Australian Rain Predictor

Andrew White

1/10/2020

Introduction

This project looks at a dataset pulled from Kaggle that contains the weather data from 2007-2017 for Australia from 44 different cities. The goal of the project is to predict if it will rain tomorrow using the other weather variables. There is a lot of practical use for weather modeling, especially in Australia as they battle massive and devastating wildfires, since predicting the weather allows the firefighters to strategize where to defend next. Using data visualization, I will determine the best predictors then use them to build the model. I will use a decision tree model to make a prediction then a random forest in an attempt to improve the prediction. I will use Accuracy, Sensitivity, Specificity, and Balanced Accuracy to measure the success of each model. I am using the packages- tidyverse, caret, lubridate, rpart, and randomForest.

Data loading and cleaning

The first step is to load all the required packages

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate",repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
```

Next load the dataset

```
url<-"https://github.com/adwhite3/Edx-Final/blob/master/weatherAUS.csv?raw=TRUE"
data<-read_csv(url)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Date = col_date(format = ""),
##   Location = col_character(),
##   Evaporation = col_logical(),
##   Sunshine = col_logical(),
##   WindGustDir = col_character(),
##   WindDir9am = col_character(),
##   WindDir3pm = col_character(),
##   RainToday = col_character(),
##   RainTomorrow = col_character()
## )

## See spec(...) for full column specifications.
```

I will be making a classification tree to predict if there will or will not be precipitation tomorrow in Austrailia Data ranges from 2007-2017 First I remove RISK_MM column because it is future information that will interfere with the prediction.

```
data$RISK_MM<-NULL
```

Now I can begin looking at data

```
summary(data)
```

```
##       Date              Location            MinTemp         MaxTemp
##  Min.   :2007-11-01   Length:142193      Min.   :-8.50   Min.   :-4.80
##  1st Qu.:2011-01-06   Class :character   1st Qu.: 7.60   1st Qu.:17.90
##  Median :2013-05-27   Mode  :character   Median :12.00   Median :22.60
##  Mean   :2013-04-01                      Mean   :12.19   Mean   :23.23
##  3rd Qu.:2015-06-12                      3rd Qu.:16.80   3rd Qu.:28.20
##  Max.   :2017-06-25                      Max.   :33.90   Max.   :48.10
##                                          NA's   :637     NA's   :322
##     Rainfall      Evaporation      Sunshine       WindGustDir
##  Min.   :  0.00   Mode :logical   Mode :logical   Length:142193
##  1st Qu.:  0.00   FALSE:240       FALSE:2308      Class :character
##  Median :  0.00   TRUE :1563      TRUE :326       Mode  :character
##  Mean   :  2.35   NA's :140390    NA's :139559
##  3rd Qu.:  0.80
##  Max.   :371.00
##  NA's   :1406
##  WindGustSpeed     WindDir9am         WindDir3pm         WindSpeed9am
##  Min.   :  6.00   Length:142193      Length:142193      Min.   :  0
##  1st Qu.: 31.00   Class :character   Class :character   1st Qu.:  7
##  Median : 39.00   Mode  :character   Mode  :character   Median : 13
##  Mean   : 39.98                                         Mean   : 14
##  3rd Qu.: 48.00                                         3rd Qu.: 19
##  Max.   :135.00                                         Max.   :130
##  NA's   :9270                                           NA's   :1348
##   WindSpeed3pm    Humidity9am      Humidity3pm      Pressure9am
##  Min.   : 0.00   Min.   :  0.00   Min.   :  0.00   Min.   : 980.5
##  1st Qu.:13.00   1st Qu.: 57.00   1st Qu.: 37.00   1st Qu.:1012.9
##  Median :19.00   Median : 70.00   Median : 52.00   Median :1017.6
##  Mean   :18.64   Mean   : 68.84   Mean   : 51.48   Mean   :1017.7
##  3rd Qu.:24.00   3rd Qu.: 83.00   3rd Qu.: 66.00   3rd Qu.:1022.4
##  Max.   :87.00   Max.   :100.00   Max.   :100.00   Max.   :1041.0
##  NA's   :2630    NA's   :1774     NA's   :3610     NA's   :14014
##   Pressure3pm       Cloud9am        Cloud3pm         Temp9am
##  Min.   : 977.1   Min.   :0.00    Min.   :0.0     Min.   :-7.20
##  1st Qu.:1010.4   1st Qu.:1.00    1st Qu.:2.0     1st Qu.:12.30
##  Median :1015.2   Median :5.00    Median :5.0     Median :16.70
##  Mean   :1015.3   Mean   :4.44    Mean   :4.5     Mean   :16.99
##  3rd Qu.:1020.0   3rd Qu.:7.00    3rd Qu.:7.0     3rd Qu.:21.60
##  Max.   :1039.6   Max.   :9.00    Max.   :9.0     Max.   :40.20
##  NA's   :13981    NA's   :53657   NA's   :57094   NA's   :904
##     Temp3pm       RainToday         RainTomorrow
##  Min.   :-5.40   Length:142193      Length:142193
##  1st Qu.:16.60   Class :character   Class :character
```

```
##  Median :21.10   Mode  :character   Mode  :character
##  Mean   :21.69
##  3rd Qu.:26.40
##  Max.   :46.70
##  NA's   :2726
```

```
head(data)
```

```
## # A tibble: 6 x 23
##   Date       Location MinTemp MaxTemp Rainfall Evaporation Sunshine
##   <date>     <chr>      <dbl>   <dbl>    <dbl> <lgl>       <lgl>
## 1 2008-12-01 Albury      13.4    22.9      0.6 NA          NA
## 2 2008-12-02 Albury       7.4    25.1      0   NA          NA
## 3 2008-12-03 Albury      12.9    25.7      0   NA          NA
## 4 2008-12-04 Albury       9.2    28        0   NA          NA
## 5 2008-12-05 Albury      17.5    32.3      1   NA          NA
## 6 2008-12-06 Albury      14.6    29.7      0.2 NA          NA
## # ... with 16 more variables: WindGustDir <chr>, WindGustSpeed <dbl>,
## #   WindDir9am <chr>, WindDir3pm <chr>, WindSpeed9am <dbl>,
## #   WindSpeed3pm <dbl>, Humidity9am <dbl>, Humidity3pm <dbl>,
## #   Pressure9am <dbl>, Pressure3pm <dbl>, Cloud9am <dbl>, Cloud3pm <dbl>,
## #   Temp9am <dbl>, Temp3pm <dbl>, RainToday <chr>, RainTomorrow <chr>
```

