscVal +
    TotalFullBath3 + TotalBsmtSF + MasVnrArea + PoolArea + LotFrontage
    OpenPorchSF + GarageCars + ScreenPorch + OverallCond + QualityIndex
    WoodDeckSF + BsmtUnfSF + TotRmsAbvGrd + YearBuilt + TotalHalfBath2,
    data = train.clean)

Residuals:
    Min      1Q  Median      3Q     Max
-515893   -9920     616    9919  207219

Coefficients:
                 Estimate Std. Error t value             Pr(>|t|)
(Intercept)     35391.056  62559.541   0.566             0.571661
TotalSqftCalc      75.719      2.659  28.475 < 0.0000000000000002 ***
PriceSqft        1896.032     45.310  41.846 < 0.0000000000000002 ***
MiscVal            -9.115      0.932  -9.780 < 0.0000000000000002 ***
TotalFullBath3  10208.830   1928.581   5.293   0.000000135764856 ***
TotalBsmtSF        -4.974      3.484  -1.428             0.153549
MasVnrArea         22.607      4.227   5.348   0.000000101183787 ***
PoolArea          -71.529     16.144  -4.431   0.000009998468161 ***
LotFrontage      -175.139     36.840  -4.754   0.000002163174653 ***
OpenPorchSF       -29.931      9.876  -3.031             0.002476 **
GarageCars       4206.774   1246.184   3.376             0.000753 ***
ScreenPorch        29.127     10.152   2.869             0.004166 **
OverallCond     -8027.837   1047.943  -7.661   0.000000000000031 ***
QualityIndex     1017.301    145.804   6.977   0.000000000004310 ***
WoodDeckSF         13.794      5.054   2.730             0.006408 **
BsmtUnfSF         -13.633      3.385  -4.028   0.000058723130983 ***
TotRmsAbvGrd     2227.058    772.591   2.883             0.003994 **
YearBuilt         -86.212     32.121  -2.684             0.007346 **
TotalHalfBath2 -10007.683   4972.343  -2.013             0.044308 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 25710 on 1688 degrees of freedom
Multiple R-squared:  0.9089,    Adjusted R-squared:  0.9079
F-statistic: 935.2 on 18 and 1688 DF,  p-value: < 0.00000000000000022
```

With QualityIndex removed from the training and test data, the stepwise auto selection process create the following model, replacing QualityIndex with OverallQual and ThreeSsnPorch:

```
> summary(stepwise.lm)

Call:
lm(formula = SalePrice ~ TotalSqftCalc + PriceSqft + MiscVal +
    TotalFullBath3 + OverallQual + TotalBsmtSF + MasVnrArea +
    PoolArea + LotFrontage + OpenPorchSF + GarageCars + ScreenPorch +
    OverallCond + WoodDeckSF + YearBuilt + BsmtUnfSF + TotRmsAbvGrd +
    TotalHalfBath2 + ThreeSsnPorch, data = train.clean)

Residuals:
    Min      1Q  Median      3Q     Max
-515813   -9799     571    9528  206742

Coefficients:
                 Estimate Std. Error t value             Pr(>|t|)
(Intercept)    35346.8768 63018.1061   0.561             0.574940
TotalSqftCalc     76.3286     2.6657  28.633 < 0.0000000000000002 ***
PriceSqft       1895.5059    46.4448  40.812 < 0.0000000000000002 ***
MiscVal           -9.0719     0.9351  -9.701 < 0.0000000000000002 ***
TotalFullBath3 10527.5928  1931.7665   5.450        0.00000005790 ***
OverallQual     5189.1602   850.1891   6.104        0.00000000128 ***
TotalBsmtSF       -5.9917     3.4891  -1.717             0.086116 .
MasVnrArea        21.9973     4.2497   5.176        0.00000025361 ***
PoolArea         -73.5379    16.1809  -4.545        0.00000589183 ***
LotFrontage     -172.1872    37.0642  -4.646        0.00000365299 ***
OpenPorchSF      -28.6392     9.9065  -2.891             0.003890 **
GarageCars      4212.2080  1251.0839   3.367             0.000777 ***
ScreenPorch       30.8668    10.1954   3.028             0.002503 **
OverallCond    -2590.6885   632.8507  -4.094        0.00004446501 ***
WoodDeckSF        14.3761     5.0714   2.835             0.004641 **
YearBuilt       -100.5492    32.7079  -3.074             0.002145 **
BsmtUnfSF        -12.8143     3.3873  -3.783             0.000160 ***
TotRmsAbvGrd    2186.5974   777.1107   2.814             0.004953 **
TotalHalfBath2 -9669.6751  4989.9644  -1.938             0.052811 .
ThreeSsnPorch     38.9110    27.0411   1.439             0.150348
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 25780 on 1687 degrees of freedom
Multiple R-squared:  0.9084,    Adjusted R-squared:  0.9073
F-statistic: 880.1 on 19 and 1687 DF,  p-value: < 0.00000000000000022
```

<u>This is the exact same model as newer version of forward.lm; because of this we are going to proceed with evaluating only two auto-generated models, forward.lm and backward.lm.</u>

This is the junk.lm model after Quality index is removed from the training and test data:

```
> summary(junk.lm)

Call:
lm(formula = SalePrice ~ OverallQual + OverallCond + GrLivArea +
    TotalSqftCalc, data = train.df)

