Final Project Report

ENSF 612 Project

December 15th, 2023

Created By:
Mehreen Akmal
Jenn Bushey
Eric Diep
Hao Liu
Corey Yang-Smith

Table of Contents

Introduction and Motivation

Canada's housing market has boomed but subsequently it has become the eighth least affordable housing market in the world [1]. Historically, these outrageous prices have been localized to only a few major Canadian cities, but recently, due to job opportunities, immigration, market conditions, and the unaffordability of other Canadian cities, Calgary has started seeing a surge in housing prices with increased demand, and a limited and lagging supply market.

With continued interprovincial and intraprovincial immigration, and a large population of young adults being locked out of the housing market due to rising rates and prices, there may not be a slowdown in sight for the average Canadian home-owner.

Our project focuses on predicting the assessed value as a proxy for home sale prices for homes in Calgary. Given various factors about the macro-economic climate (unemployment rates, average house prices, supply, and demand considerations), the community climate (local crime rates, location, community density), and individual housing considerations (land size, land use designation, etc), we aim to predict the housing prices as to provide Calgarians with the additional insight needed to make their purchasing decisions.

Data Collection

Our data collection came primarily from The City of Calgary's Open Data Portal, with a handful of datasets coming from other sources. We were able to either download raw csv files, or link our jupyter notebooks to the data sets through the City's public API, which has allowed us to read in data as json or csv.

We wanted to select datasets that revealed information about housing affordability at various scales:

- **Macro-Scale**: We want data that reveals the economic climate of Calgary at a certain point in time, and how this might influence housing prices.
- **Community-Scale**: We want information that reveals how desirable specific communities may be, and how this might influence housing prices.
- **Individual-Scale**: We want information specific to an individual house, and how these metrics may influence the housing prices.

After extensive research, we ended up with a total of 9 datasets. As to include them in the categories above, we have summarized our datasets and their respective categories in Table 1 below.

Table 1: A summary table of all our datasets categorized by scale-of-impact.

Dataset Category	Dataset Description
Macro Scale	Calgary Census Data
	Prosperous City Indicators
	Interprovincial Migration Data
	Historical CPI
Community-Scale	Historical Community Populations
	Community Crime & Disorder Statistics
	Building Permits
	Development Permits
Individual-Scale	Historical Property Assessments

From these datasets, we identified 16 features to predict our target vector (assessed value) summarized in Table 2 below.

Table 2: Summary table of data collection and feature selection.

#	Target Feature	Category	Dataset
0	Assessed Value	Target Variable	Calgary Assessed Property Values
1	Assessed Year	Economic Indicator	Any Dataset
2	Average Home Price	Economic Indicator	Prosperity Indicator
3	Unemployment Rate	Economic Indicator	Prosperity Indicator
4	Housing Starts	Economic Indicator	Prosperity Indicator
5	Retail Sales	Economic Indicator	Prosperity Indicator
6	Community Population	Group Price Indicator	Census
7	Occupied Dwellings	Group Price Indicator	Census
8	Community Name/Code	Group Price Indicator	Calgary Assessed Property Values
9	Crime Count	Group Price Indicator	Calgary Crime & Disorder Data
10	Resident Count	Group Price Indicator	Historical Community Populations
11	Dwelling Count	Group Price Indicator	Historical Community Populations
12	Land Use Designation	Individual Price Indicator	Calgary Assessed Property Values
13	Property Type	Individual Price Indicator	Calgary Assessed Property Values
14	Land Size	Individual Price Indicator	Calgary Assessed Property Values
15	Migration Count	Economic Indicator	Interprovincial Migration Data
16	Developer Count	Group Price Indicator	Developer Permits
17	Money Spent in Community	Group Price Indicator	Building Permits
18	Inflation Rate (CPI)	Economic Indicator	Historical CPI

Data Inspection and Validation

We inspected each dataset and defined the data types of each of the features. For each dataset we executed the following steps:

1) Imported data into PySpark and converted to RDD
2) Extracted features relevant to our analysis
    a) Only extracted features that had informative values and contained majority of the information (columns with mostly null values were not used as features)
3) Filtered features for data relevant to our analysis (discussed more in depth later):
    a) Data only includes years 2017 - 2019
    b) Data is limited to residential building types and single family homes
4) Checked for missing values and used following two methods to replace the null values:
    a) Dropped Null Values - only few (dropped whole row)
    b) Replaced Null values with average value
5) Checked data quality
    a) Source of Data is credible. Viewed statistics of data to analyze that the range of values along with the average to ensure they were reasonable for each feature
6) Checked for consistency among all datasets for join condition
    a) Ensured all datasets contained the join condition.
        i) Historical Community Populations and Calgary Prosperous Dataset were missing some years of data that was found in PDF format. The data was entered manually into a csv.
    b) Joined macro datasets based on years 2017-2019
    c) Joined community datasets based on the Year and Community Code
    d) Joined macro, community, and individual dataset based on Year and Community Code.

These steps are further explained in detail in sections Data Filtering and Data Transformation.

Data Filtering

An extensive part of our project is related to the filtering of our various datasets. For brevity, we will summarize the common data filtering operations that were present between all our sets, and further explain some of the more complicated, domain-specific filtering that we had to perform.

Common Filtering

Common to all our sets, we observed null values and either dropped them, or replaced them with the average value for that column. For most of the datasets we did not make any attempts to rectify nulls through replacement with average values or otherwise; rather, due to the size of our dataset and the scarcity of null values, we are confident that dropping these will not impact our results; however, for the building permits datasets we replaced null values with the mean of the column.

We also dropped the extraneous columns that were not relevant to our datasets. As highlighted in Table 2, there are 18 key features to predict our assessed values. Due to the number of our datasets, we had to maintain a few columns throughout our datasets as sets to join on. For these operations, we chose YEAR as a column to join our Macro-Scale data on, and the YEAR and COMM_CODE columns to join the Community-Scale, and Individual-Scale datasets on. As such, these columns are common throughout the datasets that belong to those categories.

Domain Specific Filtering

The backbone of our data is the assessed property values dataset, which carries information related to the land use designation of each land parcel. Land use designations allow the city to classify certain housing types by various factors (such as if they have a side yard, where their garage access is, if it is single detached vs multi-residential, etc).

For the purposes of aiming for a high accuracy, and then expand in the future if desired, we have filtered out much of the land use types that are not relevant to our analysis. We targeted single family detached housing primarily, but with a few caveats. Simply put, we have excluded any non-residential housing, and have excluded mobile homes, mixed-use residential and multi-residential housing. This leaves us with single-detached houses of various sizes - some with secondary suites, rear garages, zero-lot-lines, and more. We have kept low-density housing (duplexes, townhouses, etc) as these are housing types accessible and desirable for the average home-buyer.

Additionally, we noticed some great outliers within our dataset that correspond to invalid points (i.e. an entire neighborhood development being classified as a single land use, before it gets repartitioned). As such, we have defined minimum and maximum bounds on housing size and prices for what a “reasonable” house would be in Table 3 below.

Table 3: Minimum and maximum bounds for individual housing attributes.

	Minimum	**Maximum**
House Size	50sm (~540 sq ft)	1000sm (~10 750 sq ft)
Assessed Value	$50,000	$3,000,000

Data Transformations

After filtering our separate datasets, we then began the process of consolidating our various datasets into a final dataset to be used in our machine learning model.

We needed to convert the RDDs to PySpark dataframes as our data requires OneHot encoding and the PySpark library OneHotEncoder[1] takes a *pyspark.sql.DataFrame* as the dataset parameter. To reduce the number of data type transformations, we used the PySpark machine learning models that also take dataframes as an input.

Each individual dataset dataframe was cached after it was created to speed up the notebook when we combine the datasets and perform EDA on them later.

We also released the cache by calling unpersist on dataframes as needed to release earlier versions as needed and to free up memory for our models.

Finally, we performed various data transformations in order to prepare our dataset for machine learning.

Data Combination

As previously mentioned, we broadly grouped our datasets into three categories: Macro-Scale, Community-Scale, and Individual-Scale datasets. As such, we began our data-merging by joining our single datasets into their categorical counterparts.

