### Kwangho Kim

### BAN 502

### Project 1

#install.packages("glmnet")  
#install.packages("corrplot")  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✔ ggplot2 3.3.6 ✔ purrr 0.3.4  
## ✔ tibble 3.1.7 ✔ dplyr 1.0.9  
## ✔ tidyr 1.2.0 ✔ stringr 1.4.0  
## ✔ readr 2.1.2 ✔ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(tidymodels)

## ── Attaching packages ────────────────────────────────────── tidymodels 0.2.0 ──

## ✔ broom 0.8.0 ✔ rsample 0.1.1  
## ✔ dials 0.1.1 ✔ tune 0.2.0  
## ✔ infer 1.0.0 ✔ workflows 0.2.6  
## ✔ modeldata 0.1.1 ✔ workflowsets 0.2.1  
## ✔ parsnip 0.2.1 ✔ yardstick 0.0.9  
## ✔ recipes 0.2.0

## ── 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 4.1-4

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

#install.packages("ggcorrplot")  
library(ggcorrplot)  
#install.packages("GGally")  
library(GGally)

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

library(plyr)

## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.  
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:  
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##   
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

## The following object is masked from 'package:purrr':  
##   
## compact

library(e1071) #often needed for various statistical tasks

##   
## 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) #for threshold selction  
#install.packages("vip")  
library(vip)

##   
## Attaching package: 'vip'

## The following object is masked from 'package:utils':  
##   
## vi

#tree visualization  
library(rpart) #for classification trees

##   
## Attaching package: 'rpart'

## The following object is masked from 'package:dials':  
##   
## prune

library(rpart.plot) #for plotting trees  
library(RColorBrewer) #better visualization of classification trees  
library(rattle) #better visualization of classification trees

## 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.

# random forest  
#install.packages("rattle")  
library(ranger) #for random forests

##   
## Attaching package: 'ranger'

## The following object is masked from 'package:rattle':  
##   
## importance

#install.packages("randomForest")  
library(randomForest) #also for random forests

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ranger':  
##   
## importance

## The following object is masked from 'package:rattle':  
##   
## importance

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

# XGBoost  
#install.packages("xgboost")  
library(xgboost)

##   
## Attaching package: 'xgboost'

## The following object is masked from 'package:rattle':  
##   
## xgboost

## The following object is masked from 'package:dplyr':  
##   
## slice

#install.packages("usemodels")  
library(usemodels) #new package :)

Read in Ames, Iowa dataset

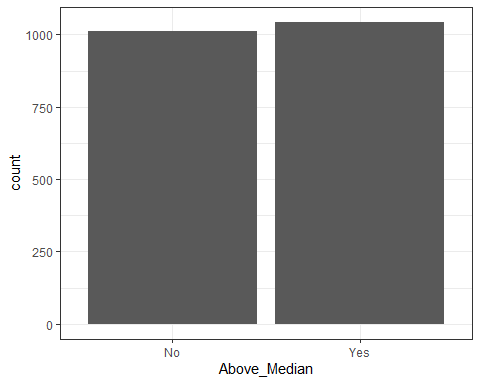
library(readr)  
ames\_student <- read\_csv("ames\_student.csv")

## 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.

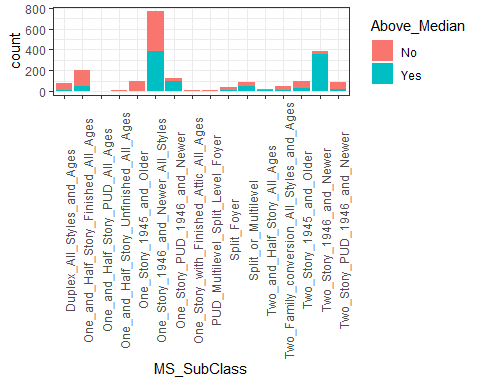
#View(ames\_student)  
#str(ames\_student)  
#summary(ames\_student)  
ames1 <- ames\_student  
# ames1 <- ames1 %>% select("Above\_Median","Overall\_Qual","Gr\_Liv\_Area","Garage\_Cars","Garage\_Area","Total\_Bsmt\_SF","First\_Flr\_SF","Full\_Bath","Year\_Built","Year\_Remod\_Add","Neighborhood")  
ames1 <- ames1 %>%  
 mutate(Neighborhood = as\_factor(Neighborhood)) %>%  
 mutate(Above\_Median = factor(Above\_Median)) %>%  
 mutate(Overall\_Qual = factor(Overall\_Qual,levels=c("Very\_Poor","Poor","Fair","Below\_Average","Average","Above\_Average","Good","Very\_Good" ,"Excellent", "Very\_Excellent")))   
  
#summary(ames2)  
#str(ames2)

visualize

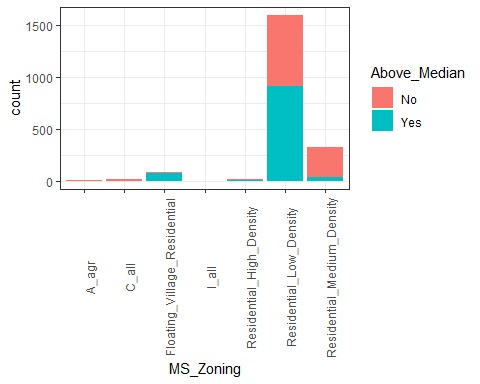
#ggplot(ames1, aes(x=Pclass, fill = Above\_Median)) + geom\_bar() + theme\_bw() + theme(axis.text.x = element\_text(angle=90))  
# ggplot(ames1, aes(x=Above\_Median, y= Lot\_Frontage)) + geom\_boxplot() + theme\_bw()  
  
#unselect predictor  
  
ggplot(ames1, aes(x=Above\_Median)) + geom\_bar() + theme\_bw()



