Final\_Project

Connor Hayes

December 5, 2018

# Overview:

##### The dataset comprised of housing data related to home sales in Ames, Iowa from 2006 - 2010. I found the data on kaggle, at the link below, and it look to me a commonly used data set for statistical analysis. The aim of the dataset is to better predict the sells price of a house.

<https://www.kaggle.com/c/house-prices-advanced-regression-techniques>

house = read.csv('housedata.csv', header = TRUE)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(ggplot2)  
library(RANN)  
library(ranger)  
library(rpart)  
library(e1071)

The target variable is the house sales price. The sales price was originally a numeric variable, so to make it work with the models, we made it categorical by dividing the data. The data is split into house that were sold for above or below 140k. I used $140K because thats what Zillow estimated the median home in Ames, Iowa cost in 2010. The model to be able to predict if a house was cheap or expensive compared to Zillows median price. The target variable is coded as 1 and 2. 1 is below average and 2 is above average.

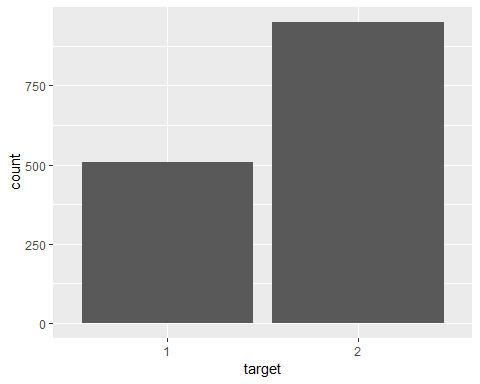
This data set orginally had about 80 variables, but to simiplify the data I reduced it down to 18 variables that I felt were the most important. It has 5 numeric, 12 categorical, and the target variable. The variable reprsent factors like year the house was sold, year it was remodelled, number of rooms, garage size, lot area, neighborhood, overall quailty, etc. Each of the varaibles is unique enough that they should be strong predictors of house sales price.

names(house)[1] = "target"  
house$target[house$target <= 140000] <- 1  
house$target[house$target>140000] <- 2  
house$target = as.factor(house$target)  
house$YrSold = as.factor(house$YrSold)  
house$YearRemodAdd = as.factor(house$YearRemodAdd)  
house$MoSold = as.factor(house$MoSold)  
house$TotRmsAbvGrd = as.factor(house$TotRmsAbvGrd)  
house$GarageCars = as.factor(house$GarageCars)  
house$OverallQual = as.factor(house$OverallQual)  
summary(house)

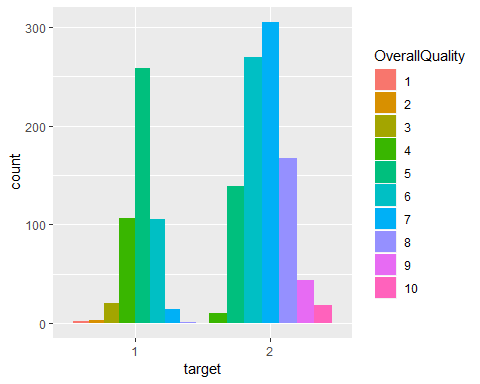
## target MSSubClass LotFrontage LotArea LotConfig   
## 1:509 Min. : 20.0 Min. : 21.00 Min. : 1300 Corner : 263   
## 2:951 1st Qu.: 20.0 1st Qu.: 59.00 1st Qu.: 7554 CulDSac: 94   
## Median : 50.0 Median : 69.00 Median : 9478 FR2 : 47   
## Mean : 56.9 Mean : 70.05 Mean : 10517 FR3 : 4   
## 3rd Qu.: 70.0 3rd Qu.: 80.00 3rd Qu.: 11602 Inside :1052   
## Max. :190.0 Max. :313.00 Max. :215245   
## NA's :259   
## Neighborhood BldgType HouseStyle OverallQual YearRemodAdd  
## NAmes :225 1Fam :1220 1Story :726 5 :397 1950 :178   
## CollgCr:150 2fmCon: 31 2Story :445 6 :374 2006 : 97   
## OldTown:113 Duplex: 52 1.5Fin :154 7 :319 2007 : 76   
## Edwards:100 Twnhs : 43 SLvl : 65 8 :168 2005 : 73   
## Somerst: 86 TwnhsE: 114 SFoyer : 37 4 :116 2004 : 62   
## Gilbert: 79 1.5Unf : 14 9 : 43 2000 : 55   
## (Other):707 (Other): 19 (Other): 43 (Other):919   
## Exterior1st MasVnrArea BsmtFinSF1 TotRmsAbvGrd GarageCars  
## VinylSd:515 Min. : 0.0 Min. : 0.0 6 :402 0: 81   
## HdBoard:222 1st Qu.: 0.0 1st Qu.: 0.0 7 :329 1:369   
## MetalSd:220 Median : 0.0 Median : 383.5 5 :275 2:824   
## Wd Sdng:206 Mean : 103.7 Mean : 443.6 8 :187 3:181   
## Plywood:108 3rd Qu.: 166.0 3rd Qu.: 712.2 4 : 97 4: 5   
## CemntBd: 61 Max. :1600.0 Max. :5644.0 9 : 75   
## (Other):128 NA's :8 (Other): 95   
## PavedDrive MoSold YrSold   
## N: 90 6 :253 2006:314   
## P: 30 7 :234 2007:329   
## Y:1340 5 :204 2008:304   
## 4 :141 2009:338   
## 8 :122 2010:175   
## 3 :106   
## (Other):400

This chart shows the break down of the target variable

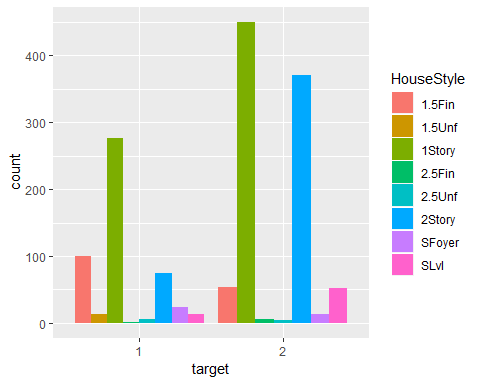
ggplot(data=house)+geom\_bar(mapping = aes(x=house$target))+labs(x='target')

 The chart shows the break down of housing quality by target variable. There is a noticable difference between quality rating for the two target variables.

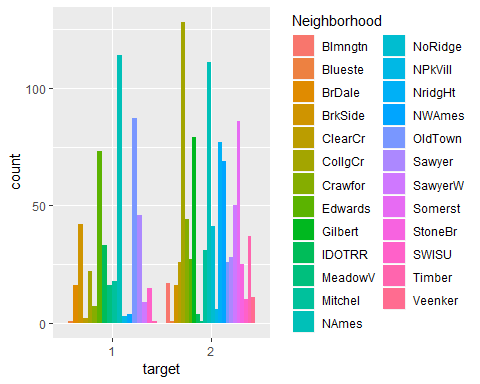
ggplot(data=house)+ geom\_bar(mapping = aes(x=house$target, fill=house$OverallQual), position = "dodge") + labs(x='target', fill = 'OverallQuality')

 Compares the style of the house, mostly the number of floors, to the target variable. There is again a noticable difference between the types of houses for represented by the target variables.

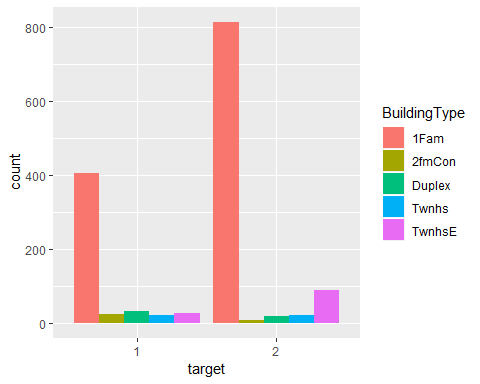
ggplot(data=house)+ geom\_bar(mapping = aes(x=house$target, fill=house$HouseStyle), position = "dodge") + labs(x='target', fill = 'HouseStyle')

 This chart shows the number of houses sold in each neighborhood for each target variable. There are some major differences between the two target variables.

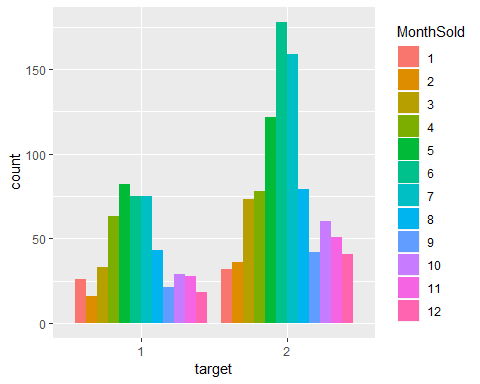
ggplot(data=house)+ geom\_bar(mapping = aes(x=house$target, fill=house$Neighborhood), position = "dodge") + labs(x='target', fill = 'Neighborhood')

 Shows the breakdown of type of building for each target variable. Most of the houses were single famly homes.

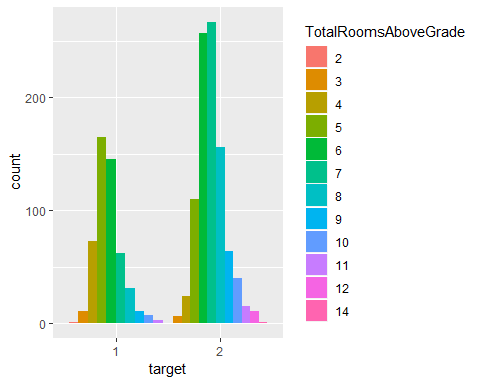
ggplot(data=house)+ geom\_bar(mapping = aes(x=house$target, fill=house$BldgType), position = "dodge") + labs(x='target', fill = 'BuildingType')

 The chart shows how many houses were sold in each month for the target variables. The spring and summer months were the most common.

