**A house price evaluation system**

**Overview**

This project predicts the prices of houses in Burlington, Vermont. An unbiased and objective evaluation of home prices based on big data is beneficial to both buyers and sellers of houses. This project can be furthered improved by taking relevant economic factors (e.g. interest rates, average income, rents) into investigation to produce more accurate evaluation results for different housing markets. Potentially, this project can offer helpful suggestions for local real-estate companies or individual home buyers or owners.

**Method**

While values of house properties are based on supply and demand which in turn could be an outcome of various intangible and hard-to-measure factors, we will look into the most direct factors of house values: various properties of a house. We will unveil how the value of a house depends on its various of properties such as its location, its year-of-built, its size etc. based on various machine learning and statistical models.

This approach is reasonable on its own right since the effects of various properties of a house on its value is subjected to the current economic environment. By modelling the relation between the properties of houses and house values, various economic factors that take effect on supply and demand actually play roles in our evaluation system even though in an implicit way.

**Data**

The main data set to be used is the “City of Burlington Property Details” data, which is open to public in <https://data-burlingtonvt.opendata.arcgis.com/datasets/276ccff527454496ac940b60a2641dda_0>

This data set contains 10832 records on taxable house property details including “Number of Units”, “Number of Rooms”, “Heat Type”, “Building Type”, “Current Building Value” etc.

One other relevant dataset is the “City Property Sales History” data which provides information on a list of properties sold in the City of Burlington. This dataset can be found in

<https://data-burlingtonvt.opendata.arcgis.com/datasets/f906a4f4403f41afbef5617e1efaecd0_0>

**Goal**

While the current values of house properties are given in the “City of Burlington Property Details” data, it is not clear how various aspects of a house is related to its current value. We attempt to establish a predictive model to demonstrate the effect of various features of a building on its value. The “Current Building Value” information in the “City of Burlington Property Details” data set instead will be used to test the accuracy of our predictive model.

The usage of the model is to give an estimated value or a range of estimated values based on the information on a house. Interested home buyers/sellers can use our established model of a house price evaluation system as a reference before making a decision.

**Data Preparation**

While the data set contains sufficiently many columns which describe details of housing properties, empty records, mistyped data etc. need to be dealt with before the conduct of data analysis and machine learning. The data set also contains irrelevant information of housing properties such as the ID number of the property, owners’ names etc. which be discarded for the purpose of better data analysis. What have been done to the original data set are 1. preliminary selection of relevant columns 2. treatments to empty data; 3. encoding of non-numeric data and 4. treatments to suspicious data and outliers. Details for each of these data preprocessing steps will be explained below.

**1.Relevant Columns**

The original data set contains columns such as 'AccountNumber', 'ParcelID', 'SpanNumber', which serve as identifiers of housing properties, and columns consisting of owners' names. Most of these identifier/name columns will be deleted. The data set that is used for data analysis contains the columns from the original data set which the author believes to be relevant for analysis of sale prices and current values. The columns that are preserved from the original data set are FID(at least one identifier should be kept), StreetNumber, StreetName, LandUse, CurrentAcres, TotalGrossArea, FinishedArea, CurrentValue, CurrentLandValue, CurrentYardItemsValue, CurrentBuildingValue, BuildingType, HeatFuel, HeatType, Grade, YearBlt, SalePrice, NumofRooms, NumofBedrooms, NumofUnits, ZoningCode, Foundation, Depreciation, PropertyCenterPoint. Most of the column names suggest by their meaning what information is stored. StreetNumber and StreetName together indicate locations of housing properties. PropertyCenterPoint stores the tuples consisting of latitudes and longitudes of housing properties.

**2.Empty data**

Since no information will be obtained from empty data, and empty data is of no use to machine learning algorithms, we will have to treat the empty data either by deleting or filling with reasonable estimated values. The percentage of missing data for all columns are listed in the table below.

|  |  |
| --- | --- |
| Columns | Missing Percentage |
| StreetNumber | 0.0462% |
| StreetName | 0% |
| Unit | 80.8346% |
| LandUse | 0% |
| CurrentAcres | 0% |
| TotalGrossArea | 0% |
| FinishedArea | 0% |
| CurrentValue | 0% |
| CurrentLandValue | 0% |
| CurrentYardItemsValue | 0% |
| CurrentBuildingValue | 0% |
| BuildingType | 3.7020% |
| HeatFuel | 4.5513% |
| HeatType | 3.9051% |
| Grade | 3.7020% |
| YearBlt | 0% |
| SalePrice | 0% |
| NumofRooms | 0% |
| NumofBedrooms | 0% |
| NumofUnits | 0% |
| Columns | Missing Percentage |
| ZoningCode | 0.5078% |
| Foundation | 4.1544% |
| Depreciation | 0% |
| PropertyCenterPoint | 2.0033% |

Based on the missing percentage for each column, we will delete the entire column named ‘Unit’ since this column lost over 80% of data, it will have little use in analysis. After the deletion of the ‘Unit’ column, we will delete rows which contain missing data since other columns do not contain large amounts of missing values.

