Part 3 Ensemble

```
library(mltools)
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(rpart)
library(xgboost)
library(adabag)
```

```
## Loading required package: caret
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
## Loading required package: lattice
```

```
## Loading required package: foreach
```

```
## Loading required package: doParallel
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
adultsData <- read.csv("adult.data", header=FALSE)
names(adultsData) <- c("age", "workclass", "fnlwgt", "education", "education_num", "marital_s
tatus", "occupation", "relationship", "race", "sex", "capital_gain", "capital_loss", "hours_p
er_week", "native_country", "income")

sapply(adultsData, function(x) sum(is.na(x)))
```

```
##            age      workclass         fnlwgt      education  education_num
##              0              0              0              0              0
## marital_status     occupation   relationship           race            sex
##              0              0              0              0              0
##   capital_gain   capital_loss hours_per_week native_country         income
##              0              0              0              0              0
```

```
adultsData <- adultsData[(complete.cases(adultsData)),]
sum(is.na(adultsData))
```

```
## [1] 0
```

```
adultsData$workclass <- factor(adultsData$workclass)
adultsData$education <- factor(adultsData$education)
adultsData$marital_status <- factor(adultsData$marital_status)
adultsData$occupation <- factor(adultsData$occupation)
adultsData$relationship <- factor(adultsData$relationship)
adultsData$race <- factor(adultsData$race)
adultsData$sex <- factor(adultsData$sex)
adultsData$native_country <- factor(adultsData$native_country)
adultsData$income <- factor(adultsData$income)

set.seed(12345)

sample <- sample(c(TRUE,FALSE), nrow(adultsData), replace=TRUE, prob=c(0.8,0.2))
train <- adultsData[sample, ]
test <- adultsData[!sample, ]

print("Correlation between age and income: ")
```

```
## [1] "Correlation between age and income: "
```

