**Predicting House Prices in Ames, Iowa**

Final Project: OR 568  
December 11, 2017  
George Mason University

*By Jem Anderson, Snigdha Cheekoty, Wanru Li, Yang Liu, and Michael Ruhl*

A house is one of the most expensive things many people will buy in their lives, and many people are left wondering what really goes into assessing the value of a home, and how much a house will really sell for based on many different criteria such as living area, number of bedrooms, or whether it has a garage. With these questions in mind, Kaggle hosted a competition[[1]](#footnote-1) for users to predict housing prices based on a dataset with dozens of predictors.

The competition provided a training set of 1,460 houses containing the sales price and 79 predictors. The test set of 1,459 rows and 79 predictors was also provided, but the sale price omitted. After performing research into the dataset, we discovered that the competition data set was derived from one created by Dean De Cock of Truman State University. According to a paper in the Journal of Statistics Education[[2]](#footnote-2), Prof. De Cock obtained from the Ames, Iowa City Assessor’s Office a set of 3,970 records described by 113 variables for property sales in the city between 2006 and 2010. He removed variables that would be difficult for a “non-expert” to understand (like the weighting and adjustment factors the city used to calculate property tax assessments) and culled records that had odd features (like stand-alone garages and storage areas that contained no living space and condominiums that had living space but no lot size/characteristic information) or had been sold multiple times to produce a set of 2930 sales records whose characteristics are described by 80 variables. Of the variables, 19 were continuous (ex: sizes of rooms, length of road frontage, etc.), 14 were cardinal (ex: numbers of bedrooms & bathrooms, years constructed/upgraded, etc.), 23 were ordinal (ex: quality evaluations of various traits), and 23 were nominal (ex: location, style, etc.). Sales Price ranges from $12,789-$755,000

**Data Preprocessing**

There were significant amounts of missing data for the *Lot Frontage* (490 missing records), *Garage Yr Built* (159 missing records) and the two masonry veneer predictors: *Mas Vnr Type* and *Mas Vnr Area* (both with the same 23 missing records). There were also a few, isolated missing predictors for basement-related predictors (*Bsmt Qual, Bsmt Cond, Bsmt Exposure, BsmtFin Type 1, BsmtFin SF 1, BsmtFin Type 2, BsmtFin SF 2, Bsmt Unf SF, Total Bsmt SF, Bsmt Full Bath,* and *Bsmt Half Bath*), for *Electrical*, and for garage-related predictors (*Garage Finish, Garage Cars, Garage Area, Garage Qual* and *Garage Cond*). Missing data and NA values wreak havoc on many models and must be removed, changed or imputed. The study team elected to perform only limited imputation before providing a common dataset to the model producers.

A review of the known data for the *Lot Frontage* predictor (Figure 1) revealed there were significant numbers of houses with frontages of 21, 24, 50, 60, 65, 70, 75 and 80 feet. There was no simple, effective means readily available to estimate the missing values, and a concern that imputing so many values might skew the results; the team thus chose to leave the blank values unchanged and allow the modelers to handle the missing data individually, as best suited their model.

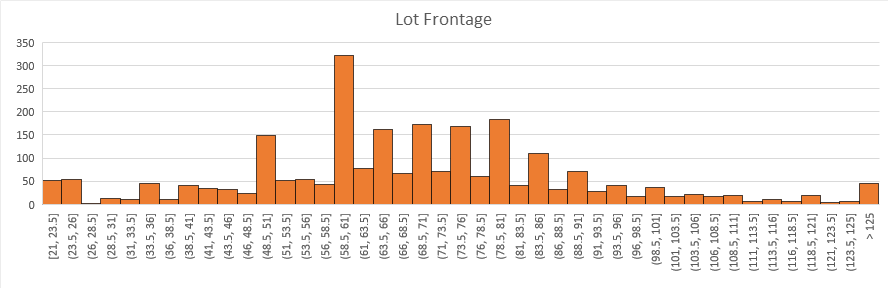


Figure 1 - The Lot Frontage parameter proved to be a difficult value to impute, so handling the missing values was left up to the individual modeler.

Next, the team examined the *Garage Yr Built* predicted values and found that of the 159 records with missing values, 157 were clearly houses that had no garage. The team chose to designate a value of “1099” for those records, to differentiate them from houses with garages. One of the two other records with missing *Garage Yr Built* predictor data had a large (360 sq ft), single-car, detached garage, and appeared to be one of several in that neighborhood with similar characteristics, constructed at the same time as the rest of the house. The final record was also missing data for other garage characteristics; there was not enough information to confidently impute the values, and the team decided to leave the record unchanged. Similarly, there was insufficient information to impute values for the 23 records with missing entries for *Mas Vnr Type* and *Mas Vnr Area* predictors; those predictor values were left undesignated.

The records with missing predictor data regarding their basements included one with missing values for *Bsmt Full Bath* and *Bsmt Half Bath*; its other predictors depicted the house as having no basement and both values for both were imputed to be zero. A second record was missing predictor values for all its basement-related predictors (*Bsmt Qual, Bsmt Cond, Bsmt Exposure, BsmtFin Type 1, BsmtFin SF 1, BsmtFin Type 2, BsmtFin SF 2, Bsmt Unf SF, Bsmt Full Bath* and *Bsmt Half Bath*), and we assumed this meant the house had no basement, and the values were imputed to “NA” or zero as appropriate to reflect that belief. There were three records with unfinished basements and missing values for *Bsmt Exposure*; they were imputed to NA values. Finally, one record had a missing predictor for *BsmtFin Type 2* and did not have other information that would allow imputation of a value; the predictor value was left unmodified. The record with missing data for its *Electrical* predictor was located in a neighborhood whose other houses constructed at that time used standard circuit breakers with Romex wiring; this value was thus imputed to *SBrkr*.

Imputation thus did not change the number of records in the file and provided minimal changes to the original values. The team also chose not to modify original NA values at this stage. This allowed the team to use the full dataset for models like single tree which are robust to missing data, and to choose other options for models that were more sensitive to missing records.

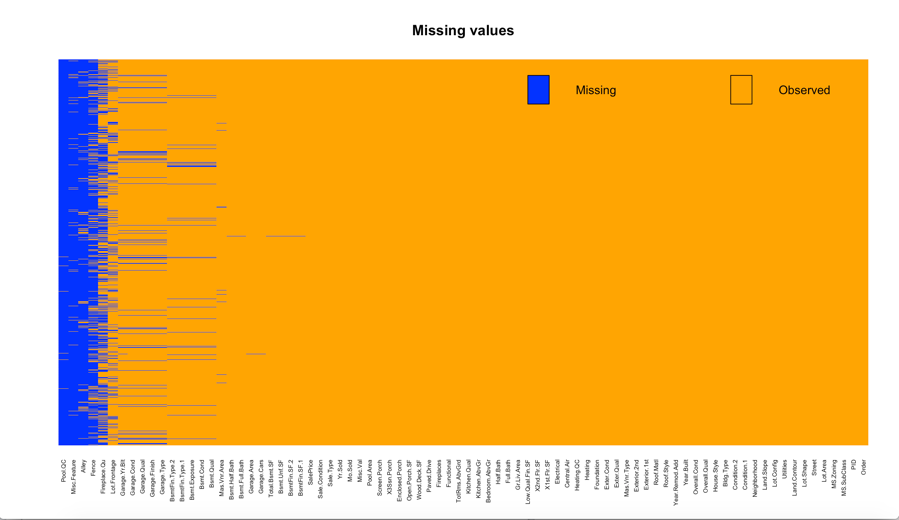


Figure 2 - An accounting of the quantity of missing cells based on predictor in our dataset.

