Università degli Studi di Ferrara

Università degli Studi di Ferrara

Corso di Laurea in Informatica

Analisi predittiva del churn nel settore utility: Un approccio basato sul machine learning

Relatore:

Prof. Fabrizio Riguzzi

Laureando:

Alex Frizziero

Anno Accademico 2023 – 2024

Indice

Pagina

1 Introduzione **7**

2 Descrizione del problema **11**

2.1 Introduzione al churn rate 11

2.2 Il problema del churn nel contesto utility 12

2.3 I fattori che contribuiscono al churn 13

2.4 Analisi predittiva del churn 14

2.5 Sfide dell'analisi predittiva 15

2.5.1 Etica e privacy nell'analisi predittiva 16

2.6 L'integrazione di dati esterni 16

3 Dataset e tecniche di analisi **19**

3.1 Descrizione dei dataset 19

3.1.1 Definizione della classe target 23

3.2 Data visualization 24

3.3 Preprocessing dei dati 27

3.3.1 Gestione dei dati mancanti 27

3.3.2 Tecniche di encoding per variabili categoriche 28

3.4 Preparazione avanzata del dataset 30

3.4.1 Feature selection 31

3.4.2 Bilanciamento delle classi 33

4 Metodologie di sviluppo in azienda **35**
4.1 Metodologia Agile . . . 35
4.2 Framework Scrum e concetti chiave . . . 36
4.2.1 Strumenti e tecnologie di supporto . . . 37
4.3 Fasi del processo di sviluppo . . . 40
4.3.1 Pianificazione . . . 40
4.3.2 Sviluppo . . . 40
4.3.3 Testing e rilascio . . . 41
4.4 Strumenti di sviluppo e infrastrutture . . . 42
4.4.1 Linguaggi e ambienti di sviluppo . . . 42
4.4.2 Servizi e infrastrutture . . . 43

5 Sviluppo dei modelli predittivi **45**
5.1 Algoritmi di machine learning utilizzati . . . 45
5.1.1 Logistic Regression . . . 47
5.1.2 Support Vector Machine (SVM) . . . 48
5.1.3 Gradient Boosting: . . . 49
5.1.4 Adaptive Boosting (AdaBoost) . . . 50
5.1.5 Random Forest . . . 51
5.1.6 Artificial Neural Network (ANN) . . . 52
5.2 Metriche di valutazione delle performance . . . 53
5.2.1 Confusion Matrix . . . 53
5.2.2 Accuracy . . . 54
5.2.3 Precision, Recall e F1-score . . . 55
5.2.4 AUC-ROC . . . 56
5.3 Addestramento e valutazione . . . 57
5.4 Modello 1 . . . 59
5.5 Modello 2 . . . 61
5.6 Modello 3 . . . 62

6 Analisi dei risultati **65**
6.1 Valutazione Modello 1 . . . 65
6.1.1 Risultati su campione ridotto . . . 66
6.1.2 Esiti finali . . . 68

6.1.3 Considerazioni - Modello 1 71
6.2 Valutazione Modello 2 72
6.2.1 Risultati su campione ridotto 73
6.2.2 Esiti finali 74
6.2.3 Considerazioni - Modello 2 77
6.3 Valutazione Modello 3 78
6.3.1 Panoramica risultati parziali 78
6.3.2 Risultati su campione ridotto 82
6.3.3 Esiti finali 83
6.3.4 Considerazioni - Modello 3 86
6.4 Considerazioni finali 86

7 Conclusioni **89**

Bibliografia **91**

Capitolo 1

Introduzione

Negli ultimi anni il settore utility ha visto una crescente competitività. In questo scenario, è sempre più comune che un cliente cambi azienda, pertanto la capacità di prevedere il churn è diventata cruciale per mantenere una base di utenti stabile e soddisfatta. Le aziende che operano in questo settore devono adottare strategie proattive per identificare i segnali di abbandono ed intervenire tempestivamente per ridurre il churn. L'analisi predittiva, basata su tecniche di machine learning, rappresenta un approccio potente per affrontare questa sfida, permettendo di individuare pattern nei dati e di prendere decisioni mirate.

L'obiettivo principale di questo lavoro di tesi è illustrare lo sviluppo e la valutazione di modelli di machine learning per la previsione del churn, specifici per il settore utility. Attraverso l'analisi di dati storici dei clienti, l'applicazione di tecniche di data analysis e la costruzione di algoritmi di machine learning, questa ricerca mira a fornire strumenti e metodi efficaci per identificare i clienti a rischio di abbandono e supportare le strategie aziendali di *customer retention*. L'analisi e lo sviluppo di questo progetto sono frutto di un esperienza formativa svolta presso l'Università degli Studi di Ferrara, nella quale ho avuto l'occasione di collaborare con un team di sviluppo dell'azienda Doxee. Durante questa esperienza, ho avuto l'opportunità di applicare concetti, teorie e strategie nel campo dell'analisi dei dati

e del machine learning, settori che mi appassionano profondamente e a cui sono particolarmente legato.

La tesi è articolata in sette capitoli, ognuno dei quali è dedicato ad un aspetto specifico del progetto. Il capitolo due, introduce il problema del churn, in particolare nel contesto utility, analizzando le possibili cause e le implicazioni per le aziende e delineando le sfide principali che si pongono nella previsione e gestione del fenomeno. Viene inoltre discusso il ruolo cruciale delle tecniche predittive nel supportare decisioni aziendali.

Nel terzo capitolo viene illustrata la struttura generale dei dataset utilizzati, con una descrizione dettagliata delle variabili e delle tecniche di analisi preliminari e avanzate applicate per la manipolazione ed il trattamento dei dati. Si esplorano anche le sfide legate alla qualità e alla preparazione dei dati per l'analisi, come la gestione dello sbilanciamento o dei dati mancanti.

Il quarto capitolo esamina lo sviluppo del progetto all'interno del contesto aziendale, descrivendo le metodologie di lavoro adottate e gli strumenti impiegati dal team. Nello specifico, vengono esposte le fasi dello sviluppo, le risorse impiegate e le tecnologie di supporto utilizzate per collaborare e organizzare il lavoro. Inoltre, vengono introdotti gli strumenti e le infrastrutture impiegate per la creazione dei modelli predittivi.

Nel quinto capitolo viene presentata una panoramica degli algoritmi di machine learning selezionati per l'analisi predittiva del churn, con una discussione sui criteri di scelta e i metodi di valutazione delle performance. Viene fornita anche una descrizione delle fasi di sviluppo vere e proprie che hanno portato alla creazione dei modelli finali. Questo capitolo include anche una descrizione dettagliata delle specifiche caratteristiche e le motivazioni che hanno guidato la creazione di ciascun modello, oltre che delle strategie, delle tecniche e degli algoritmi utilizzati per lo sviluppo.

Il sesto capitolo presenta i risultati ottenuti dai modelli addestrati, fornendo una valutazione della loro performance in relazione alle metriche più appropriate. A supporto dei risultati conseguiti, vengono mostrati grafici e tabelle relativi al comportamento di ciascun modello sottoposto a diverse fasi di valutazione. Il capitolo si chiude con una considerazione generale su tutti i modelli messi a confronto.

Nel settimo ed ultimo capitolo vengono sintetizzati nuovamente gli obiettivi ed

il lavoro svolto, evidenziando i punti di forza e le limitazioni dello studio. Vengono inoltre proposte possibili direzioni future per ulteriori ricerche e miglioramenti dei modelli predittivi, nel campo dell'analisi del churn nel settore utility. Si conclude con una valutazione complessiva dell'esperienza alla quale ho potuto prendere parte durante il periodo accademico.

Capitolo 2

Descrizione del problema

2.1 Introduzione al churn rate

Il *churn* è l'interruzione, da parte di un cliente, dell'acquisto di prodotti o servizi di un azienda. Questo fenomeno coinvolge moltissime imprese, in particolar modo, quelle che ne risentono di più appartengono al settore *utility*, che fondano il proprio *business* per l'appunto sull'erogazione di servizi dedicati ai cittadini. Sebbene comunemente il settore *utility* comprenda aziende che forniscono beni essenziali come energia, gas, acqua e gestione dei rifiuti, faremo riferimento a questo ambito con un interpretazione più ampia, comprendendo anche i servizi di telecomunicazioni, vista la loro importanza per la vita quotidiana e nella società moderna. Vedremo più nel dettaglio in seguito le implicazioni che derivano dal *customer churn*, per ora ci basti pensare in maniera intuitiva, che in generale per un azienda è più difficile e costoso acquisire nuovi clienti piuttosto che mantenere quelli già presenti, ma prima di analizzare le cause, le conseguenze e le possibili soluzioni, mostriamo come il problema si definisce matematicamente.

Esiste una formula associata al calcolo del *churn rate*, indicata come la divisione del numero di clienti persi in un determinato periodo di tempo, dal cui risultato si ricava il dato percentuale moltiplicando per cento.

Tasso di abbandono:

$$(\frac{Clienti\ persi}{Clienti\ totali\ all'inizio\ del\ periodo\ di\ tempo}) \times 100$$

2.2 Il problema del churn nel contesto utility

Un *report* di ARERA (Autorità di Regolazione per Energia Reti e Ambiente) risalente a Febbraio 2022 [1], mostra i seguenti dati: il 59,7% delle famiglie italiane ha scelto il mercato libero per la fornitura di energia elettrica. Si tratta di una crescita complessiva del +2,4% rispetto ai sei mesi precedenti. Simili sono i risultati registrati per la fornitura di gas naturale, il 62% delle famiglie ha scelto il mercato libero con un aumento del +1,8% rispetto alle precedenti rilevazioni.

Questo cambiamento non si è limitato ai clienti domestici, ma ha coinvolto anche aziende con cifre analoghe. Dati più recenti, forniti sempre da ARERA, aggiornati a Luglio 2024 [2] indicano come la quota del mercato libero risulti pari al 76,5%. Uno studio di Fred Reicheld di Bain and Company [3], mostra come un aumento del 5% nella fidelizzazione dei clienti, può produrre più del 25% di aumento dei profitti, questo perchè i clienti fidelizzati tendono ad utilizzare più servizi di un azienda nel tempo, i costi per servirli diminuiscono ed inoltre possono indirizzare altri verso quest'ultima.

Indagini di Statista [1], mostrano che anche aziende leader del settore come Vodafone e Telecom Italia (TIM) non siano estranee al fenomeno del churn. In particolare nel 2023 TIM [4] ha avuto un tasso di abbandono del 12,8%, mentre Vodafone [5] in Europa nel Q2 2023/24 ha registrato il valore di churn rate più basso attorno al 15,5% in Turchia e il valore più alto in Spagna pari al 26,6%. Ora osservando questi dati nel contesto sempre più diffuso della liberalizzazione del mercato e dell'aumento della concorrenza che ne deriva, possiamo facilmente accorgerci dell'impatto significativo che il customer churn ha sulla redditività e sulla stabilità aziendale e della necessità di strategie per combatterlo. Purtroppo però la perdita di entrate non è l'unica conseguenza, tra le tante, questo può portare a problematiche come calo della competitività, effetto sulla reputazione, impatto sulla strategia di crescita o compromissione di attività o progetti.

[1] Statista è una piattaforma globale di dati e business intelligence con un'ampia raccolta di statistiche, report e approfondimenti. Fondata in Germania nel 2007

2.3 I fattori che contribuiscono al churn

I motivi che possono portare un cliente ad abbandonare un'azienda possono essere variegati e molteplici, e molto spesso non facili da identificare. E' limitante infatti pensare che tutto si riduca ad una questione di costi per il compratore, fattore sicuramente rilevante, ma non l'unico da considerare. Tra le varie cause ad esempio, oltre i già citati fattori economici, potrebbero esserci elementi relativi alla qualità del servizio, come interruzioni, errori di fatturazione o una scarsa attenzione nell'assistenza, oppure fattori contrattuali o riguardanti il mercato; in alcuni casi essi possono anche non dipendere dall'azienda o dal contesto strettamente legato al servizio, ma essere di natura circostanziale o personale. Qualsiasi sia la natura dei fattori e delle cause che portano come conseguenza ultima l'abbandono del cliente, nella maggioranza dei casi, è certamente vero che il churn è il risultato di una serie di episodi nel tempo.

Come ampiamente descritto da Bain and Company nell'articolo *Breaking the back of customer churn* [6], l'ultimo evento a portare il cliente a lasciare l'azienda non è mai isolato, rappresenta solo la scintilla che determina la rottura del rapporto, in generale dopo un lungo periodo di erosione della fiducia. Questo frangente più o meno ampio che sia, viene descritto in sei fasi:

1. *Pre-churn*: circostanze che aprono la falla, come aspettative errate durante il processo di vendita o un problema latente nell'installazione.

2. *Root causes*: interazioni che creano frustrazione.

3. *Tipping point*: uno o più eventi che predispongono il cliente all'abbandono.

4. *Final trigger*: evento che fa scattare la decisione di abbandonare, come un'offerta di un concorrente o la raccomandazione di un amico.

5. *Churn*

6. *Post-churn*: interazioni con l'azienda che rendono i clienti più o meno propensi a ritornare.

Se a questo aggiungiamo che intervenire tardi per recuperare la fiducia del cliente non è solo più costoso ma anche meno efficace, risulta evidente l'importanza di

adottare giuste strategie per arginare gli effetti e ridurre l'abbandono dei fruitori. Sottolineando quanto già detto, le cause del churn possono essere molte, da questo ne ricaviamo che per quante siano le iniziative, più o meno efficaci, per ridurre questo valore, è utopico pensare di annullarlo, ed è pertanto più intelligente concentrarsi sul sottoinsieme di cause che hanno impatto maggiore.

2.4 Analisi predittiva del churn

Abbiamo visto quali possono essere le cause e le conseguenze che ne derivano, ora ci chiediamo: "e se fosse possibile identificare in anticipo i clienti a rischio di abbandono?" Una soluzione giunge dal campo della *data science*: negli ultimi anni, l'utilizzo di tecniche di *machine learning*[2] ha ottenuto una notevole attenzione, grazie all'efficacia dimostrata dagli studi sull'argomento. E' infatti sufficiente una ricerca anche superficiale, per accorgersi dell'innumerevole varietà di tecniche di machine learning, che negli anni hanno portato a riscontri positivi nel cercare di contenere il problema del customer churn. L'idea è quella di partire dalla natura matematicamente ben definita del problema e costruire dei modelli che possano imparare e fare inferenza dai dati. Esistono diversi contesti di applicazione delle tecniche di machine learning, che dipendono dallo scopo che si vuole raggiungere, il campo più legato al nostro problema è la *knowledge extraction.*[3] Esistono due approcci principali, ma non unici, all'estrazione di conoscenza; ci riferiamo a questi con il nome di paradigmi di learning, e sono:

1. *Supervised learning*: sfrutta il concetto della *ground truth*, ovvero la caratteristica del *dataset* di avere i dati etichettati, in altre parole, nel nostro caso, i valori della classe target *churned* sono conosciuti, essenziali per la validazione del modello.

2. *Unsupervised learning*: caso in cui i dati non siano etichettati, il modello deve quindi lavorare su dati grezzi e cercare dei pattern senza una guida.

Nel caso del supervised learning esistono almeno tre categorie di modelli considerabili: modelli semplici di classificazione, modelli *ensemble*, che fondono più tecniche

[2]Materia che si occupa di creare sistemi che apprendono o migliorano le *performance*, dato un insieme di dati dal quale fare analisi e previsioni

[3]Pratica di estrarre conoscenza a partire da un insieme di dati

o algoritmi per ottenere dei risultati migliori, e reti neurali. Lo sviluppo di modelli basati su reti neurali tuttavia, presenta spesso delle complicazioni non indifferenti. I principali fattori che possono dissuadere da questo tipo di implementazione riguardano ad esempio, la difficoltà nell'interpretare i processi decisionali interni al modello, per questo vengono spesso considerati come delle *black box*, oppure, i notevoli costi in termini di costi computazionali, o ancora, la difficile ottimizzazione dei parametri e della struttura della rete. Questi non sono che alcuni dei problemi che coinvolgono questo tipo di modelli, in seguito approfondiremo meglio caratteristiche e differenze delle categorie citate in precedenza.

2.5 Sfide dell'analisi predittiva

Il termine *big data* non sarà di certo nuovo ai lettori, riguarda set di dati grandi e complessi e difficili da analizzare con i metodi tradizionali. Fondano la propria struttura sul concetto delle tre "V", ovvero, volume, che si riferisce alla quantità enorme di dati generata, varietà, riguarda i diversi tipi di dati generati, e velocità, cioè la rapidità con la quale i dati sono prodotti. L'analisi di questa categoria di dati è ormai essenziale in qualsiasi operazione di marketing e non solo, e si lega perfettamente al concetto di machine learning citato in precedenza. Risulta evidente come le due definizioni si integrino armoniosamente, come pezzi di un puzzle, soprattutto nel contesto dell'analisi predittiva. Tuttavia, tale integrazione può comportare diverse sfide e complicazioni.

Qualsiasi sia l'algoritmo scelto per la costruzione del modello, molti possono essere i fattori che ne influenzano le prestazioni, sebbene, nella maggioranza dei casi riconducibili ai dati. Alcuni di questi problemi possono essere rappresentati, dalla quantità di dati disponibili, gli eccessi in entrambi gli estremi richiedono un attenta analisi prima del loro impiego, o ancora, la scarsa qualità dei dati, informazioni incomplete o inaccurate possono portare a previsioni errate. Altri possono essere lo sbilanciamento delle classi, specialmente nel caso della churn prediction: è infatti molto comune che la parte di clienti che hanno abbandonato, sia molto inferiore al suo complemento; oppure, la caratteristica dei dati di poter variare nel tempo, portando il modello a non adattarsi a questi cambiamenti. Se è vero che i big data sono uno strumento potentissimo è altrettanto vero che se non trattati con le tecniche adeguate possono portare ad effetti controproducenti nel

loro impiego. Questi metodi verranno affrontati in maniera dettagliata nei capitoli successivi.

2.5.1 Etica e privacy nell'analisi predittiva

Una parentesi, tutt'altro che secondaria, legata all'utilizzo dei big data nell'analisi predittiva, è rappresentata da questioni etiche e di privacy. Uno dei principali problemi riguarda la protezione dei dati personali. Non è raro infatti sentire parlare di rischi relativi alla violazione della privacy, in parte come conseguenza dell'enorme quantità di informazioni raccolte, rendendo l'implementazione di adeguate misure di sicurezza, da parte delle aziende, necessaria. Una questione più sottile, ma altrettanto rilevante, riguarda invece la discriminazione. L'analisi predittiva può involontariamente portare a risultati discriminatori, a causa di *bias*[4] contenuti nei dati, che possono manifestarsi con effetti che dipendono dal contesto. Si pensi ad esempio al campo della churn prediction: il modello potrebbe identificare erroneamente certi gruppi demografici come più propensi all'abbandono, o suggerire soluzioni basate su caratteristiche discriminatorie. Più gravi invece possono essere le conseguenze in contesti come l'accesso al credito o le assunzioni. Non è obbiettivo di questo lavoro di tesi affrontare queste problematiche, data anche la natura particolarmente delicata delle stesse, ma è importante quantomeno avere consapevolezza della questione e dell'importanza di mitigare questi fattori.

2.6 L'integrazione di dati esterni

Abbiamo visto l'importanza dei dati, e come questi possono impattare le prestazioni di un modello. La categoria dei dati esterni, riguarda informazioni non direttamente legate al rapporto tra cliente e azienda, ma che possono influenzare il comportamento del cliente. Esempi di questi possono essere fattori meteorologici, che potrebbero condizionare il consumo di energia, o indicatori socio-economici, impattando la capacità di pagamento dei clienti. Risulta evidente come l'integrazione di questi dati potrebbe migliorare la precisione dei modelli predittivi, tuttavia, è altrettanto chiaro come avere accesso a questi possa essere complesso. Esistono aree di ricerca che si occupano di trovare e analizzare soluzioni per

[4] Termine del linguaggio scientifico che indica tendenza, inclinazione, distorsione.

poter raccogliere e valutare informazioni esterne rilevanti, permettendo di portare una comprensione più olistica e contestualizzata del fenomeno del churn e potenzialmente dando vita a nuove tecniche per trattarlo.