**3.Non-numeric data**

We will convert categorical data to numeric data by transforming a category to the number of times the category appears in its corresponding column. This frequency method values the importance of categories by their appearing frequencies. More analytically reasonable encoding methods such as OneHotEncoding could be performed when machine learning comes into play. The data in columns ‘StreetName’, ‘LandUse’, ‘BuildingType’, ’HeatFuel’, ‘HeatType’, ‘ZoningCode’ and ‘Foundation’ are converted to numerical values by this frequency method. The column ‘Grade’ is converted to 0-19 based on what the author believes to be a reasonable rating system (Poor<Fair<Average<Good<VeryGood<Custom<Excellent). The column ‘PropertyCenterPoint’ is separated into two columns ‘PropertyCenterPoint\_x’ and ‘PropertyCenterPoint\_y’ which store latitudes and longitudes.

**4.Suspicious data**

The data pertained to each column is checked. Rows that contain exceptional values are deleted. The existence of exceptionally high or low values in numeric-typed columns and infrequent categories in categorical-typed columns could be due to data input errors or if not, due to exceptional factors which cannot be measured easily. We will avoid to use these unusual data in the development of machine learning models.

Since the goal of the project is to estimate values/prices of RESIDENTIAL housing properties, rows with non-residential land use purpose and building types are deleted.

The relation CurrentValue = CurrentLandValue + CurrentYardItemsValue +CurrentBuildingValue has been found. One record(row) violates this equation and hence be deleted.

FinishedArea is proportional to TotalGrossArea. No record has an unusual high/low FinishedArea/TotalGrossArea quotient. No record is dropped based on this criterion.

SalePrice is proportional to CurrentValue. 13 records have unusual high/low sale prices, to be more specific, unusual high/low SalePrice/CurrentValue quotients, and are hence deleted from the data set. These records are dropped since an unusual high/low sale price is very likely due to buyers/sellers’ unusual financial situation and the unusual housing market which we have little or no measure of based on the original data set. We will not take this unusual data points to build our predictive models.

**After the data cleaning, the data to be used contains 4308 rows.**

**Exploratory Data Analysis**

The relationships among features(columns) are investigated. A preliminary data analysis based on visualizations and statistical inferences is conducted to answer the following questions. 1.How do CurrentValue and SalePrice depend on other features? 2. Are there any strong relations between pairs of features? The first question is basically the main question throughout this project. A preliminary answer is given. The question itself will be revisited and answered in more depth as the project develops. The second question is also important for the simplicity of predictive models.

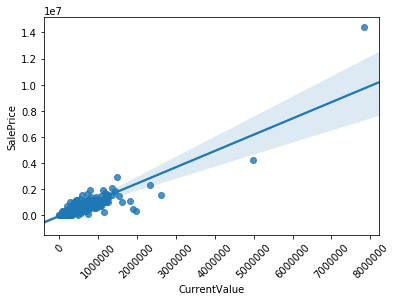
**Some important or interesting findings are described below.**

**1**.SalePrice is proportional to CurrentValue. They have a strong linear relationship.

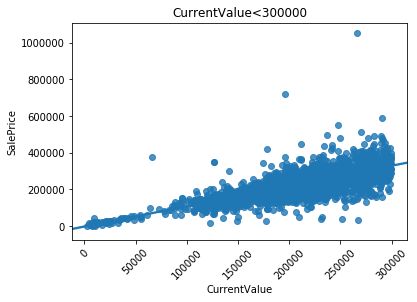
**Statistics:**

A simple linear regression (OLS) model with SalePrice being the dependent variable and CurrentValue being the independent variable gives R-squared: 0.809 and a zero p-value for the t-test.

**Visualizations:**



We will truncate the data set to preserve only the data with CurrentValue less than 300000, and fit the linear model again. The linear relationship is easier to visualize for the truncated data set as the following plot shows.

