Analisi del più comune tra i problemi della *supply chain*: i "*backorder*"

Daniele Antonio Adduci, Silvia Bordogna, Davide Brinati, Stefano Daraio, Davide Pecchia

Abstract
Spesso capita di voler effettuare un acquisto di qualunque genere e, dopo aver deciso cosa comprare, di notare la scritta, molto prossima al tasto 'acquista', che ci informa: 'pezzi rimasti: 1' oppure 'articolo in esaurimento'. Verificandosi frequentemente, soprattutto con oggetti molto richiesti, in alcuni casi i venditori permettono ai loro clienti di ordinare quantità superiori alle loro disponibilità di magazzino, garantendo poi di spedire la restante merce una volta ottenuta dal fornitore. La quantità ordinata che sarà disponibile successivamente è definita dal termine inglese di "*back-order*", ovvero un ordine effettuato ma non ancora disponibile per essere evaso. Le possibili problematiche che conseguono a questo tipo di pratica sono molteplici e di diversa natura, ed è per questo che per un'azienda potrebbe risultare molto utile sapere, in anticipo, se un certo bene in futuro andrà in "*back-order*" (con una certa probabilità) o meno. Utilizzando un dataset su questo argomento, abbiamo prima calcolato delle statistiche esplorative per approfondire la conoscenza del fenomeno; dopodichè abbiamo utilizzato diverse tecniche di *feature selection* per eliminare gli attributi non necessari; dopo aver estratto dei campioni per ridurre la dimensionalità del dataset, si è svolta la cross validation di vari modelli, in modo tale da poter estrapolare quelli migliori; questi sono poi stati applicati al test set e, una volta individuata la tecnica migliore per selezionare gli attributi, sono stati validati una seconda volta per ottenere il modello più efficace.

Indice

Introduzione

I processi aziendali che permettono alle imprese di svolgere la loro attività operativa sono molteplici ed hanno tutti un'elevata importanza dato che, trascurandone uno, si potrebbero verificare problemi in grado di compromettere l'operatività. Tra le attività più importanti e cruciali a cui un'azienda deve prestare particolare attenzione troviamo i rapporti con i fornitori e la conseguente gestione del magazzino (punto chiave soprattutto per le imprese di produzione e per le nuove piattaforme di e-commerce). Gestione che dovrebbe essere quanto più ottimizzata possibile al fine di minimizzare i costi ed evitare possibili problematiche con i clienti. Rendere e mantenere efficiente un magazzino non è un compito facile: sicuramente su ciò influiscono molti fattori quali la dimensione dell'impresa, il mercato di riferimento, la tipologia di beni trattati ecc...
Dunque essere consapevoli dei prodotti a disposizione e delle quantità è di fondamentale importanza. Perchè ci sia sempre disponibilità di merce all'interno del magazzino, tendenzialmente le aziende prendono misure preventive affinchè non si trovino improvvisamente (ma soprattutto inconsapevolmente) senza uno o più beni di cui necessitano o per evadere un ordine effettuato da un cliente, oppure necessari per effettuare lavorazioni per la produzione di articoli poi venduti dall'azienda stessa. Per questo le tempistiche degli ordini da effettuare vanno valutate con attenzione, soprattutto nel caso in cui le materie od i prodotti di cui si necessita non siano temporaneamente disponibili presso il venditore; o ancora, non si è in possesso della quantità necessaria di merci per far fronte all'ordine di un cliente.
Analizzando il problema da questo lato, se il cliente finale dovesse ordinare una certa quantità di beni e l'azienda non ne avesse in magazzino il numero esatto per fronteggiare esaurientemente l'ordine, la differenza tra la quantità presente a magazzino e quella ordinata andrà in "*back-order*", ovvero diviene

un ordine che verrà evaso dall'impresa solo quando ci sarà la disponibilità dei pezzi. Per fornire un esempio molto semplice: se una persona dovesse comprare 5 telefoni ma il venditore ne avesse a disposizione solo 3 'in-stock' (in magazzino) pronti per la spedizione, i 2 restanti saranno "*back-order*" e verranno recapitati in seguito al cliente. Questa pratica potrebbe portare svariati problemi: il compratore potrebbe non voler aspettare più tempo per la ricezione della merce e dunque cambiare venditore; la reputazione dell'azienda verrebbe intaccata nel caso in cui ci fossero dei ritardi nella consegna dei beni preordinati e non ancora ricevuti. Sapere dunque se un prodotto andrà o meno in "*back-order*" è di fondamentale importanza per un'impresa ed è il problema che si è affrontato in questo elaborato. Si è cercato infatti, attraverso i dati a disposizione, di prevedere questo fenomeno, analizzando, tra le altre cose, per ogni prodotto: le quantità a magazzino, i dati sulle vendite e le previsioni di vendita per periodi di tempo differenti.
La soddisfazione del cliente è l'obiettivo principe di ogni impresa, che di conseguenza è sempre alla ricerca di metodologie differenti e nuove per l'ottimizzazione dei propri processi ed in grado di affrontare le possibili criticità, tra le quali, appunto, la gestione dei "*back-order*".[1]

