QBUS6850

Lecture 6

Advanced Classification Techniques I

BUSINESS SCHOOL

QBUS6850 Team

THE UNIVERSITY OF SYDNEY

❑ Topics covered

- ➢ Decision trees intuition
- ➢ CART- classification and regression trees
- ➢ Decision stump
- ➢ ID3 algorithm
- ➢ Growing and pruning
- ➢ Node impurity
- ➢ Entropy and information gain

❑References

- ➢ Friedman et al., (2001), Chapters 9.2, 10, 15, 16
- ➢ James et al., (2014), Chapter 8
- ➢ Bishop, (2006), Chapters 14.3 - 14.4
- ➢ Alpaydin, (2014), Chapter 9

Learning Objectives

❑ Understand the intuition of decision trees

❑ Understand how decision trees works

❑ Understand what is decision stump

❑ Understand CART- Classification and Regression Trees

❑ Understand the growing and pruning

❑ Understand how ID3 algorithm works

❑ Understand Entropy and information gain

Decision Tree Intuition

Decision Trees:

❑ Non-parametric (any other nonparametric method we learnt before?)

❑ Supervised learning method that can be used for both classification and regression.

❑ Through incorporating a set of if-then-else rules, decision tree can be employed to predict target variable given data features

- Try to discover the **pattern** under which the customer will purchase the product

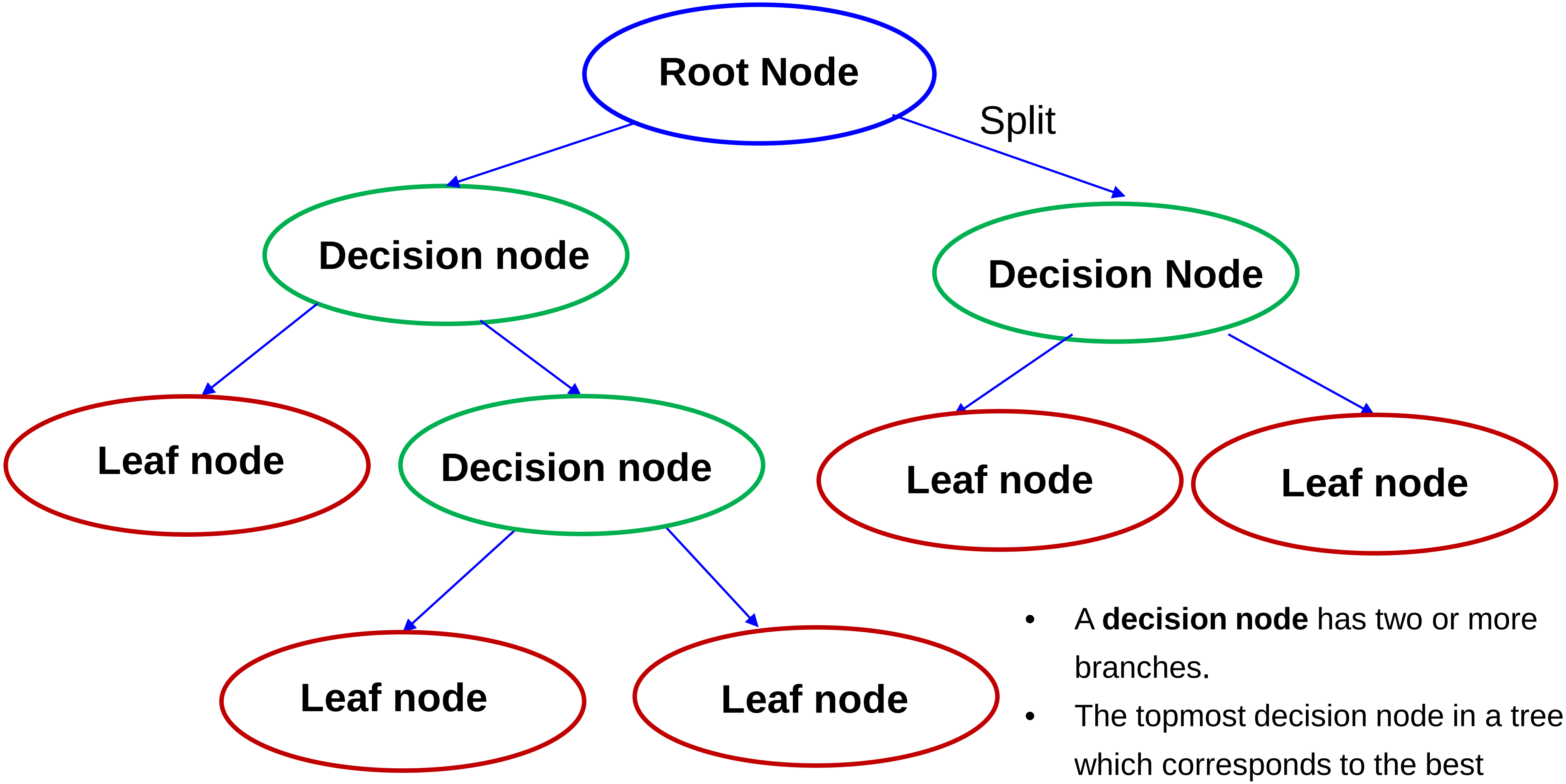
- Divide data set into subsets (branches of a tree)

- Check whether the **stopping criteria** is met

```
 If yes
       stop dividing
 Else
       keep dividing
```

- For a new customer, based on the features, we can see which subset the customer will fall into

Decision Tree



- A **decision node** has two or more branches.
- The topmost decision node in a tree which corresponds to the best predictor called **root node.**
- **Leaf node** represents a decision.

Decision Tree Types

Decision trees used in machine learning are of two main types:

- ❑ Classification tree analysis is when the predicted outcome is the class to which the data belongs. Target variable is categorical.
- ❑ Regression tree analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital). Target variable is continuous.

Classification And Regression Tree (CART), Breiman et al., (1984). An umbrella term used to refer to both of the above techniques.

Classification Trees

- ❑ The task of growing a classification tree is quite similar to the task of growing a regression tree

- ❑ Categorical response variable, e.g. yes/no, 1/0

- ❑ For a classification tree, we predict that each observation belongs to the most commonly occurring class (mode) of training observations in the region to which it belongs

- ❑ In interpreting the results of a classification tree, we are often interested not only in the class prediction corresponding to a particular leaf node region, but also in the class **proportions** among the training observations that fall into that region

THE UNIVERSITY OF SYDNEY

Decision Tree Classification

Customer	Income	Education	Marital Status	Purchase
1	Medium	University	Single	Yes
2	High	University	Single	No
3	High	University	Married	No
4	Low	University	Single	Yes
5	Low	High school	Single	Yes
6	Low	High school	Married	No
7	Medium	High school	Married	Yes
8	High	University	Single	No
9	High	High school	Single	Yes
10	Low	High school	Single	Yes
11	High	High school	Married	Yes
12	Low	University	Married	No
13	High	University	Single	No
14	Medium	University	Married	Yes
15	Medium	High school	Single	Yes

We can have duplicated records.

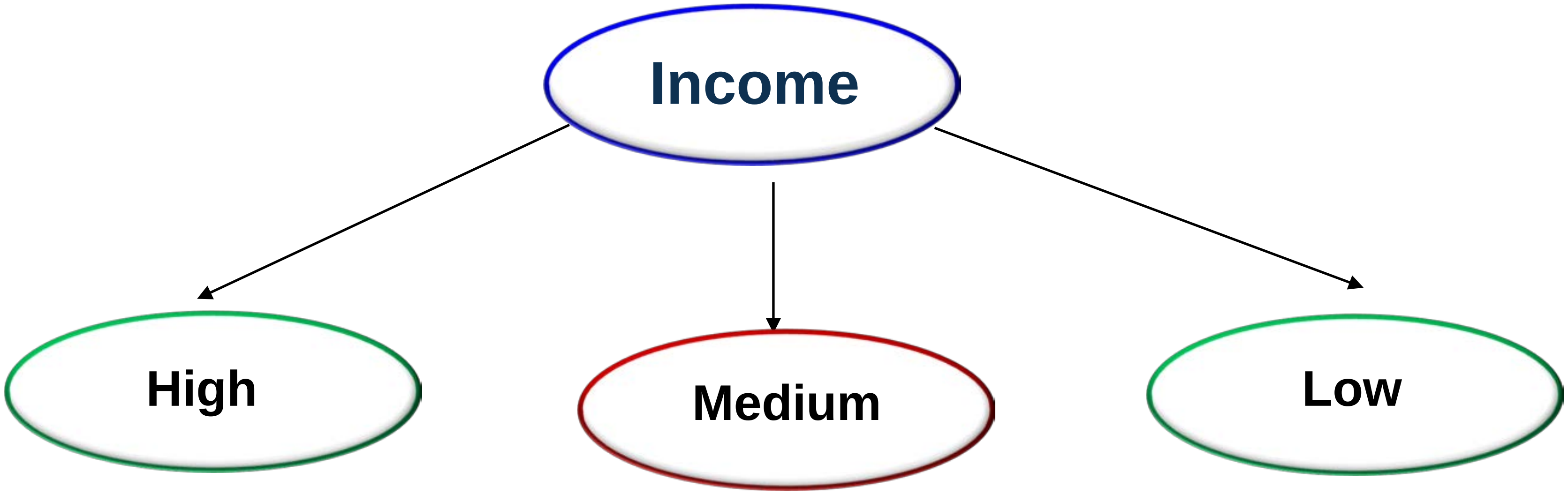
- ❑ We need to build the tree from the root node with one feature and then split examples into subsets

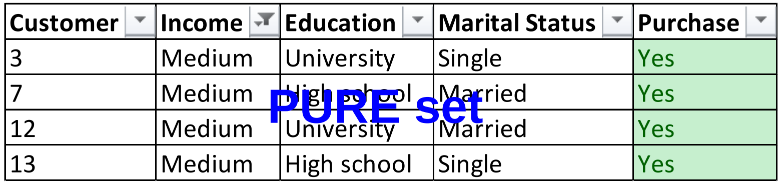
- ❑ How to select this feature?

- ❑ Idea: a good feature splits the examples into subsets that are (ideally) "all positive" or "all negative“

- ❑ Purity

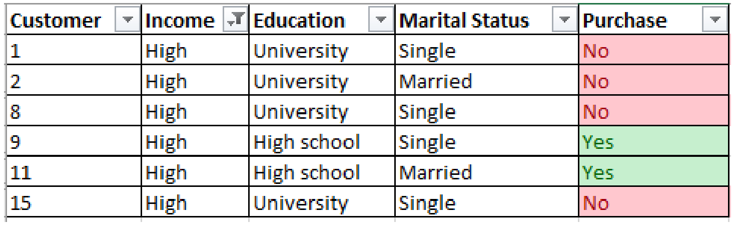
Let’s start the decision tree with feature income. Why?