Residuals:
    Min      1Q  Median      3Q     Max
-534710  -20055   -1534   15298  262006

Coefficients:
                 Estimate  Std. Error t value             Pr(>|t|)
(Intercept)   -109080.606    6853.128 -15.917 < 0.0000000000000002 ***
OverallQual     32442.868     902.398  35.952 < 0.0000000000000002 ***
OverallCond      -218.796     862.519  -0.254                  0.8
GrLivArea          23.258       3.411   6.817      0.0000000000128 ***
TotalSqftCalc      30.944       2.102  14.719 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 40690 on 1702 degrees of freedom
Multiple R-squared:  0.7698,	Adjusted R-squared:  0.7692
F-statistic:  1423 on 4 and 1702 DF,  p-value: < 0.00000000000000022
```

Display the final estimated models and their VIF values for each of these four models in your report.

```
> sort(vif(forward.lm),decreasing=TRUE)
 TotalSqftCalc    TotalBsmtSF      BsmtUnfSF    OverallQual     PriceSqft   TotRmsAbvGrd      YearBuilt     GarageCars TotalFullBath3
     12.039344       6.088095       5.183023       3.788864      3.605552       3.472082       2.529610       2.270156       1.613832
    MasVnrArea    LotFrontage    OverallCond    OpenPorchSF    WoodDeckSF       PoolArea TotalHalfBath2        MiscVal    ScreenPorch
      1.573853       1.391601       1.366219       1.231517      1.200939       1.107650       1.067638       1.051815       1.045868
 ThreeSsnPorch
      1.011518
> sort(vif(backward.lm),decreasing=TRUE)
    FirstFlrSF    BsmtFinSF1      BsmtUnfSF    OverallQual   SecondFlrSF      PriceSqft   TotRmsAbvGrd      YearBuilt     GarageCars
      5.099102      5.039201       3.920149       3.826619      3.810137       3.631887       3.506776       2.593264       2.276545
TotalFullBath3    MasVnrArea     BsmtFinSF2    LotFrontage   OverallCond    OpenPorchSF     WoodDeckSF       PoolArea   LowQualFinSF
      1.647103      1.597010       1.570819       1.456051      1.367317       1.233192       1.216692       1.113851       1.092457
TotalHalfBath2       MiscVal    ScreenPorch  ThreeSsnPorch
      1.071735      1.058008       1.047337       1.014914
> sort(vif(stepwise.lm),decreasing=TRUE)
 TotalSqftCalc    TotalBsmtSF      BsmtUnfSF    OverallQual     PriceSqft   TotRmsAbvGrd      YearBuilt     GarageCars TotalFullBath3
     12.039344       6.088095       5.183023       3.788864      3.605552       3.472082       2.529610       2.270156       1.613832
    MasVnrArea    LotFrontage    OverallCond    OpenPorchSF    WoodDeckSF       PoolArea TotalHalfBath2        MiscVal    ScreenPorch
      1.573853       1.391601       1.366219       1.231517      1.200939       1.107650       1.067638       1.051815       1.045868
 ThreeSsnPorch
      1.011518
> sort(vif(junk.lm),decreasing=TRUE)
    GrLivArea TotalSqftCalc   OverallQual   OverallCond
     3.401612      3.007441      1.714257      1.019195
>
```

With the chart above, removing QualityIndex greatly reduces the VIF values for the other variables in their respective models, thus reducing collinearity in each model as well.

Model Comparison: **Now that we have our final models, we need to compare the in-sample fit and predictive accuracy of our models. For each of these four models compute the adjusted R-Squared, AIC, BIC, mean squared error, and the mean absolute error for each of these models for the training sample. Each of these metrics represents some concept of 'fit'. In addition to the values provide the rank for each model in each metric. If a model is #2 in one metric, then is it #2 in all metrics? Should we expect each metric to give us the same ranking of model 'fit'.**

	R-Squared	AIC	BIC	MSE	MAE
forward.lm	0.9084	39543.98	39543.98	19710.14	13719.12
backward.lm	0.9084	39548.85	39679.47	19774.38	13751.21
junk.lm	0.7698	41086.35	41119.01	34966.59	25134.85

Based on the table above, forward.lm has lower AIC, BIC, MSE and MAE values than backward.lm, as well as have the same R2 value. forward.lm has also less explanatory variables in its model as well. While forward.lm outperforms or is equal to backward.lm in the above metrics, I do not expect that model comparisons will always have a dominating model over another one. If one model has higher AIC and lower BIC compared to another model (or vice versa), we might have to favor one lower score over the other based on the natures of each criterion. AIC tends to favor more complex models, while BIC performs model calculations better with those based on larger datasets.

(3) ***Predictive Accuracy***
In predictive modeling, we are interested in how well our model performs (predicts) out-of-sample. That is the point of predictive modeling. For each of the four models compute the Mean Squared Error (MSE) and the Mean Absolute Error (MAE) for the test sample. Which model fits the best based on these criteria? Did the model that fit best in-sample predict the best out-of-sample? Should we have a preference for the MSE or the MAE? What does it mean when a model has better predictive accuracy in-sample then it does out-of-sample?

We calculate Mean Absolute error (MAE) and Mean Squared Error (MSE) for each of the models with the test dataset below:

```
> # NOTE: forward.MAE/MSE = stepwise.MAE/MSE, since same model after removing QualityIndex
> forward.test <- predict(forward.lm,newdata=test.clean);
> forward.residuals <- test.clean$SalePrice - forward.test
> forward.absres <- abs(forward.residuals)
> forward.MAE <- mean(forward.absres)
> forward.MAE
[1] 13719.12
> forward.MSE <- sqrt(mean(forward.residuals^2))
> forward.MSE
[1] 19710.14
>
> backward.test <- predict(backward.lm,newdata=test.clean);
> backward.residuals <- test.clean$SalePrice - backward.test
> backward.absres <- abs(backward.residuals)
> backward.MAE <- mean(backward.absres)
> backward.MAE
[1] 13751.21
> backward.MSE <- sqrt(mean(backward.residuals^2))
> backward.MSE
[1] 19774.38
>
> junk.test <- predict(junk.lm,newdata=test.clean);
> junk.residuals <- test.clean$SalePrice - junk.test
> junk.absres <- abs(junk.residuals)
> junk.MAE <- mean(junk.absres)
> junk.MAE
[1] 25134.85
> junk.MSE <- sqrt(mean(junk.residuals^2))
> junk.MSE
[1] 34966.59
```

The forward.lm model has better (lower) MAE and MSE than backward.lm such that the differences in errors/residuals are smaller. With the range of residuals of SalePrice in the forward.lm model to be anywhere between -$80000 to $120000, MSE results in this study can be very biased based on the very large squared values calculated for these residuals. MSE can be preferred over MAE when dealing with sets of much smaller residuals, as squaring those values can help highlight differences between models.

A model has better predictive accuracy in-sample compared to out-sample due to that the model has been fitted by those in-samples, such that the residuals between predicted vales and actual values for in-sample datapoints can be expected to be smaller compared to out-sample datapoints. We'll see this in section 4, where in-sample (training) prediction grades for each model are better than out-sample (test) prediction grades.

(4) ***Operational Validation***

We have validated these models in the statistical sense, but what about the business sense? Do MSE or MAE easily translate to the development of a business policy? Typically, in applications we need to be able to hit defined cut-off points, i.e. we set a policy that we need to be p% accurate. Let's define a variable called PredictionGrade, and consider the predicted value to be 'Grade 1' if it is within ten percent of the actual value, 'Grade 2' if it is not Grade 1 but within

fifteen percent of the actual value, Grade 3 if it is not Grade 2 but within twenty-five percent of the actual value, and 'Grade 4' otherwise.

Produce these prediction grades for the in-sample training data and the out-of-sample test data. Note that we want to show these tables in distribution form, not counts. Distribution form is more informative and easier for your reader (and you!) to understand, hence we have normalized the table object.

forward.lm in-sample (training) Prediction Grades:

```
> forward.trainTable/sum(forward.trainTable)
forward.PredictionGrade
   Grade 1: [0.0.10] Grade 2: (0.10,0.15] Grade 3: (0.15,0.25]     Grade 4: (0.25+]
          0.74809607           0.13239602           0.06795548           0.05155243
```

forward.lm out-sample (test) Prediction Grades:

```
> forward.testTable/sum(forward.testTable)
forward.testPredictionGrade
   Grade 1: [0.0.10] Grade 2: (0.10,0.15] Grade 3: (0.15,0.25]     Grade 4: (0.25+]
          0.72144847           0.12534819           0.09888579           0.05431755
```

backward.lm in-sample (training) Prediction Grades:

```
> backward.trainTable/sum(backward.trainTable)
backward.PredictionGrade
   Grade 1: [0.0.10] Grade 2: (0.10,0.15] Grade 3: (0.15,0.25]     Grade 4: (0.25+]
          0.74868190           0.13005272           0.06912712           0.05213825
```

backward.lm out-sample (test) Prediction Grades:

```
> backward.testTable/sum(backward.testTable)
backward.testPredictionGrade
   Grade 1: [0.0.10] Grade 2: (0.10,0.15] Grade 3: (0.15,0.25]     Grade 4: (0.25+]
          0.72144847           0.12116992           0.10306407           0.05431755
```

junk.lm in-sample (training) Prediction Grades:

```
> junk.trainTable/sum(junk.trainTable)
junk.PredictionGrade
   Grade 1: [0.0.10] Grade 2: (0.10,0.15] Grade 3: (0.15,0.25]     Grade 4: (0.25+]
           0.4815466            0.1616872            0.1921500            0.1646163
```

junk.lm out-sample (test) Prediction Grades:

```
> junk.testTable/sum(junk.testTable)
junk.testPredictionGrade
   Grade 1: [0.0.10] Grade 2: (0.10,0.15] Grade 3: (0.15,0.25]     Grade 4: (0.25+]
           0.4456825            0.1977716            0.1894150            0.1671309
```

How accurate are the models under this definition of predictive accuracy? How do these results compare to our predictive accuracy results? Did the model ranking remain the same?

Note: The GSEs (Fannie Mae and Freddie Mac) rate an AVM model as 'underwriting quality' if the model is accurate to within ten percent more than fifty percent of the time. Are any of your models 'underwriting quality'?

For forward.lm, about 75% of in-sample datapoints performed within Grade 1, which is well over the Fannie Mae/Freddie Mac underwriting quality standards by almost 25%. About 72% of out-sample datapoints performed within Grade 1, which is above underwriting quality standards by 22%. backward.lm performed with about the same percentages as well under Grade 1 for in-sample and out-sample datapoints as well.

In regards to junk.lm, 48% of in-sample datapoints performed within Grade 1, which misses the mark for underwriting quality standards by 2%. About 45% of out-sample datapoints performed within Grade 1, which misses underwriting quality standards by 5%. While it is not 'underwriting quality', junk.lm almost qualifies for such a small model that only has four explanatory variables.

(5) 'Best' Model Selection

For which ever model you find to be "Best" after the automated variable selection procedures and all of these comparisons, you will need to re-visit that model and clean it up, as well as conduct residual diagnostics. Frankly, the end of an automated variable selection process is in many ways a starting point. What kinds of things do you want to check for and "clean up"?

Given that forward.lm has a lower AIC, BIC, MAE, and MSE than backward.lm, as well as the same R2, we will select forward.lm as our 'best' model.

Shown earlier, we have the following summary for forward.lm:

```
> summary(forward.lm)

Call:
lm(formula = SalePrice ~ OverallQual + TotalSqftCalc + PriceSqft +
    MiscVal + TotalFullBath3 + TotalBsmtSF + MasVnrArea + PoolArea +
    LotFrontage + OpenPorchSF + GarageCars + ScreenPorch + OverallCond
    WoodDeckSF + YearBuilt + BsmtUnfSF + TotRmsAbvGrd + TotalHalfBath2
    ThreeSsnPorch, data = train.clean)

Residuals:
    Min      1Q  Median      3Q     Max
-515813   -9799     571    9528  206742

Coefficients:
                 Estimate Std. Error t value              Pr(>|t|)
(Intercept)    35346.8768 63018.1061   0.561              0.574940
OverallQual     5189.1602   850.1891   6.104         0.00000000128 ***
TotalSqftCalc     76.3286     2.6657  28.633 < 0.0000000000000002 ***
PriceSqft       1895.5059    46.4448  40.812 < 0.0000000000000002 ***
MiscVal           -9.0719     0.9351  -9.701 < 0.0000000000000002 ***
TotalFullBath3 10527.5928  1931.7665   5.450         0.00000005790 ***
TotalBsmtSF       -5.9917     3.4891  -1.717              0.086116 .
MasVnrArea        21.9973     4.2497   5.176         0.00000025361 ***
PoolArea         -73.5379    16.1809  -4.545         0.00000589183 ***
LotFrontage     -172.1872    37.0642  -4.646         0.00000365299 ***
OpenPorchSF      -28.6392     9.9065  -2.891              0.003890 **
GarageCars      4212.2080  1251.0839   3.367              0.000777 ***
ScreenPorch       30.8668    10.1954   3.028              0.002503 **
OverallCond    -2590.6885   632.8507  -4.094         0.00004446501 ***
WoodDeckSF        14.3761     5.0714   2.835              0.004641 **
YearBuilt       -100.5492    32.7079  -3.074              0.002145 **
BsmtUnfSF        -12.8143     3.3873  -3.783              0.000160 ***
TotRmsAbvGrd    2186.5974   777.1107   2.814              0.004953 **
TotalHalfBath2 -9669.6751  4989.9644  -1.938              0.052811 .
ThreeSsnPorch     38.9110    27.0411   1.439              0.150348
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 25780 on 1687 degrees of freedom
Multiple R-squared:  0.9084,    Adjusted R-squared:  0.9073
F-statistic: 880.1 on 19 and 1687 DF,  p-value: < 0.00000000000000022
```

While forward.lm has less variables than backward.lm, we want see if we can further simplify the model without impacting R2 significantly.

We created a reduced model, forward.alt.lm, which removes the following variables:
ThreeSsnPorch: has high p-value
TotalHalfBath2: has high p-value
TotalBsmtSF: has high p-value and shows some collinearity with TotalSqftCalc as it is used in part of its calculation

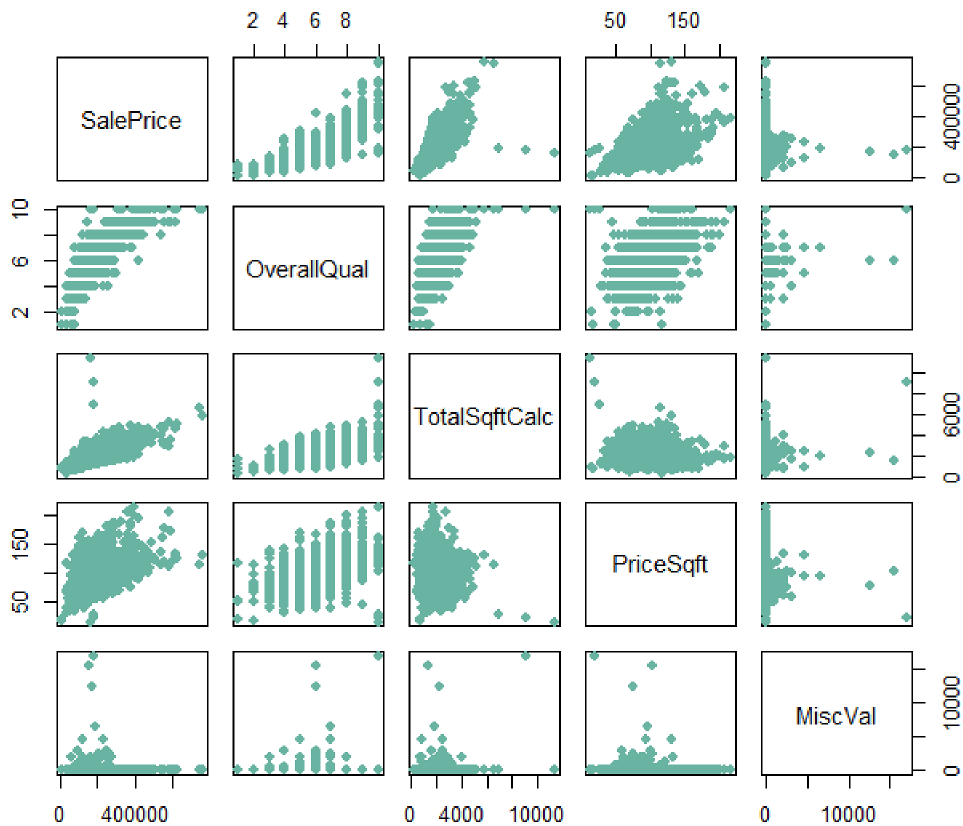
```
> summary(forward.alt.lm)

