BT2101 Tutorial 1
Post-Pruning

Pruning

- Why do we need to prune the decision tree?
  - Avoid a tree growing too complex (i.e., overfit)
- Pruning methods
  - Pre-pruning (Chapter 9.3, p.128)
  - Post-pruning (Chapter 9.4, p.130)
- Post-pruning
  - Reduced error pruning, Error based pruning, Pessimistic error pruning, Cost complexity pruning

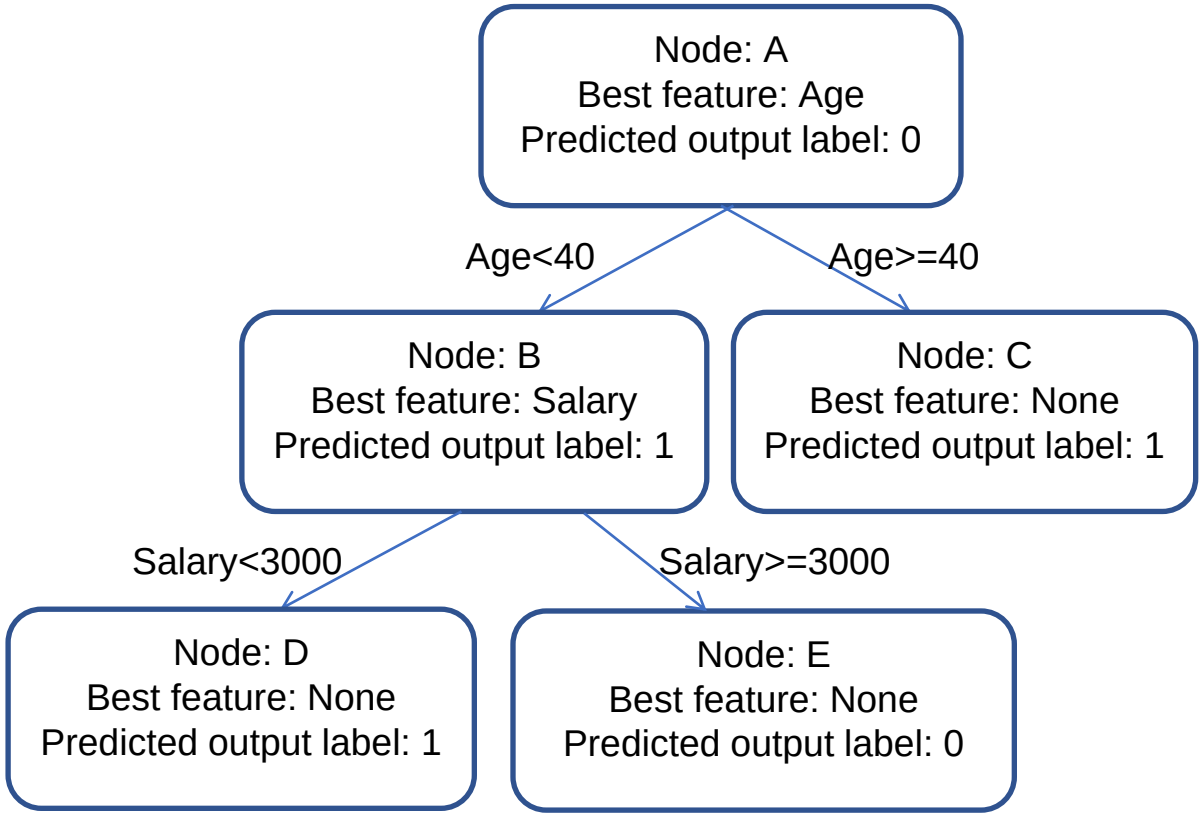
Post-pruning

- Textbook Chapter 9.4

- Procedure:
  - Data: Training data samples, and pruning data samples
  - Build/Train a tree model with **training data samples**
  - Pass **pruning data samples** into the respective tree nodes(s) of the original training model, and calculate the **error rate** of each node
  - Find all the "**candidate branches**" (TBD for pruning), compare the **weighted error rate** after splitting (i.e., child nodes) with the **error rate** before splitting (i.e., parent node), and decide whether to prune this branch
  - Recursively do post-pruning on the training model, until the structure of the training model does not change
  - End.