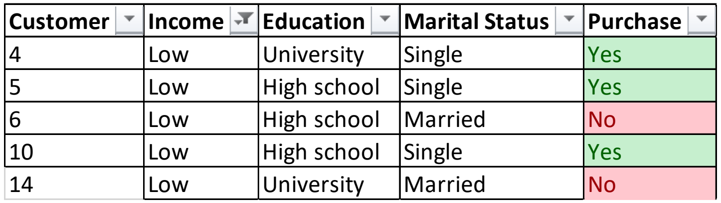


Customer	Income	Education	Marital Status	Purchase
3	Medium	University	Single	Yes
7	Medium	High school	Married	Yes
12	Medium	University	Married	Yes
13	Medium	High school	Single	Yes

PURE set

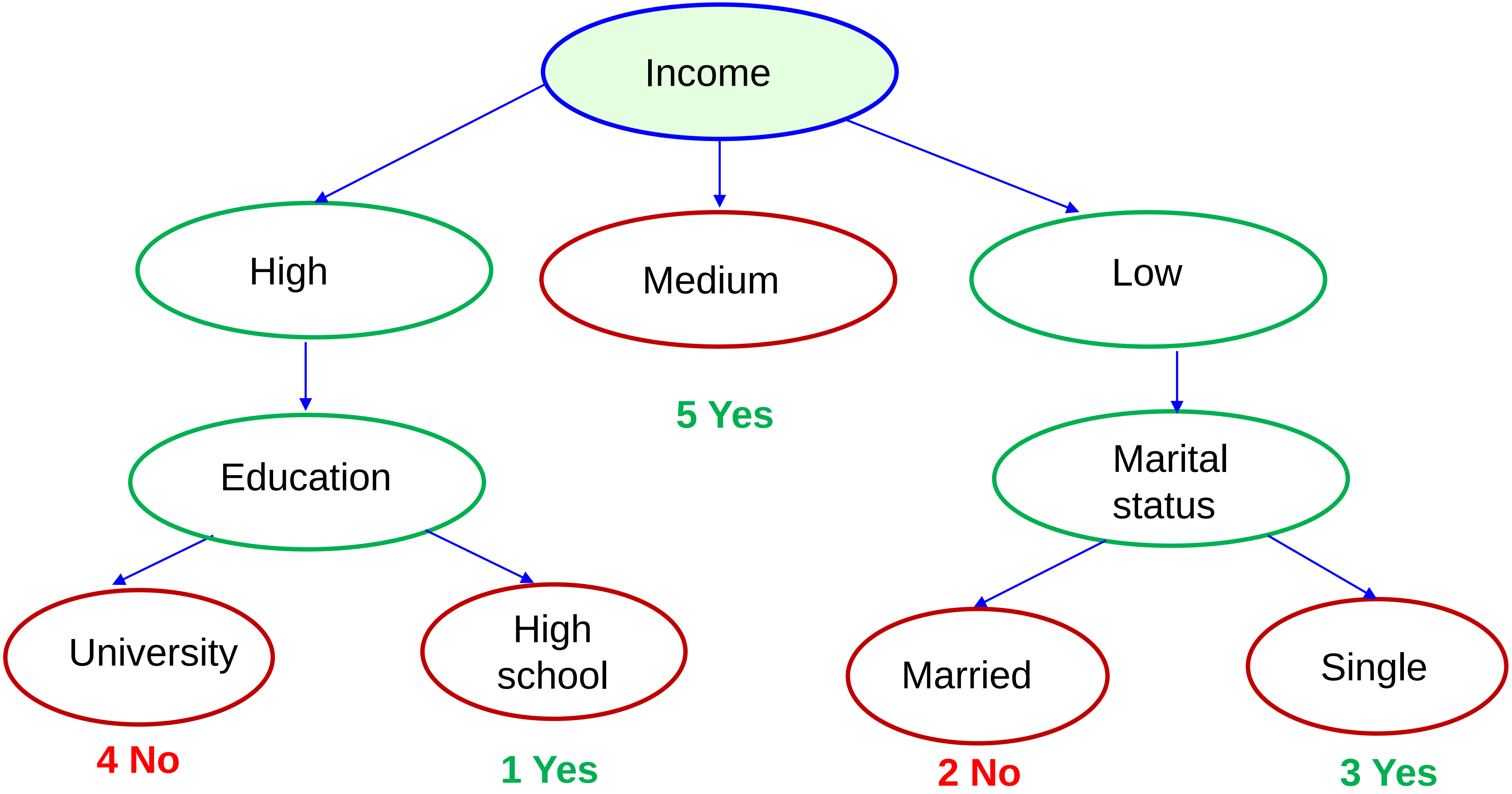


Customer	Income	Education	Marital Status	Purchase
1	High	University	Single	No
2	High	University	Married	No
8	High	University	Single	No
9	High	High school	Single	Yes
11	High	High school	Married	Yes
15	High	University	Single	No

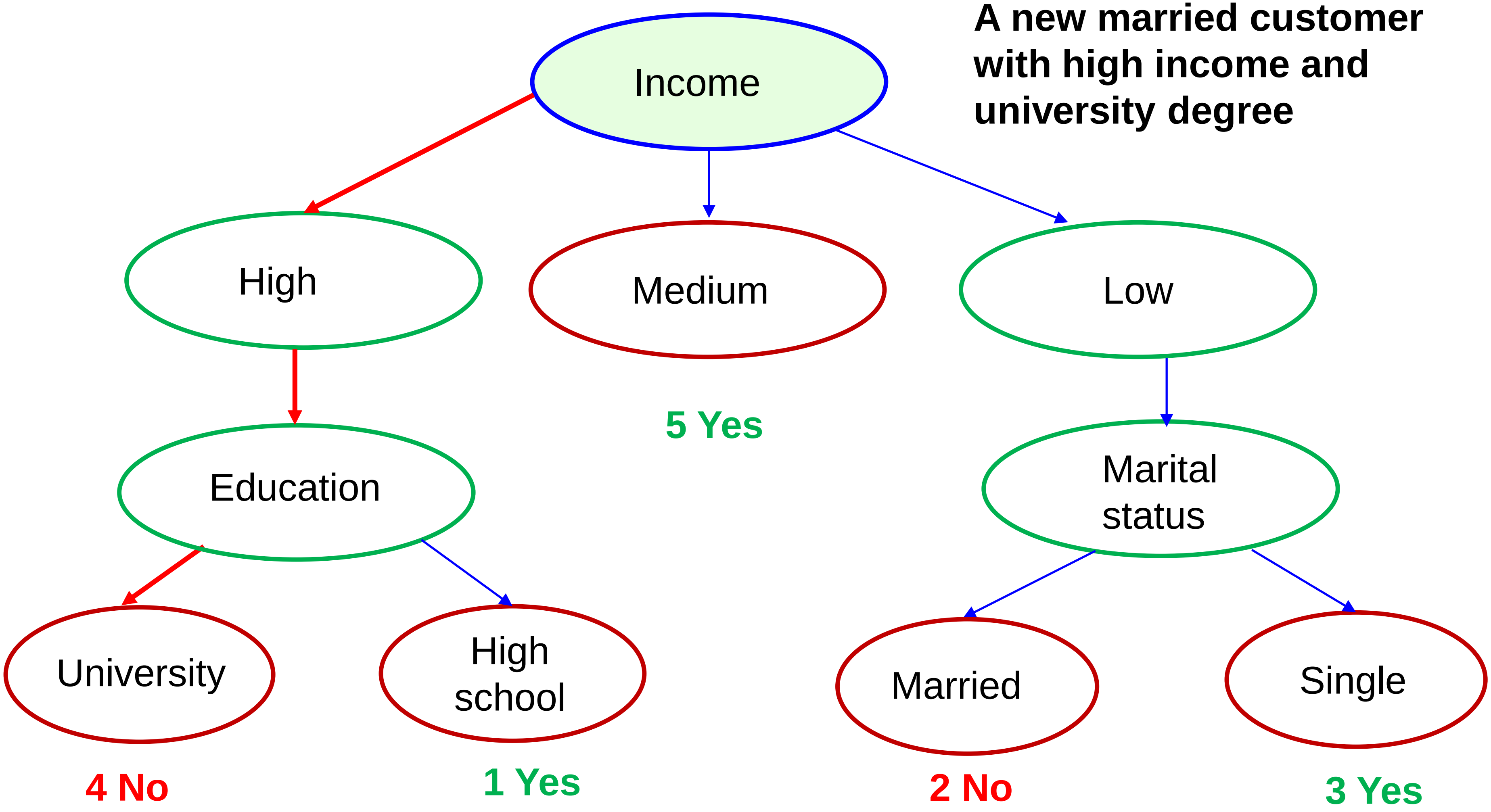


Customer	Income	Education	Marital Status	Purchase
4	Low	University	Single	Yes
5	Low	High school	Single	Yes
6	Low	High school	Married	No
10	Low	High school	Single	Yes
14	Low	University	Married	No

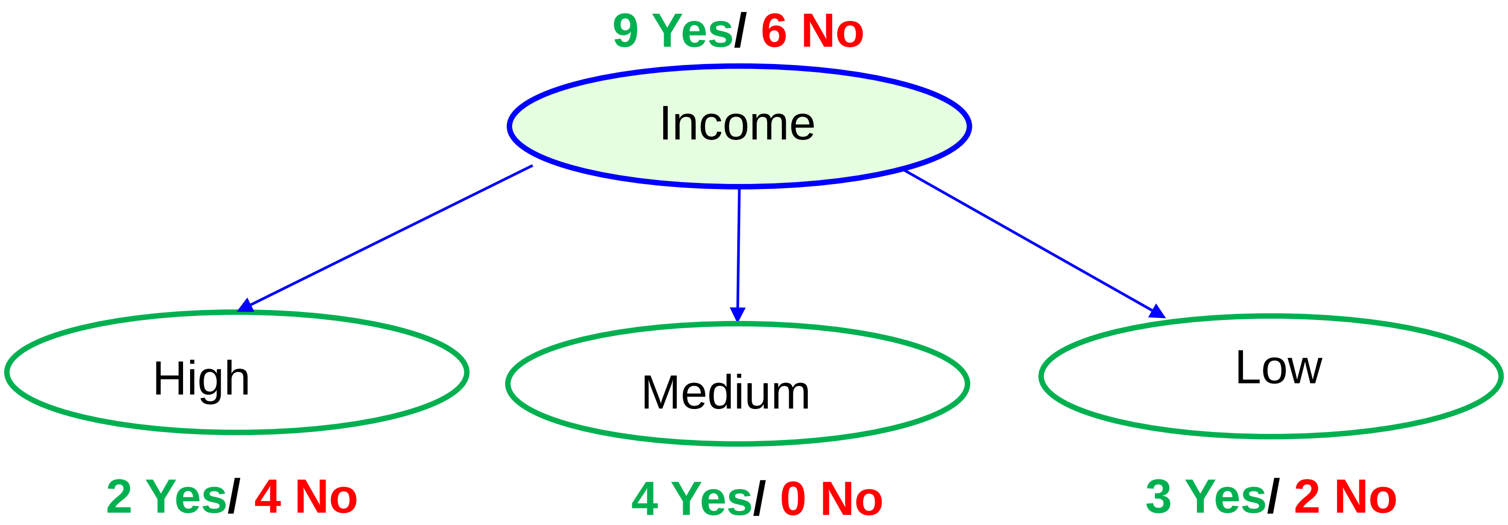
Let’s start the decision tree with feature income. Why?