Call:
lm(formula = SalePrice ~ OverallQual + TotalSqftCalc + PriceSqft +
    MiscVal + TotalFullBath3 + MasVnrArea + PoolArea + LotFrontage +
    GarageCars + OverallCond + YearBuilt + BsmtUnfSF + TotRmsAbvGrd +
    ScreenPorch + OpenPorchSF + WoodDeckSF, data = train.clean)

Residuals:
    Min      1Q  Median      3Q     Max
-516830   -9564     543    9498  211083

Coefficients:
                 Estimate Std. Error t value             Pr(>|t|)
(Intercept)    55405.9240 62170.4779   0.891             0.372952
OverallQual     5221.2155   851.1790   6.134   0.0000000010642736 ***
TotalSqftCalc     72.8850     1.7433  41.809 < 0.0000000000000002 ***
PriceSqft       1907.0428    46.2428  41.240 < 0.0000000000000002 ***
MiscVal           -9.3603     0.9301 -10.064 < 0.0000000000000002 ***
TotalFullBath3 10890.4332  1922.4767   5.665   0.0000000172694056 ***
MasVnrArea        21.2908     4.2482   5.012   0.0000005959221488 ***
PoolArea         -74.2529    16.1662  -4.593   0.0000046902744643 ***
LotFrontage     -171.8718    37.0463  -4.639   0.0000037637792221 ***
GarageCars      4116.4523  1248.4374   3.297             0.000997 ***
OverallCond    -2620.3062   631.3660  -4.150   0.0000348687375742 ***
YearBuilt       -112.3799    32.1269  -3.498             0.000481 ***
BsmtUnfSF        -16.9221     2.2073  -7.666   0.0000000000000297 ***
