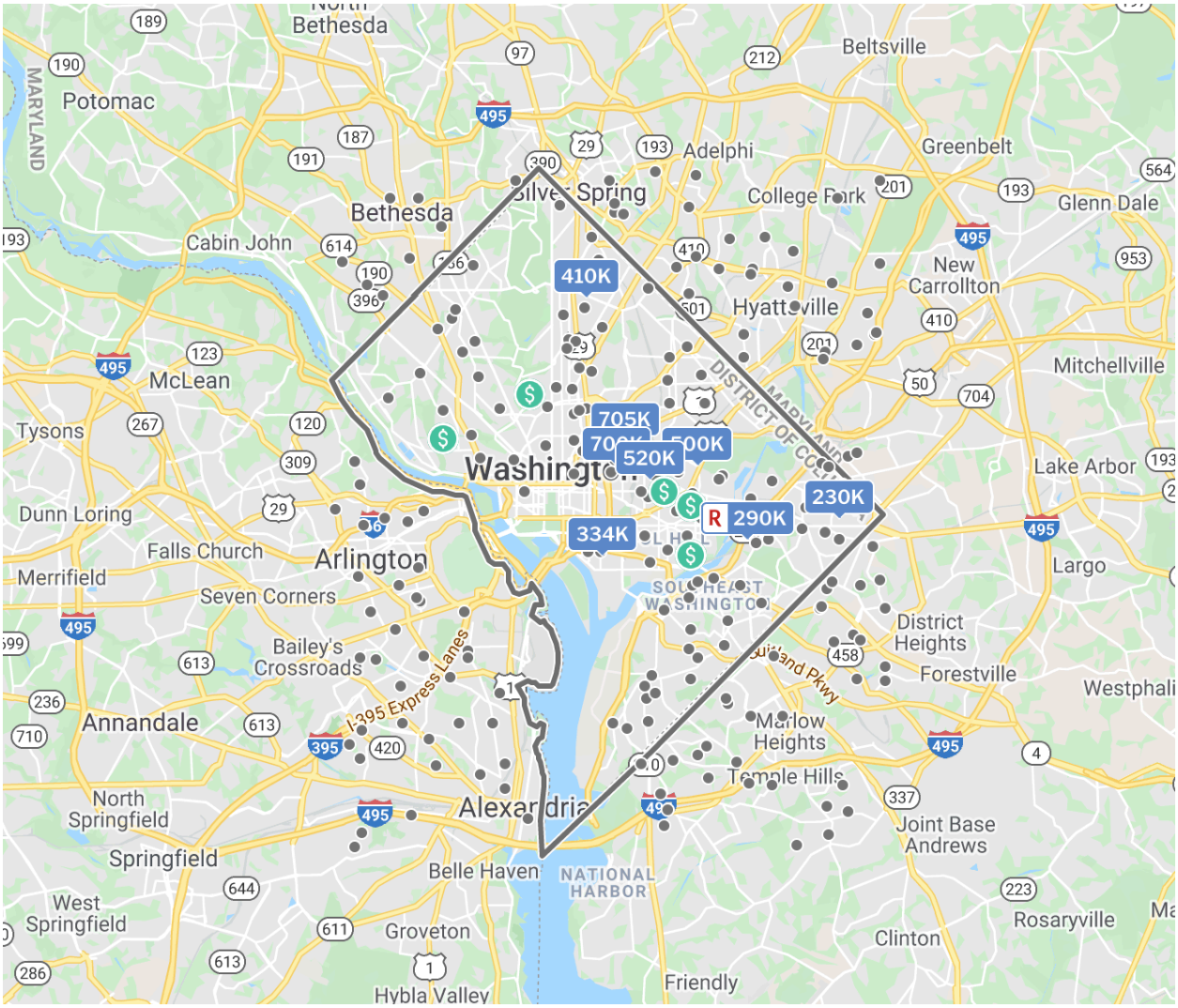
Project Report
on
Data Analysis and Modeling of DC Residential Properties

By:
Rajat Dua

Introduction



Price of Residential property in DC dependent on the location within the state. However, the median property cost in DC is about $685,000, which is more than double the national median of $325,000. One of the biggest things impacting the housing market in DC is the gross domestic product of the U.S. Another major influence on the U.S. housing market is the Federal Reserve's monetary policy. That approach to economics both signals and responds to consumer spending habits. In addition, employment also has a major impact on the housing market for any year in DC.

Objective

The objective of this project is as follows:

- Statistically analyze and interpret DC_Properties_Trimmed data.

- Perform Exploratory Data Analysis of Washington DC Residential Properties data in order to understand and interpret the impact of numerical and categorical variables on the Price Distribution of Properties in DC Metropolitan area.
- Apply different Machine Learning Algorithms in order to determine which algorithm best able to capture variance in the data and also serve as a useful tool to predict future price outcome based on current data.

Data Description

There are 46 columns and 28900 rows in the dataset. Out of the 46 variables in the dataset, there are 24 numerical variables and 22 categorical variables in the dataset.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28900 entries, 0 to 28899
Data columns (total 46 columns):
BATHRM                28900 non-null int64
HF_BATHRM             28900 non-null int64
HEAT                  28900 non-null object
AC                    28900 non-null object
NUM_UNITS             28900 non-null int64
ROOMS                 28900 non-null int64
BEDRM                 28900 non-null int64
AYB                   28900 non-null int64
YR_RMDL               28900 non-null int64
EYB                   28900 non-null int64
STORIES               28900 non-null float64
SALEDATE              28900 non-null object
PRICE                 28900 non-null int64
QUALIFIED             28900 non-null object
SALE_NUM              28900 non-null int64
GBA                   28900 non-null int64
BLDG_NUM              28900 non-null int64
STYLE                 28900 non-null object
STRUCT                28900 non-null object
GRADE                 28900 non-null object
CNDTN                 28900 non-null object
EXTWALL               28900 non-null object
ROOF                  28900 non-null object
INTWALL               28900 non-null object
KITCHENS              28900 non-null int64
FIREPLACES            28900 non-null int64
```

```
numeric_data = df.select_dtypes(include = [np.number])

categorical_data = df.select_dtypes(exclude = [np.number])

print('There are {0} numerical and {1} categorical features in the data'.\
     format(numeric_data.shape[1], categorical_data.shape[1]))
```

```
There are 24 numerical and 22 categorical features in the data
```

We are interested in analyzing the effect of variables in predicting the price of properties in the DC area. Therefore, we use 'Price' column in the data set as the dependent y variable in the dataset and all other variables are independent X variables in the dataset. Our main goal is to estimate the predicting power of certain variables on Price of the property.

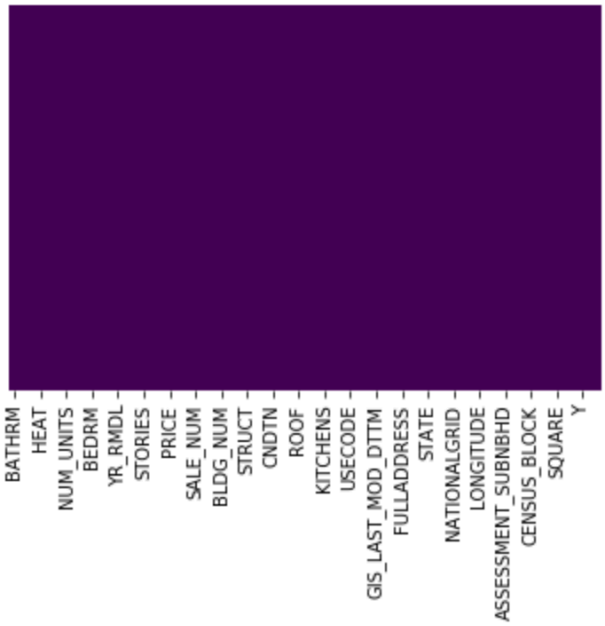
Exploratory Data Analysis

I have created multiple plots and figures on python in order to examine the relationship of multiple columns with Price variable.

Missing Data

We are interested in detecting any missing values in our dataset. For that, I have plotted a heat map which will spot any missing data. The output of my analysis is as follows:

```
: sns.heatmap(df.isnull(), yticklabels = False, cbar = False, cmap = 'viridis')
: <matplotlib.axes._subplots.AxesSubplot at 0x1a235a65c0>
```



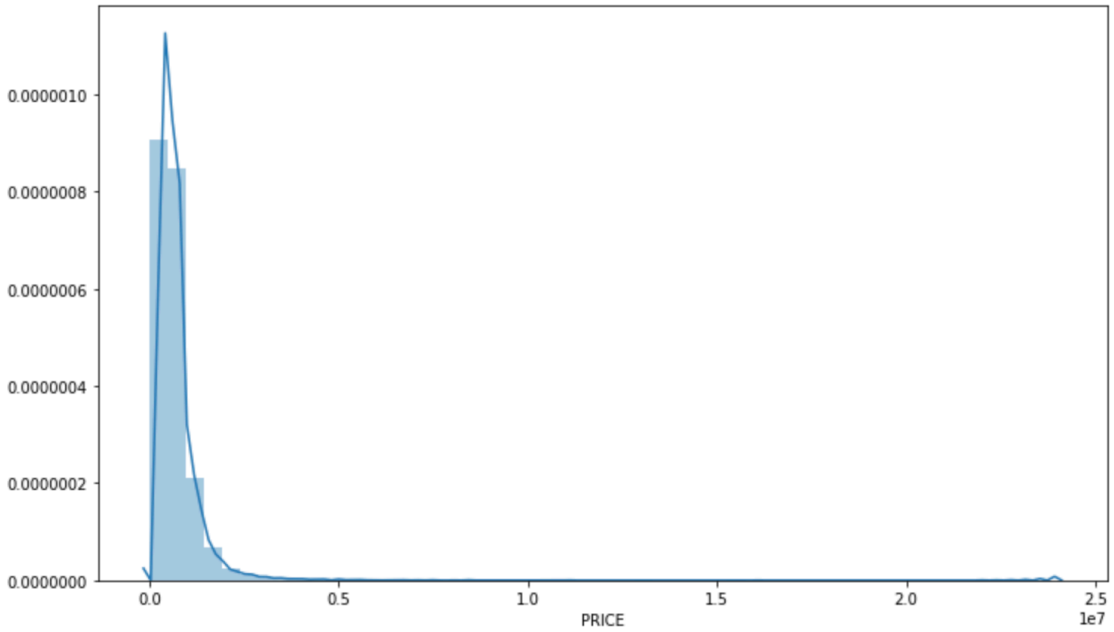
As can be clearly shown from the heatmap, there are no missing values in the dataset.

Checking the distribution of Price Column

Statistical analysis on a particular variable works best if the distribution of the variable is normally distributed. However, in many real-life situations, distributions are not normally distributed, and they need to be transformed using a function such as log transformation in order to exhibit normal distribution.

```
|: plt.figure(figsize = (12,7))
   sns.distplot(df['PRICE'])
```

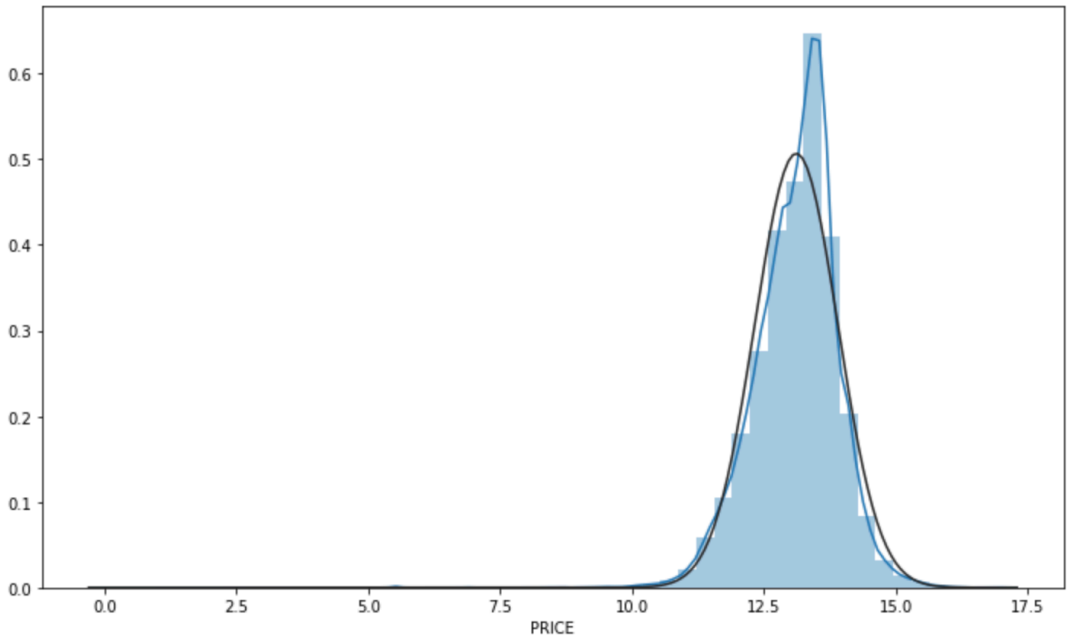
```
|: <matplotlib.axes._subplots.AxesSubplot at 0x1a2478a0f0>
```



This distance plot clearly depicts that the price variable in not normally distributed and rather heavily skewed towards the right. This suggests that majority of houses in DC area are priced closer to the median price in the area. There are a number of houses which are lower than the median price and only a few which are priced way more than the median price. This distribution is a perfect example of non-normal distribution.