```
cor(train$age, as.numeric(train$income))
```

```
## [1] 0.2355458
```

```
print("Correlation between capital gain and income: ")
```

```
## [1] "Correlation between capital gain and income: "
```

```
cor(train$capital_gain, as.numeric(train$income))
```

```
## [1] 0.2236056
```

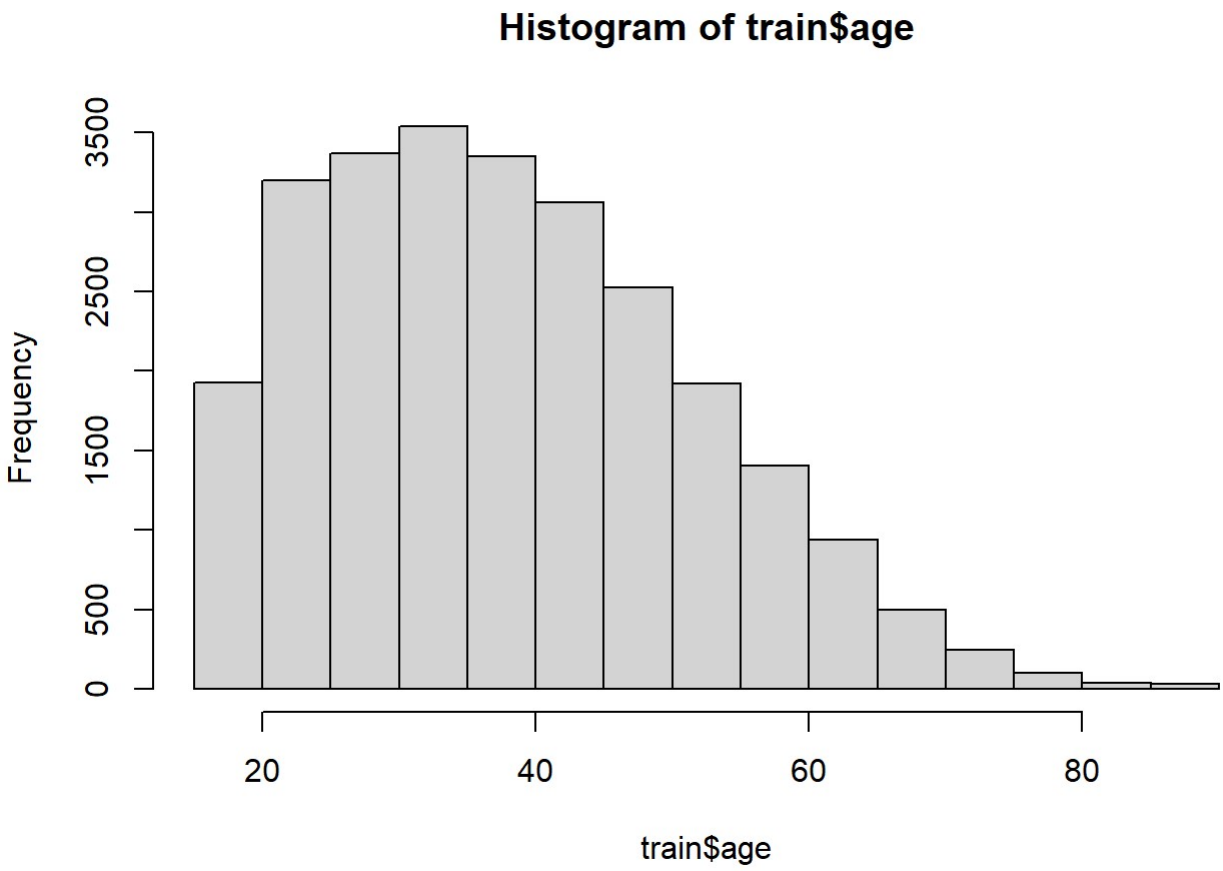
```
print("Correlation between capital loss and income: ")
```

```
## [1] "Correlation between capital loss and income: "
```

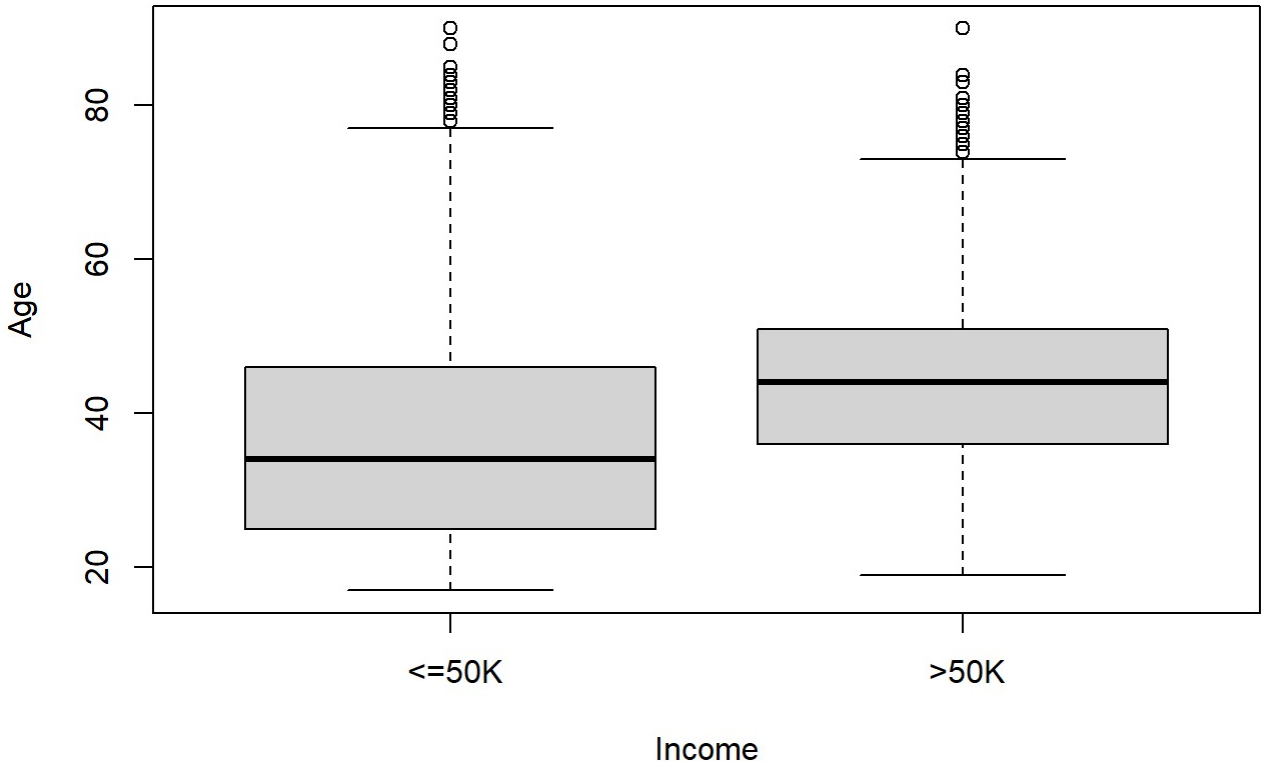
```
cor(train$capital_loss, as.numeric(train$income))
```

```
## [1] 0.1572704
```

```
hist(train$age)
```



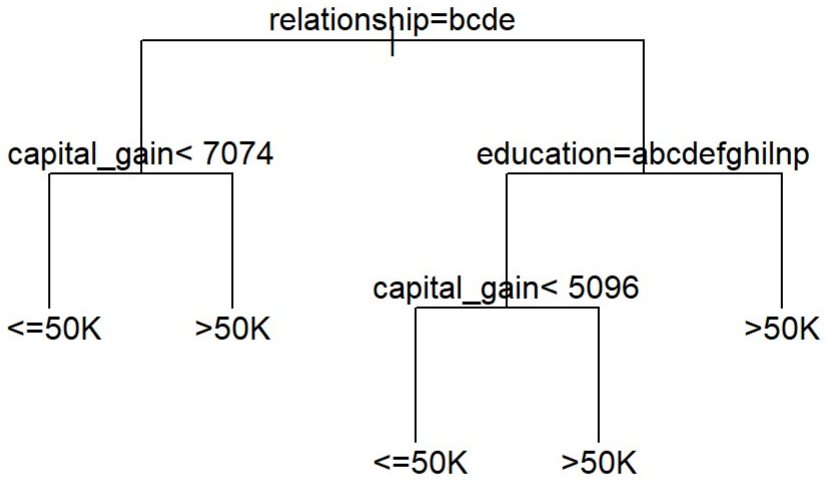

```
boxplot(train$age~train$income, xlab="Income", ylab="Age")
```




```
tree1 <- rpart(income~., data=train, method="class")
tree1
```

```
## n= 26157
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 26157 6319  <=50K (0.75842031 0.24157969)
##    2) relationship= Not-in-family, Other-relative, Own-child, Unmarried 14269  936  <=50K
(0.93440325 0.06559675)
##      4) capital_gain< 7073.5 14022  700  <=50K (0.95007845 0.04992155) *
##      5) capital_gain>=7073.5 247   11  >50K (0.04453441 0.95546559) *
##    3) relationship= Husband, Wife 11888 5383  <=50K (0.54719044 0.45280956)
##      6) education= 10th, 11th, 12th, 1st-4th, 5th-6th, 7th-8th, 9th, Assoc-acdm, Assoc-vo
c, HS-grad, Preschool, Some-college 8281 2756  <=50K (0.66718995 0.33281005)
##       12) capital_gain< 5095.5 7862 2346  <=50K (0.70160265 0.29839735) *
##       13) capital_gain>=5095.5 419    9  >50K (0.02147971 0.97852029) *
##      7) education= Bachelors, Doctorate, Masters, Prof-school 3607  980  >50K (0.27169393
0.72830607) *
```

```
plot(tree1, uniform=TRUE, margin = 0.2)
text(tree1)
```




```
pred1 <- predict(tree1, newdata = test, type="class")
treeTab <- table(pred1, test$income)
treeAcc <- (sum(diag(treeTab))/sum(treeTab))

treeTab
```

```
##
## pred1     <=50K  >50K
##    <=50K   4635   770
##    >50K     247   752
```

```
print("Rpart decision tree: ")
```

```
