House Price Prediction

Members: Anthony and teammates

Motivation

House price is worrying most young people who expect to own a house.

With the rapid development of computers and promising information technology, it is increasingly likely to understand different factors on house price through the analysis of house prices' data. The analysis and prediction of housing data can help people buy a more desirable house with less money. Real estate developers could also find more popular house types from the data and formulate more reasonable sales strategies. Therefore, we hold that the data analysis of house prices is interesting and worthwhile.

Data analyses

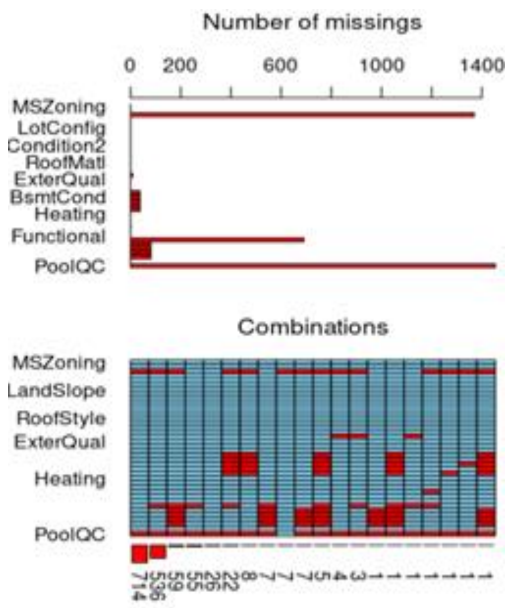
For data analysis, we plan to use it to figure out the importance of features on Sale Price, and try to analyze the properties of these features.

Plan:

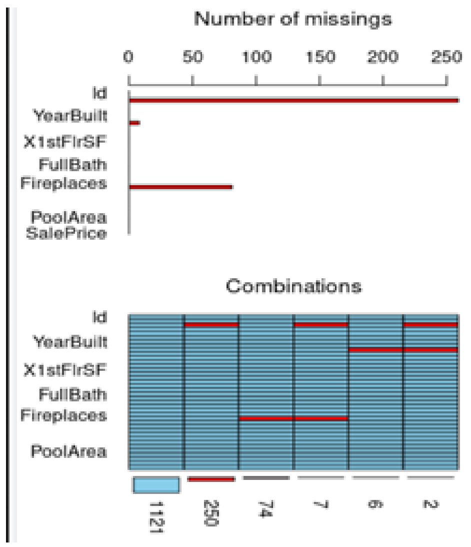
This part is allocated to two of our group numbers, one to analyze these data at a macro level, the other one to figure out whether some data will implicitly inflect sale-price.

Part 1: In this part, we first use function aggr() to visualize the number of "NA" in each feature, then we found that for integer columns NA it contains are less than character columns.

Left side



are

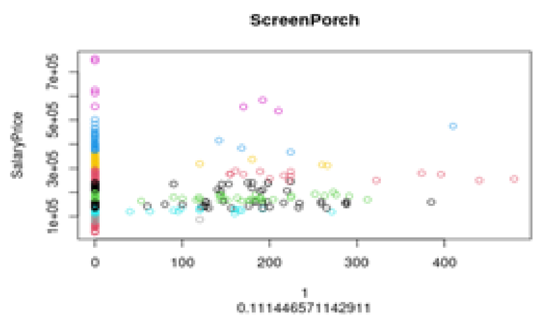


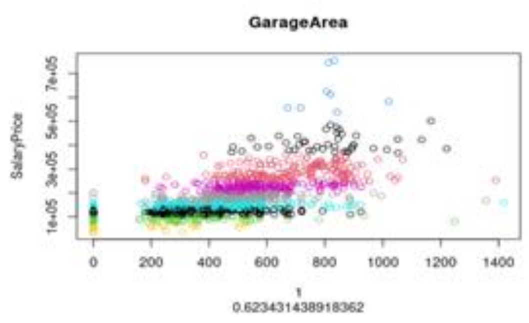
character columns, right side are integer columns.