```
from scipy.stats import norm
plt.figure(figsize = (12,7))
sns.distplot(np.log(df['PRICE']), fit = norm)
```

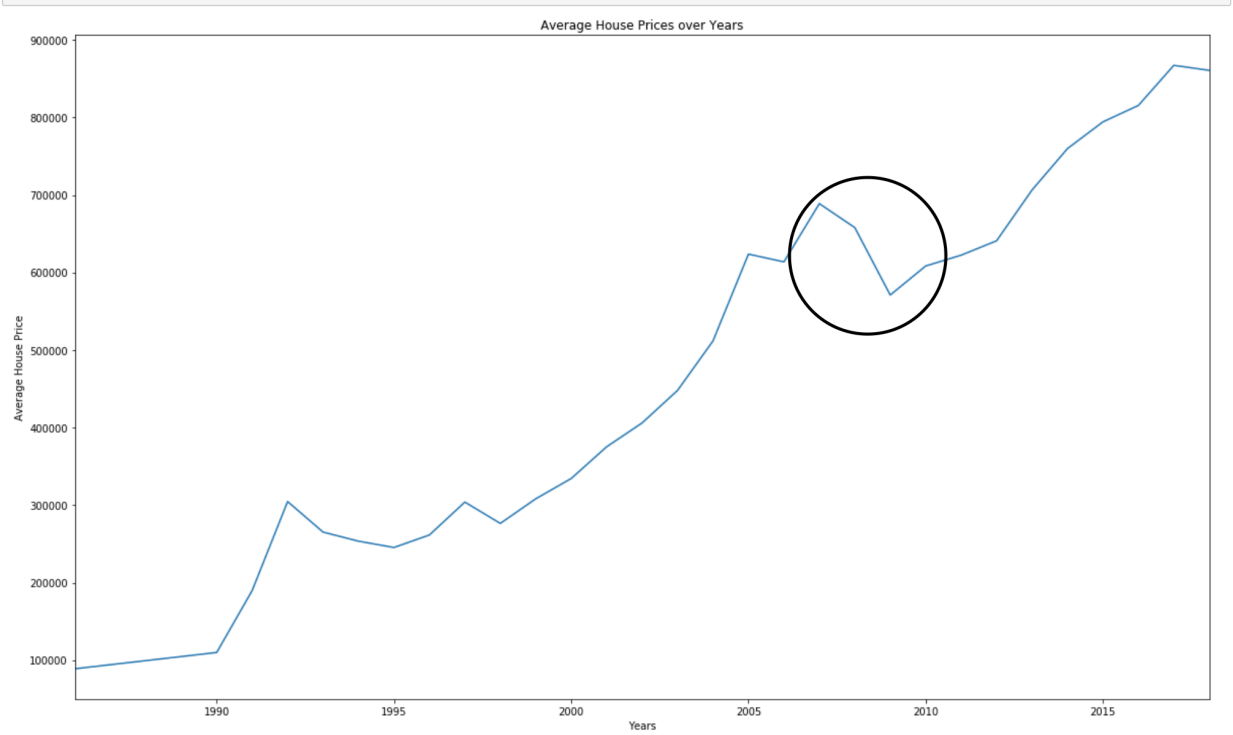
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a23794748>
```



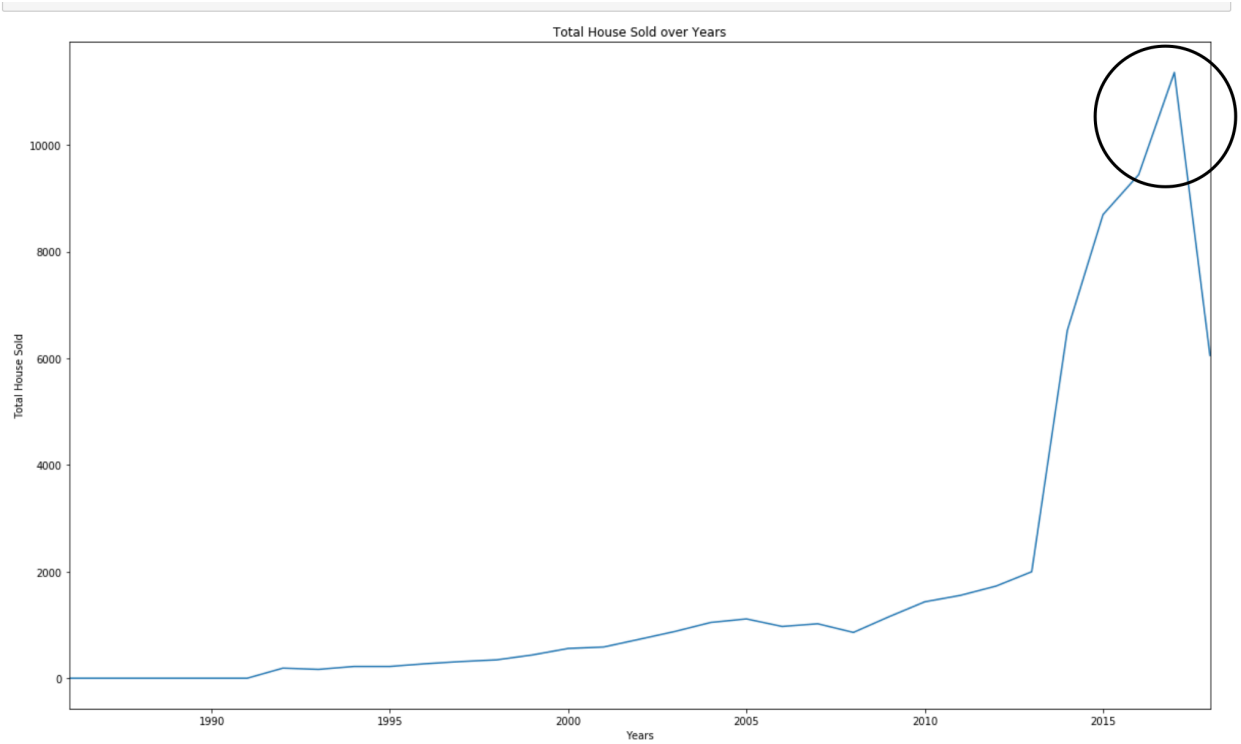
Analyzing distance plot using the log transformed price data shows that the transformed data is more likely to display normal distribution as you can see from the figure above.

Distribution of Average House Prices over Years Sold

In order to examine the relationship of Average House prices in the DC area over years, I fist converted the SALEDATE column from string to datetime variable and then extracted Months and Years from that variable and stored them as new variables: SALEMONTH and SALEYEAR respectively. I have plotted the average value of property sold in the DC area over years:



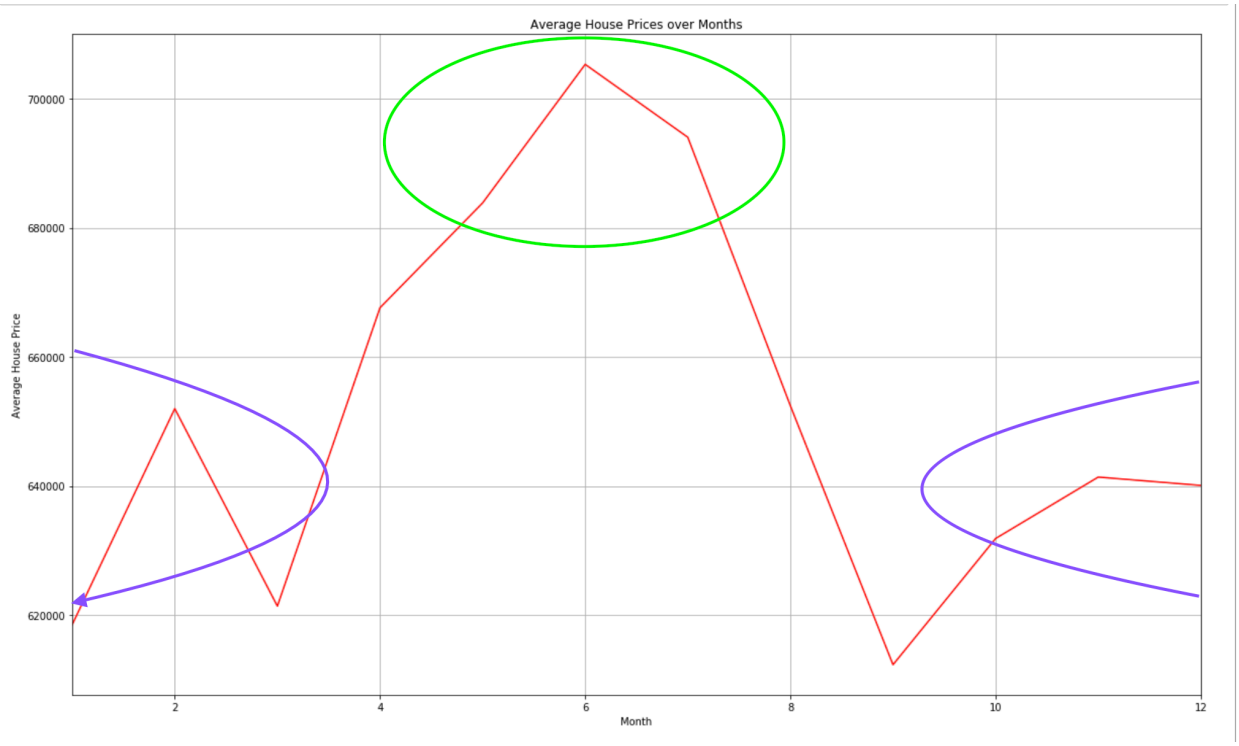
Some of the key features from this like chart is that the overall trend of average house prices in DC area is increasing. However, we could also witness a sharp decline in average house prices in the area during financial recession period from 2008 – 2009.



The above line plot shows that the number of residential properties sold over the years has an overall increasing trend, with sharp increment from 2014 – 2016 and then decline afterwards. Sharp incline may be attributed to recovery of economy post-recession period and increase in availability of affordable housing in the area.

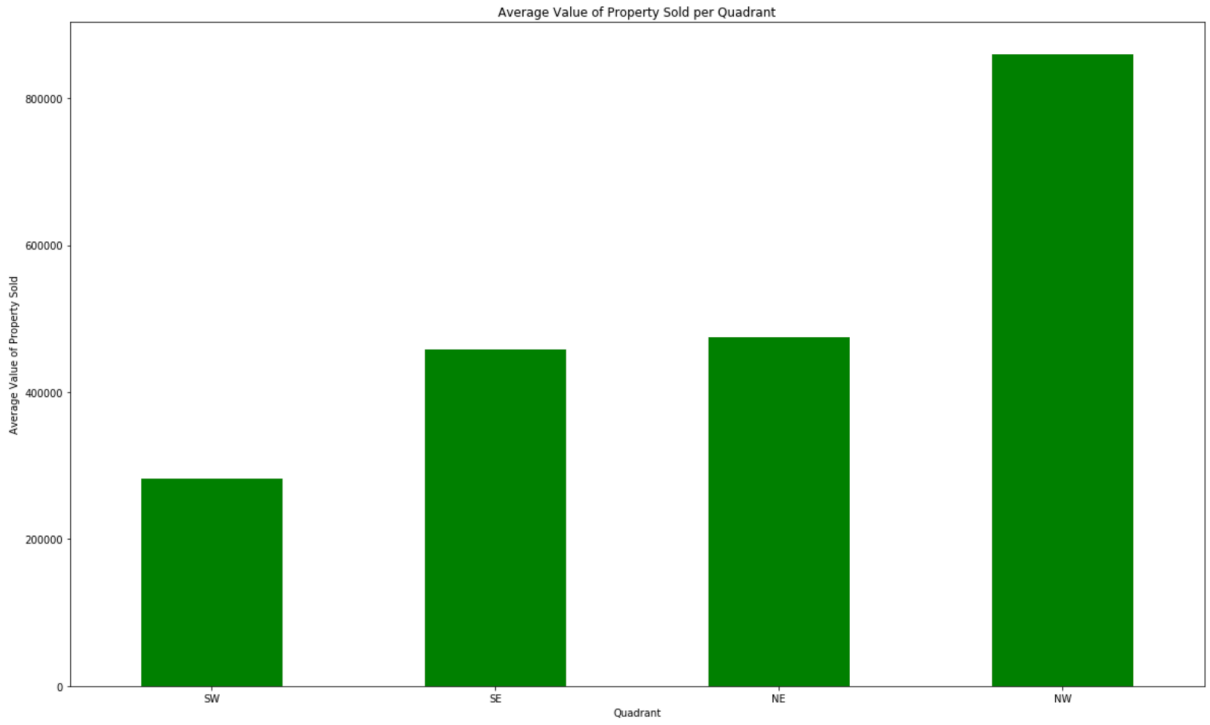
Distribution of Average House Prices over Months Sold

