Lab 7 Bagging and Random Forests

Problem Statement

We will continue exploring the titanic dataset and use the bagging and random forests to predict whether a person would survive or not, and compare the results with decision trees.

Dataset

The `titanic3` dataset we used last time has a predictor `pclass` (passenger class) that has value only 1. We will update the dataset with the one on moodle.

First you need to download `titanic3.csv` to a local directory. Then use the following code to obtain the dataset. Note that you need to install the packages `dplyr` to read in the dataset and process the dataset.

```r
library(dplyr)
titanic3 <- read.csv("yourLocalDirectory\\titanic3.csv")
titanic3 <- select(titanic3,-name, -ticket, -boat, -body, -home.dest, -cabin) %>%
  mutate(embarked = factor(embarked),
         sex = factor(sex),
         pclass=factor(pclass))
summary(titanic3)
```

You need to replace \ with \\ in yourLocalDirectory. For instance, your local directory is

C:\myDocument\RPrograms\Session9

Then

```r
titanic3 <- read.csv("C:\\myDocument\\RPrograms\\Session9\\titanic3.csv", header=FALSE)
```

Each row in the data is a passenger. Columns are features:

- `survived`: 0 if died, 1 if survived
- `embarked`: Port of Embarkation (Cherbourg, Queenstown, Southampton)
- `sex`: Gender
- `sibsp`: Number of Siblings/Spouses Aboard
- `parch`: Number of Parents/Children Aboard
- `fare`: Fare Payed
- `pclass`: Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
- `age`: Age

Questions

1) `survived` is a numeric value. We need to first transform it to a categorical value and saved it as a new variable survived01. Use `titanic3$survived01 = as.factor(titanic3$survived)` to do so and check that this variable has been included in the dataset.

2) Install the package of `randomForest` and include this package into your code. In order to call the `randomForest()` function, all the missing value rows need to be dealt with. The simplest way is to remove those rows. Use `titanic3 <- na.omit(titanic3)` to do that.

3) Use a seed to set half of the dataset to be training dataset and the other half to be test dataset.

4) Use the training dataset to build a bagged model for

- y: `survived`
- `x`: all the features other than `survived` and `survived01`.

Compute the mean error rate on the test dataset.

Remark: You might get a warning message, saying that

In randomForest.default(m, y, ...) : The response has five or fewer unique values. Are you sure you want to do regression?

Ignore the message for now. It's doable and you will get a bagged model anyway.

5) Using the same training and test dataset, build a bagged model for

- y: `survived01`
- x: all the features other than `survived` and `survived01`

a) Find out on how many trees your model is built and the OOB error

b) Compute the mean error rate on the training dataset.

6) Plot the variable importance plot for the two bagged models you built in 4) and 5) and comment whether the importance coincides.

7) Plot a graph that shows the test error rate of a single tree (red dashed line), the mean test error rates for majority vote (black curve) and the test error rates for averaging the probabilities (blue curve), both in relation to the number of trees. Add a legend if you can.

8) Plot a graph that shows the best value of mtry for the random forest model

- y: `survived01`
- x: all the features other than `survived` and `survived01`
- `mtry`: range from 1 to 7

9) Play with mtry and ntree, plot a graph that shows test error rate vs ntree for different mtry, and find the best/reasonably good combination of mtry and ntree from the plot. Add a legend if you can.