Decision Trees

Decision Trees - 1

- Algorithms used for both Classification and Regression
- Works effectively with non-linear dataset
- DT can be looked at like “rules” that can be understood by humans and implemented on datasets
- Uses the Greedy algorithm technique
- Core algorithm to build decisions is called **ID3,** that employs a top-down approach
- Dataset is split on the ***most significant feature (***using ***Entropy / Information Gain)***

- DT represents an inverted tree having the following attributes:
  - ❑ Decision node : Test for split of an attribute
  - ❑ Edge : split of an attribute
  - ❑ Leaf node : value of the target attribute
  - ❑ Path : a series of test to arrive at the final decision

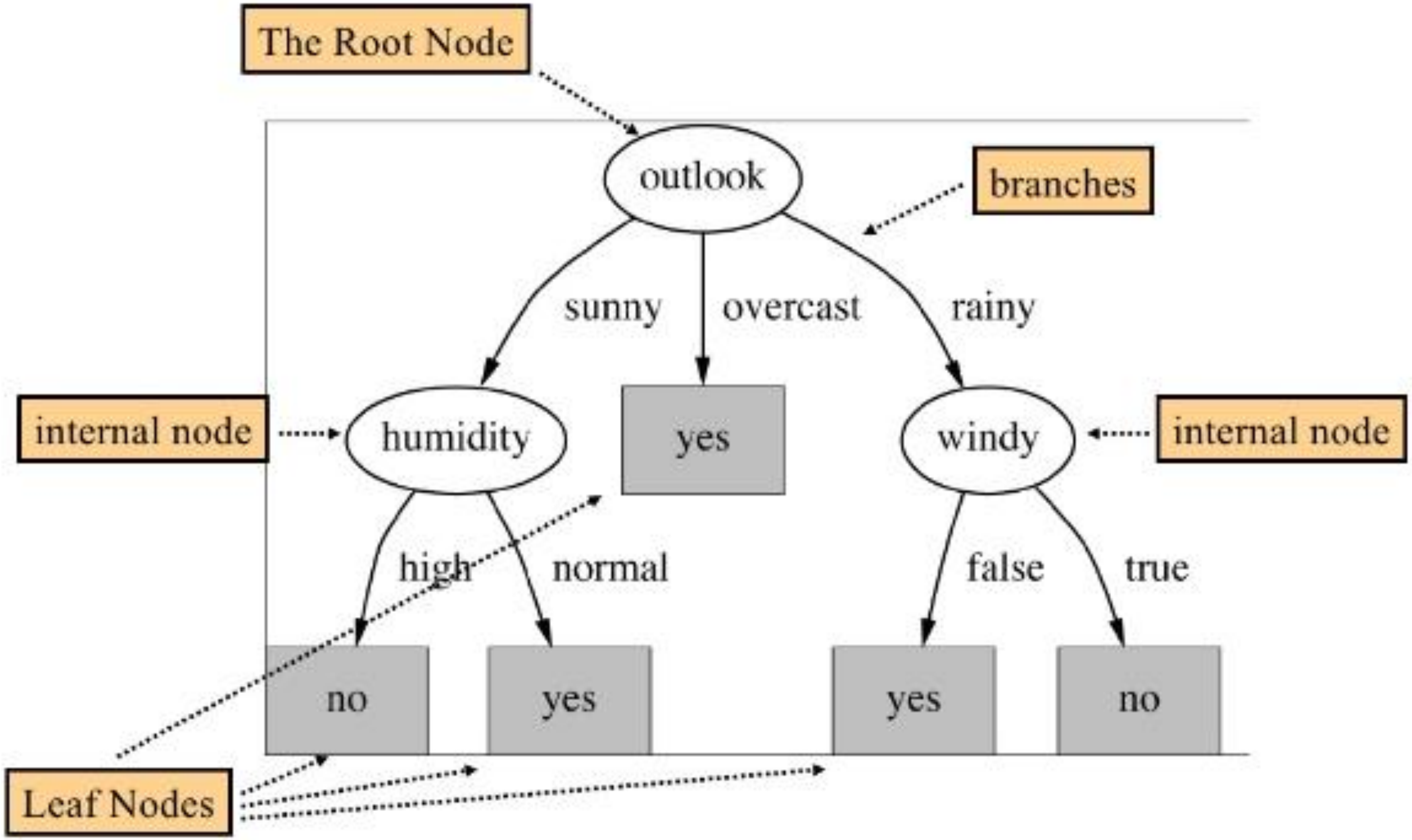
- Using recursion, sub-trees are formed based on features not used in the higher nodes
- Divide and rule
- DT splits data until it reaches a “pure” state
  - ➢ Pure subset is one where there are only **positive** outcomes. No further split

Greedy algorithm technique

- A choice made which seems appropriate at that point of time
- A local-optimum choice that would lead to a global-optimum solution
  - ➢ But doesn't happen always
- Algorithm does not go back and reverse its decision
  - ➢ has only 1 shot to make the local optimum choice

IDE, C4.5, C5.0 -> Entropy and Information Gain
CART -> Gini Index model

Decision Tree Terminology



Pic courtesy: Internet

Decision Trees

- Conditions for split can be given during the model building process. For example:
  - ❑ Control the depth of the tree
  - ❑ Split the data if the minimum result for a condition is ‘n’ [n is any positive number]
  - ❑ Control the Complexity Parameter (to include or exclude splits)

rpart.control() function in R lets you to specify the different conditions

- **Advantages**
  - ❑ Interpretable
  - ❑ Easy to understand
  - ❑ Scalable
  - ❑ Robust

- With more features, Trees can grow large and may become difficult to understand
- Smaller trees have better accuracy than larger trees
- Test dataset may become difficult to generalise (*tips example*)