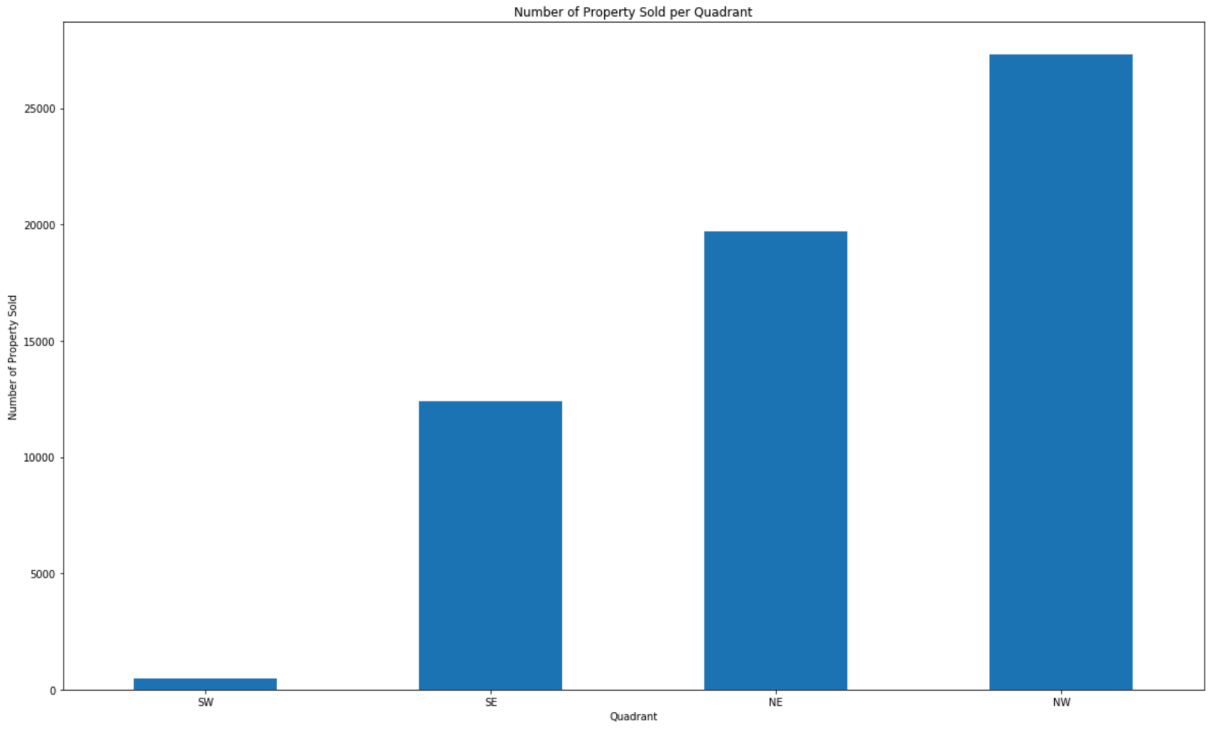
Residential real estate business is a cyclical business, that is, there are certain months when the price of residential properties booms because of rising demand and there are certain months or seasons when the market value of property falls. This relationship can be best examined by the plot showing the distribution of average house prices sold over months:



It can be clearly shown that the early summer to fall seasons are the times where the demand for real estate is really high as customer is more likely to buy property during that time. However, late Fall and Winter months from October to March are times when property rates are relatively low due to decline on demand during that period.

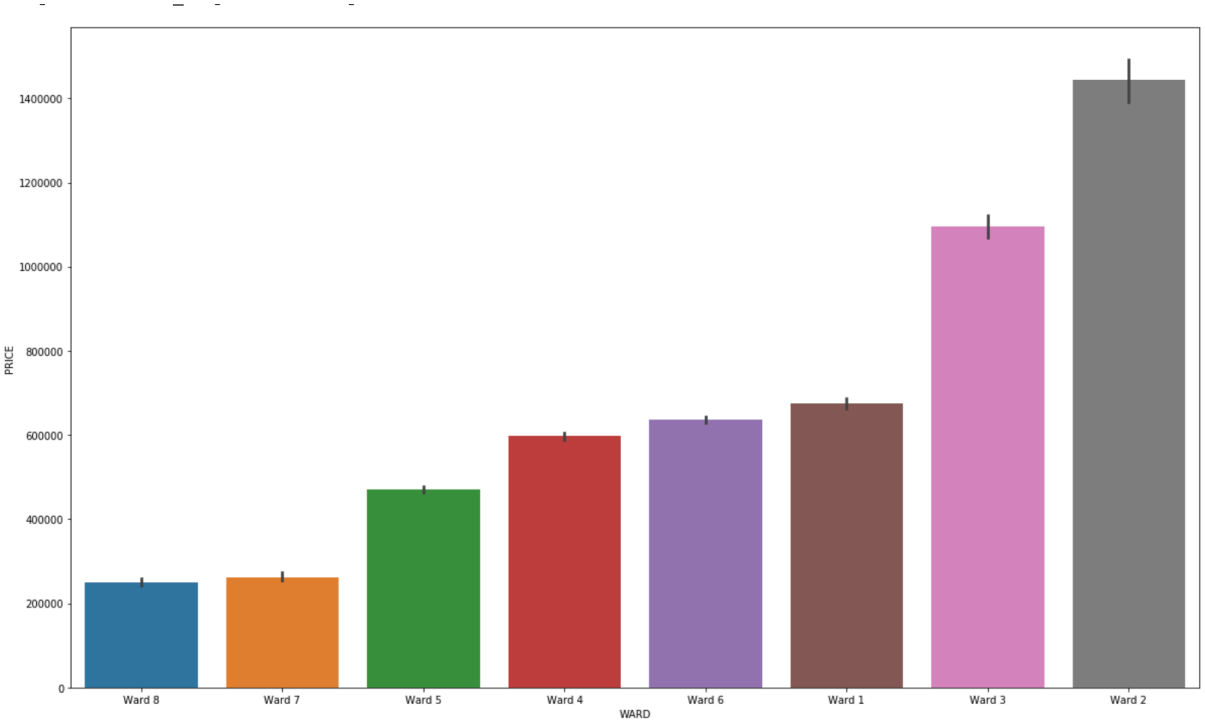
Distribution of Property Sold per Quadrant

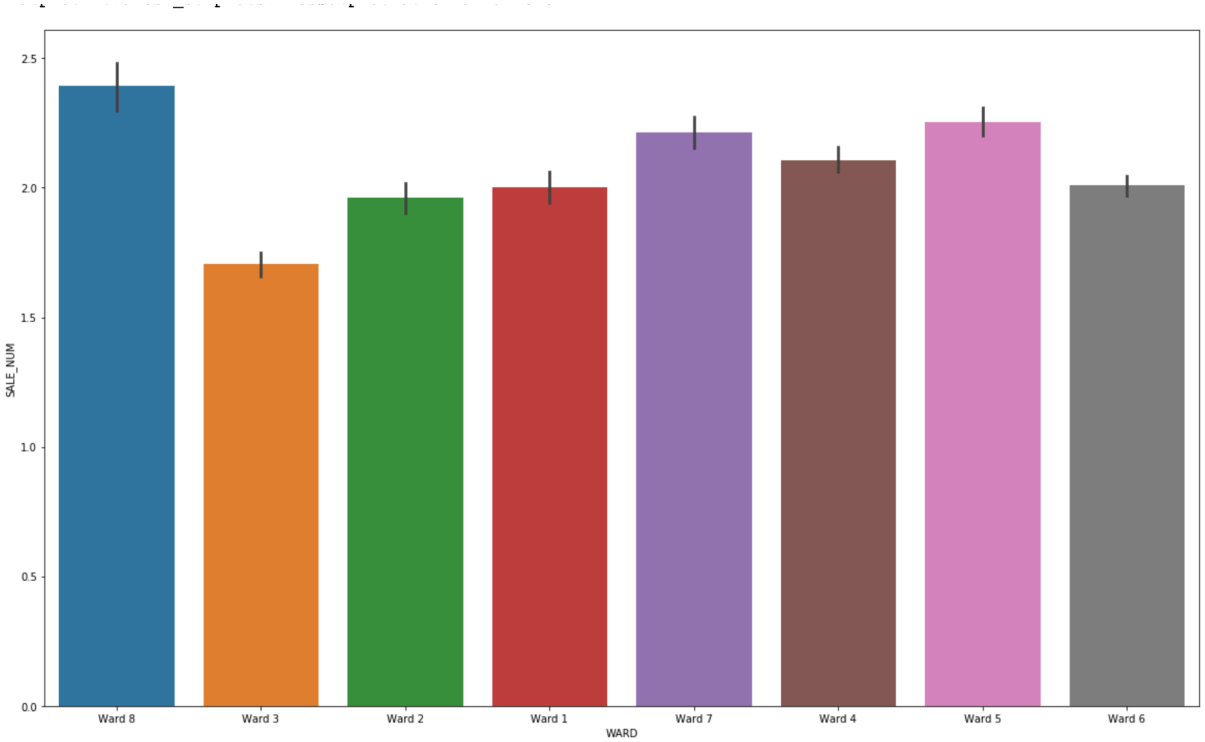




These bar plots clearly depict that NW quadrant is the region where maximum number of properties sold and one whose average price is the maximum in the state. SW region on the other hand has least number of properties sold for least average value.

