Lab-06

Decision Trees

First we need to install the relevant packages

```
#install.packages("readr")
#install.packages("dplyr")
```

Now we can load the data.

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.3.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.3.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
titanic3 <- "https://goo.gl/At238b" %>%
  read_csv %>% # read in the data
  select(survived, embarked, sex, sibsp, parch, fare) %>%
  mutate(embarked = factor(embarked), sex = factor(sex))
```

```
## `curl` package not installed, falling back to using `url()`
```

```
## Parsed with column specification:
## cols(
##   pclass = col_character(),
##   survived = col_integer(),
##   name = col_character(),
##   sex = col_character(),
##   age = col_double(),
##   sibsp = col_integer(),
##   parch = col_integer(),
##   ticket = col_character(),
##   fare = col_double(),
##   cabin = col_character(),
##   embarked = col_character(),
##   boat = col_character(),
##   body = col_integer(),
##   home.dest = col_character()
## )
```

```
## Warning: package 'bindrcpp' was built under R version 3.3.3
```

```
titanic3
```

```
## # A tibble: 1,309 x 6
##    survived embarked    sex sibsp parch     fare
##       <int>   <fctr> <fctr> <int> <int>    <dbl>
##  1        1        S female     0     0 211.3375
##  2        1        S   male     1     2 151.5500
##  3        0        S female     1     2 151.5500
##  4        0        S   male     1     2 151.5500
##  5        0        S female     1     2 151.5500
##  6        1        S   male     0     0  26.5500
##  7        1        S female     1     0  77.9583
##  8        0        S   male     0     0   0.0000
##  9        1        S female     2     0  51.4792
## 10        0        C   male     0     0  49.5042
## # ... with 1,299 more rows
```

Part 1: Feature transformation

We will need to convert the survived variable from a numerical to categorical, as this will be our target. We can do this by using the as.factor() command.

```
titanic3$survived = as.factor(titanic3$survived)
```

Part 2: Build tree

To construct a tree, first we will need to import the tree library.

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.3.2
```

Now we can build a tree by selecting our target variable and predictors.

```
dead = tree(titanic3$survived ~ titanic3$embarked +
              titanic3$sex + titanic3$sibsp + titanic3$parch +
              titanic3$fare, titanic3)
```

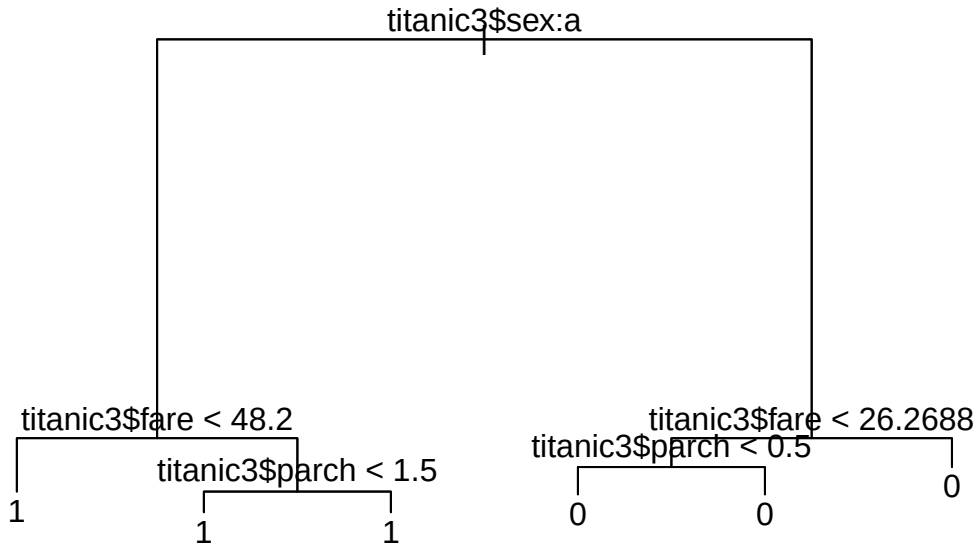
We can also view a summary of the tree, which will give us structural and statistical information.

```
summary(dead)
```

```
##
## Classification tree:
## tree(formula = titanic3$survived ~ titanic3$embarked + titanic3$sex +
##     titanic3$sibsp + titanic3$parch + titanic3$fare, data = titanic3)
## Variables actually used in tree construction:
## [1] "titanic3$sex"   "titanic3$fare"  "titanic3$parch"
## Number of terminal nodes:  6
## Residual mean deviance:  0.9582 = 1246 / 1300
## Misclassification error rate: 0.2205 = 288 / 1306
```