Predict whether someone will buy or sell stocks

Day	company_perf	exchange_rate	gold_price	Action
1	High	Increase	Stable	Sale
2	Medium	Increase	Unstable	Purchase
3	High	Increase	Unstable	Purchase
4	Medium	Decrease	Stable	Purchase
5	Low	Decrease	Unstable	Purchase
6	Medium	Increase	Stable	Sale
7	High	Decrease	Stable	Purchase
8	Medium	Increase	Stable	Sale
9	High	Increase	Stable	Sale
10	Medium	Decrease	Unstable	Purchase
11	High	Decrease	Stable	Purchase
12	Medium	Increase	Stable	Sale
13	Medium	Decrease	Unstable	Purchase
14	Low	Increase	Stable	Purchase
15	Low	Increase	Stable	Purchase

16	**Low**	**Decrease**	**Stable**	**????**

Company_perf High, Medium, Low
Exch_rate Increase, Decrease
Gold_price Stable, Unstable

Action 9 Purchase / 6 Sale

company_perf

High

3 P / 2 S

1	High	Increase	Stable
3	High	Increase	Unstable
7	High	Decrease	Stable
9	High	Increase	Stable
11	High	Decrease	Stable

Medium

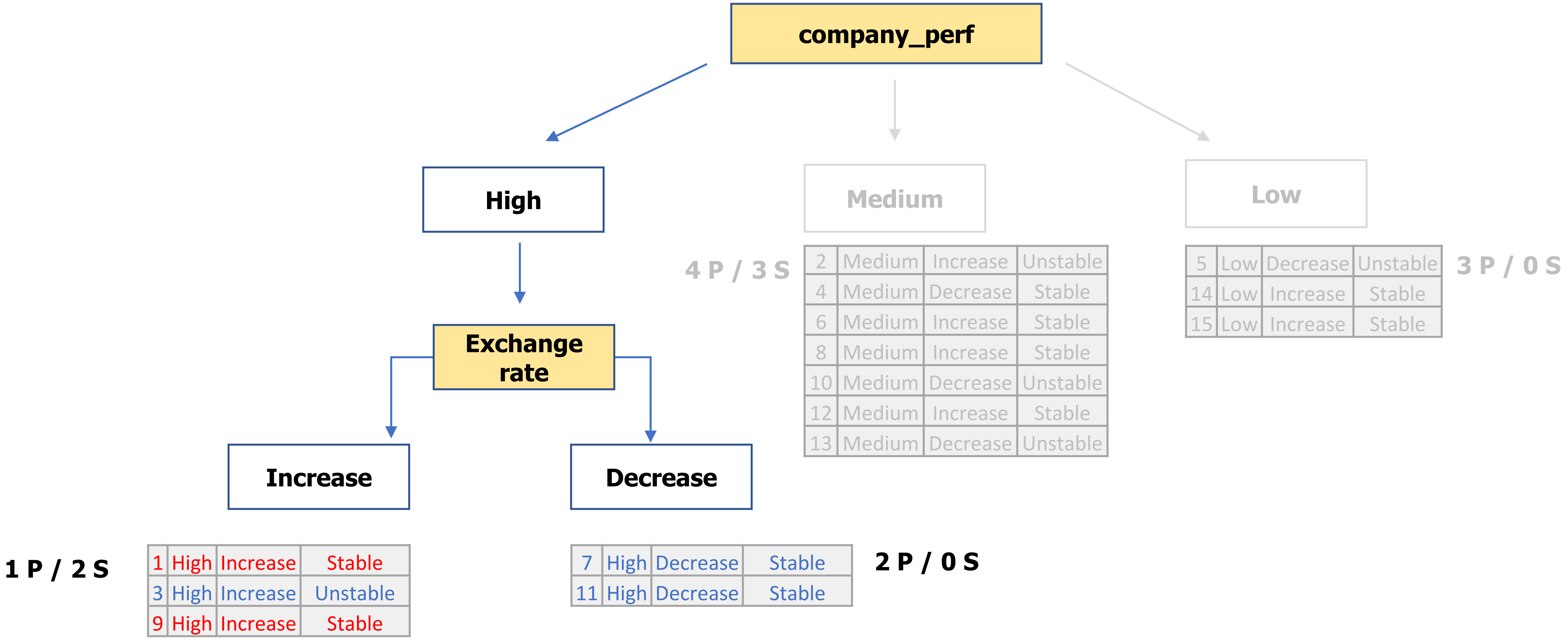
4 P / 3 S

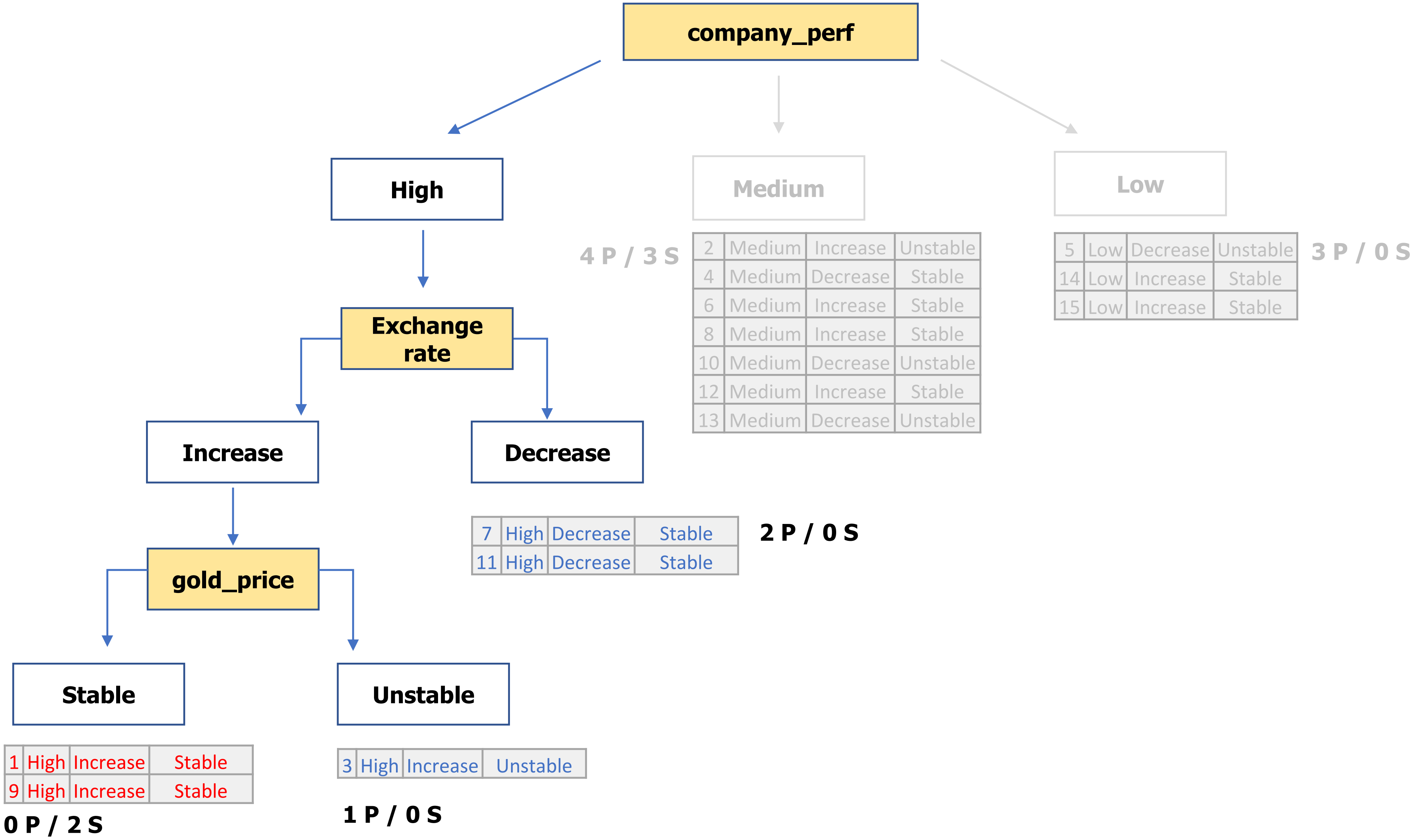
2	Medium	Increase	Unstable
4	Medium	Decrease	Stable
6	Medium	Increase	Stable
8	Medium	Increase	Stable
10	Medium	Decrease	Unstable
12	Medium	Increase	Stable
13	Medium	Decrease	Unstable