Simple Example

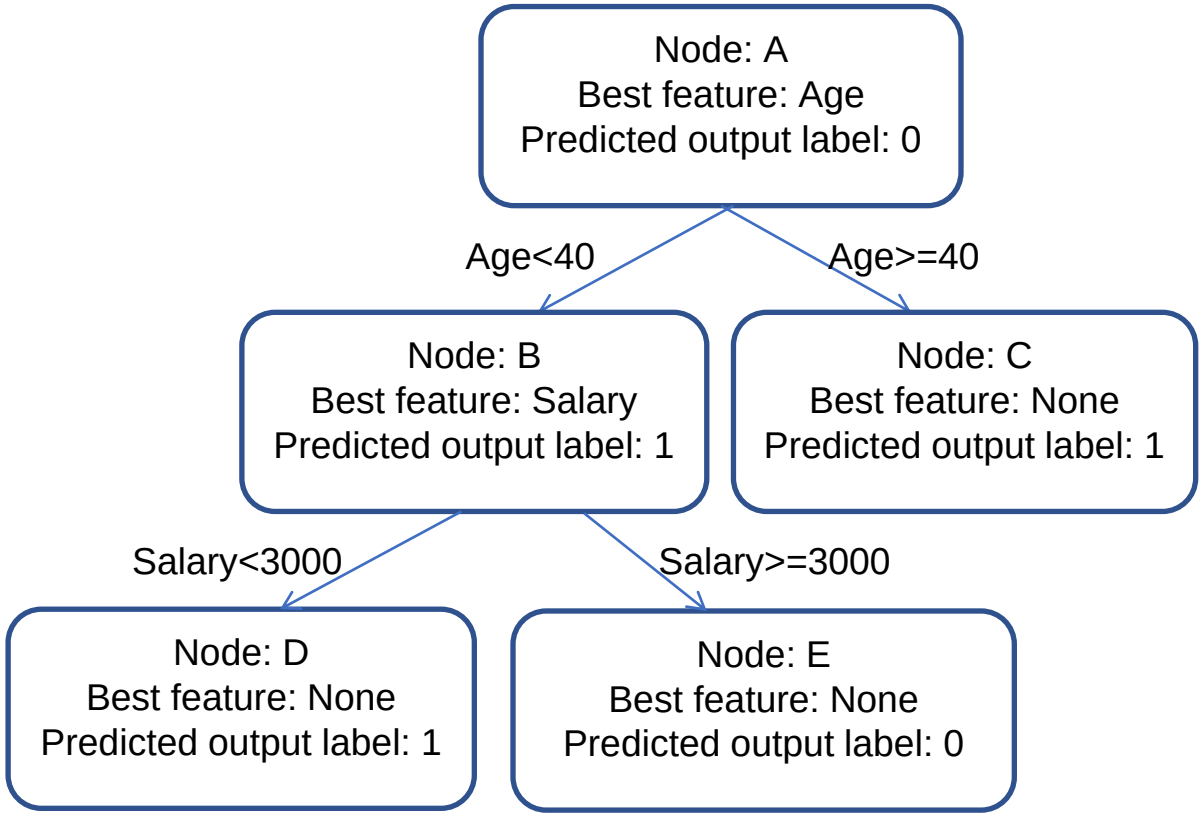
- Build a tree with **training data samples** and get:



- Q&A:
- The "predicted output label" of each node in the training model is calculated by "**Majority voting**" method.
- For example, suppose in Node A (with 100 **training data samples**), 70 outputs are 1s, and 30 outputs are 0s, then according to majority voting, the "predicted output label" should be 1.
- You get "predicted output label" when you build your training model using **training data samples**.

Simple Example

- Pass **pruning data samples** into the tree nodes(s) of the original training model, and calculate the **prediction error rate** of each node

