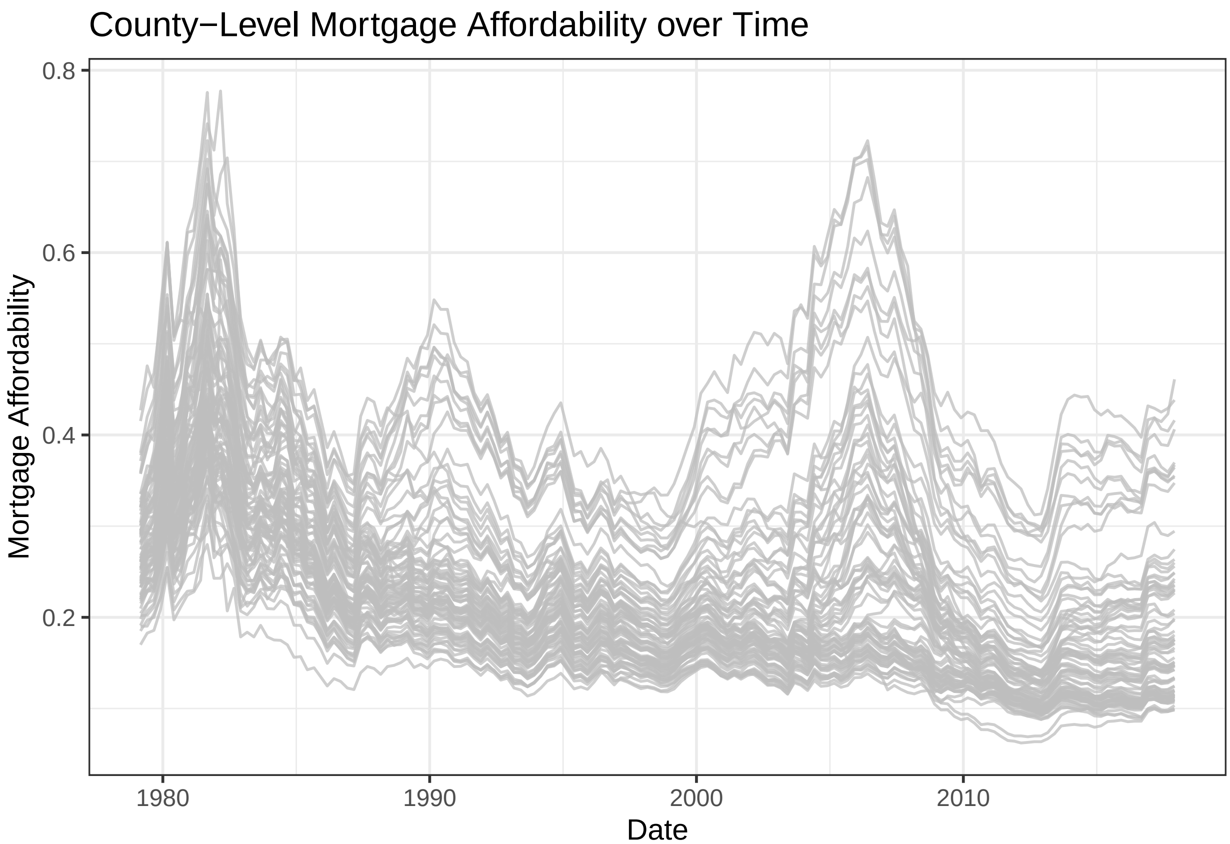
Classification

Zackary Frazier

```r
library(tidyverse)
library(lubridate)
library(caret)
library(rpart)
library(ROCR)
library(broom)
theme_set(theme_bw())

csv_file <- "Affordability_Wide_2017Q4_Public.csv"
tidy_afford <- read_csv(csv_file) %>%
  filter(Index == "Mortgage Affordability") %>%
  drop_na() %>%
  filter(RegionID != 0, RegionName != "United States") %>%
  dplyr::select(RegionID, RegionName, matches("^[1|2]")) %>%
  gather(time, affordability, matches("^[1|2]")) %>%
  type_convert(col_types=cols(time=col_date(format="%Y-%m")))
tidy_afford
```

```
## # A tibble: 12,480 x 4
##    RegionID RegionName                         time       affordability
##       <dbl> <chr>                              <date>             <dbl>
##  1   394913 New York, NY                       1979-03-01         0.262
##  2   753899 Los Angeles-Long Beach-Anaheim, CA 1979-03-01         0.358
##  3   394463 Chicago, IL                        1979-03-01         0.262
##  4   394514 Dallas-Fort Worth, TX              1979-03-01         0.301
##  5   394974 Philadelphia, PA                   1979-03-01         0.204
##  6   394692 Houston, TX                        1979-03-01         0.243
##  7   395209 Washington, DC                     1979-03-01         0.254
##  8   394856 Miami-Fort Lauderdale, FL          1979-03-01         0.268
##  9   394347 Atlanta, GA                        1979-03-01         0.248
## 10   394404 Boston, MA                         1979-03-01         0.222
## # ... with 12,470 more rows
```

