

# House Price Prediction

**Prepared by: Ahmed Aly Hassan Mahmoud**

**ID: 192006**

Following CRISP-DM (Cross Industry Standard Process for Data Mining) life cycle

# 1. Introduction:

The last decade has witnessed recurrent inconsistency in the U.S. housing market due to economic fluctuations and market changes. In the United States, the annual sales of existing single-family homes from 2005 to 2008 fell by 30%, while their average price dropped about 26%, and new construction declined by a 64% (Lombra, 2012).

Investment is a business activity that most people are interested in this globalization era. There are several objects that are often used for investment, for example, gold, stocks and property. In particular, property investment has increased significantly since 2011, both on demand and property selling

Investment in Real state is still the safest way as many people consider. As House prices increase every year, there is a need for a system to predict house prices in the future. House price prediction can help buyer, seller and broker to determine the selling price of a house and can help the customer to arrange the right time to purchase a house. There are many factors that influence the price of a house which include physical conditions, concept and location.

We need a proper prediction on the real estate and the houses in housing market we can see a mechanism that runs throughout the properties buying and selling. Buying a house will be a life time goal for most of the individual but there are lot of people making huge mistakes in United States of America right now when buying the properties. Most of the people are buying properties unseen from the people they don't know by seeing the advertisements and all over the grooves coming around the America one of the common mistakes is buying the properties that are too expensive but its' not worth it. In the housing market 2017, there is a survey that in the year 2016 the house sold in the America were about 5.42 million but the starter home inventory down up to 10.7% from 2015.

This research aims to predict house prices in the city of Pittsburgh (**Pennsylvania**) with regression analysis. Selection of affect variables and regression analysis is used to determine the optimal coefficient in prediction.

## 2. Business Understanding

### 2.1 Pervious work

A variety of research has been developed to model housing prices and property values. First developed in the 60's, hedonic regression has been the most common approach because it allows the total housing expenditure to be decomposed into the values of the individual components. Hedonic modeling is a widespread method in which a property is assumed to be a heterogeneous good and can be broken down into characteristics such as structural features (e.g. year built, square footage, etc.) and locational factors (e.g. distance to commercial areas and major streets, etc.). The relationship between sale prices and housing characteristics along with neighborhood properties is measured by the hedonic regression (Rosen, 1974).

While the hedonic method is widely acceptable for accommodating real estate attributes to make predictions, one disadvantage is its complexity. Issues such as variable interactions, heteroscedasticity (non-constant variance), multi-collinearity, non-linearity and outlier data points can hinder the performance of the hedonic price model in housing prices. The second disadvantage is that hedonic models are limited within a specific studied location. Thus, hedonic models are generally used to gain insight into one particular market and it is difficult to generalize across different geographic regions (Sirmans et al., 2005).

## 2.2 Currently

The topic of real estate is not only the topic you just have to deal with. It can also be very interesting. There are plenty of TV Shows, for instance, [\*Property Brothers\*](#), of which plot is based on examples of people buying and renovating houses. This particular one is the most famous in the world and has been running already for almost a decade. For many people houses are also an investment that generates profits.

The descent in home values raises questions among home owners and shoppers regarding how accurately the value of homes can be assessed and what attributes determine the desirability of certain homes compared to other houses on the market.

####8-KomagometowneAnh\_Thesis2016

This instability in the housing market necessitates the need for better methods for assessing housing prices. Consequently, the predictive accuracy of housing models has gained much attention among scholars and has been extensively studied. In particular, we are interested in methods that can estimate the value of a home based on its attributes in comparison to the current market price of similar houses. This ability to predict housing prices is important to anyone buying or selling a home, as well as to investors making asset allocation decisions.

Regardless of motives of buying and selling real estate, both sides agree on a price. It is always good to know **how much** a house is worth, what is the expected transaction price. Furthermore, it may be even more important **why** the price is like that, what has an impact on it.

Let us start with a couple of questions that allow to define and understand problems regarding house pricing.

- The seller does not know how to increase the value of the apartment so that the investment outlay is lower than the added value (e.g. building a pool may increase the price and renovating the bathroom is not worth it).
- The seller does not know how much to sell the apartment for (he makes an offer on the portal and does not know if the price is adequate).

- The buyer does not know how much the apartment is worth (as above, whether the price is adequate).
- Commercial problem: auction services may be interested in tools to support sellers and buyers (to highlight the sections in the offers that most affect the price).

An accurate prediction on the house price is important to prospective homeowners, developers, investors, appraisers, tax assessors and other real estate market participants, such as, mortgage lenders and insurers (Frew and Jud, 2003). Traditional house price prediction is based on cost and sale price comparison lacking of an accepted standard and a certification process. Therefore, the availability of a house price prediction model helps fill up an important information gap and improve the efficiency of the real estate market (Calhoun, 2003).

Over the years, the problem of property evaluation was solved in many different ways. Statistical tools used by analysts to explain house prices range from simple linear regression to more complex techniques such as artificial neural networks.

### 3. Data Understanding

The crucial element in machine learning task for which a particular attention should be clearly taken is the data. Indeed the results will be highly influenced by the data based on where did we find them, how are they formatted, are they consistent, is there any outlier and so on. At this step, many questions should be answered in order to guarantee that the learning algorithm will be efficient and accurate.

#### 3.1 Data source:

This dataset contains data on all Real Property parcels that have sold since 2013 in Allegheny County, PA.

Allegheny County (Pennsylvania) and the City of Pittsburgh both publish their data through the Western Pennsylvania Regional Data Center. The Western Pennsylvania Regional Data Center supports key community initiatives by making public information easier to find and use. The Data Center maintains Allegheny County and the City of Pittsburgh's open data portal, and provides a number of services to data publishers and users. The Data Center also hosts datasets from these and other public sector agencies, academic institutions, and non-profit organizations. The Data Center is managed by the University of Pittsburgh's Center for Social and Urban Research, and is a partnership of the University, Allegheny County and the City of Pittsburgh.

<https://catalog.data.gov/dataset/allegheny-county-property-sale-transactions>

Download link:

<https://catalog.data.gov/dataset/allegHENY-county-property-sale-transactions/resource/5a6688a2-d307-44a7-9bac-82af2f2b0509>

### 3.2 Description of variables in the dataset

Data contains 24 house features plus the price and the id columns, along with 307K observations. Below the Table describes the variables in the data set

Field Name	Field Description	Example	Additional Data Details
PARID	Parcel Identification Number	0030A00190000000	A 16 character unique identifier for the parcel.
PROPERTYHOUSENUM	Property Location House Number	3016	House Number for the physical location of property; may be zero or blank.
PROPERTYFRACTION	Property Location House Number Fraction	43832	Continued House Number information for the physical location of property; such as 1/2, the house number ending range, or a letter.
PROPERTYADDRESSDIR	Property Location Street Directional	(e.g. N, S, etc....)	Directional indicator of the street name for the physical location of the property.
PROPERTYADDRESSSTREET	Property Location Street Name	HARCUM	Street name for the physical location of the property.
PROPERTYADDRESSSUF	Property Location Street Type	WAY	Suffix of the street name for the physical location of the property.
PROPERTYADDRESSUNITDESC	Property Location Unit Description	(e.g. UNIT, APT, STE, BLDG, etc...)	Unit description for the physical location of the property.
PROPERTYUNITNO	Property Location Unit Number	(e.g. 1, 3C, 115A, B13, etc....)	Unit number for the physical location of the property.
PROPERTYCITY	Property Location City Name	PITTSBURGH	City where the property is physically located.
PROPERTYSTATE	Property Location State Name	PA	State where the property is physically located.
PROPERTYZIP	Property Location Zip code, first 5 digits	15203	Zip Code where the property is physically located.
SCHOOLCODE	School District Code	47	School district number associated with specified parcel.

