KNIME (autore: Vittorio Albertoni)

Premessa

A differenza di quanto avviene se utilizziamo altro software libero per analisi su big data, KNIME ci consente di fare moltissime cose, praticamente tutte quelle immaginabili, senza scrivere una riga di codice.

E' nato nel 2006 presso l'università di Costanza e il suo nome è un acronimo che ricorda, con KN la sua città natale Konstanz, con la IM le parole Information Mining e si chiude con una E che fa bello. Anche se, data l'origine, si sarebbe portati a pronunciarlo alla tedesca, tutti lo pronunciano all'inglese e dicono `naim`.

Indice

1 Installazione e primo avvio **2**

2 Come si lavora **2**

3 Un esempio classico numerico **3**

4 Un esempio diverso **6**

5 Altro esempio classico **9**

6 Estensioni **14**

7 Per saperne di più **14**

8 Se proprio serve... **14**

1 Installazione e primo avvio

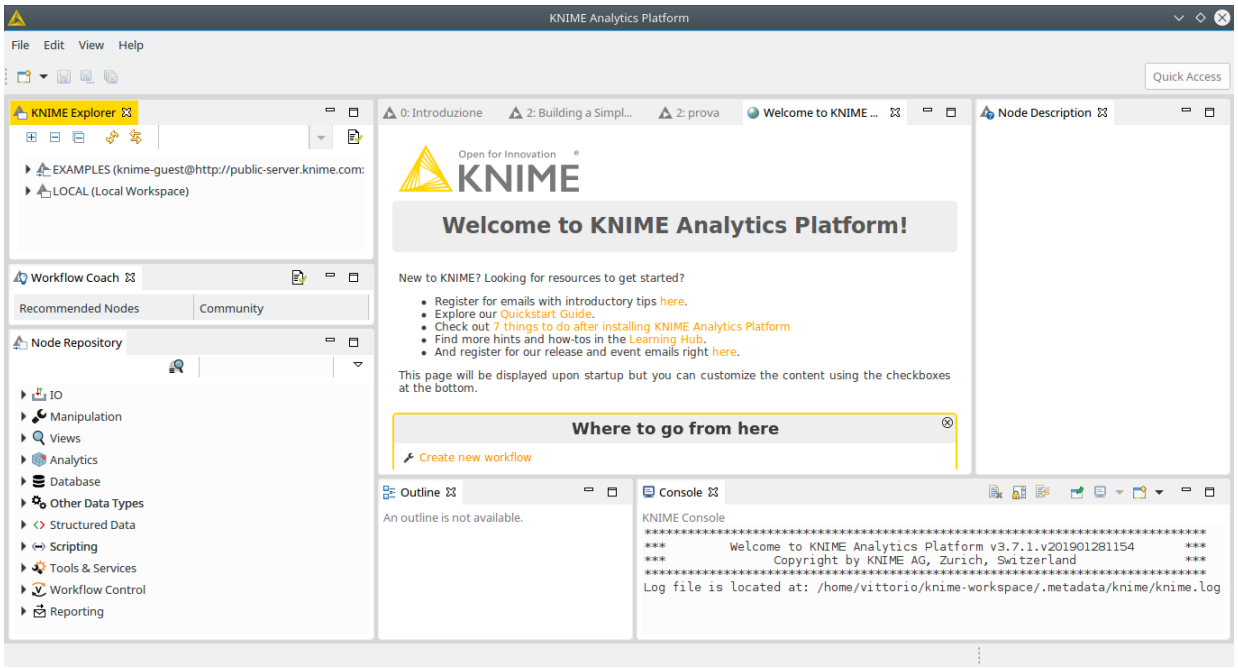
All'indirizzo *https://www.knime.com/* troviamo tutto su KNIME.

Nella sezione Download ci viene offerto di entrare a far parte della famiglia lasciando i nostri dati (Register for Help & Updates); che lo facciamo o meno, proseguendo (Download KNIME) troviamo quanto serve per installare KNIME sul sistema operativo che più ci fa comodo (Linux, Mac OS X o Windows); proseguendo ancora (Get Started) troviamo le istruzioni per l'installazione.

Nella directory ove è avvenuta l'installazione troviamo l'eseguibile (`knime.exe` per Windows, `knime` per Linux e Mac OS X). Nel caso di Windows e Linux, se riteniamo, dobbiamo costruirci noi una voce di lancio nel menu, nel caso di Mac OS X potremo semplicemente spostare in Applications l'icona che si genera al momento dell'installazione ed avvalerci di questa per il lancio.

Al primo lancio ci viene chiesto di confermare la directory di lavoro (knime-workspace) o di settarne un'altra.

KNIME, nella versione 3.7.1 del 29 gennaio 2019, si presenta così



La finestra centrale, dove al primo lancio compare il benvenuto, è l'editor della sequenza di lavoro, il workflow editor.

La finestrella in alto a sinistra è l'Explorer e vi compaiono i file relativi ai nostri lavori con la possibilità di organizzarli in cartelle.

La finestrella in basso a destra è una Console nella quale veniamo aggiornati su ciò che stiamo facendo, con segnalazione di warning o errori.

La finestra in basso a sinistra è il Repository, il magazzino, dei così detti nodi (nodes), che sono le funzionalità che ci offre KNIME per fare le nostre analisi.

La finestra in alto a destra contiene la descrizione della funzionalità selezionata nel Repository in basso a sinistra.

2 Come si lavora

Se svolgiamo un'analisi di dati con gli strumenti che ci mettono a disposizione i linguaggi R o Python dobbiamo praticamente creare noi, scrivendone il codice, delle funzioni per le varie fasi dell'analisi e, anche se le funzioni le troviamo in una libreria, l'analisi si compie eseguendo un programma, di cui sempre noi scriviamo il codice, che si avvale di queste funzioni.

Trasportandoci in KNIME, le funzioni per le varie fasi dell'analisi sono i nodi (**nodes**) e il programma completo che richiama queste funzioni collegandole tra loro si chiama **workflow**.

Per svolgere l'analisi dobbiamo semplicemente trascinare nel workflow editor i nodi che ci servono, configurarli per ciò che devono fare utilizzando finestre di dialogo intuitive e collegarli nell'ordine voluto per lo sviluppo dell'analisi: il tutto senza scrivere una riga di codice. Il codice l'ha scritto per noi lo sviluppatore del nodo in linguaggio Java.

Attenzione, comunque, che quello di evitarci la scrittura del codice è l'unico alleggerimento che ci dà KNIME, perché tutto il resto dobbiamo saperlo fare noi. Certo è che poter fare un'analisi senza necessariamente conoscere i segreti della sintassi di scipy o di pandas non è una semplificazione da poco.

