Automated Variable Selection, Multicollinearity, and Predictive Modeling

Claire Boetticher, MSDS 410, SEC 56

November 1, 2020

Section 1: Introduction

The original Ames housing data set contains information from the Ames Assessor's Office for tax assessment purposes, specifically computing assessed values for single residential properties sold in Ames, Iowa from 2006 to 2010. The objective of this analysis is to explore the relationship between predictor variables related to home size, lot size, and build material quality and the response variable, sale price. Regression models are built in a predictive modeling framework, automated variable selection techniques are explored for model identification, cross-validation is applied to assess the models' predictive accuracy, and statistical model validation is compared to business model validation based on results.

Section 1.1: Sample Definition

The original dataset includes 2930 observations of 82 variables. Certain observations are not appropriate for the development and evaluation of regression models for this study so seven drop conditions are applied to narrow the observation subset to the most appropriate sample. If an observation does not satisfy the condition, it is removed from the eligible sample.

Table 1: Drop Conditions for Ames Dataset

Condition	Count of Observations Dropped Based on Condition
Not SFR	505
Non-Normal Sale	423
Street Not Paved	6
Built Pre-1950	489
No Basement	28
LT 800 SqFt	9
Not Public Utilities	1

Following removal of the observations from Table 1, the eligible sample includes 1469 observations from the original 2930. Seven additional variables are created for model development and assessment (Table 2).

Table 2: Additional Variables

Variable Name	Logic
QualityIndex	Product of OverallQual and OverallCond
TotalSqftCalc	Sum of BsmtFinSF1,SF2, and GrLivArea
HouseAge	Years since house was built (from 2020)
LotSpace	Indicator variable (1=Corner, 0=Inside)
HouseStyle	Indicator variable (1=2Story, 0=1Story)
KitchenQuality	Factor with 5 levels (Ex, Gd, TA, Fa, Po)
BasementCondition	Factor with 6 levels (Ex, Gd, TA, Fa, Po, NA)

Section 1.2: Train/Test Split

In order to assess the regression models' performance both in- and out-of-sample, basic cross-validation with a 70/30 train test split is applied to divide the sample. 70 percent of the 1469 observations are used for in-sample model development and 30 percent are used for out-of-sample model assessment. Table 3 shows resulting counts for subsequent model development and testing, reflecting the split.

Table 3: Train and Test Split

Dataset	Count	Percent of Total
Train	1037	70.5922
Test	432	29.4078
Total	1469	100

Section 2: Model Identification and In-Sample Model Fit

Fifteen candidate predictor variables are selected to assess predictive influence, including the additional variables defined in Section 1.1.

Table 4: Selected Variables and Definitions

Variable	Definition
SalePrice	Sale price in $ (Response Variable)
LotArea	Lot size in square feet
TotalBsmtFinSF	Total square feet of basement area
FullBath	Full bathrooms above grade
HalfBath	Half baths above grade
BedroomsAbvGr	Bedrooms above grade
TotRmsAbvGrd	Total number of rooms above grade (does not include bathrooms)
Fireplaces	Number of fireplaces
GarageArea	Size of garage in square feet
QualityIndex	Index value for quality: product of OverallQual (rating of overall material and finish of the house, 1=very poor to 10=very excellent) and OverallCond (rating of overall condition of the house, 1=very poor to 10=very excellent)
TotalSqftCalc	Total square footage of house
HouseAge	Years since house was built (from 2020)
LotSpace	Indicator variable (1=Corner, 0=Inside)
HouseStyle	Indicator variable (1=2Story, 0=1Story)
KitchenQuality	Factor with 5 levels (Ex, Gd, TA, Fa, Po)
BasementCondition	Factor with 6 levels (Ex, Gd, TA, Fa, Po, NA)

Three automated variable selection techniques are used with the defined training set and the variables listed in Table 4 to find the 'best' model, measured on in-sample fit and predictive performance out-of-sample. The techniques included are forward variable selection, backward variable selection, stepwise variable selection, and a fourth 'junk' model that includes correlated predictor variables for comparison purposes to the three models based on automatic variable selection.

Variance Inflation Factor (VIF) values, which measure how much the variance of a regression coefficient is inflated due to multicollinearity in the model, are calculated for each model as a potential diagnostic for that condition. Multicollinearity, where collinearity exists between three or more variables with the possibility of redundancy between predictor variables, could cause regression estimates to be unstable. Additional model metrics are calculated to further evaluate model performance and models are ranked according to each: Adjusted R-Squared, the Akaike information criterion (AIC), Bayesian information criterion (BIC), Mean Squared Error (MSE), Mean Absolute Error (MAE).

Section 2.1: Forward Variable Selection

Forward variable selection entails starting with an Intercept model with no predictors and adding predictors one by one based on subsequent variables with the highest simple correlation with the response variable, SalePrice. If the regression coefficient of the variable is significantly different from zero, it is retained and the process is continued. Table 5 contains regression results using this technique.