As shown in Figure 2, the dataset had substantial quantities of missing data. To manage missing values, where needed, each modeler used variations similar to code in Figure 3 to loop through each column, count the number of cells with more than *x* NA values, and remove those columns from the set. Modelers set the threshold *x* to values such as 100 or 150. Once these columns were removed, a separate section of code (Figure 4) was used to remove any rows which had blank values.

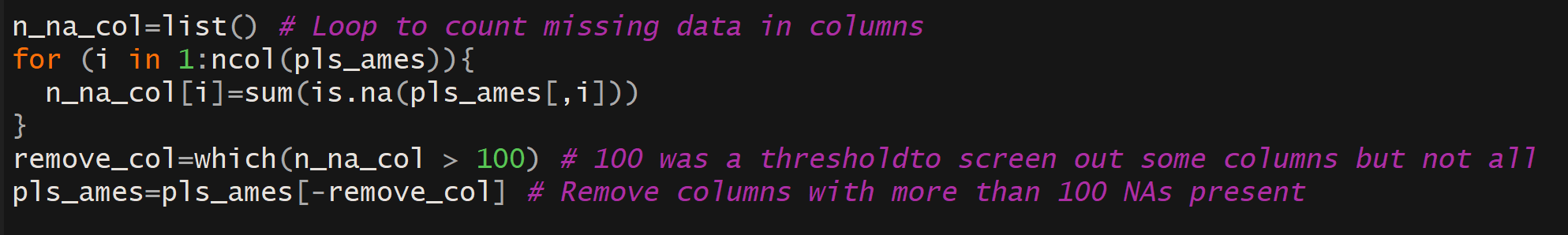


Figure 3 - A section of code used to remove columns with over 100 missing values. Some team members used code to remove columns with more than 150 missing values for their models.

The threshold to control how many values triggered a column for removal allowed the modeler the opportunity to balance the ratio of rows:columns removed. Setting a threshold of 2,000, for example, would mean that almost no columns were removed, but when part two was performed, many more rows would be removed.

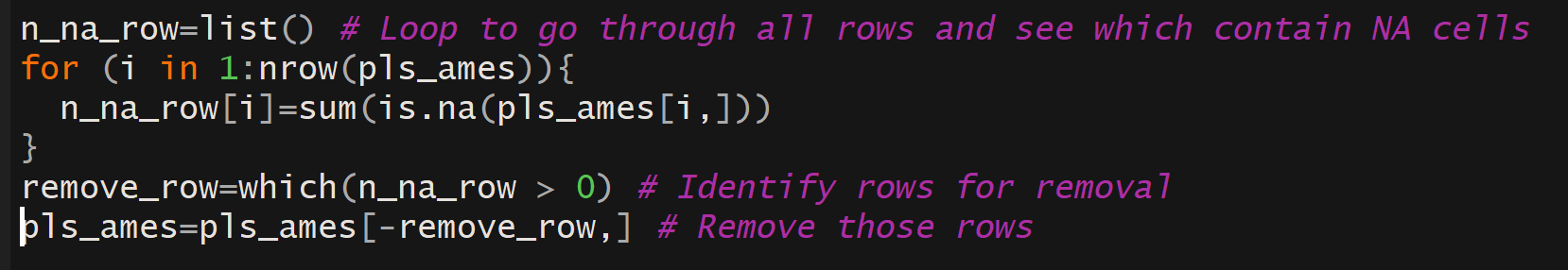


Figure 4 - A section of code to remove all rows with any missing values.

**Partial Least Square**

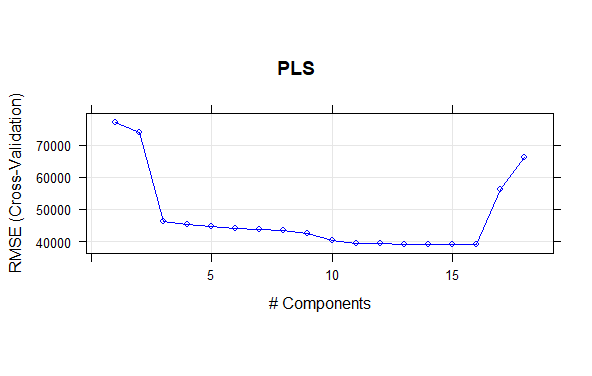
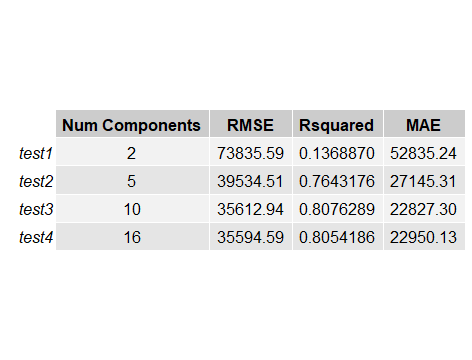


Figure 5 - The results from PLS show that 16 was the optimal number of components

To build a Partial Least Square model, categorical variables had to be treated as factors by R, so the source data was imported with the setting enabled to *Treat Strings as Factors*. The modeler removed columns with more than 100 blank NAs, removed all rows with NAs, and then removed Near Zero Variance predictors. This left a transformed dataset of 2,823 rows and 53 columns. The modeler used an 80-20 split between training and test data, and ran four rounds of testing (Figure 5) to see what the optimal number of components was.

As the plot from a 10-fold cross validation test shows (Figure 5), 16 was the optimal number of components. The error rate dropped substantially after 2 components, and saw only incremental improvements after that. PLS determined that the most important predictors for price were *Overall.Qual, Bedroom.AbvG*, and *Fireplace*. While overall quality and the number of bedrooms above ground are expected, fireplace was surprising to the modeler.

**Linear Regression**

To preprocess the data for a regression model, Near Zero Variance predictors were first removed. Categorical features had missing values imputed as “None”, numerical variables had blanks changed to 0, and for some features which had very few missing values (below 100), the features were simply removed. After the imputation and cleaning, we had a clean dataset of 2662 rows and 82 columns. A 75-25 split was used for training and test data. The target variable *SalePrice* was skewed (Figure 6), so a log transformation was applied to turn it into a normal distribution, and other numeric values also had a lot transformation applied.

