Real Estate Data Analysis

Prepared and submitted by,

Kumpati, Paul Sastry

Baltha, Dheeraj Varma

Kumbham, Shashidhar Reddy

Manda, Rajesh Kumar

Introduction

The performance of real estate varies greatly depending on the region. Due to differing macroeconomic conditions, different countries might differ. Local variables such as economic activity or supply might cause cities within the same nation to differ. Some areas or sub-sectors (for example, luxury condominiums vs. mass market homes) might perform significantly differently within a metropolis.

Cluster analysis reveals patterns in data, allowing you to see which groupings of attributes are more likely to behave similarly and which are more likely to behave differently.

Cluster analysis may also be used to identify time periods in which property market performance is more or less comparable. Government action has a significant impact on many real estate markets. Many substantial changes in legislation may occur, affecting price and investing behavior.

- **Descriptive analytics -** This type of analysis explains what happened over a certain time period (real estate trends). How much has rental income risen in the last five years, for example? In the previous year, what was the vacancy rate? Descriptive analytics examines data from a variety of sources in order to obtain meaningful historical insights.
- **Diagnostic analytics -** This type of analysis looks at previous data to figure out why something happened. Diagnostic analysis, for example, helps elucidate why rental properties have such a high turnover rate.
- Predictive analytics - Predictive real estate data analytics, as the name implies, predictions what will happen in the future. It makes a home market prediction utilizing data from diagnostic and descriptive analytics to detect clusters, exceptions, and patterns.An investor, for example, can utilize real estate predictive analytics to anticipate how much rent will rise in the following two years.
- Prescriptive analytics - The major goal of this type of analysis is to prescribe what needs to be done in the future to take advantage of a chance or avoid a problem. Consider someone who wishes to invest in an Airbnb property. Should they buy a house with two, three, or four bedrooms? How many bathrooms do you think it should have? Prescriptive Airbnb analytics will assist determine what has to be done to increase occupancy rates and ensure future cash flow. This type of study, on the other hand, makes use of current technology and real estate

investment instruments such as algorithms, business rules, and machine learning.

Background and Related Work

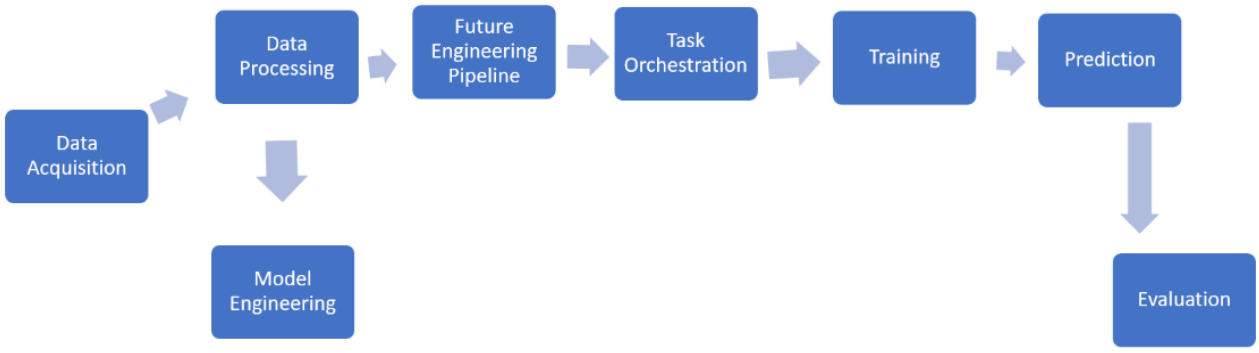
Maintaining a solid handle on altering values in the ever-changing Commercial and Residential Real Estate markets can be difficult, but the cost of a knowledge gap can be significant. A lack of market understanding may lead to unwise investments or money left on the table for individuals and companies with big Real Estate holdings. To close these gaps, a complete Real Estate Market Analysis employing innovative proptech technologies is required.

The majority of Real Estate Market Analysis is learning about the real- world circumstances surrounding the property in question's geographic location. Here are the top five new geographical data streams you should be incorporating as soon as you begin your analysis:

- **POI information**
- **Demographics of Human Mobility and Traffic**
- **Financial Information**
- **Data about housing and parcels**

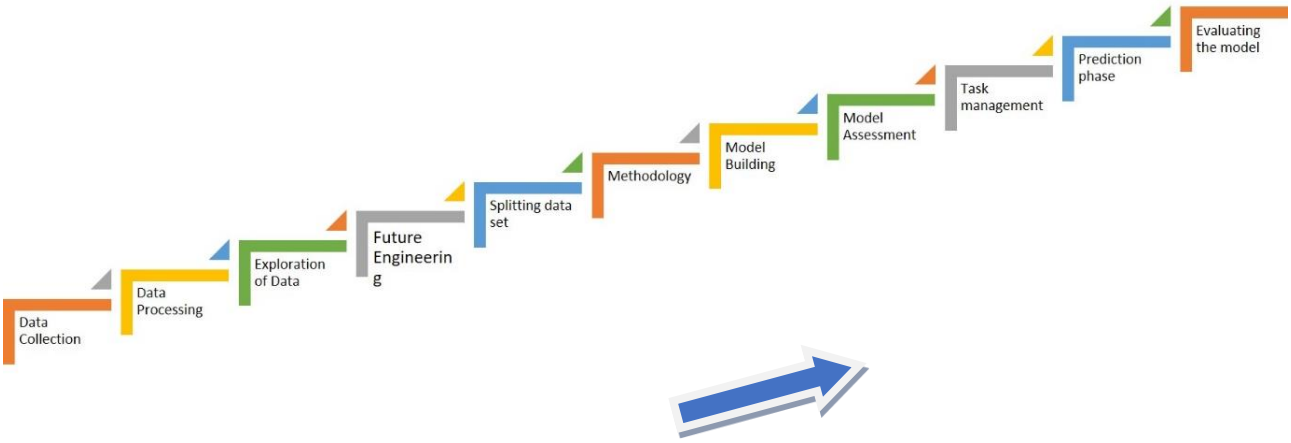
Last but not least, any comprehensive Real Estate Market Analysis should make use of current housing and parcel data to provide a full picture of property prices. This information is frequently made public, but it may be obtained in a variety of places, in different forms, and at different spatial scales. Finding extensive and dependable data sources, such as Tinsa's Real Estate data or Landgrid's national parcel data, becomes a priority. And having a resource like CARTO's Data Observatory on hand may make incorporating these datasets (as well as datasets from all of the aforementioned categories) into your Real Estate Market Analysis much easier.

