POLITECNICO DI TORINO

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Matematica

Tesina di Laurea

Approcci innovativi nello sviluppo di modelli di Machine Learning per la previsione di churn sui clienti residenziali gas and power

Candidato:
Sara Campi (mat. 265055)

Relatore:
Paolo Garza (DAUIN)
Tutor Aziendali:
Lorenzo Montelatici, Mattia Sanvito

Anno Accademico 2019-2020

Indice

1 Introduzione **3**

2 Il problema del *churn* **5**

2.1 Che cos'è il *churn* 5

2.2 Perché la *customer retention* è importante per i profitti di un'azienda 6

2.3 Le principali cause di abbandono 9

2.4 Strategie comuni: la letteratura 10

2.4.1 *Customer service skills* 11

2.4.2 Consigli per portare a termine con successo il *customer onboarding* 12

2.4.3 Seguire e perseguire continuamente il successo del cliente 12

2.4.4 Imparare dai propri errori. 14

2.5 IL ML nella *churn prevention* 14

3 Il caso Edison **17**

4 Modello attuale: feature building, tecniche, modelli e KPIs **19**

4.1 Scelta dell'orizzonte temporale 19

4.2 Costruzione della *ground truth* 19

4.3 Manipolazione dei dati a disposizione 21

4.3.1 Gestione dei dati mancanti 21

4.4 Scelta delle KPIs 22

4.4.1 *Decision threshold* 23

4.4.2 Metriche di valutazione 23
4.5 Modello impiegato 26
4.6 Risultati e spunti per il lavoro successivo 26

5 Soluzione proposta: il modello ibrido 28
5.1 Aggiunta delle informazioni sui rinnovi contrattuali 29
5.2 Split del dataset 29
5.3 *Data visualisation* 29
5.4 Studio della correlazione tra fattori 31
5.5 Split del dataset di train 32
5.6 *Oversampling* 33
5.7 *Feature selection* 34
5.7.1 Calcolo degli iperparametri per la Random Forest . . . 35
5.7.2 Individuazione del numero ottimale di features 37
5.7.3 Individuazione delle features 39
5.7.4 Creazione delle "*macro-features*" 39
5.8 *Clustering* 42
5.8.1 Individuazione del numero ottimale di clusters 44
5.8.2 Applicazione e valutazione del clustering 45
5.8.3 Analisi dei cluster 47
5.8.4 Caso controllo: clustering solo sulle macro-features . . 50
5.9 Classificazione 52
5.9.1 Scelta e descrizione dei modelli di classificazione 53
5.9.2 Train dei modelli 54
5.9.3 Introduzione dei modelli *cost sensitive* 56
5.9.4 Test dei modelli 62

6 Risultati ottenuti 63
6.1 Cluster separati, classificazione su tutti i dati 63
6.2 Cluster separati, classificazione sulle colonne corrispondenti alle macro-features 70
6.3 Clustering come label del dataset, classificazione su tutti i dati 75

6.4 Clustering come label del dataset, classificazione sulle colonne corrispondenti alle macro-features 77

7 Conclusioni e next steps **80**

Bibliografia **83**

Capitolo 1

Introduzione

Il seguente lavoro di tesi si concentrerà sullo studio del problema del *churn*, o abbandono dei clienti residenziali, nel contesto del mercato delle *utilities*, all'interno del quale si colloca Edison S.p.A., presso le cui sedi è stato svolto il suddetto lavoro. Lo studio si concentrerà principalmente sull'individuazione di tecniche di *Data Mining*, *Feature Extraction* e *Machine Learning* all'avanguardia nella manipolazione dei dati, e nella previsione dell'abbandono dei clienti, al fine di fornire, in futuro, alla sezione marketing, dei validi strumenti di prevenzione dell'abbandono e ritenzione dei clienti. In particolare, il lavoro si soffermerà inizialmente sullo studio del problema del churn in generale (Capitolo 2), con esempi illustri nella letteratura, per poi approfondire le cause storiche che hanno portato Edison S.p.A. a voler investire nella ricerca di soluzioni innovative e competitive al problema (Capitolo 3). In seguito, verrà presentato un breve *excursus* del lavoro precedentemente svolto dai *Data Scientist* Edison per la soluzione del problema (Capitolo 4): la costruzione delle *features*, la scelta dei modelli e dei *KPIs*, i risultati ottenuti. A questo punto il lettore avrà tutti gli strumenti necessari per la comprensione e giustificazione del lavoro svolto dall'autore, che ha interessato soprattutto i seguenti punti (Capitolo 5): tecniche di *oversampling* per il bilanciamento del *dataset* (che per sua natura presenta due classi molto sproporzionate); rielaborazione parziale ed ampliamento delle features, con - in particolare -

aggiunta delle features relative alla storicità dei rinnovi contrattuali; *feature selection*: confronto e valutazione di vari modelli per la riduzione della dimensionalità del database; *clustering* del dataset: uno dei punti fondamentali di innovazione apportata è stata infatti la clusterizzazione dei clienti in classi di rischio al fine di aiutare il modello in fase di allenamento; confronto di vari modelli *ensembles* per la classificazione; introduzione dei modelli *cost sensistive*. Quest'ultima fase in particolare è stata cruciale nella soluzione delle problematiche legate alla classificazione di dataset sbilanciati, e ha proiettato tutto il lavoro sotto una nuova luce, a partire dall'interpretazione delle features e dall'introduzione di una matrice di costo, fino alla scelta delle KPIs. Il lavoro si concluderà (Capitoli 6 e 7) con l'analisi dei risultati e con uno sguardo ai futuri miglioramenti di cui sicuramente potrà essere oggetto il modello.

Capitolo 2

Il problema del *churn*

2.1 Che cos'è il *churn*

Il *churn rate*, o tasso di abbandono o tasso di defezione, esprime la percentuale di clienti che ha abbandonato un servizio in un dato periodo di tempo rispetto al numero totale di clienti che ne ha usufruito nello stesso periodo [1]. Nel caso specifico di un servizio di *utilities* come quello fornito da Edison S.p.A. ai clienti *gas & power*, si parla di churn dei clienti aziendali nel momento in cui un cliente decide di recidere dal proprio contratto stipulato con la società in favore – quasi sempre - di un'altra società *competitor*.

Il concetto di churn è legato a filo doppio con il fenomeno della liberalizzazione del mercato: nel momento in cui il cliente ha la possibilità di ottenere il medesimo servizio da più di una società, infatti, esso naturalmente opterà per quello che si rende più competitivo sul piano del *trade-off* qualità - prezzo.

Questo concetto si applica anche al campo delle utilities: dal 1 Luglio 2007, infatti, il mercato della vendita di gas ed energia elettrica è diventato libero. In particolare, in tale data è stata ufficialmente introdotta una triplice distinzione nello scenario economico della fornitura di gas ed energia elettrica, che si riflette nella segmentazione del servizio in tre figure che per necessità di legge devono rimanere distinte. *In primis*, la **trasmissione**

dell'energia elettrica o gas è affidata ad un'unica società (nel mercato italiano rispettivamente Terna e ItalGas); successivamente la prima parte della **distribuzione** del servizio - dalla centrale alle sottostazioni, in genere - è affidata appunto a società distributrici (per la rete elettrica è per esempio Enel); infine, il cliente sceglie di stipulare il contratto di fornitura con una delle varie società di **vendita**. In questo ultimo contesto entra in gioco la liberalizzazione del mercato, che ha permesso la nascita di molte società tra loro in competizione sul prezzo.

Questo significa che, se prima il cliente era vincolato a stipulare un contratto con l'unico ente italiano distributore e venditore, oggi la scelta è ampia e il passaggio da venditore a venditore è tanto agile da incentivare il cliente a stipulare un nuovo contratto di volta in volta in base all'offerta più conveniente.

2.2 Perché la *customer retention* è importante per i profitti di un'azienda

A seguito, dunque, della liberalizzazione del mercato, le aziende hanno iniziato a rendersi conto che la ritenzione dei clienti ricopriva un fattore determinante nel bilancio. Uno dei primi illuminanti studi in materia è quello svolto dalla Bain & Company dal titolo "Zero defections: quality comes to services" [2]. In particolare, gli autori hanno analizzato - tra gli altri - i dati provenienti da diverse banche degli USA riguardo alle sottoscrizioni a carte di credito. I dati hanno messo in luce che, se durante il primo anno la banca si trova a dover sostenere dei costi di *acquiring* del nuovo cliente, col passare del tempo questo guadagna sempre più fiducia nel servizio stipulato, il che lo conduce a usare più frequentemente la carta, con conseguente aumento costante dei guadagni per la società di credito. Inoltre, la longevità del cliente è inversamente proporzionale ai costi di *caring* che il cliente stesso richiede: da un lato, banalmente, col tempo la società impara a conoscere le esigenze dei suoi affiliati e dunque impiegherà meno tempo e denaro nel soddisfarle;

dall'altro, si osserva che molto spesso i clienti di vecchia data sono disposti a pagare anche un sovrapprezzo per un servizio affidabile a cui sono affezionati, piuttosto che tentare la sorte con un competitor magari meno caro ma di cui non si conosce la qualità del servizio (maggiori dettagli in [2]). Si è notato infine che i clienti di lunga data, oltre a garantire un'entrata costante, forniscono alla società un servizio gratuito di pubblicità: gli autori di [2] riportano ad esempio che una delle maggiori società edilizie degli USA ha verificato che il 60% delle sue vendite derivavano da un servizio di passaparola ad opera di clienti soddisfatti.

Tutti questi fattori contribuiscono, come mostra la Figura 2.1, ad un *trend* costante nell'aumento dei guadagni in funzione del trascorrere degli anni di affiliazione del cliente. Questo suggerisce che, per la società, perdere un cliente e acquisirne uno nuovo, dal punto di vista economico non è affatto comparabile alla *retention* del primo cliente, ma anzi comporta un costante investimento in campagne pubblicitarie, costi di *updating* dei database ecc., a fronte di uno scarso guadagno nell'immediato.

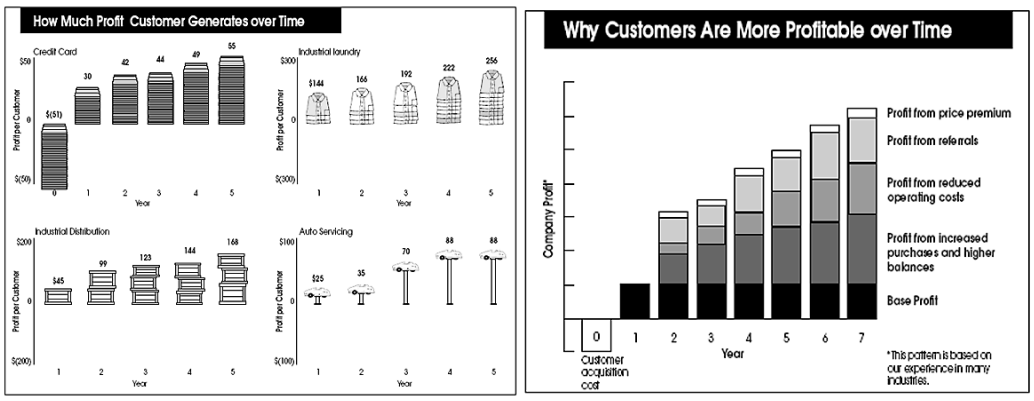


Figura 2.1: Andamento del profitto prodotto da un cliente in funzione degli anni di affiliazione in vari settori economici (sinistra), e sua segmentazione in funzione della provenineza del guadagno nel caso carte di credito (destra). (*img credits*: [2])

In particolare, è utile osservare quella che prende il nome di *defection curve*: un grafico che mette in relazione l'andamento generale dei profitti del-

l'azienda in funzione di una riduzione del *defection rate*, ovvero dell'aumento del tasso di ritenzione dei clienti. A titolo di esempio, si consideri la Figura 2.2: essa mostra chiaramente come anche piccole variazioni - dell'ordine di pochi punti percentuali - del tasso di abbandono dei clienti, siano responsabili di consistenti aumenti di guadagno per l'azienda, fino alla duplicazione degli stessi in alcuni casi.

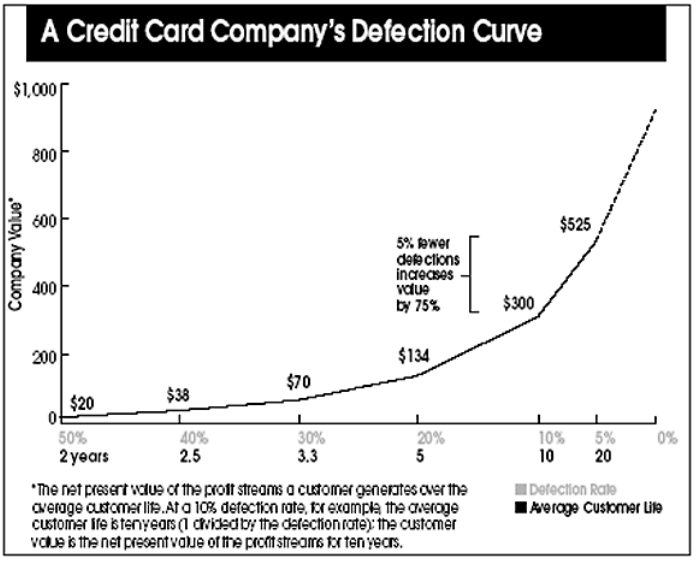


Figura 2.2: Come varia il profitto di una banca in funzione della progressiva riduzione del tasso di *churn* (*img credits*: [2])

Si osservi che lo studio giustifica dunque un costante investimento nell'*upgrade* della qualità del servizio offerto, rappresentando un cambio di rotta rispetto alla tendenza comune del marketing di ridurre i costi fissi. Volendo citare dati numerici, lo stesso paper riporta che in ambito di società di credito, una riduzione dei costi fissi del 10% equivale - in termini di guadagni netti - ad un 2% di riduzione del *defection rate*: piccoli accorgimenti nelle politiche di *customer care* sono dunque da preferirsi alle condotte tradizionali di risparmio e *scouting* di nuovi clienti.

A questo punto dunque diventa necessario individuare degli strumenti

per prevenire l'abbandono dei clienti aziendali; per fare questo occorre innanzitutto chiarire quali siano le maggiori cause che conducono un cliente ad abbandonare l'azienda.

2.3 Le principali cause di abbandono

Il blog Groove, nell'articolo "Why customer churn happens, and what you can do about it" [3], identifica tra le cause di abbandono di clienti quattro categorie principali:

1. **Disservizio.** Il cliente può decidere di rescindere dal contratto nel caso in cui ritenga che l'azienda non abbia prestato accurata assistenza o se addirittura si sente trascurato: uno studio effettuato da Oracle, infatti, attesta che 9 clienti su 10 abbandonano il servizio stipulato con una società a causa di una cattiva esperienza [4].

2. **Cattiva gestione dell'*onboarding*.** Con il termine *onboarding* si indica il processo di acquisizione di un cliente da parte dell'azienda. Sebbene questo processo sia frequentemente confuso con l'atto effettivo di firma del contratto da parte del cliente, queste due fasi quasi mai coincidono. L'onboarding si definisce infatti come l'insieme dei processi adottati dall'azienda nell'introduzione di un nuovo cliente alle dinamiche aziendali, al fine di costruire con lui una buona e durevole relazione di *benefit* [5]. Spesso, infatti, accade che la società ritenga un cliente acquisito quando per la prima volta esso sottoscriva ad un servizio, non tenendo tuttavia conto che, nel tempo che intercorre tra questa prima fase e il momento in cui per la prima volta il cliente realizza la propria soddisfazione, esso può facilmente stancarsi e abbandonare [3].

3. **Mancata realizzazione delle aspettative del cliente.** Anche nel caso in cui l'azienda porti a buon fine il processo di onboarding, al fine di preservare il cliente essa deve costantemente porre attenzione alla sua

soddisfazione. Infatti, nel caso in cui si sentisse trascurato, o più specificatamente ritenesse di non trarre più - o di non aver ancora tratto, dopo un consistente lasso di tempo - alcun vantaggio dalla affiliazione alla società o al servizio, si incorrerebbe in un caso di abbandono a favore del competitor che garantisse un vantaggio maggiore.

4. **Cause naturali.** Esiste comunque una serie di fattori parzialmente indipendenti dalla volontà dell'azienda che può portare all'abbandono del cliente aziendale. Si pensi al caso in cui il cliente non avesse più bisogno del servizio fornito dall'azienda, oppure - nel caso specifico di un servizio utilities - all'eventualità della vendita della casa e del conseguente distacco delle utenze. In realtà, anche in questi casi, se l'azienda dovesse aver lasciato il cliente completamente soddisfatto, non incorrerebbe in alcuna perdita per l'azienda: il cliente potrebbe sì andarsene, ma per esempio consigliando il *brand* a conoscenti, oppure sottoscrivendo un nuovo contratto nella casa nuova.

2.4 Strategie comuni: la letteratura

In un contesto sempre più libero e globalizzato, la competizione è il carburante dei mercati. Qualsiasi scelta compiamo quotidianamente, ci troviamo sempre a dover decidere a quale competitor affiliarci. Dalla scelta del supermercato e del brand per il singolo prodotto all'acquisto di un'auto, dalla sottoscrizione di una polizza assicurativa alla scelta del servizio di *clouding* per la nostra azienda: ogni azione è frutto di una costante azione di marketing da parte delle società rivali al fine di ottenere la fetta di mercato più grande. Come già accennato, tuttavia, si sta rendendo sempre più evidente come, prima di acquisire nuovi clienti, per qualsiasi azienda sia importante trattenere i clienti che già possiede. Insomma, la *fidelity* del cliente è l'aspetto da tutelare, e dovrebbe essere importante ancor prima delle campagne di *recruiting*.

Come fanno, allora, le aziende a convincere i propri clienti a non cedere alle tentazioni di uno degli innumerevoli competitor, che magari li allettano con offerte vantaggiose o premi vari? In base alle cause di churn elencate al paragrafo precedente, si analizzano di seguito le *customer retention strategies* consigliate e vagliate in letteratura.

2.4.1 *Customer service skills*

Il blog *Groove* nell'articolo [6], elenca alcune delle accortezze che occorre adottare nella gestione del cliente e delle sue richieste:

- **Sviluppo di empatia** anche attraverso strumenti diretti quali domande e quiz rivolti al cliente e al dipendente.
- **Puntare alla chiarezza prima che all'obiettivo di vendita**. Occorre abbandonare la convinzione che il cliente ricorra all'assistenza solo quando completamente informato sulle questioni che riguardano il suo accordo contrattuale con l'azienda. La tendenza generale è, infatti, quella di rivolgersi l'assistenza per qualsiasi problematica in cui si incorra nel percorso. Per questo motivo l'azienda dovrebbe istruire i propri membri del *customer service* a non dare nulla per scontato, ma anzi a partire dal presupposto che *less is more*: il cliente non è interessato ai dettagli ma alla risoluzione dei problemi, e questo dovrebbe essere l'obiettivo da perseguire.
- **Rendere il proprio servizio di assistenza *customer friendly***, il che include adottare un tono ed un linguaggio disponibili, mezzi di comunicazione adattabili all'età e alle disponibilità del cliente, e mostrarsi educati ma chiari e precisi nelle comunicazioni e risposte.

2.4.2 Consigli per portare a termine con successo il *customer onboarding*

"Proper onboarding isn't done to prevent churn; it's done to ensure the customer achieves their Desired Outcome. Retention comes from that." - Lincoln Murphy, *growth consultant.*

Nel suo articolo "The Secret to Successful Customer Onboarding" [7], che analizza i requisiti minimi del corretto onboarding per servizi di tipo SaaS, Murphy suggerisce che l'azienda può ritenere di aver portato a termine l'onboarding del cliente in due casi: se esso ha realizzato il suo primo "successo" da quanto ha sottoscritto il contratto, oppure se non lo ha ancora ottenuto ma per la prima volta realizza che la relazione con l'azienda potrebbe effettivamente rappresentare per lui un vantaggio. A questo punto il problema si sposta sul capire cosa rappresenti per il cliente un successo: la soluzione è semplice, occorre domandare. Ogni cliente ha obiettivi e tempi in cui li realizza, specifici e individuali. L'azienda dovrebbe quindi - soprattutto, ma non solo -, in questa fase, prestare grande attenzione al cliente, mantenendosi frequentemente in contatto con lui e chiedendogli esplicitamente quale sia il suo obiettivo e se lo abbia raggiunto o meno.