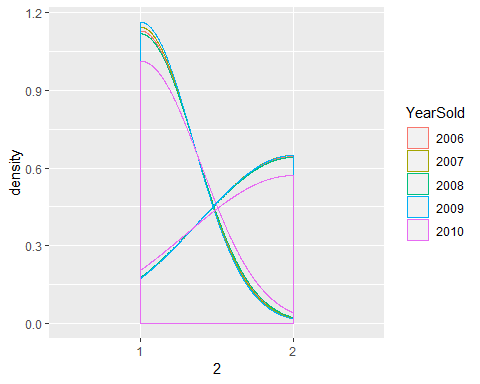
ggplot(data=house)+ geom\_bar(mapping = aes(x=house$target, fill=house$MoSold), position = "dodge") + labs(x='target', fill = 'MonthSold')

 This chart shows the number of rooms per house for each target variable. There are noticable differences in the number of rooms between the two target variables.

ggplot(data=house)+ geom\_bar(mapping = aes(x=house$target, fill=house$TotRmsAbvGrd), position = "dodge") + labs(x='target', fill = 'TotalRoomsAboveGrade')

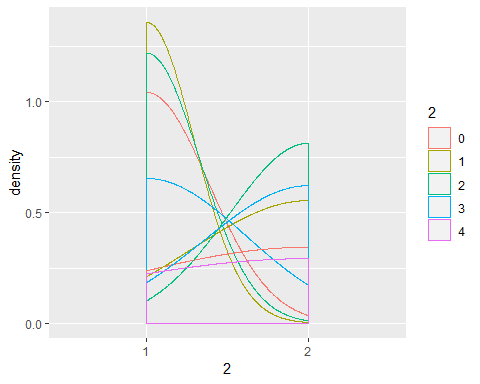
 This chart shows the density of the number of houses sold for each target variable.

ggplot(data=house)+geom\_density(mapping = aes(x=house$target,color=house$YrSold))+labs(x=house$target,color='YearSold')

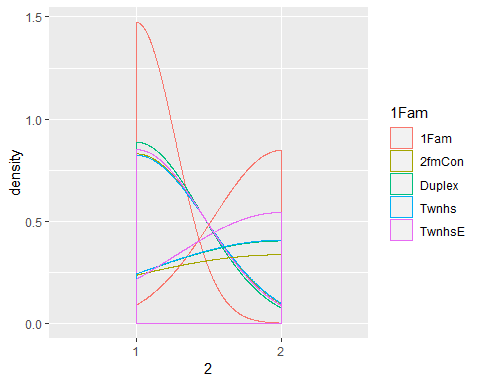
 This chart shows the number of car garage for each of the target variables. The more expensive house tend to have larger car garages.

ggplot(data=house)+geom\_density(mapping = aes(x=house$target,color=house$GarageCars))+labs(x=house$target,color=house$GarageCars)

## Warning: Groups with fewer than two data points have been dropped.

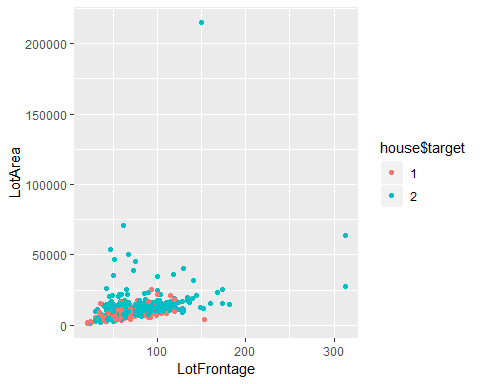
 This chart shows the density of building type for each target variable. Most of the home sales were of single family homes.

ggplot(data=house)+geom\_density(mapping = aes(x=house$target,color=house$BldgType))+labs(x=house$target,color=house$BldgType)

 This plot compares lot frontage to lot area with different colors each target variable. The more expensive homes had more variablilty in area and frontage.

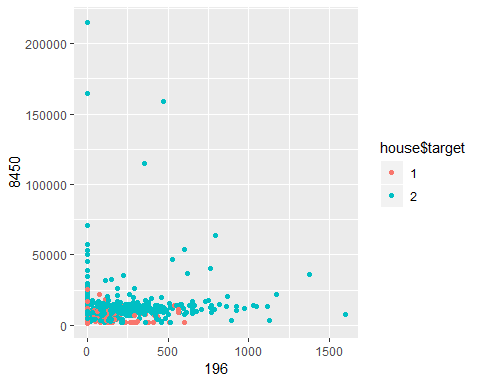
ggplot(data=house)+geom\_point(mapping=aes(x=house$LotFrontage, y=house$LotArea, color = house$target))+labs(x='LotFrontage',y='LotArea')

## Warning: Removed 259 rows containing missing values (geom\_point).

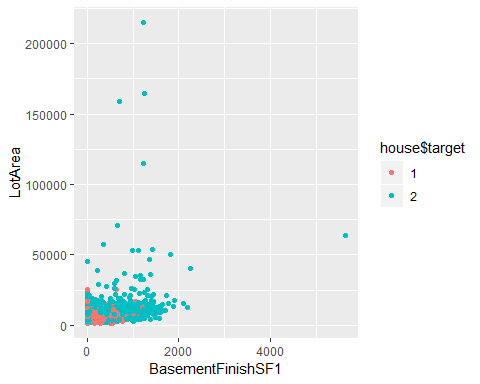
 The plot compares masonary area to lot area with different colors for each target variable. The more expensive houses were again much more spread out compared to the cheaper houses.

ggplot(data=house)+geom\_point(mapping=aes(x=house$MasVnrArea, y=house$LotArea, color = house$target))+labs(x=house$MasVnrArea,y=house$LotArea)

## Warning: Removed 8 rows containing missing values (geom\_point).

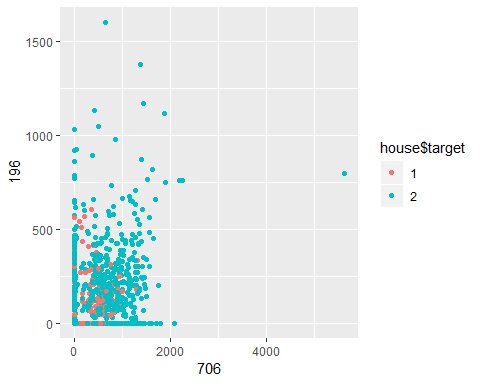
 This plot compare the basement finished area with the colors representing the different target variables. Most of the data is bunched together expect for a handful of outliers.

ggplot(data=house)+geom\_point(mapping=aes(x=house$BsmtFinSF1, y=house$LotArea, color = house$target))+labs(x='BasementFinishSF1',y='LotArea')

 This plot compares finished basement area to masonary area with the target variable represented by the colors. Most of the houses have within the same range of masonary area except a few outliers.

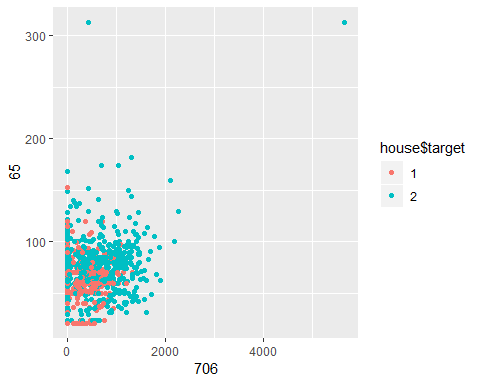
ggplot(data=house)+geom\_point(mapping=aes(x=house$BsmtFinSF1, y=house$MasVnrArea, color = house$target))+labs(x=house$BsmtFinSF1,y=house$MasVnrArea)

## Warning: Removed 8 rows containing missing values (geom\_point).

 This plot compares finished basement area to lot frontage with the target variable represented by the colors. Most of the houses have within the same range of finished basement area except a few outliers.

ggplot(data=house)+geom\_point(mapping=aes(x=house$BsmtFinSF1, y=house$LotFrontage, color = house$target))+labs(x=house$BsmtFinSF1,y=house$LotFrontage)

## Warning: Removed 259 rows containing missing values (geom\_point).



house1 = house

### Handling Missing Data

The first way I handled the missing data was to eliminate the observations that had any missing data. There is not that much missing data with only a few variable missing data, so eliminating some data should not have a big impact on the model overall. The second method I just was the KnnImpute. Third was the Median Impute method.

house1 = house1[complete.cases(house1), ]  
sum(is.na(house1))

## [1] 0

preProcess\_knn <- preProcess(house, method='knnImpute')  
knnhouse <- predict(preProcess\_knn, newdata = house)  
sum(is.na(knnhouse))

## [1] 0

preProcess\_med<- preProcess(house,method ='medianImpute')  
medhouse <- predict(preProcess\_med, newdata = house)  
sum(is.na(medhouse))

## [1] 0

#### Baseline Models.

These models run basic ranger, random forest models for each of the 3 immpuation methods. This will set a base for which model is likely to be the best predictor.

set.seed(10)  
model1a <- train(target~.,data = house1, method = "ranger",   
 trControl = trainControl(method ="cv", number = 7, verboseIter = TRUE))

## + Fold1: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold1: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold1: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold1: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold1: mtry=159, min.node.size=1, splitrule=gini   
## - Fold1: mtry=159, min.node.size=1, splitrule=gini   
## + Fold1: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold1: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold1: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold2: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold2: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold2: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold2: mtry=159, min.node.size=1, splitrule=gini   
## - Fold2: mtry=159, min.node.size=1, splitrule=gini   
## + Fold2: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold3: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold3: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold3: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold3: mtry=159, min.node.size=1, splitrule=gini   
## - Fold3: mtry=159, min.node.size=1, splitrule=gini   
## + Fold3: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold4: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold4: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold4: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold4: mtry=159, min.node.size=1, splitrule=gini   
## - Fold4: mtry=159, min.node.size=1, splitrule=gini   
## + Fold4: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold5: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold5: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold5: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold5: mtry=159, min.node.size=1, splitrule=gini   
## - Fold5: mtry=159, min.node.size=1, splitrule=gini   
## + Fold5: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold6: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold6: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold6: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold6: mtry=159, min.node.size=1, splitrule=gini   
## - Fold6: mtry=159, min.node.size=1, splitrule=gini   
## + Fold6: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold7: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold7: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold7: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold7: mtry=159, min.node.size=1, splitrule=gini   
## - Fold7: mtry=159, min.node.size=1, splitrule=gini   
## + Fold7: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry=159, min.node.size=1, splitrule=extratrees   
## Aggregating results  
## Selecting tuning parameters  
## Fitting mtry = 80, splitrule = gini, min.node.size = 1 on full training set

model1b <- train(target~.,data = knnhouse, method = "ranger",   
 trControl = trainControl(method ="cv", number = 7, verboseIter = TRUE))

