MATH 4322 - Lecture 18

Tree Based Methods: Random Forest

Dr. Cathy Poliak, cpoliak@uh.edu

University of Houston

Recall Building a Tree

Below are the steps of building a decision tree (ISLR, page 309):

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$;
   (a) Repeat steps 1 and 2 on all but the kth fold of the training data.
   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.
4. Average the results for each value of α, and pick α to minimize the average error.
5. Return the subtree from Step 2 that corresponds to the chosen value of α.

Growing a Classification Tree

- Task of growing a classification tree is similar to the regression trees, we use recursive binary splitting to grow a classification tree.
- Recall criterion for regression tree is **RSS**.

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Since the response is categorical we cannot calculate the residual standard error. Thus we use other criterion to grow a classification tree.

- Classification error rate - The fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max_k(\hat{p}_{mk}).$$

- Gini index - A measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- Entropy - information gain measurement $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

Decision Trees: Advantages and Disadvantages

Several **advantages** of decision trees as a model:

1. Easy to explain, visualize and interpret.
2. More closely mirror human decision-making than regression and certain other methods.
3. Easily handle both quantitative and qualitative predictors (and responses).

The biggest downside:

1. Trees generally do not have the same level of predictive accuracy as some other regression and classification approaches.

However, by aggregating many decision trees (e.g. bagging, random forests), the predictive performance of trees can be vastly improved.

Decision Trees: Issues

Decision trees have the following disadvantages:

- They struggle with prediction accuracy.
- They suffer from high variance - different subsets of the same data could yield drastically different results.

Bagging

Since we do not have access to multiple training sets, we can bootstrap, by taking repeated samples from the (single) training data set.

1. We generate B different bootstrapped training data sets.
2. We train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$.
3. Average all the predictions to obtain

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

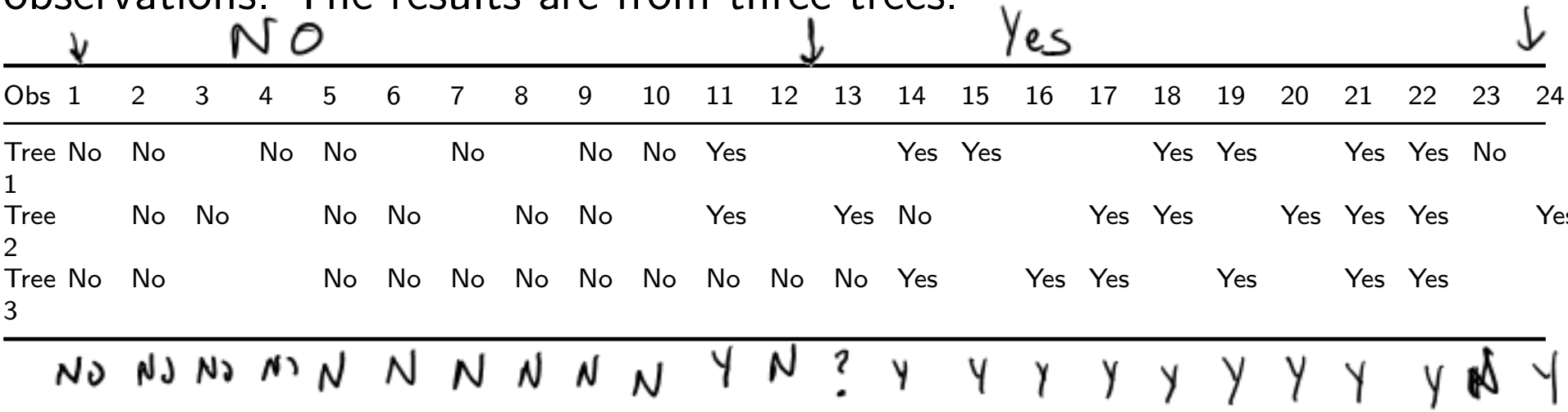
This is called **bagging** - Bootstrap AGGregatING.

