EE 660 Project

Airbnb Dataset Price Prediction

Project Report

By

1. Omer Solakli
2. Rohan Amrapurkar
3. Tamoghna Chattopadhyay

Content

1. **Abstract**

3

1. **Problem Statement** 4
   1. Literature Review 4
   2. Prior and Related Work 4
   3. Overview of Approach 5
2. **Project Formulation**
   1. Dataset and Exploratory Data Analysis
   2. Data Pre-Processing
   3. Feature Selection
      1. Feature Selection using Lasso
      2. Feature Selection using Ridge
   4. Main Dataset
   5. Model Selection
      1. Regression
      2. Discussion on Sparsity
      3. Decision Trees
      4. Random Forests
      5. Boosting
   6. Performance Metrics
      1. R2 or Coefficient of Determination
      2. Mean Squared Error
      3. Mean Absolute Error
      4. Median Absolute Error
3. **Results**
4. **Interpretation of Results**
5. **Future Work**
6. **References**
7. Abstract

The Airbnb Housing price validation is important for potential users of the website – both property listers and customers to make informed decisions and can help in establishment of real estate policies. There are a lot of factors which play an important role in determining the price of the listing, such as the neighborhood, Number of rooms in the house, Number of amenities provided etc. This study uses machine learning algorithms as a research methodology to develop a listing price prediction model. This study analyses the price predictability capability of different algorithms like linear regression, tree based regressions etc. along with other attributes like sparsity, algorithmic complexity etc. The performance of each model is evaluated using four different metrics, the value, mean squared error, mean absolute error and median absolute error.

*Keywords :* Regression, Decision Tree, Boosting, Ensemble Learning, Random Forest, Airbnb dataset, Listing Price Prediction, Regularization

1. Problem Statement

This study explores the use of different machine learning regression algorithms to predict the nightly price of an Airbnb listing in the city of Los Angeles. It also aims to find the best features of the listing (e.g., Locality, Number of guests it accommodates, Number of beds, Amenities, Ratings, Reviews, etc.) which help in accurately predicting the nightly price. It includes 2 methods of feature selection, data preprocessing, and cross-validation for parameter selection. Another of the goals is to compare various metrics - R2, Mean Squared Error, Mean Absolute Error and Median Absolute Error - obtained after applying several algorithms like Linear Regression, Random Forest Regression, Gradient Boosting Regression, Lasso Regression, Ridge Regression, XgBoost, and AdaBoost Regression using Decision Tree.

Airbnb’s popularity is increasing, it is now bigger than the top five hotel brands put together. Surprisingly enough, it does not own any of the spaces listed on its website! Hosts list their spaces on the website for customer to reserve. The methods in this study can be used as a reference for hosts to set the price on their listings. This creates a sort of a recommender system, which helps hosts and gives them an approximate estimate of the value they can get for their listing.

The complexity of the problem arises due to its high dimensionality, type of features and the difference in the dataset caused by small changes in features. The dataset has a large number of data points which increases the time complexity. Also, it contains many features, 95 to be exact. Of these, 35 are numeric type, including both continuous and discrete. There are 7 categorical features and 53 others too. Preprocessing all these features increases the complexity of the process. Introducing sparsity through lasso regularization helps in reducing the high dimensionality of feature space. Also, the house can be similar, but the price of the listing can change just based on one feature, i.e., the locality or neighborhood. This possible dependencies increase the complexity too. Thus, the main complexities arise from the huge amounts of data points and the number of preprocessing steps required to reduce the high feature dimensionality.

* 1. Literature Review

Most of the information about the different algorithms used are from class notes and the *Scikit* learn library documentation. The documentations consist of detailed descriptions of all the algorithms used along with examples of use cases. Also, *stack exchange* and *Quora* was utilized to get an explanation on gradient tree boosting and it’s difference with *XgBoosting*.

* 1. Prior and Related work – None
  2. Overview of Approach

Our approach to solve the problem is not to reduce the dimensionality without having a significant idea. We know that location is crucial for Airbnb listing prices and we didn’t want disregard it. One hot encoding the neighborhood and the other categorical features increased the dimensionality of the problem. To reduce it, we used exploratory data analysis to assess which features are significant and which regression methods to use based on exploratory data set.

We would like to reduce the dimension using ridge and lasso feature selection so that to assess which neighborhoods (and other categorical features) in Los Angeles are actually important feature for its price. The reason why we are using two feature selection technique lies behind the fact that Lasso giving more sparse solutions for the optimization problem, hence more reduced dimensionality whereas Ridge yields higher number of features both depending of the value of the penalty parameter alpha. Hence, we wanted to see which would work better for our case.

After reducing the dimension in both ways, we simply built our hypothesis set based on the exploratory dataset and tuned the parameters using cross-validation. We use the parameters that results in minimum MSE since it’s the base error metric. Then, we trained and tested using the original dataset.

3 Project Formulation and Implementation

The Dataset used for this study is sourced from the *Insider Airbnb* website. It is a collection of all the publicly available information from the *Airbnb* website. It contains information on all the listings in Los Angeles on November 4, 2018.

