Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Final Project

Airbnb New User Bookings

Authors:
Licciardello Matteo - 799368 - m.licciardello@campus.unimib.it
Manenti Federico - 790032 - f.manenti3@campus.unimib.it

May 13, 2020

Abstract

L'utilizzo di tecniche di apprendimento automatico volte alla classificazione rappresenta uno sviluppo fondamentale per i business di tutto il mondo, ancor più se questo opera attraverso modalità esclusivamente digitali e deve sfruttare solamente i dati raccolti, siano essi specifici o generici, per poter soddisfare al meglio un bisogno dell'utente. In questo elaborato verranno presentati dei possibili algoritmi per la previsione della destinazione della prima prenotazione che un nuovo utente effettuerà sulla piattaforma Airbnb, partendo dalla risoluzione del problema con un algoritmo di previsione noto e proseguendo affinando le tecniche di classificazione e manipolazione dei dati utilizzate. Infine viene proposta un'analisi dei risultati e delle features che risultano più importanti per ottenere quel risultato.

1 Introduzione

Lo sviluppo della digitalizzazione, in ambito turistico, ha portato alla nascita di agenzie di viaggio online, altresì dette OTA (*Online Travel Agencies*) che permettono di prenotare un viaggio, o parte di esso, in modo semplice e rapido. Una delle società di punta appartenenti a questo business è *Airbnb*: un *marketplace* online di alloggi, di proprietà di agenzie locali di terze parti iscritte al servizio, alle quali offre visibilità senza interferire nella gestione delle strutture. L'offerta di Airbnb consiste di oltre 100.000 possibili destinazioni distribuite in 220 paesi nel mondo, con un bacino d'utenza che supera i 750 milioni di utenti. [1]

Il seguente progetto è ispirato ad una competizione avviata sul sito Kaggle denominata *Airbnb New User Bookings* [2]. La sfida proposta consiste nella previsione di alcune specifiche prenotazioni che verranno effettuate dagli utenti di Airbnb. In particolare, si vogliono prevedere i paesi che i nuovi utenti iscritti alla piattaforma sceglieranno come loro prima destinazione. Tale pratica è resa possibile dalla grande quantità di dati raccolti e possiede una duplice utilità: dal punto di vista aziendale effettuare una corretta previsione delle intenzioni dell'utente permette di mostrare a quest'ultimo le offerte a cui è più interessato e con maggiori probabilità di acquisto, di conseguenza incrementare il numero delle prenotazioni; dal punto di vista dell'utente invece mettere in evidenza l'offerta che più gli si addice rappresenta la possibilità di soddisfare un suo bisogno, ancor prima di averlo esplicitamente espresso.

Nel task affrontato, per ogni utente è possibile effettuare un massimo di 5 previsioni sulla destinazione della sua prima prenotazione. Ciò permette di identificare il compito svolto come un problema di *classificazione muti-etichetta* [3]. Le diverse previsioni riportate, per ogni utente, vengono valutate in base all'ordine che possiedono: infatti la metrica utilizzata nella competizione è la *Normalized Discounted Cumulative Gain* (NDCG [4]) definita come segue:

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{1}$$

- k valore massimo fissato a 5, per ogni utente possono essere restituite fino a cinque previsioni ordinate in ordine decrescente di probabilità
- rel_i rilevanza della previsione alla posizione i, assume valore 1 se corrisponde alla *ground truth*, altrimenti assume valore 0

$$NDCG_k = \frac{DCG_k}{IDCG_k} \tag{2}$$

- $IDCG_k$ rappresenta il massimo DCG ottenibile data una query
- $NDCG$ assume sempre un valore compreso tra 0 ed 1, più questo coefficiente cresce e più risulta alta la rilevanza della previsione

La metrica descritta verrà successivamente utilizzata per valutare la bontà della manipolazione dei dati effettuata e delle capacità di previsione dei modelli proposti confrontando le previsioni ottenute rispetto ai dati già in nostro possesso. La metodologia seguita per ottenere delle buone previsioni comporta una prima fase di *preprocessing* dei dati volta a migliorarne la qualità e renderli il più puliti ed informativi possibile. Successivamente è stato possibile analizzare e processare i dati a disposizione tramite diverse tecniche di *machine learning*, sono quindi stati selezionati tre classificatori sui quali effettuare le prove: *Support Vector Machine* [5], *XGBoost* [6] e *Neural Network* [7]. Infine, agli ultimi due modelli è stata applicato un processo di *AutoML* per poter ottimizzare gli iper-parametri sulla base dei risultati ottenuti.