SCHOOLDESC	School District Name	City Of Pittsburgh	School district name associated with specified parcel.
MUNICODE	Municipality Code (Tax District)	116	Municipality number associated with specified parcel.
MUNIDESC	Municipality Name	16th Ward - PITTSBURGH	Municipality name associated with specified parcel.
RECORDDATE	Date Sale was Recorded	41969	Date the sale was recorded in the Department of Real Estate. Format is mm/dd/yyyy.
SALEDATE	Sale Date	41947	Date the sale occurred. Format is mm/dd/yyyy.
PRICE	Sale Price	35000	Amount paid for the sale (also called consideration).
DEEDBOOK	Deed Book Number	15811	Cross-reference to the physical book number where the deed is on file in the Department of Real Estate.
DEEDPAGE	Page Number	248	Page within the specified deed book corresponding to the first page of this physical deed.
SALECODE	Sale Validation Code	AA	A subjective categorization code denoting whether or not the sale price was representative of current market value. These categorizations are subject to change as sales are reviewed by the Office of Property Assessments or during a reassessment. See further explanation in the "Sale Validation Codes Details" resource.
SALEDESC	Sale Validation Code Description	SALE NOT ANALYZED	A description of the code denoting whether or not the sale price was representative of current market value. These categorizations are subject to change as sales are reviewed by the Office of Property Assessments or during a reassessment. See further explanation in the "Sale Validation Codes Details" resource.
INSTRTYP	Instrument Type	SD	A code describing the type of document used to record the real property transfer.
INSTRTYPDESC	Instrument Description	Sheriffs Deed	A description of the type of document used to record the real property transfer.

## 4. Data Preparation

### 4.1 Reading the Dataset

#### Mongodb installation and dataload

##### Read from Database source (Mongodb)

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. MongoDB is a document database, which means it stores data in JSON-like documents. We believe this is the most natural way to think about data, and is much more expressive and powerful than the traditional row/column model.

##### Read Dataset from Database Source

```
In [6]: import pymongo
from pymongo import MongoClient

In [7]: client = MongoClient('localhost',27017)

In [8]: client.list_database_names()
Out[8]: ['House_Price_DB', 'admin', 'config', 'local']

In [9]: #select database
db = client['House_Price_DB']
#select the collection within the database
test = db.House_Price_Project
#convert entire collection to Pandas dataframe
data = pd.DataFrame(list(test.find()))

In [10]: data.head()
Out[10]:
```

	_id	PARID	PROPERTYHOUSENUM	PROPERTYFRACTION	PROPERTYADDRESSDIR	PROPERTYADDRESSSTREET	PROPERTYADI
0	60316b0e77fc7f0448dd9e07	1075F00108000000	4720			HIGHPOINT	
1	60316b0e77fc7f0448dd9e08	0011A00237000000	0			LOMBARD	

##### Read Dataset as CSV file in Pandas

As a second option we can read the dataset from the csv file we downloaded. We will use the `read_csv()` function from Pandas Python package:

```
import pandas as pd
import numpy as np
data = pd.read_csv('House_Price.csv')
```

### 4.2 Getting A Feel of the Dataset

Let's display the first few rows of the dataset to get a feel of it:

**data.head()**

	PARID	PROPERTYHOUSENUM	PROPERTYFRACTION	PROPERTYADDRESSDIR	PROPERTYADDRESSSTREET	PROPERTYADDRESSSUF	PROPERTYADDRE
0	1075F00108000000	4720.0		NaN	HIGHPOINT	DR	
1	0011A00237000000	0.0		NaN	LOMBARD	ST	
2	0011J00047000000	1903.0		NaN	FORBES	AVE	
3	0113B00029000000	479.0		NaN	ROOSEVELT	AVE	

Now, let's get statistical information about the numeric columns in our dataset. We want to know the mean, the standard deviation, the minimum, the maximum, and the 50th percentile (the median) for each numeric column in the dataset:

```
In [8]: data.describe(include=[np.number], percentiles=[.5]) \
        .transpose().drop("count", axis=1)
```

Out[8]:

	mean	std	min	50%	max
PROPERTYHOUSENUM	1440.010444	2063.976108	0.0	603.0	39389.0
PROPERTYZIP	15169.627437	92.813989	15003.0	15204.0	16229.0
SCHOOLCODE	28.999058	15.068535	1.0	29.0	50.0
MUNICODE	688.607075	340.616427	101.0	874.0	953.0
PRICE	172554.424233	973742.816640	0.0	76000.0	148752900.0

From the table above, we can see, for example, that the average Amount paid for the sale in our dataset is 172554 \$ with a standard deviation of 973742 .Similarly, we can get a lot of information about our dataset variables from the table.

Then, we move to see statistical information about the non-numerical columns in our dataset:



```
In [9]: data.describe(include=[np.object]).transpose() \
        .drop("count", axis=1)
```

Out[9]:

	unique	top	freq
PARID	219493	0431B00017000000	20
PROPERTYFRACTION	1530		303363
PROPERTYADDRESSDIR	4	N	3505
PROPERTYADDRESSSTREET	9123	WASHINGTON	1709
PROPERTYADDRESSSUF	48	ST	78459
PROPERTYADDRESSUNITDESC	10	UNIT	5497
PROPERTYUNITNO	1248	2	90
PROPERTYCITY	98	PITTSBURGH	166630
PROPERTYSTATE	1	PA	307756
SCHOOLDESC	46	Pittsburgh	76332
MUNIDESC	175	Penn Hills	11103
RECORDDATE	2398	2012-10-26	594
SALEDATE	3042	2016-04-29	597
DEEDBOOK	3633	TR18	1284
DEEDPAGE	2431	1	1806
SALECODE	47	3	60015
SALEDESC	28	LOVE AND AFFECTION SALE	60015
INSTRYP	29	DE	176104
INSTRYPDESC	29	DEED	176104

In the table we got, count represents the number of non-null values in each column, unique represents the number of unique values, top represents the most frequent element, and freq represents the frequency of the most frequent element.

## 4.3 Data Cleaning

### 4.3.1 Dealing with Missing Values

We should deal with the problem of missing values because some machine learning models don't accept data with missing values. Firstly, let's see the number of missing values in our dataset. We want to see the number and the percentage of missing values for each column that actually contains missing values.

