Eco Bici 2017 Decision Tree Classification

MD01 CIC IPN

Emanuel Becerra Soto

December 2018

Introduction

The dataset analyzed was a subset of the EcoBici 2017 trips data. All the trips made on 2017 are approximately 10,000,000 but after some cleaning a 20% of the data was taken for exploratory analysis.

Here we further divide this 20% percent into only the trips connecting the top 10 most common destination stations and the top 50 starting stations. For a total of ~57,000 trips.

The classification question was the following one: Given the starting point of a trip and some other information like: duration, user's sex and age and the hour and day when it was made, how feasible is to predict the ending point?

Specifically if knowing that the trip started on one of the top 50 most used stations and ended on one of the top 10. Hence our target classes in the classification task would be the top 10 most popular destinations.

The C5.0 algorithm an improvement over the C4.5 algorithm (Quinlan, 1993) was used for building the decision tree.

Using boosting of 100 alternative trees a final accuracy of 47% was reached.

Global Settings

```r
#! /usr/bin/env Rscript

# Emanuel Becerra Soto
# December 2018

# Setting set for reproducibility
set.seed(1631)

####### Libraries #######

library(C50)
library(tidyverse)
library(ggthemes)
library(partykit)

####### Functions #######

# Sort a numeric factor, useful for plotting
sort_numeric_factor <- function( Factor ){
  levels( Factor ) <- Factor%>%
    levels()%>%
    as.numeric()%>%
    sort()%>%
```

Figure 1: EcoBici

```r
    as.character()
  return( Factor )
}

# Subset a dataframe
partition_data <- function(data_all, train = 0.6, test = 0.2){
  validation <- 1 - (train + test)
  n <- nrow(data_all)

  n_train <- floor(n * train)
  n_test <- ceiling(n * test)
  n_validation <- n - (n_train + n_test)

  random_idx <- sample(1:n)
  idx_train <- random_idx[1:n_train]
  idx_test <- random_idx[(n_train+1) : (x <- (n_train+1 + n_test-1))]
  idx_validation <- random_idx[ (x+1) : n]

  idx_train <- sort(idx_train)
  idx_test <- sort(idx_test)
  idx_validation <- sort(idx_validation)

  data_train <- data_all[idx_train,]
  data_test <- data_all[idx_test,]
  data_val <- data_all[idx_validation,]

  return(list(data_train, data_test, data_val))
}
```

Loading and transforming the data

```r
####### Loading the data #######

file <- '2017_bikes_val_4.2.csv'
bikes <- read_csv(file)

bikes$station_start <- as.character(bikes$station_start)
bikes$station_end <- as.character(bikes$station_end)
```

```r
####### Most used stations #######

# Counting the most used stations
station_usage_start <- arrange( as.tibble( table(bikes$station_start) ), desc(n) )
station_usage_start$perct_start <- ( station_usage_start$n / sum(station_usage_start$n) )
station_usage_end <- arrange( as.tibble( table(bikes$station_end) ), desc(n) )
station_usage_end$perct_end <- ( station_usage_end$n / sum(station_usage_end$n) )

# Joining start and end stations usage percent onto one table
station_usage <- inner_join(station_usage_start, station_usage_end, by = 'Var1')
station_usage <- rename(station_usage, st_id = Var1 , n_start = n.x, n_end = n.y)
station_usage <- arrange(station_usage, desc(n_end))
```

```r
# Printing the top 10 most used stations by destination
# station_usage[1:10,]
top_10_destinations <- station_usage$st_id[1:10]

top_50_starting <- arrange(station_usage, desc(n_end) )$st_id[1:50]
```

The top 10 destinations are: 27, 266, 1, 18, 271, 43, 21, 217, 64, 25

The top 50 starting stations are: 27, 266, 1, 18, 271, 43, 21, 217, 64, 25, 36, 182, 16, 47, 15, 74, 23, 267, 38, 134, 174, 19, 146, 116, 28, 17, 136, 24, 295, 51, 41, 32, 46, 56, 54, 141, 14, 270, 52, 63, 29, 208, 261, 59, 53, 20, 7, 158, 84, 10

For the classification task, some features were hand-picked, mainly for two reasons: Some features are relatively redundant like: duration and if a trip is exceeded on more than 45 min. (EcoBici service charges a fee for exceeding the 45 min mark). For dimensionality reduction as some variables are believed to not to affect in a considerable way, like the bike's ID.

