UNIVERSITÀ DEGLI STUDI DI TERAMO

Big Data Analytics

Classificare in KNIME

Prof.ssa Romina Eramo
Università degli Studi di Teramo
Dipartimento di Scienze della Comunicazione
reramo@unite.it

Introduzione alla classificazione

» Dopo il machine learning per predire numeri (regressione), la classificazione ci permette di:

- assegnare automaticamente elementi a **classi predefinite**
- costruire modelli che "riconoscono" categorie nei dati
- applicare ML a decisioni pratiche: rischio, preferenze, comportamenti

L’importanza di dividere in classi

» **Perché classifichiamo?**
- La mente umana semplifica il mondo attraverso categorie
- Le classi permettono di capire, organizzare e prendere decisioni
- Fondamentale sia nella vita quotidiana sia nel lavoro

» **Esempi reali**
- Banca: clienti “aziende” vs “persone”
- Biblioteca: catalogazione per aree tematiche
- Retail: segmenti di clienti per personalizzare promozioni
- Assicurazioni: classi di rischio
- Medicina: categorie di rischio clinico

Classificazioni deterministiche vs probabilistiche

» **Deterministiche**
  - Le regole sono note e chiare
  - Ogni elemento appartiene sicuramente a una classe

» **Probabilistiche**
  - Regole parziali o incerte
  - La macchina stima la **probabilità di appartenenza** alle classi
  - Usate per comportamenti futuri, marketing, propensione all'acquisto, rischio ecc.

Perché implementare un classificatore automatico?

1. **Automazione di attività ripetitive**
   - Classificare elementi uno a uno è lento e costoso
   - Un modello permette classificazione **in tempo reale**
   - Esempio: nuovi utenti registrati → classificazione automatica
2. **Quando le regole NON sono note**
   - I pattern sono complessi o nascosti
   - La macchina trova connessioni invisibili agli umani
   - Esempio: rilevamento frodi con carte di credito

Apprendimento supervisionato per classificare

» **Perché supervised learning?**
  - Una macchina può imparare le classi a partire da esempi già etichettati.

» **Processo (identico alla regressione):**
  - **Partitioning** → divisione training/test
  - **Learning** → apprendimento dei parametri
  - **Prediction** → assegnazione delle classi sul test set
  - **Scoring** → valutazione della classificazione

» **Target**
  - È una **variabile categorica** (classe)

Alberi decisionali

» Un **albero decisionale (decision tree)** è una rappresentazione grafica di un processo decisionale:

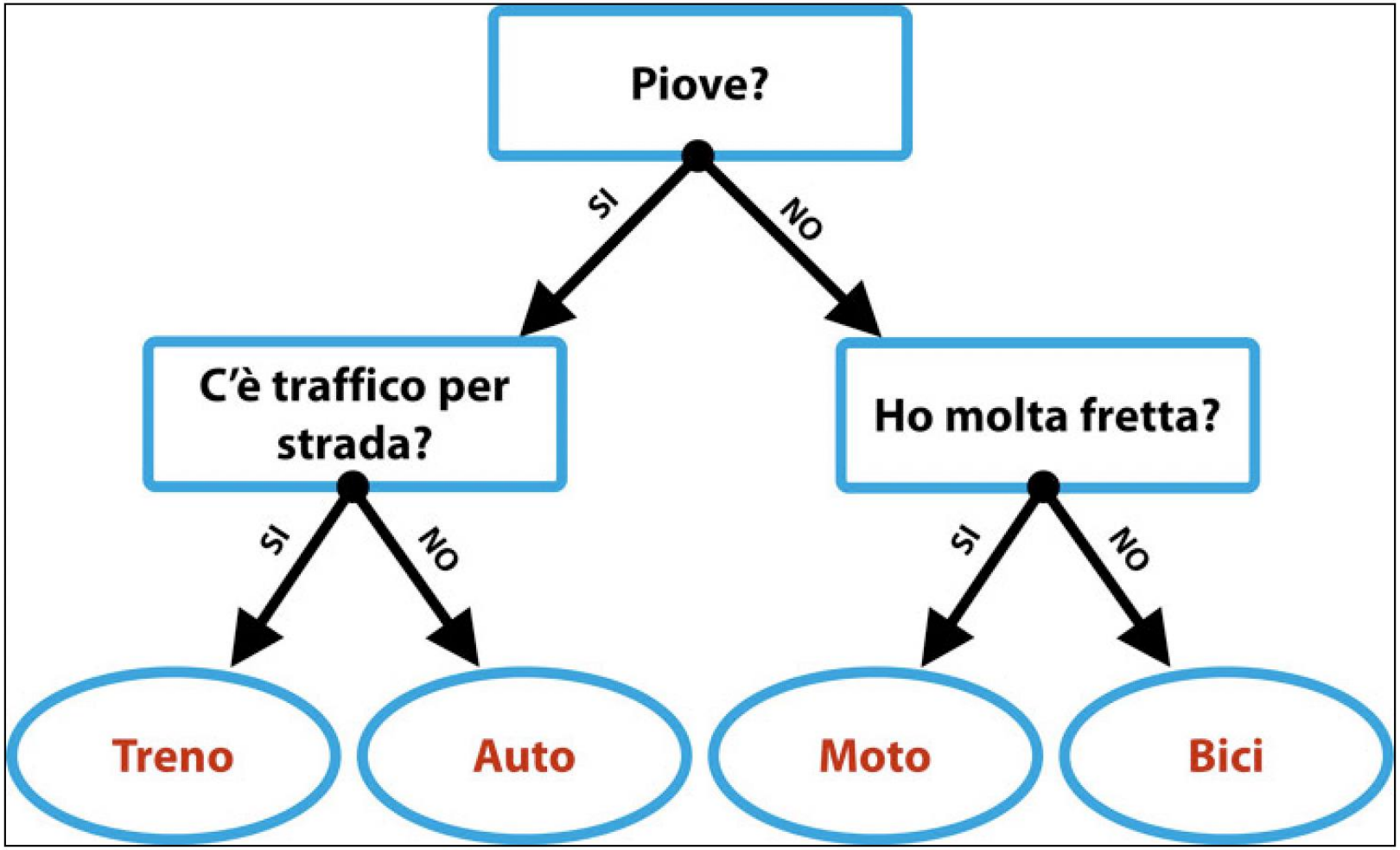
- Ogni **nodo** = una domanda
- Ogni **ramo** = una risposta
- Ogni **foglia** = una decisione o una classe assegnata

» Quando la domanda è:

👉 *"A quale classe appartiene questo elemento?"*

- l'albero diventa un **modello di classificazione**.

Esempio: Scegliere il mezzo di trasporto



Come “impara” un albero

» **Algoritmi più comuni**
  - CHAID, ID3, C4.5, CART, C5.0 …

» **Come funzionano**
  - Partono dalla **radice**
  - Testano possibili modi di dividere i dati
  - Scelgono lo **split migliore** (domanda migliore)
  - Ripetono il processo fino alle foglie
  - Si fermano quando non si può più migliorare la purezza delle classi

Scelta della miglior suddivisione

» Per decidere **quale domanda porre** a ogni livello, il modello valuta misure di impurità:

Indice di Gini

$$I_G = 1 - \sum p_i^2$$

Entropia

$$H = -\sum p_i \log p_i$$

» Obiettivo: scegliere lo split che genera la **classificazione più "pura"**.

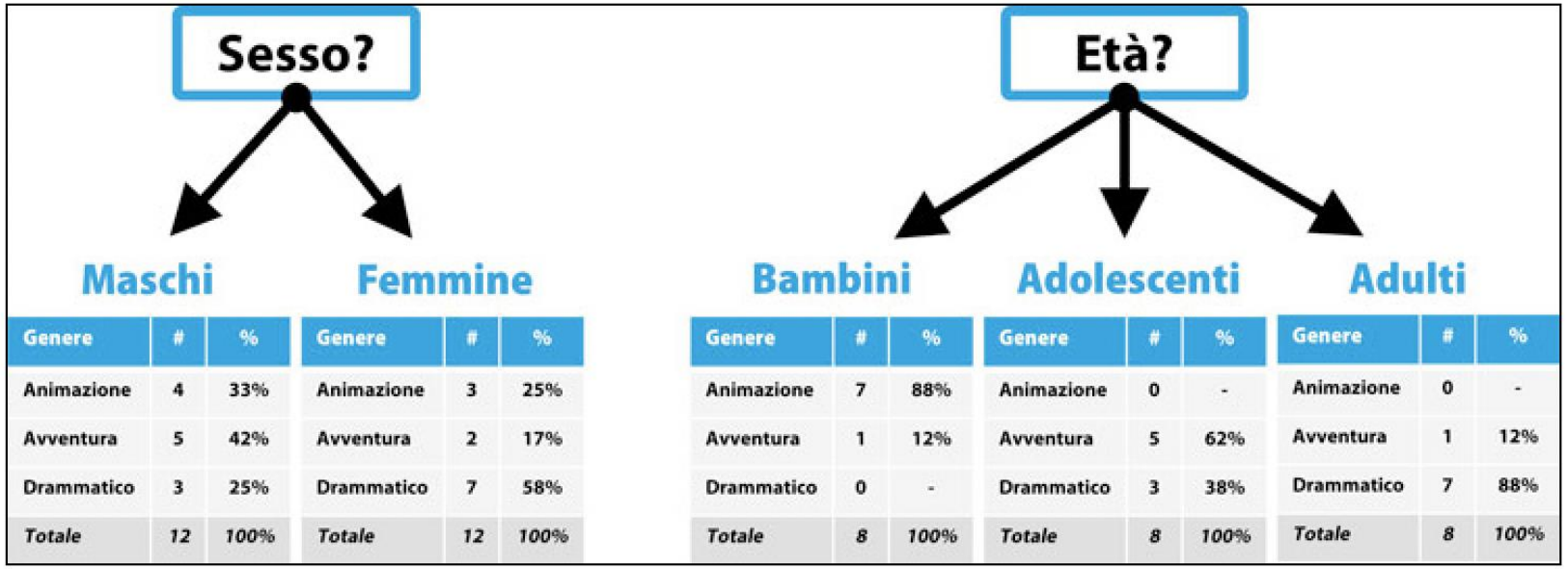
Scelta della miglior suddivisione
Esempio: preferenze cinematografiche

