

Wondering if this is a good time to buy or sell a property?

...

Using Python, PostgreSQL, AWS RDS, Machine Learning and Tableau,
our dashboard should help users with making that decision.

Project link and team presentation

Github: https://github.com/grittins/Team1_Project

Team members:

- Anne Lecomte @padawanne - anne-lecomte@hotmail.com
- Derek Mears @mearsdj - derek.j.mears@gmail.com
- Rezwan Ferdous @grittins - grittins1@gmail.com
- Shivali Sahai @shivalisahai - shivali.sahai@gmail.com
- Wayne Jin @jinwei1207 - waynejin0110@gmail.com

Communication Protocols:

We are communicating through Slack and working together on Tuesdays and Thursdays on Zoom (within the class hours and afterwards using another Zoom room).

We are also meeting once a week, when it fits our respective schedules.

Predicting Toronto Housing Market

Our project aims at predicting Toronto house prices and with that; to consider the best timing to buy or sell a property.

This question impacts every Torontonians in some way, regardless of their status or current employment situation. We chose to investigate this as we had been wondering how an average person could afford a house in Toronto in the current economy. The goal is therefore to predict average house prices while taking into account the type of houses (attached/detached/condo), the location, the timing, interest rates, inflation rate and recession periods.

This could be used by governments and policy makers looking for ways to balance the growth of house prices, but also by realtors, investors, or any individual interested in the Toronto housing market.

Table of contents

- Project Overview
- Overview of the tools
- Datasets Description and Sources
- Before/After of the primary dataset
- Preparing the data
- ERD & Database
- Machine Learning Model
- Dashboard Overview

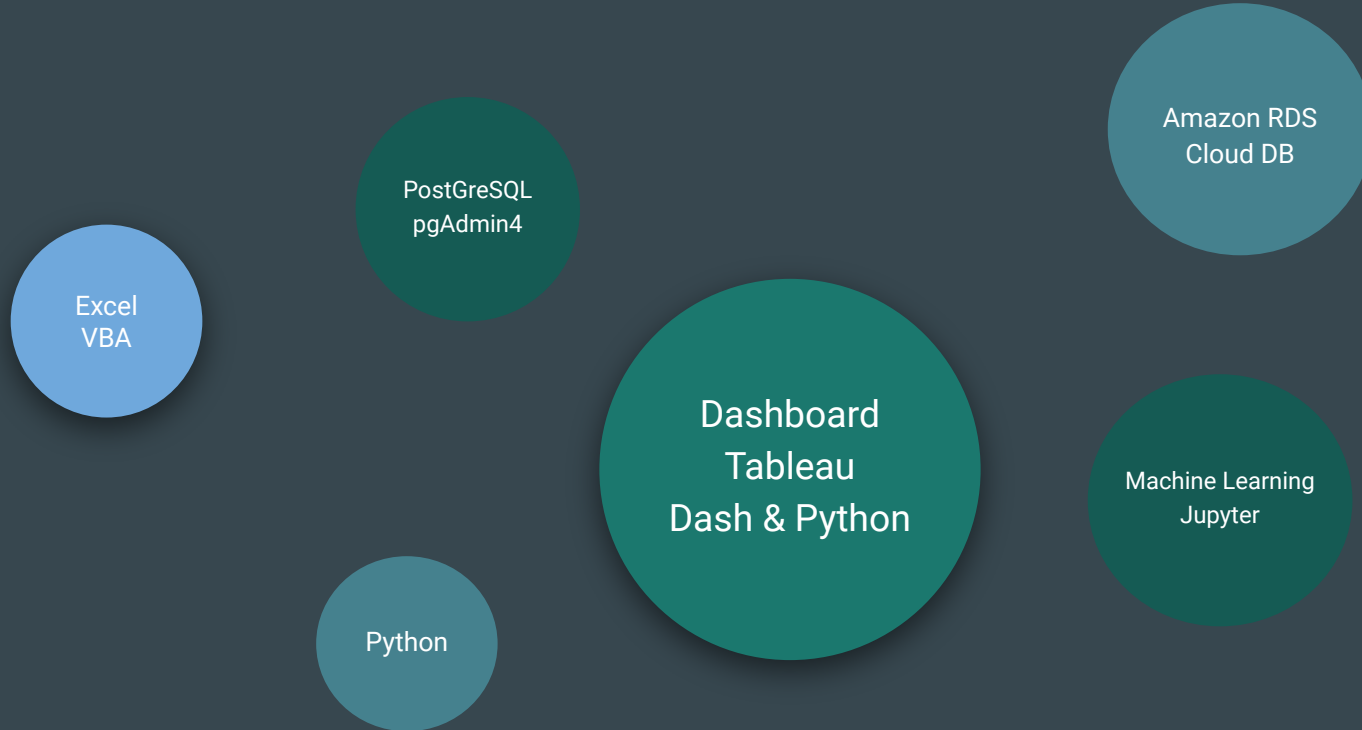
Project overview

Using datasets capturing Toronto House Prices of the last 21 years (2001-2022), Canada Recession Indicator, Mortgages Rates, Inflation Rates, we will build a ML model to predict the prices for the next year.

