Machine Learning

Università Roma Tre
Dipartimento di Ingegneria
Anno Accademico 2021 - 2022

Classificazione:
Alberi di Decisione

Sommario

- Introduzione ai Decision Trees
- Esempio di applicazione
- Feature split learning
- Decision Stump
- Algoritmo greedy decision tree learning
- Classificazione mediante Decision Trees

Alberi di Decisione

- Vediamo ora un altro metodo per la classificazione, molto utile nella pratica.
- Un albero di decisione prende come ingresso un oggetto o una situazione descritta da un insieme di attributi (features) e restituisce una "decisione".
- Effettua dunque una "classificazione" della situazione presentata in input.

Esempio di applicazione

[valutazione richiesta prestito]

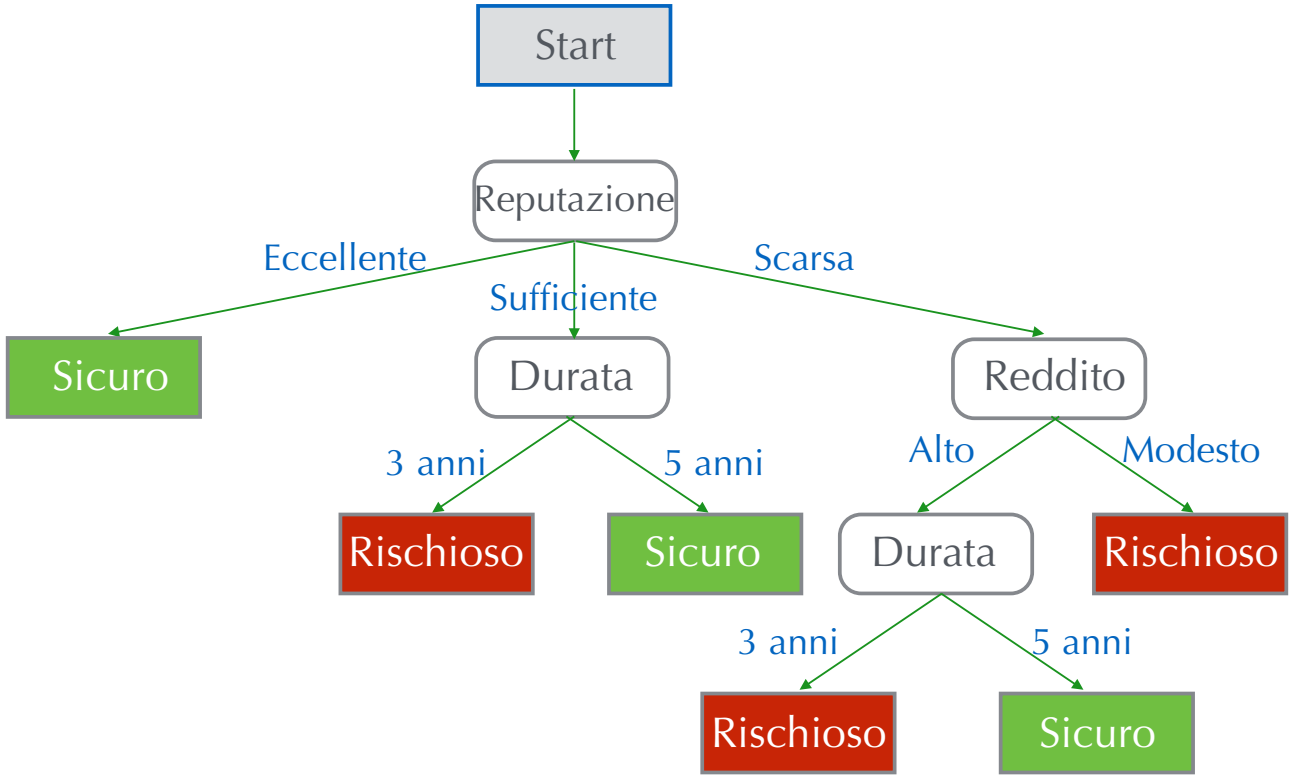
- Vediamo un esempio di applicazione, relativo alla valutazione di richieste di prestito da parte di un cliente alla propria banca:

Esempio di applicazione

- Nel formulare questo problema come un problema di apprendimento dobbiamo anzitutto decidere quali proprietà, o attributi (features), sono disponibili per descrivere esempi (osservazioni) nel dominio.
- In genere, alcune delle caratteristiche prese in considerazione i tali casi sono le seguenti:
  - reputazione cliente (e.g., ha pagato regolarmente vecchi prestiti?)
  - reddito cliente
  - durata prestito
  - altre informazioni personali (età, motivo per il prestito, ecc.)

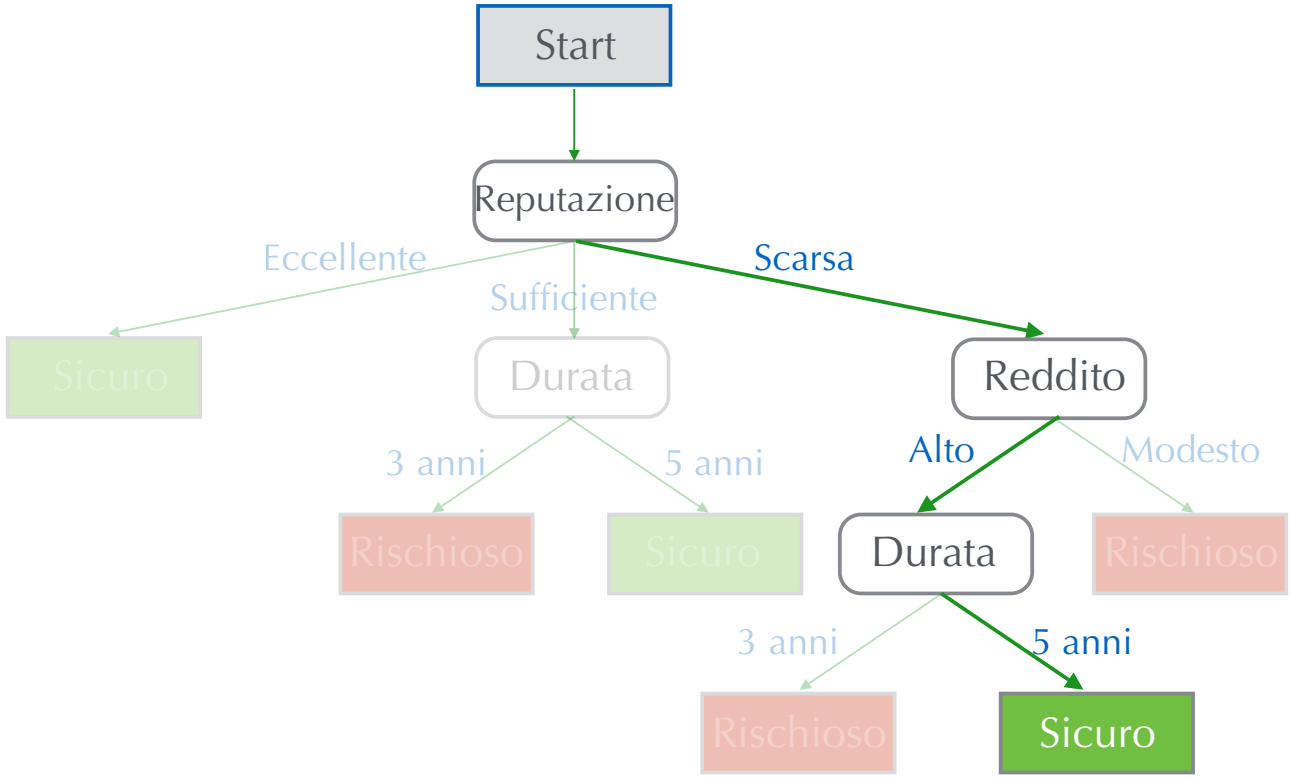
Decision Tree Classifier

Esempio di decision tree per il problema in esame:

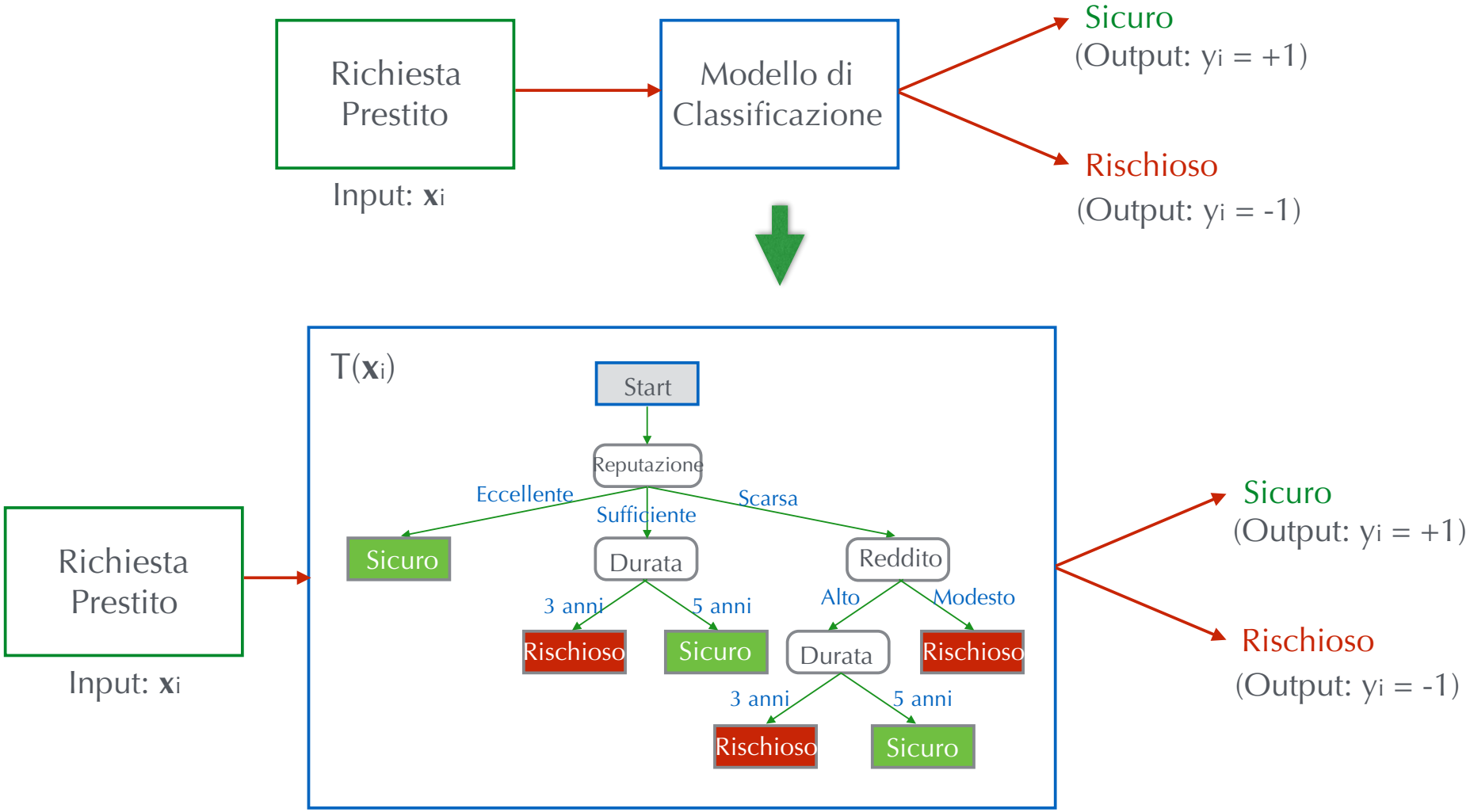


Valutazione richiesta prestito

$\mathbf{x}_i$ = (Reputazione = Scarsa, Reddito = Alto, Durata = 5 anni)



Decision Tree Model

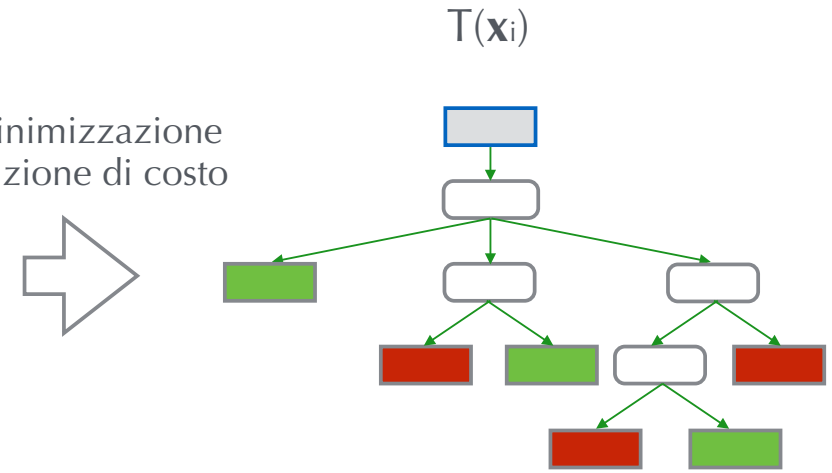


Apprendimento albero dai dati

- Vediamo come sia possibile costruire (ossia apprendere) un decision tree a partire da un certo numero di osservazioni:

Reputazione	Durata	Reddito	y_i
eccellente	3 anni	alto	Sicuro
sufficiente	5 anni	modesto	Sicuro
sufficiente	3 anni	alto	Rischioso
scarso	5 anni	alto	Sicuro
sufficiente	5 anni	modesto	Sicuro
scarso	3 anni	alto	Rischioso
scarso	5 anni	modesto	Rischioso
sufficiente	3 anni	alto	Rischioso
eccellente	3 anni	modesto	Sicuro

Minimizzazione funzione di costo



Metrica di Qualità

[quality metric]

- La metrica che si usa misura la frazione delle previsioni errate fornite dall'albero:

$$\text{Errore} = \frac{\#\text{previsioni\_errate}}{\#\text{esempi}}$$

- Ovviamente:
  - miglior valore possibile: 0.0
  - peggior valore possibile: ?