```r
tidy_afford %>%
  ggplot(aes(x=time,y=affordability,group=factor(RegionID))) +
  geom_line(color="GRAY", alpha=3/4, size=1/2) +
  labs(title="County-Level Mortgage Affordability over Time",
         x="Date", y="Mortgage Affordability")
```



Can we predict if mortgage affordability will increase or decrease a year from now?

```
outcome_df <- tidy_afford %>%
  mutate(yq = quarter(time, with_year=TRUE)) %>%
  filter(yq %in% c("2016.4", "2017.4")) %>%
  select(RegionID, RegionName, yq, affordability) %>%
  spread(yq, affordability) %>%
  mutate(diff = `2017.4` - `2016.4`) %>%
  mutate(Direction = ifelse(diff>0, "up", "down")) %>%
  select(RegionID, RegionName, Direction)
outcome_df
```

```
## # A tibble: 80 x 3
##    RegionID RegionName      Direction
##       <dbl> <chr>           <chr>
##  1   394304 Akron, OH       down
##  2   394312 Albuquerque, NM down
##  3   394318 Allentown, PA   down
##  4   394347 Atlanta, GA     up
##  5   394355 Austin, TX      up
##  6   394357 Bakersfield, CA down
##  7   394358 Baltimore, MD   down
##  8   394367 Baton Rouge, LA up
```

```
## 9   394378 Bellingham, WA  up
## 10  394388 Birmingham, AL  down
## # ... with 70 more rows
```

```r
predictor_df <- tidy_afford %>%
  filter(year(time) <= 2016)
```

Question: Is a decision tree model better than a random forest model for this data?

