A REPORT ON

HOUSE PRICE PREDICTION

BY

V.SAI SUSANTH

(19STUCHH010173)

Department

Data Warehousing and Mining (DWM)

Section – A

ABSTRACT

House rate forecasting is an important topic of actual estate. The literature tries to derive beneficial understanding from historic records of belongings markets. Machine mastering strategies are implemented to analyze historic property transactions in India to discover beneficial models for residence shoppers and sellers. Revealed is the high discrepancy among residence fees within the most high-priced and maximum low priced suburbs within the metropolis of Mumbai. Moreover, experiments exhibit that the Multiple Linear Regression that is based totally on mean squared blunders dimension is a aggressive method.

ABBREVIATIONS

SqM Square meter

SqFt Square Feet

LR Linear

Regression XGB

Xtreme Gradient Boost

rmse Root Mean

Square Error Log

Logarithmie

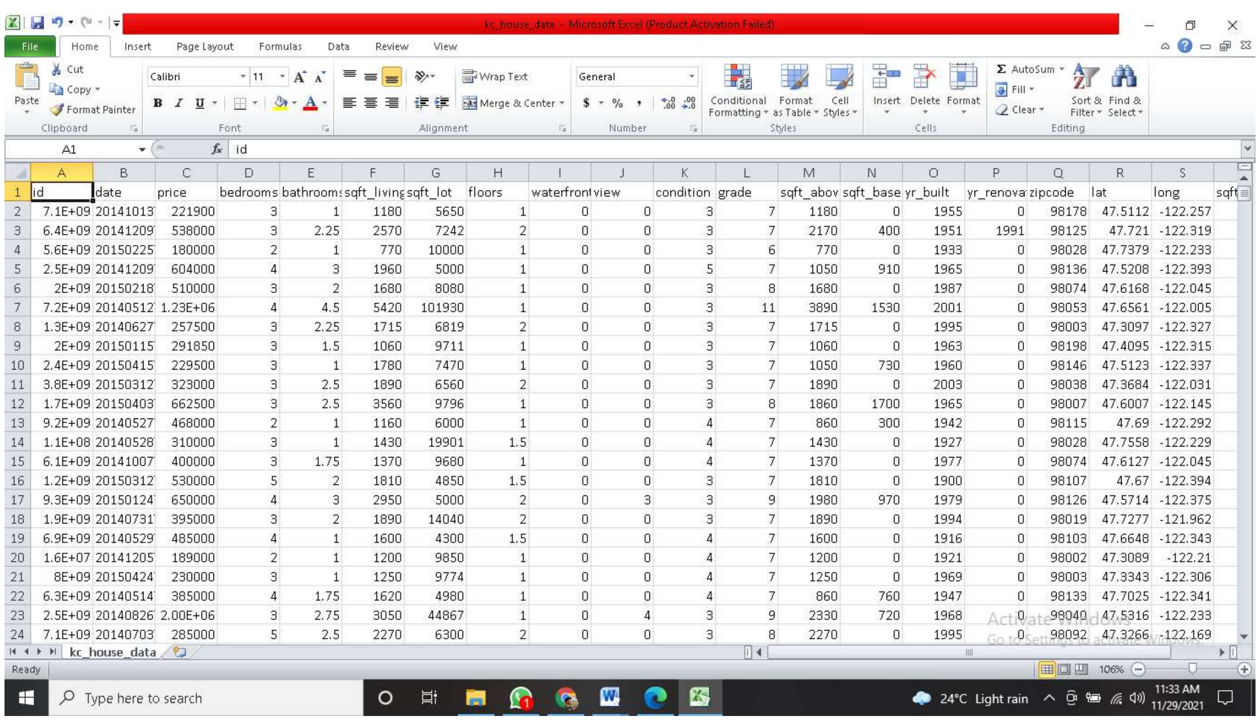
TABLE OF CONTENTS

CHAPTER-1

Need and motivation

Having lived in India for such a lot of years if there may be one element that I have been taking for granted, it's that housing and condominium costs preserve to upward push. Since the housing disaster of 2008, housing fees have recovered remarkably properly, specially in foremost housing markets. However, inside the 4th zone of 2016, I become amazed to examine that Bombay housing charges had fallen the maximum in the last 4 years. In fact, median resale fees for condos and coops fell 6.Three%, marking the first time there has been a decline on the grounds that Q1 of 2017. The decline has been partly attributed to political uncertainty domestically and overseas and the 2014 election. So, to maintain the transparency among customers and additionally the contrast may be made clean through this model. If customer unearths the fee of residence at some given website higher than the rate expected by way of the version, so he can reject that residence.

