Ames, Iowa: Which characteristics predict if a home will sell above or below the median home value?

## Set-Up & Selection

library(tidyverse)

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.2 ✔ readr 2.1.4  
✔ forcats 1.0.0 ✔ stringr 1.5.0  
✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
✔ purrr 1.0.1   
── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks stats::filter()  
✖ dplyr::lag() masks stats::lag()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(tidymodels)

── Attaching packages ────────────────────────────────────── tidymodels 1.1.0 ──  
✔ broom 1.0.4 ✔ rsample 1.1.1  
✔ dials 1.2.0 ✔ tune 1.1.1  
✔ infer 1.0.4 ✔ workflows 1.1.3  
✔ modeldata 1.1.0 ✔ workflowsets 1.0.1  
✔ parsnip 1.1.0 ✔ yardstick 1.2.0  
✔ recipes 1.0.6   
── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
✖ scales::discard() masks purrr::discard()  
✖ dplyr::filter() masks stats::filter()  
✖ recipes::fixed() masks stringr::fixed()  
✖ dplyr::lag() masks stats::lag()  
✖ yardstick::spec() masks readr::spec()  
✖ recipes::step() masks stats::step()  
• Learn how to get started at https://www.tidymodels.org/start/

library(glmnet) #for Lasso, ridge, and elastic net models

Loading required package: Matrix  
  
Attaching package: 'Matrix'  
  
The following objects are masked from 'package:tidyr':  
  
 expand, pack, unpack  
  
Loaded glmnet 4.1-7

library(GGally) #create ggcorr and ggpairs plots

Registered S3 method overwritten by 'GGally':  
 method from   
 +.gg ggplot2

library(ggcorrplot) #create an alternative to ggcorr plots  
library(MASS) #access to forward and backward selection algorithms

Attaching package: 'MASS'  
  
The following object is masked from 'package:dplyr':  
  
 select

library(leaps) #best subset selection  
library(lmtest) #for the dw test

Loading required package: zoo  
  
Attaching package: 'zoo'  
  
The following objects are masked from 'package:base':  
  
 as.Date, as.Date.numeric

library(splines) #for nonlinear fitting  
library(e1071)

Attaching package: 'e1071'  
  
The following object is masked from 'package:tune':  
  
 tune  
  
The following object is masked from 'package:rsample':  
  
 permutations  
  
The following object is masked from 'package:parsnip':  
  
 tune

library(ROCR)  
library(caret)

Loading required package: lattice  
  
Attaching package: 'caret'  
  
The following objects are masked from 'package:yardstick':  
  
 precision, recall, sensitivity, specificity  
  
The following object is masked from 'package:purrr':  
  
 lift

library(rpart)

Attaching package: 'rpart'  
  
The following object is masked from 'package:dials':  
  
 prune

library(rpart.plot)  
library(RColorBrewer)  
library(rattle)

Loading required package: bitops  
  
Attaching package: 'bitops'  
  
The following object is masked from 'package:Matrix':  
  
 %&%  
  
Rattle: A free graphical interface for data science with R.  
Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.  
Type 'rattle()' to shake, rattle, and roll your data.

library(gridExtra)

Attaching package: 'gridExtra'  
  
The following object is masked from 'package:dplyr':  
  
 combine

library(vip)

Attaching package: 'vip'  
  
The following object is masked from 'package:utils':  
  
 vi

library(ranger)

Attaching package: 'ranger'  
  
The following object is masked from 'package:rattle':  
  
 importance

library(usemodels)

Above I have loaded in the required packages. Then I load in the ames dataset.

ames = read\_csv("ames\_student-1.csv")

Rows: 2053 Columns: 81  
── Column specification ────────────────────────────────────────────────────────  
Delimiter: ","  
chr (47): MS\_SubClass, MS\_Zoning, Street, Alley, Lot\_Shape, Land\_Contour, Ut...  
dbl (34): Lot\_Frontage, Lot\_Area, Year\_Built, Year\_Remod\_Add, Mas\_Vnr\_Area, ...  
  
ℹ Use `spec()` to retrieve the full column specification for this data.  
ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

First I mutate ames to convert character variables to factors.

ames=  
 ames%>%  
 mutate\_if(is.character,as\_factor)

Next I check that our response variable, Above\_Median, is ordered correctly so that our negative response is first.

levels(ames$Above\_Median)

[1] "Yes" "No"

ames = ames %>% mutate(Above\_Median = fct\_relevel(Above\_Median, c("No","Yes")))  
levels(ames$Above\_Median)

[1] "No" "Yes"

Then I look at the summary for the ames set. My aim to eliminate variables that have low variance, or may be heavily skewed one way or another.

summary(ames)

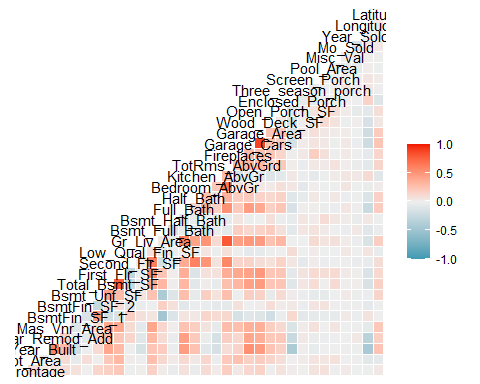
MS\_SubClass MS\_Zoning   
 One\_Story\_1946\_and\_Newer\_All\_Styles :772 Residential\_Low\_Density :1600   
 Two\_Story\_1946\_and\_Newer :383 Residential\_High\_Density : 20   
 One\_and\_Half\_Story\_Finished\_All\_Ages:204 Floating\_Village\_Residential: 87   
 One\_Story\_PUD\_1946\_and\_Newer :129 Residential\_Medium\_Density : 326   
 One\_Story\_1945\_and\_Older : 98 C\_all : 17   
 Two\_Story\_1945\_and\_Older : 95 A\_agr : 2   
 (Other) :372 I\_all : 1   
 Lot\_Frontage Lot\_Area Street Alley   
 Min. : 0.00 Min. : 1300 Pave:2046 No\_Alley\_Access:1914   
 1st Qu.: 43.00 1st Qu.: 7500 Grvl: 7 Paved : 45   
 Median : 62.00 Median : 9548 Gravel : 94   
 Mean : 57.38 Mean : 10258   
 3rd Qu.: 78.00 3rd Qu.: 11600   
 Max. :313.00 Max. :215245   
   
 Lot\_Shape Land\_Contour Utilities Lot\_Config   
 Slightly\_Irregular : 714 Lvl:1833 AllPub:2052 Corner : 359   
 Regular :1275 HLS: 94 NoSewr: 1 Inside :1495   
 Moderately\_Irregular: 53 Bnk: 81 CulDSac: 135   
 Irregular : 11 Low: 45 FR2 : 56   
 FR3 : 8   
   
   
 Land\_Slope Neighborhood Condition\_1 Condition\_2 Bldg\_Type   
 Gtl:1951 North\_Ames : 327 Norm :1771 Norm :2027 OneFam :1706   
 Mod: 89 College\_Creek: 183 Feedr : 113 Feedr : 12 TwnhsE : 157   
 Sev: 13 Old\_Town : 181 Artery : 67 PosA : 4 Twnhs : 67   
 Edwards : 129 RRAn : 35 Artery : 4 Duplex : 76   
 Somerset : 119 PosN : 24 PosN : 3 TwoFmCon: 47   
 Gilbert : 109 RRAe : 19 RRNn : 1   
 (Other) :1005 (Other): 24 (Other): 2   
 House\_Style Overall\_Qual Overall\_Cond   
 One\_Story :1052 Average :587 Average :1143   
 Two\_Story : 590 Above\_Average:518 Above\_Average: 376   
 One\_and\_Half\_Fin: 225 Good :411 Good : 286   
 SLvl : 90 Very\_Good :237 Very\_Good : 98   
 SFoyer : 56 Below\_Average:169 Below\_Average: 73   
 Two\_and\_Half\_Unf: 19 Excellent : 70 Fair : 35   
 (Other) : 21 (Other) : 61 (Other) : 42   
 Year\_Built Year\_Remod\_Add Roof\_Style Roof\_Matl Exterior\_1st  
 Min. :1875 Min. :1950 Hip : 404 CompShg:2023 VinylSd:705   
 1st Qu.:1953 1st Qu.:1965 Gable :1607 WdShake: 8 MetalSd:319   
 Median :1972 Median :1993 Mansard: 9 Tar&Grv: 17 Wd Sdng:313   
 Mean :1971 Mean :1984 Gambrel: 14 WdShngl: 3 HdBoard:303   
 3rd Qu.:2000 3rd Qu.:2004 Shed : 5 Roll : 1 Plywood:151   
 Max. :2010 Max. :2010 Flat : 14 Metal : 1 CemntBd: 90   
 (Other):172   
 Exterior\_2nd Mas\_Vnr\_Type Mas\_Vnr\_Area Exter\_Qual   
 VinylSd:699 Stone : 166 Min. : 0.0 Typical :1272   
 MetalSd:317 None :1231 1st Qu.: 0.0 Good : 682   
 Wd Sdng:302 BrkFace: 638 Median : 0.0 Excellent: 78   
 HdBoard:277 BrkCmn : 17 Mean : 103.8 Fair : 21   
 Plywood:190 CBlock : 1 3rd Qu.: 164.0   
 CmentBd: 90 Max. :1600.0   
 (Other):178   
 Exter\_Cond Foundation Bsmt\_Qual Bsmt\_Cond   
 Typical :1787 CBlock:880 Typical :911 Good : 80   
 Good : 213 PConc :911 Good :849 Typical :1833   
 Fair : 43 Wood : 4 Excellent :178 Poor : 4   
 Excellent: 9 BrkTil:216 No\_Basement: 57 No\_Basement: 57   
 Poor : 1 Slab : 36 Fair : 57 Fair : 76   
 Stone : 6 Poor : 1 Excellent : 3   
   