Date Preparation

Here we combine our predictor with our outcomes. To train our data we'll need our data to show how affordability changes over time for each region, so we'll spread the affordability data over the time periods.

```r
total_df <- predictor_df %>%
  inner_join(y=outcome_df) %>%
  spread(time, affordability) %>%
  select(-RegionName)
```

```
## Joining, by = c("RegionID", "RegionName")
```

```r
# standardize the data
for(i in 3:ncol(total_df)) {
  col_mean <- sapply(total_df[,i], mean)
  col_sd <- sapply(total_df[,i], sd)
  for(k in 1:nrow(total_df)) {
    total_df[k, i] <- (total_df[k, i] - col_mean) / col_sd
  }
}

head(total_df)
```

```
## # A tibble: 6 x 154
##   RegionID Direction `1979-03-01` `1979-06-01` `1979-09-01` `1979-12-01`
##      <dbl> <chr>            <dbl>        <dbl>        <dbl>        <dbl>
## 1   394304 down            -0.770     -0.753         -0.741       -0.836
## 2   394312 down             0.263      0.357          0.344        0.240
## 3   394318 down            -0.608     -0.826         -0.933       -0.762
## 4   394347 up              -0.475     -0.463         -0.449       -0.360
## 5   394355 up               0.150     -0.00858        0.110        0.247
## 6   394357 down             0.370      0.472          0.441        0.423
## # ... with 148 more variables: `1980-03-01` <dbl>, `1980-06-01` <dbl>,
## #   `1980-09-01` <dbl>, `1980-12-01` <dbl>, `1981-03-01` <dbl>,
## #   `1981-06-01` <dbl>, `1981-09-01` <dbl>, `1981-12-01` <dbl>,
## #   `1982-03-01` <dbl>, `1982-06-01` <dbl>, `1982-09-01` <dbl>,
## #   `1982-12-01` <dbl>, `1983-03-01` <dbl>, `1983-06-01` <dbl>,
## #   `1983-09-01` <dbl>, `1983-12-01` <dbl>, `1984-03-01` <dbl>,
## #   `1984-06-01` <dbl>, `1984-09-01` <dbl>, `1984-12-01` <dbl>,
## #   `1985-03-01` <dbl>, `1985-06-01` <dbl>, `1985-09-01` <dbl>,
## #   `1985-12-01` <dbl>, `1986-03-01` <dbl>, `1986-06-01` <dbl>,
## #   `1986-09-01` <dbl>, `1986-12-01` <dbl>, `1987-03-01` <dbl>,
## #   `1987-06-01` <dbl>, `1987-09-01` <dbl>, `1987-12-01` <dbl>,
## #   `1988-03-01` <dbl>, `1988-06-01` <dbl>, `1988-09-01` <dbl>,
## #   `1988-12-01` <dbl>, `1989-03-01` <dbl>, `1989-06-01` <dbl>,
## #   `1989-09-01` <dbl>, `1989-12-01` <dbl>, `1990-03-01` <dbl>,
```

```
## #   `1990-06-01` <dbl>, `1990-09-01` <dbl>, `1990-12-01` <dbl>,
## #   `1991-03-01` <dbl>, `1991-06-01` <dbl>, `1991-09-01` <dbl>,
## #   `1991-12-01` <dbl>, `1992-03-01` <dbl>, `1992-06-01` <dbl>,
## #   `1992-09-01` <dbl>, `1992-12-01` <dbl>, `1993-03-01` <dbl>,
## #   `1993-06-01` <dbl>, `1993-09-01` <dbl>, `1993-12-01` <dbl>,
## #   `1994-03-01` <dbl>, `1994-06-01` <dbl>, `1994-09-01` <dbl>,
## #   `1994-12-01` <dbl>, `1995-03-01` <dbl>, `1995-06-01` <dbl>,
## #   `1995-09-01` <dbl>, `1995-12-01` <dbl>, `1996-03-01` <dbl>,
## #   `1996-06-01` <dbl>, `1996-09-01` <dbl>, `1996-12-01` <dbl>,
## #   `1997-03-01` <dbl>, `1997-06-01` <dbl>, `1997-09-01` <dbl>,
## #   `1997-12-01` <dbl>, `1998-03-01` <dbl>, `1998-06-01` <dbl>,
## #   `1998-09-01` <dbl>, `1998-12-01` <dbl>, `1999-03-01` <dbl>,
## #   `1999-06-01` <dbl>, `1999-09-01` <dbl>, `1999-12-01` <dbl>,
## #   `2000-03-01` <dbl>, `2000-06-01` <dbl>, `2000-09-01` <dbl>,
## #   `2000-12-01` <dbl>, `2001-03-01` <dbl>, `2001-06-01` <dbl>,
## #   `2001-09-01` <dbl>, `2001-12-01` <dbl>, `2002-03-01` <dbl>,
## #   `2002-06-01` <dbl>, `2002-09-01` <dbl>, `2002-12-01` <dbl>,
## #   `2003-03-01` <dbl>, `2003-06-01` <dbl>, `2003-09-01` <dbl>,
## #   `2003-12-01` <dbl>, `2004-03-01` <dbl>, `2004-06-01` <dbl>,
## #   `2004-09-01` <dbl>, `2004-12-01` <dbl>, ...
```

Now we create the 10-folds and create the training set and the testing set.

```
set.seed(1234)

partitionRule <- createFolds(total_df$Direction, k=10, list=F)
trainingSet <- total_df[partitionRule,]
testingSet <- total_df[-partitionRule,]

names(trainingSet) <- make.names(colnames(trainingSet))
names(testingSet) <- make.names(colnames(testingSet))

splitRule <- trainControl(method='cv',
                          number=10,
                          classProbs=TRUE,
                          summaryFunction=twoClassSummary)
```

Here I test the accuracy of our models using predictions and express the results as a confusion matrix. I use the predict function instead of the train function because the train function keeps causing problems when I run the prediction function.

