Tree-Based Methods

Jessica VanElls

November 29, 2018

The purpose of this document is to explore tree-based methods using the Ames, Iowa housing data set. I will focus on both predictive accuracy and model interpretation.

Data

The following section will deal with data transformation. Here, I used the same methods as in the midterm.

```
train <- read.csv("train.csv", na.strings="placeholder")  # some of the categorical variables ha
ve value "NA" but it doesn't mean null
test <- read.csv("test.csv", na.strings="placeholder")

house <- rbind(train, data.frame(test, SalePrice=rep(1, nrow(test))))
```

Dealing with NA's

Because there were also strings that were "NA" as part of some scales, I noted which columns shouldn't contain the string "NA"", and I change those strings to a true `NA`.

```
## Store all columns that can have "NA" as a valid entry
na_names = c("Alley", "BsmtQual", "BsmtCond", "BsmtExposure", "BsmtFinType1", "BsmtFinType2", "F
ireplaceQu", "GarageType", "GarageQual", "GarageCond", "PoolQC", "Fence", "MiscFeature", "MasVnr
Type")

## Replace "NA" strings with true NA in training data
for (j in 1:ncol(house)) {
  if (sum(colnames(house)[j]==na_names)==0) {  # if the column shouldn't contain "NA"...
    for (i in 1:nrow(house)) {
      if (house[i,j]=="NA") {
        house[i,j] <- NA  # if the column shouldn't contain NA but the cell is "NA", then give i
t a null
      }
    }
  }
}
```

Checking and Changing Data Types

I will be converting scales to factors (e.g., quality) because they describe a condition, not a quantity. There are mixed opinions on how these should be handled, but I am chosing to use the "nominal categorical" method. Years will be treated as integers; months will be treated as factors.

```
## Scales / Factors
house$MSSubClass <- as.factor(house$MSSubClass)
house$MSZoning <- as.factor(house$MSZoning)
house$Street <- as.factor(house$Street)
house$Alley <- as.factor(house$Alley)
house$LotShape <- as.factor(house$LotShape)
house$LandContour <- as.factor(house$LandContour)
house$Utilities <- as.factor(house$Utilities)
house$LotConfig <- as.factor(house$LotConfig)
house$LandSlope <- as.factor(house$LandSlope)
house$Neighborhood <- as.factor(house$Neighborhood)
house$Condition1 <- as.factor(house$Condition1)
house$Condition2 <- as.factor(house$Condition2)
house$BldgType <- as.factor(house$BldgType)
house$HouseStyle <- as.factor(house$HouseStyle)
house$OverallQual <- as.factor(house$OverallQual)
house$OverallCond <- as.factor(house$OverallCond)
house$RoofStyle <- as.factor(house$RoofStyle)
house$RoofMatl <- as.factor(house$RoofMatl)
house$Exterior1st <- as.factor(house$Exterior1st)
house$Exterior2nd <- as.factor(house$Exterior2nd)
house$MasVnrType <- as.factor(house$MasVnrType)
house$ExterQual <- as.factor(house$ExterQual)
house$ExterCond <- as.factor(house$ExterCond)
house$Foundation <- as.factor(house$Foundation)
house$BsmtQual <- as.factor(house$BsmtQual)
house$BsmtCond <- as.factor(house$BsmtCond)
house$BsmtExposure <- as.factor(house$BsmtExposure)
house$BsmtFinType1 <- as.factor(house$BsmtFinType1)
house$BsmtFinType2 <- as.factor(house$BsmtFinType2)
house$Heating <- as.factor(house$Heating)
house$HeatingQC <- as.factor(house$HeatingQC)
house$CentralAir <- as.factor(house$CentralAir)
house$Electrical <- as.factor(house$Electrical)
house$KitchenQual <- as.factor(house$KitchenQual)
house$Functional <- as.factor(house$Functional)
house$FireplaceQu <- as.factor(house$FireplaceQu)
house$GarageType <- as.factor(house$GarageType)
house$GarageFinish <- as.factor(house$GarageFinish)
house$GarageQual <- as.factor(house$GarageQual)
house$GarageCond <- as.factor(house$GarageCond)
house$PavedDrive <- as.factor(house$PavedDrive)
house$PoolQC <- as.factor(house$PoolQC)
house$Fence <- as.factor(house$Fence)
house$MiscFeature <- as.factor(house$MiscFeature)
house$SaleType <- as.factor(house$SaleType)
house$SaleCondition <- as.factor(house$SaleCondition)

## Should be numeric...
house$LotFrontage <- as.integer(house$LotFrontage)
house$LotArea <- as.integer(house$LotArea)
house$MasVnrArea <- as.integer(house$MasVnrArea)
house$BsmtFinSF1 <- as.integer(house$BsmtFinSF1)
```

```
house$BsmtFinSF2 <- as.integer(house$BsmtFinSF2)
house$BsmtUnfSF <- as.integer(house$BsmtUnfSF)
house$TotalBsmtSF <- as.integer(house$TotalBsmtSF)
house$X1stFlrSF <- as.integer(house$X1stFlrSF)
house$X2ndFlrSF <- as.integer(house$X2ndFlrSF)
house$LowQualFinSF <- as.integer(house$LowQualFinSF)
house$GrLivArea <- as.integer(house$GrLivArea)
house$BsmtFullBath <- as.integer(house$BsmtFullBath)
house$BsmtHalfBath <- as.integer(house$BsmtHalfBath)
house$FullBath <- as.integer(house$FullBath)
house$HalfBath <- as.integer(house$HalfBath)
house$BedroomAbvGr <- as.integer(house$BedroomAbvGr)
house$KitchenAbvGr <- as.integer(house$KitchenAbvGr)
house$TotRmsAbvGrd <- as.integer(house$TotRmsAbvGrd)
house$Fireplaces <- as.integer(house$Fireplaces)
house$GarageCars <- as.integer(house$GarageCars)
house$GarageArea <- as.integer(house$GarageArea)
house$WoodDeckSF <- as.integer(house$WoodDeckSF)
house$OpenPorchSF <- as.integer(house$OpenPorchSF)
house$EnclosedPorch <- as.integer(house$EnclosedPorch)
house$X3SsnPorch <- as.integer(house$X3SsnPorch)
house$ScreenPorch <- as.integer(house$ScreenPorch)
house$PoolArea <- as.integer(house$PoolArea)
house$MiscVal <- as.integer(house$MiscVal)

## Dates: years as integers, months as factors
house$YearBuilt <- as.integer(house$YearBuilt)
house$YearRemodAdd <- as.integer(house$YearRemodAdd)
house$GarageYrBlt <- as.integer(house$GarageYrBlt)
house$MoSold <- as.factor(house$MoSold)
house$YrSold <- as.integer(house$YrSold)
```

NA Revisited

Change the string “NA” to “N/A” for variables that are allowed to have “NA” as a value (e.g., Alley). I don’t change the “NA” strings to `NA` here (I did it earlier) because otherwise the change of class insert interpolated values instead of `NA` s.

```
## Store all columns that can have "NA" as a valid entry
na_names = c("Alley", "BsmtQual", "BsmtCond", "BsmtExposure", "BsmtFinType1", "BsmtFinType2", "F
ireplaceQu", "GarageType", "GarageQual", "GarageCond", "PoolQC", "Fence", "MiscFeature", "MasVnr
Type")

## Re-level "NA" columns
for (j in 1:ncol(house)) {
  if (sum(colnames(house)[j]==na_names)!=0) {  # if the column can contain "NA" as a string in t
he training data...
    levels(house[,j])[levels(house[,j])=="NA"] <- "N/A"
  }
}
```

Remove NA Columns

Remove columns from both sets which have too many NAs in the training set, and then remove rows from the training set with NAs left.

```
## Remove NA columns
ct_na <- rep(0, length=ncol(house))
for (j in 1:ncol(house)) {
  ct_na[j] <- sum(is.na(house[,j]))
}
house <- house[-c(1:80)[ct_na>50]]
```

Interpolate NA values in the data

```
## Note which columns need to have NAs interpolated
ct_na <- rep(0, length=ncol(house))
for (j in 1:ncol(house)) {
  ct_na[j] <- sum(is.na(house[,j]))
}
ct_na
```

```
 [1]  0  0  4  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  1
[24]  1  0 23  0  0  0  0  0  0  0  1  0  1  1  1  0  0  0  1  0  0  0  0
[47]  2  2  0  0  0  0  1  0  2  0  0  0  1  1  0  0  0  0  0  0  0  0  0
[70]  0  0  0  0  0  0  1  0  0
```

```
na_boo <- ifelse(ct_na!=0, T, F)
colnames(house)[na_boo]
```

```
 [1] "MSZoning"     "Utilities"    "Exterior1st"  "Exterior2nd" 
 [5] "MasVnrArea"   "BsmtFinSF1"   "BsmtFinSF2"   "BsmtUnfSF"   
 [9] "TotalBsmtSF"  "Electrical"   "BsmtFullBath" "BsmtHalfBath"
[13] "KitchenQual"  "Functional"   "GarageCars"   "GarageArea"  
[17] "SaleType"
```

```
## Choose most common factor
house$MSZoning[is.na(house$MSZoning)] <- "RL"
house$Utilities[is.na(house$Utilities)] <- "AllPub"
house$Exterior1st[is.na(house$Exterior1st)] <- "VinylSd"
house$Exterior2nd[is.na(house$Exterior2nd)] <- "VinylSd"
house$Electrical[is.na(house$Electrical)] <- "SBrkr"
house$KitchenQual[is.na(house$KitchenQual)] <- "TA"
house$Functional[is.na(house$Functional)] <- "Typ"
house$SaleType[is.na(house$SaleType)] <- "WD"

for (i in 1:nrow(house)) {
  ## Check logic on Masonry veneer
  if (is.na(house$MasVnrArea[i])) {
    if (house$MasVnrType[i] == "None") {
      house$MasVnrArea[i] <- 0  # if there isn't any masonry veneer, then the NA should be repla
ced with 0
    } else {
      house$MasVnrArea[i] <- mean(house$MasVnrArea, na.rm=T)  # if there is veneer, replace with
 the average
    }
  }

  ## Basement
  if (is.na(house$BsmtFinSF1[i])) {
    if (house$BsmtQual[i] != "N/A") {
      # if there is a basement, then use the average
      house$BsmtFinSF1[i] <- mean(house$BsmtFinSF1, na.rm=T)
    } else {
      # if there isn't a basement, use 0
      house$BsmtFinSF1[i] <- 0
    }
  }
  if (is.na(house$BsmtFinSF2[i])) {
    if (house$BsmtQual[i]!="N/A") {
      house$BsmtFinSF2[i] <- mean(house$BsmtFinSF2, na.rm=T)
    } else {
      house$BsmtFinSF2[i] <- 0
    }
  }
  if (is.na(house$BsmtUnfSF[i])) {
    if (house$BsmtQual[i]!="N/A") {
      house$BsmtUnfSF[i] <- mean(house$BsmtUnfSF, na.rm=T)
    } else {
      house$BsmtUnfSF[i] <- 0
    }
  }
  if (is.na(house$TotalBsmtSF[i])) {
    if (house$BsmtQual[i]!="N/A") {
      house$TotalBsmtSF[i] <- mean(house$TotalBsmtSF, na.rm=T)
    } else {
      house$TotalBsmtSF[i] <- 0
    }
  }
```

```
  if (is.na(house$BsmtFullBath[i])) {
    if (house$BsmtQual[i]!="N/A") {
      house$BsmtFullBath[i] <- mean(house$BsmtFullBath, na.rm=T)
    } else {
      house$BsmtFullBath[i] <- 0
    }
  }
  if (is.na(house$BsmtHalfBath[i])) {
    if (house$BsmtQual[i]!="N/A") {
      house$BsmtHalfBath[i] <- mean(house$BsmtHalfBath, na.rm=T)
    } else {
      house$BsmtHalfBath[i] <- 0
    }
  }

  ## Garage
  if (is.na(house$GarageCars[i])) {
    if (house$GarageType[i] != "N/A") {
      # if there is a garage, then use the average
      house$GarageCars[i] <- mean(house$GarageCars, na.rm=T)
    } else {
      # if there isn't a basement, use 0
      house$GarageCars[i] <- 0
    }
  }
  if (is.na(house$GarageArea[i])) {
    if (house$GarageType[i] != "N/A") {
      # if there is a garage, then use the average
      house$GarageArea[i] <- mean(house$GarageArea, na.rm=T)
    } else {
      # if there isn't a basement, use 0
      house$GarageArea[i] <- 0
    }
  }
}


## Note which columns need to have NAs interpolated
ct_na <- rep(0, length=ncol(house))
for (j in 1:ncol(house)) {
  ct_na[j] <- sum(is.na(house[,j]))
}
ct_na
```

```
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[71] 0 0 0 0 0 0 0 0
```

```
na_boo <- ifelse(ct_na!=0, T, F)
colnames(house)[na_boo]  # all set!
```

```
character(0)
```

Split

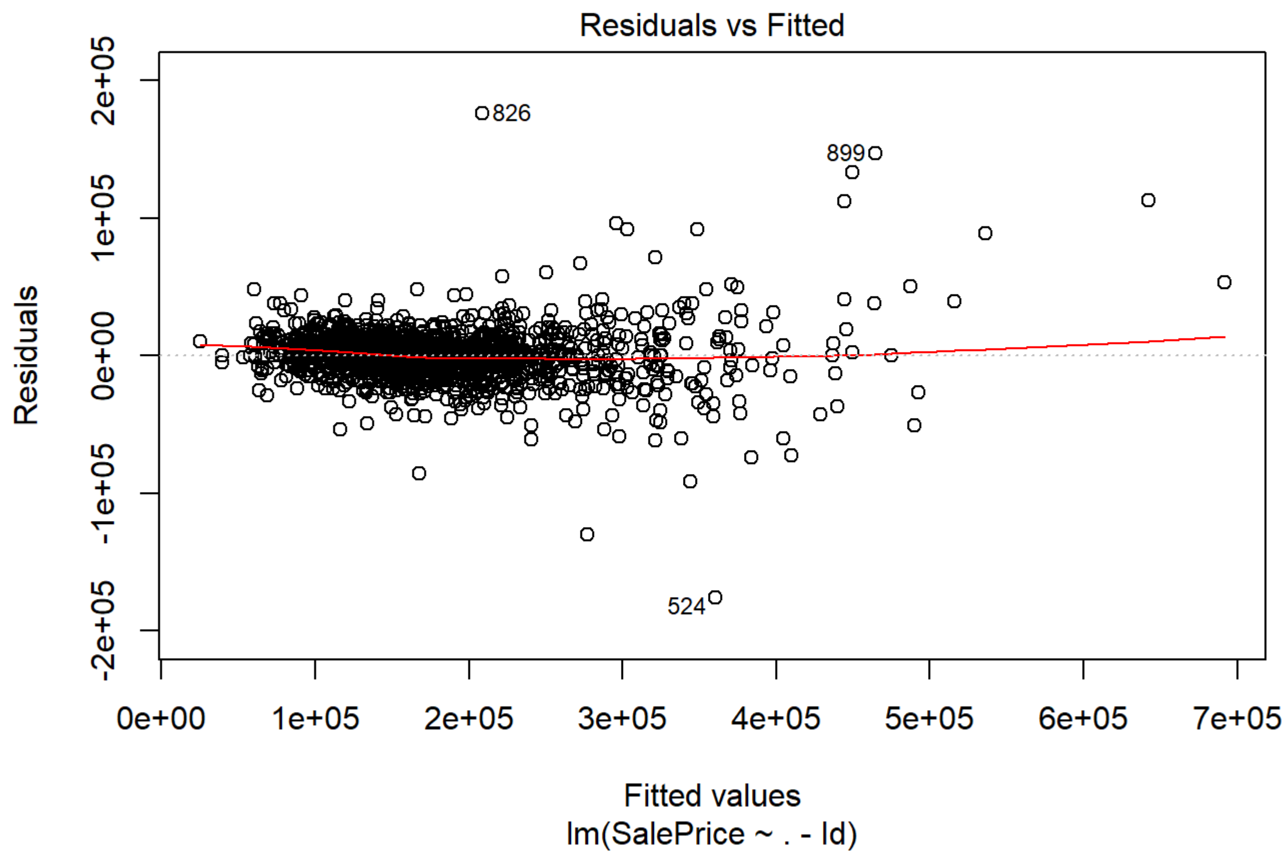
The following code splits the data back into the training and testing sets.