2.4.3 Seguire e perseguire continuamente il successo del cliente

Se è vero che portare a termine con successo l'onboarding del cliente è *condicio sine qua non* alla customer retention, è pur vero che - come accennato già molte volte - il cliente è soggetto ad un costante *pressing* da parte dei competitors, e nel momento in cui dovesse ritenere di non trarre più alcuno o scarso vantaggio dal servizio a cui è attualmente iscritto, potrebbe guardare altrove senza pensarci troppo. Seguire dunque attentamente il percorso di ogni cliente è la chiave di volta della customer retention.

"Ciò che non si può misurare, non si può migliorare" (Lord Kelvin): occorre dunque individuare un modo per quantificare la soddisfazione del cliente.

Numerosi sono i casi in letteratura che attestano, negli ultimi anni, l'impegno delle aziende - e in particolare dei rispettivi settori marketing e *data science* - nell'individuazione di metriche e metodi per la definizione della *customer satisfaction*. Un esempio illustre è quello di HubSpot, piattaforma che fornisce servizi online quali strumenti per la gestione di blog, dei *social media*, delle email, servizi per la creazione e gestione di *database* contatti eccetera. HubSpot per prima introdusse una misura con cui valutare la propensione all'abbandono dei clienti: il CHI (*Customer Happiness Index*). Il CHI teneva conto di quali specifici servizi del pacchetto offerto da HubSpot fossero maggiormente utilizzati da ciascun cliente, attribuendo un punteggio maggiore alle features che risultassero maggiormente apprezzate. Ad esempio, il servizio HubSpot per il SEO (*Search Engine Optimization*) ottenne un CHI piuttosto negativo, in quanto richiedeva parecchio lavoro da parte del cliente e una volta ottenuto il risultato perdeva di utilità, con conseguente abbandono del servizio da parte del cliente. Al punteggio contribuivano poi fattori come la frequenza e la durata di utilizzo del servizio, tant'é che servizi come quello di database o di *blog managing* garantivano un successo duraturo al cliente, quindi una probabilità molto inferiore di abbandono e di conseguenza ottennero CHI alti. (Per maggiori dettagli consultare [8])

Il problema di questo metodo di valutazione è tuttavia alla luce del sole: come anche HubSpot stesso realizzò in breve tempo, il fatto che un cliente utilizzi un servizio non significa necessariamente che sia totalmente soddisfatto da esso.

Un punto di vista interessante, come suggerisce Skok nell'articolo sopracitato, è quello di tener conto del *business outcome*, il quale permette di rispondere a domande come: qual è stato il vantaggio del cliente nell'usare un determinato servizio, tenendo conto che non è gratis? In altre parole: quanto è fruttato al cliente l'investimento fatto nello stipulare un contratto con la società coinvolta? Rispondere a questa domanda e ad altre - il servizio funziona bene? ci sono *bug* che fanno perdere tempo e pazienza? - significa fornire complessivamente un buon CHI.

A questo punto occorre invertire il punto di vista: non interessa sapere quanto il cliente sia felice, ma piuttosto quanto è probabile che non lo sia, ovvero che abbandoni il servizio. Occorre cioè individuare una formula per un *customer prediction score*. A tal fine si dovrà tener conto di fattori quali:

- pagamento (in tempo) delle bollette
- rinnovi
- cambi di intestatario del contratto
- attività nelle relazioni con altri clienti/con l'azienda
- il cliente ha parlato del servizio con nuovi potenziali acquirenti?

Cercare di dare una risposta a tutti questi punti è alla base della costruzione di un modello efficace nella lotta al customer churn.

2.4.4 Imparare dai propri errori.

Per quanto un'azienda possa migliorare i propri risultati, ci saranno sempre rescissioni dei contratti e abbandoni da parte dei clienti. Quello che in ultima analisi il marketing può fare è trarre vantaggio dai propri errori. Strategie vincenti possono essere ad esempio quella di intervistare i clienti in uscita, indagando sulle motivazioni dell'abbandono e se questo avvenga per motivi di insoddisfazione oppure per cause "naturali"; invogliare i futuri ex clienti a portare con loro il nostro servizio o quantomeno a suggerirlo a terzi.

2.5 IL ML nella *churn prevention*

In seguito alla progressiva presa di coscienza dell'importanza della previsione dell'abbandono dei clienti, si è cercato di dare una risposta rigorosa a questo problema. Essa giunge dal campo della *data science*, in particolare attraverso lo strumento del *machine learning*: costruire una conoscenza approfondita

della natura matematica del problema e conseguentemente individuare modelli predittivi che bene li rappresentino sono i due step che costituiscono la tattica migliore per affrontare preventivamente il fenomeno del churn. Questo è supportato da innumerevoli testimonianze in letteratura: è sufficiente una ricerca anche superficiale per rendersi conto della varietà di tecniche di machine learning che negli anni gli scienziati hanno testato nel tentativo di arginare il problema del customer churn.

A questo punto, dunque, resta solo da decidere quale siano le tecniche che meglio si adattino al problema di cui si tratterà in questo lavoro di tesi. Come si potrà vedere meglio nel Capitolo 5, anche attraverso i paper citati in analisi, storicamente i data scientist hanno intrapreso la via del ML in termini di sviluppo di modelli di classificazione per la suddivisione dei clienti nelle due classi dei churners e non churners. A questo risultato si è giunti vagliando la validità di svariati tipi di modelli. Esistono infatti due approcci diversi all'estrapolazione di informazioni: il *supervised learning*, che sfrutta la conoscenza della *ground thruth* - ovvero l'etichetta churner-non churner - per vagliare la validità del modello, e l'*unsupervised learnig*, che al contrario parte dal presupposto dell'ignoranza dell'etichetta di classificazione e cerca di estrapolare regole nascoste che descrivano la somiglianza tra i record. Questa seconda linea di condotta genera tutta via quella che in gergo prende il nome di *black box*, ovvero cela le proprie regole di comportamento, rendendo imprevedibile e difficilmente maneggiabile i propri parametri e risultati. Per quanto riguarda invece il *supervised learning*, esistono almeno tre categorie di modelli considerabili: modelli semplici di classificazione, modelli ensemble (che reiterano con varie metodologie i modelli semplici al fine di ottimizzarne i risultati) e reti neurali. Le differenze tra questi approcci verrà analizzata in seguito. Come vedremo, tuttavia, la via delle reti neurali presuppone una caratterizzazione stringente del dataset in termini di informazione fornita, per cui la loro applicazione esula dalla capacità di studio di questo lavoro.

Le scuole di pensiero in termini di classificazione di questo tipo di dataset si sono divise tra chi favorisce l'uso dei modelli supervised, chi quelli unsu-

pervised e chi ancora ha provato ad unire le due. A quest'ultima categoria si è ispirato il presente lavoro di tesi, cercando di testare l'efficacia della combinazione di un modello di clustering con i classici modelli di classificazione ensemble.

Capitolo 3

Il caso Edison

Per rispondere alla liberalizzazione del mercato dell'energia, Edison ha costituito una società dedicata - Edison Energia - che si occupasse della vendita ai clienti finali del servizio luce e gas.

La qualità e la competitività del servizio hanno permesso a Edison di raggiungere più di un milione di clienti nel corso degli anni. Per soddisfare le richieste, spesso diverse, di un così grande numero di clienti, è stato istituito un *customer care* dedicato, che ha come obiettivo quello di massimizzare la soddisfazione del cliente e assicurarsi la sua fedeltà.

Con l'esplosione della cultura dei dati, sono stati messi in piedi dei sistemi aziendali che ne permettessero al collezione, l'organizzazione e l'analisi; questo si è rivelato particolarmente utile nel momento in cui le capacità di calcolo e di analisi si sono sviluppate a tal punto da permettere di osservare le informazioni dei clienti anche tramite algoritmi automatici.

Il processo di trasformazione digitale dell'azienda ha investito anche l'area di marketing, offrendo all'unità di business degli strumenti digitali che rispondessero in modo innovativo a delle esigenze specifiche; il caso che trattiamo è il problema della retention dei clienti residenziali, ossia quello di minimizzare il tasso di churn in quest'ambito.

Lo strumento ideato permette di associare ad ogni cliente, sulla base delle informazioni realmente disponibili, una probabilità di churn nel medio

periodo: il team interno di Data Scientist ha infatti sviluppato un *tool* che, utilizzando modelli avanzati di machine learning, permettesse di estrarre le informazioni realmente utili a partire dai dati grezzi.

Questo ha permesso di indirizzare in modo ancora più preciso le campagne di retention: gli operatori del customer care riescono ad avere evidenza dei clienti con più necessità di intervento, e diventa più semplice portare alla luce per tempo le eventuali problematiche del cliente, in modo da poterle risolvere.

In ottica di migliorare continuamente la soluzione proposta, il seguente lavoro di tesi si propone di esplorare alcune possibilità di sviluppo del modello, con l'obbiettivo di rendere il tool ancora più preciso, versatile ed efficace.

Capitolo 4

Modello attuale: feature building, tecniche, modelli e KPIs

Di seguito si analizzano le caratteristiche del modello attuale, al fine di fornire una concreta base di partenza all'analisi del lavoro di tesi.

4.1 Scelta dell'orizzonte temporale

I dati a disposizione consistono in un database aggiornato che contiene le informazioni, statistiche e reali, dei contratti gas e luce sottoscritti da Edison e dalle sue società affiliate con clienti residenti nel territorio italiano. Il primo step nella creazione del modello è rappresentato dalla scelta dell'orizzonte temporale di dati da includere: al fine di assicurarsi una migliore osservazione dell'evolversi del mercato nel medio periodo, si è optato per l'utilizzo dei dati raccolti nell'arco dei 24 mesi antecedenti alla data di analisi, tenendo gli ultimi 6 come finestra temporale utile al test del modello.

4.2 Costruzione della *ground truth*

Quello che forse ha rappresentato la sfida più *challenging* del problema è la costruzione della *ground truth*, ovvero dell'etichetta 0/1 che risponde alla

domanda: questo cliente rappresenta un caso di churn?

Ad una prima analisi potrebbe sembrare che questa domanda coincida con la seguente: il contratto attualmente in analisi è ancora attivo, oppure è stato cancellato? In realtà, presto i data scientist si sono resi conto che le circostanze che conducono alla cancellazione di un contratto non sono così banali, ma anzi possono includere dinamiche di vario tipo. In particolare, sono state distinte le seguenti eccezioni alla classe dei churners:

- **Voltura**: il contratto non è più attivo in quanto il cliente ha richiesto una voltura, ovvero un cambio dell'intestatario del contratto. Questo non coincide con un caso di churn, in quanto di fatto non si è perso alcun cliente, anche se a livello di database figura come cancellazione (e a seguire riapertura) di un contratto.

- **Distacco per morosità**: a volte rientra negli interessi dell'azienda perdere un cliente, come nel caso in cui questi sia tendente alla morosità, ai ritardi o ai mancati pagamenti. In tali casi l'azienda, ove non riesca a recuperare il credito, distacca la linea al cliente, e lo elimina dalla lista dei possibili nuovi contratti. In tal caso non parliamo di churner.

- **Disdetta**: nel caso in cui, per varie motivazioni, il cliente debba abbandonare e lasciar vuoto un domicilio, egli richiede il distacco degli allacci per luce e gas, che non verranno ripristinati a meno di future indicazioni. In tal caso la cancellazione del contratto non rientra nei casi di churn, ma piuttosto si parla di disdetta - o chiusura - del contratto.

Eliminate dunque queste eccezioni, quello che rimane sono effettivamente i casi di abbandono del cliente, ovvero quelli in cui il cliente ha scelto di *switchare* ad altro venditore.

In questo modo il dataset è ora dotato di una nuova colonna, etichettata come "*churn label*", che costituisce l'informazione base per il problema di classificazione.

4.3 Manipolazione dei dati a disposizione

Il dataset così come creato è frutto delle informazioni raccolte in vari contesti:

- **CUSTOMER BASE**: è la tabella più fitta, e contiene informazioni varie sui contratti e sui loro intestatari. In particolare, include informazioni specifiche sul contratto e sul cliente, come età, sesso, se il contratto sia attivo o meno ecc., così come informazioni di natura demografica sul luogo di residenza e sulle sue caratteristiche climatiche, sulla popolazione.
- **CAMPAGNE PROMOZIONALI**: contiene i record dei contatti presi dai *call centers* dell'azienda con i clienti nell'ambito di campagne promozionali.
- **VAS**: riguarda i *servizi a valore aggiunto* stipulati da alcuni clienti con Edison, in aggiunta alla fornitura base.
- **FATTURATO ENERGIA ELETTRICA/GAS**: contiene i dati relativi alle fatture emesse da Edison per i contratti di vendita di energia elettrica e gas.
- **CONTATTI CLIENTE-AZIENDA**: contiene i record dei contatti instaurati dai clienti con Edison e le loro motivazioni.

A questo punto i data scientist hanno affrontato la fase vera e propria di *feature building*, andando a manipolare le colonne preesistenti e creandone di nuove, più significative dal punto di vista delle informazioni necessarie alla creazione del modello.

4.3.1 Gestione dei dati mancanti

Per loro natura, i dati raccolti sono caratterizzati da una certa sparsità, presentano - cioè - parecchi *NaN values*, ovvero dati mancanti. In generale, esistono varie tecniche per affrontare questo problema: se per esempio i dati mancanti sono in percentuale molti di più di quelli presenti, di solito si

preferisce ignorare del tutto quella feature, in quanto maneggiarla potrebbe alterarne la reale distribuzione e influenzare i risultati del modello; se invece i dati mancanti fossero presenti ma non predominanti, allora si può procedere al completamento dell'informazione in due modi: se il dato è numerico e continuo si può pensare di calcolare la media dei valori presenti, e di sostituire ai NaN il valore ottenuto; se invece avessimo a che fare con features discrete, per esempio un valore numerico ma non continuo oppure delle stringhe, allora la tecnica è la medesima ma al posto della media si calcolerebbe la moda dei valori presenti (ovvero il valore più frequente in quella classe).

Questo è quello che è stato fatto dai colleghi, per cui alla fine di questa fase la tabella si presenta completa e pronta alla manipolazione.

4.4 Scelta delle KPIs

A questo punto, prima di procedere con lo studio dei modelli adatti al problema, occorre soffermarsi sulla natura dello stesso e decidere quali siano le metriche che in seguito verranno utilizzate nella valutazione delle prestazioni (parliamo di KPIs, ovvero: *Key Performance Indicator*, indicatore chiave di prestazione).

Questa scelta, infatti, è strettamente legata alla natura dei dati e del problema. Nel nostro caso, quello che ci troviamo ad affrontare è un *unbalanced dataset*, ovvero un insieme sbilanciato. In particolare, questo sbilanciamento si riferisce alla netta predominanza della classe dei non churners, o 0, su quella dei churners, o 1.

Quando ci si trova ad affrontare problemi di questo tipo, occorre rivedere l'intero approccio machine learning, il che include due aspetti in particolare: l'individuazione della *decision threshold* dei modelli, e le metriche di valutazione.

4.4.1 *Decision threshold*

Con questo termine si intende la soglia di confidenza con cui il modello separa i casi classificati come negativi da quelli positivi.

La gran parte dei modelli di classificazione, infatti, attribuisce ad ogni record uno *score*, un punteggio compreso tra 0 e 1, che indica la *confidence* con cui il modello stesso è in grado di attribuire a quel determinato record l'etichetta di una classe o dell'altra. Se per esempio si considerano gli score per la classe 1, un record con uno score di 0.2 sarà classificato come positivo con una confidence molto bassa, ovvero: il modello ritiene con una certezza di 0.8 che quel record sia da classificarsi come negativo.

A questo punto, interviene il concetto di decision trhreshold: esso è di fatto il valore soglia che trasforma gli score in etichette vere e proprie, quel valore che rappresenta la confidence minima di cui ha bisogno il modello per etichettare un record come positivo o negativo. In generale, questo valore è settato a 0.5, ma nel caso di modelli unbalanced, che possiedono cioè molti record negativi e pochissimi positivi, il modello si allena in modo tale da avere un confidenza bassissima nell'etichettare un record come positivo. Occorre dunque, di volta in volta, suggerire "manualmente" al modello quale soglia adottare nell'etichettare i dati, in modo tale da evitare troppe misclassificazioni.

4.4.2 Metriche di valutazione

Consideriamo la matrice di confusione di un problema, e definiamo le quattro classi che vi compaiono: *True Positive*, *True Negative*, *False, Positive* e *False Negative* (vedi Figura 4.1).

Una delle metriche in assoluto più usate nella valutazione di modelli di *supervised learning* è la ***accuracy***, definita come:

$$\text{accuracy} = \frac{TPs + TNs}{TPs + TNs + FPs + FNs}$$

ovvero come il rapporto tra la somma delle entrate diagonali della tabella e il numero totale dei record del database, o meglio, del training set. Si consideri

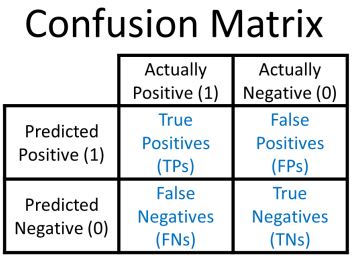


Figura 4.1: Matrice di confusione di un qualsiasi problema di classificazione.

tuttavia un problema sbilanciato, che abbia ad esempio il 99% dei record negativi e solo l'1% positivi: se in un caso il modello classificasse tutti i record come appartenenti alla classe dominante, esso sarebbe ovviamente fallimentare (non classificando correttamente nemmeno uno dei casi a rischio, che sono quelli di interesse), ma comunque l'accuracy score del modello sarebbe del 99%, un punteggio molto alto, ma fuorviante. Appare dunque chiaro che l'accuracy score sia in questo caso da scartare nella valutazione dell'efficienza dei modelli.

Altre metriche spesso adoperate in machine learning sono la ***precision*** e la ***recall***, definite rispettivamente come (Figura 4.2):

$$\text{precision} = \frac{TPs}{TPs + FPs}, \quad \text{recall} = \frac{TPs}{TPs + FNs}$$

Al variare della *decision threshold* del modello, precision e recall hanno un andamento opposto: infatti, classificando tutti i casi come positivi la precision è uguale all'incidenza dei casi positivi, mentre la recall è pari a 1, valore massimo; se invece tutti i record venissero assegnati alla classe 0, allora la recall andrebbe a 0 mentre la precision tende asintoticamente a 1.

Valutare dunque un modello in funzione solo di una o dell'altra metrica è, nel caso di dataset sbilanciati, ancora una volta fuorviante. Nel caso estremizzato citato sopra, prediligendo la precision si tenderebbe a favorire modelli che classificano tutti i record come negativi, viceversa scegliendo modelli a recall alta, tutti i record rientrerebbero nella classe dei positivi

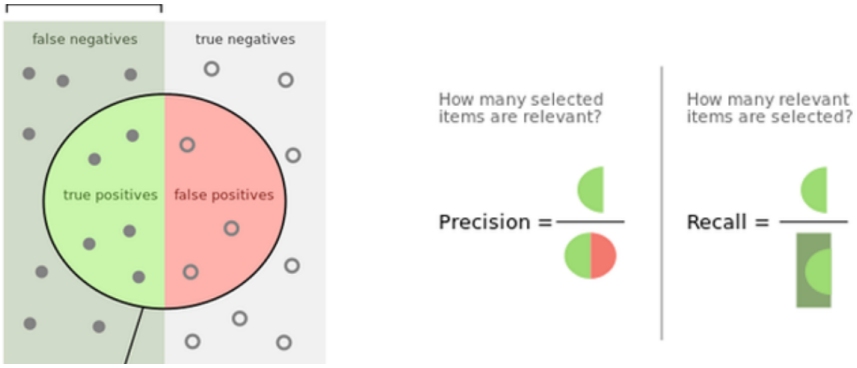


Figura 4.2: Rappresentazione del calcolo di precision e recall di un modello.

