House Prices: Advanced Regression Techniques

Machine Learning

October 14, 2021

Carly Leong

Kaggle ID: carlyleong

Best Score: .12252

Section One: Introduction

The purpose of this assignment was to predict final sale prices of houses from the data set that was given. Kaggle gave a train and test data set and our job was to clean and create models by coding in R to predict the House prices of the test data set. Once we created our models and predicted the house prices, you submit your predictions to Kaggle and you are given RMSE score. The lower the error score, the more accurate your predictions are.

1.1 Original Data Set

The Original data set has 79 explanatory variables describing residential homes in Ames, Iowa.

1.2 Cleaning of the Data Set

I first read the data set from Kaggle. In order to clean, I combined both test and datasets to make sure I got rid of all NA's and applied all of my functions onto both data sets. I read all strings as factors in as True to work with a more universal data set when cleaning. My rationale for cleaning/ exploration was to go column by column for:

- getting rid/inputting of NA's
- adding additional columns
- plotting the linear correlation between a feature and the sale price to see its relevance
- creating ordered levels
- Grouping (if needed)

1.3 Elimination of Variables

Before cleaning any NAs, I explored all of the features by plotting them with Sale price and seeing what kind of correlation they have. My criteria for elimination of variables:

- External research to see how these features affect the Sale Price
- Kaggle's website: If a feature had a level that was 95% >, I would just eliminate that variable, due to the fact that it is unnecessary.
- If any variables showed obvious multicollinearity, I got rid of one of those variables
- Ggplot to see correlation

Before doing any cleaning and plotting, based off of research, theory, and Kaggle's website, I got rid of LotFrontage, Alley, Utilities, YearBuilt, MoSold, Street, Condition2, RoofMatl, MasVnrArea, Heating, GarageYrBlt, PavedDrive, Exterior2nd.

Section Two: Column by Column Breakdown

2.1 Introduction

For dealing with the factored columns, I went column by column doing my data cleaning, transformations, factoring, and data manipulation. This way was more organized and I could keep track better. I then went column by column when dealing with outliers in a separate section. I had the same process and criteria when dealing with rating and grouping factors. Below, are those processes.

2.2 Rating Factor Columns Process

Criteria for "rating columns"

- Column has a very clear ordinal rating
- Ex. (Poor, fair, typical, good, ex)

How I rated them

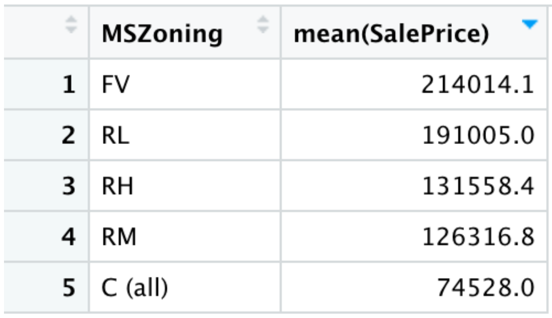
- Created a rating based on the descriptions from the text with the lowest rating being 1
- I ordered the factor columns based on the rating
- After ordering, I converted them into numeric
    - Ex. Poor-1, fair- 2 …. Excellent would be 5
- Made ratings consistent for all factor ordering; the lowest number would correlate to the lowest level that had the lowest average sale price

Dealing with NA's

- When creating the rating, I would create a level for NA's that was called something else
- Ex. for BsmtQual, I added the level "NB" for NA
- I would rate the NA level as the lowest
- After, I would input the NA's as NB's because now there is a level for the NA's to go into

<u>2.3</u> <u>Process for Grouping and Leveling Factors</u>

1) Look at the Kaggle statistics of the column, I asked myself:
    a) Is the data over 90 percent of the same?
    b) How is the data broken up, what are the percentages of each level?
2) Do external research on each of the levels:
    a) What points should be grouped together?
    b) What is the highest quality material from research?
2) Did analysis on each of the levels and their average sale price
    a) Created a table ranking them (see example below)
    b) Created a ggplot
3) Once levels are ranked, I grouped them into a new level (if necessary, based on above points)
4) Lastly, from the above information, I would order them by their new levels



	MSZoning	mean(SalePrice)
1	FV	214014.1
2	RL	191005.0
3	RH	131558.4
4	RM	126316.8
5	C (all)	74528.0

Figure 1: example of exploring the data

<u>2.4 Column by Column Cleaning, Transformations, and Manipulations</u>

MS Zoning

- 4 NA Values found from filtering

- Inputted similar values from similar houses with similar features for the NA's

ID Column
- Used for inputting purposes for MS Zoning
- Deleted column

Neighborhood
- Reduced levels from 25-12
  - Ranked average sale price of each neighborhood
  - Grouped similar levels in pairs (one level with 3) and assigned each level to a number (1-12) with 1 being the lowest average sale price
  - Total levels: 12
  - By average sale price of each level, I ordered the levels from 1:12