```
dim(data)
```

```
## [1] 142193     23
```

I will check what percent of each column is NA

```
colSums(is.na(data))/nrow(data)*100
```

```
##          Date      Location       MinTemp       MaxTemp      Rainfall
##     0.0000000     0.0000000     0.4479827     0.2264528     0.9887969
##   Evaporation      Sunshine   WindGustDir WindGustSpeed    WindDir9am
##    98.7320051    98.1475881     6.5615044     6.5193083     7.0418375
##    WindDir3pm  WindSpeed9am  WindSpeed3pm   Humidity9am   Humidity3pm
##     2.6569522     0.9480073     1.8495988     1.2476001     2.5388029
##   Pressure9am   Pressure3pm      Cloud9am      Cloud3pm       Temp9am
##     9.8556188     9.8324109    37.7353316    40.1524688     0.6357556
##       Temp3pm     RainToday  RainTomorrow
##     1.9171127     0.9887969     0.0000000
```

Evaporation, Sunshine, and both Cloud columns have too many NA's to be useful predictors so I can remove them

```
drops<-c("Evaporation","Sunshine","Cloud9am","Cloud3pm")
data<-data[,!(names(data)%in%drops)]
rm(drops)
na.exclude(data)%>%dim()
```

```
## [1] 112925     19
```

```
data<-na.exclude(data)
```

I will change RainTomorrow and RainToday from character class to factor so it is easier to sort them and use operators with them

```
data$RainTomorrow<-as.factor(data$RainTomorrow)
data$RainToday<-as.factor(data$RainToday)
nrow(data)
```

```
## [1] 112925
```

Over 100.000 observations is still plenty of data for predicting so I can remove all other NA's without impacting the results significantly Now I can split the data in test and training sets

```
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(y = data$RainTomorrow, times = 1, p = 0.1, list = FALSE)
train<- data[-test_index,]
temp <- data[test_index,]

validation <- temp %>%
  semi_join(train, by = "RainTomorrow") %>%
  semi_join(train, by = "RainToday") %>%
  semi_join(train, by = "Humidity3pm")%>%
  semi_join(train, by = "Pressure3pm")%>%
  semi_join(train, by = "Date")
```

Add rows removed from validation set back into data set

```
removed <- anti_join(temp, validation)
```

```
## Joining, by = c("Date", "Location", "MinTemp", "MaxTemp", "Rainfall", "WindGustDir", "WindGustSpeed"
```

```
data <- rbind(data, removed)
rm(url, removed, temp, test_index)
```

Data Exploration

Look at number of cities and date range of the data

```
range(train$Date)
```

```
## [1] "2007-11-01" "2017-06-25"
```

```
length(unique(train$Location))
```

```
## [1] 44
```

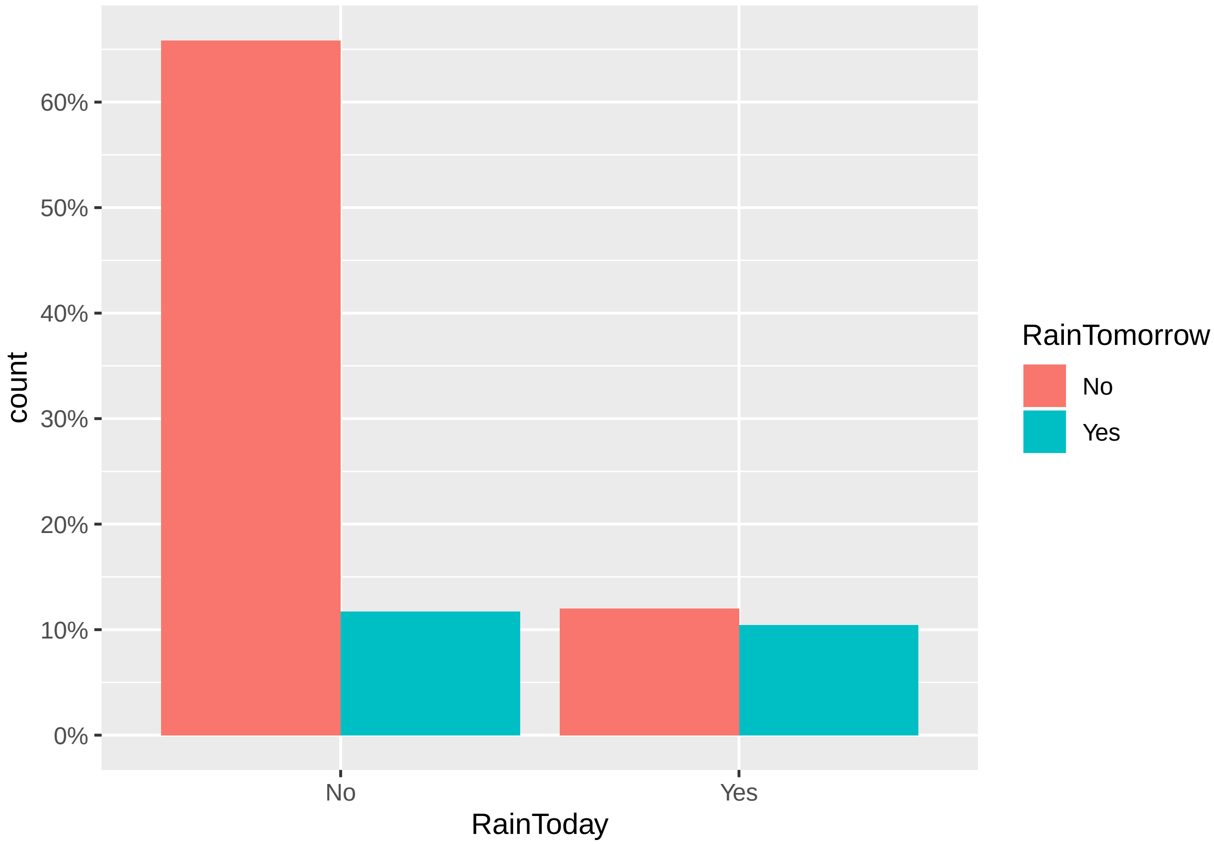
There is data from 44 Australian cities from 2007-2017

