University of Waterloo

STAT 844 WINTER 2019

Final Project Report

YimingLi(20803118)

Instructor:
Kun Liang

April 15, 2019

Contents

1. Motivation and introduction of the problem

House price becomes one of the biggest problems almost for everyone in everywhere. The price is so high that many people are under the pressure of buying a house or the overwhelming house loan. Moreover, the growing house price makes people more panic and accelerates their buying plan.

This project aims to explore the real reasons behind the high house price, and how those factors impact the price of the house, in other words, using given housing information to predict the house price. Hopefully, we can give some suggestions when customers buy a new house.

2. Data

2.1 Dataset Description

Our dataset is uploaded from kaggle: https://www.kaggle.com/c/house-prices-advanced-regression-techniques This dataset displays almost every feature of residential homes in Ames, Iowa, and gives the sale price of the house.

This dataset contains training set($1460 \cdot 81$) and testing set($1459 \cdot 80$). Our response variable–'SalesPrice' is included in the training set, which represents the property's sale price in dollars for the corresponding houses. For the input variables, 48 are categrical variables and 31 are continues variables, namely Street,Alley,HouseStyle,YearBuilt, Heating,YrSold, etc. The detailed explanations of 80 input variables can be viewed in the dataset link above.

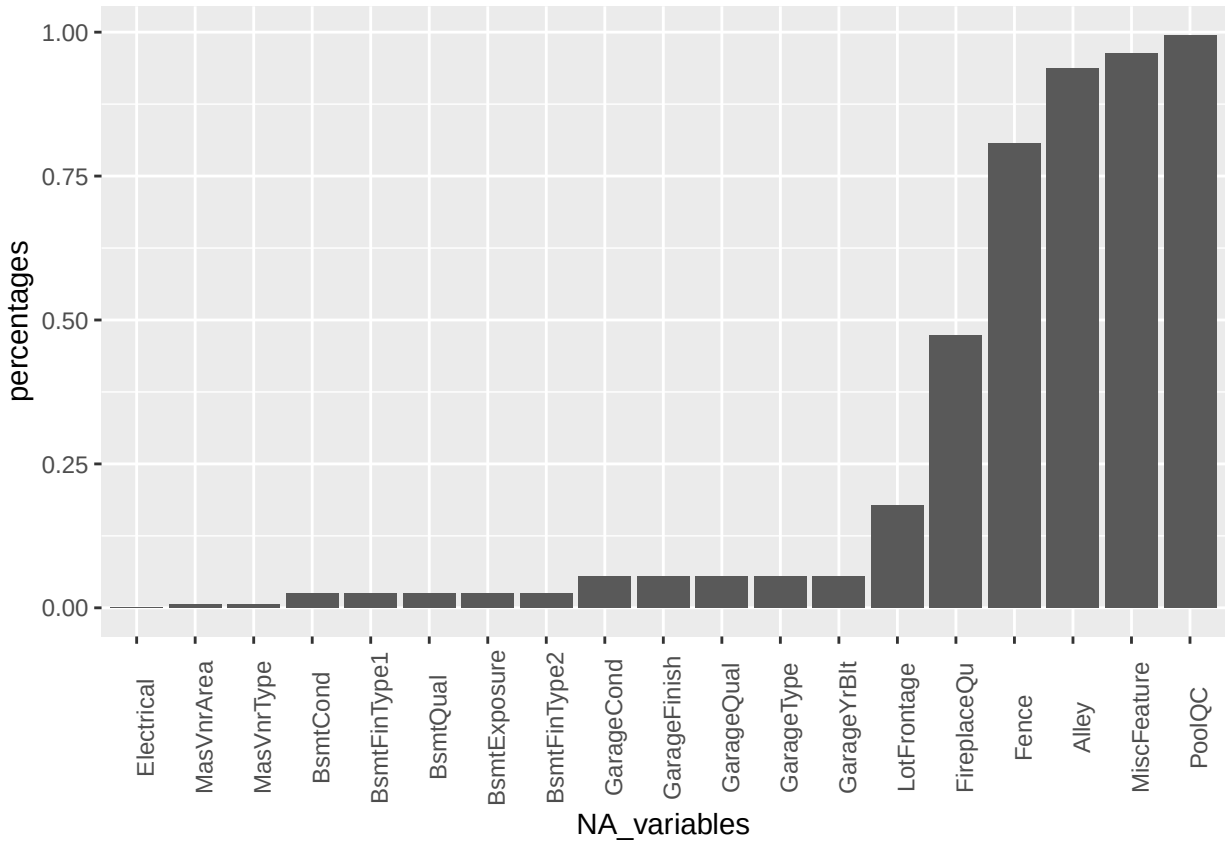
2.2 Overview and fill NAs

First, lets get an ovweview of the data's structure:

```
## 'data.frame':    1460 obs. of  10 variables:
##  $ Id         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning   : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage: int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea    : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street     : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley      : chr  NA NA NA NA ...
##  $ LotShape   : chr  "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour: chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities  : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
```

We can see from the structure that the data mainly contains 'int' and 'character' variables, and there are some NAs in even only first 10 rows. For a clearer understanding of NAs im this data, we findout all columns have NAs and make it into a barplot, to see how much percentage in each variables.

```
##  LotFrontage        Alley   MasVnrType   MasVnrArea     BsmtQual
##          259         1369            8            8           37
##     BsmtCond BsmtExposure BsmtFinType1 BsmtFinType2   Electrical
##           37           38           37           38            1
##  FireplaceQu   GarageType  GarageYrBlt GarageFinish   GarageQual
##          690           81           81           81           81
##   GarageCond       PoolQC        Fence  MiscFeature
##           81         1453         1179         1406
```



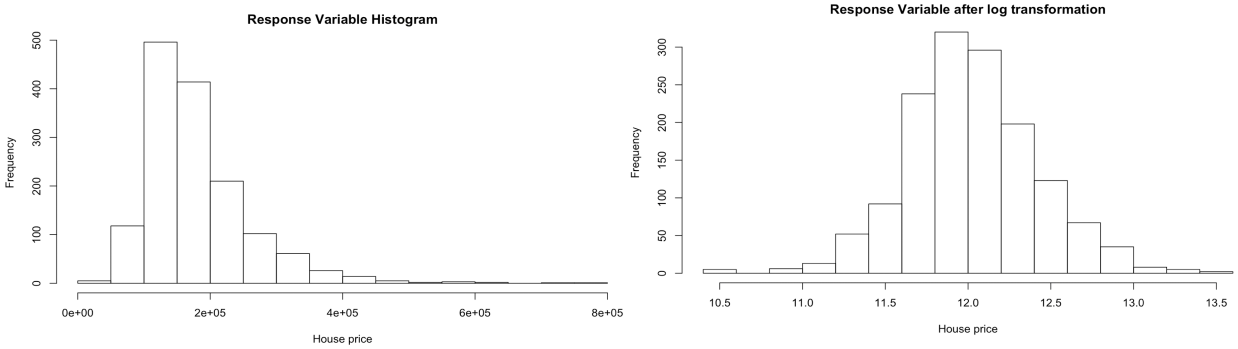
For the last six variables in the barplot, over 15% of the data are NAs, too many NAs makes few value to study on those variables, we delete them from the dataset and focus on the rest.

Then, we fill the rest NAs in our data by median(numeric variables) and the most frequent characters(category variables).

3. Data Preprocessing

3.1 Response variables preprocess

Our response variable is Salesprice of the house, range from 34900 to 755000, which has a Mean of 180921 and the Median is 163000. To understand the distribution, we make a density plot.



We can see from the left plot that the distirbution of response variable is left skewed, to fix this, we apply a log tansformation, and the result on the right plot is quite good.

3.2 Explanatory variables preprocess

In the raw dataset, some explanatory variables has confusing format and types, which can cause bias in our analysis, we need modify those variables before put into our model in the following steps:

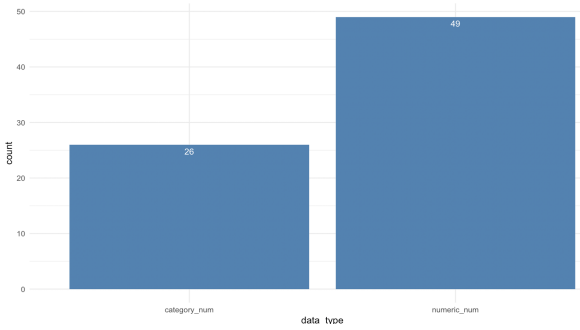
step 1: Transforming some numerical variables which should be categorical into categorical(ex:'YrSold','MoSold','YearBuilt')

step 2:Transforming categorical variables which have ordering meaning into numerical.

For example, BsmtQual(Basement Quality) has 6 levels,"None","Po", "Fa","TA","Gd","Ex", and those words really evaluat the Quality of the basement and have order meaning, so we transform them into numeric 1,2,3,4,5,6 respectively.

step 3: Unifying the type of no order-meaning Categorical variables as factor.

Now our data is consist of two types of explanatory variables, namely numeric and factor. We use a barplot to show its composition.



