Zillow's Home Value Prediction

By *Aneeq ur Rehman, Boyan Wei, Fangling Liu,*
Wendong Liu, Xin Sun

Supervisor: *Dr. Mauricio Alvarex*

December 13, 2017

Contents

Abstract

In this project, the error estimate was computed for predicting the price for real housing estates across three regions in the United States. Data for these housing estates is acquired through Zillows website. The company forecasts the price of real housing estates every year and computes a metric of accuracy on its forecast known as logerror. In this project, a couple of classical models were applied to predict the price of the real estate housings in the form of this metric. In addition to this, different features or combinations of features to train models are identified to improve the accuracy of these predictions. Finally, reverse engineering was performed to analyze areas where the models does not generalize.

1 Introduction

Zillow is an online real estate data base company founded in 2006. The company gives an estimated price of different housing estates scattered across the United States, and gives as estimated price known as Z-estimate, for the sales prices for real estate housing every month. It then computes a metric of accuracy on its forecast known as log error. The log error is the difference in the prices between the actual sales price of the housing estate and the estimated price given by the Zillows algorithm.

The available data for this project is dispersed across three regions which are Los Angeles, Orange Ventura and California. Data is available in the form of csv files which are processed and explained later in the upcoming sections of the report. The motivation for this project lies in the design and development of predictive models to estimate the log error for the real housing estates across United States. This is known as **predictive modeling**. The predictions themselves will be in the form of the log error for training data - which are given in the form of csv files for months October to December across two years; 2016 and 2017. Another task is to identify which features are highly relevant for predictive model building and hence give an accurate prediction of log error. This is known as **feature selection**.

In order to perform the tasks mentioned above, a key task is to prepare the data for machine learning algorithms. This includes data cleansing, which involves removing outliers, dealing with missing values and handling the text features from the data set to train the model. Dealing with missing values itself affects model performance. The data is split into three different sets with different percentages $(30\%, 50\%, 97.5\%)$ of missing values removed and these are explained later in the upcoming sections of the report. After data preparation, the data is sampled and divided into training and test sets for model training and evaluation. Data is then sampled by creating stratified samples into test and training sets.

After data preparation and sampling, predictive modeling is carried out and the value of the log error is computed using different models, and two metrics to measure model performance - known as the **R2** score and the **Root mean square error (RMSE)**- are computed. Furthermore the predictions of the model are analyzed. The predictions are then analyzed such that predictions which fall out of two standard deviations of the error (the difference in actual and predicted value of the response variable) are considered as a **bad** predictions and values within two standard deviations of the error are considered as **good** predictions. Finally the bad predictions are analyzed to see which features of the housing estate, the model does not perform well on. This process is known as **reverse engineering**.

In addition to predictive model building, feature selection is performed on different models by using two different methods known as the **Wrapper** methods and the **Filter** methods. The details of these methods are also explained later in the upcoming sections but the idea of feature selection is to identify features or combinations of features to improve model performance. Two metrics are used to measure model performance which are the R squared value (R^2) and mean square error (MSE).

2 Literature Review

In this section, the literature review to carry out the project is explored.

2.1 Data Preparation

2.1.1 Data visualization

In any machine learning project, it is very important to visualize the underlying data and decide the machine learning algorithm to use. It helps to refine models by including important features or devising new features that help to visualize data and to train models more effectively for unseen data. The underlying data may also require some numerical transformation [1] for better user interpretation. Numerical transformation involves changing the underlying distribution to some other distribution that is more suitable for data analysis. [1] It can also involve the inclusion of new features that can produce meaningful results for the users.

An analysis of the data is also pivotal to understand the kind of machine learning task to carry out, for example, visualization of data can help us to identify whether the data has clusters or scatter or that it follows a particular pattern. This helps in the identification of the machine learning task to perform i.e supervised learning or unsupervised learning. In addition to this, visualization of data also helps in detecting correlations and relationships among different variables in the data set all of which are important factors to consider when building models.

2.1.2 Data Cleansing

The next task is data cleansing which entails the following:

- Filtering outliers
- Handling Missing attributes
- Encoding Text Attributes
- Creating new attributes

2.1.2.1 Filtering Outliers: Data can have points that deviate from the norm. The occurrence of such points in the data set is usually due to an error in the recording of the readings or some mistake in the observed data. Such data points are known as outliers or anomalies. The rule of thumb to remove outliers is to calculate the standard deviation of the data set and any data points that lie out of three standard deviations of the data set are considered as outliers and are removed. In order to remove outliers from the data set, care needs to be taken that the removal of the outliers does not affect the assumptions and results on which the model was based on initially. If both results and assumptions of the model change, then there is a need to test the model with and without the outliers. [2]

2.1.2.2 Handling Missing attributes: Another important aspect of data cleansing is the handling of missing values inside the data set. There are three options to take care of missing values in the data set, which are as follows: [3]

1. Remove the corresponding missing values in the data set.
2. Remove the whole corresponding feature or attribute if the missing values exceed a particular percentage.

3. Set the missing value to some other value such as the median or mode or the mean of that attribute. The value can also be set to zero in some circumstances.

4. A probabilistic approach can also be carried out to remove missing values inside the data set.

2.1.2.3 Encoding Text Attributes: Most of the machine Learning algorithms work with numbers rather than text attributes and this is because dealing with numerical attributes is computationally faster compared to dealing with text attributes. This is preferred even more when there are large data sets and when the machine learning algorithms have to span the whole data set. In these type of circumstances, it is preferred to encode the text attribute or labels to numbers. There are two methods available to encode text attributes and they are outlined as follows:

Integer Encoding: In Integer Encoding, the text attributes are encoded with an integer. This type of encoding is preferred on ordinal variables i.e variables where values occur in a predefined order [4]. Each label is represented by an integer.

One Hot Encoding: The second method is known as one Hot Encoding and is used for those variables when there is no natural order among the variables. [4] The text attributes are converted into binary vectors and this methodology is known as one hot encoding. The vectors are binary variables for each unique text label.

Both of these Encoding schemes have a trade off between the models run time complexity and the reliability of the predictions offered by the model, for instance, for very large data sets using hot encoding can be computationally expensive for the model as it may take a long time to run, this is made even worse for those algorithms which span the whole data set several times over a large number of iterations. In these types of cases integer encoding may be useful but again the drawback of integer encoding is that for variables which have no ordered relationship among their values, the model is trained on the assumption that there is a relationship among the values of the variables. In this case, the model will give poor predictive results and may even completely fail to generalize for unseen data.

In addition to the above,the data also needs to be checked for data redundancy. Duplicate values inside the data set need to be removed.

2.1.3 Data Sampling

Models train and perform well when the training data that has a sufficient number of instances of every feature inside the data set. Training improves when there are an equal number of instances of the prediction variable in the test and training data sets. The more the number of equal instances of different scenarios for the model, the better the training of the model and the more it generalizes on unseen data. Some of the ways to sample data include:

- Random Sampling

- Stratified Sampling

Random Sampling : In random sampling, the data is shuffled randomly and some portion of the data is used for training and some for testing and validation. This type of sampling can introduce a bias in the predictions, as there is a possibility that the predictions generated by the model are skewed. This is because the training data may end up containing instances of one or a few scenarios only hence the model may fail to generalize completely.

Stratified Sampling : In this type of Sampling, equal instances of some attributes are picked for the training and test data set. This reduces the amount of bias in the training and validation data sets. Each equal instance of an attribute is placed under a category or strata. Care needs to be taken that each strata is large enough and there are a small number of stratas [3]. This is a more preferred approach for modeling and aids the model to generalize well on unseen data.

2.2 Predictive Modeling

The task of Model building entails dividing the processed and clean data into training and test sets. A model with a particular type of machine learning algorithm is used such that the model is trained on the training data set and the model is tested on the test data set. The performance of the model is measured using metrics such as training error for the performance of the model on the training set and generalization error for the model performance on the test error. If both the training and the generalization error are low, then this is an indicator that the model selected works well for the chosen data set.

There are different techniques available for effective model building and these can be summed up as follows:

- Hold Out Validation
- K Fold Cross Validation
- Leave One Out Cross Validation

2.2.1 Hold Out Validation

In Hold Out Validation the data is divided into training and test sets respectively keeping a bench mark of 80-20 ideally such that 80% of the data is used for training and 20% of data is used for validation. Since a portion of data is taken out for test, so this known as hold out validation.

2.2.2 K Fold Cross Validation

In K Fold Validation, the data is divided into K Folds and trained on $K - 1$ of these folds. The model is tested on the fold left and the generalization/validation error is computed. After computing the generalization error the portion of data set used for test is put back and then another part of the data set is used for test. This process is repeated K times and the generalization error is averaged out.

2.2.3 Leave One Out Cross Validation

In this type of validation, the process is repeated n times, where n is the number of data points in the data set. The model is trained on $n-1$ of the data points and tested at one data point each time. The errors are then averaged out. This method is not advisable when the the number of data points are large.

2.2.4 Metrics to measure Model Performance

The following entails the metrics to measure model performance:

- Mean Square Error
- R Squared (R^2) Statistic

Mean Square Error: The mean square error is the square of the difference between the actual and the predicted value. It measures the variance around the regression line. The mean square error can be used as a generalization error when the model is deployed on the test data set and the lower the mean square error, the better is the performance of the model. The following equation gives the MSE for the model [5].

$$Mean\ Square\ Error(MSE) = \frac{\sum_{i=0}^{i=n}(Actual - Predicted)^2}{Total\ Samples} = \frac{\sum_{i=0}^{i=n}(y_i - f(x_i))^2}{n} \tag{1}$$

Here $f(x)$ is the prediction function for the selected model and x_i is the is i-th value of x in the test data set. The standard deviation for MSE is known as RMSE and is calculated as follows [6]:

$$Standard\ Deviation\ = \sqrt{\frac{\sum_{i=0}^{i=n}(Actual - Predicted)^2}{Total\ Samples - Parameters\ Estimated}} \tag{2}$$

$$= \sqrt{\frac{\sum_{i=0}^{i=n}(y_i - f(x_i))^2}{n-p}}$$

The denominator of the equation here has number of samples minus the parameter to be estimated. So for each parameter to be estimated, a degree of freedom is lost [6].

R- Squared (R^2) Statistic: The R-squared statistic is another measure to measure the accuracy of the model performance. It explains how much of the variability of the response variable (the log error predictions in this case) is explained by the model[7]. The R^2 statistic for a model performance is given as follows: [6]

$$R^2 = 1 - \frac{Residual\ sum\ of\ Squares\ (RSS)}{Total\ sum\ of\ squares\ (TSS)} \tag{3}$$

Here RSS measures the total variance in the response variable that is left unexplained by the model and the total sum of squares is the total variance in the response variable [6]. The higher the R-squared value for a model, the more the model performs better and the higher the accuracy of the predictions.

2.3 Feature Selection

Feature selection could be also called variable selection or attribute selection which is an important data reprocessing task. It is the selection of the features which are most relevant to the target value. Suppose dealing with a supervised learning problem, the number of features is very large, but there may be only a few characteristics that will affect the results. Even dealing with a simple linear category, if the number of sample features exceeds n, but the VC dimension of the function is still n, then unless the number of training sets is greatly expanded, there could lead to the problem of over-fitting. In such cases, feature selection could be used to reduce the number of features.

In general, considering the selection of features from some aspects: On the one hand, consider whether the features diverge. If a feature does not diverge, for example, the variance is close to 0, which means that there are no differences between the values of this feature, this feature is not useful for the prediction. On the other hand, consider the correlation between features and target values. It is obvious that the features with high relevance should be preferentially selected. The features with low relevance should be dropped.

Feature selection has lots of benefits as it helps to understand the underlying data really well. Furthermore selection of features also helps to reduce requirements to store large sets of data. The models training and evaluation times are also significantly reduced and selection of relevant features prevents the creation of overly complex models and avoids over-fitting problems and improves predictions [8].

There are some differences between feature selection and dimensionality reduction. Both approaches try to reduce the number of features. Dimensionality reduction achieves this based on the relationship between features, such as creating a new feature by combining different but related ones, which changes the original feature space. While the method of feature selection is to select a subset from the original feature set, without changing the original feature space[9].