```r
sort(table(train$Location),decreasing =TRUE)
```

```
##
##           Darwin           Hobart         Brisbane            Perth
##             2810             2754             2745             2714
##    SydneyAirport MelbourneAirport     PerthAirport     MountGambier
##             2647             2639             2618             2615
##           Cairns          Mildura          Woomera       Townsville
##             2614             2600             2600             2596
##    NorfolkIsland         Ballarat        GoldCoast         Portland
##             2584             2568             2558             2542
##        Nuriootpa            Cobar       Wollongong        NorahHead
##             2517             2516             2512             2503
##       WaggaWagga         Adelaide         Canberra         Watsonia
##             2500             2479             2471             2462
##             Sale     AliceSprings          Bendigo            Moree
##             2456             2446             2428             2377
##     CoffsHarbour          Walpole       PearceRAAF           Albury
##             2273             2256             2222             2179
##    BadgerysCreek      Witchcliffe         Dartmoor           Sydney
##             2103             2078             2072             2060
##      Tuggeranong        Melbourne      Williamtown         Richmond
##             2046             2027             1960             1838
##       Launceston             Nhil            Uluru        Katherine
##             1394             1353             1289              611
```

Next I will figure out which data are the best predictors for Rain Tomorrow and remove the irrelevant columns

```r
train%>%ggplot(aes(RainToday, fill= RainTomorrow))+
  geom_bar(position = "dodge")+
  scale_y_continuous(labels = function(x) paste0(round(x/nrow(train)*100,1),"%"),breaks = seq(0,nrow(tra
```

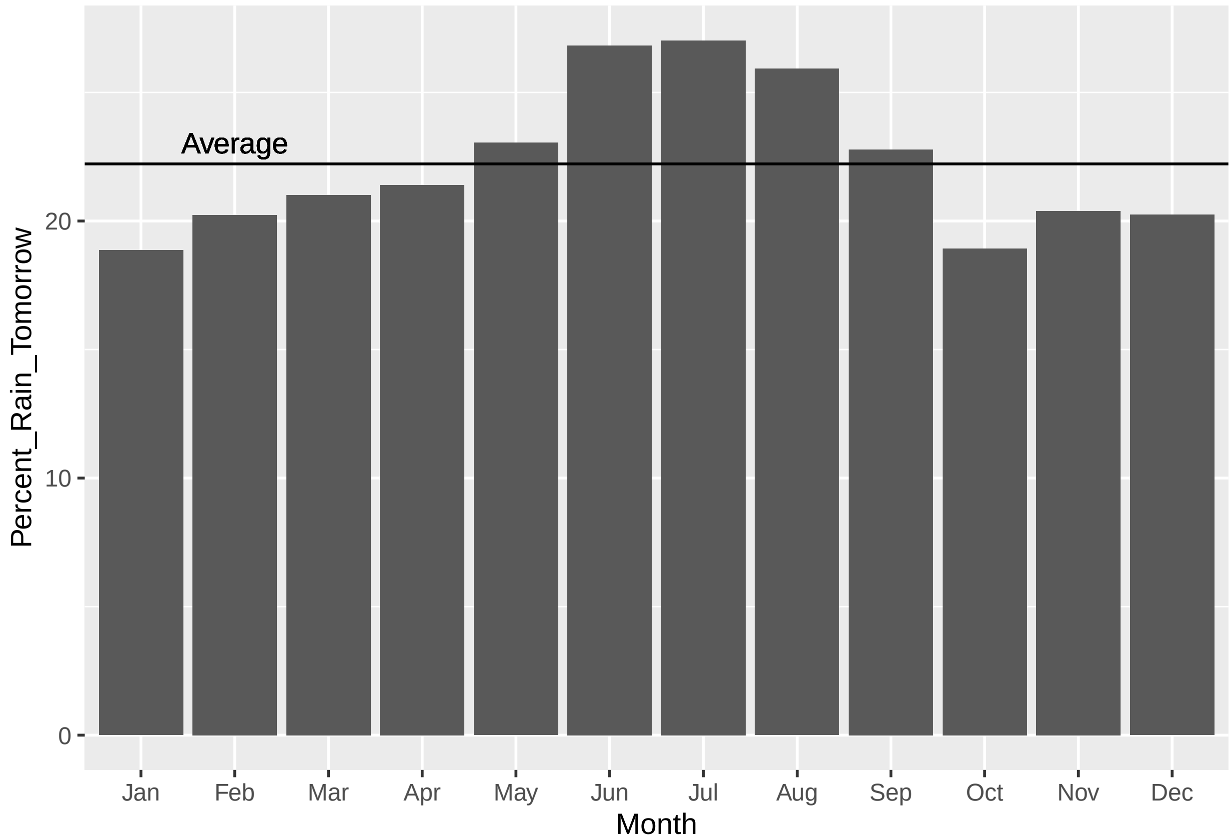


When it rains today, it is far more likely to rain tomorrow. In almost half of all occurences when it Rained Today, it Rained Tomorrow.

