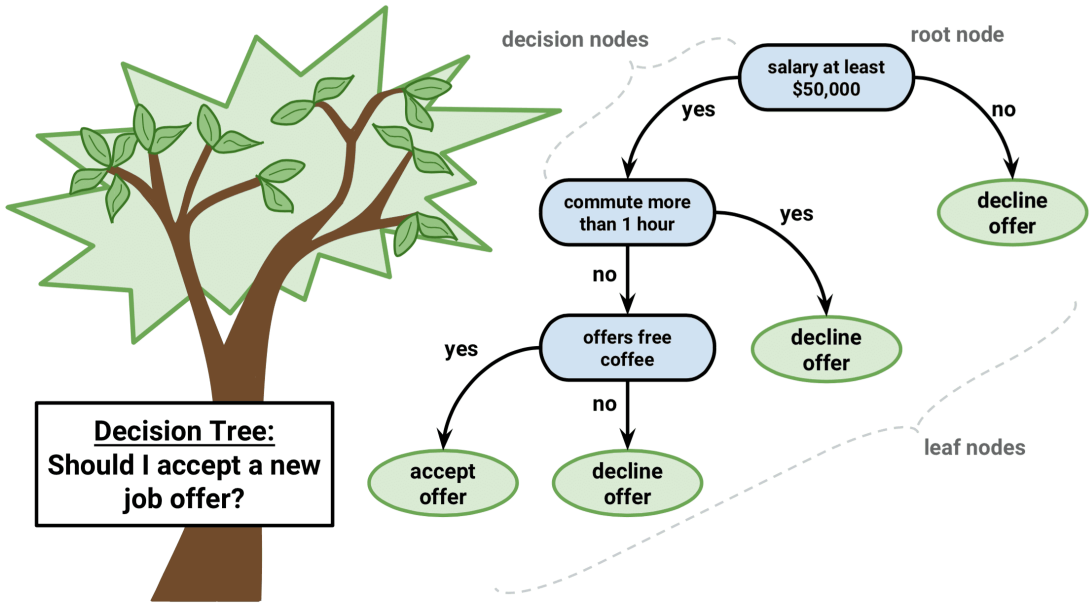
Classification and Regression Tree

Yang Wang

College of Charleston

Decision tree

- Goal: predict Y for given X
- Supervised method
- May be used for both classification and regression
- Decision is made by splitting parent node



picture: https://www.datacamp.com/community/tutorials/decision-trees-R

Classification tree

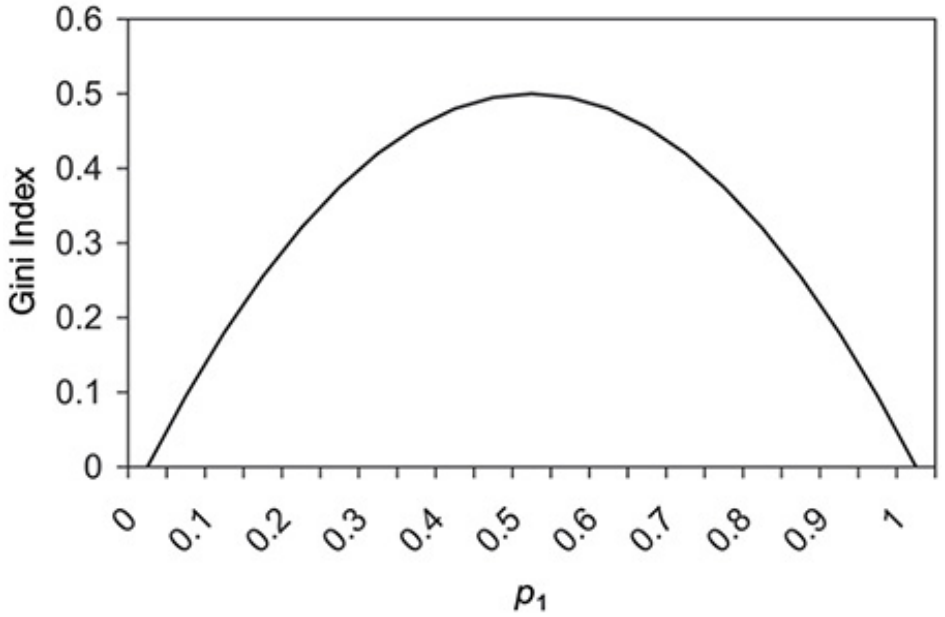
- Categorical predictors
  - To handle categorical predictors, group the categories into two subsets
  - Example: A categorical predictor with three categories {a, b, c} can be split in 3 ways into two subsets: {a} and {b, c}; {b} and {a, c}; {c} and {a, b}
- Normalization: not needed

Questions

- How to partition?
- When to stop partitioning?
- How to assign class labels to a new observation?

Measures of impurity

- Suppose there are m classes and p_k denotes that proportion of records belonging to k^{th} class in rectangle A
- Gini impurity index
  - $I(A) = 1 - \sum_{k=1}^{m} p_k^2$
  - When all records belong to the same class, $I(A) = 0$
  - When all m classes are equally represented, $I(A) = 1 - 1/m$
  - For a two-class problem, Gini impurity measure is at its peak when $p_k = 0.5$

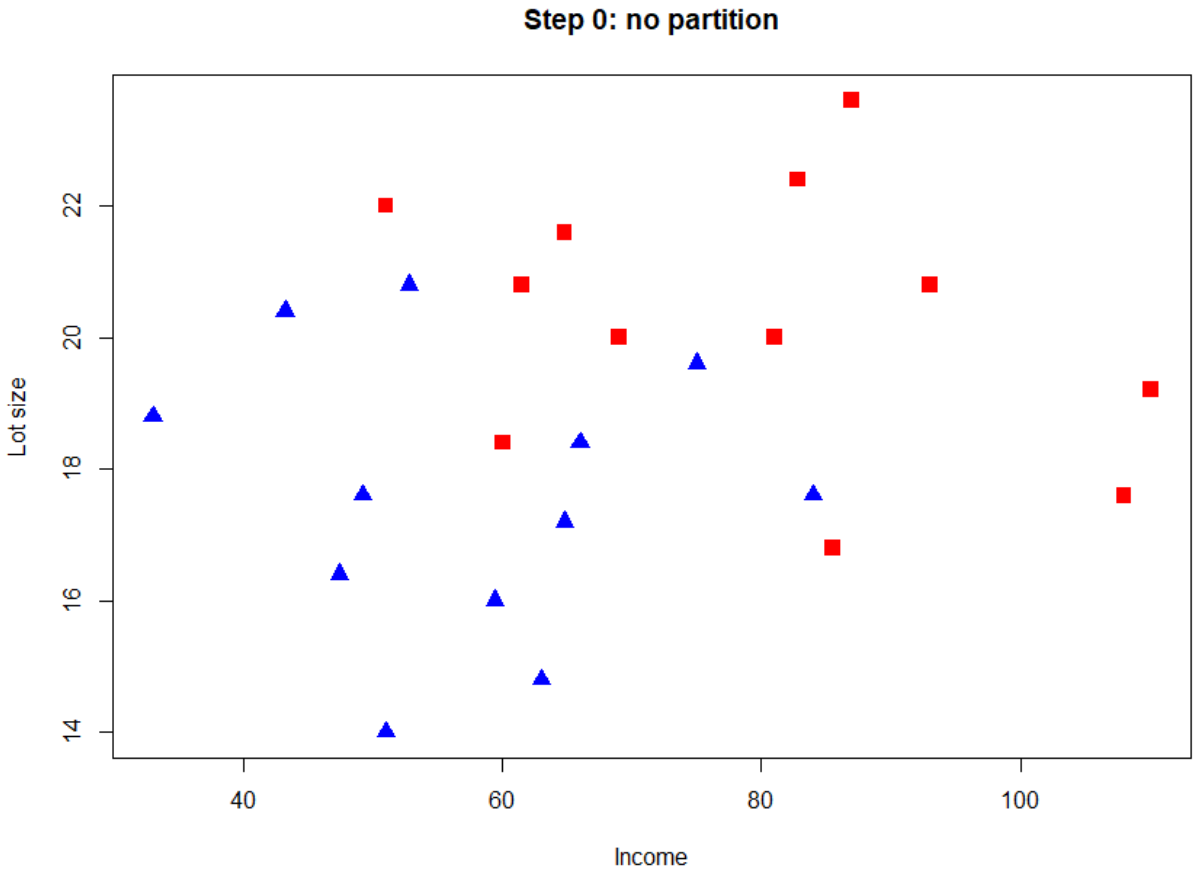


Measures of impurity

- Suppose there are m classes and p_k denotes that proportion of records belonging to k^{th} class in rectangle A
- Entropy
  - $entropy(A) = -\sum_{k=1}^{m} p_k log_2(p_k)$
  - When all records belong to the same class, $entropy(A) = 0$
  - When all m classes are equally represented, $I(A) = log_2(m)$
  - For a two-class problem, entropy measure is maximized when $p_k = 0.5$