3 Experimental Design and Methodology

3.1 Block Diagram of work flow

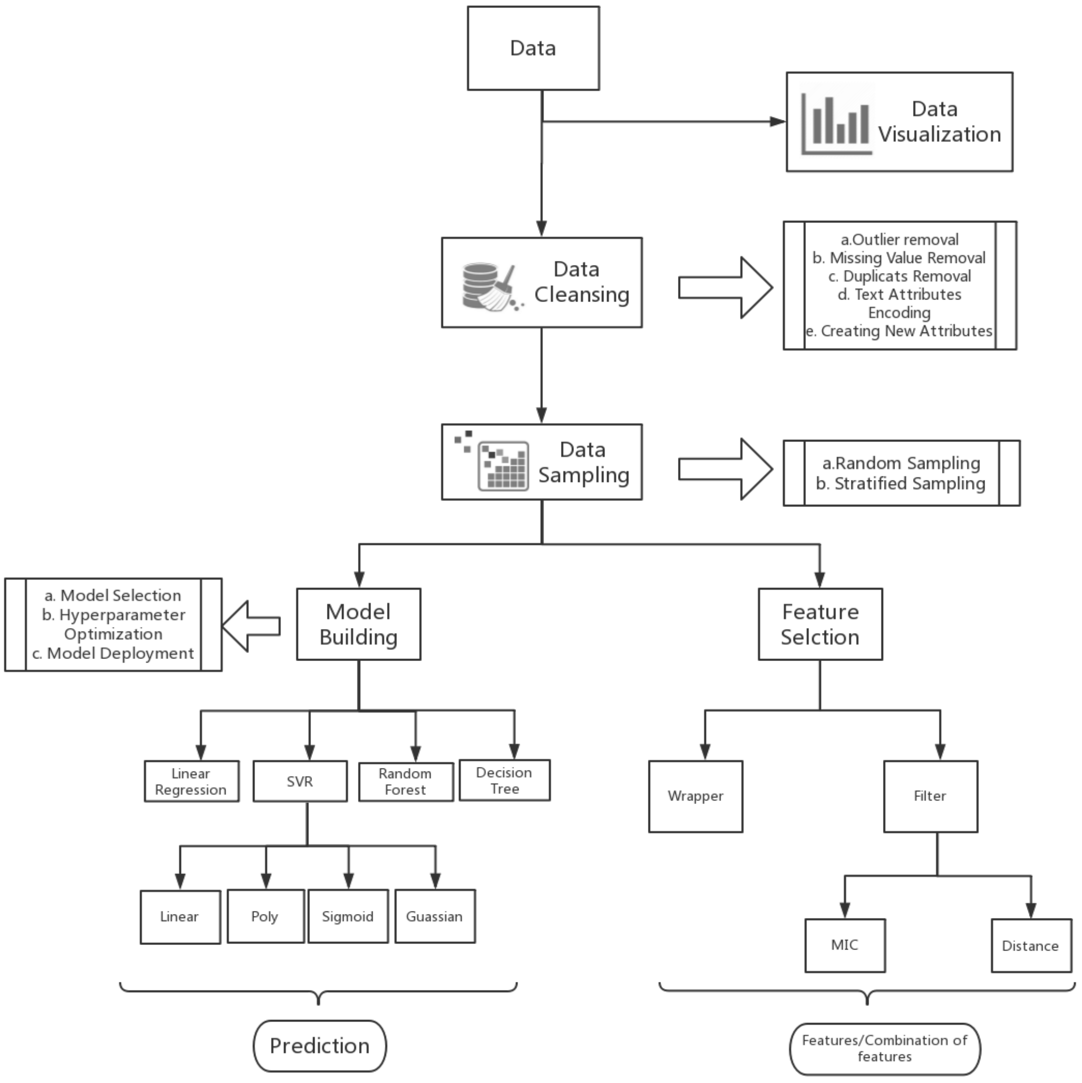


Figure 1: Block Diagram of Intended Methodology

3.2 Data Preparation

3.2.1 Data Extraction

As discussed in detail in the literature review, that the data needs to be prepared for the machine learning algorithms. Raw Data for this project is dispersed into five csv files, two of which contain the features of every household and these two csv files are known as `properties_2016.csv` and `properties_2017.csv` respectively . Each real estate housing was labeled with a parcel id. The parcel id serves as a unique identifier for every housing estate.

The `properties_2016.csv` and `properties_2017.csv` file contains the number of unique features associated with each parcel id. It contains the count of the number of different features present in each real estate housing for some months in 2016 and 2017 respectively. There are fifty seven different features for each parcel id and these features serve as descriptors for every housing estate in terms of the accessories a housing estate contains (bathroom count, bed room count etc), the geographical location of every housing estate(latitude, longitude, country code, city id etc).

The other two csv files which are known as `train_2016.csv` and `train_2017.csv` contain the transaction date, which is date when the particular parcel id was sold and the log error value for every parcel id. The log error is the difference between the actual sale price of the house and the price estimated by Zillows algorithms (Z-estimate). The log error is given by the following equation below: [10]

$$logerror = \log{(Zestimate)} - \log(Actual\ Sales\ Price) \tag{4}$$

A positive value of the log error means that the *Zestimate* is an over estimate i.e. the forecast on the sales price exceeds the true value of the housing estate. A negative value of the log error means that the *Zestimate* is an under estimate i.e. the forecast on the sales price is an is lower than the true sales prices of the housing estate. These two csv files are first converted into data frames with the help of pandas module and then all of the csv files are merged together on the basis of their parcel id. The `properties.csv` and the `train.csv` are merged together on the basis of their parcel id. The merge itself is on the basis of left join between the two files `properties.csv` and `train.csv` for year 2016 and 2017 respectively. Finally the two merged files (`properties_2016.csv` and `train_2016.csv`) and (`properties_2017.csv` and `train_2017.csv`) are merged together to create a consolidated data frame. The join here is an outer join, which takes a union of the two data frames. This consolidated data frame was then analyzed and there are sixty columns in total in this data frame and the type and count of each column is outlined in table 1 below.

Feature Type	**Feature Count**
Int	01
Float	53
Text / Object	06

Table 1: Properties csv file Feature detail

From Table 1 above it is seen that the consolidated data frame contains six text attributes, fifty three float attribute and one integer attribute. The data is now ready for visualization and analysis.

3.2.2 Data Visualization

Visualization of data is important to reveal important characteristics of the data. The latitude and longitude of the parcel ids were extracted and a cmap was plotted to see the geographical locations which had a high log error. Figure 2 below shows the variation in log error across different regions in the United States. The parcel ids are given for regions in Ventura, Los Angeles and Orange.

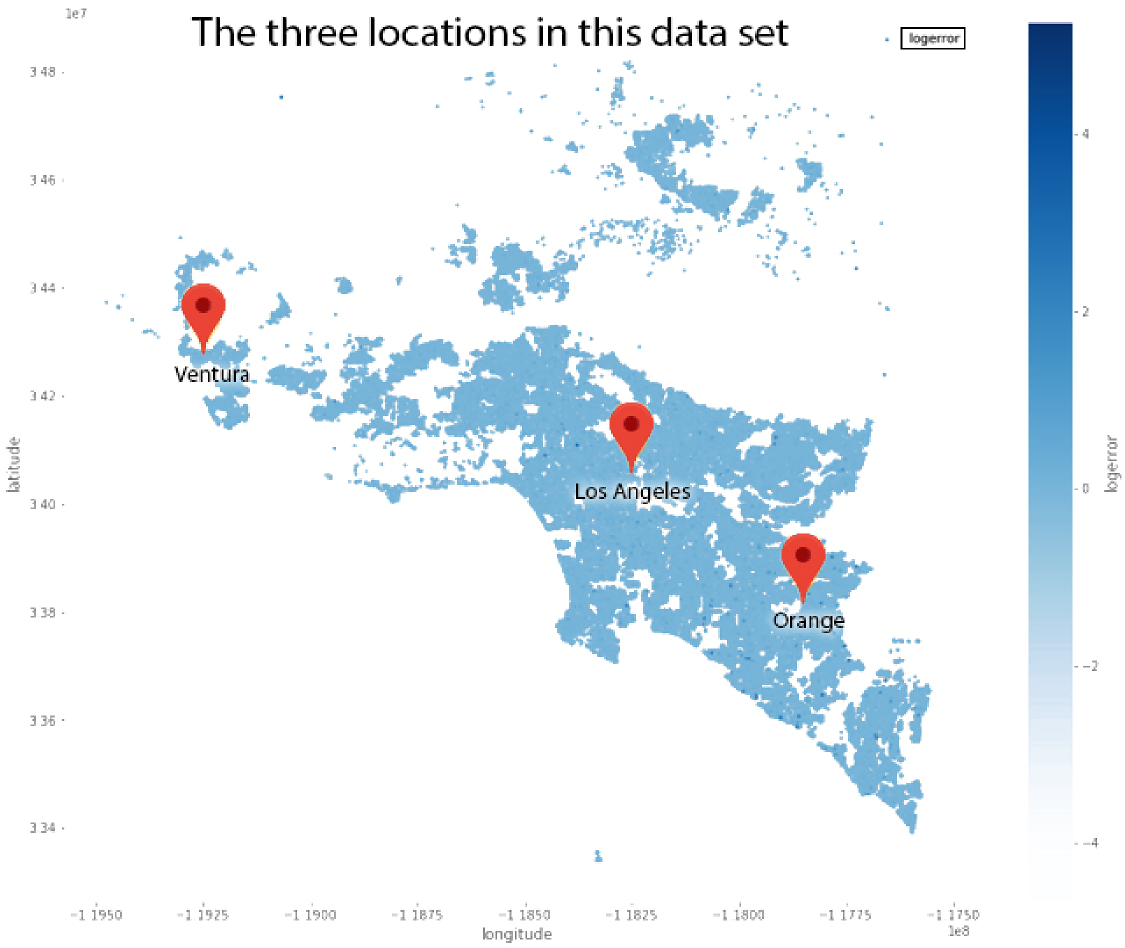


Figure 2: Variation of log error across different regions

The response variable (log error) was now tested for normality. The following piece of code was completed in R where a normality test was done on the log error. Tests such as the Anderson-Darling Test were carried out on the values of the response variable. The test assumes that the given distribution follows a specified distribution under the null hypothesis. Outliers were removed from the log error values and it was observed that the Anderson darling test gave a $p-value < 2.2e-16$ with a test statistic value of $A = 6232.2$ [11]. This indicates that the log error values do not follow a normal distribution. A quantile to quantile plot further showed the that the log error values do not follow a normal distribution and is shown in the graphs below.

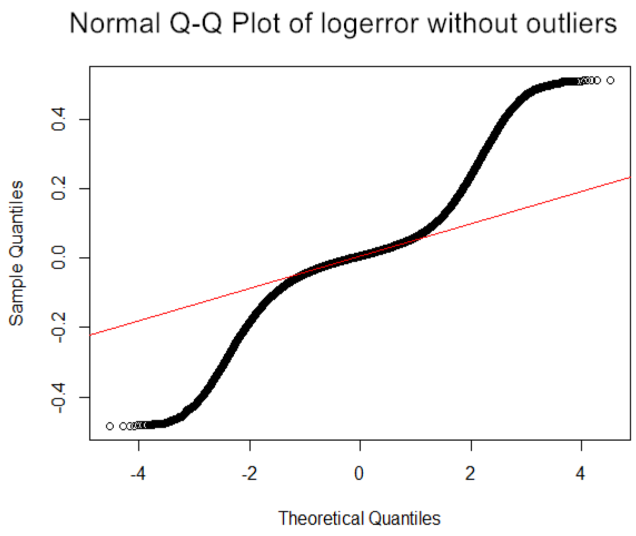


Figure 3: Variation of log error across different regions

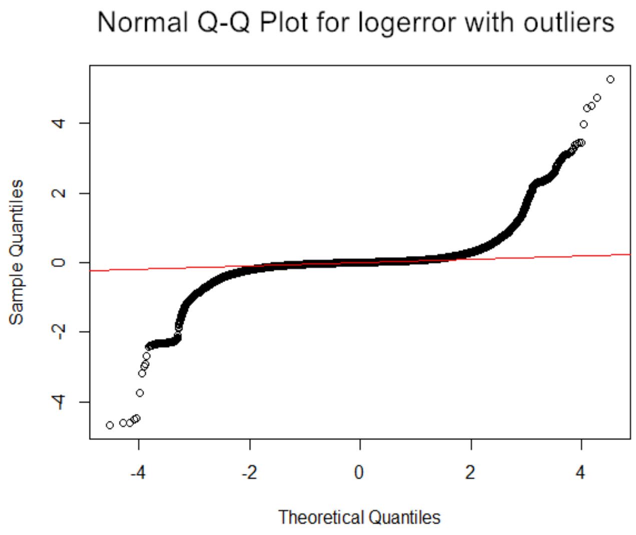


Figure 4: Variation of log error across different regions

As seen in figure 3 and figure 4 above, it can be seen that the log error values with and without the outliers are away from the regression line in the q-q plot. If the data set follows a normal distribution, then these points lie close to the regression line. In both the figures, it is evident that regardless of whether the outliers are removed from the data set, the log error does not follow a normal distribution. Furthermore from the two figures it can be concluded that the data set is non parametric in nature and that non parametric models should be used to get good predictions for the values of the log error.

After testing the response variable for normality, the Spearman correlation was then computed using the `corr()` by `scikit-learn` and correlation of log error with attribute was investigated which are summarized in the table 14 in the **Appendix section**. The `corr()` command generates a Spearman correlation matrix with respect to the attribute of interest (log error in this case).

In addition to the above it was observed that the log error had outliers that deviated from the normal log error values. A graph was plotted to observe the variation of log error as shown in Figure 5 below. It can be seen that that some values of log error deviate from the distribution of the log error. Outliers are observed at both ends as all of the points in the data set are plotted. These values are clearly a mistake in recording in the values of the log error accurately and need to be removed from the data set as they will affect the accuracy of the predictions of the model.