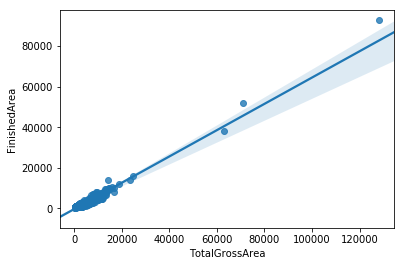


**2**.TotalGrossArea is proportional to FinishedArea. They have a strong linear relationship.

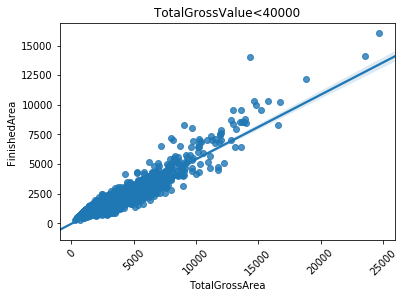
**Statistics:**

A simple linear regression (OLS) model with FinishedArea being the dependent variable and TotalGrossArea being the independent variable gives R-squared: 0.942 and a zero p-value for the t-test.

**Visualizations:**



We will truncate the data set to preserve only the data with TotalGrossValue less than 40000, and fit the linear model again. The linear relationship is easier to visualize for the truncated data set as the following plot shows.

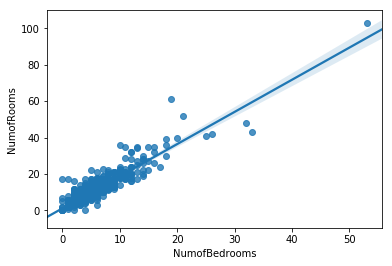


**3**.NumofRooms is proportional to NumofBedrooms. They have a strong linear relationship. This could mean most of rooms are bedrooms in a large portion of housing properties.

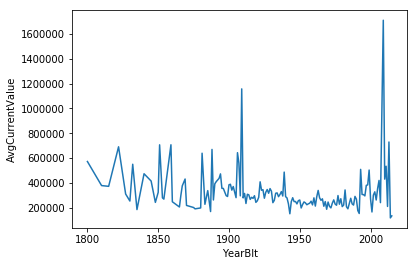
**Statistics:**

A simple linear regression (OLS) model with NumofRooms being the dependent variable and NumofBedrooms being the independent variable gives R-squared: 0.827 and a zero p-value for the t-test.

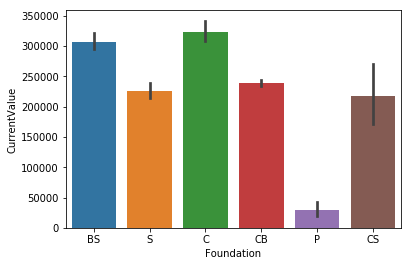
**Visualizations:**



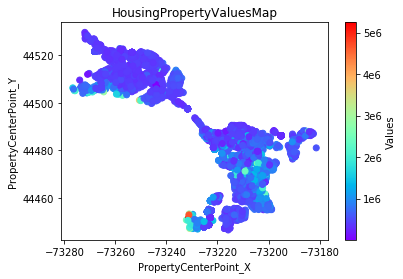
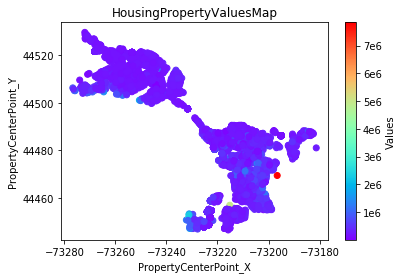
**4**.The average current values do not have large fluctuations for the housing properties built in the 20th century. However, properties built in the early 20th century or early 21st century might have high values.

****

**5**.The average current value of housing properties with the ‘P’ foundation tend to be lower than the average current values of housing properties with other foundations.

****

**6**.The current values of housing properties in the South part of the city tend to be lower than the average current values of housing properties in the North part of the city. The highest value has been found in the South part of the city though. The first plot is based on the full cleaned data set. The second plot truncates the cleaned data set to only preserve the data with CurrentValue<4000000.

****

**Machine Learning and Regression Analysis**

We leverage 4 regression models: Linear Regression, classical Gradient Boosting Regression, Random Forest Regression and XGBoosting Regression to estimate the sale prices of housing properties. Parameters in each model will be tuned to search for better models in terms of estimation accuracy. Two standards are used to evaluate the performances of models:

* RMSE: Root Mean Square Error (https://en.wikipedia.org/wiki/Root-mean-square\_deviation)
* MAPE: Mean Absolute Percentage Error (https://en.wikipedia.org/wiki/Mean\_absolute\_percentage\_error )

The following steps will be done to set up machine learning models.

