Philadelphia Property Value Prediction

By: Billy Witanto, Muhammad Rivaldi Prabowo, Vinsensia Fresian Meiliana

Presentation Outline

Background

- Philadelphia city is one of the **hottest real estate market** in the US.
- To do property transaction we need building price and usually we get this from appraisal.
- With this prediction we can predict the price **without aprraisal** and this will **increase in success transaction.**

Stakeholder

Property Agent

Issue & Why it is important

The price is overvalue / undervalue → There are differences between the price and average market value with the same criteria

Target/Goals

Predict the property price based on its characteristic and/or location

Data Understanding

About Data

Source:https://www.kaggle.com/adebayo/philadelphia-buildings-database
Data maker: Philadelphia Government
Data last update: July 2020
File: 2 file CSV & 1 GEOJSON

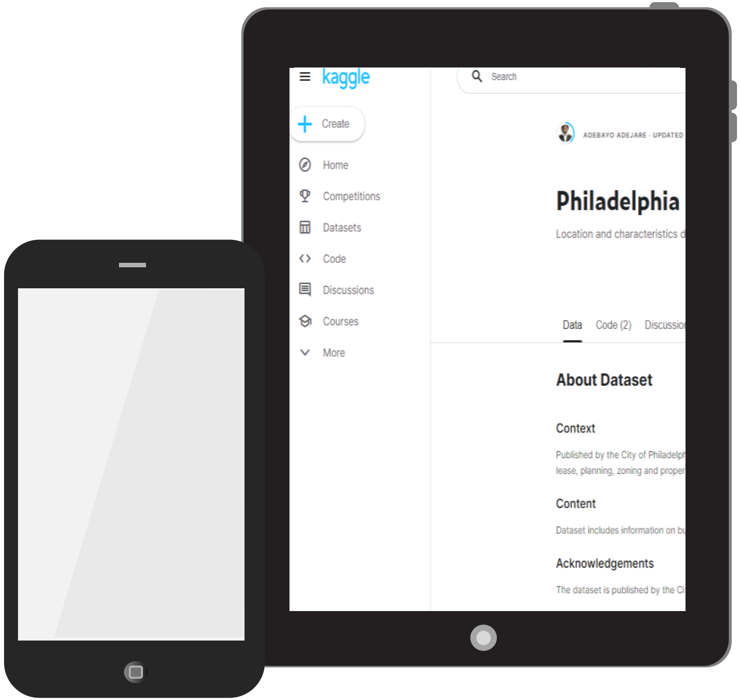
1. Footprints Table: 543278 rows × 12 columns
2. Properties Table: 581456 rows × 75 columns (unique data)

Columns and Rows Interpretation

- Footprint Table
- Properties Table

Columns Use

The list of columns that will or won't be use



Footprints Table Columns Interpretation

	FOOTPRINTS	OPA PROPERTIES	NOTES
Building Identification	OBJECT ID		Identification number
	BIN		Unique building identifier
	FCODE		Building/Skywalk
	BUILDING NAME		Common known
	ADDRESS	**LOCATION**	**It has similar definition to column in OPA Properties**
Building Criteria	**BASE ELEVATION**	**STORIES**	
	APPROX HGT		
	MAX HGT		
	SHAPE AREA	**TOTAL AREA**	
	SHAPE LENGTH	**FRONTAGE**	
	PARCEL ID NUM		Parcel is not an unique identifier for unit/building
	PARCEL ID SOURCE		

Properties Table Columns

To ease the exploratory data analysis process, we grouped the columns based on its types and its connection with other column

Property Location (Use)
Location
Street Designation

Property Location (Drop)
Beginning Point
House Number & Extension
Street Name & Direction

1. Beginning point → No precise info about the coordinate
2. Location = house_number + house_extension + street_direction + street_name + street_designation (Others are part of location)

Classification of Property (Use)
Building Code Desc. & Building Code
Category Code Desc. & Category Code
Unit, Zoning, Zip Code, Unfinished

Classification of Property (Drop)
-

Properties Table Columns Interpretation

Property Specifications (Use)
Market Value, Sale Price, Sale Date
Central Air & Fireplaces
Other Building
Topography & View Type
Exterior & Interior Condition
Quality Grade & General Construction
Total bathrooms, bedrooms, rooms, and stories

Property Specifications (Use)
Year Built
Parcel Number & Shape
Total Area & Livable Area
Frontage & Depth
Basement
Garage Spaces & Type
Fuel & Hyter Type

Property Specifications (Drop)
Off Street Open
Separate Utilities
Sewer
Site Type
Utility

1. Too many missing value
2. Off Street Open : There is no specific definition about it

Properties Table Columns Interpretation

Administration (Use)
Recording Date
Registry Number
Mailing Street
Owner 1 & 2

Administration (Drop)
Exempt Building & Land
Homestead Exemption
Geographic Ward
Mailing Address 1&2
Mailing care of

Administration (Drop)
Mailing City State, Street, & Zip
Market Value Date
Object ID
State & Street Code
Suffix

Administration (Drop)
Taxable Building & Land
Year Built Estimate
Assessment Date
Book and Page
Census Tract
Cross Reference
Date Exterior Condition

1. Logically doesn't impact building price/market value:
   Exemption, Geographic Ward, Book and Page, Census Tract
2. Too many missing value:
   Mailing, Market Value Date, Suffix, Year Built, Assessment Date, Cross Reference, Date Exterior Condition
3. Too many unique value:
   Object ID, State & Street Code, Taxable

Cleaning Method

Check Data Type

Make sure that the data type is already correct.

Check Unique Value

Check is there any anomaly data and whether the data can be use for prediction or not.
(Too many unique data → Overfitting)

Check Missing Value

Make sure there is no NaN data.

Check Anomaly Data & Repair

There are some data that has 0 value or blank, we need to determine whether is actually NaN value or else.

Fill Missing Value

1. Median/Mode
2. Drop data
3. Drop Columns
4. Based on other column

Wrong Data Type, Anomaly Data, and Missing Value

Examples

1.

4.8.2 Year Built

```
df['year_built'].dtypes
```

```
dtype('O')
```

2.

```
df = df.astype({'year_built': float})
```

```
ValueError: could not convert string to float: '196Y'
```

```
# Change '196Y' become 1960 since Y is typo
index=df[df['year_built']=='196Y'].index
df.at[index,'year_built']=1960
```

3.

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

```
array(['1920', '0000', '1960',
       '2014', '2019', '1924',
       '1954', '1939', '1929',
       '1902', '1944', '1981',
```

4.

```
df['year_built'].describe()
```

```
count    546347.000000
mean       1772.411619
std         538.052928
min           0.000000
25%        1920.000000
50%        1925.000000
75%        1950.000000
max        2020.000000
Name: year_built, dtype: floa
```

5.

```
len(df[df['year_built']==0])
```

```
46012
```

6.

```
pd.set_option('display.max_rows',10)
df1=df[(df['year_built']==0)&(~df['building_code_description'].str.contains('VACANT'))&(~df['building_code_description'].str.cont
```

```
df1=df1[df1['location'].isin(dupe['dupe'])].sort_values(by='location')
df1[df1['year_built']!=0][['location','year_built']]
```

location	year_built

```
index=df[df['year_built']==0].index
df.loc[index,'year_built']=np.nan
```

Check Data Type

Data type of year built is wrong ⟶ Change Data Type ⟶ It's error ⟶ Check Error ⟶ Repair

Check Unique Value & Anomaly data

Check unique value ⟶ Anomaly Data ⟶ Check Describe ⟶ Check 0 value ⟶ Check data that can be use to fulfill 0 value

Fill Missing Value

No data can be use ⟶ Change to NaN ⟶ Will be dropped later

Fill Missing Value using Median

Examples

1.

```
i_nan_liv=df[df['total_livable_area'].isna()==True].index
```

```
df.loc[i_nan_liv][['total_area','zoning','building
```

	total_area	zoning	building_code_description
2614	640.00	RSA5	ROW W/GAR 4 STY MASONRY
11830	437.40	RSA5	VACANT LAND RESIDE < ACRE
32320	3390.34	RSA5	VACANT LAND RESIDE < ACRE
529198	1268182.35	SPAIR	CAR/TRUCK RENTAL MASONRY

2.

```
i_nanto0=df[(df['total_livable_area'].isna()==True)&((df['category_code_descri
```

```
df.loc[i_nanto0,'total_livable_area']=0
```

3.

```
lv_otoNAN=df[(df['total_livable_area']==0)&(~df['building_code_description'].str.contains('VACANT'))&(~df['building_code_desc
  &(~df['building_code_description'].str.contains('CAR LOT'))&(~df['category_code_description'].str.contains('Commercial'))
  &(~df['category_code_description'].str.contains('Indus'))
  &(~df['building_code_description'].str.contains('PARKING'))].index
```

```
df.loc[lv_otoNAN,'total_livable_area']=np.NaN
```

```
df[df['total_livable_area'].isna()==True].index
```

There are 427 NaN data that will be changed using median data which grouped by building code.

```
df['total_livable_area']=df.groupby('building_code_description')['total_livable_area'].apply(lambda x: x.fillna(x.median()))
```

```
df['total_livable_area'].isna().sum()
```

```
26
```

Check Missing Value & Anomaly Data

Check NaN data → Change data that should have 0 livable area → Check Anomaly data (0 value for residential) → Change to NaN

Fill Missing Value

Change NaN data using median (group by building code) → Rest NaN data will be drop later

Drop Column

Examples

1.

```
df['general_construction'].value_counts()
```

```
A     443296
B      31412
E      11735
C      10616
```

```
df['general_construction'].isna().sum()
```

```
64155
```

```
df['general_construction'].isna().sum()/len(df)*100
```

```
11.036166215965155
```

2.

```
df['quality_grade'].unique() # Ordinal
```

```
array([nan,  3.,  6.,  4.,  2.,  5.,  1.,  0.])
```

```
df['quality_grade'].isna().sum()
```

```
554297
```

```
df['quality_grade'].isna().sum()/len(df)*100 #
```

```
95.35209765428786
```

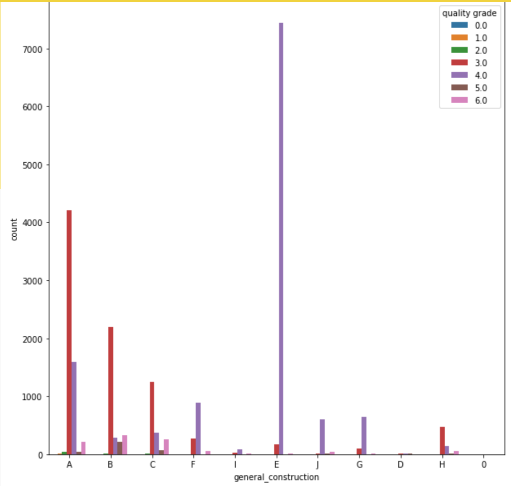
3.

```
df.groupby('general_construction')[['quality_grade','interior_condition','exterior_condition']].median()
```

general_construction	quality_grade	interior_condition	exterior_condition
0	3.0	3.0	3.0
A	3.0	4.0	4.0
B	3.0	4.0	4.0

```
df.groupby('general_construction')[['quality_grade','interior_condition','exterior_condition']].mean()
```

general_construction	quality_grade	interior_condition	exterior_condition
0	3.000000	2.914286	2.885714



Check Unique Value & Missing Value

Check unique value ⟶ Check missing value ⟶ Check pattern ⟶ No pattern

Fill Missing Value?