```
monthly_rain_tomorrow<-train%>%filter(month(Date),RainTomorrow=="Yes")%>%count(month(Date))
monthly_days<-count(train,month(train$Date))
pct_rain_tomorrow_monthly<-data_frame("Month"=factor(month.abb,levels = month.abb),"Percent_Rain_Tomorr
```

Look at a histogram of the monthly average rainfall relative to the average rainfall which is represented by the horizontal line

```
pct_rain_tomorrow_monthly%>%ggplot(aes(Month, Percent_Rain_Tomorrow))+
  geom_col()+
  geom_hline(yintercept = mean(pct_rain_tomorrow_monthly$Percent_Rain_Tomorrow))+
  geom_text(aes(2,mean(pct_rain_tomorrow_monthly$Percent_Rain_Tomorrow), label= "Average", vjust=-.5))
```



The Australian rainy season is from June to August

Does the amount of rain today predict the likelihood of rain tomorrow? A few extreme days with lots of rain will make it difficult to visualize a density plot so by checking the standard deviation of days with rain (y) I can tell how to limit the x-axis

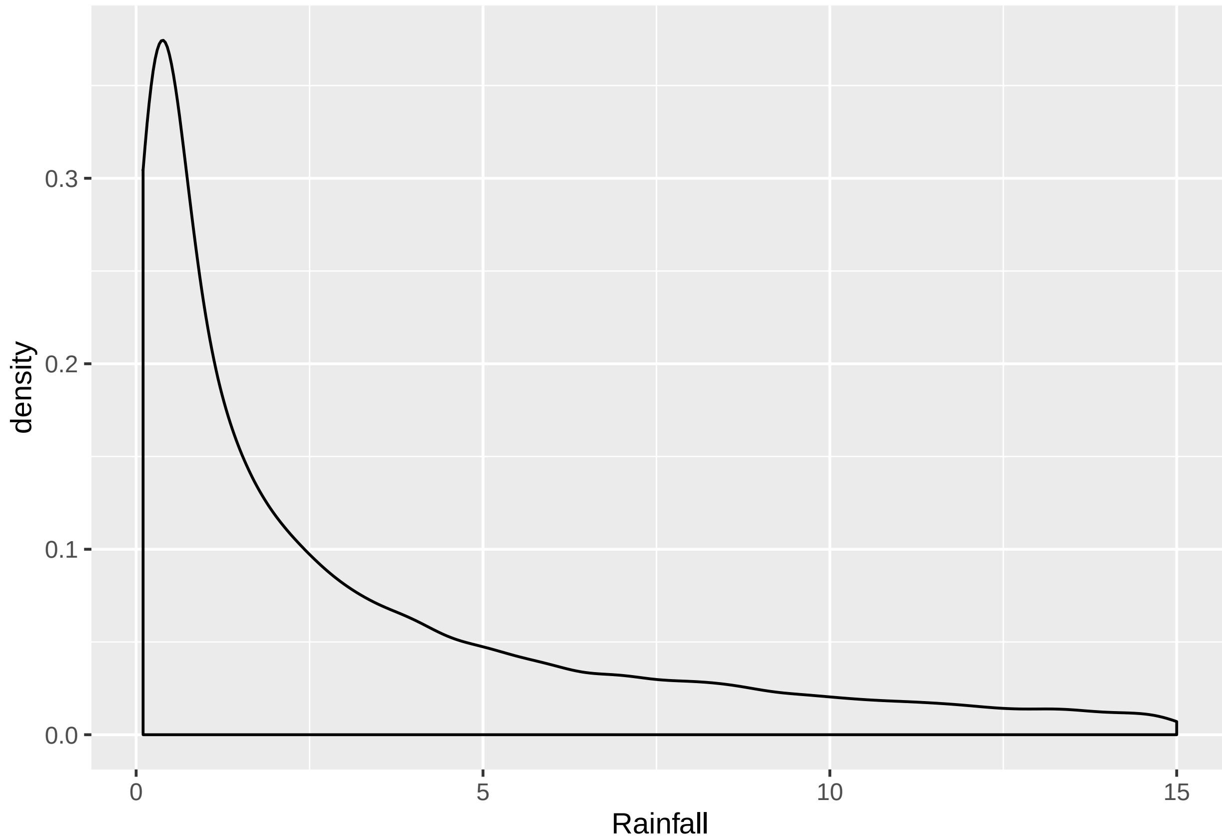
```
tail(sort(train$Rainfall),n=10)
```

```
##  [1] 206.8 208.5 210.6 219.6 225.0 236.8 247.2 268.6 278.4 367.6
```

```
y<-filter(train,train$Rainfall>=0.1)
n<-filter(train,train$Rainfall==0)
sd(y$Rainfall)
```