Once the data is cleaned, a PostGreSQL database is set up. We will then connect it to the Amazon RDS cloud in order to use the data within our ML model.

We will run the predictions, export the results and finally use Tableau to create an interactive dashboard for users to test and learn if the timing fits their budget.

Overview of the tools



Datasets Description and Sources

- [TRREB](#) (Toronto Regional Real Estate Board) quarterly house sales reports, from 2001 to 2022.
 - We were given access to internal csv reports that are not found on their website, and these are the files we are using. We first started working with the pdf files found on their website and converting these to excel.
- [Bank of Canada](#) Mortgage Rates
- Canada Recession Indicator
- [Toronto Neighbourhoods GeoJSON](#)

What the raw data looked like -vs- After Conversion

Quarterly Community Tables_Desktop								
Parameter values								
Name	Value							
Date	Q1 2011							
Geography	All values							
HomeType	Detached							
DataGrid1								
Area	Municipality	Community	Sales	Dollar Volume	Average Price	New Listings	Average SP/LP	Average DOM
Durham	Ajax	Ajax - Unknown	0	\$0	\$0	1	0.0 %	0
Durham	Ajax	Central	57	\$17,524,100	\$307,440	94	98.3 %	22
Durham	Ajax	Central East	10	\$3,962,000	\$396,200	16	97.6 %	34
Durham	Ajax	Central West	56	\$20,712,200	\$369,861	91	98.5 %	32

The data we are using was presented into individual quarterly reports that we converted into a single csv file, using Python (pandas and openpyxl) and Excel VBA.

Below is a snippet of our final csv file, containing 21 years of house prices, by location and house type.

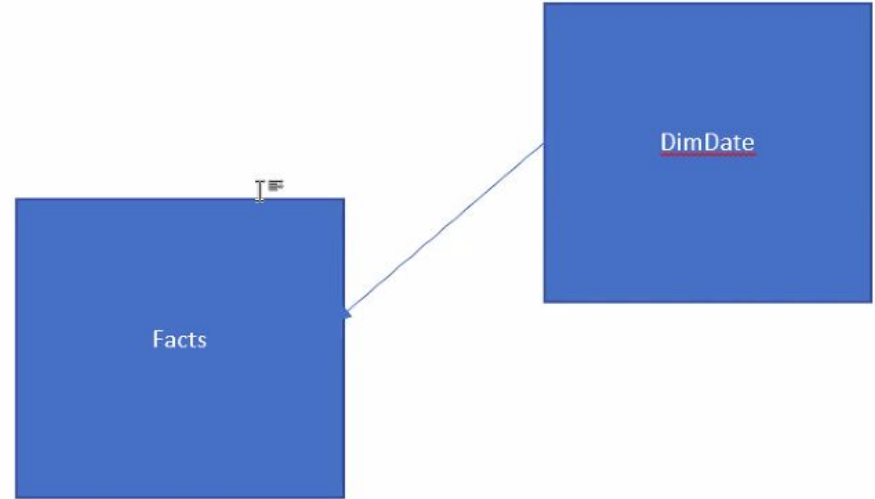
	Area	Municipal	Communi	Sales	Dollar Vol	Average P	New Listir	Average S	Average C	Date	Building_Type
0	Durham	Ajax	Central	13	2294900	176530.8	16	0.981408	41.76923	2001Q1	Att/Row/Twnhouse
1	Durham	Ajax	Central W	8	1500900	187612.5	10	0.981944	36.875	2001Q1	Att/Row/Twnhouse
2	Durham	Ajax	Northwes	0	0	0	2	0	0	2001Q1	Att/Row/Twnhouse
3	Durham	Ajax	South Eas	0	0	0	4	0	0	2001Q1	Att/Row/Twnhouse
4	Durham	Brock	Sunderlan	1	0	0	0		0	2001Q1	Att/Row/Twnhouse
5	Durham	Claringtor	Bowmanv	21	2739400	130447.6	32	0.976857	27.42857	2001Q1	Att/Row/Twnhouse
6	Durham	Claringtor	Courtice	25	3457600	138304	34	0.981166	29.28	2001Q1	Att/Row/Twnhouse
7	Durham	Oshawa	Centenni	1	0	0	0		0	2001Q1	Att/Row/Twnhouse
8	Durham	Oshawa	Central	2	0	0	9		0	2001Q1	Att/Row/Twnhouse

Preparing the data

With the help of our amazing TA Sina, we've decided a date format to be consistent within our datasets.