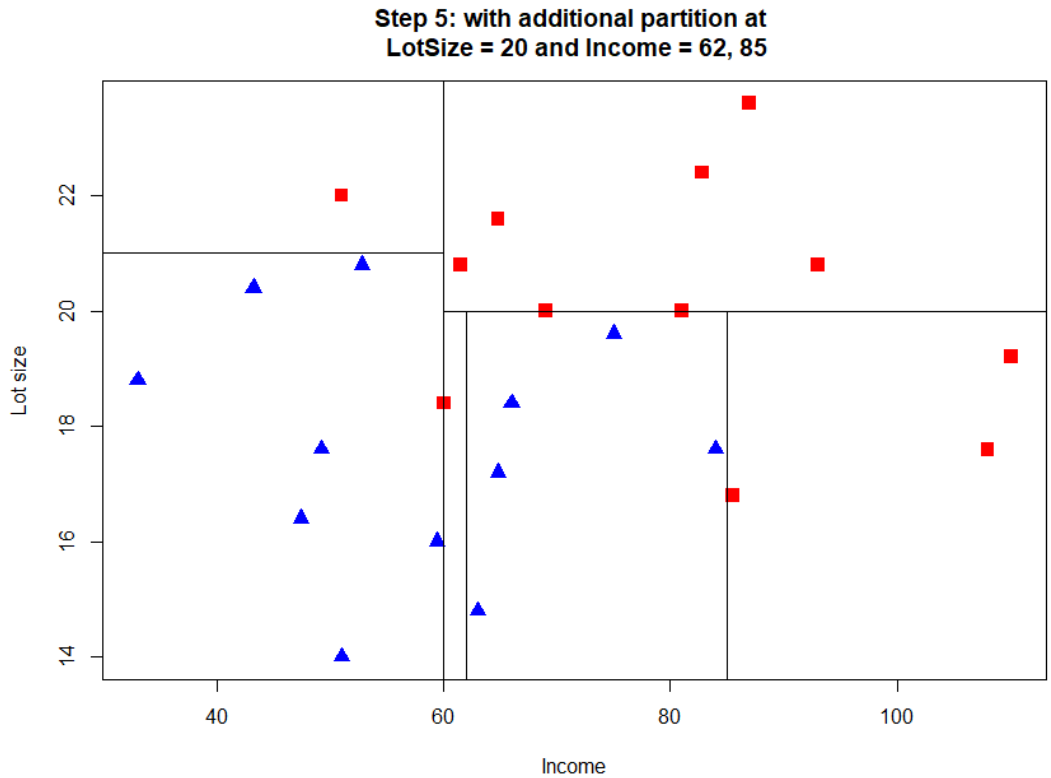
How to partition?

- Consider the dataset called RidingMowers.csv. The response variable Ownership denotes whether a family owns a riding mower or not.The predictors are Income and Lotsize of the associated household.
- Income ranges from $33-$110.1K. LotSize ranges from 14-23.6 sqft.
- There are 12 owners and 12 non-owners.



Classification tree

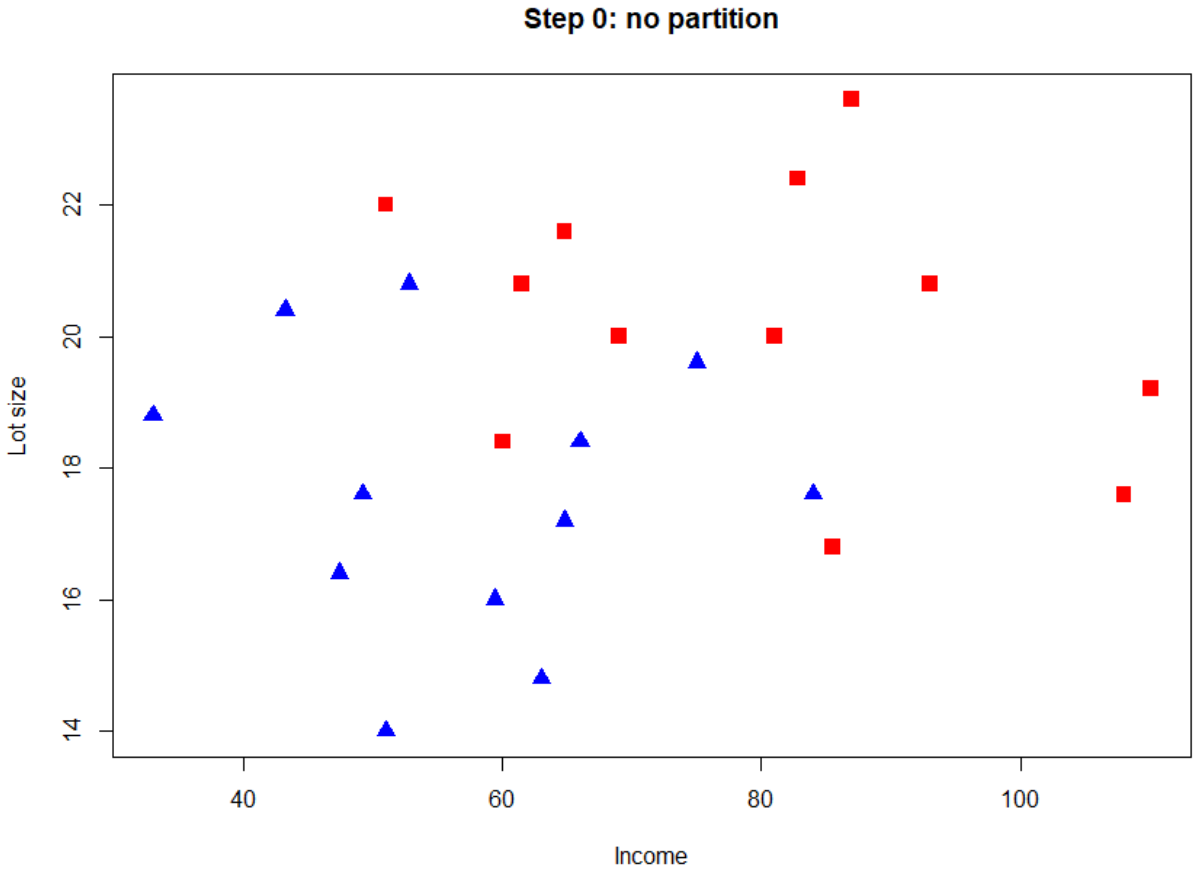
- Let Y be the response variable and $X_1, X_2, \ldots X_p$ be the predictors
- Recursive Partitioning
  - Divide the p-dimensional space or predictor variables into nonoverlapping multidimensional rectangles so that each rectangle is as homogeneous (or "pure") as possible
  - The partition is to be done recursively