## + Fold1: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold1: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold1: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold1: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold1: mtry=159, min.node.size=1, splitrule=gini   
## - Fold1: mtry=159, min.node.size=1, splitrule=gini   
## + Fold1: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold1: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold1: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold2: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold2: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold2: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold2: mtry=159, min.node.size=1, splitrule=gini   
## - Fold2: mtry=159, min.node.size=1, splitrule=gini   
## + Fold2: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold3: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold3: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold3: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold3: mtry=159, min.node.size=1, splitrule=gini   
## - Fold3: mtry=159, min.node.size=1, splitrule=gini   
## + Fold3: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold4: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold4: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold4: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold4: mtry=159, min.node.size=1, splitrule=gini   
## - Fold4: mtry=159, min.node.size=1, splitrule=gini   
## + Fold4: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold5: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold5: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold5: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold5: mtry=159, min.node.size=1, splitrule=gini   
## - Fold5: mtry=159, min.node.size=1, splitrule=gini   
## + Fold5: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold6: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold6: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold6: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold6: mtry=159, min.node.size=1, splitrule=gini   
## - Fold6: mtry=159, min.node.size=1, splitrule=gini   
## + Fold6: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold7: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold7: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold7: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold7: mtry=159, min.node.size=1, splitrule=gini   
## - Fold7: mtry=159, min.node.size=1, splitrule=gini   
## + Fold7: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry=159, min.node.size=1, splitrule=extratrees   
## Aggregating results  
## Selecting tuning parameters  
## Fitting mtry = 80, splitrule = gini, min.node.size = 1 on full training set

model1c <- train(target~.,data = medhouse, method = "ranger",   
 trControl = trainControl(method ="cv", number = 7, verboseIter = TRUE))

## + Fold1: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold1: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold1: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold1: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold1: mtry=159, min.node.size=1, splitrule=gini   
## - Fold1: mtry=159, min.node.size=1, splitrule=gini   
## + Fold1: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold1: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold1: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold2: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold2: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold2: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold2: mtry=159, min.node.size=1, splitrule=gini   
## - Fold2: mtry=159, min.node.size=1, splitrule=gini   
## + Fold2: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold3: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold3: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold3: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold3: mtry=159, min.node.size=1, splitrule=gini   
## - Fold3: mtry=159, min.node.size=1, splitrule=gini   
## + Fold3: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold4: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold4: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold4: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold4: mtry=159, min.node.size=1, splitrule=gini   
## - Fold4: mtry=159, min.node.size=1, splitrule=gini   
## + Fold4: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold5: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold5: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold5: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold5: mtry=159, min.node.size=1, splitrule=gini   
## - Fold5: mtry=159, min.node.size=1, splitrule=gini   
## + Fold5: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold6: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold6: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold6: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold6: mtry=159, min.node.size=1, splitrule=gini   
## - Fold6: mtry=159, min.node.size=1, splitrule=gini   
## + Fold6: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold7: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold7: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold7: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold7: mtry=159, min.node.size=1, splitrule=gini   
## - Fold7: mtry=159, min.node.size=1, splitrule=gini   
## + Fold7: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry=159, min.node.size=1, splitrule=extratrees   
## Aggregating results  
## Selecting tuning parameters  
## Fitting mtry = 159, splitrule = gini, min.node.size = 1 on full training set

print(model1a)

## Random Forest   
##   
## 1195 samples  
## 17 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Cross-Validated (7 fold)   
## Summary of sample sizes: 1024, 1024, 1024, 1025, 1024, 1024, ...   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa   
## 2 gini 0.8669615 0.7099560  
## 2 extratrees 0.8552460 0.6833086  
## 80 gini 0.8761561 0.7371369  
## 80 extratrees 0.8636149 0.7142781  
## 159 gini 0.8661261 0.7161442  
## 159 extratrees 0.8577719 0.7014538  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 80, splitrule = gini  
## and min.node.size = 1.

print(model1b)

## Random Forest   
##   
## 1460 samples  
## 17 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Cross-Validated (7 fold)   
## Summary of sample sizes: 1251, 1252, 1252, 1251, 1252, 1251, ...   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa   
## 2 gini 0.8664165 0.6936131  
## 2 extratrees 0.8493086 0.6510574  
## 80 gini 0.8801100 0.7382768  
## 80 extratrees 0.8670803 0.7132367  
## 159 gini 0.8725912 0.7223240  
## 159 extratrees 0.8623055 0.7019881  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 80, splitrule = gini  
## and min.node.size = 1.

print(model1c)

## Random Forest   
##   
## 1460 samples  
## 17 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Cross-Validated (7 fold)   
## Summary of sample sizes: 1251, 1251, 1251, 1251, 1253, 1251, ...   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa   
## 2 gini 0.8616639 0.6815176  
## 2 extratrees 0.8438230 0.6373924  
## 80 gini 0.8704902 0.7182621  
## 80 extratrees 0.8609407 0.6989656  
## 159 gini 0.8732409 0.7244061  
## 159 extratrees 0.8596034 0.6945633  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 159, splitrule = gini  
## and min.node.size = 1.

All three models produced similar results, but the the Knn data set produced the best accuracy reading of 0.88 with a mtry of 80, splitrule = gini and min.node.size =1. The second best result was the median data set with the same tuning parameters.

### Recoding

I choose to recoded a few variables that had more then 5 levels. The recoded variables included mounth sold, year remodeled, Neighborhood, and Total rooms. The categories were reduced for each variable, with months grouped by season, years reduced by decades or half-decades, and others recoded to reduce the number of levels.

Rechouse1 = house1  
Recknnhouse = knnhouse  
Recmedhouse = medhouse

levels(Rechouse1$TotRmsAbvGrd) = c("4", "4", "4", "5", "6", "7", "8", "9", "10", "10", "10", "10")  
levels(Rechouse1$MoSold) = c("W", "W", "Sp", "Sp", "Sp", "S", "S", "S", "F", "F", "F", "W")  
levels(Rechouse1$YearRemodAdd) = c("1950", "1950", "1950", "1950", "1950", "1950", "1950", "1950", "1950", "1950", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "2000", "2000", "2000", "2000", "2000", "2000", "2006", "2006", "2006", "2006", "2010")  
levels(Rechouse1$HouseStyle) = c("15", "15", "1", "25", "25", "2", "1", "1" )  
levels(Rechouse1$Neighborhood) = c("B", "B", "B", "B", "C", "C", "C", "C", "M", "M", "M", "M", "N", "N", "N", "N", "N", "S", "S", "S", "S", "S", "S", "T", "T")  
summary(Rechouse1)

## target MSSubClass LotFrontage LotArea LotConfig   
## 1:451 Min. : 20.00 Min. : 21.00 Min. : 1300 Corner :200   
## 2:744 1st Qu.: 20.00 1st Qu.: 59.00 1st Qu.: 7418 CulDSac: 45   
## Median : 50.00 Median : 69.00 Median : 9250 FR2 : 33   
## Mean : 57.23 Mean : 70.03 Mean : 9954 FR3 : 4   
## 3rd Qu.: 70.00 3rd Qu.: 80.00 3rd Qu.: 11248 Inside :913   
## Max. :190.00 Max. :313.00 Max. :215245   
##   
## Neighborhood BldgType HouseStyle OverallQual YearRemodAdd  
## B: 83 1Fam :989 15:149 5 :333 2000 :248   
## C:271 2fmCon: 28 1 :668 6 :282 1950 :239   
## M:134 Duplex: 47 25: 18 7 :262 2006 :219   
## N:346 Twnhs : 40 2 :360 8 :141 1990 :201   
## S:324 TwnhsE: 91 4 : 98 1970 :128   
## T: 37 9 : 42 1960 :101   
## (Other): 37 (Other): 59   
## Exterior1st MasVnrArea BsmtFinSF1 TotRmsAbvGrd GarageCars  
## VinylSd:443 Min. : 0.0 Min. : 0.0 4 : 87 0: 74   
## MetalSd:189 1st Qu.: 0.0 1st Qu.: 0.0 5 :229 1:312   
## Wd Sdng:168 Median : 0.0 Median : 351.0 6 :337 2:637   
## HdBoard:160 Mean : 102.7 Mean : 425.8 7 :254 3:168   
## Plywood: 71 3rd Qu.: 160.0 3rd Qu.: 689.5 8 :162 4: 4   
## CemntBd: 52 Max. :1600.0 Max. :5644.0 9 : 61   
## (Other):112 10: 65   
## PavedDrive MoSold YrSold   
## N: 81 W :139 2006:264   
## P: 25 Sp:361 2007:265   
## Y:1089 S :501 2008:251   
## F :194 2009:272   
## 2010:143   
##   
##

levels(Recknnhouse$TotRmsAbvGrd) = c("4", "4", "4", "5", "6", "7", "8", "9", "10", "10", "10", "10")  
levels(Recknnhouse$MoSold) = c("W", "W", "Sp", "Sp", "Sp", "S", "S", "S", "F", "F", "F", "W")  
levels(Recknnhouse$YearRemodAdd) = c("1950", "1950", "1950", "1950", "1950", "1950", "1950", "1950", "1950", "1950", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "2000", "2000", "2000", "2000", "2000", "2000", "2006", "2006", "2006", "2006", "2010")  
levels(Recknnhouse$HouseStyle) = c("15", "15", "1", "25", "25", "2", "1", "1" )  
levels(Recknnhouse$Neighborhood) = c("B", "B", "B", "B", "C", "C", "C", "C", "M", "M", "M", "M", "N", "N", "N", "N", "N", "S", "S", "S", "S", "S", "S", "T", "T")  
summary(Recknnhouse)

