Data Science

Week 7

Fitting a Model to Data (1) - Decision Tree

Outline

1. Models and Prediction
2. Supervised Segmentation - Information Gain
3. Supervised Segmentation - Decision Tree
4. Probability Estimation
5. Example: Addressing the Churn Problem with Tree Induction

Example in Week 2: Predicting Customer Churn

- MegaTelCo, a largest telecommunication firm, has a major problem with customer **retention** in the wireless business.
- 20% of cell phone customers leave when their contracts expire, and it is getting increasingly difficult to acquire new customers. Customers switching from one company to another is called *churn*.
- You have been called in to help understand the problem and to devise a solution. Attracting new customers is much more expensive than retaining existing ones, so a good deal of marketing budget is allocated to prevent churn. Marketing has already designed a special retention offer.

Your task is to devise a precise, step-by-step plan for how the data science team should use MegaTelCo's vast data resources to decide which customers should be offered the special retention deal prior to the expiration of their contracts.

Converting “Predicting Customer Churn” to *Supervised* Segmentation

- Following our example of data mining for churn, we will begin by thinking of predictive modeling as *supervised* segmentation—how can we segment the population into groups that differ from each other with respect to some quantity of interest.
- In particular, how can we segment the population with respect to something that we would like to predict or estimate.
- The target of this prediction can be something we would like to avoid (negative light), such as ***which customers are likely to leave the company when their contracts expire***, which accounts have been defrauded, or which web pages contain objectionable content.

- The target might instead be cast in a positive light, such as which consumers are most likely to respond to an advertisement or special offer.

Attributes and Target

- One of the fundamental ideas of data mining: finding or selecting important, informative variables or "attributes" of the entities described by the data. For example, the age of customer maybe an attribute to predict customer churn.
- A key to supervised data mining is that we have some target quantity we would like to predict or to otherwise understand better. Often this quantity is unknown or unknowable at the time we would like to make a business decision, such as whether a customer will churn soon after her contract expires.
- Having a target variable crystalizes our notion of finding informative attributes: is there one or more other variables that reduces our uncertainty about the value of the target?
- This also gives a common analytics application of the general notion of correlation discussed above: we would like to find knowable attributes that correlate with the target of interest -that reduce our uncertainty in it.

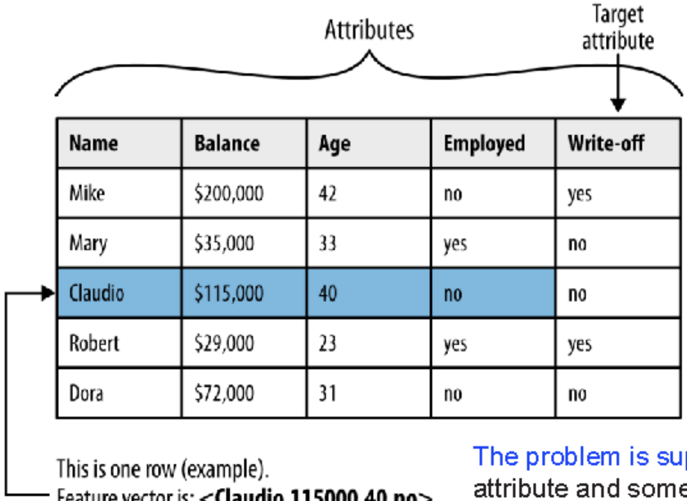
Terminology of Predictive Modeling

- Supervised learning is model creation where the model describes a relationship between a set of selected variables (attributes or features) and a predefined variable called the target variable.

- The model estimates the value of the target variable as a function (possibly a probabilistic function) of the features.

- For our churn-prediction problem we would like to build a model of the propensity to churn as a function of customer account attributes, such as age, income, length with the company, number of calls to customer service, overage charges, customer demographics, data usage, and others.

Example Problem of Credit Write-off Prediction


Attributes
Target attribute

Name	Balance	Age	Employed	Write-off
Mike	$200,000	42	no	yes
Mary	$35,000	33	yes	no
Claudio	$115,000	40	no	no
Robert	$29,000	23	yes	yes
Dora	$72,000	31	no	no

This is one row (example).
Feature vector is: **<Claudio,115000,40,no>**
Class label (value of Target attribute) is **no**

- An instance or example represents a fact or a data point—in this case a historical customer who had been given credit.
- An instance is described by a set of attributes (fields, columns, variables, or features).
- An instance is also sometimes called a feature vector, because it can be represented as a fixed-length ordered collection (vector) of feature values.

The problem is supervised because it has a target attribute and some “training” data where we know the value for the target attribute. It is a classification (rather than regression) problem because the target is a category (yes or no) rather than a number.

Outline

1. Models and Prediction
2. Supervised Segmentation - Information Gain
3. Supervised Segmentation - Decision Tree
4. Probability Estimation
5. Example: Addressing the Churn Problem with Tree Induction

Supervised Segmentation

- Often we are interested in applying data mining when we have many attributes, and are not sure exactly what the segments should be.
- In our churn-prediction problem, who is to say what are the best segments for predicting the propensity to churn?
- If there exist in the data segments (on some attributes) with significantly different values for the target variable, we would like to be able to extract them (segments) automatically.

