Final Project Report

Cumming House Price Prediction

Anitha Subramanian

27th April 2020

Business Understanding

Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. As a resident of Cumming (GA) region, I would like to study the residential house prices range in this region. This will be helpful for real-estate firms and individual home buyers and sellers to make a decision on buying and selling houses.

Business Problem

With the improvement of people's living standards, the demand for house increases. Home values Georgia have increased 7.5% over the last 12 months and are forecast to rise another 5.5% in the next year. Generally the problems faced during buying a house are, Buyers are generally not aware of factors that influence the house prices. They rely on real estate agents for buying and selling which creates a middle man and increases the cost of houses. People believe that it depends on Square foot, neighborhood and no.of bedrooms. But there are other factors that influence the house prices are No.of storeys, lot size, Basement, etc.,

Dataset

The dataset is the prices and features of residential houses sold from February 20 2020 to April 4 2020 in Cumming, GA , obtained from a real estate agent. This is simple dataset with only 10 features and 387 houses with sold prices. Although the dataset is very small with only 387 examples, it will definitely help us to explore various techniques to predict the house prices in Cumming region of Georgia.

The dataset consists of features in various formats. It has numerical data such as prices, number of bed rooms, number of bathrooms, square foot, lot size and categorical features such as Zip code, House type, Basement type. I have splitted the dataset into training and testing set with roughly 80/20, with 216 row instances for training set and 54 row instances for testing set. Below picture shows sample 10 rows from the dataset.

```
house_df.head(10)
```

	Address	Zip code	Price	No.of Bedrooms	No.of Bathrooms	Sqft	HouseType	Basementtype	Lot Size (Acres)	Year Built
0	3235 Chimney cove Dr	30041	500000.0	3	4	2700.0	Basement	Finished	0.75	1991
1	3980 Sawmill ct	30040	389000.0	5	4	3150.0	No Basement	Slab	0.41	2009
2	4130 Creststone ct	30040	412000.0	5	4	3200.0	Basement	Finished	0.40	2015
3	5805 Kent Pl	30040	439900.0	4	3	3004.0	Basement	Unfinished	0.27	2015
4	4305 Celbridge Pass	30040	444900.0	3	3	2060.0	No Basement	Slab	0.12	2050
5	7515 Austin Mill Dr	30041	349900.0	5	3	3700.0	Basement	Finished	0.50	2001
6	3570 Sandown Ct	30041	316000.0	5	4	3087.0	Basement	Finished	0.31	2006
7	5830 Rogers Road	30040	324995.0	4	3	2548.0	No Basement	Slab	0.78	1999
8	7450 Olivia View ct	30028	349450.0	5	4	3220.0	No Basement	Slab	0.27	2018
9	4795 Roosevelt Cir	30040	320000.0	4	3	2265.0	No Basement	Slab	0.22	2011

Proposed analytics solution

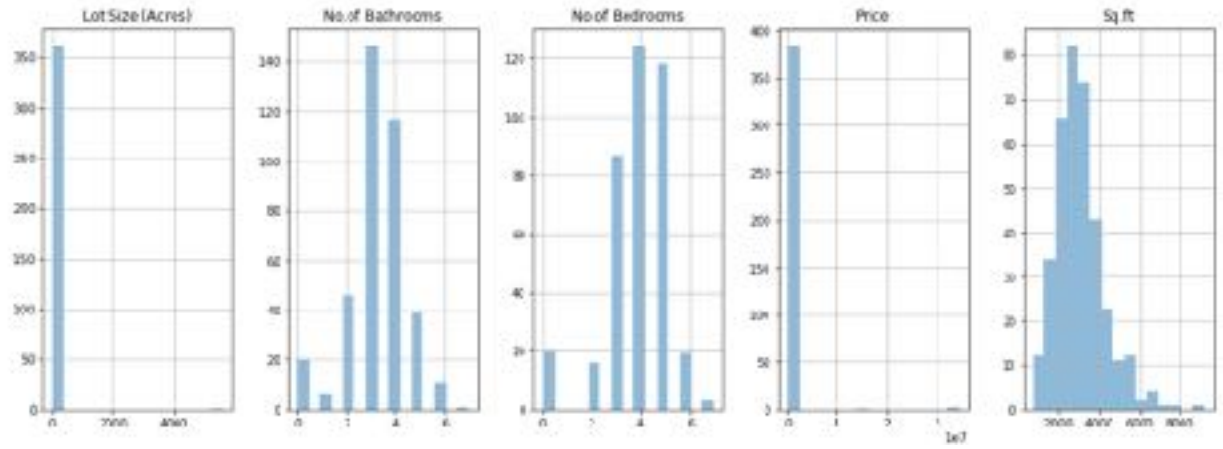
In this project, we will develop and evaluate the performance and the power of a model trained and tested on data collected from houses in Cumming region of Georgia state. Once we find a good performing model, we will use the model to predict monetary value of a house located at the Cumming area. So the target variable here will be Price. Since it is a continuous target, we will be performing Regression (Supervised Machine learning) on the dataset. Build a good model with minimized error and good efficiency and robustness. A model like this would be very helpful for real estate agents who could make use of the information provided in a daily basis.