Table 5: Forward Variable Selection

	Dependent variable:
	SalePrice
TotalSqftCalc	45.99***
	(1.67)
KitchenQualityFa	-55,153.74***
	(12,862.00)
KitchenQualityGd	-58,547.54***
	(3,832.23)
KitchenQualityTA	-63,857.44***
	(4,473.48)
HouseAge	-977.93***
	(69.23)
QualityIndex	1,473.17***
	(134.22)
GarageArea	57.94***
	(6.03)
LotArea	1.32***
	(0.16)
Fireplaces	6,109.68***
	(1,530.14)
BasementConditionFa	37,003.07**
	(16,963.33)
BasementConditionGd	37,964.91**
	(15,921.37)
BasementConditionTA	44,241.22***
	(15,503.80)
FullBath	4,114.55*
	(2,296.55)
Constant	42,766.07**
	(18,155.01)
Observations	1,037

R^2	0.87
Adjusted R^2	0.87
Residual Std. Error	26,549.31 (df = 1023)
F Statistic	525.92*** (df = 13; 1023)
Note:	*p**p***p<0.01

This model retained nine of the original fifteen predictor variables, not selecting TotalBsmtFinSF. HalfBath, BedroomsAbvGr, TotRmsAbvGr, LotSpace, and Style based on insignificant regression coefficients. TotalSqftCalc is a reasonable first variable for selection. Kitchen quality as the next result is somewhat curious considering its negative relationship with SalePrice from this model. The intercept adjustments for HouseAge and QualityIndex reflect a reasonable and significant impact on SalePrice as well, with the age of a house correlating indirectly with SalePrice and QualityIndex correlating directly. The F-statistic for the model is 525.92. Compared with the critical F value with 13 and 1023 degrees of freedom (1.7297), the calculated value is much larger, providing evidence against results occurring by chance.

Table 6: Variance Inflation Factor Values for Forward Variable Selection Model

Variable	GVIF	Degrees of Freedom
TotalSqftCalc	1.9660	1
KitchenQuality	2.8178	3
HouseAge	2.5254	1
QualityIndex	1.5760	1
GarageArea	1.7206	1
LotArea	1.5662	1
Fireplaces	1.4809	1
BasementCondition	1.0888	3
FullBath	2.2217	1

Variance inflation factor values shown in Table 6, as a potential diagnostic for multicollinearity among variables, are not particularly high for this model, ranging from approximately one to three. The highest value of 2.8178 for KitchenQuality is not of particular concern since that is an indicator variable with 4 levels and a small proportion of cases in the reference category (Ex quality), which will result in a higher VIF value. VIF results for most of these variables suggest that standard error may not be overly inflated and the variables included do not pose a particular issue for collinearity.

Section 2.2: Backward Variable Selection

Backward variable selection begins with the full regression equation and successively eliminates variables one at a time based on contribution to the reduction of error sum of squares. Table 7 shows regression results for the model with this technique applied. The same model is returned as forward variable selection, though variables are eliminated in a different order than their addition in forward variable selection.

Table 7: Backward Variable Selection

	Dependent variable:
	SalePrice
LotArea	1.32***
	(0.16)
FullBath	4,114.55*
	(2,296.55)
Fireplaces	6,109.68***
	(1,530.14)
GarageArea	57.94***
	(6.03)
QualityIndex	1,473.17***
	(134.22)
TotalSqftCalc	45.99***
	(1.67)
HouseAge	-977.93***
	(69.23)
KitchenQualityFa	-55,153.74***
	(12,862.00)
KitchenQualityGd	-58,547.54***
	(3,832.23)
KitchenQualityTA	-63,857.44***
	(4,473.48)
BasementConditionFa	37,003.07**
	(16,963.33)
BasementConditionGd	37,964.91**
	(15,921.37)
BasementConditionTA	44,241.22***
	(15,503.80)
Constant	42,766.07**
	(18,155.01)
Observations	1,037
R^2	0.87
Adjusted R^2	0.87
Residual Std. Error	26,549.31 (df = 1023)
F Statistic	525.92*** (df = 13; 1023)
Note:	*p**p***p<0.01

With the same model returned as forward variable selection, the same VIF values are also calculated, shown in Table 8.

Table 8: Variance Inflation Factor Values for Backward Variable Selection Model

Variable	GVIF	Degrees of Freedom
LotArea	1.1566	1
FullBath	2.2217	1
Fireplaces	1.4809	1
GarageArea	1.7206	1
QualityIndex	1.5760	1
TotalSqftCalc	1.9660	1
HouseAge	2.5253	1
KitchenQuality	2.8178	3
BasementCondition	1.0888	3

Section 2.3: Stepwise Variable Selection

The stepwise variable selection technique is similar to forward selection except that at each stage, the possibility exists of deleting a variable as well. Inclusion and deletion criteria are the same as for forward and backward variable selection, respectively. A single linear regression model with TotalSqftCalc selected as the predictor is used to initialize stepwise selection. With TotalSqftCalc as the first predictor variable selected in the forward variable selection process, this is a reasonable choice. Akaike's information criterion (AIC) serves criteria for the stepwise procedure, where addition is terminated once the addition of a variable causes no reduction in AIC. AIC is essentially a metric enabling assessment of model fit and simplicity. Table 9 shows regression results for the model with this technique applied. The same model is returned as forward and backward variable selection, with variables added in the same order as the forward variable selection model.