Bagging for Decision Trees

- Bagging can improve predictions for many regression methods, it is particularly useful for decision trees.
- To apply bagging to regression trees:
  1. Construct B regression trees using B bootstrapped training sets.
  2. Average the resulting predictions.
- These trees are grown deep, and are not pruned. Thus each individual tree has high variance but low bias.
- Averaging these B trees reduces the variance.
- To apply bagging to classification:
  1. We record the class predicted by each of the B trees
  2. Take a *majority vote*: the overall prediction is the most commonly occurring class among the B predictions.

Mower Example

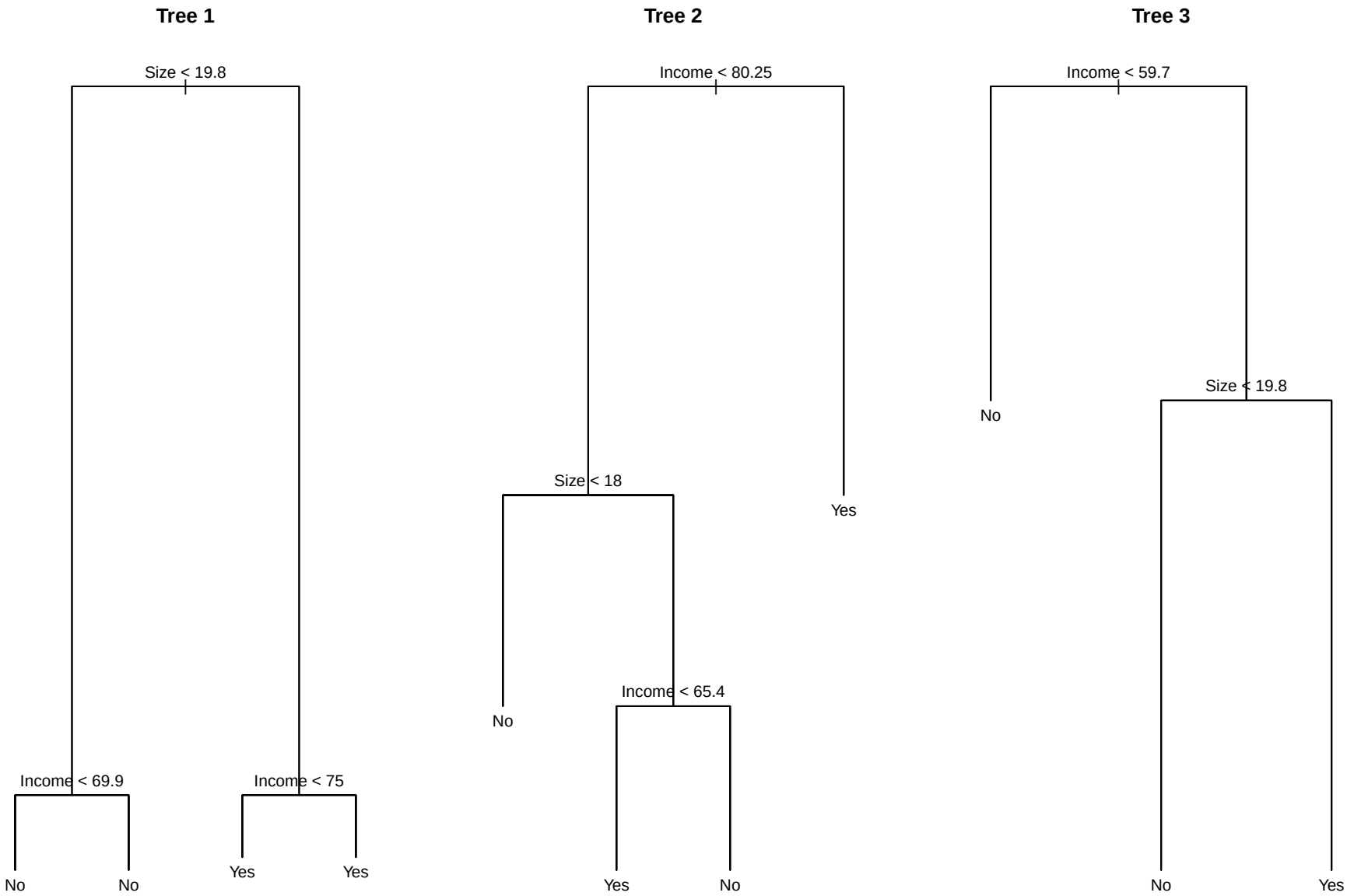
I took a random sample with replacement 3 times based on the 24 observations. The results are from three trees.



