Week 9: Decision Trees and Random Forests

DSC 365: Introduction to Data Science

2024-10-22

Supervised vs Unsupervised Learning

Learning techniques fall into two categories:

1. Supervised learning: Use input data (predictors) to predict the value of an output data (response variable). If the output data is continuous, we call it regression. If the output data is categorical, we call it classification.

- You're familiar with some (simple) supervised learning techniques already: like a linear model: y ~ x1 + x2 + x3

2. Unsupervised learning: There is no response variable. We try to learn the pattern of the input data, usually by clustering them into several groups.

Tree-Based Methods

- Can be used for both regression and classification
  - Regression models have a quantitative response variable (and can thus often be visualized as a geometric surface), classification models have a categorical response (and are often visualized as a discrete surface, i.e., a tree).
- These involve stratifying or segmenting the predictor space into a number of simple regions
  - Have a set of decision rules that can be summarized in a tree

Example: Marijuana legalization

The General Social Survey is a wide-ranging survey conducted biannually to measure cultural shifts in American society. We can use the GSS to get an idea of how popular opinion has changed.

```r
GSS <- read.csv("../../Week 9/slides/GSS2016.csv")
glimpse(GSS)
```

```
Rows: 9,423
Columns: 18
$ YEAR     <int> 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2~
$ NEWSFROM <chr> "Not applicable", "Not applicable", "Not applicable", "Not ap~
$ HAPPY    <chr> "Pretty happy", "Not too happy", "Not too happy", "Not too ha~
$ RELIG    <chr> "Catholic", "None", "Catholic", "Catholic", "Protestant", "No~
$ GRASS    <chr> "Don't know", "Legal", "Not applicable", "Not legal", "Not le~
$ COURTS   <chr> "About right", "Too harsh", "Not harsh enough", "Not harsh en~
$ ENERGY   <chr> "Too little", "Too little", "Don't know", "About right", "Don~
$ EDUC     <chr> "Not applicable", "Too little", "Too little", "Not applicable~
$ ENVIR    <chr> "Not applicable", "Too little", "Too little", "Not applicable~
$ POLVIEWS <chr> "Slightly liberal", "Liberal", "Don't know", "Liberal", "Slig~
$ PARTYID  <chr> "Democrat", "Democrat", "Democrat", "Republican", "Independen~
$ REGION   <chr> "Middle atlantic", "Middle atlantic", "Middle atlantic", "Mid~
$ INCOME   <chr> "$25000 or more", "$15000 - 19999", "$20000 - 24999", "$8000 ~
$ SEX      <chr> "Male", "Female", "Female", "Female", "Female", "Male", "Fema~
$ DEGREE   <chr> "Bachelor", "Bachelor", "Lt high school", "Lt high school", "~
$ AGE      <chr> "31", "23", "71", "82", "78", "40", "46", "80", "31", "No ans~
$ MARITAL  <chr> "Never married", "Never married", "Divorced", "Widowed", "Mar~
$ BALLOT   <chr> "Ballot b", "Ballot b", "Ballot a", "Ballot b", "Ballot c", "~
```

Let's Clean Our Data! Yay!

- Let's only look at one year, say 2016, and remove "Not applicable from our response"

```r
GSS <- GSS %>% filter(YEAR==2016) %>%
  filter(GRASS != 'Not applicable')
```

- Want just two groups for responses: Legal and Not legal

```r
GSS <- GSS %>%
mutate(LEGAL = ifelse(GRASS=='Legal', 'Legal', 'Not legal'))
```