In further analysis more dimensions could be used.

```r
####### Selecting features and top 10 #######

# Feature Selection by hand
names(bikes)
```

```
##  [1] "sex"                "age"                "station_start"
##  [4] "station_end"        "bike"               "trip_time_seconds"
##  [7] "leave_month"        "leave_day"          "leave_hour"
## [10] "leave_minute"       "leave_second"       "leave_epoch"
## [13] "leave_epoch_day"    "arrive_month"       "arrive_day"
## [16] "arrive_hour"        "arrive_minute"      "arrive_second"
## [19] "arrive_epoch"       "arrive_epoch_day"   "exceeding"
## [22] "exceeding_hour"     "exceeding_type"
```

```r
hand_picked_features <- c('station_end', 'station_start', 'sex', 'age',
                          'trip_time_seconds', 'leave_month',
                          'leave_day', 'leave_hour', 'leave_minute',
                          'exceeding')
# Only some variables are keeped
bikes <- bikes[ hand_picked_features ]

# Top 10 destinations
bikes_top <- filter(bikes, station_end %in% top_10_destinations)
bikes_top <- filter(bikes_top, station_start %in% top_50_starting)

knitr::kable(bikes_top[1:10,],
caption = "10 trips")
```

Table 1: 10 trips

station_end	station_start	sex	age	trip_time_seconds	leave_month	leave_day	leave_hour	leave_minute	e
1	270	M	40	1339	ene	dom	5	47	I
18	1	M	29	173	ene	dom	7	36	I
43	27	F	29	464	ene	dom	8	0	I
64	64	F	42	334	ene	dom	8	56	I
1	1	F	26	1504	ene	dom	9	52	I
1	1	F	26	725	ene	dom	10	19	I

station_end	station_start	sex	age	trip_time_seconds	leave_month	leave_day	leave_hour	leave_minute
27	43	F	62	2127	ene	dom	10	48
21	21	M	41	1207	ene	dom	11	8
21	21	M	29	5922	ene	dom	11	10
43	23	M	39	795	ene	dom	11	39

Exploratoy analysis

The relationship of how much and which stations are connected between the selected ones was investigated.

A heat map of the Top 10 destinations vs the Top 50 starting points was made.

```r
t_10 <- table(bikes_top$station_start, bikes_top$station_end)
pt_10 <- prop.table( table(bikes_top$station_end, bikes_top$station_start), margin = 1 )

# Getting a tibble of prob of cross stations
prob_cross_stations <- as_tibble(pt_10)
prob_cross_stations <- arrange( prob_cross_stations, as.numeric(Var1), as.numeric(Var2) )
prob_cross_stations$Var1 <- factor(prob_cross_stations$Var1)
prob_cross_stations$Var2 <- factor(prob_cross_stations$Var2)

# Sorting the stations for a better plot
prob_cross_stations$Var1 <- sort_numeric_factor(prob_cross_stations$Var1)
prob_cross_stations$Var2 <- sort_numeric_factor(prob_cross_stations$Var2)

ggplot(data = prob_cross_stations, aes(x = Var2, y = Var1, fill = n))+
  geom_tile()+scale_fill_viridis_c(name='Relative Frequency')+
  ggtitle('Ecobici Top 10 Destinations vs Top 30 Starting Stations')+
  labs(x = 'Starting Station',
       y = 'Destination',
       subtitle = 'Which stations are the most connected?')+
  scale_y_discrete(labels = c('1' = '1 / Rio Sena-Rio Balsas', '18' = '18 / Reforma-Rio Rhin',
                              '21' = '21 / Reforma-Dublin', '25' = '25 / Reforma-Estocolmo',
                              '27' = '27 / Reforma-Havre', '43' = '43 / Juarez-Revillagigedo',
                              '64' = '64 / Sonora-Amsterdam', '217' = '217 / Euler-Horacio', '266' = '2
                              '271' = '271 / Av Central-J Meneses'))+
  theme_hc()+
  theme( axis.text.x = element_text(size = 4),
         legend.text = element_text(size = 8) )
```

