Progetto di: Architetture Dati

Andrea Premate 829777, Tiberio Falsiroli 874971

AA 2021/2022

Contents

1 Introduction:

In questa prima parte osserveremo da vicino le premesse necessarie a definire il lavoro da noi svolto e il suo valore.

1.1 Inspector Gadget:

Inspector Gadget è un framework che semplifica la stratificazione delle capacità di monitoraggio e debug su un "Dataflow Engine" esistente. Inspector gadget inoltre fornisce astrazioni per osservare i dati nel loro percorso attraverso il flusso, etichettare parti di questi dati, visualizzare tag nei punti di osservazione, scambiare messaggi tra coppie di punti di osservazione e/o con un nodo coordinatore centrale.

1.2 KNIME:

Riteniamo doveroso spendere due parole per introdurre la piattaforma da noi utilizzata per questo progetto: KNIME. Grazie a questo tool è stato possibile costruire un workflow di prova ed implementare dei nodi (all'interno del workflow stesso) per eseguire le operazioni in stile Inspector Gadget. Questo tool permette in maniera grafica, quindi intuitiva, di aggiungere/togliere nodi per il monitoring, la modifica, la rimozione, l'unione e l'analisi di un insieme di dati. Grazie a KNIME è possibile implementare anche dei task di machine learning, come da noi fatto per rendere piu significativo il test. KNIME mette a disposizione una serie molto vasta di nodi "prefabbricati" per specifici task, tuttavia permette anche la creazione di Java Snippet totalmente customizzabili per poter soddisfare richieste più specifiche, una opzione a cui siamo ricorsi spesso nel nostro progetto.

1.3 Obbiettivi progettuali:

L'obbiettivo di questo progetto è quello di implementare una concretizzazione del framework Inspector gadget applicandola ad un workflow esistente (creato da noi), per fare in modo che eventuali bug o incongruenze sorte durante il cammino dei dati attraverso il workflow, vengano risolti o almeno identificati e notificati.

Le funzionalità principali sulle quali ci siamo concentrati sono:

- Identificazione di dati/operazioni che causano crash al workflow.
- Monitoring dei dati in vari punti del workflow per assicurarci la consistenza.
- Calcolo di statistiche sui dati delle tabelle per garantire l'healtness del workflow

In seguito in questa relazione verranno approfonditi tutti gli "Inspector Gadget behaviours" da noi implementati nel nostro workflow di esempio.

2 Dataset

Per il nostro workflow abbiamo preso un dataset esistente dal sito UCI Machine Learning Repository, adatto ad essere sottoposto a task di machine learning. Il dataset in questione è: ***Statlog (Shuttle) Data Set*** ed è composto da nove attributi numerici.

Data Set Characteristics:	Multivariate	**Number of Instances:**	58000	**Area:**	Physical
Attribute Characteristics:	Integer	**Number of Attributes:**	9	**Date Donated**	N/A
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	162502

Questo set di dati è stato originariamente generato per estrarre regole comprensibili per determinare le condizioni in cui un atterraggio automatico sarebbe preferibile al controllo manuale di un veicolo spaziale. Il compito è decidere quale tipo di controllo della nave dovrebbe essere impiegato. Il dataset dello shuttle contiene 9 attributi, tutti numerici. Sono disponibili 7 valori per l'etichetta della classe (target): Nel dataset non sono presenti valori mancanti. La sua composizione è di 58000 entries e sono state così distribuite per il task di machine learning:

- Training Set: 43500 entries
- Test Set: 14500 entries

La tabella in questione rappresenta è formata da 9 attributi tutti numerici Nel corso del workflow i dati vengono "contaminati" tramite l'inserimento di righe corrotte, modifiche scorrette e cancellazioni, per poter evidenziare il lavoro svolto dai nostri nodi di monitoring.