Table 9: Stepwise Variable Selection

	Dependent variable:
	SalePrice
TotalSqftCalc	45.99^{***}
	(1.67)
KitchenQualityFa	$-55,153.74^{***}$
	(12,862.00)
KitchenQualityGd	$-58,547.54^{***}$
	(3,832.23)
KitchenQualityTA	$-63,857.44^{***}$
	(4,473.48)
HouseAge	-977.93^{***}
	(69.23)
QualityIndex	$1,473.17^{***}$
	(134.22)
GarageArea	57.94^{***}
	(6.03)
LotArea	1.32^{***}
	(0.16)
Fireplaces	$6,109.68^{***}$

	(1,530.14)
BasementConditionFa	37,003.07**
	(16,963.33)
BasementConditionGd	37,964.91**
	(15,921.37)
BasementConditionTA	44,241.22***
	(15,503.80)
FullBath	4,114.55*
	(2,296.55)
Constant	42,766.07**
	(18,155.01)
Observations	1,037
R^2	0.87
Adjusted R^2	0.87
Residual Std. Error	26,549.31 (df = 1023)
F Statistic	525.92*** (df = 13; 1023)
Note:	*p**p***p<0.01

As with forward and backward variable selection, the same model returned for stepwise selection also has the same VIF values calculated (Table 10).

Table 10: Variance Inflation Factor Values for Stepwise Variable Selection Model

Variable	GVIF	Degrees of Freedom
TotalSqftCalc	1.9660	1
KitchenQuality	2.8178	3
HouseAge	2.5253	1
QualityIndex	1.5760	1
GarageArea	1.7206	1
LotArea	1.1566	1
Fireplaces	1.4809	1
BasementCondition	1.0888	3
FullBath	2.2217	1

Section 2.4: Junk Model

A 'junk' model is developed as a point of comparison for the other three models. It is intentionally designed with collinearity in mind, including variables created from and highly dependent on each other: QualityIndex is the product of OverallQual and Overall Cond; likewise, TotalSqftCalc includes GrLivArea in its sum. Table 11 shows regression results for this model, all with significant p-values. This model's results, such as a relatively high Adjusted R-squared value and an F-statistic of 1113.97 (compared with the critical F value of 2.2227 with 5 and 1031 degrees of freedom), demonstrate the need to evaluate relationships amongst predictor variables and not implicitly trust results suggesting model goodness-of-fit and significance.

Table 11: Junk Variable Selection

	Dependent variable:
	SalePrice
OverallQual	56,736.12***
	(4,682.02)
OverallCond	34,118.05***
	(5,206.71)
QualityIndex	-5,431.94***
	(871.63)
GrLivArea	31.04***
	(3.34)
TotalSqftCalc	39.92***
	(2.06)
Constant	-295,181.10***
	(28,647.24)
Observations	1,037
R^2	0.84
Adjusted R^2	0.84
Residual Std. Error	28,971.11 (df = 1031)
F Statistic	1,113.97*** (df = 5; 1031)
Note:	*p**p***p<0.01

OverallQual, OverallCond, and QualityIndex all show very high VIF values (Table 12), to be expected given their built-in correlation. TotalSqftCalc and GrLivArea have much smaller VIF values, though they might be higher if the junk model included the other two variables making up TotalSqftCalc (BsmtFinSF1 and BsmtFinSF2).

Table 12: Variance Inflation Factor Values for Junk Model

Variable	GVIF	Degrees of Freedom
OverallQual	43.2682	1
OverallCond	29.9796	1
QualityIndex	55.8185	1
GrLivArea	3.2138	1
TotalSqftCalc	2.5171	1

Section 2.5: Model Comparison

Table 13 shows results of various metrics calculated across all four models as a means of assessing performance in-sample.