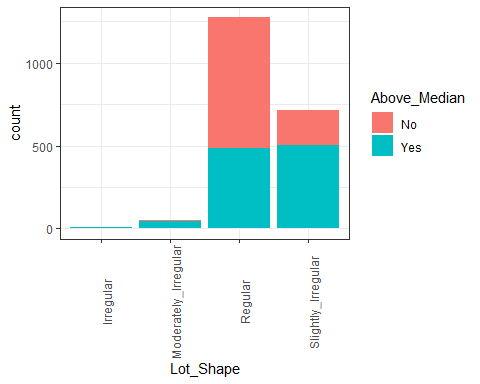
ggplot(ames1, aes(x=MS\_SubClass, fill = Above\_Median)) + geom\_bar() + theme\_bw() + theme(axis.text.x = element\_text(angle=90))



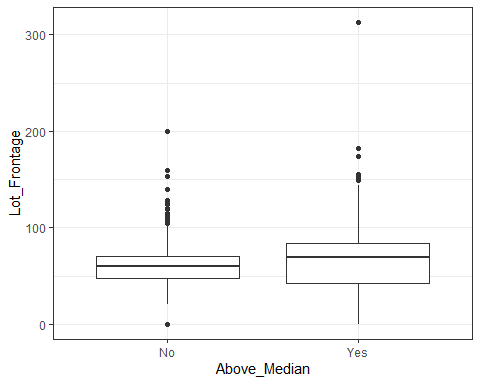
ggplot(ames1, aes(x=MS\_Zoning, fill = Above\_Median)) + geom\_bar() + theme\_bw() + theme(axis.text.x = element\_text(angle=90))



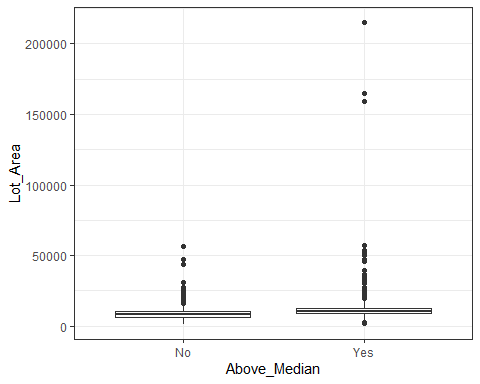
ggplot(ames1, aes(x=Lot\_Shape, fill = Above\_Median)) + geom\_bar() + theme\_bw() + theme(axis.text.x = element\_text(angle=90))



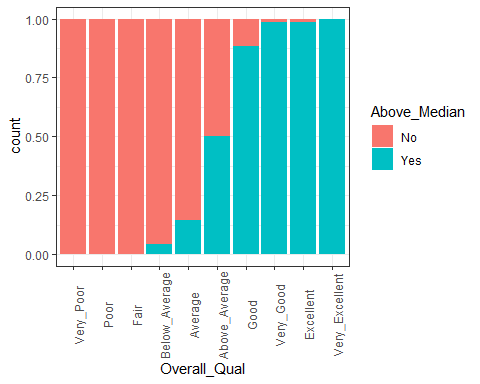
ggplot(ames1, aes(x=Above\_Median, y=Lot\_Frontage)) + geom\_boxplot() + theme\_bw()



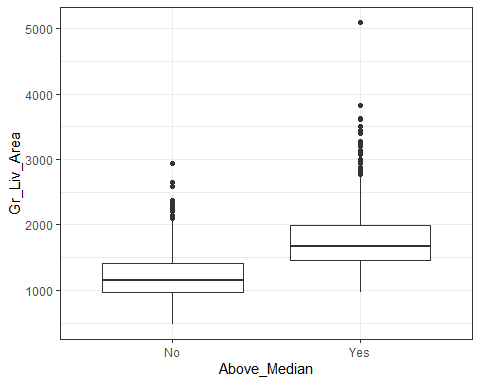
ggplot(ames1, aes(x=Above\_Median, y=Lot\_Area)) + geom\_boxplot() + theme\_bw()



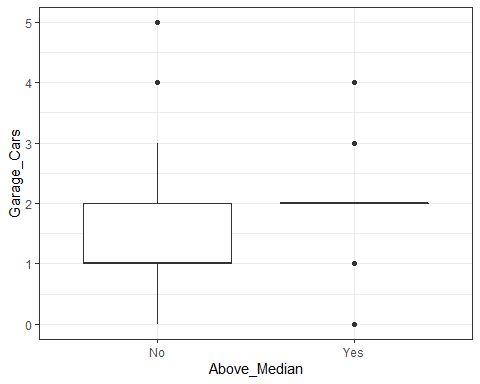
#selected predictors  
  
ggplot(ames1, aes(fill=Above\_Median, x=Overall\_Qual)) + geom\_bar(position="fill") + theme\_bw()+ theme(axis.text.x = element\_text(angle=90))



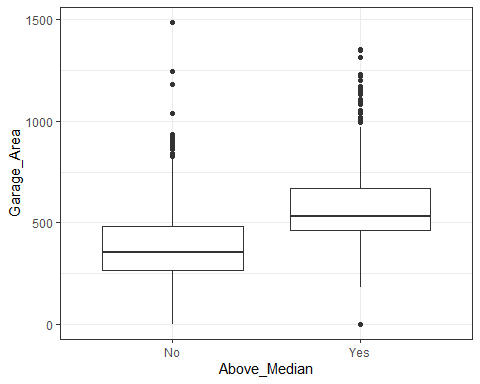
ggplot(ames1, aes(x=Above\_Median, y=Gr\_Liv\_Area)) + geom\_boxplot() + theme\_bw()



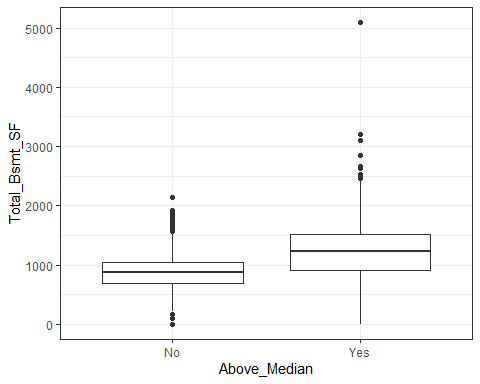
ggplot(ames1, aes(x=Above\_Median, y=Garage\_Cars)) + geom\_boxplot() + theme\_bw()