The approach to solve this problem was is shown in Figure 1. It follows standard techniques in solving regression problems, in preprocessing data, reducing complexity and dimensionality, selecting parameters and models and accessing the performance.

*Figure 1*

* 1. Dataset and Exploratory Data Analysis

The Dataset contains 43205 data points and consists of 95 features like, latitude, longitude, zip code, neighborhood, accommodates, number of bathrooms, bedrooms, beds, amenities etc. It also includes the price of the listing as put up by the host. A sample of what the dataset looks like is shown in Figure 2. Some of the important features and their data types are listed in the table below.

The dataset was split into 2 parts, the first containing 5000 data points, randomly selected from the dataset to be used for Exploratory Data Analysis. The remainder of the dataset, about 38000 data points were used to train models and evaluate their performance.

Exploratory Data Analysis was helpful in selecting the best set of features and in turn reduce the dimensionality of the models. The Lasso and Ridge models were used as feature selection techniques for the problem. The Lasso model was used to increase the sparsity of the weights and the Ridge model was used to penalize the weights in turn reducing the weight values.

It was also useful in making meaningful visualizations, like developing a heatmap of the properties with respect to their costs. It gave an estimate of the variation of nightly price with respect to features like number of beds, number of people it accommodates, bed type, property type, room type, review and locality.

A screen shot of a computer

Description generated with very high confidence

*Figure 2: Sample of the dataset*

|  |  |  |
| --- | --- | --- |
| Feature Name | Interpretation | Type |
| id | Notation for the listing | Numeric |
| host\_since | The date since host has been a user | Categorical |
| neighbourhood | Neighbourhood of the house | String |
| zipcode | ZIP | Numeric |
| latitude | Latitude of the location | Numeric |
| longitude | Longitude of the location | Numeric |
| property\_type | The type of the listing | Categorical |
| room\_type | The type of room in the listing | Categorical |
| bathrooms | Number of bathrooms in the house | Numeric |
| bedrooms | Number of bedrooms in the house | Numeric |
| bed\_type | Type of bed in the bedroom | Categorical |
| amenities | Types of amenities provided | String |
| price | Price of the listing | Numeric |

*Important features presented in the dataset.*

The Figures below, show visualizations made using the exploratory dataset.