Data Exploration and Preprocessing

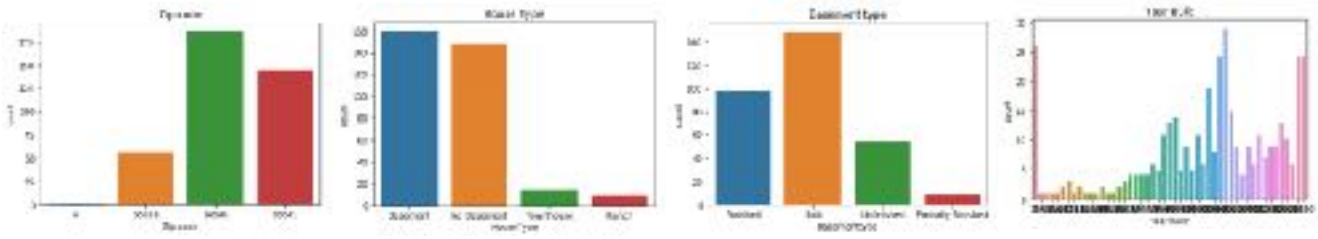
Data Exploration refers to the initial and important step in any data analysis or machine learning. We must first understand and develop a comprehensive view of the data before extracting relevant data for further analysis.

This Cumming house prices dataset contains 387 rows and 10 features. The features are Address, Zip code, Price, No.of Bedrooms, No.of Bathrooms, Sq.ft, House Type, Basement type, Lot size (Acres), Year Built. The below section shows the visualization of numerical and categorical attributes in the dataset.

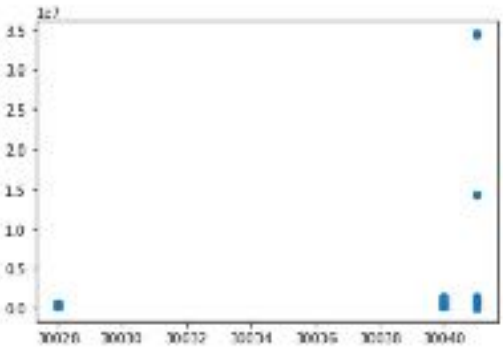
Visualization of numerical attributes



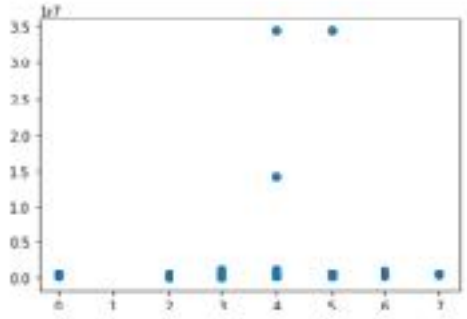
Visualization of categorical attributes



```
plt.scatter(house_df['Zip code'],house_df['Price'])
<matplotlib.collections.PathCollection at 0x1a25a37d30>
```



```
#Identify relationship between count of bedrooms and Price
plt.scatter(house_df['No.of Bedrooms'],house_df['Price'])
<matplotlib.collections.PathCollection at 0x1a1cf17ac8>
```



From the above left graph, We could see that Zip code 30041 has more houses sold and also expensive region. From the right graph, It is obvious that 4 and 5 bedrooms are pricey in Cumming region.

Data Quality Report

The data quality report is for visualizing the key values like mean, median, mode, maximum value, etc., in the dataset based on the nature of attributes. The challenge with data quality is that there are no clear and simple formulas for determine if data is correct. We are just generating the data quality report to get to know about the features. For example, if the Price shows minimum value as 63200 (as per our dataset), then it states that the selected Cumming dataset has house prices higher than 63200. By observing the maximum value, we can say that the dataset has price range from 63200 to 3.44M.