La documentazione dettagliata del dataset può essere consultata a questo indirizzo: http://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)

3 Workflow

In questa sezione andremo ad analizzare passo passo il nostro workflow. Come possiamo notare in figura 1, il workflow è diviso in tre macro-aree:

- **Main Workflow Process**: racchiude al suo interno tutti i nodi necessari per i task più comuni, come la connessione al database, la creazione di tabelle e l'esecuzione di task di machine learning.
- **Processing dei Controlli**: racchiude al suo interno tutti i nodi che eseguono un task necessario per implementare una funzionalità di inspector gadget. Possiamo dire che questo layer è una specie di engine alimentato dal layer sottostante per la realizzazione del layer sovrastante ovvero il monitor inspector gadget.
- **Inspector gadget Monitor**: racchiude tutti i nodi che mostrano i risultati dell'analisi inspector gadget sulla salute del workflow. Ipoteticamente un operatore, leggendo l'output di questi nodi può acquisire una serie di informazioni utili e/o alert riguardanti la salute del flusso dei dati e agire di conseguenza.

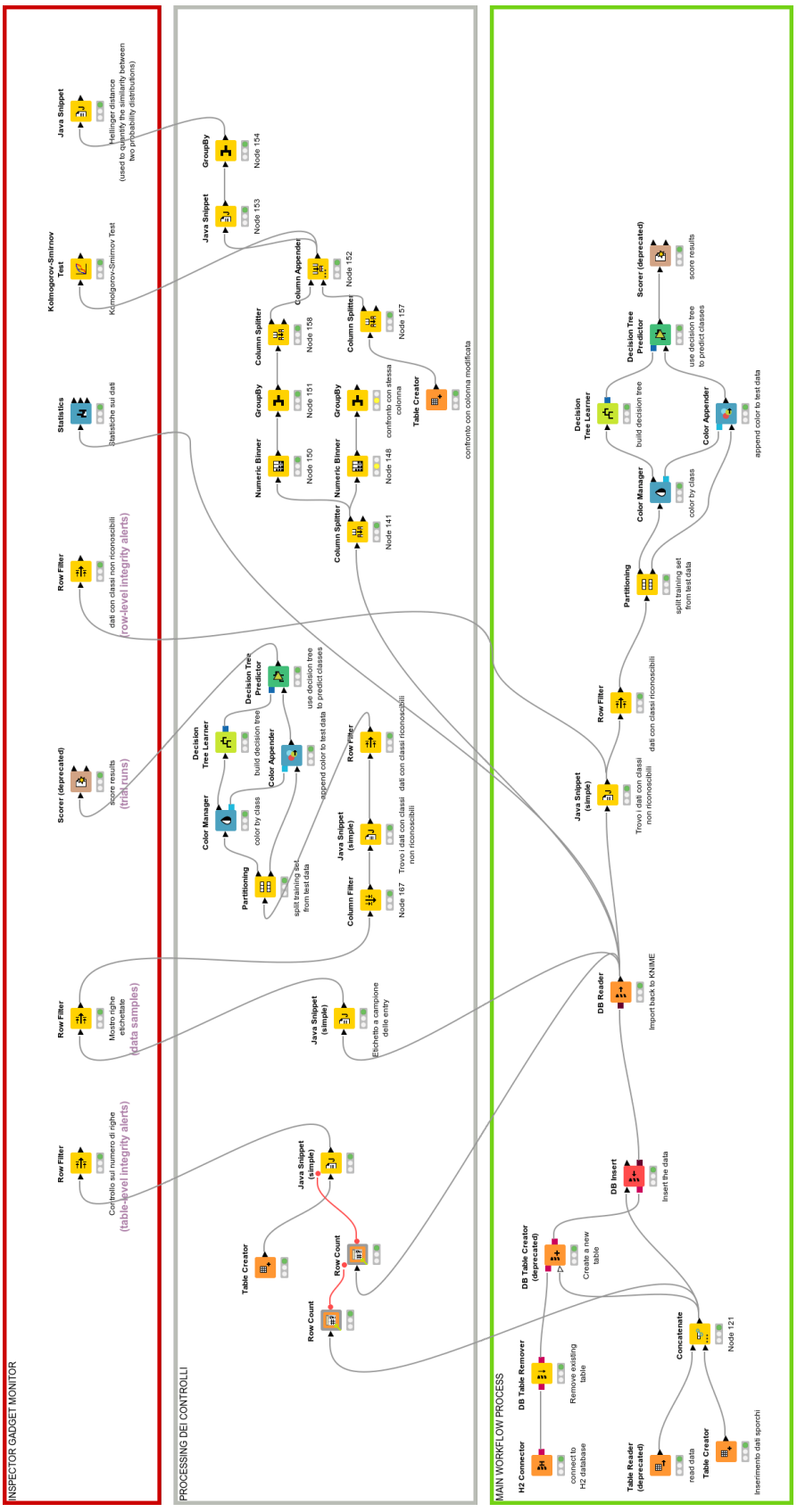


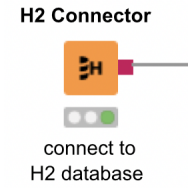
Figure 1: Workflow

3.1 Main Process Workflow

In questa sezione analizzeremo nel dettaglio i nodi usati per la creazione del workflow di esempio.

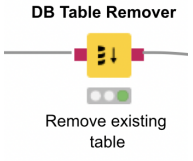
3.1.1 H2 Connector

Questo nodo crea una connessione a un file di database H2 tramite il suo driver JDBC. Questo nodo utilizza il modello di URL JDBC del driver selezionato per creare l'URL del database concreto. *Documentazione*.



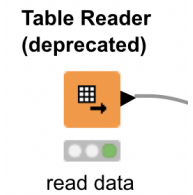
3.1.2 DB Table Remover

Questo nodo rimuove una tabella con il nome specificato dal database descritto dalla connessione DB in ingresso. Questo nodo è necessario nel nostro workflow per fare in modo che ogni esecuzione sia indipendente dalla precedente, in parole povere, serve per pulire i residui di dati dell'esecuzione precedente, ancora presenti in memoria. *Documentazione*.