Per cominciare a lavorare su un'analisi normale si sceglie da menu FILE ▷ NEW, NEXT nella finestra successiva, si inserisce il nome del progetto nella finestra ancora successiva e si conclude con FINISH.

Il workflow editor assumerà l'aspetto di un foglio bianco di carta quadrettata pronto a ricevere il nostro lavoro.

3 Un esempio classico numerico

Dal lontano 1877, ad opera di Francis Galton, si parla di regressione quando si ha a che fare con l'individuazione della dipendenza di un dato da un altro: originariamente Galton riferì il concetto allo studio della relazione tra l'altezza dei figli e l'altezza dei padri.

Galton non era un matematico e gli strumenti che ha usato lui non erano quelli che conosciamo oggi ma il concetto è rimasto e i matematici hanno escogitato strumenti di analisi che si spingono addirittura all'analisi della dipendenza di una variabile da più variabili (la così detta regressione multipla) con la misurazione statistica dell'attendibilità della dipendenza individuata in modo da poter consapevolmente valutare o predire quanto possa succedere in presenza di mutazioni di queste variabili.

Oggi la regressione è il punto forte del così detto apprendimento automatico supervisionato nell'ambito del machine learning: la macchina esamina dei dati reali, apprende le relazioni che legano questi dati e prospetta situazioni derivanti dalla variazione dei dati stessi.

Ai tempi in cui lavoravo con queste cose, già negli anni '70 del secolo scorso, utilizzando scomodi mainframe al posto dei moderni computer, eravamo tutti convinti che si trattasse di strumenti per una modellistica descrittiva, non osando nemmeno chiamarla modellistica esplicativa: oggi se ne fa addirittura strumento di intelligenza artificiale. Spero che non si esageri e che qualche computer non arrivi a convincerci che il diffondersi del diabete dipende dall'andamento del raccolto delle pere.

Ma veniamo all'esempio che vorrei proporre.

Partiamo da un piccolo dataset, di dimensioni sufficienti per capirci, contenuto in un file .csv:

y	x1	x2	x3
34	12	11	21
72	26	20	45
87	44	12	58
28	32	14	37
16	21	11	24
97	76	7	75

nel quale abbiamo una variabile y accostata al manifestarsi di sei diverse terne di variabili x.

Ci proponiamo di verificare quale sia il più probabile valore della variabile y in presenza della terna inedita in cui x1 valga 11, x2 valga 22 e x3 valga 33.

Con un data set così piccolo possiamo benissimo farcela con un foglio di calcolo.

La soluzione che propongo è ottenuta con Calc di LibreOffice ma è in tutto simile a quanto si otterrebbe con Excel.

	A	B	C	D	E	F	G	H
1	y	x1	x2	x3	coefficiente x3	coefficiente x2	coefficiente x1	intercetta
2	34	12	11	21	4,1221660955561	-2,6165470365235	-2,6483505679491	2,8799687886557
3	72	26	20	45	0,9333210188685	1,6669864665457	0,9483239593533	20,055216082116
4	87	44	12	58	0,9726259532285	8,883144272154	#N/D	#N/D
5	28	32	14	37	23,687301609642	2	#N/D	#N/D
6	16	21	11	24	5607,5128290135	157,82050431981	#N/D	#N/D
7	97	76	7	75				

Importato nelle colonne A, B, C e D il nostro file .csv, inseriamo nella cella E2 la seguente formula di matrice
{=REGR.LIN(A2:A7; B2:D7; 1; 1)}
ed otteniamo i seguenti risultati che più interessano:
. nelle celle da E2 a H2, in ordine inverso, l'intercetta (il punto dell'asse ortogonale verticale da cui parte la linea interpolante) e i coefficienti delle variabili x: per un riferimento ho indicato queste definizioni nelle celle superiori da E1 a H1;
. nella cella E4 il coefficiente di determinazione, altrimenti chiamato R quadro: se questo indicatore è prossimo all'unità, come accade nel nostro caso, significa che il modello ottenuto interpreta con attendibilità molto elevata le relazioni esistenti tra le variabili[1].
A questo punto siamo abbastanza sicuri che, in presenza dei valori di x1, x2, x3, rispettivamente 11, 22 e 33, il valore di y sia:
2,879 - 2,648 x 11 - 2,616 x 22 + 4,122 x 33 = 52,225

Agli stessi risultati perveniamo con il seguente script Python

```
import pandas as pd
dati = pd.read_csv('/home/vittorio/Documenti/dati.csv')
X = dati[['x1','x2','x3']]
y = dati['y']
from sklearn.linear_model import LinearRegression
regressore = LinearRegression()
regressore.fit(X, y)
print('intercetta', regressore.intercept_)
coefficienti = pd.DataFrame(regressore.coef_, X.columns, columns = ['coefficienti'])
print(coefficienti)
y_calcolati = regressore.predict(X)
from sklearn import metrics
print('R2', metrics.r2_score(y, y_calcolati))
```

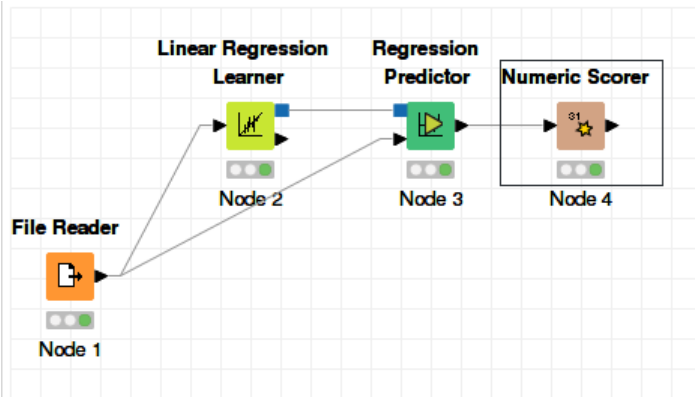
eseguendo il quale otteniamo

```
intercetta 2.87996878866
    coefficienti
x1     -2.648351
x2     -2.616547
x3      4.122166
R2 0.972625953229
```

con perfetta coincidenza con quanto ottenuto con il foglio di calcolo.

[1] A chi abbia necessità di approfondire questi passaggi suggerisco il manualetto «Calc di Libre Office» allegato all'articolo «Matematica e Statistica con Calc» del giugno 2016, archiviato nel mio blog all'indirizzo *www.vittal.it* sotto la voce «Software libero». Lo stesso contenuto si trova nella mia pubblicazione reperibile nelle librerie on line, sia in formato cartaceo sia in formato pdf, «Conti, matematica e statistica con il foglio di calcolo».