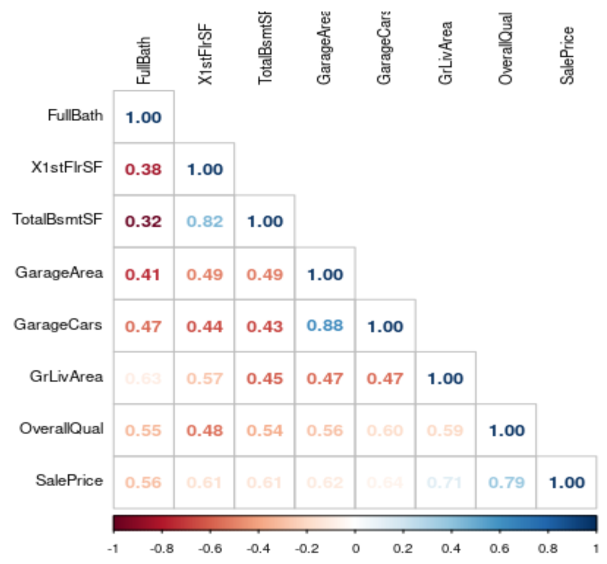
If we don’t transfer these NA into other acceptable data, they would bring many problems for us. So we transfer NA in character columns to “Not acceptable”, transfer NA in integer to data estimated by function mice().

At the beginning, we decided to transfer NA in integer into 0 or the average number of this column. But later when we use plot() and cor() to draw out their relationship, we found there isn't one correlation larger than 0.5 in all 38 integer columns. **After reference to others' solutions, we chose to use mice() to fix NA. mice() function could create multiple imputations for multivariate missing data. And the method is based on Fully Conditional Specification, where each incomplete variable is imputed by a separate model.** This method could ensure data it generated won't affect or be influenced by other irrelevant factors, which could prevent results from the overdue slope.

As mentioned before, we use cor() to evaluate the relationship between sale price and other integer features. Then we first use plot then use corplot to visualize its relevance.





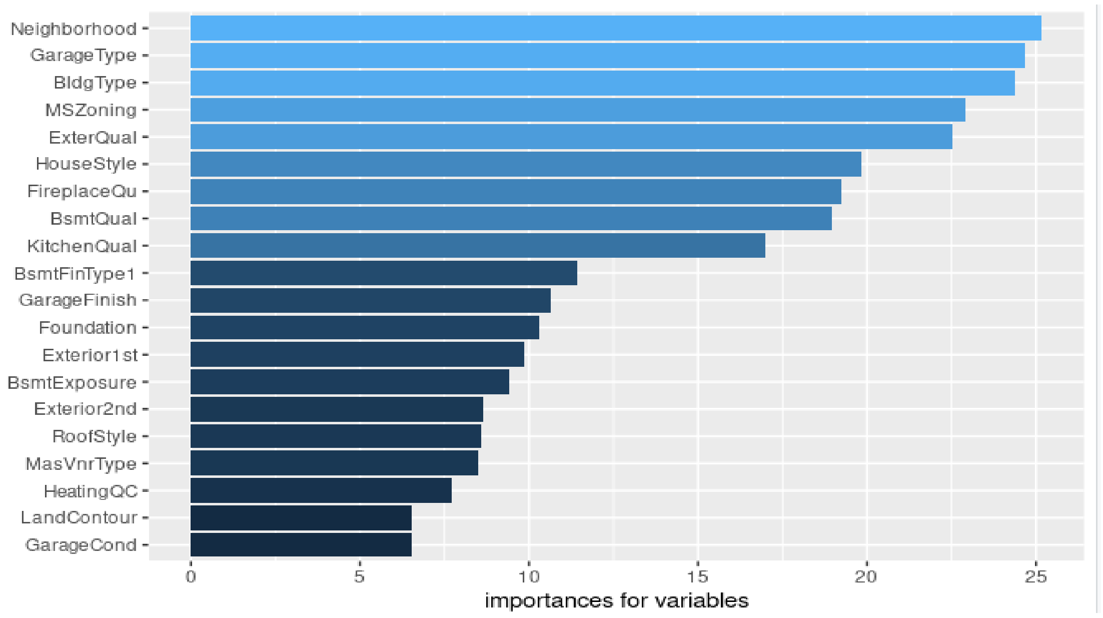


	FullBath	X1stFlrSF	TotalBsmtSF	GarageArea	GarageCars	GrLivArea	OverallQual	SalePrice
FullBath	1.00							
X1stFlrSF	0.38	1.00						
TotalBsmtSF	0.32	0.82	1.00					
GarageArea	0.41	0.49	0.49	1.00				
GarageCars	0.47	0.44	0.43	0.88	1.00			
GrLivArea	0.63	0.57	0.45	0.47	0.47	1.00		
OverallQual	0.55	0.48	0.54	0.56	0.60	0.59	1.00	
SalePrice	0.56	0.61	0.61	0.62	0.64	0.71	0.79	1.00