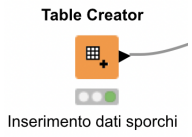
3.1.3 Table Reader

Questo nodo server per importare all'interno di Knime i nostri dati della tabella SHUTTLE.table presente nel file system. *Documentazione*.



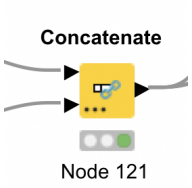
3.1.4 Table Creator

Ci siamo avvalsi di questo nodo per l'inserimento di dati "sporchi" aggiungendo in totale 7 righe con dominio non accettabile per il valore Target della tabella (Classi inesistenti, o con attributo mancante). *Documentazione*.



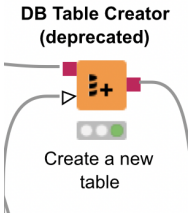
3.1.5 Concatenate

Questo nodo serve per concatenare la tabella con i dati originali con la nostra tabella contenente i dati "sporchi" *Documentazione*.



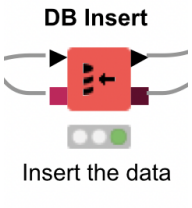
3.1.6 DB Table Creator

Questo nodo serve per creare una tabella nel formato accettato da un database H2. *Documentazione*.



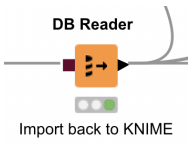
3.1.7 DB Insert

Questo nodo si occupa di inserire nel database la tabella appena creata riempita grazie a questo nodo con i dati originali e i dati "sporchi" aggiunti manualmente da noi. *Documentazione*.



3.1.8 DB Reader

Importa nuovamente i dati di interesse dal database al nostro workflow knime grazie all'esecuzione di una query. *Documentazione*.

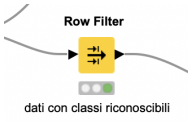


3.1.9 Java Snippet

Grazie a questo nodo è possibile separare le entry della tabella aventi classi non riconoscibili dal sistema (marcate con un nuovo attributo intero posto a 1). Queste vengono isolate e rimosse dalla tabella originale per permettere al dataset di proseguire senza generare errori. Questo nodo è approfondito al paragrafo 4.4

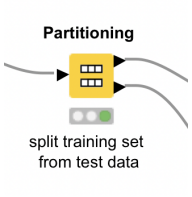
3.1.10 Row Filter

In questo nodo tutte le classi non riconoscibili, vengono filtrate e rimosse dalla tabella.