* A preliminary feature selection based on the results on previous Exploratory Data Analysis to avoid dependence among feature variables and to filter out irrelevant features.
* One-Hot-Encoding to categorical data. In previous data processing, categorical data has been encoded to numerics by a frequency method, which does not preserve all the information on the original data set. One-Hot-Encoding, on the other hand, captures all the information on the original data set. Hence, we will use One-Hot-Encoding to further process the data for the purpose of achieving better performances of machine learning models.

**A preliminary feature selection for machine learning is as follows.**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=dbff7b9cd20c3443aa932173edc9c0ee1e36f67c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f736f636861636861692f43617073746f6e6550726f6a656374312f646266663762396364323063333434336161393332313733656463396330656531653336663637632f4d616368696e654c6561726e696e67526573756c74732e6970796e62&nwo=sochachai%2FCapstoneProject1&path=MachineLearningResults.ipynb&repository_id=154411438&repository_type=Repository#A-preliminary-feature-selection-for-machine-learning-is-as-follows.)

* **1.**FID should not be used in any type of analysis.
* **2.**Since StreetName, StreetNumber, ZoningCode are all location variables which are less significant in determining SalePrice than the coordinate variables, as seen in Statistical Analysis, we will not use them as features in machine learning. However, we will preserve PropertyCenterPoint\_x and PropertyCenterPoint\_y.
* **3.**It has been found in the data cleaning step that CurrentValue = CurrentLandValue+CurrentYardItemsValue+CurrentBuildingValue. CurrentValue and SalePrice is linearly related. We will preserve SalePrice, CurrentLandValue, CurrentYardItemsValue, CurrentBuildingValue but ignore CurrentValue in machine learning algorithms.
* **4.**Recall NumofRooms and NumofBedrooms are strongly linearly related, and that SalePrice depends more on NumofRooms than on NumofBedrooms based on the correlation matrix, we will use NumofRooms instead of NumofBedrooms as a feature in machine learning algorithms.

**Training and Test Datasets**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=dbff7b9cd20c3443aa932173edc9c0ee1e36f67c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f736f636861636861692f43617073746f6e6550726f6a656374312f646266663762396364323063333434336161393332313733656463396330656531653336663637632f4d616368696e654c6561726e696e67526573756c74732e6970796e62&nwo=sochachai%2FCapstoneProject1&path=MachineLearningResults.ipynb&repository_id=154411438&repository_type=Repository#Training-and-Test-Datasets)

**# Set the target and variables**

X = df\_feature.drop(columns=['SalePrice'])

y = df\_feature.SalePrice

**# Generate Training and Test sets, and perform scaling**

**from** **sklearn.model\_selection** **import** train\_test\_split X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0) **from** **sklearn.preprocessing** **import** StandardScaler scaler = StandardScaler() X\_train = scaler.fit\_transform(X\_train) X\_test = scaler.transform(X\_test)

**Gradient Boosting Regression**

**Parameter Tuning**

We search models with the the lowest RMSE and the lowest MAPE. The loss function is chosen to be 'huber'. Three parameters are chosen to be tuned.[¶](https://render.githubusercontent.com/view/ipynb?commit=dbff7b9cd20c3443aa932173edc9c0ee1e36f67c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f736f636861636861692f43617073746f6e6550726f6a656374312f646266663762396364323063333434336161393332313733656463396330656531653336663637632f4d616368696e654c6561726e696e67526573756c74732e6970796e62&nwo=sochachai%2FCapstoneProject1&path=MachineLearningResults.ipynb&repository_id=154411438&repository_type=Repository#Next,-we-search-models-with-the-the-lowest-RMSE-and-the-lowest-MAPE.-The-loss-function-is-chosen-to-be-'huber'.-Three-parameters-are-chosen-to-be-tuned.)

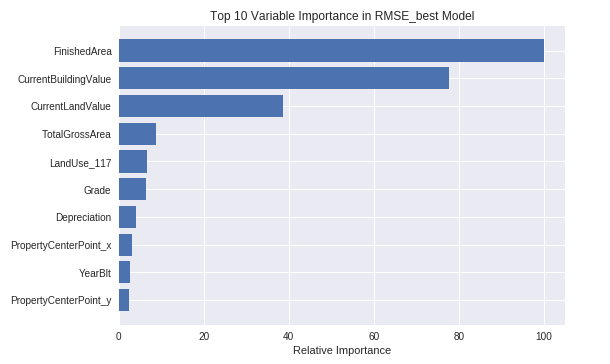
* N\_estimators: The number of boosting stages to perform. from 50 to 500 with a step of 50.
* Max\_depth: the maximum depth of the individual regression estimators from 2 to 8.
* Alpha: set the weights of the least square error and the least absolute deviation as the loss function 'huber' ues a combination of the two to evaluate error. We test the alphas of 0.1, 0.3, 0.5, 0.7, 0.9 ##### Details on the parameters can be found in (<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html>). A finer grid search on more parameters will potentially result in better models. We use the current tuning method which is much less time-expensive but gives some satisfying results also.