Capitolo 3

Dataset e tecniche di analisi

Prima di entrare nel vivo dell'argomento è necessario fare alcune premesse. In questo capitolo, non verranno trattate le fasi di estrazione e aggregazione dei dati, come anche parte delle elaborazioni iniziali, fasi che hanno visto coinvolti altri membri del *team* di sviluppo. Inoltre nel periodo di lavoro sono stati utilizzati più *dataset*, che differiscono principalmente per arco temporale, o per l'aggiunta o la modifica di alcune delle *feature*, dei quali vedremo comunque una descrizione approfondita in seguito. I dataset in questione comprendono dati di diversa provenienza. Una porzione dei dati è di proprietà dell'azienda cliente, mentre la restante parte appartiene all'azienda consulente (Doxee), la stessa con la quale ho collaborato allo sviluppo durante il periodo di tirocinio.

3.1 Descrizione dei dataset

Al fine di facilitare la comprensione, categorizziamo i dataset utilizzati in due distinte tipologie:

1. *Training set:* dataset utilizzati durante la fase di addestramento dei modelli.

2. *Test set:* dataset impiegati per la valutazione delle prestazioni dei modelli.

Un'analisi dettagliata delle differenze specifiche di ciascun dataset verrà presentata nel capitolo dedicato allo sviluppo dei modelli. Per ora è sufficiente evidenziare che la principale distinzione tra le due categorie risiede nella selezione dell'intervallo temporale: il training set comprende i dati relativi ai clienti nel quadrimestre luglio-ottobre 2023, mentre il test set è costituito dai dati, pertinenti al mese di novembre dello stesso anno. Passando invece a quelle che sono le caratteristiche comuni ai due gruppi, troviamo sicuramente una simile dimensionalità e una simile struttura, in termini di *record* e feature. Il Codice 1 mostra la struttura del dataset elencandone le classi e il tipo dei dati che contengono.

Codice 1: struttura del dataset.

```
[6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3225609 entries, 0 to 3225608
Data columns (total 21 columns):
 #   Column                          Dtype
---  ------                          -----
 0   user_id                         object
 1   mobile_fee_changed_this_month   bool
 2   wifi_fee_changed_this_month     bool
 3   pending                         bool
 4   download_ok_norm                float64
 5   payment_status_norm             float64
 6   mobile_fee_average_norm         float64
 7   number_of_mobile_offers_norm    float64
 8   wifi_fee_norm                   float64
 9   wifi_offer                      object
 10  gender                          object
 11  cap                             int64
 12  nationality                     object
 13  age_norm                        float64
 14  vat                             bool
 15  payment_method                  object
 16  emails_delivered_norm           float64
 17  emails_opened_norm              float64
 18  cta_rate_norm                   float64
```

```
 19  partition                       object
 20  churned                         object
dtypes: bool(4), float64(9), int64(1), object(7)
memory usage: 430.7+ MB
```

I dataset in esame presentano una varietà di attributi, ciascuno dei quali rappresenta un aspetto specifico del comportamento del cliente o delle caratteristiche del servizio offerto. Questi *input*, permettono al modello di identificare i *pattern* e le tendenze che possono preannunciare il churn. Data la natura variegata dei dati, tra le manipolazioni iniziali è stato necessario implementare un processo di normalizzazione, questo spiega il nome delle singole colonne all'interno del dataset. Per normalizzazione si intende la trasformazione di una variabile per renderla confrontabile con altre. Questo processo, svolto da un collaboratore, è avvenuto impiegando due tecniche:

1. *feature scaling:* consiste nel portare i valori numerici all'interno di un intervallo compreso tra zero e uno.

2. *Standardizzazione:* procedimento grazie al quale è possibile ottenere una nuova distribuzione i cui valori di media e varianza siano uguali a zero.

Nella tabella 3.1 sono elencate e descritte le feature comuni a tutti i dataset.

Colonna	Descrizione
user_id	Codice del cliente o del contratto associato, usato per identificare anonimamente la riga
mobile_fee_changed_this_month	Indica se ci sono state variazioni nell'offerta mobile nel mese corrente
wifi_fee_changed_this_month	Indica se ci sono state variazioni nell'offerta wifi nel mese corrente
pending	Indica se ci sono pagamenti arretrati associati al contratto
download_ok_norm	Numero di download avvenuti con successo

Continua nella pagina successiva

Colonna	Descrizione
payment_status_norm	Indica se ci sono pagamenti arretrati e il relativo importo
mobile_fee_average_norm	Costo mensile per l'offerta mobile, valore medio con più abbonamenti
number_of_mobile_offers_norm	Numero di offerte mobili attive
wifi_fee_norm	Abbonamento mensile per l'offerta wifi, se presente
wifi_offer	Nome dell'offerta wifi associata al contratto, se presente
gender	Sesso dell'intestatario del contratto
cap	Codice postale del paese di residenza del cliente
nationality	Nazionalità dell'intestatario del contratto
age_norm	Età anagrafica dell'intestatario del contratto
vat	Specifica se il contratto è stipulato con partiva IVA o meno
payment_method	Specifica il metodo di pagamento associato all'offerta
emails_delivered_norm	Numero di email inviate, utile per valutare l'interazione del cliente
emails_opened_norm	Numero di email aperte, utile per valutare l'interazione del cliente
cta_rate_norm	Volte in cui una mail è stata aperta, utile per valutare l'interazione del cliente
churned	Indica se il cliente ha abbandonato oppure no

Tabella 3.1: Descrizione delle singole colonne del dataset

Analizzando più nel dettaglio la struttura del dataset, ci accorgeremo che i dati, sono caratterizzati da una certa sparsità, presentano cioè dei valori mancanti. In particolare nelle colonne come *wifi_fee_norm* e *wifi_offer* la mancanza di informazioni è determinata dal fatto che non tutti i clienti hanno un'offerta wifi e di conseguenza non pagano l'abbonamento corrispondente. Nel caso invece di classi

come *gender*, *nationality* ed *age_ norm*, la caraenza è data dal fatto che i suddetti clienti sono in possesso di partita iva, che nel nostro dataset è rappresentata dalla colonna *vat*. La mancanza di dati nella colonna *payment_ method*, invece, potrebbe essere riconducibile ad errori in fase di raccolta. La figura 3.1 mostra tramite una rappresentazione visiva la distribuzione dei dati mancanti suddivisi per classe. Vedremo in seguito come e con quali tecniche il problema dei dati mancanti è stato affrontato.

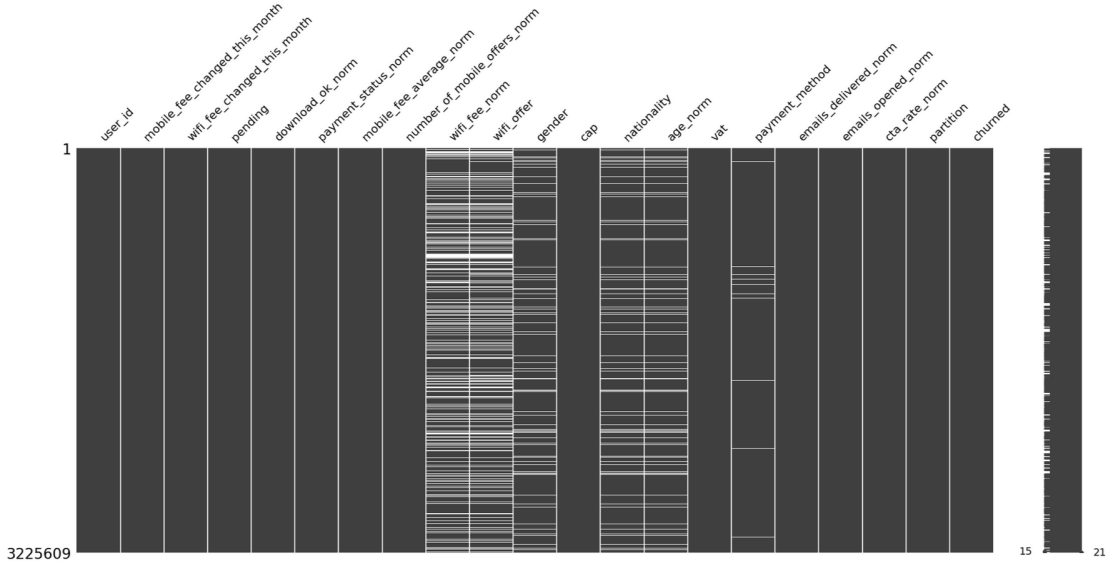

Figura 3.1: distribuzione dei dati mancanti

3.1.1 Definizione della classe target

Nel contesto del nostro studio, la variabile dipendente, oggetto della nostra analisi, è rappresentata dalla classe churn. Questa variabile costituisce l'output del modello che intendiamo sviluppare. Il nostro approccio si inquadra nel paradigma dell'apprendimento supervisionato (supervised learning), descritto in precedenza, in quanto il dataset a nostra disposizione è etichettato. La variabile *target* è rappresentata come una classe binaria, codificata nel seguente modo: il valore Y indica che il cliente ha abbandonato, N altrimenti. La natura binaria della classe target ci consente di affrontare il problema come un *task* di classificazione, dove l'obbiettivo è predire l'appartenenza di ciascun cliente ad una delle due categorie, sulla base delle feature presenti nel dataset. Come ben rappresentato dalla figura 3.2, il

dataset è sbilanciato. In particolare vediamo come ci sia una netta predominanza della classe negativa N, o non churned, rispetto alla classe positiva Y, o churned. In contesti come il nostro, è molto comune avere a che fare con dataset molto sbilanciati, proprio per la natura del problema. Una delle sfide maggiori nello sviluppare un buon modello deriva dallo sbilanciamento; un ruolo decisivo è infatti affidato alla scelta delle tecniche e delle metriche di valutazione più adeguate. Vedremo nel capitolo dedicato come la questione dello sbilanciamento è stata affrontata.

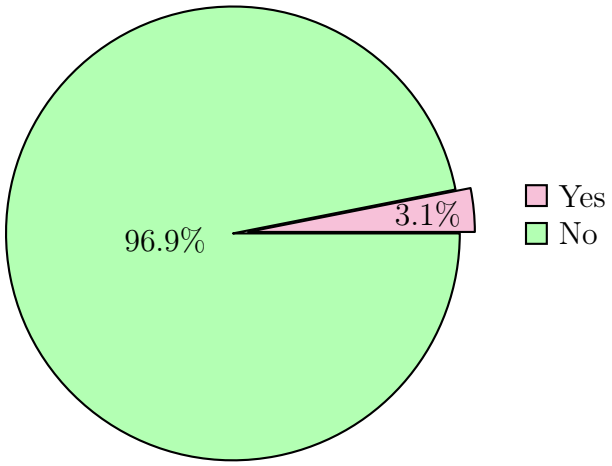

Figura 3.2: Aerogramma che mostra lo sbilanciamento della classe target

3.2 Data visualization

Nell'ambito dell'analisi dei dati, la fase di data visualization rappresenta un passaggio importante per comprendere meglio la distribuzione dei dati attraverso rappresentazioni grafiche intuitive. L'aerogramma in figura 3.3 mostra la distribuzione di valori della feature *payment_method*, dalla quale si può notare come la maggioranza della classe appartenga al tipo Addebito su cc bancario o postale.

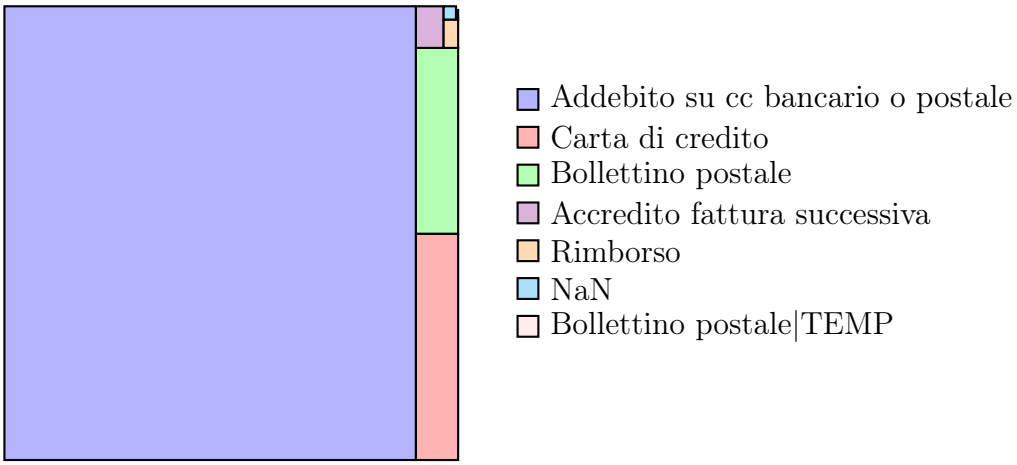

Figura 3.3: Aerogramma che mostra lo sbilanciamento della classe target

Il grafico in figura 3.4 mostra la distribuzione dei pagamenti arretrati, identificati dalla feature *pending*, suddivisi per clienti churn e non churn.

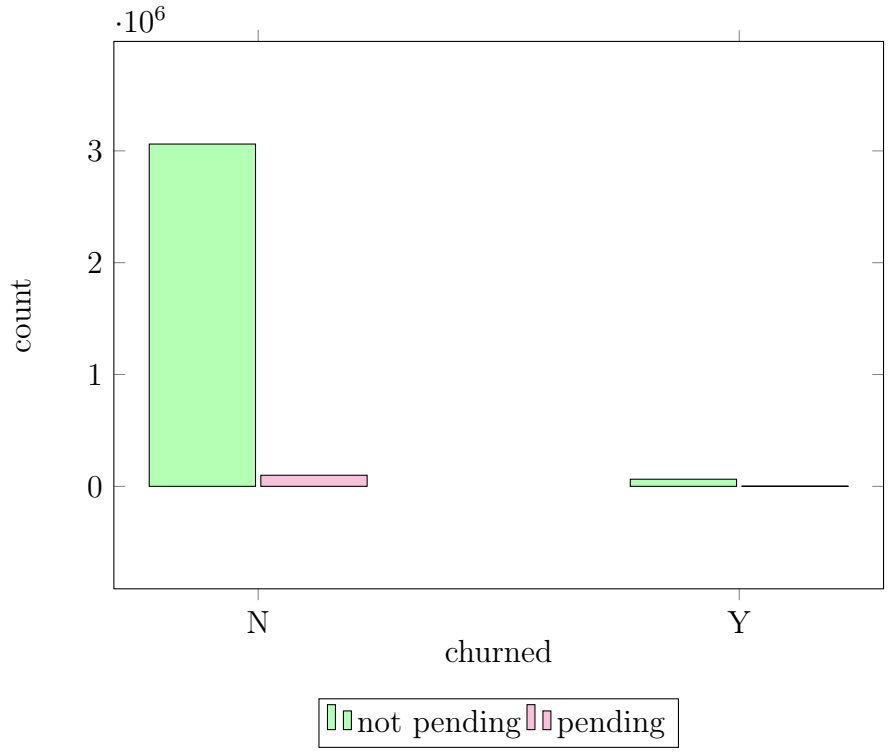

Figura 3.4: Istogramma che mostra la distribuzione della classe pending

Il forte sbilanciamento nella distribuzione dei dati suggerisce che la maggioranza della clientela non presenta insoluti. Nei casi in cui si riscontrano arretrati, questi sono prevalentemente associati ai clienti che mantengono il rapporto con l'azienda. Tale correlazione potrebbe essere giustificata dal fatto che la presenza di pagamenti pendenti potrebbe precludere la possibilità di cessare il rapporto con l'azienda.

Il grafico in figura 3.5 mostra la distribuzione dei clienti che hanno stipulato un contratto con o senza partita IVA, rappresentati dalla colonna *vat*, suddivisi per clienti churn e non churn.

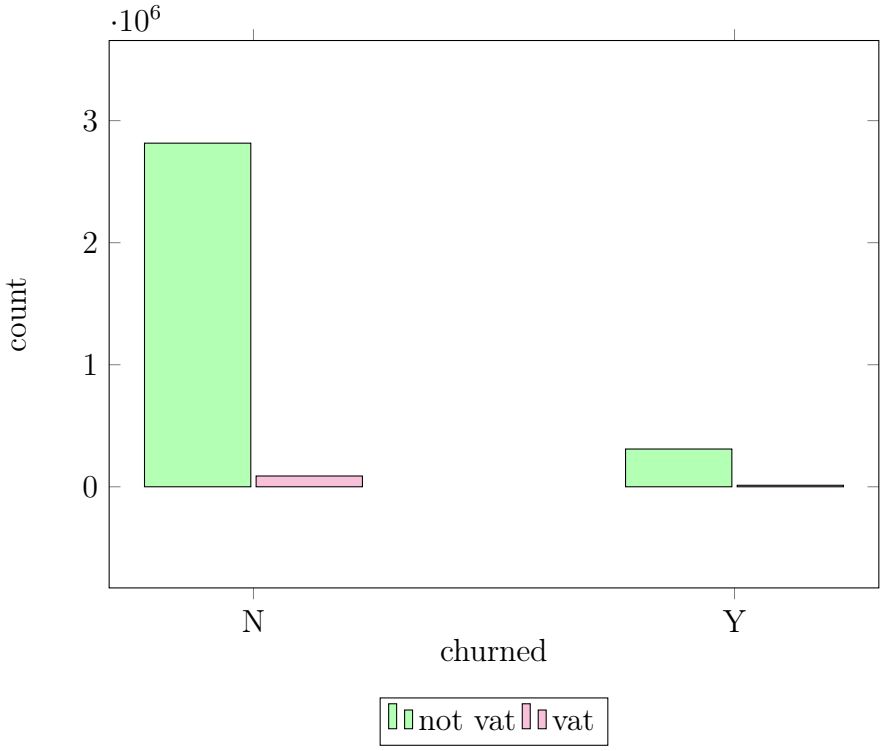

Figura 3.5: Istogramma che mostra la distribuzione della classe vat

L'analisi dell'istogramma rivela alcune informazioni significative: i clienti titolari di partita IVA rappresentano una minoranza sia tra coloro che hanno cessato il rapporto con l'azienda sia tra quelli che lo hanno mantenuto. La maggior parte dei clienti che hanno interrotto il servizio appartiene alla categoria di coloro che sono privi di partita IVA. Tuttavia, in termini proporzionali, i clienti con partita IVA sembrano manifestare una maggiore propensione all'abbandono, rispetto al più ampio segmento di clienti senza partita IVA.

3.3 Preprocessing dei dati

La fase di *preprocessing* dei dati rappresenta una parte cruciale nel processo di analisi, in quanto prepara il terreno per l'applicazione efficace degli algoritmi di machine learning. In questa sezione descriveremo le tecniche adottate per raffinare e ottimizzare il nostro dataset, per prepararlo alle fasi di analisi successive.

Un punto chiave nel preprocessing è stato la creazione di un *subset* del dataset originale. Data la considerevole dimensione dell'insieme di partenza, si è ritenuto opportuno procedere all'estrazione di un campione rappresentativo. Questa decisione è stata motivata dai seguenti fattori:

1. *Costi computazionali:* la dimensione ridotta del subset ha permesso di condurre analisi più rapide ed efficienti, riducendo i costi computazionali sull'*hardware* disponibile.

2. *Rappresentatività:* il subset è stato creato mantenendo le caratteristiche chiave del dataset originale, garantendo la validità delle analisi.

3. *Flessibilità:* poter scegliere arbitrariamente la dimensione dell'insieme su cui lavorare offre notevoli vantaggi in termini di elasticità.

Con l'avanzare delle fasi di sviluppo si ha avuto la possibilità di sfruttare hardware più performanti, le dimensioni del subset pertanto, sono variate in maniera conforme alla capacità dei calcolatori. Il processo di creazione del subset ha seguito una metodologia di campionamento stratificato, assicurando che la distribuzione della variabile target rimanesse la stessa di quella del dataset originale.

3.3.1 Gestione dei dati mancanti

Come abbiamo già visto i nostri dataset sono caratterizzati dalla presenza di dati mancanti al loro interno. In generale esistono varie tecniche per affrontare il problema, se i dati mancanti sono molto maggiori di quelli presenti si preferisce ignorare del tutto la feature in quanto manipolarli potrebbe alterarne il significato. Nel caso in cui vi siano dei dati mancanti ma non in maniera dominante, solitamente gli approcci sono due: se la feature è numerica continua allora si utilizza il valore medio di quella feature nelle istanze in cui è nota; se invece la variabile è

discreta allora utilizziamo la moda, cioè il valore che appare più frequentemente in quella feature nelle istanze in cui è nota. Nella tabella 3.2 si vedono nel dettaglio le manipolazioni effettuate e le tecniche di imputazione utilizzate.