Obs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Tree 1	No	No		No	No		No		No	No	Yes			Yes	Yes			Yes	Yes		Yes	Yes	No	
Tree 2		No	No		No	No		No	No		Yes		Yes	No			Yes	Yes		Yes	Yes	Yes		Yes
Tree 3	No	No			No	No	No	No	No	No	No	No	No	Yes		Yes	Yes		Yes		Yes	Yes		

# of missclassified = 2

Trees


Tree 1
Size < 19.8
Income < 69.9
Income < 75
No
No
Yes
Yes
Tree 2
Income < 80.25
Size < 18
Income < 65.4
No
Yes
Yes
No
Tree 3
Income < 59.7
Size < 19.8
No
No
Yes

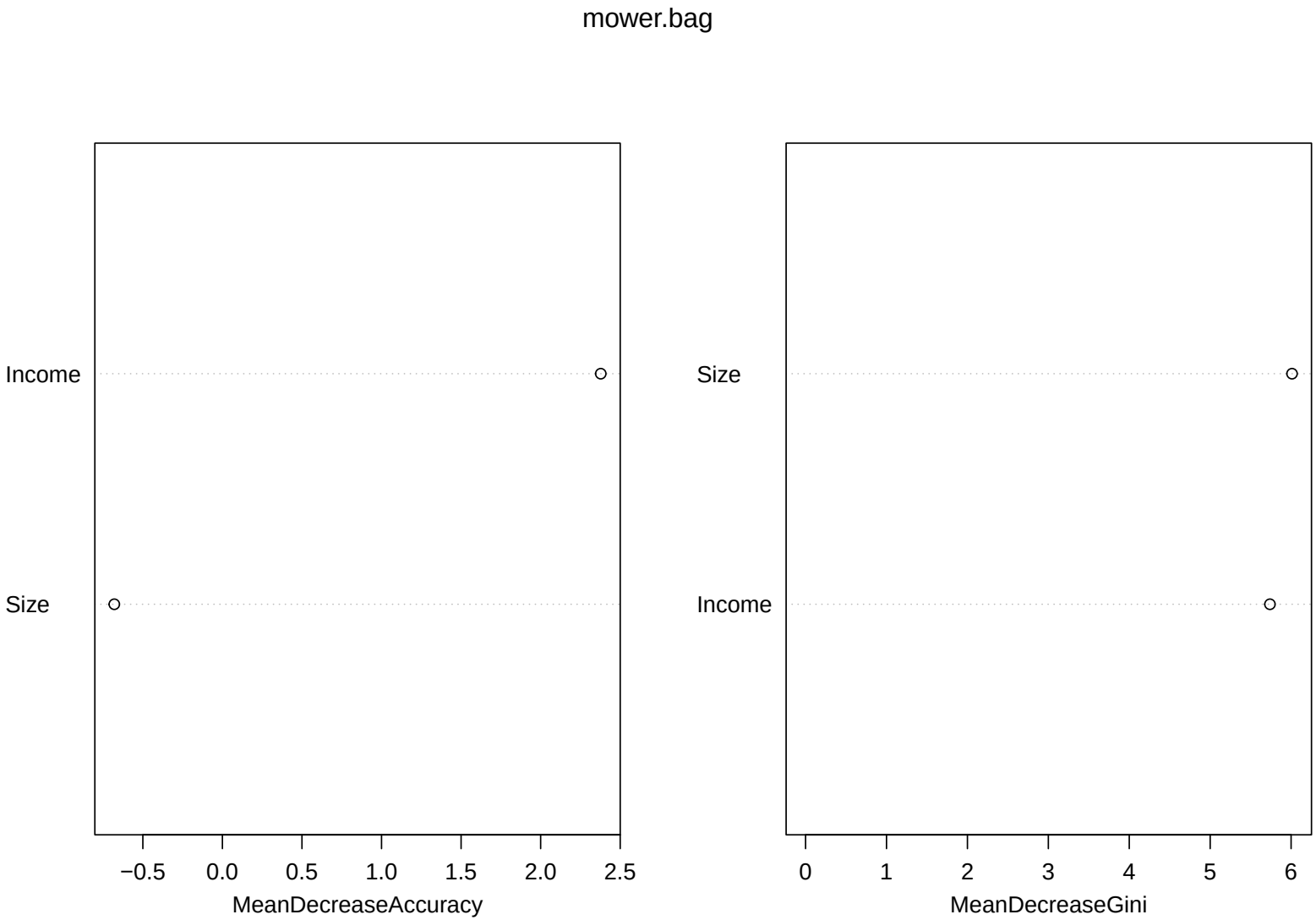
Variable Imporatance

- The **Mean Decrease Accuracy** plot expresses how much accuracy the model losses by excluding each variable. The more the accuracy suffers, the more important the variable is for the successful classification.

$$\sum \hat{p}_i (1 - \hat{p}_i)$$

- The **mean decrease in Gini coefficient** is calculated based on the mean decrease in the Gini index each time when the tree is split on that variable. R weighs the impurities by the raw counts, not the proportions.
- The higher the value of mean decrease accuracy or mean decrease Gini score, the higher the importance of the variable in the model.

Variable Importance Plot



Random Forests

- **Random forests** adds a small tweak to further improve on bagging:
  1. Random forests also grow B large un-pruned trees, but
  2. Each time a tree split is considered, it picks a random subset of $m \approx \sqrt{p}$ predictors from the full set of p predictors.

  If classification if regression $p/3$

  This leads to **decorreltaing** the bagged trees. Less chance to have same variable dominating each bagged tree (which would have led to high correlation between estimates).

- This stabilizes the variance of the estimate.
- If $m = p$, then this is bagging.

Random Forest in R

- Growing a random forest proceeds in exactly the same way as Boosting, except that we use a smaller value of the `mtry` argument.
- By default, `randomForest()` uses $p/3$ variables when building a random forest of regression trees and $\sqrt{p}$ when building a random forest of classification trees.

