Customer Churn Detection

Gadda Alberto, ID: 824029

CdLM Data Science, Università degli studi di Milano Bicocca

INTRODUCTION

La churn prediction è un task che consiste nell'identificazione dei clienti con un alto rischio di lasciare la compagnia o di cancellare la sottoscrizione ad un servizio. Riuscire a prevedere con il dovuto anticipo il comportamento di un cliente permette alle aziende di applicare in maniera efficace strategie di retention con il fine di evitare l'abbandono. L'acquisizione di un nuovo cliente si stima essere fino a 6 volte più costosa rispetto alle strategie tipicamente utilizzate per la retention [1]. E' per questo motivo dunque che, sviluppando un accurato modello di churn detection, un'azienda può mantenere la maggior parte della sua customer base ed evitare di incorrere nei costi di estensive campagne di customer acquisition per ricostruirla.

In questo elaborato si è deciso di sviluppare un metodo finalizzato alla prevenzione di questo evento sfavorevole, utilizzando principalmente due diverse famiglie di modelli di Machine Learning: le *feed-forward* Neural Network e i Tree Based Models.

Si è utilizzato un dataset raccolto da un'azienda che opera nel settore delle telecomunicazioni, all'interno del quale hanno raccolto informazioni sui propri clienti, monitorandoli nel tempo e prendendo nota di quelli che hanno commesso churn.

Sono state esplorate diverse soluzioni al problema dello sbilanciamento delle classi (Over Sampling, Under Sampling, Anomaly Detection), e i risultati ottenuti sono stati confrontati sulla base di metriche pensate *ad hoc* per misurare la bontà del modello nella situazione di interesse.

Oltre a cercare di identificare il miglior modello rispetto a queste metriche, si è anche provato a valutarli in relazione ad un punto di vista economico, in modo da poter alla fine stimare se il modello sviluppato possa portare un vantaggio all'azienda.

Si è infine cercato effettuare *model explainability* con il fine di analizzare quali sono le feature più legate al fenomeno del churn.

DATASET

Il dataset in esame contiene dati appartenenti a clienti di un'azienda che lavora nel settore delle comunicazioni. Gli utenti sono stati monitorati durante l'utilizzo dei servizi offerti dall'azienda sulle sue piattaforme web e sono stati raccolti dati sulle loro abitudini e preferenze (e.g. i temi di interesse più cercati su Google e le fasce orarie nelle quali sono risultati più attivi). Questa operazione di profilazione degli utenti da parte dell'azienda ha permesso di creare una ricca base di dati finalizzata ad una migliore comprensione della propria clientela. E' a partire da questo dataset che saranno dunque svolte le analisi per determinare un modello efficace di churn detection.
Prima di procedere è utile introdurre più dettagliatamente la struttura del dataset e le variabili in esso contenute.
Il dataset è composto da 330586 righe e 102 colonne. Trattandosi di un task di classificazione supervisionata, una feature rappresenta la variabile target da prevedere (*Pdisc*). Si tratta di una variabile dicotomica che assume il valore 1 quando il cliente è un churner e il valore 0 altrimenti. Le rimanenti sono variabili esplicative, raggruppabili come segue:

- **external_id:** Codice identificativo del cliente.
- **how_many_ok_urls:** Numero di pagine web correttamente analizzate dal motore semantico durante la raccolta dati finalizzata alla composizione di questo dataset.
- **how_many_ko_urls:** Numero di pagine web che non sono potute essere analizzate a causa di problemi.
- **os_:** Sistema operativo utilizzato dall'utente. Insieme di colonne one-hot encoded.
- **browser_:** Browser utilizzato dall'utente. Insieme di colonne one-hot encoded.
- **feriale:** Percentuale $[0, 1]$ di pagine visitate nei giorni feriali della settimana dall'utente.
- **weekend:** Percentuale $[0, 1]$ di pagine visitate nei giorni festivi della settimana dall'utente.
- **L x_y:** Percentuale $[0, 1]$ di testi compresi tra la lunghezza x e la lunghezza y tra quelli visitati dall'utente.
- **categories_:** percentuale $[0, 100]$ di pagine appartenenti alla categoria _, tra quelle visitate dall'utente.
- **admants_:** categorie semantiche di cui non è noto il significato.
- **CINEMA, CALCIO, SPORT ... :** Insieme di variabili dummy che indicano se l'utente ha attivato il servizio di riferimento.
- **FLG:** Indica la tipologia di abbonamento scelto dall'utente.
- **STB:** Indica la tipologia di decoder utilizzato dall'utente.

- **DATA_RIF:** Data in cui è stata effettuata la rilevazione.

Il dataset è stato diviso in training, validation e test set con un campionamento casuale stratificato rispetto alle variabili *Pdisc* e *DATA_RIF*. Questa scelta è stata necessaria a causa della natura sbilanciata del problema: i churner sono solo una piccola percentuale (3%) dell'intera clientela, ed è necessaria una loro presenza omogenea all'interno delle diverse partizioni.