Colonna	Manipolazione
nationality	Data la natura discreta della classe, i valori mancanti sono riempiti con il valore moda ITA
payment_method	Valori mancanti riempiti con il nuovo valore *unknown*
wifi_offer	Valori mancanti riempiti con il nuovo valore *none*
gender	Data la natura discreta della classe, i valori mancanti sono riempiti con il valore moda 0 (Maschio)
wifi_fee_norm	Data la natura continua della classe, i valori mancanti sono riempiti con il valore media
age_norm	Data la natura continua della classe, i valori mancanti sono riempiti con il valore media

Tabella 3.2: Tecniche impiegate per la gestione dei valori NaN

3.3.2 Tecniche di encoding per variabili categoriche

Proseguendo nel contesto della nostra analisi, la trasformazione delle variabili categoriche in formato numerico rappresenta una fase cruciale del preprocessing dei dati. Questo processo è essenziale per rendere le informazioni discrete interpretabili dagli algoritmi di machine learning, preservando allo stesso tempo le informazioni e il significato dei dati stessi. I nostri dataset contengono diverse variabili discrete, come ad esempio il metodo di pagamento o la nazionalità del cliente. Le tecniche di codifica impiegate sono *one-hot encoding* e *label encoding*, il cui utilizzo dipende dalla natura della variabile e dalla sua cardinalità. La tecnica one-hot encoding trasforma ciascun valore discreto di una variabile categorica, in una variabile binaria che assume valore uno se il valore è presente, e zero altrimenti. Label encoding invece associa ad ogni valore discreto un corrispondente valore numerico, trasformando quindi la variabile categorica e non creandone di nuove. La prima tecnica è solitamente applicata a variabili nominali con poche categorie, mentre la seconda

è spesso utilizzata per variabili ordinali, che quindi hanno un ordinamento intrinseco. Nella tabella 3.3 vengono descritte le trasformazioni associate a ciascuna classe.

Colonna	Manipolazione
gender	I valori categorici F e M sono stati rimpiazzati rispettivamente con i valori 1 e 0
churned	I valori categorici Y e N sono stati rimpiazzati rispettivamente con i valori 1 e 0
churned	Data la ridotta dimensionalità della classe, la codifica applicata è one-hot
payment_method	Data la ridotta dimensionalità della classe, la codifica applicata è one-hot
wifi_offer	Data la grande varietà di valori della classe, la codifica applicata è label encoding

Tabella 3.3: Tecniche di codifica applicate alle feature

Sebbene label encoding per sua natura non si addica ad essere applicato a classi come metodo di pagamento, la scelta è stata guidata dal fatto che questa codifica riduce notevolmente la dimensionalità, ed inoltre potrebbe esistere un ordinamento implicito della classe, basata sul rischio di mancato pagamento del cliente, identificando una correlazione con la variabile pending.

Nell'ultima fase del preprocessing sono state rimosse dal dataset le colonne non utili all'analisi. Le scelte sono state dettate da uno scarso livello informativo come nel caso di *user_id* o *partition*, oppure da un forte sbilanciamento interno come *cta_rate_norm* o *payment_status_norm*. Tutte le manipolazioni effettuate nelle fasi di preprocessing sono state svolte rispettando i requisiti e i vincoli interni imposti dall'azienda Doxee [7]. Come mostrato da Codice 2, al termine del ciclo di modellazione il dataset risultante ha la seguente forma.

Codice 2: struttura del dataset dopo preprocessing.

[28]:
```
subset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 49999 entries, 2873644 to 1289497
Data columns (total 25 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   mobile_fee_changed_this_month                   49999 non-null  bool
 1   wifi_fee_changed_this_month                     49999 non-null  bool
 2   pending                                         49999 non-null  bool
 3   download_ok_norm                                49999 non-null  float64
 4   mobile_fee_average_norm                         49999 non-null  float64
 5   number_of_mobile_offers_norm                    49999 non-null  float64
 6   wifi_fee_norm                                   49999 non-null  float64
 7   gender                                          49999 non-null  float64
 8   age_norm                                        49999 non-null  float64
 9   vat                                             49999 non-null  bool
 10  emails_delivered_norm                           49999 non-null  float64
 11  emails_opened_norm                              49999 non-null  float64
 12  churned                                         49999 non-null  int64
 13  nationality_ITA                                 49999 non-null  bool
 14  nationality_NUE                                 49999 non-null  bool
 15  nationality_UE                                  49999 non-null  bool
 16  payment_method_Accredito Fattura Successiva     49999 non-null  bool
 17  payment_method_Addebito su c/c bancario o postale  49999 non-null  bool
 18  payment_method_Bollettino Postale               49999 non-null  bool
 19  payment_method_Bollettino Postale|TEMP          49999 non-null  bool
 20  payment_method_Carta di Credito                 49999 non-null  bool
 21  payment_method_Non significativo                49999 non-null  bool
 22  payment_method_Rimborso                         49999 non-null  bool
 23  payment_method_unknown                          49999 non-null  bool
 24  wifi_offer_encoded                              49999 non-null  int64
dtypes: bool(15), float64(8), int64(2)
memory usage: 4.9 MB
```

3.4 Preparazione avanzata del dataset

Dopo la fase iniziale di preprocessing, è seguita un ulteriore fase di manipolazione, basata sull'implementazione di tecniche più sofisticate allo scopo di migliorare ulteriormente la qualità dei dati. In particolare sono state impiegate tecniche

di *upsampling* per gestire il problema dello sbilanciamento della classe target, e metodologie di *feature selection* per selezionare il sottoinsieme di caratteristiche più rilevanti dall'insieme di variabili disponibili. Analizziamo nel dettaglio le tecniche e gli algoritmi utilizzati nelle rispettive sottosezioni.

3.4.1 Feature selection

Nel vasto panorama dei dati disponibili, non tutte le informazioni hanno lo stesso valore predittivo. La selezione delle caratteristiche, è un processo che mira ad identificare un sottoinsieme ottimale di variabili che rappresentano al meglio il problema. Ridurre la dimensionalità dei dati può portare diversi vantaggi, tra cui una minore complessità del modello, oltre che ad un miglioramento delle prestazioni nelle previsioni. Gli svantaggi nell'impiego di queste tecniche dipendono dall'algoritmo scelto, ma nella maggior parte dei casi tendono ad essere computazionalmente costose e la selezione delle feature può essere influenzata dal rumore presente nel dataset.

La tecnica scelta nel nostro approccio è stata la *Recursive Feature Elimination*. RFE appartiene alla categoria dei metodi *wrapper*, poichè utilizza un algoritmo di apprendimento per valutare l'importanza delle caratteristiche. Il funzionamento alla base dell'algoritmo è il seguente: viene addestrato un modello sull'intero dataset e vengono classificate le feature con dei punteggi di importanza; al termine di ogni addestramento viene eliminata la classe determinata come meno importante. Il processo viene ripetuto in maniera ricorsiva, fino al raggiungimento di un criterio di stop. Nella nostra implementazione la valutazione delle prestazioni viene effettuata con *cross-validation*[1], parliamo infatti di RFECV. I parametri accettati dall'algoritmo sono:

1. *Estimator:* definisce l'algoritmo di apprendimento da utilizzare per il processo di selezione.

2. *CV:* definisce la strategia di cross validation e il numero di folds da considerare.

[1]Tecnica per valutare un modello dividendo i dati disponibili in più sottoinsiemi, e utilizzando ciascun sottoinsieme a turno per testare il modello, mentre i restanti sottoinsiemi vengono utilizzati per addestrarlo

3. *Scoring:* metrica da utilizzare per la valutazione della cross validation
4. *Min_feature_to_select:* numero minimo di caratteristiche da selezionare.
5. *Step:* numero o proporzione di caratteristiche da eliminare ad ogni iterazione.

I valori assegnati ai parametri nel Codice 3 sono solo indicativi, li vedremo meglio nel capitolo dedicato gli input scelti.

Codice 3: Implementazione di RFECV sui dati di training.

```
[ ]: cv = StratifiedKFold(5)
     rfecv = RFECV(RandomForestClassifier(), cv=cv, scoring='roc_auc', n_jobs=-1)
     rfecv.fit(X, y)
```

```
[ ]: RFECV(cv=StratifiedKFold(n_splits=5, random_state=None, shuffle=False),
           estimator=RandomForestClassifier(), n_jobs=-1, scoring='roc_auc')
```

```
[ ]: selected_features = X.columns[rfecv.support_]
```

```
[ ]: print("Numero di features selezionate:", len(selected_features))
     print("Features selezionate:", selected_features)
```

```
Numero di features selezionate: 16
Features selezionate: Index(['pending', 'download_ok_norm',
'mobile_fee_average_norm',
       'number_of_mobile_offers_norm', 'wifi_fee_norm', 'gender', 'age_norm',
       'emails_delivered_norm', 'emails_opened_norm', 'nationality_ITA',
       'payment_method_Addebito su c/c bancario o postale',
       'payment_method_Bollettino Postale', 'payment_method_Carta di Credito',
       'payment_method_Rimborso', 'payment_method_unknown',
       'wifi_offer_encoded'],
      dtype='object')
```

Tra le varie tecniche testate va menzionata la *Principal Component Analysis*. Sebbene PCA non sia propriamente una tecnica di feature selection ma piuttosto di *feature extraction*, il cui risultato è un insieme di nuove caratteristiche, anche questa tecnica porta ad una riduzione della dimensionalità e può essere utile in un processo di selezione.

3.4.2 Bilanciamento delle classi

Ignorare lo sbilanciamento può portare a modelli con scarse capacità di generalizzazione, forte bias e potenziali conseguenze negative. Nei casi più comuni, il modello potrebbe diventare troppo "abituato" a predire la classe di maggioranza, producendo previsioni che riflettono solo le caratteristiche di quella classe, ignorando quella svantaggiata. Metriche come l'accuratezza non sono ritenute affidabili sui dati di questo tipo, poichè le prestazioni del modello dipendono unicamente dalla classe maggioritaria. Il sovracampionamento, come anche il sottocampionamento, è una tecnica molto utilizzata quando si ha a che fare con insiemi di dati sbilanciati. L'obbiettivo dell'upsampling è quello di aumentare il numero di campioni della classe minoritaria, per bilanciare meglio il dataset e migliorare le prestazioni dei modelli. Gli algoritmi per realizzare l'upsampling sono molteplici, dai più semplici come *random oversampling* che duplica casualmente campioni già presenti nella classe minoritaria, ai più complessi come *Generative Adversarial Networks* in cui vengono generati dei campioni sintetici e una rete discriminativa cerca di distinguere tra quelli reali e quelli artificiali. In base all'algoritmo scelto vi possono essere diversi svantaggi, generalmente tecniche più semplici possono portare ad *overfitting*[2], mentre quelle più complesse possono richiedere una maggiore capacità computazionale. Inoltre se i dataset di partenza contengono rumore, tecniche di sovracampionamento potrebbero aumentarlo ulteriormente, peggiorando notevolmente la qualità dei dati.

Per trovare una soluzione al problema dello sbilanciamento nel nostro caso, sono state testate più tecniche, sia di sovracampionamento che di sottocampionamento, quella che ha daro i risultati più positivi è stata *Synthetic Minority Over-sampling Technique*. L'idea soggiacente alla tecnica è quella di identificare dei campioni della classe minoritaria, e per ognuno di questi selezionare i suoi k elementi più vicini sulla base della distanza euclidea; successivamente vengono generati i nuovi campioni sintetici come combinazione lineare di quelli selezionati. I parametri fondamentali dell'algoritmo SMOTE sono due:

1. *Sampling_strategy:* Rappresenta il rapporto tra il numero di campioni della classe minoritaria e quella maggioritaria dopo il ricampionamento.

[2]Situazione indesiderata che si verifica quando un modello di machine learning fornisce previsioni accurate sui dati di training, ma non su nuovi dati.

2. *K_ neighbors:* Specifica il numero di vicini da considerare nel campionamento.

Generando campioni sintetici, SMOTE aiuta il modello a generalizzare meglio, portando ad una riduzione del rischio di overfitting, mentre il suo svantaggio più grande è il costo computazionale.

Codice 4: Implementazione di SMOTE sui dati di training.

```
[ ]: smote = SMOTE(random_state=42, sampling_strategy=0.1, k_neighbors=10)
     X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

[ ]: X_train_resampled.shape

[ ]: (39964, 16)
```

Il Codice 4 mostra una delle applicazioni dell'algoritmo, la scelta dei valori per i parametri verrà discussa dettagliatamente nel capitolo dedicato all'addestramento dei modelli.

Capitolo 4

Metodologie di sviluppo in azienda

Prima di discutere approfonditamente della creazione e dell'analisi dei modelli, affrontiamo il tema delle metodologie adottate dal team di sviluppo di cui ho fatto parte. Nello specifico verrà data una panoramica sugli approcci *Agile* e *Scrum* come metodologia di riferimento, sulle fasi dello sviluppo e i software utilizzati.

4.1 Metodologia Agile

La metodologia Agile rappresenta un tipo di approccio allo sviluppo e alla gestione dei progetti i cui principali obbiettivi sono offrire una maggiore flessibilità, migliorare la comunicazione tra i membri del team e dare una maggiore trasparenza sul progetto, rispetto ad altri metodi di sviluppo. Tipicamente nei progetti in ambito data science, dove i requisiti tendono ad evolversi velocemente e dove sono necessarie modifiche e miglioramenti continui, l'approccio Agile è molto diffuso, poichè permette una risposta rapida ed efficiente a questi cambiamenti. Inoltre avere un *feedback* continuo su come sta procedendo lo sviluppo è essenziale per poter rispondere correttamente alle esigenze di business, come abbiamo visto nella sezione 2.2.

4.2 Framework Scrum e concetti chiave

Se consideriamo Agile come una filosofia allora possiamo dire che Scrum è la sua applicazione. Esistono varie implementazioni pratiche della metodologia Agile e sebbene tutte seguano il manifesto, ognuna di esse presenta delle differenze soprattutto in termini di concetti chiave. Il principio alla base di Scrum è di suddividere il lavoro in cicli chiamati *sprint*, solitamente dalla durata di due (come nel nostro caso specifico) o quattro settimane a seconda del progetto, con lo scopo di poter offrire valore al termine di ogni iterazione. Tra i concetti chiave di Scrum vi è la suddivisione dei ruoli. Come nell'applicazione più classica dell'approccio, il team di cui ho fatto parte, *Analytics*, era composto dalle seguenti figure:

1. *Product Owner:* responsabile della definizione delle funzionalità e delle caratteristiche del prodotto.

2. *Scrum Master:* persona incaricata della gestione e della pianificazione delle pratiche di Scrum.

3. *Development Team:* gruppo di sviluppatori di cui ho fatto parte, composto da studenti tirocinanti e professionisti interni all'azienda.

Un altra importante componente del framework riguarda gli eventi di Scrum, ossia un insieme di riunioni che coinvolge i membri del team, con scopi e frequenza differenti:

1. *Stand-up:* meeting giornaliero in cui il team si aggiorna sui recenti sviluppi del progetto.

2. *Grooming:* evento nel quale vengono create e stimate le attività necessarie all'avanzamento, avviene prima di ogni planning.

3. *Planning:* riunione in cui si pianificano le attività da portare a termine nello sprint successivo, la frequenza con cui avviene dipende dalla durata dello sprint.

4. *Technical Demo:* evento che avviene al termine dello sprint in cui viene presentato il lavoro svolto.

5. *Functional Demo:* evento di presentazione a cui partecipano anche *stakeholder*, durante il quale viene presentato il progetto al raggiungimento di un certo traguardo.

6. *Retrospective:* riunione in cui il team discute in modo critico dei successi e dei fallimenti dello sprint terminato cercando di trarne insegnamento.

Infine per completare la panoramica sul framework Scrum utilizzato, elenchiamo quelli che vengono definiti come artefatti, ovvero strumenti che il team sfrutta durante le fasi di sviluppo. Nel nostro caso sono tre:

1. *Product Backlog:* insieme di attività che devono essere completate dal team per la riuscita dei progetti.

2. *Sprint Backlog:* sottoinsieme di attività che vengono selezionate per essere completate in un ciclo di sprint.

3. *Increment:* insieme di tutti gli elementi completati durante il ciclo di sprint e che sono pronti ad essere presentati o rilaciati.

L'adozione di un approccio Scrum ha sicuramente avuto un impatto positivo, favorendo una stretta comunicazione e collaborazione tra i membri del team, permettendo di adattarci rapidamente ai cambiamenti ed aumentando notevolmente l'efficienza nello sviluppo.

4.2.1 Strumenti e tecnologie di supporto

L'impiego di strumenti appropriati è stato fondamentale per supportare efficacemente il processo Agile e facilitare la collaborazione del team. Ciascuno degli strumenti adottati ha avuto un ruolo specifico nel nostro flusso di lavoro, per la gestione delle attività, la comunicazione, la documentazione o il controllo di versione, garantendoci di mantenere degli standard elevati. Vediamo nel dettaglio i principali.

Lo strumento più importante, impiegato per la gestione del flusso di lavoro, è stato *Jira*[8], software sviluppato da *Atlassian*[9] che permette di tenere traccia e di pianificare l'evoluzione dei progetti. Avendo scelto Scrum come framework, Jira è stato essenziale per la gestione degli artefatti e il controllo degli sprint, ciò

ha permesso ad esempio, di tenere traccia dei backlog, dove sono stati inseriti e prioritizzati *tasks*, *stories* e *bugs* relativi ai vari progetti. Durante gli sprint planning, oltre che per l'assegnazione dei task e il monitoraggio del progresso di essi in tempo reale, Jira si è rivelato fondamentale per stimare le varie attività, che nel nostro team avveniva tramite il metodo del planning poker. L'impiego di questa tecnica prevede che ogni membro del team assegni dei valori numerici, che solitamente seguono la *sequenza di Fibonacci*[1], alle attività da stimare. Questi valori prendono il nome di *story points*, e rappresentano una misura del tempo, dove la durata delle attività è direttamente proporzionale alla grandezza del numero. La risorsa principale che ha permesso tutto questo, è stata la Scrum Board integrata nel software di gestione, un area da cui tramite un interfaccia grafica chiara è possibile gestire e visualizzare l'avanzamento del lavoro durante gli sprint.

Nonostante l'approccio Agile non ponga una rilevanza primaria sulla documentazione, il team ha adottato *Confluence*[10], un altro strumento di casa Atlassian, per supportare le attività di documentazione. Confluence è una piattaforma progettata per facilitare la creazione e la condivisione di conoscenza all'interno del team, offrendo funzionalità come la collaborazione in tempo reale, gestione della sicurezza e l'integrazione con altri strumenti della suite Atlassian, tra cui Jira. Il team ha utilizzato Confluence per documentare tutti gli aspetti relativi ai progetti, includendo i requisiti di business, le specifiche tecniche e le decisioni di sviluppo. Questo approccio ha consentito al gruppo di mantenersi costantemente aggiornato sullo stato e l'evoluzione dei progetti. Confluence si è dimostrato particolarmente efficace nel facilitare il trasferimento di conoscenza, aspetto cruciale soprattutto durante l'inserimento di nuovi membri nel team, situazione da me sperimentata in prima persona. Oltre alle sue funzioni primarie, il team ha sfruttato le potenzialità di Confluence per scopi aggiuntivi, come la registrazione e l'analisi dei feedback durante le riunioni di Retrospective e la documentazione dei costi associati all'adozione di nuovi servizi.

Il terzo strumento impiegato, legato alla suite Atlassian, è stato Bitbucket[11]. Bitbucket è una piattaforma dedicata al controllo versione basata su *Git* [2], che permette l'*hosting* di *repository* di codice in ambiente *cloud*. Un servizio come

[1]Successione di numeri interi dove ciascun numero è la somma dei due precedenti, fatta eccezione per i primi due che sono 0 e 1.

[2]Software di versionamento tra i più diffusi a livello globale.

Bitbucket risulta di fondamentale importanza quando si collabora con i membri di un team, permettendo di gestire le modifiche in modo distribuito e senza conflitti. Inoltre lo strumento offre funzionalità di sicurezza come controllo degli accessi e persistenza del codice. Ogni progetto del team disponeva di un repository dedicato, a cui solo i membri del gruppo potevano accedere, ed ogni progetto sufficientemente complesso utilizzava uno strumento di *build automation* associato, al fine di ottimizzare il processo di sviluppo. Questi strumenti si occupano di eseguire una serie di attività necessarie per la costruzione del software, come ad esempio compilazione del codice, esecuzione di test o la creazione di documentazione, riducendo gli errori e migliorando l'efficienza. L'integrazione di tutti gli strumenti della suite ha consentito di mantenere una forte coordinazione oltre a favorire la produttività nelle fasi di sviluppo.