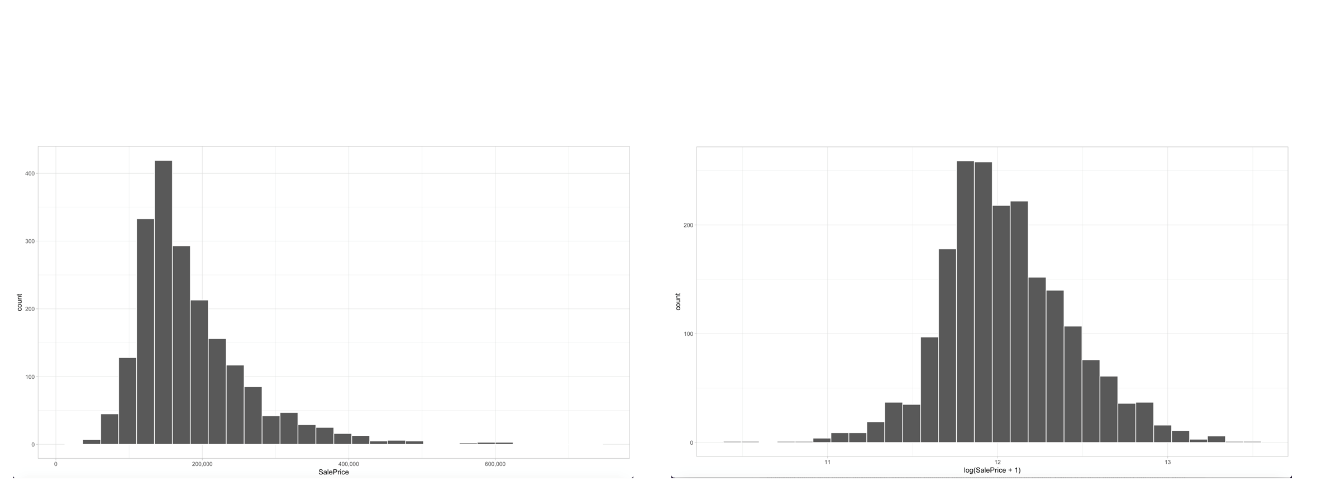


Figure 6 - The target value Sale Price was skewed, so a log transformation was applied

Correlation matrixes and coefficients were made and computed. As seen in Figure 7, there are several strong relation variables to predict house price. For example, the correlation coefficients between *OverallQual* and *SalePrice* is 0.79, which means there is a strong relationship between them. Also, the number of enclosed porches are negatively correlated with year built. Some potential buyers likely do not want an enclosed porch, so recently house developers have been building less enclosed porches.

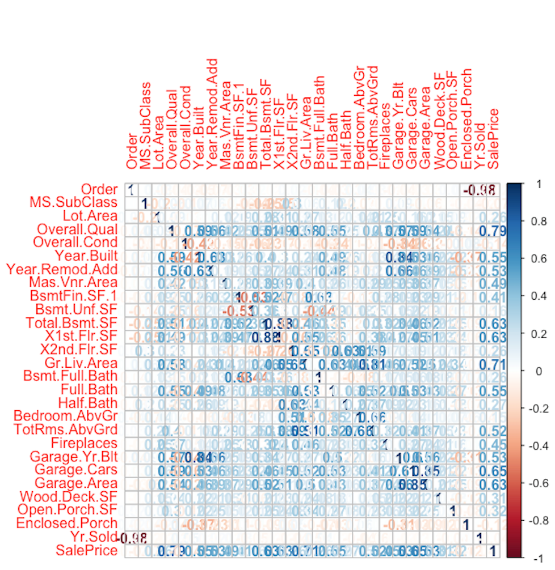
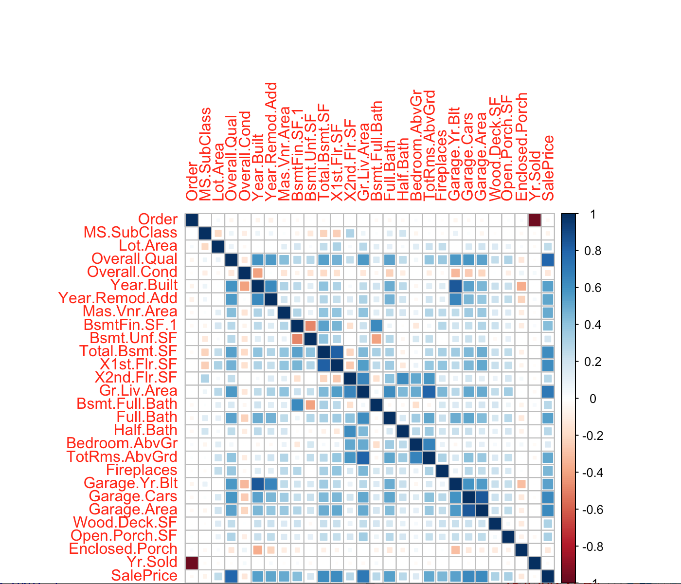


Figure 7 - Correlation Matrix and Correlation Coefficients

Two different linear regressions were run, the first involving all of the predictors. In general, the one with all predictors fit this dataset well, according to the RMSE and R-squared. We see in Figure 8 that our data meet the regression assumption very well, and residuals are lined well on the straight dashed line. In the residuals vs. fit plot, it looks like there is evidence of heteroscedasticity as the errors increase with the size of the fitted value. Not all outliers are very influential in linear regression analysis. However, there might exist some outliers that have a strong influential aspect in this dataset, because we can find these outliers in the residuals vs leverage plots.

