DCU School of Computing Assignment Submission

Student Name(s):	Hannah O Connor, Catherine Mooney (Group 22)
Student Number(s):	16382283, 16416052
Programme:	BSc in Computer Applications
Project Title:	CA4010 Report - Predicting Irelands' Residential Property Prices.
Module code:	CA4010
Lecturer:	Mark Roantree

Predicting Ireland's Residential Property Prices

Table of Contents

Declaration 2
Introduction 3
Idea and Dataset description 3
- Idea & Goal 3
- Dataset Description 4
- Original Dataset 4

Data Preparation 4
- Preprocessing 4

Data Cleaning 5
- Missing Values 6
- Discrepancy Detection 7
- Noisy Data 9

Data Transformation 10
- Geocoding 10
- Public Transport Stops 13
- Electoral Districts 15
- Census Data 15

Data Analysis 16
Algorithm 16
- Description 16
- Implementation 17
- Predictions 17
- Results & Analysis 19

Experiments 20
Reflection 21
References 22

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying is a grave and serious offence in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion, or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references.

I have not copied or paraphrased an extract of any length from any source without identifying the source and using quotation marks as appropriate. Any images, audio recordings, video or other materials have likewise been originated and produced by me or are fully acknowledged and identified.

This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study. I have read and understood the referencing guidelines found at http://www.library.dcu.ie/citing&refguide08.pdf and/or recommended in the assignment guidelines.

I understand that I may be required to discuss with the module lecturer/s the contents of this submission.

I/me/my incorporates we/us/our in the case of group work, which is signed by all of us.

Signed: Hannah O'Connor, Catherine Mooney

Date: 18/12/2019

Introduction

It's no doubt that housing and rent prices within Dublin have skyrocketed in recent years and has been an ongoing discussion among various social platforms. Being young and looking for a new house in the city or even trying to rent for college or a new job has been more of a struggle than ever before.
According to a global survey of expatriate workers, Dublin has been ranked the worst city in the world right now for affordable housing. [1] We have decided to investigate the various aspects that are causing the high demand and continuous increase in prices for property.
Our dataset was originally sourced from Kaggle, however, it actually originates from the Property Price Register (PPR) site [2]. The data in the PPR website is updated every couple of days in contrast to the Kaggle version, which was last updated in April this year. Having first examined this dataset, it gave us the idea to try and investigate this, looking into factors that can influence prices for the coming years. The following report is based upon trying to predict Ireland's Residential Property Prices using the data from propertypriceregister.ie and the 2016 Census Data. Here, we will discuss our findings in detail.
For reference throughout the report, we have added a 100k subset of the original dataset and a subset of the dublin set with the work we performed for the data cleaning workshop. We have also attached both the finished kildare dataset, containing census data, and the trained Kildare dataset.

The tools & libraries used within this project:

- ★ Python3
- ★ Scikit-Learn (sklearn)
- ★ Matplotlib & plot.ly
- ★ numPy
- ★ Pandas
- ★ Geopandas
- ★ Google Fusion Table
- ★ Google Geocoding API
- ★ Python requests lib
- ★ Jupyter notebook
- ★ Catboost

Idea and Dataset description

Idea and Goal

Our initial idea was to generate a way to make a prediction of Irish property prices based on the information within the property price register and any other research we conducted from outside sources. As our work progressed, it became abundantly clear that it wasn't feasible to base our prediction off the original attributes in the dataset and that we would need to to do major planning to seek out ways to enhance the set, to ultimately make the prediction as accurate as possible.

Dataset Description

The dataset was obtained from **propertypriceregister.ie**, which collects all of the residential properties sold across the country from the years 2010 to now.
The dataset was downloaded in the form of csv file and originally had 393,544 rows. With a dataset of that size, the computations were quite slow when it came to running big tasks. So we decided to split up each county into their own separate file.

Original Dataset

Head of original dataset:

	Date of Sale (dd/mm/yyyy)	Address	Postal Code	County	Price (€)	Not Full Market Price	VAT Exclusive	Description of Property	Property Size Description
0	01/01/2010	5 Braemor Drive, Churchtown, Co.Dublin	NaN	Dublin	€343,000.00	No	No	Second-Hand Dwelling house /Apartment	NaN
1	03/01/2010	134 Ashewood Walk, Summerhill Lane, Portlaoise	NaN	Laois	€185,000.00	No	Yes	New Dwelling house /Apartment	greater than or equal to 38 sq metres and less...
2	04/01/2010	1 Meadow Avenue, Dundrum, Dublin 14	NaN	Dublin	€438,500.00	No	No	Second-Hand Dwelling house /Apartment	NaN
3	04/01/2010	1 The Haven, Mornington	NaN	Meath	€400,000.00	No	No	Second-Hand Dwelling house /Apartment	NaN
4	04/01/2010	11 Melville Heights, Kilkenny	NaN	Kilkenny	€160,000.00	No	No	Second-Hand Dwelling house /Apartment	NaN

As you can see from the above shot, there was little information given for each house, but still contained a lot of null values. We also found that the majority was basic categorical data, only containing 1 column of numerical data, the price.