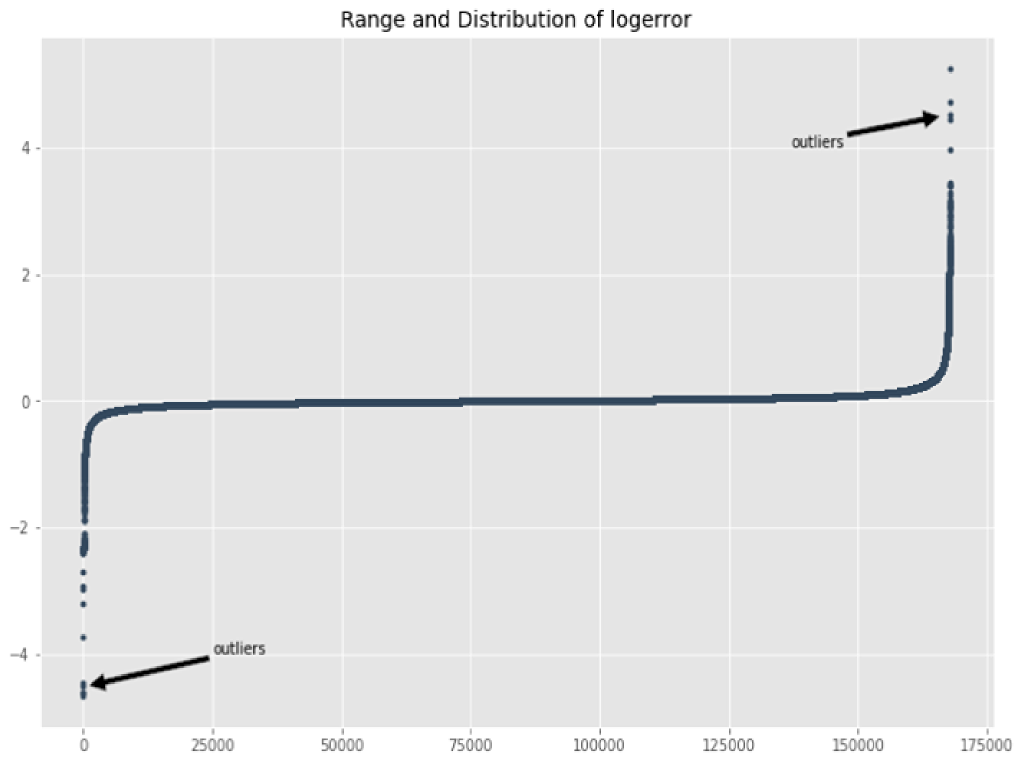


Figure 5: Variation of log error

In addition to the above, a graph was plotted for the count of every feature and the percentage of values of every feature inside the data set. It can be seen from Figure 6 below that in this data set there are a lot of missing values which need to be taken care off before the model can be trained. The original data set, explained later in the section was split into three where attributes were removed if the value exceeded a specific percentage.

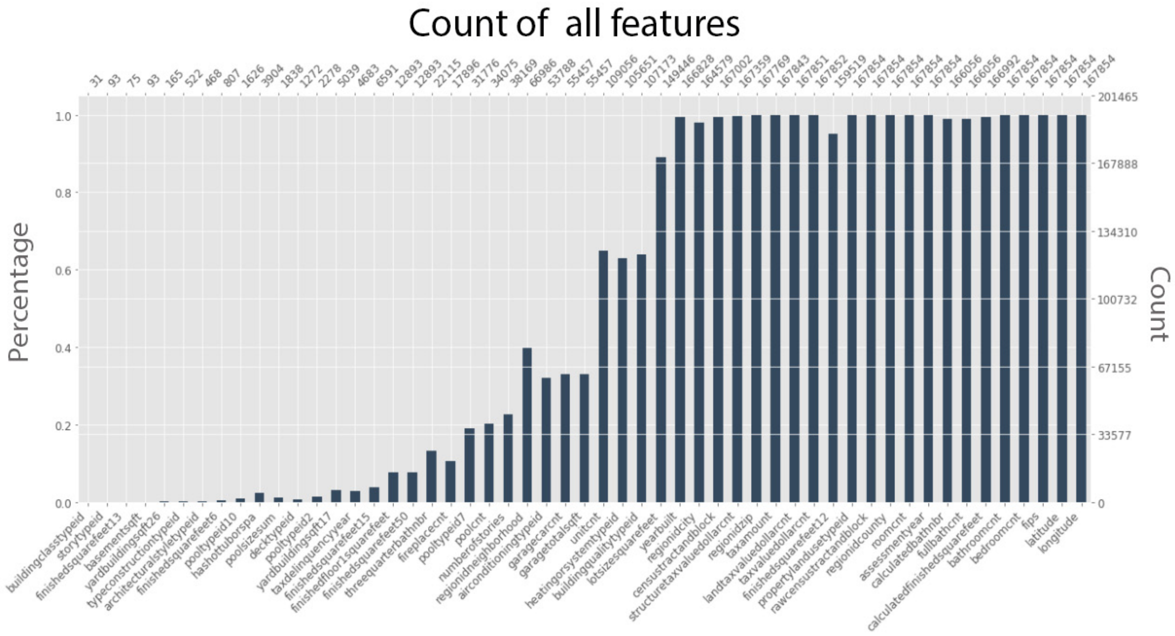


Figure 6: Percentage of Values Present in the Data Set

3.2.3 Data Cleaning:

Data Redundancy Check: After gaining an insight into the data and visualizing it, the data was cleaned. The data was first checked for redundancy. The duplicate and redundant rows were removed using the `drop_duplicates` function available by the `pandas` package in python. Any parcel id that was repeated across the same year was removed as this showed that the particular parcel id had been sold twice in a year which was clearly a mistake in data acquisition.

Outliers Removal: In addition to data redundancy check, the data was also checked for outliers. As shown in the graph above for log error, points that lie out of three standard deviations of the log error value were removed. The data was then checked for missing values.

Filtering Missing Values: The next task consisted of dealing with missing attributes inside the data set. As indicated in the literature review that the the missing values inside the data set can be dealt with by either removing the whole attribute or some rows from the data set or by setting the missing values to some other value such as the median or mode inside the data set. Using the options indicated in the literature review; three data sets were created, based on the percentage of missing values of features inside the data set.

The percentage of missing values for every feature inside the data set was computed and in the three data sets created, all missing values greater than 30,50 and

97.5 % were removed. Table 2 below shows the summary of three data sets created. These three data sets are refereed to as **30, 50 and 97.5% datasets.**

Data Set	Total Features	No of Numerical Features	No of Text Features
97.5 %	42	39	3
50 %	31	28	3
30 %	27	25	2

Table 2: Features left after features with a specific percentage of Missing Value Removed

Table 2 above shows the percentage of data set when features with missing values greater than 97.5, 50 and 30 % are removed from the data set. The text attributes of all three data sets were now separated and the text and the numerical attributes were dealt separately.

Handling Numerical Attributes: The numerical attributes of all three data sets which had missing values were now replaced by their medians.The `Scikit-Learn` class in python has a special function known as the `Imputer` for taking care of missing values [3]. The `Imputer` computes the median of each numerical attribute inside the data set and then it replaces each of these numerical attributes with their median or whatever strategy that is specified by the user. The numerical attributes for all of these three data sets were replaced with their medians.

Handling Text Attributes: The transaction date feature was removed and it was split into month and year only. The month and year were encoded as one hot vectors. Scikit learn has a class known as `OneHotEncoder()` to take care of this. The other two parameters to take care off were the `property zoning desc` and `property country land use code`. For both of these attributes (`property country land use code` and `property zoning desc`) , integer encoding was done. Scikit learn has a class known as `Label Encoder()` and this assigns integer codes to every unique label. For the missing values in the text attributes, both of these attributes were iterated over and a dictionary data structure stored the count of every label that occured inside the dataset. From the count of the each label, it calculated the relative frequency of the label. The relative frequency of each label inside the data set was given by equation 5 as follows:

$$Relative\ Frequency\ of\ Label = \frac{Frequency\ of\ Label\ inside\ the\ feature}{Total\ count\ of\ all\ labels\ inside\ the\ Feature} \tag{5}$$

From the relative frequency of each of these labels, the function `stats.rv_discrete()` was used to assign the integer codes randomly to each missing text value for both of these attributes. The rest of the labels were then replaced with their codes as determined by the `Label Encoder()` class of `Scikit-Learn`. The data is now ready for sampling.

3.3 Data Sampling

There are two techniques to sample data as discussed in the literature review. An initial analysis was carried out on the original data set to compare the error generated by the random sampling and the error generated by stratified sampling. First all three data sets, the log error values were assigned labels based on the variation of its values for sampling. Table 3 below sums up the assignment of log error labels for all three data sets.

Label No	**Log Error Range**	**No of Values**
01	[-0.507775,-0.03)	36647
02	[-0.03,0)	35636
03	[0,0.025)	37977
04	[0.025,0.06)	28152
05	[0.06,0.0508085]	26437

Table 3: 30 and 50 percent data set Stratas for log error values

The log error values were then sampled using stratified sampling and random sampling and test set generated by each of these sampling techniques was compared and the results are shown in Table 4 below.

Label No	**Overall**	**Random**	**Stratified**	**Random % error**	**Strat % error**
01	0.222	0.223	0.222	0.624	0.00764
02	0.216	0.213	0.216	-1.06	-0.00370
03	0.230	0.230	0.230	0.168	0.00737
04	0.170	0.171	0.170	0.368	-0.00761
05	0.160	0.160	0.160	-0.0617	-0.00810

Table 4: Sampling bias- Stratified Sampling vs. Random Sampling for Data sets 30 and 50 percent

From Table 4 above it can be seen that the test set generated by stratified sampling has values identical to the original data set whereas the random sampling set has a rather biased distribution. [3].

From the table above and as discussed in the literature review,it is evident that the data needs to be sampled such that there will be an equal instance of the response variable to ensure that the model to be deployed is trained effectively to work on unseen data. It is for this reason that stratified sampling is chosen over random sampling. For each of the three data sets, the data is stratified based on the log error labels assigned. Care is taken to ensure that each strata is large enough and that there are less number of stratum in total.

3.4 Predictive Modeling

Once the data was sampled for each of the sets and divided into test and training tests, different models were selected to predict the log error values. From data visualization and cleansing, it is realized that this is a supervised learning task and a regression problem, where the value of the response variable is to be determined from a given a set of features and that the models to be selected here should be non parametric in nature.

Two metrics to measure model performance are selected which are the R squared statistic and the mean square error. The Models are tested using both hold out validation and K Fold Cross Validation. Some models require hyper parameters to be optimized before they can be used and a the `Grid SearchCV()` feature has been used to select the best hyper parameters for model performance.

3.4.1 Model Selection:

Initially a linear model was selected as the this is one of the most common models to use for regression problems. Although not used for non parametric data, the sklearn class transforms normalizes the dependent variable X when calculating the predictions. So the linear regression used here is a non parametric linear regression in fact. The R2 and the mean square values for the non parametric linear model were calculated using both hold out Validation and stratified k fold sampling.

In addition to linear model, additional non parametric models were used to estimate the value of log error. These included Decision Tree and Random Forest algorithms.

Decision Trees are selected because of the following reasons: [12]

- They require little preparation of data.
- The computational cost of decision trees in data prediction is logarithmic to the number of data points used in training of the decision tree [12]. For Large data sets such as Zillows, this is a potential advantage to use.
- They require little assumptions and can even work if some of the assumptions required for the decision tree model contradict the assumptions of the original data.

A potential problem with Decision Trees algorithm is that it can over fit the data by increasing the complexity of the underlying tree [12]. A random forest combines the results of a lot of different decision trees based on samples of different sizes. It then averages all of the predictions of all of these trees. The sample size is always the same in length as the original sample [13].

In addition to the above models, another technique used for non-parametric modeling is Support Vector Regression. The support vector regression class in `scikit-learn` uses different kernels for regression. Kernel density estimates (KDEs) are a common form of non parametric modeling and modeling using different types of these kernels was done to get investigate the predictions for the log error.

3.4.2 Hyper Parameter Optimization:

Before some of the models mentioned above could be used, their parameters needed to be optimized. This process is known as **Hyper Parameter optimization**. Grid search algorithms are available to get the optimal value of these hyper parameters. For SVR, `GridSearchCV()` can be used to find optimal hyper parameters to use. Decision and Random Forest use `RandomizedSearchCV()` to optimize their parameters using a cross validated search over the parameters.

For the Random Forest model, three parameters which are the `n_estimators` and the `min_sample_leaf` and `max_features` are used for hyper parameter optimization. The `n_estimators` feature gives the number of decision trees that predict the response variable, the higher the number of decision trees, the more accurate the predictions [14]. A higher number of decision trees also implies that the algorithm will average the predictions for a large number of decision trees, which can be computationally expensive so the processing power and the speed of the search algorithm also needs to taken into consideration. The `max_features` features is the number of features that are allowed for the decision tree to choose randomly when calculating the predictions for the response variable. The `min_sample_leaf` indicates the number of leaf nodes of the decision tree. The higher the leaf nodes of a decision tree, the less prone is the model to capture noise. It is preferred that the minimum number of leaf nodes is greater than fifty [14].

For the Decision Tree model, two parameters which are the `min_samples_leaf` and the `min_samples_split` are optimized. These `min_samples_leaf` are again the number of leafs that the decision tree needs to have as a minimum. The `min_samples_split` are the number of splits that limit how much a subtree has to divide, for large data sets, it is recommended to increase this value [?].

For the SVR regression, various kernels are used. SVR uses kernel density estimation, which gives a density function from the sampled data. There are different types of kernels available in the sklearn class and all of these have been used. Due to run time complexity of SVR and time constraints, the grid search algorithm was only run for the Gaussian kernel and hyper parameter tuning for other kernels was done on a hit and trial basis. For SVR the parameters that need to be tuned are the penalty parameter C and the kernel coefficient γ [15]. The parameter γ controls the degree of fitting the training data model whereas the parameter C regulates this and prevents the model from over fitting the data. The kernel coefficient γ also controls the generalization error. Figure 7 below shows grid search for SVR for Guassian Kernel.

