House Price Prediction

Harshit Krishnakumar (2000114341), Manasa Makam (2000112843), Sushmita Sivaprasad (2000101429)

Abstract
Owing to the highly volatile nature of housing industry, there is a growing need to find robust and accurate methods to predict house prices. This paper presents different approaches to predict house prices using advanced regression techniques on the Ames housing data set provided in the Kaggle competition "House Prices: Advanced Regression Techniques". First we discuss different approaches in data pre-processing, perform feature selection using random forests and correlation matrices, and compare the results of different regression techniques on the clean data. Lastly, we reflect upon the impact of methods followed in the pre-processing and feature selection steps upon the results of the model.

Keywords
Linear Regression, Gradient Boosting, XGBoost, Ensemble Methods, Housing Sales Price Prediction

[1] *Computer Science, School of Informatics and Computing, Indiana University, Bloomington, IN, USA*

Contents

Introduction

Historically in the U.S., housing market is known to be a good indicator of the strength of the economy . When people sense a positive growth in the economy, they tend to be bullish towards buying better homes. Since house prices tend to grow in a positive economy, buying houses is seen as an investment towards the future. The housing industry in U.S. is booming at a rate of 6.2% a year, to a current total house value of \$27.5 trillion[1]. Owning a house is part of the "American Dream", and every year millions of people are looking to buy or upgrade their houses. This calls for better and accurate methods to evaluate house prices.

This report details different methods to describe the house prices in Ames, Iowa based on different amenities and features of the house. The data for this project was taken from the Kaggle competition "House Prices: Advanced Regression Techniques" and it has different variables related to features of the house, age, renovation year and location. Our goal is to indentify the most valuable features which would contribute towards predicting house prices. Choosing the right features plays a very crucial role in the accuracy of the model that we build. We start with different data cleaning / pre-processing techniques, feature selection approaches and look at different modeling techniques to get the best model for the given data. We evaluate the models based on multiple parameters and reflect upon the impact of pre-processing and feature selection steps on the actual model building process.

1. Background

The data provided has 80 variables with a total of 1460 observations for training dataset and a similar number of records in test data. Target variable here is the house prices, which is continuous and hence lends itself to linear regression. Out of the 80 input variables, there are 30 continuous and 50 categorical variables, out of which 12 are ordinal and the rest are nominal variables. We aim to represent the house price as a function of a subset of these 80 variables using Linear Regression techniques.

1.1 Linear Regression

Linear Regression is a statistical technique which attempts to predict a target variable based upon one or more independent variables. We try to fit a straight line through the data, called the regression line, which is used to predict the target variable. The linear regression line will be of the form

$$y = a_0 + a_1x_1 + a_2x_2 + a_3x_3 \dots a_nx_n + e \tag{1}$$

Where a_0 is the intercept, the sequence $a_1, a_2, a_3 \dots a_n$ are the regression coefficients of each individual variable. e is the error term which is also called residual. The intercept and the coefficients are estimated using the training data in the step called model fitting. Not all points from the training data will fall exactly on the regression line, in other words, since we try to fit a straight line through the data, there will be points which are off the line. The measure by which they are off is represented by the error term e.

Linear Regression needs to be used with the following requirements, in decreasing order of importance[2]:

1. **Continuous Variables:** The target variable is required to be a continuous numeric variable, and input variables need to be numeric as well. This does not mean that Linear Regression cannot handle categorical data in input variables, it has to be converted to numbers before passing it into the model
2. **Linearity:** There should be a linear relationship between each of the independent variables and target variable. This can be checked using scatter plots by visual examination. However, if this requirement is not met, linear regression will still model it assuming linearity between variables.
3. **Independence:** The input variables to the model need to be independent of each other. This can be checked using a correlation matrix. Detailed analyses can be performed using t- test and Chi Squared tests to check the independence of variables if required.
4. **Equal Variance:** This property is called homoscedasticity, where the variance of target variable across different ranges of the input variables are checked. The target variable needs to be uniformly distributed around the regression line, and the errors should be roughly equal along the regression line. This can also be checked using scatter plot between input and output variables.

