Assignment #5
Jennifer M. Wanat

Introduction:

In this report our objective is to be able to provide estimates of home values for the typical home in Ames, Iowa. The data set was obtained from the Ames, Iowa Assessor's Office and assembled by Dr. Dean De Cock at Truman State University. In order to build a model to provide estimates of home values, an exploratory data analysis (EDA) of the data set was completed prior to this analysis, which allowed for the selection of the most promising predictor variables.

The data documentation was reviewed to understand the types of variables collected. As the goal is to predict the typical home value, some conditions were dropped so that the data set represented the typical single-family home. The data set was randomly split into a training set and test set to allow for cross-validation of the predictive models.

Automated variable selection techniques utilized the training data set for model identification. The generated model summaries and parameters were provided, and assessed for predictive accuracy by examining model selection criteria. The predictive accuracy of each model was assessed with the test data set. Finally, all models were assigned prediction grades for the training and test data sets. The analysis was conducted with the R programming language.

Data:

The data set contains 82 variables measured from 2930 individual residential properties sold in Ames, IA from 2006 to 2010. Refer to the data documentation for description of the variables.

Section 1: Sample Definition and Data Split

Section 1.1: Sample Definition

From the data set, the conditions listed in Table 1 were dropped and not used for the sample population. This process of elimination is referenced as waterfall conditions, as the first condition dropped will result in a smaller sample size in which the subsequent condition to be dropped would be applied. This process continues until all waterfall conditions have been processed.

The waterfall conditions were selected to create a data set that represented typical single-family, detached homes in Ames, IA. The data documentation indicated that there were 5 observations from the original data set that were either outliers or unusual sales. The drop condition for observations greater than 4,000 square feet dropped these observations from the data set, if the prior waterfall conditions had not already done so.

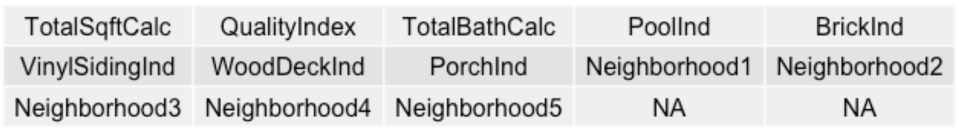
Table 1: Waterfall conditions.

Variable	Drop Condition	Number of Properties Dropped	Sample Population
			2930
Building Type	Not equal to single-family detached	505	2425
Sale Condition	Not equal to normal	423	2002
Street	Not paved	6	1996
Year Built	Built Pre-1950	489	1507
Above Grade Living Area	Less than 800 square feet	9	1498
Above Grade Living Area	Greater than 4,000 square feet	1	1497
Lot Area	Greater than 100,000 square feet	3	1494
Bedroom	No bedrooms	4	1490

The resulting sample population data set (a data frame called eligible.population) contained 82 variables from 1490 individual residential properties.

Additional predictor variables that may add accuracy to a predictive model were added to the sample population data set. With the exception of TotalSqftCalc, QualityIndex, and TotalBathCalc, the variables added were indicator variables used to designate the presence or absence of a factor variable, such as brick or vinyl siding. TotalSqftCalc combined the square footage of finished basement area and above grade living area. QualityIndex was a number calculated from the product of the overall quality and overall condition rank of the home. The TotalBathCalc added the number of full and half baths found in the basement and above grade living area.

Table 2: Additional predictor variables added to data set. NA indicates null value.



TotalSqftCalc	QualityIndex	TotalBathCalc	PoolInd	BrickInd
VinylSidingInd	WoodDeckInd	PorchInd	Neighborhood1	Neighborhood2
Neighborhood3	Neighborhood4	Neighborhood5	NA	NA

Section 1.2: The Train/Test Split

In order to assess model performance, the sample population data set was split into a 70/30 train/test split. This would allow for cross-validation after a predictive model has been developed. The train data set would be used for in-sample model development and the test data set would be used for out-of-sample model assessment of predictive accuracy.

Table 3: Observation counts and percentage of train and test data sets.

	Number of Observations	Percentage
Train Set	1046	70.2
Test Set	444	29.8
Total	1490	100

Section 2: Model Identification and In-Sample Model Fit

A pool of 19 candidate predictor variables was selected for modeling purposes. The predictor variables were a combination of discrete and continuous variables, and were saved into a data frame called train.clean1.

Table 4: Initial selection of candidate predictor variables.
SalePrice is the Response Variable.

train.clean1 data frame	LotArea	LandContour	YearBuilt	Electrical
	GrLivArea	BedroomAbvGr	TotRmsAbvGrd	GarageType
	GarageYrBlt	GarageArea	SalePrice	TotalSqftCalc
	QualityIndex	TotalBathCalc	PoolInd	BrickInd
	VinylSidingInd	PorchInd	Neighborhood4	Neighborhood5

After model identification was completed and assessed, it was determined that four variables were poor predictors for sale price modeling. As a result, GarageType, Electrical, PorchInd, and TotRmsAbvGrd were dropped for modeling purposes. The reduced pool of variables was saved into a data frame called train.clean. Table 5 represents the variables used for prediction modeling.

Table 5: Final selection of candidate predictor variables.
SalePrice is the Response Variable.

train.clean data frame	LotArea	LandContour	YearBuilt	GrLivArea
	BedroomAbvGr	GarageYrBlt	GarageArea	SalePrice
	TotalSqftCalc	QualityIndex	TotalBathCalc	PoolInd
	BrickInd	VinylSidingInd	Neighborhood4	Neighborhood5

Utilizing the training data set, three automated variable selection techniques were used to select predictor variables to generate the best models. The three techniques used were forward selection, backward elimination, and stepwise method. Models were assessed with the metric Akaike Information Criterion (AIC), which will penalize model complexity with too many predictor variables. Small values of AIC are preferred. The model with the lowest AIC value generated from each variable selection technique was then assessed for model fit.

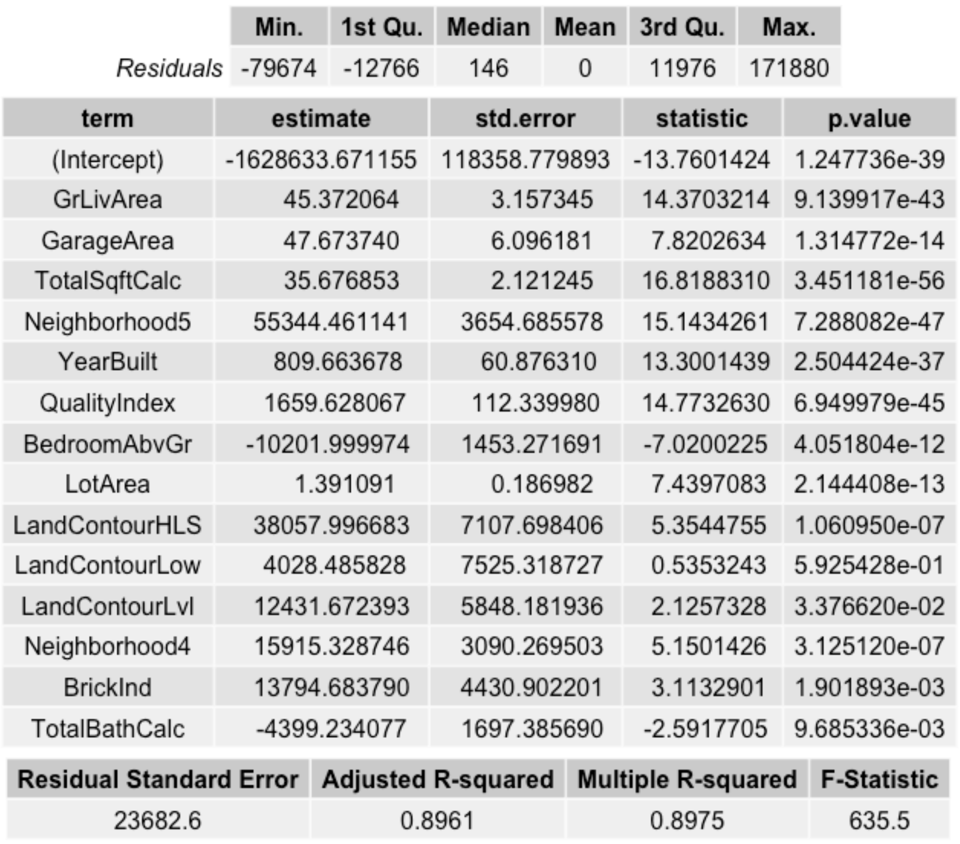
Section 2.1: Forward Variable Selection

Forward selection begins with linear regression containing only an intercept. Predictor variables are added until a model with the lowest AIC has been achieved. The following model was generated with forward selection:

Forward Selection Model: $E(\text{Sale Price}) = b_0 + b_1\text{GrLivArea} + b_2\text{GarageArea} + b_3\text{TotalSqftCalc} + b_4\text{Neighborhood5} + b_5\text{YearBuilt} + b_6\text{QualityIndex} + b_7\text{BedroomAbvGr} + b_8\text{LotArea} + b_9\text{LandContourHLS} + b_{10}\text{LandContourLow} + b_{11}\text{LandContourLvl} + b_{12}\text{Neighborhood4} + b_{13}\text{BrickInd} + b_{14}\text{TotalBathCalc}$

$$E(\text{Sale Price}) = -1628633.671 + + 45.372\text{GrLivArea} + 47.674\text{GarageArea} + 35.677\text{TotalSqftCalc} + 55344.461\text{Neighborhood5} + 809.664\text{YearBuilt} + 1659.628\text{QualityIndex} - 10202.000\text{BedroomAbvGr} + 1.391\text{LotArea} + 38057.997\text{LandContourHLS} + 4028.486\text{LandContourLow} + 12431.672\text{LandContourLvl} + 15915.329\text{Neighborhood4} + 13794.684\text{BrickInd} - 4399.234\text{TotalBathCalc}$$

Table 6: Forward Selection Model.



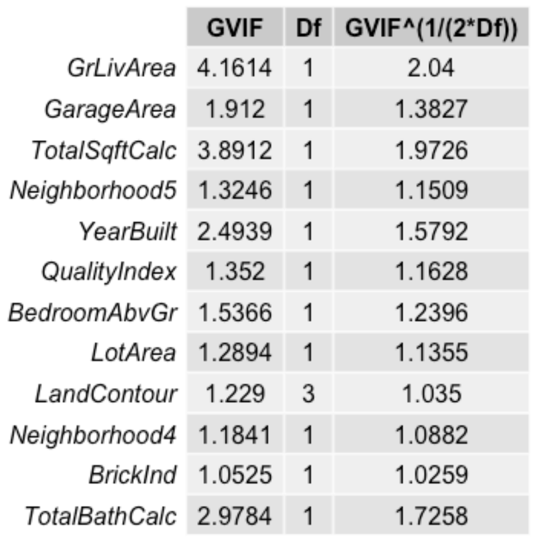
	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Residuals	-79674	-12766	146	0	11976	171880

term	estimate	std.error	statistic	p.value
(Intercept)	-1628633.671155	118358.779893	-13.7601424	1.247736e-39
GrLivArea	45.372064	3.157345	14.3703214	9.139917e-43
GarageArea	47.673740	6.096181	7.8202634	1.314772e-14
TotalSqftCalc	35.676853	2.121245	16.8188310	3.451181e-56
Neighborhood5	55344.461141	3654.685578	15.1434261	7.288082e-47
YearBuilt	809.663678	60.876310	13.3001439	2.504424e-37
QualityIndex	1659.628067	112.339980	14.7732630	6.949979e-45
BedroomAbvGr	-10201.999974	1453.271691	-7.0200225	4.051804e-12
LotArea	1.391091	0.186982	7.4397083	2.144408e-13
LandContourHLS	38057.996683	7107.698406	5.3544755	1.060950e-07
LandContourLow	4028.485828	7525.318727	0.5353243	5.925428e-01
LandContourLvl	12431.672393	5848.181936	2.1257328	3.376620e-02
Neighborhood4	15915.328746	3090.269503	5.1501426	3.125120e-07
BrickInd	13794.683790	4430.902201	3.1132901	1.901893e-03
TotalBathCalc	-4399.234077	1697.385690	-2.5917705	9.685336e-03

Residual Standard Error	Adjusted R-squared	Multiple R-squared	F-Statistic
23682.6	0.8961	0.8975	635.5

The variance inflation factor (VIF) for each predictor variable was calculated. If any of the predictors have more than one degree of freedom (Df), then the generalized variance inflation factor (GVIF) is calculated. The VIF or GVIF is a measurement of collinearity between predictors. Small values of VIF/GVIF are preferred. There is no value of GVIF of concern, although it is noted that GrLivArea and TotalSqftCalc reflect some collinearity. This is expected as GrLivArea is a component of TotalSqftCalc.

Table 7: Forward Selection Model VIF



	GVIF	Df	GVIF^(1/(2*Df))
GrLivArea	4.1614	1	2.04
GarageArea	1.912	1	1.3827
TotalSqftCalc	3.8912	1	1.9726
Neighborhood5	1.3246	1	1.1509
YearBuilt	2.4939	1	1.5792
QualityIndex	1.352	1	1.1628
BedroomAbvGr	1.5366	1	1.2396
LotArea	1.2894	1	1.1355
LandContour	1.229	3	1.035
Neighborhood4	1.1841	1	1.0882
BrickInd	1.0525	1	1.0259
TotalBathCalc	2.9784	1	1.7258

Section 2.2: Backward Variable Selection

Backward elimination begins with linear regression containing all predictors. Predictor variables are removed until a model with the lowest AIC has been achieved. The following model was generated with backward elimination:

Backward Elimination Model: E(Sale Price) = b_0 + b_1LotArea + b_2LandContourHLS + b_3LandContourLow + b_4LandContourLvl + b_5YearBuilt +b_6GrLivArea + b_7BedroomAbvGr + b_8GarageArea + b_9TotalSqftCalc + b_{10}QualityIndex + b_{11}TotalBathCalc + b_{12}BrickInd + b_{13}Neighborhood4 + b_{14}Neighborhood5

E(Sale Price) = -1628633.671 + 1.391LotArea + 38057.997LandContourHLS + 4028.486LandContourLow + 12431.672LandContourLvl + 809.664YearBuilt + 45.372GrLivArea + -10202.000BedroomAbvGr + 47.674GarageArea + 35.677TotalSqftCalc + 1659.628QualityIndex - 4399.234TotalBathCalc + 13794.684BrickInd + 15915.329Neighborhood4 + 55344.461Neighborhood5

Table 8: Backward Elimination Model.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Residuals	-79674	-12766	146	0	11976	171880

term	estimate	std.error	statistic	p.value
(Intercept)	-1628633.671155	118358.779893	-13.7601424	1.247736e-39
LotArea	1.391091	0.186982	7.4397083	2.144408e-13
LandContourHLS	38057.996683	7107.698406	5.3544755	1.060950e-07
LandContourLow	4028.485828	7525.318727	0.5353243	5.925428e-01
LandContourLvl	12431.672393	5848.181936	2.1257328	3.376620e-02
YearBuilt	809.663678	60.876310	13.3001439	2.504424e-37
GrLivArea	45.372064	3.157345	14.3703214	9.139917e-43
BedroomAbvGr	-10201.999974	1453.271691	-7.0200225	4.051804e-12
GarageArea	47.673740	6.096181	7.8202634	1.314772e-14
TotalSqftCalc	35.676853	2.121245	16.8188310	3.451181e-56
QualityIndex	1659.628067	112.339980	14.7732630	6.949979e-45
TotalBathCalc	-4399.234077	1697.385690	-2.5917705	9.685336e-03
BrickInd	13794.683790	4430.902201	3.1132901	1.901893e-03
Neighborhood4	15915.328746	3090.269503	5.1501426	3.125120e-07
Neighborhood5	55344.461141	3654.685578	15.1434261	7.288082e-47

Residual Standard Error	Adjusted R-squared	Multiple R-squared	F-Statistic
23682.6	0.8961	0.8975	635.5

The VIF or GVIF for each predictor variable was calculated. The VIF or GVIF is a measurement of collinearity between predictors. Small values of VIF/GVIF are preferred. There is no value of GVIF of concern, although it is noted that GrLivArea and TotalSqftCalc reflect some collinearity. This is expected as GrLivArea is a component of TotalSqftCalc.