## [1] "Rpart decision tree: "
```

```
print("Accuracy: ")
```

```
## [1] "Accuracy: "
```

```
treeAcc
```

```
## [1] 0.841193
```

```
print("Error rate: ")
```

```
## [1] "Error rate: "
```

```
(1 - treeAcc)
```

```
## [1] 0.158807
```

```
print("Sensititvity: ")
```

```
## [1] "Sensititvity: "
```

```
(treeTab[1,1] /(treeTab[1,1] + treeTab[2,1]))
```

```
## [1] 0.949406
```

```
print("Specificity: ")
```

```
## [1] "Specificity: "
```

```
(treeTab[2,2] /(treeTab[2,2] + treeTab[1,2]))
```

```
## [1] 0.4940867
```

```
treeMcc <- mcc(pred1, test$income)
print(paste("mcc = ", treeMcc))
```

```
## [1] "mcc =  0.520250019410089"
```

```
rf <- randomForest(income~., data=train, importance = TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = income ~ ., data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.54%
## Confusion matrix:
##         <=50K  >50K class.error
##  <=50K  18523  1315  0.06628692
##  >50K    2226  4093  0.35227093
```

```
pred2 <- predict(rf, newdata = test, type = "response")
acc_rf <- mean(pred2==test$income)
mcc_rf <- mcc(factor(pred2), test$income)
print(paste("accuracy=", acc_rf))
```

```
## [1] "accuracy= 0.865865084322299"
```

```
print(paste("mcc=", mcc_rf))
```

```
## [1] "mcc= 0.610759564002877"
```

The Random Forest is slightly more accurate than the decision tree and has a better MCC value.

```
train_label <- ifelse(train$income==" >50K", 1, 0)
train_matrix <- data.matrix(train[, -15])
model <- xgboost(data=train_matrix, label=train_label, nrounds=100, objective='binary:logisti
c')
```

```
## [1]  train-logloss:0.541134
## [2]  train-logloss:0.457044
## [3]  train-logloss:0.404679
## [4]  train-logloss:0.371644
## [5]  train-logloss:0.348491
## [6]  train-logloss:0.331757
## [7]  train-logloss:0.319486
## [8]  train-logloss:0.310882
## [9]  train-logloss:0.302309
## [10] train-logloss:0.296444
## [11] train-logloss:0.292142
## [12] train-logloss:0.289140
## [13] train-logloss:0.285738
## [14] train-logloss:0.283267
## [15] train-logloss:0.280138
## [16] train-logloss:0.277118
