Introduction to Machine Learning
Homework

House Prediction Using Advanced Regression Techniques

Members:
Andrew Kamya (1849786)

Supervisor:
Prof. Dr. Dirk Valkenborg

April 10, 2025

Contents

Abstract

The main purpose of this project was to predict the sales price of some residential houses in Ames, Iowa from year 2006 to 2010. To be able to successfully answer this, various statistical learning methods were used on the train data set to come up with the best model. Exploratory data analysis was done to check which features are highly related to the prices of the houses. Unsupervised learning method, that is, Principal component analysis was used as an exploratory technique on the continuous predictors to get an insight of which continuous predictors are of importance. Predictive models such as multiple linear regression, ridge regression, lasso regression, bagging, random forest and boosting techniques were trained and the best was selected by the test mean square error criterion. Boosting had the best performance and therefore it was used to predict the sale price of the given test data set. Finally, the ground living area, overall quality, neighborhood and size of the garage area of a house are highly related to the sale price. The Kaggle score obtained:: 0.13361, Top 40% Leaderboard.

Keywords : *Boosting, Mean Squared Error, Lasso, Prediction, Regression*

1 Introduction

Property, Mortgages and real estate has of recent years emerged to be one of the most liked investment all over the world. This is caused majorly by rapid increase in the population sizes being experienced both in the rural and urban areas. Despite a house being one of the most valued asset that an individual, family or group of people could have, owning a house is not easy due to high monetary value attached to them. In many house transactions the prices are mainly influenced by their geographical locale among other physical attributes such as the size, number of rooms, the year it was built, etc.
These factors subject house prices to fluctuation over the year. In the past, determining the price of a particular house bearing certain characteristics was a problem. However, in recent times, modern statistical techniques such as machine learning and artificial intelligence has made it easier to predict the house prices based on the defining characteristics.
In this report, machine learning techniques were applied in predicting analysis prices of the houses in the test data from 'Ames housing data set' using both supervised and unsupervised machine learning techniques. The report is comprised of 5 sections, in this part, background information has been discussed. In section 2, 3, 4, and 5, data description, data cleaning and feature engineering, methods, results conclusion, challenges and recommendation were discussed respectively.

2 Data description

Ames housing data from kaggle.com was used in this analysis. The original dataset consisted of two parts; train data set contained 1460 observations with a continuous response variable *SalePrice* whilst, test data set contained 1459 observations. There were 80 predictor variables which was a mixture of continuous and discrete variables. The predictor variables described house characteristics that will be helpful to a customer before the transaction is done.

3 Methodology

3.1 Data cleaning and Feature Engineering

The train dataset and the test dataset were combined for an overall cleaning process. There were 34 covariates with at least one "NA" value. We first checked for the meaning of "NA" in each case. If "NA" came from the fact that the feature was not present in the house, then it was replaced by value 0 for continuous variable, else stated as "None" for categorical one. The other "NA" which were deemed to be real missing value, were first assumed to be missing at random. Then, missing values were imputed with the variable median for continuous cases, and with variable mode for categorical predictor. By doing this, the distributions of the variables were not affected. After having a complete dataset, new variables were created. $AgeoHouse$ (the house age) at the time of sale which was calculated by subtracting $YearBuilt$ from $YrSold$, $AgeoGarage$ (the garage age)calculated by subtracting $GarageYrBlt$ from $YrSold$ and $TotalSF$ was calculated as the sum of $TotalBsmtSF$, $SecondndFlrSF$ and $FirststFlrSF$. Also, $Remodel$ was created with different levels based on the length of time since the house was renovated until the selling date. Having these new variables capturing the information, $YrSold$, $YearBuilt$, $GarageYrBlt$, $YearRemodAdd$, $TotalBsmtSF$, $SecondndFlrSF$ and $FirststFlrSF$ were dropped from the dataset. $AgeoGarage$ was also dropped as it was highly correlated to $AgeoHouse$. Id was dropped as being an index only.

In summary, at the end of the cleaning process, a complete dataset with 74 predictors was obtained and taken to the analysis stage.

3.2 Exploratory Data Analysis (EDA)

Data exploration was conducted to get insights of how the $SalePrice$ relates to the predictor variables. For continuous predictor variables, a correlation matrix was used to see which covariates are highly correlated with the response variable ($SalePrice$), that is $\rho > 0.5$, and whether there is presence of multicollinearity between the predictors themselves while box plots were used for the discrete and categorical variables. To ease EDA, an unsupervised learning method (*principal component analysis*) was employed. PC analysis is a tool used for better data visualization and pre-processing by reducing the dimension of the predictor variables before supervised learning methods are used. Here, biplots, scree plots and cumulative variance plots were used.