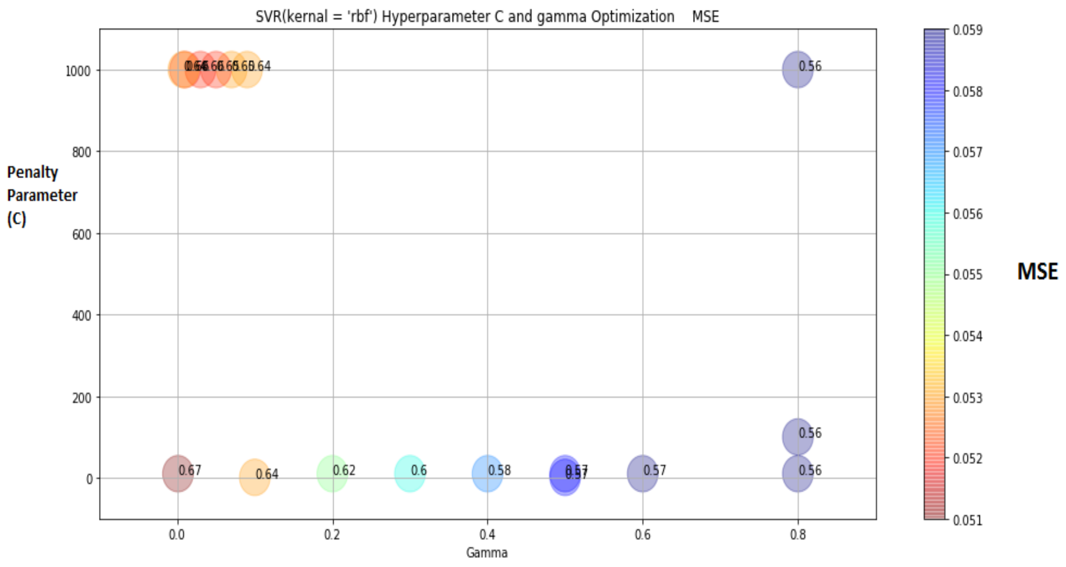


Figure 7: Hyper Parameter tuning SVR, Gaussian Kernel

For Grid Search CV, it can be seen from the Figure 7 above that the `GridSearchCV()` function computes the Mean square error for different values of the penalty parameter C and γ. The criterion to optimize the parameters here is to minimize the mean square error. For the Gaussian Kernel, it can be seen that the mean square error value is small when γ is small and C is large enough.

Table 5,6,7 below summarize the best hyper parameters for different models and the corresponding optimized mean square error values that were obtained as a result of tuning these parameters through the `RandomizedSearchCV` and `GridSearchCV()` classes of `scikit-learn` for all three data sets (30%, 50% and 97.5%).

Model	**Hyper Parameters**	**MSE**	**Standard Dev. MSE**
Linear	c:100 γ: Auto	0.00399	0.0632
Random Forest	`max_features`: 14 `min_samples_leaf`: 51 `n_estimators`: 55	0.00258	0.0508
Decision Tree	`min_features_split`: 7 `min_samples_leaf`: 180	0.0514	0.226
SVR- Guassian	c: 10 γ: 0.001	0.00399	0.0632
SVR- Poly	c: 100 γ: 0.1	0.00397	0.0630
SVR- Sigmoid	c:100 γ: 0.00001	0.00397	0.0630
SVR- Linear	c:10 γ: Auto	0.00360	0.0600

Table 5: Optimal Hyper parameters for non parametric models for 30 percent data set

Model	**Hyper Parameters**	**MSE**	**Standard Dev. MSE**
Linear	c: 1 γ: Auto	0.00400	0.0632
Random Forest	`max_features`: 14 `min_samples_leaf`: 51 `n_estimators`: 55	0.00259	0.0509
Decision Tree	`min_features_split`: 4 `min_samples_leaf`: 171	0.0514	0.226
SVR- Guassian	c: 10 γ: 0.001	0.00399	0.0632
SVR- Poly	c: 100 γ: 0.01	00.00395	0.0628
SVR- Sigmoid	c: 100 γ: 0.00001	0.00397	0.0630
SVR- Linear	c:10 γ: Auto	0.00348	0.0590

Table 6: Optimal Hyper parameters for non parametric models for 50 percent data set

Model	**Hyper Parameters**	**MSE**	**Standard Dev. MSE**
Linear	c: 100 γ: Auto	0.00388	0.0623
Random Forest	`max_features`: 14 `min_samples_leaf`: 51 `n_estimators: 93`	0.00267	0.0517
Decision Tree	`min_features_split`: 7 `min_samples_leaf`: 180	0.00130	0.0361
SVR- Guassian	c: 10 γ: 0.001	0.00265	0.0515
SVR- Poly	c: 10 γ: 0.01	0.00205	0.0453
SVR- Sigmoid	c: 100 γ: 0.00001	0.00205	0.0453
SVR- Linear	c:10 γ: Auto	0.00278	0.0527

Table 7: Optimal Hyper parameters for non parametric models for 97.5 percent data set

3.4.3 Model Evaluation:

After tuning the hyper parameters and selecting appropriate models, Hold Out Validation and Stratified K Fold Cross Validation were performed and two metrics to measure the models performance were used. The R-squared value and the mean Square Error value were used as the main identifiers to measure the models performance. The ideal target was to achieve a model with a high R-squared value as this explains the amount of variability in the response variable that is explained by the model and the mean square error. The mean square error served as a generalization error to measure the effectiveness of model on unseen data. The end goal was to pick a model with a low mean square error and a high R-squared value.

3.5 Feature Selection

3.5.1 Feature Scaling

Feature scaling could be defined as the method used to standardize the range of variables. When the variables have very different ranges, algorithms cannot not predict well so scaling aligns the range of variables. There are two main methods to scale features which are min-max scaling and standardization. The min-max scaling method is to scale the data to [0,1] or [-1, 1]. Scaling to what extent depends on the nature of the data. The formula for this method is as follows:

$$x^{`} = \frac{x\breve{\ }min(x)}{max(x)\breve{\ }min(x)} \tag{6}$$

In this formula, x is the initial value, and x' is the scaled value. According to the formula, the values are shifted and returned to make their values from 0 to 1 by subtracting the min value and dividing by the max minus the min. In this project, a transformer called `MinMaxScaler()` is provided by `Scikit-Learn` which achieve the same affect.

Standardization is different from the min-max scaling. It transforms variables to have zero mean and unit variance. The formula for this method is as follows:

$$x^{`} = \frac{x - \bar{x}}{\sigma} \tag{7}$$

According to the formula, first it subtracts the mean value (so standardized values always have a zero mean), and then it divides by the variance so that the resulting distribution has unit variance. Standardization is much less affected by outliers than Min-max scaling [3]. In this project, a transformer called `StandardScaler()` provided by `Scikit-Learn` which achieves the same affect.

According to the form of feature selection, the feature selection method can be divided into three types

3.5.2 Methodology

3.5.3 Filter

Filter feature selection methods score each feature according to its divergence or correlation and rank the features by the scores. The methods often set threshold value or number of selected features then select features by the threshold value and the number. Filters evaluate the features as a pre-processing step, which are independent of the model [8].

3.5.3.1 Mutual information Content (MIC): According to the marginal probability theory, it can be concluded as follows that:

$$P(X,Y) = P(X)P(Y) \tag{8}$$

This is these two random variables are independence, furthermore, if X and Y are independent, the observed X will make no effect to the distribution of Y, the process of proof is:

$$\begin{aligned} P(Y|X) &= P(X,Y)P(X) = P(X)P(Y)P(X) = P(Y)P(Y|X) \\ &= P(X,Y)P(X) = P(X)P(Y)P(X) = P(Y) \end{aligned}$$

From this, the independence shows that if the distribution of Y changes with a observed X, but the independence just shows if there exists a relationship between X and Y, it can't shows the extent of the relationship, so the mutual information can be defined as:

$$I(X;Y) = \int X \int Y P(X,Y) log P(X,Y)P(X)P(Y) \tag{9}$$

After some transformation:

$$\begin{aligned} I(X;Y) &= \int X \int Y P(X,Y) log(X,Y)P(X)P(Y) \\ &= \int X \int Y P(X,Y) log(X,Y)P(X) - \int X \int Y P(X,Y) log(Y) \\ &= \int X \int Y P(X)P(Y|X) log(Y|X) - \int log(Y) \int X P(X,Y) \\ &= \int X P(X) \int Y P(Y|X) log P(Y|X) - \int Y log P(Y)P(Y) \\ &= -\int X P(X) H(Y|X=x) + H(Y) \\ &= H(Y) - H(Y|X) \end{aligned}$$

In this function, $H(Y)$ means the entropy of Y:

$$H(Y) = -\int Y P(Y) \log P(Y) \tag{10}$$

Which means the uncertainty of Y, the dispersion of distribution is higher, the entropy of Y is higher, and $H(Y|X)$ shows that in the situation which X is observed, the uncertainty of Y, the function shows above means the amount changed because of the import of X, if X and Y are more related, the value of I $(X;Y)$ is higher, the maximum of $I(X;Y)$ is $H(Y)$, which means X and Y are totally related. When X and Y are independent, $I(X;Y) = 0$, import X doesn't make any effect to Y. In this project , the mutual information is used to calculate the relevancy of logerror and other features. The maximal information coefficient in different grid cells selects the best features. In python, the MIC value is calculated through `minepy` package, and in different data sets, the results showed in different ways.

3.5.3.2 Distance Correlation This distance correlation covers the deficit of Pearson correlation in some degree. Cause in non-linear relationship, the Pearson correlation might be zero, but these two variables might be non-linear relevant, but if the distance correlation value is zero, these two variables must be independent. A convincing demonstration will need to account for variability in the resources that are used to attack a problem, the size of the problem, variance in stochastic algorithms, definition of success and other difficult issues.[16] Here shows python code about distance correlation, cause this code is clear `def sim_distance(x, y):`

```
dist = sum([x[i] - y[i]**2 for i in range(len(x))])**0.5
return 1/(1+dist)
```

3.5.4 Wrapper

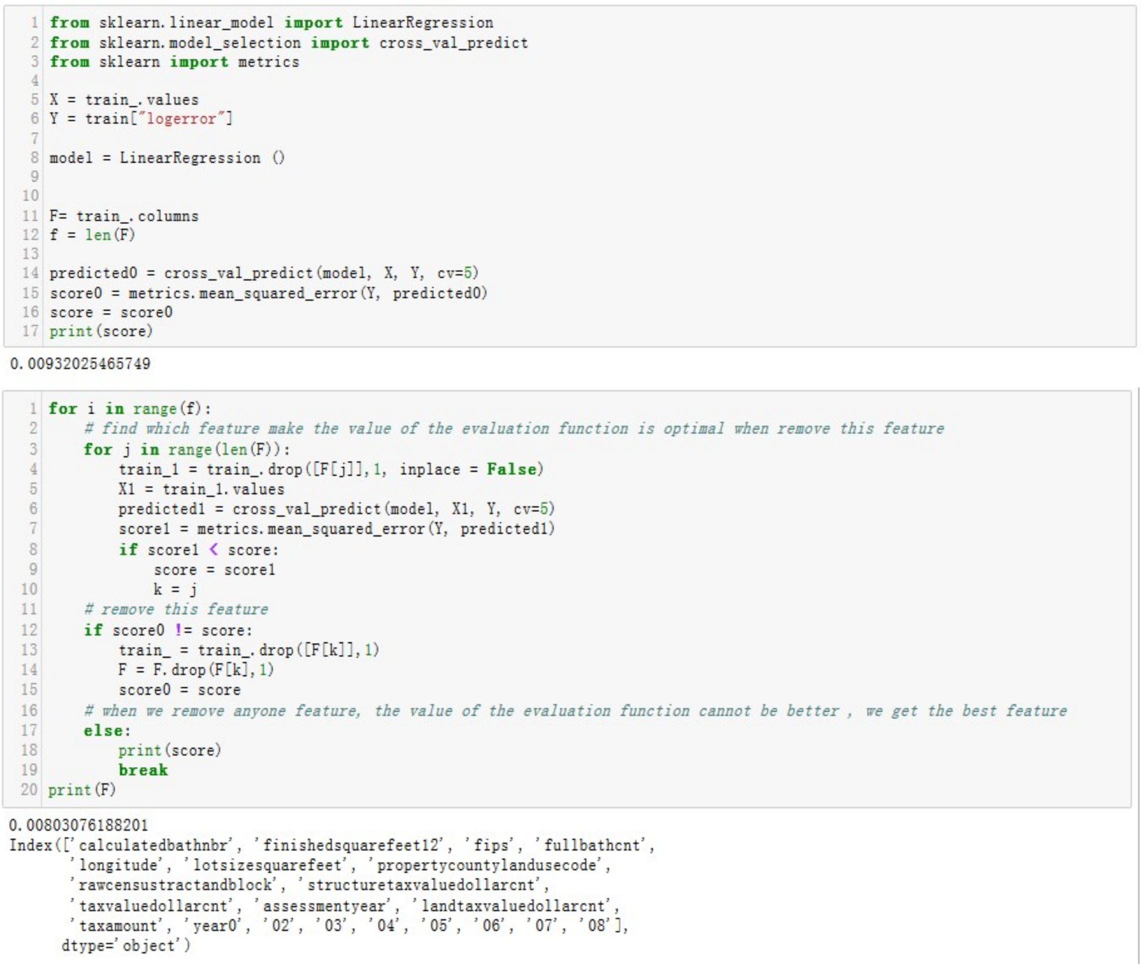
Wrapper feature selection methods considers the selection of a subset as a search optimization problem which generates different combinations, evaluates combinations, and compares them with other combinations. By this way, wrapper feature selections finds the best subset of features which makes the prediction optimal. This approach is closely related to predictor model and evaluating systems. Wrapper method is to build lots of subset of features and then evaluates which feature subset make the prediction best. By this way, the methods drop the redundant features and find the best features for model. Generally, the wrapper methods could be more effective than filter method because this method choose the features according to the predictor model.

