Understanding Advanced Regression Techniques By Analyzing Housing Price Dataset.

OPIM 5503 Data Analytics Using R

Team Name: We R Ready

Team Members: Jiehao Pan, Santosh Mysore Vasudevan,

Siqi Wan, Yan Tan, Yanjun Yang

**Contents**

1. Introduction

2. What is Prediction with Trees

2.1 Random Forest

2.2 Gradient boosting

3. Dataset Description

4. Exploratory Data Analysis

5. Data Preparation

6. Predictive Modeling

6.1 Linear Regression Model

6.2 Random Forest Model

6.3 Gradient Boost Model

7. Model Comparison

8. Package Used

References

**1. Introduction**

In real life statistical modeling, we will usually be dealing with high-dimensional dataset with large numbers of predictor variables but relatively small sample size. Hence, there is a need to implement statistical techniques which select the most informative features out of a large set of predictor variables. The most commonly used technique to deal with a lot of predictors is the generalized linear modeling.

However, is generalized linear modeling the best option when dealing with the high dimensional dataset? Are there any other regression algorithms we can use to improve the model performance?

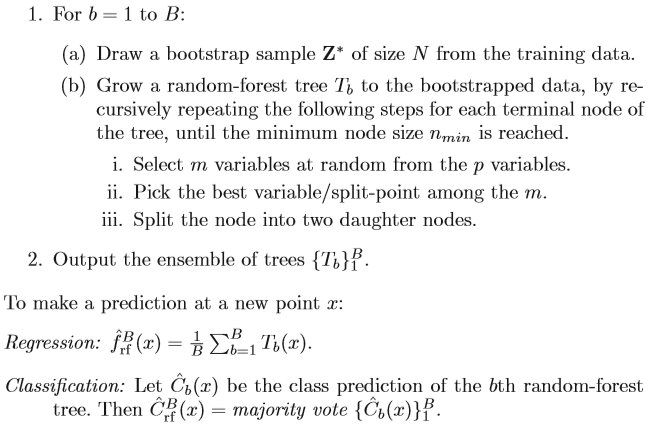
In this report, we will try to understand different advanced regression modeling techniques. We will use data from Kaggle competition: Housing Price Prediction to create regression models.

**2. What is Prediction with Trees**

2.1 Random Forest

Random Forest is an ensemble learning method for classification, regression and other tasks, that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of classes or mean prediction of the individual trees.

The algorithm of Random Forest is as follows:



The basic idea of random forest is very similar to bagging in the sense that it bootstrap samples. It takes a resample of our observed data and then it rebuilds classification or regression trees on each of those bootstrap samples.

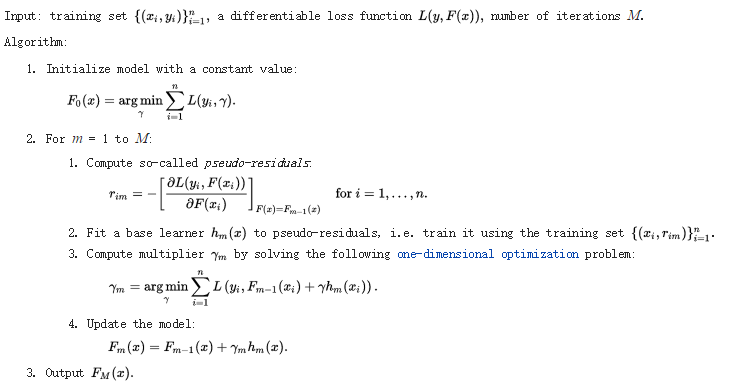
The one difference of random forest is that at each split, when it splits the data each time for classification or regression tree, it also bootstraps the variables. In other words, only a subset of the variables is considered at each potential split. This makes for a diverse set of potential trees that can be built.

In general, the idea of random forest is to grow a large number of trees. Then it either votes or averages those trees in order to get the prediction for a new outcome.