### Getting the number of missing values in each column

```
In [10]: num_missing = data.isna().sum()
```

### Excluding columns that contains 0 missing values

```
In [11]: num_missing = num_missing[num_missing > 0]
```

### Getting the percentages of missing values

```
In [12]: percent_missing = num_missing * 100 / data.shape[0]
```

Out[15]:

	Missing Values	Percentage
PROPERTYADDRESSUNITDESC	301795	98.063076
PROPERTYUNITNO	301616	98.004913
PROPERTYADDRESSDIR	295134	95.898699
PRICE	2478	0.805183
PROPERTYADDRESSSUF	1359	0.441584
RECORDDATE	1226	0.398368
DEEDPAGE	327	0.106253
DEEDBOOK	317	0.103004
PROPERTYHOUSENUM	18	0.005849
PROPERTYADDRESSSTREET	8	0.002599
PROPERTYCITY	1	0.000325
PROPERTYZIP	1	0.000325

#### 4.3.2 Imputation

Missing value imputation is one of the biggest challenges encountered by the data scientist. In addition, most machine learning algorithms are not powerful enough to handle missing data. Missing data can lead to ambiguity, misleading conclusions, and results [27]. There are two types of missing values [28]; the first type is called missing completely at random (MCAR). MCAR can be expressed as:

$$(R|X,,)=P(R|\mu)$$

Where  $R$  is the response indicator variables,  $X$  are independent of data variables, and  $Z$  is latent. The second type is called missing at random (MAR), which can be expressed as:

$$(R=r | X=x, Z=z, \mu) = (R=r | X_0=x_0)$$

There are two methods of handling missing data, namely ignoring missing data and imputation of missing data. Ignoring missing data is a simple technique which deletes the cases that contain missing data. The disadvantages of this method are that it reduces the size of the dataset, and it uses a different sample size for different variables. Imputation of missing data is a technique that replaces missing data with some reasonable data values [27]. However, the imputation of missing data method has two types, single imputation, and multiple imputations. Single imputation contains several approaches, such as mean imputation and regression imputation. Mean imputation is the most common approach of missing data replacement [27]. It replaces the missing data with sample mean or median. However, it has a disadvantage which is if missing data are enormous in number, then all those data are replaced with the same imputation mean, which leads to change in the shape of the distribution. Regression imputation is a technique based on the assumption of the linear relationship between the attributes. The advantage of regression imputation over mean imputation is that it was able to preserve the distribution shape [27].

### 4.3.3 Deleting Unimportant Columns

According to the dataset dictionary hereunder some columns that can be removed which doesn't have a crucial impact on the house pricing issue itself.

- 1- **PARID:** A 16 character unique identifier for the parcel.
- 2- **PROPERTYHOUSENUM:** House Number for the physical location of property; may be zero or blank.
- 3- **PROPERTYFRACTION:** Continued House Number information for the physical location of property; such as 1/2, the house number ending range, or a letter.
- 4- **PROPERTYADDRESSDIR:** Directional indicator of the street name for the physical location of the property.
- 5- **PROPERTYADDRESSSUF:** Suffix of the street name for the physical location of the property.
- 6- **PROPERTYADDRESSUNITDESC:** Unit description for the physical location of the property.
- 7- **PROPERTYUNITNO:** Unit number for the physical location of the property.
- 8- **PROPERTYSTATE:** State where the property is physically located.
- 9- **PROPERTYZIP:** Zip Code where the property is physically located.
- 10- **SCHOOLDESC:** School district name associated with specified parcel.

- 11- **MUNICODE**: Municipality number associated with specified parcel.
- 12- **MUNIDESC**: Municipality name associated with specified parcel.
- 13- **RECORDDATE**: Date the sale was recorded in the Department of Real Estate. Format is mm/dd/yyyy.
- 14- **DEEDBOOK**: Cross-reference to the physical book number where the deed is on file in the Department of Real Estate.
- 15- **DEEDPAGE**: Page within the specified deed book corresponding to the first page of this physical deed.
- 16- **INSTRTYP**: A code describing the type of document used to record the real property transfer.
- 17- **INSTRTYPDESC**: A description of the type of document used to record the real property transfer.
- 18- **SALECODE**: A subjective categorization code denoting whether or not the sale price was representative of current market value. These categorizations are subject to change as sales are reviewed by the Office of Property Assessments or during a reassessment. See further explanation in the "Sale Validation Codes Details" resource.

#### 4.3.4 Deleting Duplicates & null values

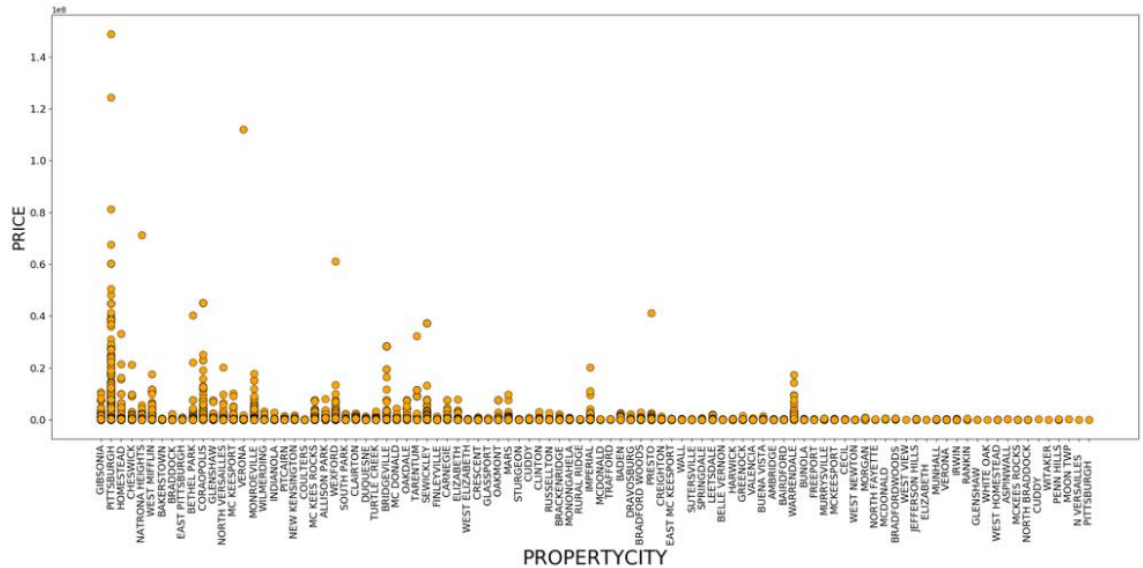
An important part of Data analysis is analyzing Duplicate Values and removing them. We found 22285 records are duplicated through the dataset, which will mislead during the data analysis and in model building stage.

Then, we dropped null values throughout the data cleaning process. Now Data shape is (283318, 6)

## 5 Exploratory Data Analysis

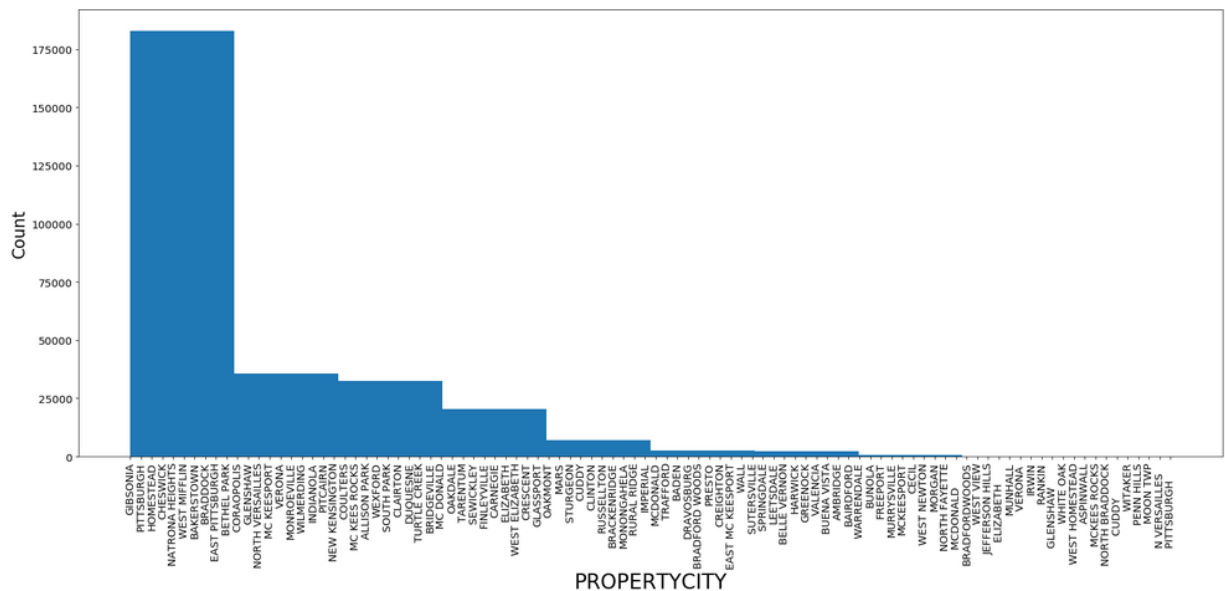
### 5.1 Relationship between PROPERTYCITY and PRICE.