Architecture Diagram



The dataset is collected from the kaggle website. Then we process the data by cleaning and tranforming such as data cleaning and data handling. Then we choose the model and train the dataset in that particular model. And pipelines are created for that model and we set tasks orchestration. Then after we train the model with our desired algorithms and we predict the values of test datast and evaluation will be done after running the model successfully.

Workflow diagram



The workflow diagram can be depicted as shown in figure. First we collect the Real estate data from kaggle website and we process it using hadoop map reduce and we explore the datest and plan for future engineering. We clean and transform the dataset into null free values and then split the dataset into training set and test set. We choose the methodology of what models are appropriate to use on this dataset and we build the model on training dataset. We then assess the model and perform task management and in prediction phase the model will predict the values of test dataset and evaluting will done.

Dataset

Detailed Description of Dataset

For real estate investors, we will give high-quality property information. Get property information based on particular criteria such as the URL for the property, the location, the property type, the price, and the image URL.

Data on Real Estate (Commercial, Residential, and Industrial): Number of Bedrooms Location Address Bathrooms Square Footage Number of Bathrooms Name of the Owner Coordinates of the Floorspace Description Facilities Price per ft/m2 POI Data from property sellers and data from real estate agencies
Our team collects, processes, and validates property data for additional analysis, market research, and competition analysis. We offer a speedy return time and real facts that may be enhanced by study. Fields Include: - Current Ownership (include mailing addresses) - Physical Property Address - Last Sale & Mortgage Data - Assessment Value, Fiscal Year and Tax Amounts - Property Usage - Additional 35 Property and Building Characteristics. Data available for Marketing Lists/Promotions, Bulk Licensing, Website, Portals & API's and Analytics. When it came to looking for a home to buy, real estate websites were a valuable source of information. Not only are websites a significant source of information, but smartphone and tablet search engines are increasingly becoming popular ways for consumers, particularly young people, to locate real estate. The primary parcel file contains information on the original owner and the land. Add a file providing drawing vectors for the records of dwellings; Any extra addresses for a shipment are stored in the Additional Address file.

Assessment file contains data about the evaluated value; Appraisal file containing data about appraised worth; Commercial Apt contains commercial apartment data; Commercial File contains primary commercial

data; Commercial File Data on commercial interiors and exteriors Entrance data from assessor visits; Dwelling fileEntrance data from appraiser visits: Other Structures and Yard Enhancements File for Sales Tax Rates File, which contains tax charges and payments for the current billing cycle by taxing district authority and property class; and Tax Payments File, which contains tax charges and payments for the current billing cycle by taxing district authority and property class.

Detailed Design of Features with Diagrams

Our real estate data model is divided into three categories:

- Locations and estates
- Contacts and clients
- Contracts and business dealings

Employee is the only table that isn't in any of the subject areas.

Please keep in mind that the employee and estate tables in the Clients and contacts subject area, as well as the client table in the Contracts and transactions subject area, are only copies to make the model easier to understand. We'll start with the personnel table, then move on to Estates and locations, then Clients and contacts, and finally Contracts and transactions.

The employee table will be our first stop. It's straightforward: each employee's first name and last name are stored. Other information might be included, such as the employee's tax ID number, birth date, residence, work title, and so on. However, because we won't be focused on the employees in this model, all we'll need is a means to link people to activities (like being assigned to a task or contract). This table will also allow us to keep track of which employees were involved in each client interaction.

The client table offers information on all of the clients with whom we've ever worked, both present and future. What is the definition of a potential client? It might be someone who has expressed an interest in buying, selling, or renting property from us in the future. We need to keep track of such clients' contact information and properties for future reference.

Because they are not critical, the final five properties in this table are nullable. We'll almost certainly need to save information for at least one contact person, but we may not know who that person will be until later.

The contact table is the second and last table in this topic area. We'll keep track of everything we've done with clients here. We'll utilize this data to improve our future operations — for example, if a client requests that we rent a certain estate when it becomes

available, we'll keep track of that request and notify them when the estate is ready.

Analysis of Data

Data Preprocessing

Data Pretreatment: To provide highly accurate and insightful results, data preprocessing is a common stage in machine learning. The more the dependence on the provided outcomes, the higher the data quality. Large real -world datasets are characterized by incomplete, noisy, and inconsistent data. By filling in missing or incomplete data, reducing noise, and addressing inconsistencies, data preprocessing improves data quality.

There are a variety of reasons why data is incomplete. Customer information for sales transaction data is an example of an attribute that may not always be available. Due to a misunderstanding or device faults, relevant data may not be recorded.Noisy data can be caused by a variety of factors (having incorrect attribute values). It's possible that the data gathering equipment were defective. At the time of data entry, there might have been human or computer mistakes. Data transfer errors can also occur. Inconsistencies in naming conventions or data codes utilized, as well as

inconsistent formats for input fields such as date, might result in incorrect data.

A variety of data preparation methods are available, including:

- Cleaning of data
- Integration and transformation of data
- Reduction of data

Data Cleaning and Handling Outliers

In [5]:
```
dataset = dataset.drop(['Price_Flag','X','Y','CMPLX_NUM','FULLADDRESS','LONGITUDE','CITY','STATE','NATIONALGRID','CENSUS_BLOCK','

dataset.GBA = dataset.GBA.fillna(dataset.GBA.mean())

dataset.AYB = dataset.AYB.fillna(dataset.AYB.median())

dataset.STORIES = dataset.STORIES.fillna(dataset.STORIES.median())

dataset.KITCHENS = dataset.KITCHENS.fillna(dataset.KITCHENS.median())

dataset.NUM_UNITS = dataset.NUM_UNITS.fillna(dataset.NUM_UNITS.median())

dataset.YR_RMDL = dataset.YR_RMDL.fillna(dataset.YR_RMDL.median())

dataset.LIVING_GBA = dataset.LIVING_GBA.fillna(dataset.LIVING_GBA.mean())

dataset.STYLE = dataset.STYLE.fillna(dataset.STYLE.mode()[0])

dataset.STRUCT = dataset.STRUCT.fillna(dataset.STRUCT.mode()[0])

dataset.GRADE = dataset.GRADE.fillna(dataset.GRADE.mode()[0])

dataset.CNDTN = dataset.CNDTN.fillna(dataset.CNDTN.mode()[0])

dataset.EXTWALL = dataset.EXTWALL.fillna(dataset.EXTWALL.mode()[0])

dataset.ROOF = dataset.ROOF.fillna(dataset.ROOF.mode()[0])

dataset.INTWALL = dataset.INTWALL.fillna(dataset.INTWALL.mode()[0])

dataset.ASSESSMENT_SUBNBHD  = dataset.ASSESSMENT_SUBNBHD.fillna(dataset.ASSESSMENT_SUBNBHD.mode()[0])

dataset.isnull().sum()
```