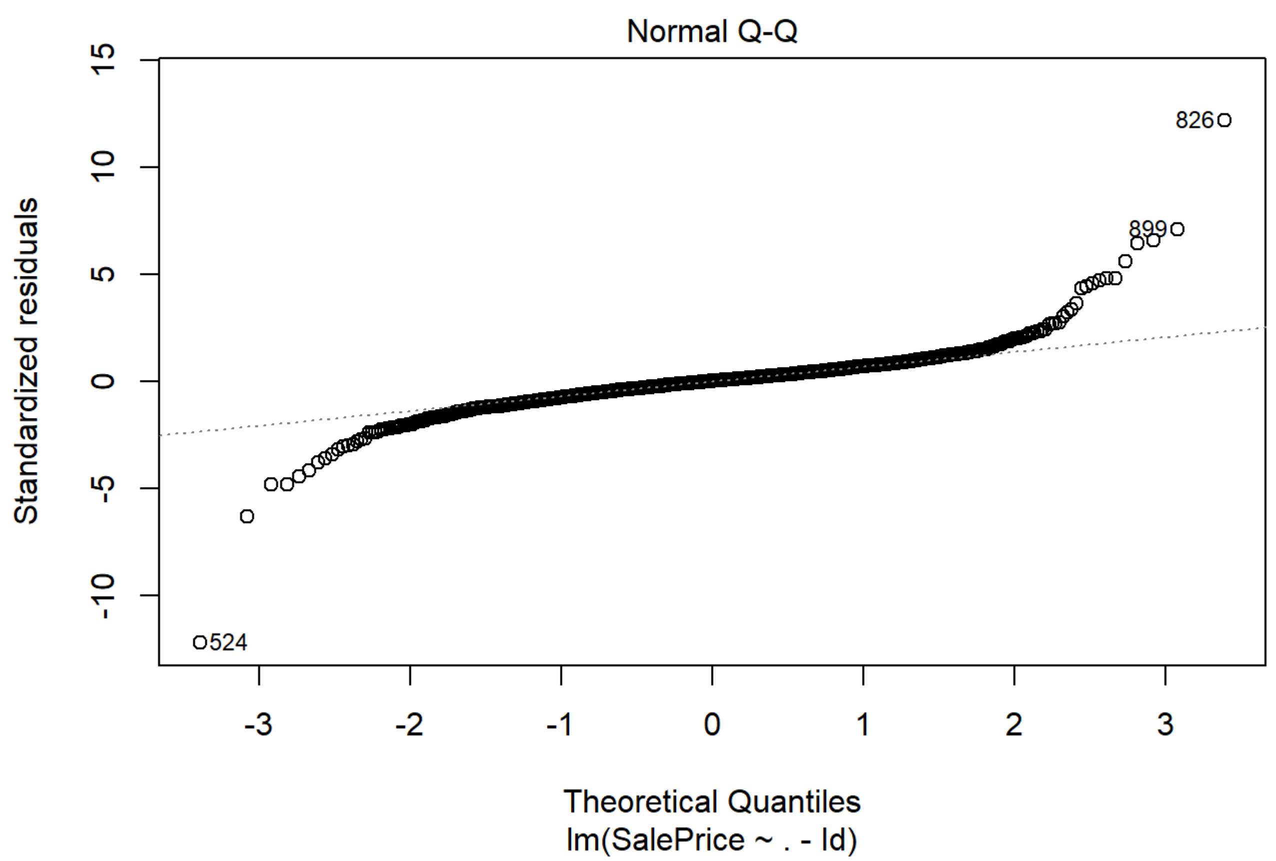
```
train <- house[1:1460,]
test <- house[1461:2919,]
```

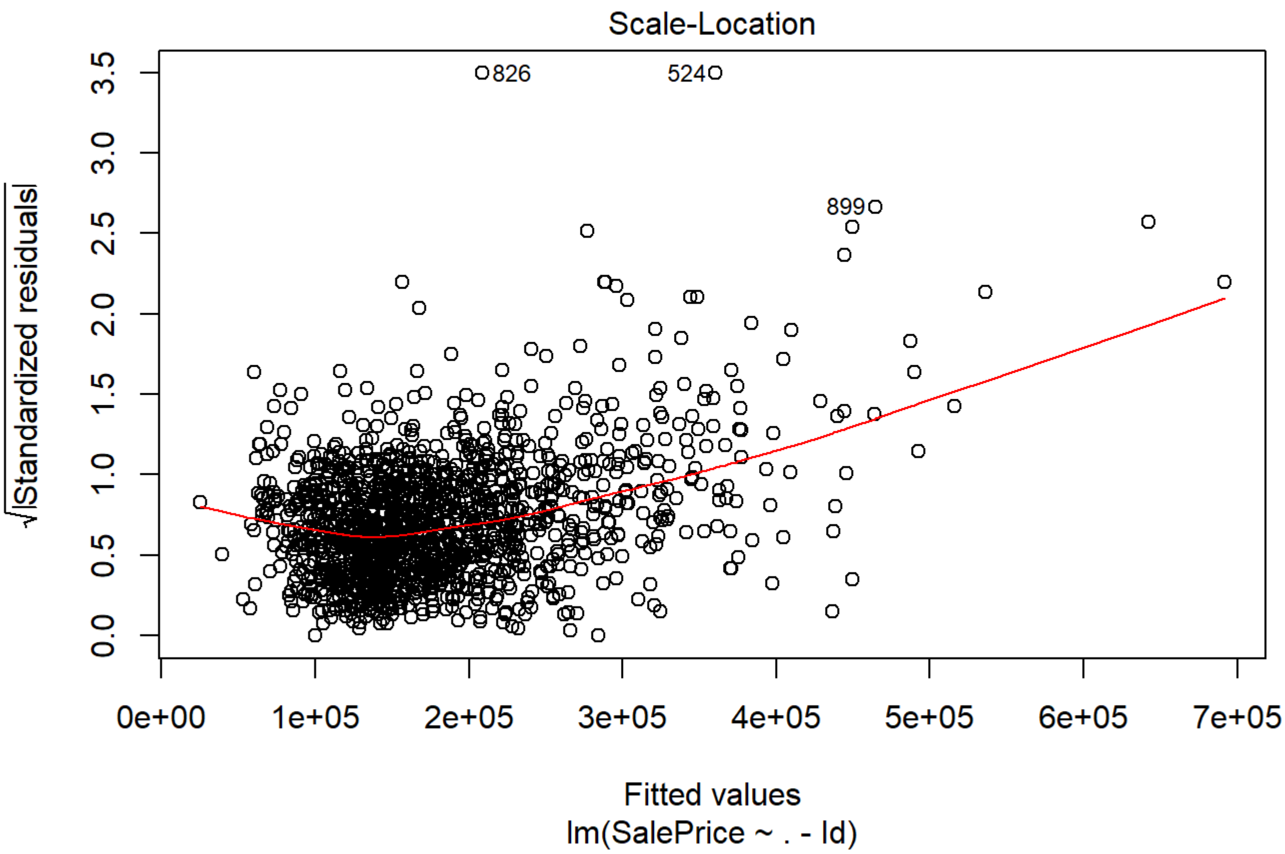
Linear Diagnostics

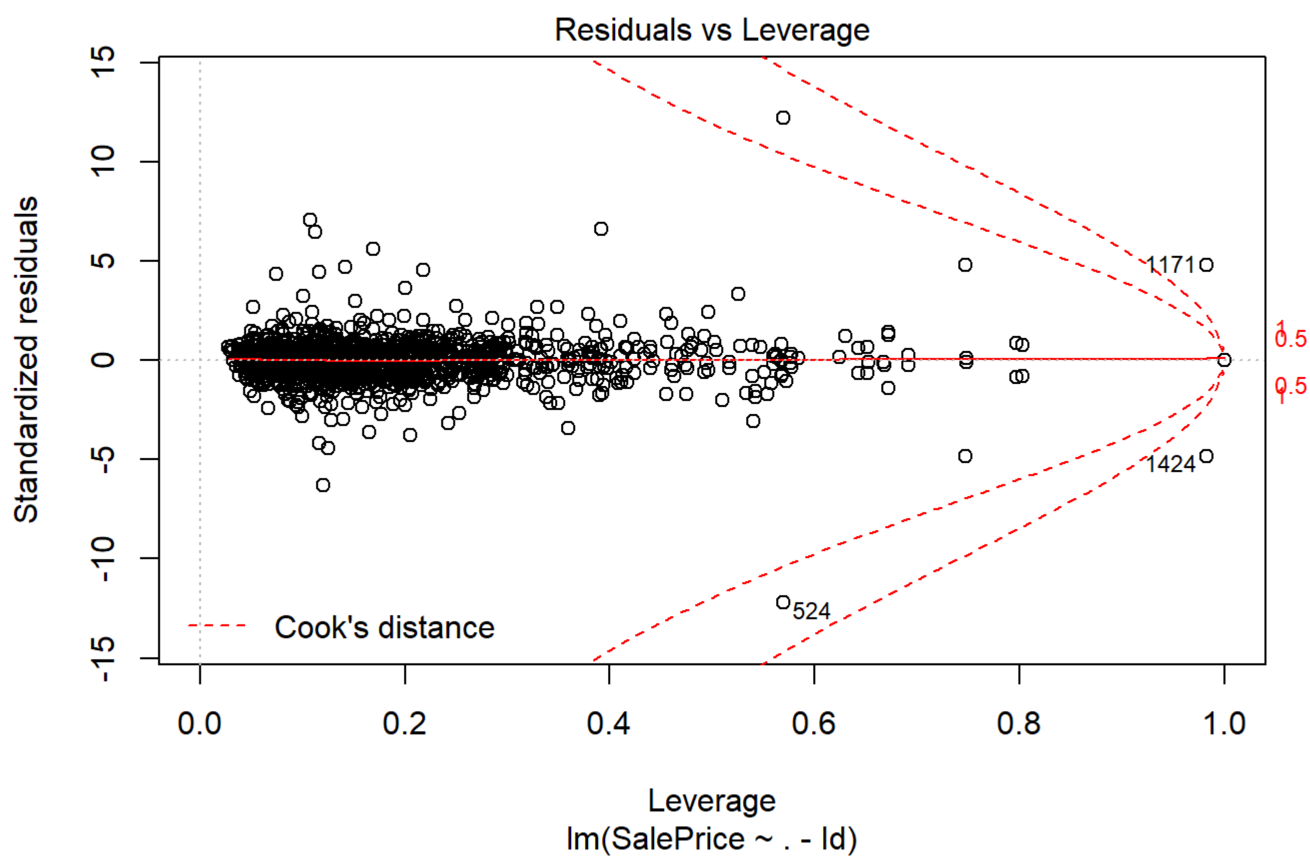
The following section will look at linear diagnostics to evaluate any other data changes that need to be made.

```
lin_fit <- lm(SalePrice~.-Id, data=train)
plot(lin_fit)
```


Residuals vs Fitted
Residuals
2e+05
1e+05
0e+00
-1e+05
-2e+05
826
899
524
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
6e+05
7e+05
Fitted values
lm(SalePrice ~ . - Id)


Normal Q-Q
Standardized residuals
15
10
5
0
-5
-10
826
899
524
-3
-2
-1
0
1
2
3
Theoretical Quantiles
lm(SalePrice ~ . - Id)


Scale-Location
826
524
899
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
6e+05
7e+05
Fitted values
lm(SalePrice ~ . - Id)


Residuals vs Leverage
1171
1424
524
Cook's distance
1
0.5
0.5
1
Standardized residuals
-15
-10
-5
0
5
10
15
0.0
0.2
0.4
0.6
0.8
1.0
Leverage
lm(SalePrice ~ . - Id)

Based on the warning message, points 121, 186, 250, 325, 332, 346, 375, 398, 532, 582, 594, 664, 808, 819, 941, 945, 998, 1006, 1182, 1225, 1264, 1269, 1291, 1314, 1363, 1378 should be removed. Points 1171 and 1424 also have high leverage, so they will be removed. The residuals vs. fitted plot looks pretty good (so the data is reasonably linear and tere is a fairly constant variance of the error terms). The outliers (826, 524) are removed because they are also high leverage points.

```
train <- train[-c(121,186,250,325,332,346,375,398,524,532,582,594,664,808,819,826,941,945,998,10
06,1171,1182,1225,1264,1269,1291,1314,1363,1378,1424),]
```

After removing the statistically “bad” points, the variable `Utilities` can be removed because they all have the same value

```
train <- train[,-9]; test <- test[,-9]
```

It was discussed during the midterm that models may fit better when the response is log-transformed.

Here, I set a seed for reproducibility.

```
set.seed(1)
```

Decision Trees

In this section, I will explore a very large tree and then prune the tree down. I will try this using both the response as-is and the log-transformed response.