A new married customer with high income and university degree



Best feature of splitting



9 Yes/ 6 No

Income

High

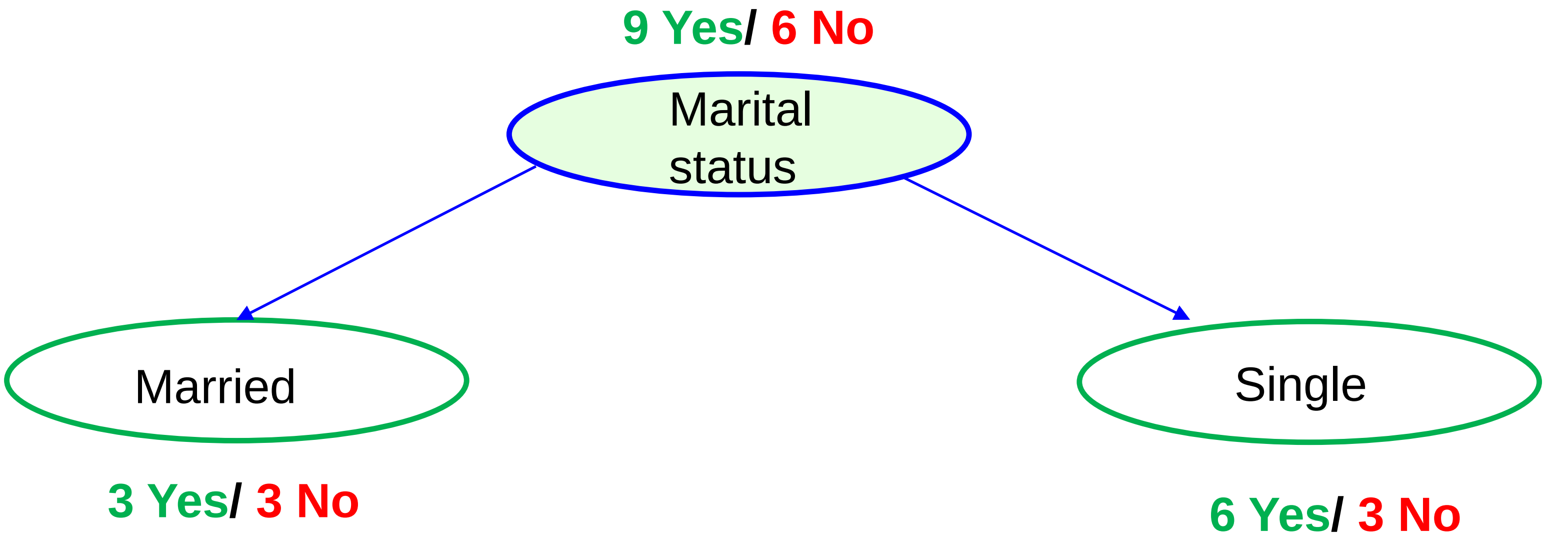
Medium

Low

2 Yes/ 4 No

4 Yes/ 0 No

3 Yes/ 2 No



9 Yes/ 6 No

Marital status

Married

Single

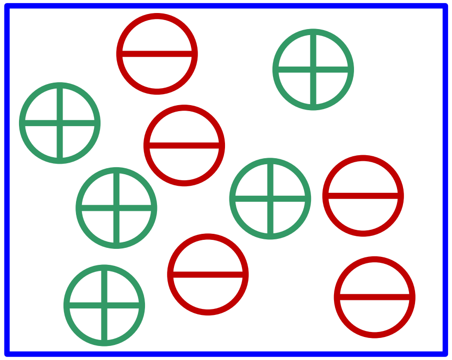
3 Yes/ 3 No

6 Yes/ 3 No

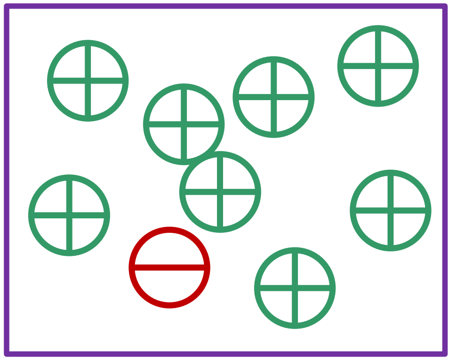
Entropy Intuition

- Entropy is a concept originally from physics and measures the disorder in a data set
- In decision trees, we use entropy $\boldsymbol{H(S)}$ to measure of the amount of uncertainty in the data set $\boldsymbol{S}$.
- The entropy will be a small value if the dataset is pure.

- **Smaller entropy, less disorder, higher PUTIRY (CERTAINTY)**
- **Larger entropy, more disorder, higher IMPUTIRY (UNCERTAINTY)**



$\boldsymbol{H(S) = 1}$



$\boldsymbol{H(S) = 0.469}$

A glass of water and ice cubes, which one is purer?

Measure the **PURITY** of the split:

Aim to be more certain about Yes/No after the spit

- Pure set: (4 yes/0 no)=> 100% certain
- Impure set: 3 yes/3 no => 50% certain and 50% uncertain

- Impure set: 1 yes/3 no => 25% certainty and 75% uncertain
  Should be as **PURE** as
- Impure set: 3 yes/1 no => 75% certainty and 25% uncertain

Entropy Calculation

Entropy $\boldsymbol{H}(\boldsymbol{S})$ is a measure of the amount of uncertainty in the data set $\boldsymbol{S}$. The entropy will be a **small** value if the dataset is **pure**.

$$H(\mathbf{S}) = \sum_{k=1}^{K} p_k(\mathbf{S}) \log_2\left(\frac{1}{p_k(\mathbf{S})}\right) = -\sum_{k=1}^{K} p_k(\mathbf{S}) \log_2\big(p_k(\mathbf{S})\big)$$

- $\mathbf{S}$: The current (data) set for which entropy is being calculated (changes every iteration of the ID3 algorithm)
- $p_k(\mathbf{S})$: The proportion of the number of elements in class k to the number of elements in set $\mathbf{S}$. K classes in total in $\mathbf{S}$.
- $\sum_{k=1}^{K} p_k(\mathbf{S}) = 1$
- $p_k(\mathbf{S}) \log_2\big(p_k(\mathbf{S})\big)$ equals zero when $p_k(\mathbf{S}) = 0$.

Entropy- Two Classes

More specifically, for a training set with $\boldsymbol{p}$ **positive examples** and $\boldsymbol{n}$ **negative examples**:

$$H(\mathbf{S}) = -\boxed{\frac{p}{p+n}}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Equivalently:

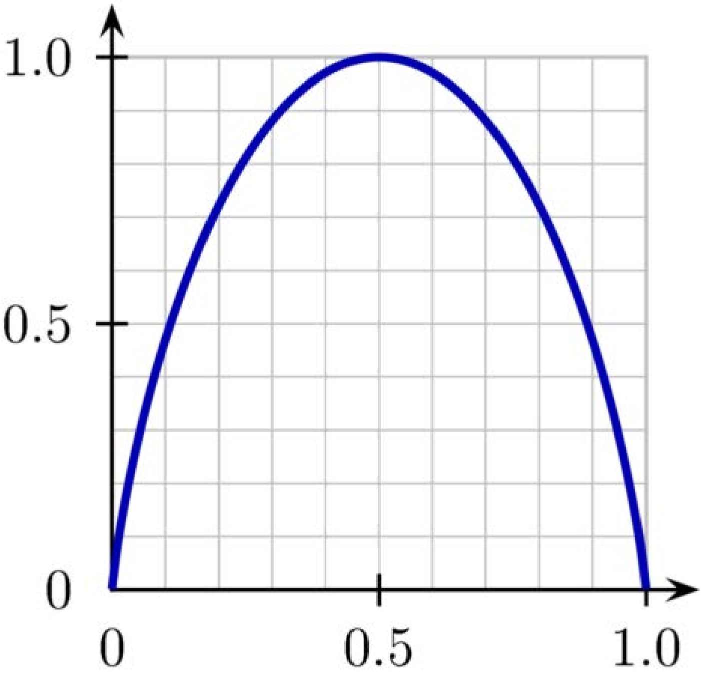
% of positive examples in ***S***

$$H(\mathbf{S}) = -\boxed{p_+(\mathbf{S})}\log_2 p_+(\mathbf{S}) - p_-(\mathbf{S})\log_2 p_-(\mathbf{S})$$

Interpretation: assume an item belongs to **S**, how many **bits** of information are required to tell whether **x** is positive or negative. The smaller it is, the higher certainty.

Entropy- Two Class

A two class problem



When $\boldsymbol{H}(\boldsymbol{S}) = \mathbf{0}$, the set $\boldsymbol{S}$ is perfectly classified, e.g. all elements in $\boldsymbol{S}$ are of the same class

$$p_{-}(\mathbf{S}) = 0.5, p_{+}(\mathbf{S}) = 0.5, H(\mathbf{S}) = 1$$

$$p_{-}(\mathbf{S}) = 0, p_{+}(\mathbf{S}) = 1, H(\mathbf{S}) = 0$$

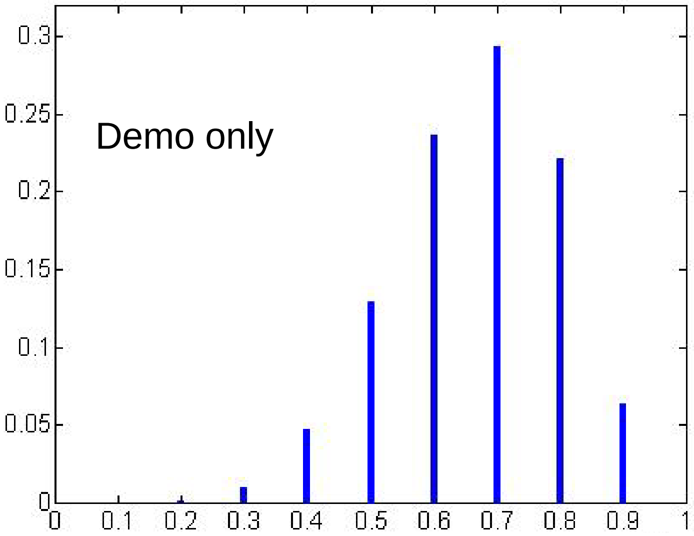
Symmetric

$$p_{+}(\mathbf{S}) = 0, p_{-}(\mathbf{S}) = 1, H(\mathbf{S}) = 0$$

Entropy- Multiple Classes

If there are more than two classes: $1,2,\ldots,K$:

$$\begin{aligned} H(\mathbf{S}) = & -p_1(\mathbf{S})\log_2 p_1(\mathbf{S}) \\ & -p_2(\mathbf{S})\log_2 p_2(\mathbf{S}) \\ & -p_3(\mathbf{S})\log_2 p_3(\mathbf{S}) \\ & \ldots\ \ldots \\ & -p_K(\mathbf{S})\log_2 p_K(\mathbf{S}) \end{aligned}$$

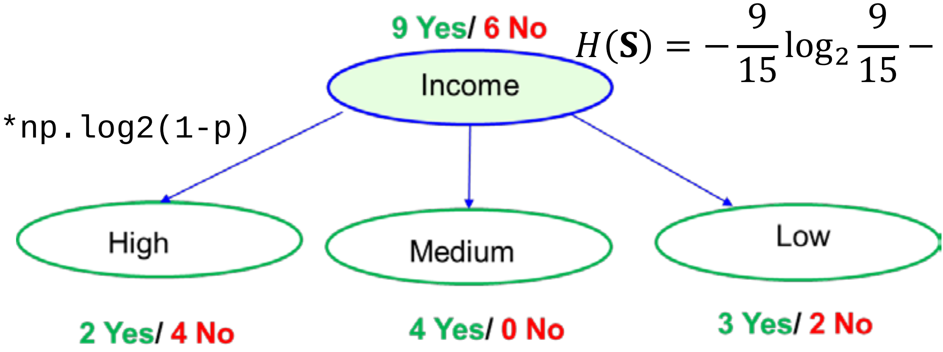


K* classes in total in *S

$$\sum_{k=1}^{K} p_i(\mathbf{S}) = 1$$

Example

```
# entropy calculator
p= 3.0/5
H= - p*np.log2(p)-(1-p)*np.log2(1-p)
```



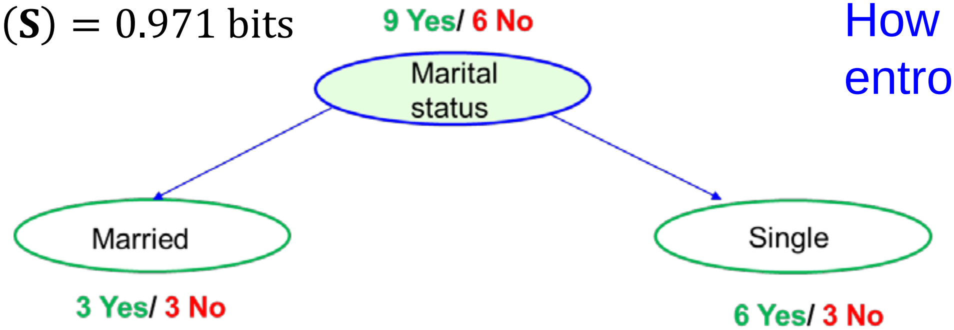
$$H(\mathbf{S}) = -\frac{9}{15}\log_2\frac{9}{15} - \frac{6}{15}\log_2\frac{6}{15} = 0.971 \text{ bits}$$

$$H(\mathbf{S}_2) = 0 \text{ bits}$$

$$H(\mathbf{S}_1) = -\frac{2}{6}\log_2\frac{2}{6} - \frac{4}{6}\log_2\frac{4}{6} = 0.918 \text{ bits}$$

$$H(\mathbf{S}_3) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971 \text{ bits}$$

$$H(\mathbf{S}) = 0.971 \text{ bits}$$



How to merge these entropy together?

$$H(\mathbf{S}_1) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1 \text{ bit}$$

$$H(\mathbf{S}_2) = -\frac{6}{9}\log_2\frac{6}{9} - \frac{3}{9}\log_2\frac{3}{9} = 0.918 \text{ bits}$$

Other Measurements

- Entropy is not the only measurement of selecting the best feature to split

- Other measurements include:
  - Gini index

$$H(\mathbf{S}) = \sum_{k=1}^{K} p_i(\mathbf{S})(1 - p_i(\mathbf{S}))$$

  - The Gini index and the entropy are similar numerically

  - Misclassification rate: not sufficiently sensitive for tree-growing. James et al., (2014).

