ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Riconoscimento di frodi attraverso la modellazione del comportamento degli utenti

Tesi di laurea in
Big Data

Relatore
Dott. Enrico Gallinucci

Candidato
Alex Ravaglia

III Sessione di Laurea
Anno Accademico 2019-2020

Sommario

Lo scopo principale di questa tesi è quello di voler mostrare un approccio per il riconoscimento di frodi bancarie. Vengono descritte e fornite soluzioni per diversi problemi, tra i quali la manipolazione di dataset con distribuzioni sbilanciate dei dati tra le classi. Vengono addestrati e testati diversi algoritmi di classificazione dimostrando come Random Forest e XGBoost siano gli approcci migliori. Un aspetto principale che permette di migliorare notevolmente i risultati ottenuti è quello di basare la classificazione sul comportamento degli utenti. Viene mostrato come sia possibile tramite un processo di feature engineering proporre nuovi attributi che permettano di catturare relazioni tra una transazione bancaria e le operazioni passate effettuate dallo stesso utente. Viene infine proposta l'implementazione di un prototipo basato su un'infrastruttura big data che possa analizzare streaming di dati.

Indice

Elenco delle figure

Introduzione

La frode è un problema che è sempre esistito in diverse forme, lo sviluppo di nuovi strumenti tecnologici ha aumentato le possibilità introducendo nuove modalità d'attacco. Al giorno d'oggi utilizzare carte di credito e di debito è diventata la normalità negli acquisti. Per questo motivo le frodi che sono correlate a queste forme di pagamento sono sempre più in aumento. Le perdite finanziarie non colpiscono solo negozianti e banche (che devono rimborsare le spese) ma anche i clienti. Se un cliente subisce una frode dovrà pagare alla banca dei tassi di interesse più elevati perchè potrebbe essere considerato come un profilo a rischio. Una frode, inoltre, compromette anche la reputazione e l'immagine di un commerciante. Se un cliente durante un acquisto rimane vittima di una frode, potrebbe perdere la fiducia nel venditore ed affidarsi alla concorrenza per acquisti futuri. Dato un set di dati derivanti da transazioni con carte di credito, il riconoscimento delle frodi è il processo mediante il quale si identifica se una nuova transazione appartiene alla classe delle transazioni fraudolente o legittime. Un sistema di questo tipo non solo dovrebbe rilevare le operazioni fraudolente, ma dovrebbe farlo in maniera conveniente. Il costo del controllo di queste operazioni non dovrebbe eccedere la perdita dovuta alle frodi stesse. Non è semplice calcolare l'impatto delle frodi, in genere, poiché le banche non rilasciano troppe informazioni sull'ammontare delle perdite subite. La misura può essere data limitandosi a considerare le azioni illecite che vengono identificate. È impossibile valutare la dimensione delle perdite relative a frodi che non sono state scoperte o segnalate [12]. La Association for Payment Clearing Services (APACS) ha stimato che nel Regno Unito il totale delle perdite causate da frodi su carte di credito ha raggiunto la quota di 440,3 milioni di sterline nel 2010 [24]. A livello mondiale invece il picco è stato raggiunto nel 2014 con 11,27 miliardi di dollari, di cui la metà secondo Lexis Nexis [3], solo negli Stati Uniti d'America. La Banca Centrale Europea [1] riporta che, nel 2012, è stata stimata una perdita per frodi pari a 1,33 miliardi di euro, un aumento del 14,8% rispetto l'anno precedente; equivale a circa 1€ ogni 2635€ spesi. In particolare, il 60% di queste frodi sono avvenute con pagamenti su internet, il 23% da pagamenti mediante POS (bancomat o carta di credito nei negozi fisici) e il 17% da punti di prelievo ATM. Dal 2011 al 2012 le frodi online sono aumentate del 21% in seguito

all'aumento delle transazione avvenute su internet.

Il seguente elaborato di tesi ha lo scopo di voler analizzare e mostrare una serie di metodologie e applicazioni nell'ambito del riconoscimento delle frodi bancarie. Sebbene esistano diversi meccanismi per risolvere il problema, le principali soluzioni consistono nell'utilizzare algoritmi di classificazione. In primo luogo, viene affrontato il problema su come sia opportuno trattare la tipologia di dati a disposizione. Si è soliti avere molte informazioni di operazioni legittime, ma poche relative a operazioni fraudolente; questi dati infatti, sono più rari da osservare. Durante le analisi, può essere importante l'intervento da parte di personale specializzato che valuti l'operato di un modello predittivo. Per questo motivo è fondamentale che il sistema fornisca un criterio di classificazione che possa essere interpretabile da parte di un essere umano. Si pensi che la valutazione finale di un'operazione bancaria, sia frutto dell'integrazione del risultato fornito da un modello di machine learning e il parere di un esperto. Il processo vero e proprio di identificazione di un'operazione fraudolenta non è banale, poiché devono essere prese in considerazione diverse variabili. Dare una valutazione osservando una singola transazione può non essere sufficiente. Ciò che per un utente è considerato normale potrebbe essere un'anomalia per un altro. La modellazione del comportamento di un utente ricopre quindi un ruolo centrale nella definizione di soluzioni a questo problema. La classificazione di una transazione può avvenire in momenti diversi. Esistono sistemi che eseguono controlli a posteriori per verificare se un utente ha subito una frode senza che se ne sia reso conto, limitando così il danno. Un altro contesto di lavoro, potrebbe prevedere che la classificazione sia eseguita prima che il pagamento venga accettato. È richiesto che un sistema che effettui questo tipo di valutazione fornisca un risultato in tempo reale. Questo tipo di analisi richiede che esistano sistemi in grado di processare grandi quantità di dati e che forniscano risposte nel minor tempo possibile. Viene per questo motivo presentato un prototipo basato su un' infrastruttura big data che soddisfi i requisiti richiesti.

Struttura della tesi. Nell'ottica di voler presentare questo progetto di tesi, ne viene descritta in questo paragrafo la principale struttura. Per ogni capitolo viene brevemente riportato il tema principale e fornita una linea guida sulla sua composizione.

Il Capitolo 1 vuole fornire una panoramica delle principali metodologie presenti in letteratura per affrontare il problema delle frodi. Sono descritti i principali algoritmi di classificazione e metodologie per gestire dati con distribuzioni non omogenee. Viene infine analizzata una possibile implementazione reale che si basi su tecnologie big data.

Nel Capitolo 2 vengono descritte le diverse tipologie di analisi che possono essere eseguite e ci si concentra sulla proposta di una serie di tecniche di feature engi-

neering che mirano a catturare aspetti comportamentali dell'utente. Si riflette sul ruolo che questi assumono nel processo di previsione e di quali siano le principali metriche da utilizzare per valutare gli approcci descritti.
Il Capitolo 3 mostra un caso d'uso reale e l'utilizzo concreto di tecniche che mirano a basare la previsione su singole transazioni. Viene affrontato e gestito il problema delle distribuzioni dei dati e mostrati i risultati che diversi classificatori riescono a raggiungere, evidenziando quali possano essere i migliori approcci possibili.
Nel Capitolo 4 si affronta il problema in maniera differente su un nuovo caso d'uso. Viene mostrato quale sia il ruolo dell'utente e vengono applicate le tecniche di modellazione del comportamento; ruolo chiave ha la fase di feature engineering per catturare queste informazioni.
Dopo aver discusso le diverse metodologie e averle applicate su dataset reali mostrandone i risultati, si presenta nel Capitolo 5 l'implementazione di un prototipo che realizza un sistema software basato su tecnologie big data in grado di gestire grandi flussi di dati in tempo reale.
Vengono, infine, riportate le conclusioni e riassunti i principali contributi forniti.

Capitolo 1

Stato dell'arte

In questo capitolo vengono presentate e descritte le principali metodologie utilizzate in letteratura per risolvere il problema del riconoscimento delle frodi. Vengono descritti i principali approcci supervisionati e non supervisionati. Si affronta il problema della gestione di dataset con distribuzioni sbilanciate tra le classi, sono presentati i principali algoritmi di classificazione e le metriche di valutazione che è opportuno utilizzare per valutare i modelli. Viene infine fatta una panoramica sulle principali implementazioni reali di sistemi di questo tipo.

1.1 Descrizione generale del problema

Nel lavoro di ricerca di [46] si tratta il tema su come risolvere il problema del riconoscimento delle frodi basandosi su un modello predittivo. Tale sistema assegnerà un punteggio ad ogni transazione sulla misura di quanto ritiene che questa possa essere una frode. Le transazioni alle quali viene assegnato un valore superiore a una certa soglia vengono classificate come operazioni fraudolente. Oltre ai sistemi automatici un ruolo importante è quello che può essere assunto da analisti e personale specializzato. Controllano le segnalazioni effettuate dal sistema automatico e forniscono un feedback per ogni possibile allarme. Il riscontro fornito da un esperto può essere usato per migliorare la parte di previsione automatica. Un meccanismo di controllo potrebbe essere creato sulla base delle regole definite da esperti di dominio, un sistema che si basi solo su queste informazioni però richiederebbe una supervisione e il continuo intervento da parte di personale specializzato. Un'alternativa si basa sull'uso di tecniche di machine learning, in questo modo si possono scoprire nuovi pattern di dati fraudolenti ed effettuare previsioni, basandosi solamente sui dati a disposizione. Le tecniche di machine learning consistono nel riuscire ad apprendere un modello predittivo sulla base dell'analisi di un insieme di esempi. Il modello è una funzione che predice quanto una transazione

possa essere considerabile fraudolenta partendo da un insieme di feature che la descrivono. Vi sono diversi motivi per i quali utilizzare tecniche di apprendimento automatico: si rivela un approccio interessante rispetto far svolgere il lavoro a personale specializzato. I dati da analizzare sono costituiti da flussi continui di informazioni in ingresso al sistema in ogni momento. Inoltre i record che devono essere valutati spesso sono ad elevata dimensionalità, questo vuol dire che ciascuna transazione è descritta da molte variabili. Vi è spesso una correlazione spaziale e/o temporale tra le operazioni fraudolente, un modello di machine learning potrebbe inferire questo tipo di informazione in maniera automatica. Alcuni modelli possono essere integrati con i feedback degli analisti in modo che si riescano a unire i benefici dell'avere un sistema automatico con l'esperienza di personale specializzato nell'analisi di questo tipo di informazioni. Nello sviluppo di sistemi che intendono risolvere questo tipo di problema è importante considerare la centralità del cliente. Ai fini di volergli garantire un'esperienza soddisfacente, è opportuno limitare il numero di falsi allarmi. Bisogna considerare tra l'altro che solo una piccola frazione delle transazioni è una frode, spesso meno del 0.1% [38].

1.2 Approcci supervisionati e non

Come descritto nel lavoro di [46], esistono diverse strategie per risolvere il problema della classificazione delle frodi. Possono esserci soluzioni supervisionate [13] e non [52]. Tecniche supervisionate assumono che siano note le etichette delle classi di tutti i dati nel dataset, in modo che si sappia se ogni transazione presente è legittima o fraudolenta. Tramite questo meccanismo il modello viene addestrato sulla base della conoscenza dell'etichetta di ciascuna transazione. In un approccio non supervisionato invece, non è nota la classe di appartenenza di nessuna transazione. Ci si basa su una strategia di riconoscimento degli outlier o tramite tecniche di riconoscimento di anomalie che permettano di associare una transazione a un comportamento fraudolento ritenuto non conforme agli altri dati [12]. Tecniche non supervisionate spesso generano falsi allarmi, per questo motivo potrebbe essere una buona idea combinare entrambi gli approcci come presentato in [40]. Vengono definite in [46] 4 principali tecniche supervisionate:

- Profilazione.
- Classificazione.
- Modelli cost-sensitive.
- Connessioni tra i dati.

1.2.1 Profilazione

Per ogni classe possono essere creati differenti profili [51]. In [50] viene proposto l'uso della metrica "Weight of Evidence" come misura di similarità tra profili. Per ognuno di questi possono essere identificate una serie di regole da parte di un esperto di dominio. Nuove regole possono essere definite facendo riferimento all'esperienza di personale specializzato o all'analisi di modelli statistici. In [17] viene proposto un esempio di sistema nel quale vengono estratte le transazioni legittime tramite un insieme di regole. Bisogna considerare che nel tempo vi possono essere delle evoluzioni, sia nel comportamento degli utenti, sia nelle attività criminali. Per questo motivo i profili devono essere aggiornati costantemente con nuove regole per fare fronte ai cambiamenti.

1.2.2 Classificazione

La classificazione è l'approccio standard con il quale risolvere questo tipo di problemi [43]. In letteratura esistono e sono stati adottati diversi algoritmi come : regressione logistica, reti neurali, svm, alberi decisionali, k-nn, random forest. Tra questi i decision tree sono molto usati nell'ambito del riconoscimento di frodi [29]. Tutti questi approcci potrebbero soffrire del problema dello sbilanciamento delle classi. Le reti neurali e gli approcci deep vengono applicati in diversi contesti e anche in questo ambito possono mostrare risultati interessanti [26]. Il problema è che sono modelli *black box* ovvero il criterio tramite il quale si basa la classificazione non è interpretabile da esperti di dominio. In applicazioni reali può essere necessario che i risultati ottenuti tramite un modello predittivo di questo tipo debbano essere compresi e interpretati da personale specializzato al fine di rendere la classificazione attendibile. Per questo motivo si preferiscono soluzioni che offrono dei modelli *white box* come: alberi decisionali o approcci basati su regole. Random forest ha dimostrato di essere uno tra i migliori algoritmi di classificazione per problemi di riconoscimento di frodi [23].

1.2.3 Metodi cost-sensitive

Il motivo principale per il quale si cercano di riconoscere le frodi è il fatto che si vogliono ridurre le perdite in denaro. In un approccio cost-sensitive si assegna un peso differente a errori commessi su classi diverse [25]. Il peso di un errore di classificazione sarà maggiore nel classificare una frode come operazione legittima piuttosto che il contrario. Questo meccanismo potrebbe essere considerato come un approccio alternativo per gestire il problema dello sbilanciamento dei dati tra le classi [28]. In questo modo, errori nella classificazione di elementi appartenenti alla classe minoritaria hanno un costo maggiore rispetto errori commessi su elementi

della classe maggioritaria. In letteratura sono disponibili diversi algoritmi di questo tipo e si basano sulla tecnica del boosting. Un esempio sono AsymBoost[54], AdaCost[30], DataBoost[33]. Alcuni classificatori basano l'errore su una misura che dipende dalla singola transazione e quindi dalla somma in denaro coinvolta [49]. Questi tipi di approcci sono utili quando l'obiettivo primario è quello di ridurre al minimo le perdite in seguito a frodi. Viene assegnato un errore maggiore nel caso di classificazione di frodi come operazioni legittime. Considerato il meccanismo di funzionamento, si osserva che la tendenza di questo tipo di sistemi è quella di classificare come frodi quelle transazioni sulle quali vi è incertezza. Il motivo è dato dal fatto che si cercano di minimizzare le operazioni fraudolente non predette riducendo così possibili perdite. Per questo motivo vengono generati molti falsi positivi. Può essere una strategia non efficace nei casi laddove si vogliano ottenere solo poche segnalazioni mirate.

1.2.4 Connessioni tra i dati

Analizzare i collegamenti tra i dati potrebbe aiutare le operazioni di riconoscimento di frodi [31]. Non tutte le attività illegali potrebbero essere identificate analizzando dati e cercando pattern identificabili come fraudolenti. L'analisi tra i collegamenti delle operazioni e lo studio dei grafi permette di identificare gruppi di transazioni fraudolente che altrimenti non verrebbero catturate[51]. Nel lavoro proposto in [53] viene presentato un framework per l'analisi di transazioni bancarie tramite il quale oltre le normali informazioni è possibile definire una serie di feature relative alle connessioni tra i dati. Si dimostra che l'aggiunta di questo tipo di feature può migliorare in maniera significativa i risultati ottenuti.

1.2.5 Metodi non supervisionati

Metodi non supervisionati permettono di lavorare con dati non etichettati ovvero di cui non si conosce la reale classe di appartenenza. Uno dei vantaggi potrebbe essere quello che non vi è il problema dello sbilanciamento delle classi, tuttavia questo tipo di approccio sembra non aver riscosso troppo interesse in letteratura [12]. Tecniche di questo tipo si basano spesso su combinazioni tra la profilazione degli utenti e il riconoscimento degli outlier. È opportuno identificare un modello di comportamento che possa fungere da linea guida, l'analisi mira a identificare elementi che se ne discostano[39]. Una di queste metodologie è la *Peer group Analysis* [40], gli utenti vengono raggruppati in diversi profili e si identificano le frodi sulla base dello scostamento di ogni transazione dal profilo dell'utente. [46] riflette sul fatto che per definire un outlier o un'anomalia è prima di tutto opportuno definire quando un esempio possa considerarsi tale. Quando è possibile definire una regione o distribuzione di dati come normale, allora tutte le osservazioni non

conformi possono essere definibili come outlier. In ogni caso le sfide sono diverse, la definizione di una regione che possa rappresentare tutti i comportamenti normali non è banale. Un utente tende ad evolvere nel corso del tempo e modificare le proprie abitudini. Un agente attaccante cerca di adottare un comportamento che possa sembrare il più normale e realistico possibile. È complicato non confondere il rumore nei dati e le frodi. Si possono distinguere due differenti tipologie di outlier: locali e globali. Globale significa che il dato è anomalo rispetto l'intero dataset. Un outlier invece è tale solamente a livello locale se presenta un'anomalia rispetto ad un ristretto gruppo di dati.

1.3 Algoritmi di classificazione

Una delle tecniche più utilizzate nel riconoscimento di frodi è l'uso di modelli che effettuano una classificazione di un transazione bancaria. Esistono diversi algoritmi ognuno con i suoi punti di forza e debolezze, ogni metodologia effettua la classificazione basandosi su un criterio di riconoscimento differente. Segue una panoramica delle principali metodologie di classificazione.

1.3.1 Decision Tree

I decision tree [48] sono una delle tecniche di classificazione più usate; permettono di rappresentare un' insieme di regole di classificazione secondo un'organizzazione gerarchica. Un albero è composto da un insieme di nodi in relazione tra loro tramite archi etichettati ed orientati. Ogni percorso dalla radice a una foglia identifica una regola di classificazione, a ciascuna foglia viene associata una classe. Esistono diversi algoritmi basati sui decision tree, un esempio possono essere: Hunt's algorithms, C4.5, SPRINT, CART. Identificare il miglior decision tree possibile è un problema np-completo, motivo per il quale la creazione degli alberi si basa su euristiche. Tra i principali benefici che si hanno nell'usare questo tipo di modelli, risulta che il processo di classificazione è molto veloce ed è un meccanismo robusto nel trattare dati con correlazioni tra attributi. Per definire un algoritmo basato su un approccio ad albero decisionale è opportuno che vengano definiti diversi aspetti tra i quali:

- Definizione di una tipologia di partizionamento.
- Criterio per definire il miglior partizionamento.
- Criterio per definire il momento di stop nella creazione dell'albero.
- Metodo di valutazione.

La strategia di partizionamento può dipendere dalla tipologia di attributo: nominale, ordinale, quantitativo. Si può inoltre definire se il partizionamento deve essere binario o N-ario. Ai fini della valutazione della bontà di ciascun partizionamento, si possono usare diverse misure di purezza per valutare gli insiemi generati. Un esempio di metriche utilizzabili potrebbero essere : gini index, entropia, misclassification error. Definire un criterio di stop nella creazione dell'albero significa determinare una strategia secondo la quale è opportuno fermare il meccanismo di partizionamento. Un esempio potrebbe essere quello di fermarsi quando tutti i dati che appartengono a uno stesso nodo sono della stessa classe, oppure quando il numero di record in un nodo è inferiore a un certo valore di soglia.

1.3.2 Random Forest

Random forest [14] è un metodo di bagging, ovvero una metodologia tramite la quale si addestra un algoritmo su porzioni diverse del training set. Si avranno in output diversi modelli ognuno addestrato su un sottoinsieme di dati. In Random Forest i singoli classificatori sono dei decision tree (CART di default). La classificazione finale tiene conto del risultato ottenuto su ogni singolo albero decisionale e viene definita la classe finale tramite un meccanismo di majority vote rule. Ai fini di rendere indipendenti il più possibile i diversi modelli, ogni decision tree è creato su un sottoinsieme di attributi. Questo meccanismo viene utilizzato per evitare che tutti i decision tree effettuino le stesse scelte andando poi di fatto a essere tutti molto simili. Vengono fatti due tipi di bagging, uno sui dati scelti per il training e uno a livello di feature.

1.3.3 Gradient Boosting

Gradient Boosting [32] è un meccanismo simile all'approccio presentato in Random Forest. Il modello predittivo finale si basa sull'integrazione di diversi decision tree. Gradient boosting addestra diversi modelli in maniera graduale e sequenziale. La differenza principale è che Random Forest crea modelli tra loro indipendenti, Gradient Boosting crea gli alberi uno alla volta in maniera sequenziale. Ad ogni iterazione viene addestrato un nuovo classificatore per compensare ciò che l'insieme di classificatori prodotti non riescono a catturare. Le mancanze che devono essere colmate vengono identificate tramite discesa del gradiente della funzione di loss. Questo modello riesce a mostrare dei buoni risultati lavorando su dati non bilanciati tra le classi. Un algoritmo che si basa su questa tecnica è XGBoost.

1.3.4 Support Vector Machine - SVM

SVM [22] è un meccanismo tramite il quale la classificazione viene effettuata assegnando una classe ad ogni record sulla base della sua posizione in uno spazio multidimensionale. Si prenda come riferimento uno spazio che possa contenere tutti i dati e lo si partizioni in modo da definire due regioni disgiunte, tutti i dati nella stessa regione avranno la stessa classe, mentre i dati in regioni diverse avranno valori di classe differenti. Il problema si trasforma nel riuscire a determinare la migliore superficie decisionale. Parliamo di SVM lineare se la superficie di separazione è un iperpiano, i record sono linearmente separabili se esiste un iperpiano che operi questa divisione. Se non linearmente separabili ci saranno inevitabilmente degli errori di classificazione poichè non esiste nessun iperpiano in grado di effettuare questa separazione. Possono essere definite superfici di separazione complesse, in questo caso parliamo di SVM non linerare. SVM nasce come classificatore binario, ma vi sono delle estensioni che lo possono adattare a problemi multi classe. SVM cerca di determinare l'iperpiano di separazione tra le classi che massimizza il margine. Il margine può essere definito come la distanza minima tra un dato di una classe e l'iperpiano. L'iperpiano ottimo secondo SVM è quello che soddisfa i vincoli di separazione dei dati e massimizza il margine. Nel caso non tutti i pattern siano separabili, è necessario rilassare i vincoli di separazione. SVM non lineare permette di separare i dati definendo superfici complesse [44]. Viene definito un mapping dei pattern in uno spazio a più alta dimensionalità, qui saranno maggiori i gradi di libertà e sarà possibile definire un iperpiano tramite l'approccio spiegato. Per opportuni mapping, è possibile ricondurre le operazioni da risolvere nello spazio aumentato a una funzione "Kernel" sugli stessi dati nello spazio di partenza. Possono essere definite diverse funzioni di kernel: polinomiale, radial basis function (rbf), rete neurale con 2 layer.

1.3.5 Regressione Logistica

La regressione logistica [47] è un modello statistico che sfrutta una funzione logistica per modellare un variabile binaria. È un'attività predittiva che viene utilizzata per misurare la relazione tra la variabile dipendente e una o più variabili indipendenti stimando le probabilità tramite una funzione logistica.

1.3.6 Nearest Neighbor

Un record viene classificato da un algoritmo Nearest Neighbor [7] osservando tutti i dati nel training set e assegnando la stessa classe dell'elemento a lui più vicino. L'assunzione che viene fatta è che in uno spazio multidimensionale se due record sono "vicini", allora probabilmente appartengono alla stessa classe. Per misurare

la distanza è opportuno avere una metrica alla quale fare riferimento. Un esempio potrebbe essere la distanza euclidea o più in generale la metrica di Minkowski. La regola nearest neighbor produce un partizionamento dello spazio, noto come tassellazione di Voronoi. Ogni elemento del training set determina un tassello all'interno del quale i pattern saranno assegnati alla stessa classe. Un modo per rendere questo meccanismo più robusto, è quello di effettuare la classificazione di un record osservando i k record più vicini. Il record può essere assegnato alla classe che presenta il maggior numero di esempi tra quelli selezionati. Per avere un meccanismo meno sensibile al valore di k ogni record può fornire un contributo pesato sulla distanza dall'elemento da classificare. È necessario che per operare correttamente gli attributi abbiano la stessa scala di valori e quindi siano stati normalizzati in fase di pre-processing.

1.4 Bilanciamento delle classi

In [46] viene fatta un'analisi del problema che si presenta nell'addestrare modelli predittivi su dataset composti da dati non distribuiti omogeneamente tra le classi. Molti algoritmi non sono progettati per lavorare su dati con classi fortemente sbilanciate [11]. Quando la distribuzione dei dati non è omogenea, un algoritmo di machine learning, tende a classificare le istanze come appartenenti alla classe più popolosa, questo causa una bassa recall per la classe più rara. Per far fronte a questo problema possono essere prese soluzioni che operino a livello dei dati o dell'algoritmo [20]. A livello dei dati significa bilanciare le distribuzioni, trasformare il training set in modo che i dati siano distribuiti equamente tra le classi. Il problema, in alternativa, può essere risolto direttamente a livello algoritmico in modo che questo sia adattato a gestire e riconoscere classi con pochi dati. Diversi studi hanno dimostrato che i classificatori standard ottengono performance migliori quando addestrati su set di dati bilanciati[55]. È possibile ribilanciare i dati tra le classi tramite diversi meccanismi: undersample, oversample o una combinazione delle due. Le metodologie che si basano su undersample [27] mirano a ridurre la cardinalità della classe maggioritaria rimuovendone alcuni esempi. Nell' oversample[27], al contrario, ci si concentra sulla classe minoritaria, partendo dai dati disponibili si creino nuovi dati "sintetici" generati dalla combinazione delle informazioni disponibili. Un'ultimo approccio prevede di applicare entrambe le metodologie, ovvero effettuare un undersample della classe maggioritaria e un oversample della classe minoritaria.