2 Dataset

I dati utilizzati sono stati caricati sul post di Kaggle relativo alla competizione [2] e si riferiscono ai soli clienti statunitensi dal 2010 al 2015 della società, in particolare vengono forniti sei *dataset* in formato *CSV*:

1. *train_users.csv* contenente informazioni relative agli utenti, alla registrazione sul sito ed alla destinazione scelta
2. *test_users.csv* come il precedente, contenente informazioni relative agli utenti ed alla registrazione sul sito, ma non alla destinazione scelta
3. *sessions.csv* contenente informazioni relative alle sessioni di navigazione sul sito da parte degli utenti
4. *countries.csv* contenente informazioni statistiche sulle destinazioni scelte dagli utenti
5. *age_gender_bkts.csv* contenente statistiche riguardanti le informazioni personali dichiarate dagli utenti in fase di registrazione
6. *sample_submission.csv* dataset che stabilisce il formato della consegna da utilizzare su Kaggle

Dei dataset riportati, solo *train_user.csv* e *sessions.csv* venegono utilizzati per la costruzione dei modelli in quanto gli altri non risultano di interesse per il task affrontato. In particolare il dataset *test_user.csv* inizialmente non è stato sfruttato a causa della mancanza della variabile target e quindi l'impossibilità di ottenere un riscontro immediato della metrica *NDCG*; a fine progetto però è stato utilizzato, dopo esser stato elaborato nello stesso modo del *train_set*, per creare previsioni e confrontare il punteggio ottenuto sulla *leaderboard* relativa della piattaforma Kaggle.

Train_users.csv

Il dataset, contenente 213.452 record, è stato selezionato come punto di partenza per l'analisi dei dati in quanto in esso sono contenute tutte le informazioni di base riguardanti gli utenti. Si tratta di 16 attributi così descritti:

- *id*: identificativo dell'utente
- *date_account_created*: data di creazione dell'account Airbnb
- *timestamp_first_active*: data ed ora della prima attività dell'utente
- *date_first_booking*: data della prima prenotazione
- *gender*: variabile categorica che indica il genere dell'utente
- *age*: età dichiarata dall'utente

- *signup_method* e *signup_flow*: metodo di iscrizione e pagina da cui l'utente si è registrato
- *language*: preferenza linguistica dell'utente
- *affiliate_channel*: tipologia di marketing che ha portato alla registrazione dell'utente
- *affiliate_provider* e *first_affiliate_tracked* : informazioni di marketing
- *signup_app*: portale di accesso tramite il quale l'utente si è iscritto
- *first_device_type* e *first_browser*: primo dispositivo e applicazione web utilizzati per eseguire la prima attività
- *country_destination*: attributo **target** che indica la prima destinazione scelta dall'utente. Contiene informazioni relative a 10 possibili nazioni di destinazione, un valore *other* che contiene gli stati non esplicitamente rappresentati ed un ultimo valore *NDF* che rappresenta gli utenti che ancora non hanno effettuato una prenotazione. Il dataset a disposizione è evidentemente sbilanciato, come riportato in Appendice in Figura 1.

Sessions.csv

Il dataset relativo alle sessioni dell'utente contiene tutte le informazioni, dal 1/01/14 in avanti, riguardanti la navigazione dell'utente su ciascuna pagina del sito o dell'app. Nel suddetto dataset sono contenute 10.567.737 sessioni di navigazione descritte tramite 6 attributi:

- *user_id*: identificativo dell'utente
- *action*: azione effettuata dall'utente
- *action_type* e *action_detail*: rispettivamente tipologia e dettagli riguardanti l'azione effettuata dall'utente
- *device_type*: tipologia di dispositivo con cui l'utente ha avviato la sessione
- *secs_elapsed*: durata della sessione misurata in secondi

2.1 Preprocessing

La fase iniziale di esplorazione dei dataset si concretizza nel preprocessing dei dati a disposizione: questo insieme di operazioni include le fasi di pulizia

ed adattamento dei dati al fine di estrarre informazioni utili al miglioramento delle previsioni. Osservando i due dataset ci si accorge che c'è la possibilità, da parte di alcuni attributi categorici, di assumere molti valori diversi anche se questi hanno una frequenza molto bassa. Per porre rimedio si è deciso di scegliere una soglia sotto cui i record poco presenti vengono sostituiti con un generico valore *other*, così facendo si riducono di molto i possibili valori assumibili dalle variabili consentendo quindi la riduzione delle dimensioni del dataset quando le features categoriche verranno trasformate in variabili di tipo *dummy*.

