*Zeeshan Latifi*

*Northwestern University*

*Assignment 5*

Predict 410 Fall 2017

**Introduction:**

This is an exploratory data analysis of housing data for Ames, Iowa. In this analysis, we will be using determining factors that can help predict the sales prices for a typical home in Ames, Iowa. The data has been provided by DeCock (2011). We will be looking for several predictor variables that will help determine our response variable: Sales Price.

To do this, we’ll work through many aspects of data analysis. Initially, we’ll evaluate our data, define the sample population, set up a predictive modeling framework, explore the use of automated variable selection techniques for model validation, assess the predictive accuracy of our model using cross-validation, and compare and contrast the difference between a statistical model validation and an application model validation. From our analyses, we’ll be able to assess whether our response variable, sale price, can be predicted accurately for new observations of the predictor variable.

**Sample Population:**

There are 82 variables and 2,930 observations in the Ames, Iowa data set. We begin by conducting a waterfall in R to clean our data set. By evaluation of each variable, we’ve identified what constitutes a ‘typical’ home in Ames.

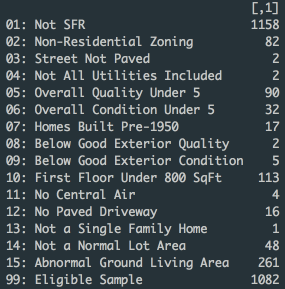
We began with filtering on the ‘SubClass’ field. This field identifies the class of the home. The decision was to keep only homes that are:

* 1-STORY 1946 & NEWER ALL STYLES
* 2-STORY 1946 & NEWER
* SPLIT OR MULTI-LEVEL

We then removed all non-residential zoning, keeping only residential high, medium, and low density. Next, removed all homes that were not on a paved street and did not have all public utilities included. A ‘typical’ home should have all standard utilities available.

To keep with our assumptions, the decision was made to only include homes that are in overall condition and quality of a 5 or higher. This means homes quality and condition ranked ‘average’ or higher. The same decision was made when filtering on the homes’ exterior quality and condition. To eliminate homes that may skew our data set, only houses that were built in 1950 or later were included. Per our definition of the ‘typical’ home, we decided to only include homes that have square footage of 800 ft.2 or higher for the first level. With that in mind we also eliminated homes without a paved driveway or central air. Also, disregarded townhomes and homes with lot areas less than 5,000 ft2 and above 20,000 ft2. Additionally, we removed homes that had above 2,000 ft.2 of above ground living area, as this is atypical in Ames, Iowa. Finally, our sample will not include homes without a garage. With these transformations, the observations were reduced down to 1,082. Figure 1 displays the count for each of the reductions.

Figure 1:



**Predictive Modeling Framework:**

In order to have a model that will be able to predict sale price of a home in Ames, we’ll need to be able to assess it out-of-sample. We will follow the 70/30 train/test split for our analyses. With a train/test split we now have two data sets: one for in-sample model development and one for out-of-sample model assessment. This is the most basic form of model cross-validation. We will estimate the models on the 70% of the data (training data) identified as the training data set, and then examine the predictive accuracy on the remaining 30% of the data (test data). Figure 2 below outlines the breakdown of the sample population partition for both data sets.

Figure 2:



**Model Identification by Automated Variable Selection and In-Sample Model Fit:**

For our automated variable selection, we’ve decided to pull several variables that are believed to be good predictors of sales price based on our previous assignments. Figure 3 below contains the variables that are going to be a part of our pool of candidate predictor variables. When beginning the selection of variables, we calculated some additional variables:

* Quality Index: Overall Quality \* Overall Condition
* Total Square Footage: Basement Finish 1 + Basement Finish 2 + Above Ground Living Area
* Total Baths: Basement Full and Half Baths + Full and Half Baths

To conduct our automated variable selection, we will utilize R to create a drop list of variables. The remaining variables will be used to for selection. Figure 3 below contains the variables that are going to be a part of our pool of candidate predictor variables.