A picture containing text, map

Description generated with very high confidence

Figure 3: Heat Map of prices in Los Angeles



Figure 4



Figure 5

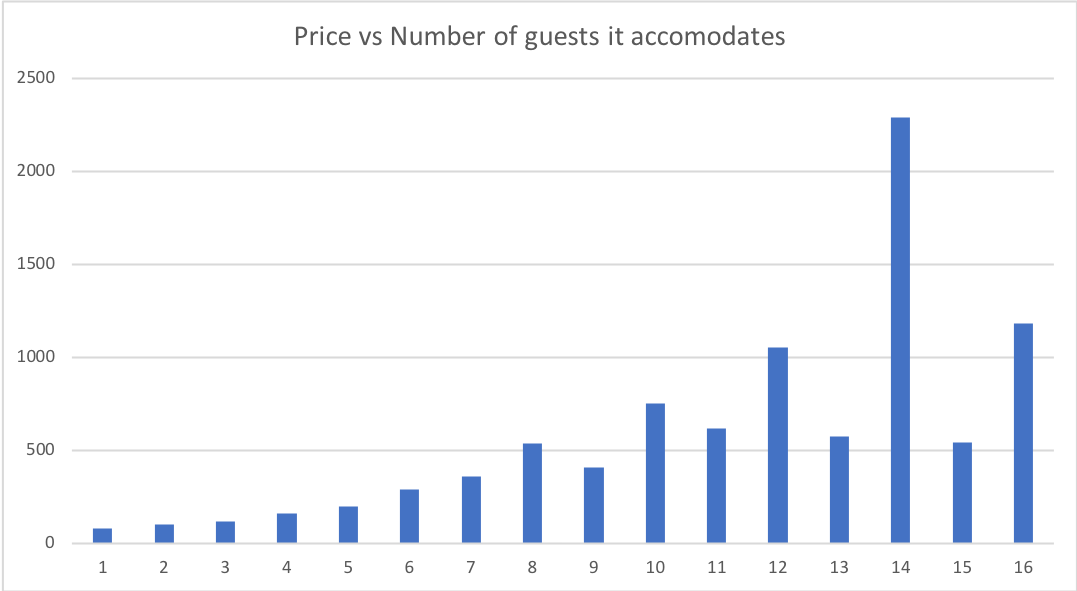


Figure 6

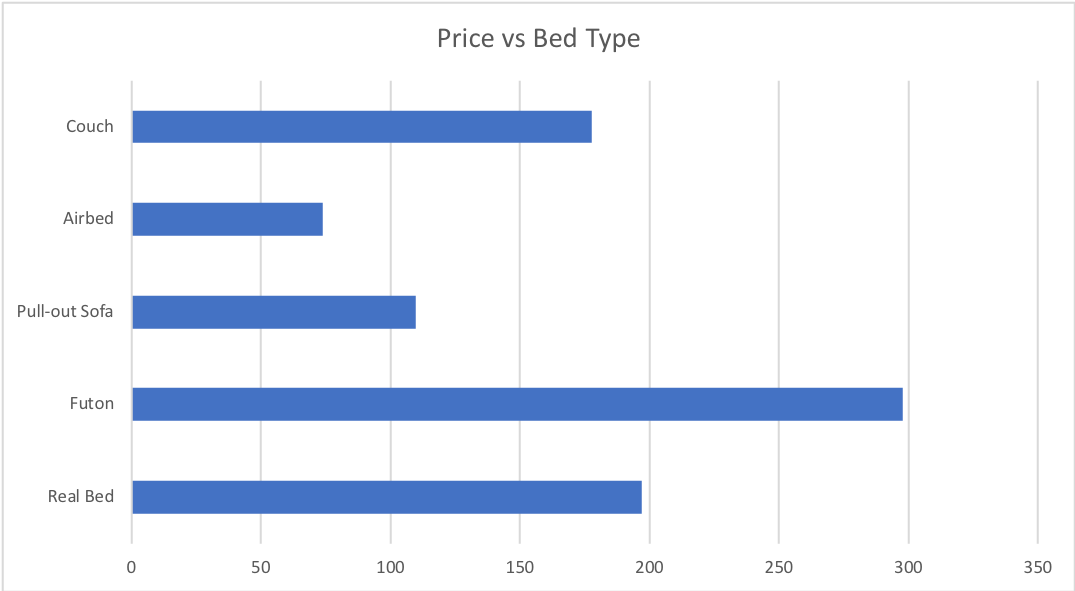


Figure 7

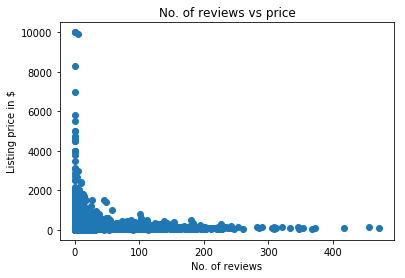


Figure 8

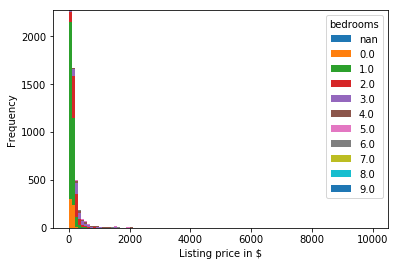


Figure 9

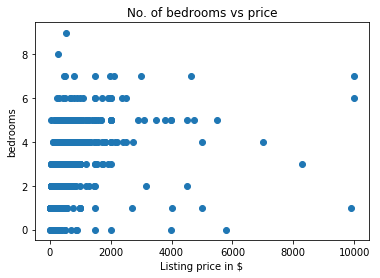


Figure 10

* 1. Data Preprocessing

After analyzing the exploratory dataset, it was observed that there are several features which are redundant. It is fair to assume that features like *host identity, host verification, guest profile picture, url, calendar last scraped* and a few others would not play a role in predicting or determining the price of a listing and hence were subsequently dropped. Further, there were a few features and data points with missing values. In case the data point (listing) had more than 10 missing feature values, the data point was dropped as well. Similarly, if a feature had more than 30% missing data, the feature was dropped. Finally, listing which were ‘removed’ but which were still a part of the dataset were deleted as well. To maintain uniformity, features the contained *‘$’* and *‘,’* were converted to floating point values and the symbols were dropped.

There were several features which were text based. Amenities, was a feature with each point being a list of utilities, (e.g., *{TV, WI-FI, AC, Heater etc.}* ). The number of utilities in each list was counted and each data point was replaced with this value. The data also included categorical features like bed type, property type neighborhood etc., these were either one hot encoded, or given numeric value or ranking. Features like property type and bed type were given numeric values from 1 to 5. This helped in giving a higher score to bed types which were *‘Real Beds’* and a lower score bed types which were *‘Air Beds’*. A similar approach was used for room types as well. The remainder of the categorical features were one hot encoded into a binary format.

Imputation was necessary on features with missing data points. Depending on the type of feature, the missing data points were imputed either using the feature mean or the feature median. Features with data type object were imputed using the most frequent value in the column and features with any other data type were imputed using the mean of the feature. The *get\_dummies* class in *pandas* was used for *one hot encoding* (converting to binary) categorical features.

Finally, a simple *standardization* technique of converting the data into zero mean and unit variance was implemented by subtracting each data point with the mean and then dividing the resultant by the standard deviation. This was carried out using the *preprocessing* library in *sklearn* and the *Standard Scaler* class.

After the data preprocessing step, the number of features increased to 283. Same preprocessing techniques were applied to the main dataset as well.

* 1. Feature Selection

For datasets with a large number of features or dimensionality, it is important to reduce the complexity. In case of a large number of features, there is a possibility that several features turn out to be redundant, several features have either the same value or a very low variance. The model does not tend to ‘learn’ much from such features. These kinds of features need to be identified and eliminated. For this reason, feature selection and dimensionality reduction is important. 2 methods of feature selection, Lasso and Ridge, are used and applied to the models mentioned in the abstract. The libraries *linear\_models and metrics* from *sklearn* was used.

