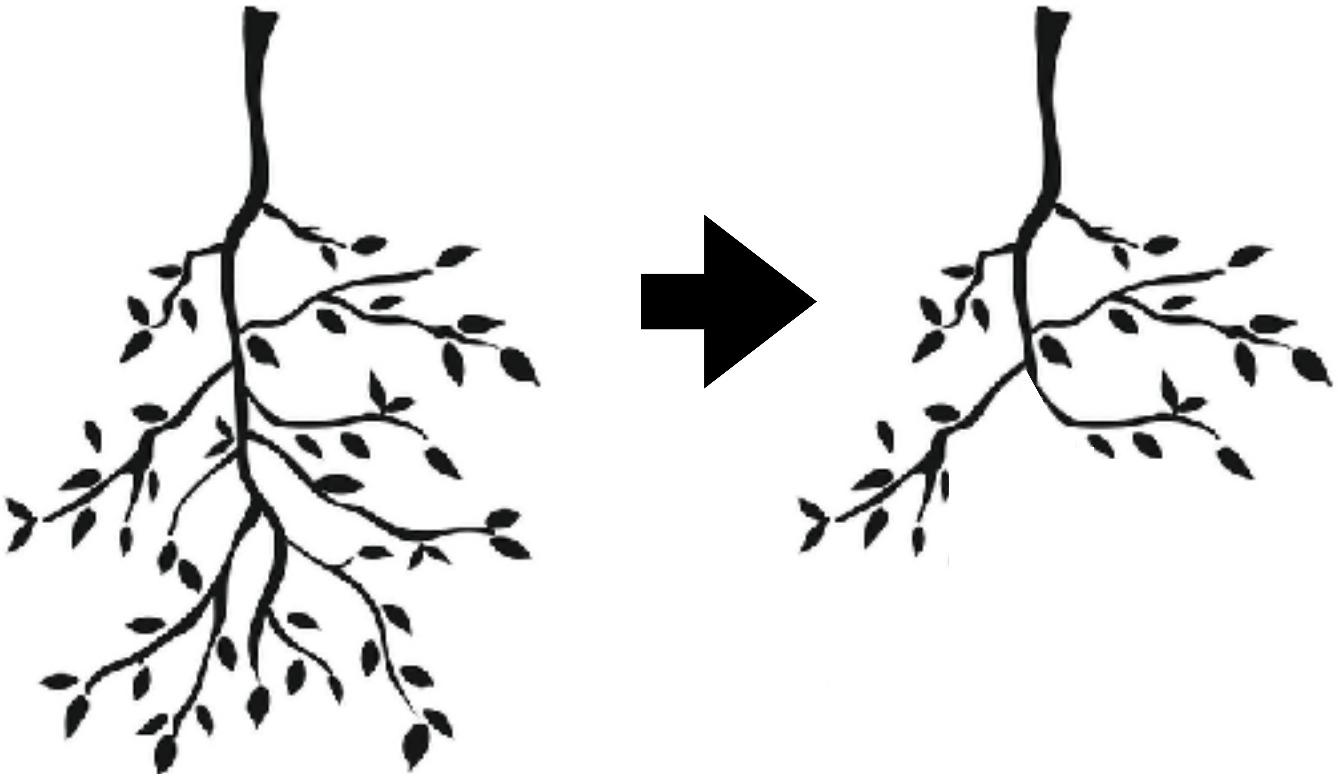
CO3091 - Computational Intelligence and Software Engineering

Lecture 23



Decision Trees — Part III

Leandro L. Minku

Overview

- Previous lectures:
  - What decision trees are in the context of machine learning?
  - How to build classification and regression trees with categorical or ordinal input attributes.

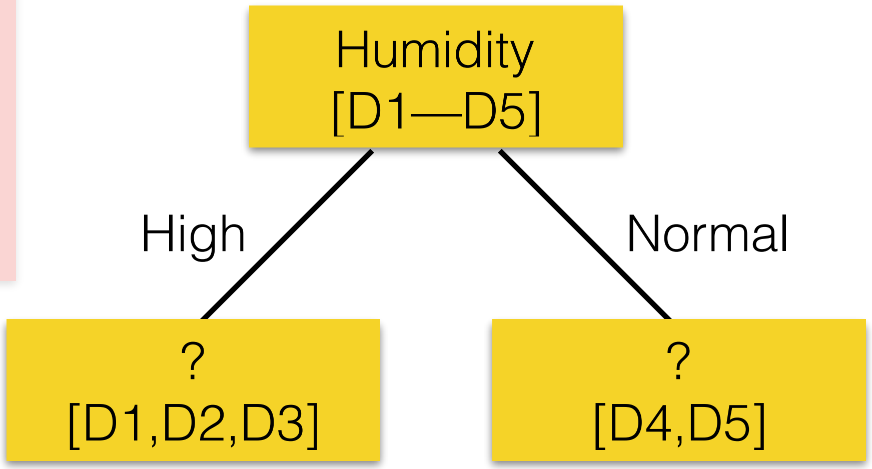
- This lecture:
  - How to deal with numerical input attributes?
  - How to deal with missing attribute values?
  - How to avoid overfitting?
  - Applications of decision trees.

Overview

- Previous lectures:
  - What decision trees are in the context of machine learning?
  - How to build classification and regression trees with categorical or ordinal input attributes.
- This lecture:
  - **How to deal with numerical input attributes?**
  - How to deal with missing attribute values?
  - How to avoid overfitting?
  - Applications of decision trees.

Branches for Categorical or Ordinal Input Attributes

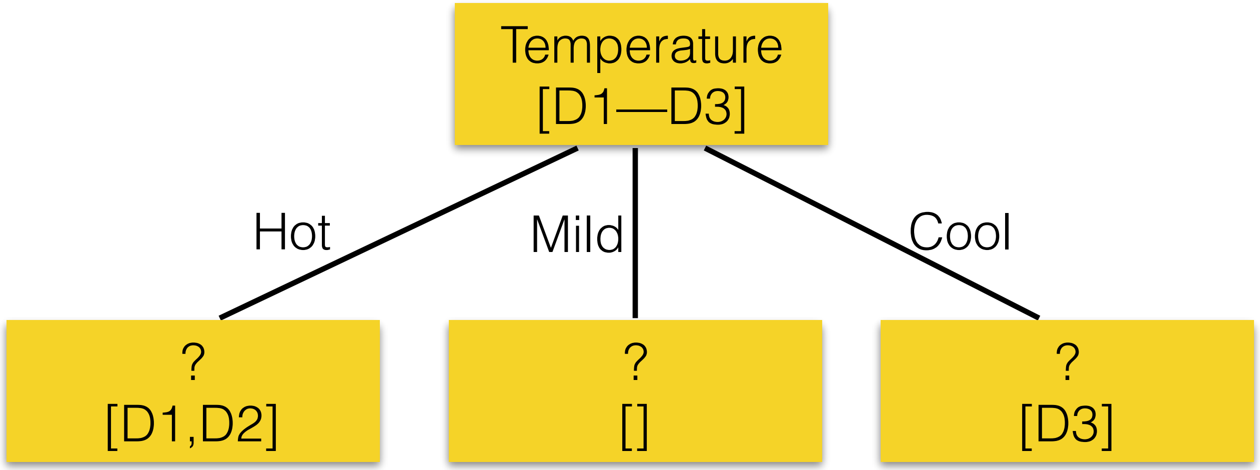
Day	Wind	Humidity	Play
D1	Strong	High	No
D2	Strong	High	No
D3	Weak	High	No
D4	Strong	Normal	Yes
D5	Strong	Normal	Yes



Branches for Categorical or Ordinal Input Attributes

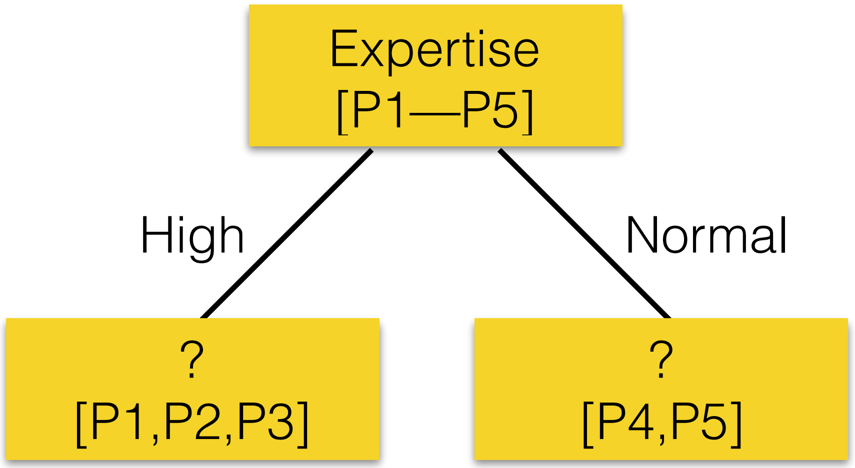
Day	Temperature	Play
D1	Hot	No
D2	Hot	No
D3	Cool	Yes

Temperature ∈ {hot, mild, cool}



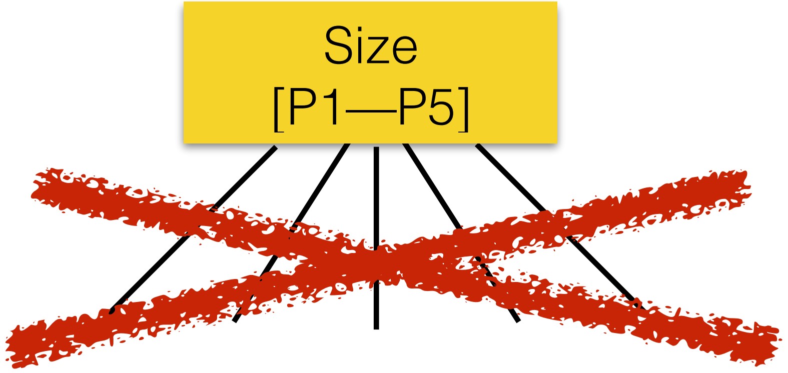
Branches for Categorical or Ordinal Input Attributes