Figure 3:

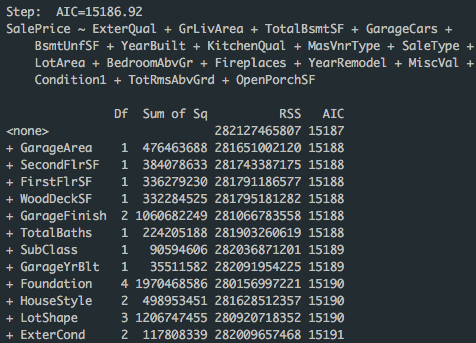
|  |  |
| --- | --- |
| **Variable** | **Description** |
| SubClass | (Nominal): Identifies the type of dwelling involved in the sale. |
| LotArea | (Continuous): Lot size in square feet |
| LotShape | (Ordinal): General shape of property |
| Condition1 | (Nominal): Proximity to various conditions |
| HouseStyle | (Nominal): Style of dwelling |
| YearBuilt | (Discrete): Original construction date |
| YearRemodel | (Discrete): Remodel date (same as construction date if no remodeling or additions) |
| MasVnrType | (Nominal): Masonry veneer type |
| ExterQual | (Ordinal): Evaluates the quality of the material on the exterior |
| ExterCond | (Ordinal): Evaluates the present condition of the material on the exterior |
| Foundation | (Nominal): Type of foundation |
| BsmtUnfSF | (Continuous): Unfinished square feet of basement area |
| TotalBsmtSF | (Continuous): Total square feet of basement area |
| FirstFlrSF | (Continuous): First Floor square feet |
| SecondFlrSF | (Continuous): Second floor square feet |
| GrLivArea | (Continuous): Above grade (ground) living area square feet |
| BedroomAbvGr | (Discrete): Bedrooms above grade (does NOT include basement bedrooms) |
| KitchenQual | (Ordinal): Kitchen quality |
| TotRmsAbvGrd | (Discrete): Total rooms above grade (does not include bathrooms) |
| Fireplaces | (Discrete): Number of fireplaces |
| GarageYrBlt | (Discrete): Year garage was built |
| GarageFinish | (Ordinal): Interior finish of the garage |
| GarageCars | (Discrete): Size of garage in car capacity |
| GarageArea | (Continuous): Size of garage in square feet |
| WoodDeckSF | (Continuous): Wood deck area in square feet |
| OpenPorchSF | (Continuous): Open porch area in square feet |
| MiscVal | (Continuous): $Value of miscellaneous feature |
| SaleType | (Nominal): Type of sale |
| OverallCond | (Ordinal): Rates the overall condition of the house |
| OverallQual | (Ordinal): Rates the overall material and finish of the house |
| SalePrice | (Continuous): Sale price $$ |
| TotalSqftCalc | Basement Finish 1 + Basement Finish 2 + Above Ground Living Area |
| TotalBaths | Basement Full and Half Baths + Full and Half Baths |
| QualityIndex | Overall Quality \* Overall Condition |

This new data frame in R will be called ‘train.clean’. This will be our data set going forward with the automated variable selection. Forward, backward, and stepwise model identification techniques will be used in order to find the best possible model. To accomplish this, we will use the stepAIC function in R.

Forward Selection:

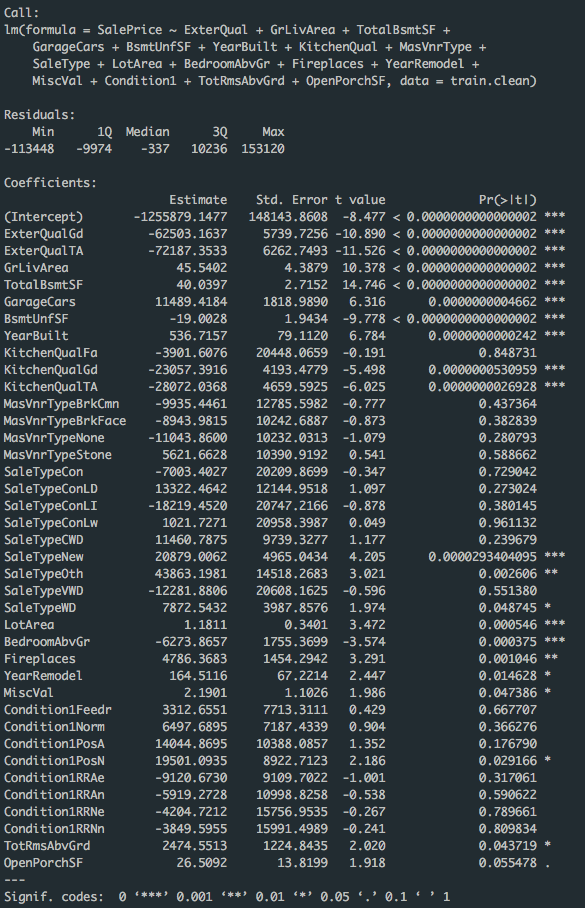
Figures 4 and 5 below illustrate the stepAIC output for the lowest value of AIC as well as the summary of the model for forward selection.