Calculation of impurity measures

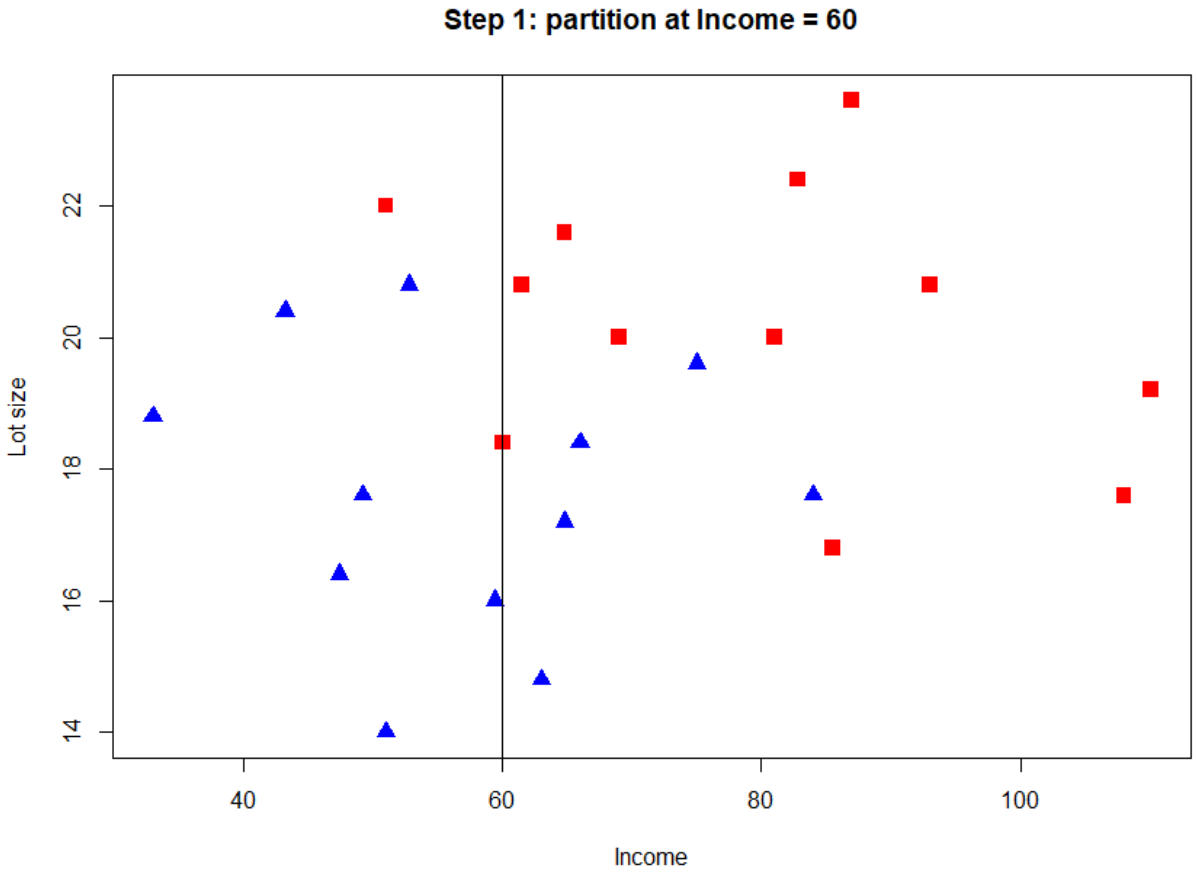
Step 0: 12 owners and 12 non-owners

- $Gini = 1 - (0.5^2 + 0.5^2) = 0.5$
- $entropy = -(0.5log_2(0.5) + 0.5log_2(0.5)) = -log_2(0.5) = 1$



How to partition?

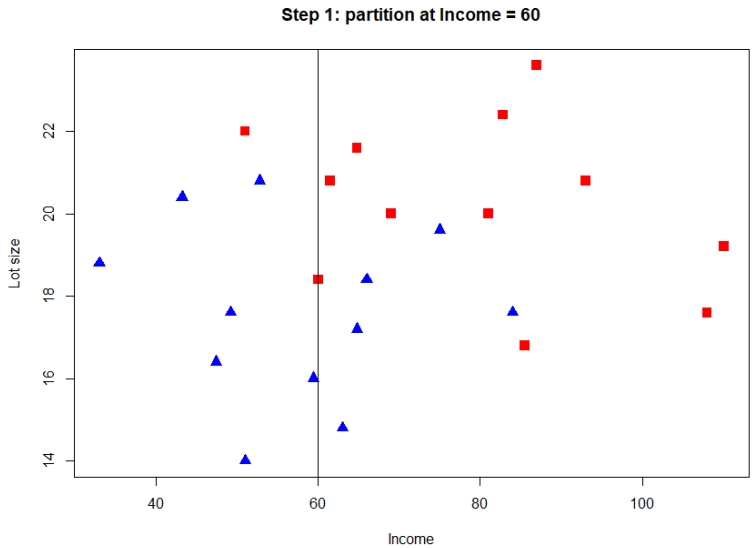
- Partition at Income=60
- Left rectangle has 7 non-owners and 1 owner
- Right rectangle has 5 non-owners and 11 owners



Calculation of impurity measures

Step 1: 7 non-owners and 1 owner in left rectangle; 5 non-owners and 11 owners in right rectangle

- $Gini_{left} = 1 - ((7/8)^2 + (1/8)^2) = 0.219$
- $entropy_{left} = -((7/8)log_2(7/8) + (1/8)log_2(1/8)) = 0.544$
- $Gini_{right} = 1 - ((5/16)^2 + (11/16)^2) = 0.430$
- $entropy_{right} = -((5/16)log_2(5/16) + (11/16)log_2(11/16)) = 0.896$
- For the combined impurity of the two rectangles created by the split, take a weighted average of the left and right impurity
- Combined Gini = $(8/24)0.219 + (16/24)0.430 = 0.359$
- Combined entropy = $(8/24)0.544 + (16/24)0.896 = 0.779$
- Reduction in Gini from step 0 to step 1 = $0.5 - 0.359 = 0.141$
- Reduction in entropy from step 0 to step 1 = $1 - 0.779 = 0.221$

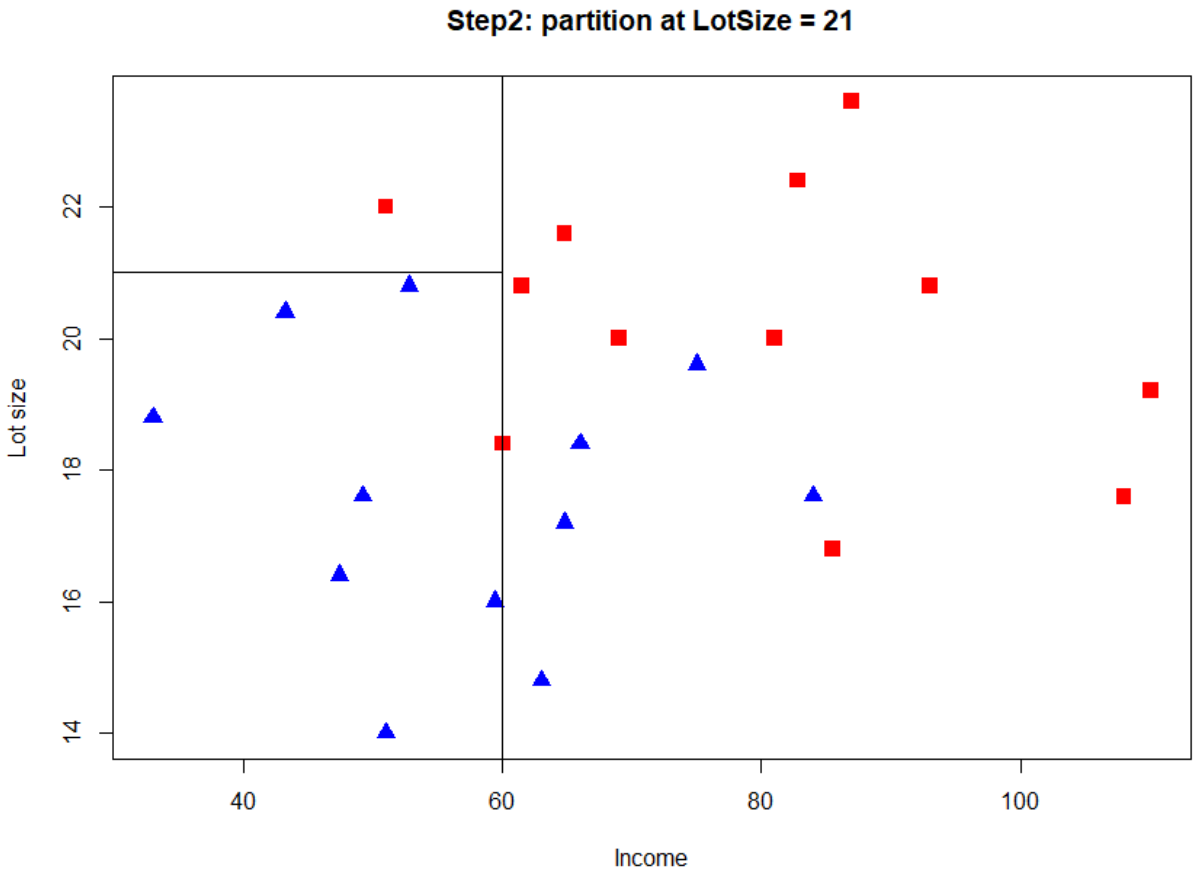


How to partition?