(trovando quindi modelli con accuracy nulla o quasi, data la definizione di cui sopra e la natura sbilanciata del problema).

In questi casi giunge però in aiuto una nuova metrica, l'***f1-score***, che è di fatto la media armonica di precision e recall:

$$\text{f1} = 2\frac{p \cdot r}{p + r}$$

Impiegando questa metrica, si è in grado di individuare i modelli con una buona prestazione anche su dataset di tipo sbilanciato.

Concludiamo citando un altro tipo di metrica efficiente in casi come quello trattato, la ***average precision***: questa è di fatto una media ponderata della precision di cui sopra calcolata al variare della decision threshold. In questo modo, non sarà il data scientist a dover effettuare di volta in volta il calcolo della *best threshold* in funzione di modello e dati, ma in automatico la metrica terrà conto del variare del punteggio allo spostarsi della soglia, fornendo una visione d'insieme delle prestazioni del modello stesso.

Su questa opzione è ricaduta la scelta dei colleghi durante la prima stesura del modello, mentre - come vedremo - per il presente lavoro di tesi si è prediletto l'uso dell'f1-score.

4.5 Modello impiegato

Il modello scelto inizialmente per la previsione dell'abbandono dei clienti è l'*LGBMClassifier* del pacchetto *lightGBM* di Python (documentazione alla pagina [10]), un modello *ensemble* di tipo *boosting*, ovvero un modello machine learning che applica ripetutamente al dataset un modello semplice, di volta in volta pesando in maniera più significativa gli errori effettuati allo step precedente; in tal maniera si spera di ottenere un modello preciso e accurato nei confronti della classe di minoranza (referenze alla pagina [9]). In particolare, il modello adottato ha come classificatore base il *decision tree classifier*, ed è customizzabile in funzione dei dati grazie alla varietà di parametri di cui è dotato. La peculiarità del LGBM rispetto ad altri modelli che verranno analizzati i seguito, è che la costruzione dell'albero di base non avviene in modo *level wise* (in genere l'albero splitta i dati in funzione della feature che ritiene più significativa), ma *leaf wise* ovvero scegliendo la foglia, il record, che si ritiene condurrà a risultati migliori [29].

Prima di approdare alla scelta di questo modello, sono state eseguite delle prove per mettere a confronto le prestazioni di alcuni modelli più o meno semplici (modelli semplici, ensemble di tipo boosting e di tipo bagging, che descriveremo in seguito); inoltre, una *param grid cross validation* è stata impiegata al fine di individuare i parametri ottimali del modello in funzione dei dati a disposizione.

Si fa notare che i colleghi non hanno valutato l'opzione *reti neurali* in quanto esse necessitano di una certa *feature importance* di cui erano privi i dati, per cui non sarebbero riuscite nella distinzione dei singoli record.

4.6 Risultati e spunti per il lavoro successivo

I risultati ottenuti dai colleghi mostravano netti segnali di *overfitting* del modello, fenomeno riconducibile in primis ad una mancata fase di *feature selection*, implementata dunque nel lavoro che segue.

Il modello così come è stato presentato, inoltre, ha lasciato un certo spazio di manovra per quanto riguarda l'implementazione dei modelli *supervised*: su quest'aspetto, dunque, si concentra maggiormente il lavoro di tesi.

Infine, un aspetto molto interessante del dataset è proprio il suo unbalancing: il modello attuale lo affrontava solo in termini di oversampling del dataset, ma il lavoro a seguire presenterà nuove promettenti tecniche per risolverlo mettendolo sotto una luce completamente nuova.

Capitolo 5

Soluzione proposta: il modello ibrido

Di seguito verrà descritto l'approccio adottato nel miglioramento del modello preesistente. Sicuramente il punto di vista innovativo più significativo è stato quello di introdurre un modello *ibrido*, ovvero uno che integrasse aspetti di unsupervised learnig (in sostanza, il clustering del dataset) all'interno di un modello puramente supervised. Inoltre, come verrà analizzato in dettaglio più avanti, il problema dell'unbalancing del dataset è stato trattato anche attraverso l'introduzione dei modelli *cost sensitive*, ovvero di modelli che - a seguito della definizione di *costi* per la misclassificazione dei record - mirano a massimizzare il "risparmio" derivante dall'uso di un certo tipo di modello, forzandolo - in ultima analisi - a favorire la corretta classificazione dei casi positivi su quelli negativi.

Premessa alla trattazione che segue è che, per pura comodità, il lavoro svolto si è concentrato solo sui clienti del settore luce, tralasciando quelli del settore gas. Tuttavia, per analogie di struttura dati, il modello potrebbe essere tranquillamente replicabile in quest'ultimo caso.

5.1 Aggiunta delle informazioni sui rinnovi contrattuali

Il primo *improving* del modello consiste nell'introduzione nel dataset delle informazioni relative ai rinnovi contrattuali. Questa informazione è infatti importante nell'individuazione dei churners, in quanto un cliente storico è sicuramente meno propenso al churn di un nuovo cliente. Partendo dunque da una tabella contenente lo storico delle informazioni relative ai rinnovi sui contratti dei clienti, sono state estratte informazioni computazionalmente utili al modello. Le colonne presentavano un numero di *missing values* ragionevole, per cui si è optato per le tecniche classiche di *filling* degli stessi.

5.2 Split del dataset

Anche se è una procedura standard, va sottolineato che tutte le operazioni che seguono sono state effettuate su un sottoinsieme dei dati in possesso. Infatti, prima delle analisi è stata applicata una funzione di split del dataset, in modo tale di ottenere due insiemi, X_train e X_test, distinti e complementari. Lo studio delle features, della loro distribuzione e correlazione, e l'allenamento del modello verranno svolti sul primo, conservando il secondo per la fase terminale dell'iter, ovvero il test del modello vero e proprio.

5.3 *Data visualisation*

Si procede a questo punto con una serie di plot ai fini di evidenziare la distribuzione dei dati e la loro suddivisione nelle due classi, churners e non churners, così come sono state costruite dai colleghi nella versione precedente del modello.

Questa fase ha lo scopo di farsi un'idea iniziale di quali features potrebbero o meno influenzare la risposta. Ad esempio, features totalmente o in gran parte rappresentate da una sola delle due classi influenzeranno maggiormente

la risposta, mentre altre distribuite egualmente tra le due classi potranno essere scartate. Allo stesso modo, nel caso di colonne in cui comparisse un solo valore su tutto il dataset, queste potrebbero tranquillamente essere escluse dal processo.

Per questa operazione si è scelto di usare *violin plot* per le features continue (si tratta di plot che evidenziano la distribuzione delle features, con i rispettivi quantili, in parallelo tra le due classi), mentre per le features discrete la scelta è ricaduta su classici *count plot* (si vedano esempi dei due plot in Figura 5.1).

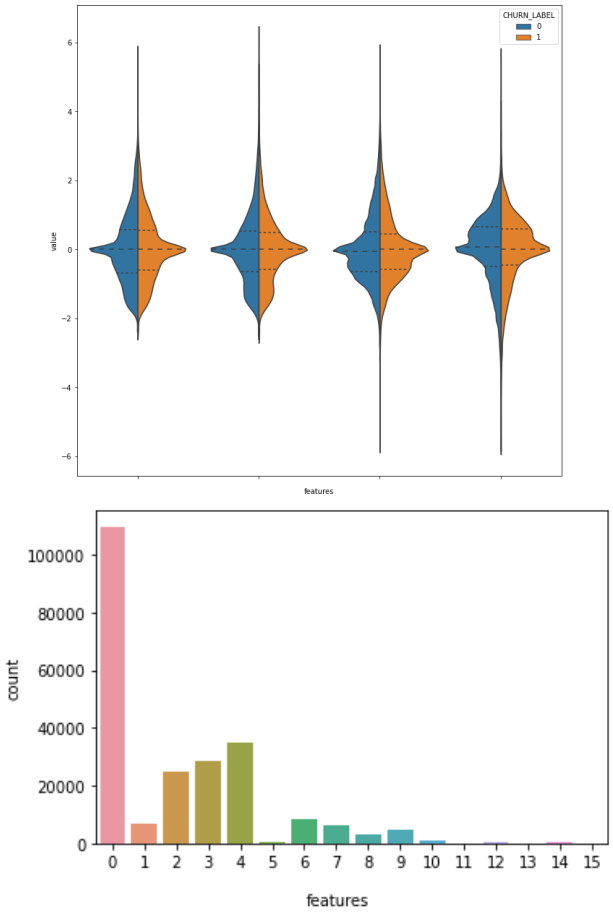


Figura 5.1: Esempio di *violin plot* (sopra) e di *count plot* (sotto) per la visualizzazione della distribuzione delle colonne.

Per ragioni di riservatezza sui dati non è possibile riportare le immagini prodotte in questa fase. Basti sapere che già dopo questo primo step è stato possibile eliminare dal processo tre colonne del dataset, in quanto costituite tutte dallo stesso valore.

5.4 Studio della correlazione tra fattori

Il passo successivo è la ricerca di eventuali correlazioni tra i fattori. Individuare le relazioni tra le features è importante nell'iter dello sviluppo di un modello predittivo: features correlate, infatti, non aumentano l'informazione fornita alla variabile risposta, ma al contrario appesantiscono il modello conducendolo sulla strada dell'overfitting. Per questo motivo si è scelto di eseguire degli studi mirati al fine di individuare eventuali correlazioni tra i fattori.

Si noti che esistono dei gruppi di features che sicuramente saranno correlati tra loro: se per esempio si considera il gruppo delle informazioni sulle percentuali delle coppie, single o altro tipo di famiglia presenti in città, allora sicuramente queste tre sommeranno a uno. Per questa ragione, due delle tre saranno sufficienti a fornire tutta l'informazione, mentre la terza sarà ridondate e dipendente dalle altre due, motivo per cui potrà essere esclusa dal modello. Analogamente la situazione si ripete per le altre informazioni che possono essere considerate complementari.

Per questo studio sono stati impiegati due diversi tipi di grafico:

- *heatmaps*, utili nello studio nella correlazione di un insieme di features in genere maggiore di due;
- *joint plot*, per confrontare la distribuzione di due features e mettere in evidenza eventuali relazioni.

A titolo di esempio, si considerino i grafici riportati in Figura 5.2.

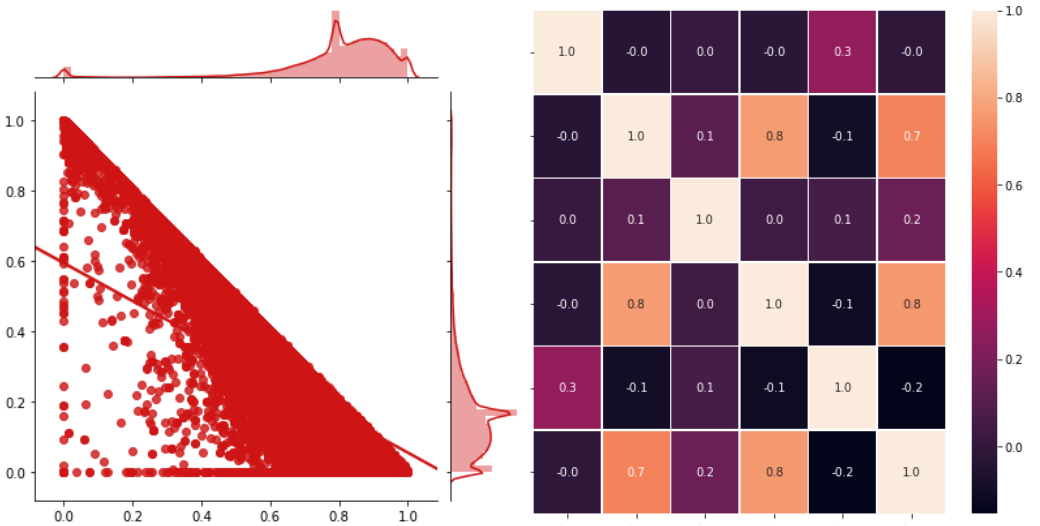


Figura 5.2: Esempio di *jointplot* (a sinistra) e di *heatmap* (a destra) per la visualizzazione della correlazione tra le colonne. Nel primo esempio appare chiara una correlazione tra i due fattori, le cui istanze per la maggior parte possono essere regresse su una retta (i punti appartenenti agli assi verticale e orizzontale possono essere sintomo di *NaN values* precedentemente presenti e riempiti per comodità con il valore nullo); per quanto riguarda la *heatmap*, invece, valori di correlazione come 0.7 o 0.8 indicano una chiara correlazione tra le due colonne corrispondenti nella matrice.

Analogamente al paragrafo precedente, non si riportano nel dettaglio i risultati dello studio, che sono comunque valsi un ulteriore snellimento delle colonne a favore di una più efficace prassi di feature selection.

5.5 Split del dataset di train

Al fine di ottimizzare ulteriormente l'allenamento del modello che andremo a svolgere in seguito, si è reso opportuno eseguire un nuovo split, questa volta di *X_ train*, al fine di ottenere due nuovi sottoinsiemi: sul primo - a seguito di opportune modifiche di cui parleremo poco più avanti -, chiamato *X_ training*, il modello verrà allenato (parliamo di *train* del modello), mentre

sul secondo, *X_ validation*, di dimensione inferiore, verrà verificata la sensatezza del modello come allenato, prima di sottoporlo alla fase di test finale: in questo caso si parla di *validation* del modello.

Questa fase si rende necessaria al fine di scongiurare l'ipotesi di overfitting del modello. In realtà, prassi ottimale in questo senso sarebbe quella di effettuare quest'ultimo split in un contesto di cross-validazione, al fine di validare ogni volta il modello su dati diversi scelti casualmente. Data tuttavia l'importanza del numero di dati e gli strumenti tecnologici a disposizione, questa operazione - computazionalmente molto onerosa - è stata snellita, riducendosi ad un solo split del dataset. Al fine di assicurarsi comunque la buona riuscita della prassi, tutte le suddivisioni sono state eseguite in modo *stratificato* (tenendo cioè conto della proporzione tra le due classi e riproducendola nei sottoinsiemi prodotti). Inoltre, come vedremo in seguito, i modelli verranno scelti in base agli score calcolati su *X_ validation*, ma in seguito ri-allenati su *X_ train* nel suo complesso prima di passare alla fase di test, in modo tale da avere a disposizione il maggior numero di informazioni possibile.

5.6 *Oversampling*

Come è già stato molte volte accennato, il dataset con cui ci si trova a lavorare soffre di un problema di *unbalancing*, il che mette a rischio il corretto allenamento di qualsiasi modello di classificazione. Dopo una ricerca in letteratura, si è reso evidente che una delle tecniche maggiormente adoperate in casi come questo è quella dell'*oversampling*. Si tratta, in realtà, di un insieme di tecniche volte ad ampliare il numero dei record appartenenti alla classe in minoranza, in modo tale da andare a bilanciare - almeno parzialmente - lo squilibrio del dataset. I record in questo modo aggiunti al dataset possono essere l'esatta riproduzione di alcuni - o tutti - quelli presenti, oppure del *bias* può essere introdotto, in modo tale da variare il dataset ed evitare l'overfitting.

La tecnica adottata per questo lavoro è quella dello SMOTE (*Synthetic Minority Over-sampling Technique*) (documentazione alla pagina [11]). Questa tecnica considera volta per volta un record - o riga, o punto nello spazio generato dalle colonne - del dataset e calcola i suoi k *nearest neighbors* nello spazio sopracitato. Al fine di generare un nuovo record, si consideri a questo punto il vettore compreso tra uno di questi k "vicini" e il record reale in considerazione, e lo si moltiplichi per un numero casuale compreso tra 0 e 1. Infine, si sommi scalarmente il vettore così trovato al punto preso in considerazione all'inizio; il risultato è un nuovo record da aggiungere al dataset.

Trattandosi di un dataset già di per sé numeroso, si è preferito fare un compromesso tra il completo bilanciamento del dataset e velocità computazionale. Per questa ragione l'oversampling è stato eseguito fino ad ottenere un dataset con una percentuale di churners pari al 20% sul totale.

Si fa notare che l'algoritmo di SMOTE utilizzato nel codice, e la cui documentazione è riportata in bibliografia, è quello compreso nella libreria dei modelli *cost sensistive*, il cui funzionamento e scopo verrà illustrato in seguito. Non cambia tuttavia il principio di funzionamento della tecnica di oversampling, per cui le indicazioni di cui sopra possono essere intese a titolo generale.

5.7 *Feature selection*

A questo punto si entra nel vivo del modello, introducendo un nuovo step rispetto al suo predecessore: la *feature selection*. Al fine di migliorare le prestazioni del modello, e in particolare di ridurre l'overfitting, si è scelto infatti di ridurre la dimensionalità del dataset (che a questo punto consisteva di 95 colonne circa), ovvero di eliminare ulteriori features che apportassero poca o nessuna informazione alla variabile risposta.

Per compiere questa operazione sono state impiegate in serie due tecniche: innanzitutto, il numero ottimale di features da selezionare è stato individuato

applicando una *Random Forest Classification* e valutandone di volta in volta i risultati con il calcolo dell'f1-score; in seguito, si sono individuate propriamente quali fossero le features più rilevanti utilizzando una *Recursive Feature Elimination*.

5.7.1 Calcolo degli iperparametri per la Random Forest

Innanzitutto è stata eseguita una *Grid Search Cross Validation* al fine di individuare i parametri della *random forest* che meglio soddisfacessero le condizioni del dataset. La random forest è un modello di classificazione di tipo *ensemble*, il che significa che il modello di classificazione base - il *decision tree*, di fatto un albero che splitta i record in funzione della feature che di volta in volta ritiene sia più significativa nella caratterizzazione delle classi - viene ripetuto più volte in serie, e il risultato è funzione di quelli intermedi. Nello specifico, si tratta di un modello ensemble a logica *bagging*: questo significa che ad ogni iterazione il modello estrapola dal dataset un sottoinsieme di record con reinserimento (si parla di *bootstrap*) ed allena il decision tree su questi dati. Alla fine delle iterazioni, la label per la classe viene assegnata ad ogni record con la logica della maggioranza (documentazione alla pagina [12]).

Sfogliando, dunque, un dizionario in cui ai parametri *min_ samples_ split*, *n_ estimators*, *max_ depth* e *max_ features* viene assegnata una lista di diversi possibili valori (si veda per riferimento l'immagine 5.3), la *grid search* allena - di volta in volta su un sottoinsieme di dati diverso, prodotto di una cross-validazione - il modello della *random forest* con la combinazione di parametri attuale, calcolandone poi le *prediction* e valutandole per mezzo dell'f1-score. Trattandosi questa di una metrica da massimizzare, la combinazione di iperparametri scelti per il modello è quella a cui corrisponde il massimo f1-score.

Si fa notare che la random forest possiede inoltre, nell'elenco dei parametri customizzabili, anche quello del *class_ weight*. Questo parametro indica la possibilità di attribuire ai record del dataset un peso diverso da quello

```
param_grid = {
    'min_samples_split': [3, 5],
    'n_estimators': [100, 200, 300],
    'max_depth': [5, 10, 15],
    'max_features': [10, 20]
}
```

Figura 5.3: Griglia parametrica associata al calcolo dei *best parametres* della *random forest*. Si noti che il valore *max_depth* on contemplava l'ipotesi del *None*: in questo modo ci si è tutelati nei confronti del prolungarsi esagerato dei tempi computazionali per le simulazioni.

omogeneo, in modo tale, eventualmente, da forzare il modello su alcuni dati e fargliene tralasciare altri. Si tratta di uno strumento fondamentale nella trattazione di dataset sbilanciati come quello in analisi. Per questa ragione, indipendentemente dai risultati della grid search, si è scelto di fissare questo parametro al valore $class\_weight = 'balanced\_subsample'$, che significa che il modello calcola - di volta in volta sulla base dei dati prodotti dal bootstrap per il modello attuale - un peso specifico per i record delle due classi, inversamente proporzionale all'estensione della classe stessa, dunque pesando di più la classe di minoranza rispetto a quella di maggioranza.