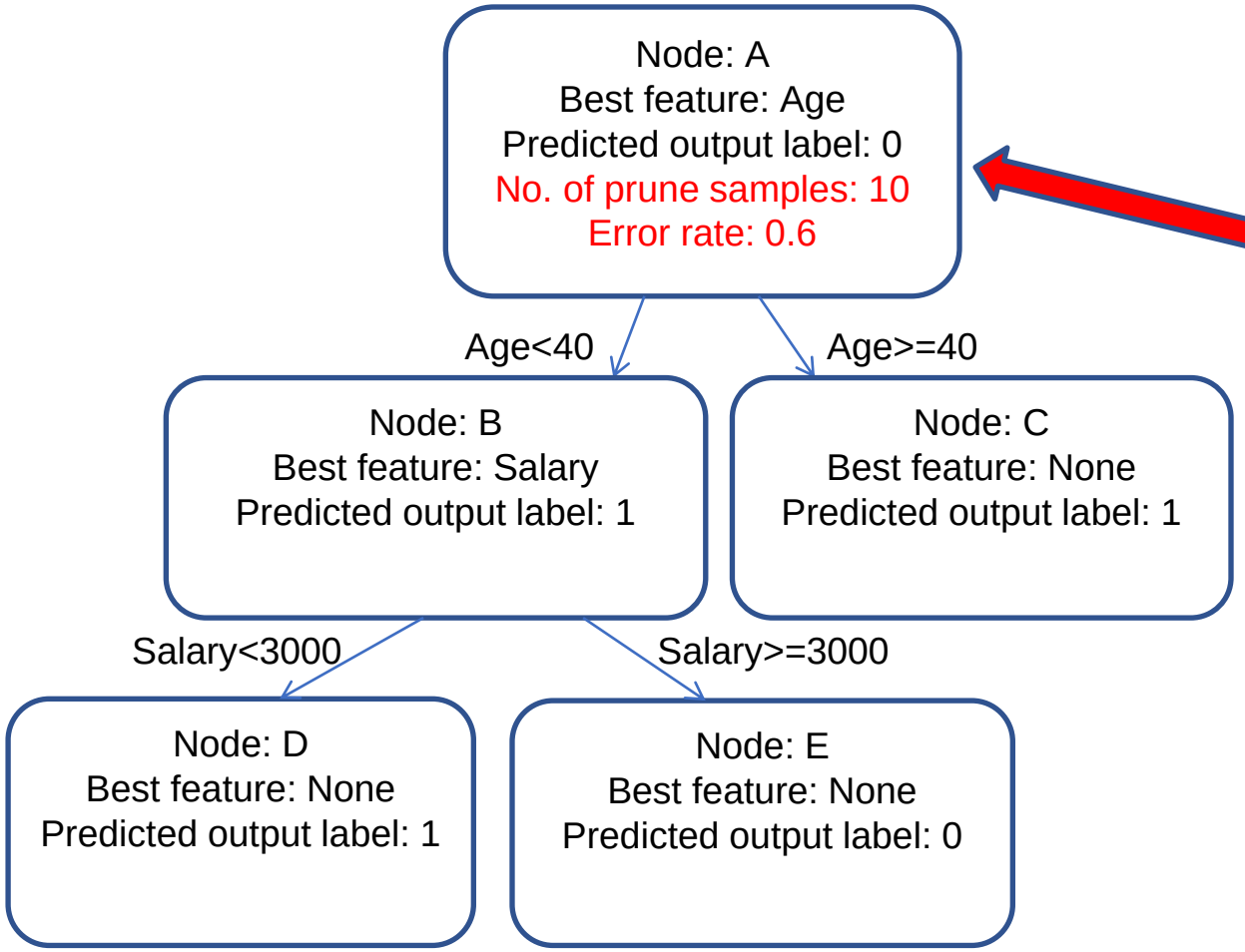


- Suppose the size of pruning data samples is 10, which means you have 10 data samples in the **pruning data:**

ID	Age	Salary	Output label
1	20	3000	1
2	25	4000	0
3	50	1500	0
4	45	2500	1
5	33	4500	1
6	23	5000	0
7	27	1800	0
8	37	7000	1
9	39	1000	1
10	29	2400	1

Simple Example

- Pass **pruning data samples** into the tree nodes(s) of the original training model, and calculate the **prediction error rate** of each node

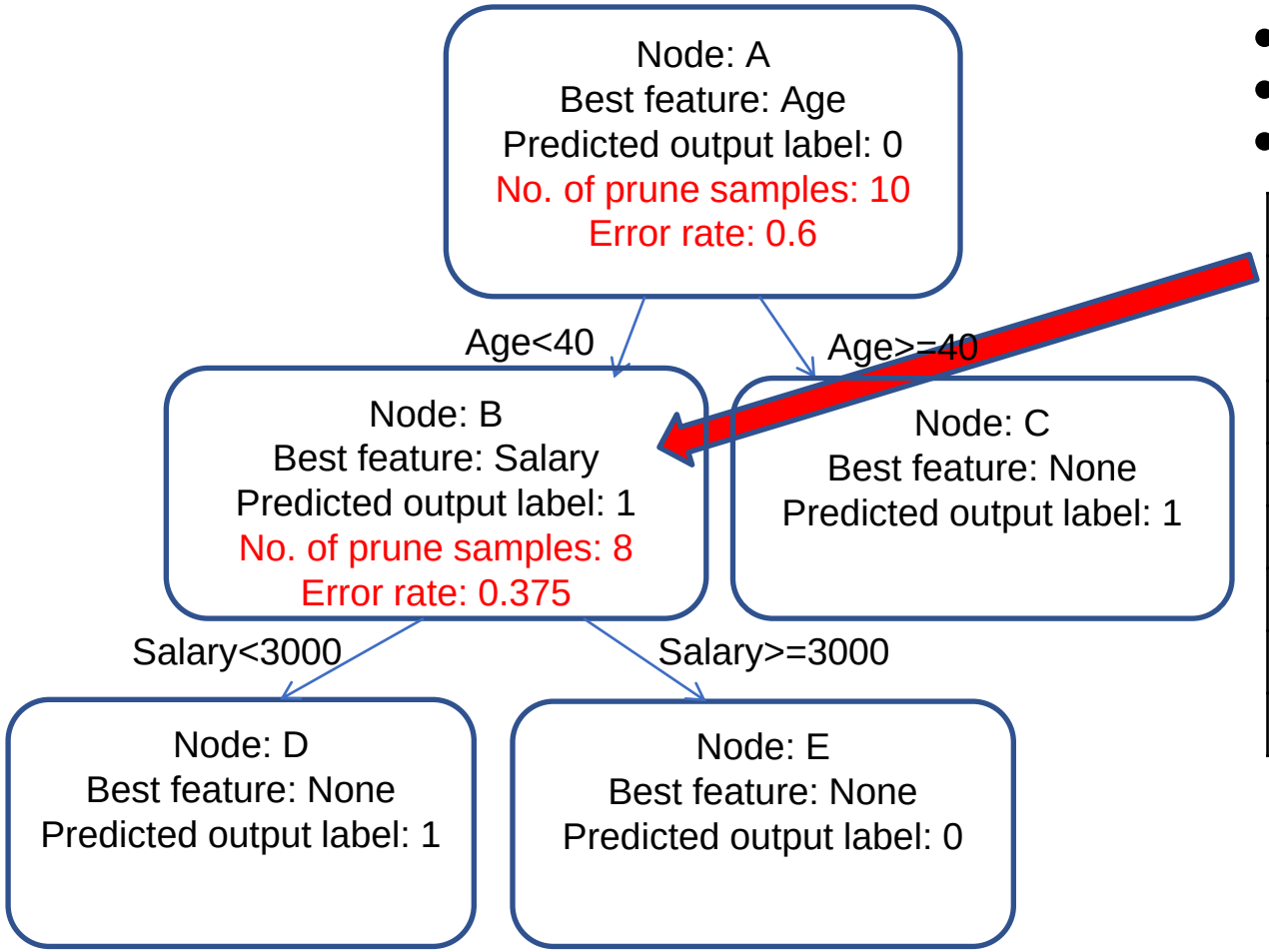


- Store **pruning data samples** in Node A
- In pruning samples of Node A: 6 **1s** and 4 **0s**
- Predicted output label is 0 (from training model)
- Error rate of Node A: 6/10=0.6