- Compare the reduction in impurity across all possible splits in all possible predictors
- Choose the split for which maximum reduction in impurity is achieved
- Continue to split until rectangles are ”pure”

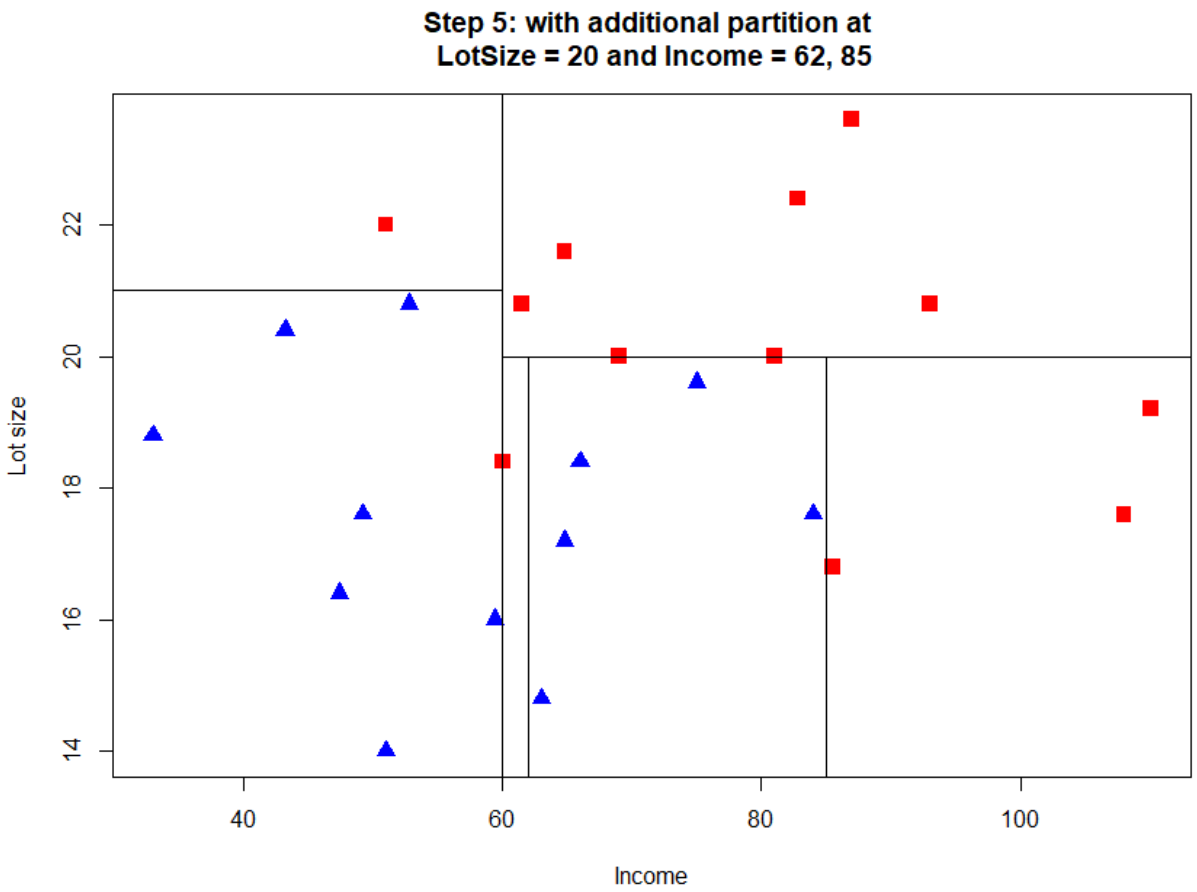
How to partition?

- Another partition in left rectangle at Lotsize=21
- Left top rectangle has 1 owner
- Left bottom rectangle has 7 non-owners
- Right rectangle has 5 non-owners and 11 owners

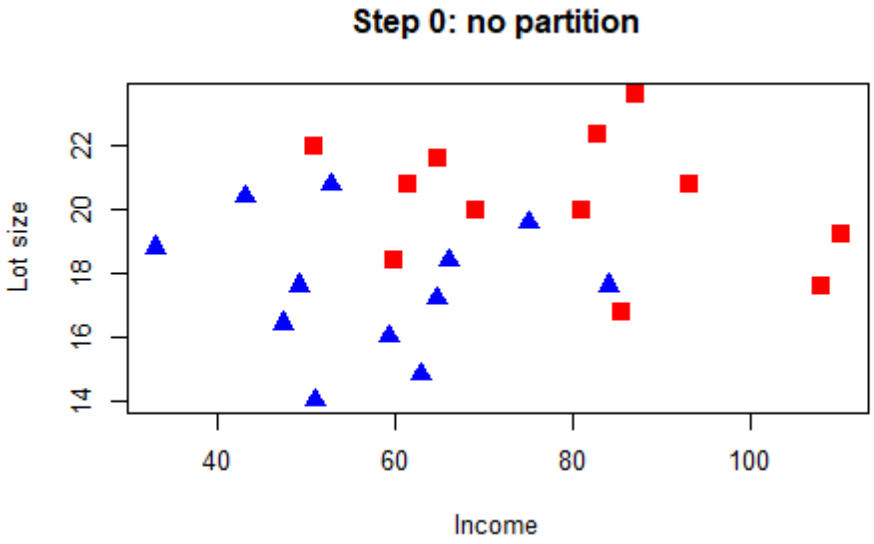


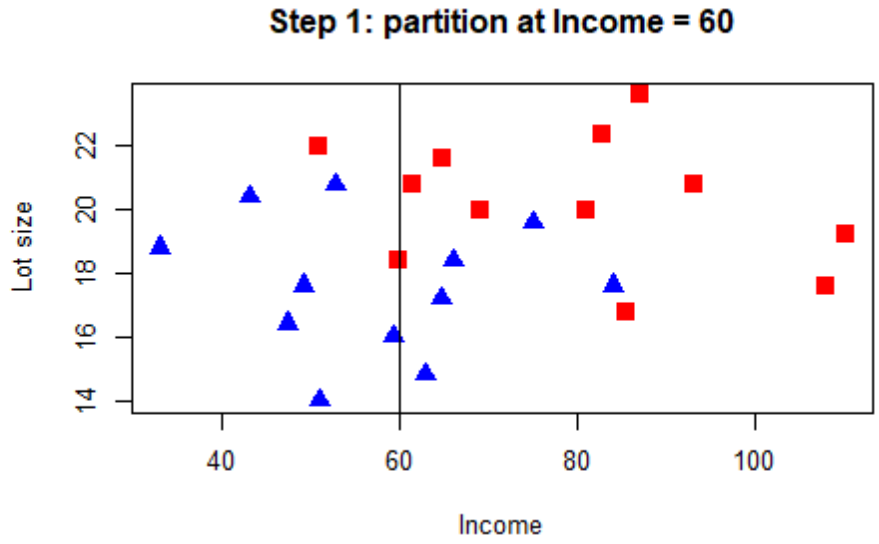
How to partition?

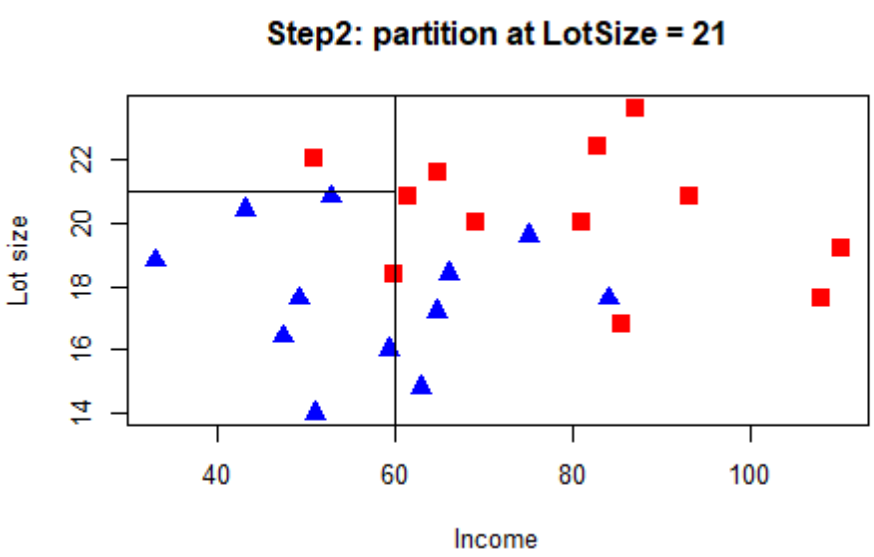
- More partitions in right rectangle
  - First at Lotsize = 20
  - Then the bottom right rectangle at Income = 62 and 85
- Now all rectangles are ”pure” (contain members of a single class)