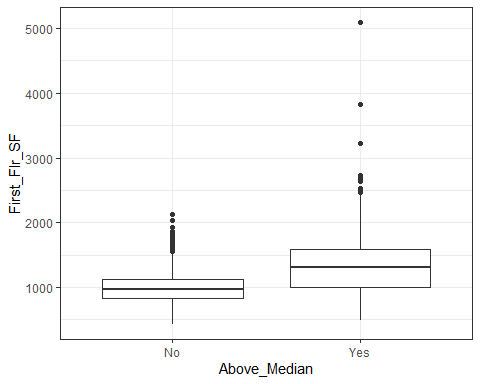
ggplot(ames1, aes(x=Above\_Median, y=Garage\_Area)) + geom\_boxplot() + theme\_bw()



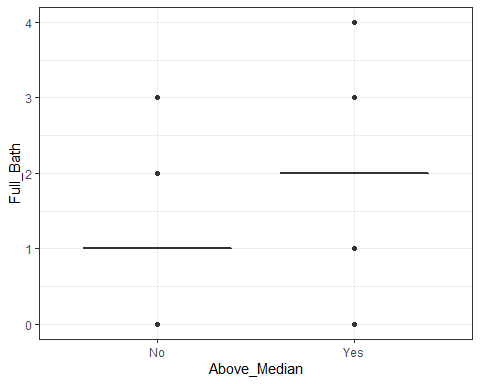
ggplot(ames1, aes(x=Above\_Median, y=Total\_Bsmt\_SF)) + geom\_boxplot() + theme\_bw()



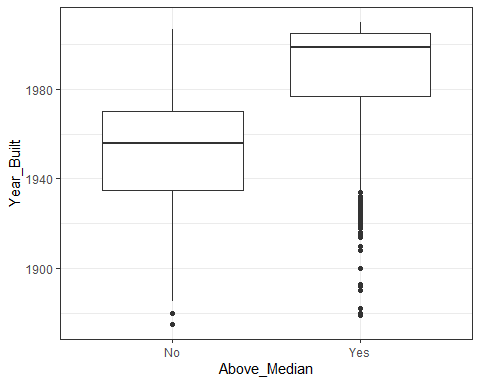
ggplot(ames1, aes(x=Above\_Median, y=First\_Flr\_SF)) + geom\_boxplot() + theme\_bw()



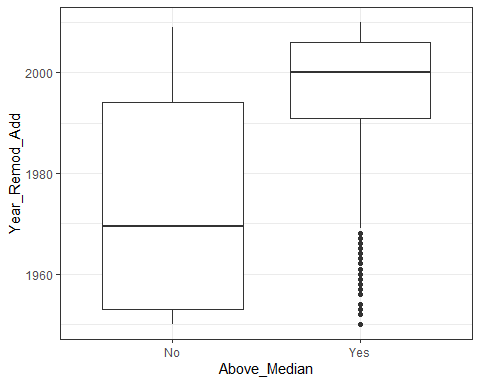
ggplot(ames1, aes(x=Above\_Median, y=Full\_Bath)) + geom\_boxplot() + theme\_bw()



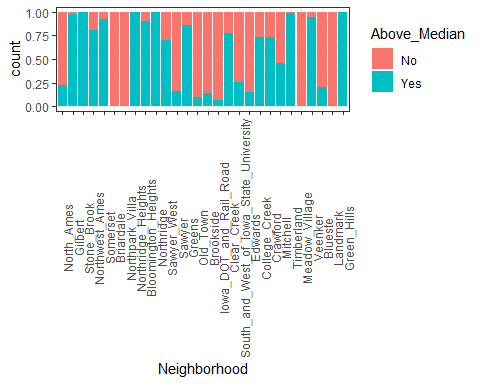
ggplot(ames1, aes(x=Above\_Median, y=Year\_Built)) + geom\_boxplot() + theme\_bw()



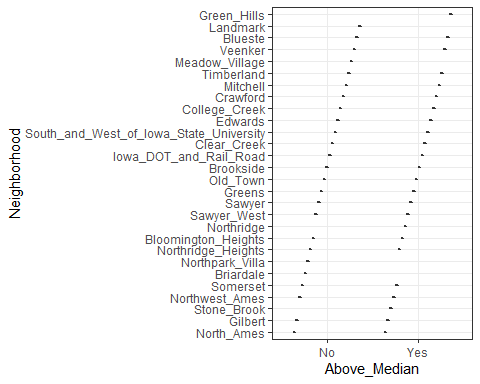
ggplot(ames1, aes(x=Above\_Median, y=Year\_Remod\_Add)) + geom\_boxplot() + theme\_bw()



ggplot(ames1, aes(fill=Above\_Median, x=Neighborhood)) + geom\_bar(position="fill") + theme\_bw()+ theme(axis.text.x = element\_text(angle=90))

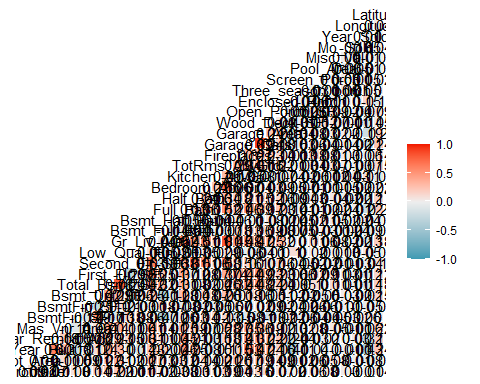


ggplot(ames1, aes(x=Above\_Median, y=Neighborhood)) + geom\_boxplot() + theme\_bw()



#use "ggcorr" to create a correlation matrix with labels and correlation reported to two decimals  
ggcorr(ames1, label = "TRUE", label\_round = 2)