Figure 4:



The output above provides the name of the variable, that is dropped, the change in degrees of freedom, the sum of squares explained by the dropped variable, the residual sum of squares for each subset model, and the value of the AIC statistics to be used to compare models. This is the final step in the output. Adding any more predictors will then increase the AIC value. As shown in the figure above, the model’s lowest AIC value is at 15186.92. Figure 5 provides the summary statistics for the forward selection model. Note the adjusted R2 at 0.8864.

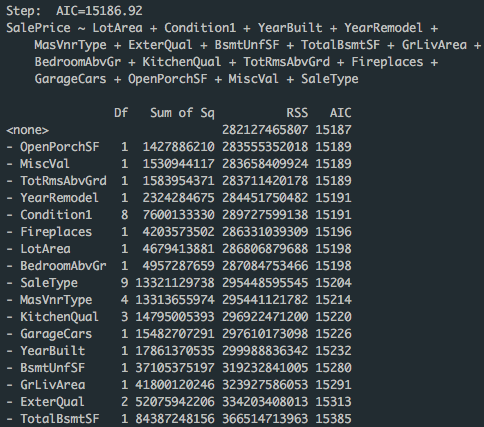
Figure 5:



Backward Selection:

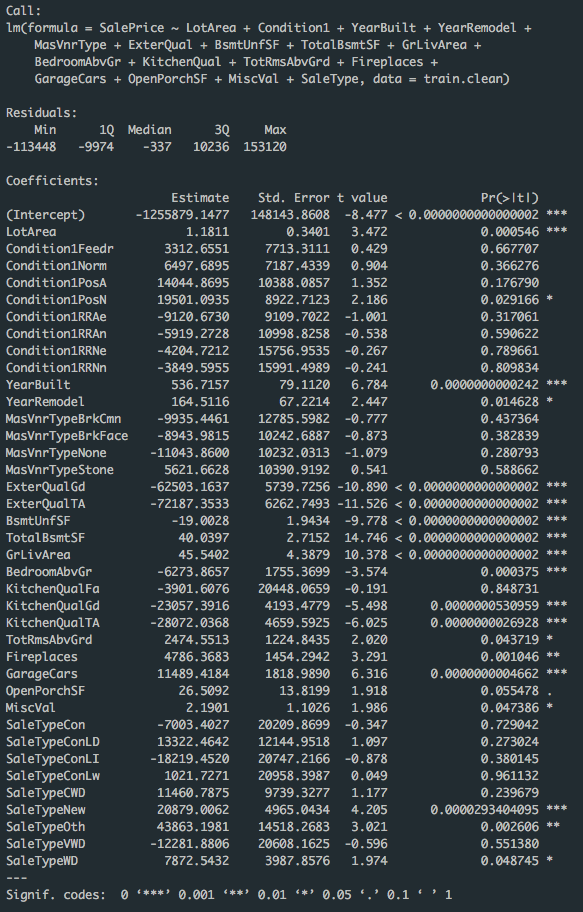
Figures 6 and 7 below illustrate the stepAIC output for the lowest value of AIC as well as the summary of the model for backward selection.

Figure 6:



This is the final step in the output. Subtracting any more predictors will then increase the AIC value. As shown in the figure above, the model’s lowest AIC value is at 15186.92. Figure 7 provides the summary statistics for the forward selection model. Note the adjusted R2 at 0.8864 similar to the forward selection model.