2.1.1 Age

La prima considerazione viene effettuata sull'attributo relativo all'età degli iscritti. Si è notato che, in fase di registrazione, un gruppo ristretto di utenti ha impostato l'anno di nascita al posto dell'età: questi dati sono stati corretti effettuando una sottrazione tra la data della registrazione e l'anno di nascita dichiarato. Il secondo problema riguardante le età è la possibilità di inserire una data falsa: per poter effettuare una distinzione sensata tra età plausibili ed età fittizie si è fatto riferimento alle *policy* del servizio di Airbnb, il quale non permette l'iscrizione ad utenti di età inferiore ai 18 anni, pertanto si è deciso di trasformare in valori nulli l'età di tutti gli utenti inferiori alla soglia minima; lo stesso procedimento è stato effettuato per gli utenti che dichiarano un'improbabile età maggiore di 100 anni.

2.1.2 Date

La fase successiva consiste nell'estrazione di informazioni dalle date presenti nel dataset, a partire dall'uniformazione del formato. Successivamente, avendo a disposizione la data precisa sia del giorno di iscrizione alla piattaforma, sia del giorno in cui è stata eseguita la prima attività, per ogni utente è stato calcolato un nuovo attributo denominato *time_lag* che indica la differenza in secondi tra i due istanti temporali. Inoltre viene effettuato l'*encoding* degli attributi che possiedono una ciclicità intrinseca, ad esempio le 24 ore presenti in un giorno. Prendendo l'esempio appena fatto, il valore massimo assumibile dalle ore è 23 mentre la prima ora del giorno successivo è 0 ma ciò non significa che la differenza tra i due orari sia di 23 ore. Questa tecnica è volta ad uniformare la differenza nel valore del dato che si ha al termine del ciclo e viene messa in pratica applicando le seguenti formule:

$$X_{sin} = \sin\left(\frac{2 * \pi * X}{max(X)}\right) \tag{3}$$

$$X_{cos} = \cos\left(\frac{2 * \pi * X}{max(X)}\right) \tag{4}$$

Nelle formule (3) e (4) l'oggetto X considerato rappresenta il valore attuale del dato mentre *max(X)* corrisponde al massimo valore che esso può assumere. Le *features* del dataset a cui è stata applicata questa trasformazione sono relative a: i giorni della settimana, i giorni del mese e i mesi dell'anno. Le nuove features estratte rappresentano un'informazione totalmente sostitutiva di quella precedente, pertanto si rimuovono i vecchi attributi.

2.1.3 Informazioni di sessione

Tutti i dati riguardanti le sessioni di navigazione sono stati raccolti utilizzando lo stesso identificativo delle informazioni personali, questo permette di collegare ciascun utente ai dati raccolti durante le sessioni di navigazione sulla piattaforma. A partire dai dati raccolti, per ogni utente, viene calcolato il numero diverso di: azioni, tipo d'azione e dettaglio dell'azione effettuate sul portale di Airbnb, inoltre è stato calcolato il tempo di utilizzo per ogni dispositivo e quale sia il più utilizzato. Successivamente sono state create nuove variabili per ogni utente a partire da *secs elapsed* quali: somma, minimo, massimo, media, mediana, deviazione standard, skewness del tempo trascorso sulla piattaforma, giorni di pausa tra le sessioni, numero di pause lunghe (più di 300000 secondi) e brevi (meno di 3600) e infine numero di sessioni. Terminata l'estrazione delle informazioni, sono state unite al dataframe di base tramite un operazione *join* sulla variabile id.

2.1.4 Operazioni conclusive

Conclusa la fase di pulizia ed arricchimento dei dati viene applicata un'ulteriore operazione per preparare il dataset all'utilizzo da parte dei classificatori. Nello specifico, le variabili numeriche non intervallari vengono standardizzate sottraendo ad ogni valore la media dei valori di quell'attributo e successivamente dividendo per la deviazione standard:

$$X = \frac{X - \mu}{\sigma} \tag{5}$$

Le variabili categoriche vengono invece trasformate in variabili di tipo *dummy*, ovvero una serie di attributi binari che rappresentano tutti i possibili valori assumibili dalla variabile categorica. Quest'ultima operazione fa salire il numero di features del dataset a 404. Infine tutti i *missing values* vengono sostituiti da un valore noto.