Decision Tree

```
tree <- rpart(Direction~., data=trainingSet)
treePred <- predict(tree, newdata=testingSet, type='vector')
tp <- prediction(treePred, testingSet$Direction)

treePred[treePred == 1] <- 'down'
treePred[treePred == 2] <- 'up'

confusionMatrix(factor(treePred), factor(testingSet$Direction))
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction down up
##       down   12 14
##       up     11 33
##
##                Accuracy : 0.6429
##                  95% CI : (0.5193, 0.7539)
##     No Information Rate : 0.6714
##     P-Value [Acc > NIR] : 0.7404
##
##                   Kappa : 0.2167
##
##  Mcnemar's Test P-Value : 0.6892
##
##             Sensitivity : 0.5217
##             Specificity : 0.7021
##          Pos Pred Value : 0.4615
##          Neg Pred Value : 0.7500
##              Prevalence : 0.3286
##          Detection Rate : 0.1714
##    Detection Prevalence : 0.3714
##       Balanced Accuracy : 0.6119
##
##        'Positive' Class : down
##
```

Here I rest the predictions made by a random forest. Interestingly, it's predictions are less accurate than the decision tree. This may imply that the vastness of the amounts of data are skewing the overall effeciveness.

Random Forests

```r
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

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

```r
forest <- randomForest(ifelse(Direction == 'up', 1, 0)~.,data=trainingSet, type='raw')
forestPred <- predict(forest, newdata=testingSet, type='response')
fp <- prediction(forestPred, testingSet$Direction)

forestPred[forestPred >= 0.5] <- 'up'
forestPred[forestPred < 0.5] <- 'down'
```

```r
confusionMatrix(factor(forestPred), factor(testingSet$Direction))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction down up
##       down   19 34
##       up      4 13
##
##                Accuracy : 0.4571
##                  95% CI : (0.3374, 0.5806)
##     No Information Rate : 0.6714
##     P-Value [Acc > NIR] : 0.9999
##
##                   Kappa : 0.077
##
##  Mcnemar's Test P-Value : 2.546e-06
##
##             Sensitivity : 0.8261
##             Specificity : 0.2766
##          Pos Pred Value : 0.3585
##          Neg Pred Value : 0.7647
##              Prevalence : 0.3286
##          Detection Rate : 0.2714
##    Detection Prevalence : 0.7571
##       Balanced Accuracy : 0.5513
##
##        'Positive' Class : down
##
```

Here I generate three functions for getting ROC data each taylored to the different models.

```r
# a function to obtain performance data
# (tpr and fpr) over the given cross validation
get_roc_data_tree <- function(df, ntree, cv_partition, type, fit_control) {
  mean_fpr <- seq(0, 1, len=100)
  aucs <- numeric(length(cv_partition))

  res <- lapply(seq_along(cv_partition),  function(i) {
    fit <- rpart(Direction~., data=trainingSet)

    preds <- predict(fit, newdata=testingSet,type="vector")

    perf <- ROCR::prediction(preds, testingSet$Direction) %>%
      ROCR::performance(measure="tpr", x.measure="fpr")

    fpr <- unlist(perf@x.values)
    tpr <- unlist(perf@y.values)


    interp_tpr <- approxfun(fpr, tpr)(mean_fpr)
    interp_tpr[1] <- 0.0
```

```
    data_frame(fold=rep(i, length(mean_fpr)), fpr=mean_fpr, tpr=interp_tpr)
  })


  do.call(rbind, res)
}

get_roc_data_forest <- function(df, ntree, cv_partition, type, fit_control) {
  mean_fpr <- seq(0, 1, len=100)
  aucs <- numeric(length(cv_partition))
  test <- testingSet
  test$Direction[test$Direction == 'up'] <- 1
  test$Direction[test$Direction == 'down'] <- 0

  res <- lapply(seq_along(cv_partition),  function(i) {
    fit <- randomForest(ifelse(Direction == 'up', 1, 0)~.,
                        data=trainingSet, type='raw')

    preds <- predict(fit, newdata=testingSet,type="response")
    preds[preds >= 0.5] <- 1
    preds[forestPred < 0.5] <- 0

    perf <- ROCR::prediction(preds, test$Direction) %>%
      ROCR::performance(measure="tpr", x.measure="fpr")

    fpr <- unlist(perf@x.values)
    tpr <- unlist(perf@y.values)


