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

Assignment Overview

In this assignment we finish our OLS regression model building activities to predict home sale price (SALEPRICE) using the variables in the AMES data set. As with the previous Modeling Assignments, this one walks you through a number of modeling experiences. Here, the modeling experiences are about automated variable selection procedures which are a way to sift through large numbers of potential explanatory variables to narrow in on those that are most likely useful in predicting the response variable. Then, to examine model validity. This entails seeing how well the developed predictive model works on new and novel similar data. The idea is to answer the question, does the model export or generalize past the data on which it was developed. The tasks for this assignment are delineated below. Each task should be completed and written about separately. This is not necessarily the sequence of steps of what you should do in a modeling setting, but it is intended to give you perspective on modeling.

Preparatory Work

This assignment assumes you are using the same Sample Population as in previous Modeling Assignments 1, 3, and 4. If you need to make adjustments to the Sample Population, please do so and report what you've done. Examine the categorical variables in the Ames Data Set. On first principles (i.e. your reasoning) which seem most likely to be related to, or predictive of, SALESPRICE? For those categorical variables that seem most reasonable or interesting, find summary statistics for Y (i.e. means, medians, std dev., etc) BY the levels of the categorical variable. Which categorical variable(s) have the greatest mean difference between levels? Why is this an important quality to look for? Create dummy coded variables for the interesting categorical variables that may be predictive of SALEPRICE. Keep in mind, the more categorical variables you want to include in your analysis, the more work required in dealing with those variables. This work goes up exponentially with the number of categorical variables retained and their numbers of levels. Be brutally honest about the potential for a categorical variable to be predictive. If you must, fit regression models to determine R-squared for the categorical variables of interest, and then select only those that have reasonably large R-squared values. You will want to compute and retain the summary statistics for SALESPRICE by group for these interesting categorical variables that you wish to retain for further analysis.

Assignment Tasks

For the tasks in this assignment, the response variable will be: SALEPRICE (Y). The remaining variables will be considered potential explanatory variables (X's).

(1) The Predictive Modeling Framework

A defining feature of predictive modeling is assessing model performance out-of-sample. We will use uniform random number to split the sample into a 70/30 train/test split. 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.

```
# Set the seed on the random number generator so you get the same split every
time that you run the code.
set.seed(123)
my.data$u <- runif(n=dim(my.data)[1],min=0,max=1);

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

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

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

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

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

(2) Model Identification by Automated Variable Selection

Create a pool of candidate predictor variables. This pool of candidate predictor variables needs to have at least 15-20 predictor variables, you can have more. The variables should be a mix of discrete and continuous variables. You can include dummy coded or effect coded variables, but not the original categorical variables. Include a well-designed list or table of your pool of candidate predictor variables in your report. **NOTE: If you need to create additional predictor variables, then you will want to create those predictor variables before you perform the train/test split outlined**

in (2). Also note that we will be using our two variables QualityIndex and TotalSqftCalc in this section.

The easiest way to use variable selection in R is to use some R tricks. If you have small data sets (small number of columns), then these tricks are not necessary. However, if you have large data sets (large number of columns), then these tricks are NECESSARY in order to use variable selection in R effectively and easily.

Trick #1: we need to create a data frame that only contains our response variable and the predictor variables that we want to include as our pool of predictor variables. We will do this by creating a drop list and using the drop list to shed the unwanted columns from `train.df` to create a 'clean' data frame.

```
drop.list <-
c('SID','PID','LotConfig','Neighborhood','HouseStyle','YearBuilt','YearRemodel',
   'Exterior1','BsmtFinSF1','BsmtFinSF2','CentralAir','YrSold','MoSold','SaleCondi
   tion','u','train','I2010','BsmtFullBath','BsmtHalfBath','FullBath','HalfBath',
'FireplaceInd1','FireplaceInd2','OverallQual','OverallCond','PoolArea','GrLivArea
');

train.clean <-train.df[,!(names(my.data) %in% drop.list)];
```

Model Identification: Using the training data find the 'best' models using automated variable selection using the techniques: forward, backward, and stepwise variable selection using the R function `stepAIC()` from the MASS library. Identify (list) each of these three models individually. Name them `forward.lm`, `backward.lm`, and `stepwise.lm`.

Note that variable selection using `stepAIC()` requires that we specify the upper and lower models in the `scope` argument. We want to perform a 'full search' or an 'exhaustive search', and hence we need to specify the upper model as the Full Model containing every predictor variable in the variable pool (or in our clean data frame!), and the lower model as the Intercept Model. Both of these models are easy to specify in R.

Trick #2: specify the upper model and lower models using these R shortcuts.

```
# 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);

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

Note that all of these models use the `train.clean` data set. We will use these three models to initialize and provide the formula needed for the `scope` argument.

Trick #3: use the R function `formula()` to pass your shortcut definition of the Full Model to the `scope` argument in `stepAIC()`. Be sure to read the help page for `stepAIC()` to understand the `scope` argument and its default value.

```
# 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.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)
```

Note that we do not specify any data sets when we call `stepAIC()`. The data set is passed along with the initializing model in the `object` argument.