3 Approccio metodologico

Come accennato precedentemente (Sezione 1), il problema da affrontare è una classificazione di tipo multi-etichetta, ovvero una tipologia di classificazione non binaria in cui la variabile obiettivo può assumere più di un valore. Durante la fase di pianificazione del progetto si è cercato, sia dalla letteratura sia attingendo a conoscenze pregresse, un algoritmo di machine learning supervisionato che meglio si adattasse al problema proposto: la selezione dell'algoritmo da utilizzare è pesantemente influenzata dall'evidente sbilanciamento dei valori della classe obiettivo, già riportato nell'Appendice in Figura 1, che comporta notevoli difficoltà nella restituzione della sola previsione corretta. Inoltre il codice è stato eseguito su *Google Colab*[8] che fornisce un numero limitato di risorse, per cui l'obiettivo, oltre a trovare un classificatore efficace, è utilizzare un modello leggero che non esaurisca le risorse messe a disposizione dalla piattaforma.
Al fine di approfondire al meglio i classificatori scelti, si eseguono una serie di test empirici iniziali in cui il dataset viene partizionato, in modo stratificato sulla variabile target, in un *train set* contenente l'80% dei dati ed in un *test set* contenente il restante 20%. Quest'operazione permette di evidenziare solo le grosse differenze tra gli algoritmi e non ha valenza statistica, ma permette una scrematura iniziale dei modelli che hanno comportamenti evidentemente peggiori di altri sia in termini di performance che di tempo.

Si è quindi deciso di partire da un algoritmo di tipo Support Vector Machine (Sezione 3.2) di facile implementazione, che fornisse una base di partenza, per poi migliorare valutando tutte le possibili alternative. Appurato che tale approccio fosse appena sufficiente per conseguire gli obiettivi della competizione e abbia un consumo troppo elevato di risorse in termini temporali, si è deciso di cambiare strategia ed applicare un modello basato su reti neurali (Sezione 3.3) pur nella consapevolezza che questa strategia potesse risultare la più influenzata dallo sbilanciamento della classe target. Per migliorare ulteriormente il risultato si è rivelato fondamentale adottare un

approccio basato su alberi decisionali, in particolare approfondendo l'utilizzo dell'algoritmo di classificazione XGBoost (Sezione 3.4). Tutti i classificatori sono stati valutati in base alla misura NDCG riportata nella Sezione 1.

3.1 Automated Machine Learning

L'*Automated Machine Learning* (abbreviato in AutoML), è un processo che, correttamente configurato, permette di automatizzare una parte del flusso di lavoro riguardante un progetto di machine learning, aiutando per esempio il *data scientist* nella selezione dell'algoritmo e nell'ottimizzazione dei suoi iper-parametri [9]. In questo progetto, più precisamente, è stato applicato *Hyperparameter Optimization* che permette, a differenza dell'approccio tradizionale in cui il data scientist è chiamato ad effettuare comparative tra i vari iper-parametri cambiandoli manualmente secondo la sua interpretazione dei risultati precedenti, di specificare inizialmente quali iper-parametri del modello siano oggetto di analisi e di definire uno spazio di ricerca in cui trovare, idealmente, la miglior combinazione possibile. Ciò è possibile sfruttando la *Bayesian Optimization*: un processo sequenziale per l'ottimizzazione globale di una funzione *black box*. Considerando il contesto del problema, si vuole massimizzare il risultato della funzione $NDCG$. A differenza di altri processi di *Hyperparameter Optimzation* come *Random Search* o *Grid Search* in cui le valutazioni della funzione obiettivo sono fatte con selezione casuale degli iper-parametri, nel primo caso, o semplicemente esplicitandoli dall'utente, nel secondo, nella *Bayesian Optimization* si specifica lo spazio di ricerca degli iper-parametri; inizialmente sono scelti alcuni punti (combinazione dei valori degli iper-parametri) per approssimare la funzione da ottimizzare e successivamente dopo ogni iterazione si seleziona un nuovo punto, grazie ad una *funzione di acquisizione*, che aiuta ad approssimare al meglio la funzione obiettivo e, idealmente, permette dunque di ottenere il massimo globale. Questo automatismo consente, a parità di tempo di lavoro, di testare combinazioni di iper-parametri più efficaci rispetto a *Grid e Random Search*, offrendo quindi una maggiore possibilità di ottenere migliori risultati.
In questo progetto, l'utilizzo di tale tecnica è stato consentito da *pyGPGO* [10] una libreria *Python* che consente di effettuare un'ottimizzazione bayesiana tramite diversi modelli surrogati (tra i quali *Gaussian Process* e *Random Forest*). Nel caso in esame si è utilizzato *Random Forest* in quanto alcuni iper-parametri risultano essere numeri naturali.
Oltre al modello surrogato è necessario scegliere la funzione d'attivazione

da utilizzare. La libreria *pyGPGO* ne offre diverse: nel progetto sono state utilizzate *Probability of improvement* e *Expected improvement*. Entrambe le funzioni sono molto sfruttate in letteratura, ma si differenziano nel fatto che la prima risulta sbilanciata rispetto all'*exploitation*, ossia cercare i nuovi punti in un piccolo spazio di ricerca vicino alla soluzione migliore corrente; la seconda invece è più bilanciata tra *exploitation* ed *exploration*, ossia la ricerca dei nuovi punti da proporre in aree meno esplorate della funzione.[11]