Per quanto riguarda la comunicazione interna, lo strumento adottato è stato *Microsoft Teams* (MT). Questa piattaforma rappresenta una delle soluzioni più diffuse al mondo per la collaborazione, permettendo un collegamento immediato e costante tra i membri del team. Tra le funzionalità principali offerte da MT, vi è la possibilità di pianificare ed effettuare riunioni virtuali, una caratteristica ampiamente utilizzata nel nostro contesto sia per gli eventi Scrum che per incontri di varia natura con il gruppo di lavoro. La possibilità di comunicare tramite *chat* invece, non solo rappresenta un metodo di interazione più rapido, ma mette anche a disposizione un area per poter condividere file di diversa tipologia con altri collaboratori, integrandosi perfettamente con le altre applicazioni della suite Microsoft. Data la partecipazione di ogni membro del gruppo allo sviluppo in modalità remota, l'impiego di uno strumento come Teams ha assunto un'importanza ancor più rilevante nel facilitare la collaborazione e la coesione del team.

Come già accennato, la vera potenza di tutti questi strumenti è emersa dalla loro integrazione, ad esempio con le discussioni di Teams che facevano riferimento ai task su Jira o ai documenti di Confluence, o il collegamento tra i repository su Bitbucket e le attività in Jira. L'impiego di questi servizi e la loro integrazione ha supportato in maniera efficace il processo di sviluppo, facilitando la collaborazione, la tracciabilità e la pianificazione in ogni fase del progetto.

4.3 Fasi del processo di sviluppo

In questa sezione verranno discusse le fasi chiave del processo di sviluppo, ciascuna con obiettivi specifici e attività distinte, costruite sui principi delle metodologie Agile e Scrum.

4.3.1 Pianificazione

Dato il mio subentro nel team a progetto già avviato, parti delle fasi di pianificazione iniziale, come la raccolta dei requisiti, non mi hanno visto coinvolto, tuttavia di fondamentale importanza è stata la definizione degli obiettivi. L'obiettivo primario del mio lavoro all'interno del team era studiare strategie e sviluppare un modello di machine learning, capace di prevedere in anticipo e in maniera efficiente i clienti a rischio di churn. Altri obiettivi includevano la creazione di *report* interattivi per gli stakeholder e la documentazione dei progressi e delle performance legate al modello. Questi obiettivi sono stati registrati in Jira e monitorati durante tutto il ciclo di vita del progetto.

La successiva creazione di un backlog dinamico relativo al prodotto ha consentito di tenere traccia delle evoluzioni del modello durante i vari cicli di sprint, non tanto in termini di funzionalità ma piuttosto riguardo alle prestazioni del modello stesso. Tra le attività più dispendiose in termini di tempo e risorse vi sono state quelle per ottimizzare le performance dei vari modelli, oltre allo studio e all'analisi iniziale svolta sui dati. I task sono poi stati di volta in volta inseriti nel backlog, che è stato costantemente aggiornato e rivisto durante le sessioni di sprint planning.

4.3.2 Sviluppo

Il mio ingresso nel team ha coinciso con la partecipazione a quattro cicli di sprint consecutivi. Durante le sessioni di sprint planning il team si riuniva per selezionare le attività da sviluppare nel prossimo sprint, della durata di due settimane. Ogni task veniva assegnato ai membri del gruppo sulla base del progetto a cui stava lavorando o collaborando. Sebbene ciascuna attività fosse assegnata a un singolo sviluppatore, era prevista la possibilità di ricevere assistenza qualora necessario. Nel mio caso specifico, durante i primi sprint, al completamento delle attività seguiva una fase di revisione condotta da un collaboratore. Durante queste riunioni

il team discuteva le dipendenze, le possibili difficoltà e le risorse necessarie per completare le attività, rispettando i vincoli imposti da Scrum nello sviluppo. Al termine di ogni riunione, il piano dello sprint veniva documentato in Jira con relativi task e scadenze di ciascuno di essi.

Un aspetto fondamentale dello sviluppo era dato dagli *Stand-up meeting*. Con frequenza giornaliera e mattiniera, il team si riuniva per una durata di circa quindici minuti, durante il quale a turno ogni membro del gruppo discuteva rapidamente tre punti chiave: cosa ha fatto ieri, cosa farà oggi e se ci sono ostacoli che impediscono il progresso. Essere sempre costantemente aggiornati aveva lo scopo di garantire una risposta repentina ad eventuali problemi, offrire una comunicazione trasparente e migliorare la collaborazione. Nel mio lavoro il primo sprint è stato cruciale poichè ha coinvolto una fase di studio intensa del problema e lo sviluppo del primo modello. I restanti tre cicli di sviluppo si sono concentrati nel miglioramento delle prestazioni del modello, anche in seguito alla scoperta di un bug contenuto nei dati, dando luce allo sviluppo di tre diversi modelli ognuno con caratteristiche differenti. Ad ogni evoluzione nello sviluppo corrispondeva la relativa documentazione e al termine di ogni sprint ero coinvolto nell'attività di creazione del report sul modello per l'azienda cliente.

Da un punto di vista interno invece, ogni due settimane il team si riuniva per presentare il lavoro svolto, un opportunità per mostrare i recenti sviluppi, raccogliere feedback e discutere di eventuali modifiche o miglioramenti. Ogni dimostrazione veniva accompagnata dalla relativa documentazione del progetto e la riunione veniva registrata e resa disponibile ad eventuali figure assenti. La chiusura dei ogni sprint avveniva in concomitanza con la sessione di retrospective, anche in questo caso tutto documentato su Confluence e monitorato nel tempo per garantire un miglioramento continuo nello sviluppo.

4.3.3 Testing e rilascio

Durante il progetto, il mio ruolo è stato principalmente focalizzato sulle fasi di pianificazione iniziali e sviluppo. Il mio coinvolgimento si è concluso durante la fase di sviluppo, pertanto non ho partecipato alle fasi di *testing*, tranne che per le valutazioni delle performance del modello, e rilascio del prodotto. Nonostante ciò, ho potuto contribuire attivamente alle fasi principali dello sviluppo, analizzando

e documentando in maniera approfondita strategie e criticità, producendo anche tre modelli predittivi con peculiarità differenti, gettando una solida base per il proseguimento del progetto.

4.4 Strumenti di sviluppo e infrastrutture

Nel corso del progetto, abbiamo fatto affidamento su una serie di *software* e servizi *cloud* all'avanguardia, per supportare lo sviluppo, sia durante l'analisi come anche in fase di costruzione del modello. In particolare nella fase di addestramento l'evoluzione dei servizi utilizzati è stata guidata dalla necessità di una maggior potenza di calcolo, in modo da accelerare lo sviluppo.

4.4.1 Linguaggi e ambienti di sviluppo

Il linguaggio di programmazione adottato per il nostro progetto è stato *python*, molto diffuso nell'ambito della data science grazie alla sua versatilità e al ricco supporto offerto dalle molte librerie disponibili. Molte di queste librerie sono state cruciali nello sviluppo sia per il trattamento dei dati che per la creazione e la valutazione stessa dei modelli; tra le principali vi sono:

1. *Numpy[12]:* impiegata per operazioni numeriche e manipolazioni di array e matrici.

2. *Pandas[13]:* impiegata per la manipolazione e l'analisi dei dati.

3. *Scikit-learn[14]:* mette a disposizione strumenti per l'addestramento dei modelli e funzioni per misurarne le performance.

4. *Seaborn e Matplotlib[15][16]:* fondamentali per la visualizzazione dei dati e la creazione di grafici.

Il linguaggio python è stato impiegato in un ambiente *Jupyter Notebook*, vantaggioso per la flessibilità che offre nel poter combinare efficacemente l'esecuzione del codice, la visualizzazione dei risultati e la documentazione, oltre che la possibilità di collaborazione. La natura di linguaggio interpretato ha infatti permesso di eseguire codice Python in celle separate, consentendo un'analisi iterativa e una

rapida sperimentazione. Tra gli altri vantaggi, il notebook può inoltre essere arricchito con porzioni di testo, formule matematiche o grafici, rendendo evidente la potenza e l'importanza dell'utilizzo di questo strumento in contesti come il nostro. In una prima fase dello sviluppo, tutte le elaborazioni sono state condotte in locale con una ridotta capacità di calcolo, da qui la necessità di estrarre un sottoinsieme di dati rappresentativo dal dataset originale. In questo contesto è stato impiegato *DataSpell*, un *IDE* appartenente alla suite di *JetBrains*[3], robusto e specializzato per la data science. Oltre ad offrire strumenti di sviluppo comuni come autocompletamento del codice, strumenti di *debug* o controllo delle versioni, DataSpell mette a disposizione anche funzioni sofisticate per la *data visualization* e integrazioni con database.

4.4.2 Servizi e infrastrutture

Un contributo essenziale nello sviluppo, è stato dato da *Amazon Web Services*[17] (AWS)[4], che ha fornito strumenti ed infrastrutture indispensabili al raggiungimento di obiettivi che vanno dal salvataggio o la manipolazione dei dati, alla necessità di avere piattaforme di calcolo più performanti. Sebbene io non abbia partecipato al processo di *Extract, transform, Load* (ETL), dei dati e le successive implicazioni, dove molti dei servizi di AWS sono stati impiegati, ho comunque avuto modo di sfruttare alcuni di questi per lo sviluppo dei modelli. I servizi chiave sono stati:

1. *Amazon S3 (Simple Storage Service)[18]:* utilizzato per l'archiviazione di grandi volumi di dati, memorizzati come oggetti all'interno di risorse chiamate *bucket*.

2. *AWS Glue DataBrew[19]:* tipicamente impiegato per operazioni ETL, nel mio caso specifico è stato impiegato per la creazione di un *job* che confrontasse i risultati delle predizioni con i dati dei mesi successivi.

3. *Amazon Athena[20]:* servizio *query* che ci ha permesso di eseguire interrogazioni sui dati memorizzati su S3, visualizzare e scaricarne i risultati.

[3] Azienda di sviluppo software che produce strumenti e ambienti di sviluppo per aziende e specialisti in campo IT.

[4] Azienda che fornice servizi di cloud computing, flessibili e disponibili su scala globale.

Dopo un periodo iniziale di addestramenti in locale, si è pensato di spostare il progetto su AWS *SageMaker Studio Lab*[21], un servizio gratuito che offre un ambiente di sviluppo per il machine learning. La piattaforma mette a disposizione un infrastruttura in cloud completamente gestita da AWS della quale è possibile usare l'hardware per l'archiviazione e il calcolo. Questo ci ha permesso di sfruttare una macchina più performante per velocizzare il processo di sviluppo e ha giovato in termini di collaborazione data la natura del servizio. Nonostante Studio Lab sia completamente gratuito, presenta delle limitazioni dettate da limiti di tempo imposti nell'utilizzo dell'infrastruttura, nello specifico sono a disposizione un massimo di quattro ore di GPU e otto di CPU per sessione, con ripristino giornaliero. Sebbene non sia chiara l'effettiva potenza dell'hardware alla base della macchina, questa ha comunque contribuito allo sviluppo permettendo, anche se di poco, di misurare le prestazioni del modello con dei subset di training di dimensione maggiore. Nell'ultima fase dello sviluppo il team ha pensato di spostare nuovamente il progetto, questa volta su AWS *SageMaker Studio*[22], la controparte a pagamento di Studio Lab. Oltre a tutte le feature offerte dalla piattaforma gratuita, AWS Studio integra anche funzionalità più sofisticate, ma il suo punto di forza, che ha motivato il cambio, è la possibilità di poter determinare le specifiche hardware dell'istanza. Ogni piano presenta dei costi orari associati, che variano in base alla potenza di calcolo, nel nostro caso l'istanza scelta montava 16 vCPU e 64 GiB di memoria, che hanno ridotto notevolmente le attese in fase di addestramento e hanno consentito training fino a 500 mila record in maniera efficiente.

La combinazione di tutti questi servizi e strumenti, dettagliatamente documentati in fase di sviluppo, ha garantito la flessibilità e la potenza necessaria per poter affrontare nel migliore dei modi le sfide imposte dal progetto, consentendo anche una coesione e una collaborazione maggiore con il team.

Capitolo 5

Sviluppo dei modelli predittivi

In questo capitolo affronteremo in maniera dettagliata la costruzione dei modelli, inclusa la scelta degli algoritmi, delle metriche di valutazione, e delle tecniche di ottimizzazione, motivando ciascun punto. I risultati e le prestazioni ottenute, verranno tuttavia trattate più avanti per mantenere una certa scorrevolezza, saranno pertanto analizzate nel capitolo successivo. Inoltre avendo già discusso le fasi di preparazione dei dati e feature engineering, non ci ripeteremo, ciononostante ove necessario verranno esaminate le peculiarità dei singoli dataset o i valori utilizzati per i parametri degli algoritmi impiegati nel preprocessing. Vediamo innanzitutto una panoramica delle tecniche utilizzate per poi passare ad una descrizione specifica dei singoli modelli.

5.1 Algoritmi di machine learning utilizzati

Durante il primo ciclo di sprint, che ha compreso un importante fase di analisi, sono stati testati più algoritmi per la previsione del churn. Gli algoritmi di machine learning sono la chiave di tutto il processo di addestramento, possiamo immaginarli come una serie di regole e passi mediante i quali vengono trovati patttern e schemi che consentono al modello di apprendere dai dati. Vi è una grande varietà di algoritmi che si prestano meglio a contesti diversi, come ad esempio il paradigma

di learning, la forma ed il tipo dei dati, o il task con cui si ha a che fare. Tutti gli algoritmi utilizzati nel nostro caso sono nel contesto dell'addestramento supervisionato, con task di classificazione. Ognuno di questi ha quindi caratteristiche uniche e presenta vantaggi e svantaggi specifici. Prima di entrare nel dettaglio di ciascun algoritmo, diamo anticipatamente una definizione di *ensemble learning* (apprendimento d'insieme): tecnica che combina le previsioni di più modelli semplici per ottenere una prestazione complessiva migliore rispetto all'applicazione dei singoli modelli. A sua volta l'apprendimento d'insieme può essere suddiviso in tre tecniche:

1. *Bagging:* abbreviazione di *Bootstrap Aggregating*, prevede di dividere il dataset in campioni casuali (bootstrap) e di addestrare su ogni partizione un modello semplice, al cui termine vengono combinate le previsioni per ottenere il risultato finale. Possiamo, se aiuta, immaginare la tecnica come una serie di training fatti in parallelo al termine dei quali si sceglie il miglior modello. L'obiettivo è quello di migliorare la stabilità riducendo la varianza, particolarmente efficace nel caso di dataset rumorosi.

2. *Boosting:* tecnica dove ciascun modello in sequenza cerca di correggere gli errori del precedente. Vengono assegnati dei pesi identici a tutti gli esempi. Durante l'addestramento vengono aumentati i pesi degli esempi classificati erroneamente, al termine vengono combinate le previsioni di tutti i modelli in una finale. Vantaggioso nel ridurre il bias e migliorare la capacità predittiva complessiva.

3. *Stacking:* tecnica in cui vengono combinati più modelli in un unico metamodello. I singoli modelli di base vengono addestrati sull'intero dataset. Le predizioni generate da questi vengono poi utilizzate come input del metamodello che restituirà la previsione finale. In questo modo vengono combinati i benefici dei vari modelli in uno singolo, aumentando tuttavia la complessità.

Queste tecniche rappresentano una variante più potente dei modelli semplici, permettendo di ottenere previsioni più accurate e robuste, ma con lo svantaggio di una maggiore complessità del modello.

Un altro argomento di cui è necessario discutere anticipatamente per comprendere meglio i paragrafi seguenti sono i *Decision Tree*[23]. Gli alberi decisionali

rappresentano una tecnica di addestramento molto potente e diffusa nel contesto del machine learning, utilizzabile sia per problemi di classificazione che di regressione. La struttura di ogni albero può variare, ma in generale sono composti da:

1. *Nodo Radice:* è il nodo da cui viene effettuata la prima divisione (*split*), basata su una caratteristica.

2. *Nodi Intermedi:* ciascun nodo intermedio è associato a una feature del dataset, ogni nodo divide i dati in base ad un valore specifico di quella feature.

3. *Nodi Terminali:* chiamati anche foglie, sono i nodi finali dell'albero e rappresentano le previsioni del modello.

4. *Rami:* rappresentano il percorso di congiunzione tra i nodi, ogni ramo corrisponde ad una decisione presa dall'algoritmo.

Questa tecnica rappresenta una soluzione estremamente versatile sopratutto per problemi di classificazione, il cui più grande punto di forza è l'interpretabilità del modello.

5.1.1 Logistic Regression

L'algoritmo di regressione logistica stima la probabilità che si verifichi un evento sulla base delle variabili indipendenti usate come input del modello. Contrariamente a quanto suggerisce il nome, Logistic Regression è comunemente utilizzato per risolvere task di classificazione binaria, dove l'obiettivo è predire una delle due classi possibili con un certo valore di probabilità. Il modello comincia combinando linearmente le variabili in input, e successivamente viene applicata una funzione sigmoide per trasformare il risultato in un valore di probabilità compreso tra zero e uno. La funzione sigmoide è quella che nella rappresentazione cartesiana assume una forma ad "S". Una volta ottenuta la probabilità, è possibile classificare l'osservazione: dato un valore di *threshold*, se la probabilità è maggiore allora il risultato è positivo (1), altrimenti, negativo (0). Il valore di soglia può essere scelto arbitrariamente ma nel caso di classificazione binaria è impostato di default a 0.5. Al termine di questi passaggi viene utilizzata una funzione di costo, che misura le

prestazioni del modello sulla base degli errori che commette. Più il valore di costo è piccolo e migliori sono le performance del modello. Questi passaggi si ripetono per un certo numero di volte, fino al raggiungimento di una limite massimo o di un criterio di stop. Questo algoritmo si distingue principalmente da *Linear Regression* per il contesto in cui viene utilizzato, infatti la regressione lineare è adatta in contesti di previsione di valori continui. Inoltre il risultato prodotto da questo modello è un numero continuo e non un valore di probabilità. Nell'implementazione offerta da Scikit-learn l'algoritmo può essere configurato con vari parametri, alcuni dei quali sono:

1. *tol:* valore di tolleranza che determina quando fermare l'ottimizzazione in base alla variazione della funzione di costo.
2. *solver:* definisce l'algoritmo di ottimizzazione da utilizzare.
3. *max_iter:* numero massimo di iterazioni per gli algoritmi di ottimizzazione.
4. *class_weight:* Vengono assegnati dei pesi alle classi imputando un valore maggiore alla classe di minoranza. Il modello verrà penalizzato di più per gli errori su quella classe, bilanciandone l'importanza.

Logistic Regression rappresenta una soluzione leggera e che richiede poche risorse computazionali, tuttavia, oltre a non essere adatto a problemi non lineari, le previsioni possono essere fortemente influenzate da rumore presente nei dati. La scelta di utilizzare di questo algoritmo è stata motivata principalmente dalla sua semplicità e dall'interpretabilità dei risultati. Oltre a ciò, richiedere poche risorse computazionali è stata una caratteristica essenziale, sopratutto nella prima fase dello sviluppo in cui gli addestramenti avvenivano in locale.

5.1.2 Support Vector Machine (SVM)

Come l'algoritmo precedente, anche Support Vector Machine[24] fa parte dei modelli lineari. L'idea soggiacente alla tecnica è di trovare un "iperpiano" in uno spazio bidimensionale o multidimensionale che separi al meglio due classi di dati. Durante la fase di ricerca dell'iperpiano viene generata una linea, nel caso il problema sia bidimensionale, ed un iperpiano altrimenti. La posizione e l'orientamento del separatore sono determinati dall'equazione relativa all'iperpiano, che

comprende un vettore di pesi e un termine bias. I punti nel grafico vengono poi classificati come appartenenti ad una classe piuttosto che ad un altra in base al segno del risultato che ottengono nella separazione (positivo o negativo). Durante la separazione l'algoritmo cerca di massimizzare il margine, ovvero la distanza tra l'iperpiano e i punti più vicini di ciascuna classe, questi punti sono chiamati vettori di supporto. L'obiettivo della massimizzazione della distanza permette di ottenere una separazione più netta, migliorando la capacità di classificazione di nuovi punti del modello. I concetti di *Soft Margin* e *Hard Margin* determinano la tolleranza agli errori, nel primo caso consentiamo la presenza di errori, situazione comune quando i dati non sono perfettamente separabili. Nel secondo caso l'approccio è più rigoroso ed è applicabile quando i dati sono perfettamente separabili. La quantità di errori tollerata è determinata dal valore di un iperparametro C, più è alto il valore di C, minore sarà la tolleranza per gli errori e più stretto sarà il margine. Oltre ai già citati parametri di regolarizzazione C e della funzione di separazione, i parametri configurabili forniti da Scikit-learn per l'implementazione di SVC (Support Vector Classifier), rimangono simili a quelli già visti per Logistic Regression. Support Vector Machine risulta molto efficace nel caso in cui i dati siano facilmente separabili, tuttavia la complessità computazionale aumenta con l'aumentare della dimensione del dataset. Nel nostro caso l'algoritmo è stato impiegato come metro di paragone per i risultati restituiti dagli altri algoritmi, specialmente durante l'utilizzo di un subset dalla dimensione ridotta.