Out[5]:
```
Unnamed: 0          0
BATHRM              0
HF_BATHRM           0
HEAT                0
```

```
AC                    0
NUM_UNITS             0
ROOMS                 0
BEDRM                 0
AYB                   0
YR_RMDL               0
EYB                   0
STORIES               0
PRICE                 0
QUALIFIED             0
SALE_NUM              0
GBA                   0
BLDG_NUM              0
STYLE                 0
STRUCT                0
GRADE                 0
CNDTN                 0
EXTWALL               0
ROOF                  0
INTWALL               0
KITCHENS              0
FIREPLACES            0
USECODE               0
LANDAREA              0
GIS_LAST_MOD_DTTM     0
SOURCE                0
LIVING_GBA            0
ZIPCODE               0
LATITUDE              0
ASSESSMENT_NBHD       0
ASSESSMENT_SUBNBHD    0
CENSUS_TRACT          0
WARD                  0
SQUARE                0
dtype: int64
```

The data preprocessing of the real estate dataset is done successfully and all null values and incomplete columns are removed.

Filling in missing values, reducing noise, resolving discrepancies, and finding and deleting outliers in data are all examples of data cleaning.

Data integration is the process of combining data from several sources into a logical data storage, such as a data warehouse.

Normalization and other data modifications are possible. Normalization, for example, might boost the accuracy and efficiency of mining algorithms that use distance measurements.

Data reduction can be accomplished by removing redundant characteristics or clustering, for example.

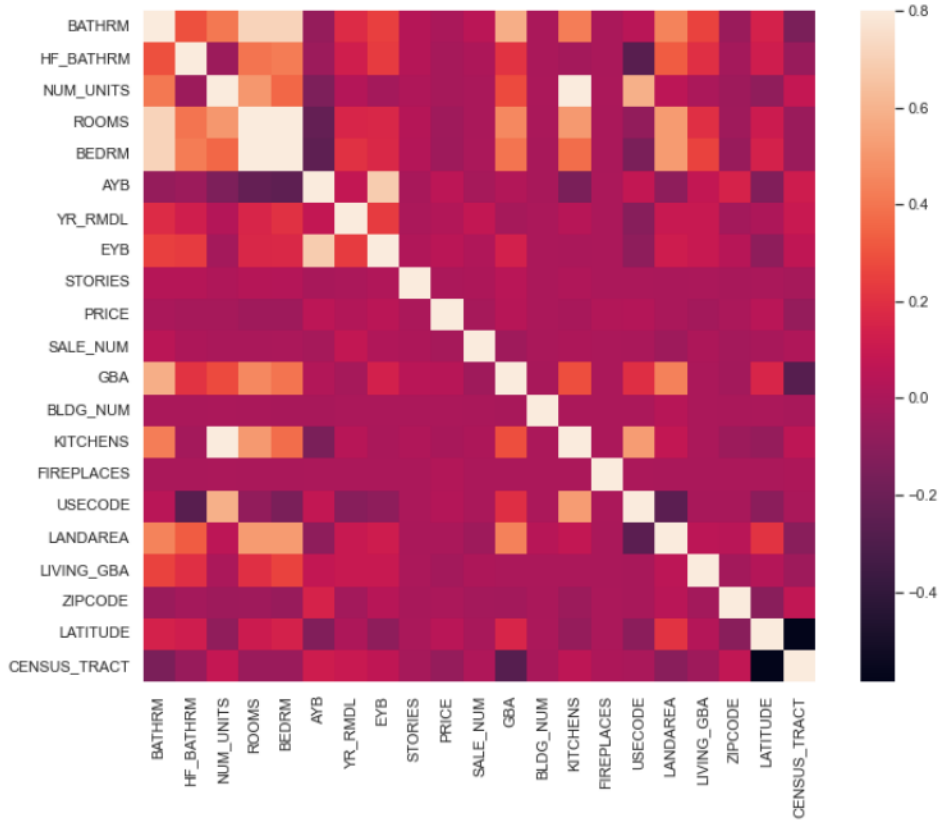
Graph Model with Explanation

The following concept map highlights the most essential ideas in the context of the property graph:

With 37 features we cant say which feature is related to house prices so we find the correlation of price with every other variable.
We plot a heat map as follows by specifying the subplots and size.
The graph can be as follows:

In [7]:

```
corrmat = train_dataset.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
```



We find 10 independent features that contribute to house price.

Price, AYB, EYB, longitude, latitude, GBA, usecode, fireplaces YR_RMDL, BLDG_NUM, STORIES and ZIPCODE these are 10 independent values.

GBA - Gross building area in square feet

AYB - The earliest time the main portion of the building was built

EYB - The year an improvement was built more recent than actual year built

Latitude - The location of the propert

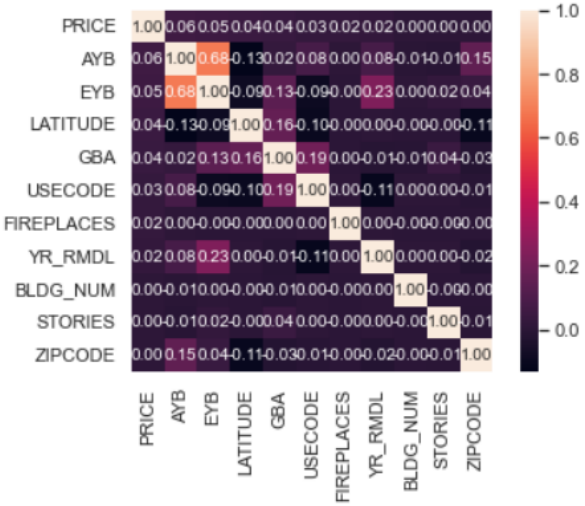
Kitchens - Number of kitchens

Stories - Number of stories in primary dwelling

Yr_Model - Year structure was remodeled

In [8]:

```
k = 11 #number of variables for heatmap
cols = corrmat.nlargest(k, 'PRICE')['PRICE'].index
cm = np.corrcoef(train_dataset[cols].values.T)
sns.set(font_scale=1.00)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10},
plt.show()
```

