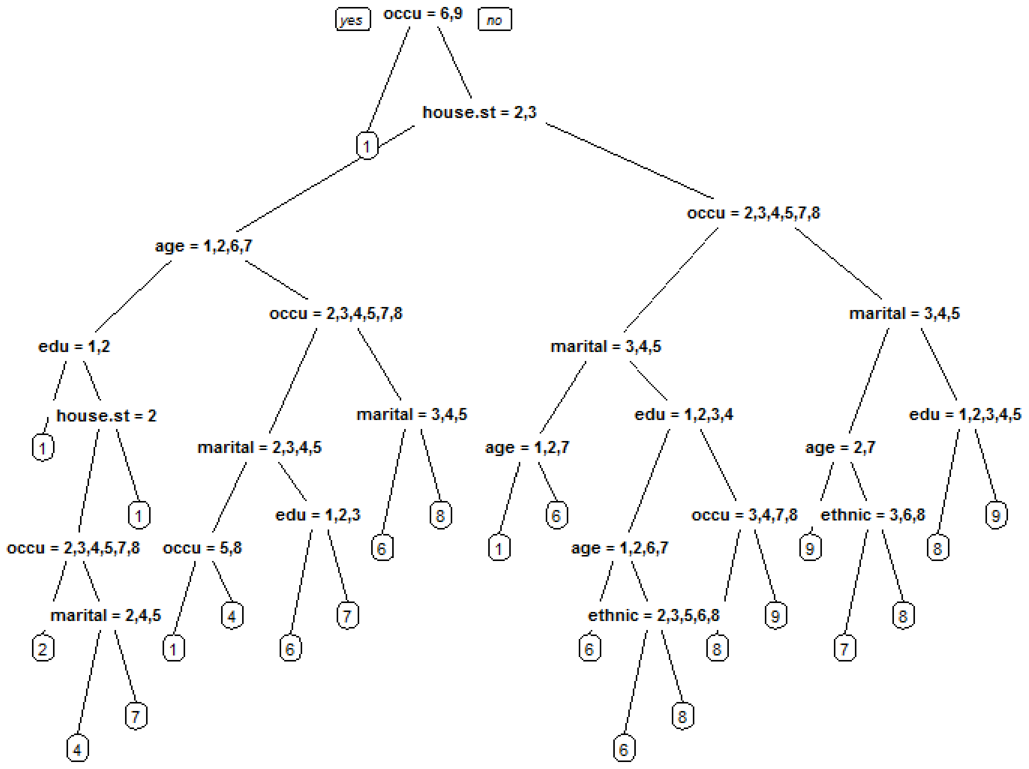
Statistics 315B
Homework 01

Amy Zhang (amyxz@stanford.edu)
Mina Jean Hanna (mjhanna@stanford.edu)
Wei Xia (wei4@stanford.edu)

April 28, 2016

1. Income Tree:



The primary split is on Occupation; being unemployed or a student was very predictive of low income. Other major variables affecting the income were householder status, renting or living with parents/family were also predictive of low income, as well as less education and a younger age.

When cp = 9.6578e-04, the pruned tree is optimal in terms of minimal cross validation error. But the tree has 26 splits, which is not easy to print here. In order to show the tree below, we chose cp = 1.2417e-03, which has 21 splits:

(a) Surrogate splits were used. A surrogate split is an alternative split in the decision tree to be used in the case that the primary split variable is missing. The surrogate is the best mimic/predictor of the primary split because it is correlated with the primary split.