Low

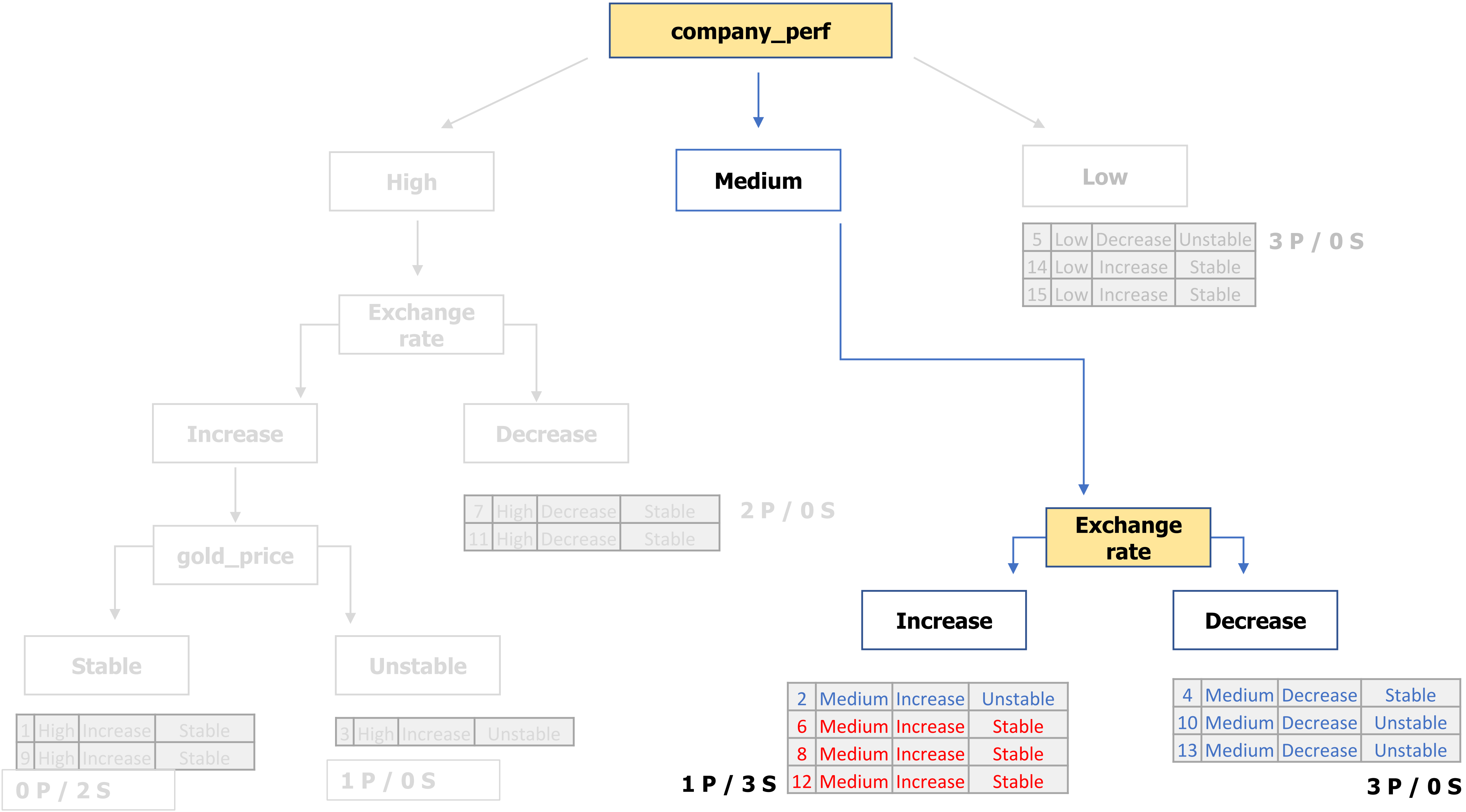
3 P / 0 S

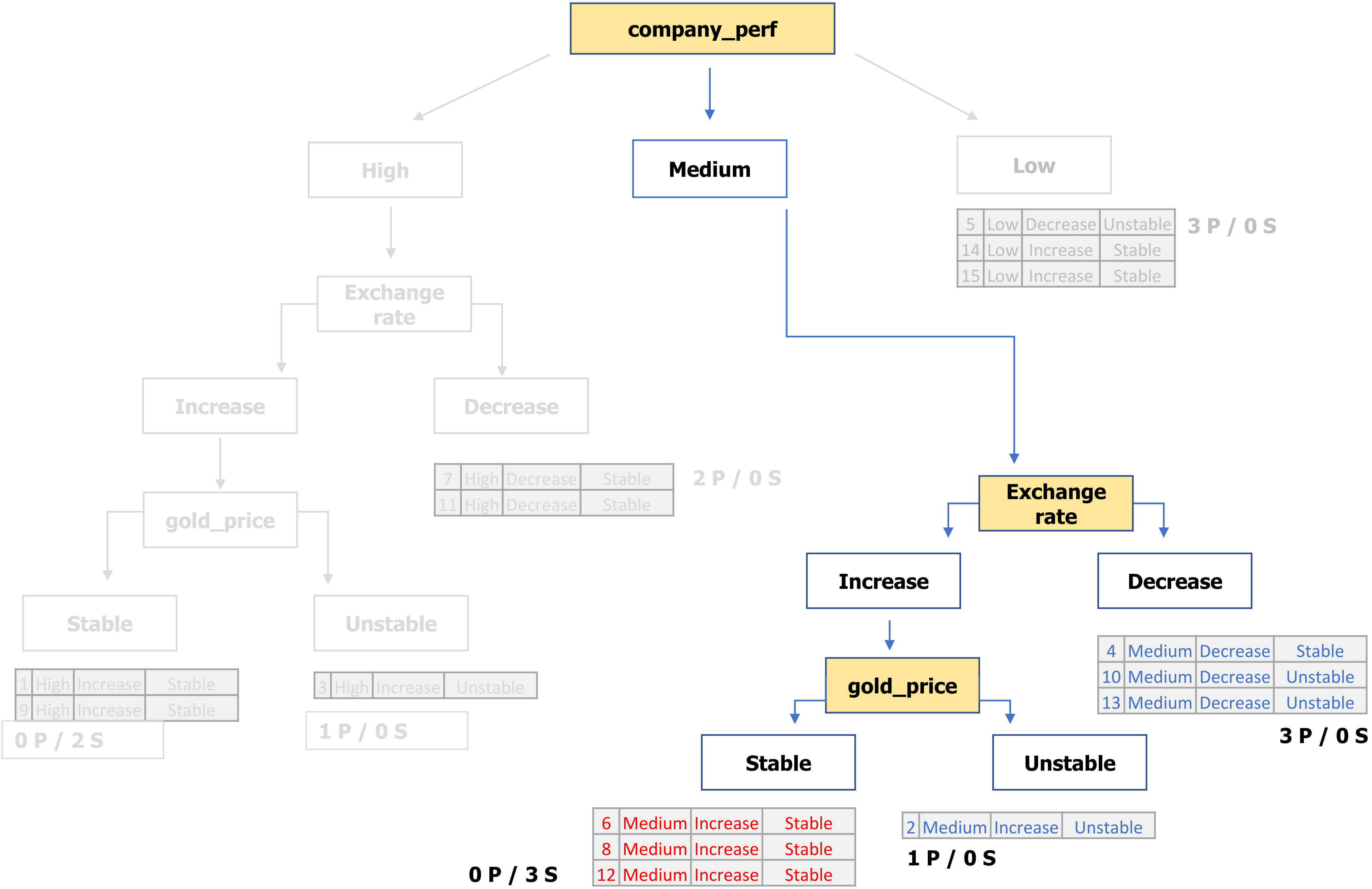
5	Low	Decrease	Unstable
14	Low	Increase	Stable
15	Low	Increase	Stable


company_perf
High
Medium
Low
4 P / 3 S
2 Medium Increase Unstable
4 Medium Decrease Stable
6 Medium Increase Stable
8 Medium Increase Stable
10 Medium Decrease Unstable
12 Medium Increase Stable
13 Medium Decrease Unstable
5 Low Decrease Unstable
14 Low Increase Stable
15 Low Increase Stable
3 P / 0 S
Exchange rate
Increase
Decrease
1 P / 2 S
1 High Increase Stable
3 High Increase Unstable
9 High Increase Stable
7 High Decrease Stable
11 High Decrease Stable
2 P / 0 S



