ERG Group Project: House Prices Prediction

Sun Yan 119020045 He Shuqing 119010096 Han Yi 119020013 Zhao Qianfan 119020562

December 23, 2020

Abstract

House prices can be affected by many factors from type of utilities available, style of dwelling to rating of basement finished area and fireplace quality. Our goal is to predict the likely house prices according to some potential influencing factors. After analyzing all of the explanatory data, we conducted feature engineering by combining and transforming variables. Next, we imputed omitted variables, fixed skewness of independent and dependent variables, removed features high-correlated or with near-zero variance and deleted outliers. In the process of model building, we tried different kinds of regression models: OLS-based, tree-based and distance-based. Finally we selected six models, which are forward step-wise selection model, PCR, Ridge&Lasso, GAM, Random Forest and SVM. To obtain an excellent performance, we decided to ensemble all the mdoels by assigning optimal weights to each one. Eventually, we used the well-tuned ensembled model to predict the test data and obtained a RMSLE of 0.12340, which ranked 18% on Kaggle (by December 2020).

Introduction

With a size of $9.6 trillion in 2019, the real estate market is one of the hugest market in the world (MSCI, 2019). Residential housing prices in this industry fluctuate on a day-to-day basis, providing critical information about the value of houses, as well as buyers' and sellers' sentiment. Given the dynamic nature of the real estate market, a regression model able to predict housing prices could play a significant role. The goal of this model is two-fold. First and most importantly, it should accurately predict a house's price with necessary information. Secondly, the model should generate some insights on the influential factors of housing prices; in other words, the model should reveal which characteristics of a house are mostly related to its price. With such prediction and inference capabilities, a regression model could contribute to the valuation decisions of real estate dealers and appraisers. It could also provide guidance to home owners and house builders on how to increase their houses' value.

Materials and Methods

Only part of our codes are shown here, more can refer to our github: Code

The codes for the preprocess part are packed into the github file "preprocessing/preprocess.R".

```
source('~/Documents/Github/ERG-Project/preprocessing/preprocess.R')
train <- read.csv('~/Documents/Github/ERG-Project/data/train_full.csv')[,-1]
```

Data

The research built its model based on the training set of the Ames Housing Dataset available on Kaggle (2020). The Ames Housing Dataset, prepared by Dean De Cock (2011), describes the sale of 2930 residential

properties in Ames, Iowa, U.S. from 2006 to 2010. It is usually considered as the most expanded and modernized version of the classic Boston Housing Dataset and was therefore chosen for this study.

The training set contains 1460 observations and 79 predictors, with "Sale Price" as the response variable. Among the 79 predictors, 23 are nominal variables, 23 are ordinal, 13 discrete, and 20 continuous.

Features were divided into four categories for later manipulation.

环境		配置		属性		评分	
2 MSZoning	地区划分	34 BsmtFinSF1	地下室的完工大小	4 LotArea	房屋占地的面积	17 OverallQual	评估房子的整体材料和装饰
3 LotFrontage	和街道的直线距离			7 LotShape	房屋占地的形状	18 OverallCond	对房子的整体状况进行评估
		37 BsmtUnfSF	地下室的未完工大小	15 BldgType	房屋类型（独户、双户、复式、联排别墅内部、两端）	28 ExterCond	评估外部材料的状态
10 LotConfig	与周围房屋街道相对位置	38 TotalBsmtSF	地下室总面积	16 HouseStyle	房屋风格（一层、两层）		
				20 YearRemodAdd	改型日期(如无改型或加建，则与建造日期相同)		
		42 Electrical	电力系统	21 RoofStyle	屋顶风格（平、三角、双重斜坡）	33 BsmtFinType1	地下室完工面积等级
13 Condition1	和周边路段的毗邻度（街道铁路公园）	47 BsmtFullBath	地下室全浴室				
		48 BsmtHalfBath	地下室半浴室（只含便池与洗手盆）	23 Exterior1st	房屋外盖	40 HeatingQC	供热质量和条件
		49 FullBath	地面以上全浴室	24 Exterior2nd		53 KitchenQual	评估厨房质量
		50 HalfBath	地面以上半浴室	25 MasVnrType	表层砌体类型	57 FireplaceQu	评估壁炉质量
		51 Bedroom	地面以上卧室	26 MasVnrArea	砌体贴面面积，平方英尺	63 GarageQual	评估车库质量
				29 Foundation	基础类型（砖瓦、石头、木质、混凝土、煤渣砖）	64 GarageCond	评估车库状态
		54 TotRmsAbvGrd	地面以上房间（不含浴室）	43 1stFlrSF	一楼面积		
		56 Fireplaces	壁炉数量	44 2ndFlrSF	二楼面积	73 Fence	评估栅栏质量
		58 GarageType	车库位置				
		59 GarageYrBl	车库建造的年份	46 GrLivArea	地面以上生活区面积		
				76 MoSold	售出月份		
		65 PavedDrive	铺面车道	77 YrSold	售出年份		
				78 SaleType	销售类型（责任房契、约定利息...）		
				79 SaleCondition	销售状况（普通、贸易、抵押...）		

Based on Exploratory Data Analysis (EDA), the research conducted following preprocessing procedures.

Feature Engineering

The dataset includes some similar features. For instance, both the predictor "Overall Quality" and "Overall Condition" depict the overall state of a house. Combining these overlapping features could tune down the dataset's noise. In light of this consideration, this study combined the feature "Overall Quality" and "Overall Condition" into a new feature. Same procedure was also applied to "Garage Quality" & "Garage Condition," "Exterior Quality" & "Exterior Condition," and "Basement Quality" & "Basement Condition."

Besides combining overlapping features, the study also transformed some ordinal features into numeric features. For example, the predictor "Basement Exposure" contains four categories: "No,""Mn,""Av," and "Gd," standing for no exposure, minimum exposure, average exposure, and good exposure, respectively; the study has encoded "No" as 1, "Mn" as 2, "Av" as 3, etc. In this way, the information regarding the order of various values were transformed into forms that could be recognized by machine learning algorithms.

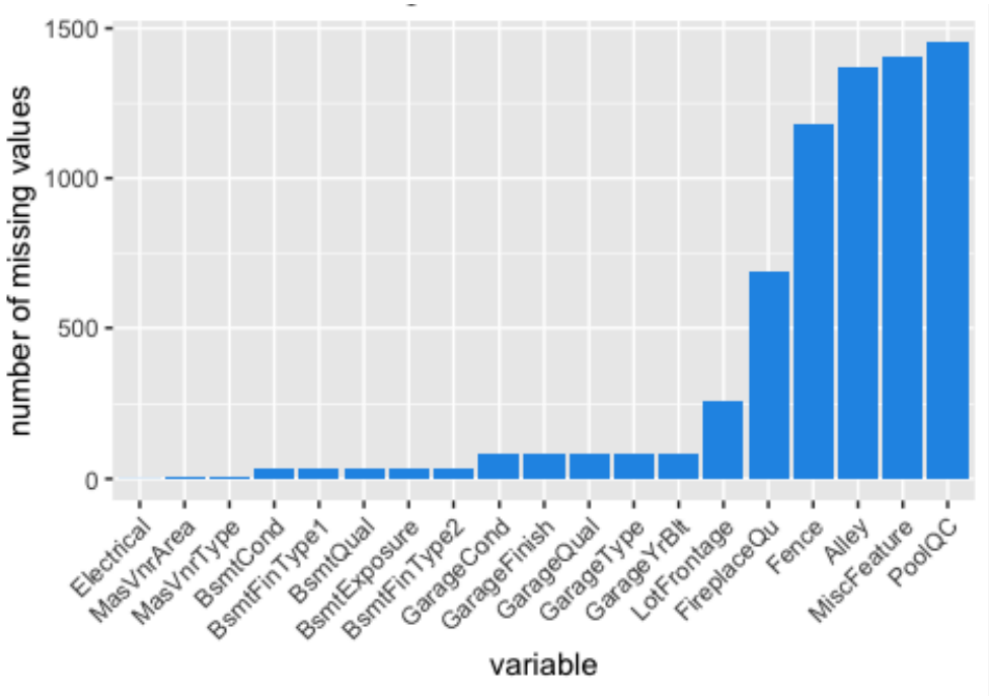
Part of the codes are shown:

```
# Kitchen Quality
x$KitchenQual[x$KitchenQual == 'Ex'] = 4
x$KitchenQual[x$KitchenQual == 'Gd'] = 3
x$KitchenQual[x$KitchenQual == 'TA'] = 2
x$KitchenQual[x$KitchenQual == 'Fa'] = 1
x$KitchenQual[x$KitchenQual == 'Po'] = 0
x$KitchenQual <- as.numeric(x$KitchenQual)

# Fireplace Quality
x$FireplaceQu[x$FireplaceQu == 'Ex'] = 5
x$FireplaceQu[x$FireplaceQu == 'Gd'] = 4
x$FireplaceQu[x$FireplaceQu == 'TA'] = 3
x$FireplaceQu[x$FireplaceQu == 'Fa'] = 2
x$FireplaceQu[x$FireplaceQu == 'Po'] = 1
x$FireplaceQu[is.na(x$FireplaceQu)] = 0
x$FireplaceQu <- as.numeric(x$FireplaceQu)
```

Missing Value Imputation

In the training dataset, 19 features contain "NA." However, a closer observation revealed that only 4 features ("Lot Frontage,""Masonry Veneer Type,""Masonry Veneer Area," and "Electrical") have true missing values. For the rest 15 features, "NA" simply indicates the absence of the corresponding metric. For example, an "NA" in the feature "Pool Quality" means that this house doesn't have a pool, rather than this value is missing.

