POLITECNICO DI TORINO
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Sviluppo di un sistema a supporto dei processi di validazione dei finanziamenti basato su tecniche di classificazione

Relatore
prof. Luca Cagliero

Candidato
Valentin Popov

Dicembre 2018

Sommario

L'analisi dei dati finanziari ha acquisito e sta acquisendo un ruolo sempre più importante nella società odierna. Essa assume questa importante funzione, in quanto consiste in un insieme di tecniche finalizzate a ricavare delle informazioni su cosa è successo e su cosa succederà, per poter prendere delle decisioni strategiche e operazionali. Esempi di utilizzo possono essere l'individuazione delle frodi finanziarie, caratterizzazione dei propri clienti, analisi del rischio e dell'esposizione e individuazione dei cattivi pagatori.

Secondo il World Bank Group, il "Bank nonperforming loans to total gross loans" in l'Italia nel 2017 si aggira intorno al 14.4%, cioè, il rapporto tra la cifra lorda percentuale a rischio di insolvenza sul totale lordo delle cifre concesse in prestito è 14.4% [1]. Tenuto conto della criticità di tale argomento, i vari istituti finanziari si sono dotati di uffici appositi per la valutazione della bontà del cliente come pagatore e di conseguenza dell'aggiornamento costante delle variabili che possono condizionare tale valutazione, supportandosi anche con istituti privati che permettono l'accesso a differenti dati finanziari legati ai singoli individui, quali il CRIF [2].

In questa tesi saranno trattati i dati finanziari dell'area *Consumer Credit*. L'area Consumer Credit si occupa di fornire ai privati i prestiti a breve e medio termine utilizzati per finanziare l'acquisto di merci o servizi per il consumo personale o per rifinanziare i debiti [3]. Le applicazioni che forniscono l'analisi dei dati finanziari offrono agli uffici proposti degli strumenti che permettono di distinguere un "buon pagatore" da un "cattivo pagatore". Tale distinzione permette a sua volta di individuare con miglior tempismo i finanziamenti a rischio e ridurre le perdite ad essi associate.

Vista questa esigenza e la gran mole di dati da analizzare per il raggiungimento dello scopo preposto, è utile applicare le tecniche di *Machine Learning*.

Il Machine Learning costituisce un insieme di algoritmi di creazione di un modello che apprende da un insieme di dati, dopo di che riceve degli input e ritorna in output delle predizioni su questi, creando in modo induttivo un modello a partire dagli esempi. Questo tipo di apprendimento, però, non tiene conto di alcuni fattori umani, quali la discriminazione razziale e sociale, ed è per questo che i modelli basati su algoritmi di intelligenza artificiale, vanno usati come supporto alle decisioni e non come totale sostituto della parte umana.

I sistemi che attribuiscono i punteggi di credibilità finanziaria alle persone sono già in uso nel mondo, ad esempio il governo cinese ha implementato un modello che svolge questo compito, come descrive l'articolo "China's New Tool for Social Control: A Credit Rating for Everything" di Wall Street Journal [4]: questo sistema oltre a valutare i classici attributi dei dati finanziari mette in correlazione i dati provenienti dai canali "social" con i dati che descrivono l'attività di volontariato e rispetto delle leggi. Il punteggio fornito dal sistema viene utilizzato per stabilire la possibilità di accedere ai prestiti, servizi internet o viaggi.

Visto l'interesse delle società che operano nel settore finanziario di minimizzare le perdite dovute ai prestiti non restituiti, considerati gli aspetti etici derivanti dal Machine Learning, si vuole creare un sistema software a supporto delle decisioni atto a fornire una predizione dell'esito di un finanziamento, dove per esito si intende la conclusione positiva o negativa del pagamento totale del prestito.

Per l'implementazione del sistema descritto sopra, è stata utilizzata l'architettura che prevede i seguenti passi: estrazione dei dati, processing e normalizzazione, creazione del modello predittivo, valutazione del modello su un dato di test.

L'intero processo di analisi ed implementazione è stato svolo presso la società di consulenza informatica *Blue Reply s.r.l.* Sono stati utilizzati i dati storici dei prestiti, l'anagrafica dei clienti ed inoltre è stata consultata una fonte di dati esterna che raccoglie le informazioni di referenza creditizia per la previsione e il controllo dei rischi finanziari - CRIF.

