**Name: Dylan Dehond**

**Report Name:** Kaggle Competition – Housing Prices Competition for Kaggle Learn Users

**Kaggle Name:** Dylan De

Competition Link: <https://www.kaggle.com/competitions/home-data-for-ml-course/overview>

**Total number of Teams on Leaderboard:** 36,499

**Position on Leaderboard:** 459 (Top Percentile)

A picture containing timeline

Description automatically generated

**Graphical user interface, text, application, email

Description automatically generated**

**Data Exploration – Initial Discovery**

As someone who is interested in real estate and pays attention to housing prices — specifically Toronto, I chose to attempt this Housing Price Competition.

I began this regression model building exercise with some few preconceived notions about the dataset. Specifically, around independent variable collinearity and independent variable predictive power on the dependent variable (Sale Price). For example, I thought average sale price would increase over time; hence I assumed the year variable could be a valuable multiplier variable when trying to engineer new features. I also thought neighbourhood would be a key variable because everyone knows, Location! Location! Location!

For context, I used R for all aspects of this regression building exercise. You will notice at the top of my code I have a ton of libraries. I tend to pull forward the same initial set up when starting a new project in R (lines 1-14). I am a library hoarder so the list of libraries will continue to grow over time until someone informs me why I should not do continue this practice 😉.

After importing both train and test datasets. I performed data exploration to familiarize myself with the datasets. I ran the “nrow()” function on each dataset to get a bearing of how many records I would have to work with.

I was excited about testing my assumptions, so I created a line chart that showed avg. sale price by year. I did this by grouping the data by year, summarising sale price into a mean value and then using “ggplot()” to visualize the data. Through this line chart I saw the range of data for the train set was, house sales between 2006 and 2010. I was surprised to see that the average sale price decreased by about 6% from 2007 to 2008 and never recovered. It then clicked that in 2008 the US housing crisis occurred and since this dataset was from houses sold in Ames, Iowa a negative impact to housing prices should be expected to an extent. With this information I engineered a dummy variable that would indicate whether the house was sold pre-crisis or post-crisis. When testing this variable in my regression models this variable did not have significance in predicting sale price, so I abandoned it. Since year sold as an integer would not work out well given the non-linear trend of housing prices over time that I was seeing, I treated it as a factor rather than an integer. I then quantified that year sold did not have significance to the extent I was expecting so I abandoned my focus on that variable. While exploring these variables, I also created some histograms using the year sold variable and sale price variable to see if these familiar variables could point out any skew or abnormalities in the dataset — I did not see anything insightful through this effort.

I moved on to plotting sale price by neighbourhood and saw that while there are a lot of neighbourhoods close in average sale price, some neighbourhoods have a marginally higher average sale price; hence, I thought this variable would have some significance when trying to predict house prices that reside in the upper or lower pricing tiers. I also ran a histogram to look at the relationship between lot size and sales price. While I could make out a correlation, the outliers made the chart difficult to look at. I then attempted to create a correlation matrix, to quantify some of the relationships between Sale Price and the independent variables; however, due to missing data I was unable to create this matrix, so I had to first address the missing data.

**Data Wrangling**

I used the “miss\_var\_summary()” function to gather an understanding of the missing variables in both the train and test sets. Seeing that there was missing data in both datasets, and variables that had missing values seemed to be consistent across both train and test datasets I combined the datasets for this data wrangling effort using the “rbind()” function. I first had to create a SalePrice column for the test set and insert NA values in this new column, so that the dataset columns matched.

When initially importing the data from the csv files I converted strings to factors and when running the “str()” function to get an understanding of data type I noticed there were some variable “type integer”, which needed changing to “type factor”. To expedite this effort, I created a vector that contained all variables requiring Type conversion and used an “lapply()” function to loop through the vector applying the factor conversion. Again, I made the decision to have date variables “type factor” as the relationship between the date variables and sale price was non-linear. I did not make this decision to convert all these variables right off the start, it was more of an iterative process, as I went about building my model and reviewing summary statistics.

**Data Wrangling – Missing Data**

At first, I was overwhelmed by all the missing data. I began by looking at missing values for “Factor Type” variables and quickly realized that the majority of this missing data exists because many “Factor Type” variables are features of the house/property, and some houses/properties do not have these features, so an NA value was used. I also corroborated this by looking at the data description text file provided — in retrospect I should have given this data description text file a more thorough read through. I decided to change all these NA values to “None” so that I could differentiate between actual missing data easier.

For each factor variable column with NAs I used an “ifelse()” function with an embedded “is.na()” with TRUE returning “None”; however, the output was actually changing all the other variables to integers. I resolved this by nesting the above statement in a “factor()” function.

Throughout this data wrangling effort I had set up a simple linear regression where I was plugging in and running different combinations of independent variables I thought would have relevance (which I will speak to in a bit). Specifically, I felt MSZoning would have significance as the elements in this variable highlight key distinctions in the property sold. The MSZoning categorical variable identifies the general zoning classification of the sale. For ex. It specifies whether property sold was agriculture, commercial, medium-density residential, etc.…. When using this variable in my initially linear regression I saw that it had significance (p-value <5% for many of the elements), but when trying to predict on the test set the missing elements were causing issues. There were only 4 missing values for this variable, so I used the same “ifelse()” approach above, but this time I replaced “None” with the mode element as about 78% of records held the mode element. I tried using Multiple Imputation techniques, which I will speak to in a bit, but essentially that did not work out too well, so being diligent of the effort/reward trade-off I took the above approach.

