Prenotazioni visite mediche: è possibile prevedere se il paziente si presenterà all'appuntamento?

Luca Cassani, Andrea Cesano, Emanuela Elli, Paolo Giua, Eleonora Rossi

CdLM Data Science, Università degli studi di Milano Bicocca
Corso di Machine Learning, Anno Accademico 2021-2022

Aquarela Advanced Analytics **[1] ha riscontrato che circa il 30% dei pazienti non si presenta alle visite mediche di controllo. Il fenomeno può avere diverse motivazioni, dalla dimenticanza dell'appuntamento alla scarsa volontà di presentarsi, anche da parte di soggetti affetti da diverse patologie. È possibile prevedere se un paziente si presenterà o meno alla visita medica? Partendo dal dataset Kaggle *"Medical Appointment No Shows"* [2] si cercherà di verificare questa ipotesi.**

1 Introduzione ed obiettivi

Al fine di studiare il comportamento degli individui che prenotano visite mediche a cui però non si presentano, l'azienda informatica *Aquarela Advanced Analytics* di Florianópolis (Brasile) [1] ha raccolto un set di dati relativi alle prenotazioni effettuate dai pazienti con i rispettivi appuntamenti previsti nel periodo tra il 2015 e il 2016. I dati disponibili sono aggiornati al 2017 e consistono in oltre 100 mila appuntamenti, di cui sono stati registrati sia i dettagli relativi alla prenotazione, quali ad esempio data ed ora, sia alcune informazioni anamnestiche del paziente.

Questo lavoro nasce con lo scopo di indagare tra le informazioni fornite all'interno del dataset per prevedere se i pazienti futuri siano soggetti più o meno propensi a non presentarsi agli appuntamenti. In questo modo si vogliono ridurre le potenziali perdite di denaro e di ore di lavoro dovute alla mancata presentazione.

2 Descrizione dataset

Nel dataset `KaggleV2-May-2016.csv`, disponibile su Kaggle [2], vengono raccolti i dati di 110.527 appuntamenti per visite mediche associati a 14 attributi differenti. I dati sono relativi alle prenotazioni da Novembre 2015 a Giugno 2016 con i rispettivi appuntamenti

previsti nel periodo tra Aprile 2016 e Giugno 2016. Gli attributi forniti si distinguono in:

- **PatientId**: numero identificativo del paziente;
- **AppointmentID**: numero identificativo dell'appuntamento;
- **Gender**: sesso del paziente;
- **AppointmentDay**: giorno ed orario dell'appuntamento;
- **ScheduledDay**: giorno ed orario di prenotazione della visita;
- **Age**: età del paziente;
- **Neighbourhood**: luogo in cui si trova lo studio medico dell'appuntamento;
- **Scholarship**: indica se il paziente riceve aiuti finanziari dal governo;
- **Hipertension**: indica se il paziente è affetto da ipertensione;
- **Diabetes**: indica se il paziente è affetto da diabete;
- **Alcoholism**: indica se il paziente soffre di alcolismo;
- **Handcap**: livello di handicap del paziente (vi sono 5 livelli, da 0 a 4, che denotano progressivamente la gravità dell'handicap sofferto);
- **SMS_received**: il paziente ha ricevuto un reminder SMS dell'appuntamento;
- **No-Show**: indica se il paziente NON si è presentato all'appuntamento.

3 Preprocessing

Al fine di rendere il set di dati più adatto all'analisi sono state implementate le seguenti modifiche attraverso l'utilizzo della piattaforma Knime [3].

3.1 Manipolazione dei dati

Per prima cosa si è notato che l'attributo *AppointmentDay* riporta un valore dell'orario pari a "00:00:00" mentre l'attributo *ScheduledDay* riporta un orario specifico. Per questo motivo si è deciso di sostituire tutti i valori relativi agli orari della variabile *ScheduledDay* egualmente con "00:00:00", tramite l'utilizzo del nodo `String Manipulation`, al fine di eseguire confronti con granularità giornaliera.
Dopodiché, siccome gli attributi *ScheduledDay* e *AppointmentDay* vengono letti dalla piattaforma Knime come stringhe, si è deciso di effettuare una conversione in variabili di tipo "Zoned Date Time" con il nodo `String to Data&Time` (in questo modo si memorizzano tutti i campi di data e ora, con una precisione di nanosecondi ed un fuso orario con un offset di zona utilizzato per gestire date e ore locali ambigue).