2.1 Gradient Boosting

Gradient boosting is a [machine learning](https://en.wikipedia.org/wiki/Machine_learning) technique for [regression](https://en.wikipedia.org/wiki/Regression_(machine_learning)) and [classification](https://en.wikipedia.org/wiki/Classification_(machine_learning)) problems, which produces a prediction model in the form of an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning) of weak prediction models, typically [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning). It builds the model in a stage-wise fashion like other [boosting](https://en.wikipedia.org/wiki/Boosting_(meta-algorithm)) methods do, and it generalizes them by allowing optimization of an arbitrary [differentiable](https://en.wikipedia.org/wiki/Differentiable_function) [loss function](https://en.wikipedia.org/wiki/Loss_function).

The algorithm of Gradient Boosting is as follows:



The basic idea of gradient boosting is to take k classifiers. Then create a classifier that combines these classification functions together, and weights them together. The goal is to minimize error on the training set.

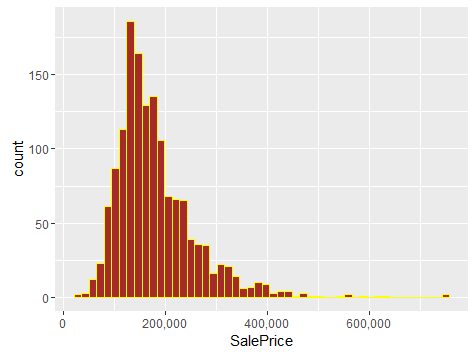
**3. Dataset Description**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| SalePrice | Continuous | The property's sale price in dollars. This is the target variable that we are trying to predict. |
| MSSubClass | Continuous | The building class |
| MSZoning | Categorical | The general zoning classification |
| LotFrontage | Continuous | Linear feet of street connected to property |
| LotArea | Continuous | Lot size in square feet |
| Street | Categorical | Type of road access |
| Alley | Categorical | Type of alley access |
| LotShape | Categorical | General shape of property |
| LandContour | Categorical | Flatness of the property |
| Utilities | Categorical | Type of utilities available |
| LotConfig | Categorical | Lot configuration |
| LandSlope | Categorical | Slope of property |
| Neighborhood | Categorical | Physical locations within Ames city limits |
| Condition1 | Categorical | Proximity to main road or railroad |
| Condition2 | Categorical | Proximity to main road or railroad (if a second is present) |
| BldgType | Categorical | Type of dwelling |
| HousStyle | Categorical | Style of dwelling |
| OverallQual | Continuous | Overall material and finish quality |
| OverallCond | Continuous | Overall condition rating |
| YearBuilt: | Continuous | Original construction date |
| YearRemodAdd: | Continuous | Remodel date |
| RoofStyle | Categorical | Type of roof |
| RoofMatl | Categorical | Roof material |
| Exterior1st: | Categorical | Exterior covering on house |
| Exterior2nd | Categorical | Exterior covering on house (if more than one material) |
| MasVnrType | Categorical | Masonry veneer type |
| MasVnrArea | Continuous | Masonry veneer area in square feet |
| ExterQual | Categorical | Exterior material quality |
| ExterCond | Categorical | Present condition of the material on the exterior |
| Foundation | Categorical | Type of foundation |
| BsmtQual | Categorical | Height of the basement |
| BsmtCond | Categorical | General condition of the basement |
| BsmtExposure | Categorical | Walkout or garden level basement walls |
| BsmtFinType1 | Categorical | Quality of basement finished area |
| BsmtFinSF1 | Continuous | Type 1 finished square feet |
| BsmtFinType2 | Categorical | Quality of second finished area (if present) |
| BsmtFinSF2 | Continuous | Type 2 finished square feet |
| BsmtUnfSF: | Continuous | Unfinished square feet of basement area |
| TotalBsmtSF | Continuous | Total square feet of basement area |
| Heating | Categorical | Type of heating |
| HeatingQC | Categorical | Heating quality and condition |
| CentralAir | Categorical | Central air conditioning |
| Electrical | Categorical | Electrical system |
| 1stFlrSF | Continuous | First Floor square feet |
| 2ndFlrSF | Continuous | Second floor square feet |
| LowQualFinSF | Continuous | Low quality finished square feet (all floors) |
| GrLivArea | Continuous | Above grade (ground) living area square feet |
| BsmtFullBath | Continuous | Basement full bathrooms |
| BsmtHalfBath | Continuous | Basement half bathrooms |
| FullBath | Continuous | Full bathrooms above grade |
| HalfBath | Continuous | Half baths above grade |
| BedroomAbvGr | Continuous | Number of bedrooms above basement level |
| Kitchen | Continuous | Number of kitchens |
| KitchenQual | Categorical | Kitchen quality |
| TotRmsAbvGrd | Continuous | Total rooms above grade (does not include bathrooms) |
| Functional | Categorical | Home functionality rating |
| Fireplaces | Continuous | Number of fireplaces |
| FireplaceQu | Categorical | Fireplace quality |
| GarageType | Categorical | Garage location |
| GarageYrBlt | Continuous | Year garage was built |
| GarageFinish | Categorical | Interior finish of the garage |
| GarageCars | Continuous | Size of garage in car capacity |
| GarageArea | Continuous | Size of garage in square feet |
| GarageQual | Categorical | Garage quality |
| GarageCond | Categorical | Garage condition |
| PavedDrive | Categorical | Paved driveway |
| WoodDeckSF | Continuous | Wood deck area in square feet |
| OpenPorchSF | Continuous | Open porch area in square feet |
| EnclosedPorch | Continuous | Enclosed porch area in square feet |
| 3SsnPorch | Continuous | Three season porch area in square feet |
| ScreenPorch | Continuous | Screen porch area in square feet |
| PoolArea | Continuous | Pool area in square feet |
| PoolQC | Categorical | Pool quality |
| Fence | Categorical | Fence quality |
| MiscFeature | Categorical | Miscellaneous feature not covered in other categories |
| MiscVal | Continuous | $Value of miscellaneous feature |
| MoSold | Continuous | Month Sold |
| YrSold | Continuous | Year Sold |
| SaleType | Categorical | Type of sale |
| SaleCondition | Categorical | Condition of sale |

