**Zillow Prize: Zillow’s Home Value Prediction (Zestimate)**

**Kamran Ahmad, Akshaykumar Patel**

{ahmad.k, patel.ak}@husky.neu.edu

INFO 6105, Fall 2018, Northeastern University

1. **Abstract**

Property cost have important impact on individual, families, businesses and governments. Companies like Zillow have implemented methods to provide automated estimates of the housing prices without any help of professionals. Zillow is one of the trusted marketplace for people to discover various housing property around them. Zillow’s Zestimate home value prediction helps user to see the predicted price for the property and help them to decide to choose the property.

In this paper, we develop models to predict the logerror value for the housing data points where log error = log(Zestimate)- log (Sales price). The original dataset was taken from the Kaggle competition and after performing encyclopaedic analysis, we enhanced it by implementing various data cleaning techniques like KNN methodology to fill out the void data points. Since it is regression problem, we have deployed various regression algorithms like Random forest Regressor, Gradient boosting Regressor, Ada Boosting Regressor and XG Boosting Regressor. After comparing all the results from these models, we found that for XG Boosting regressor, we have achieved **0.000742** error score (near to zero) for the dataset. Apart from the implementing regressor algorithms, we have also done hyperparameter tuning for each algorithm which was not implemented in Kaggle competition. Hyperparameter tuning helped us to choose the best parameter for that algorithm and achieve better result for the dataset.

This report contains the introduction to the problem and comprehensive analysis on the dataset followed by data cleaning. Then we have applied several regression models and related hyperparameter tuning for each model. After comparing the score of each model, we proposed the best model which is XG Boosting regressor with a error score of **0.000742**.

1. **Introduction**

Zillow is the leading real estate and rental marketplace dedicated to empowering consumers with data and knowledge around the place they call home. Zillow’s Zestimate home valuation has shaken up the U.S. real estate industry since first released 11 years ago. “Zestimates” are estimated home values based on 7.5 million statistical and machine learning models that analyse hundreds of data points on each property. And, by continually improving the median margin of error (from 14% at the onset to 5% today).

As part of this project our goal is to analyse various Machine learning algorithms and we will be able to apply different Machine learning algorithms and we will be able to compare them and see the results to know which algorithm is more suited in predicting the future sale prices of homes. In this Kaggle competition, Submissions are evaluated on Mean Absolute Error between the predicted log error and the actual log error.

The log error is defined as:

logerror=log(Zestimate)−log(SalePrice)logerror=log(Zestimate)−log(SalePrice)

and it is recorded in the transactions training data.

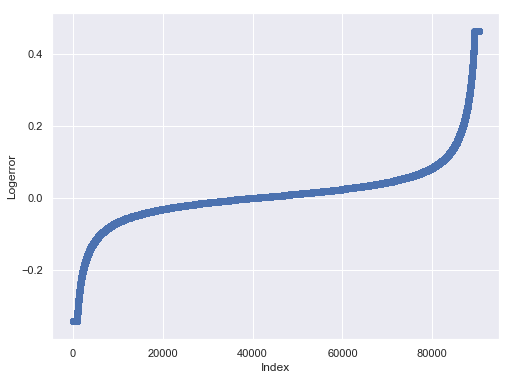
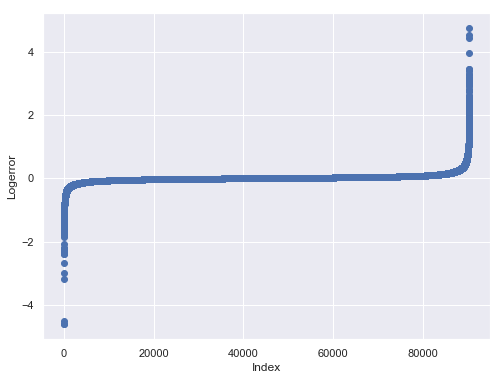
We have used housing data of Los Angeles, Orange and Ventura, California over the period of 2016-17 which contain around 10,000 data points with 58 columns of property. As the dataset contained more number of null values and inconsistence data which could affect the prediction, we performed data cleaning techniques to generate data using KNN algorithm. We have also replaced few null values with either with mean or dropped the irrelevant entries. Further Exploratory data analysis performed along with the feature importance and Multi-Collinearity check. For prediction, we used supervised learning models like Random forest Regressor, Gradient boosting Regressor, Ada Boosting Regressor and XG Boosting Regressor along with their respective hyperparameter tuning. After comparing the results among these models, we concluded that xyz is best fit for the prediction.

1. **Exploratory Data Analysis:**

Exploratory Data Analysis (EDA) is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to

1. Maximize insight into a data set
2. Uncover underlying structure
3. Extract important variables
4. Detect outliers and anomalies
5. Test underlying assumptions
6. Develop parsimonious models
7. Determine optimal factor settings.