We first combined our Macro-Scale datasets on the Year column, and joined the Community-Scale datasets on the Year and Community_Code columns. Our Individual-Scale dataset did not need any merging as it only consists of our Calgary Assessed Property Values.

Afterwards, we took these intermediate datasets and combined them to create our desired final_df that we have described in Table 1.

Data Transformations

To further process and transform our data for use in the PySpark machine learning models, we needed to use **StringIndexer**, **OneHotEncoding, VectorAssembler,** and **Pipeline.**

Scaling

We explored the need to scale our datasets but we ultimately selected machine learning models that do not require scaling since each feature is considered individually at each stage of the tree-based models and each feature receives a unique weight in the linear regression model.

StringIndexer[2]

[1] OneHotEncoder - PySpark Documentation
[2] StringIndexer - PySpark Documentation

We have many string data types that must be converted to a numerical value in order to be processed by our machine learning models. The following columns: “Community Code, Land Use Designation” were converted to numeric values using StringIndexer.

OneHotEncoder[3]

Following the StringIndexer, we applied OneHotEncoder to normalize the importance of the “Community Code, Land Use Designation” columns and convert the numerical data to encoded vectors for use in our machine learning models.

VectorAssembler[4]

To combine our columns with numeric data and the encoded columns and to produce the format required for the machine learning models, we needed to use the VectorAssembler to combine the columns and OneHot encoded data to the vector format taken by the machine learning models.

Pipeline[5]

To combine the stages of our transformations with our models, we used a pipeline to combine all data transformation steps with the model training.

[3] OneHotEncoder - PySpark Documentation
[4] VectorAssembler - PySpark Documentation
[5] Pipeline - PySpark Documentation

Exploratory Data Analysis

After data preprocessing was completed, we performed exploratory data analysis on our combined dataset to determine patterns, relationships and insights within the dataset. This involved Data Summarization and Data Visualization.

Data Visualization

First, we conducted a correlation analysis to assess the relationships between all the features. We then visualized the data through heatmap, pairplot on one of the datasets and scatter plot to identify any patterns or clusters that may exist in the dataset.

