Dati, algoritmi e frontiere dell'informatica

Esercitazione n.1

L'obiettivo dell'esercitazione è il seguente:

- **Applicare algoritmi di data mining per la classificazione al fine di analizzare dati reali mediante l'utilizzo dell'applicazione RapidMiner.**

Dati strutturati

- **Il dataset denominato Users (Users.xls) raccoglie dati anagrafici e lavorativi relativi a circa 1000 persone contattate da un'azienda per proporgli l'iscrizione ad un loro servizio. Per tali utenti è noto se, dopo essere stati contattati, si sono iscritti al servizio proposto oppure no (valore del campo Response). La campagna di promozione del servizio continua e il personale della compagnia deve decidere chi, tra un elenco di circa 30000 persone non ancora contattate (Prospects.xls), potrebbe essere interessato al servizio. Idealmente, per massimizzare gli incassi e minimizzare le spese, vorremmo contattare tutte e solo le persone interessate al servizio sponsorizzato.**
- **La lista completa degli attributi dei dataset a disposizione (Users.xls e Prospescts.xls) è riportata di seguito.**
-

(1) Age
(2) Workclass
(3) Education
(4) Marital status
(5) Occupation
(6) Relationship
(7) Race
(8) Sex
(9) Native country
(10) Response

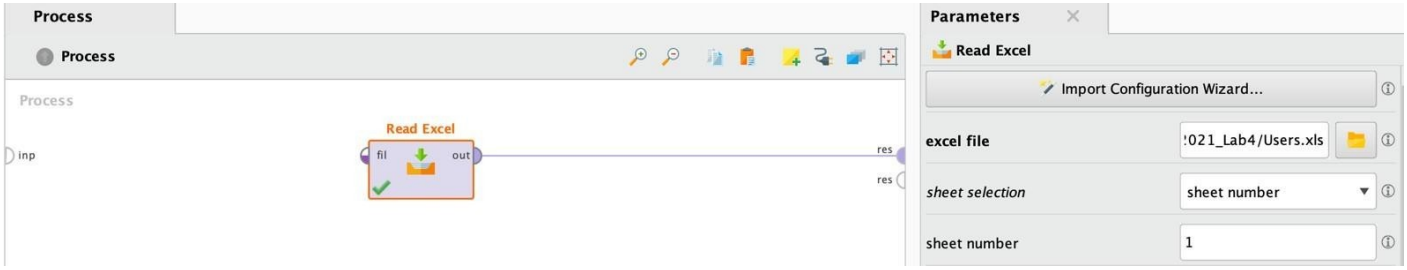
- **Obiettivo 1 – Identificazione potenziali utenti interessati ad un determinato servizio promosso in una campagna di marketing**

Gli analisti della compagnia vogliono decidere quali persone contattare e quali no per proporgli il servizio attualmente in promozione. Come possiamo risolvere il problema? In questo laboratorio vedremo come usare la classificazione per predire quali utenti è meglio contattare e quali no durante la campagna di marketing. Quali sono gli attributi predittivi? Qual è l'attributo da predire?

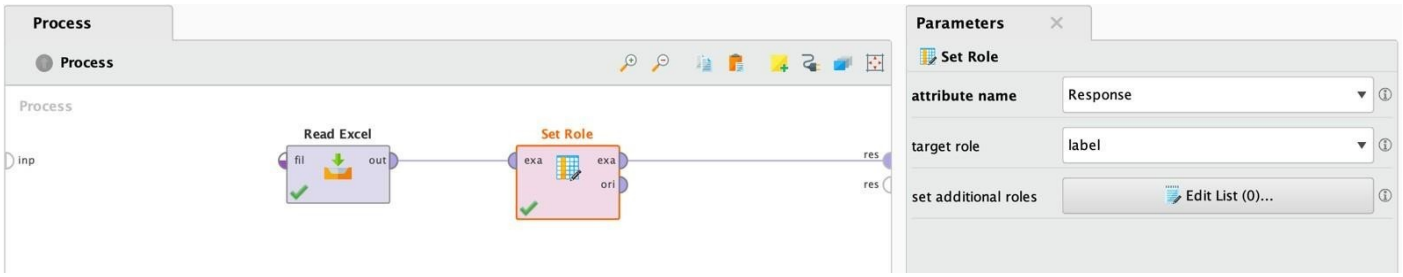
A tale scopo, gli analisti decidono di utilizzare inizialmente un albero di decisione. Si ricorda che il problema della classificazione è composto da due parti: generazione del modello (sui dati di training) e applicazione del modello (sui dati per cui l'etichetta di classe è ignota).