5.1.3 Gradient Boosting:

Dopo aver visto i modelli semplici, analizziamo gli algoritmi di ensemble. Come suggerito dal nome, questa tecnica appartiene alla categoria degli algoritmi di boosting, in quanto vengono addestrati in sequenza una serie di modelli deboli, che vengono combinati per ottenerne uno forte. I modelli alla base di GB[25] sono tipicamente basati su alberi decisionali, ed ogni nuovo albero è costruito per correggere gli errori di quello precedente, minimizzando una funzione di perdita specifica. Ad ogni iterazione vengono utilizzati i gradienti, ovvero gli errori commessi dal modello, per guidare l'ottimizzazione. Questo processo viene ripetuto più volte, fino al raggiungimento di un criterio di stop o ad un numero massimo di iterazioni. Sulla base della tecnica Gradient Boosting sono nate anche delle

implementazioni più complesse, come *Extreme Gradient Boosting (XGBoost)*[26] che tra le varie caratteristiche, include una serie di ottimizzazioni e la capacità del modello di gestire autonomamente dati sparsi. Per quanto riguarda i parametri di Gradient Boosting nell'implementazione di Scikit-learn, troviamo:

1. *loss:* rappresenta la funzione di perdita specifica da ottimizzare.
2. *n_estimators:* parametro comune a tutti i metodi di ensemble, definisce il numero di stimatori, in questo caso di alberi decisionali.
3. *min_samples_split:* numero minimo di campioni richiesti per dividere un nodo.
4. *min_samples_leaf:* numero minimo di campioni presenti in una foglia.
5. *learning_rate:* specifica il contributo di ogni singolo albero nell'addestramento.

Questi rappresentano solo alcuni dei parametri configurabili, e spesso la scelta dei valori corretti può essere molto complicata e lunga. Il tuning degli iperparametri e la complessità computazionale rappresentano infatti gli svantaggi più grandi nell'impiego di questo algoritmo. Tra i vantaggi invece troviamo la flessibilità offerta dagli alberi decisionali, e la capacità del modello di "autocorreggersi", caratteristiche che ne hanno motivato l'impiego.

5.1.4 Adaptive Boosting (AdaBoost)

Anche in questo caso abbiamo a che fare con un algoritmo di boosting, pertanto vengono combinati più modelli deboli per creare un classificatore forte[27]. Ogni modello viene addestrato prestando più attenzione agli errori commessi dai modelli precedenti, ponendo un peso maggiore agli esempi che sono stati classificati erroneamente. Nell'implementazione più classica, i modelli alla base di AdaBoost sono alberi decisionali, che hanno la peculiarità di avere altezza uno. Questi alberi sono chiamati *stump*, e sono formati da un nodo radice e due foglie. Ogni campione del dataset ha inizialmente lo stesso peso, calcolato come $\frac{1}{N}$, dove N rappresenta il numero di campioni. Durante l'addestramento i pesi cambiano: se la classificazione è corretta il valore diminuisce, se invece è errata aumenta, questo

assicura che il modello successivo si concentri sui campioni con peso maggiore. Ad ogni modello viene assegnato un punteggio che determina il suo impatto nel modello finale, questo valore cresce in maniera inversamente proporzionale al numero di errori commessi. In altre parole meno errori commette il modello e più avrà un ruolo decisivo nella costruzione del classificatore finale. Queste fasi si ripetono fino al raggiungimento di un criterio di stop o di un limite massimo di iterazioni. Gli iperparametri di AdaBoost non permettono una grande configurazione, ma piuttosto definiscono le caratteristiche dei modelli deboli, come la quantità di stimatori o il learning rate, come abbiamo visto nel caso di Gradient Boosting. Questo algoritmo risulta estremamente efficiente in termini di risorse e semplice da utilizzare, caratteristiche fondamentali nel nostro studio. Tuttavia le prestazioni possono soffrire nel caso di dati rumorosi e, se non gestito correttamente, il rischio di overfitting aumenta.

5.1.5 Random Forest

L'ultimo algoritmo di ensemble utilizzato è stato Random Forest[28], una delle applicazioni più comuni della categoria Bagging. Come suggerito dal nome, anche in questo caso abbiamo a che fare con un modello basato su alberi decisionali, le cui caratteristiche possono essere controllate tramite dei parametri. La tecnica prevede di combinare più modelli, per migliorare le prestazioni complessive del modello finale. L'algoritmo Random Forest può essere impiegato sia per problemi di classificazione che per problemi di regressione, a seconda dell'implementazione cambia il modo in cui viene calcolato il risultato. Durante l'addestramento, ogni albero decisionale viene allenato su un sottoinsieme casuale di dati, estratto dal dataset di partenza. Ogni albero fornice una predizione, alla quale segue una fase di *voting*. Nei problemi di classificazione la votazione consiste nel contare quante volte ciascuna classe della stessa istanza è stata predetta, quella che compare più volte (maggior numero di voti) definisce la previsione finale. Dato il grande numero di iperparametri configurabili, questo algoritmo risulta tanto potente quanto complesso, di conseguenza anche meno interpretabile e richiede più tempo e risorse per l'addestramento. Alcuni di questi sono:

1. *max_ depth:* indica la profondità massima di ogni albero.

2. *max_features:* specifica il numero massimo di feature da considerare per la costruzione degli alberi.

3. *bootstrap:* determina se gli alberi sono costruiti su campioni diversi o sull'intero dataset.

4. *max_leaf_nodes:* rappresenta il numero massimo di foglie che un albero può avere.

Nonostante ciò Random Forest si rivela essere molto robusto ed efficiente nel gestire l'overfitting, rendendolo uno dei suoi maggiori pregi, specie nel caso di dataset di grandi dimensioni. Oltre ad essere un algoritmo molto gettonato nei problemi di churn prediction, questi vantaggi sono le caratteristiche che ne hanno motivato l'impiego.

5.1.6 Artificial Neural Network (ANN)

Nelle fasi più avanzate dello sviluppo è stata implementata una rete neurale[1] basata sull'algoritmo *Multi-layer Perceptron* (MLP)[29]. MLP è una tecnica che si adatta bene sia a problemi sia di classificazione che di regressione. Il modello è costituito da diversi strati di neuroni (o nodi) interconnessi, che ne definiscono la struttura:

1. *Input layer:* ogni nodo in questo strato rappresenta una variabile indipendente.

2. *Hidden layer:* gruppo di uno o più strati, nei quali i dati vengono processati e trasformati per fornire una risposta.

3. *Output layer:* in questo strato il numero di neuroni varia a seconda del problema, qui viene fornita la risposta finale del modello.

Nella fase iniziale i pesi delle connessioni tra i nodi sono impostati a valori casuali. Successivamente tramite la tecnica di *forward propagation*, ogni input dopo somme e moltiplicazioni di pesi e bias viene passato ad una funzione di attivazione, e il risultato propagato ai neuroni successivi. Ad ogni iterazione segue una fase di *backpropagation* allo scopo di ottimizzare il modello. La tecnica consiste nell'aggiornare i pesi in modo da ridurre l'errore, questo processo, come suggerito, dal

[1] categoria di modelli di machine learning, il cui funzionamento è ispirato al cervello umano.

nome avviene "all'indietro", in quanto comincia dallo strato di output e arriva a quello di input. Sebbene questa sia una soluzione estremamente potente ed utile nel contesto di problemi complessi, il suo utilizzo è stato abbandonato dopo una prima fase di test. Le motivazioni che hanno spinto a rinunciare a questo algoritmo sono state molteplici, tra cui l'estrema complessità del modello e la grande quantità di risorse necessarie per l'addestramento, oltre che per la difficoltà di scegliere e ottimizzare i parametri correttamente e la scarsa interpretabilità dei risultati.

Ogni modello presenta vantaggi e svantaggi specifici che lo rendono più o meno adatto a seconda di fattori come le risorse di calcolo disponibili, l'interpretabilità, la complessità del problema o la natura dei dati. La scelta dei modelli finali impiegati è stata determinata dalla valutazione di queste caratteristiche oltre che dai risultati restituiti da ciascun algoritmo. Per questi motivi il primo ed il terzo modello hanno visto l'implementazione di Random Forest, mentre il secondo è stato addestrato utilizzando Adaptive Boosting.

5.2 Metriche di valutazione delle performance

Nell'ambito dell'analisi predittiva, la valutazione accurata dei modelli di machine learning è fondamentale per garantire che le previsioni siano affidabili ed ottimali. Le metriche di performance (KPI) rappresentano uno strumento indispensabile per poter quantificare queste necessità. La scelta di queste metriche può dipendere da molti fattori che dipendono dalla natura del problema e dalla natura dei dati. Vediamo nel dettaglio alcuni degli strumenti e delle KPI comunemente utilizzate nel contesto dell'analisi predittiva, e la scelta delle stesse nel nostro problema.

5.2.1 Confusion Matrix

La matrice di confusione è uno strumento fondamentale nella valutazione delle performance, in quanto permette di capire con una rappresentazione grafica, come il modello classifica i dati. All'interno di una matrice 2x2, nel caso di problemi di classificazione binaria, vengono catalogati gli esempi in forma numerica, suddivisi per:

1. *True Negative (TN):* sono gli esempi negativi che il modello classifica correttamente come negativi.

2. *False Positive (FP):* è il numero di esempi negativi che il modello classifica erroneamente come positivi.

3. *False Negative (FN):* rappresenta il numero di esempi positivi classificati erroneamente come negativi.

4. *True Positive (TP):* sono gli esempi positivi che il modello riconosce correttamente come positivi.

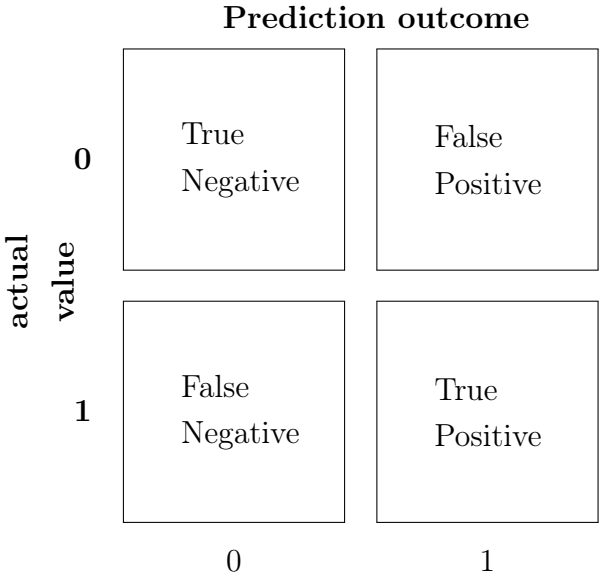

Figura 5.1: rappresentazione grafica della matrice di confusione

Come si evince dalla figura 5.1, questa rappresentazione consente una rapida valutazione di dove il modello sta commettendo errori e di quale tipo, permettendo un analisi più approfondita delle cause e delle possibili soluzioni. Le classi TP, TN, FP ed FN sono concetti fondamentali nella misura delle performance, sono pertanto ripresi ed utilizzati anche dalle metriche di valutazione.

5.2.2 Accuracy

L'accuratezza rappresenta la proporzione degli esempi classificati correttamente, sul totale degli esempi. Questa metrica viene pertanto calcolata come il rapporto

tra il numero di predizioni corrette, e il numero totale di predizioni. La formula associata all'accuratezza è la seguente:

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

In contesti in cui si ha a che fare con dataset sbilanciati, come spesso accade nei problemi di analisi del churn, non è una metrica da considerare affidabile. Immaginiamo il caso in cui il 95% dei record nel nostro dataset appartenga alla classe negativa e il restante 5% a quella positiva. Se il nostro modello predicesse sempre la classe negativa avremo comunque un *accuracy* del 95%, ovvero un punteggio alto ma fuorviante, perchè il modello non classifica nemmeno un esempio positivo come tale.

5.2.3 Precision, Recall e F1-score

Tra le metriche che danno un quadro più completo delle performance del modello, sopratutto in contesti con dataset sbilanciati, troviamo la precisione e il *recall*. La *precision* misura la proporzione di esempi predetti come positivi che sono effettivamente positivi. Il valore di precisione si calcola tramite la seguente formula:

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

Risulta essere particolarmente quando l'obiettivo è ridurre i falsi positivi e quindi ottenere previsioni più precise.

Il recall, detto anche sensibilità, misura la proporzione di tutti gli elementi positivi trovati, sul totale dei positivi reali. La formula associata al calcolo del recall è:

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

Questa metrica è spesso utilizzata quando si vogliono riconoscere più positivi possibili, anche a costo di commettere più errori nella predizione. Spesso queste due metriche sono in conflitto, ad esempio, se volessimo massimizzare il recall e quindi volessimo minimizzare i falsi negativi, questo ci porterebbe ad aumentare il numero di esempi classificati come positivi e di conseguenza ad avere potenzialmente una minore precisione. Oppure nel caso ci interessi massimizzare la precision e quindi volessimo minimizzare i falsi positivi, questo ci porterebbe a diminuire il numero di esempi classificati come positivi e quindi un recall minore.

La terza metrica che analizziamo è chiamata *F1-score* o *F-measure* e rappresenta un compromesso tra precisione e sensibilità. L'utilizzo di F1-score risulta particolarmente utile nei casi in cui si voglia trovare un equilibrio tra le due metriche, in quanto questa è definita come la media armonica delle due. Il valore della F-measure si può calcolare tramite la seguente formula:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Tra le varie caratteristiche, maggiore è la differenza tra le due misure e minore sarà il valore della F-measure, mentre più sono vicine e più la F-measure sarà circa la metà dei due valori.

5.2.4 AUC-ROC

La *Reciver Operating Characteristic* (ROC)[30], rappresenta una curva definita all'interno di un grafico bidimensionale, che illustra le performance del modello in termini di benefits (TP) e costi (FP). Il grafico è costruito a partire dai valori di FPR (*False Positive Rate*), sull'asse delle ascisse, e TPR (*True Positive Rate*), sull'asse delle ordinate. FPR e TPR rappresentano altre due metriche di valutazione e sono rispettivamente, la proporzione di esempi classificati come positivi tra quelli che sono in realtà negativi e la proporzione di esempi classificati come positivi tra il totale dei positivi reali (misura analoga al recall).

$$FPR = \frac{TP}{FP + TN} \qquad TPR = \frac{TP}{TP + FN}$$

I valori di TPR e FPR definiscono poi le coordinate della curva all'interno del grafico. Più i punti sono collocati nell'angolo in alto a sinistra migliori sono le performance del modello, mentre se il classificatore è situato nella metà del triangolo inferiore le prestazioni sono pessime. Per comparare i classificatori occorre rappresentare l'indice di performance di ROC con un valore: l'area sotto la curva (AUC) ci fornice questo valore. Il risultato è compreso tra 0 e 1 (un valore minore di 0.5 indicherebbe delle pessime classificazioni), e maggiore è l'area e migliori sono le prestazioni del modello. Un caso particolare avviene quando il punteggio di AUC-ROC è esattamente 0, ciò significherebbe che il classificatore distingue benissimo le classi, ma inverte positivi con negativi. Questa metrica fornisce una

misura complessiva della capacità di discriminazione del modello, e questo risulta particolarmente utile ed efficace in presenza di dataset sbilanciati.

Nel corso dello sviluppo, AUC-ROC è stata scelta come metrica principale per la valutazione, dato il suo focus sulla capacità discriminatoria del modello in contesti sbilanciati. Il suo utilizzo è stato affiancato da metriche come precisione, recall ed F1-score, e strumenti come la matrice di confusione, per garantire un quadro più completo. Ogni modello è stato sviluppato con un idea di ottimizzazione diversa, cercando di fornire soluzioni differenti allo stesso problema.

5.3 Addestramento e valutazione

Il processo di sviluppo dei modelli ha seguito delle fasi ben definite, partendo dalla preparazione del dataset fino alla valutazione dei modelli stessi. La creazione di tutti i modelli ha previsto l'utilizzo di due dataset il primo dedicato all'addestramento del modello ($train\_test\_set$), mentre il secondo alla valutazione delle performance ($evaluation\_set$). Nonostante le differenti tecniche utilizzate, per ognuno di questi sono state impiegate delle procedure standard come lo *split* del dataset di training. Questa tecnica ha generato due insiemi X_train e X_test distinti, dei quali il primo è stato utilizzato per l'allenamento del modello, mentre il secondo per una fase di test preliminare. Durante l'addestramento il modello usa variabili indipendenti e dipendenti, mentre durante il test le previsioni sono fatte su dati non visti durante il training, e la classe target viene utilizzata per confrontare queste previsioni con i valori reali di y. Avendo a disposizione due dataset differenti già divisi, questa procedura non era essenziale, ma è stata comunque implementata per poter misurare anticipatamente su un insieme più piccolo le prestazioni del modello, riducendo il costo computazionale. Una volta sviluppato il modello e fatta una prima valutazione, la misura delle performance effettive veniva fatta sul dataset dedicato. Su questo insieme sono state applicate le stesse manipolazioni svolte sul set di training, con l'aggiunta che la classe target veniva separata delle altre features ed utilizzata per il confronto finale una volta ottenute le previsioni. In questa fase il modello non è stato minimamente toccato, pertanto sua la capacità predittiva e le performance sono il risultato di ciò che ha "imparato" durante l'addestramento. E' importante sottolineare che le prestazioni del modello risultano in tutti i casi inevitabilmente più basse nell'evaluation_set piuttosto che nel

test_set, in quanto sono presenti molti più dati ed è più facile commettere errori. In seguito alla valutazione, i risultati, che in tutti gli evaluation_set riguardano il mese di novembre 2023, venivano importati su un istanza s3 e confrontati tramite un job di glue con i dati relativi ai tre mesi successivi (dicembre 2023, gennaio 2024, febbraio 2024), per poter misurare quanto in anticipo riusciva a predire il modello. In altre parole si voleva verificare se i clienti predetti come falsi positivi, in un futuro si sarebbero trasformati in veri positivi.

Durante la costruzione del primo modello (*Modello 0*), è stato rilevato un problema significativo che ha influenzato l'affidabilità delle performance iniziali. In particolare, durante la fase di validazione, ci siamo accorti che le prestazioni del modello risultavano insolitamente elevate rispetto alle attese, sollevando sospetti sulla correttezza del processo. Dopo un'analisi approfondita, è emerso che il dataset conteneva un *bug* che comprometteva la qualità dei dati di input. Questo bug si manifestava attraverso la mancanza di alcune informazioni relative ai record dell'ultima bolletta di clienti churned, come ad esempio il nome dell'offerta o l'importo della stessa. Questa situazione ha portato ad una stima non attendibile delle performance del modello durante l'addestramento e la valutazione, che pertanto non verranno affrontate e discusse in questo lavoro di tesi. Dopo la risoluzione del problema e la conseguente rielaborazione del dataset, è stato necessario ripetere l'intero processo di addestramento e valutazione del modello, creando il *Modello 1*. Questo ha permesso di ottenere performance più realistiche e attendibili, riflettendo la reale capacità del modello di generalizzare su dati non visti. Il successivo *Modello 2* è nato dalla necessità di implementare nuove soluzioni per cercare di migliorare le performance del precedente, il che ha richiesto una nuova fase di studio del problema, con una conseguente nuova fase di sviluppo associata. Con il *Modello 3* si è voluto affrontare il problema con una strategia diversa, cercando di preferire una maggior precisione, invece di massimizzare il recall, soluzione che portava il modello a commettere anche molti più errori. Con questa nuova strategia l'obiettivo era quello di ridurre le previsioni errate del modello, fornendo dei risultati più affidabili, anche se con un recall minore. Nonostante mancassero dei requisiti specifici sulle prestazioni, questa soluzione, sebbene riconosca meno positivi, potrebbe essere comunque preferibile per l'azienda cliente, in quanto permette l'adozione di strategie contro l'abbandono più mirate sui clienti più a rischio.