In base alle osservazioni fatte e ai risultati della grid search, il modello risultante (chiamato per semplicità *model_best* nel resto della trattazione) risulta così caratterizzato:

$$model\_best = RandomForestClassifier(random\_state = 12, max\_depth = 10,$$

$$max\_features = 20, min\_samples\_split = 5, n\_estimators = 300,$$

$$n\_jobs = -1, class\_weight =' balanced\_subsample')$$

Il modello con la combinazione ottimale di iperparametri è quello utilizzato in seguito ogni qual volta ci si trovasse a dover ricorrere ad una random forest.

5.7.2 Individuazione del numero ottimale di features

A questo punto la random forest ottimizzata come sopra è stata ricorsivamente applicata al dataset, di volta in volta con un numero minore di features. In particolare, i dati ricorsivamente privati di qualche colonna venivano sottoposti al modello di classificazione; a questo punto, al fine di valutare in modo ottimale le performance dello stesso, occorreva calcolare la decision threshold del modello. Per fare questo è stata elaborata una procedura che in seguito verrà utilizzata ogni qual volta richiesto:

1. innanzitutto la funzione *precision_ recall_ curve* di Python [13] calcola - per un vettore omogeneo di possibili valori della decision threshold - i corrispondenti valori di precision e recall, eventualmente plottabili come da Figura 5.4;

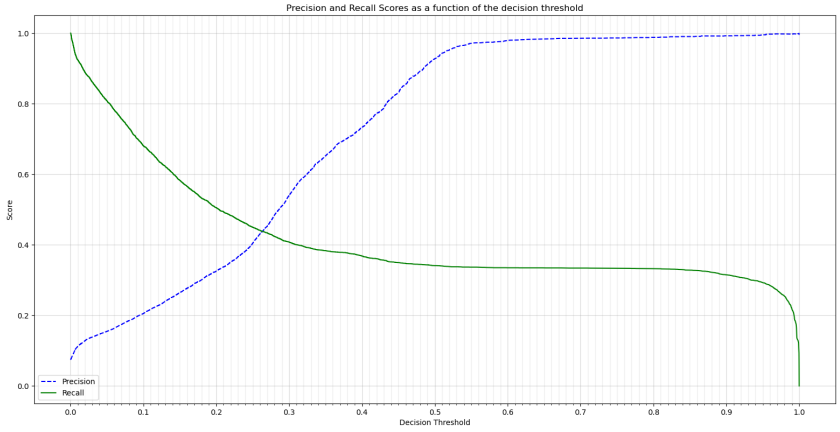


Figura 5.4: Plot esemplificativo di un grafico congiunto di precision e recall al variare della decision threshold del modello.

2. a questo punto, l'f1-score viene calcolato "a mano" per mezzo della formula di media armonica tra precision e recall;

3. in funzione del valore massimo di f1-score così calcolato, si risale alla *best threshold* per il modello in uso.

Così facendo, confrontando le prestazioni in funzione del numero di colonne impiegato, e tenendo conto di un certo trade-off tra accuratezza del modello e overfitting, è stato possibile notare che il dataset ridotto ad un numero di colonne pari a 50 - ovvero quasi della metà rispetto a quello originale - forniva al modello informazione sufficiente ad una buona classificazione dei churners (come si vede dalla Figura 5.5).

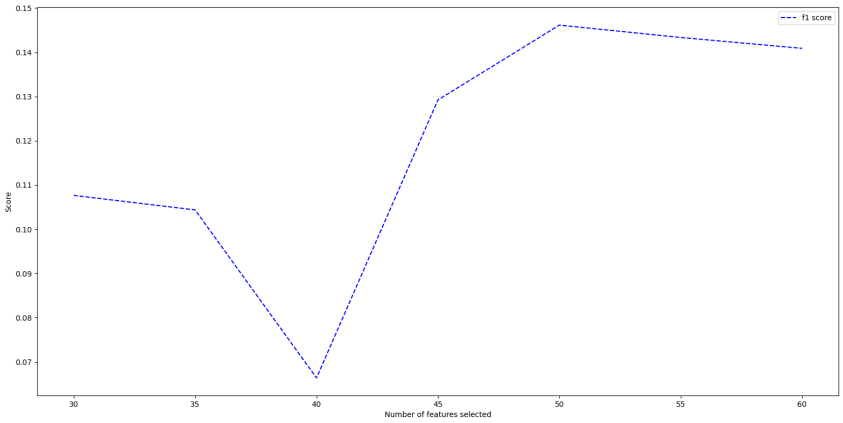


Figura 5.5: Come si osserva dalla figura, in corrispondenza di 50 colonne il modello di classificazione garantisce un ottimo locale in termini di f1-score, pur avendo ridotto significativamente la cardinalità del dataset.

Per completezza, si fa presente che in prima analisi si era pensato di considerare come *best threshold* quella corrispondete al punto di incontro tra il grafico della precision e quello della recall: osservando infatti la formula per il calcolo dell'f1-score si era pensato che fosse ragionevole assumere questo punto come massimo locale dell'f1 stesso. Analizzando tuttavia dei casi limite, si è giunti alla conclusione che fosse preferibile eseguire un calcolo preciso dello score, onde evitare perdite di generalità o errori di assunzione.

5.7.3 Individuazione delle features

In conclusione, l'individuazione di queste 50 features è avvenuta applicando la *Recursive Feature Elimination, RFE* [14]. Questo metodo allena ricorsivamente un modello di classificazione - nel nostro caso la random forest a parametri ottimizzati di cui sopra - e di volta in volta elimina la colonna che totalizza la minor *feature importance*, ovvero che spiega in misura minore la variabile di classificazione del dataset. Eliminando un ridotto numero di colonne per step, la RFE cerca di escludere collinearità e dipendenze tra le features (che potrebbero sopravvivere anche a seguito del processo di analisi della *feature correlation* eseguito in precedenza). Il processo si ripete fino ad ottenere il numero di colonne indicato nei parametri, e - nel caso in analisi - suggerito dallo step precedente. Una via alternativa per giungere al medesimo risultato potrebbe essere quella di saltare lo step descritto al paragrafo precedente, implementando direttamente la RFE con cross-validazione sul numero di features da selezionare (funzione RFECV della medesima libreria Python). Come sempre in questi casi, il processo escluderebbe totalmente l'ipotesi di overfitting - scongiurata comunque in parte dal processo di suddivisione nei tre dataset analizzato sopra - ma prolungherebbe troppo i tempi di computazione, per cui si è optato per la prima via.

Al termine di questa operazione, il numero delle colonne è stato ridotto su tutti e tre i sottoinsiemi di dati. Per fini che si renderanno chiari più avanti, si è scelto di eseguire un'ultima manipolazione sulle colonne del dataset, mirata a raggrupparle ulteriormente in termini di informazione fornita.

5.7.4 Creazione delle "*macro-features*"

Come accennato sopra, prima di procedere al clustering del dataset si è scelto di effettuare un raggruppamento delle colonne dello stesso in quelle che sono state chiamate *macro-features*: in sostanza, le colonne sono state raggruppate in gruppi in base al tipo di informazione fornita. Sono state individuate 9 categorie:

- composizione della famiglia
- grado di propensione tecnologica del cliente
- caratteristiche climatiche della zona di residenza
- situazione economica del cliente in base alle caratteristiche dell'abitazione/zona di residenza
- consumi e relativi costi bolletta
- valore del cliente dal punto di vista dell'affiliazione con l'azienda
- campagne informative e contatti cliente-azienda
- crediti e debiti in bolletta
- rinnovi contrattuali, anzianità del cliente

A questo punto, le informazioni riportate dalle colonne legate alle singole macro categorie sono state proiettate volta per volta in un'unica nuova colonna, riportante il nome della macro categoria stessa. Il metodo utilizzato per giungere a questo risultato è la PCA (*Principal Component Analysis*): come si apprende dalle guide consultabili in [15] e [16], la PCA è uno strumento di linearizzazione impiegato nella decomposizione di un dataset multivariato (ovvero con un numero di colonne superiore a 1) in un insieme di *componenti* ortogonali tra loro, le quali rappresentino singolarmente il massimo quantitativo di varianza possibile. Ciò significa che le nuove variabili - o componenti - saranno organizzate in modo discendente per percentuale di varianza spiegata, con la componente numero 1 che spiega la maggiore quantità di varianza, la componente numero 2 che ne spiega un po' meno e così via fino all'n-esima componente. Trattandosi, poi, di una trasformazione lineare, ogni colonna coinvolta nella PCA influenza ognuna delle n componenti in maniera diversa. Il grado di influenza può essere studiato prendendo in considerazione la *matrice delle componenti degli autovalori* prodotta dalla PCA (in Python,

*pca.components_*): per ogni riga, corrispondente ad una componente, le entrate della matrice indicano i valori degli autovalori corrispondenti alle colonne cinvolte nella PCA. Prendendo in considerazione questi valori in modulo (*absolute value*), si può concludere quali features influenzino maggiormente la componente principale considerata. Prendendo poi gli autovalori con segno, si deduce anche se la colonna influenzi in modo diretto o indiretto la componente.

In base al funzionamento di questo procedimento, dunque, si è scelto di eseguire 9 PCA a singola componente (le colonne corrispondenti ad una macro categoria sono state, cioè, proiettate su un nuovo spazio unidimensionale). Si precisa che, per caratteristica propria della PCA in Python, si è resa necessaria una scalarizzazione preliminare a varianza unitaria delle features.

A questo punto, al fine di poter maneggiare meglio queste nuove colonne del dataset, si è presa in considerazione la matrice degli autovalori analizzata sopra, singolarmente per ogni PCA. Nel dettaglio, ogni qual volta venisse generata una nuova colonna del dataset, lo *user* del programma poteva visualizzare in modo ordinato quali fossero le componenti - ovvero le colonne originali - che maggiormente la influenzassero, e se lo facessero direttamente o inversamente.

In questo modo, per esempio, si è scoperto che la macro categoria corrispondente al livello tecnologico del cliente è influenzata fortemente dal fatto che il cliente abbia o meno il cellulare (anche se - stranamente - in modo indiretto...) e il telefono, mentre il suo valore come affiliato della società è tanto maggiore quanto più longevo è il suo contratto, come ragionevolmente ci si aspettava.

Questa operazione è giustificata dalla volontà di entrare in possesso di informazioni *user friendly* che aiutassero a visualizzare meglio i risultati dello step successivo: il clustering dei dati. In seguito, infatti, i risultati di questa operazione verranno plottati in funzione di combinazioni di queste macrofeatures, e la conoscenza delle stesse ci permetterà di trarre conclusioni sulla

composizione dei gruppi prodotti dal clustering.

Come sempre, l'operazione è stata eseguita in parallelo sugli insiemi di train, validation e test.

5.8 *Clustering*

Ecco ora che il dataset è pronto per affrontare il primo vero step di innovazione del modello rispetto a quello precedentemente esistente.

L'idea è quella di combinare una fase di unsupervised learning, rappresentata dal clustering, con i modelli supervised già implementati in passato. Lo spunto è stato suggerito in parte dallo studio di casi di letteratura. Come ad esempio suggeriscono Huang e Kechadi in [19], si presume che clienti aventi caratteristiche simili possano essere più predisposti a comportarsi in modo simile nel futuro. Da qui, allenare un modello di classificazione su dei dati semi-classificati, ovvero che abbiano un'etichetta comune (come quella fornita dal clustering), dovrebbe fornire risultati migliori che non allenandolo sull'intero insieme di train. Alla stessa conclusione sono giunti Rajamohamed e Manokaran, che in [22] hanno dimostrato come la pre-suddivisione del dataset in cluster possa migliorare le prestazioni del classificatore impiegato in seguito. Analogo discorso viene affrontato da Bose e Chen in [21], che tuttavia si pongono una ulteriore domanda: in che punto del modello l'implementazione del clustering conduce a risultati migliori? Conviene, cioè, dividere i dati in cluster e a quel punto implementare classificatori separati sui vari gruppi (come hanno fatto gli autori precedenti), oppure forse il prodotto del clustering interpretato come nuova colonna del dataset aiuta meglio il classificatore ad allenarsi? La risposta che gli autori hanno fornito è che, nel caso delle analisi da loro svolte, i risultati migliori si sono ottenuti utilizzando il secondo metodo, ovvero includendo l'etichetta del clustering come ulteriore colonna del dataset. Diversamente, gli studi compiuti da Hung, Yen e Wang giungono alla conclusione opposta, affermando che nel loro caso, in generale, le performance dei modelli predittivi allenati sulle singole clas-

si di clienti fornissero risultati migliori di quelli allenati sull'intero dataset (cfr. [20]). Data quindi l'ambivalenza dei risultati, riconducibile sicuramente alla natura differente dei dati, alla relativa semplicità dei modelli adottati (*decision tree* con *boosting* nel caso di [21], senza in [20]), e alle differenti metriche impiegate nella valutazione dei risultati (*top decile lift* [23], *hit ratio* o precision), per gli esperimenti a seguire si è scelto di mettere nuovamente a confronto le due ipotesi. Si anticipa che i risultati migliori, in questo caso, si sono ottenuti dividendo il dataset in cluster.

Quello su cui tuttavia i paper concordano, e che è stato verificato in corso d'opera attraverso un confronto tra le principali tecniche conosciute, è che il tipo di clustering che meglio si confà a questo tipo di problemi è il *k-Means Clustering* (documentazione alla pagina [17] e [18]). Questo mira a minimizzare la varianza intra-gruppo (o *inertia*, o *within-cluster sum-of-squares*):

$$\sum_{i=0}^{N} \min_{\mu_j \in C} \left(||x_i - \mu_j||^2 \right)$$

dove N è la cardinalità del dataset (il numero di righe), C è l'insieme dei cluster, con cardinalità n, e $\{\mu_j\}_{j=1}^{n}$ sono i punti medi, o *centroidi*, dei clusters. Questi centroidi, che identificano in modo univoco i clusters, solitamente non sono record del dataset ma vivono nello stesso spazio geometrico. L'algoritmo segue una procedura iterativa: inizialmente crea n partizioni (con n numero di clusters assegnato dall'utente) e assegna i punti d'ingresso a ogni partizione, o casualmente o usando alcune informazioni euristiche, quindi calcola il centroide di ogni gruppo; costruisce in seguito una nuova partizione associando ogni punto d'ingresso al gruppo il cui centroide è più vicino ad esso; infine vengono ricalcolati i centroidi per i nuovi gruppi e così via, finché l'algoritmo non converge (da [18]).

Il pregio di questo algoritmo è quello di scalare bene in funzione del numero di righe del dataset, e di convergere velocemente a meno di rari casi. Inoltre, l'inerzia dei cluster è un buon indicatore della *coerenza* interna degli stessi; essa tuttavia soffre di alcuni problemi:

- il calcolo dell'inerzia parte dall'assunzione che i clusters siano convessi ed isotropici (ovvero distribuiti uniformemente in tutte le direzioni), per questa ragione non è un buon indicatore in caso di cluster allungati e/o dalla forma irregolare. Per ovviare al problema, il processo di clusterizzazione verrà in seguito verificato tramite il calcolo di metriche specifiche per il caso in analisi;

- la distanza euclidea soffre del *curse of dimensionality*: cresce all'aumentare della dimensionalità dello spazio in cui si lavora, e l'inerzia è ua misura non normalizzata per cui il problema potrebbe non convergere. Questo problema può essere scongiurato applicando una riduzione della dimensione dello spazio, cosa che è stata fatta in precedenza (vedi paragrafo *Feature selection*);

- l'algoritmo converge quasi sempre, ma spesso il risultato può essere un minimo locale, non globale. Per ovviare a questo problema occorrerebbe ripetere più volte il processo variando l'inizializzazione dei centroidi. In alternativa, l'algoritmo di Python è stato implementato con l'aggiunta del parametro *init='k-means++'* che sceglie centroidi molto distanti tra loro in modo tale da sperare in una convergenza migliore dell'algoritmo. Questa opzione è stata scelta per il modello a seguire.

Di seguito si descrive nel dettaglio il procedimento affrontato.

5.8.1 Individuazione del numero ottimale di clusters

La prima operazione svolta è stata l'individuazione del numero ottimale di cluster in cui suddividere il modello. Trattandosi infatti di unsupervised learning vero e proprio, non si era in possesso di alcuna informazione sulla struttura dei dati, né di conseguenza di alcuna possibile segmentazione dei clienti.

Per questa ragione, i dati sono stati sottoposti a quello che prende il nome di *elbow method*: attraverso una pipeline costituita da una scalarizzazione

preliminare dei dati e da un k-means in cui di volta in volta viene aggiornato il numero di cluster, si valuta la bontà del clustering stesso attraverso il calcolo dell'inerzia. Plottando poi il grafico delle inerzie in funzione del numero di cluster, ci si aspetta che esso formi un vero e proprio gomito in coincidenza del numero ottimale di cluster da usare.

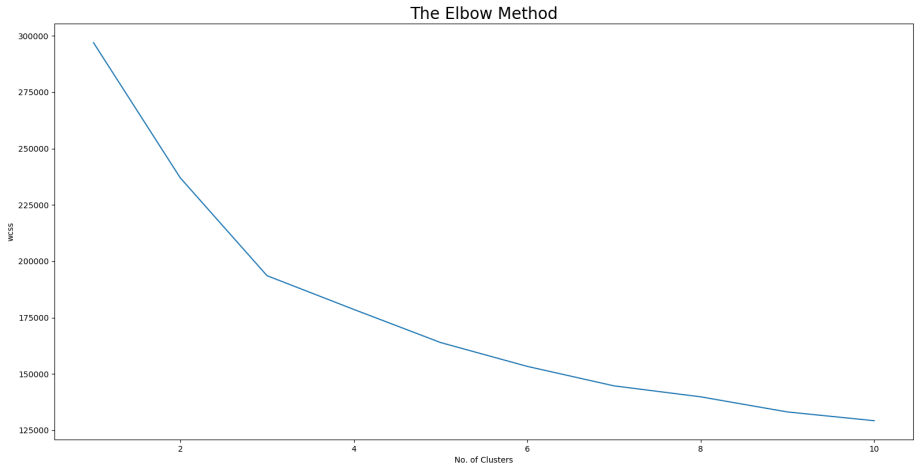


Figura 5.6: Rappresentazione dell'*elbow method*: si vede chiaramente che 3 sia il numero ottimale di cluster in cui suddividere il dataset in analisi

Il metodo ha avuto successo: come si osserva dalla Figura 5.6, in coincidenza di $\#cluster = 3$ il grafico subisce un netto cambio di inclinazione, da cui si deduce che i dati possano essere sensatamente divisi in 3 sottoinsiemi.

5.8.2 Applicazione e valutazione del clustering

Trovato il numero di cluster da utilizzare, si può dunque procedere all'implementazione del clustering vero e proprio. Riutilizzando la struttura della pipeline descritta in precedenza (*scaler* più k-means), l'insieme dei dati è sttao segmentato in tre sottogruppi. Questa volta la valutazione delle prestazioni è avvenuta attraverso il calcolo di tre score differenti, scelti *ad hoc*

in modo tale che non richiedessero la presenza di un'etichetta di verifica, in quanto ovviamente mancante:

- *Silhouette score.* Il coefficiente di silhouette si definisce per ogni record del dataset e si compone a sua volta di due score: la distanza media tra il punto e tutti gli altri punti dello stesso cluster (a), e la distanza media tra il punto e tutti gli altri punti nel cluster più vicino (b). Dati questi valori, il coefficiente di silhouette per il punto in questione si calcola come:
$$s = \frac{b - a}{\max(a, b)}$$
Il silhouette score del modello è definito come la media tra tutti i coefficienti di silhouette così calcolati. Questo score è sempre compreso tra -1 e 1, dove -1 indica un clustering fallimentare, mentre valori prossimi a 1 sono associati a cluster molto densi e ben separati. Punteggi intorno allo 0 indicano sovrapposizione tra i cluster. Generalmente punteggi alti si ottengono per cluster convessi: per questa ragione un k-means generalmente non produrrà cluster con silhouette score massimi.

