# Predicting Sales Prices for Ames Housing Data

1. **Introduction**

I will be exploring housing data in order to build a regression model that predicts the final sales prices for homes in Ames, Iowa. The dataset was provided by *Dean De Cock* from Truman State University and is available to the public for data science practice and competition. The following link is used to access the data as well as provide more in-depth descriptions about the dataset and its underlying variables: <https://www.kaggle.com/c/house-prices-advanced-regression-techniques>.

Essentially, I’m hoping to determine a regression model that does the “best job” predicting the sales prices for homes in Ames, and in order to do this, I will be utilizing feature engineering, variable selection methods, resampling techniques, and other various data cleaning procedures. Models that use penalized regression methods will then be produced and analyzed in order to see which particular one fits the data most appropriately and most accurately predicts sales prices. The final model will be determined by numerous contributing factors, such as Kaggle score, RMSE value, and other statistical knowledge and intuition.

**Data Description:** The original data joins the train and test sets to create a larger data set of 2,919 observations and 81 total variables. It consists of 80 explanatory variables that explain almost every characteristic of residential homes in Ames and consists of both discrete and continuous numeric variables as well as categorical variables. It also contains the response variable, *SalePrice*. There are 20 continuous explanatory variables relating to the dimensions of the house, 14 discrete explanatory variables that count the number of items in a house (i.e., the number of rooms in a house- such as bedrooms, bathrooms, etc.), and the rest are categorical variables ranging anywhere between 2 and 28 classes.

1. **Feature Engineering**

Before merging the train and test datasets, I had to drop *Id* from the train and test sets and *SalePrice* from the train set (*SalePrice* will be reinstated later). I then created four new variables using feature engineering: (1) *TotalBathrooms* (2) *Age* (3) *Remodeling* (4) *New*. Feature engineering techniques that deal with the creation of *TotalBathrooms*, *Age*, *Remodeling*, and *New*, can be attributed to Erik Bruin and can be accessed via the following kernel link: <https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda/report>.

*TotalBathrooms* combines the four bathroom variables of *FullBath*, *HalfBath*, *BsmtFullBath*, and *BsmtHalfBath*,and represents the total number of bathrooms in a house. The “cor()” function helps show that *TotalBathrooms* and *SalePrice* are quite positively correlated (r ≈ .63), so combining these variables into one may be more important for predicting sales prices compared to having four distinct variables that do not appear to be significant with *SalePrice* on their own. These five bathroom variables and their correlations are compared and demonstrated in [Table 1](#_Table_1). As we can see, *TotalBathrooms* is the most correlated so the four original bathroom variables are dropped from the dataset and only *TotalBathrooms* is kept.

# Table 1

Table 1 – Demonstrates the five bathroom variables and their correlations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TotalBathrooms** | **FullBath** | **HalfBath** | **BsmtFullBath** | **BsmtHalfBath** |
| 0.6277554 | 0.5606638 | 0.5606638 | 0.2271222 | -0.01684415 |

The *Age* variable represents the difference between the *YrSold* and *YearRemodAdd* variables, which are the year the house was sold and the year it was remodeled, respectively. Its purpose is to see if the “age” of a house impacts its final sales price, although it’s safe to assume that newer homes (i.e., homes that have lower *Age* values) will ultimately have higher sales prices. The *Remodeling* variable is introduced as a categorical “yes” or “no” variable that indicates if *Age* is based on a remodeling date or not. If *YearBuilt* is equal to *YearRemodAdd*, it indicates if a home has had some sort of remodeling work done and can entail that particular homes may be worth less than others that aren’t based on a remodeling date.

# Figure 1

Figure 1 – Scatterplot of *SalePrice* vs. *Age*

Chart, scatter chart

Description automatically generated

The scatterplot in [Figure 1](#_Figure_1) (above) shows the negative correlation *Age* has with *SalePrice*, which helps indicate that older homes are worth less than newer ones, and the bar plot in [Figure 2](#_Figure_2) (below) compares non-remodeled homes (denoted as “None”) to remodeled ones (denoted as “Remodeling”) and shows that homes that had no remodeling work are worth more.

