UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Gestionale

Strumenti avanzati per la prognostica di impianti produttivi

Advanced tools for the prognosis of production plants

Relatore: Chiar.mo
Prof. Maurizio Bevilacqua

Tesi di Laurea di:
Alessandra Cittardi

Anno Accademico 2019 / 2020

Sommario

1. Introduzione 2

1.1 Descrizione delle caratteristiche della linea 3

1.2 Descrizione dei dati forniti 6

2. Il Data Mining 8

2.1 Interdisciplinarità del Data Mining 15

2.2 I task del Data Mining 18

2.3 Alberi di decisione: metodologia 23

2.4 Regole di associazione: metodologia 28

3. Il software Weka 30

3.1 Il setting dei parametri in Weka 34

3.2 L'algoritmo J48 (C4.5 algorithm) 41

3.3 L'algoritmo Random Forest 44

3.4 L'algoritmo Apriori 47

4. Applicazione delle tecniche di Data Mining ai dati forniti 50

4.1 Realizzazione del dataset 50

4.2 Alberi di decisione: applicazione 53

4.3 Il Random Forest: applicazione 66

4.4 Regole di associazione: applicazione 68

5. Conclusioni 71

6. Bibliografia e sitografia 73

1. Introduzione

L'obiettivo di questa trattazione consiste nell'eseguire un'analisi dei dati relativi agli eventi di guasto che si verificano in una linea di assemblaggio automatizzata in un orizzonte temporale di un anno e mezzo.

L'analisi è effettuata ricorrendo a delle specifiche tecniche di Data Mining: gli alberi di decisione e le regole di associazione.

Dopo una prima descrizione generale delle caratteristiche della linea e della natura dei dati, nel capitolo 2 viene introdotta la tematica del Data Mining, è esplicitato il contesto in cui è inserito, i legami con altre discipline e le attività tipiche (task). Viene dato inoltre ampio spazio alla trattazione delle metodologie relative ai decision trees ed alle association rules.

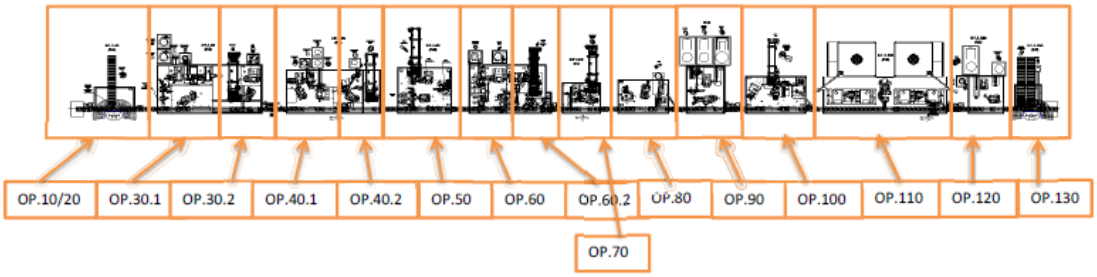
Nel capitolo 3, dopo aver introdotto i vari aspetti ed il funzionamento del software Weka, cioè lo strumento mediante il quale è eseguita l'analisi, viene fornita una descrizione dettagliata degli algoritmi utilizzati, delle modalità con cui interagiscono con i dati e dei rispettivi parametri.

Il capitolo 4 si focalizza sulla fase di sperimentazione, cioè di applicazione delle tecniche di Data Mining, e quindi degli algoritmi, sui dati forniti. È spiegata inoltre tutta l'attività di pre-processing eseguita sui dati grezzi e la creazione di un dataset ad hoc da dare in input al software.

All'applicazione segue poi una descrizione ed una valutazione conclusiva dei risultati ottenuti.

1.1 Descrizione delle caratteristiche della linea

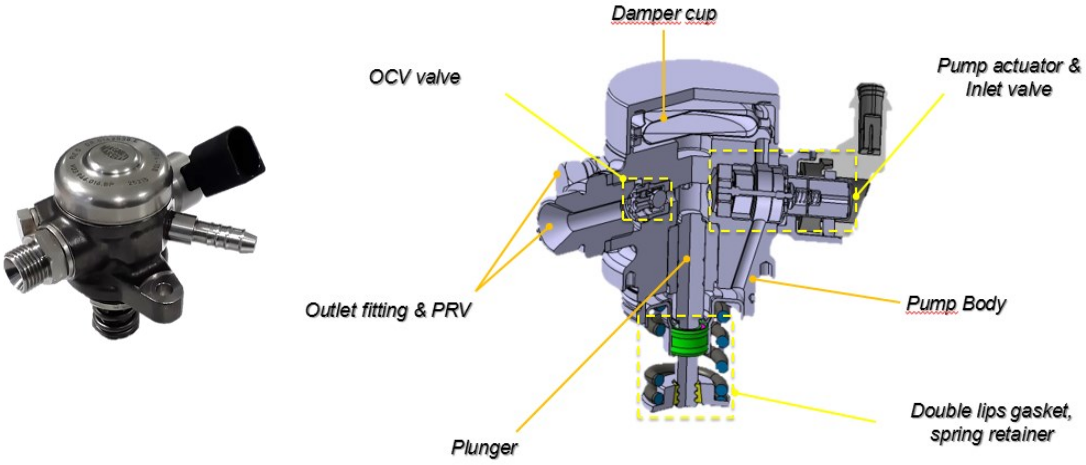
I dati che verranno in seguito analizzati sono relativi agli eventi di guasto di una linea di assemblaggio dell'azienda Magneti Marelli. Si tratta di una linea completamente automatica, costituita da una serie di operations (OP), ognuna delle quali caratterizzata da uno specifico tempo ciclo. La seguente immagine mostra il layout, la sequenza delle operations ed il rispettivo tempo ciclo.



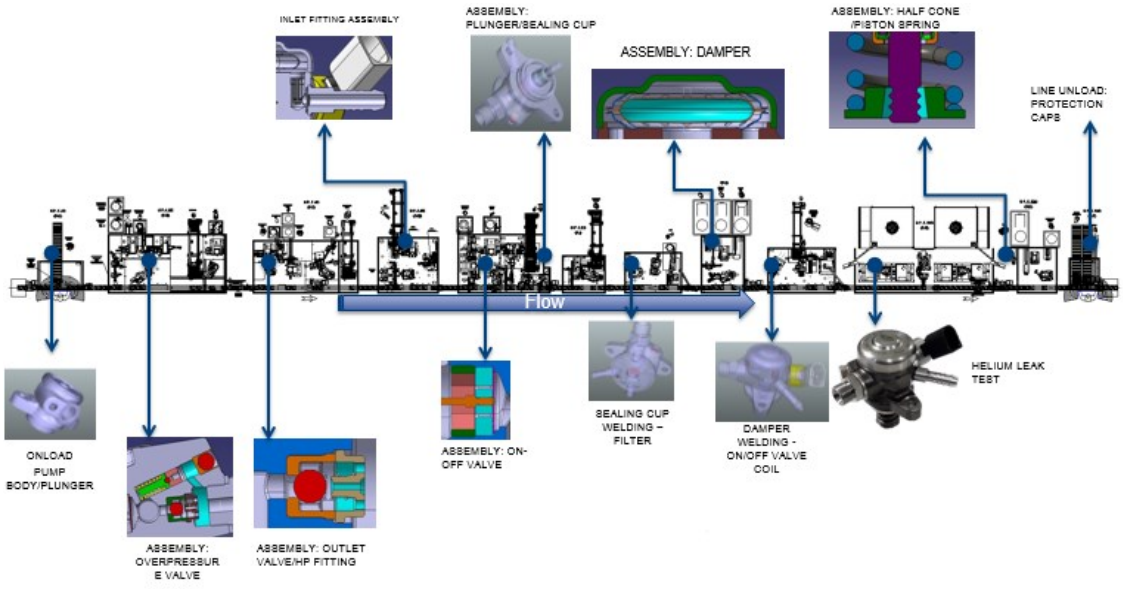
TC PER OPERAZIONE:

OP 30.1: 15.16 s
OP 30.2: 14.9 s
OP 40.1: 17.24 s
OP 40.2: 15.30 s
OP 50: 15.75 s
OP 60.1: 15.61 s
OP 70: 15.25 s
OP 80: 14.64 s
OP 90: 14.38 s
OP 100: 16.52 s
OP 110: 16.76 s
OP 120: 15.82 s

Il collo di bottiglia è rappresentato dall'OP.110, che ha un tempo ciclo prossimo ai 17s. Il processo di assemblaggio viene eseguito su un unico prodotto, la Pompa PHP4, di cui è riportata sia l'immagine completa, sia la sua sezione che ne esplicita la struttura.



Il grafico sottostante riporta il workflow della linea di assemblaggio.



Le singole attività svolte in ogni OP sono utili per contestualizzare gli eventi di guasto ed è quindi opportuno schematizzarle:

- OP 10: l'input è costituito dai pezzi della pompa, si esegue l'assemblaggio tra pompa e plunger. Presenza di sensori di controllo ottico per verificare se è possibile proseguire;
- OP 30.1: si esegue l'accoppiamento della molla, la verifica del corretto assemblaggio ed il processo di calibrazione;
- OP 30.2: inserimento della valvola di aspirazione ed altri componenti acquistati, poi si scarica il pezzo;
- OP 40.1: prelievo della pompa da parte del robot, dopo aver verificato che la posizione della pompa sia corretta;
- OP 40.2: attraverso sensori ottici si verifica la presenza dei pezzi, si esegue l'accoppiamento e l'avvitamento;
- OP 50: si esegue il piantaggio, la saldatura, il controllo della saldatura tramite sensore ottico;
- OP 60: controllo della valvola di aspirazione e verifica, per mezzo di una foto, del corretto posizionamento per valutare un eventuale intervento di un

operatore. L'attività di controllo si articola in diverse fasi che vanno dal posizionamento dello spillo nella pompa, presenza della molla, verifica della corretta posizione di saldatura e controllo della saldatura. Contestualmente all'attività di controllo si esegue anche la cianfrinatura.

- OP 70: prelievo dei pezzi dall'OP 10 ed esecuzione delle letture sulla pompa relativamente al plunger ed ai data mart della pompa. Si verifica che il plunger entri nella pompa e si esegue il piantaggio del sealing cup;
- OP 80: saldatura;
- OP 90: inserimento dumper spacer + dumper cushion + damper spacer. Un tastatore controlla la presenza dei tre componenti e si verifica la quota;
- OP 100: prelievo della pompa dal pick and place + stazione di saldatura damper + piantaggio e saldatura bobina;
- OP 110: verifica delle saldature e della tenuta della pompa;
- Op 120: controllo della quota dei semiconi;

1.2 Descrizione dei dati forniti

I dati forniti sono contenuti in due fogli Excel, “Data Collector e “Data guasti”. Il primo foglio tra le varie voci riporta, per ogni giorno e per ognuno dei tre turni giornalieri, il valore percentuale dell’OEE. L’OEE (Overall Equipment Effectiveness) è un indicatore globale di efficienza delle risorse produttive ed è ricavato a partire dai valori percentuali di Disponibilità, Performance e Qualità, anch’essi presenti nel file.

$$OEE = Disponibilità * Performance * Qualità$$

La *Disponibilità* esprime il tempo effettivamente disponibile rispetto al tempo operativo pianificato di potenziale funzionamento della linea.
Dal tempo teorico (espresso in minuti) si sottraggono quindi le availability losses che, nel caso specifico, consistono principalmente in perdite in termini di tempo per la risoluzione dei guasti, manutenzione programmata e setup.

$$Disponibilità = \frac{tempo\ operativo\ effettivo}{tempo\ operativo\ teorico}$$

La *Performance* è definita dal rapporto tra il tempo operativo netto ed il tempo operativo effettivo. Il tempo operativo netto è ottenuto sottraendo dal tempo operativo effettivo le performance losses, ovvero le perdite legate a micro fermate, variazione del tempo ciclo rispetto allo standard, mancanza materiali diretti e mancanza di alimentazione da postazione a monte/valle.

$$Performance = \frac{tempo\ operativo\ netto}{tempo\ operativo\ effettivo}$$

La *Qualità* è espressa come il rapporto tra il tempo operativo a valore aggiunto ed il tempo operativo netto (definito nel passaggio precedente). Il tempo operativo a valore aggiunto è ricavato sottraendo dal tempo operativo netto i minuti delle quality losses, ovvero i minuti persi a causa di fermo per problemi di qualità, minuti di produzione scarto e rework.

$$Qualità = \frac{tempo\ operativo\ a\ valore\ aggiunto}{tempo\ operativo\ netto}$$

Dato che la maggior parte dei dati e dei valori contenuti nel foglio Data Collector è inglobata nel calcolo dell'OEE, al fine dell'analisi si prenderà in considerazione tale indicatore.

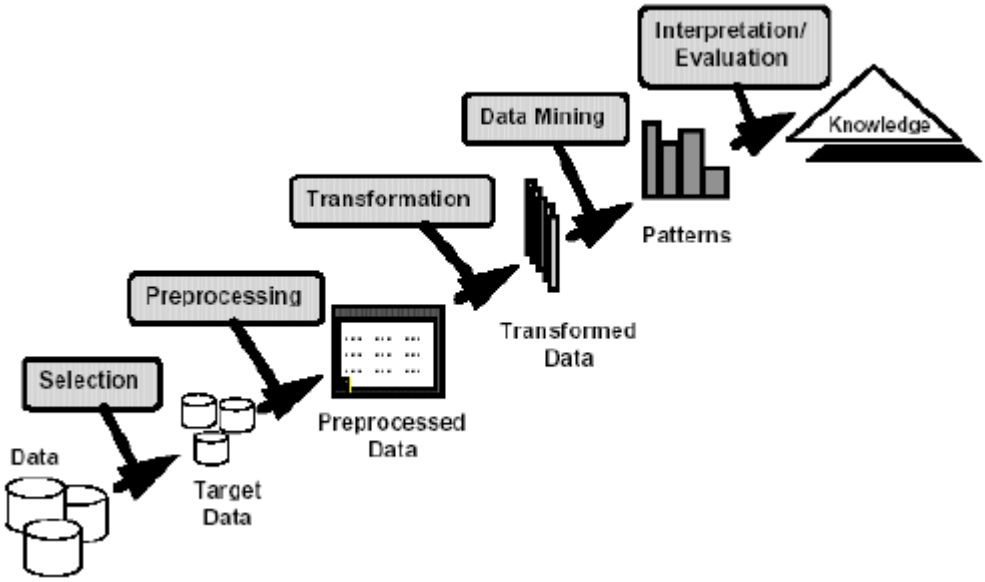
Il secondo foglio, Data Guasti, riporta invece tutti gli eventi di guasto che si manifestano nello specifico giorno e turno. Accanto alla tipologia di guasto viene indicata anche la rispettiva OP ed il tempo, espresso in minuti, impiegato per la sua risoluzione.

L'arco temporale preso in esame va da gennaio 2019 fino a giugno 2020 e comprende un totale di 877 istanze.

L'obiettivo finale dell'analisi sta nel trovare una correlazione tra i dati, sia tra i guasti e l'OEE, sia tra i vari eventi di guasto. Per ottenere questo risultato si ricorrerà a delle tecniche di Data Mining, le cui metodologie verranno introdotte dal capitolo seguente.

2. Il Data Mining

Il concetto di Data Mining è inquadrato all'interno del processo di Knowledge Discovery in Databases (KDD), ovvero "*un processo di identificazione di pattern validi, nuovi, potenzialmente utili, comprensibili da un ampio volume di dati*"[1]. Il Data Mining rappresenta uno dei cinque steps di cui si compone il KDD. L'intero processo di Knowledge Discovery in Databases è riassunto nello schema sottostante.



È utile definire e distinguere le varie fasi del KDD includendo anche quella di Data Mining:

- *Data selection*: effettuata a partire dalla comprensione del dominio applicativo su cui si intende operare, cioè una collezione di dati rilevanti che contengono informazioni relative all'area oggetto di interesse, definendo così l'ambito della raccolta dati. L'obiettivo di questa fase è di individuare le sorgenti di dati disponibili ed estrarne un campione per l'analisi preliminare. Tali dati sono spesso contenuti nei Data Warehouse, cioè delle basi di dati utilizzate per il supporto alle decisioni. In altri casi invece si trovano

[1] Fayyad, 1996

all'interno di databases o in specifici file. Insieme ai dati è necessario acquisire anche i metadati, ovvero delle informazioni che includono la definizione dei dati, descrizione dei tipi e dei valori potenziali.

- *Data preprocessing*: consiste nell'effettuare una pre-elaborazione dei dati al fine di renderli strutturati e di evitare quindi che dati di scarsa qualità vadano ad influire negativamente sul processo di Data Mining. Ciò si rende necessario dal momento che i dati tendono ad essere soggetti a rumore, omissioni ed inconsistenza, sia per la quantità, sia a causa della loro origine eterogenea. Per questo motivo il preprocessing esegue una serie di operazioni sui dati:
  - *Pulizia (data cleaning)*: attività mirata a garantire un sufficiente livello di qualità dei dati eliminando quelli ridondanti, compilando i valori omessi, identificando le anomalie.
  - *Integrazione (data integration)*: si uniscono i dati provenienti da più fonti in modo da ridurre le inconsistenze e le ridondanze derivanti dal fatto che potrebbero sussistere differenti descrizioni dello stesso attributo in più database.
  - *Riduzione (data reduction)*: mirata a comprimere i dati in un volume più ristretto, mantenendo però lo stesso contenuto.

- *Data trasformation*: consiste prevalentemente nella risoluzione delle eterogeneità dei dati. Tra le strategie di trasformazione utilizzate è possibile elencare l'aggregazione, la normalizzazione e la discretizzazione. In tal modo i dati vengono predisposti all'uso operativo.

- *Data Mining*: si definisce come l'attività di scoperta ed interpretazione di relazioni, pattern e regolarità nascoste in una grande mole di dati con lo scopo di ottenere un maggior livello di informazione e conoscenza sia sugli oggetti del database/dataset che si sta analizzando, sia sul mondo reale

(qualora il database ne sia una sua rappresentazione). È possibile quindi definire il Data Mining come il cuore del processo di KDD.

- *Valutazione ed interpretazione dei pattern*: permette di trarre delle implicazioni applicative dai pattern individuati. È necessario però tenere conto che non tutti i pattern ricavati attraverso il Data Mining sono rilevanti per l'utente. Per distinguere quelli utili si ricorre a delle misurazioni oggettive basate sulla struttura del pattern estratto e sulla statistica che lo sostiene. Una volta "filtrati", i pattern vengono documentati per un futuro utilizzo.

Implementazione: consiste nell'utilizzo dei risultati. I pattern scoperti, conseguentemente alla loro interpretazione, vengono utilizzati per prendere decisioni operative nel mondo reale in modo da validare la scoperta di conoscenza. Il successo di quest'ultima fase determina l'efficacia dell'intero processo di KDD.