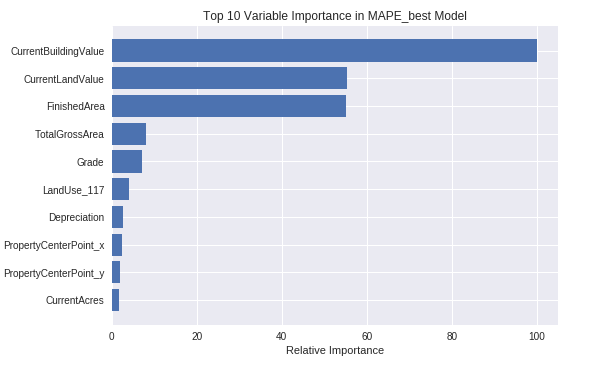
**Accuracies**

* The model with the lowest RSME is the model with loss='huber', n\_estimator=100, max\_depth=6, alpha=0.9. It has RSME=98382 and MAPE=20.07%.
* The model with the lowest MAPE is the model with loss='huber', n\_estimator=150, max\_depth=4, alpha=0.9. It has RSME=99671 and MAPE=19.51%.

**Feature Importance**

The following charts show the Top-10 important features in the model with the lowest RSME and the model with the lowest MAPE.





**Conlusions[¶](https://render.githubusercontent.com/view/ipynb?commit=dbff7b9cd20c3443aa932173edc9c0ee1e36f67c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f736f636861636861692f43617073746f6e6550726f6a656374312f646266663762396364323063333434336161393332313733656463396330656531653336663637632f4d616368696e654c6561726e696e67526573756c74732e6970796e62&nwo=sochachai%2FCapstoneProject1&path=MachineLearningResults.ipynb&repository_id=154411438&repository_type=Repository" \l "Conlusions)**

* The estimation of SalePrice of the Gradient Boosting Regression can achieve an RSME about $99000 and an MAPE about 20%.
* The dominating features in the Gradient Boosting Regression are CurrentBuildingValue, CurrentLandValue, and FinishedArea.
* Other features that are less significant but also helpful in the estimation of sale prices are TotalGrossArea, Grade of the housing property, Depreciation of the housing property, CurrentAcres, Location, IsMobileHome(LandUse\_117), YearBlt.

**XGBoosting Regression**

**Parameter Tuning**

We search models with the the lowest RMSE and the lowest MAPE. Three parameters are chosen to be tuned.[¶](https://render.githubusercontent.com/view/ipynb?commit=dbff7b9cd20c3443aa932173edc9c0ee1e36f67c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f736f636861636861692f43617073746f6e6550726f6a656374312f646266663762396364323063333434336161393332313733656463396330656531653336663637632f4d616368696e654c6561726e696e67526573756c74732e6970796e62&nwo=sochachai%2FCapstoneProject1&path=MachineLearningResults.ipynb&repository_id=154411438&repository_type=Repository#Next,-we-search-models-with-the-the-lowest-RMSE-and-the-lowest-MAPE.-Three-parameters-are-chosen-to-be-tuned.)

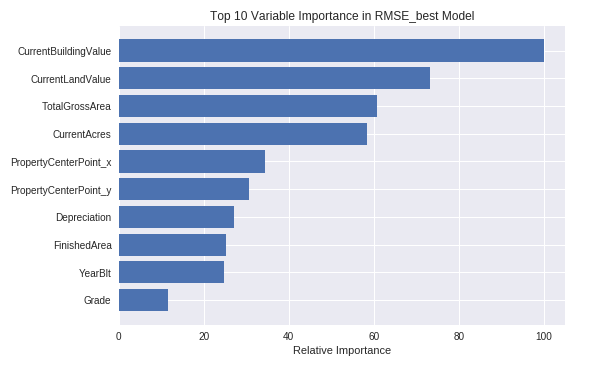
* N\_Estimators: Number of boosted trees to fit from 50 to 500 with a step of 50.
* Max\_depth: Maximum depth of the individual regression estimators from 2 to 8.
* Learning\_rate(Eta): Boosting learning rate(Step size shrinkage used in update to prevents overfitting). We test the etas of 0.1, 0.3, 0.5, 0.7, 0.9 ##### Details on the parameters can be found in (<https://xgboost.readthedocs.io/en/latest/python/python_api.html>). A finer grid search on more parameters will potentially result in better models. We use the current tuning method which is much less time-expensive but gives some satisfying results also.