In further sections, we show that our data satisfies each of these assumptions and that Linear Regression can be used in this scenario.

1.2 Ensemble Techniques

In real world, basic models like Linear Regression can only give limited prediction accuracy, due to a range of limitations. They might not be able to accommodate the entire variance of data, and there might be problems like over fitting. We now look at a range of methods called ”Ensemble Techniques”, which try to combine the results of multiple basic models built on training data, to give better prediction results. Ensemble methods perform best when the individual models are significantly different from each other. Different methods branch out from the common label ”Ensemble techniques”, depending on how the basic models are built and end results are combined[3].

Bagging is the group of techniques where a base learner is selected and a large number (n) of base models are built independently on small samples of training data, each selected using random sampling with replacement. Whenever a new data point comes in, we run all the n models on test data, and either take an average of the prediction (in case of continuous target variable) or take a vote (in case of categorical). Bagging helps to reduce over-fitting the training data set.

Boosting is similar to bagging, in the sense that we take a base learner algorithm and build multiple models on training data. The results of each model is influenced by the results of the previous models. In the end, all the n models are combined based on weighted averages. Boosting techniques are preferred when the output of a the base models are giving poor results and to reduce model bias.

Random Forests are a large number of decision trees, each built based upon a subset of data. It is different from Bagging techniques in the sense that, Random Forests also randomize factors considered. The results of each tree is combined using vote/average to give the final prediction. Random Forests are poor for continuous variables since they cannot predict beyond the range of target values in the training data. However random forests can be a good method for feature selection.

Ensemble methods are not limited to the three techniques described, but these are the most commonly used ones in machine learning applications. In our project, we use boosting techniques with linear regression as the base model. We now look in depth at different boosting techniques.

1.3 Adaboost

Adaboost was the first ever boosting technique implemented. Here, we take samples of training data to iteratively build multiple models, but instead of using random sampling, we assign weights for each data point. These weights represent the probability of the point getting selected in the sampling process. At the end of each round of model building, we update the weights for each data point. We give higher weights for the ”least represented” or ”difficult to fit” points in the current model, based on the residuals/errors for every point. In this process, we end up with n models, and we assign weights to each model based on its prediction accuracy, with higher weights assigned to the models with better performance on training data. The results of each of these models are then combined by taking weighted averages.

1.4 Gradient Boosting

There are three elements invoved in gradient boosting:

1. A base predictor
2. Choice of loss function to calculate residuals

3. An additive function to minimize the residuals of previous model

In every iteration, Gradient Boosting technique tries to fit the residuals from the previous iteration to a function, which is then weighted and added to the existing model. Residuals are updated based on the new model, and the next iteration aims at fitting the updated residuals.

2. Algorithm and Methodology

2.1 Data Pre Processing

This step deals with preparing the data to be fed into models. As a general rule of thumb, the data that goes into models should be clear of missing values, noise, outliers and any inconsistencies in the data should be resolved. Further to improve model accuracy, we eliminated variables which did not account for much variance in the target variable in the feature reduction step. As a final step, we converted ordinal variables to numbers, and categorical variables to dummy variables.

2.1.1 Missing Value Treatment

The data that we got from Kaggle did not have a large number of missing values. The variables which had missing values are given in table 1.

Table 1. Missing Value Treatment

Variable	Replaced with
LotFrontage	Mean value of its Zip
MasVnrArea	0
GarageYrBlt	2020
MasVnrType	”None”

We replaced LotFrontage nulls with the mean LotFrontage of all the houses in its Zip. We get the zip codes based on the attribute neighbourhood given in data for every house. MasVnrArea and MasVnrType talk about total area of wall covered with masonry veneer and the type of masonry. We assume that nulls in this data means that there is no masonry work done on the house. Similarly, we assume that a null in GarageYrBuilt indicates the absence of garage in the house, hence we replace them with a future year. In order to accommodate for any null values in new data that we might run through the model, we run loops to check for nulls in all the remaining columns, and replace them with the mean for categorical variables and the mode for categorical variables.