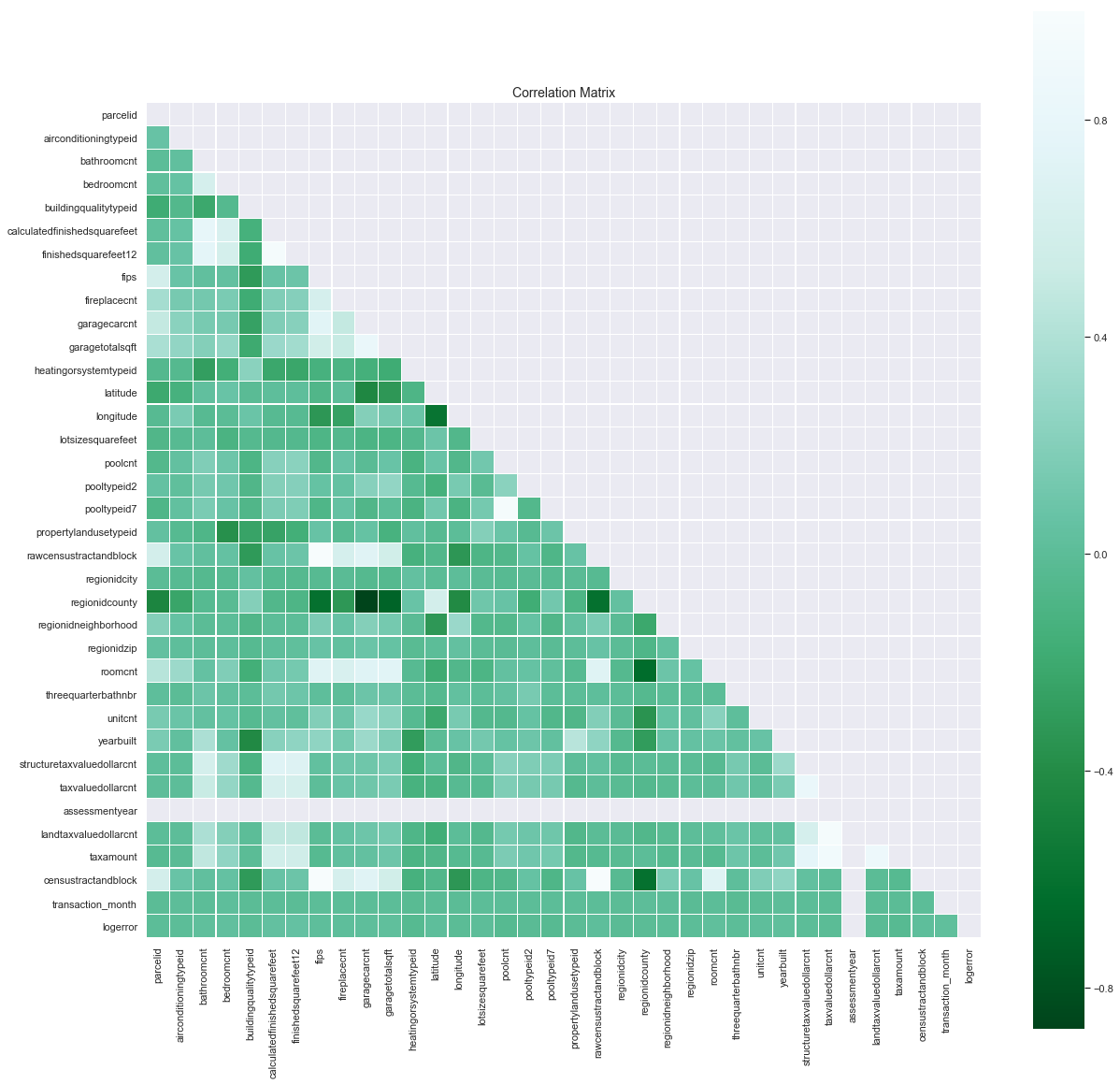
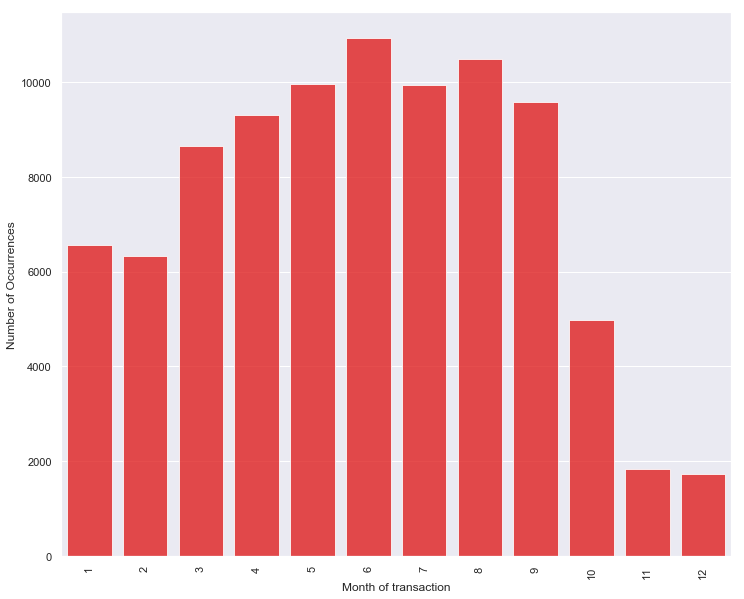
The EDA approach is precisely that--an approach--not a set of techniques, but an attitude/philosophy about how a data analysis should be carried out. We started with univariate analysis on our dependent variable ‘logerror’. Univariate analysis is the simplest form of analysing data. “Uni” means “one”, so in other words your data has only one variable. It doesn’t deal with causes or relationships (unlike regression) and its major purpose is to describe; it takes data, summarizes that data and finds patterns in the data. The graph shows how “logerror” is scattered in the sample dataset. By plotting the target value in the graph shows the outliers reside in the dataset which might affect the model fitting. From this, we decided threshold (minimum-1% and maximum 99%) for the dataset to remove the outliers. Below are the graphs which shows the before outliers and after outliers removal of “logerror” column from the dataset.