## Warning in ggcorr(ames1, label = "TRUE", label\_round = 2): 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

 select predictors

ames1 <- ames1 %>% select("Above\_Median","Overall\_Qual","Gr\_Liv\_Area","Garage\_Cars","Garage\_Area","Total\_Bsmt\_SF","First\_Flr\_SF","Full\_Bath","Year\_Built","Year\_Remod\_Add","Neighborhood")

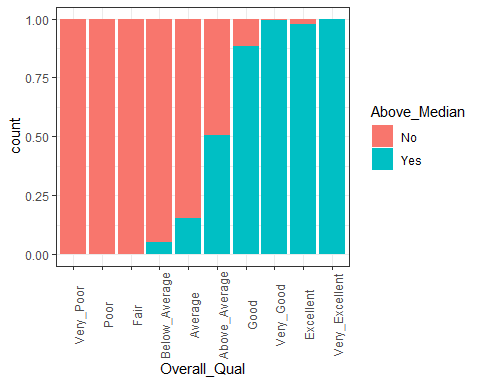
split the data

set.seed(123)   
ames\_split = initial\_split(ames1, prob = 0.80, strata = Above\_Median)  
train = training(ames\_split)  
test = testing(ames\_split)  
  
summary(train)

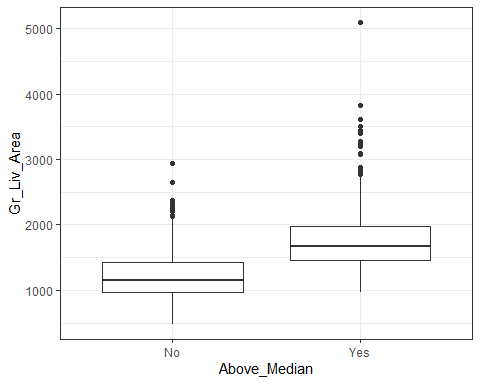
## Above\_Median Overall\_Qual Gr\_Liv\_Area Garage\_Cars Garage\_Area   
## No :757 Average :431 Min. : 480 Min. :0.000 Min. : 0   
## Yes:782 Above\_Average:394 1st Qu.:1142 1st Qu.:1.000 1st Qu.: 326   
## Good :307 Median :1452 Median :2.000 Median : 478   
## Very\_Good :180 Mean :1498 Mean :1.776 Mean : 473   
## Below\_Average:134 3rd Qu.:1733 3rd Qu.:2.000 3rd Qu.: 576   
## Excellent : 44 Max. :5095 Max. :5.000 Max. :1488   
## (Other) : 49   
## Total\_Bsmt\_SF First\_Flr\_SF Full\_Bath Year\_Built Year\_Remod\_Add  
## Min. : 0 Min. : 432.0 Min. :0.000 Min. :1875 Min. :1950   
## 1st Qu.: 786 1st Qu.: 879.5 1st Qu.:1.000 1st Qu.:1951 1st Qu.:1964   
## Median : 984 Median :1080.0 Median :2.000 Median :1971 Median :1992   
## Mean :1050 Mean :1164.0 Mean :1.569 Mean :1970 Mean :1984   
## 3rd Qu.:1304 3rd Qu.:1402.5 3rd Qu.:2.000 3rd Qu.:2000 3rd Qu.:2004   
## Max. :5095 Max. :5095.0 Max. :4.000 Max. :2010 Max. :2010   
##   
## Neighborhood  
## North\_Ames :241   
## Old\_Town :147   
## College\_Creek:130   
## Edwards : 98   
## Somerset : 92   
## Gilbert : 79   
## (Other) :752

visualized train data set

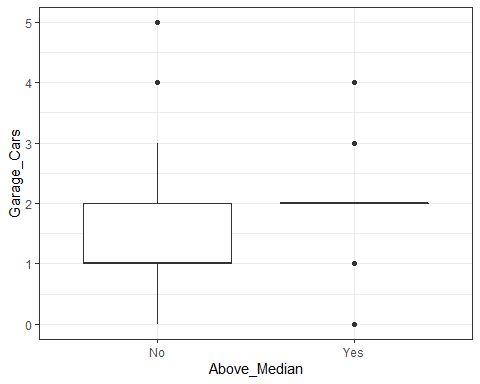
#selected predictors  
  
ggplot(train, aes(fill=Above\_Median, x=Overall\_Qual)) + geom\_bar(position="fill") + theme\_bw()+ theme(axis.text.x = element\_text(angle=90))



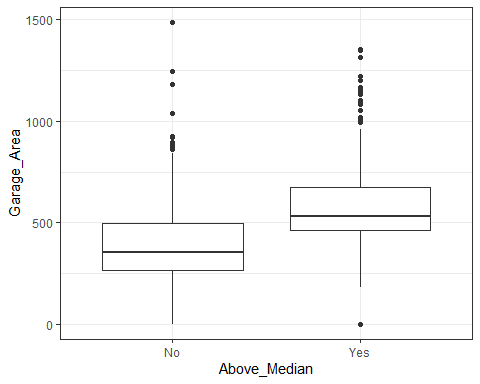
ggplot(train, aes(x=Above\_Median, y=Gr\_Liv\_Area)) + geom\_boxplot() + theme\_bw()



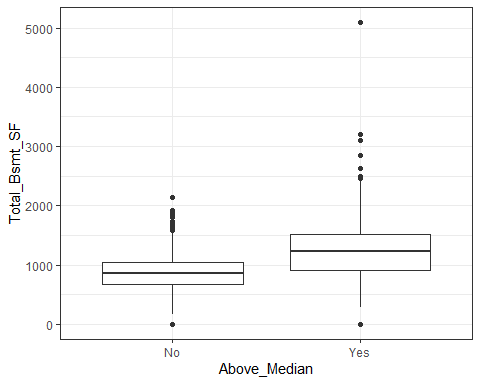
ggplot(train, aes(x=Above\_Median, y=Garage\_Cars)) + geom\_boxplot() + theme\_bw()



