Ensembles of Trees: Predicting Housing Prices

Suzana Iacob

24/10/2019

This project predicts house prices based on house characteristics. The dataset description can be found here: https://amstat.tandfonline.com/doi/abs/10.1080/10691898.2011.11889627 (https://amstat.tandfonline.com/doi/abs/10.1080/10691898.2011.11889627)

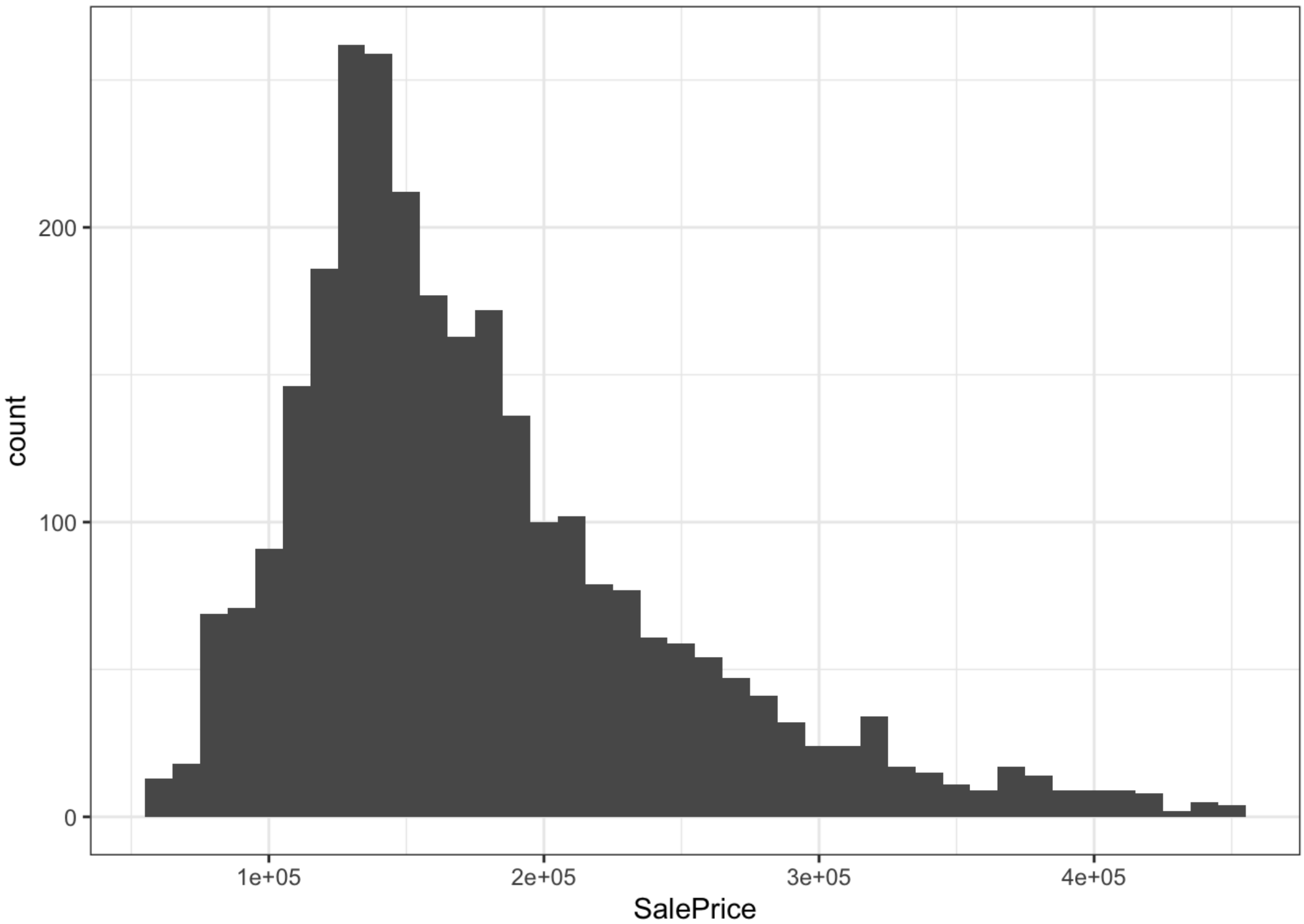
Data Exploration

```
ames = read.csv("ames.csv")
ames$SalePrice = as.numeric(ames$SalePrice)
```

We are analyzing the Ames Housing dataset, containing 74 predictors, intending to predict SalePrice. Our data includes house information, such as the size of the lot, number of rooms, year of construction, building materials and more.