Table 9: Backward Elimination Model VIF.

	GVIF	Df	GVIF^(1/(2*Df))
LotArea	1.2894	1	1.1355
LandContour	1.229	3	1.035
YearBuilt	2.4939	1	1.5792
GrLivArea	4.1614	1	2.04
BedroomAbvGr	1.5366	1	1.2396
GarageArea	1.912	1	1.3827
TotalSqftCalc	3.8912	1	1.9726
QualityIndex	1.352	1	1.1628
TotalBathCalc	2.9784	1	1.7258
BrickInd	1.0525	1	1.0259
Neighborhood4	1.1841	1	1.0882
Neighborhood5	1.3246	1	1.1509

Section 2.3: Stepwise Variable Selection

Stepwise method selection begins with single linear regression containing the TotalSqftCalc predictor. Predictor variables are removed or added until a model with the lowest AIC has been achieved. The following model was generated with the stepwise method:

Stepwise Method Model: E(Sale Price) = b_0 + b_1TotalSqftCalc+ b_2YearBuilt + b_3Neighborhood5 + b_4QualityIndex +b_5GrLivArea + b_6GarageArea + b_7BedroomAbvGr + b_8LotArea + b_9LandContourHLS + b_{10}LandContourLow + b_{11}LandContourLvl + b_{12}Neighborhood4 + b_{13}BrickInd + b_{14}TotalBathCalc

E(Sale Price) = -1628633.671 + 35.677TotalSqftCalc + 809.664YearBuilt + 55344.461 Neighborhood5 + 1659.628QualityIndex + 45.372GrLivArea + 47.674GarageArea - 10202.000 BedroomAbvGr + 1.391LotArea + 38057.997LandContourHLS + 4028.486LandContourLow + 12431.672LandContourLvl + 15915.329Neighborhood4 + 13794.684BrickInd - 4399.234TotalBathCalc

Table 10: Stepwise Method Model.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Residuals	-79674	-12766	146	0	11976	171880

term	estimate	std.error	statistic	p.value
(Intercept)	-1628633.671155	118358.779893	-13.7601424	1.247736e-39
TotalSqftCalc	35.676853	2.121245	16.8188310	3.451181e-56
YearBuilt	809.663678	60.876310	13.3001439	2.504424e-37
Neighborhood5	55344.461141	3654.685578	15.1434261	7.288082e-47
QualityIndex	1659.628067	112.339980	14.7732630	6.949979e-45
GrLivArea	45.372064	3.157345	14.3703214	9.139917e-43
GarageArea	47.673740	6.096181	7.8202634	1.314772e-14
BedroomAbvGr	-10201.999974	1453.271691	-7.0200225	4.051804e-12
LotArea	1.391091	0.186982	7.4397083	2.144408e-13
LandContourHLS	38057.996683	7107.698406	5.3544755	1.060950e-07
LandContourLow	4028.485828	7525.318727	0.5353243	5.925428e-01
LandContourLvl	12431.672393	5848.181936	2.1257328	3.376620e-02
Neighborhood4	15915.328746	3090.269503	5.1501426	3.125120e-07
BrickInd	13794.683790	4430.902201	3.1132901	1.901893e-03
TotalBathCalc	-4399.234077	1697.385690	-2.5917705	9.685336e-03

Residual Standard Error	Adjusted R-squared	Multiple R-squared	F-Statistic
23682.6	0.8961	0.8975	635.5

The VIF or GVIF for each predictor variable was calculated. The VIF or GVIF is a measurement of collinearity between predictors. Small values of VIF/GVIF are preferred. There is no value of GVIF of concern, although it is noted that GrLivArea and TotalSqftCalc reflect some collinearity. This is expected as GrLivArea is a component of TotalSqftCalc.

Table 11: Stepwise Method Model VIF.

	GVIF	Df	GVIF^(1/(2*Df))
TotalSqftCalc	3.8912	1	1.9726
YearBuilt	2.4939	1	1.5792
Neighborhood5	1.3246	1	1.1509
QualityIndex	1.352	1	1.1628
GrLivArea	4.1614	1	2.04
GarageArea	1.912	1	1.3827
BedroomAbvGr	1.5366	1	1.2396
LotArea	1.2894	1	1.1355
LandContour	1.229	3	1.035
Neighborhood4	1.1841	1	1.0882
BrickInd	1.0525	1	1.0259
TotalBathCalc	2.9784	1	1.7258

Section 2.4: Model Comparison

For model comparison purposes, a "junk" model was prepared. This model used the four predictor variables of OverallQual, OverallCond, QualityIndex, GrLivArea, and TotalSqftCalc. The QualityIndex is the product of OverallQual and OverallCond variables, and GrLivArea is a component of TotalSqftCalc. This model will reflect collinearity, which was observed in the VIF values. The following model is the junk model:

$$\text{Junk Model: E(Sale Price)} = b_0 + b_1\text{OverallQual} + b_2\text{OverallCond} + b_3\text{QualityIndex} + b_4\text{GrLivArea} + b_5\text{TotalSqftCalc}$$

$$\text{E(Sale Price)} = -\ 270418.162 + 54009.109\text{OverallQual} + 30492.142\text{OverallCond} - 4987.280\text{QualityIndex} + 31.316\text{GrLivArea} + 38.58\text{TotalSqftCalc}$$

Table 12: Junk Model Summary.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Residuals	-95006	-17093	-925	0	14094	182497

term	estimate	std.error	statistic	p.value
(Intercept)	-270418.16192	27050.221097	-9.996893	1.573196e-22
OverallQual	54009.10897	4562.259691	11.838236	2.014303e-30
OverallCond	30492.14223	5026.932340	6.065755	1.836222e-09
QualityIndex	-4987.27981	866.195742	-5.757682	1.121786e-08
GrLivArea	31.31552	3.236817	9.674789	2.939729e-21
TotalSqftCalc	38.58337	2.025012	19.053398	1.169214e-69

Residual Standard Error	Adjusted R-squared	Multiple R-squared	F-Statistic
28749.3	0.8473	0.848	1160.4

The VIF for each predictor variable was calculated. There were high values of VIF (≥ 29) for the OverallQual, OverallCond, and QualityIndex variables, and this is a result of QualityIndex being equal to the product of OverallQual and OverallCond.   It is noted that GrLivArea and TotalSqftCalc reflect some collinearity. This is expected as GrLivArea is a component of TotalSqftCalc, but the VIF values are considered low.

Table 13: Junk Model VIF.

	VIF
OverallQual	44.49179
OverallCond	29.30509
QualityIndex	56.22512
GrLivArea	3.08998
TotalSqftCalc	2.46313

For each of the models, the adjusted R-squared, AIC, Bayesian Information Criterion (BIC), mean standard error (MSE) and mean absolute error (MAE) were calculated. These metrics are used to evaluate the model fit with the training set data. A high value of adjusted R-squared and a small value of AIC, BIC, MSE and MAE is desired. All three models determined from the forward selection, backward elimination and stepwise method had the same values for each criterion. The junk model had a lower adjusted R-squared value and higher AIC, BIC, MSE and MAE values, which reflects a less accurate model.

Table 14: Comparison of Model In-Sample Fit.

	Adjusted R-squared	AIC	BIC	MSE	MAE
Forward	0.8961	23712	23791	552705678	16664
Backward	0.8961	23712	23791	552705678	16664
Stepwise	0.8961	23712	23791	552705678	16664
Junk	0.8473	24454	24488	821784043	20562

The model metrics were compared against each other and ranked. As the forward selection, backward elimination and stepwise method had the same values for each criterion, all three models were tied for first. The junk model was ranked second.