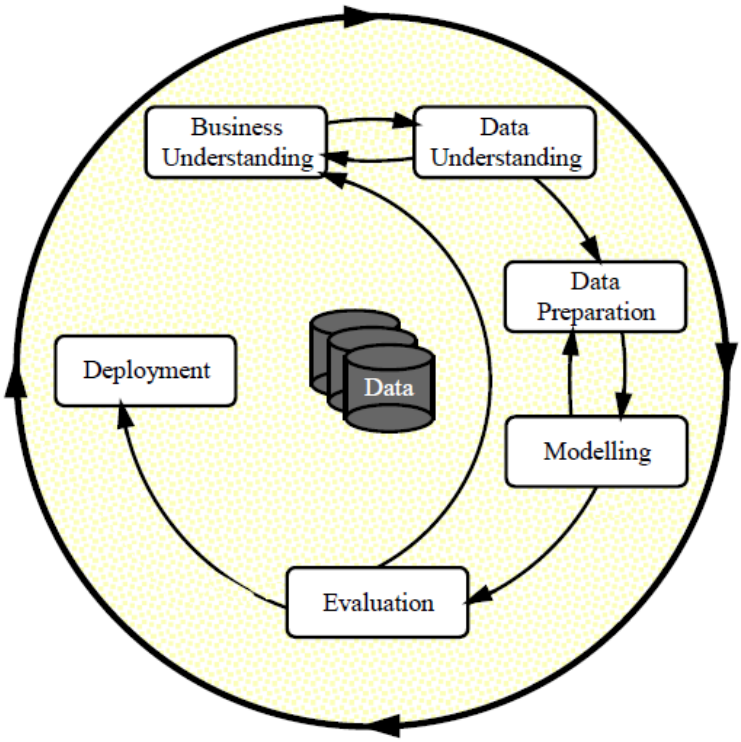
Come si evince da quanto appena descritto, il principale obiettivo del KDD è quello di estrarre conoscenza di alto livello a partire da informazioni di basso livello e quindi processare in modo automatico grandi moli di dati identificando i pattern più significativi.
Con il termine "pattern" si intende una struttura, un modello ricorrente nei dati che deve avere come caratteristiche principali quello di essere:

- *Valido* sui dati con un certo grado di confidenza;
- *Comprensibile* dal punto di vista sintattico e semantico affinché possa essere interpretato dall'utente;
- *Potenzialmente utile*, cioè in grado di supportare l'utente del processo decisionale;

In questo contesto il Data Mining è responsabile di estrarre tali pattern e generare modelli a partire dall'output fornito dalla fase di data preprocessing. Proprio per questo, il data preprocessing può essere considerato come le fondamenta, mentre

il Data Mining rappresenta il pilastro del processo di Knowledge Discovery in Databases.

Ponendo ora l'attenzione sul Data Mining, è possibile andare a definire tutti passi fondamentali utilizzando un approccio metodologico che prende il nome di metodologia CRISP-DM. L'acronimo CRISP-DM sta per Cross Industry Standard Process for Data Mining e definisce un modello di processo che fornisce una struttura finalizzata a realizzare progetti di Data Mining indipendentemente da quale settore industriale e quali tecnologie si stanno prendendo in considerazione. Ciò si è reso necessario dal momento che il Data Mining ha bisogno di un approccio standardizzato che renda possibile trasformare dei business problems in dei Data Mining task, applicando opportune trasformazioni nei dati e fornendo dei mezzi per validare l'efficacia dei risultati.
Il ciclo di vita del progetto di Data Mining si struttura in sei fasi ed è mostrato di seguito.



Non vi è tuttavia una sequenza rigorosa tra le fasi. Le frecce hanno solamente la funzione di mostrare le principali e più frequenti dipendenze tra le varie fasi ed, in uno specifico progetto, la scelta relativa alla fase che deve seguire dipende esclusivamente dall'esito della fase precedente.
Le sei fasi della metodologia CRISP-DM sono descritte di seguito:

- *Business understanding*: consiste nel comprendere gli obiettivi ed i requisiti di progetto, convertirli in un problema di Data Mining ed eseguire un piano di progetto preliminare finalizzato al raggiungimento di tali obiettivi.

- *Data understanding*: si basa su una raccolta preliminare dei dati e prosegue con l'identificazione dei problemi di qualità nei dati. Vengono svolte delle analisi sui dati con lo scopo di individuare caratteristiche nascoste. Business understanding e Data understanding sono strettamente collegati perché la formulazione di un problema di Data Mining richiede come prerequisito la comprensione dei dati disponibili.

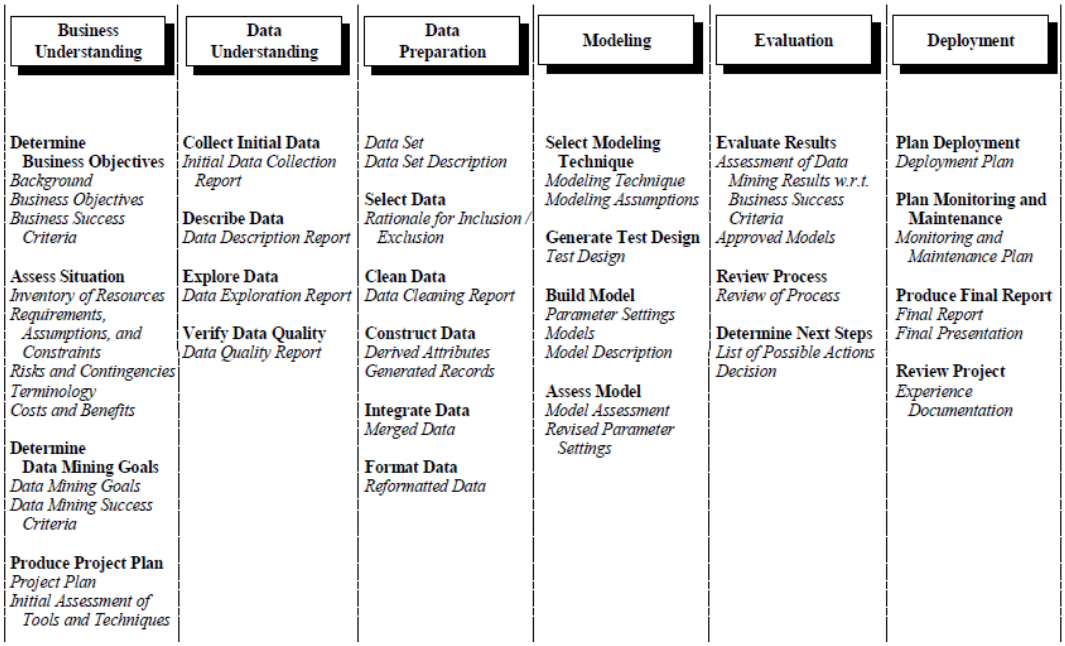
- *Data preparation*: include le attività necessarie a costruire il dataset a partire dai dati grezzi. Consiste nella selezione degli attributi, nella definizione di nuovi, nella pulizia e nella trasformazione dei dati.

- *Modelling*: vengono selezionate ed applicate al dataset diverse tecniche di Data Mining. Alcune di queste tecniche richiedono uno specifico formato di dati. Tra le varie tecniche si sceglie quella che garantisce un maggior grado di accuratezza.

- *Evaluation (valutazione)*: effettua la valutazione del modello realizzato rivisitando tutti i passaggi seguiti della sua costruzione per essere certi di aver raggiunto l'obiettivo. In fase conclusiva deve essere presa una decisione relativamente a quale utilizzo deve essere destinato il risultato ottenuto.

- *Deployement (utilizzo)*: il modello creato è pronto per essere utilizzato. È quindi importante ricorrere ad un piano di utilizzo e mantenimento di questo modello. Spesso non è il data analyst, bensì l'utente finale che deve portare avanti questa fase, che a seconda dei requisiti può risultare semplice o complessa.

Nella tabella sottostante si trovano collocate e schematizzate le fasi sopra citate con i aggiunta le rispettive sotto-fasi:



Business Understanding	Data Understanding	Data Preparation	Modeling	Evaluation	Deployment
Determine Business Objectives *Background* *Business Objectives* *Business Success Criteria* **Assess Situation** *Inventory of Resources* *Requirements, Assumptions, and Constraints* *Risks and Contingencies* *Terminology* *Costs and Benefits* **Determine Data Mining Goals** *Data Mining Goals* *Data Mining Success Criteria* **Produce Project Plan** *Project Plan* *Initial Assessment of Tools and Techniques*	**Collect Initial Data** *Initial Data Collection Report* **Describe Data** *Data Description Report* **Explore Data** *Data Exploration Report* **Verify Data Quality** *Data Quality Report*	*Data Set* *Data Set Description* **Select Data** *Rationale for Inclusion / Exclusion* **Clean Data** *Data Cleaning Report* **Construct Data** *Derived Attributes* *Generated Records* **Integrate Data** *Merged Data* **Format Data** *Reformatted Data*	**Select Modeling Technique** *Modeling Technique* *Modeling Assumptions* **Generate Test Design** *Test Design* **Build Model** *Parameter Settings* *Models* *Model Description* **Assess Model** *Model Assessment* *Revised Parameter Settings*	**Evaluate Results** *Assessment of Data Mining Results w.r.t. Business Success Criteria* *Approved Models* **Review Process** *Review of Process* **Determine Next Steps** *List of Possible Actions* *Decision*	**Plan Deployment** *Deployment Plan* **Plan Monitoring and Maintenance** *Monitoring and Maintenance Plan* **Produce Final Report** *Final Report* *Final Presentation* **Review Project** *Experience Documentation*

La descrizione, appena proposta, delle fasi della metodologia tornerà utile nel momento dell'applicazione delle tecniche di Data Mining ai dati forniti, che verrà esplicitata nel capitolo 4.

Le tecniche di Data Mining possono essere applicate ad un vasto numero di processi decisionali. Le maggiori aree di interesse sono:

- Marketing: analisi del comportamento dei consumatori basati sul modello di acquisto dei prodotti, determinazione delle strategie di mercato, segmentazione e fidelizzazione dei clienti.

- Finanza: analisi delle prestazioni degli investimenti e valutazione di opzioni finanziarie.
- Analisi dei rischi (risk analysis)
- Produzione/assemblaggio: finalizzato all'ottimizzazione delle risorse, come macchine, materiali, forza lavoro, oppure alla progettazione ottimale di processi di produzione/assemblaggio e struttura di impianti produttivi.
- Salute: analisi dell'efficacia di determinati trattamenti.

In particolare l'applicazione di queste tecniche nell'ambito produttivo si è resa indispensabile dal momento che, a partire dallo scorso decennio, le aziende si sono trovate davanti ad un incremento della domanda a causa delle richieste frequenti di prodotti sempre più personalizzati. La diretta conseguenza è stata quella di un aumento della complessità, poiché gli stessi prodotti venivano precedentemente realizzati secondo la logica della produzione di massa. La complessità di produzione può essere distinta in complessità esterna ed interna. Quella esterna è legata all'aumento della domanda di mercato ed è gestibile ottimizzando l'efficienza dei processi della supply chain, mentre quella interna è determinata dalla varietà dei prodotti e può essere diminuita tenendo sotto controllo tale varietà.

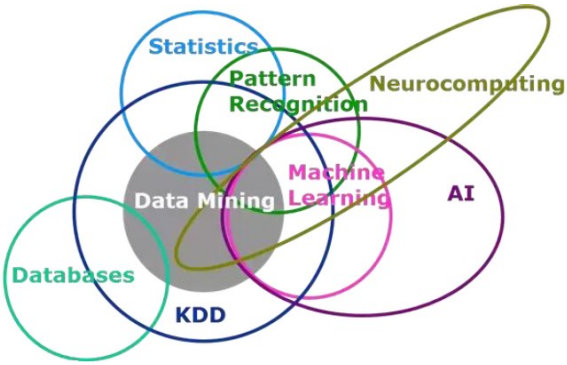
A questa trasformazione così repentina si è aggiunto anche l'aumento della comunicazione tra uomini e macchinari nell'ambiente dell'officina e l'avanzare del processo di cambiamento noto con il nome di Industry 4.0.

Tutto ciò ha portato all'incremento dei dati di produzione da cui risultava possibile estrarre delle relazioni e delle interdipendenze con l'obiettivo di raggiungere un maggior livello di conoscenza e migliorare i processi. Ciò ha portato allo sviluppo di discipline come il Data Mining.

2.1 Interdisciplinarità del Data Mining

Il Data Mining non è però un concetto a sé stante, ma è un processo strettamente interconnesso con altri ambiti dell'informatica come l'intelligenza artificiale ed il Machine Learning e di altre scienze legate alla matematica quali la statistica, e si sviluppa dalla loro integrazione.

Per descrivere l'entità di questa interdisciplinarità è possibile ricorrere al seguente grafico.



L'uso dell'Intelligenza Artificiale (AI) e in modo più specifico del Machine Learning (ML) e del Data Mining (DM), permette di ricavare dai dati grezzi la conoscenza utile per una moltitudine di applicazioni.

Per spiegare l'interconnessione esistente tra le discipline è opportuno darne prima di tutto una definizione generale che ne esplicita la natura e gli obiettivi.

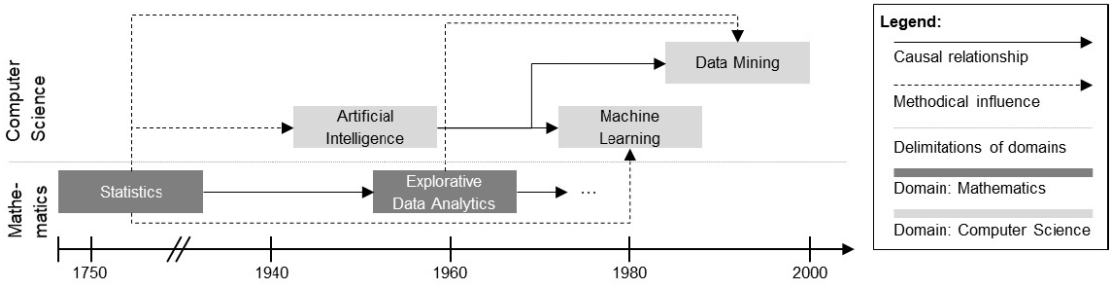
L'Intelligenza artificiale (AI) è un campo della scienza emerso a partire da altri ambiti come l'informatica, la matematica, le neuroscienze ed altre discipline scientifiche. Da un punto di vista scientifico l'obiettivo dell'AI è quello di comprendere i principi della rappresentazione della conoscenza, mentre da un punto di vista ingegneristico è quello di creare degli agenti computazionali in grado di risolvere problemi del mondo reale allo stesso modo oppure in modo migliore rispetto all'uomo.

Il Machine Learning rappresenta invece un sottodominio dell'AI che ha la funzione di rendere gli agenti computazionali abili di acquisire conoscenze relative ad un task e di risolverne specifici problemi. Questo apprendimento avviene per mezzo di dati storici. Il ML offre strumenti utili per l'AI e rappresenta la base per altri sottodomini. Il Data Mining costituisce un ulteriore sottodominio in cui vengono applicate metodologie di Machine Learning e metodi statistici al fine di effettuare il pre-processing e l'analisi dei dati.

La differenza sostanziale risiede nel fatto che nel Machine Learning la conoscenza viene immagazzinata implicitamente con lo scopo di ottimizzare le prestazioni degli agenti computazionali, mentre nel Data Mining, i metodi di ML sono impiegati in modo che prima venga acquisita conoscenza dai dati ed in un secondo momento venga memorizzata e visualizzata esplicitamente, rendendola accessibile e interpretabile per l'utente finale.

In questo contesto si inserisce anche la statistica, una scienza formale inclusa nell'ambito della matematica e che costituisce un supporto al Data Mining. In particolare la statistica mira ad acquisire informazioni quantitative dalle osservazioni sia per trarre conclusioni dai dati, sia per descrivere i dati ed ha un ruolo attivo nel Data Mining, dal momento che questo ultimo si affida a metodi statistici.

Di seguito è riportato in figura anche lo sviluppo storico delle varie discipline insieme al tipo di relazione che vi sussiste. Il grafico è stato realizzato suddividendo i due ambiti scientifici, quello matematico e quello informatico.



Lo sviluppo si è articolato fondamentalmente in tre momenti, corrispondenti a tre specifici obiettivi. Il primo di essi consiste nello sviluppo di strumenti in grado di imitare il modo di agire e di pensare dell'uomo. Già nel 1950 Turing aveva dato una prima definizione del concetto che solo in seguito assumerà il nome di Intelligenza Artificiale. Si era infatti diffusa l'ipotesi secondo la quale, attraverso una conoscenza espressa sotto forma di connessioni logiche e ragionamento automatico, si potesse raggiungere un livello di intelligenza paragonabile a quello umano.

Il secondo obiettivo è correlato allo sviluppo di strumenti per risolvere specifici problemi. In quest'ottica nel 1987 Rumelhart e McClelland teorizzarono che un computer fosse in grado di apprendere delle regole osservando le connessioni tra i dati. Ciò aumentò l'interesse nel campo del Machine Learning. Collegando infatti delle diverse unità di calcolo era possibile creare una nuova architettura flessibile ed allo stesso tempo robusta costituendo una rete neurale. Oltre alla rete neurale, si aggiunsero anche ulteriori metodi di machine learning come i kernel methods (ad esempio il support vector machines) e metodi gerarchici come ad esempio i decision trees.

Il terzo ed ultimo obiettivo concerne lo sviluppo di strumenti volti ad identificare e spiegare i patterns ricavati dai dati. In seguito all'aumento esponenziale della quantità dei dati emersa a partire dalla fine degli anni 90, emerge una nuova disciplina, il Data Mining, sviluppata unendo molteplici campi tra cui l'intelligenza artificiale, il Machine Leraning e la statistica.

2.2 I task del Data Mining

I task, cioè le attività tipiche del Data Mining, permettono di distinguere due tipi di modelli di conoscenza:

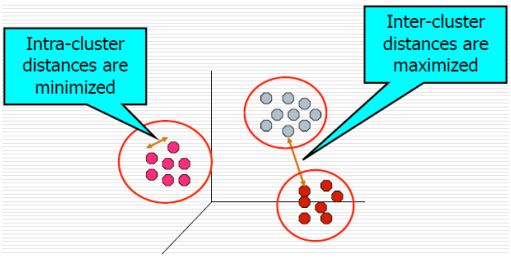
- *Modelli descrittivi*: trovano dei pattern che descrivono i dati. Tali pattern devono essere interpretabili dall'uomo. La conoscenza è rappresentata sotto forma di modello che rispecchia i dati e le relazioni tra i dati.
- *Modelli predittivi*: sfruttano alcune variabili e quindi una qualche caratteristica dei dati per predire il valore non noto o futuro di altre variabili. La conoscenza è rappresentata dalla predizione di condizioni future, relazioni e trend.

Tra i modelli descrittivi è possibile elencare:

- Ricerca di regole di associazione (association rules)
- Clustering

Le *regole di associazione* sono dei pattern del tipo X→Y in cui X rappresenta la premessa ed Y la conseguenza e costituiscono quindi delle dichiarazioni relative ai valori di determinati attributi corrispondenti a specifici items.

Il *Clustering* (detto anche segmentazione), ha come scopo quello di suddividere un certo numero osservazioni in gruppi chiamati cluster. Gli elementi dello stesso cluster devono avere le stesse caratteristiche e lo stesso comportamento di un attributo individuato come riferimento. L'organizzazione dei cluster è tale per cui possono essere sovrapposti ed organizzati in modo gerarchico. L'esempio grafico sottostante ne permette l'immediata comprensione.