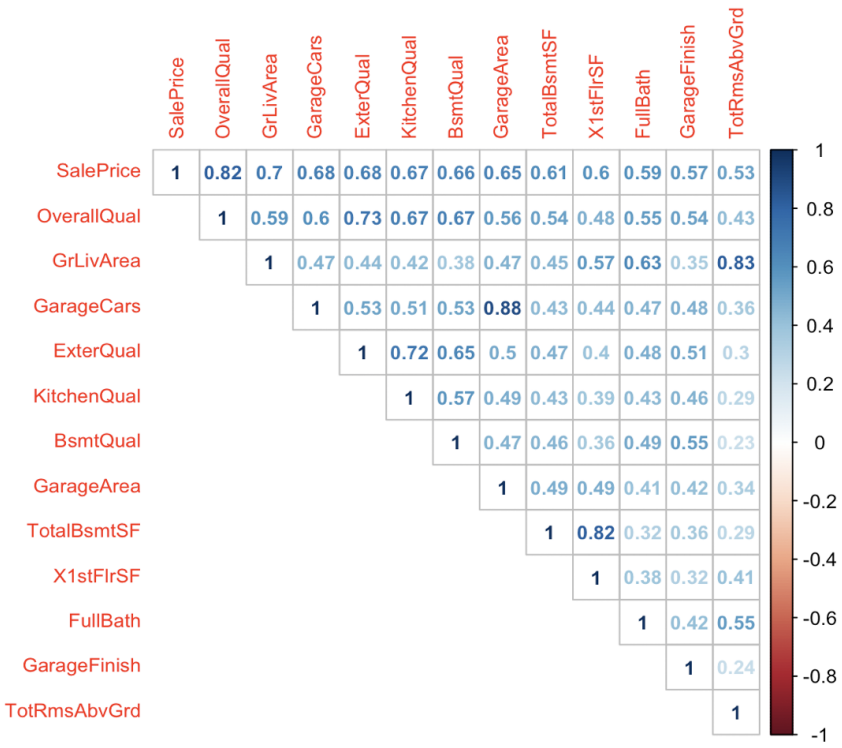
We can see from the plot that there are totally 75 explanatory variables, and the number of the explanatory variables will be too much if we directly use them as training dataset.Therefore, we try to do dimension reduction in the next section.

3.3 Feature reduction

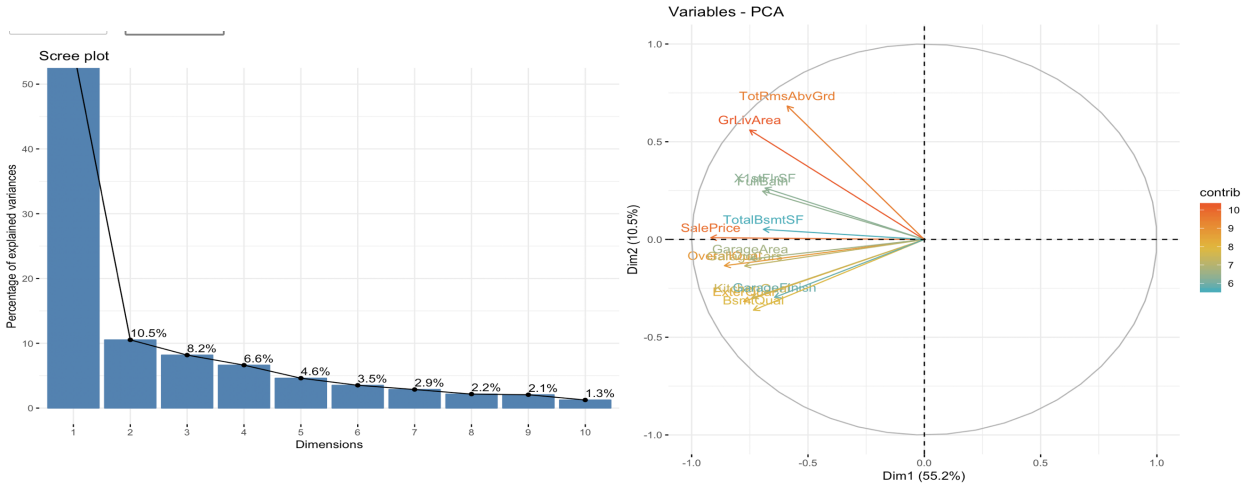
There are 2 types of explanatory variables in our dataset, namely Numeric and Category, and we implement different resonables feature selection method for them respectively.

3.3.1 Feature reduction-Numeric

For numeric independent variables, we first compute the correlation between them and our dependent variable(Salesprice). We filter those independent variables having a correlation over 0.5 with Salesprice and drop the rest. Below is a correlation plot:



From the correlation plot, we can see there are 12 variables having a correlation over 0.5 with slaes price. However, we can also find high correlations between some explanatory variables, for example, TotalBsmtSF(Total Bashroom)&GrLivArea,TotalBsmtSF&X1stFlrSF(1st Floor Area). To aviod the heteroscedasticity and satisfy the assumption of parametric statistical methods, such as linear regression, we use PCA(principle component analysis) to process further feature reduction.



We can see from the left plot that the percentage of explained variables has a dramatically decreasing after the 1st one and drop to only 1.3% for the 10th variable. Therefore, we pick up the top 10 numeric explanatory variables using the ranking of contribution in the right plot.

We also apply log transformation to weaken their skewness and do Centering and Scaling to put them on the same scale.

So far, we get 10 representative numeric explanatory variables, we will discuss category variables then.

3.3.2 Feature reduction-Category

For 26 category variables, we first fit all of them with a linear regression, and use both 'forward' and 'backward' stepwise regression to select impotant categorical varibles. By several iterations to drop unsignificant fators, we finally get the optimal model. We can see the details in the R output below:

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
train$SalePrice ~ MSZoning + Street + LandContour + Utilities +
    LotConfig + Neighborhood + Condition1 + Condition2 + BldgType +
    HouseStyle + YearBuilt + YearRemodAdd + RoofStyle + RoofMatl +
    Exterior1st + Exterior2nd + MasVnrType + Foundation + Heating +
    Electrical + GarageType + GarageYrBlt + MoSold + YrSold +
    SaleType + SaleCondition

Final Model:
train$SalePrice ~ MSZoning + LotConfig + Neighborhood + Condition1 +
    BldgType + HouseStyle + YearBuilt + YearRemodAdd + RoofStyle +
    RoofMatl + Exterior1st + MasVnrType + Foundation + Heating +
    Electrical + GarageType + GarageYrBlt + YrSold + SaleCondition


          Step Df     Deviance Resid. Df Resid. Dev       AIC
1                                  1045   33.06677 -4699.988
2      - MoSold 11 1.782582e-01      1056   33.24502 -4714.138
3  - Condition2  6 2.127959e-01      1062   33.45782 -4716.823
4 - Exterior2nd 14 5.798212e-01      1076   34.03764 -4719.738
5      - Street  1 5.321696e-05      1077   34.03769 -4721.736
6    - SaleType  8 3.406123e-01      1085   34.37831 -4723.198
7   - Utilities  1 7.887983e-03      1086   34.38620 -4724.863
8 - LandContour  3 1.073095e-01      1089   34.49350 -4726.314
```

In the AICstep summary above, we can see that 7 categorical variables are dropped, there are 19 categorical variables left in the final model.

For of 19 categorical variables, some of them specify information about year, and contain too much levels(e.g.'YearBuilt'range from 1817 to 2010, GarageYrBlt range from 1900 to 2010). To save the cost when training, we use every 20 years as a boundary from 1900 to 2020, and aggregate levels during the same 20-year-period.

Moreover,to fit the input format of some advanced training model, we convert the form of all categorical variables into hot_onecode, which denotes levels by 0 and 1. In addition,if the sum of dummies in a column is less than dummies, we delete this explanatory variable.

We finish our feature reduction by join the numerical and categorical variables together. After combining, our explanatory matrix has 1460 rows and 173 columns, which is composed by 10 numerical variables and 162 categorical variables.

Now, we end the data preprocessing,and we will discuss different training model in the following sections.

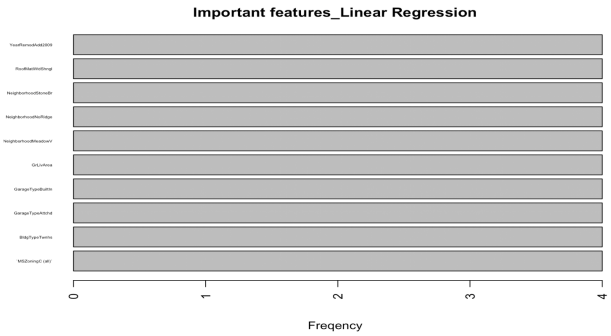
4. Linear Model

In this part, we first fit our data with simple linear regression model, then we apply different kind of robust regression for top 10 significant factors and compare those results. We also implement some smoothing method to explore the interaction between specific explanatory variables and apply lasso regression at last.

4.1 Linear Regression

In the linear regression training process, we use 5-fold cross validation. By diving our dataset into 5 fold, we use 4 folds as training data, and use the rest one to evaluate the model using RMSE(root-mean-square). We compute the average of 5 root-mean-square as a score of our linear model. Also, we extract the top 10 significant variables of each time, and make a table by their showing up times.

```
[1] 0.1494546
 [1] "`MSZoningC (all)`"   "BldgTypeTwnhs"       "GarageTypeAttchd"
 [4] "GarageTypeBuiltIn"   "GrLivArea"           "NeighborhoodMeadowV"
 [7] "NeighborhoodNoRidge" "NeighborhoodStoneBr" "RoofMatlWdShngl"
[10] "YearRemodAdd2009"
```



In the left R output, we can see the average of RMSE is 0.1494546, which is not bad. We also plot the table result on the right. In 10 important explanatory variables, Only "GrLivArea"(Ground living Area) is numerical, others are categorial.

Next, we try to understand how those significant factors influence the Saleprice by applying robust regressions and smoothing method.

4.2 Robust Linear Regression

To figure out the coefficient of top 10 important explanatory variables, we fit them with a linear regression again, but this time, we use Huber psi M-estimator, Tukey bisquare weight-ψ M-estimator and least trimmed squares estimator.