3.3 Model Fitting

3.3.1 Multiple Linear Regression Model

Multiple linear regression is a basic and commonly used modelling technique since it offers significant advantages regarding interpretation and also competitive to non-linear methods (1). Its general form is specified as below:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon \tag{1}$$

where Y is the response variable, $X_1, X_2, .., X_p$ are the predictors, p is the number of predictors and $\epsilon \sim N(0, \sigma^2)$ is the error term. In this project, backward stepwise selection was used to

identify a subset of candidate variables. The model was then checked for satisfaction of linear regression assumptions.

3.3.2 Shrinkage techniques

Here model was fitted with all the predictor variables. These techniques use a regularization method to the estimates of the coefficients, that is, by shrinking the coefficient estimates towards zero. It has been studied that shrinking significantly reduces the variation of the coefficient estimates (Gareth, 2017). In this analysis both ridge and lasso regression were applied. Ridge regression The difference between ridge regression and least squares method is small where in ridge regression there is shrinkage property in play. It aims at obtaining parameter estimates that minimize

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}) + \lambda\sum_{j=1}^{p}\beta^2 \quad \text{where} \;\; \lambda > 0 \tag{2}$$

The first term is the residual sum of squares (RSS) and the second term is the shrinkage penalty whereas λ is the tuning parameter. λ works to control the effect of equation 1 to the parameter estimates. when $\lambda = 0$, least squares estimates are yielded. Likewise, as λ increases(tends to ∞) the ridge regression coefficients approaches zero. To note, shrinkage is not applied to the overall mean of the response (β_0). The cross validation method is used to determine the best λ that fits the data well. Lasso regression Similarly the main objective of lasso regression is to obtain the subset of predictors that minimize the prediction error for a quantitative response variable therefore, improving prediction accuracy and interpretability of the model. The first and second terms of equation 3 are similar to equation 2.

$$RSS + \lambda\sum_{j=1}^{p}|\beta_j| \quad \text{where} \;\; \lambda > 0 \tag{3}$$

After shrinkage, predictors with a lasso regression coefficient equal to zero are excluded from the fitted model, whilst predictors with non-zero regression coefficients are considered most strongly associated with the response variable hence the lasso regression analysis helps determine which predictors are most important in explaining the response. Also selection of the best lambda was conducted using cross validation technique.

3.3.3 Tree-Based Methods

In regression trees, multiple trees are produced and combined to obtain a single consensus prediction. A lower prediction error is witnessed when large number of trees are combined but this has a disadvantage of difficult interpretability. Bagging and Random Forest This is a bootstrap aggregation learning method. Bootstrap is used to obtain B separate training data sets sampled with replacement hence creating B regression trees. A model is fitted and an average prediction is obtained.

$$\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x) \tag{4}$$

There is no need to perform cross validation for the bagged models to estimate the test error. On average, according to (Gareth James, 2017), each bagged tree consist of two-thirds of the observation. The kept-aside observations also called the "Out-of-Bag" (OOB) observations for each tree are used to predict the response of the i^{th} observation. This produces $\approx \frac{B}{3}$ predictions

for the i^{th} observation which are then averaged to give a single prediction. The OOB can then be used to calculate the overall OOB MSE which is a valid estimate of the test error for the bagged models.
More so, Random forests provide an extension over bagged trees by way of decorrelating the tree by random selection of predictors. When building the decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split potential candidate from the full set of p predictors. The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose m $p/3$. This process goes on iteratively and Prediction is given based on the aggregation of predictions from a number of trees. Boosting Boosting is similar to Bagging except that the trees are built sequentially, that is, every tree is grown based on the information obtained from the trees that had already grown. This method is considered slow since the model is fitted to the residuals then a tree is built from these residuals (iteratively updating the residuals) and these slowly improve the prediction. Boosting method uses three tuning parameters; the number of trees (selection by CV), shrinkage parameter λ and the number of splits of each tree which controls the complexity of boosted ensemble (Gareth, 2017).

3.4 Model Selection

Validation set approach was used to assess the performance of different models. The train dataset was randomly divided into two parts: (1) a training set on which different models were fitted and (2) a validation set on which fitted model performance were checked by assessing MSE from the predicted response versus the observed values. This validation MSE provided an estimate of the test error rate. The model with lower validation MSE performed better.

