Chapter 8 in ISL: Tree-Based Methods

9 April 2019

Upcoming Course Schedule

HW 5: Due next Friday, April 12 by 11:59 PM.

Exam: On Friday, April 12, your final exam will be made available. You will have until Sunday, April 21 11:59 PM to finish your exam.

- Exam will be made available via a personal repo. You will get a message on Slack with a link; click that link to create the repo via GitHub Classroom.
- Let's hope GitHub classroom doesn't fail like last time -_-.

Project: There are two parts remaining for your project. You have presentations April 23 and/or 25. Project Checkpoint 3 Due Monday April 29 by 11:59 PM.

Presenations

- These will be delivered during the final week of classes, April 23 and/or 25.
- Each group should prepare a 15 minute (maximum) presentation emphasizing the data, question, methods, and conclusions of their project. You should prepare slides (this can be done easily in Rmarkdown) which shows appropriate graphical displays and aids the delivery.
- Aim it at a smart audience with statistical training to the level of multiple linear regression but not beyond. Any modeling techiques learned in this class should be summarized. For example, you should not just say "We did KNN" but also explain the basic idea of how it works, why it might be better than linear regression, etc.
- Points (25 pts) will be allocated for clear articulations of the data you used, of the question(s) of interest, notable aspects of the data (most important aspects of EDA), the approach you used to answer your research question(s), the reason you chose that approach, and the conclusions you were able to draw.
- How to allocate presentation days... All one day? Two on Tuesday and two on Thursday?

Project Check Point 3

- Due Monday April 29 by 11:59 PM.

Each group should submit one report. It must consist of the following sections. (page numbers are relative as your report should render in html)

1. **Introduction** (1-2 page) Write three to four paragraphs introducing the research problem and describing the specific research hypothesis. Cite any information sources in parentheses or foot- or end- notes.
2. **EDA** (about 5 pages) This section should incorporate some material from the Midterm report (with any changes suggested in my grading thereof) as well as any new analyses which seem desirable as you have made further progress. Only the most relevant analyses should be included by now. You *may* put the rest in an Appendix (but you probably shouldn't).
3. **Results and analysis** (about 5 pages) Address the specific question(s) of interest with methods of your choice (these can graphical, numerical, or statistical models). Be sure to justify the choices you make, and explain why your calculations lead to the results that you discuss. Models without justification, figures presenting irrelevant material, or large tables of uninterpreted numbers will be treated harshly.

4. **Discussions/results** (1-2 pages) What are your conclusions? Identify a few key findings, and discuss, with reference to the supporting evidence. Can you come up with explanations for the patterns you have found? Suggestions or recommendations for future work? How could your analysis be improved?

Total page length: **10 to 15 pages** + appendix as necessary. Be sure to describe your graphics carefully. Plots with no description are useless. ## PC 3 Rubric

Words (8 / 8) The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, divided into complete sentences logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.

Numbers (1 / 1) All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.

Pictures (7 / 7) Figures and tables are easy to read, with informative captions, axis labels and legends, and are placed near the relevant pieces of text.

Code (4 / 4) The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. The text of the report is free of intrusive blocks of code and useless output. If you use R Markdown, all calculations are actually done in the file as it knits, and only relevant results are shown and explained.

Exploratory Data Analysis (15 / 15) Your job is to clearly explain the data in such a way so that the audience feels comfortable and has a reasonable understanding of the data. Variables are examined individually and bivariately. Features/observations are discussed with appropriate figure or tables. The relevance of the EDA to the questions and potential models is clearly explained.

Results and analysis (30 / 30) The statistical summaries are clearly related to, or possibly derive from, the substantive questions of interest. Any assumptions are checked by means of appropriate diagnostic plots or formal tests. Limitations from un-fixable problems are clearly noted. The actual estimation of parameters, predictions, or other calculations are technically correct. All calculations based on estimates are clearly explained, and also technically correct.

Conclusions (10 / 10) The substantive questions are answered as precisely as the data and the model allow. The chain of reasoning from estimation results about models, or derived quantities, to substantive conclusions is both clear and convincing. If uncertainties in the data mean the answers to some questions must be imprecise, this too is reflected in the conclusions.

Extra credit (0 / 0) Up to twenty points may be awarded for reports which are unusually well-written, where the code is unusually elegant, where the analytical methods are unusually insightful, or where the analysis goes beyond the required set of analytical questions.

Grade: 75 out of 75 possible

Problems With CART (Classification And Regression Trees) Models

Some intial advantages of trees relate to their simple construction and interpretability:

Pros:

- Trees are very easy to explain (much easier than even linear regression).
- Some people believe that decision trees mirror human decision.
- Trees can easily be displayed graphically no matter the dimension of the data.
- Trees can easily handle qualitative predictors without the need to create dummy variables.

When we create trees, often the observations will be grouped into relatively (compared to the total sample) small groups. This leads to a couple of problems:

Cons:

- We may overfit to training observations. This leads to poor predictive accuracy (high test variability).
- Small changes in the data can cause large changes in to over all tree.

Remember, we group like observations into regions (tree leaves) of the predictor space: $R_1, \ldots, R_M$.