```
[1] "Huber"
                (Intercept)              RoofMatlWdShngl
                11.89507074                   0.36254493
        NeighborhoodStoneBr              YearRemodAdd2009
                 0.31693911                   0.28505228
                  GrLivArea           GarageTypeBuiltIn
                 0.24350039                   0.22862988
           GarageTypeAttchd          NeighborhoodNoRidge
                 0.18535428                   0.16875419
               BldgTypeTwnhs          NeighborhoodMeadowV
                -0.06002747                  -0.32697802
trainData$`MSZoningC (all)`
                -0.57332470
```

```
[1] "Tukey bisquare weight"
                (Intercept)              RoofMatlWdShngl
                11.89593376                   0.35863627
        NeighborhoodStoneBr              YearRemodAdd2009
                 0.31106105                   0.27994211
                  GrLivArea           GarageTypeBuiltIn
                 0.24293615                   0.23440983
           GarageTypeAttchd          NeighborhoodNoRidge
                 0.18837699                   0.16488199
               BldgTypeTwnhs          NeighborhoodMeadowV
                -0.06468945                  -0.33168161
trainData$`MSZoningC (all)`
                -0.54222650
```

```
[1] "least trimmed squares "
                (Intercept)         NeighborhoodStoneBr
                11.68369557                   0.59728978
           GarageTypeAttchd                    GrLivArea
                 0.42213862                   0.22634334
        YearRemodAdd2009 trainData$`MSZoningC (all)`
                 0.16265711                   0.03069962
              BldgTypeTwnhs
                -0.19556568
```

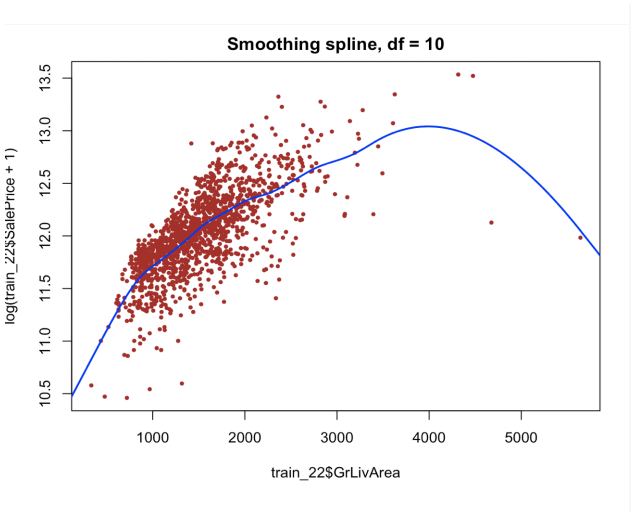
It is clear that the order of those important variables are the same, and their coefficients are similar too(Due to the singular problem happens when fit least trimmed squares estimator,we don not contain GarageTypeBuiltIn,NeighborhoodMeadowV,NeighborhoodStoneBr,and RoofMatlWdShngl in the model).

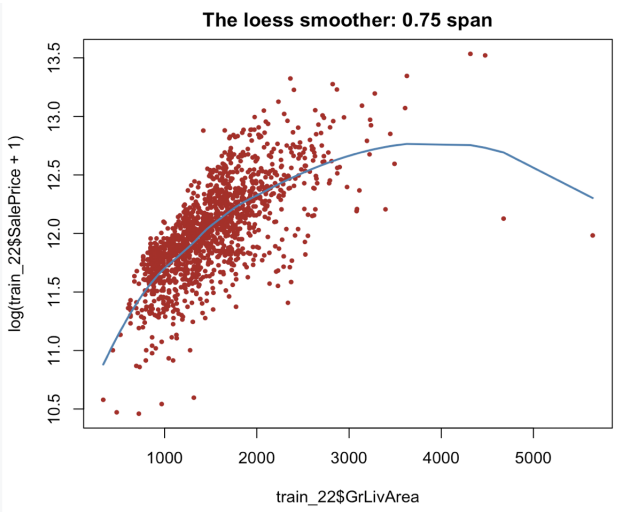
RoofMatlWdShngl(Wood Shingles) has the largest positive coefficient in Huber and Tukey models, while MSZoningC(a commercial zoning classification of the sale) has the most negative influence to the sale price, While it shows a positive influence in least trimmed squares estimator model. Moreover, BldgTypeTwnhs(Type of dwelling:Townhouse Inside Unit) has a negative impact in all 3 models.

4.3 Smoothing methods

In top 10 impotant variables, GrLivArea(Above grade (ground) living area square feet) is the only numerical varialble. We then focusing on how GrLivArea determine the Saleprice and its interaction with other variables.

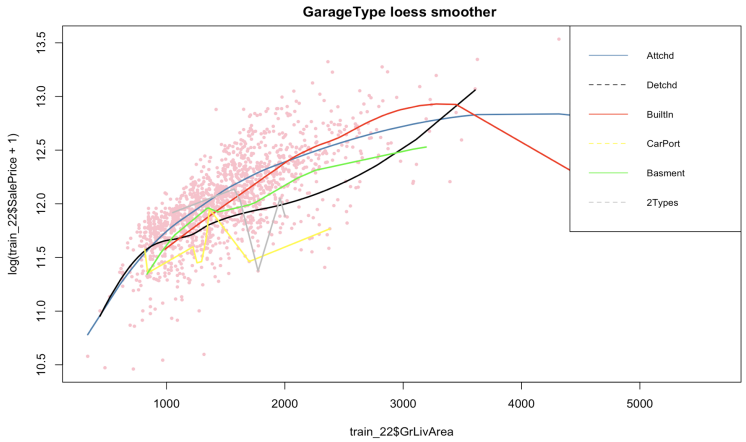
Firstly, we fit a spline smoothing model and a Locally Weighted Regression(Loess) for GrLivArea and Saleprice.





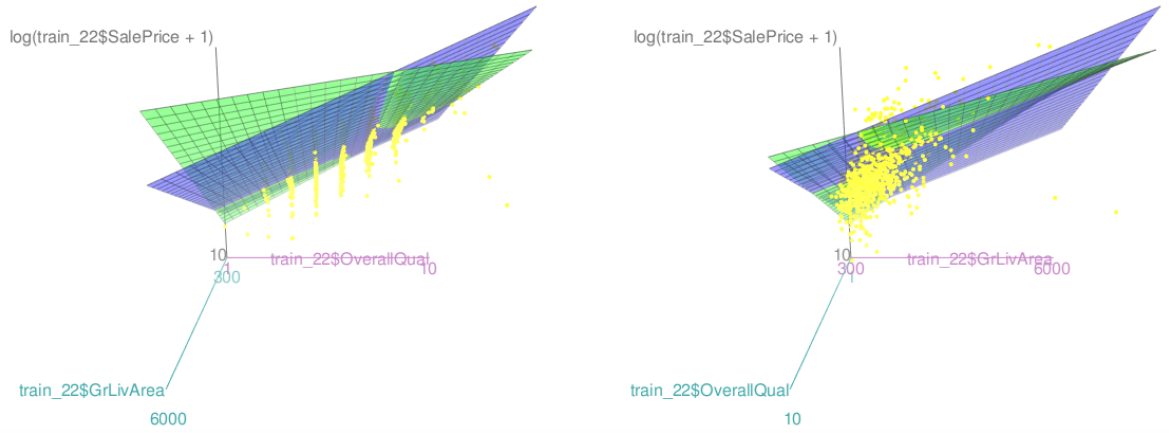
Both two methods shows a increasing trend of price as GrLivArea goes up followed by a downward trend when the price exceeds about 4000. However, loess is a more reasonable model, because it is not influenced too much by the extreme outliers, while price has a remarkable drop for extreme GrLivArea values, which is unrealistic.Therefore, we prefer loess and wil apply it in the following analysis.

It is surprised that when GrLivArea over 4000, the large the area, the lower the price. We try to figure out this strange phenomenon, and we slove this question by exploring the relationship among GrLivArea, GarageType, and Salesprice. This time, we sort our data into different categories by Garagetypes, and fit loess model for every groups.



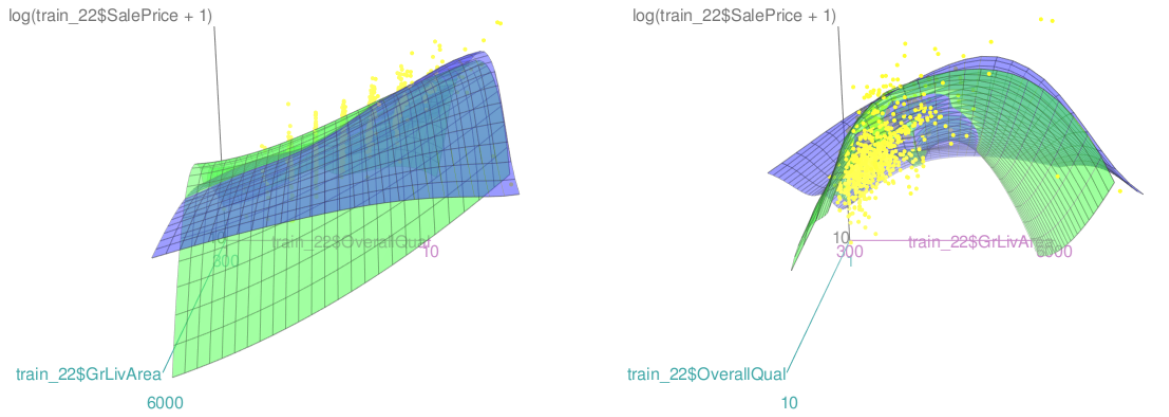
The graph shows that for house having GrLivArea over 4000, their garage is either Attached or Built in.Since Built in garage and Attached garage are also counted as part of the GrLivArea, the real GrLivArea for most house so-called over 4000 GrLivArea may not true, or less than 4000, which is definitely a negative impact for a luxury big house. Other reasons for decreaing price might be the worse design for Built in garage and Attached garage in terms of indoor appearance or Practicality.

Next, we explore more about GrLivArea with another numerical variable, OverallQual(overall quality) by fit different multiple explanatory models.



We first fit them into an interaction model and a linear regression model. In the left graph,x-axis(horizontal) represents OverallQual, y-axis stands for GrLivArea, and z-axis means Salesprice. In the right graph,x-axis(horizontal) represents GrLivArea, y-axis stands for OverallQual, and z-axis means Salesprice. The Blue plane is the model of linear regression and the greeen plane is the model of interaction.We can see that when GrLivArea and OverallQual are both small, the green plane is under the blue plane, which means their interaction is negative. The green plane exceed the blue one when large value of GrLivArea and small value of OverallQual and shows a positive interaction. However, it is intersting to see that there is a negative interaction between OverallQual and GrLivArea, in other words, the blue plane is above the green one when both factors have a big value.

Next, we are going to compare loess model and smoothing spline model on multiple variables.



For two planes in each graph, blue plane is the model of smoothing spline, green plane is the loess model.We can see from two graphs that when 2 explanatory variables are small, blue plane is under the green plane, which means the smoothing model's predicted price is small than that of loess. However, when two variables become large, the green plane is always above the blue plane, and smoothing model has a higher predicted Saleprice. We can conclude that the smoothing method is more sensitive to those outliers with pretty large values than loess method.

4.4 Lasso regression

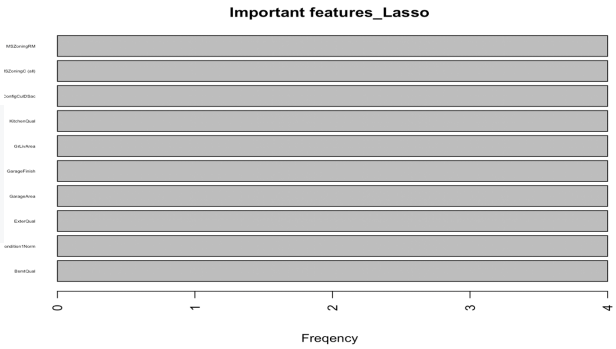
Lasso (least absolute shrinkage and selection operator) performs both variable selection and regularization so that it enhances the prediction accuracy and interpretability of the statistical model,it can alter the model

fitting process by selecting only a subset of the provided explanatory variables in the final model.

In our dataset, there are too many dimensions, which makes Lasso a perfect method to apply. Every time a new parameter is introduced, we will add a penalty in the loss function, so when optimizing the model, Lasso can appropriately control the number of our explanatory variables to get the best model.

We also use 5-fold crossvalidation, and use the average RMSE as our score. Also, top 10 importance variables in each group are recorded to create a importance feature list.