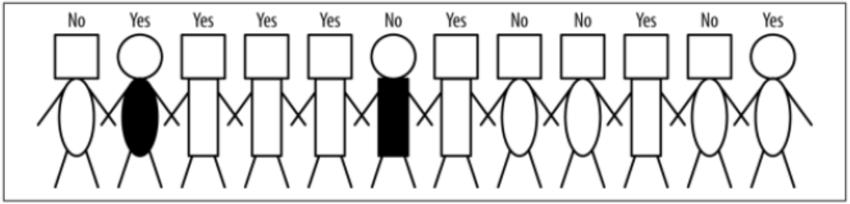
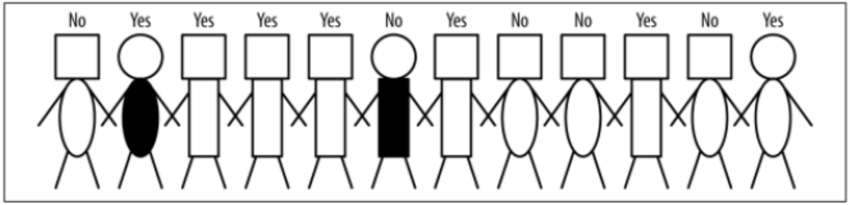


Figure 3-2. A set of people to be classified. The label over each head represents the value of the target variable (write-off or not). Colors and shapes represent different predictor attributes.

Informative Attributes for Supervised Segmentation

- This brings us to our fundamental concept: how can we judge whether a variable (attribute) contains important information about the target variable? How much?
- We would like automatically to get a selection of the more informative variables with respect to the particular task at hand (namely, predicting the value of the target variable).
- Let's consider a binary (two class) classification problem, and think about what we would like to get out of it.
- To be concrete, the figure in the last page shows a simple segmentation problem: twelve people represented as stick figures.

Example for Selecting Informative Attributes -1



- There are two types of heads: square and circular; and two types of bodies: rectangular and oval; and two of the people have gray bodies while the rest are white. These are the attributes we will use to describe the people.
- Above each person is the binary target label, *Yes* or *No*, indicating (for example) whether the person becomes a loan write-off.

Example for Selecting Informative Attributes -2

- Attributes:
  - head-shape: square, circular
  - body-shape: rectangular, oval
  - body-color: gray, white
- Target variable:
  - write-off: Yes, No

→

which of the attributes would be best to segment these people into groups, in a way that will distinguish write-offs from non-write-offs?

Entropy

- For classification problems we can address all the issues by creating a formula that evaluates how well each attribute splits a set of examples into segments, with respect to a chosen target variable. Such a formula is based on a *purity measure.*
- The most common splitting criterion is called *information gain*, and it is based on a purity measure called *entropy*. Both concepts were invented by one of the pioneers of information theory, Claude Shannon (Shannon, 1948).
- Entropy is a measure of disorder that can be applied to a set, such as one of our individual segments. Consider that we have a set of *properties* of members of the set, and each member has one and only one of the properties. In supervised segmentation, the member properties will correspond to the values of the target variable.
- Disorder corresponds to how mixed (impure) the segment is with respect to these properties of interest. So, for example, a mixed up segment with lots of write-offs and lots of non-write-offs would have high entropy.

Formula of Entropy

Entropy measures the amount of information in a random variable

Each p_i is the probability (the relative percentage) of property i within the set, ranging from $p_i = 1$ when all members of the set have property i, and $p_i = 0$ when no members of the set have property i.

for binary classification

$$H(X) = -p_0 \log_2 p_0 - p_1 \log_2 p_1 \,,\ X = \{1, 0\}$$

for classification in c classes

$$H(X) = -\sum_{i=1}^{c} p_i \log_2 p_i = \sum_{i=1}^{c} p_i \log_2 1/p_i \,,\ X = \{1, \ldots, c\}$$

Example:

$$H(X) = -\sum_{i=1}^{8} 1/8 \log_2 1/8 = -\log_2 1/8 = \log_2 8 = 3$$

Calculation of the Entropy

- As a concrete example, consider a set *S* of 10 people with seven of the *non-write-off* class and three of the *write-off* class. So:

- p(non-write-off) = 7 / 10 = 0.7
  p(write-off) = 3 / 10 = 0.3

- entropy(S) $= - 0.7 \times \log_2 (0.7) - 0.3 \times \log_2 (0.3)$
  $\approx - 0.7 \times - 0.51 - 0.3 \times - 1.74$
  ≈ 0.88

