Choosing a Model To Predict Condo Prices in Burlington

Jana Taha

400279123

A thesis presented for the degree of

Doctor of Philosophy

Contents

1 Data Collection:

To be able to create a prediction model, we first need a data set. Therefore, I started looking and asking where I could get data on the Burlington House Market. I decided to go with the website, zoocasa.com. Zoocasa.com is a real estate website in Canada and the GTA (Greater Toronto Area) that features local listings and sold data back to 2003. In the website, you can filter by province and city, and so I was able to find data on real estate listings sold in Burlington in the last three years. The website can be further divided into Neighbourhoods and house types. I decided to include all 24 neighbourhoods, but only consider apartments. I decided to go with condos only, because condos have attributes that do not apply for some houses and vise versa.

There was around 2800 condos listed as sold in Burlington, divided by neighbourhood. Each neighbourhood had a set of condos divided into multiple of pages.(there is only 24 condos per page). To access each condos information, you had to click on each condo and it will open a new page with some information about the listing. It wouldn't have been efficient for me to click on each condo and copy down the information into a data set. Neither would it have been efficient for me to save 2800 html files into my laptop, then create a code the reads html webpages and binds them into a data set. To why, I decided to create an an RESTful API program in R using rvest[1] and jsonlite[2] packages. These two packages will help me scrap data from the website into a data frame efficiently. Using my program, I ended up with 21 data sets, one for each neighbourhood (some of the 24 neighbourhoods did not have condos sold in them)

2 My Data

Now that I have all my data, in another R script, I combined all 21 data sets into one. With my RESTful API program, I ended up with a data frame that contains information that I did not need. For example, my data set contained the column ,status of the house, which we know is going to be "sold" for all the listings in the data set. So I decided to remove all the unnecessarily columns such as the status, open house status, dat.type (its listing for all of them), rental status (false for all), and so on.

Our data set had a unique column (MLS number) for each condo sold. The MLS number is a number assigned to a real estate listing on the MLS® system. Every time a house is listed for sale

on the MLS® system, it gets an MLS® number. If the listing ends, the MLS® number is 'used up'.

We can not consider addresses to be unique, because the same apartment can be sold more than once within 3 years.

Since each listing had its own web page, there were some attributes that were on one page and not the other. So I made sure my data set only contain attributes that were found in all the 2800 pages. So now I have a complete data set with over 30 variables (predictors) and our response variable, price sold.

3 Exploring Variables and Missing Data

We now have a data set with 2696 rows and 36 Variables

3.1 Response Variable

For this report, I want to predict the price of apartments in Burlington. My data set contains the variable Price.Sold, which I am going to choose as my response variable. Even though there is also the variable Listed.Price, I am interested in this report predict how much a condo in Burlington would be *sold* for. Using ggplot2[3] package in R, I generated the following histogram:

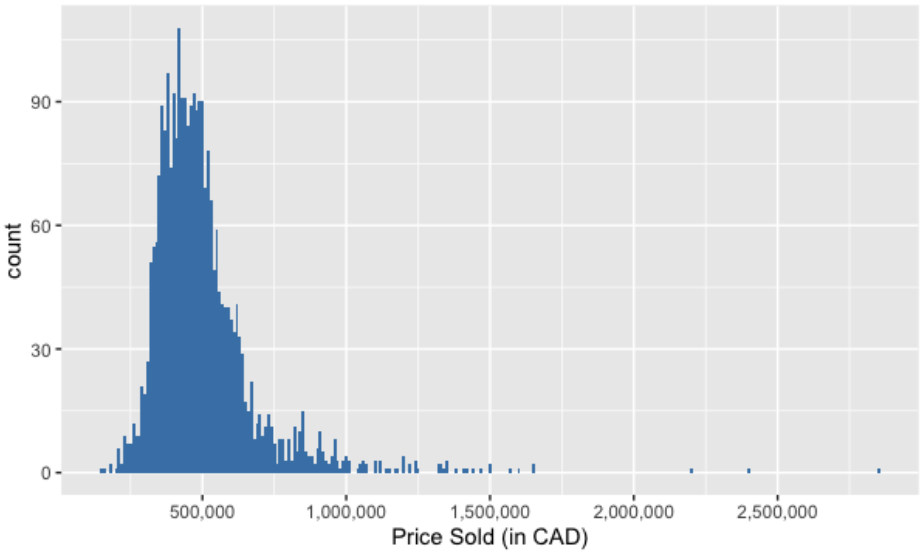

Figure 1: Histogram of the Prices Sold

We observe in Figure 1 above that Sales price is right skewed, which is expected as not a lot of people can afford expensive condos. We also see three outliers toward the end, were three condos were sold for over 2 million CAD. This

3.2 Missing Data

Even though my data set now only contains attributes that were found in all pages, we still encountered variables that had NA values:

```
> NAcol <- which(colSums(is.na(df)) > 0)
> sort(colSums(sapply(df[NAcol], is.na)), decreasing = TRUE)
 Extra.Bathrooms   Extra.Bedrooms      Approx..Age    Laundry.Level          Stories
            2696             2162              830              637               14
Maintenance.Fees      Postal.Code
               8                6
```