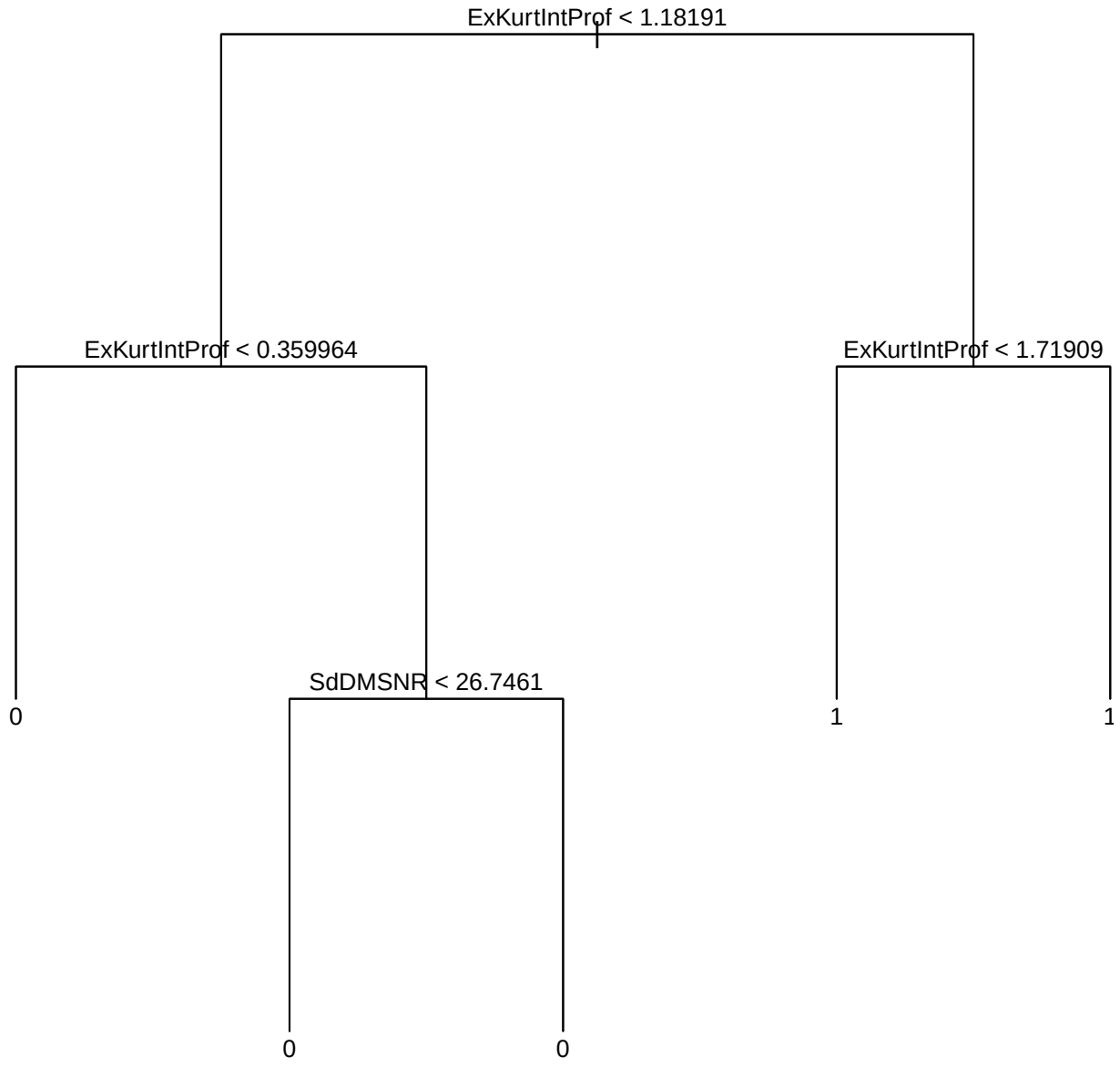
Our prediction is the mean of the response for the training observations within the mth box:

$$\widehat{y}_{R_m} = \frac{1}{n_{R_m}} \sum_{i:\underline{x} \in R_m} y_i$$

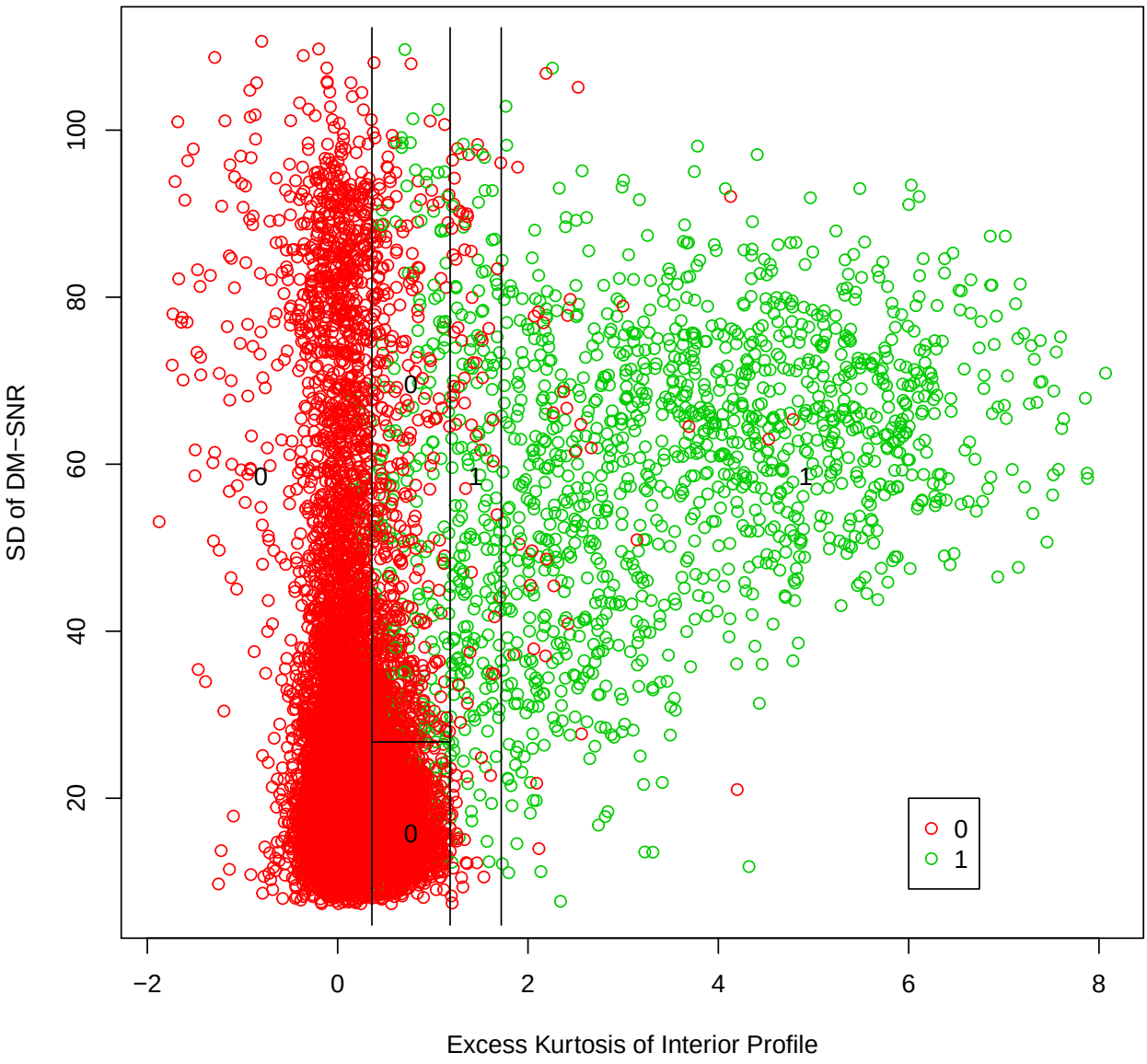
What is the variance of this sample mean, versus the variance of a sample mean using the entire data?