Nella fase di estrazione dei dati, è stata creata una collezione di dati a partire da alcune tabelle del Data Warehouse che contengono la storia dei prestiti personali e l'anagrafica dei clienti. Per rispettare le normative sulla privacy dei dati, essi sono stati anonimizzati in modo da non poter risalire ad un singolo individuo. In questa collezione di dati denominata dataset, ogni record rappresenta il prestito arricchito con le informazioni presenti sull'anagrafica al momento del finanziamento e altri attributi aggregati che descrivono la storia finanziaria di colui che richiede il prestito.

Oltre a queste informazioni è stata considerata anche la storia finanziaria passata del cliente: ad esempio si è tenuto conto di quanti prestiti aveva avuto prima, quanti sono stati saldati con successo, quanti prestiti sono stati concessi da altre società finanziere presenti nel CRIF.

Per la fase di addestramento del sistema sono stati considerati i finanziamenti saldati dal 2008-12-31 al 2013-12-31. Invece per valutare la precisione del modello sono stati considerati i prestiti nati tra il 2013-12-31 ed il 2014-06-30.

Per poter processare questa quantità di dati è stato necessario utilizzare dei tool di *ETL*. Con il termine ETL si fa riferimento ai processi di estrazione (Extract), trasformazione (Trasform) e caricamento (Load) di dati in un Data Warehouse. Lo strumento ETL utilizzato in questa tesi per l'estrazione dei dati storici e la creazione del dataset è *IBM InfoSphere DataStage* [6].

La fase successiva che si occupa di preparazione dei dati, essa serve a migliorare la loro qualità e renderli utilizzabili da un algoritmo di machine learning. Per risolvere questo problema è stato utilizzato *KNIME* [7], una piattaforma open source utile nell'analisi e integrazione di dati.

Alla fine di questo processo è stato creato il modello predittivo utilizzando degli algoritmi di machine learning. Per poter ottenere le performance accettabili durante l'addestramento si è scelto di utilizzare un prodotto che sfrutta il calcolo distribuito sul cloud - *Watson Data Platform IBM* [11]. Watson Data Platform al suo interno utilizza Apache Spark [12] piattaforma ad alte prestazioni studiata appositamente per algoritmi di apprendimento automatico [13]. Nel corso di lavoro sono stati creati differenti modelli utilizzando algoritmi di classificazione come *Decision Tree [17], Logistic Regression [16], Multilayer Perceptron [17], Random Forest [14]* facendo il confronto della loro precisione.

L'obiettivo è costruire un modello capace di concedere i prestiti solo ai "buoni pagatori", ovvero identificare il maggior numero possibile dei finanziamenti saldati.

Il problema sopra citato può essere formulato come un problema di classificazione binaria cost-sensitive [15], dove il costo di commettere un errore nella predizione è più elevato se non viene identificato un buon pagatore rispetto al caso in cui non viene identificato un cattivo pagatore. Il costo maggiore è dovuto al fatto che non riconoscere un buon pagatore porterebbe alla negazione di un prestito a una persona che avrebbe in realtà saldato il finanziamento.

Per costruire un modello di classificazione con elevato richiamo sulla classe d'interesse, è stata applicata una tecnica di oversampling, che consiste nel costruire il dataset per l'addestramento (training set) campionando un numero significativamente più elevato di dati della classe d'interesse al fine di permettere all'algoritmo di cogliere in modo più accurato le caratteristiche salienti dei campioni appartenenti alla classe con costo più elevato.

Dalla valutazione dei modelli risulta che l'algoritmo *Decision Tree* supera altri modelli in termini di precisione, riuscendo ad individuare il 99.9% dei prestiti saldati.Questa informazione risulterebbe utile agli uffici rischi, in quanto, grazie al modello interpretabile il *Decision Tree* riesce a mostrare le nuove tendenze comportamentali dei clienti non ancora noti.