Figure 7:

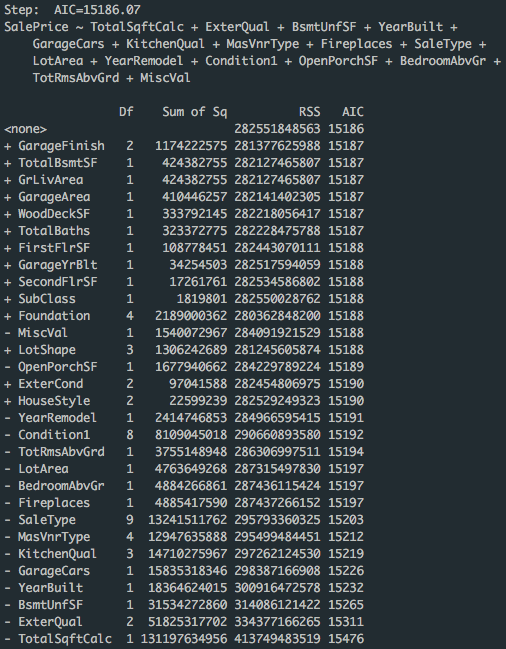




Stepwise Regression:

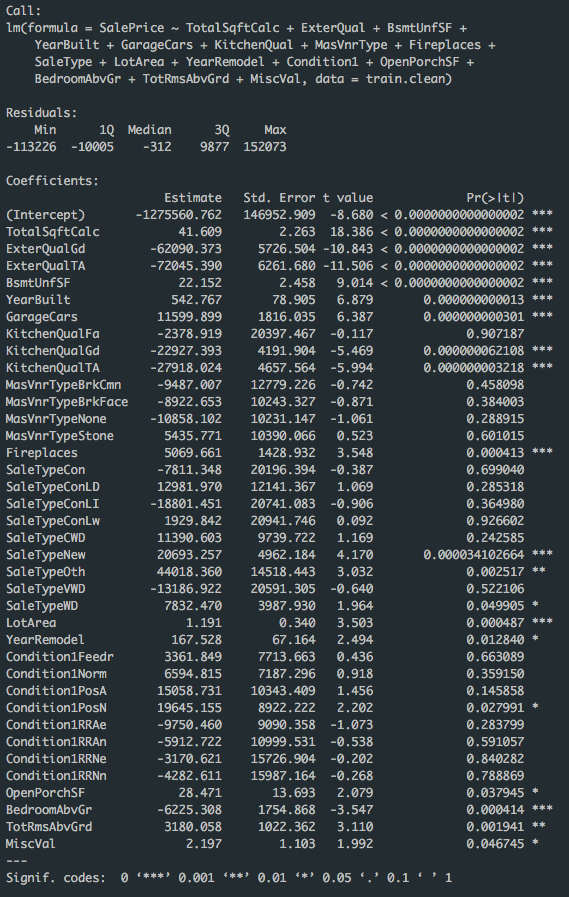
Figures 8 and 9 below illustrate the stepAIC output for the lowest value of AIC as well as the summary of the model for stepwise regression.

Figure 8:



This is the final step in the output. Further modifications of any predictors will then increase the AIC value. As shown in the figure above, the model’s lowest AIC value is at 15186.07, slightly lower than the forward and backward selection methods. Figure 9 provides the summary statistics for the forward selection model. Note the adjusted R2 at 0.8864 similar to the previous two models.

Figure 9:





Junk Model:

Figures 10 below illustrates the junk model.

Figure 10:

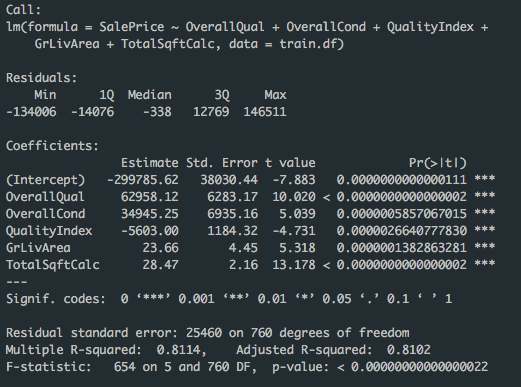
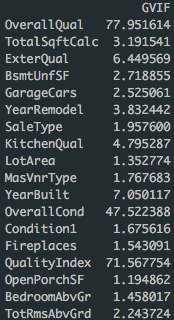
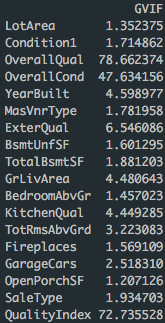
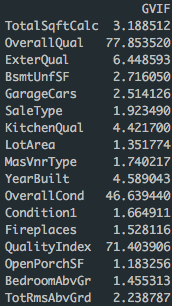


Figure 10 is the summary statistics for our junk model. The junk model is used as our baseline model. We’ve incorporated variables that we feel may be good predictors for sale price. Our data set for the junk model is also the train set. This is the data set that contains all variables prior to our drops. However, looking at the summary statistics, it seems some of variables such as OverallQual and OverallCond are highly correlated.