Condition One
- Reduced levels from 8 → 3
- Grouped by theory
  - positive attributes
  - Railroads
  - Other
- Did not order because factor was not ordinal

BldgType
- Reduced levels from 5 → 3
  - Grouped by theory/ similar features
    - All townhouses
    - All family
    - Left "1fam" alone
- Ordered levels because average sale of each of the grouped levels price drew a correlation

Housestyle
- Reduced levels from 8 → 3
  - Grouped by theory/ similar features
    - Combined 1 and 1.5, 2 and 2.5
    - Rest was "Other"
- Ordered levels because average sale of each of the grouped levels price drew a correlation

Roof Style
- Reduced levels from 6 → 2
  - Grouped by theory/ similar features
  - Did not order

Exterior1st and Exterior2nd

- Filtered NA out, explored the NA value and inputted value based on similar houses with similar features
- Reduced levels from 13 → 2
- Created analysis of grouping and ranking values by average sale price
    - Grouped levels from above analysis
    - Created two levels of the upper half and the bottom half of ranking

MasVnrType
- Filtered NA out, explored the 24 NA values
    - Replaced NA with "none"
- Reduced levels from 4 → 2
    - Grouped all of the "brick" types together
    - All others were "Other"
    - Grouping determined by external research

ExterQual and ExterCond
- See "Rating Factor Columns Process section"

Foundation
- Reduced levels from 6 → 2
    - Used external research for grouping
    - Two groups: older and newer foundation
- Ordered levels based on external research

BsmtQual, BsmtCond, BsmtExposure
- See "Rating Factor Columns Process"
- Deleted BsmtCond due to kaggle analysis of data being over 95 percent the same

BsmtFinType1, BsmtFinSF1, BsmtFinType2, BsmtFinSF2, BsmtUnfSF
- See "Rating Factor Columns Process"
- Once ordered factors are converted into numeric, I took the average of BsmtFinType1 and BsmtFinType2
    - After, I deleted BsmtFinType2 to prevent multicolliniarity
- The NA's in these columns correlated to no basement, therefore, I inputted 0 for all of the NA's

HeatingQC
- See "Rating Factor Columns Process section"

CentralAir
- See "Rating Factor Columns Process section"

Electrical
- Reduced levels from 5 → 2
    - Most data was "SBrkr"
    - All four levels were combined into "other"

- Ordered levels based of average sale price

BsmtFullBath, BsmtHalfBath, FullBath, HalfBath
- Straight forward data
- Inputted NA's with zero because no bathroom = 0, column already a numeric

KitchenQual
- See "Rating Factor Columns Process section"

Functional
- Reduced levels from 8 → 2
  - Most of the data was "typ"
  - Those levels that are anything besides typical are "other"
- Ordered by "Other" , "Typical"

Fireplaces and FireplaceQu
- See "Rating Factor Columns Process"

GarageFinish
- Reduced levels from 4 → 2
  - Grouped by finished and anything that is not finished (other_
- Inputted NA's as "other"
- Rated from "other" to "fin"

GarageType
- Reduced levels from 5 → 4
  - Based on the data, there was not enough data from "basement" and "carport"
    - Combined basement and carport as the same level as they also have similar average sale prices
- Ordered levels based on external research and average sale price of each level

Garage Qual and Cond
- See "Rating Factor Columns Process section"

Garage Cars and Area
- Inputted NA's with 0 because no garage is equal to 0

PoolArea and PoolQC
- Reduced to two levels; 1- no pool and 2- pool
  - Any house that was given a rating was grouped into the level, "pool"
    - If the house had (fa, ta, gd, ex)
  - Any others, including NA's was a "NP" no pool
- Ordered from no pool, pool
- Pool Area was deleting based on ggplot with sale price; no correlation

Fence

- Similar process to pool, but it was 1- no fence, 2- fence
    - Any rating that was given was classified as fence
    - All others including NA was classified as "NF" no fence
- Ordered from no fence to fence

MiscFeature
- Deleted column
    - Plotted with Sale price, show no correlation

MiscVal
- Deleted column
    - Plotted with Sale price, show no correlation

SaleType
- Reduced levels from 9 → 2
- Did external research to understand what the sale types were
    - Based on external research, I grouped all of the "warranties" together
    - The rest were "other"
- Did not order because factors are not ordinal

SaleCondition
- Reduced levels from 7 → 2
    - Did external research
    - Most of the column was "normal"
    - Anything other than "normal" was classified as "other"
- Ordered levels from "other" to "normal" based on average sale price