Definitely the property price is changing by the City where the property is physically located. For the below relationship we can find that the highest property price is in PITTSBURGH city which equal 124400000 \$. The second highest property price is in the same city with amount of 60250000 \$.



## 5.2 Relationship between PROPERTYCITY and number of properties.

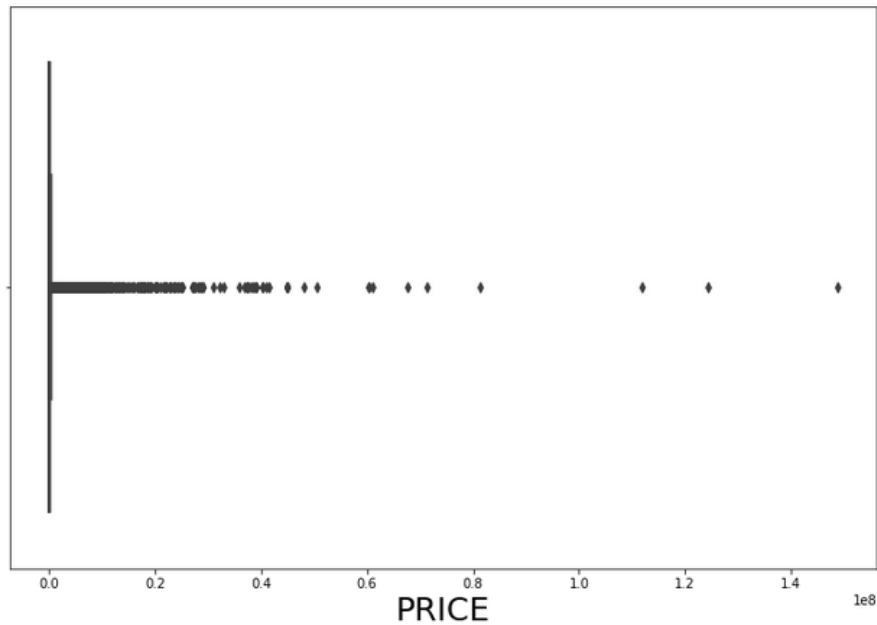
From the below figure we can conclude that the PITTSBURGH city has the most sold properties with count of 166634 property. Which is almost 54 % of the total properties in the whole dataset.



## 5.3 Boxplot to check Price outliers

Below figure indicates that there are some outliers that should be removed in order to avoid it bad impact on model prediction. They will be removed in the coming steps

```
In [30]: plt.figure(figsize=(12, 8))
sns.boxplot(x=data['PRICE'])
plt.xlabel("PRICE", fontsize=25);
```



## 5.4 Time series analysis using Prophet in Python

Prophet is a facebook's open source time series prediction. Prophet decomposes time series into trend, seasonality and holiday. It has intuitive hyper parameters which are easy to tune.

### Advantages of using Prophet

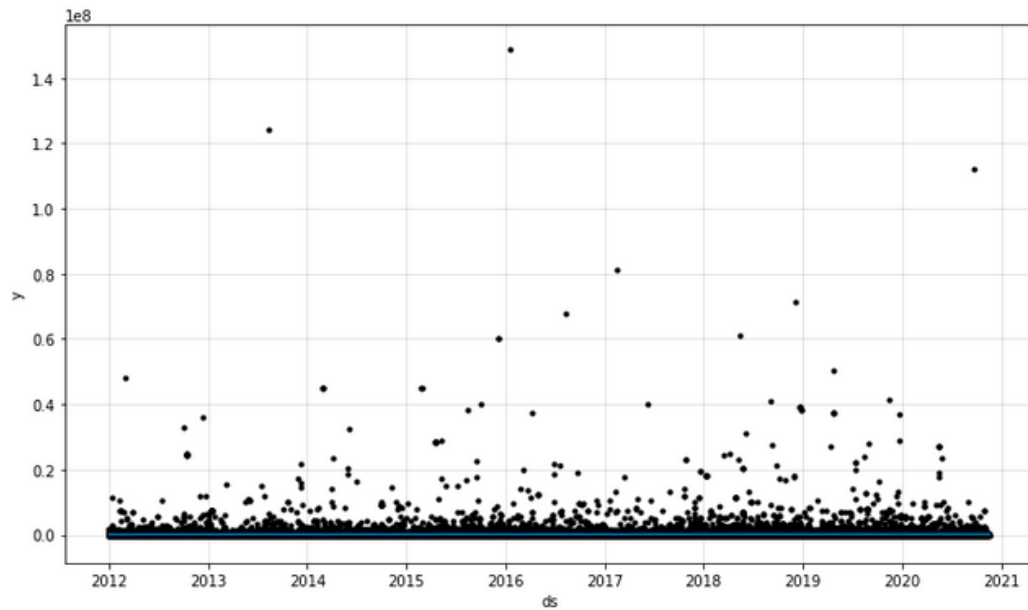
- Accommodates seasonality with multiple periods
- Prophet is resilient to missing values
- Best way to handle outliers in Prophet is to remove them
- Fitting of the model is fast
- Intuitive hyper parameters which are easy to tune

The below shows the Price distribution over the years from 2012 till 2021, we can find that the highest price occurred at 2016.

The second record is found recorder in 2013.