```
## [1] 13.38233
```

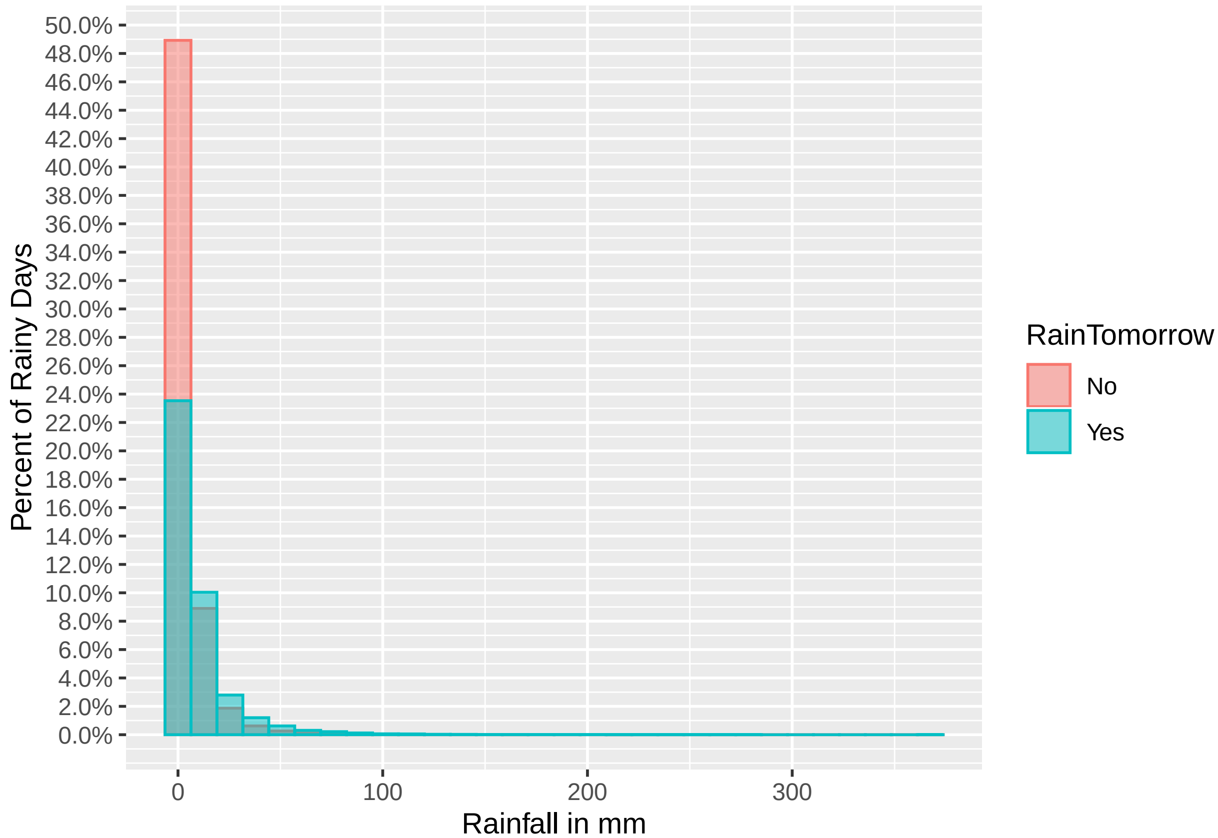
```
ggplot(train,aes(Rainfall))+
  geom_density()+
  scale_x_continuous(limits = c(0.1,15))
```



The density plot shows that the majority of days with rain get under 13mm of rain

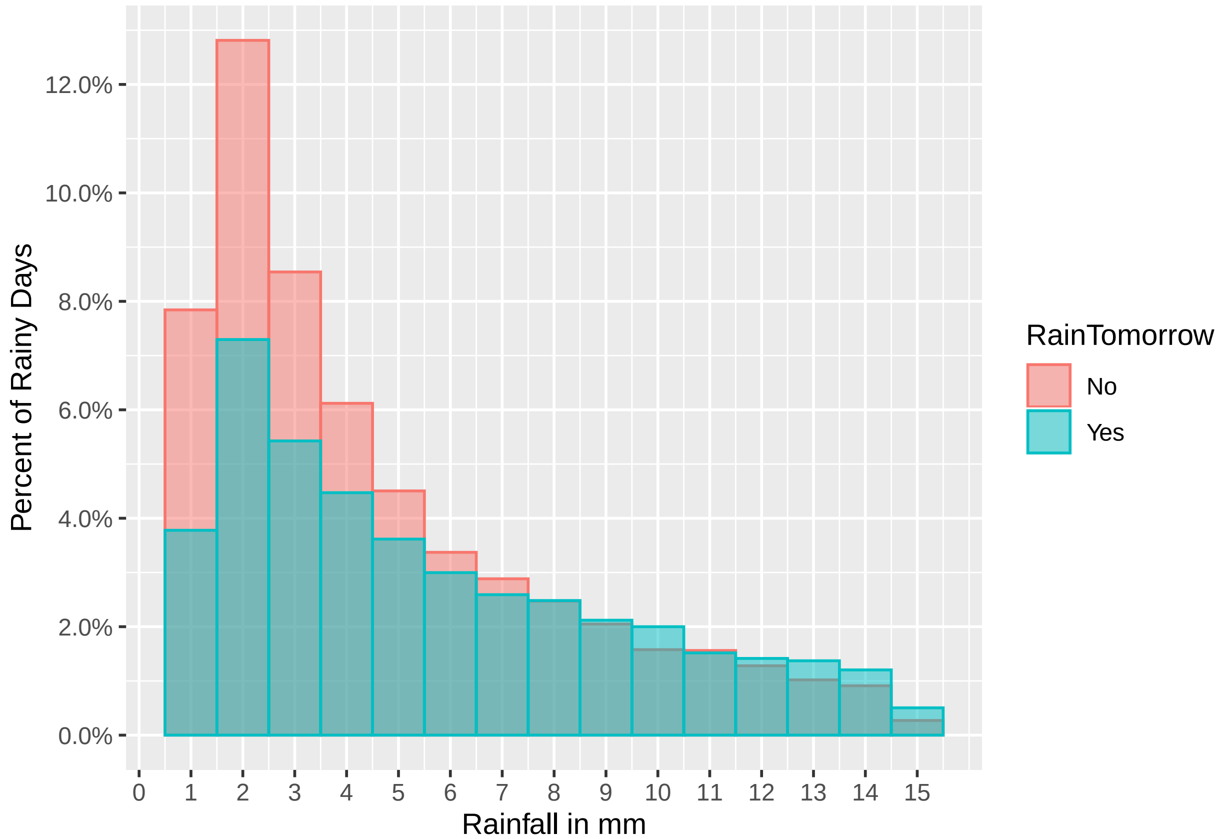
```
train %>%
  filter(Rainfall > 0.1) %>%
  ggplot(aes(x = Rainfall, fill = RainTomorrow, color = RainTomorrow)) +
  geom_histogram(aes(y = ((..count..) / sum(..count..))), position = "identity", alpha = 0.5) +
  scale_y_continuous(breaks = seq(0, 1, by = 0.02),
                     labels = scales::percent)+
  labs(x="Rainfall in mm", y="Percent of Rainy Days")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