## target MSSubClass LotFrontage LotArea   
## 1:509 Min. :-0.8723 Min. :-2.019784 Min. :-0.9234   
## 2:951 1st Qu.:-0.8723 1st Qu.:-0.413838 1st Qu.:-0.2969   
## Median :-0.1631 Median :-0.002057 Median :-0.1040   
## Mean : 0.0000 Mean : 0.030406 Mean : 0.0000   
## 3rd Qu.: 0.3098 3rd Qu.: 0.467373 3rd Qu.: 0.1087   
## Max. : 3.1466 Max. :10.004222 Max. :20.5112   
##   
## LotConfig Neighborhood BldgType HouseStyle OverallQual   
## Corner : 263 B: 93 1Fam :1220 15:168 5 :397   
## CulDSac: 94 C:329 2fmCon: 31 1 :828 6 :374   
## FR2 : 47 M:182 Duplex: 52 25: 19 7 :319   
## FR3 : 4 N:425 Twnhs : 43 2 :445 8 :168   
## Inside :1052 S:382 TwnhsE: 114 4 :116   
## T: 49 9 : 43   
## (Other): 43   
## YearRemodAdd Exterior1st MasVnrArea BsmtFinSF1   
## 2000 :310 VinylSd:515 Min. :-0.572637 Min. :-0.9727   
## 1950 :272 HdBoard:222 1st Qu.:-0.572637 1st Qu.:-0.9727   
## 1990 :245 MetalSd:220 Median :-0.572637 Median :-0.1319   
## 2006 :236 Wd Sdng:206 Mean : 0.002343 Mean : 0.0000   
## 1970 :173 Plywood:108 3rd Qu.: 0.345535 3rd Qu.: 0.5889   
## 1960 :135 CemntBd: 61 Max. : 8.263909 Max. :11.4018   
## (Other): 89 (Other):128   
## TotRmsAbvGrd GarageCars PavedDrive MoSold YrSold   
## 4 :115 0: 81 N: 90 W :169 2006:314   
## 5 :275 1:369 P: 30 Sp:451 2007:329   
## 6 :402 2:824 Y:1340 S :609 2008:304   
## 7 :329 3:181 F :231 2009:338   
## 8 :187 4: 5 2010:175   
## 9 : 75   
## 10: 77

levels(Recmedhouse$TotRmsAbvGrd) = c("4", "4", "4", "5", "6", "7", "8", "9", "10", "10", "10", "10")  
levels(Recmedhouse$MoSold) = c("W", "W", "Sp", "Sp", "Sp", "S", "S", "S", "F", "F", "F", "W")  
levels(Recmedhouse$YearRemodAdd) = c("1950", "1950", "1950", "1950", "1950", "1950", "1950", "1950", "1950", "1950", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1960", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1970", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1980", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "1990", "2000", "2000", "2000", "2000", "2000", "2000", "2006", "2006", "2006", "2006", "2010")  
levels(Recmedhouse$HouseStyle) = c("15", "15", "1", "25", "25", "2", "1", "1" )  
levels(Recmedhouse$Neighborhood) = c("B", "B", "B", "B", "C", "C", "C", "C", "M", "M", "M", "M", "N", "N", "N", "N", "N", "S", "S", "S", "S", "S", "S", "T", "T")  
summary(Recmedhouse)

## target MSSubClass LotFrontage LotArea LotConfig   
## 1:509 Min. : 20.0 Min. : 21.00 Min. : 1300 Corner : 263   
## 2:951 1st Qu.: 20.0 1st Qu.: 60.00 1st Qu.: 7554 CulDSac: 94   
## Median : 50.0 Median : 69.00 Median : 9478 FR2 : 47   
## Mean : 56.9 Mean : 69.86 Mean : 10517 FR3 : 4   
## 3rd Qu.: 70.0 3rd Qu.: 79.00 3rd Qu.: 11602 Inside :1052   
## Max. :190.0 Max. :313.00 Max. :215245   
##   
## Neighborhood BldgType HouseStyle OverallQual YearRemodAdd  
## B: 93 1Fam :1220 15:168 5 :397 2000 :310   
## C:329 2fmCon: 31 1 :828 6 :374 1950 :272   
## M:182 Duplex: 52 25: 19 7 :319 1990 :245   
## N:425 Twnhs : 43 2 :445 8 :168 2006 :236   
## S:382 TwnhsE: 114 4 :116 1970 :173   
## T: 49 9 : 43 1960 :135   
## (Other): 43 (Other): 89   
## Exterior1st MasVnrArea BsmtFinSF1 TotRmsAbvGrd GarageCars  
## VinylSd:515 Min. : 0.0 Min. : 0.0 4 :115 0: 81   
## HdBoard:222 1st Qu.: 0.0 1st Qu.: 0.0 5 :275 1:369   
## MetalSd:220 Median : 0.0 Median : 383.5 6 :402 2:824   
## Wd Sdng:206 Mean : 103.1 Mean : 443.6 7 :329 3:181   
## Plywood:108 3rd Qu.: 164.2 3rd Qu.: 712.2 8 :187 4: 5   
## CemntBd: 61 Max. :1600.0 Max. :5644.0 9 : 75   
## (Other):128 10: 77   
## PavedDrive MoSold YrSold   
## N: 90 W :169 2006:314   
## P: 30 Sp:451 2007:329   
## Y:1340 S :609 2008:304   
## F :231 2009:338   
## 2010:175   
##   
##

model2b <- train(target~.,data = Recknnhouse, method = "ranger",   
 trControl = trainControl(method ="cv", number = 7, verboseIter = TRUE))

## + Fold1: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold1: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold1: mtry=36, min.node.size=1, splitrule=gini   
## - Fold1: mtry=36, min.node.size=1, splitrule=gini   
## + Fold1: mtry=70, min.node.size=1, splitrule=gini   
## - Fold1: mtry=70, min.node.size=1, splitrule=gini   
## + Fold1: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold1: mtry=36, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry=36, min.node.size=1, splitrule=extratrees   
## + Fold1: mtry=70, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry=70, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold2: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold2: mtry=36, min.node.size=1, splitrule=gini   
## - Fold2: mtry=36, min.node.size=1, splitrule=gini   
## + Fold2: mtry=70, min.node.size=1, splitrule=gini   
## - Fold2: mtry=70, min.node.size=1, splitrule=gini   
## + Fold2: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry=36, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry=36, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry=70, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry=70, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold3: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold3: mtry=36, min.node.size=1, splitrule=gini   
## - Fold3: mtry=36, min.node.size=1, splitrule=gini   
## + Fold3: mtry=70, min.node.size=1, splitrule=gini   
## - Fold3: mtry=70, min.node.size=1, splitrule=gini   
## + Fold3: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry=36, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry=36, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry=70, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry=70, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold4: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold4: mtry=36, min.node.size=1, splitrule=gini   
## - Fold4: mtry=36, min.node.size=1, splitrule=gini   
## + Fold4: mtry=70, min.node.size=1, splitrule=gini   
## - Fold4: mtry=70, min.node.size=1, splitrule=gini   
## + Fold4: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry=36, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry=36, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry=70, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry=70, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold5: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold5: mtry=36, min.node.size=1, splitrule=gini   
## - Fold5: mtry=36, min.node.size=1, splitrule=gini   
## + Fold5: mtry=70, min.node.size=1, splitrule=gini   
## - Fold5: mtry=70, min.node.size=1, splitrule=gini   
## + Fold5: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry=36, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry=36, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry=70, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry=70, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold6: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold6: mtry=36, min.node.size=1, splitrule=gini   
## - Fold6: mtry=36, min.node.size=1, splitrule=gini   
## + Fold6: mtry=70, min.node.size=1, splitrule=gini   
## - Fold6: mtry=70, min.node.size=1, splitrule=gini   
## + Fold6: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry=36, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry=36, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry=70, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry=70, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold7: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold7: mtry=36, min.node.size=1, splitrule=gini   
## - Fold7: mtry=36, min.node.size=1, splitrule=gini   
## + Fold7: mtry=70, min.node.size=1, splitrule=gini   
## - Fold7: mtry=70, min.node.size=1, splitrule=gini   
## + Fold7: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry=36, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry=36, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry=70, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry=70, min.node.size=1, splitrule=extratrees   
## Aggregating results  
## Selecting tuning parameters  
## Fitting mtry = 36, splitrule = gini, min.node.size = 1 on full training set

print(model2b)

## Random Forest   
##   
## 1460 samples  
## 17 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Cross-Validated (7 fold)   
## Summary of sample sizes: 1251, 1251, 1251, 1252, 1252, 1251, ...   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa   
## 2 gini 0.8712800 0.7085402  
## 2 extratrees 0.8466008 0.6485518  
## 36 gini 0.8788021 0.7348574  
## 36 extratrees 0.8637218 0.7028417  
## 70 gini 0.8733208 0.7226440  
## 70 extratrees 0.8623482 0.6992990  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 36, splitrule = gini  
## and min.node.size = 1.

Recoding seems that decrease the accuracy of the model as the accuracy decreased from 0.88 to 0.867.

