Churn Analysis: creazione di un modello di machine learning profittevole in ambito marketing

Ermellino Andrea, Kolyszko Matteo, Serino Antonio, Valente Sofia, Maggi Lucrezia

Dipartimento di Informatica Sistemistica e Comunicazione, DISCO , Università degli Studi di Milano Bicocca

Piazza dell'Ateneo Nuovo, 1, 20126 Milano MI

Abstract: Il customer churn rate riveste un ruolo fondamentale nel monitoraggio dell'andamento del business in termini di percentuale di consumatori che durante un determinato periodo di tempo ha "abbandonato" l'azienda, ponendo fine al rapporto. L'obiettivo di questa analisi è addestrare un modello in grado di assegnare agli utenti uno score veritiero di propensità al churn. Per far ciò sono stati addestrati modelli di Neural Network e Random Forest adottando diverse tecniche di campionamento al fine di risolvere il problema della classe sbilanciata, sono state confrontate le performance dei due modelli per individuarne il migliore che è stato utilizzato per l'ottenimento dello score sopracitato. È stata infine calcolata e confrontata la Propensity to churn con la percentuale media di churners reali, al fine di poter analizzare ancor meglio se le predizioni del miglior modello individuato potessero avere un effettivo riscontro nei dati reali.

Key words: Machine Learning, Churn Analysis, SMOTE, Random Forest, Neural Network.

Indice

1 Data Preparation

1.1 Descrizione del dataset

La versione iniziale del dataset è composta da 102 colonne e 330586 righe contenente informazioni sulle abitudini di navigazione di un gruppo di clienti di una piattaforma online.
In particolare è possibile distinguere macro-categorie di colonne ognuna delle quali descrive un diverso fattore:

- colonne con dati numerici che descrivono quali URL visitati dall'utente funzionavano e quali no;
- colonne con dati binari che indicano il sistema operativo e il browser utilizzato dagli utenti;
- colonne con dati numerici normalizzati nel range [0,1] che indicano il tempo di utilizzo della piattaforma in una determinata fascia oraria e in un determinato periodo della settimana;
- colonne con dati numerici normalizzati nel range [0,1] rappresentanti la lunghezza dei testi letti dai clienti;
- colonne con dati numerici normalizzati nel range [0,1] rappresentanti le categorie di interesse dei clienti.

- colonne con dati binari che indicano i pacchetti acquistati dall'utente e i servizi offerti preferiti.

Di nostro particolare interesse è la colonna *Pdisc*, questa infatti rappresenta la variabile binaria di target che indica se un cliente ha abbandonato l'azienda o meno.

1.2 Analisi esplorativa

L'analisi esplorativa parte dal conteggio di quanti churner sono presenti all'interno del dataset: la classe è particolarmente sbilanciata con il 97% di utenti no churner e il restante 3% churner. Successivamente si è andato a calcolare per i diversi tipi di contratti stipulati e per i diversi sistemi operativi utilizzati dagli utenti, il numero di churners: in particolare è stato possibile notare che la maggior parte dei churner sono quelli che non ususfruiscono del contratto STB_MYSKYHD e che usufruiscono del contratto STB_HD, mentre per i sistemi operativi la maggior parte dei churner sono i clienti che utilizzano Linux. E' stata effettuata inoltre l'analisi della correlazione presente tra la variabile di churn *Pdisc*, le categorie di interesse degli utenti e i pacchetti acquistati: in questa si è verificato che non sono presenti valori di correlazione significativi.

1.3 Preprocessing

Il preprocessing dei dati inizia con l'eliminazione di 8 righe duplicate e di 6 righe nelle quali per le colonne CINEMA, CALCIO, SPORT e SKY_FAMIGLIA, contenenti valori booleani, erano stati ritrovati valori pari a 2. Sono inoltre state eliminate tutte le colonne "admants" in quanto contenenti solo valori pari a 0. Prima di effettuare l'addestramento dei modelli il dataset è stato ordinato sulla base della colonna DATA_RIF che successivamente è stata eliminata insieme alla colonna EXTERNAL_ID in quanto non fornivano informazioni utili per l'analisi.

