Predicting Response with Decision Trees

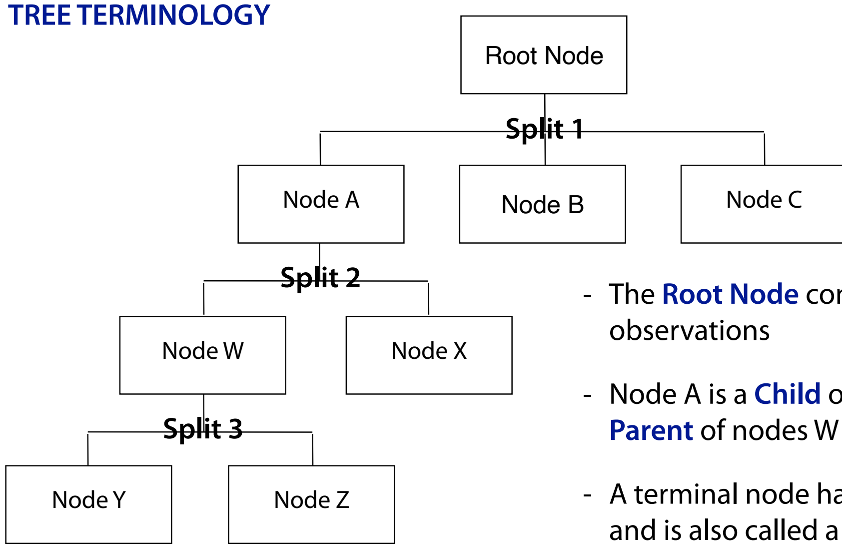
Professor Song Yao
Olin Business School

Customer Analytics

Basic Logic behind Decision Trees

How do we refer to the different parts of a tree?

TREE TERMINOLOGY



- The **Root Node** contains all observations
- Node A is a **Child** of the root and **Parent** of nodes W and X
- A terminal node has no children and is also called a **Leaf**
- There are **five leaves** B, C, X, Y, and Z (i.e., **five segments** of customers)

Fundamental concept of decision trees

CLASSIFICATION TREES

- The dependent variables is **categorical** (non-metric), e.g., buy/not buy

REGRESSION TREES

- The dependent variable is **numeric** (metric) , e.g. how much did a customer spend?

Basic Logic behind Decision Trees

- Each leaf (a segment of customers) in the tree is grown by splitting customers based on their independent variables (e.g., recency, gender, income, …)
- The values of DV of customers of one leaf (e.g., buy or not buy, amount of spending) is different from those of customers in other leaves (as much as possible)

A Toy Example: First set of branches

We use a simple example to show how the CHAID (Chi-squared Automatic Interaction Detection) method works

- Predict response with customer demographics
- Dependent variable: Response -- "response"
- Independent variables:
  - Age -- "age"
  - Income -- "income"
  - Gender -- "female"

Response: DV

```
Response Distribution:
            Count  Percentage (%)
response
0            799            79.9
1            201            20.1
```

Age

```
Age Distribution:
      Count  Percentage (%)
age
1      320            32.0
2      351            35.1
3      329            32.9
```

Income

```
Income Distribution:
          Count  Percentage (%)
income
1          208            20.8
2          487            48.7
3          305            30.5
```

Female

```
Female Distribution:
          Count  Percentage (%)
female
0          517            51.7
1          483            48.3
```

CHAID EXAMPLE: ROOT NODE (TOP LEVEL)

Age variable:

- Cross tab every possible combination of two values of "age" with "response"
- Combine the two values that are the least significantly different from each other
- Stop if all remaining categories are significantly different in predicting response

```
# Compare age groups 1&2
age12 = chaid_demo[chaid_demo['age'].isin([1, 2])]
chi2_age12 = chi2_contingency(pd.crosstab(age12['response'],
                                          age12['age']), correction=False)
print(chi2_age12.pvalue)

# Compare age groups 1&3
age13 = chaid_demo[chaid_demo['age'].isin([1, 3])]
chi2_age13 = chi2_contingency(pd.crosstab(age13['response'],
                                          age13['age']), correction=False)
print(chi2_age13.pvalue)

# Compare age groups 2&3
age23 = chaid_demo[chaid_demo['age'].isin([2, 3])]
chi2_age23 = chi2_contingency(pd.crosstab(age23['response'],
                                          age23['age']), correction=False)
print(chi2_age23.pvalue)
```

```
7.84511863307372e-05
1.842310598122182e-06
0.3548097350546101
```

All possible combinations of different values of “age”: 12, 13, 23

Age 1’s response rates are significantly different from Age 2 and Age 3 (according to Chi2 test).

Treat age=2 and age=3 as equivalent in terms of predicting response
--> **combine them into one category**

CHAID EXAMPLE: ROOT NODE (TOP LEVEL)

Age variable: (take 2)

- Cross tab every possible combination of two values of "age" with “response"
  - Combine the two values that are the least significantly different from each other
    - Age 1 vs. Age 23
- Stop if all remaining categories are significantly different in predicting response

```
# Create new age grouping (1 vs 2&3 combined)
chaid_demo['ageNEW'] = np.where(chaid_demo['age'] == 1, 1, 23)
chi2_ageNEW = chi2_contingency(pd.crosstab(chaid_demo['response'],
                                           chaid_demo['ageNEW']), correction=False)
print(chi2_ageNEW.pvalue)
```

```
1.6626069931911872e-06
```

Age=1 and age=(2 or 3) are significant predictors of response
--> **Stop and go to next variable**