1. Dati

1.1 Descrizione dei dataset

L'obiettivo della ricerca è quello di riuscire a prevedere se un'articolo in vendita andrà o meno in "*back-order*". I dati a nostra disposizione sono stati reperiti sulla piattaforma Kaggle (www.kaggle.com), da cui abbiamo scaricato 2 dataset differenti: uno di training ed uno di test per effettuare la finale validazione dei modelli risultati migliori dalle analisi svolte.[2]

1.2 Analisi esplorativa

Dalla visione ed analisi delle statistiche esplorative è emerso subito un forte problema di *class imbalance*, ovvero la variabile target (nel nostro caso una variabile binaria denominata "*went on backorder*") assumeva per la maggior parte il valore "*No*" (nel 99,33% dei casi) mentre la classe minoritaria, il valore "*Yes*", compariva solo nello 0,67% dei casi.

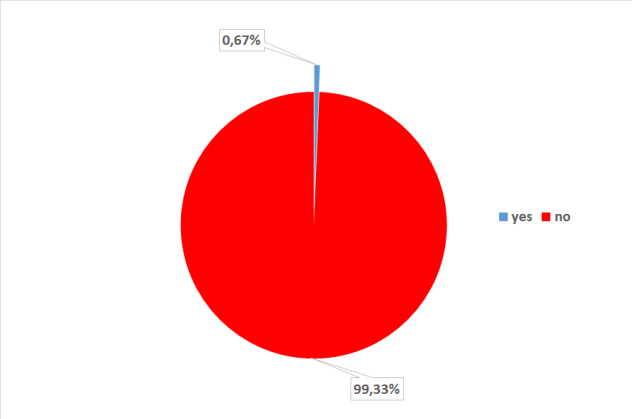

Figure 1. Rapporto tra "*Yes*" e "*No*" all'interno del dataset

Per far fronte a questo problema saranno proposte in seguito diverse metodologie.
Abbiamo poi analizzato le variabili presenti nel dataset per avere una miglior comprensione del fenomeno e capire se fossero necessari interventi nella successiva fase di preprocessing.
É emerso che non fossero necessari interventi per la gestione degli outliers in quanto erano irrilevanti rispetto alla mole di dati analizzata; inoltre non si è ritenuto necessario svolgere processi di trasformazione delle variabili data la loro natura e distribuzione.

2. Preprocessing

Il dataset di training è costituito da 1 687 860 osservazioni e 23 colonne che vanno a definire importanti indicatori di vendita per ogni articolo gestito a magazzino; questi sono : "*national inv*", "*lead time*", "*in transit qty*", "*forecast 3 month*", "*forecast 6 month*", "*forecast 9 month*", "*sales 1 month*", "*sales 3 month*", "*sales 6 month*", "*sales 9 month*", "*min bank*", "*potential issue*", "*pieces past due*", "*perf 6 month avg*", "*perf 12 month avg*", "*local bo qty*", "*deck risk*", "*oe constraint*", "*ppap risk*", "*stop auto buy*", "*rev stop*", "*went on backorder*", "*sku*".
Di seguito una breve descrizione delle variabili il cui significato non è evidente dalla nomenclatura utilizzata: "*national inv*" rappresenta l'attuale livello delle scorte in magazzino del prodotto, il "*lead time*" è il tempo di transito necessario per la spedizione, espresso in settimane, "*in transit qty*" è la quantità di prodotto in transito dal fornitore originario, "*min bank*" è la quantità minima raccomandata da tenere in magazzino, "*pieces past due*" indica gli ordini arretrati per l'oggetto, "*local bo qty*" indica la quantità di ordini relativi a scorte scadute. Sono poi presenti una serie di variabili binarie : "*potential issue*" contrassegna i prodotti per cui c'è un problema di identificazione delle origini mentre "*deck risk*", "*oe constraint*", "*ppap risk*", "*stop auto buy*","*rev stop*" delineano dei fattori di rischio. La variabile "*sku*" è stata rimossa in quanto trattasi della chiave primaria della nostra tabella e quindi non porta alcun valore aggiunto nell'interpretazione della variabile target (ma anzi crea solo problematiche in termini di performance computazionali e nella fase di analisi).