Data sets looks as follows:



1	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_abov	sqft_base	yr_built	yr_renova	zipcode	lat	long
2	7.1E+09	20141013	221900	3	1	1180	5650	1	0	0	3	7	1180	0	1955	0	98178	47.5112	-122.257
3	6.4E+09	20141209	538000	3	2.25	2570	7242	2	0	0	3	7	2170	400	1951	1991	98125	47.721	-122.319
4	5.6E+09	20150225	180000	2	1	770	10000	1	0	0	3	6	770	0	1933	0	98028	47.7379	-122.233
5	2.5E+09	20141209	604000	4	3	1960	5000	1	0	0	5	7	1050	910	1965	0	98136	47.5208	-122.393
6	2E+09	20150218	510000	3	2	1680	8080	1	0	0	3	8	1680	0	1987	0	98074	47.6168	-122.045
7	7.2E+09	20140512	1.23E+06	4	4.5	5420	101930	1	0	0	3	11	3890	1530	2001	0	98053	47.6561	-122.005
8	1.3E+09	20140627	257500	3	2.25	1715	6819	2	0	0	3	7	1715	0	1995	0	98003	47.3097	-122.327
9	2E+09	20150115	291850	3	1.5	1060	9711	1	0	0	3	7	1060	0	1963	0	98198	47.4095	-122.315
10	2.4E+09	20150415	229500	3	1	1780	7470	1	0	0	3	7	1050	730	1960	0	98146	47.5123	-122.337
11	3.8E+09	20150312	323000	3	2.5	1890	6560	2	0	0	3	7	1890	0	2003	0	98038	47.3684	-122.031
12	1.7E+09	20150403	662500	3	2.5	3560	9796	1	0	0	3	8	1860	1700	1965	0	98007	47.6007	-122.145
13	9.2E+09	20140527	468000	2	1	1160	6000	1	0	0	4	7	860	300	1942	0	98115	47.69	-122.292
14	1.1E+08	20140528	310000	3	1	1430	19901	1.5	0	0	4	7	1430	0	1927	0	98028	47.7558	-122.229
15	6.1E+09	20141007	400000	3	1.75	1370	9680	1	0	0	4	7	1370	0	1977	0	98074	47.6127	-122.045
16	1.2E+09	20150312	530000	5	2	1810	4850	1.5	0	0	3	7	1810	0	1900	0	98107	47.67	-122.394
17	9.3E+09	20150124	650000	4	3	2950	5000	2	0	3	3	9	1980	970	1979	0	98126	47.5714	-122.375
18	1.9E+09	20140731	395000	3	2	1890	14040	2	0	0	3	7	1890	0	1994	0	98019	47.7277	-121.962
19	6.9E+09	20140529	485000	4	1	1600	4300	1.5	0	0	4	7	1600	0	1916	0	98103	47.6648	-122.343
20	1.6E+07	20141205	189000	2	1	1200	9850	1	0	0	4	7	1200	0	1921	0	98002	47.3089	-122.21
21	8E+09	20150424	230000	3	1	1250	9774	1	0	0	4	7	1250	0	1969	0	98003	47.3343	-122.306
22	6.3E+09	20140514	385000	4	1.75	1620	4980	1	0	0	4	7	860	760	1947	0	98133	47.7025	-122.341
23	2.5E+09	20140826	2.00E+06	3	2.75	3050	44867	1	0	4	3	9	2330	720	1968	0	98040	47.5316	-122.233
24	7.1E+09	20140703	285000	5	2.5	2270	6300	2	0	0	3	8	2270	0	1995	0	98092	47.3266	-122.169

Data exploration:

Data exploration is the first step in information evaluation and generally includes summarizing the principle characteristics of a statistics set, along with its length, accuracy, preliminary patterns in the facts and different attributes. It is normally performed by using data analysts the usage of visual analytics gear, but it may additionally be executed in extra advanced statistical software program, Python.