Passi per risolvere il problema e sua implementazione in RapidMiner:

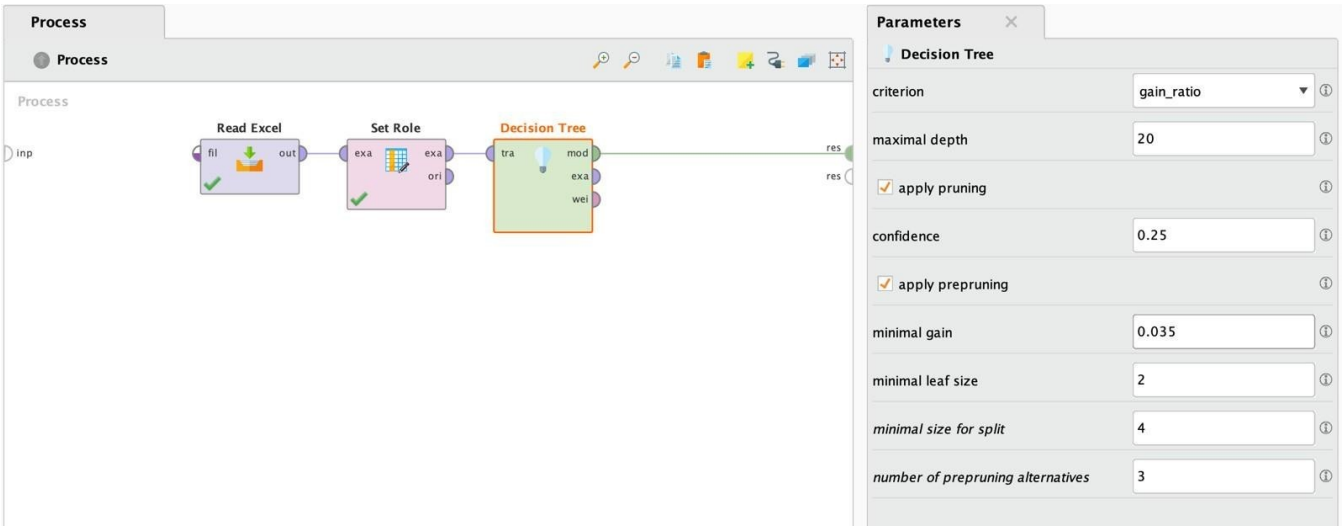
- Come primo passo è necessario creare un processo che crea il modello di classificazione
- Caricare i dati presenti in Users.xls usando l'operatore **Read Excel**. Usare anche in questo caso il Wizard per importare in modo corretto i dati.



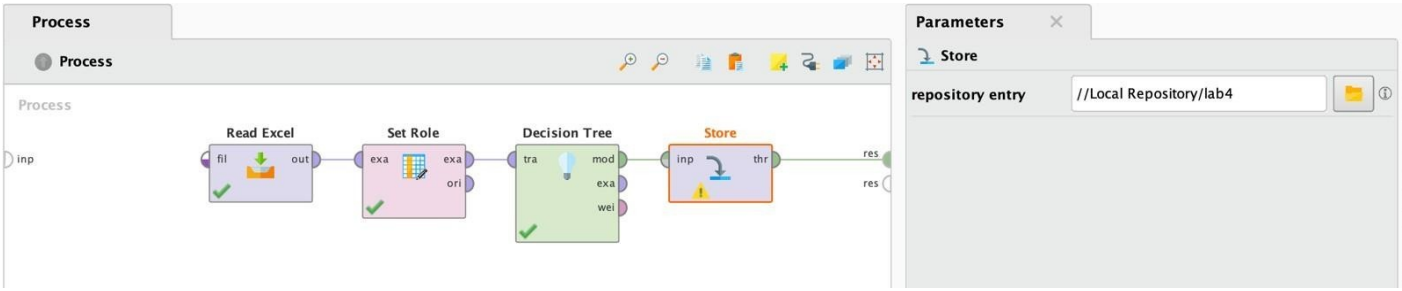
- Specificare qual è l'attributo di classe (attributo da predire) usando l'operatore **Set Role**. Assegnare il ruolo "label" all'attributo di classe.