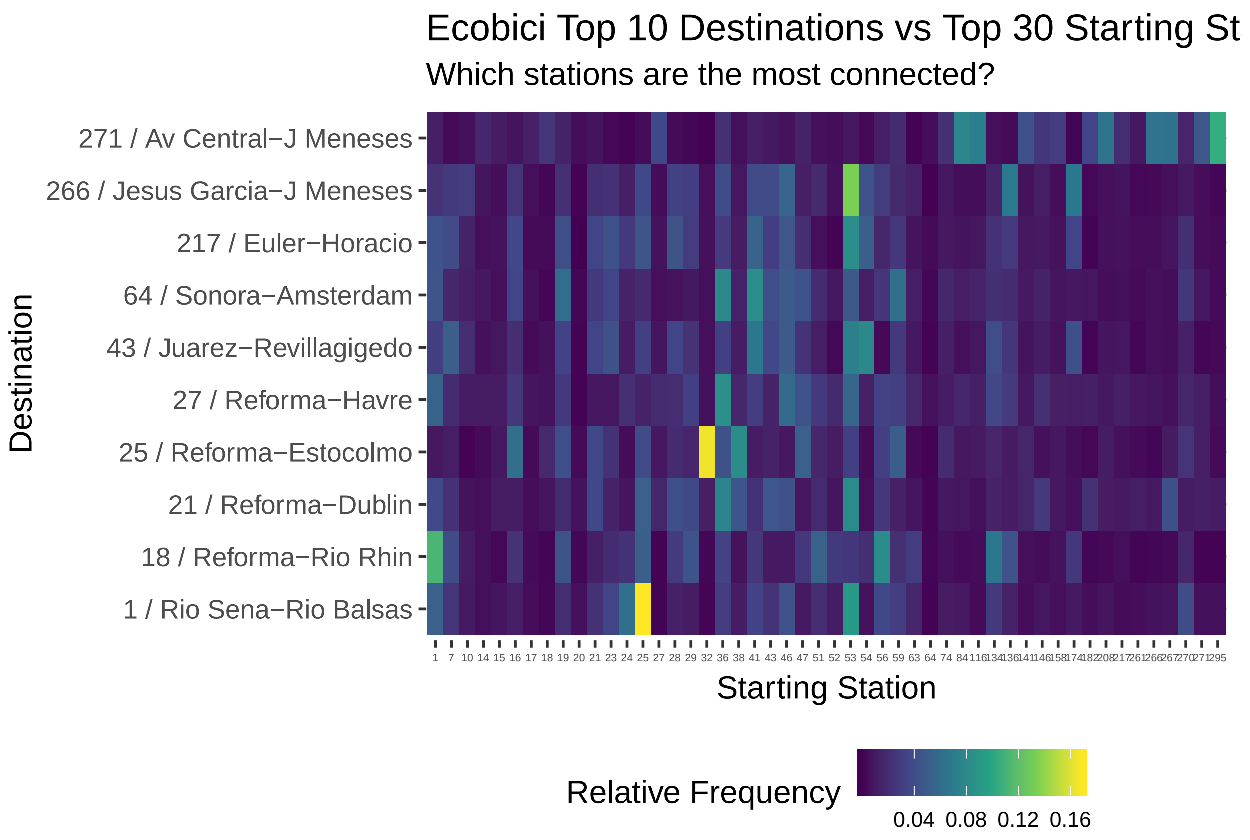

Figure 1.- The heatmap shows how the incoming traffic to a destination (Y axis) is distributed over the top 50 starting points (X axis). It could be seen that some stations account for more than 16% of the traffic to a destination. For instance the from the station 25 comes ~16% of the traffic to the station 1 similarly between 32 and 25 stations. Also the distributions of incoming traffic look relatively different from each other.

```r
bikes_top$station_end <- factor(bikes_top$station_end)
bikes_top$station_start <- factor(bikes_top$station_start)

bikes_top$station_end <- bikes_top$station_end %>%
  sort_numeric_factor()

bikes_top$station_start <- bikes_top$station_start %>%
  sort_numeric_factor()

# Barplot End counts
ggplot(data = bikes_top, aes(x=station_end, fill = station_start))+
  geom_bar()+scale_fill_viridis_d()
```