The chosen format was “yyyyq”, which would be “20224” for the last trimester of 2022.

- 1- Every table at a quarter grain
- 2- Every fact table will have one column called -> YearQuarter-> yyyyq
- 3- One date dimension that includes-> yyyyq, yyyy, qq



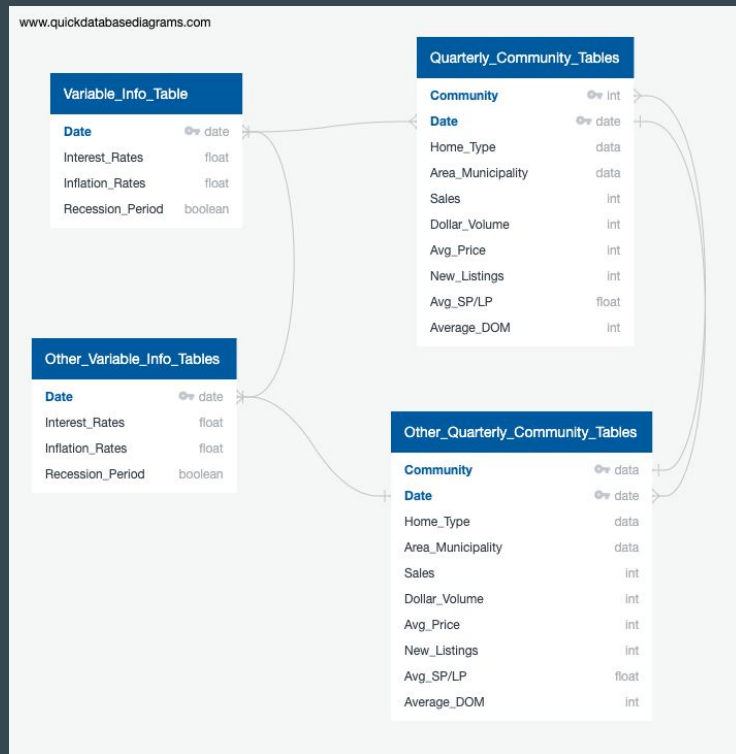
ERD & Database

On the right, you can see the ERD we based our database on.

We started locally with PostgreSQL to sketch the database we desired, then we connected it to an online database using AWS RDS.

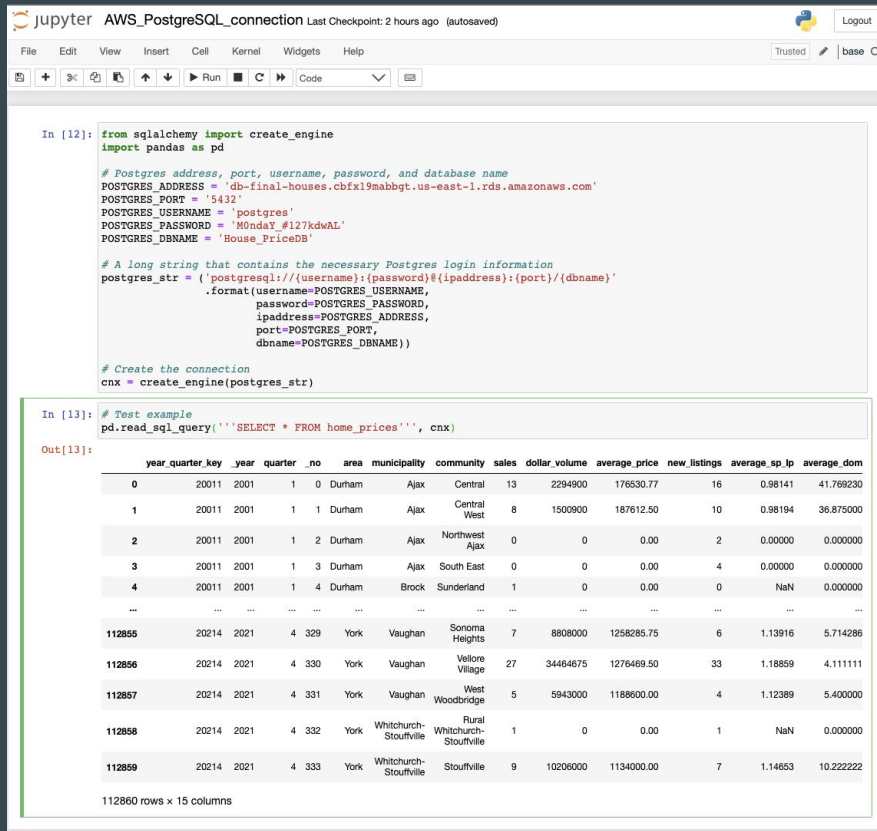
This is how our Machine Learning Model will access the data and predict the house prices for the next 4 quarters.