* + 1. Feature Selection using Lasso based techniques

In our problem we use one hot encoding for neighborhood information, that leads increase in the dimension of our data. We need to assess which features are important for our final models to consider. In order to do that, we standardize the features so that we can assess their importance using their weight values. According to Tibshirani’s book, the Lasso Method for Variable Selection in the Cox Model, he advises to standardize the dummy variables for regularization purposes so that penalization is fair for all features. Thus, for feature selection purposes we standardize the categorical features but for the model selection we do not.

Lasso is also a regularization technique based on L1 norm. It estimates sparse coefficients, i.e., it reduces the number of variables upon which the solution depends. This is done by assuming a Laplacian prior on the weights which yields an objective function which is:

Here, is the penalty that allows for the increase in sparsity of the weight vector. A larger value of will result in a more sparse solution. also penalizes the weights in terms on magnitude, resulting in a more sparse and lower magnitude solution of weights.

In this case, cross-validation is used to select the best value (also called α in the code) for the L1 regularizer. The validation curve was then plotted, and the α value was selected on the basis of best generalization. Which simply means that, select that alpha value where the median absolute error (or R2 score) for the train set is equal to or almost equal to the median absolute error (or R2 score) of the validation set. Once the best α has been selected, it is used to select features. The unimportant features were considered those with weight values in the interval of [-0.1, 0.1]. A new data frame was then created, dropping the unimportant features, for training the model.

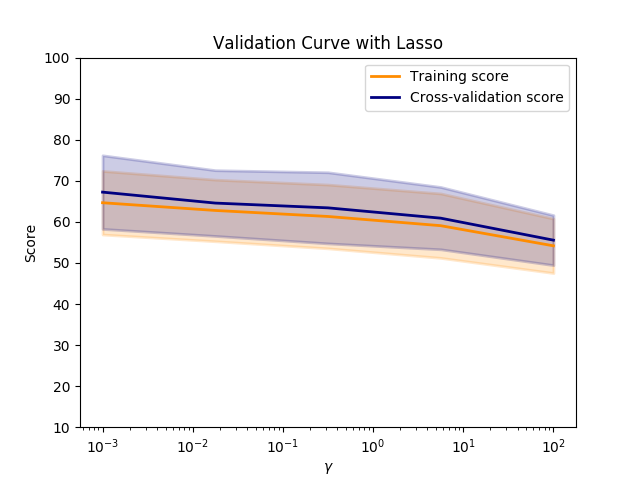


Figure 11: Validation curve for Lasso MSE



Figure 12: Validation curve for Lasso R square

* + 1. Feature Selection using Ridge based techniques

This is also termed as L2 regularizer as it is based on L2 norm or the Euclidean distance. By imposing a Gaussian prior to the weights, where the variance controls the strength of the prior we obtain an objective function with a term that controls the weights.

Here α is the complexity penalty that controls the weight vector. Larger the value of , greater is the amount of shrinkage. The algorithmic complexity is the same as the linear regression without regularization.

In this case, cross-validation using train-test split is used to select the best value (also called α in the code) for the L2 regularizer. The validation curve was then plotted, and the α value was selected on the basis of best generalization. Which simply means that, select that alpha value where the median absolute error (or R2 score) for the train set is equal to or almost equal to the median absolute error (or R2 score) of the validation set. Once the best α has been selected, it is used to select features. The unimportant features were considered those with weight values in the interval of [-4, 4]. A new data frame was then created, dropping the unimportant features, for training the model.

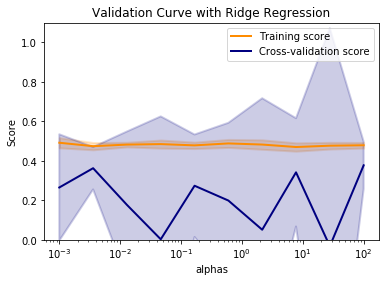


Figure 13: Validation curve with Ridge Regression

Each of these feature selection techniques were used for the various machine learning models mentioned earlier and their performances compared.

As we are applying correlation robust algorithms like random forest and xgBoost, we do not care for correlation, so we don’t use PCA (Principal Component Analysis) for future selection.

* 1. Main Dataset

After successfully selecting the most important features using the feature selection techniques mentioned above, the remainder of the dataset was used to train and test the performance various machine learning models.

The data preprocessing techniques that were used for the main dataset were similar to the ones used for the exploratory dataset. However, for the main dataset, there were approximately 35383 data points that could be used after removing or deleting data points with missing values. This dataset was then divided into training dataset, validation dataset and test dataset.

After preprocessing, the main dataset, the unimportant features as identified earlier were dropped essentially keeping only the important features. Since there were two methods of feature selection, each machine learning model was implemented two times, using each feature selection technique.

*Linear Regression, Lasso Regression, Ridge Regression, Gradient Boosting Regressor, Random Forest Regressor, AdaBoost Regressor, and XgBoost Regressors* were used to predict the nightly price of each listing in the test dataset along with cross validation to select the best parameters for training.