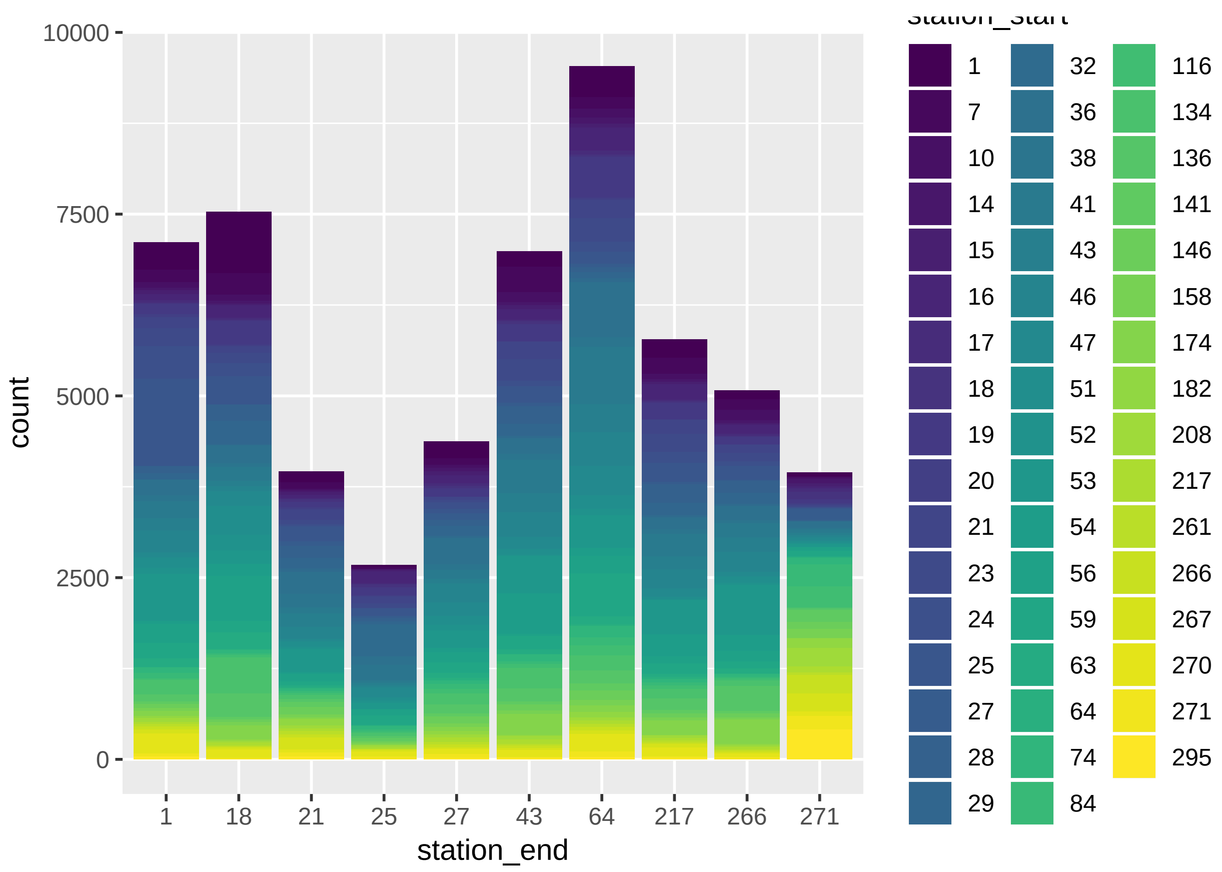

Figure 2.- A barplot of the counts of the to-be classified stations and how much traffic is incoming from the top 50. The station number doesn't mean anything as it is label for the station, but in the map stations with close numbers tend to be closer. This could appreciated in the bar plot as the station 271 has plenty (relative to the other stations) of trips incoming from the 200's stations.

Training the C5.0 tree

The data was partitioned into train and test sets, with 80% and 20% of the data respectively.

The classification tree was obtained from the C5.0 algorithm with a boosting of 100 trees.