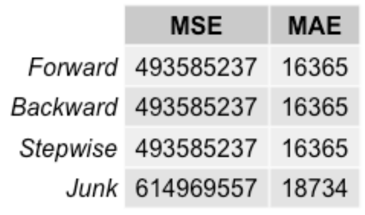
Table 15: Rank of Model In-Sample Fit.

	Adjusted R-squared	AIC	BIC	MSE	MAE
First	Model 1, 2, and 3	Model 1, 2, and 3	Model 1, 2, and 3	Model 1, 2, and 3	Model 1, 2, and 3
Second	Junk	Junk	Junk	Junk	Junk

Section 3: Predictive Accuracy

The predictive accuracy was assessed with the out-of-sample test data set by calculating the MSE and MAE for each model. Lower values of MSE and MAE are desired, and the model criterions were compared against each other.

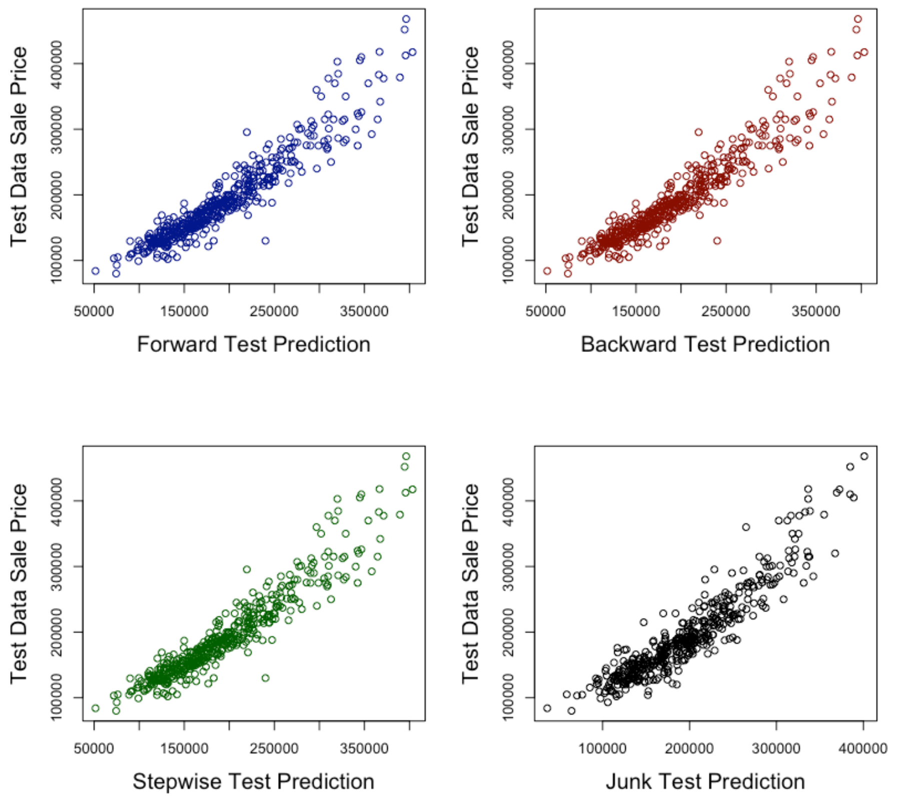
Table 16: Predictive Accuracy of Each Model with the Out-of-Sample Data Set.



	MSE	MAE
Forward	493585237	16365
Backward	493585237	16365
Stepwise	493585237	16365
Junk	614969557	18734

All models had the same MSE and MAE values with the exception of the junk model. The junk model had higher values of MSE and MAE, which indicate less predictive accuracy. The MSE and MAE values of the test data set for the forward, backward, and stepwise models were lower than those calculated from the train data set. This indicates that the models are not overfit to the train data set.

Figure 1: Scatterplot of sale price versus predicted sale price for each model.



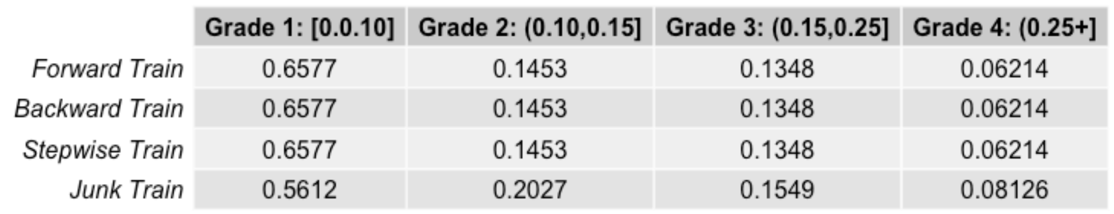
Scatterplots of sale price versus the predicted sale price with the out-of-sample (test) data set were prepared for each model. Visually, predictive performance was similar between all models, with a slight heteroscedastic variance observed as the sale price increased.

Section 4: Operational Validation

Prediction grades for the in-sample (train) data set and out-of-sample (test) data set were calculated. Grades were determined by the predictive accuracy of each model. A grade of 1 is ideal, followed by grade 2, grade 3 and grade 4.

The forward selection, backward elimination and stepwise method generated the same predictive accuracy with the in-sample (train) data set. All three models were accurate to within ten percent of sale price 65.77% of the time. The three models were accurate with 10-15 percent of the sale price 14.53% of the time, 15-25 percent of the sale price 13.48% of the time, and greater than 25 percent of the sale price 6.214% of the time. The junk model performed worse with a predictive accuracy within ten percent of sale price 56.12% of the time.

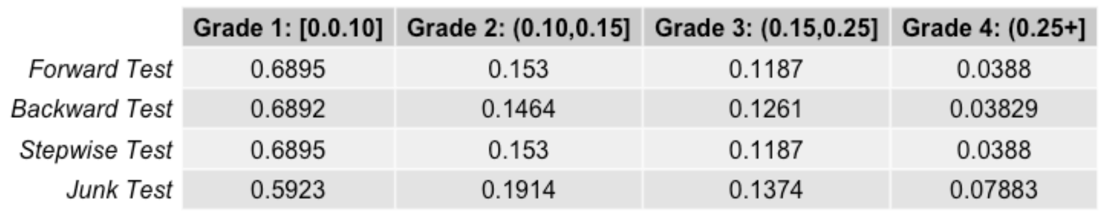
Table 17: Prediction Grades of In-Sample Data Set by Model.



	Grade 1: [0.0.10]	Grade 2: (0.10,0.15]	Grade 3: (0.15,0.25]	Grade 4: (0.25+]
Forward Train	0.6577	0.1453	0.1348	0.06214
Backward Train	0.6577	0.1453	0.1348	0.06214
Stepwise Train	0.6577	0.1453	0.1348	0.06214
Junk Train	0.5612	0.2027	0.1549	0.08126

The forward selection and stepwise method generated the same predictive accuracy with the out-of-sample (test) data set and were approximately three percent more accurate than the in-sample data set. The two models were accurate to within ten percent of sale price 68.95% of the time. The two models were accurate with 10-15 percent of the sale price 15.3% of the time, 15-25 percent of the sale price 11.87% of the time, and greater than 25 percent of the sale price 3.88% of the time. The backward elimination performance was almost identical, but just slightly lower than the forward and stepwise models. The backward elimination model was accurate to within ten percent of sale price 68.92% of the time, within 10-15 percent of the sale price 14.64% of the time, within 15-25 percent of the sale price 12.61% of the time, and greater than 25 percent of the sale price 3.829% of the time. The junk model performed worse with a predictive accuracy within ten percent of sale price 59.23% of the time. All models had better prediction accuracy with the out-of-sample data set when compared to the in-sample data set.

Table 18: Prediction Grades of Out-of-Sample Data Set by Model.



