Post02: EDA + Single Decision Tree on the Titanic Dataset

Jun Seo Park

December 1, 2017

Introduction

The RMS Titanic. The greatest ship to ever set sail (at the time of its maiden voyage). Approximately 2,224 passengers and crew. And unfortunately, over 1,500 of them perished in a tragic disaster. But… who was most likely to survive? What factors played a role in determining survival on the Titanic? Can we predict an individual's fate from the person's characteristics?

In STAT 133, we've learned the basic techniques of data manipulation in R; however, many of the assignments have been structured and guided. In this blog post, we start from scratch and explore the Titanic dataset, a classic dataset that describes the ship's passengers and their fate. From a blank slate, we will work to create insights that answer the questions listed above. Lastly, we use our cleaned dataset to build a model that could help us to predict survival.

Data Loading

The dataset we will be using is a modified version of the Kaggle Titanic dataset, and is available in the Github repository. Although R includes a Titanic dataset by default, it is limited in comparison to the Kaggle equivalent and thus is not as suitable for our needs.

First, we will need the packages `readr`, `dplyr`, and `ggplot2`. Then we read in the dataset, and take a sneak peek at the data:

```
require(tidyverse)

titanic <- read_csv('titanic.csv')

glimpse(titanic)
```

```
Observations: 891
Variables: 12
$ passenger_ID <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15...
$ survived     <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0...
$ class        <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3...
$ name         <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Br...
$ sex          <chr> "male", "female", "female", "female", "male", "ma...
$ age          <chr> "22", "38", "26", "35", "35", "??", "54", "2", "2...
$ sib_sp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4...
$ par_ch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1...
$ ticket       <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113...
$ fare         <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583,...
$ cabin        <chr> "??", "C85", "??", "C123", "??", "??", "E46", "??...
$ embarked     <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C",...
```

Wow - we have a lot of data to work with. Let's go through the variables (data dictionary available on Kaggle):

- `passenger_ID` : an ID number for each passenger
- `survived` : 1 for survived, 0 for perished
- `class` : Socioeconomic class; 1 = Upper, 2 = Middle, 3 = Lower
- `name` : Name
- `sex` : Gender
- `age` : Age, in years
- `sib_sp` : Number of siblings/spouses aboard
- `par_ch` : Number of parents/children aboard
- `ticket` : Ticket number
- `fare` : Money paid for ride fare
- `cabin` : Cabin number
- `embarked` : Location embarked; C = Cherbourg, Q = Queenstown, S = Southampton

Data Cleaning

The first step in any data cleaning process should be to look through the dataset and identify any anomalies. We could use `head(titanic)`, but this may not reveal any patterns that happen beyond the first six observations. Alternatively, we can use `View(titanic)` to scan through the entire dataset - at which point we notice that there are `??` values in place of what should probably be `NA` (to signify a missing datapoint). This is an easy fix:

```
titanic[titanic=='??'] <- NA
```

Next, we look at the variable types. The command `glimpse(titanic)` earlier revealed each column's data type, but `read_csv` coerces all data to numeric or characters by default. We can refer to the dictionary and/or use intuition to see which variables might need to be coerced into a different data type. In this Titanic dataset, `survived` should be a logical, and `age` should be a number. Furthermore, `class`, `sex` and `embarked` should be factors (since they are categorical variables).

```
titanic$survived <- as.logical(titanic$survived)

titanic$age <- as.numeric(titanic$age)

titanic$sex <- as.factor(titanic$sex)
titanic$embarked <- as.factor(titanic$embarked)
titanic$class <- as.factor(titanic$class)
levels(titanic$class) <- c('Upper', 'Middle', 'Lower')
```

Note 1: It seems like `age` should have been read in as a numeric - isn't age in years in numbers? What actually happened though, was that the entire column was read in as characters due to the presence of the `??` values. This is an example of why it's important to review the entire dataset after reading it into R. Alternatively, this particular issue could have been avoided if we had passed the parameter `na='??'` to `read_csv`; however, before reading in the CSV, we had no way of knowing that NA values were coded as `??` (unless we had opened the CSV file beforehand using a text editor - but who ever does that?)

