# Introduction

In this project, our aim is to compare and predict the price of house sales over a span one year, from August 2012 to August 2013. All these houses are located in different neighborhoods of Manhattan. For this project, we have used jupyter notebook and python version 3. The dataset used in this project is the Manhattan.csv, which is a dataset originally comprises of 27400 entries (rows) and 21 attributes (columns).

# Dataset Quality

The Manhattan dataset is not of good quality. Instead of the usual format that datasets are usually in, the first 3 rows of the dataset comprise of introductory information about the dataset, and the attribute names are in the 4th row. The actual dataset values start from the 5th row. This makes it difficult to directly load and start using the dataset using pandas .csv() function.

It is not well cleaned either and has several columns which have a large number of missing values. A few such columns are:

RESIDENTIAL UNITS with 16372 missing values

COMMERCIAL UNITS with 23962 missing values

LAND SQUARE FEET with 22906 missing values

GROSS SQUARE FEET with 23069 missing values

Moreover, all the columns are of object type instead of the numerical columns being of a numeric type, which means we need to transform these columns into numeric columns before we can work with this dataset. Additionally, since the numeric columns are in object type, they also have unwanted symbols in them, such as "&" and ",".

Also, in addition to having the usual NULL values, the dataset also consists of numeric rows with 0 values and string rows with empty strings. These are technically NULL values as well, and so we must deal with them before working with this dataset.

# Dataset Cleaning

As already mentioned, the dataset is extremely unclean, so the first step is cleaning the data. This was probably the most time-consuming step.

First, we upload the dataset and make it so that the format of the dataset is the same as the general format of most csv files. We removed the first three redundant rows, and made the fourth row the headings of the columns in the dataframe.

Next, we removed some of the redundant columns mentioned in the problem statement given, as well as rename some incorrect columns (“SALE\nPRICE” as “SALE PRICE”).

After this, we removed the ‘,’ and ‘$’ symbols from all numeric columns, and converted the numeric columns to float. I did not want to risk converting it into int, in case there were some decimal values. I also converted the date column to the standard datetime format for pandas dataframe.

Next, I dealt with the missing values. For the numeric attributes, I replaced the 0s with NaN values. For the string attributes, I did the same with the empty strings. Now that all the NULL values in the dataset were here, I started working on it. The columns with an extremely large number of missing values were removed. Another column I removed at this step apart from the ones already mentioned in the ‘Dataset Quality’ part was the BUILDING CLASS CATEGORY attribute, since it had too many categories and was causing problems with implementing the model. Next, I removed the rows with missing values for those columns which did not have a large number of missing values. And finally, for the following three rows which had a moderate number of missing values: TOTAL UNITS, YEAR BUILT, SALE PRICE, I applied the SimpleImputer() class to them, since I felt that the first 2 attributes were important in determining the price of the house, and the 3rd attribute is our target variable.

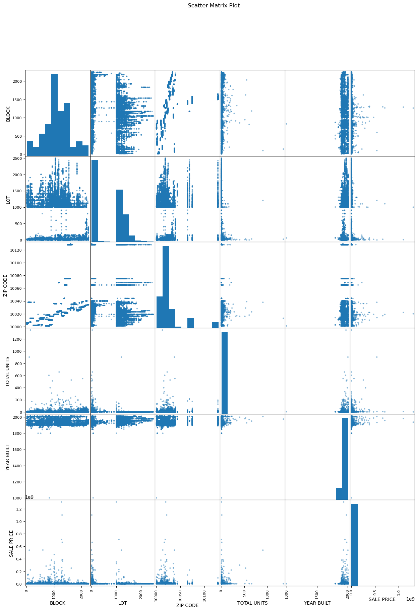
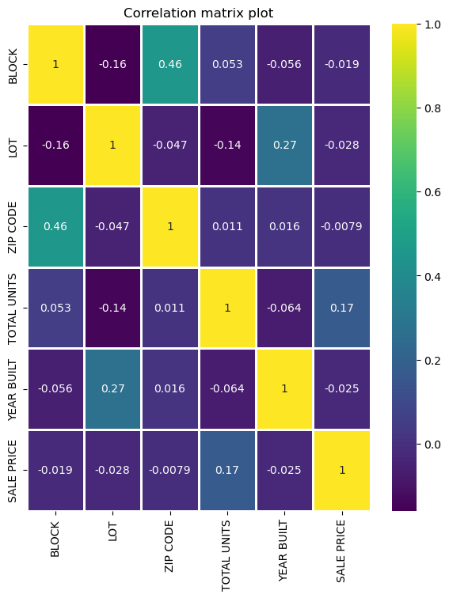
After dealing with missing values, I checked for outliers with the .hist() function, and removed some of the outliers present in the dataset.