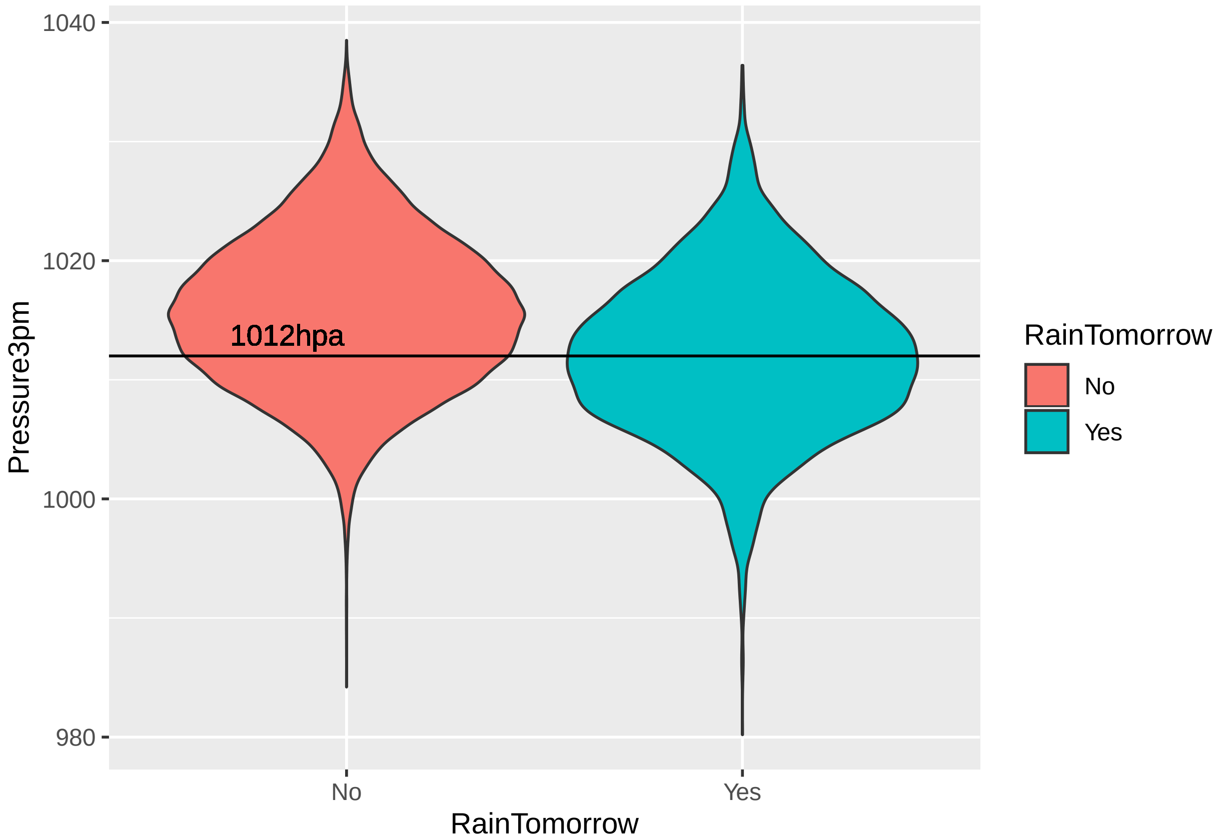
After a certain amount of rainfall, it is more likely to rain tomorrow, so I will zoom in to figure out the cutoff

```
train %>%
  filter(Rainfall > 1 & Rainfall < 15) %>%
  ggplot(aes(x = Rainfall, fill = RainTomorrow, color = RainTomorrow)) +
  geom_histogram(aes(y = ((..count..) / sum(..count..))), position = "identity", alpha = 0.5, binwidth =
  scale_x_continuous(breaks = seq(0, 15, by = 1)) +
  scale_y_continuous(breaks = seq(0, 1, by = 0.02),
                     labels = scales::percent)+
  labs(x="Rainfall in mm", y="Percent of Rainy Days")
```