**4.Exploring Data Analysis**

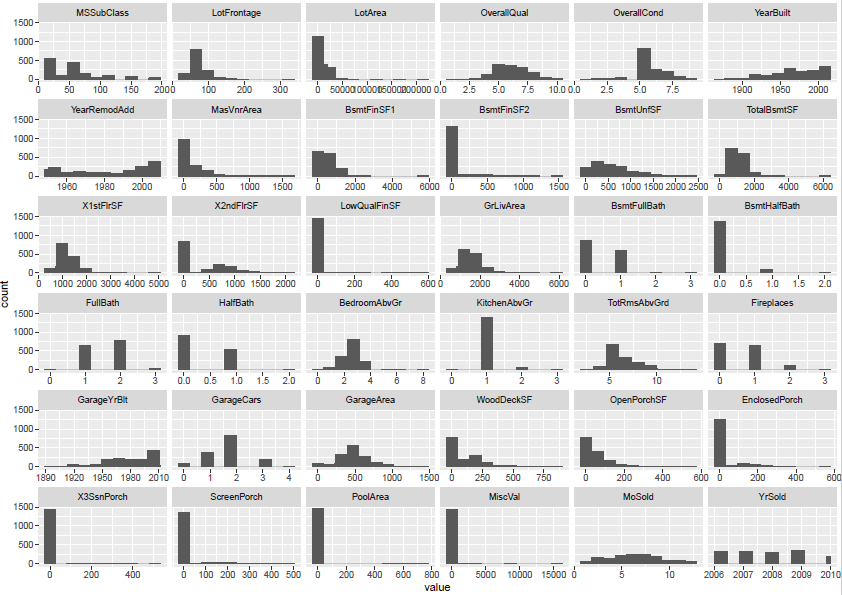
**4.1**

The target variable here is the sales price of the house. The below histogram shows how the sales price of the house is distributed in the train data set.