CHAID EXAMPLE: ROOT NODE (TOP LEVEL)

Income variable:

- Continue with "income" and "response": Possible combinations for income, 12, 13, 23
- Combine the two values that are the least significantly different from each other
- Stop if all remaining categories are significantly different in predicting response

```
# Compare income groups 1&2
income12 = chaid_demo[chaid_demo['income'].isin([1, 2])]
chi2_income12 = chi2_contingency(pd.crosstab(income12['response'],
                                            income12['income']), correction=False)
print(chi2_income12.pvalue)

# Compare income groups 1&3
income13 = chaid_demo[chaid_demo['income'].isin([1, 3])]
chi2_income13 = chi2_contingency(pd.crosstab(income13['response'],
                                            income13['income']), correction=False)
print(chi2_income13.pvalue)

# Compare income groups 2&3
income23 = chaid_demo[chaid_demo['income'].isin([2, 3])]
chi2_income23 = chi2_contingency(pd.crosstab(income23['response'],
                                            income23['income']), correction=False)
print(chi2_income23.pvalue)
```

```
0.10568540009415126
0.04793679612791425
1.4713066443738882e-05
```

Treat income=1 and income=2 as equivalent in terms of predicting response (Chi2 test insignificant

--> combine them into 1 category

CHAID EXAMPLE: ROOT NODE (TOP LEVEL)

Income variable: (take 2)

- Cross tab every possible combination of two values of "income" with "response"
  - Combine the two values that are the least significantly different from each other
    - Income 12 and Income 3
- Stop if all remaining categories are significantly different in predicting response

```
# Create new income grouping (1&2 combined vs 3)
chaid_demo['incomeNEW'] = np.where(chaid_demo['income'].isin([1, 2]), 12, 3)
chi2_incomeNEW = chi2_contingency(pd.crosstab(chaid_demo['response'],
                                             chaid_demo['incomeNEW']), correction=False)
print(chi2_incomeNEW.pvalue)
```

```
4.884343487542051e-05
```

income=(1 or 2) and income=3 are significant predictors of response

--> Stop and go to next variable

CHAID EXAMPLE: ROOT NODE (TOP LEVEL)

Female variable:

- Cross tab every possible combination of two values of "female" with "response"
- Combine the two values that are the least significantly different from each other
- Stop if all remaining categories are significantly different in predicting response

```
chi2_female = chi2_contingency(pd.crosstab(chaid_demo['response'],
                                           chaid_demo['female']), correction=False)
print(chi2_female.pvalue)

### based on the p values, female is the most significant variable as it has the lowest p value
### we will use female to split the data this round
```

```
2.0152561677568327e-07
```

female=0 and female=1
are significant predictors
of response
-->
Stop and proceed to selecting the variable to first split the sample

CHAID EXAMPLE: ROOT NODE (TOP LEVEL)

Select variable for first sample partition:

- Cross tab every final variable (after combining categories) with “response”
- Select the variable with the smallest p-value

```
chi2_female = chi2_contingency(pd.crosstab(chaid_demo['response'],
                                           chaid_demo['female']), correction=False)
print(chi2_female.pvalue)

### based on the p values, female is the most significant variable as it has the lowest p value
### we will use female to split the data this round
```

```
2.0152561677568327e-07
```

```
# Create new age grouping (1 vs 2&3 combined)
chaid_demo['ageNEW'] = np.where(chaid_demo['age'] == 1, 1, 23)
chi2_ageNEW = chi2_contingency(pd.crosstab(chaid_demo['response'],
                                           chaid_demo['ageNEW']), correction=False)
print(chi2_ageNEW.pvalue)
```

```
1.6626069931911872e-06
```

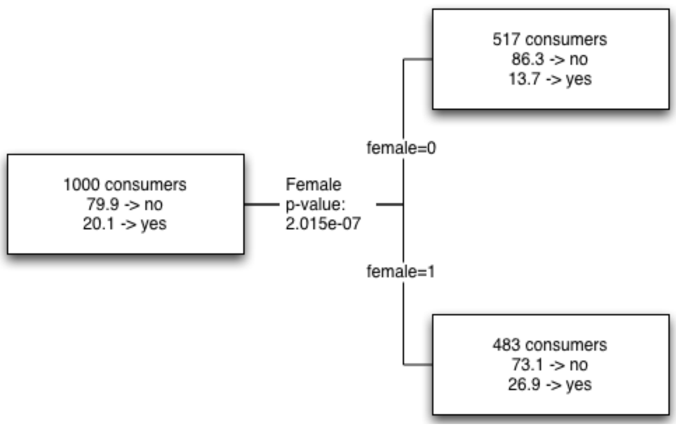
```
# Create new income grouping (1&2 combined vs 3)
chaid_demo['incomeNEW'] = np.where(chaid_demo['income'].isin([1, 2]), 12, 3)
chi2_incomeNEW = chi2_contingency(pd.crosstab(chaid_demo['response'],
                                              chaid_demo['incomeNEW']), correction=False)
print(chi2_incomeNEW.pvalue)
```

```
4.884343487542051e-05
```

Female split has the smallest p-value
Pick gender for sample partition at root note