5.4 Modello 1

Le tecniche e le strategie adottate riflettono quelle impiegate per la costruzione del modello zero, in quanto lo sviluppo del primo modello è avvenuto in seguito alla risoluzione del bug che coinvolgeva i dati di training. L'obiettivo primario che ha guidato la creazione del Modello 1 è stato quello di massimizzare l'identificazione dei clienti a rischio di abbandono. La dimensione del subset scelta per l'addestramento di questo modello è stata di 50 mila record, sufficientemente piccola per eseguire il training in locale, mantenendo una certa rapidità nel processo.

Dopo una prima fase di preparazione dei dati si è optato per eseguire un sovracampionamento, aggiustando la proporzione tra le classi, in modo da favorire la capacità discriminativa del modello. La tecnica di oversampling impiegata è stata SMOTE[31], descritta dal Codice 5. Il rapporto desiderato tra la classe minoritaria e quella maggioritaria è stato ottenuto tramite l'assegnazione di $sampling\_strategy = auto$, che ha portato ad una proporzione 1:1. In altre parole al termine del sovracampionamento all'interno del dataset erano presenti il 50% di clienti churned e il 50% di not churned.

Codice 5: Implementazione di SMOTE per l'upsampling.

```
[ ]: smote = SMOTE(random_state=42, sampling_strategy=auto, k_neighbors=5)
     X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
```

In aggiunta all'utilizzo di SMOTE, si è pensato di ridurre la dimensione del dataset eliminando le feature meno rilevanti. Per questa procedura è stata adottata una tecnica di selezione delle caratteristiche, nello specifico l'algoritmo Recursive Feature Elimination with Cross-Validation (RFECV)[32], come mostrato dal Codice 6. Sebbene in questa fase di sviluppo non si sia posta particolare attenzione all'ottimizzazione dei parametri, il valore assegnato alla cross validation è stato 10, e la variante impiegata è stata *StratifiedKFold* ($StratifiedKFold = 10$), per garantire che ciascuna divisione avesse una rappresentanza proporzionale delle classi. Per quanto riguarda la scelta della metrica di valutazione per la selezione delle feature, si è utilizzata $scoring = auc\_roc$, data la sua versatilità.

Codice 6: Implementazione di RFECV per la feature selection.

```
[ ]: cv = StratifiedKFold(10)
rfecv = RFECV(RandomForestClassifier(), cv=cv, scoring='roc_auc', n_jobs=-1)
rfecv.fit(X, y)
```

Terminata questa fase preparatoria avanzata e diviso il dataset in X_train e X_test, è cominciato il vero e proprio processo di addestramento. In questa fase, l'algoritmo Random Forest è stato selezionato per la sua notevole flessibilità e robustezza. Tale scelta è stata ulteriormente avvalorata dai risultati preliminari ottenuti durante l'analisi iniziale, in cui questa tecnica si è distinta per le sue prestazioni superiori rispetto agli altri algoritmi impiegati. Nel Codice 7 è possibile vedere quali parametri sono stati utilizzati durante l'addestramento del modello.

Codice 7: Implementazione di Random Forest per l'addestramento.

```
[ ]: model_rf = RandomForestClassifier(n_estimators=1000,
                                      n_jobs = -1,
                                      random_state =50,
                                      max_features = 'sqrt',
                                      bootstrap=False,
                                      max_leaf_nodes=100
                                      )

model_rf.fit(X_train_resampled, y_train_resampled)
# Make predictions
prediction_test = model_rf.predict(X_test)
```

La scelta dei parametri è stata guidata dall'intenzione di ridurre il rischio di overfitting. Un alto numero di alberi ($n\_estimator$), tende a migliorare la stabilità e la precisione del modello, aumentando tuttavia le risorse necessarie per l'addestramento. Inoltre $max\_leaf\_nodes = 100$ definisce un limite massimo al numero di foglie che ciascuno stimatore può avere, contribuendo a ridurre il rischio di sovradattamento. Tra gli altri parametri rilevanti troviamo: $bootstrap = False$, che indica al modello di addestrare gli alberi sull'intero dataset anzichè su campioni diversi, e *max_features='sqrt'*, che rappresenta un valore standard nei problemi di

classificazione come il nostro. Al termine del processo di addestramento, abbiamo proseguito con una fase iniziale di test, seguita da una valutazione conclusiva delle prestazioni del modello. Quest'ultima è stata condotta mediante l'impiego di metriche selezionate e strumenti appropriati. Le prestazioni di questo e dei successivi due modelli saranno trattate nel prossimo capitolo.

5.5 Modello 2

In seguito al secondo sprint, che ha coinvolto lo sviluppo del Modello 1, ci si è concentrati sull'ottimizzazione delle performance e il tuning degli iperparametri, azioni che hanno portato alla creazione del Modello 2. Con questo è importante sottolineare che anche durante lo sviluppo del Modello 2 la strategia adottata è stata massimizzare l'identificazione dei clienti a rischio di churn. Durante questo periodo, durato un intero sprint, sono state testate nuove tecniche, in particolare per le fasi di preprocessing avanzato e addestramento del modello. Alcune di queste sono state l'implementazione di una strategia di undersampling tramite l'impiego di *Random Under-Sampler*[33], oppure di dimensionality reduction, mediante l'utilizzo di *Principal Component Analysis*[34]. Il funzionamento dell'algoritmo Random Under-Sampler si basa sulla riduzione del numero di esempi della classe maggioritaria, selezionando casualmente un sottoinsieme di campioni, la proporzione delle classi può essere gestita tramite il tuning degli iperparametri. In tutti i casi queste tecniche non hanno portato ad un miglioramento delle prestazioni del modello, pertanto sono state mantenute le già citate SMOTE e RFECV, con gli stessi parametri. Tra le differenze con il Modello 1 troviamo l'eliminazione della feature *payment_method*, suggerita da un collaboratore, che ha portato ad un lieve incremento delle performance. Oltre a questo, la maggior parte dei cambiamenti si riflettono sulla fase di addestramento, in quanto sono stati impiegati due nuovi algoritmi: AdaBoost e Multi-Layer Perceptron. Sebbene l'implementazione della rete neurale artificiale sia poi stata abbandonata a causa degli eccessivi fattori critici, non irrilevanti sono stati i miglioramenti portati da AdaBoost. Potendo sfruttare un nuovo hardware leggermente più performante, combinato con un minore consumo di risorse dato da AdaBoost rispetto a Random Forest, la dimensione del dataset di training utilizzata in questo sviluppo è stata di 500 mila record. Un altro vantaggio di AdaBoo-

st, come mostrato da Codice 8, è un più semplice tuning degli iperparametri.

Codice 8 Implementazione di AdaBoost per l'addestramento.

```
[ ]: ada_model = AdaBoostClassifier(random_state=1,
                                   n_estimators=300,
                                   learning_rate=0.5,
                                   algorithm='SAMME')

     ada_model.fit(X_train_resampled, y_train_resampled)
     predict_y = ada_model.predict(X_test)
```

La scelta degli iperparametri, anche in questo caso, è stata guidata dall'intenzione di ridurre il rischio di overfitting. Per ottenere questo risultato è fondamentale trovare un giusto bilanciamento tra il numero di stimatori e l'impatto di ciascuno di essi. A differenza di Random Forest, questo algoritmo risulta meno robusto al sovradattamento, pertanto il valore di $learning\_rate = 0.5$ è stato impostato per garantire un apprendimento più lento e stabile. Inoltre il parametro *algorithm='SAMME'* rappresenta un valore standard per problemi di classificazione, in quanto può gestire sia problemi di classificazione binaria che multi-classe. A differenza della controparte *SAMME.R* che associa un valore di probabilità ad ogni risposta del modello, l'algoritmo SAMME si basa invece su previsioni di tipo discreto. Come per il Modello 1, terminato l'addestramento sono seguite le fasi di test preliminare e valutazione finale del modello.

5.6 Modello 3

A differenza dei precedenti due, con il Modello 3, si è preferito favorire la precisione anzichè la sensibilità. In questo contesto, tranne che in un caso particolare, non sono state impiegate nuove tecniche, ma piuttosto sono stati adattati i parametri di quelle già impiegate, per supportare la nuova strategia. Tra le modifiche apportare al dataset rispetto al Modello 2 vi è stata la reintegrazione della feature *payment_method*, e la variazione della dimensione del subset di training ove necessario. Durante lo sviluppo sono stati eseguiti addestramenti utilizzando diversi subset, per avere un quadro più chiaro del cambiamento delle performan-

ce al variare delle dimensioni del dataset, in particolare si spaziava tra 50 mila, 100 mila e 500 mila record. Tra le tecniche che hanno subito delle modifiche nei parametri troviamo SMOTE. In questo contesto sono stati provati diversi valori per *sampling_stragety*, che oscillano tra 0.1 e 0.9. Questa variazione è stata fondamentale per ottenere il valore di precisione migliore tra tutti i possibili. Inoltre un altra modifica è stata apportata a $k\_neighbors = 10$, aumentando il numero di vicini da considerare nel campionamento, migliorando la qualità dei nuovi dati sintetici. Questo sviluppo ha coinvolto l'utilizzo di un ulteriore nuovo hardware più performante, fattore che ha dato la possibilità di implementare una tecnica per il tuning automatico degli iperparametri. *RanomizedSearchCV* è una tecnica di ricerca di iperparametri che effettua un numero di addestramenti (*n_iter*) nei quali seleziona una serie casuale di valori da utilizzare, partendo da una griglia (*param_distribution*) definita dall'utente. Al termine di tutte le iterazioni restituisce i parametri migliori sulla base di un punteggio dato dalla metrica scelta (*scoring*). RSCV non garantisce l'identificazione dei parametri migliori in senso assoluto, bensì individua la configurazione più efficace tra le combinazioni esaminate durante le iterazioni. Un approccio potenzialmente più esaustivo è rappresentato da *GridSearchCV*, il quale, a differenza di RandomizedSearchCV, non è vincolato da un numero prestabilito di iterazioni. GSCV, infatti, valuta sistematicamente tutte le possibili combinazioni di parametri all'interno della griglia, restituendo la configurazione globalmente ottimale. Il principale svantaggio di GridSearchCV risiede nell'elevato numero di risorse computazionali richiesto per la sua implementazione. Questo fattore, nonostante i progressi nell'hardware disponibile, ne ha impedito l'utilizzo nel contesto attuale, favorendo l'adozione di una tecnica computazionalmente meno onerosa come RandomizedSearchCV.

Codice 9 Implementazione di RandomizedSearchCV per la ricerca dei parametri.

```
[ ]: param_dist = {'n_estimators': randint(100, 1000),
                  'max_features': ['log2', 'sqrt'],
                  'max_depth': [None, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
                  'min_samples_split': randint(2, 20),
                  'min_samples_leaf': randint(1, 20),
                  'bootstrap': [True, False],
```

```
                'criterion': ['gini', 'entropy'],
                'max_leaf_nodes': randint(2, 200)
}

model_rf = RandomForestClassifier(random_state=42)

random_search = RandomizedSearchCV(model_rf,
                                   param_distributions=param_dist,
                                   n_iter=50,
                                   cv=5,
                                   scoring='roc_auc',
                                   n_jobs=-1)

random_search.fit(X_train_resampled, y_train_resampled)

# Make predictions
prediction_test = random_search.best_estimator_.predict(X_test)
print(metrics.accuracy_score(y_test, prediction_test))
```

```
0.97944
```

[ ]:
```
best_params = random_search.best_params_
print("Migliori parametri:", random_search.best_params_)
```

```
Migliori parametri: {'bootstrap': False, 'criterion': 'entropy', 'max_depth':
100, 'max_features': 'sqrt', 'max_leaf_nodes': 168, 'min_samples_leaf': 2,
'min_samples_split': 4, 'n_estimators': 661}
```

Come mostrato dal Codice 9, l'esecuzione di RSCV restituisce i migliori parametri per l'algoritmo impiegato. Il problema più grande dell'algoritmo Random Forest riguardava la sua complessità computazionale e la difficoltà nel tuning degli iperparametri. Entrambe queste criticità sono state in parte risolte dalle migliori capacità di calcolo e dall'impiego di RSCV, questi fattori hanno portato ad una nuova implementazione dell'algoritmo nello sviluppo del Modello 3. Come per i due precedenti, al termine della fase di addestramento sono seguite le due fasi di test e di valutazione finale del modello, i cui risultati saranno analizzati nel prossimo capitolo.

Capitolo 6

Analisi dei risultati

Ora che sono stati descritti il processo di sviluppo e le caratteristiche di ciascun modello, possiamo presentare e discutere i risultati ottenuti nell'analisi predittiva del churn. In questo capitolo verranno valutate le performance dei tre modelli con l'ausilio delle metriche più pertinenti, inoltre saranno esaminati anche gli aspetti di applicabilità di ciascuno di essi. E' importante ribadire che in questa parte non verranno affrontati i risultati del Modello 0, seppur ottimi, poichè non attendibili. Per comodità le performance di ciascun modello saranno divise in due sottosezioni: la prima per illustrare gli esiti durante la fase di test, mentre la seconda dedicata alle valutazioni finali. Inoltre ogni sezione sarà arricchita con un iniziale breve riepilogo sulle tecniche utilizzate e le considerazioni sui risultati.

6.1 Valutazione Modello 1

Si presenta di seguito una sintetica rassegna delle caratteristiche del Modello 1, organizzate per categoria:

1. **Obiettivo:** la strategia adottata nella progettazione è stata massimizzare il recall, in modo da identificare il maggior numero possibile di clienti churn.

2. **Dataset:** la dimensione del subset scelta per lo sviluppo è stata di 50 mila record.
3. **Selezione delle feature:** la tecnica di feature selection adottata in questa fase è stata RFECV, numero ottimale di feature 15.
4. **Ricampionamento:** per l'oversampling è stato impiegato l'algoritmo SMOTE, bilanciando il dataset con una proporzione 1:1.
5. **Algoritmo predittivo:** il modello finale è stato addestrato utilizzando Random Forest come classificatore.

6.1.1 Risultati su campione ridotto

I risultati presentati in questa parte sono relativi alla fase preliminare di test, pertanto le previsioni sono state eseguite su uno split di dimensione pari al 20% del subset di training. La figura 6.1 mostra l'andamento delle predizioni fatte dal Modello 1, sul campione ridotto di dati.

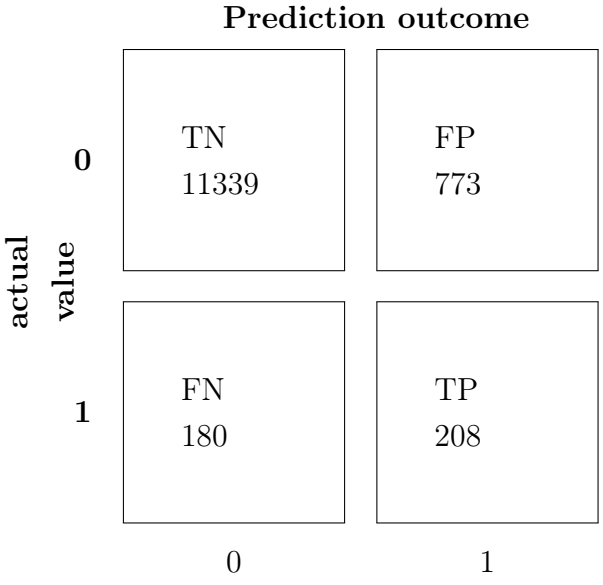

Figura 6.1: Matrice di confusione del modello 1 in fase di test

Il seguente Codice 10 ci fornice una panoramica delle performance iniziali del modello tramite l'utilizzo di metriche come precisione, sensibili-

tà e F-measure. Sebbene sia presente anche il valore di accuracy riscontrato, data la natura del nostro problema non può essere considerato come una misura attendibile, sarà pertanto escluso dalla nostra analisi.

Codice 10 Misure di performance di Modello 1 in fase di test.

```
[ ]: print(classification_report(y_test, prediction_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96     12112
           1       0.21      0.54      0.30       388

    accuracy                           0.92     12500
   macro avg       0.60      0.74      0.63     12500
weighted avg       0.96      0.92      0.94     12500
```

Per quanto riguarda la metrica AUC_ROC, la figura 6.2 mostra l'andamento della curva all'intero di un grafico cartesiano. La linea rossa è indicativa del *trade off* tra TPR e FPR, e quindi della capacità discriminativa del modello. Durante lo sviluppo dei modelli non sono stati ricavati i valori delle AUC, tuttavia le prestazioni rimangono comunque intuibili dall'andamento dalle curve nei grafici.

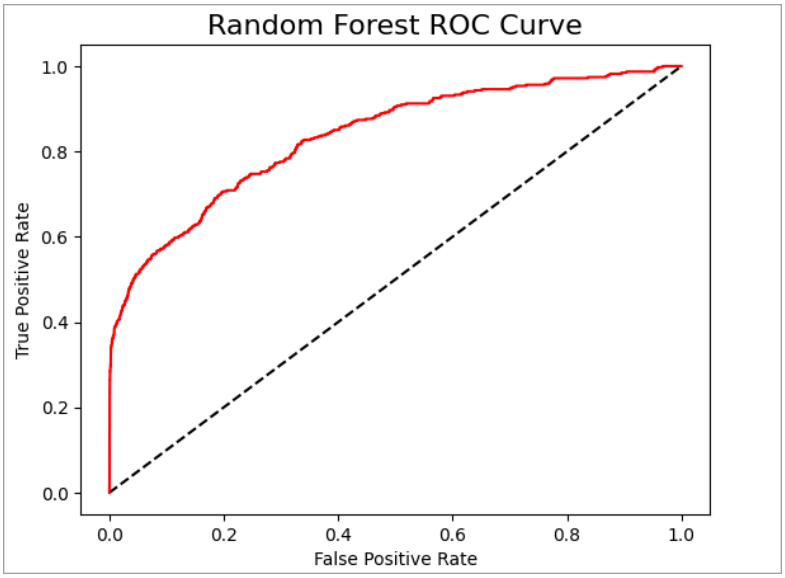

Figura 6.2: Rappresentazione grafica curva ROC

In questa fase la curva si trova al di sopra della diagonale tratteggiata, avvicinandosi all'angolo in alto a sinistra, suggerendo una discreta capacità del modello di distinguere tra le classi.

6.1.2 Esiti finali

Questa sottosezione è dedicata all'analisi dei risultati ottenuti nell'ultima fase di valutazione. Il dataset utilizzato per le previsioni del modello contiene i dati dei clienti relativi al mese di novembre 2023, per un totale di circa 3 milioni di record. La matrice di confusione in figura 6.3 ci mostra come il modello ha eseguito le classificazioni.

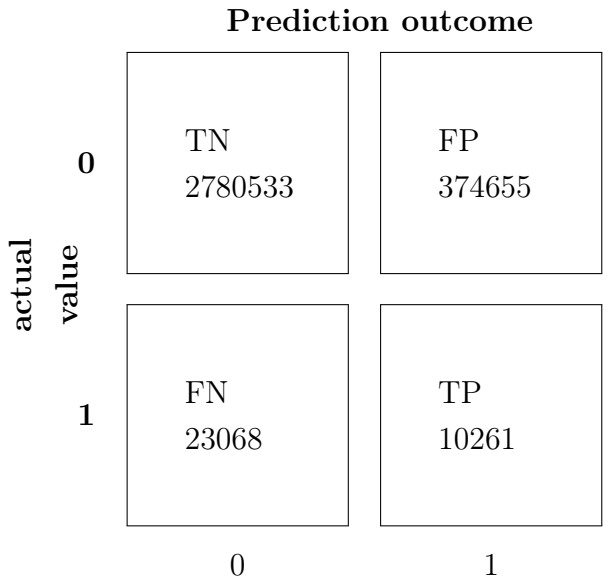

Figura 6.3: Matrice di confusione del modello 1 in fase di valutazione

Il report delle performance finali del Modello 1 suddiviso per classe e metrica, è mostrato dal Codice 11.

Codice 11 Misure di performance di Modello 1 in fase di valutazione.

```
[ ]: print(classification_report(y_prediction, prediction_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.88      0.93   3155188
           1       0.03      0.31      0.05     33329

    accuracy                           0.88   3188517
   macro avg       0.51      0.59      0.49   3188517
weighted avg       0.98      0.88      0.92   3188517
```

Inoltre come per la fase di test, la figura 6.4 è rappresentativa dell'andamento della curva ROC all'interno del piano. A differenza del grafico precedente, vediamo come

la curva si avvicini molto di più alla diagonale, suggerendo delle scarse performance nel distinguere le classi.