1.4.1 Random over/under sample

È una delle tecniche più semplici. Random undersampling consiste nel rimuovere elementi dalla classe più popolosa fintanto che il dataset non è bilanciato. L'assunzione di base che viene fatta è che alcune osservazioni sono ridondanti. Il rischio è quello di rimuovere osservazioni rilevanti soprattutto se fatto con metodi non supervisionati. Random oversampling consiste nell'aumentare il numero delle istanze della classe minoritaria duplicando i dati presenti fintanto che non si hanno le stesse distribuzioni tra le classi. Uno dei rischi principali è quello di aumentare l'overfitting del modello sulla classe minoritaria[27]. Un altro problema è dato dal fatto che tramite questo meccanismo non viene aggiunta informazione.

1.4.2 SMOTE

SMOTE [19] è un approccio dove l'oversample della classe minore viene fatto generando esempi "sintetici" e in seguito viene fatto un under-sampling della classe maggioritaria. Gli esempi artificiali vengono generati nel seguente modo: preso un punto (della classe minoritaria) si trovano i "k" elementi ad esso più vicini e si calcolano le distanze da ognuno di essi. Su ogni segmento viene generato in maniera randomica un nuovo record. Partendo da tale meccanismo ne sono derivati molti altri che si presentano come delle varianti di Smote [36], [9], [15]. Viene riportato in fig. 1.1 il meccanismo di funzionamento di SMOTE.

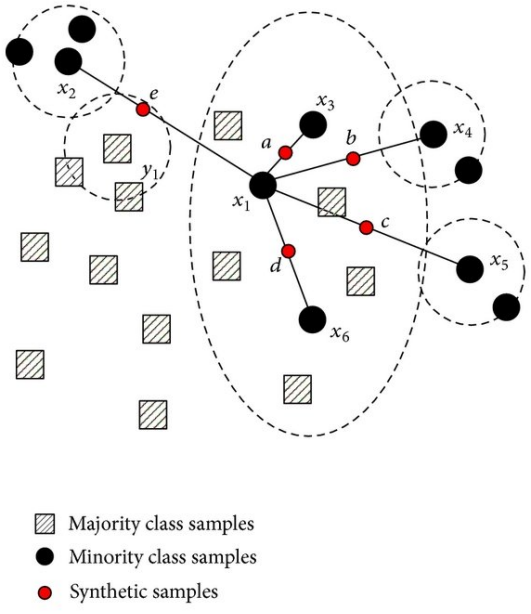

Figura 1.1: Applicazione di SMOTE

1.4.3 Adasyn

Adasyn [35] è un approccio che si basa sullo stesso principio di SMOTE[19] ma genera gli esempi sintetici in maniera differente. Per ogni record selezionato della classe minoritaria, vengono identificati i "k" record ad esso più vicini, il nuovo esempio che deve essere creato, non viene generato sull'asse che collega i due punti come in Smote, ma in un intorno dell'asse. Questo permette di avere più varianza nella distribuzione dei dati. Inoltre questo meccanismo segue una natura adattiva, vengono creati più esempi per i dati più difficili da separare. Vengono generati più punti nelle zone dove c'è una maggiore distribuzione di dati appartenenti alla classe maggioritaria.

1.4.4 SMOTE Tomek

Effettuare un over-sample della classe minore può bilanciare la distribuzione delle classi. Il problema è che potrebbero insorgere altri tipi di problemi, spesso i cluster di dati relativi ad ogni classe non sono ben definiti, gli esempi della classe maggioritaria potrebbero invadere lo spazio dei dati della classe minoritaria. In maniera analoga potrebbe avvenire l'inverso, i nuovi esempi della classe minoritaria vengono generati nello spazio della classe maggioritaria. Un classificatore che deve operare in queste situazioni potrebbe andare in overfitting. Nell'ottica di voler creare cluster di dati meglio separabili, in [10] viene proposto di applicare Tomek links al set di dati sui cui è stato fatto l'oversampling. Invece di rimuovere solamente gli esempi della classe maggioritaria, Tomek link rimuove elementi di entrambe le classi in modo da avere i cluster delle due classi ben definiti. Il processo di oversmpling della classe minoritaria può essere eseguito con SMOTE. Viene mostrato in fig. 1.2 il processo di lavoro.

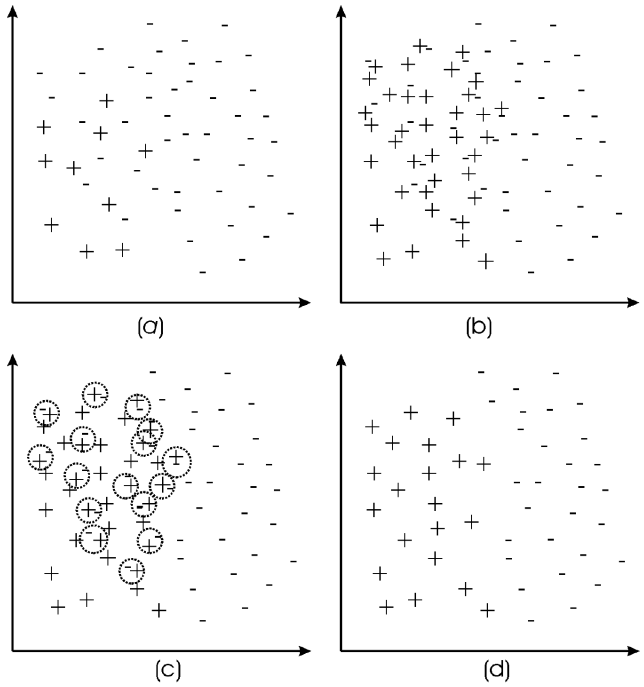

Figura 1.2: (a) Dataset originale. (b) Bilanciamento del dataset con SMOTE. (c) Vengono identificati i tomek link. (d) Rimossi gli esempi che aggiungono rumore.

Tomek link

Tomek Link [5] è un meccanismo di pulizia dei dati e rimozione del rumore. Dati 2 esempi E_i e E_j appartenenti a 2 classi differenti, si definisca $d(E_i, E_j)$ la loro distanza. Una coppia (E_i, E_j) è chiamata Tomek link, se non esiste un elemento E_k, tale che $d(E_i, E_k) < d(E_i, E_j)$ oppure che $d(E_j, E_k) < d(E_i, E_j)$. Due istanze formano un Tomek Link se: uno dei due elementi è "rumore" oppure entrambi si trovano vicino al confine che suddivide le classi. Questi dati possono essere rimossi. Tomek link può essere visto come meccanismo per la pulizia dei dati e la rimozione del rumore tra le classi.

1.4.5 SMOTE ENN

SMOTE ENN [10] è meccanismo è simile a SMOTE Tomek links. La differenza è che ENN tende a rimuovere più esempi di Tomek Link, può essere considerato come un metodo di pulizia più radicale.

Edited Nearest Neighborhood (ENN)

Neighborhood Cleaning Rule (NCL) [41] usa *Wilson's Edited Nearest Neighbor rule (ENN)* [56] per rimuovere gli esempi della classe maggioritaria. ENN rimuove ogni esempio la cui etichetta è diversa rispetto i suoi due elementi più vicini. Per ogni elemento E_i del training set, vengono identificati i 3 record più vicini. Se E_i appartiene alla classe maggioritaria e la classificazione fornita dai 3 record più vicini contraddice la classe originale di E_i, allora E_i viene rimosso. Se E_i appartiene alla classe minoritaria, e la classifizione assegna un altro valore di classe, allora sono rimossi i 3 elementi della classe maggioritaria.

1.5 Metriche di valutazione

Le metriche di valutazione vengono utilizzate per avere una misura quantitativa relativa all'abilità del modello di machine learning nel risolvere il compito per il quale è stato progettato. Una delle metriche di valutazione più usate è l'accuratezza. Tramite questa metrica è possibile avere una misura su quanti record siano classificati correttamente dal modello predittivo. Viene calcolato il rapporto tra i record classificati correttamente per ogni classe e i record totali che erano da classificare. Il valore è compreso tra 0 e 1, più il valore è vicino all'1 e più la classificazione è efficace. Tuttavia in problemi dove la distribuzione dei dati tra le classi non è bilanciata, questa misura può non essere adatta [34]. Altre metriche che si basano sui valore della matrice di confusione [8] possono essere: precisione, recall e F-measure. Precisione e recall hanno due andamenti opposti, riuscire ad alzare il valore di una delle due misure ne determina la diminuzione dell'altra. F-measure da pari importanza a precisione e recall, la f1-measure ad esempio è la loro media armonica. Può essere considerabile come una metrica di riferimento in problemi con distribuzioni di dati tra le classi sbilanciate [18]. Un'altra misura accettata per valutare dataset di questo tipo è la AUC [18].

1.6 Big data

Tutti gli approcci e i meccanismi di analisi presentati devono essere utilizzati in un sistema reale che effettui la previsione e il riconoscimento di frodi. Un sistema progettato a questo scopo deve tenere in considerazione diversi requisiti. Deve essere progettato in modo da avere un input costituito da un flusso continuo di dati. A fronte di ciascun input, il sistema deve fornire una risposta in tempo reale. È richiesto quindi che sia reattivo e che riesca a gestire grandi quantità di dati con una bassa latenza. Per ottemperare ai requisiti richiesti è opportuno fare riferimento a un sistema distribuito che funzioni su un'infrastruttura big data. Un

esempio di come sia possibile creare un'infrastruttura di questo tipo viene data in [21] dove viene descritta l'infrastruttura Hadoop per il riconoscimento delle frodi in Alibaba.

1.6.1 Un'implementazione reale

Nel lavoro descritto in [16] viene proposta un'implementazione scalabile e realistica di un sistema atto al riconoscimento delle frodi. Il progetto comprende le diverse fasi di progettazione, implementazione e test di un'infrastruttura che integri diversi componenti facenti parte dell'ecosistema Apache. L'architettura deve gestire un input composto da grandi quantità di dati in arrivo in tempo reale, gestire operazioni di feature engineerig, memorizzazione e classificazione delle transazioni bancarie. Deve essere una soluzione scalabile che fornisca un risultato attendibile. Si pensi che in un sistema finale vi possano essere due livelli di controllo tra loro integrati. Un sistema automatico basato su un approccio di machine learning e un sistema manuale di analisi delle transazioni fatto da personale specializzato. Se una transazione viene classificata come fraudolenta non deve essere accettata dalla banca.

1.6.2 Stack tecnologico

Nella soluzione descritta in [16] viene proposto uno stack tecnologico basato sull'unione di diversi componenti quali: Kafka, Spark e Cassandra.
Kafka è un sistema basato sulla gestione distribuita di code di messaggi. Può produrre e convogliare dati provenienti da diverse sorgenti. In generale Kafka può essere considerato come uno strumento utile per inviare grandi quantità di dati con una bassa latenza [37]. Apache Spark è un motore d' esecuzione che esegue computazioni in memoria. Comprende una libreria per il machine learning: MLib [42] e per lo streaming. Tra le potenzialità di Spark vi è il fatto che rende possibile effettuare analisi sia di tipo batch che streaming. Spark può essere utilizzato per svolgere diversi compiti: aggregazione dei dati ai fini del feature engineering e classificazione. Cassandra è un database progettato per essere scalabile e distribuito. I dati sono memorizzati su diversi nodi all'interno della rete, evitando così di avere un single point of failure.

Capitolo 2

Definizione di un modello comportamentale

In questo capitolo verranno descritte le diverse tipologie di analisi che si possono eseguire su dati di transazioni bancarie. Tali metodologie verranno poi applicate su due dataset differenti e i risultati verranno discussi nei prossimi capitoli. Le seguenti tecniche sono il prodotto di un'analisi della letteratura, analisi del dominio del problema e un contributo personale. Si vuole mostrare una soluzione nel risolvere il problema del riconoscimento delle frodi modellando il comportamento degli utenti.
Partendo da transazioni bancarie è richiesto di produrre un risultato che identifichi una transazione come "legittima" o "fraudolenta". Per fare questo è opportuno analizzare una transazione bancaria e sulla base di una serie di metodologie determinarne la classe di appartenenza. Il problema può essere approcciato con tecniche di machine learning classiche, si può addestrare un modello e applicarlo ai dati. Spesso però, questa tecnica si può rivelare poco efficace. Il motivo è da ricercare nel fatto che le informazioni presenti relative a una singola transazione potrebbero non essere sufficienti per il riconoscimento di pattern fraudolenti.

2.1 Tipologie di Analisi

L'analisi di una transazione ai fini della classificazione può essere eseguita in diversi modi, ne vengono proposti due:

- Analisi a livello utente.
- Analisi singola transazione.

Sono due approcci differenti ognuno con i suoi punti di forza e debolezza. Ogni approccio fa delle asserzioni che possono essere più o meno valide, in ogni caso

con entrambe le soluzioni possono essere raggiunti risultati interessanti. A volte è necessario adottare un tipo di analisi piuttosto che un'altra poichè le informazioni presenti nei dati da classificare non permetterebbero di fare diversamente.

2.1.1 Analisi a livello di singola transazione

Con "analisi a livello di singola transazione" si intende un'analisi che basa la classificazione della transazione mettendo questa in relazione con tutte le transazioni che compongono il dataset. Per fare questo possono essere utilizzati diversi algoritmi di machine learning. Un esempio potrebbe essere *K-NN*: la classificazione viene effettuata basandosi sulle transazioni che compongono il dataset più simili alla transazione in oggetto. In questo modo se una transazione è simile a delle frodi c'è una probabilità che possa essere considerata tale, viceversa si tratterebbe di un' operazione legittima. Questo meccanismo permette di riconoscere pattern di dati basandosi su delle similarità tra oggetti della stessa classe. Tale meccanismo viene utilizzato da diversi algoritmi; *svm* divide uno spazio n-dimensionale in modo da separare i dati con etichette differenti. *Decision Tree* esamina le frodi e cerca di creare dei percorsi differenti in funzione della tipologia di pattern osservato. Verrà creato un albero decisionale con diversi percorsi dalla radice alle foglie. Ogni foglia viene assegnata ad una classe specifica e ogni percorso che parte da una radice e arriva a una foglia determina un meccanismo di classificazione. Possono esserci diverse foglie che rappresentano le frodi, questo determina il fatto che non esiste un pattern unico di riconoscimento di una frode, al contrario identifica il fatto che possono essercene diverse tipologie. Questo potrebbe essere uno dei motivi per il quale algoritmi come *Decision Tree* e *Random Forest* raggiungono i risultati migliori.

2.1.2 Analisi a livello utente

L'analisi a livello di singola transazione basa la previsione classificando una transazione alla classe delle transazioni a questa più simili. Si assume in maniera implicita che tutte le frodi debbano essere tra loro più simili rispetto a quanto lo possano essere nei confronti di operazioni legittime. Questa assunzione può essere considerata vera ma entro determinati limiti. Una transazione relativa a una frode bancaria a danno di una persona con un grosso patrimonio, potrebbe essere molto differente da quella subita da una persona con un patrimonio più esiguo. Si pensa che le somme monetarie coinvolte e le modalità di attacco possano avvenire in maniera differente nelle casistiche presentate. Motivo per il quale è necessario raffinare il concetto di "transazione fraudolenta/legittima" trasformandolo in qualcosa di più simile a: "transazione legittima/fraudolenta per l'utente *XYZ*". Questo tipo di analisi mira a mettere in relazione la transazione da analizzare con

tutte le altre operazioni eseguite dallo stesso utente. Per effettuare questo tipo di analisi è opportuno avere a disposizione uno storico dati per ogni utente altrimenti non sarebbe possibile effettuare un confronto. Si consideri inoltre che a differenza del precedente approccio, non si potrà confrontare la frode con le altri frodi subite dallo stesso utente perchè si assume che in uno scenario normale un utente non abbia uno storico di frodi a proprio carico. L'idea di base, quindi, è quella di confrontare una transazione di un utente con le sue transazioni passate e sulla base di questo confronto far emergere una relazione tra il comportamento attuale e un comportamento definibile come "standard" per un utente. Il modello addestrato a differenza del modello precedente andrà a effettuare la classificazione basandosi sulla relazione tra il comportamento che emerge all'interno di una transazione in relazione al comportamento abituale tenuto nelle transazioni passate. È importante mostrare che di base viene fatta l'assunzione che un utente abbia uno storico di transazioni e soprattutto effettui più o meno gli stessi tipi di transazioni in modo che si possa identificare un suo profilo.

2.1.3 Quale analisi scegliere

Come descritto, per ogni tipologia di analisi vengono fatte delle assunzioni che possono essere considerabili da un esperto di dominio più o meno valide. Si potrebbe analizzare la bontà di ciascuna analisi basandosi sui risultati raggiunti, questo tipo di confronto sarà frutto dei prossimi capitoli. In ogni caso quando si decide su quale tipologia di analisi fare riferimento, è opportuno considerare il tipo di dato sul quale si deve lavorare. Se non sono presenti informazioni relative all'utente che ha svolto una determinata transazione, o non si ha uno storico dati, si è costretti ad adottare un approccio sulle singole transazioni a meno di non riuscire a ottenere in qualche altro modo le informazioni necessarie.

Dati non in chiaro

Ricercando in rete e consultando alcuni tra i principali repository di dataset come Kaggle[1] o Uci[2], si evince come questa tipologia di dato sia spesso fornita dopo fasi di anonimizzazione e opacizzazione dei dati e delle informazioni. Sono dati estremamente sensibili e per motivi di privacy non possono essere resi pubblici senza essere oscurati. Dovendo lavorare con un dataset in questo formato l'unica possibilità è quella di analizzare le singole transazioni non potendo ricostruire la storia di un utente. Un'alternativa potrebbe essere quella di aggregare i dati al livello di dettaglio interessato cercando di effettuare un lavoro di *reverse engineering* partendo dai dati anonimizzati. Questo procedimento può dipendere dalla

[1] `https://www.kaggle.com/`
[2] `https://archive.ics.uci.edu/ml/index.php`

qualità del processo di anonimizzazione che è stato usato e può produrre risultati più o meno attendibili. Si identifichino tre livelli di mascheramento dei dati:

- Singolo valore.
- Intera feature.
- Gruppi di feature.

Sulla base del livello di opacizzazione dei dati si può decidere come affrontare il problema. L' analisi su ogni singola transazione può essere usata anche nei casi di mascheramento più marcato. Il motivo risiede nel fatto che se l'operazione di mascheramento è fatta nel modo corretto, elementi simili in un spazio non trasformato mostreranno una relazione anche in uno spazio trasformato.

Singolo Valore

I valori per una determinata feature sono scalati rispetto specifiche costanti. Non si conosce il reale valore assunto ma sono mantenute determinate proprietà di relazioni tra i dati. Esempio: se una somma in denaro è 10x rispetto un'altra, anche se i valori sono trasposti su una scala differente questa relazione rimane apprezzabile.

Intera Feature

Oltre a mascherare il reale valore del dato, viene mascherato il nome della feature. In questo caso si avranno delle feature rinominate come "X-1", "X-2" e non si può sapere cosa stiano a rappresentare.

Gruppi di Feature

Simile alla casistica precedente, in aggiunta vi è il fatto che alcune feature sono la combinazione di altre che sono state rimosse. Può essere usato l'algoritmo *PCA*[6] per produrre nuove feature partendo da un dataset in chiaro.

Dati in chiaro

Nel caso si abbiano i dati in chiaro, o si riesca a rifarsi a questa casistica si può decidere l'approccio da utilizzare, tuttavia potrebbe valere la pena classificare le transazioni analizzando il comportamento degli utenti.

2.2 Feature Engineering

Se siamo nella casistica secondo la quale non si riesce ad attribuire un significato ai dati, il meccanismo di lavoro porterà a un modello difficilmente interpretabile, non potendo comprendere la natura dei dati, sarà ancora più difficile comprendere per quali motivi reali una transazione viene classificata come frode. L'impossibilità di comprendere la natura della classificazione avviene anche per tutte quelle tecniche che offrono modelli *black-box* come gli approcci neurali o algoritmi quali *svm*. Questi modelli si contrappongono a quelli *white-box* tramite i quali è possibile avere una motivazione del perchè una transazione viene classificata in un determinato modo. Capire il processo decisionale può essere importante nel momento che si devono giustificare e spiegare certe soluzioni. Un modello potrebbe inferire in maniera automatica determinate regole che un esperto di dominio potrebbe smentire, in questi casi deve essere opportuno intervenire e correggere tali inesattezze. Nel caso si lavori con dati mascherati e non si riesca ad avere una narrativa relativa alle feature e al valore assunto, non si può fare molto per avere risultati interpretabili. Il modello si baserà sulle informazioni disponibili e su correlazioni implicite che ogni specifico algoritmo di machine learning riuscirà a evidenziare. Questo meccanismo potrebbe non essere sufficiente e le conoscenze implicite identificate dal modello potrebbero non essere verificabili. Potendo lavorare su dati in chiaro, guidati da un esperto di dominio si possono analizzare le distribuzioni dei dati nelle diverse classi. Queste operazioni potrebbero essere utili per avere un processo tramite il quale si estraggono informazioni dai dati e successivamente si addestra un modello su dati con un contenuto informativo aggiuntivo. Tramite questo meccanismo si possono misurare dei miglioramenti nell'efficacia, grazie al contributo dei nuovi dati inseriti. Questo procedimento prende il nome di **Feature engineering**.
Fare feature enginering può essere utile per aumentare e fare emergere le informazioni implicite presenti nei dati. Possono esserci diverse tecniche di feature engineering. Basandosi sul dominio di interesse si può pensare di voler esplicitare una serie di informazioni che permettano di relazionare una transazione con tutte le altre appartenenti allo stesso utente. Ciò che deve emergere da queste relazioni è il fatto che una determinata feature o gruppo di feature possano essere simili o diverse da una linea guida "di massima" che identifica un comportamento abituale o standard di un utente.

2.2.1 Strategia di classificazione con nuove feature

Effettuando un'analisi a livello utente è possibile mettere in relazione ogni singola transazione con la storia passata dello stesso utente. L'assunzione di base che viene fatta è che se una transazione è in linea con il comportamento che l'utente ha tenuto fino a quel momento, allora probabilmente l'operazione è legittima, vi-

ceversa, se la transazione è molto differente rispetto alla storia passata, allora vi è più probabilità che possa trattarsi di una frode. Questo tipo di informazione viene aggiunta andando ad estendere la transazione con una serie di feature aggiuntive che ne rappresentino una relazione con la storia passata. Successivamente con i dati così trasformati può essere addestrato un modello di machine learning. Il modello predittivo baserà la classificazione focalizzandosi sulle nuove feature aggiunte. Vi è una differenza tra il modello addestrato su questo tipo di dati e quello che considera transazioni normali. Basandosi su dati non trasformati la classificazione si baserà su un fattore di similarità e somiglianza tra elementi della stessa classe. Anche in questa nuova casistica l'algoritmo rimane lo stesso quindi il procedimento è il medesimo, ma cambiano le informazioni sulle quali viene effettuata la classificazione. Un modello addestrato su transazioni normali, analizza informazioni del tipo :
" transazione bancaria effettuata in una determinata regione, con un particolare browser su un determinato device mobile e coinvolge una somma in denaro".
Semplificando ciò che succede, il modello analizza dati di questo tipo e potrebbe inferire che ci sono "browser" sui quali è più probabile che si verifichino delle irregolarità oppure che determinate combinazioni tra "tipologie di device","somme coinvolte","Browser" sono più inclini a verificarsi in operazioni fraudolente. Effettivamente possono esserci relazioni di questo tipo, e questo meccanismo permetterebbe di catturarle, però l'idea di legare una specifica transazione con la storia passata di un utente permetterebbe di avere transazioni più complesse ed elaborate. Si pensi piuttosto di poter analizzare transazioni così trasformate:
" transazione bancaria effettuata in una determinata regione diversa da quella nella quale l'utente è solito operare, viene utilizzato un determinato tipo di device mobile e browser che non sono mai stati utilizzati dall'utente ed è coinvolta una somma in denaro molto più alta rispetto il solito".
L'analisi viene sempre fatta a livello di transazione con il medesimo algoritmo, ma le informazioni a disposizione per tracciare un confine tra "legittimo" e "fraudolento" sono differenti. Ciò che si cerca di far emergere non è tanto il fatto che per esempio un'operazione venga eseguita con un determinato dispositivo mobile, così sarebbe difficile definirla come frode. Piuttosto, è più interessante il fatto che potrebbe essere una frode se quello specifico dispositivo non è mai stato utilizzato da quell'utente per fare transazioni bancarie. Il processo è più complesso dell'esempio riportato, non verrà coinvolta una sola feature, ne verranno aggiunte diverse. Ciò che viene fatto emergere all'interno di una transazione è una relazione di abitudinarietà tra il comportamento presentato in una transazione e quello avuto nelle transazioni passate.
L'obiettivo principale è quello di far emergere relazioni tra un comportamento osservato in una transazione e il comportamento avuto nelle transazioni passate. Per

fare questo è opportuno analizzare lo storico dati di ogni utente e identificare un comportamento standard. Infine mettere in relazione i comportamenti. Si deve tenere conto del fatto che un utente possa cambiare comportamento nel tempo. È pertanto utile analizzare i dati su finestre temporali di una certa ampiezza.

2.2.2 Definizione di un comportamento standard

Per identificare un comportamento definibile "standard" è opportuno analizzare la storia di utente ed estrarre una serie di informazioni che definiscano il comportamento più diffuso. Viene così identificata una linea guida di massima che rappresenta le abitudini di un utente. Essendo presenti diverse tipologie di feature, è opportuno definire cosa significhi trovare un comportamento abituale per valori di feature categoriche e numeriche. Si pensa che definire un valore abituale per una feature categorica possa essere fatto esaminando l'istanza che occorre con una maggiore frequenza. Per una feature numerica invece, può essere usato un indice statistico, per esempio il valore medio.