Below picture shows the Data Quality Report for Numerical and Categorical features.

```
Data Quality Report for Continuous features
Total records: 387
```

	Data Type	Count	Missing	Cardinality	Minimum	1st Qrt.	Mean	Median	3rd Qrt.	Maximum	Standard deviation
Price	float64	386	1	274	63100.00	295326.25	596711.126943	359450.00	449431.50	34400000.0	2.547623e+06
No.of Bedrooms	int64	387	0	7	0.00	3.00	3.912145	4.00	5.00	7.0	1.398042e+00
No.of Bathrooms	int64	387	0	8	0.00	3.00	3.299742	3.00	4.00	7.0	1.294247e+00
Sq.ft	float64	366	21	341	730.00	2258.25	3050.366120	2937.00	3638.50	9201.0	1.175749e+03
Lot Size(Acres)	float64	362	25	99	0.02	0.21	16.142265	0.31	6.58	5663.0	2.976155e+02

```
Data Quality Report for Categorical features
Total records: 387
```

	Data Type	Count	Missing	Cardinality	Mode	Mode Frequency	Mode Percentage
Zip code	int64	387	0	4	30040	186	48.06
HouseType	object	331	56	4	Basement	100	41.94
Basementtype	object	309	78	4	Slab	149	38.59
Year Built	int64	387	0	49	2000	29	7.49

From analyzing both the Data Quality Report and Bar chart graph, we can say that Cumming region falls into 4 zip code. The cardinality in data quality report shows that there are only 4 unique values for the Zip code feature. Same way, Bar chart shows that 0 (Wrong data), 30028,30040,30041 are the zip code for Cumming region.

This is how the Data Quality Report needs to be interpreted. And this plays major role in understanding our data.

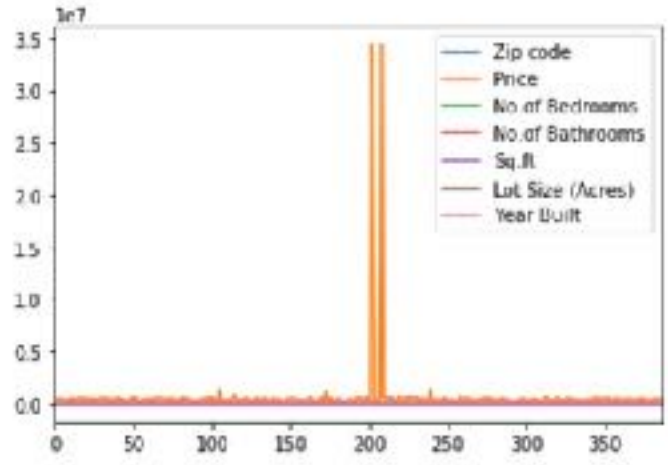
Missing Values and Outliers

One of the major tasks in data preprocessing is handling missing values and outliers. Missing values can arise from information loss as well as dropouts and nonresponse from the study participants. Outliers refers to extreme high or extreme low values that abnormally lie outside the overall pattern of a distribution of variables.

Visualization of features with missing values and outliers in our dataset

```
#Visualization of features using plot before Outliers and Missing values
house_df.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e1d171470>
```



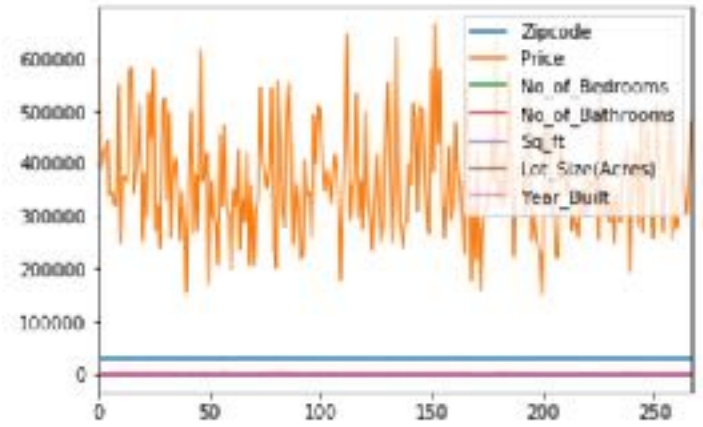
Missing values and outliers are obvious in any dataset. Missing values can be treated in many ways like drop the feature, drop records (complete case analysis),derive missing indicator and imputing missing values. Here in out dataset, I have imputed the missing values using Knn imputation. Each sample's missing values are imputed using the mean value from n_neighbors.

