CMPT 353 Final Project

Akshay Agrawal

301153611

Tek Donald Duy

301223216

Nov 30, 2019

Predicting Real Estate Sales Prices through Machine Learning

The Problem

Realtors make their living guiding people through what is most likely the largest single purchase of their lives. They add value in several ways, however, with the plethora of research tools available for buyers, buyers are more informed and less likely to need a realtor for initial research. A key area in which value is added by Realtors is in advising clients on pricing. A realtor has access to all the historical sales data in the area, and based on that data, they suggest a range in which the property will likely sell (or the property is realistically worth). This is typically lower than the list price, and in most markets there is an expectation that there will be some negotiation (therefore some room is built into the list price to negotiate). We'll be looking at the purchase side of the real estate transaction. Our objective was to see if by inputting all this data into a neural network, could we predict with some degree of accuracy what that sale price should be?

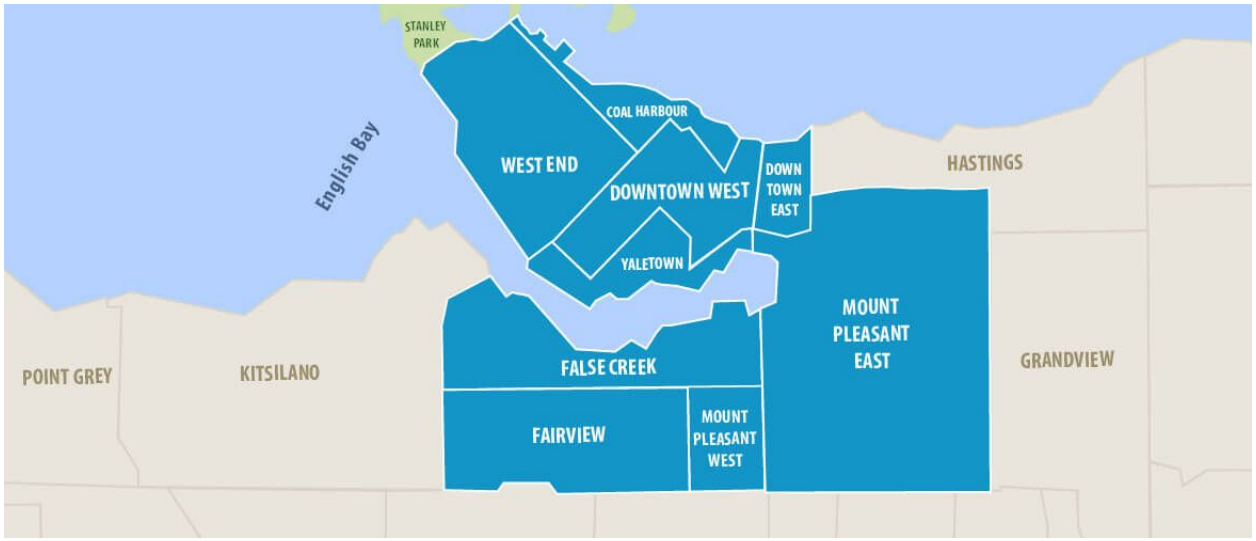

Figure 1. A visualization of the Sub Areas of Downtown Vancouver. The Sub Areas focused on in this project will be Downtown, Coal Harbour, Yaletown and the West End.
Source: https://www.vancouveruniquecondos.com/neighbourhoods/

The Data

The dataset we used to train the network consists of historical sales data in Vancouver, BC. Specifically apartments in the downtown core consisting of four sub areas (Downtown, Coal Harbour, Yaletown and the West End), which can be seen in **Figure 1**. The data has values for ~15,000 past sales with 20 attributes resulting in 300,000 initial data points. This shrunk to ~180,000 data points after the cleaning process.

