Oscar Hernandez
MSDS410
Summer 2018
Assignment 5

Contents

Introduction

The purpose of this report is to provide results from the steps involved in fitting regression models that use the Ames Housing data. The housing data set contains 2930 observations spread across 83 variables which are either continuous, discrete, ordinal or nominal. The variables are representative of housing characteristics such as bathroom square footage, size of garage, and exterior quality. The report will summarize key aspects of the model building process. Specifically, the report will cover how the sample population data was determined, results from exploring the use of automated variable selection techniques for model identification, an assessment of the predictive accuracy of our models and a summary of the differences between a statistical model validation and an application model validation. The final goal is to determine which model is most appropriate to predict the value of a "typical" home.

Section 1: Sample Definition and Data Split

Sample Definition

As mentioned, it is necessary to define a "typical" home. The "typical" home will represent our population of interest and will ultimately become our sample population data. Therefore, we will create "drop conditions" from the original housing data. The "drop conditions" that have been chosen to represent our "typical" home will be placed in a waterfall which is displayed in Figure 1.

Drop Condition	Observations
01: Not Residential	168
02: Not SFR	440
03: Lot Area Greater than 20,000 Square Feet	81
04: Poor House Condition	11
05: Built Pre-1950	533
06: Poor Exterior Condition	152
07: Does not have all utilities	1
08: Home Non-Functional	5
09: Non-Normal Sale	267
10: Sale Price Greater than $400,000	26
99: Eligible Sample	1246

Figure 1: Drop Condition Waterfall

The "drop conditions" listed above represent items that are not representative of a "typical home" in Ames, Iowa. For example, the decision was made to exclude homes priced over $400,000 which resulted in 26 non-eligible observations. Furthermore, it makes sense to remove observations where the property sale took place in a non-residential zone (168 observations) and the property was not a single-family home which is indicated by the "SFR" tag (440 observations). The waterfall created provides us with a sample of 1,246 eligible observations from the original 2930 in the housing data. Therefore, our current sample population contains 1,246 observations.

The Train/Test Split

Our next step is to develop a predictive modeling framework. This will be done by splitting our sample population into two data sets: one for in-sample model development (training data) and one for out-of-sample model assessment (testing data). An important characteristic of predictive modeling is assessing model performance on the out-of-sample data set. Therefore, we will utilize a uniform random number to split the sample population into a 70/30 train/test split (refer to Code section to review the R script utilized in this process).

Figure 2 provides a summary table of the observation counts in the sample population, train data and test data. The observations counts of the train data and test data should equal the counts in the sample population.

Data Set	Observation Counts
Sample Population	1246
Train Data	880
Tests Data	366

Figure 2: Table of Observation Counts

Model Identification and In-Sample Model Fit

This section of the report will be an exploration of the automated variable selection processes, results from in-sample model fitting and a comparison of the models. Our first step is to create a pool of candidate predictor variables which will be mix of discrete and continuous variables. We want to note that there was no specific justification as to why these candidate predictor variables were chosen (i.e. correlation to the response variable) since we are exploring the use of automated variable selection processes.

Furthermore, we have created two new predictor variables – QualityIndex and TotalSqftCalc. These predictor variables are simply re-expressions of some original variables that were included in the Ames Housing data and will be included in the pool of candidate predictor variables. Subsequently, we will not be using any of the original variables as candidate predictor variables that were used in the re-expression calculations. Figure 3 is a table that displays the pool of candidate predictor variables and the type of data that it represents. There is a total of 15 candidate predictor variables in the pool.

Variable	Type of Data
QualityIndex	Discrete
TotalSqftCalc	Continuous
LotArea	Continuous
GarageArea	Continuous
MiscVal	Continuous
WoodDeckSF	Continuous
OpenPorchSF	Continuous
LowQualFinSF	Continuous
BsmtUnfSF	Continuous
TotalBsmtSF	Continuous
GarageCars	Discrete
FullBath	Discrete
HalfBath	Discrete
BedroomAbvGr	Discrete
TotRmsAbvGrd	Discrete

Figure 3: Table of Candidate Predictor Variables

As part of the model-building process, we created a “train.clean” dataframe that only included the candidate predictor variables plus the SalePrice variable (response variable). This was done to effectively and easily use R in the variable selection process. The Code section of this report outlines how the “train.clean” dataframe was created.