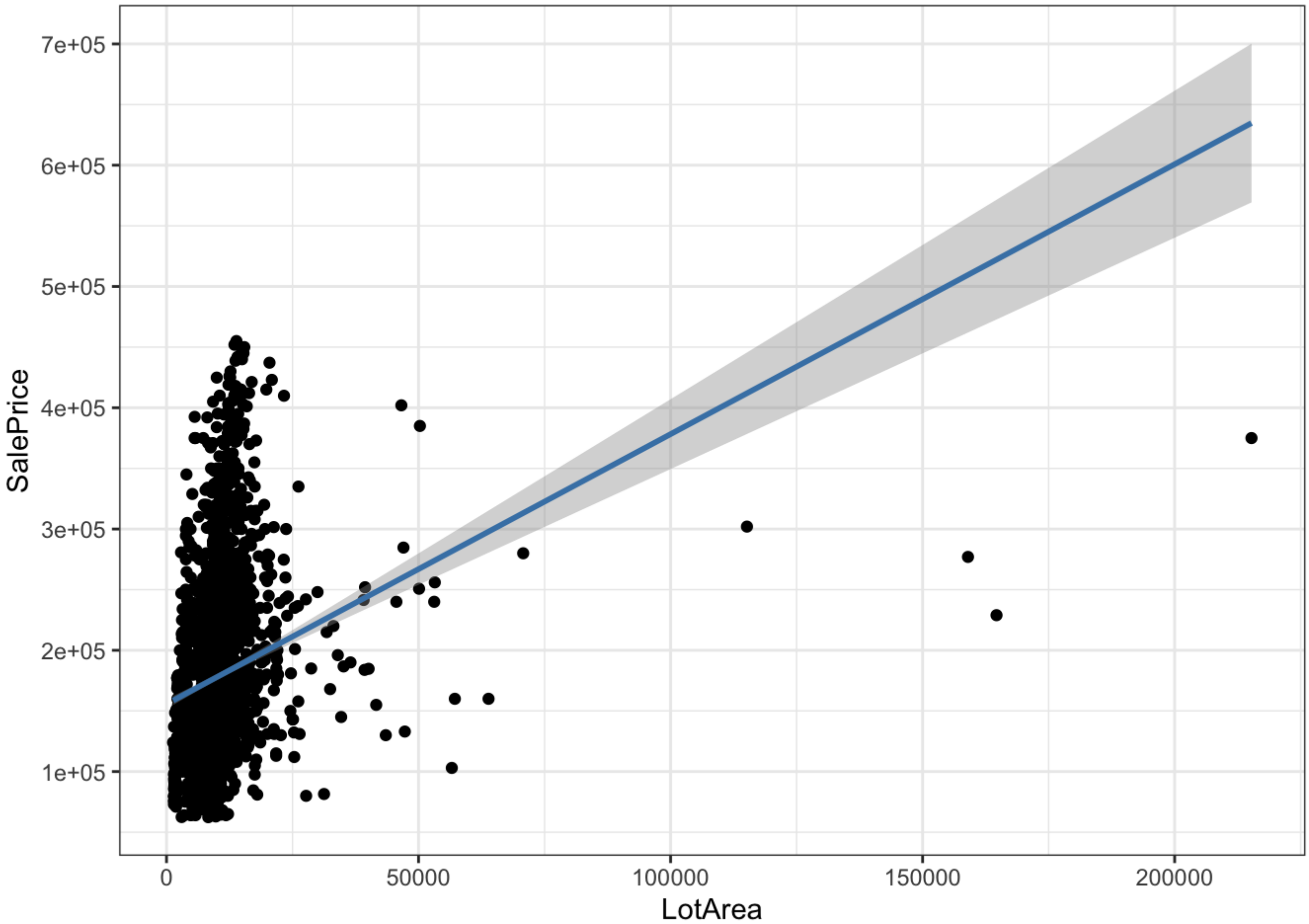
Our first step is investigating the target variable. We notice that SalePrice has a long tail to the right, meaning that there are few very expensive houses and the majority of houses revolve around $100,000-$300,000.

```
ggplot(data=ames[!is.na(ames$SalePrice),], aes(x=SalePrice)) +
        geom_histogram(binwidth = 10000) +
        scale_x_continuous(breaks= seq(0, 1000000, by=100000)) + theme_bw()
```



Secondly, we can inspect some of the predictors and see which ones influence Sale Price and to what extent. We use our domain knowledge and we can assume that the size of the property(LotArea) will be a strong predictor for price.

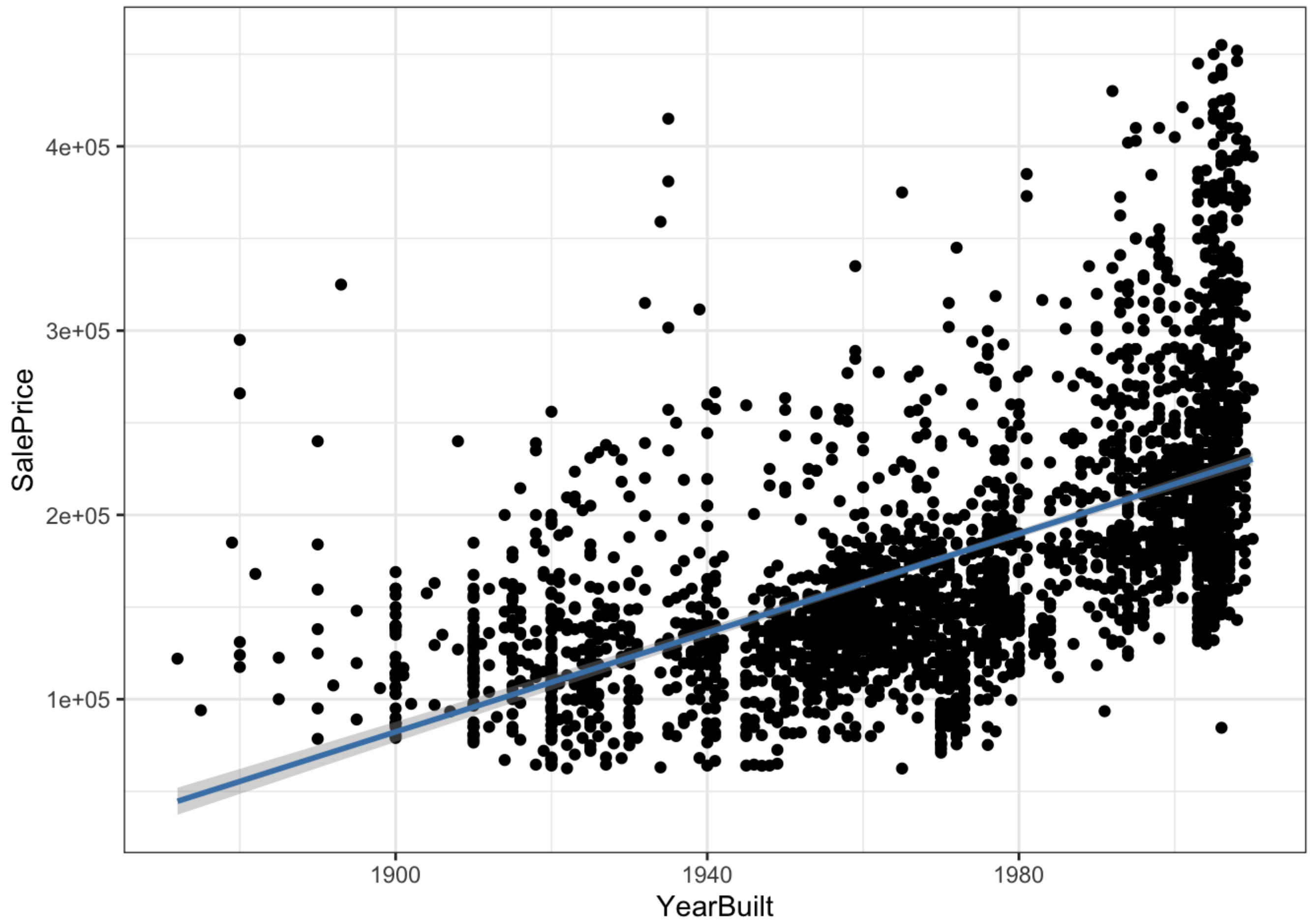
```
ggplot(data=ames[!is.na(ames$SalePrice),], aes(x=LotArea, y=SalePrice))+
        geom_point() + geom_smooth(method = "lm",color="steelblue", aes(group=1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=100000)) + theme_bw()
```