Decision trees require the `tree` library.

```
library(tree)
```

Big Tree

```
tree.log.model <- tree(log(SalePrice)~.-Id, data=train)
print(tree.log.model); print(summary(tree.log.model))
```

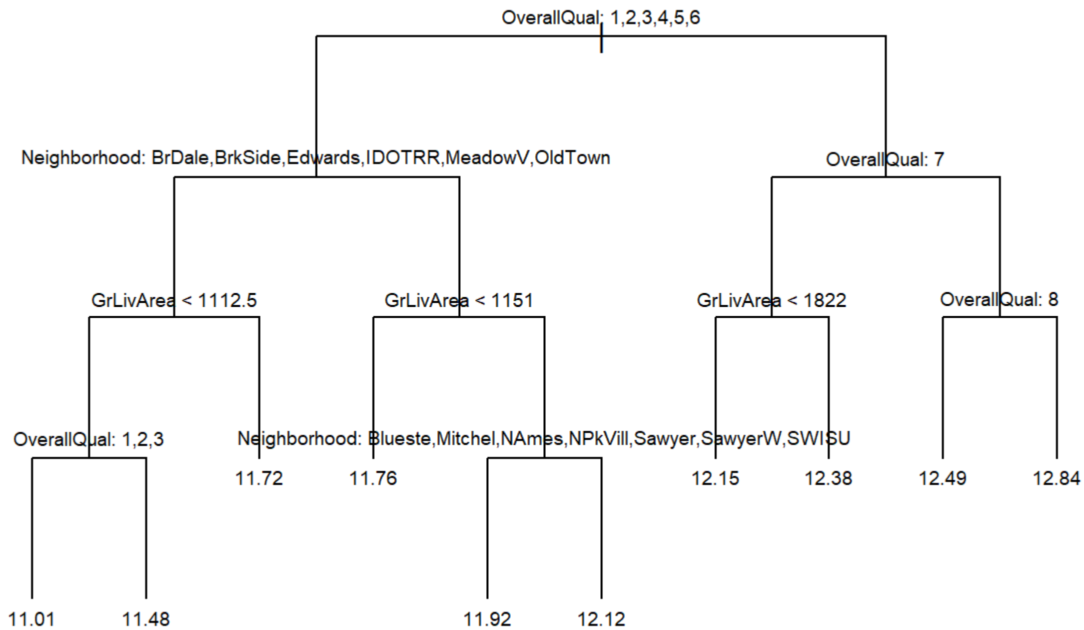
```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 1430 227.500 12.02
   2) OverallQual: 1,2,3,4,5,6 892  72.990 11.81
     4) Neighborhood: BrDale,BrkSide,Edwards,IDOTRR,MeadowV,OldTown 299  26.700 11.60
       8) GrLivArea < 1112.5 123  10.110 11.43
        16) OverallQual: 1,2,3 13   1.177 11.01 *
        17) OverallQual: 4,5,6 110   6.259 11.48 *
       9) GrLivArea > 1112.5 176  10.730 11.72 *
     5) Neighborhood: Blueste,ClearCr,CollgCr,Crawfor,Gilbert,Mitchel,NAmes,NPkVill,NridgHt,NWAm
es,Sawyer,SawyerW,Somerst,SWISU,Timber,Veenker 593  26.720 11.91
      10) GrLivArea < 1151 228   6.000 11.76 *
      11) GrLivArea > 1151 365  12.170 12.01
        22) Neighborhood: Blueste,Mitchel,NAmes,NPkVill,Sawyer,SawyerW,SWISU 205   5.301 11.92 *
        23) Neighborhood: ClearCr,CollgCr,Crawfor,Gilbert,NridgHt,NWAmes,Somerst,Timber,Veenker
160   3.363 12.12 *
   3) OverallQual: 7,8,9,10 538  49.010 12.37
     6) OverallQual: 7 316  14.100 12.22
      12) GrLivArea < 1822 214   6.778 12.15 *
      13) GrLivArea > 1822 102   3.728 12.38 *
     7) OverallQual: 8,9,10 222  17.920 12.58
      14) OverallQual: 8 165   8.658 12.49 *
      15) OverallQual: 9,10 57   4.142 12.84 *
```

```
Regression tree:
tree(formula = log(SalePrice) ~ . - Id, data = train)
Variables actually used in tree construction:
[1] "OverallQual"  "Neighborhood" "GrLivArea"
Number of terminal nodes:  10
Residual mean deviance:  0.03953 = 56.14 / 1420
Distribution of residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.122000 -0.109600  0.007322  0.000000  0.119000  0.734800
```

The “big” tree fit to the log-transformed response has 11 terminal nodes using `OverallQual`, `Neighborhood`, `GrLivArea`, and `CentralAir` as key predictors.

The following code creates the plot.

```
plot(tree.log.model, type="uniform")
text(tree.log.model, pretty=0, cex=0.6)
```



The following code creates the prediction using the big-tree model.

```
tree.log.prediction <- exp(predict(tree.log.model, newdata=test))  # exp to return to actual pri
ce instead of log price

write.csv(data.frame(Id=test$Id, SalePrice=tree.log.prediction), "Tree_Log_Prediction.csv", row.
names=F)
```

The Kaggle score for this model was **0.22102**.

The following code uses cross-validation to predict the error.

```
set.seed(428) # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=10, labels=FALSE) # split data into 10 folds

out <- c()
for (i in 1:10) {
  train.cv <- train[fold.index!=i,]
  test.cv.X <- train[fold.index==i,-77]
  test.cv.y <- train[fold.index==i,77]

  tree.log.model.cv <- tree(log(SalePrice)~.-Id, data=train.cv)
  tree.log.prediction.cv <- exp(predict(tree.log.model.cv, newdata=test.cv.X))
  tree.log.rmsle.cv <- sqrt(mean((log(tree.log.prediction.cv)-log(test.cv.y))^2))
  out <- c(out, tree.log.rmsle.cv)
}
mean(out)
```

```
[1] 0.2091555
```

The estimated RMSLE is 0.2091555 for the (unpruned) decision tree method; this is quite close to the true test RMSLE.

Pruned Tree

The following code performs cross-validation to determine the optimal tree size, then prunes the tree to that size.

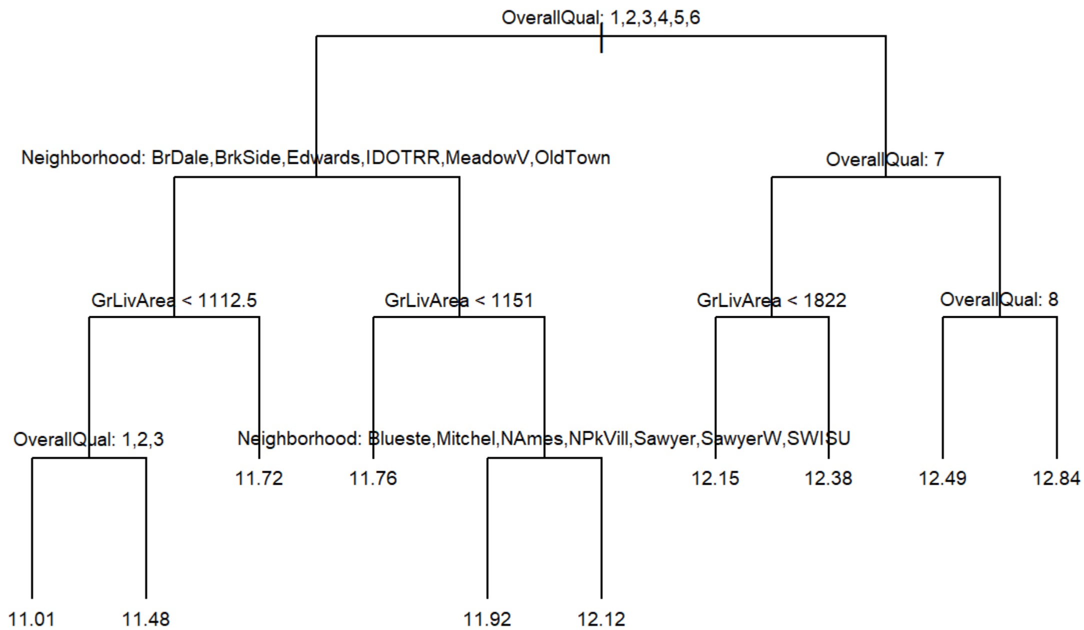
```
set.seed(428)  # set seed for consistency
cv.tree.log.model <- cv.tree(tree.log.model, FUN=prune.tree, K=10)
tree.prune.log.size <- cv.tree.log.model$size[which.min(cv.tree.log.model$dev)]
tree.prune.log.model <- prune.tree(tree.log.model, best=tree.prune.log.size)
```

The following code describes the pruned tree.

```
tree.prune.log.model
```

```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 1430 227.500 12.02
   2) OverallQual: 1,2,3,4,5,6 892  72.990 11.81
     4) Neighborhood: BrDale,BrkSide,Edwards,IDOTRR,MeadowV,OldTown 299  26.700 11.60
       8) GrLivArea < 1112.5 123  10.110 11.43
        16) OverallQual: 1,2,3 13   1.177 11.01 *
        17) OverallQual: 4,5,6 110   6.259 11.48 *
       9) GrLivArea > 1112.5 176  10.730 11.72 *
     5) Neighborhood: Blueste,ClearCr,CollgCr,Crawfor,Gilbert,Mitchel,NAmes,NPkVill,NridgHt,NWAm
es,Sawyer,SawyerW,Somerst,SWISU,Timber,Veenker 593  26.720 11.91
      10) GrLivArea < 1151 228   6.000 11.76 *
      11) GrLivArea > 1151 365  12.170 12.01
        22) Neighborhood: Blueste,Mitchel,NAmes,NPkVill,Sawyer,SawyerW,SWISU 205   5.301 11.92 *
        23) Neighborhood: ClearCr,CollgCr,Crawfor,Gilbert,NridgHt,NWAmes,Somerst,Timber,Veenker 
160   3.363 12.12 *
   3) OverallQual: 7,8,9,10 538  49.010 12.37
     6) OverallQual: 7 316  14.100 12.22
      12) GrLivArea < 1822 214   6.778 12.15 *
      13) GrLivArea > 1822 102   3.728 12.38 *
     7) OverallQual: 8,9,10 222  17.920 12.58
      14) OverallQual: 8 165   8.658 12.49 *
      15) OverallQual: 9,10 57   4.142 12.84 *
```

```
plot(tree.prune.log.model, type="uniform"); text(tree.prune.log.model, pretty=0, cex=0.6)
```




```
summary(tree.prune.log.model)
```

```
Regression tree:
tree(formula = log(SalePrice) ~ . - Id, data = train)
Variables actually used in tree construction:
[1] "OverallQual"  "Neighborhood" "GrLivArea"
Number of terminal nodes:  10
Residual mean deviance:  0.03953 = 56.14 / 1420
Distribution of residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.122000 -0.109600  0.007322  0.000000  0.119000  0.734800
```

The pruned tree has 10 nodes and uses the predictors `OverallQual`, `Neighborhood`, and `GrLivArea`; this pruned tree no longer uses `CentralAir` as a predictor.

The following code creates the prediction using the pruned-tree model.

```
tree.prune.log.prediction <- exp(predict(tree.prune.log.model, newdata=test))
write.csv(data.frame(Id=test$Id, SalePrice=tree.prune.log.prediction), "Tree_Prune_Log_Predictio
n.csv", row.names=F)
```

The Kaggle score for this model was **0.22102**. This method performed the same as the big decision tree, which had one more terminal node than the pruned tree.

The following code uses cross-validation to predict the error.

```
set.seed(428) # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=10, labels=FALSE) # split data into 10 folds

out <- c()
for (i in 1:10) {
  train.cv <- train[fold.index!=i,]
  test.cv.X <- train[fold.index==i,-77]
  test.cv.y <- train[fold.index==i,77]

  tree.log.model.cv <- tree(log(SalePrice)~.-Id, data=train.cv)
  tree.log.model.cv.cv <- cv.tree(tree.log.model.cv, FUN=prune.tree, K=10)
  tree.log.model.size.cv <- tree.log.model.cv.cv$size[which.min(tree.log.model.cv.cv$dev)]
  tree.prune.log.model.cv <- prune.tree(tree.log.model.cv, best=tree.log.model.size.cv)
  tree.prune.log.prediction.cv <- exp(predict(tree.prune.log.model.cv, newdata=test.cv.X))
  tree.prune.log.rmsle.cv <- sqrt(mean((log(tree.prune.log.prediction.cv)-log(test.cv.y))^2))
  out <- c(out, tree.prune.log.rmsle.cv)
}
mean(out)
```

```
[1] 0.2091555
```

The estimated RMSLE is 0.2091555 for the pruned decision tree method; this is fairly close to the true test RMSLE.