2 Feature Selection

Al fine di ottenere un modello più parsimonioso è stato deciso di implementare la feature selection, ossia la selezione delle caratteristiche maggiormente significative rispetto alle altre. Nel nostro caso si è scelto di utilizzare l'approccio di tipo Wrapper, più nello specifico, si è scelto l'algoritmo chiamato SequentialFeatureSelector(SFS). SFS è una procedura di tipo greedy in cui, ad ogni iterazione, viene scelta la migliore feature da aggiungere alle nostre feature già selezionate sulla base di uno score attribuito tramite cross validation. La procedura viene ripetuta fino ad aver raggiunto il numero di feature desiderate.

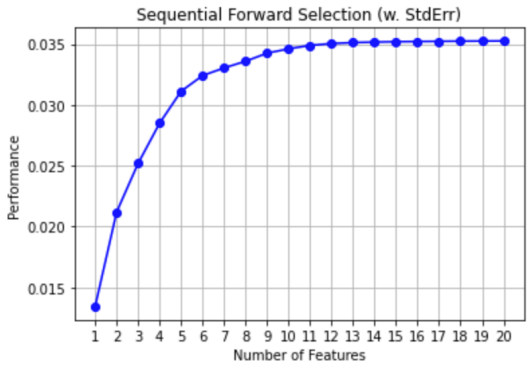

Figura 1: Numero di Feature Ideali

Come è possibile vedere dal grafico il numero di feature ideali corrisponde a 15, pertanto tutti i modelli verranno addestrati con un numero di colonne ridotto del 80%

3 Addestramento Modelli

In questa sezione del documento viene descritto il processo di addestramento dei modelli di Machine Learning selezionati: Random Forest e Artificial Neural Network. La selezione dei due modelli è stata guidata dal fatto che entrambi sono in grado di lavorare con un elevato numero di variabili riuscendo ad ottenere tempi di addestramento computazionalmente sostenibili.
L'obiettivo è addestrare un modello in grado di individuare un segnale che dia informazioni sul "livello di propensità" di un utente ad abbandonare il servizio analizzando il suo comportamento.
Per evitare problemi legati all'overfitting, tutti i modelli sono stati validati mediante K-Fold Cross Validation.

3.1 Problema della classe sbilanciata

Prima di procedere con l'addestramento dei modelli è necessario effettuare delle considerazioni sulla distribuzione dei valori della classe da predire (*Pdisc* ha valore 0 se l'utente non ha abbandonato il servizio, 1 se invece ha abbandonato). Come emerso dall'analisi esplorativa, la classe è estremamente sbilanciata: gli utenti che hanno abbandonato il servizio sono appena il 3% delle osservazioni totali, dunque gli utenti che invece non hanno abbandonato il servizio costituiscono il restante 97%.

Siccome l'interesse è per la classe rara (*Pdisc* = 1), nella divisione del dataset in train e test set è impossibile adottare una tecnica di campionamento casuale poiché il set di addestramento potrebbe non contenere osservazioni della classe rara , così come è inutile effettuare un campionamento stratificato perché pur mantenendo le proporzioni originali delle due classi, il modello addestrato su tale set performerebbe come una *ZeroRule*: andando a predire sempre il valore 0 si otterrebbe il 97% di Accuracy ma il modello risulterebbe essere completamente inutile per l'obiettivo (predire la classe rara).
Per risolvere tale problema sono state testate diverse tecniche di campionamento alternative che permettono di bilanciare le due classi nel train e nel test set:

- *Random Undersampling*: tecnica di campionamento che permette di bilanciare le due classi andando ad estrarre casualmente un numero di osservazioni della classe maggioritaria pari al numero di osservazioni della classe minoritaria. In questo modo sia nel train che nel test set c'è un rapporto del 50:50 delle due classi, tuttavia viene utilizzato solamente il 6% delle osservazioni totali del dataset, perdendo informazione;
- *Random Oversampling*: tecnica di campionamento che permette di bilanciare le due classi andando a selezionare casualmente delle osservazioni della classe minoritaria aggiungendole al train e al test set fino ad ottenere un rapporto del 50:50 fra le due classi;
- *Synthetic Minority Over-sampling Technique (SMOTE)*: tecnica di oversampling che permette di bilanciare le due classi andando a creare delle osservazioni sintetiche della classe minoritaria sfruttando i *K nearest neighbors* nello spazio delle variabili delle osservazioni appartenenti alla classe minoritaria.

Entrambi i modelli sono stati addestrati adottando ognuna delle tre tecniche di campionamento descritte per poi essere confrontati.