Next, it was necessary to fit and specify an upper model (upper.lm) as the Full model and a lower model (lower.lm) as the Intercept model. Furthermore, we fitted a simple linear regression model (sqft.lm) that will be used to initialize the stepwise selection process. These models will be used as part of the inputs in the stepAIC() function in R. This is the only function for classical model selection in R and therefore, will be used to fit subsequent models in the sections below. The Full Model, Intercept Model and SLR model output summary is displayed in Figure 4, 5, 6, respectively. Also, we will not be explicitly writing out each model in mathematical form because these are not the models that are of true concern to us.

	Estimate	Std. Error t	t-value	Pr(>|t|)
(Intercept)	-55360.00	6347.00	-8.722	< 2e-16
LotArea	0.97	0.35	2.755	0.01
BsmtUnfSF	44.78	4.75	9.433	< 2e-16
TotalBsmtSF	-13.98	5.82	-2.402	0.02
LowQualFinSF	39.89	90.27	0.442	0.66
FullBath	10170.00	2302.00	4.42	0.00
HalfBath	490.70	2236.00	0.219	0.83
BedroomAbvGr	-11410.00	1806.00	-6.319	0.00
TotRmsAbvGrd	3893.00	1181.00	3.296	0.00
GarageCars	14040.00	2573.00	5.459	0.00
GarageArea	29.53	8.89	3.321	0.00
WoodDeckSF	15.58	6.25	2.492	0.01
OpenPorchSF	35.41	14.89	2.378	0.02
MiscVal	-1.32	3.06	-0.433	0.66
QualityIndex	1569.00	121.40	12.927	< 2e-16
TotalSqftCalc	57.35	4.12	13.915	< 2e-16

Figure 4: Full Model Output Summary

	Estimate	Std. Error	t-value	Pr(>|t|)
(Intercept)	187965.00	2062.00	91.14	<2e-16

Figure 5: Intercept Model Output Summary

	Estimate	Std. Error	t-value	Pr(>|t|)
(Intercept)	43287.90	4756.51	9.10	<2e-16
TotalSqftCalc	69.42	2.18	31.84	<2e-16

Figure 6: SLR Model Output Summary

Forward Variable Selection

Forward Variable Selection process involves starting with the Intercept model and then subsequently adding predictor variables (one at a time) that because of its addition produces the lowest AIC for the model. The process continues until adding any of the remaining predictor variables increases the current AIC of the model. As a result of using the stepAIC() function, we were able to fit a model using forward variable selection.

***Forward Linear Model**: SalePrice = -55505.18 + 57.78TotalSqftCalc + 45.27BsmtUnfSF + 14090.11GarageCars + 1569.82QualityIndex + 10094.77FullBath - 11395.10BedroomAbvGr - 14.75TotalBsmtSF + 0.96LotArea + 29.37GarageArea + 3903TotRmsAbvGrd + 15.73WoodDeckSF + 35.9OpenPorchSF*

We can see that using forward selection process resulted in the HalfBath, LowQualFinSF and MiscVal variables not being included in the model. Figure 7 displays a statistical summary table of the Forward Linear Model. We can see that each of the predictor variables has a statistically significant coefficient (assuming significance level of 0.05). Also, we can say that the overall model has statistical significance based on the p-value associated with the model's F-statistic. The Forward Linear Model has some positive qualities but it is too soon to deem it the "best" model.