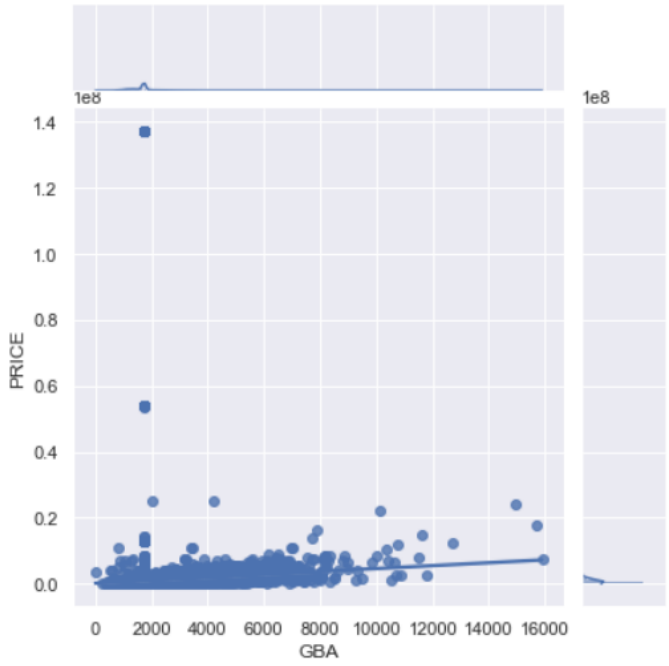


Now we plot the graph between Gross building area vs price

```
# Gross Building Area vs  Price

sns.jointplot(x=train_dataset['GBA'], y=train_dataset['PRICE'], kind='reg')
```

Out[10]:

```
<seaborn.axisgrid.JointGrid at 0x211b1f4f5b0>
```

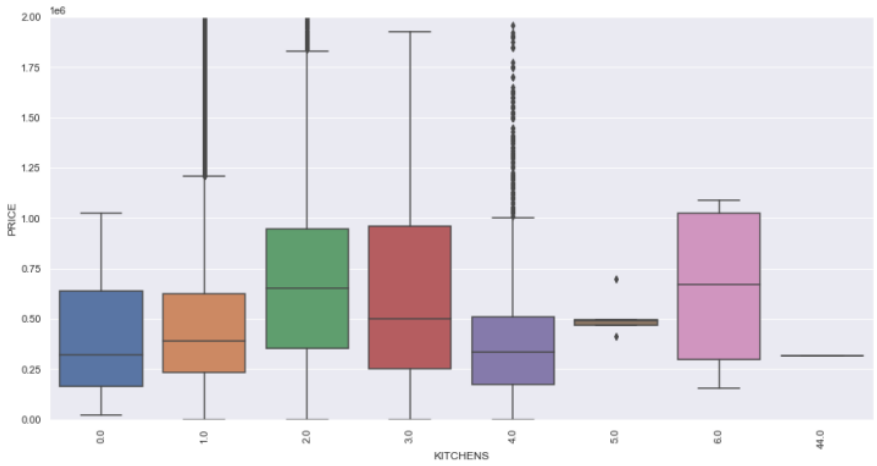


By this we can conclude that people pay more living area.

Now we plot the graph between kitchen and price.

In [12]:

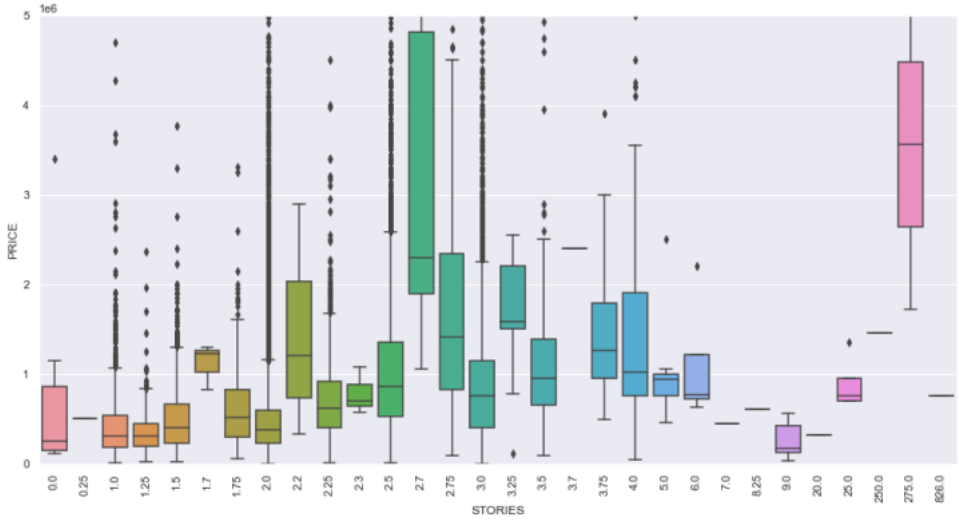
```
# Kitchens vs  Price

var = 'KITCHENS'
data = pd.concat([train_dataset['PRICE'], train_dataset[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="PRICE", data=data)
fig.axis(ymin=0, ymax=2000000);
plt.xticks(rotation=90);
```



Now we plot the graph between stories and price

In [13]:

```
# Stories vs  Price

var = 'STORIES'
data = pd.concat([train_dataset['PRICE'], train_dataset[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="PRICE", data=data)
fig.axis(ymin=0, ymax=5000000);
plt.xticks(rotation=90);
```



Nodes and connections, like ideas and their interactions in the diagram above, can (should) have "names" (officially termed labels for nodes and types for relationships).

The arrowheads represent the direction of the relationships.

Properties, which are "key / value" pairs such as Colour: Red, can be associated with both Nodes and Relationships. The key is referred to as "Property Name" in the data model. Property graphs are similar to idea maps in that they don't have a standard format (like in UML, e.g.). The author's preferred style is shown above, but you are welcome to use your own. You've done your job if you're able to communicate effectively with your audience.

The labelled property graph model is the most versatile data model paradigm currently available.

The names and the structure are the most significant aspects (the nodes and the relationships). By contributing substance, the qualities enhance the solution structure. Properties are essentially simply names, but they may also denote “identity."

Implementation

- **Algorithms**

We have used Decision tree regression, weighted KNN, support vector machine regression, ridge regression, random forest regression, and gradient boosting trees.

1. Decision tree regression

   In this algorithm, the model builds in the form of tree like structure. It Analyses the dataset and splits into subsets with an associated decision tree. It is a supervised machine learning algorithm; the important node can be considered as the root node and the least important nodes are considered as child nodes. The decision root node and child nodes have 2 decision nodes. based on the data set we create the decision tree by giving the maximum depth of the tree while training the real estate dataset. The decision tree trains the real estate dataset into tree like structure and then we find the root mean square of the algorithm.

2. Weighted KNN regression

   It is similar to K-nearest neighbors algorithm, we use weighted KNN instead of KNN because the hyper parameter is too sensitive in KNN, if it is small it can be sensitive and if its too large

it may include many points from other classes. We divide the class labels based on our dataset and we give the key value pairs (x,y) we create training set of observations x with class y. Here x is the new observation where y is class label to be predicted.