We have determined that gender is the most important predictor

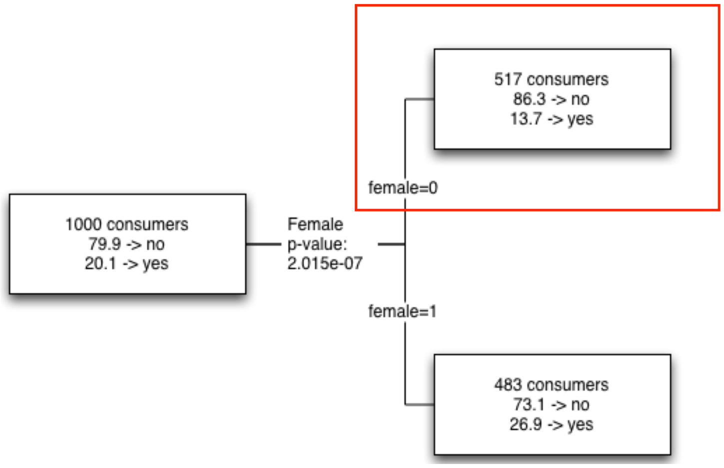
CHAID EXAMPLE: FIRST DATA PARTITION



A Toy Example Continued: Growing the Tree

We proceed with the female=0 (i.e., male) branch

CHAID EXAMPLE: FIRST DATA PARTITION



CHAID EXAMPLE: FIRST CHILD NODE (where FEMALE=0)

Age variable:

- Cross tab every possible combination of two values of "age" with "response"
- Combine the two values that are the least significantly different from each other
- Stop if all remaining categories are significantly different in predicting response

```
# Compare age groups 1&2 for males
male_age12 = chaid_demo[(chaid_demo['age'].isin([1, 2])) &
                        (chaid_demo['female'] == 0)].copy()
chi2_male_age12 = chi2_contingency(pd.crosstab(male_age12['response'],
                                               male_age12['age']), correction=False)
print(chi2_male_age12.pvalue)

# Compare age groups 1&3 for males
male_age13 = chaid_demo[(chaid_demo['age'].isin([1, 3])) &
                        (chaid_demo['female'] == 0)].copy()
chi2_male_age13 = chi2_contingency(pd.crosstab(male_age13['response'],
                                               male_age13['age']), correction=False)
print(chi2_male_age13.pvalue)

# Compare age groups 2&3 for males
male_age23 = chaid_demo[(chaid_demo['age'].isin([2, 3])) &
                        (chaid_demo['female'] == 0)].copy()
chi2_male_age23 = chi2_contingency(pd.crosstab(male_age23['response'],
                                               male_age23['age']), correction=False)
print(chi2_male_age23.pvalue)
```

```
0.36734481098374727
0.2117626995367813
0.7097908825361634
```

Treat age=2 and age=3 as equivalent in terms of predicting response
--> **combine them into 1 category**

CHAID EXAMPLE: FIRST CHILD NODE (FEMALE=0)

Age variable: (take 2)

- Cross tab every possible combination of two values of "age" with "response"
- Combine the two values that are the least significantly different from each other
- Stop if all remaining categories are significantly different in predicting response

```
# Create new age grouping (1 vs 2&3 combined) for males
male_data = chaid_demo[chaid_demo['female'] == 0].copy()
male_data['ageNEW'] = np.where(male_data['age'] == 1, 1, 23)
chi2_male_ageNEW = chi2_contingency(pd.crosstab(male_data['response'],
                                                male_data['ageNEW']), correction=False)
print(chi2_male_ageNEW.pvalue)
```

```
0.22786305389769565
```

Age=1 and age=(2 or 3) are **not** significant predictors of response

--> Combine all age categories, i.e. ignore age as predictor for males!

Stop and go to next variable

CHAID EXAMPLE: FIRST CHILD NODE (FEMALE=0)

Income variable:

- Cross tab every possible combination of two values of "income" with "response"
- Combine the two values that are the least significantly different from each other
- Stop if all remaining categories are significantly different in predicting response

```
# Compare income groups 1&2 for males
male_income12 = chaid_demo[(chaid_demo['income'].isin([1, 2])) &
                           (chaid_demo['female'] == 0)].copy()
chi2_male_income12 = chi2_contingency(pd.crosstab(male_income12['response'],
                                                  male_income12['income']), correction=False)
print(chi2_male_income12.pvalue)

# Compare income groups 1&3 for males
male_income13 = chaid_demo[(chaid_demo['income'].isin([1, 3])) &
                           (chaid_demo['female'] == 0)].copy()
chi2_male_income13 = chi2_contingency(pd.crosstab(male_income13['response'],
                                                  male_income13['income']), correction=False)
print(chi2_male_income13.pvalue)

# Compare income groups 2&3 for males
male_income23 = chaid_demo[(chaid_demo['income'].isin([2, 3])) &
                           (chaid_demo['female'] == 0)].copy()
chi2_male_income23 = chi2_contingency(pd.crosstab(male_income23['response'],
                                                  male_income23['income']), correction=False)
print(chi2_male_income23.pvalue)
```

```
0.933515557879672
0.042925591804953415
0.005081143114016782
```

Treat income=1 and income=2 as equivalent in terms of predicting response

--> **combine them into 1 category**

CHAID EXAMPLE: FIRST CHILD NODE (FEMALE=0)

Income variable: (take 2)