Grazie a questa conversione è stato possibile utilizzare il nodo `Date&Time Difference` per calcolare le variazioni tra *ScheduledDay* e *AppointmentDay*. Tali valori sono stati successivamente inseriti in un nuovo attributo chiamato *DaysToAppointment*, mentre le variabili utilizzate per il calcolo della differenza sono state eliminate dal dataset poiché non più rilevanti per l'apprendimento del modello.
Riscontrando valori negativi all'interno del nuovo attributo *DaysToAppointment*, ritenuti inaccettabili perché indicherebbero una data dell'appuntamento precedente alla data di prenotazione, si è utilizzato il nodo `Rule-based Row Filter` per eliminare tutti i record il cui valore della variabile fosse negativo. La medesima operazione è stata eseguita anche per l'attributo *Age*, avendo riscontrato nuovamente la presenza di valori negativi (si suppone siano frutto di errori in fase di acquisizione). Non sono stati riscontrati valori mancanti in nessuno degli attributi disponibili.

3.2 Rimozione degli attributi

L'ultima fase del preprocessing dei dati si concentra sulla rimozione degli attributi meno significativi tramite il nodo `Column Filter`. In primis si è deciso di eliminare la variabile *AppointmentID* poiché indica il numero identificativo dell'appuntamento, ma al contempo ogni record è già identificato univocamente attraverso il valore dell'attributo *RowID* (indica il numero identificativo di ogni record all'interno del dataset), pertanto tale attributo è stato giudicato ridondante ai fini dell'analisi.
Il secondo attributo eliminato è *PatientId* poiché nel dataset erano presenti record di pazienti con

lo stesso ID, ma diversi valori negli altri attributi. Mediante questa rimozione si considera ogni record come indicativo di un paziente univoco.
Per quanto riguarda l'attributo *Neighbourhood* un approccio possibile sarebbe stato quello di effettuare una binarizzazione. In questo modo, però, si sarebbe aumentata la dimensionalità del dataset di tanti attributi quanti sono i possibili valori assunti dall'attributo *Neighbourhood* (circa 80). Per tale motivo si è deciso di non utilizzare questa tecnica e di rimuovere l'attributo.

4 Classificazione

4.1 Gestione del dataset sbilanciato

Il dataset oggetto di studi è fortemente sbilanciato: sui 110.527 records osservati, 88.207 hanno valore "No" per la variabile di classe *No-Show*.
Si sono valutate varie possibilità per gestire questa situazione: undersampling, oversampling e la costruzione di un'opportuna matrice dei costi. Adottando una tecnica di undersampling, ovvero rimuovendo records dalla classe più numerosa, si può incorrere in una rischiosa perdita di informazione. D'altro canto applicando oversampling, ovvero creando nuovi records della classe meno rappresentata (sfruttando ad esempio il nodo `SMOTE`[4] di Knime), si rischia di creare un modello con problemi di overfitting sui dati di partenza. Viste queste problematiche si è deciso di costruire una matrice dei costi.
Per fare ciò si è ipotizzato quanto sarebbe dispendioso gestire i due tipi di errore di classificazione per un ospedale. Si assume che nel caso di pazienti classificati come positivi, ovvero che secondo il modello non si presenteranno alla visita, l'ospedale potrebbe implementare un sistema di invio di promemoria attraverso chiamate (sostenendo i costi associati all'impiego del personale e della chiamata stessa), dunque nel caso di un paziente falso positivo verrebbe effettuata una chiamata inutile. Il caso di falsi negativi, ovvero persone che secondo il classificatore si presenteranno e invece non si presentano, genererebbe un mancato guadagno e quindi uno spreco di risorse molto maggiore.
Per questo si è quindi assegnato un costo pari ad 1 ai falsi positivi ed un costo pari a 20 ai falsi negativi. Le classificazioni esatte si valutano a costo 0. La matrice utilizzata dunque è la seguente:

		predicted class	
		-1	+1
actual class	-1	0	1
	+1	20	0

4.2 Classificatori

Si vogliono ora istruire dei modelli di classificazione per identificare i pazienti che potrebbero non presentarsi a una visita medica prenotata.
Visto che, dopo il preprocessing, gli attributi esplicativi sono esclusivamente numerici o binari, c'è grande libertà nella scelta dei possibili modelli di classificazione da applicare, dunque sono stati utilizzati vari modelli per identificare il più adatto:

- **Albero di decisione J48**: è un modello di classificazione euristica. Gli alberi decisionali operano delle partizioni tra i records del dataset a seconda dei valori assunti da determinati attributi esplicativi, generando una struttura ad albero. Per ogni nodo dell'albero assegnano la classe più rappresentata dai records presenti.
- **Random Forest**: è anche questo un modello di classificazione euristica, derivato dagli alberi di decisione. Una Random Forest è costituita da più alberi decisionali e a seconda di alcuni parametri valuta a quale albero assegnare ogni record per la classificazione.
- **Support Vector Machine**: appartiene alla classe degli algoritmi di separazione, ovvero che operano delle partizioni nello spazio degli attributi per identificare il valore dell'attributo di classe. Nel nostro caso è stato implementato con il nodo `SPegasos` di Weka.
- **Multi-Layer Perceptron**: è anche questo un modello di separazione, che appartiene nello specifico alla categoria delle reti neurali. Il MLP è una rete di neuroni che si

muovono in modo unidirezionale, dall'input alla variabile di output, passando attraverso un numero variabile di livelli nascosti.

- **Rete Bayesiana**: è un modello probabilistico, che si basa sulla formula di Bayes per calcolare la probabilità che il record appartenga a un dato valore della classe di output condizionatamente ai valori dei suoi attributi di input. Questo modello è derivato da quello di Naive Bayes, ed è spesso preferito a quest'ultimo perchè non richiede l'assunzione di indipendenza condizionata.

Vista la presenza della matrice dei costi si è usato come learner per ogni classificatore il nodo di Knime `CostSensitiveClassifier`[5], implementato da Weka e come inducer il nodo `Weka Predictor`.

5 Misure di performance

Per valutare la performance di un classificatore è necessario introdurre il concetto di "matrice di confusione", una rappresentazione contenente il conteggio dei records classificati come veri positivi (TP), falsi positivi (FP), veri negativi (TN) e falsi negativi (FN). Partendo da questi valori è possibile ottenere degli indicatori, elencati di seguito, utilizzati per determinare l'efficacia del classificatore.

- **Accuracy**: rapporto tra records correttamente classificati rispetto alla totalità dei records disponibili.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- **Precision**: rapporto tra records classificati correttamente come positivi rispetto a tutti quelli classificati come positivi.

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

- **Recall**: rapporto tra records classificati correttamente come positivi rispetto alla totalità dei records effettivamente positivi.

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

- **F1-score**: media armonica ottenuta combinando Precision e Recall ed utilizzata come valore riassuntivo delle due.

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Definiti questi indicatori, occorre ricordare che il dataset di partenza non è bilanciato dal momento che l'attributo di classe è ripartito, approssimativamente, in misura dell'80% sul valore "no" e del restante 20% sul valore "yes". Questo sbilanciamento rappresenta un problema, dato che gli attributi non hanno lo stesso peso e che in un'ottica di riduzione dei costi è più importante classificare correttamente la classe rara (*NoShow: yes*) rispetto all'altra. Questo porta a due implicazioni.

In primo luogo, considerando che l'interesse di questa ricerca è limitato alla corretta identificazione dei records con realizzazione "yes", l'indice di Accuracy non risulta significativo. Volendo minimizzare il numero di falsi negativi (caso più costoso), si è deciso pertanto di limitarsi al confronto tra gli indici di Recall per valutare la performance dei classificatori utilizzati.

In secondo luogo, dato che non tutti i casi associati alla matrice di confusione hanno lo stesso peso, è necessario introdurre un'ulteriore misura di performance: il costo.