[8] argues that there are three things should be defined in this method: (i) how to search all possible subset of the features; (ii) which model to use for prediction; (iii) how to evaluate the feature with the given model. In this project, all of these are defined as follows: (i): this project searches the subset of features by the backward selection method. (ii): this project uses Linear Regression model, Decision Tree model and RandomForest model to find the best features for each three models. (iii): this project uses Cross Validation to evaluate the features, according to calculate the MSE (mean-square error).

Suppose dataset have n features, so there are $2^n - 1$ possible feature subsets. If n is very large, it is hard to search all possible $2^n - 1$ feature subsets because it could cost too much time. Therefore, some heuristic algorithms such as forward selection or backward selection could be used to realize the research of the subset of features. This project uses backward selection method to search features for three data sets (NewFinal30, NewFinal50, NewFinal97).

The example wrapper method feature selection code for NewFinal30 data set using Linear Regression model is as follows:

As the code shown in Figure 8 below, the first step is to build the subset of all features, which named F. Use Linear Regression to fit the variables and target values. Get the prediction of target value. Then evaluate the subset of all features by calculating the MSE (mean-square error) according to the prediction target value and real target value. The second step is to drop one of feature from this subset. The new subset is named F_1. Then calculate the MSE. After that, drop another feature from F and also calculate the MSE. Loop this step for every feature in F. If there are n features in F, there are n MSE score. The feature which corresponds to the smallest MSE score is the feature need to drop. Just drop it. The third step is to loot all above steps. There is a subset of features whose score is smallest.



```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_predict
from sklearn import metrics

X = train_.values
Y = train["logerror"]

model = LinearRegression ()


F= train_.columns
f = len(F)

predicted0 = cross_val_predict(model, X, Y, cv=5)
score0 = metrics.mean_squared_error(Y, predicted0)
score = score0
print(score)
```

```
0.00932025465749
```

```
for i in range(f):
    # find which feature make the value of the evaluation function is optimal when remove this feature
    for j in range(len(F)):
        train_1 = train_.drop([F[j]],1, inplace = False)
        X1 = train_1.values
        predicted1 = cross_val_predict(model, X1, Y, cv=5)
        score1 = metrics.mean_squared_error(Y, predicted1)
        if score1 < score:
            score = score1
            k = j
    # remove this feature
    if score0 != score:
        train_ = train_.drop([F[k]],1)
        F = F.drop(F[k],1)
        score0 = score
    # when we remove anyone feature, the value of the evaluation function cannot be better , we get the best feature
    else:
        print(score)
        break
print(F)
```

```
0.00803076188201
Index(['calculatedbathnbr', 'finishedsquarefeet12', 'fips', 'fullbathcnt',
       'longitude', 'lotsizesquarefeet', 'propertycountylandusecode',
       'rawcensustractandblock', 'structuretaxvaluedollarcnt',
       'taxvaluedollarcnt', 'assessmentyear', 'landtaxvaluedollarcnt',
       'taxamount', 'year0', '02', '03', '04', '05', '06', '07', '08'],
      dtype='object')
```

Figure 8: Wrapper method code

When drop any feature, the MSE could not decline. This subset is the best subset of features.

3.6 Reverse Engineering:

After evaluating the model performance, the predictions of the model were analyzed in general to identify the test cases where the model fails to generalize. More of this analysis is shared in the next section.

4 Results and Conclusions:

4.1 Predictive Modeling Results:

As stated in the previous sections, different models were selected to analyze the performance of models on different datasets, Table 8 summarizes the performance of the models for Hold out Validation without hyper parameter optimization for different datasets.

4.1.1 Hold Out Validation Results:

Without Hyper Parameter Optimization: The Table below shows the variation of performance of different models for each of the three data sets with greater than 30 %. 50 % and 97.5 % missing values removed. This was tested with three models, which are the non-parametric linear regression model, decision tree and the random forest model. From the Table 8 below, the following observations are made:

- The R-squared value and the mean square error values for 30 and 50 % data sets are quite similar and there is little variation in these two values for both of these data sets. This is also an indicator that the feature combination in this data set does not affect the model performance significantly.
- For 97.5 % Data set, the highest R-squared values and the lowest mean square error values are observed across all three models. This could be attributed to the fact that 97.5 % data set contains the most information to train the models however, one concern is the reliability of the results offered by the 97.5 % data set as seen in the data visualization section of the report, a potential problem of this data set is that a lot of features have missing values. Since the total number of rows in the original data set are 167887, values greater than 97.5% would constitute with a count of about 4198 or less. In a data set of 168,000 values, replacing about 163,802 values or less for such features is not an accurate way to train models and this is for two reasons. Firstly many housing estates may not even have those features and secondly there is very little evidence for the median to be correct as there is a chance that new instances may arise, the more the data will be obtained for such housing estates. It is also for this reason that this data set is excluded for most of the analysis, and analysis is mostly done for 30 and 50 % data sets.
- Performance for models improves as more powerful models are selected. From Table 8 it cab be seen that random forest model performs better than linear and Decision Tree Models.

- For all the 3 data sets, it is seen that decision tree has lower values of R2 (0.352 and 0.674) and higher value of mean square error compared to the other models and the reason for this is that the hyper parameters have not been optimized for each of the models.

HOV-Without Hyper Parameter Optimization									
Data Set	**Linear Model**			**Random Forest**			**Decision Tree**		
	R2	**MSE**	**Std**	**R2**	**MSE**	**Std**	**R2**	**MSE**	**Std**
30%	0.614	0.00311	0.0557	0.628	0.00300	0.0548	0.352	0.00523	0.0723
50%	0.614	0.0311	0.0557	0.629	0.00299	0.0547	0.357	0.00519	0.0720
97.5%	0.732	0.00213	0.0459	0.763	0.00225	0.0476	0.674	0.00260	0.0510

Table 8: HOV-Without Hyper Parameter Optimization

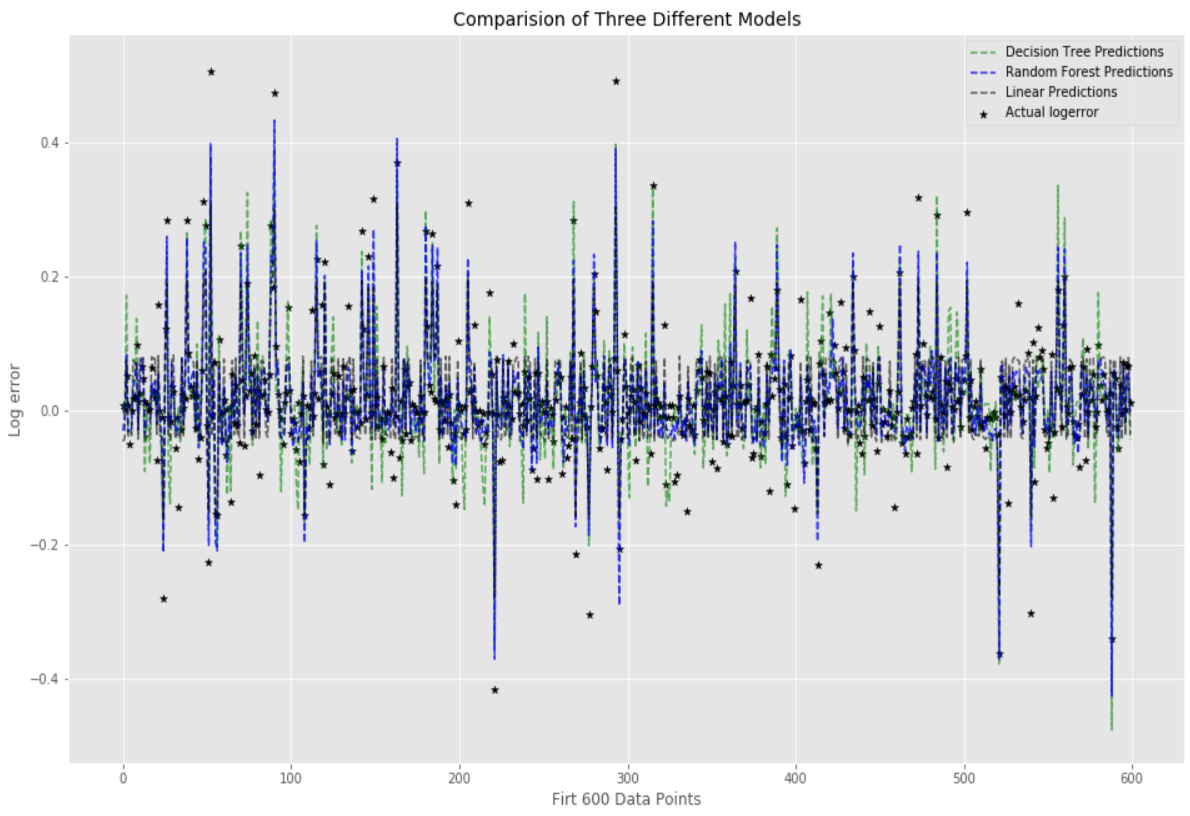


Figure 9: Hold Out Validation for different models

Furthermore, the predictions were also plotted for the first six hundred points as shown in Figure 9 above for values in the test data set to get a visual aspect of the goodness of fit of the three models. From the graph above it can be seen that the Random Forest model generalizes the most even without hyper parameter tuning compared to the other two models. This is because the random forest models averages the predictions from many different types of decision tree classifiers. The analysis is repeated with hyper parameters tuned for all models and another non parametric modeling technique known as the **support vector regression** is explored.

With Hyper Parameter Optimization: The analysis was carried out using hyper parameter optimization for all the models. In addition to this, another modeling technique known as the SVR is also explored. The analysis shows that with hyper parameter optimization, the performance of decision trees improves significantly and the performance of all other models also improves. Since there are no hyper parameters to tune for the linear model, the performance of the linear models remains the same. The analysis is also performed with hyper parameter optimization for SVR models and it the following inferences are drawn both tables:

1. From Table 9 below, for data set with greater than 30 % and 50 % values removed, for all models, including the SVR models with different kernels and for the other three models, there is little difference between the R-squared values and MSE error values for both of the tables. This is also an indicator that the combination of features used in these households, have little impact of the model performance. The 50 % data set has two more features which are the `propertyzoningdesc` and `unitcnt`. The spearman correlation of `unitcnt` feature with logerror is -0.0074, and this is also the reason why the the 30% data set has a slightly higher value of R-squared statistic and a lower Mean square error as compared to the 50 % data set.

2. In Table 10 below, For 30 % data set the sigmoid kernel gives the highest R squared value (0.492) and the lowest MSE (0.063). For the 50 % data set the poly kernel gives the highest R squared value (0.494) and lowest MSE (0.0629).

3. In Table 10 below, For the 97.5 % data set, the R2 and the MSE values improve significantly again. For the same kernel the value the R squared value increases to approximately 1.5 times and the MSE decreases by 1.3 times approximately. The best kernel for SVR for this data set is sigmoid. For 97.5 % data set the problem again lies with the reliability of the results for reasons mentioned above.

4. For both Table 9 and Table 10,Overall with hyper parameter optimization, it is observed that for the 30 % and 50 % data sets, the Random Forest Model offers the best results (R2 value of 0.68). For the 97.5 % data set it is observed in Table 9, that the Decision Tree model offers the best results (R2 value of 0.835) which is interesting to note as the random forest model is a special case of the decision tree model.

HOV-Best Hyper Parameter									
Data Set	**Linear Model**			**Random Forest**			**Decision Tree**		
	R2	**MSE**	**Std**	**R2**	**MSE**	**Std**	**R2**	**MSE**	**Std**
30%	0.614	0.00311	0.0557	0.680	0.00258	0.0508	0.672	0.00265	0.0515
50%	0.614	0.0311	0.0557	0.678	0.00260	0.0510	0.672	0.00265	0.0515
97.5%	0.732	0.00213	0.0459	0.793	0.00165	0.0406	0.835	0.00131	0.0363

Table 9: HOV-Best Hyper Parameter Results

In addition to the above the different kernels of SVR were also plotted for the first six hundred data points as shown in Figure 10 below for the test data test for the 97.5 % data set to get some visual analytics for the goodness of fit of different

Data Set	Linear			Sigmoid		
	R2	**MSE**	**Std**	**R2**	**MSE**	**Std**
30%	0.489	0.632	0.251	0.492	0.0630	0.251
50%	0.488	0.0633	0.251	0.492	0.0630	0.251
97.5%	0.658	0.0527	0.230	0.747	0.0454	0.213
	Poly			**Guassian**		
	R2	MSE	Std	R2	MSE	Std
30%	0.488	0.633	0.251	0.489	0.632	0.251
50%	0.494	0.0629	0.251	0.489	0.0632	0.251
97.5%	0.737	0.0463	0.215	0.673	0.0516	0.227

Table 10: Parameter Optimization for SVR Kernels

SVR kernels. As seen from the Figure 10 below, the sigmoid and poly kernels fit the data points well. For this data set the sigmoid kernel gives the best goodness of fit which is in accordance with the results of Table 10.