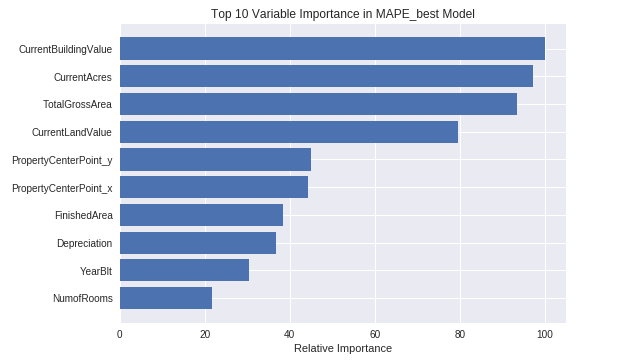
**Accuracies**

* The model with the lowest RSME is the model with n\_estimator=50, max\_depth=6, learning\_rate=0.1. It has RSME=97922 and MAPE=20.54%.
* The model with the lowest MAPE is the model with n\_estimator=100, max\_depth=6, learning\_rate=0.1. It has RSME=99236 and MAPE=20.00%.

**Feature Importance**

The following charts show the Top-10 important features in the model with the lowest RSME and the model with the lowest MAPE.





**Conlusions[¶](https://render.githubusercontent.com/view/ipynb?commit=dbff7b9cd20c3443aa932173edc9c0ee1e36f67c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f736f636861636861692f43617073746f6e6550726f6a656374312f646266663762396364323063333434336161393332313733656463396330656531653336663637632f4d616368696e654c6561726e696e67526573756c74732e6970796e62&nwo=sochachai%2FCapstoneProject1&path=MachineLearningResults.ipynb&repository_id=154411438&repository_type=Repository" \l "Conlusions)**

* The estimation of SalePrice of the XGBoosting Regression can achieve an RSME about $98000 and an MAPE about 20%.Compared to the classical Gradient Boosting Regression, XGBoosting Regression has the power of reducing RMSE but does not seem to reduce MAPE.
* The top 4 features in the XGBoosting Regression are CurrentBuildingValue, CurrentLandValue, CurrentAcres and TotalGrossArea. Other important features are FinishedArea, Depreciation of a housing property, Location, YearBlt, NumofRooms and Grade of a housing property.
* The top ten features for XGBoosting are similar to their counterpart in the classical Gradient Boosting. Unlike the classical Gradient Boosting where the top three features are dominating, the top 4 features in XGBoosting do not dominate among the top 10 features.

**Random Forest Regression**

**Parameter Tuning**

We search models with the the lowest RMSE and the lowest MAPE. Three parameters are chosen to be tuned.[¶](https://render.githubusercontent.com/view/ipynb?commit=dbff7b9cd20c3443aa932173edc9c0ee1e36f67c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f736f636861636861692f43617073746f6e6550726f6a656374312f646266663762396364323063333434336161393332313733656463396330656531653336663637632f4d616368696e654c6561726e696e67526573756c74732e6970796e62&nwo=sochachai%2FCapstoneProject1&path=MachineLearningResults.ipynb&repository_id=154411438&repository_type=Repository#Next,-we-search-models-with-the-the-lowest-RMSE-and-the-lowest-MAPE.-Three-parameters-are-chosen-to-be-tuned.)

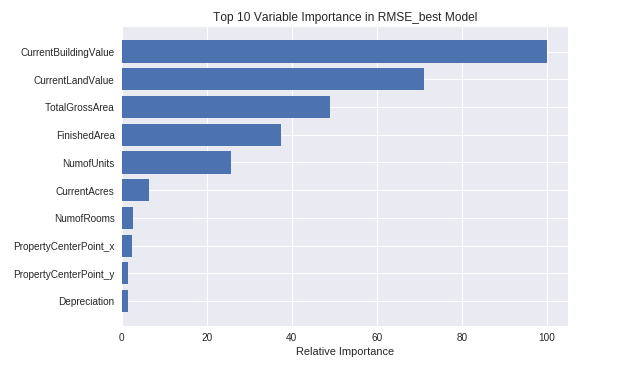
* N\_Estimators: Number of trees from 100 to 500 with a step of 100.
* Max\_depth: Maximum depth of the individual tree from 100 to 500 with a step of 100. The default is unlimited. We choose a small range to avoid overfitting.
* Min\_samples\_split: The minimum number of samples required to split an internal node from 2 to 10 with a step size of 2. Note that the default is 2. We choose larger numbers to avoid overfitting. ##### Details on the parameters can be found in <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html> . A finer grid search on more parameters will potentially result in better models. We use the current tuning method which is much less time-expensive but gives some satisfying results also.