Tra i modelli predittivi si trovano:

- Classificazione
- Regressione
- Individuazione di anomalie

I problemi di *classificazione* hanno la funzione di assegnare ad una serie di osservazioni un insieme di classi predefinite.

Si presuppone l'esistenza di un classificatore, un algoritmo che, a partire da un insieme di records (training set) in cui ogni record è costituito da un certo numero di attributi, determina un modello per ciascun attributo della classe. Il modello deve essere in grado di indicare il valore dell'attributo in funzione dei valori di altri attributi presenti nel training set. Solitamente come classificatori si utilizzano gli alberi di decisione. I decision trees verranno trattati ampiamente nel capitolo successivo e applicati in un secondo momento ai dati forniti.

La restante parte in cui è stato suddiviso il dataset (test set) è utilizzata par la validazione e quindi per assegnare, in modo per quanto possibile accurato, i record non noti ad una specifica classe. Dal test set si ricava l'accuratezza del modello.

I problemi di classificazione si dividono principalmente in due tipologie a seconda del numero di classi coinvolte nel problema.

La prima tipologia è quella dei problemi di classificazione binaria che ricorrono solamente a due classi. La seconda invece, è costituita dai problemi multi-classe, i quali prendono in esame un numero di classi maggiore di due. Questi ultimi risultano più complessi di quelli di natura binaria.

Dal punto di vista implementativo, una sostanziale differenza risiede nel fatto che l'errore di classificazione aumenta all'aumentare del numero delle classi perché il classificatore deve riuscire a partizionare i dati in base ad un più elevato numero di classi. Una dimostrazione concreta di ciò è presente nell'analisi che verrà proposta nel capitolo 4.

Per calcolare l'errore di classificazione si ricorre ad una particolare matrice, chiamata Matrice di confusione (Confusion matrix).

Tale matrice ha quindi la funzione di valutare quanto un classificatore riesce ad effettuare una classificazione accurata. Considerando la situazione in cui le classi sono due, essa assume la seguente forma:

	CLASSE STIMATA	
CLASSE VERA	T1	M1
	M2	T2

- T1: istanze di classe vera 1 e classe stimata 1;
- T2: istanze di classe vera 2 e classe stimata 2;
- M1: istanze di classe vera 1 e classe stimata 2;
- M2: istanze di classe vera 2 e classe stimata 1;

Con la lettera T vengono indicate le classificazioni correttamente eseguite, cioè quelle per le quali la classe vera corrisponde con la classe stimata. Con M si indicano invece le mis-classificazioni, presenti quando tale corrispondenza non sussiste. Dividendo la somma delle istanze per cui si verifica una errata classificazione per la totalità delle istanze appartenenti ad entrambe le classi, si ricava l'errore di classificazione calcolato in base alla seguente formula:

$$e = \frac{M1 + M2}{N1 + N2} = \frac{\#\ volte\ in\ cui\ si\ sbaglia\ la\ stima}{\#\ istamze\ totali}$$

Dalla stessa matrice è possibile quantificare anche l'accuratezza, calcolata in base alla formula:

$$Accuracy = \frac{T1 + T2}{T1 + T2 + M1 + M2}$$

Una volta elencate e descritte le varie tipologie di problemi è opportuno fare una valutazione anche dal punto di vista implementativo, inquadrando tali problemi nei vari ambiti del Machine Learning e definendo a quale categoria appartengono.
Il Machine Learning, basandosi sul concetto di inferenza, cioè di deduzione, riesce ad estrarre la soluzione del problema tramite un processo di apprendimento a partire dai singoli dati. L'operato del Machine Learning può essere suddiviso in due fasi: la prima relativa alla scoperta all'interno di un sistema di dipendenze non note, mentre la seconda consiste invece nella stima di queste dipendenze in modo da riuscire a predire dei nuovi output.
I Machine Learning tasks vengono infatti usualmente classificati in tre categorie:

- *Supervised learning*
- *Unsupervised learning*
- *Reinforcement learning*

Nel *supervised learning* il sistema apprende in modo induttivo una funzione. Tale funzione serve a predire il valore di una variabile (variabile dipendente) a partire da un insieme di variabili (variabili indipendenti). Le istanze rappresentano tutti i possibili valori della funzione in cui ognuna di esse è descritta attraverso un insieme di attributi. Si definiscono inoltre dei dati di training che sono costituiti da un sottoinsieme di istanze di cui si conosce la variabile di output.
La denominazione di "supervised learning" (apprendimento supervisionato) è imputabile al fatto che ai dati appartenenti al training set vengono preventivamente associate delle etichette (labels) che riconducono alla classe di appartenenza, mentre ai dati che restano fuori dal training set, cioè la parte rimanente delle osservazioni/esempi/record vengono assegnate le etichette delle varie classi sulla base del modello costruito a partire dal training set.
Un tipico esempio di supervised learning è costituito dai problemi di classificazione e regressione: la differenza sta nel fatto che la classificazione ha il compito di predire classi predeterminate e distinte, mentre la regressione predice valori

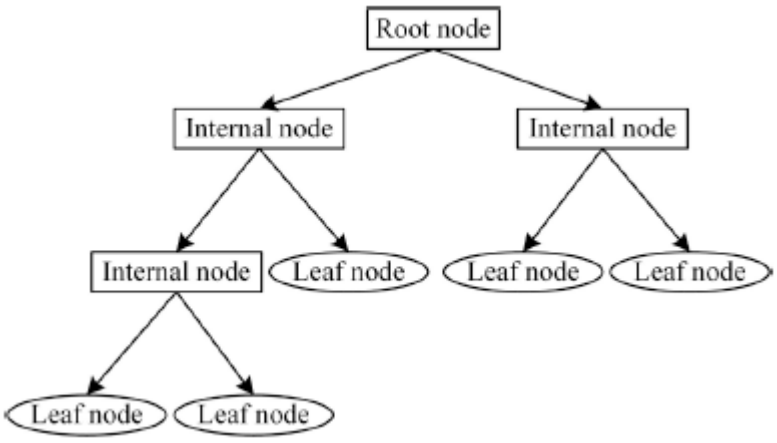
numerici. Le metodologie di supervised learning che si trovano più frequentemente sono gli alberi di decisione, il Random Forest (RF), multiple Linear Regression (MLR), Logistic Regression(LR), l'Istance Based Learning (IBL) che include il Genetic Algorithms, Artificial Neural Networks ANN), bayesian Network (BN), Support Vector Machine (SVM).

Nell'*unsupervised learning* il sistema di apprendimento cerca invece di ricavare associazioni tra le variabili oppure una struttura nascosta nei dati. La differenza tra il supervised e l'unsupervised learning risiede nel fatto che in questo ultimo i dati di training sono costituite da istanze prive dell'etichetta che le identifica.
In concreto, dal momento che non esiste nessuna etichetta da cui la macchina ha la possibilità di apprendere, è il modello stesso che deve scoprire i pattern dal dataset in input attraverso determinate regole o associazioni.
Appartengono all'unsupervised learning problemi quali l'Association Rule Mining, cioè la ricerca di regole di associazione ed il Clustering. Nel caso del clustering ad esempio l'etichetta della classe è sconosciuta e la segmentazione finalizzata alla creazione di cluster/gruppi omogenei viene effettuata senza nessun tipo di conoscenza pregressa, da cui il nome di apprendimento non supervisionato.

Nel *reinforcement learning* invece, il sistema interagisce con un ambiente dinamico, innescando il processo di apprendimento.
Il sistema non conosce a priori il comportamento dell'ambiente e di conseguenza l'apprendimento è ottenuto a partire da una serie di tentativi ed errori.
Questo tipo di apprendimento è solitamente utilizzato nei sistemi autonomi.

2.3 Alberi di decisione: metodologia

Gli alberi di decisione rappresentano uno dei principali algoritmi di apprendimento induttivo utilizzati nell'ambito dei problemi di classificazione o regressione. Sono contraddistinti da una struttura gerarchica generata ricorrendo ad un approccio di tipo top-down e presentano tre tipologie di nodi: il nodo radice, i nodi interni e le foglie, come evidenziato dall'immagine sottostante.



All'interno della struttura i nodi e gli archi hanno uno specifico ruolo:

- Ciascun nodo interno rappresenta una variabile/attributo
- Ciascun ramo uscente da un nodo corrisponde ad uno dei possibili valori che l'attributo/variabile può assumere
- Ogni foglia assegna una classificazione

Il nodo radice è il quello da cui si diramano tutti i nodi successivi attraverso un procedimento ricorsivo che consente di arrivare ad una foglia e quindi ad effettuare una classificazione. Un determinato nodo è indicato come padre rispetto ai nodi da esso generati, mentre può essere chiamato figlio rispetto al nodo da cui discende e che quindi lo ha generato.

Se nel momento in cui si arriva ad una foglia avviene l'assegnazione di un'etichetta corrispondente ad una classe (class label), si parla di alberi di classificazione,

altrimenti se l'assegnazione rimanda ad un valore numerico della variabile e l'albero viene utilizzato per predire variabili di tipo quantitativo, si tratta di alberi di regressione. In questa trattazione ai fini applicativi si ricorrerà, nel capitolo 4, agli alberi di classificazione.

La procedura generale per effettuare una classificazione di un'istanza, generando un albero di decisione, consiste nel partire dal nodo radice e, a seconda del risultato del test effettuato su un determinato attributo, seguire un ramo piuttosto che un altro, fino a raggiungere la foglia che mostra l'etichetta relativa alla classificazione effettuata. Più in dettaglio, i passaggi sottesi a tale processo sono principalmente tre:

- Per ciascun nodo selezionare una regola di splitting. In questo modo si determinano le variabili ed il loro valore soglia che serviranno per effettuare la partizione del dataset.
- Per ogni nodo stabilire quando è opportuno procedere splittando, oppure quando conviene fermarsi e considerare il nodo come terminale.
- Assegnare un'etichetta a ciascun nodo terminale individuando una regola per l'assegnazione della classe ad ogni foglia.

Il primo passaggio, ovvero l'individuazione di un criterio di splitting, si basa sul calcolo di un indice di natura statistica finalizzato a determinare la migliore partizione in corrispondenza di ogni predittore. A tal fine ad ogni passaggio successivo si vuole ridurre l'eterogeneità rispetto al passaggio precedente ricorrendo ad uno dei seguenti criteri di split che consentono di selezionare l'attributo ottimo ai fini della classificazione:

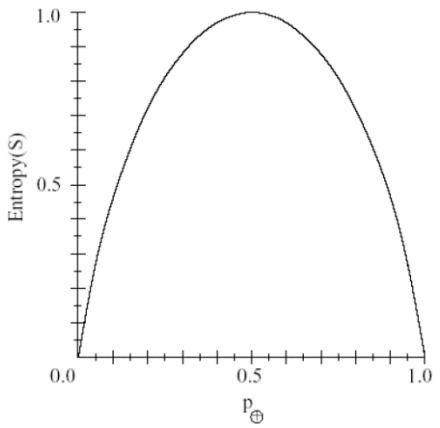
$$Entropia = -\sum_{j=1}^{C} p(j|t) log_2\, p(j|t)$$

j= 1…C rappresenta le classi

p(j|t) è la frequenza relativa della classe j al nodo t

Il grafico seguente mostra l'evolvere dell'entropia nel caso in cui la classificazione avvenga esclusivamente su due classi. Di conseguenza i valori che essa può assumere sono inclusi tra 0 e 1. $0 \leq H(S) \leq 1$

L'Entropia H(S) ha quindi il ruolo di misurare l'impurità dei records/attributi utilizzati per generare l'albero.



Se $H(S) = 0$ significa che tutti i records appartengono ad una sola delle due classi. Ciò implica di conseguenza maggiore informazione.

Se $H(S) = 1$ significa che i records vengono equamente distribuiti tra entrambe le classi;

Nel caso generale, cioè quando le classi sono più di due, i valori minimi e massimi che tale indice può assumere sono rispettivamente 0 e $log\ n_C$.

Maggiore risulta l'entropia, maggiore è la disomogeneità che contraddistingue i dati ed è più difficile assegnare la propria classe ad ognuno dei records.

All'entropia si collega un diverso indice, l'*Information Gain*, ovvero il guadagno d'informazione, che rappresenta la riduzione di entropia che si ha effettuando lo splitting. Più nello specifico l'Information Gain, a partire dal concetto di Entropia, opera misurando l'impurità di ogni split e seleziona l'attributo che riduce maggiormente l'Entropia.

In alternativa è possibile ricorrere ad un diverso indice, il Gini Index, calcolato come:

$$1 - \sum_{j=1}^{C} p(j|t)^2$$

In cui p(j|t) è la frequenza relativa della classe j al nodo t.

Considerando che la classificazione venga effettuata in generico numero di classi pari a C, i valori che il Gini Index può assumere, sono compresi tra un valore minimo pari a 0,0 ed un valore massimo uguale a $(1 - 1/n_c)$.

Il valore minimo si ha quando i record sono assegnati in modo omogeneo a tutte le classi.

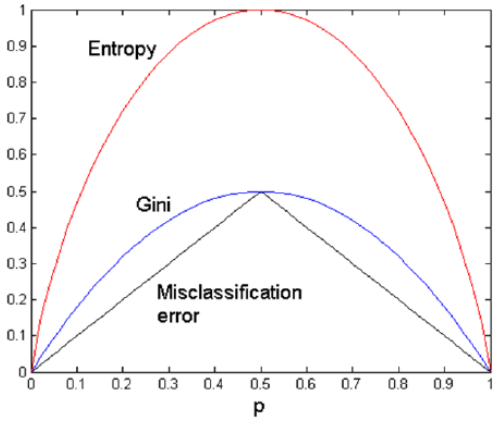
Il valore massimo si trova invece quando tutte le osservazioni appartengono ad una sola delle classi individuate. Si tratta della situazione ottimale.

L'obiettivo consiste quindi nel determinare un criterio di splitting basato sulla minimizzazione del Gini Index.

Una ulteriore alternativa è costituita dal ricorso all'Errore di classificazione come indice su cui basare il criterio di splitting. È espresso come:

$$1 - max_{j \in C}[p(j|t)]$$

Ipotizzando una classificazione basata su due classi, il seguente grafico permette un confronto tra i vari indici appena analizzati.



L'ordinata esprime il valore dell'indice considerato, l'ascissa invece tiene conto del fatto che la totalità degli esempi potrebbe essere classificata in esclusivamente una delle due classi (p=1) oppure potrebbe essere distribuita equamente tra entrambe le classi (p=0,5).

Solitamente l'indice utilizzato per effettuare lo splitting varia a seconda del tipo di algoritmo a cui si ricorre per costruire l'albero. L'algoritmo C 4.5, che verrà

successivamente trattato ed applicato ai dati forniti, utilizza l'Information Gain Ratio che, come prima esplicitato, sottintende il concetto di Entropia. Altri algoritmi come ad esempio il CART ricorrono invece al Gini Index.

Oltre allo splitting, un ulteriore elemento da prendere in esame quando si costruisce un albero di decisione è il pruning, cioè la potatura dell'albero. Infatti, la presenza di un numero troppo elevato di rami potrebbe provocare, da una parte un elevato rumore nei dati, dall'altra una limitata accuratezza ed un consistente errore di classificazione. Esistono perciò dei meccanismi di pruning distinti tra *pre-pruning* e *post-pruning* che adottano dei criteri di stop e di rimozione di specifiche parti dell'albero al fine di massimizzare l'accuratezza.

Il pre-pruning adotta dei criteri di stop che si innescano quando:

- La totalità delle istanze appartiene alla medesima classe
- Tutti gli attributi/record assumono lo stesso valore
- Effettuando un ulteriore splitting non migliora la misura dell'omogeneità espressa grazie agli indici prima esplicitati.
- Il numero delle istanze è inferiore ad un valore soglia indicato come user-specified threshhold.

Il post-pruning ricorre invece ad un approccio bottom-up che rimuove i nodi ed i rami da un albero completo. Si parte dai testing data e si individua il best pruned tree. Quando sui dati di testing si evidenzia che l'errore migliora in conseguenza alla sostituzione di una parte dell'albero con una foglia, si genera un *albero pruned* al posto del fully grown tree. Per quanto riguarda la classificazione e l'assegnazione dell'etichetta in corrispondenza della foglia, si determina la classe prevalente caratterizzante il sotto albero che è stato sostituito.

2.4 Regole di associazione: metodologia

La ricerca di regole di associazione (Association Rule Mining) è una metodologia inquadrata all'interno dei problemi di unsupervised learning. È stata introdotta agli inizi degli anni '90 da Agrawal ed è finalizzata alla scoperta relazioni significative tra i vari items di un dataset.

Le regole di associazione sono costituite da una premessa e da una conseguenza e sono poste sotto forma di dichiarazione del tipo $if \rightarrow then$. Vengono utilizzate per estrarre regolarità (pattern) da grandi moli di dati. Ogni regola viene quindi rappresentata con la notazione $X \rightarrow Y$, che esprime il fatto che se una generica transazione include l'item X, allora include anche l'item Y.

I campi applicativi dell'ARM sono molteplici ed includono il marketing, la bioinformatica, la manifattura e il quality engineering. Il contesto in cui si trovano più frequentemente le regole di associazione è quello del marketing, in particolare nell'ambito del *market basket analysis*, una tecnica utilizzata per analizzare le abitudini di acquisto dei clienti e adottare di conseguenza delle strategie di marketing sulla base dell'identificazione delle relazioni esistenti tra un elevato numero di articoli acquistati. Un classico esempio è fornito dai prodotti presenti nei supermercati, in cui l'obiettivo da raggiungere è quello di avvicinare fisicamente articoli che vengono frequentemente comprati insieme, in modo da spingerne la vendita.

Per valutare la validità e l'attendibilità di una regola di associazione si ricorre principalmente a due parametri di misura: il supporto e la confidenza che sono calcolati in base alle seguenti formule:

$$Supporto = P(X \cap Y) = \frac{N(X \cap Y)}{N}$$

Tale parametro esprime il rapporto tra il numero di istanze che includono sia gli items X che Y ed il numero totale di istanze.

Il supporto coincide con la rilevanza statistica della regola dal momento che, affinché una regola possa ritenersi tale, deve essere supportata da un campione sufficientemente ampio.
L'altro parametro è la confidenza, definita come segue:

$$Confidenza = \frac{N(X \cap Y)}{N(X)} = \frac{P(X \cap Y)}{P(X)} = P(Y|X)$$

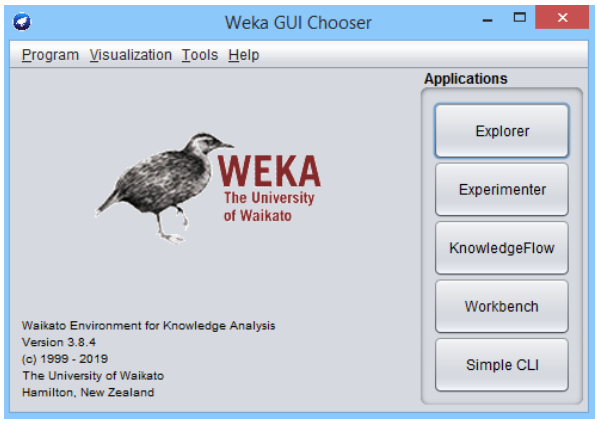
La confidenza rappresenta la frazione del numero di transazioni che contengono l'item Y tra tutte quelle che contengono X. Misura quindi quanto fortemente Y è determinato da X, cioè la forza della regola e corrisponde con quella che in statistica prende il nome di probabilità condizionata P(Y|X).
Oltre ai due appena descritti, esiste anche un ulteriore parametro degno di nota chiamato *Lift* e calcolato come il rapporto tra la confidenza e la probabilità di Y come indicato dalla formula:

$$Lift = \frac{P(Y|X)}{P(Y)}$$

Il Lift ha la funzione di mettere in evidenza la dipendenza statistica esistente tra X e Y. Se assume un valore superiore ad 1, significa che la correlazione è positiva, altrimenti si tratta di una correlazione negativa.