 Bsmt\_Exposure BsmtFin\_Type\_1 BsmtFin\_SF\_1 BsmtFin\_Type\_2  
 Gd : 199 BLQ :196 Min. :1.00 Unf :1740   
 No :1331 Rec :216 1st Qu.:3.00 LwQ : 64   
 Av : 284 ALQ :298 Median :3.00 BLQ : 47   
 Mn : 179 GLQ :578 Mean :4.21 Rec : 79   
 No\_Basement: 60 Unf :602 3rd Qu.:7.00 GLQ : 23   
 LwQ :106 Max. :7.00 No\_Basement: 58   
 No\_Basement: 57 ALQ : 42   
 BsmtFin\_SF\_2 Bsmt\_Unf\_SF Total\_Bsmt\_SF Heating   
 Min. : 0.00 Min. : 0.0 Min. : 0 GasA :2019   
 1st Qu.: 0.00 1st Qu.: 226.0 1st Qu.: 793 GasW : 21   
 Median : 0.00 Median : 460.0 Median : 988 Grav : 6   
 Mean : 52.57 Mean : 561.2 Mean :1055 Wall : 5   
 3rd Qu.: 0.00 3rd Qu.: 801.0 3rd Qu.:1304 Floor: 1   
 Max. :1526.00 Max. :2336.0 Max. :5095 OthW : 1   
   
 Heating\_QC Central\_Air Electrical First\_Flr\_SF Second\_Flr\_SF   
 Fair : 61 Y:1916 SBrkr :1887 Min. : 432 Min. : 0.0   
 Typical : 618 N: 137 FuseA : 126 1st Qu.: 882 1st Qu.: 0.0   
 Excellent:1040 FuseF : 33 Median :1088 Median : 0.0   
 Good : 333 FuseP : 6 Mean :1168 Mean : 326.1   
 Poor : 1 Unknown: 1 3rd Qu.:1402 3rd Qu.: 701.0   
 Max. :5095 Max. :1862.0   
   
 Low\_Qual\_Fin\_SF Gr\_Liv\_Area Bsmt\_Full\_Bath Bsmt\_Half\_Bath   
 Min. : 0.000 Min. : 480 Min. :0.0000 Min. :0.00000   
 1st Qu.: 0.000 1st Qu.:1137 1st Qu.:0.0000 1st Qu.:0.00000   
 Median : 0.000 Median :1447 Median :0.0000 Median :0.00000   
 Mean : 4.973 Mean :1499 Mean :0.4301 Mean :0.05796   
 3rd Qu.: 0.000 3rd Qu.:1737 3rd Qu.:1.0000 3rd Qu.:0.00000   
 Max. :1064.000 Max. :5095 Max. :3.0000 Max. :2.00000   
   
 Full\_Bath Half\_Bath Bedroom\_AbvGr Kitchen\_AbvGr   
 Min. :0.000 Min. :0.0000 Min. :0.000 Min. :1.000   
 1st Qu.:1.000 1st Qu.:0.0000 1st Qu.:2.000 1st Qu.:1.000   
 Median :2.000 Median :0.0000 Median :3.000 Median :1.000   
 Mean :1.564 Mean :0.3751 Mean :2.855 Mean :1.047   
 3rd Qu.:2.000 3rd Qu.:1.0000 3rd Qu.:3.000 3rd Qu.:1.000   
 Max. :4.000 Max. :2.0000 Max. :6.000 Max. :3.000   
   
 Kitchen\_Qual TotRms\_AbvGrd Functional Fireplaces   
 Typical :1070 Min. : 3.000 Typ :1896 Min. :0.000   
 Good : 790 1st Qu.: 5.000 Min2 : 54 1st Qu.:0.000   
 Excellent: 142 Median : 6.000 Min1 : 51 Median :1.000   
 Fair : 50 Mean : 6.442 Mod : 27 Mean :0.603   
 Poor : 1 3rd Qu.: 7.000 Maj1 : 15 3rd Qu.:1.000   
 Max. :15.000 Maj2 : 6 Max. :4.000   
 (Other): 4   
 Fireplace\_Qu Garage\_Type Garage\_Finish Garage\_Cars   
 Good :538 Attchd :1204 Fin :509 Min. :0.000   
 No\_Fireplace:993 BuiltIn : 127 Unf :872 1st Qu.:1.000   
 Typical :409 Basment : 29 RFn :563 Median :2.000   
 Poor : 36 Detchd : 549 No\_Garage:109 Mean :1.774   
 Excellent : 21 No\_Garage : 108 3rd Qu.:2.000   
 Fair : 56 CarPort : 15 Max. :5.000   
 More\_Than\_Two\_Types: 21   
 Garage\_Area Garage\_Qual Garage\_Cond Paved\_Drive   
 Min. : 0 Typical :1839 Typical :1872 Partial\_Pavement: 42   
 1st Qu.: 320 No\_Garage: 109 No\_Garage: 109 Paved :1848   
 Median : 478 Fair : 85 Fair : 53 Dirt\_Gravel : 163   
 Mean : 472 Good : 16 Excellent: 1   
 3rd Qu.: 576 Excellent: 2 Poor : 8   
 Max. :1488 Poor : 2 Good : 10   
   
 Wood\_Deck\_SF Open\_Porch\_SF Enclosed\_Porch Three\_season\_porch  
 Min. : 0.00 Min. : 0.00 Min. : 0.00 Min. : 0.000   
 1st Qu.: 0.00 1st Qu.: 0.00 1st Qu.: 0.00 1st Qu.: 0.000   
 Median : 0.00 Median : 27.00 Median : 0.00 Median : 0.000   
 Mean : 93.52 Mean : 48.17 Mean : 23.02 Mean : 2.799   
 3rd Qu.: 168.00 3rd Qu.: 72.00 3rd Qu.: 0.00 3rd Qu.: 0.000   
 Max. :1424.00 Max. :742.00 Max. :584.00 Max. :407.000   
   
 Screen\_Porch Pool\_Area Pool\_QC Fence   
 Min. : 0.00 Min. : 0.000 No\_Pool :2047 No\_Fence :1661   
 1st Qu.: 0.00 1st Qu.: 0.000 Excellent: 2 Minimum\_Privacy : 225   
 Median : 0.00 Median : 0.000 Typical : 2 Good\_Privacy : 81   
 Mean : 16.68 Mean : 1.339 Fair : 1 Good\_Wood : 77   
 3rd Qu.: 0.00 3rd Qu.: 0.000 Good : 1 Minimum\_Wood\_Wire: 9   
 Max. :576.00 Max. :800.000   
   
 Misc\_Feature Misc\_Val Mo\_Sold Year\_Sold Sale\_Type   
 None:1978 Min. : 0.00 Min. : 1.000 Min. :2006 WD :1789   
 Gar2: 5 1st Qu.: 0.00 1st Qu.: 4.000 1st Qu.:2007 New : 163   
 Shed: 66 Median : 0.00 Median : 6.000 Median :2008 COD : 54   
 Othr: 3 Mean : 60.12 Mean : 6.189 Mean :2008 ConLD : 16   
 Elev: 1 3rd Qu.: 0.00 3rd Qu.: 8.000 3rd Qu.:2009 ConLI : 8   
 Max. :17000.00 Max. :12.000 Max. :2010 CWD : 8   
 (Other): 15   
 Sale\_Condition Longitude Latitude Above\_Median  
 Normal :1712 Min. :-93.69 Min. :41.99 No :1010   
 Partial: 169 1st Qu.:-93.66 1st Qu.:42.02 Yes:1043   
 Family : 30 Median :-93.64 Median :42.03   
 Abnorml: 121 Mean :-93.64 Mean :42.03   
 Alloca : 16 3rd Qu.:-93.62 3rd Qu.:42.05   
 AdjLand: 5 Max. :-93.58 Max. :42.06

#looking at these results, to decide which variables to remove or slim down

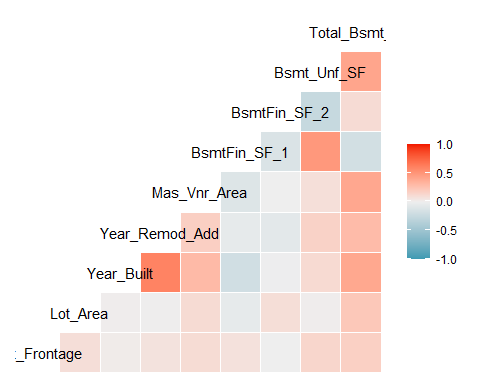
I also look at correlation between our predictor variables. To avoid multicollinearity, I want to observe and remove any predictor variables with correlation with one another.

amescorr1 = ames %>%  
 dplyr::select(  
 Lot\_Frontage,  
 Lot\_Area,  
 Year\_Built,  
 Year\_Remod\_Add,  
 Mas\_Vnr\_Area,  
 BsmtFin\_SF\_1,  
 BsmtFin\_SF\_2,  
 Bsmt\_Unf\_SF,  
 Total\_Bsmt\_SF  
 )  
  
amescorr2 = ames %>%  
 dplyr::select(  
 First\_Flr\_SF,  
 Second\_Flr\_SF,  
 Low\_Qual\_Fin\_SF,  
 Gr\_Liv\_Area,  
 Bsmt\_Full\_Bath,  
 Bsmt\_Half\_Bath,  
 Full\_Bath,  
 Half\_Bath,  
 Bedroom\_AbvGr,  
 Kitchen\_AbvGr,  
 TotRms\_AbvGrd,  
 Fireplaces,  
 Garage\_Cars,  
 Garage\_Area,  
 )  
  
amescorr3 = ames %>%  
 dplyr::select(  
 Wood\_Deck\_SF,  
 Open\_Porch\_SF,  
 Enclosed\_Porch,  
 Three\_season\_porch,  
 Screen\_Porch,  
 Pool\_Area,  
 Misc\_Val,  
 Year\_Sold,  
 Mo\_Sold,  
 Year\_Built,  
 Longitude,  
 Latitude  
 )  
  
ggcorr(ames)