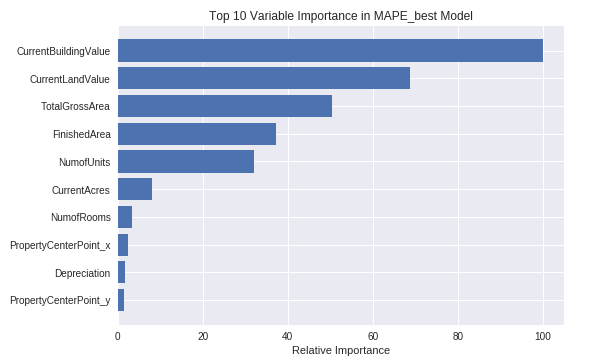
**Accuracies**

* The model with the lowest RSME is the model with n\_estimators=100, max\_depth=100, min\_samples\_split=10. It has RSME=105350 and MAPE=20.01%.
* The model with the lowest MAPE is the model with n\_estimators=500, max\_depth=100, min\_samples\_split=10. It has RSME=106277 and MAPE=19.98%.

**Feature Importance**

The following charts show the Top-10 important features in the model with the lowest RSME and the model with the lowest MAPE.





**Conlusions[¶](https://render.githubusercontent.com/view/ipynb?commit=dbff7b9cd20c3443aa932173edc9c0ee1e36f67c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f736f636861636861692f43617073746f6e6550726f6a656374312f646266663762396364323063333434336161393332313733656463396330656531653336663637632f4d616368696e654c6561726e696e67526573756c74732e6970796e62&nwo=sochachai%2FCapstoneProject1&path=MachineLearningResults.ipynb&repository_id=154411438&repository_type=Repository" \l "Conlusions)**

* The estimation of SalePrice of the XGBoosting Regression can achieve an RSME about $105000 and an MAPE about 20%.Compared to Boosting Algorithms, the Random Forest Regression models we have found are not doing better jobs in reducing RSME and MAPE.
* The top 5 features in our tuned Random Forest Regression models are CurrentBuildingValue, CurrentLandValue, TotalGrossArea, FinishedArea and NumofUnits. They are dominant over other features.
* The top ten features for XGBoosting are similar to their counterpart in the Boosting Algorithms.

**Summary**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=dbff7b9cd20c3443aa932173edc9c0ee1e36f67c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f736f636861636861692f43617073746f6e6550726f6a656374312f646266663762396364323063333434336161393332313733656463396330656531653336663637632f4d616368696e654c6561726e696e67526573756c74732e6970796e62&nwo=sochachai%2FCapstoneProject1&path=MachineLearningResults.ipynb&repository_id=154411438&repository_type=Repository#Final-Remarks)

* 1.With the author's best effort, the model with the least RMSE is achieved by XGBoosting Regression, which gives an RMSE of $97922 and an MAPE of 20.54%.
* 2.With the author's best effort, the model with the least MAPE is achieved by the classical Gradient Boosting Regression, which gives an RMSE of $99671 and an MAPE of 19.51%.
* 3.Based on what we have found so far, Boosting seems to beat Bagging in terms of accurate estimation of Sale Prices.
* 4.For practical use, the author would recommend clients to use the least MAPE model given by Gradient Boosting Regression(See 2) based on the finding that for most models the MAPE rarely gets below 20% and the fact that this model also has low RMSE.
* 5.Top features to look for in transaction of a housing properties are: CurrentBuildingValue, CurrentLandValue, FinishedArea, TotalGrossArea, CurrentAcres, YearBlt, Depreciation, Location(Latitude&Longitude), Grade, LandUse\_IsMoblieHome.

**Code**

The following is the code for XGBoosting Regression, including parameter tuning, feature importance plot etc. Code for data preparation, data analysis and other results could be found in the author’s Github repositories upon request.