2.1 Pulizia dei dati

Tramite l'analisi esplorativa dei dati è emersa la presenza di valori mancanti per alcune variabili dipendenti; questi valori sono stati imputati originariamente nel dataset utilizzando due metodologie differenti

- Osservazioni valorizzate con "*Null*"
- Osservazioni valorizzate con valori numerici diversi (-99)

Individuati tali valori mancanti sono stati tutti uniformati come "*Null*"; abbiamo quindi proceduto con la valorizzazione di quest'ultimi andandoli a sostituire utilizzando il metodo della regressione lineare.

La prima metodologia relativa ai valori mancanti riguarda la variabile "*lead time*"; l'individuazione di questa casistica è evidente dal calcolo delle statistiche esplorative in quanto questi valori sono rappresentati nei nostri dati come "*Null*" e riconosciuti da Knime come "*missing values*".
Per le variabili "*perf 6 month avg*" e "*perf 12 month avg*" invece i valori missing sono presenti nel dataset iniziale con il valore numerico "-99". Considerando che il range numerico di queste due variabili va da 0 ad 1, questi valori sono stati considerati come *missing values* da Knime impostando il pattern corrispondente nella fase iniziale di caricamento del dataset.
Abbiamo considerato la possibile diversa natura di questa differenza inputazionale, e crediamo che questo possa essere dovuto ad una diversa fonte dei dati alla base della costruzione del dataset oppure ad una diversa interpretazione del motivo per cui non è presente tale valore (ad esempio per motivi legati all'impossibilità di calcolare tale valore nelle variabili di performance, che quindi denoterebbe un interpretazione diversa). Al fine però di uniformare tali valori e procedere con l'analisi abbiamo deciso di renderli tutti "*Null*".

2.2 Missing Replacement

Come ultima fase abbiamo quindi proceduto con la sostituzione dei *missing values*.
Considerando che la nostra variabile target è fortemente sbilanciata (i valori "*Yes*" sono meno dell' 1%) abbiamo optato per approcci diversi a seconda dei valori da essa assunta.
In caso fosse presente il valore dominante "*No*" abbiamo deciso di eliminare le righe che presentino valori missing, considerando l'ampiezza del campione e il fatto che le righe eliminate sarebbero circa l'8% del totale.
Al fine di preservare il maggior numero possibile di osservazioni per i valori "*Yes*" della variabile target, abbiamo deciso di predire il valore più probabile dove non presente. Per fare questo abbiamo utilizzato il modello di regressione lineare multivariato, applicando il learner sulla porzione di dataset non avente valori mancanti collegato ad un predittore che andrà ad imputare i valori previsti sull'altra porzione. Questa operazione è stata fatta per ognuna delle 3 variabili che presentavano valori mancanti.
Rispetto al dataset iniziale abbiamo quindi individuato e uniformato la codifica per i valori dei *missing values*; abbiamo quindi eliminato le righe aventi almeno un valore mancante dove questo non portava ad una perdita in termini di valore informazionale, essendo il dataset molto ampio (in corrispondenza della variabile target "*No*"), mentre per la variabile target "*Yes*", avente molte meno osservazioni, abbiamo usato un modello per predire i valori mancanti.

3. Metodologie e problemi

Conclusa la fase di gestione dei valori mancanti abbiamo realizzato tre flussi per verificare se l'eliminazione di alcuni attributi potesse migliorare le performance dei modelli scelti e alleggerire il carico computazionale.

Vista la numerosità del training set (1 687 859 record totali, 1 558 812 al netto della procedura di *missing replacement*), abbiamo scelto di allenare i modelli scelti utilizzando un campione di esso. Tuttavia, il dataset risulta molto sbilanciato per quanto riguarda l'attributo di classe "*went on backorder*" , per tale ragione è stato necessario effettuare un oversampling prima di procedere oltre.

3.1 Riduzione dimensionalità

Le tecniche utilizzate per ridurre la dimensionalità del dataset sono *Attribute Selection* e *Linear Correlation*, naturalmente, per valutare gli eventuali benefici derivanti da tale riduzione, i modelli sono stati allenati anche senza alcun intervento.
Il metanodo dedicato alla *Attribute Selection* è composto da due *AttributeSelectedClassifier*: il primo è stato configurato al fine di individuare gli attributi fondamentali. L'algoritmo di ricerca scelto è il *Greedy Stepwise* le cui performance sono valutate con la *Correlation Feature Selection* (Cfs); la validazione avviene usando il modello *Random Forest* come classificatore. Il risultato è un sottoinsieme di attributi composto da "*forecast 3 month*", "*pieces past due*" e "*local bo quantity*". Nel secondo nodo è stato utilizzato nuovamente il modello "*Random Forest*", selezionando gli attributi tramite *InfoGain* ed utilizzando il criterio di ranking per ordinarli ed ottenere l'output. Sulla base dei dati così ottenuti abbiamo deciso di conservare i tre campi calcolati dal primo nodo in aggiunta ai più significativi ottenuti dal secondo:

- *perf 6 month avg*
- *lead time*
- *national inv*
- *forecast 6 month*
- *pieces past due*
- *local bo qty*
- *went on backorder*

Per quanto riguarda l'analisi di *Linear Correlation*, per ciascun attributo numerico è stata calcolata la correlazione con gli altri e nel caso in cui fosse superiore al valore di soglia del 90%, è stato eliminato. Seguendo tale criterio sono stati esclusi i seguenti attributi:

- *perf 6 month avg*
- *forecast 3 month*
- *forecast 6 month*
- *sales 3 month*
- *sales 6 month*
- *sales 9 month*
- *min bank*

3.2 Class imbalance

Come accennato in precedenza, il dataset di training è affetto da un evidente problema di *class imbalance*: la classe positiva conta infatti soltanto 11 293 record contro 1 547 519 della classe negativa (appena lo 0,67% del totale).
Allenare i modelli senza applicare tecniche di oversampling può portare al paradosso di ottenere valori molto alti di *Accuracy*, ma scarsi per quanto riguarda *Precision* e *Recall*, metriche di gran lunga più interessanti in contesti di rarità della classe positiva. Per ovviare al problema, il dataset è stato diviso in due partizioni contenenti rispettivamente i record con attributo target positivo e negativo. Tramite *bootstrapping*, abbiamo triplicato i record identificati con "*Yes*" e della restante parte è stato selezionato solo il 5% delle osservazioni. Il risultato è dunque composto da 111 254 righe (77 375 con "*went on backorder*" = "*No*" e 33 879 con "*Yes*"), con all'incirca una riga con attributo positivo ogni tre (30,45%).[3][4]

3.3 Modelli utilizzati

Dopo aver ridotto la dimesionalità del dataset tramite sampling, sono stati scelti 8 modelli da testare tramite cross validation, al fine di determinare i migliori da destinare poi alla validazione sul test set. I modelli scelti sono:

- *J48*: modello euristico di Weka
- *Random Forest*: modello euristico di Weka, settato con numero di alberi = 10
- *KNN*: K Nearest Neighbor; modello euristico basato su alberi. Come parametro abbiamo inserito 3 *nearest neighbors* che il modello dovrà considerare nella classificazione
- *AdaBoostM1*: é un modello utilizzato per migliorare le performance di un determinato classificatore. Abbiamo utilizzato nuovamente la *Random Forest*, inserendo il numero di iterazioni = 10 ed evitando il re-sampling.
- *Logistic*: modello di Weka basato sulla regressione.
- *MLP*: Neural Network; modello separativo di Weka. Abbiamo settato il numero di *hidden layers* pari a 3, impostato la dimensione del set pari a 30 ed il treshold a 20.
- *Naïve Bayes*: modello probabilistico.
- *Naïve Bayes (Weka)*: modello probabilistico di Weka. Abbiamo settato i paramentri in modo tale che venisse utilizzata la *supervised discretization*

Tutti i modelli sono stati reiterati per 10 volte durante la cross validation e a tutti è stato impostato un *random seed* comune per poter permettere i confronti.

4. Analisi preliminare dei modelli

L'obiettivo della fase di analisi preliminare è stato quello di testare un numero consistente di combinazioni di tecniche di classificazione e subset di variabili differenti. Ciascuno degli algoritmi di *learning* elencati al punto 3.1 è stato allenato per ciascuna metodologia di *feature selection*, per un totale di 16 modelli di classificazione. Si noti che ciascuno modello è stato allenato anche direttamente sui dati ricampionati, senza dunque alcun intervento sulla selezione delle variabili, per testare in tal modo l'eventuale incremento apportato dai nodi di *feature selection*.
Una prima versione dell'analisi era stata disegnata per includere nelle combinazioni un'altra tecnica di ricampionamento nota come *Synthetic Minority Over-sampling Technique*. Tale via è stata abbandonata per ragioni di insufficienza di potere computazionale data l'ampiezza del dataset. Un'ulteriore tecnica di *feature selection*, ovvero *Principal Component Analysis*, ha invece inizialmente preso parte a questa fase, non avendo apportato però risultati utili, si è scelto successivamente di rimuovere la tecnica dal flusso. In virtù dei risultati ottenuti si è valutato che una tecnica che richiedesse la standardizzazione degli attributi non fosse adatta per molte delle variabili contenute in questo dataset, asimmetriche e con alta dispersione.
Si tenga presente che il modello di Regressione logistica è stato volontariamente introdotto come baseline; si è ritenuto infatti interessante conoscere le prestazioni di un modello semplice, per poter valutare la reale necessità di modelli più complessi a livello computazionale ed interpretativo.
Le metriche utilizzate per comparare i modelli di classificazione ottenuti sono quelle derivanti dalla *Confusion Matrix*, ovvero *Recall*, *Precision* e *F-Measure* oltre che alle curve ROC e relativo indice *AUC*. Nelle tabelle a seguire non si è riportato il valore di *Accuracy*, sapendo che nei casi di *class imbalance* come quello in cui ci troviamo, il suo valore è del tutto fuorviante. Essa infatti assumerebbe valori prossimi a uno anche qualora un learner ZeroR predicesse tutti i dati del test con il valore della classe maggioritaria.
Infine si è ritenuto ragionevole considerare il mancato riconoscimento di un "*back-order*" più oneroso e grave per l'economia dell'azienda rispetto alla previsione di un falso positivo. Per questo motivo si è stabilito di prediligere modelli con *Recall* più alta, indipendentemente dal livello di *Precision*, per favorire il massimo livello di riconoscimento della classe minore. Per questo i valori secondo cui verranno selezionati i modelli più promettenti di fatto saranno *Recall* e *AUC*.