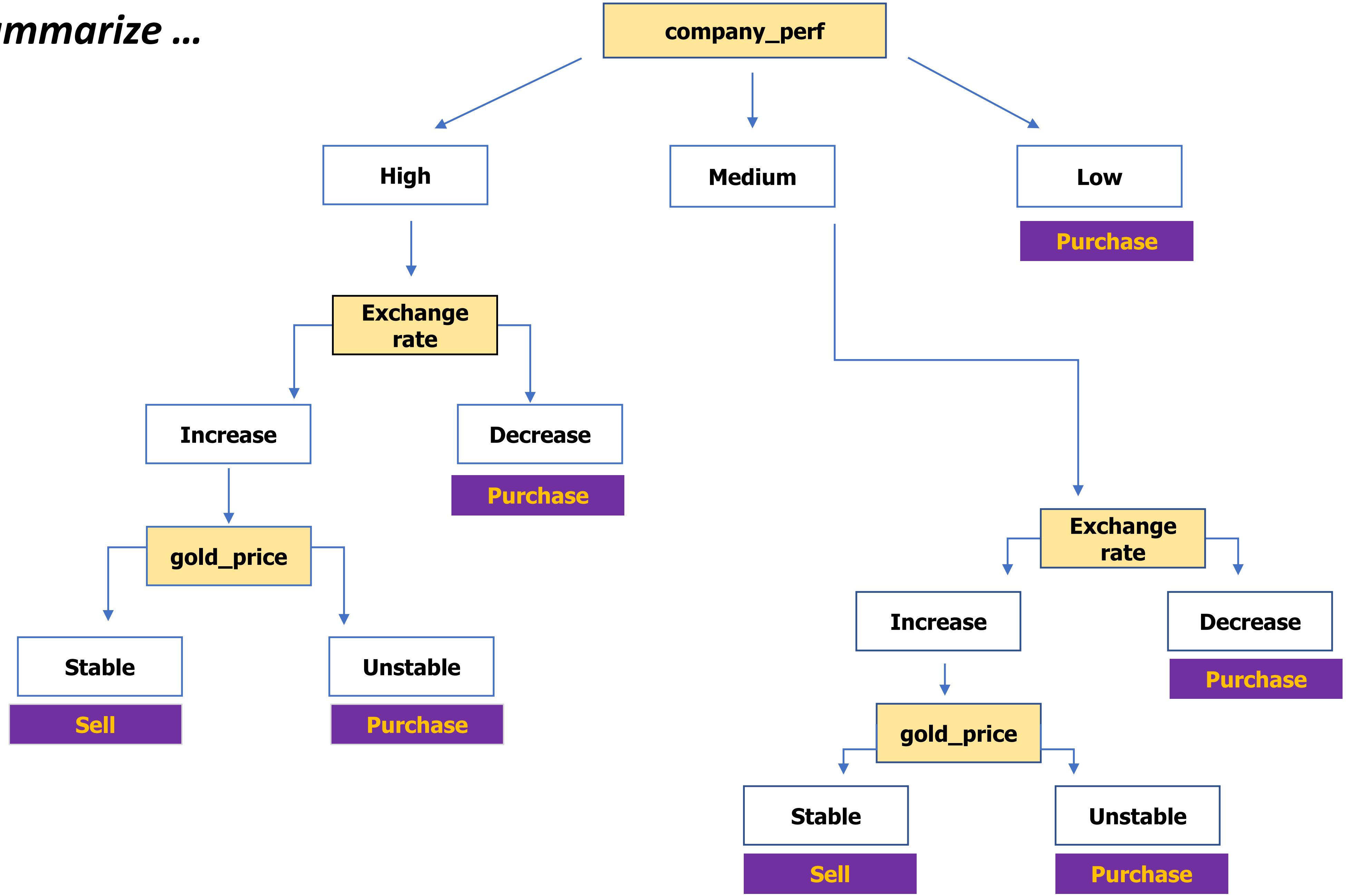
company_perf
High
Medium
Low
Exchange rate
Increase
Decrease
gold_price
Stable
Unstable
4 P / 3 S
2 Medium Increase Unstable
4 Medium Decrease Stable
6 Medium Increase Stable
8 Medium Increase Stable
10 Medium Decrease Unstable
12 Medium Increase Stable
13 Medium Decrease Unstable
5 Low Decrease Unstable
14 Low Increase Stable
15 Low Increase Stable
3 P / 0 S
7 High Decrease Stable
11 High Decrease Stable
2 P / 0 S
1 High Increase Stable
9 High Increase Stable
0 P / 2 S
3 High Increase Unstable
1 P / 0 S