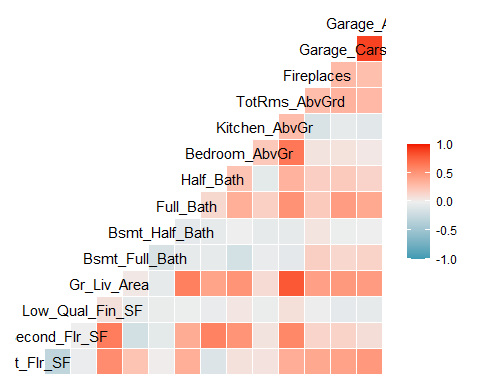
Warning in ggcorr(ames): data in column(s) 'MS\_SubClass', 'MS\_Zoning',  
'Street', 'Alley', 'Lot\_Shape', 'Land\_Contour', 'Utilities', 'Lot\_Config',  
'Land\_Slope', 'Neighborhood', 'Condition\_1', 'Condition\_2', 'Bldg\_Type',  
'House\_Style', 'Overall\_Qual', 'Overall\_Cond', 'Roof\_Style', 'Roof\_Matl',  
'Exterior\_1st', 'Exterior\_2nd', 'Mas\_Vnr\_Type', 'Exter\_Qual', 'Exter\_Cond',  
'Foundation', 'Bsmt\_Qual', 'Bsmt\_Cond', 'Bsmt\_Exposure', 'BsmtFin\_Type\_1',  
'BsmtFin\_Type\_2', 'Heating', 'Heating\_QC', 'Central\_Air', 'Electrical',  
'Kitchen\_Qual', 'Functional', 'Fireplace\_Qu', 'Garage\_Type', 'Garage\_Finish',  
'Garage\_Qual', 'Garage\_Cond', 'Paved\_Drive', 'Pool\_QC', 'Fence',  
'Misc\_Feature', 'Sale\_Type', 'Sale\_Condition', 'Above\_Median' are not numeric  
and were ignored



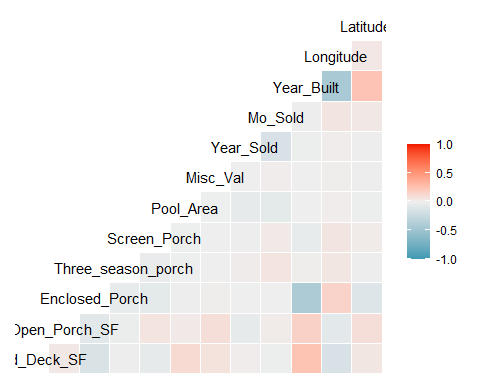
ggcorr(amescorr1)



ggcorr(amescorr2)



ggcorr(amescorr3)



With this information in mind, I remove variables from the Ames dataset. I selected these based on graphical representation of the variable vs. the response variable, variance, and strong co-linearity with other predictor variables. This set had a lot of significant multicollinearity for example, Garage Area and Garage Cars were strongly correlated, so I removed Garage Area. Ground Living Area was correlated with many variables, such as Total Rooms Above Ground, Second Floor Square Footage, and First Floor Square Footage. Other variables, such as Heating QC had poor variance, with a massively larger count of one level than the others. This would interfere with the model and needed removal. Some variables graphically demonstrated no relationship or a poor relationship with Above\_Median, such as Overall Condition, and were removed due to this.

ames\_cleaned = ames %>%  
 dplyr::select(  
 -Street,   
 -Alley,   
 -Utilities,   
 -Lot\_Config,  
 -Land\_Slope,   
 -Land\_Contour,  
 -Condition\_1,  
 -Condition\_2,   
 -Roof\_Style,  
 -Roof\_Matl,  
 -BsmtFin\_Type\_1,   
 -BsmtFin\_SF\_1,  
 -BsmtFin\_Type\_2,  
 -BsmtFin\_SF\_2,  
 -Bsmt\_Cond,  
 -Bsmt\_Exposure,  
 -Bsmt\_Unf\_SF,  
 -Bsmt\_Qual,  
 -Heating,   
 -Electrical,   
 -Functional,   
 -Heating\_QC,   
 -Fence,   
 -Garage\_Cond,  
 -Garage\_Qual,  
 -Garage\_Finish,  
 -Functional,  
 -Misc\_Feature,  
 -Sale\_Type,  
 -Pool\_QC,  
 -Lot\_Frontage,  
 -Bldg\_Type,  
 -Overall\_Cond,  
 -First\_Flr\_SF,  
 -Second\_Flr\_SF,  
 -Low\_Qual\_Fin\_SF,  
 -Wood\_Deck\_SF,  
 -Open\_Porch\_SF,  
 -Enclosed\_Porch,  
 -Three\_season\_porch,  
 -Screen\_Porch,  
 -Wood\_Deck\_SF,  
 -Mas\_Vnr\_Area,  
 -Bsmt\_Full\_Bath,  
 -Bsmt\_Half\_Bath,  
 -Year\_Remod\_Add,  
 -Paved\_Drive,  
 -Fireplaces,  
 -Fireplace\_Qu,  
 -Mas\_Vnr\_Type,  
 -Exterior\_2nd,  
 -Garage\_Area,   
 -TotRms\_AbvGrd,   
 -Year\_Remod\_Add,   
 -Bedroom\_AbvGr,   
 -Longitude,  
 -Latitude,  
 -Exterior\_1st,  
 -Exter\_Qual)

Now it’s time to build our models. I start by splitting the data into a training set and a testing set with strata set as Above\_Median.

#Splitting into a training and testing set. We will build the models off of the training set and then test their accuracy on the testing set. Setting Above\_Median as strata guarantees a distribution in both sets centered around the response variable.  
  
set.seed(123)   
ames\_cleaned\_split = initial\_split(ames\_cleaned, prop = 0.70, strata = Above\_Median)  
train = training(ames\_cleaned\_split)  
test = testing(ames\_cleaned\_split)

## Logistic Regression Model

Next I construct the logistic regression model. I chose logistic regression due to the binary nature of the response variable, Above\_Median. I use step\_other for variables that have levels of smaller count. This will improve the model by avoiding excessive dummy variables for less-frequent levels. It will condense these less-frequent levels into an “other” category. I use step\_dummy to prepare factor variables into a numeric representation to enable machine modeling.

#Setting up the logistic regression model  
ames\_log\_model =  
 logistic\_reg(mode="classification")%>%  
 set\_engine("glm")  
  
#recipe for log reg using step\_other to condense and step\_dummy to set our factor variables as such  
ames\_log\_recipe = recipe(Above\_Median~.,train)%>%  
 step\_other(MS\_SubClass, threshold = 0.01) %>%  
 step\_other(Neighborhood, threshold = 0.01) %>%  
 step\_other(Overall\_Qual, threshold = 0.01) %>%  
 step\_other(Kitchen\_Qual, threshold = 0.01) %>%  
 step\_other(House\_Style, threshold = 0.01) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
#combining together recipe and model  
logreg\_wf=workflow()%>%  
 add\_recipe(ames\_log\_recipe)%>%  
 add\_model(ames\_log\_model)  
  
#fitting the model  
ames\_log\_fit = fit(logreg\_wf, train)

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

#summarizing the results of our model  
summary(ames\_log\_fit$fit$fit$fit)