Big Tree

```
tree.model <- tree(SalePrice~.-Id, data=train)
print(tree.model); print(summary(tree.model))
```

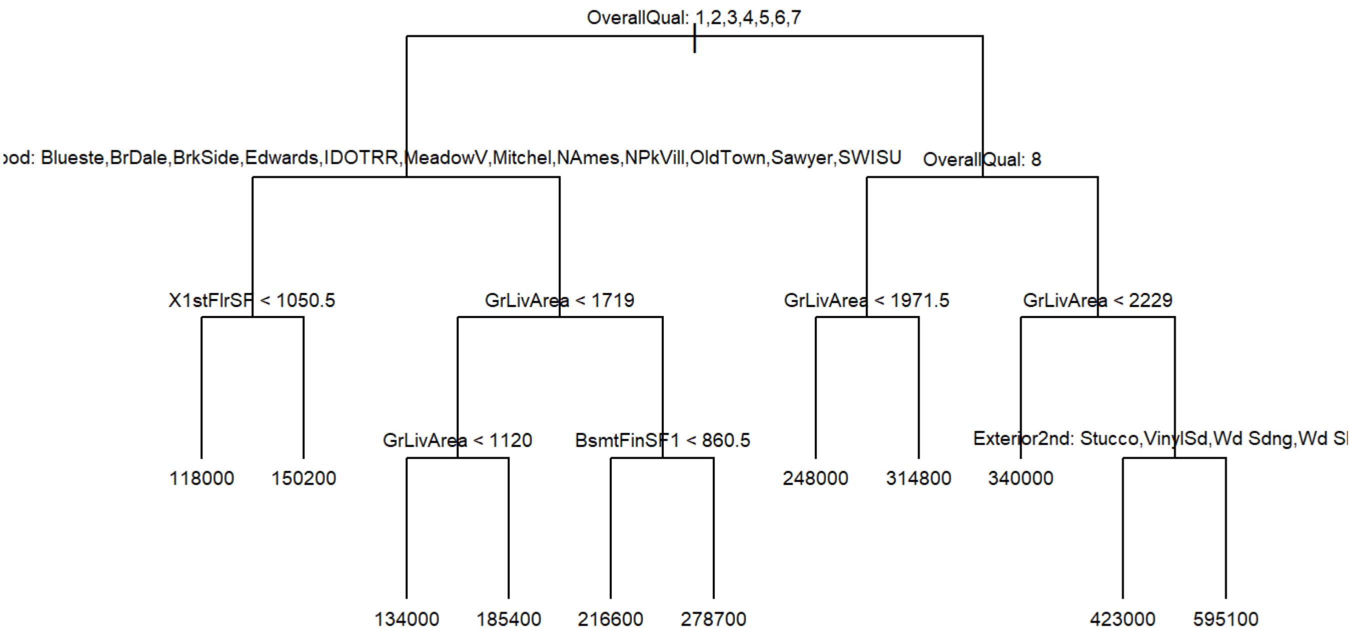
```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 1430 8.916e+12 180200
   2) OverallQual: 1,2,3,4,5,6,7 1208 2.934e+12 157500
     4) Neighborhood: Blueste,BrDale,BrkSide,Edwards,IDOTRR,MeadowV,Mitchel,NAmes,NPkVill,OldTow
n,Sawyer,SWISU 699 8.244e+11 131500
       8) X1stFlrSF < 1050.5 406 3.222e+11 118000 *
       9) X1stFlrSF > 1050.5 293 3.252e+11 150200 *
     5) Neighborhood: Blmngtn,ClearCr,CollgCr,Crawfor,Gilbert,NoRidge,NridgHt,NWAmes,SawyerW,Som
erst,StoneBr,Timber,Veenker 509 9.861e+11 193300
      10) GrLivArea < 1719 343 3.729e+11 176500
        20) GrLivArea < 1120 59 2.058e+10 134000 *
        21) GrLivArea > 1120 284 2.233e+11 185400 *
      11) GrLivArea > 1719 166 3.188e+11 227800
        22) BsmtFinSF1 < 860.5 136 1.647e+11 216600 *
        23) BsmtFinSF1 > 860.5 30 5.937e+10 278700 *
   3) OverallQual: 8,9,10 222 1.970e+12 303800
     6) OverallQual: 8 165 6.560e+11 273500
      12) GrLivArea < 1971.5 102 2.195e+11 248000 *
      13) GrLivArea > 1971.5 63 2.625e+11 314800 *
     7) OverallQual: 9,10 57 7.230e+11 391600
      14) GrLivArea < 2229 34 7.566e+10 340000 *
      15) GrLivArea > 2229 23 4.229e+11 467900
        30) Exterior2nd: Stucco,VinylSd,Wd Sdng,Wd Shng 17 1.956e+11 423000 *
        31) Exterior2nd: CmentBd,HdBoard,ImStucc 6 9.585e+10 595100 *
```

```


Regression tree:
tree(formula = SalePrice ~ . - Id, data = train)
Variables actually used in tree construction:
[1] "OverallQual"  "Neighborhood" "X1stFlrSF"    "GrLivArea"
[5] "BsmtFinSF1"   "Exterior2nd"
Number of terminal nodes:  11
Residual mean deviance:  1.384e+09 = 1.964e+12 / 1419
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-263000  -20000   -1833       0   17250  223200
```

The “big” tree fit to the log-transformed response has 12 terminal nodes using `OverallQual`, `Neighborhood`, `1stFlrSF`, `GrLivArea`, `BsmtFinSf1`, `MoSold`, and `MasVnrArea` as key predictors. This includes more and different predictors than the log-transformed response, but in this case `CentralAir` was left out as a predictor.

The following code creates the plot.

```
plot(tree.model, type="uniform")
text(tree.model, pretty=0, cex=0.6)
```



The following code creates the prediction using the big-tree model.

```
tree.prediction <- predict(tree.model, newdata=test)
write.csv(data.frame(Id=test$Id, SalePrice=tree.prediction), "Tree_Prediction.csv", row.names=F)
```

The Kaggle score for this model was **0.24409**. This big tree performed worse than the big tree that had a log-transformed response.

The following code uses cross-validation to predict the error.

```
set.seed(428) # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=10, labels=FALSE) # split data into 10 folds

out <- c()
for (i in 1:10) {
  train.cv <- train[fold.index!=i,]
  test.cv.X <- train[fold.index==i,-77]
  test.cv.y <- train[fold.index==i,77]

  tree.model.cv <- tree(SalePrice~.-Id, data=train.cv)
  tree.prediction.cv <- predict(tree.model.cv, newdata=test.cv.X)
  tree.rmsle.cv <- sqrt(mean((log(tree.prediction.cv)-log(test.cv.y))^2))
  out <- c(out, tree.rmsle.cv)
}
mean(out)
```

```
[1] 0.2245694
```

The estimated RMSLE is 0.2245694 for the (unpruned) decision tree method; this is fairly close to the true test RMSLE.

Pruned Tree

The following code performs cross-validation to determine the optimal tree size, then prunes the tree to that size.

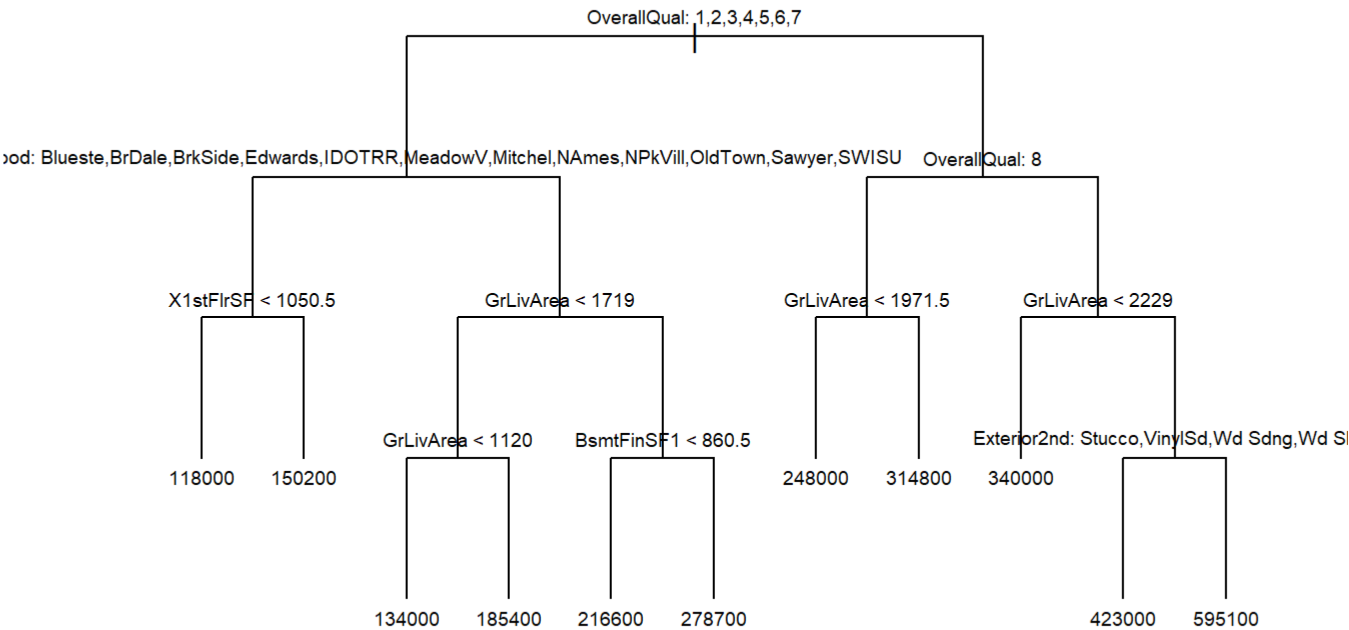
```
set.seed(428)  # set seed for consistency
cv.tree.model <- cv.tree(tree.model, FUN=prune.tree, K=10)
tree.prune.size <- cv.tree.model$size[which.min(cv.tree.model$dev)]
tree.prune.model <- prune.tree(tree.model, best=tree.prune.size)
```

The following code describes the pruned tree.

```
tree.prune.model
```

```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 1430 8.916e+12 180200
   2) OverallQual: 1,2,3,4,5,6,7 1208 2.934e+12 157500
     4) Neighborhood: Blueste,BrDale,BrkSide,Edwards,IDOTRR,MeadowV,Mitchel,NAmes,NPkVill,OldTow
n,Sawyer,SWISU 699 8.244e+11 131500
       8) X1stFlrSF < 1050.5 406 3.222e+11 118000 *
       9) X1stFlrSF > 1050.5 293 3.252e+11 150200 *
     5) Neighborhood: Blmngtn,ClearCr,CollgCr,Crawfor,Gilbert,NoRidge,NridgHt,NWAmes,SawyerW,Som
erst,StoneBr,Timber,Veenker 509 9.861e+11 193300
      10) GrLivArea < 1719 343 3.729e+11 176500
        20) GrLivArea < 1120 59 2.058e+10 134000 *
        21) GrLivArea > 1120 284 2.233e+11 185400 *
      11) GrLivArea > 1719 166 3.188e+11 227800
        22) BsmtFinSF1 < 860.5 136 1.647e+11 216600 *
        23) BsmtFinSF1 > 860.5 30 5.937e+10 278700 *
   3) OverallQual: 8,9,10 222 1.970e+12 303800
     6) OverallQual: 8 165 6.560e+11 273500
      12) GrLivArea < 1971.5 102 2.195e+11 248000 *
      13) GrLivArea > 1971.5 63 2.625e+11 314800 *
     7) OverallQual: 9,10 57 7.230e+11 391600
      14) GrLivArea < 2229 34 7.566e+10 340000 *
      15) GrLivArea > 2229 23 4.229e+11 467900
        30) Exterior2nd: Stucco,VinylSd,Wd Sdng,Wd Shng 17 1.956e+11 423000 *
        31) Exterior2nd: CmentBd,HdBoard,ImStucc 6 9.585e+10 595100 *
```

```
plot(tree.prune.model, type="uniform"); text(tree.prune.model, pretty=0, cex=0.6)
```




```
summary(tree.prune.model)
```

```
Regression tree:
tree(formula = SalePrice ~ . - Id, data = train)
Variables actually used in tree construction:
[1] "OverallQual"  "Neighborhood" "X1stFlrSF"    "GrLivArea"
[5] "BsmtFinSF1"   "Exterior2nd"
Number of terminal nodes:  11
Residual mean deviance:  1.384e+09 = 1.964e+12 / 1419
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-263000  -20000   -1833       0   17250  223200
```

The pruned tree has 12 nodes and uses the predictors `OverallQual` , `Neighborhood` , `1stFlrSF` , `GrLivArea` , `BsmtFinSF1` , and `MoSold` . This tree is the same as the unpruned model, so there is no need to go farther.

Bagging

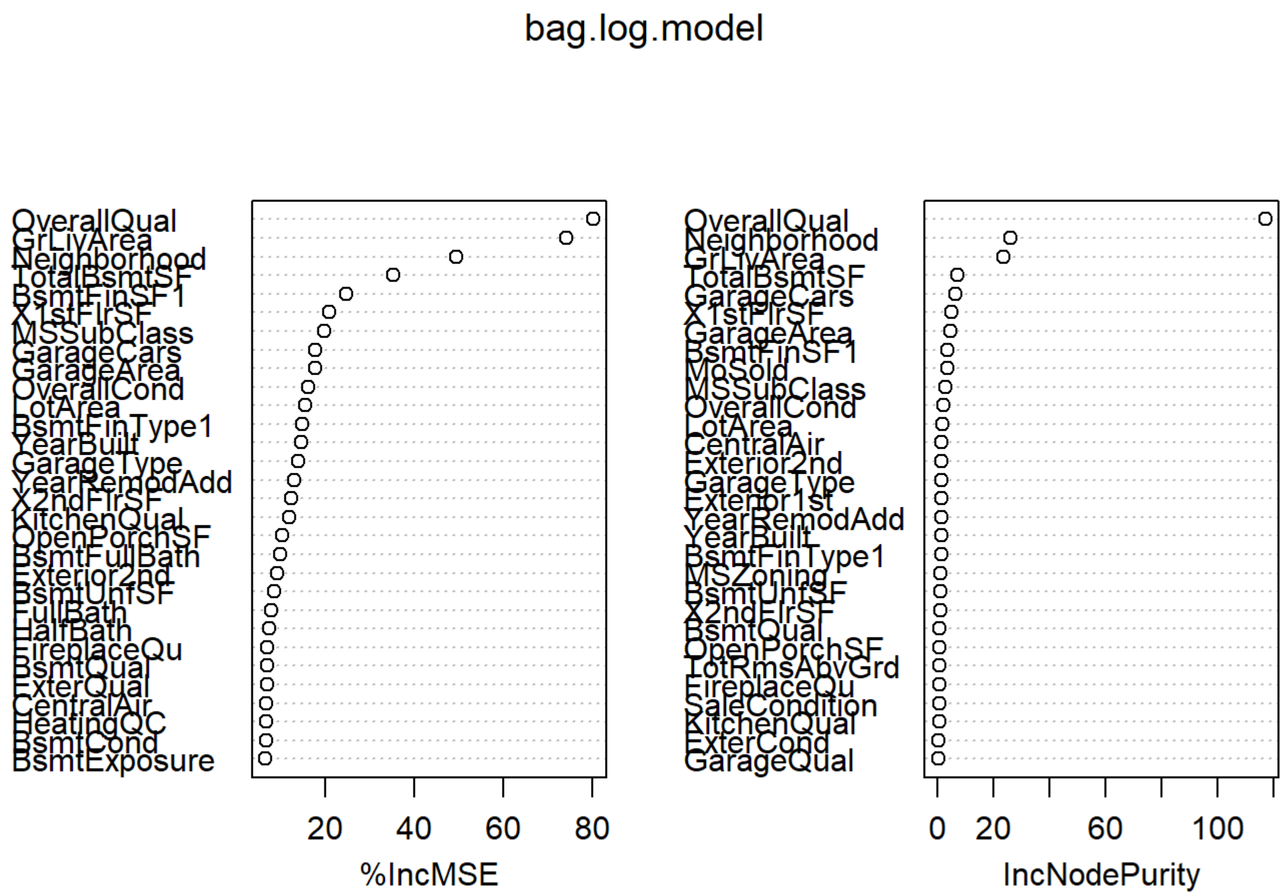
```
library(randomForest)

bag.mtry <- ncol(train)-2  # split candidate count; don't include response or ID
```

This next section, like the previous, will try to fit the bagging model to both the log-transformed response and the response as-is. The maximum number of trees that provided good computational time was `ntree=500`.

```
bag.log.model <- randomForest(log(SalePrice)~.-Id, data=train, mtry=bag.mtry, importance=T, ntre
e=500)
```