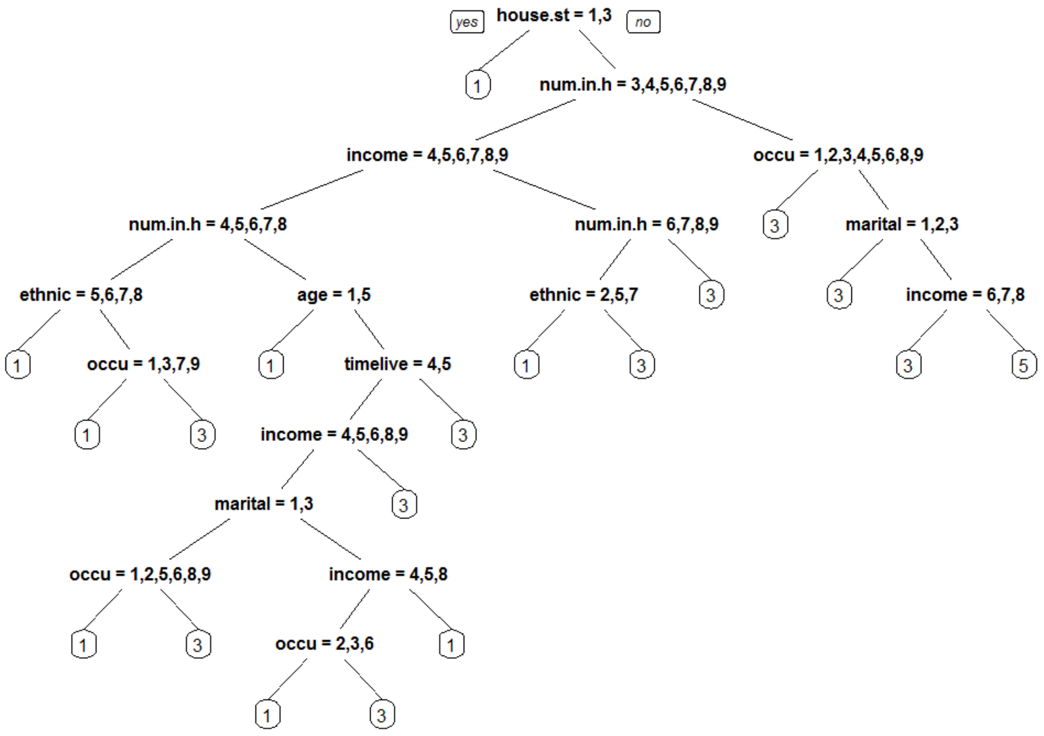
An example is the root node, with a primary split at Occupation, but a surrogate of Age :

```
Node number 1: 8993 observations,    complexity param=0.08498896
  predicted class=1  expected loss=0.8059602  P(node) =1
    class counts:  1745   775   667   813   722  1110   969  1308   884
   probabilities: 0.194 0.086 0.074 0.090 0.080 0.123 0.108 0.145 0.098
  left son=2 (1843 obs) right son=3 (7150 obs)
  Primary splits:
      occu         splits as  RRRRRLRRL, improve=495.5213, (136 missing)
      age          splits as  LRRRRRR,   improve=495.4613, (0 missing)
      edu          splits as  LLRRRR,    improve=400.2626, (86 missing)
      house.status splits as  RRL,       improve=396.2327, (240 missing)
      marital      splits as  RRRRL,     improve=295.3057, (160 missing)
  Surrogate splits:
      age          splits as  LRRRRRR, agree=0.859, adj=0.314, (136 split)
      edu          splits as  LLRRRR,  agree=0.832, adj=0.185, (0 split)
      house.status splits as  RRL,     agree=0.832, adj=0.185, (0 split)
```

(b) In my case, SEX = 1, MARITAL = 1, AGE = 3, EDUCATION = 6, OCCUPATION = 1, HOUSE.ST = 2
and continuning down the tree, it predict 8. 50,000 - 74,9999. (However, this is incorrect.)

```
Here is the related R code:
library(rpart)
library(rpart.plot)

set.seed(100)
df.incomedata = read.csv("Marketing/Income_Data.txt", header = FALSE)

incomedata_names = c("income", "sex", "marital", "age", "edu", "occu",
                     "timelive", "dual_income", "num.in.house", "p.under18",
                     "house.status", "hometype", "ethnic", "language")

names(df.incomedata) = incomedata_names

factors = c(2, 3, 4, 5, 6, 8, 11, 12, 13, 14)
df.incomedata[, factors] = lapply(df.incomedata[, factors], factor)

temp = rpart.control(cp = 0)
fit = rpart(income ~ ., data = df.incomedata,
            method = 'class', control = temp)

printcp(fit)

fit.prune = prune(fit, cp = 1.2417e-03)
rpart.plot(fit.prune)
summary(fit)
```

2. Housetype Tree:



The tree essentially says that if an individual (in the pool of people who answered the questionaire) a property or live with family/parents, the type of home you live in will be a house. Which clearly makes sense. It also shows that individuals and/or families with number of persons in the household less than 3 are likely to be living in an apartment. Older individuals will likely also live in large houses.

Missclassification Error $\left(\frac{1}{N_m}\sum_{i\in R_m} I(y_i \neq k(m))\right)$:

printcp indicates what the root node error and relative error are. The missclassification error is the product of the root node error and relative error:

```
printcp(pruned.housetype)

Classification tree:
rpart(formula = typeofhouse ~ ., data = df.housetypedata, method = "class",
    control = temp)

Variables actually used in tree construction:
[1] age          ethnic       house.status income       marital      num.in.house occu
[8] timelive

Root node error: 3694/9013 = 0.40985

n= 9013

         CP nsplit rel error  xerror     xstd
1 0.3256632      0   1.00000 1.00000 0.012640
2 0.0174607      1   0.67434 0.67515 0.011498
3 0.0078506      3   0.63942 0.64808 0.011351
4 0.0037899      4   0.63156 0.63752 0.011291
```

```
   5 0.0035192      5   0.62777 0.63725 0.011290
   6 0.0032485      7   0.62074 0.63454 0.011274
   7 0.0013535      8   0.61749 0.62832 0.011238
   8 0.0010828     10   0.61478 0.63156 0.011257
   9 0.0010828     18   0.60585 0.63129 0.011255
```