Sempre agli stessi risultati arriviamo utilizzando i quattro nodi KNIME che vediamo in questo workflow:



Anziché scrivere il codice Python come abbiamo fatto prima, procediamo così:
. apriamo KNIME, da menu FILE ▷ NEW creiamo il nostro progetto e il nostro workflow editor comparirà come un foglio quadrettato bianco;
. andiamo nel repository dei nodi (finestra in basso a sinistra), apriamo la voce IO e, al suo interno, la voce READ: nell'elenco selezioniamo il nodo FILE READER e lo trasciniamo con il mouse nel workflow editor;
. l'icona del nodo appare con il semaforo rosso e un punto esclamativo che ci richiama al dovere di configurare il nodo: ciò che facciamo con click destro sull'icona del nodo, scelta della voce CONFIGURE e inserimento del path al file di testo .csv che dobbiamo esaminare nella finestrella ENTRY ASCII DATA FILE LOCATION;
. il semaforo diventa giallo: per farlo diventare verde e poter proseguire scegliamo EXECUTE dal solito menu che si apre con click destro o, a nodo selezionato, clicchiamo sul pulsante della barra degli strumenti;
. ora nel repository dei nodi apriamo la voce ANALYTICS, al suo interno apriamo la voce MINING e, all'interno di questa apriamo la voce LINEAR/POLYNOMIAL REGRESSION dove finalmente troviamo il nodo che ci serve ora, che si chiama LINEAR REGRESSION LEARNER, e lo trasciniamo nel worflow editor sulla destra del nodo precedente, appena un po' più in alto per far vedere meglio i successivi collegamenti;
. colleghiamo questo nodo al precedente trascinando il mouse tra la freccina di uscita del primo e la freccina di entrata del secondo;
. solito semaforo rosso e invito alla configurazione che facciamo con click destro sull'icona del nodo, scelta della voce CONFIGURE e, nella finestra di dialogo, fare in modo che nella finestrella TARGET sia selezionata la Y e, nella finestrella INCLUDE siano presenti le variabili X1, X2 e X3;
. il semaforo diventa giallo e lo facciamo diventare verde con EXECUTE, come abbiamo fatto prima con l'altro nodo;
. a questo punto la macchina ha imparato e, aperto con click destro il menu del nodo LINEAR REGRESSION LEARNER, se scegliamo, sul fondo, la voce COEFFICIENTS & STATISTICS, visualizziamo i seguenti risultati

Table "Coefficients and Statistics"

Row ID	S Variable	D Coeff.	
Row1	x1	-2.648	0
Row2	x2	-2.617	1
Row3	x3	4.122	0
Row4	Intercept	2.88	2

che sono i soliti ottenuti prima con gli altri metodi;
. ma dobbiamo verificare che abbia imparato bene e, per farlo, applichiamo quanto ha imparato la macchina ai dati ottenendo una serie di variabili y calcolate da raffrontare con quelle effettive:

scegliamo pertanto dal repository dei nodi ANALYTICS ▷ MINING ▷ LINEAR/POLYNOMIAL REGRESSION il nodo REGRESSION PREDICTOR e lo portiamo nel workflow editor, sulla destra del nodo precedente;
. in questo nuovo nodo devono confluire gli elementi imparati e i dati cui applicarli: la prima cosa si fa collegando l'uscita del modello regressivo determinato in fase di learning, rappresentato da un quadratino blu sulla destra del nodo LEARNER, con l'entrata del modello nel PREDICTOR, rappresentato da un quadratino blu sulla sinistra del nodo PREDICTOR e la seconda cosa si fa collegando i dati letti dal nodo READER con la relativa entrata nel PREDICTOR;
. configuriamo il PREDICTOR con la solita procedura, indicando un nome per la colonna dei dati calcolati, per esempio `y_calcolati` e diamo EXECUTE;
. per giudicare che l'accostamento tra dati calcolati e dati effettivi sia tale da dimostrare che la macchina ha imparato bene abbiamo bisogno di attivare un altro nodo che troviamo in ANALYTICS ▷ MINING ▷ SCORING e si chiama NUMERIC SCORER: lo individuiamo, lo trasciniamo nel workflow editor, sempre sulla destra del nodo precedente, al quale lo colleghiamo, e lo configuriamo in modo che, nella finestra di configurazione OPTIONS, la REFERENCE COLUMN sia `y` e la PREDICTED COLUMN sia `y_calcolati`;
. diamo EXECUTE e, aperto con click destro il menu del nodo, se scegliamo, sul fondo, la voce STATISTICS, visualizziamo i seguenti risultati

Table "Scores" - Rows: 5

Row ID	D ycalco...
R^2	0.973
mean abs...	4.825
mean squ...	26.303
root mean...	5.129
mean sign...	-0

tra i quali ritroviamo lo stesso indicatore R quadro molto elevato, a testimonianza dell'attendibilità delle nostre valutazioni.

Per tre strade diverse siamo arrivati a fare la stessa cosa: del resto la matematica che sta dietro è sempre la stessa. Quale sia la strada più comoda lo deve giudicare l'analista. Non escludendo che lo stesso analista preferisca lavorare con Python per certe analisi, con KNIME per altre e con un foglio di calcolo per altre ancora.

E' comunque doveroso, da parte mia, un chiarimento sull'accostamento dell'uso del foglio di calcolo in alternativa agli altri due strumenti in questo esempio semplificato.

Si sarà notato che con il foglio di calcolo si risolve tutta l'analisi con l'inserimento di una formula di matrice mentre con gli altri due strumenti si procede per gradi: si leggono i dati, si analizzano per trovare relazioni tra loro, si applicano queste relazioni ai dati reali e si confrontano i risultati ottenuti con quelli reali. Quest'ultimo modo di procedere introduce gradualità e opportunità di differenziazione nell'esecuzione dei compiti: per esempio ci può consentire di applicare le relazioni imparate non a tutti i dati reali, come ho fatto io negli esempi e come fa il foglio di calcolo, ma solo a un campione di questi tratto dal dataset originale ma che non abbia partecipato all'apprendimento. Con il foglio di calcolo fare questo richiederebbe molto più impegno che non usando gli altri strumenti, che dispongono di tools proprio per fare questo; e fare questo significa essere più tranquilli circa l'attendibilità del nostro lavoro.

Infine non dimentichiamo che gira una scherzosa definizione dei big data: «i big data sono quando non basta più un foglio Excel». Il che vuol dire di tener presente la quantità di dati da elaborare, perché non è detto che un foglio di calcolo sia adeguato per trattarla.

4 Un esempio diverso

Non sempre il data scientist è chiamato a trattare dati numerici: spesso, soprattutto nelle analisi dei big data, si ha a che fare con dati espressi in linguaggio umano.