From the graph above we see that SalePrice indeed shows an upward trend with LotArea, but we have a number of outliers outside of the critical mass. Moreover, there are several very large properties at a relatively small price, and the fitted line has a large variance at the end (heteroscedasticity). This leads us to believe there are other factors aside from/instead of LotArea predicting the price.

We next look at the YearBuilt, assuming that older houses will be less expensive, and based on the plot below this is a correct assumption.

```
ggplot(data=ames[!is.na(ames$SalePrice),], aes(x=YearBuilt, y=SalePrice))+
        geom_point() + geom_smooth(method = "lm",color="steelblue", aes(group=1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=100000)) + theme_bw()
```



Unfortunately, we have 74 possible predictors and investigating them manually is extremely time-consuming. In addition, we could have multicollinearity relationships, meaning that some of the predictors are related to each other (in fact we can assume this to be the case since we have several measures of the number of rooms for example).

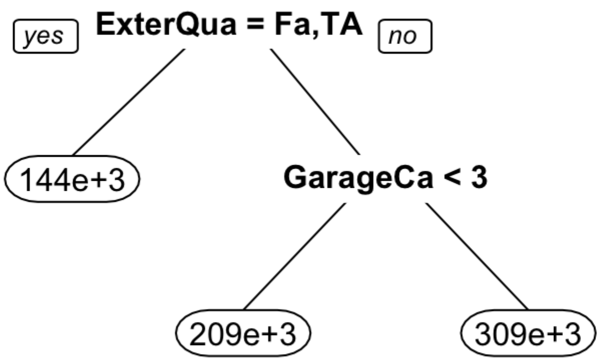
Predictive analytics can help us create models that give accurate predictions and tell us which predictors are significant.

Regression Trees

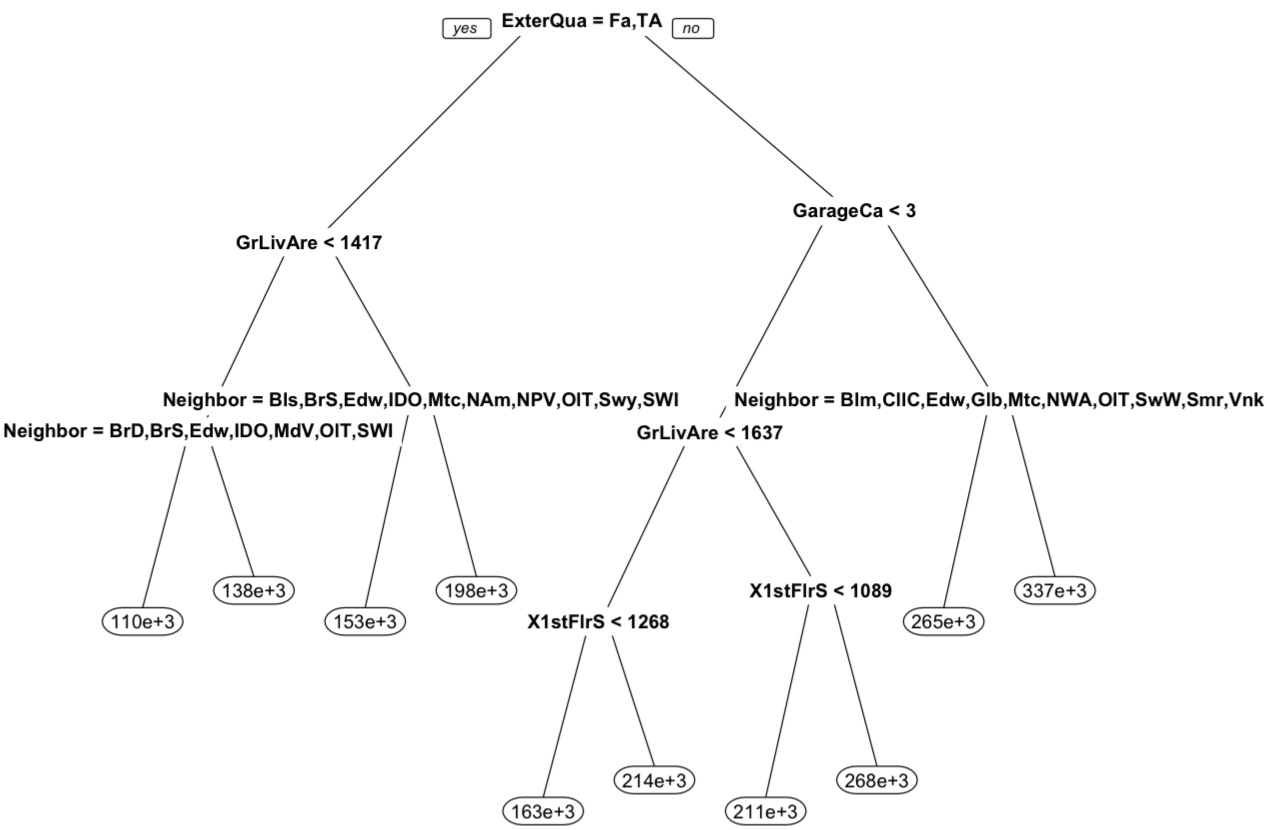
We build models with different levels of complexity.

```
set.seed(657)
split = createDataPartition(ames$SalePrice, p = 0.65, list = FALSE)
ames.train = ames[split,]
ames.test = ames[-split,]
amesTree = rpart(SalePrice ~ ., data=ames.train,cp=0.1, minbucket=25)
amesTree2 = rpart(SalePrice ~ .,data=ames.train, cp=0.01, minbucket=25)
amesTree3 = rpart(SalePrice ~ .,data=ames.train, cp=0.001, minbucket=25)
```