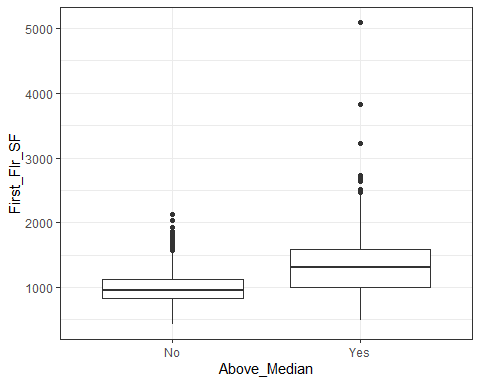
ggplot(train, aes(x=Above\_Median, y=Garage\_Area)) + geom\_boxplot() + theme\_bw()



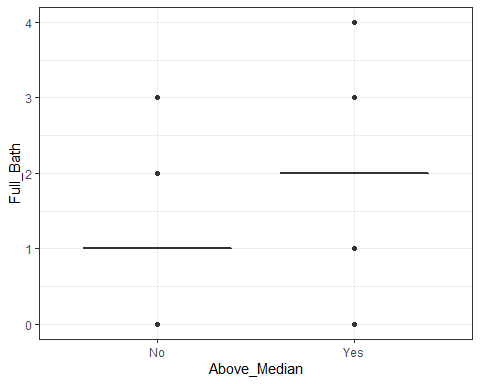
ggplot(train, aes(x=Above\_Median, y=Total\_Bsmt\_SF)) + geom\_boxplot() + theme\_bw()



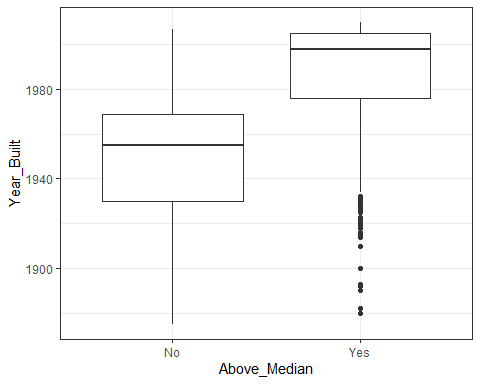
ggplot(train, aes(x=Above\_Median, y=First\_Flr\_SF)) + geom\_boxplot() + theme\_bw()



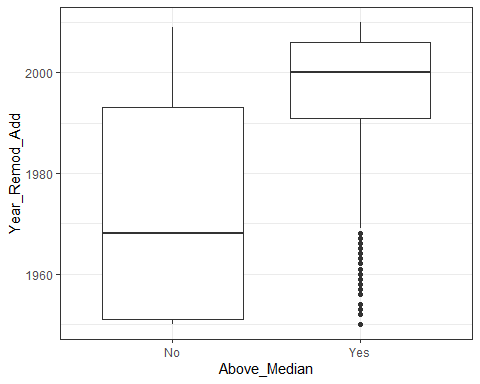
ggplot(train, aes(x=Above\_Median, y=Full\_Bath)) + geom\_boxplot() + theme\_bw()



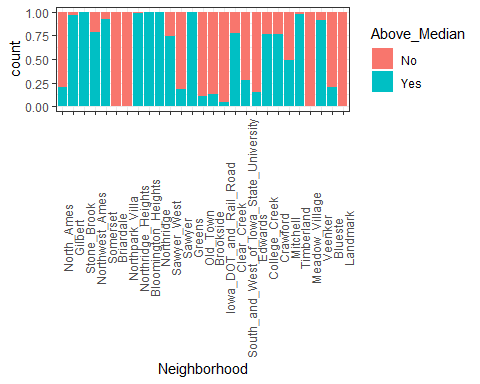
ggplot(train, aes(x=Above\_Median, y=Year\_Built)) + geom\_boxplot() + theme\_bw()



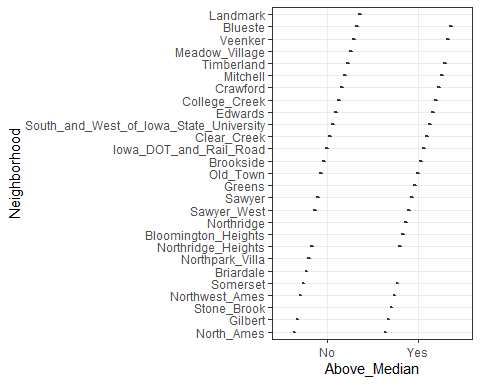
ggplot(train, aes(x=Above\_Median, y=Year\_Remod\_Add)) + geom\_boxplot() + theme\_bw()



ggplot(train, aes(fill=Above\_Median, x=Neighborhood)) + geom\_bar(position="fill") + theme\_bw()+ theme(axis.text.x = element\_text(angle=90))

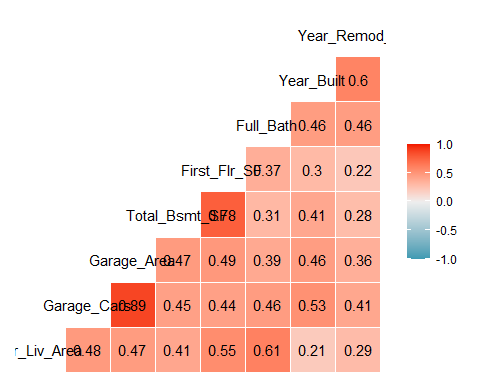


ggplot(train, aes(x=Above\_Median, y=Neighborhood)) + geom\_boxplot() + theme\_bw()



#use "ggcorr" to create a correlation matrix with labels and correlation reported to two decimals  
ggcorr(train, label = "TRUE", label\_round = 2)

## Warning in ggcorr(train, label = "TRUE", label\_round = 2): data in column(s)  
## 'Above\_Median', 'Overall\_Qual', 'Neighborhood' are not numeric and were ignored



#1 Logit Model

ames\_model =   
 logistic\_reg(mode = "classification") %>% #note the use of logistic\_reg and mode ="classification"  
 set\_engine("glm") #standard logistic regression engine is glm  
  
ames\_recipe = recipe(Above\_Median ~ ., train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
logreg\_wf = workflow() %>%  
 add\_recipe(ames\_recipe) %>%   
 add\_model(ames\_model)  
  
ames\_fit = fit(logreg\_wf, train)