Since our primary dataset is quarterly based, we have had to adapt the other sets.



AWS - Jupyter Connection

We linked the database to Jupyter Notebook as seen here.



The screenshot shows a Jupyter Notebook interface with the title "AWS_PostgreSQL_connection". The notebook contains two code cells. The first cell, labeled "In [12]:", imports the necessary libraries and sets up the database connection parameters. The second cell, labeled "In [13]:", tests the connection by running a SQL query. The output of the query is displayed as a table with 15 columns and 112860 rows.

```
In [12]: from sqlalchemy import create_engine
import pandas as pd

# Postgres address, port, username, password, and database name
POSTGRES_ADDRESS = 'db-final-houses.cbfxi9mabbgt.us-east-1.rds.amazonaws.com'
POSTGRES_PORT = 5432
POSTGRES_USERNAME = 'postgres'
POSTGRES_PASSWORD = 'M0ndaY.#127kdwAL'
POSTGRES_DBNAME = 'House_PriceDB'

# A long string that contains the necessary Postgres login information
postgres_str = ('postgresql://{username}:{password}@{ipaddress}:{port}/{dbname}'
               .format(username=POSTGRES_USERNAME,
                       password=POSTGRES_PASSWORD,
                       ipaddress=POSTGRES_ADDRESS,
                       port=POSTGRES_PORT,
                       dbname=POSTGRES_DBNAME))

# Create the connection
cnx = create_engine(postgres_str)
```

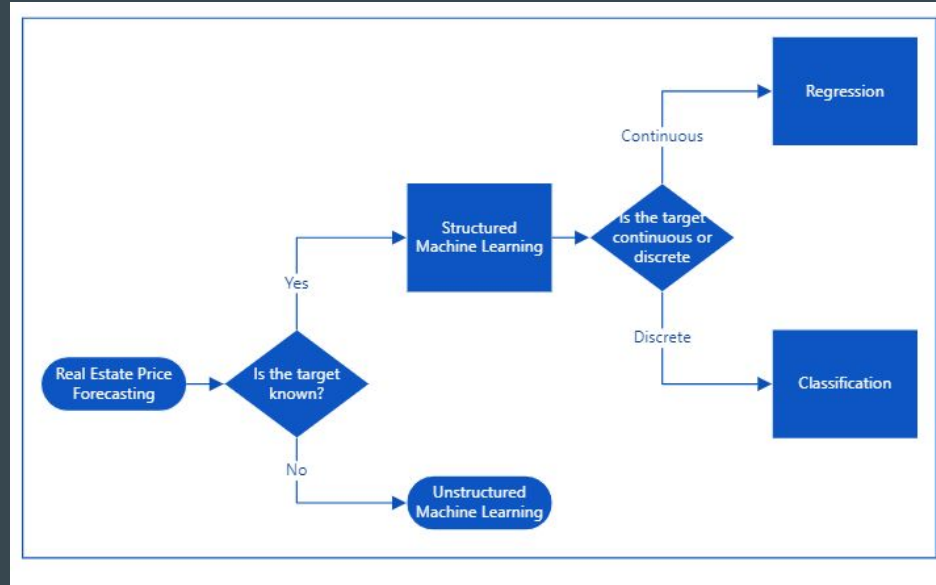
```
In [13]: # Test example
pd.read_sql_query('SELECT * FROM home_prices', cnx)
```

Out[13]:

	year	quarter	key	year	quarter	_no	area	municipality	community	sales	dollar_volume	average_price	new_listings	average_sp_lp	average_dom
0	20011	2001		1	0	Durham	Ajax	Central	13	2294900	176530.77	16	0.98141	41.769230	
1	20011	2001		1	1	Durham	Ajax	Central West	8	1500900	187612.50	10	0.98194	36.875000	
2	20011	2001		1	2	Durham	Ajax	Northwest Ajax	0	0	0.00	2	0.00000	0.000000	
3	20011	2001		1	3	Durham	Ajax	South East	0	0	0.00	4	0.00000	0.000000	
4	20011	2001		1	4	Durham	Brock	Sunderland	1	0	0.00	0	NaN	0.000000	
...	
112855	20214	2021		4	329	York	Vaughan	Sonoma Heights	7	8808000	1258285.75	6	1.13916	5.714286	
112856	20214	2021		4	330	York	Vaughan	Vellere Village	27	34464675	1276469.50	33	1.18859	4.111111	
112857	20214	2021		4	331	York	Vaughan	West Woodbridge	5	5943000	1188600.00	4	1.12389	5.400000	
112858	20214	2021		4	332	York	Whitchurch-Stouffville	Rural Whitchurch-Stouffville	1	0	0.00	1	NaN	0.000000	
112859	20214	2021		4	333	York	Whitchurch-Stouffville	Stouffville	9	10206000	1134000.00	7	1.14653	10.222222	