4.1 Cross Validation

Come tecnica per selezionare il learner ed individuare il modello con le migliori performance predittive attese, si è scelto di utilizzare la tecnica di Cross Validation con 10 *folds* con random seed pari a 12345, per garantirne la replica identica. Considerando la natura dei dati, la presenza di *class imbalance* e l'alta dispersione degli attributi predittivi, si è ritenuto importante utilizzare una metodologia che potesse portare a

risultati più robusti, ovvero il più possibile indipendenti da outliers e da variazioni del dataset di training e test. Utilizzare un metodo computazionalmente meno oneroso quale *Hold Out* o *Iterated Hold Out* ci farebbe incorrere nel rischio di allenare il modello su un set di dati troppo particolare, con conseguenti effetti di overfitting. Con il metodo di cross validation ci assicuriamo che ogni osservazione compaia lo stesso numero di volte in fase di training e una sola volta in fase di test. Inoltre si è deciso di indicare la classe predetta come variabile di stratificazione in modo da garantire che ciascuna classe sia presente con la stessa quota in ogni partizione.
La Tabella 1 riporta le performance dei classificatori applicati in seguito al nodo di *attribute selection* che come è stato illustrato nella sezione 3.3 mantiene solo 3 variabili. I modelli migliori, sia secondo *AUC* che *Recall*, risultano *AdaBoostM1*, *Random Forest*, *J48* e *KNN*. Nella parte bassa della classifica vi sono invece i modelli probabilistici, *Multi Layer Perceptron* e Regressione Logistica. Si noti come quest'ultimo modello abbia valori molto diversi per *F-Score* e *AUC*, ad indicare come l'utilizzo di un treshold differente potrebbe far variare di molto l'attribuzione della classe.

Table 1. CV Performace with Attribute Selection

Classification Model	*Recall*	*Precision*	*F*	*AUC*
AdaBoostM1	0.923	0.872	0.897	0.975
Random Forest	0.919	0.870	0.894	0.977
J48	0.883	0.812	0.846	0.943
KNN	0.760	0.834	0.795	0.930
Naive Bayes Weka	0.707	0.760	0.733	0.901
MLP Weka	0.745	0.755	0.750	0.844
Logistic	0.105	0.681	0.182	0.720
Naive Bayes	0.143	0.514	0.224	0.612

Seguono gli indici di peformance riferiti ai modelli applicati sul dataset in seguito alla *feature selection* applicata con *Linear Correlation Filter*, secondo cui le variabili che apportano informazione non ridondante sono 15 su 22. Le performance risultano globalmente migliori rispetto al caso precedente ed il maggior incremento si verifica per *J48*. Si confermano comunque come migliori i modelli che applicano *Decision Tree*, con e senza tecniche di *ensemble*, oltre a *KNN*.

Table 2. CV Performace with Linear Correlation Filter

Classification Model	*Recall*	*Precision*	*F*	*AUC*
AdaBoostM1	0.971	0.913	0.941	0.987
Random Forest	0.965	0.912	0.938	0.990
J48	0.934	0.859	0.895	0.959
KNN	0.879	0.846	0.862	0.968
MLP Weka	0.792	0.780	0.786	0.878
Naive Bayes Weka	0.772	0.746	0.759	0.913
Logistic	0.147	0.708	0.244	0.751
Naive Bayes	0.141	0.519	0.221	0.635

Infine si riportano i valori di performance riferiti al gruppo di modelli allenati su tutte le variabili a disposizione, ovviamente ad eccezione di *Sku*. In quest'ultimo gruppo si verifica un lieve peggioramento globale delle metriche a disposizione.