Information Gain

- How much information do we gain if we disclose/split the value of some features?
- Answer: uncertainty before minus uncertainty after
- **Information Gain (IG)** or reduction in entropy from the feature test
- Information Gain is a measure of the disorder/uncertainty decrease achieved by splitting the data set S
- Choose the feature split with the **largest** IG

$$\boxed{\textbf{Information Gain}} = \textbf{Entropy before} - \boxed{\textbf{Entropy after}}$$

We want this term to be large

Weighted sum of Entropy.
We want this term to be small.

Information Gain

Information gain IG(A) is the measure of the difference in entropy from before to after the data set $\boldsymbol{S}$ is split on an feature A.

In other words, how much **uncertainty** in $\boldsymbol{S}$ was **reduced** after splitting set $\boldsymbol{S}$ on feature A.

$$IG(\mathbf{S}, A) = H(\mathbf{S}) - EH(A)$$

$H(\mathbf{S})$ – Entropy of set $\boldsymbol{S}$

$EH(A)$ – Expected entropy with split by feature A

Expected Entropy

A selected feature A with J distinct values, e.g. feature “income” has $J = 3$ possible values “high”, “medium” and “low”, partitions the training set $\boldsymbol{S}$ into J subsets/branches $\boldsymbol{S}_1, \boldsymbol{S}_2, \dots, \boldsymbol{S}_J$

The **expected entropy** with split by feature A is:

Weights based on size of the subset

$$EH(A) = \sum_{j=1}^{J} \boxed{\frac{|\boldsymbol{S}_j|}{|\boldsymbol{S}|}} \boxed{H(\boldsymbol{S}_j)}$$

Note this is the entropy of the subset, calculated according to the target categories

$\boldsymbol{S}$: the current (data) set for which entropy is being calculated

$\boldsymbol{S}_j$: subset j

Expected entropy is a measurement of subsets impurity.

Information Gain Example

Entropy before split. High impurity.

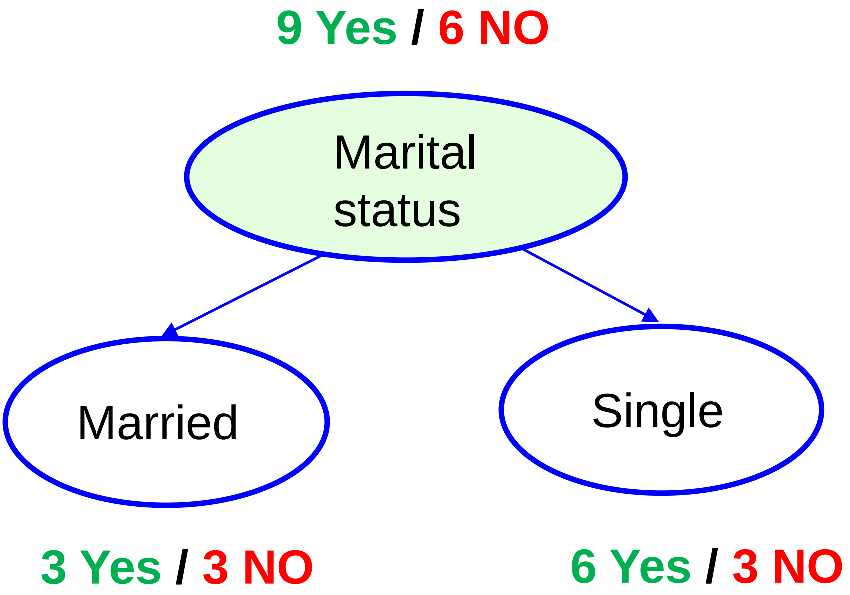
$$H(\mathbf{S}) = -\frac{9}{15}\log_2\frac{9}{15} - \frac{6}{15}\log_2\frac{6}{15} = 0.971 \text{ bits}$$

Weights based on size of the subsets to S

$$IG(S,A)$$

$$= H(S) - \boxed{\frac{6}{15}} H(S_{\text{Married}}) - \boxed{\frac{9}{15}} H(S_{\text{Single}})$$

$$= 0.97 - \frac{6}{15}\times 1 - \frac{9}{15}\times 0.91 = 0.0239$$



If split on “marital status”, we would **GAIN** 0.0239 bits on certainty.
Or we are 0.0239 bits more certain.

Entropy after split

$$H(S_{\text{Married}}) = 1 \qquad H(S_{\text{Single}}) = 0.918$$

Information Gain Drawback

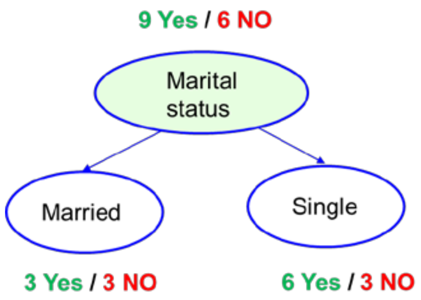
- ❑ IG favours split on an feature with many values (many leaf nodes): causing bias
- ❑ If 1 feature splits in many more classes than another, it has an (unfair) advantage if we use information gain
- ❑ The Gain-Ratio is designed to compensate for this problem

$$\text{GainRatio} = \frac{\text{Information Gain}}{\boxed{\text{Split Entropy}}}$$

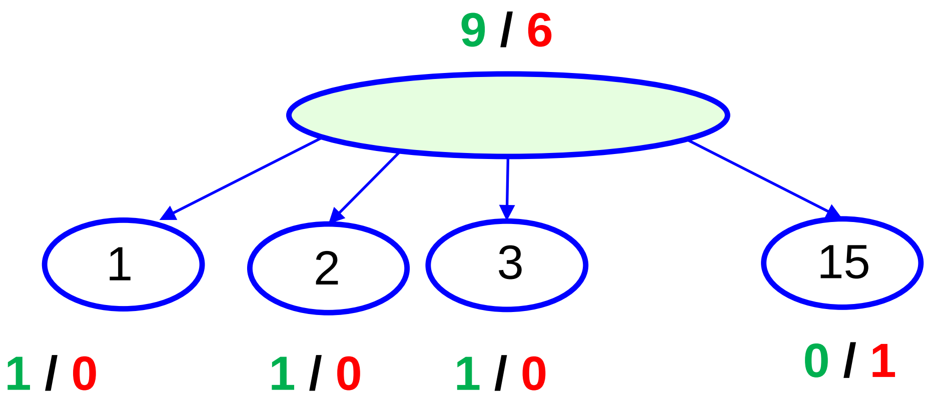
Penalize split with too many small subsets

$$\text{Split\_Entropy}(\boldsymbol{S}, A) = -\sum_{j=1}^{J} \frac{|\mathbf{S}_j|}{|\mathbf{S}|} \log_2 \frac{|\mathbf{S}_j|}{|\mathbf{S}|}$$

Split Entropy Example



$$\text{Split Entropy} = -\frac{6}{15}log_2\left(\frac{6}{15}\right) - \frac{9}{15}log_2\left(\frac{9}{15}\right) = 0.971$$



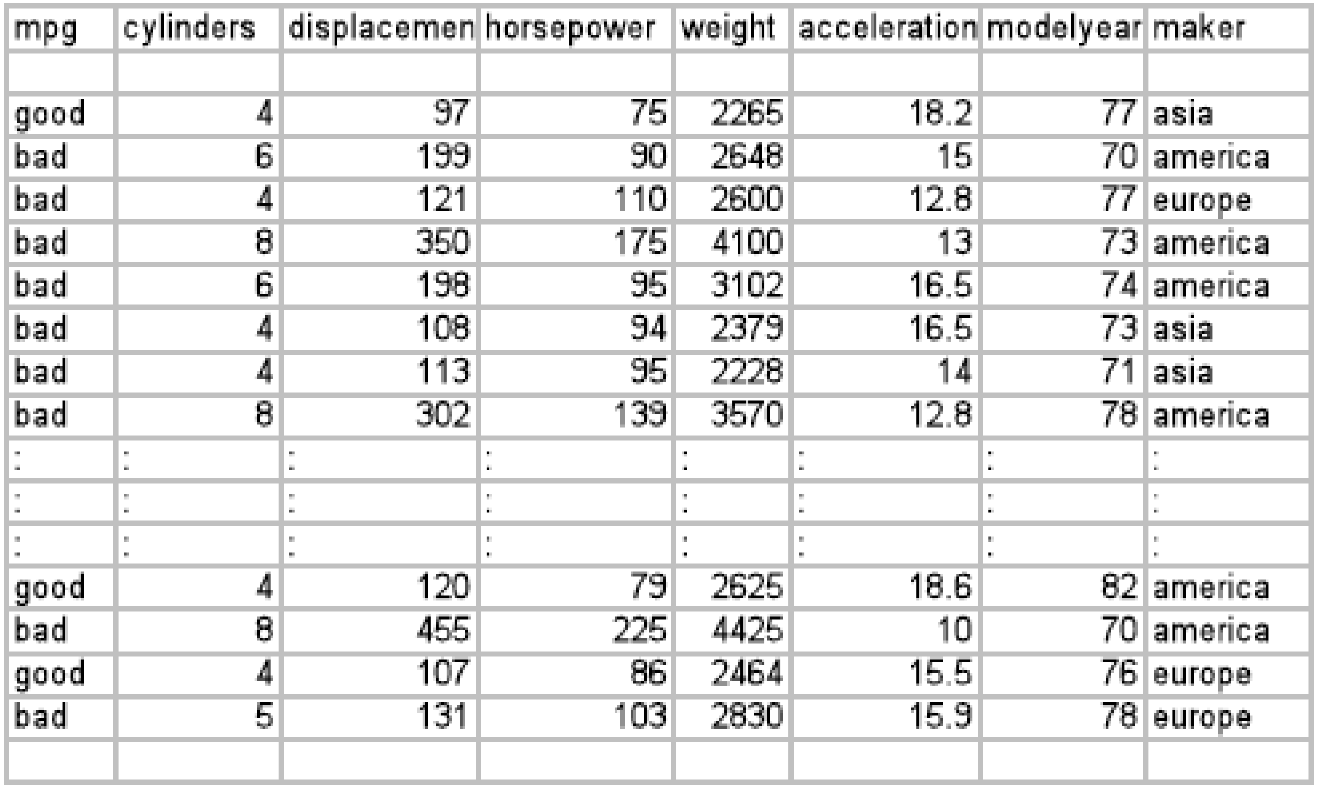
Penalize split with too many small subsets, although the IG for such split is high.

$$\text{Split Entropy} = -15\left[\frac{1}{15}log_2\left(\frac{1}{15}\right)\right] = 3.907$$

Split over Numeric Features

- ➢ What should we do if some of the features are numeric/continuous?
- ➢ We use the form of $x < \theta$ where θ is called a splitting value or cutting point.