    interp_tpr <- approxfun(fpr, tpr)(mean_fpr)
    interp_tpr[1] <- 0.0


    data_frame(fold=rep(i, length(mean_fpr)), fpr=mean_fpr, tpr=interp_tpr)
  })


  do.call(rbind, res)
}


compute_auc <- function(curve_df) {
  curve_df %>%
    group_by(fold) %>%
    summarize(auc=pracma::trapz(fpr, tpr))
}
```

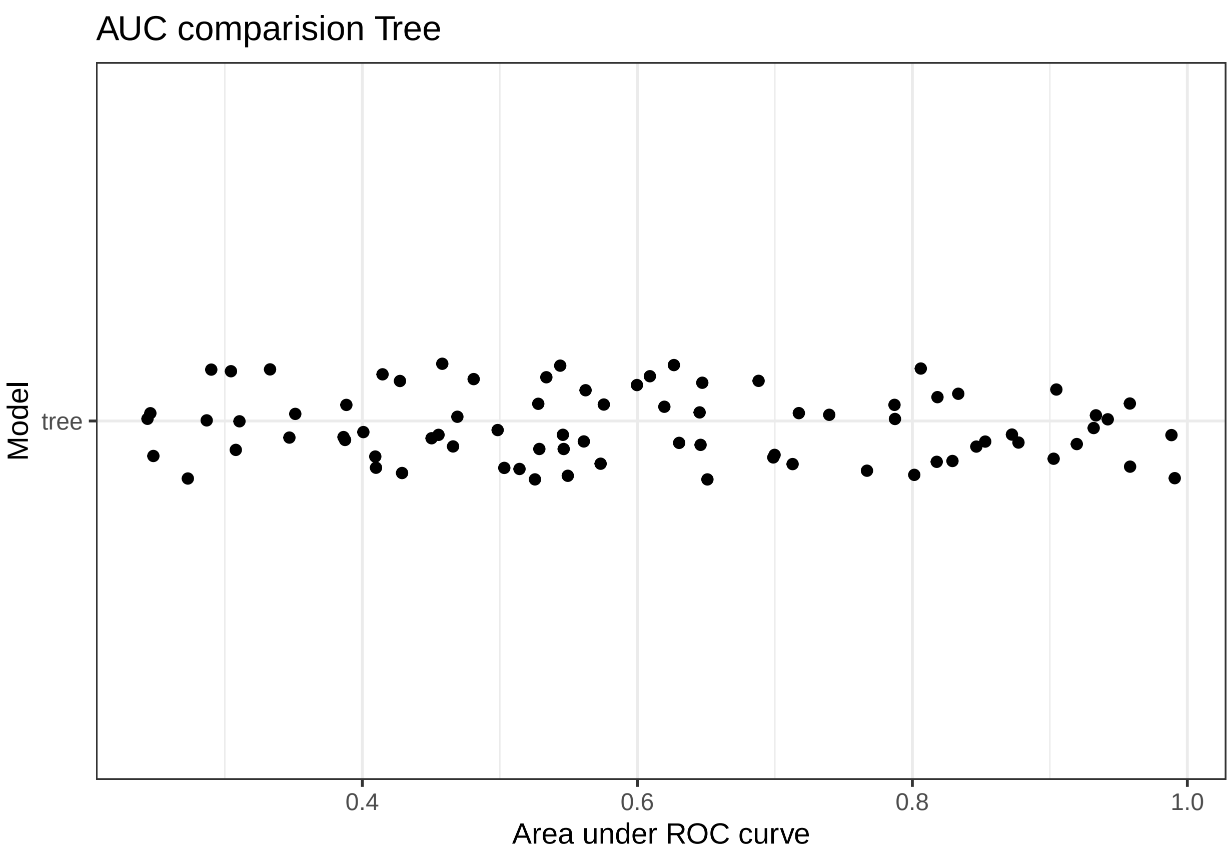
Here I get the performance data for 500 trees and 500 random forests. This allows me to retrieve a sufficiently large enough amount of data for ROC and AUROC analysis.