Distribution of Property Sold per Wards

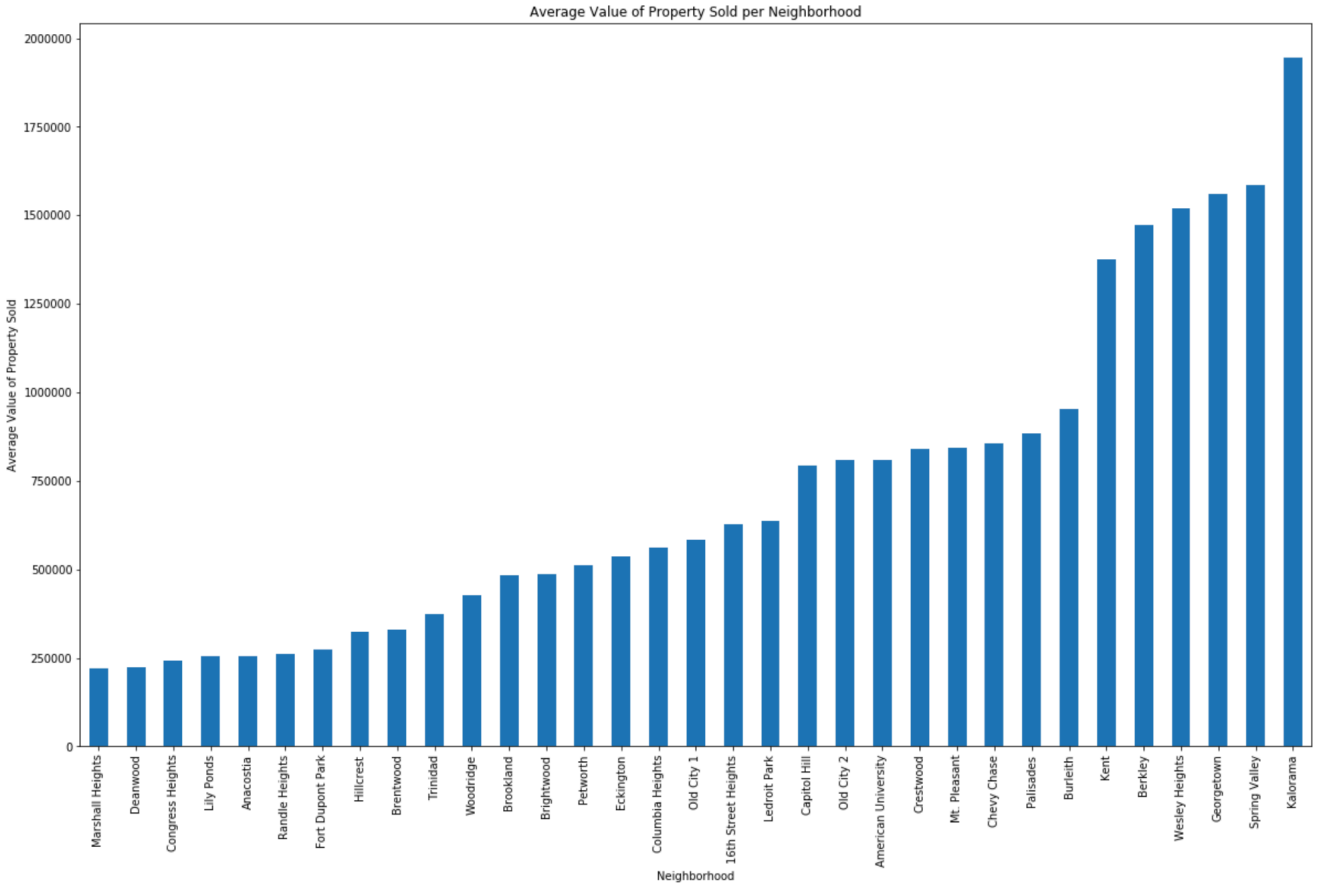


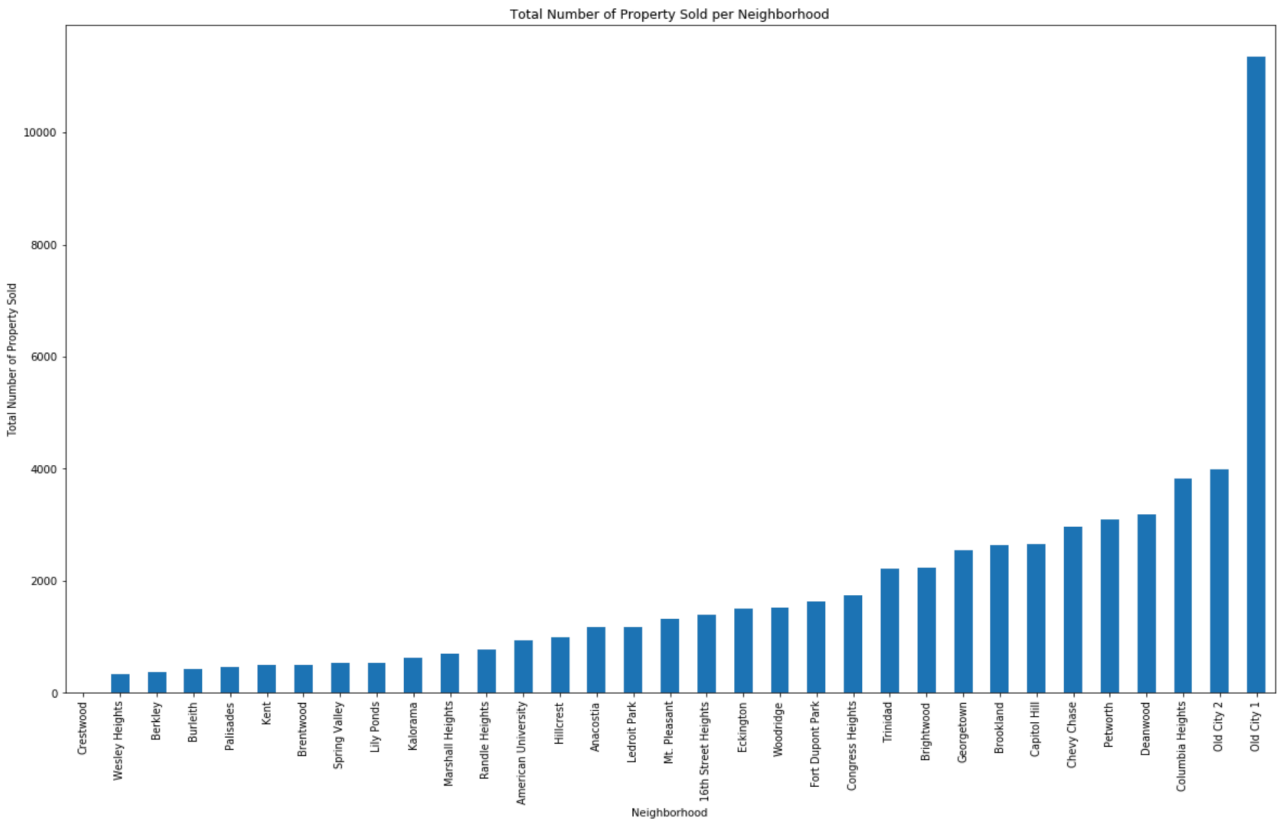


In generating these bar plots I have used seabon.barplot library. These plots display following properties:

1. Average house price of Ward 2 appears to be maximum, whereas the total number of properties sold in that district is average.
2. Maximum number of properties are sold in Ward 8, and their average prices are the least. Ward 7 and 5 are other examples where more number of properties are sold for lower average price.

Distribution of Property Sold per Neighborhood

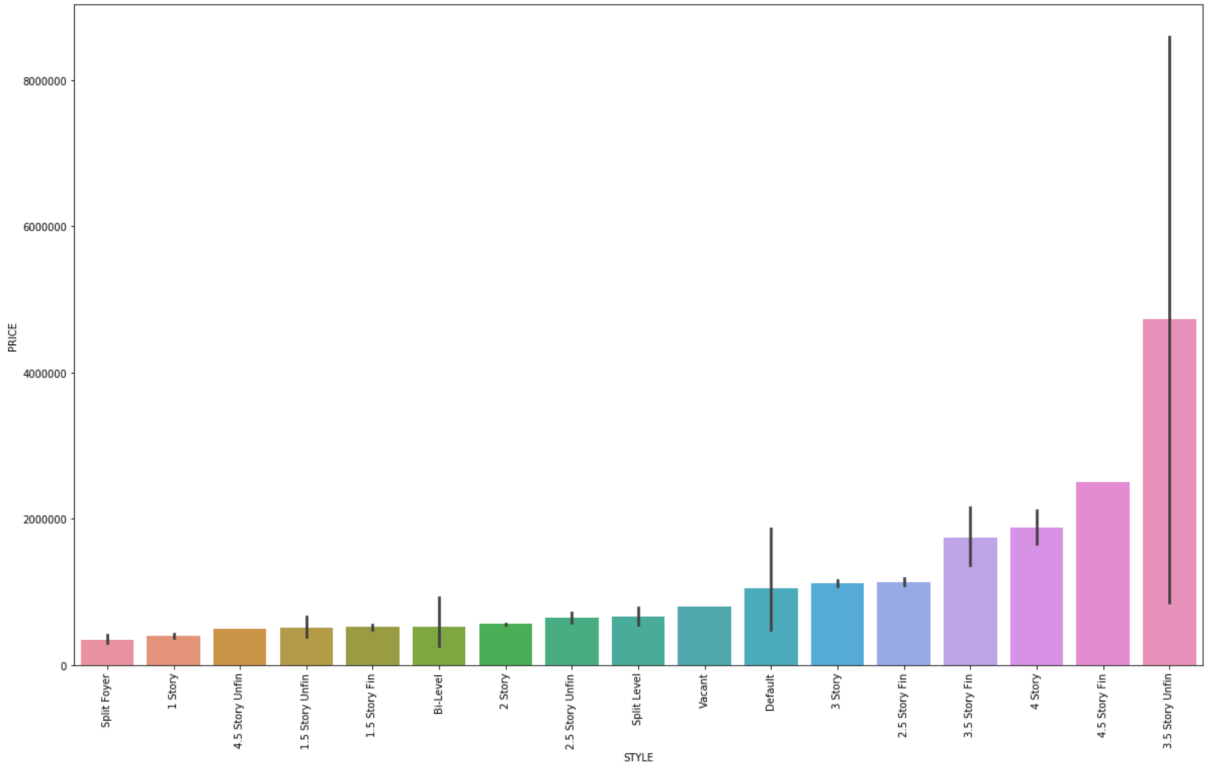


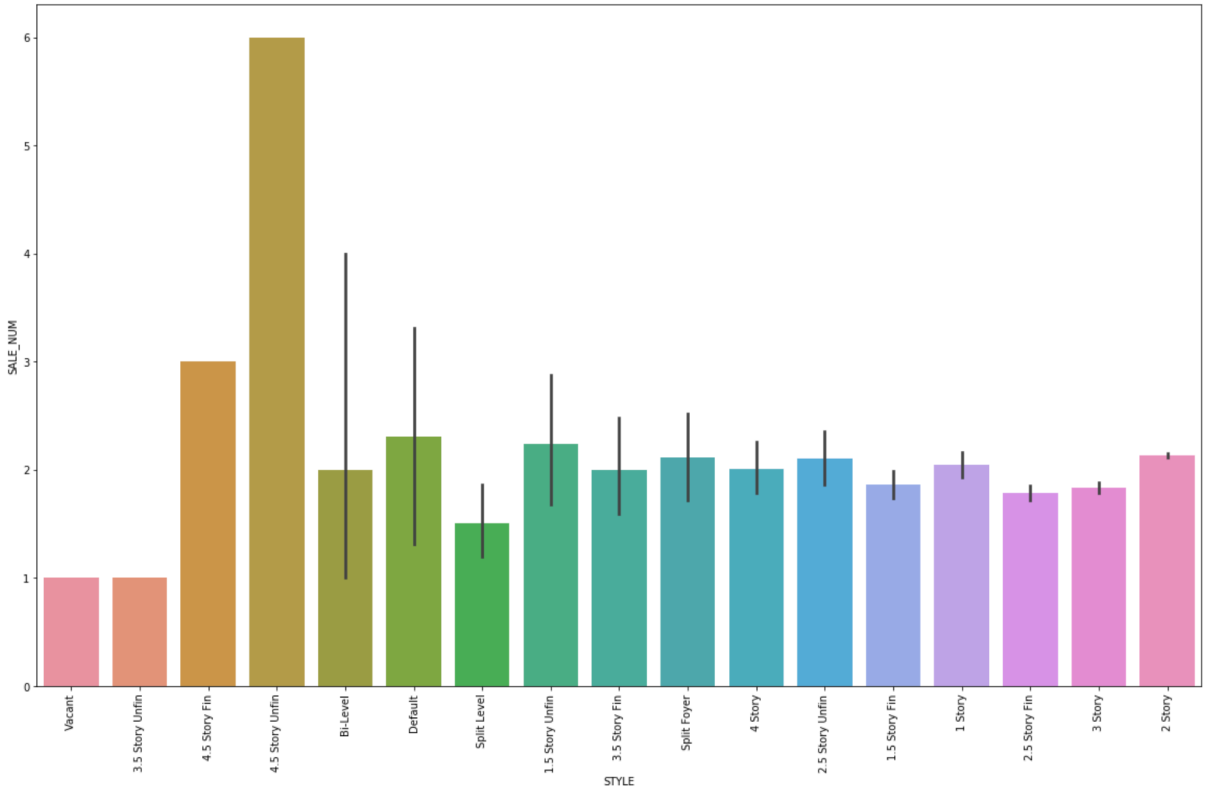


These plots display following properties:

1. Kalorama, Spring Valley, and George Town are the most expensive neighborhoods in DC to buy residential real estate. The total number of properties sold in those regions is below average.
2. Old City 1 & 2, and Columbia heights seen to attract most customers to buy residential properties in DC. The average cost of properties in those regions are around median price range.