**Figure 1: Before Outlier Removal Figure 2: After Outlier Removal**

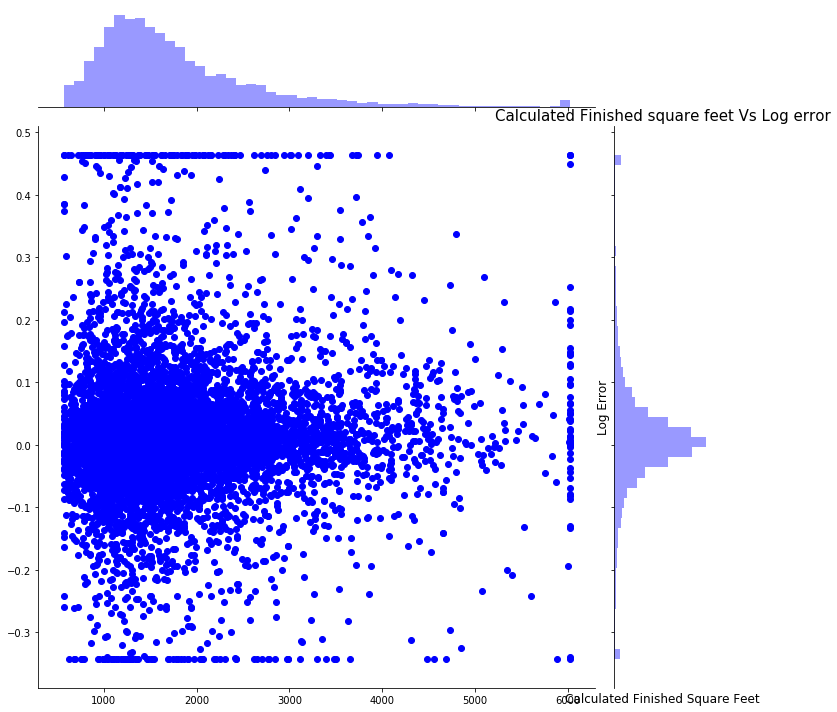
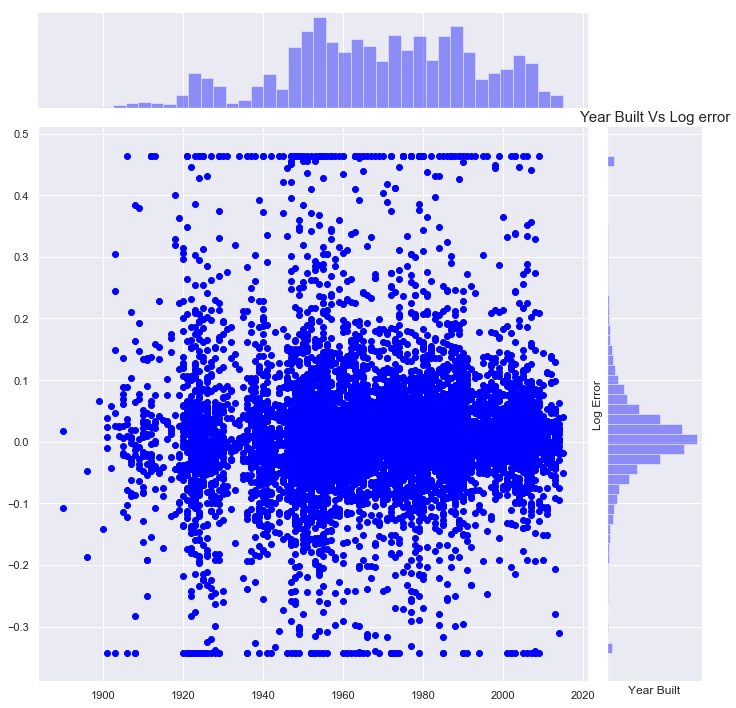
To check the distribution of data points in the dataset as per the date, we plotted the bar chart (Figure 3) with shows the number of transaction done per month. Data has all the transactions before October 15, 2016, plus some of the transactions after October 15, 2016. So, we have shorter bars in the last three months.