Il sistema implementato sarà inserito come supporto negli uffici rischio per fornire un aiuto agli utenti che devono valutare il rischio nel concedere un prestito finanziario. Il nuovo sistema grazie all'apprendimento automatico impara direttamente dai dati offre il modello più flessibile e dinamico rispetto a quello attualmente utilizzato negli uffici rischio trattati in questo lavoro.

I futuri sviluppi di questo sistema prevedono l'implementazione di una parte che consente l'apprendimento continuo. Il sistema con l'utilizzo delle librerie IBM Watson impara dai nuovi dati, valuta la sua precisione e sostituisce il vecchio modello con quello nuovo se quest'ultimo lo supera in termini di precisione.

Ringraziamenti

Desidero ringraziare tutte le persone che mi sono stati vicine durante il lavoro di tesi, in particolar modo la mia famiglia ed il professore Luca Cagliero per il supporto.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Il lavoro di tesi analizza i dati finanziari allo scopo di costruire un modello predittivo utilizzando le tecniche di machine learning per poter individuare i cattivi pagatori nel ambito *Consumer Credit*. Durante l'implementazione è stato utilizzato un processo che prevede i seguenti passi: estrazione dei dati, processing e normalizzazione, creazione del modello predittivo, valutazione del modello su un dato di test.

Contesto applicativo

Analisi di dati finanziari assume un ruolo sempre più importane nella società odierna. Questo fenomeno è dovuto alla sua capacità di fornire delle informazioni su cosa è successo e cosa succederà a partire dai dati di business immagazzinati dall'azienda. Esempi di utilizzo possono essere l'individuazione delle frodi finanziarie, caratterizzazione dei propri clienti, analisi del rischio e dell'esposizione e individuazione dei cattivi pagatori. Visto l'interesse delle aziende di avere questo tipo di supporto, si vuole costruire un sistema software capace di fare la predizione dell'esito di un finanziamento, dove per esito si intende la conclusione positiva o negativa del pagamento totale del prestito. L'itero processo di analisi ed implementazione sono stati svoli presso la società di consulenza informatica *Blue Reply s.r.l.* utilizzando i dati storici dei prestiti, l'anagrafica dei clienti ed inoltre è stata consultata una fonte di dati esterna che raccoglie le informazioni di referenza creditizia per la previsione e il controllo dei rischi finanziari - CRIF .

Dati utilizzati

Per poter implementare il sistema descritto sopra innanzitutto dobbiamo studiare di che tipo di dati esso necessita, capire la natura dei dati e come integrarli tra loro.

Siamo di fronte ad un sistema software che ha lo scopo di fornire il supporto alle società che erogano i prestiti ai privati. Quindi i dati trattati sono di tipo finanziario che descrivono i prestiti: importo complessivo del denaro finanziato, numero di rate, importo della rata e altri informazioni tecniche riguardanti il prestito. Un altra area di dati riguarda l'anagrafica dei clienti che hanno preso questi prestiti, qui si trovano le informazioni sulla data di nascita, luogo di residenza, stato civile ecc. La terza area che riguarda i dati di interesse riporta le informazioni relativi alla referenza creditizia per la previsione e il controllo dei rischi finanziari presso le altre società in Italia, questi dati sono forniti dalla società CRIF - Centrale Rischi Finanziari S.p.A. [2].

Apprendimento automatico

Vista la notevole mole di dati e la necessità di creare un sistema capace di fare le previsioni sono stati adottate le tecniche di *Machine Learning*[20]. Conosciuto anche con il nome Apprendimento automatico costituisce un insieme di algoritmi di creazione di un modello che apprende da un insieme di dati, dopo di che riceve degli input e ritorna in output delle predizioni su questi, costruendo creando in modo induttivo un modello a partire dagli esempi.

Apprendimento automatico è una disciplina relativamente recente, viene sviluppata all'interno di intelligenza artificiale negli anni cinquanta. Marvin Minsky, Arthur Samuel e Frank Rosenblatt per primi si mostrarono interessati alla possibilità che le macchine imparassero dai dati. In questi anni vengono definite le prime reti neurali, costituite da singoli percettroni [17]. Sempre in questi anni Alan Turing, propose l'ipotesi di una macchina che impara, ovvero in grado di apprendere e alla fine diventare intelligente [21].

