Decision Trees and One Rule Predictor

The Part Time Economist

2024-06-08

Rules of Thumb

Detailed statistical analysis can be great for achieving maximum predictive accuracy, but sometimes we just want a quick "Rule of Thumb" that can give a reasonably good prediction for most circumstances. In this lesson, we will use R to create simple decision making rules and evaluate the power of those rules.

Getting Started

For this lesson, we need to install a few packages.

```
#install.packages("tidyverse")
#install.packages("OneR")
#install.packages("earth")
#install.packages("rpart")
#install.packages("rpart.plot")
library(OneR)
library(tidyverse)
library(earth)
library(rpart)
library(rpart.plot)
```

Visualizing the Data

Before we start the analysis, let's look at the etitanic data set using the str command. This data set contains various characteristics of passengers on the Titanic and whether or not they survived the disaster. To conduct our analysis we need to convert the survived variable (currently represented as a 0 or 1) to a factor variable.

```
str(etitanic) #View data structure
```

```
## 'data.frame':    1046 obs. of  6 variables:
##  $ pclass  : Factor w/ 3 levels "1st","2nd","3rd": 1 1 1 1 1 1 1 1 1 1 ...
##  $ survived: int  1 1 0 0 0 1 1 0 1 0 ...
##  $ sex     : Factor w/ 2 levels "female","male": 1 2 1 2 1 2 1 2 1 2 ...
##  $ age     : num  29 0.917 2 30 25 ...
##  $ sibsp   : int  0 1 1 1 1 0 1 0 2 0 ...
##  $ parch   : int  0 2 2 2 2 0 0 0 0 0 ...
```

```
etitanic$survived <- as.factor(etitanic$survived) #Convert to factor
```

Zero Rule Prediction

Remember, the goal of this lesson is to come up with some quick rules for making predictions.

The simplest type of prediction is known as a **Zero Rule** prediction in which we simply predict the average outcome. If the majority of frogs are poisonous, we will always predict that an unknown frog is poisonous. If

the stock market goes up 51% of the days it is open, we will *always* predict that the stock market goes up. In the Titanic example, we can use the table command to show that the majority of passengers did not survive; therefore, if we knew nothing else about a passenger, we would predict their non-survival.

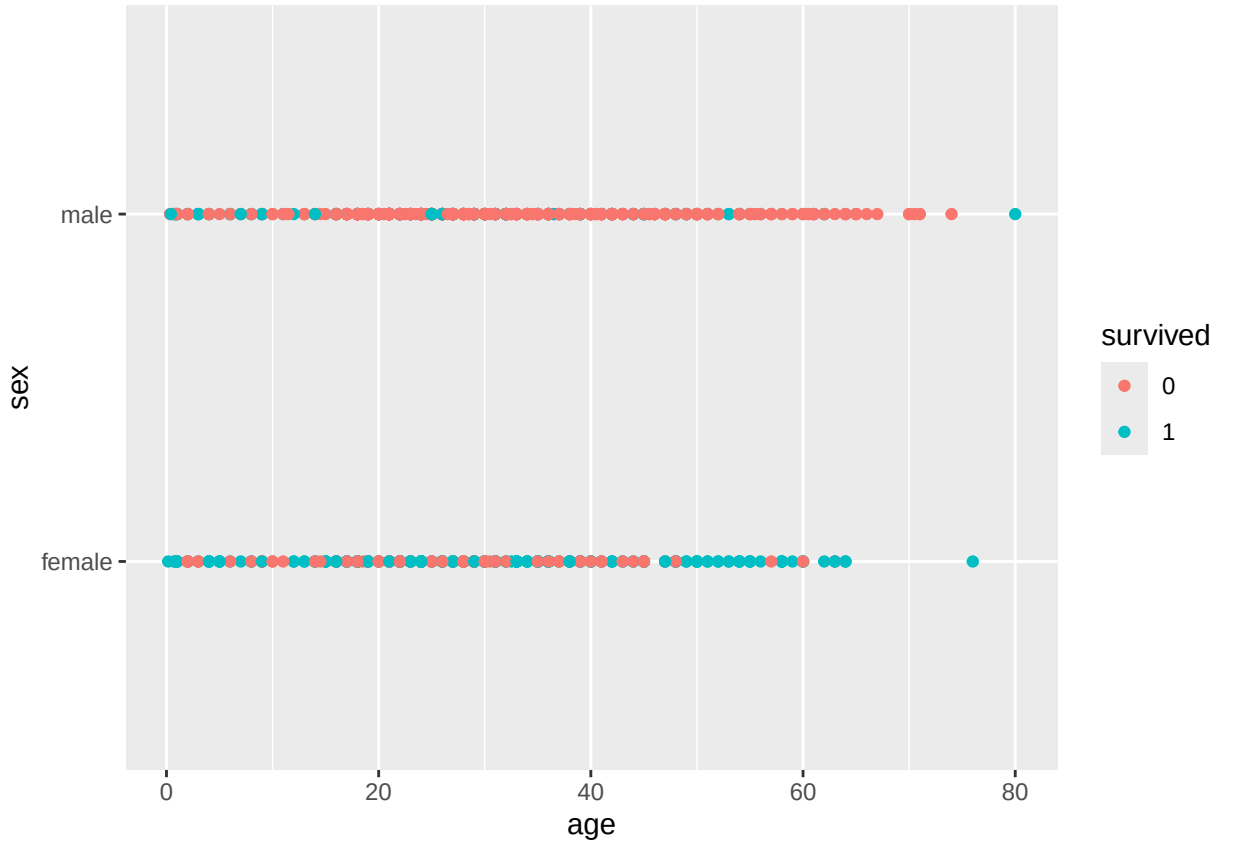
```
table(etitanic$survived) #shows count of survivors (1) vs non-survivors (0)
```

```
##
##   0   1
## 619 427
```