2.1.2 Outliers and Noise

Outliers can be checked by plotting box plots for continuous variables. We checked every variable for outliers, and found that the variables did not have any significant outliers which needed to be removed. The data around the third quantile was continuous, and there were no variables which had extreme values. Further there was no noise in the data, all the variables had values which were in line with the data description given.

2.1.3 Feature Selection

Feature Selection is an important step in the model building process. We select a subset of variables from the original set, which are most influential towards the target variable.

Mean Decrease Accuracy Using Random Forests

We used random forests as a model to evaluate the importance of variable. Random Forests is a cluster of decision trees, each built upon a subset of features. Every node in the individual decision trees is split based on a single attribute.

We used mean decrease accuracy method to measure the impact of each feature on the fit of the model. We iterate through every column and shuffle its values, and compare the model fit with the fit for un-shuffled values. If a variable is not very significant, the model fit would not change much when its values are shuffled. We score each column based on how much the fit of model (r^2 score) has changed.

Since the target variable in this case is a continuous variable, we used RandomForestRegressor method which comes as part of sklearn library in Python to implement this. The algorithm used is given below[4]:

```
Mean Decrease Accuracy:

Randomly split the input data into test and train
Fit Random Forest Model on the training data
Predict for test data and store the predictions in a separate list
Calculate $r^2$ score for the prediction and store in a variable
for Every column in input data do
    Randomly Shuffle test data for the selected column
    Predict target values for shuffled data using the model built
    Calculate $r^2$ score for the new predictions
    Compare the $r^2$ scores between original data and shuffled data
    $score = (actual\ r^2\ score - shuffled\ r^2\ score)/actual\ r^2\ score$
    Append the scores to a list along with the column name shuffled
end for
Print the sorted list based on scores, the higher the score, the more important it is for the predictions
```

Using the sorted scores obtained from the above step, we took the first 30 variables to progress with our next steps. We arrived at this number by trial and error on the actual model.

Correlation Matrix

Since we are using a Linear Regression base model in Gradient Boosting, which assumes independence between input variables, we need to ensure that there are no variables which are highly correlated. Pearson's Correlation gives the strength of linear relationship between two variables. We get the correlation matrix for all pairs of variables and filter out those with at least 0.7 correlation. We now remove the least significant variable of each pair based on the scores calculated using Mean Decrease Accuracy in the previous step.

2.1.4 Ordinal and Nominal Variables

As the final step of data pre-processing, we converted all the ordinal variables to numeric values, giving a score for each category. All the ordinal variables in our data had the same categories, hence we replaced them as shown in Table 2.

Table 2. Ordinal Value Treatment

Value	Replaced with
Excellent	5
Good	4
Typical	3
Fair	2
Poor	1

Nominal categorical variables were converted to indicator variables with 1 and 0 values using the python function get_dummies() in the library pandas.

2.1.5 Neighbourhood - zip code from Google Maps

We observed that Neighbourhood had close to 25 categories, which was the highest among other categorical variables. A lot of groups in a categorical variable performance problems due to overfitting. Hence we grouped the similar neighbourhoods into same category by adding the zip code of the neighbourhood. We got the zip codes of each neighbourhood using the Python library ”requests”. We passed a search request to Google maps by passing the neighbourhood concatenated with the Google maps url from Python. We were able to extract zip codes from the json file that Google returned, and we tagged neighbourhood to its respective zip codes. The Neighbourhood column was later deleted.