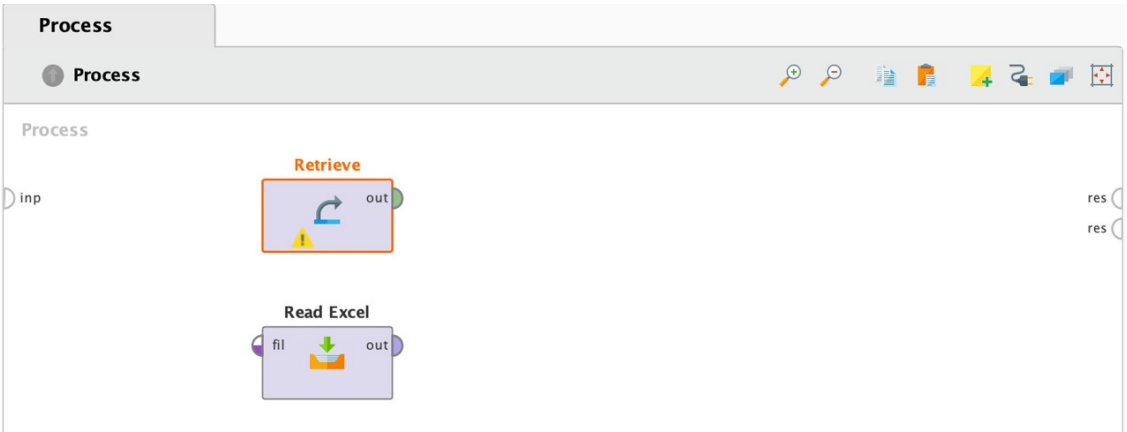
- Applicare l'algoritmo per la generazione del modello di classificazione. In questo caso usiamo l'albero di decisione che è implementato dall'operatore ***Decision Tree***. Fornire in ingresso all'operatore i dati e settare i parametri (maximal depth e minimal gain come riportato nella slide successiva).
- Collegare l'output mod che contiene il modello all'uscita (res) del processo.



- Provare ad eseguire il processo e analizzare le caratteristiche del modello (albero generato).
  - Quale attributo è considerato dall'algoritmo il più selettivo al fine di predire la classe di un nuovo dato di test?
  - Qual è l'altezza dell'albero di decisione generato?
  - Trovare un esempio di partizionamento puro all'interno dell'albero di decisione generato.
  - Analizzare l'impatto del minimal gain (considerando il gain ratio come criterio di splitting) e del maximal depth sulle caratteristiche dell'albero di decisione generato.
- Tornare all'albero di decisione e modificare il processo in modo da salvare il modello generato. Per svolgere tale operazione usare l'operatore **Store** collegando al suo ingresso l'output dell'operatore Decision Tree (selezionare come output quello che contiene il modello). Indicare tramite il parametro "repository entry" di "Store" il percorso dove si vuole salvare il modello.

