**FINAL PROJECT**

**Step 2:**

**DATA PREPARATION**

So Arbitrage RE IKE was happy to provide a snap shot of the Greek residential real estate market and send me an excel file containing 136,289 listings with 37 different fields for each listing as of 2/2/2019, a bit dated but did not make a difference for building testing a deep learning algorithm. Many field names and variables were in Greek characters so a great deal of cleaning was required.



**Cleaning and Preparing the Data – cleandata.py**

I used the excel pivot function to get some sense of the quality of the data fields such as special characters, blanks as well as a general sense of what the data fields and values looked like. Following this I saved the excel file as a comma delimited CSV file for cleandata.py to clean up and covert to calculatable values.

The following cleaning tasks and checks were performed:

* *Number of listings: 136289 Database Shape: (136289, 36)*
* *Deleting 326 listings with no address*
* *Deleting 13 listings with no geolocations*
* *Number of listings: 135950 Database Shape: (135950, 36)*
* *listings at unkwown floor: 1 from 135950*
* *Ignoring 53 listings with floor > 9*
* *Excluded 1092 listings Not House or apartment*
* *Found 75883 apartment and 58922 houses*
* *Found 73690 listings with LivingRooms number > 0 and 0 with 0 vs 61115 undefined*
* *Found 10903 listings with Kitchens number > 0 and 0 with 0 vs 123902 undefined*
* *Found 55806 listings with WC number > 0 and 0 with 0 vs 78999 undefined*
* *Found 119635 listings with numBathrooms number > 0 and 0 with 0 vs 15170 undefined*
* *Found 131609 listings with numBedrooms number > 0 and 0 with 0 vs 3196 undefined*
* *Found 73492 listings with BuildingZone number > 0 and 0 with 0 vs 61313 undefined*
* *Found 28374 listings with DistanceFromSea number > 0 and 0 with 0 vs 106431 undefined*
* *Found 7091 listings under construction*
* *Found 28599 listings with NewlyBuilt attribute checked*
* *Found 72690 listings with Storage attribute checked*
* *Found 71143 listings with Views attribute checked*
* *Found 12965 listings with RoofTop attribute checked*
* *Found 12480 listings with SwimmingPool attribute checked*
* *Found 58683 listings with RoadFront attribute checked*
* *Found 27358 listings with Corner attribute checked*
* *Found 23119 listings with Renovated attribute checked*
* *Found 19519 listings with YearRenovated attribute checked*
* *Found 7505 listings with NeedsRenovation attribute checked*
* *Found 1292 listings with ListedBuilding attribute checked*
* *Found 2787 listings with Neoclassico attribute checked*
* *Found 11497 listings with ProfessionalUse attribute checked*
* *Found 36003 listings with Luxury attribute checked*
* *Found 9187 listings with StudentAccomodation attribute checked*
* *Found 14595 listings with SummerHouse attribute checked*
* *Found 3452 listings with Unfinished attribute checked*
* *Found 6364 listings with year renovated < 2010 of 19519 shown as renovated. Switched designation to not Renovated*
* *Found 3452 listings under construction and ensured they are designated as needing renovation*
* *Assigned rating 3 to 118944 summer houses that are <500 meters and 2 to 8901 summer houses <2000 meters from sea and 1 to the 1794 rest.*
* *Found 70390 listings with parking*

Also

* *Converted all blanks to 0’s or -1’s depending on the variable*
* *Converted all strings to numbers where appropriate*
* *Exclude listing that had a floor of 9 or more as an erroneous entry*
* *Create category field for 0: apartment 1: house*
* *if number of bathrooms, living rooms, kitchens, WC, BuildingZone is blank put -1 for Unknown*
* *Ensure pricePerSqm = price / area*
* *Year of construction set to 0 if unknown*
* *Year of construction set to -1 if underconstruction*
* *Changed parking designations from 'Οχι' (No) and 'Ναι' (yes) to 0 and 1. Blanks should treated as 0*
* *Merged the geolocation dataframe with the data datafram using the 'address field as an index using the unique address field*
* *finally the dataframe created to hold all this information was merged with the dataframe that with the geolocations for each neighbourhood that were uploaded from the NeighbourhoodTown-geolocs.txt file.*