2.2 Model Building

We used Gradient Boosting with Linear Regression as the base model to predict the house prices based on the processed data. Gradient Boosting Regression in Python sklearn library builds an additive model in a forward stage-wise method. The algorithm for Gradient Boosting Regression is given below[5]:

```
GradientBoostingRegressor[6]
Input: training data set with independent numeric variables and dependent continuous variable $\{(X_i, Y)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations $M$.
Initialize model with a constant value:
$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$.
for m = 1 to M: do
    Compute pseudo-residuals:
    $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$    for $i = 1, \ldots, n$.
    Fit a base learner $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    Compute multiplier $\gamma_m$ by solving the following one-dimensional optimization problem:
    $\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$.
    Update the model:
    $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$.
end for
Output $F_M(x)$.
```

2.3 Cross Validation

We performed a k-fold cross validation on our model to check the consistency of its performance. k-fold cross validation is available as a package in sklearn of Python, and it works by splitting the training the data set into k separate partitions. It iterates k times, and at every iteration, it considers one of the k partitions as the test set and the rest of them as the training set to fit the model. It calculates the error rates in terms of least squared or fit in terms of r^2 score for every iteration and gives an average score for the entire model.

Building a model using Cross Validation technique helps to reduce over fitting the data, which is an inherent problem in Gradient Boosting.

3. Experiments and Results

3.1 Features Selected

We took the top 30 features sorted in a non increasing order, these features and their scores are given in the table 3.

3.2 Base Model - Linear Regression

We started our project by running a linear regression with lea on Overall Quality, Greater Living Area, Total Basement SF and Garage Cars variables. This was to get an initial idea of how the base learner performed. We did a 5 fold cross validation on this model to get a fair idea of how a base model performs. We could not go beyond 5 folds since we had less than 1500 records in the training data. The model resulted in a r^2 value of 0.67, which was not a great result for predictions.

3.3 Gradient Boosting

In order to improve the prediction accuracy, we used gradient boosting regressor with 5 fold cross validation. The base model was linear regression and the loss function that we used was huber loss. The parameters that we used for the model are:

n_estimators	It indicates the number of iterations of gradient boosting algorithm (50)
max_depth	Maximum depth of the gradient boosted trees (30)
learning_rate	Weight of each individual model which is added to the original function (0.5)
loss	Loss function of the base model (huber)

Using this technique our r^2 score improved to 0.84. This shows that the model fits training data better. We got the best

Table 3. Top 30 features from Mean Decrease Accuracy

Variable	Significance Score
OverallQual	0.4786
GrLivArea	0.1429
TotalBsmtSF	0.0283
BsmtFinSF1	0.0222
GarageCars	0.0193
1stFlrSF	0.0177
2ndFlrSF	0.0125
LotArea	0.0102
YearBuilt	0.0089
GarageArea	0.008
Zip	0.0076
TotRmsAbvGrd	0.0072
YearRemodAdd	0.0056
LotFrontage	0.0052
OverallCond	0.0051
BsmtQual	0.0049
GarageYrBlt	0.004
KitchenQual	0.0039
MasVnrArea	0.0035
WoodDeckSF	0.0034
FullBath	0.0032
BsmtExposure	0.0032
OpenPorchSF	0.0031
MoSold	0.0029
FireplaceQu	0.0029
BsmtUnfSF	0.0029
ExterQual	0.0023
BedroomAbvGr	0.0022
Fireplaces	0.0017

results by taking the top 30 variables from the feature selection step. We iterated over the number of variables selected and found the ideal number to be the top 30 variables ordered by the significance scores obtained from Mean Decrease Accuracy. Figure 1 shows the results of different number of variables vs r^2 score.

We see that compared to the basic model of linear regression, gradient boosting has improved the prediction accuracy by 25%. This model takes into consideration the top 30 features, out of which Overall Quality and Living Area are the ones with highest significance scores. This looks meaningful since we know that spacious and better quality homes are generally priced higher. A surprising insight that we got from the analysis is that the feature zip code is not in the top 5 most significant variables, which is contrary to our expectation that houses cost more in a better neighbourhood. This might be because the area we took into consideration is not too diverse and large to see a difference in neighbourhood quality.