**4.2 Distribution of important features.**

There are a lot of important features (variables) which contribute to the sales price of the house. They are as follows:



**5. Data Preparation**

Our data preparation includes two steps: Missing values handling, and split raw data.

5.1 Missing Values handling

We wrote a function to calculate missing values for all the columns. The results show missing values exists in both numeric variables and categorical variables. The tables indicatesmissing values for train and test dataset.

|  |  |  |
| --- | --- | --- |
| Table of Missing Values-train | | |
|  | Variable Names | No. of Missing |
| 1 | LotFrontage | 259 |
| 2 | Alley | 1369 |
| 3 | MasVnrType | 8 |
| 4 | MasVnrArea | 8 |
| 5 | BsmtQual | 37 |
| 6 | BsmtCond | 37 |
| 7 | BsmtExposure | 38 |
| 8 | BsmtFinType1 | 37 |
| 9 | BsmtFinType2 | 38 |
| 10 | Electrical | 1 |
| 11 | FireplaceQu | 690 |
| 12 | GarageType | 81 |
| 13 | GarageYrBlt | 81 |
| 14 | GarageFinish | 81 |
| 15 | GarageQual | 81 |
| 16 | GarageCond | 81 |
| 17 | PoolQC | 1453 |
| 18 | Fence | 1179 |
| 19 | MiscFeature | 1406 |

|  |  |  |
| --- | --- | --- |
| Table of Missing Values-test | | |
|  | Variable Names | No. of Missing |
| 1 | MSZoning | 4 |
| 2 | LotFrontage | 227 |
| 3 | Alley | 1352 |
| 4 | Utilities | 2 |
| 5 | Exterior1st | 1 |
| 6 | Exterior2nd | 1 |
| 7 | MasVnrType | 16 |
| 8 | MasVnrArea | 15 |
| 9 | BsmtQual | 44 |
| 10 | BsmtCond | 45 |
| 11 | BsmtExposure | 44 |
| 12 | BsmtFinType1 | 42 |
| 13 | BsmtFinSF1 | 1 |
| 14 | BsmtFinType2 | 42 |
| 15 | BsmtFinSF2 | 1 |
| 16 | BsmtUnfSF | 1 |
| 17 | TotalBsmtSF | 1 |
| 18 | BsmtFullBath | 2 |
| 19 | BsmtHalfBath | 2 |
| 20 | KitchenQual | 1 |
| 21 | Functional | 2 |
| 22 | FireplaceQu | 730 |
| 23 | GarageType | 76 |
| 24 | GarageYrBlt | 78 |
| 25 | GarageFinish | 78 |
| 26 | GarageCars | 1 |
| 27 | GarageArea | 1 |
| 28 | GarageQual | 78 |
| 29 | GarageCond | 78 |
| 30 | PoolQC | 1456 |
| 31 | Fence | 1169 |
| 32 | MiscFeature | 1408 |
| 33 | SaleType | 1 |

After checking the raw data to understand the missing values, we went back to the raw data and found the values are missing for different kinds of reasons. Hence we decided to treat the missing values in different ways as follows.

(1) Recode categorical variable missing values into No category

Some variables displayed as “NA” which means missing value according to R. But actually, they have significant meanings of “no”. For example, the variable “GarageQual” indicates the quality of garage. In detail, “Ex” denotes “excellent”, “Gd” means “good” and so on. “NA” here just shows “No garage”, and not that this value is missing. So , we recoded these kinds of variables to make some values meaningful. There were in total 14 variables which we imputed missing values in this way for both train and test dataset: Alley, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, FireplaceQu, GarageType, GarageFinish,GarageQual, GarageCond, PoolQC, Fence, MiscFeature.

(2) Imputation based on logical rules

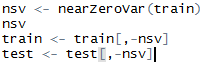
After recoding the missing values into the “No” category, we fond there still missing values in some of the variables. However, the number of missing values are small, typically less than 10, so we checked the raw data and inferred a logical reasonable value as follows:

Categorical variables (eg: Electrical and MasVnrType): We found the number of records for each category level, picked the level with most records and then recoded the missing value into that level.