» Dataset con 24 persone → predire il genere preferito
  – Preferenze descritte per sesso e classe di età

Sesso	Età	Genere preferito	Sesso	Età	Genere preferito
M	Bambino	Animazione	M	Bambino	Animazione
M	Adolescente	Avventura	M	Adolescente	Avventura
M	Adulto	Avventura	M	Adulto	Drammatico
M	Bambino	Animazione	M	Bambino	Animazione
M	Adolescente	Avventura	M	Adolescente	Avventura
M	Adulto	Drammatico	M	Adulto	Drammatico
F	Bambino	Animazione	F	Bambino	Animazione
F	Adolescente	Avventura	F	Adolescente	Drammatico
F	Adulto	Drammatico	F	Adulto	Drammatico
F	Bambino	Animazione	F	Bambino	Avventura
F	Adolescente	Drammatico	F	Adolescente	Drammatico
F	Adulto	Drammatico	F	Adulto	Drammatico

Variabili numeriche negli alberi

» Gli alberi gestiscono anche variabili continue:
- Suddividendo tramite **soglie** (es. età < 15, ≥ 15, ≥ 35…)
- Gli split possono variarsi **più volte** lungo l'albero
- Le soglie vengono scelte usando Gini/Entropia



Genere	#	%
Animazione	4	33%
Avventura	5	42%
Drammatico	3	25%
Totale	*12*	*100%*

Genere	#	%
Animazione	3	25%
Avventura	2	17%
Drammatico	7	58%
Totale	*12*	*100%*

Genere	#	%
Animazione	7	88%
Avventura	1	12%
Drammatico	0	-
Totale	*8*	*100%*

Genere	#	%
Animazione	0	-
Avventura	5	62%
Drammatico	3	38%
Totale	*8*	*100%*

Genere	#	%
Animazione	0	-
Avventura	1	12%
Drammatico	7	88%
Totale	*8*	*100%*

Scelta della miglior suddivisione
Esempio: preferenze cinematografiche

» Dataset con 24 persone → predire il genere preferito.
» Possibili primi split:
  – **Per sesso**
  – **Per età**
» **Calcolo numerico della scelta migliore**

Split per sesso	Split per età
IG(maschi) ≈ 0.65 IG(femmine) ≈ 0.57 Media pesata ≈ 0.61	Media degli indici ≈ 0.30

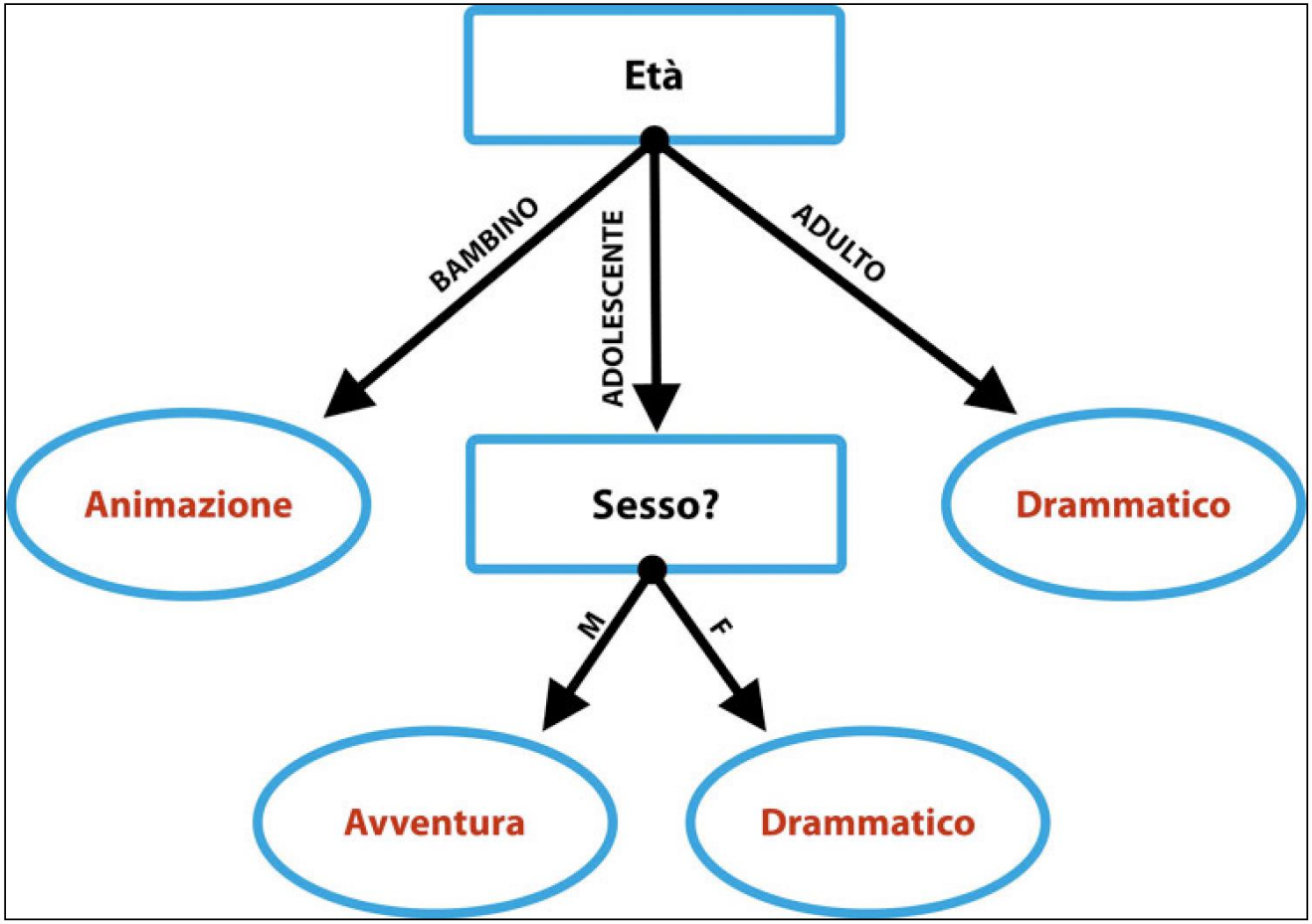
👉 **Meglio lo split per età**: produce rami più "puri"

Pruning

» Molti algoritmi includono la **potatura**:
  - Rimozione di rami inutili
  - Modellazione più semplice
  - Minore rischio di overfitting
  - Albero più comprensibile per l'utente

» Esempio: se il sesso non cambia le preferenze dopo un certo livello → il ramo viene potenziato.

Pruning


Età
BAMBINO
ADOLESCENTE
ADULTO
Animazione
Sesso?
Drammatico
M
F
Avventura
Drammatico

Tutorial: Predire chi cambierà operatore

» Un grande operatore di telecomunicazioni sta perdendo clienti in modo continuo.

» La direzione vuole:

- Capire le **cause sistemiche** di insoddisfazione
- Attivare **offerte mirate** per utenti a rischio migrazione

» **Vincoli**

- Non è possibile abbassare ulteriormente le tariffe
- Occorre lavorare con i dati disponibili per migliorare la *customer retention*

Dataset

- *CustomerID* — identificativo univoco
- *Gender* — Male / Female
- *SeniorCitizen* — 1 se cliente ≥ 65 anni
- *Partner* — Yes / No
- *Dependents* — Yes / No
- *Tenure* — mesi di permanenza
- *PhoneService* — servizio voce attivo
- *MultipleLines* — più linee telefoniche?
- *InternetService* — DSL / Fiber optic / No
- *OnlineSecurity*, *OnlineBackup*, *DeviceProtection*
- *TechSupport*
- *StreamingTV*, *StreamingMovies*
- *Contract* — Month-to-month / One year / Two year
- *PaperlessBilling* — Yes / No
- *PaymentMethod* — carte, RID, bonifico ecc.
- *MonthlyCharges* — costo mensile
- *TotalCharges* — storico totale
- **Churn** — variabile target: Yes / No

- Dataset disponibile qui: https://www.kaggle.com/datasets/blastchar/telco-customer-churn

Obiettivo del modello Churn

» Vogliamo Costruire una macchina che sappia **riconoscere automaticamente** se un cliente:
  - è **a rischio migrazione** (*Churn = Yes*)
  - oppure **rimarrà attivo** (*Churn = No*)

» **Perché un albero decisionale?**
  - È **interpretabile**, passo per passo
  - È **ideale per spiegare** il processo di decisione al Commercial Retention Team
  - Può essere riutilizzato per classificare **tutti i clienti**, presenti e futuri

Svolgimento

1. **Importiamo i file.csv**
   - Drag & drop dei due file
   - File Reader riconosce correttamente intestazioni e formato
2. **Prime analisi esplorative**
   - Usiamo il nodo **Statistics** per capire:
   - distribuzioni (es. *Tenure*)
   - composizione del dataset
   - presenza di valori mancanti
3. **Gestione dei valori mancanti**
   - Usiamo il nodo **Missing Value**:
     - String → *Remove Row*
     - Integer → *Remove Row*
     - Double → *Remove Row*

Perché è fondamentale stratificare (classificazione)

» Nella classificazione bisogna mantenere la **proporzione originale** delle classi nel training e nel test set.

