Toronto House Price



**Group 8**

**Debjani Mukherjee**

**Armenak Nargizyan**

**Harris Nazumudeen**

**Jing Wang**

**Shuowen Xian**

[Introduction 3](#_Toc58794620)

[Dataset 3](#_Toc58794621)

[Data Preparation 3](#_Toc58794622)

[Analysis 4](#_Toc58794623)

[Feature engineering 4](#_Toc58794624)

[Scatter Plot Visualization 6](#_Toc58794625)

[Correlation Analysis 7](#_Toc58794626)

[Price Analysis 8](#_Toc58794627)

[Condo 8](#_Toc58794628)

[Spark MlLib / Prediction 11](#_Toc58794629)

[Indexing 11](#_Toc58794630)

[Spark Pipeline 12](#_Toc58794631)

[Linear Regression Model 12](#_Toc58794632)

[Random Forest Model 13](#_Toc58794633)

[Conclusion 14](#_Toc58794634)

# Introduction

Toronto is the largest city in Canada. The population is only 6 million and most of them live and work in the southern area. The population density is very varied and that has an impact on the house price. The goal of the project is to build a price indicator using machine learning models of Spark MLLib.

# Dataset

We researched on the Greater Toronto Housing Data available from zoocasa . This dataset was available on Github. The dataset shows the sale of individual residential property in GTA from to 2019-2020 . The variables include nominal, ordinal, discrete, and continuous data types. The variables which have physical property measurement and also computation variables are used in the city’s assessment process. The same variables also contributed to the assessment of the property values

## Data Preparation

During the data cleaning section of this analysis, the following data had been transformed

* full\_adress was transformed into city
* sqft was transformed into a single number
* parking field showing "no parking" were replaced with ‘0
* bedroom+den replaced with numbers

Once the data was cleansed, our team looked for the factors that determined which variables have the most influence on the volatility of housing market value. We explored feature engineering, data visualizations, linearity, correlation analysis, and modeling to predict the prices and compare them against the observed prices.After conducting all the models and methods, we were able to segregate the influential predictors like the number of bedrooms, bathrooms, sq feet, garage from weaker predictors like parking. Our final objective was to build a predictive model with a high degree of accuracy. There are many data points in the data set but we have cleaned the data and considered only the below-mentioned data points in our analysis.

Here is a metadata for final dataset we used for our investigation:

|  |  |
| --- | --- |
| **Column** | **Description** |
| final\_price | Real estate final price in CAD |
| bedrooms | The number of bedrooms |
| bathrooms | The number of bathrooms |
| sqft | The size of real estate in sqft |
| parking | The number of parking lots |
| description | The overall description of real estate coming from property broker |
| type | Property type, e.g . Detached, Semi-Detached…. |
| city | City name where property is located |

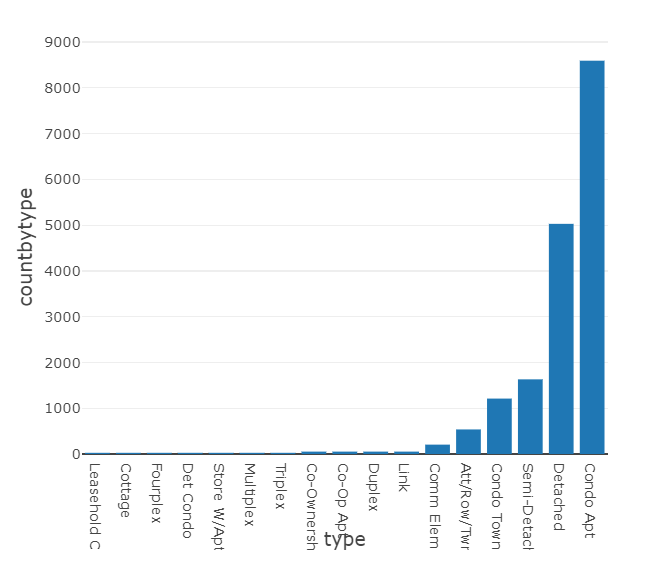
# Analysis

Steps on Spark as follows

* Identify Features/Labels
* Index and vectorize
* Create Pipelines
* Create Training/Test sets
* Create Model
* Train Model
* Test Model