- Cross tab every possible combination of two values of "income" with "response"
- Combine the two values that are the least significantly different from each other
- Stop if all remaining categories are significantly different in predicting response

```
# Create new income grouping (1&2 combined vs 3) for males
male_data = chaid_demo[chaid_demo['female'] == 0].copy()
male_data['incomeNEW'] = np.where(male_data['income'].isin([1, 2]), 12, 3)
chi2_male_incomeNEW = chi2_contingency(pd.crosstab(male_data['response'],
                                                    male_data['incomeNEW']), correction=False)
print(chi2_male_incomeNEW.pvalue)
```

```
0.0026573404504472826
```

income=(1 or 2) and income=3 are significant predictors of response

--> **Stop and proceed to choose which variable to split the sample next**

CHAID EXAMPLE: FIRST CHILD NODE (FEMALE=0)

Select variable for second sample partition (in the female=0 branch):

- Cross tab every final variable (after combining categories) with “response”
- Select the variable with the smallest p-value


```
# Create new income grouping (1&2 combined vs 3) for males
male_data = chaid_demo[chaid_demo['female'] == 0].copy()
male_data['incomeNEW'] = np.where(male_data['income'].isin([1, 2]), 12, 3)
chi2_male_incomeNEW = chi2_contingency(pd.crosstab(male_data['response'],
                                                    male_data['incomeNEW']), correction=False)
print(chi2_male_incomeNEW.pvalue)
```

```
0.0026573404504472826
```


```
# Create new age grouping (1 vs 2&3 combined) for males
male_data = chaid_demo[chaid_demo['female'] == 0].copy()
male_data['ageNEW'] = np.where(male_data['age'] == 1, 1, 23)
chi2_male_ageNEW = chi2_contingency(pd.crosstab(male_data['response'],
                                                 male_data['ageNEW']), correction=False)
print(chi2_male_ageNEW.pvalue)
```

```
0.22786305389769565
```

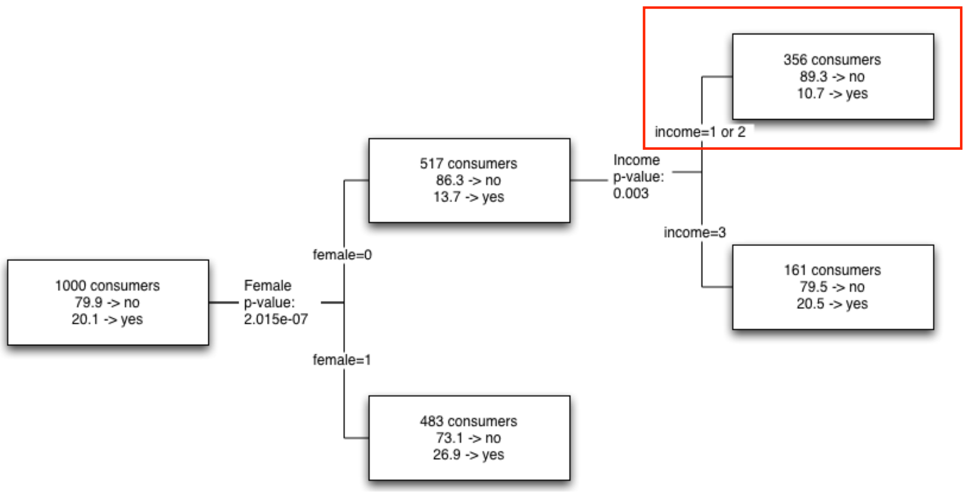
Smallest p-value

-->

pick income for sample partition in the female=0 branch

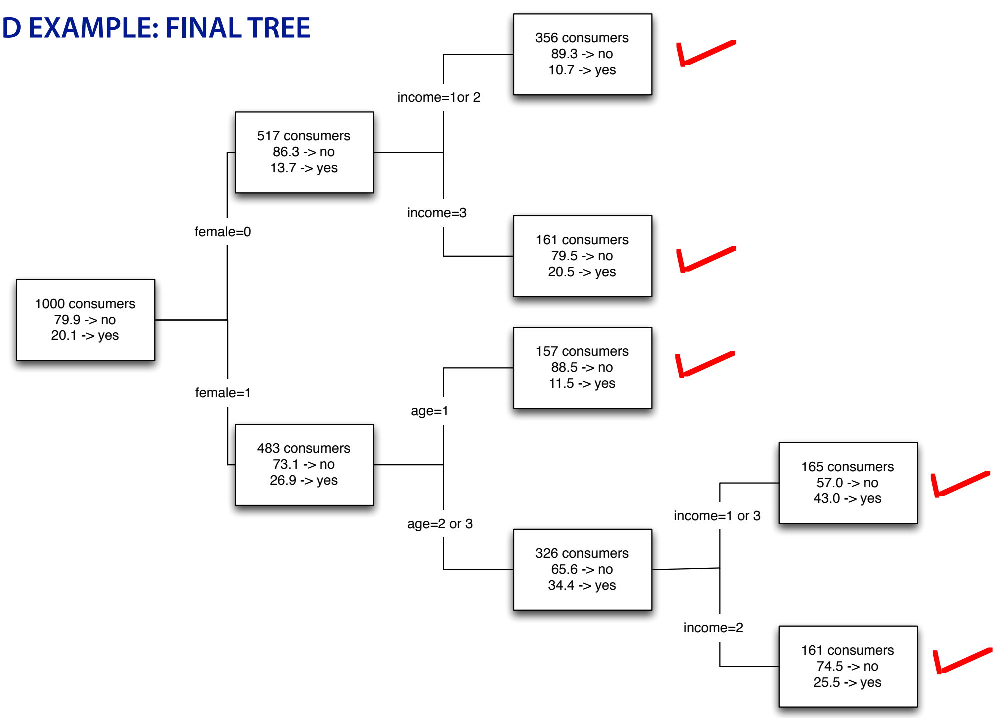
We have determined that income is the most important predictor for men