3.2.1 Bedrooms/ Bathrooms

Extra.Bathrooms and Extra.Bedroom, are variables that contain the number of any extra bedrooms or bathroom found on the condo. That could mean a den or a room/bathroom in the basement. Since the number of NA values in Extra.Bathrooms equals the total number of observations, we decide to remove the column. As non of our listings have any extra bathrooms. As for Extra.Bedrooms, I decided to change all the NA values to 0. Then I created a new column called Total.Bedrooms, that adds up the values of Bedrooms and Extra.Bedrooms.

```
> summary(factor(df$Bedrooms))
   0    1    2    3    4    5
   2  657 1172  822   41    2
> summary(factor(df$Bathrooms))
   1    2    3    4    5    6
 677 1332  531  152    3    1
> summary(factor(df$Total.Bedrooms))
   0    1    2    3    4    5    7
```

```
   2   353 1320   900   110    10     1
```

After looking at the summary of my three variables Bedrooms, Bathrooms, and Total.Bedrooms, I decided to group some of the categories. So now lets look at a visualization of the data:

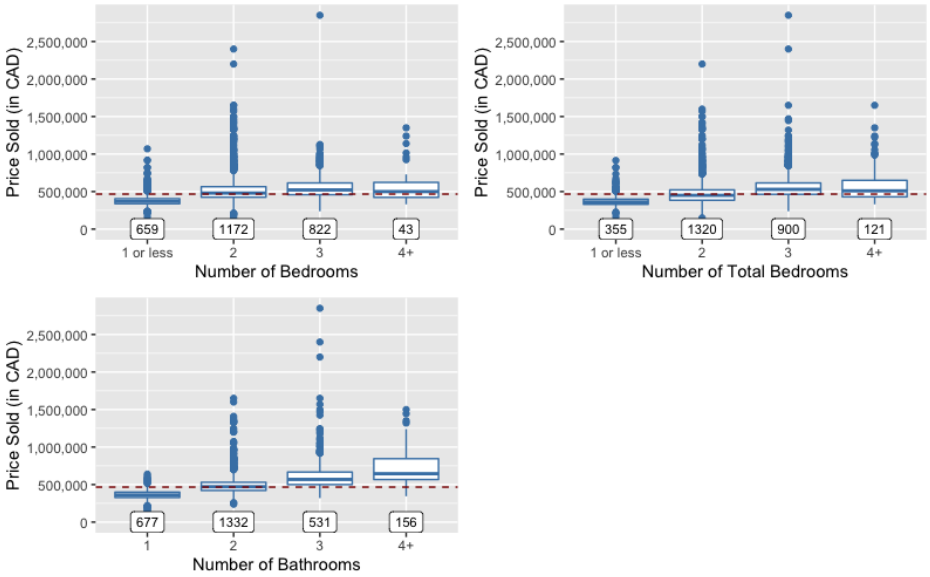

Figure 2: Bedrooms/Bathrooms Vs. Prices Sold

From Figure 2 above, we can definitely see that as the number of Bathrooms increases, the median of Price sold increases. As for Bedrooms and Total.Bedrooms, we notice that the 3 and 4+ bedrooms have around the same median, with 3 bedrooms median a little bit higher. But when I take the mean, the 4+ category have the highest mean of price sold. We also notice the same trend in both variables, therefore for the rest of the paper, I am only going to consider Total.Bedrooms. variable above look ordinal to me, so I am going to treat them as such.

3.2.2 Approximate Age

There are 830 NA values in Approx..Age, but before we deal with the NA values, lets look at the summary of the variable:

```
> summary(df$Approx..Age)
  0-5 years 11-15 years 16-30 years 31-50 years  6-10 years   New years
        353         157         700         408         184          29
51-99 years  6-15 years        NA's
         33           2         830
```

Notice that there is a category called 0-5 years and another one "New" years, so I am going to merge the 2 groups into one. Also notice how we have the 3 groups 6 - 10, 11-15, and 6 - 15 years. Therefore, I am going to merge the three categories into one.

Instead of just removing all 830 NA values, I decided to use the Condos addresses to get the approximate age. That is, condos are usually in apartment buildings and compounds. So there is a big chance that we will have the same address (street number and street name only) duplicated more than once in our data frame. I am hoping that for the NA values, I am going to find the address duplicated in another observations that contain the approximate age of the building.

In R, I created a new column that includes both Street number and street address. Then I generated a list of all addresses of the NA values. Next, using a recursive function in R, for each address in the list I extracted all the other duplicated addresses, and extracted their approximate age if possible. With that, I ended up with the following updated summary of Approx..Age:

```
> summary(df$Approx..Age)
0-5 years 16-30 years 31-50 years 51-99 years  6-15 years        NA's
      301         938         689          43         602         123
```

We were able to decrease the NA values from 830 to 123. As for the rest of the NA value, I am just going to remove the observations.