One Rule Prediction

Obviously, the Zero Rule predictor isn't the best way of making predictions because it completely ignores all information about the individual data point. The One Rule predictor can greatly increase our predictive accuracy by identifying the **single most important** characteristic of our data. To emphasize that point, let's visualize the etitanic data.

```
ggplot(data = etitanic)+ geom_point(aes(x = age, y = sex, color = survived))
```



Although the majority of passengers did not survive, we see that the survival rates depend on a variety of factors. Notably, the age and gender of the passengers seem to make a difference. . . so which of these factors is most important?

To generate a One Rule predictor in R, we simply use the OneR command, then pass in the dependent variable (survived in this case) followed by the data we are using. Note: the "~." simply means that we want the one rule algorithm to use all available variables when determining the most important predictor.

```
one_rule <- OneR(survived~., data = etitanic) #OneR syntax(dependent var, data set)
summary(one_rule) #use this to see accuracy of rule
```

```
##
## Call:
## OneR.formula(formula = survived ~ ., data = etitanic)
##
## Rules:
## If sex = female then survived = 1
## If sex = male   then survived = 0
##
## Accuracy:
## 815 of 1046 instances classified correctly (77.92%)
##
## Contingency table:
##          sex
## survived female  male  Sum
##      0       96 * 523  619
##      1    * 292   135  427
##      Sum    388   658 1046
## ---
## Maximum in each column: '*'
##
## Pearson's Chi-squared test:
## X-squared = 300.5, df = 1, p-value < 2.2e-16
```

As the output indicates, the most important predictor of survival is the passenger's gender. If the passenger is female, we predict survival, and if they are male, we predict mortality. Even with this simple rule, we achieve a 78% predictive accuracy. . . .but what if we want to do better?

Decision Trees with rpart

A decision tree extends the One Rule predictor by considering additional characteristics. In our example, we know that the gender of the passenger is the most important predictor of survival, but we can imagine that age and cabin class play a role as well. The rpart algorithm allows us to create a "flow chart" to easily predict outcomes.