- Change variables to proper type

```r
GSS$AGE <- as.numeric(GSS$AGE)
```

```
Warning: NAs introduced by coercion
```

```
head(GSS)
```

```
  YEAR       NEWSFROM        HAPPY    RELIG     GRASS           COURTS
1 2016             Tv Pretty happy     None     Legal        Too harsh
2 2016 Not applicable   Very happy Catholic Not legal       Don't know
3 2016 Not applicable   Very happy     None     Legal       Don't know
4 2016          Radio   Very happy     None     Legal Not harsh enough
5 2016 Not applicable   Very happy Catholic Not legal Not harsh enough
6 2016 Not applicable Pretty happy     None Not legal Not harsh enough
       ENERGY           EDUC          ENVIR             POLVIEWS     PARTYID
1  Too little     Too little     Don't know              Liberal Independent
2  Too little     Too little    About right         Conservative  Republican
3 About right Not applicable Not applicable     Slightly liberal    Democrat
4  Too little     Too little     Too little     Slightly liberal    Democrat
5 About right     Too little     Too little Slghtly conservative Independent
6 About right Not applicable Not applicable         Conservative  Republican
           REGION         INCOME    SEX         DEGREE AGE       MARITAL
1     New england $25000 or more   Male    High school  61 Never married
2     New england $25000 or more   Male       Bachelor  72       Married
3     New england        Refused Female       Graduate  55       Married
4     New england $25000 or more Female Junior college  53       Married
5 Middle atlantic $25000 or more Female    High school  23       Married
6 Middle atlantic $25000 or more   Male Junior college  71      Divorced
    BALLOT     LEGAL
1 Ballot b     Legal
2 Ballot c Not legal
3 Ballot c     Legal
4 Ballot b     Legal
5 Ballot c Not legal
6 Ballot c Not legal
```

Testing data v. training data

Goal: Use Age to predict people's opinion of marijuana legalization.

```
set.seed(365)
test_id <- sample(1:nrow(GSS), size=round(0.4*nrow(GSS)))
TEST <- GSS[test_id,]
TRAIN <- GSS[-test_id,]
```

How many people in the training data set support marijuana legalization?

```
TRAIN %>% group_by(LEGAL) %>% summarize(n=n())
```

```
# A tibble: 2 x 2
  LEGAL         n
  <chr>     <int>
1 Legal       694
2 Not legal   480
```

Decision Trees

Decision trees: A tree-like model of decisions and their possible consequences

- Has flowchart-like structure in which each…
  - Internal node represents a "test" on an attribute (decision node),
  - Branch represents the outcome of the test,
  - Leaf node represents a class label (decision taken after computing all attributes).
- The paths from root to leaf represent classification rules.
- Can be applied on both regression and classification problems.

Decision Trees (Classification)

We predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.

How to decide to split? + Gini Index:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

where $\hat{p}_{mk}$ is the proportion of training observations in the mth region that are from the kth class + Gini Index is a measure of node purity (a pure node contains observations from a single class)

```
+ used to evaluate quality of a split (create a split if makes the node more pure)
```

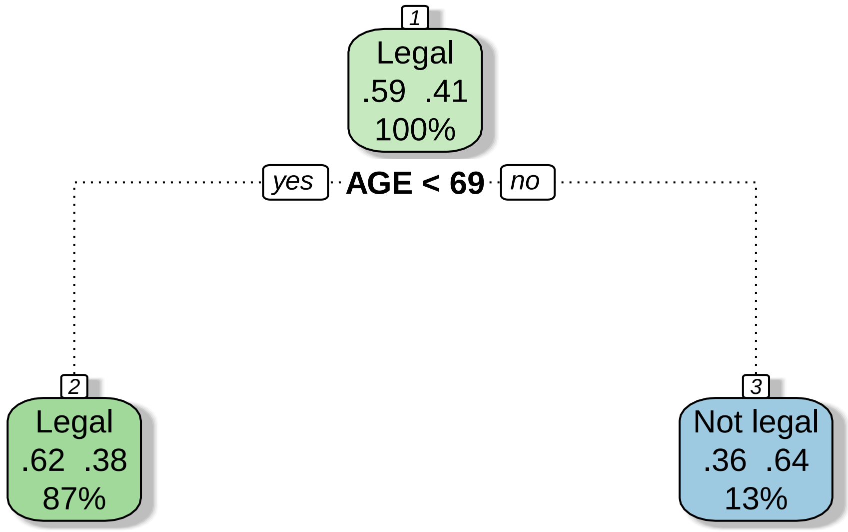
Fitting A Decision Tree (Classification)

```r
#install.packages('rpart')
library(rpart)
rpart(LEGAL~AGE, data=TRAIN, na.action = na.pass)
```

```
n= 1174

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 1174 480 Legal (0.5911414 0.4088586)
  2) AGE< 68.5 1026 385 Legal (0.6247563 0.3752437) *
  3) AGE>=68.5 148  53 Not legal (0.3581081 0.6418919) *
```