To understand the correlation between different features, present in our dataset, we plotted a heat map and discovered that the correlation between the parameters 'fips', 'fireplacecnt', 'garagetotalsqft', ‘gargagecarcnt’ and 'assessmentyear' is very strong which is shown in the heat map (Figure 4) drawn below.

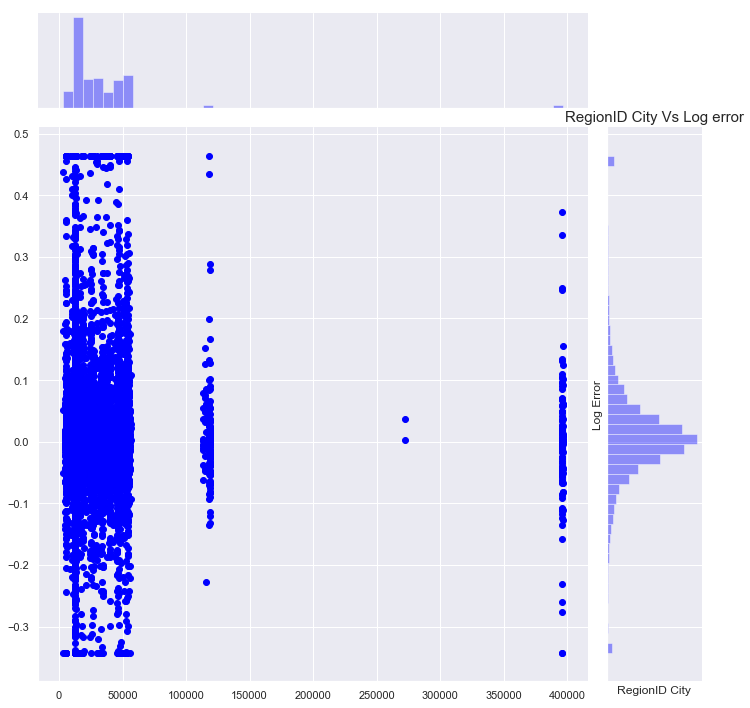
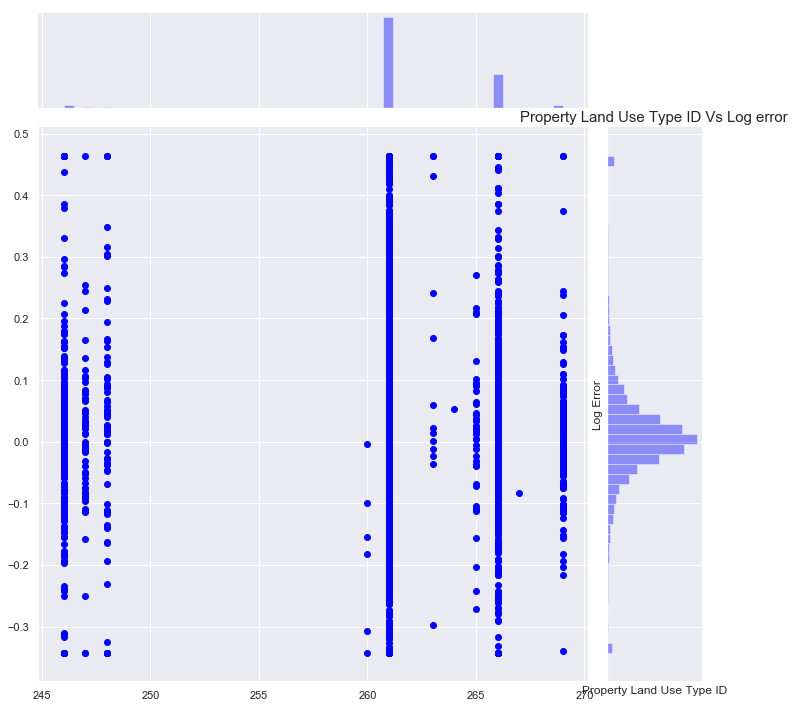