- *Calinski-Harabasz score.* Questo indice è definito come il rapporto tra la somma della dispersione intra-cluster e la dispersione totale inter-cluster, dove con dispersione si intende la somma delle distanze tra i punti al quadrato. Il punteggio è tanto più alto quanto più densi e meglio separati sono i clusters. Analogamente al silhouette score, cluster convessi generano punteggi maggiori, mentre negli altri casi ci si assesterà su valori medi.

- *Davies-Bouldin score.* Questo indice rappresenta la *somiglianza* media tra clusters, dove la somiglianza è una misura che mette a confronto la distanza tra cluster con la loro dimensione. Il valore minimo di questo score è 0, e valori prossimi allo zero indicano una partizione molto buona dei dati. Il calcolo di questo score è inoltre molto più veloce di quello del silhouette score.

```
11:13:40: Evaluating clusters...
Silhouette score is  0.23236186689807212
Calinski-Harabasz score is  45800.02563891376
Davies-Bouldin score is  1.6443583497907184
```

Figura 5.7: Scores calcolati per il clustering come descritto sopra.

In Figura 5.7 si riportano i risultati ottenuti. Come si può osservare, essi si collocano in una fascia intermedia: sebbene non siano del tutto negativi (il silhouette score è maggiore di zero, il Calinski-Harabasz molto alto), il che suggerisce una qualche sensatezza dell'applicazione di un clustering ai dati, d'altro canto la loro vicinanza ai casi limite (silhouette prossimo più a zero che a uno, Davies-Bouldin maggiore di 1) suggerisce che i cluster sono in parte sovrapposti e non del tutto ben formati (non sempre convessi, ad esempio). Questi risultati sono comunque in linea con la natura del problema: trattandosi infatti di record totalmente casuali, corrispondenti a clienti reali e dunque disomogenei per comportamento, quella del clustering è in parte una forzatura analitica volta al fitting di un modello matematico all'interno di una situazione reale di difficile trattazione.

Ritenendo comunque soddisfacenti i risultati, si procede all'analisi della composizione dei cluster, nella speranza di trarne qualche tipo di informazione.

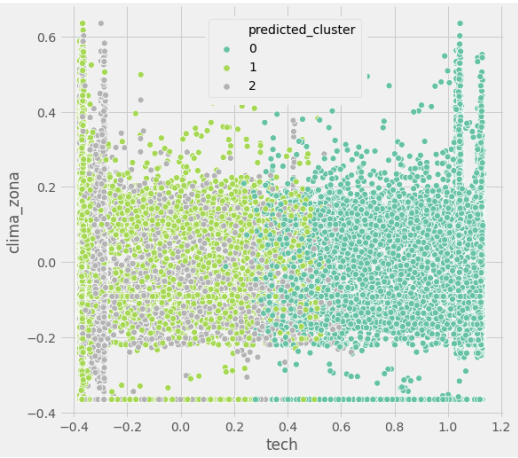
5.8.3 Analisi dei cluster

In questa fase ci si sofferma sull'analisi della composizione dei cluster. Questo sarà fondamentale nel trarre conclusioni sull'efficacia dell'introduzione di un modello unsupervised nella *pipeline* del modello: se infatti si dovessero individuare, come auspicato, delle caratteristiche ricorrenti tra i membri di uno stesso gruppo, il clustering assumerebbe un significato reale e tangibile nella classificazione dei clienti.

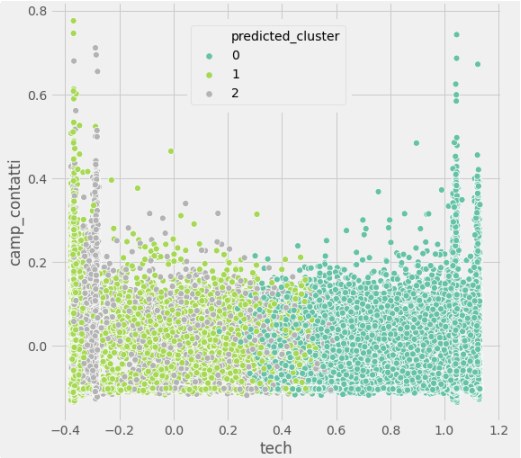
Questa analisi si svolgerà attraverso lo studio dei plot dei cluster in spazi bidimensionali costituiti da coppie di *macro-features*, quelle create al

paragrafo 5.4.7.

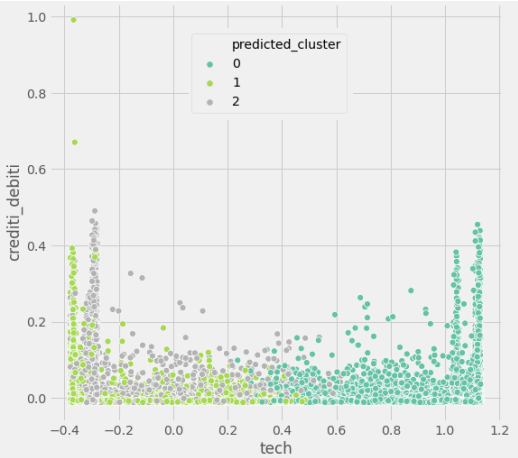
Di seguito si riporta una serie dei plot più significativi così generati, per poi stilare le conclusioni che se ne sono tratte.



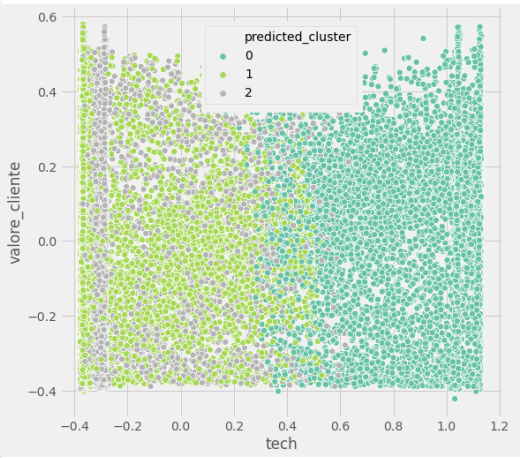
(a) Plot dei cluster in funzione di *tech* e *clima_ zona*.



(b) Plot dei cluster in funzione di *tech* e *camp_ contatti*.

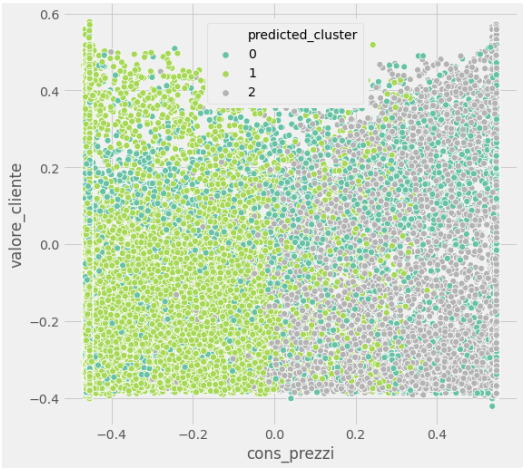


(c) Plot dei cluster in funzione di *tech* e *crediti_ debiti*.

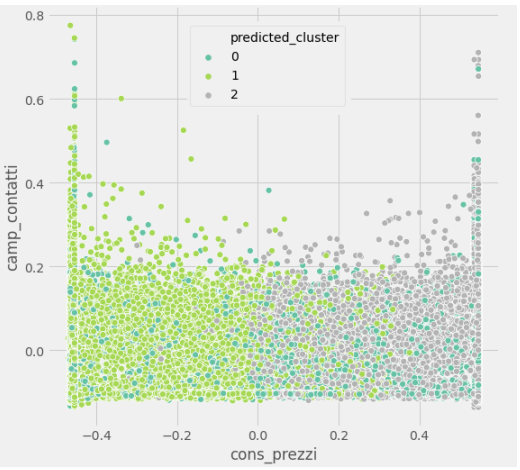


(d) Plot dei cluster in funzione di *tech* e *valore_ cliente*.

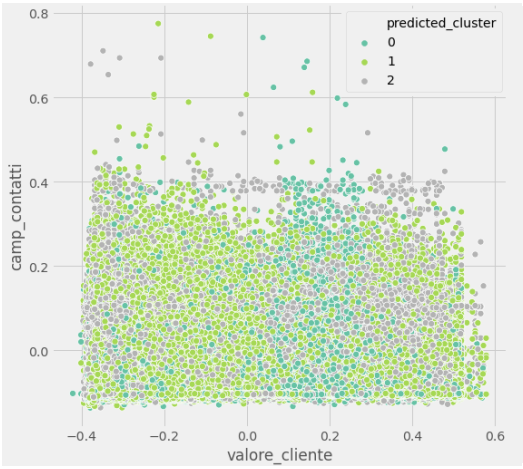
Quelli riportati sono solo alcuni dei grafici prodotti dalla proiezione dei cluster in funzione di combinazioni di macro-features, ma sicuramente sono significativi. Ecco cosa si può dedurre dalla loro analisi:



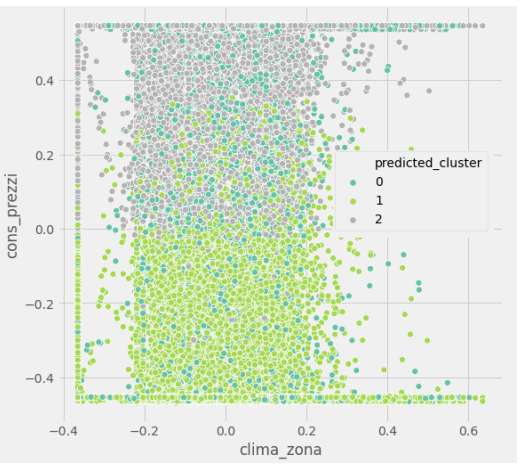
(e) Plot dei cluster in funzione di *cons_ prezzi* e *valore_ cliente*.



(f) Plot dei cluster in funzione di *cons_ prezzi* e *camp_ contatti*.



(g) Plot dei cluster in funzione di *valore_ cliente* e *cons_ prezzi*.



(h) Plot dei cluster in funzione di *clima_ zona* e *cons_ prezzi*.

Figura 5.8: Analisi dei clsuter.

- dai primi quattro grafici si osserva che il cluster 0 è associato a valori alti di grado tecnologico (a loro volta corrispondenti - in base a quanto visto dall'analisi delle macro-features - alla presenza di cellulari e telefoni), in netta contrapposizione con gli altri due cluster che invece si attestano su valori bassi di questa variabile;

- osservando il quinto e il sesto grafico, si nota che il livello di consumi e i conseguenti prezzi delle bollette dividono nettamente i cluster 1 e 2; in particolare il primo si attesta su valori bassi di questa variabile, il secondo su valori medio-alti. L'analisi delle macro-features conferma la composizione più intuitiva di questa variabile, per cui si può affermare che i clienti siano fortemente segmentati per fasce di consumi. Anche l'ultimo grafico conferma questa tesi, anche se questa volta i consumi si trovano sull'asse delle ascisse;

- l'immagine (g) farebbe pensare che il clsuter 0 sia costituito da clienti il cui valore si attesta per la maggior parte attorno a valori medi, mentre glia ltri due cluster sono costituiti da clienti dal comportamento più estremo.

In conclusione, dunque si può affermare con una certa sicurezza che: i clienti del cluster 0 sono quelli associati ad un alto grado tecnologico, mentre le loro caratteristiche in funzione delle altre features possono variare; i clienti del cluster 1 hanno in genere basso grado tecnologico e consumi in bolletta ridotti; infine i clienti del cluster 2, anch'essi associati ad un basso grado tecnologico, tendono a consumare di più e dunque ad avere bollette più cospicue.

5.8.4 Caso controllo: clustering solo sulle macro-features

Al termine del processo di clustering ci si è trovati a dover risolvere un dubbio: finora si era pensato ad eseguire le operazioni su tutto l'insieme di dati, ma cosa assicura che invece non fosse meglio trattare solo le informazioni corrispondenti alle colonne "macro-features" costruite in precedenza? Se infatti utilizzare tutte le colonne suggeriva un rischio di overfitting, dall'altro ridurre il processo alle sole macro-featutres avrebbe potuto comportare una grossa perdita di informazione.

Ai fini di una corretta analisi, dunque, il processo è stato rieseguito in parallelo in entrambe le casistiche, e di conseguenza anche tutta l'analisi

a seguire. Si anticipa già in questa fase che, se da un lato la valutazione del solo clustering potesse suggerire una confrontabilità dei due processi, in seguito - procedendo cioè nella fase di classificazione - è apparso chiaro come suddividere i dati solo in funzione delle nuove 9 colonne portasse ad una troppo consistente perdita di informazione, con conseguente cattivo design dei tre gruppi e dunque scarsa se non pessima performance dei classificatori.

L'analisi del modello proseguirà dunque mettendo in evidenza i risultati ottenuti dal clustering su tutto il dataset.

5.9 Classificazione

Si entra ora nel vivo della fase di supervised learning del modello: l'implementazione di modelli di classificazione.

In questa fase ci si è nuovamente trovati di fronte ad una serie di scelte:

- *il modello da usare*: la scelta sarebbe infatti potuta ricadere su modelli semplici, *ensemble*, o entrambi, e ancora, ad esempio, tra i modelli ensemble si sarebbe potuto distinguere tra modelli a *boosting* e modelli a *bagging*, e così via;
- *quali dati usare*: ancora una volta infatti occorreva scegliere se sfruttare tutta l'informazione disponibile, oppure se usare solo quella contenuta nelle macro-features;
- *in che ordine assemblare la fase di unsupervised learning con quella supervised*, ovvero: è più redditizio implementare il modello in parallelo sui tre cluster separati (idea iniziale), oppure utilizzare l'informazione del clustering come una nuova colonna da aggiungere al dataset per poi implementare il modello su tutti i dati?

A tutte queste domande si è cercato di dare una risposta implementando i vari modelli candidati in 4 combinazioni di dati in parallelo:

1. tre cluster separati, dati completi a disposizione
2. tre cluster separati, solo le colonne coincidenti con le macro-features
3. tutti i record insieme con informazione aggiuntiva del cluster come nuova colonna, dati completi a disposizione
4. tutti i record insieme con informazione aggiuntiva del cluster come nuova colonna, solo le colonne coincidenti con le macro-features

Di seguito si descrive la prassi comune seguita per i quattro casi, e in seguito si analizzeranno i risultati ottenuti.

5.9.1 Scelta e descrizione dei modelli di classificazione

Innanzitutto, in base anche alle analisi svolte dai colleghi per il modello preesistente, la scelta dei modelli da testare si è concentrata su classificatori *ensemble*, tralasciando quindi modelli semplici e ovviamente le reti neurali.

Prima di introdurre i modelli impiegati, occorre precisare la differenza che sussiste tra modelli ensemble di tipo *boosting* e di tipo *bagging*. Si ricorda intanto che con *modelli ensemble* si intendono tutti quei modelli di machine learning che implementano più volte il medesimo modello di classificazione semplice (generalmente un albero, ma non solo) ottimizzandone gli iperparametri al fine di ottenere il miglior risultato in funzione delle caratteristiche del dataset. Questo procedimento può essere affrontato in due modi: da un lato, come accennato in precedenza al paragrafo 4.5, i modelli di tipo *boosting* traggono informazione ricorsivamente, pesando di volta in volta più significativamente i record misclassificati dal modello implementato allo step precedente; d'altro canto, i modelli che utilizzano la logica del *bagging* (come la random forest descritta al paragrafo 5.7.1) implementano in parallelo lo stesso modello su sotto-dataset ottenuti da quello originale attraverso la tecnica del *bootstrap* (estrazione casuale con reinserimento), dopo di che ad ogni record viene assegnata la label della classe di appartenenza secondo la regola della maggioranza (la label che gli sia stata assegnata più volte sarà quella definitiva).

I modelli selezionati per il confronto sono:

- *Random Forest Classifier* [12]: già analizzata in precedenza, ricordiamo che: l'opzione *bootstrap* è a discrezione dell'utente, il che significa che la random forest può essere implementata fittando ogni volta il decision tree su tutto il dataset; introduce inoltre il parametro *max_features* con il quale si indica il numero massimo di colonne che l'albero dovrà di volta in volta utilizzare: in questo modo il modello pesa diversamente non solo i record ma anche le features, al fine di ottimizzare al massimo il risultato.

- *Bagging Classifier* [24]: algoritmo che stabilizza la varianza data dall'effetto *black box* del suo estimatore base, il *decision tree*, introducendo randomizzazione grazie all'impiego della tecnica del bagging.

- *AdaBoost Classifier* [25]: algoritmo che implementa la tecnica del boosting al suo estimatore base (generalmente decision tree).

- *Gradient Boosting Classifier* [26]: la peculiarità di questo modello ensemble è che applica il boosting al decision tree di base offrendo la possibilità di impiegare funzioni obiettivo diverse.

- *ExtraTree Classifier* [27]: o *extremely randomized tree*, implementa un nuovo metodo per la selezione delle features da utilizzare di volta in volta per il decision tree.

- *XGBoost Classifier*[28]: unico modello che non appartenga alla libreria *scikit-learn* di Python, oltre all'LGBM, si tratta di un upgrade del Gradient Boosting; in particolare l'XGB effettua calcoli più precisi del gradiente della funzione obiettivo, e la stabilizza grazie all'impiego di norme L1 e L2.

- *LightGBM Classifier* [10], per confronto con il modello attuale.

Al fine di implementare modelli ottimali, sulla falsariga di quanto eseguito per la *random forest* impiegata nella *feature selection*, per ognuno dei classificatori è stata implementata una *grid search CV* che vagliasse varie combinazioni di iperparametri e ne valutasse l'efficienza. Ad ogni step è stata impiegata la metrica f1-score per la valutazione delle *prediction*. Al termine di questa prima fase - ripetuta ovviamente per ognuna delle 4 combinazioni di dati di cui sopra - si è potuto procedere all'allenamento dei modelli.

5.9.2 Train dei modelli

A questo punto si è proceduto con l'allenamento dei modelli sugli insiemi di train, e con la loro valutazione su quelli di validation. Per ogni modello, il

classificatore è stato allenato e la *decision threshold* ottima è stata individuata tramite calcolo dell'f1 score (come descritto in precedenza nel paragrafo 5.7.2); a questo punto, le prediction sono state ricalcolate in funzione della nuova threshold. Inoltre, nell'ottica di avere a disposizione per la fase di test modelli allenati sul maggior insieme di dati possibili, e dunque nella speranza di ridurre ulteriormente la varianza del modello, l'allenamento - ma non la prediction - dei modelli è stato ripetuto anche su tutto l'insieme di train (ovvero sull'unione di train e validation).

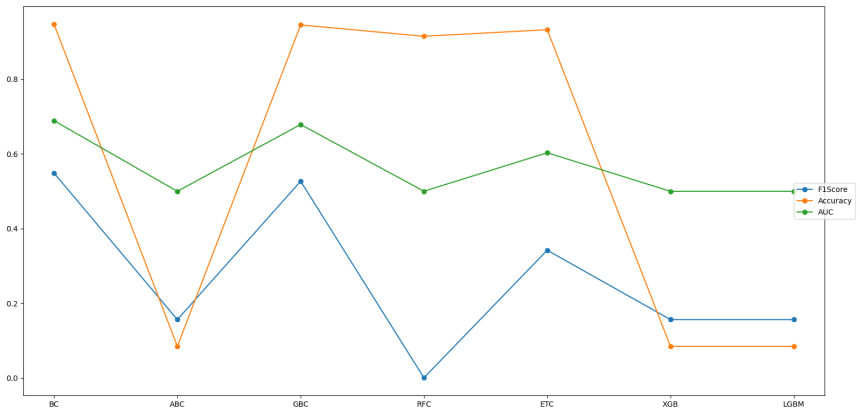


Figura 5.9: Esempio di plot degli scores dei modelli calcolati su un insieme di train: come si può notare, i modelli BC, GBC e ETC forniscono risultati molto migliori rispetto agli altri modelli, basandosi sul calcolo dell'f1-score.