We analyse the distance between the points in training set. Then we group the class points and predict the points using distance weighted voting. The root mean square error will be evaluated.

3. Random Forest algorithm

   The Random Forest classifier is an ensemble learning technique that uses many classifiers to solve a problem and improve the model's performance. On diverse subsets, numerous decision trees are created, and an average is found for effective performance. Increasing the number of trees in the model can improve its accuracy, but only to a limited extent, depending on the dataset. We get numerous decision trees, some of which correctly anticipate the values and some of which do not. However, by integrating all of the variables, the values can be precisely predicted. We design decision trees using data points from the

training set. We take into account all of the decision trees and take the average of them all.

4. Gradient boosting trees

   Gradient boosting algorithm is used to predict continuous values such as the cost, weight. Here We predict the house prices of Washington DC. This regression will differentiate current prediction and the target value. This difference can be considered as residue. This residue will be included in present input so that we get correct target. Repeating this step again improves the model efficiency.

- **Explanation of implementation with results**

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)


import os
print(os.listdir("C:"))

import seaborn as sns
import matplotlib.pyplot as plt

from scipy import stats
from scipy.stats.stats import pearsonr
from scipy.stats import norm

from scipy.stats import skew

sns.set(style='white', context='notebook', palette='deep')



# Any results you write to the current directory are saved as output.
```

```
['.ipynb_checkpoints', 'Code Rajesh.ipynb', 'CrimePrediction.ipynb', 'd-c-residential-properties-analysis.ipynb', 'dc-housing-Rajesh.ipynb', 'DC_Properties.csv', 'My Music', 'My Pictures', 'My Videos']
```

Here we import the numpy, pandas, os, seaborn, matplot and scipy libraries. Numpy is used for mathematical functions, linear algebra function. Pandas is used for data preprocessing and visualizing. Seaborn is also for data

visualization(statistical graphs). Matplotlib is a library for creating interactive visualizations.

Importing the dataset

Importing the dataset

In [2]:
```
dataset=pd.read_csv("DC_Properties.csv")

dataset.head()
```

```
C:\Users\91703\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3444: DtypeWarning: Columns (18,19,20,21,22,23,24,3
3,34,35,37,43,45) have mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[2]:

	Unnamed: 0	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	...	LONGITUDE	ASSESSMENT_NBHD	ASSESSMENT_SUBI
0	0	4	0	Warm Cool	Y	2.0	8	4	1910.0	1988.0	...	-77.040832	Old City 2	040 D Old
1	1	3	1	Warm Cool	Y	2.0	11	5	1898.0	2007.0	...	-77.040764	Old City 2	040 D Old
2	2	3	1	Hot Water Rad	Y	2.0	9	5	1910.0	2009.0	...	-77.040678	Old City 2	040 D Old
3	3	3	1	Hot Water Rad	Y	2.0	8	5	1900.0	2003.0	...	-77.040629	Old City 2	040 D Old
4	4	2	1	Warm Cool	Y	1.0	11	3	1913.0	2012.0	...	-77.039361	Old City 2	040 D Old

5 rows × 49 columns

As shown in above fig we import the dataset. And display the head of the dataset and the dataset head is displayed.

We create dummy variable and store the dataset and perform the operations on dummy dataset.

a. Split the data into 3 parts – training_data, test_data and unknown_data

In [3]:
```
#dataset.isnull().sum()

dummy_dataset = dataset

dummy_dataset['Price_Flag'] = np.where(dummy_dataset.PRICE > 0 , 1,0)

unknown_dataset = dummy_dataset[dummy_dataset.Price_Flag != 1]

unknown_dataset.shape
```

Out[3]: (60741, 50)

In [4]:
```
from sklearn.model_selection import train_test_split

dataset = dummy_dataset[dummy_dataset.Price_Flag != 0]
```

As the dataset consists of manu null values, so we clean the data that is we preprocess the data as follows

Data Cleaning and Handling Outliers

In [5]:
```
dataset = dataset.drop(['Price_Flag','X','Y','CMPLX_NUM','FULLADDRESS','LONGITUDE','CITY','STATE','NATIONALGRID','CENSUS_BLOCK','

dataset.GBA = dataset.GBA.fillna(dataset.GBA.mean())

dataset.AYB = dataset.AYB.fillna(dataset.AYB.median())

dataset.STORIES = dataset.STORIES.fillna(dataset.STORIES.median())

dataset.KITCHENS = dataset.KITCHENS.fillna(dataset.KITCHENS.median())

dataset.NUM_UNITS = dataset.NUM_UNITS.fillna(dataset.NUM_UNITS.median())

dataset.YR_RMDL = dataset.YR_RMDL.fillna(dataset.YR_RMDL.median())

dataset.LIVING_GBA = dataset.LIVING_GBA.fillna(dataset.LIVING_GBA.mean())

dataset.STYLE = dataset.STYLE.fillna(dataset.STYLE.mode()[0])

dataset.STRUCT = dataset.STRUCT.fillna(dataset.STRUCT.mode()[0])

dataset.GRADE = dataset.GRADE.fillna(dataset.GRADE.mode()[0])

dataset.CNDTN = dataset.CNDTN.fillna(dataset.CNDTN.mode()[0])

dataset.EXTWALL = dataset.EXTWALL.fillna(dataset.EXTWALL.mode()[0])

dataset.ROOF = dataset.ROOF.fillna(dataset.ROOF.mode()[0])

dataset.INTWALL = dataset.INTWALL.fillna(dataset.INTWALL.mode()[0])

dataset.ASSESSMENT_SUBNBHD  = dataset.ASSESSMENT_SUBNBHD.fillna(dataset.ASSESSMENT_SUBNBHD.mode()[0])

dataset.isnull().sum()
```

Out[5]:
```
Unnamed: 0            0
BATHRM                0
HF_BATHRM             0
HEAT                  0
AC                    0
NUM_UNITS             0
ROOMS                 0
BEDRM                 0
AYB                   0
YR_RMDL               0
EYB                   0
STORIES               0
PRICE                 0
QUALIFIED             0
SALE_NUM              0
GBA                   0
BLDG_NUM              0
STYLE                 0
STRUCT                0
GRADE                 0
CNDTN                 0
EXTWALL               0
ROOF                  0
INTWALL               0
KITCHENS              0
FIREPLACES            0
USECODE               0
LANDAREA              0
GIS_LAST_MOD_DTTM     0
SOURCE                0
LIVING_GBA            0
ZIPCODE               0
LATITUDE              0
ASSESSMENT_NBHD       0
ASSESSMENT_SUBNBHD    0
CENSUS_TRACT          0
WARD                  0
SQUARE                0
dtype: int64
```

The preprocessing data has been completed successfully.