```
[1] 0.1424195
 [1] "BsmtQual"          "Condition1Norm"    "ExterQual"
 [4] "GarageArea"        "GarageFinish"      "GrLivArea"
 [7] "KitchenQual"       "LotConfigCulDSac" "MSZoningC (all)"
[10] "MSZoningRM"
```



We can see from the output that the RMSE is lower than that of linear regression, and many dummies variables like Garage types and Neignborhood names are replaced by categorical variables as "BsmtQual"(Basement Quality) ,ExterQual(Exterior material quality) and numerical variables as "GarageArea". Overall, Lasso gives us more reasonalble impotant features and a lower RMSE.

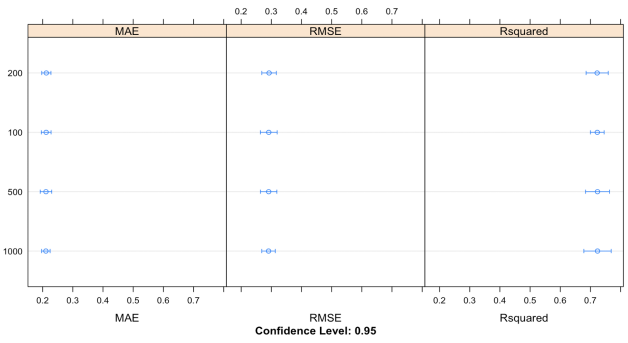
5. Tree based model

In this section, we will implement some tree based model and do some hyperparameter tuning to improve the accuracy of our predictions.

5.1 Randomforest Model

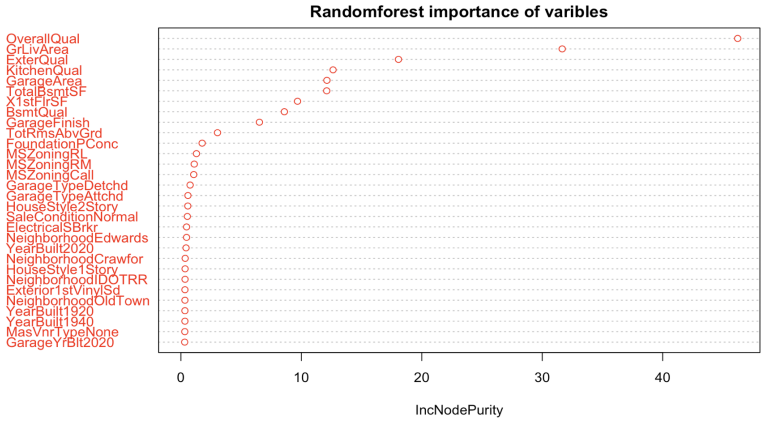
Randomforest model is an efficient classifier, firstly, we just use the defualt parameter to train our model. We use 5-fold crossvalidation, and use the average RMSE as our score.

The result gives us a average RMSE of 0.1482815, which is lower than Lasso. Then we change the number of trees in Randomforest Model, we use 100,200,500,and 1000 and run 5-fold crossvalidation for each parameter to find the optimal parameter.




```
Call:
summary.resamples(object = results)

Models: 100, 200, 500, 1000
Number of resamples: 5

MAE
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
100  0.1979391 0.2002283 0.2146524 0.2116764 0.2154309 0.2301314    0
200  0.1993840 0.1997197 0.2140262 0.2122377 0.2230053 0.2250534    0
500  0.1905507 0.2036880 0.2081767 0.2110151 0.2246673 0.2279930    0
1000 0.1986416 0.2054754 0.2059882 0.2106870 0.2154576 0.2278721    0

RMSE
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
100  0.2668518 0.2726579 0.2972722 0.2917227 0.2996878 0.3221439    0
200  0.2727698 0.2761329 0.2892398 0.2927383 0.3116259 0.3139229    0
500  0.2624767 0.2829090 0.2850487 0.2914608 0.3129498 0.3139200    0
1000 0.2803630 0.2819246 0.2824463 0.2910263 0.2875167 0.3228807    0