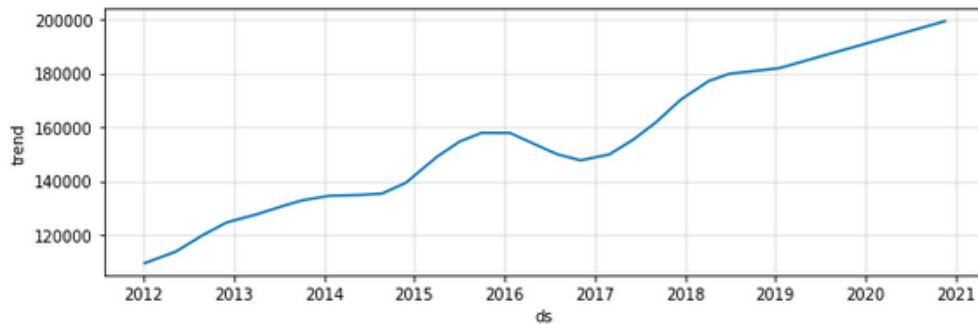
```
In [51]: forecast = m.predict(future)
#forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

```
In [52]: fig1 = m.plot(forecast)
```



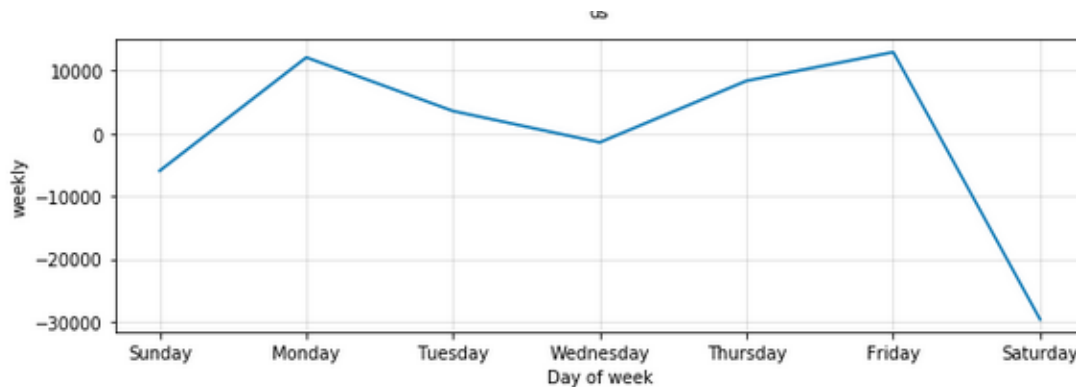
### Price forecast along different period of time - Yearly

```
In [53]: fig2 = m.plot_components(forecast)
```



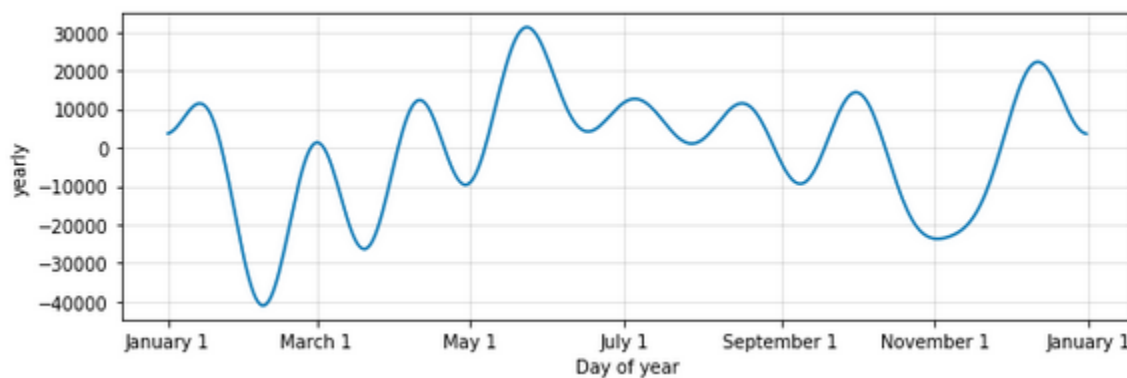
We can find that the price was increasing from 2012 to 2016 then it slightly fall at 2017. It has become upwards again gradually till the end of 2020.

### Price forecast along different period of time – Day of the week



Highest prices are recorded on Mondays and Fridays. Whereas Saturdays is almost the lowest which is normal behavior because it's the weekend.

### Price forecast along different period of time – Day of Year



Lowest prices was recorded January and March and the highest sale price is found between May and July.

## 5.5 Feature Engineering

### 5.5.1 Feature Engineering on Categorical Data

While a lot of advancements have been made in various machine learning frameworks to accept complex categorical data types like text labels. Typically any standard workflow in feature engineering involves some form of **transformation** of these categorical values into numeric labels and then applying some **encoding scheme** on these values. We load up the necessary essentials before getting started.



**Transforming Nominal Attributes:** Nominal attributes consist of discrete categorical values with no notion or sense of order amongst them. The idea here is to transform these attributes into a more representative numerical format which can be easily understood by downstream code and pipelines. Let's look at a new dataset pertaining to video game sales. This dataset is also available on **Kaggle** as well as in my **GitHub** repository.

Learning models accept only numbers as input, and since our dataset contains categorical features, we need to encode them in order for our dataset to be suitable for modeling. We will encode our categorical features using label encoding technique which transforms the categorical variable into a number of binary variables based on the number of unique categories in the categorical variable

We need to convert the following columns: PROPERTYCITY, PROPERTYADDRESSSTREET and SALEDESC

```
In [77]: from sklearn.preprocessing import LabelEncoder

gle = LabelEncoder()
PROPERTYCITY_labels = gle.fit_transform(data['PROPERTYCITY'])
PROPERTYCITY_mappings = {index: label for index, label in
                          enumerate(gle.classes_)}
PROPERTYCITY_mappings
```

```
Out[77]: {0: 'ALLISON PARK',
          1: 'AMBRIDGE',
          2: 'ASPINWALL',
          3: 'BADEN',
          4: 'BAIRDFORD',
          5: 'BAKERSTOWN',
```

```
In [80]: gle = LabelEncoder()
PROPERTYADDRESSSTREET_labels = gle.fit_transform(data['PROPERTYADDRESSSTREET'])
PROPERTYADDRESSSTREET_mappings = {index: label for index, label in
                                   enumerate(gle.classes_)}
PROPERTYADDRESSSTREET_mappings
```

```
Out[80]: {0: '10TH',
          1: '11TH',
          2: '12TH',
          3: '13TH',
          4: '14TH',
          5: '15TH',
          6: '16TH',
          7: '17TH',
          8: '18TH',
```

```
In [85]: gle = LabelEncoder()
SALEDESC_labels = gle.fit_transform(data['SALEDESC'])
SALEDESC_mappings = {index: label for index, label in
                      enumerate(gle.classes_)}
SALEDESC_mappings
```

```
Out[85]: {0: 'BANK/FINANCIAL INSTITUTION',
1: 'BUILDING NOT YET ASSESSED',
2: 'CHANGED AFTER SALE',
3: 'CITY TREASURER SALE',
4: 'CORPORATION TRANSFER',
5: 'CORRECTIVE DEED / DUPLICATE SALE',
```

Now, how data looks like after all these conversions

```
In [87]: data.head()
```

```
Out[87]:
```

	PROPERTYADDRESSSTREET	PROPERTYCITY	SCHOOLCODE	SALEDATE	PRICE	SALEDESC
0	3735	34	20	2012-09-27	120000.0	12
1	4769	70	47	2015-01-06	1783.0	3
2	2888	70	47	2012-10-26	4643.0	3
3	6970	70	29	2017-03-06	0.0	12
4	1327	70	47	2015-02-04	27541.0	10

## 5.5.2 Outlier Removal

Outliers are unusual values in your dataset, and they can distort statistical analyses and violate their assumptions. Unfortunately, all analysts will confront outliers and be forced to make decisions about what to do with them. Given the problems they can cause, you might think that it's best to remove them from your data. But, that's not always the case. Removing outliers is legitimate only for specific reasons.

Outliers can have many causes, such as:

- Measurement or input error.
- Data corruption.