Drop column

Detailed Exploratory Data Analysis (EDA)

This section will help us to:

1. Understand the characteristics of variables
2. Discover the relationships between variables

Post-Cleaning Variables

28 original columns + 6 new columns

Building code description	Central air	Depth	View type	Number stories	Street designation	Parcel number
Category code description	Fireplaces	Frontage	Sale price	Number of rooms	Zip code	Parcel shape
[1] Exterior condition	Unfinished	Total area	Market value	Number of bedrooms	Topography	Location
[1] Interior condition	Other building	Total livable area	[2] Sale date	Number of bathrooms	[5] Zoning	[3] Year built
↓			↓		↓	↓
[1] Overall condition			[2] Sale year		[5] New zoning	[3] Property age
			[2] Sale year group			

Column Type					
Categorical	Ordinal	Boolean	Numerical	Date Time	Unique Values

Already Dropped

[4] Parking spaces ← [4] Garage spaces

Post-Cleaning Variables

13 col.

Location	
Parcel shape	Sale year group
Building code description	Street designation
Category code description	Zip code
Central air	Topography
View type	Zoning
Parking spaces	New zoning

14 col.

Property age	Year built	
Depth	Fireplaces	Number stories
Frontage	Sale price	Number of rooms
Total area	Market value	Number of bedrooms
Total livable area	Sale year	Number of bathrooms

3 col.

Exterior condition
Interior condition
Overall condition

2 col.

Unfinished
Other building

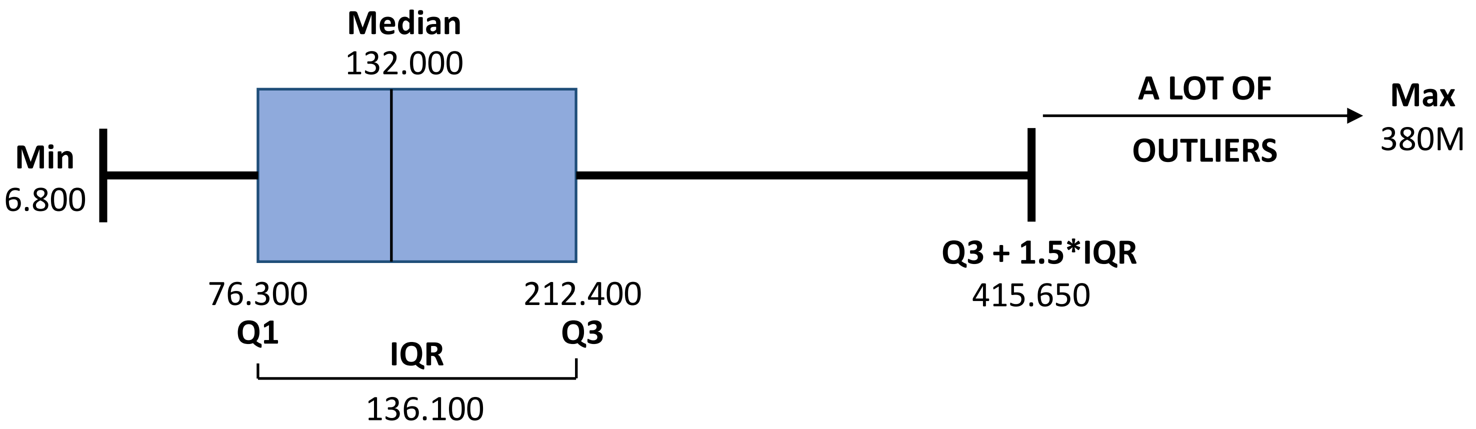
1 col. each

Parcel number
Sale date

Total rows: 495.390

Column Type					
Categorical	Ordinal	Boolean	Numerical	Date Time	Unique Values

Label Analysis: Market value
Distribution



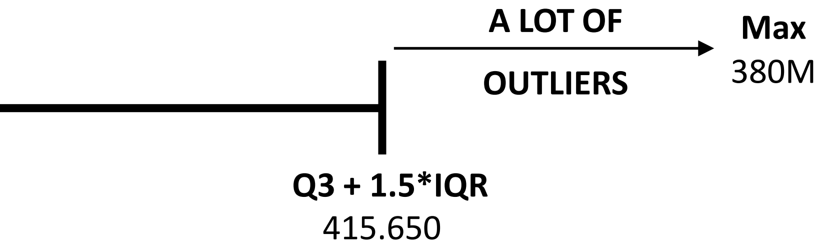
Market Value Range	Counts (Percentage)	
6.800 to 76.300	123.885	25,01%
>76.300 to 132.000	123.826	25,00%
>132.000 to 212.400	124.349	25,10%
>212.400 to 415.650	87.013	17,56%

Dataset is **dominated by** property market value in range of **6.800 to 415.650 USD.**

Even without the presence of outliers, the distribution is already **positively skewed.**

Label Analysis: Market value

Outliers Distribution



Market Value Range	Counts (Percentage)	
>415.650 to 618.900	19.753	3,99%
>618.900 to 1M	9.358	1,89%
>1M to 2M	4.232	0,85%
>2M to 10M	2.326	0,47%
>10M to 100M	590	0,12%
>100M	58	0,01%

With the presence of outliers, the distribution became **extremely not normal.**

Label Analysis: Market value

It is important to note that the dataset is:

A REAL-WORLD DATA

We want our model to cover the outliers as well. Thus, carefully handling the outliers is a must.

Recommendation

1. Drop the outliers with contextual outlier analysis.
2. Transform the label with transformation, scaling, etc.

Analysis Workflow

Location							
Parcel shape	Sale year group						
Building code description	Street designation	Property age	Year built				
Category code description	Zip code	Depth	Fireplaces	Number stories			
Central air	Topography	Frontage	Sale price	Number of rooms	Exterior condition		
View type	Zoning	Total area	Market value	Number of bedrooms	Interior condition	Unfinished	Parcel number
Parking spaces	New zoning	Total livable area	Sale year	Number of bathrooms	Overall condition	Other building	Sale date

Column Type					
Categorical	Ordinal	Boolean	Numerical	Date Time	Unique Values

Analysis Workflow

1. Eliminate unnecessary columns

Parcel shape
Building code description
Category code description
Central air
View type
Parking spaces