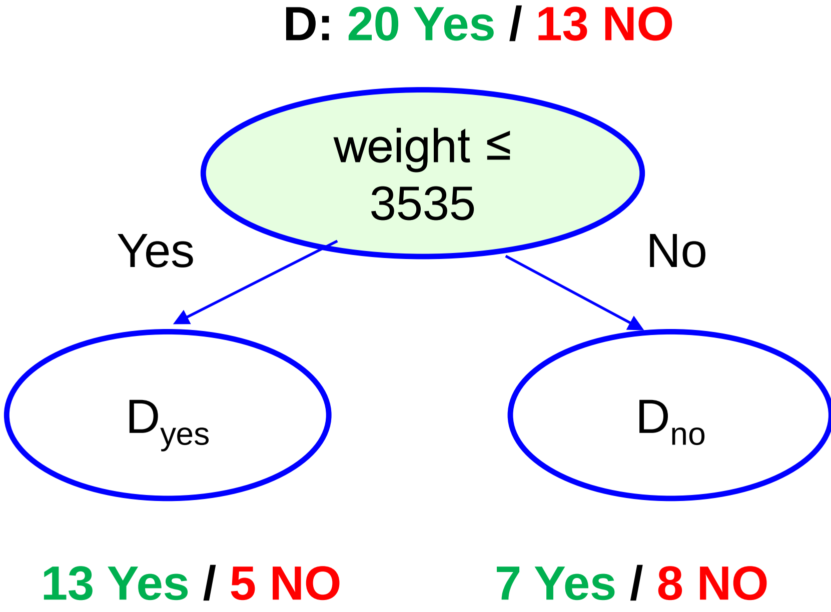
Infinite number of possible split values!!!

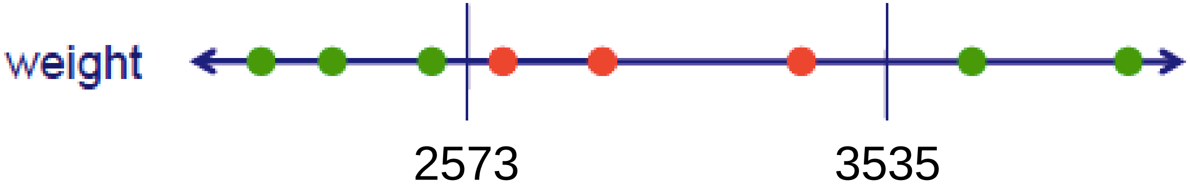


mpg	cylinders	displacemen	horsepower	weight	acceleration	modelyear	maker
good	4	97	75	2265	18.2	77	asia
bad	6	199	90	2648	15	70	america
bad	4	121	110	2600	12.8	77	europe
bad	8	350	175	4100	13	73	america
bad	6	198	95	3102	16.5	74	america
bad	4	108	94	2379	16.5	73	asia
bad	4	113	95	2228	14	71	asia
bad	8	302	139	3570	12.8	78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
good	4	120	79	2625	18.6	82	america
bad	8	455	225	4425	10	70	america
good	4	107	86	2464	15.5	76	europe
bad	5	131	103	2830	15.9	78	europe

Split over Numeric Features

- Splits on numeric features use a threshold
- How to decide a threshold like 3535 in the diagram
- Consider a feature A. Sort the values of A in data
- Evaluate split thresholds in intervals between instances of different classes





ID3 Algorithm: Summary

Ross Quinlan, 1986

The ID3 algorithm begins with the original set **S** as the root node.

For each iteration of the algorithm:

- Loop through every unused feature of the set S and calculates the information gain $\boldsymbol{IG}(\boldsymbol{S})$ of that feature.
- Select the feature which has the largest information gain value, **best feature of splitting**
- S is then split by the **selected feature**, e.g. income, to produce subsets of the data.
- The algorithm continues to loop on each subset, **excluding** features used before.

Stopping Criteria

- ❑ All elements in the subset belong to the same class (Yes or No, 1 or 0, + or -), then the node is turned into a leaf node and labelled with the class of the examples
- ❑ No more features to be selected, while the examples still do not belong to the same class (some are 1 and some are 0), then the node is turned into a leaf node and labelled with the most common class of the examples in the subset
- ❑ No examples in the subset, for example if there is no example with age >= 100. Then a leaf node is created, and labelled with the most common class of the examples in the parent set.

What to do if...

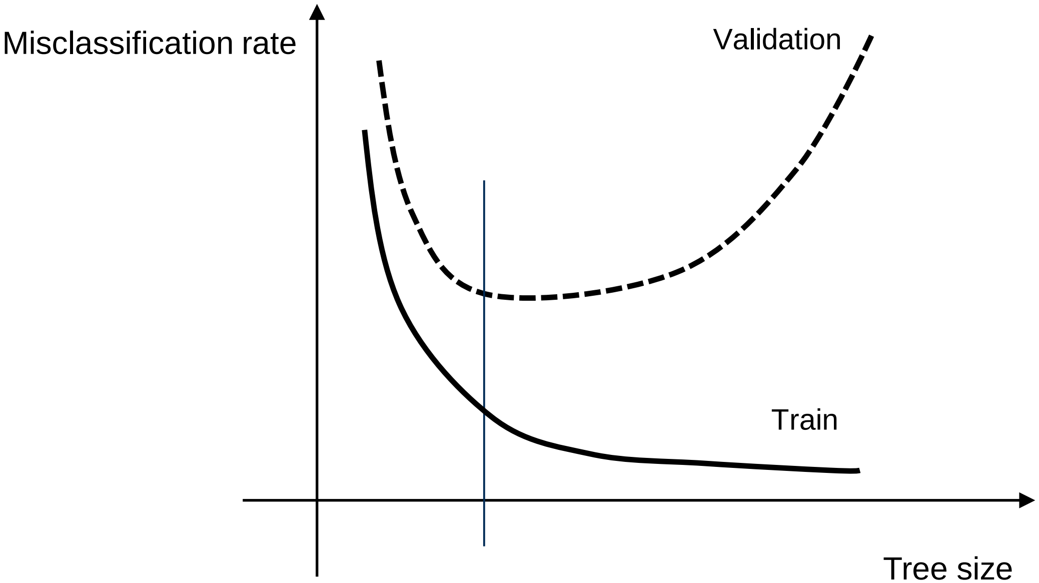
- ❑ In some leaf nodes there are no examples:
  - ➢ Choose yes or no according to the number of yes/no examples at parent

- ❑ Some examples have the same features but different label: we have an error/noise
  - ➢ Stop and use majority vote

- ❑ In the applications of our unit, we focus more on decision tree with **binary** split. Also, scikit-learn uses an optimised version of the CART algorithm which constructs binary trees. This is because scikit-learn tree algorithms only deal with numeric features.

Overfitting in Decision Trees

- ❑ If we keep growing the tree until perfect classification for the training set we might over-fit
- ❑ For example, we can keep splitting the tree until each node contains 1 example
- ❑ This will fit perfectly on the training data, while NOT work on the new test data



Tree Pruning

Prepruning:

Stop growing when data split is not statistically significant. For example: stop tree construction when node size is smaller than a given limit, or impurity of a node is below a given limit. (faster)

Postpruning:

Grow the whole tree, then prune subtrees which overfit on the validation set. (more accurate)

We don’t touch those techniques

How to Avoid Overfitting?

- ❑ **Prepruning :** stop splitting when there is no statistically significant:
  - ➢ Stop when Info-Gain (Gain-Ratio) is smaller than threshold
  - ➢ Stop when there are p, e.g. $p = 5$, examples in each leaf node
- ❑ **Postpruning:** grow the tree, then post-prune it based on validation set
- ❑ **Regularization:** penalize complex trees by minimizing with “complexity” = “# of leaf nodes”. $|T|$ indicates the number of leaf nodes of the tree T

Note: if tree grows, complexity grows, but entropy shrinks (uncertainty decreases).

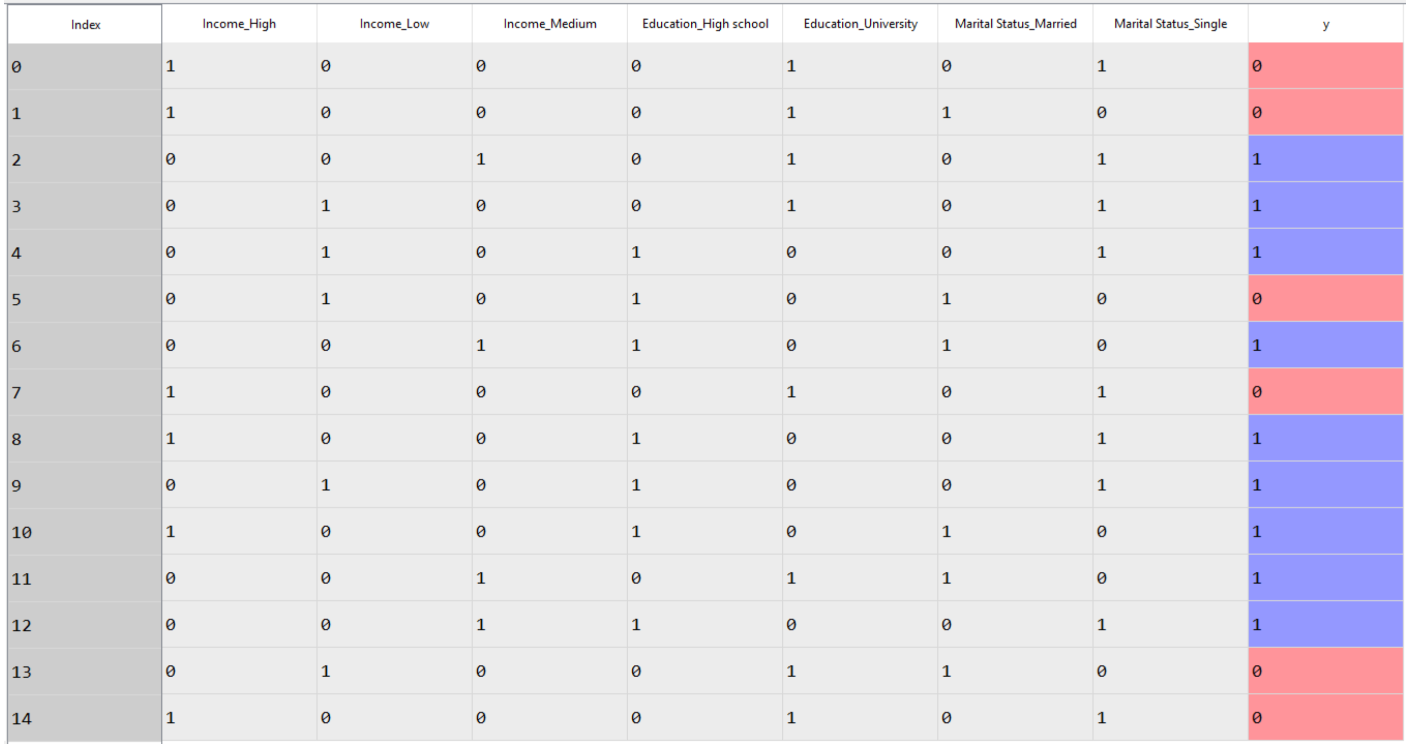
$$\sum_{\text{All leaf nodes}} H(S_j) + \lambda * |T|$$

- ❑ Compute many trees on subsets of data and test: pick the best, or do prediction vote

- ❑ Random Forests are state of the art classifiers!

Python Example

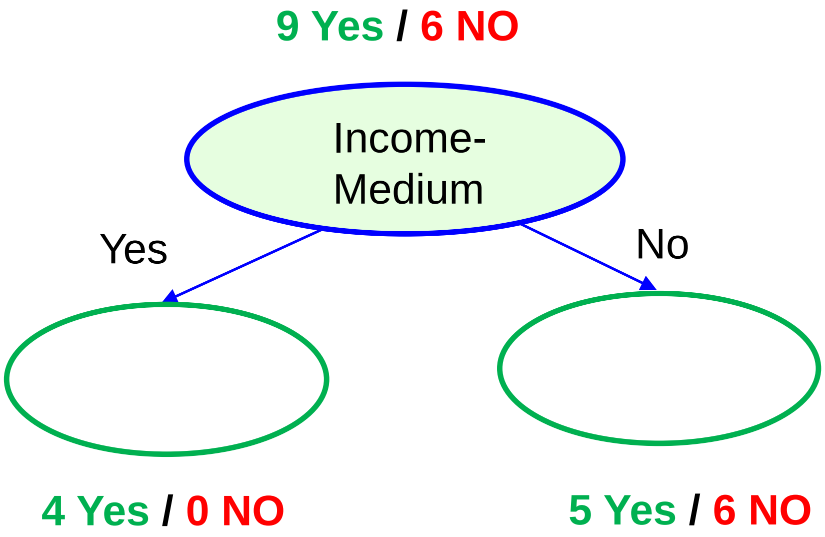
In the real implementation, we transform the categorical features into dummy variables.