Call:  
stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
  
Coefficients: (1 not defined because of singularities)  
 Estimate Std. Error  
(Intercept) 2.512e+02 2.281e+02  
Lot\_Area 7.320e-05 3.724e-05  
Year\_Built 2.921e-02 1.658e-02  
Total\_Bsmt\_SF 1.431e-03 6.651e-04  
Gr\_Liv\_Area 4.464e-03 7.756e-04  
Full\_Bath 9.816e-01 3.963e-01  
Half\_Bath 3.009e-01 4.088e-01  
Kitchen\_AbvGr -1.186e+01 7.575e+00  
Garage\_Cars 1.667e+00 4.135e-01  
Pool\_Area 2.635e-02 5.495e+00  
Misc\_Val 2.446e-04 1.107e-04  
Mo\_Sold 3.133e-02 5.621e-02  
Year\_Sold -1.547e-01 1.134e-01  
MS\_SubClass\_Two\_Story\_1946\_and\_Newer 5.434e+00 2.743e+00  
MS\_SubClass\_One\_Story\_PUD\_1946\_and\_Newer -3.852e-01 9.765e-01  
MS\_SubClass\_One\_and\_Half\_Story\_Finished\_All\_Ages 4.828e+00 2.299e+00  
MS\_SubClass\_Two\_Story\_PUD\_1946\_and\_Newer -1.814e+01 1.340e+03  
MS\_SubClass\_Split\_or\_Multilevel 2.278e+01 8.658e+03  
MS\_SubClass\_One\_Story\_1945\_and\_Older -8.929e-01 1.614e+00  
MS\_SubClass\_Duplex\_All\_Styles\_and\_Ages 5.669e+00 7.556e+00  
MS\_SubClass\_Split\_Foyer 4.339e+00 7.573e+00  
MS\_SubClass\_Two\_Family\_conversion\_All\_Styles\_and\_Ages 6.169e-01 1.527e+00  
MS\_SubClass\_Two\_Story\_1945\_and\_Older 4.768e+00 2.680e+00  
MS\_SubClass\_Two\_and\_Half\_Story\_All\_Ages -1.359e-01 8.871e+00  
MS\_SubClass\_other -5.114e+00 3.057e+00  
MS\_Zoning\_Residential\_High\_Density 5.295e+00 1.842e+00  
MS\_Zoning\_Floating\_Village\_Residential 8.702e+00 4.828e+03  
MS\_Zoning\_Residential\_Medium\_Density -1.134e+00 9.039e-01  
MS\_Zoning\_C\_all -2.070e+01 3.829e+03  
MS\_Zoning\_A\_agr -1.152e+01 2.963e+04  
MS\_Zoning\_I\_all -3.085e+01 2.923e+04  
Lot\_Shape\_Regular -4.903e-02 3.322e-01  
Lot\_Shape\_Moderately\_Irregular -1.385e+00 1.252e+00  
Lot\_Shape\_Irregular 4.883e+00 2.546e+01  
Neighborhood\_Gilbert 4.789e-01 1.022e+00  
Neighborhood\_Stone\_Brook 3.452e+01 3.399e+03  
Neighborhood\_Northwest\_Ames 4.213e-01 5.808e-01  
Neighborhood\_Somerset 1.476e+01 4.638e+03  
Neighborhood\_Northridge\_Heights 1.698e+01 1.340e+03  
Neighborhood\_Northridge 1.248e+01 3.406e+03  
Neighborhood\_Sawyer\_West -6.529e-01 9.027e-01  
Neighborhood\_Sawyer 5.033e-01 5.985e-01  
Neighborhood\_Old\_Town -1.999e-01 1.305e+00  
Neighborhood\_Brookside 1.888e+00 1.070e+00  
Neighborhood\_Iowa\_DOT\_and\_Rail\_Road 3.360e+00 1.534e+00  
Neighborhood\_Clear\_Creek 3.591e+00 1.233e+00  
Neighborhood\_South\_and\_West\_of\_Iowa\_State\_University 1.906e-01 1.242e+00  
Neighborhood\_Edwards 2.064e-01 7.185e-01  
Neighborhood\_College\_Creek -1.499e-01 8.361e-01  
Neighborhood\_Crawford 3.612e+00 1.098e+00  
Neighborhood\_Mitchell 1.138e+00 6.671e-01  
Neighborhood\_Timberland 1.372e+00 1.454e+00  
Neighborhood\_Meadow\_Village -1.431e+01 5.048e+03  
Neighborhood\_other 6.917e-01 1.039e+00  
House\_Style\_Two\_Story -5.544e+00 2.769e+00  
House\_Style\_One\_and\_Half\_Fin -4.902e+00 2.258e+00  
House\_Style\_SLvl -2.230e+01 8.658e+03  
House\_Style\_SFoyer -3.895e+00 7.542e+00  
House\_Style\_Two\_and\_Half\_Unf 1.473e+00 8.900e+00  
House\_Style\_other 3.005e+00 8.849e+00  
Overall\_Qual\_Average -7.451e-01 3.553e-01  
Overall\_Qual\_Good 1.712e+00 5.285e-01  
Overall\_Qual\_Very\_Good 7.952e+00 3.255e+00  
Overall\_Qual\_Excellent -6.218e+00 1.645e+00  
Overall\_Qual\_Below\_Average -3.780e+00 1.141e+00  
Overall\_Qual\_Fair -4.042e+01 4.632e+03  
Overall\_Qual\_Very\_Excellent 1.050e+01 4.654e+03  
Overall\_Qual\_other -6.082e+00 4.824e+03  
Exter\_Cond\_Good 5.022e-01 4.915e-01  
Exter\_Cond\_Fair -3.632e+00 1.457e+00  
Exter\_Cond\_Excellent 1.926e+00 1.501e+00  
Exter\_Cond\_Poor NA NA  
Foundation\_PConc 1.193e+00 4.813e-01  
Foundation\_Wood -4.354e+00 5.144e+00  
Foundation\_BrkTil 3.922e-01 8.039e-01  
Foundation\_Slab -4.392e+00 3.562e+00  
Foundation\_Stone 1.363e+01 2.112e+03  
Central\_Air\_N -2.516e+00 1.265e+00  
Kitchen\_Qual\_Good 4.737e-01 3.772e-01  
Kitchen\_Qual\_Excellent 3.579e+00 1.306e+00  
Kitchen\_Qual\_Fair -1.484e+00 1.510e+00  
Kitchen\_Qual\_other -1.566e+01 2.923e+04  
Garage\_Type\_BuiltIn 1.890e+00 1.060e+00  
Garage\_Type\_Basment 1.490e+00 9.262e-01  
Garage\_Type\_Detchd -1.195e+00 4.654e-01  
Garage\_Type\_No\_Garage -4.168e+00 8.064e+00  
Garage\_Type\_CarPort -2.132e+01 5.695e+03  
Garage\_Type\_More\_Than\_Two\_Types 1.204e+00 1.318e+00  
Sale\_Condition\_Partial -1.812e+00 9.273e-01  
Sale\_Condition\_Family -4.476e+00 1.437e+00  
Sale\_Condition\_Abnorml -2.953e+00 7.304e-01  
Sale\_Condition\_Alloca 6.398e+00 5.065e+00  
Sale\_Condition\_AdjLand -1.254e+01 1.894e+04  
 z value Pr(>|z|)   
(Intercept) 1.101 0.270810   
Lot\_Area 1.966 0.049354 \*   
Year\_Built 1.761 0.078167 .   
Total\_Bsmt\_SF 2.152 0.031404 \*   
Gr\_Liv\_Area 5.756 8.63e-09 \*\*\*  
Full\_Bath 2.477 0.013255 \*   
Half\_Bath 0.736 0.461683   
Kitchen\_AbvGr -1.566 0.117378   
Garage\_Cars 4.032 5.52e-05 \*\*\*  
Pool\_Area 0.005 0.996174   
Misc\_Val 2.211 0.027061 \*   
Mo\_Sold 0.557 0.577290   
Year\_Sold -1.363 0.172803   
MS\_SubClass\_Two\_Story\_1946\_and\_Newer 1.981 0.047599 \*   
MS\_SubClass\_One\_Story\_PUD\_1946\_and\_Newer -0.395 0.693206   
MS\_SubClass\_One\_and\_Half\_Story\_Finished\_All\_Ages 2.100 0.035727 \*   
MS\_SubClass\_Two\_Story\_PUD\_1946\_and\_Newer -0.014 0.989193   
MS\_SubClass\_Split\_or\_Multilevel 0.003 0.997901   
MS\_SubClass\_One\_Story\_1945\_and\_Older -0.553 0.580030   
MS\_SubClass\_Duplex\_All\_Styles\_and\_Ages 0.750 0.453127   
MS\_SubClass\_Split\_Foyer 0.573 0.566708   
MS\_SubClass\_Two\_Family\_conversion\_All\_Styles\_and\_Ages 0.404 0.686301   
MS\_SubClass\_Two\_Story\_1945\_and\_Older 1.779 0.075246 .   
MS\_SubClass\_Two\_and\_Half\_Story\_All\_Ages -0.015 0.987776   
MS\_SubClass\_other -1.673 0.094390 .   
MS\_Zoning\_Residential\_High\_Density 2.874 0.004052 \*\*   
MS\_Zoning\_Floating\_Village\_Residential 0.002 0.998562   
MS\_Zoning\_Residential\_Medium\_Density -1.254 0.209686   
MS\_Zoning\_C\_all -0.005 0.995687   
MS\_Zoning\_A\_agr 0.000 0.999690   
MS\_Zoning\_I\_all -0.001 0.999158   
Lot\_Shape\_Regular -0.148 0.882675   
Lot\_Shape\_Moderately\_Irregular -1.106 0.268812   
Lot\_Shape\_Irregular 0.192 0.847891   
Neighborhood\_Gilbert 0.469 0.639426   
Neighborhood\_Stone\_Brook 0.010 0.991897   
Neighborhood\_Northwest\_Ames 0.725 0.468241   
Neighborhood\_Somerset 0.003 0.997460   
Neighborhood\_Northridge\_Heights 0.013 0.989884   
Neighborhood\_Northridge 0.004 0.997075   
Neighborhood\_Sawyer\_West -0.723 0.469487   
Neighborhood\_Sawyer 0.841 0.400324   
Neighborhood\_Old\_Town -0.153 0.878288   
Neighborhood\_Brookside 1.765 0.077589 .   
Neighborhood\_Iowa\_DOT\_and\_Rail\_Road 2.190 0.028534 \*   
Neighborhood\_Clear\_Creek 2.914 0.003574 \*\*   
Neighborhood\_South\_and\_West\_of\_Iowa\_State\_University 0.153 0.878044   
Neighborhood\_Edwards 0.287 0.773969   
Neighborhood\_College\_Creek -0.179 0.857730   
Neighborhood\_Crawford 3.290 0.001003 \*\*   
Neighborhood\_Mitchell 1.705 0.088169 .   
Neighborhood\_Timberland 0.944 0.345325   
Neighborhood\_Meadow\_Village -0.003 0.997738   
Neighborhood\_other 0.665 0.505780   
House\_Style\_Two\_Story -2.002 0.045306 \*   
House\_Style\_One\_and\_Half\_Fin -2.171 0.029928 \*   
House\_Style\_SLvl -0.003 0.997944   
House\_Style\_SFoyer -0.516 0.605508   
House\_Style\_Two\_and\_Half\_Unf 0.165 0.868568   
House\_Style\_other 0.340 0.734198   
Overall\_Qual\_Average -2.097 0.035968 \*   
Overall\_Qual\_Good 3.240 0.001197 \*\*   
Overall\_Qual\_Very\_Good 2.443 0.014573 \*   
Overall\_Qual\_Excellent -3.780 0.000157 \*\*\*  
Overall\_Qual\_Below\_Average -3.313 0.000922 \*\*\*  
Overall\_Qual\_Fair -0.009 0.993037   
Overall\_Qual\_Very\_Excellent 0.002 0.998199   
Overall\_Qual\_other -0.001 0.998994   
Exter\_Cond\_Good 1.022 0.306875   
Exter\_Cond\_Fair -2.493 0.012672 \*   
Exter\_Cond\_Excellent 1.283 0.199501   
Exter\_Cond\_Poor NA NA   
Foundation\_PConc 2.478 0.013204 \*   
Foundation\_Wood -0.846 0.397321   
Foundation\_BrkTil 0.488 0.625633   
Foundation\_Slab -1.233 0.217613   
Foundation\_Stone 0.006 0.994853   
Central\_Air\_N -1.989 0.046742 \*   
Kitchen\_Qual\_Good 1.256 0.209087   
Kitchen\_Qual\_Excellent 2.741 0.006123 \*\*   
Kitchen\_Qual\_Fair -0.983 0.325617   
Kitchen\_Qual\_other -0.001 0.999573   
Garage\_Type\_BuiltIn 1.783 0.074584 .   
Garage\_Type\_Basment 1.609 0.107631   
Garage\_Type\_Detchd -2.568 0.010242 \*   
Garage\_Type\_No\_Garage -0.517 0.605230   
Garage\_Type\_CarPort -0.004 0.997013   
Garage\_Type\_More\_Than\_Two\_Types 0.913 0.361147   
Sale\_Condition\_Partial -1.954 0.050686 .   
Sale\_Condition\_Family -3.115 0.001841 \*\*   
Sale\_Condition\_Abnorml -4.042 5.29e-05 \*\*\*  
Sale\_Condition\_Alloca 1.263 0.206574   
Sale\_Condition\_AdjLand -0.001 0.999472   
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
(Dispersion parameter for binomial family taken to be 1)  
  