**Figure 3: Number of Occurrence per month Figure 4: Multicollinearity check among the features**

We conducted the bivariate analysis. Bivariate analysis is the simultaneous analysis of two variables (attributes). It explores the concept of relationship between two variables, whether there exists an association and the strength of this association, or whether there are differences between two variables and the significance of these differences. In our bivariate analysis we have studied how our log error varies with some significant variables in our dataset and below are the respective graphs for the same

**Figure 5: Calculated Finished Square Feet Vs Log Error Figure 6: Year Built Vs Log error:**

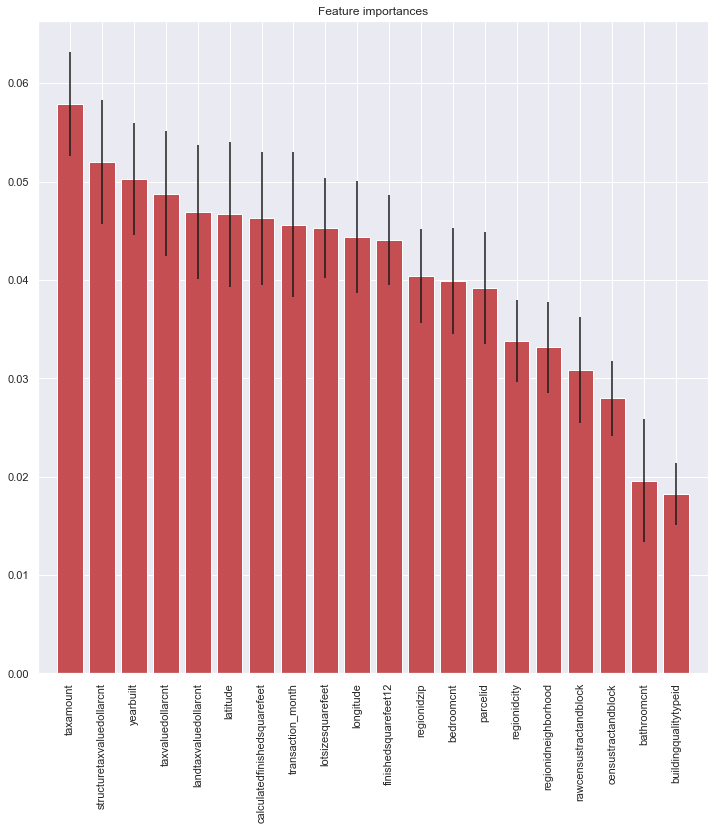
**Figure 7: Region Id City Vs Log Error Figure 8: Property Land Use Type ID Vs Log error**

**(4) Features Importance and Selection:**

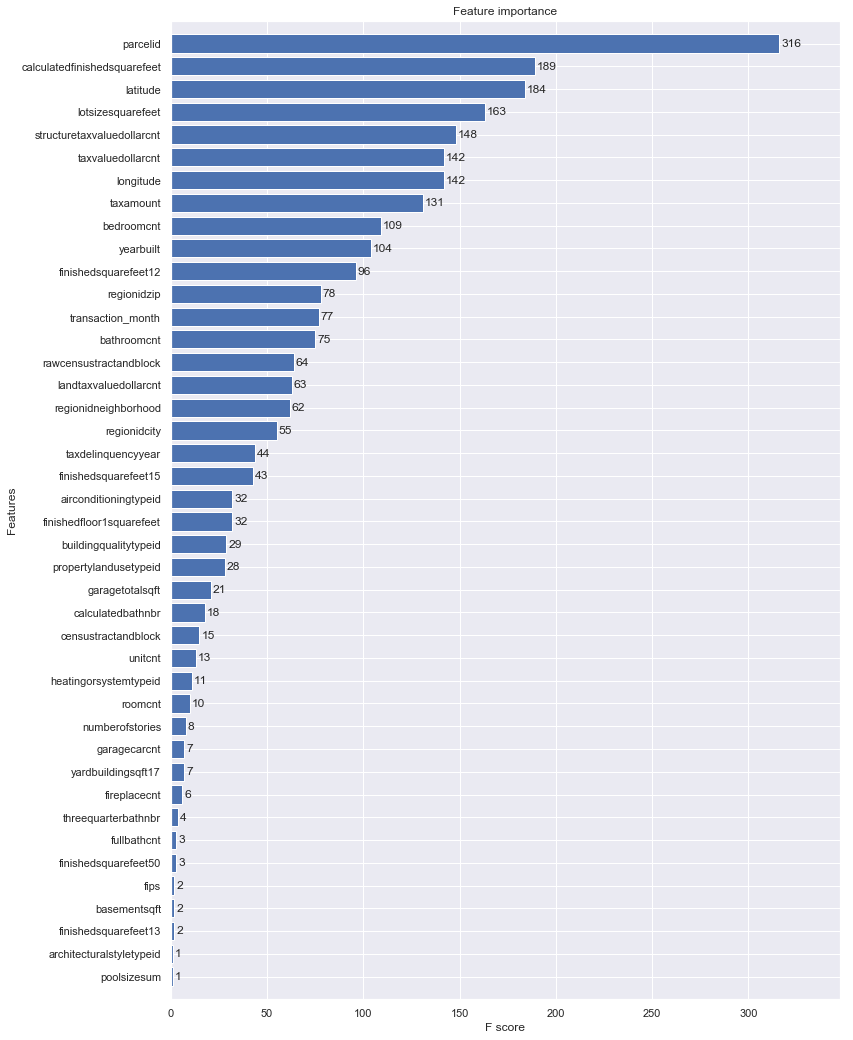
The number of features in the current dataset is 58 which are related to geographical information of the property. If we use all the features to train the model, then it might overfit the model. To avoid such problem, we have analysed the all the feature using Extra tree regressor and XG Boost Regressor algorithm to get the important feature from the dataset which contribute most to do prediction of target variable.