A questo punto è stato eseguito un plot degli score dei modelli in funzione delle prediction calcolate sull'insieme di validation; in particolare i plot comprendono f1-score, accuracy score e AUC score (*Area Under the Curve*, dove con curva si intende la *ROC curve*). In Figura 5.9 si riporta un caso esemplificativo dei risultati, tralasciandone la descrizione dettagliata per il Capitolo 6. Come si intuisce da questo primo caso, si è constatato che i modelli BC, GBC e ETC fossero in quasi tutti i casi i migliori dal punto di vista dell'f1-score, che di fatto è la metrica di riferimento per questo lavoro.

Di conseguenza, per gli step successivi si è scelto di tenere in considerazione solo questi tre modelli.

5.9.3 Introduzione dei modelli *cost sensitive*

In questa fase del lavoro entra in gioco un nuovo elemento rispetto al modello attuale. Come già largamente illustrato in precedenza, il problema maggiore dei dati coi quali ci si trovava a lavorare era il forte carattere di *unbalancing* delle due classi. Varie tecniche finora erano state affrontate, riassumibili nelle categorie dell'*oversampling* dei dati e dell'attribuzione (nel contesto di modelli di classificazione) di *pesi diversi* ai record del dataset in funzione della disproporzione delle classi.

Questi strumenti da soli non hanno tuttavia risolto il problema dell'unbalancing, per cui ci si è trovati a dover pensare ad una nuova strategia. Dalla letteratura viene in aiuto una nuova classe di modelli, pensata appositamente per la classificazione dei dataset fortemente sbilanciati: si tratta dei modelli *cost sensitive*, la cui documentazione è reperibile alla pagina [30]. La principale novità di questi modelli rispetto a quelli standard è l'introduzione della cosiddetta *matrice di costo*: si tratta appunto di una matrice le cui entrate corrispondono a dei costi, figurati o meno a seconda del tema del dataset a cui vengono applicati, corrispondenti alla misclassificazione dei record. Nel dettaglio, si consideri un problema di classificazione binario come quello in analisi, e si costruisca una matrice 2x2 come da Figura 5.10: le entrate della matrice sono dunque i costi associati ai falsi positivi e ai falsi negativi (rispettivamente C_{FP_i} e C_{FN_i}) e i costi per le corrette classificazioni (i C_{TP_i} e C_{TN_i}, solitamente nulli).

Una volta costruita questa matrice (e come vedremo in seguito, la sfida principale di questi modelli risiede proprio nel calcolo di costi adatti), essa verrà affiancata al dataset nell'allenamento dei modelli di classificazione. La libreria è dotata infatti di modelli molto simili a quelli trattati in precedenza, come il *decision tree*, la *random forest* e il *bagging classifier*, che operano con le logiche dei corrispettivi classici, ma la cui funzione di ottimizzazione non

	Actual Positive $y_i = 1$	Actual Negative $y_i = 0$
Predicted Positive $c_i = 1$	C_{TP_i}	C_{FP_i}
Predicted Negative $c_i = 0$	C_{FN_i}	C_{TN_i}

Figura 5.10: Rappresentazione della costruzione della matrice di costo per un problema di classificazione binario. Img credits: [30]

si baserà più sul calcolo di metriche standard - come era stato l'f1-score nei casi di cui sopra. Questa volta, il modello verrà ottimizzato sulla base del calcolo di una nuova metrica, i *savings*. La logica di questa metrica è intuitivamente spiegata dal suo nome: ci si chiede infatti quale sia il *risparmio* derivante dall'applicazione di un modello cost sensitive ottimizzato, rispetto alla classificazione degli score sulla base di un modello *naive*, semplice, non ottimizzato in funzione della matrice di costo. Nello specifico, i savings si calcoleranno come la differenza tra la *perdita in costo* derivante dall'impiego del suddetto modello semplice e quella conseguente all'applicazione del modello cost sensitive, che ci si augura sia minima.

Introdotta dunque la logica sottostante all'impiego di questo pacchetto di modelli, quello che resta da illustrare è il procedimento scelto per la costruzione della matrice di costo. Dal momento che quello della previsione del churn è un problema con un risvolto economico tangibile, la costruzione dei costi di misclassificazione si è basata sulla risposta alle seguenti due domande:

- a quanto ammonterebbe la perdita per la società nel momento in cui non fosse predetto l'abbandono di un cliente?
- a quanto ammonterebbe la perdita per la società nel momento in cui un cliente venisse etichettato come churner ma in ultima analisi non abbandonasse la società?

La risposta a queste domande fornirà rispettivamente il costo per i *false positive* e per i *false negative*.

Come accennato in precedenza, il calcolo della matrice di costo non è affatto banale. L'idea sottostante è che la matrice debba forzare i modelli a favorire i casi positivi, in minoranza, rispetto a quelli negativi. Occorre tuttavia trovare un trade-off tra lo sbilanciamento da ognuna delle due parti: anche assegnare troppi record alla classe dei churners non sarebbe infatti ideale, in quanto i costi di *caring* associati alla ritenzione di un possibile churner non sono trascurabili. Un altro punto controverso è l'orizzonte temporale in funzione del quale calcolare il costo, o il guadagno, derivante dalla perdita o ritenzione di un cliente: alcune scuole di pensiero suggeriscono che di basare queste quantità sul comportamento medio nell'arco di 12 mesi; altre invece tendono a preferire una proiezione più lunga, basata sulla stima di permanenza del cliente nell'azienda. Data la struttura temporale dei dati a disposizione, si preannuncia che la scelta effettuata per questo lavoro di tesi è la prima.

Di seguito si analizza la struttura della matrice di costo per come è stata calcolata per il presente lavoro.

$$
\begin{aligned}
C_{TP_i} &= 0 \\
C_{TN_i} &= 0 \\
C_{FP_i} &= \text{costo\_caring\_medio} \\
C_{FN_i} &= \text{consumi\_12M}_i \cdot \text{prezzo/kWh}_i + \text{num\_VAS\_attivi}_i \cdot \text{valore\_medio\_VAS} + x_i(\text{VAS})
\end{aligned}
$$

dove

$$
\begin{aligned}
\text{valore\_medio\_VAS} &= 10 \\
\text{costo\_caring\_medio} &= 20 \\
x_i(\text{VAS}) &= \begin{cases} 0 & \text{if num\_VAS\_attivi\_last6M}_i = 0 \\ 0.65 \cdot \text{valore\_medio\_VAS}, & \text{if num\_VAS\_attivi\_last6M}_i \neq 0 \end{cases}
\end{aligned}
$$

Dunque, come accennato, i costi per le classificazioni corrette sono nulli; il costo dei falsi positivi è rappresentato dal costo medio di un'azione di

caring sul cliente; infine, il costo per la misclassificazione dei casi positivi è articolato in tre parti: esso è infatti la somma del profitto che il cliente avrebbe prodotto durante l'anno, addizionato del profitto derivante dalle sue eventuali sottoscrizioni a *Value Added Services*, e del profitto variabile derivante dalla sua sottoscrizione a nuovi servizi. Quest'ultimo fattore, in particolare, è legato alla *fidelity* del cliente (è infatti zero se il cliente non aveva sottoscritto VAS negli ultimi 6 mesi) e ad un fattore di 0.65 che indica la probabilità calcolata che un cliente storico della società sia candidabile al *cross selling*, ovvero che sottoscriva nuovi servizi (come testimoniato da Ayat Shukairy nell'articolo "Customer Acquisition Vs.Retention Costs - Statistics And Trends" [31], percentuale per altro molto maggiore rispetto al 15% medio calcolato per i nuovi clienti).

Figura 5.11: Un nuovo cliente ha molta meno probabilità di effettuare *cross buying* all'interno dell'azienda rispetto ad uno già affiliato. Img credits: [31]

Come deducibile dalle formule, i costi non saranno uguali per ogni cliente, dipendendo infatti da fattori specifici e variabili come i consumi e i relativi costi in bolletta. Questo implica che a seconda del comportamento del cliente, il modello sarà incentivato a porre più o meno attenzione alla sua corretta classificazione, favorendo ad esempio clienti che portano un vantaggio maggiore all'azienda o clienti con un'affiliazione di durata maggiore.

Si precisa che i valori costanti per il costo medio di un'azione di caring e del valore medio di un VAS sono stati stimati in base a informazioni for-

nite dal settore marketing. Questi valori, da intendersi in Euro, non sono tuttavia da considerarsi precisi: i dati in possesso non permettevano infatti di effettuare un calcolo esatto, ma piuttosto le stime sono state fondate sull'esperienza nel campo maturata dai data scientist con la collaborazione dei quali questo lavoro è stato svolto. Come vedremo in seguito, questo - insieme alla sperimentalità della costruzione parametrica della matrice - potrebbe rappresentare un problema per l'efficacia della matrice e dunque dei modelli cost sensitive nel complesso. La risposta arriverà in fase di studio dei risultati.

Introdotta la logica dei modelli cost sensitive, si va ora ad illustrare l'uso che se ne è fatto all'interno del lavoro di tesi. Questo in particolare si è manifestato in due aspetti:

1. *thresholding optimization* dei modelli, ovvero ottimizzazione dei modelli di classificazione standard in funzione della nuova metrica dei *savings*;
2. implementazione del *cost sensitive decision tree classifier*.

Thresholding optimization dei modelli

In questa fase si è scelto di mostrare le potenzialità dei modelli cost sensitive mettendo in luce come, se osservati dal punto di vista di una metrica costruita *ad hoc* per i modelli unbalanced, anche i più promettenti modelli di classificazione standard possano non eccellere nella classificazione del dataset.

Per fare questo si è ricorsi al modello *thresholding optimization*: si tratta di una funzione che prende in input le prediction calcolate da modelli di classificazione standard sulla base dei dati di train, e le aggiusta ricalcolando la decision threshold ottimizzandola in funzione del calcolo dei savings [32].

Eseguita questa operazione per i tre modelli selezionati dalla fase precedente, è stato dunque eseguito un plot che confrontasse f1-score e savings dei tre modelli e della loro ottimizzazione in funzione dei savings (a titolo esemplificativo, si veda Figura 5.12).

Cost sensitive decision tree classifier

A questo punto al dataset è stato applicato un modello di classificazione cost sensitive, il *cost sensitive decision tree classifier*. Si tratta del corrispettivo del decision tree della libreria *Scikit-Learn*, con la differenza che qua la funzione obbiettivo punta a massimizzare i savings piuttosto che l'f1-score. La scelta è ricaduta solo su questo modello (per altro semplice e non ensemble) in quanto la libreria presentava parecchie incompletezze nel codice sorgente, e adattarla anche solo per i primi modelli e metriche ha richiesto parecchio tempo e competenze di programmazione. Uno studio più approfondito delle potenzialità di questi modelli sarà centrale nello sviluppo futuro del lavoro.

Allenato dunque il modello, i suoi valori di savings e di f1-score sono stati calcolati, e il loro plot è stato affiancato a quelli di cui sopra. Di seguito, la Figura 5.12 mostra i risultati nel suo complesso.

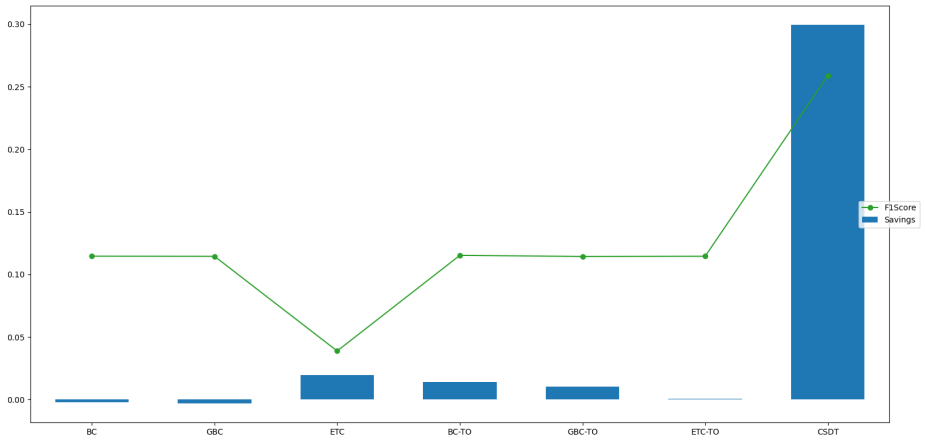


Figura 5.12: Nel grafico esemplificativo si può notare come la thresholding optimization dei primi due modelli abbia totalizzato savings score maggiori dei modelli non ottimizzati. Come verrà discusso nel prossimo capitolo, tuttavia, questi modelli non hanno sempre sortito l'effetto desiderato.

5.9.4 Test dei modelli

Si può dunque procedere con la fase finale dell'iter di classificazione: il test dei modelli allenati in precedenza, sfruttando l'ultima parte di dati, X_test. Ancora una volta questa fase avviene in parallelo per tutte e quattro le combinazioni di dati trattate in precedenza.

Il test dei modelli consiste nel calcolo delle prediction in funzione degli insiemi di test usando i modelli allenati in precedenza sui dati di train, e sulla loro valutazione attraverso le due metriche di cui sopra: f1-score e savings. Ancora una volta i risultati sono stati plottati per confronto in modo analogo a quanto riportato in Figura 5.12.

Nel Capitolo successivo si analizzano nel dettaglio i risultati ottenuti e si traggono le conclusioni su quanto osservato.

Capitolo 6

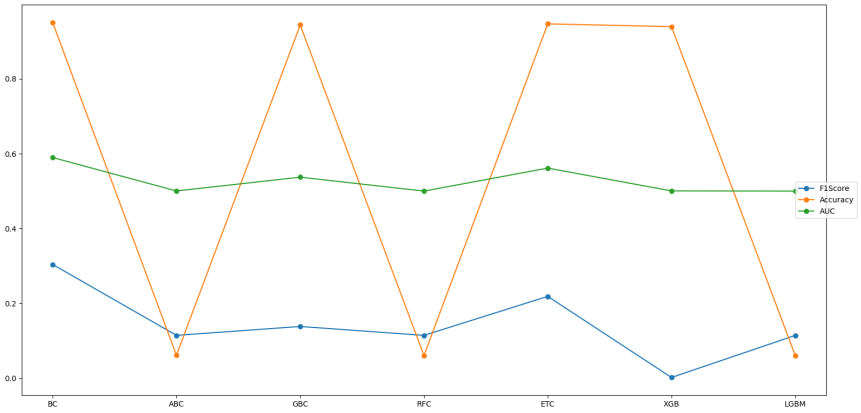
Risultati ottenuti

Per fini di completezza, il presente Capitolo è organizzato come segue: si analizzano separatamente i risultati ottenuti per ognuna delle quattro combinazioni di dati di cui sopra; in seguito si traggono le conclusioni su quanto osservando, in termini del o dei modelli che hanno meglio rappresentato i dati a disposizione.

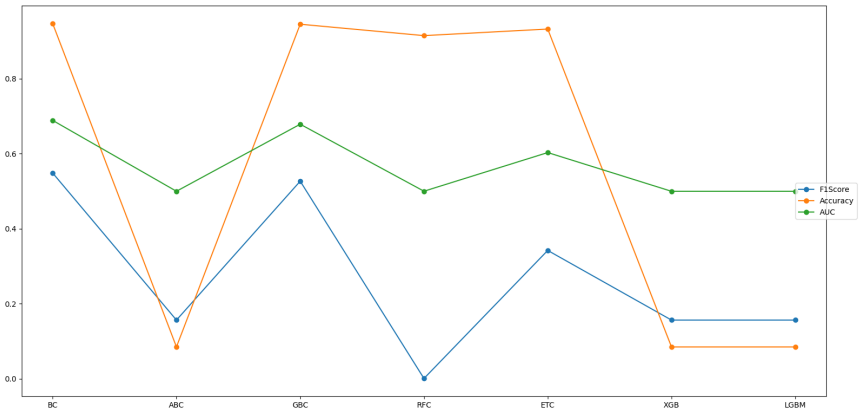
6.1 Cluster separati, classificazione su tutti i dati

La prima combinazione di dati trattata è stata quella in cui i dati venivano separati in funzione della label prodotta dal clustering, ma sfruttando comunque tutta l'informazione prodotta dalle colonne selezionate in fase di feature selection. I modelli di classificazione sono stati quindi allenati separatamente e in parallelo sui tre cluster (quindi anche la fase di individuazione della migliore combinazione di iper parametri è stata customizzata per ogni cluster), per poi venire testati ancora separatamente sui rispettivi cluster di test.

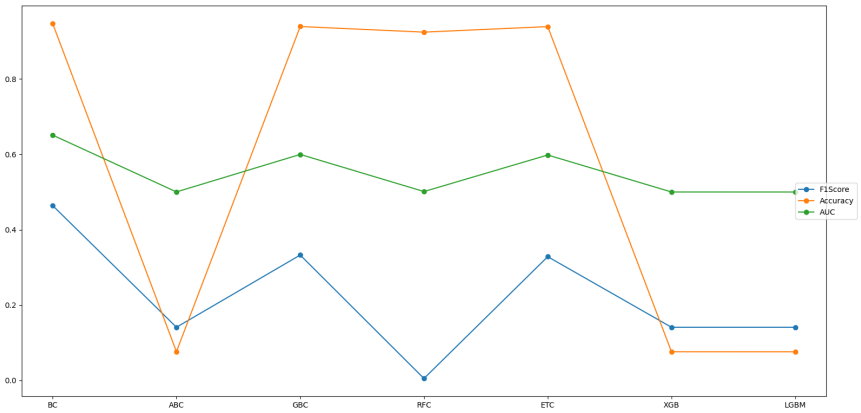
Di seguito si riportano i plot dei risultati e se ne illustra il significato.



(a) Allenamento dei modelli di classificazione sul cluster 0.



(b) Allenamento dei modelli di classificazione sul cluster 1.



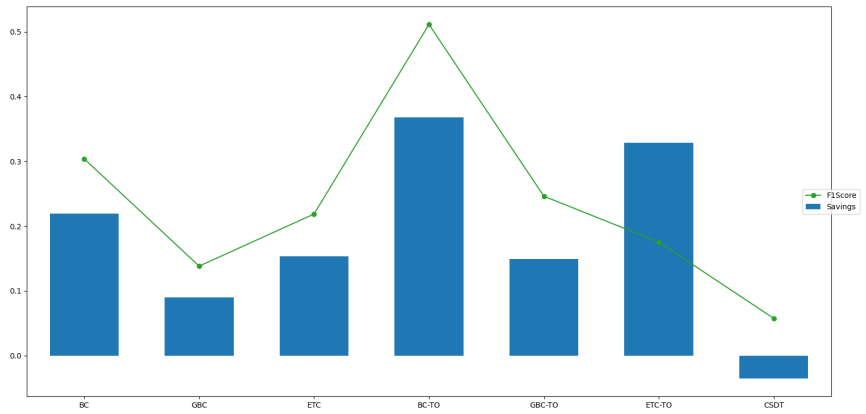
(c) Allenamento dei modelli di classificazione sul cluster 2.

Figura 6.1: Plot degli score in legenda calcolati per i modelli di classificazione allenati separatamente sui tre cluster.