The Figure below shows the steps involved in predicting the prices of using the main dataset.

*Figure 14: Workflow on main dataset*

* 1. Model Selection
     1. Regression

Linear Regression fits a linear model with coefficients w = (w1 , …, wn) such that it minimizes the sum of square of the error between observed response and the predicted response by the linear approximation.

In terms of algorithmic complexity, even though linear regression has a closed form solution, the implementation on *sklearn* computes the solution using singular value decomposition of X. So, if X is a matrix of size (m, n), then the cost is O(mn2), where, n >= m.

Section 3.3 describes the regularization techniques (L1 and L2 regularizers). These can also be used for training regression models and predicting outputs.

For the exploratory dataset, a 5 fold cross validation is used to estimate the best parameter α for Lasso and Ridge based Regression. No cross validation was performed for simple linear regression. Similar cross validation techniques were used for the main dataset as well.

3.5.2 Discussion on Sparsity

In the above section, we briefly touched on the topics of using different regularization to obtain a sparse or non-sparse solution. The other way of introducing sparsity would be to change the cost function. For example, ridge regression uses the sum of squares of errors as the loss function and when we include L2 regularization, we get a non-sparse solution. But when L1 regularization term is added, i.e., lasso regularization, we introduce sparsity through the L1 lasso term.

* + 1. Decision Trees

Decision Trees are non-parametric supervised learning method that predicts the value of the target variable by learning simple decision rules inferred from the data features. The algorithmic complexity of using the tree is logarithmic in number of data points used to train the data. As the number of nodes (or depth) of the tree increases, the hypothesis space grows. In the worst case, exponentially many nodes might be needed as each leaf node is used to represent each possible input. Several base estimators can be combined to improve the generalization in a method called ensemble learning. Two types of ensemble learning are:

1. Averaging Methods – In this method, we build several estimators independently and then average the prediction. Example – *random forests*, *bagging* etc.
2. Boosting Methods – In this method, the base estimators are built sequentially tying to reduce the bias of the combined estimator. Example – *Adaboost*, *gradient tree boosting* etc.

3.5.4 Random Forests

The algorithm creates several decision trees by introducing randomness in the construction and average them to obtain the final regressor. While training each of the decision trees, a random set of samples from the training set are drawn with replacement.

A cross validation technique was used to select the best values for *number of estimators*. The criterion used was *‘mse’*. It was observed that a good value for *n\_estimators* is 200.

3.5.5 Boosting

There are three methods of Boosting which we used:

1. *Adaboost* – In this, a sequence of weak learners is fit on modified versions of data and a weighted majority vote is combined through them to produce the final predictions.
2. *Gradient Tree Boosting* – In this, we start with a differentiable loss function to minimize. We minimize with respect to the predictor function so that the total loss is minimized. This error is minimized by building models sequentially and combining them. New models are added to existing models to correct the errors until no further improvements can be made.
3. *xgBoost* – The principle is similar to Gradient Boosting, the difference being that it uses a more regularized model to control over-fitting and hence gives better performance. It also has improved data structures for better cache utilization which makes the algorithm faster.

While implementing Boosting algorithms, 5-fold cross validation was done in each case, to find the best value of the parameters – depth of the tree and number of estimators. For example, in the case of *Adaboost* algorithm, we got the value of depth of tree as 13 and the number of estimators as 10 in the case of lasso regularization feature selected dataset. Similarly, this was done for all the boosting algorithms. The Boosting methods were implemented for both the exploratory and main datasets.

Some of the important parameters for the boosting algorithms taken under consideration are:

1. *n\_estimators* - The number of boosting stages to perform by the algorithm. The default value for it is 100.
2. criterion - The function to measure the quality of a split. Supported criteria are *“mse”* for the mean squared error, which is equal to variance reduction as feature selection criterion and minimizes the L2 loss using the mean of each terminal node, *“friedman\_mse”,* which uses mean squared error with Friedman’s improvement score for potential splits, and *“mae”* for the mean absolute error, which minimizes the L1 loss using the median of each terminal node. The default is *mse*.
3. *max\_depth* - The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than *min\_samples\_split* samples.

3.6 Performance Metrics

The different performance metrics used by us to compute the errors in the predictions and compare our results are as follows:

3.6.1 R2 or Coefficient of Determination

It is the amount of variation in the data that is explained with the model. It gives information about the goodness of fit of the model. An R2 value of 1 indicates that the regression line perfectly fits the data.

This is called the total sum of squares and is proportional to the variance of the data. Also, there is regression sum of squares,

So, the error is expressed as:

3.6.2 Mean Squared Error

It is the mean value of the squared deviations of the predictions from the true value over the test space. The *MSE* is desired to be as low as possible. A *MSE* of 0 means that the model predicts the observations perfectly.

3.6.3 Mean Absolute Error

It is a measure of the difference between two continuous variables. This uses the same scale as the data being measured and hence is a scale-dependent measurement.