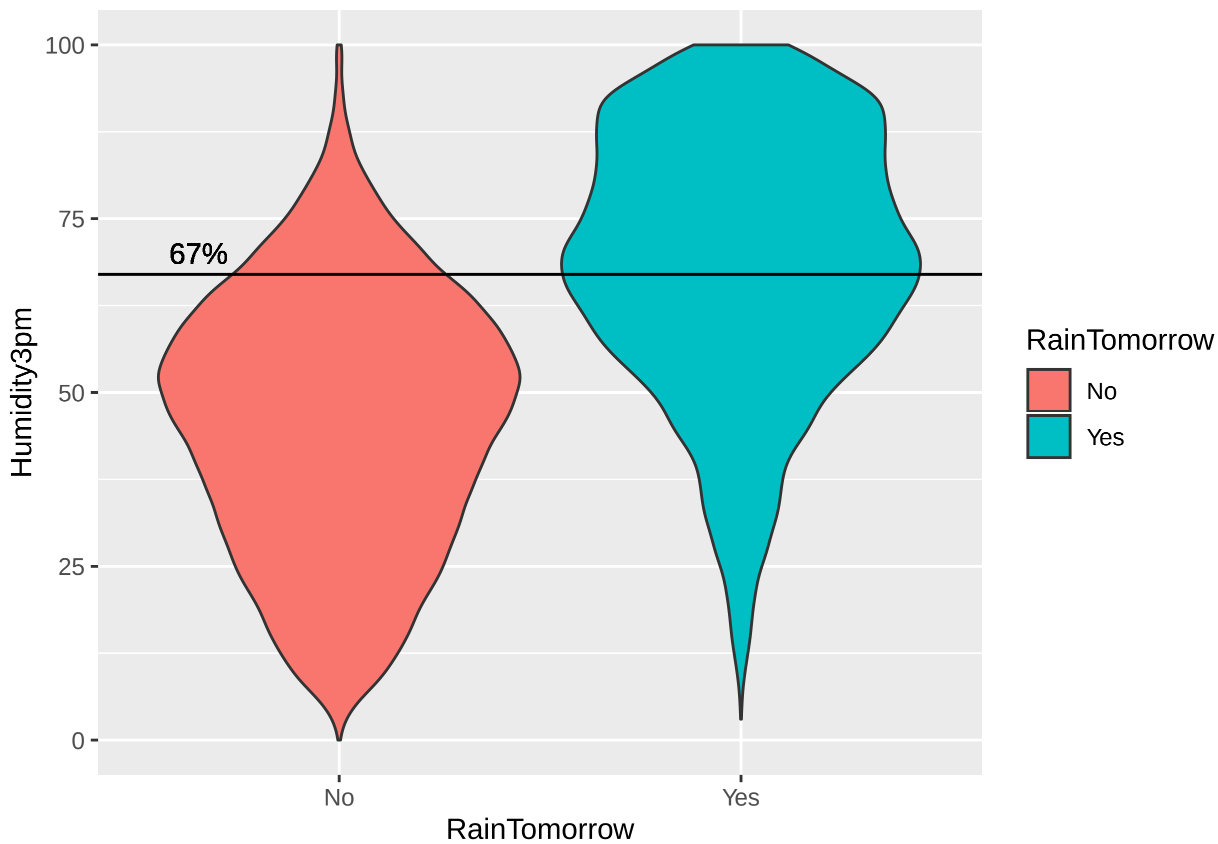
After 8mm of rain it is more likely to rain the next day than not rain Next I will check if Pressure at 3pm is a good predictor

```
n%>%ggplot(aes(x=RainTomorrow, y=Pressure3pm, fill= RainTomorrow)) +
  geom_violin()+
  geom_hline(yintercept = 1012)+
  geom_text(aes(.85,1012), label= "1012hpa", vjust=-.5)
```



As the violin chart shows, below 1012hpa, it is more likely to rain tomorrow Time to check if Humidity at 3pm is a good predictor of rainfall

```
train%>%ggplot(aes(x=RainTomorrow, y=Humidity3pm, fill= RainTomorrow)) +
  geom_violin()+
  geom_hline(yintercept = 67)+
  geom_text(aes(.65,67), label= "67%", vjust=-.5)
```



This shows us that it is more likely to rain tomorrow when humidity at 3pm is above 67%

Rain today, months June to August, more than 8mm of rainfall, pressure at 3pm below 1012hpa, and humidity at 3pm above 67% are the best predictors

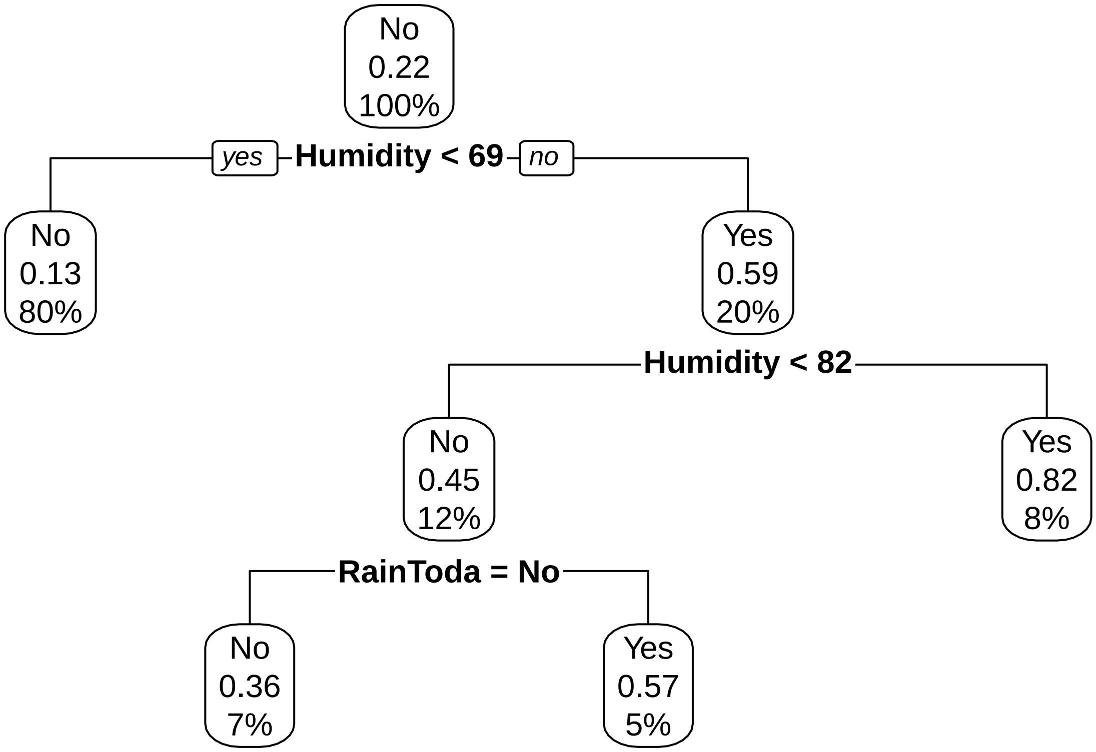
Methods/Analysis

Now to create the prediction model with these predictors

```
predictors<- c("Date","Rainfall", "Humidity3pm","Pressure3pm", "RainToday","RainTomorrow")
train<-train[,(names(train)%in%predictors)]
Decision_Tree<-rpart(RainTomorrow~ RainToday + Date + Humidity3pm + Pressure3pm, data = train, method =
```

I can visualize the tree to see where it makes the breaks

```
prp(Decision_Tree, type = 2, extra = "auto", branch = 1)
```



The rpart function automatically chose the predictors Humidity and Rain Today

```
prediction_class_1<-predict(Decision_Tree, validation, type = "class")
confusionMatrix(prediction_class_1,validation$RainTomorrow,positive = "Yes")
```