3. Il software Weka

Weka (Waikato Environment for Knowledge Analysis) è un software open source sviluppato dall'University of Waikato in Nuova Zelanda. Utilizza la GNU General Public License (GPL), è scritto interamente in linguaggio Java e presenta una GUI (Graphical User Interface) che consente di scegliere tra varie opzioni per l'analisi, interagire con i file contenenti dati (dataset) e visualizzare i risultati ottenuti.



Weka fornisce dei metodi per effettuare il pre-processing dei dati e mette a disposizione una serie di algoritmi di Machine Learning con lo scopo di estrapolare informazioni utili e generare dei modelli implementando tali algoritmi.
Sono presenti cinque interfacce che consentono l'interazione uomo-macchina: Explorer, Experimenter, Knowledge flow, Workbench, Simple CLI.

- *Explorer*: è costituito da un pannello che permette di attingere ai dati posti su file, su database SQL, o su URL. I dati possono poi andare incontro ad un pre-processing attraverso lo strumento "filter" oppure la cancellazione di attributi superflui ai fini dell'analisi. Più in dettaglio, il pre-processing tramite i filtri consente di effettuare varie operazioni tra cui:
    - Discretizzazione: discretizzazione di un range di attributi numerici in attributi di tipo nominal;

- Normalizzazione: “normalize” consente di ricondurre un qualunque intervallo finito di valori in un diverso intervallo [0,1], mentre “standardize” effettua la standardizzazione di tutti gli attributi numerici in modo che abbiano media pari a zero e varianza unitaria.
- Trasformazione di attributi: ad esempio numeric to nominal, nominal to binary, numeric to string, ecc..
- Gestione valori mancanti: replace missing values sostituisce i valori mancanti per gli attributi numeric e nominal con altri valori compatibili.

Il pannello Explorer mostra infine anche l’istogramma relativo agli attributi selezionati e alcuni dati statistici ad essi collegati.

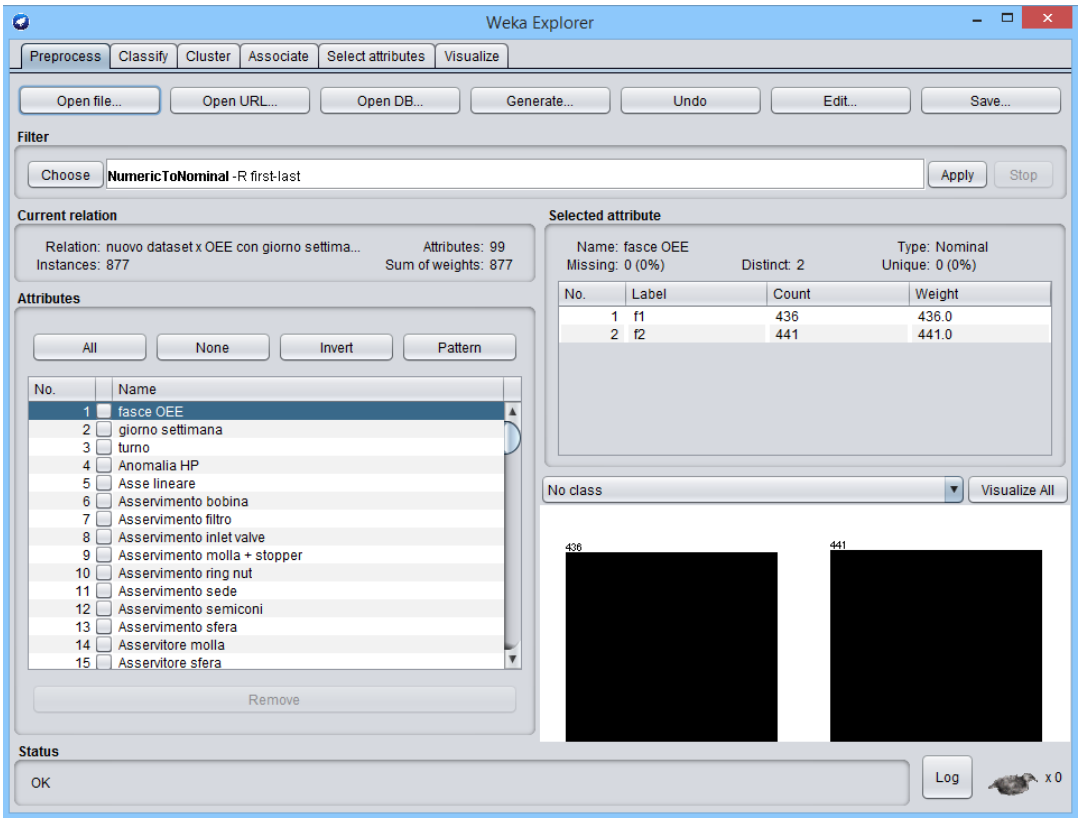
- *Experimenter*: rappresenta l’interfaccia per la comparazione dei risultati relativi a diversi modelli di apprendimento. Più concretamente, consente di applicare in contemporanea differenti algoritmi di Data Mining su un unico o su più dataset, in modo tale da riuscire a confrontare più schemi di apprendimento. Questa interfaccia è adatta per problemi di classificazione o regressione.

- *Knowledge flow*: è un’interfaccia alternativa all’Explorer che consente di visualizzare graficamente il flusso di dati all’interno di un sistema. Per prima cosi vengono selezionati dei componenti da una barra delle applicazioni, li si dispone in una finestra in cui saranno successivamente connessi andando a costruire un grafo che analizza ed elabora i dati.

- *Workbench*: serve a raccogliere le funzioni delle altre interfacce precedenti in un’unica interfaccia.

- *Simple CLI*: è un'interfaccia testuale (Simple Command Line Interface) in cui è possibile inserire righe di comando ed accedere a tutte le tecniche di apprendimento.

Per l'analisi dei dati forniti si farà riferimento all'interfaccia Explorer. La seguente immagine illustra come appare tale l'interfaccia, una volta aperto il file relativo al dataset (in formato csv) creato appositamente per questo scopo.



Il data pre-processing è stato effettuato trasformando gli attributi da numeric a nominal, in quanto il type nominal è più adatto a rappresentare l'informazione relativa al verificarsi o meno di un determinato evento di guasto. Inoltre sono stati eliminati due attributi poco significativi quali la data ed il valore numerico dell'OEE, lasciando però quello relativo alla suddivisione dell'OEE in fasce, la cui metodologia verrà esplicitata successivamente in modo dettagliato. Nella finestra "selected attribute" vengono specificate informazioni numeriche e statistiche circa l'attributo che si sceglie di selezionare ed al di sotto è posto l'istogramma le cui colonne

mostrano il conteggio di ciascuna categoria in cui viene suddiviso ogni attributo. A questo punto Weka offre diversi metodi di analisi, la classificazione (Classify), il clustering (Cluster), le regole di associazione (Associate), la selezione di attributi rilevanti per una specifica funzione (Select attributes), la visualizzazione di relazioni tra gli attributi (Visualize). Sono tutte scelte opzionabili cliccando sul rispettivo pannello. In questa trattazione i dati verranno analizzati attraverso alberi di decisione e regole di associazione. La procedura da seguire per ottenere un valido risultato consiste nel:

- Individuare l'algoritmo da utilizzare tra quelli disponibili e impostarne le proprietà;
- Definire come interpretare il risultato e valutarne l'attendibilità;
- Visualizzare e commentare il risultato ottenuto tramite l'algoritmo;

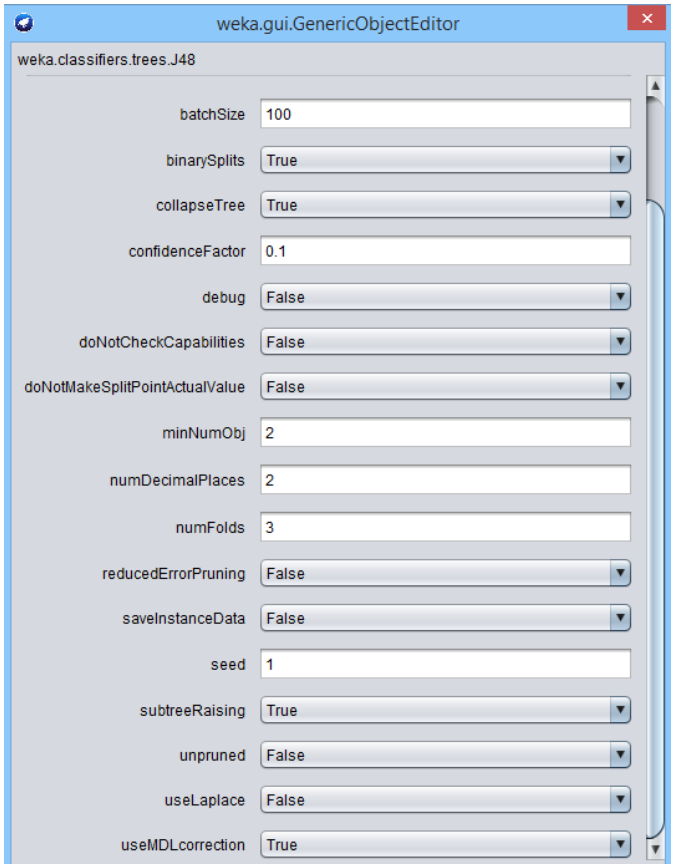
Ai fini dell'analisi gli algoritmi utilizzati saranno, per gli alberi di decisione, il J48 (l'implementazione in Java dell'algoritmo C4.5) ed il Random Forest. Per le regole di associazione invece si ricorrerà all'algoritmo Apriori. Questi algoritmi verranno ampiamente descritti rispettivamente nei paragrafi 3.2, 3.3, 3.4.
L'attendibilità del risultato dei decision trees e quindi anche il grado di accuratezza ottenuto nella classificazione può essere giudicato osservando il numero di istanze correttamente classificate espresso sia in termini numerici, sia in percentuale rispetto al totale delle istanze. Per le regole di associazione è invece necessario valutare il valore minimo scelto per supporto e confidenza e capire se la regola estratta è utile per supportare il decisore oppure se è frutto della pura casualità.

3.1 Il setting dei parametri in Weka

Prima di lanciare l'algoritmo, è opportuno impostare correttamente una serie di parametri fondamentali aprendo la finestra relativa alle proprietà.

A seconda del tipo di algoritmo utilizzato variano i parametri su cui effettuare il setting.

Nel caso degli alberi di decisione generati tramite l'algoritmo J48 (C4.5) i valori e le proprietà opzionabili sono illustrati dalla seguente finestra.



Tra i parametri utili per generare il decision tree è possibile elencare:

- *Binary splits*: se impostato su "TRUE" consente di generare un albero di decisione di tipo binario effettuando uno splitting binario su attributi di tipo nominal;

- *Collapse Tree*: stabilisce se rimuovere le parti di albero che non riducono l'errore di training. Anche questo parametro può essere settato su "TRUE" o su "FALSE";
- *Debug*: se impostato su "TRUE" il classificatore il classificatore fornisce ulteriori informazioni sulla console;
- *Do Not Check Capabilities*: se settato su "TRUE", le capacità del classificatore non vengono controllate prima che questo vanga costruito;
- *Min Num Obj*: è il minimo numero di istanze per foglia, corrispondente cioè con il minimo numero di istanze che devono essere classificate;
- *Save Instance Data*: stabilisce se salvare i dati di training per la visualizzazione;
- *Use MDL correction*: se impostato su "TRUE" permette di ricorrere alla "MDL (Minimum Description Length) correction" quando si effettua lo splitting su attributi numerici. Ciò produce alberi di dimensioni inferiori;

Molti dei parametri restanti sono invece relativi alla potatura dell'albero (pruning), sono spesso dipendenti l'uno dall'altro e sono proposti di seguito:

- *Confidence factor*: a seconda del valore impostato si definisce il grado di potatura. Valori bassi del Confidence factor implicano un pruning maggiore. Il valore di default proposto da Weka è 0,25 ma può essere aumentato o diminuito andando ad influire sul livello di specificità/generalità dell'albero.
- *Reduced Error Pruning*: se impostato su "TRUE" abilità una strategia di potatura alternativa rispetto a quella dell'algoritmo C4.5, adottando una potatura mirata alla riduzione dell'errore di pruning;
- *Num Folds*: determina l'ammontare dei dati utilizzato quando si opta per la strategia "ReducedErrorPruning". Un "fold" serve per il pruning, i restanti vengono utilizzati per generare l'albero;
- *Subtree Raising*: nella fase di pruning, se impostato su "TRUE", permette di attuare l'operazione di "subtree raising", cioè di eliminazione di un nodo e di redistribuzione delle istanze.

- *Unpruned*: se impostato su “TRUE” genera un albero privo di potatura, se “FALSE” si ricorre invece di default al metodo di potatura dell’algoritmo C4.5.

Il pruning costituisce un passaggio indispensabile sia nell’ottica dell’ottimizzazione dell’efficienza computazionale, sia nell’ottenimento di un’adeguata accuratezza di classificazione. Il meccanismo su cui si basa il pruning è legato al confronto tra il valore dell’errore prima e dopo ogni potatura per poi stabilire quale soluzione lo minimizza. Come accennato nel paragrafo 2.3 nella descrizione metodologica degli alberi di decisione, esistono due tipi di pruning che verranno ora trattati più in dettaglio, il pre-pruning ed il post-pruning.

Nel pre-pruning si interrompe lo sviluppo dell’albero quando le informazioni diventano meno attendibili, cioè quando non esiste un’associazione significativa dal punto di vista statistico tra l’attributo e la classe in corrispondenza di un determinato nodo.

Nel post-pruning invece, viene prima generato l’albero nella sua interezza (fully grown decision tree) e successivamente vengono tagliate quelle parti non ritenute affidabili. In questo contesto sono previste due tipologie di operazioni: il subtree replacement ed il subtree raising.

La prima ha un approccio bottom up in cui si valuta la sostituzione di un sottoalbero solo dopo aver considerato tutti i singoli sottoalberi.

Diversamente, nel subtree raising si elimina un nodo redistribuendo le istanze. Si tratta di un’operazione che richiede un tempo maggiore rispetto al replacement (sostituzione).

In Weka, se si sceglie di utilizzare il metodo di potatura caratteristico dell’algoritmo C4.5, per regolare il grado di potatura e quindi la specificità/generalità della struttura dell’albero ed allo stesso tempo minimizzare l’errore di classificazione, è sufficiente far variare il valore del parametro Confidence factor precedentemente introdotto. Tale parametro può assumere valori compresi tra 0 ed 1. Con valori vicini allo zero, si riduce la dimensione e la complessità dell’albero e si tagliano via

nodi scarsamente rilevanti che tendono ad incrementare l'errore di classificazione. Man mano che aumenta il confidence factor invece, l'algoritmo richiede maggior tempo per fornire il risultato in output e l'albero ottenuto possiede un numero di nodi e sottoalberi più elevato rispetto al caso precedente, diventando meno legibile e tendenzialmente meno accurato.

Un approccio differente da quelli visti fino ad ora è fornito dal "Reduced Error Pruning". È un metodo che suddivide i dati in due sottoinsiemi: il training set ed il validation set.

Se l'errore di classificazione del nuovo albero che ha subito la potatura è uguale o inferiore a quello dell'albero privo di pruning e se il sottoalbero non contiene nessun sottoalbero con la stessa proprietà, si effettua il pruning sostituendo il sottoalbero con una foglia.

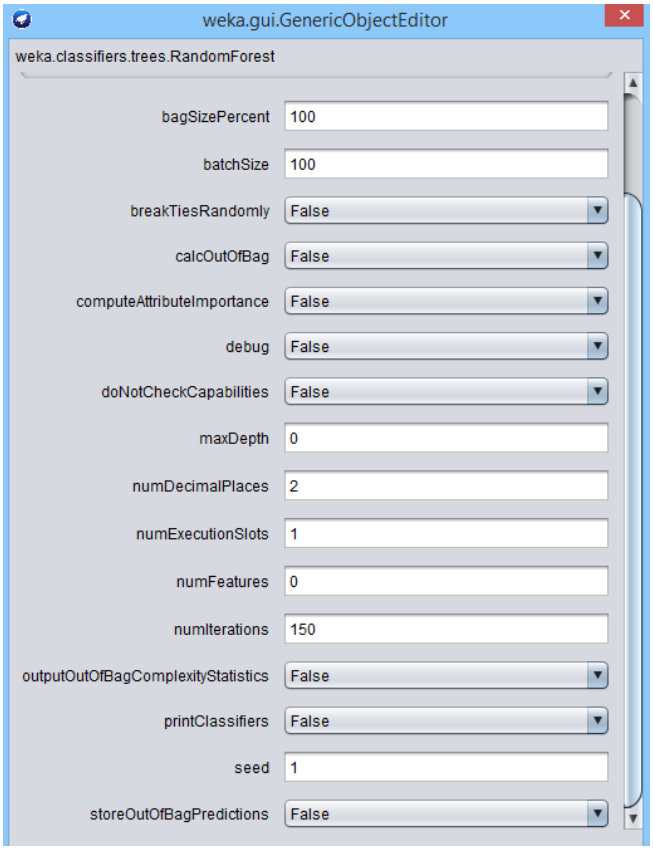
Se le condizioni appena descritte non si dovessero verificare, l'albero non deve essere potato. In questo modo la potatura viene effettuata sul nodo che garantisce la migliore accuratezza sul validation set. Questo approccio presenta però dei limiti quando la mole di dati non è elevata, mentre il vantaggio consiste sostanzialmente nel fatto che è caratterizzato da una complessità computazionale di tipo lineare.

Rimanendo sempre nell'ambito della classificazione, oltre al J48 (C4.5) si ricorrerà anche ad un ulteriore algoritmo: il Random Forest.

In modo analogo al precedente tale algoritmo, implementato in Weka, richiede di impostare una serie di parametri chiave.