Il training set è stato utilizzato per l'addestramento dei diversi modelli con diverse configurazioni di parametri, mentre il validation set è stato utilizzato per scegliere il modello più promettente. Il test set è stato utilizzato esclusivamente nell'ultima fase per la verifica su dati vergini delle performance del modello migliore, in modo tale da stimare in maniera robusta le performance attese sui dati reali che il modello elaborerà quando verrà messo in produzione.

EDA AND DATA PREPROCESSING

E' noto come i dati in contesto aziendale siano spesso affetti da problemi di data quality. Per evitare dunque di incorrere in uno scenario di *"garbage in, garbage out"*, ossia in cui il modello produce performance insufficienti a causa di dati in input non adeguati, è necessario effettuare delle analisi esplorative sul dataset per comprenderne al meglio la natura e verificare la presenza di eventuali criticità. Queste analisi vengono svolte esclusivamente sul training set e ogni modifica viene poi riportata alla altre partizioni senza sfruttare informazioni relative alle distribuzioni di queste ultime.

Come parte di questo processo, in primo luogo sono state eliminate le colonne che assumono valori costanti in tutte le osservazioni, poiché non contengono di fatto alcun valore informativo. Le colonne eliminate per questa ragione sono le seguenti: tutte le colonne relative alle categorie *admants_*, *browser_chromium*, *browser_edge*, *browser_android*, *browser_ie*, *os_osx*, *os_bsd*, *categories_emotions*.
Per quanto riguarda la colonna *DATA_RIF* si nota che il periodo di osservazione è esclusivamente compreso tra Maggio 2017 e Settembre 2017. In Fig. 1 è possibile vedere la distribuzione dei churner nei diversi mesi.
Per catturare l'informazione del numero di servizi posseduti da ogni cliente è stata creata una variabile aggiuntiva con il conteggio dei pacchetti attivi (CINEMA, CALCIO, SPORT, SKY FAMIGLIA, ...). Sono stati inoltre riscontrati 6 casi in cui le variabili relative ai servizi assumono valore 2 (a differenza delle restanti osservazioni per le quali vengono assunti valori $\{0,1\}$). Non essendo disponibili ulteriori informazioni in merito, si è deciso di non alterare questo dato.

Osservando in Fig. 2 la distribuzione del numero di servizi attivi in relazione alla variabile target, è possibile notare

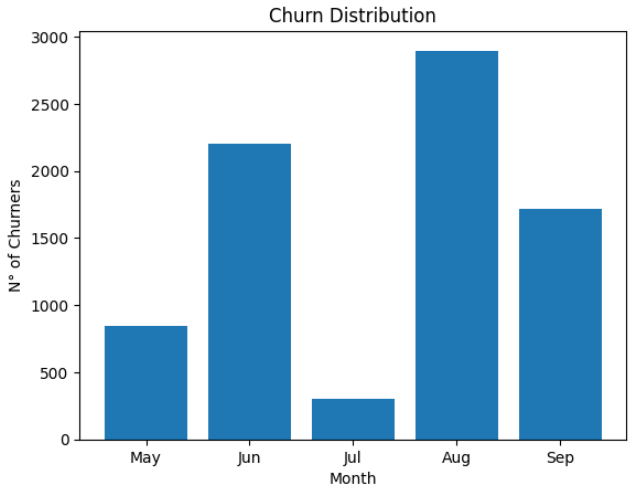

Figure 1. Distribuzione del numero di churner nei diversi mesi di rilevazione. Il minimo numeri di abbandoni registrato è a Luglio, il massimo ad Agosto.

una significativa differenza tra la media di servizi attivi dei churner (4.18) e quella dei clienti fedeli (5.85).

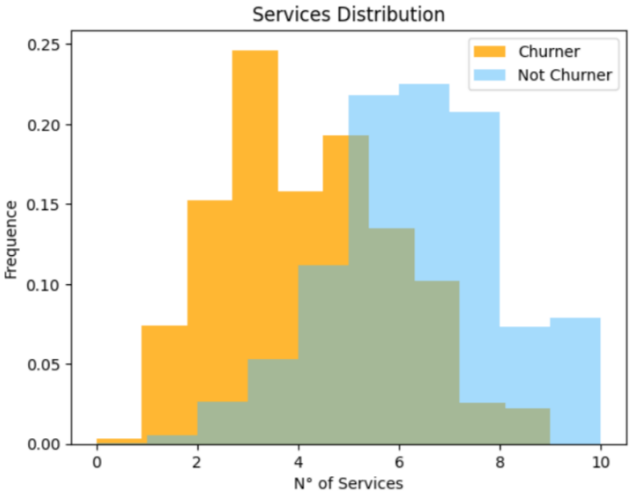

Figure 2. Distribuzione del numero di pacchetti attivi per le due diverse classi di utenti. Si nota che i churner assumo valori in media più bassi.

Per approfondire questo aspetto, si evidenzia in Fig. 3 il fatto che la percentuale di churner tra gli utenti che non hanno nessun pacchetto attivo è quasi di quattro volte superiore al corrispettivo con almeno un pacchetto attivo.