2.2.3 Analisi su finestre temporali

È opportuno considerare che il comportamento mostrato in una transazione debba essere messo in relazione con la storia passata di quell'utente. Potrebbe però essere più rilevante piuttosto che analizzare l'intera storia, fare un confronto basandosi solamente su valori più recenti o in maniera più generale: su finestre temporali di una determinata ampiezza. Un esempio potrebbe essere valutare se una somma in denaro coinvolta in una transazione sia in linea con il comportamento tenuto dall'utente nell'ultima settimana piuttosto che confrontarla con l'intero storico dati disponibile. Per fare questo si può pensare che per ogni transazione sia opportuno effettuare un confronto solamente con le informazioni relative a finestre di dati che facciano riferimento a un Δ "t" di tempo finito antecedente la transazione. Un esempio potrebbe essere effettuare il confronto solamente con tutte le transazioni avvenute nei 7 o 30 giorni precedenti.

2.2.4 Analisi sulla frequenza delle attività

Una possibile analisi nel voler identificare anomalie comportamentali potrebbe essere quella di studiare la frequenza delle attività. Un'anomalia potrebbe essere identificata sulla base della frequenza con la quale avvengano certe operazioni. Questa informazione potrebbe essere più interessante se combinata con finestre temporali (descritte nel precedente paragrafo). Un esempio sul contributo informativo fornito: contare il numero di transazioni effettuate da un utente nella

precedente settimana. Se un utente è solito effettuare un numero definito di transazioni e improvvisamente in una settimana ne esegue un numero molto più elevato, questo potrebbe essere sinonimo di anomalie da verificare. Questa ricerca può essere resa più interessante se viene specificata una feature sulla quale effettuare l'analisi. Si specializza l'esempio: si valutino le relazione che potrebbero emergere se venisse considerata anche la somma in denaro coinvolta nella transazione. Si avrebbe qualcosa del tipo:
"Numero di operazioni eseguite nell'ultimo periodo che hanno coinvolto somme in denaro maggiori di una certa cifra". Si consideri che l'esempio così riportato possa essere più interessante. Identificare un'anomalia di questo tipo in operazioni dove sono coinvolte somme in denaro considerevoli è diverso rispetto farlo su spostamenti di piccole somme.

2.2.5 Analisi sugli scostamenti

Nel confronto tra il comportamento manifestato in una transazione e quello definibile "standard" è importante considerare se certi valori tra le feature sono gli stessi. Per esempio se il dispositivo mobile con il quale si è eseguita l'operazione è quello abituale. Sarebbe però sbagliato estendere lo stesso ragionamento a valori di feature numeriche. Non avrebbe senso confrontare che la somma coinvolta in una transazione sia la stessa delle precedenti. Per questo motivo per feature quantitative è opportuno effettuare un'analisi relativa agli scostamenti dei valori da una misura di riferimento. Per le feature categoriche nominali non è possibile misurare quanto un valore si discosti dalla moda. Per feature numeriche invece, può essere interessante evidenziare se un valore di discosta oltre certe soglie rispetto un valore di riferimento (che potrebbe essere il valore medio della feature nelle transazioni passate). Come per gli altri tipi di analisi può essere interessante combinare questa misura con un risultato ottenuto su finestre temporali ed esaminandone la frequenza.

2.3 Estendibilità dell'analisi

L'analisi basata sugli utenti, può essere fatta in maniera differente mostrando due possibili approcci. Il modello predittivo può essere ottenuto effettuando l'addestramento sui dati di tutti gli utenti storici. In questo caso, il modello, in fase di classificazione avrà già visto tutti i possibili utenti. In contrapposizione a questo, si potrebbe pensare di avere un modello che non per forza sia stato addestrato sui dati di tutti gli utenti che dovranno essere analizzati.
Sono due tipi di analisi differenti. Nella prima tutti gli utenti sono noti quando viene addestrato il modello e successivamente verranno classificate transazioni di

utenti i cui dati sono serviti per l'addestramento. Nella seconda analisi, vengono classificate transazioni di utenti le cui informazioni non hanno fatto parte dei dati usati per l'addestramento. In entrambi i casi deve essere presente uno storico dati per ogni utente.

2.3.1 Analisi di clienti storici

Un modello addestrato sui dati di alcuni utenti è specifico nel classificare le loro transazioni. Un sistema del genere potrebbe essere utilizzato in un contesto di funzionamento dove non si hanno nuovi clienti e in fase di addestramento del modello si conoscono già tutti gli utenti e per ognuno di essi si hanno una serie di dati. Un meccanismo utile per il riconoscimento di frodi all'interno delle operazioni di clienti storici.

2.3.2 Analisi di nuovi utenti

Un modello potrebbe essere addestrato su un insieme di clienti ma riuscire a estendere il criterio di classificazione anche a utenti mai visti. Questa assunzione può essere fatta perchè il modello basa l'addestramento sulle informazioni aggiuntive fornite che sono il frutto dei meccanismi di feature engineering presentati. Ogni nuova transazione ha le informazioni relative alla transazione stessa e in aggiunta una serie di feature che ne identifichino la relazione rispetto il comportamento standard. L'algoritmo di machine learning baserà l'apprendimento cercando di riconoscere e fare emergere specifiche relazioni che esistono tra l'etichetta della classe e i nuovi dati aggiunti. Volendo semplificare questo concetto e fare un esempio, si pensi che un utente esegua 9 transazioni e poi subisca una frode. Se si vanno ad analizzare le 9 transazioni trasformate si potrebbe osservare che le informazioni aggiuntive hanno tra loro una certa similarità, questo perchè ognuna delle 9 transazioni è più o meno in linea con il comportamento standard dell'utente. La decima transazione invece è una frode; se si vanno ad analizzare i campi aggiuntivi si potrebbe notare che questi sono decisamente diversi dalle feature aggiunte nelle precedenti 9 transazioni legittime. Si potrebbe generalizzare affermando che nelle informazioni aggiunte vi è un elemento che potrebbe essere simile in operazioni della stessa classe. A fronte di questi ragionamenti si può pensare che in fase di addestramento del modello non siano importanti gli utenti specifici, piuttosto è più rilevante cercare di catturare il maggior numero possibile di relazioni diverse che possono sussistere tra una transazione e le transazioni pregresse.

2.4 Metriche di valutazione

Una delle metriche che vengono spesso usate in machine learning per valutare la bontà di un modello è l'accuratezza. In questo dominio non è opportuno usare tale metrica perchè non rappresenta un valore reale che rispecchi la qualità della classificazione. Il motivo è dato dal fatto che spesso si lavora con dataset sbilanciati, dove la distribuzione dei dati tra le classi non è la stessa. Banalmente considerando un dataset con l' 1% di frodi e il 99% di transazioni legittime, si addestri un modello che predica sempre : "Non frode". Tale modello avrebbe un'accuratezza del 99% nonostante non riesca a riconoscere nemmeno una frode. Per tale motivo si faccia riferimento a metriche differenti come precisione e recall.

2.4.1 Precisione e recall

Si considerano entrambe le metriche riferite alla classe "frode". L'obiettivo deve essere quello di ottimizzare sia la precisione che la recall, bisogna però considerare il fatto che le due metriche sono tra loro correlate e nell'aumentare una si rischia di diminuire il valore dell' altra e viceversa. Ottimizzare la precisione avrebbe un determinato effetto sul risultato, ottimizzare la recall un altro. Una soluzione ottimale sarebbe quella di avere i benefici di una precisione alta uniti a quelli di un'alta recall.

2.4.2 Ottimizzazione della precisione

La precisione è una misura che si basa sulla quantità di falsi positivi che vengono identificati. Una precisione alta starebbe a significare che si è confidenti sul fatto che una transazione classificata come frode sia effettivamente una frode. Ridurre il numero di falsi positivi per la classe frode porterebbe a un sistema che limita il numero di falsi allarmi. Se un'operazione è prevista come frode allora probabilmente lo è. Volendo essere sicuri di avere una precisione alta, si potrebbe rischiare di classificare come frodi solo quei dati sui quali si è molto sicuri, questo porterebbe però a identificare come non frodi molti dati sui quali vi è incertezza, sicuramente tra questi vi saranno delle frodi che non si riusciranno a identificare. Il prezzo per un'ottimizzazione della precisione potrebbe essere quello di ottenere un elevato numero di falsi negativi e quindi andare ad abbassare la recall.

2.4.3 Ottimizzazione della recall

La recall è una misura che si basa sulla quantità di falsi negativi identificati. Una recall alta starebbe a significare che si è confidenti di avere identificato una buona parte delle frodi. Ridurre il numero di falsi negativi per la classe frode, porterebbe

a un sistema che limita il numero di frodi non identificate. Se un'operazione viene identificata come "non frode" allora probabilmente non è una frode. Volendo essere sicuri di avere una recall alta, si potrebbe rischiare di classificare come frodi anche tutte quelle transazioni sulle quali vi è incertezza, sicuramente tra questi dati vi saranno transazioni legittime. Il prezzo per un'ottimizzazione della recall potrebbe essere quello di ottenere un elevato numero di falsi positivi e quindi andare ad abbassare la precisione.

2.4.4 Valutazioni metriche

Si cerca di ottimizzare la precisione quando si vuole essere sicuri che quando una transazione viene identificata come frode, questa lo sia veramente con un buon livello di certezza, minimizzando così i falsi allarmi. Si cerca di ottimizzare la recall quando si vuole essere sicuri di identificare quasi tutte le frodi con la consapevolezza che alcune segnalazioni saranno falsi allarmi. La scelta se ottimizzare una o l'altra dipende molto dal dominio applicativo reale e da come il sistema verrà utilizzato. Per questo motivo sono ritenute importanti entrambe le misure e si pensi di poter identificare diversi modelli che riescano ad adattarsi alle specifiche esigenze reali. È quindi importante massimizzare la precisione o la recall sulla base del caso d'uso ma tenendo sempre un occhio di riguardo a entrambe. Un modello con il 95% di precisione e il 4% di recall non è un buon modello perché identifica pochissime frodi (seppure con molta precisione). Allo stesso modo, nella situazione inversa, 95% di recall e 4% di precisione non è un buon risultato poiché le frodi vengono identificate quasi tutte, ma il motivo potrebbe essere dato dal fatto che la quasi totalità delle transazioni vengono classificate come frodi senza che ci sia un vero meccanismo di selezione.

Capitolo 3

Analisi a livello delle transazioni

In questo capitolo vengono riportati i risultati ottenuti da uno studio eseguito su un dataset reale. Viene fatta un'analisi a livello di singole transazioni.
Partendo da dati anonimizzati e oscurati si sono valutate le diverse tipologie di analisi che potessero essere eseguite. Valutando le metodologie proposte in sezione 2.1, si pensi di effettuare un'analisi a livello delle transazioni. Considerato il fatto che il significato dei dati è stato oscurato ai fini della privacy, non è possibile effettuare uno studio che permetta di analizzare i dati aggregati per utenti o la creazione di nuove feature che possano essere interpretabili da esperti di dominio. Il modello che verrà prodotto sarà per tale motivo un modello non interpretabile. Verranno testati diversi algoritmi di classificazione e applicate le metodologie presentate. Ai fini di voler quantificare la problematica del bilanciamento delle classi e valutare le implicazioni che ne derivano, verranno eseguite le stesse analisi addestrando i diversi modelli sia su training set sbilanciati, che bilanciati.

3.1 Il Dataset

Il dataset in esame[1] è rilasciato da Wordline e il Machine Learning Group dell'Università di Bruxelles.

3.1.1 Struttura

Il dataset è composto da 284 807 transazioni bancarie. Solamente una piccola parte di queste è una frode, le restanti operazioni sono registrate come legittime. Le frodi sono solo 492, ovvero lo 0.172% del totale.

[1] `https://www.kaggle.com/mlg-ulb/creditcardfraud`

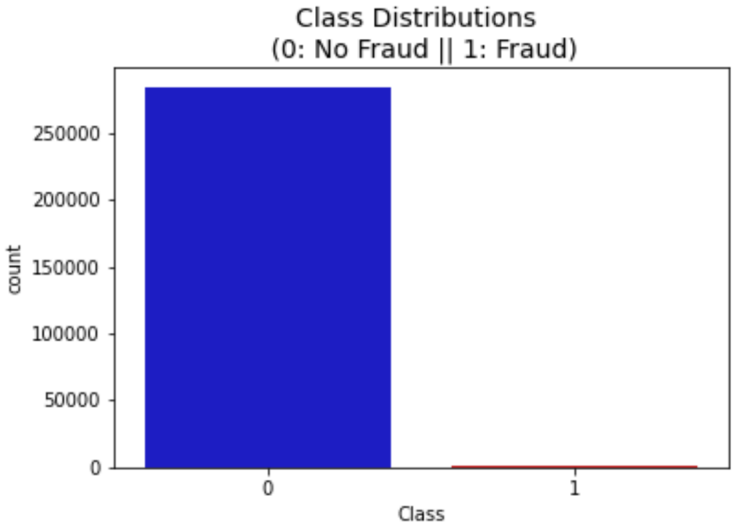

Figura 3.1: Distribuzione dei dati tra le classi.

Ogni transazione è composta da 30 feature numeriche. Come discusso in sezione 2.1.3, i dati sono resi disponibili dopo essere stati anonimizzati. I valori e le feature stesse sono il risultato di una trasformazione PCA operata sui dati originali. Le feature sono rinominate come "V_1, V_2 , ... , V_28", solamente due feature non sono state trasformate e sono in chiaro. Un'informazione relativa a un riferimento temporale in modo che sia possibile ordinare le transazioni e la somma in denaro coinvolta nell'operazione. Ogni transazione è etichettata come: frode / non-frode, per questo motivo possono essere utilizzate tecniche di addestramento supervisionate.

3.1.2 Train e Test

Il dataset originale viene diviso in set di test e set di train con un rapporto 1:4. Volendo eseguire diverse analisi verranno creati due differenti training set. Uno con i dati bilanciati tra le classi e uno invece con le distribuzione originali. Al fine di creare un test set bilanciato, si parte da quello sbilanciato e vengono applicate una serie di metodologie di bilanciamento. Per non invalidare i risultati, tutte le operazioni e le trasformazioni verranno applicate solamente sul training set. Non vengono eseguite operazioni di bilanciamento sul test set perchè non rispecchierebbe la distribuzione delle frodi in un contesto reale.

3.2 Pre-processing

Prima di addestrare un classificatore, è opportuno eseguire una serie di trasformazioni sui dati per far si che eventuali algoritmi di classificazione non soffrino di determinati problemi in fase di addestramento. Il fatto di trattare dati non normalizzati o non bilanciati può essere un problema per diversi algoritmi. È quindi opportuno eseguire una fase di preparazione dei dati prima che essi vengano utilizzati per l'addestramento di un modello. Ai fini di non voler invalidare i risultati che si otterranno, è opportuno dividere i dati in due set differenti: train e test. Tutte le analisi e decisioni verranno prese analizzando il solo set di train.

3.2.1 Metodi di normalizzazione

Il processo di normalizzazione può prevedere di scalare i dati su range di valori differenti rispetto quelli naturali. Per alcuni algoritmi come k-nn e svm che basano il loro funzionamento su relazioni di prossimità tra dati, è importante che le feature siano scalate. Attributi con differenti spazi di variazione andrebbero a influenzare il corretto processo decisionale dell'algoritmo. Al fine di standardizzare i dati vengono usati due differenti meccanismi di normalizzazione forniti dalla libreria scikit-learn[2].

Standard Scaler

Standard Scaler[3] trasforma i valori di ogni feature basandosi su media e deviazione standard. Dopo la trasformazione tutte le feature avranno deviazione standard = 1.

Robust Scaler

Robust Scaler[4] trasforma i dati sulla base del range interquartile IQR. IQR è il range tra il 1° quartile (25°quantile) e il 3° quartile (75°quantile). I dati mantengono la distribuzione originale ma in un range ridotto. Questa metodologia è più robusta nel gestire gli outlier.

[2] `https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler`

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html`

3.2.2 Valutazioni sull'efficacia

Si analizzano i risultati ottenuti con entrambe le metodologie di normalizzazione. Ai fini della valutazione bisogna considerare che il risultato dipende anche da altri fattori come: algoritmo di classificazione e metodo di bilanciamento. Per tale motivo le valutazioni sono fatte a parità di trasformazioni applicate sui dati.
In fig. 3.2 il confronto tra Standard Scaler e Robust Scaler su dati bilanciati con SMOTE e classificati con Random Forest.

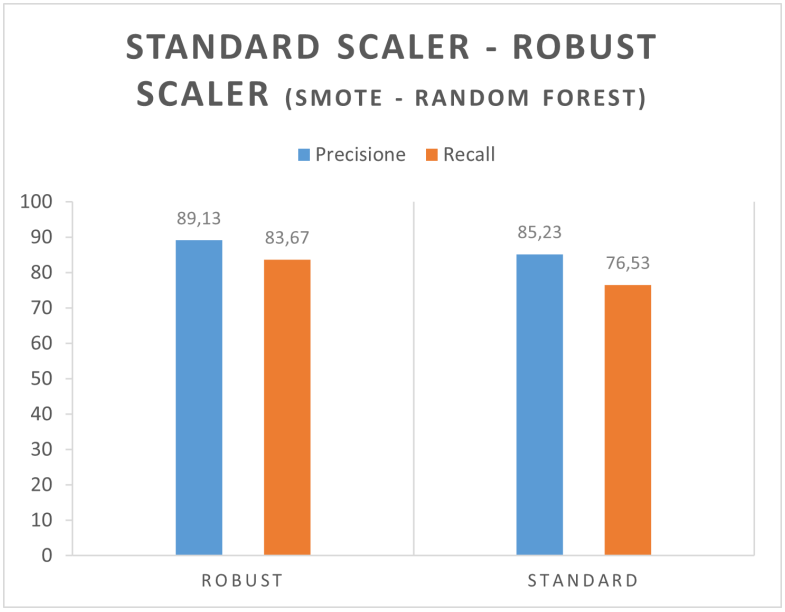

Figura 3.2: SMOTE, Random Forest. Robust scaler - Standard scaler

In fig. 3.3 il confronto tra Standard Scaler e Robust Scaler su dati bilanciati con SMOTE e classificati con Regressione Logistica.

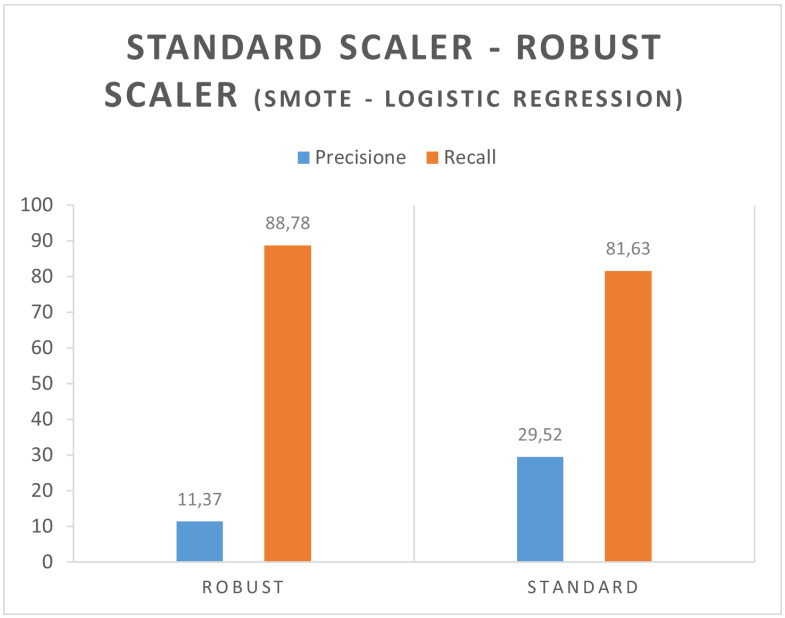

Figura 3.3: SMOTE, Regressione logistica. Robust scaler - Standard scaler

In fig. 3.4 confronto tra Standard Scaler e Robust Scaler su dati bilanciati con SMOTE e classificati con SVM.

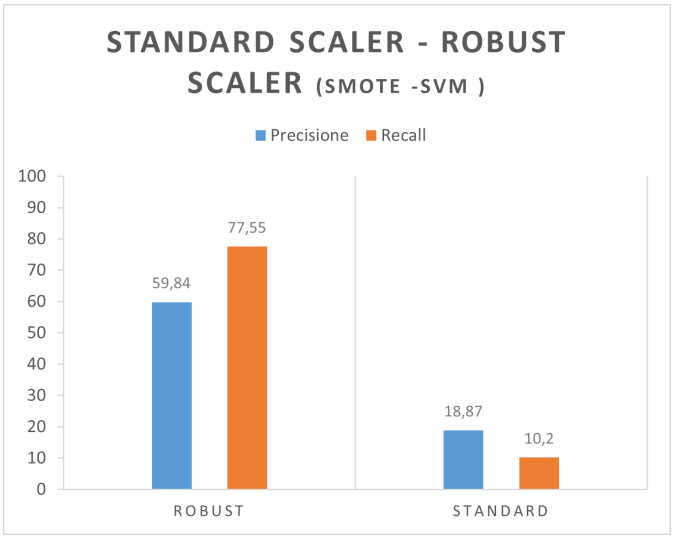

Figura 3.4: SMOTE, SVM. Robust scaler - Standard Scaler

In fig. 3.5 confronto tra Standard Scaler e Robust Scaler su dati bilanciati con SMOTE e classificati con XGBoost.

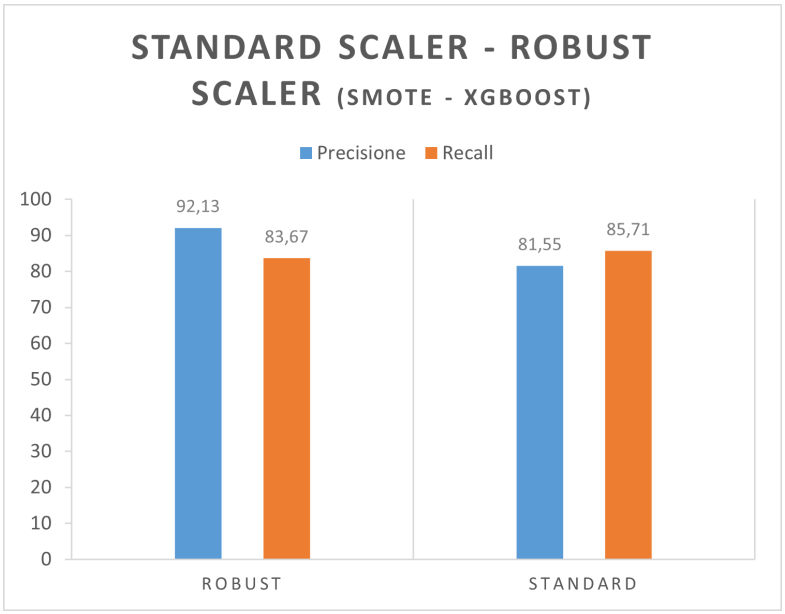

Figura 3.5: SMOTE, XGBoost. Robust scaler - Standard SCaler

Analizzando i risultati ottenuti, si osserva che alcuni algoritmi come SVM possono essere molto sensibili alla metodologia utilizzata. In Random forest il divario tra i risultati è marcato ma meno significativo rispetto SVM. La regressione logistica in generale ha problemi nel definire il confine tra le classi perchè riporta un risultato sulla precisione abbastanza basso, sembra però che grazie a Standard Scaler riesca a migliorare questo valore. Analizzando infine XGBoost si mostra come per questo algoritmo Robust Scaler riesca a migliorarne la precisione, mentre Standard Scaler la recall.
In conclusione si può dire che Robust Scaler permette di raggiungere risultati migliori, anche se possono esserci delle casistiche che si comportano in modo inverso. Si può valutare di considerare la metodologia che permette di raggiungere il risultato migliore in funzione degli specifici algoritmi di bilanciamento e classificazione.

3.3 Bilanciamento

Il problema del bilanciamento dei dati è spesso presente quando si lavora in questo dominio. All'interno del dataset solo lo 0.17% dei dati è relativo a frodi, la restante parte identifica transazioni legittime. Lavorando su tipi di dati quantitativi, può

essere necessario bilanciare le classi prima di addestrare il modello. Algoritmi di bilanciamento ve ne sono diversi tra i quali: SMOTE, Adasyn, SMOTE+Tomek e SMOTE+Enn. Il meccanismo principale che viene utilizzato, è quello di creare nuovi dati della classe minoritaria e/o diminuire i dati della classe maggioritaria. Nel gestire il problema del bilanciamento si è deciso di bilanciare solo i dati del training set in modo da non invalidare le osservazioni del test. Il modello dovrà lavorare in un contesto reale dove i dati sono altamente sbilanciati quindi ai fini della valutazione è opportuno lasciare le distribuzioni originali nel test set in modo che i risultati siano rappresentativi di un contesto realistico.
In fig. 3.6 vengono riportati i risultati con diversi metodi di bilanciamento classificando i dati con Random Forest.