The results of the bagged model on the log-transformed response are as follows:

```
varImpPlot(bag.log.model)
```




```
sort(importance(bag.log.model)[,1], decreasing=T)
```

```
  OverallQual     GrLivArea  Neighborhood   TotalBsmtSF    BsmtFinSF1 
   80.2568505    74.1464945    49.3642776    35.3044672    24.7068498 
     X1stFlrSF    MSSubClass    GarageCars    GarageArea   OverallCond 
   20.8831759    19.8807514    17.8485055    17.8042704    16.2326745 
      LotArea  BsmtFinType1     YearBuilt    GarageType  YearRemodAdd 
   15.5972094    14.8024530    14.7196887    13.9204147    13.0125633 
    X2ndFlrSF   KitchenQual   OpenPorchSF  BsmtFullBath   Exterior2nd 
   12.4164566    11.8860807    10.2478527     9.8406122     9.2371837 
    BsmtUnfSF      FullBath      HalfBath   FireplaceQu      BsmtQual 
    8.6472302     7.9665978     7.4115755     7.0821273     6.9504196 
    ExterQual    CentralAir     HeatingQC      BsmtCond  BsmtExposure 
    6.8658701     6.7048432     6.6808736     6.6475488     6.5754705 
     MSZoning   Exterior1st  BedroomAbvGr    GarageQual    Fireplaces 
    6.5517306     6.3700609     6.2273754     6.0551870     5.9941007 
 TotRmsAbvGrd    Foundation  KitchenAbvGr    MasVnrArea    HouseStyle 
    5.9910296     5.6435642     5.4878444     5.2533658     4.9638154 
     BldgType    MasVnrType      LotShape    WoodDeckSF     ExterCond 
    4.8920739     4.0728051     3.8862704     3.5293644     3.1987301 
  ScreenPorch    GarageCond        YrSold        MoSold    Functional 
    2.6276770     2.0243622     1.9455294     1.9265431     1.6616402 
   BsmtFinSF2     LandSlope  BsmtHalfBath     RoofStyle EnclosedPorch 
    1.6217148     1.4821513     1.2408883     0.6996978     0.6059462 
   PavedDrive   MiscFeature    Condition1  LowQualFinSF SaleCondition 
    0.5865199     0.5677298     0.5221729     0.4219599     0.3774627 
   X3SsnPorch      SaleType      PoolArea   LandContour  BsmtFinType2 
    0.2309159     0.2162236     0.0000000    -0.1656398    -0.3928575 
     RoofMatl       MiscVal     LotConfig        PoolQC    Condition2 
   -0.5406951    -0.6167683    -0.6515625    -0.8167276    -1.0010015 
       Street         Alley         Fence       Heating    Electrical 
   -1.0156620    -1.2605710    -1.2942645    -2.0672752    -2.1663978 
```

```
sort(importance(bag.log.model)[,2], decreasing=T)
```

```
 OverallQual  Neighborhood     GrLivArea   TotalBsmtSF    GarageCars
1.172607e+02  2.597052e+01  2.348575e+01  6.993373e+00  6.475515e+00
   X1stFlrSF    GarageArea    BsmtFinSF1        MoSold    MSSubClass
5.035382e+00  4.678298e+00  3.391889e+00  3.347425e+00  2.857895e+00
 OverallCond       LotArea    CentralAir   Exterior2nd    GarageType
1.961702e+00  1.585453e+00  1.460481e+00  1.418408e+00  1.343719e+00
 Exterior1st  YearRemodAdd     YearBuilt  BsmtFinType1      MSZoning
1.275285e+00  1.241484e+00  1.230820e+00  1.186984e+00  8.938785e-01
   BsmtUnfSF     X2ndFlrSF      BsmtQual   OpenPorchSF  TotRmsAbvGrd
8.488568e-01  8.471809e-01  7.941423e-01  6.328579e-01  5.725768e-01
 FireplaceQu SaleCondition   KitchenQual     ExterCond    GarageQual
5.474854e-01  5.341282e-01  5.221640e-01  4.364415e-01  4.079064e-01
   HeatingQC  BsmtExposure      BsmtCond    WoodDeckSF      FullBath
3.904911e-01  3.513144e-01  3.505551e-01  3.368362e-01  3.189838e-01
BsmtFullBath     ExterQual EnclosedPorch    Functional    MasVnrArea
3.018443e-01  3.010762e-01  2.914297e-01  2.677537e-01  2.625416e-01
BedroomAbvGr        YrSold    GarageCond    Fireplaces         Fence
2.565523e-01  2.445999e-01  2.283211e-01  1.956824e-01  1.953176e-01
 LandContour    Condition1  BsmtFinType2     LotConfig    Foundation
1.768634e-01  1.707095e-01  1.551138e-01  1.545689e-01  1.534291e-01
  PavedDrive         Alley      HalfBath      SaleType    HouseStyle
1.517248e-01  1.381025e-01  1.316029e-01  1.301482e-01  1.274957e-01
    LotShape   ScreenPorch    MasVnrType     RoofStyle    Electrical
1.231767e-01  1.217964e-01  1.154393e-01  1.150495e-01  1.075517e-01
    BldgType     LandSlope    BsmtFinSF2       Heating  KitchenAbvGr
7.196176e-02  7.101699e-02  5.173483e-02  5.003869e-02  4.995252e-02
LowQualFinSF  BsmtHalfBath    X3SsnPorch       MiscVal   MiscFeature
3.143547e-02  2.302697e-02  2.148498e-02  2.065335e-02  1.875217e-02
    RoofMatl        PoolQC    Condition2        Street      PoolArea
1.688381e-02  1.452383e-02  1.231950e-02  3.625610e-03  2.237532e-03
```

In terms of both prediction accuracy `[,1]` and purity `[,2]`, the three most important variables are `OverallQual`, `GrLivArea`, and `Neighborhood`; this is consistent with the results from the decision trees. `TotalBsmtSF` is also a somewhat important predictor (more in terms of prediction accuracy than purity), which is consistent with the results for the non-log-transformed response decision tree model.

The following code creates a prediction using the bagging model.

```
bag.log.prediction <- exp(predict(bag.log.model, newdata=test))
write.csv(data.frame(Id=test$Id, SalePrice=bag.log.prediction), "Bagging_Log_Prediction.csv", ro
w.names=F)
```

The Kaggle score for this model was **0.15489**. This shows significant improvement over the decision tree method.

The following code uses cross-validation to predict the error.

```
set.seed(428) # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=10, labels=FALSE) # split data into 10 folds

out <- c()
for (i in 1:10) {
  train.cv <- train[fold.index!=i,]
  test.cv.X <- train[fold.index==i,-77]
  test.cv.y <- train[fold.index==i,77]

  bag.log.model.cv <- randomForest(log(SalePrice)~.-Id, data=train.cv, mtry=bag.mtry, importance
=T, ntree=500)
  bag.log.prediction.cv <- exp(predict(bag.log.model.cv, newdata=test.cv.X))
  bag.log.rmsle.cv <- sqrt(mean((log(bag.log.prediction.cv)-log(test.cv.y))^2))
  out <- c(out, bag.log.rmsle.cv)
}
mean(out)
```

```
[1] 0.1427881
```

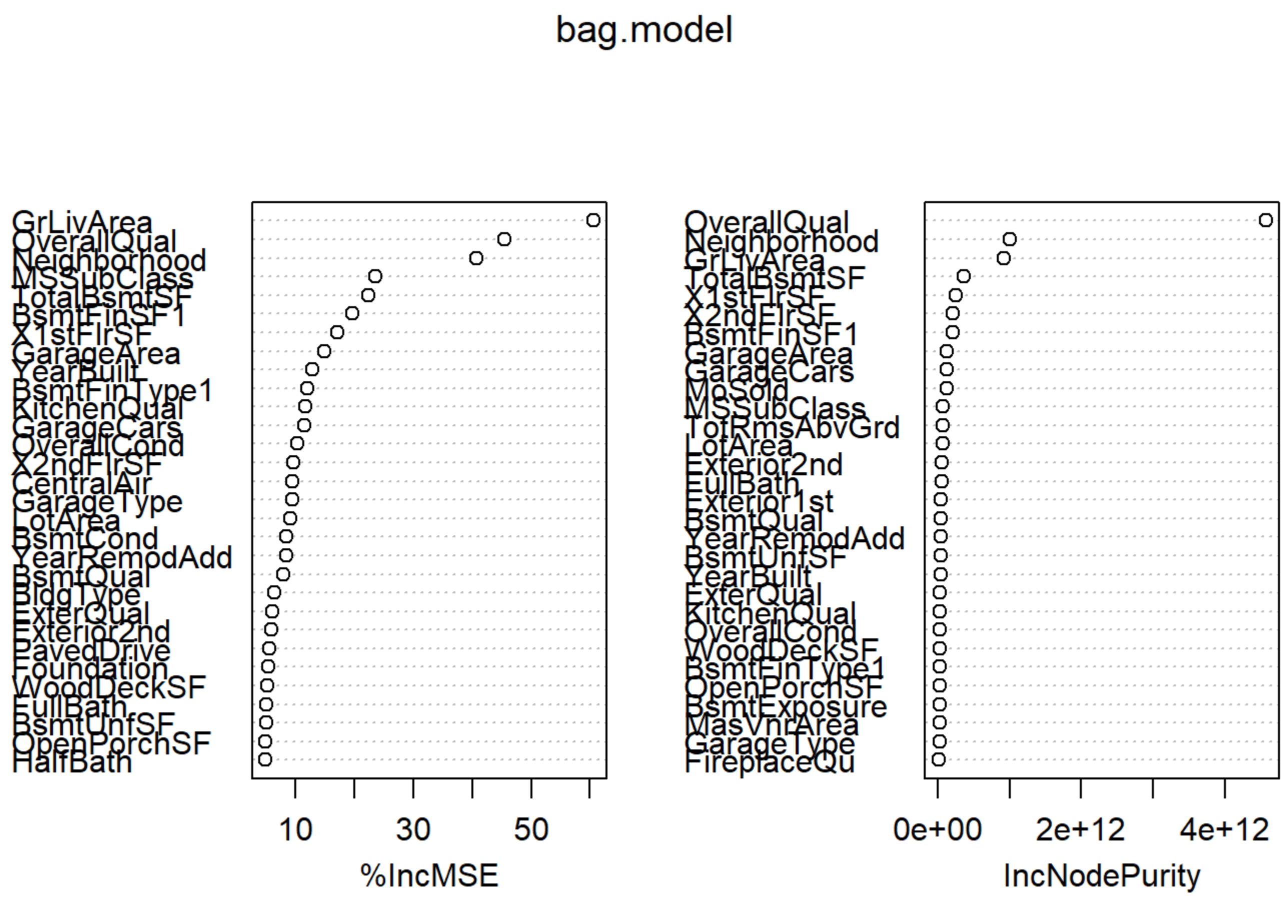
The estimated RMSLE is 0.1427881 for the bagging method; this is fairly close to the true test RMSLE.

The following code fits a bagging model to the data, but doesn’t log-transform the response.

```
bag.model <- randomForest(SalePrice~.-Id, data=train, mtry=bag.mtry, importance=T, ntree=500)
```

The results of the bagged model on the non-transformed response are as follows:

```
varImpPlot(bag.model)
```




```
sort(importance(bag.model)[,1], decreasing=T)
```

```
    GrLivArea   OverallQual  Neighborhood    MSSubClass   TotalBsmtSF 
  60.59333181   45.56003186   40.77695762   23.51591799   22.48447993 
   BsmtFinSF1     X1stFlrSF    GarageArea     YearBuilt  BsmtFinType1 
  19.75802191   17.07995720   14.86325499   12.82823353   12.11719882 
  KitchenQual    GarageCars   OverallCond     X2ndFlrSF    CentralAir 
  11.71784324   11.51966091   10.32519420    9.67328940    9.54093634 
   GarageType       LotArea      BsmtCond  YearRemodAdd      BsmtQual 
   9.44476465    9.23649323    8.45687129    8.43123347    7.92733398 
     BldgType     ExterQual   Exterior2nd    PavedDrive    Foundation 
   6.39258024    6.15360767    5.95786129    5.56192681    5.51650740 
   WoodDeckSF      FullBath     BsmtUnfSF   OpenPorchSF      HalfBath 
   5.31849287    5.09475229    5.01636612    4.98216961    4.96414058 
  Exterior1st SaleCondition  BsmtExposure     HeatingQC  BsmtFullBath 
   4.94346234    4.94298649    4.54575661    4.52457009    4.40732476 
 KitchenAbvGr    GarageQual  BedroomAbvGr    MasVnrArea    HouseStyle 
   4.30151434    4.23303423    4.23119925    4.16400913    3.57077600 
     MSZoning    GarageCond   FireplaceQu         Alley    MasVnrType 
   3.42821109    3.28666239    3.22181886    3.18176753    2.92654394 
    RoofStyle  TotRmsAbvGrd      SaleType  BsmtHalfBath    Condition1 
   2.92276087    2.76536017    2.74270194    2.33162797    2.09235311 
 BsmtFinType2      LotShape   ScreenPorch     ExterCond EnclosedPorch 
   2.05110208    1.98227115    1.68520462    1.62243625    1.49222777 
   BsmtFinSF2  LowQualFinSF    Functional    Fireplaces    X3SsnPorch 
   1.20855229    1.16475005    1.10937156    1.10718105    0.62630478 
  LandContour      RoofMatl     LandSlope       Heating        Street 
   0.34855960    0.20559750    0.13557436    0.01558379    0.00000000 
     PoolArea         Fence    Electrical   MiscFeature        MoSold 
   0.00000000   -0.05765457   -0.29640174   -0.42829702   -0.62043243 
      MiscVal     LotConfig        PoolQC    Condition2        YrSold 
  -0.83300242   -0.90464039   -1.00100150   -1.41252541   -1.62138434 
```

```
sort(importance(bag.model)[,2], decreasing=T)
```

```
  OverallQual  Neighborhood     GrLivArea   TotalBsmtSF     X1stFlrSF 
 4.576966e+12  1.001657e+12  9.161547e+11  3.563538e+11  2.449930e+11 
    X2ndFlrSF    BsmtFinSF1    GarageArea    GarageCars        MoSold 
 2.092046e+11  2.016263e+11  1.249908e+11  1.224555e+11  1.212271e+11 
   MSSubClass  TotRmsAbvGrd       LotArea   Exterior2nd      FullBath 
 7.264862e+10  7.129839e+10  6.625353e+10  5.199145e+10  5.051469e+10 
  Exterior1st      BsmtQual  YearRemodAdd     BsmtUnfSF     YearBuilt 
 4.526345e+10  4.485674e+10  4.291098e+10  3.993900e+10  3.311773e+10 
    ExterQual   KitchenQual   OverallCond    WoodDeckSF  BsmtFinType1 
 3.131083e+10  3.062799e+10  2.872770e+10  2.496277e+10  2.471658e+10 
  OpenPorchSF  BsmtExposure    MasVnrArea    GarageType   FireplaceQu 
 2.368344e+10  2.293845e+10  2.030680e+10  2.004585e+10  1.637433e+10 
SaleCondition    CentralAir  BedroomAbvGr     LotConfig      SaleType 
 1.461987e+10  1.312680e+10  9.270183e+09  8.936029e+09  8.506967e+09 
       YrSold    MasVnrType  BsmtFullBath      BsmtCond   ScreenPorch 
 8.482592e+09  8.375514e+09  7.826875e+09  7.620512e+09  7.562020e+09 
   Fireplaces     HeatingQC      HalfBath      LotShape EnclosedPorch 
 7.379051e+09  7.156644e+09  6.560922e+09  6.421534e+09  6.090483e+09 
  LandContour     RoofStyle    GarageQual    GarageCond      MSZoning 
 5.333845e+09  5.213701e+09  4.321954e+09  4.043866e+09  3.993117e+09 
   Functional    Condition1         Alley    HouseStyle    Foundation 
 3.820424e+09  3.557361e+09  3.396950e+09  3.383651e+09  3.381264e+09 
    ExterCond  BsmtFinType2      BldgType    PavedDrive    BsmtFinSF2 
 3.189091e+09  3.015736e+09  2.706814e+09  2.549077e+09  2.186420e+09 
        Fence      RoofMatl     LandSlope  KitchenAbvGr    Electrical 
 2.059356e+09  2.050341e+09  1.994604e+09  1.435246e+09  1.326879e+09 
     PoolArea  LowQualFinSF  BsmtHalfBath    X3SsnPorch        PoolQC 
 1.167374e+09  1.067953e+09  1.015859e+09  7.485637e+08  5.869585e+08 
      Heating       MiscVal   MiscFeature    Condition2        Street 
 5.483117e+08  3.006778e+08  1.829275e+08  6.508716e+07  1.763133e+07 
```