 Null deviance: 1991.74 on 1436 degrees of freedom  
Residual deviance: 375.19 on 1345 degrees of freedom  
AIC: 559.19  
  
Number of Fisher Scoring iterations: 20

My model has an AIC of 559.19. By itself this does not necessarily mean much. However, if we were to add or subtract variables from the model, I could use the AIC to compare the quality of the model. A lower AIC is considered to be an increase in quality.

From our model we can see which variables are significant in the determining the probability of a home selling above or below the median home value. Most appear to follow a logical pattern, but I am not content with House Style and Overall Quality. House Style’s negative coefficient for Two Story appears to be in direct contract with MS\_Subclass’s positive coefficient for Two Story 1946 and Newer. Furthermore, House Style contradicts the visualization I performed in the preliminary descriptive data analysis.

With this in mind I will continue to assess the performance of this model. I will use accuracy, sensitivity, specificity, and AUC to determine model quality on both the training and testing sets.

#Developing predictions  
predictions = predict(ames\_log\_fit, train, type="prob")

Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
prediction from rank-deficient fit; attr(\*, "non-estim") has doubtful cases

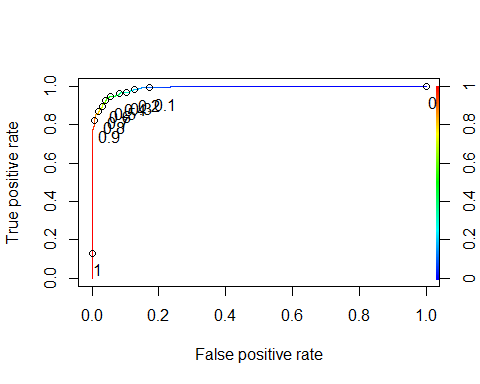
head(predictions)

# A tibble: 6 × 2  
 .pred\_No .pred\_Yes  
 <dbl> <dbl>  
1 0.937 6.31e- 2  
2 0.999 5.03e- 4  
3 1 2.22e-16  
4 1 2.22e-16  
5 0.994 6.06e- 3  
6 0.640 3.60e- 1

#ROCR plot  
predictions = predict(ames\_log\_fit, train, type="prob")[2] #extracting the "yes" prediction

Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
prediction from rank-deficient fit; attr(\*, "non-estim") has doubtful cases

ROCRpred = prediction(predictions, train$Above\_Median)  
ROCRperf = performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



#AUC  
as.numeric(performance(ROCRpred, "auc")@y.values)

[1] 0.9895119

#balance sensitivity & specificity - cut off/threshold  
opt.cut = function(perf, pred){  
 cut.ind = mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],   
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf, ROCRpred))

[,1]  
sensitivity 0.9383562  
specificity 0.9561528  
cutoff 0.5606287

From this, we have an AUC of 0.9895119. The closer an AUC is to 1, the better.

#I then use the cutoff value found from ROCR and used this to create a confusion matrix and pinpoint the accuracy of my model with this cutoff.  
  
t1 = table(train$Above\_Median,predictions > 0.5606287)  
t1

FALSE TRUE  
 No 676 31  
 Yes 45 685

(t1[1,1]+t1[2,2])/nrow(train)

[1] 0.947112

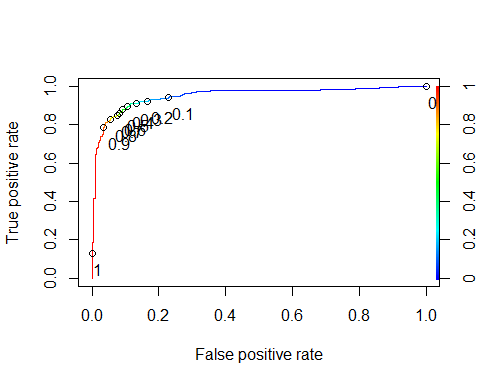
The accuracy of the model on the training set with a cut off of 0.5606287 is 0.946112.

Next we will check the accuracy of the model on the testing set.

#Developing predictions  
predictions2 = predict(ames\_log\_fit, test, type="prob")  
head(predictions2)

# A tibble: 6 × 2  
 .pred\_No .pred\_Yes  
 <dbl> <dbl>  
1 6.97e- 1 0.303  
2 3.94e- 5 1.00   
3 2.22e-16 1   
4 2.22e-16 1   
5 2.22e-16 1   
6 4.84e-13 1.00

#ROCR plot  
predictions2 = predict(ames\_log\_fit, test, type="prob")[2] #extracting the "yes" prediction  
ROCRpred = prediction(predictions2, test$Above\_Median)  
ROCRperf = performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



#AUC  
as.numeric(performance(ROCRpred, "auc")@y.values)

[1] 0.9512121

#balance sensitivity & specificity - cut off/threshold  
opt.cut = function(perf, pred){  
 cut.ind = mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],   
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf, ROCRpred))

[,1]  
sensitivity 0.9105431  
specificity 0.8844884  
cutoff 0.3447664

AUC for the testing set is 0.9512121. This is lower than on the training set but is still very close to 1. The cutoff for the model on the testing set is 0.3447664. The sensitivity is 0.9105431 and specificity is 0.8844884.

t2 = table(test$Above\_Median,predictions2 > 0.3447664)  
t2

FALSE TRUE  
 No 268 35  
 Yes 29 284

(t2[1,1]+t2[2,2])/nrow(test)

[1] 0.8961039

Here we can see the accuracy on the testing set to be 0.8961039.

## Logistic Regression with Lasso

While the previous logistic regression had good accuracy, sensitivity, and specificity, I was not content with all of the variable coefficients. I decided to trial a lasso logistic regression to see if the model can be improved.

set.seed(123)  
folds=vfold\_cv(train,v=5)

#use models generates a code template.  
use\_glmnet(formula = Above\_Median~., data = train)

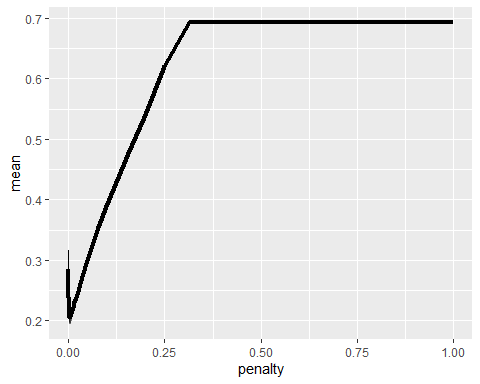
glmnet\_recipe <-   
 recipe(formula = Above\_Median ~ ., data = train) %>%   
 step\_zv(all\_predictors()) %>%   
 step\_normalize(all\_numeric\_predictors())   
  
glmnet\_spec <-   
 logistic\_reg(penalty = tune(), mixture = tune()) %>%   
 set\_mode("classification") %>%   
 set\_engine("glmnet")   
  
glmnet\_workflow <-   
 workflow() %>%   
 add\_recipe(glmnet\_recipe) %>%   
 add\_model(glmnet\_spec)   
  
glmnet\_grid <- tidyr::crossing(penalty = 10^seq(-6, -1, length.out = 20), mixture = c(0.05,   
 0.2, 0.4, 0.6, 0.8, 1))   
  
glmnet\_tune <-   
 tune\_grid(glmnet\_workflow, resamples = stop("add your rsample object"), grid = glmnet\_grid)

#I modified the usemodels template for glmnet by including step\_other and step\_dummy for the same purposes in the previous logistic regression. I also included step\_normalize to scale and center the variables as required by lasso regressions.  
  