Table 13: Model Predictive Accuracy Metrics and Ranking (In-Sample)

Model	Adjusted R-Squared (Rank)	AIC (Rank)	BIC (Rank)	MSE (Rank)	RMSE (Rank)	MAE (Rank)
Forward	0.8682 (1)	24086.12 (1)	24160.28 (1)	695349752 (1)	26369.49 (1)	18639.59 (1)
Backward	0.8682 (1)	24086.12 (1)	24160.28 (1)	695349752 (1)	26369.49 (1)	18639.59 (1)
Stepwise	0.8682 (1)	24086.12 (1)	24160.28 (1)	695349752 (1)	26369.49 (1)	18639.59 (1)
Junk	0.8431 (2)	24259.25 (2)	24293.86 (2)	834468841 (2)	28887.17 (2)	20800.16 (2)

The three models using automated variable selection, all returning the same results, out-perform the junk model across all metrics. The difference in calculations is relatively small for Adjusted R-squared, which would not stand alone as a suitable metric for performance given the variables included in the junk model. The difference is larger across the other metrics, with higher AIC and BIC values for the junk model, used as a proxy for model fit and simplicity. MSE values for all four models are high, on the hundreds of millions scale. Representing the average squared difference between the estimated values and the actual value, MSE assigns more weight to bigger errors and may reflect outliers worth examining. Though an ideal (and virtually impossible) MSE is zero, there is no scale for what constitutes a 'high' MSE; it serves more in this case as a comparison method between predictive models. MAE, as the average of all absolute error, is generally less sensitive to outliers and less biased for higher values. As a point of comparison on the same scale, RMSE (the square root of MSE) is larger than MAE for all four models but the ranking remains the same of forward, backward, and stepwise models versus the junk model. Taken together, this set of metrics provides a useful approach to comparing the four models' performance, however they do not provide an objective picture of model goodness-of-fit necessarily without attention to the potential issues of applying automated variable selection versus a regimented process of validation through calculation and attention to single predictors, other possible combinations of predictors, and also interaction amongst predictors.

Section 3: Predictive Accuracy

To assess how well each model performs out-of-sample, thus its potential utility for predictive modeling, accuracy metrics are calculated using the test dataset. Table 14 results show MSE, RMSE, and MAE values for all four models and their relative performance ranking.

Table 14: Model Predictive Accuracy Metrics and Ranking (Out-of-Sample)

Model	MSE (Rank)	RMSE (Rank)	MAE (Rank)
Forward	764150847 (2)	27643.28 (2)	18201.22 (1)
Backward	764150847 (2)	27643.28 (2)	18201.22 (1)
Stepwise	764150847 (2)	27643.28 (2)	18201.22 (1)
Junk	692309614 (1)	26311.78 (1)	18908.85 (2)

In terms of MSE (and thus RMSE), the junk model performs better than the forward, backward, and stepwise variable selection models out-of-sample. In terms of MAE, the forward, backward, and stepwise models perform better than the junk model with lower error values. Comparing in-sample versus out-of-sample performance, the error values for MSE/RMSE are higher for the forward, backward, and stepwise models on testing data than on training data but lower for testing data for MAE. This suggests potential issues with performance on unseen data. As for the different ranking of

MSE versus MAE, this could possibly signal issues with outliers, to which MSE scoring is more sensitive. For the junk model, all error values for the two metrics are higher on training data than testing. This does not necessarily indicate decent performance on unseen data but more likely issues with the model design and need for adjustment to account for collinearity.

Section 4: Operational Validation

In order to determine operational validity of the four models in concrete business terms, with a scoring method and criteria for performance, a PredictionGrade variable is created and each predicted value is scored on how close it is to the actual SalePrice value, in both train and test datasets. An observation with a Grade 1 score falls within 10 percent of the actual SalePrice value. Observations with a Grade 2 score are within 15 percent of the actual value, Grade 3 within 25 percent, and Grade 4 otherwise. The bounds on the error serve to assess the model's distribution of accuracy in-sample and out-of-sample as a way to determine potential utility and future performance on unseen data. Table 15 shows Grade results for each model with the training data. With the forward, backward, and stepwise variable selection models in-sample, approximately 60 percent of observations' predicted SalePrice values fall within 10 percent of the actual SalePrice and another 20 percent fall within 15 percent. Five percent of observations received a Grade 4. The junk model has closer to 50 percent of observations at Grade 1, similar distributions for Grades 2 and 3, and higher proportions of Grade 4 scores.

Table 15: Operational Validation – Distribution by Grade (In-Sample Performance)

Model	Grade 1	Grade 2	Grade 3	Grade 4
Forward	0.6008	0.2025	0.1427	0.0540
Backward	0.6008	0.2025	0.1427	0.0540
Stepwise	0.6008	0.2025	0.1427	0.0540
Junk	0.5535	0.2035	0.1553	0.0878

Assessing out-of-sample performance on these models using the Grade criteria, performance is slightly higher for forward, backward, and stepwise variable selection models but the distribution across all Grade levels is roughly the same as with the training dataset (Table 16). This suggests the potential for reliable performance on similar but unseen data for those three models. The junk model's distribution of Grade scores out-of-sample also closely resembles that for in-sample performance, with slightly better performance reflected in the Grade 1 and 2 counts and fewer in the lower score 3 and 4.