Distribution of Property Sold per Style



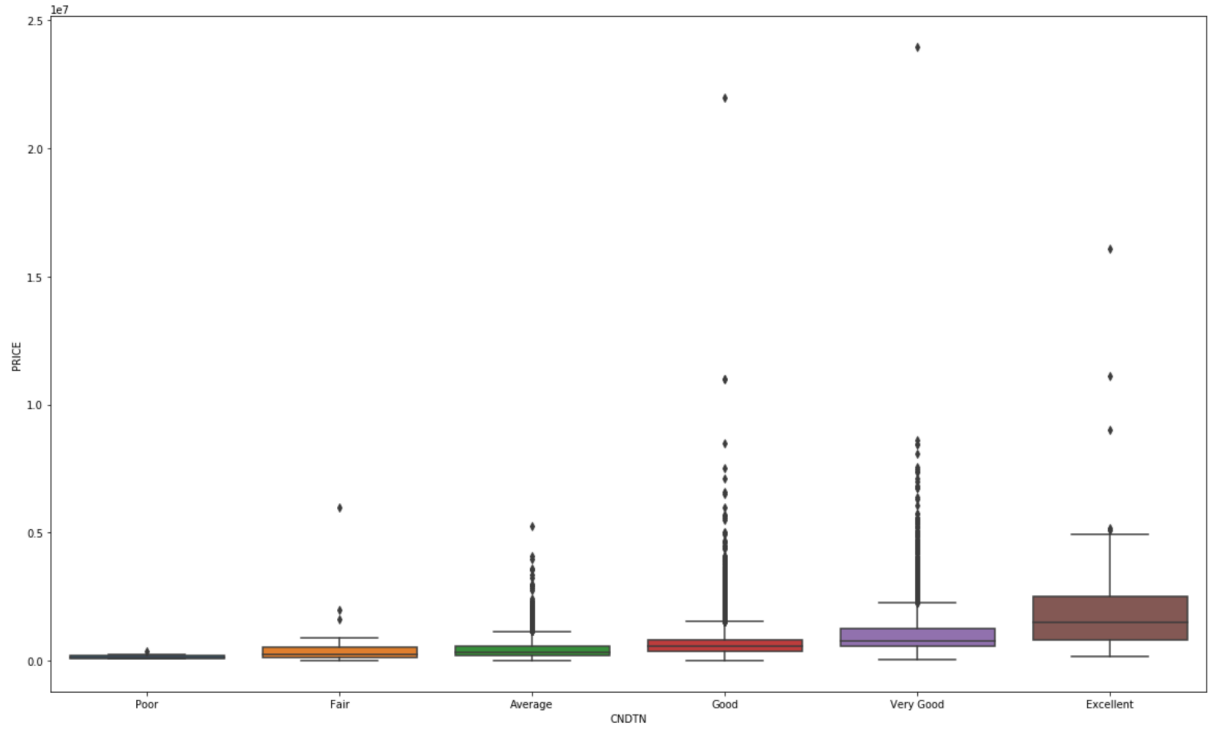


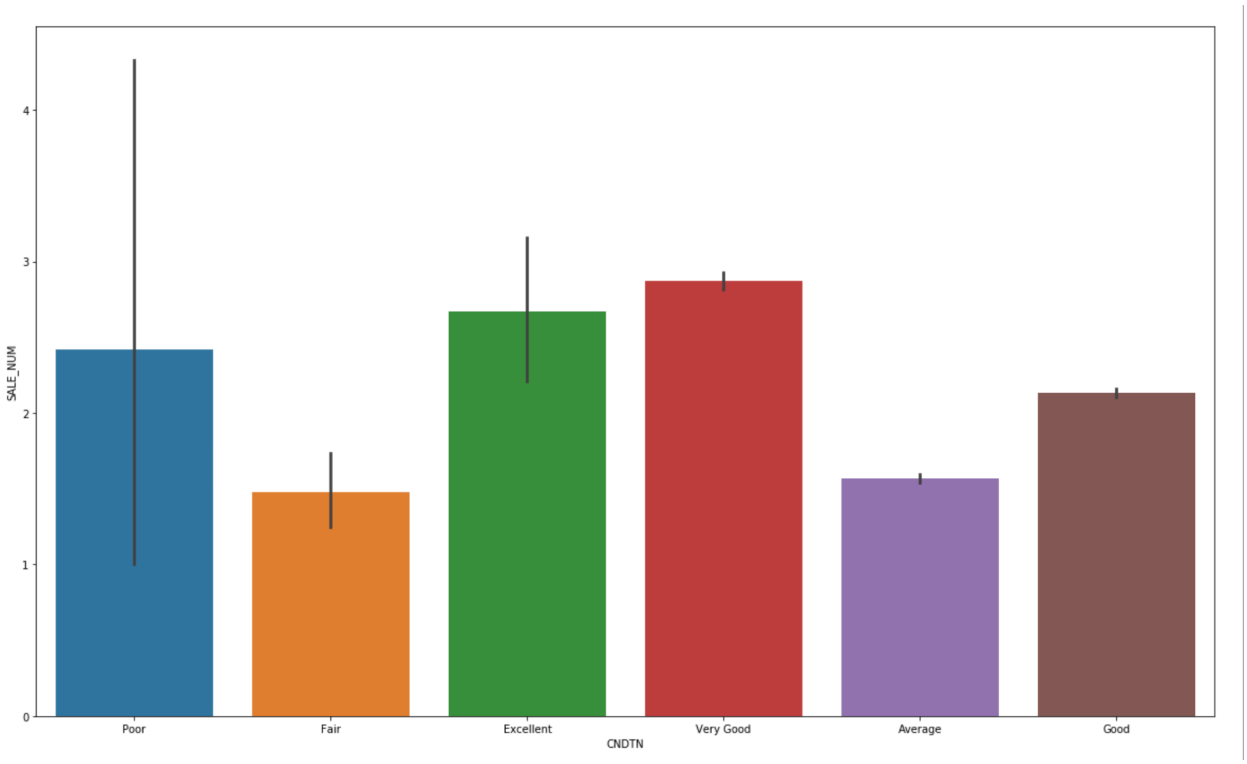
These plots display following properties:

1. 3.5 story style apartment is sold for the highest price in the DC area. Because of that, the number of such apartments sold in DC is the minimum.
2. 4.5 story unfin style apartments are the most popular amongst the buyers since they were sold the maximum and for a relatively low average price.

Distribution of Property Sold versus Condition

In order to clean the data, I have removed one row from the dataset which have condition specified as 'Default'. The rest of the data I have plotted box plot for average price and bar plot for total number of units sold.



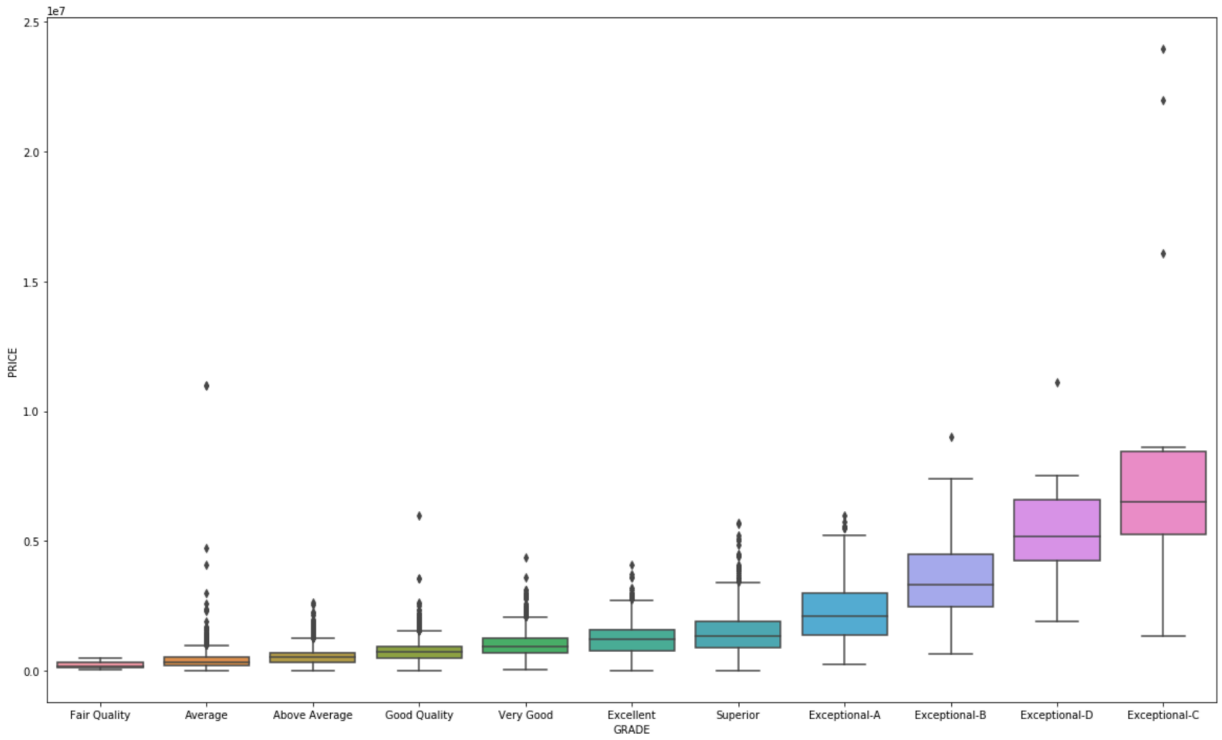


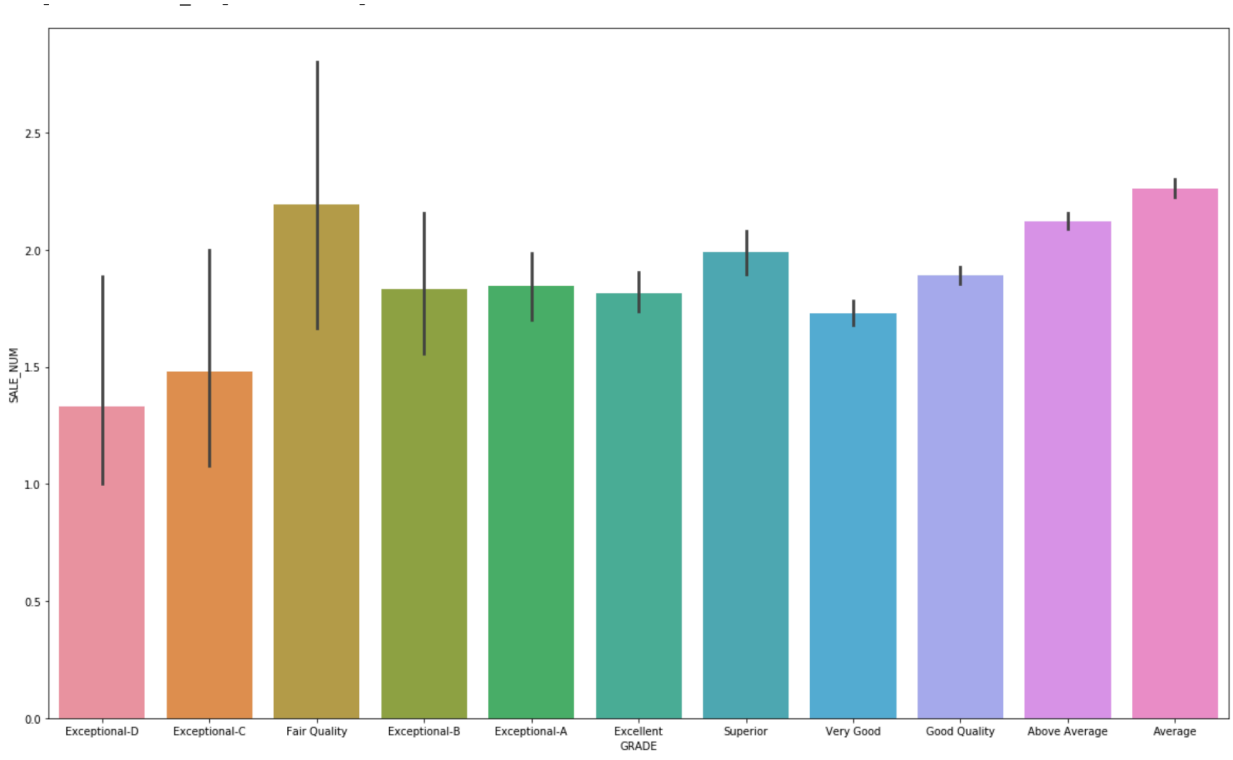
These plots display following properties:

1. The properties whose condition are Excellent are sold for the highest price whereas the properties with poor condition are sold for cheapest price.
2. The number of Very Good condition properties are sold the most followed closely by Excellent condition properties whereas Fair and Average condition properties are sold the least.

Distribution of Property Sold versus Grade

I have plotted box plot for average price and bar plot for total number of units sold.





These plots display following properties:

1. The properties whose grade is Exceptional-C are sold for the highest price on average whereas the properties with fair quality grade are sold for cheapest price.
2. The number of Fair Quality grade properties are sold the most followed closely by Average grade properties whereas Exceptional-D and Exceptional-C grade properties are sold the least.

Condition Mapping

```
Map_Cond = {"Poor": 1, "Fair": 2, "Average": 3, "Good": 4, "Very Good": 5, "Excellent": 6}

df.CNDTN = df.CNDTN.apply(lambda x: Map_Cond[x])
```

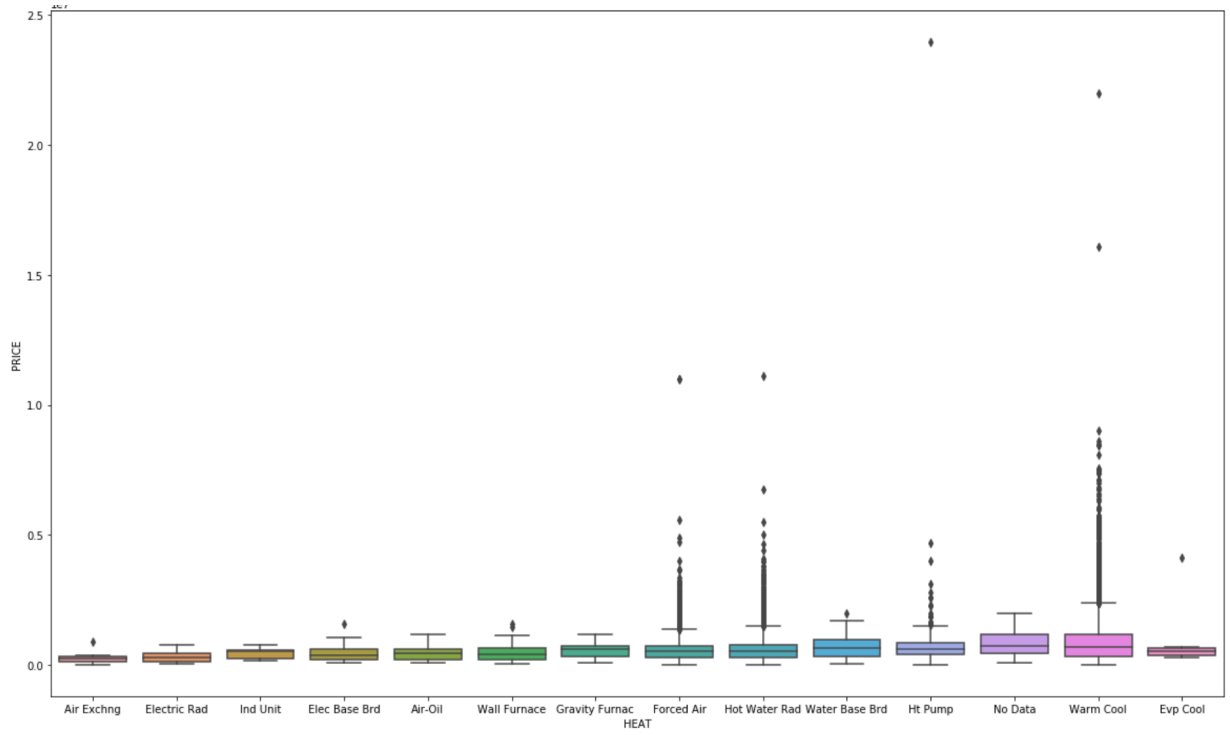
Grade Mapping

```
Map_Grade = {"Fair Quality": 1, "Average": 2, "Above Average": 3, "Good Quality": 4, "Very Good": 5, "Excellent": 6,
             "Superior": 7, "Exceptional-A": 8, "Exceptional-B": 9, "Exceptional-C": 10, "Exceptional-D": 11}

df.GRADE = df.GRADE.apply(lambda x: Map_Grade[x])
```