» **Esempio intuitivo:**

- Dataset di 1.000 pazienti
- Solo il **3%** è malato
- Se il training set contiene solo 1–2 malati → il modello non impara nulla

» **Soluzione:**

👉 **Stratified Sampling**

- Assicura che *Churn=Yes* e *Churn=No* siano distribuiti proporzionalmente.

Come funziona lo stratified sampling

» Il campionamento avviene **classe per classe**

» Mantiene le proporzioni originali tra training e test

» Evita risultati distorti su **accuracy, precision, recall, confusion matrix**

- 💡 È la metodologia consigliata per TUTTI i problemi di classificazione.

Nodo Partitioning (Table Partitioner)

» **Impostazioni consigliate:**
  - **Relative[%] = 80** → training set
  - **Stratified sampling** sulla colonna **Churn**
  - **Use random seed** → 123456 (per riproducibilità)
» **Risultato:**
  - Stesso rapporto *Churn=Yes / Churn=No* nel training e test set
  - Workflow ripetibile e coerente
» **Regole per dataset sbilanciati (nota)**
  - Quando la classe minoritaria è molto piccola:
    » Undersampling
    » Oversampling
    » **SMOTE** (Synthetic Minority Oversampling Technique) I
      - In KNIME: Manipulation → Column → Transform

Nodo Decision Tree Learner

» Il **Decision Tree Learner** costruisce un **albero decisionale per la classificazione**, imparando dai dati di training.

» Parametri fondamentali

- **Class column**
  - » La Colonna target deve essere **nominale, n**el nostro caso: **Churn**
- **Quality measure**
  - » Criterio per scegliere il miglior split, opzioni:

Gini index	Gain ratio
Misura l'**impurità** delle classi Più è basso → classificazione più "pura"	Basato sulla **riduzione dell'entropia** Misura quanta incertezza viene eliminata con uno split

» 👉 Entrambe sono valide e ampiamente utilizzate. Nella maggior parte dei casi producono alberi simili.

Nodo Decision Tree Learner

- Pruning (potatura dell'albero)
  - » Metodi disponibili

MDL (Minimum Description Length)
Bilancia semplicità e capacità predittiva

Reduced Error Pruning
Taglia rami che non migliorano l'accuratezza

- Min number records per node
  - » Imposta il **numero minimo di esempi** necessari per creare uno split.
    - Valori bassi → albero molto profondo (rischio overfitting)
    - Valori alti → albero più semplice (rischio underfitting)

    👉 Serve per trovare il **giusto equilibrio** tra accuratezza e generalizzazione.

Nodo Decision Tree Learner

- **Overfitting negli alberi decisionali**
  - Se: *Min number records per node = 1*
    - nessun pruning attivo
  - Allora:
    - albero estremamente complesso
    - accuratezza altissima sul training set
    - accuratezza bassa sul test set
  - 👉 **Caso tipico di overfitting**

Output del Decision Tree Learner

» Un unico output (quadrato blu): **modello di albero decisionale**

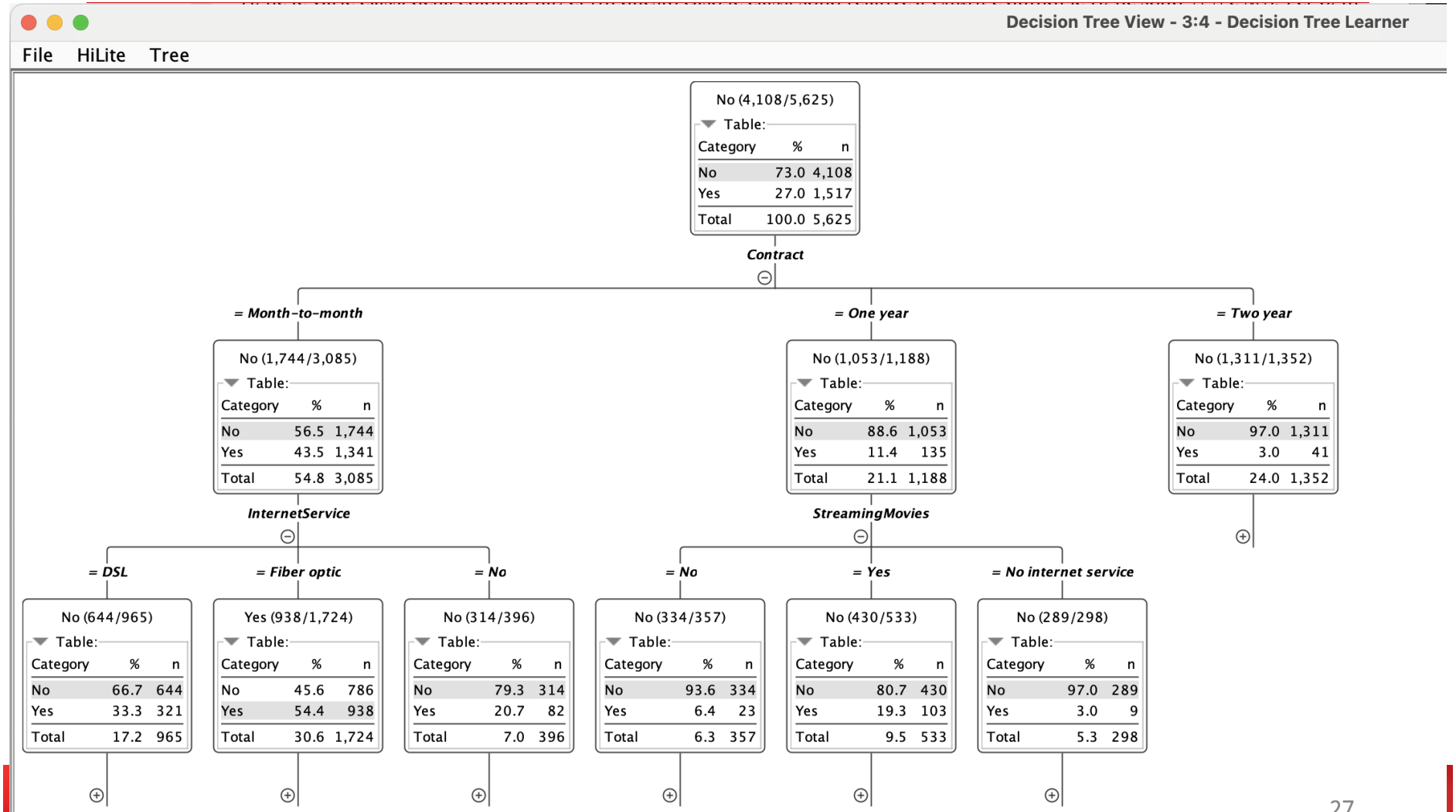
» **Come visualizzarlo:** Selezionare il nodo → *View* → *Decision Tree View*

» Ogni nodo contiene:
- classi (*Yes / No*)
- numero di elementi
- percentuali

» La **classe maggioritaria** è evidenziata in grigio

» **Nodo radice: m**ostra la distribuzione complessiva di *Churn* nel training set.

Output del Decision Tree Learner



Output del Decision Tree Learner

- **Variabile più discriminante: Contract**
  - 👉 All'aumentare della durata del contratto **diminuisce la probabilità di churn**
- **Esempi di risultati significativi**
  - Contratto **Two year**: solo **3.1%** di churn
  - Contratto **Month-to-month**: **42.8%** di churn
  - 👉 La durata del contratto è un **fortissimo predittore**
- Espandendo il ramo *Month-to-month*:
  - Lo split successivo riguarda **InternetService**
  - I clienti con **Fiber optic**:
    - più churn (*55.2% Yes*) che non churn
  - 👉 Il tipo di servizio Internet è un ulteriore fattore critico
- **Valutazione qualitativa dell'albero**
  - L'albero ottenuto è **coerente con il dominio** evidenzia fattori chiave:
    - durata del contratto
    - tipo di servizio Internet
  - 👉 Ottimo punto di partenza per il modello di customer retention

Decision Tree Predictor

» Il **Decision Tree Predictor** applica l'albero decisionale appreso dal learner per:
  - classificare tutte le righe in input
  - assegnare a ciascun cliente una **classe predetta** (*Churn = Yes / No*)

» **Input richiesti**
  - **Modello** (albero decisionale dal learner – quadrato blu)
  - **Dataset di test** (dal Partitioning)

» **Output**
  - Dataset di test arricchito con:
  - **Prediction (Churn)** → classe predetta

Probabilità di appartenenza alle classi

» **Opzione avanzata: Append columns with normalized class distribution**
  - Aggiunge una colonna per ogni classe
  - Riporta la **probabilità di appartenenza** alla classe

» **Esempio**
  - Se una foglia contiene: 75% *Yes e* 25% *No*

  → Tutte le righe che arrivano a quella foglia:
    » sono classificate come *Yes*
    » hanno probabilità *Yes = 75%, No = 25%*

» **Attenzione al concetto di probabilità: ⚠ Probabilità ≠ certezza assoluta**
  - Anche se una foglia contiene solo *Yes* nel training
  - Non significa che la predizione sia corretta al 100%

  👉 I modelli sono **probabilistici**

  👉 Le certezze assolute sono rare nel ML

Nodo Scorer

» Il nodo **Scorer** confronta:
  - **First Column** → classe reale (*Churn*)
  - **Second Column** → classe predetta (*Prediction (Churn)*)

» **Output**
  - **Confusion Matrix**
  - **Metriche di valutazione**:
    - » accuracy
    - » sensitivity (recall)
    - » precision
    - » ecc.