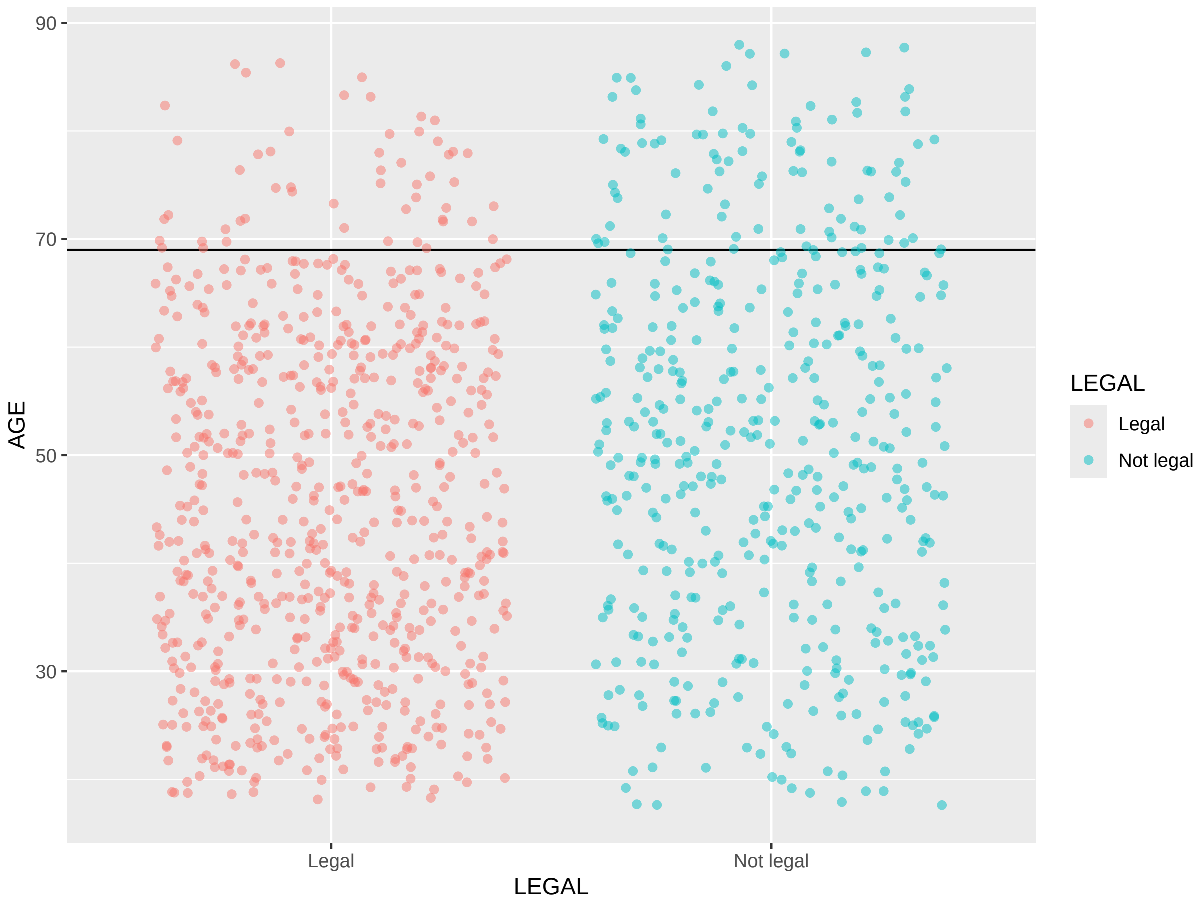
Visualizing a Decision Tree (Classification)

```r
#install.packages("rattle")
library(rattle)
tree <- rpart(LEGAL~AGE, data=TRAIN, na.action = na.pass)
fancyRpartPlot(tree)
```