A Plot of the Entropy

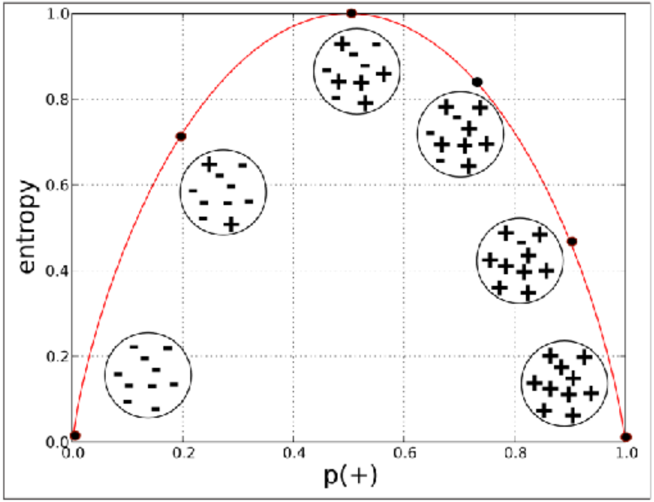


Figure 3-3. Entropy of a two-class set as a function of p(+).

- Figure shows a plot of the entropy of a set containing 10 instances of two classes, + and –.

- We can see then that entropy measures the general disorder of the set, ranging from zero at minimum disorder (the set has members all with the same, single property) to one at maximal disorder (the properties are equally mixed).

Information Gain (IG)

- We would like to measure how informative an attribute is with respect to our target: how much gain in information it gives us about the value of the target variable.
- An attribute segments a set of instances into several subsets. Entropy only tells us how impure one individual subset is.
- Fortunately, with entropy to measure how disordered any set is, we can define information gain (IG) to measure how much an attribute improves (decreases) entropy over the whole segmentation it creates.

Formula of IG

- Information gain measures the *change in entropy* due to any amount of new information being added; here, in the context of supervised segmentation, we consider the information gained by splitting the set on all values of a single attribute.
- Let's say the attribute we split on has *k* different values. Let's call the original set of examples the *parent* set, and the result of splitting on the attribute values the *k* *children* sets.
- Information gain is a function of both a parent set and of the children resulting from some partitioning of the parent set—how much information has this attribute provided?

$$IG(parent, children) = entropy(parent) - [p(c_1) \times entropy(c_1) + p(c_2) \times entropy(c_2) + \cdots + p(c_k) \times entropy(c_k)]$$

Calculation of the IG

- Notably, the entropy for each child (c_i) is weighted by the proportion of instances belonging to that child, $p(c_i)$.
- This addresses directly our concern from above that splitting off a single example, and noticing that that set is pure, may not be as good as splitting the parent set into two nice large, relatively pure subsets, even if neither is pure.
- As an example, consider the split in Figure. This is a two-class problem (• and ★). Examining the figure, the children sets certainly seem "purer" than the parent set. The parent set has 30 instances consisting of 16 dots and 14 stars, so:

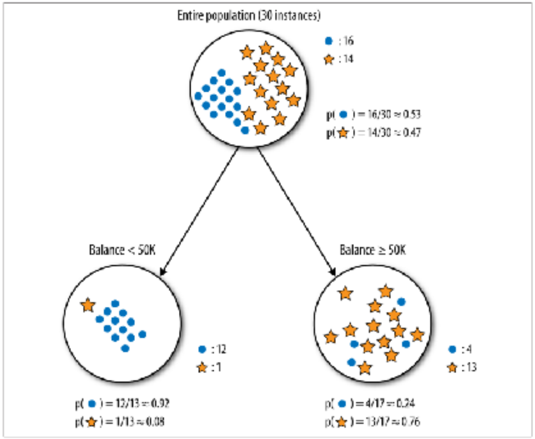


Figure 3-4. Splitting the "write-off" sample into two segments, based on splitting the Balance attribute (account balance) at 50K.

Calculation of the IG

entropy(parent) = - p(•) × $\log_2$ p(•) - p(☆) × $\log_2$ p(☆)

≈ - 0.53 ×(- 0.9) - 0.47 × (- 1.1) ≈ 0.99 (very impure)

The entropy of the left child is:

entropy(Balance < 50K) = - p(•) × $\log_2$ p(•) - p(☆) × $\log_2$ p(☆)

≈ - 0.92 × (- 0.12) - 0.08 × (- 3.7) ≈ 0.39

The entropy of the right child is:

entropy(Balance ≥ 50K) = - p(•) × $\log_2$ p(•) - p(☆) × $\log_2$ p(☆)

≈ - 0.24 × (- 2.1) - 0.76 × (- 0.39) ≈ 0.79

IG = entropy(parent) - p(Balance < 50K) × entropy(Balance < 50K)

- p(Balance ≥ 50K) × entropy(Balance ≥ 50K)

≈ 0.99 - 13/30 × 0.39 - 17/30 × 0.79 ≈ 0.37

This split (Balance ≥ 50K or <50K)reduces entropy substantially. In predictive modeling terms, the attribute provides a lot of information on the value of the target.