```
In [91]: def remove_outlier(df, col):
q1 = df[col].quantile(0.25)
q3 = df[col].quantile(0.75)

iqr = q3 - q1
lower_bound = q1 - (1.5 * iqr)
upper_bound = q3 + (1.5 * iqr)

out_df = df.loc[(df[col] > lower_bound) & (df[col] < upper_bound)]
return out_df
```

```
In [93]: remove_outlier(data, 'PRICE')
```

Out[93]:

	PROPERTYADDRESSSTREET	PROPERTYCITY	SCHOOLCODE	SALEDATE	PRICE	SALEDESC
0	3735	34	20	2012-09-27	120000.0	12
1	4769	70	47	2015-01-06	1783.0	3
2	2888	70	47	2012-10-26	4643.0	3
3	6970	70	29	2017-03-06	0.0	12
4	1327	70	47	2015-02-04	27541.0	10
...	...	...	...	...	...	...
307748	2970	70	46	2019-05-15	45000.0	27
307749	5641	70	22	2019-05-14	168000.0	27

### 5.5.3 Remove zero values

We have to drop rows which contain zero records of Price column because there is no meaning to have a price equal to zero. So, I decided to delete all rows with zero value via following method.

#### Remove Zero Values

```
data[(data != 0).all(1)]
```

### 5.5.4 Data Normalization

**Data Normalization** is a common practice in machine learning which consists of transforming **numeric columns** to a **common scale**. In machine learning, some feature values differ from others multiple times. The features with higher values will dominate the leaning process. However, it does not mean those variables are more important to predict the outcome of the model. **Data normalization** transforms multiscale data to the same scale. After normalization, all variables have a **similar influence** on the model, improving the stability and performance of the learning algorithm.

There are multiple normalization techniques in statistics. **In this project I used the method provided by sklearn package: `sklearn.preprocessing.normalize`**

**Normalization** of datasets is a common requirement for many machine learning estimators implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance.

In practice we often ignore the shape of the distribution and just transform the data to center it by removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation.

For instance, many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the L1 and L2 regularizers of linear models) assume that all features are centered around zero and have variance in the same order. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

```
In [166]: # Normalize the data attributes for the dataset.
from sklearn import preprocessing
normalized = pd.DataFrame(preprocessing.normalize(data))

normalized.columns = data.columns
normalized.head()
```

Out[166]:

	PROPERTYADDRESSSTREET	PROPERTYCITY	SCHOOLCODE	PRICE	SALEDESC
0	0.031010	0.000275	0.000167	0.999519	0.000100
1	0.936089	0.013598	0.009262	0.351378	0.000591
2	0.527055	0.012627	0.008601	0.849694	0.000549
3	0.048018	0.002502	0.001705	0.998842	0.000363
4	0.997531	0.057937	0.039465	0.000840	0.004198

As we can notice that all values are between 0 and 1, so they are now meaningful and at the same scale and ready for modeling to get significance results.

### 5.5.5 Train – Test Split

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the

model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

- Train Dataset: Used to fit the machine learning model.
- Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model. This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values. The train-test procedure is appropriate when there is a sufficiently large dataset available.

## 6. Modeling Building

**6.1 Multiple Linear Regression:** Multiple Linear Regression (MLR) is a supervised technique used to estimate the relationship between one dependent variable and more than one independent variables. Identifying the correlation and its cause-effect helps to make predictions by using these relations, the prediction accuracy of the model is essential; the complexity of the model is of more interest. However, Multiple Linear Regression is prone to many problems such as multi collinearity, noises, and overfitting, which effect on the prediction accuracy.

Regularized regression plays a significant part in Multiple Linear Regression because it helps to reduce variance at the cost of introducing some bias, avoid the overfitting problem and solve ordinary least squares (OLS) problems. There are two types of regularization techniques L1 norm (least absolute deviations) and L2 norm (least squares). L1 and L2 have different cost functions regarding model complexity

### Algorithm Formulation

When implementing linear regression of some dependent variable  $y$  on the set of independent variables  $\mathbf{x} = (x_1, \dots, x_r)$ , where  $r$  is the number of predictors, you assume a linear relationship between  $y$  and  $\mathbf{x}$ :  $y = \beta_0 + \beta_1 x_1 + \dots + \beta_r x_r + \varepsilon$ . This equation is the **regression equation**.  $\beta_0, \beta_1, \dots, \beta_r$  are the **regression coefficients**, and  $\varepsilon$  is the **random error**.

Linear regression calculates the **estimators** of the regression coefficients or simply the **predicted weights**, denoted with  $b_0, b_1, \dots, b_r$ . They define the **estimated regression function**  $(\mathbf{x}) = b_0 + b_1 x_1 + \dots + b_r x_r$ . This function should capture the dependencies between the inputs and output sufficiently well.

The **estimated** or **predicted response**,  $(\mathbf{x}_i)$ , for each observation  $i = 1, \dots, n$ , should be as close as possible to the corresponding **actual response**  $y_i$ . The differences  $y_i - (\mathbf{x}_i)$  for all observations  $i = 1,$

...,  $n$ , are called the **residuals**. Regression is about determining the **best predicted weights**, that is the weights corresponding to the smallest residuals.

To get the best weights, you usually **minimize the sum of squared residuals** (SSR) for all observations  $i = 1, \dots, n$ :  $SSR = \sum_i (y_i - f(\mathbf{x}_i))^2$ . This approach is called the **method of ordinary least squares**.

## Regression Performance

The variation of actual responses  $y_i$ ,  $i = 1, \dots, n$ , occurs partly due to the dependence on the predictors  $\mathbf{x}_i$ . However, there is also an additional inherent variance of the output.

The **coefficient of determination**, denoted as  $R^2$ , tells you which amount of variation in  $y$  can be explained by the dependence on  $\mathbf{x}$  using the particular regression model. Larger  $R^2$  indicates a better fit and means that the model can better explain the variation of the output with different inputs.

The value  $R^2 = 1$  corresponds to  $SSR = 0$ , that is to the **perfect fit** since the values of predicted and actual responses fit completely to each other.

**Underfitting** occurs when a model can't accurately capture the dependencies among data, usually as a consequence of its own simplicity. It often yields a low  $R^2$  with known data and bad generalization capabilities when applied with new data.

**Overfitting** happens when a model learns both dependencies among data and random fluctuations. In other words, a model learns the existing data too well. Complex models, which have many features or terms, are often prone to overfitting. When applied to known data, such models usually yield high  $R^2$ . However, they often don't generalize well and have significantly lower  $R^2$  when used with new data.