The missing integer variable values were minor, except for Lot Frontage, which had 486 missing values (~17%); however, you cannot see correlations when running a correlation matrix if the variable has missing values. I felt that Lot Frontage was a key variable as it specifies a key dimension of the property and is a commonly advertised feature when buying/selling property. I ended up replacing missing values with the mean for the respective variable using the aforementioned “ifelse()” function.

**Data Wrangling – Missing Data – Multiple Imputation**

Throughout my data wrangling efforts, I had made a few attempts to replace missing values using multiple imputation, but due to the large number of variables, differing types (integer and factor), data wrangling iteration, and long run-times I eventually abandoned the effort in favour of discretionary imputation (using my judgement). I was not overly concerned about this, as the missing data was minor, nor did I feel it would materially impact the variables that had significance. When running multiple imputation I installed the “naniar” package, and used the “mice()” function. I then used the “complete()” function and used one of the imputed datasets as my new completed datasets. I used the “meth=’cart’” as it worked well with the mix of factor and integer missing values.

**Data Exploration – Correlation Matrix**

After I had addressed the missing values I was able to run a correlation matrix using “cor()” to get a better understanding of my integer variables and the dependent. Focusing on correlation between the dependant and independent variables, what stood out were strong correlations with “OverallQual” 0.8, living area variables and bathrooms.

Table

Description automatically generated

**Feature Engineering**

Seeing that square footage variables for different floors of the property all had mid to top tier correlations in this grouping and that generally total square footage is a key property feature I decided to create a new variable that calculated total square footage for the house/building. I created an integer variable called “TotSF” which is the sum of “GrLivArea”, “X1stFlrSF”, and “X2ndFlrSF”. When running an updated correlation matrix I saw that the “TotSF” variable has a 0.8 positive correlation, which is an improvement from “TotalBsmtSF” 0.6 and “X1stFlrSF” 0.6. I decided to use this same approach for bathrooms and bedrooms. For bathrooms I combined “FullBath” 0.6 and “HalfBath” 0.3 variables to create “TotBath” by multiplying HalfBath by 0.5 and then adding that to FullBath. Unfortunately, this new variable “TotBath” returned a seemingly unimproved positive correlation of 0.6, but since in my eyes total bathrooms is a key feature, I kept this new variable (I later realized I was missing “BsmtHalfBath, but my model had already scored well so I didn’t make the update). Finally, I was going to engineer a total bedroom’s feature, but I realized that “BedroomAbvGr” was only 0.3 and I did not see a below ground bedroom variable, so it wasn’t possible.

Table

Description automatically generated

**Cross-Fold Validation**

I split my combined dataset back into the original train and test sets after my data was cleaned and features engineered.

**Modelling — Linear Regression – All Variables – Early Days**

I was keen on building my model using the TTT approach; hence including all variables, and then eliminating variables as I work towards optimizing my model. The idea was to set up this model using “~.” To include all independent variables and then remove variables using “-*InsertVariableName*”, all while cleaning my dataset, feature engineering, and testing various interactions.

I initially struggled with this approach. My “lm()” regression model code was constantly returning errors, due to missing values, factors not having enough elements, and others errors I didn’t fully grasp at the time. I initially pushed forward and tried to create a model using variables of interest that were not causing errors, but the adjusted r-squared of my model was hovering around .780-.820 range and when trying to predict sale price on my test set, it was outputting NAs in rows wherever there was a missing independent variable. At this time, I was focused on outputting and submitting a prediction to Kaggle so that I could have an initial benchmark, which I could then iterate on by testing different interactions, tweaking features, and applying transformations. As a result, I did a lot of patch work just to get a model running that did not have NAs. These earlier submissions yielded poor results, hovering around the 50th percentile. Further interaction and variable inclusion/exclusion tweaks was not showing material improvements in model performance either. This led me towards a renewed focus on cleaning and transforming my dataset thoroughly.

As you will see in my code, for initial regression modelling efforts, I didn’t utilize cross-fold validation, as I was focused on getting my model to run without any errors, once my data was cleaned properly, I then utilized cross-fold validation which allowed me to gauge model performance without having to submit to Kaggle.

**Modelling — Linear Regression – All Variables – After proper data wrangling efforts**

Once I had cleaned my dataset more thoroughly, I was finally able to create and run a linear regression model using “~.” “reg.norm.all.1 <- lm(SalePrice ~., train\_data.1)”. I was pleasantly surprised to see an adjusted r-squared value of 0.9146, plus now that I had all variables in my model, I had a great view of their coefficients and significance (p-values). Looked like the extra data wrangling effort paid off. However, when using the “plot()” function to run diagnostics on the model I noticed some issues. First off, In the upper and lower quantiles of the “Normal Q-Q” plot, the standardized residuals resembled an exponential distribution; if the distribution is normal, I would expect it to show a relatively straight line. I also saw that my “Scale-Location” plot had a big check mark looking relationship, and finally my “Residuals vs Leverage” plot had some outliers. As a starting point, I decided to address the exponential distribution concerns by applying a log transformation to my dependant variable. This transformation seemed to improve some of the skew I was seeing in my “Normal Q-Q” plot.