# Figure 2

Figure 2 – Bar Plot of *SalePrice* vs. *Remodeling*

Shape

Description automatically generated with medium confidence

Lastly, the *New* variable is introduced and uses *YrSold* and *YearBuilt* in order to indicate whether or not a home is “new” by filtering based on if a home was sold the same year that it was built, although it’s soon discovered that *New* and 24 other variables are “near-zero variance” variables, which are variables with very low frequencies of unique values, and so they were removed from the data. These, the four original bathroom variables, and *GrLivArea* (mentioned later on) were the only variables that were removed before enacting any variable selection methods (i.e., forward, backwards, and stepwise selection), and a full disclosure of which “near-zero variance” variables were removed can be seen within the [Appendix](#Near_Zero_Variance_Variables).

I then noticed that five variables had a lot of missing values, such as *Alley*, *PoolQC*, *Fence*, *MiscFeature*, and *FireplaceQu*. However, in contrast to methods performed in last semester’s project, I decided to keep these variables since these missing values were not actually “missing” but rather implied that a particular home doesn’t have a particular characteristic. For example, an “NA” for *PoolQC* simply meant that a home did not have a pool, and so pool quality was undefined for that particular home. I replaced these “NA” values with “None,” or a similar value that indicates that an observation is missing that specific characteristic, and changed *Alley*, *Fence*, and *MiscFeature* into factors, as well as *PoolQC* and *FireplaceQu* into integers which resulted in no more “missing values” for these variables. It’s also important to state that *Alley*, *PoolQC*, and *MiscFeature* were actually removed for being “near-zero variance” variables as well. Again, these techniques used for manipulating these five variables can be attributed to Erik Bruin and can be accessed via the following link: <https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda/report>.

The remaining 54 variables were filtered as either numeric or categorical variables, and the numeric variables were imputed using mean imputation, and the categorical by mode imputation, in order to fully deal with any missing values. I then recombined the two numeric and categorical groups and created a natural cubic spline (with five DF) for *GrLivArea*, a continuous numeric variable that represents the above ground living area for a home (in square feet). Five new variables were created and renamed as: *GrLivArea1*, *GrLivArea2*, etc., and the original version of the variable was dropped so that only the variables created from the spline were used for further analysis (58 total variables). [Figure 3](#_Figure_3) below shows the plot for the natural cubic spline.

# Figure 3

Figure 3 – Natural Cubic Spline of *SalePrice* vs. *GrLivArea*

*Chart, line chart

Description automatically generated*

1. **Ridge/Lasso/Elastic Net Results**

Before building models utilizing penalized regression methods, I first used the resampling method of splitting the data into train and test sets. *LotFrontage* and *LotArea* were two predictors that needed to be standardized, and *SalePrice* was reinstated into the data and standardized as well (all three by log transformation). I then used 10-fold cross-validation with forward, backwards, and stepwise variable selection methods to determine the 18 most significant predictors from the remaining 58, and I used the predictors chosen from stepwise selection since it had the best RMSE. The predictors chosen from stepwise selection can be seen in [Figure 4](#_Figure_4) below.

# Figure 4

Figure 4 – Shows the predictors chosen from stepwise variable selection and their VIF scores

Text

Description automatically generated

Next, I measured each predictor’s VIF score in order to see their correlation with one another (seen in [Figure 4](#_Table_2)). If predictors are highly correlated with each other, you may run into multicollinearity issues. Luckily, none of my predictors seemed to be highly correlated with one another since I obtained low VIF scores, although *GrLivArea3* had the highest VIF score of “6.797692” which may be attributed to the fact that it was a variable created from the spline. Still, I didn’t obtain a VIF score above “10” and penalized regression actually helps against variables that may be highly correlated anyway (by shrinking their coefficient estimates), so I found that removing more predictors was unnecessary.

I then fitted ridge regression, lasso, and elastic net models to see if I can obtain better RMSE values. 10-fold cross-validation was used for ridge regression and lasso, and repeated 10-fold cross validation (repeated five times) was used for elastic net. I obtained lambda values of “0.0326314,” “0.001479795,” and “0.004293856” for ridge, lasso, and elastic net, respectively, and I obtained an alpha of “0.2” for elastic net as well. The small lambda values imply that my predictors’ coefficients weren’t shrunk by much by these penalized regression methods. Moreover, the solution path for each model can be seen in [Figure 5](#_Figure_5) below.