Data Visualization:

Data visualization is the graphical illustration of statistics and information. By using visual factors like charts, graphs, and maps, data visualization gear provide an reachable way to look and recognize developments, outliers, and patterns in statistics. In the world of Big Data, statistics visualization gear and technology are crucial to examine huge quantities of data and make facts-driven selections.

Data selection:

Data selection is described as the process of determining the best information type and supply, as well as suitable contraptions to acquire facts. The major goal of facts selection is to decide the perfect statistics type, source, and instrument(s) that lets in facts scientists/analysts to get insights of the facts.

Language used:

Python is a pc programming language regularly used to construct web sites and software, automate duties, and behavior records analysis. Python is a widespread reason language, meaning it can be used to create a diffusion of various programs and is not specialised for any particular problems.It is widely used in scientific and numeric computing:

Libraries in python:

Python Libraries are a hard and fast of beneficial capabilities that cast off the want for writing codes from scratch. There are over 137,000 python libraries present today. Python libraries play a crucial role in growing system getting to know, statistics technology, facts visualization, photo and facts manipulation packages, and more.

Libraries used are:

- Numpy
- Pandas
- Matplotlib
- Seaborn
- Scikt-learn

Packages in python:

A Python package usually consists of several modules.The package is simply a namespace. The package also contains sub-packages inside it.

Packages used:

- Import

CHAPTER-2

MODELS USED

Among the large choice of learning algorithm I chose to use Linear Regression
because my dependent variable is continuous. The goal was to model a relationship between
y , the dependent variable and x , .. , x , the independent variable where is the number of 1
. p p
them. The general form of this is given by the following equation :
y = f(x) + ε.

Modules in python

The module is a simple Python file that contains collections of functions and global variables and with having a .py extension file.

A dataset is needed for building up a regression model. As the project is based on the residence fee predictions, we need to have dataset that is composed info about price, bedrooms, square feet per residing, flooring, waterfront, view, circumstance, grade, rectangular above, rectangular ft basement, 12 months built, 12 months renovated, zip code, range, longitude etc After having the dataset, its important to look which column is important and which isn't. The essential objective is to make a version which can supply an excellent prediction on the charge of the house based totally on the variables. By using linear regression for the dataset, you may if it may deliver properly accuracy or not. So, the accuracy depends on the on what form of data we are operating with, for credit score chance statistics with an accuracy of 80% won't be accurate enough but for records using NLP it might be excellent. We can't truely define "excellent accuracy" but something above eighty five% is ideal. The intention in this dataset is to attain an accuracy of 85%.

After having the dataset, the following step to proceed with is by uploading the dataset and libraries.

```
[1]  import pandas as pd
     import numpy as np
     import scipy as sp
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression
     from sklearn import metrics

     %matplotlib inline
```

- **SciPy** is a collection of packages for mathematics, science, and engineering.
- **Pandas** is a data analysis and modelling library.
- **IPython** is a powerful interactive shell that features easy editing and recording of a work session, and supports visualizations and parallel computing.
- **Matplotlib** is a *plotting library* for Python
- **Seaborn** is a *Python data visualization library* based on matplotlib

- **Sklearn** is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistent interface in Python.

  sklearn.model_selection is a method for setting a blueprint to analyze data and then using it to measure new data.

  sklearn.linear_model is a class of the sklearn module that contains different functions for performing machine learning with linear models.

  train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data

  metrics module implements various loss, score, and utility functions to measure classification performance.

After that,load the dataset with pandas and look into the head of the data to know how it looks like .

Loading the data :

```
[ ]  data= pd.read_csv("kc_house_data.csv")
     data.head() #top 5 rows of the dataset will be displayed
```

Fig 2.2 loading data

- Pandas **read_csv()** function **imports** a CSV file to DataFrame format.
- **head()** function returns top 5 rows of the dataset provided.

