Lab 7 - Bagging and Random Forests

Question 1

`survived` is a numeric value. We need to first transform it to a categorical value and save 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.

```
library(dplyr)

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
t3 = read.csv("titanic3")
t3 = select(t3, -name, -ticket, -boat, -body, -home.dest, -cabin)
t3$survived01 = as.factor(t3$survived)
head(t3)
  pclass survived    sex   age sibsp parch     fare embarked survived01
1    1st        1 female 29.00     0     0 211.3375        S          1
2    1st        1   male  0.92     1     2 151.5500        S          1
3    1st        0 female  2.00     1     2 151.5500        S          0
4    1st        0   male 30.00     1     2 151.5500        S          0
5    1st        0 female 25.00     1     2 151.5500        S          0
6    1st        1   male 48.00     0     0  26.5500        S          1
```

Question 2

Install the package `randomForest` and include this in 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.

```
library(randomForest)
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'
The following object is masked from 'package:dplyr':

    combine
t3 = na.omit(t3)
```

Question 3

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

```
set.seed(8)
train = sample(nrow(t3), nrow(t3)/2)
test = t3[-train, ]
x_test = test[, -c(2, 9)]
survived.test = t3$survived[-train]
survived01.test = t3$survived01[-train]
```

Question 4

Use the training dataset to build a bagged model for: y: `survived` x: all the features other than

survived and survived01 Compare the mean error rate on the test dataset.

```
set.seed(6)
t3.bag.reg = randomForest(survived ~ .-survived01, data=t3, subset=train, mtry=7, importance=TRUE)
Warning in randomForest.default(m, y, ...): The response has five or fewer
unique values. Are you sure you want to do regression?
t3.bag.reg

Call:
 randomForest(formula = survived ~ . - survived01, data = t3,      mtry = 7, importance = TRUE, subset =
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 7

          Mean of squared residuals: 0.1658243
                    % Var explained: 32.49
t3.pred.reg = predict(t3.bag.reg, newdata=test)
t3.pred.cls = ifelse(t3.pred.reg <= 0.5, "0", "1")
mean(t3.pred.cls != survived.test)
[1] 0.2183908
```

Question 5

Using the same training and test datasets, build a bagged model for: y: survived01 x: all the features other than survived and survived01 a. Find out how many trees your model has built and the OOB error. b. Compute the mean error rate on the training dataset.

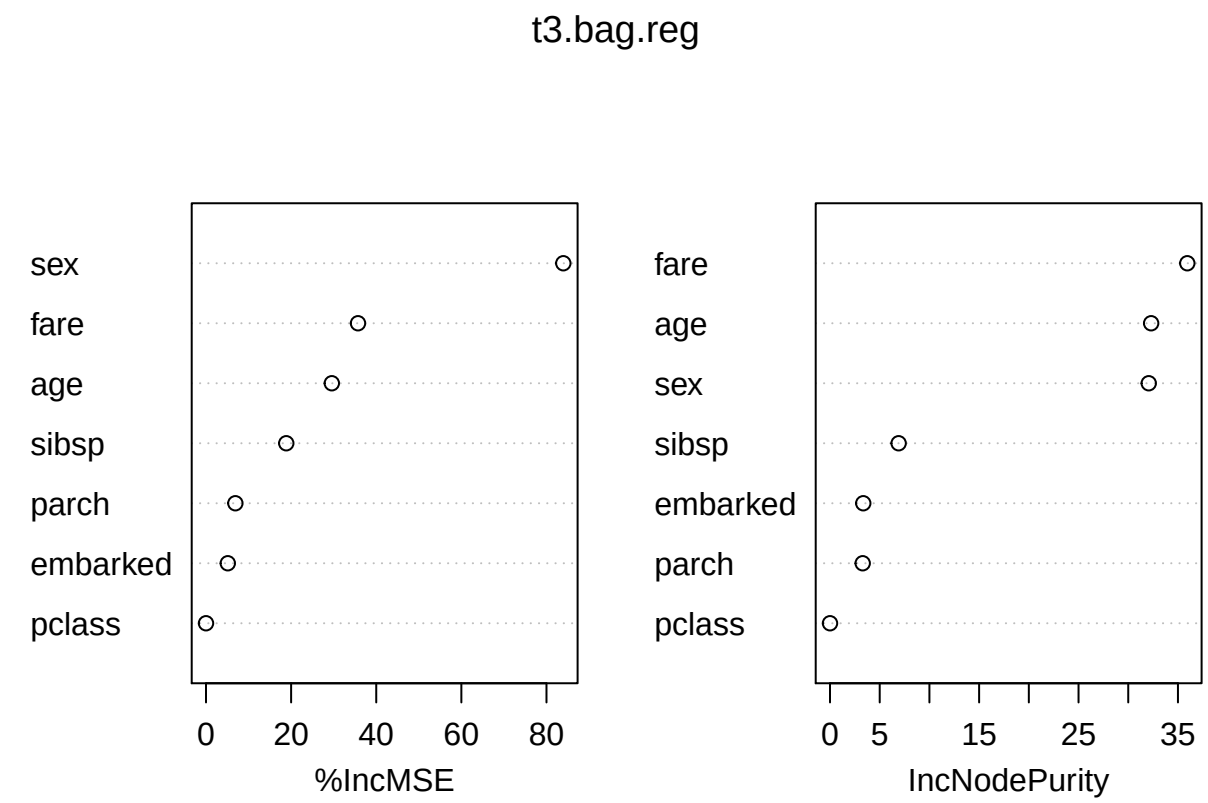
```
t3.bag.cls = randomForest(survived01 ~ .-survived, data=t3, subset=train, mtry=7, importance=TRUE)
t3.bag.cls

Call:
 randomForest(formula = survived01 ~ . - survived, data = t3,      mtry = 7, importance = TRUE, subset =
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 23.8%
Confusion matrix:
    0   1 class.error
0 237  58   0.1966102
1  66 160   0.2920354
(t3.bag.cls$confusion[2,1] + t3.bag.cls$confusion[1,2]) / t3.bag.cls$ntree
[1] 0.248
```