# Figure 5

Figure 5 – The solution path for each penalized regression model

Chart, line chart

Description automatically generated

# Table 2

Table 2 – The RMSE values and Kaggle scores for ridge regression, lasso, and elastic net

|  |  |  |
| --- | --- | --- |
| **Predictive Model** | **RMSE** | **Kaggle Score** |
| Ridge Regression | 0.1403621 | 0.14228 |
| Lasso | 0.1402423 | 0.13987 |
| Elastic Net | 0.1358499 | 0.14013 |

The RMSE values and Kaggle scores for these regression methods can be seen in [Table 2](#_Table_2_1) above. Ridge regression had the highest RMSE and worst Kaggle score between the three while elastic net had the lowest RMSE, but it didn’t have the best Kaggle score. As a result, I decided to use my lasso model as my final model since it obtained the best Kaggle score. Since lasso can also perform variable selection as well, it removed *Electrical* and *GarageType* from my model. The final model with 16 predictors is:

= 9.9632 + (0.0429)LotArea + (0.0823)OverallQual + (0.0535)OverallCond + (0.0028)YearBuilt + (0.0001)BsmtFinSF1 + (0.0002)TotalBsmtSF + (0.0001)X2ndFlrSF + (0.0149)FireplaceQu + (0.0569)GarageCars + (0.0115)TotalBathrooms – (0.0074)BsmtExposure + (0.0951)CentralAir + (0.0038)Fence + (0.0101)Remodeling + (0.0003)GrLivArea3 + (0.00002)GrLivArea4

1. **How this analysis improves upon the project**

My analysis improves many aspects from last semester’s project. One of the biggest causes for concern from the project was that we removed observations that did not meet a certain statistical threshold (i.e., we removed observations that had studentized residual values outside of a specific range), which is not a valid reason for removal. In this analysis, I did not remove observations based on this criterion. Furthermore, I utilized feature engineering to create four new variables in hope that they would be more significant for predicting *SalePrice* compared to the individual variables that they were based on. I did not remove variables that I previously thought had large amounts of “missing values” and instead opted to manipulate values in a more informative and statistically induced way, and I didn’t remove the 27 random variables like we did in the project since it’s not a data-driven form of variable selection and can make it difficult to anticipate potentially significant relationships. I did, however, remove 25 “near-zero variance” variables since they’re considered to be less predictive. Lastly, I created a natural cubic spline (with five DF) for *GrLivArea* in order to deal with interpolation and to satisfy continuity constraints, and I did not set a model limitation of 10 predictors for my variable selection methods since doing so restricted my previous analysis’s overall predictive capabilities.

1. **Conclusion/Limitations**

My lasso model obtained the best Kaggle score and so I used it as my final model. The most significant predictors in predicting sales prices were *OverallQual*, *CentralAir*, *GarageCars*, *OverallCond*, and *LotArea*, which were consistently chosen by variable selection and penalized regression methods. The predictors that were chosen can definitely influence home sales prices and they also make a lot of sense logically since people tend to care about the overall quality and condition of their homes, their garage and lot sizes, and having central air conditioning. It was also neat to see feature engineering and spline creation “pay off” since *TotalBathrooms*, *GrLivArea3*, and *GrLivArea4* were all selected as predictors for my final model. One major trend that I noticed was that people were willing to spend more for homes that were of good quality and had a lot of space, such as for storage, cars, and other things.

In terms of limitations, setting a model restriction of 18 predictors for variable selection may have restricted my model’s predictability. However, it’s still an increase from my previous restriction of 10 predictors from the project and I was also trying to not overfit the data. I also could have done a better job looking for potential interaction effects between predictors within the data. If specific predictors affected how others relate to *SalePrice*, including interaction effects could have improved my results.

Overall, penalized regression improved upon my group’s results from the project, and I was able to obtain better Kaggle scores and more reasonable RMSE values for my models. [Figure 6](#_Figure_6_1) demonstrates my “top performing model” by Kaggle score.