Il cost-sensitive learning prevede l'utilizzo di una matrice di costo, come spiegato nel sottoparagrafo 4.1 "Gestione del dataset sbilanciato", che viene combinata con la matrice di confusione mediante la seguente funzione di costo:

$$Costo = C_{TN} \cdot TN + C_{FP} \cdot FP + C_{FN} \cdot FN + C_{TP} \cdot TP$$

In questo modo ad ogni classificatore verrà assegnato un costo differente, in funzione della sua capacità di classificare records correttamente, e questi si potranno confrontare secondo il criterio di minimizzazione del costo.

Ultimo indice di valutazione delle performance è la curva ROC (*Receiver Operation Characteristic*) e la relativa misurazione dell'area sottesa dalla curva (AUC - *Area Under the Curve*). Si tratta di una rappresentazione cartesiana utilizzata per mostrare il legame tra veri positivi rispetto al totale dei positivi (TPR, True Positive Rate, indicati in ordinate) e falsi positivi sul totale dei positivi (FPR, False Positive Rate, indicati in ascisse).

Questa rappresentazione permette di confrontare diversi classificatori senza considerare la distribuzione dell'attributo di classe o il costo dell'errore, ma verificando invece il numero di falsi positivi sopportabili, guardando ai valori assunti dall'area sotto la curva.

6 Analisi dei risultati

6.1 Risultati con tutti gli attributi

Come già anticipato, per la valutazione delle performance sono stati utilizzati il costo, la Recall e la curva ROC per le motivazioni esplicitate in precedenza. É stata inizialmente condotta la classificazione (e la successiva valutazione dei modelli) utilizzando tutti gli attributi del dataset non rimossi durante la fase di preprocessing.

	Acc.	Prec.	Recall	Costo
J48	0,202	0,202	1	29.108
RF	0,482	0,255	0.814	44.977
MLP	0,469	0,265	0.92	30.569
SVM	0,798	//	0	147.280
BN	0,427	0,256	0,96	26.479

Nella tabella appena presentata sono riportati i valori associati alle misure di performance per ogni modello di classificazione.

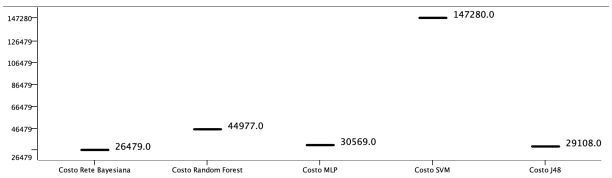

Figura 1: *Boxplot costi*

Dal boxplot relativo ai costi emerge che ad SVM è associato un costo di 147.280 e, pertanto, si posiziona ultimo tra i classificatori utilizzati. La Rete Bayesiana, invece, con un valore di 26.479 risulta essere il migliore classificatore in relazione al costo, seguito da J48 con 29.108.

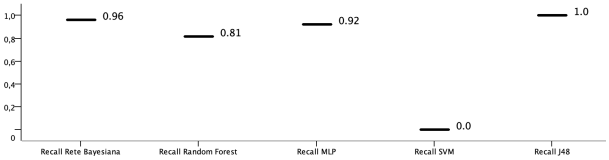

Figura 2: *Boxplot Recall*

Dall'analisi della Recall emerge che J48 ha la più alta Recall (pari a 1), seguito dalla Rete Bayesiana (pari a 0.96) e da MLP (pari a 0.92). SVM, invece, si posiziona ultimo con una Recall di 0. Da menzionare il fatto che J48 risponda solo con la classe positiva (*No-Show*).

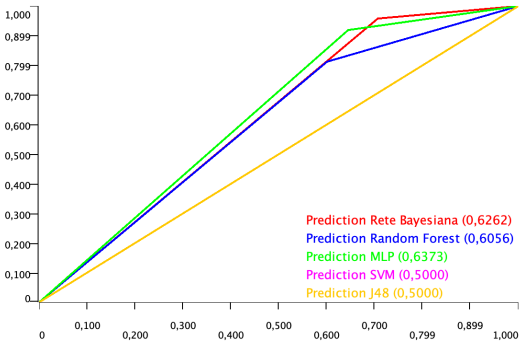

Figura 3: *Curva ROC*

In ultimo, con riferimenti alla curva ROC e all'area sottesa alla curva (AUC), MLP e la Rete Bayesiana risultano generare i modelli migliori con AUC pari, rispettivamente, a 0,6373 e a 0,6262.
Non è comunque possibile indicare quale tra i due sia il migliore in senso assoluto: MLP, infatti, ha migliori performance considerando la copertura di True Positive fino a circa il 92%, valore per il quale entrambi i classificatori rispondono con circa il 68% di False Positive. Da questa soglia in poi, la rete Bayesiana risponde con un numero di False Positive inferiore rispetto a MLP.
SVM e J48, invece, risultano essere i peggiori in quanto presentano un valore AUC di 0,5. Le curve

ROC associate mostrano lo stesso andamento del modello "Zero Rule".