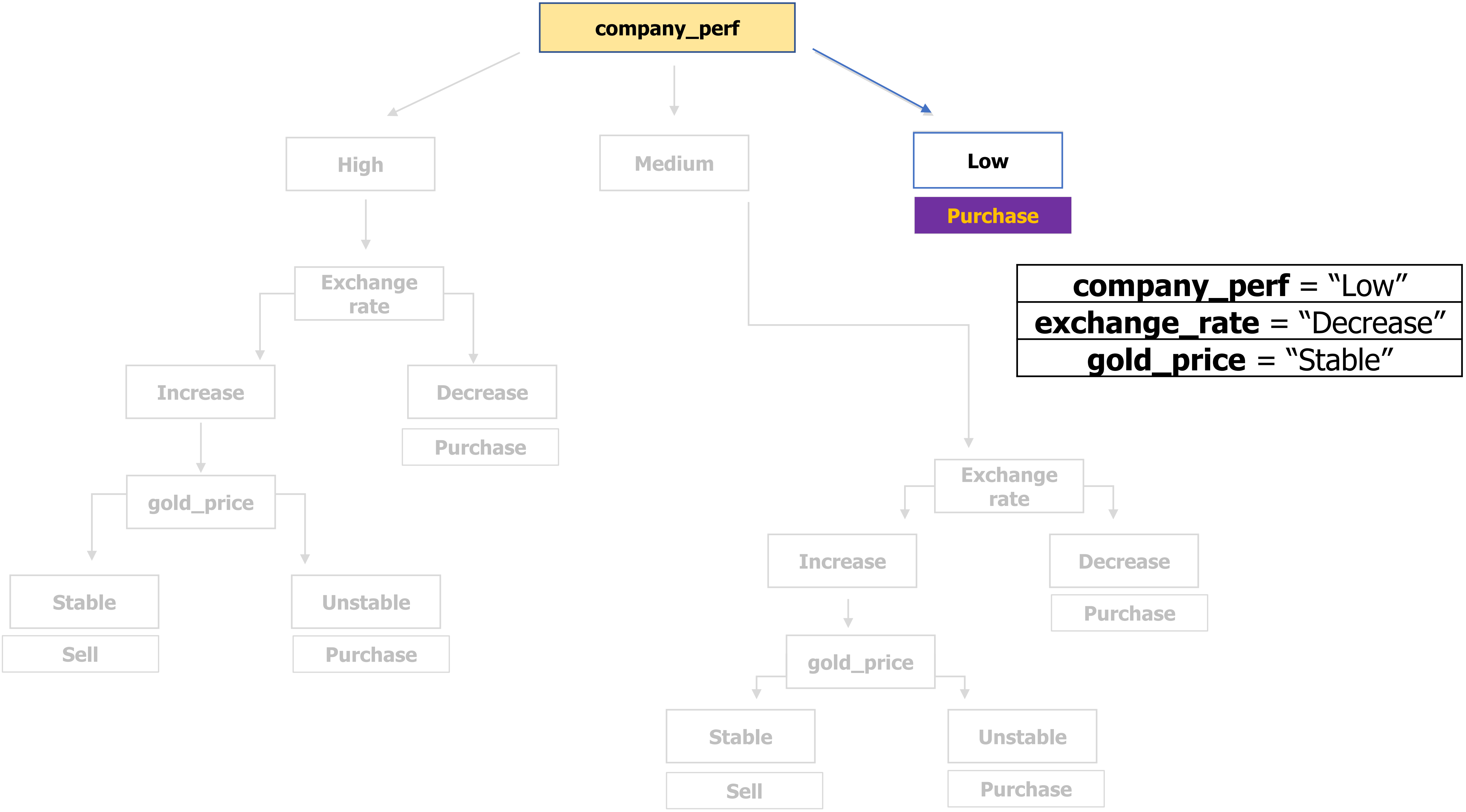

company_perf
High
Medium
Low
5 Low Decrease Unstable
14 Low Increase Stable
15 Low Increase Stable
3 P / 0 S
Exchange rate
Increase
Decrease
7 High Decrease Stable
11 High Decrease Stable
2 P / 0 S
gold_price
Stable
Unstable
1 High Increase Stable
9 High Increase Stable
0 P / 2 S
3 High Increase Unstable
1 P / 0 S
Exchange rate
Increase
Decrease
2 Medium Increase Unstable
6 Medium Increase Stable
8 Medium Increase Stable
12 Medium Increase Stable
1 P / 3 S
4 Medium Decrease Stable
10 Medium Decrease Unstable
13 Medium Decrease Unstable
3 P / 0 S



company_perf
High
Medium
Low
5 Low Decrease Unstable
14 Low Increase Stable
15 Low Increase Stable
3 P / 0 S
Exchange rate
Increase
Decrease
7 High Decrease Stable
11 High Decrease Stable
2 P / 0 S
gold_price
Stable
Unstable
1 High Increase Stable
9 High Increase Stable
0 P / 2 S
3 High Increase Unstable
1 P / 0 S
Exchange rate
Increase
Decrease
4 Medium Decrease Stable
10 Medium Decrease Unstable
13 Medium Decrease Unstable
3 P / 0 S
gold_price
Stable
Unstable
6 Medium Increase Stable
8 Medium Increase Stable
12 Medium Increase Stable
0 P / 3 S
2 Medium Increase Unstable
1 P / 0 S

To summarize …




company_perf
High
Medium
Low
Purchase
company_perf = "Low"
exchange_rate = "Decrease"
gold_price = "Stable"
Exchange rate
Increase
Decrease
Purchase
gold_price
Stable
Sell
Unstable
Purchase
Exchange rate
Increase
Decrease
Purchase
gold_price
Stable
Sell
Unstable
Purchase

Split criteria

Entropy
Total information held relating to the target variable (Binary) (IDE / C4.5)
More information, better will be the result

Entropy (I) of the target variable

- ❑ Measures homogeneity of the sets
- ❑ Tells us how pure / impure a set is
- ❑ e.g. In a binary classification dataset, if **S** is the dataset having + and – classes, then Entropy (*I*nformation) is measured as:

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

where

$\mathbf{p_{(+)}}$ = % of positive class
$\mathbf{p_{(-)}}$ = % of negative class

- Interpretation of Entropy
  - ✓ 0 <= I <= 1
  - ✓ Number of bits that is needed to identify if an item in the given dataset is + or –
  - ✓ For a pure subset, number of bits = 0
  - ✓ For a tie, number of bits = 1

Information Gain

- ❑ Significant variable to split is determined by **Information Gain**
- ❑ Measure that determines how well a given attribute splits the dataset
- ❑ This measure is used at every step to determine the next best attribute
- ❑ Information (I) is needed to classify an object