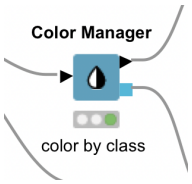
3.1.11 Partitioning

Nodo usato per dividere la tabella in train e test set. Nel nostro caso, come indicato in precedenza, abbiamo deciso di usare 43500 entry per il trainset e 14500 per il testset. *Documentazione*.



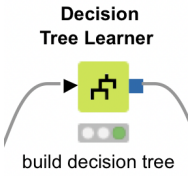
3.1.12 Color Manager

Fissato un attributo associa un colore differente a ogni elemento differente del suo dominio. Nel nostro caso ad ogni classe veniva associato un colore diverso. *Documentazione*.



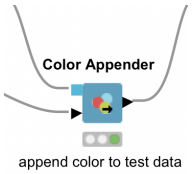
3.1.13 Decision Tree Learner

Questo nodo rappresenta un decision tree. L'attributo target deve essere nominale; gli altri attributi possono essere sia numerici (come nel nostro caso) sia nominali. *Documentazione*.



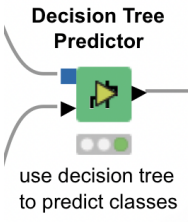
3.1.14 Color Appender

Applica l'associazione fatta nel nodo "Color Manager" ad ogni record presente nella tabella. *Documentazione*.



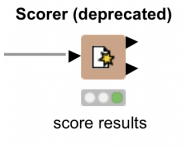
3.1.15 Decision Tree Predictor

Questo nodo utilizza un albero decisionale esistente (passato attraverso la porta di input) per prevedere il valore della classe del test set. *Documentazione*.



3.1.16 Scorer

Confronta due colonne in base alle coppie di valori di attributo e mostra la matrice di confusione e una tabella contenente alcune misure di performance. *Documentazione*.



4 Inspector Gadget Monitor

In questa sezione verranno analizzati nel dettaglio gli "Inspector Gadget behaviours" implementati per il monitoring del nostro flusso di dati. L'idea è quella di fornire un "monitor" dove le anomalie e controlli di healtness vengano presentati in maniera compatta.

4.1 Table-level Integrity Alerts

Descrizione: Lanciare un alert quando un dataset presente in passaggi intermedi dell'elaborazione viola un certo vincolo, ad esempio la sua cardinalità deve essere > 0.

Nel nostro specifico caso abbiamo scelto di effettuare un controllo sul numero di righe presenti nella nostra tabella principale in due punti del workflow: al momento del caricamento della table e nel momento della sua rilettura dal database. Questo ci permette di capire se durante la manipolazione dei dati qualche riga è andata perduta per qualche motivo o se sono state inserite entry illegalmente.

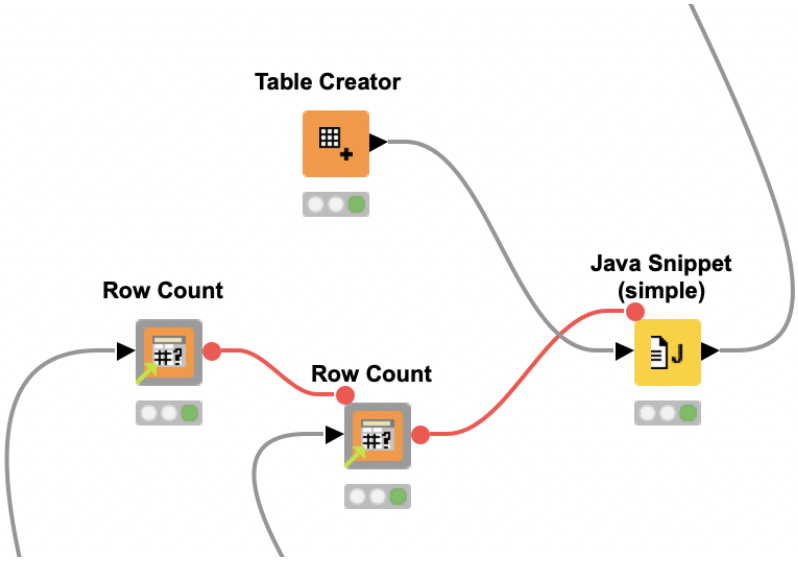


Figure 2: Parte del workflow riguardante Table-level Integrity Alerts.