At this cp (cp = 0.0010828), we get the optimal tree, so the
missclassification error = 0.60585 * 0.40985 = 0.2483

```
Here is the related R code:
#problem 2
df.housetypedata = read.csv("Marketing/Housetype_Data.txt", header = FALSE)
housetype_names = c("typeofhouse", "sex", "marital", "age", "edu", "occu",
                    "income", "timelive", "dual_income", "num.in.house",
                    "p.under18", "house.status", "ethnic", "language")
names(df.housetypedata) = housetype_names

factors = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)
df.housetypedata[, factors] = lapply(df.housetypedata[, factors], factor)

temp = rpart.control(cp = 0)
fit.housetype = rpart(typeofhouse ~ ., data = df.housetypedata,
            method = 'class', control = temp)
printcp(fit.housetype)

temp = rpart.control(cp = 1.0828e-03)
pruned.housetype = rpart(typeofhouse ~ ., data = df.housetypedata,
                     method = 'class', control = temp)
rpart.plot(pruned.housetype)

summary(pruned.housetype)
printcp(pruned.housetype)
```

3. The target function is the function that minimizes the risk of incorrect prediction on future data.

   The target function will not be accurate if training data is used to calculate the target function, but then the generating function for data changes when producing future data. Unfortunately this is common in many real life scenarios.

4. The empirical risk would not be accurate if the training data is not a representative sample of the population data.

5. In order to be useful, this search algorithm must be guaranteed to produce a function in a finite amount of time. Thus, the set of all functions that we search over must inherently be smaller than the infinite set of all possible functions.

   In addition, using the class of all possible functions makes the searcher more vulnerable to overfitting the data. If all possible functions are available, and we pick one that minimizes the error on the training set, we may not be choosing one that generalizes well.

6. The prediction risk is comprised of 3 parts: the irreducible error, the square of the bias and the variance.

   When we choose a function with excessively high bias, we underfit the data and do not model the complexity of the process. For example, if using linear regression to model an exponential process, large values of x may be underestimated.

   When we choose a function with excessively high variance, we overfit the data and model non-existing complexities in the data and/or ovefit to noise. For example, if using a 21 degree polynomial to model an linear process with random variance for which we only have 20 datapoints.

   So in order to minimize the expected loss, we will experience the trade-off between bias and variance, in order to minimize the prediction risk.

7. Withdrawn

8. We should not choose a surrogate split first because by definition the primary split is the split that is the most influential split on the data. Surrogate split may be very correlated with primary split, but still it will introduce bad samples to the sub-region, which will dilute the real sample features, thus predicting less accuracy.

9. Proof:
In order to min $\sum_{i=1}^{N}[y_i - F(x_i)]^2 = \sum_{m=1}^{M}\sum_{x_i\in R_m}[y_i - F(x_i)]^2$, let $\frac{\partial \sum_{m=1}^{M}\sum_{x_i\in R_m}[y_i-F(x_i)]^2}{\partial c_m} = 0$

$$\begin{aligned}
\frac{\partial \sum_{m=1}^{M}\sum_{x_i\in R_m}[y_i - F(x_i)]^2}{\partial c_m} &= 0 \\
\sum_{m=1}^{M}\sum_{x_i\in R_m} 2(y_i - F(x_i))(-\frac{\partial F(x_i)}{\partial c_m}) &= 0 \qquad (1) \\
\sum_{m=1}^{M}\sum_{x_i\in R_m} (y_i - F(x_i))(\frac{\partial F(x_i)}{\partial c_m}) &= 0
\end{aligned}$$

As $F(x_i) = c_m I(x_i \in R_m)$, then $\frac{\partial F(x_i)}{c_m} = I(x_i \in R_m)$. Substitute into equation (1):

$$\begin{aligned}
\sum_{m=1}^{M}\sum_{x_i\in R_m} (y_i - c_m I(x_i\in R_m))(I(x_i\in R_m)) &= 0 \\
\sum_{m=1}^{M}(c_m \sum_{x_i\in R_m} I(x_i\in R_m) - \sum_{x_i\in R_m} y_i) &= 0 \qquad (2) \\
c_m &= \frac{\sum_{x_i\in R_m} y_i}{\sum_{x_i\in R_m} I(x_i\in R_m)}
\end{aligned}$$

10.