4 Results

4.1 Exploratory Data Analysis

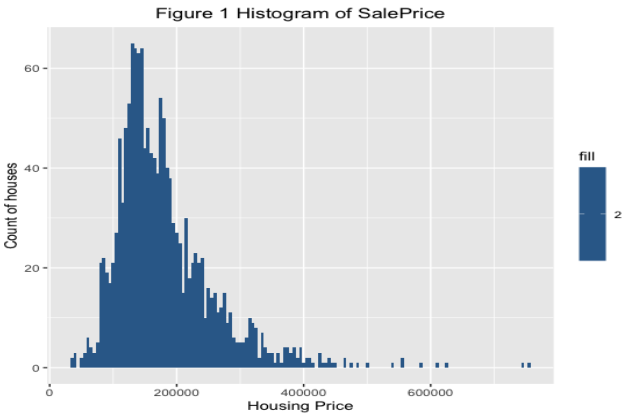

Figure 1: *Sale price histogram*

Figure 2: *Correlation matrix*

Figure 1 shows that most of the houses that were sold had a sale price of about 130000 whereas Figure 2 shows the correlation matrix for the continuous variables whose correlation with the Sale price was above 0.5, that is, $|\rho| > 0.5$ and $TotalSF$ has the highest correlation with the sale price followed by $GrLivArea$ and so on, also the house age has a moderate negative correlation with the sale price.

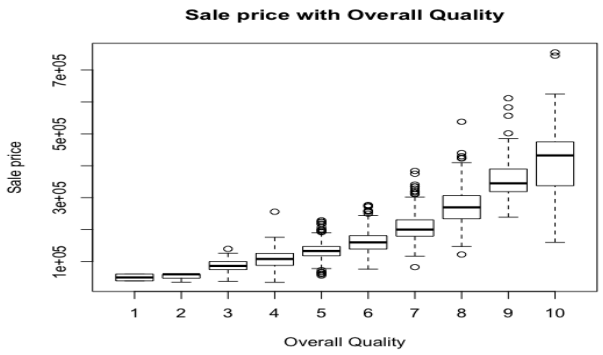

Figure 3: *Boxplot for Overall quality*

Figure 4: *Boxplot for Street*

Figure 3 and 4 show the distribution of the house sale price among the different levels of each of the covariates and using figure 3, the sale price seems higher for houses of better quality and for figure 4 to be almost the same for the two road types.

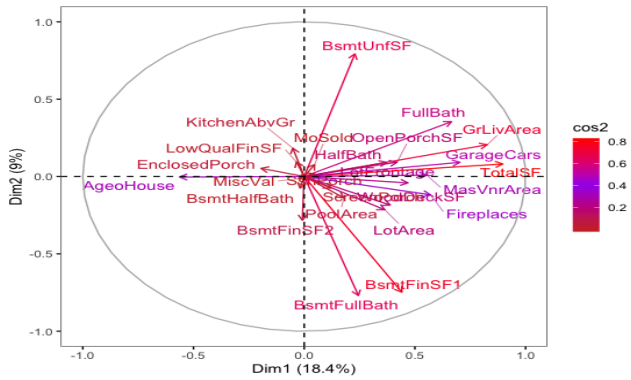

Figure 5: *Biplot for PCA*

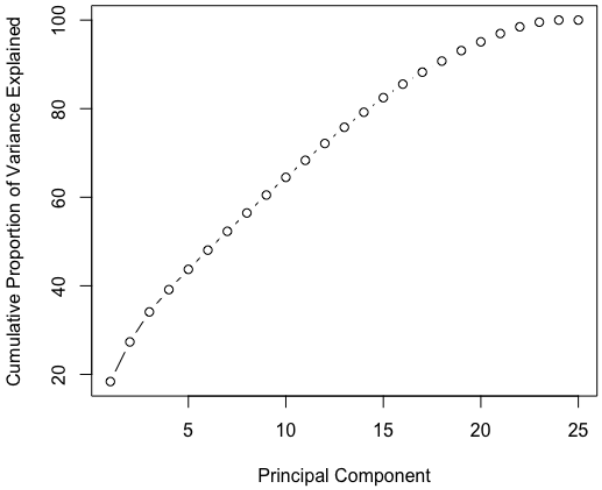

Figure 6: *Cumulative variance for PC's*

The Biplot in Figure 5 summarizes PCA as an unsupervised statistical learning method. Standardization of all the continuous variables to create proportionality with same unit was done. In this plot, variables like $TotalSF$, $TotalRmsAbvGrd$, $GrLivArea$, $BsmtFinSF1$,$BsmtFullBath$ and $BsmtUnfSF$ seem to have a higher effect when interpreting the first principal component. However, variables that are closed to the center of the plot are less important in explaining the components. Furthermore, Figure 6 showing the cumulative proportion of variance explained by each of the 22 principal components.

Before further analysis was conducted, and with the information from Figure 1 which indicated that $SalePrice$ is right skewed, a logarithmic transformation was applied on $SalePrice$ to create a normal distribution before data was trained on it. After obtaining the predicted $\log(SalePrice)$ for the test data, back-transformation was done.