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

Develop predictions on train

predictions = predict(ames\_fit, train, type="prob") #develop predicted probabilities

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

head(predictions)

## # A tibble: 6 × 2  
## .pred\_No .pred\_Yes  
## <dbl> <dbl>  
## 1 0.879 0.121   
## 2 0.975 0.0248   
## 3 0.987 0.0131   
## 4 1.00 0.00000000183  
## 5 1.00 0.00000000271  
## 6 0.863 0.137

Let’s extract just the “Yes” prediction.

predictions = predict(ames\_fit, train, type="prob")[2]

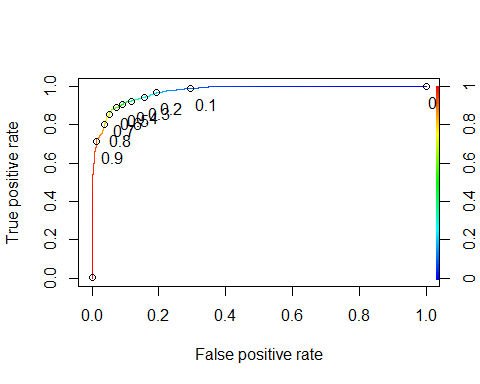
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

head(predictions)

## # A tibble: 6 × 1  
## .pred\_Yes  
## <dbl>  
## 1 0.121   
## 2 0.0248   
## 3 0.0131   
## 4 0.00000000183  
## 5 0.00000000271  
## 6 0.137

Threshold selection

#Change this next line to the names of your predictions and the response variable in the training data frame  
ROCRpred = prediction(predictions, train$Above\_Median)   
  
###You shouldn't need to ever change the next two lines:  
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))

 Area under the curve (AUC). AUC is a measure of the strength of the model. Values closer to 1 are better. Can be used to compare models.

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

## [1] 0.972367

#Determine threshold to balance sensitivity and specificity  
#DO NOT modify this code  
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 0.9143223  
## specificity 0.9048877  
## cutoff 0.4845588

Test thresholds to evaluate accuracy

#confusion matrix  
#The "No" and "Yes" represent the actual values  
#The "FALSE" and "TRUE" represent our predicted values  
t1 = table(train$Above\_Median,predictions > 0.4845588)  
t1

##   
## FALSE TRUE  
## No 685 72  
## Yes 68 714

Calculate accuracy

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

## [1] 0.9090318

Can apply trial and error to maximize accuracy (here trying 0.5 as threshold)

t1 = table(train$Above\_Median,predictions > 0.5)  
t1

##   
## FALSE TRUE  
## No 687 70  
## Yes 73 709

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

## [1] 0.9070825

Threshold = 0.6

t1 = table(train$Above\_Median,predictions > 0.6)  
t1

##   
## FALSE TRUE  
## No 703 54  
## Yes 87 695

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

## [1] 0.9083821

set.seed(123)  
cv\_model <- train(  
 Above\_Median ~.,  
 data = train,  
 method = "glm",  
 family = "binomial"  
)

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

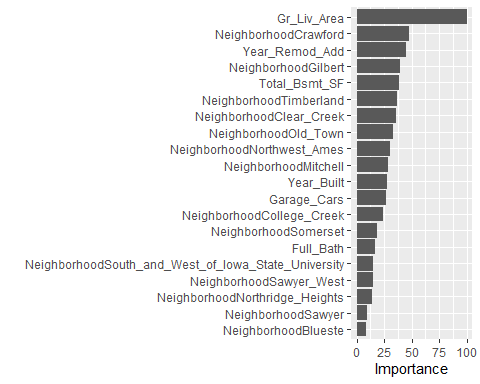
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

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

vip(cv\_model, num\_features = 20, method = "model")



prediction on Test

Develop predictions on train

predictions = predict(ames\_fit, test, type="prob") #develop predicted probabilities

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

head(predictions)

## # A tibble: 6 × 2  
## .pred\_No .pred\_Yes  
## <dbl> <dbl>  
## 1 9.92e- 1 0.00838  
## 2 1.73e- 2 0.983   
## 3 3.90e-10 1.00   
## 4 1.32e- 9 1.00   
## 5 6.88e- 9 1.00   
## 6 6.67e- 2 0.933

Let’s extract just the “Yes” prediction.

predictions = predict(ames\_fit, test, type="prob")[2]

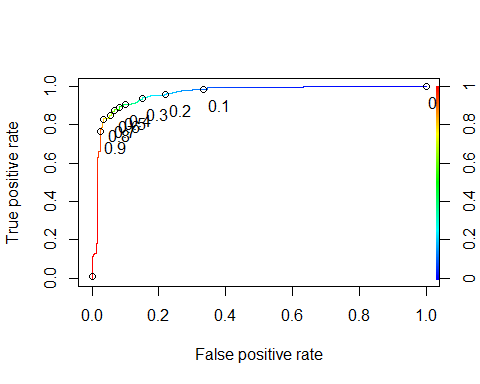
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

head(predictions)

## # A tibble: 6 × 1  
## .pred\_Yes  
## <dbl>  
## 1 0.00838  
## 2 0.983   
## 3 1.00   
## 4 1.00   
## 5 1.00   
## 6 0.933

Threshold selection

#Change this next line to the names of your predictions and the response variable in the training data frame  
ROCRpred = prediction(predictions, test$Above\_Median)   
  
###You shouldn't need to ever change the next two lines:  
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))

 Area under the curve (AUC). AUC is a measure of the strength of the model. Values closer to 1 are better. Can be used to compare models.