Project	Expertise	Effort
P1	High	100
P2	High	110
P3	High	90
P4	Normal	500
P5	Normal	550



Branches for Numerical Input Attributes

Project	Size	CPU	Effort
P1	15	50	10
P2	150	70	496
P3	13	30	6
P4	160	100	510
P5	165	40	500

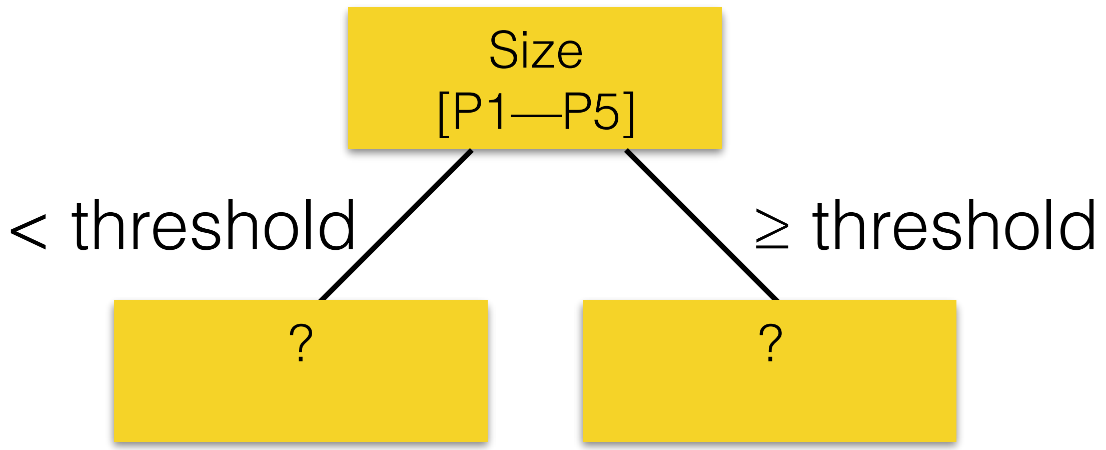


Creating a branch for each possible numerical value is infeasible! Too many different possible values.

How Many Branches To Create?

Most algorithms will create two branches, which separate data according to the numerical input attribute based on a threshold.

Project	Size	CPU	Effort
P1	15	50	10
P2	150	70	496
P3	13	30	6
P4	160	100	510
P5	165	40	500



Potential Thresholds

Given an input attribute to be investigated for a split

Project	Size	CPU	Effort
P1	15	50	10
P2	150	70	496
P3	13	30	6
P4	160	100	510
P5	165	40	500

Sort the examples based on this input attribute

Project	Size	CPU	Effort
P3	13	30	6
P1	15	50	10
P2	150	70	496
P4	160	100	510
P5	165	40	500

14

82.5

155

162.5

Potential thresholds

How to Decide Which Threshold to Use?

- When deciding the input attribute to split on, we consider numerical input attributes together with their potential thresholds.
- We calculate the information gain or the reduction in variance separately for each pair of numeric input attribute + threshold.
- We then choose the pair that provides the highest information gain / reduction in variance.

Reduction in Variance for Size Threshold 14

Project	Size	CPU	Effort
P3	13	30	6
P1	15	50	10
P2	150	70	496
P4	160	100	510
P5	165	40	500

14

82.5

155

162.5

$$\text{VarRed(examples,A)} = \text{Variance(examples)} - \sum_{v_i \in \text{Values(A)}} \frac{|\text{examples}_{vi}|}{|\text{examples}|} \text{Variance}(\text{examples}_{vi})$$

$$\text{VarRed(P1-P5,Size14)} = \text{Variance(P1-P5)} - \underbrace{\boxed{\frac{1}{5} \text{x Variance(P3)}}}_{v_i < 14} - \underbrace{\boxed{\frac{4}{5} \text{x Variance(P1–2,P4–5)}}}_{v_i \geq 14}$$

$$\text{VarRed(P1-P5,Size14)} = 58{,}591.04 - \frac{1}{5} \text{x } 0 - \frac{4}{5} \text{x } 45{,}413 = 22{,}260.64$$

Deciding on an Attribute to Split

- For numerical input attributes, calculate the information gain / reduction in variance for all pairs of numerical input attribute+threshold.
  - E.g.: Size14, Size82.5, Size155, Size162.5, CPU35, CPU45, CPU60, CPU85.
- For categorical or ordinal input attributes, calculate the information gain / reduction in variance as discussed in the previous lecture.
- Choose the input attribute (+threshold) that leads to the highest information gain / reduction in variance.

Overview

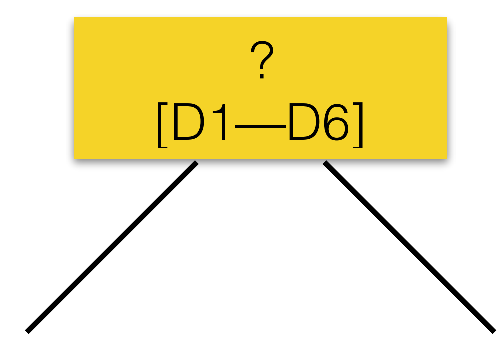
- Previous lectures:
  - What decision trees are in the context of machine learning?
  - How to build classification and regression trees with categorical or ordinal input attributes.

- This lecture:
  - How to deal with numerical input attributes?
  - **How to deal with missing attribute values?**
  - How to avoid overfitting?
  - Applications of decision trees.

Learning With Missing Input Attribute Values

In real world applications, there is frequently some missing input attribute values.

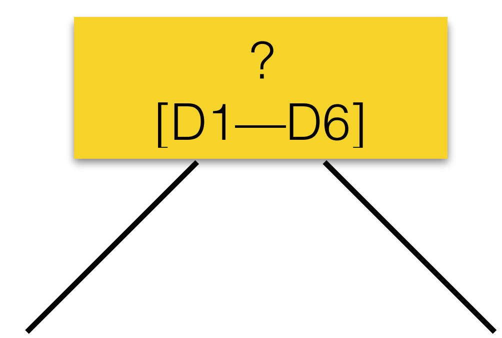
Day	Wind	Humidity	Play
D1	Strong	?	No
D2	Strong	High	No
D3	Weak	High	No
D4	Strong	Normal	Yes
D5	Strong	Normal	Yes
D6	Strong	Normal	Yes



Learning With Missing Input Attribute Values

In order to calculate information gain / reduction in variance, you can replace missing values by the most common attribute value among the examples in the node being split.

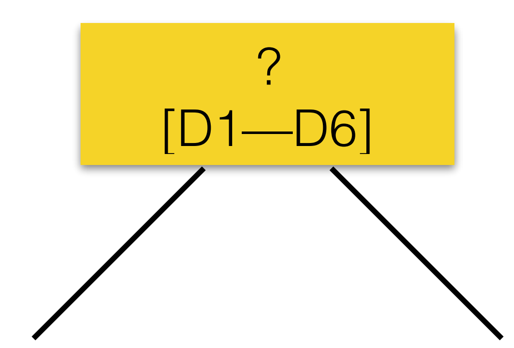
Day	Wind	Humidity	Play
D1	Strong	Normal	No
D2	Strong	High	No
D3	Weak	High	No
D4	Strong	Normal	Yes
D5	Strong	Normal	Yes
D6	Strong	Normal	Yes