Learning Goal

- Il nostro obiettivo è dunque quello di costruire un albero di decisione che minimizzi il Classification Error sui dati di training, calcolato mediante la metrica di qualità che abbiamo definita.
- Purtroppo questo è un task estremamente difficile:
  - abbiamo un numero esponenziale di possibili alberi da considerare
  - problema NP-hard
  - possiamo però utilizzare delle euristiche che funzionano bene in pratica

Algoritmo greedy decision tree learning

Vediamo informalmente come poter procedere per costruire un albero di decisione:

- 1. Cominciamo da un albero “vuoto” e consideriamo tutti gli esempi disponibili.
- 2. Selezioniamo la feature “migliore” con la quale possiamo partizionare (split) i dati in base ai diversi valori che essa può assumere.
- 3. Per ogni split:
  - Se non ci sono altre operazioni da fare, costruire foglia con la previsione.
  - Altrimenti, continua la costruzione dell’albero a partire dallo split che stiamo considerando.

Algoritmo greedy decision tree learning

Primo problema:

- 1. Cominciamo da un albero "vuoto". Consideriamo tutti gli esempi disponibili.
- 2. Selezioniamo la feature "migliore" con la quale possiamo partizionare (split) i dati in base ai vari valori che essa può assumere.

feature selection

- 3. Per ogni split:
  - Se non ci sono altre operazioni da fare, costruire foglia con la previsione.
  - Altrimenti, continua la costruzione dell'albero a partire dallo split che stiamo considerando.

Algoritmo greedy decision tree learning

Secondo problema:

- 1. Cominciamo da un albero "vuoto". Consideriamo tutti gli esempi disponibili.
- 2. Selezionamo la feature "migliore" con la quale possiamo partizionare (split) i dati in base ai vari valori che essa può assumere.
- 3. Per ogni split:

stopping conditions

- Se non ci sono altre operazioni da fare, costruire foglia con la previsione.
- Altrimenti, continua la costruzione dell'albero a partire dallo split che stiamo considerando.

Algoritmo greedy decision tree learning

Chiamata ricorsiva:

- 1. Cominciamo da un albero "vuoto". Consideriamo tutti gli esempi disponibili.
- 2. Selezioniamo la feature "migliore" con la quale possiamo partizionare (split) i dati in base ai vari valori che essa può assumere.
- 3. Per ogni split:
  - Se non ci sono altre operazioni da fare, costruire foglia con la previsione.
  - Altrimenti, continua la costruzione dell'albero a partire dallo split che stiamo considerando.

chiamata ricorsiva

Predizioni con Decision Stump

[feature selection]

22 18

Root node: relativo a tutte le osservazioni.

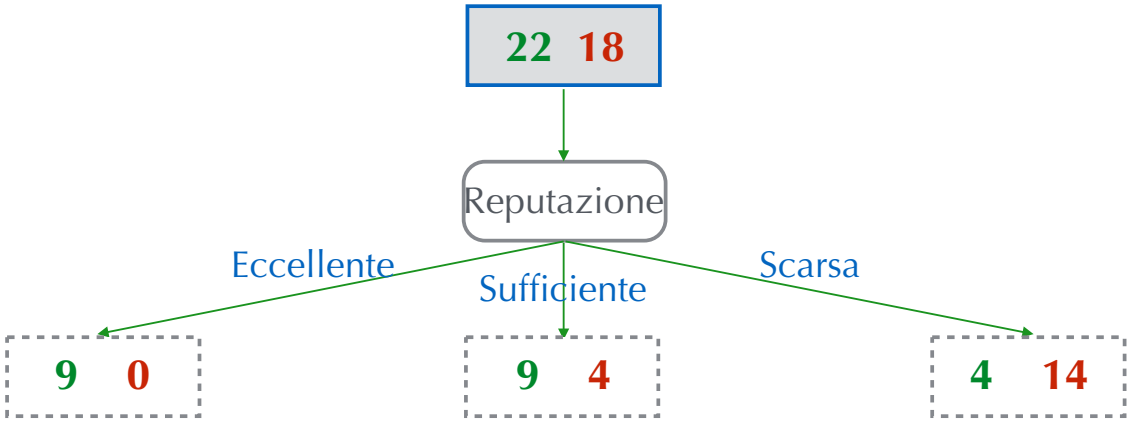
22: output “Sicuro”
18: output “Rischioso”

Ora dobbiamo selezionare una feature
(feature selection problem)

Predizioni con Decision Stump

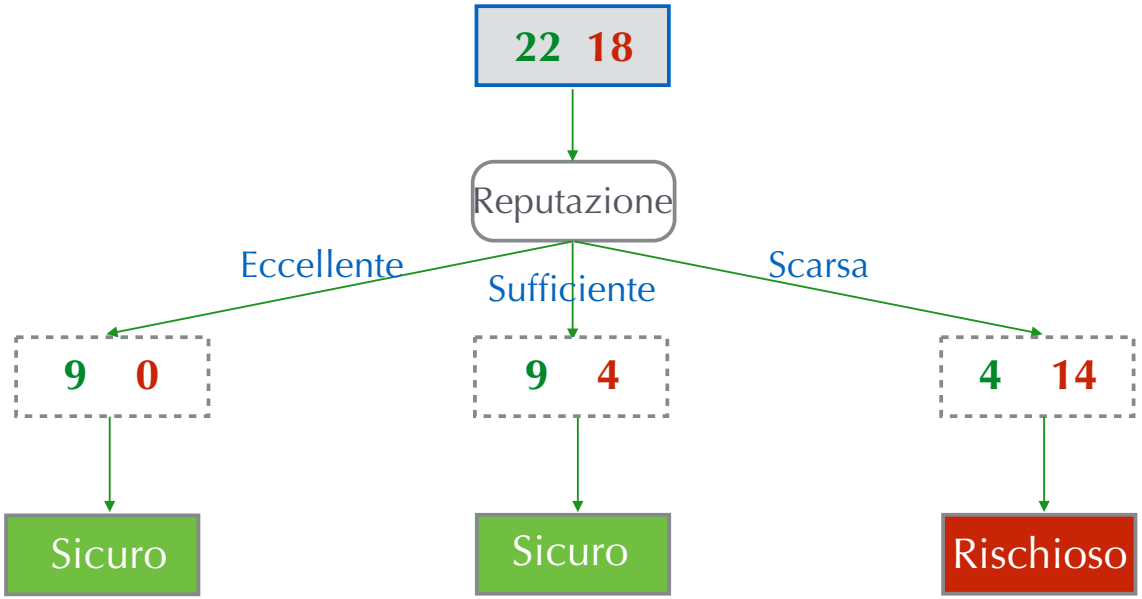
[feature: Reputazione]

Se scegliamo “Reputazione”:



Decision Stump

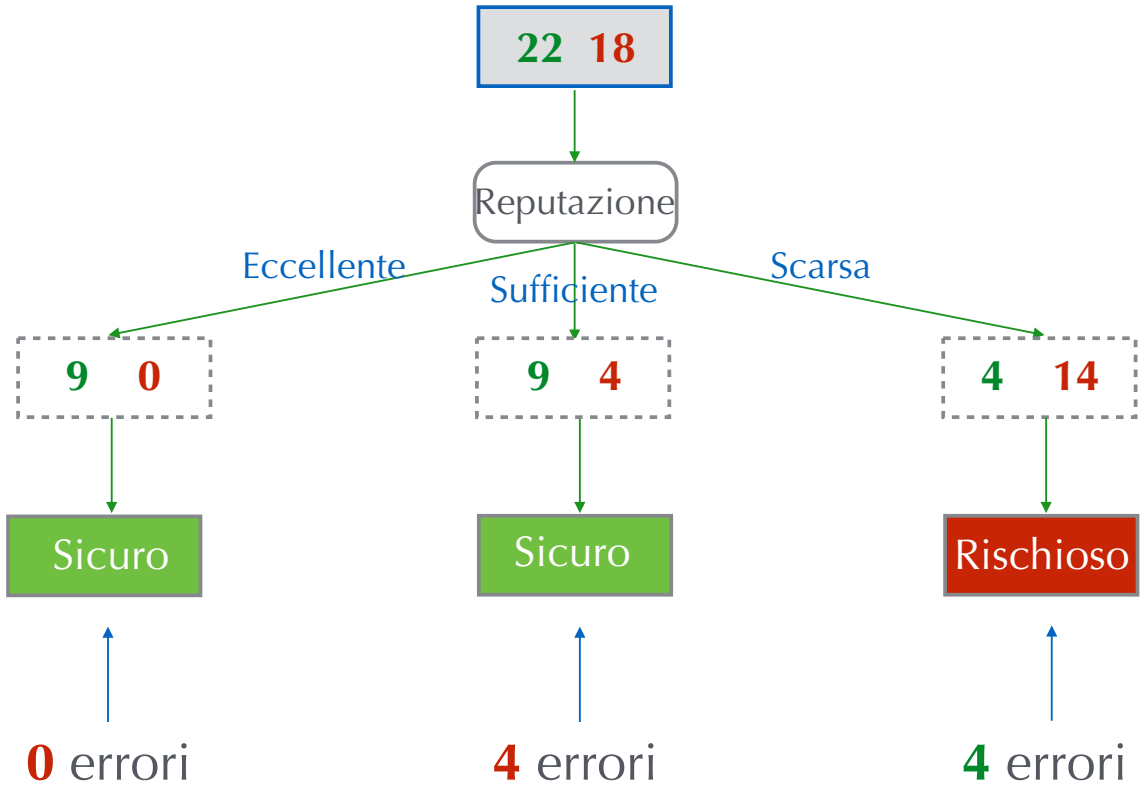
[feature: Reputazione]



set $\hat{y}$ = “majority value”

Decision Stump

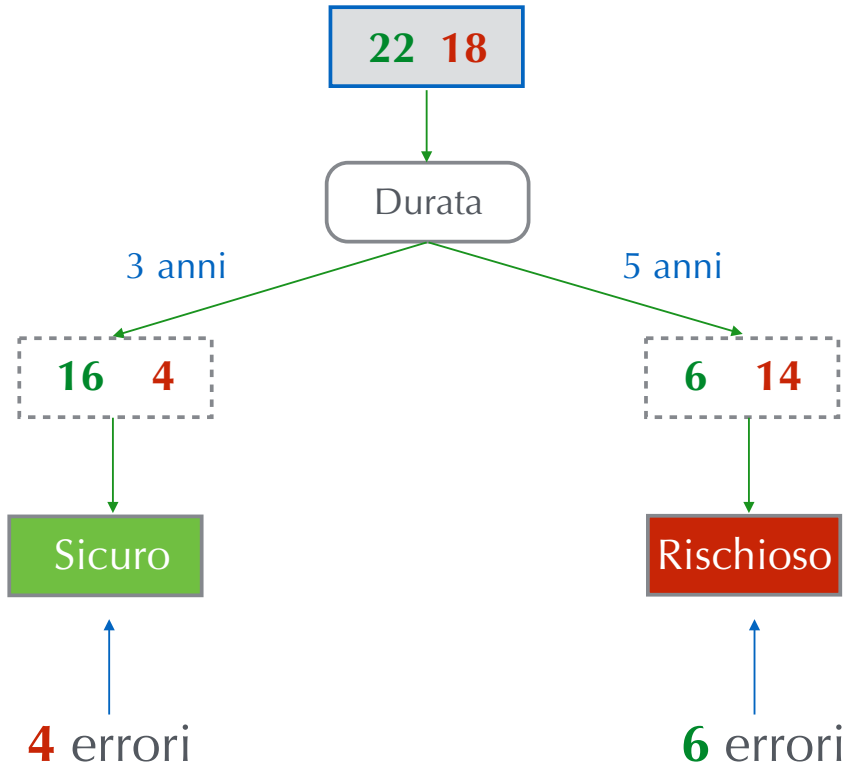
[feature: Reputazione]



Decision Stump

[feature: Durata]

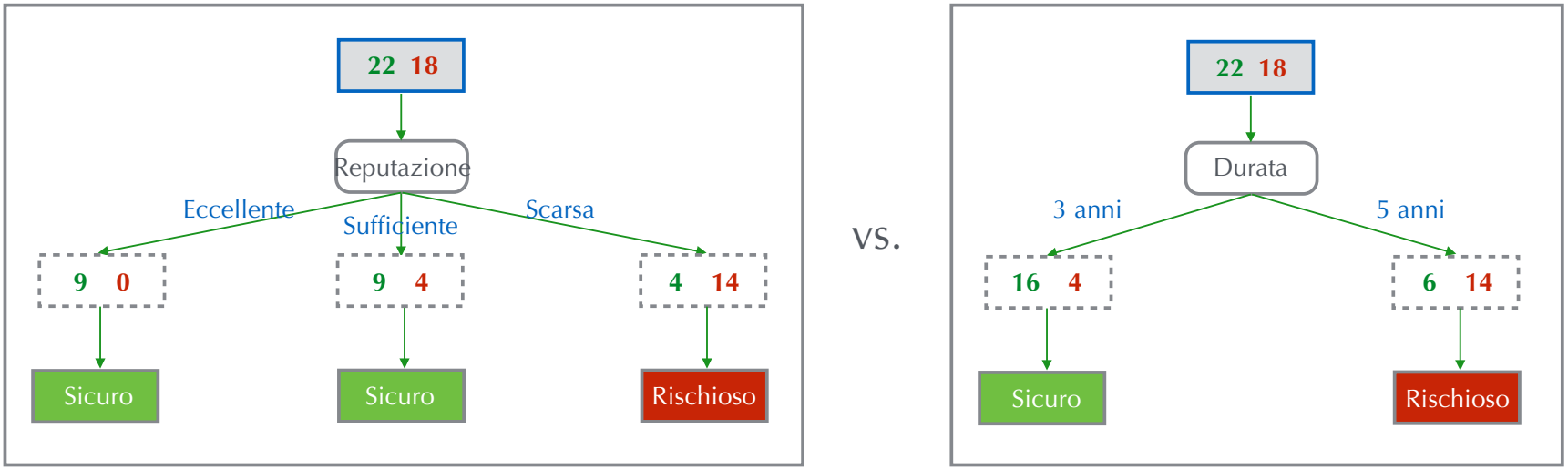
Se scegliamo “Durata”:



Selezione della migliore feature

[feature selection]

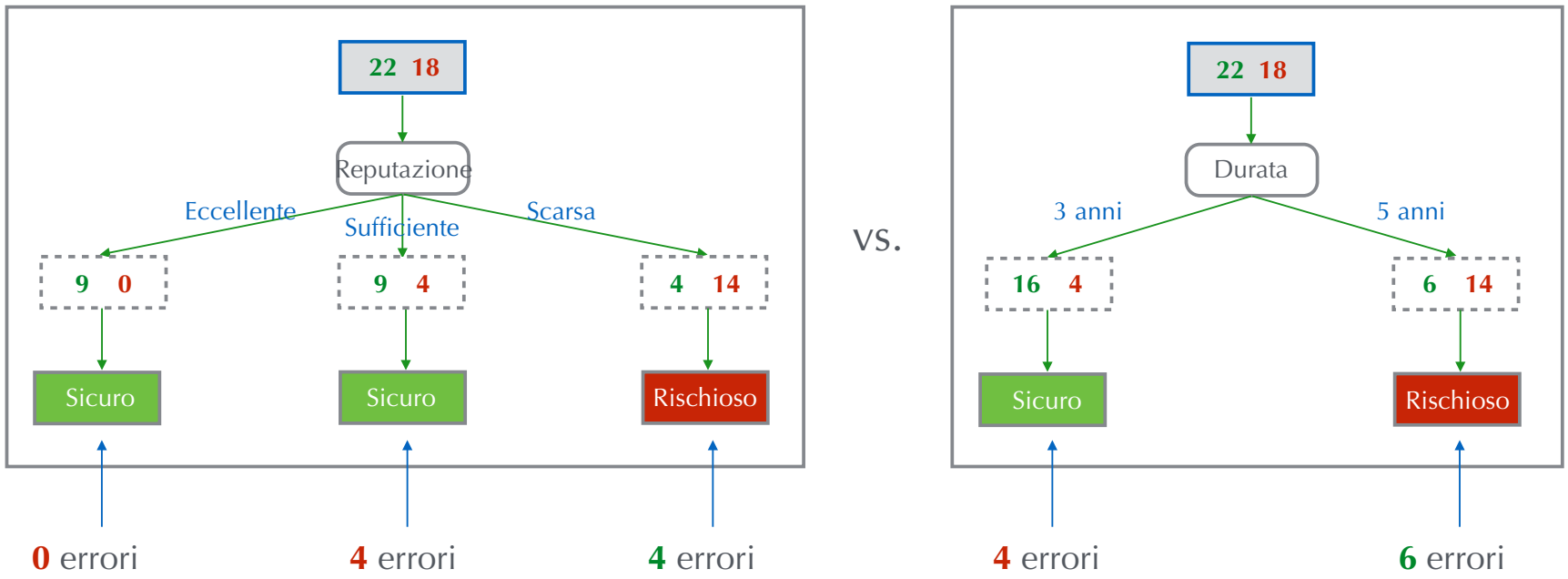
Dobbiamo definire un criterio per la scelta della migliore feature:



Selezione della migliore feature

[feature selection]

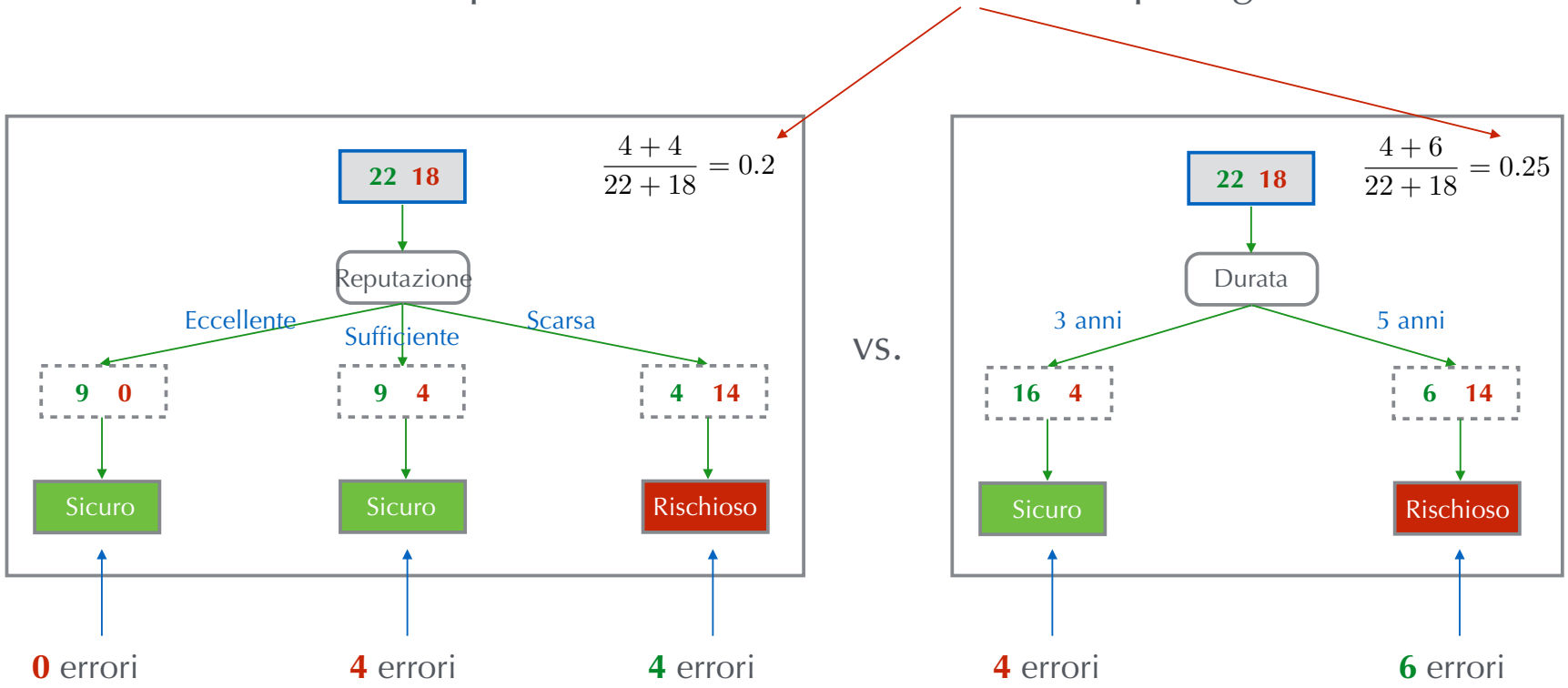
Per far questo consideriamo gli errori già visti in precedenza …….