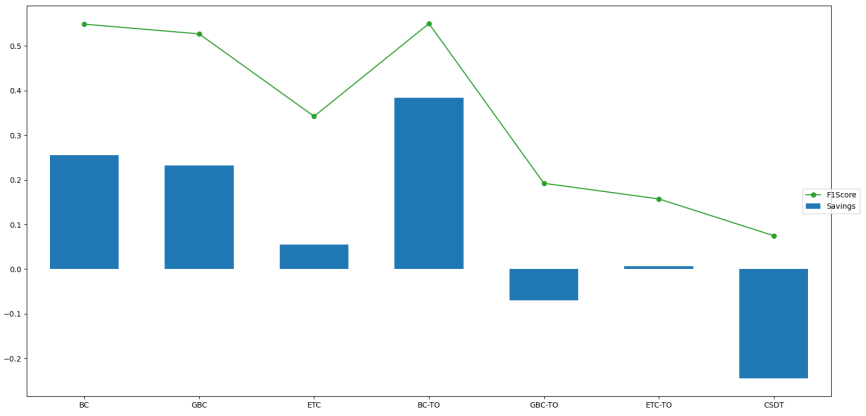
I plot in Figura 6.1 mostrano chiaramente che, come già anticipato, i tre modelli che meglio classificano il dataset di train (si ricorda che la metrica di riferimento è sempre l'f1-score) sono il Bagging Classifier, il Gradeint Boosting Classifier e l'ExtraTrees Classifier: per questa ragione d'ora in avanti saranno gli unici tre modelli ad essere considerati. Notiamo inoltre che il modello usato per il modello precedente, il LightGBM, ha prestazioni generalmente scarse: questo suggerisce che il presente lavoro è sulla buona strada per apportare miglioramenti nelle prestazioni rispetto al predecessore. In figura sono riportati gli score calcolati in termini di accuracy, f1-score e ROC: come si può notare, in alcuni casi prendere in considerazione l'accuracy score avrebbe potuto essere fuorviante, totalizzando essa punteggi molto alti anche laddove l'f1-score fosse invece basso (si veda ad esempio l'RFC nei cluster 1 e 2, figure (a) e (b) di 6.1).

A questo punto, selezionati i tre modelli più promettenti, si passa alla fase di thresholding optimization con calcolo dei savings, e all'allenamento del modello cost sensitive. I risultati sono osservabili in Figura 6.2. Quello che si può dedurre dalle immagini è che i modelli cost sensitive (la thresholding optimization e il CSDT) hanno fornito risultati contrastanti: se da un lato, ad esempio, il Bagging Classifier ottimizzato per i savings ha sempre fornito i risultati aspettati, totalizzando savings e f1-score maggiori rispetto al suo corrispettivo non ottimizzato, dall'altro il Gradient Boosting ha risposto in maniera non univoca al processo di ottimizzazione (migliorando i savings per il cluster 0 ma peggiorando negli altri due), così come l'ExtraTrees che nei primi due cluster ha migliorato le prestazioni ma è peggiorato nell'ultimo. Inoltre, il CSDT - che dovrebbe intuitivamente essere il modello che totalizza i savings più alti - ha invece totalizzato score negativi in tutti e tre i casi. Questo comportamento è probabilmente riconducibile ad una cattiva costruzione della matrice di costo: si tratta infatti, come accennato in precedenza, del punto chiave dell'implementazione di questi modelli, ma al contempo del più complesso. La costruzione parametrica della matrice si fonda su conoscenze profonde della natura economica del problema, la quale

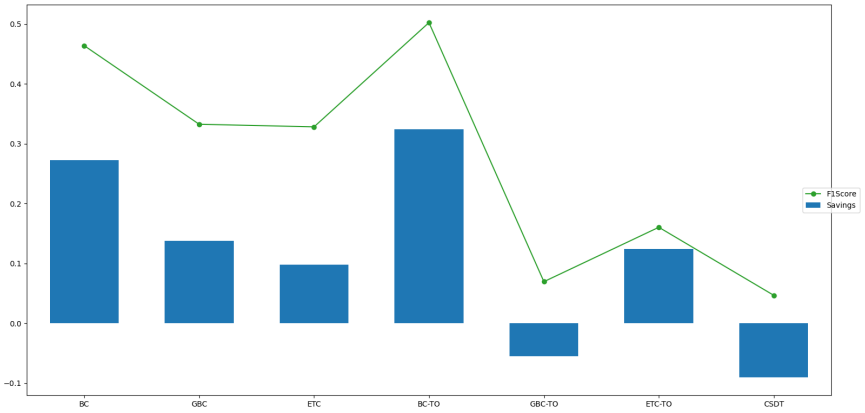
purtroppo non è stata soddisfatta dai dati in possesso. Se dunque questo tipo di modelli rimane promettente sulla carta nell'ottica della risoluzione dei problemi di classificazione su dataset sbilanciati, molto è il lavoro che rimane da fare nel contesto della loro comprensione, il quale esula per tempo e mezzi dagli scopi di questo lavoro di tesi.



(a) Score calcolati per il cluster 0.



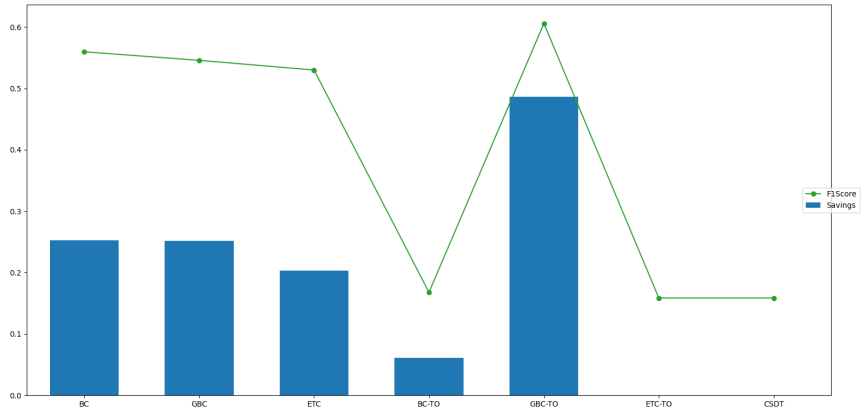
(b) Score calcolati per il cluster 1.



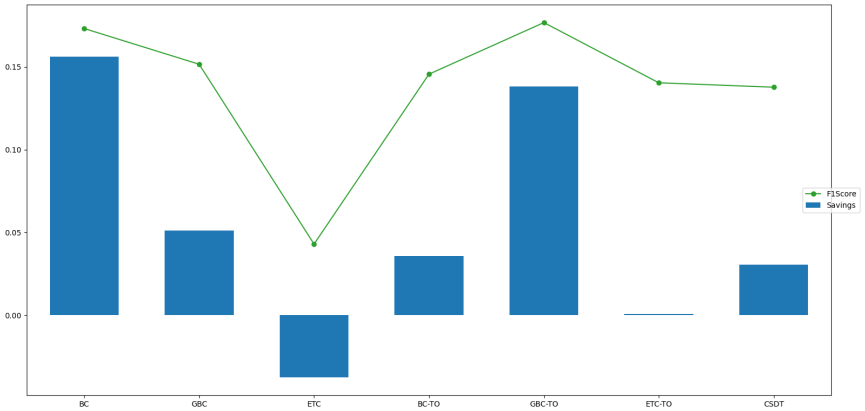
(c) Score calcolati per il cluster 2.

Figura 6.2: Plot di f1-score e savings calcolati, nei tre cluster, per i tre modelli migliori, la loro thresholding optimization e il cost sensitive decision tree.

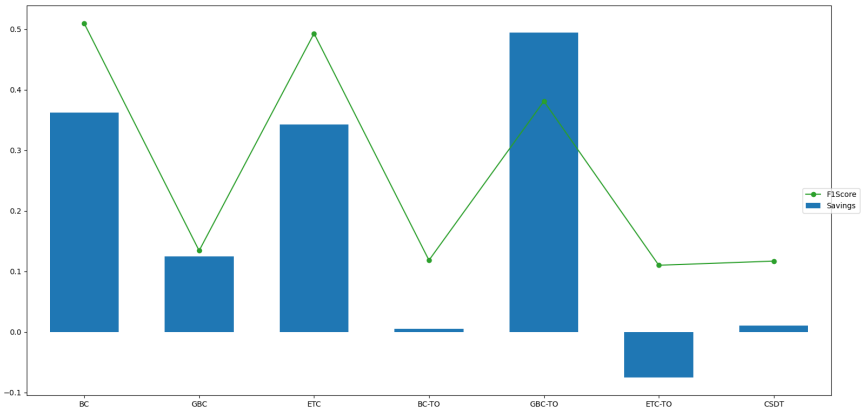
Passando infine alla fase di test dei modelli, ci si aspetta che il modello BC-TO totalizzi i punteggi migliori per tutti e tre i cluster, seguito dal suo corrispettivo non ottimizzato in funzione dei savings. I risultati ottenuti sono consultabili in Figura 6.3.



(a) Test dei modelli di classificazione sul cluster 0.



(b) Test dei modelli di classificazione sul cluster 1.



(c) Test dei modelli di classificazione sul cluster 2.

Figura 6.3: Plot dell'f1-score e dei savings totalizzati dai modelli in fase di test sui tre cluster.

Innanzitutto appare chiaro che le aspettative rispetto al comportamento dei modelli in fase di train non sono state del tutto soddisfatte: il BC-TO è di fatto, in fase di test, uno dei modelli peggiori, mentre il suo corrispettivo non ottimizzato si è comportato bene anche in fase di test, attestandosi tra i modelli migliori dal punto di vista dei savings e dell'f1-score. Inaspettatamente, inoltre, il GBC-TO, che in fase di train era quasi sempre stato il modello con gli score più bassi, ora in fase di test si attesta in tutti e tre i cluster sul podio della classificazione, quantomeno dal punto di vista dei savings score. Non sono migliorate invece le prestazioni del CSDT.

Quello che si può concludere, dunque, è:

- il modello BC-TO soffre probabilmente di overfitting: esso totalizza infatti buoni punteggi in fase di train, ma punteggi molto più bassi in fase di test, e questo comportamento è quasi sempre dovuto al fenomeno dell'overfitting sui dati di train;

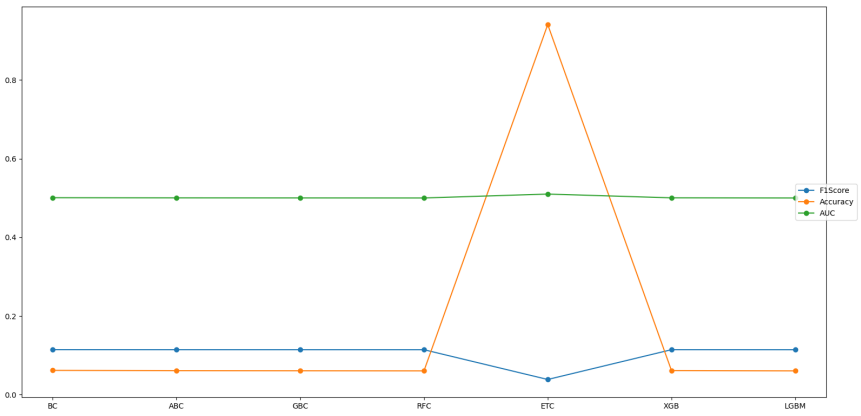
- il GBC-TO invece soffre probabilmente del problema inverso: esso infatti presenta alta varianza, per cui si potrebbe concludere che il modello sia troppo generalizzato rispetto al dataset su cui è stato allenato. Questo discorso è esteso anche al suo modello base, il GBC, anche se è affetto da varianza leggermente minore;

- l'ETC è infine un modello dal comportamento ambiguo: osservando solo il cluster 1 si potrebbe affermare che soffra di overfitting, ma concentrandosi sugli altri due cluster si noterebbe il comportamento esattamente opposto. Per quanto riguarda il suo corrispettivo ottimizzato, esso si comportava bene solo in un caso in fase di train, mentre totalizza score bassi in tutti i casi in fase di test, per cui lo si può escludere dal computo dei modelli migliori.

6.2 Cluster separati, classificazione sulle colonne corrispondenti alle macro-features

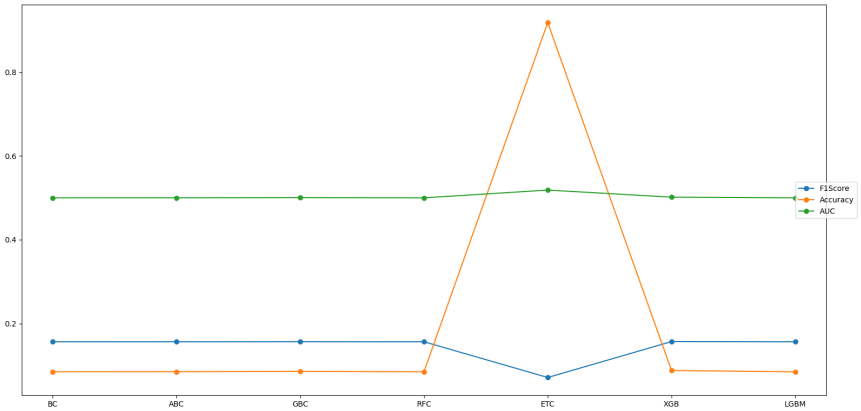
Ripetiamo l'analisi eseguita al paragrafo precedente per questa seconda combinazione di dati: qua il clustering si riflette ancora nella suddivisione dei dati in tre gruppi, ma questa volta la classificazione sfrutterà solo l'informazione legata alle colonne corrispondenti alle macro-features. L'idea di fondo è quella di verificare se questa riduzione della dimensionalità possa migliorare i casi di overfitting, o se al contrario la perdita di informazione prevalga sulle prestazioni dei modelli.

Come si può osservare da Figura 6.4, già in fase di train appare chiaro che la risposta a questa domanda protenda verso la seconda opzione: gli score totalizzati dai modelli sono infatti omogeneizzati verso il basso, con prestazioni decisamente peggiori rispetto a quelle del caso precedente.

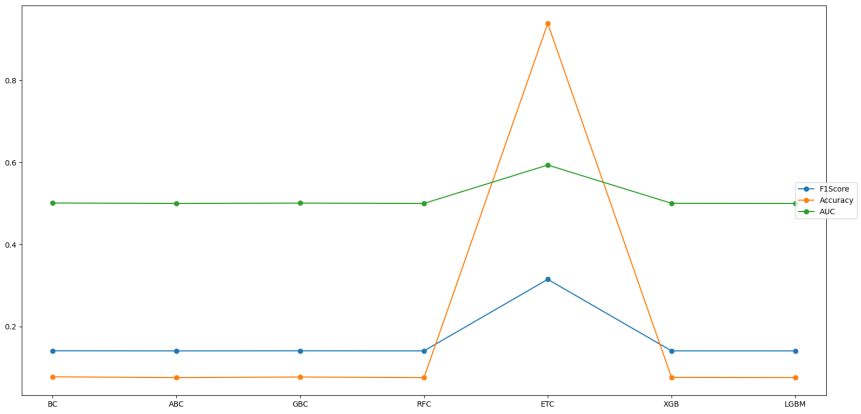


(a) Allenamento dei modelli di classificazione sul cluster 0.

Inoltre il comportamento dell'ETC è ambiguo, nel caso del cluster 1 esso non si posizionerebbe nemmeno tra i tre modelli migliori. Tuttavia, per ragioni di coerenza di procedura, si è proseguiti con le stesse scelte del paragrafo precedente.



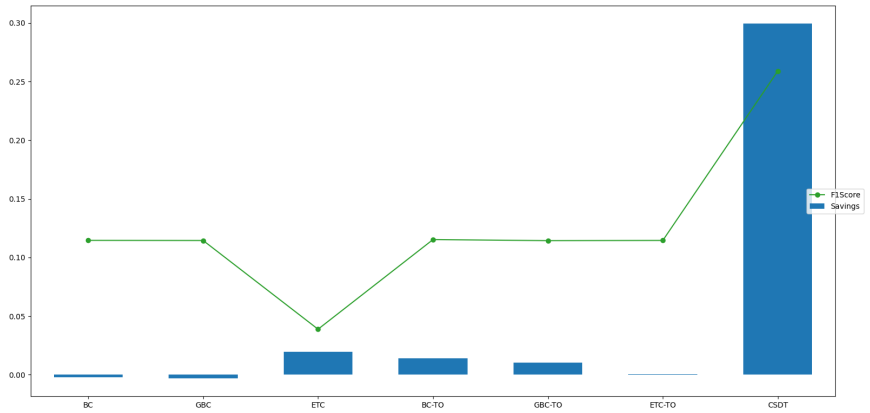
(b) Allenamento dei modelli di classificazione sul cluster 1.



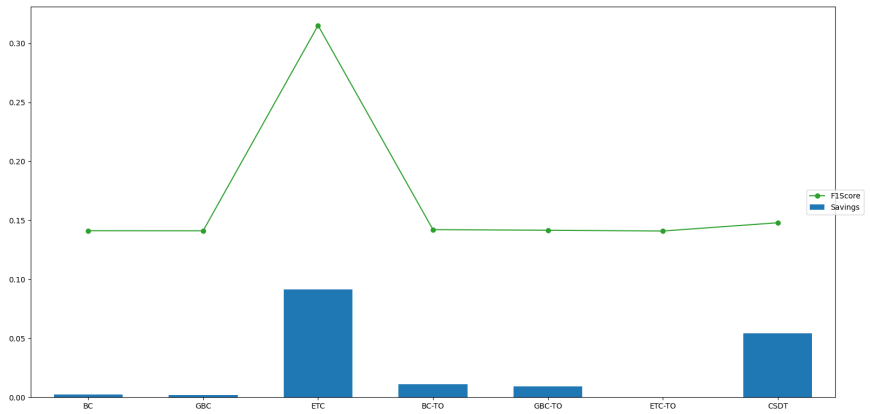
(c) Allenamento dei modelli di classificazione sul cluster 2.

Figura 6.4: Plot degli score in legenda calcolati per i modelli di classificazione allenati separatamente sui tre cluster.

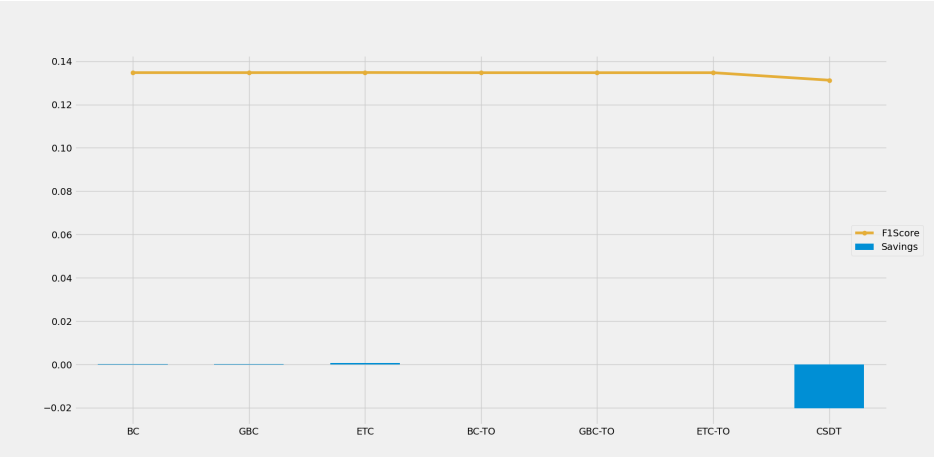
Applicando poi il modello di thresholding optimization ai modelli, si può osservare che i modelli cost sensitive agiscono quasi sempre come ci si aspetterebbe, in generale incrementando i savings score dei modelli base, tranne nel caso del modello RTC che ancora una volta ha un comportamento controintuitivo. Per quanto riguarda il CSDT, esso totalizza savings score incoraggianti, soprattutto nel caso del cluster 0. I risultati di questa fase sono rappresentati in Figura 6.5.



(a) Score calcolati per il cluster 0.



(b) Score calcolati per il cluster 1.

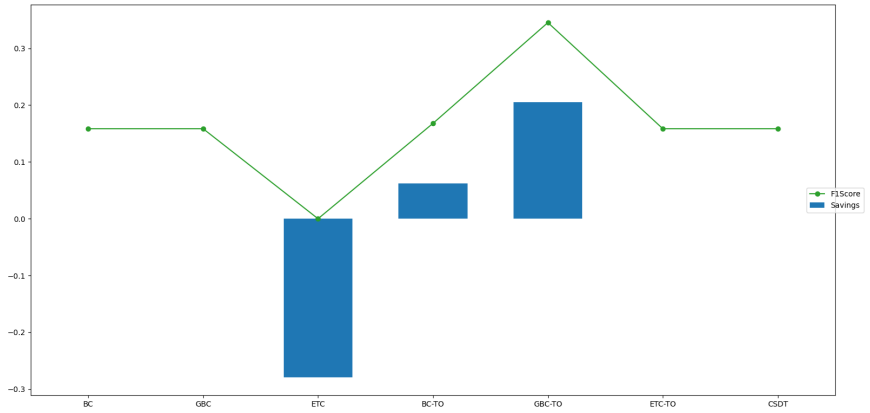


(c) Score calcolati per il cluster 2.