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

## [1] 0.9600351

#Determine threshold to balance sensitivity and specificity  
#DO NOT modify this code  
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 0.9080460  
## specificity 0.9051383  
## cutoff 0.4247532

Test thresholds to evaluate accuracy

#confusion matrix  
#The "No" and "Yes" represent the actual values  
#The "FALSE" and "TRUE" represent our predicted values  
t1 = table(test$Above\_Median,predictions > 0.424752)  
t1

##   
## FALSE TRUE  
## No 229 24  
## Yes 24 237

Calculate accuracy

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

## [1] 0.9066148

#2 Create decision tree

ames2\_recipe = recipe(Above\_Median ~., train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification") #notice different mode here for a regression tree  
  
ames2\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(ames2\_recipe)  
  
ames2\_fit = fit(ames2\_wflow, train)

Plot the tree

ames2\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

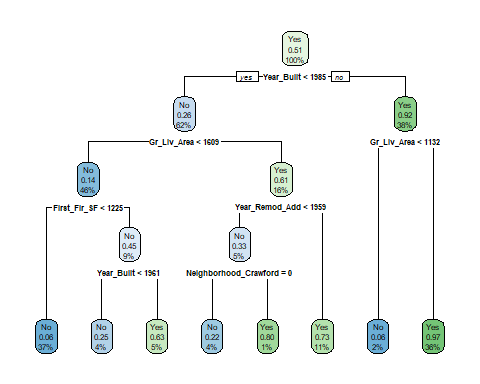
## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## Please use `extract\_fit\_parsnip()` instead.

## n= 1539   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 1539 757 Yes (0.49187784 0.50812216)   
## 2) Year\_Built< 1984.5 949 242 No (0.74499473 0.25500527)   
## 4) Gr\_Liv\_Area< 1609 708 96 No (0.86440678 0.13559322)   
## 8) First\_Flr\_SF< 1225 568 33 No (0.94190141 0.05809859) \*  
## 9) First\_Flr\_SF>=1225 140 63 No (0.55000000 0.45000000)   
## 18) Year\_Built< 1960.5 67 17 No (0.74626866 0.25373134) \*  
## 19) Year\_Built>=1960.5 73 27 Yes (0.36986301 0.63013699) \*  
## 5) Gr\_Liv\_Area>=1609 241 95 Yes (0.39419087 0.60580913)   
## 10) Year\_Remod\_Add< 1958.5 75 25 No (0.66666667 0.33333333)   
## 20) Neighborhood\_Crawford< 0.5 60 13 No (0.78333333 0.21666667) \*  
## 21) Neighborhood\_Crawford>=0.5 15 3 Yes (0.20000000 0.80000000) \*  
## 11) Year\_Remod\_Add>=1958.5 166 45 Yes (0.27108434 0.72891566) \*  
## 3) Year\_Built>=1984.5 590 50 Yes (0.08474576 0.91525424)   
## 6) Gr\_Liv\_Area< 1131.5 33 2 No (0.93939394 0.06060606) \*  
## 7) Gr\_Liv\_Area>=1131.5 557 19 Yes (0.03411131 0.96588869) \*

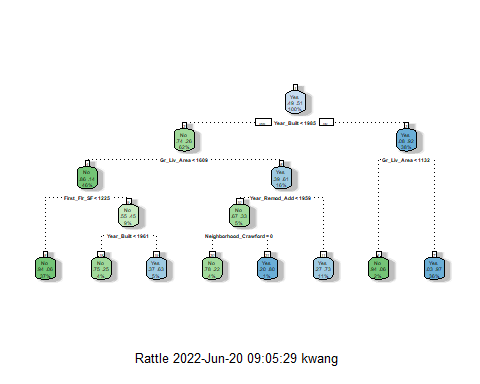
tree = ames2\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

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

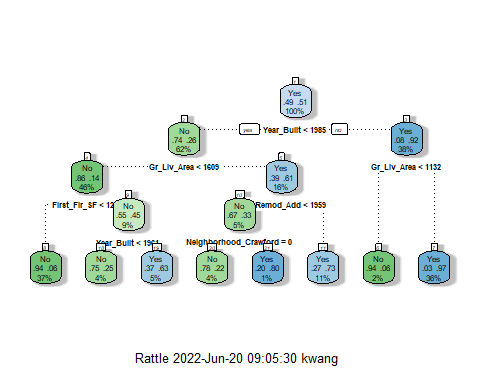
rpart.plot(tree)



fancyRpartPlot(tree)



fancyRpartPlot(tree, tweak=1.5)



ames2\_fit$fit$fit$fitcptable

## NULL

Develop predictions on the training set

train\_preds = predict(ames2\_fit, train, type = "class")  
head(train\_preds) #see first six predictions to verify that predictions are in correct form

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

confusionMatrix(train\_preds$.pred\_class,train$Above\_Median,positive="Yes") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 663 65  
## Yes 94 717  
##   
## Accuracy : 0.8967   
## 95% CI : (0.8804, 0.9114)  
## No Information Rate : 0.5081   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.7932   
##   
## Mcnemar's Test P-Value : 0.02638   
##   
## Sensitivity : 0.9169   
## Specificity : 0.8758   
## Pos Pred Value : 0.8841   
## Neg Pred Value : 0.9107   
## Prevalence : 0.5081   
## Detection Rate : 0.4659   
## Detection Prevalence : 0.5270   
## Balanced Accuracy : 0.8964   
##   
## 'Positive' Class : Yes   
##

Develop predictions on the testing set

test\_preds = predict(ames2\_fit, test, type = "class")  
head(test\_preds) #verify that predictions are in correct form

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

Examine performance metrics on the testing set.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 212 29  
## Yes 41 232  
##   
## Accuracy : 0.8638   
## 95% CI : (0.8311, 0.8923)  
## No Information Rate : 0.5078   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7274   
##   
## Mcnemar's Test P-Value : 0.1886   
##   
## Sensitivity : 0.8889   
## Specificity : 0.8379   
## Pos Pred Value : 0.8498   
## Neg Pred Value : 0.8797   
## Prevalence : 0.5078   
## Detection Rate : 0.4514   
## Detection Prevalence : 0.5311   
## Balanced Accuracy : 0.8634   
##   
## 'Positive' Class : Yes   
##