3.2 Valutazione delle performance

Per valutare la bontà dei modelli sono stati presi in considerazione i principali indicatori di performance: Accuracy, Precision, Recall, F-measure, AUC. Mentre l'Accuracy e l'AUC sono calcolate tenendo conto sia delle osservazioni della classe maggioritaria che della classe minoritaria, gli indicatori di Precision, Recall e F-Measure a cui si fa riferimento nelle sezioni successive di questo documento sono calcolati tenendo in considerazione le sole osservazioni appartenenti alla classe rara, ovvero quella di interesse per l'analisi. Gli indicatori vengono calcolati a partire dal numero di osservazioni correttamente classificate dal modello, e dal numero di osservazioni che invece vengono classificate in modo errato:

- *TP e TN*: numero di istanze appartenenti alla classe positiva (TP) o alla classe negativa (TN) che vengono correttamente classificate;
- *FP e FN*: numero di istanze appartenenti alla classe positiva (FN) o negativa (FP) che vengono classificate in modo errato.

L'Accuracy di un modello di classificazione indica il numero di predizioni corrette effettuate dal modello e assume valore nell'intervallo [0,1]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Nonostante sia un indicatore estremamente diffuso, poiché in questo caso l'obiettivo dell'analisi è predire quando le osservazioni appartengono ad una classe specifica (valore di *Pdisc* = 1), l'Accuracy non è un indicatore che da solo può indicare efficacemente la bontà di un modello. Decisamente più adatti al caso specifico in esame, sono gli indicatori di Precision e Recall di un modello. La Precision indica la frazione di record effettivamente positivi tra tutti quelli predetti come tali:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

La Recall invece, definita anche come True Positive Rate, indica la porzione di record positivi classificati correttamente dal modello:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Entrambi gli indicatori assumono valori nell'intervallo [0,1].
Tuttavia, spesso ci si trova davanti ad un tradeoff tra i due indicatori, un miglioramento della Recall porta ad un peggioramento della Precision e viceversa. Per questo motivo si ricorre all'utilizzo di misure alternative che riassumono i valori di Precision e Recall, come

la F-Measure.
La F-Measure è la media armonica tra Recall e Precision e anche questa, come gli indicatori descritti in precedenza, assume valori nell'intervallo [0,1]:

$$F - Measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (4)$$

L'ultimo indicatore numerico di performance preso in considerazione fa riferimento alla curva ROC: una curva che mette in relazione il rate di falsi positivi con il rate dei veri positivi. Tracciata la curva, l'area sottesa al grafico (AUC) risulta essere un indicatore di performance di particolare rilevanza.

3.3 Addestramento con Random Undersampling

Adottando questa tecnica di campionamento vengono utilizzate 19892 osservazioni delle 330mila iniziali ottenendo un rapporto del 50:50 tra le due classi. L'undersampling ha permesso di bilanciare il set di osservazioni, ma allo stesso tempo comporta una notevole perdita di informazione.
Di seguito viene riportato il confronto delle performance dei due modelli:

Classificatore	Accuracy	Precision	Recall	F-Measure	AUC
Neural Network	0.95	0.24	0.35	0.28	0.75
Random Forest	0.83	0.09	0.54	0.16	0.73

Tabella 1: Confronto Random Undersampling

La Neural Network performa leggermente meglio della Random Forest essendo più precisa nella predizione della classe rara. Anche gli indicatori di F-Measure e AUC sono superiori per la Neural Network che dunque risulta essere il modello migliore se si adotta il random undersampling come tecnica di campionamento.
Tuttavia, la perdita di informazione dovuta all'utilizzo di una minima parte (6%) delle osservazioni totali presenti nel dataset influisce notevolmente sulle performance dei modelli, che risultano essere discrete, ma sono facilmente migliorabili adottando tecniche che permettono di sfruttare al massimo le informazioni dell'intero set di dati.

3.4 Addestramento con Random Oversampling

Adottando questa tecnica di campionamento vengono utilizzate tutte le osservazioni presenti nel set di dati, in più per bilanciare le due classi vengono selezionate casualmente delle osservazioni della classe rara e vengono aggiunte al set. In totale vengono utilizzate 641252 osservazioni divise in train e test set. In questo modo non viene persa informazione, ma anzi viene amplificata la presenza di osservazioni appartenenti alla classe rara. Ciò potrebbe portare all'overfitting dei modelli, per questo motivo vengono validati mediante K-Fold Cross Validation.
Di seguito viene riportato il confronto delle performance dei due modelli:

Classificatore	Accuracy	Precision	Recall	F-Measure	AUC
Neural Network	0.94	0.22	0.34	0.27	0.75
Random Forest	0.95	0.22	0.26	0.24	0.69

Tabella 2: Confronto Random Oversampling

Mentre la Neural Network performa in maniera molto simile al modello addestrato con random undersampling, la Random Forest al contrario registra un netto miglioramento nella Precision ma un dimezzamento della Recall.
Anche in questo caso, la rete neurale risulta essere la scelta migliore rispetto alla Random Forest.