Random Forests In R

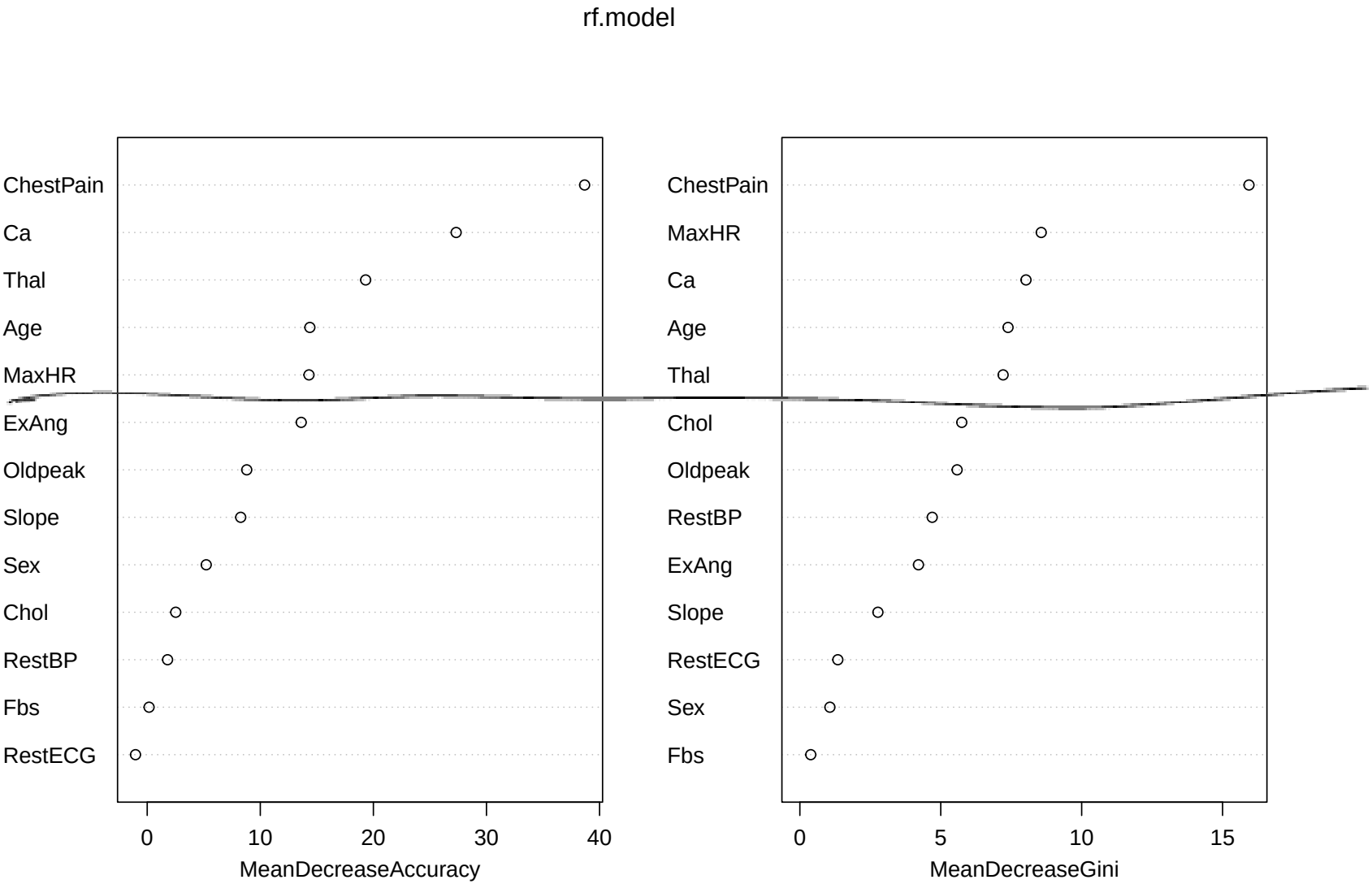
The *mtry* in the `randomForest()` function is changed to fit a random forest. This is the number of predictors to be considered at tree splits.

```
#Random Forest Model
# Using Heart Data set
library(randomForest)
Heart = na.omit(Heart); Heart$X = NULL
n = nrow(Heart); p = ncol(Heart)-1
B = 1000
set.seed(100)
train = sample(1:nrow(Heart),nrow(Heart)/2+0.5)
Heart$AHD = as.factor(Heart$AHD)
Heart$ChestPain = as.factor(Heart$ChestPain)
Heart$Sex = as.factor(Heart$Sex)
Heart$Thal = as.factor(Heart$Thal)
AHD.test = Heart[-train,"AHD"]
X.test = Heart[-train,-14] #Take away the AHD column
rf.model = randomForest(AHD ~., data = Heart, subset = train,
                        xtest = X.test, ytest = AHD.test,
                        ntree = B, mtry = sqrt(p), importance = TRUE)
```

Results

```
Call:
 randomForest(formula = AHD ~ ., data = Heart, xtest = X.test,      ytest = AHD.tes
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4 = √13

        OOB estimate of  error rate: 14.77%
Confusion matrix:  ← Training confusion matrix
    No Yes class.error
No  72  11   0.1325301
Yes 11  55   0.1666667
               Test set error rate: 17.57%
Confusion matrix:    ← test confusion matrix
    No Yes class.error
No  69   8   0.1038961
Yes 18  53   0.2535211
```