3.2 Support Vector Machine

Una SVM è un algoritmo di machine learning supervisionato che rappresenta i record del dataset in uno spazio n-dimensionale. Trovare la soluzione significa trovare gli iper-piani che separino al meglio gli elementi diversi, in altre parole si vogliono isolare in base al valore della variabile target i record. Il classificatore implementato è denominato *Support Vector Classifier* ed è stato prelevato dalla libreria *sklearn* messa a disposizione da *Scikit-Learn* [12].
Sul modello preso in considerazione sono stati effettuati numerosi test empirici, utilizzando sia kernel *lineare* che *RBF*, ma l'algoritmo, pur ottenendo risultati sufficienti, si è sempre dimostrato molto lento nell'esecuzione. Per questa ragione non è stato preso in considerazione per eseguire test più approfonditi con AutoML per la ricerca degli iper-parametri ottimali e i risultati non verranno riportati nella sezione dedicata.

3.3 Neural Network

La rete neurale è un modello di calcolo matematico composto da neuroni artificiali, interconnessi tra loro, i quali elaborano i segnali ricevuti e trasmettono il risultato ai nodi successivi. Le reti neurali in particolare richiedono un complesso meccanismo di addestramento per poter rendere al meglio: esso è composto da epoche, durante le quali i nodi coinvolti assumono dei pesi per trovare (al termine di ogni epoca) la migliore interpretazione possibile del segnale ricevuto. La stima della bontà dei pesi per ogni epoca è ottenuta valutando una funzione obiettivo, da minimizzare, chiamata *loss*. L'interpretazione viene riveduta e corretta, grazie al valore di loss ottenuto, al passare di ogni epoca, in modo da convergere verso un risultato finale che rappresenti la massima capacità di interpretazione dell'intero modello. Il processo è possibile grazie all'utilizzo dell'algoritmo di *backpropagation* in combinazione con un metodo di *ottimizzazione*.

La rete neurale è stata implementata a partire dalla libreria *Keras* [13] che mette a disposizione le funzioni per costruire, modificare ed addestrare una rete neurale. Dopo alcune prove preliminari ci si è resi conto che la struttura migliore per questo compito risulti essere una rete neurale *fully connected* senza layer di tipo *Dropout* e *BatchNormalization*. L'ottimizzatore scelto è *Adam*[14]. Inoltre, come riportato nell'Appendice in Figura 1, la variabile obiettivo risulta molto sbilanciata; per risolvere questo problema dunque è stata utilizzata una *matrice di pesi* calcolata grazie alla libreria *sklearn*.
Per evitare di incappare nel problema dell'overfitting è stata sfruttata la tecnica della *validation* ed inoltre è stata implementata la callbacks *Early Stopping* inclusa in Keras che permette di monitorare una quantità e stoppare il training della rete quando non è più rispettata una condizione, in questo caso quando il *validation loss score* smette di decrescere.
Successivamente, ritenendo questo algoritmo efficace e sufficientemente rapido nell'esecuzione, si è deciso di applicare AutoML per ottimizzare alcuni degli iper-parametri principali della rete utilizzando il seguente spazio di ricerca:

- Learning rate compreso tra 0.0001 a 0.01 (unica variabile continua)
- Numero di neuroni primo layer compreso tra 256 e 1024
- Numero di hidden layer compreso tra 0 e 3
- Numero di neuroni hidden layer (nel caso ci siano) compreso tra 16 e 1024
- Batch size compreso tra 1 e 128

L'esecuzione prevede una valutazione iniziale formata da 10 punti a cui si sussegue una fase di miglioramento con un budget di 70 iterazioni. Come già esposto, la funzione da ottimizzare è *NDCG* che viene valutata sul test set creato ad ogni iterazione dell'ottimizzazione bayesiana. L'intero processo viene ripetuto per le due diverse funzioni di acquisizione utilizzate.

3.4 XGBoost

Si tratta di una libreria open-source che implementa l'algoritmo di *Gradient Boosting* in modo scalabile e molto efficiente. [15] Il *Gradient Boosting* è un tecnica di machine learning che produce un modello predittivo nella forma di un insieme di modelli predittivi deboli, tipicamente alberi di decisione, con lo scopo di minimizzare una *loss function* scelta arbitrariamente. La pratica di

boosting consiste nella creazione di nuovi modelli per compensare gli errori dei modelli precedentemente creati, ed aggiungersi a questi per effettuare una previsione finale. È un modello orientato alla rapidità di esecuzione, adatto quindi ad un dataset con molte features, e molto flessibile in quanto può essere utilizzato sia per effettuare una regressione, sia per ricavare delle previsioni come in questo caso. La libreria *XGBoost* implementa il modello *XGBClassifier* che è stato utilizzato per la risoluzione del problema.
Essendosi rivelato particolarmente efficace e veloce nei test preliminari, si è deciso di applicare anche a XGBoost l'ottimizzazione degli iper-parametri. Inoltre, al contrario della rete neurale, non è stato necessario l'utilizzo di una matrice di pesi.
Lo spazio di ricerca utilizzato è il seguente:

- Learning rate compreso tra 0.01 e 1 (unica variabile continua)
- Max depth compreso tra 1 e 20
- Numero di alberi compreso tra 1 e 100
- Delta compreso tra 1 e 10

Le prestazioni dell'algoritmo hanno permesso di applicare una *5-fold-cross-validation* ad ogni iterazione di AutoML che rende la valutazione del modello molto più robusta. La funzione da ottimizzare, in questo caso, è ancora *NDCG* che però restituisce come esito la media dei risultati dei 5 test set creati dalla cross validation ad ogni iterazione. Anche in questo caso, per l'ottimizzazione, le valutazioni iniziali sono 10 e il budget disponibile è impostato a 70 iterazioni. Il processo è ripetuto per entrambe le funzioni di acquisizione scelte.

4 Risultati e valutazioni

Prima di procedere con l'esposizione dei risultati, si vuole evidenziare che, al fine di poter confrontare diverse tecniche, sono stati scelti dei *seed* che permettessero la replicabilità dei risultati. L'esecuzione di tali algoritmi è avvenuto per mezzo del servizio *Google Colab* che mette a disposizione macchine con 25 GB di RAM e 68 GB di spazio su disco e diverse tipologie di GPU, inoltre la piattaforma, nella verisione gratuita, permette sessioni di lavoro fino a 12 ore consecutive e potenza computazionale limitata.

I risultati finali ottenuti in seguito al processo di AutoML dai classificatori Neural Netowrk ed XGBoost, con le funzioni di acquisizione EI (*Expected Improvemnet*) ed PI (*Probability of Improvemnet)*, sono descritti nella tabella [Tab. 1].

Table 1: Risultati

Modello	**Tempo (s)**	**NDCG Score**
Neural Network (EI)	108.32	0.82689
Neural Network (PI)	112.13	0.82800
XGBoost (EI)	8.82	0.83164
XGBoost (PI)	11.63	0.83189

I tempi riportati non possiedono rilevanza statistica, ma sono indicativi del tempo necessario all'addestramento di ogni modello. Gli iper-parametri ottimali, per ogni modello e funzione di attivazione, ottenuti con AutoML sono riportati nell'Appendice. Infine il modello XGBoost (PI) viene anche utilizzato per generare le previsioni del dataset *test_user.csv* ottenendo un *NDCG* pari a 0.88006, viene anche riportato il risultato vincitore della competition pari a 0.88697. Altri risultati possono essere trovati sulla *leaderboard* relativa alla challange.

5 Discussione dei risultati

Il fine di questo studio consiste nel trovare un algoritmo di classificazione che, combinato a tecniche di manipolazione dei dati, sia in grado di restituire un buon punteggio NDCG pur rimanendo efficiente dal punto di vista computazionale. Analizzando i risultati si nota che il modello migliore sia XGBoost PI, ma si sottolinea anche che la differenza con gli altri, indipendentemente dalla funzione di acquisizione utilizzata, è quasi irrilevante. Osservando i risultati del processo di AutoML e i grafici 2 e 3 a loro associati presenti nell'Appendice, si evince che in entrambi i modelli la funzione d'acquisizione che permette di raggiunge le prestazioni migliori è la *Probability of Improvement*. Ciò sta ad indicare, probabilmente, che in questo task sia stata più utile la ricerca di nuove soluzioni vicine nello spazio di ricerca indicato per gli iper-parametri (*exploitation*), rispetto all'esplorazione di regioni meno note (*exploration*).