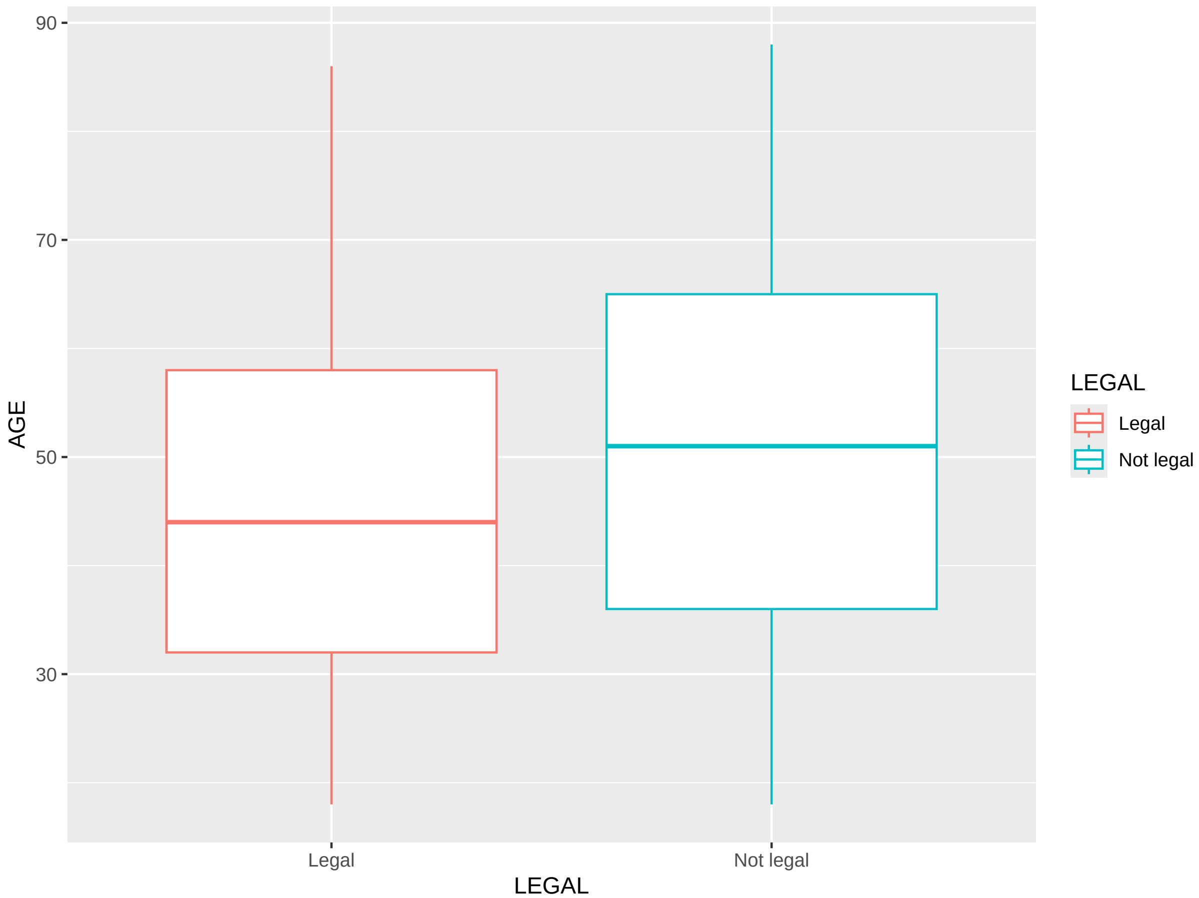


Rattle 2024−Aug−15 15:00:04 akl95311

```
TRAIN %>% ggplot(aes(x=LEGAL, y=AGE)) +
  geom_hline(yintercept=69, col='black') +
  geom_jitter(alpha=0.5, aes(col=LEGAL))
```




```
TRAIN %>% ggplot(aes(x=LEGAL, y=AGE)) +
geom_boxplot(aes(col=LEGAL))
```