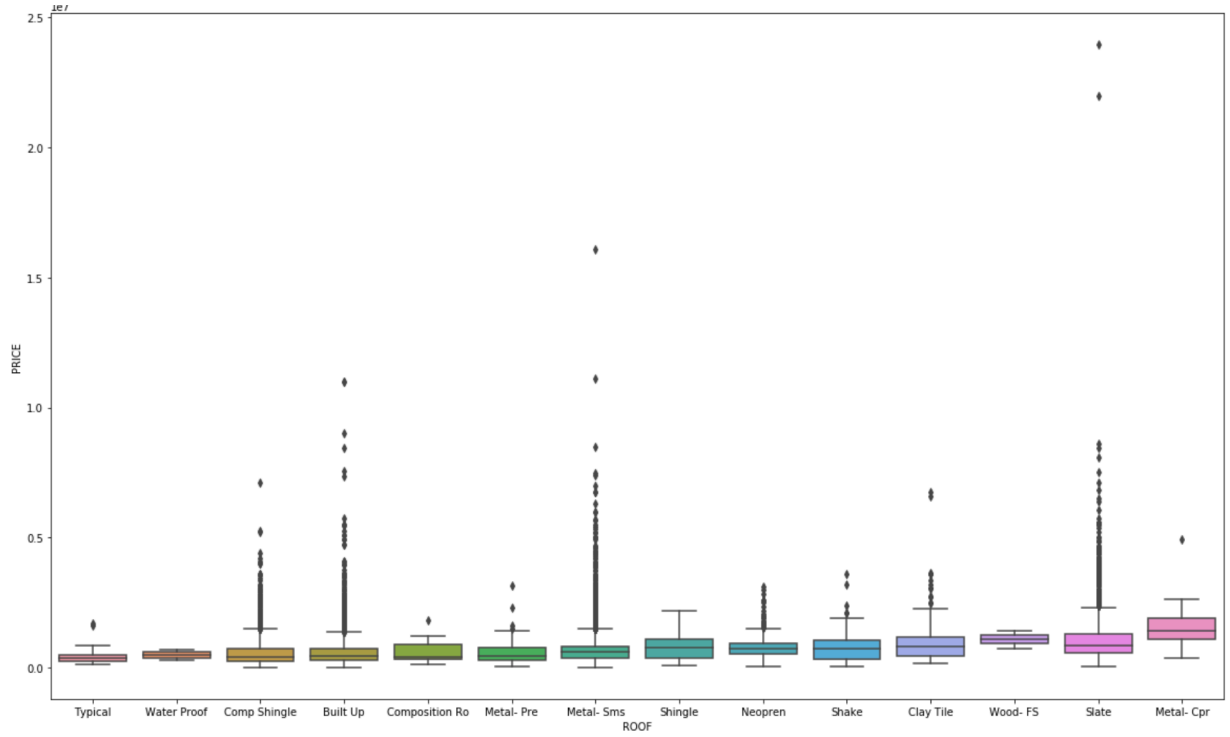
I have mapped condition and grade with numerical attributes in order to convert these columns from categorical to numerical variables.

Distribution of Property Sold versus Heat



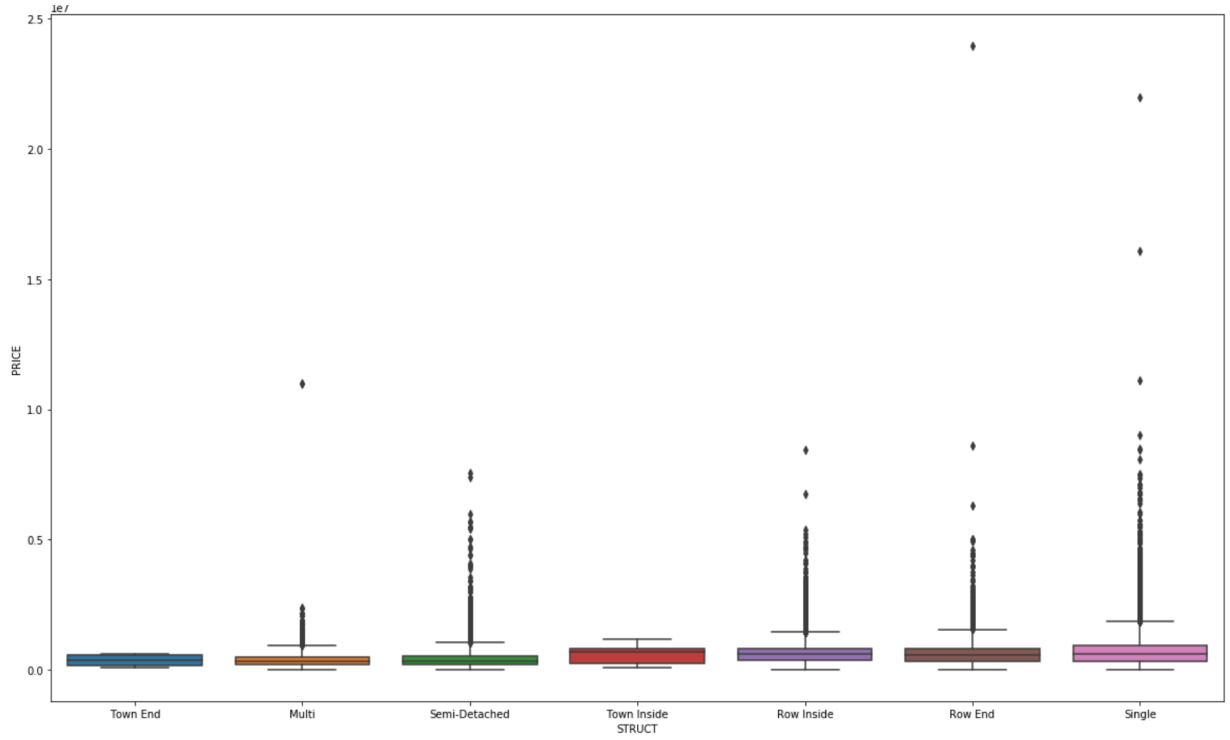
The availability and quality of heating and cooling units in the property does not have high correlation with the price of the property. They might impact the price, since houses with installation of warm cool heating units seem have a higher price on average, but not by higher margin. Therefore, Heat column does not provide much information about the price of property.

Distribution of Property Sold versus Roof



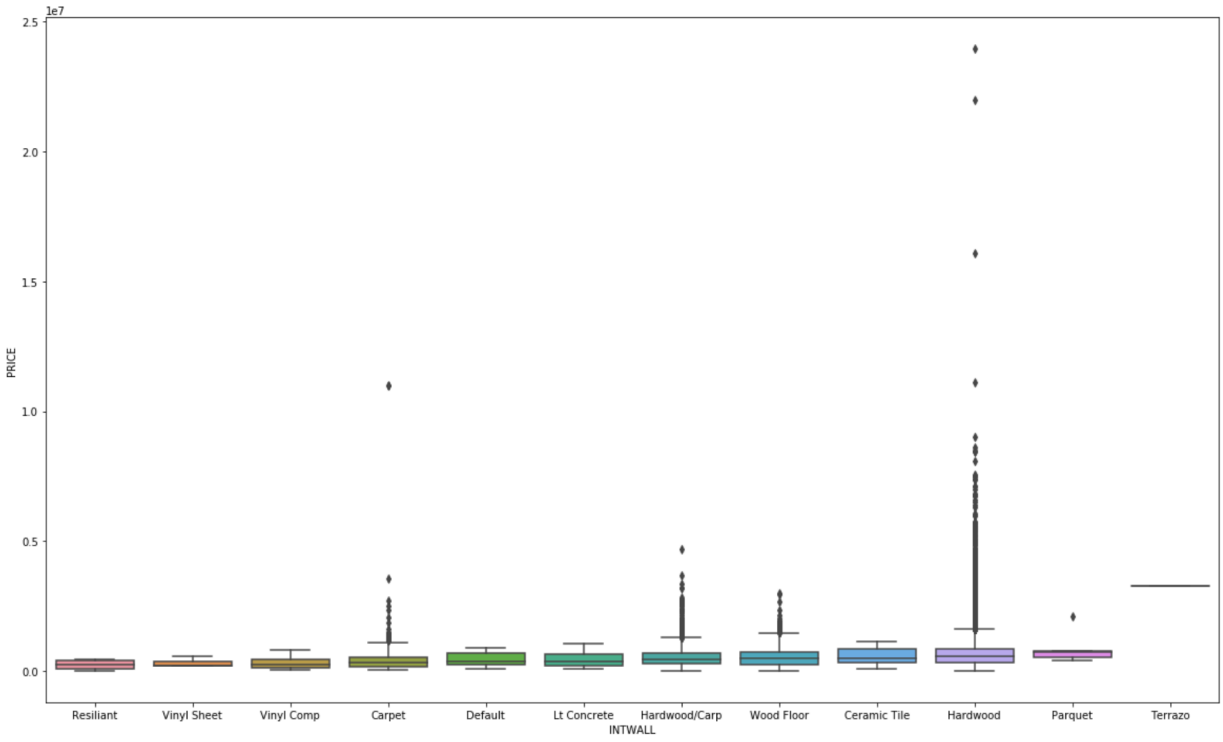
Houses with Metal-Cpr roofs have the highest median price, whereas houses with Typical roofs have the lowest median price. Houses with Metal-Sms and Slate roofs have the highest variability in price range owing to the presence of high number of outliers as depicted in the boxplot above.

Distribution of Property Sold versus Structure



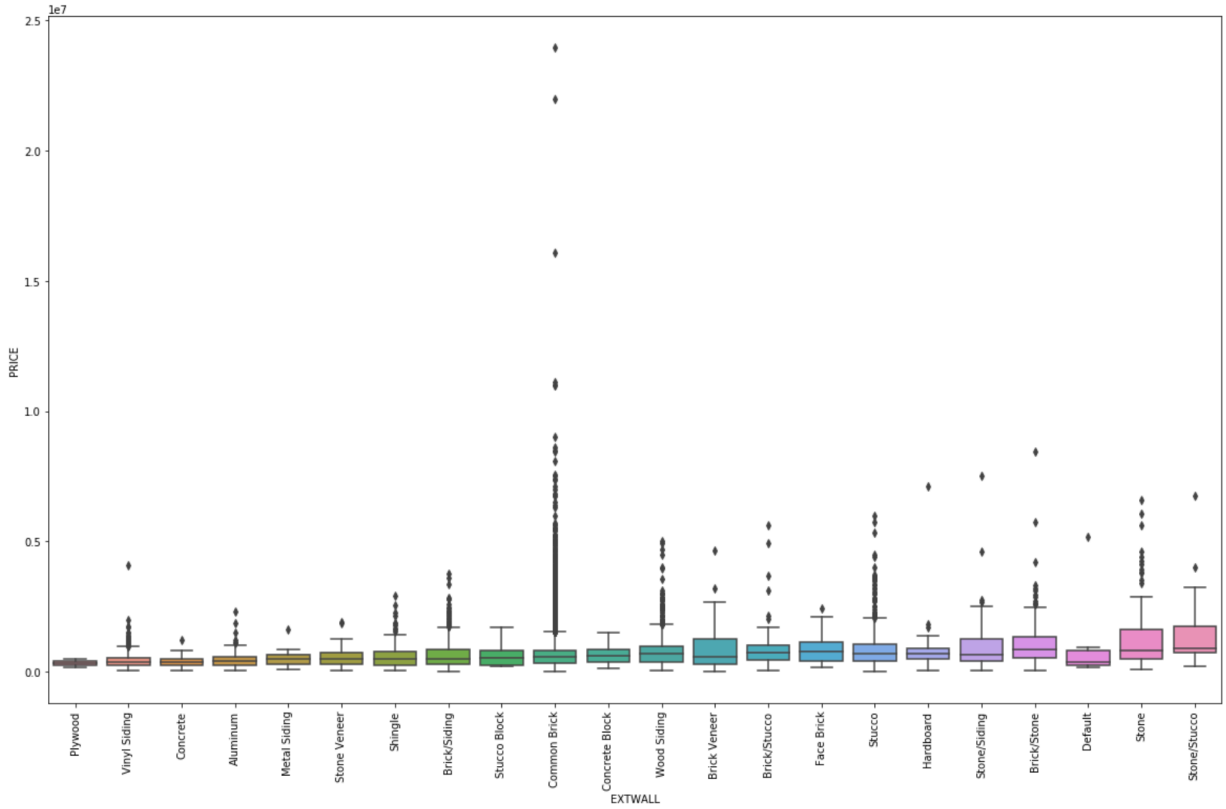
The availability and quality of structure does not have high correlation with the price of the property. They might impact the price a little since the houses with Single structure have the highest variability in price range owing to the presence of high number of outliers as depicted in the boxplot above. Therefore, Structure column does not provide much information about the price of property.