ID	Age	Salary	Output label
1	20	3000	1
2	25	4000	0
3	50	1500	0
4	45	2500	1
5	33	4500	1
6	23	5000	0
7	27	1800	0
8	37	7000	1
9	39	1000	1
10	29	2400	1

Simple Example

- Pass **pruning data samples** into the tree nodes(s) of the original training model, and calculate the **prediction error rate** of each node

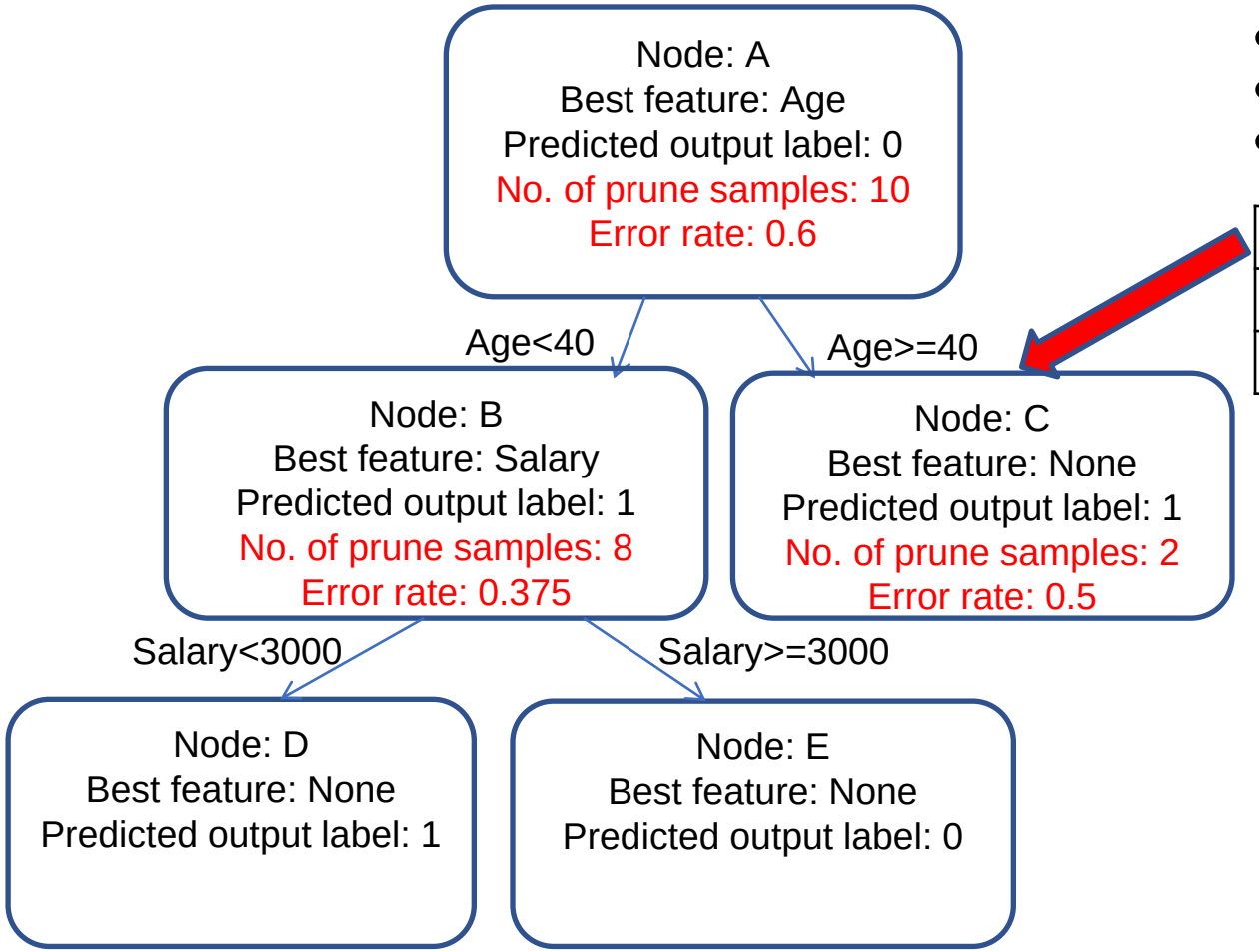


- Store **pruning data samples** in Node B
- In pruning samples of Node B: 5 **1s** and 3 **0s**
- Predicted output label is 1 (from training model)
- Error rate of Node B: 3/8=0.375

ID	Age	Salary	Output label
1	20	3000	1
2	25	4000	0
5	33	4500	1
6	23	5000	0
7	27	1800	0
8	37	7000	1
9	39	1000	1
10	29	2400	1

Simple Example

- Pass **pruning data samples** into the tree nodes(s) of the original training model, and calculate the **prediction error rate** of each node