CHAID EXAMPLE: THIRD DATA PARTITION



One more split completes the tree

CHAID EXAMPLE: FINAL TREE



How do we use Decision Trees to make decision

Decision trees are built through recursive partitioning

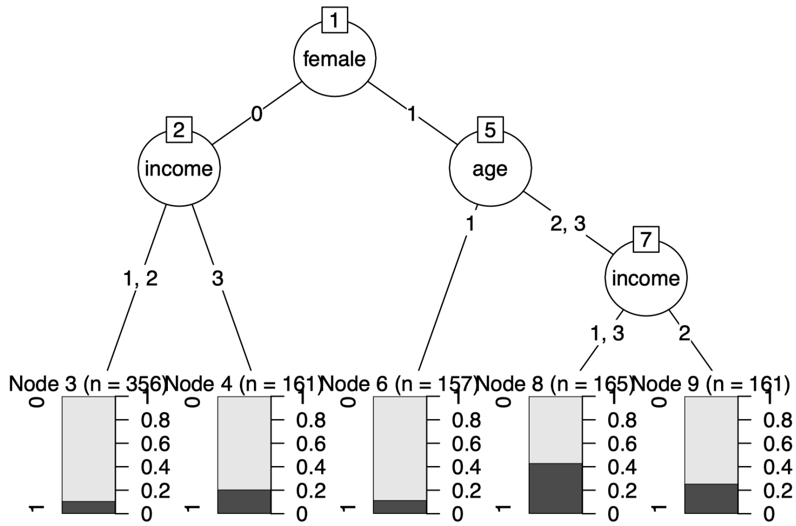
IDEA OF DECISION TREES ALGORITHMS

- Recursive partitioning = an iterative process of splitting the data into partitions and them splitting each partition into a sub-partitions
- Partition = MECE (**M**utually **E**xclusive and **C**ollectively **E**xhaustive)
- Start with Root node (includes all observations in the training sample)
- Each branch finds a single variable to split the data in 2 or more groups
  - Algorithm tries to break up the data, often using every possible split on every variable (brute force --> may require huge computing power)
    - For binary tree, suppose ages run 18-94 and consider splitting on ≤18/>18 versus ≤19/>19 versus ≤20/>20 versus...versus ≤93/>93
    - The best split is the one which best separates groups as measured by the dependent variable
- We focused on CHAID as an example. There are many different implementations of this basic algorithm (e.g., CHAID, CART, C4.5, etc.).

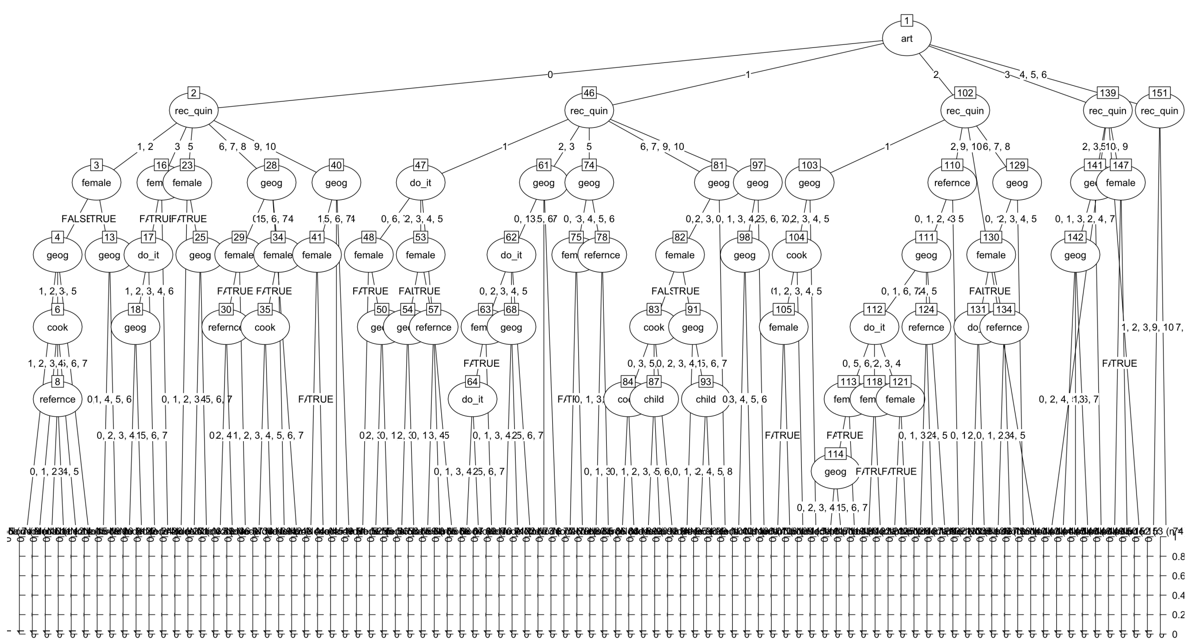
R and Python both have functions for CHAID, but they are somewhat cumbersome (particularly Python)

CHAID EXAMPLE: FINAL TREE

```r
chaid_demo_model <- chaid(response ~ age + female + income, data = chaid_demo)
plot(chaid_demo_model)
```