Rsquared
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
100  0.6926986 0.7190599 0.7298535 0.7227383 0.7321562 0.7399232    0
200  0.6741678 0.7214136 0.7297990 0.7222959 0.7317860 0.7543133    0
500  0.6721759 0.7157659 0.7357208 0.7234037 0.7372798 0.7560760    0
1000 0.6916819 0.7000114 0.7054269 0.7234961 0.7409818 0.7793784    0
```

From the graph and output, we can see that the distribution of R-squred for 4 parameters are nearly the same, while when ntree equals 1000, the means and medians of both RMSE and MAE are the lowest. Therefore, we set the number of trees as 1000, and run the same training process. The result improves to an average RMSE of 0.1465942, although is also larger than that of Lasso.The reason of this might be the excessive number of variables in Randomforest model than that of Lasso. Anyway, their average RMSE is close, we can say Randamforest with 1000 tree is a good medel. We then plot its importance features.



We can see from the graph that in the top 10 important features, randomforest introduces more numerical variables, such as OverallQual,GrlivArea and GarageArea, and it gives OverallQual the largest weight.MSzonigC(commercial zoning classification of the sale) is not in top 10 is a big difference compared with linear models. In addition, features ranked out of top 10 have only few weights.

5.2 GBM Model

GBM(Gradient boosting model) is a tree based model that assembles weak prediction models like decision trees into strong learners, which is more powerful boosting classifier than linear model or randomforest.We first use the default value to apply a 5-fold cross validation.

The result is really disappointing, an average RMSE of 0.1578925, and by far the worst one. There must something wrong with our parameter, so

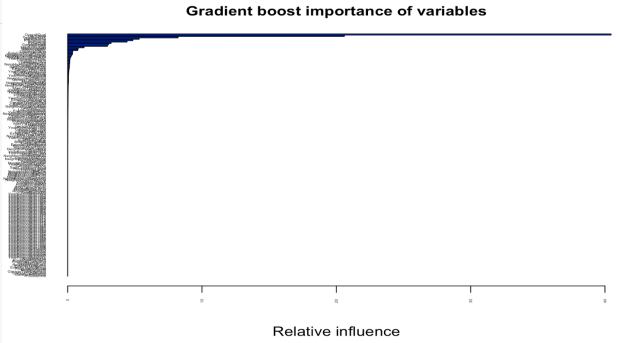
we use caret package in R to find our right hyperparameter combination. We set possible values of ntrees eaquals 100,500,and 800, interaction.depth equals 1,3, and 5, number of minobsinnode equals 5,10, and 20, and shrinkage 0.01,0.1 and 0.3.By 5-fold cross validation, we find the best combination.

	n.trees <dbl>	interaction.depth <dbl>	shrinkage <dbl>	n.minobsinnode <dbl>
21	800	5	0.01	5

1 row

We run the training process again with the new parameter, and our average RMSE decrese to 0.1386404, which is a huge improvement, and by far the best model. We also plot the importance features below.

	var <fctr>	rel.inf <dbl>
OverallQual	OverallQual	4.045179e+01
GrLivArea	GrLivArea	2.059580e+01
TotalBsmtSF	TotalBsmtSF	8.215289e+00
GarageArea	GarageArea	5.316555e+00
KitchenQual	KitchenQual	4.842592e+00
BsmtQual	BsmtQual	4.419308e+00
ExterQual	ExterQual	3.226655e+00
GarageFinish	GarageFinish	3.037142e+00
X1stFlrSF	X1stFlrSF	2.958784e+00
MSZoningCall	MSZoningCall	1.215675e+00
MSZoningRM	MSZoningRM	7.831214e-01
MSZoningRL	MSZoningRL	7.468157e-01
ElectricalSBrkr	ElectricalSBrkr	3.708668e-01
SaleConditionNormal	SaleConditionNormal	3.653050e-01
NeighborhoodCrawfor	NeighborhoodCrawfor	3.501324e-01
NeighborhoodEdwards	NeighborhoodEdwards	2.854259e-01
NeighborhoodClearCr	NeighborhoodClearCr	2.220980e-01
YearRemodAdd1997	YearRemodAdd1997	1.862916e-01



We can see from the graph that the top 10 important features of GBM are the same with that of Randonforest, and the difference is only some of their orders. Similarly, GBM also give much weight to top important features and few weight to others.

5.3 Xgboost Model

XGBoost(Extreme Gradient Boosting) is one of the most popular machine learning algorithm these days. It has recently been dominating applied machine learning and Kaggle competitions. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

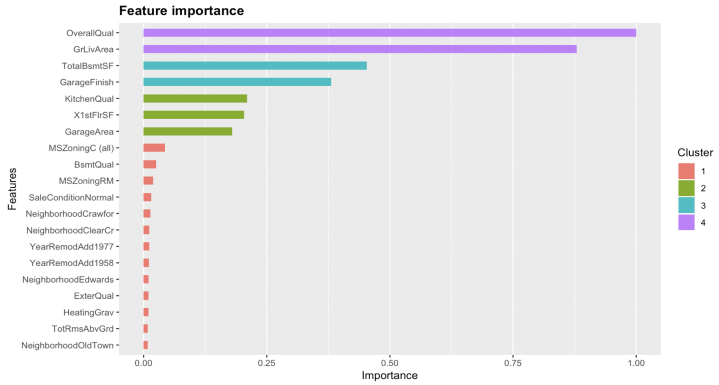
Now, let us apply Xgboost on our data. Same as training process for other model, we use 5-fold crossvalidation and record their RMSE, then evaluate the model by average RMSE.

Surprisedly, with default parameter, the average RMSE is 0.1514321, which is the lowest of all the model we try. It seems that hyperparameter tuning is necessary. We set posiible nrounds equals 50,100,200, and 300,eta equals 0.1,0.2,0.3,0.4, and 0.5,max_depth equals 2,3,4,5, and 6, and min_child_weight equals 1,2,and 3.By using caret training 5-fold cross validation grid searching, we get our best parameter.

	nrounds <dbl>	max_depth <dbl>	eta <dbl>	gamma <dbl>	colsample_bytree <dbl>	min_child_weight <dbl>	subsample <dbl>
72	300	2	0.2	0	1	3	1

1 row

As I expected, this new model gives us a lowest average RMSE,0.1385127. Let us check the important features.



Although the top 10 important features has no difference with other 2 tree based models, we can see that Xgboost divide them into 4 clusters, and give more weight to top features.

6. Statistical Conclusions

In conclusion, after several feature reduction methods, we have tried linear regression, Lasso regression, Randomforest, Gradient Boosting, and Xgboost in this project. By using a 5 fold cross valition to train the model, we use the average RMSE compare the accuracy of those models.

ID <chr>	RMSE_default <dbl>	RMSE_bestmodel <dbl>
Linear_Regression	0.1494546	0.1494546
Lasso	0.1424195	0.1424195
Randomforest	0.1467538	0.1465942
Gradient Boosting_Machine	0.1578925	0.1386404
Xgboost	0.1514321	0.1385127

(note:the first RMSE column is RMSE of default parameters, the second RMSE column is RMSE after hyperparameter tuning. Linear regression and Lasso do not change their parameter, so RMSE are the same.)

We can see that alought RMSE of 5 models are very close,after hyperparameter tuning, Xgboost and Gradient boosting machine give us lower RMSE, so I will say those two model are the best. However, if we also consider about training time, lasso model is more efficient, because, it spends much more time to train a Xgboost or Gradient boosting machine model, and the time used on hyperparameter tuning is relatively long.

6. Conclusions in the context of the problem

According to our project, it is hard to predict the housing Saleprice in Ames, Iowa with a high precision, because there are too many factors can impact the Saleprice. However, we do find some top important features, which shows up in any method we apply.

The first one is GrLivArea(Above grade (ground) living area square feet), the larger the area, the higher the price. However,it is not the case for house having a GrLivArea over 4000, as the Area of built-in basement will be counted as part of those house. Moreover, OverallQul(overall quality), ExterQual(Evaluates the quality of the material on the exterior), and KitchenQual(kitchen quality) are also importnat factors, the better the quality, the higher the price.

Other significant factors are Neighborhood, and RoofMatl(Roof materials). Houses located in StoneBr(StoneBr Stone Brook) and NoRidge(Northridge) are usually with a higherprice, while house in MeadowV(Meadow Village) is the way cheaper.

In addition, if it is possible, remember to identify the general zoning classification of the sale as Commerical, as the salesprice often lower in this method of transaction.

Below is a wordCloud of all the top important variables.

7. Further work

Altough in our project, we apply several method to reduce the dimension of the training data, while the performance of the feature selection is not quite good, especially in the randomforest models, further work may study on a better way of feature reduction process.

Moreover, the dataset only contains 1460 samples, and the size is not enough to make a solid model or general conclusion. It is a good idea to collect more sample or use some data agumentation method to increase the size of data.

8.Appendix

8.1 Details of explanatory variables

Below is a link which explains the details of all explanatory variables in this dataset:

https://github.com/DS-100/sp17/tree/master/materials/hw/hw6

8.2 Literature

For data preprocessing, we refer to:

https://www.kaggle.com/tannercarbonati/detailed-data-analysis-ensemble-modeling

https://www.kaggle.com/paultimothymooney/get-your-model-ready-to-submit

https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/stepAIC.html

http://www.sthda.com/english/wiki/factoextra-r-package-easy-multivariate-data-analyses-and-elegant-visualization

For 5-fold cross validation format, we refer to:

https://gist.github.com/duttashi/a51c71acb7388c535e30b57854598e77

For smoothing method, we refer to:

http://sas.uwaterloo.ca/~rwoldfor/courses/AdvancedRegression/Lectures/2017/Smoothing.html

For tree models training, we refer to:

https://xgboost.readthedocs.io/en/latest/R-package/xgboostPresentation.html

For hyperparameter tuning, we refer to: https://cran.r-project.org/web/packages/caret/vignettes/caret.html

8.3 RMarkdown Code of this project

```
train<-read.csv("train.csv",stringsAsFactors = FALSE)
```

```
str(train[1:10])
```

```
NAinput_variables <- apply(train, 2, function(l){
length(which(is.na(l) == T))
})
NAinput_variables[which(NAinput_variables!=0)]
```

```
library(ggplot2)
perc = apply(train, 2, function(l){
length(which(is.na(l) == T))/length(l)
})
perc=perc[which(perc!=0)]
perc=perc[order(perc)]
oo=data.frame(jj=names(perc),kk=unname(perc))
oo$Team3 <- reorder(oo$jj, oo$kk)
p=ggplot(oo, aes(y=kk))+geom_bar(aes(x=Team3), data=oo, stat="identity")+theme(axis.text.x = element_te
p
```

```
NA_vari=c('Electrical','MasVnrType','MasVnrArea',"BsmtCond","BsmtFinType1","BsmtQual","BsmtExposure","Bs

for (k in NA_vari) {
    if(class(train[k][1,])=='character'){
      train[k][is.na(train[k])]=names(sort(table(train[k]),decreasing = TRUE)[1])
    }else{
      train[k][is.na(train[k])]=median(train[k][,], na.rm = TRUE)
    }

}
```

```
perc = apply(train, 2, function(l){
length(which(is.na(l) == T))/length(l)
})
perc=perc[which(perc!=0)]
perc=perc[order(perc)]
perc
```