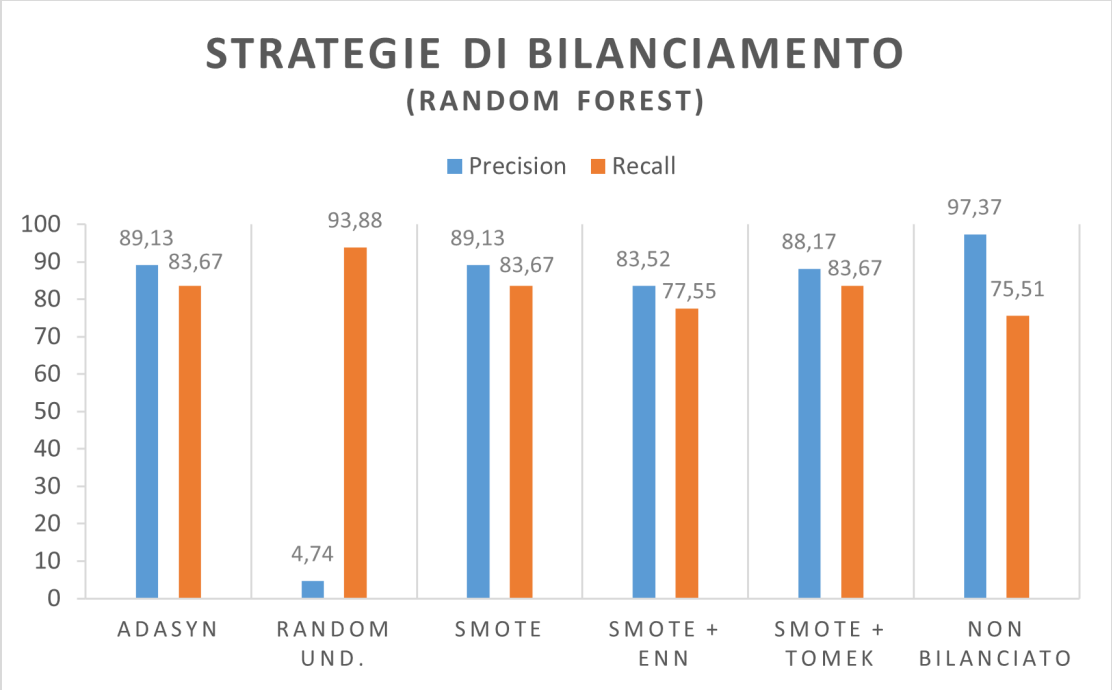

Figura 3.6: Metodi di bilanciamento con Random Forest

In fig. 3.7 vengono riportati i risultati con diversi metodi di bilanciamento classificando i dati con Decision Tree.

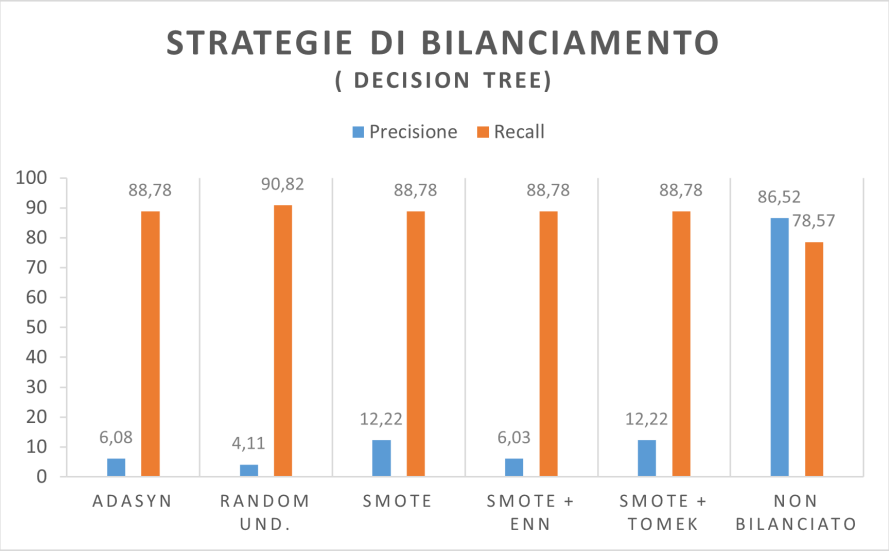

Figura 3.7: Metodi di bilanciamento con Decision Tree

In fig. 3.8 vengono riportati i risultati con diversi metodi di bilanciamento classificando i dati con la regressione logistica.

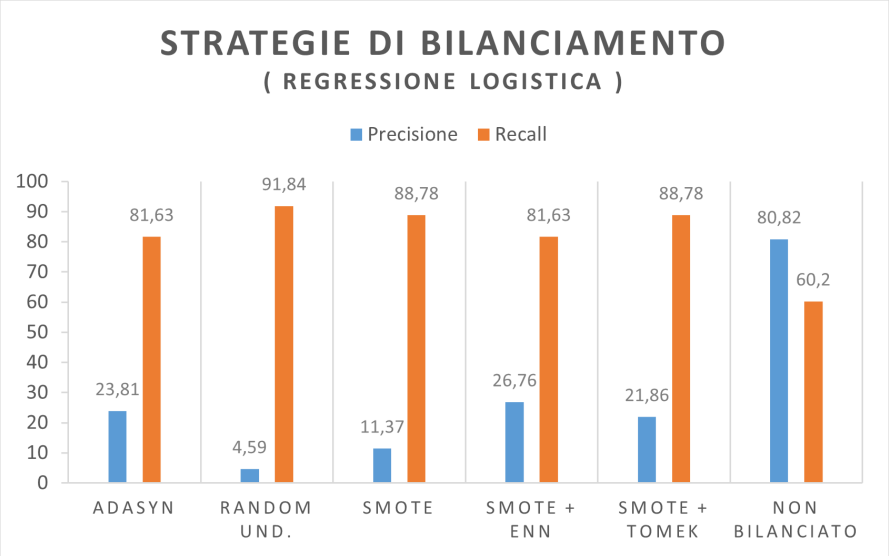

Figura 3.8: Metodi di bilanciamento con la regressione logistica

In fig. 3.9 vengono riportati i risultati con diversi metodi di bilanciamento classificando i dati con SVM.

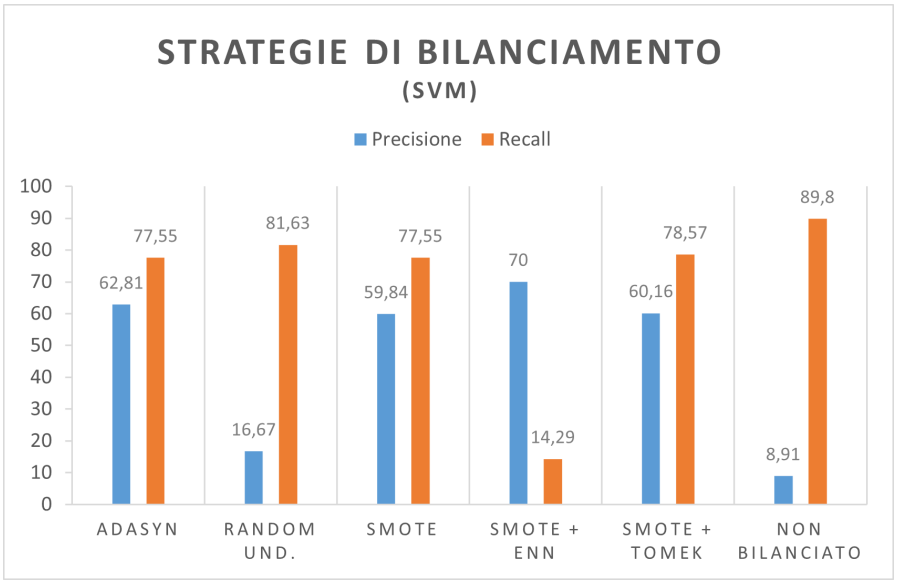

Figura 3.9: Metodi di bilanciamento con svm

In fig. 3.10 vengono riportati i risultati con diversi metodi di bilanciamento classificando i dati con K-NN.

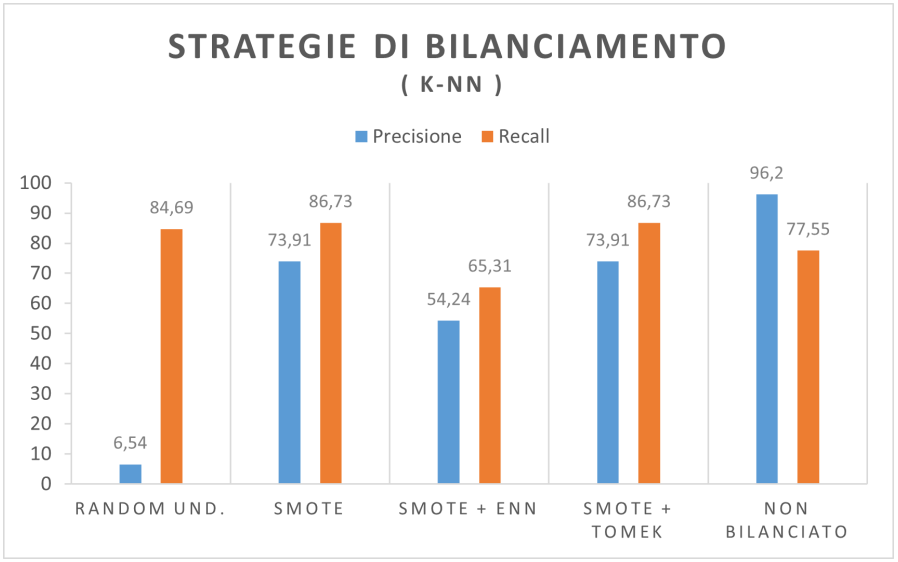

Figura 3.10: Metodi di bilanciamento con k-nn

In fig. 3.11 vengono riportati i risultati con diversi metodi di bilanciamento classificando i dati con XGBoost.

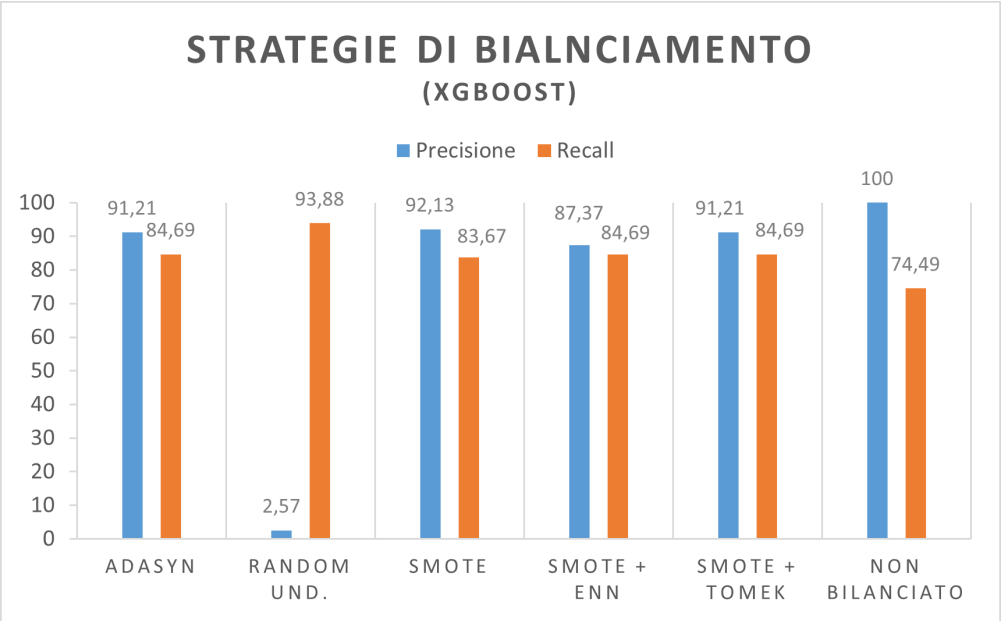

Figura 3.11: Metodi di bilanciamento con XGBoost

3.3.1 Valutazioni sui metodi di bilanciamento

Analizzando i risultati riportati per ogni algoritmo, si analizzino i dati e si valuti se il processo di bilanciamento è opportuno. Si valuti tra le metodologie testate quale sia la migliore.

Bilanciamento vs non bilanciato

La prima analisi che occorre fare è valutare se il processo di bilanciamento sia necessario o meno. Osservando i risultati riportati in sezione 3.3 si evince come diversi algoritmi riescano a raggiungere risultati interessanti anche su dati sbilanciati. Decision Tree per esempio è meno efficace su dati bilanciati e un discorso analogo lo si potrebbe fare per la regressione logistica. Svm invece beneficia del bilanciamento delle classi riuscendo a guadagnare fino al +54% sulla precisione. Random Forest, k-nn e XGBoost presentano un risultato molto interessante. Si osserva che il trend è lo stesso, il processo di bilanciamento alza la recall ma abbassa il valore della precisione. Non definiremo se le performance siano migliori su dati bilanciati o sbilanciati, lasceremo la decisione di tale affermazione al momento di applicazione del modello in un contesto reale. Dipenderà infatti come descritto in

sezione 2.4.4 dalle specifiche necessità richieste. Possiamo definire in questa analisi che con un algoritmo come Random Forest, nel caso si voglia un sistema con una precisione alta (97,37 %) e una recall un po' più bassa (75,51 %) sarà opportuno non bilanciare i dati. Viceversa se tra i requisiti è richiesto un sistema che riesca a individuare un maggior numero di frodi avendo una recall più alta (83,67 %) e mostrando una precisione più bassa (89,13 %) allora sarà opportuno bilanciare i dati.

Confronto algoritmi di bilanciamento

Una delle tecniche più veloci che si può utilizzare per bilanciare i dati è effettuare un random undersampling. Analizzando i risultati si evince come tale meccanismo sia quello meno efficace. Il motivo potrebbe essere dato dal fatto che per bilanciare i dati vengono rimosse in maniera casuale un numero di istanze dalla classe maggioritaria fintanto che le distribuzioni tra le due classi non sono le stesse. Avendo poche frodi, la quasi totalità dei dati viene scartata nel processo di bilanciamento. Questo porta a una fase di addestramento eseguita su troppi pochi esempi. Osservando i risultati ottenuti con Random Forest, SVM e XGBoost si può affermare che non vi è una grossa differenza tra i diversi meccanismi di bilanciamento. Nel caso di SVM in base all'esigenza di voler avere una recall un po' più alta converrebbe usare Smote+Tomek, viceversa Adasyn per migliorare la precisione. Per Random Forest e XGBoost invece la scelta tra Adasyn e Smote sembra essere indifferente.

3.4 Classificazione

Al fine di identificare quale classificatore riuscisse a mostrare i risultati migliori, sono stati testati diversi algoritmi di classificazione noti in letteratura come descritto in sezione 1.3. Gli algoritmi selezionati sono:

- Regressione Logistica
- K-Nearest neighbour
- Svm
- Decision Tree
- Random Forest
- XGBoost

Le implementazioni degli algoritmi di classificazione sono quelle fornite dalla libreria *Scikit-learn* [5] [45]. Ad eccezione di XGBoost[6].

3.4.1 Analisi risultati

Per ogni algoritmo sono state testate diverse combinazioni tra le metodologie di normalizzazione e bilanciamento descritte. In fig. 3.12 vengono riportati i risultati più interessanti.

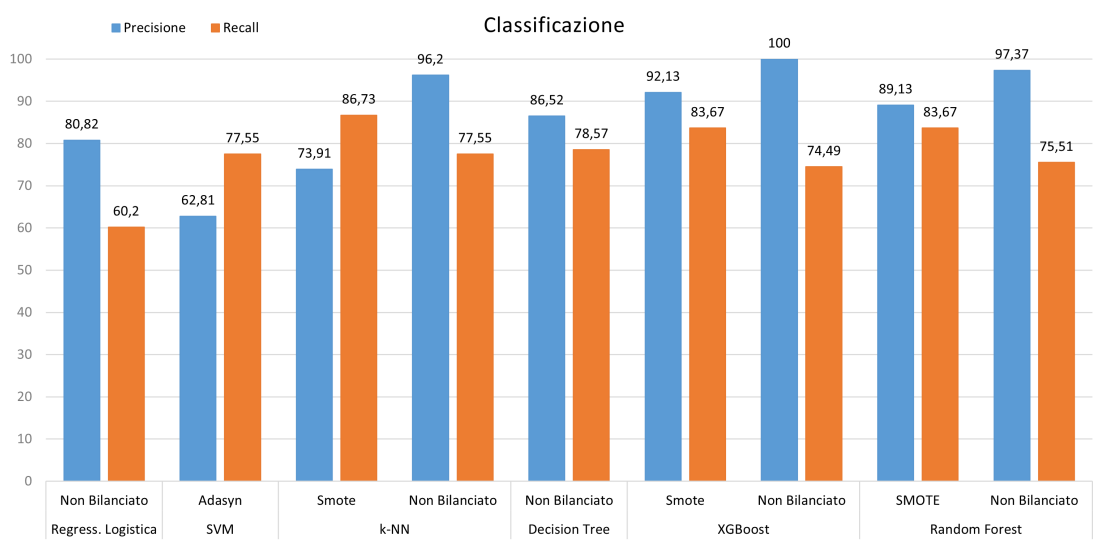

Figura 3.12: Risultati della classificazione dei principali algoritmi

Ai fini della valutazione dei modelli si considerino le metriche precisione e recall come analizzato in sezione 2.4.4. I modelli che hanno ottenuto i risultati migliori in termini di precisione e recall sono Random Forest e XGBoost. Anche k-NN riesce a ottenere uno dei risultati migliori. È necessario però fare una considerazione, tutte le metodologie riportate impiegano un tempo inferiore al secondo per classificare l'intero test set (circa 57 000 record), k-NN impiega 477 sec. Si consideri che il costo computazionale di una classificazione con questo algoritmo, è proporzionale al quadrato degli esempi nel training set. k-NN per questi motivi potrebbe avere problemi di performance nel caso vi siano molti dati. Bisogna inoltre considerare che il modello predittivo in un contesto reale farà parte di un sistema che dovrà effettuare le classificazioni in real time.

[5] `https://scikit-learn.org/stable/`
[6] `https://xgboost.readthedocs.io/en/latest/index.html`

Regressione logistica e SVM presentano due risultati opposti. La regressione ottiene risultati migliori sulla precisione, mentre SVM sulla recall. Decision tree riesce ad ottenere un risultato su dati sbilanciati che si avvicina a quello dei migliori classificatori. Osservando questi risultati si conclude affermando che l'approccio migliore per effettuare la classificazione di transazioni bancarie è utilizzare algoritmi quali Random Forest e XGBoost. Si riescono ad ottenere dei buoni risultati sui dati originali soprattutto nel caso che la priorità sia avere una precisione molto alta. Il processo di bilanciamento con SMOTE permette di alzare il livello della recall andando così ad identificare più frodi, alzando però, anche il valore dei falsi positivi.

Capitolo 4

Analisi basata sul comportamento degli utenti

In questo capitolo vengono riportati i risultati ottenuti da uno studio eseguito su un dataset reale. In particolare assume un ruolo chiave la modellazione del comportamento degli utenti. Vengono applicate le metodologie proposte nel capitolo 2.

4.1 Il Dataset

Il dataset in esame[1] è fornito da Vesta[2], un'azienda specializzata nel fornire soluzioni per i pagamenti online.

Struttura

Il dataset è costituito da un insieme di transazioni bancarie. I dati sono forniti tramite due dataset differenti: "train transaction" e "train identity". Il primo dataset è composto da 590 000 record e vengono fornite una serie di informazioni relative alle singole transazioni. Il secondo dataset invece contiene 144 000 dati e vuole essere un'aggiunta di informazioni per alcuni record del primo dataset. È possibile integrare le informazioni effettuando un join sull'identificativo di ogni transazione. Integrando i dati si otterranno record costituiti da 435 feature. Come per il primo dataset (sezione 3.1) e come trattato in sezione 2.1.3 spesso si deve lavorare su dataset sui quali sono state fatte delle operazioni di anonimizzazione prima che i dati siano resi pubblici. Il dataset in esame non ha tutte le feature in chiaro. Vi sono varie feature che sono oscurate e l'unica cosa che si può sapere è un'idea del

[1] `https://www.kaggle.com/c/ieee-fraud-detection/overview`
[2] `https://trustvesta.com/`

significato che in linea di massima potrebbe avere. Nonostante questo, una parte dei dati è in chiaro e disponibile. Le principali informazioni interpretabili sono: il device usato per la transazione, la somma in denaro coinvolta, un timestamp, una serie di informazioni sulla carta di credito e informazioni relative a cliente e venditore coinvolti. I dati presenti sono relativi a una finestra temporale di 6 mesi.

4.1.1 Preparazione del dataset

Ai fini di voler realizzare un modello che possa basare la classificazione analizzando il comportamento degli utenti, è opportuno che ogni transazione sia associata al cliente che ha eseguito l'operazione. Il problema è che in nessun record è presente un attributo che permetta di identificare questa informazione. Per poter creare un modello che si basi su dati storici e faccia un confronto con le transazioni passate, è opportuno che per ogni utente si abbiano un minimo di transazioni. La modellazione del comportamento deve poi essere eseguita su una serie di feature che devono essere interpretabili e che una volta aggregate possano dare un contributo informativo aggiuntivo. È importante analizzare i dati per comprendere la natura delle informazioni presenti ed eventualmente rielaborarli per far si che possano essere utilizzabili.

Aggiunta informazioni utenti

Al fine di voler analizzare il comportamento degli utenti, è opportuno sapere come minimo quali transazioni sono associate ad una stessa persona. Esaminando i dati si osserva che questo tipo di informazione non è presente, in fase di anonimizzazione è stato rimosso questo livello di dettaglio. Per tale motivo è opportuno ricreare l'informazione mancante. Basandosi sul lavoro di [4] e facendo delle valutazioni appropriate si può pensare di definire un livello di aggregazione dei dati su specifici attributi che permettano di definire in modo univoco un utente. Le feature selezionate a questo scopo sono: un codice della carta di credito, la regione di residenza, un codice di prodotto che è unico per tutte le transazioni di uno stesso utente e un'informazione relativa alla prima transazione eseguita.

Selezione degli utenti

Si vuole analizzare la storia di ogni utente e mettere in relazione ogni transazione con le operazioni passate. Per fare questo è opportuno definire un numero minimo di transazioni che ogni utente deve possedere. Parte di queste potranno servire per addestrare il modello e la restate parte per i test. Nell'analisi presentata si considerano solo gli utenti che hanno almeno 7 transazioni nello storico dati.

Selezione di feature

Considerato il tipo di analisi che deve essere eseguita è opportuno fare affidamento a feature che siano interpretabili. L'importante non è solo riuscire a ottenere buoni valori nelle misure della precisione e della recall, è altrettanto importante avere un modello che possa essere interpretabile. Per giustificare i criteri di classificazione, questi devono essere chiari per una persona che li voglia analizzare. La previsione di un sistema di questo tipo potrebbe essere vagliata da personale specializzato che per lavorare ha bisogno di sapere il perchè sono state predette determinate classi. Un modello di questo tipo si definisce *white box*. Per avere questo livello di comprensione è opportuno agire a due livelli:

- Algoritmo di classificazione
- Dati

Alcuni modelli come Decision Tree permettono di poter estrarre una serie di regole in modo che il criterio di classificazione appreso sia interpretabile anche da esseri umani. Non avrebbe molto senso però avere la definizione di una serie di regole ma non riuscire a interpretarle perchè i dati sono oscurati. Per esempio, avendo una regola del tipo:
"Se feature1 <X, feature2 <Y allora la transazione è una frode".
Non è interpretabile fintanto che non conosciamo cosa stiano a rappresentare "feature1" e "feature2". Questo è il motivo per il quale è importante lavorare su dati in chiaro. Ogni record del dataset a disposizione ha 345 feature, solamente una piccola parte di queste è in chiaro. Ai fini del tipo di analisi che si vuole fare e del risultato che si vuole raggiungere, si è deciso di selezionare solamente le feature che avessero un significato concreto. Ogni record del nuovo dataset risulta avere 22 feature. È importante considerare che avere attributi di cui se ne comprende il significato, potrebbe essere persino più rilevante dell' avere un algoritmo di classificazione *white box*. Si pensi di addestrare un modello *black box* su un insieme di feature, lo stesso modello poi venga addestrato sulle stesse feature + 1 feature o gruppo di nuove feature. La differenza tra i risultati dei due modelli sarebbe da attribuire al contributo informativo aggiunto dagli attributi introdotti nell'addestramento del secondo modello. Guidati da un esperto di dominio si possono creare nuovi gruppi di feature che si pensa possano essere rilevanti ai fini del riconoscimento di frodi, e di volta in volta osservare il contributo che viene dato sul risultato.

Gestione etichette Frodi

Osservando i dati a disposizione e basandosi sul meccanismo di etichettatura dei dati[2], si ritiene che le informazioni non siano appropriate per il tipo di analisi

da eseguire. Nei dati sembrerebbe che si cerchino di prevedere utenti fraudolenti piuttosto che transazioni fraudolente. Al fine di avere un dataset idoneo al tipo di analisi opportuna sono state eseguite una serie di trasformazioni. Si è deciso di rietichettare le classi relative alle frodi. L'assunzione di base che è stata fatta è che una frode è un'operazione eseguita a nome di un utente che in realtà viene eseguita da qualcun altro. Si vuole realizzare un modello che riesca a riconoscere questo tipo di situazione. Ad ogni utente sono state aggiunte delle transazioni eseguite da altri utenti e queste sono state marcate come frodi. In questo modo ogni utente ha un set di operazioni eseguite, quelle realmente appartenenti all'utente saranno considerate come non frodi, le operazioni che invece sono state aggiunte e che erano di altri utenti sono etichettate come frodi. In questo modo, decidendo il numero di frodi da aggiungere ad ogni utente si possono decidere le distribuzioni degli elementi per ogni classe. Inserendo circa una transazione fraudolenta ad ogni utente si ha che il 93% dei dati è relativo a operazioni legittime, mentre il 7% a operazioni fraudolente.

Preprocessing

Molti algoritmi non sono predisposti per lavorare con feature categoriche, motivo per il quale è necessario modellare questo tipo di dato tramite rappresentazione: *one-hot-encoded*. Prima dell'addestramento è opportuno rimuovere alcune feature per evitare che possano influenzare il risultato. Si rimuove l'identificativo della transazione e il timestamp. Informazioni utili ai fini della fase di feature engineering ma che introdurrebbero del rumore nel processo di addestramento. Può essere rimosso anche l'identificativo dell'utente, viene fatta una valutazione in sezione 4.1.2 sulle implicazioni che questo comporta.

4.1.2 Tipo di analisi

Il dataset deve essere diviso in set di training e set di test ai fini dell'addestramento di un modello predittivo. Considerata la tipologia di dati a disposizione, tale operazione può essere eseguita in diversi modi. Si ragioni sul fatto che in funzione di come si agisca, possono essere fatte asserzioni diverse e ciò determina il tipo di analisi effettuata. Vengono proposte due strategie:

- Analisi valida su utenti storici.
- Analisi valida anche per nuovi utenti.