New Data Example: Predicting Pulsar Stars

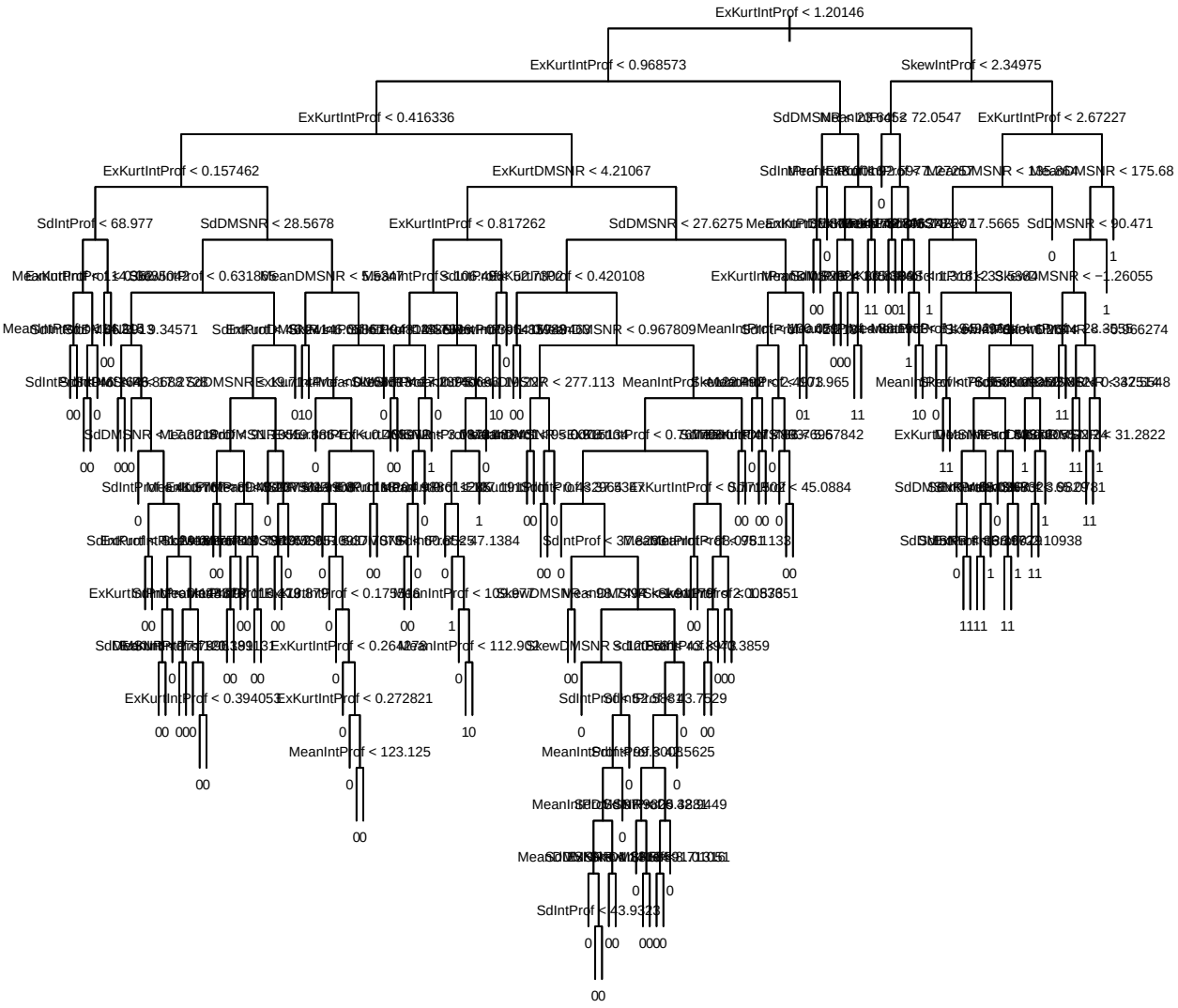
```
library(tree)
library(caret)
stars <- read.csv('data/stars.csv') # No NAs luckily enough.
stars$pulsar <- as.factor(stars$pulsar)

train.ind <- sample(1:nrow(stars), nrow(stars)/2)
test.ind <- (1:nrow(stars))[-train.ind]

tree <- tree(pulsar ~ ., data = stars, subset = train.ind)
tree.gini <- tree(pulsar ~., data = stars, subset = train.ind, split = "gini")

plot(tree, type = "uniform")
text(tree, cex=0.8)
```




```
plot(stars$ExKurtIntProf, stars$SdDMSNR, col = as.numeric(stars$pulsar)+1,
     pch = 1,
     xlab = "Excess Kurtosis of Interior Profile", ylab = "SD of DM-SNR")
partition.tree(tree, ordvars = c("ExKurtIntProf", "SdDMSNR"), col = "black", add = TRUE)
legend(6,20,legend=unique(stars$pulsar),
       col=unique(as.numeric(stars$pulsar)+1), pch = 1)
```




```
plot(tree.gini, type = "uniform") # Maybe Gini not so good...
text(tree.gini, cex=0.5)
```




```
test_data = stars[test.ind,]
test_data$class = predict(tree, test_data, type='class')
test_data$prob = predict(tree, test_data, type='vector')[,2]
test_data = mutate(test_data, correct = class==pulsar) # Make new "correct" variable

# Misclassification Rate
predclass = data.frame(truth = stars$pulsar[test.ind])
predclass$tree = predict(tree, newdata = stars[test.ind,], type='class') ## for later
```

Bagging: Introduction

Many methods (trees included) tend to be designed to have lower bias but high variance.

This means that if we split the training data into two parts at random and fit a regression or decision tree to each part, the results could be quite different.

In contrast, a low variance/high bias estimator would yield similar across many different training sets.

Bagging, also known as **bootstrap aggregation**, is a general purpose procedure for reducing variance. We'll use it specifically in the context of trees, but it can be applied much more broadly.

Suppose we have n uncorrelated observations $Z_1, \ldots, Z_n$, each with variance σ^2.

What is the variance of

$$\overline{Z} = \frac{1}{n}\sum_{i=1}^{n} Z_i?$$

More generally, if we have B separate (uncorrelated) training sets, $1, \ldots, B$, we can form B separate model fits, $\widehat{f}^1(x), \ldots, \widehat{f}^B(x)$, and then average them:

$$\widehat{f}_B(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}^b(x)$$

Bagging: The Bootstrap Part

Of course, this isn't practical as having access to many training sets is unlikely. We therefore turn to the bootstrap to simulate having many training sets.

Suppose we have data $Z_1, \ldots, Z_n$ and we want to get an idea of the sampling distribution of some statistic $f(Z_1, \ldots, Z_n)$. Then we do the following

1. Choose some large number of samples, B.
2. For each $b = 1, \ldots, B$, draw a new dataset from $Z_1, \ldots, Z_n$, call it $Z_1^*, \ldots, Z_n^*$. Remember, in bootstrapping, we sample from the orginal dataset with replacement!
3. Compute $\widehat{f}_b^* = \widehat{f}(Z_1^*, \ldots, Z_n^*)$.