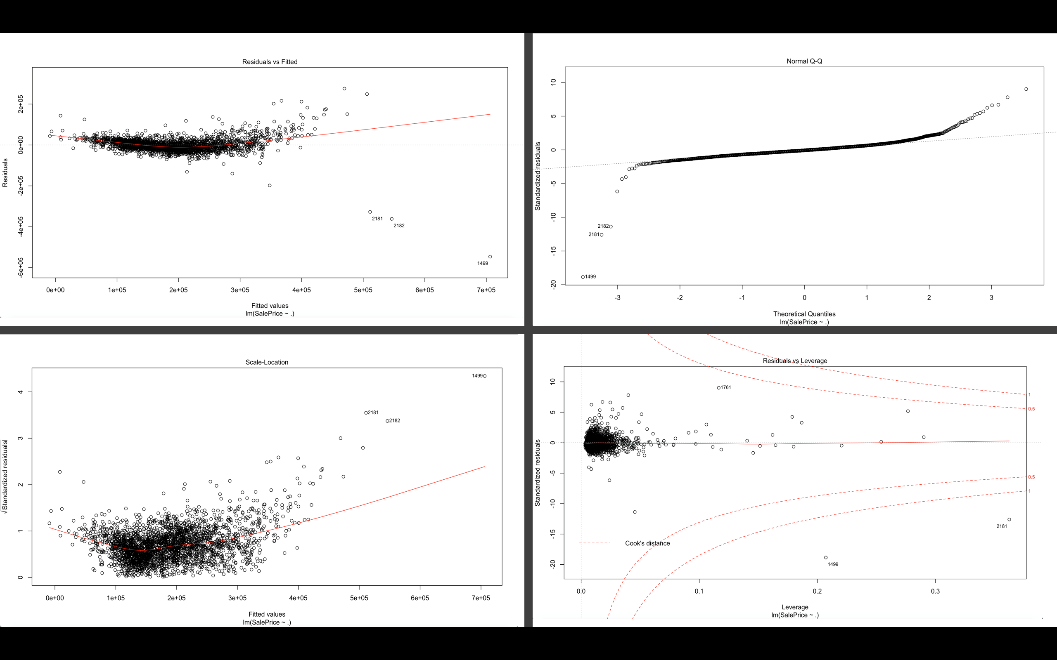


Figure 8 - Results of a linear regression involving all predictors

The second linear regression involved only 6 variables (Figure 9). Based on correlation coefficients, the modeler choose 6 high correlation variables with correlation coefficients greater than 0.6 to make a plot with predictor variable (SalePrice): *Overall.Qual* (0.79), *Gr.Liv.Area* (0.71), *Garage.Cars* (0.65), *Total.Basm.SF* (0.63) and *1st.Flr.SF* (0.63).

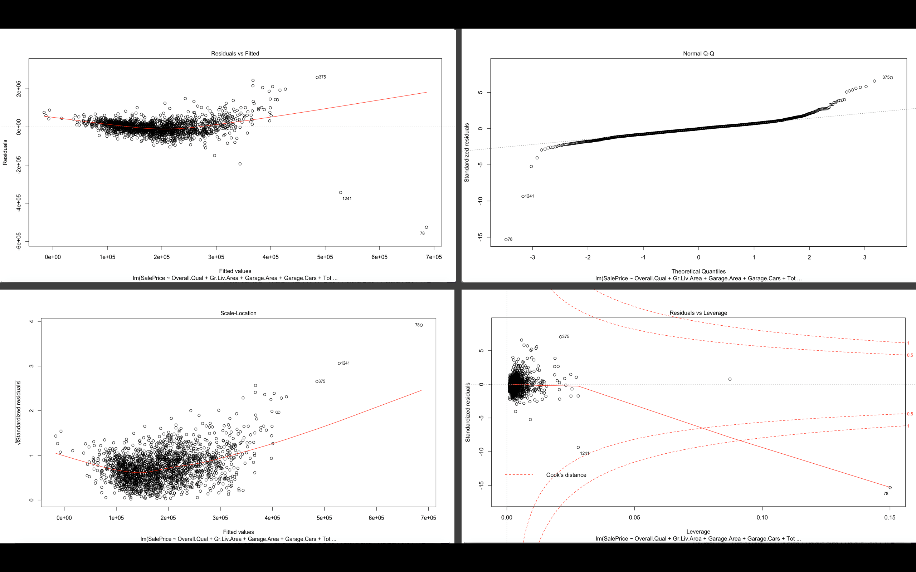


Figure 9 – Results of a linear regression involving 6 predictors

Comparing the two different models side by side, we see that the best performance was given by the model using all predictors, seen in Table 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Regression** | **RMSE** | **R-Squared** | **MAE** |
| All Predictors | 31059.85 | 0.832454 | 19664.5 |
| 6 Predictors | 37654.73 | 0.7757815 | 24221.4 |

Table 1 – Results of the Linear Regression Models

**Support Vector Machine**

To do SVM, the NA removal method outlined earlier in the paper was used, and columns were removed which had more than 150 missing values, which when also removing rows, brought the dataset down to 2,822 rows and 72 columns. This dataset was used to create the data for the first SVM model, with the second SVM dataset came after removing Near Zero Variance columns, reducing it further to 52 columns. Third, the modeler produced backward selection to select variables by using the likelihood ratio test. This model contains 42 variables. The final model was fitted based on the third model; the only difference was we add the Near Zero Variance function into the final one after completing the likelihood ratio test. Then we compare the performance of the four models and find out the final model performed the best.

The full model was produced based on the dataset after we removed all the missing values. We split 75% of the data into training set and 25% into the testing dataset. Then we used the 10-fold cross validation to choose the tuning parameter cost (cost=16) based on the training dataset. To visualize how to choose the tuning parameter, we can look at Figure 10, it tells the darker the color is, the better the performance of SVM would be. In this case the cost should be less than 30 to provide a good model performance. To evaluate the performance of the model, we calculate the RMSE and R-squared based on the testing dataset. Figure 10 shows the code to perform the SVM and the result of the RMSE is 33900.41 and R-squared is approximately 0.8043. The reason why we created a model with the least data preprocessing procedures is because we want to consider this full model as the baseline model that can provide a comparison to identify the improvement of other models.