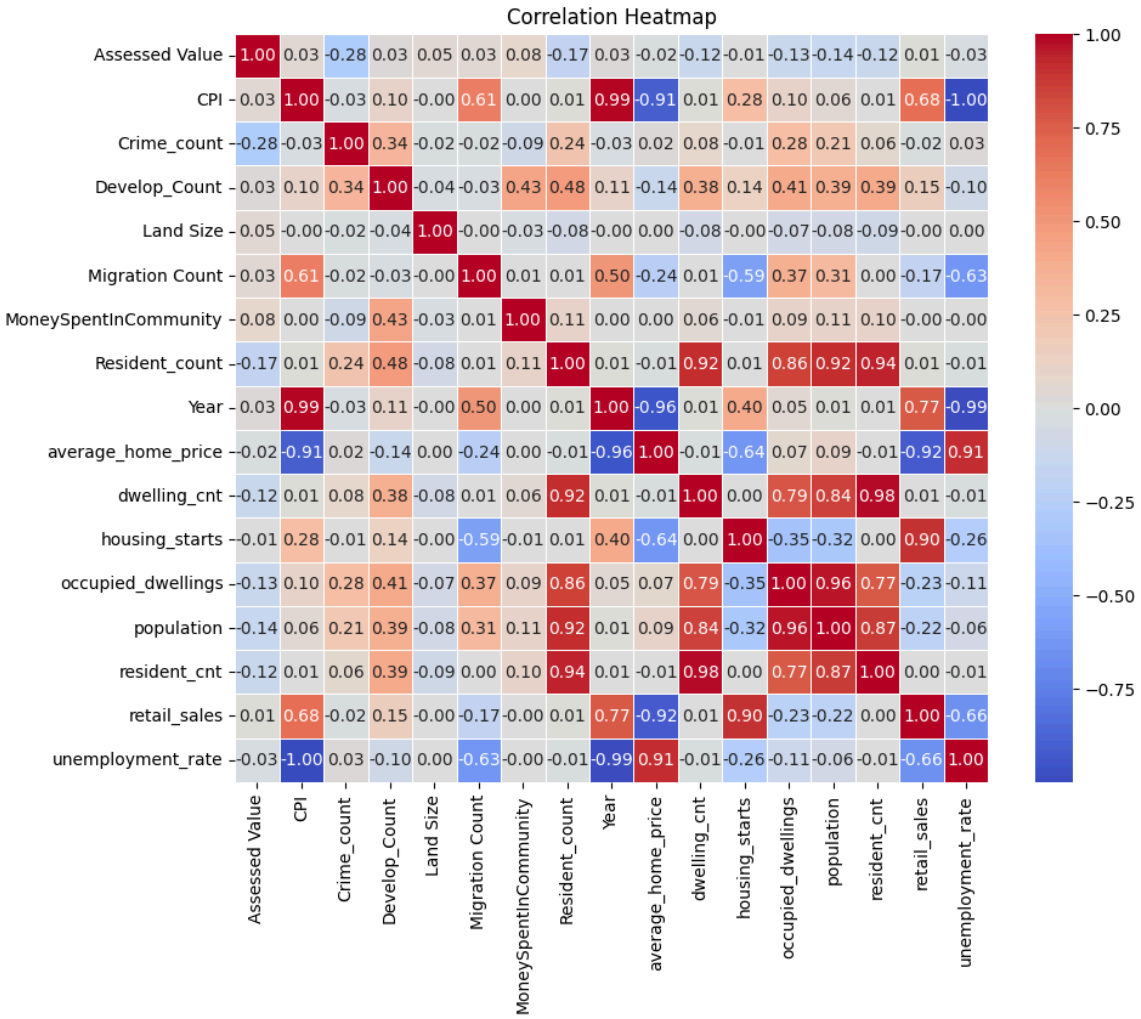

Figure 1: Correlation Heatmap for the dataset

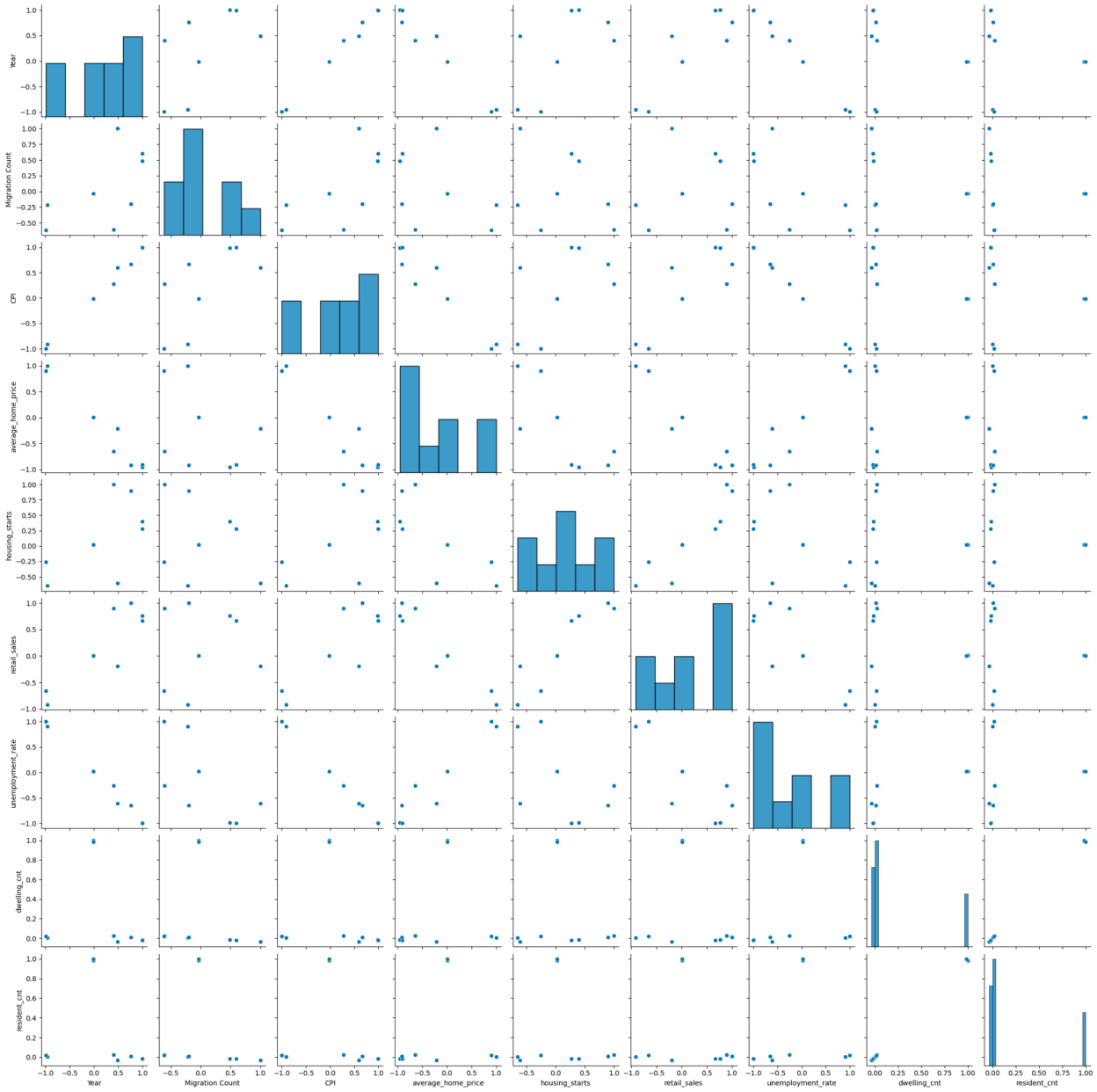

Figure 2: Pair plot for macro-scale dataset

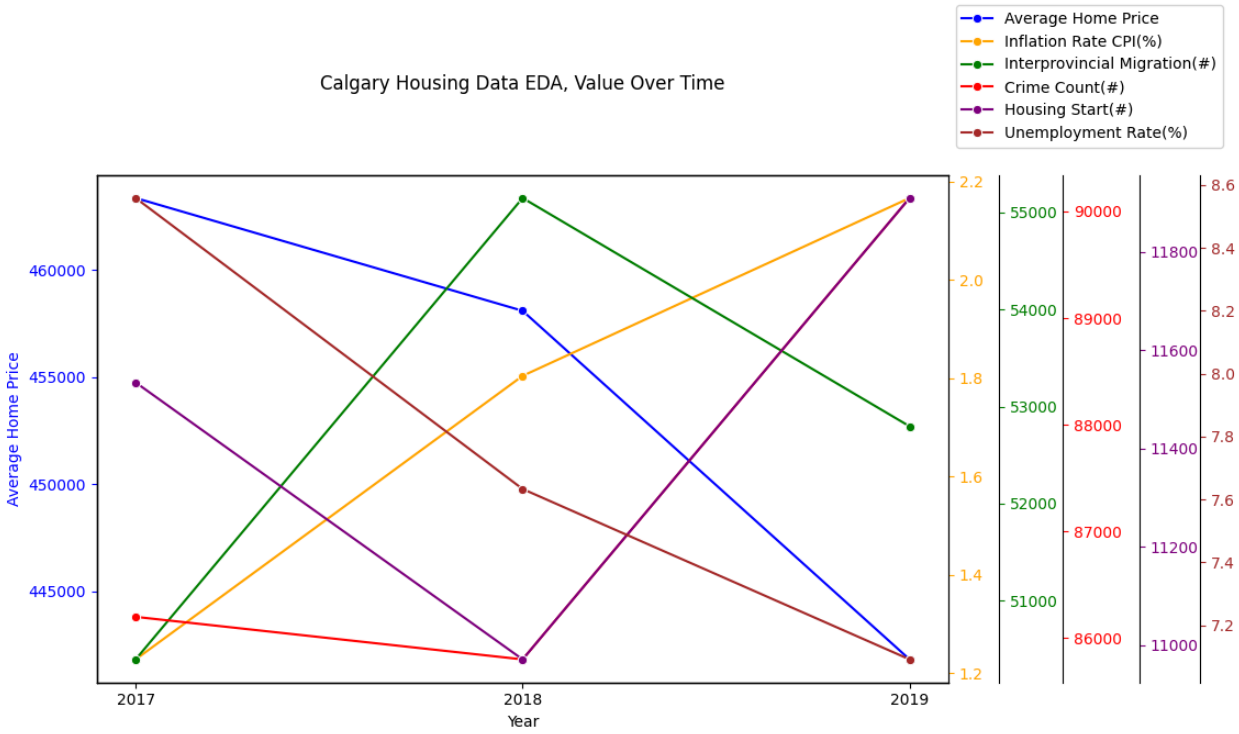

Figure 3: Key feature plot over time

Model Building & Results

After our preliminary data analysis and data transformation was completed, we were ready to select our regression models to begin training. While our dataset is likely non-linear, we experimented with a variety of models to develop a good understanding of how these different models interact with our data. We will talk about the models that we chose, why we chose them, and why we think they performed how they did.

Once our model exploration is done, we will select the best performing model to further finetune using various machine learning techniques such as Cross Validation and Grid Search to hone in on the optimal hyperparameters to use with our model.

	Train RMSE	Test RMSE	Train MAE	Test MAE	Train R2	Test R2
Decision Tree	193151.504037	194097.486527	122116.927474	122248.748748	0.300555	0.303110
Random Forest	189314.015297	190508.029579	116548.805825	117120.821782	0.328072	0.328647
Linear Regression	157535.321973	157535.321973	88922.043610	88922.043610	0.540929	0.540929

Figure 4: Results table summary from final notebook.

Decision Tree Regression[6]

Decision Tree Regression is the first non-linear model we chose for the following reasons:

- It is a simple and easy to understand method that does not require heavy tuning.
- Feature scaling is not required.

```
Training R2 score: 0.3005552149729088
Training RMSE: 193151.5040371326
Training MAE: 122116.92747411721

Testing R2 score: 0.3031102041363577