Sale year group
Street designation
Zip code
Topography
Zoning
New zoning

Property age
Frontage
Total area
Total livable area

Depth
Fireplaces
Market value
Sale year

Number stories
Number of rooms
Number of bedrooms
Number of bathrooms

Unfinished
Other building

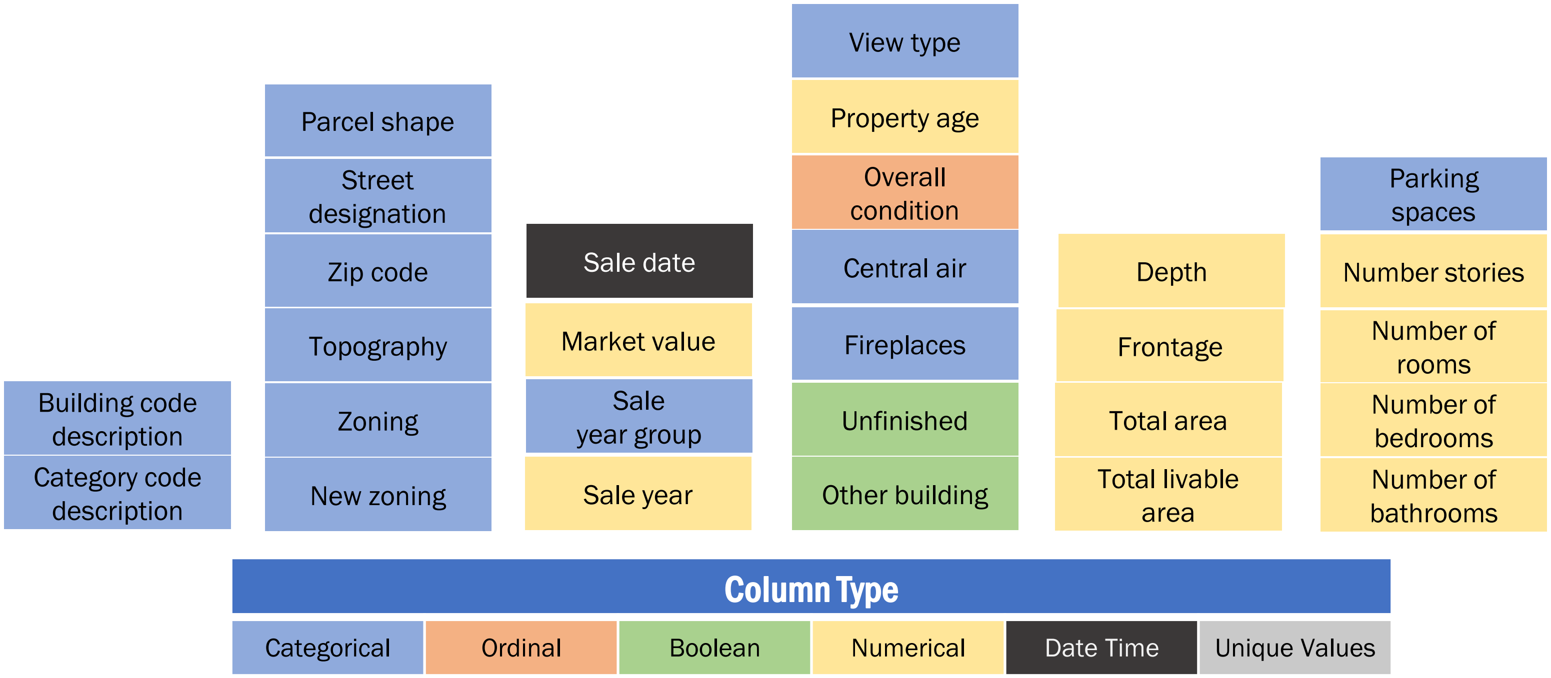
Overall condition

Sale date

Column Type					
Categorical	Ordinal	Boolean	Numerical	Date Time	Unique Values

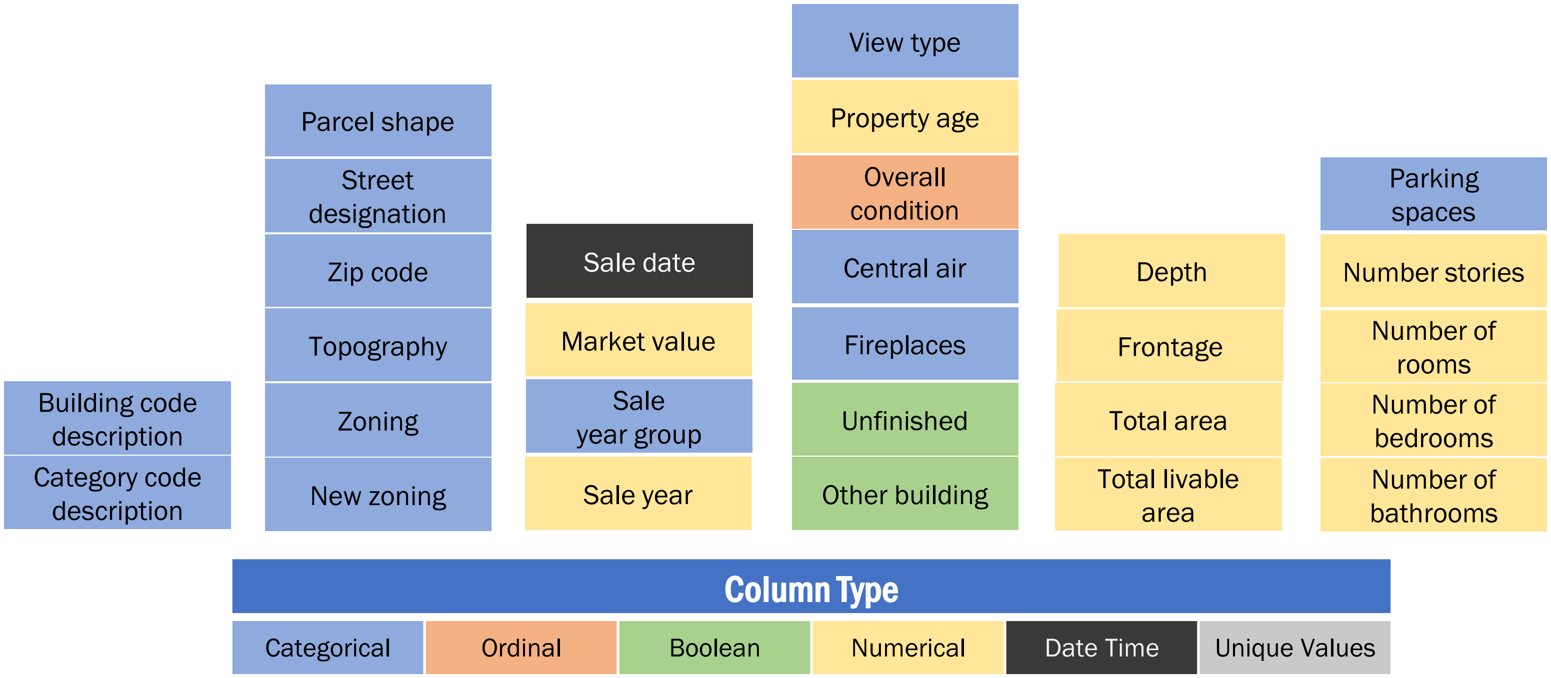
Analysis Workflow

2. Grouping



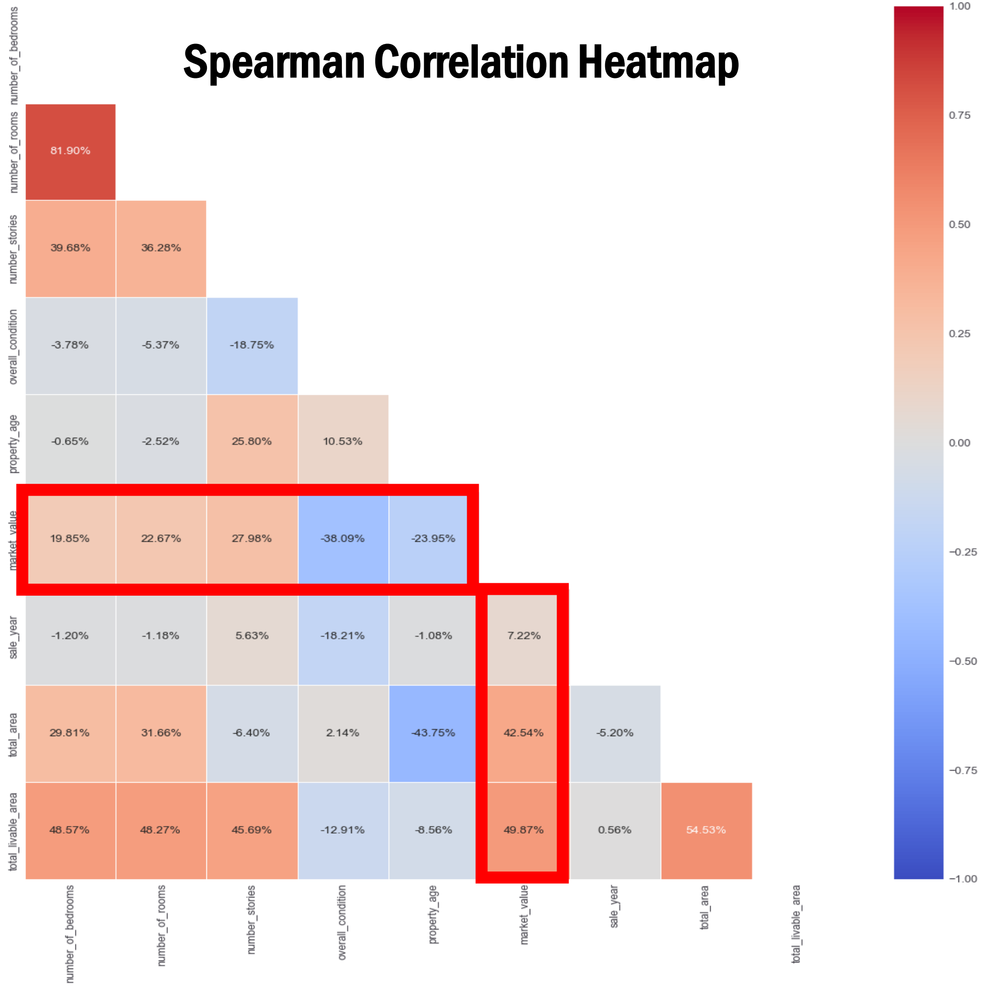
Analysis Workflow

3. Approach: Univariate, Bivariate (Feature vs Feature, Feature vs Target)



Numerical Features and Market Value Correlations

Spearman Correlation Heatmap



Numerical Features Distribution

	feature	statistics	p-value
0	number_of_bedrooms	1.148223e+06	0.0
1	number_of_rooms	1.183700e+06	0.0
2	number_stories	9.678649e+05	0.0
3	overall_condition	1.260157e+05	0.0
4	property_age	5.582315e+04	0.0
5	sale_year	1.132798e+05	0.0
6	total_area	3.367957e+06	0.0
7	total_livable_area	1.691928e+06	0.0

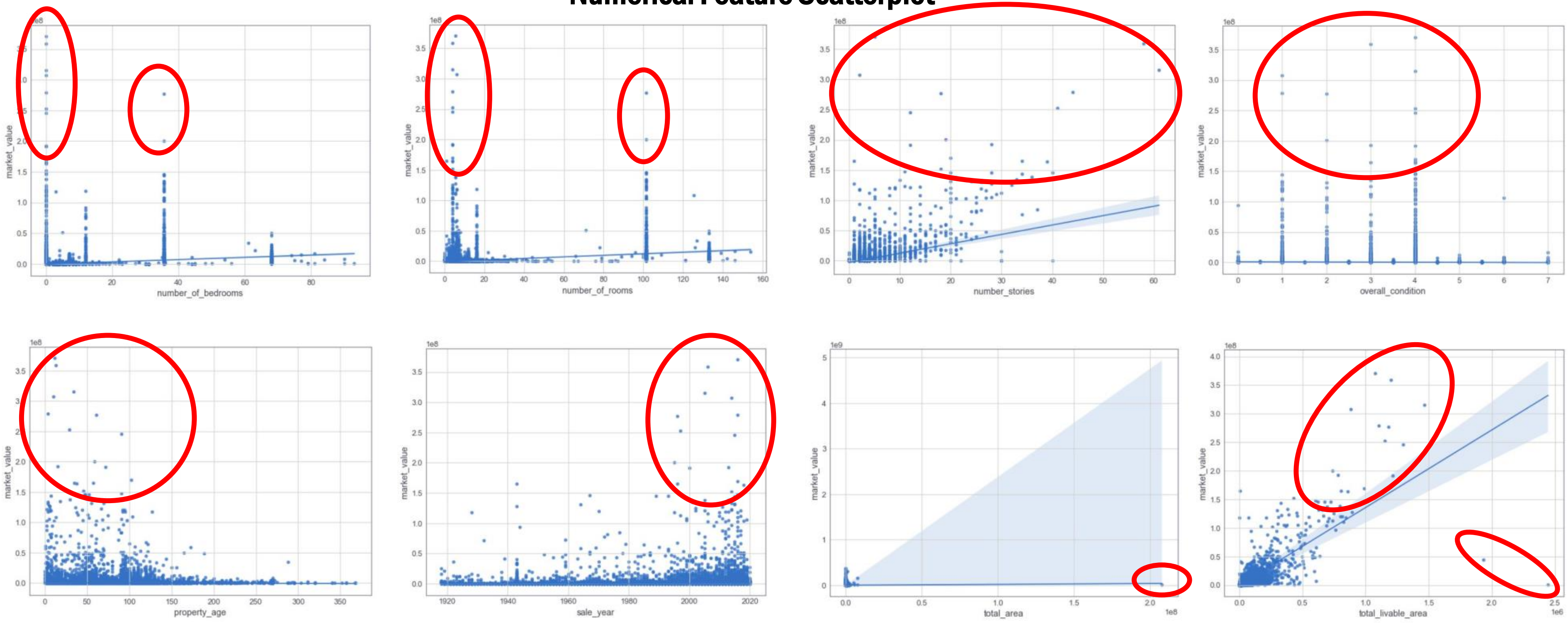
Most of the numerical features are not normally distributed.

Top 5 highest correlation features with market value:

1. Total livable area (49.57%)
2. Total area (42.54%)
3. Overall condition (-38.09%)
4. Number stories (27.98%)
5. Property age (-23.95%)

Contextual Outlier Analysis

Numerical Feature Scatterplot



Identified Contextual Outliers

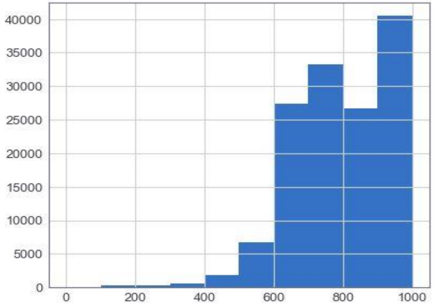
Analysis:

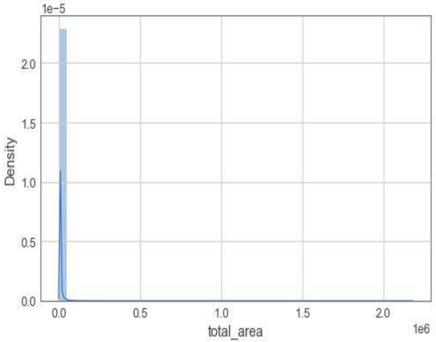
1. Most of the contextual outliers appear above 150.000.000 of market value (drop data).
2. Values above 2.500.000 in total area are considered to be outliers (drop data).
3. Values above 1.250.000 in total livable area are considered to be outliers (drop data).

Outlier Analysis

Removing Extreme Outliers (above Q1 and Q3) from total area and total livable area

Total area

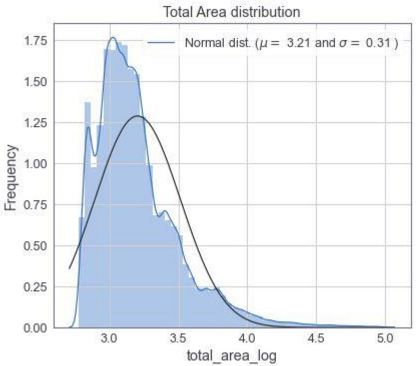




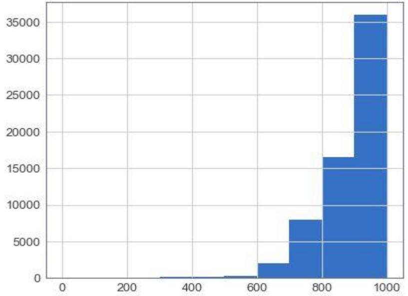
From histogram above, we considered there are abrupt value change in 600, so we can consider it as extreme outliers Q1 value (drop data).

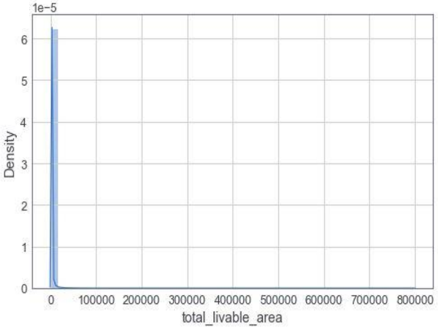
From histogram above, we consider data above 100.000 feet are extreme outliers Q3 value (drop data).