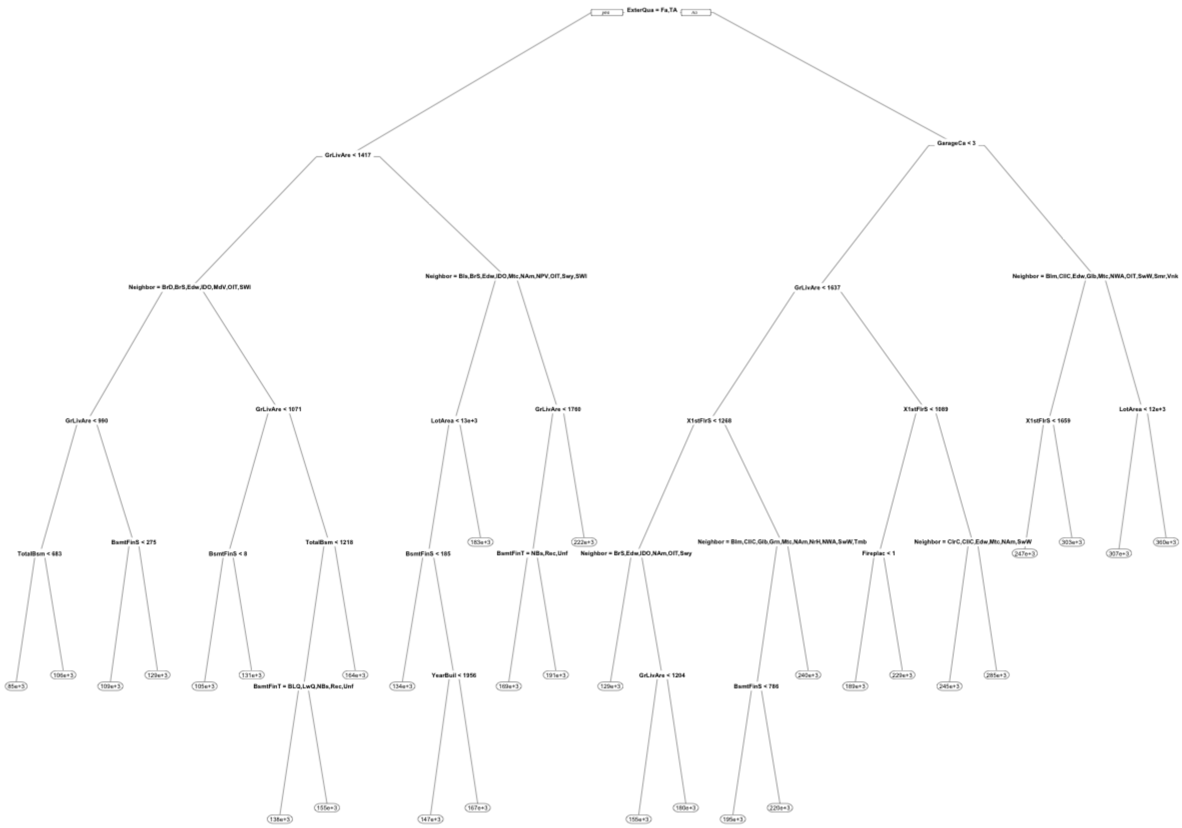
```
prp(amesTree)
```




```
prp(amesTree2)
```



```
prp(amesTree3)
```



We tried 3 trees with different CP values. Our first one is very simple, and the last one is very complex and not too interpretable. By the nature of CART the first splits are always the same and we note the most important variables as: * ExterQual (exterior material quality) * GrLivArea (ground living area) * GarageCars (size of garage) * Neighborhood * X1stFlrSF (first floor area)

This somehow matched the intuition from question a when we assumed size would influence price, but we were looking at a different variable. YearBuilt is not among the top 5, but we see that the last tree used YearBuilt to split.