Note 2: `survived` is technically also a categorical variable. However, it's imperative that we classify it as a logical; the reasoning behind this decision will be revealed in a later section.

We won't be using the `ticket` and `cabin` variables since they have too many missing values and they don't seem helpful (at least, for the purposes of our analysis; these variables can actually be used for deeper analysis beyond our scope).

```
titanic <- select(titanic, -ticket, -cabin)
```

Also, let's combine the sibling-spouses variable (`sib_sp`) and parents-children variable (`par_ch`) into one variable called `fam` for simpler analysis.

```
titanic <- mutate(titanic, fam=sib_sp+par_ch)
```

Lastly, there's one limitation with the dataset - there are a lot of missing values in the `age` column. There are several ways to work around this; one of them is called missing value imputation, in which we would use data from other columns to estimate what a value might equal. That's a little too advanced for our purposes; for simplicity, we'll just omit every row that has an `NA` value in any column. This isn't the best technique (particularly when there are this many `NA` values), but this is simply for the purposes of our blog post. Moving forward, keep in mind that our dataset is no longer fully representative of the Titanic's passengers.

```
titanic <- na.omit(titanic)
```

Now we revisit the data frame structure, and find that the data is formatted to our liking:

```
glimpse(titanic)
```

```
Observations: 712
Variables: 11
$ passenger_ID <int> 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1...
$ survived     <lgl> FALSE, TRUE, TRUE, TRUE, FALSE, FALSE, FALSE, TRU...
$ class        <fctr> Lower, Upper, Lower, Upper, Lower, Upper, Lower,...
$ name         <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Br...
$ sex          <fctr> male, female, female, female, male, male, male, ...
$ age          <dbl> 22, 38, 26, 35, 35, 54, 2, 27, 14, 4, 58, 20, 39,...
$ sib_sp       <int> 1, 1, 0, 1, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 1...
$ par_ch       <int> 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0...
$ fare         <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 51.8625...
$ embarked     <fctr> S, C, S, S, S, S, S, S, C, S, S, S, S, S, S, Q, ...
$ fam          <int> 1, 1, 0, 1, 0, 0, 4, 2, 1, 2, 0, 0, 6, 0, 0, 5, 1...
```

Data analysis

Data scientists often say that data science projects should always start with a question. That's definitely true, and to find the answer to that overarching question, we can start off with small questions like: what is the average age of the passengers aboard the Titanic?

```
mean(titanic$age)
```

```
[1] 29.64209
```

Seems about right; young adulthood would be a great time to take a cruise ship. Alright, then let's jump right into the meat of the dataset - we want to know the overall survival rate.

```
mean(titanic$survived)
```

```
[1] 0.4044944
```

Remember when we set `survived` to be a logical? Here's where it comes back into play; since logicals are technically numerics stored as 0's and 1's, the entire column will sum up to be the total number of people who survived. Then, taking the mean will give us the survival rate. Note that this is not possible if we had changed it to a factor, because the `mean` function requires either a numeric or logical as input.

Now here comes the interesting part: what if we group the data by socioeconomic class, or by gender, or by number of family members? Are people of higher socioeconomic status more likely to survive? Does gender make a difference? Family size? Let's see:

```
summarize(group_by(titanic, class), mean(survived))
```

```
# A tibble: 3 x 2
   class `mean(survived)`
  <fctr>            <dbl>
1  Upper        0.6521739
2 Middle        0.4797688
3  Lower        0.2394366
```

```
summarize(group_by(titanic, sex), mean(survived))
```

```
# A tibble: 2 x 2
     sex `mean(survived)`
  <fctr>            <dbl>
1 female        0.7528958
2   male        0.2052980
```

```
summarize(group_by(titanic, fam), mean(survived))
```