Total area final data distribution in log curve



Total livable area

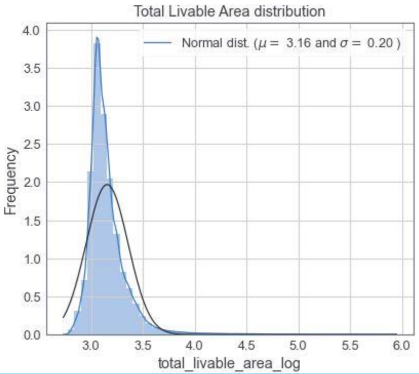




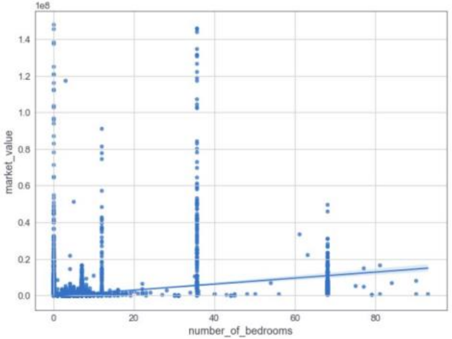
From histogram above, we considered there are abrupt value change in 600, so we can consider it as extreme outliers Q1 value (drop data).

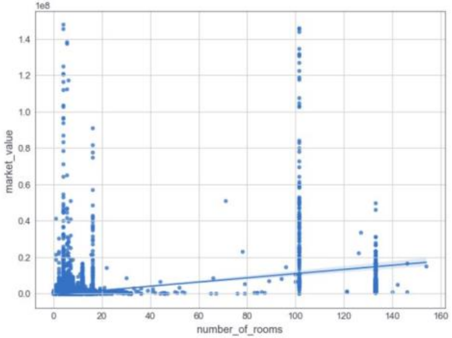
Early histogram of total livable area.

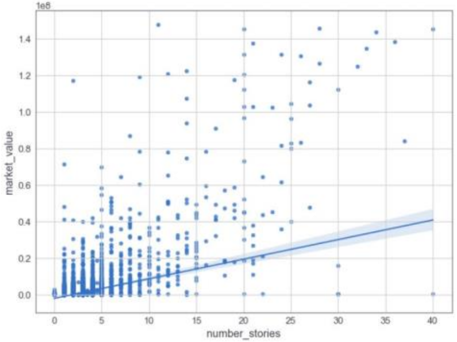
Total livable area final data distribution in log curve

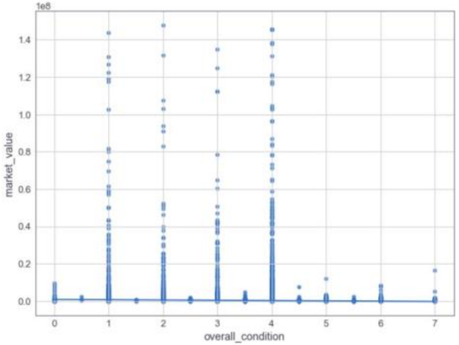


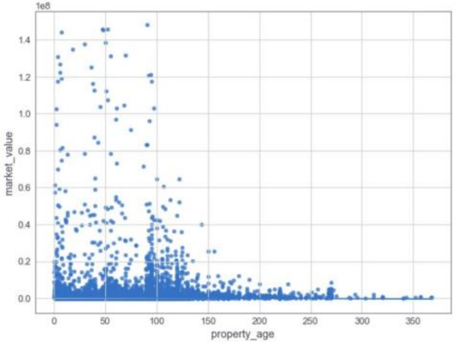
Final Numerical Features Scatterplots

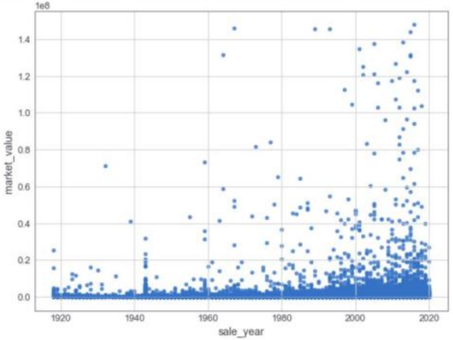


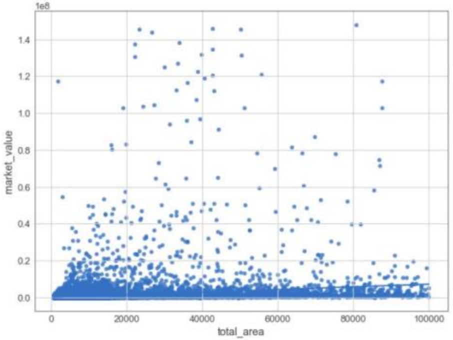


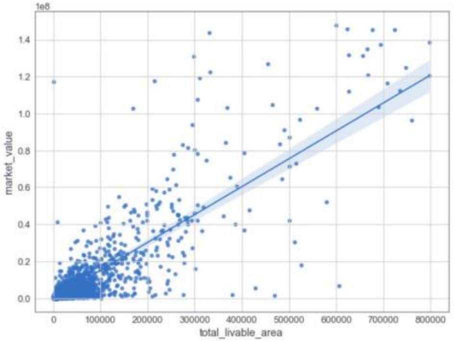








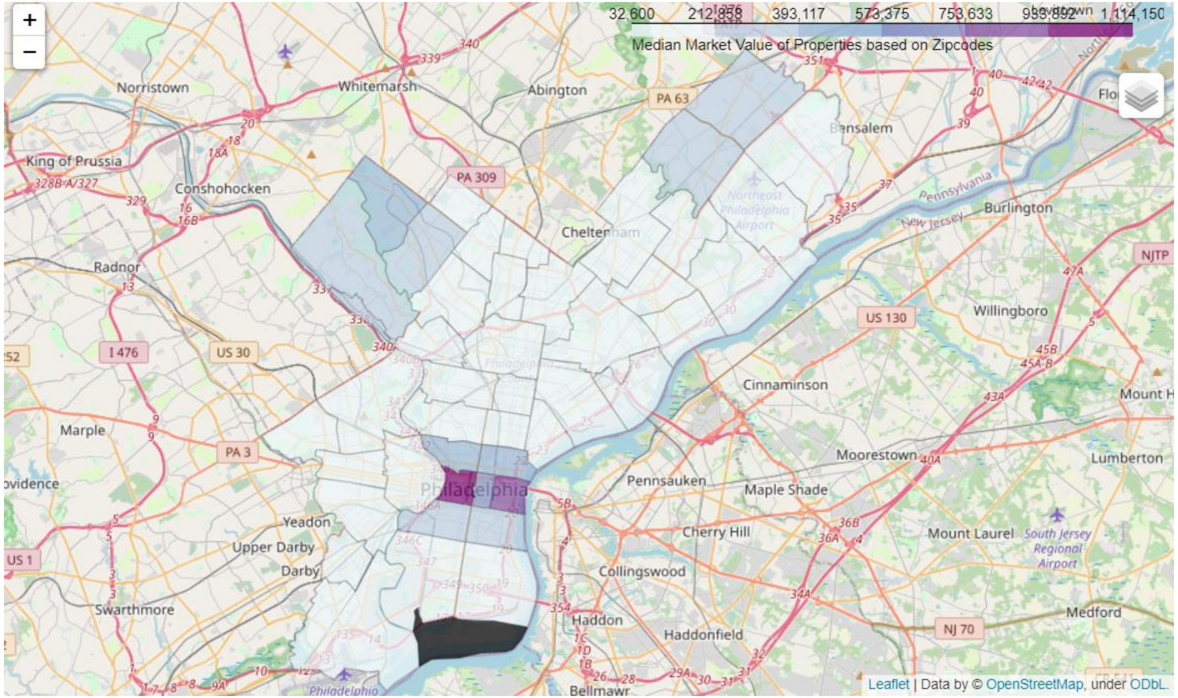




Location Analysis

Zip codes and market value

Outliers were affecting the scale

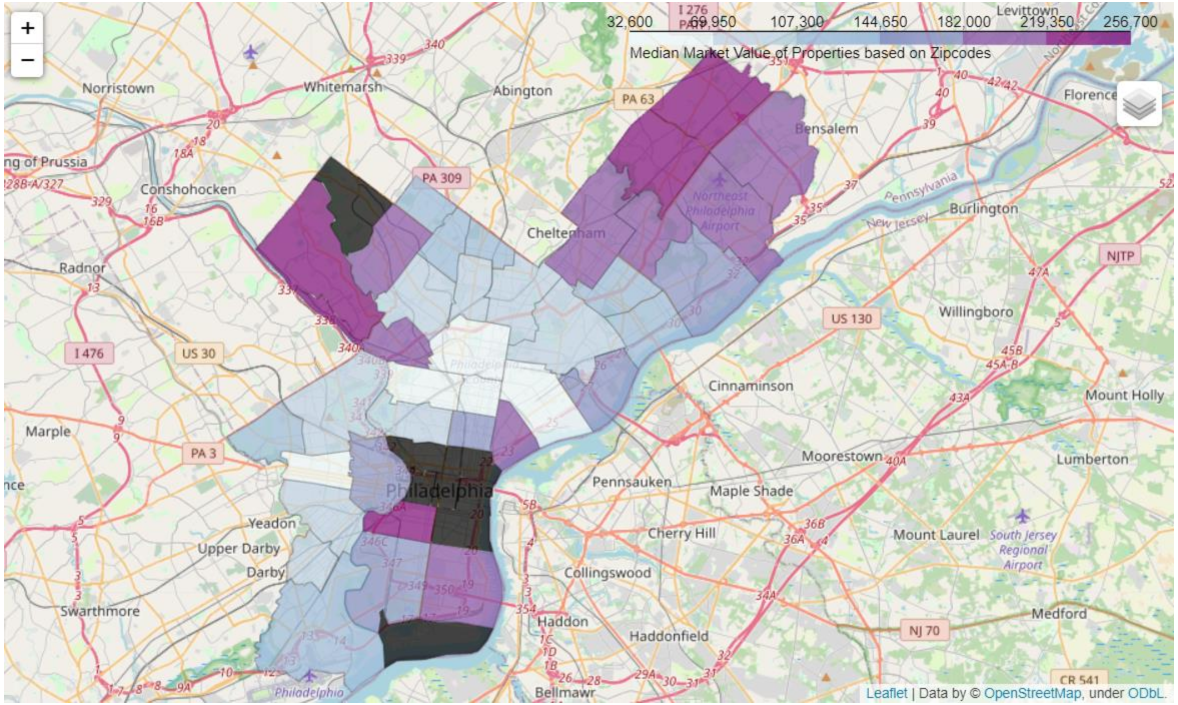


Market value was affected by its distance to city center. Why the price is high in the most far zip codes though?