$$\begin{aligned}
Improvement &= -\left(\sum_{i\in R_l}(y_i-\bar{y_l})^2+\sum_{i\in R_r}(y_i-\bar{y_r})^2-\sum_{i\in R_m}(y_i-\bar{y_n})^2\right)\\
&= -\left(\sum_{i\in R_l}(\bar{y_l}^2-2y_i\bar{y_l})+\sum_{i\in R_r}(\bar{y_i}^2-2y_i\bar{y_r})-\sum_{i\in R_m}(\bar{y_i}^2-2y_i\bar{y_n})\right)
\end{aligned} \tag{3}$$

Because $\sum_{i\in R_l}(\bar{y_l})=n_l\bar{y_l}$ and $\sum_{i\in R_r}(\bar{y_r})=n_r\bar{y_r}$

$\bar{y_n}=\frac{n_l\bar{y_l}+n_r\bar{y_r}}{n}$

$n=n_l+n_r$

Substitute above functions to improvments, then we can have

$$\begin{aligned}
Improvment &= -\left(\sum_{i\in R_l}(\bar{y_l}^2-2y_i\bar{y_l})+\sum_{i\in R_r}(\bar{y_i}^2-2y_i\bar{y_r})-\sum_{i\in R_m}(\bar{y_i}^2-2y_i\bar{y_n})\right)\\
&= n_l\bar{y_l}^2+n_r\bar{y_r}^2-n\bar{y_n}^2\\
&= n_l\bar{y_l}^2+n_r\bar{y_r}^2-(n_l+n_r)(\frac{n_l\bar{y_l}+n_r\bar{y_r}}{n})^2\\
&= n_l\bar{y_l}^2+n_r\bar{y_r}^2-\frac{(n_l\bar{y_l}+n_r\bar{y_r})^2}{n}\\
&= \frac{n_ln_r\bar{y_l}^2+n_ln_r\bar{y_r}^2-2n_ln_r\bar{y_l}\bar{y_r}}{n}\\
&= \frac{n_ln_r(\bar{y_l}-\bar{y_r})^2}{n}
\end{aligned} \tag{4}$$

11. Without loss of generality, assume that we move one observation y_* from R_l to R_r,

$$\begin{aligned}
\bar{y_l}\prime &= \frac{n_l * \bar{y_l} - y_*}{(n_l - 1)} \\
\bar{y_r}\prime &= \frac{n_r * \bar{y_r} + y_*}{(n_r + 1)} \\
Improvement &= \frac{(n_l - 1)(n_r + 1)(\bar{y_l}\prime - \bar{y_r}\prime)^2}{n} - \frac{n_l n_r (\bar{y_l} - \bar{y_r})^2}{n}
\end{aligned} \tag{5}$$

12. Surrogate Splits may not be effective if variables are uncorrelated, and another method may be more effective.

    As compared to Surrogate Splits, treating Missing As Terminal is more efficient, but has higher bias. Missing as Terminal will not continue the tree, and so will not model variation in data missing that value. This may be especially impactful if a variable at the root of the tree is missing, and instead of looking for other ways to predict y, we stop searching. However, if that value is especially predictive of the data and a meaningful prediction can not be made without it, Missing as Terminal may be advantageous. Surrogate splits has the bad effect on the further splits, because the surrogate splits will assign the wrong samples to the wrong sub-region. Missing as Terminal can hold the missing samples and for those un-missing sample, it will make better prediction in the further splits.

    Max Training Data strategy sends the send the observation to the daughter node which contained the most training. The decision is made based on the priori probability of that split and the training data. If the training data has a different distribution of the data in some sub-region, then it will not performs well. As to surrogate splits, the decision is made based on the surrogate split variable. If that variable is not correlated with the primary split, then it will make bad decisions.

13. Disadvantage:

    (a) By including a branch for "missing," we loose the computational advantages of a binary tree.
    (b) With more splits, it will likely to create a bigger tree, which will be more computational.
    (c) With missing value samples being separated out, the number of un-missing samples in one sub-region will be decrease, thus introducing more bias.

    Advantage:

    (a) For samples in each sub-region, they will be more similar. As long as the number of samples is big, it will predict with more accuracy. In surrogate-split strategy, some samples in the same sub-region is chosen by the surrogate split, which is not as accuracy as the primary split.

    This strategy is likely to show a surrogate effect because the child split of the node that represents "missing" is likely to be on a variable correlated with the parent node – it does not, however, guarantee it and there is no mechanism that directly encourages it. A dataset without missing values with this strategy alone would not produce a meaningful tree, but we can try different techniques for dealing with them.

    One idea is to :

    (a) determine the primary split
    (b) pretend that the primary variable does not exist in the dataset
    (c) determine what the primary split would be in the complete absence of that variable
    (d) use the variable from (3) as the child of the "missing" branch

    This would produce an a way to proceed at any variable that is missing, and the next decision would likely be over another similarly meaningful variable.