Table 3. CV Performace (all attributes)

Classification Model	*Recall*	*Precision*	*F*	*AUC*
AdaBoostM1	0.923	0.875	0.897	0.975
Random Forest	0.919	0.812	0.846	0.943
J48	0.883	0.812	0.846	0.943
KNN	0.760	0.834	0.795	0.930
MLP Weka	0.745	0.755	0.750	0.844
Naive Bayes Weka	0.707	0.760	0.733	0.901
Logistic	0.105	0.681	0.182	0.720
Naive Bayes	0.143	0.514	0.224	0.612

Il primo aspetto che emerge dai risultati ottenuti è che, indipendentemente dal tipo di *feature selection*, i modelli più promettenti sono sempre gli stessi. Infatti i modelli basati su *Decision tree* sembrano performare sempre meglio rispetto a quelli probabilistici.

5. Selezione dei modelli migliori

In questa sezione ci occuperemo di prendere in considerazione i risultati raggiunti dai vari modelli, dopo essere stati sottoposti alla cross validation, optando per i tre modelli che hanno raggiunto livelli più apprezzabili in termini di *F-measure*, *AUC*, *Accuracy* e soprattutto di *Recall* (escluderemo il modello *AdaBoostM1* in quanto computazionalmente troppo oneroso). Il motivo che ci porta a preferire quest'ultima misura è il fatto che essa può anche essere vista come il *True positive Rate*:

$$TPR = \frac{TP}{TP+FN}$$

dove per TP intendiamo True Positive, per FN intendiamo False Negative e per classe positiva si intende la classe minoritaria ("*Yes*" nel nostro caso). Tale misura ci fornisce una grandezza che esprime la capacità del classificatore di non errare la predizione sulla classe rara, e di classificare correttamente il valore della variabile target "*Yes*" senza sbagliare e dunque non classificandola come "*No*". Valori bassi di *Recall* (*TPR*) ci indicano che il modello utilizzato non è in grado di riconoscere i valori "*Yes*", i quali vengono erroneamente classificati come "*No*", facendo aumentare per l'appunto il numero dei False Negative. Abbiamo quindi scelto di dare un peso maggiore a questa misura, cercando di massimizzarla, poiché riteniamo che giungere ad un modello di classificazione che riesca a classificare correttamente gli "*Yes*" (anche sbagliando nel classificare parte dei "*No*"), comporti una miglior soluzione per il problema in questione: meglio avere un prodotto in più in magazzino che non averlo proprio! I migliori 3 modelli risultanti verranno applicati al test set, verificando le performance addestrando il learner su tre differenti configurazioni del training set (selezionando attributi differenti). Una volta ottenuta la miglior tecnica di *feature selection* confrontando i risultati dei 3 modelli, si applicherà la cross validation dei modelli sul training set (senza gli attributi

eliminati) per evitare di incorrere in problemi di overfitting dei modelli. Infine si utilizzerà il test set per selezionare il modello migliore in termini di *Recall* dopo aver addestrato i 3 learner utilizzando 2 differenti tecniche di sampling.

5.1 Valutazione delle performance

Una volta estrapolati i valori delle performance su dieci iterazioni attraverso la cross validation, utilizzando più modelli e su diverse configurazioni del training set (selezionando gli attributi tramite *InfoGain* / *Cfs*, *Linear Correaltion* ed anche mantenendo tutti gli attributi), è possibile confrontare i modelli: abbiamo selezionato quindi i tre più performanti, li applicheremo al test set, per poi comparare le misure di performance. Il processo di classificazione è stato svolto con i seguenti learner: *RandomForest* di weka, *J48* e infine *KNN* (K Nearest Neighbor). I suddetti modelli sono stati dapprima addestrati utilizzando come training set ciò che ci è giunto come output dal processo di *attribute selection* tramite filtri. La riduzione della dimensionalità, da 22 a 7 attributi, ci ha permesso di sviluppare un algoritmo migliore sia in termini di tempo che di memoria utilizzata. I risultati relativi a *Precision*, *Recall*, *F-measure* e *AUC* sono i seguenti:

Table 4. Performace on classification (Cfs e InfoGain)

Classification Model	*Recall*	*F*	*AUC*	*Accuracy*
Random Forest	0.449	0.156	0.825	0.941
J48	0.612	0.151	0.782	0.917
KNN	0.461	0.146	0.708	0.935

Si può facilmente notare dalla tabella che nel caso in cui il training set venga sottoposto ad un processo di attribute selection tramite filtri, il learner che raggiunge risultati migliori in termini di *Recall* è il *J48*.
Ora esporremo i risultati di performance dei tre classificatori, addestrati sul training set precedentemente sottoposto al processo di analisi delle correlazioni lineari fra le variabili, al fine di rimuoverne le più ridondanti.