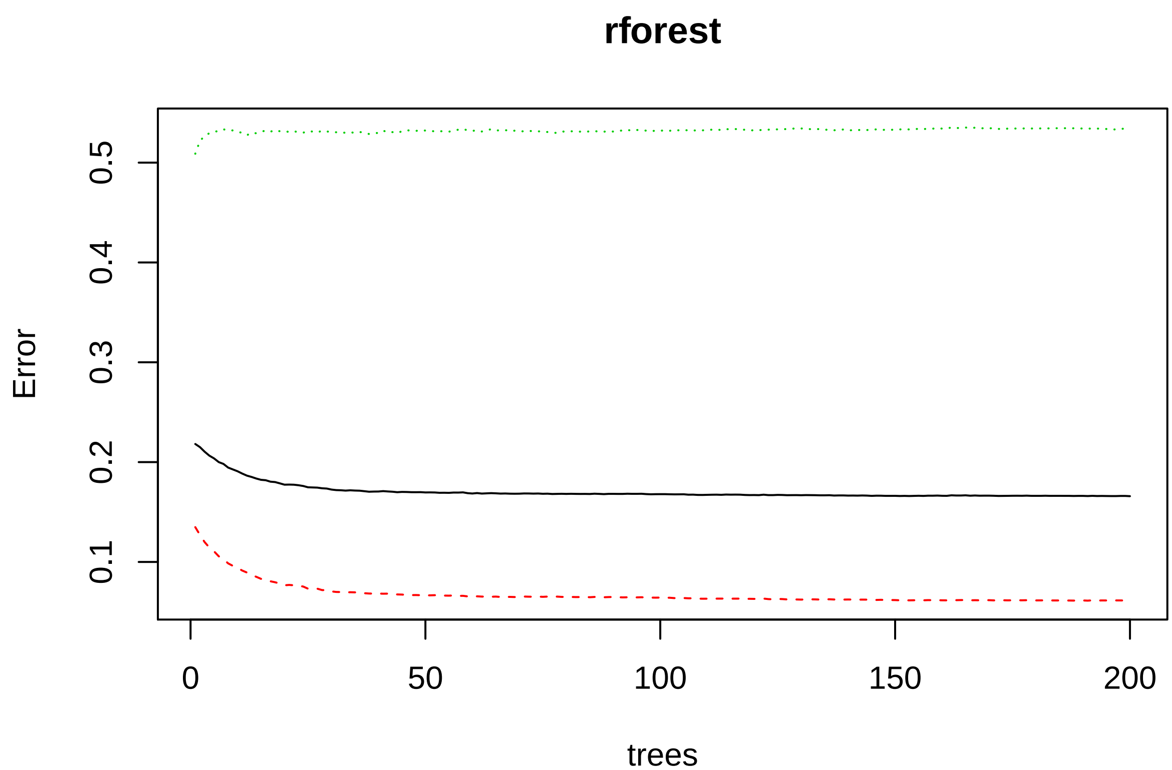
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  8369 1429
##        Yes  403 1065
##
##                Accuracy : 0.8374
##                  95% CI : (0.8304, 0.8442)
##     No Information Rate : 0.7786
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4469
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.42702
##             Specificity : 0.95406
##          Pos Pred Value : 0.72548
##          Neg Pred Value : 0.85415
##              Prevalence : 0.22137
```

```
##          Detection Rate : 0.09453
##    Detection Prevalence : 0.13030
##       Balanced Accuracy : 0.69054
##
##        'Positive' Class : Yes
##
```

Using a confusionMatrix, I can check the metrics for success. The decision tree model has an 83.83% accuracy which is pretty good. However, the Sensitivity is 42.7% which is a bit low and the Balanced Accuracy is only 69.05%. But by using Random Forrests I can improve the balanced accuracy and sensitivity

First I create a randomForest with the selected predictors and look at how many trees I need

```
rforest<-randomForest(RainTomorrow~ RainToday + Date + Humidity3pm + Pressure3pm, data = train, ntree=2
plot(rforest)
```



The error stops changing significantly around 50 trees so I can limit the forest to 50 trees to save computing power

```
rforest<-randomForest(RainTomorrow~ RainToday + Date + Humidity3pm + Pressure3pm, data = train, ntree=5
```

Now I can test to see if the randomForest improves the prediction of Rain Tomorrow

```
prediction_class_2<-predict(rforest,validation, type = "class")
confusionMatrix(prediction_class_2,validation$RainTomorrow, positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  8215 1308
##        Yes  557 1186
##
##                Accuracy : 0.8345
##                  95% CI : (0.8275, 0.8413)
##     No Information Rate : 0.7786
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4618
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4755
##             Specificity : 0.9365
##          Pos Pred Value : 0.6804
##          Neg Pred Value : 0.8626
##              Prevalence : 0.2214
##          Detection Rate : 0.1053
##    Detection Prevalence : 0.1547
##       Balanced Accuracy : 0.7060
##
##        'Positive' Class : Yes
##
```

With RandomForest, our new accuracy decreased slightly by 0.3% to 83.44%. However, the Sensitivity rose 4.7% to 47.47% and the Balanced Accuracy rose 1.52% to 70.57%

Conclusion

The Random Forest method slightly improved the balanced accuracy of the decision tree by raising the sensitivity. However, the Specificity dropped 1.74% which explains why the balanced accuracy did not increase further. With a specificity still above 90%, I think improving the low Sensitivity is worth the trade-off. One area of future research is altering the node size of the random forest to get a better accuracy, but my computer just kept freezing everytime I tried so with more computing power, perhaps a high accuracy could be achieved. Decision trees are good for making binary decisions, and with something as voluminous as weather data, there might be other machine learning models that could be better. One of the advantages of decision trees is it makes it easier to understand how the machine comes to a particular outcome since we can visualize each split of the branches.