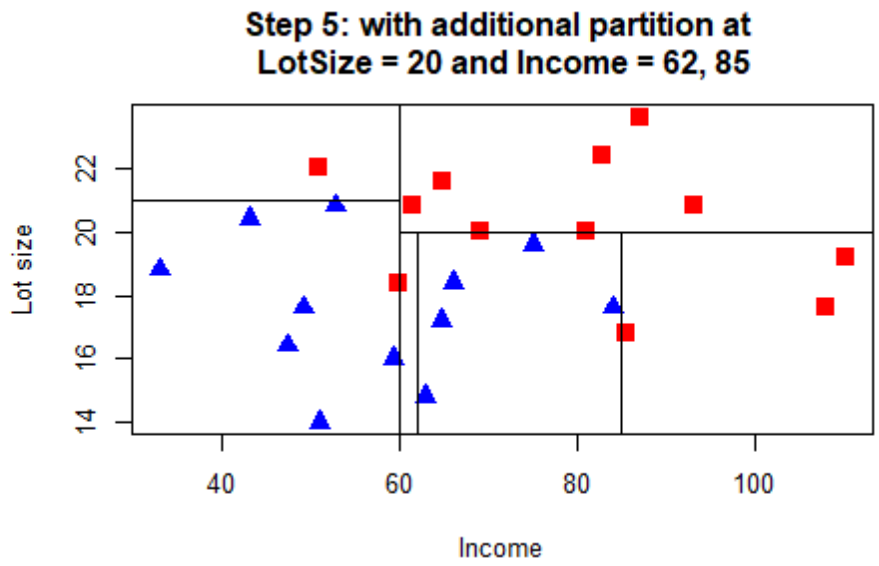


How to partition?



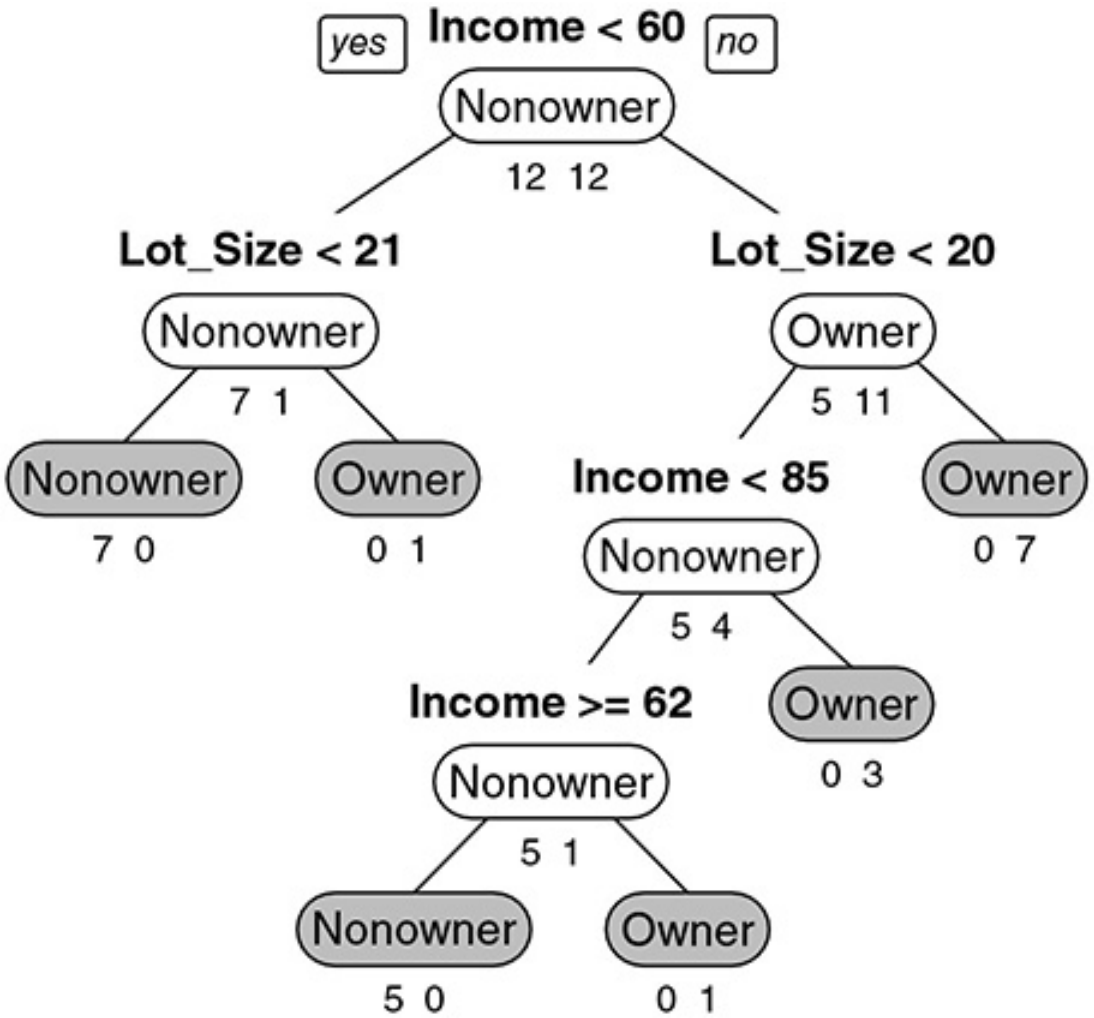






How to partition?