We can also visualise the tree by using the plot command.

```
plot(dead)
text(dead)
```



Part 3: Sampling

a: Split data

First we split the dataset into training and test sets.

```
# Define the sample size (50% of dataset)
sample_size <- floor(0.5 * nrow(titanic3))


set.seed(1)

# Split data
train_index <- sample(seq_len(nrow(titanic3)), size = sample_size)


# Apply split to datset
train <- titanic3[train_index, ]
test <- titanic3[-train_index, ]
```

b: Create a new tree using training set

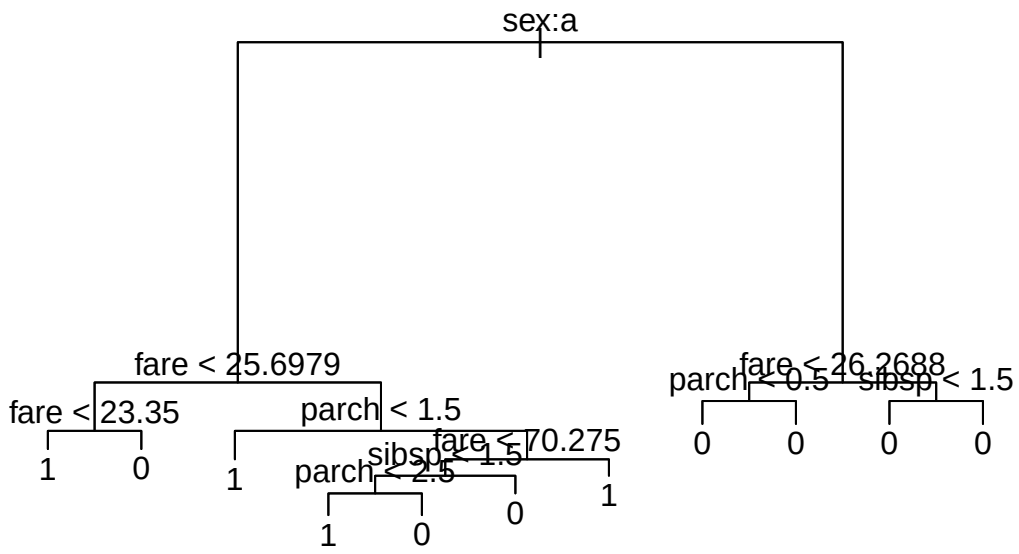
We can now build a new tree with the same target and predictor variables using the training set.

```
# dot denotes everything else
dead2 = tree(survived ~ ., train)

summary(dead2)
```

```
##
## Classification tree:
## tree(formula = survived ~ ., data = train)
## Variables actually used in tree construction:
## [1] "sex"   "fare"  "parch" "sibsp"
## Number of terminal nodes:  11
## Residual mean deviance:  0.9059 = 580.7 / 641
## Misclassification error rate: 0.2071 = 135 / 652
```

```
plot(dead2)
text(dead2)
```




```
dead2
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 652 880.900 0 ( 0.59356 0.40644 )
##    2) sex: female 237 268.200 1 ( 0.25316 0.74684 )
##      4) fare < 25.6979 124 165.500 1 ( 0.38710 0.61290 )
##        8) fare < 23.35 119 155.700 1 ( 0.36134 0.63866 ) *
##        9) fare > 23.35 5   0.000 0 ( 1.00000 0.00000 ) *
```

```
##      5) fare > 25.6979 113  76.500 1 ( 0.10619 0.89381 )
##       10) parch < 1.5 85  25.960 1 ( 0.03529 0.96471 ) *
##       11) parch > 1.5 28  35.160 1 ( 0.32143 0.67857 )
##         22) fare < 70.275 19  26.290 1 ( 0.47368 0.52632 )
##           44) sibsp < 1.5 14  16.750 1 ( 0.28571 0.71429 )
##             88) parch < 2.5 9   0.000 1 ( 0.00000 1.00000 ) *
##             89) parch > 2.5 5   5.004 0 ( 0.80000 0.20000 ) *
##           45) sibsp > 1.5 5   0.000 0 ( 1.00000 0.00000 ) *
##         23) fare > 70.275 9   0.000 1 ( 0.00000 1.00000 ) *
##    3) sex: male 415 428.800 0 ( 0.78795 0.21205 )
##      6) fare < 26.2688 298 272.800 0 ( 0.82886 0.17114 )
##       12) parch < 0.5 271 223.300 0 ( 0.85609 0.14391 ) *
##       13) parch > 0.5 27  37.100 0 ( 0.55556 0.44444 ) *
##      7) fare > 26.2688 117 146.000 0 ( 0.68376 0.31624 )
##       14) sibsp < 1.5 102 133.600 0 ( 0.63725 0.36275 ) *
##       15) sibsp > 1.5 15   0.000 0 ( 1.00000 0.00000 ) *
```