Location Analysis

Zip codes and market value

Outliers dropped

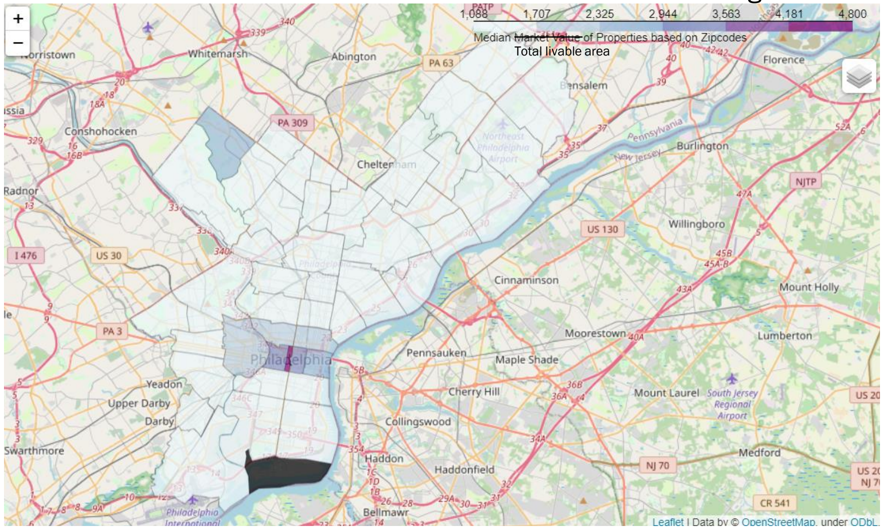


Market value was affected by its distance to city center. Why the price is high in the most far zip codes though?

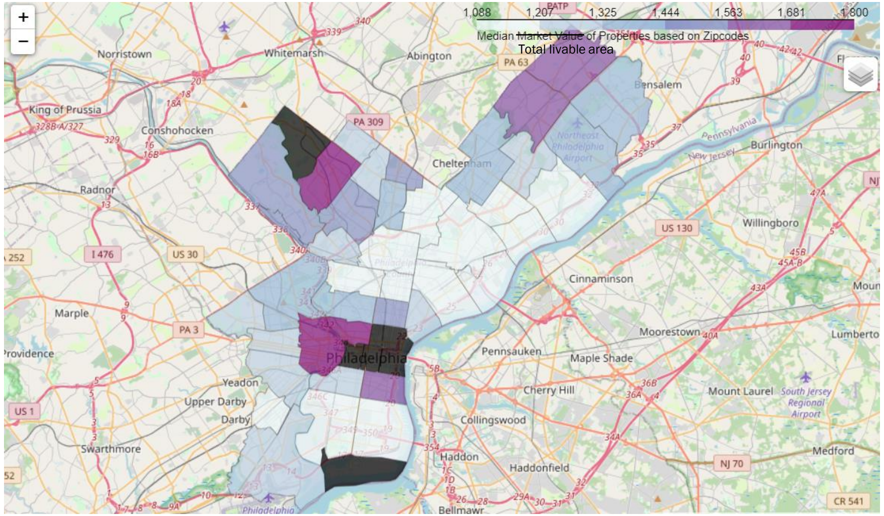
Location Analysis

Zip codes and total livable area

Outliers were affecting the scale



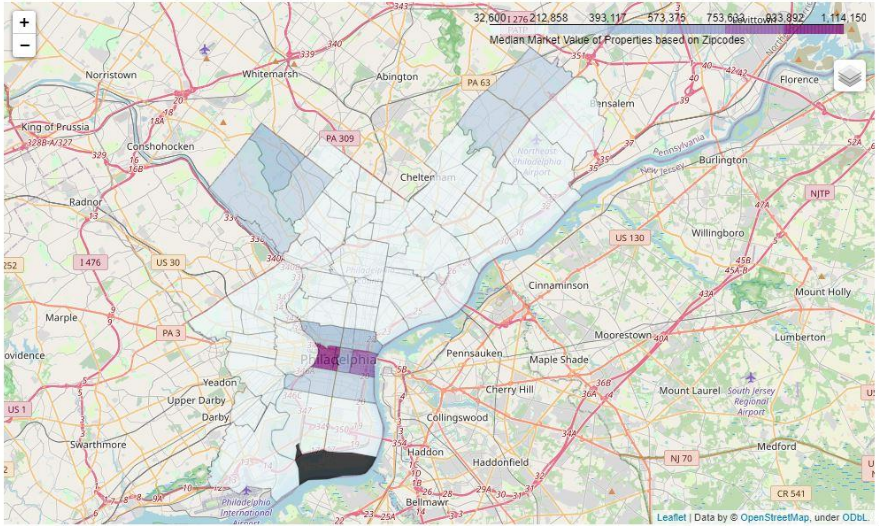
Outliers dropped



The further the property from city center, the higher its median of total livable area. Explaining the high median market value in previous folium maps.

Feature Engineering for Model 2

1. Distance based on Zipcode



Grouped by distance from city center

City Center : [19103, 19102, 19107, 19106, 19130, 19123, 19146, 19147]
Adjacent City Center : [19121, 19122, 19125, 19145, 19148, 19104]
Near City Center : [19112, 19153, 19142, 19143, 19139, 19151, 19131, 19132, 19133, 19134]
Far From City Center : [19129, 19140, 19124, 19137, 19144, 19141, 19120, 19135, 19149]
Very Far From City Center NW : [19128, 19118, 19119, 19150, 19138, 19126, 19127]
Very Far From City Center NE : [19111, 19152, 19136, 19115, 19114, 19116, 19154]

***Reference**: Greater Center City Housing: 2020 Strong Fundamentals (Interrupted) Center City District, Central Philadelphia Development Corporation*

2. Simplify fireplaces

```
df['have_fireplaces']=df['fireplaces'].apply(lambda x: 0 if x=='0' else 1)
```

executed in 188ms, finished 16:11:13 2022-03-22

```
df['have_fireplaces'].value_counts()
```

executed in 13ms, finished 16:11:13 2022-03-22

```
0    470774
1     13284
Name: have_fireplaces, dtype: int64
```

Feature Engineering for Model 2

3. Extracting words and grouping it based on building code description

The idea is we use regex or text mining method to extract particular words from building code and then grouped it into four categories: (House, Condominium, Apartment, Others)

1. Make "Condominium" column (fill with 0 and 1)

```
df['condominium']=df['building_code_description'].apply(lambda x: 1 if (('CONDO' in x)&('PARKING' not in x)) else 0)
```

```
df['condominium'].sum()
```

executed in 14ms, finished 16:11:14 2022-03-22

776

2. Make "Apartment" column (fill with 0 and 1)

```
df['apartment']=df['building_code_description'].apply(lambda x: 1 if ('APT' in x) else 0)
```

```
df['apartment'].sum()
```

executed in 14ms, finished 16:11:15 2022-03-22

48234

3. Make "House" column (fill with 0 and 1)

```
house_index=df[~df['building_code_description'].str.contains('|'.join(list1))].index
```

```
#list word in building_code_description that can describe everything other than house
list1=['CONDO','PARKING','SPACE','LOFT','MISC','APT','HOTEL','MOTEL','OFF','STR','STORE','WORSHIP','AUTO','PKG','BLD','SCHOOL',
 'REST','BAR','CARE','HEALTH','PARK','AMUSE','VACANT','TAVERN','CLEANING','CEMETERY','WAREHOUSE','MFG','JUNKYARD','VACANT']
```

```
df.loc[house_index,'house']=1
```

executed in 46ms, finished 16:11:17 2022-03-22

```
df['house'].sum()
```

executed in 13ms, finished 16:11:17 2022-03-22

416567

4. Make "Others" column (fill with 0 and 1)

```
other_index=df[(df['building_code_description'].str.contains('|'.join(list1)))&
          (~df['building_code_description'].str.contains('APT'))&
          ~((df['building_code_description'].str.contains('CONDO'))&
            (~df['building_code_description'].str.contains('PARKING')))].index
```

```
df.loc[other_index,'other']=1
```

executed in 14ms, finished 16:11:20 2022-03-22

```
df['other'].sum()
```

executed in 14ms, finished 16:11:20 2022-03-22

18481

Make list "a" fill with "Condominium", "Apartment", "House", "Others" based on value "1" in each columns before.

```
a=[]
for i in range(len(df)):
    if df['condominium'].loc[i]==1:
        a.append('Condominium')
    elif df['apartment'].loc[i]==1:
        a.append('Apartment')
    elif df['house'].loc[i]==1:
        a.append('House')
    else:
        a.append('Other')
```

```
len(a)
```

executed in 14ms, finished 16:11:46 2022-03-22

484058

```
df['new_building_desc']=a
```

executed in 30ms, finished 16:11:46 2022-03-22

Make "new_building_desc" column filled with value from list "a"

Feature Selection for Model 1

Numerical Features

Features after data analysis and data cleansing:

1. depth
2. exterior_condition
3. frontage
4. interior_condition
5. number_of_bathrooms
6. number_of_bedrooms
7. number_of_rooms
8. number_stories
9. parcel_number
10. overall_condition
11. property_age
12. sale_year
13. total_area
14. total_livable_area
15. year_built

Domain knowledge, EDA, and outlier analysis

Selected features:

1. number_of_bedrooms
2. number_of_rooms
3. number_stories
4. overall_condition
5. property_age
6. sale_year
7. total_area
8. total_livable_area

Reason to unuse several features:

1. overall_condition columns is the combination between exterior_condition and interior condition.
2. total_area values are multiplication value of depth and frontage.
3. property_age come from year_built column.
4. parcel_number, because it contain many unique values.
5. It's not common to use number_of_bathroom as feature in house pricing.

Categorical Features

Features after data analysis and data cleansing:

1. building_code_description
2. category_code_description
3. central_air
4. fireplaces
5. location
6. new_zoning
7. other_building
8. parcel_shape
9. parking_spaces
10. sale_date
11. sale_year_group
12. street_designation
13. topography
14. unfinished
15. view_type
16. zip_code
17. zoning

Domain knowledge, EDA, and outlier analysis

Selected features:

1. building_code_description
2. category_code_description
3. central_air
4. fireplaces
5. new_zoning
6. other_building
7. parcel_shape
8. parking_spaces
9. street_designation
10. topography
11. unfinished
12. view_type
13. zip_code

Reason to unuse several features:

1. location, because it contain many unique values.
2. sale_date and sale_year_group, because it contain many unique values and can be replace by sale_year value.
3. zoning, because we replace its values by create new_zoning columns.

Feature Selection for Model 2

Numerical Feature

Features after data analysis and data cleansing:

1. depth
2. exterior_condition
3. frontage
4. interior_condition
5. number_of_bathrooms
6. number_of_bedrooms
7. number_of_rooms
8. number_stories
9. parcel_number
10. overall_condition
11. property_age
12. sale_year
13. total_area
14. total_livable_area
15. year_built

Domain knowledge, EDA, and outlier analysis

Selected features:

1. number_of_bedrooms
2. number_of_rooms
3. number_stories
4. overall_condition
5. property_age
6. sale_year
7. total_area
8. total_livable_area

Reason to unuse several features:

1. overall_condition columns is the combination between exterior_condition and interior condition.
2. total_area values are multiplication value of depth and frontage.
3. property_age come from year_built column.
4. parcel_number, because it contain many unique values.
5. It's not common to use number_of_bathroom as feature in house pricing.

Categorical Feature

Features after data analysis and data cleansing:

1. building_code_description
2. category_code_description
3. central_air
4. fireplaces
5. location
6. new_zoning
7. other_building
8. parcel_shape
9. parking_spaces
10. sale_date
11. sale_year_group
12. street_designation
13. topography
14. unfinished
15. view_type
16. zip_code
17. zoning

Domain knowledge, EDA, and outlier analysis

Selected features:

1. apartment
2. condominium
3. category_code_description
4. central_air
5. Distance_city_center
6. have_fireplaces
7. house
8. new_zoning
9. other_building
10. parcel_shape
11. parking_spaces
12. street_designation
13. topography
14. unfinished
15. view_type
16. zip_code

Reason to unuse several features:

1. building_code_description, because already grouped into condominium, apartment, and house.
2. location, because it contain many unique values.
3. sale_date and sale_year_group, because it contain many unique values and can be replace by sale_year value.
4. zoning, because we replace its values by create new_zoning columns.