Result shows us that in **integer features** only two features' covariance with sale price are above 0.7, which are OverallQual(0.79) and GrLivArea(0.71),

For character columns, we tentatively transfer all NA into "Not acceptable", and we use random-forest to estimate their importance to sale price, in addition to estimate all features.

In the first time, we set ntree as 200, but after test results were quite unstable, the picture shows the names of the highest three features are always changing, which is out of our expectation.. Then we change ntree as 300, result shows more normal and we know of the importance of different features.



Left side is the first 20 characters' importance right is the fist 20 features' importance of all

Hence, we conclude that features with high relevant relationship with sale-price are:

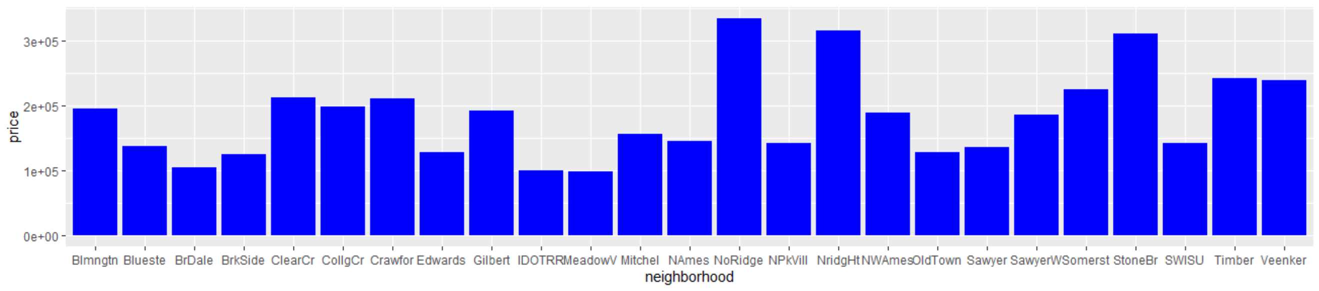
OverallQual, GrLivArea, TotalBsmtSF, BsntFinSF1, also may contain Neighborhood, ExterQual.

Data Processing

The relation between SalePrice and some important factors

(1) Neighborhood

In America, community or neighborhood influences the housing price greatly. For example, the crime rate is relatively high in some communities, the housing price there is very likely to be lower. **Hence, we divide the neighborhood into three levels, based on the price.**



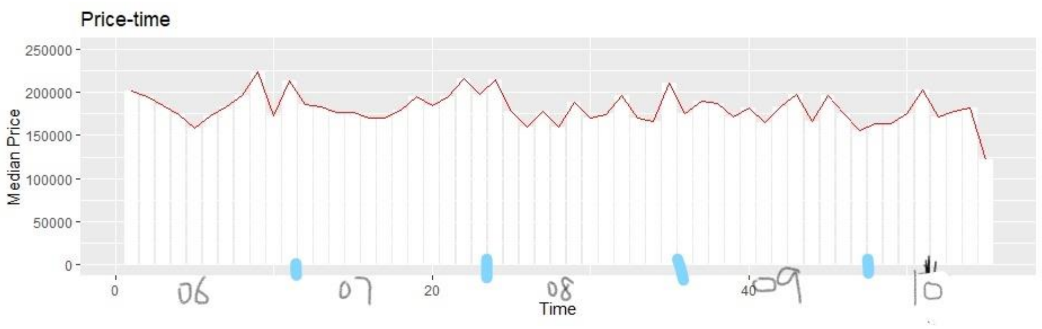
We can see in there sale price higher in NoRidge and NridgHt than BrDale and MedowV. And after searching NoRidge and NridgHt are in a safer area, we also consider about the visualization of characters by random forest and found neighborhood could greatly influence sale price.