Output:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180	0	1955	0
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170	400	1951	1991
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	3	6	770	0	1933	0
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050	910	1965	0
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680	0	1987	0

zipcode	lat	long	sqft_living15	sqft_lot15
98178	47.5112	-122.257	1340	5650
98125	47.7210	-122.319	1690	7639
98028	47.7379	-122.233	2720	8062
98136	47.5208	-122.393	1360	5000
98074	47.6168	-122.045	1800	7503

Next to know more about the dataset,we use describe() function.

```
data.describe() #Descriptive statistics
```

Fig 2.3 Describe function

- The **describe**() method is used for calculating some statistical data like percentile, mean and std, count etc.. of the numerical values of the Series or DataFrame.

Output:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000

grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15
21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
7.656873	1788.390691	291.509045	1971.005136	84.402258	98077.939805	47.560053	-122.213896	1986.552492	12768.455652
1.175459	828.090978	442.575043	29.373411	401.679240	53.505026	0.138564	0.140828	685.391304	27304.179631
1.000000	290.000000	0.000000	1900.000000	0.000000	98001.000000	47.155900	-122.519000	399.000000	651.000000
7.000000	1190.000000	0.000000	1951.000000	0.000000	98033.000000	47.471000	-122.328000	1490.000000	5100.000000
7.000000	1560.000000	0.000000	1975.000000	0.000000	98065.000000	47.571800	-122.230000	1840.000000	7620.000000
8.000000	2210.000000	560.000000	1997.000000	0.000000	98118.000000	47.678000	-122.125000	2360.000000	10083.000000
13.000000	9410.000000	4820.000000	2015.000000	2015.000000	98199.000000	47.777600	-121.315000	6210.000000	871200.000000

```
data.isnull().sum()  #Data not having any NaNs
```

Fig 2.4 Checking if data has any missing values.

- **isnull()** function returns a specified value if the expression is NULL.
- **sum()** function adds up all the numerical values in an iterable(such as a list).
  Output:

```
id                 0
date               0
price              0
bedrooms           0
bathrooms          0
sqft_living        0
sqft_lot           0
floors             0
waterfront         0
view               0
condition          0
grade              0
sqft_above         0
sqft_basement      0
yr_built           0
yr_renovated       0
zipcode            0
lat                0
long               0
sqft_living15      0
sqft_lot15         0
dtype: int64
```

Next, a plot for all the variables of the dataset should be plotted inorder to visualize the data and see what we can infer from the given data and identified the top 5 features of house price prediction among provide variables.

```
names=['price','bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','vie
w','condition','grade','sqft_above','sqft_basement','zipcode','lat','long']
df=data[names]
correlations= df.corr()
fig=plt.figure()
ax=fig.add_subplot(111)
cax=ax.matshow(correlations,vmin=-1,vmax=1)
fig.colorbar(cax)
ticks=np.arange(0,15,1)
ax.set_xticks(ticks)
ax.set_yticks(ticks)
ax.set_xticklabels(names)
ax.set_yticklabels(names)
plt.show()
```

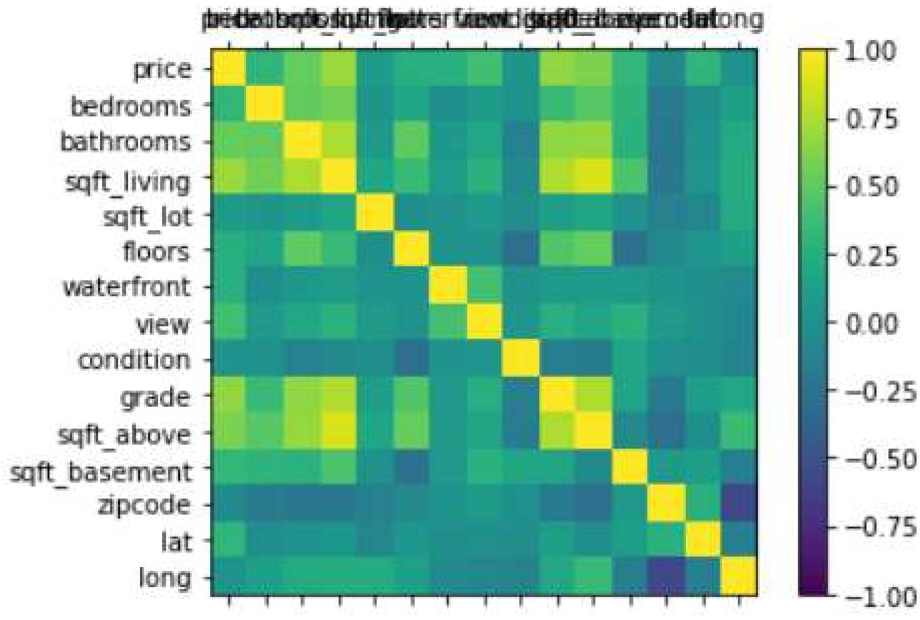
Fig 2.4

- **df.corr()** method is used for creating the correlation matrix. It is used to find the pairwise correlation of all columns in the dataframe. For any non-numeric data type columns in the dataframe it is ignored.