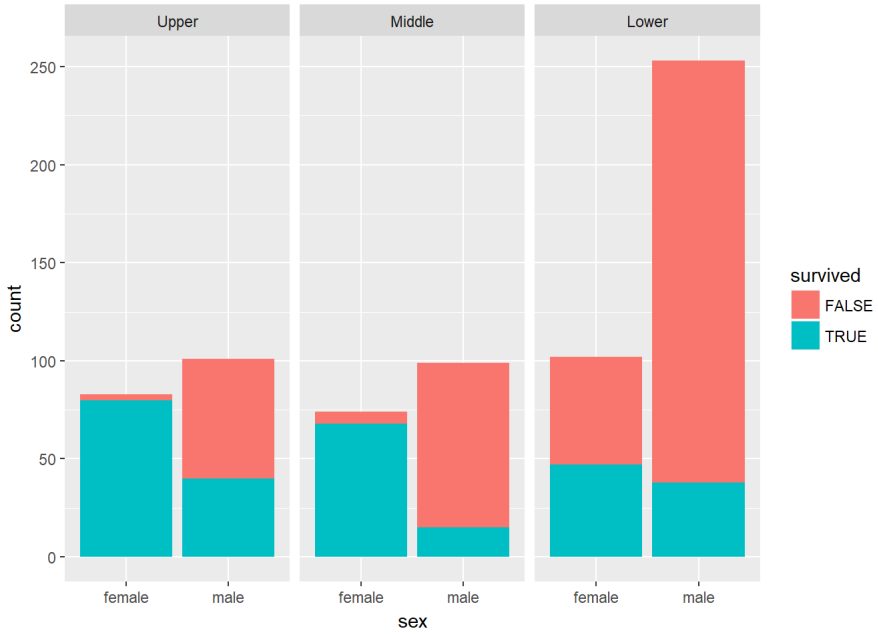
```
# A tibble: 8 x 2
    fam `mean(survived)`
  <int>            <dbl>
1     0        0.3184080
2     1        0.5467626
3     2        0.5698925
4     3        0.7777778
5     4        0.2727273
6     5        0.1363636
7     6        0.3333333
8     7        0.0000000
```

The higher the socioeconomic class, the higher the rate of survival. And women have a much higher rate of survival than men (makes sense - women and children first). For families, it looks like people with 1-3 family members aboard had the highest rate of survival. Now, let's try visualizing our findings using `ggplot2`.

Data Visualization

We can use bar charts and color by factor to observe the interactions between three variables. Let's start off with the relationship between survival rate, socioeconomic class, and gender.

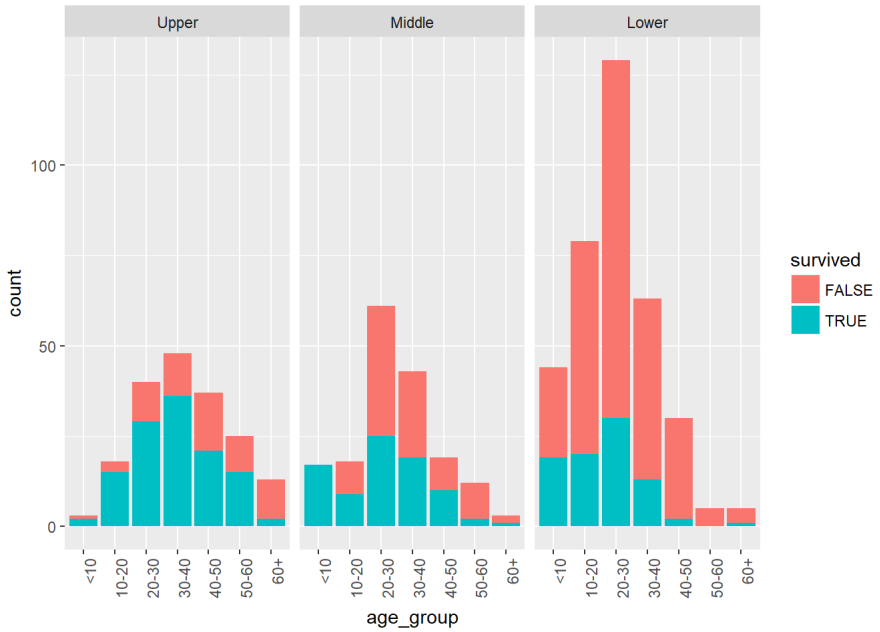
```
ggplot(titanic, aes(x=sex, fill=survived)) +
  geom_bar() +
  facet_grid(~ class)
```