```r
library(dplyr)
```

```r
train=train %>% dplyr::select(-c(LotFrontage,FireplaceQu,Fence,Alley,MiscFeature,PoolQC))
```

```r
for (k in c('YrSold','MoSold','YearBuilt','YearRemodAdd','GarageYrBlt')) {
  train[,k]=as.factor(train[,k])
}
```

```r
order_categories = c("BsmtQual", "BsmtCond", "GarageQual", "GarageCond", "ExterQual", "ExterCond", "Hea

BsmtQual = c("None", "Po", "Fa", "TA", "Gd", "Ex")
BsmtCond = c("None", "Po", "Fa", "TA", "Gd", "Ex")
GarageQual = c("None", "Po", "Fa", "TA", "Gd", "Ex")
GarageCond = c("None", "Po", "Fa", "TA", "Gd", "Ex")
ExterQual = c("Po", "Fa", "TA", "Gd", "Ex")
ExterCond = c("Po", "Fa", "TA", "Gd", "Ex")
HeatingQC = c("Po", "Fa", "TA", "Gd", "Ex")
KitchenQual = c("Po", "Fa", "TA", "Gd", "Ex")
BsmtFinType1 = c("None", "Unf", "LwQ", "Rec", "BLQ", "ALQ", "GLQ")
BsmtFinType2 = c("None", "Unf", "LwQ", "Rec", "BLQ", "ALQ", "GLQ")
Functional = c("Sal", "Sev", "Maj2", "Maj1", "Mod", "Min2", "Min1", "Typ")
BsmtExposure = c("None", "No", "Mn", "Av", "Gd")
GarageFinish = c("None", "Unf", "RFn", "Fin")
LandSlope = c("Sev", "Mod", "Gtl")
LotShape = c("IR3", "IR2", "IR1", "Reg")
PavedDrive = c("N", "P", "Y")
CentralAir=c("N","Y")

order_names = list(BsmtQual, BsmtCond, GarageQual, GarageCond,ExterQual, ExterCond, HeatingQC,KitchenQua

i = 1
for (c in order_categories) {
  train[, c] = as.numeric(factor(train[, c], levels = order_names[[i]]))
  i = i + 1
}
```

```r
character_names=names(which(sapply(train, is.character)))
```

```r
for (k in character_names) {
  train[,k]=as.factor(train[,k])
}
```

```r
category_num=length(which(sapply(train, is.factor)))
numeric_num=length(which(sapply(train, is.numeric)))
```

```r
library(ggplot2)
# Basic barplot
count=c(category_num,numeric_num)
data_type=c('category_num','numeric_num')

p<-ggplot(data=data.frame(count,data_type), aes(x=data_type, y=count)) +
  geom_bar(stat="identity", fill="steelblue")+geom_text(aes(label=count), vjust=1.6, color="white", size
  theme_minimal()
p
```

```r
summary(train$SalePrice)
```

```r
summary(train$SalePrice)
```

```r
boxplot(train$SalePrice)
```

```r
hist(train$SalePrice,main='Response Variable Histogram',xlab='House price')
```

```r
train$SalePrice <- log(train$SalePrice + 1)
```

```r
hist(train$SalePrice,main='Response Variable after log transformation',xlab='House price')
```

```r
library(corrplot)
numeric_var=train[, which(sapply(train, is.numeric))]
numeric_cor=cor(numeric_var, use="pairwise.complete.obs")
numeric_cor1=as.matrix(sort(numeric_cor[,'SalePrice'], decreasing = TRUE))
numeric_cor2=names(which(apply(numeric_cor1, 1, function(x) abs(x)>0.5)))
numeric_cor <- numeric_cor[numeric_cor2, numeric_cor2]
corrplot(numeric_cor, method="number", tl.pos = "lt", tl.cex = 0.8,cl.cex = 0.8,number.cex=0.8,type="up
```

```r
library(factoextra)
train_num.pca <- prcomp(train[,numeric_cor2], scale = TRUE)
fviz_pca_var(train_num.pca,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = FALSE     # Avoid text overlapping
             )
fviz_screeplot(train_num.pca, addlabels = TRUE, ylim = c(0, 50))
```

```r
sort(train_num.pca$center,decreasing = TRUE)
num_names=names(sort(train_num.pca$center,decreasing = TRUE)[1:11])
```

```r
library(moments)
library(MASS)
num_data=train[,num_names]
skewed_vari=sapply(num_names, function(x) {
    skewness(num_data[[x]])
})
skew_names=names(skewed_vari[abs(skewed_vari) > 0.5])
for (i in skew_names){
  num_data[[i]] <- log(num_data[[i]] + 1)
}
```

```r
library(caret)
standard_model <- preProcess(num_data, method=c("center", "scale"))
numeric_data<- predict(standard_model, num_data)
dim(numeric_data)
```

```r
library(MASS)
categories_var=train[, which(sapply(train, is.factor))]
fit <- lm(train$SalePrice ~. , data =categories_var)
step <- stepAIC(fit, direction="both")
```

```r
step$anova
```

```r
categories_name=attr(terms(step), "term.labels")
```

```
train$YearBuilt=as.numeric(as.character(train$YearBuilt))
train$YearBuilt[which(train$YearBuilt<=1900)]=1900
train$YearBuilt[which((train$YearBuilt>1900)&(train$YearBuilt<=1920))]=1920
train$YearBuilt[which((train$YearBuilt>1920)&(train$YearBuilt<=1940))]=1940
train$YearBuilt[which((train$YearBuilt>1940)&(train$YearBuilt<=1960))]=1960
train$YearBuilt[which((train$YearBuilt>1960)&(train$YearBuilt<=1980))]=1980
train$YearBuilt[which((train$YearBuilt>1980)&(train$YearBuilt<=2000))]=2000
train$YearBuilt[which((train$YearBuilt>2000)&(train$YearBuilt<=2020))]=2020
train$YearBuilt=as.factor(train$YearBuilt)

train$GarageYrBlt=as.numeric(as.character(train$GarageYrBlt))
train$GarageYrBlt[which(train$GarageYrBlt<=1900)]=1900
train$GarageYrBlt[which((train$GarageYrBlt>1900)&(train$GarageYrBlt<=1920))]=1920
train$GarageYrBlt[which((train$GarageYrBlt>1920)&(train$GarageYrBlt<=1940))]=1940
train$GarageYrBlt[which((train$GarageYrBlt>1940)&(train$GarageYrBlt<=1960))]=1960
train$GarageYrBlt[which((train$GarageYrBlt>1960)&(train$GarageYrBlt<=1980))]=1980
train$GarageYrBlt[which((train$GarageYrBlt>1980)&(train$GarageYrBlt<=2000))]=2000
train$GarageYrBlt[which((train$GarageYrBlt>2000)&(train$GarageYrBlt<=2020))]=2020
train$GarageYrBlt=as.factor(train$GarageYrBlt)
```

```
categories_data=as.data.frame(model.matrix(~.-1, train[,categories_name]))
dim(categories_data)
```

```
categories_dummies=categories_data[,-which(colSums(categories_data)<c)]
dim(categories_dummies)
```

```
SalePrice1=train$SalePrice
numeric_data$SalePrice= NULL
train_new<- cbind(numeric_data,categories_dummies,SalePrice1)
dim(train_new)
dim(numeric_data)
dim(categories_dummies)
```

```
#Randomly shuffle the data
train_new1<-train_new[sample(nrow(train_new)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(train_new1)),breaks=4,labels=FALSE)
#Perform 10 fold cross validation
linear_rmse=c()
important_names=c()
for(i in 1:4){
#Segement your data by fold using the which() function
testIndexes <- which(folds==i,arr.ind=TRUE)
testData <- train_new[testIndexes, ]
trainData <- train_new[-testIndexes, ]
#Use the test and train data partitions however you desire...
lm_train<-lm(trainData[,173] ~ ., data=trainData[,-173])
prediction <- predict(lm_train, testData, type="response")
model_output<-cbind(testData,prediction)
model_output$log_pre<-model_output$prediction
model_output$log_sp<-model_output$SalePrice
rmse(model_output$log_sp,model_output$log_pre)
linear_rmse=c(linear_rmse,rmse(model_output$log_sp,model_output$log_pre))
important_names=c(important_names,names(sort(abs(lm_train$coefficients),decreasing = TRUE)[2:30]))
```

```
}
```

```
mean(linear_rmse)
names(sort(table(important_names),decreasing = TRUE)[1:10])
barplot(sort(table(important_names),decreasing = TRUE)[1:10],main="Important features_Linear Regression
```

```
huber_fit=rlm(SalePrice1 ~ trainData$`MSZoningC (all)`+BldgTypeTwnhs+GarageTypeAttchd+GarageTypeBuiltIn
print('Huber')
sort(huber_fit$coefficients,decreasing = TRUE)
```

```
bisquare_fit=rlm(SalePrice1 ~ trainData$`MSZoningC (all)`+BldgTypeTwnhs+GarageTypeAttchd+GarageTypeBuil
print('Tukey bisquare weight')
sort(bisquare_fit$coefficients,decreasing = TRUE)
```

```
lm_fit=lm(SalePrice1 ~ trainData$`MSZoningC (all)`+BldgTypeTwnhs+GarageTypeAttchd+GarageTypeBuiltIn+GrL
sort(lm_fit$coefficients,decreasing = TRUE)
```

```
ltsreg_fit=ltsreg(SalePrice1 ~ trainData$`MSZoningC (all)`+BldgTypeTwnhs+GarageTypeAttchd+GrLivArea+Yea
print('least trimmed squares ')
sort(ltsreg_fit$coefficients,decreasing = TRUE)
```