3.6.4 Median Absolute Error

The median absolute error is useful basically it is essentially insensitive to outliers (as long as there aren't too many of them). This is because it is the median of all the absolute values of the residuals, and the median is unaffected by values at the tails. So, this loss function can be used to perform [robust regression](https://en.wikipedia.org/wiki/Robust_regression). In contrast, the mean squared error can be highly sensitive to outliers, and mean absolute error can be somewhat sensitive to outliers (although less so than the mean squared error).

Note that using the median absolute error only corrects for outliers in the response/target variable, not for outliers in the predictors/feature variables.

All these error metrics were applied to results of the different models on both the exploratory and main dataset and the results are compared based on these metrics.

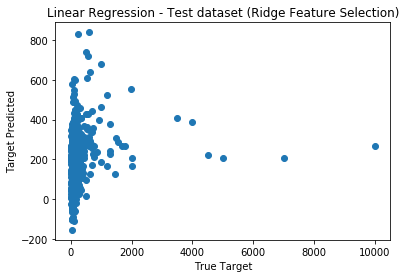
1. Results

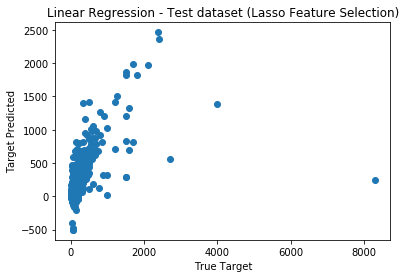
For the exploratory dataset, the results for training data are as followed :

|  |  |  |
| --- | --- | --- |
| **Model** | **Lasso Regularization Feature Selection** | **Ridge Regularization Feature Selection** |
| Linear Regression | Train R2: 0.460771  Train MSE: 112699.0  Train MAE: 120.368  Train MdAE: 71.8074 | Train R2: 0.470278  Train MSE: 114057.0  Train MAE: 123.621  Train MdAE: 71.9417 |
| Ridge Regression | Train R2: 0.50464824774  Train MSE: 91635.9200188  Train MAE: 114.647350181  Train MdAE: 67.9775831791 | Train R2: 0.460996584788  Train MSE: 107741.014363  Train MAE: 118.393858424  Train MdAE: 69.7982541206 |
| Lasso Regression | Train R2: 0.493520552963  Train MSE: 93694.4502327  Train MAE: 109.662524432  Train MdAE: 57.1338404249 | Train R2: 0.473940133237  Train MSE: 112007.443673  Train MAE: 115.25785065  Train MdAE: 63.0713711506 |
| Random Forest | Train R2: 0.45448  Train MSE: 14364.36917  Train MAE: 31.629587912  Train MdAE: 20.66500 | Train R2:  Train MSE:  Train MAE:  Train MdAE: |
| XGBoost | Train R2: 0.928866038768  Train MSE: 12685.1040267  Train MAE: 44.696708187  Train MdAE: 26.2062416077 | Train R2: 0.634045303358  Train MSE: 72638.2715097  Train MAE: 107.721115738  Train MdAE: 57.9860572815 |
| AdaBoost | Train R2: 0.987718030819  Train MSE: 2658.38462186  Train MAE: 33.3731058558  Train MdAE: 21.9298428844 | Train R2: 0.98882  Train MSE: 2407.29  Train MAE: 30.9086  Train MdAE: 19.8574 |
| Gradient Boost | Train R2: 0.919835  Train MSE: 16144.65693  Train MAE: 49.8042031  Train MdAE: 29.32140 | Train R2: 0.902682  Train MSE: 18568.47489  Train MAE: 53.209386  Train MdAE: 34.57892 |

For the exploratory dataset, the results for testing data are as followed :

|  |  |  |
| --- | --- | --- |
| **Model** | **Lasso Regularization Feature Selection** | **Ridge Regularization Feature Selection** |
| Linear Regression | Test R2: 0.165278  Test MSE: 182458.304729  Test MAE: 127.228907  Test MdAE: 66.218658 | Test R2: 0.041332  Test MSE: 244336.123976  Test MAE: 167.092359  Test MdAE: 95.460335 |
| Ridge Regression | Test R2: 0.4146522512  Test MSE: 189868.19866871  Test MAE: 122.19659277  Test MdAE: 62.378580583 | Test R2: 0.300629965  Test MSE: 384219.361161  Test MAE: 150.01496  Test MdAE: 77.1115613666 |
| Lasso Regression | Test R2: 0.5450  Test MSE: 88199.3436321  Test MAE: 119.3002183  Test MdAE: 56.1952260469 | Test R2: 0.48336956  Test MSE: 136719.0218  Test MAE: 114.523376119  Test MdAE: 66.06521297 |
| Random Forest | Test R2: 0.43365  Test MSE: 126060.4462  Test MAE: 88.934  Test MdAE: 28.993 | Test R2: 0.398257  Test MSE: 144212.8261  Test MAE: 92.4763  Test MdAE:33.4378 |
| XGBoost | Test R2: 0.3668  Test MSE: 168546.489097  Test MAE: 89.325812  Test MdAE: 26.287411 | Test R2: 0.111479  Test MSE: 230790.129840  Test MAE: 136.355123  Test MdAE: 41.748962 |
| AdaBoost | Test R2: 0.375405  Test MSE: 73734.373263  Test MAE: 76.066478  Test MdAE: 27.311435 | Test R2: 0.050012  Test MSE: 412278.382105  Test MAE: 158.279013  Test MdAE: 60.753428 |
| Gradient Boost | Test R2: 0.3413085  Test MSE: 152649.933  Test MAE: 87.8957  Test MdAE: 30.2264 | Test R2: 0.2638  Test MSE: 174211.7132  Test MAE: 91.3718  Test MdAE:34.2238 |