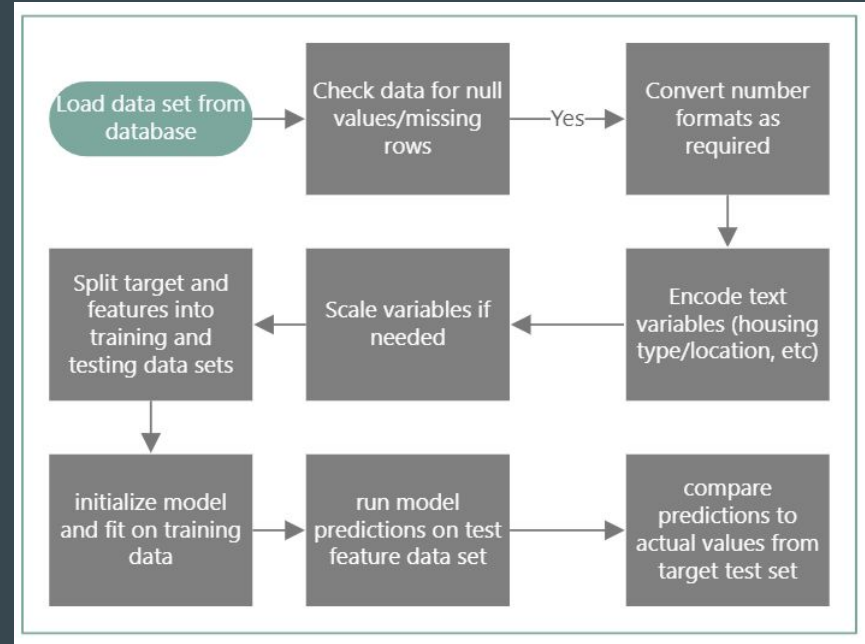
112860 rows x 15 columns

Machine Learning



This flowchart indicates how we decided to go with a supervised Machine Learning, using sklearn linear model for price prediction.

The flowchart will guide us for building our ML model and throughout our analysis.



Linear Regression approach will be modelled for the factors affecting house prices. For the model, the following X-variables will be taken into account.

- Time Period
- House-Type
- Location
- Interest Rates
- Inflation Rate
- Recession Period

A simple database diagram was sketched for the initial understanding of how the datapoints are connected (as seen on our ERD).

Data Preprocessing

The preprocessing can be seen here.

It includes these steps:

- strip symbols from numeric columns, including ',', '\$', and '%'
- convert columns to float
- filter data to for 'Toronto' only (i.e. use column 'area' and filter for 'Toronto')
- filter to include only rows where Average_Price>0
- check for additional na rows (there aren't any)
- group data by community and building type and count rows. if less than 30 rows, filter out.

```
Untitled-1 ML_Analysis.py x ID ? ↓ ↑ ↺ ↻ □
Users > wayne > Desktop > Analysis_Project > Final Project > ML_Analysis.py > ...

10 df = df_in.copy()
11
12 for col in float_cols:
13     for s in ['$', ',', '%', ' ']:
14         df.loc[:, col] = df.loc[:, col].str.replace(s, '')
15
16 df.loc[:, col] = df.loc[:, col].astype('float')
17 for col in date_cols:
18     df.loc[:, col] = df.loc[:, col].apply(lambda x: pd.Period(x))
19 #add in year and quarter cols, drop original date column
20 if 'Date' in df.columns:
21     df['Year'] = df.Date.dt.year
22     df['Quarter'] = df.Date.dt.quarter
23 return df
24
25 cleaning raw data (type conversion and symbol removal from strings)
26 f_all = frame_format(df_all, ['Average Price', 'Average SP/LP'], ['Date'])
27 f_all.set_index('Date', inplace=True)
28 df_all.drop('Date', inplace=True, axis=1)
29 filter dataset to toronto only, require avg price to >0, and take only community/date groups that have
30 more than 30 quarters of data available
31 f_all_toronto = df_all.loc[df_all.Area=='Toronto',:]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
df_all = frame_format(df_all, ['Average Price', 'Average SP/LP'], ['Date'])
File "/Users/wayne/Desktop/Analysis_Project/Final Project/ML_Analysis.py", line 14, in frame_format
df.loc[:, col] = df.loc[:, col].str.replace(s, '')
File "/Users/wayne/opt/anaconda3/lib/python3.8/site-packages/pandas/core/indexing.py", line 961, in __getitem__
return self._getitem_tuple(key)
File "/Users/wayne/opt/anaconda3/lib/python3.8/site-packages/pandas/core/indexing.py", line 1140, in _getitem_tuple
return self._getitem_lowerdim(tup)
File "/Users/wayne/opt/anaconda3/lib/python3.8/site-packages/pandas/core/indexing.py", line 867, in _getitem_lowerdim
section = self._getitem_axis(key, axis=i)
File "/Users/wayne/opt/anaconda3/lib/python3.8/site-packages/pandas/core/indexing.py", line 1282, in _getitem_axis
return self._get_label(key, axis=axis)
File "/Users/wayne/opt/anaconda3/lib/python3.8/site-packages/pandas/core/indexing.py", line 1153, in _get_label
return self.obj.xs(label, axis=axis)
File "/Users/wayne/opt/anaconda3/lib/python3.8/site-packages/pandas/core/generic.py", line 3849, in xs
return self[key]
File "/Users/wayne/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py", line 3505, in __getitem__
indexer = self.columns.get_loc(key)
File "/Users/wayne/opt/anaconda3/lib/python3.8/site-packages/pandas/core/indexes/base.py", line 3623, in get_loc
raise KeyError(key) from err
KeyError: 'Average Price'
© (base) a0@mei ~ %
```