4.2 Model Fitting

4.2.1 Multiple Linear Regression

Utilizing backward stepwise selection, a subset of 15 candidate variables was used to fit a multiple linear regression. The adjusted R^2 was obtained at 0.8908. However, this model did not satisfy the linear regression assumption of normality as illustrated by the residual plot in Figure 7. In addition, other fitting procedures instead of least squares can yield better prediction accuracy. Hence, we explored other methods, namely shrinkage techniques and tree-based methods in the coming sections.

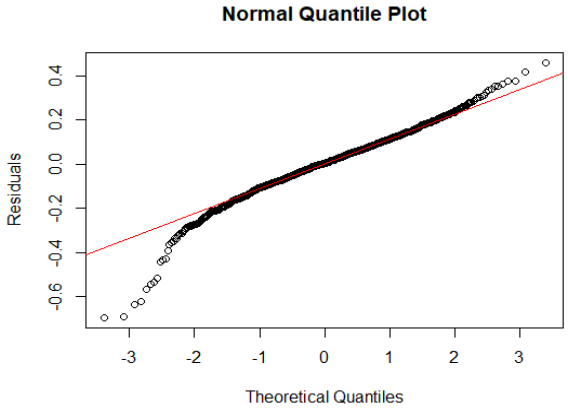

Figure 7: *Normal quantile plot*

4.3 Shrinkage Techniques

4.3.1 Lasso Regression

Figure 8 (left) illustrates the design in which the predictors are dropped from the model as the penalty increases. Using the aforementioned method in section 3.3.2, the shrinkage penalty forces the coefficients of the unimportant variables to zero as the shrinkage/log(lambda) increases. This method can be seen a seen a variable selection method. Also, a 10-fold cross-validation was conducted in order to select best value of the tuning parameter (λ). Therefore the best lambda value of 1494.73 and thus an MSE of 1212259254. The first vertical line on the second plot (right) shows the value of (λ) at which the minimum cross validation error is achieved. The second vertical line is the maximum value of cross validation error within one standard error of the minimum. The area on the right of maximum cross validation error increases gradually and this can be due to overfitting the data. The numbers on top of the figure give the number of non-zero coefficients.

The most challenging part of the Lasso regression is the analyst do not have control over variable selection especially in presence of the correlated variables. It only selects one and set all the others to zero.

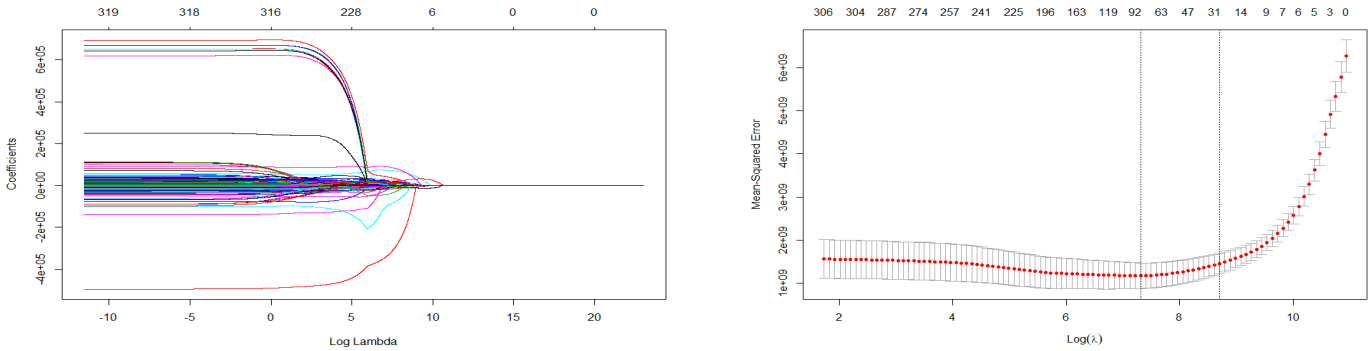

Figure 8: *The standardized lasso coefficients on the housing price data set as a function of tuning parameter (left). The lasso plot showing the best lambda (right).*

4.3.2 Ridge Regression

According to the theory is section 3.3.2, it can also be observed in Figure 2 (*left*) that the shrinkage penalty do not force the coefficients of the model to exactly zero rather the coefficients are made close to zero. Same as in lasso regression, a 10-fold cross validation was used to select the best value of the tuning parameter. In Figure 9 (*right*), mean square error is plotted. It can be observed that the minimum validation error can be approximately 11 (in log-scale). Similar behavior is observed by gradual increase in MSE as value of log λ increases. The best lambda was obtained to be $\lambda = 47820.4$ and MSE =1211215433.