Numeric or Integer variables: We checked the raw data to see the features (basement type, garage type and etc) of a certain observation and gave a reasonable value for the missing value. For example, the observation with “BsmtFullBath” missing shows no basement, so the missing values in“BsmtFullBath” should be 0.

(3) Deleting Near Zero Variance Variables

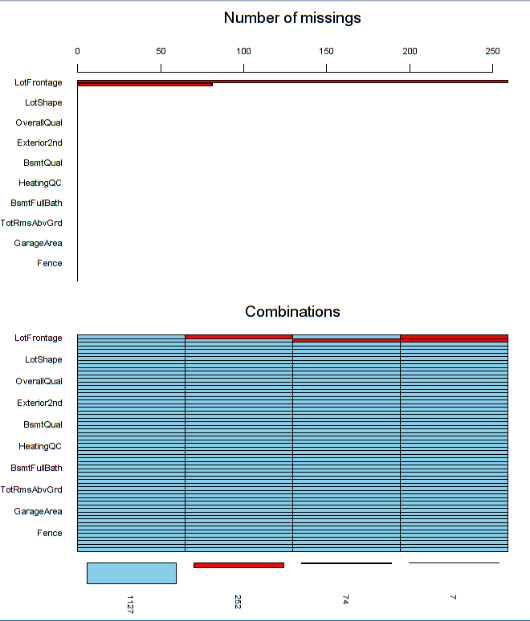
We used function “nearZeroVar” (caret package) to find the variables with low variation. We found near 20 low variance variables and we used the remaining 60 variables for our further prediction models.



(4) Impute and Standardize Variable

After recoding and deleting zero variance variables, the remaining missing values and their plots are below:

|  |  |
| --- | --- |
| Variable | Count |
| LotFrontage | 259 |
| GarageYrBlt | 81 |



Variable LotFrontage:

The result of counting missing values shows that the numeric variable “LotFrontage” has 259 missing values which is a big proportion of the total 1460 observations. We used the (KNN) K Nearest Neighbor transform to impute these missing values. The neighbors are taken from a set of objects for which the object property value is known. We installed package **RANN** and implemented the following:

preObj <- preProcess(train,method="knnImpute")

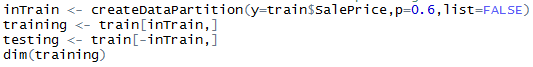
train$LotFrontage <- predict(preObj,train)$LotFrontage

Year-Related Variables:

It is obvious that the age of house will affect housing price. In our case, the indicator of house age is the year when the houses were built. So we transformed the year into age of the house by a simple subtraction “2016-YearBuilt” (Also the same case for variable “YearRemodAdd” and “GarageYrBlt”)

5.2 Raw Data Splitting

In order to test the performance of our model, we separated the raw train dataset into 60% training and 40% testing using the following function in R:



**6 Predictive Modeling**

The models we used for prediction are Linear Regression Model, Random Forest Model and Gradient Boost Model. We created linear regression model as a baseline, then we created random forest and gradient boost model to test if prediction model with trees will perform better than linear regression model.

6.1 Linear Regression Model

# (1) Principal component analysis:

# To deduct collinearity, we firstly conducted PCA procedure as follows:

preProc <- preProcess(training[,-60],method="pca",pcacomp=2)

trainingLR <- predict(preProc,training)

testingLR <- predict(preProc,testing)

As a result, we successfully reduce 60 predictors to 52 predictors.