Open Porch
- Inputted the NA's with the mean

2.4 Cleaning Summary

Original # of Columns	Columns Deleted	Columns Added	# Columns Left
79	21	1	59

Section Three: Outliers

3.1 Outlier Process

1) Used ggplot to plot out the variable and the average sale price
    a) If ggplot shows no correlation, would delete the column
2) Would spot the outliers
    a) Would also look for if there are many 0's and determine to input those zeroes with the mean
3) Create a mean of all of the data including the outliers
4) Created a loop to get rid of outliers based off of number found in the ggplot
5) In some instances when there are many zeroes
6) Would input those zeroes with the mean
7) Depending on the ggplot exploration, I either;

a) Deleted the column
b) Used a loop to get rid of outliers and inputting the mean
c) Inputted mean for zeroes
d) Used both the loop and the inputted mean for the zeros

3.2 Columns in Outlier Process

The columns that I used in the outlier process were LotArea, BsmtUnfSF, TotalBsmtSF, GrLivArea, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, x3SsnPorch, ScreenPorch. See example below in Figure 2 of the outlier transformation of total bsmt SF. As you can see, the manipulation of the outlier caused the graph to become increasingly linear.

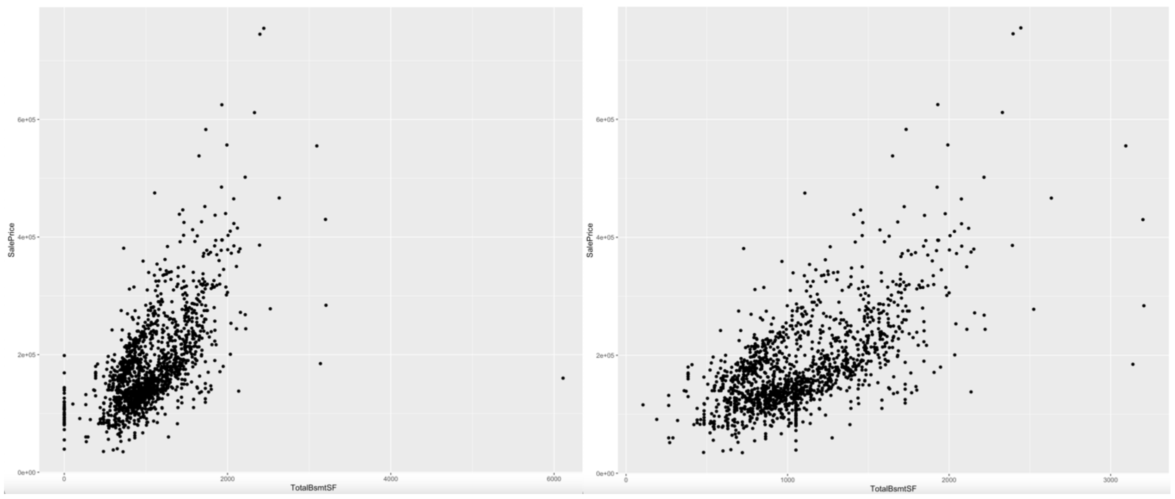


Figure 2: Total BSMT SF, before (left) and after (right) outlier transformation

Section Four: Cleaning /Outlier Errors and Problems

I ran into many minor issues when coding. However, the below two issues are the biggest issues that I faced when cleaning and checking for outliers.

4.1 Issue #1

I struggled with reading in the data with stringsAsfactors = TRUE or FALSE. I started at False, but I quickly realized that it was hard to clean the data and then convert back to a factor for every single column. It was harder to explore the data. I read the data in as TRUE after I ran into this issue. This helped with my organization with all variables being more universal and I was able to apply NA cleaning, grouping, data manipulation, and releveling processes much easier.

4.2 Issue #2

I came into most of my issues while trying to run my first plain OLS validation model. My biggest error was mismatching levels due to not having enough data in each level. I solved this issue by going back to the data cleaning (see "process for grouping and leveling factors section above). The reduction of levels by grouping allowed there to be enough data in each level to be able to create models from.

Section Five: Modeling

5.1 Introduction

In order to model, I transformed the Sale price of the train set by logging it. I did this because it creates increased linearity which will make the modeling more accurate. For the final predictions, I transformed the predicted sale prices back into the normal form by exponential transformation.