4. Conclusions and Future Work

This project is an attempt to improve the predictions of a linear regression algorithm through gradient boosting. We also observed the importance of data pre-processing on the predictions. Selecting the right set of features will play a crucial role in the performance of the model. This is not a generic model, in the sense that we tuned the parameters of this model towards the data provided for Ames, Iowa.

As part of future work, we can make the code a generic one, which will find the best set of parameters for data from any other city. This can be implemented using Grid Search method in Python sklearn, which will loop through a range of parameter values to find the best set. We can improve our algorithm to replace missing values by using techniques like nearest neighbour or random forests to predict the missing values. We can experiment by using sophisticated techniques like XGBoost in sklearn and consider using other Ensemble methods like stacking and check the prediction accuracy for these models.

Acknowledgments

This project was submitted as part of the Data Mining course work (Fall 2016) at Indiana University Bloomington. We would like to extend our thanks and express our sincere gratitude to our advisor/prof Dr. Mehmet Dalkilic for motivating us to pursue this project. We would also like to thank Mr. Hasan Kurban and other AI's for providing us with resources and guiding us through the course.

Appendix

Hereby stating that all work herein is solely ours. We used the following packages in Python:

- sklearn version 0.18.1
- numpy version 1.11.2
- pandas version 0.19.1
- seaborn version 0.7.1
- requests 2.11.1

We executed this project on Asus laptop on a Windows 10 OS with python 2.7 version. The system configuration is Intel Core i7 processor with 16 GB ram and 64 bit operating system.

References

[1] http://fortune.com/2016/01/26/real-estate-global-economy/

[2] The Assumptions of the Linear Regression Model. Michael A. Poole and Patrick N. O'Farrell, Transactions of the Institute of British Geographers No. 52 (Mar., 1971).

[3] A working guide to boosted regression trees. J. Elith, J. R. Leathwick, T. Hastie, First published: 8 April 2008

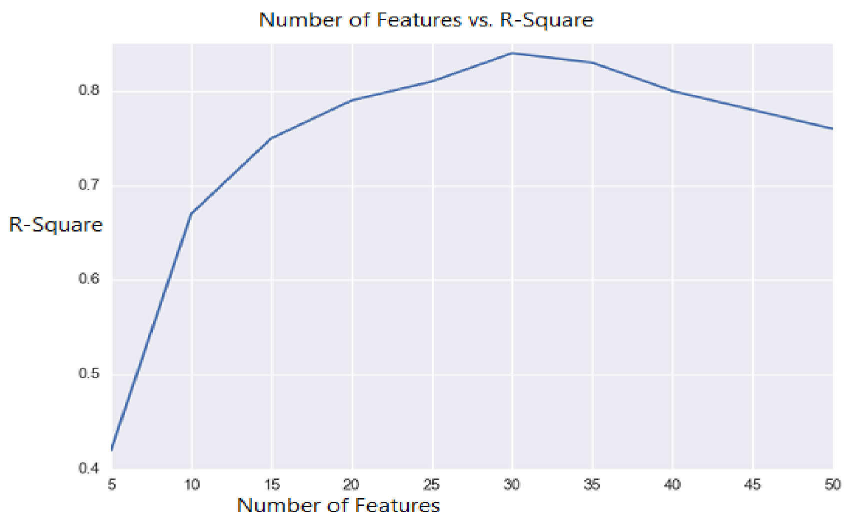

Figure 1. Feature Selection based on r^2 value

[4] http://blog.datadive.net/selecting-good-features -part-iii-random-forests/

[5] Greedy Function Approximation: A Gradient Boosting Machine. Jerome H. Friedman,The Annals of Statistics,Vol. 29, No. 5(Oct., 2001), pp. 1189-1232

[6] https://en.wikipedia.org/wiki/Gradient_boosting#Algorithm

Github Link to Source Code

`https://github.com/ManasaMakam/DM/blob/master/Housing_Problem.py`