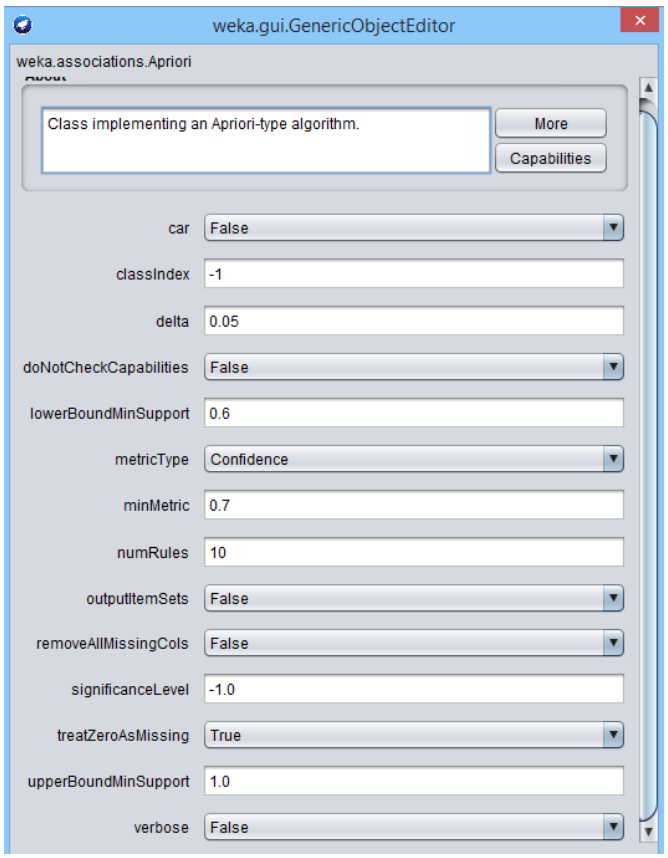
Uno dei parametri più significativi è rappresentato dal "*Num Iterations*" perché stabilisce il numero di alberi che vengono generati contemporaneamente nel momento in cui si preme il pulsante start.

Relativamente ai rimanenti parametri verrà fornita una breve descrizione e spiegazione di seguito.



- *Bag Size Percent*: dimensione di ogni "bag" come percentuale della dimensione del training set;
- *Break Ties randomly*: effettua una scelta in modo casuale quando diversi attributi sembrano ugualmente validi;
- *Max Depth*: corrisponde alla profondità massima dell'albero. il valore di default è 0 ed esprime profondità illimitata;
- *Compute attribute Importance*: calcola l'importanza degli attributi tramite la diminuzione dell'impurità media;
- *Calc Out of Bag*: indica se viene calcolato l'errore out-of-bag;
- *Num Features*: imposta il numero degli attributi scelti in modo random. Se il valore è pari a zero, significa che si utilizza int ($\log_2$(#predictors)+1);
- *Print classifier*: stampa in output i singoli classificatori;
- *Store Out of Bag*: indica se memorizzare le previsioni out-of-bag;

Un discorso analogo vale per l'algoritmo Apriori che verrà spiegato in ogni suo dettaglio nel paragrafo 3.4. I parametri che devono essere regolati in riferimento a questo algoritmo sono proposti di seguito.



- *Delta*: riduce il supporto in modo iterativo fino a quando non si raggiunge il valore di supporto minimo oppure fino a quando non viene generato un numero di regole uguale a quello richiesto dall'utente;
- *Output itemset*: se impostato su "TRUE" vengono mostrati in output gli itemset generati, altrimenti non risultano visibili per l'utente finale;
- *NumRules*: numero complessivo di regole da trovare impostato dall'utente;
- *Treat Zero as Missing*: se impostato su "TRUE", lo zero viene considerato al pari di un valore mancante.

  Tale impostazione si rende necessaria nel caso dell'analisi in questione dal momento che il dataset è composto prevalentemente da valoro zero legati

al fatto che è molto più frequente il caso in cui il guasto non si verifica rispetto al caso in cui si verifica. La notazione utilizzata nella creazione del infatti, come specificato più avanti, è contraddistinta da valori 1 e 0.

- *Remove all missing cals*: rimuove tutte le colonne caratterizzate interamente da valori mancanti.

  Questa situazione non si presenta nel caso dei dati forniti perché ogni evento di guasto si manifesta almeno una volta nel corso del periodo considerato.
- *Metric type*: imposta il tipo di metrica in base alla quale classificare le regole. Le scelte opzionabili sono: “confidence”, “lift”, “leverage”, “conviction”.
- *Lower Bound min support*: limite inferiore per il supporto minimo. Si tratta di un parametro di fondamentale importanza perché rappresenta la rilevanza statistica e stabilisce quindi se una regola possa ritenersi affidabile (se il supporto è alto), oppure casuale (se il supporto è basso).
- *Upper bound min support*: limite superiore per il supporto minimo. L’algoritmo interviene diminuendo iterativamente il supporto minimo a partire da questo valore evitando però di scendere al di sotto del lower bound min support.
- *Min metric*: coincide con il minimo valore accettabile per il tipo di metrica che è stata scelta attraverso il parametro “metric type”.

3.2 L'algoritmo J48 (C4.5 algorithm)

L'algoritmo J48 è la versione implementata in Weka, nel linguaggio Java, dell'algoritmo C4.5. Il C4.5 è un algoritmo sviluppato da Ross Quinlan nel 1993 ed è ampiamente utilizzato per generare gli alberi di decisione. Rappresenta l'evoluzione dell'algoritmo ID3, sviluppato dallo stesso Quinlan nel 1986. Le differenze sostanziali sono legate al fatto che il C4.5:

- È in grado di manipolare anche attributi di natura continua, oltre a quelli di tipo discreto;
- Può trattare dati di training caratterizzati da valori mancanti in corrispondenza di determinati attributi. Attributi con valori mancanti non vengono considerati nel calcolare il gain e l'entropia;
- Effettua la potatura dell'albero una volta che questo è stato costruito interamente (post-pruning);

L'albero generato per mezzo dell'algoritmo C4.5 è il risultato di un processo di apprendimento a partire da un training set, in cui ogni esempio è contraddistinto da una coppia attributo-valore. L'approccio utilizzato prende il nome di "divide and conquer" e si tratta di una strategia Greedy che costruisce l'albero in maniera ricorsiva, procedendo in modo top-down. Tale approccio si basa sulla suddivisione dei record a seconda del valore degli attributi con l'obiettivo di ottimizzare un determinato criterio. In questo modo, per poter costruire l'albero, il training set viene partizionato ricorsivamente in sottoinsiemi più piccoli.

L'algoritmo C4.5 utilizza la misura dell'Information Gain Ratio (cioè l'Information Gain Normalizzato) come criterio di splitting per selezionare l'attributo in corrispondenza di ogni nodo. Si opta per l'attributo che possiede il massimo valore di Information Gain Ratio (IGR) ed allo stesso modo si sceglie il nodo radice. L'algoritmo va avanti su ogni sottoinsieme e quando si verifica che la totalità dei campioni appartiene alla medesima classe, si effettua alla classificazione, assegnando una foglia.

Originariamente, il criterio di splitting si basava sull'Information Gain, anziché sull'Information Gain Ratio (IGR). La misura dell'Information Gain era tuttavia influenzata dal numero di stati/valori dell'attributo su cui doveva essere effettuato lo splitting. Ciò che si verificava è che veniva data la priorità ad attributi caratterizzati da un elevato numero di stati. Per ovviare a ciò, Quinlan ha normalizzato il valore dell'Information Gain di un determinato attributo X, calcolando l'Information Gain Ratio, a partire dall'Information Gain, secondo la seguente formula:

$$Information\ Gain\ Ratio\ (IGR) = \frac{IG(X)}{SplitInfo(X)}$$

In cui:

IG: Information Gain;

SplitInfo: entropia della distribuzione delle istanze nella partizione;

La differenza tra IGR ed Entropia sta nel fatto che l'Entropia rappresenta l'impurità che contraddistingue una collezione di osservazioni/esempi, mentre l'Information Gain Ratio identifica la riduzione di entropia determinata dal partizionamento di tali osservazioni/esempi. Nella pratica quanto appena descritto si traduce nel fatto che nel momento in cui da un attributo si origina un ramo, si valuta la riduzione che subisce il valore dell'entropia a seguito di questa ramificazione.

Lo splitting si arresta quando non esiste più alcun attributo con un valore positivo di Information Gain Ratio oppure nel caso in cui si raggiunga il minimo numero di istanze per ogni foglia, solitamente 2. In aggiunta a questi due criteri di stop, la generazione di nuovi rami si arresta in mancanza di un attributo valido su cui effettuare lo splitting.

Tutti gli steps che riassumono la metodologia generale da seguire quando si costruisce un decision tree ricorrendo a questo algoritmo sono elencati di seguito:

i. Selezionare il dataset da fornire in input all'algoritmo per effettuare il pre-processing;
ii. Selezionare i classificatori;

iii. Calcolare l'Entropia, l'Information Gain, l'Information Gain Ratio degli attributi;
iv. Effettuare il processing del dataset fornito in input attraverso l'algoritmo C4.5;
v. Sempre per mezzo dell'algoritmo generare l'albero di decisione;

Alcuni dei passaggi appena descritti, come ad esempio il calcolo dell'Information Gain Ratio per effettuare lo splitting, vengono eseguiti implicitamente in Weka nel momento in cui si lancia l'algoritmo, premendo il pulsante "start" e non risultano quindi visibili all'utente finale, il quale ottiene in output direttamente l'albero di decisione insieme ad una serie di parametri utili per valutarne l'accuratezza e l'affidabilità.

Lo pseudocodice relativo all'algoritmo C4.5 è riassunto nei passaggio sottostanti:

I. Controllare tutti i casi base;
II. Per ogni attributo *"a"* ricavare l'*Information Gain Ratio* da ogni splitting effettuato sull'attributo ***a*;**
III. Selezionare l'*a-best*, cioè l'attributo con il valore più alto di *Information Gain Ratio*;
IV. Creare un nodo che splitta su *a-best*;
V. Effettuando lo splitting sull'*a-best* e procedendo ricorsivamente si ottengono via via dei nuovi nodi considerati come nodi figli;

3.3 L'algoritmo Random Forest

L'algoritmo Random Forest si basa su un insieme di metodi di apprendimento, appartenenti alla categoria dei *supervised learning*, proposti per la prima volta da Breiman nel 2001 e successivamente revisionati da Cutler ed utilizzati per problemi di classificazione e regressione. Tale algoritmo opera costruendo una moltitudine di modelli di alberi di decisione $[h(X,\theta_k);k=1,...]$, utilizzati per il training, in cui ciascuno di essi è completamente indipendente dagli altri. Nel caso dei problemi di classificazione ogni decision tree effettua separatamente una predizione e, successivamente le previsioni dei singoli alberi vengono combinate in modo da ottenere il risultato finale della classificazione.

Di seguito sono riportati tutti i passaggi seguiti dall'algoritmo:

- Si considera che il training set sia costituito da N campioni. Attraverso un campionamento bootstrap e un campionamento random si estrae un insieme N_{tree} di campioni e si costruisce N_{tree} alberi di classificazione. I campioni che ogni volta non vengono estratti vanno a far parte dell'N_{tree} out-of-bag (OOB).
- Presupponendo la presenza di M fattori/caratteristiche/variabili, si estraggono attraverso il campionamento random un numero di fattori inferiore a M, indicato con M_{try}. Tra gli M_{try} selezionare l'attributo ottimo su cui effettuare lo splitting tenendo conto che è necessario minimizzare l'impurità presente ad ogni nodo e si procede con lo splitting.
- L'insieme degli alberi di classificazione che sono stati generati con questa strategia costituisce il Random Forest. I nuovi dati vengono suddivisi e classificati in base al classificatore. L'esito finale della classificazione è ottenuto per maggioranza. Il Random Forest infatti, si serve della totalità degli alberi per effettuare la classificazione facendo in modo che tali alberi votino optando per la classe che risulta più popolare.

Per ottenere un risultato affidabile il numero di N_{tree} dovrebbe essere abbastanza elevato da garantire la diversità di ciascun classificatore dall'altro, ma allo stesso tempo l'aumento di N_{tree} incrementa la complessità e riduce l'interpretabilità dell'algoritmo. Se M_{try} è molto piccolo ne risentirà l'accuratezza di classificazione di ogni singolo albero, al contrario se si considera M_{try} elevato, la diversità sarà minore e maggiore risulterà l'accuratezza di ogni singolo albero. Molti degli studi presenti in letteratura propongono solitamente di scegliere $M_{try} = \sqrt{\mathrm{M}}$.

Come precedentemente accennato, l'algoritmo Random Forest introduce il concetto di out-of-bag (OOG). L'OOG è spiegabile con il fatto che nella costruzione dell'insieme dei decision trees, nella costituzione di ognuno di essi (per semplificare si suppone il k-esimo), circa un terzo dei campioni non contribuisce alla generazione dell'albero. Questi campioni prendono il nome di out-of-bag riferiti al k-esimo albero. Ogni campione OOB ha la funzione di stimare l'errore riferito a tutti quei campioni che restano fuori dal training set e questo errore rappresenta una stima dell'errore di generalizzazione che caratterizza il modello.

L'algoritmo Random Forest riesce inoltre, a partire dai campioni OOB, a valutare l'importanza dei vari fattori/caratteristiche. Assumendo infatti che un determinato fattore possa subire una variazione, riesce a misurarne l'importanza calcolando la Mean Decreasing Accuracy (MDA) in base alla formula proposta di seguito:

$$MDA = \sum (errOOB_n - errOOB_o)/N_{tree}$$

In cui:

$errOOB_n$ rappresenta l'errore OOB conseguente alla variazione del fattore;

$errOOB_o$ rappresenta l'errore OOB prima della variazione del fattore;

Minore risulta il valore di MDA, più elevata è l'importanza di quella determinata caratteristica.

Definire l'importanza di ogni caratteristica è utile al fine di creare un modello che includa per lo più le variabili di maggior rilievo, in modo da risultare di facile interpretazione, da ridurne la varianza e diminuirne il costo computazionale ed il tempo necessario per il training.

I vantaggi nell'uso di questo tipo di algoritmo sono riassunti nel seguenti punti:

- È capace di operare in modo efficiente su dataset contenenti elevate quantità di dati;
- Possiede un'eccellente accuratezza tra tutti i vari algoritmi;
- Riesce a minimizzare gli errori nella definizione delle classi in caso di dataset particolarmente sbilanciati;
- Supporta metodi efficaci per stimare i dati mancanti, mantenendo un elevato livello di accuratezza anche nel caso di presenza di una elevata quantità di dati mancanti;

3.4 L'algoritmo Apriori

L'algoritmo Apriori è uno tra i più diffusi ed utilizzati algoritmi per l'Association Rules Mining (ARM) ed è stato sviluppato da Agrawal e Srikant nel 1994. Nel 2006 è stato indicato dall'IEEE International Conference on Data Mining (ICDM) tra i dieci algoritmi più influenti nell'ambito della ricerca.

La sua funzione consiste nel ricercare ed estrarre itemset frequenti da un ampio database per poter successivamente ricavare una serie di regole associative finalizzate all'ottenimento di conoscenza dai dati. Per spiegare in cosa consiste, sono elencate di seguito una serie di notazioni e definizioni fornite dallo stesso Agrawal nel 1993 nella formulazione del problema:

- $I = \{i_1, i_2, \dots i_m\}$ l'insieme degli items;
- D l'insieme delle transazioni in cui ogni transazione t è inclusa in I;
- X l'insieme degli items ottenuti a partire da I;
- $X \to Y$ è una regola di associazione in cui $X \subseteq I,\ Y \subseteq I,\ X \cap Y = \emptyset$;
- c è la confidenza della regola se il valore di c espresso in percentuale sulle transazioni in D che contengono l'insieme X, contengono anche l'insieme Y;
- s è il supporto della regola se il valore di s espresso in percentuale sulle transazioni in D contiene l'insieme $X \cup Y$.

L'obiettivo del problema consiste nel determinare un numero specifico di regole di associazione caratterizzate da un valore di confidenza e supporto più elevato del valore del valore minimo specificato dall'utente per entrambi i parametri.

A questo scopo prima di esplicitare l'algoritmo si stabiliscono tutte le condizioni iniziali:

- $L_k =$ insieme dei k-large itemset;
- $C_k =$ insieme dei k-itemset candidati;
- $D =$ insieme delle transazioni, in cui $t \subset D$;

Dopodiché è importante analizzare l'algoritmo Apriori, il cui fine è quello di ricercare i frequent itemset. Il termine frequenza di un itemset è espresso in riferimento del numero di istanze che lo soddisfano. Per itemset frequente si intende un itemset caratterizzato da una frequenza relativa che supera il valore minimo (soglia) indicato per il supporto.

L1 = {frequent 1-itemsets};
for(k = 2;$L_{k-1} \neq 0$; k + +){
C_k = set of new candidates;
for all transactions $t \subset D$
for all k-subsets m of t
if ($m \subset C_k$) m.count + +
L_k = {$n \subset C_k | \; n.count \geq minsupp$}
}
Set of all frequent itemsets = $\cup_k L_k$*;*

Una volta eseguito l'Apriori, per ricavare le regole di associazione è indispensabile applicare un nuovo algoritmo che stabilisce che per la totalità dei large itemset e per ogni itemset *l*, è necessario trovare tutti i sottoinsiemi non vuoti di *l* per ogni sottoinsieme *a* di *l*, in modo che la regola sia:

$$a \rightarrow (l - a) \qquad se \; \frac{supporto\ (l)}{supporto(a)} > confidenza\ minima$$

Di conseguenza, oltre alla soglia di supporto si definisce anche il valore minimo accettabile (soglia) per la confidenza che ha la funzione di indicare la forza della regola.

for (each frequent itemset l){
generate all non-empty subsets of l
for (each non-empty subset a of l) {
output_rule: $a \rightarrow (l - a)$ *if support*(l)*/support*$(a) \geq$ *min confidence}*
}

Nel caso generale, da *m* items si possono ricavare fino a 2^m frequent itemset. Ognuna di queste operazioni richiede una elevata quantità di risorse computazionali e di memoria per il fatto che ad ogni iterazione l'intero database deve essere scansionato, con l'obiettivo di ottenere il supporto per i nuovi candidati. Per questo motivo le prestazioni dell'algoritmo tendono a deteriorarsi all'aumentare della dimensione del database ed a diventare inaccettabili ad esempio nel caso di database contenenti dati provenienti da social network o di ampi dataset di natura bioinformatica.

L'algoritmo Apriori quindi, a causa della sua complessità esponenziale, oltre a richiedere un elevato spazio in memoria, necessita di un elevato tempo di risposta e provoca un costoso spreco di tempo in quanto ha a che fare spesso con un ampio numero di candidati, con itemsets molto frequenti e con basso supporto minimo.

4. Applicazione delle tecniche di Data Mining ai dati forniti

Dopo aver definito le metodologie di Data Mining e indicato i rispettivi algoritmi in grado di metterle in pratica, è ora necessario mostrare i risultati restituiti dall'applicazione di tali algoritmi ai dati forniti in input attraverso l'uso del software Weka. A tal fine si è reso indispensabile l'operazione di *data preparation,* (una delle fasi del processo di Data Mining definite dall'approccio metodologico CRISP-DM), mirata alla selezione degli attributi ed all'esecuzione di tutte le trasformazioni che consentono la realizzazione di un dataset a partire dai dati grezzi. A tale processo seguiranno poi le fasi di *modelling*, *evaluation* e *deployement*, tutte implicitamente analizzate in questo capitolo. Il paragrafo successivo descrive il procedimento seguito per creare il dataset.

4.1 Realizzazione del dataset

A partire dai dati forniti dai file *Data Collector* e *Data Guasti*, le informazioni rilevanti sono state incorporate creando in Excel un unico dataset che riporta tutti gli eventi di guasto (in totale 96), la data, il turno e l'OEE relativo ai rispettivi turno e data. Nell'impostare il dataset, dalla mole di dati sono state eliminate le righe che presentavano delle incongruenze o delle omissioni riscontrate durante il confronto tra i file *Data Collector* e *Data Guasti*. In alcuni casi infatti, si manifestava la mancata corrispondenza tra le righe relative al giorno ed al turno del file *Data Guasti* e quelle omologhe del file *Data Collector*, originando degli spazi privi di informazione sui guasti verificatisi.