We used gradient boosting to retrieve importance scores for each attribute. Generally, importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, the higher its relative importance. This importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other. Importance is calculated for a single decision tree by the amount that each attribute split point improves the performance measure, weighted by the number of observations the node is responsible for. The feature importance’s are then averaged across all the decision trees within the model.

Below is the bar chart of the relative importance’s of the features.



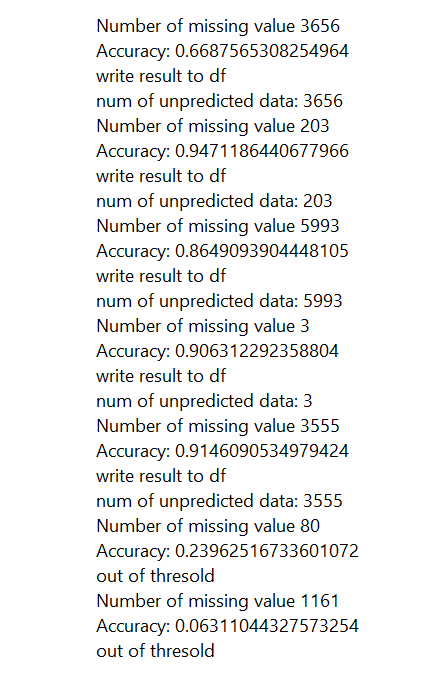
We have also plotted graph using xgb.importance, that returns a graph of feature importance measured by an f score. F Score is a metric that simply sums up how many times each feature is split on.



**(4) Data Cleaning:**

We have analysed the various features from the dataset and found few features having redundant data or null values in it. We used various techniques to remove irrelevant information from that features which can help to do better prediction. Techniques includes replacing the null values with relevant default values, taking mean for the feature for null value, removing the feature which is having same kind of information as the other feature, calculating null values from the other feature with are related to each other and implementing the classification algorithm like KNN classifier to predict the null vales based on training the KNN model using non missing value to predict the missing value. We have also checked the accuracy of KNN model which predict the missing value to check if accuracy is beyond the threshold limit (50 % accuracy). The predicted data will be replaced with the missing value if its KNN model have accuracy more than 50%.

Result which shown in the left side image are from the KNN classifier algorithm which shows the Number of missing value of target feature, accuracy of the KNN classifier algorithm and number of predicted data. Form the result we can see that; last two features are not having model accuracy more than 50% so the function rejects those features to update the predicted data from the KNN model. Here, latitude and longitude are the base feature which is being trained in the KNN model to predict the missing value since these features have very less value in the dataset.



**Figure 9: KNN Algorithm result**

**(5) Log error Prediction:**

We have used various Boosting and Bagging algorithm like Gradient Boosting regressor, XG Boosting regressor, Ada Boosting regressor and Random forest algorithm. We have done hyperparameter for each algorithm using GridSearchCV function to get the best parameters for each algorithm. Below is the description of the algorithms which is used in our research.