Testing RMSE: 194097.4865268364
Testing MAE: 122248.74874840726
```

The R2 scores for both training and testing are very low. It may be because the features selected may not be relevant. A further feature extraction analysis may be needed to select the correct features.

```
+-------+------------------+------------------+
|summary|    Assessed Value|        prediction|
+-------+------------------+------------------+
|  count|            737670|            737670|
```

[6] Decision Tree Regressor - PySpark Documentation

```
|   mean|  522150.778554096|522150.77855409146|
| stddev|230951.96963214135|126614.60496444722|
|    min|           50000.0| 412988.2230689297|
|    max|         3000000.0| 1360903.494623656|
+-------+------------------+------------------+
```

The mean prediction value seems to be reasonably inline with the assessed value , while the min and max are very far off. It appears that the model performs well with the mean value. Belows are the potential reasons for this phenomenon:

- Bias towards average values: it is possible that the training data has more average-valued instances that make the tree biased toward these values.
- Decision tree does not handle extreme values well. The difference in house prices are quite large. There are few houses in the range of millions-dollar while the majority are in the hundreds of thousands range. The million-dollar houses would act as the outliers that impact the performance of the decision tree negatively.
- Data distribution is poor, which is skewed toward the average house price.

Possible solution: preprocess the data to better handle the outliers

Random Forest Regression[7]