Index	Income_High	Income_Low	Income_Medium	Education_High school	Education_University	Marital Status_Married	Marital Status_Single	y
0	1	0	0	0	1	0	1	0
1	1	0	0	0	1	1	0	0
2	0	0	1	0	1	0	1	1
3	0	1	0	0	1	0	1	1
4	0	1	0	1	0	0	1	1
5	0	1	0	1	0	1	0	0
6	0	0	1	1	0	1	0	1
7	1	0	0	0	1	0	1	0
8	1	0	0	1	0	0	1	1
9	0	1	0	1	0	0	1	1
10	1	0	0	1	0	1	0	1
11	0	0	1	0	1	1	0	1
12	0	0	1	1	0	0	1	1
13	0	1	0	0	1	1	0	0
14	1	0	0	0	1	0	1	0

Used expected entropy as impurity measurement to select the best feature for 1st split (depth 1).

[0.82584103752696802,
 0.97095059445466847,
 0.72895548841643465,
 0.78514543156506744,
 0.78514543156506744,
 0.95097750043269369,
 0.95097750043269369]

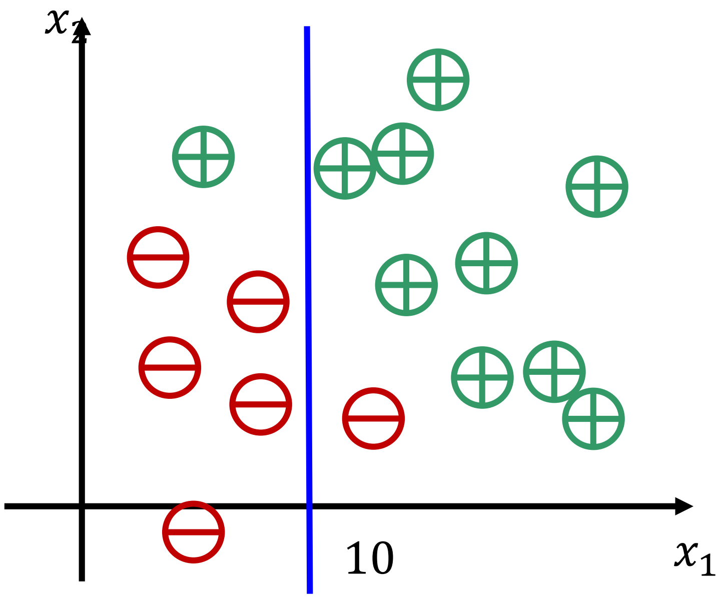


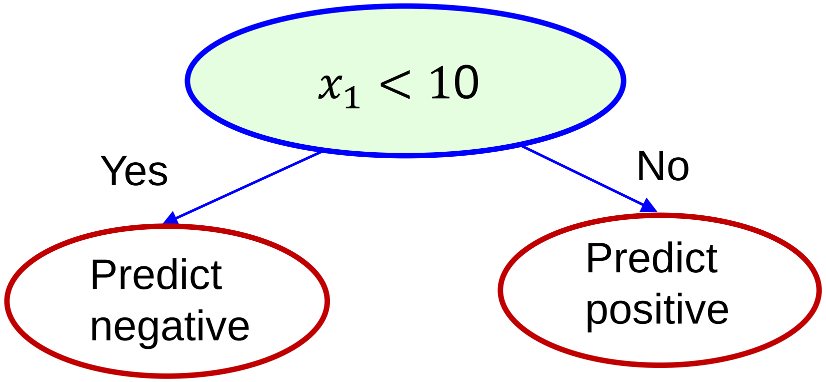
Income-Medium is selected as the best feature.

Left node: [0. 4.] => [0 no, 4 yes]
Right node: [6. 5.] => [6 no, 5 yes]

Decision Stump

- A decision stump is a decision tree consisting of only one-level.
- A decision tree with one root node which is immediately connected to the leaf nodes.
- We will use this concept to explain the boosting of the next lecture

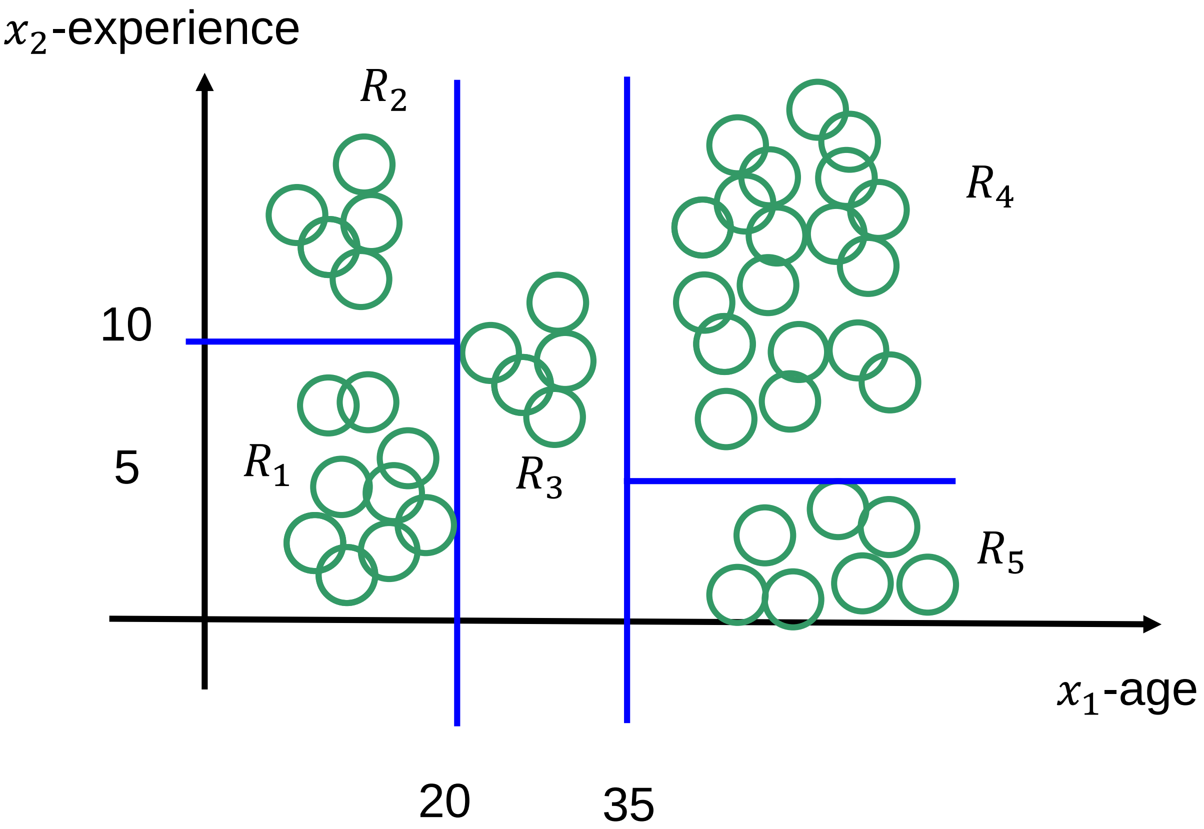




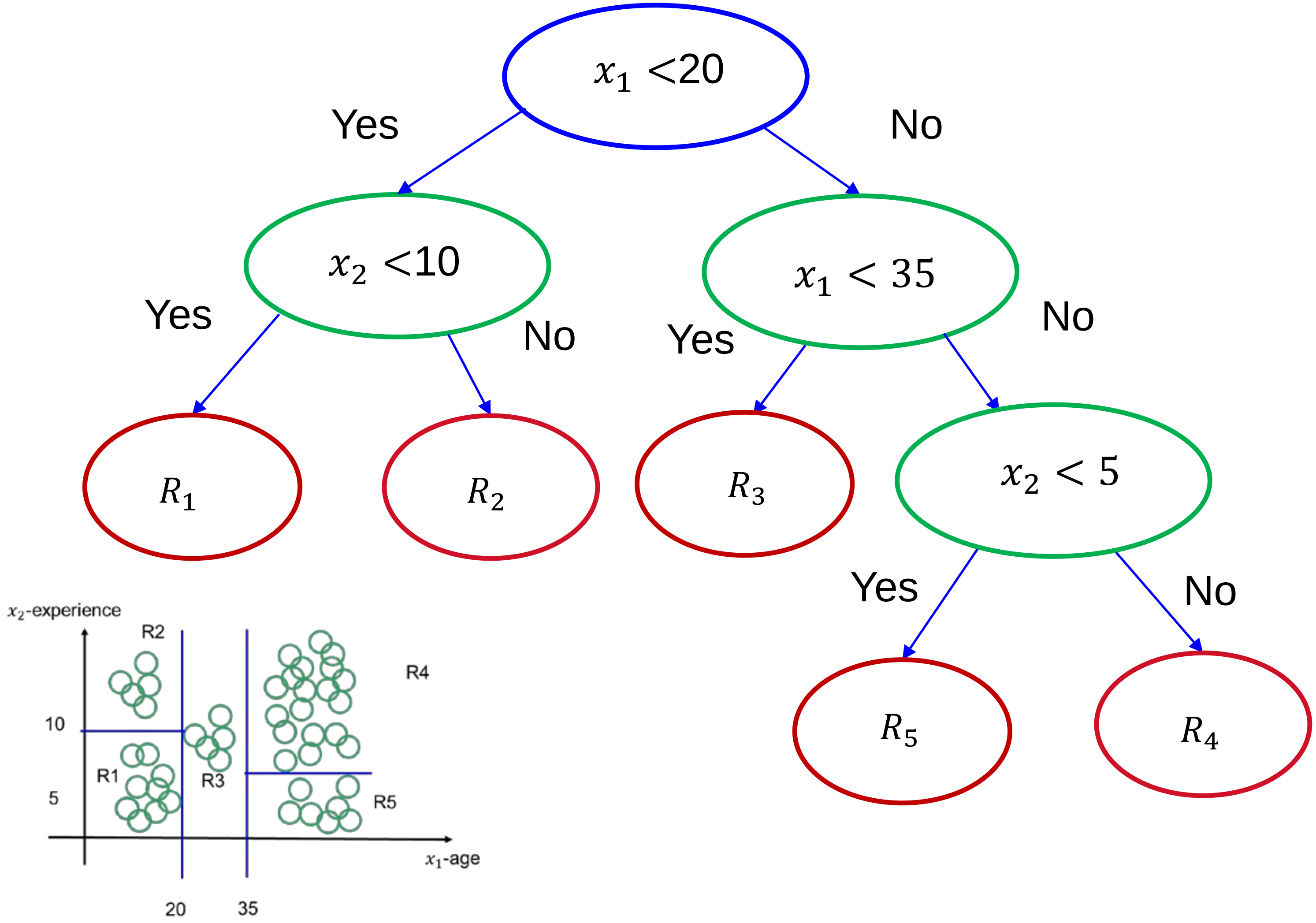
Regression Tree

James et al., (2014)

Regression Tree



Regression Tree



x_1 <20

Yes

No

x_2 <10

Yes

No

R_1

R_2

$x_1 < 35$

Yes

No

R_3

$x_2 < 5$

Yes

No

R_5

R_4

Building Regression Tree

Two steps of building a regression tree:

1. Partition the feature space: the set of possible values for $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ into J distinct and non-overlapping regions for R_1, R_2, …, R_J

2. For a new observation that falls into the region R_j, we make the same prediction, which is simply the **mean** of the response values for the training examples in R_j

Building Regression Tree

How do we construct the regions $R_1, R_2, \ldots, R_J$?
The goal is to find regions $R_1, R_2, \ldots, R_J$ that minimize the **Loss function**

$$\sum_{j=1}^{J} \sum_{n:\, \mathbf{x}_n \in R_j} \left(t_n - \hat{t}_{R_j}\right)^2$$

$\hat{t}_{R_j}$ is the mean response for the training examples within the j_{th} region.