Now, instead of having B separate training sets, we do B bootstrap samples. $\widehat{f}_1^*(x), \ldots, \widehat{f}_B^*(x)$, and then average them:

$$\widehat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}_b^*(x)$$

This process is known as Bagging (bootstrap aggregation).

Bagging trees

The procedure for trees is the following

1. Choose a large number B.
2. For each $b = 1, \ldots, B$, grow an unpruned tree on the b^{th} bootstrap draw from the data.
3. Average the predictions of all these trees together.

Each tree, since it is unpruned, will have high variance and low bias.

Therefore averaging many trees results in an estimator that has lower variance and still low bias.

Caveat: Be careful bootstrapping time series data.

Bagging the Pulsar Data

```
par(mfrow = c(3,2))
B = 6
set.seed(1)
for(i in 1:B){

  boot <- stars[sample(1:nrow(stars), replace = T),]
```

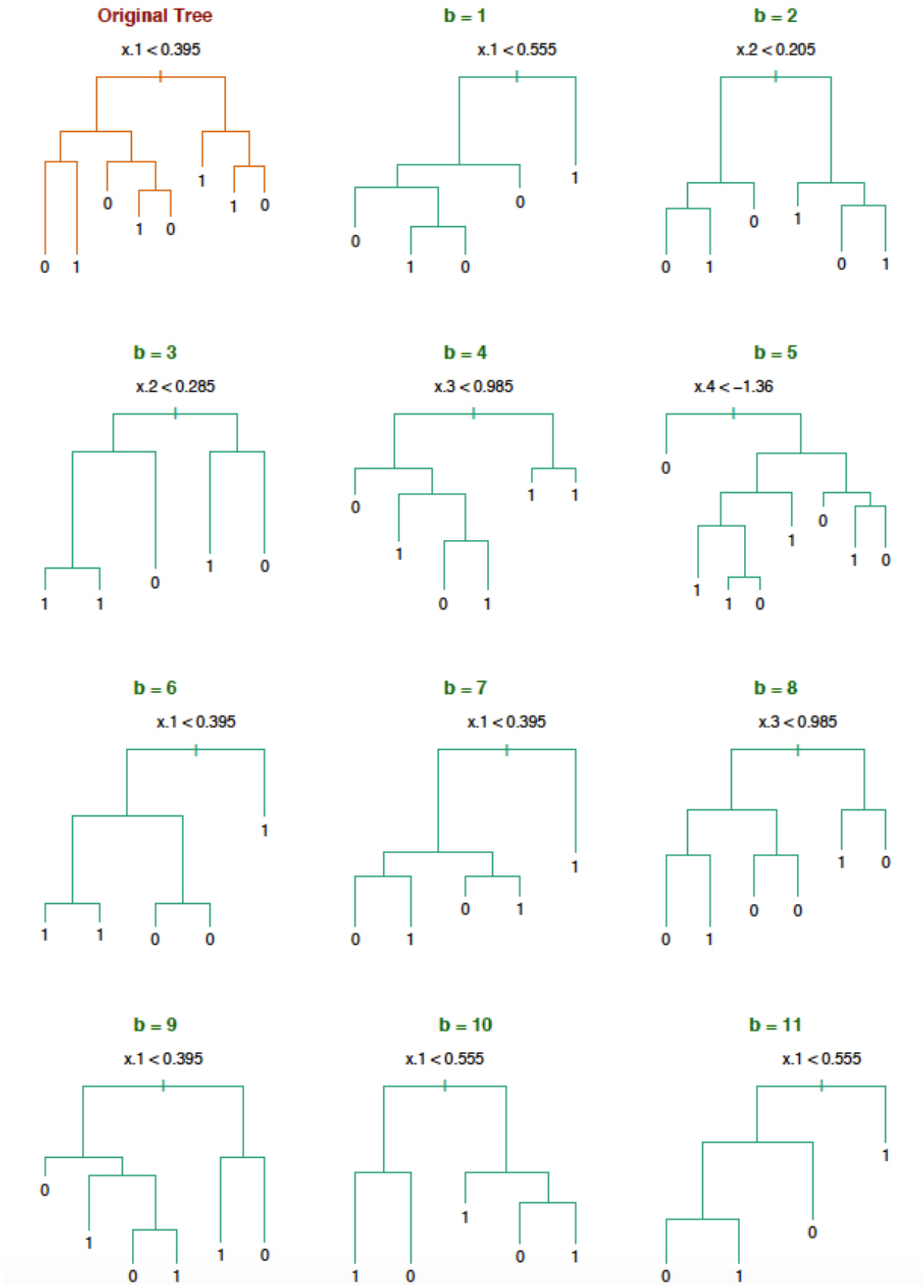
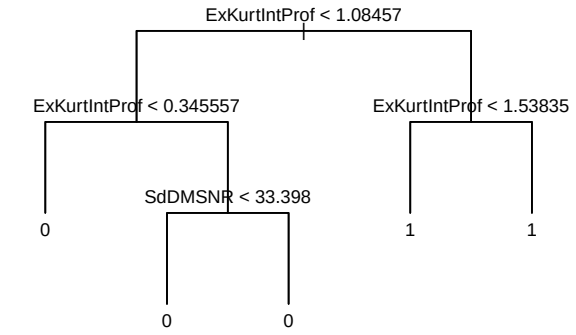
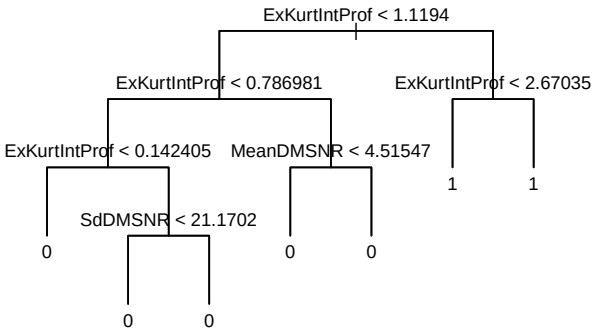
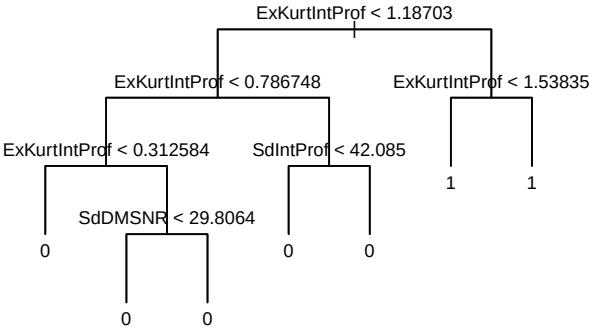


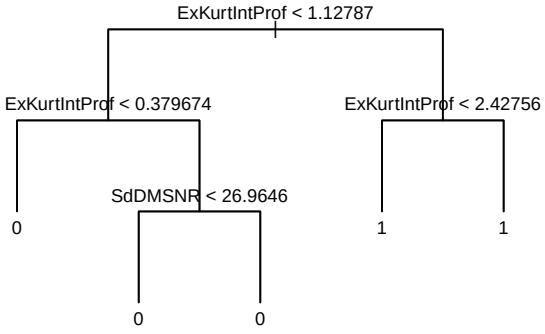
Figure 1: Bagging visualization: Creating several trees!