Random Forest Regression is the second non-linear model we chose for the following reasons:

- It is an ensemble method that creates multiple trees to predict the results. This prevents overfitting and allows to achieve better results than the Decision Tree Regression model.
- The Gradient Boosting Regression model outputs better results but takes longer to train the model so it is a better idea to start with random forest.
- Lastly, this model is able to capture complex relationships.

```
Training R2 score: 0.328071934057403
Training RMSE: 189314.01529663897
Training MAE: 116548.80582546536

Testing R2 score: 0.32864712554234077
Testing RMSE: 190508.02957907395
Testing MAE: 117120.82178238743
```

Based on the validation scores, it is evident that the random forest regression model is a poor fit for our dataset. For the best scores, we want the smallest error and an R2 score close to 1. This result was expected as the previous tree-based model also performed poorly.

```
+-------+------------------+------------------+
|summary|    Assessed Value|        prediction|
+-------+------------------+------------------+
|  count|            737670|            737670|
```

[7] Random Forest Regressor - PySpark Documentation

```
|   mean|  522150.778554096| 522154.0258744721|
| stddev|230951.96963214135| 94844.04682258982|
|    min|           50000.0| 384935.7312727134|
|    max|         3000000.0|1221118.0539344735|
+-------+------------------+------------------+
```

Since random forest is a collection of decision trees, the reasons for poor performance of random forest would be similar to the decision trees mentioned previously. They are biased toward average values and poor at handling outliers.