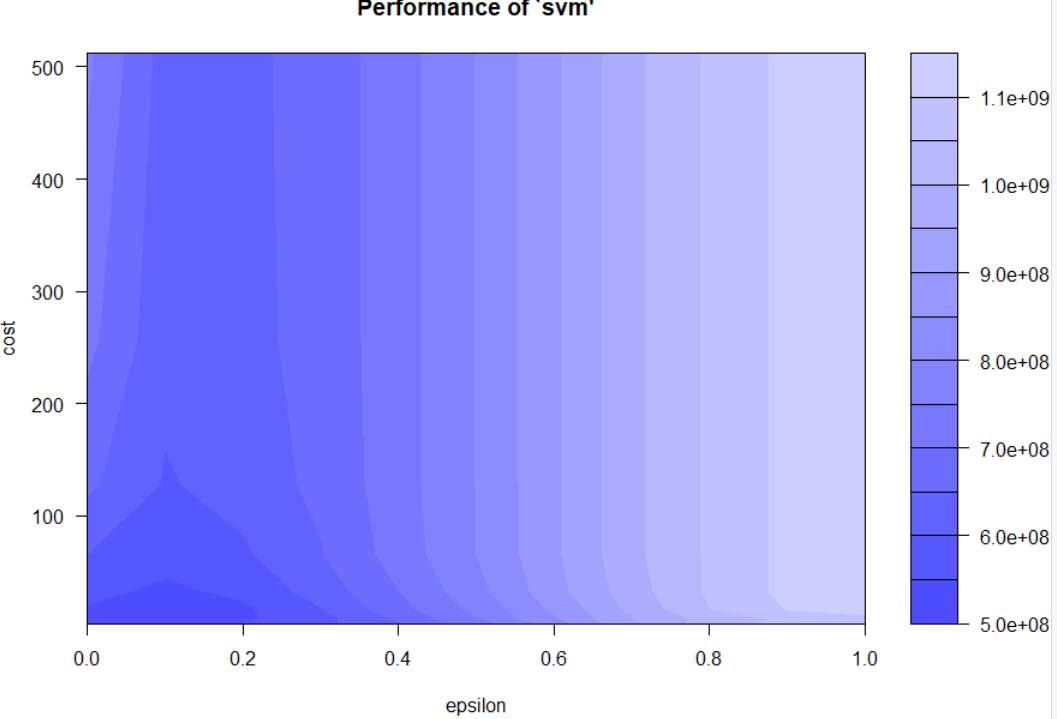
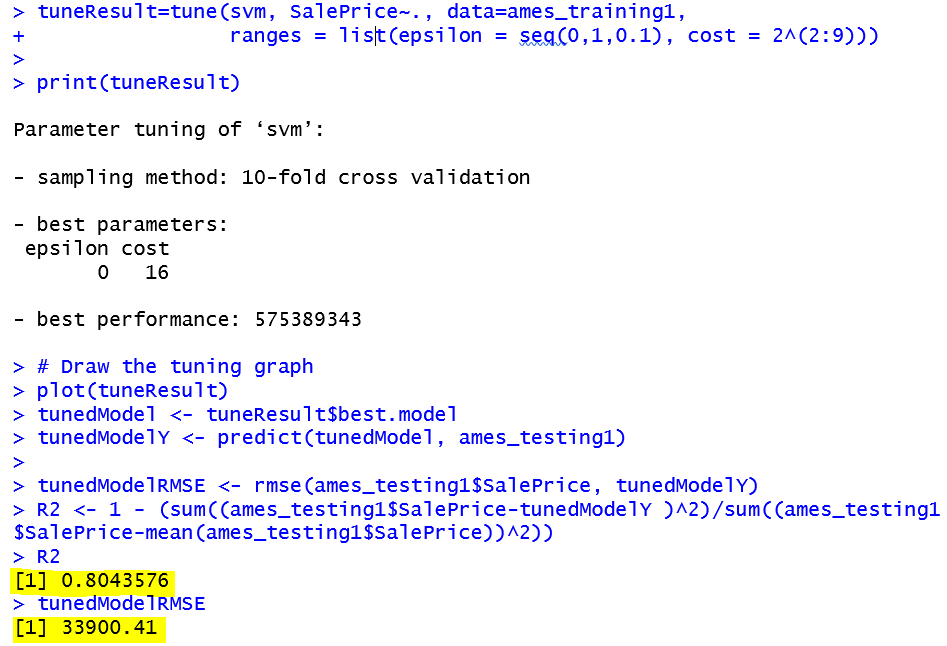
 

Figure 10: SVM performance plot with different values of cost and epsilon (left) and results (right)

The next version of the model involves updating the current dataset by applying the Near Zero Variance function to remove the predictors that have no variance. Our goal is to select variables that contain the most information. This gave us a dataset with 52 variables. We keep the proportion of the training and testing dataset the same as the full model. The 10-fold cross validation gave us the tuning parameter cost is 8. The RMSE is 30410.14, R-squared is 0.8425692.

Another version of the model was then developed. As we know, Lasso and elastic net are typically used for variables selections. However, backward variable selection can do the same job and it is a very straightforward method to help the modeler to understand the relationship between each predictors and response. This method was implemented at the beginning. In our case, we use the likelihood ratio test to test the significance of each variable. Some findings satisfy our intuition. For example, variables like the roof style and the paved drive are not significant to the sale price, but the overall quality is very significant. A common intuition is that the number of bedrooms is supposed to be significant to the sale price, because most people care about the number of bedrooms when purchasing a house. However, our likelihood test result showed that the number of bedrooms is not significant to the sale price.

We use the same data as the full model. First, we fitted a linear regression including all the variables. Then, we fit another linear regression with one variable removed and then we do the likelihood ratio test to test the significance of the variable that we removed in the second linear regression model. When testing the significance of variable *MS.SubClass*, the p value is 0.3421, which is not significant. So, we remove this variable. Similar tests were developed for all the variables. Finally, 42 variables were selected. Based on this dataset, we built a SVM model with same procedure mentioned in previous SVM model procedure. We got the cost is 8 selected by using the cross validation. The RMSE for this model is 30410.14, R-squared is 0.8425692.

The final model is based on the backward selection mentioned in the previous SVM model. Additionally, we removed the Near Zero Variance variables. Then the total number of variables become 30. Cross validation was used to choose the cost based on the training set, which is cost=8. From the testing set, we get the RMSE=28376.17, R-squared=0.8629243.

Of our tests, the last result (Table 2) indicated that the fourth model is the best which removed the near-zero-variance variables and selected the variable by the likelihood ratio test. The full model is worse than others, this can be interpreted as, since it has the most variables, the SVM model may overfit the dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SVM Models** | **Total Variables** | **RMSE** | **R-squared** | **MAE** |
| Full model | 72 | 33900.41 | 0.8043 | 14968.41 |
| Second model | 52 | 30410.14 | 0.8426 | 13945.34 |
| Third model | 42 | 30410.14 | 0.8426 | 13945.34 |
| Final model | 30 | 28376.17 | 0.8629 | 13301.35 |

Table 2 – SVM Model Results

**Multivariate Adaptive Regression Splines**

With their inherent feature selection capability, Multivariate Adaptive Regression Spline (MARS) models are insensitive to predictors with zero or Near Zero Variance. But they still require additional preprocessing to eliminate NA values (whether provided in the original set or created in R to address missing values). We therefore imported the previously imputed dataset with strings not treated as factors to make it easier to manage the necessary transformations. The large number of missing records for *Lot Frontage* (490, or nearly 17%) made removing rows unattractive, and imputing those values might have skewed the results if we did so poorly (refer to the previous section on data imputation). For this reason, the *Lot Frontage* predictor was removed from the data frame used in constructing the model. We also recoded the thousands of NA values in the categorical predictors as “None”. Then we changed those columns from strings to factors and removed rows whose values were missing to reduce the dataset to 2,905 records of 78 predictors (plus the twin identifier columns from the original data). We then divided those records 70-30 into training and test data and used the train function in the caret package to optimize a model with degree = 1 and pruned at 22 predictors (nprune = 22), without preprocessing the data further. This produced a test set RMSE of 28,300.90, an R2 of 0.8734323, and MAE of 17,515.68