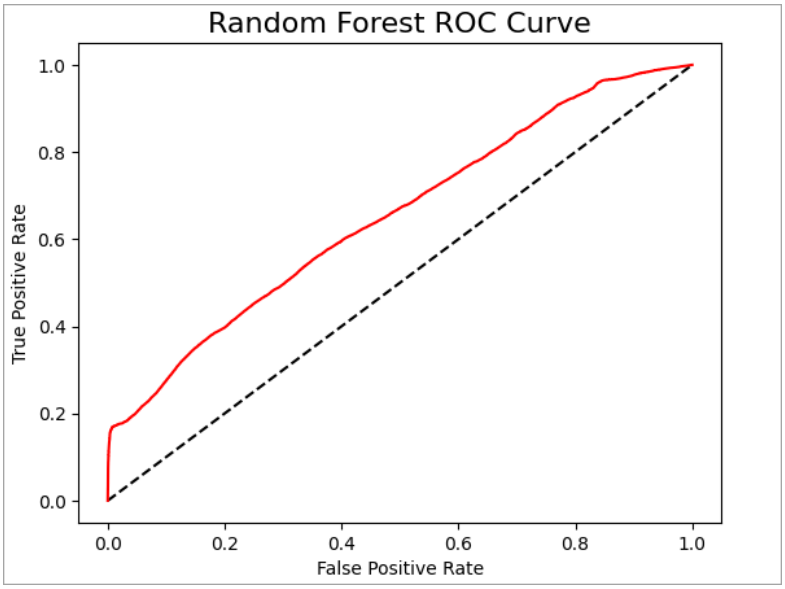

Figura 6.4: Rappresentazione grafica curva ROC, in fase di valutazione

L'immagine in figura 6.5 mostra un report finale delle prestazioni del Modello 1. I risultati sono stati calcolati considerando le classificazioni relative al mese di novembre 23, confrontate con i positivi dei tre mesi successivi (dicembre 23, gennaio 24, febbraio 24, marzo24)

Figura 6.5: Report di performance del Modello 1

6.1.3 Considerazioni - Modello 1

Dai grafici e dai numeri forniti dalle varie metriche possiamo notare come le prestazioni del Modello 1 siano piuttosto basse, sopratutto in termini di precisione. Questo trend si poteva intuire già dai primi risultati ottenuti durante la fase di test, come conseguenza degli scarsi valori dei KPIs e dalle classificazioni mostrate dalla matrice di confusione. Come era prevedibile inoltre con l'aumentare della quantità di dati, è aumentata anche la probabilità del modello di commettere errori, situazione che ha portato ad un ulteriore abbassamento delle performance. Per quanto riguarda il confronto con i mesi successivi, si può notare un lieve incremento degli indicatori, in termini di sensibilità e di precisione. Questo incremento è dovuto al fatto che durante il confronto, parte dei falsi positivi identificati si sono trasformati in veri positivi, riducendo gli errori commessi dal modello. E' interessante notare anche, come nel mese immediatamente successivo alla prediction (dicembre 23), i true positive aumentano rispetto a quelli predetti nel mese di novembre 23. Un valore di precisione così basso è la causa dell'elevato numero di falsi positivi generati dal modello, che si traduce in una grande quantità di errori commessi durante la classificazione. Anche il punteggio ottenuto dalla metrica di sensibilità

è critico, in quanto il modello è riuscito ad individuare solo poco più di un terzo dei clienti che hanno abbandonato, lasciando i restanti due terzi come non rilevati. In conclusione, nonostante l'obiettivo di questo modello fosse massimizzare l'identificazione dei clienti churn, i risultati ottenuti si sono rivelati non sufficienti, richiedendo degli sforzi significativi per migliorarne le performance, e una revisione completa delle tecniche utilizzate. Tra le probabili cause dell'ottenimento di questi risultati troviamo sicuramente la limitata dimensione del subset di addestramento, imposta dalla capacità di calcolo disponibile in questa fase. Aumentare la quantità di dati impiegata durante il training è utile per migliorare la capacità del modello di generalizzare, in quanto una dimensione non sufficientemente grande potrebbe limitare la variabilità e la rappresentatività dei casi di churn.

6.2 Valutazione Modello 2

Con i risultati ottenuti da Modello 1 è risultata evidente la necessità di svolgere una nuova fase di analisi, provando nuove tecniche e concentrandosi maggiormente sul tuning degli iperparametri. Nonostante ciò, terminato questo studio e fatte sufficienti prove, molte delle nuove tecniche impiegate, sopratutto nella manipolazione dei dati, non hanno avuto un impatto significativo, pertanto la maggior parte degli algoritmi impiegati è rimasta invariata. Vediamo ora un breve recap delle caratteristiche del Modello 2:

1. **Obiettivo:** ancora una volta la strategia è rimasta massimizzare il recall.
2. **Dataset:** la dimensione del subset utilizzato in questa fase è di 500 mila record.
3. **Selezione delle feature:** come per il modello precedente, la tecnica di feature selection è rimasta RFECV, numero di feature ottimali 6.
4. **Ricampionamento:** anche in questa fase è stato mantenuto l'algoritmo SMOTE, sono state provate più proporzioni per il bilanciamento ma quella ideale è rimasta 1:1.
5. **Algoritmo predittivo:** per questo modello l'algoritmo impiegato nell'addestramento è stato AdaBoost.

6.2.1 Risultati su campione ridotto

I risultati mostrati dalla figura 6.6, rappresentano le classificazioni effettuate dal Modello 2 durante la fase preliminare di test, sul campione ridotto dei dati.

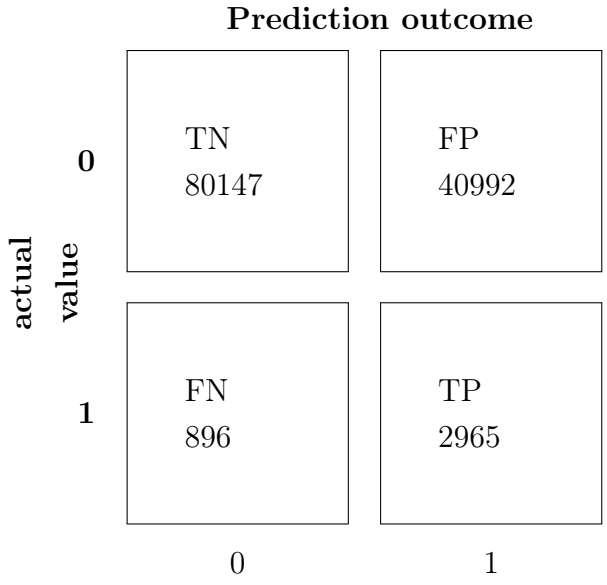

Figura 6.6: Matrice di confusione del modello 2 in fase di test

Per quanto riguarda la panoramica sulle performance del modello, si faccia riferimento al Codice 12 mostrato di seguito.

Codice 12 Misure di performance di Modello 2 in fase di test.

```
[ ]: print(classification_report(y_test, prediction_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.66      0.79    121139
           1       0.07      0.77      0.12      3861

    accuracy                           0.66    125000
   macro avg       0.53      0.71      0.46    125000
weighted avg       0.96      0.66      0.77    125000
```

La rappresentazione grafica dell'andamento della curva ROC invece, può essere osservato dall'immagine in figura 6.7

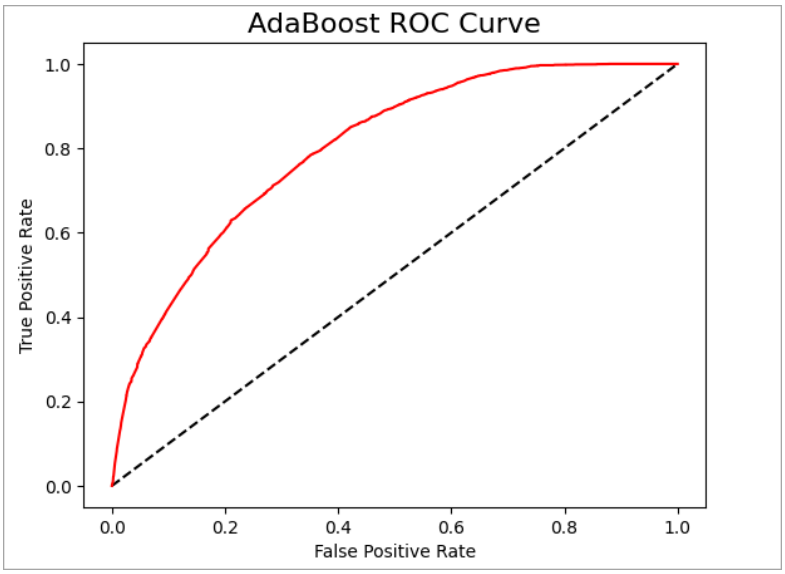

Figura 6.7: Rappresentazione grafica curva ROC, in fase di test

6.2.2 Esiti finali

Passiamo ora all'analisi dei risultati di Modello 2 ottenuti durante la fase di validazione. Il dataset utilizzato è il medesimo impiegato per la valutazione di Modello 1, quindi l'arco temporale dei dati fa riferimento al mese di novembre 2023 e la dimensione è di circa 3 milioni di record. La figura 6.8 mostra le classificazioni svolte dal modello durante la fase di valutazione, attraverso l'utilizzo di una matrice di confusione.

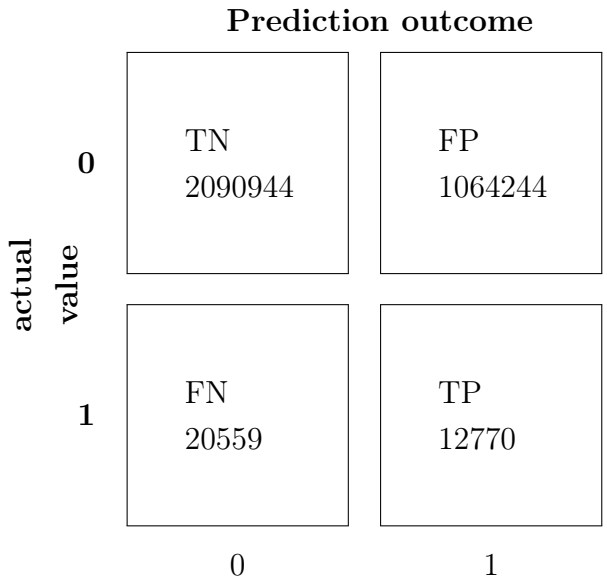

Figura 6.8: Matrice di confusione del modello 2 in fase di valutazione

Gli esiti ottenuti dalle classificazioni svolte si riflettono nei valori di performance mostrati dal Codice 13.

Codice 13 Misure di performance di Modello 2 in fase di valutazione.

```
[ ]: print(classification_report(y_prediction, prediction_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.66      0.79   3155188
           1       0.01      0.38      0.02     33329

    accuracy                           0.66   3188517
   macro avg       0.50      0.52      0.41   3188517
weighted avg       0.98      0.66      0.79   3188517
```

L'immagine in figura 6.9 mostra invece la rappresentazione della curva ROC all'interno del grafico cartesiano.

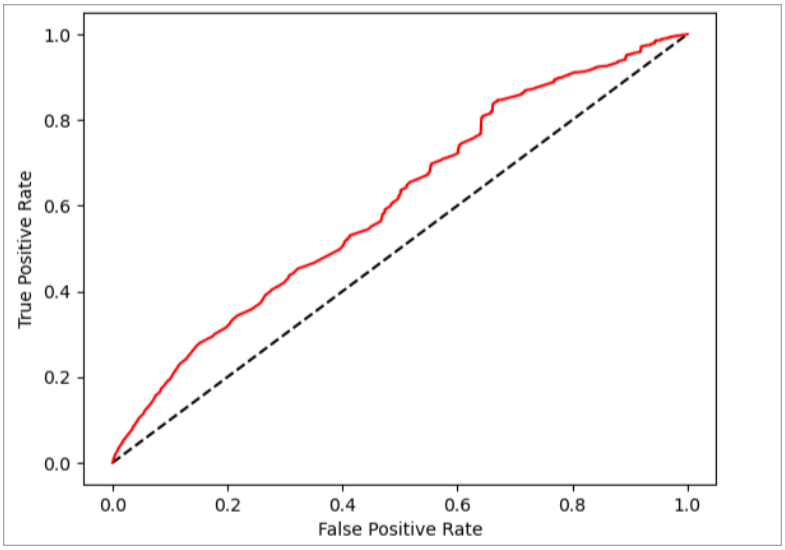

Figura 6.9: Rappresentazione grafica curva ROC, in fase di valutazione

In ultima istanza, per concludere la panoramica relativa al Modello 2, viene fornito il report generale sulle performance e sulle caratteristiche mostrato in figura 6.10

Model Training – Scikit-Learn 2

*Metrics over the 4 months after training

5,2% Precision **65,8%** Accuracy **46,5%** Recall

Metrics	Month 1	Month 2	Month 3	Month 4	Tot
True Positives	12770 (38,31%)	15857 (48,85%)	13954 (50,68%)	14217 (49,19%)	56798 (46,47%)
False Negatives	20559 (61,69%)	16603 (51,15%)	13578 (49,32%)	14684 (50,81%)	65424 (53,53%)
False Positives	1,02M				
True Negative	2,04M				

Complete model results

Training Datasets Preparation

- Filled the null data and encoding
- Dataset rebalanced, changed churned/not churned ratio to 50/50
- Dimensionality reduction and Feature selection
- used 4 month of data and aggregated information to be compatible with training
- training records 500K

Training

- Tool used Studio with JupyterLab
- Training Time 35**m**
- Training Cost
- Optimization metric **AUC-ROC**

Conclusions

- Prediction done on month 1 (month after training) and the table shows churn rate in the 4 months after the training ones

doxee

Figura 6.10: Report di performance del Modello 2

6.2.3 Considerazioni - Modello 2

Anche il secondo modello purtroppo ha disatteso le aspettative, restituendo dei risultati piuttosto scarsi, in linea con quelli ottenuti da Modello 1. La nuova fase di ottimizzazione e le nuove tecniche impiegate si sono rivelate non sufficienti al fine di ottenere un miglioramento delle performance. Nonostante ciò questi sforzi hanno portato ad un lieve incremento della sensibilità del modello, preannunciato inizialmente dai risultati ottenuti in fase di test. Al termine dell'addestramento, sulle previsioni svolte nel quadrimestre successivo ai dati di training, il valore di recall si attesta attorno ad un debole 50%, suggerendo che il modello riesce a classificare correttamente circa la metà dei clienti che hanno abbandonato. Anche per il Modello 2 si può notare un lieve incremento delle performance nei mesi di dicembre 23 e gennaio 24, con un picco nel terzo mese. Un trend che si può osservare in entrambi i modelli è che l'aumentare della sensibilità si riflette in una diminuzione della precisione. Questo fenomeno è dovuto al fatto che un valore di recall alto significa che il modello è più propenso a classificare un esempio come positivo, di conseguenza oltre ad aumentare il numero di veri positivi, aumenta anche quello dei falsi positivi. Poichè la precisione tiene conto dei falsi positivi, un loro aumento si traduce in una diminuzione della precisione. Anche i risultati mostrati dalle curve ROC in entrambe le fasi di test e valutazione, sono in linea con quelle del Modello 1, mostrando che il modello fatica a distinguere le due classi, commettendo molti errori. E' importante sottolineare come nello sviluppo del Modello 2 si sia posta l'attenzione anche sull'adozione di nuove metriche di ottimizzazione, in particolare F1-score e precisione, tuttavia senza ottenere dei miglioramenti in termini di prestazioni. Le misure di performance e la grande quantità di falsi positivi generati da Modello 2 potrebbero essere indicatori che il modello soffre di overfitting. Sebbene questi risultati suggeriscano che il modello non generalizzi bene su nuovi dati, per avere la conferma che si tratta di sovradattamento sarebbe utile confrontarli con le performance sui dati di addestramento. In assenza di questo confronto non ne possiamo avere la certezza, ma risulta comunque evidente la difficoltà del modello nel classificare correttamente le osservazioni. Questi risultati ci hanno interrogato sul fatto se proseguire con questa strategia fosse veramente la direzione corretta da seguire. Pertanto nello sviluppo del Modello 3 si è voluto seguire una strada differente, il che ha richiesto una nuova fase di studio e analisi del problema.

6.3 Valutazione Modello 3

I risultati ottenuti dai precedenti due modelli, hanno spinto lo sviluppo verso una nuova strategia. Nella creazione del Modello 3 si è voluto ridurre la quantità di errori commessi dal classificatore, anche a costo di identificare un minor numero di clienti churn, in altre parole si è favorita la precisione. Da un punto di vista aziendale avere un modello più preciso potrebbe essere preferibile da un punto di vista di risorse impiegate. Se è vero che predire una quantità maggiore di clienti a rischio di churn aumenti la capacità dell'azienda di intervenire, è altrettanto vero che un modello più preciso possa favorire degli interventi più mirati sui singoli clienti. Questi sono stati alcuni dei fattori che hanno portato allo sviluppo del Modello 3, di cui di seguito è presente un breve recap:

1. **Obiettivo:** aumentare la precisione del modello.
2. **Dataset:** nel corso dello sviluppo sono state testate dimensioni differenti, precisamente da 50 mila, 100 mila e 500 mila record.
3. **Selezione delle feature:** ancora una volta RFECV si riconferma la tecnca di feature selection più efficace, numero di feature ottimali 16.
4. **Ricampionamento:** nuovamente l'algoritmo impiegato è stato SMOTE, e l'oversampling è stato misurato attraverso più sampling_strategy, la migliore è stata 0.1.
5. **Tuning automatico:** il nuovo hardware ha permesso l'implementazione di RandomizedSearchCV.
6. **Algoritmo predittivo:** nel terzo modello è stato nuovamente utilizzato Random Forest, scelta motivata dall'impiego di una tecnica di tuning automatico degli iperparametri.

6.3.1 Panoramica risultati parziali

Prima di descrivere i risultati finali in fase di test e in fase di valutazione e con l'utilizzo di quali parametri siano stati ottenuti, forniamo una breve panoramica delle performance ottenute dal Modello 3 su diverse dimensioni del training set

e con diverse strategie di campionamento. I risultati descritti di seguito fanno riferimento alle prediction svolte nel mese di novembre 23, composto da 3.188.517 record, per comodità i true negative non sono riportati nelle tabelle, ma rimangono comunque facilmente ricavabili dal totale qualora servissero.

Training set 50 mila record

La tabella 6.1 mostra i risultati della prediction senza l'utilizzo della tecnica SMOTE e con l'impiego di un subset di training di 50 mila record.

Precision	Recall	F1-score	TP	FP	FN
86%	14%	24%	4718	789	28611

Tabella 6.1: Risultati ottenuti senza SMOTE

La successiva tabella 6.2 fa riferimento alle classificazioni ottenute con l'utilizzo di una strategia strategia di campionamento pari a 0.1 e con lo stesso subset.

Precision	Recall	F1-score	TP	FP	FN
69%	16%	26%	5291	2353	28038

Tabella 6.2: Risultati ottenuti con sampling_strategy=0.1

La tabella 6.3 mostra invece i risultati ottenuti dalle classificazioni utilizzando *sampling_strategy=0.2*, mantenendo lo stesso subset.

Precision	Recall	F1-score	TP	FP	FN
25%	17%	20%	5578	16321	27751

Tabella 6.3: Risultati ottenuti con sampling_strategy=0.2

Da questo momento si comincia a notare un trend, ovvero che all'aumentare della proporzione della classe minoritaria, la precisione diminuisce ed aumenta il

recall. Questo fenomeno si può osservare anche dai risultati descritti dalla tabella 6.4 che utilizza *sampling_strategy=0.3*.

Precision	Recall	F1-score	TP	FP	FN
14%	17%	15%	5795	36276	27534

Tabella 6.4: Risultati ottenuti con sampling_strategy=0.3

Proseguendo con i successivi risultati, la tabella 6.5 mostra le classificazioni ottenute con l'impiego di *sampling_strategy=0.5*.

Precision	Recall	F1-score	TP	FP	FN
4%	18%	7%	5937	134946	27392

Tabella 6.5: Risultati ottenuti con sampling_strategy=0.5

Avvicinandosi sempre di più alla proporzione 1:1 tra le classi, la tabella 6.6 descrive le osservazioni ottenute dal modello utilizzando *sampling_strategy=0.7*.

Precision	Recall	F1-score	TP	FP	FN
3%	20%	5%	6760	209824	26569

Tabella 6.6: Risultati ottenuti con sampling_strategy=0.7

Infine presentiamo con la tabella 6.7, i risultati ottenuti dal modello con l'impiego di *sampling_strategy=0.9*.