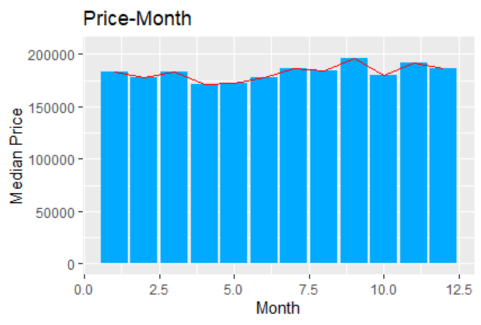
To prepare data: Divide the neighborhoods into three groups.

(2) Time (Year, Month)

Initially, we thought that the financial crisis occurred in 2006-2010, so the impact of the year on housing prices may be significant. Also, we tentatively explore the correlation between month and price. However, after analysis and comparison on the result, we concluded that the year difference has limited effect on housing prices. Also, we observe that the influence of the month is not significant as well. Thus in the end, we do not take Yrsold and Mosold into account.

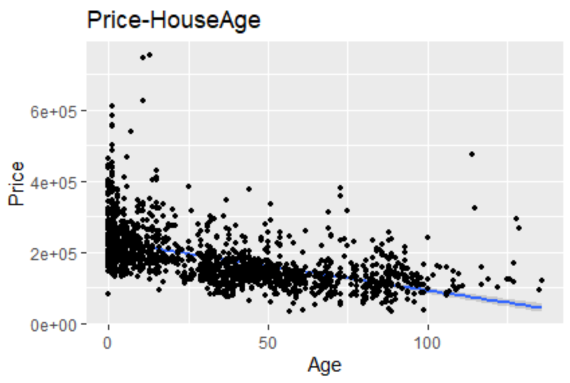
Not to Consider the time series since it has limited influence.

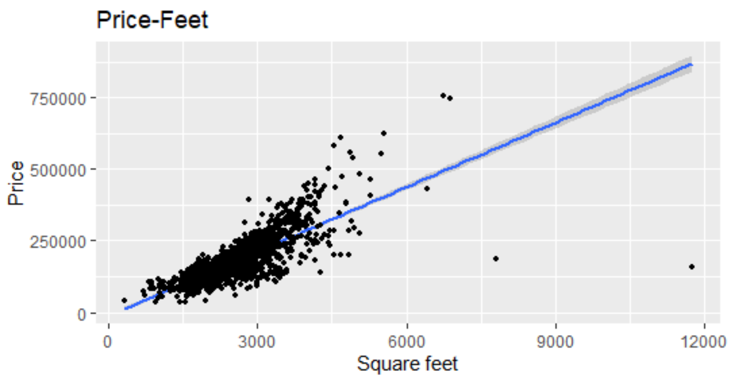




(3) Time (HouseAge)

The HouseAge is a significant factor in housing price. **By YrSold - YearBuilt**, we can get the relationship between Price and HouseAge. The negative correlation is significant.
To prepare data: YearBuilt=YrSold-YearBuilt





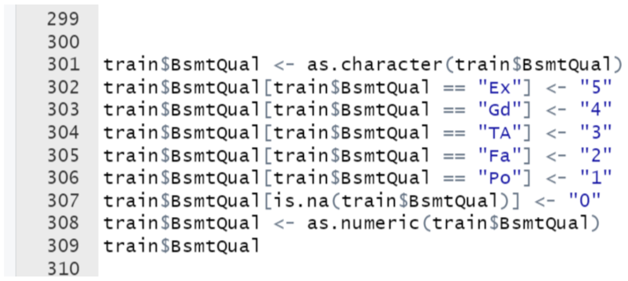
(4) Square

It should be noted that the feet of area is composed of GrLivArea and TotalBsmtSF. In such case, we **add them together** to see the relationship between square and sailing price. The result shows that the correlation relationship is significant. Moreover, we observe some outliers, which should be deleted since they may negatively influence the relationship.