Table 16: Operational Validation – Distribution by Grade (Out-of-Sample Performance)

Model	Grade 1	Grade 2	Grade 3	Grade 4
Forward	0.6343	0.1898	0.1343	0.0417
Backward	0.6343	0.1898	0.1343	0.0417
Stepwise	0.6343	0.1898	0.1343	0.0417
Junk	0.5903	0.2106	0.1458	0.0532

Comparing this method of operational validity to predictive accuracy, assuming the proportion of observations meeting the criteria for a Grade 1 score as a comparable measure of success to MSE and MAE, model ranking is the same in-sample, with forward, backward, and stepwise models out-performing the junk model. For out-of-sample performance with the same comparison, model ranking with respect to Grade 1 scoring still places the three automated variable selection models over the junk model; this mirrors the ranking on predictive accuracy out-of-sample for MAE scores but not MSE scores, where the junk model performs slightly better than the other three models. Defined by 'underwriting quality' where a model is accurate to within 10 percent more than 50 percent of the time, all of these models for this sample meet that criteria (Table 15 and 16). If operational requirements for model performance were defined at falling within 15 percent of the actual predicted SalePrice value, having 80 percent of training and test data pass that requirement could be considered decent performance with this sample as well (combining the first two Grade distributions). Evaluating performance in this manner can give a bit more structure and context to the nebulous task of measuring based on a combination of standard metrics. The concept of operational validity depends on the organization itself, the quality of data, and the criticality and risks of decisions made based on model output, to name only a few factors. The definition of 'success' will thus vary.