```
curve_tree <- get_roc_data_tree(df=total_df, ntree=500, cv_partition=partitionRule,
                                    type='rpart', fit_control=splitRule) %>%
  mutate(model="tree")
auc_tree <- compute_auc(curve_tree) %>%
```

```
  mutate(model="tree")


curve_forest <- get_roc_data_forest(df=total_df, ntree=500,
                        cv_partition=partitionRule,
                        type='forest', fit_control=splitRule) %>%
  mutate(model="forest")
auc_forest <- compute_auc(curve_forest) %>%
  mutate(model="forest")
```

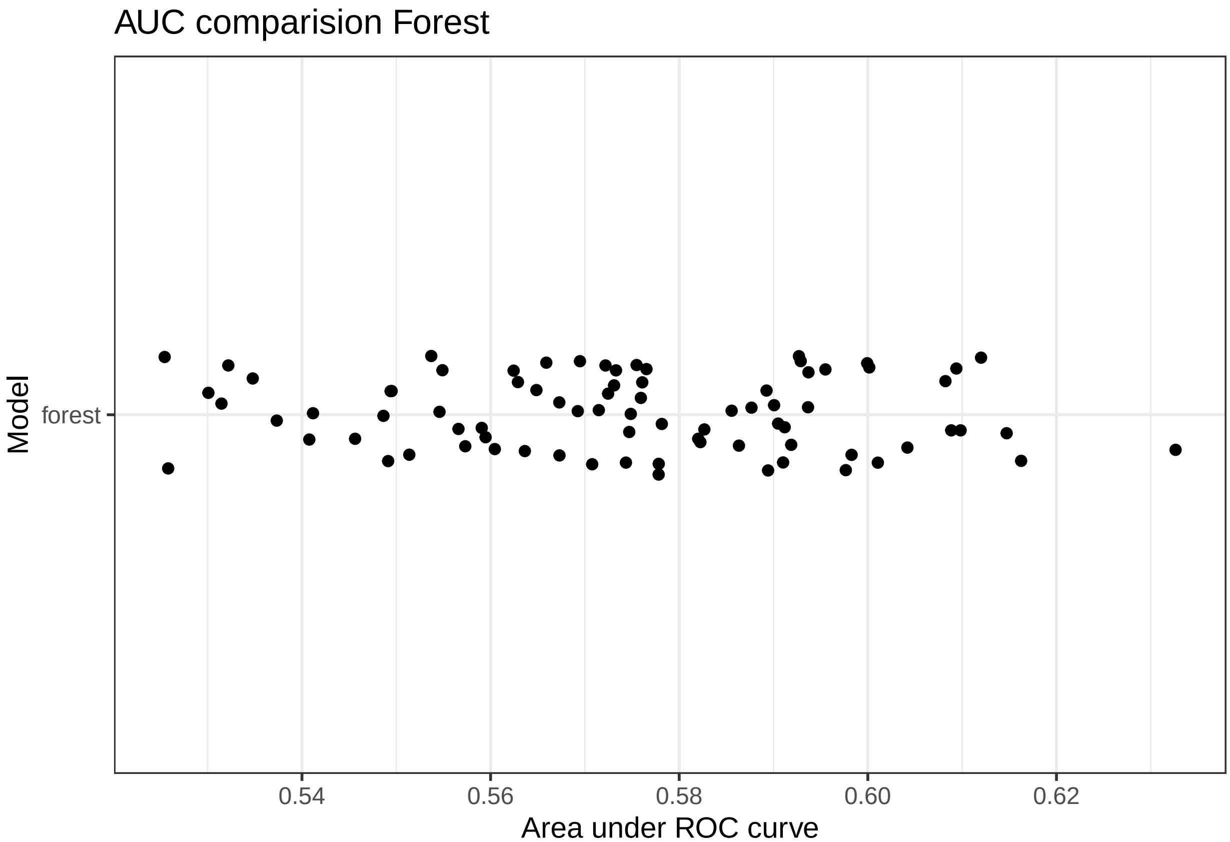
Here I compare the AUC of each the tree model and the forest model. These analyses are expressed as two different graphs because the combination of the data into one graph causes the forest data to appear as a small dot, and that's not very helpful for analysis. It appears the tree model has significantly more variance as the threshold changes while the forest model's AUC consistently has a value of 0.6.