glmnet\_recipe <-   
 recipe(formula = Above\_Median ~ ., data = train) %>%   
 step\_other(MS\_SubClass, threshold = 0.01) %>%  
 step\_other(MS\_Zoning, threshold = 0.01) %>%  
 step\_other(Exter\_Cond, threshold = 0.01) %>%  
 step\_other(Neighborhood, threshold = 0.01) %>%  
 step\_other(Overall\_Qual, threshold = 0.01) %>%  
 step\_other(Kitchen\_Qual, threshold = 0.05) %>% #I had to increase the threshold on Kitchen Quality to avoid variance issues by capturing the "other" categories  
 step\_other(House\_Style, threshold = 0.01) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())%>%  
 step\_normalize(all\_predictors(), -all\_nominal())  
  
glmnet\_spec <-   
 logistic\_reg(penalty = tune(), mixture = 1) %>% #mixture of 1 to trigger lasso (as opposed to 0 would trigger ridge)  
 set\_mode("classification") %>%   
 set\_engine("glmnet")   
  
glmnet\_workflow <-   
 workflow() %>%   
 add\_recipe(glmnet\_recipe) %>%   
 add\_model(glmnet\_spec)   
  
glmnet\_grid = grid\_regular(penalty(), levels = 100)  
  
#Using mean log loss for our lambda metric to generate probabilities.  
glmnet\_tune =  
 tune\_grid(glmnet\_workflow, resamples = folds,   
 grid = glmnet\_grid, metrics = metric\_set(mn\_log\_loss))

Next I will take this model and plot penalty vs mean of log loss. I want to try to find the optimal penalty value in order to be closest

glmnet\_tune %>%  
 collect\_metrics() %>%  
 ggplot(aes(penalty, mean)) +  
 geom\_errorbar(aes(  
 ymin = mean - std\_err,  
 ymax = mean + std\_err  
 ),  
 alpha = 0.5  
 ) +  
 geom\_line(size = 1.5) +  
 theme(legend.position = "none")

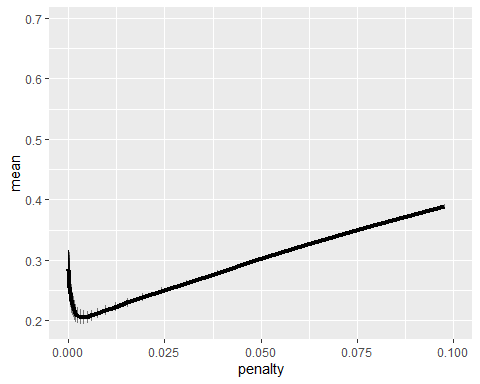
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
ℹ Please use `linewidth` instead.



Zooming in on this penalty value peak.

glmnet\_tune %>%  
 collect\_metrics() %>%  
 ggplot(aes(penalty, mean)) +  
 geom\_errorbar(aes(  
 ymin = mean - std\_err,  
 ymax = mean + std\_err  
 ),  
 alpha = 0.5  
 ) +  
 geom\_line(size = 1.5) +  
 theme(legend.position = "none") +  
 xlim(0,0.1)

Warning: Removed 10 rows containing missing values (`geom\_line()`).



This appears to peak very close to 0.000-0.005. Next I can extract this best value for mean of log loss.

best\_mnlog = glmnet\_tune %>%  
 select\_best("mn\_log\_loss")  
best\_mnlog

# A tibble: 1 × 2  
 penalty .config   
 <dbl> <chr>   
1 0.00376 Preprocessor1\_Model076

Our ideal penalty value is thus 0.003764936.

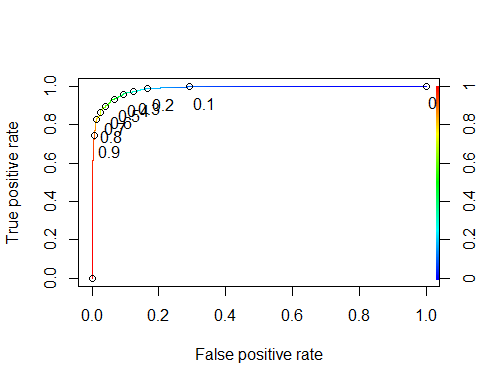
final\_lasso = glmnet\_workflow %>% finalize\_workflow(best\_mnlog)  
lasso\_fit = fit(final\_lasso, train)

tidy(lasso\_fit)

# A tibble: 89 × 3  
 term estimate penalty  
 <chr> <dbl> <dbl>  
 1 (Intercept) 0.328 0.00376  
 2 Lot\_Area 0.0765 0.00376  
 3 Year\_Built 0.264 0.00376  
 4 Total\_Bsmt\_SF 0.633 0.00376  
 5 Gr\_Liv\_Area 1.33 0.00376  
 6 Full\_Bath 0.477 0.00376  
 7 Half\_Bath 0.122 0.00376  
 8 Kitchen\_AbvGr -0.770 0.00376  
 9 Garage\_Cars 0.660 0.00376  
10 Pool\_Area 0 0.00376  
# ℹ 79 more rows

The coefficients for the lasso logistic regression appear more in line with logical expectations than the logistic regression alone. But how is the model performing? I will look at thresholds for this model.

#generating our predictions based on the model  
predictions = predict(lasso\_fit, train, type="prob")[2]  
  
#generating the ROC  
ROCRpred = prediction(predictions, train$Above\_Median)   
ROCRperf = performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



#AUC  
as.numeric(performance(ROCRpred, "auc")@y.values)

[1] 0.9853888

#balance sensitivity & specificity - cut off/threshold  
opt.cut = function(perf, pred){  
 cut.ind = mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],   
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf, ROCRpred))

[,1]  
sensitivity 0.9315068  
specificity 0.9405941  
cutoff 0.5121469

For this model on the training set, the sensitivity is 0.9315068, specificity is 0.9405941, and AUC is 0.9853888. Let’s use the cutoff value to find the accuracy.

t3 = table(train$Above\_Median,predictions > 0.5121469)  
t3

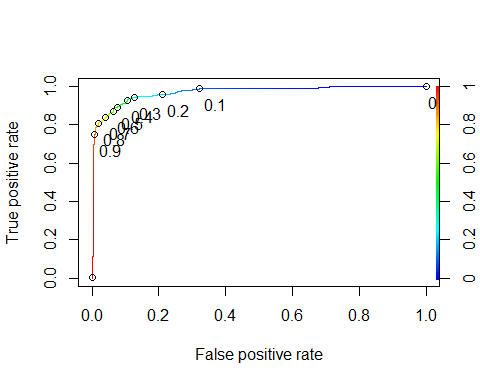
FALSE TRUE  
 No 665 42  
 Yes 51 679

(t3[1,1]+t3[2,2])/nrow(train)

[1] 0.9352818

This model has an accuracy of 0.93528118 on the training set.

#generating our predictions based on the model  
predictions = predict(lasso\_fit, test, type="prob")[2]  
  
#generating the ROC  
ROCRpred = prediction(predictions, test$Above\_Median)   
ROCRperf = performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



#AUC  
as.numeric(performance(ROCRpred, "auc")@y.values)

[1] 0.9695378

#balance sensitivity & specificity - cut off/threshold  
opt.cut = function(perf, pred){  
 cut.ind = mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],   
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf, ROCRpred))

[,1]  
sensitivity 0.9329073  
specificity 0.8943894  
cutoff 0.3949216

For this model on the testing set, the AUC is 0.9695378, sensitivity is 0.9329073, and specificity is 0.8943894. All of these out-perform the previous logistic regression on the testing set (0.9512121 AUC, sensitivity 0.9105431 and specificity 0.8844884). We will use the cut-off value to see if it out-performs with accuracy as well.

t4 = table(test$Above\_Median,predictions > 0.3949216)  
t4

FALSE TRUE  
 No 271 32  
 Yes 21 292

(t4[1,1]+t4[2,2])/nrow(test)

[1] 0.913961

On the testing set the logistic lasso regression has an accuracy of 0.913961. This out-performs the previous logistic regression (accuracy on testing set of 0.8961039).

Overall, the logistic lasso regression out-performs the previous logistic regression without lasso when applied to the testing set for both AUC and accuracy. Furthermore, the coefficients generated make more logical sense and correspond with the previous descriptive analysis.

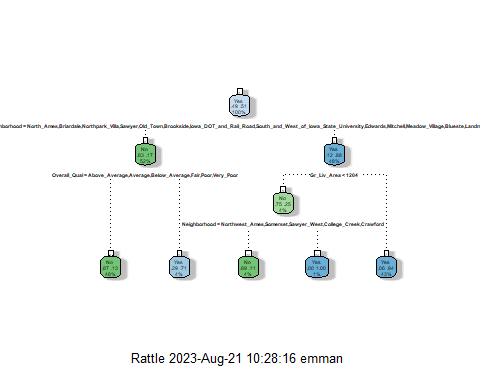
## Classification Tree

Classification trees are decision trees that allow predictions to be made based off of the interaction between variables. I first built a simple classification tree off of the training set.

#Building the tree  
ames\_classtree\_recipe = recipe(Above\_Median~., train)  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>%  
 set\_mode("classification")  
  
ames\_classtree\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(ames\_classtree\_recipe)  
  
ames\_classtree\_fit = fit(ames\_classtree\_wflow, train)  
  
#tree fit  
tree = ames\_classtree\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
ℹ Please use `extract\_fit\_parsnip()` instead.

fancyRpartPlot(tree, tweak=0.75)



The tree looks alright, if a bit difficult to read. Next I need to assess the accuracy of the tree. I do this using a confusion matrix.