Distribution of Property Sold versus Interior Wall



The availability and quality of interior wall does not have high correlation with the price of the property. They might impact the price a little since the houses with Hardwood wall have the highest variability in price range owing to the presence of high number of outliers as depicted in the boxplot above. Therefore, Interior Wall column does not provide much information about the price of property.

Distribution of Property Sold versus Exterior Wall



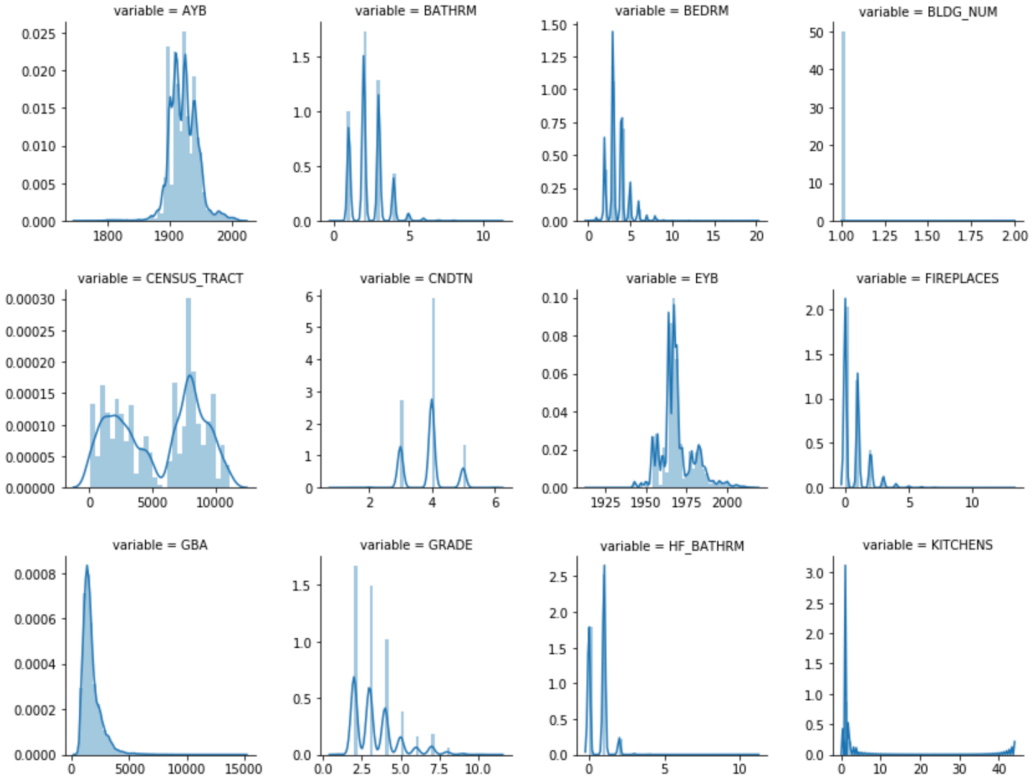
Houses with Stone/Stucco exterior walls have the highest median price, whereas houses with Plywood Exterior Walls have the lowest median price. Houses with Common Brick exterior walls have the highest variability in price range owing to the presence of high number of outliers as depicted in the boxplot above.

Feature Selection

Handling Numerical Variables

In order to determine the predictive power of numerical variables, I plot a facet grid of distance plots of each numerical variables in the dataset, as follows:

```
f = pd.melt(df, value_vars = sorted(numeric_data))
g = sns.FacetGrid(f, col = 'variable', col_wrap=4, sharex = False, sharey = False)
g = g.map(sns.distplot, 'value')
```

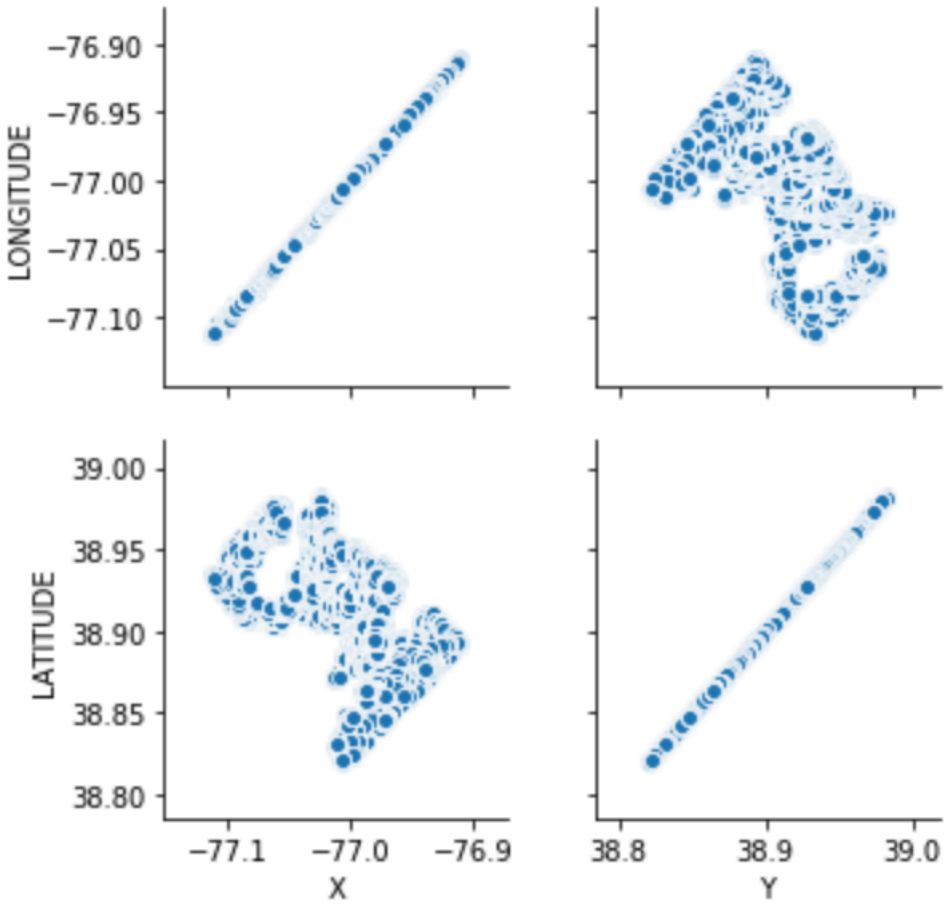


From these plots, I select the variables which are continuous as the numerical variables in the dataset and rest of the variables which are discrete, I convert them to string variables, as follows:

```
df['AYB'] = df.AYB.apply(lambda x: str(x))
df['EYB'] = df.EYB.apply(lambda x: str(x))
df['YR_RMDL'] = df.YR_RMDL.apply(lambda x: str(x))
df['ZIPCODE'] = df.ZIPCODE.apply(lambda x: str(x))
df['SALEYEAR'] = df.SALEYEAR.apply(lambda x: str(x))
df['SALEMONTH'] = df.SALEYEAR.apply(lambda x: str(x))
```

Variables such as AYB, EYB, YR_RMDL, ZIPCODE, SALEYEAR, and SALEMONTH are discreate variables and I, therefore, converted them to string or categorical variables in the dataset.

In addition, there are some variables in the dataset which are highly correlated to each other and also convey the same information. For example, numerical variables X and Y in the dataset is highly correlated and similar to LONGITUDE and LATITUDE columns in the dataset. I checked their relationship using the pair plot as below:

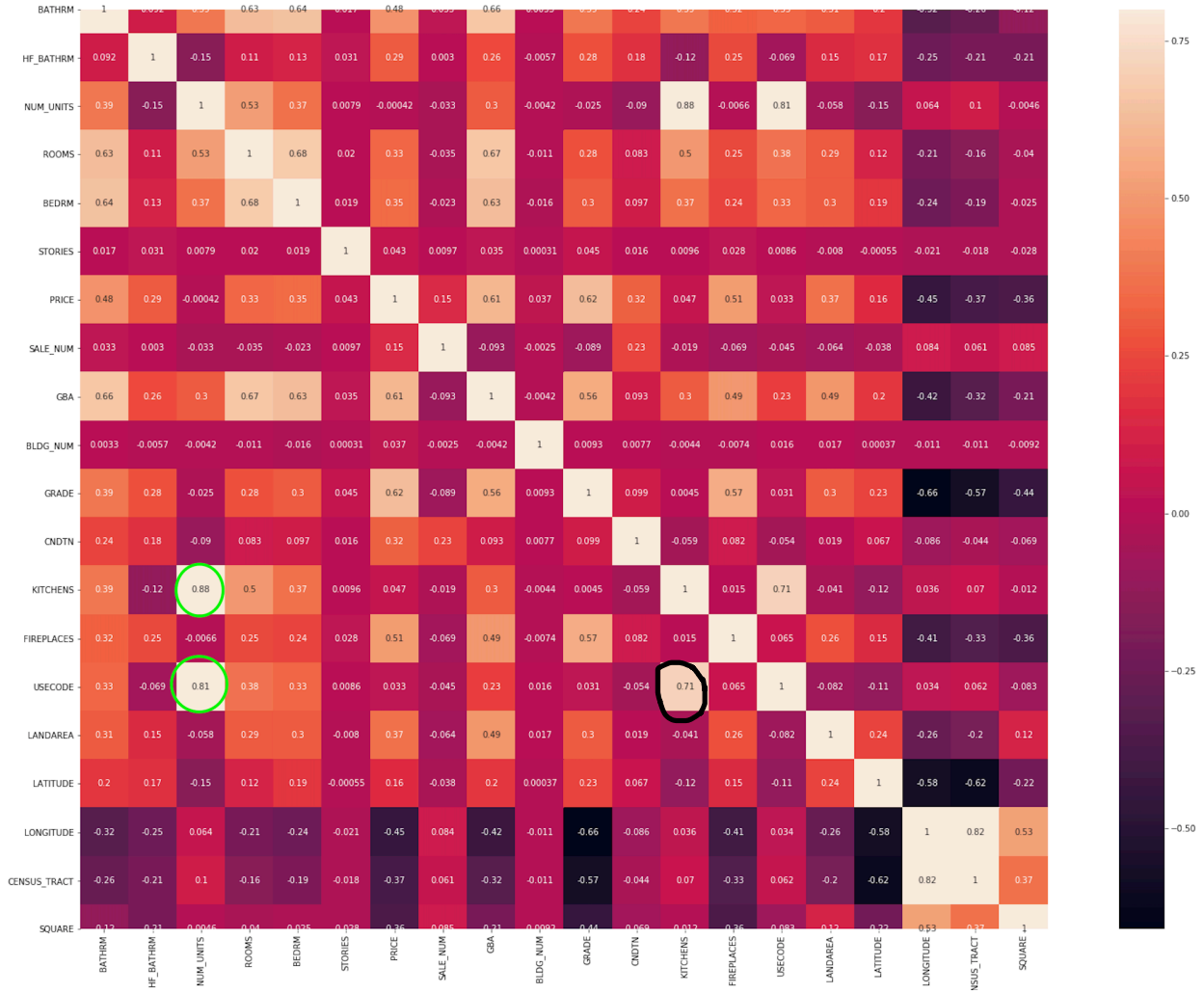


Using these plots, I have dropped columns X and Y from the data set since including them can cause the problem of multicollinearity.

	STYLE	STORIES
5849	2 Story	2.0
20189	2 Story	2.0
10737	2 Story	2.0
19858	2 Story	2.0
16563	2 Story	2.0

Next, I have analyzed that both Style and Stories columns convey similar information, so I have decided to drop STYLE column from the dataset.

Correlation Between Numerical Variables



From this correlation matrix, it can be clearly seen that both kitchen and usecode columns are highly correlated to num_units column and kitchen and usecode are also highly correlated to each other. Therefore, I have dropped usecode column from the dataset.

```
cols_to_drop = ['SALEDATE', 'AYB', 'EYB', 'BLDG_NUM', 'GIS_LAST_MOD_DTTM', 'SOURCE', 'FULLADDRESS', 'CITY',
                'STATE', 'ZIPCODE', 'NATIONALGRID', 'LATITUDE', 'LONGITUDE', 'ASSESSMENT_SUBNBHD', 'CENSUS_TRACT',
                'CENSUS_BLOCK', 'SQUARE']
df.drop(columns = cols_to_drop, inplace = True)
```

Furthermore, I have recognized certain columns which are mentioned above that does not contribute much in determining the price of residential house in DC. These variables are mostly location variables which are common for majority of dataset and also some redundant variables

such as SALEDATE. These variables do not capture much in terms of determining the pricing of residential real estate. Therefore, I have dropped these columns from the dataset.