```
ggplot(auc_tree, aes(x=model, y=auc)) +
  geom_jitter(position=position_jitter(0.1)) +
  coord_flip() +
  labs(title="AUC comparision Tree",
       x="Model",
       y="Area under ROC curve")
```




```
ggplot(auc_forest, aes(x=model, y=auc)) +
  geom_jitter(position=position_jitter(0.1)) +
  coord_flip() +
  labs(title="AUC comparision Forest",
       x="Model",
```

```
      y="Area under ROC curve")
```



Now we use linear regression to analyze the differences between the models. The estimate being positive shows that the tree model is slightly better at predicting the the results. The small p.value indicates that we can ignore the null hypothesis that both models are equivalent in terms of measuring the data.

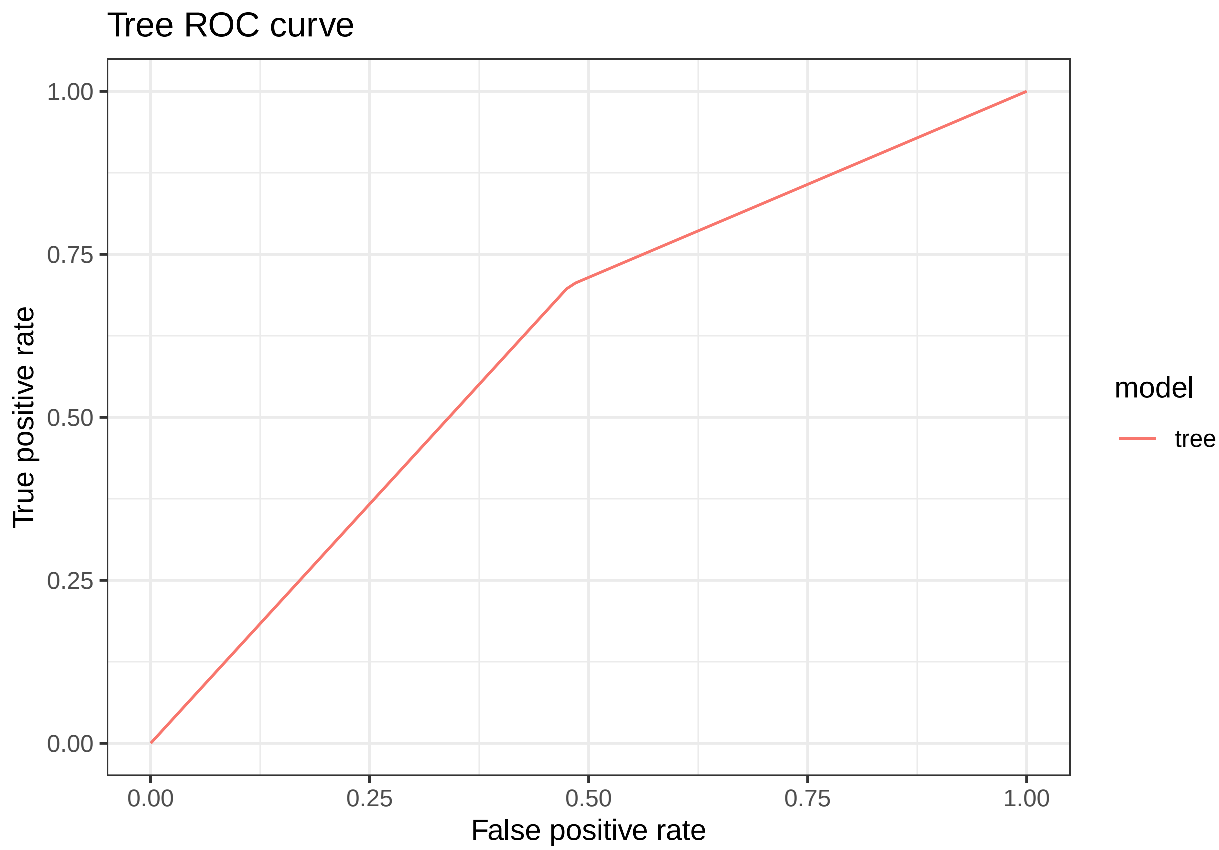
```
library(broom)