Learning With Missing Input Attribute Values

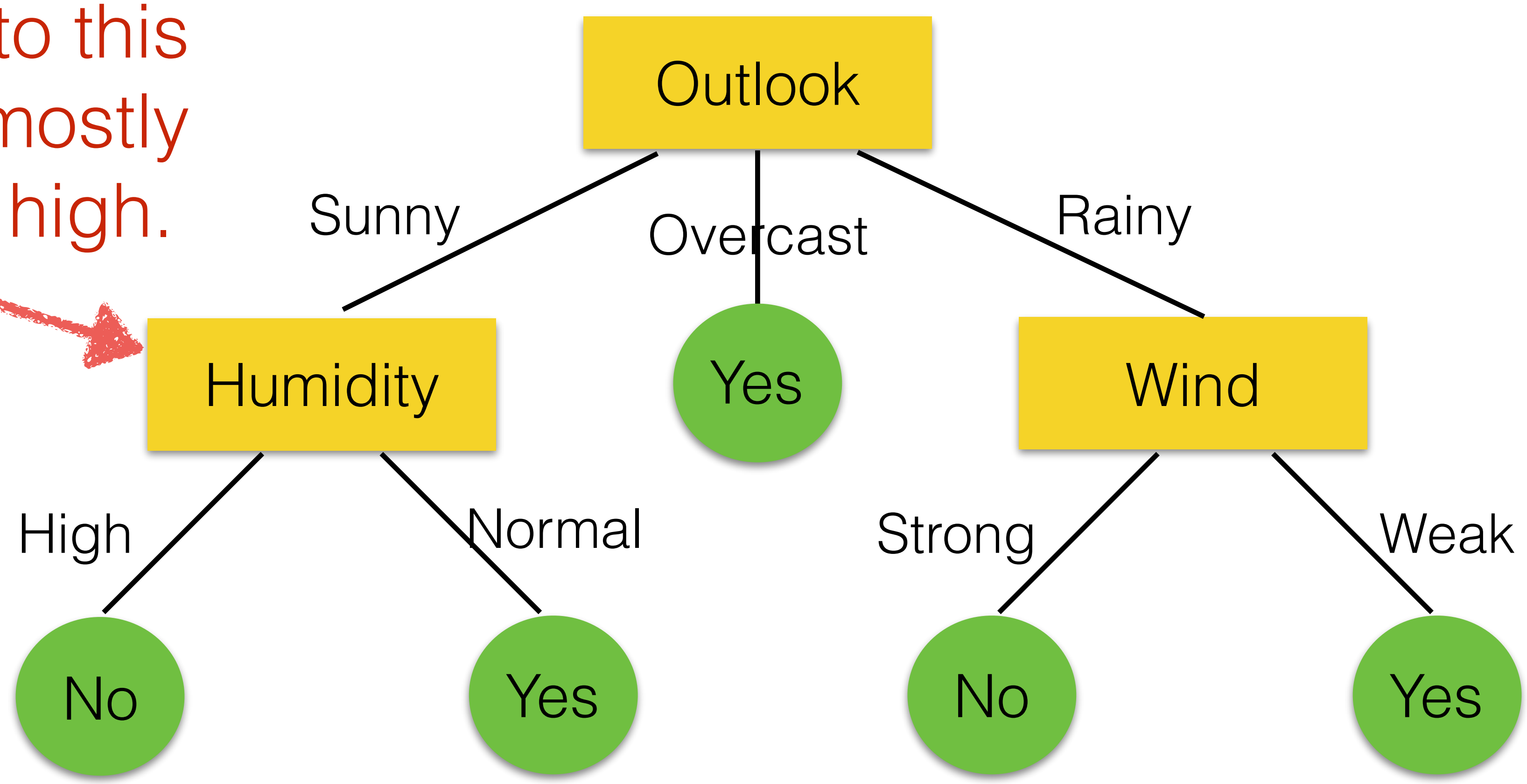
Or… you can replace missing values by the most common values among examples of the same class in the node.

Day	Wind	Humidity	Play
D1	Strong	High	No
D2	Strong	High	No
D3	Weak	High	No
D4	Strong	Normal	Yes
D5	Strong	Normal	Yes
D6	Strong	Normal	Yes



Predicting With Missing Input Attribute Values

Training examples associated to this node were mostly of humidity high.



So, we assume that this instance would also have humidity high.

What would be the prediction for an instance [outlook=sunny, temperature=hot, humidity=?,wind=strong]?

Overview

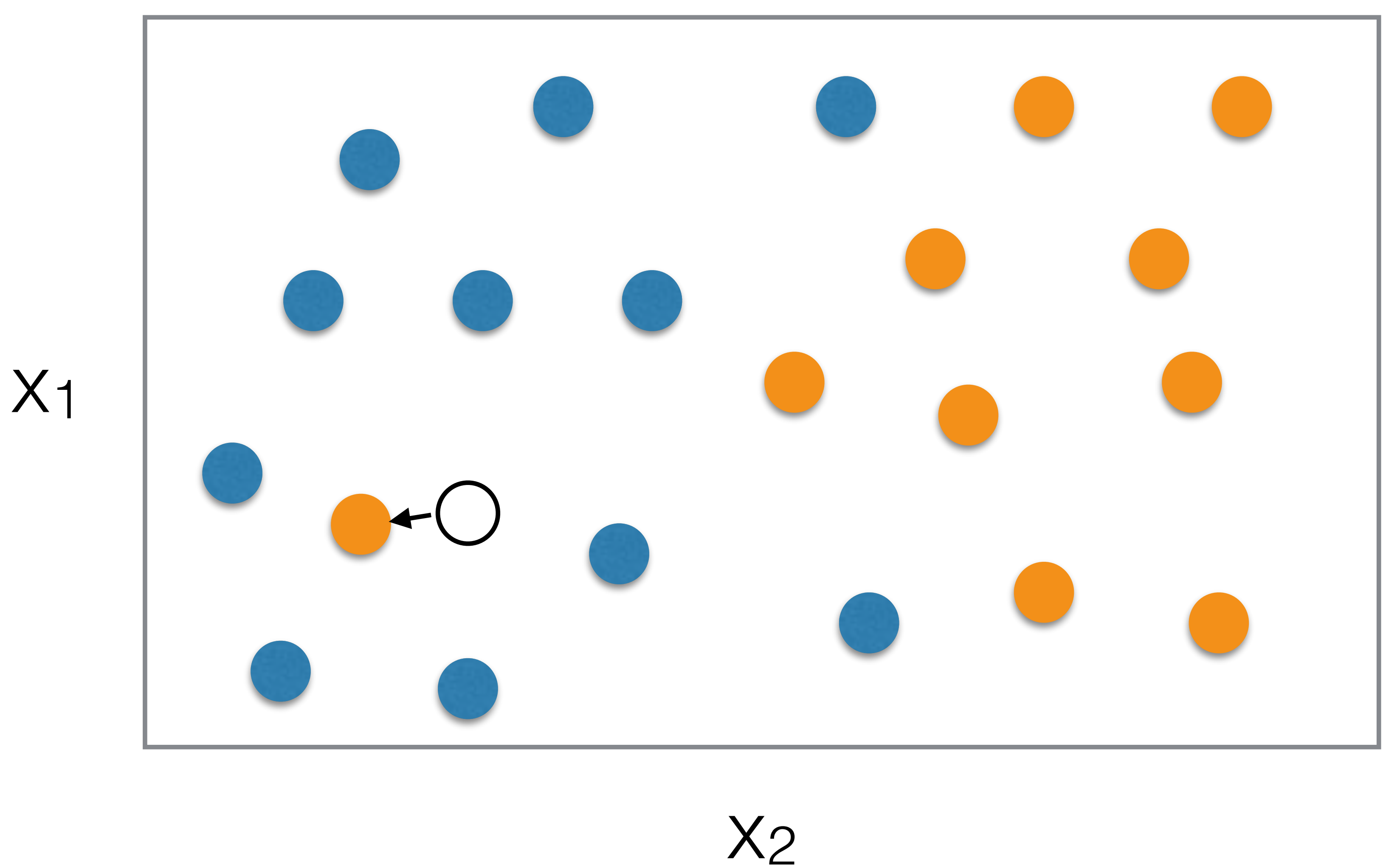
- Previous lectures:
  - What decision trees are in the context of machine learning?
  - How to build classification and regression trees with categorical or ordinal input attributes.
- This lecture:
  - How to deal with numerical input attributes?
  - How to deal with missing attribute values?
  - **How to avoid overfitting?**
  - Applications of decision trees.

Overfitting

- Decision trees will learn models that predict [almost] all training examples perfectly.
- This is similar to k-NN using k=1.
- Perfectly learning the training examples does not mean that the model will be able to generalise well to unseen data.
- How to reduce overfitting?

Reducing Overfitting

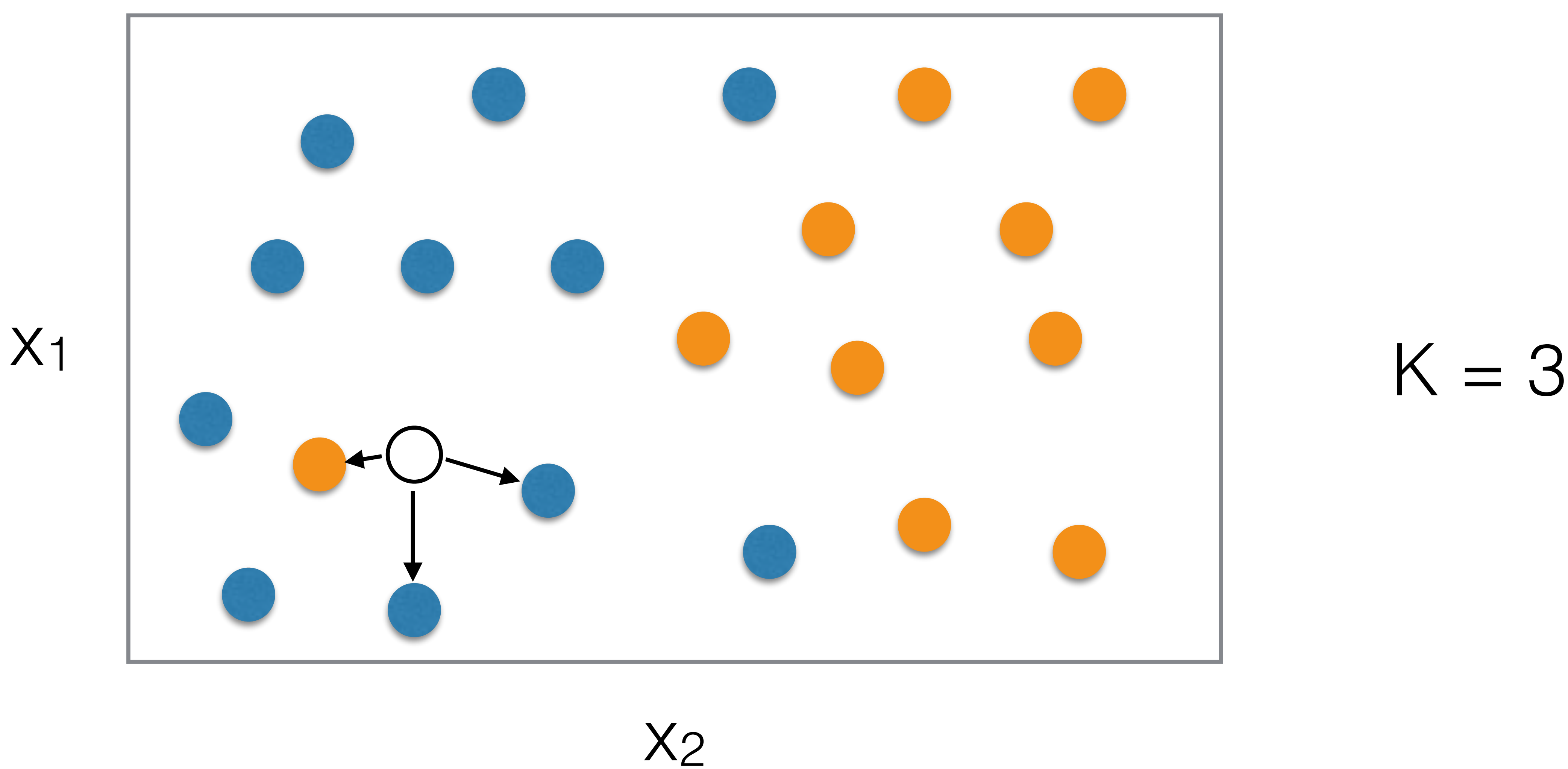
- In k-NN:
  - Predictions are made based on the k nearest neighbours.
  - Increasing k can help to reduce overfitting.
  - Increasing k too much can result in underfitting.