Outliers can also be handled using Clamp transformation, IQR method, Standard Deviation method and 2% data from top/bottom will be removed. Here in our Cumming house prices dataset I removed the outliers using IQR (InterQuartile Range) method. It works by identifying interquartile range (difference between first quartile and third quartile).

Visualization of features after handling missing values and outliers in our dataset:

```
#Visualization of features after removing outliers and missing values
imputeddf.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1dbaaf28>
```



Normalization

Data Normalization is a technique to change the numeric values in the dataset to a common scale, without distorting differences in the range of values. Normalization improves the numerical stability of the model. Normalization is required only when features have different ranges.

Here in our dataset, No.of Bedrooms has ranges from 1-7 and Lot size has different range from 0.1 to 3. Also there are string values for certain features like House type, Basement type. These string values will be changed to numerical using label encoder. To bring all the feature values to same range, Normalization will be performed. There are many ways to do Data Normalization, here I chosen MinMaxScaler for transforming the values.

The below screenshot shows the data after normalization.

	Zipcode	No_of_Bedrooms	No_of_Bathrooms	Sq_ft	HouseType	Basementtype	Lot_Size(Acres)	Year_Built
0	1.0	0.2	0.666667	0.408869	0.000000	0.000000	0.005238	0.189500
1	0.0	0.6	0.666667	0.468617	0.333333	0.666667	0.021498	0.700333
2	0.0	0.6	0.666667	0.525184	0.000000	0.000000	0.001805	0.695833
3	0.0	0.4	0.333333	0.327250	0.000000	1.000000	0.230085	0.695833
4	0.0	0.2	0.333333	0.262180	0.333333	0.666667	0.005238	1.000000
5	1.0	0.6	0.333333	0.514300	0.000000	0.000000	0.457142	0.804167
6	1.0	0.0	0.000000	0.485640	0.000000	0.000000	0.150952	0.708333
7	0.0	0.4	0.333333	0.437342	0.333333	0.666667	0.723810	0.354167
8	0.0	0.4	0.333333	0.523328	0.333333	0.666667	0.190476	0.512500
9	1.0	0.6	1.000000	0.555036	0.000000	0.000000	0.314286	0.562500
10	0.0	0.0	0.333333	0.258400	1.000000	0.666667	0.000000	0.687500

Feature Selection

Having irrelevant features in our dataset can decrease the accuracy of the models and make the model learn based on irrelevant features. To avoid this. We can perform feature selection process in our dataset. Feature selection is the process of picking the features which contribute most to the prediction variable. We can do the feature selection using filter or wrapper methods.

I implemented two ways of feature selection for the Cumming house prices dataset. One is using

Filter method - Correlation Co-efficient Scores, where the features are selected using correlation scores method.

Below picture shows the correlation scores for the features,

The second method I used for feature selection is Sequential Forward Selection (Wrapper method). Features are selected using scoring method. After doing this, “Year_Built” is the feature with less score. (Since I have very few features, am not implementing feature selection this in my dataset).

Model Selection and Evaluation

For regression models, we try to solve the following problem: given a processed list of features for a house, we would like to predict its potential sale price. Linear regression is a natural choice of baseline model for regression problems. So I first ran linear regression including all features.Let us see the performance metrics and models I used for building Cumming house prices prediction.

Evaluation Metrics

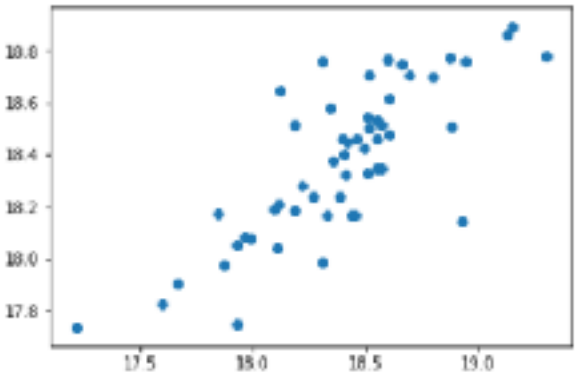
There are many metrics used for evaluating models on regression machine learning problems. Based on the accuracy and being widely used metric, I have taken Root Mean Squared Error (RMSE) as my Evaluation metric. RMSE is the square root of the average squared difference between prediction and actual observation. There is a scikit learn library available for RMSE and can be added as **"from sklearn.metrics import mean_squared_error".** Once we are done with training the model and testing it, we can find the error rate between actual and predicted sale price of a house. Based on this error value, we can decide with the perfect model for our requirement. Lower the RMSE error value better the model.