# Figure 6

Figure 6 – Shows my lasso model’s Kaggle score



1. **Appendix (Code)**

### STAT 6302 Data Assignment 1 ###

#Load packages

library(dplyr)

library(tidyr)

library(plyr)

library(ggplot2)

library(gmodels)

library(agricolae)

library(multcomp)

library(Sleuth2)

library(MASS)

library(car)

library(glmnet)

library(caret)

library(leaps)

library(bestglm)

library(VIM)

library("VIM")

library(forcats)

library(stringr)

library(WriteXLS)

library(splines)

#Read in dataset

train = read.csv("train.csv")

train = data.frame(train)

test = read.csv("test.csv")

test = data.frame(test)

#Drop variables

train.drop = train[,!(names(train) %in% c("Id","SalePrice"))]

test.drop = test[,!(names(test) %in% c("Id"))]

#Combine train and test set

house = rbind(train.drop, test.drop)

# Feature Engineering #

#Combine bathroom variables (Erik Bruin)

house$TotalBathrooms = house$FullBath + (house$HalfBath\*0.5) + house$BsmtFullBath + (house$BsmtHalfBath\*0.5)

#Create "Age" and 'Remodeling' variables (Erik Bruin)

house$Age = as.numeric(house$YrSold)-house$YearRemodAdd

house$Remodeling <- ifelse(house$YearBuilt==house$YearRemodAdd, 0, 1)

house$Remodeling[house$Remodeling == 0] <-"None"

house$Remodeling[house$Remodeling == 1] <-"Remodeling"

#Create New variable (Erik Bruin)

house$New <- ifelse(house$YrSold==house$YearBuilt, 1, 0)

table(house$New) #Way more old houses than new ones

#Keep only 'TotalBathrooms' from the bathroom variables

house = subset(house, select = -c(FullBath,HalfBath,BsmtFullBath,BsmtHalfBath))

str(house)

#Correlation between 'SalePrice' and bathroom variables

#SalePreice ~ FullBath

SalePrice = train$SalePrice

FullBath = train$FullBath

cor(FullBath, SalePrice) #r=0.5606638

#SalePreice ~ HalfBath

HalfBath = train$HalfBath

cor(FullBath, SalePrice) #r=0.5606638

#SalePreice ~ BsmtFullBath

BsmtFullBath = train$BsmtFullBath

cor(BsmtFullBath, SalePrice) #r=0.2271222

#SalePreice ~ BsmtHalfBath

BsmtHalfBath = train$BsmtHalfBath

cor(BsmtHalfBath, SalePrice) #r=-0.01684415

#See which variables have the greatest amount of NA values and see how to deal w/them

sapply(house, function(x) sum(is.na(x))) #Alley: 2721, PoolQC: 2909, Fence: 2348, MiscFeature: 2814, FireplaceQu: 1420

#Make 'Alley' into a factor to get rid of NAs

#Erik Bruin

house$Alley[house$Alley == "Grvl"] <-'Gravel'

house$Alley[house$Alley == "Pave"] <-'Paved'

house$Alley[is.na(house$Alley)] <- 'No Alley'

house$Alley <- as.factor(house$Alley)

str(house$Alley)

#For 'PoolQC,' get rid of NAs by changing 'NA' to 'None' and since ordinal change to integer

#Erik Bruin

house$PoolQC[is.na(house$PoolQC)] <- 'None'

Revalue <- c('None' = 0, 'Po' = 1, 'Fa' = 2, 'TA' = 3, 'Gd' = 4, 'Ex' = 5)

house$PoolQC<-as.integer(revalue(house$PoolQC, Revalue))

str(house$PoolQC)

#For 'Fence,' get rid of NAs by changing 'NA' to 'No Fence' and convert to factor

#Erik Bruin

house$Fence[is.na(house$Fence)] <- 'No Fence'

house$Fence <- as.factor(house$Fence)

str(house$Fence)

#For 'MiscFeature,' get rid of NAs by changing 'NA' to 'None' and convert to factor (there's no "Elev" value, so will become 'factor w/ 5 levels')