# Data Exploration and Visualization

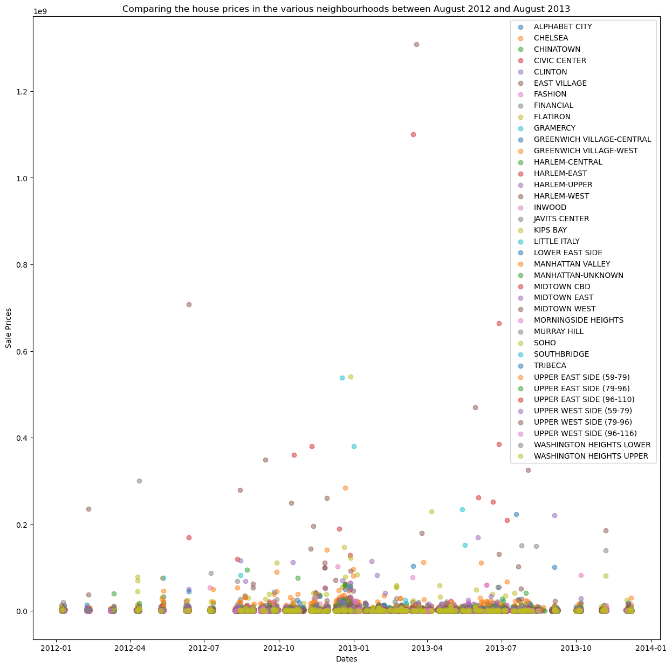
For this step, I first started by exploring the categorical attributes. I checked how many categories were there in each of the following categorical attributes: NEIGHBORHOOD, TAX CLASS AT PRESENT, TAX CLASS AT TIME OF SALE, BUILDING CLASS AT PRESENT, BUILDING CLASS AT TIME OF SALE.

Next, I checked the correlation between the different independent variables and the target variable using the .corr() function. However, it appeared that none of the numerical attributes have a very strong correlation with the target variable.

I also plotted the correlation matrix and the scatter matrix next. From the correlation matrix, one interesting find was that the 2 independent variables – BLOCK and LOT have a high correlationship with one another. Even though the two independent variables were correlated, I felt that they were important factors in determining the house price and decided to keep them. For the scatter matrix, however, we do not see the strong correlation between those two independent attributes.

We also plotted the following scatter-plot for the SALE PRICE vs SALE DATE for the various neighborhoods, and obtained the following plot:



# Applying the ML models

## Data Preprocessing:

Now, I started with preprocessing our data so that I can feed it to the ML model. First, I separated the dataframe into x (matrix of features) and y (target variable). Next, I split the datetime object into 3 columns – year, month and date, since otherwise we cannot use them in our model. I also applied feature engineering to remove certain redundant columns which I felt were not going to contribute a lot to the result (ADDRESS, BUILDING CLASS AT PRESENT, BUILDING CLASS AT TIME OF SALE). Then we split the dataset into training and test set.

The next step is to apply the various transformations. I applied OrdinalEncoder() for the categorical attributes, in order to convert them into labels and StandardScaler() on the numerical attributes, in order to scale them. I also applied PCA() to perform dimension reduction, thus effectively reducing the number of attributes to 3.

After applying these transformations on the training set, I applied them on the test set.

## Cross-Validation

For the next step, I applied cross-validation using the DecisionTreeRegressor() to check for over-fitting and under-fitting. The values of root mean square error (rmse) is much smaller on the training dataset, as compared to the rmse scores on the validation sets which are comparatively larger. This shows that our model is over-fitting to the dataset.

## Applying the traditional ML models

In the following table, we see the models which were applied on the dataset, their rmse as well as a brief description on them:

|  |  |
| --- | --- |
| **Model Name** | **RMSE Score** |
| *Multiple Linear Regression* | *9734506.387* |
| *Polynomial Regression (degree = 2)* | *9709546.681* |
| *Stochastic Gradient Descent (SGD)* | *9755689.516* |
| *Ridge Regression with SGD* | *9771379.283* |
| *Ridge Regression* | *9734506.375* |
| *Lasso Regression with SGD* | *9742678.082* |
| *Lasso Regression* | *9734506.386* |
| *Elastic Net Regression* | *9734392.873* |
| *Linear SVR* | *10019731.676* |
| *SVR with sigmoid kernel* | *9803531.66* |
| *SVR with rbf kernel* | *9803541.929* |