lm(auc~model, data=rbind(auc_forest, auc_tree)) %>%
  tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   0.574    0.00187     308.  1.88e-221
## 2 modeltree     0.0374   0.00264      14.2 5.44e- 30
```

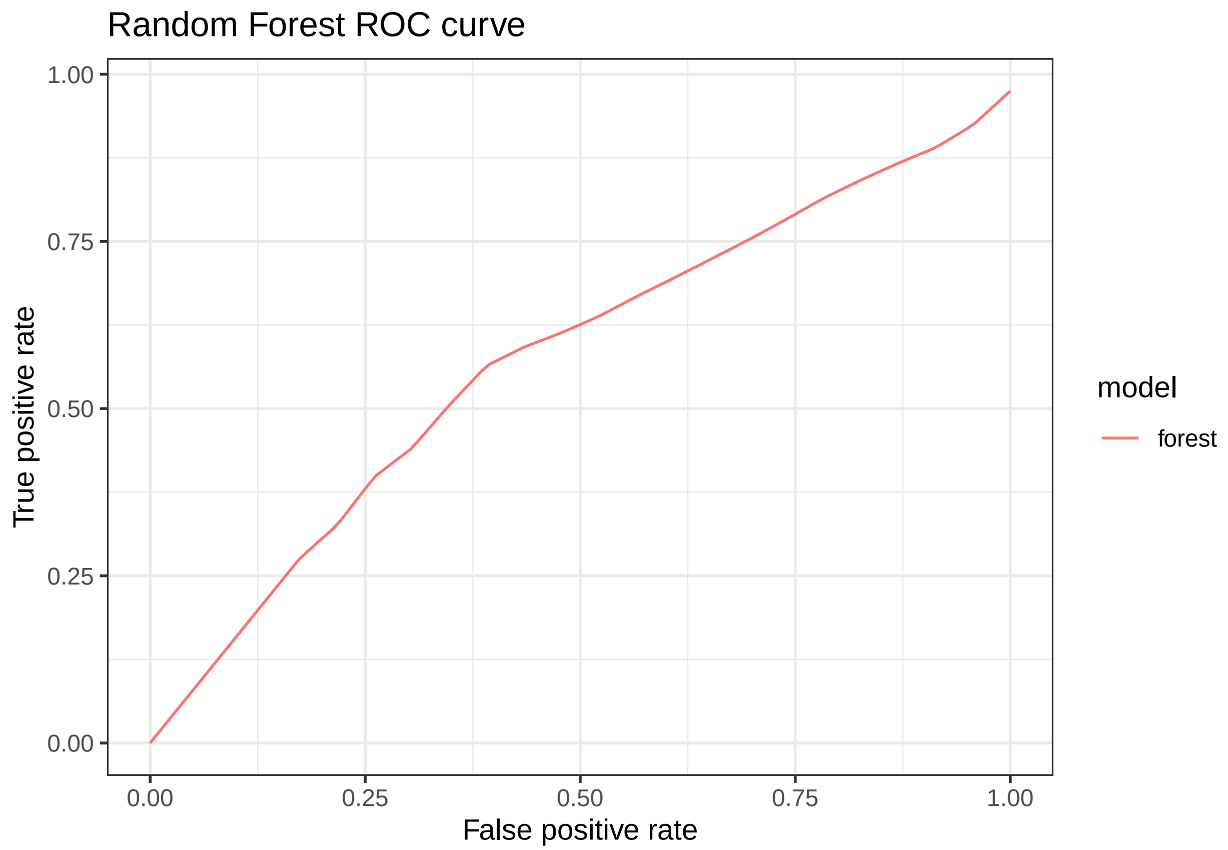
Here we can see a side-by-side comparison of the ROC curves of the tree model and the random forest model. Clearly we can see from a visual analysis that the area under the tree's curve is greater than the area under the random forest's curve.

```
curve_tree  %>%
  group_by(model, fpr) %>%
  summarize(tpr = mean(tpr)) %>%
  ggplot(aes(x=fpr, y=tpr, color=model)) +
    geom_line() +
    labs(title = "Tree ROC curve",
         x = "False positive rate",
```

```
         y = "True positive rate")
```




```
curve_forest%>%
  group_by(model, fpr) %>%
  summarize(tpr = mean(tpr)) %>%
  ggplot(aes(x=fpr, y=tpr, color=model)) +
    geom_line() +
    labs(title = "Random Forest ROC curve",
         x = "False positive rate",
         y = "True positive rate")
```



In conclusion, given this experiment, the tree method is more accurate than the random forest method for predicting affordability trends.