To create correlation matrix using pandas, these steps should be taken:

1. Obtain the data.
2. Create the DataFrame using Pandas.
3. Create correlation matrix using Pandas.

- The purpose of using **plt. figure()** is to create a figure object.The whole figure is considered as figure object.
- The **add_subplot()** method figure module of matplotlib library is used to add an Axes to the figure as part of a subplot arrangement.
- np.arange()
- plt.show()




```
Top 5 features:
1. Bedrooms
2. Bathrooms
3. sqft_living
4. sqft_above
5. grade
```

Fig 2.5 Heatmap

The diagonal is obviously equal to 1, since it represents the correlation between the same attributes. Gross area and saleable area seem to be highly correlated, meaning by adding saleable area to my learning algorithm I would not have a major changes in terms of results. As I thought, while the gross area increases the price increases too. Number of bedrooms has the lowest correlation to the price among all the variables

Now Let's convert nominal and ordinal features into category and see the datatypes of all the variables.

```
data['waterfront'] = data['waterfront'].astype('category')
data['view'] = data['view'].astype('category')
data['condition'] = data['condition'].astype('category')
data['grade'] = data['grade'].astype('category',)
data['zipcode'] = data['zipcode'].astype('category')
```

```
[ ] data.dtypes #datatypes of the variables.

id                 int64
date              object
price            float64
bedrooms           int64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront      category
view            category
condition       category
grade           category
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
zipcode         category
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
dtype: object
```

Fig 2.6 datatypes of the variables in the dataset.

Now ,lets get into exploratory data analysis to understand the data more clearly.In this we will have various plots like scatter,strip,box,regression plots etc to visualize the data to draw conclusions more accurately.

- To begin with,lets plot a regression plot using seaborn for sqft_living on x-axis and price on y-axis using *sns.regplot()*. a plot that includes a regression line we would expect that line to coincide the scatter points.

```
[ ]  sns.regplot(x='sqft_living',y='price',data=data) #Reg plot
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fce1c788dd0>
```