Evaluating a decision tree: the three C's

- Complexity parameter
- Confusion Matrix
- Classification Accuracy

Complexity parameter

It is the amount by which splitting that node improved the relative error. - So splitting that node only resulted in an improvement of 0.01, so the tree building stopped there

```
printcp(tree)
```

```
Classification tree:
```

```
rpart(formula = LEGAL ~ AGE, data = TRAIN, na.action = na.pass)

Variables actually used in tree construction:
[1] AGE

Root node error: 480/1174 = 0.40886

n= 1174

      CP nsplit rel error xerror     xstd
1 0.0875      0    1.0000 1.0000 0.035093
2 0.0100      1    0.9125 0.9125 0.034522
```

Confusion Matrix

```
TRAIN <- TRAIN %>%
  mutate(Legal_Tree = predict(tree, type='class'))

confusion_train <- tally(Legal_Tree~LEGAL, data=TRAIN)
confusion_train
```

```
           LEGAL
Legal_Tree  Legal Not legal
  Legal       641       385
  Not legal    53        95
```

```
TEST <- TEST %>%
  mutate(Legal_Tree = predict(tree, type='class', newdata = TEST))

confusion_test <- tally(Legal_Tree~LEGAL, data=TEST)
confusion_test
```

```
           LEGAL
Legal_Tree  Legal Not legal
  Legal       391       281
  Not legal    41        69
```

Classification Accuracy

Training Accuracy:

```
sum(diag(confusion_train))/nrow(TRAIN)
```

```
[1] 0.6269165
```

Testing Accuracy:

```
sum(diag(confusion_test))/nrow(TEST)
```

```
[1] 0.5882353
```

Decision Trees (Regression)

1. We divide the predictor space — that is, the set of possible values for $X_1, X_2, \ldots, X_p$ — into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

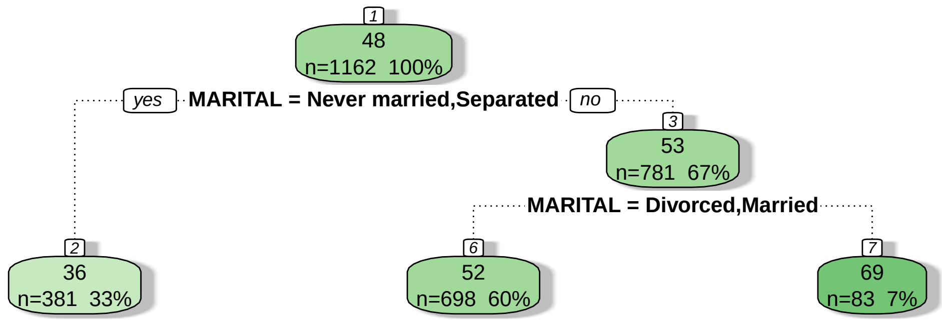
How to decide to split?

Find regions (R_j) that minimizes the residual sum of squares ($RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$). Stops when you reach some criterion.

Fitting A Decision Tree (Regression)

Let's suppose we want to use people's political view (POLVIEWS) and marital status (MARITAL) to estimate people's age.

```
tree2 <- rpart(AGE~POLVIEWS+MARITAL, data=TRAIN)
fancyRpartPlot(tree2)
```



Rattle 2024−Aug−15 15:00:05 akl95311

Prediction for Decision Regression Tree

We can still use the predict function to predict our regression decision tree outputs. Can then find the RMSE using these predictions. Can also print out the complexity parameter information to assess fit.

```
TEST <- TEST %>% filter(MARITAL != "No answer")

predict(tree2, TEST , method = "anova") %>% head()
```