**Random Forest Regression:**

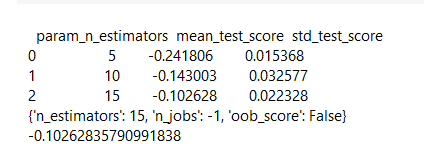
A random forest is a meta estimator that fits several classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

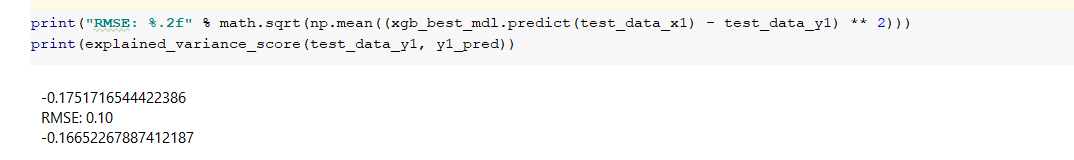
We have used below parameters to improve the predictive accuracy.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **n\_estimators** | integer, optional (default=10) | The number of trees in the forest. |
| **n\_jobs** | int or None, optional (default=None) | The number of jobs to run in parallel for both fit and predict. None` means 1 unless in a [joblib.parallel\_backend](https://joblib.readthedocs.io/en/latest/parallel.html#joblib.parallel_backend) context. -1means using all processors. |
| **oob\_score** | bool, optional (default=False) | whether to use out-of-bag samples to estimate the R^2 on unseen data. |

Random [Forest](http://i.viglink.com/?key=9c219f6b27c7571a83003d3349a56c73&insertId=0c032a91dd9e6a61&type=KW&exp=-100%3ACILITE%3A15&libId=jpojloxg01021u9s000DAc57kip4y&loc=https%3A%2F%2Fwww.r-bloggers.com%2Fhow-random-forests-improve-simple-regression-trees%2F&v=1&iid=0c032a91dd9e6a61&out=https%3A%2F%2Fwww.brownells.com%2Fsearch%2Findex.htm%3Fk%3Dforest&ref=https%3A%2F%2Fwww.google.com%2F&title=How%20Random%20Forests%20improve%20simple%20Regression%20Trees%3F%20%7C%20R-bloggers&txt=%3Cspan%3EForest%3C%2Fspan%3E) solves the instability problem using bagging. We simply estimated the desired Regression Tree on many bootstrap samples (re-sample the data many times with replacement and re-estimate the [model](http://i.viglink.com/?key=9c219f6b27c7571a83003d3349a56c73&insertId=fa8deda29c24eb44&type=KW&exp=-100%3ACILITE%3A15&libId=jpojloxg01021u9s000DAc57kip4y&loc=https%3A%2F%2Fwww.r-bloggers.com%2Fhow-random-forests-improve-simple-regression-trees%2F&v=1&iid=fa8deda29c24eb44&out=https%3A%2F%2Fwww.brownells.com%2Fsearch%2Findex.htm%3Fk%3Dmodel&ref=https%3A%2F%2Fwww.google.com%2F&title=How%20Random%20Forests%20improve%20simple%20Regression%20Trees%3F%20%7C%20R-bloggers&txt=%3Cspan%3Emodel%3C%2Fspan%3E)) and made the final prediction as the average of the predictions across the trees. The Random [Forest](http://i.viglink.com/?key=9c219f6b27c7571a83003d3349a56c73&insertId=0c032a91dd9e6a61&type=KW&exp=-100%3ACILITE%3A15&libId=jpojloxg01021u9s000DAc57kip4y&loc=https%3A%2F%2Fwww.r-bloggers.com%2Fhow-random-forests-improve-simple-regression-trees%2F&v=1&iid=0c032a91dd9e6a61&out=https%3A%2F%2Fwww.brownells.com%2Fsearch%2Findex.htm%3Fk%3Dforest&ref=https%3A%2F%2Fwww.google.com%2F&title=How%20Random%20Forests%20improve%20simple%20Regression%20Trees%3F%20%7C%20R-bloggers&txt=%3Cspan%3EForest%3C%2Fspan%3E) adds a new source of instability to the individual trees. Every time we calculate a new optimal variable-observation point to split the tree, we do not use all variables. Instead, we randomly select 2/3 of the variables. This will make the individual trees even more unstable, but bagging benefits from instability as each variable gets several chances to be in the model.

Below are our best parameters using random forest regressor:





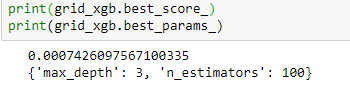
**XGBoost Regressor:**

XGBoost is well known to provide better solutions than other machine learning algorithms. In fact, since its inception, it has become the "state-of-the-art” machine learning algorithm to deal with structured data.