The variables used for the classification were: The starting station, sex , age, duration of the trip, the hour and the day.

```
####### Classification Tree #######
# partition the data before training
part_bikes_top <- partition_data(bikes_top, train = 0.8, test = 0.2)
train <- part_bikes_top[[1]]
test <- part_bikes_top[[2]]

# 2 min
# fit <- C5.0( station_end ~ station_start + sex + age +
#                trip_time_seconds + leave_hour + leave_day,
#               trials = 100,
#               data = train )

# saveRDS(fit, file = 'fit_bikes_tree_01.RDS')
```

```
fit <- readRDS('fit_bikes_tree_01.RDS')
# Results
fit
```

```
##
## Call:
## C5.0.formula(formula = station_end ~ station_start + sex + age
##  + trip_time_seconds + leave_hour + leave_day, data = train, trials = 100)
##
## Classification Tree
## Number of samples: 45592
## Number of predictors: 6
##
## Number of boosting iterations: 100 requested;  44 used due to early stopping
## Average tree size: 5463.4
##
## Non-standard options: attempt to group attributes
```

Results of the classification.

Results

```
###### In sample error ######
# Confusion Matrix on test data
predictions <- predict(fit, train)
confusion <- as.matrix ( table(predictions , train$station_end) )
confusion
```

```
##
## predictions    1   18   21   25   27   43   64  217  266  271
##         1   5118  152   70   20  100   36  142   47   39   28
##         18   124 5311   44   20  183   46  229   27   49   21
##         21    44   45 2663   18   54   10   47   19   35   35
##         25    18   19   18 1878   18    8   32   16   15   20
##         27    31   82   45   11 2777   15   56    8   11   21
##         43    43   45   34   49   44 5210   65  390  192   26
##         64   234  249   94   71  194   10 6804   23   47   72
##         217   46   42   29   31   39  236   46 4015  103   29
##         266   52   52   47   32   41   82   63   66 3574   30
##         271   15   21  104   18   50   11   39    8   11 2889
```

```
# Accuracy on training data
in_accu <- ( sum(diag(confusion)) / sum(confusion) )
in_accu
```

```
## [1] 0.8825891
```

```
###### Out of Sample Error ######
# Confusion Matrix on test data
predictions <- predict(fit, test)
confusion <- as.matrix ( table(predictions , test$station_end) )
confusion
```

```
##
## predictions    1   18   21   25   27   43   64  217  266  271
##          1    708  133   88   33   70   26  188   57   48   40
##          18   122  769   55   27  194   54  251   35   68   14
##          21    58   55  341   19   43   19   73   22   32   34
##          25    34   26   23  225   13   22   55   20   36   11
##          27    39  102   43   16  252   23   96   15   31   28
##          43    55   46   42   42   12  735   56  401  195   24
##          64   221  256   94   81  178   36 1099   37   57   77
##          217   60   44   24   27   30  292   55  437  147   23
##          266   58   62   49   44   36  110   83  123  366   15
##          271   31   23   57   14   46   12   55   15   17  513
```

```r
# Accuracy on test data
out_accu <- ( sum(diag(confusion)) / sum(confusion) )
out_accu
```

```
## [1] 0.4777154
```

The **in sample error** was **0.88**

The **out of sample error** was **0.48**

The sizable difference between the in sample error vs the out of sample suggest that our model is overfitting the training the data.

Nonetheless **0.48** is better than the expected 10% by just randomly choosing the a class.

In further analysis reducing the overfitting should be a priority.

```r
# Graph of the Confusion matrix
p_conf <- prop.table( table ( predictions, test$station_end ), margin = 1 )
p_conf_tiddy <- as_tibble(p_conf)
ggplot(data = p_conf_tiddy, aes(x = Var2, y = predictions, fill = n))+
  geom_tile()+scale_fill_viridis_c(name='Relative Frequency')+
  theme_hc()+
  theme( axis.text.x = element_text(size = 6),
         legend.text = element_text(size = 8) )+
  coord_equal()
```