Unsurprisingly, in all three socioeconomic classes, women have a much higher rate of survival than men. But it looks like upper and middle class women are almost guaranteed to live, whereas women of lower socioeconomic status did not share the same conditions.

Next, we want to see the effects of age. But we'll need to make some adjustments first - we want to categorize age into groups, and then we'll replace sex in the chart above with age.

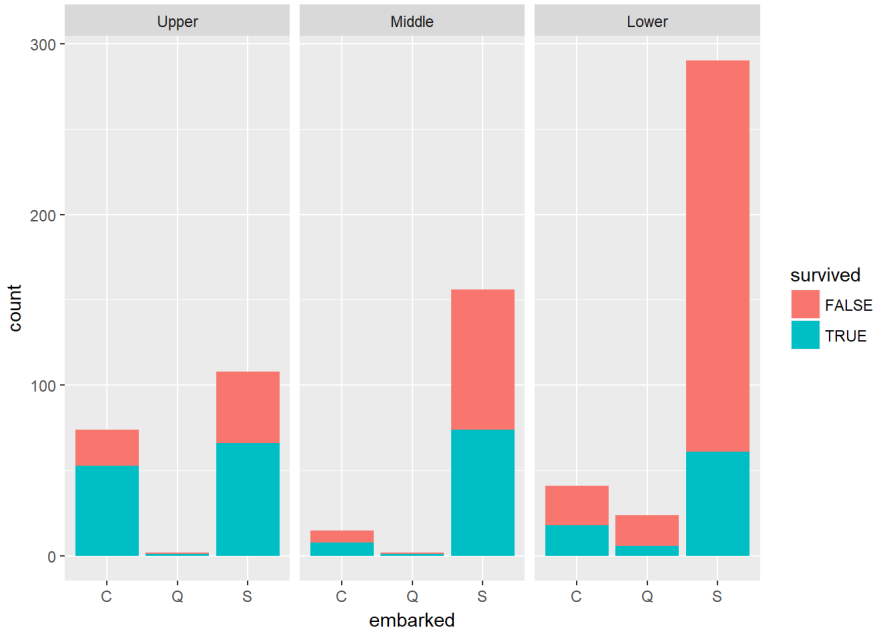
```
titanic$age_group <- ''
for (i in 1:nrow(titanic)) {
  if (titanic$age[i] <= 10) titanic$age_group[i] <- '<10'
  else if (titanic$age[i] <= 20) titanic$age_group[i] <- '10-20'
  else if (titanic$age[i] <= 30) titanic$age_group[i] <- '20-30'
  else if (titanic$age[i] <= 40) titanic$age_group[i] <- '30-40'
  else if (titanic$age[i] <= 50) titanic$age_group[i] <- '40-50'
  else if (titanic$age[i] <= 60) titanic$age_group[i] <- '50-60'
  else titanic$age_group[i] <- '60+'
}

ggplot(titanic, aes(x=age_group, fill=survived)) +
  geom_bar() +
  facet_grid(~ class) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Looks like socioeconomic class is the prevailing determinant of survival, and then younger people seem to have been favored (even in the lower socioeconomic class).

What about point of embarkment?

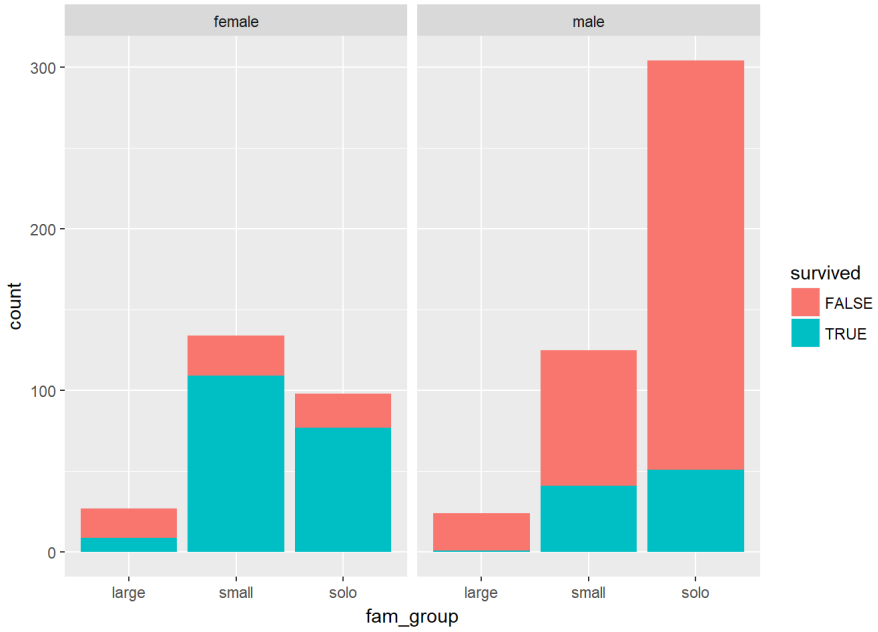
```
ggplot(titanic,  aes(x=embarked, fill=survived)) +
  geom_bar() +
  facet_grid(~ class)
```



Doesn't look like there's much to see here. Distributions seem relatively constant throughout, perhaps with the exception of Southampton in the lower class - but that's probably just because it was the lower class.

Lastly, let's look at family size. Let's group it into 3 groups: solo riders, small families (1-3 family members), and large families (4+).