Using IG for Selecting Informative Attributes

- As a second example, consider another candidate split shown in the right Figure. This is the same parent set as in Figure 3-4, but instead we consider splitting on the attribute *Residence* with three values: OWN, RENT, and OTHER. Without showing the detailed calculations:

entropy(*parent*) ≈ 0.99
entropy(Residence=OWN) ≈ 0.54
entropy(Residence=RENT) ≈ 0.97
entropy(Residence=OTHER) ≈ 0.98
IG ≈ 0.13

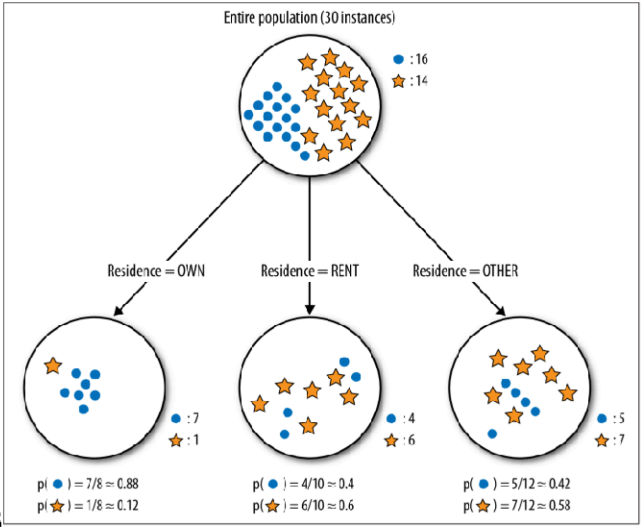


Figure 3-5. A classification tree split on the three-valued Residence attribute.

Thus, based on these data, the Residence variable is less informative than Balance (compare with slide page 19-20).

Outline

Supervised Segmentation with Tree-Structured Models

- If we select the single variable that gives the most information gain, we create a very simple segmentation. If we select multiple attributes each giving some information gain, it's not clear how to put them together.
- Recall from earlier that we would like to create segments that use multiple attributes, such as "Middle-aged professionals who reside in New York City on average have a churn rate of 5%."
- We now introduce an elegant application of the ideas we've developed for selecting important attributes, to produce a multivariate (multiple attribute) supervised segmentation.
- Consider a segmentation of the data to take the form of a "tree"

Example of classification tree

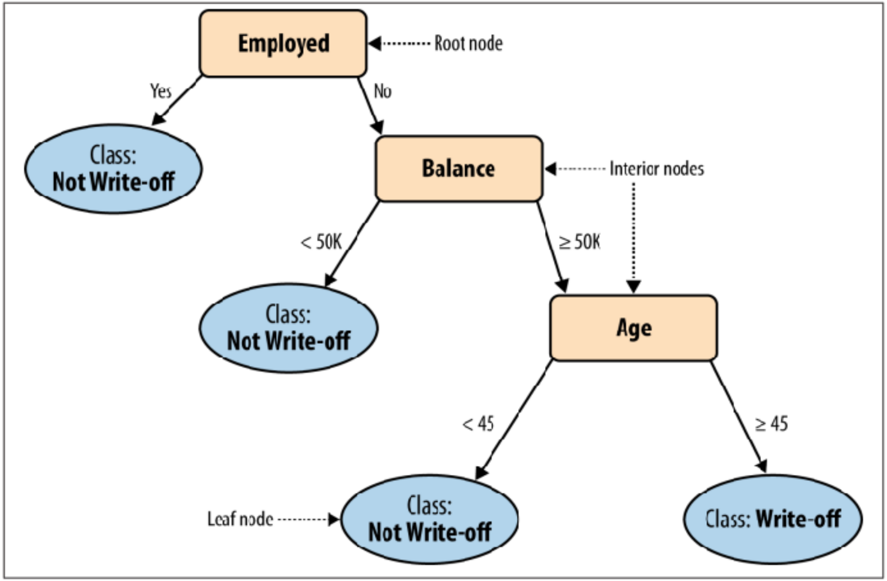


Figure 3-10. A simple classification tree.

Node, Leaf in Decision Tree

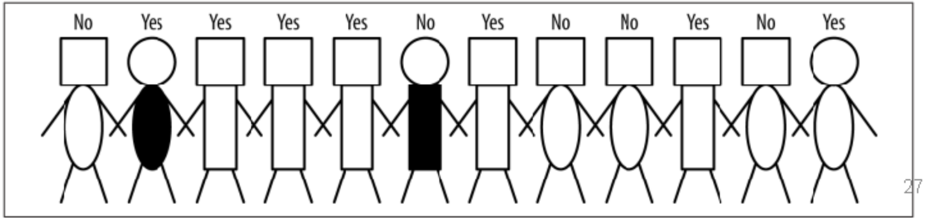
- In the page 24, the tree is upside down with the root at the top. The tree is made up of *nodes*, interior nodes and terminal nodes, and branches emanating from the interior nodes. Each interior node in the tree contains a test of an attribute, with each branch from the node representing a distinct value of the attribute.
- Following the branches from the root node down (in the direction of the arrows), each path eventually terminates at a terminal node, or *leaf*.
- The tree creates a segmentation of the data: every data point will correspond to one and only one path in the tree, and thereby to one and only one leaf. In other words, each leaf corresponds to a segment, and the attributes and values along the path give the characteristics of the segment. So the rightmost path in the tree in the figure corresponds to the segment "Older, unemployed people with high balances."
- The tree is a *supervised* segmentation, because each leaf contains a value for the target variable. Since we are talking about classification, here each leaf contains a classification for its segment. Such a tree is called a *classification tree* or more loosely a *decision tree*.