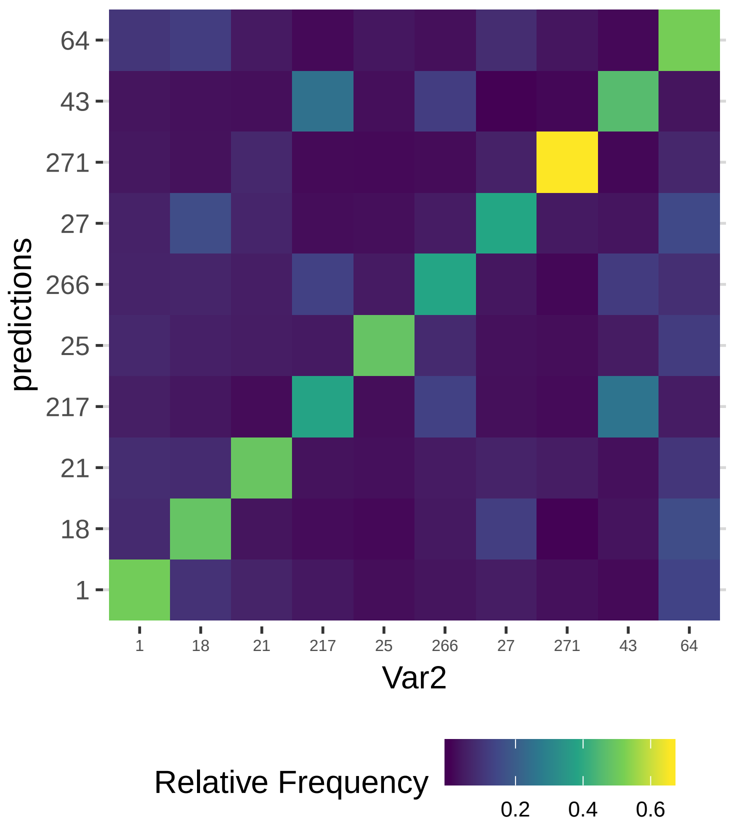

Figure 3.- Confusion matrix on the test data. The better performing prediction are the ones over station 271.

The final tree was made of 12,126 nodes, of which 5,382 are internal and 6,744 are leaf nodes.

```
########### Ploting the tree ###########

# Changing the tree to a better format
# 15 min
# party_fit <- as.party.C5.0(fit)
# saveRDS(party_fit, file = 'party_format_bikes_tree_01.RDS')
party_fit <- readRDS('party_format_bikes_tree_01.RDS')

# Writing the tree to an external file
# capture.output( print(party_fit[1]), file = 'tree_bikes_top_10.txt' )

# Number of inner nodes: 5382
# Number of terminal nodes: 6744
# 12,126
```

It is difficult to gain great insight from a visualization of a 12,126 nodes decision tree, where the rules can't be seen with out to much zooming and moving. Surprisingly the visualization algorithms from the libraries that were used weren't optimized for large trees and it was really slow to plot the whole tree, probably this functionality is intended as a detailed big decision tree visualization is difficult to read and interpret, and other methods should be used to interpret a huge decision tree. For instance the final tree was saved into a text form for future parsing.

A zooming of some parts of the tree was made just for illustrative purposes.

```r
plot(party_fit[22], gp = gpar(fontsize = 12))
```