Variable Importance Plot



Random Forest for Regression

```
library(ISLR)
Hitters2 = na.omit(Hitters)
included.vars = names(Hitters[-c(3,8:15,20)])
set.seed(100)
train = sample(1:nrow(Hitters2),nrow(Hitters2)/2+0.5)
p = 9 #No. of predictors
B = 100 #No. of bootstrap trees
rf.hitters = randomForest(log(Salary) ~.,
                          Hitters2[,included.vars], subset = train,
                          ntree = B, keep.forest = TRUE)
```

Results

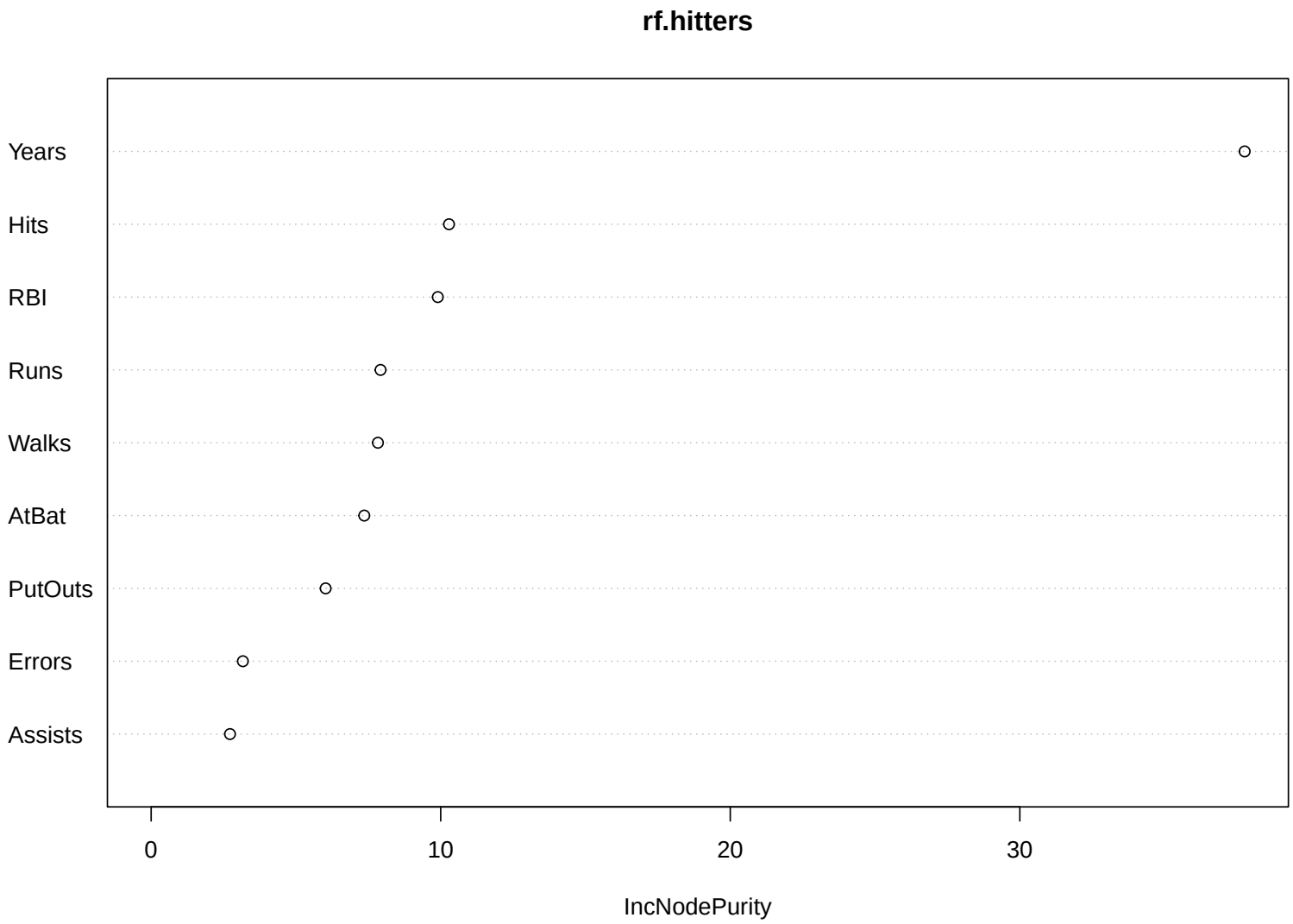
```
Call:
 randomForest(formula = log(Salary) ~ ., data = Hitters2[, included.vars],      ntr
               Type of random forest: regression
                     Number of trees: 100
No. of variables tried at each split: 3

          Mean of squared residuals: 0.2789084
                    % Var explained: 62.26
```