In terms of both prediction accuracy `[,1]` and purity `[,2]`, the two three most important variables are `OverallQual` and `Neighborhood`; this is consistent with the results from the decision trees. In terms of prediction accuracy, the variable `GrLivArea` is the number one predictor, but it is ranked third for node purity.

The following code creates a prediction using the bagging model.

```
bag.prediction <- predict(bag.model, newdata=test)
write.csv(data.frame(Id=test$Id, SalePrice=bag.prediction), "Bagging_Prediction.csv", row.names=
F)
```

The Kaggle score for this model was **0.15402**. This shows significant improvement over the decision tree method and is even better than the log-transformed bagging model.

The following code uses cross-validation to predict the error.

```
set.seed(428) # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=10, labels=FALSE) # split data into 10 folds

out <- c()
for (i in 1:10) {
  train.cv <- train[fold.index!=i,]
  test.cv.X <- train[fold.index==i,-77]
  test.cv.y <- train[fold.index==i,77]

  bag.model.cv <- randomForest(SalePrice~.-Id, data=train.cv, mtry=bag.mtry, importance=T, ntree
=500)
  bag.prediction.cv <- predict(bag.model.cv, newdata=test.cv.X)
  bag.rmsle.cv <- sqrt(mean((log(bag.prediction.cv)-log(test.cv.y))^2))
  out <- c(out, bag.rmsle.cv)
}
mean(out)
```

```
[1] 0.1463885
```

The estimated RMSLE is 0.1463885 for the bagging method on the non-transformed response; this is fairly close to the true test RMSLE.

Random Forest

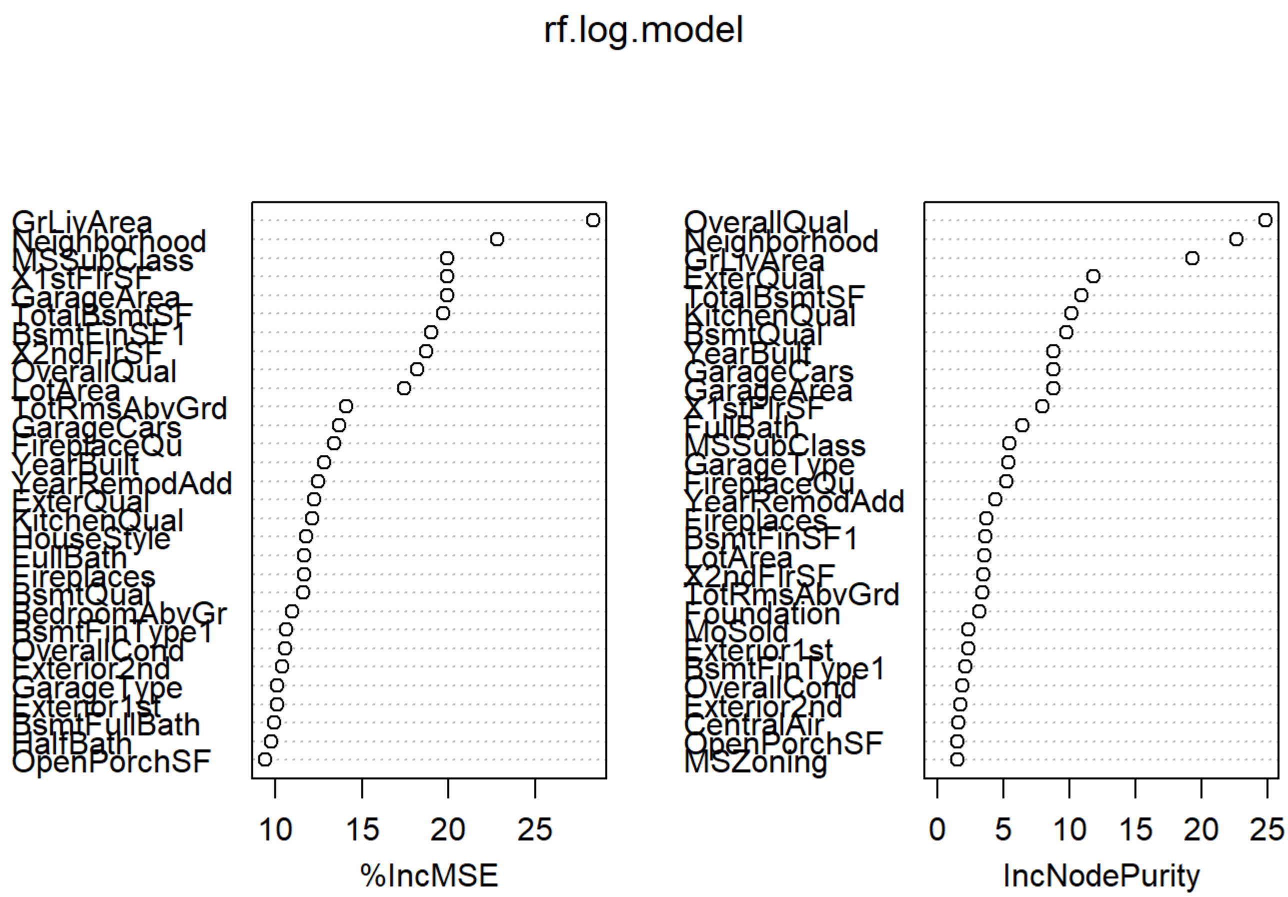
```
library(randomForest)

rf.mtry <- round(sqrt(ncol(train)-2))  # split candidate count; don't include response or ID
```

This next section, like the previous, will try to fit the random forest model to both the log-transformed response and the response as-is. The maximum number of trees that provided good computational time was `ntree=500`.

```
rf.log.model <- randomForest(log(SalePrice)~.-Id, data=train, mtry=rf.mtry, importance=T, ntree=
500)
```

The results of the random forest model on the log-transformed response are as follows:

```
varImpPlot(rf.log.model)
```




```
sort(importance(rf.log.model)[,1], decreasing=T)
```

```
    GrLivArea  Neighborhood    MSSubClass     X1stFlrSF    GarageArea
  28.36224031   22.81789224   19.94558169   19.94332924   19.90581675
  TotalBsmtSF    BsmtFinSF1     X2ndFlrSF   OverallQual       LotArea
  19.67691724   18.99868114   18.69139341   18.19503084   17.42745315
 TotRmsAbvGrd    GarageCars   FireplaceQu     YearBuilt  YearRemodAdd
  14.09743014   13.70646365   13.40304880   12.80275066   12.47056247
    ExterQual   KitchenQual    HouseStyle      FullBath    Fireplaces
  12.22211287   12.12921767   11.78761741   11.67534322   11.65998577
     BsmtQual  BedroomAbvGr  BsmtFinType1   OverallCond   Exterior2nd
  11.59073561   10.98417797   10.63594861   10.56450616   10.39036593
   GarageType   Exterior1st  BsmtFullBath      HalfBath   OpenPorchSF
  10.12993186   10.07684335    9.90834210    9.73551136    9.42306410
 BsmtExposure     BsmtUnfSF      MSZoning    WoodDeckSF      BsmtCond
   9.39609108    9.12703080    8.12511175    7.95330764    7.63416768
   MasVnrType    CentralAir    MasVnrArea      BldgType    Foundation
   7.37055417    7.33006907    7.19279288    6.91560078    6.84643202
    HeatingQC    GarageCond    GarageQual      LotShape    PavedDrive
   6.31540620    4.94448394    4.93184295    4.83587169    4.43981396
    RoofStyle  KitchenAbvGr   ScreenPorch    Functional         Alley
   4.26709697    4.07829513    3.61443221    3.44963209    3.35126552
 BsmtHalfBath        MoSold    Condition1  BsmtFinType2   LandContour
   2.74128207    2.66523533    2.51699389    2.17735756    2.08767506
    LotConfig      SaleType     LandSlope         Fence     ExterCond
   1.68313887    1.68078709    1.67777698    1.43357549    1.25135543
      Heating SaleCondition        YrSold      RoofMatl EnclosedPorch
   0.85070108    0.36670738    0.32819385    0.20688505    0.20020511
 LowQualFinSF    BsmtFinSF2    Electrical    X3SsnPorch   MiscFeature
  -0.01251876   -0.03509582   -0.13909719   -0.45617523   -0.77758538
      MiscVal        Street    Condition2        PoolQC      PoolArea
  -0.95563121   -1.09997036   -1.26432578   -1.30250643   -1.69486498
```

```
sort(importance(rf.log.model)[,2], decreasing=T)
```

```
  OverallQual  Neighborhood     GrLivArea     ExterQual   TotalBsmtSF
  24.83596007   22.62475246   19.32069311   11.81594283   10.87100246
  KitchenQual      BsmtQual     YearBuilt    GarageCars    GarageArea
  10.10644517    9.73600447    8.79617596    8.77585509    8.76498222
    X1stFlrSF      FullBath    MSSubClass    GarageType   FireplaceQu
   7.92088992    6.44733891    5.44965625    5.40150922    5.24195411
 YearRemodAdd    Fireplaces    BsmtFinSF1       LotArea     X2ndFlrSF
   4.37390383    3.65900051    3.65238416    3.52052440    3.50110679
 TotRmsAbvGrd    Foundation        MoSold   Exterior1st  BsmtFinType1
   3.37650100    3.13896440    2.33126920    2.29720186    2.11665782
  OverallCond   Exterior2nd    CentralAir   OpenPorchSF      MSZoning
   1.89906652    1.74586713    1.57445226    1.46279817    1.46211046
   GarageCond     BsmtUnfSF  BedroomAbvGr     HeatingQC    HouseStyle
   1.25203737    1.21612355    1.17700375    0.99353949    0.91484798
   MasVnrArea    GarageQual  BsmtExposure    WoodDeckSF      HalfBath
   0.86923178    0.77339696    0.74066482    0.71582734    0.60877926
SaleCondition     RoofStyle    MasVnrType    PavedDrive      BsmtCond
   0.60687610    0.51585462    0.51083626    0.49342828    0.49241031
 BsmtFullBath        YrSold      BldgType      LotShape    Functional
   0.44306119    0.39830241    0.39518445    0.37332705    0.36950494
    ExterCond      SaleType         Fence    Condition1 EnclosedPorch
   0.35774111    0.33857377    0.33731129    0.32305960    0.30771238
  LandContour  BsmtFinType2     LotConfig    Electrical       Heating
   0.29695915    0.28203933    0.27231419    0.24265664    0.21178362
    LandSlope   ScreenPorch         Alley  KitchenAbvGr    BsmtFinSF2
   0.18166543    0.16425319    0.15618629    0.14068760    0.12629211
     RoofMatl  BsmtHalfBath       MiscVal   MiscFeature  LowQualFinSF
   0.08142267    0.06951679    0.06433990    0.06121385    0.03673125
       Street        PoolQC      PoolArea    X3SsnPorch    Condition2
   0.03512957    0.03284057    0.02423484    0.02342917    0.02194812
```

In terms of both prediction accuracy `[,1]` and purity `[,2]`, the three most important variables include `GrLivArea`, and `Neighborhood`; this is consistent with the results from the decision trees and the bagging. `GrLivArea` is an important predictor of accuracy, but `OverallQual` is more important for node purity.

The following code creates a prediction using the random forest model.

```
rf.log.prediction <- exp(predict(rf.log.model, newdata=test))
write.csv(data.frame(Id=test$Id, SalePrice=rf.log.prediction), "RandomForest_Log_Prediction.csv"
, row.names=F)
```

The Kaggle score for this model was **0.15158**. This shows significant improvement over the decision tree method. This shows improvement over the bagging-log model.