Risultati del primo modello

- **Valori osservati**
  - Classificazioni corrette: **1.050**
    - 201 *True Positive*
    - 849 *True Negative*
  - Classificazioni errate: **356**
    - 172 *False Negative*
    - 184 *False Positive*
- **Accuracy: 74,7%**
  - buon punto di partenza, albero interpretabile
- **Recall: 0,54**
  - Il modello riconosce **solo il 54%** dei clienti che stanno per lasciare
  - Molti clienti a rischio **non vengono intercettati!**

Warning sui missing value

» Alcune righe del test set non vengono classificate

» Motivo:

- combinazioni di valori **mai viste nel training set**

» **Impatto**

- Fenomeno raro
- Coinvolge una sola riga
- Non preoccupante in questa fase

Analisi dell'albero decisionale

» Osservando l'albero: molti livelli (>10), foglie con pochissimi esempi (anche 2 clienti), rami molto specifici

👉 Segnali evidenti di **overfitting**

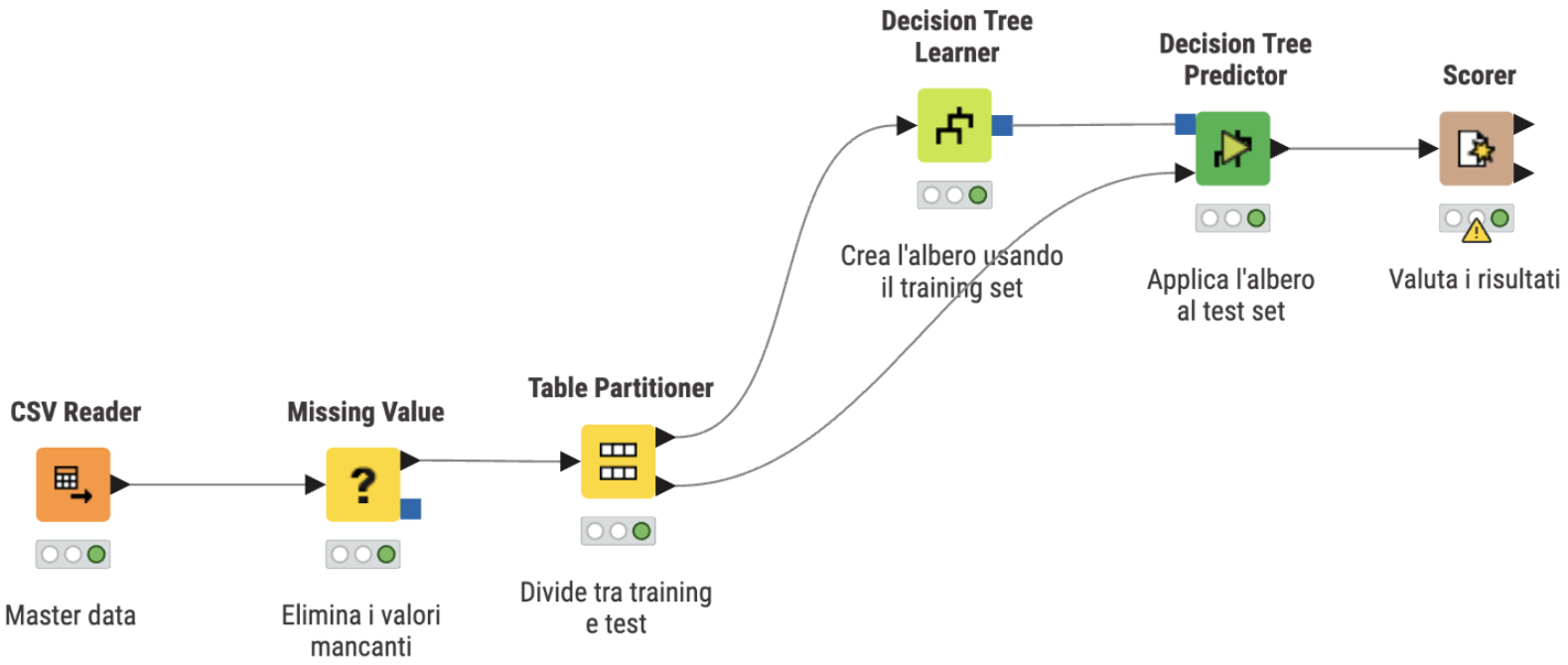
» **Prossimo passo**

- Migliorare la **generalizzazione**
- Aumentare la **sensitivity**
- Ridurre la **complessità dell'albero**

👉 Intervenendo sugli **iperparametri di pruning**

- potatura MDL
- aumento del minimo numero di record per nodo

Workflow (parziale)



Random Forest

» La **Random Forest** è un modello di **ensemble learning** che combina molti modelli "semplici" (alberi decisionali) per ottenere:

- maggiore **accuratezza**
- maggiore **robustezza**
- migliore **generalizzazione**

👉 *Più alberi possono fare una foresta.*

Ensemble learning

» **Apprendimento d'insieme**
  - Consiste nell'aggregare le predizioni di più modelli di base.

» **Tecniche principali**
  - **Bagging** (bootstrap aggregation)
  - **Boosting**
  - **Stacking**

» La Random Forest si basa sul **bagging**.

Bagging: come funziona

» Nel caso della classificazione:

1. Dal training set originale si creano **N sottoinsiemi**
2. I sottoinsiemi sono generati con **campionamento casuale con rimpiazzo** (*bootstrap*)
3. Ogni sottoinsieme allena un **albero decisionale diverso**
4. Le predizioni finali sono aggregate tramite **votazione a maggioranza**

» **Esempio**

- 65% degli alberi → classe A
- 35% degli alberi → classe B

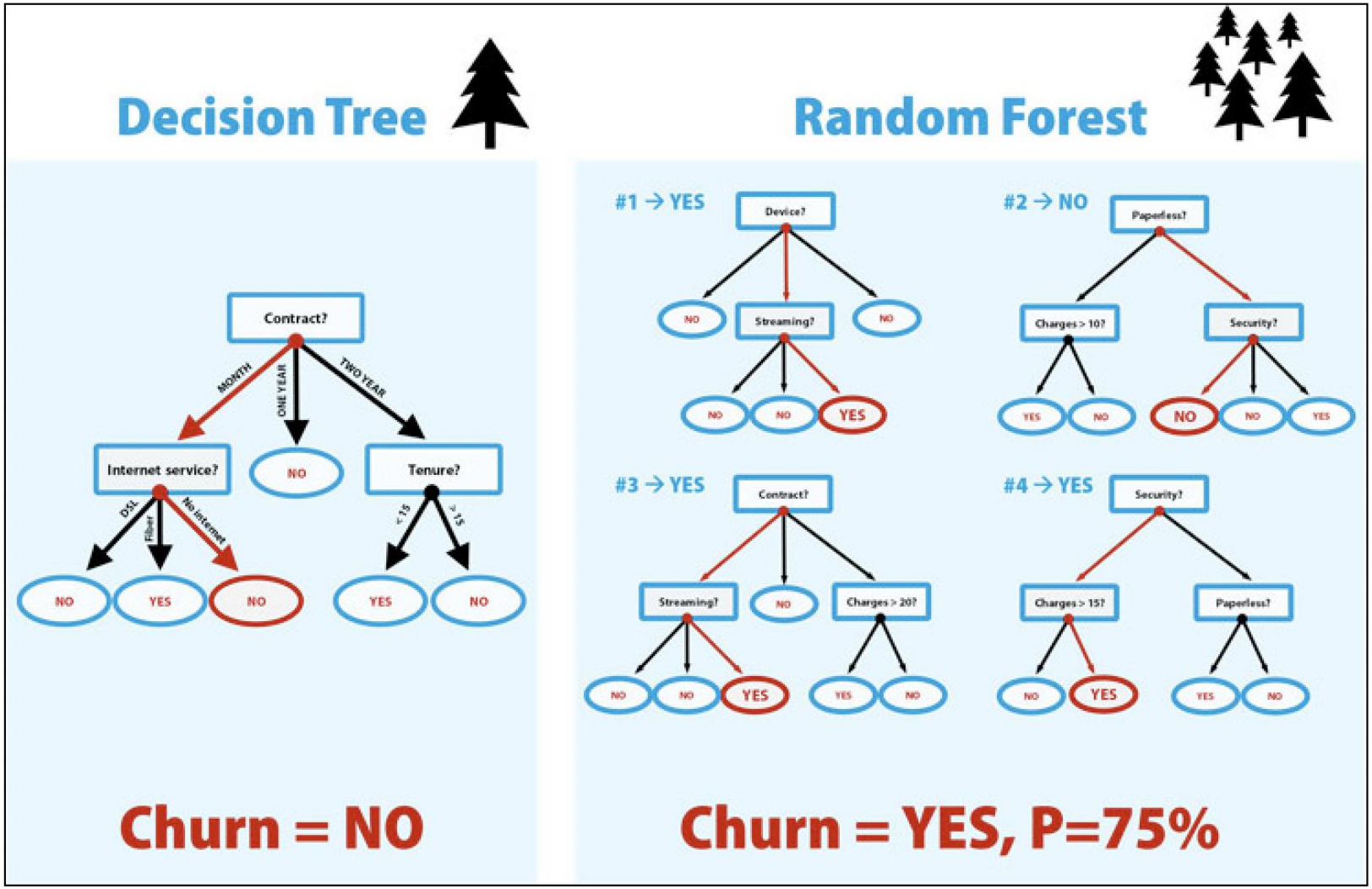
👉 Risultato finale: **classe A**

Cosa rende “random” la Random Forest

» Rispetto al bagging tradizionale:
  - ogni albero usa **solo un sottoinsieme casuale dei predittori**
  - non tutti gli alberi vedono le stesse variabili