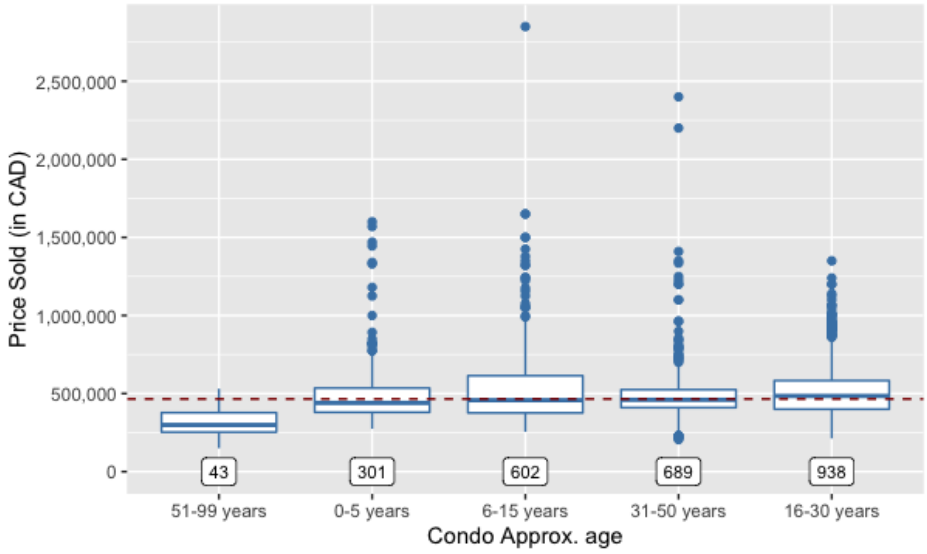

Figure 3: Bedrooms/Bathrooms Vs. Prices Sold

I was expecting that the newer is the Condo, the higher is the price sold. But from above we see that that is not true.

I also decided to change the Age variable it into an ordinal factor.

3.2.3 Laundry

There are 637 NA values in Laundry.Level. I have found a lot of problems with this variable.

```
> vec = df[!is.na(df$Laundry.Level) & df$Ensuite.Laundry == "No", c('Ensuite.Laundry', 'Laundry
> dim(vec)[1]
[1] 113
> head(vec, n = 10)
    Ensuite.Laundry Laundry.Level
48               No         Upper
54               No         Lower
75               No          Main
84               No         Upper
103              No         Lower
118              No         Upper
```

```
120              No         Lower
131              No          Main
134              No          Main
141              No         Lower
```

We can see from the above output, that there are some observations were we are told that there is no ensuite laundry, yet when we look at the Laundry.Level variable, we observe that there actually is a value there. And so I decided to further investigate some of the Condos that fall into the category above, and I noticed that some of the houses had an ensuite laundry, yet its Ensuite.Laundry variable had the value "No". Some of the other observations, really did not have an ensuite laundry, and by "main" Laundry.,level, they meant in the main floor of building but not the unit itself. But for most of the observations, it wasn't clear whether there is really an ensuite laundry in the condo or not. So if we decide to remove all the observations that falls into the category above, we will only end up with 63 observations under Ensuite.Laundry = No, which is really small relatively to our over 2000 observation. So I am going to remove Ensuite.Laundry variable from my data set.

There are over 500 value were Laundry.Level is NA yet Ensuite.Laundry's value is yes. Since this variable have a lot of NA's, and is inaccurate, we are going to exclude it too.

3.2.4 Stories and Levels

After further investigating the column "Stories", I have found that it is inaccurate. That it is, because it could mean two things: 1- The number of stories in a condo building or 2- the number of stories in the condo it self. And so I decided to remove the column, especially since I have another column "Levels" which has the number of stories in a condo unit.

```
> df[!is.na(df$Price.Sold),] %>% group_by(Levels) %>% summarise(median = median(Price.Sold), co
# A tibble: 10 x 3
   Levels          median counts
   <fct>            <dbl>  <int>
 1 Bachelor/Studio 320000      1
 2 Loft            410000     16
```

3	Apartment	413000	1348
4	Stacked Townhse	455000	95
5	Multi-Level	477500	36
6	Other	492000	9
7	2-Storey	517000	745
8	3-Storey	520000	214
9	Bungalow	599800	49
10	Bungaloft	842500	60

We have only 1 Bachelor/Studio apartment, since this is still and apartment, I decided to merge them together. I then looked into the "Multi-Level" and "other" categories, most of the observations that falls under both groups have unclear number of stories. Therefore, I removed them from my data set. Apartments and Lofts are both 1 storey condos, so I merged them together and renamed them "1-Storey". Stacked townhouses are 2-Stories, so I added them to the 2-Storey category. Bungalows are 1 1/2 Stories, and Bungaloft are a type of Bungalows with an extra loft. Therefore we merge them together into the"1 1/2 Stories" category.

I also made it into an ordinal variable

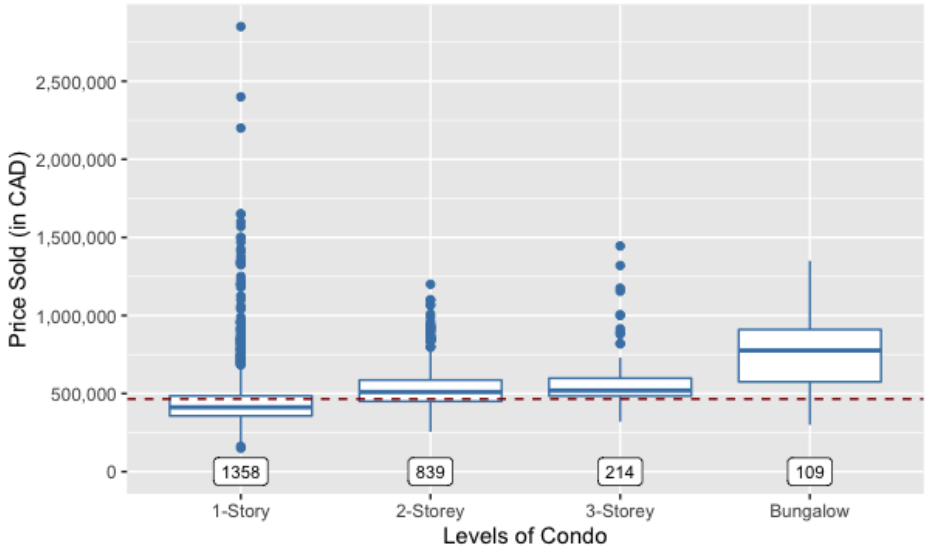

Figure 4: Bedrooms/Bathrooms Vs. Prices Sold

3.2.5 Maintenance Fees

Monthly maintenance fees can indirectly impact the property value of a condo if the fees are so high, it discourages anyone from buying it. As a result, the property stays on the market for longer than similar properties, and the asking price gets reduced to attract buyers. To test whether this is true, I am going to plot maintenance fees vs. Price. Sold. But first we have to deal with the NA values.