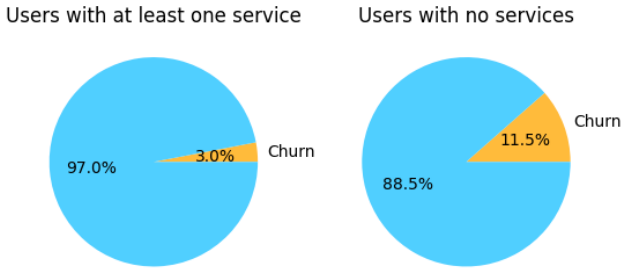

Figure 3. Confronto tra utenti che hanno almeno un pacchetto attivo e quelli che non ne hanno nessuno.

In tema di pulizia del dato sono stati osservate 2736 ripe-

tizioni dell'identificativo univoco (*external_id*), corrispondenti al 1.03% dei record. Quattro di queste osservazioni sono effettivamente record completamente identici su ogni colonna, pertanto sono stati rimossi in quanto non informativi. Le restanti ripetizioni hanno tutte la stessa data di riferimento, pertanto è stata esclusa l'ipotesi che si tratti di rilevazioni su uno stesso cliente fatte in tempi differenti, ma differiscono per i servizi attivati.

Vale la pena menzionare il fatto che la percentuale di churners nei record duplicati (6.3%) è più del doppio di quella osservata nel dataset completo (3.0%). Nonostante non siano disponibili informazioni sufficienti per approfondire le cause di questo fenomeno, una possibile spiegazione è che i churner siano soliti effettuare una modifica del proprio abbonamento prima di abbandonare completamente la piattaforma. Per non perdere traccia di questo pattern è stata creata una nuova variabile *Modified* che assume valore 1 se l'osservazione è un duplicato e 0 altrimenti.
Infine si sottolinea nuovamente il problema del forte sbilanciamento nella variabile target *Pdisc*, contenente solo il 3% di churner, che sarà affrontato nelle sezioni successive di questo elaborato.

La fase di preprocessing nel suo complesso ha prodotto un training set composto da 71 colonne e 264464 righe.

MODELS AND METHODS

Nella seguente sezione vengono introdotte le tecniche e i modelli che sono stati testati in questo elaborato.

Feed-Forward Neural Networks: Le reti neurali *feed-forward* [2] rappresentano un modello computazionale ampiamente utilizzato nell'ambito dell'apprendimento automatico e dell'intelligenza artificiale. Queste reti si ispirano alla struttura biologica e al funzionamento nel cervello umano in quanto utilizzano *neuroni*, unità computazionali interconnesse tra loro e organizzate in strati consecutivi.
Ogni neurone riceve degli input pesati dai neuroni del livello precedente, elabora queste informazioni utilizzando una funzione di attivazione non lineare e produce un output che viene passato ai neuroni del livello successivo. Questa propagazione dell'informazione avviene in maniera unidirezionale e aciclica, aspetto da cui deriva il loro nome.
L'apprendimento delle reti neurali *feed-forward* avviene attraverso un processo noto come *backpropagation* [3]. Questo algoritmo di apprendimento si basa sull'ottimizzazione dei pesi della rete al fine di ridurre l'errore tra la *ground truth* e la previsione prodotta dalla rete. Utilizzando metodi di ottimizzazione come la discesa del gradiente è possibile dunque di migliorare le sue prestazioni attraverso la minimizzazione di una funzione di costo (*loss*).
Le reti neurali *feed-forward* hanno dimostrato di essere strumenti potenti ed efficaci in diversi ambiti, tra cui il riconoscimento di pattern, la classificazione e il clustering [4]. L'adeguato design dell'architettura della rete, la scelta delle funzioni di attivazione e la corretta inizializzazione dei pesi sono tutti fattori di cruciale importanza nel determinare le prestazioni e il successo del loro utilizzo.

Tree Based Models: Una famiglia di approcci ampiamente utilizzati per problemi di classificazione e regressione sono i metodi basati su alberi, che prendono il nome dal modo in cui rappresentano e strutturano l'informazione. Ogni albero della famiglia infatti è composto da un insieme di nodi (foglie), collegati tra loro tramite archi (rami). Il nodo radice rappresenta l'intero set di dati, mentre i nodi foglia rappresentano le classi o i valori di output previsti.
La costruzione di un albero avviene attraverso una sequenza di suddivisioni (*split*) dello spazio in regioni più piccole e omogene sulla base delle caratteristiche dei dati. Queste suddivisioni sono determinate da criteri come la riduzione dell'impurità di Gini, l'entropia o l'errore quadratico medio. Attraverso un'analisi ricorsiva, l'algoritmo di apprendimento identifica i punti di separazione ottimali che massimizzano la purezza all'interno di ciascuna suddivisione.