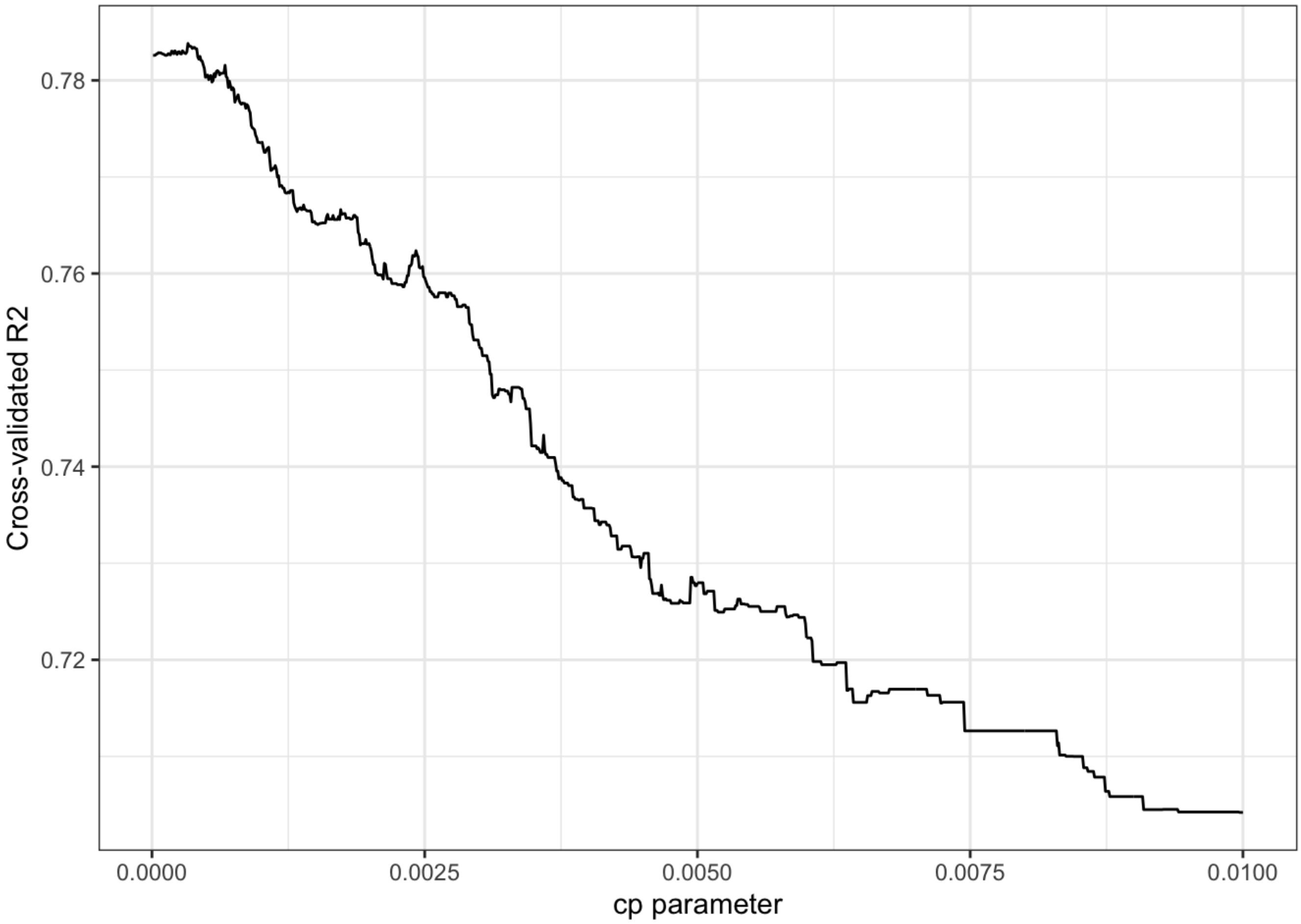
Cross Validation

We now perform cross-validation to select the best tree parameter cp.

```
cpVals <- data.frame(.cp = seq(.00001, .01, by=.00001))

set.seed(123)
cpCV = train(SalePrice~.,
             trControl=trainControl(method="cv",number=10), data=ames.train,method="r
part",minbucket=50,
             tuneGrid=cpVals, metric="Rsquared", maximize=TRUE)
```

```
ggplot(cpCV$results, aes(x=cp, y=Rsquared)) +
  geom_line() +
  theme_bw() +
  xlab("cp parameter") +
  ylab("Cross-validated R2")
```



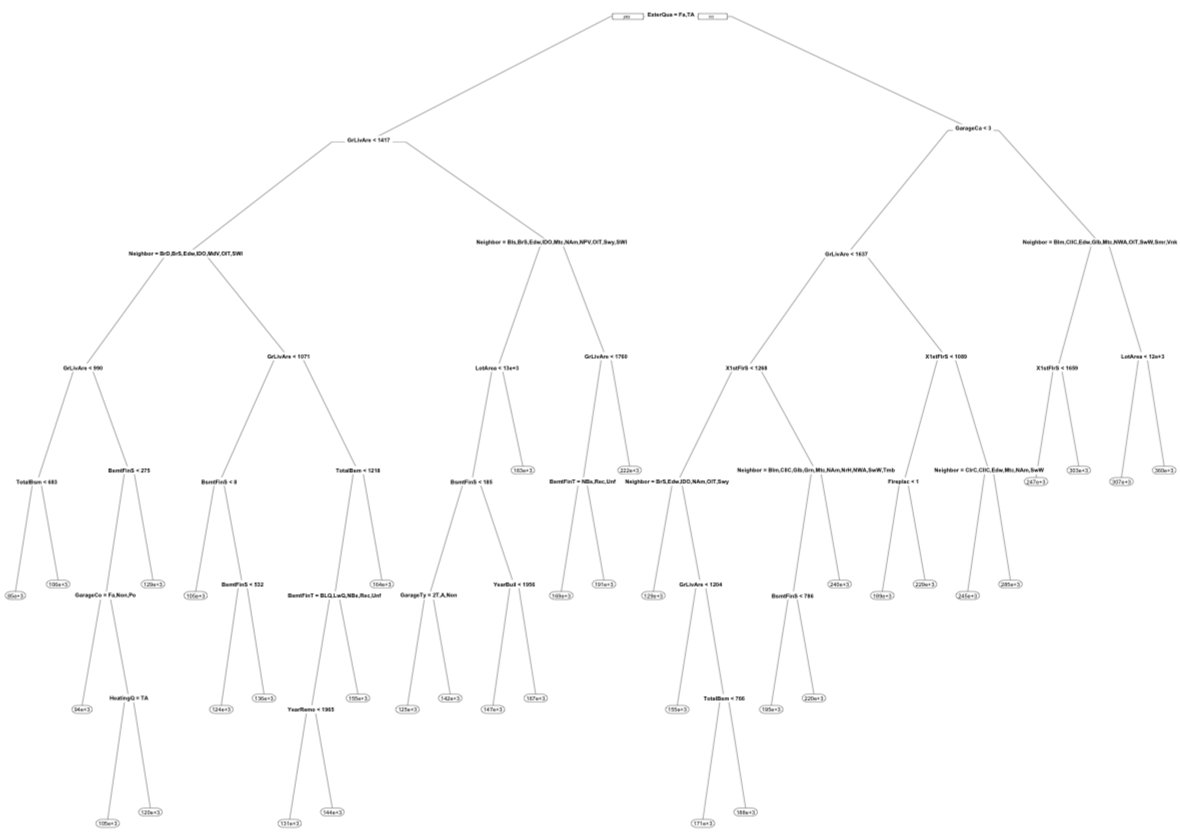
We notice that the best cp is quite low, indeed the best value is 0.00033.

```
best.cp = cpCV$bestTune
print(best.cp)
```

```
##         cp
## 33 0.00033
```

We now re-fit the tree with this value. This is a relatively deep and less interpretable tree, and we see the same variables as before, plus a few more. We have a total of 34 splits.

```
treeFinal <- rpart(SalePrice ~ ., data=ames.train, minbucket = 25, cp=best.cp)
prp(treeFinal)
```