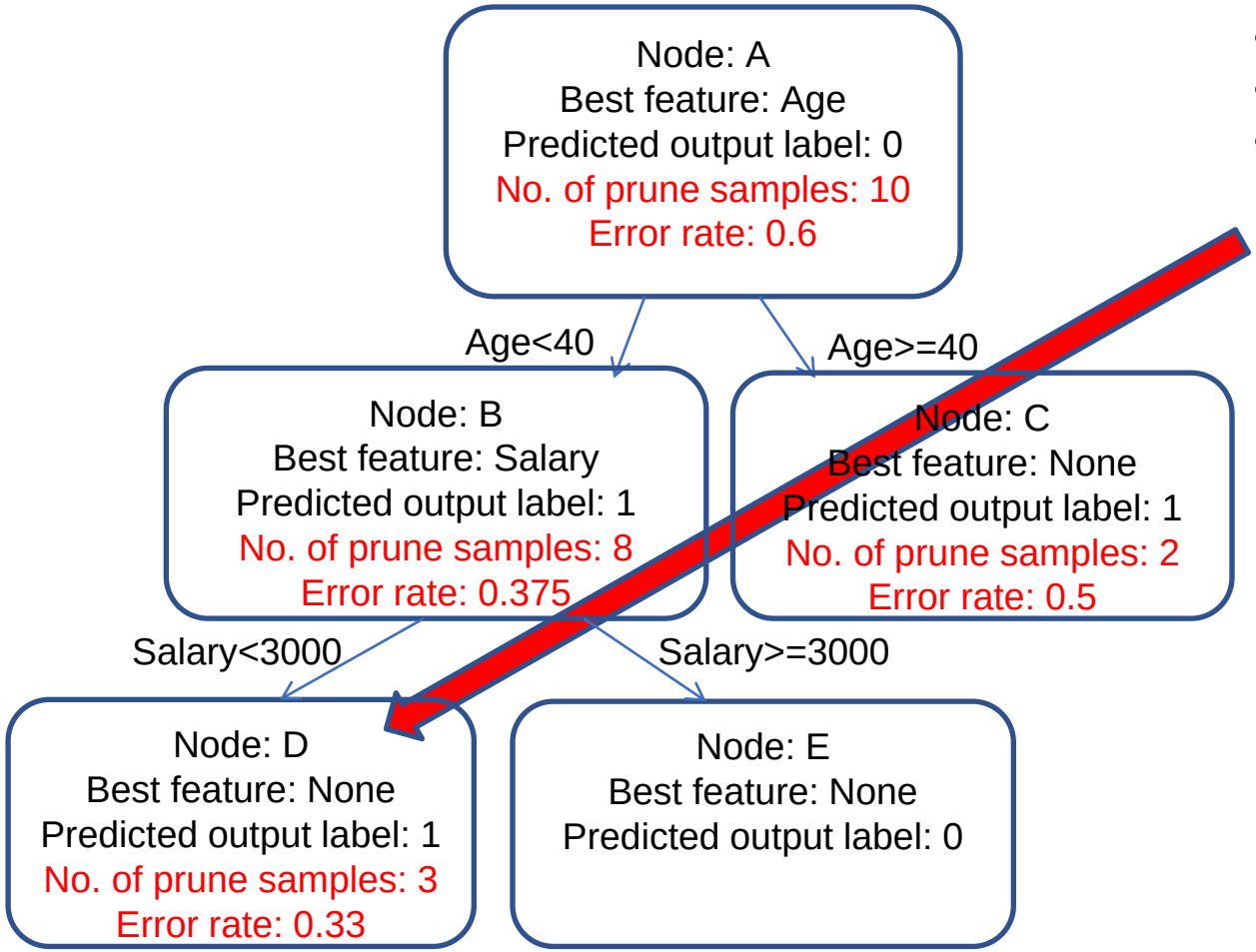


- Store **pruning data samples** in Node C
- In pruning samples of Node C: 1 **1s** and 1 **0s**
- Predicted output label is 1 (from training model)
- Error rate of Node C: 1/2=0.5

ID	Age	Salary	Output label
3	50	1500	0
4	45	2500	1

Simple Example

- Pass **pruning data samples** into the tree nodes(s) of the original training model, and calculate the **prediction error rate** of each node