Nella figura sottostante possiamo vedere la semplice implementazione del java snippet che permette il controllo descritto sopra. Dopo aver ricevuto in input il conteggio delle righe in due punti del workflow differenti, mette questi valori a confronto. Se il controllo ha esito positivo, viene salvata una entry nella tabella temporanea creata nel nodo "Table Creator" (visibile nella figura in alto) e in seguito passata al nodo per la visualizzazione nel monitor Inspector

gadget. Analogamente, nel caso di esito negativo del controllo, viene inserita una entry che lo indica.

```
Method Body
  Double bin=0.0;

if($${IRowcountDB}$$.equals($${IRowCountInit}$$)){
        bin=1.0;
}
  return bin;
```

Figure 3: Metodo per controllo sulle righe

4.2 Data Samples

Descrizione: Mostrare un sottoinsieme dei dati del workflow come sanity check per trovare dati sistematicamente errati, come ad esempio una colonna con solo valori nulli.

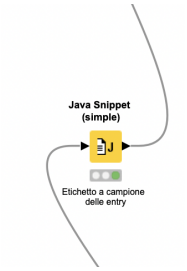


Figure 4: Parte del workflow riguardante Data Samples.

Nel nostro workflow questo controllo è reso possibile da un java snippet che racchiude un metodo (visionabile nella figura in basso) che semplicemente etichetta delle righe prese casualmente e le marca come candidate per essere mostrate nel nodo del monitor Inspector gadget. Per fare ciò, come vediamo in figura, viene aggiunta una colonna alla tabella originale e ogni N righe riempite con attributo 0.0, ne viene riempita una con attributo 1.0; questa riga (insieme alle altre marcate con 1.0) sarà mostrata come sanity check.

```
Method Body
  Double bin=0.0;
boolean val = new Random().nextInt(50)==0;

if(val){
bin=1.0;
}
  return bin;
```

Figure 5: Metodo per Data Samples

4.3 Trial Runs

Descrizione: Eseguire il workflow su un piccolo sottoinsieme dei dati di input come veloce sanity check per vedere se avviene un crash o se si ottengono risultati coerenti con le aspettative.

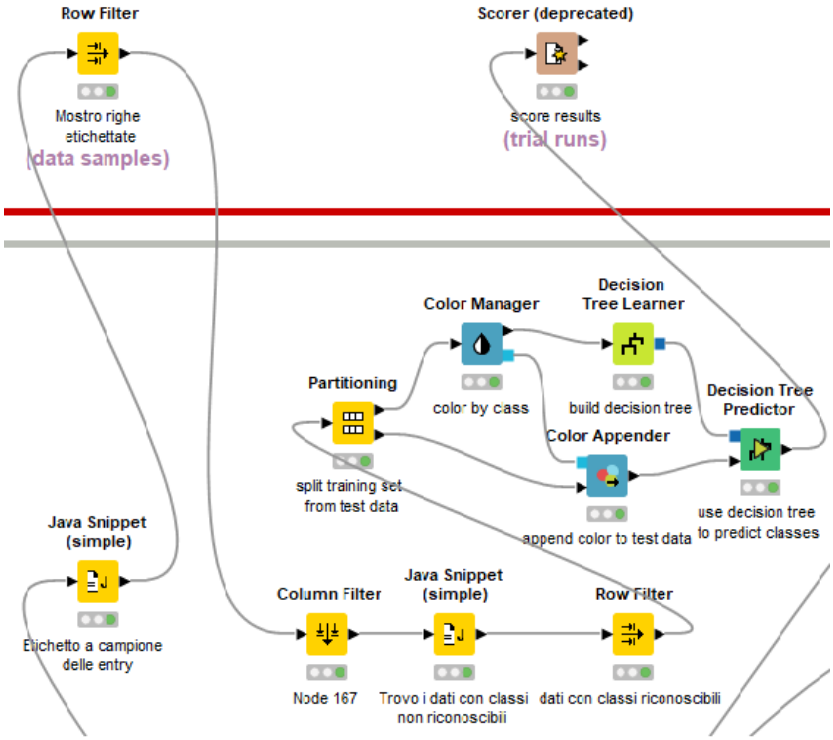


Figure 6: Parte del workflow riguardante Trial Runs.