How to use Decision Tree?

- Consider how we would use the classification tree in the Figure of the page 24 o classify an example of the person named Claudio from Table in page 7. The values of Claudio's attributes are Balance=115K, Employed=No, and Age=40.
- We begin at the root node that tests Employed. Since the value is No we take the right branch. The next test is Balance. The value of Balance is 115K, which is greater than 50K so we take a right branch again to a node that tests Age. The value is 40 so we take the left branch. This brings us to a leaf node specifying class=Not Write-off, representing a prediction that Claudio will not default.
- Another way of saying this is that we have classified Claudio into a segment defined by (Employed=No, Balance=115K, Age<45) whose classification is Not Write –off.

How to Create a Tree Model

- Combining the ideas introduced above, the goal of the tree is to provide a supervised segmentation—more specifically, to partition the instances, based on their attributes, into subgroups that have similar values for their target variables. We would like for each "leaf " segment to contain instances that tend to belong to the same class.
- Tree induction takes a divide-and-conquer approach, starting with the whole dataset and applying variable selection to try to create the "purest" subgroups possible using the attributes.
- consider the very simple example set shown previously



“Splitting” to Obtain the Largest IG

- In the example, one way is to separate people based on their body type: rectangular versus oval. This creates the two groups shown in the right figure. How good is this partitioning?
- The rectangular-body people on the left are mostly *Yes*, with a single *No* person, so it is mostly pure. The oval-body group on the right has mostly *No* people, but two *Yes* people.
- This step is simply a direct application of the attribute selection ideas presented above. Let’s consider this “split” to be the one that yields the largest information gain

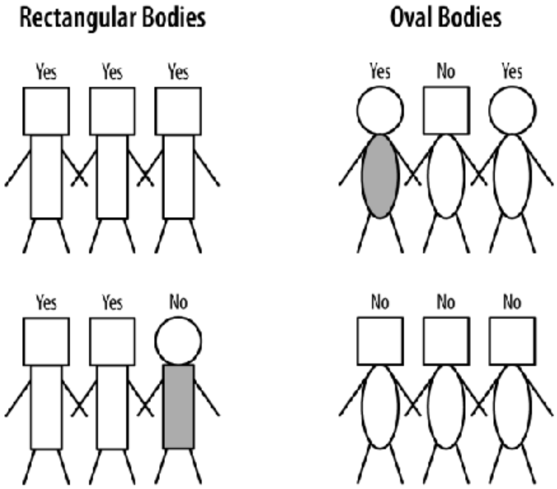


Figure 3-11. First partitioning: splitting on body shape (rectangular versus oval).

Second Partitioning

- The left and right subgroups are simply smaller versions of the problem with which we initially were faced! We can simply take each data subset and *recursively* apply attribute selection (use IG) to find the best attribute to partition it. So in our example, we recursively consider the oval-body group.

- To split this group again we now consider another attribute: head shape. This splits the group in two on the right side of the figure.
- These groups are "maximally pure" with respect to class labels and there is no need to split them further.

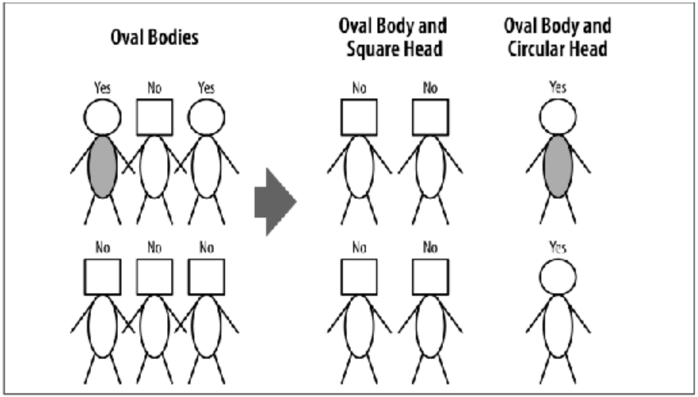


Figure 3-12. Second partitioning: the oval body people sub-grouped by head type.

Third Partitioning

- We still have not done anything with the rectangular body group on the left side of Figure in page 28, so let's consider how to split them. There are five Yes people and one No person.

- There are two attributes we could split upon: head shape (square or round), and body color (white or gray). Either of these would work, so we arbitrarily choose body color. This produces the groupings in the figure.