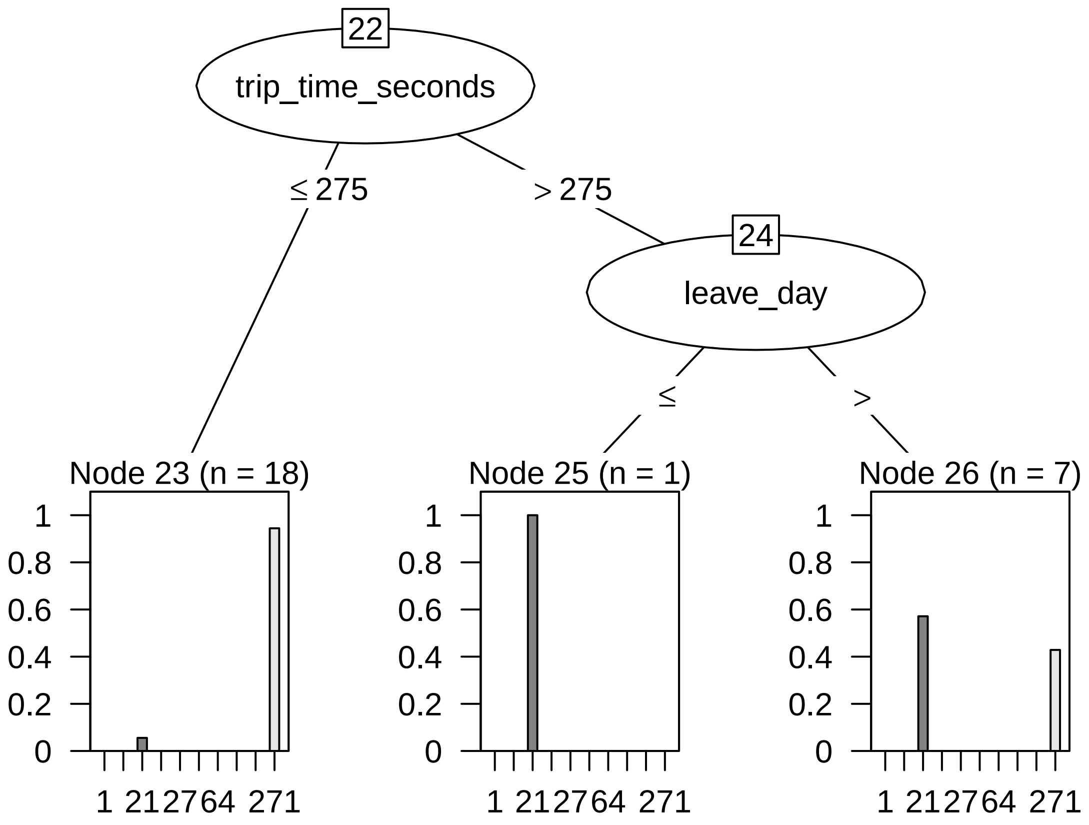

Figure 4.- Subtree at node 22.

```r
plot(party_fit[12], type = 'simple', gp = gpar(fontsize = 6))
```

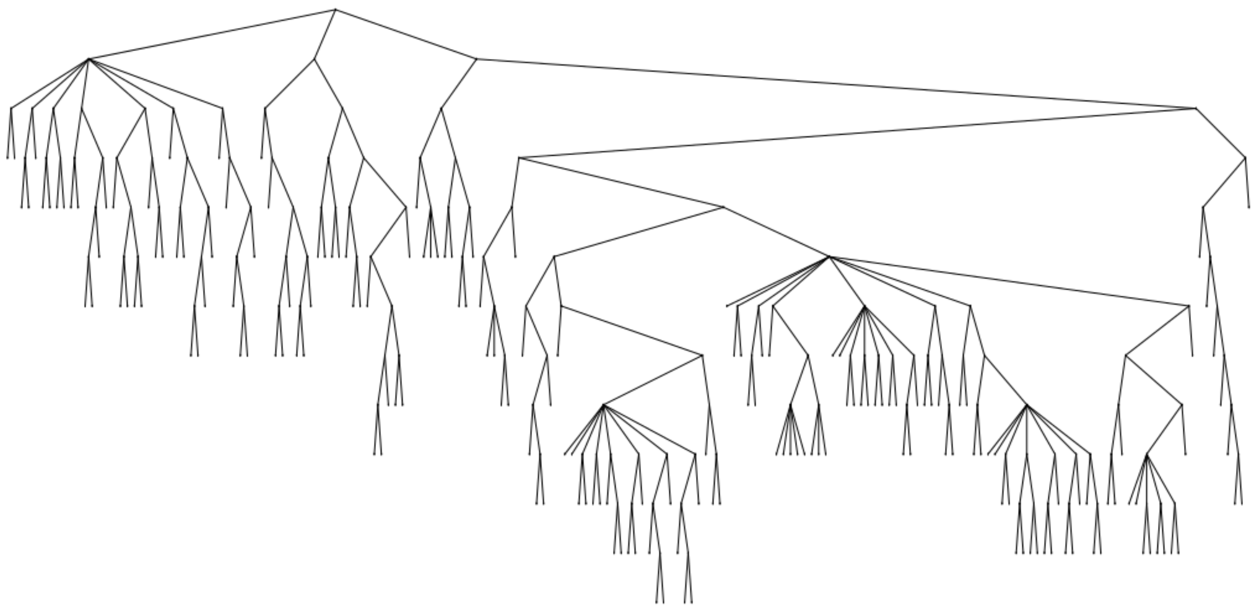

Figure 2: Topology of the tree at node 223.