	Grade 1: [0.0.10]	Grade 2: (0.10,0.15]	Grade 3: (0.15,0.25]	Grade 4: (0.25+]
Forward Test	0.6895	0.153	0.1187	0.0388
Backward Test	0.6892	0.1464	0.1261	0.03829
Stepwise Test	0.6895	0.153	0.1187	0.0388
Junk Test	0.5923	0.1914	0.1374	0.07883

Conclusion:

The forward selection and stepwise method models performed the best towards predicting sale price with the Ames Housing data set. Both models had an adjusted R-squared of 0.8961, and the lowest values of AIC, BIC, MSE, and MAE. The best prediction grades were achieved with the forward selection and stepwise method models for both the in-sample and out-of-sample data sets.

References:

Black, K. (2017). Business Statistics for Contemporary Decision Making, Ninth Edition. Hoboken, NJ: John Wiley & Sons, Inc.

De Cock, D. (2011). Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project, Journal of Statistics Education, 19:3. Retrieved from http://amstat.tandfonline.com/doi/abs/10.1080/10691898.2011.11889627#.WciC-WinFTE

De Cock, D. Ames Housing Data Documentation. Retrieved from https://ww2.amstat.org/publications/jse/v19n3/decock/DataDocumentation.txt

Fox, J. and Weisberg, S. (2011). An R Companion to Applied Regression, Second Edition. Thousand Oaks, CA: Sage Publications, Inc.

Lander, J. (2014). R for Everyone. Upper Saddle River, NJ: Addison-Wesley.

Pardoe, I. (2012). Applied Regression Modeling, Second Edition. Hoboken, NJ: John Wiley & Sons, Inc.

Stowell, S. (2014). Using R for Statistics. New York, NY: Apress.

Weisberg, S. (2014). Applied Linear Regression, Fourth Edition. Hoboken, NJ: John Wiley & Sons, Inc.

Code:

```
# Jennifer Wanat
# Fall 2017
# Ames_assignment4.R


install.packages("maptools")
install.packages("stargazer")

require(maptools)
require(gridExtra)
require(gtable)
require(sjPlot)
require(broom)

path.name <- "~/Desktop/R/"
file.name <- paste(path.name, "ames_housing_data.csv", sep = "")

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

# Show the header of the data frame;
head(ames.df)

# Show the structure of the data frame;
str(ames.df)

######################################Waterfall###########################
#Creating a waterfall of drop conditions
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$YearBuilt <1950,'04: Built Pre-1950',
                       ifelse(ames.df$GrLivArea <800,'05: Less than 800 SqFt',
                         ifelse(ames.df$GrLivArea >4000,'06: Greater than 4000
SqFt',
                           ifelse(ames.df$LotArea >100000,'07: Lot 100000
SqFt',
                             ifelse(ames.df$BedroomAbvGr <1, '08: No
Bedrooms',
                               '99: Eligible Sample')
                     )))))))

table(ames.df$dropCondition)
```

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

# Format the table as a column matrix for presentation;
as.matrix(waterfall,8,1)

# 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)

#check the structure of the data frame
str(eligible.population)

#Table of waterfall variables and counts, but not used as it did not reflect the adjusted
population count
test <- as.data.frame(t(waterfall))
str(test)
grid.table(test[c(-1)], cols = c("Variable", "Number"), rows = NULL)
#this is an alternate table of above
#same table output but left alignment
tt3 <- ttheme_default(core=list(fg_params=list(hjust=0, x=0.05)),
                      rowhead=list(fg_params=list(hjust=0, x=0)))
g <- grid.table(test[c(-1)], cols = c("Variable", "Number"), rows = NULL,
          theme = tt3)



###################################Create additional predictor variables#####

# Define total square footage
eligible.population$TotalSqftCalc <-
eligible.population$BsmtFinSF1+eligible.population$BsmtFinSF2+eligible.population$GrL
ivArea;

# Define Quality Index
eligible.population$QualityIndex <-
eligible.population$OverallQual*eligible.population$OverallCond;
str(eligible.population$QualityIndex)

# Define total bathrooms
eligible.population$TotalBathCalc <- eligible.population$BsmtFullBath +
0.5*eligible.population$BsmtHalfBath +
  eligible.population$FullBath + 0.5*eligible.population$HalfBath;
```

```
# Define Pool Indicator
eligible.population$PoolInd <- ifelse(eligible.population$PoolArea>0,1,0);

# Define Exterior Siding Type
eligible.population$BrickInd <- ifelse(eligible.population$Exterior1=='BrkFace',1,0);
eligible.population$VinylSidingInd <- ifelse(eligible.population$Exterior1=='VinylSd',1,0);

# Define Wood Deck Indicator
eligible.population$WoodDeckInd <- ifelse(eligible.population$WoodDeckSF>0,1,0);

# Define Porch Indicator - Open Porch OR Screen Porch
eligible.population$PorchInd <-
ifelse((eligible.population$OpenPorchSF>0)|(eligible.population$ScreenPorch>0),1,0);

# Define Neighborhoods by price per square foot
eligible.population$Neighborhood1 <- ifelse((eligible.population$Neighborhood ==
c("IDOTRR")|
                              (eligible.population$Neighborhood == c("OldTown")|
                                (eligible.population$Neighborhood == c("SWISU")))), 1,0);

eligible.population$Neighborhood2 <- ifelse((eligible.population$Neighborhood ==
c("BrkSide")|
                              (eligible.population$Neighborhood == c("Edwards"))), 1,0);

eligible.population$Neighborhood3 <- ifelse((eligible.population$Neighborhood ==
c("NWAmes")|
                              (eligible.population$Neighborhood == c("Crawfor")|
                                (eligible.population$Neighborhood == c("Gilbert")|
                                  (eligible.population$Neighborhood == c("NAmes")|
                                    (eligible.population$Neighborhood == c("Sawyer")|
                                      (eligible.population$Neighborhood == c("SawyerW")|
                                        (eligible.population$Neighborhood == c("Blmngtn")|
                                          (eligible.population$Neighborhood == c("ClearCr")|
                                            (eligible.population$Neighborhood ==
c("NoRidge")|
                                              (eligible.population$Neighborhood ==
c("Veenker")|
                                                (eligible.population$Neighborhood ==
c("CollgCr")|
                                                  (eligible.population$Neighborhood ==
c("Mitchel")
                                                  )))))))))))), 1,0);


eligible.population$Neighborhood4 <- ifelse((eligible.population$Neighborhood ==
c("Timber")|
                                  (eligible.population$Neighborhood == c("Somerst"))), 1,0);

eligible.population$Neighborhood5 <- ifelse((eligible.population$Neighborhood ==
c("StoneBr")|
                                  (eligible.population$Neighborhood == c("NridgHt"))), 1,0);


#######################################################
#Table of variable parameters added to eligible population data frame
#colnames(eligible.population)[84:97]
indicator1 <- t(colnames(eligible.population)[84:88])
indicator2 <- t(colnames(eligible.population)[89:93])
indicator3 <- t(colnames(eligible.population)[94:98])
indicator.table <- rbind(indicator1, indicator2, indicator3)
grid.table(indicator.table)


######################Save as .RData object##########
# Save the R data frame as an .RData object
saveRDS(eligible.population,file='/Users/jmwanat/Documents/Northwestern classes/MSPA
410/410 R/ames_assignment3.RData')


###################Random Number seed generator and Train / Test split########
# Set the seed on the random number generator so you get the same split every time that
# you run the code.
set.seed(123)
eligible.population$u <- runif(n=dim(eligible.population)[1],min=0,max=1);

# Create train/test split;
train.df <- subset(eligible.population, u<0.70);
test.df <- subset(eligible.population, u>=0.70);

# Check your data split. The sum of the parts should equal the whole.
# Do your totals add up?
dim(eligible.population)[1]
dim(train.df)[1]
dim(test.df)[1]
dim(train.df)[1]+dim(test.df)[1]

#Create table of observations in each data frame and calculate percentage split
total <- dim(eligible.population)[1]
total.train <- dim(train.df)[1]
total.test <- dim(test.df)[1]

percent.total <- round((total/total)*100, 1)
percent.train <- round((total.train/total)*100, 1)
percent.test <- signif((total.test/total)*100, 4)

total.all <- c(total.train, total.test, total)
percent.all <- c(percent.train, percent.test, percent.total)

overview.split <- cbind(total.all, percent.all)
colnames(overview.split) <- c("Number\nof Observations", "Percentage")
rownames(overview.split) <- c("Train Set", "Test Set", "Total")
grid.table(overview.split)