## [17] train-logloss:0.275094
## [18] train-logloss:0.273364
## [19] train-logloss:0.271661
## [20] train-logloss:0.269047
## [21] train-logloss:0.267285
## [22] train-logloss:0.264685
## [23] train-logloss:0.262947
## [24] train-logloss:0.261209
## [25] train-logloss:0.259870
## [26] train-logloss:0.258803
## [27] train-logloss:0.257151
## [28] train-logloss:0.256412
## [29] train-logloss:0.255819
## [30] train-logloss:0.254330
## [31] train-logloss:0.253546
## [32] train-logloss:0.252033
## [33] train-logloss:0.251498
## [34] train-logloss:0.250358
## [35] train-logloss:0.248463
## [36] train-logloss:0.247362
## [37] train-logloss:0.246050
## [38] train-logloss:0.245644
## [39] train-logloss:0.244664
## [40] train-logloss:0.244438
## [41] train-logloss:0.244023
## [42] train-logloss:0.243286
## [43] train-logloss:0.242738
## [44] train-logloss:0.241804
## [45] train-logloss:0.241374
## [46] train-logloss:0.240133
## [47] train-logloss:0.239598
## [48] train-logloss:0.238496
## [49] train-logloss:0.237009
## [50] train-logloss:0.236067
## [51] train-logloss:0.235457
```

```
## [52] train-logloss:0.235317
## [53] train-logloss:0.234871
## [54] train-logloss:0.233906
## [55] train-logloss:0.232540
## [56] train-logloss:0.231183
## [57] train-logloss:0.230380
## [58] train-logloss:0.229831
## [59] train-logloss:0.229726
## [60] train-logloss:0.229439
## [61] train-logloss:0.229088
## [62] train-logloss:0.228652
## [63] train-logloss:0.227844
## [64] train-logloss:0.226614
## [65] train-logloss:0.226237
## [66] train-logloss:0.225456
## [67] train-logloss:0.225061
## [68] train-logloss:0.224147
## [69] train-logloss:0.223532
## [70] train-logloss:0.222883
## [71] train-logloss:0.221828
## [72] train-logloss:0.221284
## [73] train-logloss:0.220770
## [74] train-logloss:0.220340
## [75] train-logloss:0.220178
## [76] train-logloss:0.220073
## [77] train-logloss:0.219680
## [78] train-logloss:0.218723
## [79] train-logloss:0.217897
## [80] train-logloss:0.216606
## [81] train-logloss:0.215998
## [82] train-logloss:0.215657