	Estimate	Std. Error	t-value	Pr(>|t|)
(Intercept)	-55510.00	6333.00	-8.77	< 2e-16
TotalSqftCalc	57.78	3.67	15.73	< 2e-16
BsmtUnfSF	45.27	4.30	10.53	< 2e-16
GarageCars	14090.00	2509.00	5.62	0.00
QualityIndex	1570.00	121.00	12.97	< 2e-16
FullBath	10090.00	2232.00	4.52	0.00
BedroomAbvGr	-11400.00	1802.00	-6.32	0.00
TotalBsmtSF	-14.75	4.89	-3.02	0.00
LotArea	0.96	0.35	2.77	0.01
GarageArea	29.37	8.72	3.37	0.00
TotRmsAbvGrd	3903.00	1179.00	3.31	0.00
WoodDeckSF	15.72	6.24	2.52	0.01
OpenPorchSF	35.95	14.65	2.45	0.01
Model F-statistic: 446.1 (**p-value**: < 2.2e-16) **R-squared**: 0.8606**; Adjusted R-squared**: 0.8587				

Figure 7: Forward Linear Model Output

Backward Variable Selection

Backward variable selection process starts with the Full model and then subsequently removes predictor variables (one at a time) that because of its removal produces the lowest AIC for the model. The process is complete when removing any one of the remaining predictor variables does not lower the current AIC of the model. As a result of using the stepAIC() function, we were able to fit a model using backward variable selection.

***Backward Linear Model**: SalePrice = -55505.18 + 0.96LotArea + 45.27BsmtUnfSF - 14.75TotalBsmtSF + 10094.77FullBath - 11395.10BedroomAbvGr + 3903TotRmsAbvGrd + 14090.11GarageCars + 29.37GarageArea + 15.73WoodDeckSF + 35.95OpenPorchSF + 1569.82QualityIndex + 57.78TotalSqftCalc*

We can see that using backward selection process resulted in the HalfBath, LowQualFinSF and MiscVal variables being removed from the model. Figure 8 displays a statistical summary table of the Backward Linear Model. We can see that each of the predictor variables has a statistically significant coefficient (assuming significance level of 0.05). Also, we can say that the overall model has statistical significance based on the p-value associated with the model's F-statistic. The Backward Linear Model has some positive qualities but it is too soon to deem it the "best" model.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-55510.00	6333.00	-8.77	< 2e-16
LotArea	0.96	0.35	2.77	0.01
BsmtUnfSF	45.27	4.30	10.53	< 2e-16
TotalBsmtSF	-14.75	4.89	-3.02	0.00
FullBath	10090.00	2232.00	4.52	0.00
BedroomAbvGr	-11400.00	1802.00	-6.32	0.00
TotRmsAbvGrd	3903.00	1179.00	3.31	0.00
GarageCars	14090.00	2509.00	5.62	0.00
GarageArea	29.37	8.72	3.37	0.00
WoodDeckSF	15.72	6.24	2.52	0.01
OpenPorchSF	35.95	14.65	2.45	0.01
QualityIndex	1570.00	121.00	12.97	< 2e-16
TotalSqftCalc	57.78	3.67	15.73	< 2e-16
Model F-statistic: 446.1 (**p-value**: < 2.2e-16) **R-squared**: 0.8606**; Adjusted R-squared**: 0.8587				

Figure 8: Backward Linear Model Output

Stepwise Variable Selection

Stepwise selection process starts with our SLR model (null model) that was created purposely for this part. It is essentially a modification of the forward selection process but is different in the sense that predictor variables that are added to the model do not necessarily stay in. Predictor variables are added (one at a time) if its addition produces the lowest AIC for the model (unless none of them produce a lower AIC than the null model, in which case the null model is kept). After the initial iteration (assuming a predictor variable has been added), the predictor variables that are already in the model are checked to see if taking them away (one at a time) would produce the lowest AIC for the model as opposed to adding a predictor variable. As a result of using the stepAIC() function, we were able to fit a model using stepwise variable selection.