The following code uses cross-validation to predict the error.

```
set.seed(428) # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=10, labels=FALSE) # split data into 10 folds

out <- c()
for (i in 1:10) {
  train.cv <- train[fold.index!=i,]
  test.cv.X <- train[fold.index==i,-77]
  test.cv.y <- train[fold.index==i,77]

  rf.log.model.cv <- randomForest(log(SalePrice)~.-Id, data=train.cv, mtry=rf.mtry, importance=T
, ntree=500)
  rf.log.prediction.cv <- exp(predict(rf.log.model.cv, newdata=test.cv.X))
  rf.log.rmsle.cv <- sqrt(mean((log(rf.log.prediction.cv)-log(test.cv.y))^2))
  out <- c(out, rf.log.rmsle.cv)
}
mean(out)
```

```
[1] 0.1401211
```

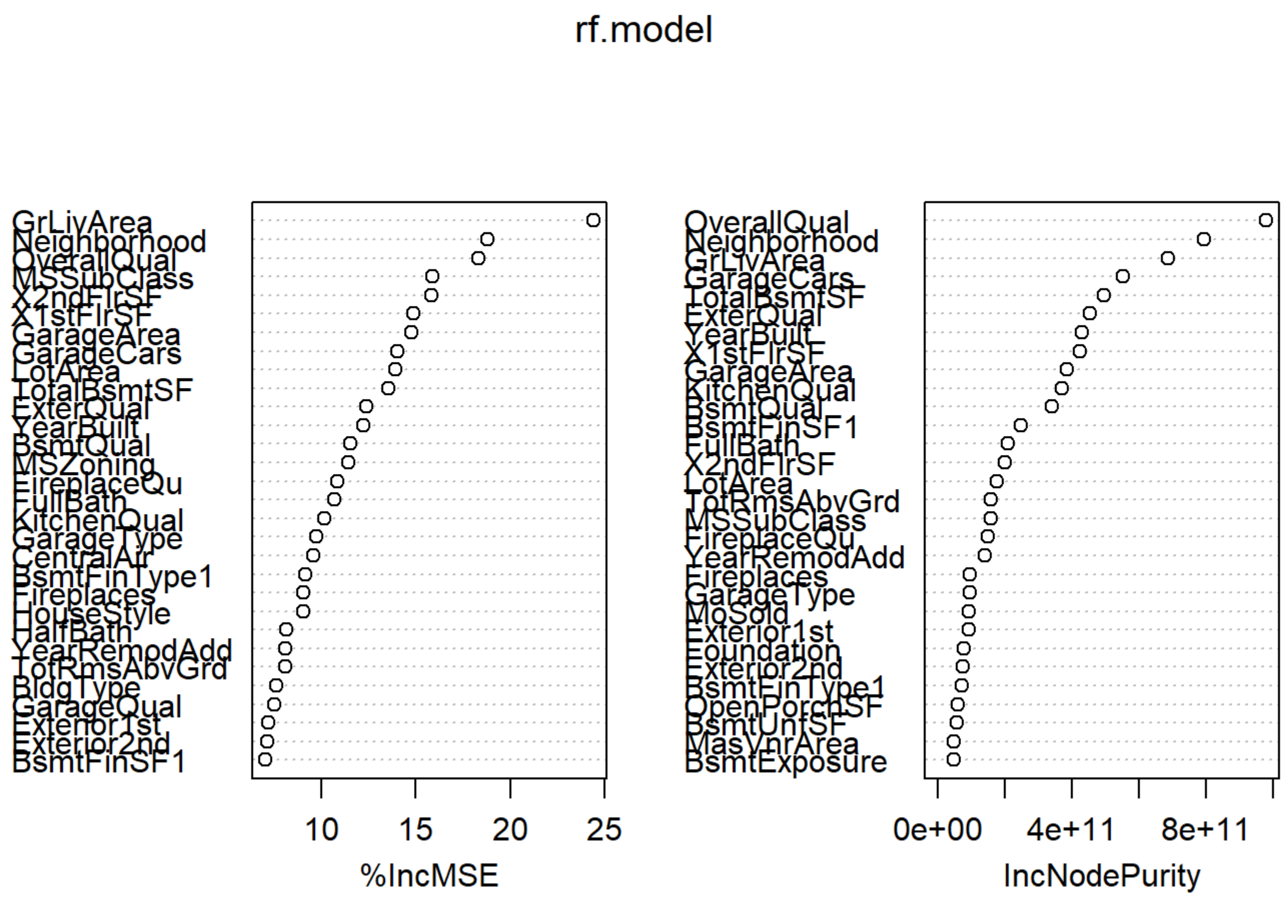
The estimated RMSLE is 0.1401211 for the random forest method; this is fairly close to the true test RMSLE.

The following code fits a random forest model to the data, but doesn't log-transform the response.

```
rf.model <- randomForest(SalePrice~.-Id, data=train, mtry=rf.mtry, importance=T, ntree=500)
```

The results of the random forest model on the non-transformed response are as follows:

```
varImpPlot(rf.model)
```




```
sort(importance(rf.model)[,1], decreasing=T)
```

```
    GrLivArea  Neighborhood   OverallQual    MSSubClass     X2ndFlrSF
 24.407833439  18.786510041  18.298926905  15.858835769  15.815107546
     X1stFlrSF    GarageArea    GarageCars       LotArea   TotalBsmtSF
 14.857228866  14.779432268  14.040723759  13.924158779  13.536590962
     ExterQual     YearBuilt      BsmtQual      MSZoning   FireplaceQu
 12.356967406  12.246369156  11.519464583  11.405106250  10.819013546
      FullBath   KitchenQual    GarageType    CentralAir  BsmtFinType1
 10.657171769  10.173005447   9.749147527   9.559419804   9.131549702
    Fireplaces    HouseStyle      HalfBath  YearRemodAdd  TotRmsAbvGrd
  9.059714866   9.053625326   8.147758708   8.079483513   8.061258953
      BldgType    GarageQual   Exterior1st   Exterior2nd    BsmtFinSF1
  7.603351215   7.505558872   7.181561112   7.123575865   7.038382770
    GarageCond    MasVnrArea    WoodDeckSF    Foundation  BedroomAbvGr
  6.460677847   6.207951652   5.921768083   5.897797472   5.872605922
  BsmtFullBath     HeatingQC    PavedDrive    MasVnrType  KitchenAbvGr
  5.779828089   5.637735665   5.247971352   5.185586252   4.770191362
   OverallCond  BsmtExposure   OpenPorchSF     BsmtUnfSF    Condition1
  4.518228723   4.472455242   4.378727492   3.807177909   3.728498997
      BsmtCond  BsmtFinType2     LandSlope         Fence      LotShape
  3.719308768   3.038352194   2.695272465   2.526889216   2.318786083
      SaleType SaleCondition     RoofStyle     ExterCond   LandContour
  2.291253179   2.167520077   2.014403190   1.801280039   1.742212581
 EnclosedPorch       MiscVal    BsmtFinSF2    Condition2        MoSold
  1.740709008   1.649672187   1.559518710   1.520969687   1.520887242
  BsmtHalfBath  LowQualFinSF         Alley    Functional   ScreenPorch
  1.496359179   1.119864523   1.095481353   1.067982067   0.672632462
   MiscFeature    Electrical     LotConfig        Street        YrSold
  0.412877636   0.411877339   0.182849425  -0.003339891  -0.156259920
    X3SsnPorch      RoofMatl        PoolQC      PoolArea       Heating
 -0.950233261  -1.390621164  -1.594905414  -1.801798405  -2.293417511
```

```
sort(importance(rf.model)[,2], decreasing=T)
```

```
  OverallQual  Neighborhood     GrLivArea    GarageCars   TotalBsmtSF
 979058081116  793109647816  686748045621  553171578513  494374287911
    ExterQual     YearBuilt      X1stFlrSF    GarageArea   KitchenQual
 453572722717  430319702783  423327208688  383970294306  369667452572
     BsmtQual    BsmtFinSF1      FullBath     X2ndFlrSF       LotArea
 339491067555  246300352602  208463968461  200706280239  174186325699
 TotRmsAbvGrd    MSSubClass   FireplaceQu  YearRemodAdd    Fireplaces
 157260420220  157204284778  147911175356  139950841210   94813396787
   GarageType        MoSold   Exterior1st    Foundation   Exterior2nd
  94161165692   91030302503   90913538142   76322181888   72642754718
 BsmtFinType1   OpenPorchSF     BsmtUnfSF    MasVnrArea  BsmtExposure
  72166739714   57669082874   54800115810   48393662203   48194098559
    HeatingQC   OverallCond    WoodDeckSF  BedroomAbvGr      HalfBath
  45509407553   42885189736   42849704259   35960855124   35239254644
   HouseStyle    MasVnrType SaleCondition     RoofStyle  BsmtFullBath
  33078843544   26014860998   22230381576   22097890742   19660950316
     MSZoning      LotShape      SaleType      PoolArea        YrSold
  19658053635   18149440847   17164626271   15629537433   15279879351
    LotConfig      BldgType   LandContour    GarageCond      RoofMatl
  14937851583   14561159582   14302248429   13063225770   12442626411
   CentralAir         Fence    GarageQual   ScreenPorch  BsmtFinType2
  12097751832   11877522500   11559575480   10036877022    9936521971
       PoolQC    Condition1    Functional      BsmtCond    BsmtFinSF2
   9620031262    9195177611    8552821916    8035882410    7555540624
EnclosedPorch     LandSlope  BsmtHalfBath     ExterCond    PavedDrive
   7553914207    7017754923    6916243562    5997453255    5196589926
 KitchenAbvGr         Alley       Heating    Electrical       MiscVal
   4300887731    3077008413    2887161154    2795320287    1124496999
  LowQualFinSF    X3SsnPorch   MiscFeature    Condition2        Street
   1101853833    1095568355     992773684     607914546     209313441
```

In terms of both prediction accuracy `[,1]` and purity `[,2]`, the two three most important variables are `GrLivArea`, `OverallQual` and `Neighborhood` (though not in the same overder for both); this is consistent with the results from the decision trees and bagging. In terms of prediction accuracy, the variable `GrLivArea` is the number one predictor, but it is ranked third for node purity.

The following code creates a prediction using the random forest model.

```
rf.prediction <- predict(rf.model, newdata=test)
write.csv(data.frame(Id=test$Id, SalePrice=rf.prediction), "RandomForest_Prediction.csv", row.na
mes=F)
```

The Kaggle score for this model was **0.15819**. This shows significant improvement over the decision tree method, but not the non-transformed bagging model, and it is worse than the log-transformed response random forest model.

The following code uses cross-validation to predict the error.

```
set.seed(428) # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=10, labels=FALSE) # split data into 10 folds

out <- c()
for (i in 1:10) {
  train.cv <- train[fold.index!=i,]
  test.cv.X <- train[fold.index==i,-77]
  test.cv.y <- train[fold.index==i,77]

  rf.model.cv <- randomForest(SalePrice~.-Id, data=train.cv, mtry=rf.mtry, importance=T, ntree=5
00)
  rf.prediction.cv <- predict(rf.model.cv, newdata=test.cv.X)
  rf.rmsle.cv <- sqrt(mean((log(rf.prediction.cv)-log(test.cv.y))^2))
  out <- c(out, rf.rmsle.cv)
}
mean(out)
```

```
[1] 0.1445637
```

The estimated RMSLE is 0.1445637 for the random forest method on the non-transformed response; this is fairly close to the true test RMSLE.

Boosting

This next section, like the previous, will try to fit the boosting model to both the log-transformed response and the response as-is. It will also choose the parameters (number of trees, interaction depth, and shrinkage) using cross-validation.

```
library(gbm)
```

Using cross-validation, I will check what combination of B, λ, and d results in the smallest estimated test error. I use common values for λ and d, as discussed in class.

```
boost.trees.params <- seq(25, 250, 25)
boost.lambda.params <- c(0.1, 0.01)
boost.depth.params <- c(1, 2, 4, 8)

boost.log.cv.rmsle.array <- array(NA, dim=c(length(boost.trees.params), length(boost.lambda.para
ms), length(boost.depth.params)), dimnames=list(boost.trees.params, boost.lambda.params, boost.d
epth.params))

set.seed(428) # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=10, labels=FALSE) # split data into 10 folds

for (i in 1:length(boost.trees.params)) {
  for (j in 1:length(boost.lambda.params)) {
    for (k in 1:length(boost.depth.params)) {

      out <- c()
      for (l in 1:10) {

        train.cv <- train[fold.index!=l,]
        test.cv.X <- train[fold.index==l,-77]
        test.cv.y <- train[fold.index==l,77]

        tmp.log.model <- gbm(log(SalePrice)~.-Id, data=train.cv, distribution='gaussian', n.tree
s=boost.trees.params[i], interaction.depth=boost.depth.params[k], shrinkage=boost.lambda.params
[j])
        tmp.prediction <- exp(predict(tmp.log.model, newdata=test.cv.X, n.trees=boost.trees.para
ms[i]))
        out[l] <- mean((log(tmp.prediction)-log(test.cv.y))^2)

#         print(paste0("i=",i,",j=",j,",k=",k,",l=",l))
      }
      boost.log.cv.rmsle.array[i,j,k] <- mean(out)
    }
  }
}

which(boost.log.cv.rmsle.array==min(boost.log.cv.rmsle.array), arr.ind=T)
```

```
   dim1 dim2 dim3
25    1    1    4
```

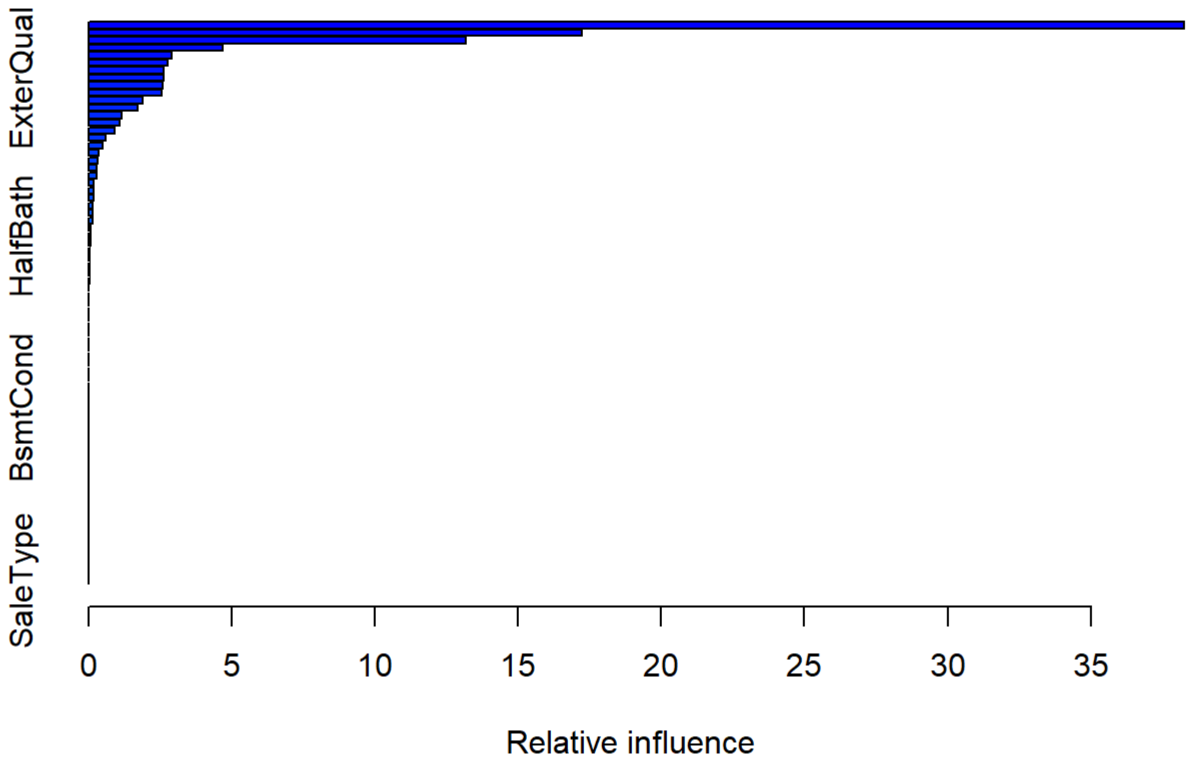
The best is 1,1,4 which is 25 trees, 0.1 shrinkage, and depth of 8. (est. RMSLE is 0.06577063).