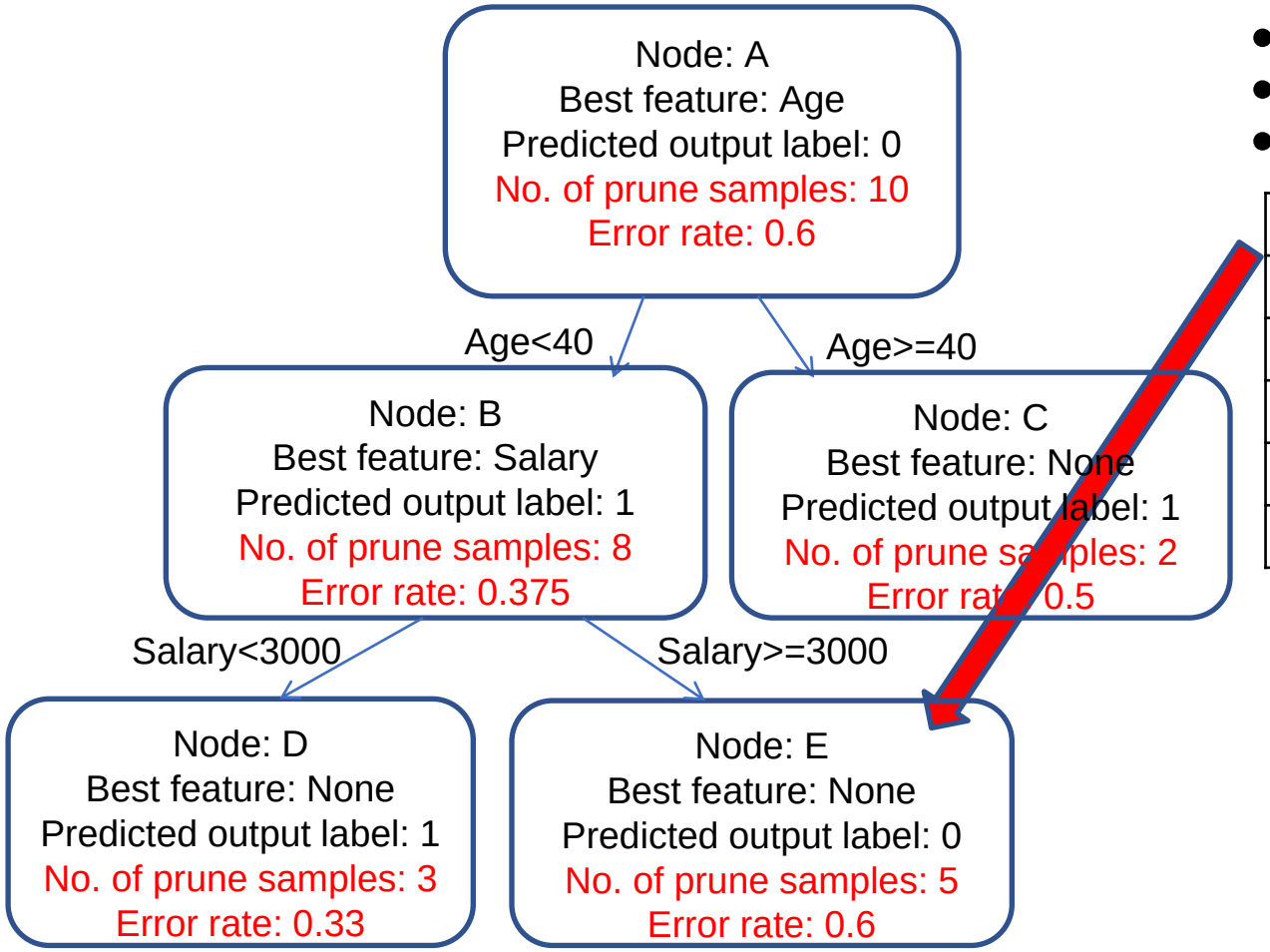


- Store **pruning data samples** in Node D
- In pruning samples of Node D: 2 **1s** and 1 **0s**
- Predicted output label is 1 (from training model)
- Error rate of Node D: 1/3=0.33

ID	Age	Salary	Output label
7	27	1800	0
9	39	1000	1
10	29	2400	1

Simple Example

- Pass **pruning data samples** into the tree nodes(s) of the original training model, and calculate the **prediction error rate** of each node