» **Perché è utile**
  - alberi più **diversi** tra loro
  - riduzione della correlazione
  - migliore capacità di cogliere pattern sottili

Dal decision tree alla random forest



Vantaggi e limiti della Random Forest

✓ **Vantaggi**

- maggiore **accuratezza**
- migliore **sensibilità**
- robusta al rumore
- meno overfitting rispetto a un singolo albero

» ⚠ **Limiti**

- modello **meno spiegabile**
- difficile da “raccontare” nel dettaglio
- interpretazione indiretta

👉 Compromesso: **accuratezza vs interpretabilità**

Perché usare la Random Forest nel churn

» Nel nostro caso:
- il **Decision Tree** è già comprensibile
- il business ha capito le cause principali del churn

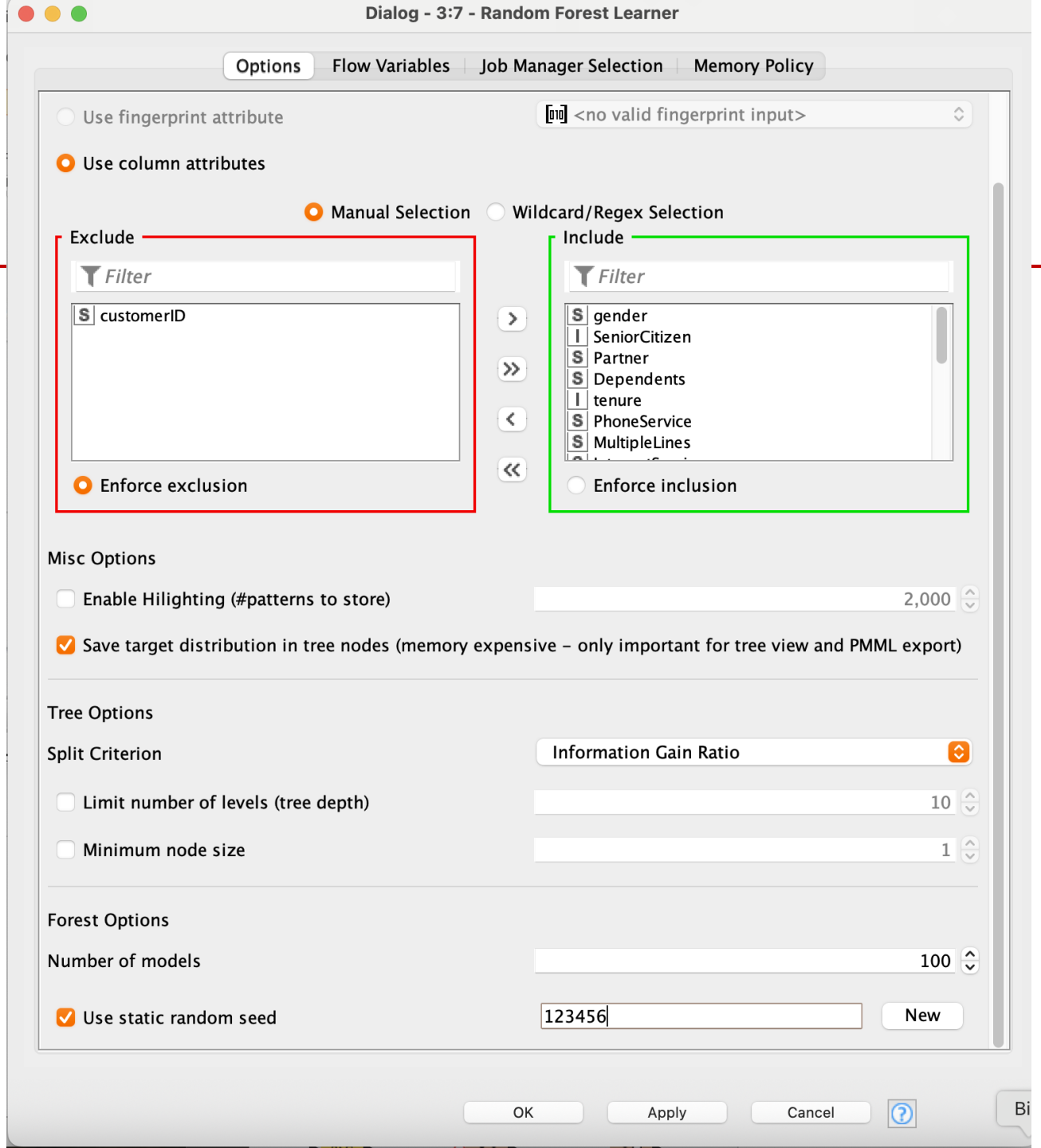
👉 Possiamo permetterci un modello meno spiegabile per:
- intercettare **più clienti a rischio**
- aumentare la **sensibilità**

Random Forest Learner in KNIME

Analytics → Mining → Decision Tree Ensemble → Random Forest → Classification

» **Parametri principali:**

- **Target Column** → Churn
- **Attribute Selection** → predittori
- **Split Criterion** → Gini Index / Information Gain Ratio
- **Number of models** → numero di alberi
- **Random seed** → per riproducibilità


Dialog - 3:7 - Random Forest Learner
Options
Flow Variables
Job Manager Selection
Memory Policy
Use fingerprint attribute
<no valid fingerprint input>
Use column attributes
Manual Selection
Wildcard/Regex Selection
Exclude
Filter
customerID
Enforce exclusion
Include
Filter
gender
SeniorCitizen
Partner
Dependents
tenure
PhoneService
MultipleLines
Enforce inclusion
Misc Options
Enable Hilighting (#patterns to store)
2,000
Save target distribution in tree nodes (memory expensive - only important for tree view and PMML export)
Tree Options
Split Criterion
Information Gain Ratio
Limit number of levels (tree depth)
10
Minimum node size
1
Forest Options
Number of models
100
Use static random seed
123456
New
OK
Apply
Cancel

Random seed nella Random Forest

» **Perché serve**
- bootstrapping casuale
- selezione casuale dei predittori

» **Effetto**
- stesso seed → stessi alberi → stessi risultati
- seed diverso → foresta diversa

👉 Utile per confrontare modelli in modo corretto.

Dalla foresta alla predizione

» **Random Forest Predictor**
  - ogni riga attraversa **tutti gli alberi**
  - ogni albero vota una classe
  - il risultato finale è scelto **a maggioranza**

» **Output possibile**
  - **Prediction (Churn)**
  - **P(Churn=Yes)** e **P(Churn=No)**

Random forest come modello di propensione

» La probabilità **P(Churn=YES)** diventa un indicatore continuo:

- valori bassi → cliente fedele
- valori intermedi → cliente indeciso
- valori alti → cliente a forte rischio

👉 Il classificatore diventa un **propensity model**.

Random Forest Predictor

Ruolo nel workflow

» Il **Random Forest Predictor** applica il modello di random forest appreso dal learner per:

- classificare ogni riga del dataset in input
- aggregare le predizioni dei singoli alberi
- restituire una **classe predetta finale**

» Ogni riga attraversa **tutti gli alberi** della foresta.

Meccanismo di votazione

» **Votazione nella random forest**
  - Ogni albero esprime un voto di classe
  - La classe finale è quella **più votata**

» **Output standard**
  - **Prediction (Churn)** → classe finale assegnata

Probabilità di classe

Append individual class probabilities

» Attivando questa opzione il nodo restituisce:

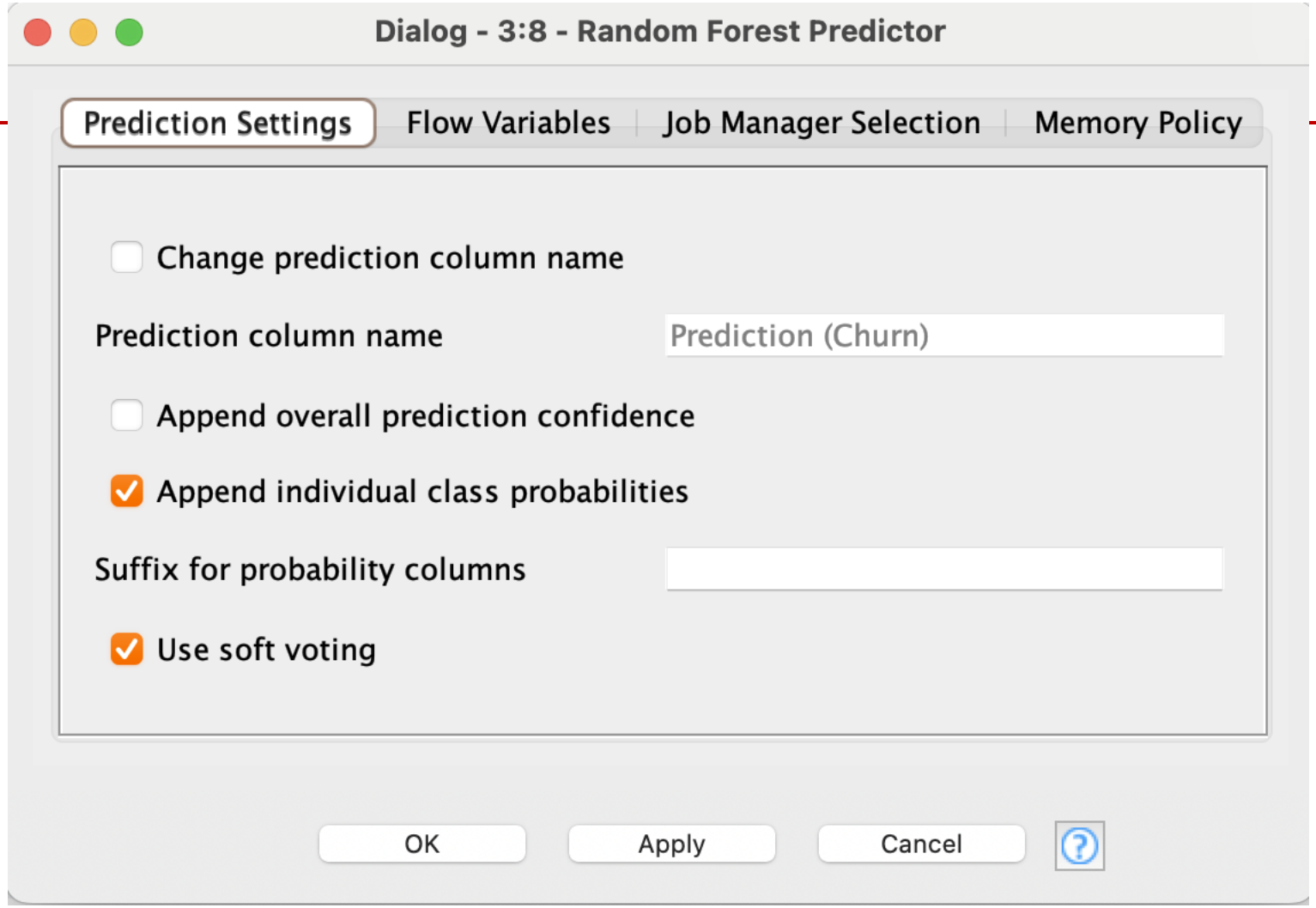
- una colonna per ogni classe
- il risultato della "votazione" degli alberi

» **Esempio**

- Su 100 alberi:
  - » 85 votano *YES*
  - » 15 votano *NO*

» Output:

- **P(YES) = 0.85**
- **P(NO) = 0.15**


Dialog - 3:8 - Random Forest Predictor
Prediction Settings
Flow Variables
Job Manager Selection
Memory Policy
Change prediction column name
Prediction column name
Prediction (Churn)
Append overall prediction confidence
Append individual class probabilities
Suffix for probability columns
Use soft voting
OK
Apply
Cancel

Soft voting

» Con **soft voting**:
- non conta solo il numero di voti
- conta anche la **sicurezza del voto** di ciascun albero

» **Come funziona**
- ogni albero pesa la votazione usando la **normalized class distribution**
- alberi "incerti" influenzano meno il risultato finale

⚠ Richiede:
- **Save target distribution in tree nodes** attivo nel learner

Dal modello di classificazione al modello di propensione

Soglia standard

» Il classificatore assegna:

- *Churn = YES* se **P(Churn=YES) > 0.5**
- *Churn = NO* se **P(Churn=YES) ≤ 0.5**

» Questa soglia è **arbitraria**.

Idea chiave: abbassare la soglia

Intuizione

» Se abbassiamo la soglia (es. da **0.5 a 0.3**):

- includiamo clienti più “indecisi”
- aumentiamo il numero di clienti intercettati
- aumentiamo la **sensibilità**

» 👉 A costo di:

- più falsi positivi
- maggiori costi di retention

Random Forest come modello di propensione

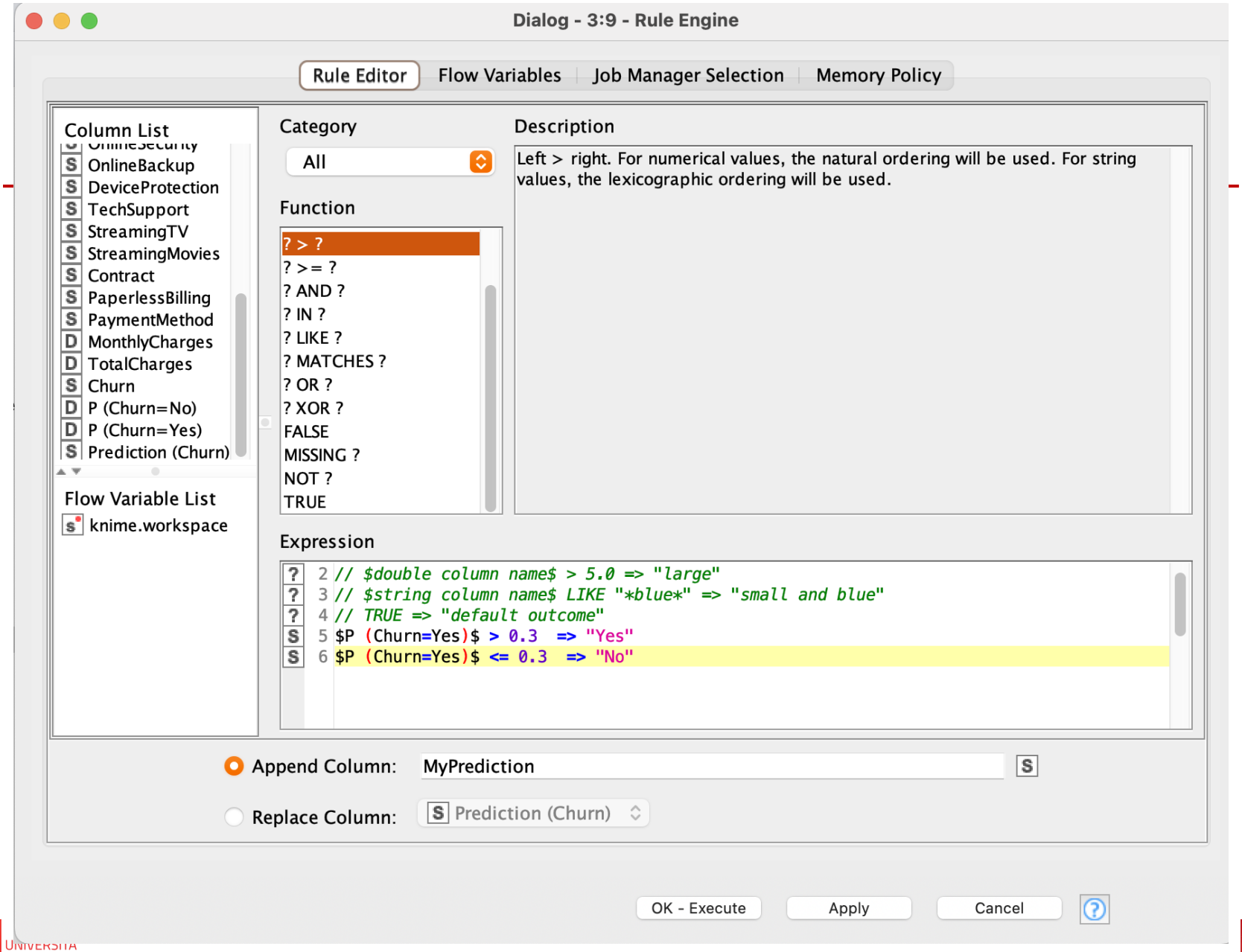
» La colonna **P(Churn=YES)** diventa un indicatore continuo:
  - vicino a **0** → cliente fedele
  - intorno a **0.5** → cliente indeciso
  - vicino a **1** → cliente molto a rischio

» Il classificatore diventa un **propensity model**.

Perché serve il Rule Engine

» **Problema**
  - La Random Forest produce probabilità, ma:
    »la classe finale è assegnata con soglia fissa (0.5)

» **Soluzione**
  - Usare **Rule Engine** per:
    »riassegnare manualmente la classe
    »Applicare una **nuova soglia personalizzata**


Dialog - 3:9 - Rule Engine
Rule Editor
Flow Variables
Job Manager Selection
Memory Policy
Column List
OnlineBackup
DeviceProtection
TechSupport
StreamingTV
StreamingMovies
Contract
PaperlessBilling
PaymentMethod
MonthlyCharges
TotalCharges
Churn
P (Churn=No)
P (Churn=Yes)
Prediction (Churn)
Flow Variable List
knime.workspace
Category
All
Function
? > ?
? >= ?
? AND ?
? IN ?
? LIKE ?
? MATCHES ?
? OR ?
? XOR ?
FALSE
MISSING ?
NOT ?
TRUE
Description
Left > right. For numerical values, the natural ordering will be used. For string values, the lexicographic ordering will be used.
Expression
// $double column name$ > 5.0 => "large"
// $string column name$ LIKE "*blue*" => "small and blue"
// TRUE => "default outcome"
$P (Churn=Yes)$ > 0.3 => "Yes"
$P (Churn=Yes)$ <= 0.3 => "No"
Append Column:
MyPrediction
Replace Column:
Prediction (Churn)
OK - Execute
Apply
Cancel

Rule Engine: cosa fa

» Il **Rule Engine**:
- crea una nuova colonna
- assegna valori in base a **regole if–then**

» **Caratteristiche**
- confronti tra colonne
- regole scritte una per riga
- sintassi:

```
condizione => valore
```

Esempio di regole

» Esempio concettuale:
  - Se **P(Churn=YES) > 0.3** → *YES*
  - Altrimenti → *NO*