Normalizing Numerical Data

I have used MinMaxScaler from sklearn.preprocessing library in python in order to normalize the remaining numerical variables in the dataset. The output of Normalizing numerical data is as follows:

	BATHRM	HF_BATHRM	NUM_UNITS	ROOMS	BEDRM	STORIES	SALE_NUM	GBA	GRADE	CNDTN	KITCHENS	FIREPLACES	LANDAREA
0	0.363636	0.000000	0.333333	0.258065	0.20	0.003632	0.000000	0.145271	0.4	0.6	0.045455	0.384615	0.009403
1	0.272727	0.090909	0.333333	0.290323	0.25	0.003632	0.142857	0.145271	0.4	0.8	0.045455	0.307692	0.009403
2	0.272727	0.090909	0.333333	0.258065	0.25	0.003632	0.000000	0.142661	0.4	0.6	0.045455	0.230769	0.009403
3	0.272727	0.090909	0.333333	0.258065	0.20	0.003632	0.000000	0.136960	0.4	0.4	0.045455	0.076923	0.009063
4	0.272727	0.090909	0.333333	0.225806	0.15	0.002421	0.214286	0.074250	0.2	0.8	0.045455	0.076923	0.007759

Handling Categorical Variables

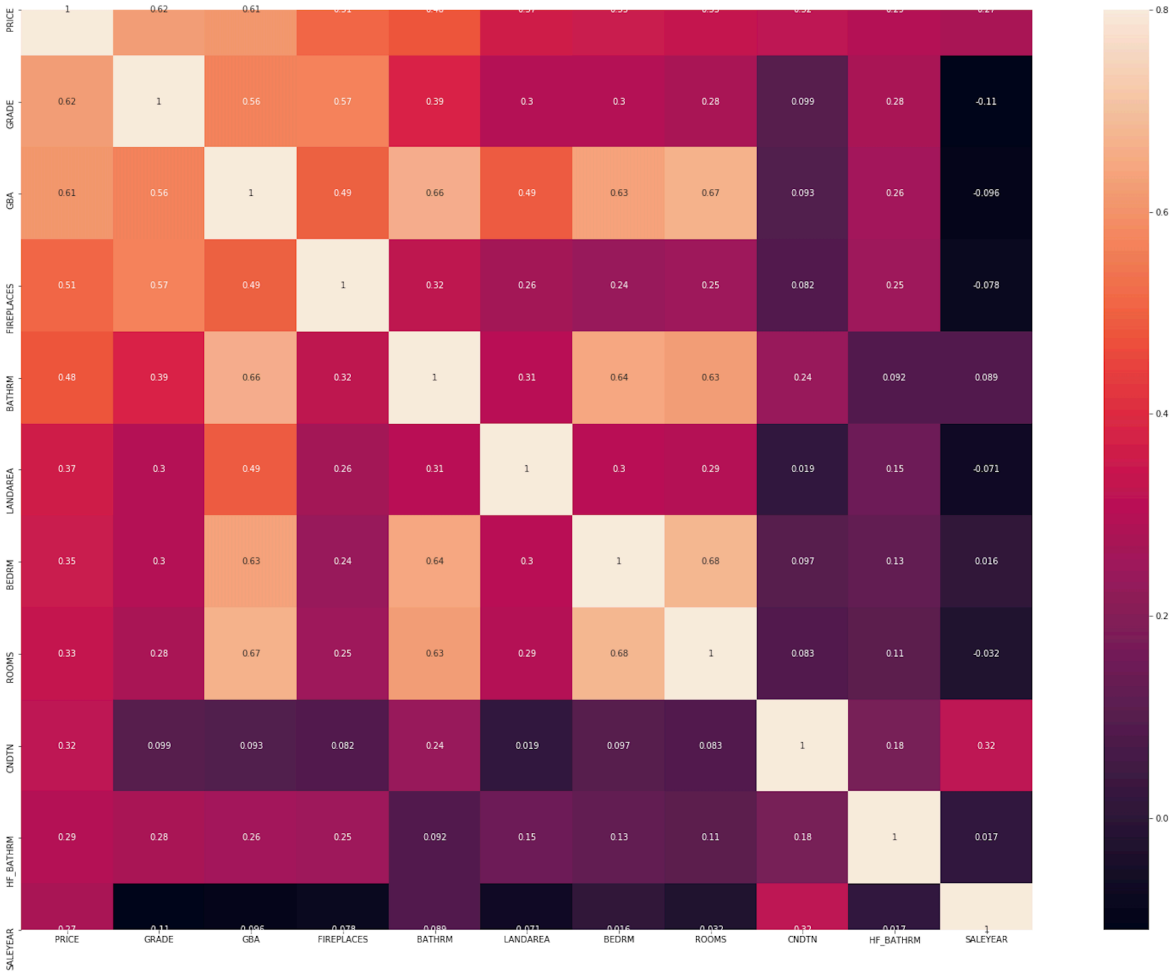
I have used LabelEncoder from sklearn.preprocessing library in python in order to convert the remaining categorical variables in the dataset to numerical variables so that they can be easily processed through various machine learning algorithms for modeling. The final dataset after concatenating numerical and categorical processed variables is as follows:

	BATHRM	HF_BATHRM	NUM_UNITS	ROOMS	BEDRM	STORIES	SALE_NUM	GBA	GRADE	CNDTN	...	QUALIFIED	STRUCT	EXTWALL	ROOF	INT
0	0.363636	0.000000	0.333333	0.258065	0.20	0.003632	0.000000	0.145271	0.4	0.6	...	0	2	5	6	
1	0.272727	0.090909	0.333333	0.290323	0.25	0.003632	0.142857	0.145271	0.4	0.8	...	0	2	5	0	
2	0.272727	0.090909	0.333333	0.258065	0.25	0.003632	0.000000	0.142661	0.4	0.6	...	0	2	5	0	
3	0.272727	0.090909	0.333333	0.258065	0.20	0.003632	0.000000	0.136960	0.4	0.4	...	0	2	5	6	
4	0.272727	0.090909	0.333333	0.225806	0.15	0.002421	0.214286	0.074250	0.2	0.8	...	0	2	5	0	

Top 10 Correlated Variables with Price Variable

```
cols = df.corr().nlargest(11, 'PRICE')['PRICE'].index

plt.figure(figsize = (30,20))
sns.heatmap(df[cols].corr(), vmax=0.8, annot=True, square = True)
plt.savefig('Corr.png')
```



In this plot, I have determined top 10 correlated variables in the dataset from both numerical and categorical variables. There variables, which are clearly shown in the plot, has high predictive power over price of residential real estate in the DC area. Therefore, I will use these 10 variables as independent X variables and Price variable as dependent y variable for modeling testing and training application.

Test Train Data

I have created a new dataset which is the subset of main dataframe, such that it contains Top 10 correlated variables with Price variable and also Price variable. The dataset is as follows:

```
df[cols].head()
```

	PRICE	GRADE	GBA	FIREPLACES	BATHRM	LANDAREA	BEDRM	ROOMS	CNDTN	HF_BATHRM	SALEYEAR
0	1095000	0.4	0.145271	0.384615	0.363636	0.009403	0.20	0.258065	0.6	0.000000	14
1	2100000	0.4	0.145271	0.307692	0.272727	0.009403	0.25	0.290323	0.8	0.090909	27
2	1602000	0.4	0.142661	0.230769	0.272727	0.009403	0.25	0.258065	0.6	0.090909	17
3	1050000	0.4	0.136960	0.076923	0.272727	0.009063	0.20	0.258065	0.4	0.090909	22
4	1430000	0.2	0.074250	0.076923	0.272727	0.007759	0.15	0.225806	0.8	0.090909	29

I have used Price column as y variable and the rest 10 columns as X variables, as follows:

```
from sklearn.model_selection import train_test_split

X = df[cols].drop(columns = 'PRICE')
y = df[cols].PRICE

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size = 0.30, random_state = 7 )
```

I have taken 70% of the data to train and the rest 30% of the data to test the dataset.

Data Modeling

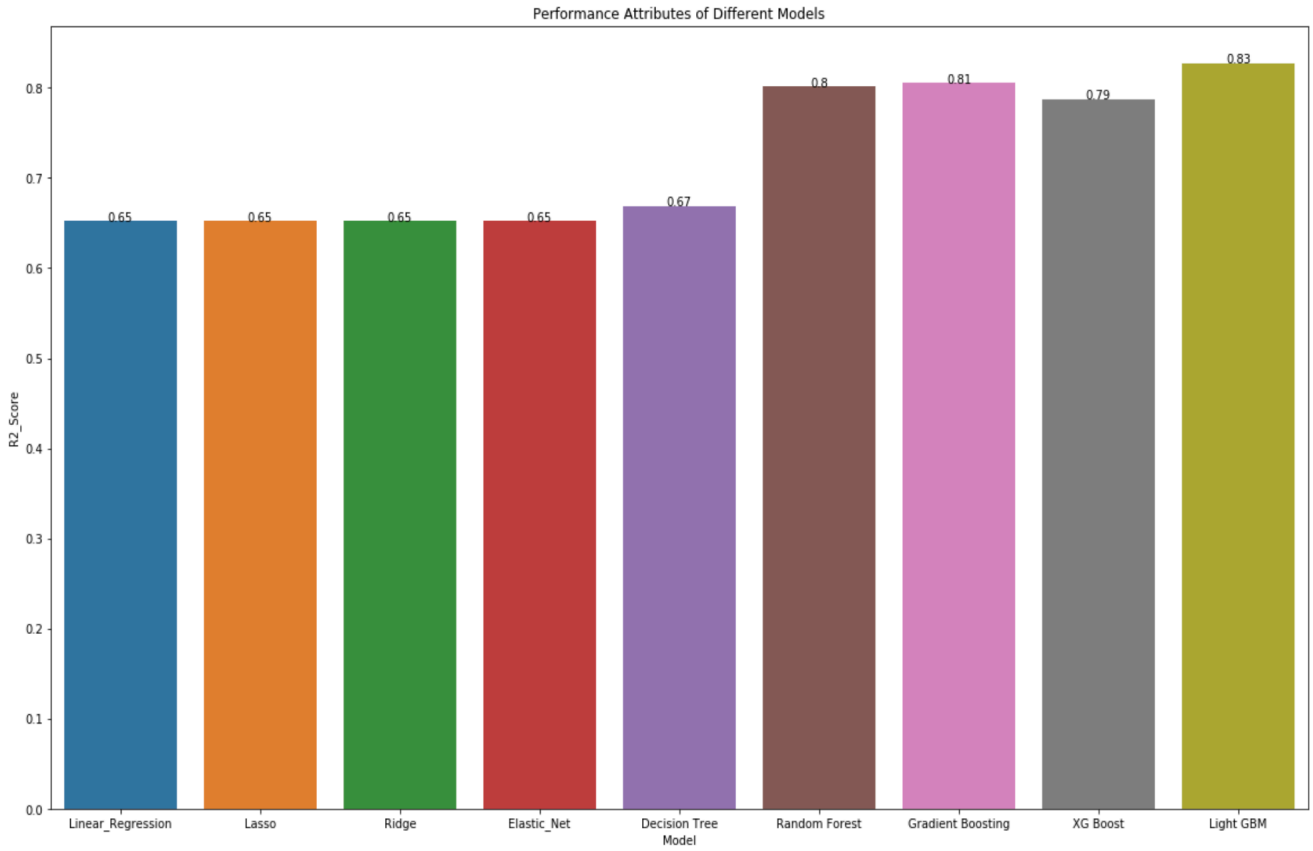
I have used following modeling techniques to analyze the data set:

1. Linear Regression
2. Lasso Regression
3. Ridge Regression
4. Elastic Net Regression
5. Decision Tree Regressor
6. Random Forest Regressor
7. Gradient Boosting Regressor
8. XG Boost Regressor
9. Light GBM Regressor

	Model	RMSE	R2_Score
0	Linear_Regression	337467.121683	0.652529
1	Lasso	337467.121587	0.652529
2	Ridge	337467.121417	0.652529
3	Elastic_Net	337464.480904	0.652534
4	Decision Tree	329324.047850	0.669095
5	Random Forest	254953.401206	0.801675
6	Gradient Boosting	252607.796075	0.805307
7	XG Boost	263887.241276	0.787532
8	Light GBM	237725.757133	0.827572

The performance of each of these models is best captured by the R2_Score. According to the R2_Score, Light GBM Regressor performs best in terms of predicting the test results, whereas Linear, Lasso and Ridge regressor performs the worst in terms of prediction.
It must be noted that after selecting the features which are highly correlated with price variable, we were successfully able to predict the movements in price variables based on top 10 highly correlated features in the dataset.

I have plotted the performance of each of the models using the bar chart as below:



Random Forest, Gradient Boosting, XG Boost and Light GBM regressors, which are ensemble-based methods, exhibits best performance whereas linear techniques such as regression performs the worst in predicting the test results.

Conclusion

The prices of residential real estate in the DC area is dependent of several factors, many of which were mentioned in the DC properties dataset. There were many features which were highly correlated and many which were irrelevant to price prediction. I was successfully able to capture the variation of price movements by plotting them against various different features in the dataset, through explanatory data analysis. I deduced the top features which best predicts pricing of houses by computing and analyzing correlation matrix. Finally, I was able to perform data modeling through various modeling techniques in order to determine which model is best suitable to predict the movements in price variable.