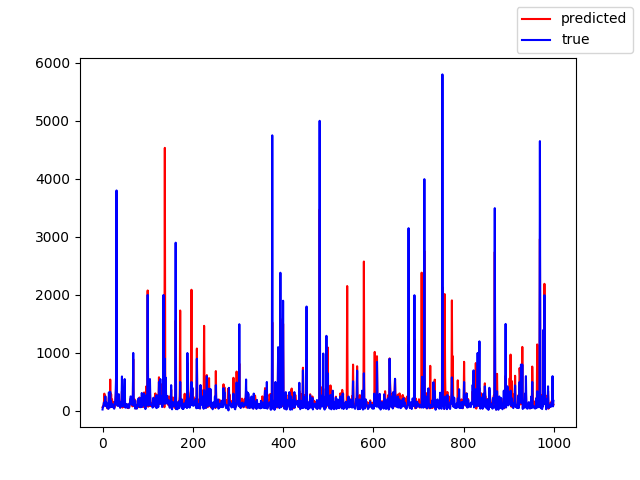
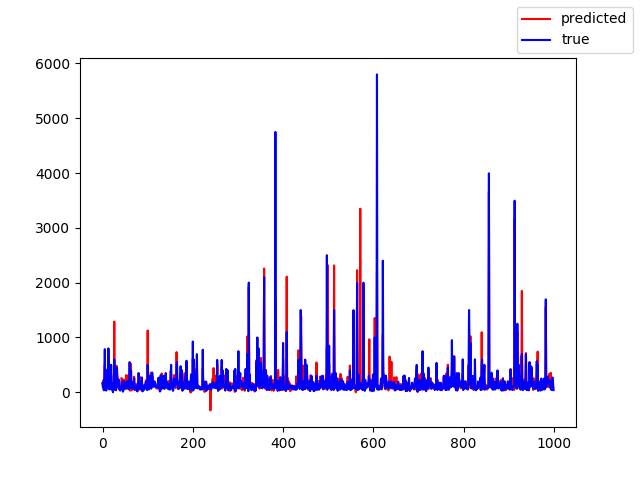


Figure 3: Random Forest



Number of Data points

Price

Figure 4: Gradient Boosting

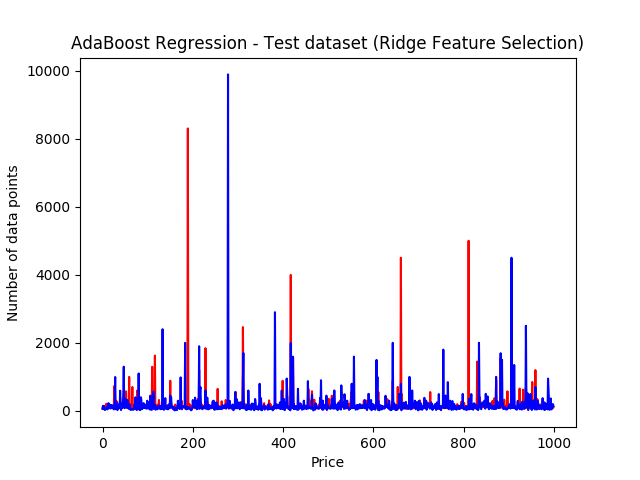


Figure 5

A screenshot of a cell phone

Description generated with very high confidence

The results for main data are as followed :

|  |  |  |
| --- | --- | --- |
| **Model** | **Lasso Regularization Feature Selection** | **Ridge Regularization Feature Selection** |
| Linear Regression | Train R2: -0.2814879823316603  Train MSE: 153424.35443635285  Train MAE: 132.55971937161888  Train MdAE: 71.95548266110382 | Train R2: -0.3273716582342536  Train MSE: 121994.64264418682  Train MAE: 141.23332483869575  Train MdAE: 77.32151689019065 |
| Ridge Regression | Train R2: -0.25668206199  Train MSE: 148808.490143238  Train MAE: 133.30948186236418  Train MdAE: 72.45225402426422 | Train R2: -0.24068254203341644  Train MSE: 139204.5498635516    Train MAE: 129.18589469575332  Train MdAE: 70.46845449601551 |
| Lasso Regression | Train R2: -0.35199520770196946  Train MSE: 174248.30172211627  Train MAE: 141.5286187198581  Train MdAE: 76.88620349476787 | Train R2: -0.36931801893340577  Train MSE167249.8699284558  Train MAE: 135.07568486381757  Train MdAE: 73.17817032393305 |
| Random Forest | Train R2: 0.4224926575335006  Train MSE: 87617.63880580584  Train MAE: 86.50274063337751  Train MdAE: 43.26371684012834 | Train R2: 0.4477526330252414  Train MSE: 74379.69490052853  Train MAE: 81.99815229206335  Train MdAE: 39.45978848470595 |
| AdaBoost | Train R2: 0.632454832971155  Train MSE: 113277.28102049492  Train MAE: 165.9149881947965  Train MdAE: 82.58897414336766 | Train R2: 0.5725548529721824  Train MSE: 99094.46515973457  Train MAE: 206.37796918422893  Train MdAE: 160.2516129032258 |
| Gradient Boost | Train R2: 0.557499853172446  Train MSE: 76383.0417928772  Train MAE: 79.74280724239341  Train MdAE:33.20358862805305 | Train R2: 0.8967316663193983  Train MSE: 23798.56641868538  Train MAE: 55.215840812925286  Train MdAE: 25.10667726703673 |