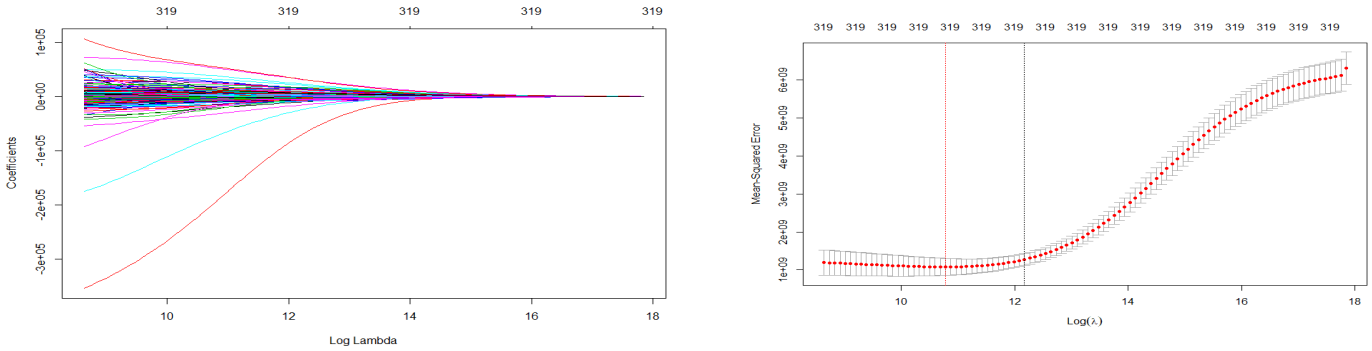

Figure 9: *The standardized ridge coefficients on the housing price data set as a function of tuning parameter (left). The ridge plot showing the best lambda (right).*

4.4 Tree-based Methods

4.4.1 Bagging and Random Forests

In order to compare the performance of Bagging and Random forests, validation set approach was used. For bagging, 74 variables were included in the analysis. The validation MSE resulted from the bagged regression tree was 978684204 and 87.89% of the variability in the response variable was explained by the included predictors. On the other hand, random forests regression was conducted with 25(74/3) candidate variables for each split. In this case, the validation MSE was obtained at 869126739 and 88.19% of variability was explained by the predictors. This result showed an improvement of random forests over bagging. Figure 10 is a graphical illustration of the variable importance based on the mean decrease of accuracy in predictions on the out of bag samples when a given variable is excluded from the model (*left*) or the total decrease

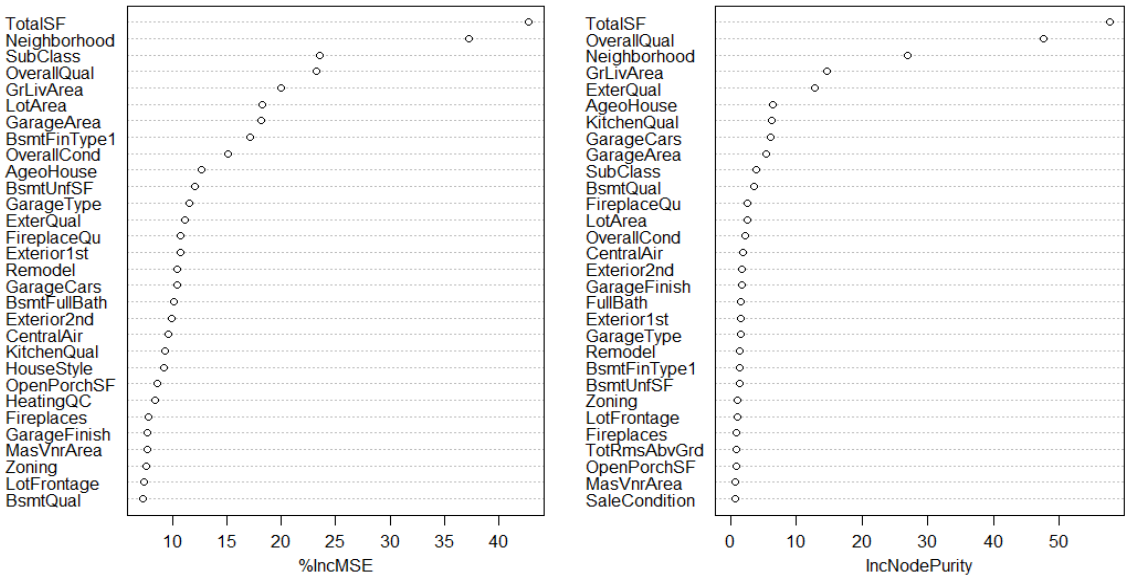

Figure 10: *Variable importance*

in node impurity that came from splits over that variable (*right*). The result indicated that, among all of the trees considered in the random forest, $TotalSF$, $OverallQual$, $Neighborhood$, $GrLivAre$ and $ExterQual$ in that order, were by far the five most important variables.