Encoding. Square rooted all the numeric varaibles, since most of these variable had large outliers. This will put these variable in a smaller range.

sqrtmedhouse = medhouse

sqrtmedhouse$MSSubClass = (sqrtmedhouse$MSSubClass)^.5  
sqrtmedhouse$LotFrontage = (sqrtmedhouse$LotFrontage)^.5  
sqrtmedhouse$LotArea = (sqrtmedhouse$LotArea)^.5  
sqrtmedhouse$MasVnrArea = (sqrtmedhouse$MasVnrArea)^.5  
sqrtmedhouse$BsmtFinSF1 = (sqrtmedhouse$BsmtFinSF1)^.5  
summary(sqrtmedhouse)

## target MSSubClass LotFrontage LotArea LotConfig   
## 1:509 Min. : 4.472 Min. : 4.583 Min. : 36.06 Corner : 263   
## 2:951 1st Qu.: 4.472 1st Qu.: 7.746 1st Qu.: 86.91 CulDSac: 94   
## Median : 7.071 Median : 8.307 Median : 97.36 FR2 : 47   
## Mean : 7.089 Mean : 8.262 Mean : 98.44 FR3 : 4   
## 3rd Qu.: 8.367 3rd Qu.: 8.888 3rd Qu.:107.71 Inside :1052   
## Max. :13.784 Max. :17.692 Max. :463.94   
##   
## Neighborhood BldgType HouseStyle OverallQual YearRemodAdd  
## NAmes :225 1Fam :1220 1Story :726 5 :397 1950 :178   
## CollgCr:150 2fmCon: 31 2Story :445 6 :374 2006 : 97   
## OldTown:113 Duplex: 52 1.5Fin :154 7 :319 2007 : 76   
## Edwards:100 Twnhs : 43 SLvl : 65 8 :168 2005 : 73   
## Somerst: 86 TwnhsE: 114 SFoyer : 37 4 :116 2004 : 62   
## Gilbert: 79 1.5Unf : 14 9 : 43 2000 : 55   
## (Other):707 (Other): 19 (Other): 43 (Other):919   
## Exterior1st MasVnrArea BsmtFinSF1 TotRmsAbvGrd GarageCars  
## VinylSd:515 Min. : 0.000 Min. : 0.00 6 :402 0: 81   
## HdBoard:222 1st Qu.: 0.000 1st Qu.: 0.00 7 :329 1:369   
## MetalSd:220 Median : 0.000 Median :19.58 5 :275 2:824   
## Wd Sdng:206 Mean : 6.014 Mean :16.45 8 :187 3:181   
## Plywood:108 3rd Qu.:12.816 3rd Qu.:26.69 4 : 97 4: 5   
## CemntBd: 61 Max. :40.000 Max. :75.13 9 : 75   
## (Other):128 (Other): 95   
## PavedDrive MoSold YrSold   
## N: 90 6 :253 2006:314   
## P: 30 7 :234 2007:329   
## Y:1340 5 :204 2008:304   
## 4 :141 2009:338   
## 8 :122 2010:175   
## 3 :106   
## (Other):400

model2c <- train(target~.,data = sqrtmedhouse, method = "ranger",   
 trControl = trainControl(method ="cv", number = 7, verboseIter = TRUE))

## + Fold1: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold1: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold1: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold1: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold1: mtry=159, min.node.size=1, splitrule=gini   
## - Fold1: mtry=159, min.node.size=1, splitrule=gini   
## + Fold1: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold1: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold1: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold1: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold2: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold2: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold2: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold2: mtry=159, min.node.size=1, splitrule=gini   
## - Fold2: mtry=159, min.node.size=1, splitrule=gini   
## + Fold2: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold2: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold2: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold3: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold3: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold3: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold3: mtry=159, min.node.size=1, splitrule=gini   
## - Fold3: mtry=159, min.node.size=1, splitrule=gini   
## + Fold3: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold3: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold3: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold4: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold4: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold4: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold4: mtry=159, min.node.size=1, splitrule=gini   
## - Fold4: mtry=159, min.node.size=1, splitrule=gini   
## + Fold4: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold4: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold4: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold5: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold5: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold5: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold5: mtry=159, min.node.size=1, splitrule=gini   
## - Fold5: mtry=159, min.node.size=1, splitrule=gini   
## + Fold5: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold5: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold5: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold6: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold6: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold6: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold6: mtry=159, min.node.size=1, splitrule=gini   
## - Fold6: mtry=159, min.node.size=1, splitrule=gini   
## + Fold6: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold6: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold6: mtry=159, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry= 2, min.node.size=1, splitrule=gini   
## - Fold7: mtry= 2, min.node.size=1, splitrule=gini   
## + Fold7: mtry= 80, min.node.size=1, splitrule=gini   
## - Fold7: mtry= 80, min.node.size=1, splitrule=gini   
## + Fold7: mtry=159, min.node.size=1, splitrule=gini   
## - Fold7: mtry=159, min.node.size=1, splitrule=gini   
## + Fold7: mtry= 2, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry= 2, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry= 80, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry= 80, min.node.size=1, splitrule=extratrees   
## + Fold7: mtry=159, min.node.size=1, splitrule=extratrees   
## - Fold7: mtry=159, min.node.size=1, splitrule=extratrees   
## Aggregating results  
## Selecting tuning parameters  
## Fitting mtry = 80, splitrule = gini, min.node.size = 1 on full training set

print(model2c)

## Random Forest   
##   
## 1460 samples  
## 17 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Cross-Validated (7 fold)   
## Summary of sample sizes: 1252, 1252, 1251, 1251, 1251, 1252, ...   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa   
## 2 gini 0.8636955 0.6874753  
## 2 extratrees 0.8527295 0.6614228  
## 80 gini 0.8780857 0.7331985  
## 80 extratrees 0.8602713 0.6957371  
## 159 gini 0.8760516 0.7294678  
## 159 extratrees 0.8568602 0.6886686  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 80, splitrule = gini  
## and min.node.size = 1.

The results are marginaly better compared to the medhouse data set results before. Accuracy = 0.8787. mrty = 80, splitrule = gini, min.node.size = 1. Still slightly lower than best reading of 0.88

myGrid = expand.grid(mtry = c(75:95), splitrule = c("gini"),  
 min.node.size = c(1:20))  
model3b <- train(target~.,data = knnhouse, method = "ranger",   
 trControl = trainControl(method ="cv", number = 3, verboseIter = TRUE),  
 tuneGrid = myGrid)

## + Fold1: mtry=75, splitrule=gini, min.node.size= 1   
## - Fold1: mtry=75, splitrule=gini, min.node.size= 1   
## + Fold1: mtry=76, splitrule=gini, min.node.size= 1   
## - Fold1: mtry=76, splitrule=gini, min.node.size= 1   
## + Fold1: mtry=77, splitrule=gini, min.node.size= 1   
## - Fold1: mtry=77, splitrule=gini, min.node.size= 1   
  
## Aggregating results  
## Selecting tuning parameters  
## Fitting mtry = 82, splitrule = gini, min.node.size = 1 on full training set

print(model3b)