**Tuned Trees and Random Forest**

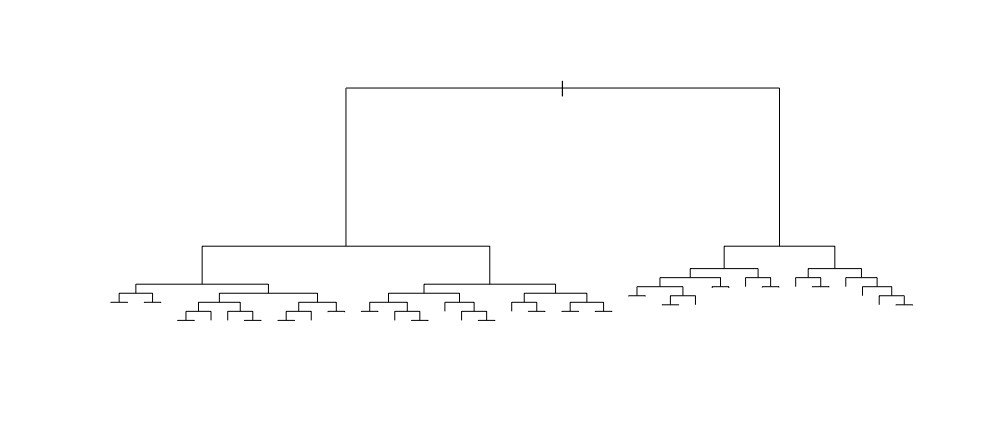
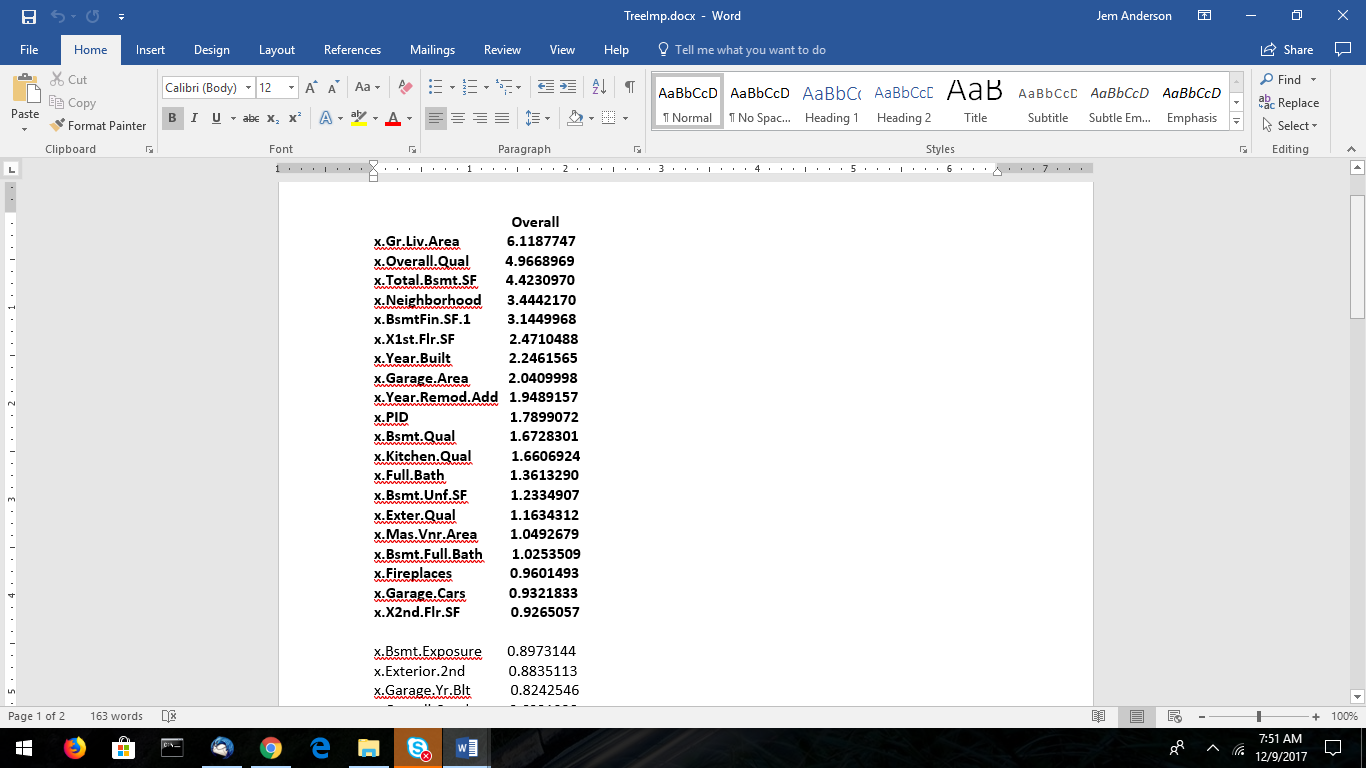
Constructing a tuned tree model starts with loading the dataset into the R environment. The Ames Housing dataset already has headers and requires no additional preprocessing; users of R Studio can set the header radio button to “Yes” and allow strings to be imported as factors.

Figure 11 - The final shape of the Random Forest tree

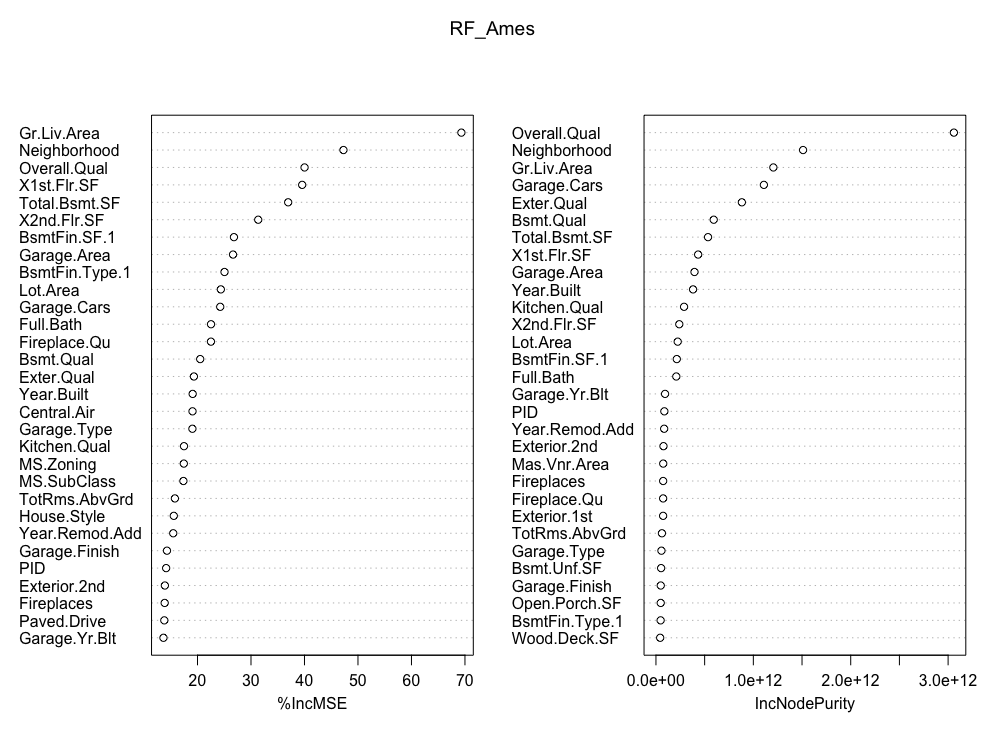
The 2,930 rows were divided in a 70-30 split into training and test data and the train function within the caret R package was used to tune a single tree model for maximum depth, then again for the complexity parameter. Then use the optimal values from those attempts (maximum depth = 11 and complexity parameter = 0.0008581731) and run a tree model using the rpart package against the test dataset to obtain the tree shown in Figure 11 (the tree has too many leaves for the standard plotting packages to provide a presentable visualization if the text is included). The most important predictors in the tuned tree model are shown in Table 3.