Attributes

List Price: The price the property was listed for
Sold Price: The price for which the property sold
Floor Area: The total square footage of the livable area inside the property
Bedrooms: Total number of bedrooms
Bathrooms: Total number of bathrooms
Maintenance Fees: The monthly strata fees (in dollars)
Year Built: The year the building was built
Parking: Number of parking spots included with the apartment
View: A boolean value specifying yes or no (Nulls were converted to no)
View Specify: If yes, specify what kind of view. (Strings were converted to a scale of 0-3)

0 - Indicates no view or no view specified

1 - Indicates view is yes, but specified as other (all values not accounted for)

2 - Indicates a mountain or city view

3 - Indicates a water, ocean or harbour view

Sold Timestamp: This is the sold date converted from a datetime value to a timestamp

Sub-Area: Downtown, Coal Harbour, Yaletown and the West End - Converted to values 1-4

Days on Market: The number of days the property was on the market prior to the sale

Data Cleaning and Analysis

Since the dataset was so large, the historical sales data was contained in multiple .csv files. Pandas was used to read each .csv file and concatenate the entirety of the dataset into a single DataFrame. The DataFrame contains column names that were not intuitive to a general audience, so they were aptly renamed to be more clear. Afterwards, it was found that columns containing dollar amounts included dollar signs and commas, which would interfere with the machine learning algorithm used. As such, these characters were removed and dollar amounts were converted to floats.

There was also the matter of converting categorical values to numerical values, which represented the majority of the cleaning. Firstly, the values present in the View column and the VwSpecify column, which contains a description of the view, were used to assign a numerical value to a new column called ValueOfView. Properties that had no view or lacked a description for a view were penalized and given a low numerical value of -1 and 0 respectively. In contrast, if a property had a view of mountains or the city, this is desirable by purchasers so these properties were given a value of 2. Likewise, if a property had a view of water, harbours or an ocean, this was given a value of 3 since these types of views are highly desirable. Additionally, if a property had a view and did not contain any of these key features, they were assigned a value of 1 since having some sort of view is better than having no view.

After finishing with the views, the SubArea column also needed to be converted to numeric values. In the Vancouver downtown area, there are only 4 recognized subareas in the dataset: VVWCC, VVWDT, VVWWE, VVWYA, which represent Coal Harbour, Downtown, the West End, and Yaletown. As a result, a new column was created called ValueOfSubArea and these sub-areas were sequentially given a value starting from 1. Coal Harbour is represented with a 1,

Downtown is represented with a 2, the West End is represented with a 3, Yaletown is represented with a 4. Next came the Locker column.

The values in the Locker column were changed from “Yes” and “No” to 1 and 0 respectively. Nulls in this column were also converted to 0. Likewise, the Parking column had nulls as well, so these null values were also converted to 0. Lastly, the Sold Date column needed to be converted from a string into a numerical value. The strings in the Sold Date column were converted into datetime format and then these datetimes were converted to a timestamp to populate a new column called Sold Timestamp.

Upon the successful conversion of categorical values to numerical values, the data was ready to be used to create a regression model *via* machine learning. Initially, only a subset of the data containing listings from 2019 were used for testing. This was done to avoid any problems with processing speed as the whole dataset is exceptionally large. The parameters used for prediction were ValueOfSubArea, DaysOnMarket, Bedrooms, Bathrooms, FloorArea, YearBuilt, Age, Locker, Parking, MaintenanceFees, SalePricePerSquareFoot, List Price, Sold Timestamp and ValueOfView. The value to be predicted was Sold Price. The data was split into training data and validation data using an 80:20 split. Then a model was fit by putting the training data through a neural net, MLPRegressor specifically. The score obtained from this model was -0.013, which was distressing since it was so low and negative.

It was determined that the columns of the DataFrame containing dollar values needed to be scaled down since the other columns contained values from a range of 1 to 10. The columns List Price and Sold Price were in and beyond the 100,000s, so they were divided by 10,000. Likewise, the columns MaintenanceFees and SalePricePerSquareFoot were in and beyond the 1000s, so these values were divided by 100. However, this amount of scaling was still not completely sufficient, since a negative score was still being given for the model. In response, a pipeline was created that used StandardScaler prior to training the neural net. This allowed the means of each parameter to centered and scaled according to unit variance, which would provide a more comprehensive scaling.