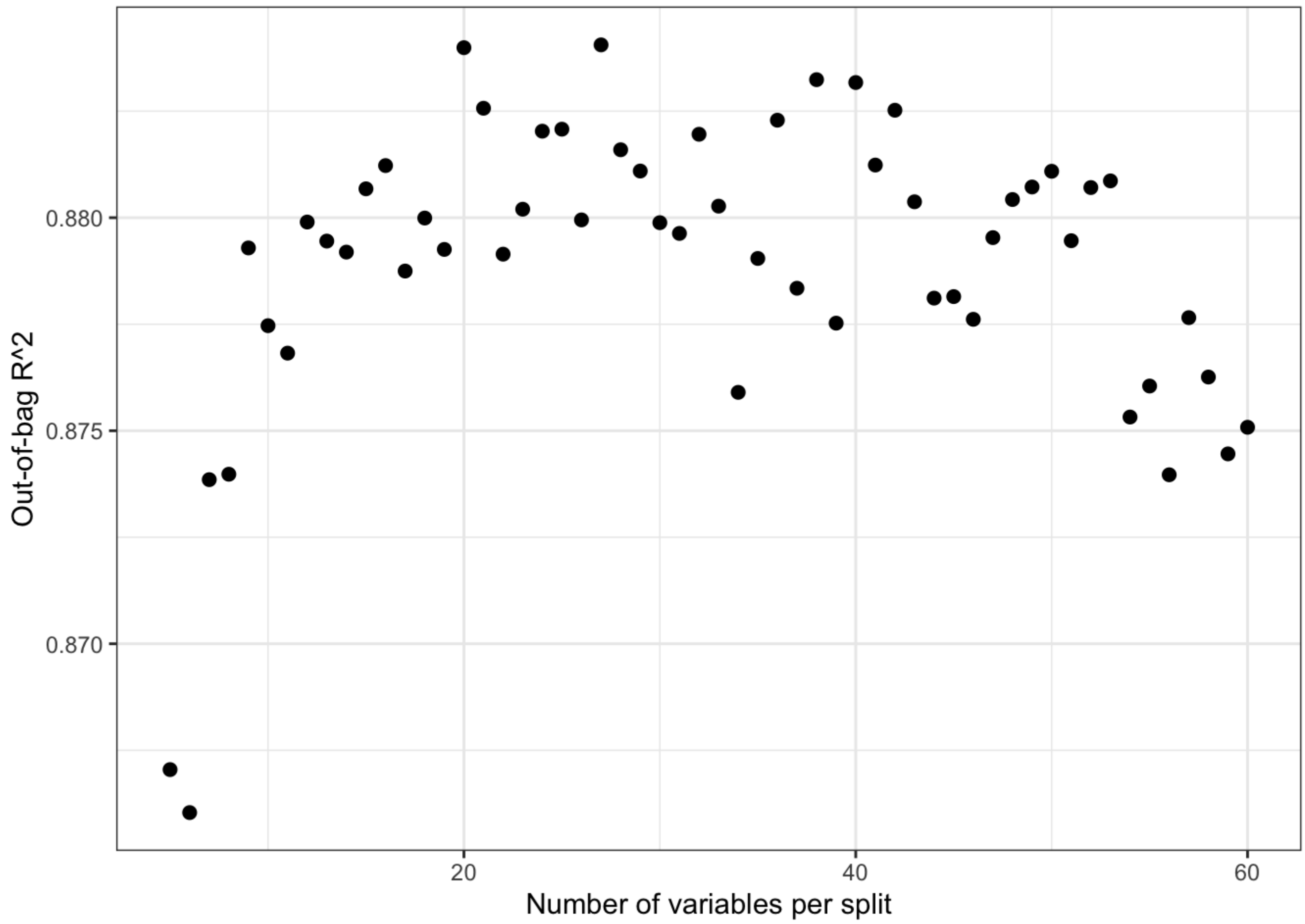
Random Forest

```
rf = randomForest(SalePrice ~ ., data=ames.train, ntree=80,mtry=5,nodesize=25)
```

We use out-of-bag predictions to select the best value for the mtry hyperparameter.

```
set.seed(123)
train.rf.oob <- train(x = ames.train %>% select(-SalePrice),
                      y = ames.train$SalePrice,
                      method="rf",
                      ntree=80,
                      nodesize=25,
                      tuneGrid=data.frame(mtry=5:60),
                      trControl=trainControl(method="oob"))
```

```
ggplot(train.rf.oob$results, aes(x=mtry, y=Rsquared)) +
  geom_point(size=2) +
  theme_bw() +
  xlab("Number of variables per split") +
  ylab("Out-of-bag R^2")
```



The best mtry value is 27, so this is how many variables we check at each split (from the 74). We see that a very small or very large mtry has a poorer performance.

```
best.mtry <- train.rf.oob$bestTune[[1]]
best.mtry
```

```
## [1] 27
```

```
rfFinal = randomForest(SalePrice~., data=ames.train, mtry=train.rf.oob$bestTune[[1]])
importance.rf <- data.frame(imp=importance(rfFinal))
importance.rf <- data.frame(var = rownames(importance.rf),
                            imp = importance.rf$IncNodePurity)
head(importance.rf[order(importance.rf$imp,decreasing = T),], n=10)
```

	var <fctr>	imp <dbl>
10	Neighborhood	1.564533e+12
23	ExterQual	1.318221e+12
42	GrLivArea	1.212239e+12
57	GarageCars	7.674374e+11
15	YearBuilt	7.078959e+11
58	GarageArea	5.108492e+11
34	TotalBsmtSF	4.937322e+11

	var <fctr>	imp <dbl>
39	X1stFlrSF	3.987153e+11
49	KitchenQual	2.863869e+11
26	BsmtQual	2.259487e+11

1-10 of 10 rows

Here we looked at the most important variables, which include **Neighborhood, ExterQual, GrLivArea, GarageCars, and YearBuilt**, very similar to what CART gave.