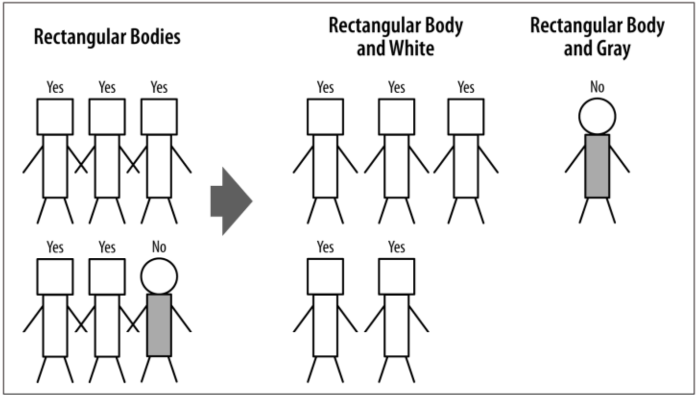
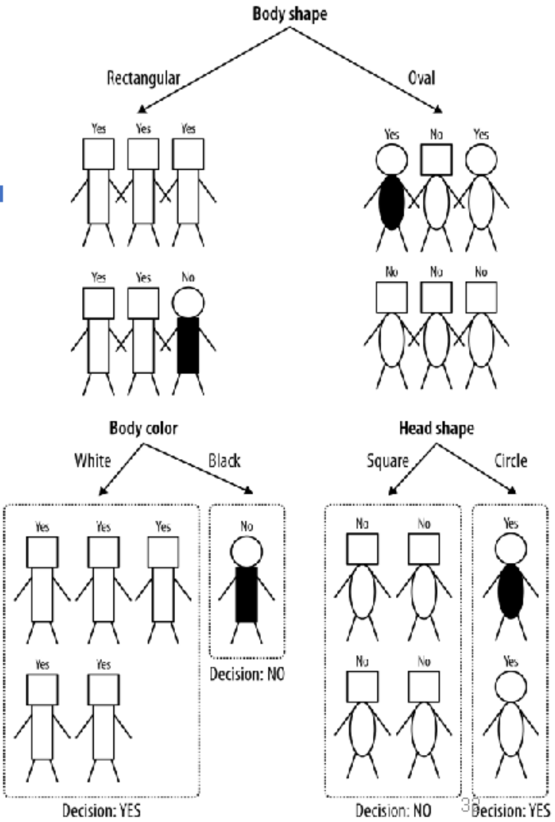


Figure 3-13. Third partitioning: the rectangular body people subgrouped by body color.

Summary for creating Tree Model

- In summary, the procedure of classification tree induction is a recursive process of divide and conquer, where the goal at each step is to select an attribute to partition the current group into subgroups that are as pure as possible with respect to the target variable.
- We perform this partitioning recursively, splitting further and further until we are done.
- When are we done? (In other words, when do we stop recursing?) It should be clear that we would stop when the nodes are pure, or when we run out of variables to split on.
- The whole Decision tree is shown in the next page.

Decision Tree



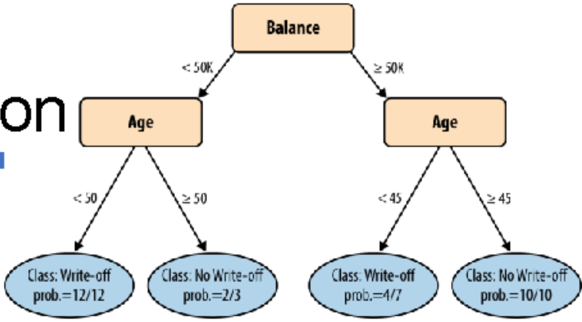
Outline

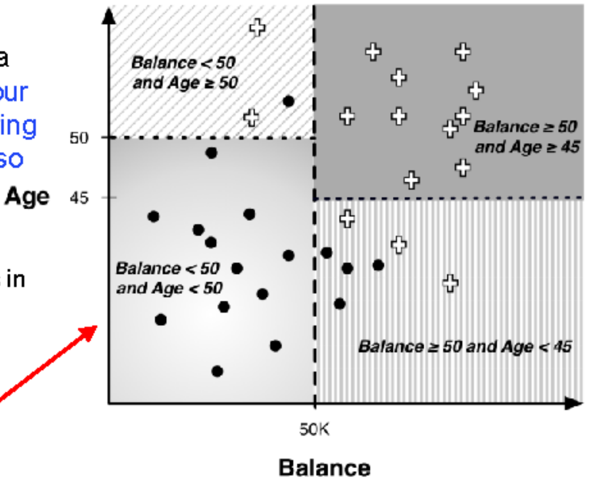
Necessary for Probability Estimation

- In many decision-making problems, we would like a more informative prediction than just a classification.
- For example, in our churn-prediction problem, rather than simply predicting whether a person will leave the company within 90 days of contract expiration, we would much rather have an estimate of the probability that he will leave the company within that time. Such estimates can be used for many purposes.
- You might then rank prospects by their probability of leaving, and then allocate a limited incentive budget to the highest probability instances.

Example for Probability Estimation

- So, in the context of supervised segmentation, we would like each segment (leaf of a tree model) to be assigned an estimate of the probability of membership in the different classes.
- The right figure more generally shows a "probability estimation tree" model for our simple write-off prediction example, giving not only a prediction of the class but also the estimate of the probability of membership in the class.