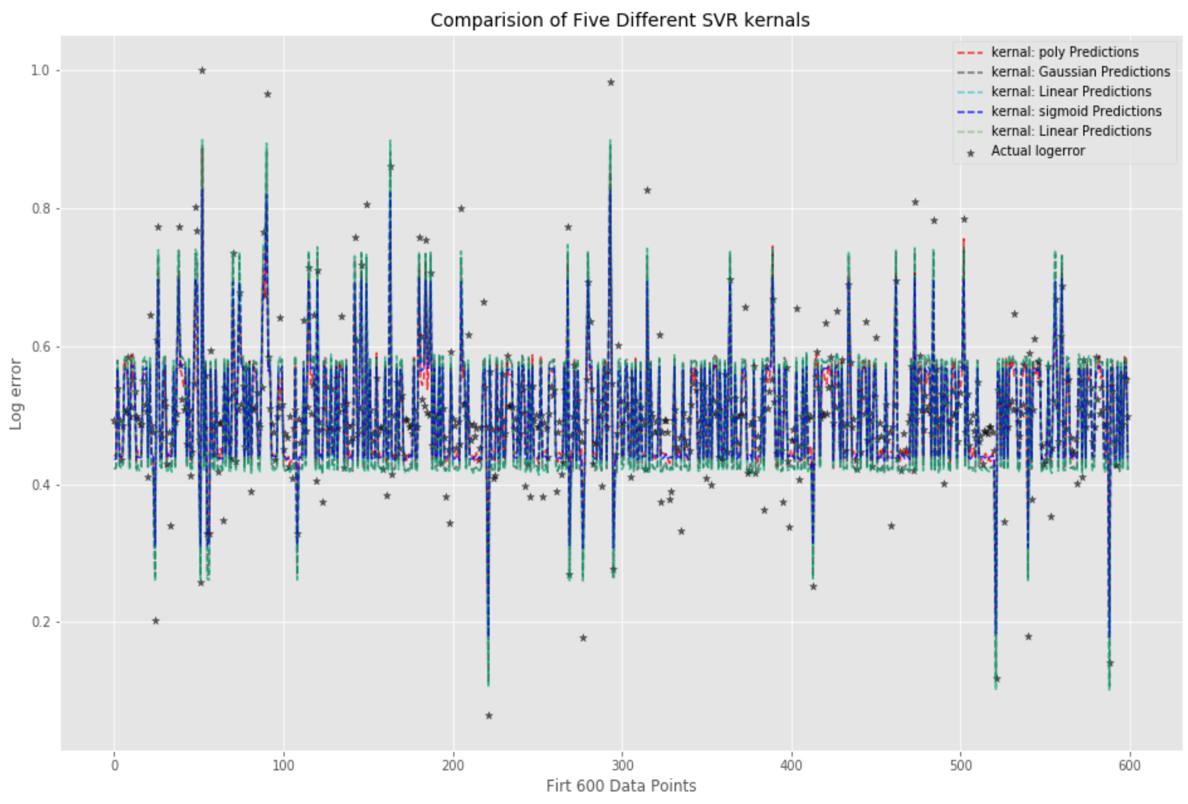


Figure 10: Hold Out Validation SVR for some data points

4.1.2 Five Fold Cross Validation:

The modeling results were repeated using K Fold Cross Validation. A 5 fold cross validation was chosen and the generalization errors (Mean square errors), R-squared values of the five folds were averaged. Table 11 below shows the average results of the five folds for cross validation. Due to time constraints and run time complexity of SVM, cross validation was not carried out. From the table, the following observations were made:

- The Random Forest model works best for all 30 and 50 % data sets giving the best values of the R squared statistic and a low generalization error (MSE) for the two models.
- The results for the 30 % and the 50 % data sets are almost identical.
- The 97.5 % data set has the highest R-squared statistic and the lowest mean square value for all models. The decision tree model works the best for the 97.5 % data set and gives a very high R-squared value of 0.836. This implies that 83.6 % of the variability in the response variable can be explained by the model however, again for this data set, there are concerns over the reliability of this value as there is not enough evidence for some features inside the data set.

Stratified 5-Fold Cross Validation Results									
Data Set	**Linear Model**			**Random Forest**			**Decision Tree**		
	R2	**MSE**	**Std**	**R2**	**MSE**	**Std**	**R2**	**MSE**	**Std**
30%	0.614	0.00311	0.0557	0.679	0.00259	0.0509	0.675	0.00262	0.0512
50%	0.614	0.00311	0.0557	0.679	0.00259	0.0509	0.675	0.00262	0.0512
97.5%	0.732	0.00213	0.0459	0.796	0.00162	0.0403	0.836	0.00130	0.0361

Table 11: 5 Fold Cross Validation Using hyper parameter tuning

4.1.3 Model Predictions for Individual Months:

Predictions Oct-Dec: The predictions for individual months in October to December 2016 for three models (non parametric linear regression model, Random Forest model, Decision Tree model) were analyzed for the 30% and the 50% data sets. Figure 11 and Figure 12 below show the Root mean square error of predictions and the R-squared values of these predictions. For the three months, October to December 2016, it is observed that the model performance is approximately similar in terms of the R-squared score and the generalization error and following conclusions are drawn:

- Random Forest gives best predictions for October and December 2016 for both data sets.
- Decision Tree Model gives the best predictions for November 2016 for both data sets.

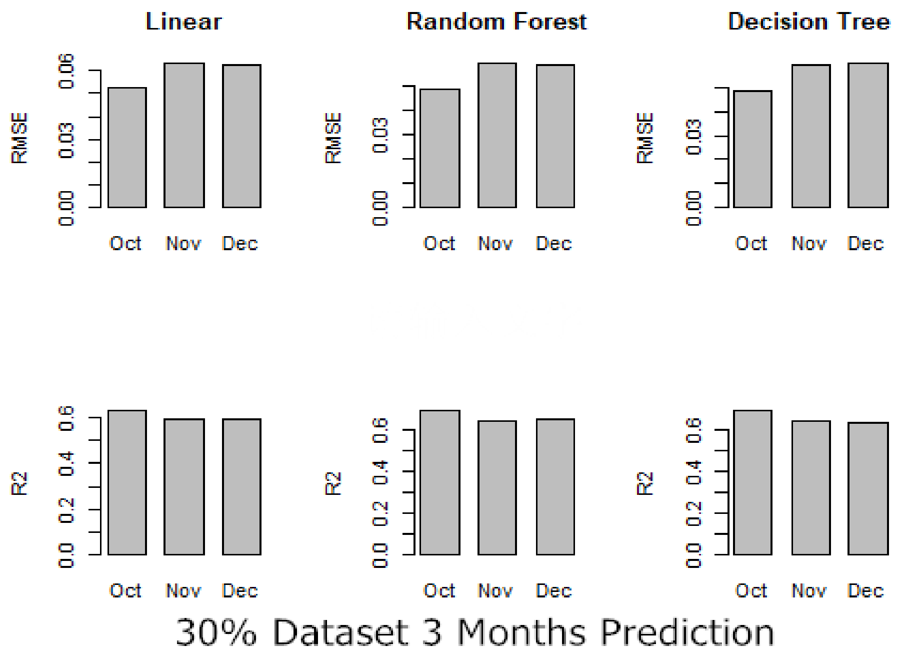


Figure 11: Stratified K Fold Predictions for 30% dataset for Different Models from October to December 2016

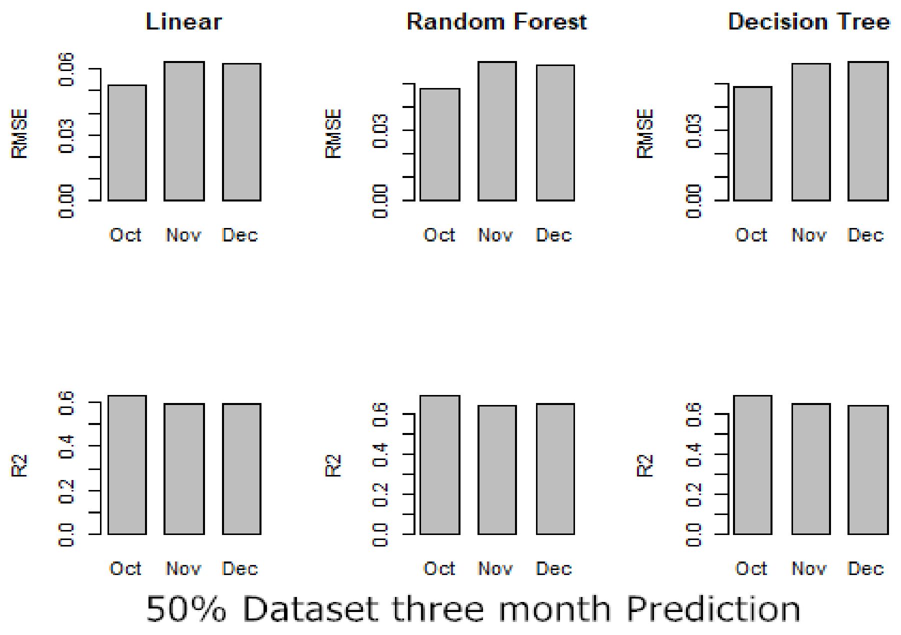


Figure 12: Stratified K Fold Predictions for 50% dataset for Different Models from October to December 2016

The models can be used to predict the log error for 2017. Since the months for 2016 were already encoded as a one hot vector with the years as 0 or 1, so for given a parcel id with a given set of features of 2017, the predictions of log error for a month can be calculated, provided that data for the parcel id is provided. In addition to the above, the model coefficients themselves can be obtained using the `coef` command that gives the models coefficients to calculate the predictions for the response variable [17].

4.2 Feature Selection Results

In addition to predictive modeling, a part of this project is to identify features or combinations of features that give the best estimate of the response variable. The features identified by the wrapper and filter methods are shown in the appendix section as figures 16 to 18 respectively in the appendix section. The wrapper methods are model dependent, and the best features for the three models (Linear,Decision Tree, Random Forest) were identified. Among the filter methods, distance method and Mutual Information Content (MIC) method were used identify relevant features for the response variable.These features were then used on models to evaluate the R-squared score and MSE error value. For the filter methods, common features using the Pearson correlation method were also identified but analysis on log error values earlier suggested that because the response variable is not normally distributed, so the shortlisted features from this method and their results are not shared in this report.

Figure 13,14 and 15 below show the results of shortlisting different features for the models. The graphs show the improvement in the R-squared statistic by using different models. From both the graphs the following conclusions can be drawn:

- The R-squared statistic improves for all models when feature feature selection is applied except for the 97.5 % data set for the random forest model. As shown in Figure 15, the R-squared value instead decreases from 0.82 (all features) both the wrapper (0.819) and filter methods. For Filter methods the value is 0.813 for MIC and 0.819 for Wrapper methods.

- It is also observed that the feature selection method is most effective on the decision tree model. For the decision tree model, the 30% and 50% datasets have a 30.89 % and 30.22 % increase in its R squared value through the wrapper methods (0.356 to 0.466 for 30 % dataset and 0.354 to 0.461 for 50 % dataset) as shown in Figure 13. For the 97.5 % data set,however, the distance filter method improves the R-squared value about 0.775. Although a 15.32% increase (0.672 to 0.775), the R squared value itself indicates that 77.5 % of the variability in the response variable can be explained by the model. However the reliability of the 97.5 % data set is a problem due to reasons mentioned above.

- For the linear non parametric regression model as shown in Figure 14, it is observed that the increase in the R-squared value is the smallest. For the 30% dataset, it is observed that the increase in the R squared value is the smallest. The increase in this statistic for the Random forest model in Figure 15 is also small.