## Random Forest   
##   
## 1460 samples  
## 17 predictor  
## 2 classes: '1', '2'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold)   
## Summary of sample sizes: 973, 974, 973   
## Resampling results across tuning parameters:  
##   
## mtry min.node.size Accuracy Kappa   
## 75 1 0.8801416 0.7398900  
## 75 2 0.8835681 0.7467124  
## 75 3 0.8767207 0.7318632  
## 75 4 0.8760348 0.7303065  
## 75 5 0.8821978 0.7437452  
## 75 6 0.8801416 0.7386836  
## 75 7 0.8815105 0.7420980  
## 75 8 0.8815161 0.7419948  
## 75 9 0.8780896 0.7341260  
## 75 10 0.8767193 0.7311290  
## 75 11 0.8773995 0.7324240  
## 75 12 0.8774051 0.7325347  
## 75 13 0.8801373 0.7388546  
## 75 14 0.8801416 0.7384355  
## 75 15 0.8787755 0.7367186  
## 75 16 0.8787712 0.7354499  
## 75 17 0.8774009 0.7322098  
## 75 18 0.8794543 0.7362867  
## 75 19 0.8794557 0.7365697  
## 75 20 0.8780840 0.7335225  
## 76 1 0.8808288 0.7405483  
## 76 2 0.8821978 0.7438067  
## 76 3 0.8794627 0.7378308  
## 76 4 0.8794641 0.7381693  
## 76 5 0.8753489 0.7284115  
## 76 6 0.8780896 0.7343558  
## 76 7 0.8753517 0.7279811  
## 76 8 0.8787755 0.7359798  
## 76 9 0.8828822 0.7449107  
## 76 10 0.8774009 0.7321926  
## 76 11 0.8835667 0.7465100  
## 76 12 0.8774023 0.7324441  
## 76 13 0.8780868 0.7345619  
## 76 14 0.8753503 0.7278791  
## 76 15 0.8760306 0.7289485  
## 76 16 0.8767178 0.7306121  
## 76 17 0.8780840 0.7335488  
## 76 18 0.8767136 0.7302849  
## 76 19 0.8774009 0.7321857  
## 76 20 0.8753433 0.7270452  
## 77 1 0.8808303 0.7408271  
## 77 2 0.8780910 0.7350741  
## 77 3 0.8794585 0.7377988  
## 77 4 0.8808317 0.7410432  
## 77 5 0.8746645 0.7270135  
## 77 6 0.8787741 0.7362344  
## 77 7 0.8760348 0.7302602  
## 77 8 0.8780896 0.7343808  
## 77 9 0.8801416 0.7387024  
## 77 10 0.8815119 0.7412024  
## 77 11 0.8767178 0.7311069  
## 77 12 0.8794599 0.7370761  
## 77 13 0.8774009 0.7329542  
## 77 14 0.8780882 0.7338362  
## 77 15 0.8760320 0.7289610  
## 77 16 0.8780868 0.7338409  
## 77 17 0.8767164 0.7308178  
## 77 18 0.8773995 0.7321531  
## 77 19 0.8746602 0.7261835  
## 77 20 0.8746602 0.7261879  
## 78 1 0.8801458 0.7393960  
## 78 2 0.8828879 0.7453663  
## 78 3 0.8787755 0.7364450  
## 78 4 0.8760348 0.7305085  
## 78 5 0.8794599 0.7370661  
## 78 6 0.8780896 0.7343558  
## 78 7 0.8760334 0.7297860  
## 78 8 0.8808288 0.7400663  
## 78 9 0.8815105 0.7428340  
## 78 10 0.8780882 0.7344079  
## 78 11 0.8774051 0.7330511  
## 78 12 0.8801416 0.7389378  
## 78 13 0.8801430 0.7389369  
## 78 14 0.8822006 0.7438270  
## 78 15 0.8808274 0.7403246  
## 78 16 0.8801388 0.7386253  
## 78 17 0.8767193 0.7308658  
## 78 18 0.8787726 0.7351963  
## 78 19 0.8774023 0.7322536  
## 78 20 0.8780825 0.7335383  
## 79 1 0.8787741 0.7362279  
## 79 2 0.8808288 0.7410418  
## 79 3 0.8822034 0.7439795  
## 79 4 0.8822006 0.7442741  
## 79 5 0.8780910 0.7348937  
## 79 6 0.8774051 0.7331940  
## 79 7 0.8774065 0.7329748  
## 79 8 0.8780910 0.7343922  
## 79 9 0.8767207 0.7308732  
## 79 10 0.8815119 0.7416836  
## 79 11 0.8808274 0.7407468  
## 79 12 0.8767178 0.7310863  
## 79 13 0.8767178 0.7308397  
## 79 14 0.8753447 0.7280367  
## 79 15 0.8780882 0.7333666  
## 79 16 0.8787755 0.7357437  
## 79 17 0.8767164 0.7308165  
## 79 18 0.8794585 0.7373076  
## 79 19 0.8773995 0.7324060  
## 79 20 0.8767150 0.7305548  
## 80 1 0.8794613 0.7373964  
## 80 2 0.8794585 0.7378198  
## 80 3 0.8787741 0.7364516  
## 80 4 0.8780896 0.7350420  
## 80 5 0.8787726 0.7361452  
## 80 6 0.8815147 0.7418915  
## 80 7 0.8767207 0.7314354  
## 80 8 0.8787741 0.7359887  
## 80 9 0.8774009 0.7331840  
## 80 10 0.8835667 0.7460701  
## 80 11 0.8794571 0.7371000  
## 80 12 0.8801416 0.7386596  
## 80 13 0.8808260 0.7400625  
## 80 14 0.8787712 0.7356632  
## 80 15 0.8801388 0.7381614  
## 80 16 0.8794557 0.7372718  
## 80 17 0.8794571 0.7368040  
## 80 18 0.8767150 0.7303189  
## 80 19 0.8780840 0.7335188  
## 80 20 0.8780811 0.7327925  
## 81 1 0.8787741 0.7361866  
## 81 2 0.8815119 0.7425893  
## 81 3 0.8787783 0.7357701  
## 81 4 0.8801458 0.7394569  
## 81 5 0.8801472 0.7394910  
## 81 6 0.8815175 0.7422525  
## 81 7 0.8794599 0.7376189  
## 81 8 0.8774051 0.7327082  
## 81 9 0.8746645 0.7273137  
## 81 10 0.8767164 0.7310730  
## 81 11 0.8774009 0.7319395  
## 81 12 0.8821964 0.7428654  
## 81 13 0.8787755 0.7359798  
## 81 14 0.8780840 0.7335395  
## 81 15 0.8753461 0.7275924  
## 81 16 0.8760306 0.7291802  
## 81 17 0.8794543 0.7367902  
## 81 18 0.8746616 0.7264584  
## 81 19 0.8780868 0.7335733  
## 81 20 0.8767164 0.7306176  
## 82 1 0.8849398 0.7504241  
## 82 2 0.8760348 0.7301975  
## 82 3 0.8787755 0.7364997  
## 82 4 0.8794613 0.7380645  
## 82 5 0.8787741 0.7359607  
## 82 6 0.8801444 0.7397057  
## 82 7 0.8760320 0.7292436  
## 82 8 0.8774037 0.7332039  
## 82 9 0.8794613 0.7369026  
## 82 10 0.8760334 0.7299608  
## 82 11 0.8787698 0.7361282  
## 82 12 0.8794585 0.7370835  
## 82 13 0.8828836 0.7449260  
## 82 14 0.8773995 0.7324193  
## 82 15 0.8780854 0.7345379  
## 82 16 0.8774051 0.7327311  
## 82 17 0.8801430 0.7389158  
## 82 18 0.8794543 0.7367645  
## 82 19 0.8821935 0.7422795  
## 82 20 0.8773995 0.7319306  
## 83 1 0.8774065 0.7335040  
## 83 2 0.8774094 0.7332384  
## 83 3 0.8780882 0.7348276  
## 83 4 0.8767178 0.7313424  
## 83 5 0.8774108 0.7339676  
## 83 6 0.8767193 0.7311122  
## 83 7 0.8760348 0.7302617  
## 83 8 0.8774051 0.7327600  
## 83 9 0.8808274 0.7398363  
## 83 10 0.8767193 0.7310788  
## 83 11 0.8767178 0.7318594  
## 83 12 0.8753461 0.7278401  
## 83 13 0.8774037 0.7329684  
## 83 14 0.8815119 0.7419085  
## 83 15 0.8787741 0.7354840  
## 83 16 0.8767136 0.7305409  
## 83 17 0.8787684 0.7353760  
## 83 18 0.8767150 0.7303028  
## 83 19 0.8746602 0.7266821  
## 83 20 0.8794557 0.7375353  
## 84 1 0.8794599 0.7385820  
## 84 2 0.8787755 0.7365247  
## 84 3 0.8787755 0.7374280  
## 84 4 0.8774079 0.7336594  
## 84 5 0.8801430 0.7394055  
## 84 6 0.8794585 0.7385635  
## 84 7 0.8774009 0.7326974  
## 84 8 0.8767164 0.7313289  
## 84 9 0.8808274 0.7403301  
## 84 10 0.8746659 0.7270248  
## 84 11 0.8787769 0.7359866  
## 84 12 0.8780910 0.7341467  
## 84 13 0.8808246 0.7397770  
## 84 14 0.8801430 0.7393768  
## 84 15 0.8787698 0.7351752  
## 84 16 0.8780868 0.7335841  
## 84 17 0.8739786 0.7251491  
## 84 18 0.8780840 0.7330433  
## 84 19 0.8760306 0.7294232  
## 84 20 0.8767164 0.7303231  
## 85 1 0.8780896 0.7353265  
## 85 2 0.8746673 0.7278105  
## 85 3 0.8821992 0.7437589  
## 85 4 0.8794599 0.7385529  
## 85 5 0.8767164 0.7320363  
## 85 6 0.8787741 0.7364496  
## 85 7 0.8780882 0.7348123  
## 85 8 0.8753517 0.7291885  
## 85 9 0.8794613 0.7376118  
## 85 10 0.8780868 0.7342920  
## 85 11 0.8787698 0.7359090  
## 85 12 0.8780868 0.7345406  
## 85 13 0.8794585 0.7370714  
## 85 14 0.8746645 0.7262553  
## 85 15 0.8774023 0.7324634  
## 85 16 0.8794543 0.7372666  
## 85 17 0.8746602 0.7256707  
## 85 18 0.8787698 0.7349707  
## 85 19 0.8767136 0.7302903  
## 85 20 0.8760292 0.7291754  
## 86 1 0.8774079 0.7334743  
## 86 2 0.8815161 0.7423960  
## 86 3 0.8774079 0.7332345  
## 86 4 0.8808288 0.7408186  
## 86 5 0.8767207 0.7318724  
## 86 6 0.8787755 0.7357143  
## 86 7 0.8774023 0.7331872  
## 86 8 0.8787726 0.7357442  
## 86 9 0.8767178 0.7315997  
## 86 10 0.8794571 0.7373304  
## 86 11 0.8835639 0.7466976  
## 86 12 0.8801388 0.7384131  
## 86 13 0.8774009 0.7326560  
## 86 14 0.8787726 0.7349583  
## 86 15 0.8746645 0.7262858  
## 86 16 0.8794529 0.7369974  
## 86 17 0.8760306 0.7291792  
## 86 18 0.8739786 0.7246047  
## 86 19 0.8760292 0.7284175  
## 86 20 0.8794529 0.7369791  
## 87 1 0.8808317 0.7415851  
## 87 2 0.8774037 0.7342291  
## 87 3 0.8794613 0.7383693  
## 87 4 0.8801458 0.7398796  
## 87 5 0.8774079 0.7335452  
## 87 6 0.8760334 0.7304671  
## 87 7 0.8801430 0.7396302  
## 87 8 0.8767178 0.7309069  
## 87 9 0.8815105 0.7423466  
## 87 10 0.8774051 0.7325598  
## 87 11 0.8815133 0.7421408  
## 87 12 0.8767164 0.7315860  
## 87 13 0.8787726 0.7354613  
## 87 14 0.8815119 0.7416598  
## 87 15 0.8794543 0.7370304  
## 87 16 0.8767164 0.7305784  
## 87 17 0.8780896 0.7348502  
## 87 18 0.8780825 0.7339637  
## 87 19 0.8787726 0.7354492  
## 87 20 0.8773995 0.7326519  
## 88 1 0.8794613 0.7380445  
## 88 2 0.8794627 0.7380404  
## 88 3 0.8794599 0.7378456  
## 88 4 0.8774079 0.7337935  
## 88 5 0.8767207 0.7316395  
## 88 6 0.8808317 0.7401076  
## 88 7 0.8801430 0.7393741  
## 88 8 0.8780882 0.7343539  
## 88 9 0.8753489 0.7278794  
## 88 10 0.8828822 0.7451174  
## 88 11 0.8760334 0.7297421  
## 88 12 0.8780868 0.7343074  
## 88 13 0.8787726 0.7361837  
## 88 14 0.8780868 0.7343525  
## 88 15 0.8801444 0.7384452  
## 88 16 0.8815119 0.7414341  
## 88 17 0.8719238 0.7200193  
## 88 18 0.8753475 0.7278311  
## 88 19 0.8774037 0.7327216  
## 88 20 0.8739786 0.7245976  
## 89 1 0.8774094 0.7339781  
## 89 2 0.8794627 0.7375710  
## 89 3 0.8760348 0.7297421  
## 89 4 0.8767178 0.7318676  
## 89 5 0.8815147 0.7414558  
## 89 6 0.8732955 0.7235168  
## 89 7 0.8746659 0.7267566  
## 89 8 0.8774051 0.7330422  
## 89 9 0.8774065 0.7330176  
## 89 10 0.8794585 0.7373236  
## 89 11 0.8780910 0.7350741  
## 89 12 0.8801430 0.7393988  
## 89 13 0.8787712 0.7364187  
## 89 14 0.8815105 0.7416652  
## 89 15 0.8774009 0.7326560  
## 89 16 0.8794571 0.7368268  
## 89 17 0.8773995 0.7326432  
## 89 18 0.8774009 0.7324366  
## 89 19 0.8787712 0.7351974  
## 89 20 0.8773995 0.7319384  
## 90 1 0.8780924 0.7351055  
## 90 2 0.8808317 0.7410182  
## 90 3 0.8767235 0.7316750  
## 90 4 0.8794613 0.7378856  
## 90 5 0.8780896 0.7348453  
## 90 6 0.8808303 0.7410483  
## 90 7 0.8767193 0.7320836  
## 90 8 0.8794571 0.7370525  
## 90 9 0.8767207 0.7319022  
## 90 10 0.8794613 0.7383392  
## 90 11 0.8739800 0.7249536  
## 90 12 0.8774065 0.7327535  
## 90 13 0.8801402 0.7384078  
## 90 14 0.8780854 0.7340479  
## 90 15 0.8801430 0.7384719  
## 90 16 0.8780882 0.7335843  
## 90 17 0.8787698 0.7354018  
## 90 18 0.8767150 0.7308280  
## 90 19 0.8753503 0.7279008  
## 90 20 0.8732913 0.7224694  
## 91 1 0.8835695 0.7479878  
## 91 2 0.8808317 0.7413235  
## 91 3 0.8808345 0.7415894  
## 91 4 0.8815161 0.7431374  
## 91 5 0.8794613 0.7380907  
## 91 6 0.8801416 0.7396433  
## 91 7 0.8746659 0.7268100  
## 91 8 0.8767207 0.7321060  
## 91 9 0.8767178 0.7323088  
## 91 10 0.8774079 0.7330932  
## 91 11 0.8780868 0.7340677  
## 91 12 0.8774037 0.7331773  
## 91 13 0.8767193 0.7306252  
## 91 14 0.8787726 0.7356937  
## 91 15 0.8774037 0.7322188  
## 91 16 0.8787684 0.7348983  
## 91 17 0.8767178 0.7311069  
## 91 18 0.8746616 0.7259548  
## 91 19 0.8767164 0.7310608  
## 91 20 0.8746602 0.7259570  
## 92 1 0.8801500 0.7397740  
## 92 2 0.8835695 0.7474654  
## 92 3 0.8808303 0.7407889  
## 92 4 0.8801430 0.7393741  
## 92 5 0.8774065 0.7329725  
## 92 6 0.8767178 0.7317909  
## 92 7 0.8739828 0.7264705  
## 92 8 0.8787741 0.7359941  
## 92 9 0.8787741 0.7359508  
## 92 10 0.8801430 0.7391763  
## 92 11 0.8767178 0.7310730  
## 92 12 0.8794571 0.7372751  
## 92 13 0.8794543 0.7372685  
## 92 14 0.8780868 0.7345357  
## 92 15 0.8773981 0.7323953  
## 92 16 0.8767164 0.7313162  
## 92 17 0.8739772 0.7243361  
## 92 18 0.8787698 0.7349689  
## 92 19 0.8732927 0.7234799  
## 92 20 0.8774009 0.7319704  
## 93 1 0.8801472 0.7393902  
## 93 2 0.8815175 0.7427014  
## 93 3 0.8780910 0.7350654  
## 93 4 0.8787769 0.7364581  
## 93 5 0.8787755 0.7356777  
## 93 6 0.8808288 0.7410384  
## 93 7 0.8794585 0.7380496  
## 93 8 0.8760334 0.7302586  
## 93 9 0.8794585 0.7373436  
## 93 10 0.8780910 0.7346027  
## 93 11 0.8753503 0.7286308  
## 93 12 0.8739800 0.7249014  
## 93 13 0.8746630 0.7267470  
## 93 14 0.8753461 0.7280710  
## 93 15 0.8760348 0.7300351  
## 93 16 0.8753461 0.7278078  
## 93 17 0.8794599 0.7366188  
## 93 18 0.8746588 0.7261610  
## 93 19 0.8739758 0.7243414  
## 93 20 0.8739786 0.7246300  
## 94 1 0.8767221 0.7322737  
## 94 2 0.8794585 0.7380213  
## 94 3 0.8767193 0.7323507  
## 94 4 0.8815147 0.7426270  
## 94 5 0.8760390 0.7305464  
## 94 6 0.8760334 0.7299907  
## 94 7 0.8794599 0.7373917  
## 94 8 0.8822006 0.7438090  
## 94 9 0.8760362 0.7302992  
## 94 10 0.8774009 0.7329332  
## 94 11 0.8801444 0.7387212  
## 94 12 0.8753517 0.7284905  
## 94 13 0.8787712 0.7356706  
## 94 14 0.8746645 0.7270407  
## 94 15 0.8767193 0.7313688  
## 94 16 0.8767178 0.7310893  
## 94 17 0.8794571 0.7368241  
## 94 18 0.8787741 0.7354584  
## 94 19 0.8739758 0.7250875  
## 94 20 0.8753461 0.7275739  
## 95 1 0.8753531 0.7286519  
## 95 2 0.8815161 0.7417430  
## 95 3 0.8787769 0.7364861  
## 95 4 0.8774094 0.7340020  
## 95 5 0.8739828 0.7264493  
## 95 6 0.8774051 0.7339547  
## 95 7 0.8801430 0.7389485  
## 95 8 0.8780854 0.7352938  
## 95 9 0.8794627 0.7376421  
## 95 10 0.8767178 0.7310863  
## 95 11 0.8774051 0.7329748  
## 95 12 0.8794613 0.7375931  
## 95 13 0.8780882 0.7338275  
## 95 14 0.8808274 0.7402844  
## 95 15 0.8746645 0.7267803  
## 95 16 0.8732927 0.7232372  
## 95 17 0.8767164 0.7310721  
## 95 18 0.8732927 0.7227358  
## 95 19 0.8746602 0.7256851  
## 95 20 0.8739786 0.7251254  
##   
## Tuning parameter 'splitrule' was held constant at a value of gini  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 82, splitrule = gini  
## and min.node.size = 1.