To prepare data:
(1) Add TotalBsmtSF to GrLivArea in train and test
(2) Delete TotalBsmtSF column in train and test
(3) Delete some outliers in train

(5) Other categorical variables

Based on our current knowledge on **XGBoost,** we need to convert each character into numeric types. It's laboursome but worthwhile. The reason why we do not use factor( ) is that the factor variables could not be used in **XGBoost.**



```r


train$BsmtQual <- as.character(train$BsmtQual)
train$BsmtQual[train$BsmtQual == "Ex"] <- "5"
train$BsmtQual[train$BsmtQual == "Gd"] <- "4"
train$BsmtQual[train$BsmtQual == "TA"] <- "3"
train$BsmtQual[train$BsmtQual == "Fa"] <- "2"
train$BsmtQual[train$BsmtQual == "Po"] <- "1"
train$BsmtQual[is.na(train$BsmtQual)] <- "0"
train$BsmtQual <- as.numeric(train$BsmtQual)
train$BsmtQual

```

Based on our current knowledge, it's not efficient to analyze all the variables provided. Thus, we need to select some significant variables, process the data properly and base on them to predict.

Description

Before establishing the model, we considered a variety of models, such as SVM, lm, xgblinear, random forest and so on, and finally we chose the XGBoost (Extreme Gradient Boosting) model and SVM model. This is notes of what we have learned.

SVM:

The SVM model maps the sample space to a high-dimensional or even infinite-dimensional feature space (Hilbert space) through a nonlinear mapping p, so that the non-linear separable problem in the original sample space is transformed into a linearly separable problem in the feature space.

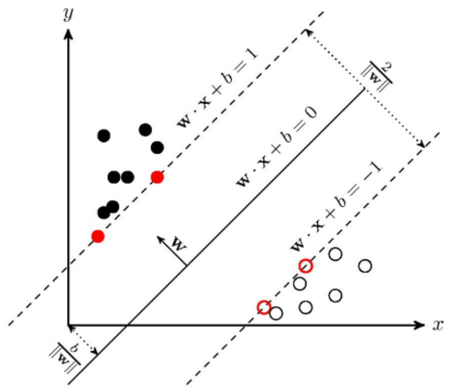
It has two main point:

①. It analyzes the linearly separable case. For the linearly inseparable case, the linear inseparable sample of the low-dimensional input space is converted into a high-dimensional feature space by using a nonlinear mapping algorithm to make it linearly separable, thereby making the high-dimensional feature space It is possible to linearly analyze the non-linear characteristics of the sample by using linear algorithms;

②.It constructs the optimal segmentation hyperplane in the feature space based on the structural risk minimization theory, so that the learner is globally optimized, and the expected risk in the entire sample space satisfies a certain upper bound with a certain probability.

Fundamental:

The basic idea of SVM learning is to solve the separation hyperplane that can correctly divide the training data set and have the largest geometric interval. As shown in below picture,



The Separating hyperplane is $\boldsymbol{w} \cdot x + b = 0$.For a linearly separable data set, such hyperplanes are indefinite, but the separating hyperplane with the largest geometric interval is unique.

Geometric interval: $\gamma_i = y_i \left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \cdot \boldsymbol{x}_i + \frac{b}{\|\boldsymbol{w}\|} \right)$

The minimum value of the geometric interval of all sample points in the hyperplane is: $\gamma = \min_{i=1,2\ldots,N} \gamma_i$

Linear support vector machine learning algorithm:

①input: Training data set: $T = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\} \boldsymbol{x}_i \in \mathbb{R}^n$
$y_i \in \{+1, -1\}, i = 1, 2, \ldots N$

②output: Separate hyperplane and classification decision function:

Ⅰ. Choose penalty parameters and construct and solve convex quadratic programming problems: $\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) - \sum_{i=1}^{N} \alpha_i$ then get the optimal solution:
$\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_N^*)^T$

Ⅱ. Calculate: $\boldsymbol{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \boldsymbol{x}_i$ $\quad b^* = y_j - \sum_{i=1}^{N} \alpha_i^* y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$

Ⅲ.Find the separating hyperplane and classification decision function:

$\boldsymbol{w}^* \cdot \boldsymbol{x} + b^* = 0$ $f(\boldsymbol{x}) = sign(\boldsymbol{w}^* \cdot \boldsymbol{x} + b^*)$

XGBoost:

The base learner in XGBoost can be either CART (gbtree) or gblinear. It has the following five advantages:

1. Can **prevent overfitting**
2. XGBoost not only uses the first-order derivative, but also uses the second-order derivative, **the loss is more accurate, and the loss can be customized.**
3. Parallel optimization of XGBoost, XGBoost's parallel is based on feature granularity.
4. Considering that the training data is sparse, the default direction of the branch can be specified for missing values or specified values, which can greatly improve the efficiency of the algorithm.
5. Support column sampling, which can not only reduce overfitting, but also reduce calculations.