Principal Components Analysis: L'Analisi delle Componenti Principali (*PCA*) [6] è una tecnica di riduzione della dimensionalità molto nota nell'ambito dell'analisi dei dati e dell'apprendimento automatico. Consiste nella creazione di un nuovo set di features incorrelate tra loro (chiamate componenti principali) che massimizzino la varianza spiegata a partire da combinazioni lineari delle variabili originali.
La *PCA* si basa sull'intuizione che la varianza spiegata sia una buona misura dell'importanza delle variabili. Questo permette di ordinare le componenti principali in maniera decrescente sulla base della quantità di varianza che spiegano. A questo punto è responsabilità dell'analista selezionare le prime $k << p$ componenti più significative, tenendo in considerazione il trade-off tra il numero di componenti escluse e la perdita di varianza spiegata che esse comportano.

Oversampling: Per affrontare la problematica dello sbilanciamento delle classi esistono numerose tecniche di oversampling, che puntano ad aumentare artificialmente il numero di campioni della classe minoritaria. Due delle tecniche più comunemente utilizzate per l'oversampling sono il Random OverSampling (*ROS*) [7] e il Synthetic Minority Over-sampling Technique (*SMOTE*) [8], entrambe focalizzate sull'aumentare la rappresentatività della classe minoritaria generando nuovi campioni artificiali.
ROS è un metodo semplice che duplica casualmente gli esempi esistenti della classe minoritaria, aggiungendoli al dataset. Questo processo di duplicazione consente di bilanciare le classi, ma può portare ad overfitting e ad una riduzione della capacità di generalizzazione del modello, specialmente in presenza di dataset di grandi dimensioni.
SMOTE è invece un approccio più sofisticato che genera nuovi campioni sintetici introducendo una combinazione lineare dei k record più simili a quello selezionato. Questo metodo preserva la struttura dei dati e introduce una maggiore varietà rispetto alla duplicazione semplice del ROS. In particolare, SMOTE evita il problema del overfitting e può produrre risultati migliori quando il dataset presenta pattern

complessi o distribuzioni non lineari.

Undersampling: L'undersampling si approccia al problema dello sbilanciamento delle classi in maniera opposta, riducendo le osservazioni della classe maggioritaria. Due delle tecniche più comunemente utilizzate sono il Random UnderSampling (*RUS*) [7] e il Near Miss [9].
RUS consiste nell'eliminazione casuale di un sottoinsieme di esempi dalla classe maggioritaria, migliorando così le performance relative alla classe minoritaria e la gestione dell'overfitting. Questa selezione casuale può tuttavia comportare l'eliminazione accidentale di osservazioni cruciali per l'identificazione del segnale presente nei dati.
Near Miss è invece un approccio più sofisticato, che seleziona i record della classe maggioritaria più vicini a quelli della classe minoritaria, così da facilitare l'identificazione dei *decision boundaries* per la classificazione. Near Miss può essere implementato in diverse varianti, come Near Miss-1, Near Miss-2 e Near Miss-3, che differiscono nel modo in cui calcolano la distanza e selezionano i record.

SHAP: Il metodo SHAP [13] è una tecnica per spiegare i modelli di Machine Learning che mira ad attribuire un "importanza delle caratteristiche" a ciascuna variabile di input utilizzata dal modello per prendere decisioni.
Il metodo SHAP si basa sulla teoria dei giochi e su un concetto chiamato "valore di Shapley", che viene utilizzato per assegnare un valore a ciascuna variabile di input in base al suo contributo nel risultato finale delle previsioni del modello.
Nella pratica, il metodo SHAP valuta le prestazioni del modello quando ciascuna variabile di input è presente e quando non lo è, utilizzando queste informazioni per determinare quanto ciascuna variabile contribuisce alla previsione finale.
È importante notare che l'ordine delle variabili influisce nel calcolo dell'importanza: fornire una variabile al modello come prima o come ultima potrebbe cambiare drasticamente l'importanza stimata.
Per questo motivo, il valore di Shapley viene ottenuto prendendo una media ponderata di tutte le possibili combinazioni di ciascun insieme di variabili.

METRICS

Il problema che si affronta in questo lavoro è un task di classificazione binaria, tuttavia a causa della natura sbilanciata delle classi è necessario analizzarlo adeguatamente, con le metriche che meglio evidenziano gli obiettivi desiderati.
Le metriche solitamente più utilizzate, come accuracy, precision e recall non sono in questo caso particolarmente informative, quindi si è deciso di creare una nuova metrica per valutare le performance dei modelli.

La metrica creata, che da qui in poi chiameremo *Churner Density in Deciles (CDD)*, premia i modelli per i quali, le persone che sono effettivamente churner, ricadono tra quelle con le probabilità predette più alte.
Prendendo il vettore delle probabilità predette, ordinandolo in maniera decrescente, e dividendolo in intervalli (10), andiamo ad analizzare quale percentuale di churner è contenuta in ognuno di essi.