(3) Model:

We created linear regression model using function ‘train’ from package“caret”.

lr <- train(SalePrice~ ., data=trainingLR, method="lm")

lrResult <- lr$finalModel

We deployed model to testing dataset and generated prediction value using function ’predict’ from package ‘caret’.

lrPred <- predict(lr,testingLR)

predLR <- data.frame(testingLR,lrPred)

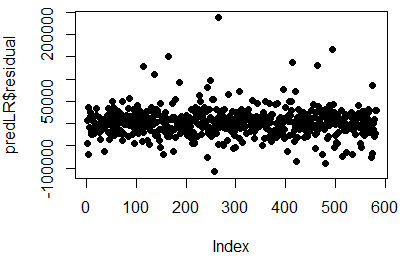
The residual plot and RMSE was coded as follows:

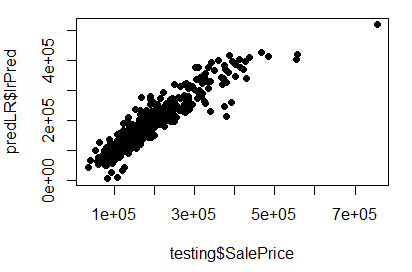
predLR$residual <- predLR$SalePrice-predLR$lrPred

plot(predLR$residual,pch=19)

rmse(log(lrPred),log(testingLR$SalePrice))

plot(testing$SalePrice,predLR$lrPred,pch=19)



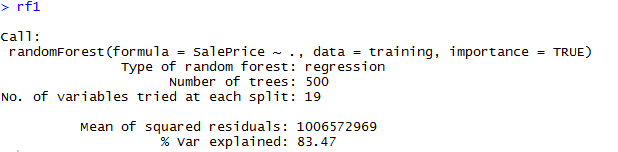


The residuals are randomly distributed around 0 as desired.

6.2 Random Forest Model

We used function randomForest from package ‘randomForest’ to create random forest model.

rf1 <- randomForest(SalePrice~ ., training, importance=TRUE)



We deployed model to testing dataset and generated prediction value using function ’predict’ from package ‘caret’.

pred <- predict(rf1, newdata=testing)

pred <- data.frame(testing,pred)

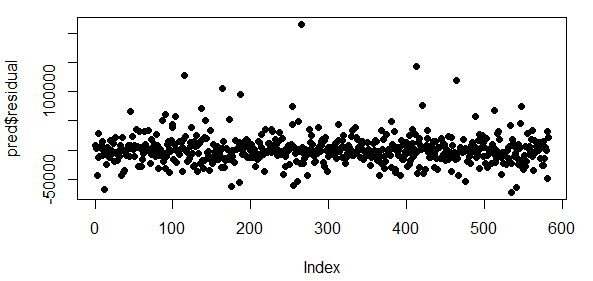
The residual plot and RMSE was coded as follows:

pred$residual <- pred$SalePrice-pred$pred

plot(pred$residual,pch=19)

rmse(log(pred$pred),log(pred$SalePrice))

plot(x=testing$SalePrice,pred$pred)

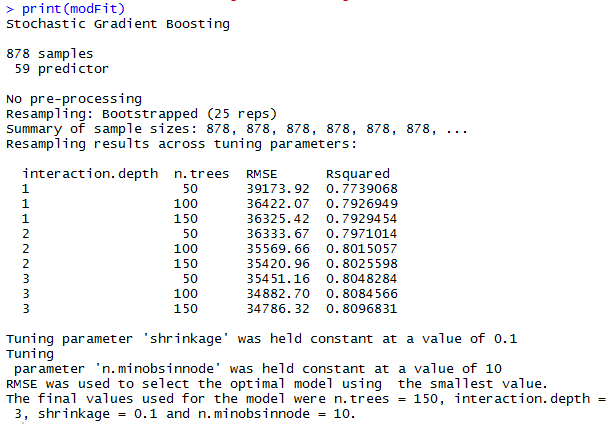


The residuals are randomly distributed around 0 as desired.