4.4.2 Boosting

A gradient boosted model with Gaussian loss function indicated that out of the 74 predictors, 68 had a non-zero influence. This was a result from iterative procedure with number of iterations=5000. with shrinkage parameter of $\lambda = 0.1$ The $Street$, $Utilities$, $Condition2$, $PoolArea$ and $PoolQC$ did not show any influence on the sale price of the house.
Moreover, $OverallQual$, $Neighborhood$, $GrLivArea$, $TotalSF$, $GarageArea$, $KitchenQual$ and $OverallCond$ were among the six top ranked factors that influenced the sale price. This method learns slowly and fits the data hard and the shrinkage parameter slows the process even further but the good thing is that this methods learns by strongly depending on the already existing trees. The summary of MSE from validation set is summarized in Table 1
XGBOOST (*Extreme Gradient Boosting*) is an optimized distributed gradient boosting library was performed. This method usest uses gradient boosting (GBM) framework at core. But it is advantageous and has better performance than GBM framework alone. This approach works by boosting algorithms that convert weak learners into strong learners. A weak learner is one which is slightly better than random guessing. Same as boosting, XGBOOST is a sequential process; i.e., trees are grown using the information from a previously grown tree one after the other. This process slowly learns from data and tries to improve its prediction in subsequent iterations. The summary of MSE for choosing this method is also summarized in Table 1.

4.4.3 Model Selection

A summary table of model performance based on validation set approach is specified below:

Model	MSE
Lasso	1212259254
Ridge	1211215433
Bagging	978684204
Random forests	869126739
Boosting	862058500
XGBOOST	840120957

Table 1: *Summary on model performance*

Based on Table 1, XGBOOST was the method which worked the best.

5 Discussion and Conclusion

Advanced regression techniques applied from the family of supervised learning methods used to train the data included multiple linear regression, ridge regression, lasso regression, bagging, random forest and boosting. From the results obtained, it was noted that the multiple linear regression (MLR) captured the highest amount of variation in the data (89.08%) as compared to the other techniques, However, this criterion is not enough to assess the model fit since our goal is prediction and the functional form of the model is not of importance. The subset selection method was used to select important predictors to be included in the MLR. Further, the best technique to train and predict was chosen based on the lowest estimate for the test MSE from the test observations via the validation-set approach. With this regard, comparison between ridge & lasso regression, the bootstrap aggregation approach, the random forest approach and boosting were made.
It was of importance to note that the $GrLvArea$, $Neighborhood$, and $OverallQual$, with $GrLvArea$ showed a consistency in predicting the sale price of a house. This is in tandem with the results from the EDA and the biplot.
The best method was using XGBOOST but according to MSE criterio, it was competing with the boosting technique.

Kaggle score = 0.13361 :::Top 40% on the Leaderboard

6 Challenges

- There was a lot of missing data especially in the cateorical variable $LotFrontage$ and this made imputation pretty difficult.
- The dataset had a lot of categorical predictor variables with very many levels, so some methods like Principal Component Regression could not be applied because it can not handle non-binary categorical variables.
- The project called for a lot of time especially when it came to data cleaning and applying the different statistical models

References

[1] James, Gareth and Witten, Daniela and Hastie, Trevor and Tibshirani, Robert; An introduction to statistical learning, 2013.