Data Preparation

Preprocessing

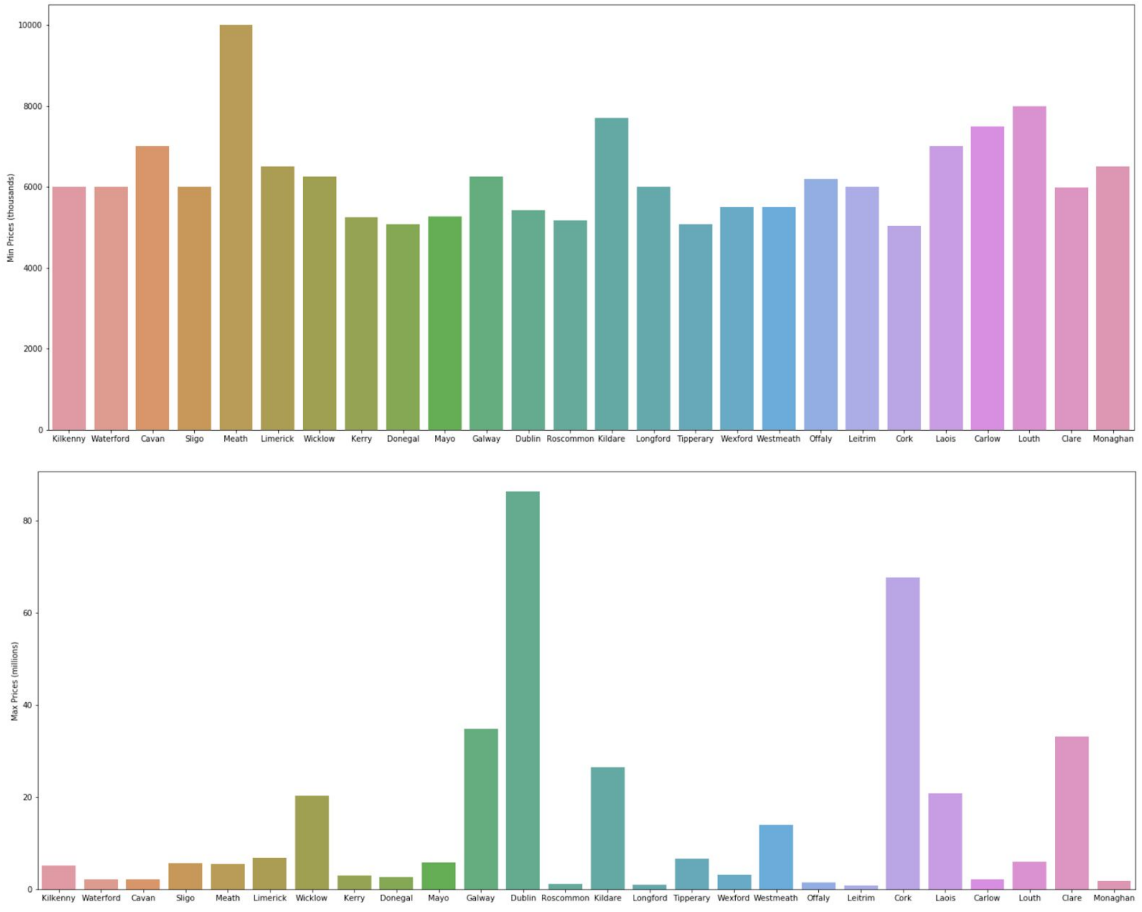
We cleaned the raw information mainly using **Pandas**. We began by examining the property price register dataset and each columns meaning. The first column which stood out to us was the 'full market price' column. On researching this, we found that houses which were not full market price, meant that they could possibly have been bought under the Affordable Houses Scheme or that only a fraction of the property was purchased. [3] Hence, deciding to remove all values which weren't full market price and the column itself, as this would have majorly affected our predictions.
We also needed to remove the rows containing Irish properties descriptions, giving the inconsities it was causing with the fadas, as shown below.

```
In [8]: df['property_description'].unique()

Out[8]: array(['Second-Hand Dwelling house /Apartment',
               'New Dwelling house /Apartment', 'Teach/Árasán Cónaithe Atháimhe',
               'Teach/Árasán Cónaithe Nua', 'Teach/?ras?n C?naithe Nua'],
              dtype=object)
```

Once this was done, we moved on to general tidying up of the dataset. For example, we converted upper case addresses and transformed column names. Also, in an attempt to gain more numerical data, we categorised the property_description columns' values to 0's for first-hand properties and 1's for second-hand.

As previously mentioned, we felt as though it would be easier to split the csv file, not just for easier computation but as there would be such a huge range in average price per county. For example, prices in the cities like Dublin and Galway, would on average, be much more expensive than the cost of a house in the smaller counties, like Carlow or Monaghan.



The first chart shown above displays the cheapest house prices per county versus the most expensive. Here, It's clear to see there are many outliers within both. Before simply just removing that data, we wanted to take a closer look. Please see section '*Outliers*' for our analysis.

Lastly, we split the **date_of_sale** column to **year_sold** and **month_sold.** Once again, creating more numerical data to work with our prediction, as we assumed the average prices would not only fluctuate by a given year, but by month also.

Data Cleaning

For our workshop we chose Data Cleaning. On examining our initial dataset, we discovered that the data is manually entered by those within the Revenue

commissioner office, meaning it had a high chance of containing human errors and missing values. Not just that but it was also stated on the site that this data is unedited by the PRSA. [4]

We were aware that there was a lot of cleaning that could be done to it in order to improve productivity and aid our prediction, so Data cleaning seemed like the best one to use. After, we felt that our dataset was in a much better state.

Below, we discuss what we believed to be the necessary steps that ensured our dataset was to be ready for analysis.

Missing Values

```
date_of_sale                0
address                     0
postal_code            305009
county                      0
price                       0
vat_exclusive               0
property_description        0
property_size          323669
```

Above shows the number of null values contained within the original set, those being in the columns **postal_code** and **property_size**. We decided to investigate the values that did exist for each to decide whether we could manually fill what's missing or drop them completely.

For **property_size**:

```
df['property_size'].unique()
```

```
array([nan,
       'greater than or equal to 38 sq metres and less than 125 sq metres',
       'greater than 125 sq metres', 'less than 38 sq metres',
       'greater than or equal to 125 sq metres'], dtype=object)
```

Here, this column had little or no use in our dataset. Not only were there too many empty values, but the information that was there was too vague to benefit us. It would have ultimately led to bias in the prediction.

For **postal_code**:

```
df['postal_code'].unique()
```

```
array([nan, 'Dublin 14', 'Dublin 2', 'Dublin 13', 'Dublin 12', 'Dublin 4',
       'Dublin 9', 'Dublin 24', 'Dublin 15', 'Dublin 22', 'Dublin 5',
       'Dublin 18', 'Dublin 6', 'Dublin 6w', 'Dublin 7', 'Dublin 16',
       'Dublin 11', 'Dublin 8', 'Dublin 3', 'Dublin 1', 'Dublin 17',
       'Dublin 20', 'Dublin 10'], dtype=object)
```

For this column, we assumed that it would have contained Eircodes, which were only introduced in 2015, hence why there could be a large number missing. Instead, it only contained Dublin areas codes.

At the time of the workshop, we had decided to choose the methods 'Fill in the missing value', for the postal_codes and 'Ignore the tuple' when it came to the property_size. For Dublin, we changed the name postal_code to then be area_code.

From that, we created a simple Python hash table which mapped the area_codes to their respective town/area. For those which didn't have an area code, we simply set the default value to be 'Co. Dublin'. Small subset:

```
dublin_codes = {
   "Dublin 1" : ["Greene St", "Bolton St", "Gardiner Place", "Summerhill", "Abbey St",
"Amiens St", "Capel St", "Dorset St","Henry St" ,"Mountjoy Sq" ,"Marlboro St", "North
Wall",
   "O'Connell St","Cathal Brugha St","Marlborough St", "Parnell Sq", "Talbot St",
"Parnell St", "Gardiner St", "Spencer Dock", "Custom House"],
   ........
 }
```

This approach did require quite some time, but we felt that it was the least bias when it came to dealing with missing values.

For other counties, it wasn't as straightforward with Dublin. We had decided to initially take the last part of the given address, that is if it's wasn't the county itself. But this approach introduced hours more workload for us for each of the other 25 counties. We ultimately decided that Dublin will contain the extra two columns 'area_code' and 'town'.