**Geo Locations – FindGeoLoc.py**

I was hoping to have geolocations but instead I was provided with 3,786 neighbourhood/town descriptions for most of the listings. Only 78,511 of the listings had postal codes. So, I decided as a first step to extra the geolocation addresses for these 3,786 neighbourhood/town descriptions using **Google’s json API** [**https://developers.google.com/custom-search/v1/overview**](https://developers.google.com/custom-search/v1/overview)

The json produced by json = requests.get(url).json() has the following structure:

{

'**results**': [

{ 'address\_components': [

{'long\_name': '1', 'short\_name': '1', 'types': ['street\_number']},

{'long\_name': '40 Martiron', 'short\_name': '40 Martiron', 'types': ['route']},

{'long\_name': 'Larisa', 'short\_name': 'Larisa', 'types': ['locality', 'political']},

{'long\_name': 'Larisa', 'short\_name': 'Larisa', 'types': ['administrative\_area\_level\_3', 'political']},

{'long\_name': 'Greece', 'short\_name': 'GR', 'types': ['country', 'political']},

{'long\_name': '412 21', 'short\_name': '412 21', 'types': ['postal\_code']}

],

'formatted\_address': '40 Martiron 1, Larisa 412 21**,** Greece',

'geometry': {'location': {'lat': **39.6443076**, 'lng': **22.4261378**},

'location\_type': 'ROOFTOP',

'viewport': { 'northeast': {'lat': 39.64565658029149, 'lng': 22.4274867802915},

'southwest': {'lat': 39.6429586197085, 'lng': 22.4247888197085}

}

},

'partial\_match': True,

'place\_id': '**ChIJ48KYAoOIWBMRLbS2YKm7ySk**',

'plus\_code': {'compound\_code': 'JCVG+PF Larissa, Greece', 'global\_code': '8GF4JCVG+PF'},

'types': ['street\_address']

},

{ 'address\_components': [{'long\_name': '1', 'short\_name': '1', 'types': ['street\_number']},

{'long\_name': 'Areos', 'short\_name': 'Areos', 'types': ['route']},

{'long\_name': 'Larisa', 'short\_name': 'Larisa', 'types': ['locality', 'political']},

{'long\_name': 'Larisa', 'short\_name': 'Larisa', 'types': ['administrative\_area\_level\_3', 'political']},

{'long\_name': 'Greece', 'short\_name': 'GR', 'types': ['country', 'political']},

{'long\_name': '412 21', 'short\_name': '412 21', 'types': ['postal\_code']}

],

'formatted\_address': 'Areos 1, Larisa 412 21, Greece',

'geometry': {'location': {'lat': 39.6427199, 'lng': 22.4275298},

'location\_type': 'RANGE\_INTERPOLATED',

'viewport': {'northeast': {'lat': 39.64406888029149, 'lng': 22.4288787802915},

'southwest': {'lat': 39.6413709197085, 'lng': 22.4261808197085}

}

},

'partial\_match': True,

'place\_id': 'Eh5BcmVvcyAxLCBMYXJpc2EgNDEyIDIxLCBHcmVlY2UiGhIYChQKEgnPm3S4gohYExF5pD5WSxRLEBAB',

'types': ['street\_address']

}

],

'**status**': 'OK'

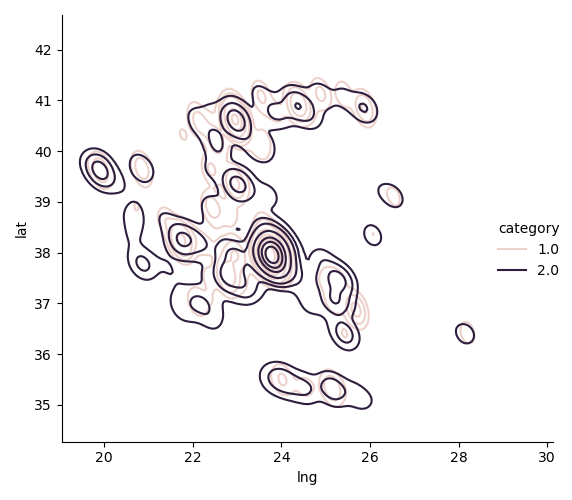
}

We notice that it has returned two addresses. We will choose the ROOFTOP one which comes first in everycase. So after we normalize the json into a list of [1 rows x 14 columns], we put it into a dataframe structure called row.

We access the latitude by row['geometry.location.lat'] and longitude at row['geometry.location.lng']. Then we perform a series of checks to see if the geolocation data is there and not a blank set the lat,lng to 0. Stored the address, latitude, longitude, and place\_id in a new csv file to upload by the datacleaning code.