######################Model Identification by Automated Variable
Selection##############
#Initial drop listed generated but not used after evaluation of model output
#kept for documentation purposes

drop.list1 <- c('SID', 'PID', 'SubClass', 'Zoning','LotFrontage', 'Street', 'Alley', 'LotShape',
'Utilities', 'LotConfig',
          'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
'OverallQual', 'OverallCond',
          'YearRemodel', 'RoofStyle', 'RoofMat', 'Exterior1', 'Exterior2', 'MasVnrType',
'MasVnrArea', 'ExterQual',
          'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure',
'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
          'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'HeatingQC', 'CentralAir',
'FirstFlrSF', 'SecondFlrSF', 'LowQualFinSF',
          'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'KitchenAbvGr',
'KitchenQual', 'Functional', 'Fireplaces',
          'FireplaceQu', 'GarageTyBlt', 'GarageFinish', 'GarageCars', 'GarageQual',
'GarageCond', 'PavedDrive', 'WoodDeckSF',
          'OpenPorchSF', 'EnclosedPorch', 'ThreeSsnPorch', 'ScreenPorch', 'PoolArea',
'PoolQC', 'Fence', 'MiscFeature',
          'MiscVal', 'MoSold', 'YrSold', 'SaleType', 'SaleCondition', 'dropCondition',
'WoodDeckInd', 'Neighborhood1',
          'Neighborhood2', 'Neighborhood3', 'Heating', 'u');

train.clean1 <-train.df[,!(names(eligible.population) %in% drop.list1)];

#Table of variables used in train.clean1 data frame
r1 <- t(colnames(train.clean1)[1:4])
r2 <- t(colnames(train.clean1)[5:8])
r3 <- t(colnames(train.clean1)[9:12])
r4 <- t(colnames(train.clean1)[13:16])
r5 <- t(colnames(train.clean1)[17:20])
train.clean1.table <- rbind(r1,r2,r3,r4,r5)
```

```
rownames(train.clean1.table) <- c('train.clean1 data frame', ' ', ' ', ' ', ' ')
grid.table((train.clean1.table))


################################Alternate drop list for Automated Variable
Selection##########
#This was the drop list used for all analysis in the assignment

drop.list <- c('SID', 'PID', 'SubClass', 'Zoning','LotFrontage', 'Street', 'Alley', 'LotShape',
'Utilities', 'LotConfig',
            'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
'HouseStyle', 'OverallQual', 'OverallCond',
            'YearRemodel', 'RoofStyle', 'RoofMat', 'Exterior1', 'Exterior2', 'MasVnrType',
'MasVnrArea', 'ExterQual',
            'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure',
'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
            'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'HeatingQC', 'CentralAir',
'FirstFlrSF', 'SecondFlrSF', 'LowQualFinSF',
            'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'KitchenAbvGr',
'KitchenQual', 'Functional', 'Fireplaces',
            'FireplaceQu', 'GarageTyBlt', 'GarageFinish', 'GarageCars', 'GarageQual',
'GarageCond', 'PavedDrive', 'WoodDeckSF',
            'OpenPorchSF', 'EnclosedPorch', 'ThreeSsnPorch', 'ScreenPorch', 'PoolArea',
'PoolQC', 'Fence', 'MiscFeature',
            'MiscVal', 'MoSold', 'YrSold', 'SaleType', 'SaleCondition', 'dropCondition',
'WoodDeckInd', 'Neighborhood1',
            'Neighborhood2', 'Heating', 'u', 'GarageType', 'Electrical', 'PorchInd',
'TotRmsAbvGrd', 'Neighborhood3');

train.clean <-train.df[,!(names(eligible.population) %in% drop.list)];

#This list is the same as drop.list1 except:
#'GarageType', 'Electrical', 'PorchInd', 'TotRmsAbvGrd',
#were also dropped from the data set

#Table of variables used in train.clean data frame
r1tc <- t(colnames(train.clean)[1:4])
r2tc <- t(colnames(train.clean)[5:8])
r3tc <- t(colnames(train.clean)[9:12])
r4tc <- t(colnames(train.clean)[13:16])
train.clean.table <- rbind(r1tc,r2tc,r3tc,r4tc)
rownames(train.clean.table) <- c('train.clean data frame', ' ', ' ', ' ')
grid.table((train.clean.table))
```

```
#########################################################################################
############
# Delete observations with missing values
#########################################################################################
############
#########This is for the first attempt at a train list

train.clean1.df2 <- na.omit(train.clean1);

dim(train.clean1)
dim(train.clean1.df2)

dim(train.clean1)-dim(train.clean1.df2)

##########This is for alternate train list - the list used for the assignment
train.clean.df2 <- na.omit(train.clean);

#this worked with the stepAIC() below
# Check the change in dimension;
dim(train.clean)
dim(train.clean.df2)

dim(train.clean)-dim(train.clean.df2)


###########################Model
Identification###########################################

#################Using
train.clean1.df2#################################################
#####This was the initial analysis, but I decided to go back and remove some predictor
variables###
#####and re-run the stepAIC. The code is left for documentation
purposes.######################
########################################################
# Define the upper model as the FULL model
#upper1.lm <- lm(SalePrice ~ .,data=train.clean1.df2);
#summary(upper1.lm)

# Define the lower model as the Intercept model
#lower1.lm <- lm(SalePrice ~ 1,data=train.clean1.df2);
#summary(lower1.lm)

# Need a SLR to initialize stepwise selection
#sqft1.lm <- lm(SalePrice ~ TotalSqftCalc,data=train.clean1.df2);
#summary(sqft1.lm)
```

```
##################StepAIC - FS, BE, and SM methods##########
# Call stepAIC() for variable selection
#forward1.lm <- stepAIC(object=lower1.lm, scope=list(upper=formula(upper1.lm),
lower=~1), direction=c('forward'));
#summary(forward1.lm)

#Model residual summary table
#grid.table(t(round(summary(forward1.lm$residuals), 0)))
#Model summary table of coefficient
#f1.coefficients <-  tidy(forward1.lm)
#grid.table(f1.coefficients, row = NULL)

#backward1.lm <- stepAIC(object=upper1.lm,direction=c('backward'));
#summary(backward1.lm)

#stepwise1.lm <-
stepAIC(object=sqft1.lm,scope=list(upper=formula(upper1.lm),lower=~1),
#                    direction=c('both'));
#summary(stepwise1.lm)



############################Alternate drop list
modeling################################
# This is the analysis used for the assignment
#################Using train.clean.df2##############
# Define the upper model as the FULL model
upper.lm <- lm(SalePrice ~ .,data=train.clean.df2);
summary(upper.lm)

# Define the lower model as the Intercept model
lower.lm <- lm(SalePrice ~ 1,data=train.clean.df2);
summary(lower.lm)

# Need a SLR to initialize stepwise selection
sqft.lm <- lm(SalePrice ~ TotalSqftCalc,data=train.clean.df2);
summary(sqft.lm)


###############Junk Model####################################
#Fourth "junk" model for model comparison purposes
junk.lm <- lm(SalePrice ~ OverallQual + OverallCond + QualityIndex + GrLivArea +
            TotalSqftCalc, data=train.df)
summary(junk.lm)
#Model residual summary table
```

```
grid.table(t(round(summary(junk.lm$residuals), 0)), rows = c('Residuals'))
#Model summary table of coefficient
j.coefficients <-  tidy(junk.lm)
grid.table(j.coefficients, row = NULL)
#Table of model ANOVA
Junk.summary <- t(model.summary(junk.lm))
colnames(Junk.summary) <- c('Residual Standard Error', 'Adjusted R-squared','Multiple
R-squared','F-Statistic')
grid.table(Junk.summary)

##################################################################
# Note: There is only one function for classical model selection in R - stepAIC();
# stepAIC() is part of the MASS library.
# The MASS library comes with the BASE R distribution, but you still need to load it;
library(MASS)