Il file risultante riporta quindi le informazioni relative alla presenza di un determinato evento di guasto in un preciso giorno e turno e l'OEE ad essi collegato. Pertanto si è indicato con:

1 : il guasto si verifica

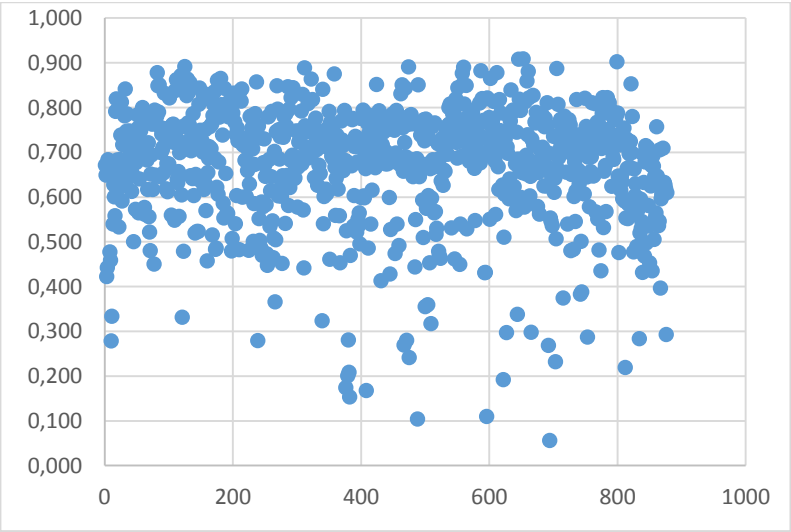
0 : il guasto non si verifica

Nel dataset, le due colonne all'estrema sinistra sono state utilizzate per indicare l'OEE e la corrispondente fascia assegnata ad ogni valore di OEE. Le colonne immediatamente accanto servono distinguere la data ed il turno. Le restanti colonne riportano invece la totalità degli eventi di guasto elencati nel file *Data Guasti,* ordinati alfabeticamente da sinistra verso destra. Nel momento in cui il guasto si verifica in corrispondenza di un determinato giorno e turno nella cella si inserisce 1, altrimenti 0.

Ai fini della classificazione i valori di OEE devono essere raggruppati in fasce. Due sono i possibili criteri per l'individuazione di tali fase: per frequenza o per ampiezza. In questo caso raggruppando per ampiezza (cioè con fasce tutte della medesima ampiezza) si otterrebbero alcune fasce contenenti un numero molto elevato di valori, altre invece un numero troppo limitato di valori.

Pertanto, dal momento che la distribuzione dei valori di OEE non è uniforme, questo approccio non è attuabile poiché comporterebbe un alto numero si misclassificazioni.

La distribuzione dei valori di OEE è rappresentata dal seguente grafico a dispersione. L'ascissa indica il numero di istanze analizzate (in totale 877), mentre l'ordinata mostra la scala di valori compresa tra 0 e 1 entro cui tali valori sono dispersi.



Come si può evincere dal grafico, la stragrande maggioranza dei valori di OEE è concentrata in un range incluso tra 0,5 e 0,9 e pertanto la distribuzione appare più fitta all'interno di questo intervallo.

Un approccio migliore è quello che consiste nel selezionare le fasce per frequenza, creando cioè fasce di diversa ampiezza, raggruppate per frequenza e contenenti quindi un numero simile o pressoché uguale di valori. In questo modo si riesce a limitare, per quanto possibile, l'errore di classificazione evidenziato dalla matrice di confusione. Il raggruppamento dell'OEE in fasce e quindi la rispettiva fascia di appartenenza è stata inserita come informazione all'interno del dataset, il quale rappresenta la base per la realizzazione di alberi di decisione e per la definizione di regole di associazione. La tabella sottostante costituisce un esempio di come è stato costruito il dataset. Ne è stata riportata solo una porzione per motivi di spazio, le informazioni complete sono contenute nel rispettivo file Excel.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	*fasce OEE*	*OEE*	*data*	*giorno settimana*	*turno*	**Anomalia HP**	**Asse lineare**	**Asservimento bobina**	**Asservimento filtro**	**Asservimento inlet valve**	**Asservimento molla + stopper**	**Asservimento ring nut**	**Asservimento sede**	**Asservimento semiconi**
2	f1	0,671	02/01/2019	3	2	0	0	0	0	0	0	1	0	0
3	f1	0,649	02/01/2019	3	3	0	0	1	0	0	0	1	0	0
4	f1	0,422	03/01/2019	4	2	0	0	0	0	0	0	0	0	0
5	f1	0,442	03/01/2019	4	3	0	0	0	0	0	0	1	0	0
6	f1	0,683	04/01/2019	5	2	0	0	0	0	0	0	0	0	0
7	f1	0,654	04/01/2019	5	3	0	0	0	0	0	0	1	0	0
8	f1	0,664	05/01/2019	6	1	0	0	0	0	0	0	0	0	0
9	f1	0,478	06/01/2019	7	1	0	0	0	0	0	0	0	0	0
10	f1	0,459	07/01/2019	1	1	0	0	0	0	1	0	0	0	0
11	f1	0,279	07/01/2019	1	2	0	0	0	0	0	0	0	0	0
12	f1	0,334	07/01/2019	1	1	0	0	0	0	0	0	1	0	0
13	f1	0,645	08/01/2019	2	1	0	0	0	0	0	0	0	0	0
14	f1	0,539	08/01/2019	2	2	0	0	0	0	0	0	0	0	0
15	f1	0,628	09/01/2019	3	1	0	0	0	0	1	1	1	0	0
16	f1	0,600	09/01/2019	3	2	0	0	0	0	0	1	1	0	0
17	f1	0,558	09/01/2019	3	3	0	0	0	1	0	1	0	0	0
18	f2	0,791	24/01/2019	4	2	0	0	0	0	0	0	0	0	0
19	f2	0,819	27/01/2019	7	2	0	0	0	0	0	0	0	0	0
20	f1	0,683	29/01/2019	2	1	0	0	0	0	0	0	0	0	0
21	f1	0,671	30/01/2019	3	1	1	0	0	0	0	0	0	0	0
22	f1	0,661	30/01/2019	3	3	0	0	0	0	0	0	0	0	0
23	f1	0,533	31/01/2019	4	1	0	0	0	0	0	0	0	0	0
24	f1	0,621	31/01/2019	4	3	0	0	0	0	0	0	1	0	0
25	f1	0,684	01/02/2019	5	1	1	0	0	0	0	0	0	0	0
26	f2	0,738	02/02/2019	6	1	0	0	0	0	0	0	0	0	0
27	f2	0,812	01/02/2019	5	3	0	0	0	0	0	0	0	0	0
28	f1	0,591	03/02/2019	7	1	0	0	0	0	0	0	0	0	0
29	f2	0,717	04/02/2019	1	3	0	0	1	0	0	0	0	0	0

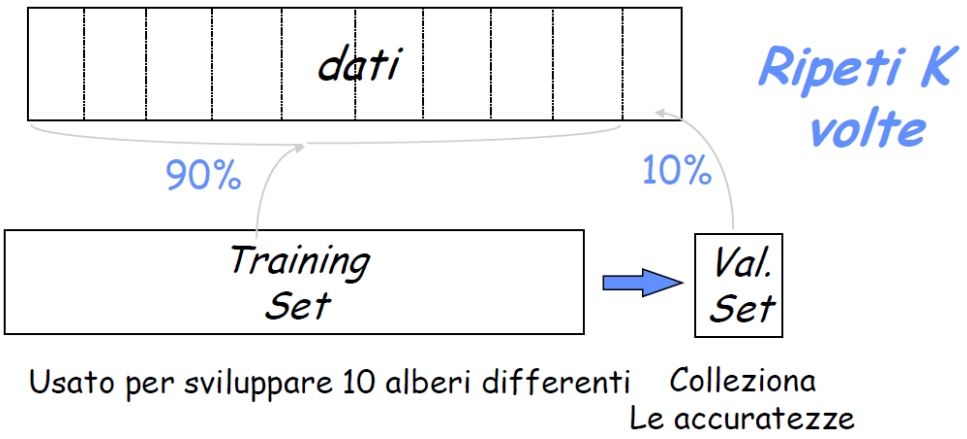
4.2 Alberi di decisione: applicazione

Fornendo in input in Weka il dataset precedentemente descritto e lanciando l'algoritmo J48, il software restituisce il corrispondente l'albero di decisione, il numero di istanze correttamente ed erroneamente classificate, la matrice di confusione ed altri parametri che verranno successivamente descritti.

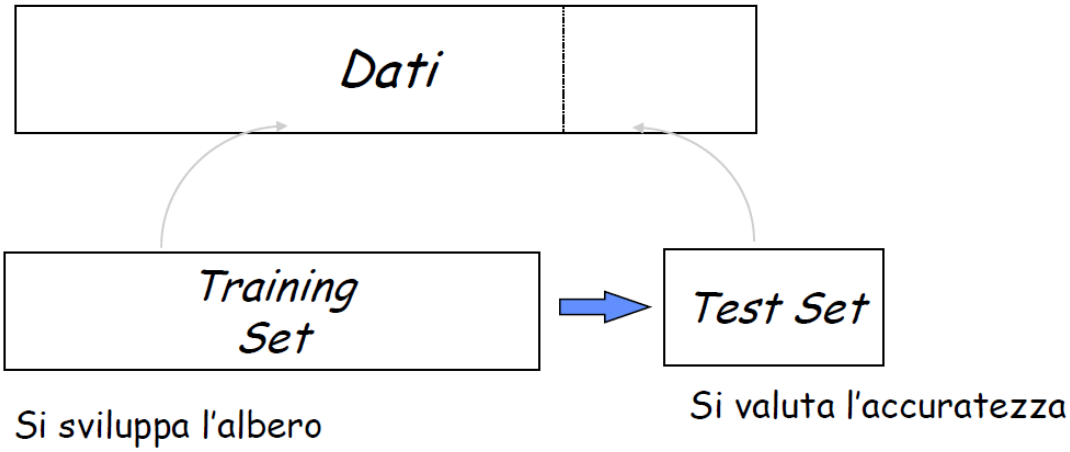
Prima di generare l'albero è però necessario scegliere il tipo di test (test options) che si intende effettuare, ovvero la tecnica che si desidera utilizzare per valutare l'efficacia di un modello. I vari tipi di test options sono elencati e spiegati di seguito:

- Use training set
- Cross-validation
- Percentage split

La **Cross-Validation** (in questo caso la k-fold Cross-validation) è un approccio che considera prima tutti i campioni, cioè l'insieme delle n osservazioni, per poi dividerli in k partizioni o folders della stessa dimensione. Nel caso dei dati in questione si considera k=10 e si effettua una suddivisione dei campioni in 10 folders. Ad ogni iterazione il primo forder serve per la validazione, mentre i restanti k-1 folder sono usati per l'addestramento (training).



Il **Percentage Split** permette di specificare la percentuale di dati del dataset che saranno usati per il training, ovvero per costruire il modello iniziale (in questo caso il 66%, che è anche il valore inserito di default da WEKA). Una volta costruito il modello, la rimanente percentuale di dati (33%), serviranno per il test, cioè per testare l'accuratezza del modello. In questo modo però, il numero di istanze sottoposte al test è nettamente inferiore rispetto alla Cross-Validation. Inoltre, nella k-fold crossvalidation (con K=10) l'algoritmo di apprendimento crea 10 blocchi diversi ed effettua il test su tutti e 10. Questo lo rende più affidabile dal punto di vista delle performance. Al contrario il percentage split esegue una singola valutazione sulle istanze dedicate al test restituendo un unico risultato.



Il **training set** effettua il test sullo stesso insieme di dati su cui si è addestrato e quindi il classificatore valuta quanto accuratamente riesce a predire la classe delle istanze su cui ha anche effettuato il training.

Nel corso dell'analisi ho utilizzato inizialmente la Cross-Validation per determinare l'accuratezza del modello e quindi l'errore di classificazione ed in un secondo momento, mantenendo gli stessi parametri impostati nel passaggio precedente, ho addestrato il decision tree sull'intero dataset set analizzando il rispettivo albero. In

questo modo la x-fold CV è usata per stimare l'errore di classificazione dell'albero costruito sull'intero dataset.

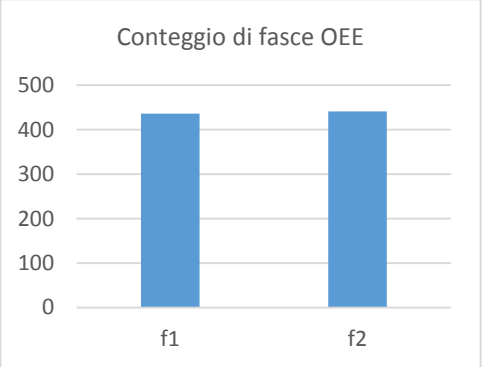
Un primo esperimento è stato svolto raggruppando i valori di OEE in tre fasce.
Il raggruppamento per frequenza e quindi l'individuazione dell'intervallo in modo da avere un numero simile di valori è stato ottenuto attraverso la funzione di Excel CONTA.SE.

Intervallo	*fascia*	*n° valori*
0< 0EE <0,648	f1	293
0,648<= OEE <0,739	f2	292,000
0,739<= OEE <1	f3	292

Ricorrendo alla Cross_validation ed impostando un confidence factor pari a 0,45 si ottiene il seguente risultato in termini di accuratezza di classificazione:

- Correctly Classified Instances 356 40.5929 %
- Incorrectly Classified Instances 521 59.4071 %

Poiché tale risultato risulta poco soddisfacente dal punto di vista dell'errore, ho omesso sia gli ulteriori parametri, sia la rappresentazione dell'albero ed ho optato per una soluzione che riducesse il numero di fasce da tre a due.



Intervallo	*fascia*	*n° valori*
0< 0EE <0,695	f1	436
0,695<= OEE <1	f2	441,000

Utilizzando la Cross-Validation il risultato migliore da un punto di vista dell'accuratezza di classificazione è quello che si ottiene ponendo un confidence factor pari a 0,10. Risulta infatti:

- Correctly Classified Instances 524 59.7491 %
- Incorrectly Classified Instances 353 40.2509 %

Oltre all'accuratezza di classificazione vengono forniti in output anche una serie di altri parametri di seguito riportati e spiegati:

Kappa statistic 0.1947
Mean absolute error 0.4573
Root mean squared error 0.5149

- *Kappa statistic*: è una misura della corrispondenza tra la classe attuale e la classe predetta. Si tratta di un parametro i cui valori possono variare tra 0 e 1. Valori più alti indicano una migliore performance del modello. Se K=0 significa che non c'è concordanza, se K=1 la concordanza è totale. Valori compresi nell'intervallo tra 0,40 a 0,59 si reputano discreti, quelli tra 0,60 e 0,79 sono notevoli ed infine quelli superiori a 0,80 sono eccezionali in quanto la concordanza è molto alta.

- *Mean absolute error*: è il rapporto tra la somma degli errori assoluti ed il numero di predizioni. Rappresenta un insieme di misure del valore previsto rispetto al valore reale e stabilisce così quanto un modello predetto si avvicina a quello reale.

Nell'esempio preso in esame il valore della Kappa statistic è molto basso in quanto è prossimo a 0,20, ben lontano dal valore di 0,40 considerato discreto nella spiegazione sopra riportata. Rappresenta tuttavia il valore migliore che è stato ottenuto mediante la Cross-Validation nel corso degli esperimenti con i dati forniti.

Il mean absolute error (MAE) che invece risulta prossimo a 0,5 è piuttosto alto dal momento che i valori ottimali dovrebbero essere vicini allo zero. Non è possibile tuttavia, nonostante i vari tentativi, avvicinarsi a tale valore a causa della presenza di un elevato numero di istanze non correttamente classificate e pertanto il modello realizzato si avvicina solo parzialmente a quello reale.

Altri parametri sono espressi nella tabella denominata “Detailed accuracy by class”

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,564	0,370	0,601	0,564	0,582	0,195	0,597	0,589	f1
	0,630	0,436	0,594	0,630	0,612	0,195	0,597	0,553	f2
Weighted Avg.	0,597	0,403	0,598	0,597	0,597	0,195	0,597	0,571	

TP Rate rappresenta la porzione di veri positivi, cioè le istanze correttamente classificate come una data classe, mentre *FP Rate* è la porzione di falsi positivi, istanze classificate in modo erroneo come appartenenti ad una determinata classe (le classi corrispondono con le fasce f1 e f2).