- ❑ **Gain (S, A) = E(S) – Σ [(S_a/S) * E(S_a)]** **(residual)**

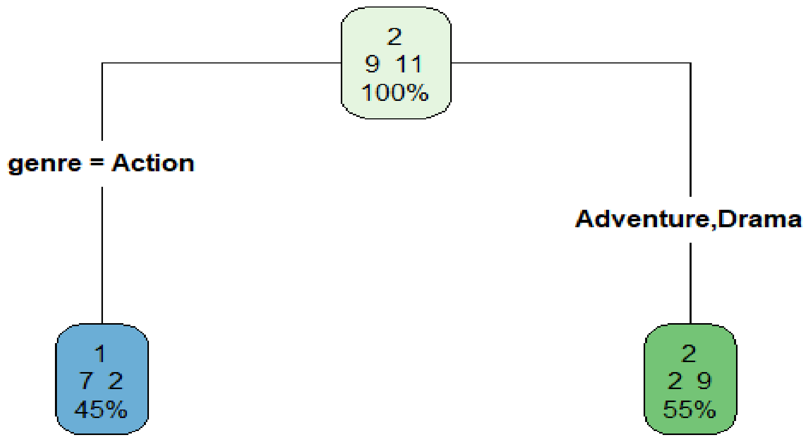
where
E(S) = Entropy calculation
$\mathbf{S_a}$ = Count of attribute value **a**
S = Total count of dataset of attribute **A**
E(S_a) = Entropy of Attribute value **a**

- ❑ **Maximum**(Gain(A)) → Best Attribute

Refer to chart

Exercise

#	creative_type	genre	rating	cat
1	Science Fiction	Action	PG-13	1
2	Fantasy	Adventure	PG-13	1
3	Fantasy	Adventure	PG	2
4	Fantasy	Drama	PG-13	2
5	Fantasy	Drama	PG-13	2
6	Science Fiction	Action	PG-13	1
7	Super Hero	Action	PG	1
8	Super Hero	Action	PG	1
9	Super Hero	Action	PG-13	2
10	Super Hero	Drama	R	2
11	Super Hero	Drama	PG-13	2
12	Science Fiction	Drama	PG-13	2
13	Science Fiction	Drama	R	2
14	Science Fiction	Action	PG	2
15	Science Fiction	Action	R	1
16	Fantasy	Action	R	1
17	Fantasy	Action	R	1
18	Fantasy	Adventure	R	1
19	Fantasy	Adventure	PG	2
20	Fantasy	Adventure	PG-13	2



Find the first best attribute to split

Pruning the Decision Tree

- Pruning is a technique to reduce the size of the Decision Tree by eliminating certain sections of the tree that provide little information to classify instances

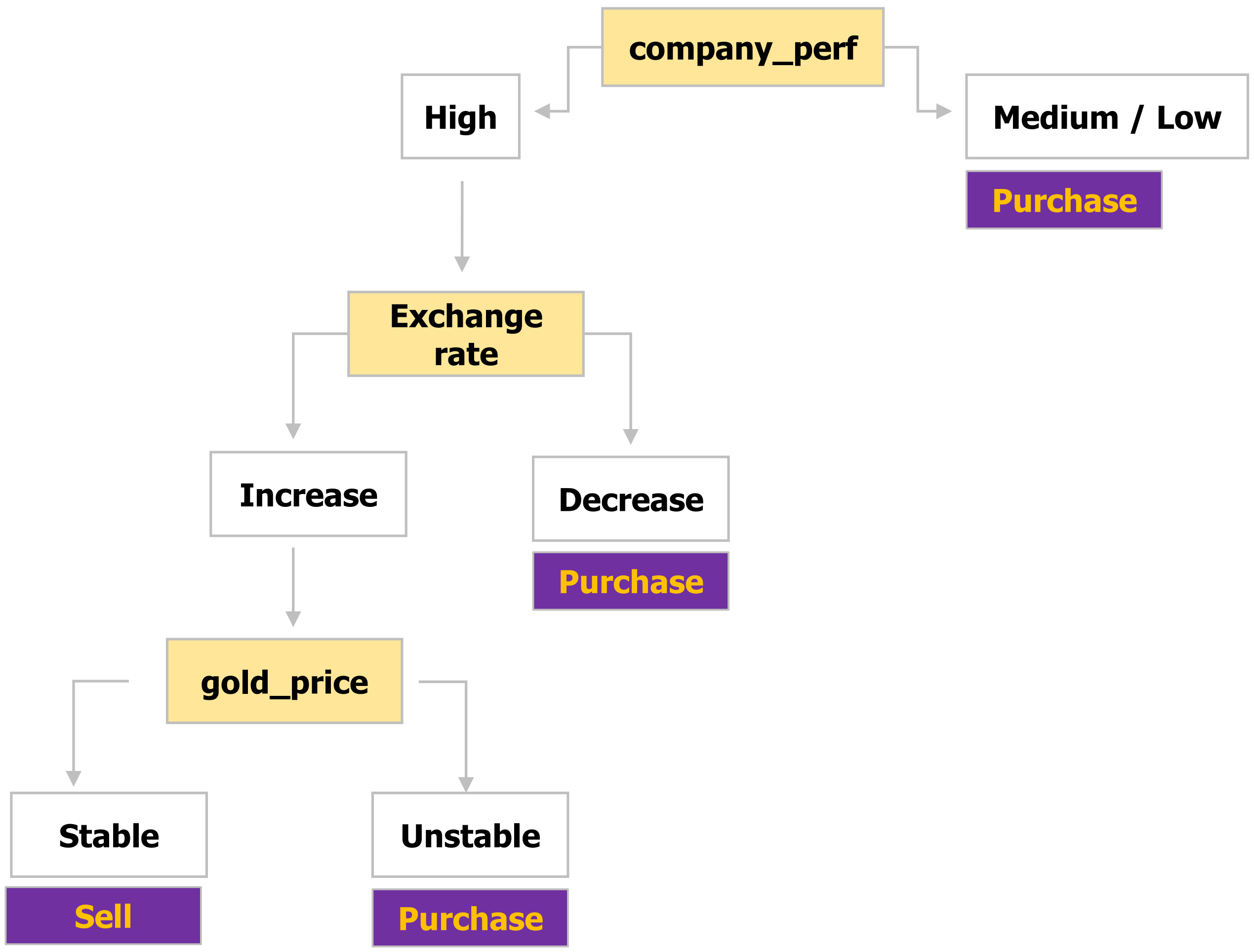
- Pruning a tree helps
  - ➢ Prevent overfitting
  - ➢ Improving accuracy