Balance
< 50K
≥ 50K
Age
Age
< 50
≥ 50
< 45
≥ 45
Class: Write-off prob.=12/12
Class: No Write-off prob.=2/3
Class: Write-off prob.=4/7
Class: No Write-off prob.=10/10



A classification tree and the partitions it imposes in instance space. The black dots correspond to instances of the class Write-off, the plus signs correspond to instances of class non-Write-off. The shading shows how the tree leaves correspond to segments of the population in instance space.

Probability Estimation in Decision Tree

- Fortunately, the tree induction ideas we have discussed so far can easily produce probability estimation trees instead of simple classification trees.
- Recall that the tree induction procedure subdivides the instance space into regions of class purity (low entropy). If we are satisfied to assign the same class probability to every member of the segment corresponding to a tree leaf, we can use instance counts at each leaf to compute a class probability estimate.
- For example, if a leaf contains *n* positive instances and *m* negative instances, the probability of any new instance being positive may be estimated as *n/(n+m)*. This is called a *frequency-based* estimate of class membership probability.

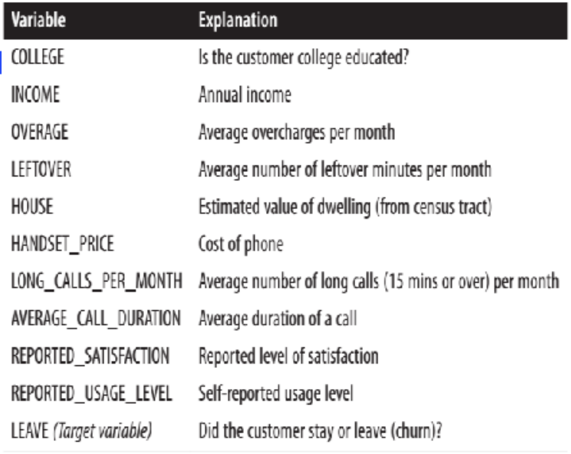
Outline

1. Models and Prediction
2. Supervised Segmentation - Information Gain
3. Supervised Segmentation - Decision Tree
4. Probability Estimation
5. Example: Addressing the Churn Problem with Tree Induction

Historical data set of 20,000 customers for churn problem

- Now that we have a basic data mining technique for predictive modeling, let's consider the churn problem again. How could we use tree induction to help solve it?
- For this example, we have a historical data set of 20,000 customers. At the point of collecting the data, each customer either had stayed with the company or had left (churned). Each customer is described by the variables listed in Table.

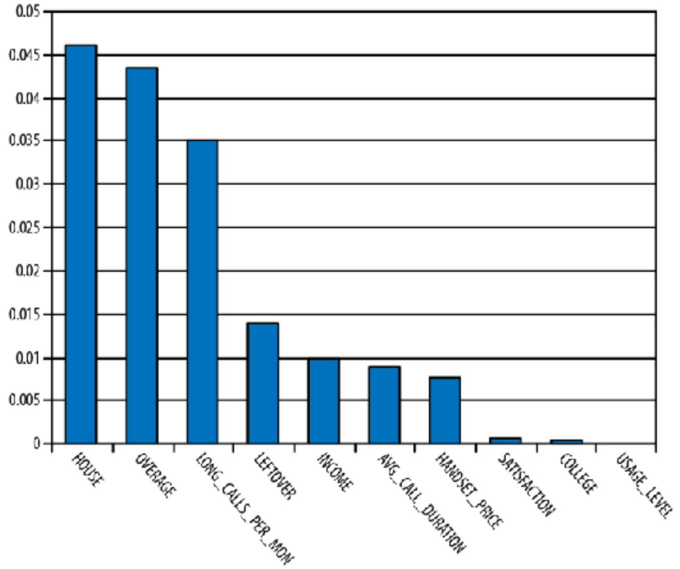
Table 3-2. Attributes for the cellular phone churn-prediction problem



Variable	Explanation
COLLEGE	Is the customer college educated?
INCOME	Annual income
OVERAGE	Average overcharges per month
LEFTOVER	Average number of leftover minutes per month
HOUSE	Estimated value of dwelling (from census tract)
HANDSET_PRICE	Cost of phone
LONG_CALLS_PER_MONTH	Average number of long calls (15 mins or over) per month
AVERAGE_CALL_DURATION	Average duration of a call
REPORTED_SATISFACTION	Reported level of satisfaction
REPORTED_USAGE_LEVEL	Self-reported usage level
LEAVE *(Target variable)*	Did the customer stay or leave (churn)?

Using IG to Evaluate Variables

- These variables comprise basic demographic and usage information available from the customer's application and account. We want to use these data with our tree induction technique to predict which new customers are going to churn.
- Before starting to build a classification tree with these variables, it is worth asking, *How good are each of these variables individually?* For this we measure the information gain of each attribute, as discussed earlier. Specifically, we apply Equation of IG to each variable independently over the entire set of instances, to see what each gains us.
- The results are in the next page, with a table listing the exact values. As you can see, the first three variables—the house value, the number of leftover minutes, and the number of long calls per month—have a higher information gain than the rest.8 Perhaps surprisingly, neither the amount the phone is used nor the reported degree of satisfaction seems, in and of itself, to be very predictive of churning.