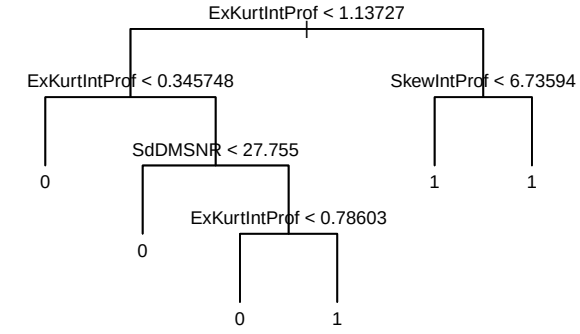
```
  tree <- tree(pulsar ~ ., data = boot)
  plot(tree, type = "uniform")
  text(tree)

}
```

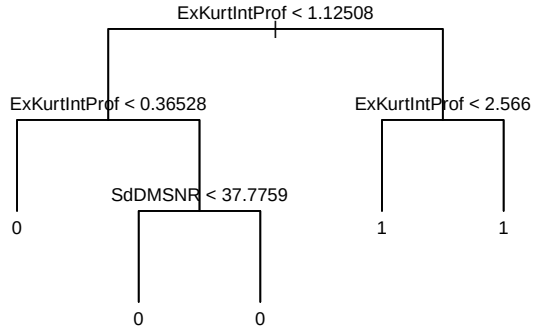













```
par(mfrow = c(1,1))
```

Bagging trees: Variable Importance Measures

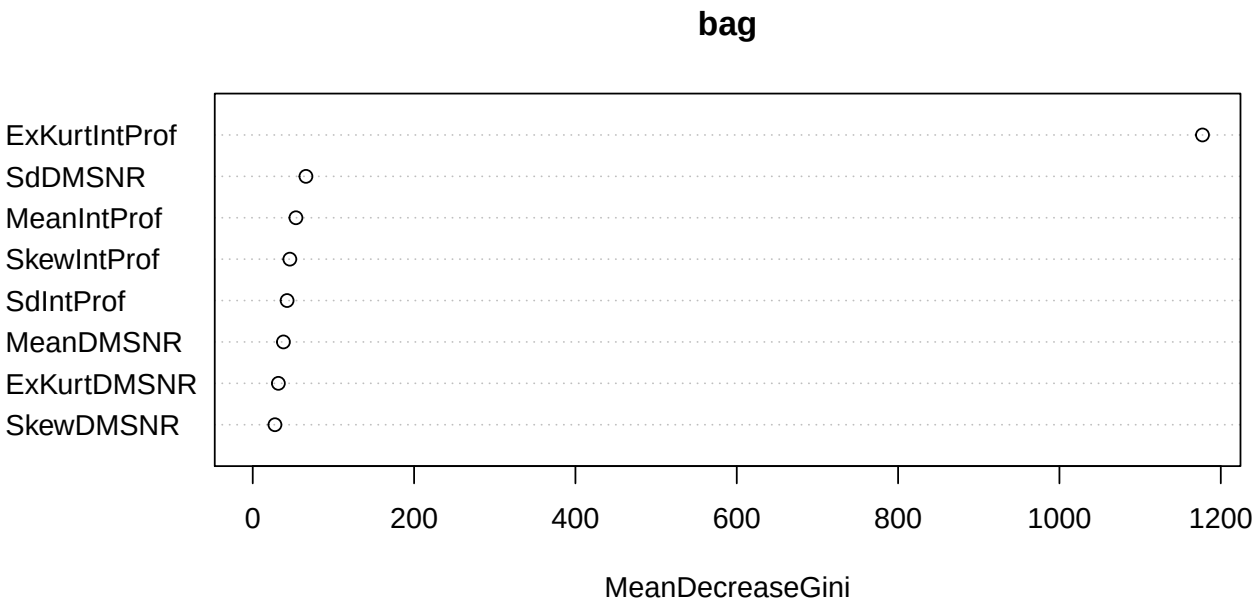
Though bagging can improve predictive performance of trees, the trade-off is we sacrificed some interpretability.

We no longer have that nice diagram that shows the segmentation of the predictor space (or, more accurately, we have B of them).

To recover some information, we can do the following:

1. For each of the b trees and each of the p variables, we record the amount that the Gini index is reduced by the addition of that variable
2. Report the average reduction over all B trees.

```
library(randomForest)

bag = randomForest(pulsar ~ .,data=stars, subset=train.ind, mtry=ncol(stars)-1)

predclass$bag = predict(bag, test_data)

out = varImpPlot(bag)
```



Notice that `ExKurtIntProf` is EXTREMELY important compared to the rest of the predictors.

Should we mostly dismiss the other predictors? (I don't know... I'm not an Astrophysicist.)

Out-of-Bag error estimation (OOB)

One can show that, on average, drawing n samples from n observations with replacement (also known as bootstrap) results in about 2/3 of the observations being selected.

The remaining one-third of the observations not used are referred to as **out-of-bag (OOB)**.

We can think of it as a for-free cross-validation.