R Codes

```
# Load Packages;
install.packages("pacman")
library(pacman)
p_load(MASS, Metrics,corrplot,randomForest,ISLR,lars,ggplot2,xgboost,Matrix,
      methods,caret,factoextra,FactoMineR,xtable,dplyr,cluster,ggthemes,glmnet,
      devtools,ggcorrplot, pls,leaps,psych,coefplot,xgboost,readr,stringr,car,readxl,
      MASS,tidyverse,Hmisc,gbm)

#Dataset
data=read.table(file.choose(), header = TRUE)

#Covert to factors
data$SubClass = as.factor(data$SubClass)
data$Zoning = as.factor(data$Zoning)
data$Street = as.factor(data$Street)
data$Alley = as.factor(data$Alley)
data$LotShape = as.factor(data$LotShape)
data$LandContour = as.factor(data$LandContour)
data$LotConfig = as.factor(data$LotConfig)
data$Utilities = as.factor(data$Utilities)
data$LandSlope = as.factor(data$LandSlope)
data$Neighborhood = as.numeric(data$Neighborhood)
data$Condition1 = as.factor(data$Condition1)
data$Condition2 = as.factor(data$Condition2)
data$BuildingType = as.factor(data$BuildingType)
data$HouseStyle = as.factor(data$HouseStyle)
data$OverallQual = as.factor(data$OverallQual)
data$OverallCond = as.factor(data$OverallCond)
data$Remodel = as.factor(data$Remodel)
data$RoofStyle = as.factor(data$RoofStyle)
data$RoofMatl = as.factor(data$RoofMatl)
data$Exterior1st = as.factor(data$Exterior1st)
data$Exterior2nd = as.factor(data$Exterior2nd)
data$MasVnrType = as.factor(data$MasVnrType)
data$ExterQual = as.factor(data$ExterQual)
data$ExterCond = as.factor(data$ExterCond)
data$Foundation = as.factor(data$Foundation)
data$BsmtQual = as.factor(data$BsmtQual)
data$BsmtCond = as.factor(data$BsmtCond)
data$BsmtExposure = as.factor(data$BsmtExposure)
data$BsmtFinType1 = as.factor(data$BsmtFinType1)
data$BsmtFinType2 = as.factor(data$BsmtFinType2)
data$Heating = as.factor(data$Heating)
data$HeatingQC = as.factor(data$HeatingQC)
```

```
data$CentralAir = as.factor(data$CentralAir)
data$Electrical = as.factor(data$Electrical)
data$FireplaceQu = as.factor(data$FireplaceQu)
data$PavedDrive = as.factor(data$PavedDrive)
data$PoolQC = as.factor(data$PoolQC)
data$MiscFeature = as.factor(data$MiscFeature)
data$SaleType = as.factor(data$SaleType)
data$SaleCondition = as.factor(data$SaleCondition)
data$BedroomAbvGr = as.factor(data$BedroomAbvGr)
data$KitchenQual = as.factor(data$KitchenQual)
data$Functional = as.factor(data$Functional)
data$GarageType = as.factor(data$GarageType)
data$GarageFinish = as.factor(data$GarageFinish)
data$GarageQual = as.factor(data$GarageQual)
data$GarageCond = as.factor(data$GarageCond)
data$PavedDrive = as.factor(data$PavedDrive)
data$PoolQC = as.factor(data$PoolQC)
data$Fence = as.factor(data$Fence)
data$MiscFeature = as.factor(data$MiscFeature)
data$SaleType = as.factor(data$SaleType)
data$SaleCondition = as.factor(data$SaleCondition)

#divide the data
data1=data
data1$TotalBsmtSF=(data1$TotalBsmtSF+data1$SecondndFlrSF+data1$FirststFlrSF)
train = data1[1:1460,-1]
test = data1[1461:2919,-c(1,79)]
dim(test)
dim(train)

#drop correlated variables and variables with high missing values
train_new <- train[,-c(34,43, 44)]
test_new <- test[,-c(34,43, 44)]
dim (train_new)
dim(test_new)

##Exploratory Data Analysis
#Boxplot
boxplot(all$SalePrice~all$Street, ylab='Sale price',xlab="Street type", main="Sale Price
boxplot(data$SalePrice~data$OverallQual, ylab='Sale price',xlab="Overall Quality", main=

#PCA.
pr.out = prcomp(trainfinal_num,scale=T)
biplot (pr.out , scale =0,cex=.6)
pr.out$sdev
```

```
#Obtaining the variance explained by each principal component
pr.var =pr.out$sdev ^2
#Computing the proportion of variance explained by each principal component
pve=100*pr.var/sum(pr.var )
plot(pve , xlab= "Principal Component", ylab="Proportion of Variance Explained", ylim=c(
#As the cumulative PVE
plot(cumsum (pve ), xlab="Principal Component", ylab ="Cumulative Proportion of Variance
#screeplot
eigen_pca = get_eigenvalue(pr.out)
fviz_pca_var(pr.out,col.var="cos2",repel=T,title="",gradient.cols=c("brown3", "purple","

## LASSO regression!
train_new$SalePrice1=log(train_new$SalePrice)
trann=train_new[,-75]
set.seed(10000)
x = model.matrix(trann$SalePrice1~.,trann)# Exclude Id column
y = trann$SalePrice1
lambdas = 10^seq(10,-5,length=10000) # Results in better plot resolution
lasso.mod = glmnet(x,y,alpha=1,lambda=lambdas) # Lasso model