How to find these regions? It is computationally infeasible to consider every possible partition of the feature space into J regions.

Splitting Criterion for Regression

- Information gain (IG) for classification is no longer available for regression as the target is not categorical

- The loss function means that we shall group data in terms of homogeneity, i.e., the target values in a group are as similar as possible, measured by the squared errors (or by the variance)

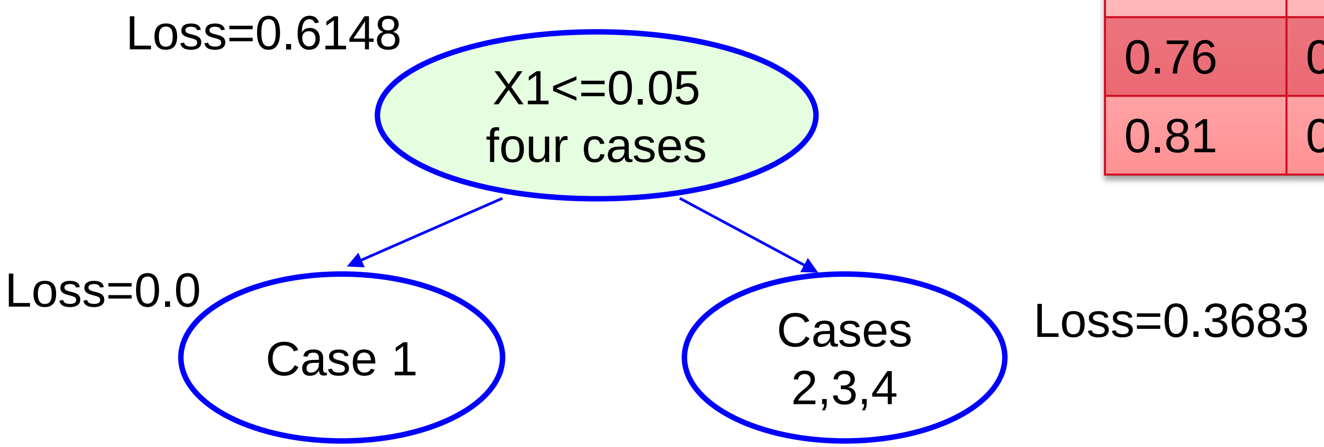
- Instead of IG, we prefer the feature and the splitting producing maximal squared error reduction, i.e.,

Loss= squared error before splitting – (expected) squared errors after splitting

Splitting for Numeric Features

- How can we select the cut-points for a chosen numeric feature?
- CART algorithm recommends using each value of that feature in the dataset as a cut-point
- Consider the example shown in table (four cases)
- Consider feature x1, the possible cut-points are (0.05, 0.32, 0.76, 0.81)
- For the cutting-point 0.05, we have

x1	x2	x2	t
0.05	0.31	0.51	0.97
0.32	0.41	0.88	0.89
0.76	0.61	0.48	0.11
0.81	0.94	0.85	0.19

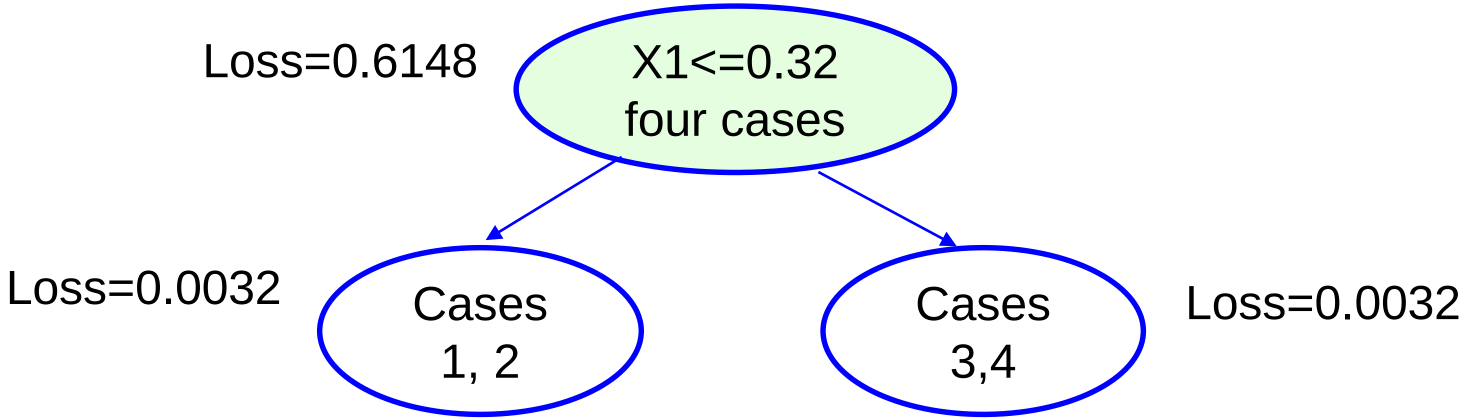


Splitting for Numeric Features

- The loss reduction is given by

  L = 0.6148 – ¼ * 0 – ¾ * 0.3683 = 0.3386

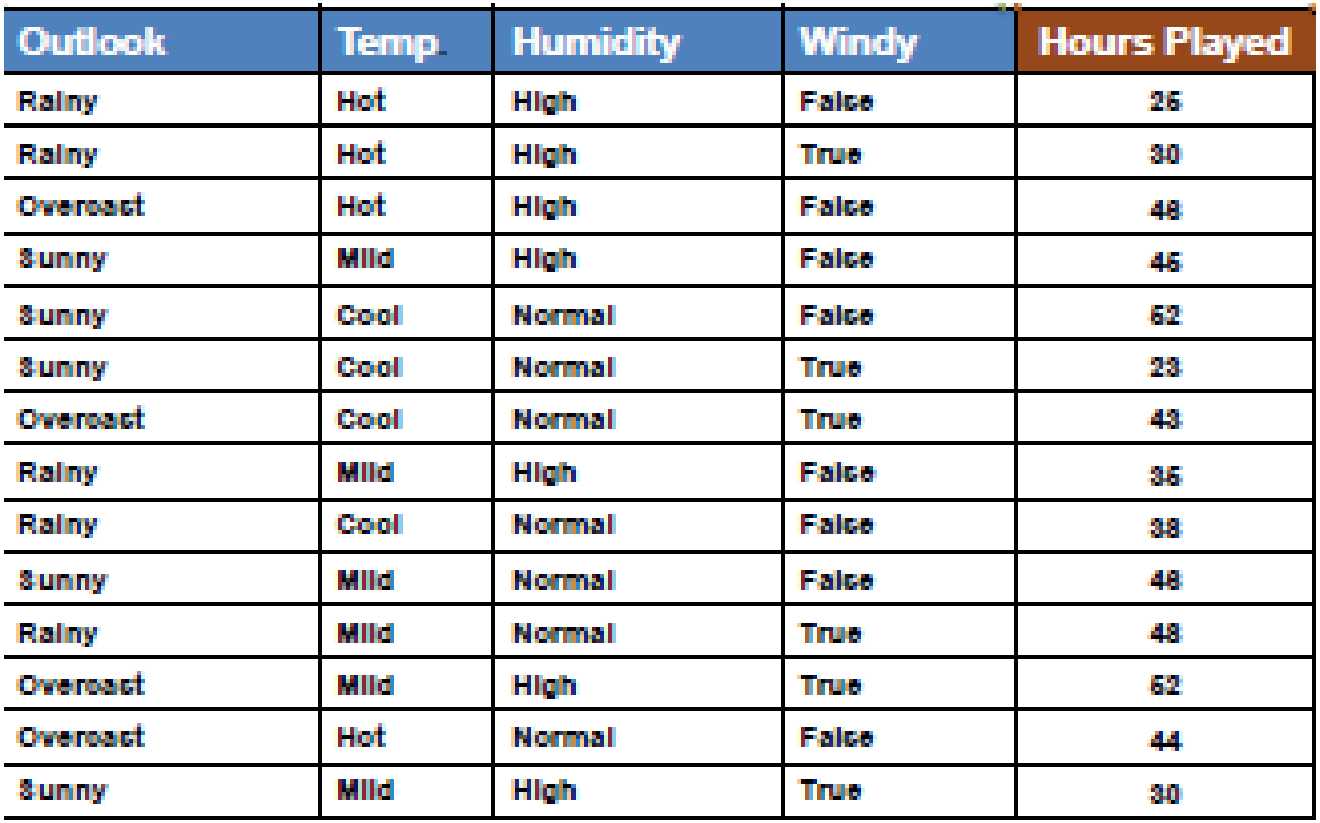
- For the cutting-point 0.32, we have



- The loss reduction is L = 0.6148 – 2/4*0.0032 – 2/4*0.0032 = 0.6116
- Thus we shall use x1 <= 0.32 for splitting (Note we have not tested other cutting-points and the features x2 and x3 yet).

Splitting for Categorical Feature

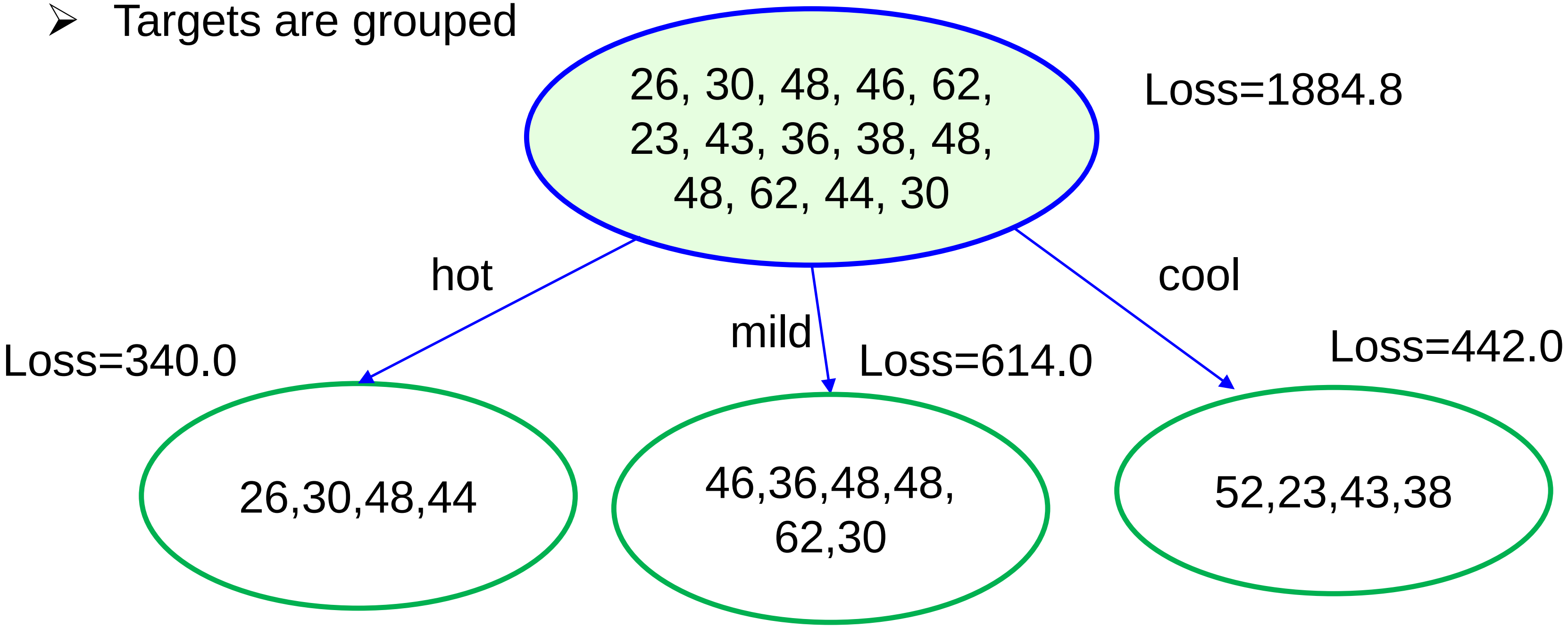
- This can be done as in ID3 algorithm for classification. Simply split the data into K groups if the feature takes K different category values: Example of 14 cases



Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	46
Sunny	Mild	High	False	45
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

Splitting for Categorical Feature

- Consider the feature Temp as an example: K=3
- Targets are grouped



- The loss reduction is

  L = 1884.8 – 4/14 *340.0 – 6/14*614.0 – 4/14*442.0 = 1398.2

- Do this for other features and compare the loss reduction.