Question 6

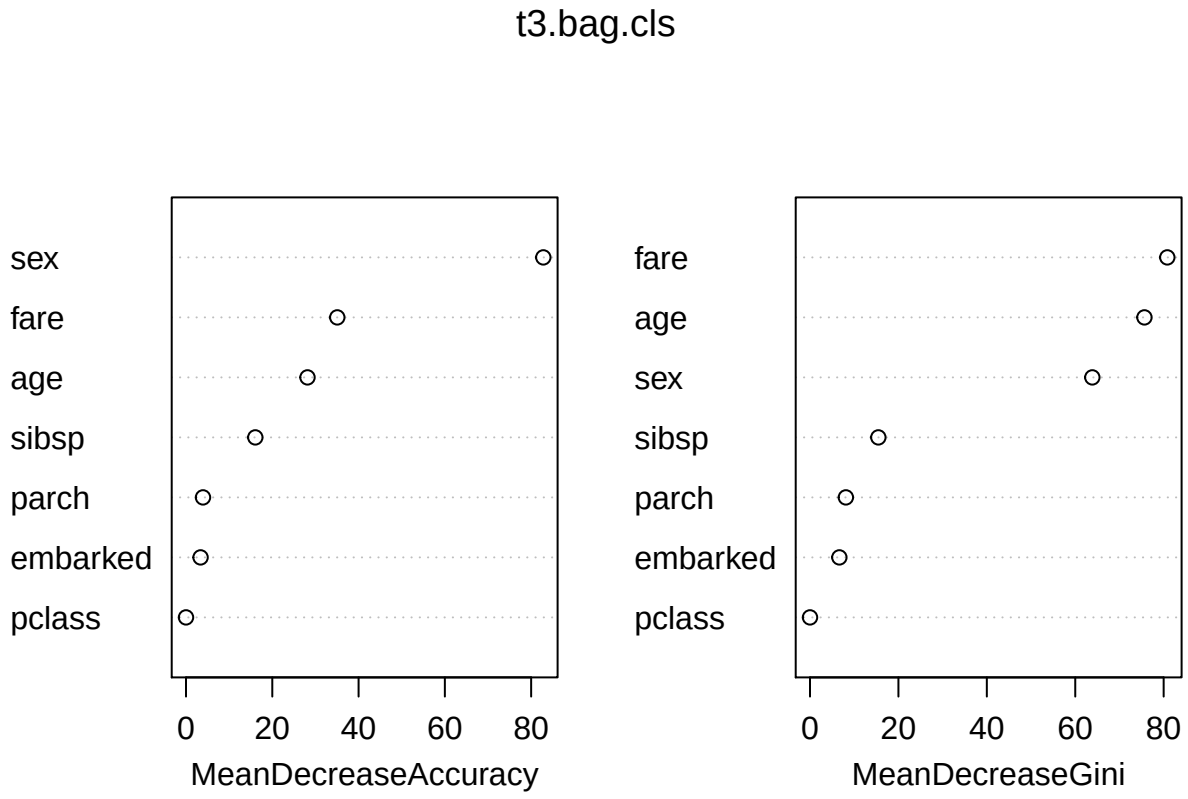
Plot the variable importance plot for the two bagged models you built in 4) and 5) and comment whether the importance coincides.

```
importance(t3.bag.reg)
           %IncMSE IncNodePurity
pclass    0.000000      0.000000
sex      83.953032     32.062009
age      29.557426     32.302545
sibsp    18.857017      6.903732
parch     6.899582      3.288090
fare     35.714588     35.940335
```

```
embarked  5.119452      3.344454
varImpPlot(t3.bag.reg)
```




```
importance(t3.bag.cls)
                  0         1 MeanDecreaseAccuracy MeanDecreaseGini
pclass    0.0000000  0.000000             0.000000         0.000000
sex      48.6199336 76.660492            82.819768        63.864605
age      21.2105774 20.004326            28.162492        75.647774
sibsp    14.3420475  6.597501            16.061276        15.460113
parch     0.2886203  4.521039             3.935445         8.114492
fare     13.4263084 35.640865            35.062100        80.825222
embarked  2.8979145  1.802079             3.374594         6.642298
varImpPlot(t3.bag.cls)
```