Decision rules from the partitioned rectangle

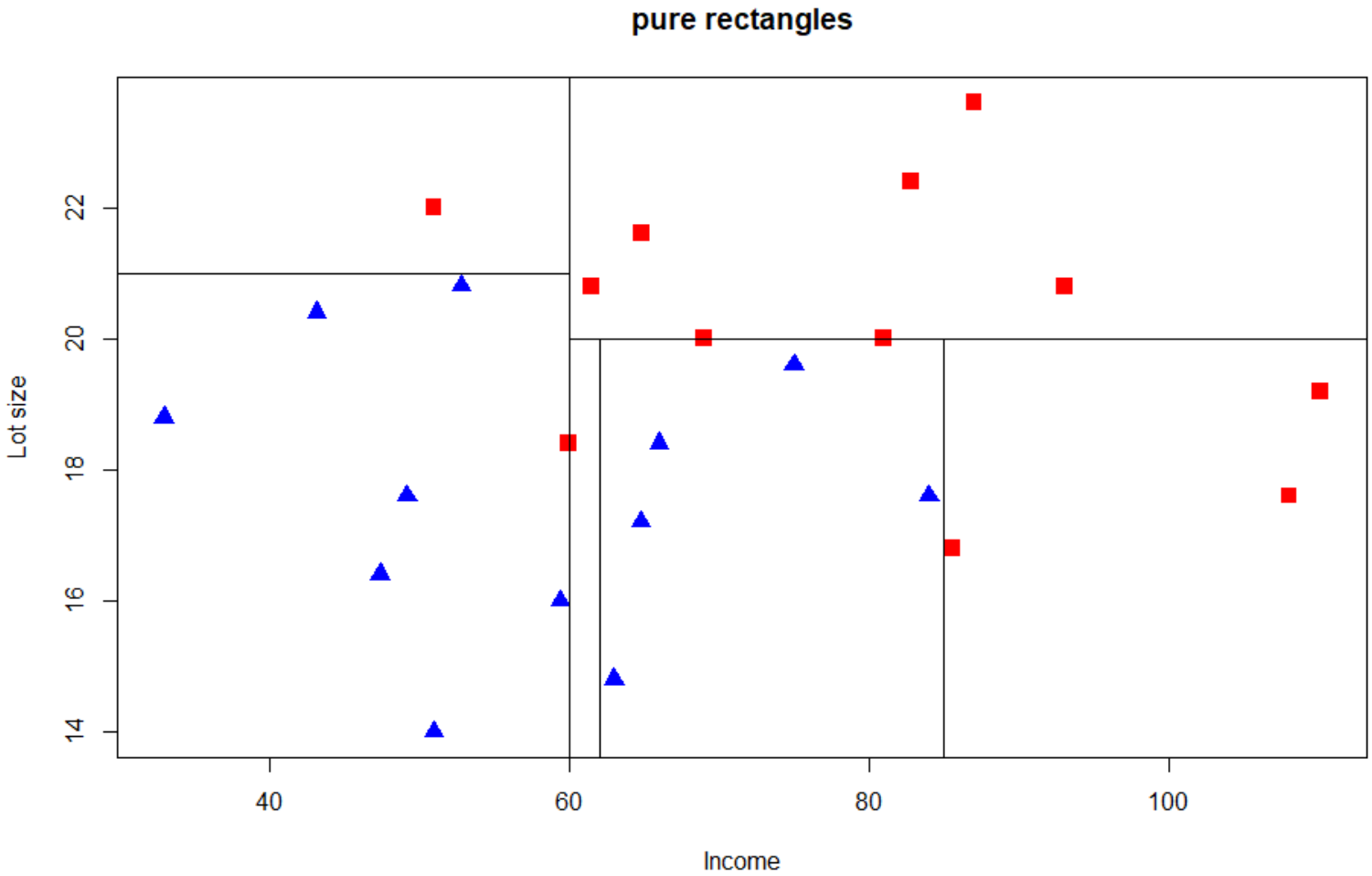


How to assign label to new observation?

- Identify the terminal (leaf) node into which the new observation belongs.
- The class with the highest occurance in that terminal nodel will be assigned to the new observation.
- If a single class is of higher importance, convert the counts of each class to a proportion and compare that proportion with a user-specified cutoff value for label assignment. Achieve this by specifying $type =' prob'$ in the *predict*() function.

Caution

- Tree structure can be quite unstable, shifting substantially depending on the sample chosen.
- A fully-grown tree will invariably lead to overfitting.



How to avoid overfitting?

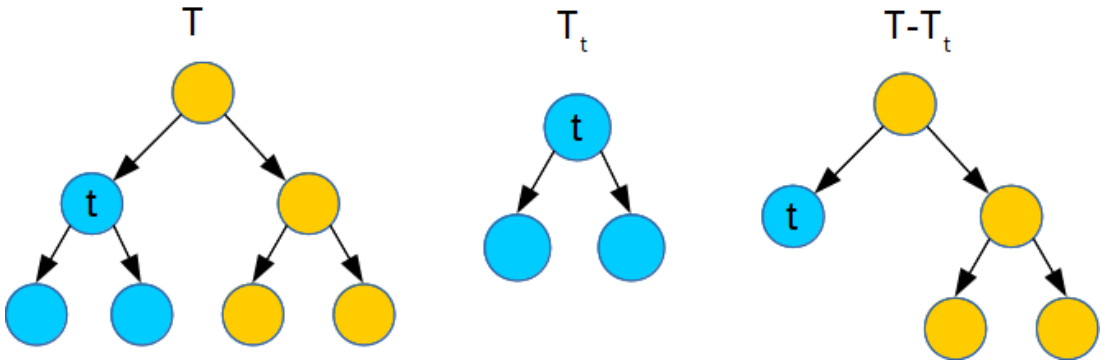
- Idea 1: Stop splitting when the reduction in in impurity is less than a pre-specified number δ
  - Problem: it is possible to find a good split after a bad split
- Idea 2: Stop splitting when the number of observations in the terminal nodes is smaller than a pre-specified number
  - It is possible to mention a minimum terminal node size (R *minbucket*) in the algorithm, however that does not always solve the overfitting problem.
- Idea 3: Cost complexity pruning
  - most popular
  - penalizes a decision tree for having too many nodes
  - grow the full tree and then reduce the size of the tree (i.e. the number of terminal nodes) by removing sections of the tree that provide little power to classify instances
  - uses a cross-validation approach

Cost complexity pruning

- The idea of cost complexity pruning is to identify a tree of optimal size that has the minimum cost complexity
- Cost complexity of tree T = $CC_\alpha(T) = err(T) + \alpha L(T)$
  - $err(T)$ is the fraction of training records that are missclassified by tree T
  - $L(T)$ is the number of terminal nodes for tree T
  - α is a penalty factor
  - When $\alpha = 0$, there is no penalty for having too many nodes in a tree. In that case, the full-grown tree (having the minimum missclassification rate) is the optimal tree.
  - When $\alpha = \infty$, the root node (null tree) is the optimal tree
  - Calculate cost complexity for a range of values for α and choose the optimal α by cross validation

Cost complexity pruning

- Cost complexity of tree T = $CC_\alpha(T) = err(T) + \alpha L(T)$
- Note: if we prune tree T to get tree $T - T_t$, the number of misclassification naturally increases *i.e.* $err(T - T_t) > err(T)$. Yet we are okay with this because this will control over fitting.
- Also note, the number of terminal nodes of the tree is decreasing after pruning, $L(T - T_t) < L(T)$.
- The idea is to find the subtree that minimizes cost complexity.



Determine α using Cross validation

- Consider a sequence of α's. Let's denote it as $\mathcal{A}$.
- Split the data into K (let's assume $K = 10$) folds
- For $k = 1, 2, \ldots 10$, using every fold except the k^{th}
  - For each α_j in $\mathcal{A}$, construct a sequence of subtrees $T_1, \ldots T_m$
  - Keep the subtree that minimizes $CC_{\alpha_j}(T)$. Let's denote is as T_{α_j}
  - For each tree T_{α_j} find prediction on the k^{th} fold (this is the test data when all folds except the k^{th} fold has been used as training data for constructing the tree).
  - Calculate the corresponding $err(T_{\alpha_j})$.
- For each α_j in $\mathcal{A}$, we now have 10 trees (one from each fold). Denote the tree from the k^{th} fold as $T_{k\alpha_j}$.
- For each α_j in $\mathcal{A}$, calculate the average test error as $\frac{1}{10}\sum_{k=1}^{10} err(T_{\alpha_j})$
- Select the parameter α that minimizes the average test error

Example of classification tree on diabetes data

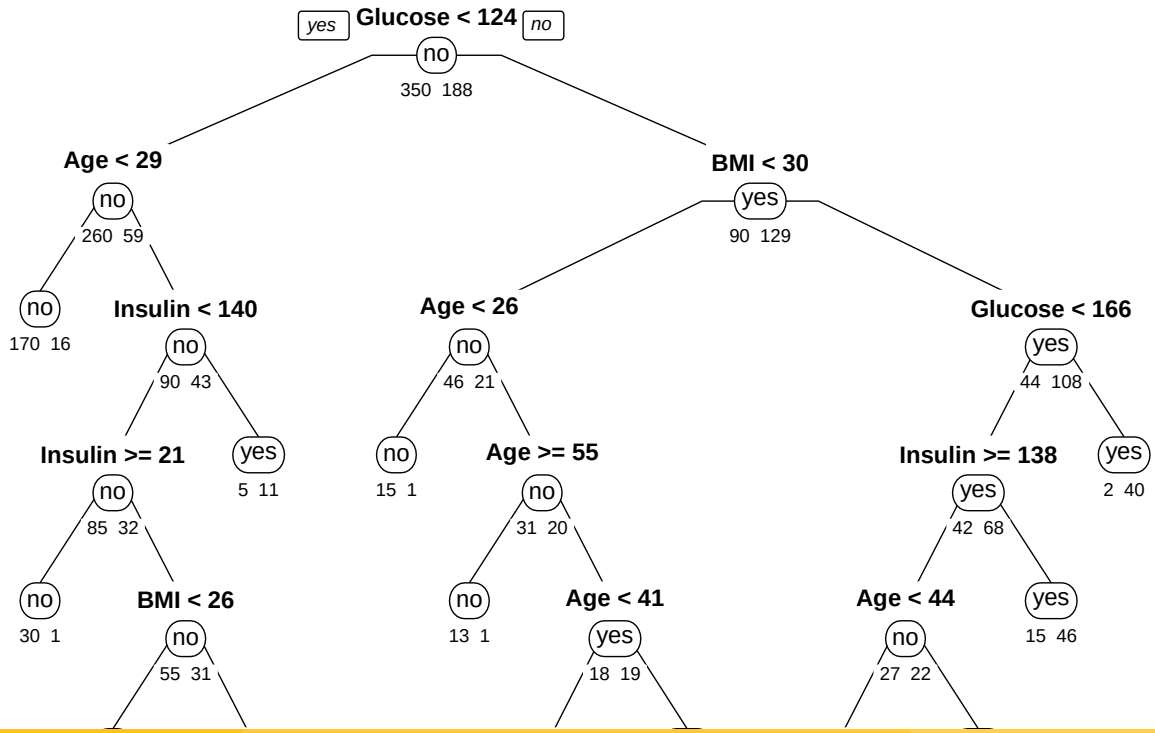
Partition the data into test and training

```
RNGkind (sample.kind = "Rounding")
set.seed(0)
### call the function for creating 70:30 partition
p2 <- partition.2(diabetes, 0.7)
training.data <- p2$data.train
test.data <- p2$data.test
```