» Risultato:
  - nuova colonna (es. **MyPrediction**)
  - classificazione più aggressiva verso il churn

Append vs Replace Column

Nel Rule Engine possiamo scegliere:

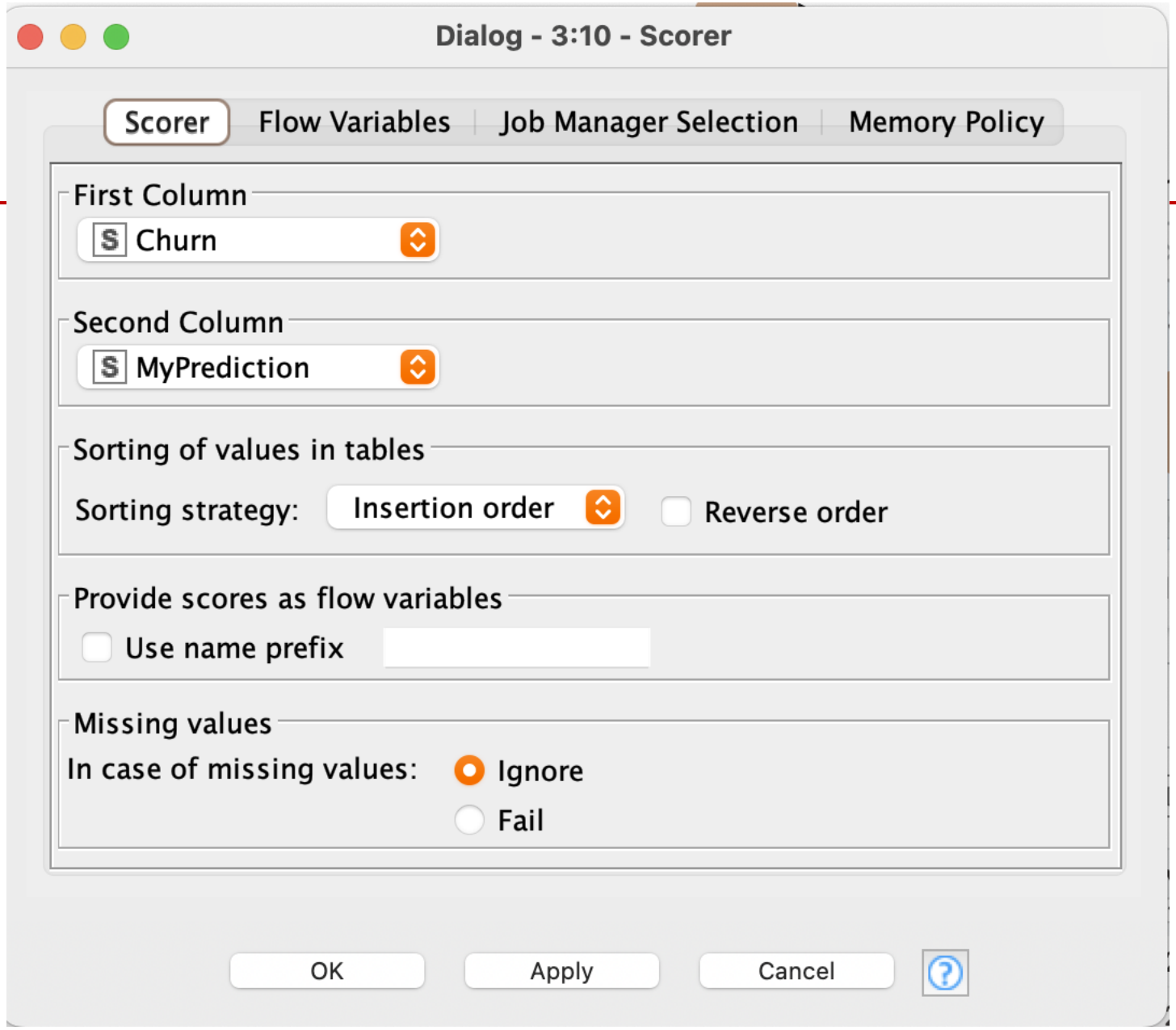
» **Append Column**

- aggiunge una nuova colonna
- mantiene la predizione originale

» **Replace Column**

- sostituisce la colonna esistente

👉 Utile per confrontare modelli diversi.


Dialog - 3:10 - Scorer
Scorer
Flow Variables
Job Manager Selection
Memory Policy
First Column
S Churn
Second Column
S MyPrediction
Sorting of values in tables
Sorting strategy:
Insertion order
Reverse order
Provide scores as flow variables
Use name prefix
Missing values
In case of missing values:
Ignore
Fail
OK
Apply
Cancel

Confusion matrix (Table)

Rows: 2 | Columns: 2

#	RowID	No Number (Integ...	Yes Number (Integer)
1	No	810	245
2	Yes	80	272

Accuracy statistics (Table)

Rows: 3 | Columns: 11

#	RowID	TruePosit... Number (...	FalsePosi... Number (...	TrueNega... Number (...	FalseNeg... Number (...	Recall Number (...	Precision Number (...	Sensitivity Number (...	Specificity Number (...	F-meas Numb
1	No	810	80	272	245	0.768	0.91	0.768	0.773	0.833
2	Yes	272	245	810	80	0.773	0.526	0.773	0.768	0.626
3	Overall	?	?	?	?	?	?	?	?	?

Effetto della riclassificazione

» Dopo la riclassificazione:
  - l'**accuracy** diminuisce
  - la **sensibilità aumenta notevolmente**
  - diminuiscono i **falsi negativi**
  - aumentano i **falsi allarmi**

» È il risultato atteso di una strategia più aggressiva.

Il trade-off fondamentale

» La sensibilità è direttamente legata ai falsi allarmi:
  - ↑ sensibilità → ↑ falsi positivi
  - ↓ sensibilità → ↑ clienti persi

» Non esiste una soluzione "migliore" in assoluto:
  - la scelta dipende dal **contesto di business**.

Analogia dell'antifurto

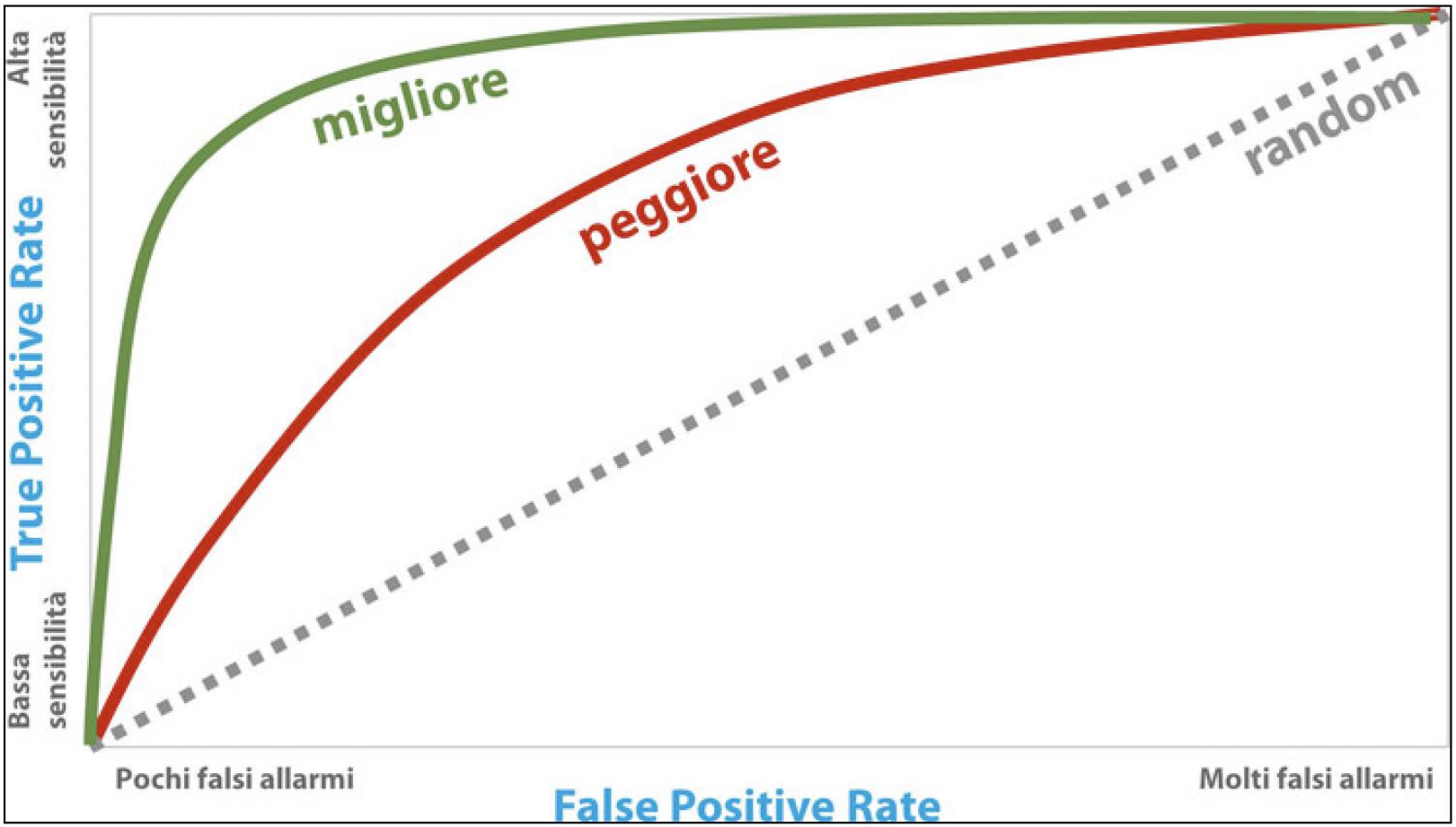
» Come un sistema di antifurto:
  - soglia alta → pochi falsi allarmi, più rischi
  - soglia bassa → molti falsi allarmi, maggiore sicurezza