Since there is only 8 NA value, I decided to remove the 8 observations. Values in Maintenance.Fees are given as characters with the $ sign, and includes a comma if the number is more than 3 digits. So I used the function str_remove_all() from the package Stringr in R to remove the dollar sign and comma from the characters and transform them into numeric numbers.

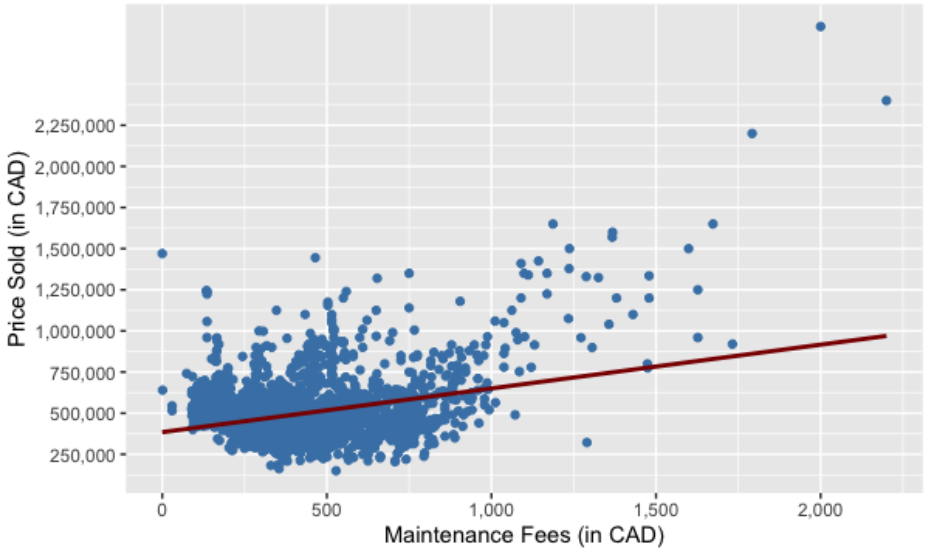

Figure 5: Maintenance Fees Vs. Prices Sold

We observe from the plot above that expensive condos have very high maintenance fees. But when we look at the average condo prices, we see no particular pattern. So on its own, maintenance fees does not tell us anything, but we could then test it correlation with other observations of the same attributes.

3.2.6 Region

Neighbourhoods with high crime rate, would have on average cheaper condos. Yet, houses in the city, near the highways and shopping centres would sell for higher prices. I have two variables that looks at a condo's location, Postal.Code and Neighbourhood. Taking aside the 6 NA values, I decided to trim my postal codes into just the first three characters. I then changed it into a factor, and got the following summary:

```
> summary(factor(df$Postal.Code))
 L2P  L3M  L3R  L5R  L6L  L7A  L7H  L7K  L7L  L7M  L7N  L7O  L7P  L7R  L7S
   1    1    1    1    3    1    3    2  746  788  118    1  281  215  275
 L7T  L8L  L9T NA's
 234    1   10    6
```

Most of these postal code only contain a very little number of houses, which wont say much to us. So maybe we could consider the other variable related to region,which is Neighbourhood.

```
> summary(df$Neighbourhood)
Aldershot Central   Aldershot South    Aldershot West       Alton North       Brant Hills
               58               166                 1                86                89
          Central         Corporate             Dynes  Elizabeth Garden     Headon Forest
              144               251               103               161               205
         Longmoor            Maple         Millcroft      Mountainside           Orchard
               22               279               179               106               198
           Palmer          Pinedale           Plains          Roseland           Tansley
              115               113                47                 6               195
         Tyandaga
               41
```

We can definitely see that Neighbourhood is much more symmetric than Postal.Code. as so I decided to get rid of the postal code variable, and use Neighbourhood instead. Since Aldershot West have only one observation, I decided to remove it as it wont say much to us.
In Figure 6 below, the plot above is a graph of the Median Price sold vs. Neighbourhood, and the one on the bottom in the mean price sold vs. Neighbourhood.