## Feature engineering

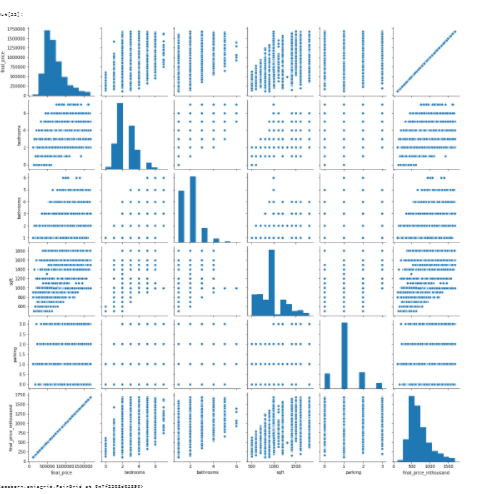
Final price change into final price in thousand dollars to scale down all later data visualization and data modeling**Type:** change type into typeIndex by applying the pyspark ML feature StringIndexer, the pipeline model only take the numeric variables



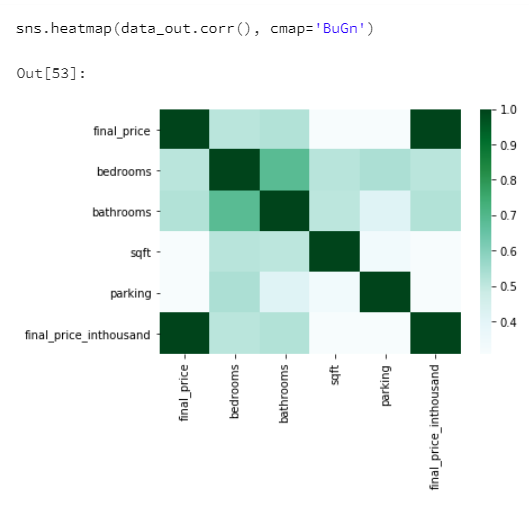
The bar graph of type by counts, there are variety of different condo types (Condo Apt, Co-Op Apt,Det Condo,…) and house types(Semi-Detached, Detached, Link,…). By using this visual, it helps us to identify how to split the dataset into Condo and House datasets properly. Condo dataset includes: Condo Apt, Leasehold Condo, Co-Op Apt, Co-Ownership Apt, Comm Element Condo, Det Condo.

Rest will be listed into house dataset.

## Scatter Plot Visualization

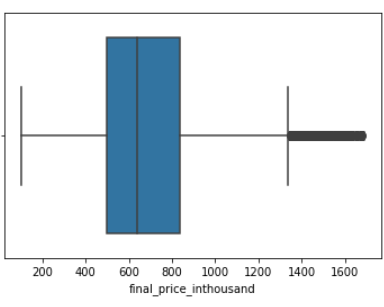


The scatterplot suggests an positive linear relationship between price and bedrooms, higher number of bedrooms of this listing have, the price gets into higher range, same relationship between price and bathrooms. It suggests significant importance of these two variables relate to price. To prove what we analyzed, we plot the correlation between variables to have second proof.



## Correlation Analysis

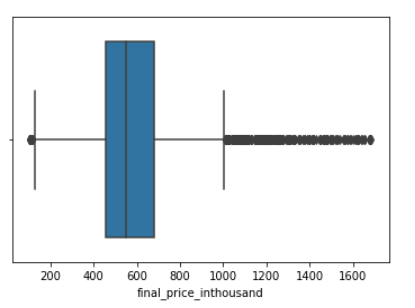
The correlation heatmap shows strong positive correlation between final price and bathrooms, bedrooms, approximately +0.8. More bedrooms mean more money, more bathrooms worth more price. Every room and bathroom accounted into the price. In addition, there are strong positive correlation, approximately +0.8, between bathrooms and bedrooms as well. We knew it is common sense that more bedrooms mean more bathrooms, the data proves what we think, also proves the data is real data reflect the real life. Parking and sqft have moderate positive correlation, approximately +0.6, with final price



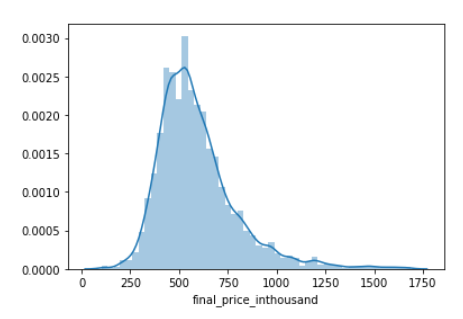
## Price Analysis

After we split dataset into Condo and House separately, as shown above, it shows most condo price falls in range 450 thousand and 700 thousand in Toronto.

### Condo

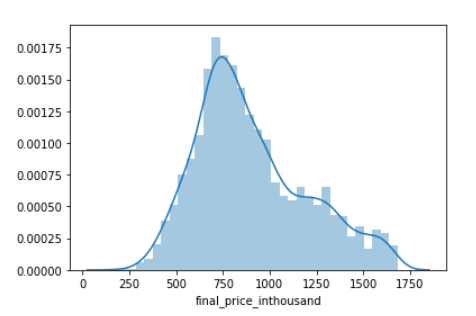


The above boxplot shows the majority of house price falls in range 650 thousand and 1100 thousand in Toronto.