cv.lasso = cv.glmnet(x,y,alpha=1) # 10-fold cv
lasso.mod.coef = extract.coef(cv.lasso) # extract non-zero coefficients!
best.lamda = cv.lasso$lambda.min
cv.lasso$cvm[cv.lasso$lambda == cv.lasso$lambda.min] # MSE of cv with best lambda
par(mfrow=c(1,1))
plot(lasso.mod, "lambda")
plot(cv.lasso)
abline(v=log(best.lamda),col="red",lty=9)

## Predicting SalePrice using test data!
test = test_new
z=model.matrix(~.,test)
lasso.pred = predict(lasso.mod,s=best.lamda,newx=z)
xx=data.frame(lasso.pred)
xx$X1=exp(xx$X1)
write.table(xx, file="F:\\project\\ISLR\\pred.csv")

## Ridge regression!
set.seed(10000)
x = model.matrix(trann$SalePrice~1.,trann) # Exclude Id column
y = trann$SalePrice1
lambdas = 10^seq(10,-2,length=10000) # Results in better plot resolution
ridge.mod = glmnet(x,y,alpha=0,lambda=lambdas) # Lasso model
cv.ridge = cv.glmnet(x,y,alpha=0) # 10-fold cv
ridge.mod.coef = extract.coef(cv.ridge) # extract non-zero coefficients!
best.lambda.ridge = cv.ridge$lambda.min
```

```
cv.ridge$cvm[cv.ridge$lambda == cv.ridge$lambda.min] # MSE of cv with best lambda
par(mfrow=c(1,1))
plot(ridge.mod,"lambda")
plot(cv.ridge)
abline(v=log(best.lamda.ridge),col="red",lty=9)

## Predicting SalePrice using test data!
test = test_new
z=model.matrix(~.,test)
ridge.pred = predict(ridge.mod,s=best.lamda.ridge,newx=z)
write.table(ridge.pred, file="F:\\project\\ISLR\\pred.csv")

#Boosting
set.seed (1)
boost.hse =gbm(trann$SalePrice1~.,data=trann$SalePrice1, distribution="gaussian",
               shrinkage=0.1, n.trees =5000 , interaction.depth =4)
boost.hse$shrinkage
yhat.boost=predict(boost.hse ,newdata=test_new,n.trees =5000) #Best pred so far
summary(boost.hse)
xx=data.frame(yhat.boost)
xx$yhat.boost=exp(xx$yhat.boost)
write.table(xx, file="F:\\project\\ISLR\\submit9113.csv")

#Multiple Linear regression - Backward selection
MLR.bwd=regsubsets(SalePrice_new~.,data=train,nvmax = 30, method = "backward")
coef(MLR.bwd, 30)
summary=summary(MLR.bwd)
summary$adjr2
MLR.bwd2<-lm(SalePrice_new~ Zoning+LotArea+Neighborhood+Condition2+OverallQual
+OverallCond +RoofMatl+Exterior1st+Foundation+HeatingQC+GrLivArea+GarageCond+SaleType
+BsmtQual+GarageQual,data=train)

summary(MLR.bwd2)
yhat.MLR=predict(MLR.bwd2)
yhat.MLR=predict(MLR.bwd2, newdata = test)
pred.MLR=exp(yhat.MLR)

qqnorm(res.MLR,main="Normal Quantile Plot", ylab="Residuals")
qqline(res.MLR,col="red")

#Bagging
set.seed(1)
bag.trainb=randomForest(SalePrice_new~.,data=train_b, mtry=74, importance=T)
bag.trainb
yhat.bag= exp(predict(bag.trainb, newdata = test_b))
```

```
testMSE.bag=mean((yhat.bag-test_b1$SalePrice)^2)
testMSE.bag

set.seed(1)
bag.train=randomForest(SalePrice_new~.,data=train, mtry=74, importance=T)
pred.bag.log  = predict(bag.train, newdata = test)
pred.bag=exp(pred.bag.log)
pred.bag

#Random forest
set.seed(1)
RF.trainb=randomForest(SalePrice_new~.,data=train_b, mtry=25, importance=T)
yhat.RF= exp(predict(RF.trainb, newdata = test_b))
RF.trainb
testMSE.RF=mean((yhat.RF-test_b1$SalePrice)^2)
testMSE.RF

set.seed(1)
RF.train=randomForest(SalePrice_new~.,data=train, mtry=25, importance=T)
pred.RF.log= predict(RF.train, newdata = test)
pred.RF=exp(pred.RF.log)

varImpPlot(RF.train, main="Random Forest Importance Plot")

###XGBOOST

train_new <- data.matrix(traindata[,-c(34,43, 44,78)])
test_new <- testdata[,-c(34,43, 44)]
traindata1 <- data.matrix(traindata[, -78])
dim(traindata1)
y_traindata1 <- data.matrix(log(traindata$SalePrice))
xgboost_train <- xgb.DMatrix(data=train_new, label = y_traindata1, missing = NaN)
testdata$SalePrice=rep(0,1459)
testdata1 <- data.matrix(test_new)
dim(testdata1)
y_testdata <- testdata$SalePrice
xgboost_test <- xgb.DMatrix(data = testdata1)

model <- xgboost(data = xgboost_train, # the data
                 nround = 5000, # max number of boosting iterations
                 subsample = 0.7, #how much of the data to use for each tree

                 max_depth = 5, #how many levels in the tree
                 eta = 0.1, #shrinkage rate to control overfitting through conservative
```

```
                eval_metric = "rmse",
                objective = "reg:linear",

names(model)=names(xgboost_test)
test_pred1 <- predict(model,xgboost_test)
test_pred1 <- predict(xgb,xgboost_test)
xx=data.frame(test_pred1)
```