Boosted Trees

We finally construct boosted trees, and for each we report the 10 top variables, which also include **Neighborhood, ExterQual, GrLivArea, GarageCars, YearBuilt, TotalBsmtSF**

```
boost.mod = gbm(SalePrice~.,data=ames.train,distribution = "gaussian",n.minobsinnode
 = 10,
                n.trees=2000, shrinkage=0.01, interaction.depth=5)
boost.mod2 = gbm(SalePrice~.,data=ames.train,distribution = "gaussian",n.minobsinnode
= 10,
                 n.trees=5000, shrinkage=0.1, interaction.depth=3)
boost.mod3 = gbm(SalePrice~.,data=ames.train,distribution = "gaussian",n.minobsinnode
= 10,
                 n.trees=3000, shrinkage=0.05, interaction.depth=6)
```

```
influence1 = summary(boost.mod, plotit=FALSE)
head(influence1, n=10)
```

	var <fctr>	rel.inf <dbl>
Neighborhood	Neighborhood	22.762723
GrLivArea	GrLivArea	17.846192
ExterQual	ExterQual	12.693063
TotalBsmtSF	TotalBsmtSF	7.129473
YearBuilt	YearBuilt	4.494242
GarageCars	GarageCars	4.254481
GarageArea	GarageArea	4.239953
KitchenQual	KitchenQual	4.010216
X1stFlrSF	X1stFlrSF	3.942274
BsmtFinSF1	BsmtFinSF1	2.762263

1-10 of 10 rows

```
influence2 = summary(boost.mod2, plotit=FALSE)
head(influence2, n=10)
```

	var <fctr>	rel.inf <dbl>
Neighborhood	Neighborhood	24.177076
GrLivArea	GrLivArea	16.696393
ExterQual	ExterQual	10.136493
TotalBsmtSF	TotalBsmtSF	8.448754
GarageArea	GarageArea	5.112630
GarageCars	GarageCars	3.555177
YearBuilt	YearBuilt	3.169584
X1stFlrSF	X1stFlrSF	2.735662
BsmtFinSF1	BsmtFinSF1	2.701574
FireplaceQu	FireplaceQu	2.152098

1-10 of 10 rows

```
influence3 = summary(boost.mod3, plotit=FALSE)
head(influence3, n=10)
```

	var <fctr>	rel.inf <dbl>
Neighborhood	Neighborhood	22.532108
GrLivArea	GrLivArea	14.327299
ExterQual	ExterQual	13.938114
TotalBsmtSF	TotalBsmtSF	6.521293
GarageArea	GarageArea	5.042572
GarageCars	GarageCars	4.451863
X1stFlrSF	X1stFlrSF	4.249857
KitchenQual	KitchenQual	3.419335
BsmtFinSF1	BsmtFinSF1	3.159695
BsmtQual	BsmtQual	2.094534

1-10 of 10 rows

oosted trees are formed of many weak learners, and they give very accurate results, as we will see. The variable importance is very similar to before, but the trees are very computationally intensive. We did not perform cross validation, we fit several numbers of trees (2000, 5000, 3000) of varying depths (5,3,6).