K = 1

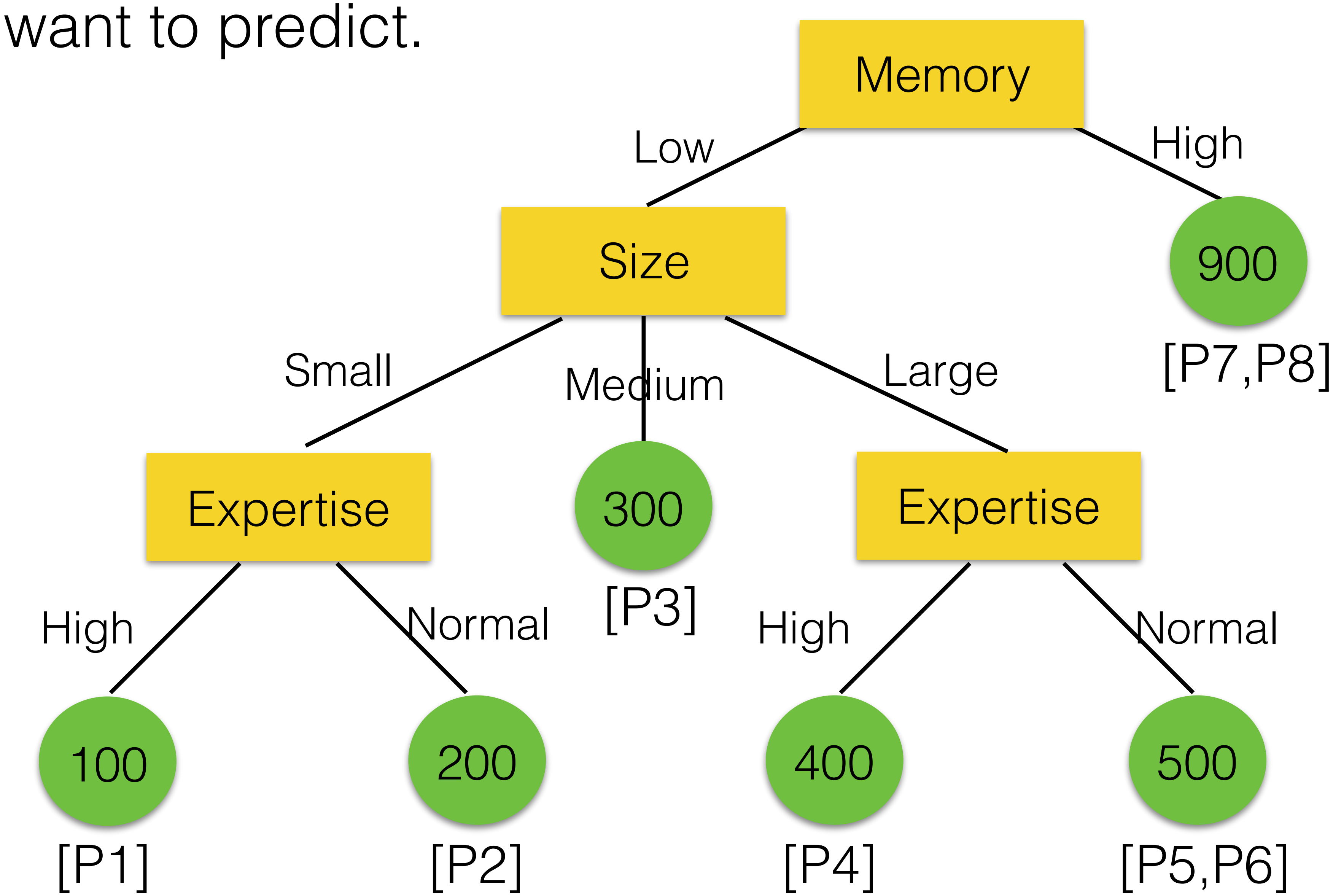
Reducing Overfitting

- In k-NN:
  - Predictions are made based on the k nearest neighbours.
  - Increasing k can help to reduce overfitting.
  - Increasing k too much can result in underfitting.



Reducing Overfitting

- In decision trees:
  - Predictions are made based on the training examples associated to the leaf node corresponding to the instance we want to predict.



When could we have more than one training example associated to a leaf?

When all examples have the same output value, or when there are no more input attributes to split on.

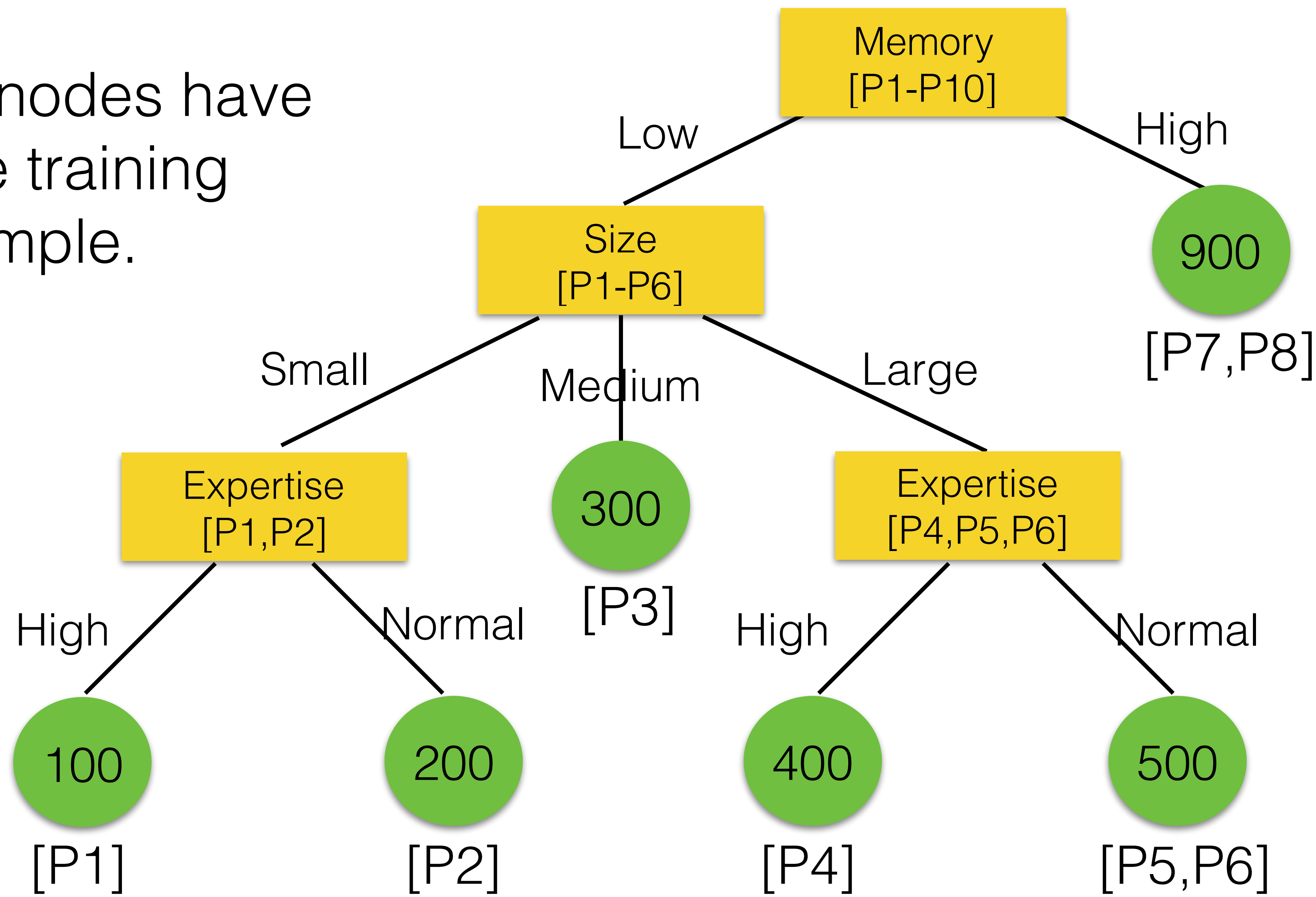
Reducing Overfitting

- We can create a parameter k to represent the minimum number of training examples associated to a leaf node.
- This works as an additional criterion to stop splitting a node:
  - If splitting the node would create children nodes with less than k training examples, stop splitting.
- Increasing k can help to reduce overfitting.
- Increasing k too much can result in underfitting.