Example of classification tree on diabetes data

Construct classification tree on training data

```
library(rpart)
library(rpart.plot)
# fit classification tree on training data
ct1 <- rpart(Outcome ~ . , data = training.data, method = "class",
             minsplit=15, minbucket = 5)
# plot tree
prp(ct1, type = 1, extra = 1, under = TRUE, split.font = 2, varlen = -10)
```



Example of classification tree on diabetes data

Variable importance

```
> ct1$variable.importance
                 Glucose                      Age                  Insulin
               60.205968                32.624821                28.659242
                     BMI            BloodPressure              Pregnancies
               26.256269                14.593970                12.142926
DiabetesPedigreeFunction            SkinThickness
               10.051140                 8.655138
```

- "An overall measure of variable importance is the sum of the goodness of split measures for each split for which it was the primary variable, plus goodness (adjusted agreement) for all splits in which it was a surrogate." (rpart documentation)
- Surrogate variables are used if the primary variable of the split is missing. Surrogates splits produce results similar to original split. Surrogate splits can be found by calling the *summary*() function.
- A variable's importance in the sum of improvement in evaluation measure produced by the nodes in which it appears.

Example of classification tree on diabetes data

Evaluate the performance of the tree on test data

```
# get prediction on the test data
pred.test = predict(ct1, test.data, type = 'class')
# create confusion matrix
library(caret)
> confusionMatrix(pred.test, test.data$Outcome, positive = "yes")
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  120  27
       yes  30  53

               Accuracy : 0.7522
                 95% CI : (0.6912, 0.8066)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 0.0007075

                  Kappa : 0.4585
 Mcnemar's Test P-Value : 0.7910815
            Sensitivity : 0.6625
            Specificity : 0.8000
         Pos Pred Value : 0.6386
         Neg Pred Value : 0.8163
             Prevalence : 0.3478
         Detection Rate : 0.2304
   Detection Prevalence : 0.3609
      Balanced Accuracy : 0.7312

       'Positive' Class : yes
```

Example of classification tree on diabetes data

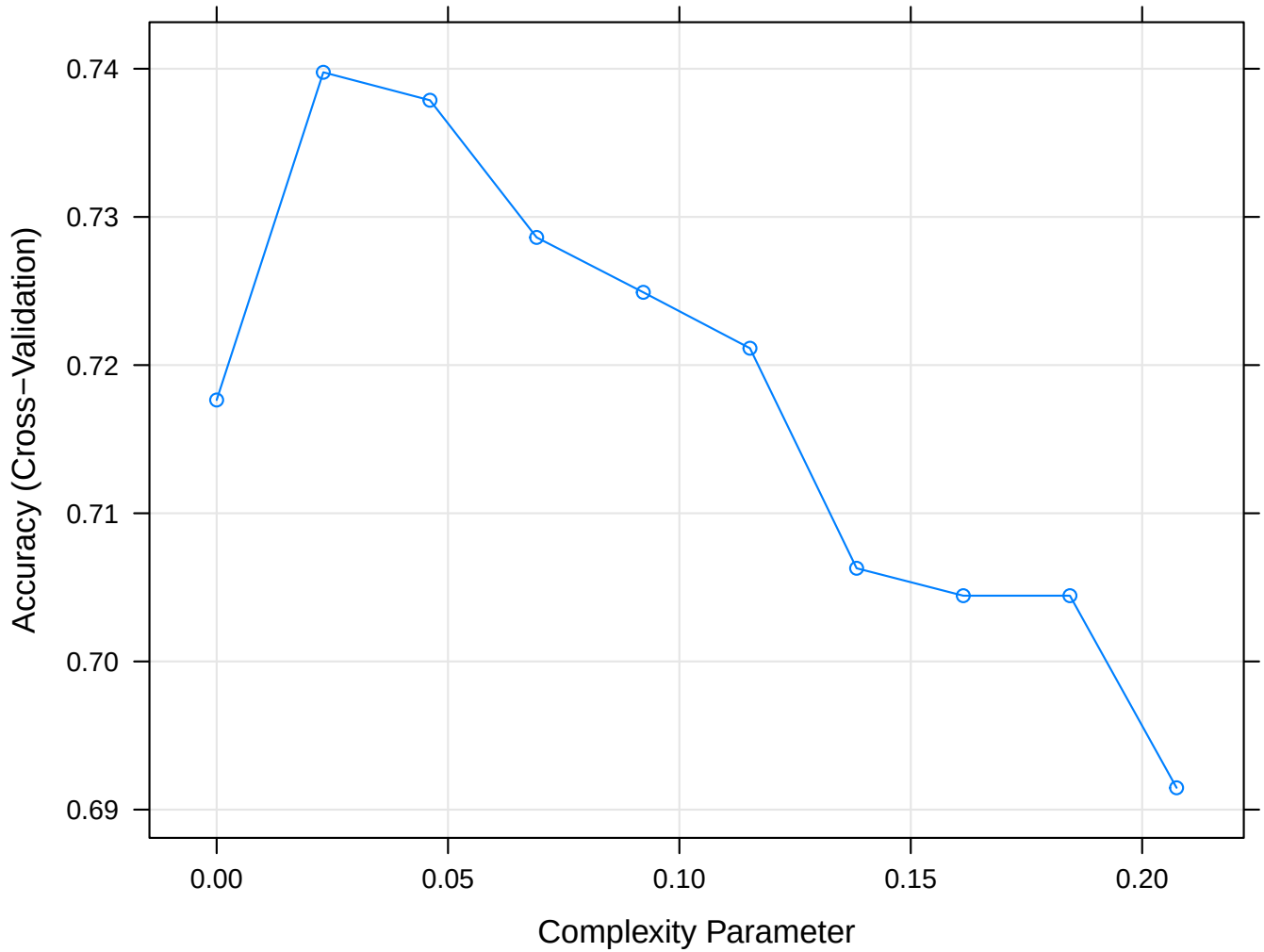
Cost complexity pruning

```
library(caret)
set.seed(0)
train_control <- trainControl(method="cv", number=10)
cv.ct <- train(Outcome ~ . , data = training.data, method = "rpart",
                   trControl = train_control, tuneLength = 10)
> print(cv.ct)
CART
538 samples
  8 predictor
  2 classes: 'no', 'yes'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 484, 484, 484, 484, 484, 485, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.00000000  0.7176450  0.3717968
  0.02304965  0.7397624  0.3779980
  0.04609929  0.7378756  0.3785874
  0.06914894  0.7286164  0.3656797
  0.09219858  0.7249126  0.3616877
  0.11524823  0.7211391  0.3565610
  0.13829787  0.7062893  0.3409367
  0.16134752  0.7044375  0.3407743
  0.18439716  0.7044375  0.3407743
  0.20744681  0.6914745  0.2872593

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.02304965.
```