#Confusion matrix for the simple tree  
treepred = predict(ames\_classtree\_fit, train, type = "class")  
head(treepred)

# A tibble: 6 × 1  
 .pred\_class  
 <fct>   
1 No   
2 No   
3 No   
4 No   
5 No   
6 No

confusionMatrix(treepred$.pred\_class,train$Above\_Median,positive="Yes")

Confusion Matrix and Statistics  
  
 Reference  
Prediction No Yes  
 No 654 95  
 Yes 53 635  
   
 Accuracy : 0.897   
 95% CI : (0.8801, 0.9122)  
 No Information Rate : 0.508   
 P-Value [Acc > NIR] : < 2.2e-16   
   
 Kappa : 0.7942   
   
 Mcnemar's Test P-Value : 0.0007512   
   
 Sensitivity : 0.8699   
 Specificity : 0.9250   
 Pos Pred Value : 0.9230   
 Neg Pred Value : 0.8732   
 Prevalence : 0.5080   
 Detection Rate : 0.4419   
 Detection Prevalence : 0.4788   
 Balanced Accuracy : 0.8974   
   
 'Positive' Class : Yes

The initial tree is not too bad. I have a sensitivity of 0.8699, specificity of 0.9250, and accuracy of 0.8974. However, I’m wondering if I can improve these.

On the training set this tree has an accuracy of 0.8970, sensitivity of 0.8699, and specificity of 0.9250. I want to see how this performs on the testing set.

treepred = predict(ames\_classtree\_fit, test, type = "class")  
head(treepred)

# A tibble: 6 × 1  
 .pred\_class  
 <fct>   
1 No   
2 Yes   
3 Yes   
4 Yes   
5 Yes   
6 Yes

confusionMatrix(treepred$.pred\_class,test$Above\_Median,positive="Yes")

Confusion Matrix and Statistics  
  
 Reference  
Prediction No Yes  
 No 267 47  
 Yes 36 266  
   
 Accuracy : 0.8653   
 95% CI : (0.8357, 0.8912)  
 No Information Rate : 0.5081   
 P-Value [Acc > NIR] : <2e-16   
   
 Kappa : 0.7306   
   
 Mcnemar's Test P-Value : 0.2724   
   
 Sensitivity : 0.8498   
 Specificity : 0.8812   
 Pos Pred Value : 0.8808   
 Neg Pred Value : 0.8503   
 Prevalence : 0.5081   
 Detection Rate : 0.4318   
 Detection Prevalence : 0.4903   
 Balanced Accuracy : 0.8655   
   
 'Positive' Class : Yes

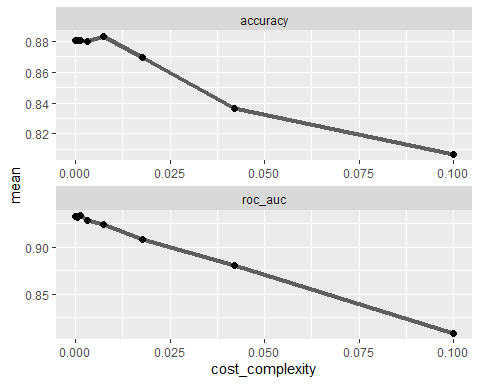
On the testing set, this tree has an accuracy of 0.8653, sensitivity of 0.8498, and specificity of 0.8812.

I want to improve this tree. To maximize the accuracy of my classification tree, I want to find the most optimal complexity parameter, or ‘cp’ value. I am going to have R do this for me.

set.seed(123) #Specifies randomness to makes sure the randomness in the code will generate the same results.  
folds=vfold\_cv(train,v=5) #5-fold cross-validation on the training data  
  
#Recipe for the tree  
ames\_classtree2\_recipe = recipe(Above\_Median~., train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
#Model for the tree  
ames\_classtree2\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%  
 set\_mode("classification")  
  
#This specifies that we are looking for   
tree\_grid = grid\_regular(cost\_complexity(),  
 levels = 25)  
  
#Workflow for the tree  
ames\_tree2\_wflow =   
 workflow() %>%   
 add\_model(ames\_classtree2\_model) %>%   
 add\_recipe(ames\_classtree2\_recipe)  
  
#Will gather information on evaluation metrics after tuning  
tree\_res =   
 ames\_tree2\_wflow %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )  
  
tree\_res

# Tuning results  
# 5-fold cross-validation   
# A tibble: 5 × 4  
 splits id .metrics .notes   
 <list> <chr> <list> <list>   
1 <split [1149/288]> Fold1 <tibble [50 × 5]> <tibble [0 × 3]>  
2 <split [1149/288]> Fold2 <tibble [50 × 5]> <tibble [0 × 3]>  
3 <split [1150/287]> Fold3 <tibble [50 × 5]> <tibble [0 × 3]>  
4 <split [1150/287]> Fold4 <tibble [50 × 5]> <tibble [0 × 3]>  
5 <split [1150/287]> Fold5 <tibble [50 × 5]> <tibble [0 × 3]>

#This will extract metrics from our plotted cost complexity  
tree\_res %>%  
 collect\_metrics() %>%  
 ggplot(aes(cost\_complexity, mean)) +  
 geom\_line(size = 1.5, alpha = 0.6) +  
 geom\_point(size = 2) +  
 facet\_wrap(~ .metric, scales = "free", nrow = 2)



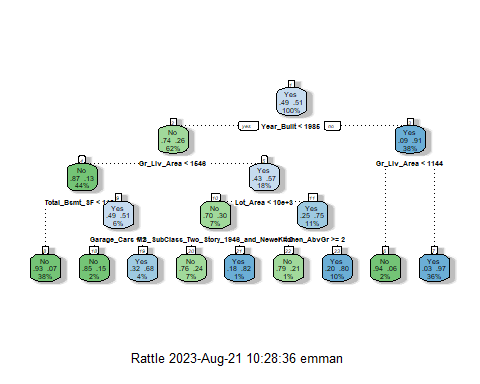
#Creates the best tree from the most optimal accuracy measurement  
  
best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

# A tibble: 1 × 2  
 cost\_complexity .config   
 <dbl> <chr>   
1 0.00750 Preprocessor1\_Model22

With the optimal accuracy and cp value captured in “best\_tree” object I can now plot the tree.

#Workflow for the new tree with the best\_tree object  
final\_classtree\_wf =   
 ames\_tree2\_wflow %>%   
 finalize\_workflow(best\_tree)  
  
#Fitting  
final\_fit = fit(final\_classtree\_wf, train)  
  
tree2 = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

fancyRpartPlot(tree2, tweak = 1.25)



Immediately, it’s notable that this tree has different variables included than the original tree.

treepred = predict(final\_fit, train, type = "class")  
head(treepred)

# A tibble: 6 × 1  
 .pred\_class  
 <fct>   
1 No   
2 No   
3 No   
4 No   
5 No   
6 No

confusionMatrix(treepred$.pred\_class,train$Above\_Median,positive="Yes")

Confusion Matrix and Statistics  
  
 Reference  
Prediction No Yes  
 No 643 70  
 Yes 64 660  
   
 Accuracy : 0.9068   
 95% CI : (0.8905, 0.9213)  
 No Information Rate : 0.508   
 P-Value [Acc > NIR] : <2e-16   
   
 Kappa : 0.8135   
   
 Mcnemar's Test P-Value : 0.6658   
   
 Sensitivity : 0.9041   
 Specificity : 0.9095   
 Pos Pred Value : 0.9116   
 Neg Pred Value : 0.9018   
 Prevalence : 0.5080   
 Detection Rate : 0.4593   
 Detection Prevalence : 0.5038   
 Balanced Accuracy : 0.9068   
   
 'Positive' Class : Yes

The new confusion matrix shows an improved accuracy of 0.9068. Sensitivity declined slightly to 0.9041 but Specificity increased to 0.9095. Overall this new tree has improved accuracy and specificity on the training. set.

Now we will look at how this tree performs on the testing set.

treepred = predict(final\_fit, test, type = "class")  
head(treepred)

# A tibble: 6 × 1  
 .pred\_class  
 <fct>   
1 No   
2 Yes   
3 Yes   
4 Yes   
5 Yes   
6 Yes

confusionMatrix(treepred$.pred\_class,test$Above\_Median,positive="Yes")

Confusion Matrix and Statistics  
  
 Reference  
Prediction No Yes  
 No 272 40  
 Yes 31 273  
   
 Accuracy : 0.8847   
 95% CI : (0.8568, 0.9089)  
 No Information Rate : 0.5081   
 P-Value [Acc > NIR] : <2e-16   
   
 Kappa : 0.7695   
   
 Mcnemar's Test P-Value : 0.3424   
   
 Sensitivity : 0.8722   
 Specificity : 0.8977   
 Pos Pred Value : 0.8980   
 Neg Pred Value : 0.8718   
 Prevalence : 0.5081   
 Detection Rate : 0.4432   
 Detection Prevalence : 0.4935   
 Balanced Accuracy : 0.8849   
   
 'Positive' Class : Yes

On the testing set we see an accuracy of 0.8847, sensitivity of 0.8722, and specificity of 0.8977. This is also improved from the previous tree.