***Stepwise Linear Model**: SalePrice = -55505.18 + 57.78 TotalSqftCalc + 45.27 BsmtUnfSF + 14090.11 GarageCars + 1569.82 QualityIndex + 10094.77 FullBath -11395.10 BedroomAbvGr -14.75 TotalBsmtSF + 0.96 LotArea + 29.37 GarageArea + 3903 TotRmsAbvGrd + 15.73 WoodDeckSF + 35.95 OpenPorchSF*

We can see that using stepwise selection process resulted in the HalfBath, LowQualFinSF and MiscVal variables not included in the model. Figure 9 displays a statistical summary table of the Stepwise Linear Model. We can see that each of the predictor variables has a statistically significant coefficient (assuming significance level of 0.05). Also, we can say that the overall model has statistical significance based on the p-value associated with the model's F-statistic. The Stepwise Linear Model has some positive qualities but it is too soon to deem it the "best" model.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-55510.00	6333.00	-8.77	< 2e-16
TotalSqftCalc	57.78	3.67	15.73	< 2e-16
BsmtUnfSF	45.27	4.30	10.53	< 2e-16
GarageCars	14090.00	2509.00	5.62	0.00
QualityIndex	1570.00	121.00	12.97	< 2e-16
FullBath	10090.00	2232.00	4.52	0.00
BedroomAbvGr	-11400.00	1802.00	-6.32	0.00
TotalBsmtSF	-14.75	4.89	-3.02	0.00
LotArea	0.96	0.35	2.77	0.01
GarageArea	29.37	8.72	3.37	0.00
TotRmsAbvGrd	3903.00	1179.00	3.31	0.00
WoodDeckSF	15.72	6.24	2.52	0.01
OpenPorchSF	35.95	14.65	2.45	0.01
Model F-statistic: 446.1 (**p-value**: < 2.2e-16) **R-squared**: 0.8606**; Adjusted R-squared**: 0.8587				

Figure 8: Stepwise Linear Model Output

Another activity that is necessary to complete is creating a fourth model that will be referred to as Junk Model. The Junk Model will be used for model comparison purposes with our automated variable selection models. The mathematical formula associated with the Junk Model is as follows:
SalePrice = ß0 + ß1OverallQual + ß2OverallCond + ß3QualityIndex + ß4GrLivArea + β_5TotalSqftCal

It is called Junk Model because it contains re-expressed variables (QualityIndex and TotalSqftCal) along with the original variables that were used to create them. The Junk Model contains highly correlated predictor variables in it. Figure 9 is a table that displays some of the correlation coefficients of the predictor variables within the Junk Model. However, we will also compute the VIF for this model. Furthermore, Figure 10 is a statistical summary output of the Junk Model. We can see that all the variables have statistical significance as well as the overall model.

Predictor Variable 1	Predictor Variable 2	Correlation Coeffecient
OverallQual	QualityIndex	**0.73**
GrLivArea	TotalSqftCalc	**0.74**
OverallCond	QualityIndex	**0.49**

Figure 9: Table of Predictor Variable Correlation

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-202500.00	27550.00	-7.35	0.00
OverallQual	46670.00	4550.00	10.26	< 2e-16
OverallCond	22690.00	5116.00	4.44	0.00
QualityIndex	-3878.00	859.50	-4.51	0.00
GrLivArea	30.71	3.11	9.88	< 2e-16
TotalSqftCalc	29.18	1.84	15.88	< 2e-16
Model F-statistic: 1008 (**p-value**: < 2.2e-16) **R-squared**: 0.8522**; Adjusted R-squared**: 0.8513				

Figure 10: Junk Model Output

Before we can move on to the model validation sub-section, it is necessary to review an issue that often arises when using automated variable selection processes on pools that contain highly correlated predicator variables. The issue is that variable selection algorithms will select the highly correlated pairs. Therefore, it is necessary to compute the VIF for each of the models that we have created including the Junk Model. Figure 11 is a table of the VIF calculations each of the models. Based on the data from the table, we should not be concerned with the VIF values for any of the predictor variables (inc. indicator variables) within our variable selection models since they are all under 10. We would be concerned if we had VIF values that were 20 or larger. Furthermore, Figure 10 provides further evidence that the predictor variables in the Junk Model are correlated.