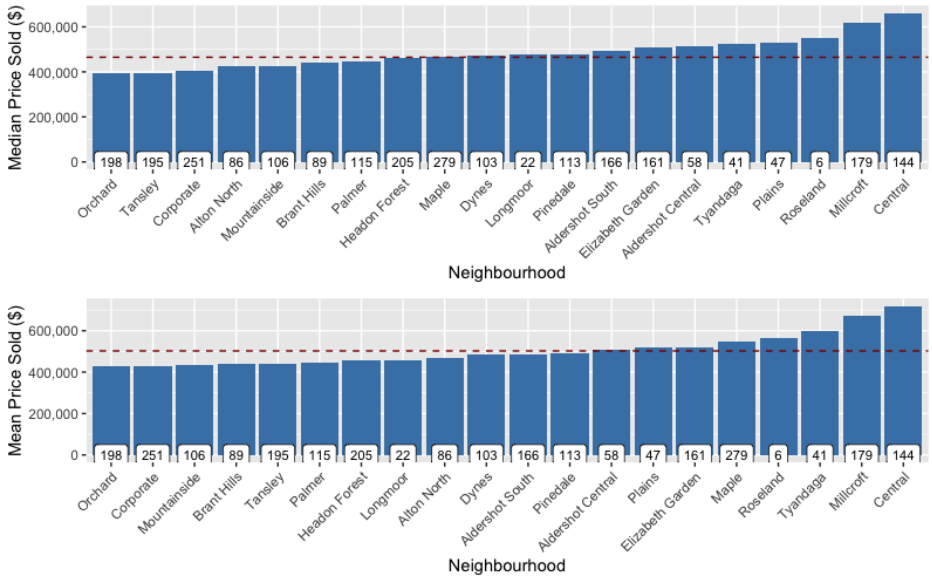

Figure 6: Neighbourhood Vs. Prices Sold

From the 2 graphs above, we conclude that Orchard is the Neighbourhood with the lowest prices and Millcroft and Central have the highest prices.

3.3 The Remaining Variables

Within 3 years, there could have been a period were house prices were high due to house crisis market. So I decided to plot the median of the price sold for each month for the three years, and got the following plot:

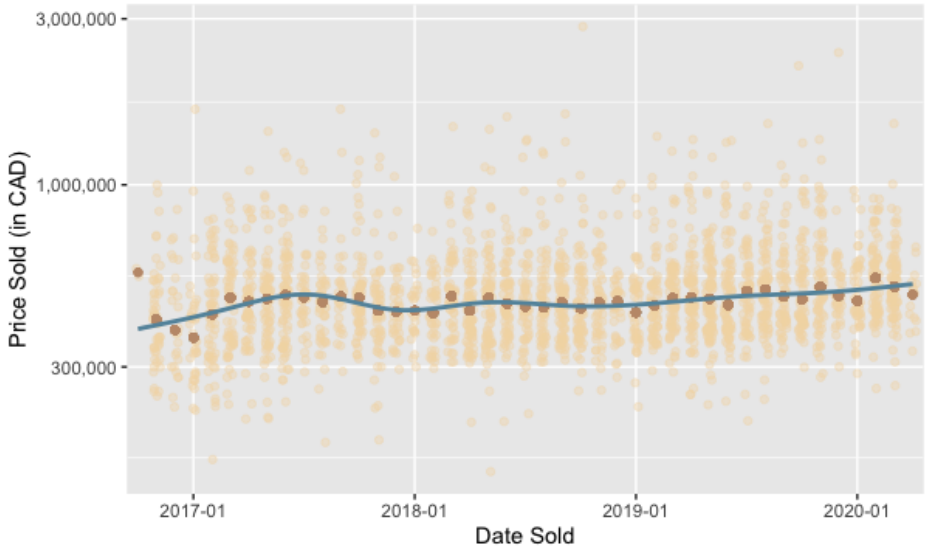

Figure 7: Time Sold Vs. Prices Sold

Other than midway through 2017, we see that the prices were constant through out time.

3.3.1 Condo Size

We have a variable called Size that has the approximate size of a condo.

```
> df$Size = reorder(df$Size, df$Price.Sold, FUN=median)
> df[!is.na(df$Price.Sold),] %>% group_by(Size) %>% summarise(median = median(Price.Sold), cou
# A tibble: 18 x 3
   Size                median counts
   <fct>                 <dbl>  <int>
 1 0-499 sq. ft.      334950       8
 2 500{599 sq. ft.    354250      68
 3 700{799 sq. ft.    368000     256
 4 600{699 sq. ft.    369950     218
```

```
 5 800{899 sq. ft.    393500     156
 6 900{999 sq. ft.    425000     163
 7 1000{1199 sq. ft.  460000     438
 8 1200{1399 sq. ft.  495000     617
 9 1400{1599 sq. ft.  567475     324
10 1600{1799 sq. ft.  666990     119
11 1800{1999 sq. ft.  743256.     76
12 3250{3499 sq. ft.  775000       2
13 2000{2249 sq. ft.  847500      36
14 2250{2499 sq. ft.  960000      19
15 2500{2749 sq. ft.  967500       8
16 2750{2999 sq. ft. 1132500       8
17 3000{3249 sq. ft. 2175000       2
18 4000{4249 sq. ft. 2400000       1
```

In the above output, I ordered the groups by the median price sold in an increasing order. We notice that as the size of a condo increases, the price increases. We also notice that there is barely any observations in the smallest and largest condo. So I am going to combine 0-499 and 500-599 sq. ft. condos into one. And combine all the condos that are more than or equal to 2000 sq ft. into one group.

We do not notice any outliers, and the size variable so far seems like our best variable. I also decided to change it into an ordinal variable.