Overfitted Decision Tree

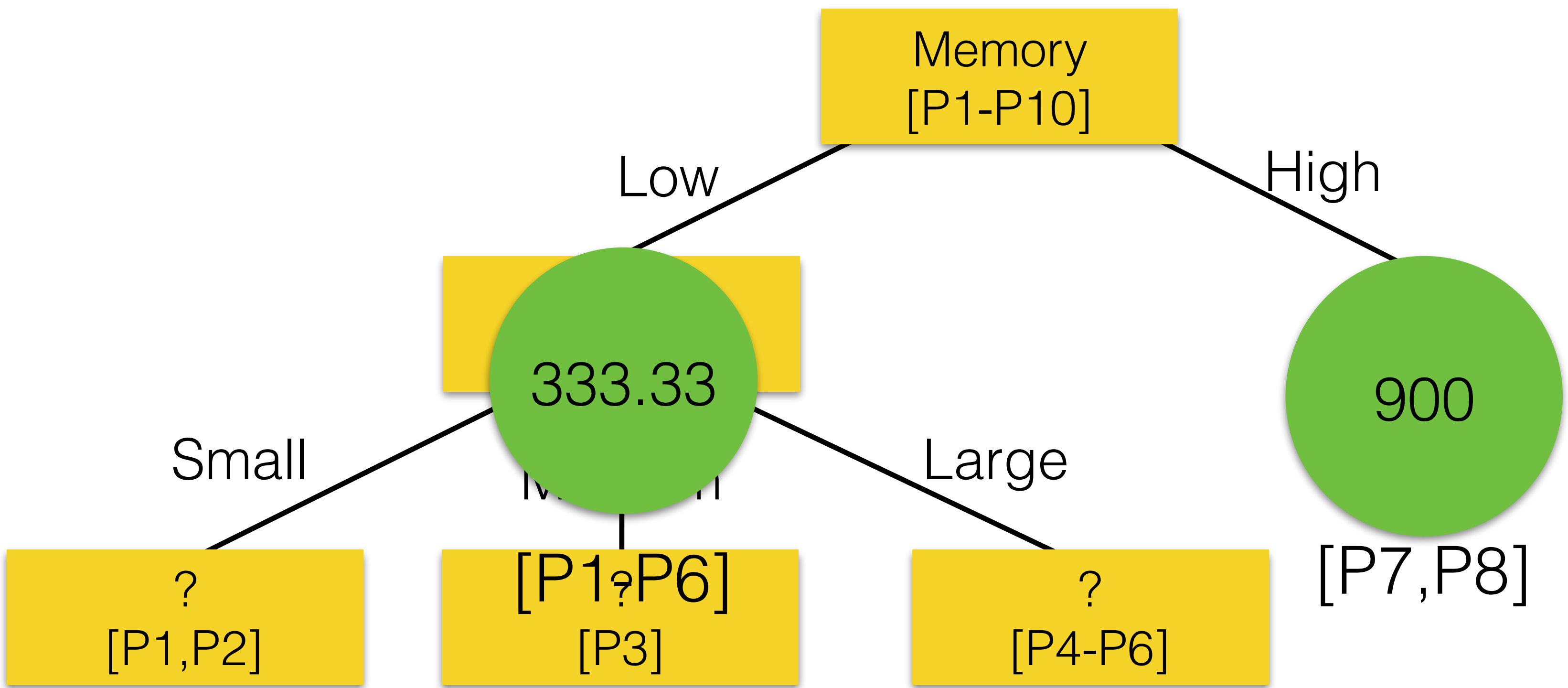
Many leaf nodes have a single training example.

k = 1



Reducing Overfitting

k = 2



Some splits are avoided, because they would lead to nodes with less than 2 training examples.

Reducing Overfitting

- We can also create a parameter D_{max} to represent the maximum depth of the tree.
- This can be seen as yet another stopping criterion.

How to Build Decision Trees Based on Training Data?

Stopping criteria for a node:

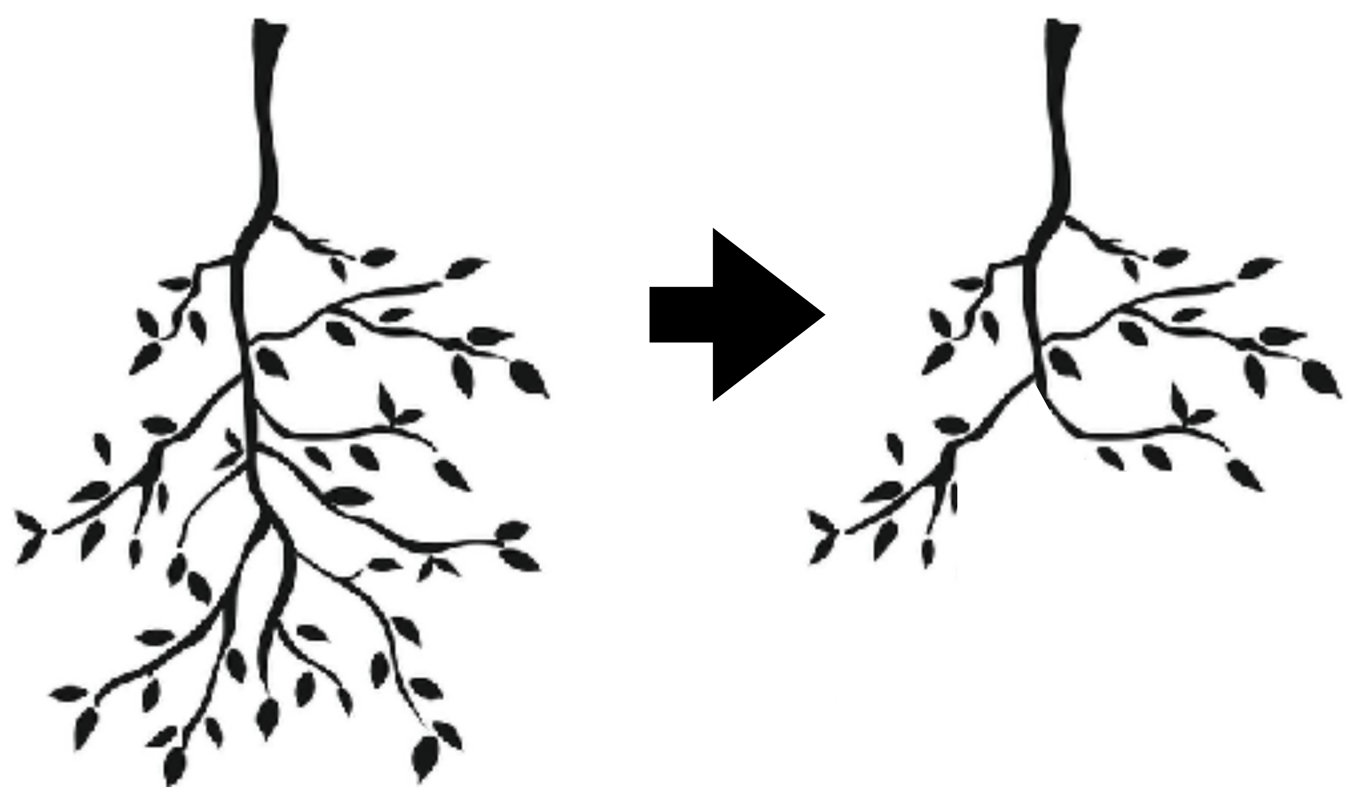
- All training examples associated to the node have the same output value.
- There are no further input attributes to split the node on.
- There are no examples associated to a node.

Additional stopping criteria for a node:

- **Minimum number of instances in a leaf node would be reached.**
- **Maximum depth of the tree is reached.**

Tree Pruning

- First build the tree without worrying about overfitting.
- Then, prune the tree, to reduce overfitting.



Reduced Error Pruning

- This pruning strategy will separate the training data into two sets.
  - One set is used for building the tree (training set).
  - The other set is used for pruning the tree (validation set).

Reduced Error Pruning

1. Build the tree using the training set.

2. For each internal node:

    a. Calculate the validation error on the tree as it is.
    b. Calculate the validation error on the tree without the subtree rooted at that node.

        - That means that that node would become a leaf node, using the majority vote or average among the outputs of the examples in that node.