Linear Regression[8]

Linear Regression models are fast to train and predict compared to other models, work well with high-dimensional large datasets and are easy to understand. For these reasons, we implemented the linear regression model as the linear model on our dataset. We initially ran the default base model and found that this model outperformed our two tree-based models. As such, we selected this model to perform more extensive tuning on to try and optimize for best accuracy. Additionally, we changed the original base linear regression model to elasticnet to allow us to tune the models' hyperparameters.

We implemented cross validation and parameter grid search to find the best fitting model and found that maxIter=10 and regParam=0.05 helps our model perform the best.

```
Training R2 score: 0.5409286686624446
Training RMSE: 157535.3219728833
Training MAE: 88922.0436103032

Testing R2 score: 0.5409286686624446
Testing RMSE: 157535.3219728833
Testing MAE: 88922.0436103032

+-------+------------------+------------------+
|summary|    Assessed Value|        prediction|
+-------+------------------+------------------+
|  count|            183837|            183837|
|   mean| 522372.1751116478|522148.61206466093|
| stddev|232508.61007913278|169805.12661882016|
|    min|           51500.0|  226220.404661756|
|    max|         3000000.0|1840984.2044857088|
+-------+------------------+------------------+
```

Based on the validation scores, it is evident that the linear model is not a good fit for our dataset. For the best scores, we want the smallest error and an R2 score close to 1. This result was expected as our dataset consists of many uncorrelated features which can be seen from the heatmap plotted during EDA.

[8] Linear Regression - PySpark Documentation

Best Model

Based on our preliminary model training, linear regression performed much better than the other models. As such, we have performed cross validation and grid search in order to select the final parameters for our best model.

We searched through the following parameters for the following values:

- maxIter: [10, 20, 30]
- regParam: [0.01, 0.05, 0.1]

We have found that the optimal parameters for our linear regressor is **maxIter=10** and **regParam=0.05**

Results

The better performance of linear regression among the ML model tested suggests that there is a linear relationship between some of the independent variables in the dataset and the dependent variable, the assessed housing price. This indicates that as the feature values increase or decrease, the assessed housing prices also increase or decrease proportionally.

Most of the features included in our analysis are at the macro and community scale, while our target assessed value is for individual houses. Although macro and community-scale data are more readily available than individual-scale data, their effectiveness in predicting the value of individual houses is significantly limited, despite the various types of machine learning models we have tried. There are only a few features at the individual scale available for our analysis, which becomes a limiting factor for the accuracy of our model's predictions.

Additionally, it is likely that there is a more complex, and potentially non-linear relationship between our attributes and the assessed value. As such, in order to perform better we could try various other models that may be better suited for our dataset. It is clear that tree-based models may not be well suited for our dataset due to the presence of outliers, but other non-linear models such as Polynomial Regression or Support Vector Machines may be promising options.

In conclusion, we were able to create a somewhat acceptable model by tuning a linear regression model for our datasets, in order to predict assessed values for homes within Calgary. Model building can be an iterative process as one works through different models, finding the optimal parameters to use for a given problem and dataset. We have seen this process unfold in our own problem-space as we have tried different models and further tuned to improve accuracy. If we were to explore this problem further, we would likely choose different non-linear models to try and achieve a better accuracy. In addition, we could perform cross validation and grid search to our tree-based models, and future non-linear models to hone in on ideal parameters. We could also perform more extensive feature engineering to identify the critical attributes for our dataset, as to remove some of the noise from having many different attributes.

References

[1] Z. Demarco, “Canada has one of the least affordable housing markets in the world: Study,” Urbanized, https://dailyhive.com/vancouver/canada-eighth-least-affordable-housing-market (accessed Dec. 14, 2023).