#3 random forest

ames3\_recipe = recipe(Above\_Median ~., train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
rf\_model = rand\_forest() %>%   
 set\_engine("ranger") %>%   
 set\_mode("classification")  
  
ames3\_wflow =   
 workflow() %>%   
 add\_model(rf\_model) %>%   
 add\_recipe(ames3\_recipe)  
  
set.seed(123)  
ames3\_fit = fit(ames3\_wflow, train)

Check out random forest details

ames3\_fit

## ══ Workflow [trained] ══════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: rand\_forest()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 1 Recipe Step  
##   
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Ranger result  
##   
## Call:  
## ranger::ranger(x = maybe\_data\_frame(x), y = y, num.threads = 1, verbose = FALSE, seed = sample.int(10^5, 1), probability = TRUE)   
##   
## Type: Probability estimation   
## Number of trees: 500   
## Sample size: 1539   
## Number of independent variables: 44   
## Mtry: 6   
## Target node size: 10   
## Variable importance mode: none   
## Splitrule: gini   
## OOB prediction error (Brier s.): 0.06906285

Predictions on train

predRF = predict(ames3\_fit, train)  
head(predRF)

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

Confusion matrix

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 733 25  
## Yes 24 757  
##   
## Accuracy : 0.9682   
## 95% CI : (0.9581, 0.9764)  
## No Information Rate : 0.5081   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9363   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9680   
## Specificity : 0.9683   
## Pos Pred Value : 0.9693   
## Neg Pred Value : 0.9670   
## Prevalence : 0.5081   
## Detection Rate : 0.4919   
## Detection Prevalence : 0.5075   
## Balanced Accuracy : 0.9682   
##   
## 'Positive' Class : Yes   
##

Predictions on test

predRF = predict(ames3\_fit, test)  
head(predRF)

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

Confusion matrix

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 236 30  
## Yes 17 231  
##   
## Accuracy : 0.9086   
## 95% CI : (0.8803, 0.932)  
## No Information Rate : 0.5078   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.8172   
##   
## Mcnemar's Test P-Value : 0.08005   
##   
## Sensitivity : 0.8851   
## Specificity : 0.9328   
## Pos Pred Value : 0.9315   
## Neg Pred Value : 0.8872   
## Prevalence : 0.5078   
## Detection Rate : 0.4494   
## Detection Prevalence : 0.4825   
## Balanced Accuracy : 0.9089   
##   
## 'Positive' Class : Yes   
##

#4 xgboost model

#use\_xgboost(Above\_Median ~., train) #comment me out before knitting

set.seed(123)  
folds = vfold\_cv(train, v = 3)

Copy and paste the model from the use\_xgboost function. Modify a few elements.

start\_time = Sys.time() #for timing  
  
xgboost\_recipe <-   
 recipe(formula = Above\_Median ~ ., data = train) %>%   
 #step\_novel(all\_nominal(), -all\_outcomes()) %>%   
 step\_dummy(all\_nominal(), -all\_outcomes(), one\_hot = TRUE)  
# step\_zv(all\_predictors())   
  
xgboost\_spec <-   
 boost\_tree(trees = tune(), min\_n = tune(), tree\_depth = tune(), learn\_rate = tune(),   
 loss\_reduction = tune(), sample\_size = tune()) %>%   
 set\_mode("classification") %>%   
 set\_engine("xgboost")   
  
xgboost\_workflow <-   
 workflow() %>%   
 add\_recipe(xgboost\_recipe) %>%   
 add\_model(xgboost\_spec)   
  
set.seed(77680)  
 xgboost\_tune <-  
 tune\_grid(xgboost\_workflow, resamples = folds, grid = 25)  
  
end\_time = Sys.time()  
end\_time - start\_time

## Time difference of 1.843965 mins

best\_xgb = select\_best(xgboost\_tune, "accuracy")  
  
final\_xgb = finalize\_workflow(  
 xgboost\_workflow,  
 best\_xgb  
)  
  
final\_xgb

## ══ Workflow ════════════════════════════════════════════════════════════════════  
## Preprocessor: Recipe  
## Model: boost\_tree()  
##   
## ── Preprocessor ────────────────────────────────────────────────────────────────  
## 1 Recipe Step  
##   
## • step\_dummy()  
##   
## ── Model ───────────────────────────────────────────────────────────────────────  
## Boosted Tree Model Specification (classification)  
##   
## Main Arguments:  
## trees = 1422  
## min\_n = 4  
## tree\_depth = 3  
## learn\_rate = 0.0257247860908885  
## loss\_reduction = 0.0122502705879233  
## sample\_size = 0.798037836283445  
##   
## Computational engine: xgboost

#fit the finalized workflow to our training data  
final\_xgb\_fit = fit(final\_xgb, train)

trainpredxgb = predict(final\_xgb\_fit, train)  
head(trainpredxgb)

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

Confusion matrix

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 739 18  
## Yes 18 764  
##   
## Accuracy : 0.9766   
## 95% CI : (0.9678, 0.9836)  
## No Information Rate : 0.5081   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9532   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9770   
## Specificity : 0.9762   
## Pos Pred Value : 0.9770   
## Neg Pred Value : 0.9762   
## Prevalence : 0.5081   
## Detection Rate : 0.4964   
## Detection Prevalence : 0.5081   
## Balanced Accuracy : 0.9766   
##   
## 'Positive' Class : Yes   
##

testpredxgb = predict(final\_xgb\_fit, test)

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 232 28  
## Yes 21 233  
##   
## Accuracy : 0.9047   
## 95% CI : (0.8759, 0.9286)  
## No Information Rate : 0.5078   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8094   
##   
## Mcnemar's Test P-Value : 0.3914   
##   
## Sensitivity : 0.8927   
## Specificity : 0.9170   
## Pos Pred Value : 0.9173   
## Neg Pred Value : 0.8923   
## Prevalence : 0.5078   
## Detection Rate : 0.4533   
## Detection Prevalence : 0.4942   
## Balanced Accuracy : 0.9049   
##   
## 'Positive' Class : Yes   
##