FW Model VIF					
TotalSqftCalc	BsmtUnfSF	TotalBsmtSF	GarageCars	GarageArea	QualityIndex
9.34	5.73	5.34	4.56	4.04	1.34
LotArea	OpenPorchSF	WoodDeckSF	TotRmsAbvGrd	FullBath	BedroomAbvGr
1.26	1.17	1.14	3.85	2.18	1.87
BW Model VIF					
TotalSqftCalc	BsmtUnfSF	TotalBsmtSF	GarageCars	GarageArea	QualityIndex
9.34	5.73	5.34	4.56	4.04	1.34
LotArea	OpenPorchSF	WoodDeckSF	TotRmsAbvGrd	FullBath	BedroomAbvGr
1.26	1.17	1.14	3.85	2.18	1.87
SW Model VIF					
TotalSqftCalc	BsmtUnfSF	TotalBsmtSF	GarageCars	GarageArea	QualityIndex
9.34	5.73	5.34	4.56	4.04	1.34
LotArea	OpenPorchSF	WoodDeckSF	TotRmsAbvGrd	FullBath	BedroomAbvGr
1.26	1.17	1.14	3.85	2.18	1.87
Junk Model VIF					
QualityIndex	OverallQual	OverallCond	GrLivArea	TotalSqftCalc	
64.09	50.85	31.20	3.03	2.22	

Figure 11: VIF Calculation Table

Overall, it can be concluded that the Forward, Backward and Stepwise variable selection procedures selected the exact same models. This characteristic is highlighted by the Figure 11 output (i.e. identical VIF values for each automated variable selection model) along with other summary outputs that were displayed in each of the prior three sub-sections dedicated to each procedure. The final estimated models are provided in those sub-sections as well. Also, we must note that the Junk Model estimated model and summary output is included the Stepwise section.

Model Comparison

The final sub-section is dedicated to model comparison between the Forward, Backward, Stepwise and Junk Model. We will compute the adjusted R-Squared, AIC, BIC, Mean Squared Error (MSE) and Mean Absolute Error (MAE) for each of those models for the training sample. The results are displayed in Figure 12.

	Adj. R^2	Rank	AIC	Rank	BIC	Rank	MSE	Rank	MAE	Rank
FW Model	0.8587	T-1st	20,188	T-1st	20,255	T-1st	521,164,471	T-1st	16,924	T-1st
BW Model	0.8587	T-1st	20,188	T-1st	20,255	T-1st	521,164,471	T-1st	16,924	T-1st
SW Model	0.8587	T-1st	20,188	T-1st	20,255	T-1st	521,164,471	T-1st	16,924	T-1st
Junk Model	0.8513	2nd	20,226	2nd	20,259	2nd	552,692,470	2nd	17,664	2nd

Figure 12: Model Comparison Table (in-Sample)

Each of these metrics represents come concept of "fit." Given that each of the automated variable selection processes selected the same model, it would logically mean that each of those models would have the identical output. We can see that those models score higher in terms of in-sample fit and predictive accuracy when compared to the Junk Model. However, it's important to note that the difference is not very large for some of the metrics (e.g. Adj. R^2 starts to show a difference in the thousandths place). In this example, there isn't much variation in the rank between the models. The automated selection models are all tied for 1st across all the metrics. Therefore, the Junk Model is 2nd across all metrics. Therefore, in this case, if a model is #2 in one metric, then it is #2 across all metrics, which is not typically the case. In general, we should not expect each metric to give us the same ranking of model "fit." There are situations where a model may rank the lowest in one metric but the highest in another. When selecting a model, we must already have established criteria that would like to base it on such as selecting a model that has the highest Adj. R^2 but not necessarily the lowest AIC.

Predictive Accuracy

Our next section is devoted to seeing how well our model performs (predicts) out-of-sample. We will calculate the MSE and MAE for each of the four models using the test sample. Figure 13 displays the outputs of our calculations for each model and ranks each the metrics.