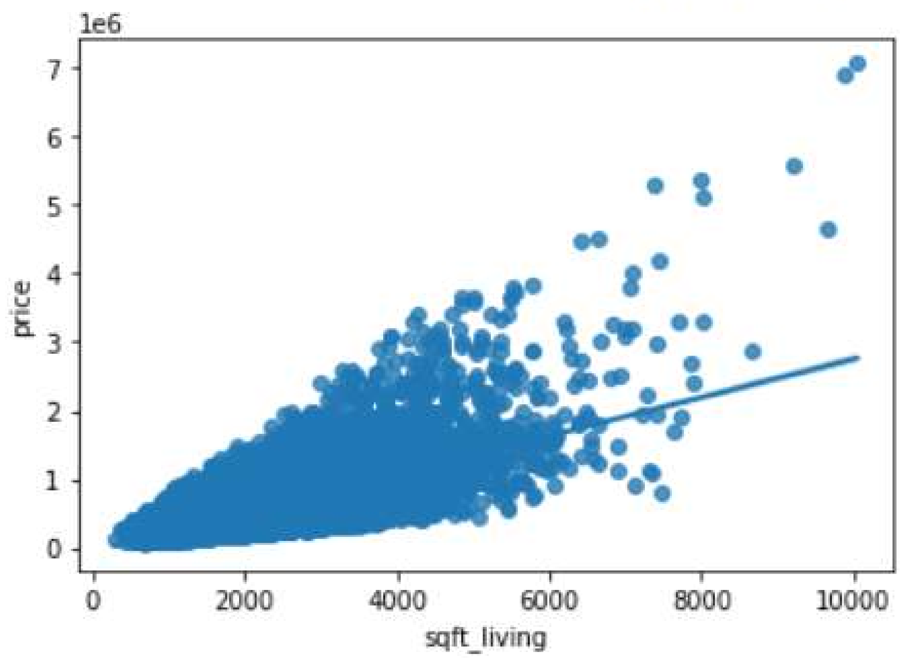


Fig 2.7 Regplot for sqft_living and price

Observation: From the graph,what conclusion to be drawn?Well,as we will see as the sqft will increase the charge progressively will increase,although statistics is focused towards a specific charge region.Higher the sqft,better the rate.We also can see that very less points are scattered as squarefeet will become better and houses with better squarefeet changed into sold much less.

● Moving on,we could visualize some greater capabilities influencing charge of residence usingscatterplots.Next, Fig 2.8 is a sqft_basement vs rate plot and Fig 2.Nine is a sqft_above vs fee plot.Lets see,what we will infer from the plot …

```
[ ]  sns.regplot(x='sqft_basement',y='price',data=data)

<matplotlib.axes._subplots.AxesSubplot at 0x7fce1d652ed0>
```

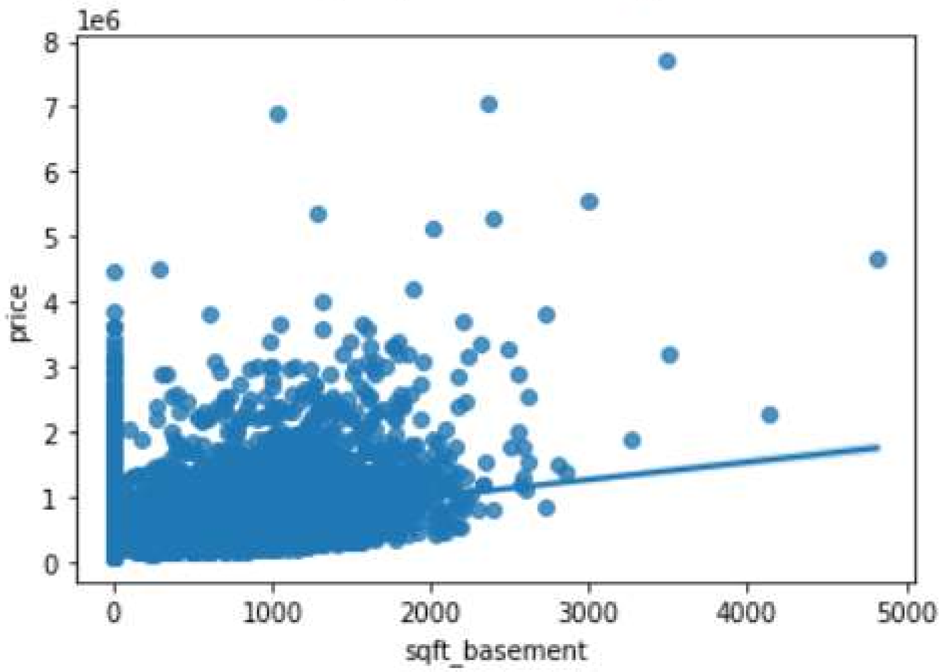


Fig 2.8 Sqft_basement vs price

```
[ ]  sns.regplot(x='sqft_above',y='price',data=data)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fce1d5cf050>

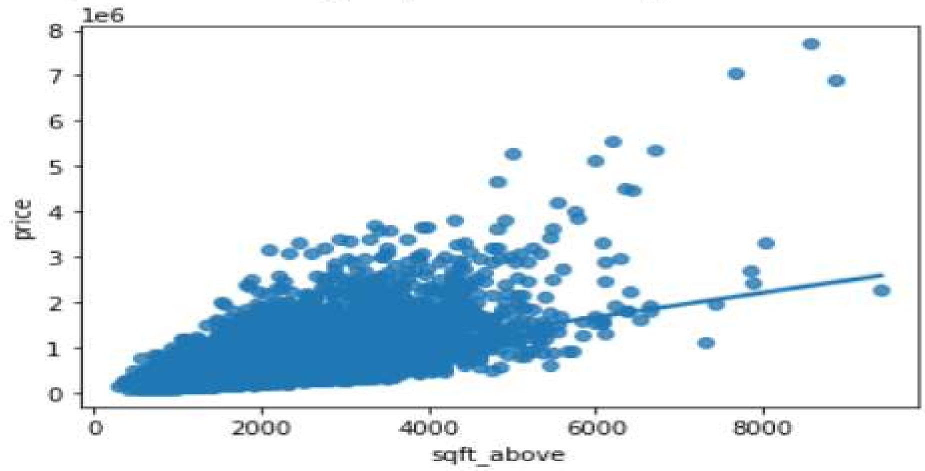


Fig 2.9 sqft_above vs price

Observation:

Let us use another type of plot which is strip plot for further visualizations. Strip plot is used to draw a scatter plot based on the category.