Notice that in our forward, backward, stepwise models there is no OverallQual, OverallCond, or QualityIndex variables present. This is because after calculating the variance inflation factor (VIF), these variables had severely high values and were very much highly correlated. Figure 11 below provides the VIF values of each variable. Once we realized the VIF values were too high, the decision was to remove it from our models, as represented by the models above. Although we did not use any indicator variables, variables with high collinearity can be disregarded when evaluating VIF.

Figure 11:

VIF: Forward VIF: Backward VIF: Stepwise

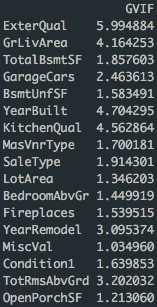
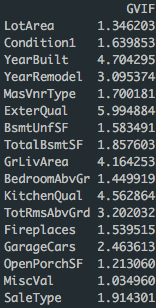
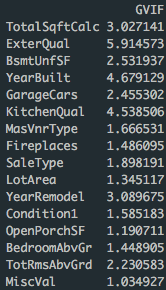
  

VIF: Junk



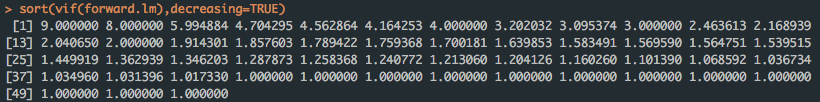
After removal of QualityIndex, OverallQual, & OverallCond

VIF: Forward VIF: Backward VIF: Stepwise

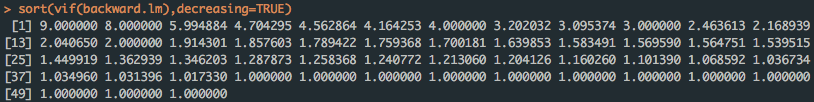
  

VIF in Descending Order

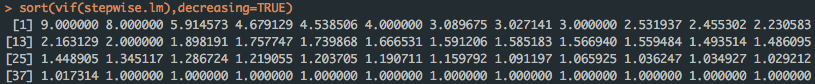
VIF Forward:



VIF Backward:



VIF Stepwise:



According to the values listed above in Figure 11, the values of VIF that were lowest were found in the stepwise regression model. This corresponds to our models above where stepwise was determined to be the best model for predictive accuracy. We will examine these models further next.

Model Comparison:

We will now compare the four models with metrics that represent some concept of ‘fit’. We’ll be ranking the metric for each model in order to determine the best model. The figure below breaks down each metric for each model:

Figure 12:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **AIC** | **BIC** | **MAE** | **MSE** | **Adj R2** | **Overall Rank** |
| **Forward** | 17362.7368970 | 17548.3841838 | 13419.4821163 | 368312618.5472388 | 0.8863657 | 2 |
| **Backward** | 17362.7368970 | 17548.3841838 | 13419.4821163 | 368312618.5472390 | 0.8863657 | 3 |
| **Stepwise** | 17361.8882664 | 17542.8943710 | 13447.5382824 | 368866643.0319906 | 0.8863511 | 1 |
| **Junk** | 17723.8363596 | 17756.3246347 | 17975.1422339 | 643230531.2298237 | 0.8101632 | 4 |

The stepwise model is ranked number 1 in AIC, BIC, and Adjusted R2. However, the MAE and MSE are lower for the forward and backward models. MAE and MSE for the forward and backward models are lower than the stepwise model. Therefore, they rank number 1 in that category. The junk model scored last in all categories, as this is expected. Based on our findings, having the lowest AIC or BIC will not always translate into having the best MAE or MSE.

**Predictive Accuracy:**

We will now evaluate how our model performs out-of-sample. Figure 13 below outlines the MAE and MSE for each model based on our test sample data (30% split).