The most important predictors are dominated by living space (*Gross Living Area, Basement Square Footage, First Floor Square Footage,* et al.), and quality measures (*Overall Quality, Basement Quality, Kitchen Quality,* et al.), with location (*Neighborhood*) also very important. These are exactly the indicators one might expect to hear from a real estate agent, which causes us to feel comfortable with these results.

As a price predictor, the tuned tree model produced a root-mean squared error of 35,218.47 and an R2 value of 0.7965692. These measures were achieved without preprocessing the data or removing outliers, both of which would be likely to improve the results somewhat.

For random forest models, additional preprocessing is needed to eliminate NA values (whether provided in the original set or created in R to address missing values). We therefore imported the previously imputed dataset with strings *not* treated as factors to make it easier to manage the necessary transformations. The large number of missing records for *Lot Frontage* (490, or nearly 17%) made removing rows unattractive, and imputing those values might have skewed the results if we did so poorly (refer to the previous section on data imputation). For this reason, the *Lot Frontage* predictor was removed from the data frame used in constructing the model. We also recoded the thousands of NA values in the categorical predictors as “None”. Then we changed those columns from strings to factors and removed rows whose values were missing to reduce the dataset to 2,905 records of 78 predictors (plus the twin identifier columns from the original data). We then built test (30%) and training (70%) sets and built random forest models with 500 and 1,500 trees and the number of variables tried at each split at the default value (mtry=26).

Table 3 – Tuned Tree Variable Importance

Both models captured over 90% of the variance (90.05% and 90.17%, respectively) and provided good results. For 1,500 trees, we achieved a root-mean squared error of 23,333.95 and R2 = 0.9223925. Figure 12 is a plot of the most important predictors, by incremental RMSE and purity.

We might have obtained somewhat better results by using the caret package train function to optimize the number of variables tried at each split.

Figure 12. Variable Importance for Random Forest Model, ntree = 1500, mtry = 26

**KNN CLASSIFICATION:**

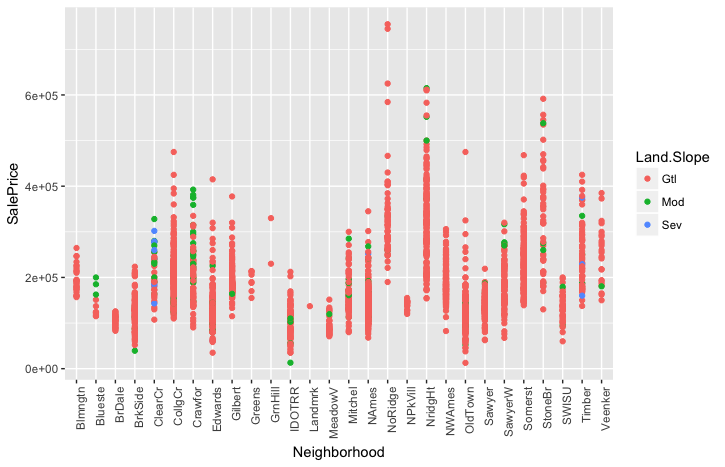
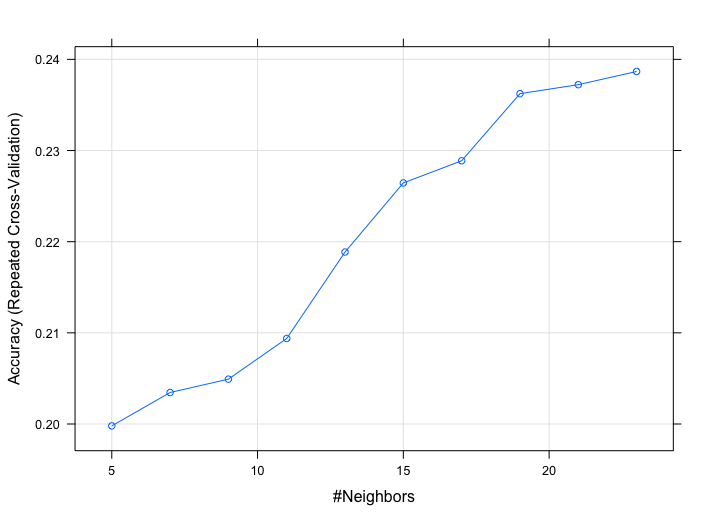
KNN (K Nearest neighbor) model was performed for classification analysis. Throughout the project, we have considered the *SalePrice* as the target variable but here, since it’s a used as a classification model, we selected *Land.Slope* as our predictor variable. The analysis that we wanted to make was to show if we could predict the classes of the target variable using the data. Figure 13 shows the distribution of *SalePrice* for each *Neighborhood* with relation to the *Land.Slope*.

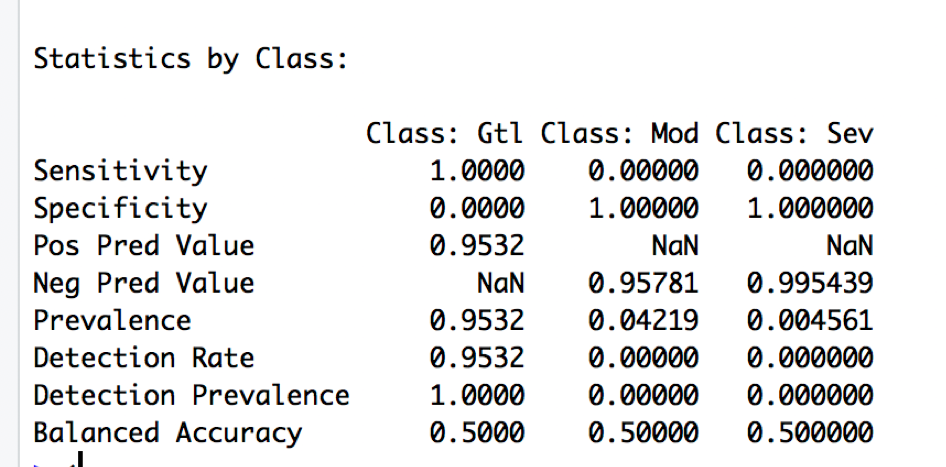
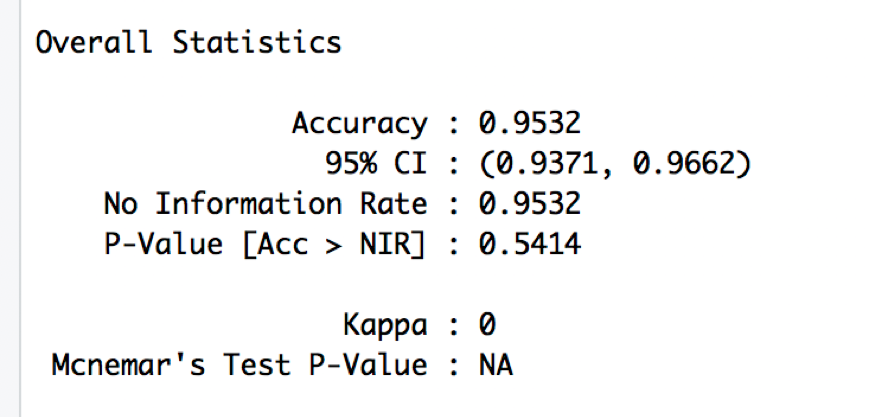
Figure 13 – The distribution of SalePrice for each Neighborhood in relation to the Land.Slope

The *Land.Slope* variable has three classes: Gtl (gentle slope), Mod (moderate slope), and Sev (severe slope). From the Figure 13, we can say that the number of houses situated on severe and moderate slopes are very less compared to the houses situated on the gentle slope.