```
titanic$fam_group <- ''
titanic$fam_group[titanic$fam == 0] <- 'solo'
titanic$fam_group[titanic$fam >= 1 & titanic$fam < 4] <- 'small'
titanic$fam_group[titanic$fam >= 4] <- 'large'

ggplot(titanic, aes(x=fam_group, fill=survived)) +
  geom_bar() +
  facet_grid(~ sex)
```



Interesting - large families really get the shorter end of the stick, and small families see the most benefit in terms of survival rates.

So from our exploratory data analysis, we've found that `sex`, `class`, and `fam_group` are three key factors in determining survival rate aboard the Titanic. The other variables, it seems we can leave out - some of them are pure identifiers (e.g. `name` and `passenger_ID`), `fare` is probably analagous to `class`, and `embarked` seems inconclusive based on our analysis. Now, let's get to the fun part: predictions.

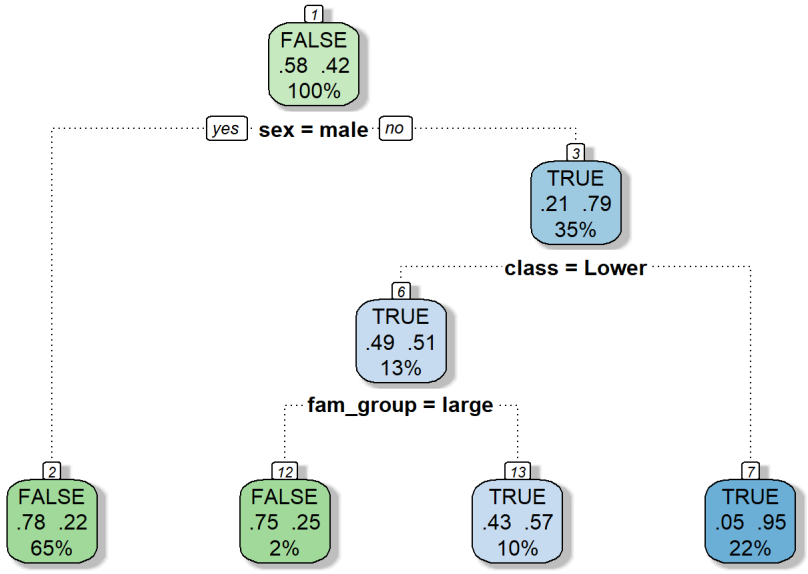
Single Decision Tree

We can use a decision tree to build a simple model of survival. The first step is to set aside 30% of the data to validate our model later on; we'll be using the remaining 70% to train the model.