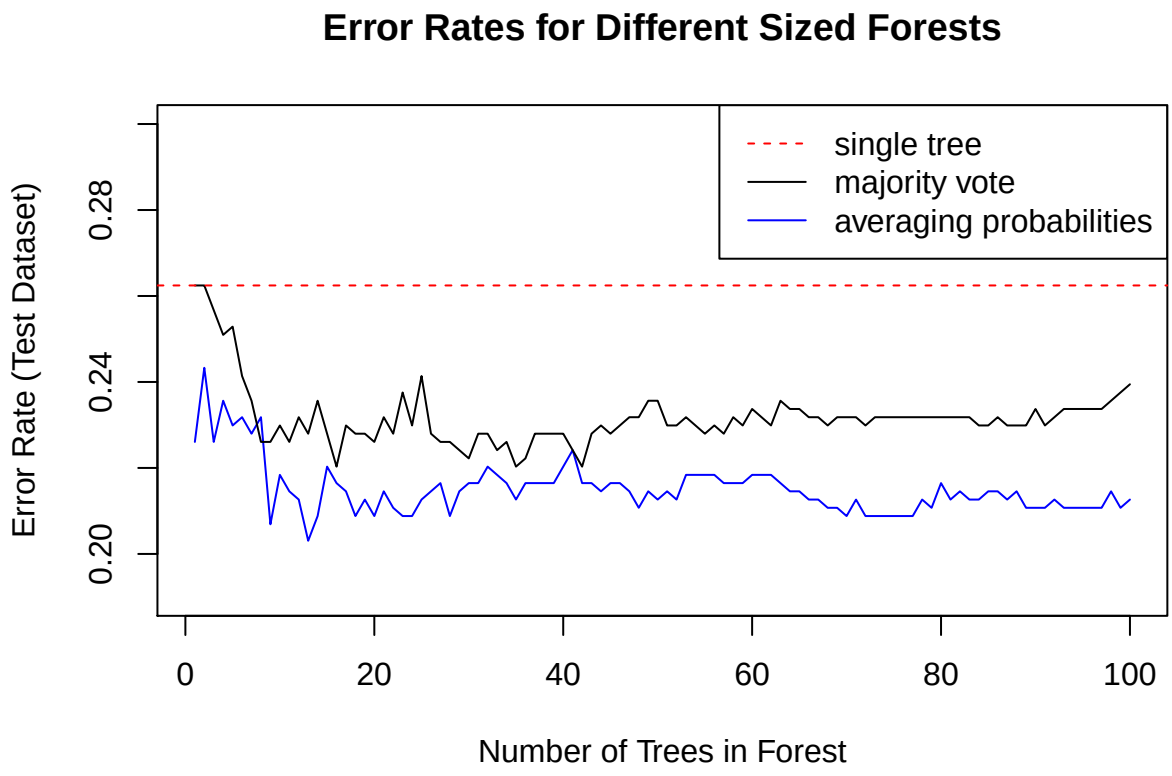


Question 7

Plot a graph that shows the test error rate of a single tree (red dashed line), the mean test error rate 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.

```
set.seed(1)
index.train = sample(nrow(t3), nrow(t3)/2)
dataset.training = t3[train, ]
dataset.test = t3[-train, ]
y.actual = dataset.test[, 9]
number.of.iterations = 100
test.error.rate.mv = rep(0, number.of.iterations)
test.error.rate.av = rep(0, number.of.iterations)
for (i in 1:number.of.iterations) {
  # Majority vote:
  set.seed(6)
  rf.fit.mv = randomForest(survived01 ~ .-survived, data=dataset.training, mtry=7, ntree=i, importance="
  y.predict.mv = predict(rf.fit.mv, newdata=dataset.test)
  confuse.mv = table(y.predict.mv, y.actual)
  test.error.rate.mv[i] = (confuse.mv[1,2] + confuse.mv[2,1]) / nrow(dataset.test)
  # Averaging Probabilities:
  set.seed(6)
  rf.fit.av = randomForest(survived ~ .-survived01, data=dataset.training, mtry=7, ntree=i, imporance="TR
  y.predict.av = ifelse(predict(rf.fit.av, newdata=dataset.test) <= 0.5, "0", "1")
  confuse.av = table(y.predict.av, y.actual)
  test.error.rate.av[i] = (confuse.av[1,2] + confuse.av[2,1]) / nrow(dataset.test)
```

```
}
plot(test.error.rate.av,
  col="blue",
  type="l",
  xlab="Number of Trees in Forest",
  ylab="Error Rate (Test Dataset)",
  main="Error Rates for Different Sized Forests",
  ylim=c(0.19,0.30)
)
lines(test.error.rate.mv,
  col="black",
  type="l"
)
abline(h=test.error.rate.mv[1], col="red", lty=2)
legend("topright",
  c("single tree", "majority vote", "averaging probabilities"),
  col=c("red", "black", "blue"),
  lty=c(2, 1, 1)
)
```