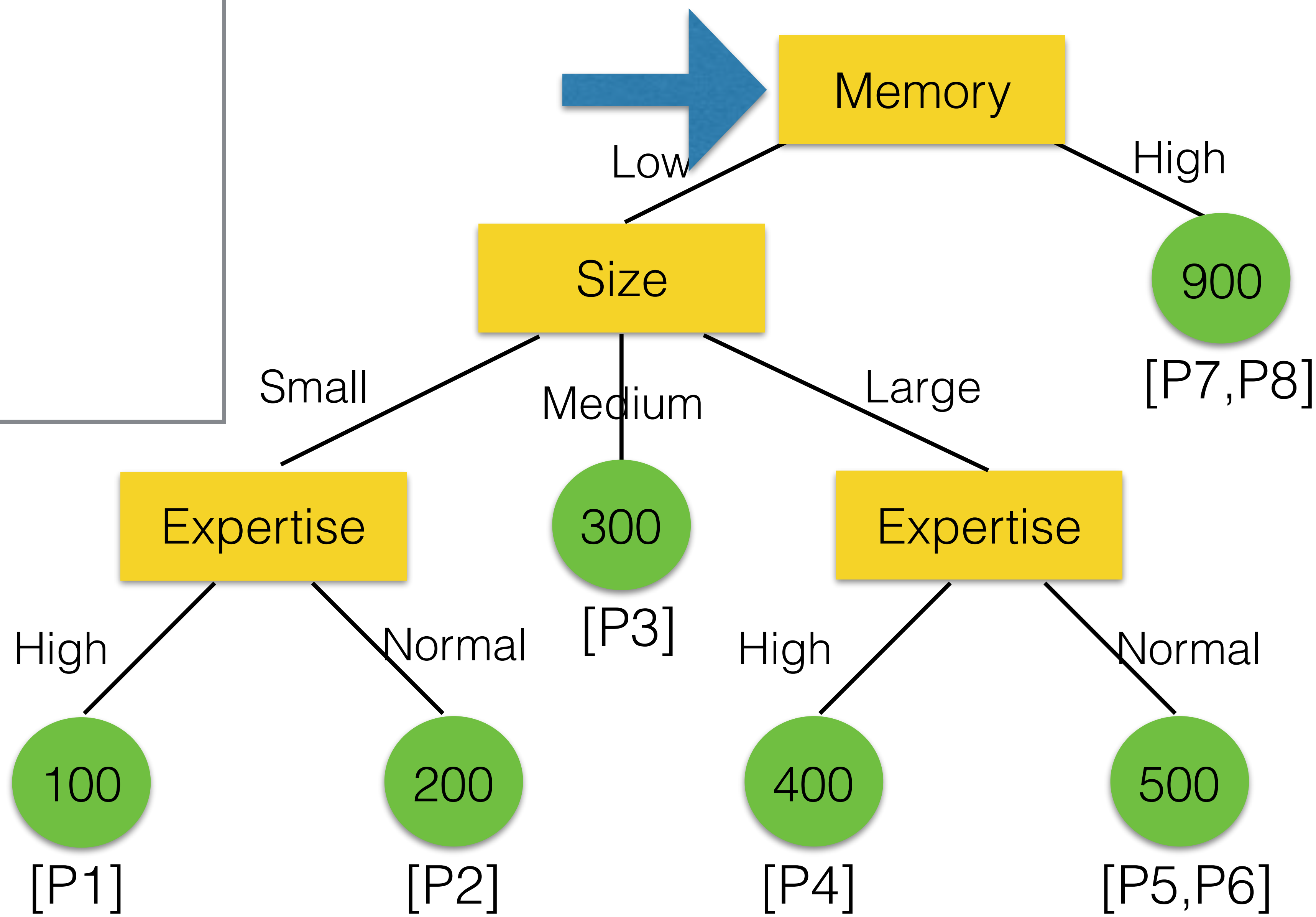
3. Choose the node corresponding to the smallest validation error.

4. If this validation error is smaller than the validation error of the tree with this node

    a. Permanently make that node a leaf node and go to 2.