Ci può essere per esempio l'esigenza di delegare a un computer la lettura di una serie di recensioni espresse dai lettori su un'opera letteraria al fine di contare i giudizi positivi e quelli negativi. Ci può essere l'esigenza di classificare delle frasi per distinguere quelle che esprimono allegria e quelle che sono tristi, quelle che sono attribuibili a persone di sinistra e quelle che sono attribuibili a persone di destra. Cose che vanno sotto il nome di sentiment analysis.

Non è questa la sede per trattare l'argomento nella sua pienezza. Tratterò solo il primo passo, quello che consiste nel leggere frasi (per semplicità ne leggeremo solo una) e nell'estrarre dalle frasi solo parole significative per l'espressione del sentimento.

La nostra dotazione di software deve contenere gli strumenti necessari:
. per quanto riguarda la dotazione Python si tratta del modulo `nltk`;
. per quanto riguarda KNIME si tratta dei nodi del Textprocessing (se non li abbiamo li dobbiamo caricare secondo quanto vedremo nel prossimo Capitolo 6).

In questo esempio mostrerò i passaggi attraverso cui depurare la frase «se passo per questa strada mi sento allegro», contenuta nel file di testo testo.txt, prima utilizzando Python e poi utilizzando KNIME: in questo caso il foglio di calcolo non serve.

Con Python raggiungiamo il nostro obiettivo con il seguente script

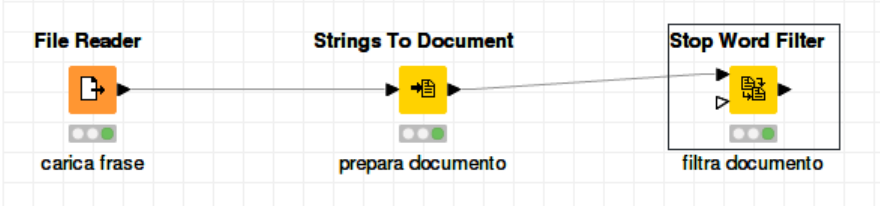
```
import io
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
stop_words = set(stopwords.words('italian'))
f = open("/home/vittorio/Documenti/testo.txt")
line = f.read()
words = line.split()
for r in words:
    if not r in stop_words:
        appendFile = open('/home/vittorio/Documenti/testo_filtrato.txt','a')
        appendFile.write(" "+r)
        appendFile.close()
```

grazie al quale, partendo dal file testo.txt contenente la frase «se passo per questa strada mi sento allegro» arriviamo al file testo_filtrato.txt contenente le parole «passo strada sento allegro» più che sufficienti per indagare il sentiment della frase da cui siamo partiti.
Da otto parole siamo passati a quattro, cioè la metà: una bella semplificazione, se pensiamo che ogni parola delle migliaia di frasi da esaminare, per capire il sentimento delle frasi stesse, va poi confrontata con gli archivi di parole tipiche dei vari sentimenti che il computer dovrà avere a disposizione per esprimere la sua «intelligenza»[2].
La scelta delle parole è avvenuta utilizzando l'archivio delle stopwords italiane di nltk per procurarci il quale, contenente stopwords anche in parecchie altre lingue, dobbiamo lanciare in una shell Python, essendo collegati a Internet, i seguenti comandi

```
import nltk
nltk.download('stopwords')
```

che ci faranno trovare quanto serve nella directory `nltk_data` della nostra area personale.

Allo stesso risultato arriviamo utilizzando i tre nodi di KNIME che vediamo nel seguente workflow:



[2]Nel nostro esempio l'intelligenza artificiale del computer si manifesta nella sua capacità di leggere una frase e di classificarla, come se il computer leggesse la frase e la capisse. In realtà è la nostra intelligenza umana che fornisce al computer tutti gli elementi per fare questa cosa.

Anziché scrivere il codice Python come abbiamo fatto prima, procediamo così:

. apriamo KNIME, da menu FILE ▷ NEW creiamo il nostro progetto e il nostro workflow editor comparirà come un foglio quadrettato bianco;

. andiamo nel repository dei nodi (finestra in basso a sinistra), apriamo la voce IO e, al suo interno, la voce READ: nell'elenco selezioniamo il nodo FILE READER e lo trasciniamo con il mouse nel workflow editor;

. l'icona del nodo appare con il semaforo rosso e un punto esclamativo che ci richiama al dovere di configurare il nodo: ciò che facciamo con click destro sull'icona del nodo, scelta della voce CONFIGURE e inserimento del path al file di testo che dobbiamo esaminare nella finestrella ENTRY ASCII DATA FILE LOCATION;

. il semaforo diventa giallo: per farlo diventare verde e poter proseguire scegliamo EXECUTE dal solito menu che si apre con click destro o, a nodo selezionato, clicchiamo sul pulsante della barra degli strumenti;

. ora nel repository dei nodi apriamo la voce OTHER DATA TYPES, al suo interno apriamo la voce TEXT PROCESSING e, all'interno di questa apriamo la voce TRANSFORMATION dove finalmente troviamo il nodo che ci serve ora, che si chiama STRINGS TO DOCUMENT, e lo trasciniamo nel worflow editor sulla destra del nodo precedente[3];

. colleghiamo il nodo precedente a questo congiungendo, per trascinamento del mouse, la freccina di uscita del primo nodo alla freccina di entrata del secondo e configuriamo il secondo nodo aprendo la finestra di configurazione con click destro e richiudendola accettando la configurazione di default;

. eseguiamo anche il secondo nodo e il semaforo diventa verde;

. l'ultimo nodo, quello che depura il nostro documento, lo troviamo nel repository dei nodi nel percorso OTHER DATA TYPES ▷ TEXT PROCESSING ▷ PREPROCESSING e si chiama STOP WORD FILTER: lo trasciniamo nel workflow editor alla destra del secondo nodo, lo colleghiamo al secondo nodo come abbiamo fatto prima tra il secondo e il primo, lo configuriamo scegliendo nell'apposita finestrella STOPWORD LISTS: ITALIAN e lo eseguiamo.

Se clicchiamo destro sul terzo nodo e nel menu che si apre scegliamo l'ultima riga PREPROCESSED DOCUMENTS vediamo la successione dei risultati delle nostre tre elaborazioni, l'ultima delle quali riporta la nostra frase di partenza ridotta alle quattro parole significative.

Lascio al lettore la valutazione, peraltro molto soggettiva, su quale sia la strada preferibile. Una cosa è certa: come nell'esempio del precedente capitolo, da una parte ci può essere la difficoltà della sintassi del linguaggio di programmazione e della conoscenza delle funzioni contenute nei vari moduli ma dall'altra c'è quella della conoscenza dei nodi, del modo di configurarli e quant'altro.