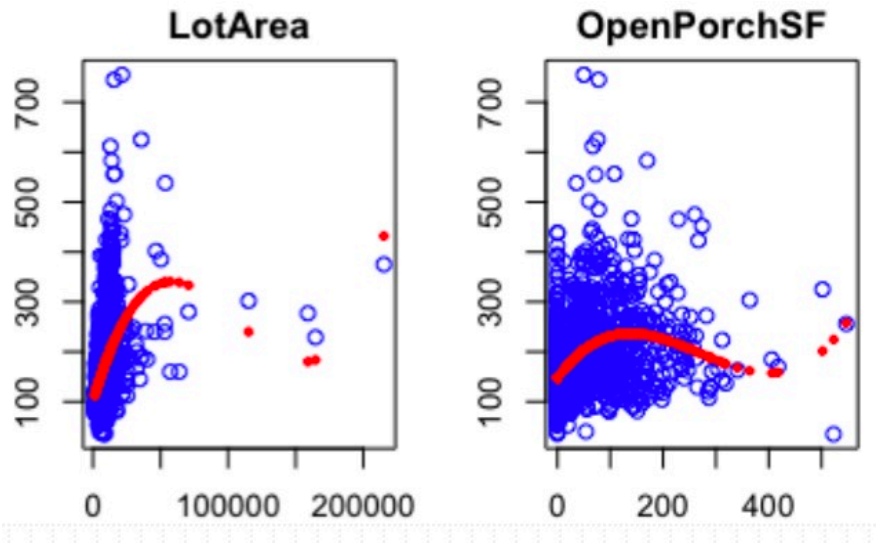
The variables with missing values were:



For the 4 features that truly contain missing values, the study chose to impute those missing values since 1) missing values could reduce the statistical power of the dataset and lead to models with higher bias (Kang, 2013), 2) some models couldn't work with missing values, and 3) in this case, the number of missing values is small and therefore easy to impute. For categorical missing values, the study adopted mode imputation: replace categorical missing values by the mode of the corresponding column. For numeric missing values, the study adopted KNN imputation: replace numeric missing values by the mean of the nearest K neighbors in the whole feature space. The hyperparameter K was set to be the square root of the number of observations, i.e. $K = 38 \approx \sqrt{1460}$ (Jonsson & Wohlin, 2004).

Feature Scaling

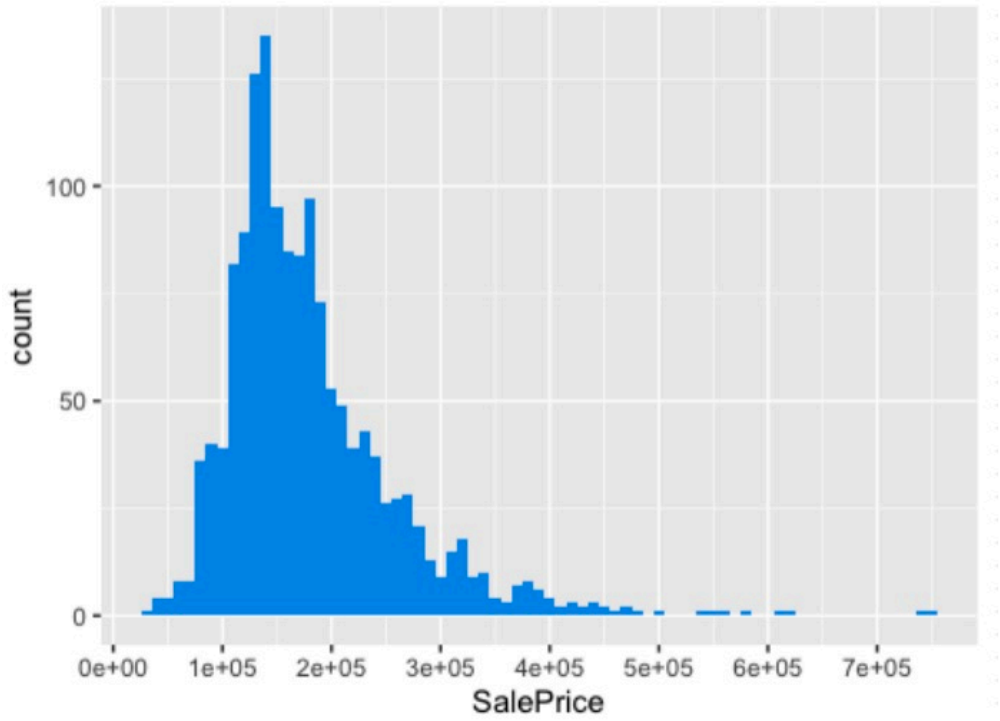
The data were scattered and hard to manipulate before scaling.



The study applied z-score standardization to the numeric features, so that their mean became 0, and standard deviation equaled 1. The process is automatically undertaken by `preProcess()` in `Caret`.

Fix the Skewness of the Response Variable

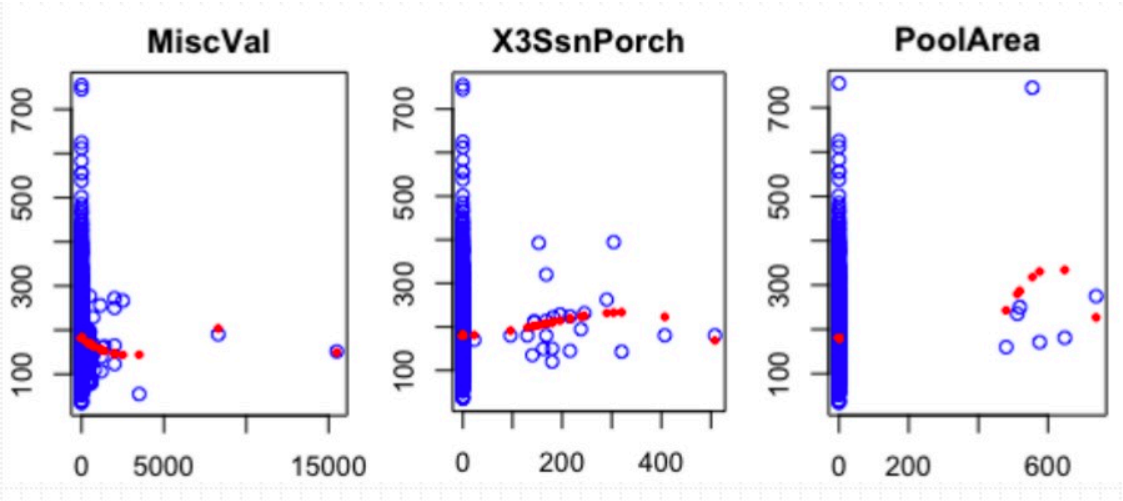
The response variable "Sale Price" is right-skewed.



To avoid volatile and biased prediction of house price, the study applied logarithmic transformation to fix the skewness.