Figure 13:

|  |  |  |
| --- | --- | --- |
|  | **MAE** | **MSE** |
| **Forward** | 14084.69 | 415431232 |
| **Backward** | 14084.69 | 415431232 |
| **Stepwise** | 14126.98 | 417760163 |
| **Junk** | 17174.77 | 593459770 |

Based on our findings, the model that fits best for the test data (out-of-sample) is either the forward selection or backward elimination models. This is in contrast to the in-sample models. According to those, the stepwise regression worked best with the lowest AIC values. However, the stepwise method did not have the lowest MAE even within the in-sample data set. Ideally there shouldn’t a preference between MAE and MSE, however, we prefer the values of MAE. This is because since MSE squared penalizes large errors more so. Interpretation of a model for predicting out of sample, MAE will be more forgiving. Generally, a better fitting model will be better at predicting in-sample data. This in turn will reflect to the out of sample data. Therefore, in our analyses, we can infer that since the model is a better predictor of in-sample, it can be considered a good predictive model. Since we based our model with the training data (in-sample) this has better predictive accuracy.

**Operational Validation:**

Within our analyses, we need to ensure that the models are within a threshold that is satisfactory to the business policies. We’ve assigned cut-off points to provide statistics on the accuracy of our predicted value. To this, in R we’ve established a variable, ‘PredictionGrade’. If the predicted value falls within 10% of the actual value, it’ll be assigned a value of Grade 1. Anything between 10%-15% will be given a Grade 2. 15%-25% will be assigned Grade 3 and anything above 25% will be assigned Grade 4.

We’ve done this for each of our models for both in-sample and out-of-sample. Figure 14 below represents the breakdown for each:

Figure 14:

Forward In-Sample



Backward In-Sample



Stepwise In-Sample



Junk In-Sample



Forward Out-of-Sample



Backward Out-of-Sample



Stepwise Out-of-Sample



Junk Out-of-Sample



Staying with the pattern of the analyses above, the stepwise regression model had the greatest predictive accuracy with the in-sample data with approximately 75% of the values were within 10% of the actual value. When evaluating out-of-sample data, forward selection and backward elimination methods were best in predictive accuracy. The model rankings look to remain the same based on the assessment of prediction grades.

**Conclusion:**

According to our findings with the several analyses above, the best model for predicting out-of-sample data for housing sale price in Ames, Iowa is the stepwise regression method. Although, the MAE was not the lowest when assessing the model both in-sample and out-of-sample (Figure 12), we felt that due to the little discrepancy between the values this model is the most accurate. The AIC and BIC values for this model were also the lowest. In addition, 75.45% of the predicted values were within the actual values, giving us the highest percentage in Grade 1 than any other model, as shown in Figure 14.

**Code Appendix:**

# Zeeshan Latifi

# 10.21.2017

# ames\_waterfall.R

# Read in csv file for Ames housing data;

# Note that back slash is an escape character in R so we use \\ when we want \;

path.name <- '/Users/Zeeshan/Desktop/PREDICT 410/Week 1/';

file.name <- paste(path.name,'ames\_housing\_data.csv',sep='');

# Read in the csv file into an R data frame;

amesiowa.df <- read.csv(file.name,header=TRUE,stringsAsFactors=FALSE);

# Single ifelse() statement

# ifelse(condition, value if condition is TRUE, value if the condition is FALSE)

# Nested ifelse() statement

# ifelse(condition1, value if condition1 is TRUE,

# ifelse(condition2, value if condition2 is TRUE,

# value if neither condition1 nor condition2 is TRUE

# )

# )

# Create a waterfall of drop conditions;

# Work the data frame as a 'table' like you would in SAS or SQL;

amesiowa.df$dropConditions <- ifelse(amesiowa.df$SubClass!= 020 & amesiowa.df$SubClass != 060 & amesiowa.df$SubClass != 080,'01: Not SFR',

ifelse(amesiowa.df$Zoning!='RH' & amesiowa.df$Zoning!='RL' & amesiowa.df$Zoning!='RM','02: Non-Residential Zoning',

ifelse(amesiowa.df$Street!='Pave','03: Street Not Paved',

ifelse(amesiowa.df$Utilities!='AllPub', '04: Not All Utilities Included',

ifelse(amesiowa.df$OverallQual<5, '05: Overall Quality Under 5',

ifelse(amesiowa.df$OverallCond<5, '06: Overall Condition Under 5',

ifelse(amesiowa.df$YearBuilt<1950, '07: Homes Built Pre-1950',

ifelse(amesiowa.df$ExterQual!='TA' & amesiowa.df$ExterQual!='Gd'& amesiowa.df$ExterQual!='Ex', '08: Below Good Exterior Quality',