This log transformation also improved the models RSE from 23,210 down to 0.1082 (I suppose it is all relative though), and the adjusted r-squared from .9146 to .9266. I submitted this model and saw that I had jumped all the way to the 11th percentile on the leaderboard. I knew I was on the right track now.

**Modelling – Linear Regression — All Variables – Utilizing Cross-Fold Validation**

At this point, it seemed like I had finally gotten past the time-consuming hump that is data wrangling, so I was excited to begin applying techniques for optimizing my regression model without running into data issues at every turn.

I utilized cross-fold validation by splitting up my train dataset (70-30) so that I could receive feedback on the performance of my model without having to submit directly to Kaggle. I used MAPE (Mean Average Percentage Error) to test model performance.

Since the model that I discussed above scored well, I ran it again using the cross-fold train set and achieved an adjusted r-squared of .9234 and a MAPE of 6.8%.

**Modelling – Stepwise Regression – Utilizing Cross-Fold Validation**

I again used the aforementioned model for stepwise regression. I tested all three stepwise directions.

Results as follows:

|  |  |  |
| --- | --- | --- |
| **Direction** | **Adjusted R-Squared** | **MAPE** |
| Forward | .9234 | 6.818% |
| Both | .9217 | 7.66% |
| Backward | .9215 | 7.69% |

The forward direction stepwise regression performed the best, but when attempting to run it on the dataset I kept receiving errors likely due to certain variables in the test-set having elements that were not present in the train-set variable. Since I already achieved a 6.8% MAPE score without stepwise and felt the effort of troubleshooting this issue was not worth the likely minor improvement in model performance, I decided to move forward with submitting the “direction = both” stepwise regression as I was able to make predictions on the test-set using this model without any issues.

This model ended up yielding the best score on Kaggle, which is surprising given that the MAPE and adjusted r-squared is worse than the model that did not utilize stepwise. I scored in the 4th percentile per the screenshot on the front page of my report.

Top Scoring Model Code: reg.log.cross.step.1 <- step(lm(log(SalePrice) ~. , cross\_fold.train), direction ="both")

I later tried training the forward stepwise regression with the full train-set, which ran without error. I was disappointed to see that my Kaggle submission using this model performed poorly, which was surprising given that I would think using more data to train a model would yield better results. I presume on average this would be the case, but maybe in this specific situation the first 1000 records that I used to train my model in the cross-fold aligned better with the test-set than the cross-fold test set. Using the full-train set to train my top scoring model (direction = both), submitting that to Kaggle and comparing results to the top scoring cross-fold-train model would likely be a better comparison.

**Modelling – Lasso & Ridge Regression – Utilizing Cross-Fold Validation - Interactions**

I was fairly satisfied with my 4th percentile Kaggle score, but I figured I would quickly attempt lasso and ridge regression to see if this score could be improved.

Since I could not use the same “~.” code that inserts all independent variables into the regression, I decided to take a stab at utilizing some of the most significant integer variables to create interactions. I tried interacting OverallQual with factor variables that had significance such as Neighborhood and MSZoning. My logic being, if you know the neighbourhood of a property and the quality of that specific property, sale price for alike properties should be tightly clustered. I threw multiple combinations of these highly touted variables into this model as well.

Unfortunately, the MAPE was disappointing, returning 11.8% and the Kaggle submission score reflected that higher MAPE appropriately. I tried different combinations of interactions, but the results did not materially improve.

I used the same variable interactions for ridge regression and this model returned an even worse MAPE of 13%, so I did not even bother submitting it to Kaggle.

**Closing Thoughts**

Overall, I am quite happy with the end results of my modelling efforts. The log transformed stepwise regression model I created put me in the top percentile, which exceeded my expectations. Looking back on the time and effort spent on this competition, I would have focused my efforts on thorough data wrangling from the start as I spent a lot of time jumping back and forth cleaning data and re-running my code again when realizing the data needed further tweaks. I would have also spent time investigating the outliers. Per my plot diagnostics there were some outliers which had influence that I did not remove, which I am sure hurt model performance. On the topic of plot diagnostics, even with the sale price log transformation, the distribution still appeared a bit skewed. It is very likely that applying transformations to some of the integer variables would have helped with this. Given this skew, I feel that my model did not perform well on top/bottom 10% sale price listings. In a real-world situation, I would spend more time addressing this, using RMSE would be a good measure of performance during this effort due to it penalizing larger errors more severely. Finally, many of the models I built used essentially every independent variable in the data set; however, in the business world, I should be spending more time understanding which variables have significance, understanding impact of collinearity, and pairing that with business acumen/context to select and engineer meaningful independent variables for use in my model.