Models

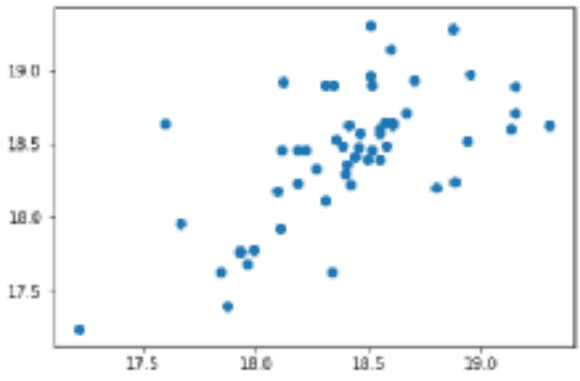
We will start with Simple Linear Regression model to predict the house prices. This model is built using LinearRegression(). Then fit the model with training set of X and y. Where X is descriptive features and y is target feature of the dataset. The model output will be predicted sale price for a house with specified zip code, bathroom, bedroom count, lot size, etc.,

Below are the models created for predicting sale price of houses in Cumming. The models are shown with scatterplot to view their performance in this dataset.

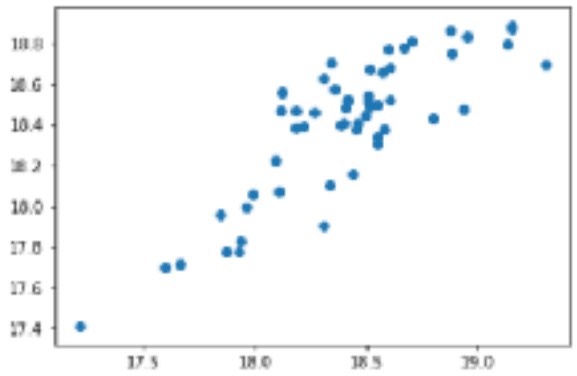
Linear Regression Model



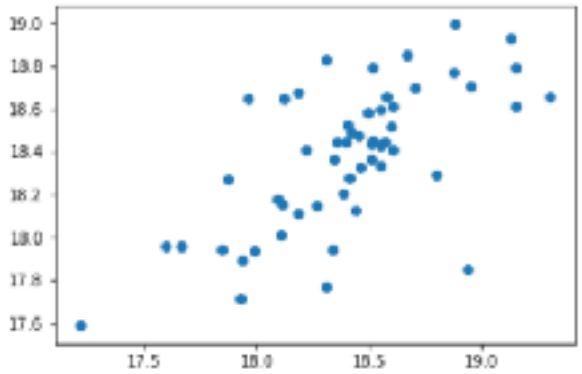
Decision Tree Regression Model



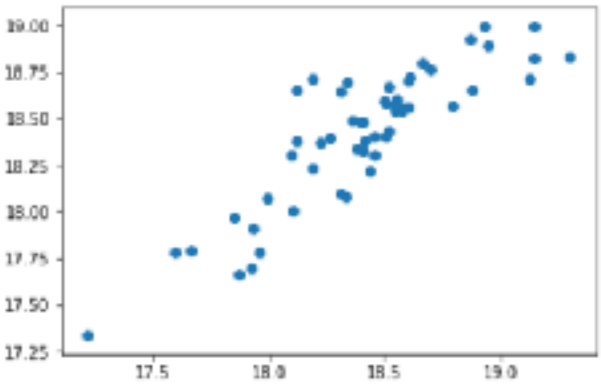
Random Forest Regression Model



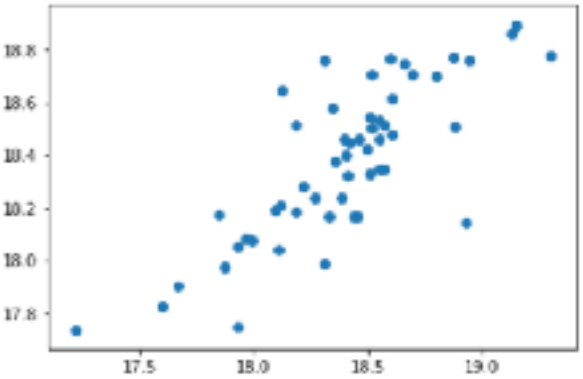
KNN Regression Model



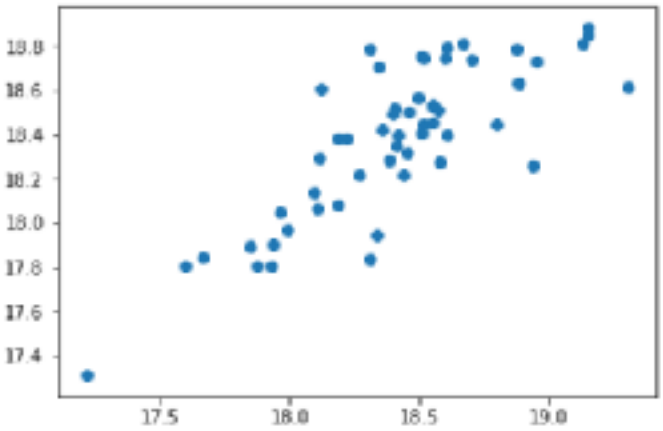
Gradient Boosting



Linear with GridSearchCV



Decision Tree with GridSeacrhCV



From the above scatter plots, it is visible that Gradient Boosting performs well compared to other models.

Sampling and Evaluation Settings

To create a model, we need to divide the dataset into train and test set. The train set is where the model is trained, while the model is evaluated on the test set. Both the train and test set are created as a result of sampling. Here the train and test set is splitted like this,

X_train, X_test, y_train, y_test = train_test_split(X, y ,test_size=0.2)

This dataset is divided into 80% train set and 20% test set. 216 rows in this dataset is utilized for training the model, wherein 54 row instances is used to test on the trained model. X is the dataset with descriptive features and y is the dataset with target feature.