Discrepancy Detection

Using Metadata

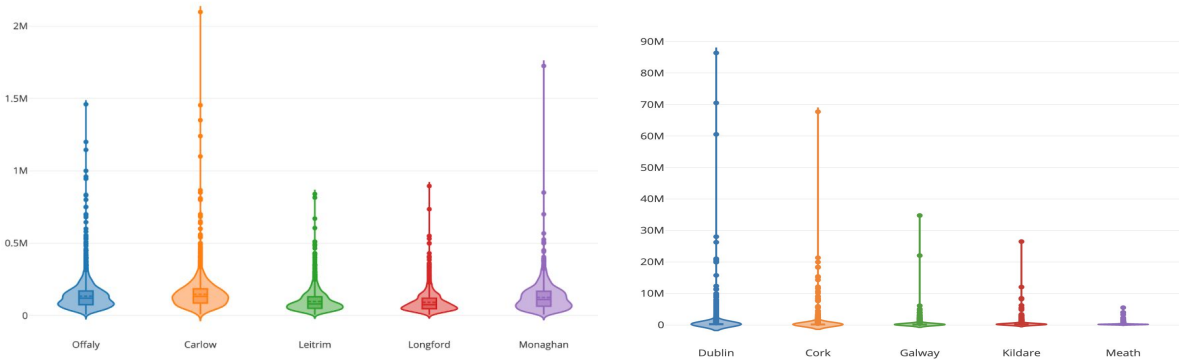
Originally, our dataset didn't contain useful metadata. With that, we researched ways to incorporate more information as the initial set didn't have as much as we'd like. Through using **Google's Geocoding API,** we could retrieve the latitude, longitude, their accuracy and the eircode of the given address. It helped to resolve incomplete and misspelled addresses like: *"3 Cois Chnoic, Tirkeenan, Momaghan"* **->** *"3 Cois Chnoic, Connolly Park, Monaghan, H18 F201"*

Result:

address	postal_code	county	price	vat_exclusive	property_description	property_size	town	area_code	eircode	longitude	latitude	accuracy
5 Braemor Drive, Churchtown, Co.Dublin	NaN	Dublin	343000.0	No	Second-Hand Dwelling house /Apartment	NaN	Churchtown	Dublin 14	D14 NX40	-6.263783	53.302391	ROOFTOP

Identifying Outliers

Through splitting the dataset we could easily identify outliers as the average varies widely across counties. We took the highest versus lowest 5 counties to discern the outliers via violin charts, as shown below.



By doing this, we examined those extreme values more closely. We wanted to take a closer look into the cases for each county.

- **Cork**: *'1 The Elysian, Eglinton Street, Cork'* - which is an entire apartment block.
- **Galway:** '*Cuirt Na Coiribe Headford Road, Cuirt Na Coiribe, Headford Road*' - Student accommodation block.
- **Kildare:** '*Castlemartin House, Kilcullen*' - a Georgian Mansion
- **Meath:** '*1-26 Blackcastle Manor, Slane Road, Navan*' - Residential development
- **Monaghan:** 'As caill Rois, Carrickmacross' - Full housing estate.

The likes of the above were dropped from the dataset as they would skew our predictions and also most not even being classified as residential property. But those weren't the only ones present. With, the best approach seems to be to trim 10% off either end for each file, removing both the extreme highs and lows.

Inconsistency data

To help with computations we converted all prices to floats and spit the date, removed Irish description, as previously discussed.

Price format

```
df['price'] = df['price'].astype(float)
```

Date format

```
for i in df.index:
  date = df.at[i,"date_of_sale"]
  month, year = date.split('/')
  if month[0] == '0':
    month = month[1]
  df.at[i,"year_sold"] = year
  df.at[i,"month_sold"] = month
```

Field Overloading

Before:

"22 Laverna Way, Castleknock, Dublin 15"

After:

22 Laverna Way, Castleknock, Dublin 15	Dublin	355000.0	No	Second-Hand Dwelling house /Apartment	NaN	Castleknock	Dublin 15	D15 KC8C	53.373652	-6.383476

With Dublin, we could discard the full address and use it's counterparts: town and area code instead.

Noisy Data

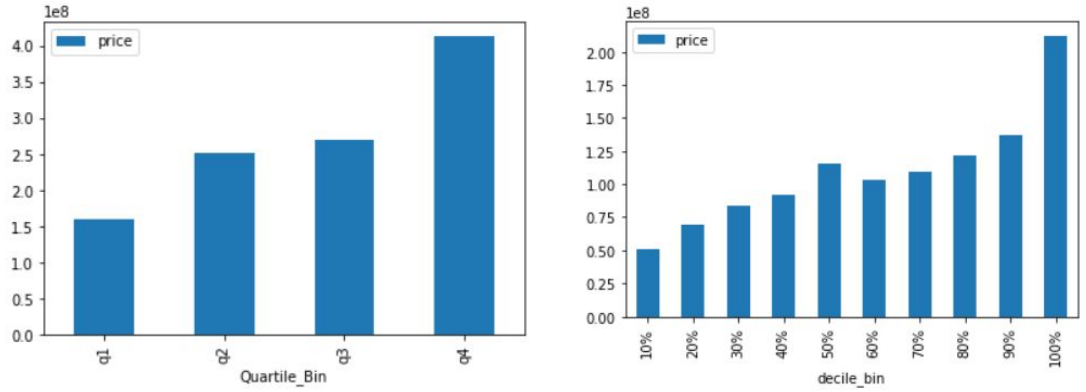
We examined the 3 possible smoothing methods. On researching the advantages and disadvantages of each, we opted for **binning**. As we split up the csv file, we applied this smoothing method to each town within Dublin, given the more data and wider price range. On deciding how to bin the prices, we chose quantile-based partitioning through the pandas **qcut method,** placing them in equal sizes. Here we created the new columns, displaying both quartile and decile bins:

	date_of_sale	address	county	price	vat_exclusive	town	area_code	quartile_bins	decile_bins
0	06/01/2010	18 Earlsfort Court, Lucan, Co. Dublin	Dublin	280000.0	No	Lucan	Co. Dublin	(275000.0, 325990.858]	(275000.0, 295000.0]
1	06/01/2010	9 Colthurst Green, Huntington Glen, Lucan	Dublin	228000.0	No	Lucan	Co. Dublin	(218375.0, 275000.0]	(200000.0, 230000.0]
2	21/01/2010	4 Caislean Riada Avenue, Lucan	Dublin	272500.0	No	Lucan	Co. Dublin	(218375.0, 275000.0]	(253000.0, 275000.0]
3	22/01/2010	13 Beech Park, Lucan	Dublin	478000.0	Yes	Lucan	Co. Dublin	(325990.858, 3432000.0]	(404500.0, 3432000.0]
4	22/01/2010	4 Larkfield Grove, Lucan	Dublin	230000.0	No	Lucan	Co. Dublin	(218375.0, 275000.0]	(200000.0, 230000.0]