» Nel churn:
  - abbassare il cut-off aumenta la sicurezza (meno clienti persi)
  - ma costa di più (più clienti contattati inutilmente)

Curve ROC

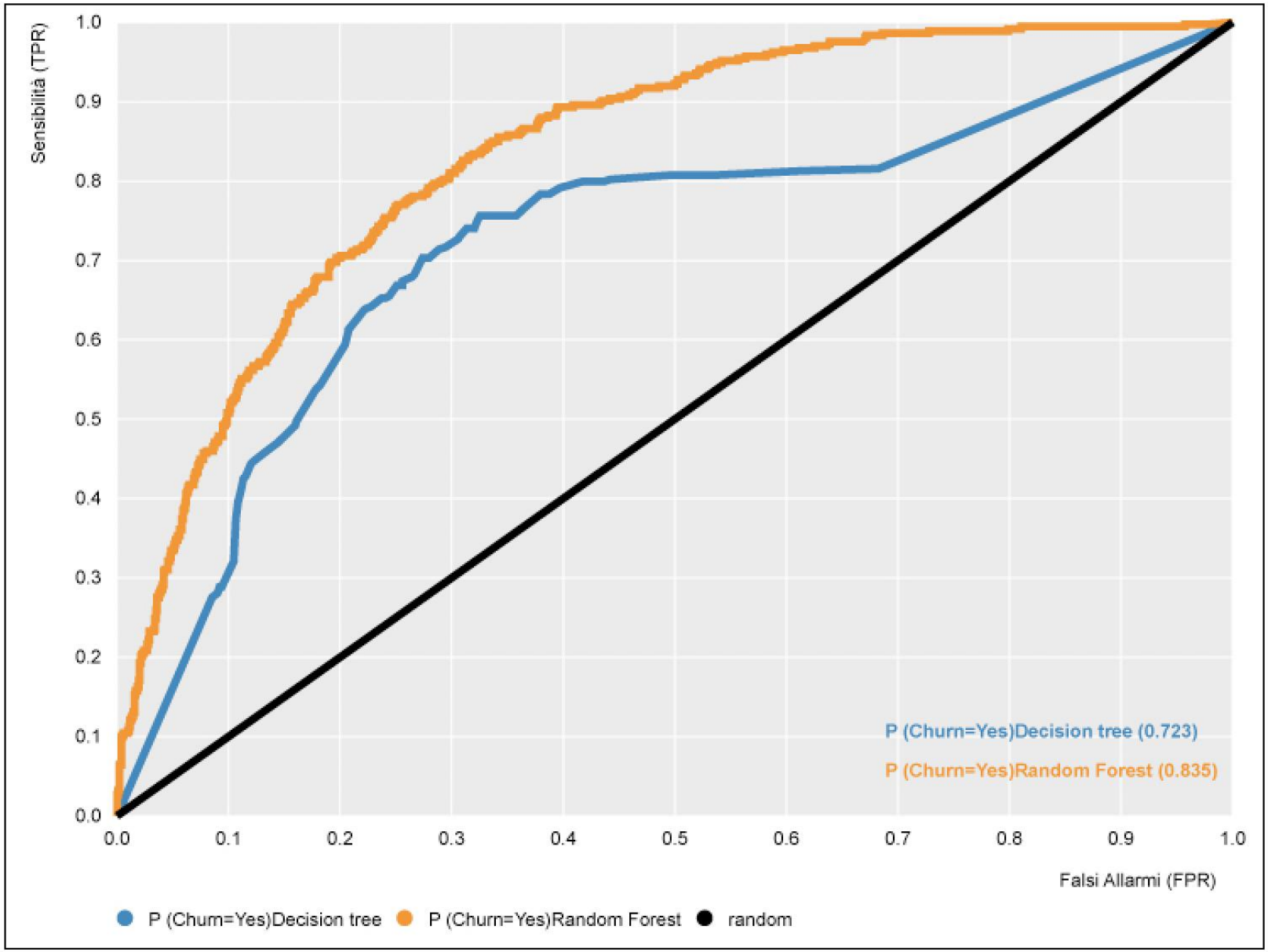
» La **curva ROC** rappresenta graficamente il trade-off:
  - asse Y → Sensitivity (TPR)
  - asse X → False Positive Rate (FPR)

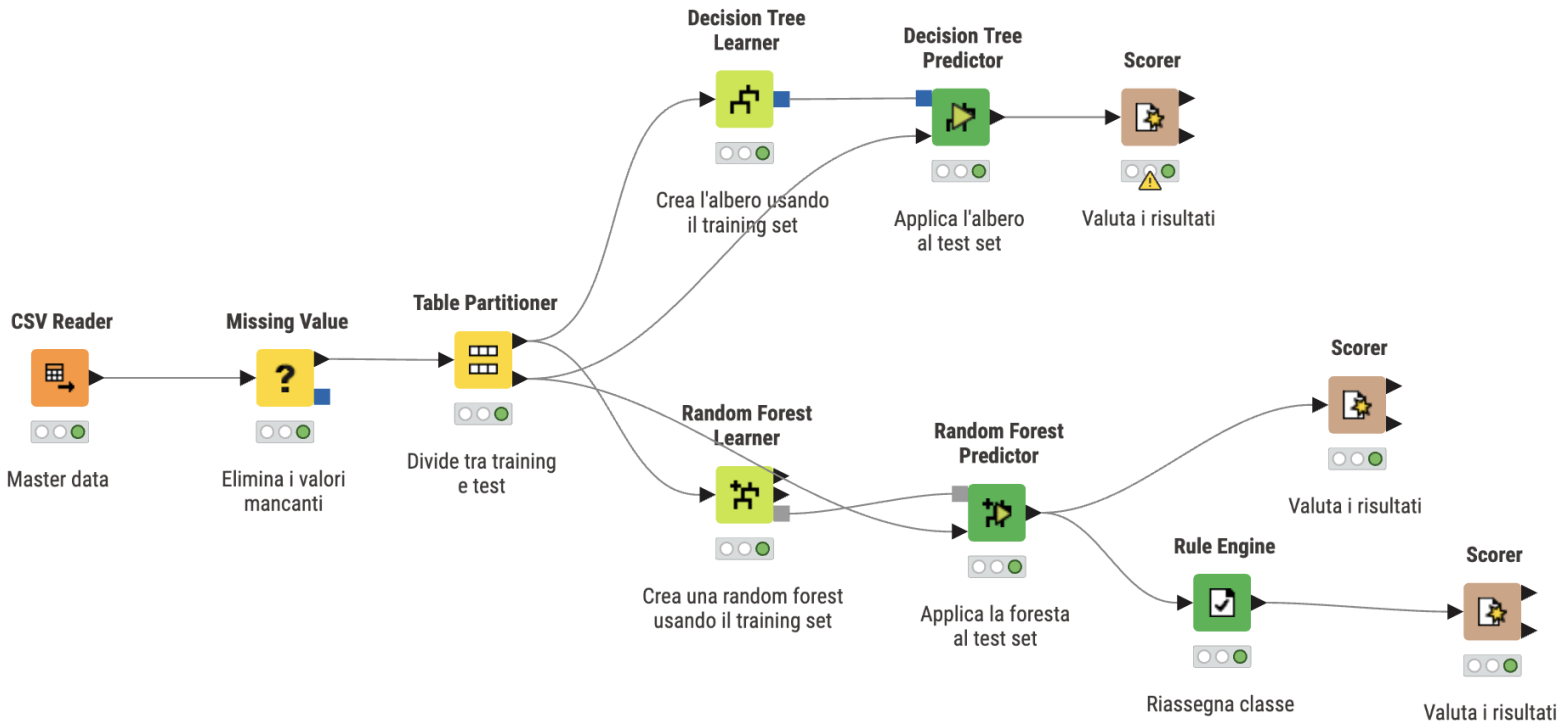
» Un buon classificatore:
  - ha una curva “alta”
  - si avvicina all’angolo in alto a sinistra


migliore
peggiore
random
True Positive Rate
Alta sensibilità
Bassa sensibilità
Pochi falsi allarmi
Molti falsi allarmi
False Positive Rate

AUC

» L'**AUC (Area Under Curve)** misura la qualità globale del modello:
  - AUC = 0.5 → classificatore casuale
  - AUC più alta → maggiore capacità discriminante
» Nel confronto:
  - Random Forest: AUC = 0.835
  - Decision Tree: AUC = 0.723
» La Random Forest è più efficace.


Sensibilità (TPR)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Falsi Allarmi (FPR)
P (Churn=Yes)Decision tree (0.723)
P (Churn=Yes)Random Forest (0.835)
P (Chum=Yes)Decision tree
P (Churn=Yes)Random Forest
random


Decision Tree Learner
Decision Tree Predictor
Scorer
Crea l'albero usando il training set
Applica l'albero al test set
Valuta i risultati
CSV Reader
Missing Value
Table Partitioner
Master data
Elimina i valori mancanti
Divide tra training e test
Random Forest Learner
Random Forest Predictor
Scorer
Valuta i risultati
Crea una random forest usando il training set
Applica la foresta al test set
Rule Engine
Scorer
Riassegna classe
Valuta i risultati