3.5 Addestramento con SMOTE

Adottando questa tecnica di campionamento viene sfruttato lo stesso numero di osservazioni del random oversampling (641252), tuttavia le osservazioni appartenenti alla classe rara che vengono aggiunte al set non sono dei duplicati selezionati casualmente ma sono delle osservazioni sintetiche generate sfruttando i K nearest neighbors nello spazio delle variabili delle osservazioni "reali".
Tale tecnica permette di bilanciare le due classi, tuttavia è doveroso chiarire che i modelli vengono addestrati su un set di dati che contiene informazioni sintetiche oltre alle informazioni reali.
Di seguito viene riportato il confronto delle performance dei due modelli:

Classificatore	Accuracy	Precision	Recall	F-Measure	AUC
Neural Network	0.95	0.25	0.28	0.26	0.73
Random Forest	0.94	0.18	0.29	0.22	0.69

Tabella 3: Confronto SMOTE

La Rete Neurale con tecnica di campionamento SMOTE ottiene delle perfomance pressoché identiche rispetto ai modelli addestrati adottando le tecniche di

campionamento precedentemente descritte.
Per quanto riguarda invece la Random Forest, anche con SMOTE le performance risultano essere molto simili a quelle ottenute adottando la tecnica di campionamento random oversampling.

In conclusione è utile guardare l'evoluzione delle performance dei due modelli adottando le varie tecniche di campionamento:

Classificatore	Campionamento	Accuracy	Precision	Recall	F-Measure	AUC
Neural Network	Random Undersampling	0.95	0.22	0.33	0.23	0.75
Neural Network	Random Oversampling	0.94	0.22	0.34	0.27	0.75
Neural Network	SMOTE	0.95	0.25	0.28	0.26	0.73

Tabella 4: Confronto performance Neural Network

Come si può facilmente notare, le tre tecniche portano a dei risultati estremamente simili nella valutazione della bontà del modello.

Classificatore	Campionamento	Accuracy	Precision	Recall	F-Measure	AUC
Random Forest	Random Undersampling	0.83	0.09	0.54	0.16	0.73
Random Forest	Random Oversampling	0.95	0.22	0.26	0.24	0.69
Random Forest	SMOTE	0.94	0.18	0.29	0.22	0.69

Tabella 5: Confronto performance Random Forest

Nel caso della Random Forest è evidente il miglioramento delle performance nel momento in cui si sfrutta un numero maggiore di osservazioni per l'addestramento, sia duplicate reali sia sintetiche.
Le performance della Neural Network sono estremamente simili se si confrontano le tre tecniche di campionamento, tuttavia c'è da considerare oltre le metriche che addestrando il modello con random undersampling si perde informazione.
La Neural Network addestrata adottando random oversampling viene utilizzata per il calcolo dell'indice di *Propensity to churn*

4 Propensity to churn

Una volta addestrata la Neural Network con random oversampling, il modello è stato utilizzato per predire la probabilità che un utente possa essere un churner oppure no. Per poter valutare la capacità del modello di individuare un "segnale", seppur debole, che indichi se un individuo possa essere un churner oppure no, il test set su cui sono state valutate le performance del modello viene sottocampionato per poter bilanciare le due classi ed effettuare un confronto. Per ogni osservazione appartenente al test set è stata predetta la probabilità di churnare con lo scopo di andare a verificare statisticamente se effettivamente le predizioni del modello corrispondessero alla realtà.
Il test set è stato ordinato in ordine decrescente in base all'indicatore di *Propensity* al churn, successivamente il set è stato suddiviso in decili ottenendo i seguenti risultati:

```
Decile 100% :

Propensity to churn minimo: 76.96
Propensity to churn massimo: 96.12

Percentuale di churners: 92.44
Percentuale di non churners: 7.56

-----------------------------------

Decile 90% :

Propensity to churn minimo: 47.55
Propensity to churn massimo: 76.96

Percentuale di churners: 83.33
Percentuale di non churners: 16.67

-----------------------------------

Decile 80% :

Propensity to churn minimo: 27.91
Propensity to churn massimo: 47.53

Percentuale di churners: 64.65
Percentuale di non churners: 35.35

-----------------------------------

Decile 70% :

Propensity to churn minimo: 21.55
Propensity to churn massimo: 27.87

Percentuale di churners: 56.06
Percentuale di non churners: 43.94

-----------------------------------

Decile 60% :

Propensity to churn minimo: 16.93
Propensity to churn massimo: 21.46

Percentuale di churners: 47.98
Percentuale di non churners: 52.02

-----------------------------------

Decile 50% :

Propensity to churn minimo: 14.78
Propensity to churn massimo: 16.93

Percentuale di churners: 41.16
Percentuale di non churners: 58.84

-----------------------------------
```