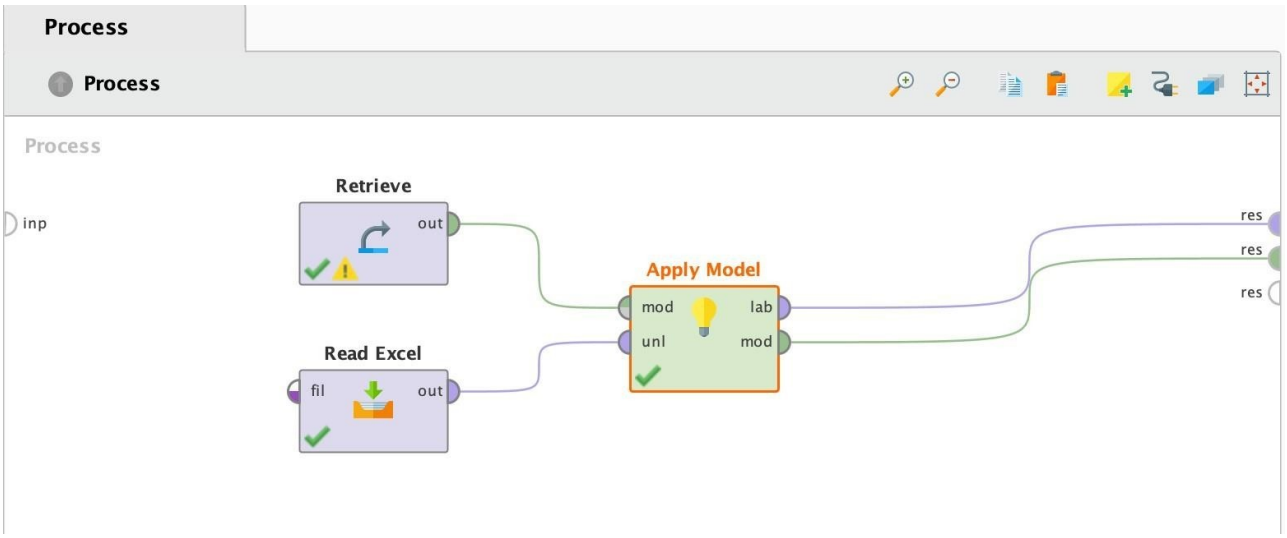


- Ora che abbiamo creato il modello di classificazione dobbiamo applicarlo ai dati presenti in Prospects.xls (utenti non ancora contattati) per decidere chi di loro contattare. Applicando il modello generato, ogni persona presente in Prospects sarà assegnata ad una delle due classi possibili: interessato (response = Positive)/non interessato (response = Negative). Caricare i dati presenti in Prospects.xls usando l'operatore **Read Excel**. Usare anche in questo caso il Wizard per importare in modo corretto i dati.
- Caricare il modello usando l'operatore **Retrieve**. Usare come file lo stesso file usato prima per salvare il modello



- Applicare il modello ai clienti Prospects.xls usando l'operatore **Apply Model** che ha due ingressi: il modello e i dati (senza etichetta) da classificare.
- Visualizzare le predizioni effettuate dal modello generato visualizzando l'output di Apply Model (l'output che si chiama lab è quello che ci interessa). Come noterete i dati sono gli stessi forniti in ingresso ma ora è presente un nuovo attributo (prediction(Response)) che contiene la predizione effettuata dal modello di classificazione per ogni utente. Vi sono inoltre altri due attributi che stimano

la probabilità di appartenenza di ogni utente alla classe Positive e Negative, rispettivamente.