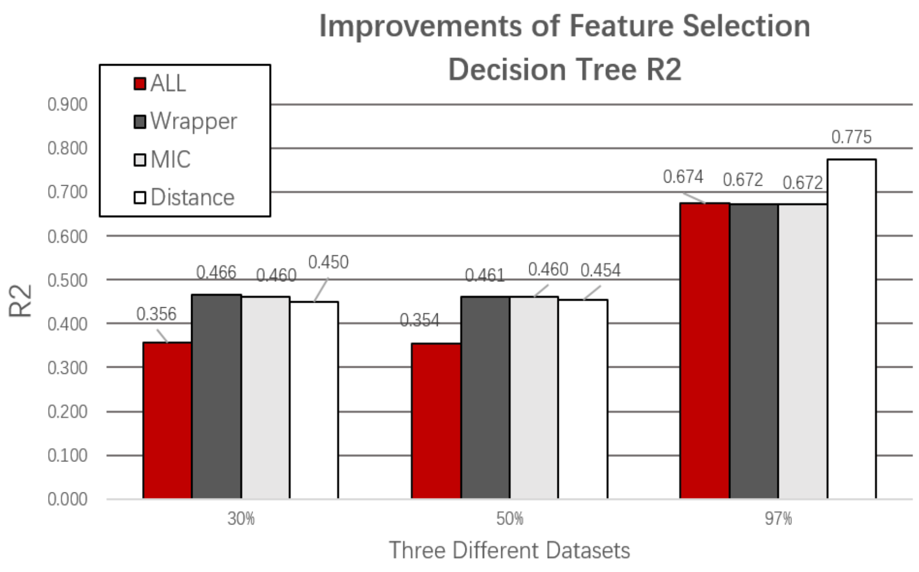


Figure 13: Feature Selection- Decision Tree

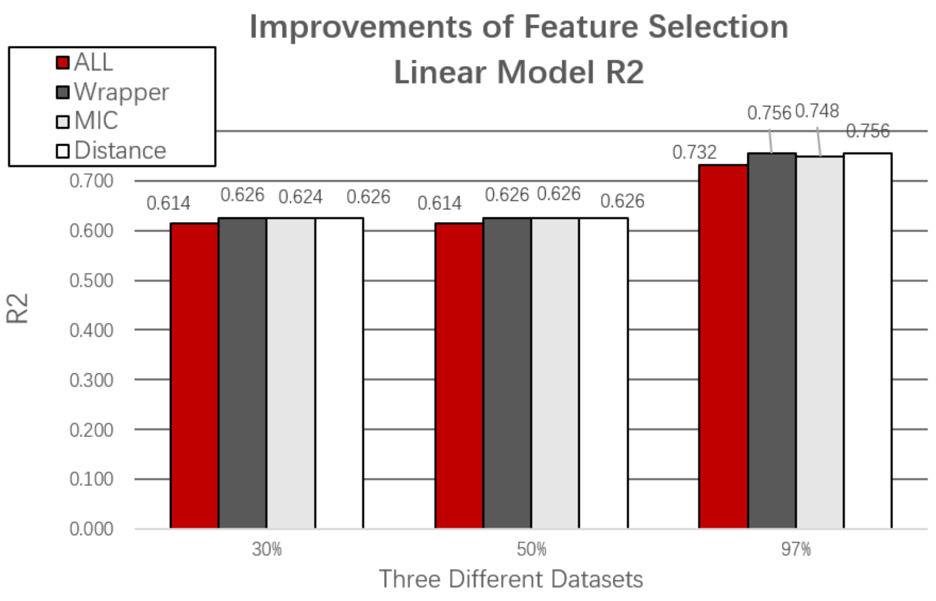


Figure 14: Feature Selection- Linear Models

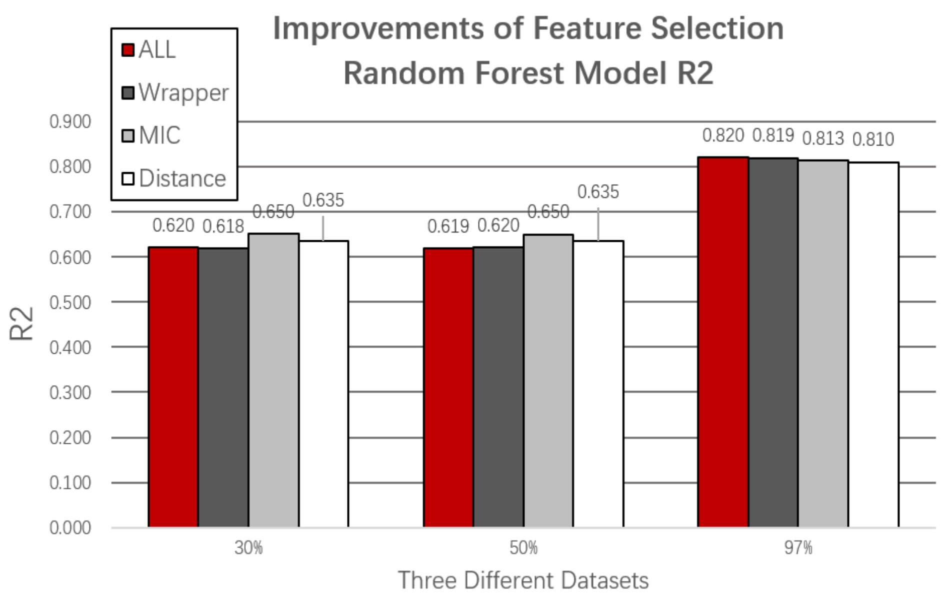


Figure 15: Feature Selection- Random Forest

4.3 Reverse Engineering:

The performance of models was also analyzed by performing five fold cross validation. For each fold the mean error on the response variable prediction (Log error) was recorded. The error was the difference between the actual value and the predicted value of the log error.

The mean and standard deviation of this error were computed for each fold and points that lied outside of two standard deviations of the error were classified as **bad** predictions and the number of data points inside this range were classified as **good** predictions.

Table 12 below shows each of the the number of data points that are good predictions and bad predictions along with the standard deviation of the log error and the mean log error at each fold. It can be seen that the number of bad predictions for linear model exceed the number of bad predictions for decision tree and random forest model.

Model	**Folds**	**mean error**	**Std error**	**mean logerror**	**Std logerror**	**good prediction**	**bad prediction**
Linear Model	1	0.000263	0.05682	0.00952	0.09264	16030	17398
	2	-0.0000916	0.06295	0.00896	0.09002	24751	8447
	3	-0.000133	0.05490	0.00856	0.08957	27642	5298
	4	-0.000012	0.05438	0.00893	0.08887	27392	5251
	5	0.0000351	0.05342	0.00893	0.08783	27247	5393
Decision Tree	1	0.00032	0.07419	0.00927	0.09152	29728	3345
	2	-0.00055	0.07347	0.00852	0.09115	30137	3030
	3	-0.000079	0.07238	0.00856	0.08929	30039	2904
	4	0.00056	0.07189	0.00893	0.0891	30223	2881
	5	0.00062	0.07161	0.00893	0.0891	15953	16609
Random Forest	1	0.0003	0.05632	0.00927	0.09152	31288	2004
	2	-0.00033	0.05627	0.00852	0.09115	31261	1951
	3	-0.000023	0.05504	0.00856	0.08929	30795	1980
	4	0.00006	0.05527	0.00893	0.0891	31035	1928
	5	-0.00003	0.05512	0.00893	0.0891	31052	1911

Table 12: Analysis for 50% Data Set

For each model the number of data points in each of these categories were recorded and two csv files were created for each model to identify the differences between the good predictions and the bad predictions. From the csv files it was noted that for the models, log error values in the range $[-0.50775, -0.03)$ and $[0.06, 0.0508085]$ were bad predictions for the models.

Table 13 below shows the R2 and MSE for each fold. The MSE for linear models is high compared to Decision Trees and Random Forest at each fold and this aligns with the results of Table 12 as we have a large number of bad predictions for the linear model. Decision Forest and Random compete at each Fold, but Random Forest gives the bests results with a high R squared value and a low mean square value.

Model	Folds	R2 score	MSE	Std.MSE
Linear Model	1	0.623	0.0565	0.0565
	2	0.627	0.0553	0.0553
	3	0.454	0.0655	0.0655
	4	0.627	0.0540	0.0540
	5	0.623	0.0544	0.0544
Decision Trees	1	0.674	0.00276	0.0525
	2	0.676	0.00261	0.0511
	3	0.675	0.00259	0.0509
	4	0.672	0.00260	0.0510
	5	0.669	0.00257	0.0507
Random Forest	1	0.678	0.00273	0.0522
	2	0.678	0.00264	0.0514
	3	0.675	0.00263	0.0513
	4	0.685	0.00241	0.0491
	5	0.677	0.00252	0.0502

Table 13: Prediction of models in 50 % Data Set with hyper parameter optimization

5 Future Work and Improvement:

The project can be improved by:

- Using `GridSearchCV()` for all kernels of SVR with more variation and choice of parameters to tune. Currently no more than two parameter choice was given due to time constraints.
- Continuing reverse engineering to see why models give a bad predictions for log errors in the range $[-0.50775, -0.03)$ and $[0.06, 0.0508085]$.
- Performing stratified K Fold for Different kernels with SVR.
- Building additional new features with models and then testing the model.
- Investigating more non parametric models and repeating results.
- Using Spearman Filter Method for Feature selection.
- Further tuning hyper parameters using `RandomizedSearch()` for Decision Trees.

References

[1] *Zhao, K. (2000).Data visualization. 8. Retrieved: 22, p.2010.* 2000.

[2] The analysis factor. (2017). outliers: To drop or not to drop. [online] available at: https://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/ [accessed 1 dec. 2017].

[3] *Ge?ron, A. (2017).Hands-On Machine Learning with Scikit-Learn and TensorFlow. [S.l.]: O'Reilly Media, Inc.* 2017.

[4] Brownlee, j. (2017).why one-hot encode data in machine learning? - machine learning mastery. [online] machine learning mastery. available at: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/ [accessed 6 dec. 2017].

[5] Scikit-learn.org. (2017).model evaluation: quantifying the quality of predictions scikit-learn 0.19.1 documentation. [online] available at: http://scikit-learn.org/stable/modules/modelevaluation.html mean-squared-error [accessed 2 dec. 2017].

[6] population, r. (2017).rmse vs standard deviation in population. [online] stats.stackexchange.com. available at: https://stats.stackexchange.com/questions/264906/rmse-vs-standard-deviation-in-population [accessed 5 dec. 2017.

[7] Frost, j. (2017).regression analysis: How do i interpret r-squared and assess the goodness-of-fit?. [online] blog.minitab.com. available at: http://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit [accessed 2 dec. 2017].

[8] Guyon, i. and elisseeff, a. (2003).an introduction to variable and feature selection. journal of machine learning research, 3, pp.1157-11182. 2003.

[9] Postma, e. (2009).dimensionality reduction: A comparative review. j mach learn res 10, p.66-71. 2011.

[10] Kaggle.com. (2017).zillow prize: Zillows home value prediction (zestimate) kaggle. [online] available at: https://www.kaggle.com/c/zillow-prize-1/data [accessed 9 nov. 2017].

[11] Razali, n. (2011).power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. journal of statistical modeling and analytics, 2(1), p.23. 2011.

[12] Scikit-learn.org. (2017).decision trees scikit-learn 0.19.1 documentation. [online] available at: http://scikit-learn.org/stable/modules/tree.html [accessed 12 dec. 2017].

[13] Scikit-learn.org. (2017).sklearn.ensemble.randomforestclassifier scikit-learn 0.19.1 documentation. [online] available at: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.randomforestclassifier.html [accessed 18 nov. 2017].

[14] Analytics vidhya. (2017).tuning random forest model | machine learning | predictive modeling. [online] available at: https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/ [accessed 17 oct. 2017].

[15] Zheng, a. (2017).evaluating machine learning models. [online] o'reilly media. available at: https://www.oreilly.com/ideas/evaluating-machine-learning-models/page/5/hyperparameter-tuning [accessed 6 dec. 2017].

[16] Charlier, r. (2009).coastal planning and management. international journal of environmental studies, [online] 66(6), p.800. available at: https://doi.org/10.1080/00207230600836518 [accessed 12 nov. 2017].

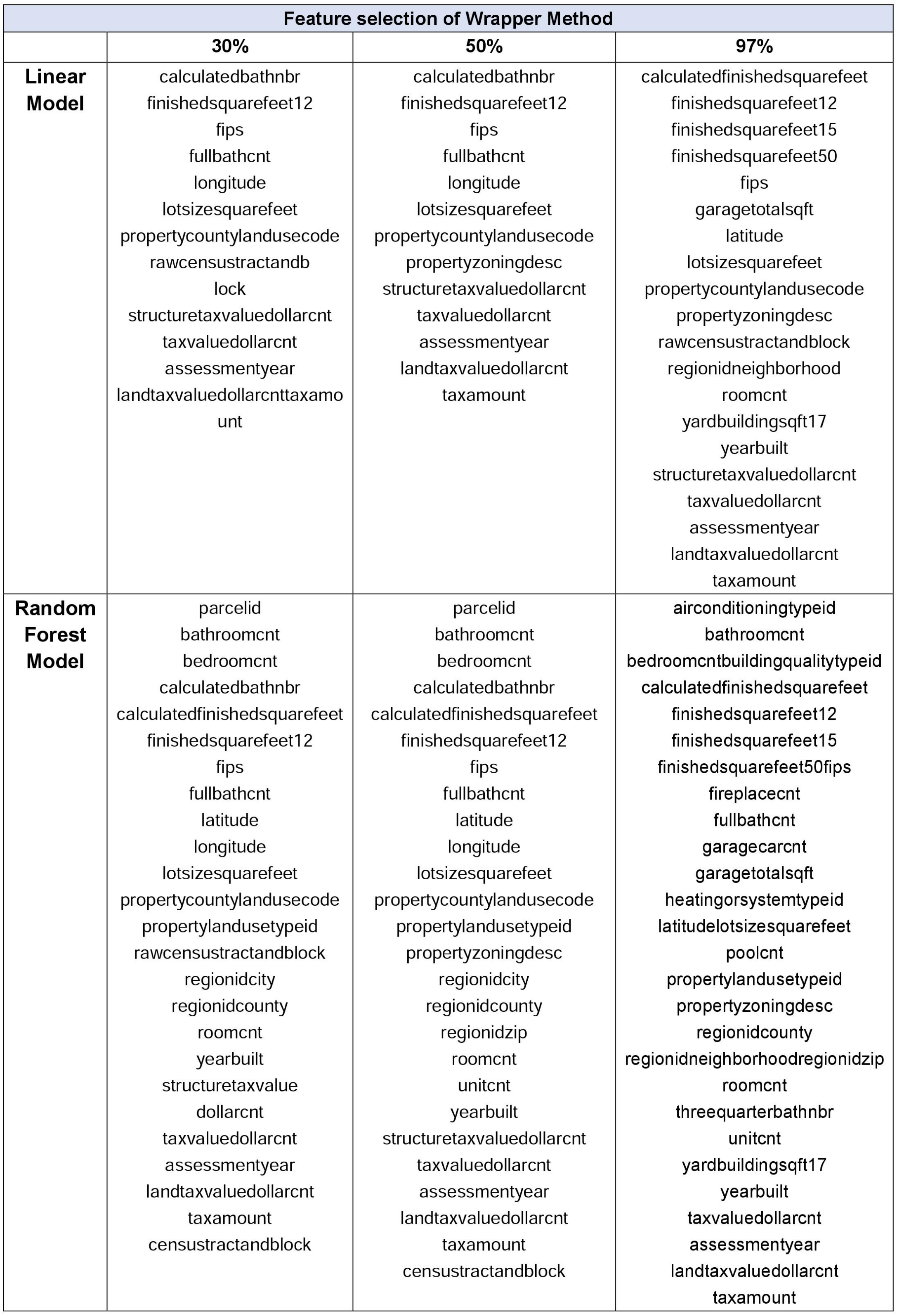
[17] Benalexkeen.com. (2017).linear regression in python using scikit-learn c ben alex keen. [online] available at: http://benalexkeen.com/linear-regression-in-python-using-scikit-learn/ [accessed 4 dec. 2017].