Delete Zero-Variance Features

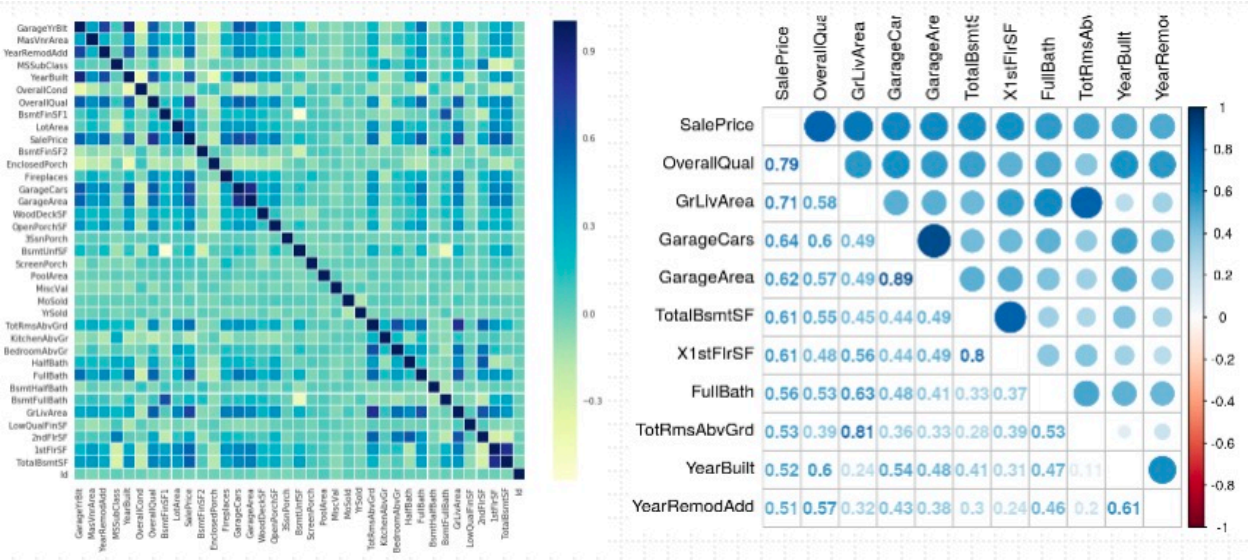
For some features in the training dataset, most observations are restricted to only one same value or category. These features are therefore redundant since they provide little information to the dataset while unnecessarily increase its variance.



The study chose the near-zero-variance function in the Caret package to detect those redundant features. 19 features who has near zero variances were therefore removed.

Reduce Collinearity

According to the correlation matrix, some features in the dataset exhibit high collinearity, posing threats to model accuracy and interpretability.



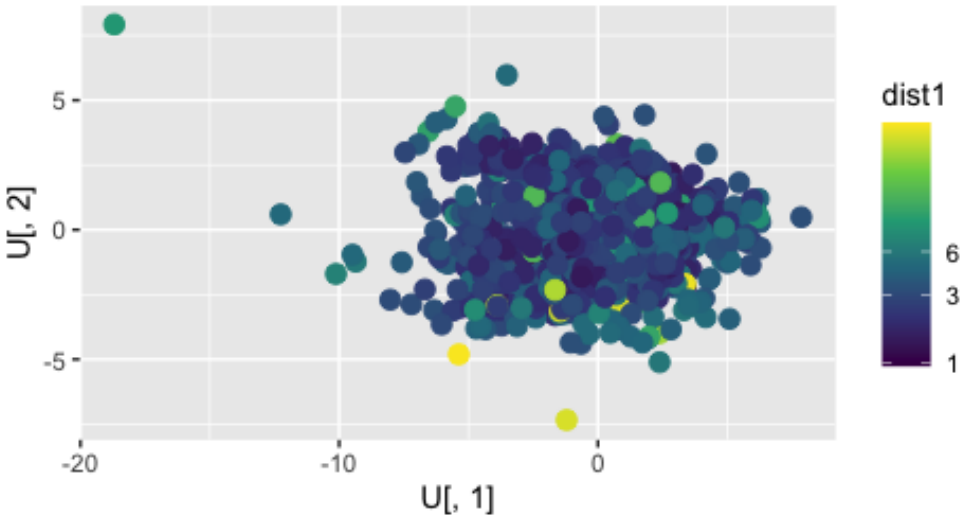
The study focused on two types of collinearity, namely 1) the collinearity between numeric features and numeric features and 2) the collinearity between categorical features and categorical features. Instead of the default Pearson index in the Caret package, the study used the more robust Spearman index to detect the first type of collinearity. For the detection of the second type of collinearity, the Cramer's V ("Cramer's V," n.d.) index in the Caret package was used. A total of 8 features were removed in this way.

Parts of the codes are shown:

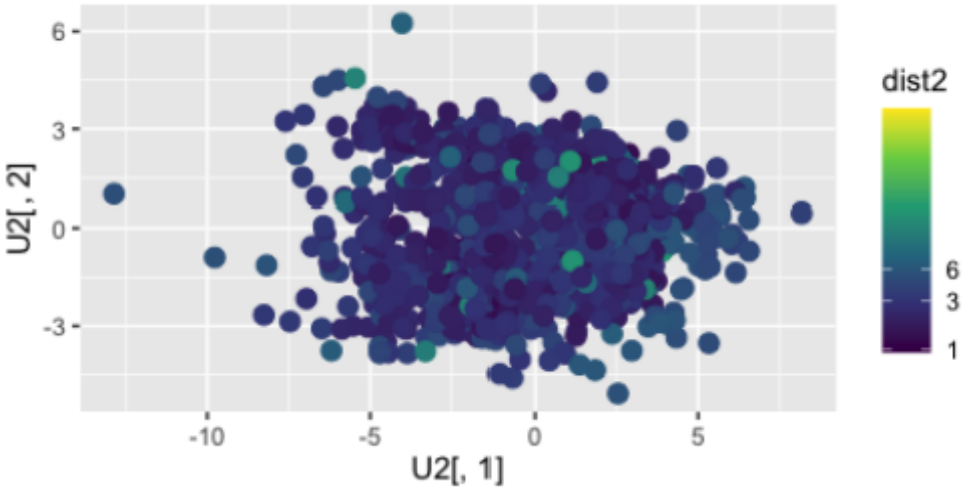
```
preprocess.corNum2Num <- function(x,index, cut = .85, cormethod = 'spearman',...){
  class(x)<- 'data.frame'
  num <- x[,index]
  descrCor <-  cor(num, method = cormethod)
  highlyCorDescr <- findCorrelation(descrCor, cutoff = cut)
  remaining <- num[,-highlyCorDescr]
  print('Preprocess: Correlation between numeric features')
  print(paste('Features removed:',names(x)[highlyCorDescr]))
  return(remaining)
}

preprocess.corCat2Cat <-function(x, index, remove = 5,...){
  class(x)<- 'data.frame'
  cramer_ind = c()
  for (i in 1:length(index)){
    for (j in 1:length(index)){
      tab = table(x[,index[i]], x[,index[j]])
      cramer_ind = append(cramer_ind, cramer(tab))
    }
  }
  cramer_ma = matrix(cramer_ind, ncol = length(index))

  index_c = c()
  for (i in 1:length(index)){
    index_c = append(index_c, mean(cramer_ma[,i]))
  }
  sort_c = sort(index_c, index.return = T, decreasing = T)
  remainCat <- index[-c(sort_c$ix[2:6])]
  print('Preprocess: Correlation between categorical features')
  print(paste('Features removed:',names(x)[index[c(sort_c$ix[2:6])]]))
  return(x[,remainCat])
}
```

Remove Outliers via PCA

The large number of features in this dataset poses a challenge to detecting outliers.



Considering the impracticability of detecting outliers by each feature, the study instead applied Principle Component Analysis (PCA) to compress the feature space to two dimensions. After PCA, the study computed the index $\frac{|x-median(x)|}{mad(x)}$ of each data point, where "mad" represents Median Absolute Deviation. Data points whose indices were higher than 6 were classified as outliers and removed.



Model

Forward Stepwise Selection

Forward stepwise selection is a subset selection method of fitting linear models. Beginning with a null model, in each step it adds one variable whose inclusion gives the most statistically significant improvement of the fit and repeats this process until no more improvement occurs. The statistically significant improvement level is judged according to the RSS,adjusted R^2, Cp or BIC. Since RSS always favors the full model, it's better to choose Cp, BIC or adjusted R^2 as a criterion.

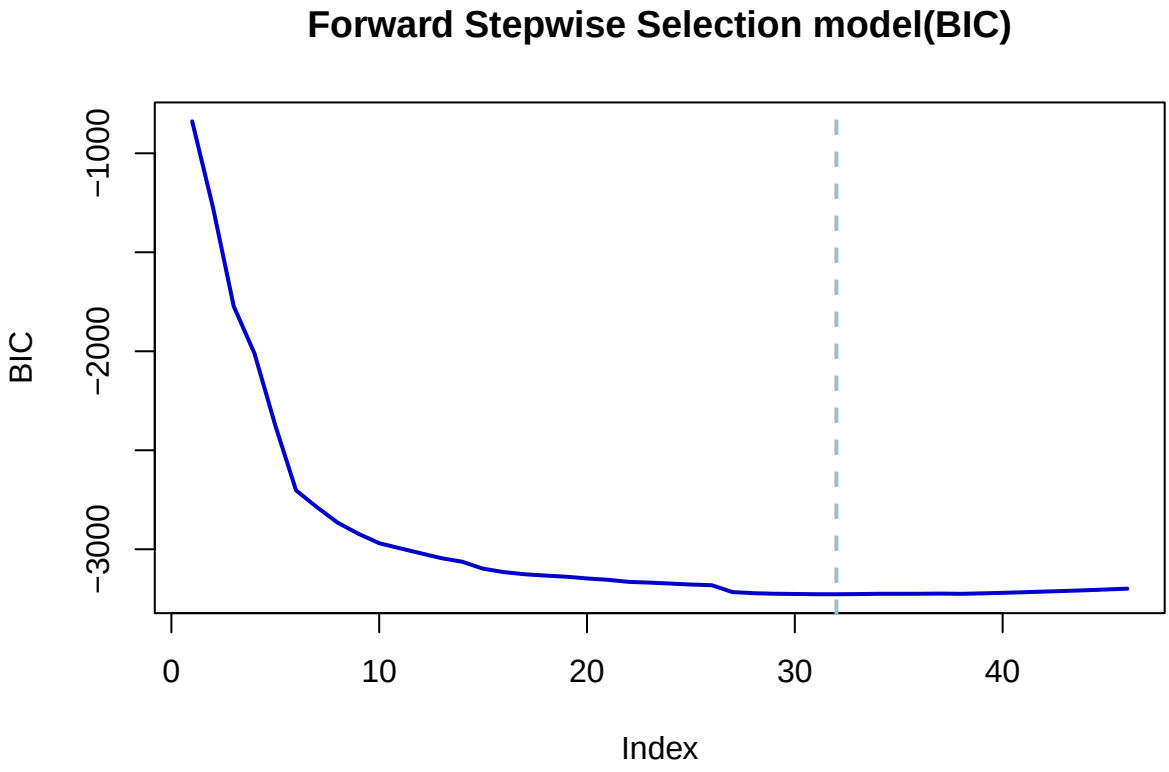
Forward stepwise selection has great advantage in removing unimportant variables and fit the model according to the importance of variables in each step.