Now we split the dataset into training set and test set.

Splitting the dataset to Test and Train after Cleaning and Data Handling

In [6]:
```
train_dataset = dataset

train_dataset = train_dataset.drop('Unnamed: 0',axis =1)

cat = len(train_dataset.select_dtypes(include=['object']).columns)
num = len(train_dataset.select_dtypes(include=['int64','float64']).columns)
print('Total Features: ', cat, 'categorical', '+',
      num, 'numerical', '=', cat+num, 'features')
```

```
Total Features:  16 categorical + 21 numerical = 37 features
```

We the dataset consists of 16 categorical attributes and 21 numnerical attributes.

Now we perform ordinary leat squares on training set.

We label the categorical data as follows:

Labelling the Categorical Data

In [14]:
```
from sklearn.preprocessing import LabelEncoder
cols = train_dataset.select_dtypes(include=['object']).columns
# Process columns and apply LabelEncoder to categorical features
for c in cols:
    lbl = LabelEncoder()
    lbl.fit(list(train_dataset[c].values))
    train_dataset[c] = lbl.transform(list(train_dataset[c].values))

train_dataset.head()
```

Out[14]:

	BATHRM	HF_BATHRM	HEAT	AC	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	...	GIS_LAST_MOD_DTTM	SOURCE	LIVING_GBA	ZIPCODE	L
0	4	0	12	2	2.0	8	4	1910.0	1988.0	1972	...	1	1	882.090907	20009.0	3
2	3	1	7	2	2.0	9	5	1910.0	2009.0	1984	...	1	1	882.090907	20009.0	3
3	3	1	7	2	2.0	8	5	1900.0	2003.0	1984	...	1	1	882.090907	20009.0	3
5	3	2	7	2	1.0	10	5	1913.0	2005.0	1972	...	1	1	882.090907	20009.0	3
7	3	1	7	2	2.0	8	4	1906.0	2011.0	1972	...	1	1	882.090907	20009.0	3

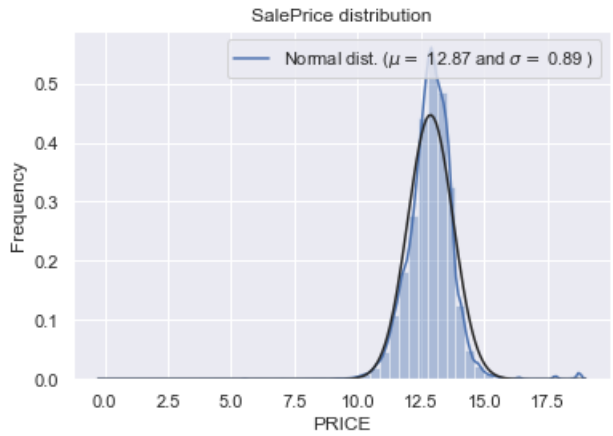
5 rows × 37 columns

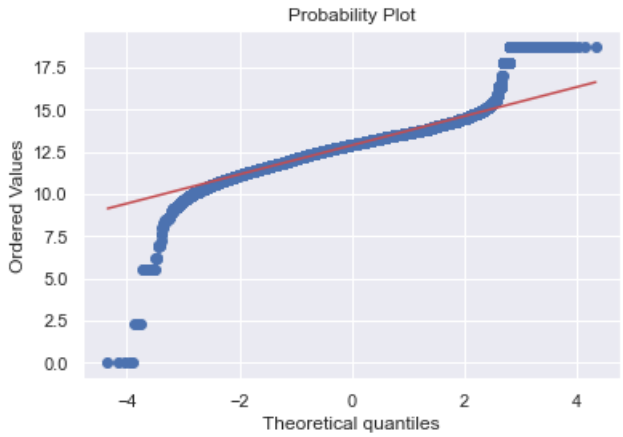
Then we find the skewness of the data.

Finding the Skewness of the Data

In [15]:
```
# We use the numpy fuction log which  applies log to all elements of the column
train_dataset["PRICE"] = np.log(train_dataset["PRICE"])

#Check the new distribution
sns.distplot(train_dataset['PRICE'] , fit=norm);

# Get the fitted parameters used by the function
(mu, sigma) = norm.fit(train_dataset['PRICE'])
print( '\n mu = {:.2f} and sigma = {:.2f}\n'.format(mu, sigma))
plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)],
           loc='best')
plt.ylabel('Frequency')
plt.title('SalePrice distribution')

fig = plt.figure()
res = stats.probplot(train_dataset['PRICE'], plot=plt)
plt.show()

y_train = train_dataset.PRICE.values

print("Skewness: %f" % train_dataset['PRICE'].skew())
print("Kurtosis: %f" % train_dataset['PRICE'].kurt())
```

```
mu = 12.87 and sigma = 0.89
```






```
Skewness: 0.300212
Kurtosis: 9.506771
```

Saleprice distribution graph is plotted and probability plot is also plotted. Skenwss(measure of symmetry)of this data is 0.300 and kurtosis(measure of data heavy tailed or light tailed relative to normal distribution) is 9.50

```
In [16]: train_dataset = (train_dataset - train_dataset.mean()) / (train_dataset.max() - train_dataset.min())

In [17]: train_dataset = pd.get_dummies(train_dataset)
         print(train_dataset.shape)

         (98216, 37)

In [18]: train_dataset, test_dataset = train_test_split(train_dataset, test_size=0.2)

         train_dataset.shape

Out[18]: (78572, 37)
```

Training dataset is splitted here.