La metrica scelta è descritta dall'equazione che segue:

$$CDD(y\_true) = \sum_{i=0}^{9} \frac{1}{i+1}(y\_true[i \cdot \frac{n}{10}; (i+1) \cdot \frac{n}{10}]) \quad \textbf{(1)}$$

dove:

- y_true è il vettore di true label, ordinato utilizzando gli indici dell'ordinamento delle probabilità predette in maniera descrescente.
  Si ottiene infatti prendendo le true label di tutte le osservazioni, partendo da quella con probabilità predetta dal modello di essere churner più alta, fino alla più bassa.
  Chiaramente ci si aspetta che le prime posizioni del vettore siano occupate da effettivi churner.

- $y\_true[a; b]$ indica la percentuale di churner presenti nella porzione di vettore che va dall'indice a fino all'indice b.

- n è il numero totale di soggetti presi in considerazione.

La formula è composta dalla sommatoria nei diversi intervalli di $y\_true[i \cdot \frac{n}{10}; (i+1) \cdot \frac{n}{10}]$, ovvero la percentuale di churner presenti all'interno dell'intervallo, moltiplicato per $\frac{1}{i+1}$, che pesa in maniera man mano meno influente gli intervalli. I churner del primo intervallo $(i = 0)$ peseranno infatti $\frac{1}{0+1} = 1$, quelli del secondo $(i = 1)$ $\frac{1}{1+1} = \frac{1}{2}$, quelli del terzo $(i = 2)$ $\frac{1}{2+1} = \frac{1}{3}$ e così via.

In questo modo vengono premiati i modelli che coglono più churner nei primi decili. I modelli che otterrano un valore più alto di questa metrica saranno quelli che soddisfano maggiormente le nostre richieste.

Questa metrica, viene accompagnata da un grafico che mostra la quantità di churner presenti nei vari decili, in maniera da fornire al lettore un'informazione più completa per poter comprendere meglio la situazione.

MODEL SELECTION

Come prima cosa è stato necessario selezionare il modello più performante sui dati a disposizione, e trovare la sua configurazione ideale.
Diversi tipi di approcci sono stati valutati, utilizzando la metrica *CDD* accompagnata da un grafico che mostra la distribuzione dei churner nei vari decili.

PCA: Dato l'alto numero di variabili esplicative presenti nel training set (71), le aspettative erano alte sulla capacità della Principal Components Analysis di ridurre in maniera efficace il numero di features da utilizzare nei modelli.

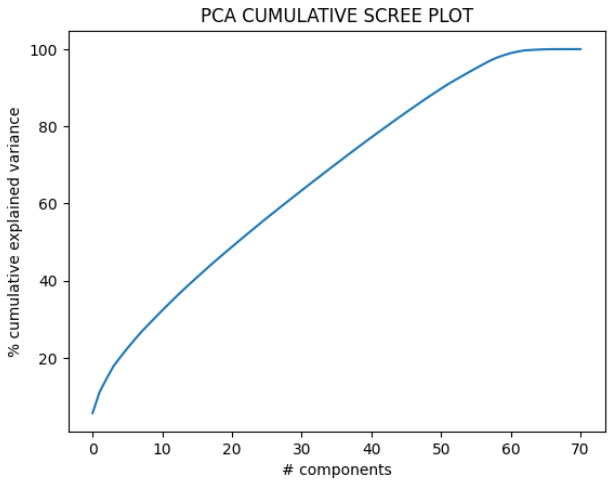

Figure 4. Varianza cumulata spiegata dalle componenti principali.

Tuttavia osservando lo Scree Plot Cumulato in Fig. 4, che mostra il trade-off tra il numero di componenti principali da selezionare e la varianza cumulata da esse spiegata, si evince che per preservare il 70% di quest'ultima sarebbe necessario mantenere le prime 35 componenti.
Osservando il grafico è possibile anche notare l'assenza del classico *gomito*, che indichi il numero di componenti principali che conviene mantenere.

Le ragioni di una limitata capacità da parte della PCA nel ridurre la dimensionalità sono probabilmente imputabili alla scarsità di correlazione lineare che intercorre tra le variabili esplicative originali. Per i motivi appena descritti, uniti al fatto che si sarebbe persa ogni possibilità di spiegabilità del modello, questo approccio di preprocessing è stato scartato, continuando con le variabili originali.

Feed-Forward Neural Networks: Per risolvere il task di classificazione, la prima famiglia di modelli valutata è stata quella delle Feed-Forward Neural Networks. Nello specifico sono state valutate due diverse reti, una più semplice (Net) e una sua variante più complessa ($Net_{complex}$).
Entrambe sono composte da una sequenza di layer densi, seguiti da una funzione di attivazione *ReLU* e da un layer di *dropout*, che congela casualmente alcuni neuroni durante l'addestramento per rendere la rete più robusta e dunque più resistente all'overfitting.
L'ultimo layer ha come funzione di attivazione la *Sigmoid*, che restituisce un output finale nell'intervallo $[0,1]$. L'utilizzo della sigmoide permette di interpretare l'output come una misura di probabilità che un record corrisponda ad un churner.