Gli altri parametri sono:

- *Precision*: rappresenta in numero di istanze realmente positive sul numero totale di istanze.

  $$Precision = \frac{TP}{(TP+FP)}$$

- *Recall*: è equivalente al TP poiché misura la porzione di istanze classificate in una determinata classe sul totale, cioè misura quanti veri positivi sono stati classificati correttamente.

  $$Recall = \frac{TP}{(TP + FN)}$$

Dalla tabella restituita dal software e sopra rappresentata è possibile notare che è presente un numero abbastanza consistente di falsi positivi, dal momento che l'FP rate è pari a 0,370 in riferimento alla classe f1 e 0,436 relativi alla classe f2. Per meglio analizzare questo aspetto è possibile fare affidamento sulla matrice di confusione che verrà descritta e commentata di seguito

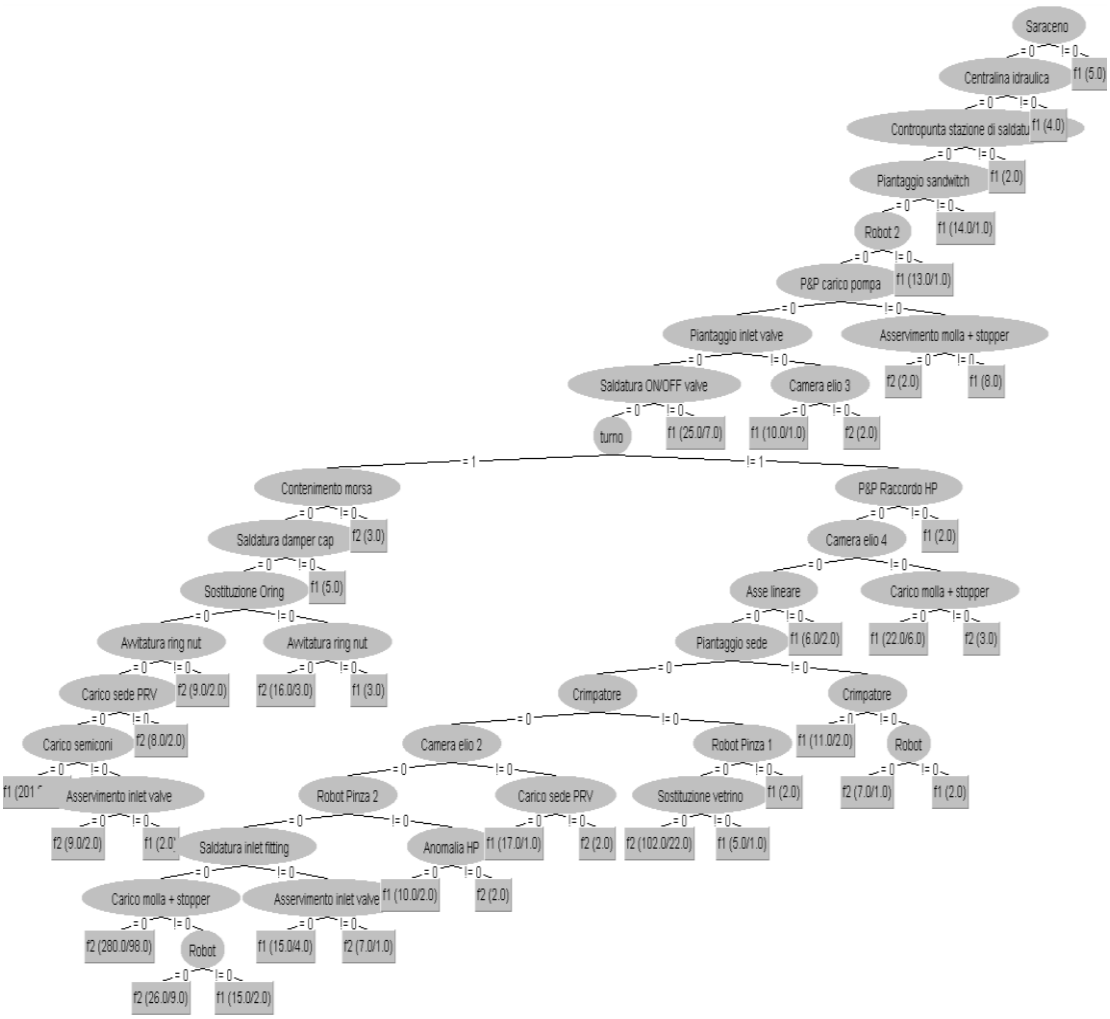
a b <-- classified as	
246 190 | a = f1	
163 278 | b = f2	

Tale matrice riporta sulla diagonale principale il numero di istanze correttamente classificate. Considerando infatti le colonne della matrice come la classe stimata e le righe come classe vera, gli elementi presenti sulla diagonale principale sono proprio quelli in cui la classe vera corrisponde con la classe stimata.
Le istanze presenti sull'altra diagonale rappresentano delle misclassificazioni (la classe vera non corrisponde con quella stimata) e vi sarà quindi un determinato errore di classificazione indicato con:

$$e = \frac{n°volte\ in\ cui\ sbaglio\ la\ stima}{totale\ campioni}$$

Quindi nel caso specifico ci sono 246 istanze classificate correttamente come f1, 278 istanze classificate correttamente come f2, mentre ciò che resta fuori è considerato errore di classificazione.
Come evidenziato, l'errore di classificazione risulta abbastanza elevato nonostante i vari tentativi svolti e le variazioni nel settaggio dei parametri. Il motivo di questo risiede probabilmente nella natura dei dati stessi e potrebbe essere legato ad esempio alla differenza delle frequenze con cui la maggior parte degli eventi di guasto si manifestano.

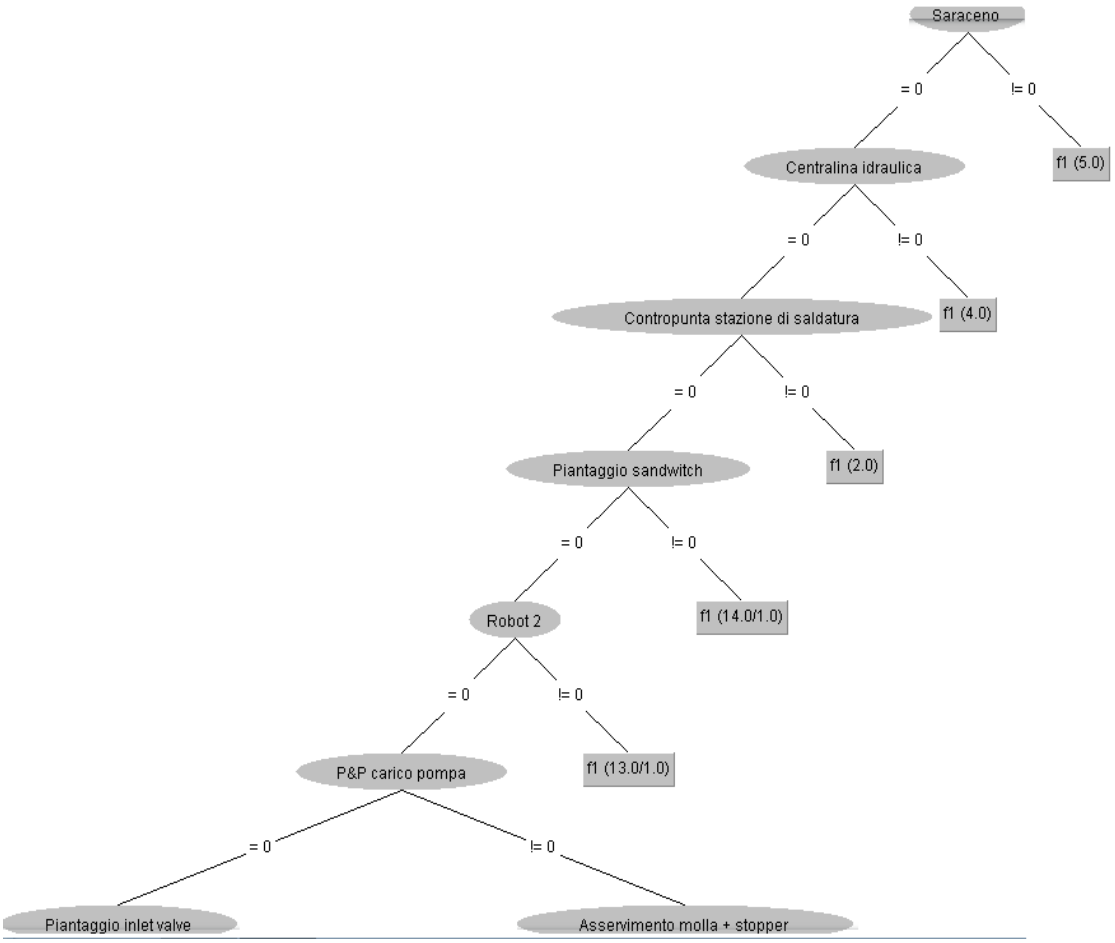
I parametri impostati per generare l'albero sono gli stessi utilizzati nella Cross-Validation e quindi il grado di potatura dell'albero, il confidence factor è pari a 0,10 e gli attributi (eventi di guasto e turno) sono di tipo "nominal" cioè categorici e quindi adatti per descrivere le due categorie, 1 se il guasto si verifica e 0 altrimenti. Il decision tree di seguito raffigurato garantisce un'accuratezza e quindi un errore di classificazione pari a quello calcolato per mezzo della Cross-Validation.



Si tratta di un albero di tipo binario in cui nell'effettuare lo splitting si valutano esclusivamente due condizioni, cioè (uguale a zero) se il guasto si verifica, (!=0, diverso da zero) se il guasto non si verifica. Ciò è ottenuto impostando su TRUE il parametro Binary splits.

A partire dal nodo radice (in questo caso costituito da “saraceno”) si effettua uno splitting. Se infatti tale evento di guasto si verifica (!=0), si procede con il ramo a destra che rimanda subito ad una foglia effettuando la classificazione, altrimenti se non si verifica (=0) si procede con il ramo sinistro che a sua volta conduce ad un diverso attributo, che effettuerà un nuovo splitting. Si segue questa logica fino a raggiungere tutte le foglie, in cui viene eseguita una classificazione. Viene effettuato quindi l’assegnamento della rispettiva fascia di OEE tra le due prese in considerazione e analizzate in precedenza (f1 e f2).

Per rendere più legibile l’esito ed individuare gli elementi che hanno un impatto maggiore da un punto di vista decisionale ho fatto uno zoom sui primi nodi dell’albero a partire dalla radice.



Come è possibile osservare, a partire dai primi nodi vengono eseguite immediatamente le prime classificazioni, in quanto gli attributi che contraddistinguono questi nodi sono maggiormente discriminanti, sebbene alcuni di essi (come ad esempio “contropunta stazione di saldatura”) abbiano una bassa frequenza di accadimento. La classificazione si dimostra coerente dato che nelle situazioni in cui tali eventi di guasto si verificano, (indicato per mezzo della notazione !=0, diverso da zero), viene assegnata la classificazione relativa alla fascia f1, il cui rispettivo intervallo di valori di OEE è compreso tra 0 e 0,695, secondo quanto esplicitato nella premessa.

A partire invece dal nodo “P&P carico pompa” l’albero si diversifica poiché, diversamente dai casi precedenti, anziché raggiungere subito una foglia con la relativa classificazione si raggiungono altri due nodi. Ciò implica che se “P&P carico pompa” si verifica allora si manifesta anche l’evento di guasto “Asservimento molla + stopper”, altrimenti si manifesterà il guasto relativo al “Piantaggio inlet valve”.

Di conseguenza al fine di massimizzare l’OEE, è possibile percorrere l’albero dalla radice fino alle foglie, cioè fino all’avvenuta classificazione, tenendo conto dell’errore di classificazione e che una eventuale incongruenza (ad esempio qualora venisse assegnata la fascia f2 al manifestarsi di un guasto), possa essere spiegata come la presenza di un guasto dal limitato o nullo impatto in termini di prestazioni della linea di assemblaggio.

In tutti i casi appena descritti tra gli attributi decisionali si è considerato, oltre alla totalità degli eventi di guasto, anche l'attributo riferito al turno. È tuttavia possibile creare alberi di decisione che includano tra gli attributi anche il giorno della settimana (da 1=lunedì fino a 7=domenica) e valutare come questo influisca sul risultato finale. Di seguito ne ho riportato i risultati ottenuti attraverso la Cross-validation ponendo il confidence factor pari a 0,12 ed ottenendo:

Correctly Classified Instances	509	58.0388 %
Incorrectly Classified Instances	368	41.9612 %

Ciò che si ottiene in termini di numero di istanze correttamente classificate è leggermente inferiore al primo albero analizzato, pur non discostandosi di molto.

Anche gli altri parametri forniti in output sono molto simili:

Kappa statistic	0.16
Mean absolute error	0.4579
Root mean squared error	0.5318

Anche in questo caso il valore della Kappa statistic è molto basso ed inferiore all'albero iniziale anche se non di molto. Risulta quindi ben lontano dal valore di 0,40 considerato discreto, pur restando il valore migliore che è stato ottenuto mediante la Cross-Validation nel corso degli esperimenti con i dati forniti considerando tra gli attributi anche il giorno della settimana.
Il mean absolute error (MAE) che invece è prossimo a 0,5 è piuttosto alto rispetto al valore ottimale che dovrebbe essere vicino allo zero. Non è possibile tuttavia, nonostante i vari tentativi, avvicinarsi a tale valore a causa della presenza di un elevato numero di istanze non correttamente classificate e pertanto il modello realizzato si avvicina solo parzialmente a quello reale.

Total Number of Instances	877

Il numero delle istanze analizzate attraverso la cross-validation è sempre pari ad 877.

```
=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0,500     0,340     0,592       0,500   0,542      0,162    0,580     0,576     f1
                 0,660     0,500     0,572       0,660   0,613      0,162    0,580     0,539     f2
Weighted Avg.    0,580     0,421     0,582       0,580   0,578      0,162    0,580     0,558
```

In questo caso la classe f2 mostra un numero di falsi positivi più elevato rispetto alla classe f1, probabilmente poiché corrisponde con la fascia caratterizzata da un range di valori è più limitato perché compresi tra 0,695 ed 1.

Matrice di confusione:

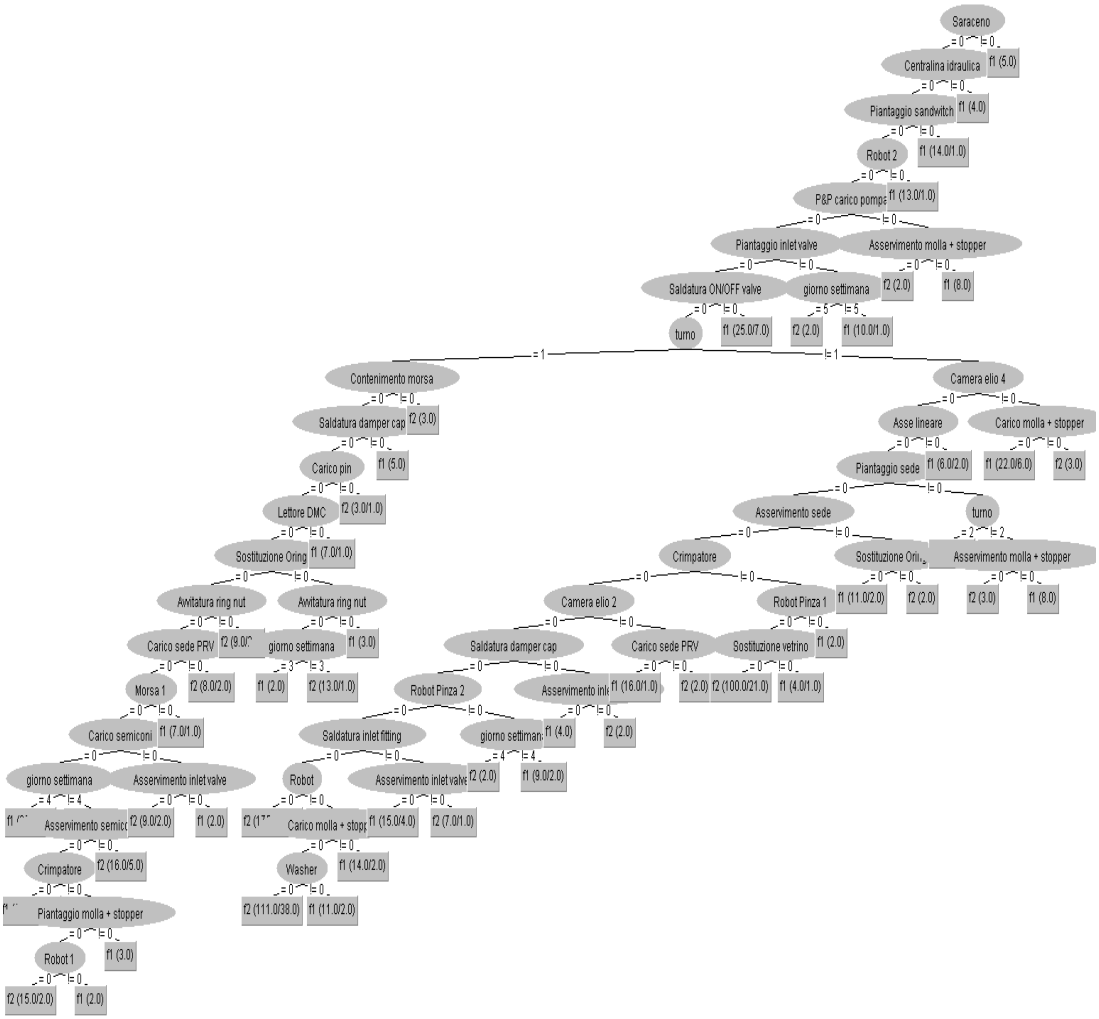
a b <-- classified as	
218 218 | a = f1	
150 291 | b = f2	

La matrice evidenzia che sono presenti 218 istanze classificate correttamente come f1, 291 istanze classificate correttamente come f2 e ciò che resta fuori è considerato errore di classificazione. In particolare si può notare un elevato numero di istanze avente classe vera f1 e classe stimata f2 e di conseguenza la misclassificazione è più marcata in tal senso.

Utilizzando il trainig set è ora possibile generare ed analizzare l'albero. Il decision tree prende come parametri di riferimento gli stessi impostati nella Cross-Validation e quindi un confidence factor pari a 0,12.

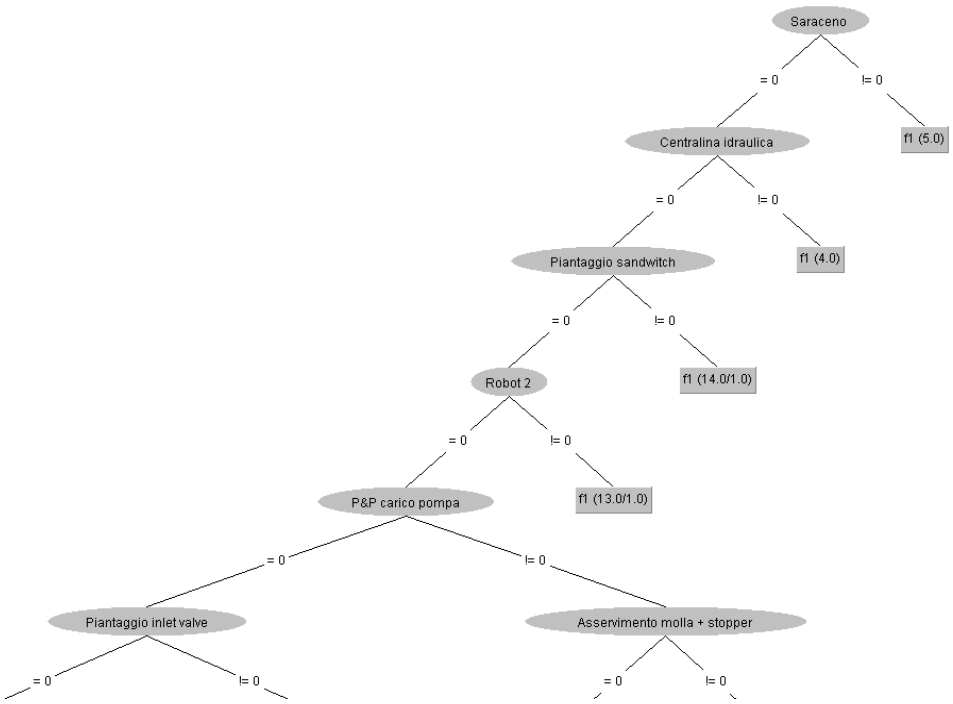
Prima di tutto è opportuno riportare l'albero completo, in un secondo momento verrà fatto un focus sui primi nodi per evidenziare gli attributi di maggior rilievo da un punto di vista della classificazione e valutare le eventuali differenze con l'albero

iniziale che escludeva l'attributo relativo al giorno della settimana. Infine vengono tratte le conclusioni. In modo analogo al caso precedente si ricorre al binary splits al fine di ottenere uno splitting di tipo binario.



L'albero presenta una disposizione dei nodi simile a quella analizzata in precedenza con l'aggiunta, in alcune sue parti, del nodo e del relativo splitting effettuato sul giorno della settimana, che tuttavia sembra non condizionare in modo significativo l'intera struttura.

Andando ad analizzare più in dettaglio i nodi principali si osserva:

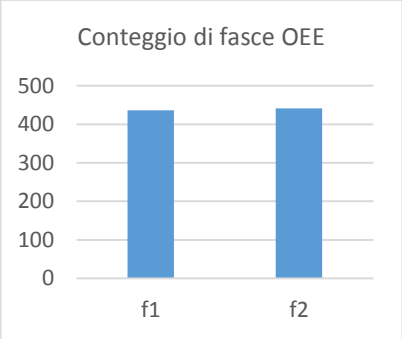


I primi nodi a partire da quello radice non presentano l'attributo riferito al giorno della settimana, il che fa pensare ad un ruolo più marginale ricoperto da questo attributo. Si possono osservare infatti per lo più gli stessi identici nodi e lo stesso criterio di classificazione presente nel primo albero (quello che escludeva l'attributo "giorno settimana").