Selezione della migliore feature

[feature selection]

…… e usiamoli per calcolare il Classification Error per ogni feature:



Scegliamo la feature con il Classification Error più basso.

Calcolo Classification Error

- Abbiamo dunque diviso il calcolo del Classification Error in due fasi:

  1. Per ogni nodo relativo ad un sottoinsieme dei dati, ottenuto considerando uno dei possibili valori della feature d'interesse, assegniamo il valore della majority class del nodo ($\hat{y}$ = "majority class").

  2. Calcolo del Classification Error considerando come predizione per ogni nodo considerato quella assegnata nel passo precedente.

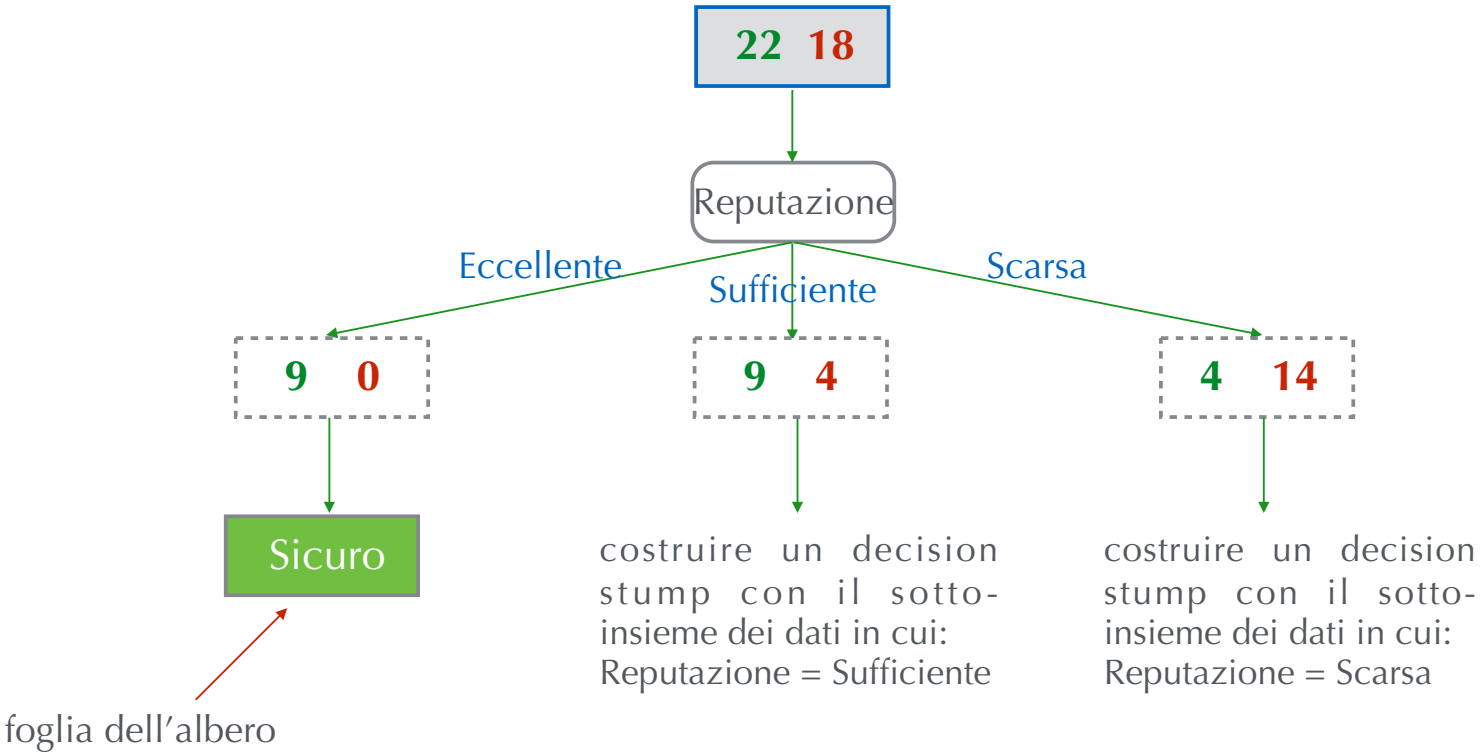
Algoritmo per Feature Split Selection

- Dato un sottoinsieme M di osservazioni disponibili (nodo dell'albero):
  - $\forall$ feature $\phi_j(\mathbf{x})$:
    - Split dei dati M in base ai valori della feature $\phi_j(\mathbf{x})$.
    - Calcolo del Classification Error per il Decision Stump della feature $\phi_j(\mathbf{x})$.
  - Scelta della feature $\phi_{j^*}(\mathbf{x})$ con il Classification Error più basso.

Tree Learning

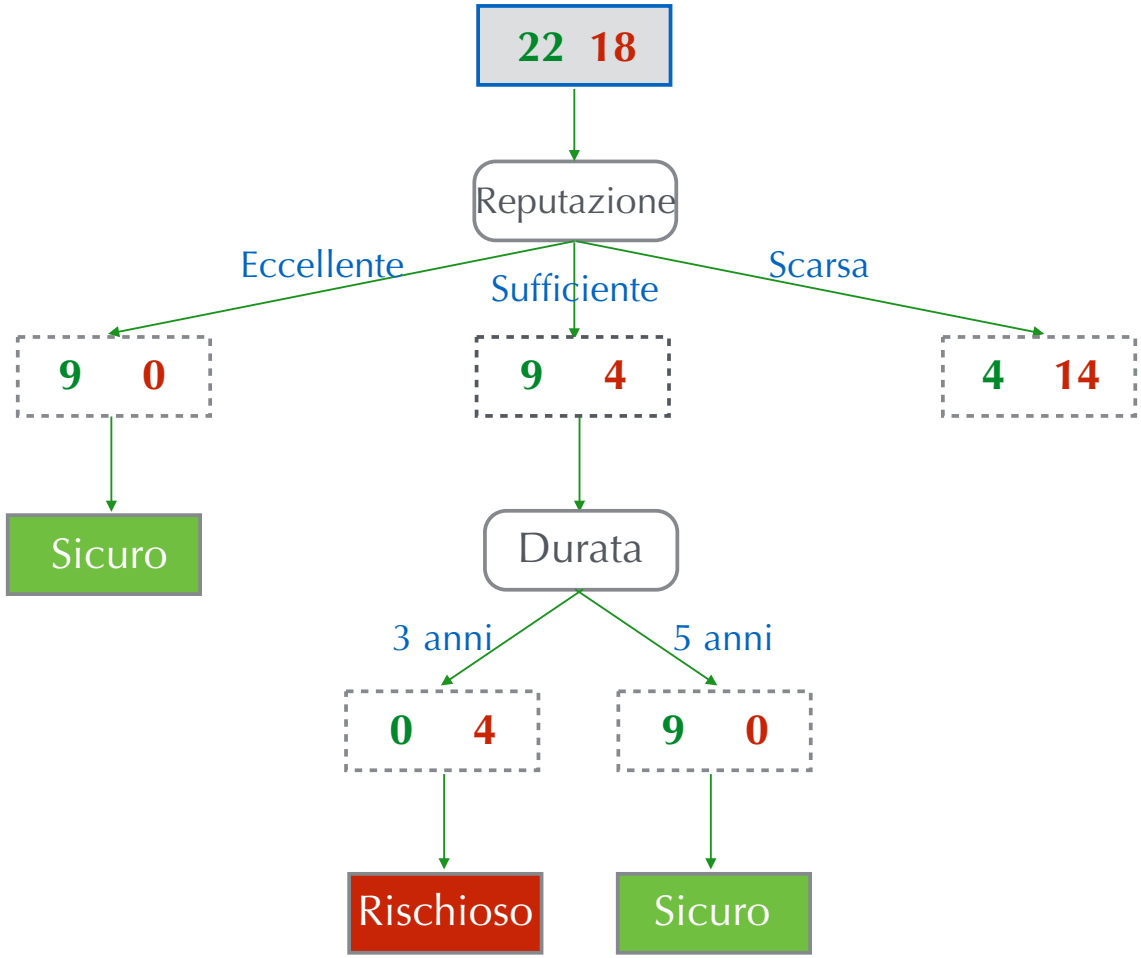
[recursive stump learning]