Generalmente la strada della scrittura dei codice dovrebbe essere la più elastica e quella che maggiormente lascia spazio all'inventiva personale.

Nel nostro piccolo esempio possiamo trovarne una prova.

Abbiamo visto che, lavorando con Python, abbiamo a disposizione una directory `nltk_data` in cui abbiamo i file delle stopwords. Se il file è tale che non depura a sufficienza le nostre frasi possiamo intervenire noi e modificarlo (è un semplice file di testo).

Lavorando con KNIME il file ci viene somministrato e basta e nemmeno sappiamo dov'è.

Una costa pare certa: le analisi con KNIME di una certa complicazione e su grandi quantità di dati sono molto più veloci sia sul piano della progettazione da parte dell'analista sia sul piano dell'esecuzione da parte della CPU del computer. Una certa cosa si dice abbia richiesto 2 giorni di lavoro fatta con Python e 3 ore di lavoro fatta con KNIME.

[3]Può accadere che, alla prima installazione del software, non venga installato il Text Processing; in tal caso non troveremo ciò che cerchiamo nel repository e dovremo installare l'estensione secondo la procedura descritta nel Capitolo 6.

5 Altro esempio classico

Dopo un esempio di apprendimento supervisionato e un esempio con trattamento di testo propongo un ultimo esempio, questa volta di apprendimento non supervisionato: segmentazione della clientela con un algoritmo di clustering.

Stiamo parlando di metodi di analisi applicabili a big data ma continuo ad esemplificare su dimensioni ridotte, per fare prima e capirci meglio.

Abbiamo fatto un'indagine sui nostri clienti - nel caso sono solo 12 ma potrebbero anche essere 120.000 - e siamo venuti a conoscere ciò che vediamo in questa tabella, memorizzata nel file clienti.csv:

cliente	cambio	prezzo	marca	frequenza
Luigi	5	5	5	3
Maria	7	7	4	2
Vittorio	6	5	5	3
Giovanni	6	6	4	2
Beatrice	5	5	4	3
Giuseppe	9	8	2	4
Elena	8	8	2	5
Antonio	9	7	3	4
Paola	10	7	3	4
Mario	9	8	2	4
Carlo	9	10	2	5
Cecilia	10	8	3	3

Le grandezze numeriche variano da 1 (per niente) a 10 (moltissimo).
La colonna «cambio» indica la propensione del cliente a cambiare negozio.
La colonna «prezzo» indica la sensibilità del cliente al prezzo della merce.
La colonna «marca» indica la preferenza del cliente per articoli di marca.
La colonna «frequenza» indica il numero medio di acquisti in un anno.

Aiutati dalla piccola dimensione del data set e dall'ordine con cui ho esposto i dati notiamo subito a occhio che abbiamo un gruppo di clienti, i primi cinque, tendenzialmente quelli che fanno meno acquisti, che hanno una moderata propensione a cambiare negozio, e sono moderatamente sensibili al prezzo e alla marca; questo gruppo si contrappone al gruppo degli altri sette, con molto più elevata propensione a cambiare negozio, molto più sensibili al prezzo, meno sensibili alla marca e mediamente migliori clienti sul piano della frequenza di acquisto.

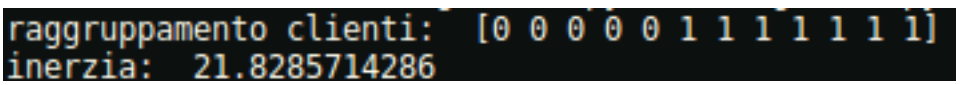
Tutte cose utili per le nostre strategie di marketing o semplicemente per qualche intervento tendente a fidelizzare meglio i sette clienti più «volatili».

Ma vediamo come anche la macchina si accorge di ciò che abbiamo adocchiato noi utilizzando il più diffuso algoritmo di clustering, K-means, noto anche per la sua leggerezza a sollievo del lavoro della CPU del computer.

Con il linguaggio Python lo facciamo con il seguente script

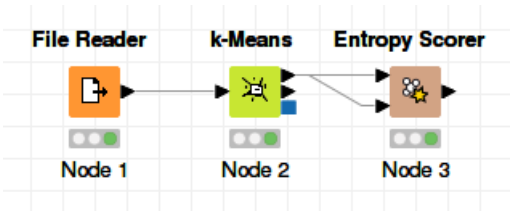
```
import pandas as pd
dati = pd.read_csv('/home/vittorio/Documenti/clienti.csv', usecols = ['cambio', 'prezzo', 'marca', 'frequenza'])
from sklearn.cluster import KMeans
gruppi = KMeans(n_clusters = 2, init = 'k-means++', tol = 0.0001)
gruppi_c = gruppi.fit_predict(dati)
print('raggruppamento clienti:  ', gruppi_c)
print('inerzia:  ', gruppi.inertia_)
```

eseguendo il quale otteniamo il risultato



da cui deriviamo che i primi cinque clienti appartengono al cluster 0 e gli altri sette al cluster 1. L'indicatore chiamato «inerzia» è una somma di errori quadratici medi derivanti dal raggruppamento proposto e, per avere una segmentazione accurata deve essere il più basso possibile. Purtroppo il suo valore assoluto che vediamo dice poco e andrebbe confrontato con altri: ne vedremo l'utilità in tal senso tra poco. Di per sé, comunque, avendo la nostra tabella di partenza dati compresi tra 1 e 10, mi pare che un valore di quasi 22 non possa essere considerato basso: ne deriva che la nostra segmentazione non è gran che, almeno dal punto di vista scientifico, anche se sul piano pratico pare funzionare abbastanza bene.

Con KNIME facciamo tutto questo con tre nodi



Il primo per leggere il file dei dati, `clienti.csv`.

Il secondo per generare i cluster.

Come abbiamo fatto con lo script Python, dove abbiamo indicato 2 nel parametro `n_clusters` di KMeans, anche in questo caso configuriamo il nodo k-Means con 2 nella finestrella NUMBER OF CLUSTERS, verificando che nella finestrella INCLUDE siano elencate tutte le quattro colonne di dati da elaborare. Lasciamo al valore di default (99) il contenuto della finestrella MAX.NUMBER OF ITERATIONS.

Il terzo nodo serve a calcolare l'indicatore della qualità della nostra segmentazione: va accuratamente collegato al nodo precedente come si vede nell'illustrazione e va configurato in modo che, nella finestra di configurazione, la REFERENCE COLUMN sia «cliente» e la CLUSTERING COLUMN sia «Cluster».