Come mostrato in figura 6 nel workflow da noi implementato si utilizza un sample proveniente da un elemento stesso della parte di monitoring: "Data samples". Su tale campione di dati viene quindi eseguito il task di classificazione esattamente come avviene nel workflow del processo principale. Idealmente vengono utilizzate le stesse funzioni del workflow principale, nel nostro caso abbiamo semplicemente copiato la parte di workflow riguardante la classificazione e l'abbiamo posizionata nella macro area "Processing dei Controlli" per una maggiore chiarezza semantica e visiva.

4.4 Row-level Integrity Alerts

Descrizione: Lanciare un alert ogni volta che un record viola un certo vincolo, ad esempio il campo X non deve essere nullo, il campo numerico Y deve essere ≥ 0. Nel nostro caso abbiamo scelto di effettuare un controllo sull'attributo target delle nostre entry.

```
Double bin;
if ($Col9$.equals("class0")||$Col9$.equals("class1") || $Col9$.equals("class2") || $Col9$.equals("class3") ||
    $Col9$.equals("class4") || $Col9$.equals("class5") || $Col9$.equals("class6")) {
      bin=1.0;
}else{
  bin = 2.0;
}

return bin;
```

Figure 7: Metodo per Row-level Integrity Alerts

Impostando a priori in un Java Snippet le classi con cui il workflow si aspetta di lavorare, siamo in grado di individuare entry anomale e in seguito di isolarle e/o mostrarle nel monitor Inspector Gadget. In figura 7 possiamo vedere il metodo racchiuso nel java snippet. Controlla ogni entry della tabella di ingresso e marca le righe che hanno una classe indesiderata. Per fare ciò aggiunge semplicemente una colonna di valori Interi alla tabella di ingresso; le entry "legali" vengono marcate con il valore 2, quelle indesiderate, con il valore 0. Lo step successivo del Workflow è un java snippet che si occupa di isolare queste entry con target non riconoscibile e di eliminarle in modo che il resto dei task (ad esempio quello di machine learning) possano essere eseguiti senza anomalie.

Figure 8: Nodo per controllo Row-level Integrity Alerts

4.5 Data summaries

Descrizione: Calcolare statistiche sui dati (ad esempio un istogramma) su una run del workflow e ad esempio confrontare tali statistiche con run precedenti che elaboravano dati simili, ad esempio che dovrebbero provenire dalla stessa distribuzione di probabilità. Lo scopo è quindi quello di trovare cambiamenti improvvisi delle distribuzioni dei dati che potrebbero indicare un errore di processing.

Questo behaviour lo abbiamo articolato in tre parti: Statistics, Komolgorov-Smirnov Test e Hellinger Distance. Vedere figura 9.