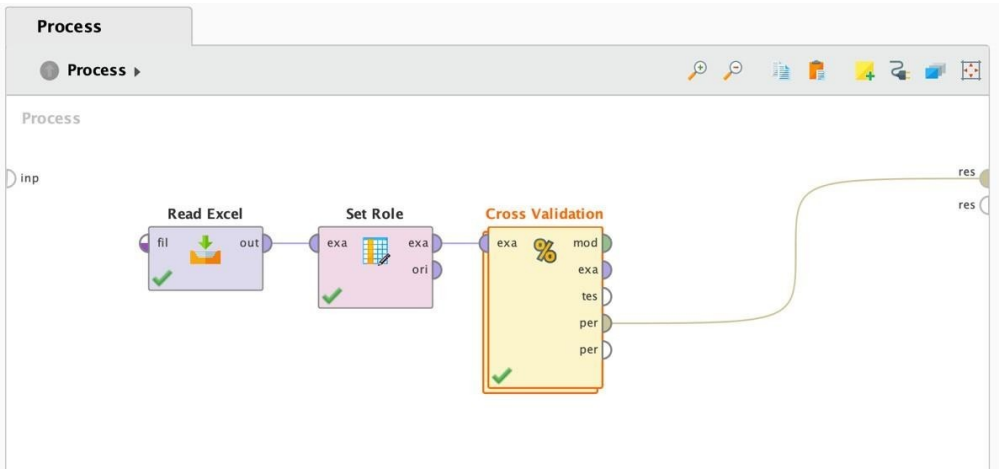


Per decidere se l'algoritmo che abbiamo usato va bene oppure no dobbiamo validare in qualche modo i risultati. Visto che per le persone presenti in Prospects la classe reale (Response) non è nota non possiamo usare quei dati per validare la qualità del modello. Per validare il modello dobbiamo usare i dati di training e un approccio tipo la cross-validation su tali dati per stimare l'accuratezza, la precisione e il richiamo del modello generato. Per fare questa operazione in RapidMiner possiamo usare un operatore apposito.

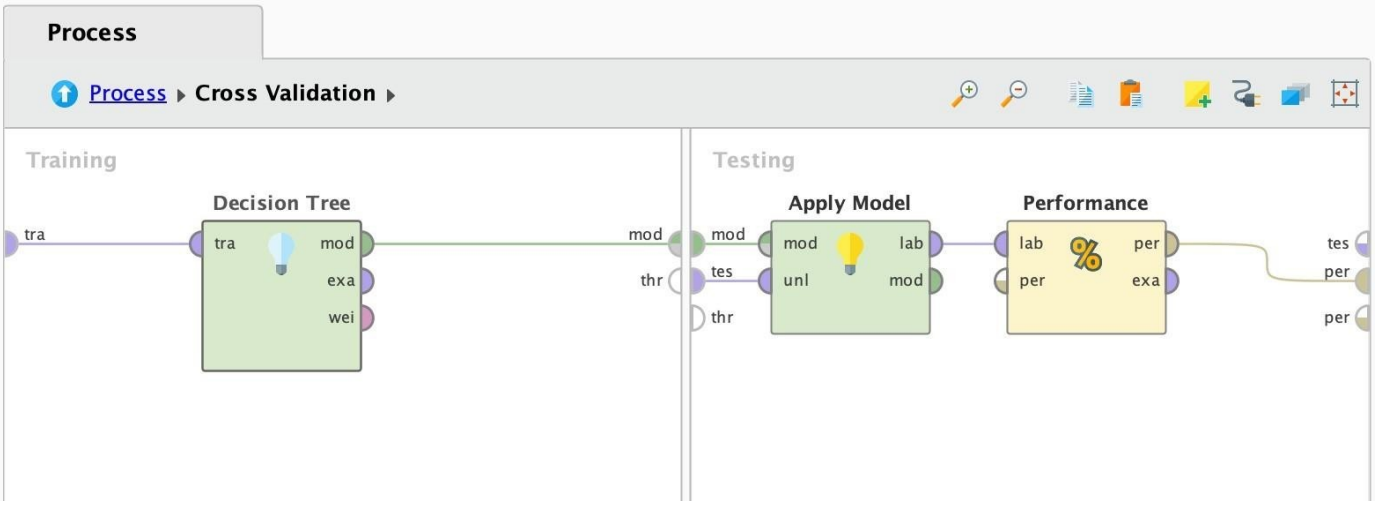
- **Obiettivo 2 – Validazione dei modelli di classificazione per mezzo della Cross Validation.**

I passi base per valutare la qualità di un classificatore tramite RapidMiner sono i seguenti:

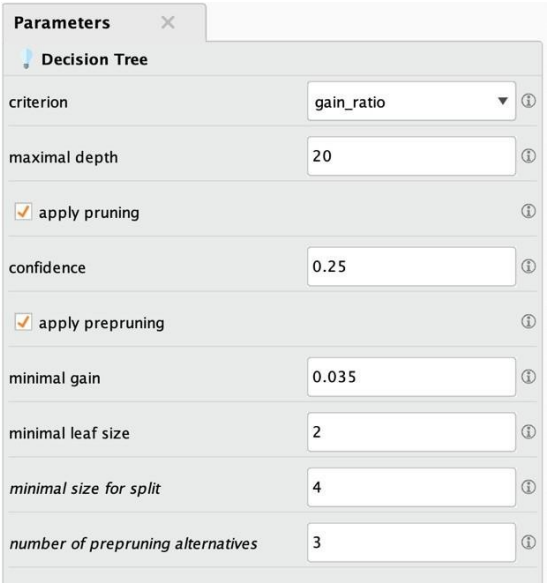
- Caricare i dati di training (Users.xls) e impostare come sempre l'etichetta di classe (con Set role).
- Inserire nel flusso del processo l'operatore **Cross Validation**. Tale operatore, tramite i suoi parametri, permette di specificare se si vuole eseguire una validazione basata su cross-validation oppure sull'approccio leave-one-out (utilizzabile solo quando si hanno pochi dati). Nel caso della scelta della cross-validation si deve specificare quanti fold si devono creare (più fold = maggiore affidabilità del risultato ma tempi più lunghi). In input si fornisce il dataset di training e come uscita si seleziona quella che si chiama "per" che restituisce indicatori relativi alle performance del modello validato (accuratezza, precisione, richiamo, ecc.).

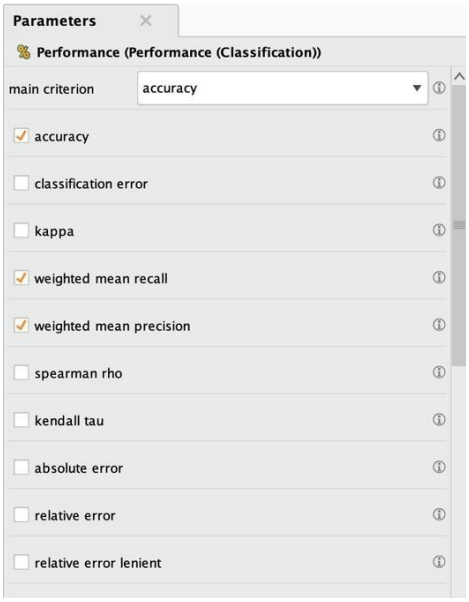


- Cross Validation è un operatore "complesso" che richiede di specificare al suo interno due sottoprocessi: uno relativo alla fase di costruzione del modello e uno relativo alla fase di applicazione e validazione. Cliccare due volte sull'operatore Cross Validation e eseguire le seguenti operazioni:



  - Inserire nella parte sinistra (area Training) l'algoritmo di classificazione che si vuole valutare (cominciare con il Decision Tree). Collegare il connettore tra dell'area di training con l'ingresso tra dell'algoritmo di classificazione e l'uscita mod con il connettore mod dell'area (sempre l'area di training).
  - Inserire nella parte destra (area Testing) prima un operatore Apply model (fornendogli in ingresso il valore di mod e tes) e poi in cascata l'operatore "Performance (Classification)". Collegare l'uscita lab di Apply model con l'ingresso lab di Performance (Classification). Per ciò che riguarda l'operatore Performance (Classification) impostare i parametri indicando quali misure analizzare. Nel nostro caso indichiamo l'accuratezza come "main criterion" perché vogliamo al momento valutare tale misura. Connettere l'uscita per di Performance (Classification) al connettore per dell'area di test.





  - Eseguire il processo e analizzare i risultati ottenuti. In particolare, analizzare con attenzione la **matrice di confusione** generata per capire come si comporta il classificatore non solo in generale ma anche sulle singole classi.

- L'accuratezza di per sè si può considerare una metrica affidabile in questo caso? Perché?
- Su quale delle due classi il classificatore performa meglio?

- **Obiettivo 3 – Random Forest**

Replicare quanto fatto fino ad ora, ma con il classificatore Random Forest al posto del Decision Tree.

- Come variano le performance (accuracy, precision e recall) rispetto al Decision Tree?

- **Obiettivo 4 – Altro algoritmo di classificazione noto**

Replicare quanto fatto fino ad ora, ma con un algoritmo di classificazione noto, diverso dai due già testati

- Come variano le performance (accuracy, precision e recall) rispetto al Decision Tree?