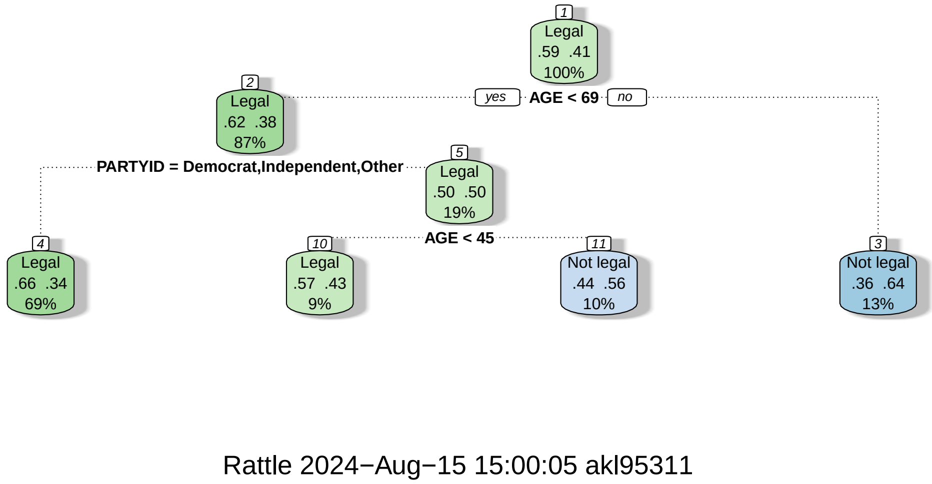
```
       1        2        3        4        5        6 
51.60315 51.60315 36.39895 51.60315 51.60315 36.39895 
```

Be Careful: Can only predict using categorical variables located in the Training Set

Try by Yourself

What if we try to use both age and political affiliation to predict the view on marijuana legalization? Visualize the tree and calculate the classification accuracy.

```
tree3 <- rpart(LEGAL~AGE+PARTYID, data=TRAIN)
fancyRpartPlot(tree3)
```




```
TRAIN <- TRAIN %>%
  mutate(Legal_Tree = predict(tree3, type='class'))

confusion_train <- tally(Legal_Tree~LEGAL, data=TRAIN)
```

```
TEST <- TEST %>%
  mutate(Legal_Tree = predict(tree3, type='class', newdata = TEST))

confusion_test <- tally(Legal_Tree~LEGAL, data=TEST)
```

Training Accuracy:

```
sum(diag(confusion_train))/nrow(TRAIN)
```

```
[1] 0.6379898
```

Testing Accuracy:

```
sum(diag(confusion_test))/nrow(TEST)
```

```
[1] 0.6197183
```

Trees Versus Linear Models

- If the relationship between the features and the response is well approximated by a linear model, then an approach such as linear regression will likely work well, and will outperform a method such as a regression tree that does not exploit this linear structure.
- If instead there is a highly nonlinear and complex relationship between the features and the response as indicated by model, then decision trees may outperform classical approaches
- But should also consider other things like testing error and interpretability

Advantages and Disadvantages of Decision Trees

- Easy to explain to people
  - Can visualize
  - Some people believe that it mirrors human decision-making
- Can handle qualitative predictors with dummy variables
- However, they generally do not have the same level of predictive accuracy as other approaches
  - Can approve prediction accuracy by aggregating many trees!

Random Forests

A random forest is collection of decision trees that are aggregated by majority rule

Random forest will expect you to have a relatively large number of input variables.

Example: Which variables are most important for predicting views on marijuana legalization?

When to use random forest

1. When there are a lot of variables and you have no idea why one may be useful to explain the response variable.
2. Potential collinearity in the predictors.

Once the random forest tells you several potential important variables, you can try to fit linear model or decision tree for interpretation

Building a Random Forest

In building a random forest, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.

- A fresh sample of m predictors is taken at each split
- Typically we choose $m \approx \sqrt{p}$

Hence, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors. Why is this a good thing?

- Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors
- By forcing each split to consider only a subset of the predictors, some splits will not even consider the strong predictor, so other predictors will have more of a chance.
  - Decorrelating the trees - making the average of the resulting trees less variable and hence more reliable.

```
#install.packages('randomForest')
library(randomForest)

forest_grass <- randomForest(as.factor(LEGAL)~NEWSFROM+HAPPY+
                               RELIG+COURTS+ENERGY+EDUC+ENVIR+
                               POLVIEWS+PARTYID+REGION+INCOME+
                               SEX+DEGREE+AGE+MARITAL+BALLOT,
                             data=TRAIN, na.action = na.omit,
                             ntree=201, mtry=4)

forest_grass
```

```
Call:
 randomForest(formula = as.factor(LEGAL) ~ NEWSFROM + HAPPY +      RELIG + COURTS + ENERGY +
               Type of random forest: classification
                     Number of trees: 201
No. of variables tried at each split: 4

        OOB estimate of  error rate: 36.75%
Confusion matrix:
          Legal Not legal class.error
Legal       528       163   0.2358900
Not legal   264       207   0.5605096
```