Eseguiti i tre nodi, nella finestra di configurazione del nodo K-MEANS, in fondo, possiamo aprire la finestra LABELED INPUT, dove vediamo il nostro dataset suddiviso nei due cluster che avevamo individuato noi a occhio e con lo script Python:

Labeled input - 0:2 - k-Means

File Hilite Navigation View

Table "clienti.csv" - Rows: 12 | Spec - Columns: 6 | Properties | Flow Variables

Row ID	S cliente	I cambio	I prezzo	I marca	I frequenza	S Cluster
Row0	Luigi	5	5	5	3	cluster_0
Row1	Maria	7	7	4	2	cluster_0
Row2	Vittorio	6	5	5	3	cluster_0
Row3	Giovanni	6	6	4	2	cluster_0
Row4	Beatrice	5	5	4	3	cluster_0
Row5	Giuseppe	9	8	2	4	cluster_1
Row6	Elena	8	8	2	5	cluster_1
Row7	Antonio	9	7	3	4	cluster_1
Row8	Paola	10	7	3	4	cluster_1
Row9	Mario	9	8	2	4	cluster_1
Row10	Carlo	9	10	2	5	cluster_1
Row11	Cecilia	10	8	3	3	cluster_1

Nella finestra di configurazione del nodo Entropy Scorer, in fondo, possiamo aprire la finestra QUALITY TABLE che è la seguente

Quality Table - 0:3 - Entropy Scorer

File Hilite Navigation View

Table "Entropy Scores" - Rows: 3 | Spec - Columns: 4 | Properties | Flow Variables

Row ID	I Size	D Entropy	D Normalized Entropy	D Quality
cluster_0	5	2.322	0.648	?
cluster_1	7	2.807	0.783	?
Overall	12	2.605	0.727	0.273

In questo caso l'indicatore di qualità è l'entropia ed è calcolato in modo più sofisticato rispetto all'indicatore che abbiamo visto in Python (inerzia). Anche in questo caso la maggiore qualità si ha quando l'indicatore è basso. Per aiutarci a valutarlo la tabella ce ne propone il valore normalizzato (tradotto in variabilità tra 0 e 1) e a fianco ci espone l'indicatore di qualità (quanto avanza tra il valore normalizzato dell'entropia e 1).
Anche da questi indicatori emerge che la nostra segmentazione su due gruppi è poco descrittiva del nostro parco clienti.

Questa volta, molto probabilmente, vince in semplicità e chiarezza KNIME.

Certo che bisogna poi anche tener conto della facilità con cui questi nostri piccoli esercizi si possano inserire in più ampi interventi informatici. Molto probabilmente, nella realtà, i clienti sono più di 5.000 e il complessivo intervento informatico deve prevedere l'invio automatico di una lettera di un certo tenore ai 728 clienti del cluster 2 e di una lettera di altro tenore ai 925 clienti del cluster 4.
Se il problema si allarga in questo modo, temo che sia meglio lavorare con Python.

Ma il nostro piccolo esempio ha una coda.

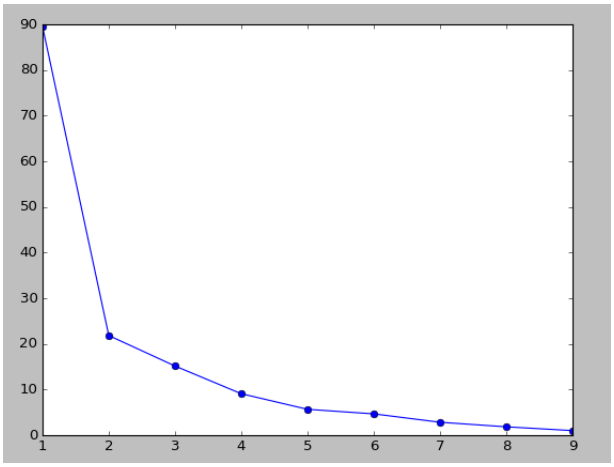
Abbiamo visto che a K-means dobbiamo dire noi il numero dei cluster da definire ed abbiamo poi un indicatore che ci lascia intendere se la classificazione è realistica o meno e questa è una bella limitazione: nel nostro caso, a occhio, abbiamo visto che sarebbe stato ragionevole limitare l'analisi a due cluster, poi abbiamo scoperto che la macchina non ha ritenuto molto buona la separazione in due gruppi. Di fronte a una dataset di migliaia di clienti, quando non abbiamo nemmeno la possibilità di farci un'idea a occhio, chi stabilisce se i cluster sufficientemente descrittivi del parco clienti siano due, sette o quattordici?

Fortunatamente troviamo risposta al problema sia con Python sia con KNIME applicando un procedimento iterativo che, approfittando della velocità di elaborazione di K-means e del poco lavoro che esso dà alla CPU, determina un numero via via crescente di cluster dandoci conto del miglioramento che ogni cluster in più fornisce alla descrizione del dataset in modo che possiamo decidere consapevolmente il numero di cluster da definire.

Riferendoci ai dati del nostro esempio, con Python tutto ciò è possibile lanciando il seguente script:

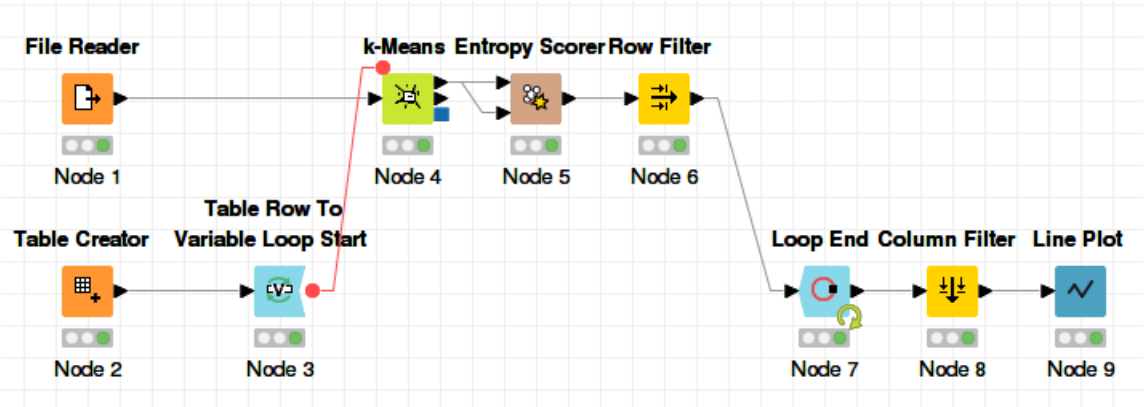
```
import pandas as pd
dati = pd.read_csv('/home/vittorio/Documenti/clienti.csv', usecols = ['cambio', 'prezzo', 'marca', 'frequenza'])
from sklearn.cluster import KMeans
inerzie = [ ]
for clusters in range(1,10):
    gruppi = KMeans(n_clusters = clusters, init = 'random', tol = 0.0001)
    gruppi.fit_predict(dati)
    inerzie.append(gruppi.inertia_)
import matplotlib.pyplot as plt
plt.plot(range(1,10), inerzie, marker = 'o')
plt.show()
```