Each time a tree is grown, we can get its prediction error on the unused observations. We average this over all bootstrap samples.

```
bag.test <-randomForest(pulsar ~ .,data=stars, subset=train.ind, mtry=ncol(stars)-1,
                        xtest=stars[test.ind,-9], ytest = stars[test.ind,9])
```

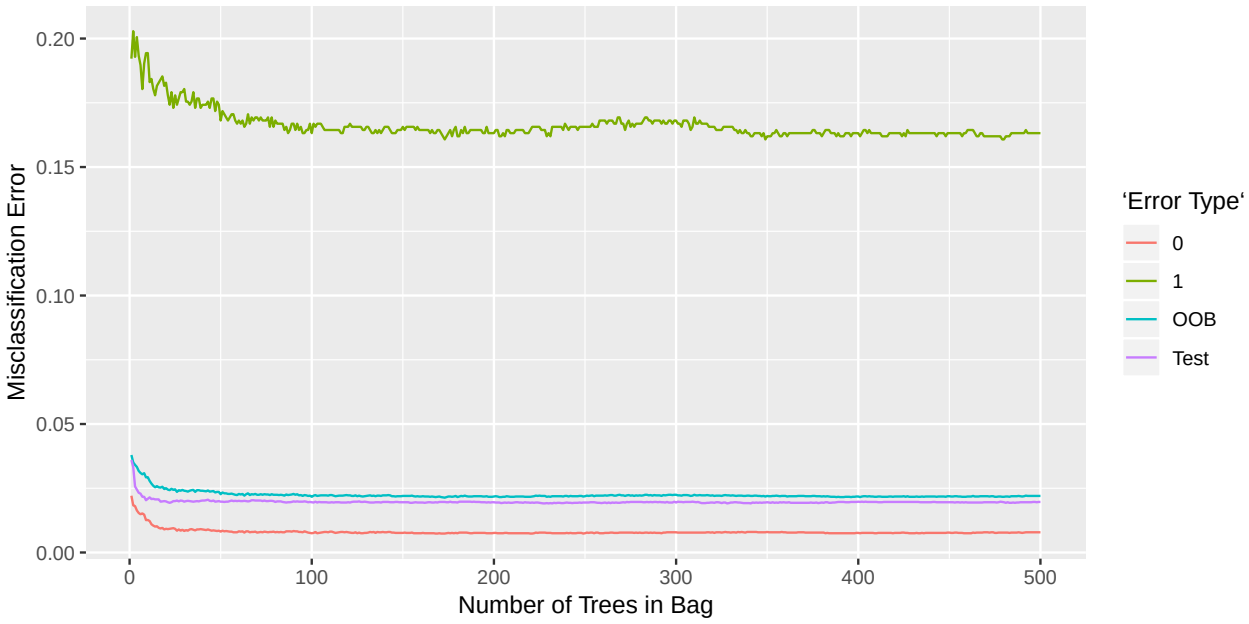
```
bag.test
```

```
##
## Call:
##  randomForest(formula = pulsar ~ ., data = stars, mtry = ncol(stars) -      1, xtest = stars[test.in
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 2.2%
## Confusion matrix:
##      0   1 class.error
## 0 8070  64 0.007868208
## 1  133 682 0.163190184
##                 Test set error rate: 1.97%
## Confusion matrix:
##      0   1 class.error
## 0 8065  60 0.007384615
## 1  116 708 0.140776699
```

```
head(bag.test$err.rate)
```

```
##             OOB          0         1
## [1,] 0.03793627 0.02208835 0.1921824
## [2,] 0.03504543 0.01834862 0.2028689
## [3,] 0.03409260 0.01803279 0.1928687
## [4,] 0.03329796 0.01636231 0.2005772
## [5,] 0.03174799 0.01537415 0.1932886
## [6,] 0.03094984 0.01488639 0.1896774
```

```
df <- as.data.frame(bag.test$err.rate)
df$Trees <- 1:nrow(bag.test$err.rate)
df$Test <- bag.test$test$err.rate[,1]

gather(df, "Error Type", "Error", -Trees) %>%
  ggplot(aes(x = Trees, y = Error, color = `Error Type`)) + geom_line() +
  xlab("Number of Trees in Bag") +
  ylab("Misclassification Error")
```



Random Forest: Decorrelated Bagging

Random Forest is a small extension of Bagging, in which the bootstrap trees are decorrelated (sort of...).

The idea is, we draw a bootstrap sample and start to build a tree.

- At each split, we randomly select m of the possible p predictors as candidates for the split.
- A new sample of size m of the predictors is taken at each split.

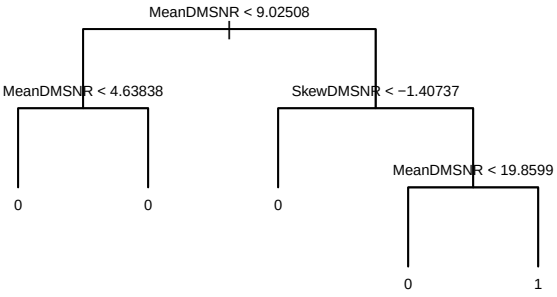
Usually, we use about $m = \sqrt{p}$

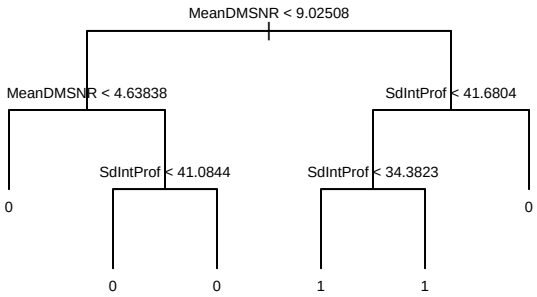
(this would be 2 predictors for `stars` data).

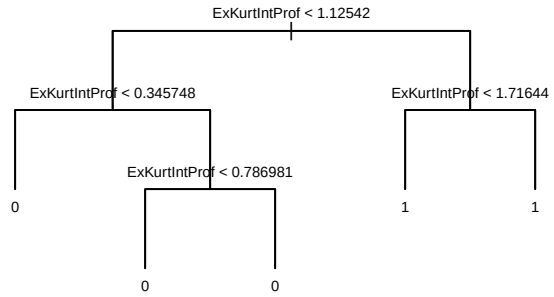
In other words, at each split, we aren't even allowed to consider the majority of possible predictors!

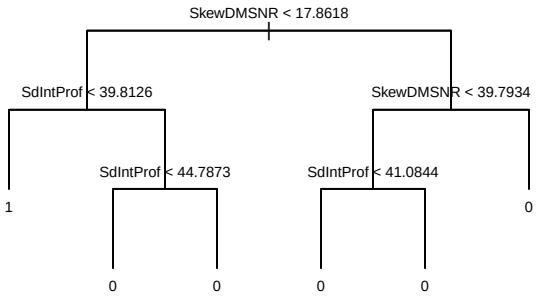
Note: The following demonstration shows a pseudo-random forest. The same two randomly selected predictors are used for the whole tree instead of a random selection of predictors at each branch.

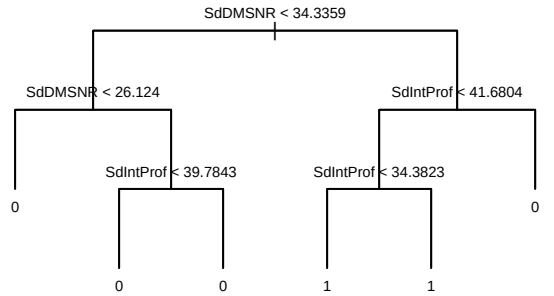
```
par(mfrow = c(3,2))

for(i in 1:6){

  columns <- sample(1:(ncol(stars)-1), sqrt(ncol(stars)-1))
  predictors.char <- paste(names(stars)[columns], collapse = " + ")

  formula <- as.formula(paste0("pulsar ~", predictors.char))

  rtree <- tree(formula, data = stars)

  plot(rtree, type = "uniform")
  text(rtree, cex=0.8)

}
```











Why Random Forests?: Problems With Bagging

What is going on here?