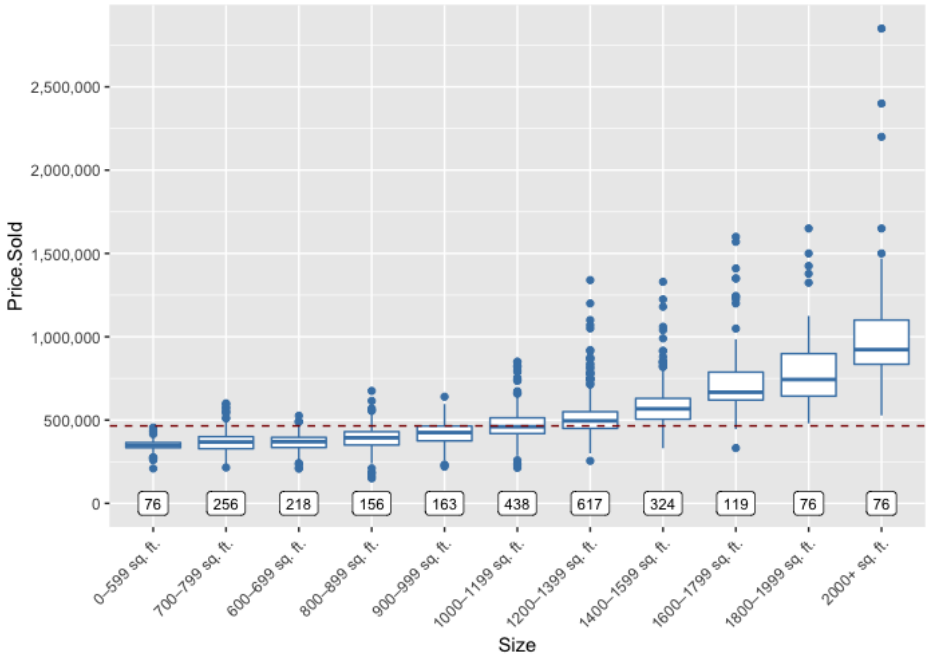

Figure 8: Size of Condo Vs. Prices Sold

4 Test and Train set

Through out the report, I am going to use supervised approaches. Therefore, I am going to split my data into a training and a test set. *Supervised statistical learning* involves building a statistical model for predicting or estimating an output based on one or more inputs[4]. That is, for each observation of the predictor measurements $x_i, i = 1, ..., n$ there is an associated response measurement y_i. The *training set* contains labelled observations and is used to build the model that relates the response to the predictors [4]. The observations in the *test set* are unlabelled or treated as such and are used to assess the efficiency of the model.[4]

After all the data cleaning that has been done in section 4, I now have 2273 observations in total. I believe that taking a 70:30 train-test split will be efficient.

When cleaning the data, I made sure I did not have any categories with small number of observations. That is, because if there was a small category that did not make it into the train set but made it into the test set, it will cause us lot of problems when assessing the performance of a statistical learning method.

To make sure all my work is reproducible, I used set.seed=720 in R. That is, all my results, graphs,

and tables included in the report will be generated with set.seed = 720. I took a random sample using R to get my train and test set.

5 Linear Regression

Suppose we have p predictors, then the *multiple linear regression* takes the form

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon \tag{6.1}$$

Where X_j represents the jth predictor and β_j is the average effect on Y of a one unit increase in X_j, holding all other predictors fixed.
In linear regression, we make predictions using the formula:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + ... + \hat{\beta}_p x_p \tag{6.2}$$

We choose $\beta_0, \beta_1, ...\beta_p$ to minimize the sum of squared residuals

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_1 - \hat{\beta}_2 x_2 - .... - \hat{\beta}_p x_p)^2 \end{aligned} \tag{6.3}$$

The values $\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p$ that minimizes RSS above are the multiple least squares regression coefficient estimates.
For p $\geq$ 2, the coefficient estimates have complicated forms, hence we can use any statistical software package to compute them.

5.1 R Application

Using lm() function in R, I fitted a linear regression model using least squares for my data.
Since a lot of the variables were categorical with multi levels, we ended up with a lot of parameters. That is because let say a variable had d levels, then that variable will have d -1 estimated coefficients.

Below is just a part of the R output:

```
> bmod1 <- lm(y_train ~., Train)
Residual standard error: 107500 on 1839 degrees of freedom
Multiple R-squared:  0.6608,        Adjusted R-squared:  0.6493
F-statistic: 57.78 on 62 and 1839 DF,  p-value: < 2.2e-16
```

The F statistic above test the Hypothesis

$$\begin{aligned} & H_0 : \text{There is no relationship between the response variable and the predictors} \\ Vs. \quad & H_a : \text{At least one of the predictors is related to the response variable.} \end{aligned}$$

Since P-value is really close to 0, we reject the null hypothesis and conclude that at least one of our predictors is related to our response variable, Price.Sold.
The R^2 value above, is a measure of how well the model explains the data. The best R^2 we can get is 1. So there might be a model out there with a better fit.

5.2 MSE

To assess the performance of a statistical learning method on a given data set, we calculate mean squared error (MSE),

$$\text{MSE} \quad = \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where $\hat{y}_i$ is the prediction for the ith observation. This measures how well a models predictions actually match the observed data. A small MSE value implies that our predicted responses are very close to the true response.
In statistical learning, to asses the performance of our model, we calculate the test sets response variable. That is, we fit a model using our training set, then use the model to predict our test sets Y values. For the rest of the paper, I am going to be using RMSE, which is just the square root of MSE.
Using R, I calculated the RMSE of the model above, and got the value, 105816.1. This is a really large value, and so we need to further investigate. In the output below, we have the first 10 predicted observations and the actual first 10 observations in the test set.

```
> res = cbind(yhat, y.test)
> head(res, n= 10)
       yhat y.test
1  449661.2 545000
5  310707.8 422500
13 605872.8 555000
14 271230.5 424900
17 288593.1 446000
19 281420.7 395000
20 281929.0 392000
24 535346.3 517500
27 273018.2 445000
29 274572.1 385000
```