**# Check the accuracy of the model with default settings.**

**import** **xgboost** **as** **xgb**

**from** **sklearn.metrics** **import** mean\_squared\_error

xgbreg = xgb.XGBRegressor(random\_state=0)

xgbreg.fit(X\_train, y\_train)

rmse = np.sqrt(mean\_squared\_error(y\_test, xgbreg.predict(X\_test)))

print("RMSE: **%.4f**" % rmse)

print("MAPE: **%.4f**" % np.mean(np.abs(xgbreg.predict(X\_test)-y\_test)/y\_test))

# Initial model

reg0 = xgb.XGBRegressor(random\_state=0)

reg0.fit(X\_train, y\_train)

rmse0 = np.sqrt(mean\_squared\_error(y\_test, reg0.predict(X\_test)))

mape0 = np.mean(np.abs(reg0.predict(X\_test)-y\_test)/y\_test)

# Initialize the best RMSE/MPE model and parameters

RMSEbest\_reg = reg0

RMSEbest\_rmse = rmse0

RMSEbest\_mape = mape0

RMSEbest\_n\_estimators = 100 # default

RMSEbest\_max\_depth = 3 # default

RMSEbest\_learning\_rate = 0.1 # default

MAPEbest\_reg = reg0

MAPEbest\_rmse = rmse0

MAPEbest\_mape = mape0

MAPEbest\_n\_estimators = 100 # default

MAPEbest\_max\_depth = 3 # default

MAPEbest\_learning\_rate = 0.1 # default

# Search for the best models

N\_estimators = np.arange(50,550,50)

Depth = np.arange(2,9,1)

Learning\_rate = np.arange(0.1,1,0.2)

for n\_est in N\_estimators:

for d in Depth:

for lr in Learning\_rate:

reg = xgb.XGBRegressor(n\_estimators=n\_est, max\_depth=d, learning\_rate=lr, random\_state=0)

reg.fit(X\_train, y\_train)

rmse = np.sqrt(mean\_squared\_error(y\_test, reg.predict(X\_test)))

mape = np.mean(np.abs(reg.predict(X\_test)-y\_test)/y\_test)

if rmse<RMSEbest\_rmse: #### Search for the model with lowest RMSE

RMSEbest\_reg = reg

RMSEbest\_rmse = rmse

RMSEbest\_mape = mape

RMSEbest\_n\_estimators = n\_est

RMSEbest\_max\_depth = d

RMSEbest\_learning\_rate = lr

if mape<MAPEbest\_mape: #### Search for the model with lowest MAPE

MAPEbest\_reg = reg

MAPEbest\_rmse = rmse

MAPEbest\_mape = mape

MAPEbest\_n\_estimators = n\_est

MAPEbest\_max\_depth = d

MAPEbest\_learning\_rate = lr

print("RMSE\_best n\_estimators = %.0f" % RMSEbest\_n\_estimators)

print("RMSE\_best max\_depth = %.0f" % RMSEbest\_max\_depth)

print("RMSE\_best learning\_rate = %.1f" % RMSEbest\_learning\_rate)

print("RMSE\_best rmse = %.4f" % RMSEbest\_rmse)

print("RMSE\_best mape = %.4f" % RMSEbest\_mape)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print("MAPE\_best n\_estimators = %.0f" % MAPEbest\_n\_estimators)

print("MAPE\_best max\_depth = %.0f" % MAPEbest\_max\_depth)

print("MAPE\_best learning\_rate = %.1f" % MAPEbest\_learning\_rate)

print("MAPE\_best rmse = %.4f" % MAPEbest\_rmse)

# Find Top-10 features

RMSEbest\_reg = xgb.XGBRegressor(n\_estimators=50,max\_depth=6,learning\_rate=0.1,random\_state=0)

RMSEbest\_reg.fit(X\_train, y\_train)

RMSEbest\_feature\_importance = RMSEbest\_reg.feature\_importances\_

# make importance relative to the max of importance

RMSEbest\_feature\_importance = 100.0 \* (RMSEbest\_feature\_importance / RMSEbest\_feature\_importance.max())

sorted\_idx = np.argsort(RMSEbest\_feature\_importance)[-10:]

print(X.columns[sorted\_idx])

#Plot Top-10 features

**import** **matplotlib.pyplot** **as** **plt**

pos = np.arange(sorted\_idx.shape[0]) + .5 plt.barh(pos, RMSEbest\_feature\_importance[sorted\_idx], align='center')

plt.yticks(pos, X.columns[sorted\_idx])

plt.xlabel('Relative Importance')

plt.title('Top 10 Variable Importance in RMSE\_best Model')

plt.show()

# The code for finding and plotting plot of Top-10 features in the model with the lowest MAPE is similar to the above and is hence omitted.