Random Forests: Prediction

```
TEST <- TEST %>%
  mutate(Legal_RF = predict(forest_grass, type='class',
                            newdata = TEST))

TEST$Legal_RF[1:5]
```

```
        1         2         3         4         5 
Not legal     Legal     Legal     Legal     Legal 
Levels: Legal Not legal
```

Variable Importance

Since each tree in a random forest uses a different set of variables, we want to keep track of which variables seem to be the most consistently influential. This is captured by the notion of importance.

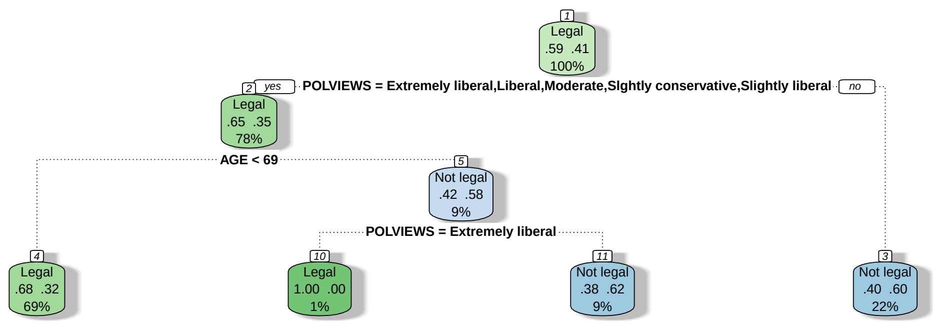
Gini is a measure of how each variable contributes to the homogeneity of the nodes and leaves in the resulting random forest (lower is more pure).

```
importance(forest_grass) %>% as.data.frame() %>%
  rownames_to_column() %>% arrange(desc(MeanDecreaseGini))
```

```
    rowname MeanDecreaseGini
1       AGE         98.99517
2  POLVIEWS         51.16884
3    REGION         49.68865
4    DEGREE         37.41108
5    INCOME         35.65191
6     RELIG         35.55887
7   MARITAL         33.76458
8    COURTS         32.47684
9   PARTYID         28.61363
10   ENERGY         27.42172
11    HAPPY         25.79851
12    ENVIR         25.38195
13     EDUC         22.04172
14 NEWSFROM         17.76259
15      SEX         16.73957
16   BALLOT         14.78124
```

Decision Tree with Selected Importance

```r
tree4 <- rpart(LEGAL~AGE+REGION+POLVIEWS, data=TRAIN)
fancyRpartPlot(tree4)
```



Rattle 2024−Aug−15 15:00:05 akl95311

Your Turn: Age

Which variables are most important for predicting ages? Use these to create a Decision Tree. Note: Check column name when you arrange the importance variables.

```r
forest_age <- randomForest(AGE~NEWSFROM+ HAPPY+RELIG+COURTS+ENERGY
                           +EDUC+ENVIR+ POLVIEWS+PARTYID+REGION+
                            INCOME+SEX+ DEGREE+LEGAL+MARITAL+BALLOT,
                           data=TRAIN,
                           ntree=201, mtry=3, na.action = na.omit)

forest_age
```

```
Call:
 randomForest(formula = AGE ~ NEWSFROM + HAPPY + RELIG + COURTS +      ENERGY + EDUC + ENVIR
               Type of random forest: regression
                     Number of trees: 201
No. of variables tried at each split: 3

          Mean of squared residuals: 214.6301
```

```
                    % Var explained: 26.32
```

```
importance(forest_age) %>% as.data.frame() %>%
  rownames_to_column() %>% arrange(desc(IncNodePurity))
```

```
    rowname IncNodePurity
1   MARITAL     71987.888
2    REGION     22675.746
3  POLVIEWS     21153.517
4    INCOME     17773.530
5    DEGREE     17247.317
6     RELIG     16593.254
7  NEWSFROM     15691.347
8   PARTYID     14172.918
9    COURTS     13872.600
10    ENVIR     12287.841
11    HAPPY     11598.273
12   ENERGY     11547.199
13     EDUC     10329.886
14    LEGAL      8537.511
15      SEX      7443.911
16   BALLOT      6319.413
```

```
tree5 <- rpart(AGE~MARITAL+REGION+POLVIEWS, data=TRAIN)
```