When we plot the data it should be no surprise we get a map that looks like Greece, lots of houses on the coast.

Chart, scatter chart

Description automatically generated

**Cleaned Data:**



**Review of the statistical significance of each field, outliers, and model importance – calcFieldStats.py**

The goal of this statistical analysis is to help us understand the relationship between housing variables and how these variables are used to predict house price. We use linear regression to understand if already any of these variables have significant linear correlation to predicting house price.

Of the 39 variable shown below we highlight the ones we will test for usefulness and correlation to housing prices.

Also as you can see from the 136,826 listings only 134,879 left after cleaning the data.

<class 'pandas.core.frame.DataFrame'>

**RangeIndex: 134879 entries, 0 to 134878**

**Data columns (total 40 columns):**

**# Column Non-Null Count Dtype**

--- ------ -------------- -----

(have lat and lng) 0 address 134879 non-null object

(have lat and lng) 1 postal\_code 134879 non-null object

2 floor 134879 non-null float64

(insufficient data) 3 heating 134879 non-null object

(included in category) 4 homeType 134879 non-null object

5 numBathrooms 134879 non-null float64

6 numBedrooms 134879 non-null float64

7 parking 134879 non-null float64

8 price 134879 non-null float64

**(better than price for y)** 9 pricePerSqm 134879 non-null object

10 area 134879 non-null float64

11 year 134879 non-null float64

12 LivingRooms 134879 non-null float64

13 Kitchens 134879 non-null float64

14 WC 134879 non-null float64

(insufficient data) 15 Orientation 134879 non-null object

16 NewlyBuilt 134879 non-null float64

17 Storage 134879 non-null float64

18 Views 134879 non-null float64

19 RoofTop 134879 non-null float64

20 SwimmingPool 134879 non-null float64

21 RoadFront 134879 non-null float64

22 Corner 134879 non-null float64

23 Renovated 134879 non-null float64

(included in Renovated) 24 YearRenovated 134879 non-null object

25 NeedsRenovation 134879 non-null float64

26 ListedBuilding 134879 non-null float64

27 Neoclassico 134879 non-null float64

28 Unfinished 134879 non-null float64

29 ProfessionalUse 134879 non-null float64

(included in SummerHouse) 30 DistanceFromSea 134879 non-null object

(insufficient data) 31 EnergyCat 134879 non-null object

32 Luxury 134879 non-null float64

33 StudentAccomodation 134879 non-null float64

34 SummerHouse 134879 non-null float64

(insufficient data) 35 BuildingZone 134879 non-null object

36 category 134879 non-null float64

37 lat 134879 non-null float64

38 lng 134879 non-null float64

(url related key field) 39 place\_id 134879 non-null object

dtypes: float64(29), object(11)

memory usage: 41.2+ MB

**Deciding on y-variable:**

The first major decision for our analysis, was to choose the appropriate housing value variable between price and price per square meter. The distribution of prices had too many outliers while price per square meter already eliminated one correlated variable to price which is size/area. Of course we would still test area as a x-variable to see if bigger houses command a greater or lower per square meter value.

The Figure 1 graph on the left hand side below, shows that the tail of the prices is much larger than the price per square meter.

Chart, line chart

Description automatically generated

To begin with decided to limit the housing listings we use to those whose price per square meter with -3 and +3 stds of the mean which will significantly reduce the y-variable tail if we were to choose pricePerSqm as the y-variable.

**Chart

Description automatically generated**

**From the above it makes more sense to use pricePerSqm and the y variable in a -3 to +3 std range. We still capture all the price variability by simply multiplying pricePerSqm x area**

**x-variable selection:**



Floor number, area (sqm), and year of construction are significant variable and we will use, but we have some listings with no floor or year of construction. May need to exclude them. Also, some areas are massive 20,000sqm. Need to look closer at this data to figure out why and if it is an outlier to be excluded from the training set. Interesting some apartments? data back to the declaration of US independence.