Feature engineering and selection

As economically relevant macro variables for forecasting housing prices, we selected

- 5 year mortgage rates
- Canadian CPI inflation
- a Canadian recession indicator.

Rather than encoding community (a text variable representing a contiguous geographical area of Toronto) with dummy variables, we generate forecasts for each community individually, under the assumption that pricing dynamics are sufficiently different by community to justify that approach. We might need to test to confirm. Plotted autocorrelation in price time series, saw evidence of statistically significant autocorr back 6 quarters. Also plotted partial autocorr, which showed significant at 1 and 2 previous quarter lags.

Description of data split into training and testing sets

Since we are dealing with time series, we can't employ a random splitter (i.e. sklearn test_train_split).

Instead we tried sktime temporal_test_train_split, but it had difficulties splitting our data accurately.

Ultimately we manually split the data using a cut off date of 2018Q4, so used 2001Q1-2018Q4 for training and 2019Q1-2021Q4 for testing. We would prefer to re-estimate entire models on rolling quarterly basis and forecast 4 quarters forward and test over longer period, but the feasibility is to be decided.

Description of model choice

Work in progress.

The initial model is **sktime NaiveForecaster**, this will be the baseline against which we can measure other models. The model parameters chosen means the model uses a seasonal window and chooses the most recent training value. The benefits of current model are simplicity and interpretability, limitations are that it's best guess for the future are exactly equal to past values. So, as an aside, this model is no good for actual use, but will be a useful baseline to compare others with.

Change in model choice

1. We tried **Facebook Prophet**, but it is not optimized for low frequency data (i.e. hard to work with and hard to get cross validation results because the auto calculators require some inputs to be specified as pandas time deltas, which can't be larger than days, and distance between quarters is variable.
2. We tried **sklearn linear regression**, but it was difficult to get look ahead forecasts without generating large dataframes of inputs.

Description of model training

Work in progress.

Trained on data split as described above. We only ran **naiveoforecaster** so far.

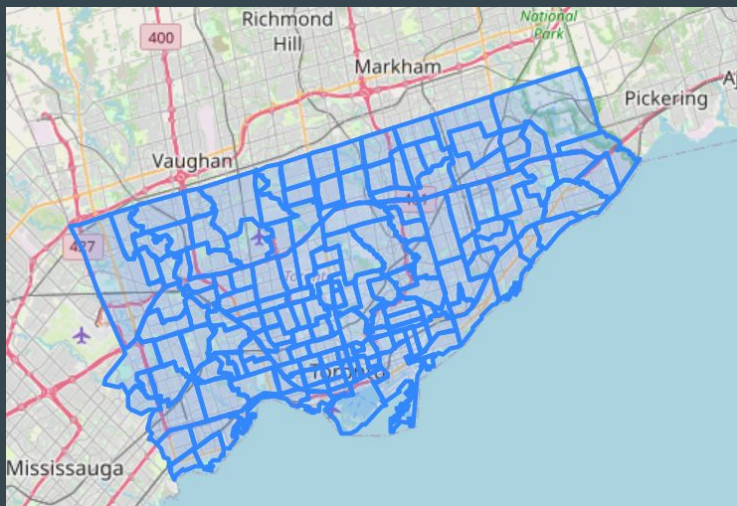
We intend to try several others and hopefully select the best model.