The examples show the Lucan dataset. We used both quartile and decile bins as they helped in further analysis as we stored the results within the dataframe with boundary labels:

	date_of_sale	address	county	price	vat_exclusive	town	area_code	quartile_bins	decile_bins
0	06/01/2010	18 Earlsfort Court, Lucan, Co. Dublin	Dublin	280000.0	No	Lucan	Co. Dublin	q3	60%
1	06/01/2010	9 Colthurst Green, Huntington Glen, Lucan	Dublin	228000.0	No	Lucan	Co. Dublin	q2	30%
2	21/01/2010	4 Caislean Riada Avenue, Lucan	Dublin	272500.0	No	Lucan	Co. Dublin	q2	50%
3	22/01/2010	13 Beech Park, Lucan	Dublin	478000.0	Yes	Lucan	Co. Dublin	q4	100%
4	22/01/2010	4 Larkfield Grove, Lucan	Dublin	230000.0	No	Lucan	Co. Dublin	q2	30%

In the example above, we can see that the second row lies within quartile 2 and decile 50% (40-50). **Retbins=True** shows us the actual labelled bin values.



Looking at the above bar charts' shape, they're consistent. Yet it's clear to see that q4 is skewed due to the 10th decile, which is likely due to the presence of outliers. Elsewhere remains quite evenly distributed. The alternative cut method prohibited us from categorizing our data to labels, meaning it wouldn't be useful for analysis or prediction.

Data Transformation

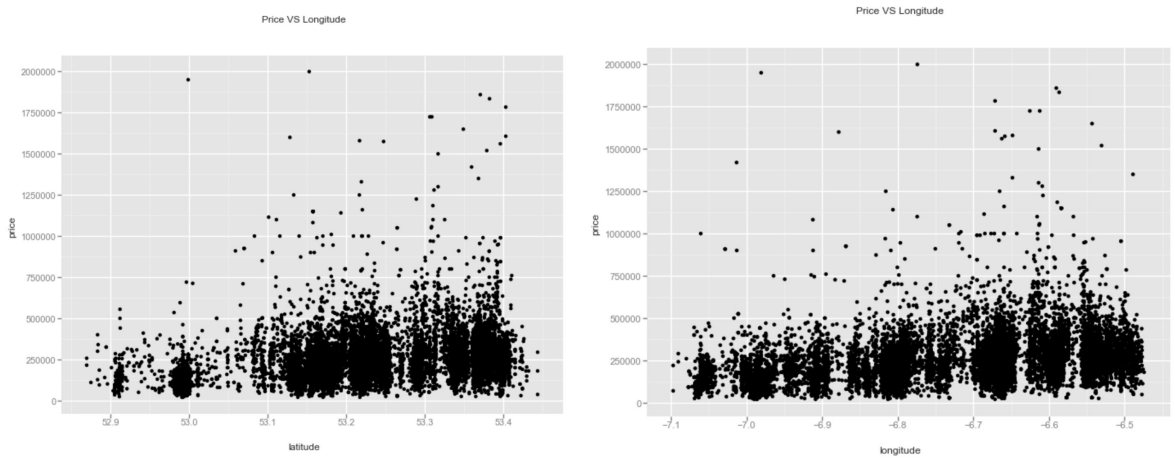
Geocoding

As said, we leveraged Googles' Geocoding API and Python's Request library to retrieve more information for the given address. In order to do this, we were required to retrieve an API key and create a url, based off the address and the key. We could then parse that URL's json response using the requests lib.

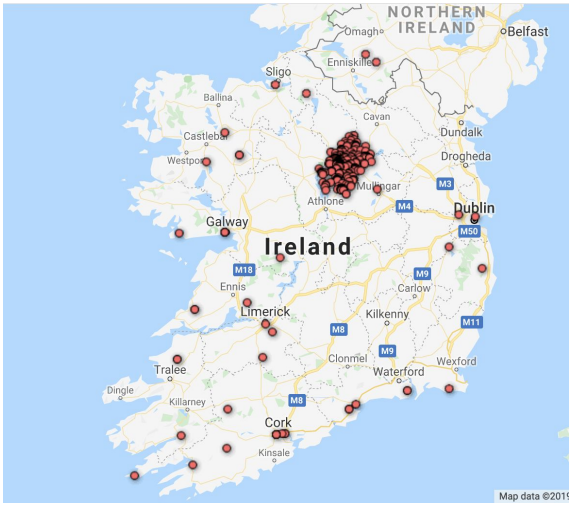
Example:

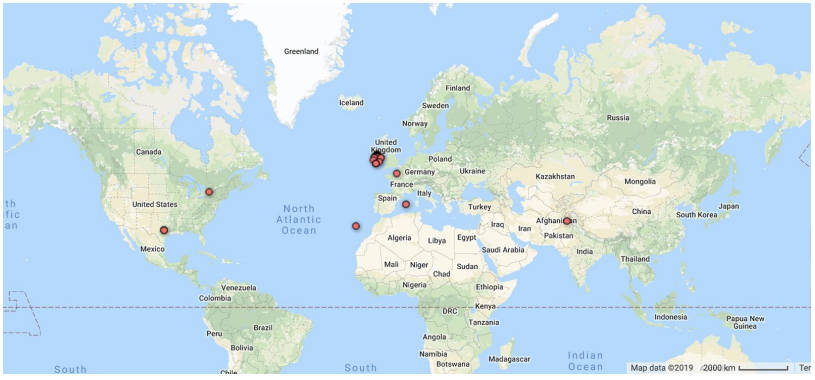
https://maps.googleapis.com/maps/api/geocode/json?address=5+Braemor+Drive,+Churchtown,+Co.Dublin,+Dublin,+Ireland&key=AIzaSyBe-vaefvOwUycBlN707CRlGhsewZq-NII"

Mapping of the lat & lng based on Kildare Prices:

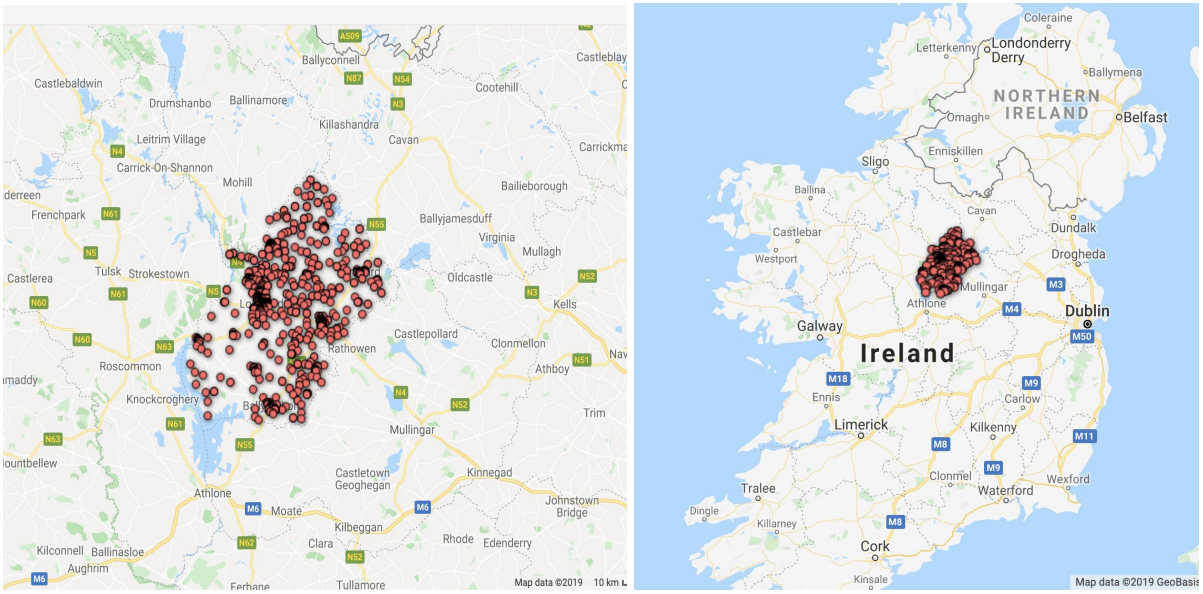


But geocoding wasn't always correct based off the given address. Through using **Google's Fusion Maps**, we could plot those coordinates to examine their correctness. As shown below, for Longford, many of the coordinates weren't within it's county. Some weren't even in Ireland.





After Cleaning:



Below shows the reported geocoded rate of error for each county.

County	Rate of Error
Dublin	22%
Kildare	10%
Meath	5%
Cork	6%
Galway	5%
Limerick	5%
Wexford	8%
Waterford	6%
Mayo	6%
Clare	8%
Cavan	7%
Westmeath	6%
Kilkenny	10%
Laois	10%

County	Rate of Error
Sligo	5%
Roscommon	7%
Leitrim	5%
Offaly	8%
Carlow	18%

Longford	12%
Monaghan	17%
Louth	7%

There were many addresses that simply translated to their county name like "*Longford, Ireland*". We dropped these values too given their inaccuracy.

On another note, even if the coordinates were based on the county, sometimes its' given address wasn't accurate. Hence, why we chose to include the column **accuracy**.

From Googles' Geocoding documentation:

- *ROOFTOP indicates that the returned result reflects a precise geocode.*
- *RANGE_INTERPOLATED indicates that the returned result reflects an approximation (usually on a road) interpolated between two precise points (such as intersections). Interpolated results are generally returned when rooftop geocodes are unavailable for a street address.*
- *GEOMETRIC_CENTER indicates that the returned result is the geometric center of a result such as a polyline (for example, a street) or polygon (region).*
- *APPROXIMATE indicates that the returned result is approximate.*

We decided to categorise those attributes into numbers to use in prediction. For **ROOFTOP** we changed this to 1, meaning best, and worst being 4, **APPROXIMATE**.

Public Transport Stops

With the idea of 'commuter towns' within Ireland, where people move to places in Dublin's bordering counties to save money on houses, we decided to include how far away (in kilometres) the nearest bus stop and train station was. Such a factor can be a big influence when buying, whether that being due to having no car or working in the city.

The public transport stop data comprised of two separate csv files, one including: Bus, ferries, Luas points and the other train stations [5]. Both included stops within Northern Ireland which we could easily drop once running both files against the borders of the electoral districts(ED). If any points did not intersect any of the ED borders, they were dropped. Here, we could also categorise points into their county, saving future computational time.

Public Transport Trains head:

	StopAreaCode	Name_lang_en	AdministrativeAreaRef	StopAreaType	GridType	Easting	Northing	Longitude	Latitude	ParentStopAreaRef	county
0	821G000014	Carlow Train Station	821	GPBS	ITM	672559.0	677188.0	-6.922991	52.840740	NaN	Carlow County
1	821GIR0001	Bagnelstown Train Station	821	GRLS	ITM	670794.0	661404.0	-6.952594	52.699135	NaN	Carlow County
2	821GA00006	Bagnelstown Train Station	821	GPBS	ITM	670794.0	661408.0	-6.952593	52.699171	821GIR0001	Carlow County
3	821GA00007	Bagnelstown Train Station	821	GPBS	ITM	670792.0	661402.0	-6.952624	52.699117	821GIR0001	Carlow County
4	821GA00008	Bagnelstown Train Station	821	GPBS	ITM	670768.0	661385.0	-6.952983	52.698967	821GIR0001	Carlow County

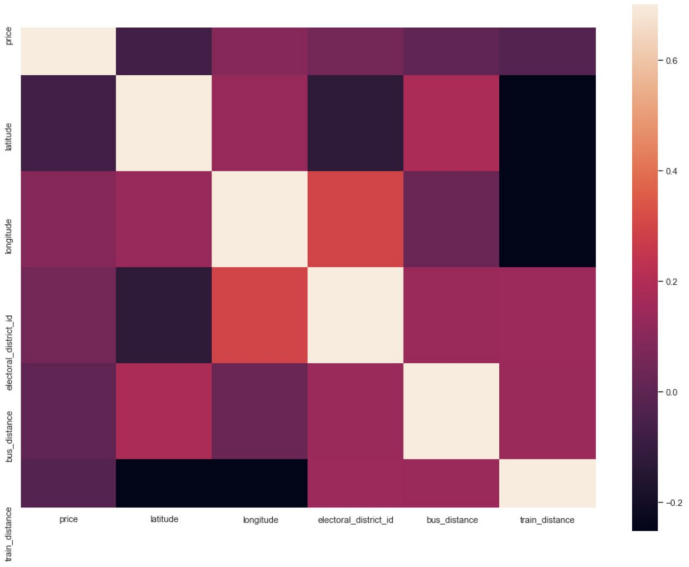
For each house, the algorithm calculates distances to all the stops in the dataframe, based off it's county. Each is stored to a list, where the smallest is kept and stored.

```
county = df.at[0,'county']
county = county + ' County'
for i in df.index:
  distances = []
  if pd.isnull(df.at[i, 'bus_distance']):
      for j in transport.index:
        if transport.at[j,'county'] == county:
          lat1 = radians(df.at[i,'latitude'])
          lon1 = radians(df.at[i,'longitude'])
          lat2 = radians(transport.at[j,'Latitude'])
          lon2 = radians(transport.at[j,'Longitude'])
          dlon = lon2 - lon1
          dlat = lat2 - lat1

          a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
          c = 2 * atan2(sqrt(a), sqrt(1 - a))

          distance = R * c
          if not pd.isnull(distance):
            distances.append(distance)
  if len(distances) != 0:
    df.at[i, 'bus_distance'] = min(distances)
```