Il risultato è questo grafico



che visualizza la diminuzione dell'inerzia (miglioramento della qualità della segmentazione) all'aumentare dei cluster, indicati sull'asse delle ascisse: dove la curva della diminuzione dell'inerzia tende ad appiattirsi diventa inutile aumentare i cluster.
Nel nostro caso vediamo che, con due cluster abbiamo diminuito bene l'inerzia ma avremmo forse da guadagnare in descrizione utile del nostro gruppo di clienti andando a quattro cluster. E ciò conferma quanto è risultato prima ma ora abbiamo un'idea più chiara sul numero ottimale dei cluster da prevedere.

Il trattamento di questo problema con KNIME è un po' più laborioso.



Partiamo con il primo nodo, FILE READER, quello con cui si legge il dataset contenuto nel file `clienti.csv`, il cui path inseriamo nella finestra di configurazione (che, ricordo, raggiungiamo con click destro sul nodo).

Con il secondo nodo, TABLE CREATOR, che sistemiamo sotto al primo, creiamo una tabella contenente tante righe quante sono le iterazioni di cui abbiamo bisogno per sperimentare il numero di cluster.
Quando prima ho usato Pyhton avevo scelto di fare 10 cicli (vedi la riga `for clusters in range(1,10)`) ed altrettanto facciamo ora. Il numero dei cluster da sperimentare va scelto facendo in modo che non sia né troppo basso (altrimenti faremmo una ricerca inutile) né troppo alto (altrimenti faremmo lavorare il computer per ore): anche per dataset molto ampi e con molta varietà di situazioni rappresentate penso che non si debba eccedere le poche decine. Nel nostro caso, 10 prove su un dataset di dodici righe direi che basta e avanza (anche perché oltre dodici sarebbe inutile andare con solo dodici dati da raggruppare).
La tabella si costruisce aprendo la finestra di configurazione e agendo nella finestra TABLE CREATOR SETTINGS come si trattasse di un foglio di calcolo.
Cliccando destro sull'intestazione della prima e unica colonna che utilizzeremo scegliamo COLUMN PROPERTIES... dal menu che si apre e, nella successiva finestra di dialogo, inseriamo il nome della colonna nella finestrella NAME: visto che stiamo usando l'algoritmo K-means è consuetudine che questo nome sia `k`; poi scegliamo NUMBER (INTEGER) nel menu a cascata della finestrella TYPE.
Infine inseriamo i numeri da 1 a 10 nelle prime dieci righe della colonna.

Eseguiamo i primi due nodi in modo da ottenere in entrambi il semaforo verde per poter proseguire.

Di fianco al secondo nodo inseriamo il nodo che, utilizzando le righe della tabella costruita nel secondo nodo, fa partire il ciclo, TABLE ROW TO VARIABLE LOOP START. Colleghiamo il secondo nodo con questo terzo e possiamo eseguire senza bisogno di configurazione. Notiamo che la forma stessa del disegno del nodo, con un incavo sulla destra, vuole richiamare che qui inizia qualche cosa che dovrà chiudersi: e la protuberanza sulla sinistra del nodo LOOP END, corrispondente all'incavo, che inseriremo alla fine del ciclo vuole richiamare l'attesa chiusura. Notiamo altresì che l'uscita del nodo non è rappresentata dalla solita freccina nera ma è rappresentata da un pallino rosso: ciò significa che da qui esce una variabile (altro non è che, in successione, il numero 1, 2, 3, ecc. fino a 10 che avevamo inserito nella colonna del secondo nodo).

Torniamo ora, nel workflow editor, in dirittura del primo nodo e inseriamo il nodo K-MEANS.
Innanzi tutto dobbiamo inserire nel ciclo ciò che farà questo nodo e, per farlo, dobbiamo collegare la variabile in uscita dal terzo nodo con quest'altro. Dal momento che, per default, i nodi non evidenziano le prese di inserimento di variabili, apriamo la configurazione di k-Means e clicchiamo sulla voce SHOW FLOW VARIABLE PORTS in modo da far comparire nel disegno del nodo queste prese, che avranno l'aspetto di pallini rossi. Colleghiamo il pallino rosso di uscita dal terzo nodo con il pallino rosso di entrata di sinistra di k-Means.
Ora colleghiamo anche il primo nodo a k-Means e configuriamo. Aperta la finestra di configurazione clicchiamo sul pulsante a destra della finestrella NUMBER OF CLUSTERS, pulsante che contiene il disegno di una V, e indichiamo `k` nella finestrella USE VARIABLE della successiva finestra di dialogo. Lasciamo a 99 il MAX. NUMBER OF ITERATIONS e verifichiamo che la finestra INCLUDE contenga tutte le quattro colonne di dati del nostro dataset.
Eseguiamo in modo da ottenere semaforo verde.

Ora dobbiamo inserire il nodo per misurare la qualità di ciò che ha fatto k-Means, ENTROPY SCORER. Lo configuriamo facendo in modo che la REFERENCE COLUMN sia «cliente» e la CLUSTERING COLUMN sia «Cluster» e eseguiamo.

Dal momento che della tabella che sforna Entropy Scorer ad ogni passaggio ci interessano solo le righe contrassegnate «overall», introduciamo un nodo per catturare solo quelle: ROW FILTER. Lo sistemiamo di fianco a Entropy Scorer, lo colleghiamo con esso e lo configuriamo scegliendo l'opzione INCLUDE ROWS BY ROW ID e inserendo `Overall` nella finestrella REGULAR EXPRESSION.