In addition to these three models identified using variable selection we will include a fourth model for model comparison purposes. We will call this model `junk.lm`. The model is appropriately named. Do we know why we are calling this model junk? Note that this model will use the `train.df` data frame since I shed all of these columns off of `train.df` when I created `train.clean`.
Remember that `train.df` and `train.clean` are essentially the same data set, `train.df` just has more columns than `train.clean` so it is perfectly okay to compare models fit on `train.df` with models fit on `train.clean`.

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

Before we go any further we should consider if we like these models. One issue with using variable selection on a pool that contains highly correlated predictor variables is that the variable selection algorithm will select the highly correlated pairs. (Hint: do we have correlated predictor variables in the junk model?)

Compute the VIF values for the variable selection models. If the models selected highly correlated pairs of predictors that you do not like, then go back, add them to your drop list, and re-perform the variable selection before you go on with the assignment. The VIF values do not need to be

ideal, but if you have a very large VIF value (like 20, 30, 50 etc.), then you should consider removing a variable so that your variable selection models are not junk too.

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

```
# Compute the VIF values
library(car)
sort(vif(forward.lm),decreasing=TRUE)
sort(vif(backward.lm),decreasing=TRUE)
sort(vif(stepwise.lm),decreasing=TRUE)
```

Did the different variable selection procedures select the same model or different models? Display the final estimated models and their VIF values for each of these four models in your report.

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

(3) *Predictive Accuracy*

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

Here is an example of how you use the `predict()` function to score your model on your out-of-sample data.

```
forward.test <- predict(forward.lm,newdata=test.df);
```

(4) *Operational Validation*

We have validated these models in the statistical sense, but what about the business sense? Do MSE or MAE easily translate to the development of a business policy? Typically, in applications we need to be able to hit defined cut-off points, i.e. we set a policy that we need to be p% accurate. Let's define a variable called PredictionGrade, and consider the predicted value to be 'Grade 1' if it is within ten percent of the actual value, 'Grade 2' if it is not Grade 1 but within fifteen percent of the actual value, Grade 3 if it is not Grade 2 but within twenty-five percent of the actual value, and 'Grade 4' otherwise.

Here is a code snippet to show you how we will produce the prediction grades.

```
# 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)


# 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)
```

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

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

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

6) For which ever model you find to be "Best" after the automated variable selection procedures and all of these comparisons, you will need to re-visit that model and clean it up, as well as conduct residual diagnostics. Frankly, the end of an automated variable selection process is in many ways a starting point. What kinds of things do you want to check for and "clean up"?
    - Quantitative variables my have been selected that logically have their coefficients reversed from what it theoretically should be. For example, a final model can have a negative coefficient relating size of home in square feet to price. That wouldn't make sense. Why would that variable be included in the model – or there is something else going on, like

multicollinearity that needs to be accounted for. That has to be fixed some way. Always remember, an easy solution for multicollinearity is to simply leave an offending variable out of the model.

- Quantitative variables may be included in the model that are statistically significant, but are not actually predictive. This is mostly an issue when the sample size is large. The issue is large sample size translates into high statistical power. If too much power is present, everything can be statistically significant. Remember, statistical significance does not mean important, it means ruling out chance as the explanation. To guard against an overfit model with too many variables, consider looking at R-squared change. Examine the impact of removing each variable from the final model, one at a time. IF R-squared change for any of the retained variables is too small – why include that variable in the model? It is not contributing to the predictive ability of the model. Think about it! Parsimony is important – simpler models are easier to explain and tend to be better in the long run out of sample! Do you really need a max fit model, or a best explanation model? You are welcome to remove variables from the "final model" until all contribute sufficiently well.
- Variables included in the final model may have coefficients that are essentially zero. These need to be checked to see if those variables are predictive. Use R-squared change to determine if the variable should be retained.
- A dummy coded variable may have been selected using the automated procedure. If this has happened, you will want to include all but one of the dummy coded variables for the associated categorical variable in the model. It does not matter whether the dummy coded variables are statistically significant or not. The purpose is for interpretation of coefficients and the inclusion of the entire categorical variable in the predictive model. Think of it as variables are used – all or nothing, not just bits and pieces.
- If you retain one or more categorical variables in your final model, you have an ANCOVA model. This means you are modeling Y with parallel planes. You should then be concerned about unequal slopes for these planes. Of concern is the interaction between the categorical variable and any of the quantitative variables. This is a challenging issue if you have a large number of quantitative explanatory variables in your final model. If interaction between the categorical variable and any of the quantitative variables is a logical possibility, you'll need to test for unequal slopes.

Once you've cleaned up your model and have come to a TRUE FINAL model, you will want to conduct the typical goodness of fit and model diagnostic analysis. Hopefully, all will go well and you won't have any issues. If this is the case:

This is your final model! Report this one with pride! Go Wildcats!

IF there is heteroscedasticity or non-linear patterns to the residuals, it is back to the drawing board and the use of transformed variables. The process starts all over again. No one said this was easy!

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

Assignment Document

Results should be presented, labeled, and discussed in the numerical order of the questions given. Please use MS-WORD or some other text processing software to record and present your answers and results. The report should not contain unnecessary results or information. Tables are highly effective for summarizing data across multiple models. The document you submit to be graded MUST be submitted in pdf format. Please use the naming convention: ModelAssign6_YourLastName.pdf.