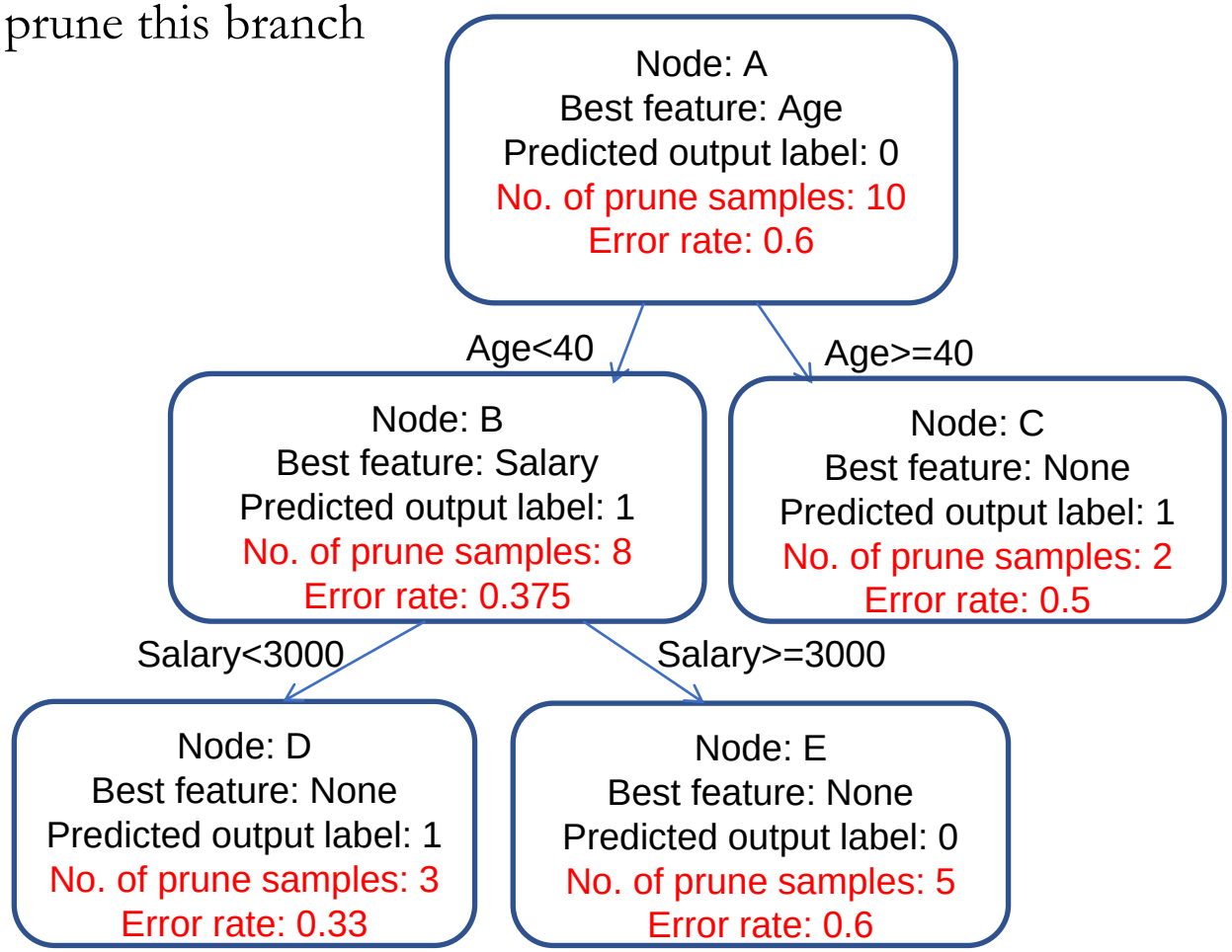


- Store **pruning data samples** in Node E
- In pruning samples of Node E: 3 **1s** and 2 **0s**
- Predicted output label is 0 (from training model)
- Error rate of Node E: 3/5=0.6

ID	Age	Salary	Output label
1	20	3000	1
2	25	4000	0
5	33	4500	1
6	23	5000	0
8	37	7000	1

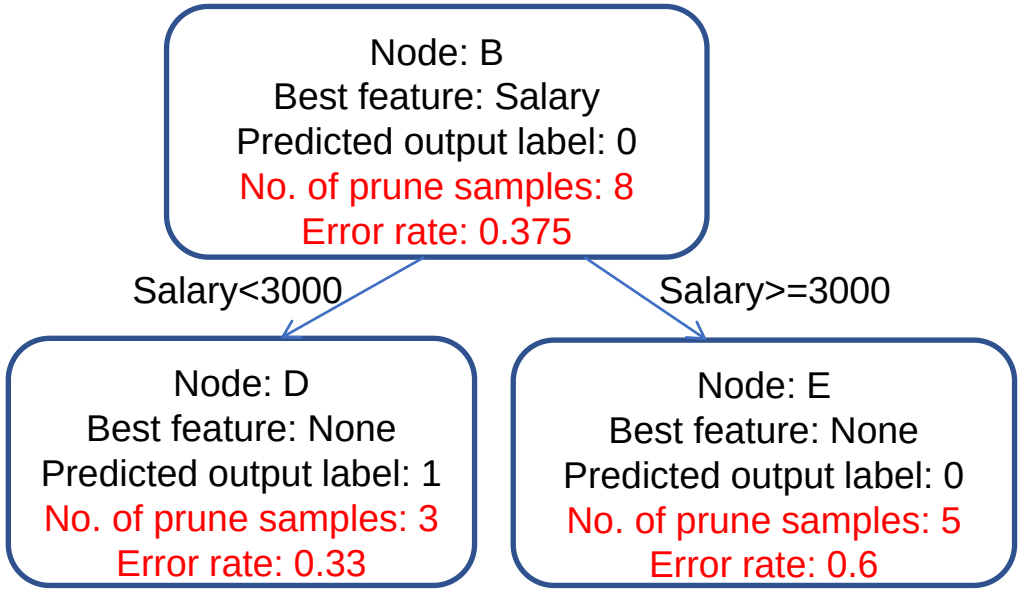
Simple Example

- Find all the "**candidate branches**" (TBD for pruning), compare the **weighted error rate** after splitting (i.e., child nodes) with the **error rate** before splitting (i.e., parent node), and decide whether to prune this branch