Example of classification tree on diabetes data

Cross validation plot

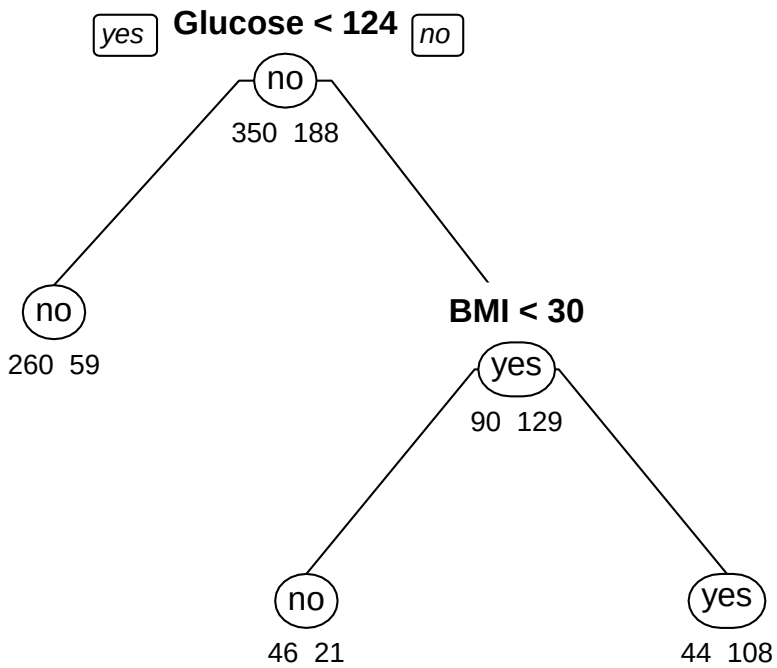
```
plot(cv.ct)
```



Example of classification tree on diabetes data

Pruned tree

```
cv.ct$finalModel
prp(cv.ct$finalModel, type = 1, extra = 1, under = TRUE, split.font = 2, varlen = -10)
```



Example of classification tree on diabetes data

Create classification rule from the tree and predict class membership for the following observation

```
Pregnancies   1
Glucose  99
BloodPressure 70
SkinThickness  16
Insulin   120
BMI  24.1
DiabetesPedigreeFunction 0.439
Age  27
Outcome ?
```

Example of classification tree on diabetes data

Evaluation of pruned tree

```
# get prediction on the test data
pred.test.prune = predict(cv.ct$finalModel, test.data, type = 'class')

# create confusion matrix
> confusionMatrix(pred.test.prune, test.data$Outcome, positive = "yes")
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  124  32
       yes  26  48

               Accuracy : 0.7478
                 95% CI : (0.6865, 0.8026)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 0.001163

                  Kappa : 0.4343
 Mcnemar's Test P-Value : 0.511482
            Sensitivity : 0.6000
            Specificity : 0.8267
         Pos Pred Value : 0.6486
         Neg Pred Value : 0.7949
             Prevalence : 0.3478
         Detection Rate : 0.2087
   Detection Prevalence : 0.3217
      Balanced Accuracy : 0.7133

       'Positive' Class : yes
```

Example of classification tree on diabetes data

Using different cut off value

```
# Predict the raw probability
pred.prob = predict(cv.ct$finalModel, test.data, type = 'prob')
cutoff <- 0.3 # if proportion of occurrences of class "yes" > cutoff then predicted label = "yes"
pred.test.cutoff <- ifelse(pred.prob[,2] >= cutoff, "yes", "no")
> confusionMatrix(as.factor(pred.test.cutoff), as.factor(test.data$Outcome), positive = "yes")
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  106  21
       yes  44  59

               Accuracy : 0.7174
                 95% CI : (0.6545, 0.7746)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 0.021072
                  Kappa : 0.4162
 Mcnemar's Test P-Value : 0.006357
            Sensitivity : 0.7375
            Specificity : 0.7067
         Pos Pred Value : 0.5728
         Neg Pred Value : 0.8346
             Prevalence : 0.3478
         Detection Rate : 0.2565
   Detection Prevalence : 0.4478
      Balanced Accuracy : 0.7221

       'Positive' Class : yes
```

Regression tree

- Tree method can also be used when the response variable Y is numerical.
- How to predict?
  - The predicted value of the response variable at the terminal node ($\hat{y}_i$) is determined by taking the average of the response variable for the training records belonging to the terminal node.
- What impurity measure to use?
  - Use sum of square deviation from the mean of the terminal node *i.e.* $\sum_{i=1}^{n_t}(y_i - \bar{y}_t)^2$ where n_t is the number of records and $\bar{y}_t$ is the mean response in the associated terminal node.
  - Consider the split for which maximum reduction in sum of square deviation is observed.
- How to evaluate performance?
  - Use the root mean square error (RMSE) *i.e.* $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$ from the test data.

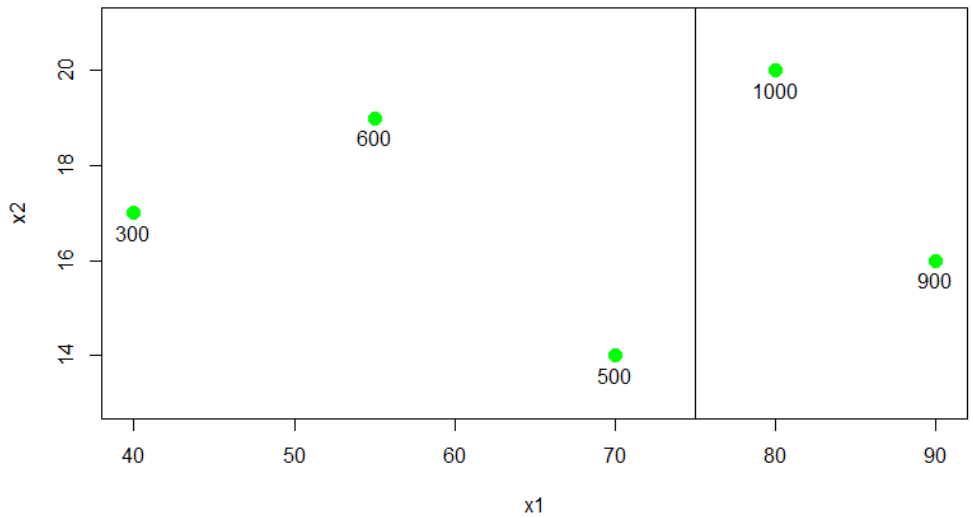
Example of impurity calculation for regression problem

Step 0: No partition

- $\bar{y} = (300 + 600 + 500 + 1000 + 900)/5 = 660$
- $Impurity = (300 - 660)^2 + (600 - 660)^2 + (500 - 660)^2 + (1000 - 660)^2 + (900 - 660)^2) = 332000$

Step 1: Partition at $x1 = 75$

- Prediction at left rectangle = $\bar{y}_{left} = (300 + 600 + 500)/3 = 466.7$
- $Impurity_{left} = (300 - 466.7)^2 + (600 - 466.7)^2 + (500 - 466.7)^2 = 46666.67$
- Prediction at right rectangle = $\bar{y}_{right} = (1000 + 900)/2 = 950$
- $Impurity_{right} = (1000 - 950)^2 + (900 - 950)^2 = 5000$
- For the combined impurity of the two rectangles created by the split, take a weighted average of the left and right impurity
- Combined Impurity = $(3/5)46666.67 + (2/5)5000 = 30000$
- Reduction in impurity due to partition = $332000 - 30000 = 302000$



Example of regression tree on autompg data

```
# use partition.2 function from myfunctions.R and create 60:40 partition
RNGkind (sample.kind = "Rounding")
set.seed(0)
p2 <- partition.2(autompg, 0.6)
mydata.train <- p2$data.train
mydata.test <- p2$data.test
# fit regression tree on training data
ct1 <- rpart(mpg ~ . , data = mydata.train, method = "anova",
             minsplit=15, minbucket = 5)
# plot tree
prp(ct1, type = 1, extra = 1, under = TRUE, split.font = 2, varlen = -10)

# get prediction on the test data
pred.test = predict(ct1, mydata.test)
# get MSE on test data
rmse_test <- sqrt(mean((mydata.test$mpg - pred.test)^2))

# fit regression tree using cost complexity cross validation
library(caret)
set.seed(0)
train_control <- trainControl(method="cv", number=10)
cv.ct <- train(mpg ~ . , data = mydata.train, method = "rpart",
               trControl = train_control, tuneLength = 10)
print(cv.ct)
cv.ct$finalModel
prp(cv.ct$finalModel, type = 1, extra = 1, under = TRUE, split.font = 2, varlen = -10)

# get prediction on the test data
pred.test = predict(cv.ct$finalModel, mydata.test)
# get MSE on test data
rmse_test_prune <- sqrt(mean((mydata.test$mpg - pred.test)^2))
```

Advantages and shortcomings of Decision tree

- Advantage:
  - Easy to understand
  - Useful for variable selection, with the most useful predictors usually showing up at the top of the tree
  - Robust to outliers, since the choice of a split depends on the ordering of a value and not on the absolute magnitudes
  - Assumes no particular relationship between the response and predictor variables
- Shortcoming:
  - Requires a very large number of records to obtain good results
  - Computationally expensive
  - Sensitive to change in data *i.e.* high variance