I also decided to include a graph of the predicted value vs the actual value:

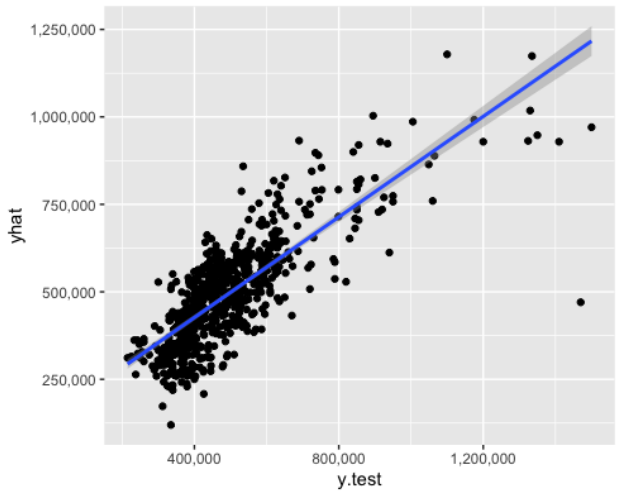

Figure 9: Predicted vs. Actual Value

6 Linear Model Selection

We could consider another least square regression model that yields better prediction accuracy and interpretability. As with to much predictors, it was hard for us to interpret the results above.
In this section, we are going to discuss the methods *Subset selection* and *Shrinkage*. In *Subset selection*, we identify a subset of our predictors that we believe is related to the response. We then fit a model using least squares on the new reduced set of predictors.
As for *Shrinkage*, we fit all the predictors, but it *shrinks* the estimated coefficients towards zero. By reducing the effect of these predictors, we reduce variance and hence get a model that is more easily interpreted. Sometime Shrinkage methods estimate some coefficients to be exactly zero, so shrinkage methods can also be used for variable selection.

6.1 Subset Selection

Some of the methods that can be used for selecting subset of predictors, are the *best subset selection* and the *step wise model selection*

6.1.1 Best subset Selection

In *best subset selection*, we fit a separate least square regression for each possible combination of the p predictors. We then then select the best model among the 2^p fitted models.
The best subset selection algorithm is shown below:

The Best Subset Algorithm

1. Let M_0 denote the null model, a model that contains no predictors.
2. For $k = 1, 2, ..., p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it M_k. The best model here is defined as the model having the smallest RSS (defined in (6.3).
3. Select a single best model from among $M_0, ...M_p$ using C_p, AIC, BIC, or adjusted R^2.

The reason why in step 3, we choose the optimal model by calculating C_p, AIC, BIC, or adjusted R^2 is because we care more about a low test error than a low training error. The formulas for C_p, AIC, BIC, and adjusted R^2 are introduced in the section below.

6.1.2 C_p, AIC, and, BIC

In step 3 of our best subset algorithm, we want to select the best model among p + 1 models. Other than that RSS and R^2 are quantities related to training error, the model containing all of the predictors will always have the smallest RSS and largest R^2. So they wont be suitable for selecting the best model, when our models have different number of predictors. Let d be the number of predictors in the model.

Then the C_p estimate of test MSE is given by:

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$$

Where $\hat{\sigma}^2$ is an estimate of the variance of the error ϵ associated with each response measurement in (6.1). A penalty of $2d\hat{\sigma}^2$ is added to the training RSS in order to adjust for the fact that the training error tends to underestimate the test error.

Akaike information criterion (AIC) is defined by:

$$AIC = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + 2d\hat{\sigma}^2)$$

Bayesian information criterion (BIC) is given by,

$$BIC = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + log(n)d\hat{\sigma}^2)$$

6.1.3 R Application

regsubsets() function from the leaps library in R preforms best sub selection by identifying the best model that contains a given number of predictors. The function has the attribute, nvmax. This is used to return as many variables as desired. In default nvmax = 8.

When I ran this function on my data set in R, I got an error that there is to many predictors. This could be because we have many categorical variables with many levels. So we to many predictors to be able to fit 2^p models.

Therefore, I going to stop with Subset selection, and move on to shrinkage.

6.2 Shrinkage Method

In shrinkage method, we fit all the predictors. The only difference from a regular least square linear models, is that we are shrinking the regression coefficients towards zero.

6.2.1 Ridge Regression

In Ridge regression, we estimate the ridge regression coefficients estimate by minimizing the following quantity:

$$\begin{aligned} & \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 \quad + \quad \lambda\sum_{j=1}^{p}\beta_j^2 \\ = \quad & \text{RSS} \quad + \quad \lambda\sum_{j=1}^{p}\beta_j^2 \end{aligned}$$

where $\lambda \geq 0$ is a tuning parameter, to be determined. Ridge regression produces a different set of coefficient estimates, $\hat{\beta}_\lambda^R$, for each value of λ

One disadvantage for ridge regression,is that we always end up including all p predictors in the final model. That is, our shrinkage penalty $\lambda \sum_{j=1}^{p} \beta_j^2$ will shrink all of the coefficients toward zero, but never to exactly zero. So if we want to fit a model that only includes the important varaibles, then *lasso* wold be a better alternative.