The following code creates using these parameter values.

```
boost.log.model <- gbm(log(SalePrice)~.-Id, data=train, distribution='gaussian', n.trees=25, int
eraction.depth=8, shrinkage=0.1)
```

The following code looks at the important variables in the boosting model.

```
summary(boost.log.model)
```



ExterQual
HalfBath
BsmtCond
SaleType
0
5
10
15
20
25
30
35
Relative influence

```
                       var     rel.inf
OverallQual     OverallQual 38.25347319
Neighborhood   Neighborhood 17.22999859
GrLivArea         GrLivArea 13.19123359
KitchenQual     KitchenQual  4.68158201
X1stFlrSF         X1stFlrSF  2.91643242
TotalBsmtSF     TotalBsmtSF  2.76603209
BsmtFinSF1       BsmtFinSF1  2.63552937
ExterQual         ExterQual  2.63501792
GarageArea       GarageArea  2.58183522
GarageCars       GarageCars  2.55178974
MSSubClass       MSSubClass  1.89101920
OverallCond     OverallCond  1.71693349
YearRemodAdd   YearRemodAdd  1.16258282
CentralAir       CentralAir  1.09524442
YearBuilt         YearBuilt  0.89759245
FireplaceQu     FireplaceQu  0.59745792
GarageType       GarageType  0.49614564
BsmtFinType1   BsmtFinType1  0.34801108
GarageCond       GarageCond  0.33537529
LotArea             LotArea  0.29106458
SaleCondition SaleCondition  0.28711105
X2ndFlrSF         X2ndFlrSF  0.19217645
WoodDeckSF       WoodDeckSF  0.19199986
PavedDrive       PavedDrive  0.18773832
TotRmsAbvGrd   TotRmsAbvGrd  0.14767040
MoSold               MoSold  0.14298958
BsmtQual           BsmtQual  0.13800055
Functional       Functional  0.08102707
HalfBath           HalfBath  0.06823329
Condition1       Condition1  0.05815569
OpenPorchSF     OpenPorchSF  0.05295882
Exterior2nd     Exterior2nd  0.05008979
KitchenAbvGr   KitchenAbvGr  0.04982033
BsmtExposure   BsmtExposure  0.03923786
Foundation       Foundation  0.03843995
MSZoning           MSZoning  0.00000000
Street               Street  0.00000000
Alley                 Alley  0.00000000
LotShape           LotShape  0.00000000
LandContour     LandContour  0.00000000
LotConfig         LotConfig  0.00000000
LandSlope         LandSlope  0.00000000
Condition2       Condition2  0.00000000
BldgType           BldgType  0.00000000
HouseStyle       HouseStyle  0.00000000
RoofStyle         RoofStyle  0.00000000
RoofMatl           RoofMatl  0.00000000
Exterior1st     Exterior1st  0.00000000
MasVnrType       MasVnrType  0.00000000
MasVnrArea       MasVnrArea  0.00000000
ExterCond         ExterCond  0.00000000
BsmtCond           BsmtCond  0.00000000
```

```
BsmtFinType2   BsmtFinType2  0.00000000
BsmtFinSF2       BsmtFinSF2  0.00000000
BsmtUnfSF         BsmtUnfSF  0.00000000
Heating             Heating  0.00000000
HeatingQC         HeatingQC  0.00000000
Electrical       Electrical  0.00000000
LowQualFinSF   LowQualFinSF  0.00000000
BsmtFullBath   BsmtFullBath  0.00000000
BsmtHalfBath   BsmtHalfBath  0.00000000
FullBath           FullBath  0.00000000
BedroomAbvGr   BedroomAbvGr  0.00000000
Fireplaces       Fireplaces  0.00000000
GarageQual       GarageQual  0.00000000
EnclosedPorch EnclosedPorch  0.00000000
X3SsnPorch       X3SsnPorch  0.00000000
ScreenPorch     ScreenPorch  0.00000000
PoolArea           PoolArea  0.00000000
PoolQC               PoolQC  0.00000000
Fence                 Fence  0.00000000
MiscFeature     MiscFeature  0.00000000
MiscVal             MiscVal  0.00000000
YrSold               YrSold  0.00000000
SaleType           SaleType  0.00000000
```

The three most important variables in this model are `OverallQual` , `GrLivArea` , and `Neighborhood` ; this is consistent with the other models.

The following code makes the prediction using the model.

```
boost.log.prediction <- exp(predict(boost.log.model, newdata=test, n.trees=25))
write.csv(data.frame(Id=test$Id, SalePrice=boost.log.prediction), "Boosting_Log_Prediction.csv",
 row.names=F)
```

The Kaggle score for this model was **0.15386**. This shows improvement over the decision trees and the bagging model for the log-transformed response; it was slightly worse than the random forest model.

Using cross-validation, I will check what combination of B, λ, and d results in the smallest estimated test error. I use common values for λ and d, as discussed in class.

```
boost.trees.params <- seq(25, 250, 25)
boost.lambda.params <- c(0.1, 0.01)
boost.depth.params <- c(1, 2, 4, 8)

boost.cv.rmsle.array <- array(NA, dim=c(length(boost.trees.params), length(boost.lambda.params),
 length(boost.depth.params)), dimnames=list(boost.trees.params, boost.lambda.params, boost.dept
h.params))

set.seed(428) # consistency of k-fold validation breaks
fold.index <- cut(sample(1:nrow(train)), breaks=10, labels=FALSE) # split data into 10 folds

for (i in 1:length(boost.trees.params)) {
  for (j in 1:length(boost.lambda.params)) {
    for (k in 1:length(boost.depth.params)) {

      out <- c()
      for (l in 1:10) {

        train.cv <- train[fold.index!=l,]
        test.cv.X <- train[fold.index==l,-77]
        test.cv.y <- train[fold.index==l,77]

        tmp.model <- gbm(SalePrice~.-Id, data=train.cv, distribution='gaussian', n.trees=boost.t
rees.params[i], interaction.depth=boost.depth.params[k], shrinkage=boost.lambda.params[j])
        tmp.prediction <- predict(tmp.model, newdata=test.cv.X, n.trees=boost.trees.params[i])
        out[l] <- mean((log(tmp.prediction)-log(test.cv.y))^2)

       # print(paste0("i=",i,",j=",j,",k=",k,",l=",l))
      }
      boost.cv.rmsle.array[i,j,k] <- mean(out)
    }
  }
}

which(boost.cv.rmsle.array==min(boost.cv.rmsle.array), arr.ind=T)
```

```
    dim1 dim2 dim3
250   10    2    4
```

```
min(boost.cv.rmsle.array)
```

```
[1] 0.06348275
```

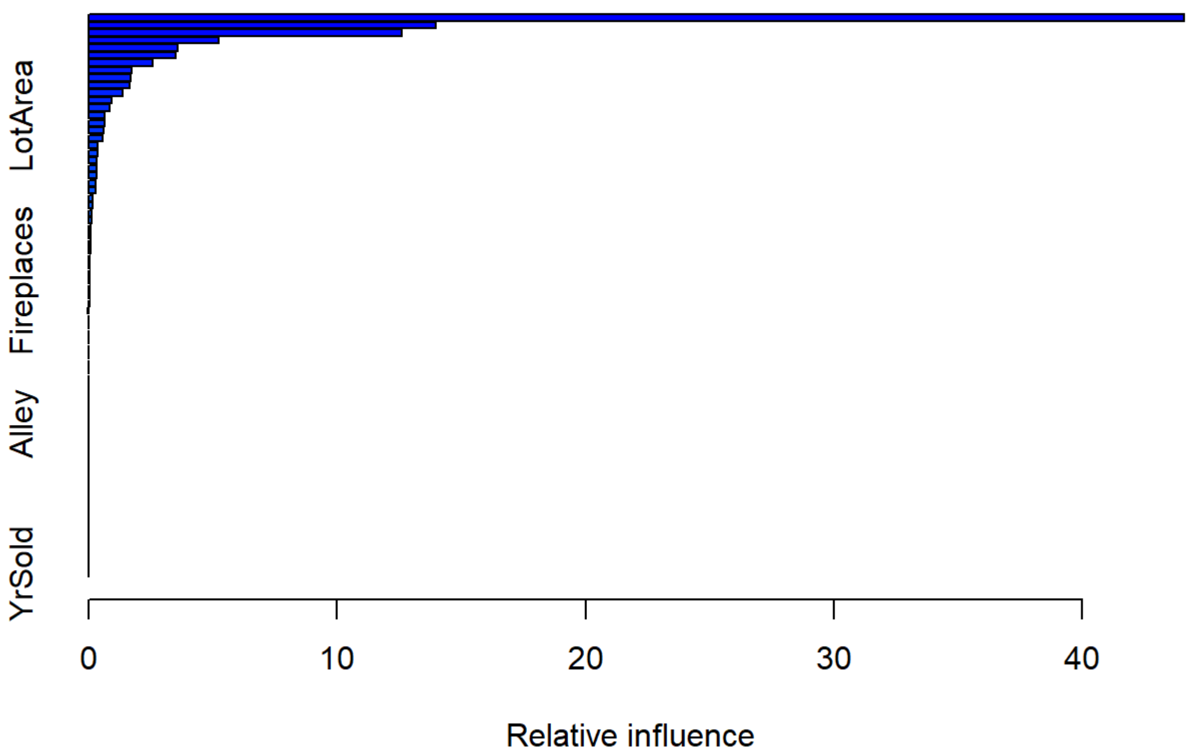
The best is 10,2,4 which is 250 trees, 0.01 shrinkage, and depth of 8. (est. RMSLE is 0.06348275).

The following code creates using these parameter values.

```
boost.model <- gbm(SalePrice~.-Id, data=train, distribution='gaussian', n.trees=250, interactio
n.depth=8, shrinkage=0.01)
```

The following code looks at the important variables in the boosting model.

```
summary(boost.model)
```



```
                         var      rel.inf
OverallQual      OverallQual 44.096681012
GrLivArea          GrLivArea 13.994997808
Neighborhood    Neighborhood 12.600573523
TotalBsmtSF      TotalBsmtSF  5.245596080
BsmtFinSF1        BsmtFinSF1  3.584458928
GarageCars        GarageCars  3.527772055
X1stFlrSF          X1stFlrSF  2.598287178
BsmtQual            BsmtQual  1.741792078
KitchenQual      KitchenQual  1.715727798
GarageArea        GarageArea  1.661308979
ExterQual          ExterQual  1.362820113
MSSubClass        MSSubClass  0.947343005
TotRmsAbvGrd    TotRmsAbvGrd  0.860829097
LotArea              LotArea  0.645812516
X2ndFlrSF          X2ndFlrSF  0.637333283
FullBath            FullBath  0.628645227
YearRemodAdd    YearRemodAdd  0.550973830
MoSold                MoSold  0.384643990
YearBuilt          YearBuilt  0.371673520
GarageType        GarageType  0.341880711
FireplaceQu      FireplaceQu  0.334185281
OverallCond      OverallCond  0.308187638
CentralAir        CentralAir  0.298608622
BsmtExposure    BsmtExposure  0.279018063
Exterior2nd      Exterior2nd  0.179847940
SaleType            SaleType  0.176825626
BsmtFinType1    BsmtFinType1  0.137084260
Exterior1st      Exterior1st  0.122709233
OpenPorchSF      OpenPorchSF  0.079142138
SaleCondition SaleCondition  0.069946226
GarageQual        GarageQual  0.066298775
HalfBath            HalfBath  0.065411616
BsmtFullBath    BsmtFullBath  0.059522641
MSZoning            MSZoning  0.032786493
Condition1        Condition1  0.030806597
Fireplaces        Fireplaces  0.030385378
WoodDeckSF        WoodDeckSF  0.027607354
BedroomAbvGr    BedroomAbvGr  0.026753123
BsmtCond            BsmtCond  0.026192187
HeatingQC          HeatingQC  0.024527728
GarageCond        GarageCond  0.019997262
BsmtUnfSF          BsmtUnfSF  0.019078693
ExterCond          ExterCond  0.013392422
ScreenPorch      ScreenPorch  0.012673556
KitchenAbvGr    KitchenAbvGr  0.012267710
MasVnrArea        MasVnrArea  0.011832283
HouseStyle        HouseStyle  0.006897067
LotShape            LotShape  0.006656396
MasVnrType        MasVnrType  0.006320278
LandContour      LandContour  0.005283731
Functional        Functional  0.004133535
LandSlope          LandSlope  0.003465416
```

```
LotConfig         LotConfig  0.003004004
Street               Street  0.000000000
Alley                 Alley  0.000000000
Condition2       Condition2  0.000000000
BldgType           BldgType  0.000000000
RoofStyle         RoofStyle  0.000000000
RoofMatl           RoofMatl  0.000000000
Foundation       Foundation  0.000000000
BsmtFinType2   BsmtFinType2  0.000000000
BsmtFinSF2       BsmtFinSF2  0.000000000
Heating             Heating  0.000000000
Electrical       Electrical  0.000000000
LowQualFinSF   LowQualFinSF  0.000000000
BsmtHalfBath   BsmtHalfBath  0.000000000
PavedDrive       PavedDrive  0.000000000
EnclosedPorch EnclosedPorch  0.000000000
X3SsnPorch       X3SsnPorch  0.000000000
PoolArea           PoolArea  0.000000000
PoolQC               PoolQC  0.000000000
Fence                 Fence  0.000000000
MiscFeature     MiscFeature  0.000000000
MiscVal             MiscVal  0.000000000
YrSold               YrSold  0.000000000
```

The three most important variables in this model are `OverallQual` , `GrLivArea` , and `Neighborhood` ; this is consistent with the other models.

The following code makes the prediction using the model.

```
boost.prediction <- predict(boost.model, newdata=test, n.trees=250)
write.csv(data.frame(Id=test$Id, SalePrice=boost.prediction), "Boosting_Prediction.csv", row.nam
es=F)
```

The Kaggle score for this model was **0.17266**. This shows improvement over the decision tree, but not over anything else.