Figura 2: Primi 6 decili e Propensity

```
Decile 40% :

Propensity to churn minimo: 11.85
Propensity to churn massimo: 14.73

Percentuale di churners: 36.36
Percentuale di non churners: 63.64

----------------------------------

Decile 30% :

Propensity to churn minimo: 10.11
Propensity to churn massimo: 11.83

Percentuale di churners: 28.28
Percentuale di non churners: 71.72

----------------------------------

Decile 20% :

Propensity to churn minimo: 8.21
Propensity to churn massimo: 10.11

Percentuale di churners: 28.54
Percentuale di non churners: 71.46

----------------------------------

Decile 10% :

Propensity to churn minimo: 1.74
Propensity to churn massimo: 8.2

Percentuale di churners: 21.23
Percentuale di non churners: 78.77

----------------------------------
```

Figura 3: Ultimi 4 decili e Propensity

Nel primo decile si può facilmente notare che ad una percentuale estremamente alta di churners (%92.44) corrispondono valori di Propensity to churn molto alti.
Guardando i decili dall'ultimo verso il primo, si nota che al crescere del range min-max dell'indice di Propensity to churn corrisponde un aumento della percentuale di churners presenti all'interno del decile. Tale comportamento evidenzia in effetti la capacità del modello di indivudare il ricercato "segnale", pur essendo debole.

5 Conclusioni e Sviluppi Futuri

L'obiettivo dell'analisi è stato quello addestrare un modello in grado di individuare un segnale che dia informazioni sul "livello di propensità" di un utente ad abbandonare il servizio analizzando il suo comportamento. Dopo aver adottato diverse tecniche di campionamento per risolvere il problema della classe sbilanciata, dopo aver confrontato le performance dei due modelli selezionati e dopo aver individuato il modello più adatto al task, è stato calcolato l'indice di Propensity to churn. Per poter valutare l'affidabilità dell'indice calcolato, è stato condotto un test su un campione di dati che ha evidenziato come effettivamente il modello sia in grado di individuare il segnale ricercato.
Il segnale individuato è comunque un segnale debole, uno sviluppo futuro di questa analisi dunque potrebbe essere la validazione del modello e il test su dati più recenti e raccolti in un range temporale più ampio, in quanto la grande sfida dei modelli anti-churn è proprio quella di mantenere alte le performance nel tempo e con dati raccolti in istanti temporali successivi e distanti dall'addestramento dello stesso.

6 Librerie Utilizzate

- seaborn;
- pandas;
- matplotlib;
- numpy;
- scikit-learn;
- glob;
- os;
- pathlib;
- tensorflow;
- tensorflow_addons;
- keras;
- mlxtend;
- scipy;
- statistics.

7 Bibliografia

Di seguito vengono elencate le librerie utilizzate per lo svolgimento del progetto:

- `https://seaborn.pydata.org`;
- `https://pandas.pydata.org/docs/`;
- `https://matplotlib.org`;

- https://numpy.org/doc/;
- https://scikit-learn.org/stable/modules/;
- https://docs.python.org/3/library/glob.html;
- https://docs.python.org/3/library/os.html;
- https://docs.python.org/3/library/pathlib.html;
- https://www.tensorflow.org/api_docs/python/tf/all_symbols;
- https://www.tensorflow.org/addons;
- https://keras.io/api/;
- http://rasbt.github.io/mlxtend/;
- https://scipy.org/;
- https://docs.python.org/3/library/statistics.html;