6 Appendix

This contains all tables and related content. for the project.

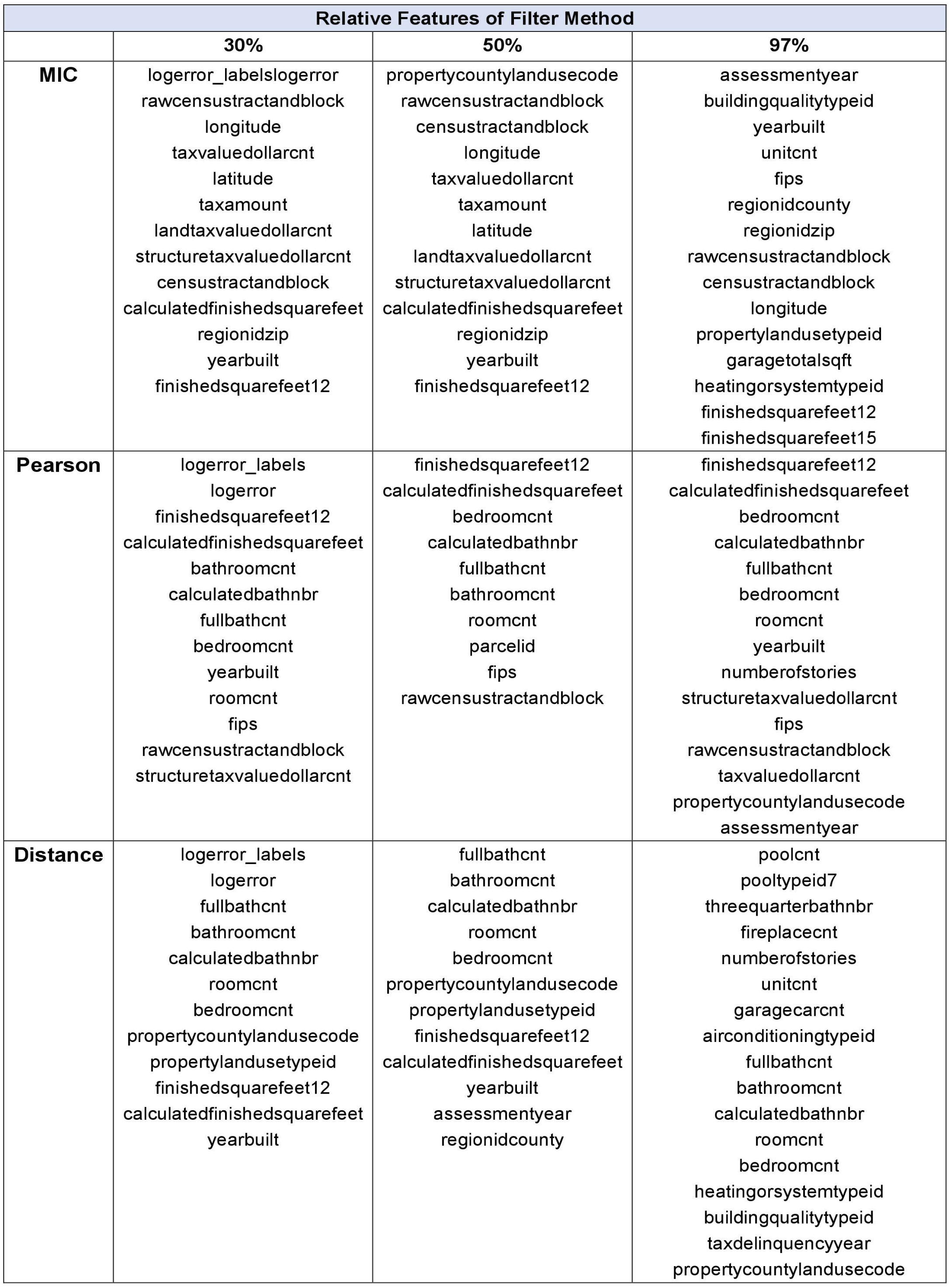
Features	**Spearman Correlation**
logerror	1.000000
basementsqft	0.330125
buildingclasstypeid	0.244949
finishedsquarefeet12	0.078404
calculatedfinishedsquarefeet	0.073464
calculatedbathnbr	0.064250
bathroomcnt	0.063668
fullbathcnt	0.059043
bedroomcnt	0.045316
numberofstories	0.045196
poolsizesum	0.038484
yearbuilt	0.037825
garagecarcnt	0.036429
fireplacecnt	0.036329
garagetotalsqft	0.036206
structuretaxvaluedollarcnt	0.031213
threequarterbathnbr	0.028589
architecturalstyletypeid	0.026799
taxvaluedollarcnt	0.026527
parcelid	0.023212
fips	0.020864
finishedsquarefeet15	0.020613
regionidzip	0.018121
roomcnt	0.017806
landtaxvaluedollarcnt	0.017700
lotsizesquarefeet	0.015810
censustractandblock	0.013357
regionidcity	0.012779
rawcensustractandblock	0.012596
longitude	0.011242
airconditioningtypeid	0.011084
assessmentyear	0.010414
taxamount	0.008095
propertylandusetypeid	0.003429
regionidneighborhood	0.000709
buildingqualitytypeid	-0.001134
unitcnt	-0.007740
latitude	-0.013298
finishedfloor1squarefeet	-0.013641
finishedsquarefeet50	-0.016487
regionidcounty	-0.020480
yardbuildingsqft26	-0.020915
finishedsquarefeet6	-0.023096
heatingorsystemtypeid	-0.026682
yardbuildingsqft17	-0.040516
taxdelinquencyyear	-0.043231
typeconstructiontypeid	-0.045499
finishedsquarefeet13	-0.138581

Table 14: Spearman Correlation of log error and all features



Feature selection of Wrapper Method			
	30%	**50%**	**97%**
Linear Model	calculatedbathnbr finishedsquarefeet12 fips fullbathcnt longitude lotsizesquarefeet propertycountylandusecode rawcensustractandb lock structuretaxvaluedollarcnt taxvaluedollarcnt assessmentyear landtaxvaluedollarcnttaxamo unt	calculatedbathnbr finishedsquarefeet12 fips fullbathcnt longitude lotsizesquarefeet propertycountylandusecode propertyzoningdesc structuretaxvaluedollarcnt taxvaluedollarcnt assessmentyear landtaxvaluedollarcnt taxamount	calculatedfinishedsquarefeet finishedsquarefeet12 finishedsquarefeet15 finishedsquarefeet50 fips garagetotalsqft latitude lotsizesquarefeet propertycountylandusecode propertyzoningdesc rawcensustractandblock regionidneighborhood roomcnt yardbuildingsqft17 yearbuilt structuretaxvaluedollarcnt taxvaluedollarcnt assessmentyear landtaxvaluedollarcnt taxamount
Random Forest Model	parcelid bathroomcnt bedroomcnt calculatedbathnbr calculatedfinishedsquarefeet finishedsquarefeet12 fips fullbathcnt latitude longitude lotsizesquarefeet propertycountylandusecode propertylandusetypeid rawcensustractandblock regionidcity regionidcounty roomcnt yearbuilt structuretaxvalue dollarcnt taxvaluedollarcnt assessmentyear landtaxvaluedollarcnt taxamount censustractandblock	parcelid bathroomcnt bedroomcnt calculatedbathnbr calculatedfinishedsquarefeet finishedsquarefeet12 fips fullbathcnt latitude longitude lotsizesquarefeet propertycountylandusecode propertylandusetypeid propertyzoningdesc regionidcity regionidcounty regionidzip roomcnt unitcnt yearbuilt structuretaxvaluedollarcnt taxvaluedollarcnt assessmentyear landtaxvaluedollarcnt taxamount censustractandblock	airconditioningtypeid bathroomcnt bedroomcntbuildingqualitytypeid calculatedfinishedsquarefeet finishedsquarefeet12 finishedsquarefeet15 finishedsquarefeet50fips fireplacecnt fullbathcnt garagecarcnt garagetotalsqft heatingorsystemtypeid latitudelotsizesquarefeet poolcnt propertylandusetypeid propertyzoningdesc regionidcounty regionidneighborhoodregionidzip roomcnt threequarterbathnbr unitcnt yardbuildingsqft17 yearbuilt taxvaluedollarcnt assessmentyear landtaxvaluedollarcnt taxamount

Figure 16: Best Features Wrapper Methods



Relative Features of Filter Method			
	30%	**50%**	**97%**
MIC	logerror_labelslogerror rawcensustractandblock longitude taxvaluedollarcnt latitude taxamount landtaxvaluedollarcnt structuretaxvaluedollarcnt censustractandblock calculatedfinishedsquarefeet regionidzip yearbuilt finishedsquarefeet12	propertycountylandusecode rawcensustractandblock censustractandblock longitude taxvaluedollarcnt taxamount latitude landtaxvaluedollarcnt structuretaxvaluedollarcnt calculatedfinishedsquarefeet regionidzip yearbuilt finishedsquarefeet12	assessmentyear buildingqualitytypeid yearbuilt unitcnt fips regionidcounty regionidzip rawcensustractandblock censustractandblock longitude propertylandusetypeid garagetotalsqft heatingorsystemtypeid finishedsquarefeet12 finishedsquarefeet15
Pearson	logerror_labels logerror finishedsquarefeet12 calculatedfinishedsquarefeet bathroomcnt calculatedbathnbr fullbathcnt bedroomcnt yearbuilt roomcnt fips rawcensustractandblock structuretaxvaluedollarcnt	finishedsquarefeet12 calculatedfinishedsquarefeet bedroomcnt calculatedbathnbr fullbathcnt bathroomcnt roomcnt parcelid fips rawcensustractandblock	finishedsquarefeet12 calculatedfinishedsquarefeet bedroomcnt calculatedbathnbr fullbathcnt bedroomcnt roomcnt yearbuilt numberofstories structuretaxvaluedollarcnt fips rawcensustractandblock taxvaluedollarcnt propertycountylandusecode assessmentyear
Distance	logerror_labels logerror fullbathcnt bathroomcnt calculatedbathnbr roomcnt bedroomcnt propertycountylandusecode propertylandusetypeid finishedsquarefeet12 calculatedfinishedsquarefeet yearbuilt	fullbathcnt bathroomcnt calculatedbathnbr roomcnt bedroomcnt propertycountylandusecode propertylandusetypeid finishedsquarefeet12 calculatedfinishedsquarefeet yearbuilt assessmentyear regionidcounty	poolcnt pooltypeid7 threequarterbathnbr fireplacecnt numberofstories unitcnt garagecarcnt airconditioningtypeid fullbathcnt bathroomcnt calculatedbathnbr roomcnt bedroomcnt heatingorsystemtypeid buildingqualitytypeid taxdelinquencyyear propertycountylandusecode

Figure 17: Best Features Filter Methods

			taxdelinquencyyear censustractandblock
Decision Tree Model	bathroomcnt bedroomcnt calculatedbathnbr calculatedfinishedsquarefeet finishedsquarefeet12 fips fullbathcnt latitude longitude lotsizesquarefeet propertycountylandusecode propertylandusetypeid rawcensustractandblock regionidcity regionidcounty regionidzip yearbuilt structuretaxvaluedollarcnt assessmentyear landtaxvaluedollarcnt taxamount censustractandblock	bathroomcnt bedroomcnt calculatedbathnbr calculatedfinishedsquarefeet finishedsquarefeet12 fips fullbathcnt latitude longitude lotsizesquarefeet propertycountylandusecode propertylandusetypeid propertyzoningdesc rawcensustractandblock regionidcity regionidzip roomcnt yearbuilt structuretaxvaluedollarcnt taxvaluedollarcnt assessmentyear landtaxvaluedollarcnt taxamount	bathroomcnt bedroomcnt calculatedbathnbr finishedfloor1squarefeet calculatedfinishedsquarefeet finishedsquarefeet15 fips fireplacecnt fullbathcnt garagecarcnt garagetotalsqft heatingorsystemtypeid latitude longitude lotsizesquarefeet poolcnt pooltypeid7 propertycountylandusecode propertylandusetypeid propertyzoningdesc rawcensustractandblock regionidcity regionidcounty regionidneighborhood regionidzip roomcnt threequarterbathnbr unitcnt yardbuildingsqft17 yearbuilt structuretaxvaluedollarcnt taxvaluedollarcnt landtaxvaluedollarcnt taxamount taxdelinquencyyear censustractandblock

s

Figure 18: Stratified K Fold results for Models