## Applying clustering and then regression on local clusters

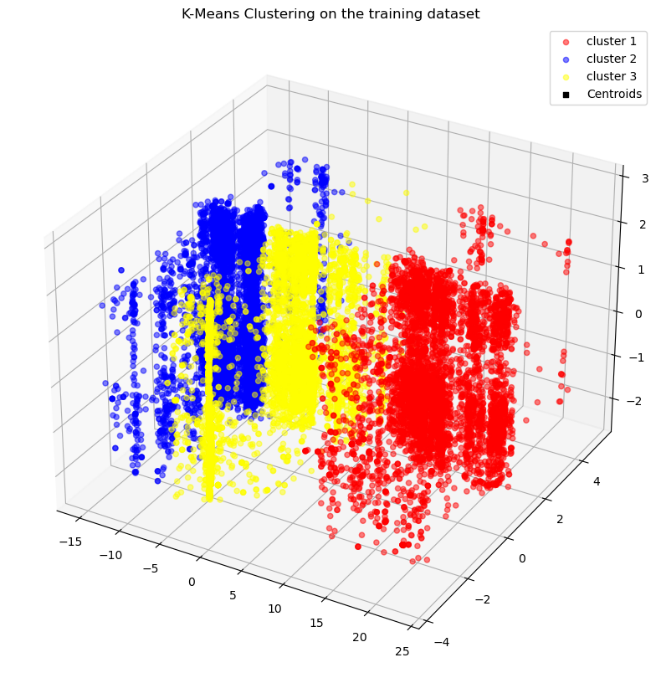
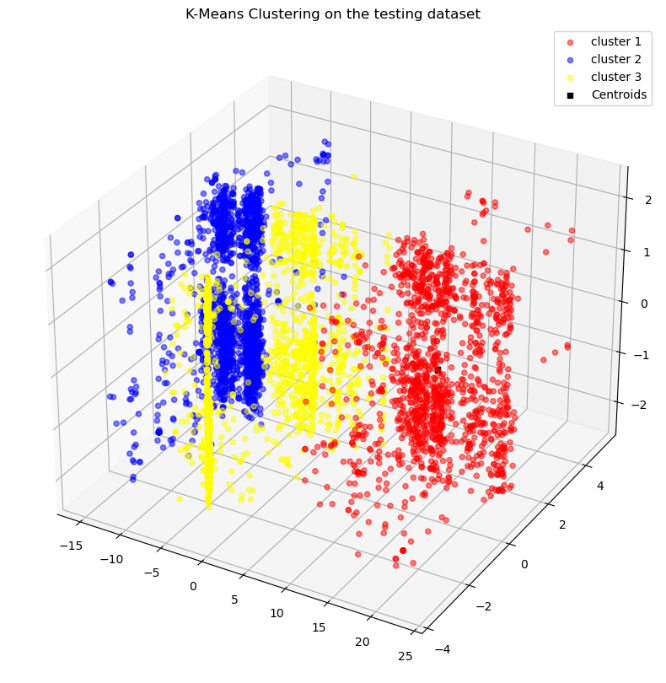
For this step, I first used K-Means clustering to cluster the points of the training dataset. I used the elbow method to determine the number of clusters, which game me 3 clusters. Next, on each of these local clusters I applied Linear SVR model (although Polynomial Regression has the lowest rmse score and I wanted to use that, however due to time constraints I could not correct it).

Next, I used the same K-means object to predict the clusters to which each of the testing dataset points will belong to. According the cluster that they belonged to, I applied the local linear regressor on them. For each local regressor, I calculated the rmse, and found the mean of the rsme obtained from the 3 regressors of the 3 clusters for the final rmse score of the model.

The rmse values obtained via this method is given below:

|  |  |
| --- | --- |
| **Model Name** | **RMSE Score** |
| *Regressor 1* | *8408499.488* |
| *Regressor 2* | *4336318.409* |
| *Regressor 3* | *13814589.214* |
| *All 3 regressors* | *8853135.7* |

The following plot shows the clustering on the training and testing dataset. There are three main clusters which appear as 3 vertical blocks in the 3D space.

# Conclusion

Thus, we can see from our models that the lowest RMSE is obtained when we first find the clusters and then apply the local regressor for each cluster on the dataset. Moreover, in my project, I used the linear regressor with the highest RMSE for this, which shows that the accuracy of the model will be further enhanced if we use a better model, such as Polynomial Regression.

Apart from the K-Means clustering and regressor method, the following 5 models have the lowest RMSE scores and thus the best performance:

1. Polynomial Regression
2. Elastic Net Regression
3. Ridge Regression
4. Lasso Regression
5. Multiple Linear Regression