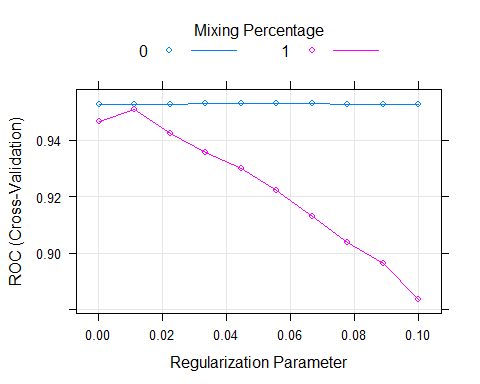
Accuracy = 0.8808 mtry = 80, splitrule = gini, min.node.size = 8

levels(knnhouse$target) = c("low", "high")  
myControl <- trainControl(method = "cv", number = 10, summaryFunction = twoClassSummary,  
classProbs = TRUE, verboseIter = TRUE)  
myGrid2 = expand.grid(alpha = 0:1,lambda = seq(0.0001, 0.1, length = 10))  
  
  
model4b <- train(target~.,data = knnhouse, method = "glmnet",   
 trControl = myControl, tuneGrid = myGrid2)

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was  
## not in the result set. ROC will be used instead.

## + Fold01: alpha=0, lambda=0.1   
## - Fold01: alpha=0, lambda=0.1   
## + Fold01: alpha=1, lambda=0.1   
## - Fold01: alpha=1, lambda=0.1   
## + Fold02: alpha=0, lambda=0.1   
## - Fold02: alpha=0, lambda=0.1   
## + Fold02: alpha=1, lambda=0.1   
## - Fold02: alpha=1, lambda=0.1   
## + Fold03: alpha=0, lambda=0.1   
## - Fold03: alpha=0, lambda=0.1   
## + Fold03: alpha=1, lambda=0.1   
## - Fold03: alpha=1, lambda=0.1   
## + Fold04: alpha=0, lambda=0.1   
## - Fold04: alpha=0, lambda=0.1   
## + Fold04: alpha=1, lambda=0.1   
## - Fold04: alpha=1, lambda=0.1   
## + Fold05: alpha=0, lambda=0.1   
## - Fold05: alpha=0, lambda=0.1   
## + Fold05: alpha=1, lambda=0.1   
## - Fold05: alpha=1, lambda=0.1   
## + Fold06: alpha=0, lambda=0.1   
## - Fold06: alpha=0, lambda=0.1   
## + Fold06: alpha=1, lambda=0.1   
## - Fold06: alpha=1, lambda=0.1   
## + Fold07: alpha=0, lambda=0.1   
## - Fold07: alpha=0, lambda=0.1   
## + Fold07: alpha=1, lambda=0.1   
## - Fold07: alpha=1, lambda=0.1   
## + Fold08: alpha=0, lambda=0.1   
## - Fold08: alpha=0, lambda=0.1   
## + Fold08: alpha=1, lambda=0.1   
## - Fold08: alpha=1, lambda=0.1   
## + Fold09: alpha=0, lambda=0.1   
## - Fold09: alpha=0, lambda=0.1   
## + Fold09: alpha=1, lambda=0.1   
## - Fold09: alpha=1, lambda=0.1   
## + Fold10: alpha=0, lambda=0.1   
## - Fold10: alpha=0, lambda=0.1   
## + Fold10: alpha=1, lambda=0.1   
## - Fold10: alpha=1, lambda=0.1   
## Aggregating results  
## Selecting tuning parameters  
## Fitting alpha = 0, lambda = 0.0556 on full training set

plot(model4b)



print(model4b)