R Script

```
####################################
# Claire Boetticher
# 11.1.2020
# Assignment 7
####################################
# Load packages
library(stargazer)
library(MASS)
library(car)
library(mctest)

##############################################################################
# Section 1: Read in the csv file and create subsample
##############################################################################

# Set working directory
path <- "/Users/clb/Documents/MSDS410/7-Variable_Selection/"
setwd(path)

# Read in csv file for Ames housing data;
path.name <- '/Users/clb/Documents/MSDS410/7-Variable_Selection/data/';
file.name <- paste(path.name,'ames_housing_data.csv',sep='');

# Read in the csv file into an R data frame;
# Reading stringsAsFactors=FALSE stops R from converting the character data into factors;
ames.df <- read.csv(file.name,header=TRUE,stringsAsFactors=FALSE);

# Create a waterfall of drop conditions;
# Work the data frame as a 'table' like you would in SAS or SQL;
ames.df$dropCondition <- ifelse(ames.df$BldgType!='1Fam','01: Not SFR',
                                ifelse(ames.df$SaleCondition!='Normal','02: Non-Normal Sale',
                                       ifelse(ames.df$Street!='Pave','03: Street Not Paved',
                                              ifelse(ames.df$YearBuilt <1950,'04: Built Pre-1950',
                                                     ifelse(ames.df$TotalBsmtSF <1,'05: No
Basement',
                                                            ifelse(ames.df$GrLivArea <800,'06: LT
800 SqFt',

ifelse(ames.df$Utilities!='AllPub','07: Not Public Utilities',
                                                                          '99: Eligible Sample')
                                                            )))))));

# Save the table
waterfall <- table(ames.df$dropCondition)

# Format the table as a column matrix for presentation
```

```
as.matrix(waterfall,7,1)

# Eliminate all observations that are not part of the eligible sample population
eligible.population <- subset(ames.df,dropCondition=='99: Eligible Sample')

# Check that all remaining observations are eligible
table(eligible.population$dropCondition)

# Save the data as an R data file;
saveRDS(object=eligible.population, file=paste(path.name,'Ames_eligible_sample_test.Rdata',sep='')
)

# Check the file by reading it back into R;
check.file <- readRDS(paste(path.name,'Ames_eligible_sample.Rdata',sep=''));
str(check.file)

##############################################################################
# Section 2: Predictive Modeling Framework
##############################################################################

# Read in subsample data
my.data <- readRDS('data/Ames_eligible_sample.RData')

# Set the seed on the random number generator to get the same split every time code is run
# 0-1: 70% interval becomes 70% training set
set.seed(123)
my.data$u <- runif(n=dim(my.data)[1],min=0,max=1)

# Define variables for later use
my.data$QualityIndex <- my.data$OverallQual*my.data$OverallCond
my.data$TotalSqftCalc <- my.data$BsmtFinSF1+my.data$BsmtFinSF2+my.data$GrLivArea
my.data$HouseAge <- 2020 - my.data$YearBuilt

# Define indicator variables

# LotSpace - from LotConfig
my.data$LotSpace <- ifelse(my.data$LotConfig=='Corner',1,0);
# my.data$LotSpace <- as.factor(my.data$LotSpace)

# Style - from HouseStyle
my.data$Style <- ifelse(my.data$HouseStyle=='2Story',1,0);
# my.data$Style <- as.factor(my.data$Style)

# KitchenQuality - from KitchenQual
# my.data$KitchenQuality <- my.data$KitchenQual
# factor(my.data$KitchenQuality,levels=c("Ex","Gd","TA","Fa","Po"))
my.data$KitchenQuality <- as.factor(my.data$KitchenQual)
```

```
# BasementCondition - from BsmtCond
my.data$BasementCondition <- as.factor(my.data$BsmtCond)

head(my.data)
names(my.data)
str(my.data)

# Create train/test split
train.df <- subset(my.data, u<.70)
test.df <- subset(my.data, u>=.70)

# Check data split. Sum of parts should equal whole
# 1469 is sample size
# 1037 = ~70% of sample
# 432 = ~30% of sample
dim(my.data)[1]
dim(train.df)[1]
dim(test.df)[1]
dim(train.df)[1]+dim(test.df)[1]

###############################################################################
# Section 3: Model Identification by Automated Variable Selection
###############################################################################
# Select 15-20 predictor variables, a mix of discrete and continuous ones
# Include QualityIndex and TotalSqftCalc

# Create dataframe with only response variable + selected predictor variable
include.list <-
  c(
    'LotArea',
    'TotalBsmtFinSF',
    'FullBath',
    'HalfBath',
    'BedroomsAbvGr',
    'TotRmsAbvGr',
    'Fireplaces',
    'GarageArea',
    'SalePrice',
    'QualityIndex',
    'TotalSqftCalc',
    'HouseAge',
    'LotSpace',
    'Style',
    'KitchenQuality',
    'BasementCondition'
  )
```

```r
# Alternate: create dataframe with only response variable + selected predictor variable
# using keep strategy

# If using drop.list
# train.clean <-train.df[,!(names(my.data) %in% drop.list)]

# If using include.list
train.clean <-train.df[,(names(my.data) %in% include.list)]

head(train.clean)

# Check for missing values
sapply(train.clean, function(x) sum(is.na(x)))

# Define upper and lowers models for stepAIC()
# AIC is penalized likelihood approach to variable selection, smallest AIC wins
# Emphasizes trade-off between model fit and complexity

help(stepAIC)

# Define the upper model as the FULL model
upper.lm <- lm(SalePrice ~ .,data=train.clean)
summary(upper.lm)