Rank	Info. gain	Attribute name
1	0.0461	HOUSE
2	0.0436	OVERAGE
3	0.0350	LONG_CALLS_PER_MON
4	0.0136	LEFTOVER
5	0.0101	INCOME
6	0.0089	AVG_CALL_DURATION
7	0.0076	HANDSET_PRICE
8	0.0003	SATISFACTION
9	0.000	COLLEGE
10	0.000	USAGE_LEVEL

Figure 3-17. Churn attributes from Table 3-2 ranked by information gain.

Explanation for Tree Model

- Applying a classification tree algorithm to the data, we get the tree shown in the next page. The highest information gain feature (HOUSE) according to Figure in the last page is at the root of the tree. This is to be expected since it will always be chosen first.
- The second best feature, OVERAGE, also appears high in the tree. However, the order in which features are chosen for the tree doesn't exactly correspond to their ranking in Figure of the last page. Why is this?
- The answer is that the table ranks each feature by how good it is *independently*, evaluated separately on the entire population of instances. Nodes in a classification tree depend on the instances above them in the tree. Therefore, except for the root node, features in a classification tree are not evaluated on the entire set of instances.
- The information gain of a feature depends on the set of instances against which it is evaluated, so the ranking of features for some internal node may not be the same as the global ranking.

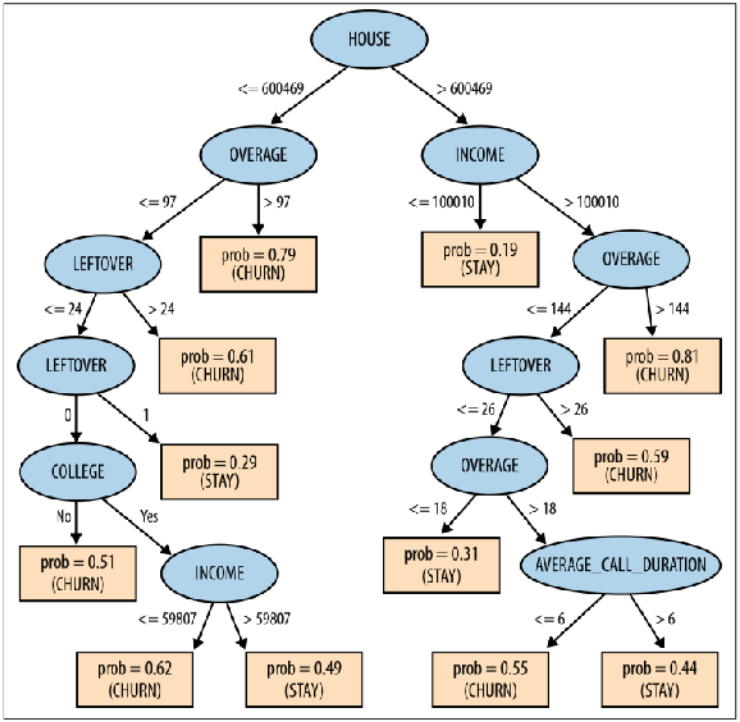


Figure 3-18. Classification tree learned from the cellular phone churn data.

Accuracy of Decision Tree

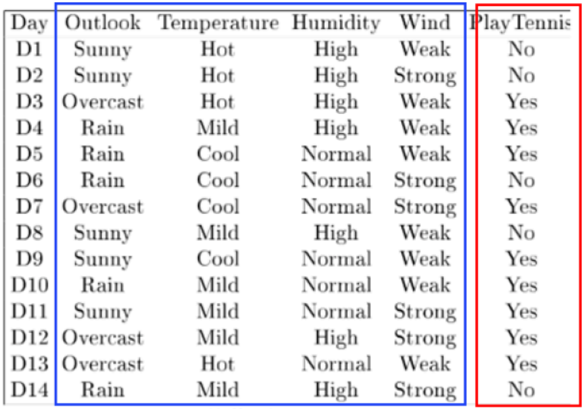
- Consider a final issue with this dataset. After building a tree model from the data, we measured its accuracy against the data to see how good of a model it is.
- Specifically, we used a training set consisting half of people who churned and the other half who did not; after learning a classification tree from this, we applied the tree to the dataset to see how many of the examples it could classify correctly. The tree achieved 73% accuracy on its decisions.

Summary

1. Models and Prediction
2. Supervised Segmentation - Information Gain
3. Supervised Segmentation - Decision Tree
4. Probability Estimation
5. Example: Addressing the Churn Problem with Tree Induction

Exercise

Imagine you play tennis, and you invite your friend. Your friend sometimes comes to join but sometimes not. For your friend, it depends on a number of factors, for example, weather, temperature, humidity, and wind. Please use the right dataset to build a decision tree which can predict whether or not your friend will join you to play tennis.



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

attributes

target

Data Science

Week 8

Fitting a Model to Data (2)-
Classification