Current accuracy score: 17.4%

The accuracy score currently in use is **mean absolute percentage error**.

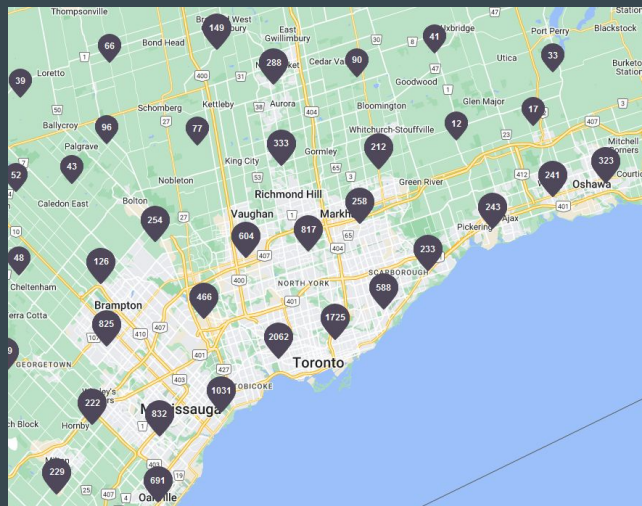
Dashboard Overview

Here is the original blueprint of our dashboard. The idea was to present the data on an interactive map, similar to these:



Source:

<https://open.toronto.ca/dataset/neighbourhoods/>



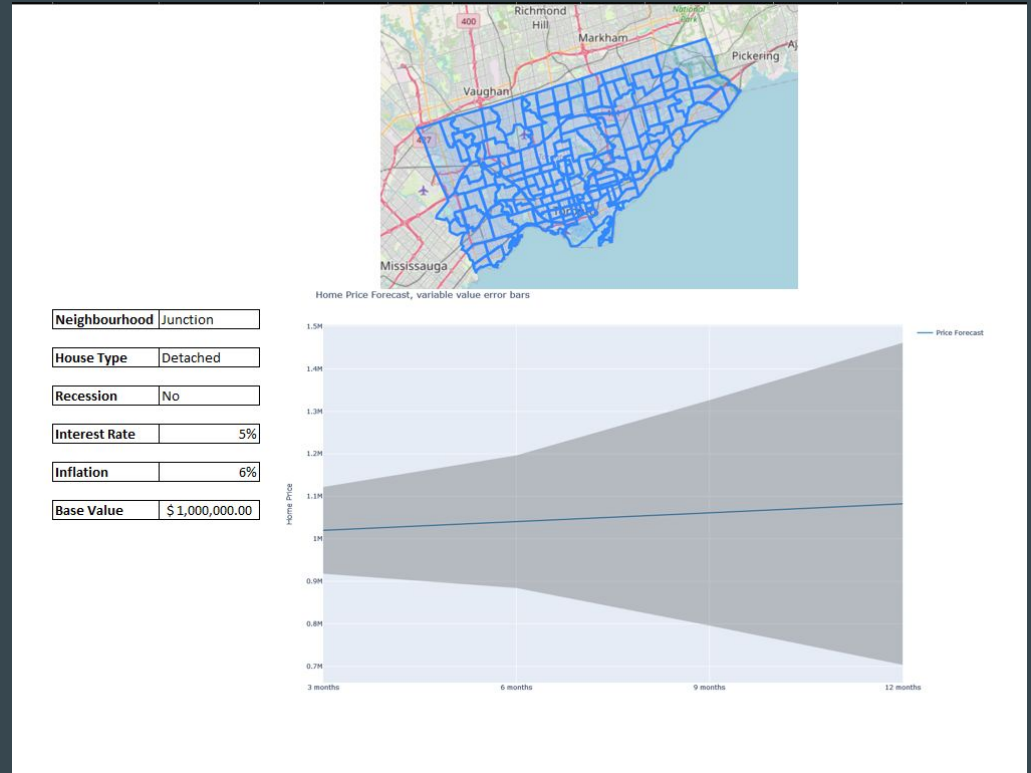
Source:

<https://www.realtor.ca/map>

The dashboard first blueprint can be seen here.