	MSE	Rank	MAE	Rank
FW Model	489,266,750	T-2nd	16,352	T-2nd
BW Model	489,266,750	T-2nd	16,352	T-2nd
SW Model	489,266,750	T-2nd	16,352	T-2nd
Junk Model	477,815,489	1st	16,243	1st

Figure 13: Model Comparison Table (out-of-Sample)

From the output, we can see that the Junk Model fits the best on the out-of-sample data when using MSE and MAE as criteria. This is a somewhat surprising discovery. The automated variable selection models that fit best in-sample did not predict the best out-of-sample. This can be seen when comparing the MSE and MAE values from Figure 12 with Figure 13. In this situation, the Junk Model predicted the best out-of-sample.

In general, there should be no preference for the MSE or the MAE. It really depends on the modeler's preference and what they feel is important when it comes to errors. However, it can be argued that MSE might be more beneficial since it gives a relatively high weight to large errors which means that it is more useful when large errors are undesirable. Another important conclusion to state is that all the models had higher predictive accuracy (as measured by MAE and MSE) on the out-of-sample data than on the in-sample data. However, there are situations where the opposite occurs which may happen because a non-random sampling method was used to split the data or there is a material difference between the observations in the train/test data (i.e. the test data had a higher percentage of outliers).

Operational Validation

The final section of this report is dedicated to validating these models in a business sense, which we have not done yet. The previous sections of this report have focused on model validation from a statistical perspective. We have used MSE and MAE as metrics to compare the predictive accuracy between models using in-sample data and out-of-sample data. These two metrics on their own do not directly translate to the development of an effective business policy. For example, a company wouldn't state that their policy is that all fitted models have an MSE less than greater to 100,000. Normally, a company would state that they want all fitted models to be p% accurate which makes more sense. A similar concept will now be used to operationally validate the four models we have fitted (using both in-sample and out-of-sample data).

To operationally validate our models, we will assign a PredictionGrade to the predicted values of each model. For example, the PredictionGrade of a predicted value will be considered Grade 1 if it is within 10% of the actual value. A lower PredictionGrade will be assigned the further away the predicted value is from the actual value. The Code section report shows the R script that was used to develop our PredictionGrade Tables. Figure 14 and Figure 15 display the PredictionGrade Tables for all our models using in-sample data and out-of-sample data, respectively.

	Grade 1: [0.0 - 0.10]	Grade 2: [0.10 - 0.15]	Grade 3: [0.15 - 0.25]	Grade 4: [0.25+]
FW Model	63.30%	16.59%	14.77%	5.34%
BW Model	63.30%	16.59%	14.77%	5.34%
SW Model	63.30%	16.59%	14.77%	5.34%
Junk Model	61.93%	18.07%	14.55%	5.45%

Figure 14: PredictionGrade Table (in-sample data)

	Grade 1: [0.0 - 0.10]	Grade 2: [0.10 - 0.15]	Grade 3: [0.15 - 0.25]	Grade 4: [0.25+]
FW Model	64.48%	19.13%	13.11%	3.28%
BW Model	64.48%	19.13%	13.11%	3.28%
SW Model	64.48%	19.13%	13.11%	3.28%
Junk Model	65.03%	19.13%	11.48%	4.37%

Figure 15: PredictionGrade Table (out-of-sample data)

The PredictionGrade tables displays the proportion of predicted variables that fell within a certain percentage of the actual value for each of our models using either in-sample or out-of-sample data. For example, Figure 14 shows that approximately 62% of the predicted values in the Junk Model were within 0-10% of the actual value. Furthermore, it can be said that 62% of the predicted values were Grade 1 denoting the highest level of predictive accuracy as defined by the predetermined ranges. By in large, a large proportion of all the predicted values from the Junk Model were Grade 1 which is a desirable characteristic. This is an example of model validation in the business sense since a company could use this when comparing the predictive accuracy of their models in adherence to a certain policy.