In total, I ran 15 models that gave final predictions that were submitted to Kaggle. I ran 9 validation models on the models that could be "tweaked" or adjusted where the validation models were necessary. I excluded validation models from Feature selection, SVM, Bagging, and Boosting. These models did not need the validation models. My best Kaggle score resulted from the cleaned OLS model (.12252)

5.2 Plain OLS Models

I ran two OLS models; one before removing the outliers, and one after removing the outliers. The results were:

Plain OLS Validation RMSE: 0.009693446
Cleaned OLS Validation RMSE: 0.008510748

Plain OLS Final Prediction RMSE: 0.12805
Cleaned OLS Final Prediction RMSE: 0.12252

As stated in the introduction, the cleaned OLS model gave me my best score at .123. The OLS model gave me my best score because of the way I cleaned my data. When re leveling and ordering, I based it off of the rank of average sale prices. Additionally, I reduced my levels and grouped data together. For the outliers and multiple zeros, I replaced them with the mean. Most of my data was generalized, therefore, that is why my OLS model worked the best.

5.3 Feature Selection

I did not create a validation model for feature selection because there is nothing to change in the model that needs to change.

Backward RSME: 0.13699
Forward RSME: 0.1374
Hybrid RSME: 0.13699

Feature selection was one of my lower RMSE scores. This is because I emphasized my variables in the data cleaning and which matches the goal of feature selection.

5.4 Ridge, Lasso, Elastic Net

I created a validation for this set because the model could be tweaked based on the validation, specifically elastic net. For the elastic net, I used three different alphas (0.25, 0.50, 0.75). Based on the best alpha that created the lowest error in the validation model, I used that alpha for the final prediction test set for elastic net.

Ridge validation RSME: 0.04303409
Lasso Validation RSME: 0.04301085

Elastic Net validation RSME and best alpha: 0.04301743 , 0.75

Ridge final prediction RSME: 0.13253
Lasso final prediction RSME: 0.13264
Elastic Net final prediction RSME: 0.13234

Ridge, Lasso, and Elastic net were also one of my lower RMSE scores. This can be attributed to the fact that they are very similar to the OLS function, but with different lambdas and alphas.

5.5 SVM
I did not create a validation model for this model because it did not need to be tweaked. I only selected columns that were numeric and factors that were ordinal. I ran into a small error when running the model, I would just be getting null. I solved this issue by inputting zero for the NA sale price in the test set.

Linear kernel final prediction RSME: 0.13184
Radial kernel final prediction RSME: 0.13789
Polynomial kernel final prediction RSME: 0.37838

It can be explained that the linear kernel model has the lowest RMSE score of the SVM models because my data is catered to be more linear.

5.6 Simple Tree, Random Forest, Bagging
For Simple Trees and Random Forests, I created a validation set. Additionally, I also created a model to choose variables by importance for both of them

Simple Tree validation RMSE: 0.01665549
Simple Tree variables chosen by importance validation RMSE: 0.0166752
Random Forest validation RMSE: 0.0111934
Random Forest variables chosen by importance validation RMSE: 0.01121044

Simple Tree final RMSE: 0.22601
Random Forest final RMSE: 0.13662
Random Forest variables chosen by importance final RMSE: 0.13722
Bagging final RMSE: 0.14012

The simple tree had a high RSME score because it does not take into account all of the variables in a linear fashion.

5.7 Boosting
I did not create a validation model for boosting because it did not need any changes.
Boosting final RMSE: 0.14496

Section Five: Discussion

In the figure below, my best models were the Clean OLS, Plain OLS, SVM: Linear. This makes sense because they are the most plain linear models out of all of the models. My data worked well this this model because I manipulated and transformed my data to be linear. I ordered most of my factors in the data, based on external research, and data exploring.

In terms of issues and errors with the models, I ran into most of my errors with the SVM model. The SVM model only take numerical and ordinal factors. I had to carefully select specific features that met that criterion. Additionally, the SVM model would not run on the test set that contained the NA's for the sale price. I solved this by inputting 0 for all of the NA's so the model would run.

Models	**RMSE**
Clean OLS	0.12252
Plain OLS	0.12805
SVM: Linear	0.13184

Figure 3: Top Three RSME Scores

<u>Section Six: Conclusion</u>

Overall, I learned a great amount from doing this assignment, both technically and personally. In terms of working with data, the most important concept that I learned was how you clean, manipulate, and transform your data. All of your models are based on your data, so if your data is not good, your model will not be good. Once I went back to cleaning my data which involved re ordering, transforming, rating, grouping, inputting, I noticed that my model improved significantly. Most of my models worked perfectly after I went back and cleaned all of my data. This shows the importance of a clean data set. I also learned the importance of external research. I replied heavily on external research, especially to do grouping and rating of levels. For some of the columns, I didn't even know what those features meant and how they even factor into how they affect the sale price. I encountered other coding issues and was able to solve them by communicating with professors and researching on the internet (see above in "errors" section).

My overall experience of this assignment was that I am capable of working with data and running models on data. I became much more advanced in coding with R, and problem solving on my own using research online.