Analisi valida su utenti storici

La divisione dei dati in set di train e di test viene eseguita in modo che una parte delle transazioni di ogni utente vadano nel set di train e la restante parte nel test

set. Il modello in fase di addestramento vede le informazioni relative a tutti gli utenti presenti, il codice identificativo di ogni utente potrebbe essere un attributo di riferimento per la classificazione. Questo tipo di divisione dei dati è utile nel caso si voglia un sistema in grado di predire le transazioni di un insieme di utenti noti.

Analisi valida anche per nuovi utenti

La divisione dei dati in set di train e di test viene eseguita in modo che tutte le transazioni di un sottoinsieme di utenti siano nel train e le restanti nel test. In questo modo l'addestramento viene eseguito sui dati relativi a un sottoinsieme di utenti. In questo tipo di analisi è opportuno rimuovere il codice identificativo di ogni utente. Il motivo è dato dal fatto che nel test set ci sono transazioni di utenti mai visti, quindi non avrebbe senso basare la classificazione sull'identificativo di un altro utente. Questo tipo di modello può essere usato quando si vuole un sistema che riesca a classificare transazioni di nuovi utenti. Se si hanno dubbi sulla correttezza di tale metodologia si faccia riferimento alla sezione 2.3.2 dove si discute la validità di tale meccanismo.

4.2 Feature Engineering

Come discusso in sezione 2.2 l'obiettivo principale della fase di feature engineering è quello di elaborare i dati a disposizione per estrarne di nuovi. Le informazioni che devono essere aggiunte sono relative alla relazione che sussiste tra una transazione e i dati storici. Un record trasformato avrà diversi tipi di feature, una parte di queste descrivono la transazione appena avvenuta, mentre le altre evidenziano una relazione tra gli attributi presenti e quelli delle transazioni passate dello stesso utente. L'idea è che aggiungere una serie di informazioni che mettano in relazione la transazione con le operazioni passate possa essere un contributo importante ai fini del riconoscimento di frodi. In ogni caso sarà poi l'algoritmo a valutare secondo i propri criteri di funzionamento (vedi sezione 1.3) su quali attributi basare la classificazione. I nuovi dati aggiunti potrebbero anche non essere rilevanti.

4.2.1 Nuovi gruppi di feature

Per far emergere le relazioni tra un'operazione e le transazioni passate dello stesso utente si sono utilizzati diversi criteri, le metodologie applicate sono quelle descritte in sezione 2.2. Sono stati identificati cinque gruppi di feature differenti:

- Frequenza attività

- Somma in denaro
- Carta di credito
- Acquirente e venditore
- Dispositivo

Lo scopo di ciascun gruppo di feature è quello di riuscire a catturare una certa situazione. I nuovi attributi forniscono un contributo informativo sufficiente a relazionare un'operazione con le transazioni passate dello stesso utente. Segue una descrizione specifica sul significato e motivo per cui ogni gruppo di feature ha ragione di esistere.

4.2.2 Frequenza Attività

Questo gruppo di feature aggiunge un insieme di informazioni relative alla frequenza di attività in relazione alla somma in denaro coinvolta. Vengono identificate tre fasce di spesa, la prima relativa alle spese minori, la seconda per quelle un po' più sostenute e la terza per le spese più importanti. Si conta per ogni periodo il numero di transazioni per ogni fascia. Sono riportate nello specifico le feature identificate:

- Conteggio del numero di transazioni nell'ultimo mese.
- Conteggio del numero di transazioni nell'ultima settimana.
- Conteggio del numero di transazioni nell'ultimo mese con una somma in denaro appartenente alla fascia 1.
- Conteggio del numero di transazioni nell'ultimo mese con una somma in denaro appartenente alla fascia 2.
- Conteggio del numero di transazioni nell'ultimo mese con una somma in denaro appartenente alla fascia 3.
- Conteggio del numero di transazioni nell'ultima settimana con una somma in denaro appartenente alla fascia 1.
- Conteggio del numero di transazioni nell'ultima settimana con una somma in denaro appartenente alla fascia 2.
- Conteggio del numero di transazioni nell'ultima settimana con una somma in denaro appartenente alla fascia 3.

Aggiungere questo tipo di informazioni potrebbe aiutare nell'identificare attività con una frequenza anomala. Potrebbe essere rilevante considerare anche l'entità della somma coinvolta.

4.2.3 Somma in denaro

Questo gruppo di feature fornisce un contributo mettendo in relazione la somma in denaro coinvolta nella transazione corrente con le somme coinvolte nelle transazioni passate dello stesso utente. "Mettere in relazione" significa valutare se la quantità in denaro in oggetto si discosta più di una determinata soglia percentuale rispetto la media del valore nei periodi precedenti. Sono riportate nello specifico le feature identificate:

- La somma in denaro coinvolta è maggiore di un +"X"% rispetto la media del mese scorso ?
- La somma in denaro coinvolta è maggiore di un +"X"% rispetto la media della settimana scorsa ?

Aggiungere questo tipo di informazione potrebbe aiutare nell'identificare spese anomale relative a somme in denaro elevate. Non si è considerato il caso in cui la somma coinvolta si discostasse considerevolmente al ribasso rispetto il valore medio. Si ritiene che sia abbastanza comune avere l'esigenza di dover effettuare piccoli acquisti, possono esserci tanti motivi per il quale un cliente possa effettuare delle piccole spese anche se non è solito fare questo tipo di attività. Inoltre, si vuole cercare di ridurre i falsi allarmi per non irritare i clienti. Si ritiene che un utente possa essere ben disposto a confermare di aver eseguito un'operazione con una somma in denaro elevata rispetto quanto lo potrebbe essere nel doverlo fare per piccole spese.

4.2.4 Dispositivo

Questo gruppo di feature fornisce un contributo basandosi sul dispositivo mobile che viene utilizzato per portare a termine la transazione. Sono riportate nello specifico le feature identificate:

- Il sistema operativo del device è quello solito ?
- Il browser è il solito ?
- Si è soliti usare questa tipologia (desktop o mobile) di device ?
- Il modello/s.o./marca è quello solito ?

- La risoluzione dello schermo è la solita ?

Aggiungere questo tipo di informazioni potrebbe aiutare nel capire se il dispositivo che si sta utilizzando è quello solito, potrebbe essere interessante evidenziare eventuali anomalie.

4.2.5 Carta di credito

Questo gruppo di feature da un contributo mettendo in relazione le informazioni relative alla carta di credito utilizzata per portare a termine la transazione. Si effettua un confronto per valutare se le informazioni coincidono o sono diverse rispetto le transazioni precedenti. Bisogna considerare che alcuni valori potrebbero non combaciare nel caso di clonazione di una carta, oppure trattandosi di transazioni online potrebbero esserci incongruenze relative alla creazione di carte virtuali.

- "card4" assume il solito valore ?
- "card6" assume il solito valore ?

Aggiungere questo tipo di informazione potrebbe essere utile nel caso di carte clonate o in presenza di anomalie nel processo di creazione di carte virtuali.

4.2.6 Acquirente e venditore

Questo gruppo di feature vuole analizzare il rapporto di continuità che potrebbe sussistere tra cliente e venditore. L'informazione a disposizione per evidenziare questo aspetto, si basa sul confronto dei domini email. Si confronta se sono abituali.

- dominio dell'utente è abituale ?
- dominio del venditore è abituale ?

Aggiungere questo tipo di informazione permette di evidenziare anomalie di rapporto tra cliente e venditore.

4.3 Risultati ottenuti

Per valutare il contributo delle tecniche di feature engineering introdotte, i dati sono stati classificati con diversi algoritmi. Vengono riportati i risultati ottenuti senza introdurre le trasformazioni del feature engineering. In seguito sono presentati i risultati con i dati trasformati, infine si mostra il contributo che ciascun gruppo di feature riesce a fornire. Sarà così possibile quantificare la validità degli

attributi introdotti. Tramite questa metodologia di lavoro, si riesce ad avere un feedback nel mentre che nuovi attributi vengono inseriti. Può essere un buon meccanismo di volta in volta raffinare e migliorare i gruppi di feature che mostrano risultati promettenti. Come detto in sezione 4.1.2 possono essere eseguite diverse tipologie di analisi. Nel prima casistica si ottiene un esito valido solo per utenti storici, nella seconda invece anche per nuovi utenti. Vengono riportati i risultati ottenuti in entrambe le analisi. Al fine di voler riportare risultati attendibili e che non fossero frutto di un qualche evento casuale, ogni valore riportato è frutto di una media fatta sui risultati di una sperimentazione ripetuta 10 volte.

4.3.1 Dati originali

Sono riportati i risultati ottenuti sui dati originali senza che venisse aggiunta nessuna feature. Il modello per inferire la classe di appartenenza si basa esclusivamente sulla transazione in oggetto. Le informazioni presenti sono del tipo:
"Browser e device utilizzato, somma in denaro coinvolta, informazioni sulla carta di credito, dominio email acquirente e venditore".
Sono riportati in fig. 4.1 i risultati ottenuti sull'analisi valida per gli utenti storici.

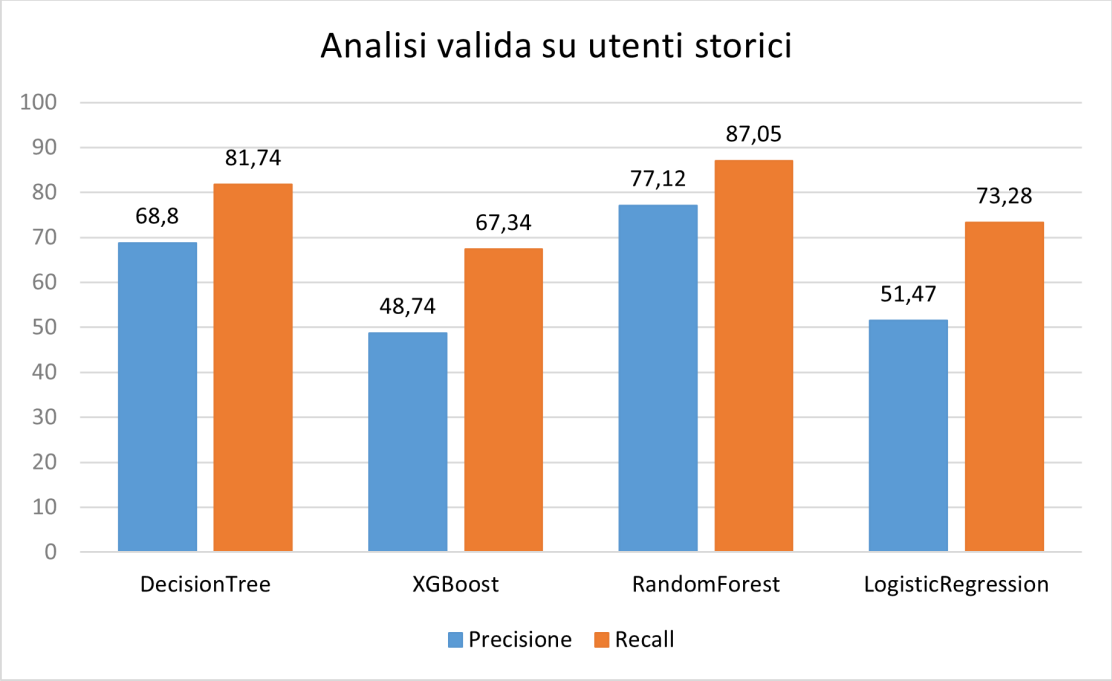

Figura 4.1: Risultato classificazione su dati originali. Analisi valida su utenti storici

Sono riportati in fig. 4.2 i risultati dell'analisi valida anche per nuovi utenti.

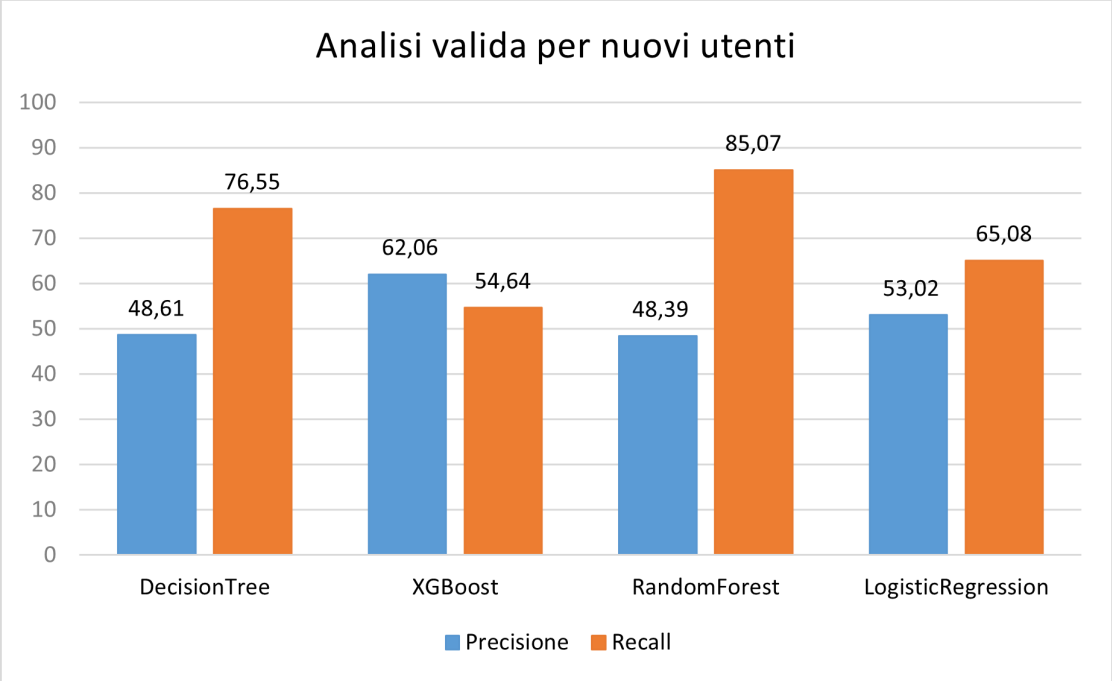

Figura 4.2: Risultato classificazione su dati originali. Analisi valida su nuovi utenti

In fig. 4.3 viene mostrato un confronto tra i risultati raggiunti per ogni algoritmo sulle due tipologie di analisi differenti (analisi valida su utenti storici vs analisi valida anche per nuovi utenti).

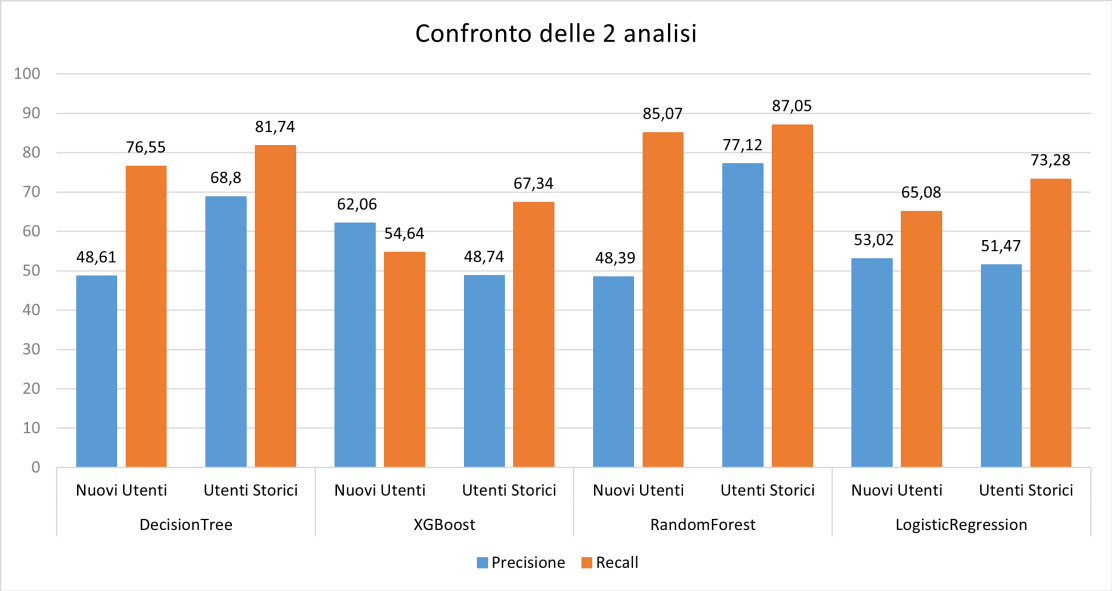

Figura 4.3: Confronto risultati classificazione analisi nuovi utenti e utenti storici

Interpretazione dei risultati

Nelle due analisi presentate si osserva che si ottengono risultati migliori analizzando esclusivamente le previsioni sui dati di utenti storici piuttosto che di nuovi utenti. Algoritmi come Decision tree, Random forest e la regressione logistica faticano a generalizzare il criterio di classificazione a nuovi utenti. In ogni caso valutando i risultati ottenuti sull'analisi limitata ai dati storici (dove i risultati sono migliori) si ritiene che si possa avere margine di miglioramento. Random forest è l'algoritmo che ha mostrato i risultati migliori con 87% di precisione e 77% di recall, a seguire: decision tree, la regressione logistica e XGBoost. Nel secondo tipo di analisi i risultati peggiorano soprattutto in termini di precisione, questo vuol dire che vengono generati molti falsi positivi. Si ritiene che il motivo per il quale i risultati ottenuti nella seconda analisi siano peggiori, è dato dal fatto che le informazioni a disposizione dai diversi algoritmi non siano sufficienti a delineare una superficie decisionale adeguata, soprattutto per estendere il criterio a nuovi utenti.

4.3.2 Dati trasformati

Vengono riportati i risultati ottenuti sui dati trasformati. Ogni transazione contiene una serie di informazioni che permettono di descriverne il contenuto, in aggiunta, sono presenti una serie di attributi che descrivono la relazione tra la transazione in oggetto e le transazioni passate dello stesso utente. Le informazioni aggiunte sono il risultato di una serie di operazioni di feature engineering, la descrizione dei nuovi attributi viene fornita nel dettaglio in sezione 4.2.1.
Sono riportati in fig. 4.4 i risultati dei principali algoritmi di classificazione ottenuti sull'analisi valida per gli utenti storici.

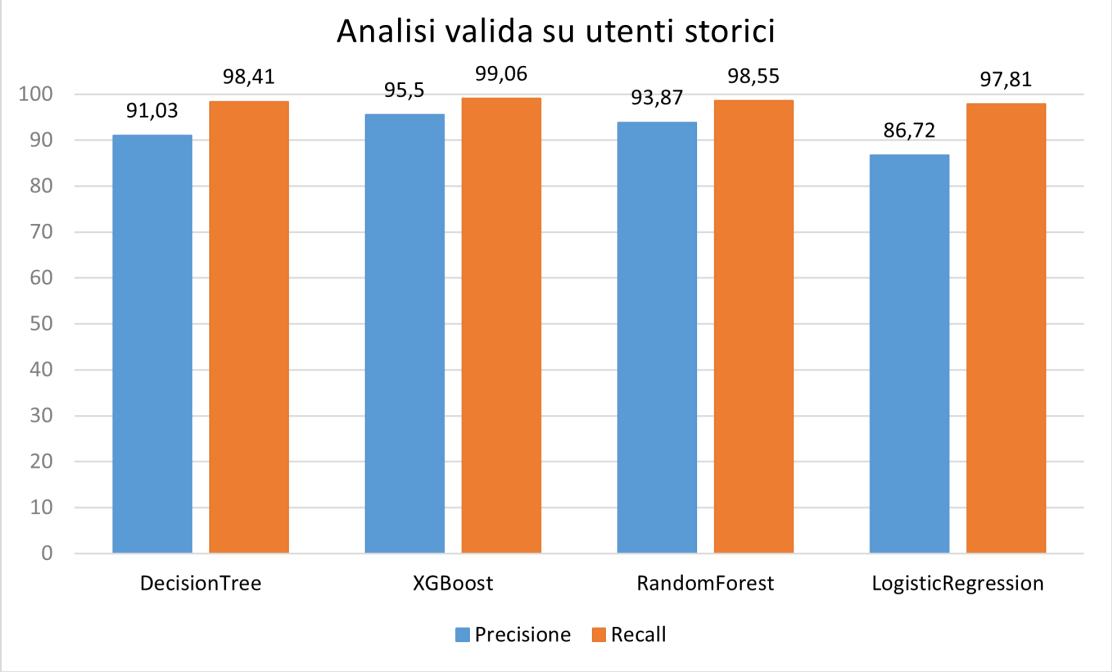

Figura 4.4: Risultato classificazione su dati trasformati. Analisi valida su utenti storici

Sono riportati in fig. 4.5 i risultati dei principali algoritmi di classificazione ottenuti sull'analisi valida anche per nuovi utenti.

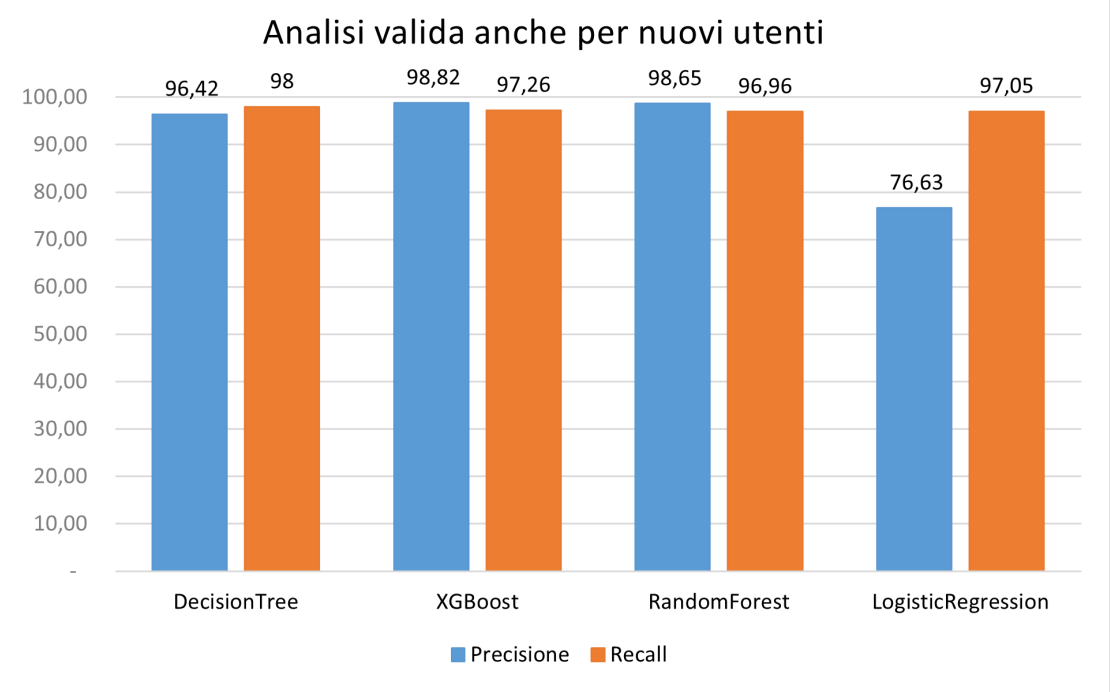

Figura 4.5: Risultato classificazione su dati trasformati. Analisi valida su nuovi utenti

Interpretazione dei risultati

Osservando i risultati raggiunti in entrambe le analisi, è evidente come il migliore approccio sia XGBoost, a seguire Random Forest, Decision Tree e Logistic Regression. Si riescono ad ottenere ottimi risultati sia in termini di precisione che di recall. Confrontando i risultati proposti, si nota che l'esito migliore lo si ottiene nell'analisi valida anche per nuovi utenti, il motivo potrebbe essere dato dal fatto che ai fini della classificazione non è importante quale utente ha commesso una transazione, ma esclusivamente il rappporto tra la transazione e la sua storia passata. I risultati mostrano come sia importante basare la classificazione su questa informazione piuttosto che sull'identificativo proprio dell'utente.

Considerazione sui dati e validità del risultato

I risultati raggiunti con i dati trasformati sembrano riuscire a fornire un'ottima tecnica per risolvere il problema del riconoscimento delle frodi. Osservando i risultati si può affermare che le operazioni di feature engineering presentate si possano

considerare una buona tecnica. È importante considerare che i dati sono relativi a transazioni reali, le etichette però sono state modificate per i motivi descritti in sezione 4.1.1. Avendo generato le etichette artificialmente si ritiene che possa esserci l'eventualità che in un contesto reale i dati possano essere diversi. Tuttavia le operazioni di manipolazione per la preparazione del dataset sono state eseguite nell'ottica di voler mantenere i dati il più fedele possibile a un contesto reale.

4.3.3 Confronto dei risultati

Sono stati mostrati i risultati ottenuti con i dati originali (sezione 4.3.1) e quelli ottenuti con i dati trasformati (sezione 4.3.2). Si vuole a questo punto mostrare un confronto tra le due metodologie e valutare il contributo che le operazioni di feature engineering hanno dato. Il confronto viene fatto per entrambe le analisi (analisi valida per utenti storici e analisi valida anche per nuovi utenti).
In fig. 4.6 sono riportati i risultati ottenuti nell'analisi valida per utenti storici addestrando il modello sui dati originali e sui dati con gli attributi del feature engineering.