# Call stepAIC() for variable selection
#########################Forward Selection#####################
forward.lm <- stepAIC(object=lower.lm, scope=list(upper=formula(upper.lm), lower=~1),
direction=c('forward'));
summary(forward.lm)

#Model residual summary table
grid.table(t(round(summary(forward.lm$residuals), 0)), rows = c('Residuals'))
#Model summary table of coefficient
f.coefficients <-  tidy(forward.lm)
grid.table(f.coefficients, row = NULL)

options(scipen = 2)
#Function for model summary statistics
model.summary <- function(model){
  residualse.summary <- round(summary(lm(model))$sigma,1)
  adjrs.summary <- round(summary(lm(model))$adj.r.squared,4)
  multrs.summary <- round(summary(lm(model))$r.squared,4)
  fstat.summary <- round(unname(summary(lm(model))$fstatistic)[1], 1)
  return(c(residualse.summary, adjrs.summary, multrs.summary, fstat.summary))
}

#Table of model ANOVA
Forward.summary <- t(model.summary(forward.lm))
colnames(Forward.summary) <- c('Residual Standard Error', 'Adjusted R-
squared','Multiple R-squared','F-Statistic')
grid.table(Forward.summary)

########################Backward Elimination####################
backward.lm <- stepAIC(object=upper.lm,direction=c('backward'));
```

```
summary(backward.lm)

#Model residual summary table
grid.table(t(round(summary(backward.lm$residuals), 0)), rows = c('Residuals'))
#Model summary table of coefficient
b.coefficients <-  tidy(backward.lm)
grid.table(b.coefficients, row = NULL)
#Table of model ANOVA
Backward.summary <- t(model.summary(backward.lm))
colnames(Backward.summary) <- c('Residual Standard Error', 'Adjusted R-
squared','Multiple R-squared','F-Statistic')
grid.table(Backward.summary)

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

#Model residual summary table
grid.table(t(round(summary(stepwise.lm$residuals), 0)), rows = c('Residuals'))
#Model summary table of coefficient
s.coefficients <-  tidy(stepwise.lm)
grid.table(s.coefficients, row = NULL)
#Table of model ANOVA
Stepwise.summary <- t(model.summary(stepwise.lm))
colnames(Stepwise.summary) <- c('Residual Standard Error', 'Adjusted R-
squared','Multiple R-squared','F-Statistic')
grid.table(Stepwise.summary)

##########The models from train.clean.df2 will be used for the rest of the assignment
#These were selected because it was a simplier model (fewer variables) with more
significant t values for the
#coefficients, and the adjustment had little effect on the r-squared.




###############Compute VIF values for the variable selection models#################

library(car)
forward.vif <- sort(vif(forward.lm),decreasing=TRUE)
backward.vif <- sort(vif(backward.lm),decreasing=TRUE)
stepwise.vif <- sort(vif(stepwise.lm),decreasing=TRUE)
sort(vif(junk.lm), decreasing = TRUE)
#These values are sorted from all the output columns, but allows for easy visualization of
the highest VIF values
```

```
#Table of VIF function output for each model
grid.table(round(vif(forward.lm), 4))
grid.table(round(vif(backward.lm), 4))
grid.table(round(vif(stepwise.lm), 4))

#Table of VIF for junk model
vif.junk <- round(vif(junk.lm),5)
rownames(vif.junk) <- c('OverallQual', 'OverallCond', 'QualityIndex', 'GrLivArea',
'TotalSqftCalc')
grid.table(vif.junk, rows = c('OverallQual', 'OverallCond', 'QualityIndex', 'GrLivArea',
'TotalSqftCalc'),
        cols = c('VIF'))


###################Compute MSE and MAE for models

#Function to calculate Adjusted R-squared, AIC, BIC, MSE, and MAE
#model = model.lm
Model.fit <- function(model){
  adj.rsquare <- round(summary(lm(model))$adj.r.squared, 4)
  AIC.fit <- round(AIC(model), 0)
  BIC.fit <- round(BIC(model), 0)
  MSE.fit <- round(mean(model$residuals^2), 0)
  MAE.fit <- round(mean(abs(model$residuals)), 0)
  return(c(adj.rsquare, AIC.fit, BIC.fit, MSE.fit, MAE.fit))
}

#Model Comparison
#Table of R-squared, AIC, BIC, MSE, and MAE for each Model
Forward <- Model.fit(forward.lm)
Backward <- Model.fit(backward.lm)
Stepwise <- Model.fit(stepwise.lm)
Junk <- Model.fit(junk.lm)
fit.compare <- rbind(Forward, Backward, Stepwise, Junk)
colnames(fit.compare) <- c('Adjusted\nR-squared', 'AIC', 'BIC', 'MSE', 'MAE')
grid.table(fit.compare)

#Model Rank
#Table of Model Rank for each criterion above (R-squared, AIC, BIC, MSE, MAE)
First <- c('Model 1, 2, and 3', 'Model 1, 2, and 3', 'Model 1, 2, and 3', 'Model 1, 2, and 3',
'Model 1, 2, and 3')
Second <- c('Junk', 'Junk', 'Junk', 'Junk', 'Junk')
model.rank <- rbind(First, Second)
colnames(model.rank) <- c('Adjusted\nR-squared', 'AIC', 'BIC', 'MSE', 'MAE')
grid.table(model.rank)
```

```
##############################Predictive Accuracy####################
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)

ft.residuals <- (test.df$SalePrice - forward.test)
bt.residuals <- (test.df$SalePrice - backward.test)
st.residuals <- (test.df$SalePrice - stepwise.test)
jt.residuals <- (test.df$SalePrice - junk.test)


#Calculate the MSE and MAE of each model
ft.mse <- round(mean(ft.residuals^2), 0)
ft.mae <- round(mean(abs(ft.residuals)), 0)

bt.mse <- round(mean(bt.residuals^2), 0)
bt.mae <- round(mean(abs(bt.residuals)), 0)

st.mse <- round(mean(st.residuals^2), 0)
st.mae <- round(mean(abs(st.residuals)), 0)

jt.mse <- round(mean(jt.residuals^2), 0)
jt.mae <- round(mean(abs(jt.residuals)), 0)


#Create table of predictive accuracy with model MSE and MAE
forward.predict.fit <- cbind(ft.mse, ft.mae)
backward.predict.fit <- cbind(bt.mse, bt.mae)
stepwise.predict.fit <- cbind(st.mse, st.mae)
junk.predict.fit <- cbind(jt.mse, jt.mae)
Predict.fit.table <- rbind(forward.predict.fit, backward.predict.fit, stepwise.predict.fit,
junk.predict.fit)
colnames(Predict.fit.table) <- c('MSE', 'MAE')
rownames(Predict.fit.table) <- c('Forward', 'Backward', 'Stepwise', 'Junk')
grid.table(Predict.fit.table)


#Plots of sale price vs test data predicted sale price
par(mfrow=c(2,2))
plot(forward.test, test.df$SalePrice,
    col = "darkblue",
    cex.lab = 1.5,
    xlab = "Forward Test Prediction", ylab = "Test Data Sale Price")
```

```
plot(backward.test, test.df$SalePrice,
    col = 'darkred',
    cex.lab = 1.5,
    xlab = "Backward Test Prediction", ylab = "Test Data Sale Price")

plot(stepwise.test, test.df$SalePrice,
    col = 'darkgreen',
    cex.lab = 1.5,
    xlab = "Stepwise Test Prediction", ylab = "Test Data Sale Price")

plot(junk.test, test.df$SalePrice,
    cex.lab = 1.5,
    xlab = "Junk Test Prediction", ylab = "Test Data Sale Price")
par(mfrow=c(1,1))


##################################Operational
Validation#############################