Modeling

Pre-Processing Scheme

Model 1

Label: market value

Numerical Features
Robust Scaling

1. total_area
2. total_livable_area
3. property_age

Categorical Features
One Hot encoding

1. central_air (3 unique values)
2. other_building (2 unique values)
3. unfinished (2 unique values)

Categorical Features
Binary encoding

1. building_code_description (444 unique values)
2. category_code_description (6 unique values)
3. fireplaces (5 unique values)
4. new_zoning (11 unique values)
5. parcel_shape (5 unique values)
6. parking_spaces (7 unique values)
7. street_designation (23 unique values)
8. topography (7 unique values)
9. view_type (8 unique values)
10. zip_code (52 unique values)

Do not preprocessed

1. number_of_bedrooms
2. number_of_rooms
2. number_stories
3. overall_condition
4. sale_year

Model 2

Label: market value

Numerical Features
Robust Scaling

1. total_area
2. total_livable_area
3. property_age

Categorical Features
One Hot encoding

1. central_air (3 unique values)
2. have_fireplaces (2 unique values)
3. other_building (2 unique values)
4. unfinished (2 unique values)

Categorical Features
Binary encoding

1. category_code_description (6 unique values)
2. distances_from_city (7 unique values)
3. new_zoning (11 unique values)
4. parcel_shape (5 unique values)
5. parking_spaces (7 unique values)
6. street_designation (23 unique values)
7. topography (7 unique values)
8. view_type (8 unique values)
9. zip_code (52 unique values)

Do not preprocesed

1. apartment
2. condomium
3. house
4. number_of_bedrooms
5. number_of_rooms
6. number_stories
7. overall_condition
8. sale_year

Data Splitting, Algorithm Model

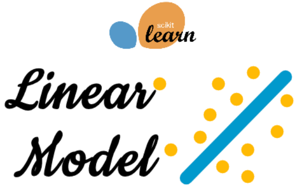
Data Splitting

```
y=df['market_value']
x=df_model
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=2020)
executed in 326ms, finished 11:20:28 2022-03-22
```

Algorithm Model

Parametric model



Linear Regression: This is the first model we propose as base model in cross-validation. Because this model is commonly used when it comes to regression modelling, also this model represented parametric method in regression modelling.

Non-Parametric model (Dataset isn't normally distributed)

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Random Forest is also a "Tree"-based algorithm that uses the qualities features of multiple Decision Trees for making decisions. The Random Forest algorithm is also very *fast* and *robust* than other common regression models (*towardsdatascience.com*).

Extreme Gradient Boosting Regression refers to a class of ensemble machine learning algorithms Ensembles are constructed from decision tree models. This gives the technique its name, "*gradient boosting*," as the loss gradient is minimized as the model is fit, much like a neural network. XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems.
(*machinelearningmastery.com*)

Metric Evaluation and Cross Validation

Metric Evaluation

1. R-squared

$$1 - \frac{SSE}{SST}$$

2. Mean squared error (MSE)

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

3. Root mean squared error (RMSE)

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Yi - \hat{Y}i)^2}$$

4. Mean absolute error (MAE)

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

5. Mean absolute percentage error (MAPE)

$$\sum_{t=1}^{n}\left|\frac{y_i - \hat{y}_i}{\hat{y}_i}\right| \times 100\%$$

In this case, where market value (numerical value) is the target, so we need an evaluation metric which can provide exact error value and don't forget that market_value cannot be in negative value (always positive). Target has a wide range value start from 5.500 to 1.500.000. To avoid bias in evaluation metric/error, we choose **mean percentage absolute error (MAPE) as main evaluation metric.** That's because this metric can well adjust with those wide range target value and don't stick to fixed error value.

Cross Validation

Model 1

	model	mean	std
0	LinReg	-0.854621	0.034950
1	Forest	-0.144754	0.004472
2	XGB	-0.348973	0.013519

Model 2

	model	mean	std
0	LinReg	-0.906468	0.029465
1	Forest	-0.133209	0.003265
2	XGB	-0.275069	0.001754

Cross Validation Conclusion

Mean absolute percentage error (MAPE) of the Random Forest model has the lowest error score **(14.47%) in Model 1** and Random Forest model has the lowest error score **(13.32%) in Model 2,** also the algoritm has the most stable (lowest standard deviation) from all models. This means the algorithm can reduce error so well, so for this model we choose **`Random Forest Regressor`** as selected base algorithm model.

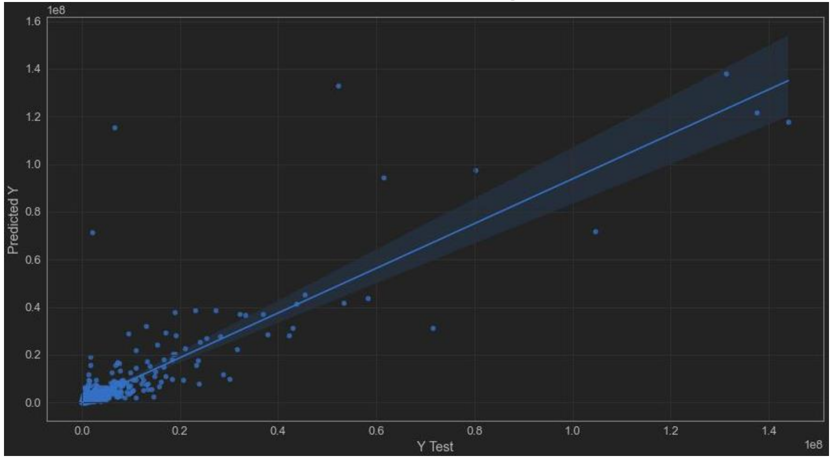
Random Forest Model Performance

Model 1

Metric Evaluation

```
MAPE SCORE:  0.13189944109111362
MAE SCORE:  36094.64084981334
MSE SCORE:  370977213985.95074
RMSE SCORE:  609078.988297865
R2 SCORE:  0.7217951203740047
```

Y Predict – Y Test Regplot

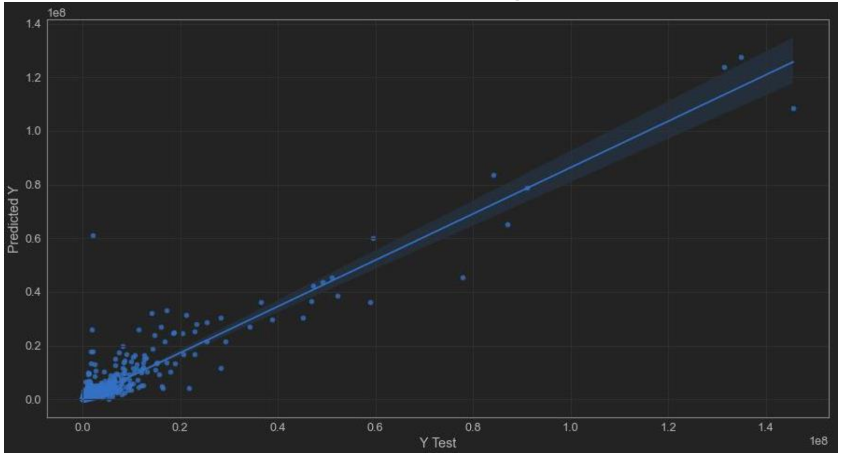


Model 2

Metric Evaluation

```
MAPE SCORE:  0.12255710383575225
MAE SCORE:  32380.079172474398
MSE SCORE:  149194029999.23434
RMSE SCORE:  386256.4303661938
R2 SCORE:  0.8927758053479059
```

Y Predict – Y Test Regplot

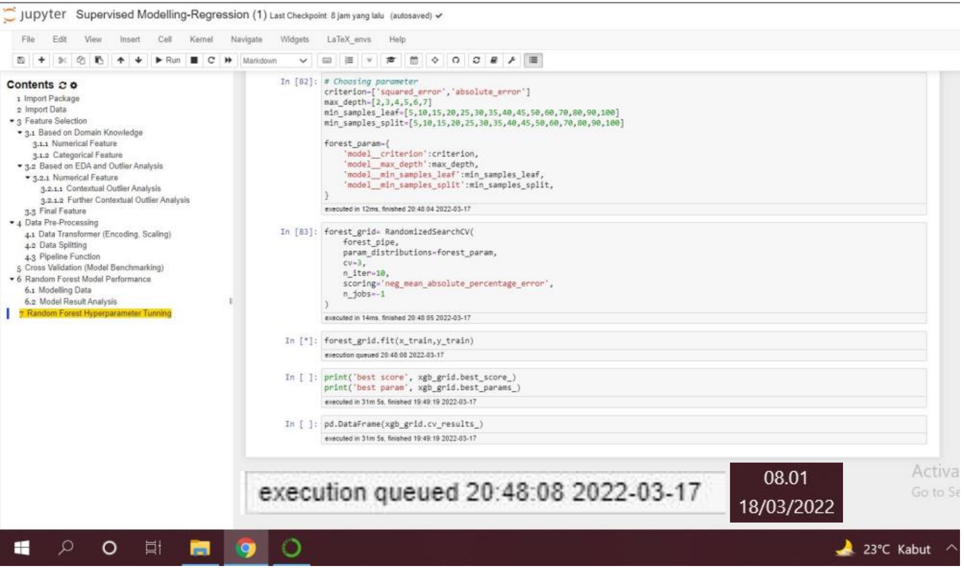


Model Performance Test Analysis

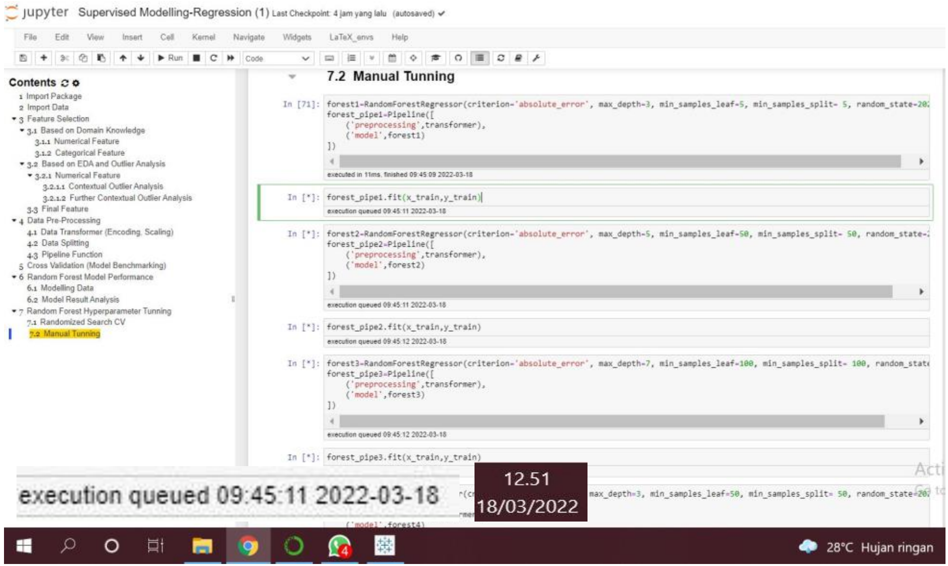
From model performance using data test we can see above Model 2 has a better MAPE score (12,2557%) than Model 1 (13,1899%), also other metric show Model 2 is better than Model 1. If the MAPE score is under 20% than it can be categorized as good and if it below 10% than it can be categorized as excellent. Our MAPE score show that the percentage of error is around 13% which mean our model has a good accuracy (*https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119199885.app1*). Extra feature engineering has an impact to improve model performance, so **we choose Model 2** for further analysis and feature engineering.