The forward stepwise selection was achieved using library "leaps". The tuning parameter nvmax was set to be the original total number of variables(= 46) and served to be the maximal number of predictors to corporate in the model. The model with the minimal BIC was chosen.

```
regfit.forward = regsubsets(Y~.,data = train, nvmax = 46, method = "forward")
reg.summary <- summary(regfit.forward)
min = which.min(reg.summary$bic)
coef <- coef(regfit.forward, min)


model.fwd <- function(x){
  matrix <- model.matrix(Y~., data =x)[,names(coef)]
  pred <- matrix%*%coef
  return(pred)
}
```

According to BIC, 32 variables were selected.

```
plot(reg.summary$bic, col = 'blue3',
     xlab = 'Index',
     ylab = "BIC",
     lwd = 2, type = 'l',
     main = 'Forward Stepwise Selection model(BIC)')
abline(v = min, lwd = 2, lty = 2, col = 'lightblue3')
text(34, 500, '32')
```



They are

```
reg.index <- names(sort(colSums(ifelse(reg.summary$outmat[1:32,] == '*',1,0)), decreasing = T)[1:32])
names(reg.summary$outmat[32,reg.index])
```

```
##  [1] "KitchenQual"          "TotalBsmtSF"          "X2ndFlrSF"
##  [4] "OverallQual"          "YearBuilt"            "X1stFlrSF"
##  [7] "GarageArea"           "Fireplaces"           "BsmtFinSF1"
## [10] "MSZoningRM"           "NeighborhoodCrawfor"  "LotArea"
## [13] "HeatingQC"            "SaleTypeNew"          "SaleConditionNormal"
## [16] "Condition1Norm"       "BldgTypeTwnhs"        "NeighborhoodBrkSide"
## [19] "NeighborhoodStoneBr"  "NeighborhoodNridgHt"  "NeighborhoodSomerst"
## [22] "MSZoningRL"           "NeighborhoodEdwards"  "BsmtFullBath"
## [25] "NeighborhoodMeadowV"  "MSZoningRH"           "MSZoningFV"
## [28] "BsmtExposure"         "GarageQual"           "SaleTypeConLD"
## [31] "MSSubClass"           "SaleConditionAdjLand"
```

Lasso Regression

Lasso regression is a type of linear regression using shrinkage method. It performs L1 regularization and adds a penalty term to lower the variance. Compared with simple regression model, lasso avoids overfitting of the data and also restricts the influence of predictor variables over the output variables by compressing their coefficients. Since lasso regression can shrink some coefficients to exactly 0, it can perform feature selection. As a result, the models generated from the lasso are generally easy to interpret.

Lasso regression was implemented through the train() function within caret package. The tuning parameter of the lasso was selected using cross-validation method.

```
fitControl <- trainControl(## 10-fold CV
                           method = "repeatedcv",
                           number = 10,
                           ## repeated ten times
                           repeats = 10)

set.seed(1)
L2Fit <- train(Y ~ ., data = train,
                 method = "glmnet",
                 trControl = fitControl)
```

Principle Component Regression

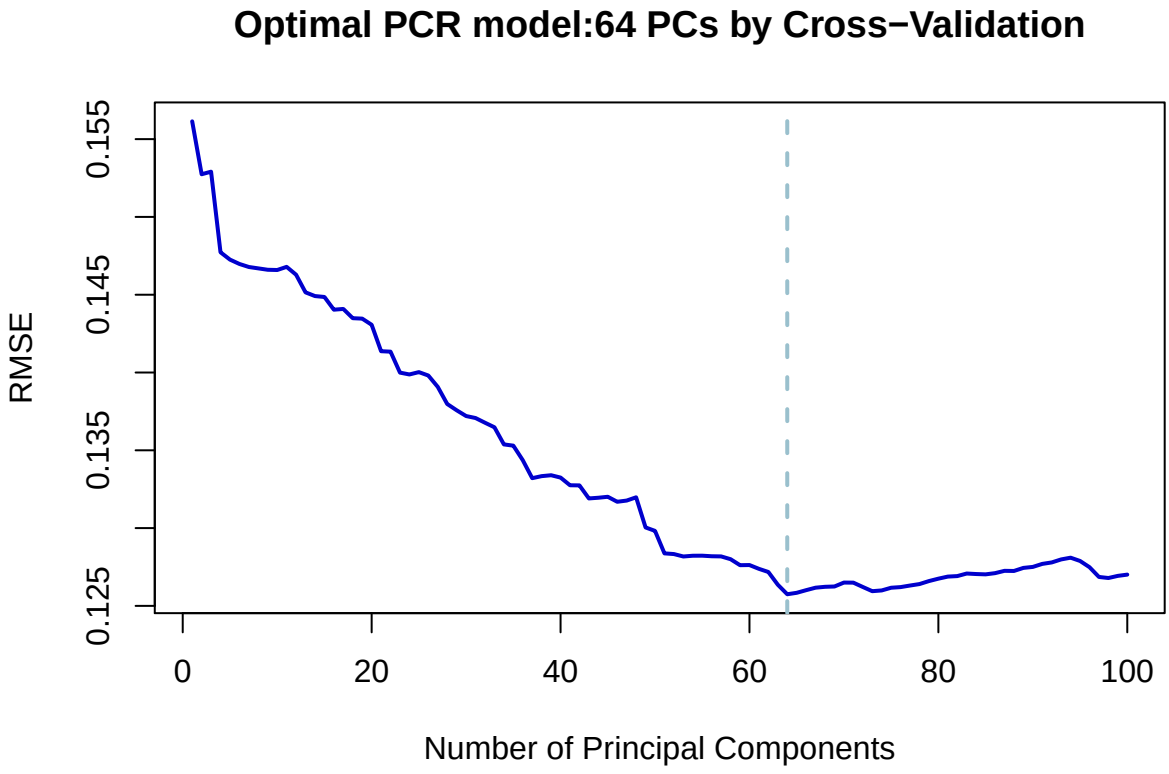
Principle component regression is based on principle component analysis. Instaed of regressing the dependent variable on the explanatory variables directly, the components of the explanatory variables are used as regressors. Since principle components with higher variances are selected, pcr can exclude some of the low-variance principal components in the regression step and result in dimension reduction.

The train() function in the caret package provides the method of principal component analysis. The argument tuneLength controls how many set of parameter values were evaluated and cross validation was applied to select the one with lowest (train) RMSE.