```
require(rpart)
require(rattle)
require(rpart.plot)
require(RColorBrewer)
require(caret)

set.seed(420)
sample_size <- 0.7 * nrow(titanic)
indices <- sample(seq_len(nrow(titanic)), size=sample_size)
train <- titanic[indices, ]
test <- titanic[-indices, ]
```

Next, we fit a model to the three aforementioned variables: `sex`, `class`, and `fam_group`. the `rpart` function does this for us - all we do is pass in the variables that we want to build our model upon, and it'll show us the best decision tree from these variables.

```
fit <- rpart(survived~sex + class + fam_group, data=train, method='class')
fancyRpartPlot(fit, sub='')
```



Looks like the biggest indicator of survival is sex; men are predicted to die. The next step is to determine whether women are in the lower socioeconomic class; if so, they are predicted to die, whereas if not, the final step in the decision tree is to determine their family size. Women from large families are predicted to die, whereas women from small families or traveling solo are predicted to live.

Lastly, we'll use the `predict` function to obtain predictions for the 30% of the data that we had set aside, then compare our predictions with the actual fate of the passengers to see how accurate we can get.

```
pred <- predict(fit, test, type='class')
confusionMatrix(pred, test$survived)
```

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE   116   21
     TRUE     19   58

               Accuracy : 0.8131
                 95% CI : (0.7543, 0.863)
    No Information Rate : 0.6308
    P-Value [Acc > NIR] : 5.001e-09

                  Kappa : 0.5966
 Mcnemar's Test P-Value : 0.8744

            Sensitivity : 0.8593
            Specificity : 0.7342
         Pos Pred Value : 0.8467
         Neg Pred Value : 0.7532
             Prevalence : 0.6308
         Detection Rate : 0.5421
   Detection Prevalence : 0.6402
      Balanced Accuracy : 0.7967

       'Positive' Class : FALSE
```

Wow! We got an accuracy of 0.8131, which isn't perfect but still quite impressive for a simple model.

Conclusion

We began with a barebones dataset with missing values and variables that weren't quite exactly what we were looking for. However, by applying techniques we learned in STAT 133 as well as new techniques from intuition, we were able to create a dataset that met our needs. Next, we were able to explore the dataset and understand the structure of the data, allowing us to grasp the kind of data that we had at our hands. This is important - it is unwise to blindly build models, and rather it is vital that as data scientists we understand our data before our computer does. Lastly, using this knowledge, we built a very simple model that predicted survival fates of passengers at an accuracy rate of 0.8131. It is important to note that there are many other techniques not covered in this post (such as cross-validation or random forests) that yield more statistically significant results; however, as an introduction to machine learning, this is a great accomplishment. Congratulations on building your first model!

References:

1. https://www.kaggle.com/c/titanic
2. http://trevorstephens.com/kaggle-titanic-tutorial/r-part-3-decision-trees/
3. https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/
4. https://stackoverflow.com/questions/24348973/how-to-retrive-overall-accuracy-value-from-confusionmatrix-in-r
5. https://stackoverflow.com/questions/1330989/rotating-and-spacing-axis-labels-in-ggplot2

6. https://en.wikipedia.org/wiki/Imputation_(statistics)
7. https://www.statmethods.net/advstats/cart.html