Implementation

In establishing the model, we considered a variety of models, such as SVM, lm, xgblinear, random forest and so on, and finally we chose the XGBoost (Extreme Gradient Boosting) model and Lasso regression model.

Detailed analysis for XGBoost:

1. GB uses the first derivative of Loss Function to f(x) to calculate the pseudo-residual for learning to generate fm(x), xgboost not only uses the first derivative, but also the second derivative.

The loss of the t times:

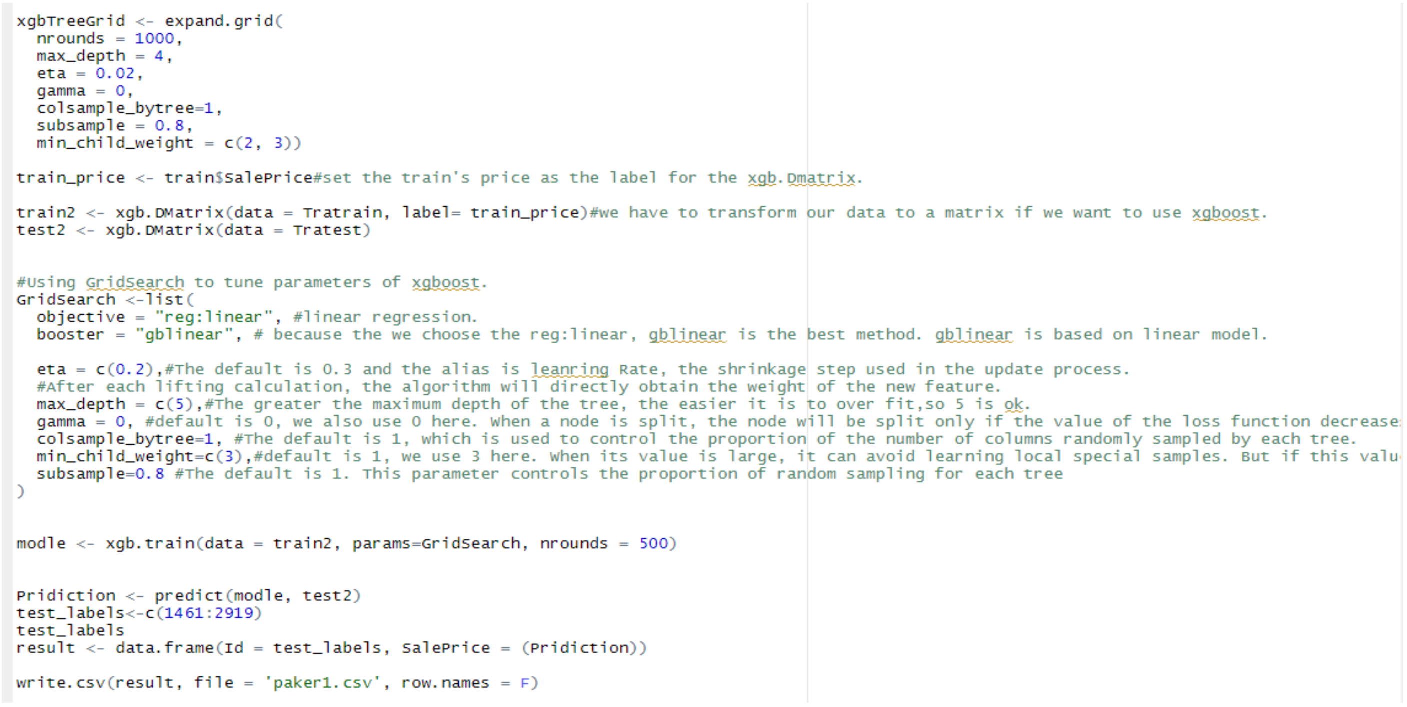
Do a second-order Taylor expansion of the above formula: g is the first derivative; h is the second derivative.

$$g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$$

$$h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$$

2. XGBoost draws on the approach of random forest and supports column sampling, which can not only reduce overfitting, but also reduce calculations. This is also a feature of XGBoost that is different from traditional gbdt.