Suppose there is 1 really strong predictor and many mediocre ones.

- Then each tree will have this one predictor in it,
- Therefore, each tree will look very similar (i.e. highly correlated).

- Averaging highly correlated things leads to much less variance reduction than if they were uncorrelated.

If we don't allow some trees/splits to use this important variable, each of the trees will be much less similar and hence much less correlated.

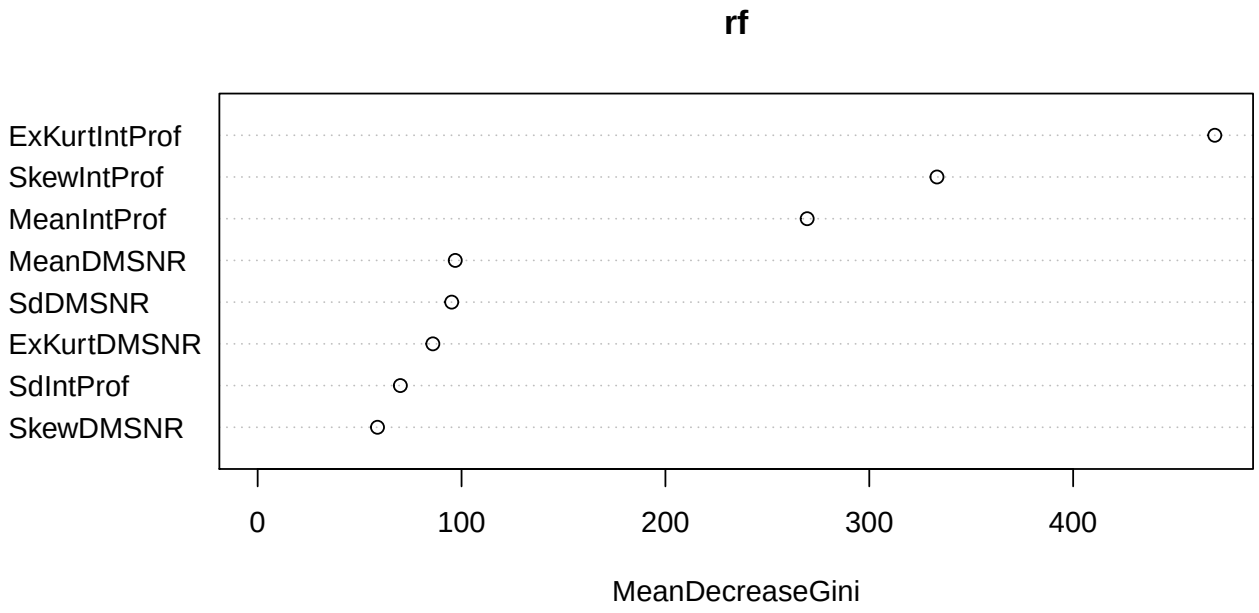
Bagging Trees is Random Forest when $m = p$, that is, when we can consider all the variables at each split.

Variable Importance in Random Forests

The form for getting variable importance is the same for random forests. We record the average of how well each variable improves the trees.

How does the variable importance differ in the random forest versus the tree bagging? Why would we see this?

```
library(randomForest)

rf = randomForest(pulsar ~ ., data=stars, subset=train.ind)

predclass$rf = predict(rf, test_data)

out = varImpPlot(rf)
```



OOB Error Estimation in Random Forests

The idea of OOB error estimation is the same in random forests.

- Get a bootstrapped sample.
- Build a tree.
- Estimate test error using observations left out of bootstrapped sample.

```
rf.test <-randomForest(pulsar ~ .,data=stars, subset=train.ind, mtry=sqrt(ncol(stars)-1),
                       xtest=stars[test.ind,-9], ytest = stars[test.ind,9])
```

```
rm(df)

rf.test
```

```
##
## Call:
##  randomForest(formula = pulsar ~ ., data = stars, mtry = sqrt(ncol(stars) -      1), xtest = stars[te
##                Type of random forest: classification
```

```
##                       Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 2.26%
## Confusion matrix:
##      0   1 class.error
## 0 8068  66 0.008114089
## 1  136 679 0.166871166
##                 Test set error rate: 1.82%
## Confusion matrix:
##      0   1 class.error
## 0 8074  51 0.006276923
## 1  112 712 0.135922330
```

```
head(rf.test$err.rate)
```

```
##             OOB          0         1
## [1,] 0.03116490 0.01941428 0.1654135
## [2,] 0.03350611 0.01896691 0.1950673
## [3,] 0.03322903 0.01762115 0.1975945
## [4,] 0.03135750 0.01607997 0.1923664
## [5,] 0.03046062 0.01426824 0.1966527
## [6,] 0.02995364 0.01475196 0.1845950
```

```
df <- as.data.frame(rf.test$err.rate)
df$Trees <- 1:nrow(rf.test$err.rate)
df$Test <- rf.test$test$err.rate[,1]

gather(df, "Error Type", "Error", -Trees) %>%
  ggplot(aes(x = Trees, y = Error, color = `Error Type`)) + geom_line() +
  xlab("Number of Trees in Bag") +
  ylab("Misclassification Error")
```