La differenza tra queste due reti sta nella quantità e nella dimensione di hidden layers densi che le compongono. $Netcomplex$ è composta da 9 hidden layers, rispettivamente di dimensione {128, 256, 512, 1024, 512, 256, 128, 64, 32}, mentre Net è composta solamente da 3 hidden layers, di dimensione { 128, 64, 32}.
Entrambe le reti sono state addestrate per 100 epoche

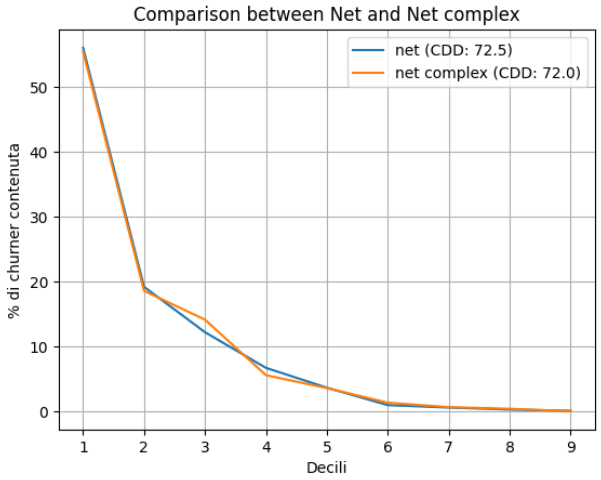

Figure 5. Confronto tra le performance sul test set di Net e Net Complex.

utilizzando *Adam* come ottimizzatore, che minimizza la *Binary Crossentropy*. E' stata utilizzata anche la tecnica di *Early Stopping* per terminare l'addestramento in caso di convergenza, definita in questo elaborato come 10 epoche consecutive in cui non si registri una riduzione della *Loss*.

Net ha prodotto performance leggermente superiori rispetto a $Net_{complex}$, e dunque si è deciso di scegliere questo tipo di architettura meno complessa. In Fig. 5 si mostra il grafico del confronto tra i due modelli.

Oversampling e Undersampling: Avendo ottenuto le performace di un primo modello, addestrato sui dati originali, si è deciso a questo punto di valutare la possibile efficacia delle tecniche di *Oversampling* e *Undersampling* descritte precedentemente.

In Fig. 6 si mostrano i risultati apportati dalle tecniche di *Oversampling*, mentre in Fig. 7 si mostrano i risultati apportati dalle tecniche di *Undersampling*.

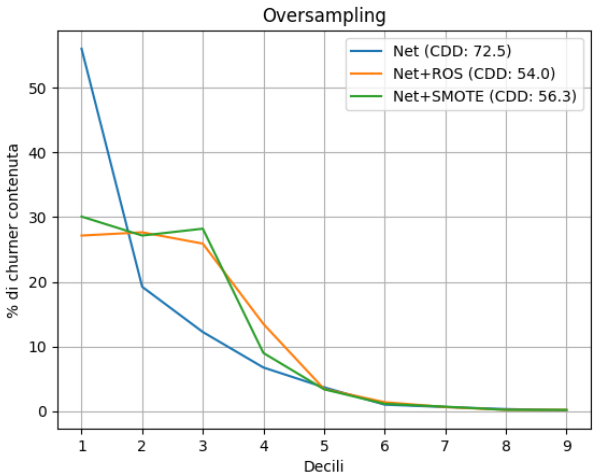

Figure 6. Confronto tra le performance di Net, sui dati originali e sui dati ottenuti tramite Oversampling.

Dal momento che queste tecniche sono risultate significativamente peggiorative, si è deciso di non utilizzarle, procedendo

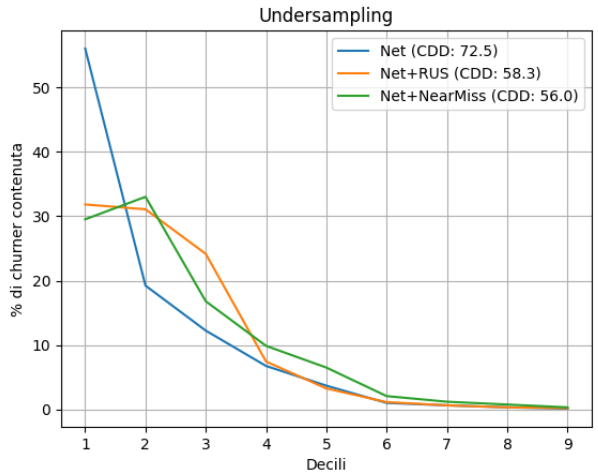

Figure 7. Confronto tra le performance di Net, sui dati originali e sui dati ottenuti tramite Undersampling

con il dataset originale.

Tree Based Models: Per quanto riguarda le tecniche Tree-based i modelli scelti da essere valutati sul test set sono *RandomForest* [10] e *XGBoost* [11].
Dopo avere provato diverse combinazioni di parametri, si è scoperto che *RandomForest* non riesce ad eguagliare le performance di Net, mentre *XGBoost* le supera, seppur di poco. I grafici relativi a questo confronto sono riportati in Fig. 8.