Table 5. Performace on classification (Linear Correlation)

Classification Model	*Recall*	*F*	*AUC*	*Accuracy*
Random Forest	0.487	0.221	0.87	0.959
J48	0.598	0.191	0.741	0.939
KNN	0.576	0.172	0.768	0.933

Anche in questo caso, il modello migliore in termini di *Recall* risulta essere il *J48* seguito da *KNN*.
Infine riportiamo i risultati di performance relativi ai tre classificatori, addestrati mantenendo il numero di attributi originale, non applicando dunque tecniche per ridurre la dimensionalità.

In questo caso il modello migliore, sempre in termini di *Recall*, risulta essere il *KNN* seguito dal *J48*.
Con questi risultati in mano, possiamo ora effettuare un confronto ed individuare la configurazione del training set che permette ai modelli di giungere ad un livello di *Recall* più alto rispetto alle altre configurazioni. Guardando dunque la colonna corrispondente di ogni tabella, risulta evidente che la configurazione del training set che ha portato a risultati di *Recall* più alti in media è quella derivante dal processo di *Linear Correlation*, che ha portato il numero di variabili esplicative da 21 a 14, eliminando quelle ridondanti. Utilizzeremo quindi queste variabili ed eseguiremo un'ultima cross validation, questa volta prima di effettuare il processo di oversampling della classe positiva (rara) e quello di undersampling di quella negativa (non rara), al fine di non incorrere in possibli problemi di overfitting. Infine, dopo i processi di riduzione della dimensionalità tramite sampling, utilizzeremo i tre modelli, già utilizzati in precedenza in ciascun processo, e confronteremo i risultati finali.

Table 6. Performace on classification (all attributes)

Classification Model	*Recall*	*F*	*AUC*	*Accuracy*
Random Forest	0.465	0.273	0.889	0.97
J48	0.541	0.199	0.695	0.947
KNN	0.58	0.156	0.791	0.925

5.2 Modello migliore

Ora viene considerato solamente il training set che si ottiene come output del processo di *Linear Correlation* filtering. Su questo abbiamo deciso di eseguire un'ultima *cross validation* usando i tre modelli utilizzati anche in precedenza (*RandomForest*, *J48*, *K Nearest Neighbor*), al fine di valutare le performance di classificazione, prima che il training set fosse sottoposto ai processi di over- ed undersampling, in modo da evitare eventuale overfitting. Le impostazioni con cui abbiamo eseguito questo processo di cross validation sono le seguenti: il numero di folds, e quindi anche delle validazioni che vengono iterate è pari a 10; il campionamento stratificato è in funzione dell'attributo di classe ("*went on backorder*").
I risultati finali di questa cross validation sono assolutamente positivi sia in termini di *Accuracy*, che supera il 99% (anche se non avrà molta rilevanza per la valutazione finale) con tutti i learner, sia in termini di *Recall*.
I prossimi processi che affronteremo in quest'ultima parte saranno quelli di *oversampling* e *undersampling*. Nel primo caso abbiamo deciso di rafforzare l'*oversampling* effettuato in precedenza quadruplicando il numero degli "*Yes*" e nel contempo riducendo quello dei "*No*" con un campionamento casuale del 5%. L'output di questo processo consiste in 122 547 tuple, che verrà utilizzato per la successiva validazione sul test set. Nel secondo caso invece, abbiamo optato per un processo di *undersampling* (che non è stato tentato in precedenza su tutti i modelli principalmente per problemi di computazione), riducendo il numero di ricorrenze della classe non rara: abbiamo tenuto solo il 2% dei "*No*" e abbiamo conservato tutti gli " *Yes*". Come output abbiamo ottenuto una tabella di 42 243 record, che sarà la seconda tipologia di set utilizzato per la validazione con il test set.
Il processo finale di classificazione quindi, è stato svolto sia per il training set sottoposto ad *oversampling* sia per quello

ottenuto dall' *undersampling*, applicandovi i tre learner già sopracitati; infine abbiamo comparato i risultati raggiunti per individuare quale tecnica porti ai migliori risultati in termini di *Recall*. I risultati sono visualizzati nella tabella 7; i classificatori seguiti da una ".O" fanno riferimento all'*oversampling*, mentre quelli seguiti da una ".U" all'*undersampling*. Risulta

Table 7. Performace on classification - Final

Classification Model	*Recall*	*F*	*AUC*	*Accuracy*
Random Forest.O	0.506	0.223	0.873	0.957
J48.O	0.602	0.179	0.72	0.933
KNN.O	0.6	0.165	0.769	0.927
Random Forest.U	0.607	0.202	0.895	0.942
J48.U	0.687	0.186	0.871	0.927
KNN.U	0.631	0.163	0.814	0.922

chiaro che livelli soddisfacenti di *Accuracy* vengono raggiunti da tutti i classificatori, indipendentemente dall'*oversampling* od *undersampling*. Ciò detto, vengono raggiunti livelli più alti di *Recall* da quei classificatori che hanno avuto come input il training set sottoposto ad *undersampling* della classe non rara.
Dal confronto dei risultati emersi, risulta quindi razionalmente preferibile fornire al modello di classificazione un training set sottoposto precedentemente ad un processo di *undersampling*, data la nostra propensione a preferire valori di *Recall* più alti. Nella *figura 2* è possibile osservare l'andamento delle curve ROC dei tre modelli selezionati.