We then did an analysis of the current variables, against the added public transport distance data, through examining their relationship and seeing the enhancements in our dataset, through the below statistical correlation matrix:

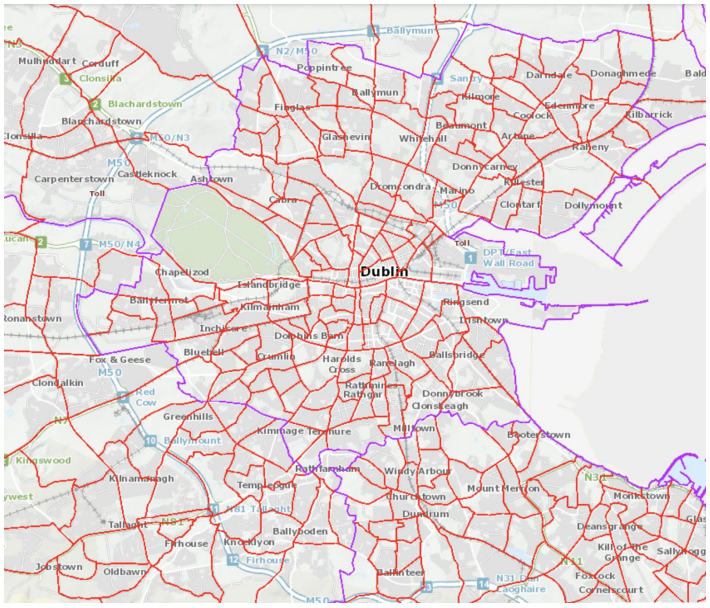


Here we can see how the different variables correlate with each other, and how closely. We used this as a way to summarize the data before analysing it further.

Electoral Districts

Upon researching how else we could transform our dataset, we came across *Electoral Districts*(EDs). In Ireland, the definition of an ED is the 'smallest legally defined administrative areas', with a total of 3,440 across the country. We were able to openly download the polygon shape file provided on the CSO Census website. [6] That way, we could attempt to retrieve census statistics based off each one.

Dublin City Example:



Source: census.cso.ie/sapmap

To get each property's ED we match up its' coordinated on the basis of if it fell within any border or intersected of an ED. Once we determined this, we could retrieve and store the information for an ED information to the corresponding row. [6]

Algorithm[7]:

```
import sys
i = 0
ED = gpd.read_file("../Downloads/electoral-divisions-gps-projection/electoral_divisions_gps.shp")
ED = ED.to_crs({'init': 'epsg:4326'})
for i in df.index:
      points = Point((df.at[i,'longitude'], df.at[i,'latitude'])) # Longitude & Latitude
      found = ED[ED.geometry.intersects(points)]
      if not found.empty:
        region = ED[ED.geometry.intersects(points)].NUTS3NAME.values[0]
        electoral_district_id = ED[ED.geometry.intersects(points)].CSOED.values[0]
        electoral_district = ED[ED.geometry.intersects(points)].EDNAME.values[0]
        df.at[i,'region'] = region
        df.at[i,'electoral_district_id'] = electoral_district_id
        df.at[i,'electoral_district'] = electoral_district
```

Census Data

Using our newly gathered ED data, we could then obtain the Census small area population statistics in csv form and query against our own rows through each ED id.

[8] In order to assess what each column meant and what would be relevant for prediction. For this we could refer to its glossary table, from the site also. Once done, we made copies of our counties files and merged both of csv files based on the ED id.

Result (Kildare.csv example):

```
['date_of_sale' 'address' 'county' 'price' 'vat_exclusive'
 'property_description' 'property_size' 'place_id' 'geocoding_url'
 'latitude' 'longitude' 'formatted_address' 'eircode' 'accuracy' 'region'
 'electoral_district_id' 'electoral_district' 'bus_distance'
 'train_distance' 'average_children' 'GEOGID' 'age_0' 'age_1' 'age_2'
 'age_3' 'age_4' 'age_5' 'age_6' 'age_7' 'age_8' 'age_9' 'age_10' 'age_11'
 'age_12' 'age_13' 'age_14' 'age_15' 'age_16' 'age_17' 'age_18' 'age_19'
 'age_20_to_24' 'age_25_to_29' 'age_30_to_34' 'age_35_to_39'
 'age_40_to_44' 'age_45_to_49' 'age_50_to_54' 'age_55_to_59'
 'age_60_to_64' 'age_65_to_69' 'age_70_to_74' 'age_75_to_79'
 'age_80_to_84' 'age_>85' 'single' 'married' 'separted' 'divorced'
 'widowed' '2_persons_family' '3_persons_family' '4_persons_family'
 '5_persons_family' '>6_persons_family' 'population' 'total_children'
 'no_children_family' '1_children_family' '2_children_family'
 '3_children_family' '4_children_family' '1_person_households'
 'cohabiting_households' 'Cohabiting_children_households'
 'married_households' 'total_apartments_houses' 'houses' 'apartments'
 '1919_to_1945_built' '1946_to_1960_built' '1961_to_1970_built'
 '1971_to_1980_built' '1981_to_1990_built' '1991_to_2000_built' '1_rooms'
 '2_rooms' '3_rooms' '4_rooms' '5_rooms' '6_rooms' '7_rooms' '>8_rooms'
 'no_central_heating' 'oil_heating' 'natural_gas_heating'
 'electricity_heating' 'coal_heating' 'peat_heating' 'liquid_heating'
 'wood_heating' 'employed' 'unemployed' 'student' 'carer' 'retired'
 'total_opp' 'no_eduction' 'primary_eduction' 'junior_cert_eduction'
 'leaving_cert_eduction' 'vocational_eduction' 'apprenticeship_eduction'
 'higher_cert_eduction' 'ordinary_bach_eduction' 'honours_bach_eduction'
 'postgrad_eduction' 'phd_eduction' '<15m_commuter' '15m_commuter'
 '30m_commuter' '45m_commuter' '1hr_commuter' '>2hr_commuter' 'no_cars'
 '1_cars' '2_cars' '3_cars' '>4_cars']
No. variables: 129
```

Data Analysis

Taking one more look further into our dataset before we began predictions. After the completion of the data cleaning and retrieval steps, we were still left will some null values that we ignored in our *Missing Values* sections. Below shows the missing percentage those null values, which was decided to simply drop.

	missing_values	percentage
property_size	13667	0.783748
eircode	8130	0.466223

Algorithm

Description

For our prediction, we decided to opt for Regression. For our prediction, we came across several potential models for implementing it. We tried **sklearn linear regression**[9], **catboost's CatBoostRegressor**[10] and **Random Forest**.However after running various tests, we found that catboost was the stronger model, given the fact it uses a gradient boosting machine learning algorithm and also able to handle categorical data. So we selected that to conduct our predictions.