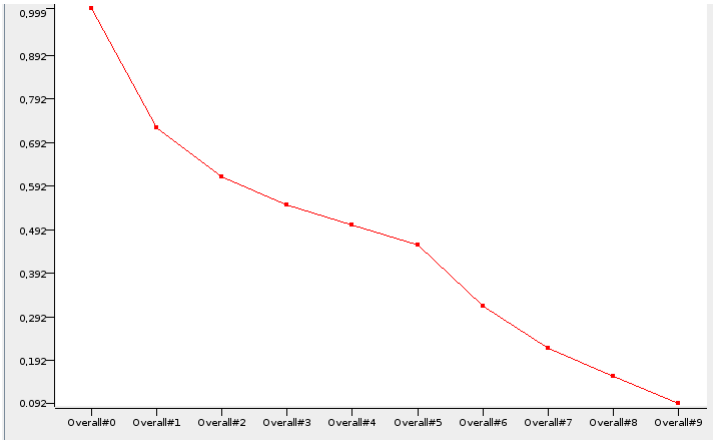
E' venuto il momento di chiudere il nostro ciclo e lo facciamo introducendo il nodo LOOP END. Verifichiamo che la configurazione abbia la scelta UNIQUE ROW IDS BY APPENDING A SUFFIX nella ROW KEY POLICY (dovrebbe essere la proposta di default) ed eseguiamo.

A ciclo terminato il nodo Loop End genera una tabella riepilogativa del ciclo stesso con tutte le dieci righe Overall che abbiamo scelto di catturare ad ogni passaggio: la potremmo estrarre aprendo il menu del nodo e cliccando in fondo su COLLECTED RESULTS.

Ma con un ulteriore piccolo sforzo possiamo estrarre un grafico, tanto per fare le stesse cose che avevamo fatto con Python.

Aggiungiamo, allora, un nodo per selezionare la colonna della tabella da evidenziare nel grafico. Il nodo si chiama COLUMN FILTER. Lo piazziamo di fianco al Loop End, lo colleghiamo con esso e lo configuriamo in modo che la finestrella INCLUDE contenga una sola colonna, per esempio scegliamo la colonna `Normalized Entropy`, e eseguiamo.

Finalmente aggiungiamo il nodo per produrre il grafico, che si chiama LINE PLOT, che, una volta collegato al precedente, possiamo eseguire nella sua configurazione di default e, scegliendo VIEW: LINE PLOT dal menu del nodo, visualizziamo il seguente grafico



Sull'asse delle ascisse abbiamo l'identificativo di riga anziché il numero dei cluster: con un paio di nodi o tre in più avremmo potuto avere il numero dei cluster. Ho rinunciato ad appesan-

tire l'esempio in quanto basta tener presente che Overall#0 corrisponde a 1 cluster, Overall#1 corrisponde a 2 cluster, ecc.

Il risultato che abbiamo, questa volta basato sul calcolo dell'entropia, è ancora più pessimistico di quello che avevamo avuto con Python, basato sul calcolo dell'inerzia, circa la possibilità di trovare segmentazioni scientificamente fondate del nostro parco clienti.

Per quanto riguarda il nostro confronto tra Python e KNIME, quest'ultimo squarcio di esempio, secondo me, depone nettamente in favore di Python.

Ma ciascuno ha le sue preferenze.

Devo comunque riconoscere che anche questo modo di lavorare con KNIME, che ricorda il mondo educativo per piccoli programmatori di Scratch, ha il suo fascino.

6 Estensioni

Per arricchire il nostro KNIME di funzioni aggiuntive abbiamo a disposizione il menu HELP ▷ INSTALL NEW SOFTWARE... con cui apriamo la finestra di dialogo AVAILABLE SOFTWARE.

Nella finestrella WORK WITH possiamo scegliere un luogo in cui trovare il software da installare: se non abbiamo altre idee ci conviene scegliere –ALL AVAILABLE SITES–.

Ci si presenta così un nutrito elenco di luoghi, tra cui fanno spicco quelli di communities di utenti KNIME e altri con contenuto specialistico.
Selezionando un luogo otteniamo l'elenco del software presente.
Se nella finestra di dialogo abbiamo selezionato l'opzione HYDE ITEMS THAT ARE ALREADY INSTALLED vedremo solo il software che non abbiamo installato, altrimenti vedremo tutto il software, contrassegnato da iconcina a colori quello non ancora installato e installabile, contrassegnato da iconcina in sfumatura di grigio quello già installato.

Il luogo dove troviamo cose di utilità generale è KNIME & EXTENSIONS. Apriamo la relativa voce e vediamo un elenco che comprende il software KNIME TEXTPROCESSING.

Per eventualmente installarlo selezioniamo la relativa voce e clicchiamo sul pulsante FINISH della finestra di dialogo: si avvia così l'installazione che richiede qualche tempo e veniamo informati sul progredire dell'installazione stessa con una piccola scritta in fondo a destra della finestra di KNIME.

7 Per saperne di più

Un buon testo italiano che ci guida ad utilizzare KNIME in una ampia serie di situazioni è Andrea De Mauro - Big Data Analytics, Analizzare e interpretare dati con il machine learning edito da Apogeo.

Ottima, ma scritta in inglese, la documentazione inserita nella stessa applicazione: selezionando un nodo nel repository in basso a sinistra troviamo la descrizione di ciò che esso fa e di come lo si fa nella finestrella in alto a destra.

Sul sito di KNIME, infine, troviamo la documentazione completa e segnalo, in particolare, l'E-Learning Course cui abbiamo accesso dalla scheda LEARNING del sito stesso. Vi troviamo utilissime lezioni su vari argomenti, con esemplificazioni e, spesso, con filmati illustrativi recitati in un inglese pronunciato all'europea, cioè molto più comprensibile ai meno iniziati alla lingua parlata, in più con lo scritto in sottotitoli.

8 Se proprio serve...

Sono partito sottolineando che il pregio di KNIME è quello di consentirci analisi di qualunque tipo sui big data senza scrivere una riga di codice.

Può tuttavia accadere che, per impostare meglio le nostre analisi, abbiamo bisogno di intervenire sul set di dati da analizzare con elaborazioni di vario tipo che aggiungano al set altri dati derivati dai precedenti o che li sostituiscano ricorrendo determinate condizioni.

Per rispondere a queste esigenze KNIME ha un nodo, che si chiama MATH FORMULA, che contiene un'ampia serie di funzioni matematiche utilizzabili per le elaborazioni necessarie: si va dalle semplici operazioni aritmetiche all'applicazione di funzioni trigonometriche e all'applicazione di funzioni di raffronto logico.

Se non troviamo qui quanto ci serve KNIME ci offre una scappatoia e ci mette a disposizione un nodo, che si chiama JAVA SNIPPER, utilizzando il quale scriviamo noi il codice della funzione che ci serve in linguaggio Java, che è il linguaggio utilizzato da KNIME.

Ed è solo in questi casi, se proprio serve, che siamo chiamati a scrivere codice.

Se, poi, non vogliamo usare il linguaggio Java, KNIME ci dà la possibilità di installare estensioni per l'uso del linguaggio R o del linguaggio Python.

Per ciascuna di queste rare esigenze rimando alle istruzioni contenute nel testo di Andrea De Mauro citato nel precedente capitolo.