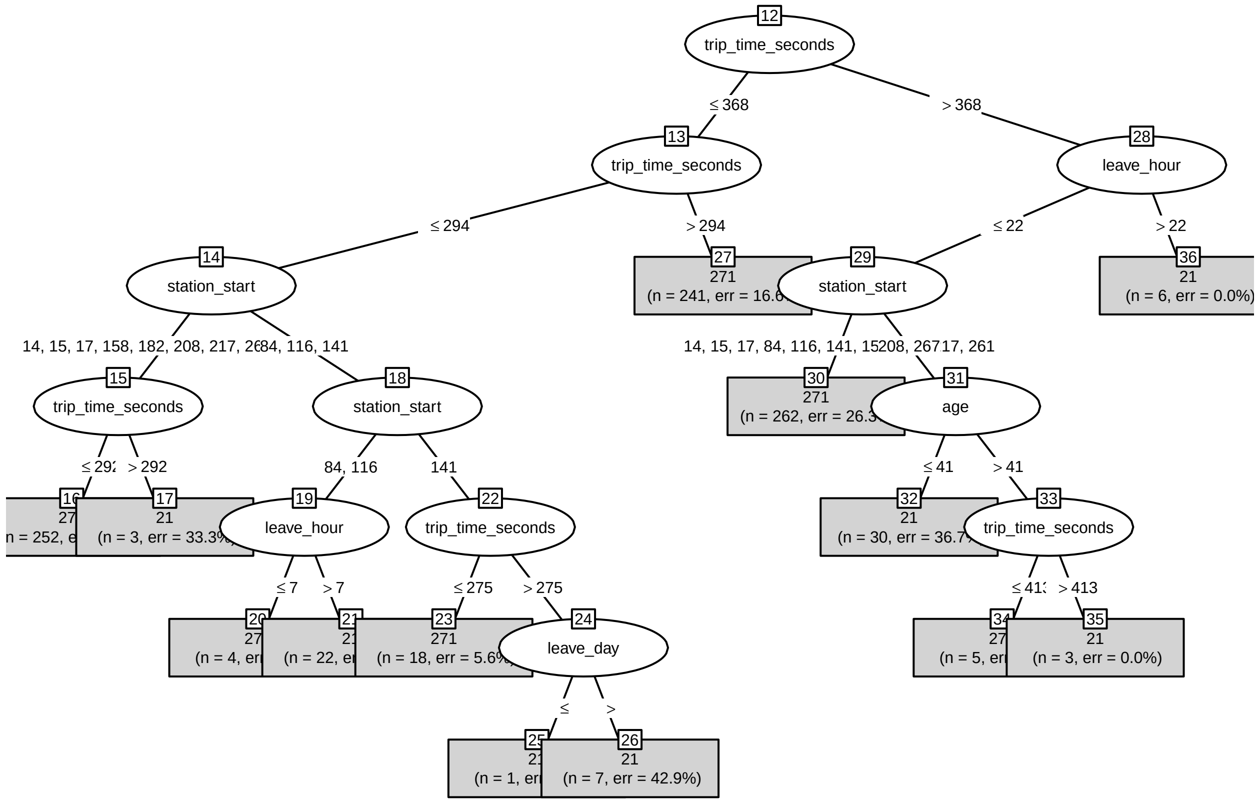

Figure 5.- Subtree at node 12.

```
# plot(party_fit[223], type = 'simple', gp = gpar(fontsize = 0))
```

Conclusions

The accuracy that was obtained is better than random but could be further improved, mainly reducing overfitting as the model seems to have quite a bit.

Visualization of big decision trees is difficult, further work could be done in that regard.

Big decision trees lost a little bit of the advantage of being relatively easy to interpret, but this result is expected as any sufficient large model is difficult to interpret.