If Time: Iris Data

Here is the data from credit card customers. One variable that credit card companies are often interested in is utilization: how much of the available credit limit is currently being "used"?

```
data("iris")
```

1. Separate into training and testing set
2. Fit a random Forest Model (Species)- Decide variable importance
3. Using your most important variables, create a decision tree
4. Evaluate your decision Tree

```
set.seed(10)
test_id <- sample(1:nrow(iris), size=round(0.4*nrow(iris)))
```

```
TEST <- iris[test_id,]
TRAIN <- iris[-test_id,]
```

```
forest_species <- randomForest(Species~Sepal.Length + Sepal.Width+
                                 Petal.Length+Petal.Width, data=TRAIN,
                               ntree=50, mtry=2, na.action =na.omit)

forest_species
```

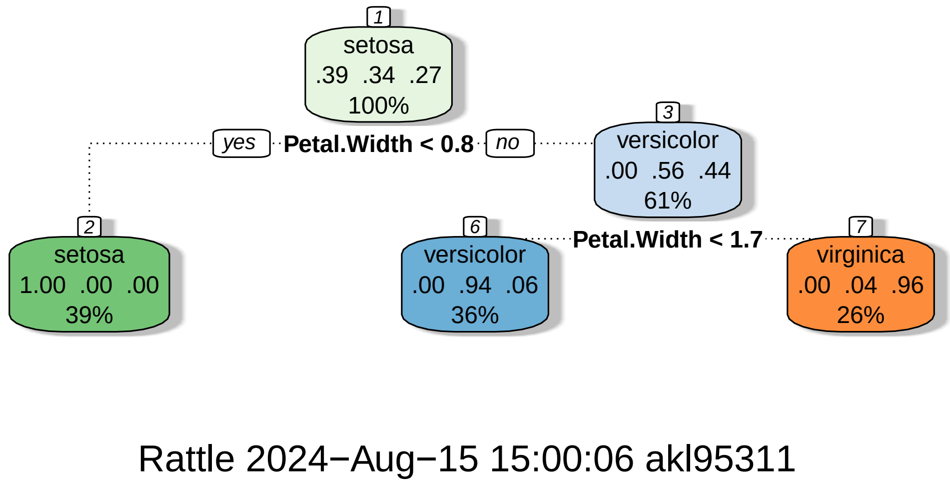
```
Call:
 randomForest(formula = Species ~ Sepal.Length + Sepal.Width +      Petal.Length + Petal.Wid
               Type of random forest: classification
                     Number of trees: 50
No. of variables tried at each split: 2

        OOB estimate of  error rate: 5.56%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         35          0         0  0.00000000
versicolor      0         29         2  0.06451613
virginica       0          3        21  0.12500000
```

```
importance(forest_species) %>% as.data.frame() %>%
  rownames_to_column() %>% arrange(desc(MeanDecreaseGini))
```

```
       rowname MeanDecreaseGini
1 Petal.Length        29.035207
2  Petal.Width        21.343675
3 Sepal.Length         6.047485
4  Sepal.Width         1.862966
```

```
tree6 <- rpart(Species ~ Petal.Width + Petal.Length, data=TRAIN)
```

```
fancyRpartPlot(tree6)
```




```
TRAIN <- TRAIN %>%
  mutate(Species_Tree = predict(tree6, type='class'))

confusion_train <- tally(Species_Tree~Species, data=TRAIN)
```

```
TEST <- TEST %>%
  mutate(Species_Tree = predict(tree6, type='class', newdata = TEST))

confusion_test <- tally(Species_Tree~Species, data=TEST)
```

Training Accuracy:

```
sum(diag(confusion_train))/nrow(TRAIN)
```

```
[1] 0.9666667
```

Testing Accuracy:

```
sum(diag(confusion_test))/nrow(TEST)
```

```
[1] 0.95
```