La costruzione dell'albero si effettua come segue:



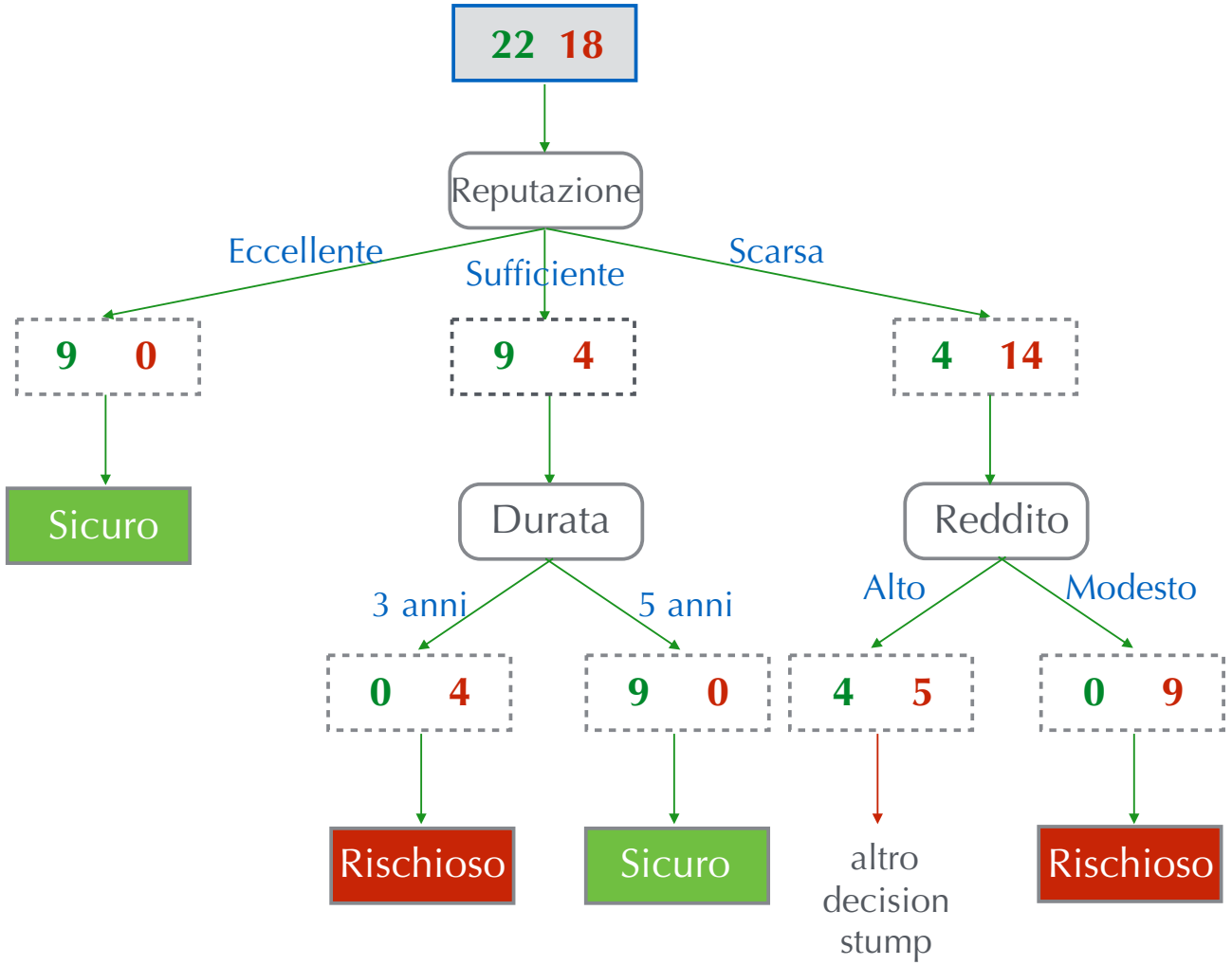
Tree Learning

[secondo livello]



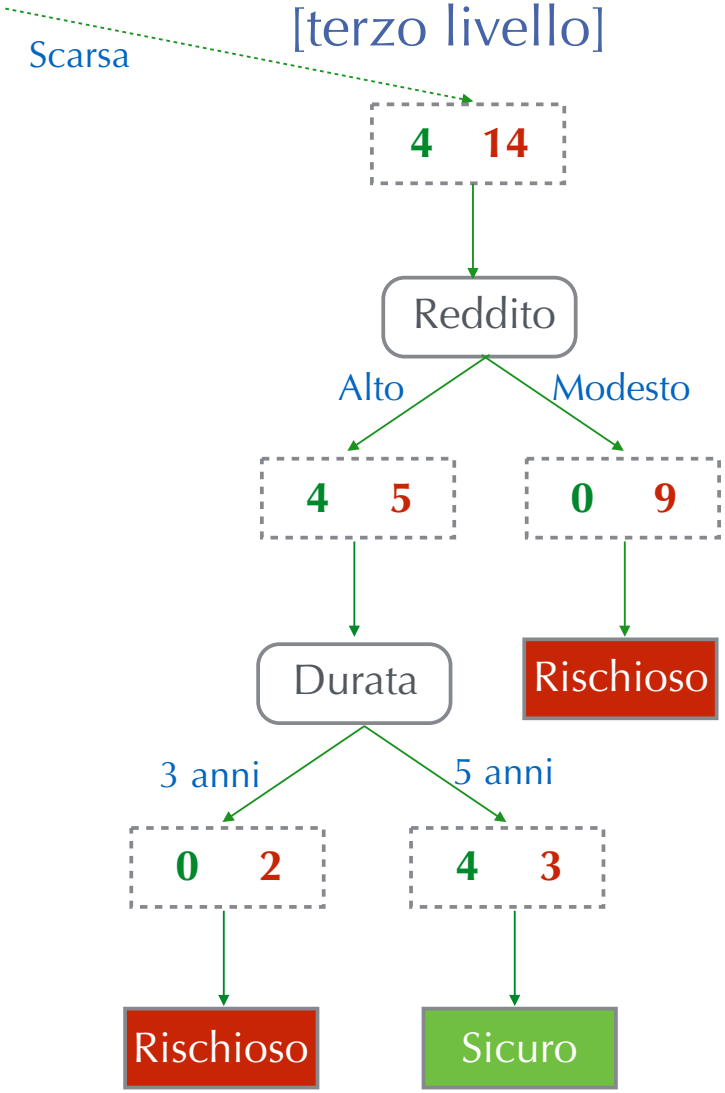
Tree Learning

[secondo livello]