Chart

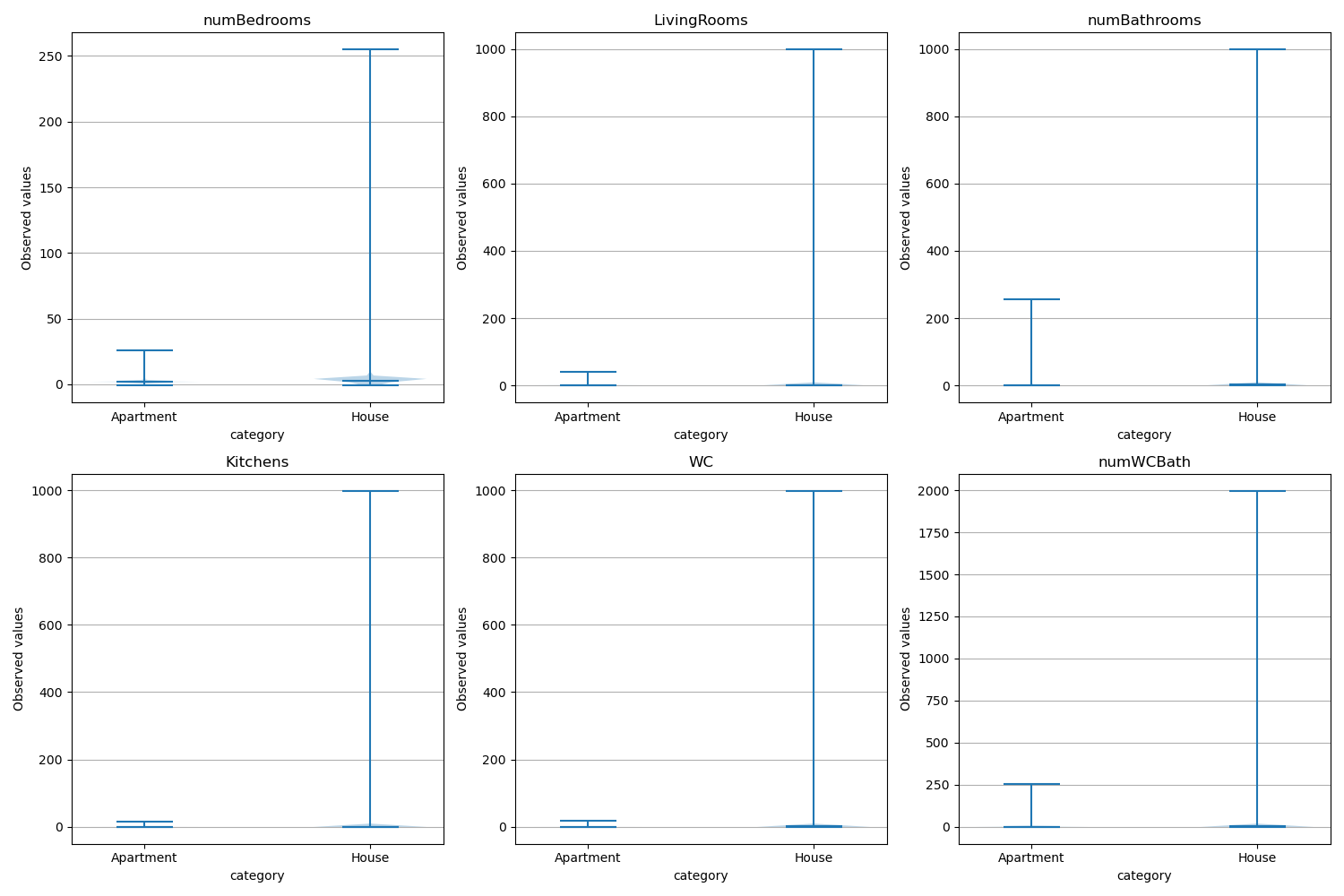
Description automatically generated

I set a limit for area to -3 to +3 stds, and year > 1850, and floor > -2 (ie eliminate listing without defined floor) then the distributions look much better improving the quality of data and reducing outliers and undefined variables. The overall usable listings drop to 113,018 from 134,879

Chart

Description automatically generated

The next batch of significant variables that are somewhat problematic is the number of rooms: living rooms, WCs, bathrooms and kitchens. The number living rooms, bedrooms, bathrooms is unknown in 25% of the case, while only of 25% list number of Kitchens and 50% number of WCs.



The outliers above look massive. Perhaps it is due to apartment blocks that are classified as one house or one apartment.