After scaling, it was noticed that not Pandas had read some of the numerical columns as strings. As a matter of precaution, every column in the DataFrame was explicitly converted to a float.

Then once again, the 2019 listings were split and the model was fit with the new training data. This time the model produced a score of 0.822, which was promising since only a subset of the data was used. When the entire dataset was used, the highest score achieved was 0.829 and after multiple tests, the lowest score achieved was 0.661.

Since an acceptable model was produced, the last 100 rows of the 2019 dataset were removed from the training and validation data. These 100 rows were saved to be used for predictions. The prediction data was cleaned in the exact same manner as the training and validation data and put through the model to predict their sold price. For these 100 properties, the predicted prices were proportional to the actual sold price by 95% to 114% on average when observing the means of predicted price/sold price. When observing the mean of the residuals, the range of the mean of the residuals were between -80,000 to 290,000. This means the predicted price was either less than the actual price by up to $80,000 or greater than the actual price by up to $290,000. The sold prices and predicted prices were then plotted on a scatter plot to visualize roughly how far apart the predicted prices were in comparison to their actual sold prices (**Fig. 2**).

Figure 2. Initial Scatter Plot of Sold Price and Predicted Price of 100 Properties in 2019.

Upon observing the scatter plot, the distances between sold price and predicted price are generally not too far off from each other visually. However, any amount of distance between the

two prices are detrimental for buyers since we are looking at differences in the 10,000s. As such, it was determined that the model might be improved by increasing the size of the hidden layers since the model was fairly close to approaching the actual sold price. Therefore, the hidden layers were increased to 18, 16 from 8, 6 to allow the model to make more accurate decisions. Surprisingly, this increased the model score to 0.951. Additionally, it tightened the mean of the residuals to be between -5.73 and 0.083 and the proportion of the predicted price to the actual sold price to a range of 97% to 102%. This is reflected in a better visualization in **Figure 3** where the predicted prices are much tighter with the actual sold price.

Figure 3. Scatter Plot of Sold Price and Predicted Price of 100 Properties in 2019 After Increasing Hidden Layers.

After these results, the combination of the model score, residuals, and visualization were deemed reasonable and thus, the model was determined to be sufficiently trained to predict active listings.

Results

The first time we trained the neural net, our score was -0.22, this was disheartening, but by investigating how the score is calculated we realized that we hadn't provided information in the

correct format (a lot of strings were present, and data wasn't normalized). Once we fixed those issues in cleaning, we got a much higher score (~0.8). This seemed great but when we tried to predict the price of an active listing the result was not realistic. For example: the neural net stated that for a property listed for 1.48 million, the sale price should be 650k. This did not seem likely so we went back to investigate. We noticed that the neural net was being trained using sale price per square foot which is something that should be in the results set, not in the training set so we removed that. Interestingly, the average score went down but we started getting more reasonable results when predicting pricing of active listings.

Active Listing 1: 2801 - 89 Nelson Street Cooperage Park (Yaletown)

List price: $1,648,800
Floor area (in square feet): 1,055
Bedrooms: 2
Bathrooms: 2
Parking stalls: 1
Storage lockers: 1
Maintenance fees: 600
Year built: 2018
Days on market: 30
Sub area: Yaletown
View: Water

ML Predicted Sale price: $1,775,915 (High)

Active Listing 2: 1110 - 1500 Hornby Street Beach Avenue (Yaletown)

List price: $1,338,000
Floor area (in square feet): 1,212
Bedrooms: 2
Bathrooms: 2
Parking stalls: 1
Storage lockers: 1

Maintenance fees: 673
Year built: 1994
Days on market: 50
Sub area: Yaletown
View: City

ML Predicted Sale price: $1,323,710 (Pass)

These results are a great starting point, having a licensed Realtor on our team, we can confirm that these are within reason for an active listing. However, the neural network also doesn't take into account certain special characteristics such as a high floor, a penthouse unit, a large deck or renovations. As such, units with these special characteristics have to be manually tweaked after the recommendation has been given.