Running OLS on the Train Dataset

In [19]:
```
import statsmodels.api as sm

from sklearn.metrics import r2_score

from sklearn.metrics import mean_squared_error

train_dataset_Y = train_dataset.PRICE.values

train_dataset_X = train_dataset.drop('PRICE',axis =1)

train_dataset_X.shape

train_dataset_X = sm.add_constant(train_dataset_X)

Pricing_model = sm.OLS(train_dataset_Y,train_dataset_X)

result = Pricing_model.fit()

print(result.summary())

print("RMSE: ",np.sqrt(mean_squared_error(result.fittedvalues,train_dataset_Y)))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.441
Model:                            OLS   Adj. R-squared:                  0.441
Method:                 Least Squares   F-statistic:                     1774.
Date:                Sun, 01 May 2022   Prob (F-statistic):               0.00
Time:                        18:53:36   Log-Likelihood:            1.5031e+05
No. Observations:               78572   AIC:                         -3.005e+05
Df Residuals:                   78536   BIC:                         -3.002e+05
Df Model:                          35
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       1.433e-05      0.000      0.112      0.911      -0.000       0.000
BATHRM         0.0991      0.003     34.311      0.000       0.093       0.105
HF_BATHRM      0.0487      0.003     15.884      0.000       0.043       0.055
HEAT          -0.0024      0.001     -3.395      0.001      -0.004      -0.001
AC             0.0193      0.001     24.567      0.000       0.018       0.021
NUM_UNITS     -0.0352      0.005     -7.812      0.000      -0.044      -0.026
ROOMS         -0.0104      0.004     -2.822      0.005      -0.018      -0.003
BEDRM         -0.0139      0.004     -3.411      0.001      -0.022      -0.006
AYB           -0.0689      0.002    -36.917      0.000      -0.073      -0.065
YR_RMDL        0.5195      0.021     24.583      0.000       0.478       0.561
EYB            0.0884      0.002     44.764      0.000       0.085       0.092
STORIES        0.0299      0.031      0.956      0.339      -0.031       0.091
QUALIFIED     -0.0067      0.000    -21.467      0.000      -0.007      -0.006
SALE_NUM       0.0847      0.001     64.808      0.000       0.082       0.087
GBA            0.1468      0.006     25.270      0.000       0.135       0.158
BLDG_NUM       0.0836      0.011      7.735      0.000       0.062       0.105
STYLE          0.0010      0.002      0.481      0.631      -0.003       0.005
STRUCT         0.0067      0.001      4.993      0.000       0.004       0.009
GRADE          0.0138      0.001     24.603      0.000       0.013       0.015
CNDTN          0.0151      0.001     28.718      0.000       0.014       0.016
EXTWALL       -0.0053      0.001     -7.253      0.000      -0.007      -0.004
ROOF           0.0058      0.001      9.084      0.000       0.005       0.007
INTWALL       -0.0021      0.001     -2.422      0.015      -0.004      -0.000
KITCHENS       0.0514      0.026      1.974      0.048       0.000       0.102
FIREPLACES     0.2677      0.036      7.491      0.000       0.198       0.338
USECODE        0.0135      0.007      2.047      0.041       0.001       0.026
LANDAREA       0.0140      0.014      0.992      0.321      -0.014       0.042
GIS_LAST_MOD_DTTM -0.0001  0.000     -0.387      0.698      -0.001       0.001
SOURCE        -0.0001      0.000     -0.387      0.698      -0.001       0.001
LIVING_GBA     0.2664      0.005     53.748      0.000       0.257       0.276
ZIPCODE       -0.0653      0.003    -20.047      0.000      -0.072      -0.059
LATITUDE      -0.0063      0.001     -5.789      0.000      -0.008      -0.004
ASSESSMENT_NBHD -0.0156    0.001    -21.933      0.000      -0.017      -0.014
ASSESSMENT_SUBNBHD 0.0175  0.001     23.289      0.000       0.016       0.019
CENSUS_TRACT  -0.0025      0.001     -3.791      0.000      -0.004      -0.001
WARD          -0.0270      0.001    -39.212      0.000      -0.028      -0.026
SQUARE        -0.0167      0.001    -26.256      0.000      -0.018      -0.015
==============================================================================
Omnibus:                    37642.837   Durbin-Watson:                   2.002
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          3117775.201
Skew:                           1.421   Prob(JB):                         0.00
Kurtosis:                      33.729   Cond. No.                     1.00e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 7.84e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
RMSE:  0.03572360276595213
```

The root mean sqaure error for training set of OLS is 0.0357 which very ideal for a model like this.

In [20]:
```
test_dataset_Y = test_dataset.PRICE.values

test_dataset = test_dataset.drop('PRICE',axis=1)

predictions = result.predict(sm.add_constant(test_dataset))

RMSE = np.sqrt(mean_squared_error(predictions,test_dataset_Y))

print('The RMSE of the predicted values is ',RMSE)
```

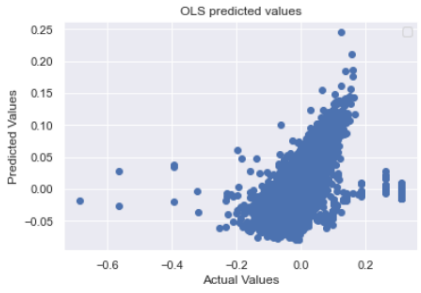
```
The RMSE of the predicted values is  0.03507673491128878
```

The root mean square error for test set of OLS is 0.03507.

Now we plot the graph between predicted values and actual values.

In [21]:
```
plt.scatter(test_dataset_Y, predictions)

plt.legend()
plt.title('OLS predicted values')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.show()
```

```
No handles with labels found to put in legend.
```



Decision Tree Regression model

Decision Tree Regression

In [25]:
```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(train_dataset_X,train_dataset_Y)

# Predicting a new result
y_pred = regressor.predict(test_dataset)

rmse = np.sqrt(mean_squared_error(y_pred, test_dataset_Y))

print("The Root mean square error of Decision Tree Regression is ", rmse)

print("The R2 value of Decision Tree Regression is ",r2_score(test_dataset_Y,y_pred))
```

```
The Root mean square error of Decision Tree Regression is  0.03214374292602364
The R2 value of Decision Tree Regression is  0.5324897652205696
```

we import the decision tree regressor and train the model with decision tree and predict the values on test datset. The root mean square error for decision tree is 0.0321 and r-square is 0.5324 these two values indicates how well the model predicts the values in terms of absolute and percentage respectively.

Weighted KNN regression

Weighted KNN for Regression

In [26]:
```
from sklearn import neighbors


knn = neighbors.KNeighborsRegressor(5)

pred_test = knn.fit(train_dataset_X,train_dataset_Y).predict(test_dataset)

RMSE = np.sqrt(mean_squared_error(test_dataset_Y, pred_test))

print("The Root Mean Squared Error of KNN Regression is ",RMSE)

print("The R2 value of KNN Regression is ",r2_score(test_dataset_Y,pred_test))
```

```
The Root Mean Squared Error of KNN Regression is  0.027224861837831914
The R2 value of KNN Regression is  0.6646258386571677
```

We imported neighbors regressor and building a model with training dataset. Here we specify the number of neighbors as 5. And then we find the RMSE value for it. RMSE for wirghted KNN is 0.0272 and R-square value is 0.664 the RMSE value is best when compared to that of Decision tree.