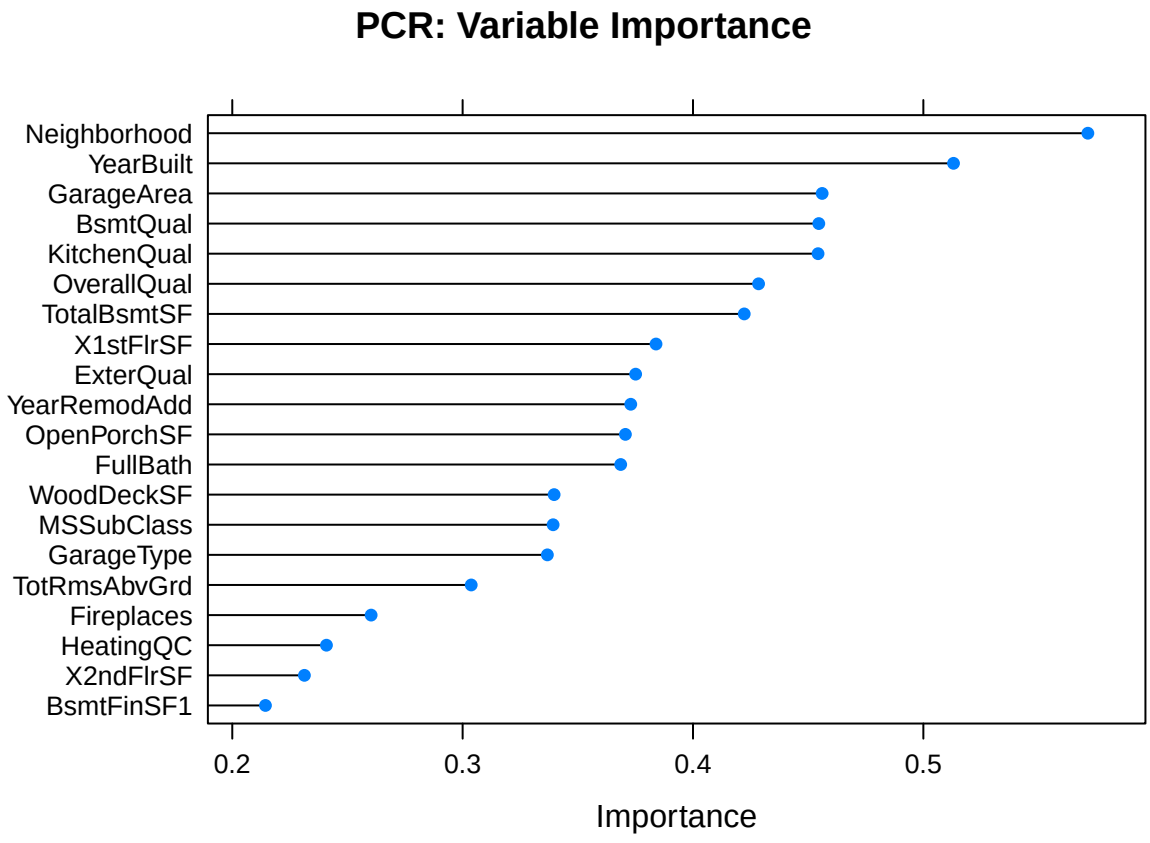
```
set.seed(1)
cv_model_pcr <- train(
  Y ~ .,
  data = train,
  method = "pcr",
  trControl = fitControl,
  tuneLength = 100
  )
```

Using cross-validation, 64 principal components were included into the final model. The variance importance rank according to PCR is shown in the graph.

```
plot(cv_model_pcr$results$ncomp,
     cv_model_pcr$results$RMSE, col = 'blue3',
     xlab = 'Number of Principal Components',
     ylab = 'RMSE',
     lwd = 2, type = 'l',
     main = 'Optimal PCR model:64 PCs by Cross-Validation')
abline(v = 64, lwd = 2, lty = 2, col = 'lightblue3')
```




```
pcrImp <- varImp(cv_model_pcr, scale = FALSE)
plot(pcrImp, top = 20, main = 'PCR: Variable Importance')
```



Random Forest

Random Forest is an improved version of bagging trees. During the splitting process, instead of choosing from all of the predictors, Random Forest limits the search to a random set of m of the p variables. In this way, the problem of tree correlation is minimized. The study chose Random Forest as one potential model since 1) Random Forest typically has outstanding prediction performance; 2) Random Forest could use Out-Of Bag (OOB) estimation to estimate the test error, so there is no need to sacrifice data for extra validation; 3) Random Forest could provide Variable Importance Measure, which is beneficial to the study's inference purpose.

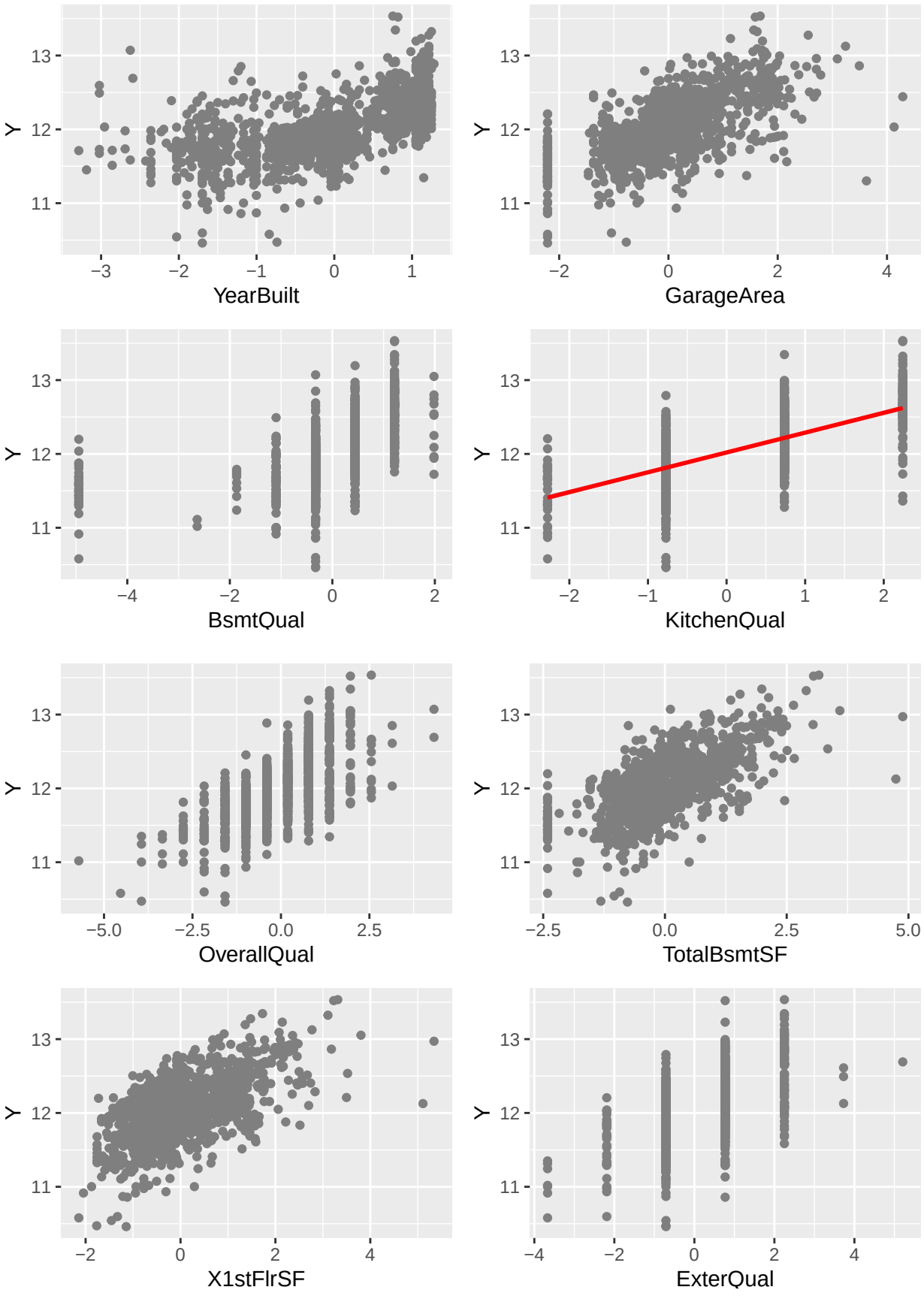
To achieve optimal performance, the study focused on tuning four hyperparameters of the Random Forest model, namely 1) "ntree:" number of trees, 2) "mtry:" the number of variables to randomly sample as candidates during each split, 3) "sampsize:" bootstrap sample size, and 4) "nodesize:" minimum number of observations within each leave. The study found that the error rate tends to be stable when the number of trees reaches 1000, so the value 1000 was chosen for the first hyperparameter. For the rest three hyperparameters, the study performed a grid search among potential candidate values. Note that the package "ranger" was used instead of the classic "randomForest" package, since "ranger" is over 6 times more computationally efficient than "randomForest" (Bradley, n.d.).