c: Evaluate the new tree using the test data

We can evaluate the new tree using the test set. First we will make predictions using the model, and return classes instead of probabilities.

```
dead.survived <- predict(dead2, newdata = test, type="class")
```

We can compare the predicted classes against the actual ones by using a confusion matrix, to check the number of correct classifications.

```
actual <- test$survived

# Confusion Matrix
table(dead.survived, actual)
```

```
##              actual
## dead.survived   0   1
##             0 370  77
##             1  52 156
```

```
# Mean error
mean(dead.survived!= actual)
```

```
## [1] 0.1969466
```

Finally we can check the model accuracy and error rate.

```
#accuracy
mean(dead.survived == actual)
```

```
## [1] 0.8030534
```

```
# Mean error
mean(dead.survived != actual)
```

```
## [1] 0.1969466
```

Part 4: Evaluate the Tree Structure

a: Cross Validate the tree

We can run a cross validation on the tree to see how it can be optimised.

```
set.seed(1)

cv.dead = cv.tree(dead2, FUN = prune.misclass)

print(cv.dead)
```

```
## $size
## [1] 11  8  4  2  1
##
## $dev
## [1] 150 150 147 149 267
##
## $k
## [1]  -Inf   0.0   2.0   2.5 117.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

b: Analysis the results

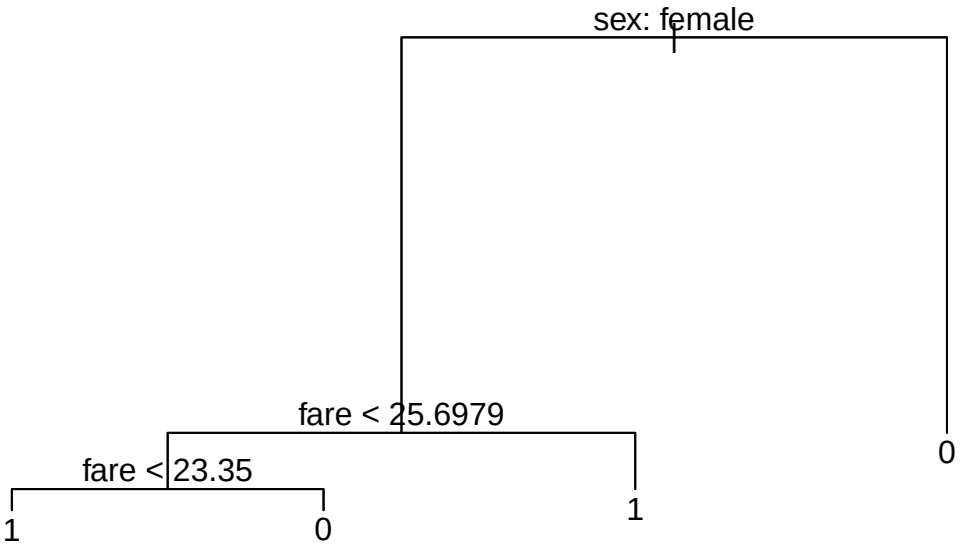
The results suggest that the optimum tree structure has 4 leaf nodes.

c: Prune the tree

We can now prune the tree using the prune function

```
# prune the tree
prune.dead = prune.misclass(dead2,best=4)

# plot the results
plot(prune.dead)
text(prune.dead, pretty=0)
```


sex: female
fare < 25.6979
fare < 23.35
1
0
1
0