XGBoost (Extreme Gradient Boosting) belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework at its core. It is an optimized distributed gradient boosting library. There are a plethora of tuning parameters for tree-based learners in XGBoost and below are the ones which we have used.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **n\_estimators** | int (default=100) | number of trees you want to build. |
| **max\_depth** | integer, optional (default=3) | determines how deeply each tree can grow during any boosting round. |

Below are our best parameters using XGBoost:



Score for the XGBoost model is as shown below:



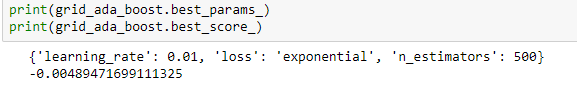
**AdaBoost Regressor:**

AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

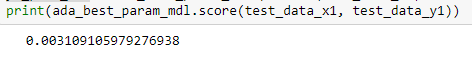
The important parameters to vary in an AdaBoost regressor are learning\_rate, loss and n\_estimators. As with the previous algorithms, we will perform a randomized parameter search to find the best scores that the algorithm can do. Below are the parameters which we have used.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| loss | {‘linear’, ‘square’, ‘exponential’}, optional (default=’linear’) | number of trees you want to build. |
| learning\_rate | float, optional (default=1.) | Learning rate shrinks the contribution of each regressor by learning\_rate. There is a trade-off between learning\_rate and n\_estimators. |
| n\_estimators | integer, optional (default=50) | The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. |

Below are the AdaBoosting best parameters and score:



Score:

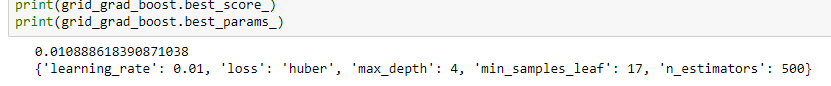


**Gradient Boosting:**

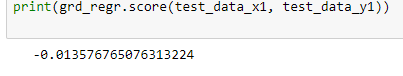
GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n\_classes\_ regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Below are the parameters which we have used.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **n\_estimators** | int (default=100) | The number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting, so a large number usually results in better performance. |
| learning\_rate | float, optional (default=0.1) | learning rate shrinks the contribution of each tree by *learning\_rate*. There is a trade-off between learning\_rate and n\_estimators. |
| loss | LossFunction | The concrete LossFunction object. |
| max\_depth | integer, optional (default=3) | Maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. Tune this parameter for best performance; the best value depends on the interaction of the input variables. |

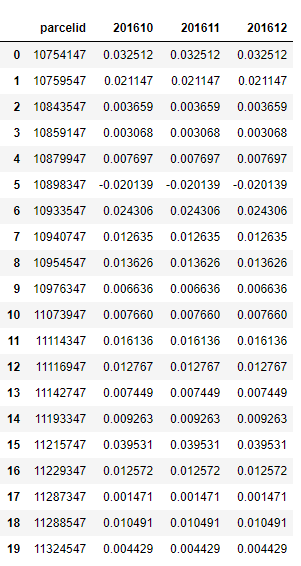
Below are the gradient boosting best score and best parameters:



Score:



From the above snippets of various supervised algorithm, we are convinced that XG Boosting regressor is giving better error-score which is near to zero (**0.000742**). We will use XG Boosting regressor model to take the predicted log-error value which will be assigned to submission.csv file which contain the parcelid of each property sold out during that time. Below results show the few log-error values for the parcelid from the submission.csv file.



**Figure 10: submission.csv file**

**(6) Discussion:**

From EDA, we can see that most of the transaction happened before October and we have detected some important feature which can be used during the prediction of logerror for the dataset. We have used classification algorithm to predict the missing value in the dataset.

In this paper, we implemented four supervised learning models to provide solution of the regression problem of predicting the log-error for the parcelid which represents the housing property across various location. We have tried to tune each regression models with hyperparameter tuning to get the best hyperparameter associated with the model to achieve better accuracy. As compared to Kaggle competition, we have implemented the hyperparameter tuning for each algorithm and achieved better results **(0**.**000742 error-score**) with XG Boosting regressor algorithm.

**(7) Citations:**

[1] [https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html%20)

[2] [https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html%20)

[3] <https://en.wikipedia.org/wiki/XGBoost>

[4] [https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html%20)

[5] [https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm%20)

[6] [https://www.geeksforgeeks.org/univariate-bivariate-and-multivariate-data-and-its-analysis/](https://www.geeksforgeeks.org/univariate-bivariate-and-multivariate-data-and-its-analysis/%20)

[7] <https://en.wikipedia.org/wiki/Exploratory_data_analysis>

[8] <https://www.kaggle.com/c/zillow-prize-1>

[9] <https://en.wikipedia.org/wiki/Hyperparameter_optimization>

[10] <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html>

[11] <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>

[12] <https://www.kaggle.com/nikunjm88/carefully-dealing-with-missing-values/notebook>