However, because the there are so many listings that don’t provide the number of the respective rooms our listings drop considerably if we require all 3 conditions to be met:

* eliminating listings if numBedrooms in unknown reduces dataset to 111,044 from 113,018
* then, also if numBathrooms in unknown reduces dataset to 101,179 from 111,044
* and the biggest impact is when we eliminate listing if unknown LivingRooms, we go down to 61279

So perhaps adding numBedrooms+LivingRooms (as many times they are interchangeable) is a better was to do that and in that respect we should probably add the kitchen as well. Also add WC to bathrooms also may make sense. The result is as follows. The listings drop only to 111,539 if we eliminate listings with no bedrooms or living rooms or kitchens and to 103,403 if we also eliminate all listings with no bathrooms or WCs.

**Chart, line chart

Description automatically generated**

Decided to just to eliminate listings with no rooms, but also eliminate the listings which post areas that are more than 200 square meters per room. Listings drop to 110,723

**Chart

Description automatically generated**

**House Features**

The remaining variables relate to house features and usage alternatives. They are Boolean variables either 1 if true or 0 if not true.

**Timeline

Description automatically generated**

I tried to create a positive and a negative variable collection and count the number of positive and negative features that each listing may have to see if the overall score has statistical significance when I develop the prediction model.

Chart

Description automatically generatedChart, bar chart, histogram

Description automatically generated

There is clear upward trend to the median pricePerSqm the more positive features a listing has and negative the more negative features it has.

Also each different home type I have put into 2 categories (House and Apartment) seem to show nice pricePerSqm distributions.

A picture containing text, sky, screenshot, day

Description automatically generated

Chart

Description automatically generated

Chart

Description automatically generatedChart, radar chart

Description automatically generated

I have now a dataset that seems to have good quality data with proper distributions and have created a few more fields that may be relevant to the statistical analysis and the deep learning model I plan to create next.

Improved and cleaned dataset includes 6 new fields.

|  |  |
| --- | --- |
| Int64Index: 110723 entries, 1 to 134877 |  |
| Data columns (total 46 columns): |  |
| # Column Non-Null Count Dtype | # Column Non-Null Count Dtype |
| --- ------ -------------- ----- | --- ------ -------------- ----- |
| 0 address 110723 non-null object | 29 ProfessionalUse 110723 non-null float64 |
| 1 postal\_code 110723 non-null object | 30 DistanceFromSea 110723 non-null object |
| 2 floor 110723 non-null float64 | 31 EnergyCat 110723 non-null object |
| 3 heating 110723 non-null object | 32 Luxury 110723 non-null float64 |
| 4 homeType 110723 non-null object | 33 StudentAccomodation 110723 non-null float64 |
| 5 numBathrooms 110723 non-null float64 | 34 SummerHouse 110723 non-null float64 |
| 6 numBedrooms 110723 non-null float64 | 35 BuildingZone 110723 non-null object |
| 7 parking 110723 non-null float64 | 36 category 110723 non-null float64 |
| 8 price 110723 non-null float64 | 37 lat 110723 non-null float64 |
| 9 pricePerSqm 110723 non-null float64 | 38 lng 110723 non-null float64 |
| 10 area 110723 non-null float64 | 39 place\_id 110723 non-null object |
| 11 year 110723 non-null float64 | 40 numWCBath 110723 non-null float64 |
| 12 LivingRooms 110723 non-null float64 | 41 numRooms 110723 non-null float64 |
| 13 Kitchens 110723 non-null float64 | 42 areaPerRoom 110723 non-null float64 |
| 14 WC 110723 non-null float64 | 43 areaPerWCBath 110723 non-null float64 |
| 15 Orientation 110723 non-null object | 44 posFeatures 110723 non-null float64 |
| 16 NewlyBuilt 110723 non-null float64 | 45 negFeatures 110723 non-null float64 |
| 17 Storage 110723 non-null float64 |  |
| 18 Views 110723 non-null float64 |  |
| 19 RoofTop 110723 non-null float64 |  |
| 20 SwimmingPool 110723 non-null float64 |  |
| 21 RoadFront 110723 non-null float64 |  |
| 22 Corner 110723 non-null float64 |  |
| 23 Renovated 110723 non-null float64 |  |
| 24 YearRenovated 110723 non-null object |  |
| 25 NeedsRenovation 110723 non-null float64 |  |
| 26 ListedBuilding 110723 non-null float64 |  |
| 27 Neoclassico 110723 non-null float64 |  |
| 28 Unfinished 110723 non-null float64 |  |
| dtypes: float64(36), object(10) |  |
| memory usage: 39.7+ MB |  |