The idea was for users to be able to make quick searches

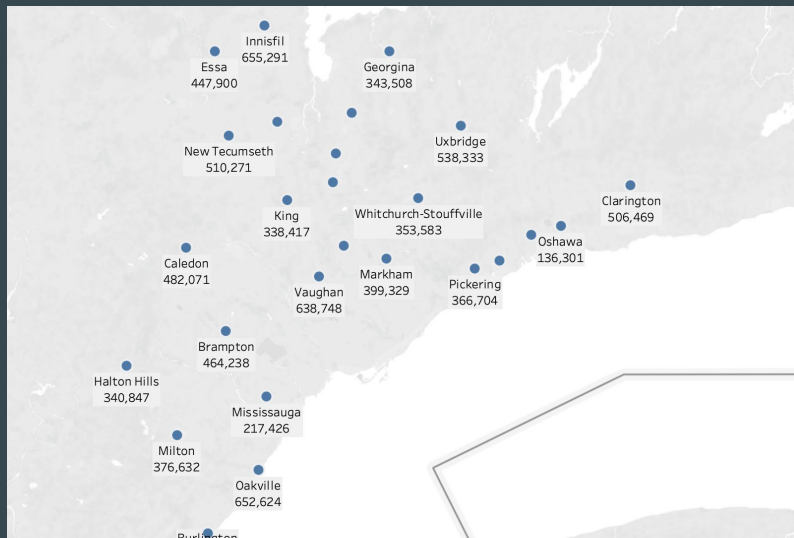
- Entering their preferred criteria
- Using the interactive map
- Accessing the prices forecast produced by our Machine Learning model



Dashboard 1

Link: <https://public.tableau.com/app/profile/wei.jin4205/viz/TorontoHouseAnalysis/Story1?publish=yes>

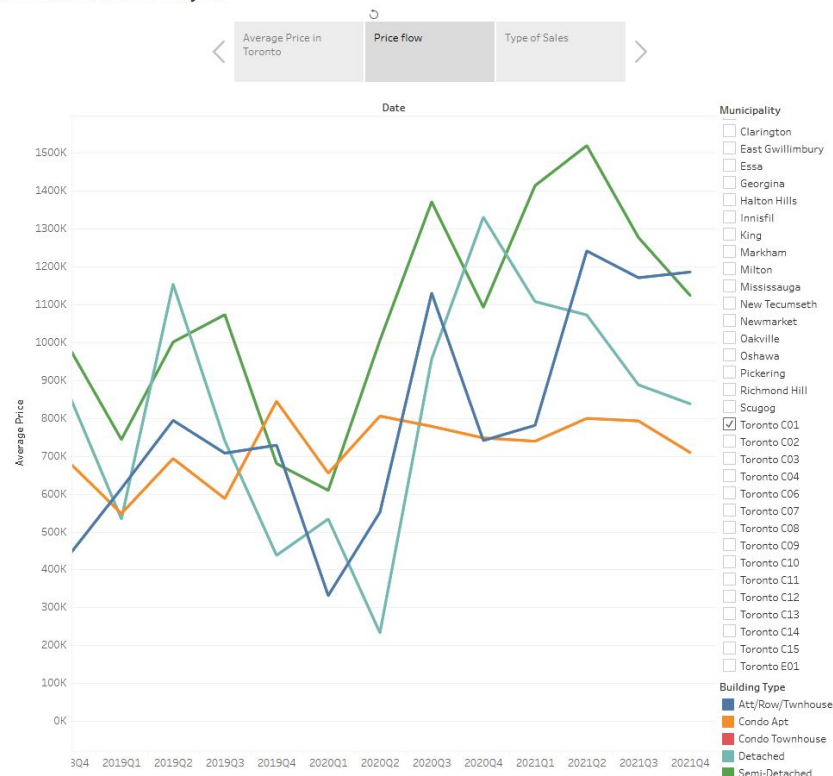
One of the visualisations is a map where users can filter through prices history according to location, time and building type.



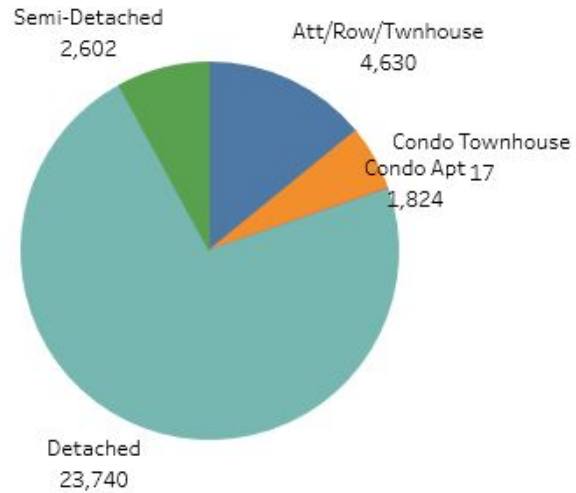
Dashboard 2

We also included a line chart of the price evolution of different building types that can then be filtered by location.

Toronto House Analysis



Dashboard 3



This pie chart displays the ratio of sales by building types.