Precision	Recall	F1-score	TP	FP	FN
3%	24%	5%	8118	281260	25211

Tabella 6.7: Risultati ottenuti con sampling_strategy=0.9

Training set 100 mila record

In una fase successiva sono stati analizzati i risultati delle predizioni del modello addestrato su 100 mila record. Dato che nell'analisi precedente il miglior compromesso nelle performance era stato offerto dall'utilizzo di SMOTE con *sampling_strategy=0.1*, questo valore è stato mantenuto anche in questo training. Pertanto la tabella 6.8 mostra i risultati ottenuti dal Modello 3 con questa configurazione.

Precision	Recall	F1-score	TP	FP	FN
47%	16%	24%	5290	5938	28039

Tabella 6.8: Risultati ottenuti con sampling_strategy=0.1

Training set 500 mila record

In ultima istanza analizziamo i risultati ottenuti dal modello addestrato su 500 mila record, mantenendo gli stessi parametri di campionamento della precedente versione. La tabella 6.9 descrive il comportamento di Modello 3 con le caratteristiche precedentemente elencate.

Precision	Recall	F1-score	TP	FP	FN
41%	16%	23%	5258	7553	28039

Tabella 6.9: Risultati ottenuti con sampling_strategy=0.1

Da questi risultati possiamo trarre diverse conclusioni. La precisione del modello cala in maniera direttamente proporzionale all'aumentare della proporzione della classe minoritaria nel ricampionamento. In altre parole più la quantità dei dati della classe di minoranza si avvicina alla quantità di quelli della classe di maggioranza e più il modello tende a fare errori, non distinguendo bene tra le due classi. A conferma di ciò troviamo il picco nel punteggio di precisione del modello addestrato senza l'utilizzo di SMOTE, i cui risultati sono mostrati dalla tabella 6.1. Al contrario invece la sensibilità aumenta in questo contesto, seppur molto

lentamente. Complessivamente, anche la media armonica rappresentata dalla metrica F1 diminuisce. Complessivamente impiegare una quantità di dati maggiore durante l'addestramento non si è riflettuto necessariamente in un aumento delle performance. Al termine di questa analisi parziale, si è pensato di utilizzare la configurazione con *sampling_strategy=0.1* e training set di 50 mila record, essendo la versione che nonostante non abbia la precisione più alta ha il valore di F1-score tra tutti migliore.

6.3.2 Risultati su campione ridotto

Riprendiamo ora la tradizionale panoramica sulle performance del modello tramite gli strumenti precedentemente utilizzati. La figura 6.11 mostra le classificazioni del modello in fase di test rappresentate da una matrice di confusione.

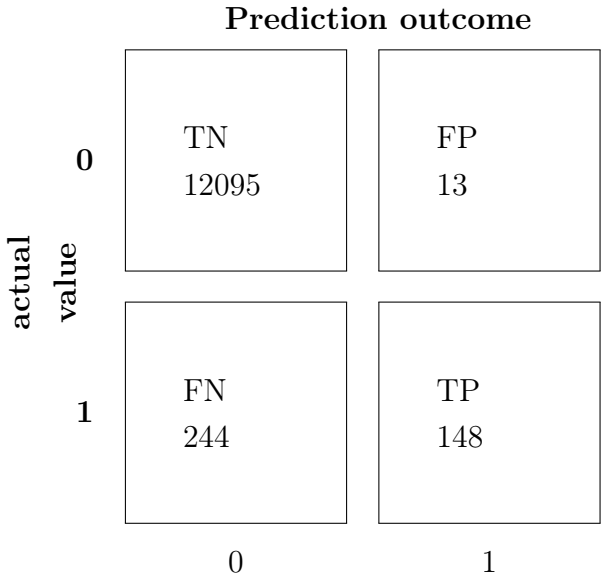

Figura 6.11: Matrice di confusione del modello 3 in fase di test

Nel Codice 14 invece sono descritte le performance ottenute dal modello durante la fase di test.

Codice 14 Misure di performance di Modello 3 in fase di test.

```
[ ]: print(classification_report(y_test, prediction_test))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     12108
           1       0.92      0.38      0.54       392

    accuracy                           0.98     12500
   macro avg       0.95      0.69      0.76     12500
weighted avg       0.98      0.98      0.98     12500
```

Per concludere la panoramica delle dei risultati ottenuti in fase di test, in figura 6.12 è mostrato l'andamento della curva ROC all'interno del piano cartesiano.

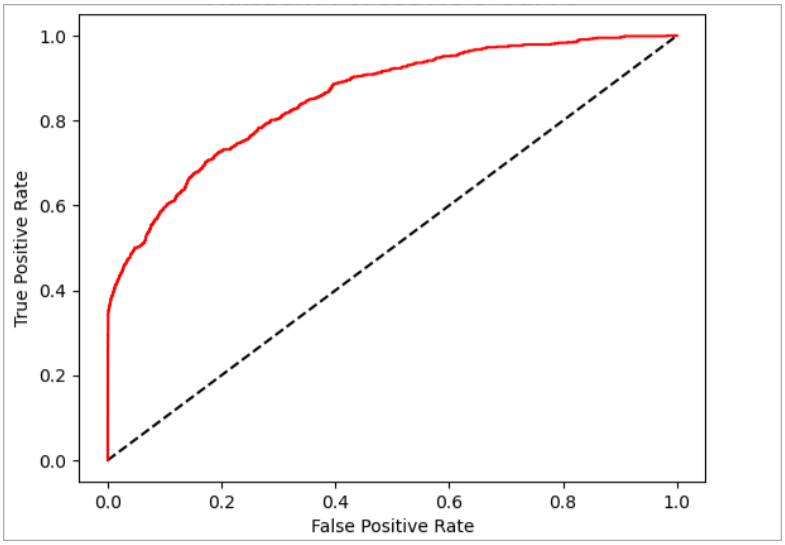

Figura 6.12: Rappresentazione grafica curva ROC, in fase di test

6.3.3 Esiti finali

Infine analizziamo i risultati finali di Modello 3, che abbiamo già parzialmente visto precedentemente, con l'ausilio degli appositi strumenti. In figura 6.13 è possibile osservare le classificazioni svolte dal modello sul dataset di valutazione.

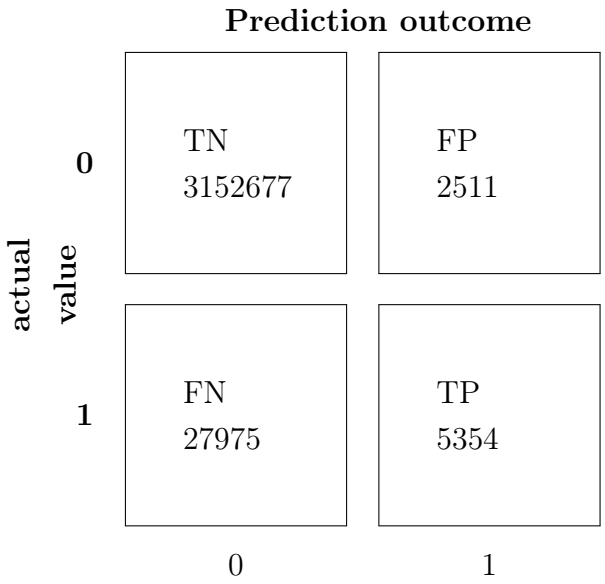

Figura 6.13: Matrice di confusione del modello 3 in fase di valutazione

Di seguito invece, il Codice 15 mostra il recap delle performance, già precedentemente descritte del modello in fase di valutazione.

Codice 15 Misure di performance di Modello 3 in fase di valutazione.

```
[ ]: print(classification_report(y_prediction, prediction_test))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00   3155188
           1       0.68      0.16      0.26     33329

    accuracy                           0.99   3188517
   macro avg       0.84      0.58      0.63   3188517
weighted avg       0.99      0.99      0.99   3188517
```

In ultima analisi invece presentiamo la misura della curva ROC, rappresentata dalla figura 6.14.

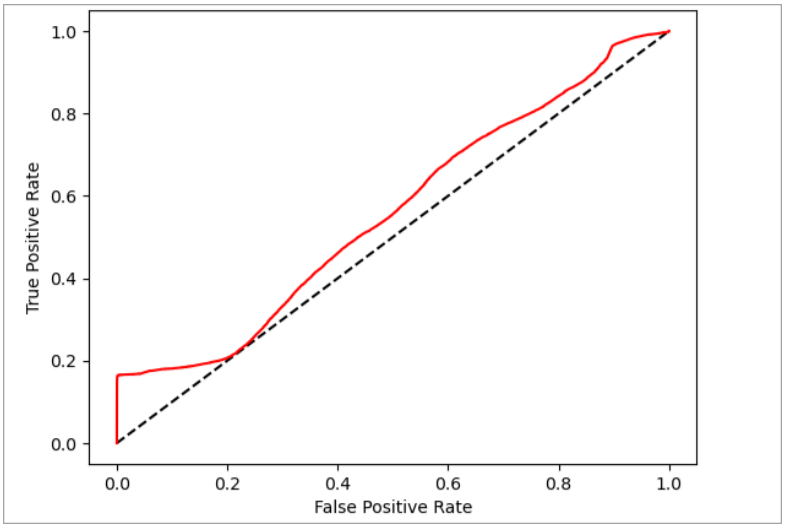

Figura 6.14: Rappresentazione grafica curva ROC, in fase di valutazione

Come per i precedenti modelli, concludiamo la panoramica sulle performance di Modello 3 fornendo un recap, in figura 6.15, dei risultati ottenuti anche nella previsione dei clienti churn nei mesi successivi a quello di valutazione (novembre 23).

Model Training – Scikit-Learn 3

*Metrics over the 4 months after training

71,2% Precision **96,2%** Accuracy **4,5%** Recall

Metrics	Month 1	Month 2	Month 3	Month 4	Tot
True Positives	5354 (16,06%)	119 (0,37%)	54 (0,20%)	77 (0,27%)	5604 (4,59%)
False Negatives	27975 (83,94%)	32341 (99,63%)	27478 (99,80%)	28824 (99,73%)	116618 (95,41%)
False Positives	2261				
True Negative	3,06M				

Complete model results

Training Datasets Preparation

- Filled the null data and encoding
- Dataset rebalanced, changed churned/not churned ratio to 10/90
- Dimensionality reduction and Feature selection
- used 4 month of data and aggregated information to be compatible with training
- training records 50K

Training

- Tool used Studio with JupyterLab
- Training Time 8**m**
- Training Cost
- Optimization metric **AUC-ROC**

Conclusions

- Prediction done on month 1 (month after training) and the table shows churn rate in the 4 months after the training ones

doxee

Figura 6.15: Report di performance del Modello 3

6.3.4 Considerazioni - Modello 3

A differenza dei precedenti due modelli, con il terzo si è voluto seguire un obiettivo differente. Questa nuova strategia ha portato ad un incremento significativo della precisione, e quindi ad una notevole riduzione della quantità di errori commessi, in termini di falsi positivi. L'aumento della precision ha però comportato un drastico calo nel valore del recall, che si è tradotto in una minore quantità di positivi riconosciuti. L'alto numero di falsi negativi indica che la maggior parte dei clienti che abbandonano non vengono correttamente identificati dal modello. Per colpa di questo divario anche la misura della curva ROC risulta piuttosto scarsa, ed è indicativa della scarsa capacità del modello distinguere correttamente le due classi, nonostante l'alta precision. Si noti come nella fase di test del modello 3 si sia registrato il valore di F1 più alto tra gli altri modelli dalla risoluzione del bug. Come per i precedenti due modelli si misura una diminuzione delle performance tra i risultati ottenuti in fase di test e quelli in fase di valutazione, per via dell'importante incremento della quantità di dati. Come emerso dai risultati parziali, aumentare la dimensione del dataset di training non ha significato un aumento anche delle performance, neppure in termini di recall. Inoltre è importante notare come le prestazioni sul singolo mese non si riflettano anche sui successivi, probabilmente anche a causa del ridotto numero di falsi positivi generati.

6.4 Considerazioni finali

Lo sviluppo di ciascuno dei tre modelli, ha purtroppo restituito dei risultati piuttosto scarsi. Tuttavia l'analisi critica e le fasi di studio sono sicuramente state fondamentali per guidare il lieve miglioramento e le strategie impiegate nella creazione. E' importante notare come ciascun modello abbia caratteristiche sostanzialmente differenti, nonostante le simili tecniche impiegate negli sviluppi. In particolare il secondo nasce dalla necessità di ottimizzare le prestazioni del primo, questo lo ha reso il modello in grado di identificare il maggior numero di clienti churn dei tre, anche nel confronto del confronto con i mesi successivi alle prediction. D'altro canto il Modello 3 ha mostrato il miglior valore di precisione, il che lo definisce come quello più affidabile e che commette meno errori, quantomeno in termini di falsi positivi. L'ideale sarebbe stato trovare un giusto compromesso tra precisione

e recall, (F1) ma gli sforzi volti a questo obiettivo nel corso dello sviluppo non hanno dato esiti positivi, costringendoci a preferire una delle due metriche.

Capitolo 7

Conclusioni

Il presente lavoro di tesi ha esplorato l'applicazione di tecniche di machine learning per l'analisi predittiva del churn nel settore utility. L'obiettivo principale è stato quello studiare il problema e sviluppare un modello in grado di identificare i clienti a rischio di abbandono, per permettere all'azienda cliente di adottare strategie adeguate per la *customer retention*. Nonostante gli scarsi risultati ottenuti, l'analisi condotta ha comunque evidenziato come l'integrazione di queste tecniche sia fondamentale e possa fornire un valore aggiunto significativo per le decisioni aziendali strategiche. Durante il processo di sviluppo, infatti, sono stati creati tre modelli, ciascuno caratterizzato da specifiche caratteristiche e vantaggi distinti. I modelli sono stati sviluppati in maniera tale da affrontare il problema da prospettive diverse. Questa diversità di approcci ha permesso di esplorare differenti strategie, fornendo una visione più completa del fenomeno del churn e delle potenziali soluzioni.

Tra gli aspetti critici che hanno determinato le performance dei modelli, si evidenziano la qualità e la disponibilità dei dati. Nello specifico, le informazioni utilizzate per l'addestramento coprivano un periodo relativamente breve, tenendo conto della natura complessa del problema in questione, inoltre la ridotta varietà dei dati presenti potrebbe aver avuto un impatto importante nei risultati finali. È ragionevole pensare che un set di dati relativo ad un arco temporale più ampio

potrebbe avere un impatto positivo significativo sull'efficacia complessiva dei modelli. In maniera analoga l'integrazione di nuove feature come *tenure*, oppure un migliore tuning degli iperparametri tramite tecniche come GridSearch potrebbe tradursi in un incremento sostanziale delle prestazioni dei modelli.

In conclusione, questa esperienza, è stata per me estremamente arricchente sia da un punto di vista accademico che personale. Durante il percorso, ho avuto l'opportunità di approfondire le mie conoscenze in ambito di analisi dei dati e machine learning, settori che non avevo precedentemente esplorato nel mio percorso di studi, affrontando sfide tecniche che hanno contribuito alla mia crescita professionale. Ritengo che questo progetto mi abbia aiutato a migliorare le mie capacità di organizzazione e di risolvere problemi, poichè ho dovuto bilanciare la ricerca e la sperimentazione, con le esigenze pratiche del progetto. Il contesto aziendale, inoltre, mi ha offerto l'opportunità di collaborare con esperti del settore, permettendomi di apprendere nuove tecnologie e metodi di sviluppo, oltre a comprendere in modo più approfondito il problema del churn nel settore utility. Nel complesso, questa esperienza ha consolidato la mia passione per la materia e mi ha fornito gli strumenti per affrontare con fiducia nuove sfide accademiche e professionali.

Bibliografia

[1] Arera. Energia: sempre più famiglie e imprese passano al mercato libero. si riducono i contratti "push", sollecitati dai call center. *Arera*, Feb 2022. https://www.arera.it/comunicati-stampa/dettaglio/it/com-stampa/22/220203cs, Accessed Jul 2024.

[2] Arera. I numeri dei servizi pubblici. *Arera*, Jul 2024. https://www.arera.it/comunicati-stampa/dettaglio/arera-i-numeri-dei-servizi-pubblici, Accessed Jul 2024.

[3] Fred Reichheld. Prescription for cutting costs. *Bain & Company. Harvard Business School Publishing*, 2001.

[4] Federica Laricchia. Italy: Tim churn rate 2015-2023. *Statista*, Jun 2024. https://www.statista.com/statistics/1064039/churn-rate-telecom-italia-tim-italy/, Accessed Jul 2024.

[5] Ahmed Sherif. Mobile customer churn rate in vodafone europe 2023. *Statista*, Nov 2023. https://www.statista.com/statistics/972046/vodafone-churn-rate-european-countries/, Accessed Jul 2024.

[6] Tom Springer, Charles Kim, Domenico Azzarello, and Jeff Melton. Breaking the back of customer churn. *Bain & Company*, 2016.

[7] Doxee. Chi siamo, Nov 2022. https://www.doxee.com/it/chi-siamo/, Accessed Aug 2024.

[8] Atlassian. Jira: Software per il monitoraggio di ticket e progetti. https://www.atlassian.com/it/software/jira, Accessed Aug 2024.

[9] Atlassian. Su di noi. https://www.atlassian.com/it/company, Accessed Aug 2024.

[10] Atlassian. Confluence: Lo spazio di lavoro pratico per i team in remoto. https://www.atlassian.com/it/software/confluence, Accessed Aug 2024.

[11] Atlassian. Git solution for teams using jira. https://bitbucket.org/product/, Accessed Aug 2024.

[12] NumPy. About us. https://numpy.org/about/, Accessed Aug 2024.

[13] Pandas. About us. https://pandas.pydata.org/about/, Accessed Aug 2024.

[14] Scikit-Learn. Scikit-learn. https://scikit-learn.org/stable/, Accessed Aug 2024.

[15] Seaborn. Statistical data visualization. https://seaborn.pydata.org/index.html, Accessed Aug 2024.

[16] MatPlotLib. Matplotlib. https://matplotlib.org/, Accessed Aug 2024.

[17] AWS. What is aws. https://aws.amazon.com/what-is-aws/, Accessed Aug 2024.

[18] AWS. Aws s3. https://aws.amazon.com/s3/, Accessed Aug 2024.

[19] AWS. Aws glue. https://aws.amazon.com/glue/, Accessed Aug 2024.

[20] AWS. Aws athena. https://aws.amazon.com/athena/, Accessed Aug 2024.

[21] AWS. Free development environment for machine learning - jupyterlab - amazon web services. https://aws.amazon.com/sagemaker/studio-lab/, Accessed Aug 2024.

[22] AWS. Machine learning service – amazon sagemaker studio – aws. https://aws.amazon.com/sagemaker/studio/, Accessed Aug 2024.

[23] Professor Fabrizio Riguzzi. Data mining and analytics, 2022. appunti del corso, lezione 3.

[24] Professor Fabrizio Riguzzi. Data mining and analytics, 2022. appunti del corso, lezione 8.

[25] Jason Brownlee. A gentle introduction to the gradient boosting algorithm for machine learning, Aug 2020. https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/, Accessed Aug 2024.

[26] Stefano V. Xgboost: L'algoritmo di machine learning potenziato che rende possibili le predizioni vincenti, Oct 2023. https://www.datakpi.ai/2023/10/07/xgboost-lalgoritmo-di-machine-learning-potenziato-che-rende-pos\sibili-le-predizioni-vincenti/, Accessed Aug 2024.

[27] Andrea Minini. Adaboost. https://www.andreaminini.com/ai/machine-learning/adaboost, Accessed Aug 2024.

[28] IBM. What is random forest?, Oct 2021. https://www.ibm.com/topics/random-forest, Accessed Aug 2024.

[29] Scikit-Learn. 1.17. neural network models (supervised). https://scikit-learn.org/stable/modules/neural_networks_supervised.html, Accessed Aug 2024.

[30] Professor Fabrizio Riguzzi. Data mining and analytics, 2022. appunti del corso, lezione 2.

[31] Imbalanced-Learn. Smote. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html, Accessed Aug 2024.

[32] Scikit-Learn. Rfecv. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html, Accessed Aug 2024.

[33] Imbalanced-Learn. Random under-sampler. https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html#imblearn.under_sampling.RandomUnderSampler, Accessed Aug 2024.

[34] Scikit-Learn. Pca. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html, Accessed Aug 2024.