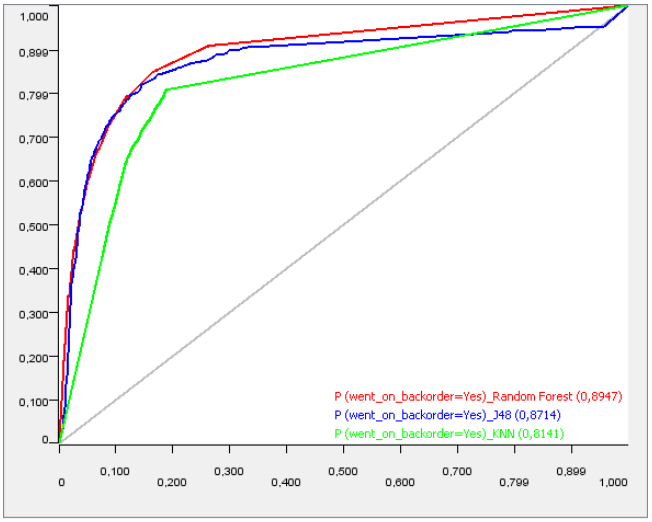

Figure 2. Curve ROC dei 3 migliori modelli selezionati, utilizzando la tecnica di undersampling del training set.

In definitiva, per il problema affrontato, il modello *J48* risulta essere il migliore, con una *Recall* pari a 0.687.

Conclusioni

Dalle analisi effettuate è emerso come, con i dati a nostra disposizione, sia stato difficile ottenere un risultato soddisfacente in termini sia di *Precision* che di *Recall* (l'*Accuracy* sempre molto alta è dovuta al forte sbilanciamento del dataset e dunque non è stata oggetto principale di considerazione). Per questo, nella valutazione del migliori modelli, abbiamo prestato attenzione ai valori della *Recall* (ovvero il *True Positive Rate*), supponendo che per un'impresa sia più prudente avere delle merci in magazzino in più (anche nel caso in cui alcune di esse non siano soggette a "*back-order*") piuttosto che rischiare di rimanere con quantità insufficienti ed incorrere in uno dei problemi citati inizialmente nel caso sia costretta ad effettuare dei "*back-order*" (in sostanza abbiamo considerato maggiormente efficace il modello che predicesse la maggior parte degli "*Yes*" correttamente, anche sbagliando nel predire alcuni "*No*"). Ovviamente ogni azienda ha una strategia di mercato che può differire profondamente da quella di un'impresa operante nel suo stesso settore. In particolare, la strategia di gestione del magazzino dipende fortemente dai costi necessari per la tenuta delle merci, il che poi influenza le tempistiche di carico e scarico del magazzino. Conoscendo i costi, sarebbe stato possibile (attraverso dei *cost sensitive classifiers*), pesare le predizioni del modello, in modo da fornire un output molto più preciso. Oltre a ciò, per migliorare ulteriormente il modello, si sarebbero potute seguire diverse vie: utilizzare una terza tecnica di *feature selection*, la quale, reiterando un determinato modello, va ad escludere quegli attributi che non sono utili alla determinazione della variabile output (nodo *feature elimination* in Knime; oppure si sarebbe potuta utilizzare una nuova tecnica di oversampling, denominata *SMOTE* (Synthetic Minority Over-sampling Technique), che avrebbe permesso di aumentare le osservazioni della classe minoritaria creandone di nuove.
Ciò non è stato eseguito principalmente per problemi computazionali dovuti alle dimensioni del dataset, ma sarebbe, con alta probabilità, servito a migliorare i risultati ottenuti. Per concludere dunque si può affermare che innanzitutto le variabili a nostra disposizione non sono sufficienti per spiegare il fenomeno con precisione. Mantenendo comunque la *Recall* come misura di valutazione (per i motivi suddetti), il modello più efficace è il *J48* che utilizza 14 dei 21 attributi esplicativi del dataset, selezionati tramite *Linear correlation*, mantenendo il 90% dell'informazione.

Sitografia

[1] - What is a backorder and how to do it right.
[2] - Dataset.
[3] - Learning from Imbalanced Classes.
[4] - How to handle Imbalanced Classification Problems in Machine Learning?.