```
from sklearn.model_selection import train_test_split
# Divide the data to train and test
X_normalized_train, X_normalized_test, y_normalized_train, y_normalized_test = train_test_split(X_normalized, y_normalized, test_size=0.2, random_state=42)

# Fitting Multiple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor_normalized = LinearRegression()
regressor_normalized.fit(X_normalized_train, y_normalized_train)

# Predicting the Test set results
y_pred_normalized = regressor_normalized.predict(X_normalized_test)

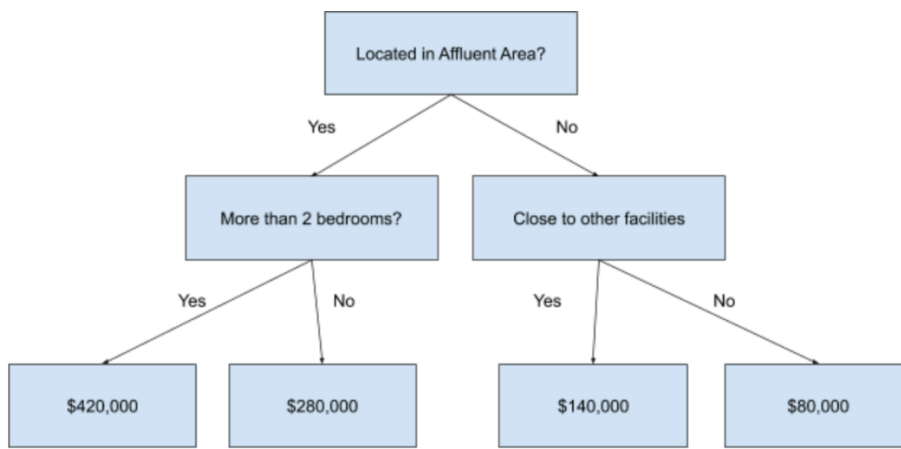
from sklearn import metrics
print("The accuracy result for The Normalized Multiple Linear Regression Model is ")
print(metrics.r2_score(y_normalized_test, y_pred_normalized))
```

```
The accuracy result for The Normalized Multiple Linear Regression Model is
0.9454325597076055
```

## 6.2 Random Forest Regression:

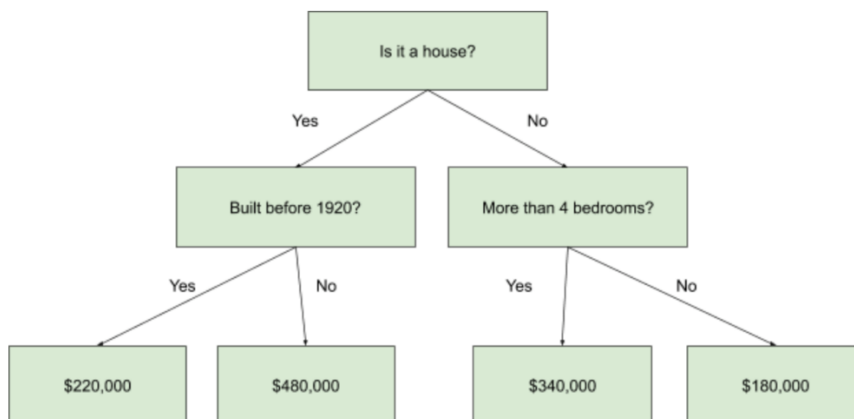
Random forest description by example: Let's start with an actual problem. Imagine you want to buy real estate, and you want to figure out what comprises a good deal so that you don't get taken advantage of.

The obvious thing to do would be to look at historic prices of houses sold in the area, then create some kind of decision criteria to summarize the average selling prices given the real-estate specification. You can use the decision chart to evaluate whether the listed price for the apartment you are considering is a bargain or not. It could look like this:



The chart represents a decision tree through a series of yes/no questions, which lead you from the real-estate description ("3 bedrooms") to its historic average price. You can use the decision tree to predict what the expected price of a real estate would be, given its attributes.

However, you could come up with a distinctly different decision tree structure:



This would also be a valid decision chart, but with totally different decision criteria. These decisions are just as well-founded and show you information that was absent in the first decision tree.

The random forest regression algorithm takes advantage of the 'wisdom of the crowds'. It takes multiple (but different) regression decision trees and makes them 'vote'. Each tree needs to predict the expected price of the real estate based on the decision criteria it picked. Random forest regression then calculates the average of all of the predictions to generate a great estimate of what the expected price for a real estate should be.

#### **Business use cases of random forest:**

- A- Predict future prices/costs:** Whenever your business is trading products or services (e.g. raw materials, stocks, labors, service offerings, etc.), you can use random forest regression to predict what the prices of these products and services will be in the future.
- B- Predict future revenue:** Use random forest regression to model your operations. For example, you can input your investment data (advertisement, sales materials, cost of hours worked on long-term enterprise deals, etc.) and your revenue data, and random forest will discover the connection between the input and output. This connection can be used to predict how much revenue you will generate based on the growth activity that you pick (marketing, direct to customer sales, enterprise sales, etc.) and how much you are willing to spend on it.
- C- Compare performance:** Imagine that you've just launched a new product line. The problem is, it's unclear whether the new product is attracting more (and higher spending) customers than the existing product line. Use random forest regression to determine how your new product compares to your existing ones.

#### **Random forest approaches:**

It is supervised in the sense that during training, it learns the mappings between inputs and outputs. For example, an input feature (or independent variable) in the training dataset would specify that an apartment has "3 bedrooms" (feature: *number of bedrooms*) and this maps to the output feature (or target) that the apartment will be sold for "\$200,000" (target: *price sold*).

Ensemble algorithms combine multiple other machine learning algorithms, in order to make more accurate predictions than any underlying algorithm could on its own. In the case of random forest, it ensembles multiple decision trees into its final decision.

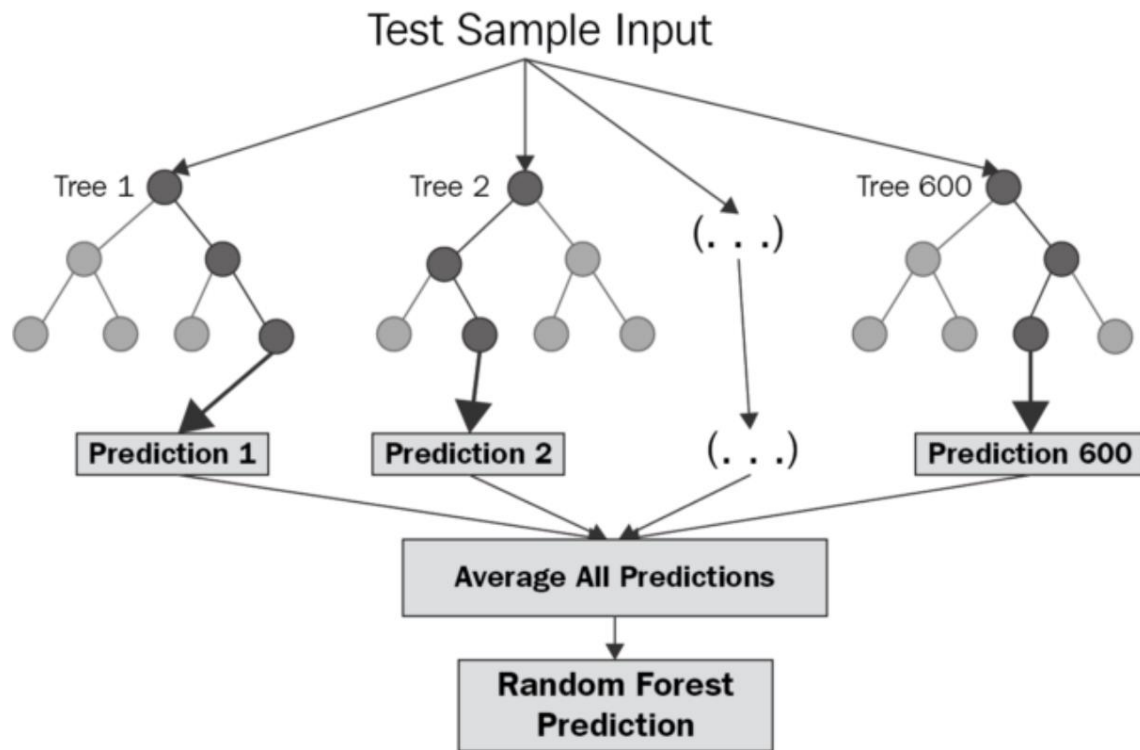
Random forest can be used on both regression tasks (predict continuous outputs, such as price) or classification tasks (predict categorical or discrete outputs). Here, we will take a deeper look at using random forest for regression predictions.

#### **The random forest algorithm follows a two-step process:**

- A- Builds  $n$  decision tree repressors (estimators).** The number of estimators  $n$  defaults to 100 in Scikit Learn (the machine learning Python library), where it is called  $n\_estimators$ . The trees are built following the specified hyper parameters (e.g. minimum number of samples at the leaf nodes, maximum depth that a tree can grow, etc.).



- B- Average prediction across estimators.** Each decision tree regression predicts a number as an output for a given input. Random forest regression takes the average of those predictions as its 'final' output.



**How to improve the RFR model:**

1. **Specify the maximum depth of the trees:** By default, trees are expanded until all leaves are either pure or contain less than the minimum samples for the split. This can still cause the trees to overfit or underfit. Play with the hyper parameter to find an optimal number for max\_depth.
2. **Increase or decrease the number of estimators:** How does changing the number of trees affect performance? More trees usually means higher accuracy at the cost of slower learning. If you wish to speed up your random forest, lower the number of estimators. If you want to increase the accuracy of your model, increase the number of trees.
3. **Specify the maximum number of features to be included at each node split:** This depends very heavily on your dataset. If your independent variables are highly correlated, you'll want to decrease the maximum number of features. If your input attributes are not correlated and your model is suffering from low accuracy, increase the number of features to be included.

## Model performance output