## Random Forests

#recipe  
ames\_rf\_recipe = recipe(Above\_Median~., train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
#model  
rf\_model = rand\_forest() %>%   
 set\_engine("ranger", importance = "permutation") %>%  
 set\_mode("classification")  
  
#putting it all together  
ames\_rf\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(ames\_rf\_recipe)  
  
set.seed(123)  
#fit model to the training set  
ames\_rf\_fit = fit(ames\_rf\_wflow, train)

Now that the random tree is set, I want to see how it performs on the testing and training sets.

trainpredrf = predict(ames\_rf\_fit, train)  
head(trainpredrf)

# A tibble: 6 × 1  
 .pred\_class  
 <fct>   
1 No   
2 No   
3 No   
4 No   
5 No   
6 No

confusionMatrix(trainpredrf$.pred\_class, train$Above\_Median,   
 positive = "Yes")

Confusion Matrix and Statistics  
  
 Reference  
Prediction No Yes  
 No 695 8  
 Yes 12 722  
   
 Accuracy : 0.9861   
 95% CI : (0.9786, 0.9915)  
 No Information Rate : 0.508   
 P-Value [Acc > NIR] : <2e-16   
   
 Kappa : 0.9722   
   
 Mcnemar's Test P-Value : 0.5023   
   
 Sensitivity : 0.9890   
 Specificity : 0.9830   
 Pos Pred Value : 0.9837   
 Neg Pred Value : 0.9886   
 Prevalence : 0.5080   
 Detection Rate : 0.5024   
 Detection Prevalence : 0.5108   
 Balanced Accuracy : 0.9860   
   
 'Positive' Class : Yes

On the training set, it has an accuracy of 0.9861, sensitivity of 0.9890, and specificity of 0.9830.

testpredrf = predict(ames\_rf\_fit, test)  
head(testpredrf)

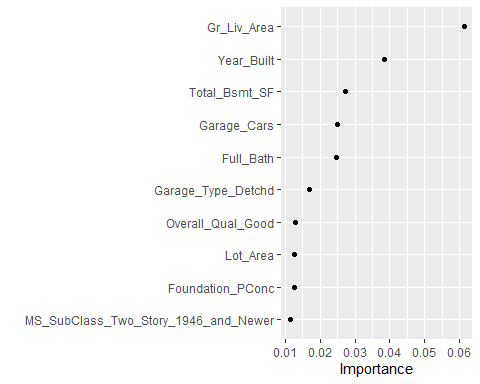
# A tibble: 6 × 1  
 .pred\_class  
 <fct>   
1 No   
2 Yes   
3 Yes   
4 Yes   
5 Yes   
6 Yes

confusionMatrix(testpredrf$.pred\_class, test$Above\_Median,   
 positive = "Yes")

Confusion Matrix and Statistics  
  
 Reference  
Prediction No Yes  
 No 284 37  
 Yes 19 276  
   
 Accuracy : 0.9091   
 95% CI : (0.8836, 0.9306)  
 No Information Rate : 0.5081   
 P-Value [Acc > NIR] : <2e-16   
   
 Kappa : 0.8183   
   
 Mcnemar's Test P-Value : 0.0231   
   
 Sensitivity : 0.8818   
 Specificity : 0.9373   
 Pos Pred Value : 0.9356   
 Neg Pred Value : 0.8847   
 Prevalence : 0.5081   
 Detection Rate : 0.4481   
 Detection Prevalence : 0.4789   
 Balanced Accuracy : 0.9095   
   
 'Positive' Class : Yes

On the testing set, it has an accuracy of 0.9091, sensitivity of 0.8818, and specificity of 0.9373.

saveRDS(ames\_rf\_fit, "ames\_rf\_fit.rds")  
ames\_rf\_fit = readRDS("ames\_rf\_fit.rds")  
ames\_rf\_fit %>% pull\_workflow\_fit() %>% vip(geom = "point")



Our chart of variable importance was informative. It should be noted that this chart does not discriminate between above or below median home value sale.

## Conclusions

Overall, these four different approaches yielded acceptable models. We can compare their performances in this table. The top performers on the testing set are in bold.

|  | Logistic Regression | Logistic Regression with Lasso | Classification/Decision Tree | Random Forests |
| --- | --- | --- | --- | --- |
| Accuracy | Train: 0.946112  Test: 0.8961039 | Train: 0.93528118  **Test: 0.913961** | Train: 0.9068  Test: 0.8847 | Train: 0.9861  Test: 0.9091 |
| Sensitivity | Train: 0.9383562  Test: 0.9105431 | Train: 0.9315068  **Test: 0.9329073** | Train: 0.9041  Test: 0.8722 | Train: 0.9890  Test: 0.8818 |
| Specificity | Train: 0.9561528  Test: 0.8844884 | Train: 0.9405941  Test: 0.8943894 | Train: 0.9095  Test: 0.8977 | Train: 0.9830  **Test: 0.9373** |
| AUC | Train: 0.9895119  Test: 0.9512121 | Train: 0.9853888  **Test: 0.9695378** | n/a | n/a |

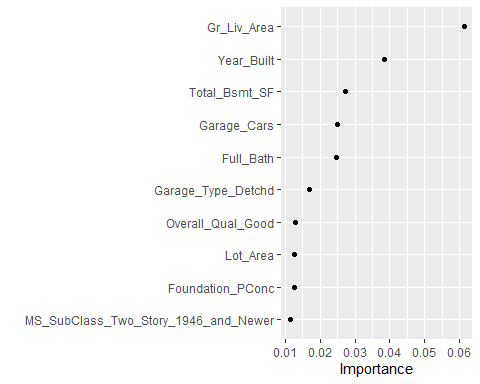
The logistic lasso regression had the best performance on the testing set in accuracy, sensitivity, and AUC. The random forest had the best performance on the testing set in specificity.

For ease of reference, I extracted the coefficient data from the logistic lasso regression and plugged them into Excel. I removed the excluded variables and ordered by coefficient strength both positively and negatively. Recall the strength of a coefficient is determined by how close its absolute value is to 1.

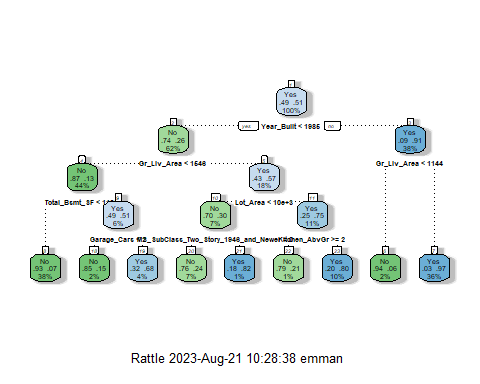
|  |  |
| --- | --- |
| **Variable** | **Coefficent** |
| Gr\_Liv\_Area | 1.332043 |
| Overall\_Qual\_Very\_Good | 0.935834 |
| Garage\_Cars | 0.66009 |
| Total\_Bsmt\_SF | 0.633479 |
| Full\_Bath | 0.476946 |
| Overall\_Qual\_Good | 0.440551 |
| Foundation\_PConc | 0.437148 |
| MS\_Zoning\_Floating\_Village\_Residential | 0.416718 |
| (Intercept) | 0.328013 |
| Neighborhood\_Crawford | 0.311684 |
| Year\_Built | 0.263913 |
| Neighborhood\_Clear\_Creek | 0.235276 |
| Neighborhood\_Timberland | 0.224135 |
| Sale\_Condition\_Alloca | 0.215948 |
| Kitchen\_Qual\_Excellent | 0.182781 |
| Kitchen\_Qual\_Good | 0.170933 |
| MS\_SubClass\_Split\_or\_Multilevel | 0.14732 |
| Half\_Bath | 0.122338 |
| MS\_SubClass\_Two\_Story\_1946\_and\_Newer | 0.115926 |
| Neighborhood\_Gilbert | 0.112638 |
| Neighborhood\_Northwest\_Ames | 0.098113 |
| Neighborhood\_Mitchell | 0.080147 |
| Exter\_Cond\_other | 0.079037 |
| Misc\_Val | 0.077807 |
| House\_Style\_SFoyer | 0.07774 |
| Lot\_Area | 0.076459 |
| House\_Style\_Two\_and\_Half\_Unf | 0.064444 |
| Garage\_Type\_BuiltIn | 0.04681 |
| Neighborhood\_Brookside | 0.039986 |
| Garage\_Type\_Basment | 0.037168 |
| Neighborhood\_Iowa\_DOT\_and\_Rail\_Road | 0.032858 |
| MS\_SubClass\_One\_Story\_PUD\_1946\_and\_Newer | 0.013008 |
| Exter\_Cond\_Good | 0.006206 |
| Year\_Sold | -0.01111 |
| Foundation\_Wood | -0.01618 |
| Neighborhood\_Sawyer\_West | -0.01686 |
| Foundation\_Slab | -0.02186 |
| Lot\_Shape\_Regular | -0.08793 |
| Kitchen\_Qual\_other | -0.12375 |
| Exter\_Cond\_Fair | -0.14993 |
| Central\_Air\_N | -0.15413 |
| MS\_Zoning\_Residential\_Medium\_Density | -0.17407 |
| Neighborhood\_Old\_Town | -0.19252 |
| Overall\_Qual\_Fair | -0.21114 |
| Garage\_Type\_CarPort | -0.21228 |
| Sale\_Condition\_Family | -0.22482 |
| Overall\_Qual\_Average | -0.25063 |
| Sale\_Condition\_Abnorml | -0.29358 |
| MS\_SubClass\_other | -0.32032 |
| Garage\_Type\_Detchd | -0.33815 |
| MS\_SubClass\_Two\_Story\_PUD\_1946\_and\_Newer | -0.43284 |
| Overall\_Qual\_Below\_Average | -0.49515 |
| Kitchen\_AbvGr | -0.77021 |

Let’s also revisit our random forests variable importance and decision tree.

saveRDS(ames\_rf\_fit, "ames\_rf\_fit.rds")  
ames\_rf\_fit = readRDS("ames\_rf\_fit.rds")  
ames\_rf\_fit %>% pull\_workflow\_fit() %>% vip(geom = "point")



fancyRpartPlot(tree2, tweak = 1.25)



Considering these three models, for home investors in Ames, Iowa, I have the following recommendations:

Prioritize investments in homes with the following features and characteristics:

* Larger ground living area
* Newer builds
* Large basements
* Large/more-car garages
* More full baths
* Good overall quality
* Concrete foundation
* Floating Village Residential-zoned homes
* Homes in the Crawford Neighborhood

Avoid investing in homes with the following features:

* More than 1 kitchen
* Detached or carport garages
* Fair, average, or below average overall quality
* Two story planned unit development homes
* Family or abnormal sale conditions