La chiave di lettura dell'albero risulta pertanto la seguente: a partire dal nodo radice, che ha come attributo saraceno, si effettua lo splitting. Se il saraceno è soggetto a guasto si arriva immediatamente alla classificazione in base alla quale l'OEE è compreso tra i valori della fascia f1 (tra 0 e 0,695), altrimenti il nodo successivo su cui ricade la valutazione ed un successivo splitting è quello relativo all'attributo "centralina idraulica". Si procede così secondo un approccio di tipo top-down fino al raggiungimento di tutte le foglie.

4.3 Il Random Forest: applicazione

Fornendo in input in Weka il medesimo dataset e ricorrendo allo stesso tipo di suddivisione in fasce/classi dell'OEE è possibile applicare l'algoritmo Random Forest.



Intervallo	*fascia*	*n° valori*
0< 0EE <0,695	f1	436
0,695<= OEE <1	f2	441,000

Il risultato ottenuto è relativo alla situazione in cui tra gli attributi si comprendono la totalità degli eventi di guasto ed il turno, ma non il giorno della settimana. Il numero di alberi generati dall'algoritmo impostato con il parametro numIteration è pari a 150.

Considerando gli attributi di tipo nominal come nei precedenti esempi e ricorrendo alla cross-validation si ottiene:

Correctly Classified Instances	516	58.8369 %
Incorrectly Classified Instances	361	41.1631 %
Kappa statistic	0.176	
Mean absolute error	0.4501	
Root mean squared error	0.498	

Anche in questo caso il valore della Kappa statistic è molto basso e risulta confrontabile con quelli ottenuti attraverso l'algoritmo J48. Di conseguenza la corrispondenza tra classe attuale e classe predetta è bassa.

Un discorso analogo vale per il mean absolute error, caratterizzato da un valore elevato, laddove invece si sarebbe dovuto idealmente avvicinare allo zero.

Total Number of Instances 877

Matrice di confusione

a b <-- classified as	
222 214 | a = f1	
147 294 | b = f2	

La matrice evidenzia che sono presenti 222 istanze classificate correttamente come f1, 294 istanze classificate correttamente come f2 e ciò che resta fuori è considerato errore di classificazione.

I risultati ottenuti con il Random Forest, in termini di parametri restituiti, sono molto simili e quasi del tutto sovrapponibili con quelli dell'algoritmo j48, cosa che evidenzia che i problemi relativi all'errore di classificazione sono strettamente dipendenti dalla natura dei dati.

Tuttavia, a differenza del J48, il Random Forest non fornisce in output il grafico relativo all'albero dal momento che, anziché generare un singolo albero, ne genera una moltitudine in questo esempio 150, da cui appunto deriva il nome di "foresta".

4.4 Regole di associazione: applicazione

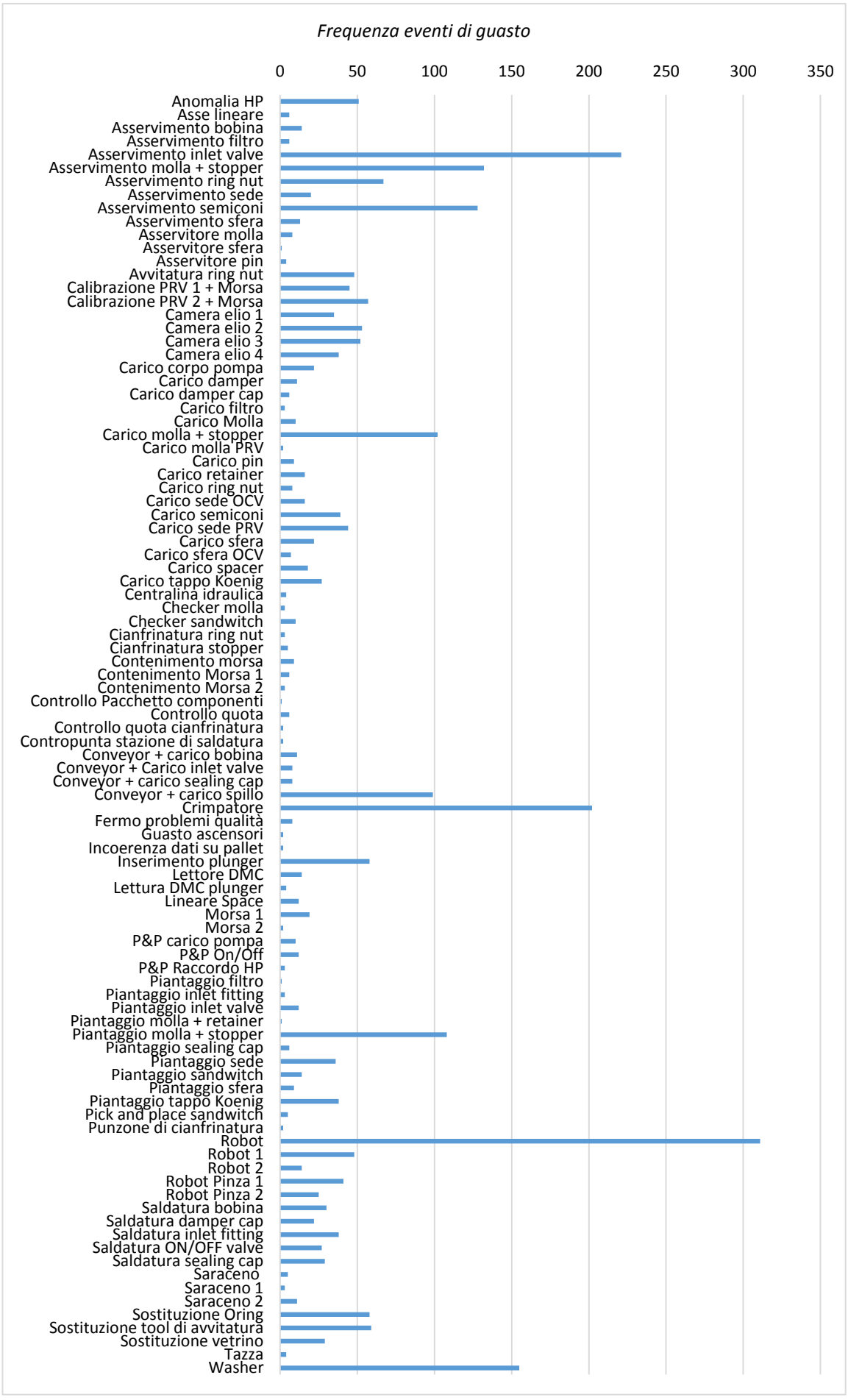
La scoperta delle association rules mediante l'algoritmo Apriori si presenta come un processo articolato in tre steps riportati di seguito:

- Step 1: scansione del database per la ricerca degli itemset frequenti. A tal fine si determina il supporto e si scartano quelli che non raggiungono il minimo valore indicato come soglia;

- Step 2: si generano tutti i sottoinsiemi dei frequent itemset e si trova il supporto per ognuno di essi;

- Step 3: a partire dagli itemset frequenti si estrapolano le regole e si calcola il valore della confidenza. Si selezionano infine le regole più forti, cioè quelle che superano il valore soglia per la confidenza.

Prima però di lanciare l'algoritmo ho eseguito in Excel un'analisi circa la frequenza con cui si manifestano i singoli guasti durante tutto il periodo considerato (gennaio 2019-giugno 2020).
Il risultato è illustrato nel grafico della pagina successiva e mostra che esistono alcuni eventi di guasto che si ripetono costantemente in tutto il periodo considerato con frequenza elevata, altri invece sono molto sporadici e si manifestano con minore frequenza. Nel complesso i guasti con frequenza particolarmente elevata sono in numero limitato, mentre molti di più sono quelli caratterizzati da bassa frequenza. Ciò naturalmente, nella determinazione delle association rules, andrà ad influenzare negativamente il parametro relativo al supporto.


Frequenza eventi di guasto
0
50
100
150
200
250
300
350
Anomalia HP
Asse lineare
Asservimento bobina
Asservimento filtro
Asservimento inlet valve
Asservimento molla + stopper
Asservimento ring nut
Asservimento sede
Asservimento semiconi
Asservimento sfera
Asservitore molla
Asservitore sfera
Asservitore pin
Avvitatura ring nut
Calibrazione PRV 1 + Morsa
Calibrazione PRV 2 + Morsa
Camera elio 1
Camera elio 2
Camera elio 3
Camera elio 4
Carico corpo pompa
Carico damper
Carico damper cap
Carico filtro
Carico Molla
Carico molla + stopper
Carico molla PRV
Carico pin
Carico retainer
Carico ring nut
Carico sede OCV
Carico semiconi
Carico sede PRV
Carico sfera
Carico sfera OCV
Carico spacer
Carico tappo Koenig
Centralina idraulica
Checker molla
Checker sandwitch
Cianfrinatura ring nut
Cianfrinatura stopper
Contenimento morsa
Contenimento Morsa 1
Contenimento Morsa 2
Controllo Pacchetto componenti
Controllo quota
Controllo quota cianfrinatura
Contropunta stazione di saldatura
Conveyor + carico bobina
Conveyor + Carico inlet valve
Conveyor + carico sealing cap
Conveyor + carico spillo
Crimpatore
Fermo problemi qualità
Guasto ascensori
Incoerenza dati su pallet
Inserimento plunger
Lettore DMC
Lettura DMC plunger
Lineare Space
Morsa 1
Morsa 2
P&P carico pompa
P&P On/Off
P&P Raccordo HP
Piantaggio filtro
Piantaggio inlet fitting
Piantaggio inlet valve
Piantaggio molla + retainer
Piantaggio molla + stopper
Piantaggio sealing cap
Piantaggio sede
Piantaggio sandwitch
Piantaggio sfera
Piantaggio tappo Koenig
Pick and place sandwitch
Punzone di cianfrinatura
Robot
Robot 1
Robot 2
Robot Pinza 1
Robot Pinza 2
Saldatura bobina
Saldatura damper cap
Saldatura inlet fitting
Saldatura ON/OFF valve
Saldatura sealing cap
Saraceno
Saraceno 1
Saraceno 2
Sostituzione Oring
Sostituzione tool di avvitatura
Sostituzione vetrino
Tazza
Washer

Per poter valutare la validità delle regole di associazione è necessario tenere conto dei due parametri fondamentali introdotti nel paragrafo 2.4: la confidenza ed il supporto.

In WEKA il supporto minimo viene espresso tra un upperBound ed un lowerBound. Affinché una regola di associazione possa ritenersi valida ed utilizzabile per trarne delle conclusioni affidabili, dovrebbe avere un supporto superiore almeno a 0,60 (cioè il 60% delle istanze). Ciò tuttavia non si verifica nel caso dei dati in questione. Ho lanciato l'algoritmo Apriori prendendo in considerazione la totalità degli attributi (inclusi il turno e il giorno della settimana) e settando su TRUE il campo "treat zero as missing", in modo da interpretare gli zeri, la cui numerosità è elevata, come dei valori mancanti al fine di ricavare dei risultati significativi.
Impostando il supporto minimo ad un valore di 0,6 l'algoritmo non estrapola nessuna regola di associazione restituendo la dicitura: "*No large itemsets and rules found!".* Questo scenario si continua a ripresentare fino a quando non si imposta un valore di supporto minimo inferiore o pari a 0,02 con confidenza ≥ 0,7.
Le regole ricavate quindi si verificano solo meno del 2% dei casi, il che implica che non hanno sufficiente rilevanza statistica, non è opportuno comunicarle al decisore e pertanto verranno omesse dalla trattazione.

5. Conclusioni

Dall'output della classificazione si evince che i risultati forniti dai due algoritmi (J48 e Random Forest) sono quasi del tutto sovrapponibili sia in termini di percentuale di istanze correttamente classificate, sia in riferimento agli altri parametri restituiti. Il numero di "Correctly Classified Instances" infatti, raggiunge un massimo di 524 su un totale di 877 istanze prese in esame, con un conseguente errore di classificazione che si attesta al di sopra del 40%. Si tratta di un errore piuttosto alto, non migliorabile neppure modificando in Weka il setting dei parametri. La situazione resta pressoché la stessa, sia modificando il numero di attributi, aggiungendone uno ulteriore come il giorno della settimana oltre ai guasti ed al turno, sia aumentando o diminuendo il numero di istanze.

Dal momento che tale errore di classificazione viene confermato anche implementando il Random Forest, si può dedurre che sia strettamente legato alla natura dei dati oppure all'attenzione di chi ha rilevato gli eventi di guasto e li ha riportati nel file "Data Guasti".

Nonostante ciò, gli alberi di decisione ottenuti propongono comunque una classificazione per lo più coerente da un punto di vista logico. Questo è dimostrabile in base al fatto che viene assegnata l'etichetta relativa alla fascia/classe f1 (a cui corrispondono valori più bassi di OEE e quindi prestazioni inferiori) in corrispondenza di quei rami che esplicitano il verificarsi del generico evento di guasto, mentre assegnano la fascia f2 (i cui valori di OEE sono inclusi in un range tra 0,695 ed 1), quando il guasto non si verifica e l'attributo assume un valore pari a zero.

Nell'applicare l'algoritmo Apriori al medesimo dataset per estrarre regole di associazione che siano significative dal punto di vista dell'informazione ottenuta, è necessario prendere in esame soltanto i valori del dataset diversi da zero, cioè quelli corrispondenti al verificarsi degli eventi di guasto. Tuttavia, a causa della limitata frequenza con cui si manifesta la maggior parte dei guasti, le regole estrapolate

sono caratterizzate da un supporto che raggiunge al massimo il valore di 0,02. Dal momento che a tale valore corrisponde un numero di istanze pari a 18 su un totale di 877, non vi è una sufficiente rilevanza statistica per stabilire la validità delle regole. Incrementando il supporto e portandolo ad un valore di 0,6, l'algoritmo non ricava invece nessun itemset frequente per cui non è possibile estrarre nessuna regola di associazione che abbia un supporto adeguato.

6. Bibliografia e sitografia

Zhu, Lin, Dafeng Qiu, Daji Ergu, Cai Ying, and Kuiyi Liu. “A Study on Predicting Loan Default Based on the Random Forest Algorithm.” *Procedia Computer Science* 162 (2019): 503–13. https://doi.org/10.1016/j.procs.2019.12.017.

Yao, Han, Xiaodong Li, Huaji Pang, Lifang Sheng, and Wencai Wang. “Application of Random Forest Algorithm in Hail Forecasting over Shandong Peninsula.” *Atmospheric Research* 244 (November 2020): 105093. https://doi.org/10.1016/j.atmosres.2020.105093.

Yan, Hang, Nan Yang, Yi Peng, and Yitian Ren. “Data Mining in the Construction Industry: Present Status, Opportunities, and Future Trends.” *Automation in Construction* 119 (November 2020): 103331. https://doi.org/10.1016/j.autcon.2020.103331.

Singh, Sudhakar, Rakhi Garg, and P K Mishra. “Performance Optimization of MapReduce-Based Apriori Algorithm on Hadoop Cluster.” *Computers & Electrical Engineering* 67 (April 2018): 348–64. https://doi.org/10.1016/j.compeleceng.2017.10.008.

Mantas, Carlos J., and Joaquín Abellán. “Credal-C4.5: Decision Tree Based on Imprecise Probabilities to Classify Noisy Data.” *Expert Systems with Applications* 41, no. 10 (August 2014): 4625–37. https://doi.org/10.1016/j.eswa.2014.01.017.

Lee, Changyong, Bomi Song, and Yongtae Park. “Design of Convergent Product Concepts Based on Functionality: An Association Rule Mining and Decision Tree Approach.” *Expert Systems with Applications* 39, no. 10 (August 2012): 9534–42. https://doi.org/10.1016/j.eswa.2012.02.099.

Klösgen, Willi, and Jan M. Żytkow, eds. *Handbook of Data Mining and Knowledge Discovery*. Oxford ; New York: Oxford University Press, 2002.

Kavakiotis, Ioannis, Olga Tsave, Athanasios Salifoglou, Nicos Maglaveras, Ioannis Vlahavas, and Ioanna Chouvarda. “Machine Learning and Data Mining Methods in Diabetes Research.” *Computational and Structural Biotechnology Journal* 15 (2017): 104–16. https://doi.org/10.1016/j.csbj.2016.12.005.

Hong, Jungyeol, Reuben Tamakloe, and Dongjoo Park. “Application of Association Rules Mining Algorithm for Hazardous Materials Transportation Crashes on Expressway.” *Accident Analysis & Prevention* 142 (July 2020): 105497. https://doi.org/10.1016/j.aap.2020.105497.

Gamarra, Carlos, Josep M. Guerrero, and Eduardo Montero. “A Knowledge Discovery in Databases Approach for Industrial Microgrid Planning.” *Renewable and Sustainable Energy Reviews* 60 (July 2016): 615–30. https://doi.org/10.1016/j.rser.2016.01.091.

Frank, E., M. Hall, L. Trigg, G. Holmes, and I. H. Witten. “Data Mining in Bioinformatics Using Weka.” *Bioinformatics* 20, no. 15 (October 12, 2004): 2479–81. https://doi.org/10.1093/bioinformatics/bth261.

Bhandari, Akshita, Ashutosh Gupta, and Debasis Das. “Improvised Apriori Algorithm Using Frequent Pattern Tree for Real Time Applications in Data Mining.” *Procedia Computer Science* 46 (2015): 644–51. https://doi.org/10.1016/j.procs.2015.02.115.

Agany, Diing D.M., Jose E. Pietri, and Etienne Z. Gnimpieba. “Assessment of Vector-Host-Pathogen Relationships Using Data Mining and Machine Learning.” *Computational and Structural Biotechnology Journal* 18 (2020): 1704–21. https://doi.org/10.1016/j.csbj.2020.06.031.

Aflori, Cristian, and Mitica Craus. “Grid Implementation of the Apriori Algorithm.” *Advances in Engineering Software* 38, no. 5 (May 2007): 295–300. https://doi.org/10.1016/j.advengsoft.2006.08.011.

http://www.cs.waikato.ac.nz/ml/weka/book.html

Atzeni, Ceri, Fraternali, Paraboschi, Torlone.“Basi di dati”. V Edizione, McGraw Hill.

Slide del corso di Tecnologie per i sistemi informativi, Univpm

Wirth, Rüdiger, and Jochen Hipp. “CRISP-DM: Towards a Standard Process Model for Data Mining.” In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 29–39. Springer-Verlag London, UK, 2000.

Drazin, Sam, and Matt Montag. “Decision Tree Analysis Using Weka.” *Machine Learning-Project II, University of Miami*, 2012, 1–3.

Frank, Eibe, Mark Hall, Geoffrey Holmes, Richard Kirkby, Bernhard Pfahringer, Ian H. Witten, and Len Trigg. “Weka-a Machine Learning Workbench for Data Mining.” In *Data Mining and Knowledge Discovery Handbook*, 1269–1277. Springer, 2009.