6.2 Feature selection

Successivamente si è deciso di effettuare un'analisi introducendo il meccanismo della feature selection. Si tratta di un metodo applicabile al dataset per ridurne la dimensionalità, selezionando gli attributi più rilevanti mediante un particolare approccio. In questo studio è stato scelto un approccio di tipo "Filter", grazie al quale gli attributi sono selezionati per mezzo di una funzione obiettivo prima dell'introduzione (e dell'apprendimento) del classificatore.
Esistono due tipi di tecniche:

- **Univariata**: Tecnica mediante la quale, in primo luogo, si prevede la scelta di una misura di associazione tra gli attributi candidati e quello di classe. Successivamente si procede all'ordinamento degli attributi sulla base della misura di associazione considerata. Infine, si selezionano i primi N attributi come attributi di input per il classificatore. Tale tecnica consente di rimuovere gli attributi non rilevanti ma non tiene in considerazione la ridondanza degli stessi.

- **Multivariata**: Tecnica che prevede la rimozione di attributi irrilevanti congiuntamente a quelli ridondanti, valutando una funzione obiettivo su sottoinsiemi di attributi.

Nel caso in esame sono state utilizzate entrambe le tecniche al fine di ridurre la dimensionalità del dataset, i tempi di inferenza del classificatore e prevenire il fenomeno di overfitting.

6.3 Risultati con feature selection

È stato introdotto, quindi, il metodo della feature selection di tipo Filter Univariato: dopo aver osservato i risultati associati al nodo `AttributeSelectedClassifier`, si è deciso di selezionare i primi 3 attributi esplicativi in quanto più rilevanti degli altri, come si può notare dalla figura sotto riportata.

```
Attribute Evaluator (supervised, Class (nominal): 9 No-show):
        Information Gain Ranking Filter

Ranked attributes:
 0.0742447   10 DaysToAppointment
 0.01107971   8 SMS_received
 0.00553704   2 Age
 0.00088527   4 Hipertension
 0.00059226   3 Scholarship
 0.00008828   5 Diabetes
 0.00000881   1 Gender
 0            6 Alcoholism
 0            7 Handcap
```

Figura 4: *Risultati ottenuti tramite il nodo* `AttributeSelectedClassifier`

In particolare, sono stati rimossi gli attributi *Gender*, *Scholarship*, *Hipertension*, *Diabetes*, *Alcoholism* e *Handcap*.

	Acc.	Prec.	Recall	Costo
J48	0,202	0,202	1	29.108
RF	0,469	0,259	0,874	37.019
MLP	0,347	0,167	0,559	85.534
SVM	0,798	//	0	147.280
BN	0,438	0,259	0,958	26.422

Da questo processo, come si evince dalla tabella sopra riportata, non si notano evidenti cambiamenti in relazione alle misure di performance rispetto alla classificazione eseguita senza feature selection.
In particolare, costo e Recall risultano pressoché invariati per quasi tutti i modelli: la Rete Bayesiana e J48 si confermano i migliori rispettivamente con un costo di 26.422 e di 29.108, e con una Recall di 0,96 e di 1,0.
MLP, invece, subisce un importante calo di performance raggiungendo un costo di 85.534 e una Recall di 0,56.
Per quanto riguarda la curva ROC e il parametro AUC, la Rete Bayesiana risulta essere la migliore per qualsiasi copertura, con un valore associato ad AUC di 0,6321.

Tale trend persiste con l'introduzione della feature selection di tipo Filter Multivariato, che ha permesso la selezione di solo 2 attributi: *Scholarship* e *DaysToAppointment*.