Diversamente rispetto all'analisi del punteggio, impostando il confronto sulla base dei tempi di elaborazione, appare evidente come esista una differenza marcata in favore dell'algoritmo XGBoost che riesce a terminare l'intero processo di apprendimento con un tempo inferiore di un ordine di grandezza rispetto alla rete neurale; ciò ha permesso, come già riportato precedentemente, l'utilizzo della strategia *cross-validation* durante il processo di AutoML che rende molto più robusta la valutazione del modello rispetto alla semplice validation utilizzata per la rete neurale.
Effettuata una valutazione delle prestazioni dei modelli prodotti, può essere di particolare interesse l'analisi delle feature più importanti sul modello XGBoost (PI) che, stando ai dati riportati, rappresenta il miglior compromesso di efficienza ed efficacia della previsione tra gli algoritmi analizzati. Lo studio della *feature importance* non è solo fine a se stessa, ma può rilevare se il preprocess effettuato ha portato ad intuizioni utili alla previsione. Per ottenerla si utilizza una funzione compresa nella libreria XGBoost che permette di calcolarla in tre modi diversi: *weigth*, *gain* e *cover*; nel caso in questione è stato utilizzato il primo che misura il numero di volte in cui una feature è utilizzata per splittare i dati in tutti gli alberi.
I 15 attributi più rilevanti vengono rappresentati nel grafico riportato in Figura 4, contenuta nell'Appendice. Il grafico evidenzia come la feature considerata più importante in assoluto sia l'attributo *time_lag* originariamente non presente nel dataset, così come tutte le features cicliche calcolate che risultano avere un notevole peso nella classificazione. Se da un lato si può affermare di aver considerevolmente migliorato la qualità dei dati grazie al preprocessing effettuato e introducendo nuove variabili considerevolmente influenti, dall'altro risulta evidente come la variabile *age*, che risultava inizialmente essere la più affetta da valori fittizi o addirittura nulli, ricopra un ruolo critico nella decisione: ciò rende particolarmente difficile migliorare ulteriormente le previsioni, in quanto queste saranno fortemente influenzate da un dato di dubbia affidabilità.

6 Conclusioni

Nell'elaborato vengono presentate diverse tecniche di machine learning e tecniche di elaborazione dei dati volte ad ottenere la miglior classificazione possibile rispetto alla metrica considerata, ovvero il punteggio NDCG. I risultati ottenuti sono frutto dei classificatori scelti, del processo di *Hyperparamater*

optimization ma anche, in particolar modo, delle nuove features introdotte nella fase di manipolazione dei dati iniziale. Il modello migliore risulta quindi essere XGBoost (PI) che raggiunge ottimi risultati anche nel contesto della *challenge*.
Un possibile miglioramento dei risultati riguarda proprio un ulteriore approfondimento della conoscenza dei dati al fine di creare features addizionali che risultino anch'esse significative per la classificazione. Inoltre, sapendo che la variabile *age* risulta molto importante ed, al tempo stesso, sia soggetta a molteplici valori mancanti o errati, uno sviluppo futuro potrebbe essere un nuovo sistema più efficace per l'immissione dei dati.

References

[1] AirBnB-Newsroom, "Fast facts," news.airbnb.com, 2020-04-14.

[2] Kaggle, "Airbnb new user bookings," kaggle.com, 2020-04-14.

[3] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 3–9, 2007.

[4] C. Distinguishability, "A theoretical analysis of normalized discounted cumulative gain (ndcg) ranking measures," 2013.

[5] R. Pupale, "Support vector machines(svm) — an overview," towardsdatascience.com, 2020-04-17.

[6] P. Mandot, "How exactly xgboost works?" medium.com, 2020-04-17.

[7] T. Yiu, "Understanding neural networks," towardsdatascience.com, 2020-04-17.

[8] Google, "Google colab resources," Colab FAQ.

[9] J. Gangele, "How does automl works?" medium.com, 2020-04-18.

[10] J. Jimenez, "pygpgo: Bayesian optimization for python," pygpgo.readthedocs.io, 2020-04-18.

[11] R. Garnett, "Bayesian methods in machine learning," cse.wustl.edu.

[12] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[13] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[15] T. Chen, "Xgboost implementation," xgboost.readthedocs.io, 2020-04-17.

Appendice

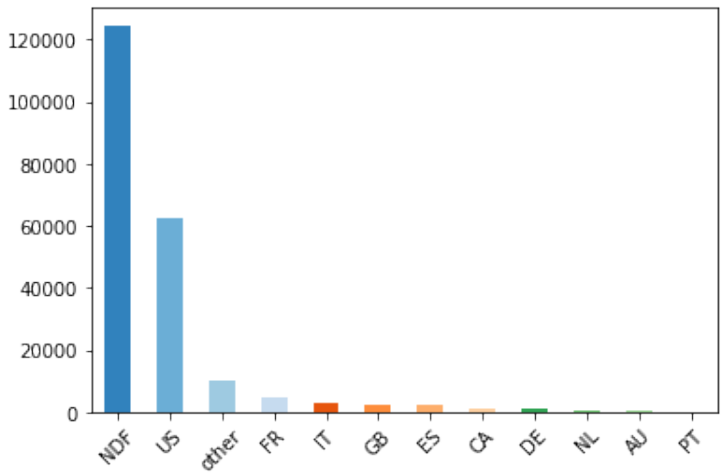

Figure 1: Distribuzione variabile target

Iper-parametri ottimali per **Rete Neurale** con *Expected Improvement*:

- Learning rate = 0.000436
- Numero di hidden layer = 1
- Numero di neuroni input layer = 399
- Numero di neuroni hidden layer = 955
- Batch size = 121

Iper-parametri ottimali per **Rete Neurale** con *Probability of Improvement*:

- Learning rate = 0.000296
- Numero di hidden layer = 2
- Numero di neuroni input layer = 797
- Numero di neuroni hidden layer = 724
- Batch size = 62

Iper-parametri ottimali per **XGBoost** con *Expected Improvement*:

- Learning rate = 0.550212
- Max depth = 4
- Numero di alberi layer = 32
- Delta = 1

Iper-parametri ottimali per **XGBoost** con *Probability of Improvement*:

- Learning rate = 0.286817
- Max depth = 5
- Numero di alberi layer = 39
- Delta = 7

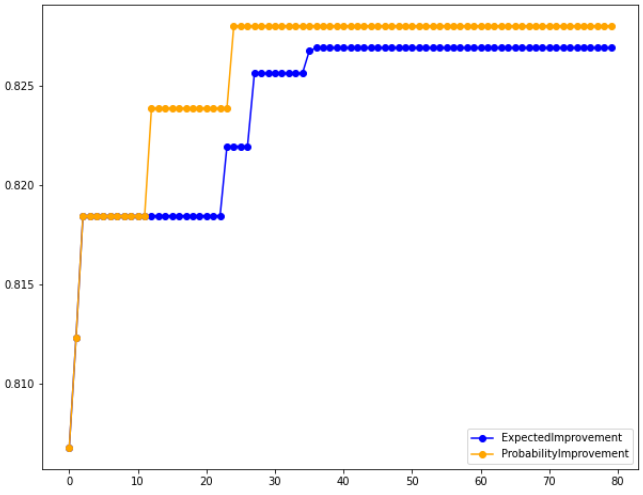

Figure 2: Trend AutoML Neural Network

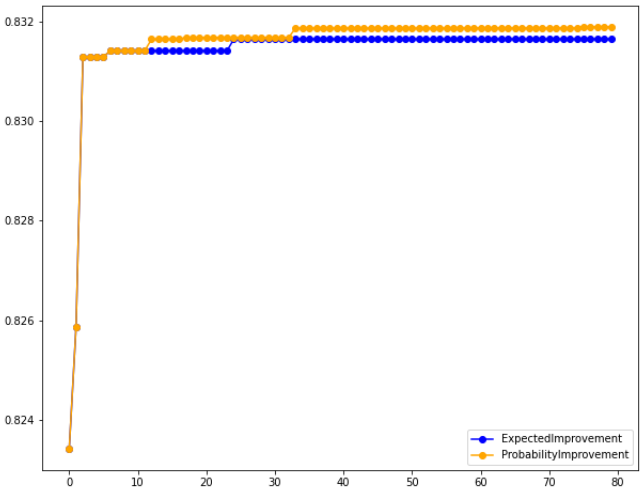

Figure 3: Trend AutoML XGBoost

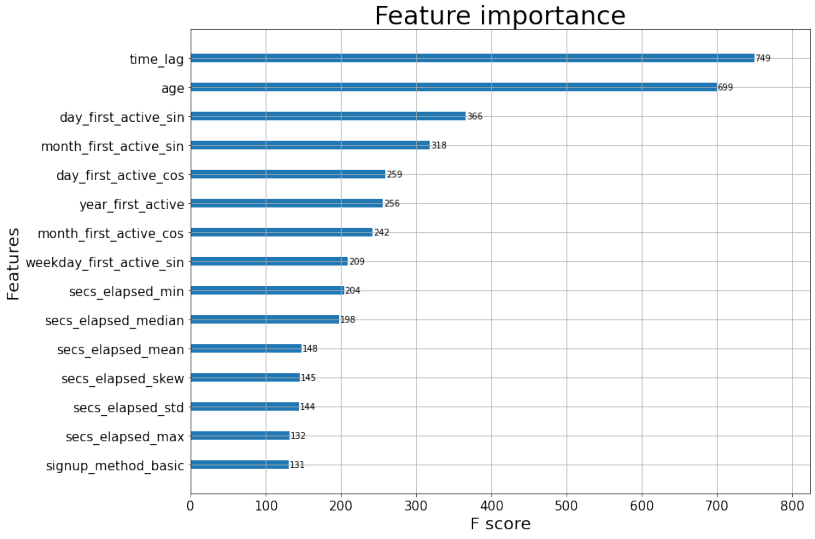

Figure 4: Feature Importance XGBoost