Algorithm Summary

Step 1: In order to perform recursive binary splitting, first select the feature x_j and the cut-points such that splitting the feature space into the regions $\{\mathbf{x}|\ x_j < \theta\}$ and $\{\mathbf{x}|\ x_j \geq \theta\}$ leads to the **greatest possible reduction** in loss function.

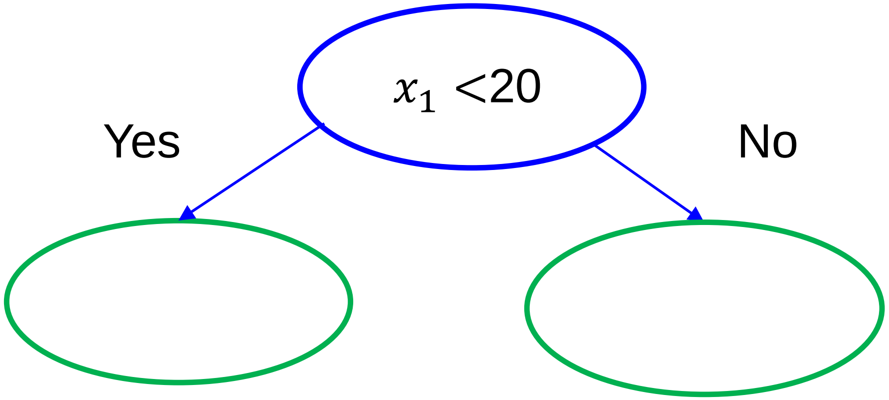
For any j and s, the pair of half-planes is defined as:

$$R_1(j,\theta) = \{\mathbf{x} \mid x_j < \theta\} \quad \text{and} \quad R_2(j,\theta) = \{\mathbf{x} \mid x_j \geq \theta\}$$

$$\sum_{n:\mathbf{x}_n \in R_1(j,\theta)} \left(t_n - \hat{t}_{R_1}\right)^2 + \sum_{n:\mathbf{x}_n \in R_2(j,\theta)} \left(t_n - \hat{t}_{R_2}\right)^2$$

where $\hat{t}_{R_1}$ is the mean response for the training examples in $R_1(j,\theta)$, and $\hat{t}_{R_2}$ is the mean response for the training examples in $R_2(j,\theta)$.

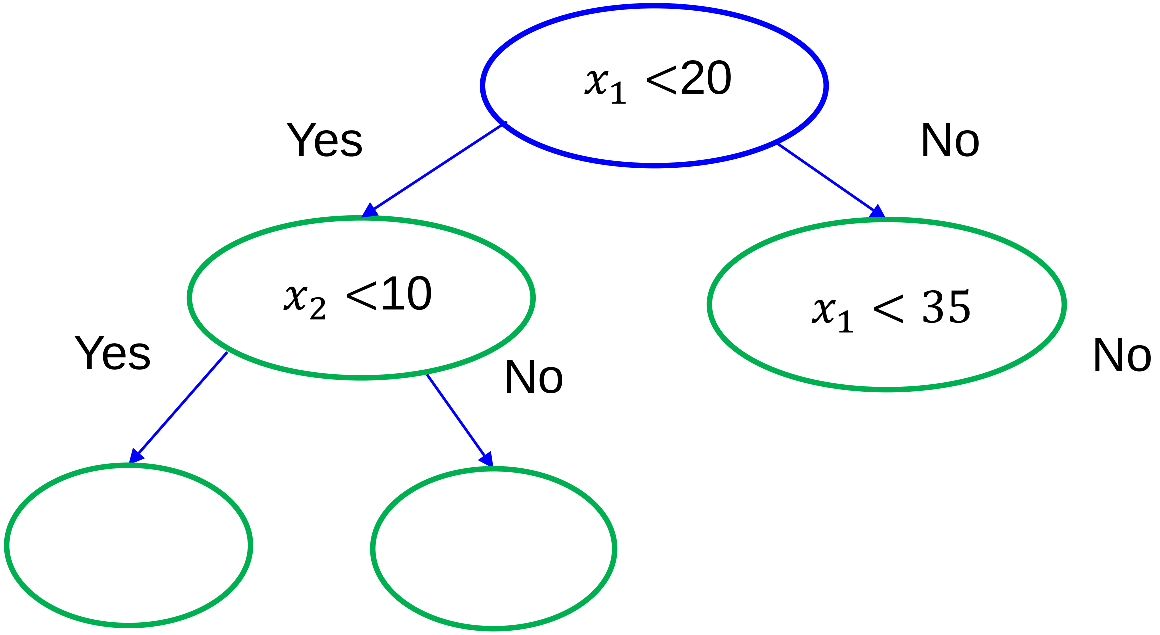
Example Result



- Repeat the process: get the **best** predictor and **best** cutpoint in order to split the data further so as to minimize the loss function within each of the resulting regions.

- Instead of splitting the entire predictor space, we split **one of the two previously identified regions (using same rule as previous slide)**

- Then we will have **three regions.**



- For one of these three regions, split again to minimize the loss function.
- The process continues until a **stopping criterion** is reached, e.g. no region contains more than five observations.
- Once the regions $R_1, R_2, \ldots, R_J$ have been created, the response for a given test example is predicted by using the **mean** of the training example in the region to which that test example belongs.

Postpruning

- The process described before may produce good predictions on the training set, but is likely to overfit the data, leading to poor validatin/test set performance.
- The resulting tree might be too complex.
- Postpruning is a strategy to grow a very large trees, and then prune it back in order to obtain a subtree.

- **Best way of pruning?**
  - Our goal is to select a subtree that leads to the lowest test loss.
  - Given a subtree, we can estimate its test loss using cross-validation or the train/validation/test sets approach.

Tree Pruning Objective

For each value of nonnegative tuning parameter λ, there is a subtree T that minimize the following loss function:

$$\sum_{p=1}^{T} \sum_{n:\, \mathbf{x}_n \in R_p} \left(t_n - \hat{t}_{R_p}\right)^2 + \lambda\,|T|$$

Here $|T|$ indicates the number of leaf nodes of the tree *T*, R_p is the region (e.g. the subset of feature space) corresponding to the p_{th} leaf node, and $\hat{t}_{R_p}$ is the predicted response associated with R_p: the **mean** of the training observations in R_p.

The tuning parameter λ controls a trade-off between the subtree's complexity and fitting to the training examples.

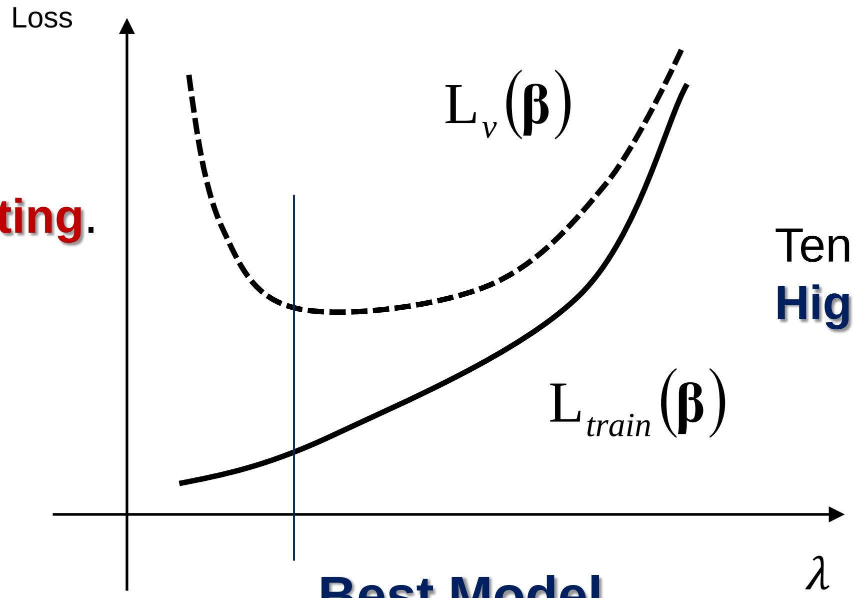
How to choose λ?

The loss function in the previous slide has similar structure compared with the regularized regressions.

Parameter λ, which plays the similar role as the regularization parameter of regularized regressions, controls a trade-off between the tree’s complexity and fitting to the training examples.

So we select λ using the cross validation strategy as in the regularized regressions.

Tend to **Overfitting**.
High variance.



Tend to **underfitting**.
High bias.

Best Model

Algorithm

1) Use recursive binary splitting to grow a large tree on the training data, stopping only when a stopping criteria is met, e.g. each leaf node has than some minimum number of observations.

2) Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of λ.

3) Use K-fold cross-validation to choose λ. Dividing the training examples into K folds. For each $k = 1, \dots, K$:
   (3.1) Repeat Steps 1 and 2 on all except the k_{th} fold of the training data.
   (3.2) Evaluate the prediction loss on the data in the k_{th} fold, as a function of λ.
   (3.3) Pick λ that minimizes the average loss.

4) Return the subtree from Step 2 that corresponds to the chosen value of λ.

The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with *Standard Deviation Reduction*.

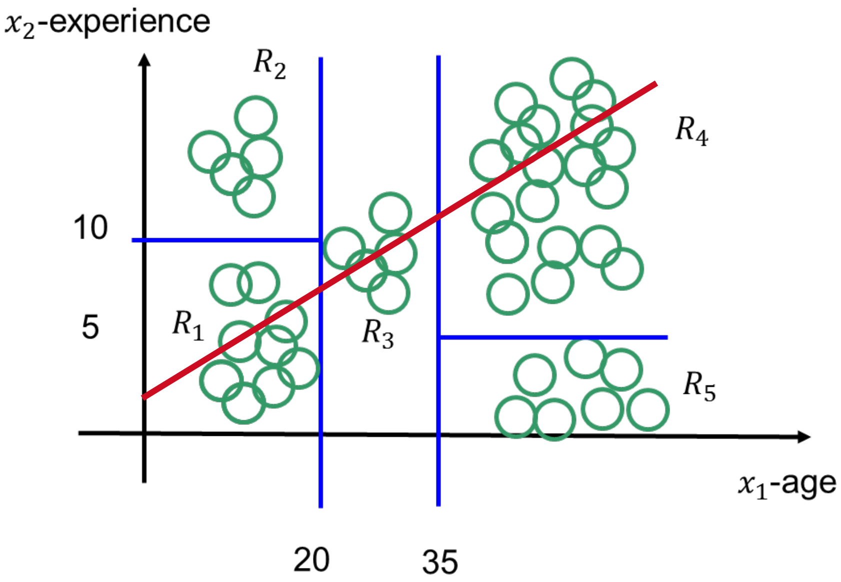
James et al.,(2014),

❑ Advantages of trees:

- ➢ Learning and classification is fast
- ➢ Decision trees closely mirror human decision making process
- ➢ Decision-making trees are easy to interpret as sets of decision rules
- ➢ There is no black box in the algorithm
- ➢ Often, trees can be used as a benchmark before more complicated algorithms are attempted
- ➢ …

❑ Disadvantages of trees:

- Trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches, especially regression tree
- Can only do axis aligned split of data (may generalise)
- Trees can be non-robust



x_2-experience
R_2
R_4
10
5
R_1
R_3
R_5
x_1-age
20
35