CHAID TREE FOR BOOKBINDERS BUYERS



The break-even response rate tells us to which cells to extend the offer

BREAK EVEN RESPONSE RATE

- Cost of mailing an offer = \$0.50
- Selling price (includes shipping) = \$18
- Wholesale price paid by Bookbinders = \$9
- Shipping costs = \$3
- Break-even = Cost to mail/net revenue per sale = .5/(18-9-3) = **8.3%**
- **Depending which leaves (segments) have response rates greater than 8.3%, we decide how to target.**

We can use decision tree based analysis for a range of purposes

USES OF DECISION TREES

- **Prediction/classification:**
  Like logistic regression, predict values of a target variable
- **Segmentation:**
  Identify relatively homogeneous groups
- **Interaction identification:**
  Identify relationships that pertain only to specific subgroups, for example for use in a logistic regression model

Alternative Decision Tree Method to CHAID

CART, Classification and Regression Tree

- **Classification tree:**
  Instead of using Chi2, Gini-index is used for splitting branches
  - Gini-index: Another metric for the difference of DV across nodes
- **Regression tree:**
  ANOVA is one example metric that can be used for splitting branches
- **K-fold cross validation automatically**

Decision trees are simple to understand and implement

ADVANTAGES OF DECISION TREES

- Actionable -- generates a set of simple rules which can be used to classify/ predict new cases (e.g. if age<25 and purchase at least 1 art book last year, will purchase "The Art History of Florence")
  - Easy to encode rules in decision systems
  - Fast to classify and predict new cases
- Provides a clear indication which variables are most important for prediction or classification

Random Forest

Decision trees also have some serious disadvantages

DISADVANTAGES OF DECISION TREES

- 'Lose' information compared to regression models b/c of categorization of continuous variables:
  - In regression model, the prediction will be different for each value of a continuous predictor, e.g. 1014 vs. 1016 vs. 2006 vs. 2100
  - In decision tree, 1014 and 1016 are likely to fall into the same node and thus have the same prediction.
- Trees can be too large to properly interpret
- Can be error-prone if the number or observations per class gets small
- Can fit well training sample but badly in test sample -- <u>overfitting problem</u>

“Ensembles” of trees produce better predictions and are less prone to overfitting

ENSEMBLE APPROACHES

- Random Forests
- Boosted Decision Trees

CORE IDEA

- Each tree *may be* a weak predictor
- Create many decision trees
  - using subsets of the original data and subsets of the independent variables (Random Forests)
  - weighting original data differently (Boosted Decision Trees)
- Go with the (weighted) average of the prediction of all the decision trees

“Random Forest” is the most popular type of decision tree ensemble

RANDOM FOREST IDEA

- Decision trees algorithm that injects randomness into fitting (invented by Leo Breiman)
- This randomness reduces overfitting
- Key idea is to create many decision trees (~500), each of them based on
  - randomly chosen subsample of the data
  - randomly chosen subset of the predictor variables at each node that is considered
- Very accurate predictor, can handle huge number of input variables

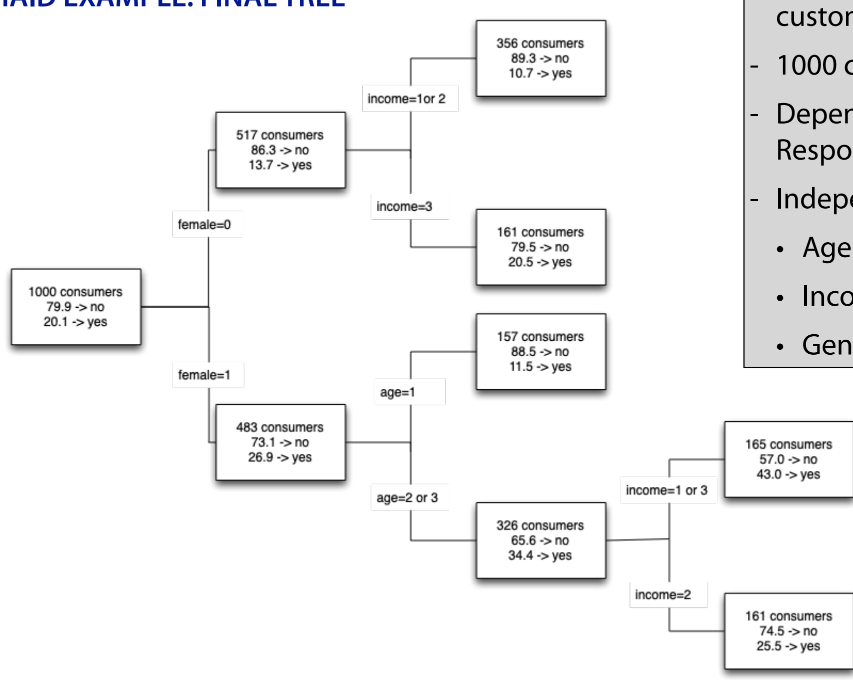
The random forest algorithm builds hundreds of trees

RANDOM FOREST ALGORITHM

- **For each tree:**
  - **Create a "Bootstrap Sample"**
    - Let's say the original sample has N observations
    - Randomly draw observations from the original data (with replacement) and form a new data of N observations
      - Some observations may appear multiple times; some may never appear
  - **Sample the variables when splitting**
    - When building a branch, randomly select a subset of variables for the split instead of considering all available variables
    - The number of variables in the subset is typically the square root of the number of independent variables (e.g. 50 variables, randomly pick 7)
    - At next node, randomly select another subset of variables
- **"Ensemble Scoring"**
  - Combine the results from all the trees by averaging the prediction of each tree