After fitting the KNN model, we have obtained the k-value with the highest accuracy as 23. (Figure 14).

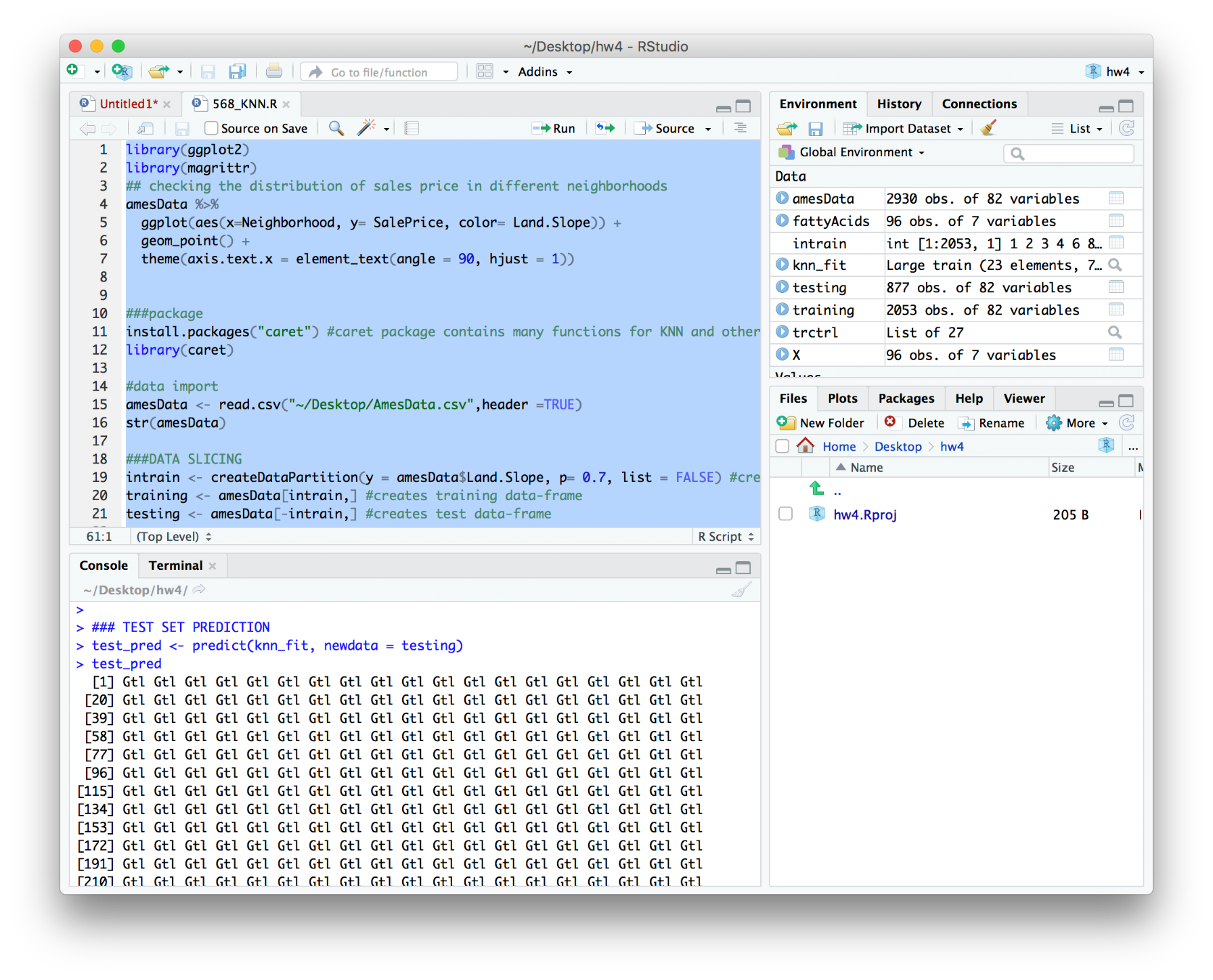
The test predictions show a good overall accuracy of 95.3% and a p-value of 0.54. (Figure 15) However, the sensitivity and specificity values are extremely high and zero for the first class and are reversed for the other two classes. This could be due to the class imbalance in the dataset. (Figure 15)

Figure 14 – K Values for Accuracy



*Figure 15 – KNN test predictions*

The test prediction results showed almost accurate predictions, (only 7 predicted incorrectly). The KNN classifier can be applied to this data in order to predict the classes of the *Land.Slope*. (Figure 16).



*Figure 16 – Sample KNN Output code*

**Model Comparison**

Since we are Predicting prices which are mostly hundreds of thousands of dollars, the error values are presented as dollars, to show how far off our predictions were. Random forest did best (Table 4).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | | **RMSE** | | **R-squared** | | **MAE** | |
| PLS | | $35,594.59 | | 0.8054 | | $22,950.13 | |
| Linear regression | | $31,059.85 | | 0.8324 | | $19,664.50 | |
| MARS model | | $28,300.90 | | 0.8734 | | $17,515.68 | |
| Tuned tree | | $35,218.37 | | 0.7965 | | $23,289.59 | |
| Random forest | | $23,333.95 | | 0.9223 | | $15,266.10 | |
| SVM | | $28,376.17 | | 0.8629 | | $13,301.35 | |
| **Model** | **Accuracy** | | **P-value** | | **Positive  prediction values** | | **Negative prediction values** |
| KNN model | 0.9532 | | 0.5414 | | 0.9532 | | 0.95781 |

*Table 4 – The best results for each model*

1. House Prices: Advanced Regression Techniques, accessed 10/1/17 online (https://www.kaggle.com/c/house-prices-advanced-regression-techniques) [↑](#footnote-ref-1)
2. De Cock, Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. Journal of Statistics Education. (11/01/2011) , 19 (3), accessed 09/27/2017 online (https://ww2.amstat.org/publications/jse/v19n3/decock.pdf) [↑](#footnote-ref-2)