#Erik Bruin

house$MiscFeature[is.na(house$MiscFeature)] <- 'None'

house$MiscFeature <- as.factor(house$MiscFeature)

str(house$MiscFeature)

#For 'FireplaceQu,' get rid of NAs by changing 'NA' to 'None' and since ordinal change to integer

#Erik Bruin

house$FireplaceQu[is.na(house$FireplaceQu)] <- 'None'

house$FireplaceQu<-as.integer(revalue(house$FireplaceQu, Revalue))

str(house$FireplaceQu)

#These 5 variables don't have NA values anymore: Alley, PoolQC, Fence, MiscFeature, and FireplaceQu

sapply(house, function(x) sum(is.na(x)))

# Near Zero Variance #

near\_zero = nearZeroVar(house)

near\_zero

#The following variables have "near-zero variance" and are removed from the data

house = house[-c(5,6,8,9,11,14,22,31,35,36,39,45,48,51,59,60,63,64,65,66,67,68,70,71,79)]

str(house)

#Street,Alley,LandContour,Utilities,LandSlope,Condition2,RoofMatl,BsmtCond,BsmtFinType2,

#BsmtFinSF2,Heating,LowQualFinSF,KitchenAbvGr,Functional,GarageQual,GarageCond,OpenPorchSF,

#EnclosedPorch,X3SsnPorch,ScreenPorch,PoolArea,PoolQC,MiscFeature,MiscVal,New

#See which variables are numeric and impute by the mean

num = house[sapply(house,is.numeric)]

str(num)

num = kNN(num, k = 10, numFun = mean)

num = num[,-c(27:52)]

#Catregorical variables

categorical = house[!sapply(house,is.numeric)]

str(categorical)

categorical = kNN(categorical, k = 10, numFun = mode)

categorical = categorical[,-c(29:56)]

#Combine (no more NAs)

house = cbind(num, categorical)

sapply(house, function(x) sum(is.na(x)))

str(house) #54 variables

# Natural Cubic Spline #

#Plot to see for nonlinear relationship of SalePrice vs. GrLivArea

GrLivArea = train$GrLivArea

SalePrice = train$SalePrice

plot(GrLivArea, SalePrice, cex=0.5, col="darkgrey")

#Create a grid of values for 'GrLivArea' at which we want predictions

lims = range(GrLivArea)

GrLivArea.grid = seq(from=lims[1], to=lims[2])

#Fit

sp.fit = lm(train$SalePrice ~ ns(GrLivArea, df = 5), data = train)

sp.pred <- predict(sp.fit, newdata=list(GrLivArea=GrLivArea.grid), se=TRUE)

se.bands <- cbind(sp.pred$fit+2\*sp.pred$se.fit, sp.pred$fit-2\*sp.pred$se.fit)

#Plot spline

plot(GrLivArea, train$SalePrice, xlim=lims, cex=.5, col="darkgrey",

main="Natural Cubic Spline",

xlab="GrLivArea",

ylab="SalePrice")

lines(GrLivArea.grid, sp.pred$fit, lwd=2, col="red")

matlines(GrLivArea.grid, se.bands, lwd=1, col="blue", lty=3)

#Create a natural spline with 5 df for the 'GrLivArea' column

spline = ns(house$GrLivArea, df=5)

#Combine the original data with the spline data

house = cbind(house, spline)

#Give the basis functions more reasonable column names

colnames(house)[55:59] = c('GrLivArea1', 'GrLivArea2', 'GrLivArea3',

'GrLivArea4', 'GrLivArea5')

#Drop original 'GrLivArea' column

drop = c('GrLivArea')

house = house[,!(names(house) %in% drop)]

#Make all variables numeric

house = data.frame(lapply(house, function(x) as.numeric(as.factor(x))))

str(house) #58 variables

#Train and test

set.seed(100)

train.index = nrow(train)

test.index = train.index + 1

total = train.index + nrow(test)

#log variables

house$LotFrontage = log(house$LotFrontage)

house$LotArea = log(house$LotArea)

#Create actual train and test sets again

final.train = house[1:train.index, ]

final.test = house[test.index:total, ]