Recall the CHAID example

CHAID EXAMPLE: FINAL TREE



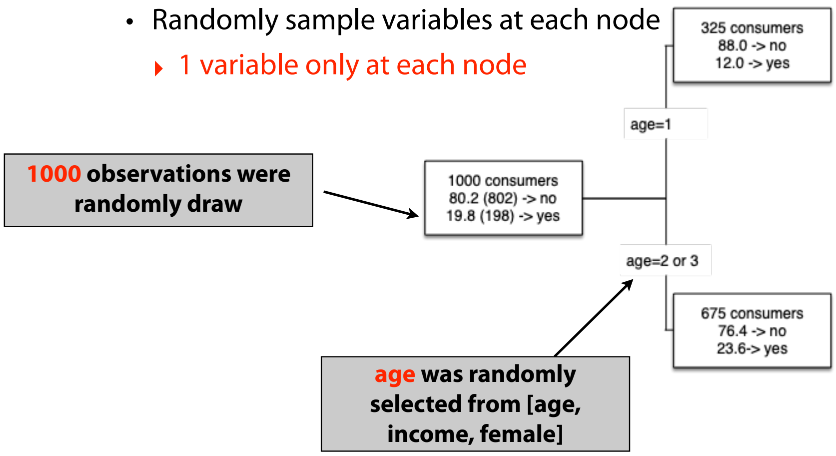
- Predict response with customer demographics
- 1000 consumers
- Dependent variable: Response -- "response"
- Independent variables:
  - Age -- "age"
  - Income -- "income"
  - Gender -- "female"

Lets predict using Random Forest

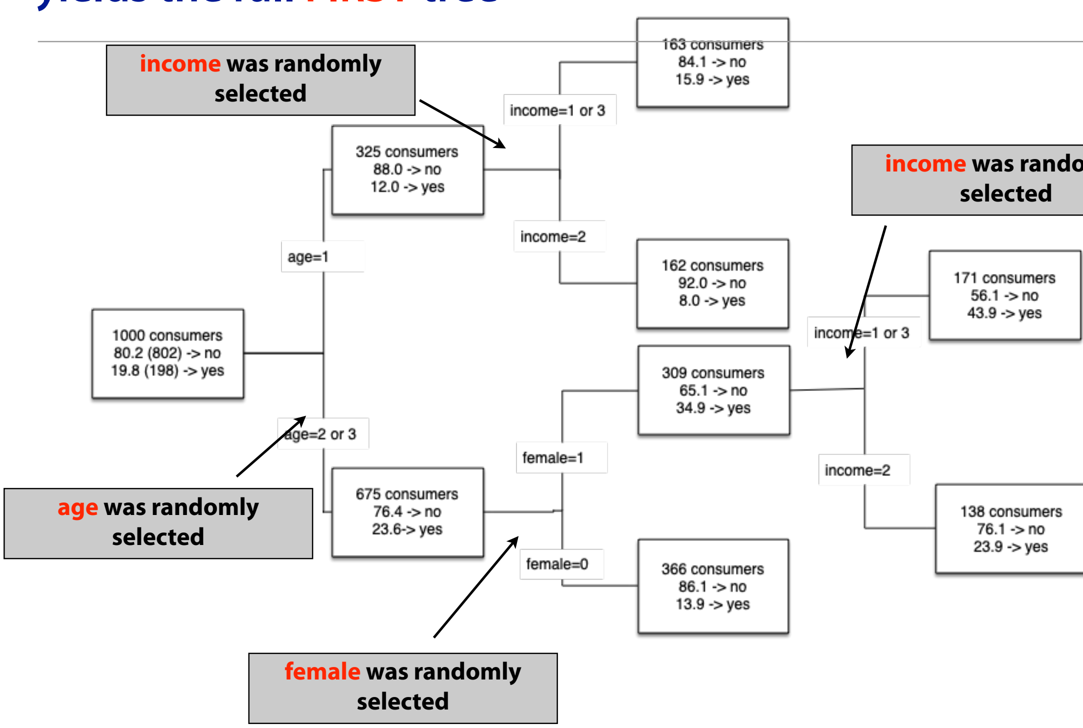
- Predict response with customer demographics
- 1000 consumers
- Dependent variable: Response -- “response”
- Independent variables:
  - Age -- “age”
  - Income -- “income”
  - Gender -- “female”