|  |  |  |
| --- | --- | --- |
| **Model** | **Lasso Regularization Feature Selection** | **Ridge Regularization Feature Selection** |
| Linear Regression | Test R2: -0.9802267923058938  Test MSE: 211242.64378285082  Test MAE: 138.49505811232814  Test MdAE: 73.35424287382557 | Test R2: -0.020305724761928934  Test MSE: 173634.8543017877  Test MAE: 137.4722986766043  Test MdAE: 77.12418726086798 |
| Ridge Regression | Test R2: -1.0446320562887434  Test MSE: 231556.28012920715  Test MAE: 136.7893948618803  Test MdAE:71.012938184536 | Test R2: -2.6123885916133416  Test MSE: 156846.6845009356  Test MAE: 140.37919739192486  Test MdAE: 66.75215283743577 |
| Lasso Regression | Test R2: 0.10568783638645152  Test MSE: 131103.97022778765  Test MAE: 137.38874849658103  Test MdAE: 78.03635659246538 | Test R2: -0.221952620892941  Test MSE: 156846.6845009  Test MAE: 139.6404856938470  Test MdAE: 74.4560377001344 |
| Random Forest | Test R2: 0.20824186439653047  Test MSE: 115001.40550877777  Test MAE: 91.60620320862277  Test MdAE: 42.42915080597352 | Test R2: -0.12013584914521847  Test MSE: 189181.89437012997  Test MAE: 95.04562021580199  Test MdAE: 38.96292271635451 |
| AdaBoost | Test R2: 0.3124977166677271  Test MSE: 144811.86730471934  Test MAE: 172.19095106734986  Test MdAE: 82.58897414336766 | Test R2: 0.22453751161206315  Test MSE: 216741.17709803724  Test MAE: 218.43505588071022  Test MdAE: 160.9939968056397 |
| Gradient Boost | Test R2: 0.3828900722944142  Test MSE: 121342.91331865962  Test MAE: 84.08776943213483  Test MdAE: 32.33825657765277 | Test R2: 0.5505589141900278  Test MSE: 97786.77322384185  Test MAE: 72.20184344842117  Test MdAE: 25.90042367237018 |

1. Final Results and Interpretation
2. The results of feature selection for the dataset showed that not all the features were necessary. We can introduce sparsity using lasso and ridge regularization and decrease the number of features.
3. The use of regularization has huge impact in improving the performance. This implies that the low performance of linear regression is not due to poor generalization but due to the inherent inability of the chosen model to not fit the data completely.
4. Tree based regressors are fairly simple but they produce noisy & weak regressors. The good thing is they are fairly robust to outliers but the regression accuracy might not as good as other models.
5. Using bagging one can fit many large trees to bootstrap resampled versions of the training data and classify by mean of the each region created.
6. The ability of decision trees to fit non linear relationships enables better performance than linear regressor. Boosting techniques enhances this performance.
7. Future Work
8. The current formulation does not use the reviews by previous users while developing the model. This could be a key component in the process of house price prediction. But this requires Natural Language Processing, which is outside the scope of this course.
9. Future work can include more parameter fine tuning to get more accurate model.
10. Also, RFECV Feature selection method can be explored to check it’s results with the different models.
11. Clustering with Kmeans algorithm can be used to cluster the listings according to zipcode or neighborhood and then predict the prices. This method has not been explored but has potential to improve results.
12. Also, Neural Network approach has not been explored to fit the dataset. As Neural Networks can learn detailed patterns, hence it can be a good algorithm to explore.
13. References
14. Thibshirani, Robert. *The Lasso for Variable Selection in the Cox Model*. Natural Sciences and Engineering Research Council of Canada, 1998.
15. Linear Models : sklearn Documentation
16. Decision Trees : sklearn Documentation
17. Ensemble Models : sklearn Documentation
18. Explanation of gradient tree boosting : Quora discussion
19. Difference between Gradient and XGBoosting : Stack Exchange
20. Chopra, Sumit et. al. Discovering the hidden structure of house prices with a non parametric latent manifold model.
21. Zhang, Pan, Shi et al. The Prediction of Booking Destination on Airbnb Dataset