#Reinstate SalePrice

final.train$SalePrice = train$SalePrice

#Correlation between SalePrice vs. TotalBathroom

SalePrice = final.train$SalePrice

TotalBathroom = final.train$TotalBathroom

cor(TotalBathroom, SalePrice) #0.6277554

#Plot showing negative correlation with 'Age' and 'SalePrice'

ggplot(data=final.train[!is.na(SalePrice),], aes(x=Age, y=SalePrice))+

geom\_point(col='black') + geom\_smooth(method = "lm", se=FALSE, color="blue", aes(group=1)) +

scale\_y\_continuous(breaks= seq(0, 800000, by=100000)) +

ggtitle("Scatterplot of SalePrice vs. Age") +

theme(plot.title = element\_text(hjust = 0.5))

#Make 'Remodeling' a character variable

final.train$Remodeling = as.character(final.train$Remodeling)

final.train$Remodeling[final.train$Remodeling == "1"] <-"None"

final.train$Remodeling[final.train$Remodeling == "2"] <-"Remodeling"

#Plot showing how remodeled homes are worth less

ggplot(final.train[!is.na(final.train$SalePrice),], aes(x=Remodeling, y=SalePrice, colour = factor(Remodeling), shape = factor(Remodeling))) +

geom\_bar(stat="identity", fill='blue') +

geom\_label(stat = "count", aes(label = ..count.., y = ..count..), size=6) +

scale\_y\_continuous(breaks= seq(0, 800000, by=50000)) +

theme\_grey(base\_size = 18)

#Change back

final.train$Remodeling = as.numeric(final.train$Remodeling)

#Shows negative correlation of SalePrice vs. Age

cor(final.train$SalePrice[!is.na(final.train$SalePrice)], final.train$Age[!is.na(final.train$SalePrice)])

#Distribution of Sales Prices of Homes in Ames (not normally distributed)

ggplot(train, aes(x=SalePrice)) + geom\_histogram(color="white", fill="black") +

labs(title="Distribution of Sales Prices of Homes in Ames, Iowa",x="Sales Price", y = "Frequency")+

theme\_classic() #not normally distributed

#Take the log of SalePrice

final.train$SalePrice = log(train$SalePrice)

#Backwards variable selection (specify 10-fold cross-validation first)

train.control <- trainControl(method = "cv", number = 10)

back.model <- train(SalePrice ~ ., data = final.train,

method = "leapBackward",

tuneGrid = data.frame(nvmax = 1:18),

trControl = train.control)

back.model$bestTune

back.model$results[18,] #RMSE

summary(back.model$finalModel)

coef(back.model$finalModel, 18)

#Forward variable selection

forward.model <- train(SalePrice ~ ., data = final.train,

method = "leapForward",

tuneGrid = data.frame(nvmax = 1:18),

trControl = train.control)

forward.model$bestTune

forward.model$results[18,] #RMSE

summary(forward.model$finalModel)

coef(forward.model$finalModel, 18)

#Stepwise variable selection

step.model <- train(SalePrice ~ ., data = final.train,

method = "leapSeq",

tuneGrid = data.frame(nvmax = 1:18),

trControl = train.control)

step.model$bestTune

step.model$results[18,] #RMSE

summary(step.model$finalModel)

coef(step.model$finalModel, 18)

#Compare variable selection methods' RMSEs

#Backward

back.model$results[18,]

#Forward

forward.model$results[18,]

#Stepwise

step.model$results[18,]

#Stepwise has the best RMSE, so we will use the variables selected from it for penalized regression

coef(step.model$finalModel, 18)

final.train = final.train %>% dplyr::select(LotArea,OverallQual,OverallCond,YearBuilt,BsmtFinSF1,TotalBsmtSF,X2ndFlrSF,FireplaceQu,GarageCars,TotalBathrooms,BsmtExposure,CentralAir,Electrical,GarageType,Fence,Remodeling,GrLivArea3,GrLivArea4,SalePrice)

final.test = final.test %>% dplyr::select(LotArea,OverallQual,OverallCond,YearBuilt,BsmtFinSF1,TotalBsmtSF,X2ndFlrSF,FireplaceQu,GarageCars,TotalBathrooms,BsmtExposure,CentralAir,Electrical,GarageType,Fence,Remodeling,GrLivArea3,GrLivArea4)