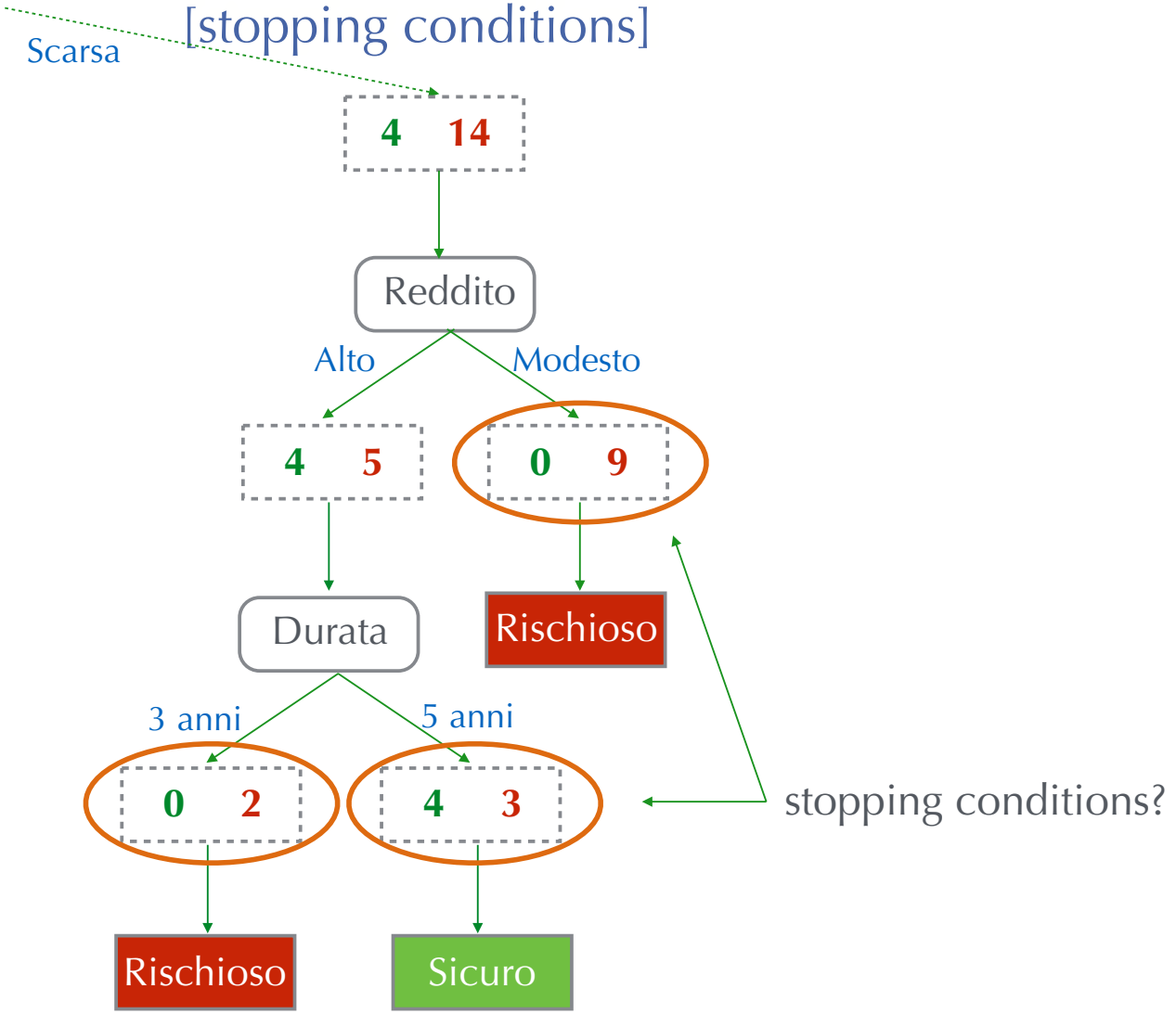
Tree Learning

[terzo livello]



Tree Learning

[stopping conditions]



Stopping Conditions

- La costruzione di un ramo dell'albero si ferma quando arriviamo ad un nodo nel quale si verifica una delle seguenti condizioni:

  1. Gli esempi relativi al nodo sono tutti di uno stesso tipo (e.g., tutti Sicuro o tutti Rischioso): scegliamo come foglia il valore in questione.

  2. Gli esempi relativi al nodo sono di tipo diverso, e non ci sono più feature da considerare: scegliamo come foglia il majority value.

  3. Nel nodo non ci sono più esempi, ma c'è ancora qualche feature non considerata nel percorso che porta a quel nodo: valore di default (e.g., maggioranza nodo genitore).

Algoritmo greedy decision tree learning

decision_tree_learning(nodo)

- 1. Start da un nodo relativo a M esempi
- 2. Feature Selection
- 3. Per ogni split:

  if Stopping Condition

  then: costruire la foglia con la previsione $\hat{y}$

  else: decision_tree_learning(nodo relativo allo split)

Selezione Feature per dividere i dati

Non ci sono altre operazioni da fare

Chiamata Ricorsiva

Algoritmo per fare previsioni mediante Decision Tree

Vediamo ora il semplice algoritmo che implementa la funzione T(**x**), ossia l'algoritmo che, a fronte di un ingresso $\mathbf{x}_i$, visita l'albero di decisione costruito nella fase di training e fornisce in output una previsione $\hat{\mathbf{y}}_i$:

predict(tree_node, input)

if tree_node corrente è una foglia

then: **return** majority class dei punti relativi alla foglia

else:

- next_node = figlio di tree_node il cui valore della feature corrisponde all'input
- **return** predict(next_node, input)

Riferimenti

- Watt, J., Borhani, R., Katsaggelos, A.K. *Machine Learning Refined*, 2nd edition, Cambridge University Press, 2020.
- James, G., Witten, D., Hastie, T., Tibishirani, R. *An Introduction to Statistical Learning*, Springer, 2013.
- Ross, S.M. *Probabilità e Statistica per l'Ingegneria e le Scienze*, 3a edizione, Apogeo, 2015.
- *Machine Learning: Classification*, University of Washington - Coursera, 2017.
- Flach, P. *Machine Learning - The Art and Science of Algorithms that Make Sense of Data*, Cambridge University Press, 2012.