```
library(splines)
train_22<-read.csv("train.csv",stringsAsFactors = FALSE)
xrange= extendrange(train_22$GrLivArea)
xnew= seq (min(xrange), max(xrange),length.out = 300)
df<-10
sm =smooth.spline(train_22$GrLivArea,log(train_22$SalePrice+1),df=10)
ypre.sm <- predict(sm,x=xnew)$y
plot(train_22$GrLivArea,log(train_22$SalePrice+1),
     col="firebrick",pch=19,cex=0.5,
     main = paste("Smoothing spline, df =", df))
lines(xnew,ypre.sm,col="blue",lwd=2)
abline(h = 0, col=adjustcolor("black", 0.5), lwd=2, lty=2)
```

```
fit1 = loess(log(train_22$SalePrice+1)~train_22$GrLivArea)
plot(train_22$GrLivArea,log(train_22$SalePrice+1),
     col="firebrick",pch=19,cex=0.5,main ="The loess smoother: 0.75 span")
Xorder = order(train_22$GrLivArea)
pre = predict(fit1)
abline(h = 0, col=adjustcolor("black", 0.5), lwd=2, lty=2)
lines(train_22$GrLivArea[Xorder],pre[Xorder],lwd=2,col="steelblue")
```

```
Attchd_fit=loess(log(train_22$SalePrice+1)[which(train$GarageType=="Attchd")]~train_22$GrLivArea[which(
Attchd_order=order(train_22$GrLivArea[which(train$GarageType=="Attchd")])
Attchd_pred=predict(Attchd_fit)


Detchd_fit=loess(log(train_22$SalePrice+1)[which(train$GarageType=="Detchd")]~train_22$GrLivArea[which(
Detchd_order=order(train_22$GrLivArea[which(train$GarageType=="Detchd")])
Detchd_pred=predict(Detchd_fit)

BuiltIn_fit=loess(log(train_22$SalePrice+1)[which(train$GarageType=="BuiltIn")]~train_22$GrLivArea[whic
BuiltIn_order=order(train_22$GrLivArea[which(train$GarageType=="BuiltIn")])
BuiltIn_pred=predict(BuiltIn_fit)
```

```
CarPort_fit=loess(log(train_22$SalePrice+1)[which(train$GarageType=="CarPort")]~train_22$GrLivArea[which
CarPort_order=order(train_22$GrLivArea[which(train$GarageType=="CarPort")])
CarPort_pred=predict(CarPort_fit)

Basment_fit=loess(log(train_22$SalePrice+1)[which(train$GarageType=="Basment")]~train_22$GrLivArea[which
Basment_order=order(train_22$GrLivArea[which(train$GarageType=="Basment")])
Basment_pred=predict(Basment_fit)

Types_fit=loess(log(train_22$SalePrice+1)[which(train$GarageType=="2Types" )]~train_22$GrLivArea[which(
Types_order=order(train_22$GrLivArea[which(train$GarageType=="2Types")])
Types_pred=predict(Types_fit)

plot(train_22$GrLivArea,log(train_22$SalePrice+1),col="pink",pch=19,cex=0.5,main ="GarageType loess smo

lines(train_22$GrLivArea[which(train$GarageType=="Attchd")][Attchd_order],Attchd_pred[Attchd_order],col=

lines(train_22$GrLivArea[which(train$GarageType=="Detchd")][Detchd_order],Detchd_pred[Detchd_order],col=

lines(train_22$GrLivArea[which(train$GarageType=="BuiltIn")][BuiltIn_order],BuiltIn_pred[BuiltIn_order]

lines(train_22$GrLivArea[which(train$GarageType=="CarPort")][CarPort_order],CarPort_pred[CarPort_order]

lines(train_22$GrLivArea[which(train$GarageType=="Basment")][Basment_order],Basment_pred[Basment_order]

lines(train_22$GrLivArea[which(train$GarageType=="2Types")][Types_order],Types_pred[Types_order],col='gr

legend("topright", legend=c('Attchd','Detchd','BuiltIn','CarPort','Basment','2Types'),
       col=c('steelblue','black', 'red','yellow','green','grey'), lty=1:2, cex=0.8)
```

```
library("rgl", lib.loc="/Library/Frameworks/R.framework/Versions/3.5/Resources/library")
source("/Users/yiming/Desktop/stat844/project/scatter3d.R")
scatter3d(log(train_22$SalePrice+1) ~ train_22$OverallQual+train_22$GrLivArea ,fit=c('linear',"interacti
snapshot3d("ozone3interaction.png")
```

```
library("rgl", lib.loc="/Library/Frameworks/R.framework/Versions/3.5/Resources/library")
source("/Users/yiming/Desktop/stat844/project/scatter3d.R")
scatter3d(log(train_22$SalePrice+1) ~ train_22$GrLivArea+train_22$OverallQual ,fit=c('linear',"interacti
snapshot3d("ozone3interaction1.png")
```

```
scatter3d(log(train_22$SalePrice+1) ~ train_22$OverallQual+train_22$GrLivArea ,fit=c("loess"),df.loess=3
```

```
#Randomly shuffle the data
train_new1<-train_new[sample(nrow(train_new)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(train_new1)),breaks=4,labels=FALSE)
#Perform 10 fold cross validation
lasso_rmse=c()
cv_lasso_names=c()
for(i in 1:4){
#Segement your data by fold using the which() function
testIndexes <- which(folds==i,arr.ind=TRUE)
testData <- train_new[testIndexes, ]
trainData <- train_new[-testIndexes, ]
#Use the test and train data partitions however you desire...
```

```
set.seed(123)
cv_lasso = cv.glmnet(as.matrix(trainData[, -173]),trainData[, 173])
preds <- predict(cv_lasso, newx = as.matrix(testData[, -173]), s = "lambda.min")

myCoefs <- coef(cv_lasso, s="lambda.min");
lasso_rmse=c(lasso_rmse,rmse(testData$SalePrice, preds))
cv_lasso_names=c(cv_lasso_names,myCoefs@Dimnames[[1]][which(myCoefs != 0 ) ][2:30])
}
```

```
mean(lasso_rmse)
names(sort(table(cv_lasso_names),decreasing = TRUE)[1:10])
barplot(sort(table(cv_lasso_names),decreasing = TRUE)[1:10],main="Important features_Lasso",xlab="Freqe
```

```
library(randomForest)
#Randomly shuffle the data
train_new1<-train_new[sample(nrow(train_new)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(train_new1)),breaks=5,labels=FALSE)
#Perform 10 fold cross validation
rf_rmse=c()

for(i in 1:5){
#Segement your data by fold using the which() function
testIndexes <- which(folds==i,arr.ind=TRUE)
testData <- train_new[testIndexes, ]
trainData <- train_new[-testIndexes, ]

#####train data renames
colnames(trainData)[colnames(trainData)=='MSZoningC (all)'] <- "MSZoningCall"
colnames(trainData)[colnames(trainData)=='RoofMatlTar&Grv'] <- "RoofMatlTarGrv"
colnames(trainData)[colnames(trainData)=='Exterior1stWd Sdng'] <- 'Exterior1stWdSdng'


#####test data renames
colnames(testData)[colnames(testData)=='MSZoningC (all)'] <- "MSZoningCall"
colnames(testData)[colnames(testData)=='RoofMatlTar&Grv'] <- "RoofMatlTarGrv"
colnames(testData)[colnames(testData)=='Exterior1stWd Sdng'] <- 'Exterior1stWdSdng'

#Use the test and train data partitions however you desire...
tr.rf<-randomForest(trainData$SalePrice1~ .,data=trainData)

rf_rmse=c(rf_rmse,rmse(testData$SalePrice1, predict(tr.rf, testData)))

}
```

```
mean(rf_rmse)
```

```
control<-trainControl(method="cv", number=5)
tunegrid <- expand.grid(.mtry=c(sqrt(ncol(x))))
modellist <- list()
for (ntree in c(100,200,500,1000)) {

    fit <- train(x=train_new[,-173], y=train_new[,173], method="rf",  tuneGrid=tunegrid, trControl=conti
    key <- toString(ntree)
    modellist[[key]] <- fit
```

```
}
# compare results
results <- resamples(modellist)
dotplot(results)
```

```
summary(results)
```

```
library(randomForest)
#Randomly shuffle the data
train_new1<-train_new[sample(nrow(train_new)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(train_new1)),breaks=5,labels=FALSE)
#Perform 10 fold cross validation
rf_rmse1=c()

for(i in 1:5){
#Segement your data by fold using the which() function
testIndexes <- which(folds==i,arr.ind=TRUE)
testData <- train_new[testIndexes, ]
trainData <- train_new[-testIndexes, ]

#####train data renames
colnames(trainData)[colnames(trainData)=='MSZoningC (all)'] <- "MSZoningCall"
colnames(trainData)[colnames(trainData)=='RoofMatlTar&Grv'] <- "RoofMatlTarGrv"
colnames(trainData)[colnames(trainData)=='Exterior1stWd Sdng'] <- 'Exterior1stWdSdng'


#####test data renames
colnames(testData)[colnames(testData)=='MSZoningC (all)'] <- "MSZoningCall"
colnames(testData)[colnames(testData)=='RoofMatlTar&Grv'] <- "RoofMatlTarGrv"
colnames(testData)[colnames(testData)=='Exterior1stWd Sdng'] <- 'Exterior1stWdSdng'