Models evaluation

We first report in-sample and out of sample metrics.

```
SSTTrain = sum((ames.train$SalePrice - mean(ames.train$SalePrice))^2)
SSTTest = sum((ames.test$SalePrice - mean(ames.train$SalePrice))^2)
print(SSTTrain)
```

```
## [1] 9.329652e+12
```

```
print(SSTTrain)
```

```
## [1] 9.329652e+12
```

```
PredictTrain.amesTree = predict(amesTree, newdata = ames.train)
PredictTest.amesTree = predict(amesTree, newdata = ames.test)
SSETrain = sum((PredictTrain.amesTree - ames.train$SalePrice)^2)
SSETest = sum((PredictTest.amesTree - ames.test$SalePrice)^2)
R2.amesTree <- 1 - SSETrain/SSTTrain
OSR2.amesTree <- 1 - SSETest/SSTTest
MAE.amesTree <- MAE(PredictTrain.amesTree, ames.train$SalePrice)
OSMAE.amesTree <- MAE(PredictTest.amesTree, ames.test$SalePrice)
RMSE.amesTree <- RMSE(PredictTrain.amesTree, ames.train$SalePrice)
OSRMSE.amesTree <- RMSE(PredictTest.amesTree, ames.test$SalePrice)

PredictTrain.amesTree2 = predict(amesTree2, newdata = ames.train)
PredictTest.amesTree2 = predict(amesTree2, newdata = ames.test)
SSETrain = sum((PredictTrain.amesTree2 - ames.train$SalePrice)^2)
SSETest = sum((PredictTest.amesTree2 - ames.test$SalePrice)^2)
R2.amesTree2 <- 1 - SSETrain/SSTTrain
OSR2.amesTree2 <- 1 - SSETest/SSTTest
MAE.amesTree2 <- MAE(PredictTrain.amesTree2, ames.train$SalePrice)
OSMAE.amesTree2 <- MAE(PredictTest.amesTree2, ames.test$SalePrice)
RMSE.amesTree2 <- RMSE(PredictTrain.amesTree2, ames.train$SalePrice)
OSRMSE.amesTree2 <- RMSE(PredictTest.amesTree2, ames.test$SalePrice)

PredictTrain.amesTree3 = predict(amesTree3, newdata = ames.train)
PredictTest.amesTree3 = predict(amesTree3, newdata = ames.test)
SSETrain = sum((PredictTrain.amesTree3 - ames.train$SalePrice)^2)
SSETest = sum((PredictTest.amesTree3 - ames.test$SalePrice)^2)
R2.amesTree3 <- 1 - SSETrain/SSTTrain
OSR2.amesTree3 <- 1 - SSETest/SSTTest
MAE.amesTree3 <- MAE(PredictTrain.amesTree3, ames.train$SalePrice)
OSMAE.amesTree3 <- MAE(PredictTest.amesTree3, ames.test$SalePrice)
RMSE.amesTree3 <- RMSE(PredictTrain.amesTree3, ames.train$SalePrice)
OSRMSE.amesTree3 <- RMSE(PredictTest.amesTree3, ames.test$SalePrice)

PredictTrain.treeFinal = predict(treeFinal, newdata = ames.train)
PredictTest.treeFinal = predict(treeFinal, newdata = ames.test)
SSETrain = sum((PredictTrain.treeFinal - ames.train$SalePrice)^2)
SSETest = sum((PredictTest.treeFinal - ames.test$SalePrice)^2)
R2.treeFinal <- 1 - SSETrain/SSTTrain
OSR2.treeFinal <- 1 - SSETest/SSTTest
MAE.treeFinal <- MAE(PredictTrain.treeFinal, ames.train$SalePrice)
OSMAE.treeFinal <- MAE(PredictTest.treeFinal, ames.test$SalePrice)
RMSE.treeFinal <- RMSE(PredictTrain.treeFinal, ames.train$SalePrice)
OSRMSE.treeFinal <- RMSE(PredictTest.treeFinal, ames.test$SalePrice)

PredictTrain.rfFinal = predict(rfFinal, newdata = ames.train)
PredictTest.rfFinal = predict(rfFinal, newdata = ames.test)
SSETrain = sum((PredictTrain.rfFinal - ames.train$SalePrice)^2)
SSETest = sum((PredictTest.rfFinal - ames.test$SalePrice)^2)
R2.rfFinal <- 1 - SSETrain/SSTTrain
OSR2.rfFinal <- 1 - SSETest/SSTTest
MAE.rfFinal <- MAE(PredictTrain.rfFinal, ames.train$SalePrice)
OSMAE.rfFinal <- MAE(PredictTest.rfFinal, ames.test$SalePrice)
RMSE.rfFinal <- RMSE(PredictTrain.rfFinal, ames.train$SalePrice)
OSRMSE.rfFinal <- RMSE(PredictTest.rfFinal, ames.test$SalePrice)

PredictTrain.boost.mod = predict(boost.mod, newdata = ames.train, n.trees=2000)
PredictTest.boost.mod = predict(boost.mod, newdata = ames.test, n.trees=2000)
```

```
SSETrain = sum((PredictTrain.boost.mod - ames.train$SalePrice)^2)
SSETest = sum((PredictTest.boost.mod - ames.test$SalePrice)^2)
R2.boost.mod <- 1 - SSETrain/SSTTrain
OSR2.boost.mod <- 1 - SSETest/SSTTest
MAE.boost.mod <- MAE(PredictTrain.boost.mod, ames.train$SalePrice)
OSMAE.boost.mod <- MAE(PredictTest.boost.mod, ames.test$SalePrice)
RMSE.boost.mod <- RMSE(PredictTrain.boost.mod, ames.train$SalePrice)
OSRMSE.boost.mod <- RMSE(PredictTest.boost.mod, ames.test$SalePrice)

PredictTrain.boost.mod2 = predict(boost.mod2, newdata = ames.train, n.trees=5000)
PredictTest.boost.mod2 = predict(boost.mod2, newdata = ames.test, n.trees=5000)
SSETrain = sum((PredictTrain.boost.mod2 - ames.train$SalePrice)^2)
SSETest = sum((PredictTest.boost.mod2 - ames.test$SalePrice)^2)
R2.boost.mod2 <- 1 - SSETrain/SSTTrain
OSR2.boost.mod2 <- 1 - SSETest/SSTTest
MAE.boost.mod2 <- MAE(PredictTrain.boost.mod2, ames.train$SalePrice)
OSMAE.boost.mod2 <- MAE(PredictTest.boost.mod2, ames.test$SalePrice)
RMSE.boost.mod2 <- RMSE(PredictTrain.boost.mod2, ames.train$SalePrice)
OSRMSE.boost.mod2 <- RMSE(PredictTest.boost.mod2, ames.test$SalePrice)

PredictTrain.boost.mod3 = predict(boost.mod3, newdata = ames.train, n.trees=3000)
PredictTest.boost.mod3 = predict(boost.mod3, newdata = ames.test, n.trees=3000)
SSETrain = sum((PredictTrain.boost.mod3 - ames.train$SalePrice)^2)
SSETest = sum((PredictTest.boost.mod3 - ames.test$SalePrice)^2)
R2.boost.mod3 <- 1 - SSETrain/SSTTrain
OSR2.boost.mod3 <- 1 - SSETest/SSTTest
MAE.boost.mod3 <- MAE(PredictTrain.boost.mod3, ames.train$SalePrice)
OSMAE.boost.mod3 <- MAE(PredictTest.boost.mod3, ames.test$SalePrice)
RMSE.boost.mod3 <- RMSE(PredictTrain.boost.mod3, ames.train$SalePrice)
OSRMSE.boost.mod3 <- RMSE(PredictTest.boost.mod3, ames.test$SalePrice)
```

ModelName <fctr>	R2 <dbl>	OSR2 <dbl>	MAE <dbl>	OSMAE <dbl>	RMSE <dbl>	OSRMSE <dbl>
Tree1	0.5564031	0.5396362	35495.938	34714.27	47349.033	46416.87
Tree2	0.7548626	0.7181315	25299.279	26622.06	35198.297	36320.23
Tree3	0.8255297	0.7648019	20447.317	23279.03	29694.612	33177.39
TreeFin	0.8293725	0.7660380	20054.179	23293.67	29365.771	33090.10
RF	0.9815866	0.8954038	6189.379	14996.63	9646.808	22125.00
Boost1	0.9659768	0.9114623	9565.961	13656.60	13113.075	20355.86
Boost2	0.9986732	0.9025047	1975.791	14821.07	2589.506	21360.79
Boost3	0.9984437	0.9103996	2165.181	14263.16	2804.508	20477.66

8 rows

After assessing all models we make the following observations, focusing on the 3 models: TreeFin (cp=0.00033), RF (random forest cross-validated) and Boost1 (n.trees=2000, shrinkage=0.01, interaction.depth=5). * CART has the worse performance (the simplest model with only 2 splits only explains 53% of the out-of-sample variation), but more splits give a good performance (76% out-of sample for TreeFin) * CART is very interpretable, especially with fewer trees * Boosted trees perform the best, across all 3 metrics (R2, MAE, RMSE) * The random forest gives very high in-sample performance 98%, but lower out-of- sample (89%) * We see a significant increase in R2, and a significant decrease in error (both MAE and RMSE) from

CART to ensemble methods. * From within the boosted trees, fewer trees performed better (2,000 and 3,000 versus 5,000), but these are still many trees (by comparison the RF only had 80 trees) * Boosted trees have very good in-sample performance, close to 100%, but lower out- of-sample (90%) which may suggest slight overfitting. * Random Forests and Boosted trees are very computationally intensive, with boosted trees taking a very long time in particular. CART models are very fast to fit since we only fit one tree