Implementation

Regression Model

```
import math
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error
from catboost import Pool, CatBoostRegressor

trained = trained_data.drop('price', axis=1)
trained_price = np.log(trained_data['price'])

X_trained, X_test, y_trained, y_test = train_test_split(trained, trained_price,test_size=0.33, random_state=42,
                                                        shuffle=False)
#X_test stays the same
X_test, X_values, y_test, y_values = train_test_split(X_test,y_test, test_size=0.33, random_state=42, shuffle=False)

trained_pool = Pool(X_trained.values, y_trained.values)
test_pool = Pool(X_test.values)
values_pool = Pool(X_values.values, y_values.values)

cbr = CatBoostRegressor(iterations=99,
                        depth=10,
                        learning_rate=0.3,
                        loss_function='RMSE',
                        random_seed=42,
                        eval_metric='RMSE',
                        use_best_model=True)
cbr.fit(trained_pool, eval_set=values_pool,plot=True, early_stopping_rounds=80)
predictions = cbr.predict(test_pool)
# calculate MAE, MSE, RMSE
print('RMSE: {}'.format(math.sqrt(mean_squared_error(y_test.values, predictions))))
print('MAE: {}'.format(math.sqrt(mean_absolute_error(y_test.values, predictions))))
print('MSE: {}'.format(mean_squared_error(y_test.values, predictions)))
```

Above shows our Regression model for predicting our properties prices using catboost.
Since our goal is to predict the price, we need to drop it from the trained dataset. Then we use sklearn's train_test_split feature to split the given dataset into train set of approximately 77 % and test of 33%.
Next, we need to fit and train the model using the trained pool data. The parameters selected represent what returned the best score. From playing around and assessing various combinations, we found that the best combination of parameters were: **iterations=99**, **depth=10** with **learning_rate=0.3**. The primary metric used to assess the prediction results, as well as the **lost_function** and overfitting detection (**eval_results**), was **RSME** - which gives an indication of how close the prediction was to the actual price values. We hardcoded the random seed to be 42, as for training purposes, we wanted to ensure that our results wouldn't differ each time.

We originally had the option of applying categorical data to this model, but given the results, we found that through dropping the categorical data and categorising what we could into integers and floats, it yield better results. By not adding the option cat_features, the default values was None and so, it only considered numerical data

Lastly, we use **cbr.precict()** to predict the results from our test file using the trained model, which created an np.array '*predictions*' of corresponding price predictions. Allowing us to then evaluate the **RMSE**, **MAE** and **MSE** scores based on that given np.array.

Given the fact that all countries outside of Dublin contained the same amount of attributes, we didn't see any issues when it came to overfitting. But because of that, Dublin needed to have it's own separate model

Predictions

Catboost

Below we discuss a sample of our predictions done for Dublin City versus the counties. For the purpose of this report, we chose to discuss just county Kildare, rather than all 25 others.

Test 1 - County Predictions

The below test results show the predictions done for county Kildare.

Without Census Data

```
RMSE: 0.1187480690510717
MAE: 0.3073984456046128
MSE: 0.014101103903358093
```

predicted_price	actual_price
407918.383716	339206.76
316255.250656	310000.00
349404.377359	300000.00
276765.134380	345000.00
295778.015103	202000.00
285524.615413	334801.00
276765.134380	341547.35
238689.288831	210000.00
437671.562606	290000.00
335452.091417	249000.00

With Census Data

```
RMSE: 0.25240166681000914
MAE: 0.45627447202716015
MSE: 0.06370660140847087
```

actual_price	predicted_price
280000.0	407918.383716
260000.0	316255.250656
340000.0	349404.377359
312000.0	276765.134380
264000.0	295778.015103
220000.0	285524.615413
403000.0	276765.134380
281000.0	238689.288831
417500.0	437671.562606
331000.0	335452.091417

Above shows the best cases for both, in terms of the lowest RMSE and highest accuracy. Here the trained dataset used, contained the housing data from 2018 and onwards.
Considering the fact that, in kildare especially, there's a high number of commuter towns across its Dublin border, which may have skewed the overall prediction slightly for this test case.

Test 2 - City Prediction

Trained Dublin dataset:

	price	property_description	town	area_code	latitude	longitude	accuracy	electoral_district_id	bus_distance	train_distance	year_sold	month_sold
0	343000.0	1	0	14	53.302391	-6.263783	1	5025	0.124643	3.978828	2010	1
1	438500.0	1	1	14	53.287187	-6.254580	1	5005	0.167311	4.678816	2010	1
2	425000.0	1	2	6	53.325279	-6.251186	1	2134	0.093346	1.746083	2010	1
3	430000.0	1	3	3	53.371245	-6.238115	1	2046	0.040475	1.163026	2010	1
4	355000.0	1	4	15	53.373652	-6.383476	1	4016	0.246069	0.651892	2010	1
...	...	...	...	...	...	...	...	...	...	...	...	...
04905	342000.0	1	40	14	53.321682	-6.326957	3	2159	0.206245	3.369375	2019	10
04906	281940.0	0	119	lin	53.520218	-6.163717	3	4028	0.000002	1.217867	2019	10
04907	290000.0	1	53	7	53.364636	-6.284503	1	2031	0.083428	1.282645	2019	10
04908	315000.0	1	58	12	53.322032	-6.319010	2	2160	0.012559	3.052392	2019	10
04909	565000.0	1	159	3	53.363432	-6.235793	1	2045	0.029679	0.562937	2019	10

- We found the performance to be better and we saw better scores once we dropped all the categorical data and transformed what we could to be numerical.

Trained set with 2015 and onwards RMSE score:

```
RMSE: 0.21418560815120777
MAE: 0.3995756591318965
MSE: 0.04587547473910272
```

Trained set with 2015 and onwards prediction results:

actual_price	predicted_price
341000.00	421817.404620
436123.35	353858.614763
377945.27	321158.527425
265000.00	273495.462606
240000.00	279070.030827
671000.00	388669.778090
500000.00	556128.563770
520000.00	423169.869912
259911.89	364822.680162
325823.00	394538.935600

We saw the best results for Dublin after having removed the housing data prior to 2017 within its training set. Yet, in the hopes of lowering the RMSE score even more, we trained houses sold only in 2018 and onwards, but once doing that we started to decrease in accuracy, retrieving a RMSE score of 0.243, as the trained set became much smaller. This could be a reflection of high housing prices spikes in 2018 and the drop that was gradually seen in 2019.