## glmnet   
##   
## 1460 samples  
## 17 predictor  
## 2 classes: 'low', 'high'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 1314, 1314, 1314, 1314, 1313, 1314, ...   
## Resampling results across tuning parameters:  
##   
## alpha lambda ROC Sens Spec   
## 0 0.0001 0.9527132 0.8447451 0.9095724  
## 0 0.0112 0.9527132 0.8447451 0.9095724  
## 0 0.0223 0.9527132 0.8467059 0.9106140  
## 0 0.0334 0.9530858 0.8368627 0.9127083  
## 0 0.0445 0.9531072 0.8349020 0.9137610  
## 0 0.0556 0.9531687 0.8309804 0.9137610  
## 0 0.0667 0.9529830 0.8309804 0.9148136  
## 0 0.0778 0.9527736 0.8309804 0.9137610  
## 0 0.0889 0.9527745 0.8290196 0.9148136  
## 0 0.1000 0.9526302 0.8290196 0.9158662  
## 1 0.0001 0.9467121 0.8309020 0.8948246  
## 1 0.0112 0.9508322 0.8231373 0.9116667  
## 1 0.0223 0.9425137 0.7936863 0.9127083  
## 1 0.0334 0.9355917 0.7465490 0.9200768  
## 1 0.0445 0.9299582 0.7092549 0.9253180  
## 1 0.0556 0.9221860 0.6503922 0.9337281  
## 1 0.0667 0.9131278 0.5777647 0.9379386  
## 1 0.0778 0.9039062 0.5285098 0.9432018  
## 1 0.0889 0.8963196 0.4774902 0.9442544  
## 1 0.1000 0.8834680 0.3988235 0.9484649  
##   
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were alpha = 0 and lambda = 0.0556.

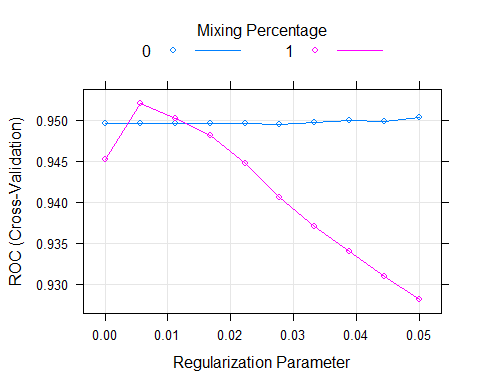
ROC = 0.9508983, Alpha = 0, lambda = 0.0334 Most model had accuracy of about 0.88 or 0.87, this glmnet model produced signifcantly results

levels(knnhouse$target) = c("low", "high")  
myControl <- trainControl(method = "cv", number = 10, summaryFunction = twoClassSummary,  
classProbs = TRUE, verboseIter = TRUE)  
myGrid3 = expand.grid(alpha = 0:1,lambda = seq(0.0001, 0.05, length = 10))  
  
  
model4t <- train(target~.,data = knnhouse, method = "glmnet",   
 trControl = myControl, tuneGrid = myGrid3)

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was  
## not in the result set. ROC will be used instead.

## + Fold01: alpha=0, lambda=0.05   
## - Fold01: alpha=0, lambda=0.05   
## + Fold01: alpha=1, lambda=0.05   
## - Fold01: alpha=1, lambda=0.05   
## + Fold02: alpha=0, lambda=0.05   
## - Fold02: alpha=0, lambda=0.05   
## + Fold02: alpha=1, lambda=0.05   
## - Fold02: alpha=1, lambda=0.05   
## + Fold03: alpha=0, lambda=0.05   
## - Fold03: alpha=0, lambda=0.05   
## + Fold03: alpha=1, lambda=0.05   
## - Fold03: alpha=1, lambda=0.05   
## + Fold04: alpha=0, lambda=0.05   
## - Fold04: alpha=0, lambda=0.05   
## + Fold04: alpha=1, lambda=0.05   
## - Fold04: alpha=1, lambda=0.05   
## + Fold05: alpha=0, lambda=0.05   
## - Fold05: alpha=0, lambda=0.05   
## + Fold05: alpha=1, lambda=0.05   
## - Fold05: alpha=1, lambda=0.05   
## + Fold06: alpha=0, lambda=0.05   
## - Fold06: alpha=0, lambda=0.05   
## + Fold06: alpha=1, lambda=0.05   
## - Fold06: alpha=1, lambda=0.05   
## + Fold07: alpha=0, lambda=0.05   
## - Fold07: alpha=0, lambda=0.05   
## + Fold07: alpha=1, lambda=0.05   
## - Fold07: alpha=1, lambda=0.05   
## + Fold08: alpha=0, lambda=0.05   
## - Fold08: alpha=0, lambda=0.05   
## + Fold08: alpha=1, lambda=0.05   
## - Fold08: alpha=1, lambda=0.05   
## + Fold09: alpha=0, lambda=0.05   
## - Fold09: alpha=0, lambda=0.05   
## + Fold09: alpha=1, lambda=0.05   
## - Fold09: alpha=1, lambda=0.05   
## + Fold10: alpha=0, lambda=0.05   
## - Fold10: alpha=0, lambda=0.05   
## + Fold10: alpha=1, lambda=0.05   
## - Fold10: alpha=1, lambda=0.05   
## Aggregating results  
## Selecting tuning parameters  
## Fitting alpha = 1, lambda = 0.00564 on full training set

plot(model4t)



print(model4t)

## glmnet   
##   
## 1460 samples  
## 17 predictor  
## 2 classes: 'low', 'high'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 1314, 1314, 1314, 1314, 1313, 1314, ...   
## Resampling results across tuning parameters:  
##   
## alpha lambda ROC Sens Spec   
## 0 0.000100000 0.9496341 0.8370196 0.9106360  
## 0 0.005644444 0.9496341 0.8370196 0.9106360  
## 0 0.011188889 0.9496341 0.8370196 0.9106360  
## 0 0.016733333 0.9496341 0.8370196 0.9106360  
## 0 0.022277778 0.9496341 0.8370196 0.9106360  
## 0 0.027822222 0.9495094 0.8370196 0.9106360  
## 0 0.033366667 0.9496752 0.8389804 0.9085307  
## 0 0.038911111 0.9499453 0.8429020 0.9095833  
## 0 0.044455556 0.9498217 0.8389804 0.9095833  
## 0 0.050000000 0.9502560 0.8409412 0.9085307  
## 1 0.000100000 0.9451787 0.8214118 0.9022259  
## 1 0.005644444 0.9520418 0.8292549 0.9106360  
## 1 0.011188889 0.9501685 0.8213725 0.9137829  
## 1 0.016733333 0.9481641 0.8056078 0.9116996  
## 1 0.022277778 0.9446732 0.7938039 0.9179934  
## 1 0.027822222 0.9405536 0.7702745 0.9200987  
## 1 0.033366667 0.9370260 0.7427843 0.9200987  
## 1 0.038911111 0.9339592 0.7192157 0.9221930  
## 1 0.044455556 0.9309438 0.7015686 0.9264035  
## 1 0.050000000 0.9281060 0.6799216 0.9285088  
##   
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were alpha = 1 and lambda  
## = 0.005644444.

ROC = 0.949917 Alpha = 0 lambda = .0056 Slightly worse than the less tuned glmnet model

model5b <- train(target~.,data = knnhouse, method = "rpart",   
 trControl = trainControl(method ="cv", number = 10, verboseIter = TRUE))

## + Fold01: cp=0.05108   
## - Fold01: cp=0.05108   
## + Fold02: cp=0.05108   
## - Fold02: cp=0.05108   
## + Fold03: cp=0.05108   
## - Fold03: cp=0.05108   
## + Fold04: cp=0.05108   
## - Fold04: cp=0.05108   
## + Fold05: cp=0.05108   
## - Fold05: cp=0.05108   
## + Fold06: cp=0.05108   
## - Fold06: cp=0.05108   
## + Fold07: cp=0.05108   
## - Fold07: cp=0.05108   
## + Fold08: cp=0.05108   
## - Fold08: cp=0.05108   
## + Fold09: cp=0.05108   
## - Fold09: cp=0.05108   
## + Fold10: cp=0.05108   
## - Fold10: cp=0.05108   
## Aggregating results  
## Selecting tuning parameters  
## Fitting cp = 0.0511 on full training set

print(model5b)

## CART   
##   
## 1460 samples  
## 17 predictor  
## 2 classes: 'low', 'high'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 1314, 1315, 1314, 1314, 1314, 1314, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.05108055 0.8171165 0.5936914  
## 0.10216110 0.7890201 0.5131629  
## 0.33988212 0.7007289 0.2074987  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.05108055.

Best accuracy of 0.8226 with a cp of 0.05108

splitIndex <- createDataPartition(knnhouse$target, p = .70, list = FALSE, times = 1)  
train1 <- knnhouse[ splitIndex,]  
test1 <- knnhouse[-splitIndex,]  
model6b = ranger(target ~., data = train1)  
pred6b = predict(model6b,data = test1)$predictions  
cm1 = confusionMatrix(pred6b, test1$target, positive = NULL)  
cm1

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction low high  
## low 128 20  
## high 24 265  
##   
## Accuracy : 0.8993   
## 95% CI : (0.8672, 0.9259)  
## No Information Rate : 0.6522   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7767   
## Mcnemar's Test P-Value : 0.6511   
##   
## Sensitivity : 0.8421   
## Specificity : 0.9298   
## Pos Pred Value : 0.8649   
## Neg Pred Value : 0.9170   
## Prevalence : 0.3478   
## Detection Rate : 0.2929   
## Detection Prevalence : 0.3387   
## Balanced Accuracy : 0.8860   
##   
## 'Positive' Class : low   
##

Accuracy = 0.8879, Balanced Accuracy = 0.8803

### Conclusion

Models have strong predicting power for home sales price. The Knn impuation was the most succesful imputation method with a baseline accuracy of 0.88. The best model was the Knn glmnet model with a ROC of 0.95. All other model still had strong predictive power with accuracies of 0.87,0.88. Recoding and Encoding did not add any predictive power. The variables were already reduced to one with strong predictive power.This was evident in the differnces between the target variables as they were quite noticable in the graphics. Overall, This dataset could be used to predict if the sales price of a home was above of below the projected median cost.