RANDOM FOREST EXAMPLE

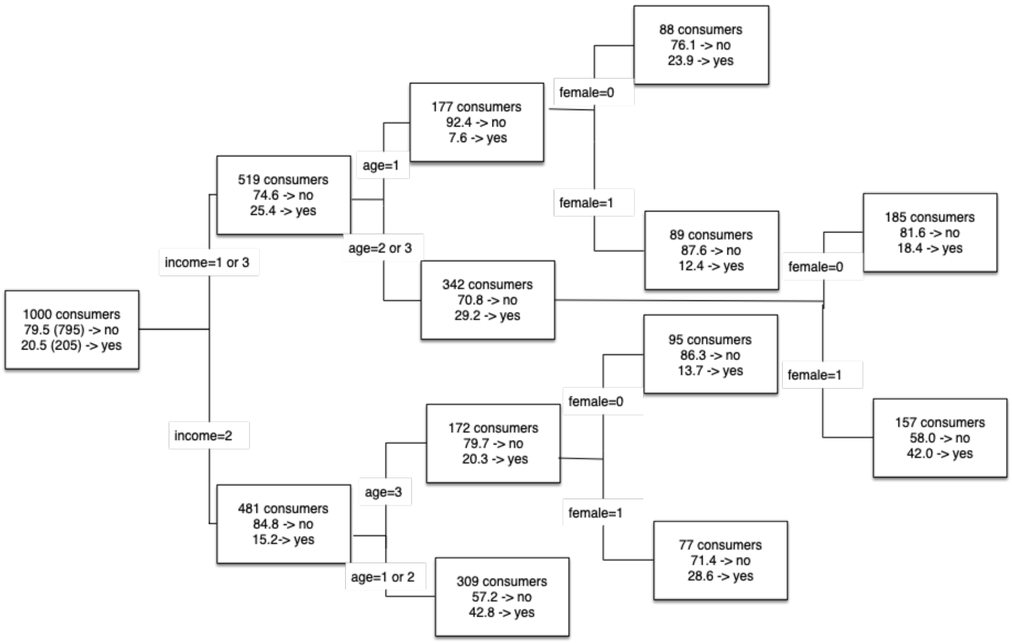
- For each tree:
  - “Bootstrap” observations
    - Draw 1000 observations from the original dataset (with replacement)
  - Randomly sample variables at each node
    - 1 variable only at each node



Repeating the random selection of variables at each node yields the full FIRST tree

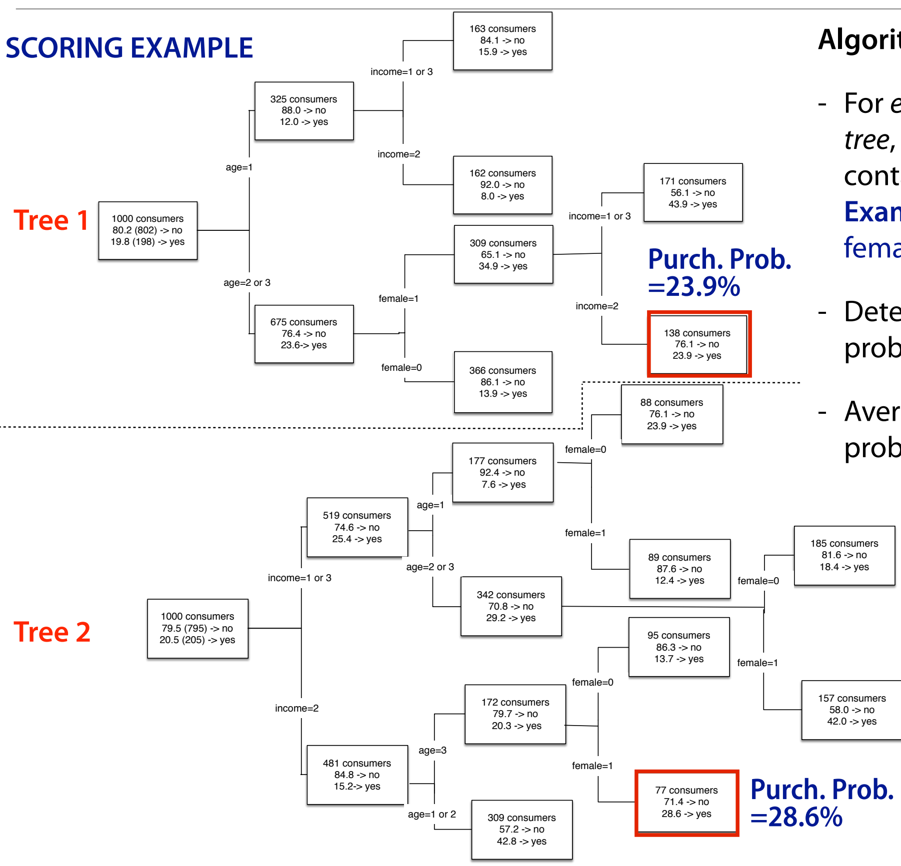


Resampling observations and *resampling* variables at each node yields the full SECOND tree



We can now "score the ensemble" of trees

SCORING EXAMPLE



Algorithm

- For *each customer* and *each tree*, determine the leaf that contains that customer
  Example: A customer female=1, age=3, income=2
- Determine the purchase probability for each leaf
- Average the purchase probabilities over the trees
  - **Predicted Purchase Probability of This Customer**
    = (23.9%+28.6%)/2
    = 26.25%

In practice we would construct and ensemble score hundreds of trees

- Instead of just having two trees, build, say, 500 of them.
  - Given a customer's age, income, and gender, the customer will be grouped into a particular leaf at each respective tree.
  - Each leaf has its different response rate prediction for that customer (e.g., 23.9% vs. 28.6%)
  - Each customer has 500 response rates from the 500 leaves
  - The average of these 500 response rates is the predicted response probability of the customer

Random Forests: Advantages and Disadvantages

ADVANTAGES

- Very accurate predictor
  - Random forests algorithm is less prone to overfitting (not completely)
    - Because of the randomness, observations and variables that may cause overfitting are always dropped at some point
  - With a single decision tree, when alternative variables have similar prediction power, it may be difficult to decide how to build a node
    - Random Forest allows such variables all contribute to the prediction because we randomly select variables at each node (each one of them has the chance to be picked)
- Can handle huge number of input variables
  - Only use a subset of variables at each node

DISADVANTAGES

- Harder to interpret the results than single decision tree
- Time consuming
  - Hundreds of trees

Many models—need to tune and compare carefully

Firewall Example: Model Comparison on the Test Sample (*Code on Canvas for your ref.*)