Trained set with 2018 onwards prediction results:

```
RMSE: 0.2430565317354116
MAE: 0.4272637413483723
MSE: 0.059076477619247156
```

Results & Analysis

With census data/without census data

As you can see from the above RMSE results for each of our predictions, we were quite taken aback by the fact that through adding the census data it actually worsen our prediction.

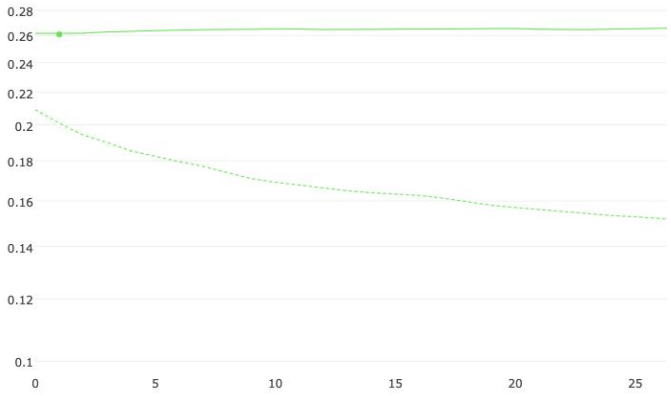
The first predictions started with the census data and we retrieved a 0.50 RMSE score. After that result, we properly assessed which columns were needed and what was redundant. Following that we were about to bring it down to roughly 0.25 RMSE, which was still more than half of the score without it. We criticize ourselves for believing that adding the census data was a good move at all.

Dublin Price Analysis

Prices grew from 2013 to 2018, however, by September this year, 2019, for the first time since 2012, the property prices in Dublin have decreased. Dublin saw a decrease of 1.3%, while the rest of Ireland saw a 3.6% increase. Dublin property prices seem to vary

compared to the rest of the country, which is why we decided to split the database up into individual counties. Also, Dublin property prices vary widely depending on the electoral district, for example, Dún Laoghaire-Rathdown 6.8%, while Fingal grew by 1.5%. [11]

Taking the data from 2019 alone:



As you can see from the above graph, the training and the test file isn't as in sync - which could reflect the major fluctuation in prices that Dublin has seen over the previous years

On another note, the fact that this dataset contains both houses and apartments, with no way to distinguish between the two, can ultimately affect our predictions majorly. We believe that if this set could have contained either houses or apartments solely, our predictions would have been a better reflection of the actual prices. Especially in Dublin where there is a high concentration of apartments.

County Price Analysis
Based on the above analysis we received RMSE for kildare at **0.118**, and we were gladly surprised. We had initially guessed that results wouldn't be as accurate given the lack of valuable data that we had in contrast to other available datasets online.

Experiments

Our results with the census data weren't actually good as we initially had hoped for. By adding that data to our original dataset, we could enrich it more and ultimately end with better predictions. We had figured that because with this data only being relevant to each electoral district(ED) it meant there was a lot of repeated data for each house in each town in an ED. Despite what we originally thought - given that the house is located in that region, the factors of that region would make a difference to a house cost.

Instead, we thought a better route to take with the ED's was to predict the prices with the best model that we found to be the most effective and get the individual predictions and compute the average cost for a house in an ED. Here are our results:

	predicted_mean	predicted_median	ED	actual_mean	actual_median
0	163060.367239	152799.894218	6001	172019.788546	152000.0
1	264002.200261	265777.208436	6040	270174.000000	270334.0
2	200310.636234	200041.953539	6066	204419.441860	200000.0
3	253858.920924	256473.867098	6034	260328.311945	260000.0
4	251979.143536	242136.185698	6037	262854.863636	260000.0
...	...	...	...	...	...
81	251692.989184	251692.989184	6089	259392.857143	290000.0
82	225010.483089	225010.483089	3017	350000.000000	350000.0
83	306015.273325	306015.273325	6076	280583.333333	289000.0
84	184911.014424	184911.014424	6045	214575.000000	167500.0
85	203972.569173	203972.569173	6052	183500.000000	170000.0

Mean

```
RMSE: 0.095175126845855556
MAE: 0.262562394964266603
MSE: 0.009058304770124696
```

Median

```
RMSE: 0.128155059265677777
MAE: 0.29846552285079688
MSE: 0.016423719215389383
```

Reflection

After careful reflection of the processing we performed on our dataset, we have come to the realisation that datasets with manually entered values, can contain a lot of human errors and null values for particular columns. We discovered multiple invalid values entered, like addresses in Dublin files outside of the county:

- "*"30 Cnoc Tiarnach, Grange End, Dunshaughlin, Dublin*"
- "*11 Wingfield, Corke Abbey, Bray", Dublin 18, Dublin*"

Given these concerns about the original dataset, we realise it may not have been suitable for machine learning, due to a lack of numerical attributes. Hence why we had decided to change our original hypothesis to predict the price of houses within an electoral district based on the Irish census data.

In conclusion, our original dataset was most definitely not suitable for machine learning. Given the chance to start over, we would have chosen a better and pre trained dataset, which would have ultimately allowed us to place more emphasis on our prediction models. A lot of our engineering time went towards both cleaning and attempt to enhance the original dataset as we best could.

References

[1] "Dublin worst city in the world for expats to find housing - survey", independent.ie, 18 December 2019, Unknown
https://www.independent.ie/business/irish/dublin-worst-city-in-the-world-for-expats-to-find-housing-survey-38749929.html

[2] Residential Property Price Register https://www.propertypriceregister.ie/

[3] "Prices Given Not Full Market Price", PRSA
https://www.propertypriceregister.ie/Website/npsra/pprweb.nsf/page/ppr-price-en

[4] "Errors", PRSA
https://propertypriceregister.ie/Website/NPSRA/pprweb.nsf/page/ppr-home-en

[5] "Nation Public Transport Nodes Access Nodes (NaPTAN)", National Transport Authority, 10/04/2017
https://data.gov.ie/dataset/2017-national-public-transport-access-nodes-naptan

[6] "Electoral Division Boundaries", Central Statistics Office Ireland
http://census.cso.ie/censusasp/saps/boundaries/eds_bound.htm

[7]https://gis.stackexchange.com/questions/64513/checking-if-lng-and-lat-fall-inside-polygons-from-esri-shapefile

[8] "Census 2016 Small Area Population Statistics", Central Statistics Office Ireland
https://www.cso.ie/en/census/census2016reports/census2016smallareapopulationstatistics/

[9] "sklearn.linear_model.LinearRegression", scikit learn, 2007 - 2019, scikit-learn developers
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[10] "CatBoostRegressor", Catboost, Unknown
https://catboost.ai/docs/concepts/python-reference_catboostregressor.html

[11] "Property Prices see slowest growth in six years - CSO", 14 Nov 2019, RTE
https://www.rte.ie/news/business/2019/1114/1090843-cso-residential-property-prices/