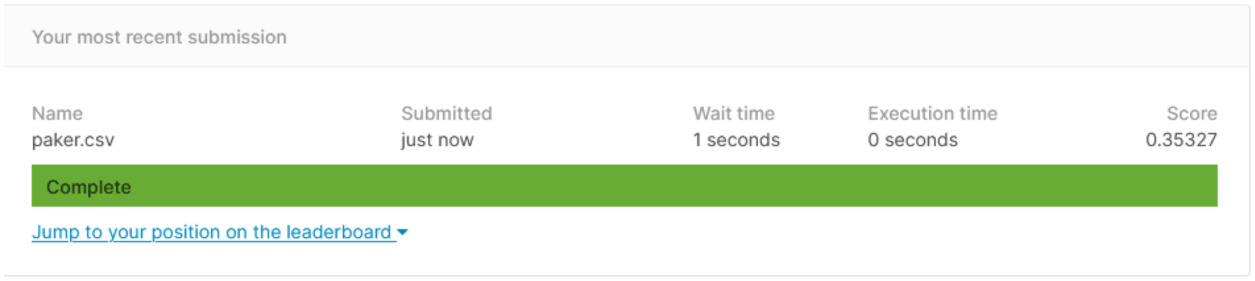
3. XGBoost divides the samples with missing values into the left subtree or the right subtree respectively, and compares the pros and cons of the objective function under the two schemes, so as to automatically divide the samples with missing values without preprocessing the missing features.



```
xgbTreeGrid <- expand.grid(
  nrounds = 1000,
  max_depth = 4,
  eta = 0.02,
  gamma = 0,
  colsample_bytree=1,
  subsample = 0.8,
  min_child_weight = c(2, 3))

train_price <- train$SalePrice#set the train's price as the label for the xgb.Dmatrix.

train2 <- xgb.DMatrix(data = Tratrain, label= train_price)#we have to transform our data to a matrix if we want to use xgboost.
test2 <- xgb.DMatrix(data = Tratest)


#Using GridSearch to tune parameters of xgboost.
GridSearch <-list(
  objective = "reg:linear", #linear regression.
  booster = "gblinear", # because the we choose the reg:linear, gblinear is the best method. gblinear is based on linear model.

  eta = c(0.2),#The default is 0.3 and the alias is leanring Rate, the shrinkage step used in the update process.
  #After each lifting calculation, the algorithm will directly obtain the weight of the new feature.
  max_depth = c(5),#The greater the maximum depth of the tree, the easier it is to over fit,so 5 is ok.
  gamma = 0, #default is 0, we also use 0 here. When a node is split, the node will be split only if the value of the loss function decrease
  colsample_bytree=1, #The default is 1, which is used to control the proportion of the number of columns randomly sampled by each tree.
  min_child_weight=c(3),#default is 1, we use 3 here. When its value is large, it can avoid learning local special samples. But if this valu
  subsample=0.8 #The default is 1. This parameter controls the proportion of random sampling for each tree
)


modle <- xgb.train(data = train2, params=GridSearch, nrounds = 500)


Pridiction <- predict(modle, test2)
test_labels<-c(1461:2919)
test_labels
result <- data.frame(Id = test_labels, SalePrice = (Pridiction))

write.csv(result, file = 'paker1.csv', row.names = F)
```

1. The XGBoost' s parameters has the default numbers. However, we can use the xgbgrid to set the value.
2. Because if we want to use the XGBoost, we have to transform the train into matrix and the data type have to be numeric.
3. The meaning of the parameters: Objective and booster are methods, we choose linear regression here. Eta's default value is 0.3 and the alias is leanring Rate, the shrinkage step used in the update process. Max_depth: the greater the maximum depth of the tree, the easier it is to over fit, so 5 is ok. Gamma's default value is 0, we also use 0 here. When a node is split, the node will be split only if the value of the loss function decreases after splitting. Colsample_bytree: The default is 1, which is used to control the proportion of the number of columns randomly sampled by each tree. Min_child_weight's default value is 1, we use 3 here. When its value is large, it can avoid learning local special samples. But if this value is too high, it will lead to under fitting. Subsample's default value is 1. This parameter controls the proportion of random sampling for each tree.
4. We use the xgb.train function to let it learn. According to our many tests, we have found that nround is around 500, so we set it 500.
5. Depending on the modle's value, we can predict the test's saleprice.
6. Create a data frame and dataset.