TotRmsAbvGrd    2970.4955   598.1691   4.966   0.0000007525984113 ***
ScreenPorch       29.8709    10.2037   2.927             0.003463 **
OpenPorchSF      -28.1591     9.8576  -2.857             0.004334 **
WoodDeckSF        14.3714     5.0769   2.831             0.004699 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 25830 on 1690 degrees of freedom
Multiple R-squared:  0.9079,	Adjusted R-squared:  0.907
F-statistic:  1041 on 16 and 1690 DF,  p-value: < 0.00000000000000022
```

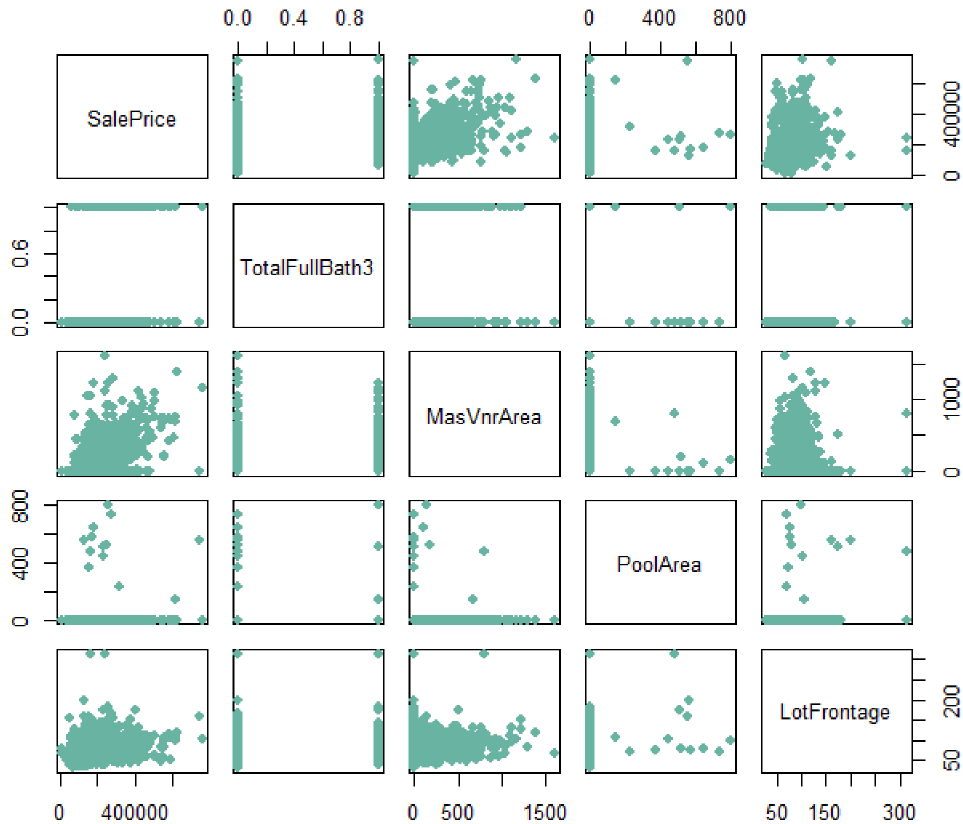
While we are trying to find ways to reduce the model, I am also curious why none of the auto-generated models did not include LotArea, a feature that is important to prospective homeowners. While manually adding LotArea to forward.alt.lm, I found that it had a very high p-value while not increasing or decreasing the overall R^2 score. While it seems that adding it would not ‘hurt’ the model statistically, it still adds complexity, and decided to leave it out.

With reduced model forward.alt.lm we see an improvement with less collinearity from evaluating the VIF:

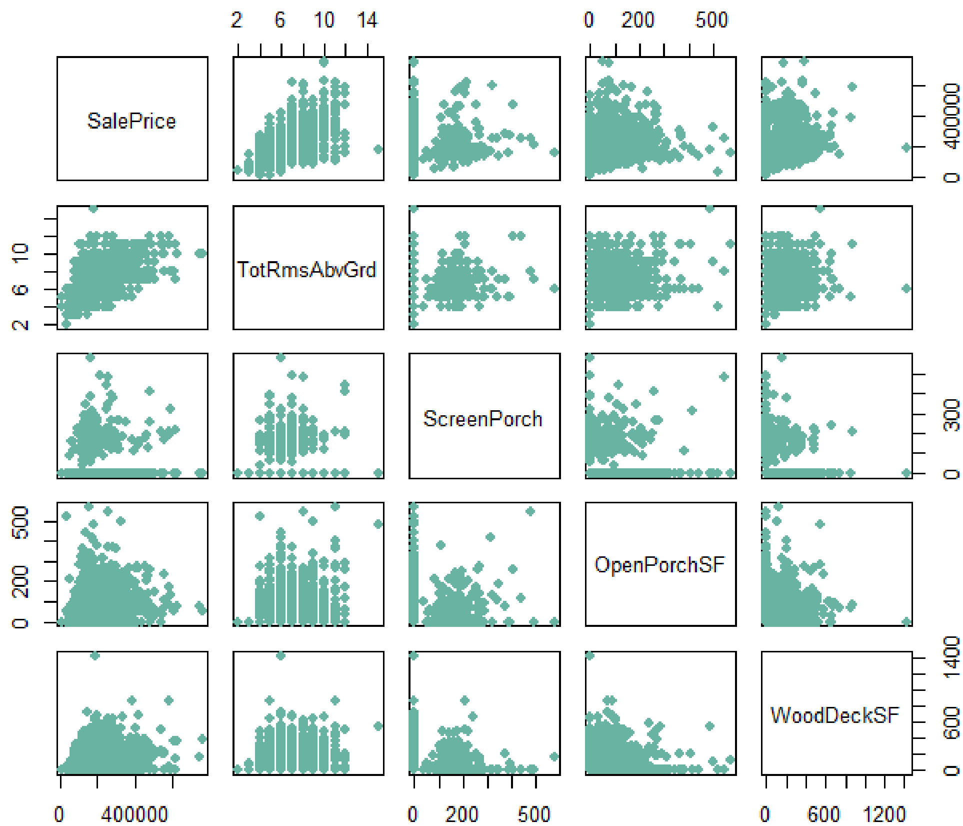
```
> sort(vif(forward.alt.lm),decreasing=TRUE)
 TotalSqftCalc     OverallQual      PriceSqft       YearBuilt      GarageCars       BsmtUnfSF    TotRmsAbvGrd TotalFullBath3      MasVnrArea