6.2.2 Lasso

In Lasso regression, we estimate the lasso coefficients, $\hat{\beta}_\lambda^L$, by minimizing the following quantity:

$$\begin{aligned} & \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 \quad + \quad \lambda\sum_{j=1}^{p}|\beta_j| \\ = \quad & \text{RSS} \quad + \quad \lambda\sum_{j=1}^{p}|\beta_j| \end{aligned}$$

Unlike ridge regression, the lasso shrinkage penalty, $\lambda \sum_{j=1}^{p}|\beta_j|$ has the effect of forcing some of the coefficient estimates to be exactly 0. So lasso can also be used to select the most important variables.

6.2.3 R application

We use the same function in R, to fit ridge regression and Lasso. We use the glment() function from the glement library in R. If you set $\alpha = 1$, then a ridge regression model is fit and if you set $\alpha = 0$, then a lasso model is fit. Even though glment() function automatically select a range of λ values, I decided to implement the function over a grid of values ranging from $\lambda = 10^{10}$ to $\lambda = 10^{-2}$ Before I can use the predict function, I need to first choose a value for λ. We can do that using cv.glment(), which by default preforms 10-fold cross validation.

Ridge and Lasso did not do much better than Linear Regression. Using R I got an MSE of 9930615198 for Ridge regression and an MSE of 11011863282 for the Lasso.

It seems like linear regression is not a good idea for our data, so in the following section we consider tree based methods.

6.3 Bagging:

Bagging can be used for reducing the variance of a statistical learning method. Generally, averaging a set of observations reduces variance. So one way to reduce the variance of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions.[4] Its not practical for us to generate multiple training sets, so instead we use bootstrapping. *Bootstrap* takes in repeated sample from a single training set. That is, in bagging, we generate B different bootstrapped training data sets then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$. Then finally average all the predictions to obtain:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

We can apply bagging to our data set using the randomForest[9] package in R. The randomForest() function takes in 2 parameters mtry and ntree. mtry is the number of predictors that should be considered for each split of the tree (in bagging we include all predictors), while ntree is the numbers of trees grown. In R, I ran a for loop for several number of trees and recorded the MSE for each different number of trees.

6.4 Random Forests:

Random forests provide an improvement over bagged trees by decorrelating the trees.[4] Just like bagging,we build a number of decision trees on bootstrapped training samples. The only difference is that each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. That is, the split can only use one of those m predictors. A new sample of m predictors is taken at each split. Often, M= $\sqrt{p}$ is used. In R, I am going to apply random forest to my data set using the same function randomForest(). I need to test for different parameters ntree and m. In our data set, we have 8 predictors. $\sqrt{15}$ is approximately 4, but I am also going to test for m=2,3,and 4, and see which m gives me the lowest MSE out of different number of trees.

In R, we found that m = 4, and ntree = 800 gave us the lowest MSE value of 4758197944.

6.4.1 Variable Importance

Bagging and Random Forests provide a better prediction accuracy than regression trees, but they improved at the expense of interpretability. That is, because in Bagging and Random Forests we are bagging a large number of trees so we wont get an easily interpreted diagram like we do in regression trees. But we can though use the importance() function, and view the importance of each variable. Two measures of variable importance are reported. One is based upon the mean decrease of accuracy in predictions on the out of bag samples when a given variable is excluded from the model. And the is a measure of the total decrease in node impurity that results from splits over that variable, averaged over all trees (measured by training RSS)

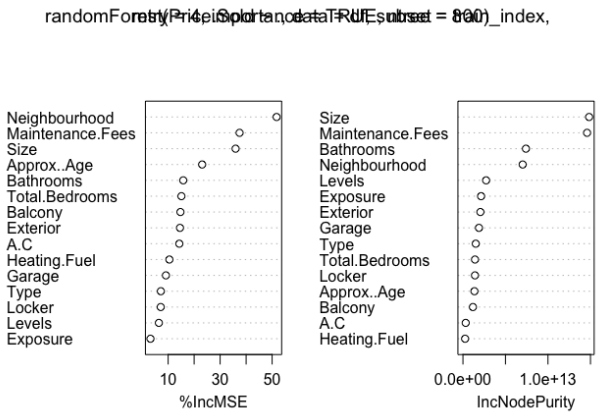

Figure 10: Predicted vs. Actual Value

From The plots above, we see that Neighbourhood and Maintenance.Fees are the most important variables. We were able to see that in the data visualization part of the project.

7 Conclusion:

We conclude that the tree based methods did much better than the linear regression methods. But in general, none of the methods did a good job. This could have to do with the data set, as we had a lot of different attributes, bit not enough observations.

8 References

1. https://www.zoocasa.com/
2. https://thatdatatho.com/2018/12/14/an-introduction-to-scraping-real-estate-data-with-rvest-and-rselenium/
3. https://www.programmableweb.com/news/how-to-access-any-restful-api-using-r-language/how-to/2017/07/21?page=2
4. https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/
5. H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.
6. adley Wickham (2019). stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.4.0. https://CRAN.R-project.org/package=stringr
7. Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2019). dplyr: A Grammar of Data Manipulation. R package version 0.8.3. https://CRAN.R-project.org/package=dplyr
8. Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra
9. Thomas Lumley based on Fortran code by Alan Miller (2017). leaps: Regression Subset Selection. R package version 3.0. https://CRAN.R-project.org/package=leaps
10. A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.
11. Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/
12. James, G., Witten, D., Hastie, T., Tibshirani, R. and Friedman, J. (2013). An Introduction to Statistical Learning. Springer: New York.