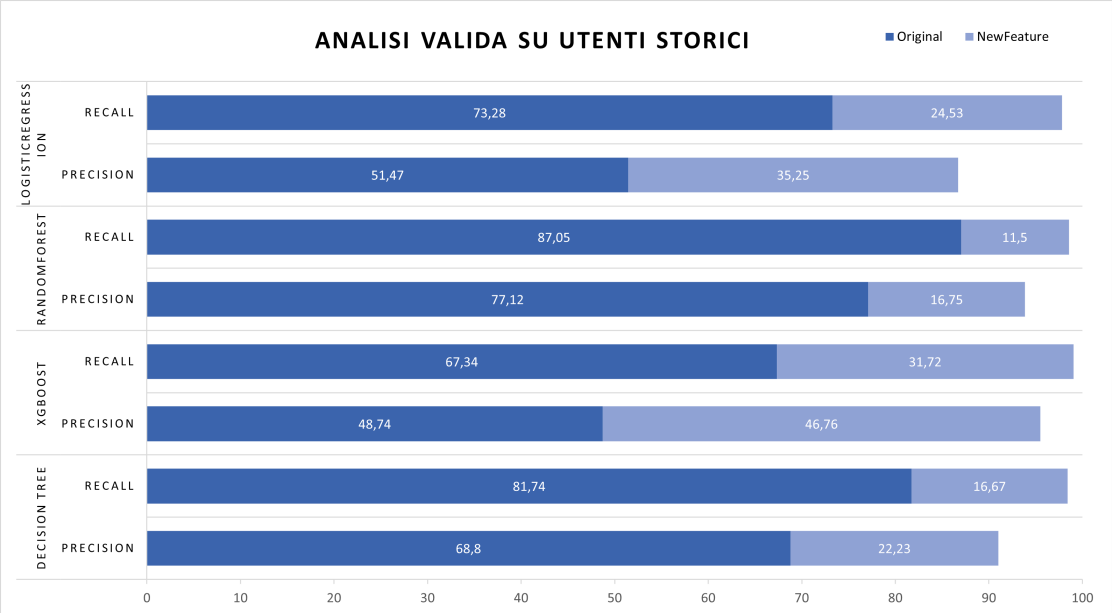

Figura 4.6: Risultati analisi valida per gli utenti storici. Dati originali e aggiunta delle nuove feature

In fig. 4.7 sono riportati i risultati ottenuti nell'analisi valida anche per i nuovi utenti addestrando il modello sui dati originali e sui dati con gli attributi del feature engineering.

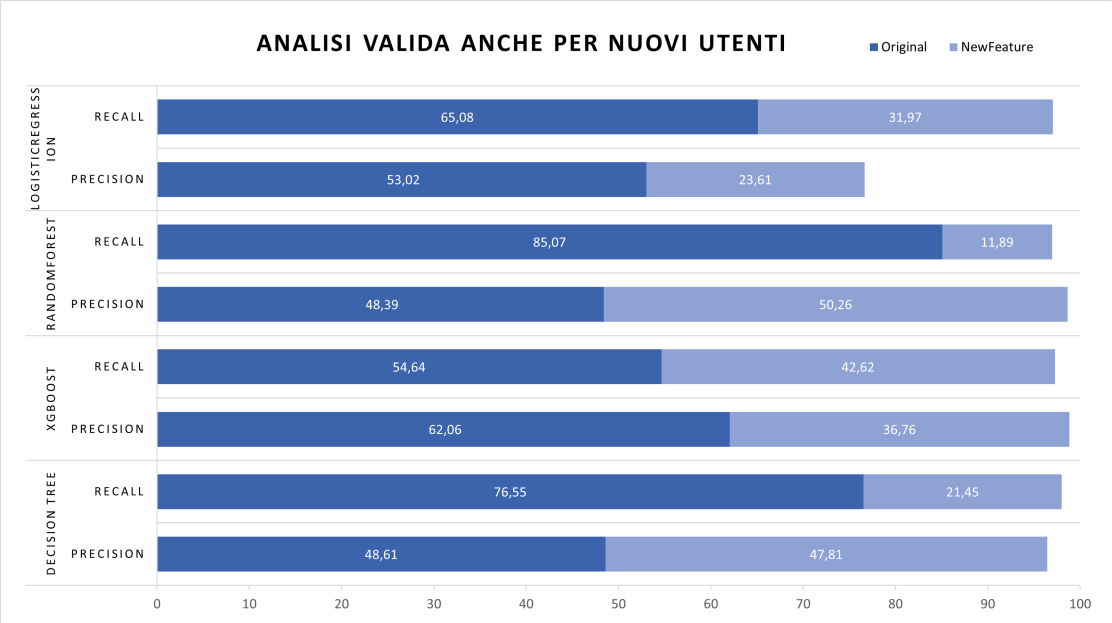

Figura 4.7: Risultati analisi valida anche per nuovi utenti. Dati originali e aggiunta delle nuove feature

Confrontando i risultati ottenuti emerge il fatto che l'aggiunta delle nuove feature si rivela una strategia vincente per migliorare l'efficacia degli algoritmi testati. Mettere in relazione il comportamento tenuto in una transazione con il comportamento tenuto nelle transazioni passate è una valida soluzione al fine di identificare anomalie nei dati che potrebbero essere frutto di operazioni fraudolente. Aggiungere questo tipo di informazioni modifica notevolmente i criteri sui quali gli algoritmi di classificazione basano la classificazione. Quando il modello si basa esclusivamente sui dati originali, non ha sufficiente contenuto informativo per definire una superficie decisionale appropriata. Osservando il contributo fornito dai nuovi attributi, si riescono ad ottenere buoni risultati anche nell'analisi valida per nuovi utenti.

4.3.4 Contributo feature engineering

In sezione 4.3.3 si è dimostrato come l'aggiunta di nuovi attributi ottenuti tramite il processo di feature engineering sia una strategia efficace nel riconoscimento delle frodi, si riescono ad ottenere buoni risultati sia sulla precisione che sulla

recall. Il contributo positivo viene dato da cinque gruppi di feature descritti in sezione 4.2.1. Ai fini di un'analisi dei risultati più specifica si valuti quanto ogni gruppo di feature sia efficace. In modo che sia possibile avere degli spunti sulle feature da raffinare per continuare a migliorare i risultati. Inoltre si è discusso in sezione 2.2 dell'importanza di avere dei risultati interpretabili. Ottenere un riscontro sul contributo di ogni gruppo di feature potrebbe essere un buon modo per comprendere la validità dei meccanismi proposti. Tale metodologia potrebbe essere un importante strumento per esperti di dominio. Si pensi infatti che per migliorare il sistema, debbano essere create nuove feature, una strategia potrebbe essere quella di raffinare le feature che han dimostrato fornire un contributo interessante.

Frequenza attività

È possibile analizzare il contributo fornito dalle feature relative alle informazioni sulla frequenza di attività in relazione alla somma in denaro coinvolta. In sezione 4.2.2 viene data una descrizione più approfondita delle feature aggiunte. I risultati che mostrano il confronto tra i dati originali e i dati con l'aggiunta di questo gruppo di feature sono mostrati in fig. 4.8.

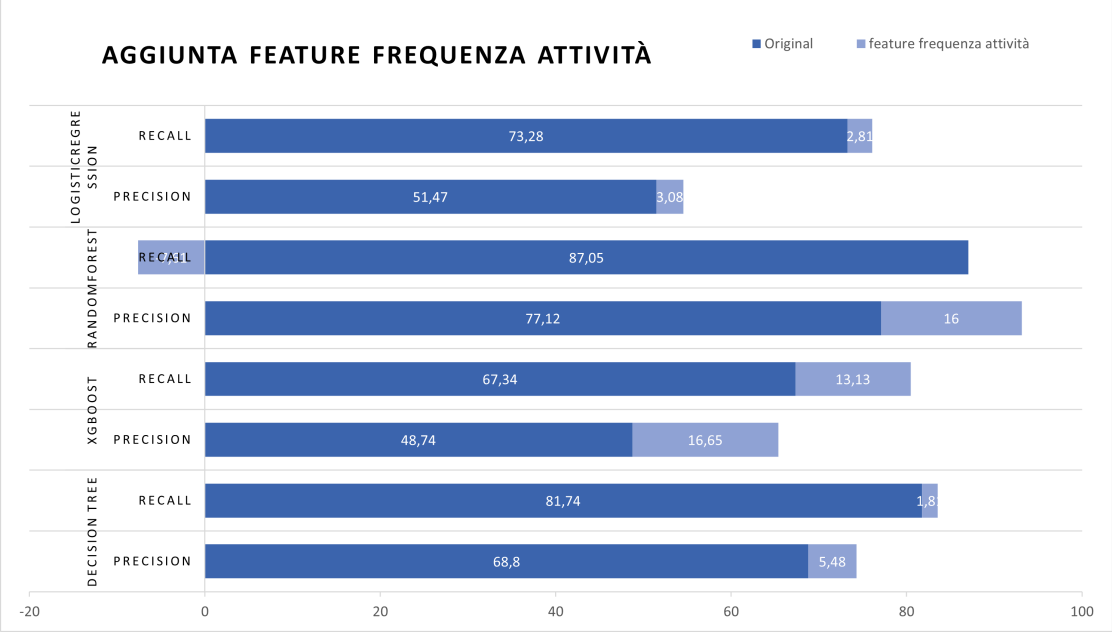

Figura 4.8: Risultati con i dati originali e aggiunta delle feature relative alla frequenza delle attività

Somma in denaro

È possibile analizzare il contributo fornito dalle feature relative alle informazioni sulle somme in denaro coinvolte nelle transazioni. Vengono evidenziati gli scostamenti del pagamento rispetto le spese abituali. Per una descrizione più approfondita delle feature aggiunte, fare riferimento alla sezione 4.2.3. I risultati che mostrano il confronto tra i dati originali e i dati con l'aggiunta di questo gruppo di feature sono mostrati in fig. 4.9.

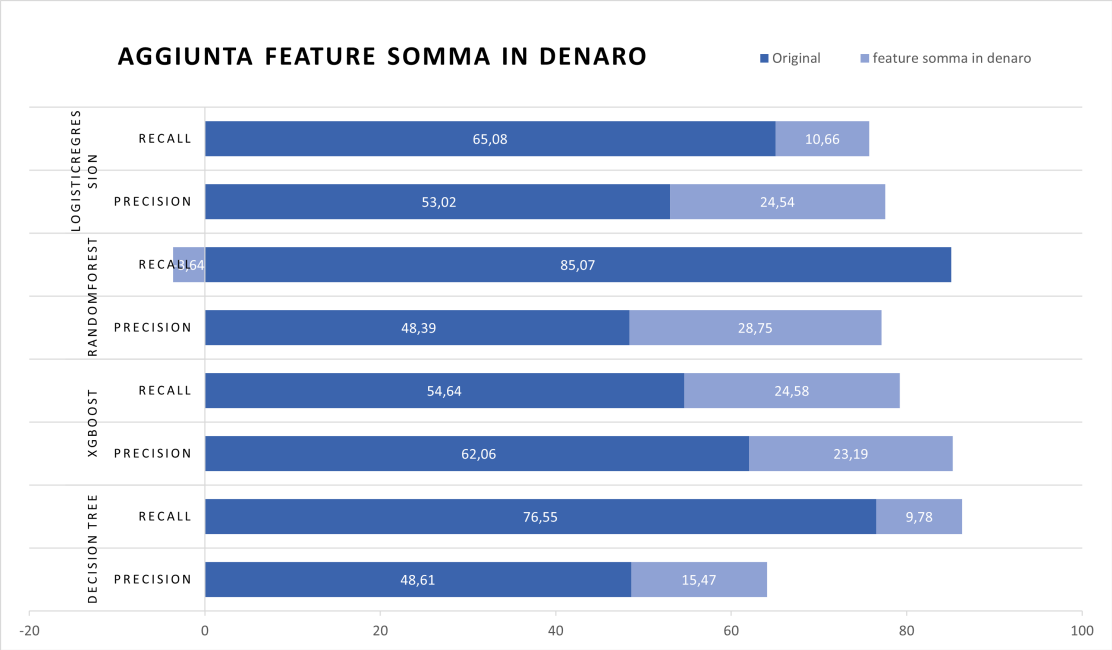

Figura 4.9: Risultati con i dati originali e aggiunta delle feature relative all'analisi delle somme in denaro coinvolte

Carta di credito

È possibile analizzare il contributo fornito dalle feature relative alle informazioni sulle carte di credito per verificare che i dati siano sempre gli stessi. Per una descrizione più approfondita delle feature aggiunte, fare riferimento a sezione 4.2.5. I risultati che mostrano il confronto tra i dati originali e i dati con l'aggiunta di questo gruppo di feature sono mostrati in fig. 4.10.

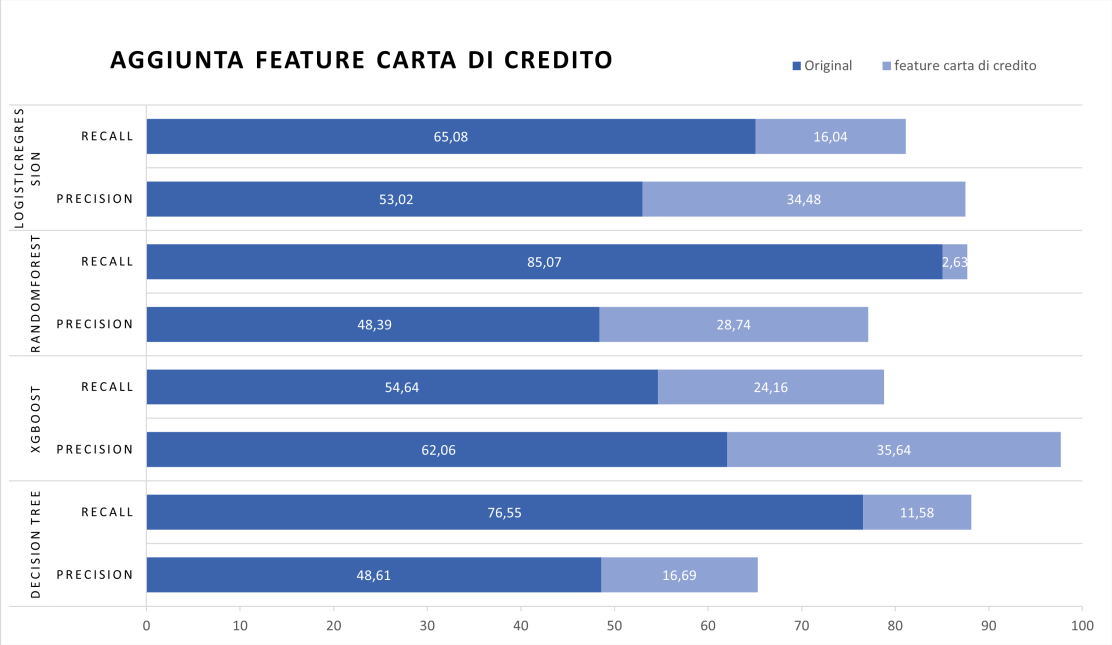

Figura 4.10: Risultati con i dati originali e aggiunta delle feature relative all'analisi della carta di credito

Acquirente e venditore

È possibile analizzare il contributo fornito dalle feature relative alle informazioni sul rapporto di continuità che potrebbe sussistere tra cliente e venditore. Per una descrizione più approfondita delle feature aggiunte, fare riferimento a sezione 4.2.6. I risultati che mostrano il confronto tra i dati originali e i dati con l'aggiunta di questo gruppo di feature sono mostrati in fig. 4.11.

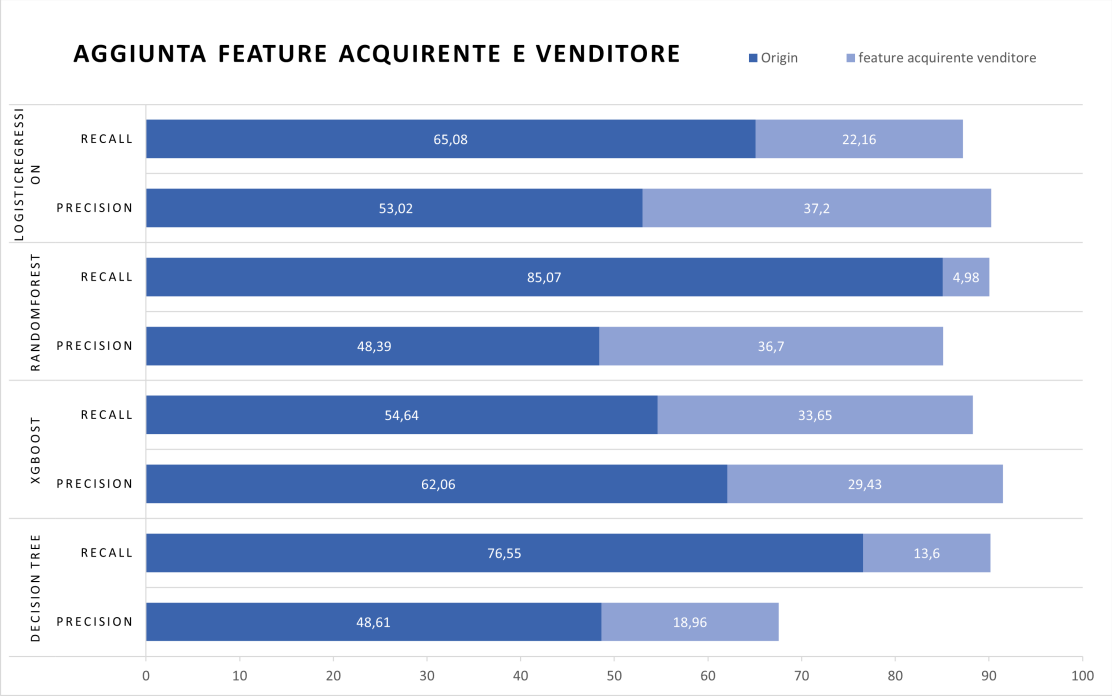

Figura 4.11: Risultati con i dati originali e aggiunta delle feature relative all'acquirente e venditore

Dispositivo

È possibile analizzare il contributo fornito dalle feature relative al dispositivo mobile usato. Per una descrizione più approfondita delle feature aggiunte, fare riferimento a sezione 4.2.4. I risultati che mostrano il confronto tra i dati originali e i dati con l'aggiunta di questo gruppo di feature sono mostrati in fig. 4.12.

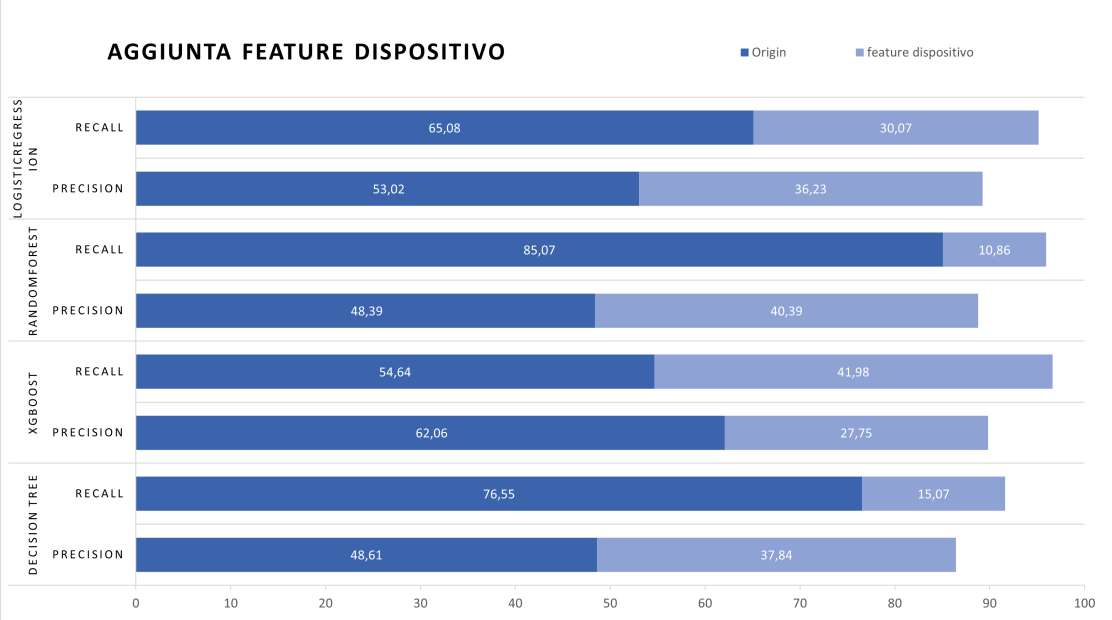

Figura 4.12: Risultati con i dati originali e aggiunta delle feature relative al dispositivo usato

Capitolo 5

Prototipizzazione

Questo capitolo ha lo scopo di voler mostrare una possibile realizzazione delle metodologie presentate tramite un sistema software applicato in un contesto reale. L'input del sistema è composto da un flusso di transazioni bancarie eseguite da diversi utenti. Deve essere fornita una previsione sul fatto che la transazione in esame si tratti di un'operazione legittima o fraudolenta. In questo capitolo viene presentata una possibile infrastruttura software basata su tecnologie big data che permetta di manipolare e gestire in tempo reale flussi continui di dati. Per la realizzazione del sistema sono state utilizzate diverse tecnologie quali principalmente: Spark, Kafka e Cassandra.

5.1 Tecnologie

Vengono descritte le possibili tecnologie che potrebbero essere usate per la realizzazione del sistema. Nello specifico ci si focalizza su Kafka, Cassandra e Spark facendo una panoramica anche delle principali librerie.

5.1.1 Spark

Apache Spark[1] è un motore di elaborazione big data open-source per eseguire computazioni su larga scala. Viene fornita un'interfaccia di programmazione che permette la realizzazione di applicativi studiati per essere eseguiti su un cluster di computer. I calcoli e i processi vengono eseguiti, implicitamente, in maniera parallela e distribuita, viene inoltre garantita tolleranza ai guasti (entro certi limiti) . Spark può essere eseguito all'interno di diversi ambienti d'esecuzione come Hadoop YARN, Mesos, EC2 o Kubernetes e può accedere a diverse sorgenti di dati come HDFS, Apache Cassandra, Apache Hbase etc. Può essere considerato

[1] `https://spark.apache.org`

una valida alternativa al paradigma: "map-reduce" proposto in Hadoop. Grazie alle computazioni eseguite in memoria viene limitato l'accesso al filesystem, può essere fino a 100x volte più veloce di "map-reduce". Prima dell'esecuzione vengono eseguite delle ottimizzazioni sui piani di accesso fisici e sull'esecuzione delle query. È possibile scrivere i programmi in diversi linguaggi di programmazione come: Java, Scala, Python e R. Spark può essere considerato un buon candidato per l'analisi di grandi quantità di dati anche grazie alle numerose librerie ed estensioni disponibili. Appartengono all' ecosistema di Spark: Spark SQL, Spark Streaming, MLib e Graphx. Ognuna di queste librerie può essere usata per svolgere compiti particolari e possono essere integrate insieme all'interno di uno stesso progetto.

Spark SQL

Spark Sql[2] è un modulo di Apache Spark che permette di lavorare con tipi di dato strutturato. L'interrogazione può essere portata a termine tramite il linguaggio SQL oppure con un insieme di API che manipolano un tipo di dato *DataFrame*. Le sorgenti dati possono essere molteplici: Hive, Avro, Parquet, ORC, JSON e JDBC. Grazie alle informazioni relative alla struttura interna dei dati, rispetto le classiche *RDD* API di Spark, Spark SQL riesce a fare delle ottimizzazioni sui piani di esecuzione delle query. È presente un ottimizzatore cost-based. La scalabilità viene fornita grazie al motore d'esecuzione di Spark.

Dataset e DataFrame Un *Dataset* è una collezione di dati distribuita. È un'interfaccia che fornisce i benefici degli *RDD*, ovvero la possibilità di sfruttare le potenzialità del lambda calcolo, unite ai benefici del motore d'esecuzione ottimizzato di Spark sql. Un *Dataset* può essere costruito a partire da oggetti nella JVM sui quali sarà poi possibile effettuare delle manipolazioni tramite trasformazioni funzionali (map, filter etc.). Un *DataFrame* è un *Dataset* organizzato secondo colonne. È equivalente a una tabella nel modello relazionale. Può essere costruito a partire da dati strutturati come tabelle Hive, database o RDD.

Spark Streaming

Spark Streaming[3] è un'estensione di Apache Spark che fornisce delle API per manipolare stream di dati real time. I dati possono entrare nel sistema da diversi sorgenti come: Apache Kafka, Kinesis o socket TCP. La manipolazione dei dati avviene esprimendo algoritmi complessi tramite funzioni ad alto livello come : map, reduce, join e window. I dati processati possono infine essere memorizzati su file system o database.

[2] `https://spark.apache.org/docs/latest/sql-programming-guide.html`
[3] `https://spark.apache.org/docs/latest/streaming-programming-guide.html`

Figura 5.1: Spark streaming

Spark Streaming ha come input uno stream di dati e partiziona i dati in batch. Ogni batch viene processato da Spark e viene generato uno stream finale di risultati, frutto delle manipolazioni di ogni batch.

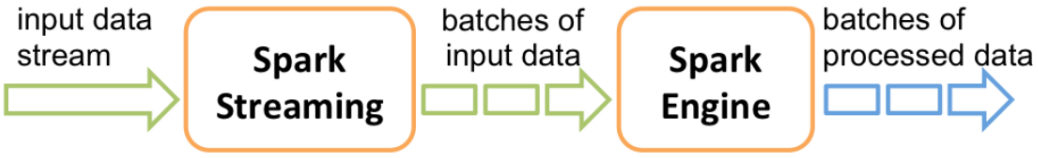

Figura 5.2: Spark streaming workflow

Spark Streaming rappresenta uno streaming di dati tramite l'astrazione: *DStream* sigla di: "discretized stream". Internamente un *Dstream* è rappresentato come una sequenza continua di *RDD*. Ogni *RDD* che compone un *Dstream* contiene i dati relativi a un certo intervallo temporale.

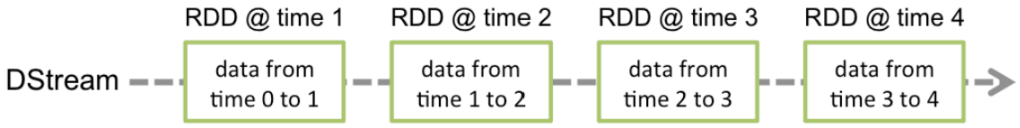

Figura 5.3: Dstream come sequenza di RDD

Ogni operazione eseguita su un *Dstream* si riflette come un'operazione sui relativi *RDD*. Viene riportato un esempio in fig. 5.4 che mostra come avviene il

processo di trasformazione da uno stream di righe di testo, a uno stream di parole tramite una trasformazione *flatMap*.