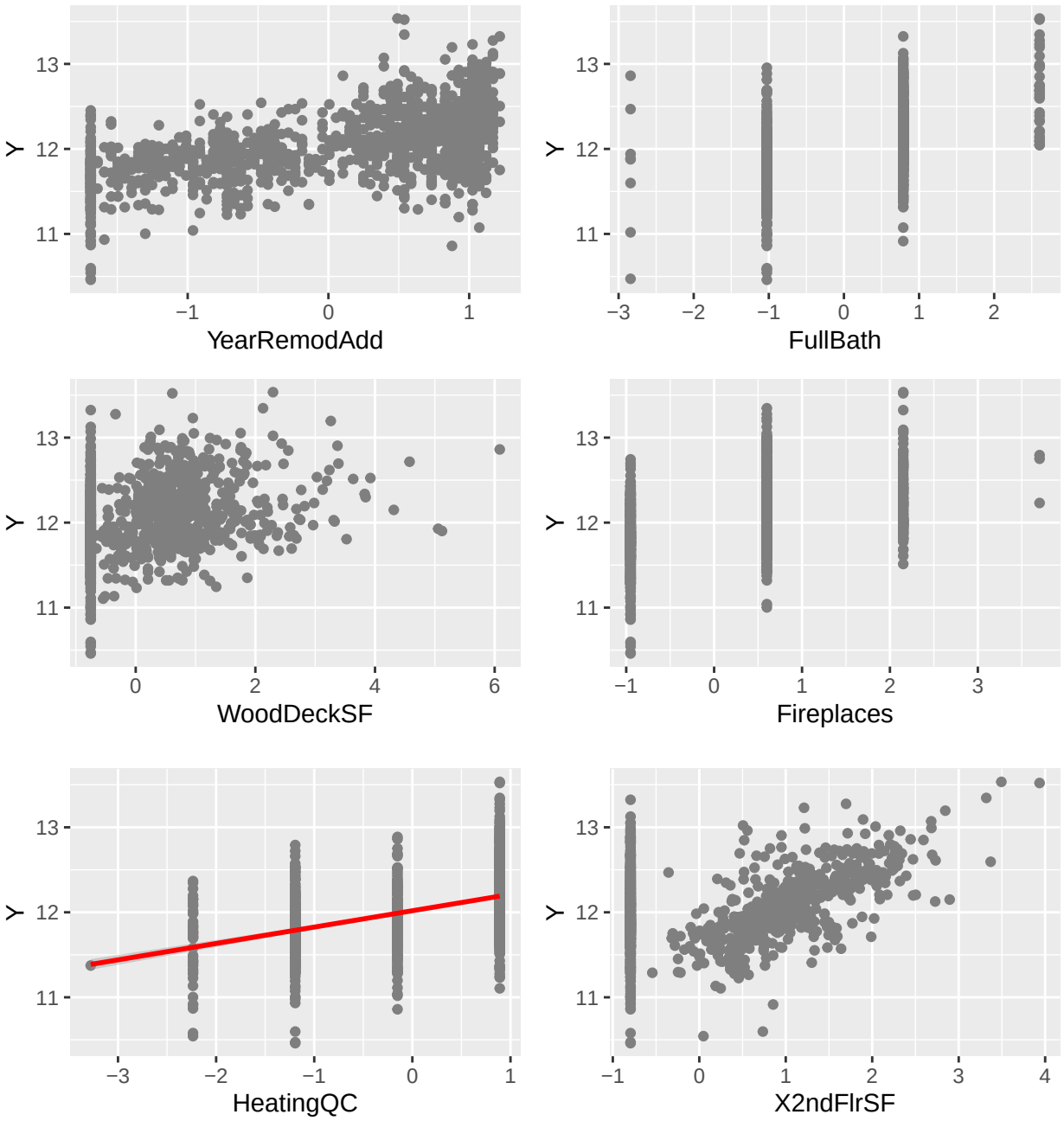
Among these 96 possible hyperparameter combinations, the study concludes that the best alternative is when mtry=20, sampsize=0.8, and nodesize=3. The random forest model was therefore trained with these hyperparameters.

GAM

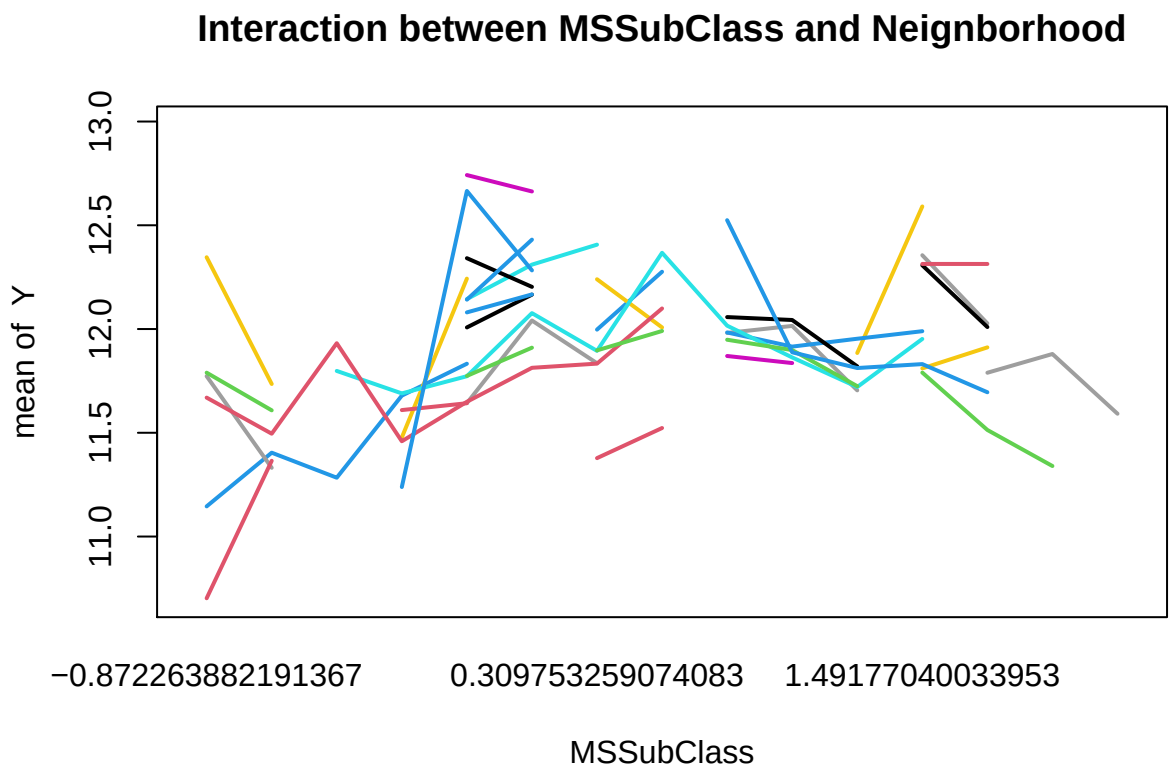
General Additive Model (GAM) is a flexible model that accounts for each feature's idiosyncratic relationship with the response. Previously, we obtained 20 important variables from PCR model.

Thereby, we decided to delve into the scatterplots of some important features and the dependent variable. By using the powerfull tool `geom_smooth` in the `ggplot2` package, we tried different spline models (e.g. `bs()`, `ns()`) and adjusted the parameter `df` from 1 to 5.

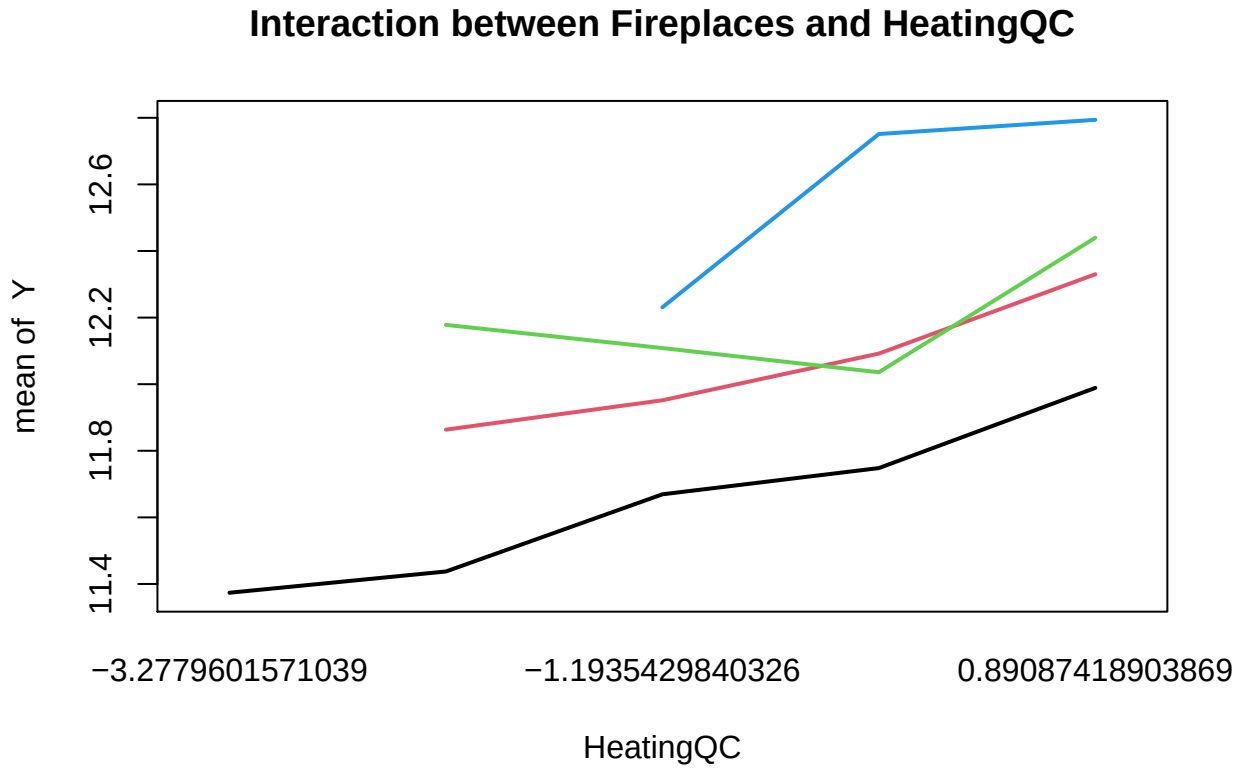

Y
13
12
11
−3
−2
−1
0
1
YearBuilt
−2
0
2
4
GarageArea
−4
−2
0
2
BsmtQual
−2
−1
0
1
2
KitchenQual
−5.0
−2.5
0.0
2.5
OverallQual
−2.5
0.0
2.5
5.0
TotalBsmtSF
X1stFlrSF
−4
ExterQual