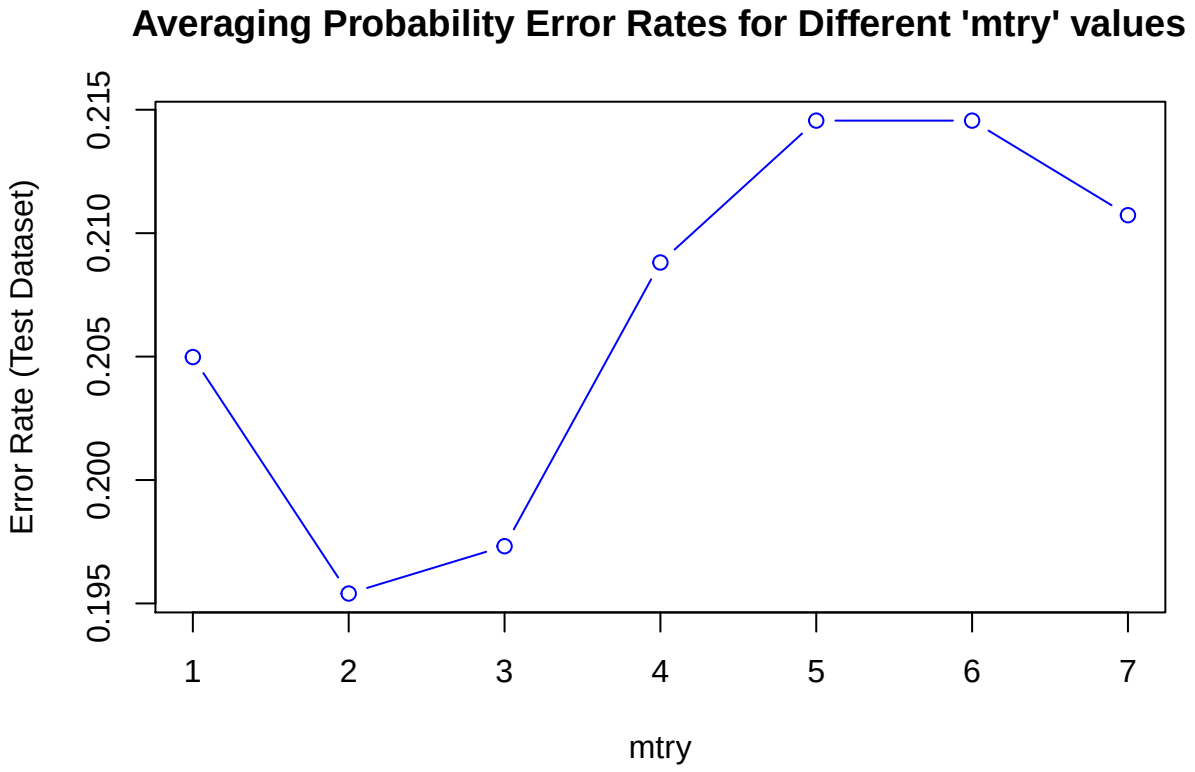


Question 8

Plot a graph tht 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

```
m = 7
test.error.rate = rep(0, m)
```

```r
for (i in 1:m) {
  set.seed(6)
  rf.fit = randomForest(survived ~ .-survived01, data=dataset.training, mtry=i, imporance=TRUE)
  y.predict = ifelse(predict(rf.fit, newdata=dataset.test) <= 0.5, "0", "1")
  confuse = table(y.predict, y.actual)
  test.error.rate[i] = (confuse[1,2] + confuse[2,1]) / nrow(dataset.test)
}
plot(test.error.rate,
  col="blue",
  type="b",
  xlab="mtry",
  ylab="Error Rate (Test Dataset)",
  main="Averaging Probability Error Rates for Different 'mtry' values"
)
```



Question 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.