Limitations

The dataset originally included an address, which was believed to have been useful. However, the neural network was not able to accept the address as it currently was a string. It was determined that the addresses could be converted into latitudes and longitudes. An API called GeoPy was found and was capable of automating this process. Unfortunately, the sheer amount of data being sent to the API caused a bottleneck in the data cleaning process. With this many requests, the API could not respond in an appropriate amount of time. As such, the address portion of the data set was discarded and subarea was used as a substitute in place of an exact latitude and longitude.

Future Considerations

An average score above 0.9 isn't bad when looking at a problem of this nature. In fact, 0.9 may even be close to what Realtors would score since sale price is such a subjective term. However, we have several recommendations for future work and how this score could potentially be improved:

1. More parameters/better parameters

Pricing a property depends on many factors, but we have accounted for the more important ones, such as location, size, age, and view. However, by increasing the number of parameters the neural net accepts, we could potentially increase the accuracy.

On a related note, there were parameters in our data which we quantified for the neural net - such as view. This was quantified on a scale of 0-3. Increasing the accuracy of this scale or the scope could significantly increase the accuracy as view is a very subjective measure that affects price greatly. We could even run some view pictures through another neural net and get that to classify that view on a more accurate scale.

2. More data

This one is fairly obvious, but more data can always help a neural net. In our case, we used all the data available for downtown Vancouver for the past 5 years. It's possible that increasing the time range or using data from outside the downtown core could increase the accuracy. However, there is a chance it could decrease as the downtown core is unique, and a net trained on data from outside could miss nuances present only in the downtown market.

3. Designing the neural net
    a. Change the activation function

A deeper understanding of the activation function could yield more appropriate weights and design for the neural net. With our limited understanding this would be guesswork at the moment.

    b. Active weighting

By understanding that a disproportionate weight was being given to sale price per square foot and eliminating that from our training set, we were able to improve our results. By increasing the weights on certain more important parameters in the beginning, we could potentially yield a better result.

    c. Neural net design (Consideration for convolution)

This again is guesswork at this time, but certain designs for neural nets yield better results for certain types of problems. For example convolutional neural networks work well for image recognition problems. A deeper understanding of theoretical machine learning could also yield a neural net design that worked better for this problem.

Experience Summaries

Donald:

- Concatenated the dataset from multiple .csv files into a single Pandas DataFrame using the Pandas library in order to clean the entire dataset concurrently
- Cleaned the dataset using Python and Pandas by removing null values, converting categorical values into numerical values, and then scaling the values in preparation for training in a Multi-layer Perceptron regressor machine learning model
- Created a pipeline that transformed the data using StandardScaler and fit a model using a Multi-layer Perceptron regressor in order to train a machine learning model that could predict a suggested sale price of a property in the Downtown Vancouver area
- Created visualization for our results using a scatter plot with Matplotlib and Seaborn to present actual values versus predicted values from the machine learning model
- Contributed to the writing of the project report, with most of my efforts being in the Data Cleaning and Analysis section

Akshay:

- Found and gathered data from MLS by setting parameters and downloading small data sets as CSV's to be concatenated later
- Designed data cleaning protocols to have data in a readable and usable format so that visualization and future manipulation would be less confusing.
- Performed cleaning on the dataset by converting subjective measures like view to objective numbers to help train the neural net.
- Investigated how *score* function works to figure out why first neural net was yielding such a low score, this led to our biggest data cleaning effort.
- Performed trial and error manipulations to improve score, this led to removing sale price per square foot and gave us our first meaningful results.
- Worked on report, specifically the Problem, Results, Attributes, Limitations, Future Considerations sections
- Designed iteratively improved the neural net after understanding how accuracy could be improved
    - First converted values from strings to float

- Normalized data
- Removed sales price per square foot
- Increased hidden layers

This led to an increase in score from -0.013 to ~ 0.95