ifelse(amesiowa.df$ExterCond!='TA' & amesiowa.df$ExterCond!='Gd'& amesiowa.df$ExterCond!='Ex', '09: Below Good Exterior Condition',

ifelse(amesiowa.df$FirstFlrSF<800, '10: First Floor Under 800 SqFt',

ifelse(amesiowa.df$CentralAir!='Y', '11: No Central Air',

ifelse(amesiowa.df$PavedDrive!='Y', '12: No Paved Driveway',

ifelse(amesiowa.df$BldgType!='1Fam', '13: Not a Single Family Home',

ifelse(amesiowa.df$LotArea<5000 | amesiowa.df$LotArea>20000, '14: Not a Normal Lot Area',

ifelse(amesiowa.df$GrLivArea>2000, '15: Abnormal Ground Living Area',

ifelse(amesiowa.df$GarageFinish=='NA', '16: No Garage',

'99: Eligible Sample')

)))))))))))))));

table(amesiowa.df$dropConditions)

# Save the table

waterfalls <- table(amesiowa.df$dropConditions);

# Format the table as a column matrix for presentation;

as.matrix(waterfalls,15,1)

# Eliminate all observations that are not part of the eligible sample population;

myeligible.population <- subset(amesiowa.df,dropConditions=='99: Eligible Sample');

# Check that all remaining observations are eligible;

table(myeligible.population$dropConditions);

head(myeligible.population)

##########################################################################################

#Assignment 5

#Part 2 predictive modeling framework

set.seed(123)

myeligible.population$u <- runif(n=dim(myeligible.population)[1],min=0,max=1);

myeligible.population$QualityIndex <- myeligible.population$OverallQual\*myeligible.population$OverallCond;

myeligible.population$TotalSqftCalc <- myeligible.population$BsmtFinSF1 + myeligible.population$BsmtFinSF2 + myeligible.population$GrLivArea;

myeligible.population$TotalBaths <- myeligible.population$BsmtFullBath + myeligible.population$BsmtHalfBath\*0.5 +

myeligible.population$FullBath + myeligible.population$HalfBath\*0.5

# Create train/test split;

train.df <- subset(myeligible.population, u<0.70);

test.df <- subset(myeligible.population, u>=0.70);

# Check your data split. The sum of the parts should equal the whole. # Do your totals add up?

dim(myeligible.population)[1]

dim(train.df)[1]

dim(test.df)[1]

dim(train.df)[1]+dim(test.df)[1]

framework.table <- matrix(c(dim(train.df)[1],dim(test.df)[1], dim(train.df)[1]+dim(test.df)[1]),ncol=3,byrow=TRUE)

colnames(framework.table) <- c("Training Set","Test Set","Total Set")

rownames(framework.table)<-c("Count")

fm.table <- as.table(framework.table)

fm.table

##########################################################################################

#Assignment 5

#Part 3 Model Identification by Automated Variable Selection

drop.list <- c('SID','PID','LotConfig','dropConditions','Utilities','Zoning','LotFrontage','Street', 'Fence',

'Exterior1','Exterior2','BsmtFinSF1','BsmtFinSF2','CentralAir','YrSold','MoSold','SaleCondition',

'u','train','I2010','BsmtFullBath','BsmtHalfBath','FullBath','HalfBath', 'FireplaceInd1',

'FireplaceInd2','RoofStyle','RoofFlat','PoolArea','LandContour','LandSlope','HeatingQC', 'PoolQC',

'Alley','FireplaceQu','MiscFeature','KitchenAbvGr','LowQualFinSF','Functional','EnclosedPorch',

'ThreeSsnPorch','PavedDrive','BldgType','RoofMat','Condition2','BsmtCond', 'Electrical','GarageQual',

'GarageCond','ScreenPorch','MasVnrArea','BsmtQual','BsmtExposure','BsmtFinType1','BsmtFinType2','Heating',

'GarageType','Neighborhood','OverallQual','OverallCond','QualityIndex');

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

head(train.clean)

colnames(train.clean)

#Model Identification

library(MASS)

# 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

sqft.lm <- lm(SalePrice ~ TotalSqftCalc,data=train.clean);

summary(sqft.lm)

#unlist(lapply(train.clean, function(x) any(is.na(x))))