- Select the tree size that minimises the cross-validated error *(R has in-built function for this)*

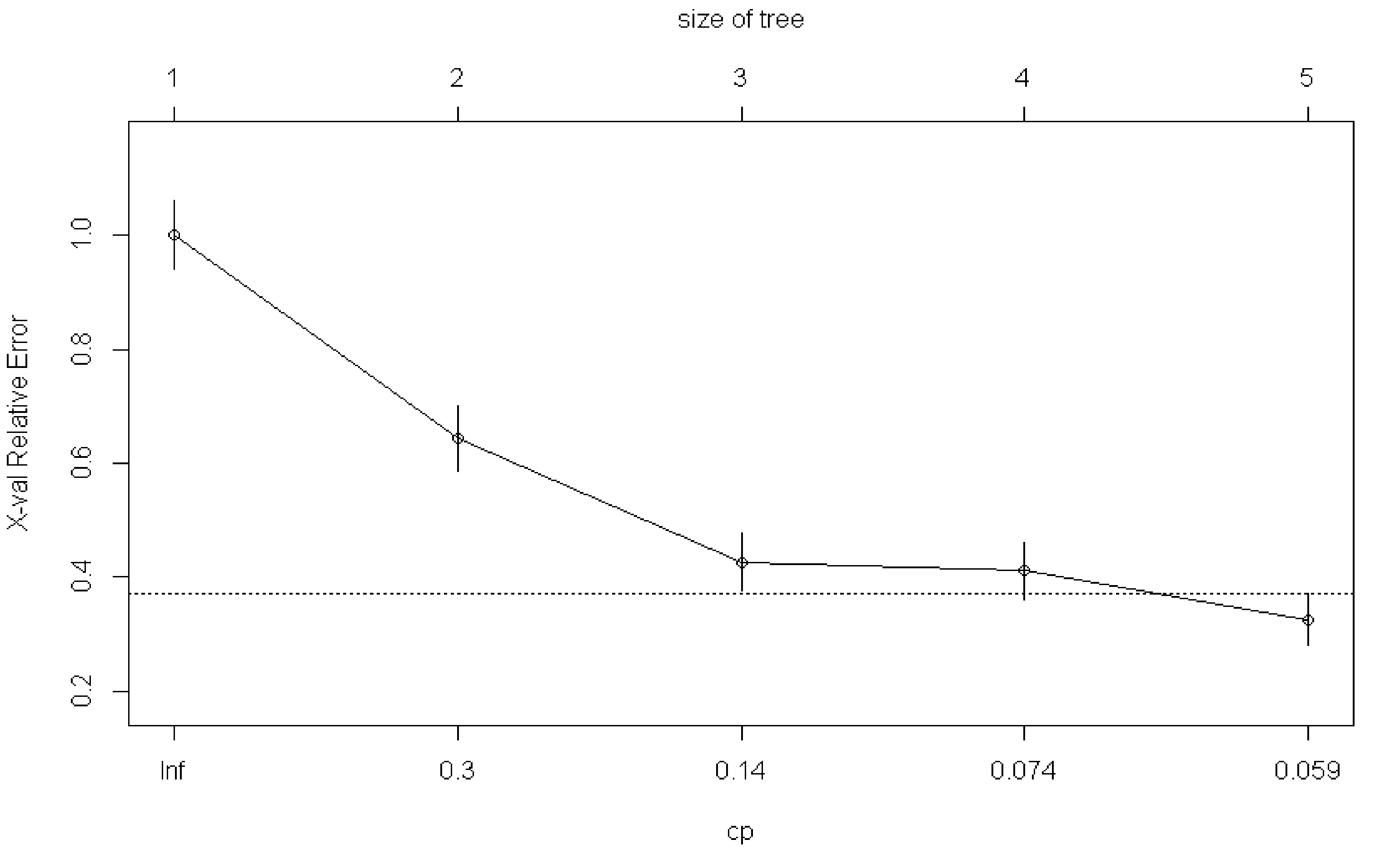
- Pruning is done using a technique known as "Complexity Parameter"

- Plotcp() provides a graphical representation of the cross-validation error summary

- Pruning the Decision Tree
  - ➢ Pre-pruning : Chi-square test
  - ➢ Post-pruning : Pruning techniques to reduce the tree size (recommended)



- Post pruning, perform prediction with the pruned tree
- Compare the results with the pre-pruned model to check effectiveness
- Plotcp() provides a graphical representation of the cross-validation error summary



Complexity Parameter (cp)

- Technique that determines
  - quality of a split
  - the total number of splits

- cp (default value) = 0.01
  - **cp = {low_value}**
    - More splits
    - Better results
  - **cp = {high_value}**
    - Lesser splits
    - May not give good results

- Next split depends on the cp

- Cross validation error reduces with each split

- Select the value of cp that corresponds to the minimum value of xerror (Cross Validation Error)

```
> printcp(basemodel)

Classification tree:
rpart(formula = lsp ~ ., data = train, method = "class", minsplit = 5,
    cp = 0.05, maxdepth = 5)

Variables actually used in tree construction:
[1] aac  alm1 gvh  mcg

Root node error: 129/235 = 0.54894

n= 235

        CP nsplit rel error  xerror     xstd
1 0.364341      0   1.00000 1.00000 0.059132
2 0.240310      1   0.63566 0.64341 0.056799
3 0.077519      2   0.39535 0.42636 0.050315
4 0.069767      3   0.31783 0.41085 0.049665
5 0.050000      4   0.24806 0.32558 0.045528
> |
```