Under this definition of predictive accuracy, we can state that all the models are above average since on average all the models had over 75% of their predicted values rated as either Grade 1 or Grade 2 (whether using in-sample or out-of-sample data). Furthermore, when compared to Figure 13 from the Predictive Accuracy section of the report (strictly using out-of-sample data), Figure 15 shows that the Junk model performed better. The Junk Model had a higher proportion of Grade 1 and Grade 2 variables than the automated variable selection models. The Junk Model also had a lower MSE and MAE (Figure 13). Therefore, the model ranking remained the same for out-of-sample data. On a lesser note, it's interesting to point out the Junk Model's rank changed when using the Figure 14 PredictionGrade Table for in-sample data than when using Figure 12's MSE/MAE results. Finally, it can be said that all our automated variable selections models could be considered "underwriting quality" based on Fannie Mae and Freddie Mac's standards.

Conclusions

Overall, this report explored the use of automated variable selection processes as part of the model building process. This approach is preferred when working with a dataset that contains a large number of variables. For example, we would not use this approach for a data set that contains 10 variables. However, the Ames Housing data contains 83 variables, so it makes sense to take this approach. The most significant conclusion from this report is to highlight that the Junk Model performed just as well both in-sample and out-of-sample and when using a business sense approach (i.e. PredictionGrade) for model validation. This serves as an example that shows correlated variables are not as big of a problem for predictive modeling as they are for statistical inference. It would appear that our Junk Model might be the most appropriate model to predict the price of a "typical" home.