# Training Data
# Abs Pct Error
########Forward Train Test####################################
forward.pct <- abs(forward.lm$residuals)/train.clean$SalePrice;
# Assign Prediction Grades;
forward.trainPredictionGrade <- 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+]')
                            )
)
forward.trainTable <- table(forward.trainPredictionGrade)
forward.trainTable/sum(forward.trainTable)
#Create table for paper
forward.traingrade <- cbind(signif((forward.trainTable[1]/sum(forward.trainTable)),4),
                  signif((forward.trainTable[2]/sum(forward.trainTable)),4),
                  signif((forward.trainTable[3]/sum(forward.trainTable)),4),
                  signif((forward.trainTable[4]/sum(forward.trainTable)),4))
colnames(forward.traingrade) <- c('Grade 1: [0.0.10]', 'Grade 2: (0.10,0.15]', 'Grade 3:
(0.15,0.25]', 'Grade 4: (0.25+]')
rownames(forward.traingrade) <- c('Forward Train')
grid.table(forward.traingrade)

########Backward Train Test####################################
backward.pct <- abs(backward.lm$residuals)/train.clean$SalePrice;
# Assign Prediction Grades;
backward.trainPredictionGrade <- 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+]')
                        )
)
backward.trainTable <- table(backward.trainPredictionGrade)
backward.trainTable/sum(backward.trainTable)
#Create table for paper
backward.traingrade <-
cbind(signif((backward.trainTable[1]/sum(backward.trainTable)),4),
          signif((backward.trainTable[2]/sum(backward.trainTable)),4),
          signif((backward.trainTable[3]/sum(backward.trainTable)),4),
          signif((backward.trainTable[4]/sum(backward.trainTable)),4))
colnames(backward.traingrade) <- c('Grade 1: [0.0.10]', 'Grade 2: (0.10,0.15]', 'Grade 3:
(0.15,0.25]', 'Grade 4: (0.25+]')
rownames(backward.traingrade) <- c('Backward Train')
grid.table(backward.traingrade)

########Stepwise Train Test#####################################
stepwise.pct <- abs(stepwise.lm$residuals)/train.clean$SalePrice;
# Assign Prediction Grades;
stepwise.trainPredictionGrade <- 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+]')
                          )
)
stepwise.trainTable <- table(stepwise.trainPredictionGrade)
stepwise.trainTable/sum(stepwise.trainTable)
#Create table for paper
stepwise.traingrade <- cbind(signif((stepwise.trainTable[1]/sum(stepwise.trainTable)),4),
          signif((stepwise.trainTable[2]/sum(stepwise.trainTable)),4),
          signif((stepwise.trainTable[3]/sum(stepwise.trainTable)),4),
          signif((stepwise.trainTable[4]/sum(stepwise.trainTable)),4))
colnames(stepwise.traingrade) <- c('Grade 1: [0.0.10]', 'Grade 2: (0.10,0.15]', 'Grade 3:
(0.15,0.25]', 'Grade 4: (0.25+]')
rownames(stepwise.traingrade) <- c('Stepwise Train')
grid.table(stepwise.traingrade)


#########Junk Train Test#########################################
junk.pct <- abs(junk.lm$residuals)/train.clean$SalePrice;
# Assign Prediction Grades;
junk.trainPredictionGrade <- 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+]')
                                )
)
junk.trainTable <- table(junk.trainPredictionGrade)
junk.trainTable/sum(junk.trainTable)
#Create table for paper
junk.traingrade <- cbind(signif((junk.trainTable[1]/sum(junk.trainTable)),4),
            signif((junk.trainTable[2]/sum(junk.trainTable)),4),
            signif((junk.trainTable[3]/sum(junk.trainTable)),4),
            signif((junk.trainTable[4]/sum(junk.trainTable)),4))
colnames(junk.traingrade) <- c('Grade 1: [0.0.10]', 'Grade 2: (0.10,0.15]', 'Grade 3:
(0.15,0.25]', 'Grade 4: (0.25+]')
rownames(junk.traingrade) <- c('Junk Train')
grid.table(junk.traingrade)

####################Summary of all Train data grades##########
#Create table for all Train data grades
all.traingrade <- rbind(forward.traingrade, backward.traingrade, stepwise.traingrade,
junk.traingrade)
colnames(all.traingrade) <- c('Grade 1: [0.0.10]', 'Grade 2: (0.10,0.15]', 'Grade 3:
(0.15,0.25]', 'Grade 4: (0.25+]')
rownames(all.traingrade) <- c('Forward Train','Backward Train','Stepwise Train','Junk
Train')
grid.table(all.traingrade)



#################################################################
# Test Data
# Abs Pct 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;

# Assign Prediction Grades;
##############Forward.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+]')
                        )
)
forward.testTable <-table(forward.testPredictionGrade)
forward.testTable/sum(forward.testTable)
```

```
#Create table for paper
forward.testgrade <- cbind(signif((forward.testTable[1]/sum(forward.testTable)),4),
                  signif((forward.testTable[2]/sum(forward.testTable)),4),
                  signif((forward.testTable[3]/sum(forward.testTable)),4),
                  signif((forward.testTable[4]/sum(forward.testTable)),4))
colnames(forward.testgrade) <- c('Grade 1: [0.0.10]', 'Grade 2: (0.10,0.15]', 'Grade 3:
(0.15,0.25]', 'Grade 4: (0.25+]')
rownames(forward.testgrade) <- c('Forward Test')
grid.table(forward.testgrade)


############Backward.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+]')
                              )
)
backward.testTable <-table(backward.testPredictionGrade)
backward.testTable/sum(backward.testTable)
#Create table for paper
backward.testgrade <- cbind(signif((backward.testTable[1]/sum(backward.testTable)),4),
                  signif((backward.testTable[2]/sum(backward.testTable)),4),
                  signif((backward.testTable[3]/sum(backward.testTable)),4),
                  signif((backward.testTable[4]/sum(backward.testTable)),4))
colnames(backward.testgrade) <- c('Grade 1: [0.0.10]', 'Grade 2: (0.10,0.15]', 'Grade 3:
(0.15,0.25]', 'Grade 4: (0.25+]')
rownames(backward.testgrade) <- c('Backward Test')
grid.table(backward.testgrade)


############Stepwise.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+]')
                              )
)
stepwise.testTable <-table(stepwise.testPredictionGrade)
stepwise.testTable/sum(stepwise.testTable)
#Create table for paper
stepwise.testgrade <- cbind(signif((stepwise.testTable[1]/sum(stepwise.testTable)),4),
                  signif((stepwise.testTable[2]/sum(stepwise.testTable)),4),
                  signif((stepwise.testTable[3]/sum(stepwise.testTable)),4),
                  signif((stepwise.testTable[4]/sum(stepwise.testTable)),4))
```

```
colnames(stepwise.testgrade) <- c('Grade 1: [0.0.10]', 'Grade 2: (0.10,0.15]', 'Grade 3:
(0.15,0.25]', 'Grade 4: (0.25+]')
rownames(stepwise.testgrade) <- c('Stepwise Test')
grid.table(stepwise.testgrade)




##########Junk.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+]')
                              )
)
junk.testTable <-table(junk.testPredictionGrade)
junk.testTable/sum(junk.testTable)
#Create table for paper
junk.testgrade <- cbind(signif((junk.testTable[1]/sum(junk.testTable)),4),
            signif((junk.testTable[2]/sum(junk.testTable)),4),
            signif((junk.testTable[3]/sum(junk.testTable)),4),
            signif((junk.testTable[4]/sum(junk.testTable)),4))
colnames(junk.testgrade) <- c('Grade 1: [0.0.10]', 'Grade 2: (0.10,0.15]', 'Grade 3:
(0.15,0.25]', 'Grade 4: (0.25+]')
rownames(junk.testgrade) <- c('Junk Test')
grid.table(junk.testgrade)

####################Summary of all Test data grades##########
#Create table for all Test data grades
all.testgrade <- rbind(forward.grade, backward.grade, stepwise.grade, junk.grade)
colnames(all.testgrade) <- c('Grade 1: [0.0.10]', 'Grade 2: (0.10,0.15]', 'Grade 3:
(0.15,0.25]', 'Grade 4: (0.25+]')
rownames(all.testgrade) <- c('Forward Test','Backward Test','Stepwise Test','Junk Test')
grid.table(all.testgrade)
```