$= 9/3$

$= \frac{1}{n}\sum_{i=1}^{\hat{}} (y_i - \hat{y}_i)^2$

$\Rightarrow R^2$

Variable Importance Plot For Regression



Boosting

- **Boosting** is another approach for improving the predictions resulting from a decision tree.
- Also a general approach that can be applied the many statistical learning methods for regression or classification.
- In *boosting* each tree is grown using information from previously grown trees. Such that each the trees are grown sequentially.

Algorithm for Boosting

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, R).
   (b) Update $\hat{f}$ by adding a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
   (c) Update the results,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

Boosting Approach

- Learns slowly
- We fit a decision to the residuals from the models.
- We then add this new decision tree into the fitted function in order to update the residuals.
- The terminal nodes are determined by the parameter d in the algorithm.
- We slowly improve $\hat{f}$ in areas where it does not perform well.

Three Tuning Parameters

Boosting has three tuning parameters:

1. The number of trees B. Use cross-validation to select B. Otherwise boosting can overfit if B is too large.
2. The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typically, $\lambda = 0.01$ or 0.001.
3. The number of d splits in each tree. This controls the complexity of the boosted ensemble.

Boosting in R

- Boosting uses the gmb package and the `gmb()` function to fit the boosted regression trees.
- Again we will look at the Hitters data set.
- Since this is a regression problem we will use `distribution = "gaussian"`.
- If it was a classification problem, then we would use `distribution = "bernoulli"`.
- Type and run the following in R:

```
#install.packages("gbm")
library(gbm)
set.seed(1)
boost.hitters = gbm(log(Salary) ~.,
                    data = Hitters2[,included.vars],
                    distribution = "gaussian")
summary(boost.hitters)
```

Results

	var	rel.inf
Years	Years	55.1637439
RBI	RBI	12.0382143
Hits	Hits	9.6689698
Walks	Walks	9.0331049
PutOuts	PutOuts	6.0596017
Errors	Errors	2.9280571
AtBat	AtBat	2.7246434
Runs	Runs	1.7266823
Assists	Assists	0.6569826

MSE obtained Boosting

Type and run the following in R:

```
set.seed(100)
train = sample(1:nrow(Hitters2),nrow(Hitters2)/2+0.5)
hitters.test = Hitters2[-train,"Salary"]
yhat.boost = predict(boost.hitters,
                     newdata = Hitters2[-train,included.vars],
                     n.trees = 100)
mean((yhat.boost- log(hitters.test))^2)
```

```
[1] 0.258397
```

We can use a different value of the shrinking parameter. The default is 0.001, let's use $\lambda = 0.2$.

```
boost.hitters = gbm(log(Salary) ~. ,
                    data = Hitters2[,included.vars],
                    distribution = "gaussian",
                    shrinkage=0.2,verbose=F)
yhat.boost=predict(boost.hitters,
                   newdata=Hitters2[-train,included.vars],
                   n.trees=100)
mean((yhat.boost-log(hitters.test))^2)
```

```
[1] 0.2335436
```

MSE Compared

lm	tree	bagging	random forest	boosting
0.4718342	0.4214516	0.3061530	0.3134921	0.2204116