Notice that there might exist some interaction within two variables. After observing the interaction plots, we discovered two significant interaction effects. For the graph of MSSubClass and Neighborhood, the colorful lines vary in slope, which indicates that the house's neighborhood would affect the relationship between Sale Price and MSSubClass.



For the graph of HeatingQC and Fireplaces, the colorful lines also vary in slope, which indicates that the placement of the fire in a house would affect the relationship between Sale Price and HeatingQC.



Eventually, we aggregated the splines as follows.

SVM

Support vector machine is a powerful supervised learning model applied in statistical problems. Based on the seperating hyperpline, it could serve as a classfier efficiently classify data into various categories. However, SVM could also be implemented in regression problems. Through maximizing the distance between support vectors and regression curve, it could "find the closest match between the data points and the actual function that is represented by them".

As the only model based on distance between data points in the foregoing toolbox, SVM regression was taken into account in the model building for this project.

The training data was fed into three SVM models with different kernels obtained from e1071 library. For each of them, parameters were tuned via cross validation. After achieving the best performance of each model, cross-validation method was again applied to estimate their accuracy.

```
# tc <- tune.control(cross = 10)
# obj_1 = tune(svm, Y~., kernal = "linear", data = train_sum,
# ranges = list(gamma = 2^(-2:2), cost = 2^(2:4)), tunecontrol = tc)
# obj_2 = tune(svm, Y~., kernal = "polynomial", data = train_sum,
# ranges = list(gamma = 2^(-2:2), cost = 2^(2:4)), tunecontrol = tc)
# obj_3 = tune(svm, Y~., kernal = "radial", data = train_sum,
# ranges = list(gamma = 2^(-2:2), cost = 2^(2:4)), tunecontrol = tc)
```

The result turned out that the model with linear kernel was reported the lowest RMSE, which was roughly 0.10.

Result

Cross-Validation

Here are the CV errors for six training models.

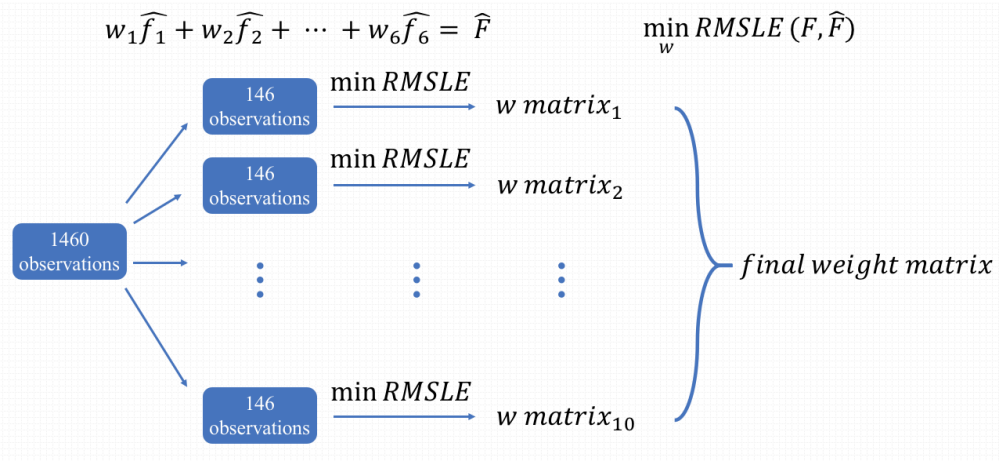
Type	Model	CV RMSLE
OLS & modifications	Forward selection regression	0.1145312
	Lasso regression	0.1176129
	PCR	0.1194666
	GAM	0.1180877
Tree-based models	Random forest	0.0643161
Distance-based models	SVR	0.1008077

Figure 1: CV RMSLE

As the table shows, Random Forest model performs the best. However, we did not put all bet on it. Instead, we decided to ensemble all the models, so as to 1) obtain more accurate and robust prediction, 2) nicely trade off between interpretability and flexibility.

Ensemble Methodology

- 1. Equally assign a weight to each model. (`coefficient <- rep(1/6,6)`)
- 2. Divide the training set into 10 parts at random. (`set.seed(10)`)
- 3. Acquire the local optimal coefficients for six models in each part of the dataset. (`optim()$par` was used. It is optimization function based on Nelder-Mead, quasi-Newtona adn conjugaet-graident algorithms. `$par` gives the)
- 4. Get the average coefficient as the ultimate weight for individual model.



Work on Test Dataset

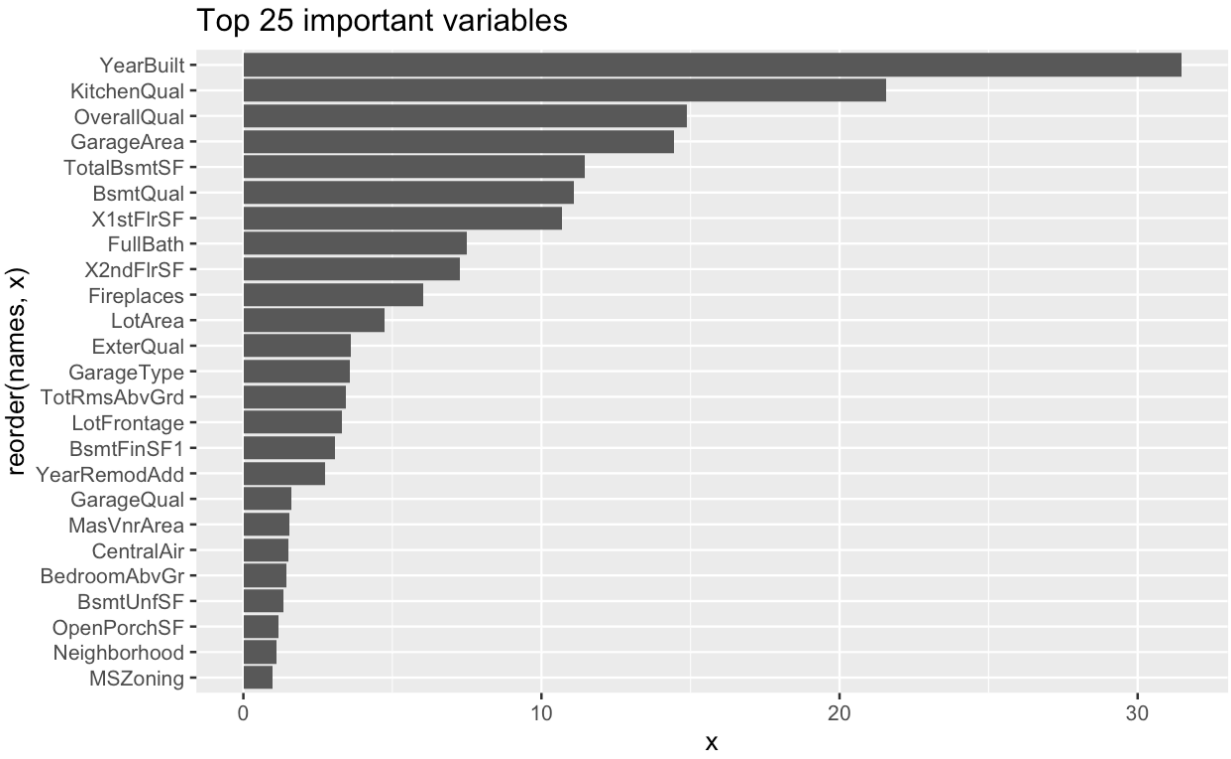
Before predicting the house price in the test data, we need to preprocess the features based on the modification of training dataset. That is, feature engineering and NA-filling. During the prediction process on test data, the SVM code was removed due to technical issue (intrinsic problem in `e1071` packages). In the end, only the remaining five models were used to predict the test data. After transforming the predicted house prices (`expm1()`), the score (RMSLE) of the prediction turned out to be 0.12340.