      5.132390        3.785528       3.562801        2.432723        2.253321        2.193986        2.050590       1.593228        1.567656
   LotFrontage     OverallCond    OpenPorchSF      WoodDeckSF        PoolArea     ScreenPorch         MiscVal
      1.385808        1.355461       1.215484        1.199658        1.102091        1.044213        1.037210
```

Here are scatterplots of the fullmodel.alt.lm explanatory variables to SalePrice:

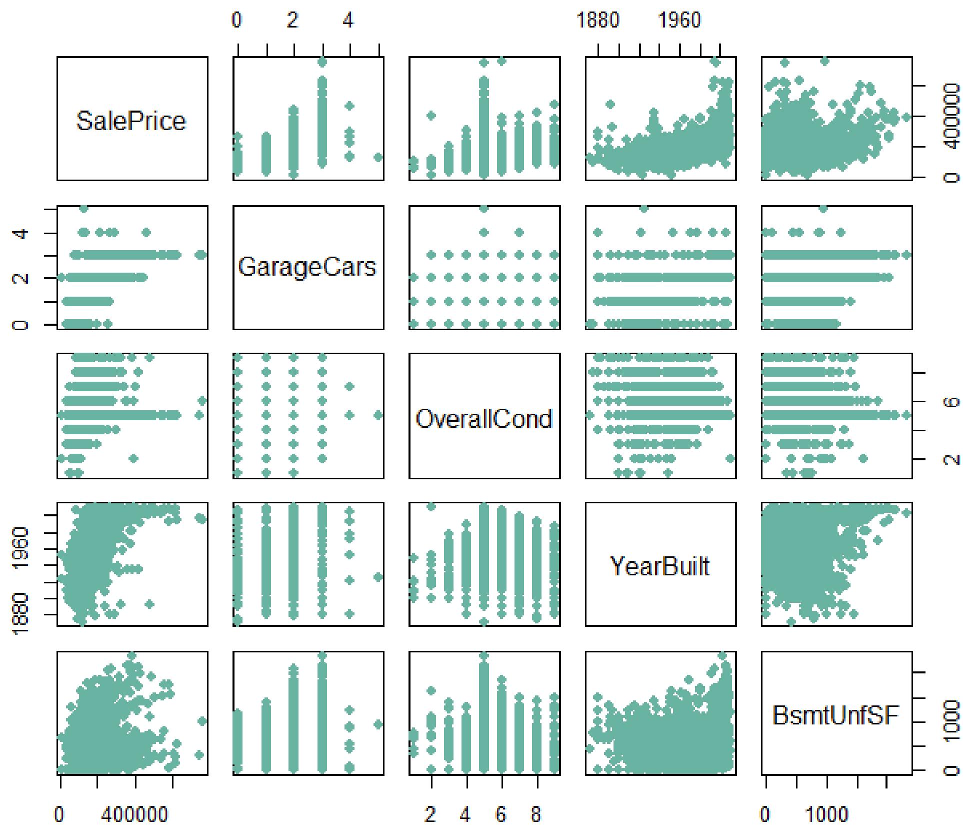




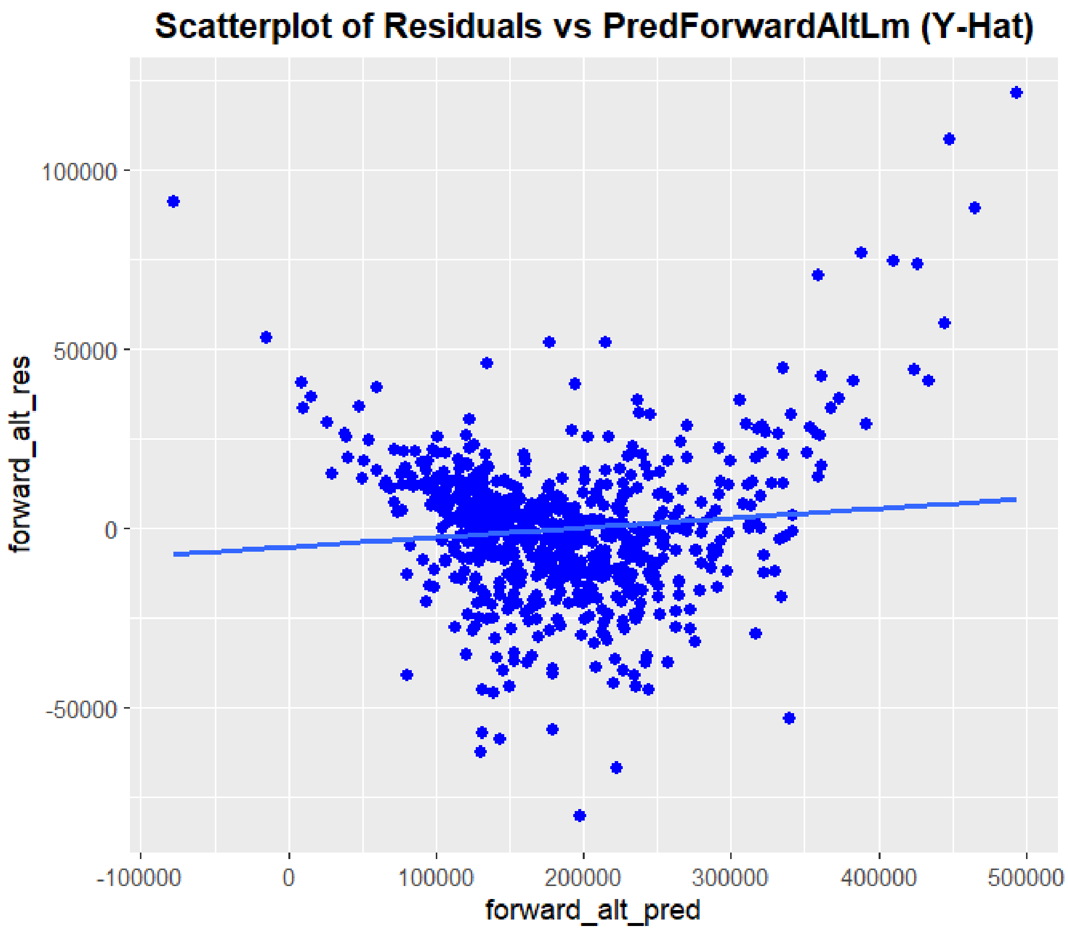
SalePrice
TotalFullBath3
MasVnrArea
PoolArea
LotFrontage
0.0
0.4
0.8
0
400
800
0
400000
0.0
0.6
1000
0
400
800
50
200
400000
500
1500
150
300



SalePrice
TotRmsAbvGrd
ScreenPorch
OpenPorchSF
WoodDeckSF
2
6
10
14
0
200
500
0
400000
0
400000
2
6
10
0
300
0
200
500
0
600
1400
0
400000
0
200
500
0
600
1200


SalePrice
GarageCars
OverallCond
YearBuilt
BsmtUnfSF
0
2
4
1880
1960
400000
0
1000
2
4
6
8

In terms of heteroskedasticity, while most values seem to be symmetrically distributed, outliers on the top and bottom can make the graph seem somewhat like a curve, similar to many other residual plots of similar models from the Ames dataset.



Let’s compare the reduced nested forward.alt.lm model with the full model forward.lm.

Hypothesis testing:

H_0: Beta17 = Beta18 = Beta19 = 0

H_A: One of Beta17, Beta18, Beta19 is not 0

Performing an ANOVA comparison between the two models gives us the following:

```
> anova(forward.alt.lm,forward.lm)
Analysis of Variance Table

Model 1: SalePrice ~ OverallQual + TotalSqftCalc + PriceSqft + MiscVal +
    TotalFullBath3 + MasVnrArea + PoolArea + LotFrontage + GarageCars +
    OverallCond + YearBuilt + BsmtUnfSF + TotRmsAbvGrd + ScreenPorch +
    OpenPorchSF + WoodDeckSF
Model 2: SalePrice ~ OverallQual + TotalSqftCalc + PriceSqft + MiscVal +
    TotalFullBath3 + TotalBsmtSF + MasVnrArea + PoolArea + LotFrontage +
    OpenPorchSF + GarageCars + ScreenPorch + OverallCond + WoodDeckSF +
    YearBuilt + BsmtUnfSF + TotRmsAbvGrd + TotalHalfBath2 + ThreeSsnPorch
  Res.Df           RSS Df  Sum of Sq      F  Pr(>F)
1   1690 1127216164139
2   1687 1121611032243  3 5605131896 2.8102 0.03822 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The comparison above gives us a p-value of 0.03822, which makes the full model forward.lm more statistically significant than forward.alt.lm.

The F-Statistic of forward.alt.lm nested within forward.lm is:

$$\mathbf{F_0} = \frac{SSE(RM)-SSE(FM)/(dim(FM)-dim(RM))}{SSE(FM)/(N-dim(FM))}$$

where dim(FM) = 19, dim(RM) = 16, N = 1707

$$= \frac{(1127216164139- 1121611032243)/(19 - 16)}{1121611032243/(1707- 19)}$$

$$= \frac{5605131896/3}{2886815085987/1688}$$

$$= 1868377299/ 1710198511= \mathbf{1.092491}$$

The $F_{Critical}$ value is **0.0719519** (in R: qf(p=0.05/2, df1=3, df2=1691), with df1=3 (df(FM)-df(RM)) and df2=1691(df(RM))

With $F_{Statistic} > F_{Critical}$, we can reject H0 and infer from this method that the full model forward.lm is more statistically significant than forward.alt.lm.

Given the tests results above, should we actually use forward.lm over forward.alt.lm? forward.alt.lm already has a very high R^2 value of 0.9079, which is only a 0.05% decrease of the variability compared to the full model. We removed three variables from the model that have a high p-value (ThreeSsnPorch, TotalHalfBath2, and TotalBsmtSF) which are not as statistically significant to the model compared to the other sixteen remaining variables. Despite the statistical evidence which favors the full model over the reduced model, I feel in this case having a simpler model (even if it’s just three less explanatory variables in this case) with a very slight reduction in R2 would be a better case in terms of interpretability and maintainability.

(6) ***For reflection / conclusions:***
After working on this problem and this data for several weeks, what are the challenges presented by the data? What are your recommendations for improving predictive accuracy? What do you think of the notion of parsimony: simpler models might be preferable over complicated models? Do we really need a max fit model or is a simpler but more interpretable model better?

As I work more with the Ames dataset, I'm learning more about the nuances about it. For example, I was curious about what is happening in regard to homes with 0 FullBath? I find out those homes have BsmtFullBath. I created a new variable TotalFullBath which combines FullBath and BsmtFullBath and now have a dataset that have homes that can better explain full bathrooms.

While I appreciate the new methodologies presented every week to help us dig further into the dataset to get a better understanding and create a more robust model, it can be a challenge to work with as we find out integrating and implementing them can cause us to start over more often through trial and error. With more practice and use, the process hopefully becomes more intuitive over time.

The variable auto selection processes should be used as a starting point for model creation, not the end, as our own experiences and expertise should play some sort of factor as well. As businesses and organizations are moving more towards data-driven models, we need to develop an understanding behind those models and document it. While I would often favor simpler models and parsimony over more complex models for the sake of interpretability, I would also have to understand how these models would react to outliers and edge cases, and be open to re-evaluate and continuously improve them as the dataset and use cases can change over time. That being said, the use of neural networks and their implementation of potentially millions of possible features at scale would make mitigating complexity a daunting task for anyone.