# Call stepAIC() for variable selection

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

summary(forward.lm)

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

summary(backward.lm)

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

summary(stepwise.lm)

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

summary(junk.lm)

# Compute the VIF values

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)

vif(forward.lm)

vif(backward.lm)

vif(stepwise.lm)

vif(junk.lm)

forward.info <- c(AIC(forward.lm),BIC(forward.lm), mean(abs(forward.lm$residuals)), mean(forward.lm$residuals^2),

summary(forward.lm)$adj.r.squared, 1)

backward.info <- c(AIC(backward.lm),BIC(backward.lm), mean(abs(backward.lm$residuals)),

mean(backward.lm$residuals^2), summary(backward.lm)$adj.r.squared, 2)

stepwise.info <- c(AIC(stepwise.lm),BIC(stepwise.lm), mean(abs(stepwise.lm$residuals)),

mean(stepwise.lm$residuals^2), summary(stepwise.lm)$adj.r.squared, 3)

junk.info <- c(AIC(junk.lm),BIC(junk.lm), mean(abs(junk.lm$residuals)), mean(junk.lm$residuals^2),

summary(junk.lm)$adj.r.squared, 4)

options(scipen = 9999)

models.info <- matrix(c(AIC(forward.lm),BIC(forward.lm), mean(abs(forward.lm$residuals)), mean(forward.lm$residuals^2),

summary(forward.lm)$adj.r.squared, 1, AIC(backward.lm),BIC(backward.lm), mean(abs(backward.lm$residuals)),

mean(backward.lm$residuals^2), summary(backward.lm)$adj.r.squared, 2, AIC(stepwise.lm),BIC(stepwise.lm), mean(abs(stepwise.lm$residuals)),

mean(stepwise.lm$residuals^2), summary(stepwise.lm)$adj.r.squared, 3, AIC(junk.lm),BIC(junk.lm), mean(abs(junk.lm$residuals)), mean(junk.lm$residuals^2),

summary(junk.lm)$adj.r.squared, 4),ncol=6,byrow=TRUE)

#models.table <- matrix(c(forward.info,backward.info,stepwise.info,junk.info, nrow=6,byrow=TRUE))

colnames(models.info) <- c('AIC','BIC','MAE','MSE','Adj R2','Rank')

rownames(models.info)<-c('Forward','Backward','Stepwise','Junk')

model.tbl <- as.table(models.info)

model.tbl

##########################################################################################

#Assignment 5

#Part 4 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)

forward.pred.mae <- mean(abs(forward.test-test.df$SalePrice))

forward.pred.mse <- mean((forward.test-test.df$SalePrice)^2)

backward.pred.mae <- mean(abs(backward.test-test.df$SalePrice))

backward.pred.mse <- mean((backward.test-test.df$SalePrice)^2)

stepwise.pred.mae <- mean(abs(stepwise.test-test.df$SalePrice))

stepwise.pred.mse <- mean((stepwise.test-test.df$SalePrice)^2)

junk.pred.mae <- mean(abs(junk.test-test.df$SalePrice))

junk.pred.mse <- mean((junk.test-test.df$SalePrice)^2)

##########################################################################################

#Assignment 5

#Part 5 Operational Validation

# Training Data

# Abs Pct Error

forward.pct <- abs(forward.lm$residuals)/train.clean$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+]')

))

forward.trainTable <- table(forward.PredictionGrade)

forward.trainTable/sum(forward.trainTable)

#--------------------------------------------------------------------------------

backward.pct <- abs(backward.lm$residuals)/train.clean$SalePrice;

# Assign Prediction Grades;

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+]')

))

backward.trainTable <- table(backward.PredictionGrade)

backward.trainTable/sum(backward.trainTable)

#--------------------------------------------------------------------------------

stepwise.pct <- abs(stepwise.lm$residuals)/train.clean$SalePrice;

# Assign Prediction Grades;

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+]')

))

stepwise.trainTable <- table(stepwise.PredictionGrade)

stepwise.trainTable/sum(stepwise.trainTable)

#--------------------------------------------------------------------------------

junk.pct <- abs(junk.lm$residuals)/train.clean$SalePrice;

# Assign Prediction Grades;

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+]')

))

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 Grades;

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)

#--------------------------------------------------------------------------------

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)

#--------------------------------------------------------------------------------

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)

#--------------------------------------------------------------------------------

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)