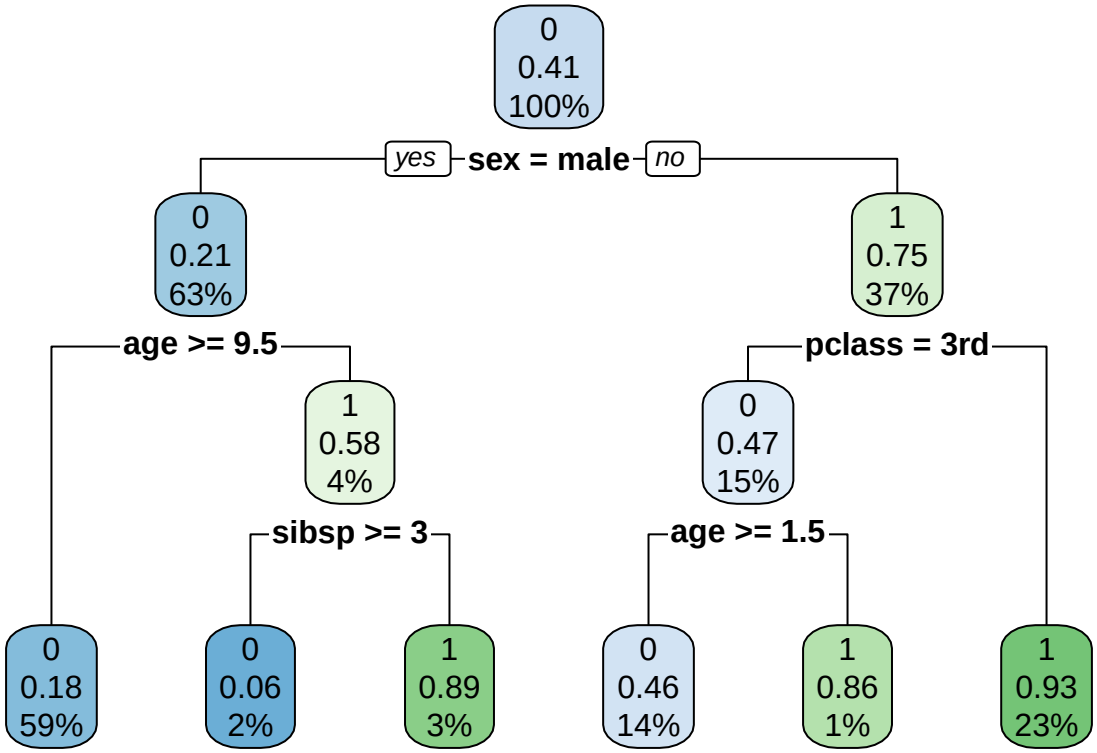
As with the One Rule algorithm, we use the command followed by the dependent variable and the data set. However, you notice an additional *maxdepth* option.The maxdepth option allows us to tell R how many predictors we want. More predictors can lead to more accurate predictions, but including too many levels makes the decision tree more complex and less of the "Rule of Thumb" than we set out to find.

The three level decision tree we will create increases accuracy to about 82% and demonstrates that although more levels are associated with more accuracy, there is diminishing returns and at some point additional complexity may not be worth the marginal improvements in accuracy.

```
decision_tree <- rpart(survived~., data = etitanic, maxdepth = 3)
#Decision Tree syntax (Dependent variable ~ Predictors, data)
#maxdepth is optional
```

By itself, the decision tree is somewhat complex, and I recommend plotting the decision tree for maximum effectiveness.

```
rpart.plot(decision_tree, type = 2)
```




```
#Plot the decision tree. There are many types of plots, but 3 is easy to interpret
```

Once the tree is plotted, we simply follow the decision splits to arrive at the predicted outcome.This is useful for individual predictions as we can quickly follow the decision tree to make relatively accurate predictions. For large numbers of predictions, we can use the predict function. I'm going to simulate some unknown data by randomly pulling observations from the data set and removing the outcome. Note that in the real world, you would NOT evaluate the model using the same data it has been trained on. This is just for example.

```
unknown_data <- etitanic%>% #This code just simulates
  sample_n(size = 10)%>%    #unknown passengers by grabbing
  select(-survived)         # 10 random and removing the outcome

#Predict command takes the decision tree we created as the first argument
#and the data we want to predict on as the second argument
predict(decision_tree, unknown_data)
```

```
##               0         1
## 174  0.82113821 0.1788618
## 5    0.06779661 0.9322034
## 243  0.06779661 0.9322034
## 1289 0.82113821 0.1788618
## 273  0.06779661 0.9322034
## 116  0.82113821 0.1788618
## 787  0.82113821 0.1788618
## 57   0.82113821 0.1788618
## 508  0.82113821 0.1788618
## 283  0.06779661 0.9322034
```

Notice that R not only gives the predicted class, but also the confidence in that prediction. Note that "confidence" in this sense is not equivalent to the confidence interval of traditional statistics.

This code is NOT part of the lesson but is provided to how the predictive accuracy of the three level decision tree was estimated.

```
library(tidymodels)
index <- initial_split(etitanic)
etitanic_train <- training(index)
etitanic_test <- testing(index)

rpart_spec <-
  decision_tree(tree_depth = 3)%>%
  set_engine('rpart') %>%
  set_mode('classification')

basic_recipe <- recipe(survived~., data = etitanic)%>%
  step_mutate(survived = as.factor(survived))

basic_workflow <- workflow(
  preprocessor = basic_recipe,
  spec = rpart_spec
)

results <- last_fit(basic_workflow, index)
results$.metrics
```

```
## [[1]]
## # A tibble: 3 x 4
##   .metric     .estimator .estimate .config
##   <chr>       <chr>          <dbl> <chr>
## 1 accuracy    binary         0.790 Preprocessor1_Model1
## 2 roc_auc     binary         0.820 Preprocessor1_Model1
## 3 brier_class binary         0.149 Preprocessor1_Model1
```