Code

```
#Load MASS library
library(MASS)

#Read in our Ames Housing Data
ames_df <- read.csv('ames_housing_data.csv', header = TRUE, stringsAsFactors = FALSE)

ames_df$dropCondition <- ifelse(!ames_df$Zoning %in% c('RH','RL','RP','RM'), '01: Not Residential',
                    ifelse(ames_df$BldgType!='1Fam','02: Not SFR',
                    ifelse(ames_df$LotArea>20000,'03: Lot Area Greater than 20,000 Square Feet',
                    ifelse(ames_df$OverallCond<3,'04: Poor House Condition',
                    ifelse(ames_df$YearBuilt<1950,'05: Built Pre-1950',
                    ifelse(!ames_df$ExterCond %in% c('Ex', 'GD', 'TA', 'Fa'),'06: Poor Exterior Condition',
                    ifelse(ames_df$Utilities!='AllPub','07: Does not have all utilities',
                    ifelse(!ames_df$Functional %in% c('Typ', 'Min1', 'Min2', 'Mod'),'08: Home Non Functional',
                    ifelse(ames_df$SaleCondition!='Normal','09: Non-Normal Sale',
                    ifelse(ames_df$SalePrice>400000, '10: Sale Price Greater than $400,000',
                    '99: Eligible Sample'))))))))))

#Create a table summarizing Drop Conditions
table(ames_df$dropCondition)

#Save table as 'waterfall'
waterfall <- table(ames_df$dropCondition)

#Display the table as a column matrix
as.matrix(waterfall, 11,1)

# 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)

# 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)

# Define these two variables for later use
eligible_population$QualityIndex <- eligible_population$OverallQual*eligible_population$OverallCond
eligible_population$TotalSqftCalc <-
eligible_population$BsmtFinSF1+eligible_population$BsmtFinSF2+eligible_population$GrLivArea

# Create train/test split
train.df <- subset(eligible_population, u<0.70)
test.df  <- subset(eligible_population, u>=0.70)
```

```
# Create train.clean dataframe to help variable selection process
# Create keep.list
keep.list <- c('QualityIndex',
'TotalSqftCalc','LotArea','GarageArea','MiscVal','WoodDeckSF','OpenPorchSF','LowQualFinSF',
            'FullBath','TotalBsmtSF','BsmtUnfSF','GarageCars','HalfBath','BedroomAbvGr','TotRmsAbvGrd',
'SalePrice')

train.clean <- train.df[,(names(eligible_population) %in% keep.list)]

# Provide summary of train.clean to see if there are any missing values
summary(train.clean)

#Need to specifiy the upper model and lower models
# 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 simple linear regression model  to initialize stepwise selection
sqft.lm <- lm(SalePrice ~ TotalSqftCalc,data=train.clean)
summary(sqft.lm)

library(sjPlot)
library(sjlabelled)

#Create forward.lm
forward.lm <- stepAIC(object=lower.lm,scope=list(upper=formula(upper.lm),lower=~1),
direction=c('forward'))
summary(forward.lm)
sjt.lm(forward.lm)

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

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

#Create junk.lm
```

```
junk.lm <- lm(SalePrice~OverallQual + OverallCond + QualityIndex + GrLivArea + TotalSqftCalc,
data=train.df)
summary(junk.lm)
sjt.lm(junk.lm)

#Calculate correlation within predictor variables
cor(train.df$OverallQual, train.df$QualityIndex)
cor(train.df$GrLivArea, train.df$TotalSqftCalc)
cor(train.df$OverallCond, train.df$QualityIndex)

#Compute the VIFs for the variable selection models
library(car)
sort(vif(forward.lm),decreasing=TRUE)
sort(vif(backward.lm),decreasing=TRUE)
sort(vif(stepwise.lm),decreasing=TRUE)
sort(vif(junk.lm),decreasing=TRUE)

#Compute AIC, BIC, MSE and MAE for each of our models
AIC(forward.lm)
AIC(backward.lm)
AIC(stepwise.lm)
AIC(junk.lm)

BIC(forward.lm)
BIC(backward.lm)
BIC(stepwise.lm)
BIC(junk.lm)

mse.forward <- mean(forward.lm$residuals^2)
mae.forward <- mean(abs(forward.lm$residuals))

mse.backward <- mean(backward.lm$residuals^2)
mae.backward <- mean(abs(backward.lm$residuals))

mse.stepwise <- mean(stepwise.lm$residuals^2)
mae.stepwise <- mean(abs(stepwise.lm$residuals))

mse.junk <- mean(junk.lm$residuals^2)
mae.junk <- mean(abs(junk.lm$residuals))

#Display MSE and MAE output
mse.forward
mse.backward
mse.stepwise
mse.junk

mae.forward
mae.backward
mae.stepwise
mae.junk

#Score each model on out-of-sample data set
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)

#Calculate residuals for predicted SalePrice and test.df SalePrice
automated_residuals <- test.df$SalePrice - forward.test

junk_residuals <-  test.df$SalePrice  -junk.test


#Calcluate MAE and MSE for each model

automated.mse <- mean(automated_residuals^2)
automated.mae <- mean(abs(automated_residuals))

junk.mse <- mean(junk_residiuals^2)
junk.mae <- mean(abs(junk_residuals))

#Display results
automated.mse
automated.mae
junk.mse
junk.mae

#Define PredictionGuide variable
#Training Data
# Abs Pct 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$residuals)/train.df$SalePrice


#Assign Prediction Grades
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+]')))
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+]')))
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+]')))
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+]')))


forward.trainTable <- table(forward.PredictionGrade)
forward.trainTable/sum(forward.trainTable)

backward.trainTable <- table(backward.PredictionGrade)
backward.trainTable/sum(backward.trainTable)

stepwise.trainTable <- table(stepwise.PredictionGrade)
stepwise.trainTable/sum(stepwise.trainTable)

junk.trainTable <- table(junk.PredictionGrade)
junk.trainTable/sum(junk.trainTable)


#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 Grade
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+]')))
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+]')))
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+]')))
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+]')))

forward.testTable <-table(forward.testPredictionGrade)
forward.testTable/sum(forward.testTable)

backward.testTable <-table(backward.testPredictionGrade)
backward.testTable/sum(backward.testTable)

stepwise.testTable <-table(stepwise.testPredictionGrade)
stepwise.testTable/sum(stepwise.testTable)

junk.testTable <-table(junk.testPredictionGrade)
junk.testTable/sum(junk.testTable)
```