```
sns.stripplot(x='bedrooms', y='price',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbcf4e3a890>
```

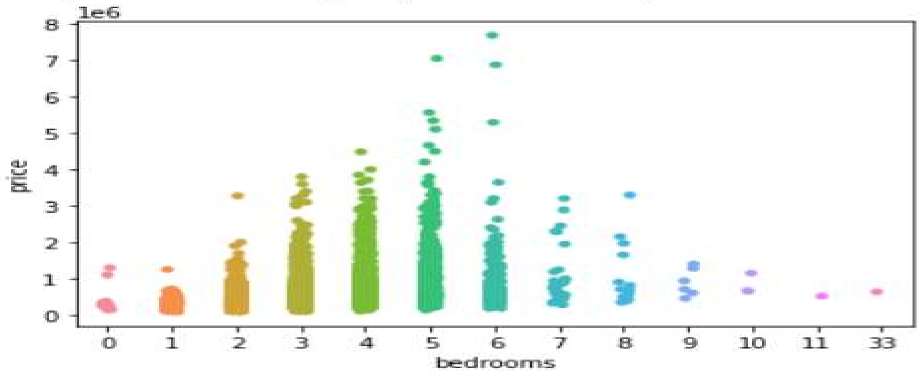


Fig 2.10 Bedrooms vs Price

Observation: By looking at this plot, we can see that up to 6 bedrooms, the price is gradually increasing but after that, the price started to decrease. One more thing I observed is most people are preferring 3 or more than 3 bedrooms but less than 6 bedrooms as there are more scattered data points.

```
sns.stripplot(x='bathrooms', y='price',data=data, size=5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fce1d4235d0>
```

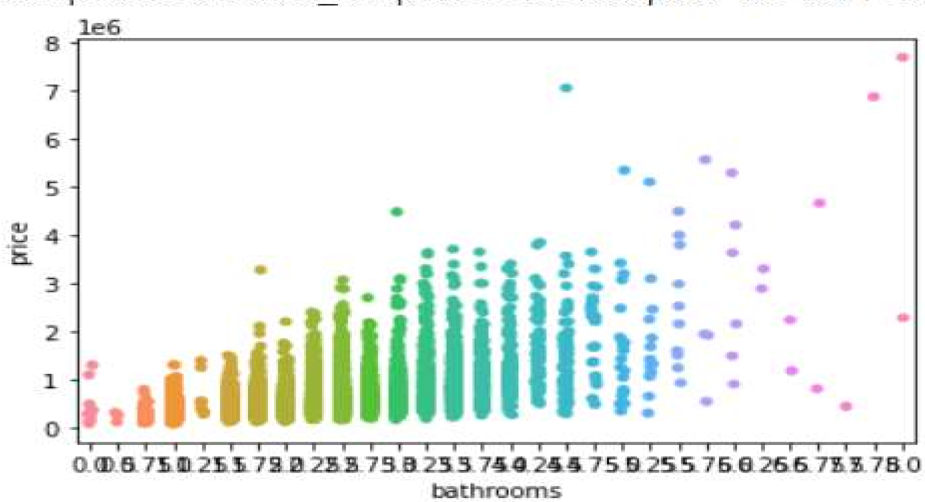


Fig 2.11 Bathrooms vs price