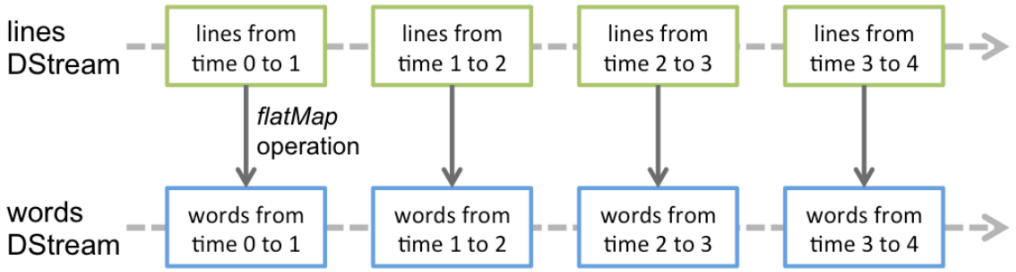

Figura 5.4: Trasformazione da testo a parole tramite flatmap

Structured Streaming

Structured streaming[4] è costruito sopra il motore d'esecuzione Spark SQL. Permette di esprimere computazioni su streaming di dati come se fossero operazioni batch. Il motore di Spark SQL, si occupa di eseguire ripetitivamente le query e aggiornare il risultato nel mentre che nuovi dati continuano ad arrivare. Lo strumento, cerca di fornire al programmatore un meccanismo tramite il quale astrarre la natura di un flusso di dati e fare sì che si possano esprimere delle computazioni come manipolazioni di *DataFrame* e *Dataset*. Internamente viene utilizzato un motore d'esecuzione che si basa su micro-batch di dati. Gli stream di dati vengono processati come una serie di piccoli *job batch* in modo tale da avere una bassa latenza (<100 ms) e far si che sia garantita un semantica "exactly-once".

[4] `https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html`

Tabelle "illimitate" L'idea chiave in Spark Structured Streaming, è quella di avere delle tabelle alle quali vengono aggiunti i dati man mano che sono disponibili. La computazione può essere espressa come una query batch su una tabella statica. Spark si occupa di eseguire la query in maniera continuativa.

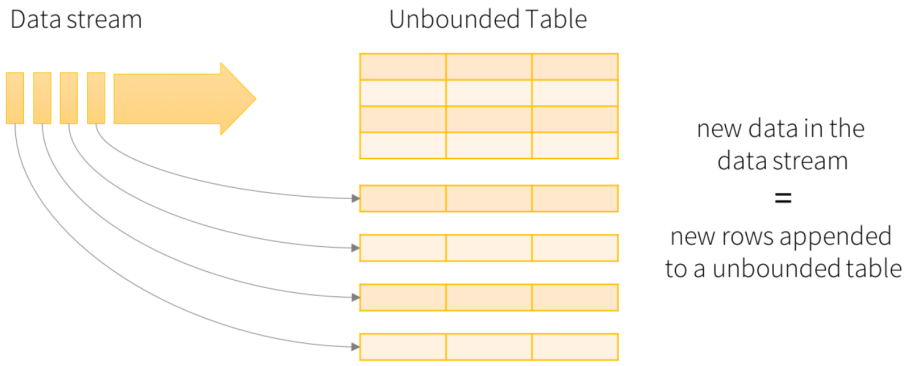

Figura 5.5: Rappresentazione dei dati in structured streaming

Una query sulla tabella in input genera una "tabella risultato". Ogni Δ "t" di tempo, nuove righe vengono aggiunte alla tabella in input e questo permette che ci possano essere degli aggiornamenti sulla "tabella risultato". Quando vi è un aggiornamento nel risultato finale, questo cambiamento deve essere propagato su una eventuale sorgente dati esterna.

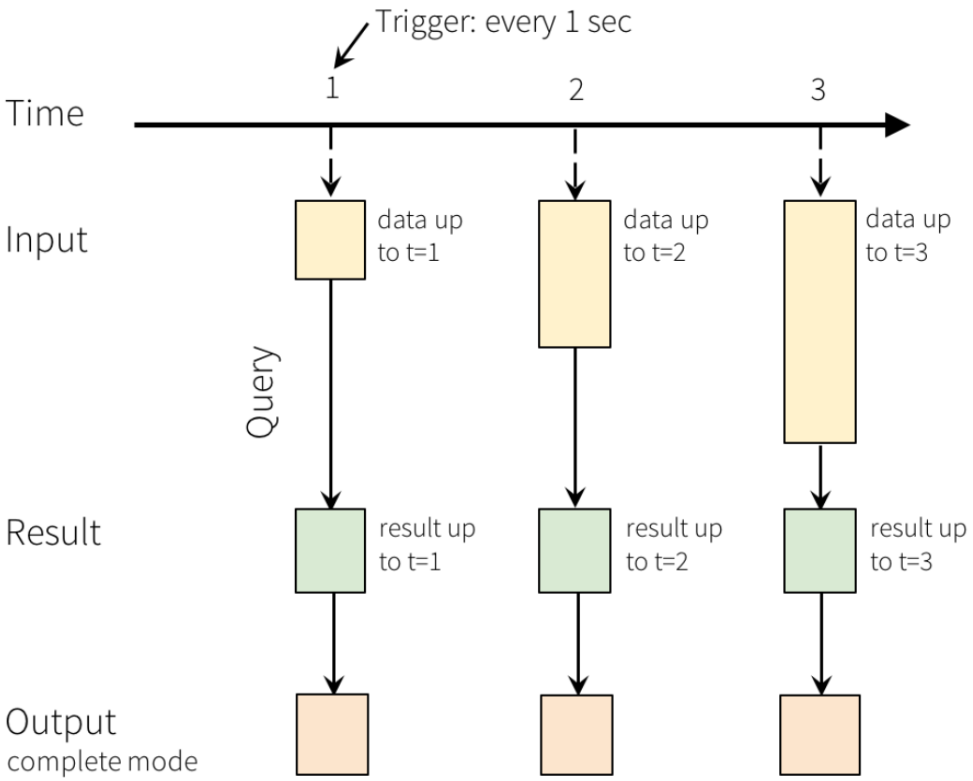

Figura 5.6: Structured streaming workflow

Considerata la natura "illimitata" della tabella, questa non viene interamente materializzata. Vengono letti gli ultimi valori disponibili, dopo essere stati processati, viene aggiornato il risultato, successivamente i dati della tabella in input vengono cancellati. Sono mantenuti i dati che permettono di rappresentare uno stato intermedio in modo che si possa partire da queste informazioni per aggiornare il risultato con i nuovi dati.

Spark MLlib

Spark Machine Learning (ML) library[5], è una libreria il cui scopo è quello di voler portare il machine learning in maniera semplice e scalabile in un'infrastruttura cluster. Vengono forniti una serie di servizi, vi sono diverse tipologie di algoritmi per risolvere problemi di: classificazione, regressione, clustering. Meccanismi per manipolare i dati come: estrazione di feature, trasformazione di dati, riduzione della dimensionalità.

[5] `https://spark.apache.org/docs/latest/ml-guide.html`

Modello dati I modelli di riferimento per la rappresentazione dei dati in Spark sono i *DataFrame* e gli *RDD*. Le API e le funzionalità di libreria di SparkML sono disponibili per entrambi i modelli. Il supporto alle API basate su *RDD* è in"maintenance mode", continueranno a essere supportate, però non verranno aggiunte nuove estensioni e feature. SparkML consiglia di fare riferimento e/o preferibilmente migrare alle DataFrame-API. I motivi sono da ricercare nel fatto che viene fornita un'interfaccia più user-friendly, si ha un modello di API uniforme rispetto differenti algoritmi di Machine Learning e differenti linguaggi.

5.1.2 Kafka

Apache Kafka[6] è una piattaforma streaming ad eventi. Lo scopo è quello di catturare i dati in ingresso in maniera real-time. Le sorgenti dei dati possono essere diverse: database, sensori, dispositivi mobili, servizi cloud o applicazioni software che pubblicano stream di dati. Gli eventi devono essere intercettati e memorizzati per far si che possano essere processate delle risposte in tempo reale. Un'architettura a stream di eventi è un flusso continuo di dati in ingresso nel sistema. Kafka può essere usato per diverse funzionalità in ambiente distribuito. Le principali sono:

- Scrivere e leggere stream di eventi continui. Gli stream possono essere interni ed esterni al sistema.
- Memorizzare stream di eventi in modo tale che possano essere reperibili in qualsiasi momento.
- Processare stream di eventi.

Struttura di Kafka

Kafka è un sistema distribuito che comprende server e client. Il server è da intendersi come un cluster di uno o più server. Alcuni di questi, formano dei "Brokers",corrispondono al livello di memorizzazione dei dati. Altri server eseguono "Kafka Connect", uno strumento per importare ed esportare stream di eventi. I client permettono di implementare applicazioni distribuite che processano stream di eventi in parallelo. Grazie a "Kafka Streams" viene fornito un livello di API per trasformare gli stream di eventi. In alternativa possono essere utilizzate altre soluzioni tecnologiche come Spark, nelle versioni Spark Streaming o Structured Streaming.

[6] `https://kafka.apache.org`

Concetti Principali

In kafka un evento è rappresentato tramite una chiave, un valore, un timestamp e una serie di metadati opzionali. Un'entità Client che scrive dati in Kafka è un "Producer", a differenza dei "Consumer" che leggono gli eventi pubblicati. Un dei principali elementi di design di Kafka è il fatto di avere Producer e Consumer completamente disaccoppiati. Gli eventi sono memorizzati in un'astrazione chiamata "Topic". I Topic in Kafka possono avere più producer e più consumer. Gli eventi non per forza devono essere cancellati dopo essere stati letti, si può definire per quanto tempo Kafka deve mantenere memorizzati questi dati. Le performance sono costanti rispetto la quantità di dati memorizzati, il servizio non degrada se si mantengono i dati più vecchi. I topic sono partizionati, quindi ogni topic ha una serie di "bucket" localizzati sui differenti kafka broker nella rete. La distribuzione spaziale dei dati è importante ai fini della scalabilità perchè permette a un'applicazione client di leggere e scrivere dati da diversi broker allo stesso tempo. Quando un evento deve essere pubblicato, viene aggiunto in una partizione del topic. Eventi con uno stesso valore di chiave sono scritti nella stessa partizione. È garantito che ogni consumer legga i dati di una partizione mantenendo lo stesso ordine con il quale sono stati scritti.

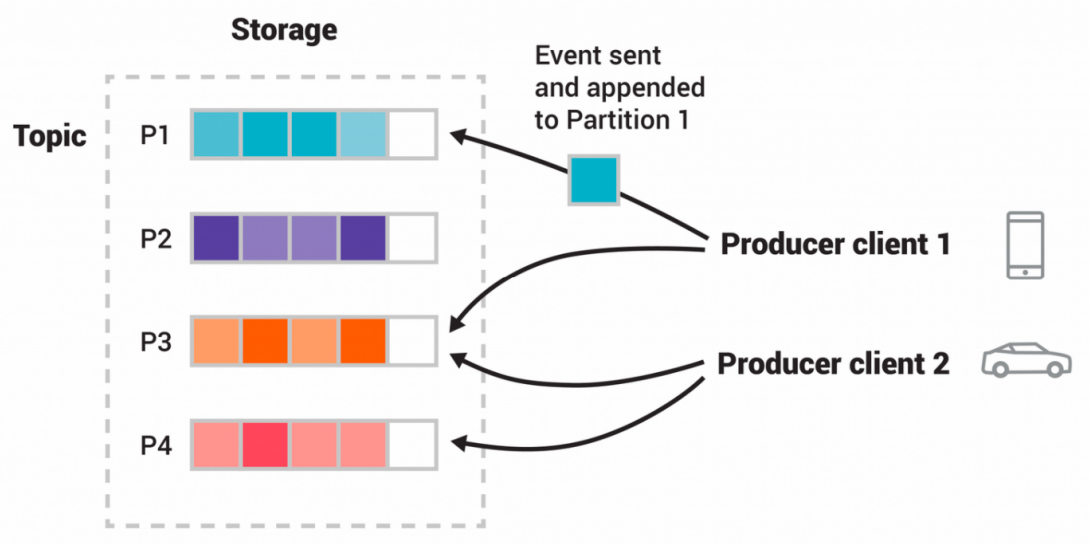

Figura 5.7: Rappresentazione di un Topic

5.1.3 Cassandra

Apache Cassandra[7] è un Database NoSQL di tipologia *wide-column*. Progettato per gestire grandi quantità di dati, fornisce un servizio con *high availability* e decentralizzazione evitando di avere *single point of failure*. Tra le principali feature di Cassandra vi è il fatto che è un database distribuito. I nodi del cluster sono replicati e i dati sono distribuiti. Ogni nodo contiene dati differenti, essendo una soluzione decentralizzata non vi è un *master* e ogni richiesta può essere gestita da ogni nodo. La strategia e il numero di repliche possono essere opportunamente configurate; questo permette una certo grado di confidenza sulla tolleranza ai guasti. I nodi che non sono più raggiungibili possono essere rimpiazzati. Come per altri database NoSQL, viene data priorità al partizionamento della rete e all' *availability* piuttosto che alla consistenza. Le letture e scritture possono essere configurate in modo tale da garantire una determinata semantica di consistenza.

Modello dei dati I dati sono memorizzati in tabelle composte da righe e colonne. Non è supportata la visione relazionale. Il modello dei dati è: "orientato alle query", sono ottimizzate specifiche interrogazioni. Le query sono progettate per accedere ad una sola tabella che deve contenere tutti gli elementi di un'interrogazione, in questo modo si può avere un accesso ai dati in lettura molto veloce. Considerando che ad ogni query corrisponde una tabella, i dati vengono duplicati tra tabelle in un procedimento che prende il nome di: denormalizzazione. La scelta di una chiave primaria e una chiave di partizionamento sono importanti per la distribuzione dei dati all'interno di un cluster. I dati sono distribuiti in un cluster di nodi, una chiave di partizionamento è usata per dividere i dati tra i nodi della rete. Una chiave di partizionamento viene generata a partire dal primo campo della chiave primaria. Ogni riga è identificata da una chiave e ha più colonne, ognuna delle quali ha un nome, un valore e un timestamp. Differentemente dai modelli RDBMS, righe diverse all'interno della stessa tabella non devono condividere lo stesso set di colonne, una colonna può essere aggiunta a una o più righe in qualsiasi momento.

[7] `https://cassandra.apache.org`

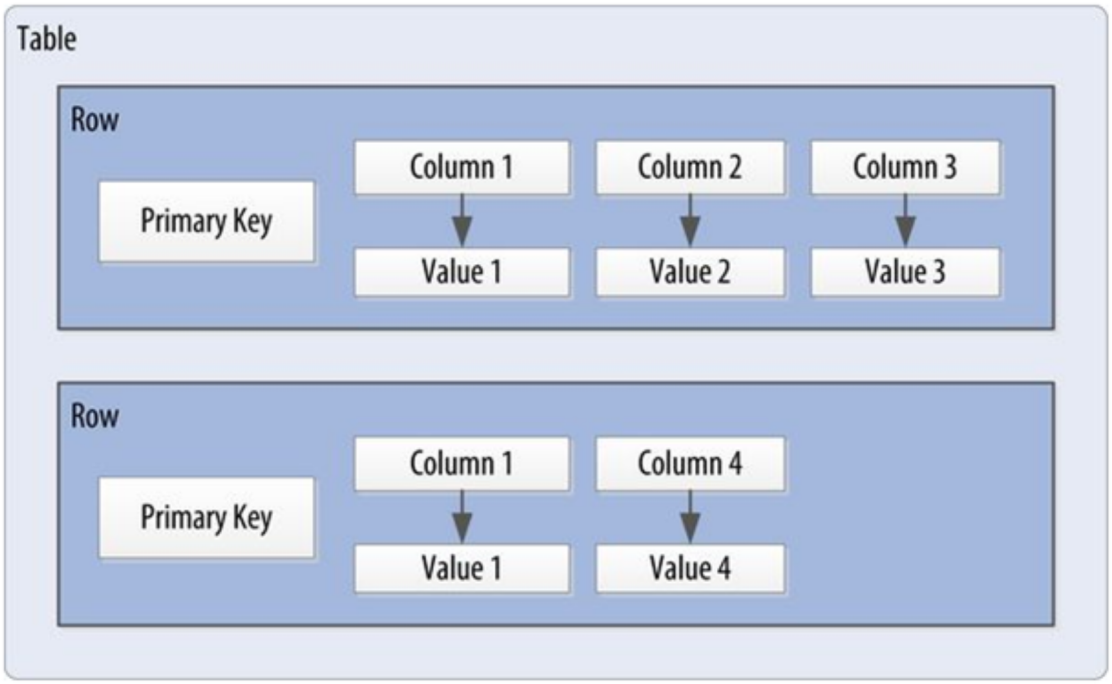

Figura 5.8: Struttura di una tabella in Cassandra

Cassandra Query Language - CQL

Il linguaggio fornito da Cassandra per eseguire delle interrogazioni è il "Cassandra Query Language - CQL". È simile a SQL, i dati sono in *tabelle* che contengono *righe* di *colonne*. Tramite questo linguaggio si possono organizzare i dati all'interno di un cluster con nodi Cassandra. Sono presenti diversi livelli di organizzazione: I "keyspace" definiscono come un dataset deve essere replicato, in quale datacenter e con quante copie. Una "Tabella" definisce il tipo di schema per una collezione di partizioni. Le "Partizioni" definiscono la parte di chiave primaria che tutte le righe devono avere. Se le interrogazioni sono ottimizzate, il valore della chiave di partizionamento deve essere espresso nella query. Le "righe" contengono collezioni di colonne identificate da una chiave primaria. Infine la "colonna" è un singolo valore con un tipo specifico, appartiene ad una riga. I join non sono supportati.

5.1.4 Selezione delle tecnologie

La parte principale del sistema è stata realizzata in Spark. Una valida alternativa sarebbe potuta essere Apache Flink. In Spark sono due i candidati che offrono un servizio di manipolazione di stream di dati: Spark Streaming o Structured Streaming. La scelta tecnologica è ricaduta sulla seconda opzione. Considerati i

modelli dei dati di riferimento, si hanno gli *RDD* e i *DataFrame*. Spark offre delle ottimizzazioni migliori sui piani di lavoro fatti su trasformazioni di *DataFrame*, inoltre è necessario considerare che dopo le trasformazioni, gli stream di dati dovranno essere processati tramite un modello di intelligenza artificiale fornito dalla libreria di Spark: MLlib. Considerato che per questa libreria, non verranno aggiunte funzionalità relative agli *RDD* ma solamente per *DataFrame*, si è deciso di adottare la tecnologia: Spark Structured Streaming. La persistenza dei dati viene eseguita da un Database NoSql. Il motivo per il quale non si è optato per una soluzione RDBMS è il fatto che la priorità viene data alla bassa latenza e distribuzione del sistema. Negli scenari di lavoro che il sistema dovrà gestire, il fatto di avere consistenza è sicuramente un elemento di importanza, ma la priorità viene riservata alla bassa latenza. Considerati diversi scenari di utilizzo, è più grave avere un sistema poco reattivo piuttosto che un sistema che non è sempre consistente. Tra le diverse soluzione NoSql, è stato scelto il database distribuito Cassandra. L'integrazione e la comunicazione con questo database viene fornita dalla libreria: "Datastax". Il livello di cattura degli eventi in input al sistema e trasmissione dei dati tra componenti viene realizzato tramite Apache Kafka.

Figura 5.9: Comunicazione Kafka-Spark-Cassandra

5.2 Architettura del prototipo

Il sistema software progettato, deve essere una soluzione che possa permettere di gestire in input, flussi continui di dati. A seguire deve esservi una fase di manipolazione in modo tale da estrarre una serie di informazioni per la creazione di nuovi dati. Prima di trasformazioni complesse, i dati in ingresso devono essere integrati con informazioni relative alla storia passata di ogni utente. In seguito alla manipolazione e trasformazione, i nuovi dati andranno ad aggiornare lo storico del sistema. È infine richiesto al sistema di effettuare una classificazione basata su un modello di intelligenza artificiale pre-addestrato.

L'architettura del sistema è composta principalmente da tre entità. I topic si occupano di inserire i dati in ingresso e in transito nel sistema, in code ordinate. L'entità che si occupa della manipolazione dei dati è spark Structured Streaming. Dopo essere stati prelevati dai topic, vi è una fase di elaborazione mediante trasformazioni eseguite su *DataFrame*. Successivamente i dati trasformati sono scritti nuovamente su code Kafka. L'ultima entità, identifica il livello di persistenza e gestione dello storico dati tramite Cassandra. Il database può essere sia un punto di memorizzazione dei dati, sia una sorgente. Si possono eseguire delle interrogazioni e ricevere flussi finiti di dati pronti per l'elaborazione.

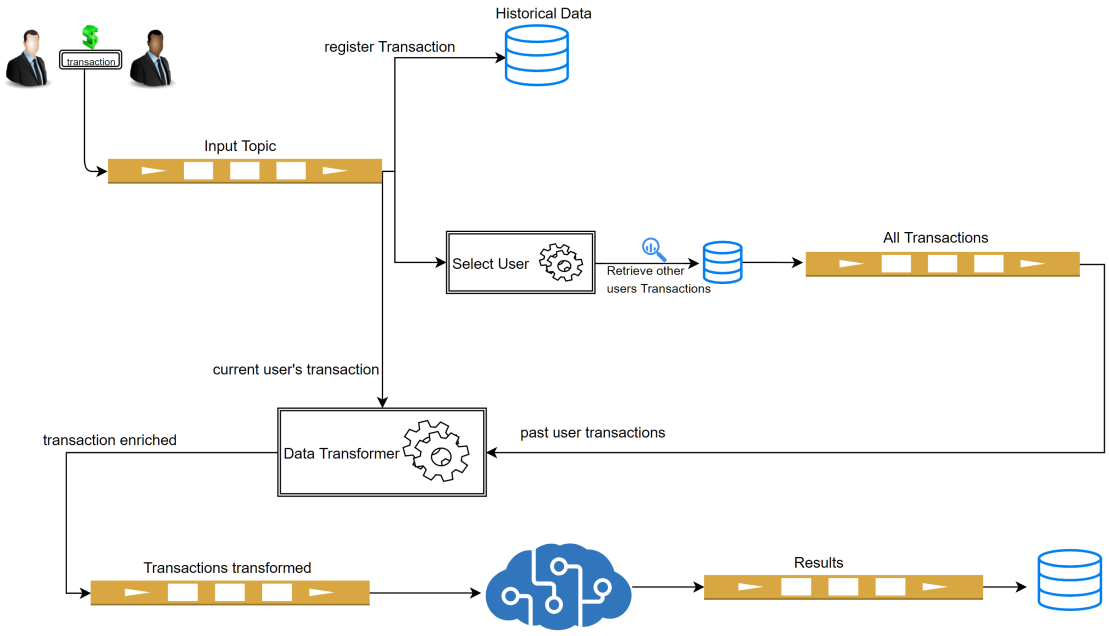

Figura 5.10: Rappresentazione del flusso di lavoro

5.2.1 Design del sistema

Considerata la natura dinamica del sistema è opportuno identificare alcuni elementi principali e modellare il dominio di riferimento basandosi su questi. Sono stati identificati:

- Sorgenti dati.
- Flussi di dati.
- Trasformazioni.

Basandosi sulle tre entità elencate è possibile avere una rappresentazione ad alto livello dell'intero sistema. Qualsiasi operazione può essere espressa come una **trasformazione** che avviene su un **flusso continuo di dati** che vengono spostati da una **sorgente** ad un'altra. Per comprendere nello specifico il design di ogni entità è opportuno entrare più nello specifico nell'analisi di ogni componente.

Flussi di dati

Un flusso di dati è un'astrazione creata con lo scopo di voler catturare la natura dinamica degli stream. In un flusso vengono letti i dati da una sorgente, si ha un meccanismo interno per trasformarli e infine i risultati vengono scritti in una sorgente differente. Possono essere identificati diversi flussi:

- **Input-stream** : Flusso che legge i dati in input al sistema. Partendo da questi, viene eseguita un'interrogazione a un database che detiene i dati storici.

- **Register-Transaction** : Flusso che legge i dati in ingresso al sistema e li memorizza in una base dati persistente.

- **Data-Transformer** : Flusso che legge i dati da diverse sorgenti, effettua un "merge" delle informazioni, applica una serie di trasformazioni che sono il frutto di un'elaborazione tra la transazione in arrivo e lo storico dati.

- **Predict** : Flusso che legge i dati da una sorgente, effettua una previsione sulla base di un modello AI pre-addestrato e invia i risultati in una sorgente finale.

Sono presenti due tipologie di flussi di dati, uno finito e uno potenzialmente infinito. Ne sono un esempio il flusso di dati in ingresso al sistema che è progettato in modo da essere continuo, e il flusso dati relativo a un'interrogazione sql che sposta un quantitativo finito di informazione. A questo scopo sono state create due astrazioni: *AbstractFinishedFlow* e *AbstractStreamingFlow*. Entrambe le classi specializzano un'idea più astratta di flusso di dati: *AbstractFlow*.