# Define the lower model as the Intercept model
lower.lm <- lm(SalePrice ~ 1,data=train.clean)
summary(lower.lm)

# Need a SLR to initialize stepwise selection, TotalSqftCalc selected as best for now
sqft.lm <- lm(SalePrice ~ TotalSqftCalc,data=train.clean)
summary(sqft.lm)

# Forward Variable Selection
# Model object uses formula component, knows dataset to use
forward.lm <- stepAIC(object=lower.lm,scope=list(upper=formula(upper.lm),lower=~1),
                      direction=c('forward'))
summary(forward.lm)

# Model output table
out.path <- '/Users/clb/Documents/MSDS410/7-Variable_Selection/report_outputs/'
file.name <- 'forwardlm.html';
stargazer(forward.lm, type=c('html'),out=paste(out.path,file.name,sep=''),
          title=c('Table XX: Forward Variable Selection'),
          align=TRUE, digits=2, digits.extra=2, initial.zero=TRUE)

# Backward Variable Selection
```

```
backward.lm <- stepAIC(object=upper.lm,direction=c('backward'))
summary(backward.lm)

# critical value of F = 1.729722
qf(.95,13,1023)

# Model output table
out.path <- '/Users/clb/Documents/MSDS410/7-Variable_Selection/report_outputs/'
file.name <- 'backwardlm.html';
stargazer(backward.lm, type=c('html'),out=paste(out.path,file.name,sep=''),
          title=c('Table XX: Backward Variable Selection'),
          align=TRUE, digits=2, digits.extra=2, initial.zero=TRUE)

# StepAIC()
stepwise.lm <-stepAIC(object=sqft.lm,scope=list(upper=formula(upper.lm),lower=~1),
                      direction=c('both'))
summary(stepwise.lm)

# Model output table
out.path <- '/Users/clb/Documents/MSDS410/7-Variable_Selection/report_outputs/'
file.name <- 'stepwiselm.html';
stargazer(stepwise.lm, type=c('html'),out=paste(out.path,file.name,sep=''),
          title=c('Table XX: Stepwise Variable Selection'),
          align=TRUE, digits=2, digits.extra=2, initial.zero=TRUE)

# Junk model with original dataset
junk.lm <-lm(SalePrice ~ OverallQual + OverallCond + QualityIndex + GrLivArea +
               TotalSqftCalc, data=train.df)
summary(junk.lm)

# critical value of F = 2.222783
qf(.95,5,1031)

# Model output table
out.path <- '/Users/clb/Documents/MSDS410/7-Variable_Selection/report_outputs/'
file.name <- 'junklm.html';
stargazer(junk.lm, type=c('html'),out=paste(out.path,file.name,sep=''),
          title=c('Table XX: Junk Variable Selection'),
          align=TRUE, digits=2, digits.extra=2, initial.zero=TRUE)

help(vif)

# Compute VIF values for variable selection models
vif(forward.lm)
vif(backward.lm)
vif(stepwise.lm)
vif(junk.lm)
```

```
# Additional calculations
imcdiag(forward.lm, method="VIF")
imcdiag(backward.lm, method="VIF")
imcdiag(stepwise.lm, method="VIF")
imcdiag(junk.lm, method="VIF")

# Sort values
sort(vif(forward.lm),decreasing=TRUE)
sort(vif(backward.lm),decreasing=TRUE)
sort(vif(stepwise.lm),decreasing=TRUE)
sort(vif(junk.lm),decreasing=TRUE)

# Model comparison, note that rank may vary depending on metric
help(AIC)
help(BIC)

# Forward model metrics from training sample

# Adjusted R squared from summary
summary(forward.lm)

# AIC and BIC
AIC(forward.lm)
BIC(forward.lm)

# MSE and MAE
mse.forward <- mean(forward.lm$residuals^2)
rmse.forward <- sqrt(mse.forward)
mae.forward <- mean(abs(forward.lm$residuals))

# Backward model metrics from training sample

# Adjusted R squared from summary
summary(backward.lm)

# AIC and BIC
AIC(backward.lm)
BIC(backward.lm)

# MSE and MAE
mse.backward <- mean(backward.lm$residuals^2)
rmse.backward <- sqrt(mse.backward)
mae.backward <- mean(abs(backward.lm$residuals))

# Stepwise model metrics from training sample
# Adjusted R squared from summary
summary(stepwise.lm)

# AIC and BIC
AIC(stepwise.lm)
BIC(stepwise.lm)

# MSE and MAE
mse.stepwise <- mean(stepwise.lm$residuals^2)
rmse.stepwise <- sqrt(mse.stepwise)
mae.stepwise <- mean(abs(stepwise.lm$residuals))

# Junk model metrics from training sample

# Adjusted R squared from summary
summary(junk.lm)

# AIC and BIC
AIC(junk.lm)
BIC(junk.lm)

# MSE and MAE
mse.junk <- mean(junk.lm$residuals^2)
rmse.junk <- sqrt(mse.junk)
mae.junk <- mean(abs(junk.lm$residuals))

##############################################################################
# Section 4: Predictive Accuracy
##############################################################################

# Evaluate out of sample performance for all 4 models
forward.test <- predict(forward.lm,newdata=test.df)
backward.test <- predict(backward.lm,newdata=test.df)
stepwise.test <- predict(stepwise.lm,newdata=test.df)
junk.test <- predict(junk.lm,newdata=test.df)

# Compute Mean Squared Error (MSE) and Mean Absolute Error (MAE) for the test sample
# for each model using residuals from lm object

# Forward model
orig_forward <- test.df$SalePrice
predicted_forward <- forward.test
d <- orig_forward - predicted_forward
mse.forwardtest <- mean((d)^2)
mae.forwardtest <- mean(abs(d))
rmse.forwardtest <-  sqrt(mse.forwardtest)
R2.forwardtest <-  1-(sum((d)^2)/sum((orig_forward-mean(orig_forward))^2))
```

```
# Backward model
orig_backward <- test.df$SalePrice
predicted_backward <- backward.test
d <- orig_backward - predicted_backward
mse.backwardtest <- mean((d)^2)
mae.backwardtest <- mean(abs(d))
rmse.backwardtest <-  sqrt(mse.backwardtest)
R2.backwardtest <-  1-(sum((d)^2)/sum((orig_backward-mean(orig_backward))^2))

# Stepwise model
orig_stepwise <- test.df$SalePrice
predicted_stepwise <- stepwise.test
d <- orig_stepwise - predicted_stepwise
mse.stepwisetest <- mean((d)^2)
mae.stepwisetest <- mean(abs(d))
rmse.stepwisetest <-  sqrt(mse.stepwardtest)
R2.stepwisetest <-  1-(sum((d)^2)/sum((orig_stepwise-mean(orig_stepwise))^2))