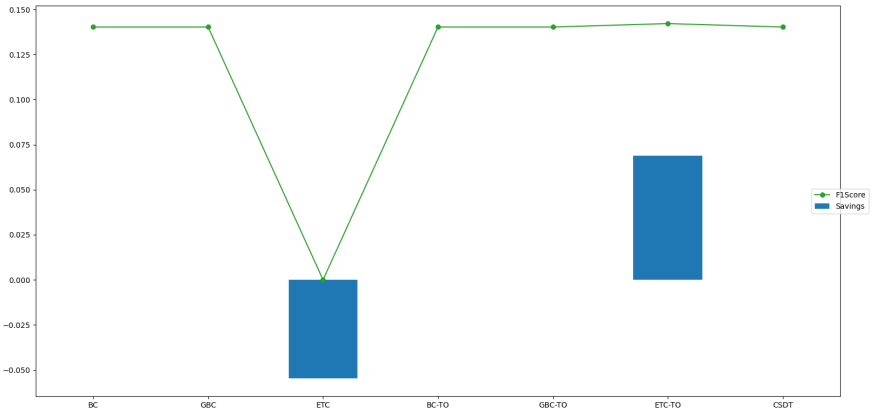
Figura 6.5: Plot di f1-score e savings calcolati, nei tre cluster, per i tre modelli migliori, la loro thresholding optimization e il cost sensitive decision tree.

L'analisi di questo set di dati si conclude con il calcolo di f1-score e savings sull'insieme di test (Figura 6.6). I risultati sono decisamente negativi: in alcuni casi il modello non ha totalizzato savings, in altri essi sono fortemente negativi. Si comportano comunque abbastanza bene BC-TO e GBC-TO, che in due cluster su tre totalizzano savings score positivi.

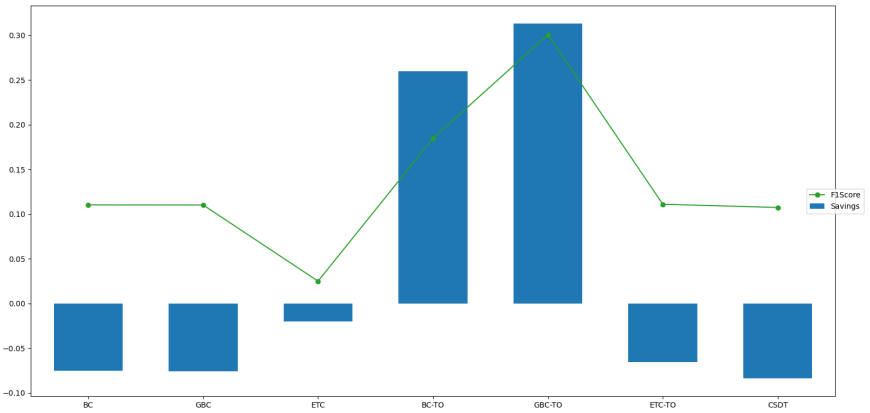
In conclusione, si può affermare che ridurre l'informazione fornita ai modelli ha apportato più svantaggi di quanti non siano stati i vantaggi: non ha infatti risolto il problema parziale dell'overfitting individuato al punto precedente, ma in compenso ha favorito una crescita esponenziale della varianza dei modelli, rendendo il loro comportamento imprevedibile e irregolare.



(a) Test dei modelli di classificazione sul cluster 0.



(b) Test dei modelli di classificazione sul cluster 1.



(c) Test dei modelli di classificazione sul cluster 2.

Figura 6.6: Plot dell'f1-score e dei savings totalizzati dai modelli in fase di test sui tre cluster.

6.3 Clustering come label del dataset, classificazione su tutti i dati

Si procede ora con lo studio del comportamento dei modelli di classificazione nel caso in cui il prodotto del clustering non si manifesti più nella suddivisione dei dati in gruppi, ma piuttosto vada a costituire una nuova label per il dataset. In questo caso - come nel successivo - non si tratteranno più tre insiemi separati di train, ma uno unico.

Si procede dunque con l'analisi visiva degli score totalizzati dai modelli di classificazione. Come si può notare dalla Figura 6.7, il comportamento dei modelli varia leggermente rispetto a quello riscontrato nel caso di dataset separati: questa volta, infatti, la Random Forest si comporta leggermente meglio degli altri modelli, totalizzando f1-score maggiori. Tuttavia, in linea con la disciplina impostata per tutto il lavoro, e osservando comunque negli altri modelli un comportamento in linea con le aspettative, si sceglie di proseguire l'analisi con i soliti tre modelli discussi in precedenza.

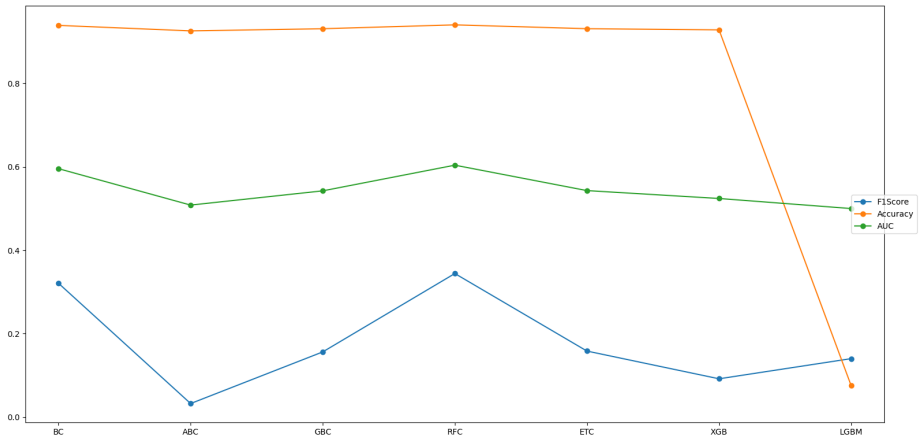


Figura 6.7: In figura, il plot degli score totalizzati dai modelli di classificazione nel caso di dataset non suddiviso in cluster.

Per questa ragione, in Figura 6.8 è possibile osservare il comportamento di BC, GBC e ETC quando sottoposti a thresholding optimization. Qua, mentre BC risponde molto bene al processo di ottimizzazione, totalizzando savings score alti, non si può dire altrettanto per gli altri due modelli. Risponde bene invece il CSDT, che si posiziona primo dal punto di vista dei savings.

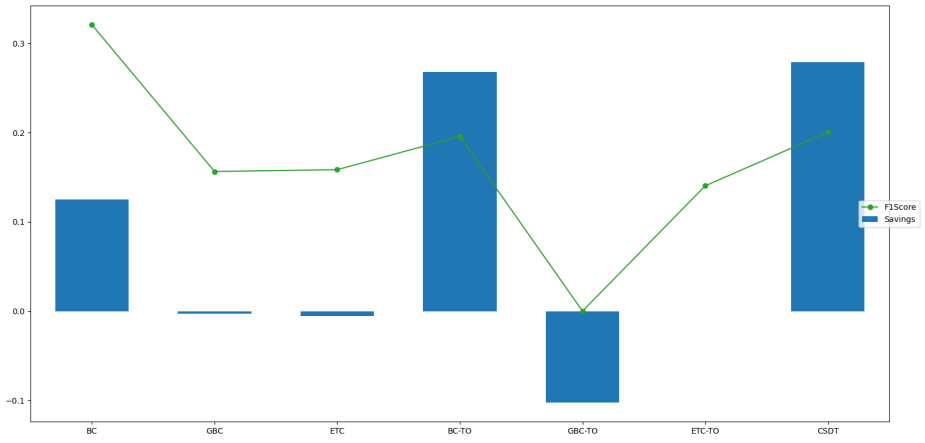


Figura 6.8: In figura, il plot di f1-score e savings totalizzati dai modelli di classificazione, dai loro corrispettivi ottimizzati e dal CSDT.

Non resta dunque che verificare se i modelli mantengono la stessa linea quando applicati al dataset di test. Osservando la Figura 6.9, verrebbe da affermare che non sia così: infatti, i tre modelli base hanno ora totalizzato score molto più alti, mentre il comportamento di BC-TO e GBC-TO si è invertito; l'unico modello ad aver soddisfatto le aspettative resta il CSDT, che ancora una volta totalizza savings piuttosto buoni. In generale, dunque, i modelli soffrono ancora una volta di una forte varianza. Ci si sarebbe aspettato forse il comportamento opposto: l'aver allenato i modelli sull'intero dataset di train dovrebbe intuitivamente condurre ad un forte overfitting dei modelli, fenomeno che invece non si è verificato, stando ai risultati.

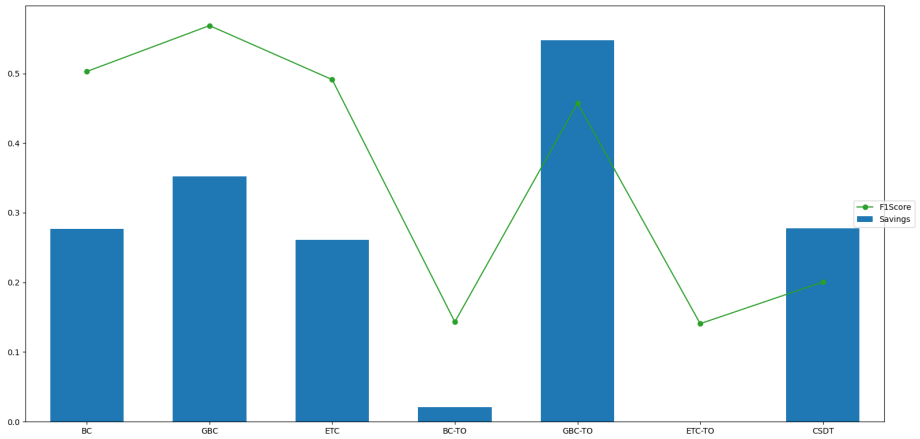


Figura 6.9: In figura, il plot di f1-score e savings totalizzati dai modelli di classificazione, dai loro corrispettivi ottimizzati e dal CSDT in fase di test.

6.4 Clustering come label del dataset, classificazione sulle colonne corrispondenti alle macro-features

In ultima analisi si consideri il dataset intero ma con l'apporto informativo costituito solo dalle colonne corrispondenti alle macro-features, e si ripeta l'iter dei paragrafi precedenti.

In fase di train, come riportato in Figura 6.10, si ritrova ancora una volta il comportamento omogeneizzato dei modelli che si era osservato al paragrafo 6.2, ovvero: i modelli tendono a totalizzare score molto simili, tendenzialmente bassi.

Selezionando sempre gli stessi tre modelli, essi vengono dunque sottoposti a thresholding optimization, il prodotto della quale è raffigurato in 6.11. Qua si osserva che il processo ha quantomeno favorito l'ottimizzazione dal punto dei vista dei savings, pur non fornendo risultati eccezionali, mentre decisamente incoraggiante è il comportamento del CSDT, che ancora una

volta totalizza i savings più alti.

Si procede alla verifica dei modelli con la fase di test i cui risultati sono stati riportati in Figura 6.12: qua si può osservare che il comportamento dei modelli rimane all'incirca coerente, anche se il GBC-TO totalizza savings score decisamente più alti di quanto ci si aspettasse.

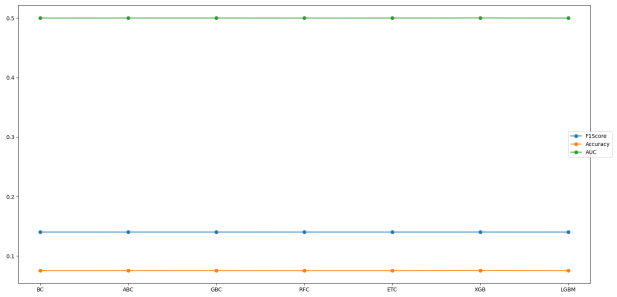


Figura 6.10: In figura, il plot degli score totalizzati dai modelli di classificazione in fase di train.

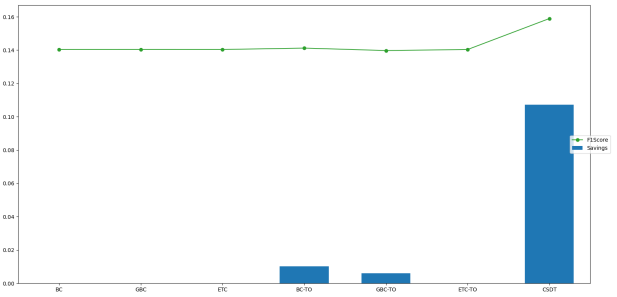


Figura 6.11: In figura, il plot di f1-score e savings totalizzati dai modelli di classificazione, dai loro corrispettivi ottimizzati e dal CSDT.

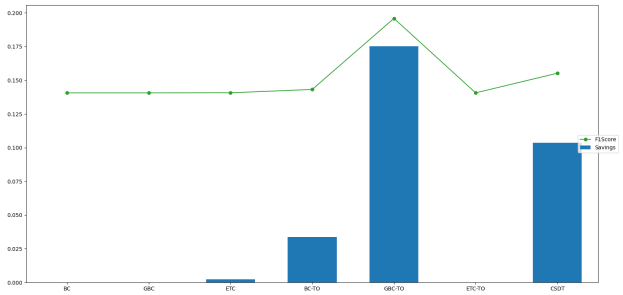


Figura 6.12: In figura, il plot di f1-score e savings totalizzati dai modelli di classificazione, dai loro corrispettivi ottimizzati e dal CSDT in fase di test.

Capitolo 7

Conclusioni e next steps

In base a quanto osservato e commentato nel Capitolo precedente, si cerca ora di trarre conclusioni sugli studi effettuati in questo lavoro di tesi.

Dal punto di vista dei modelli di classificazione ensemble classici, si può affermare con una certa sicurezza che lo scenario i cui essi hanno fornito i risultati migliori sono quelli in cui il prodotto del clustering consisteva nella suddivisione del dataset in classi, e in cui si sfruttava tutta l'informazione fornita dal punto di vista del numero di colonne (primo caso studiato nel Capitolo 6). In questa situazione, infatti - a meno di varianza standard dovuta alla natura reale del problema - i modelli hanno mantenuto un comportamento all'incirca costante e prevedibile, totalizzando f1-score piuttosto alti. Se dunque il modello di classificazione si dovesse basare soltanto su uno di questi modelli, sicuramente la scelta dovrebbe ricadere su questo iter di modellazione.

Per quanto riguarda invece il pacchetto dei modelli cost sensitive, due discorsi separati sono d'obbligo. In primis, si consideri il modello di thresholding optimization. Esso ha quasi sempre fornito risultati contrastanti: in alcuni casi i modelli di classificazione sottoposti a questa ottimizzazione miglioravano effettivamente le prestazioni dal punto di vista dei savings, in altri invece i risultati sono stati deludenti. Quello che comunque è parzialmente emerso dalle analisi è che

- nei casi di dataset unificati (clustering come label) il comportamento dell'ottimizzazione era più coerente e costante, anche se gli score totalizzati erano mediamente bassi;

- invece nei casi di cluster separati i modelli ottimizzati si comportavano in maniera ambigua, e la laddove avessero successo gli score totalizzati erano molto positivi.

Il comportamento di questo modello è dunque ambiguo, per cui non si è in grado di trarre conclusioni precise sul suo utilizzo.

Diverso è invece il discorso per quanto riguarda il cost sensitive decision tree classifier: esso, in netta contrapposizione con quanto affermato per i modelli di classificazione standard, ha dimostrato di comportarsi meglio quando applicato al dataset unificato. Questo può significare che il modello, per sua natura, si alleni meglio quando applicato a dataset ampi con alta varianza interna. Se dunque si scegliesse di intraprendere la strada dei modelli cost sensitive, il clustering potrebbe assumere un ruolo marginale nell'iter di classificazione.

Per concludere, i risultati del presente lavoro di tesi hanno lasciato un ampio margine di sviluppo, non tanto dal punto di vista dell'analisi esplorativa del comportamento dei modelli di classificazione standard, quanto piuttosto da quello dei due elementi di innovazione introdotti: il clustering e i modelli cost sensitive. Gli ultimi soprattutto, come dimostrato nella descrizione dei risultati, richiederanno in futuro un ampio studio ai fini di ottimizzarne l'impiego, in primis dal punto di vista della costruzione della matrice dei costi - questione legata intrinsecamente alla natura del problema -, in secundis da quello dell'implementazione dei modelli veri e propri, che come si è notato differiscono nel comportamento dai loro corrispettivi standard. Ma anche l'applicazione del clustering lascia ancora spazio ad ulteriori studi; per il futuro si potrebbe pensare infatti di approfondire la risposta alla domanda: in quale punto della pipeline è più efficace l'inserimento di clustering?,

così come confrontare altri tipi di clustering. Infine, si potrebbe pensare di svolgere un lavoro parallelo nell'estrapolazione delle regole del *black box* del clustering, cercando di trarre maggiori informazioni sulla natura delle classi e di conseguenza di ottimizzarle.

Bibliografia

[1] Glossario Marketing, "Definizione di Churn rate", https://www.glossariomarketing.it/significato/churn-rate/

[2] Frederick F. reichheld, W. Earl Sasser Jr, "Zero defections: quality comes to services", Harvard Business Review, settembre-ottobre 1990

[3] Len Markidan, "Why customer churn happens, and what you can do about it", https://www.groovehq.com/blog/reduce-customer-churn

[4] Oracle, "2011 Customer experience impact report - Customer and brand relationship",http://www.oracle.com/us/products/applications/cust-exp-impact-report-epss-1560493.pdf

[5] Smartsheet, "The definitive guide to client onboarding", https://www.smartsheet.com/definitive-guide-client-onboarding

[6] Erika Trujillo, "15 Customer Service Skills & How to Improve Each One (Step-by-Step)", https://www.groovehq.com/blog/customer-service-skills-improve

[7] Lincoln Murphy, "The Secret to Successful Customer Onboarding", https://sixteenventures.com/customer-onboarding

[8] David Skok, "Managing customer success to reduce churn", https://www.forentrepreneurs.com/customer-success/

[9] https://it.wikipedia.org/wiki/Boosting

[10] https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html

[11] http://albahnsen.github.io/CostSensitiveClassification/Sampling.html#costcla.sampling.smote

[12] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[13] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_curve.html

[14] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

[15] https://scikit-learn.org/stable/modules/decomposition.html#pca

[16] https://it.wikipedia.org/wiki/Analisi_delle_componenti_principali

[17] https://scikit-learn.org/stable/modules/clustering.html#k-means

[18] https://it.wikipedia.org/wiki/K-means

[19] Ying Huang, Tahar Kechadi, "An effective hybrid learning system for telecommunication churn prediction", Expert Systems with Applications, Volume 40, Issue 14, 15 October 2013, Pages 5635-5647

[20] Shin-Yuan Hung, David C. Yen, Hsiu-Yu Wang, "Applying data mining to telecom churn management", Expert Systems with Applications, Volume 31, Issue 3, October 2006, Pages 515-524

[21] Indranil Bose, Xi Chen, "Hybrid models using clustering for prediction of customer churn", International MultiConference of Engineers and Computer Scientists 2009, Vol I, IMECS 2009, March 18 - 20 2009, Hong Kong

[22] R. Rajamohamed, J. Manokaran, "Improved credit card churn prediction based on rough clusterin and supervised learning techniques", Cluster Comput 21, 65–77, 2018

[23] `https://www.statistics.com/glossary/decile-lift/`

[24] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html`

[25] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html`

[26] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html`

[27] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html`

[28] `https://xgboost.readthedocs.io/en/latest/python/python_api.html`

[29] `https://en.wikipedia.org/wiki/LightGBM`

[30] `http://albahnsen.github.io/CostSensitiveClassification/`

[31] `https://www.invespcro.com/blog/customer-acquisition-retention/`

[32] `http://albahnsen.github.io/CostSensitiveClassification/ThresholdingOptimization.html`