## [83] train-logloss:0.215093
## [84] train-logloss:0.214411
## [85] train-logloss:0.214057
## [86] train-logloss:0.213554
## [87] train-logloss:0.212657
## [88] train-logloss:0.212066
## [89] train-logloss:0.211392
## [90] train-logloss:0.211258
## [91] train-logloss:0.210596
## [92] train-logloss:0.210389
## [93] train-logloss:0.209764
## [94] train-logloss:0.209085
## [95] train-logloss:0.208661
## [96] train-logloss:0.207967
## [97] train-logloss:0.207588
## [98] train-logloss:0.207462
## [99] train-logloss:0.207371
## [100]    train-logloss:0.206344
```

```
test_label <- ifelse(test$income==" >50K", 1, 0)
test_matrix <- data.matrix(test[, -15])
probs <- predict(model, test_matrix)
pred <- ifelse(probs>0.5, 1, 0)

acc_xg <- mean(pred==test_label)
mcc_xg <- mcc(pred,test_label)
print(paste("accuracy =", acc_xg))
```

```
## [1] "accuracy = 0.868675827607745"
```

```
print(paste("mcc =", mcc_xg))
```

```
## [1] "mcc = 0.621901676924291"
```

XGBoost has better accuracy than both of the other algorithims and a better MCC value. Additionally, XGBoost ran much faster than the Random Forest and the Decision tree.

```
adab1 <- boosting(income~., data=train, boos=TRUE, mfinal=20, coeflearn='Breiman')
summary(adab1)
```

```
##            Length Class   Mode
## formula         3  formula call
## trees          20  -none-  list
## weights        20  -none-  numeric
## votes       52314  -none-  numeric
## prob        52314  -none-  numeric
## class       26157  -none-  character
## importance     14  -none-  numeric
## terms           3  terms   call
## call            6  -none-  call
```

```
pred4 <- predict(adab1, newdata=test, type="response")
acc_adab <- mean(pred4$class==test$income)
mcc_adab <- mcc(factor(pred4$class), test$income)

print(paste("accuracy =", acc_adab))
```

```
## [1] "accuracy = 0.856183635227982"
```

```
print(paste("mcc =", mcc_adab))
```

```
## [1] "mcc = 0.58447297430301"
```

The accuracy of the Boosting algorithm is not as good as the random forest or the XGBoost, but it is better than

the decision tree.