# Junk model
orig_junk <- test.df$SalePrice
predicted_junk <- junk.test
d <- orig_junk - predicted_junk
mse.junktest <- mean((d)^2)
mae.junktest <- mean(abs(d))
rmse.junktest <-  sqrt(mse.junktest)
R2.junktest <-  1-(sum((d)^2)/sum((orig_junk-mean(orig_junk))^2))

##############################################################################
# Section 5: Operational Validation
##############################################################################

# Validate all 4 models

# Define PredictionGrade variable
# Grade 1: within 10% of actual value
# Grade 2: not Grade 1 but within 15% of actual value
# Grade 3: not Grade 2 but within 25% of actual value
# Grade 4: otherwise

# Training Data
# Absolute Percent Error
forward.pct <-abs(forward.lm$residuals)/train.clean$SalePrice
backward.pct <-abs(backward.lm$residuals)/train.clean$SalePrice
stepwise.pct <-abs(stepwise.lm$residuals)/train.clean$SalePrice
junk.pct <-abs(junk.lm$residuals)/train.clean$SalePrice
```

```
# Test Data
# Absolute Percent Error
forward.testPCT <-abs(test.df$SalePrice-forward.test)/test.df$SalePrice
backward.testPCT <-abs(test.df$SalePrice-backward.test)/test.df$SalePrice
stepwise.testPCT <-abs(test.df$SalePrice-stepwise.test)/test.df$SalePrice
junk.testPCT <-abs(test.df$SalePrice-junk.test)/test.df$SalePrice

# Forward model performance on train and test data

# Assign Prediction Grades - Train
forward.PredictionGrade <-ifelse(forward.pct<=0.10,'Grade 1: [0,0.10]',
                                 ifelse(forward.pct<=0.15,'Grade 2: (0.10,0.15]',
                                        ifelse(forward.pct<=0.25,'Grade 3: (0.15,0.25]',
                                               'Grade 4: (0.25+]')))

# Normalized table (prediction rate/# of observations)
forward.trainTable <-table(forward.PredictionGrade)
forward.trainTable/sum(forward.trainTable)

# Assign Prediction Grades - Test
forward.testPredictionGrade <-ifelse(forward.testPCT<=0.10,'Grade 1: [0.0.10]',
                                     ifelse(forward.testPCT<=0.15,'Grade 2: (0.10,0.15]',
                                            ifelse(forward.testPCT<=0.25,'Grade 3: (0.15,0.25]',
                                                   'Grade 4: (0.25+]')))

# Normalized table (prediction rate/# of observations)
forward.testTable <-table(forward.testPredictionGrade)
forward.testTable/sum(forward.testTable)

# Backward model performance on train and test data

# Assign Prediction Grades - Train
backward.PredictionGrade <-ifelse(backward.pct<=0.10,'Grade 1: [0,0.10]',
                                 ifelse(backward.pct<=0.15,'Grade 2: (0.10,0.15]',
                                        ifelse(backward.pct<=0.25,'Grade 3: (0.15,0.25]',
                                               'Grade 4: (0.25+]')))

# Normalized table (prediction rate/# of observations)
backward.trainTable <-table(backward.PredictionGrade)
backward.trainTable/sum(backward.trainTable)

# Assign Prediction Grades - Test
backward.testPredictionGrade <-ifelse(backward.testPCT<=0.10,'Grade 1: [0.0.10]',
                                     ifelse(backward.testPCT<=0.15,'Grade 2: (0.10,0.15]',
                                            ifelse(backward.testPCT<=0.25,'Grade 3: (0.15,0.25]',
                                                   'Grade 4: (0.25+]')))
```

```
# Normalized table (prediction rate/# of observations)
backward.testTable <-table(backward.testPredictionGrade)
backward.testTable/sum(backward.testTable)

# Stepwise model performance on train and test data

# Assign Prediction Grades - Train
stepwise.PredictionGrade <-ifelse(stepwise.pct<=0.10,'Grade 1: [0,0.10]',
                                  ifelse(stepwise.pct<=0.15,'Grade 2: (0.10,0.15]',
                                         ifelse(stepwise.pct<=0.25,'Grade 3: (0.15,0.25]',
                                                'Grade 4: (0.25+]')))

# Normalized table (prediction rate/# of observations)
stepwise.trainTable <-table(stepwise.PredictionGrade)
stepwise.trainTable/sum(stepwise.trainTable)

# Assign Prediction Grades - Test
stepwise.testPredictionGrade <-ifelse(stepwise.testPCT<=0.10,'Grade 1: [0.0.10]',
                                      ifelse(stepwise.testPCT<=0.15,'Grade 2: (0.10,0.15]',
                                             ifelse(stepwise.testPCT<=0.25,'Grade 3: (0.15,0.25]',
                                                    'Grade 4: (0.25+]')))

# Normalized table (prediction rate/# of observations)
stepwise.testTable <-table(stepwise.testPredictionGrade)
stepwise.testTable/sum(stepwise.testTable)

# Junk model performance on train and test data

# Assign Prediction Grades - Train
junk.PredictionGrade <-ifelse(junk.pct<=0.10,'Grade 1: [0,0.10]',
                                  ifelse(junk.pct<=0.15,'Grade 2: (0.10,0.15]',
                                         ifelse(junk.pct<=0.25,'Grade 3: (0.15,0.25]',
                                                'Grade 4: (0.25+]')))

# Normalized table (prediction rate/# of observations)
junk.trainTable <-table(junk.PredictionGrade)
junk.trainTable/sum(junk.trainTable)

# Assign Prediction Grades - Test
junk.testPredictionGrade <-ifelse(junk.testPCT<=0.10,'Grade 1: [0.0.10]',
                                      ifelse(junk.testPCT<=0.15,'Grade 2: (0.10,0.15]',
                                             ifelse(junk.testPCT<=0.25,'Grade 3: (0.15,0.25]',
                                                    'Grade 4: (0.25+]')))

# Normalized table (prediction rate/# of observations)
junk.testTable <-table(junk.testPredictionGrade)
junk.testTable/sum(junk.testTable)
```