#Use the test and train data partitions however you desire...
tr.rf1<-randomForest(trainData$SalePrice1~ .,data=trainData,ntree=1000)

rf_rmse1=c(rf_rmse1,rmse(testData$SalePrice1, predict(tr.rf1, testData)))

}
```

```
mean(rf_rmse1)
```

```
varImpPlot(tr.rf1,col='red',lwd=5, sort=TRUE, n.var=min(30, nrow(x$importance)),
           type=, class=NULL, scale=TRUE,
           main='Randomforest importance of varibles')
```

```
library(gbm)
#Randomly shuffle the data
train_new1<-train_new[sample(nrow(train_new)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(train_new1)),breaks=5,labels=FALSE)
#Perform 10 fold cross validation
gbm_rmse=c()

for(i in 1:5){
#Segement your data by fold using the which() function
```

```
testIndexes <- which(folds==i,arr.ind=TRUE)
testData <- train_new[testIndexes, ]
trainData <- train_new[-testIndexes, ]

#####train data renames
colnames(trainData)[colnames(trainData)=='MSZoningC (all)'] <- "MSZoningCall"
colnames(trainData)[colnames(trainData)=='RoofMatlTar&Grv'] <- "RoofMatlTarGrv"
colnames(trainData)[colnames(trainData)=='Exterior1stWd Sdng'] <- 'Exterior1stWdSdng'


#####test data renames
colnames(testData)[colnames(testData)=='MSZoningC (all)'] <- "MSZoningCall"
colnames(testData)[colnames(testData)=='RoofMatlTar&Grv'] <- "RoofMatlTarGrv"
colnames(testData)[colnames(testData)=='Exterior1stWd Sdng'] <- 'Exterior1stWdSdng'

#Use the test and train data partitions however you desire...
gbmf=gbm(SalePrice1 ~ .,data=trainData, distribution="gaussian", n.trees=100, interaction.depth=1,n.min

prediction<-predict(gbmf, testData)
gbm_rmse=c(gbm_rmse,rmse(testData$SalePrice1, predict(gbmf, testData)))
}
```

```
mean(gbm_rmse)
```

```
gbm_control <-trainControl(method="cv", number=5)
gbm_grid = expand.grid(
n.trees = c(100,500,800),
interaction.depth = c(1,3,5),
n.minobsinnode = c(5,10,20), shrinkage = c(0.01,0.1,0.3)
)


gbm_caret=train(x=train_new[,-173], y=train_new[,173], method='gbm', trControl= gbm_control, tuneGrid=g
```

```
gbm_caret$bestTune
```

```
library(gbm)
#Randomly shuffle the data
train_new1<-train_new[sample(nrow(train_new)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(train_new1)),breaks=5,labels=FALSE)
#Perform 10 fold cross validation
gbm_rmse1=c()

for(i in 1:5){
#Segement your data by fold using the which() function
testIndexes <- which(folds==i,arr.ind=TRUE)
testData <- train_new[testIndexes, ]
trainData <- train_new[-testIndexes, ]

#####train data renames
colnames(trainData)[colnames(trainData)=='MSZoningC (all)'] <- "MSZoningCall"
colnames(trainData)[colnames(trainData)=='RoofMatlTar&Grv'] <- "RoofMatlTarGrv"
colnames(trainData)[colnames(trainData)=='Exterior1stWd Sdng'] <- 'Exterior1stWdSdng'
```

```
#####test data renames
colnames(testData)[colnames(testData)=='MSZoningC (all)'] <- "MSZoningCall"
colnames(testData)[colnames(testData)=='RoofMatlTar&Grv'] <- "RoofMatlTarGrv"
colnames(testData)[colnames(testData)=='Exterior1stWd Sdng'] <- 'Exterior1stWdSdng'

#Use the test and train data partitions however you desire...
gbmf_best=gbm(SalePrice1 ~ .,data=trainData, distribution="gaussian", n.trees=800, interaction.depth=5,r

prediction1<-predict(gbmf_best, testData)
gbm_rmse1=c(gbm_rmse1,rmse(testData$SalePrice1, prediction1))
}
```

```
mean(gbm_rmse1)
```

```
par(cex.lab=1.2,las=2,cex.axis=0.3)
summary(gbmf_best,main='Gradient boost importance of variables')
```

```
library(xgboost)
#Randomly shuffle the data
train_new1<-train_new[sample(nrow(train_new)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(train_new1)),breaks=5,labels=FALSE)
#Perform 10 fold cross validation
xgb_rmse=c()

for(i in 1:5){
#Segement your data by fold using the which() function
testIndexes <- which(folds==i,arr.ind=TRUE)
testData <- train_new[testIndexes, ]
trainData <- train_new[-testIndexes, ]

xgb_train <- xgb.DMatrix(data = as.matrix(trainData[,-173]), label= trainData$SalePrice1)
xgb_test <- xgb.DMatrix(data = as.matrix(testData[,-173]))

#Use the test and train data partitions however you desire...
xgboost_fit=xgb.train(data = xgb_train,nrounds =100)

prediction<-predict(xgboost_fit, xgb_test )
xgb_rmse=c(xgb_rmse,rmse(testData$SalePrice1, prediction))
}
```

```
mean(xgb_rmse)
```

```
xgb_control <-trainControl(method="cv", number=5)
xgb_grid = expand.grid(
nrounds = c(50,100,200,300),
eta = c(0.1,0.2,0.3,0.4,0.5),
max_depth = c(2,3,4,5,6),
gamma = 0,
colsample_bytree=1,
min_child_weight=c(1,2,3),
subsample=1
)
```

```
xgb_caret <- train(x=train_new[,-173], y=train_new[,173], method='xgbTree', trControl= xgb_control, tune
```

```
xgb_caret$bestTune
```

```
best_param<-list(
        booster = "gbtree",
        eta=0.2, #default = 0.3
        gamma=0,
        max_depth=2, #default=6
        min_child_weight=3, #default=1
        subsample=1,
        colsample_bytree=1
)
```

```
library(xgboost)
#Randomly shuffle the data
train_new1<-train_new[sample(nrow(train_new)),]
#Create 10 equally size folds
folds <- cut(seq(1,nrow(train_new1)),breaks=5,labels=FALSE)
#Perform 10 fold cross validation
xgb_rmse1=c()

for(i in 1:5){
#Segement your data by fold using the which() function
testIndexes <- which(folds==i,arr.ind=TRUE)
testData <- train_new[testIndexes, ]
trainData <- train_new[-testIndexes, ]

xgb_train <- xgb.DMatrix(data = as.matrix(trainData[,-173]), label= trainData$SalePrice1)
xgb_test <- xgb.DMatrix(data = as.matrix(testData[,-173]))

#Use the test and train data partitions however you desire...
xgbest_fit=xgb.train(data = xgb_train, params=best_param, nrounds = 300)

prediction<-predict(xgbest_fit, xgb_test)
xgb_rmse1=c(xgb_rmse1,rmse(testData$SalePrice1, prediction))
}
```

```
mean(xgb_rmse1)
```

```
library(Ckmeans.1d.dp) #required for ggplot clustering
mat <- xgb.importance (feature_names = colnames(xgb_train),model = xgboost_fit)
xgb.ggplot.importance(importance_matrix = mat[1:20], rel_to_first = TRUE)
```

```
###linear
mean(linear_rmse)
names(sort(table(important_names),decreasing = TRUE)[1:5])
######lasso
mean(lasso_rmse)
names(sort(table(cv_lasso_names),decreasing = TRUE)[1:5])
######rf
mean(rf_rmse)
rf_names=c('OverallQual','GrLivArea','ExterQual','ExterQual','TotalBsmtSF')
#####gbm
mean(gbm_rmse)
```

```
gbm_names=c('OverallQual','GrLivArea','TotalBsmtSF','GarageArea','KitchenQual')
####xgboost
mean(xgb_rmse)
xg_names=c('OverallQual','GrLivArea','TotalBsmtSF','GarageFinish','X1stFlrSF')
```

```
DT = data.table(
  ID = c("Linear_Regression","Lasso","Randomforest","Gradient Boosting_Machine","Xgboost"),
  RMSE_default = c(mean(linear_rmse),mean(lasso_rmse),mean(rf_rmse),mean(gbm_rmse),mean(xgb_rmse)),
  RMSE_bestmodel= c(mean(linear_rmse),mean(lasso_rmse),mean(rf_rmse1),mean(gbm_rmse1),mean(xgb_rmse1))
)

DT
```

```
rf_names=c('OverallQual','GrLivArea','ExterQual','ExterQual','TotalBsmtSF')
gbm_names=c('OverallQual','GrLivArea','TotalBsmtSF','GarageArea','KitchenQual')
xg_names=c('OverallQual','GrLivArea','TotalBsmtSF','GarageFinish','X1stFlrSF')
cloud_names=c(names(sort(table(important_names),decreasing = TRUE)[1:30]),names(sort(table(cv_lasso_name
cloud_data=as.data.frame(table(cloud_names))
```

```
library(wordcloud2)
names(cloud_data)[names(cloud_data)=="cloud_names"] <- "word"
names(cloud_data)[names(cloud_data)=="cloud_names"] <- "freq"
wordcloud2(data=cloud_data,color = "random-light", size = .5,ellipticity = 0.6, backgroundColor = "white
```

```
letterCloud(data=cloud_data,color = "random-light", size = 0.21,ellipticity = 0.6, backgroundColor = "bl
```