#Based off of variables chosen from stepwise variable selection

model = lm(SalePrice ~ LotArea+OverallQual+OverallCond+YearBuilt+BsmtFinSF1+TotalBsmtSF+X2ndFlrSF+FireplaceQu+GarageCars+TotalBathrooms+BsmtExposure+CentralAir+Electrical+GarageType+Fence+Remodeling+GrLivArea3+GrLivArea4, data=final.train)

summary(model)

#Check VIF scores to see which variables have inflated standard errors

vif(model)

res = cor(final.train) #correlation between variables

# Penalized Regression Methods #

#Ridge Regression (note that cv.glmnet by default does 10-fold cross-validation)

rr.glmnet = cv.glmnet(as.matrix(final.train[1:18]), final.train$SalePrice, alpha=0)

attributes(rr.glmnet)

best.lambda <- rr.glmnet$lambda.min #Optimal tuning parameter

ridge.coef = coef(rr.glmnet, s=best.lambda)

#RMSE

sqrt(rr.glmnet$cvm[rr.glmnet$lambda == rr.glmnet$lambda.1se])

#Predict

ridge.pred = predict(rr.glmnet, as.matrix(final.test), s=best.lambda)

#Lasso

lasso.glmnet = cv.glmnet(as.matrix(final.train[,1:18]), final.train$SalePrice, alpha=1)

attributes(lasso.glmnet)

best.lambda2 = lasso.glmnet$lambda.min #Optimal tuning parameter

lasso.coef = coef(lasso.glmnet, s=best.lambda2)

lasso.coef[lasso.coef !=0]

#RMSE

sqrt(lasso.glmnet$cvm[lasso.glmnet$lambda == lasso.glmnet$lambda.1se])

#Predict

lasso.pred = predict(lasso.glmnet, newx=as.matrix(final.test), s=best.lambda2)

#Elastic Net

tcontrol = trainControl(method="repeatedcv", number=10, repeats=5)

en.glmnet = train(as.matrix(final.train[,1:18]), final.train$SalePrice, trControl=tcontrol,

method="glmnet", tuneLength=10)

attributes(en.glmnet)

en.glmnet$results

en.glmnet$bestTune #Optimal tuning parameter

en.glmnet2 = en.glmnet$finalModel

en.coef = coef(en.glmnet2, s=en.glmnet$bestTune$lambda)

#RMSE

min(en.glmnet$results$RMSE)

#Predict

en.pred = predict(en.glmnet2, as.matrix(final.test), s=en.glmnet$bestTune$lambda)

#Solution paths for penalized regression

par(mfrow=c(2,2))

#Ridge

ridge.mod = glmnet(as.matrix(final.train[,1:18]), final.train$SalePrice, alpha=0)

plot(ridge.mod)

title("Ridge Regression Solution Path", line = 2.5)

#Lasso

lasso.mod = glmnet(as.matrix(final.train[,1:18]), final.train$SalePrice, alpha=1)

plot(lasso.mod)

title("Lasso Solution Path", line = 2.5)

#EN

plot(en.glmnet2)

title("Elastic Net Solution Path", line = 2.5)

#Compare penalized regression models' RMSEs and Kaggle Scores

#Ridge

sqrt(rr.glmnet$cvm[rr.glmnet$lambda == rr.glmnet$lambda.1se]) #Kaggle Score = 0.14228

#Lasso

sqrt(lasso.glmnet$cvm[lasso.glmnet$lambda == lasso.glmnet$lambda.1se]) #Kaggle Score = 0.13987

#EN

min(en.glmnet$results$RMSE) #Kaggle Score = 0.14013

#Elastic Net has the lowest RMSE but Lasso has the best Kaggle score, so I'll use Lasso

#Final model

lasso.coef #'Electrical' and 'GarageType' are removed

#Write Ridge Predictions as Excel file

WriteXLS(data.frame(ridge.pred))

#Write Lasso Predictions as Excel file

WriteXLS(data.frame(lasso.pred))

#Write EN Predictions as Excel file

WriteXLS(data.frame(en.pred))

#I converted the log values in Excel before submitting to Kaggle