Random forest regression

Random Forest Regression

In [28]:
```
from sklearn.ensemble import RandomForestRegressor
# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators = 25)
# Train the model on training data
pred = rf.fit(train_dataset_X,train_dataset_Y).predict(test_dataset)

RMSE_1 = np.sqrt(mean_squared_error(test_dataset_Y, pred))

print("The Root Mean Squared Error of Random Forest Regression is ",RMSE_1)

print("The R2 value of Random Forest Regression is ",r2_score(test_dataset_Y,pred))
```

```
The Root Mean Squared Error of Random Forest Regression is  0.02404442917048371
The R2 value of Random Forest Regression is  0.7384063648229537
```

Here we import the Random Forest regessor and we build the model by specifying the number of trees as 25. The RMSE value for random forest regression is 0.0240 and R-sqaure value is 0.7384. The RMSE value is ideal when comapred to that of weighted KNN regression.

Gradient Boosting Trees

Gradient Boosting Trees

In [29]:
```
from sklearn.ensemble import GradientBoostingRegressor

pred = GradientBoostingRegressor(n_estimators=100, learning_rate=0.3,max_depth=1, random_state=0, loss='ls').fit(train_dataset_X,

RMSE_1 = np.sqrt(mean_squared_error(test_dataset_Y, pred))

print("The Root Mean Squared Error of Gradient Boosting Regression is ",RMSE_1)

print("The R2 value of Gradient Boosting Regression is ",r2_score(test_dataset_Y,pred))
```

```
The Root Mean Squared Error of Gradient Boosting Regression is  0.032652635146770144
The R2 value of Gradient Boosting Regression is  0.5175695607957937
```

We import the gradient boosting regression and we give the no.of trees as 100 and learning rate as 0.3 max depth 1 to build this model. RMSE value is 0.0326 and R-square is 0.5175

After evaluating all models we find the accuracy of random forest regression is very efficient than any other algorithm.

Project Management

- **Implementation Status Work**

Work Completed

Description

The Real estate property of DC Dataset is collected from online resources that is Kaggle. Then the dataset is pre-processed(i.e cleaned and transformed). Now the dataset is free of null values and incomplete values. Now we find the skewness and Kurtosis of the dataset. We find the important features of the dataset so that we can predict the sale prices of house in DC. We plotted 5 graphs and founds there are 10 important features that can define the house price. Now we split the dataset into 2 parts. One is training dataset which consists of 80% data and rest 20% is test dataset. We build the model using algorithms and then predict the values of test dataset so that we can know how much the model we built is efficient. We find Root mean square error to know this. We built 5 algorithms that are Ordinary least square, decision tree, Weighted KNN, Random Forest and gradient boosting algorithm. We trained the respective models with training dataset and predicted the test dataset

values and found the RMSE values of each algorithm. Then we found that Random Forest algorithm is most efficient for this dataset.

Responsibilities

Paul Sastry Kumpati - Dealt with preprocessing of data and found the skewness and kurtosis of the data and plotted 2 graphs.

Documentation:

Introduction

background

Manda, Rajesh Kumar - Responsible for developing the models Random Forest, Decision tree and weighted KNN algorithms.

Documentation:

Architecture and work flow diagram

implementation

Kumbham, Shashidhar Reddy - Worked with finding the important features of the dataset and plotted 3 graphs

Documentation:

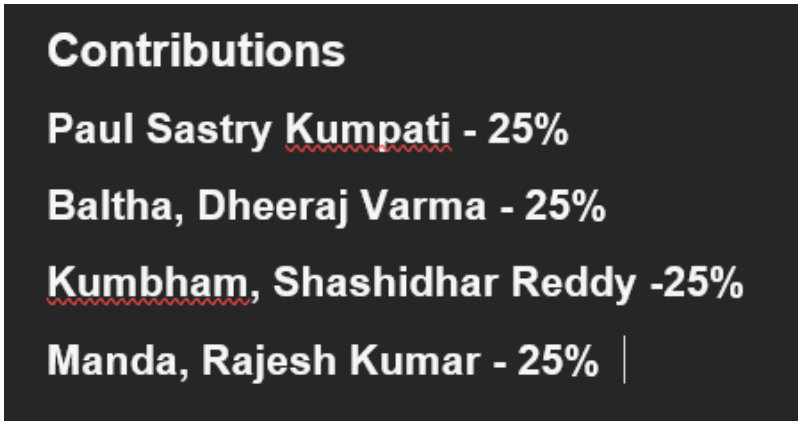
Dataset

Analysis of data.

Baltha, Dheeraj Varma - Worked on evaluating OLS and Gradient boosting regression model.

Documentation:

Results and project management.



Issues/Concerns

The quality of real estate data available on the internet is fraught with flaws and inconsistencies. In this post, we'll use examples to highlight some of the concerns and discuss some of the obstacles that come with real estate data management — gathering, cleaning, and standardizing data. We will concentrate focus on the US market, while some of the difficulties may apply to other nations as well.

• Sources of Information

In the United States, there are several data sources for real estate statistics. You have well-known web sites like Zillow.com, Trulia.com, Redfin.com, and Realtor.com, which began with "Properties for Sale" and now include "Properties for Rent."

Apartments.com, ForRent.com, Zumper.com, Apartmentlist.com, Hotpads.com, and other specialist "Properties for Rent" sources are also available. While the majority of these websites are national in scope, there are a few of specialty local sources as well.

- **Location or Address**

The location, or address, is one of the most important data pieces for a property. Real estate is a real asset that requires a permanent location (even for mobile homes, they do have to be placed on a land with an address). Most users wish to use the address as a "unique key" to match data between different databases. Because properties can't exist in space (yet), they must have a physical address on the planet.

- **Errors in data input**

On any of the lines, there are simple spelling problems.

13 Mapel St versus 13 Maple St - Wrong Street Names

1234 Main St (which does not exist) instead of 123 Main St are incorrect numerals.
Bolton vs. Boston is the wrong city name (yes both are in MA)
02134 versus 01234 is the incorrect zip code.

Any of these errors are difficult to spot or utilize when comparing data sets.

- **Address the issue of normalization.**

When you collect real estate data from one or more sources, one of the first tasks is to be able to identify and compare two properties.
To check whether a listing for sale or rent is a duplicate listing or a unique listing in one website, the data must be cleaned and normalized to a standard format, and then the data must be compared to see whether it is the same listing across different websites.

References:

https://www.kaggle.com/datasets/christophercorrea/dc-residential-properties
https://www.sciencedirect.com/science/article/pii/S1303070119300538
https://www.ukessays.com/essays/economics/literature-review-on-the-determination-of-housing-price-economics-essay.php
https://econwpa.ub.uni-muenchen.de/econ-wp/urb/papers/0408/0408001.pdf