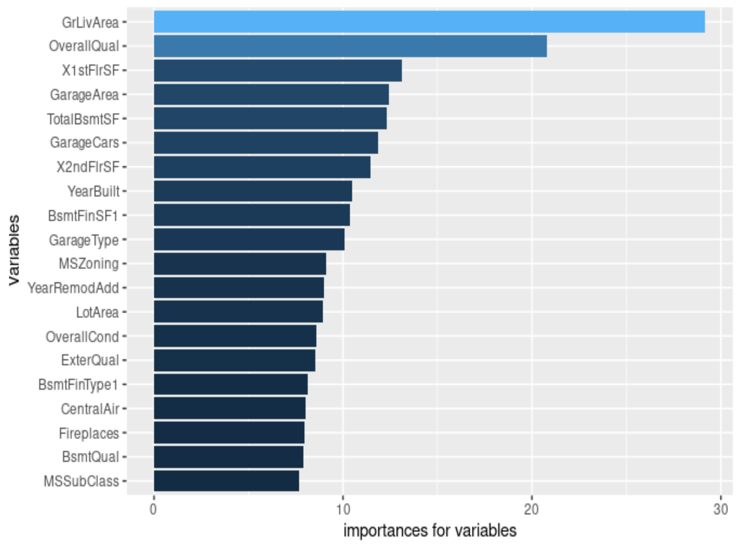
Results and Observations


Your most recent submission

Name	Submitted	Wait time	Execution time	Score
paker.csv	just now	1 seconds	0 seconds	0.35327

This is our first attempt at a data analysis project. Although the score is not ideal, we have learned a lot from this process, since we only use 16 variables to predict.

The Kaggle team name is: **happyXiuxian**.



Through data analysis, we get the following figure and find the most relevant data with the fluctuation of house prices.The way we evaluate the importance of different features is based on the result generated by random forest. Then we find that when compared with character features, integer features present greater influence on sale price. Originally when we evaluated character features, we found lots of features showed a strong relationship with sale price. This result does surprise us.

In the last we found 16 vital features are **BsmtFinType1, MSSubClass, ExterQual, BsmtQual, Foundation, KitchenQual, MSZoning, GarageType, BldgType, HouseStyle, Neighborhood, GrLivArea, OverallQual, GarageCars, YearBuilt, KitchenQual.**

Discussions & Room for improvement

Housing purchase has become a hot topic, and the evaluation and trend of housing value are becoming more and more important. Therefore, in this context, there are some problems related to the prediction of housing price.

According to the data collected before, we observe that many characteristics determine the prediction of housing price. For example, according to the data above, these characteristics are highly correlated with house prices, and together they even determine the direction of house prices. In this experiment, we also found that it is very important to use the datas that have high correlation with house prices, and eliminate the data with low correlation when predicting house prices according to characteristics.

If the data with low correlation is used, the accuracy of prediction may be updated limitedly. Therefore, we only use 16 features that have high correlation with the Sale Price. Garage area, quality evaluation of exterior materials, above ground living area square feet and total square feet of basement area are particularly important, and are also the most valued by consumers. But in the final analysis, there are too many factors that can increase the possibility of house price fluctuations. We find that prediction is only a possibility, nothing is absolute. But analysis, as a guidance, also helps us in the big environment.

Through this process, we have learned some more advanced analysis approaches and r language tools through self-study, and we benefit greatly from it. Since in this project, we only take some variables with great importance into account, thus we also notice that we fail to consider some variables with less importance. Even if they seem to be less significant individually, the impact of the sum of those variables would be more significant than expected.

Furthermore, we do not have a very good grasp of Rl, so some tools may not be used properly. We hope that as we learn more, we will resolve these problems in the near future. Many thanks for your instructive teaching this semester.