Validation error = 10

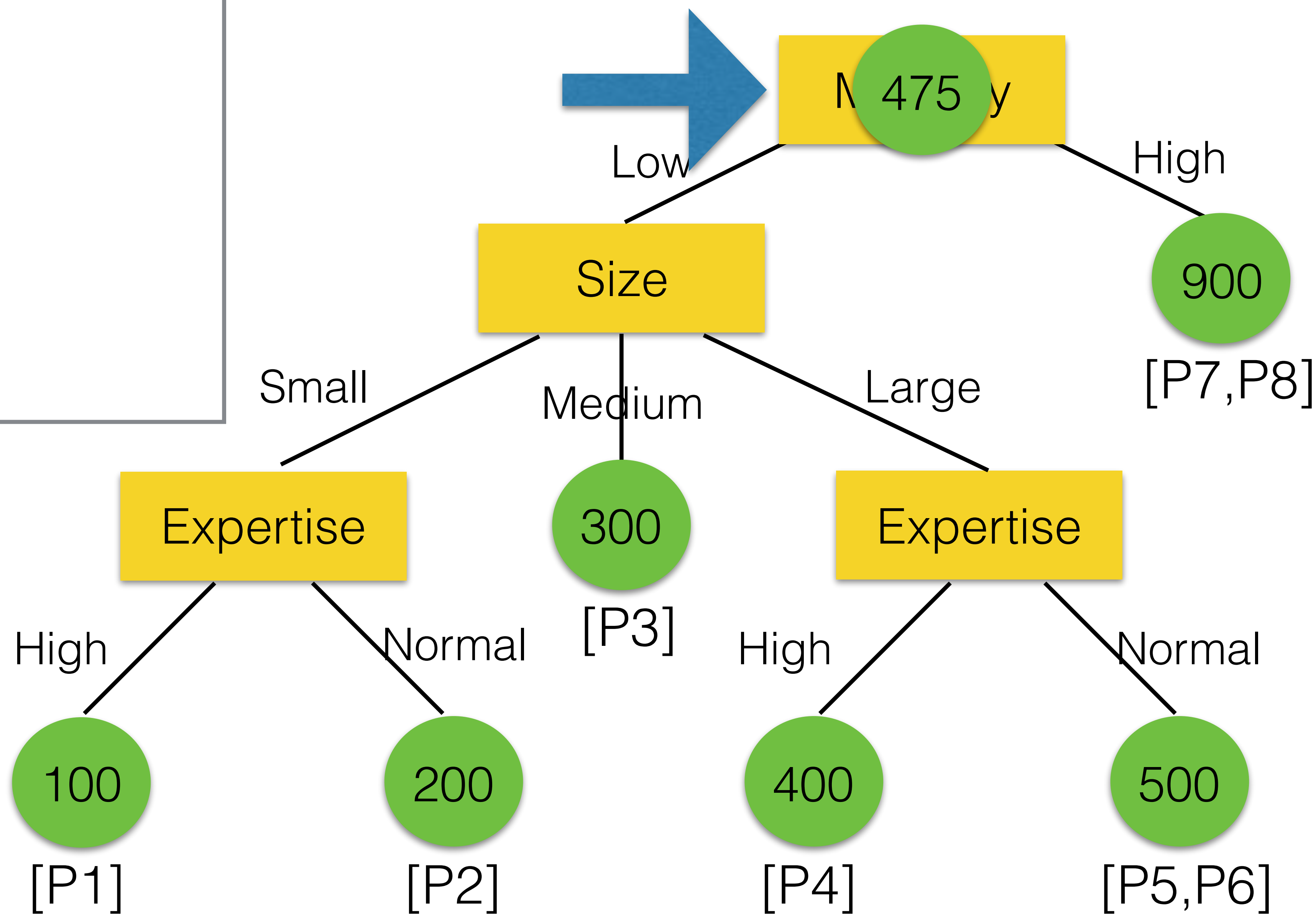
Example



Validation error = 10

Validation error without Memory node = 100

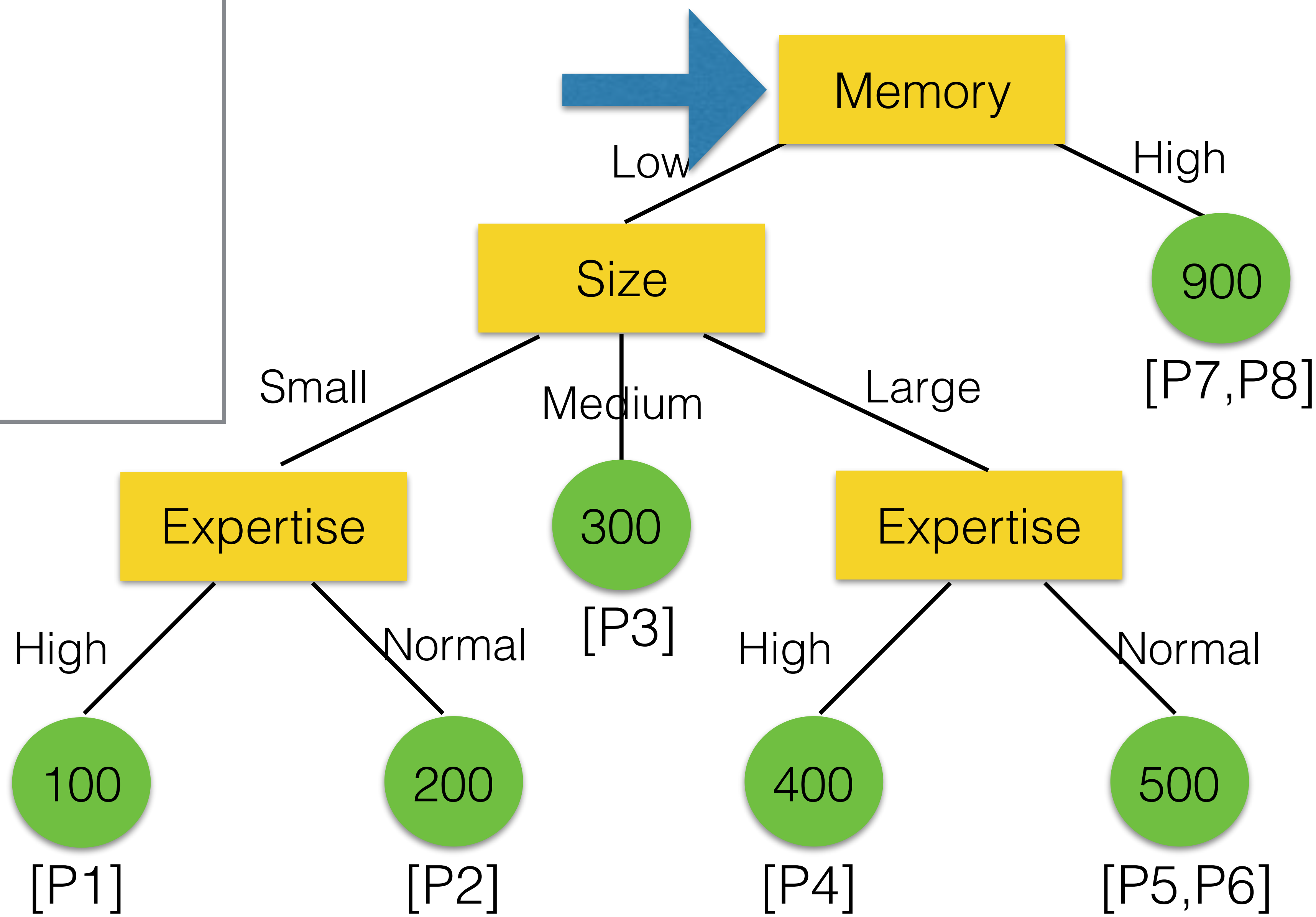
Example



Validation error = 10

Validation error without Memory node = 100

Example

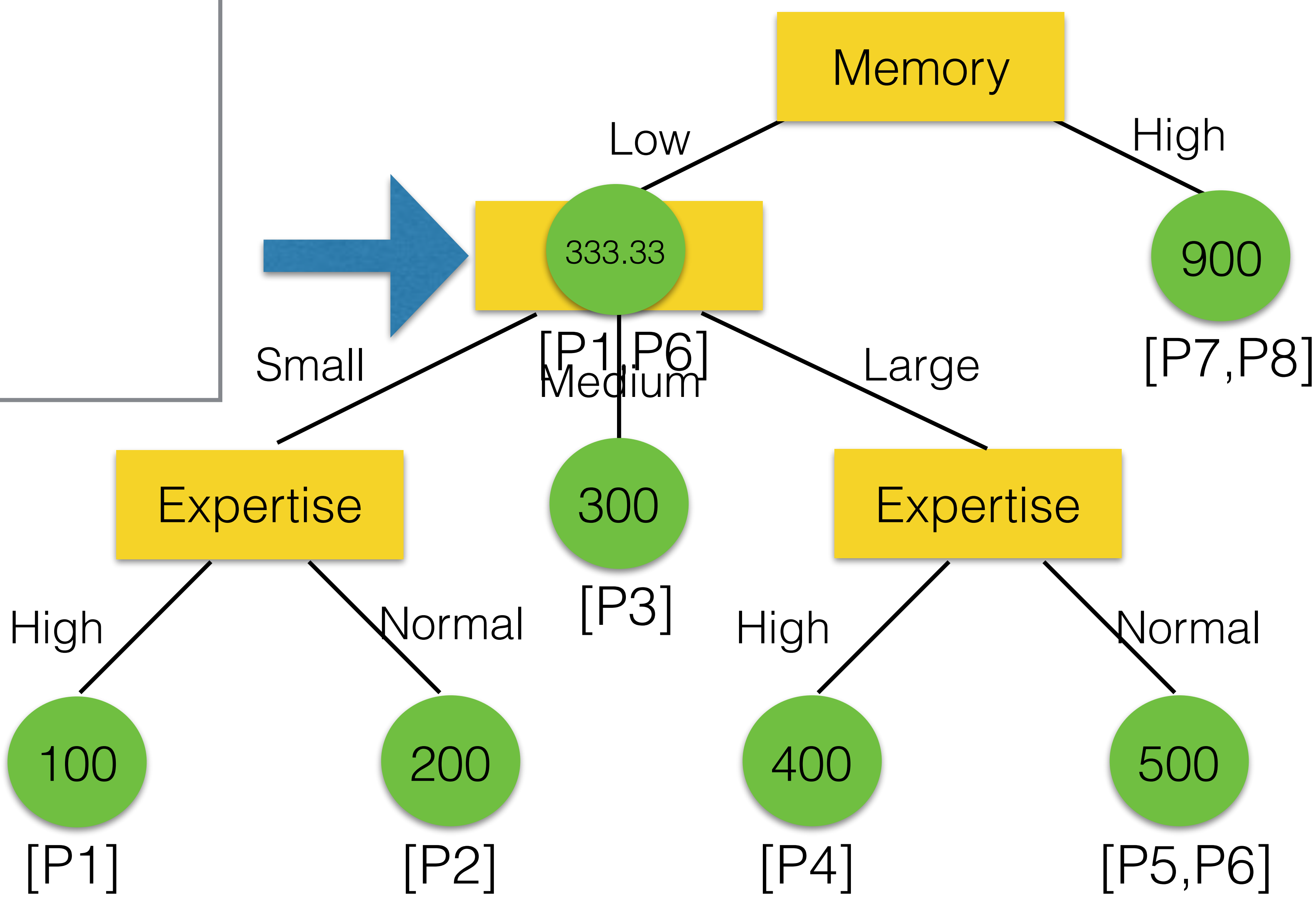


Validation error = 10

Validation error without Memory node = 100

Validation error without Size node = 8

Example



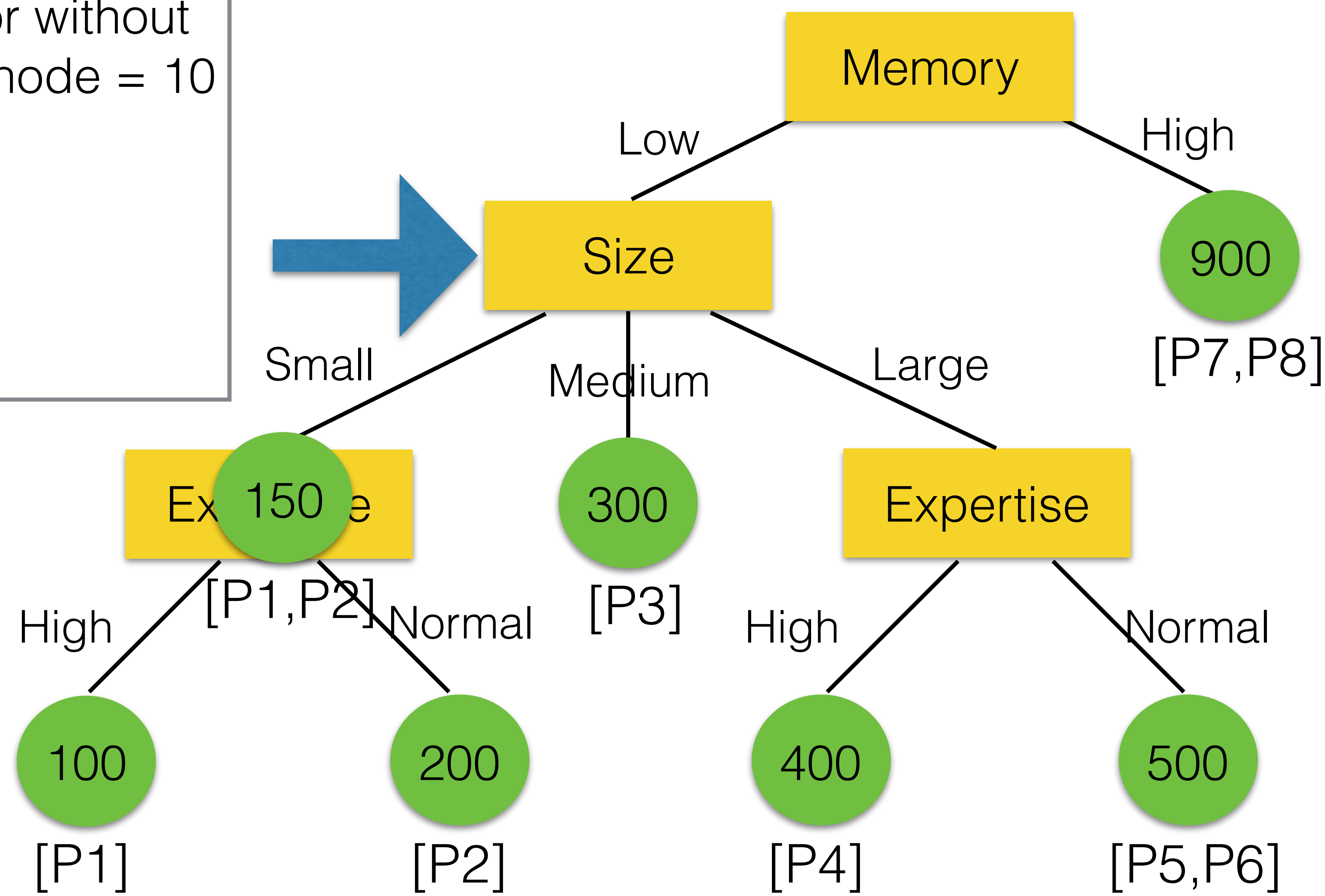
Validation error = 10

Validation error without Memory node = 100

Validation error without Size node = 8

Validation error without Expertise left node = 10

Example



Example

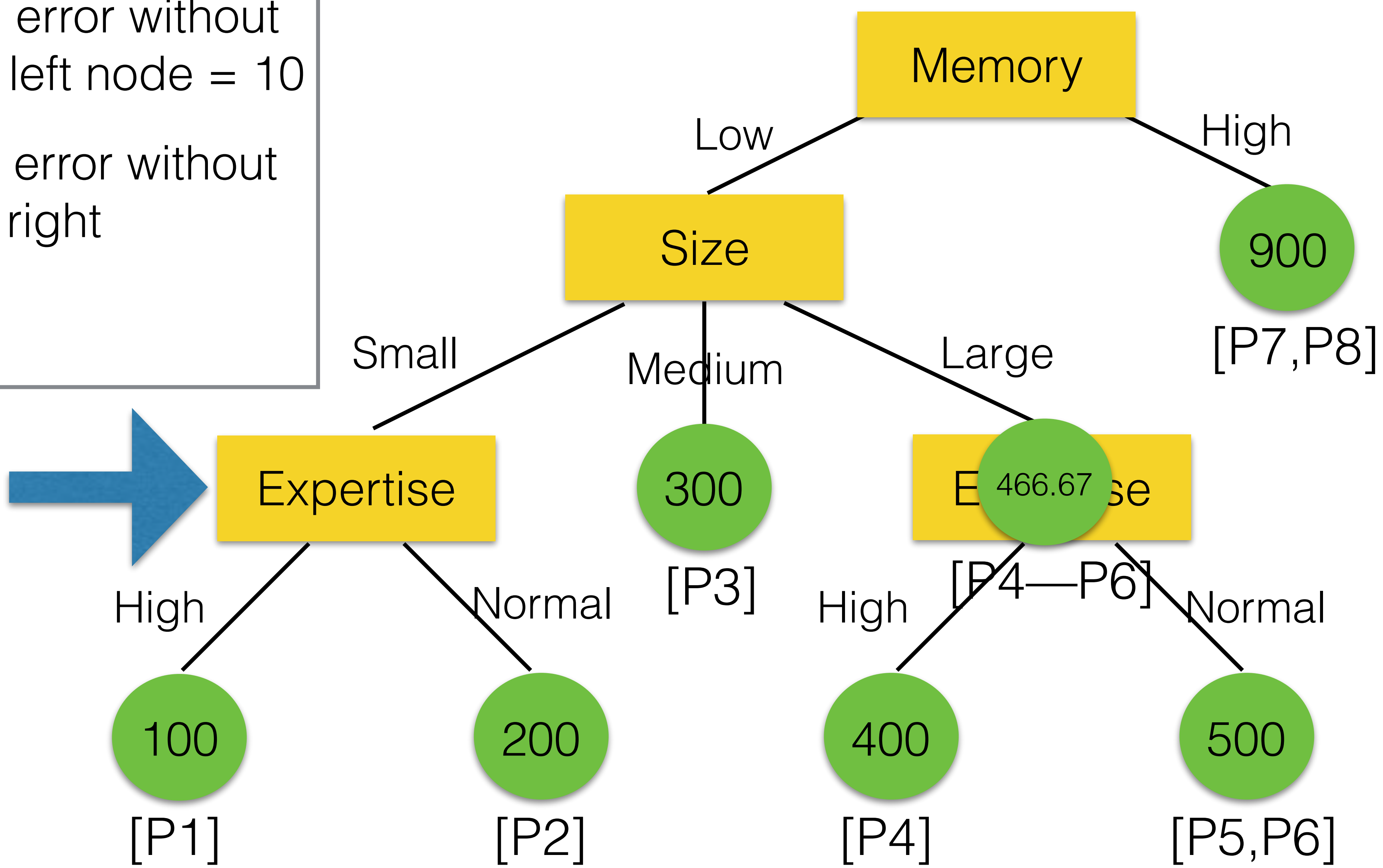
Validation error = 10

Validation error without Memory node = 100

Validation error without Size node = 8

Validation error without Expertise left node = 10

Validation error without Expertise right node = 9



Validation error = 10

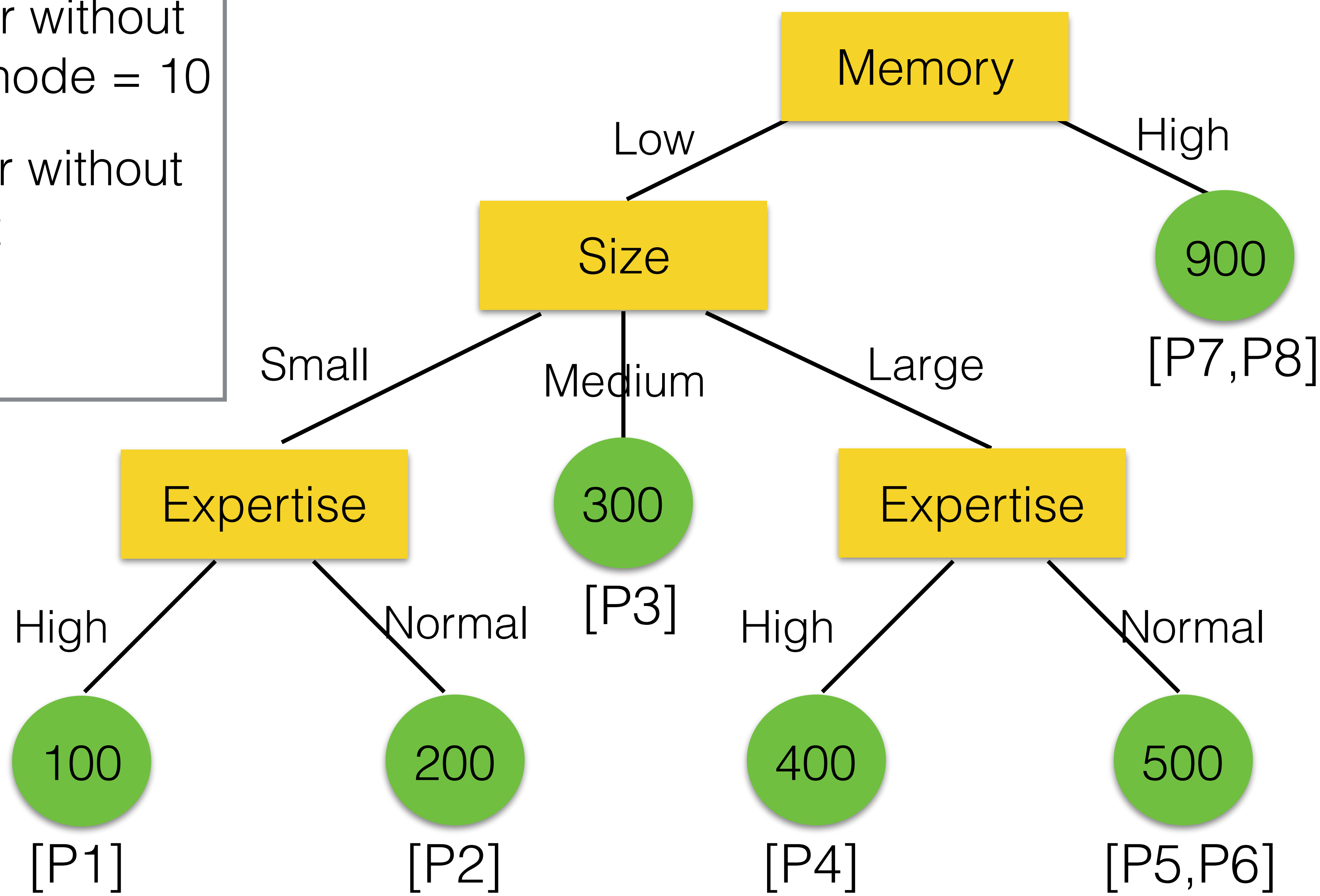
Validation error without Memory node = 100

Validation error without Size node = 8

Validation error without Expertise left node = 10

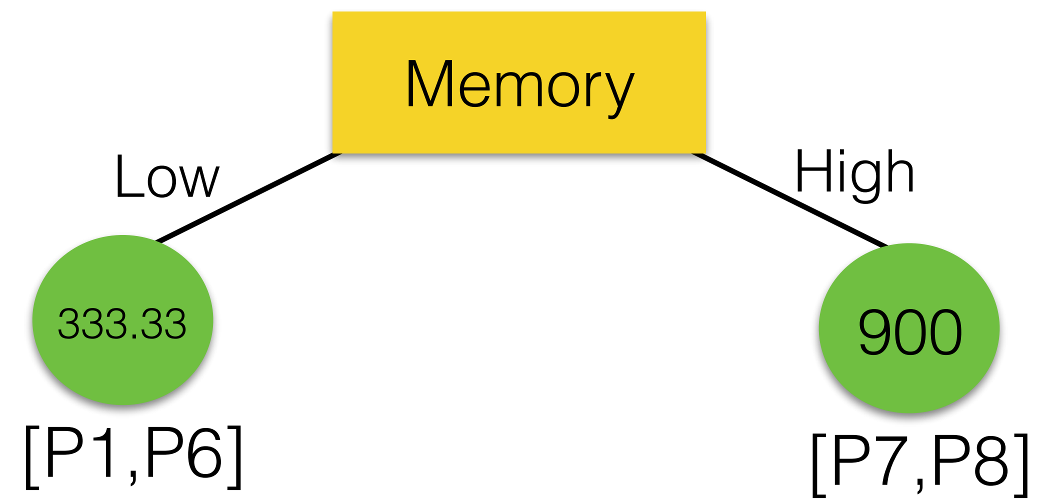
Validation error without Expertise right node = 9

Example



Validation error = 8

Example



Permanently remove the subtree.

Overview

- Previous lectures:
  - What decision trees are in the context of machine learning?
  - How to build classification and regression trees with categorical or ordinal input attributes.

- This lecture:
  - How to deal with numerical input attributes?
  - How to deal with missing attribute values?
  - How to avoid overfitting?
  - **Applications of decision trees.**

Applications

- Generally, decision trees are best suited for problems where:
  - attribute values are discrete (categorical or ordinal),
  - interpretable models are needed,
  - data may contain noise,
  - data may contain missing values.

- Decision trees have been successfully applied to a wide range of applications, e.g.:
  - classify medical patients by their diseases,
  - equipment malfunctions by their causes,
  - loan applicants by their likelihood of defaulting on payments,
  - software effort estimation.

Further Reading

Tom Mitchell

Machine Learning

London : McGraw-Hill, 1997

Chapter 3

http://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf

Menzies et al.

Sharing Data and Models in Software Engineering

Elsevier, 2014

Section 10.10 (Extensions for Continuous Classes)

http://www.sciencedirect.com/science/book/9780124172951