Hyperparameter Tuning

Randomized Search CV Tuning



Manual Tuning

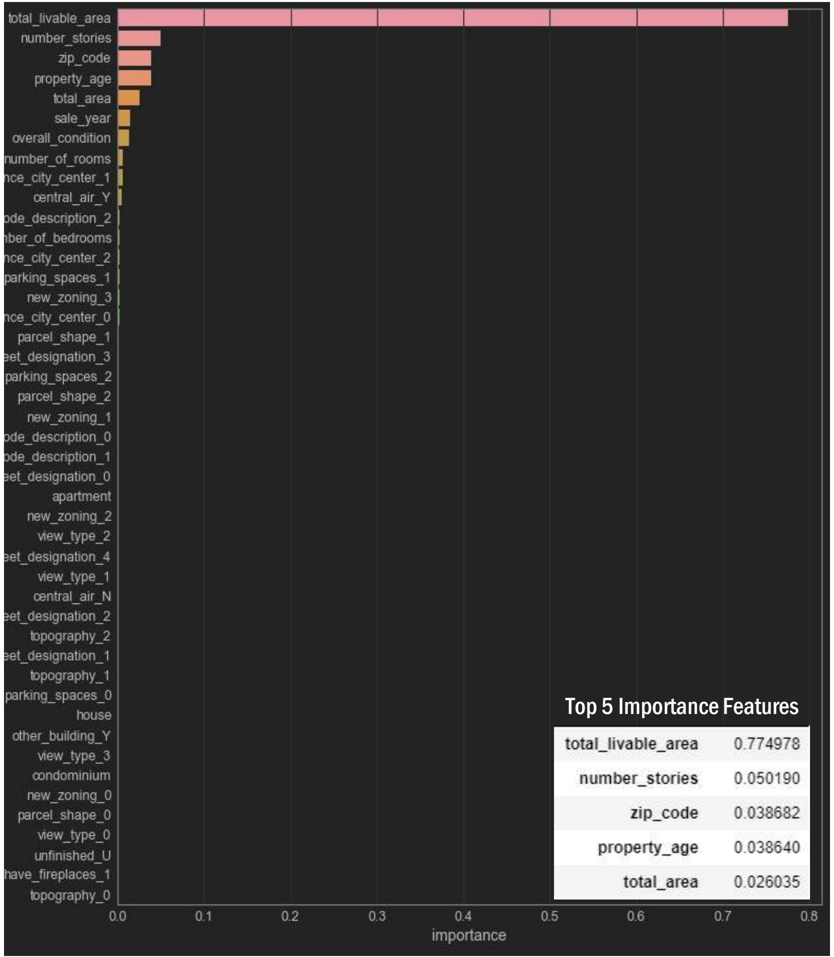


From the screenshot above, we tried to do hyperparameter tuning with **RandomizedSearchCV**, but our **computational power is not enough** to process it quickly. Execution was run in 20:48 in 17 March 2022 but still didn't finish in 08:01 in 18 March 2022 (almost 12 hours), so we decided to interrupt the kernel and run it again but this time we tried to do it **manually**. We manually tuned the parameters (Manual Tuning) and tried to run 1 iterative tunning, execution was run in 09:45 in 18 March 2022 and still couldn't finish until 12.51 in 18 March 2022 (3,5 hours) so we decided **not to use hyperparameter tuning** for this model.

(**Note**: Before these tuning, we also had already tried **GridSearchCV**.)

Feature Importance

Model 2



Limitation of The Model and Model Analysis

This model has its own limitation, this model can only use inside these criteria :

1. 5500<= market_value <=150.000.000
2. 0<= number_of_bedrooms <=93
3. 0<= number_of_rooms <=154
4. 0<= number_stories <=40
5. 0<= property_age <=368
6. 600<= total_area <=100.000
7. 600<= total_livable_area <=798.189

	Actual	Prediction	MAPE	condominium	apartment	house	total_livable_area	number_stories	zip_code	property_age	total_area
0	2068000.0	6.113153e+07	2856.069923	0	1	0	380040.0	10.0	19123	91.0	53335.00
1	1940500.0	2.584200e+07	1231.718732	0	0	0	275424.0	8.0	19148	80.0	52938.00
2	14238900.0	3.220690e+07	126.189544	0	1	0	112042.0	6.0	19104	27.0	90138.00
3	11486300.0	2.591496e+07	125.616256	0	1	0	159580.0	4.0	19122	8.0	56257.00
4	17242300.0	3.303370e+07	91.585244	0	1	0	184660.0	21.0	19102	92.0	9240.00
5	21715000.0	4.343969e+06	79.995538	0	1	0	36849.0	3.0	19103	15.0	53070.00
6	16142200.0	2.704288e+07	67.529067	0	0	0	301636.0	2.0	19132	51.0	99375.00
7	14774100.0	2.394456e+07	62.071185	0	1	0	133000.0	5.0	19104	125.0	61946.64
8	28359600.0	1.151721e+07	59.388687	0	0	0	69549.0	3.0	19103	52.0	61425.00
9	21183900.0	3.155274e+07	48.946778	0	0	0	184770.0	10.0	19104	75.0	20326.35
10	77911700.0	4.534030e+07	41.805527	0	1	0	254947.0	12.0	19121	13.0	75315.00
11	58919400.0	3.631262e+07	38.368996	0	1	0	245943.0	19.0	19147	40.0	30970.00
12	18837100.0	2.506951e+07	33.085841	0	0	0	146048.0	4.5	19107	32.0	51317.40
13	18575000.0	2.464730e+07	32.690703	0	1	0	132048.0	7.0	19104	95.0	44000.00
14	45056300.0	3.055134e+07	32.192967	0	1	0	180000.0	15.0	19103	120.0	10917.90
15	16776700.0	2.154927e+07	28.447633	0	1	0	82700.0	12.0	19103	95.0	7000.00
16	23037300.0	1.684588e+07	26.875623	0	1	0	64390.0	11.0	19106	114.0	9826.00
17	29215600.0	2.144660e+07	26.591958	0	1	0	114816.0	3.0	19104	43.0	76000.00
18	52272400.0	3.845760e+07	26.428477	0	1	0	225740.0	14.0	19107	122.0	19254.00
19	145580700.0	1.086178e+08	25.389958	0	0	0	723777.0	20.0	19107	47.0	50160.00
20	87075100.0	6.502379e+07	25.324473	0	0	0	500000.0	8.0	19103	39.0	69696.00
21	38879400.0	2.989729e+07	23.102491	0	1	0	168365.0	8.0	19103	106.0	29859.93
22	46893800.0	3.646397e+07	22.241390	0	0	0	172000.0	8.0	19104	20.0	24000.00
23	34288200.0	2.693498e+07	21.445334	0	1	0	140363.0	10.0	19107	95.0	13751.00
24	23315900.0	2.810080e+07	20.522064	0	1	0	173048.0	6.0	19106	118.0	38150.00
25	20504500.0	2.470767e+07	20.498783	0	0	0	145954.0	4.0	19106	125.0	19106.39
26	20751000.0	1.683097e+07	18.890805	0	1	0	101655.0	10.0	19107	119.0	8681.00

Not a good prediction data (MAPE above 20%)

Model 2 Analysis

Based on literature (*https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119199885.app1*), if the model has MAPE score **below 20%** it can classified as Good Model. So we will analyze Model 2 based on this threshold value.

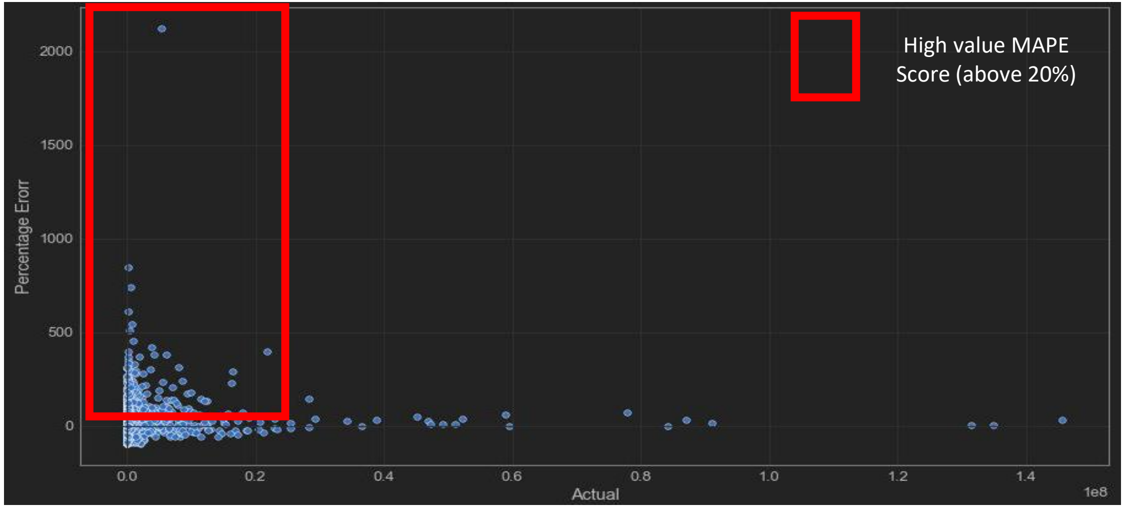
1. Confidence Area

Based on Regplot above, we analyze market_value above 20.000.000. Dataframe beside show us features from those outliers. There are 40 data test and only 14 from 40 data (35%) has MAPE score below 20% (MAPE score for good model), moreover there are very high MAPE score (2856%) it means the error is so high. Thus, we can summarize that our model can work better in property which has market_value below 20.000.000, we call this confidence area. Our model also can be use to predict market_value above 20.000.000 with lower confidence level, because there are 65% chance of our prediction can classified as not a good prediction (MAPE score above 20%).

Model 2 Analysis

2. General Error Analysis

We tried to analyze the market_value which have MAPE score above 20% (MAPE score for good model). There are 14446 data above MAPE score model from 96949 dataset test (14,9%).



From the Regplot, we can see that there are many data with MAPE values above 20%. Moreover, there are MAPE value above 2000%. This is very unwanted prediction result. We must analyze the characteristics of those properties with extremely high MAPE values.

```
D.groupby('category_code_description').mean()
```

executed in 29ms, finished 13:01:02 2022-03-25

de_description	Prediction	MAPE	market_value	total_livable_area	number_stories	number_of_rooms	zip_code	property_age	total_area
Commercial	1.276323e+06	80.236299	1.296604e+06	11778.582834	1.887226	4.842814	19130.935130	85.068862	11710.437255
Industrial	5.801218e+05	102.108488	3.334271e+05	11799.515924	1.538217	4.226115	19131.232484	86.028662	13383.907739
Mixed Use	2.412394e+05	46.283260	2.305266e+05	2705.057900	2.528950	6.112856	19134.323847	96.547596	1937.415967
Multi Family	8.112937e+05	69.478255	7.571190e+05	6889.005384	2.827950	11.563481	19131.054733	91.045312	4728.376927
Single Family	1.691683e+05	57.757278	1.581054e+05	1494.086125	2.237172	6.218450	19134.412711	93.130655	1916.362214
Vacant Land	1.346315e+05	180.482292	4.800000e+04	1280.000000	2.000000	0.000000	19153.000000	85.000000	2000.000000

1. As we can see in the data frame, Vacant Land should be an empty land without number stories, but there are value in those features, so this unwanted occurrence value cause MAPE mean score for this category has high value (180.482%).
2. Industrial category also has high MAPE mean score (102,10%), we must check features from this category.