```
sns.stripplot(x='grade', y='price',data=data, size=5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fce1d32b750>
```

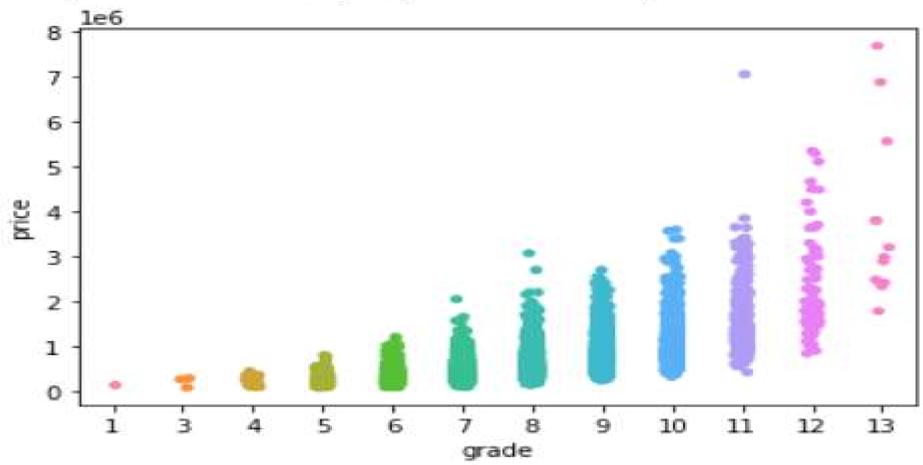


Fig 2.12 Grade vs price

<u>Observation</u>: By looking at the above strip plot, we can see that the grade /rating of the house has a great influence on the sales price. The better the ratings, the more people will be interested in buying that house. The price of the house is dependent on the grade. Higher the grade, the higher the price.

So, till now we have created some plots between some parameters we have taken from the dataset and seen the visualizations on how each one of them is influencing the price of the house. Let's remove some outliers to increase the power of your test and make the results look stronger.

We can remove outliers from data like houses with bedrooms > 9 and bathrooms>7.

```
[ ]  data=data[data['bedrooms'] < 10]
```

```
[ ]  data=data[data['bathrooms']<8]
```

After removing them, let's see how the data looks like using the head() function. This is how the head of the data looks like.

```
[ ]  data.head()
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipco
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180	0	1955	0	981
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170	400	1951	1991	981
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	3	6	770	0	1933	0	980
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050	910	1965	0	981
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680	0	1987	0	980

```
[ ]  #Building a model with top 5 parameters
```

```
[ ]  c=['bedrooms','bathrooms','sqft_living','sqft_above','grade']
     df=data[c]#taking the 5 features in a new datafram df
```

```
[ ]  df=pd.get_dummies(df,columns=['grade'], drop_first=True)#Creating dummies
```

```
y=data['price']#price column data is assigned into variable named 'y'
```

- Pandas **get_dummies()** is used for data manipulation. It converts categorical data into dummy or indicator variables.

- drop_first() method is to see whether to get k-1 dummies out of k categorical levels by removing the first level. The default value is false.

Next, we should divide the single data for different purposes(i.e.Training and Testing) using the train_test_split function by sklearn.

CONCLUSION

We have managed to put together a model that offers customers with a brand new optimization approach through thinking about destiny projections of stored fee. Several relapse strategies have been further in comparison, while one XG-assisted predictive strategy emerged. Any earlier rights mean that the works utilized in our model are form of predictions of future value as a way to tend in the direction of more sensitive values. We have created an approach that makes use of as an awful lot records as possible for our prediction machine, via those announcement thoughts about gradient amplification. While hosting generated all distribution efforts that met our rollout necessities, there are numerous upgrades that can be produced after that. They include improvements that we didn't pick out because of time constraints. A real situation for prediction frameworks might be the stacking segment. Also, our dataset takes more than a day to prepare. Instead of doing the calculations sequentially, we will use extraordinary processors and parallelize the calculations concerned, which can reduce the instruction time past the prediction c language. Including All other features within the version, we are able to make a preference for customers via selecting a district alternately locale should produce those excessive temperature maps, instead of getting into within the list.

REFERENCE

https://github.com/Shreyas3108/house-price-prediction

https://colab.research.google.com/drive/1o3NnIdsA048b2GuivJ6qfTcpFlYr-2_P#scrollTo=CXJhB21r0vEz

https://www.geeksforgeeks.org/linear-regression-python-implementation/