```
In [55]: # Random Forest Regression using 200 trees with out of bag = true
from sklearn.ensemble import RandomForestRegressor
regressor_RFR = RandomForestRegressor(n_estimators = 200, random_state = 0, oob_score=True)
regressor_RFR.fit(X_RFR_train, y_RFR_train)
y_predRandomForest = regressor_RFR.predict(X_RFR_test)
from sklearn import metrics
print( metrics.r2_score(y_RFR_test,y_predRandomForest))

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
after removing the cwd from sys.path.

0.999871436492488
```

## 7. Evaluation

### 7.1 Coefficient of Determination – R2 score

In the domain of Data Science, to solve any model it is very necessary for the engineer/developer to evaluate the [efficiency of a model](#) prior to applying it to the [dataset](#). The evaluation of the model is based on certain error metrics. The coefficient of determination is one such error metric.

Coefficient of Determination also popularly known as R square value is a regression error metric to evaluate the accuracy and efficiency of a model on the data values that it would be applied to.

R square values describe the performance of the model. It describes the variation in the response or target variable which is predicted by the independent variables of the data model.

Thus, in simple words we can say that, the R square value helps determine how well the model is blend and how well the output value is explained by the determining(independent) variables of the dataset.

The value of R square ranges between [0,1]. Have a look at the below formula!

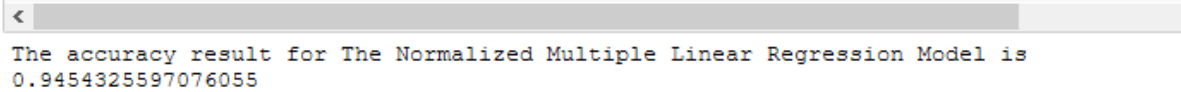
$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

- $SS_{res}$  represents the sum of squares of the residual errors of the data model.
- $SS_{tot}$  represents the total sum of the errors.

## R2 for linear Regression Model:

The accuracy result for The Normalized Multiple Linear Regression Model: 0.945

```
from sklearn import metrics
print( "The accuracy result for The Normalized Multiple Linear Regression Model is ")
print( metrics.r2_score(y_normalized_test,y_pred_normalized))
```



```
The accuracy result for The Normalized Multiple Linear Regression Model is
0.9454325597076055
```

## R2 for Random Forest Model:

The accuracy of the model = 0.999871436492488

```
In [56]: print("the accuracy of the model = " + str(metrics.r2_score(y_RFR_test,y_predRandomForest)))
```

```
the accuracy of the model = 0.999871436492488
```

## 7.2 Root Mean Square Error (RMSE)

Mean Square error is one such error metric for judging the accuracy and error rate of any machine learning algorithm for a regression problem.

So, MSE is a risk function that helps us determine the average squared difference between the predicted and the actual value of a feature or variable.

RMSE is an acronym for **Root Mean Square Error**, which is the **square root of value obtained from Mean Square Error** function.

**Using RMSE, we can easily plot a difference between the estimated and actual values of a parameter of the model.**

By this, we can clearly judge the efficiency of the model.

Usually, a RMSE score of less than 180 is considered a good score for a moderately or well working algorithm. In case, the RMSE value exceeds 180, we need to perform feature selection and hyper parameter tuning on the parameters of the model.

### RMSE for linear Regression Model:

```
In [54]: from sklearn.metrics import mean_squared_error
import math

MSE = mean_squared_error(y_normalized_test, y_pred_normalized)

RMSE = math.sqrt(MSE)
print("Root Mean Square Error:\n")
print(RMSE)

Root Mean Square Error:

0.09579787693154342
```

### RMSE for Random Forest Model:

```
In [58]: from sklearn.metrics import mean_squared_error
import math

MSE = mean_squared_error(y_RFR_test, y_predRandomForest)

RMSE = math.sqrt(MSE)
print("Root Mean Square Error for Random Forest:\n")
print(RMSE)

Root Mean Square Error for Random Forest:

0.004631905714294685
```

## 9. Model Deployment

- 1- Convert the model to deployable file using pickle

### Deployment

```
In [183]: import pickle

In [184]: pickle.dump(regressor_RFR, open('model.pkl', 'wb'))
```

## 2- Using flask as intermediate webserver to read the model and handle http request

```
from flask import Flask, request
import numpy as np
import pickle
#! pip install -U flask-cors
from flask_cors import CORS

model = pickle.load(open('model.pkl', 'rb'))
app = Flask(__name__)
CORS(app)
cors = CORS(app, resources={r"/predict": {"origins": "http://localhost:port"}})
@app.route('/predict', methods=['POST'])
def predict():
    params = request.form.getlist('params')
    inputs = [float(e) if e.isdigit() else e for e in params[0].split(',')]
    sample = np.array(inputs, ndmin=2, dtype = float)
    # res = model.predict()
    return str(model.predict(sample.tolist()))

if __name__ == "__main__":
    app.run(port=7000)
```

## 3- Create UI (website) by using Asp MVC .Net core.

# Welcome to house price prediction

Property street Code:

Property city Code:

School Code:

Sale description Code:

The predicted price is 0.10991254

Please note that the result is in a normalized form



## 10. Conclusion

In this project, we built several regression models to predict the price of some house given some of the house features. We evaluated and compared each model to determine the one with highest performance. We also looked at how some models rank the features according to their importance. In

this project, we followed the data science process starting with getting the data, then cleaning and preprocessing the data, followed by exploring the data and building models, then evaluating the results and communicating them with visualizations.

As a recommendation, we advise to use this model (or a version of it trained with more recent data) by people who want to buy a house in the area covered by the dataset to have an idea about the actual price. The model can be used also with datasets that cover different cities and areas provided that they contain the same features. We also suggest that people take into consideration the features that were deemed as most important as seen in the previous section; this might help them estimate the house price better.