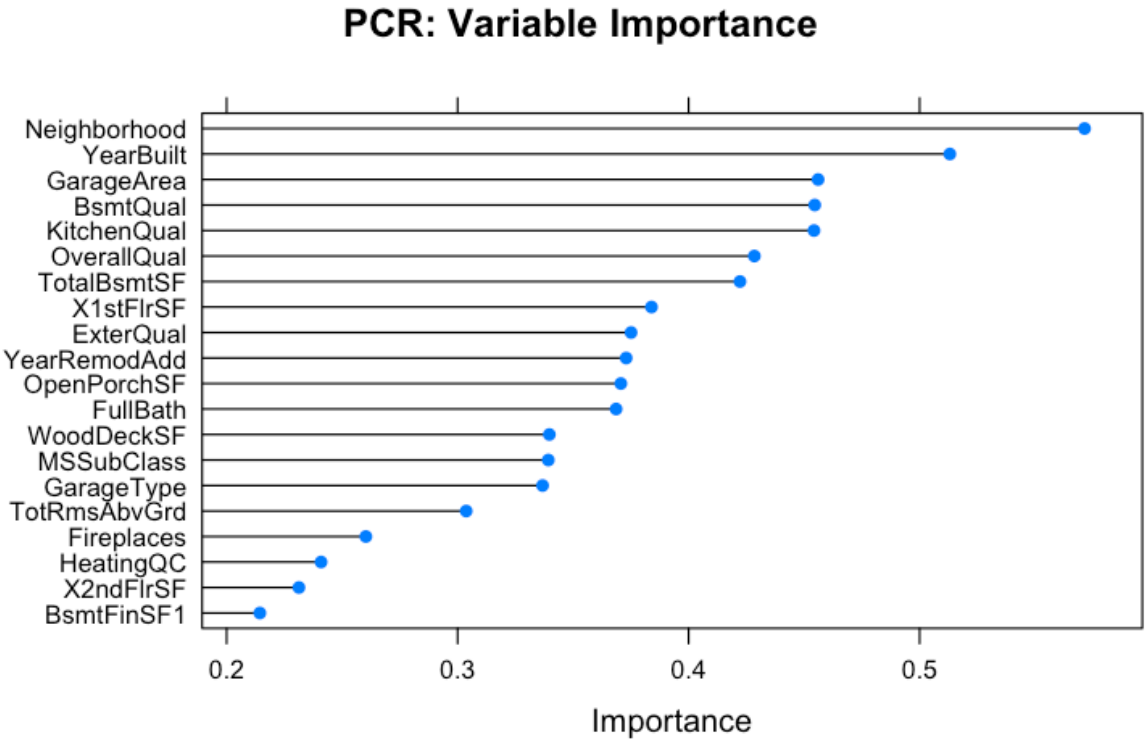
Discussion

This study aims at two parts, which serve at the guideline for all the data process.

The first purpose of the study is to achieve a relatively high accuracy of the model. In doing so, ensemble method was applied as it could balance the strength and shortcomings of individual models. As the result shows, the model after ensemble has a overwhelming advantage in prediction power.

The second purpose of the study is to infer what are the most important influential factors to housing prices. To this end, three models of the final ensemble, namely Forward Stepwise, PCR, and Random Forest, could provide valuable information. The following three graphs are the Variable Importance Measure generated by Forward Stepwise, PCR, and Random Forest, respectively.


Top 25 important variables
YearBuilt
KitchenQual
OverallQual
GarageArea
TotalBsmtSF
BsmtQual
X1stFlrSF
FullBath
X2ndFlrSF
Fireplaces
LotArea
ExterQual
GarageType
TotRmsAbvGrd
LotFrontage
BsmtFinSF1
YearRemodAdd
GarageQual
MasVnrArea
CentralAir
BedroomAbvGr
BsmtUnfSF
OpenPorchSF
Neighborhood
MSZoning
reorder(names, x)
0
10
20
30
x


PCR: Variable Importance
Neighborhood
YearBuilt
GarageArea
BsmtQual
KitchenQual
OverallQual
TotalBsmtSF
X1stFlrSF
ExterQual
YearRemodAdd
OpenPorchSF
FullBath
WoodDeckSF
MSSubClass
GarageType
TotRmsAbvGrd
Fireplaces
HeatingQC
X2ndFlrSF
BsmtFinSF1
0.2
0.3
0.4
0.5
Importance

Features	Rank	Features	Rank
KitchenQual	1	BldgTypeTwnhs	17
TotalBsmtSF	2	NeighborhoodBrkSide	18
X2ndFlrSF	3	NeighborhoodStoneBr	19
OverallQual	4	NeighborhoodNridgHt	20
YearBuilt	5	NeighborhoodSomerst	21
X1stFlrSF	6	MSZoningRL	22
GarageArea	7	NeighborhoodEdwards	23
Fireplaces	8	BsmtFullBath	24
BsmtFinSF1	9	NeighborhoodMeadowV	25
MSZoningRM	10	MSZoningRH	26
NeighborhoodCrawfor	11	MSZoningFV	27
LotArea	12	BsmtExposure	28
HeatingQC	13	GarageQual	29
SaleTypeNew	14	SaleTypeConLD	30
SaleConditionNormal	15	MSSubClass	31
Condition1Norm	16	SaleConditionAdjLand	32

From these graphs, the research reached several insights.

1) PCR attached greatest importance to the predictor "Neighborhood." Similarly, Forward Stepwise also ranked several neighborhood-related factors among top 25. These models' emphasis on the importance of neighborhood corresponds to people's common notion that location is vital to a house's value.

2) All three models emphasized the significance of the predictor "YearBuilt." Forward Stepwise ranked it as the fifth most important feature, PCR ranked it as the second, and Random Forest ranked it as the first. This reveals that the time of construction could greatly influence housing prices.

3) Besides "Neighborhood" and "YearBuilt," some other predictors also play significant roles. For instance, overall quality, basement area, and garage area all ranked fairly high on the Variable Importance Measure Plot. What's more, perhaps a bit surprisingly, Kitchen Quality appears to be one of the most important predictors, suggesting that real estate market participants should pay more attention to this index.

Conclusion

Conclusively, the process of predicting the Ames Housing Price can be summarized into 4 categories, exploratory data analysis, preprocessing, modeling and the result. In EDA process, the problems of missing value, scaling issue, redundant features, collinearity and skewed response variable were detected. During preprocessing, Missing values were translated or imputed using mode or KNN method according to their real meanings and characteristics. Scaling was carried out using z-score standardization since distance-based estimators often assume approximately standardized data. 8 features with high collinearity were removed with the help of Spearman index and 19 redundant features with small variances were deleted. The response SalePrice was fixed using logarithmic transformation.

The regression models were classified into 3 types, OLS models, tree-based models and distance-based models. Within OLS range, forward selection regression, lasso regression, principal component regression and GAM were selected to fit and predict the housing price. Support vector regression and random forest were also included since they separately represented distance-based model and tree-based model. The above 6 models were fitted and compared using cross-validation errors. Random forest outstood with the lowest cross-validation RMSLE. However, to obtain more accurate and robust prediction and also for the sake of the trade off between interpretability and flexibility, ensemble methodology was at last considered.

Before predicting the house price in the test set, the test data were preprocessed based on the modification of training data. Since the SVM model was removed due to package problem, the remaining 5 models were used to predict the test data. The score (RMSLE) of the prediction turned out to be 0.12340.

References

Bradley, B. (n.d.). Random forests. UC Business Analytics R Programming Guide. https://uc-r.github.io/random_forests

Cock, D. (2011). Ames, Iowa: Alternative to the Boston Housing Data as an end of semester regression project. Journal of Statistics Education, 19(3). https://doi.org/10.1080/10691898.2011.11889627

Hyun, K. (2013). The prevention and handling of the missing data. Korean Journal of Anesthesiology, 64(5), 402-406. doi: 10.4097/kjae.2013.64.5.402

Kaggle. (2020). House prices - Advanced regression techniques. https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview

Morgan Stanley Capital International. (2020). Market size report on global real estate. https://www.msci.com/real-estate/market-size-report

P. Jonsson and C. Wohlin, "An evaluation of k-nearest neighbour imputation using Likert data," 10th International Symposium on Software Metrics, 2004. Proceedings., Chicago, Illinois, USA, 2004, pp. 108-118, doi: 10.1109/METRIC.2004.1357895.

Cramer's V. (2020, December 27). In Wikipedia. https://en.wikipedia.org/wiki/Cram%C3%A9r%27s_V