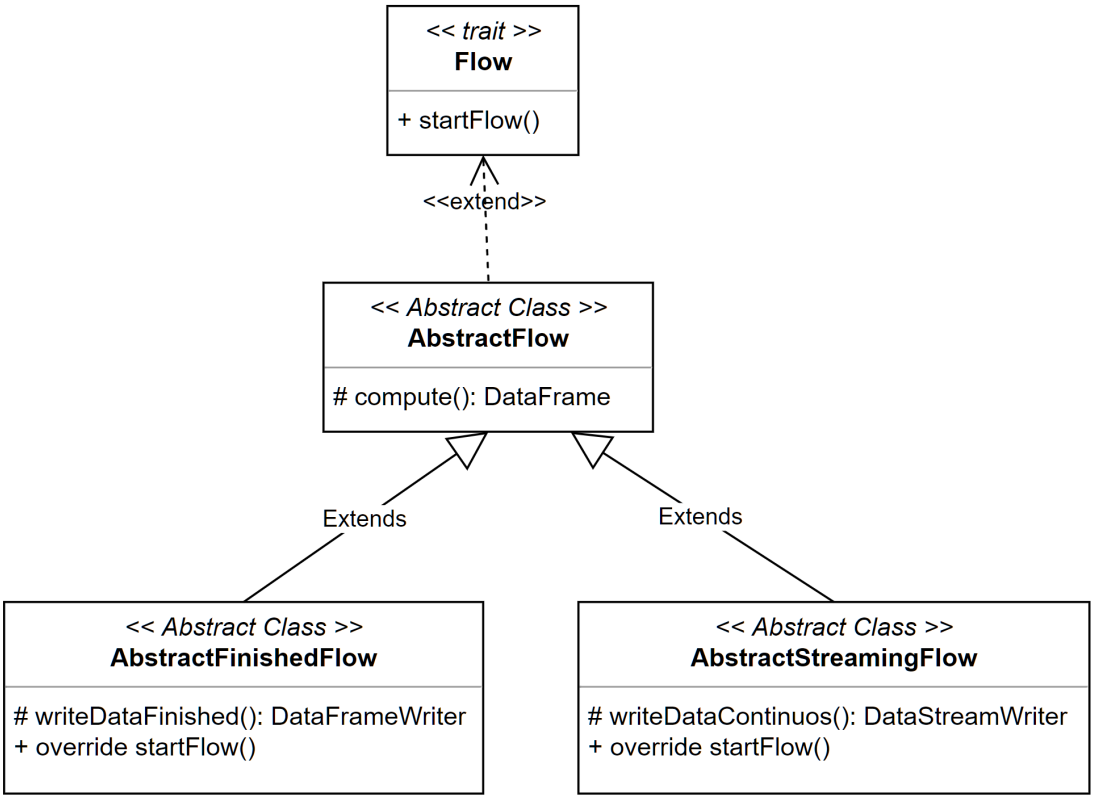

Figura 5.11: Rappresentazione UML delle principali astrazioni per rappresentare un flusso di dati

Sorgenti Dati

Una sorgente dati è un'astrazione creata con lo scopo di voler identificare uno stream di dati. È possibile interagire in due modi con una sorgente. Si devono poter leggere i dati che custodisce e si deve poter scrivere su di essa. Ogni flusso di dati descritto nel paragrafo precedente ha inizio con l'operazione di lettura di uno stream da una sorgente e termina con un'operazione di scrittura su un'altra sorgente. La sorgente non è da considerarsi propriamente come una collezione di dati, bensì come elemento statico che permette l'interazione con una collezione dati specifica. All'interno del sistema sono presenti due sorgenti: le code Kafka (Kafka Topic) e la base dati Cassandra.

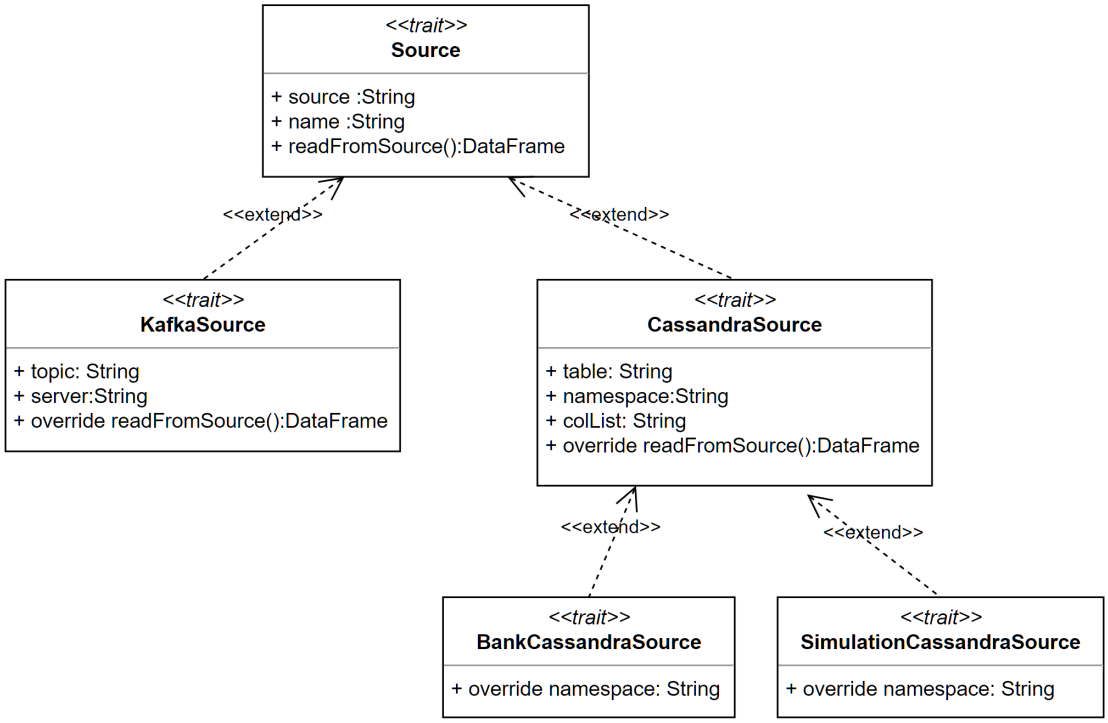

Figura 5.12: Rappresentazione UML delle principali astrazioni per rappresentare una sorgente dati

Kafka Topic

I Topic Kafka sono sorgenti che custodiscono i dati in transito tra flussi differenti. L'input del sistema è memorizzato in topic, i dati verranno consumati, elaborati e memorizzati in un'altra coda dove poi verranno consumati nuovamente da altri componenti. Anche i dati letti da un database possono essere registrati in un topic Kafka prima di essere manipolati e consumati.

Database Cassandra

In seguito a una serie di trasformazioni e risultati ottenuti, il database funge come *layer* per la persistenza delle informazioni. Ogni nuova transazione in ingresso al sistema deve essere memorizzata in modo da essere reperibile per interrogazioni future. In maniera analoga può essere rilevante memorizzare i risultati finali ottenuti. Il sistema può interrogare la base dati per richiedere una serie di informazioni, ogni richiesta genera una risposta finita e i dati potranno essere consumati direttamente dal sistema o pubblicati in una coda kafka e poi consumati successivamente interagendo con il Topic.

Trasformazioni

Le trasformazioni sono tutte le operazioni che vengono eseguite all'interno di un flusso di dati. Determinano tutte le manipolazioni che possono essere eseguite sui dati. Nel prototipo proposto, le operazioni di trasformazione e il modello di intelligenza artificiale sono stati realizzati fornendo delle soluzioni molto semplici. Sarà frutto di estensioni future andare ad estendere questa parte. È possibile principalmente definire due tipi di trasformazioni :

- La manipolazione di un *DataFrame* per aggiungere nuove feature.
- La fase predittiva che verrà eseguita da un modello di intelligenza artificiale pre-addestrato.

Altri componenti

Il sistema è composto da altri componenti minori. Un sistema per monitorare lo stato delle code di dati e un meccanismo per inviare dati all'interno del sistema simulando diversi utenti che generano delle transazioni da analizzare.

Simulazione input

Per simulare un caso d'uso reale si è deciso di leggere i dati relativi alle transazioni bancarie da file e inviarli al topic Kafka che cattura gli eventi in ingresso al sistema.

Sistema di monitoraggio

Tramite il componente *SystemMonitor* è possibile monitorare le code Kafka. In qualsiasi momento si può analizzare lo stato delle code stampandone il loro contenuto su *stdout*. Tramite questo componente, è inoltre possibile osservare il risultato delle previsioni.

5.2.2 Design delle interazioni

L'input del sistema è costituito da transazioni bancarie eseguite da diversi utenti. Ogni transazione deve essere classificata come: *Transazione legittima* o *Transazione fraudolenta*. La transazione in ingresso viene inserita in una coda Kafka. L'informazione deve essere propagata a un database per arricchire la base dati che detiene lo storico dell'utente. È opportuno che vengano recuperate dal database tutte le transazioni dello specifico utente. In questo modo con la transazione corrente e lo storico delle transazioni passate sarà possibile creare una transazione "arricchita" che abbia una serie di informazioni relative alla relazione dell'operazione corrente rispetto le altre operazioni passate. Queste informazioni aggiuntive

sono di primaria importanza ai fini della classificazione che verrà eseguita da un modello predittivo (non ancora integrato all'interno del prototipo). In seguito alla previsione il risultato può essere semplicemente stampato in output o memorizzato su un supporto che fornisca persistenza (es. database).

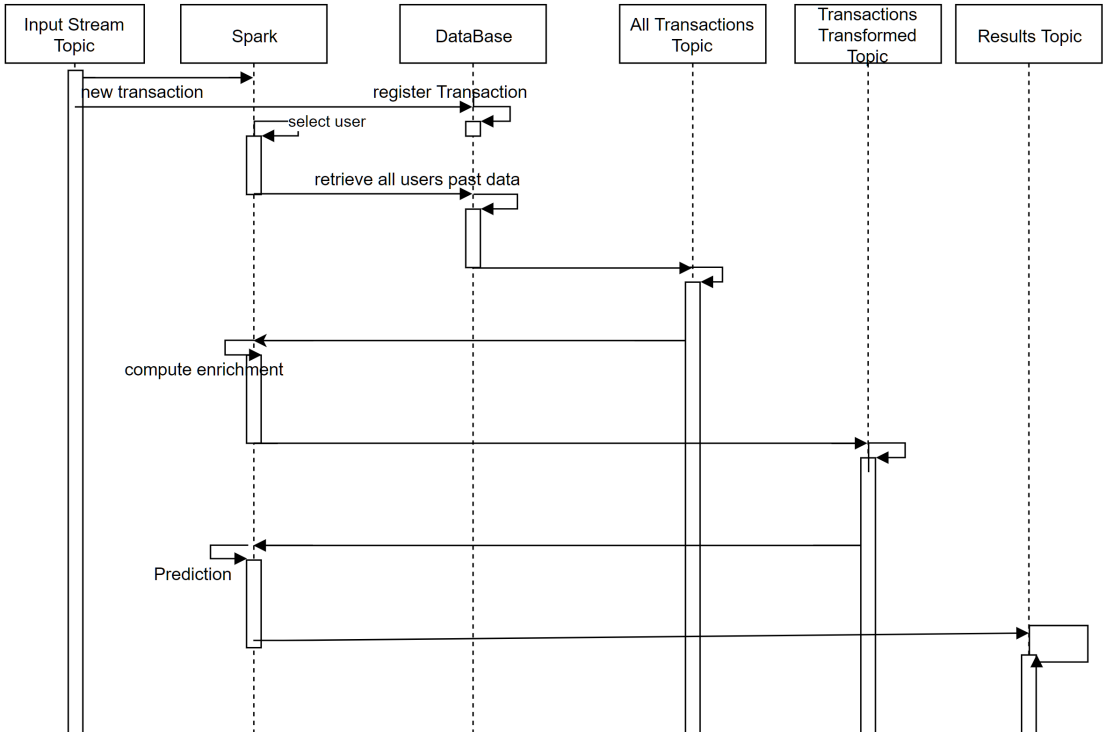

Figura 5.13: Rappresentazione UML del diagramma di sequenza del sistema

5.2.3 Validazione del prototipo

Si è deciso di analizzare diversi meccanismi al fine di comprendere se il tutto funzionasse correttamente. Nello specifico, i controlli sono stati eseguiti analizzando:

- Se il sistema funziona senza incorrere in crash o malfunzionamenti improvvisi. Valutandone la *reliability*.
- Se i dati sono correttamente spostati tra le diverse sorgenti dati.
- Flusso di lavoro complessivo.

Valutazioni *Reliability*

Si è deciso di dare una valutazione su quanto il sistema si potesse considerare *reliable*. È stato valutato avviando la simulazione e misurando il tempo durante il

quale il sistema è rimasto disponibile. Avendo un flusso di transazioni da classificare, si è misurato quanto tempo intercorresse prima che si verificassero problemi, malfunzionamenti o crash del sistema. Sono stati consumati tutti i dati della simulazione (2800 record circa) in 45 minuti avviando la simulazione considerando in input una transazione al secondo. Non si sono verificati problemi. Il medesimo test è stato eseguito 5 volte. Considerati gli esiti positivi si ritiene che in questo primo stadio, il prototipo si possa considerare sufficientemente *reliable*.

Spostamento dati

Uno dei compiti principali del sistema è quello di spostare i dati tra diversi componenti. I dati entrano nel sistema e vengono intercettati da spark, in seguito a una prima elaborazione vengono spostati su una coda kafka e poi sul database. Successivamente vengono nuovamente letti da un altro flusso di dati e dopo eventuali manipolazioni, vengono memorizzati su altre sorgenti. Un aspetto importante che necessita un'analisi specifica è la valutazione se ogni passaggio di dati tra una sorgente e l'altra avviene nella maniera corretta. È un meccanismo abbastanza complesso da monitorare, per questo motivo è stato realizzato il componente: *SystemMonitor* descritto in sezione 5.2.1. Tramite questa entità è possibile in qualsiasi momento analizzare i dati in transito in qualsiasi sorgente dati. Si può infatti stampare su standard output il continuo aggiornamento di ogni coda di dati. In questo modo analizzando il contenuto di ogni topic si è valutato se gli spostamenti dei dati avvenissero in maniera legittima.

Flusso di lavoro totale

Il sistema ha un unico obiettivo: classificare le transazioni bancarie in input come "fraudolenti" o "legittime". È richiesto che a fronte di un input atteso: ovvero una transazione bancaria, in output si abbia la classificazione desiderata. Per analizzare questo aspetto si possono astrarre tutti i procedimenti e operazioni intermedie. Se il sistema nel complesso lavora in maniera corretta, a fronte di una transazione in input, si deve avere il corrispondente output. Non avendo ancora integrato il modello predittivo, non si può valutare il modello, quello che può essere fatto però è osservare se nel complesso il flusso di lavoro rispetta tutti i passaggi e si riesce ad avere per ogni elemento in input un riferimento in output relativo al valore che verrà assunto dalla previsione. Avviando la simulazione vengono lette una serie di transazioni da file e il sistema simula una classificazione del flusso continuo. Analizzando la corrispondenza tra i dati in ingresso e i dati in output si è valutato che il comportamento fosse quello desiderato.

Valutazioni finali

Tramite il sistema descritto si è deciso di presentare una possibile soluzione e architettura per gestire la casistica di manipolazioni di flussi di dati all'interno di un sistema reale da gestire con tecnologie big data. Ci si è focalizzati sull'architettura generale e le interazioni tra componenti, realizzando un sistema funzionante. Tuttavia per avere un sistema completo è opportuno fare delle modifiche. Soprattutto per quanto riguarda la fase di integrazione di un modello di intelligenza artificiale e operazioni di feature engineering.

Conclusioni

All'interno di questo elaborato di tesi sono stati affrontati diversi aspetti legati al mondo del riconoscimento delle frodi. Ci si è concentrati principalmente nello sviluppo di una metodologia che potesse essere utile per affrontare questo problema. Si è dimostrato che non esiste un unico approccio di lavoro, possono esservi diverse soluzioni ognuna con i suoi pregi e difetti. Gli algoritmi di classificazione assumono un ruolo chiave in questo ambito. Un altro aspetto rilevante è la tipologia di dato a disposizione. Si è osservato come la struttura dei dataset, tipicamente, mostra distribuzioni degli elementi tra le classi non omogenee, spesso, sono presenti molti dati relativi a operazioni legittime e molti meno di frodi. Ne è un esempio il dataset sul quale si è lavorato, che presentava una percentuale di frodi pari allo 0,17%. Sono stati studiati e testati i principali meccanismi di bilanciamento dei dati, tra i quali: SMOTE, Adasyn, SMOTE Tomek. Si è osservato che avere dati distribuiti in maniera non omogenea tra le classi può causare problemi per alcuni algoritmi, ne è un esempio SVM; Random Forest a differenza, non risente di questo fattore. Esaminando i risultati ottenuti per i diversi algoritmi di classificazione, è subito emerso come gli approcci che si basano su alberi decisionali come Random Forest e Gradient Boosting mostrino i risultati migliori. Un elemento importante oltre alle misure di precisione e recall è il fatto di avere un modello interpretabile. Per un esperto di dominio è importante poter valutare il criterio di classificazione di un algoritmo al fine di poterne considerare la validità dell'esito. Per poter lavorare su questo tipo di informazioni, i dati devono essere in chiaro, ma nel concreto può spesso succedere che i dataset di questo tipo siano stati resi pubblici dopo una fase di anonimizzazione e opacizzazione delle informazioni per motivi di privacy. È stato necessario adottare tecniche di reverse engineering per recuperare, almeno in parte, le informazioni oscurate. È stato dato un contributo interessante presentando diverse tecniche di feature engineering. Osservando i risultati si può affermare che ha avuto un ruolo chiave la modellazione del comportamento degli utenti. L'idea è stata quella di evidenziare la relazione esistente tra una transazione e l'insieme delle operazioni passate fatte dallo stesso utente. Sono state proposte diverse metodologie per evidenziare questi legami. I principali si basano sulla definizione di un comportamento standard. Vengono in seguito evidenziate

eventuali anomalie analizzando la frequenza delle attività piuttosto che valutando gli scostamenti da specifici punti di riferimento. Un altro aspetto importante è valutare le analisi effettuate su finestre temporali di una determinata lunghezza in modo da considerare il comportamento di un utente in periodi differenti. Sono state proposte due differenti analisi: la prima mira a classificare transazioni relative a utenti storici, la seconda invece vuole essere un approccio più generale che può essere valido anche per utenti mai visti. In ogni caso, ai fini delle valutazioni sul comportamento, si deve avere uno storico dati per ogni utente. Un algoritmo di classificazione che si basa su attributi che mostrano queste relazioni tra i dati riesce a ottenere risultati molto interessanti sia in termini di precisione che di recall. È stata infine proposta e realizzata una possibile architettura software per la gestione di un sistema con infrastruttura big data. Tra le principali sfide vi è il fatto di dover gestire grandi flussi di dati e fornire risposte in tempo reale. Le principali tecnologie identificate per la realizzazione del sistema sono Spark, Kafka e Cassandra. Seppure in uno stato empirico è stata presentata una realizzazione di un sistema funzionante; sarà frutto di implementazioni future completare alcuni componenti. Nel complesso si ritiene che tramite questo elaborato di tesi si possa aver dato un riferimento sulle valutazioni di diversi algoritmi, aver mostrato diverse metodologie e fornito soluzioni su come risolvere i principali problemi. Si ritiene che la strategia proposta relativa all'analisi del comportamento degli utenti sia una metodologia valida nel processo di riconoscimento di frodi.

Sviluppi futuri. Le metodologie presentate potrebbero essere applicate su nuovi dataset con altri tipi di attributi, potrebbe essere interessante partire da questo lavoro di tesi e proporre nuove feature. Si pensi che gli attributi presentati possano essere combinati con dati relativi alla geolocalizzazione dell'utente. Le informazioni relative al dispositivo mobile potrebbero essere combinate con informazioni relative a comunicazioni TCP-IP. Un lavoro di estensione deve essere fatto anche nello sviluppo del prototipo in modo da poterne testare il comportamento su un cluster big data in un ambiente di streaming reale.

Bibliografia

[1] European Central Bank report on card fraud available. https://www.ecb.europa.eu/press/pr/date/2014/html/pr140225.en.html.

[2] fraud value strategy in Vesta Dataset. https://www.kaggle.com/c/ieee-fraud-detection/discussion/101203#588953.

[3] Lexis Nexis true cost of fraud 2016 study. https://risk.lexisnexis.com/insights-resources/research/lexisnexis-2016-true-cost-of-fraud.

[4] User-definition in Vesta Dataset. https://www.kaggle.com/dott1718/ieee-userid-proxy/notebook.

[5] Two modifications of cnn. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(11):769–772, 1976.

[6] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[7] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[8] Francisco Azuaje, Ian Witten, and Frank E. Witten ih, frank e: Data mining: Practical machine learning tools and techniques. *Biomedical Engineering Online - BIOMED ENG ONLINE*, 5:1–2, 01 2006.

[9] S. Barua, M. M. Islam, X. Yao, and K. Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, 2014.

[10] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004.

[11] Gustavo EAPA Batista, Andre CPLF Carvalho, and Maria Carolina Monard. Applying one-sided selection to unbalanced datasets. In *Mexican International Conference on Artificial Intelligence*, pages 315–325. Springer, 2000.

[12] Richard J Bolton, David J Hand, et al. Unsupervised profiling methods for fraud detection. *Credit scoring and credit control VII*, pages 235–255, 2001.

[13] R. Brause, T. Langsdorf, and M. Hepp. Neural data mining for credit card fraud detection. In *Proceedings 11th International Conference on Tools with Artificial Intelligence*, pages 103–106, 1999.

[14] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[15] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Dbsmote: Density-based synthetic minority over-sampling technique. *Applied Intelligence - APIN*, 36, 04 2011.

[16] Fabrizio Carcillo, Andrea Dal Pozzolo, Yann-Aël Le Borgne, Olivier Caelen, Yannis Mazzer, and Gianluca Bontempi. Scarff : A scalable framework for streaming credit card fraud detection with spark. volume 41, page 182–194. Elsevier BV, May 2018.

[17] Philip K Chan, Wei Fan, Andreas L Prodromidis, and Salvatore J Stolfo. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and Their Applications*, 14(6):67–74, 1999.

[18] Nitesh Chawla, David Cieslak, Lawrence Hall, and Ajay Joshi. Automatically countering imbalance and its empirical relationship to cost. *Data Min. Knowl. Discov.*, 17:225–252, 10 2008.

[19] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[20] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6, 2004.

[21] J. Chen, Ye Tao, H. Wang, and T. Chen. Big data based fraud risk management at alibaba. *The Journal of Finance and Data Science*, 1:1–10, 2015.

[22] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[23] Andrea Dal Pozzolo, Olivier Caelen, Yann-Ael Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, 41(10):4915–4928, 2014.

[24] Linda Delamaire, Hussein Abdou, and John Pointon. Credit card fraud and detection techniques: a review. *Banks and Bank systems*, 4(2):57–68, 2009.

[25] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164, 1999.

[26] Jose R Dorronsoro, Francisco Ginel, C Sgnchez, and Carlos S Cruz. Neural fraud detection in credit card operations. *IEEE transactions on neural networks*, 8(4):827–834, 1997.

[27] Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer, 2003.

[28] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[29] Wei Fan, Salvatore J Stolfo, Junxin Zhang, and Philip K Chan. Adacost: misclassification cost-sensitive boosting. In *Icml*, volume 99, pages 97–105. Citeseer, 1999.

[30] Wei Fan, Salvatore J Stolfo, Junxin Zhang, and Philip K Chan. Adacost: misclassification cost-sensitive boosting. In *Icml*, volume 99, pages 97–105. Citeseer, 1999.

[31] Andrew Fast, Lisa Friedland, Marc Maier, Brian Taylor, David Jensen, Henry G Goldberg, and John Komoroske. Relational data pre-processing techniques for improved securities fraud detection. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 941–949, 2007.

[32] Jerome Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29, 11 2000.

[33] Hongyu Guo and Herna L Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM Sigkdd Explorations Newsletter*, 6(1):30–39, 2004.

[34] Hongyu Guo and Herna L. Viktor. Learning from imbalanced data sets with boosting and data generation: The databoost-im approach. *SIGKDD Explor. Newsl.*, 6(1):30–39, June 2004.

[35] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328, 2008.

[36] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. volume 3644, pages 878–887, 09 2005.

[37] Jun Rao Jay Kreps, Neha Narkhede. Kafka: a distributed messaging system for log processing.

[38] Piotr Juszczak, Niall Adams, David Hand, Christopher Whitrow, and David Weston. Off-the-peg and bespoke classifiers for fraud detection. *Computational Statistics & Data Analysis*, 52:4521–4532, 05 2008.

[39] Yufeng Kou, Chang-Tien Lu, Sirirat Sirwongwattana, and Yo-Ping Huang. Survey of fraud detection techniques. In *IEEE International Conference on Networking, Sensing and Control, 2004*, volume 2, pages 749–754. IEEE, 2004.

[40] Maria Krivko. A hybrid model for plastic card fraud detection systems. *Expert Systems with Applications*, 37(8):6070–6076, 2010.

[41] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. pages 63–66, 06 2001.

[42] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. Mllib: Machine learning in apache spark, 2015.

[43] Eric WT Ngai, Yong Hu, Yiu Hing Wong, Yijun Chen, and Xin Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision support systems*, 50(3):559–569, 2011.

[44] A. Patle and D. S. Chouhan. Svm kernel functions for classification. pages 1–9, 2013.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[46] Andrea Dal Pozzolo. Adaptive machine learning for credit card fraud detection.

[47] Daryl Pregibon. Logistic Regression Diagnostics. *The Annals of Statistics*, 9(4):705 – 724, 1981.

[48] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, 1991.

[49] Yusuf Sahin, Serol Bulkan, and Ekrem Duman. A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40(15):5916–5923, 2013.

[50] Naeem Siddiqi. *Credit risk scorecards: developing and implementing intelligent credit scoring*, volume 3. John Wiley & Sons, 2012.

[51] Agus Sudjianto, Sheela Nair, Aijun Zhang, Daniel Kern, Fernando Cela Díaz, and Ming Yuan. Statistical methods for fighting financial crimes. *Quality control and applied statistics*, 56(1):141–142, 2011.

[52] D. K. Tasoulis, N. M. Adams, and D. J. Hand. Unsupervised clustering in streaming data. In *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, pages 638–642, 2006.

[53] Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 75:38–48, 2015.

[54] Paul Viola and Michael Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing System*, 14, 2001.

[55] Gary M Weiss and Foster Provost. The effect of class distribution on classifier learning: an empirical study. 2001.

[56] D. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. Syst. Man Cybern.*, 2:408–421, 1972.