Simple Example

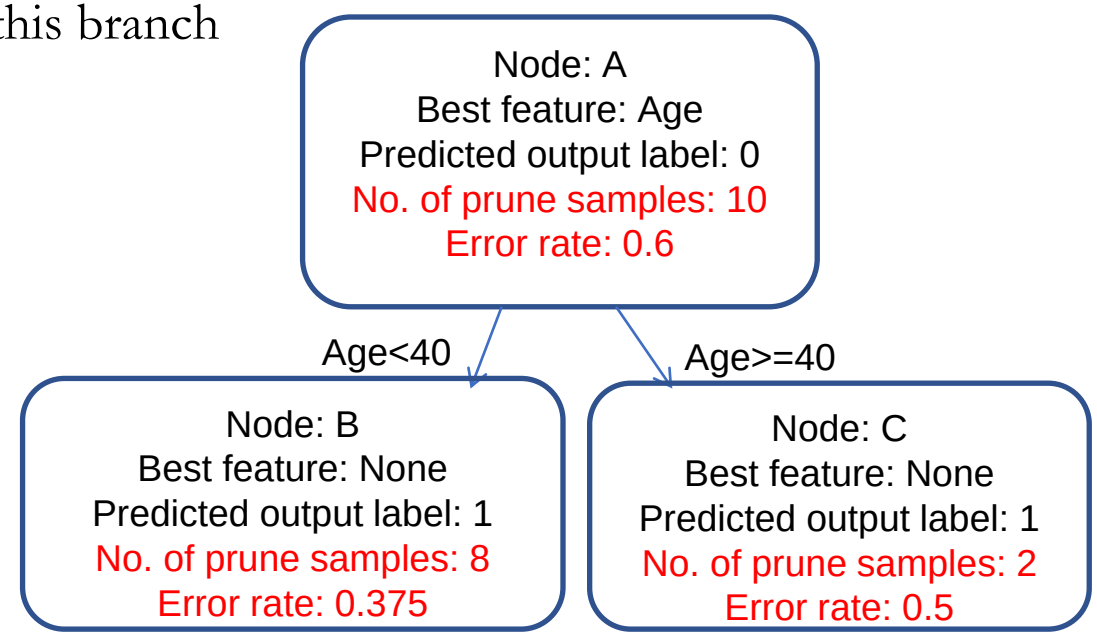
- Find all the "**candidate branches**" (TBD for pruning), compare the **weighted error rate** after splitting (i.e., child nodes) with the **error rate** before splitting (i.e., parent node), and decide whether to prune this branch



- Candidate branch: In this branch, both left- and right-child node of the parent node (non-leaf node) are leaf nodes
- Leaf node: has no child nodes
- Weighted error rate after splitting (i.e., child nodes): (3/8)*0.33+(5/8)*0.6=0.5
- Error rate before splitting (i.e., parent node): 0.375
- Weighted error rate after splitting >= Error rate before splitting
- Prune this branch: Keep Node B and Remove Node D & Node E

Simple Example

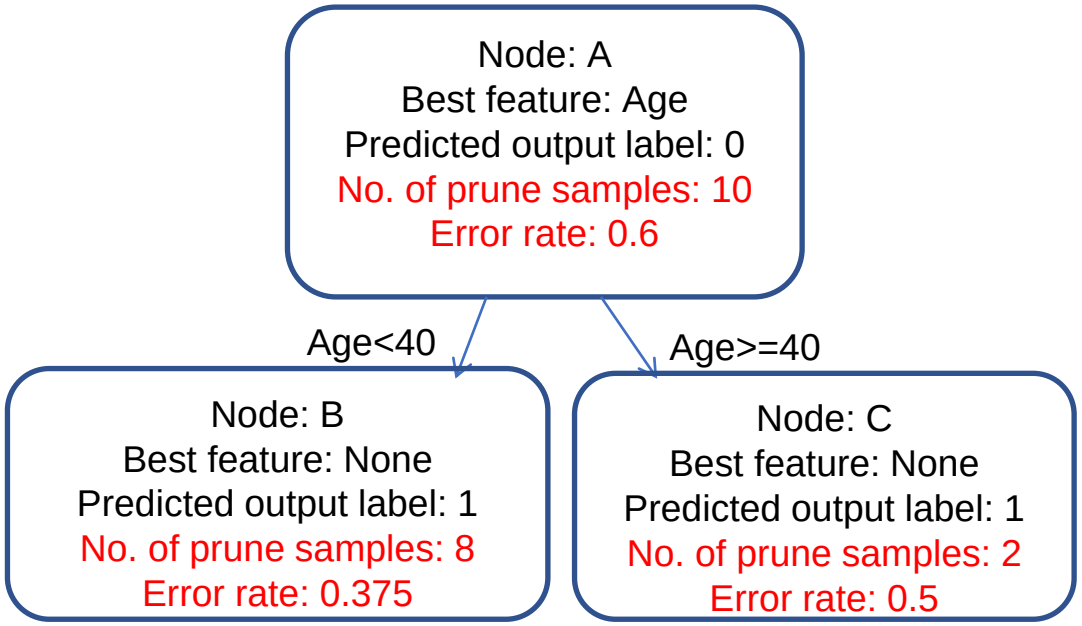
- Find all the "**candidate branches**" (TBD for pruning), compare the **weighted error rate** after splitting (i.e., child nodes) with the **error rate** before splitting (i.e., parent node), and decide whether to prune this branch



- Find a new candidate branch
- Weighted error rate after splitting (i.e., child nodes): (8/10)*0.375+(2/10)*0.5=0.4
- Error rate before splitting (i.e., parent node): 0.6
- Weighted error rate after splitting < Error rate before splitting
- Do not prune this branch

Simple Example

- Find all the "**candidate branches**" (TBD for pruning), compare the **weighted error rate** after splitting (i.e., child nodes) with the **error rate** before splitting (i.e., parent node), and decide whether to prune this branch



- Do pruning on candidate branches, until the tree structure is unchanged
- "In an extreme case, this method could lead to a decision tree being post-pruned right up to its root node, indicating that using the tree is likely to lead to a higher error rate, i.e., more incorrect classifications, than simply assigning every unseen instance to the largest class in the training data. Luckily such poor decision trees are likely to be very rare." (Chapter 9.4, p.135)

Code

https://github.com/mozartkun/BT2101_Tutorials_2018_2019_SEM1/blob/master/Tutorial%201.%20Numpy%2C%20Pandas%20and%20Decision%20Tree/Decision%20Tree_Postpruning.ipynb

Thank you!