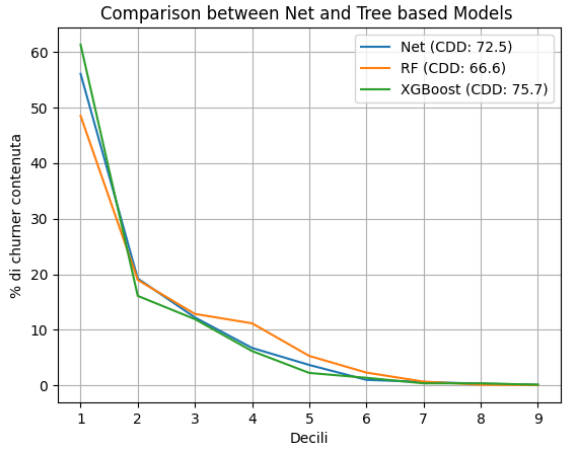

Figure 8. Confronto tra le performance di Net e dei modelli Tree-Based.

Il modello selezionato come migliore è dunque $XGBoost$, con $n\_estimators = 100$, $learning\_rate = 0.1$ e $max\_depth = 7$.

THRESHOLD SELECTION

Nella sezione precedente si è scelto il modello più performante, che riesce ad assegnare le probabilità di essere churner, in maniera più fedele alla realtà.
A questo punto serve però un metodo per definire quali sono i clienti ai quali indirizzare una campagna di retention.
Questo viene fatto selezionando la threshold ottimale, in modo che i clienti con probabilità predetta superiore ad essa vengano classificati come churner.

Si è pensato di effettuare questa scelta in relazione all'aspetto economico che deriva da tale scelta, e per questo motivo è necessario fare delle supposizioni.
Partendo dal rapporto $(1 : 5)$ menzionato in precedenza, si ipotizza che una campagna di retention possa costare all'azienda $10€$ e che il costo atteso dell'acquisizione di un nuovo cliente sia $50€$. Si assume in seguito che ad ogni previsione positiva sia associato un tentativo di retention che non per forza avrà successo (supponiamo una probabilità di successo pari a 70%).

A questo punto possiamo calcolare il costo che l'azienda dovrà sostenere per mantenere invariata la numerosità della propria customer base:

- $\mathbb{E}[C_{FP}] = 10€$
- $\mathbb{E}[C_{FN}] = 50€$
- $\mathbb{E}[C_{TN}] = 0€$
- $\mathbb{E}[C_{TP}] = (0.7 \cdot 10€) + (0.3 \cdot 60€)$, siccome in caso di fallimento nella retention (costata $10€$) incorrono anche i costi della acquisizione di un nuovo cliente (al costo di $50€$), per un costo finale di $60€$.

In Fig. 9 si mostra come il variare della soglia scelta influisca sul costo complessivo l'azienda dovrà sostenere.
Emerge che il minimo di questa funzione è situato nel punto $X = 0.16$, di conseguenza al modello scelto nella sezione precedente forniremo come regola decisionale *threshold = 0.16*.

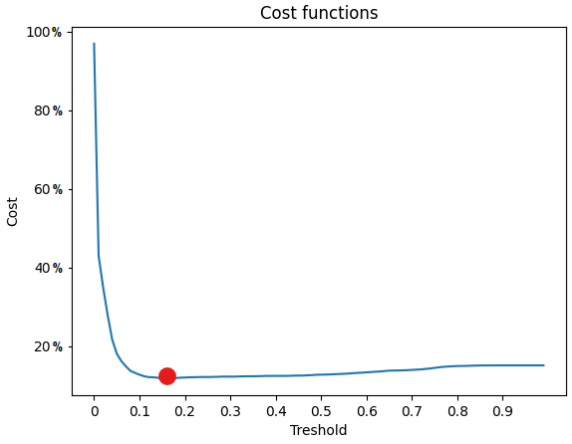

Figure 9. Costo al variare della threshold

RESULTS ON TEST SET

Tutte le analisi e le decisioni prese fino a questo punto sono state studiate sfruttando esclusivamente training set e validation set.

Il modello scelto viene a questo punto valutato anche sul test set, per poter capire se è stato creato overfitting nelle fasi

precedenti.
In Fig. 10 viene mostrato il grafico usato in precedenza per mostrare le performance dei diversi modelli, questa volta proponendo i risultati di XGBoost sul test set.
Notiamo che le performance rimangono praticamente identiche a quelle ottenute sul validation set.

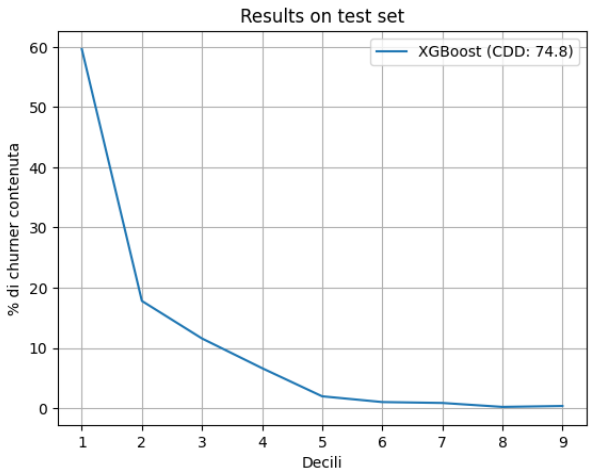

Figure 10. XGBoost valutato sul test set

Si decide infine di valutare, dal punto di vista economico, quanto l'introduzione di tale modello possa essere utile per l'azienda.