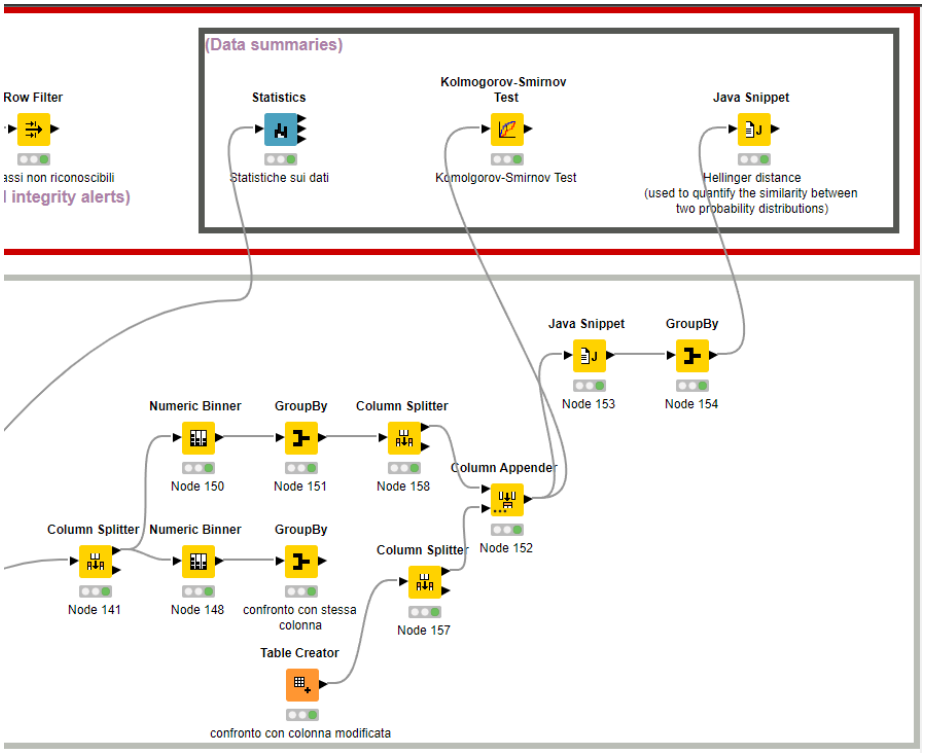


Figure 9: Parte del workflow riguardante Data Summaries.

Statistics Dopo la lettura dei dati dal DB vengono calcolate immediatamente delle statistiche relative ai dati stessi, concentrandosi sulla distribuzione dei vari attributi della table. In figura 10 possiamo vedere un esempio riguardante alcuni attributi numerici; alcune delle statistiche calcolate sono il valore minimo, il valore massimo, la media, la deviazione standard, la varianza, la skewness, la kurtosis, la somma di tutti i valori, il numero di entry mancanti, in modo opzionale è possibile calcolare la mediana (non viene calcolata di default perché computazionalmente più onerosa) ed in particolare per un controllo visivo veloce

ed efficace è riportato l'istogramma.

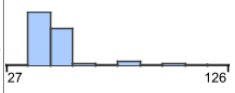
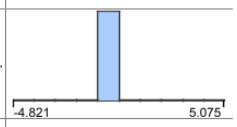
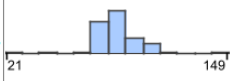
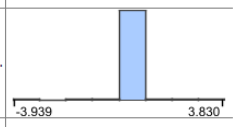
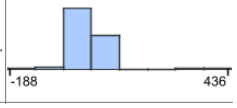
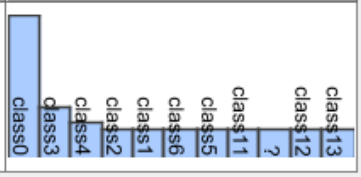
Column	Min	Max	Mean	Std....	Variance	Skewness	Kurtosis	Overall sum	No. miss...						Histogram
Col0	27	126	48.239	12.237	149.753	2.181	6.51	2,798,173	0	0	0	0	?	...	
Col1	-4,821	5,075	-0.019	77.953	6,076.722	6.439	2,647.637	-1,128	0	0	0	0	?	...	
Col2	21	149	85.348	8.902	79.253	1.096	0.544	4,950,809	0	0	0	0	?	...	
Col3	-3,939	3,830	0.26	36.519	1,333.66	31.69	7,699.154	15,061	0	0	0	0	?	...	
Col4	-188	436	34.548	21.659	469.131	-1.162	8.547	2,004,032	0	0	0	0	?	...	

Figure 10: Statistiche riguardanti i primi cinque attributi(numerici) del dataset.

Per quanto riguarda invece gli attributi nominali (vedere figura 11) è possibile visualizzare il numero di entries mancanti, l'istogramma e la frequenza assoluta e relativa.

S Column	I No. missings	Histogram
Col9	2	

S Col9	I Count (Col9)	D Relative Frequency (Col9)
class0	45586	0.786
class3	8903	0.153
class4	3267	0.056
class2	171	0.003
class1	50	0.001
class6	13	0
class5	10	0
class11	3	0
?	2	0
class12	1	0
class13	1	0

Figure 11: Statistiche riguardanti gli attributi nominali del dataset. I dati non erano ancora stati puliti dalle entries sporche quindi si può notare la presenza di valori nulli e di classi non presenti nel dataset originale.