Comparing Bagging OOB Error to Random Forest

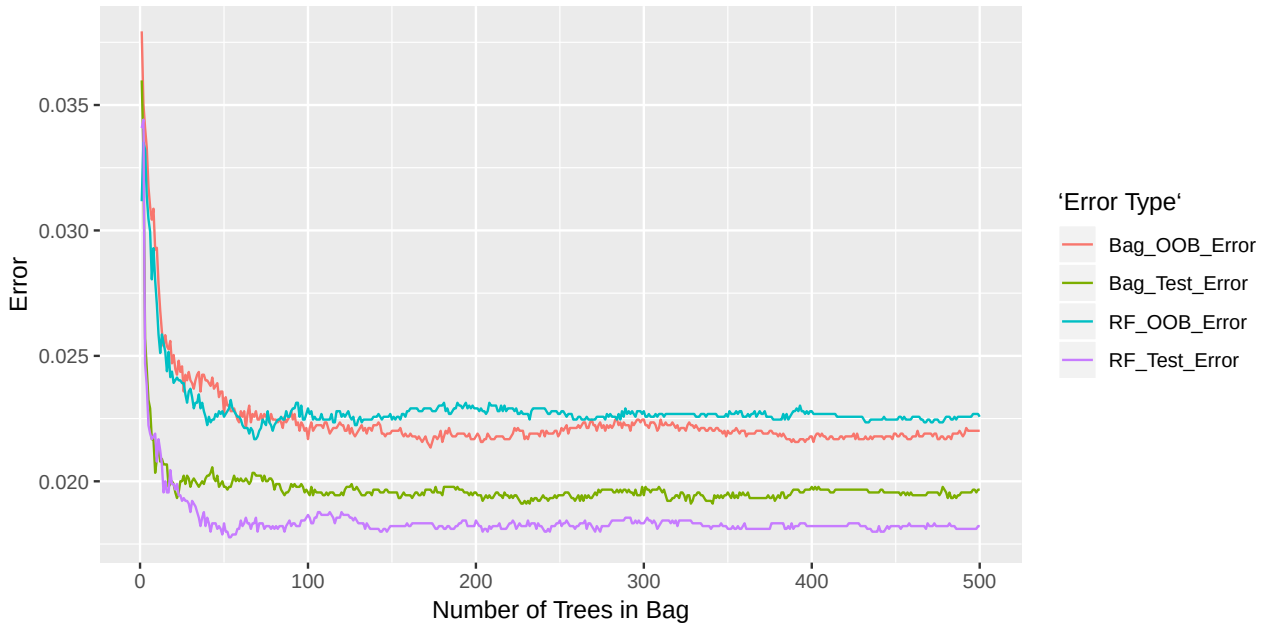
```
bag.test
```

```
##
## Call:
##  randomForest(formula = pulsar ~ ., data = stars, mtry = ncol(stars) -      1, xtest = stars[test.in
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 2.2%
## Confusion matrix:
##      0   1 class.error
## 0 8070  64 0.007868208
## 1  133 682 0.163190184
##                 Test set error rate: 1.97%
## Confusion matrix:
##      0   1 class.error
## 0 8065  60 0.007384615
## 1  116 708 0.140776699
```

```
rf.test
```

```
##
## Call:
##  randomForest(formula = pulsar ~ ., data = stars, mtry = sqrt(ncol(stars) -      1), xtest = stars[t
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 2.26%
## Confusion matrix:
##      0   1 class.error
## 0 8068  66 0.008114089
## 1  136 679 0.166871166
##                 Test set error rate: 1.82%
## Confusion matrix:
##      0   1 class.error
## 0 8074  51 0.006276923
## 1  112 712 0.135922330
```

```
df <- data.frame(Bag_OOB_Error = bag.test$err.rate[,1], Bag_Test_Error = bag.test$test$err.rate[,1],
                 RF_OOB_Error = rf.test$err.rate[,1], RF_Test_Error = rf.test$test$err.rate[,1])
df$Trees <- 1:nrow(bag.test$err.rate)

gather(df, "Error Type", "Error", -Trees) %>%
  ggplot(aes(x = Trees, y = Error, color = `Error Type`)) + geom_line() +
  xlab("Number of Trees in Bag") +
  ylab("Error")
```



Reporting results

There are two main ways classification results are reported:

- Sensitivity/specificity
- Confusion matrix

Reporting results: Sensitivity/specificity

Sensitivity: The proportion of times we label 'recession', given that 'recession' is the correct answer. (True +)
Specificity: The proportion of times we label 'no recession', given that 'no recession' is the correct answer. (True −)

We can think of this in terms of hypothesis testing. If

$$H_0: \text{ no recession},$$

then

Sensitivity: $P(\text{reject } H_0 | H_0 \text{ is false})$, that is: 1 - P(Type II error)
Specificity: $P(\text{accept } H_0 | H_0 \text{ is true})$, that is: 1 - P(Type I error)

Tree results: Confusion matrices

```
predclass$null = as.factor(rep(0,length(test.ind)))
```

NULL MODEL: All stars are not pulsars.

```
confusionMatrix(predclass$null, predclass$truth, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8125  824
##          1    0    0
##
##                Accuracy : 0.9079
##                  95% CI : (0.9017, 0.9138)
##     No Information Rate : 0.9079
##     P-Value [Acc > NIR] : 0.5093
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.90792
##              Prevalence : 0.09208
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

Tree Model

```
confusionMatrix(predclass$tree, predclass$truth, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8073  139
##          1   52  685
##
##                Accuracy : 0.9787
##                  95% CI : (0.9754, 0.9816)
##     No Information Rate : 0.9079
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.866
##  Mcnemar's Test P-Value : 4.885e-10
##
##             Sensitivity : 0.83131
##             Specificity : 0.99360
##          Pos Pred Value : 0.92944
##          Neg Pred Value : 0.98307
##              Prevalence : 0.09208
##          Detection Rate : 0.07654
##    Detection Prevalence : 0.08236
```

```
##       Balanced Accuracy : 0.91246
##
##        'Positive' Class : 1
##
```

Bagging

```
confusionMatrix(predclass$bag, predclass$truth, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8063  116
##          1   62  708
##
##                Accuracy : 0.9801
##                  95% CI : (0.977, 0.9829)
##     No Information Rate : 0.9079
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8774
##  Mcnemar's Test P-Value : 7.112e-05
##
##             Sensitivity : 0.85922
##             Specificity : 0.99237
##          Pos Pred Value : 0.91948
##          Neg Pred Value : 0.98582
##              Prevalence : 0.09208
##          Detection Rate : 0.07911
##    Detection Prevalence : 0.08604
##       Balanced Accuracy : 0.92580
##
##        'Positive' Class : 1
##
```

Random Forest

```
confusionMatrix(predclass$rf, predclass$truth, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8075  116
##          1   50  708
##
##                Accuracy : 0.9815
##                  95% CI : (0.9784, 0.9841)
##     No Information Rate : 0.9079
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.8849
##  Mcnemar's Test P-Value : 4.536e-07
##
##             Sensitivity : 0.85922
##             Specificity : 0.99385
##          Pos Pred Value : 0.93404
##          Neg Pred Value : 0.98584
##              Prevalence : 0.09208
##          Detection Rate : 0.07911
##    Detection Prevalence : 0.08470
##       Balanced Accuracy : 0.92653
##
##        'Positive' Class : 1
##
```

Classification Methods Overall

There are many, many different classification algorithms.

Different methods will work bettter in different situations.

Linear: Logisitic regression, linear discriminant analysis (LDA),
GLMNET, separating hyperplanes

Non-linear: quadratic discriminant analysis (QDA), trees (and associated methods), K-nearest neighbors (KNN), Support vector machines (SVM), Neural networks