Model 2 Analysis

3. Error Analysis based on category_code_description Industrial

A

```
E[E['category_code_description']=='Industrial'].loc[:,['Actual','Prediction','MAPE','total_livable_area','number_stories',
                    'number_of_rooms','zip_code','property_age','total_area','overall_condition','number_of_rooms',\
                    'category_code_description','apartment','house',
                    'condominium']].sort_values(by='total_livable_area', ascending=False)
```

executed in 339ms, finished 15:32:48 2022-03-24

	Actual	Prediction	MAPE	total_livable_area	number_stories	number_of_rooms	zip_code	property_age	total_area	overall_condition
478409	1940500.0	2.584200e+07	1231.718732	275424.0	8.0	2.0	19148	80.0	52938.00	4.0
477133	2686200.0	1.050265e+07	290.985407	172742.0	2.0	7.0	19122	70.0	60855.00	5.0
478538	1183900.0	3.969918e+06	235.325450	98046.0	5.0	4.0	19134	115.0	32036.55	4.0
477715	899100.0	1.457746e+06	62.133912	89330.0	3.0	4.0	19132	100.0	33110.00	5.0
477580	1604700.0	1.499459e+06	6.558298	79162.0	2.0	4.0	19142	80.0	80494.00	4.0
478112	1388500.0	3.283721e+06	136.494130	72611.0	5.0	4.0	19125	145.0	36104.00	4.0
476602	6014800.0	4.757215e+06	20.908176	72000.0	3.0	2.0	19127	2.0	41356.00	1.0
476715	754800.0	7.861400e+05	4.152093	67964.0	3.0	4.0	19124	90.0	50594.00	5.0
477466	543900.0	9.185860e+05	68.888766	67560.0	3.0	6.0	19134	90.0	23342.28	4.0
476688	805500.0	1.314902e+06	63.240472	64751.0	2.0	4.0	19124	121.0	77230.00	7.0
476797	938600.0	9.791020e+05	4.315150	62132.0	3.0	6.0	19137	85.0	25522.00	4.0
478456	498100.0	9.040980e+05	81.509335	61872.0	3.0	4.0	19134	85.0	22500.00	4.0

As we can see in the Dataframe A (dataset test), dataframe which grouped by categorical code Industrial, the highest livable_total_area has MAPE score 1231,71% and many of the top 12 highest livable_total_area has MAPE score above 20% (MAPE score for good model) (9 of 12 top data). We analyze that such high MAPE was caused by the actual market_value was set too low from the majority of data with the same specifications. This could be external factor that model cannot predict (of course it's called outliers).

B

```
E[E['category_code_description']=='Industrial'].loc[:,['Actual','Prediction','MAPE','total_livable_area','number_stories',
                    'number_of_rooms','zip_code','property_age','total_area','overall_condition','number_of_rooms',\
                    'category_code_description','apartment','house',
                    'condominium']].sort_values(by='Actual', ascending=False)
```

executed in 354ms, finished 15:33:48 2022-03-24

	Actual	Prediction	MAPE	total_livable_area	number_stories	number_of_rooms	zip_code	property_age	total_area	overall_condition
476602	6014800.0	4.757215e+06	20.908176	72000.0	3.0	2.0	19127	2.0	41356.00	1.0
480913	3417600.0	4.933880e+06	44.366807	35742.0	7.0	7.0	19107	90.0	5106.00	3.0
477257	2997800.0	1.552669e+06	48.206385	20000.0	1.0	4.0	19145	60.0	89178.33	4.0
477133	2686200.0	1.050265e+07	290.985407	172742.0	2.0	7.0	19122	70.0	60855.00	5.0
476585	2562600.0	3.009213e+06	17.428120	30000.0	2.0	2.0	19140	90.0	34386.00	4.0
478761	2367300.0	2.386916e+06	0.828623	23520.0	1.0	4.0	19123	60.0	43200.00	4.0
478624	2157900.0	1.543500e+06	28.472126	13400.0	3.0	4.0	19147	80.0	48125.00	4.0
479044	2041300.0	4.785991e+06	134.457992	41363.0	2.0	4.0	19104	80.0	68547.51	3.0
477118	2020100.0	2.743937e+06	35.831741	12800.0	2.0	4.0	19103	220.0	6400.00	4.0
478409	1940500.0	2.584200e+07	1231.718732	275424.0	8.0	2.0	19148	80.0	52938.00	4.0
477137	1900000.0	4.056040e+05	78.652421	18463.0	1.0	4.0	19129	120.0	69498.75	6.0
477915	1792500.0	1.869465e+06	4.293724	30187.0	1.0	7.0	19146	90.0	59677.00	4.0

As we can see in the Dataframe B (dataset test), dataframe grouped by categorical code Industrial, many of the top 12 highest market_value has MAPE score above 20% (MAPE score for good model) (10 of 12 top data) and some exceed 100%. We analyze that the result was caused by actual market_value was set too low from the majority of data. This could be external factor that model cannot predict (of course it's called outliers).

Model 2 Analysis

4. Error Analysis Based on External Factor

As we can see in the stacked lineplot beside, there are two era where count of sale property in Philadelphia drastically down. It happen with worldwide crisis, like economy crisis in 2006-2010 and covid pandemic in 2020, we assume this crisis can affect our market_value prediction.

From dataframe below, we can see almost of market_value prediction exceed actual market_value.

A

```
E[(E['sale_year']==2020) & (E['MAPE']>20)].loc[:,['Actual','Prediction','MAPE','total_livable_area','number_stories',
                                                   'number_of_rooms','zip_code','property_age','total_area','overall_condition','number_of_rooms',
                                                   'category_code_description','sale_year','apartment','house',
                                                   'condominium']].sort_values(by='MAPE', ascending=False)
```

executed in 318ms, finished 13:08:54 2022-03-25

	Actual	Prediction	MAPE	total_livable_area	number_stories	number_of_rooms	zip_code	property_age	total_area	overall_condition
228236	14100.0	1.527260e+05	983.163121	1472.0	1.0	5.5	19129	81.0	1810.50	4.0
62181	78300.0	4.559710e+05	482.338442	8520.0	3.0	7.0	19144	60.0	36618.17	7.0
60142	38500.0	2.107600e+05	447.428571	5152.0	3.0	7.0	19144	25.0	5390.40	7.0
207219	86000.0	4.342430e+05	404.933721	2022.0	2.0	7.0	19146	95.0	1664.48	2.0
453528	453200.0	2.023707e+06	346.537290	10168.0	2.0	7.0	19104	158.0	8100.00	4.0
80904	17200.0	6.775800e+04	293.941860	2120.0	3.0	6.0	19132	100.0	2707.14	7.0
482237	188800.0	7.180410e+05	280.318326	6172.0	3.0	12.0	19119	95.0	10261.89	3.0
217873	20100.0	7.050200e+04	250.756219	1652.0	2.0	6.0	19122	95.0	1095.00	5.0
206303	63400.0	2.219880e+05	250.138801	956.0	2.0	6.0	19146	95.0	752.50	4.0
470891	86900.0	2.938470e+05	238.143843	1336.0	2.0	5.0	19147	100.0	1046.56	4.0

B

```
F=E[(E['sale_year']==2006) | (E['sale_year']==2007) | (E['sale_year']==2008) | (E['sale_year']==2009) |
   (E['sale_year']==2010)].loc[:,['Actual','Prediction',
                                  'MAPE','total_livable_area','number_stories','number_of_rooms','zip_code','property_age',
                                  'total_area','overall_condition','number_of_rooms','category_code_description','sale_year',
                                  'apartment','house','condominium']].sort_values(by='MAPE', ascending=False)
```

executed in 45ms, finished 13:01:17 2022-03-25

```
F[F['MAPE']>20]
```

executed in 26ms, finished 13:01:22 2022-03-25

	Actual	Prediction	MAPE	total_livable_area	number_stories	number_of_rooms	zip_code	property_age	total_area	overall_condition
214800	12500.0	197725.000000	1481.800000	1924.0	3.0	4.0	19133	100.0	1418.25	2.0
483373	23600.0	319667.000000	1254.521186	1454.0	3.0	6.0	19145	13.0	1368.00	3.0
69454	8400.0	87136.000000	937.333333	612.0	1.0	5.5	19140	70.0	946.80	4.0
481340	63000.0	598486.000000	849.977778	4152.0	3.0	12.0	19104	10.0	6043.00	1.0
37019	11700.0	99774.333333	752.772080	759.0	1.0	5.5	19140	55.0	759.00	4.0
...	...	...	...	...	...	...	...	...	...	...
475772	330400.0	264174.000000	20.044189	2290.0	1.0	3.0	19149	65.0	2794.00	4.0
450072	137200.0	164673.000000	20.024052	3014.0	2.0	6.0	19143	95.0	2356.00	4.0
165267	204300.0	163395.000000	20.022026	1208.0	2.0	6.0	19125	145.0	1186.40	4.0
105466	120000.0	144025.000000	20.020833	1455.0	3.0	7.0	19119	120.0	1231.83	4.0
382117	220900.0	265101.000000	20.009507	1605.0	2.0	7.0	19116	58.0	5230.51	4.0

1917 rows × 16 columns

From dataframe A and B we use selection indexing to filter sale_year where worldwide crisis happen (economy crisis and covid pandemic) with threshold of MAPE score above 20%. We can get 380 and 1917 data in it, it means 15,9% of the data above good MAPE model score (dataset test) affected by those external crisis.

Model Conclusion and Recommendation

Conclusion

- **Background and problem statement:**
  1. With this model, we can predict the price without any of aprraisal professional, thus reducing the cost.
  2. There are differences between the price and average market value with the same criteria, we expect this model can predict market value with exact value (not overvalue or undervalue). With a justified market value, hopefully there will be increase in success transaction.

- **Model conclusion:**
  1. There are 24 features (8 numerical and 16 categorical) and 484.058 rows data for modeling purposes.
  2. Based on Cross Validation we choose **Random Forest Regressor model**, because it has the lowest MAPE score (14.47% in Model 1 and 13.32% in Model 2) and the most stable (lowest standard deviation).
  3. From comparison between Model 1 and Model 2 we **choose Model 2** over Model 1 because extra feature engineering can boost test score (dataset test) from MAPE score 13,18% in Model 1 to **12,25% in Model 2**. This extra feature engineering also boost others metric evaluation score.
  4. `**total_livable_area**` is the most importance feature in Model 2 and then followed by `number_stories`, `zip_codes`, `property_age`, `sale_year` ,`total_area`, and `overall_condition`, respectively.

- Our group can make model to predict market value property with **MAPE score (dataset test) around 12,12%** which categorized as **good model prediction.** This model **can answer the problem statement**, so the property agent has an option **not to use professional appraisal to asses market value of property** in Philadelphia, just use this model instead to **reduce operational cost.** But there are some limitation for this model, such as:
  1. 5500<= market_value <=150.000.000
  2. 0<= number_of_bedrooms <=93
  3. 0<= number_of_rooms <=154
  4. 0<= number_stories <=40
  5. 0<= property_age <=368
  6. 600<= total_area <=100.000
  7. 600<= total_livable_area <=798.189
  8. This model work better for predict market value lower than 20.000.000, but we can predict market value above 20.000.000 with lower confidence level (35% chance to get a good model result).

Model Conclusion and Recommendation

Recommendation

From the model result, this model still have opportunities to improve. To do so, we need to:

1. Read more literature to know more about the domain knowledge, this will reduce the assumption with fact.
2. Doing another extra feature engineering by exploring more about current features (deepen the feature analysis and take a look the relation between feature-feature and feature-label).
3. Fix the value in features, so value from other value can match logically with another.
4. Doing several combination change like extracting, simplify and re-categorize to get more model and get better result.
5. Doing normalization to the feature, this can make us have more model option and get better result.
6. Improvement by doing hyperparameter tuning.
7. Drop some features with low importance feature score.
8. Gather another data to increase confidence level while predict high market value properties
9. We must aware to the data which has strong affected by external factor (maybe try to generate one feature to distinguish data which affected by external factor or not).
10. Need for another extra data/feature, like data for every property sold in Philadelphia in 2020 until 2022.