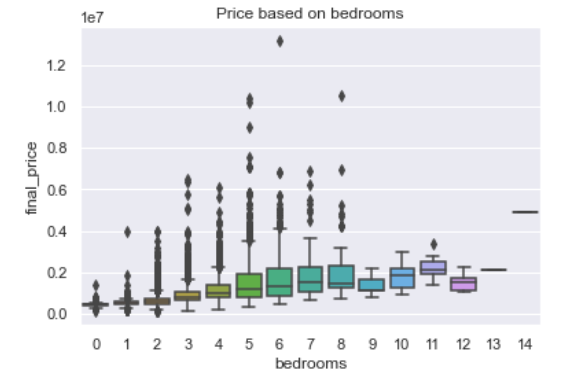
**Figure6 - Histogram Distribution for Condo**

The histogram distribution for condo data, most of data are normally distributed, but a little bit skew to the right.



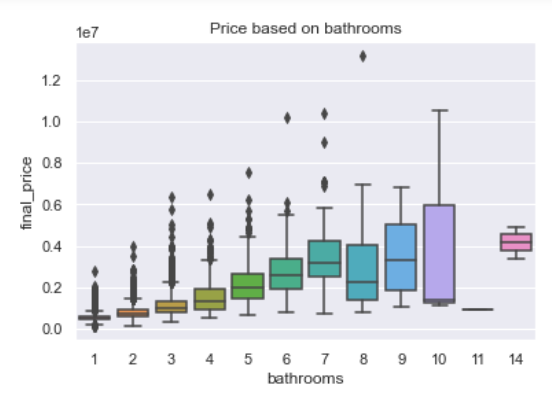
**Figure7 - Histogram Distribution for House**

The histogram distribution for condo data, data are close to symmetric normally distributed



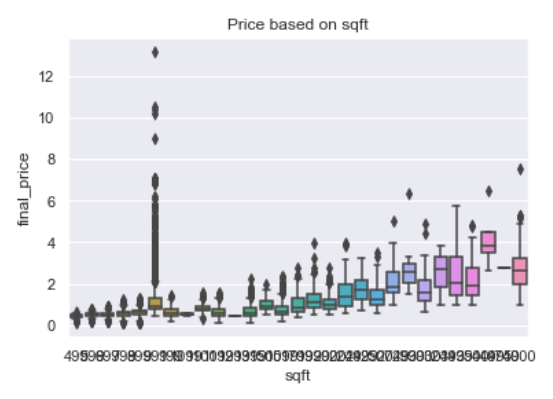
**Figure 8 - Boxplot - Price based on bedrooms**

As bedrooms increases from 1 to 8, price increases. There are many outliers.



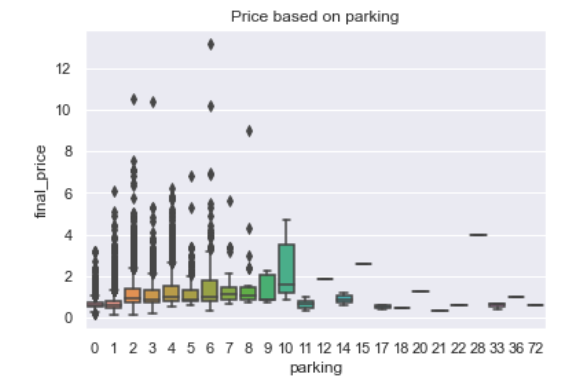
**Figure 9 - Boxplot - Price based on bathrooms**

As the bathroom increases, average of final price increases.



**Figure 10 - Boxplot - Price based on sq ft**

As the sqft increases, average of final price increases



**Figure 11 - Boxplot - Price based on sq ft**

There is no clear trend from increasing of numbers of parking.

# Spark MlLib / Prediction

The dataset had multiple data features that define the housing data, however we used the following attributes as features and attempted to arrive at a prediction for Final price of house,

|  |
| --- |
| Feature |
| Bedrooms |
| Bathrooms |
| Sqft |
| Parking |
| Type |

## Indexing

Since Spark MLlib expects feature columns to be numeric, except for “sqft” column all other columns were alphanumeric. Following are unique values in the other attribuite,

|  |  |  |
| --- | --- | --- |
| Feature | Count | Sample |
| Bedrooms | **36** | '2 + 1 beds'  '1 beds'  '1 + 1 beds'  '4 beds'  '3 beds' |
| Bathrooms | **11** | '2 baths'  '1 baths' |
| Parking | **26** | '1 parking'  'no parking'  '2 parking'  '4 parking' |
| Type | **17** | 'Condo Apt'  'Detached'  'Condo Townhouse'  'Duplex' |

All the feature columns above were indexed using Spark’s *StringIndexer* to convert them to numeric indexes. We also used *OneHotEncoder* to encode the above features into vectors.