	Acc.	Prec.	Recall	Costo
J48	0,202	0,202	1	29.108
RF	0,484	0,272	0,929	28.725
MLP	0,797	0,239	0,001	147.095
SVM	0,798	//	0	147.280
BN	0,484	0,272	0,931	28.527

Il classificatore Random Forest raggiunge J48 e la rete Bayesiana in termini di costi e Recall, allineandosi su un costo di circa 28.000 e su una Recall di 0,93.
MLP, invece, subisce un ulteriore calo di performance raggiungendo SVM, con un costo pari a circa 147.000 e una Recall pari a 0.

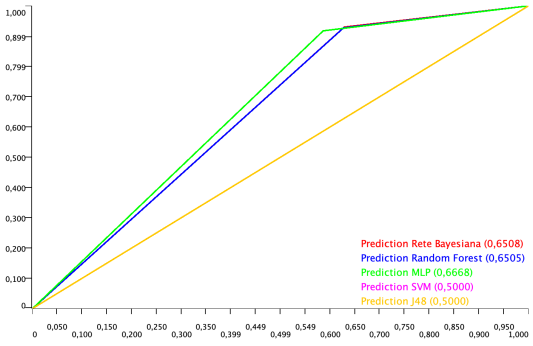

Figura 5: *Curva ROC con feature selection Multivariata*

Infine, analizzando la curva ROC, MLP ottiene i risultati migliori con un valore associato ad AUC di 0,667. Rete Bayesiana e Random Forest hanno curve sovrapponibili e, ad entrambi, è associato un valore AUC di 0,65.

7 Conclusioni e sviluppi futuri

In conclusione, confrontando come i classificatori si comportano prima e dopo la feature selection, è possibile notare misure di performance paragonabili, eccezion fatta per MLP che mostra un netto peggioramento. Quindi, tale pratica è raccomandata al fine di ottenere una dimensionalità ridotta e tempi di inferenza minori. Nel nostro caso di studi l'analisi multivariata, che considera gli attributi non singolarmente ma in sottoinsiemi, ci permette di raggiungere la maggior riduzione di dimensionalità del dataset. Questa analisi ha mostrato che, in termini di Recall e di costo, i modelli basati sul J48, la Rete Bayesiana e la Random Forest ottengono sempre i valori migliori mentre, analizzando la curva ROC, MLP ottiene il valore AUC più alto. D'altra parte, quest'ultimo ottiene un valore di Recall e di costo deludenti e non può essere preso in considerazione per il presente studio. Viceversa J48 ha la curva ROC peggiore, in quanto classifica tutti i record come facenti parte della stessa classe. Per queste motivazioni i modelli basati sulla Rete Bayesiana e sulla Random Forest, allenati in seguito all'analisi multivariata, rappresentano la scelta più adatta in quanto ottengono sempre i punteggi migliori.

Con l'intento di perfezionare l'analisi affrontata in questa sede, si potrebbero ricavare informazioni più utili incentrando questo studio su aspetti non rilevati nel dataset in esame. Ad esempio potrebbe essere interessante conteggiare il tasso di assenze pregresse di un paziente oppure aggiungere maggiori informazioni riguardo la data dell'appuntamento, come giorno della settimana o festività, così da evidenziare possibili consuetudini. Un paziente, infatti, potrebbe avere l'abitudine di non presentarsi agli appuntamenti oppure le persone potrebbero essere meno inclini a presentarsi alle visite mediche durante giorni festivi. Per queste introduzioni sarebbe però necessario una storicizzazione dei dati e maggiori informazioni riguardanti il contesto, che dev'essere supportata dall'infrastruttura sottostante e che per questo non è stato possibile introdurla durante questo studio.

Riferimenti bibliografici

[1] *Aquarela Advanced Analytics* `https://www.aquare.la/`

[2] *Kaggle dataset, "Medical Appointment No Shows".* `https://www.kaggle.com/joniarroba/noshowappointments`

[3] *KNIME* `https://www.knime.com/`

[4] *SMOTE* `https://towardsdatascience.com/smote-fdce2f605729`

[5] *CostSensitiveClassifier* `https://hub.knime.com/knime/extensions/org.knime.features.ext.weka_3.7/latest/org.knime.ext.weka37.classifier.WekaClassifierNodeFactory:f96b81db`