Per poter confrontare il modello con una baseline si ipotizza che, in assenza del metodo sviluppato, l'azienda avrebbe optato per un campionamento casuale semplice: la campagna di retention sarebbe stata sottoposta casualmente al 3% della customer base.
Tale approccio avrebbe comportato un costo di $117K$, mentre quello proprosto in questo elaborato ne comporta uno di $92K$, portando ad un risparmio di oltre $25K$ (*-21%*).

Questi valori sono calcolati in relazione alla numerosità del test set ($66K$ clienti) e chiaramente se il numero di utenti presi in considerazione fosse diverso la stima del risparmio si modificherebbe di conseguenza.

MODEL EXPLAINABILITY

Il modello selezionato, nella sua configurazione più performante, viene analizzato tramite l'algoritmo *SHAP* per capire quali sono le features più collegate al fenomeno del churn.
Emerge che le due variabili più significative risultano essere, in maniera molto netta, il mese e il numero di servizi attivi.

Per rendere il risultato più robusto, si è deciso di svolgere questo tipo di analisi anche sfruttando altre due tecniche: la feature importance che fornisce direttamente *XGBoost* e il pacchetto *LIME (Local Interpretable Model-Agnostic Explanations)*, che hanno fornito risultati coerenti, posizionando le stesse variabili ai primi posti.

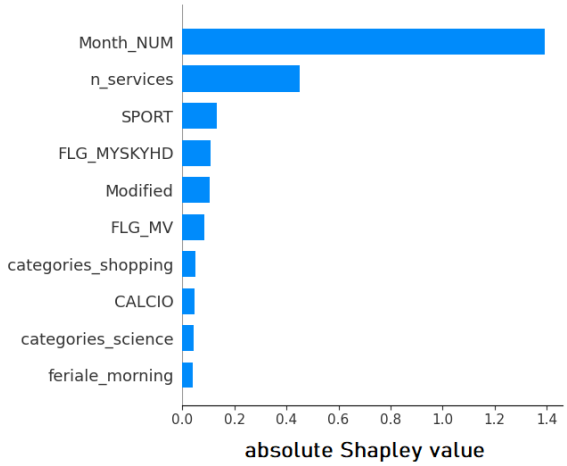

Figure 11. Top-10 feature importance

Da questa informazione l'azienda potrebbe decidere di prendere provvedimenti mirati alla risuluzione di questo problema, per esempio cercando di fornire contenuti particolarmente interessanti nel mese di Agosto (quello con tasso di churn più alto) e cercando di monitorare maggiormente i clienti che hanno pochi o addirittura zero pacchetti attivi.

CONCLUSION

In questo elaborato si è sviluppato un metodo che potrebbe aiutare un'azienda nell'identificazione dei churner all'interno della propria customer base.
I risultati ottenuti sono coerenti con gli obbiettivi di business definiti nel corso dell'elaborato. La ricerca di un modello di churn detection si è infatti svolta non solo sulla base delle metriche di performance, ma considerando anche il costo effettivo derivato dai vari possibili scenari previsti dal dominio applicativo.

Il modello ricavato si è dimostrato in grado di identificare circa il 60% dei churner nel primo decile di previsioni.
Questo risulta in un risparmio economico significativo per azienda (-21%), rispetto a metodologie più rudimentali che non sfruttano le moderne tecniche di Machine Learning.
I risultati ottenuti sono robusti, poichè confermati su un set di dati mantenuto intonso fino alla valutazione finale.

Si è infine mostrato che le variabili più collegate al fenomeno del churn, sono il mese e il numero di pacchetti attivi, fornendo un'informazione molto importante per l'azienda.

BIBLIOGRAPHY

1. https://ptgmedia.pearsoncmg.com/images/9780137058297/samplepages/9780137058297.pdf
2. https://dergipark.org.tr/en/pub/aupse/article/890416
3. https://www.sciencedirect.com/science/article/abs/pii/B9780127412528500108
4. https://www.ibm.com/topics/neural-networks
5. http://www.r-project.it/_book/alberi-di-classificazione-classification-trees-ctree.html
6. http://oldwww.unibas.it/utenti/dinardo/pca.pdf
7. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/
8. https://www.jair.org/index.php/jair/article/view/11192
9. https://link.springer.com/chapter/10.1007/978-3-540-37256-1_89
10. https://link.springer.com/chapter/10.1007/978-3-642-34062-8_32
11. https://cran.microsoft.com/snapshot/2017-12-11/web/packages/xgboost/vignettes/xgboost.pdf
12. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
13. https://shap.readthedocs.io