6.3 Gradient Boost Model

We used function ‘train’ from package ‘caret’ to create random forest model.

modFit <- train(SalePrice~.,method="gbm",data=training,verbose=FALSE)



We deployed model to testing dataset and generate prediction value using function ’predict’ from package ‘caret’.

pred2 <- predict(modFit,testing)

pred2 <- data.frame(testing,pred2)

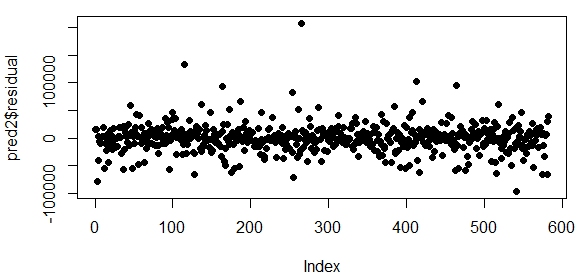
The residual plot and RMSE was coded as follows:

pred$residual <- pred$SalePrice-pred$pred

plot(pred$residual,pch=19)

rmse(log(pred$pred),log(pred$SalePrice))

plot(x=testing$SalePrice,pred$pred)



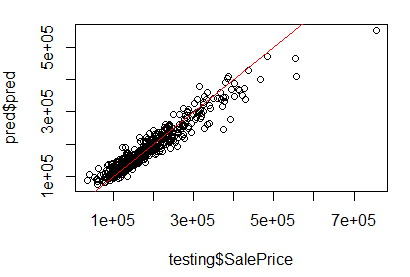
The residuals are randomly distributed around 0 as desired.

**7. Model Comparison**

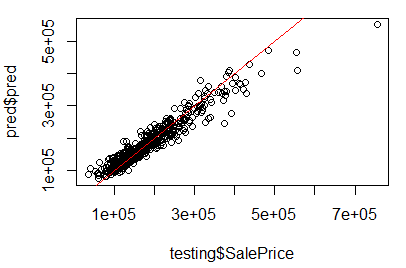
7.1 Model Comparison

The following plots are Prediction value vs Real value for testing dataset

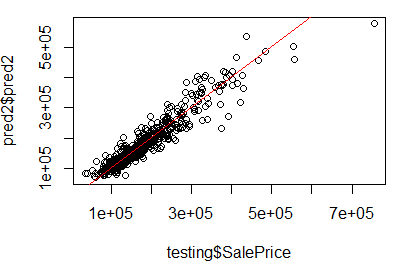
Linear Regression Model:



Random Forest Model:



Gradient Boost Model:



The following table highlights the RMSE(root-mean-square error) obtained from the different models described above:



From the plot of prediction value and real value, we can conclude prediction with trees performs better than linear regression. Hence we recommend the use of prediction with trees algorithm to predict housing price.

**8. R Packages Used**

The package Caret is the main package we use for our project.

**Package Caret:** Caret is a handy package for training and plotting classiﬁcation and regression models. We mainly used the following functions in caret in our project:

**createDataPartition**: Data splitting function to create training and testing dataset

**nearZeroVar**: Identification of near zero variance predictors

**preProcess:** Pre-processing transformation (centering, scaling etc.) can be estimated from the training data and applied to any data set with the same variables.

**train:** We use this function to create regression model. This function sets up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure.

**predict:** Predictions from the results of various model fitting functions.

**Other Packages used:**

1.Package **ggplot2**

2.Package **RANN** (for KNN imputation)

3.Package **Metrics** (for calculating rmse)

4.Package **randomForest**(for Random Forest model)

5.Package **gbm**

**Reference**

[1] Wikipedia, Random Forest

[2] Wikipedia, Gradient Boost

[3] Fortran original by Leo Breiman and Adele Cutler, R port by Andy Liaw and Matthew Wiener(October 7, 2015), Package‘randomForest’.

[4] Greg Ridgeway <gregridgeway@gmail.com> with contributions from others(March 11, 2015),Package‘gbm’.

[5] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, and Can Candan(August 5, 2016), Package‘caret’.

[6] The Elements of Statistical Learning Data Mining,Inference,and Prediction, Second Edition, Springer Series in Statistics

[7] Coursera: Practical Machine Learning by Johns Hopkins University

[8] Benjamin Hofner, Andreas Mayr, Nikolay Robinzonov and Mattthias Schmid (2014), Model-based Boosting in R - A Hands-on Tutorial Using the R Package mboost. Computational Statistics, 29:3-35.