Finally *VectorAssembler* was used to assemble the encoded feature columns.

## Spark Pipeline

We used the *Pipeline* feature of Spark MLLib, to create our model for both the models.

## Linear Regression Model

We first attempted to predict the house price using Linear Regression model. The training and test sets were created using data in spark dataframe. Below are the model parameters for creation of the model,

* **aggregationDepth:** suggested depth for treeAggregate (>= 2). (default: 2)
* **elasticNetParam**: the ElasticNet mixing parameter, in range [0, 1]. For alpha = 0, the penalty is an L2 penalty. For alpha = 1, it is an L1 penalty. (default: 0.0, current: 0.5)
* **epsilon**: The shape parameter to control the amount of robustness. Must be > 1.0. Only valid when loss is huber (default: 1.35)
* **featuresCol**: features column name. (default: features, current: features)
* **fitIntercept**: whether to fit an intercept term. (default: True)
* **labelCol**: label column name. (default: label, current: final\_price)
* **loss**: The loss function to be optimized. Supported options: squaredError, huber. (default: squaredError)
* **maxIter**: max number of iterations (>= 0). (default: 100)
* **predictionCol**: prediction column name. (default: prediction)
* **regParam**: regularization parameter (>= 0). (default: 0.0)
* **solver**: The solver algorithm for optimization. Supported options: auto, normal, l-bfgs. (default: auto)
* **standardization**: whether to standardize the training features before fitting the model. (default: True)
* **tol**: the convergence tolerance for iterative algorithms (>= 0). (default: 1e-06)
* **weightCol**: weight column name. If this is not set or empty, we treat all instance weights as 1.0. (undefined)

We decided to consider the prediction as good if the predicted price falls within 50K value of the actual final price. The Regression metrics for the model were as below,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mean Square Error | Mean Absolute Error | Root Mean Square Error | R Squared | Variance |
| 57809517866.79 | 158711.28 | 240436.09 | 0.4134 | 49497311226.73 |

The prediction was about 27% accurate.

## Random Forest Model

We next attempted to predict the house price using Random Forest Regression model to see if we could get a better prediction. The training and test sets we created earlier was again used here.

Due to limitations with the Databricks community edition, we decided to limit the parameters to following,

* Num Trees – 3 , 3
* Max Depth – 2 , 10

Below are the model parameters for creation of the model,

* StringIndexerModel: uid=StringIndexer\_1400a74c3f5b, handleInvalid=keep Out[38]: {Param(parent='StringIndexer\_1400a74c3f5b', name='handleInvalid', doc="how to handle invalid data (unseen or NULL values) in features and label column of string type. Options are 'skip' (filter out rows with invalid data), error (throw an error), or 'keep' (put invalid data in a special additional bucket, at index numLabels)."): 'keep',
* Param(parent='StringIndexer\_1400a74c3f5b', name='outputCol', doc='output column name.'): 'bedroomsIndex',
* Param(parent='StringIndexer\_1400a74c3f5b', name='stringOrderType', doc='How to order labels of string column. The first label after ordering is assigned an index of 0.
* Supported options: frequencyDesc, frequencyAsc, alphabetDesc, alphabetAsc. Default is frequencyDesc. In case of equal frequency when under frequencyDesc/Asc, the strings are further sorted alphabetically'): 'frequencyDesc',
* Param(parent='StringIndexer\_1400a74c3f5b', name='inputCol', doc='input column name.'): 'bedrooms'}

We decided to consider the prediction as good if the predicted price falls within 50K value of the actual final price. The Regression metrics for the model were as below,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mean Square Error | Mean Absolute Error | Root Mean Square Error | R Squared | Variance |
| 95310532437.29 | 184183.06 | 308724.03 | 0.668 | 197139995877.42 |

The prediction was about 25 % accurate.

# Conclusion

We created a prediction model using Spark MLlib using both Linear Regression and Random Forest Regressor . We adjust the prediction for both models to be within 50K of the real price of the houses. We were able to get predictions for both the models , however Linear Regression model gave better prediction than Random forest. The Random Forest model might have performed better if we had used higher depth and increased the number of trees. Due to limitations of databricks community edition, we limited our parameters to num trees (5,10) and depth (1,3).