For evaluating the model, mean_sqaured_error is used in this project.

mse = mean_squared_error(y_test, predictions_linear)

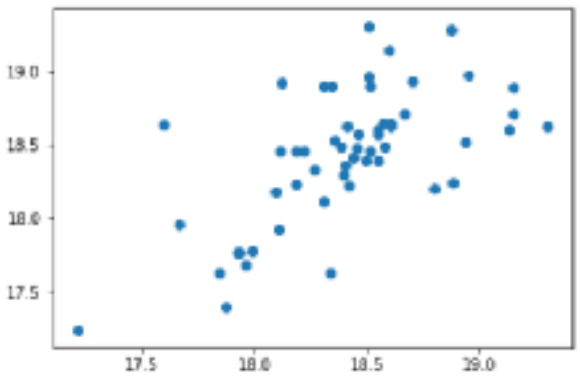
root_mean_square_error_linear = np.sqrt(mse)

root_mean_square_error_linear will display the error value for that model.

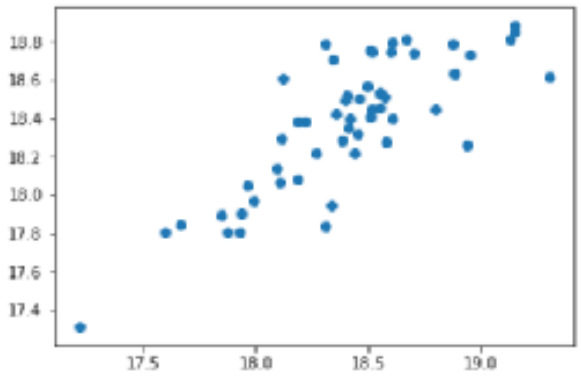
Hyper-parameter Optimization

From the above model scatter plots, we could see that Decision tree values are scattered. So I tried Hyper-parameter optimization on Decision tree to see the improvement in model with GridSearchCV. Max-depth of range from 1 to 10 has been selected, along with r2 scoring function are used as the parameters for GridSearchCV. After creating Decision Tree Regressor with GridSearchCV, we could see from the below plot that the model is improved compared to the previous one.

DT Without Optimization



DT with Optimization



Evaluation

Regression Models	Root Mean Squared Error (RMSE)
Linear Regression	0.238288
Decision Tree Regressor	0.371186
Random Forest Regressor	0.214113
KNeighbors Regressor	0.310029
Gradient Boosting Regressor	0.199622
Decision Tree with GridSearchCV	0.259637

From the above evaluation table, We could see that Gradient Boosting Regressor performed well for this dataset with less error value. By doing hyper parameter optimization, Decision Tree Regressor is improved from 0.371186 error value to 0.259637 error value.

Conclusion

For this Cumming house prices prediction dataset, the best performing model is Gradient Boosting Regressor with RMSE of 0.199622. The second good performing model is Random Forest regressor with RMSE of 0.214113. According to my analysis, living area square foot, basement type and count of bed rooms plays a major role in the sale price of houses in Cumming region.