Komolgorov-Smirnov Test Il test di Komolgorov-Smirnov è un test statistico utilizzato per verificare che due campioni di dati provengono dalla stessa distribuzione originaria. In particolare viene calcolato il p-value che assieme al livello di significatività alfa, parametro regolabile dall'utente, determina se l'ipotesi nulla H_0, ossia che i due sample di dati provengono dalla stessa distribuzione, debba essere rifiutata o no. In sintesi: si avrà come output un valore binario. Ricordiamo che il p-value è la probabilità, per una ipotesi supposta vera (ipotesi nulla), di ottenere risultati ugualmente o meno compatibili, di quelli osservati durante il test, con la suddetta ipotesi. In altri termini, il valore p aiuta a capire se la differenza tra il risultato osservato e quello ipotizzato è dovuta alla casualità introdotta dal campionamento, oppure se tale differenza è statisticamente significativa, cioè difficilmente spiegabile mediante la casualità dovuta al campionamento. Nel nostro workflow vengono mostrati due esempi di funzionamento di test statistico in questione. In uno dei due una colonna del dataset viene confrontata con se stessa ed ovviamente il risultato ottenuto è, tautologicamente, che i due sample provengono dalla stessa distribuzione di dati per qualsiasi valore di significatività alfa richiesto. Nel secondo esempio la stessa colonna del dataset viene confrontata con una versione modificata di se stessa: tanto più si modifica la distribuzione originale e tanto si ottiene un p-value minore, con la conseguenza che risulti più facile rigettare l'ipotesi nulla. In un sistema reale quello che realmente verrebbe fatto è, immaginando di rice-

vere a più riprese dei dataset simili poiché provenienti da misurazioni e processi analoghi, eseguire questo test sulle colonne corrispondenti di tali dataset allo scopo di trovare incongruenze dovute ad esempio ad errori di processing o di semantica.

Hellinger Distance La distanza di Hellinger è utilizzata per quantificare la similarità tra due distribuzioni di probabilità. In particolare date due distribuzioni di probabilità discrete $P = (p_1, ..., p_k)$ e $Q = (q_1, ..., q_k)$ la distanza di Hellinger è definita come

$$H(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2} \tag{1}$$

Nel nostro workflow abbiamo implementato tale distanza con l'utilizzo di un Java Snippet. Analogamente a quanto fatto per il test di Komolgorov-Smirnov sono mostrati nel workflow due esempi. In uno dei due una colonna del dataset viene confrontata con se stessa ed ovviamente il valore di distanza ottenuto è 0. Nel secondo esempio la stessa colonna del dataset viene confrontata con una versione modificata di se stessa: tanto più si modifica la distribuzione originale e tanto si ottiene un valore di distanza maggiore. Processando dataset diversi ma che dovrebbero provenire dalla stessa distribuzione di dati poiché provenienti da misurazioni e processi analoghi, si otterrebbero le distanze relative tra tali distribuzioni. Elevando a modello di riferimento una di tali distribuzioni(per attributo) è possibile monitorare tutte le distribuzioni provenienti da dataset successivi confrontandole con quella di riferimento. Immaginando quindi di processare più dataset simili in modo sequenziale avremmo una successione di distanze e diventa quindi piuttosto semplice rilevare una distribuzione anomala: è sufficiente ad esempio confrontare il valore di distanza ottenuto con la media di tutti i valori di distanza, oppure notificare un alert quando tale valore di distanza risulta il massimo rispetto allo storico di distribuzioni processate sino a quel momento. In scenari in cui possa essere previsto un cambio di distribuzione di dati è possibile utilizzare più distribuzioni di riferimento e da qui quindi riconoscere automaticamente la semantica del dato stesso. Ad esempio si immagini una situazione in cui si considerano due dataset analoghi ma per quantificare il tempo in uno vengono utilizzati i giorni mentre nell'altro vengono utilizzate le settimane: avendo due distribuzioni di riferimento, una per casistica, analizzando un dataset generico sarebbe possibile dire se 5, specifico campo di un record, si riferisca ai giorni o alle settimane.