I capitoli successivi descrivono le tecniche evolute e le versioni moderne degli algoritmi di apprendimento automatico applicandoli ai dati finanziari con lo scopo di individuare i cattivi pagatori. Il prossimo capitolo intitolato "Dominio di interesse" presenta l'apprendimento automatico e le sue applicazioni, inoltre introduce l'architettura del sistema che è stato realizzato.

Capitolo 2

Dominio di interesse

In questo capitolo saranno presentati le *applicazioni* di machine learning senza approfondire i dettagli tecnici, trattati nel Capitolo 4. Innanzitutto faremo una panoramica delle applicazioni di apprendimento automatico nei nostri giorni, per poi passare alla definizione dell'architettura del sistema che usa i dati finanziari realizzata in questa tesi. Sarà presentato prima il processo di addestramento di un modello di predizione e successivamente l'implementazione di questo processo e gli strumenti utilizzati.

2.1 Applicazioni di machine learning

L'apprendimento automatico viene utilizzato in molti campi dove per risolvere il problema non è possibile creare un algoritmo esplicito. Con questi algoritmi si possono affrontare i problemi come:

1. **Individuare la posta elettronica indesiderata**: Questa applicazione viene utilizzata dai maggiori fornitori del servizio di posta elettronica come *Gmail* [22] per individuare i messaggi indesiderati e spostarli in una apposita cartella chiamata **spam**. Spam di solito contiene i messaggi pubblicitari, tentativi di fishing o altre communicazioni indesiderate. Sul server di posta elettronica i messaggi vengono analizzati con algoritmi di classificazione utilizzando l'analisi predittiva, in continuo aggiornamento per stabilire se un particolare messaggio è spam o meno.

2. **Video sorveglianza**: L'applicazione degli algoritmi di machine learning in campo di video sorveglianza si basa sul *image recognition*[23] che è un inseme di metodi finalizzati all'identificazione dei pattern all'interno delle immagini in formato non strutturato. Con questi metodi possiamo identificare le persone all'interno di un immagine (Video) e riconoscere oggetti potenzialmente pericolosi.

3. **Frodi finanziarie**: L'apprendimento automatico viene applicato con successo nell'individuare le frodi finanziarie e il riciclaggio di denaro. Un sistema di rilevamento antiriciclaggio di denaro può individuare le relazioni e somiglianze tra i dati e, imparare a riconoscere le anomalie, classificare e prevedere gli eventi indesiderati. La soluzione sfrutta le tecniche di apprendimento supervisionato, come la classificazione delle transazioni sospette [24].

4. **Supporto alle strutture sanitarie** : L'analisi predittiva, un caso specifico di apprendimento automatico, può essere utilizzata a supporto delle decisione per le strutture sanitarie. Queste tecniche vengono applicate per la predizione di malattie o per fornire delle informazioni aggiuntive che aiutano nella prescrizione delle cure per i pazienti[25].

5. **Natural language processing (NLP)** : detta anche l'elaborazione del linguaggio naturale è il processo di riconoscimento automatico da parte di un calcolatore delle informazioni scritte o parlate in una lingua naturale. NLP è un processo molto difficile da automatizzare vista l'intrinseca ambiguità del linguaggio umano [26]. Possiamo citare tra le applicazioni che sfruttano queste tecnologie: Siri - assistente digitale sviluppato da Apple Inc.[27], Amazon Alexa assistente personale intelligente sviluppato da Amazon [28].

Le aziende sono sempre più interessate ad utilizzare questo tipo di applicazioni, sopratutto quelle che trattano e immagazzinano grosse quantità di dati di loro business. Il mercato insieme al mondo accademico offre in risposta a questa esigenza i nuovi sviluppi tecnologici nel settore di apprendimento automatico. In futuro sarà sempre più facile estrarre delle informazioni utili a partire dai dati aziendali immagazzinati che possono essere utilizzate sia per fornire le nuove offerte che per minimizzare le perdite dovute agli eventi non ancora noti alla direzione aziendale.

2.2 Stato dell'arte

Considerati questi esempi vediamo in che modo le tecniche di machine learning possono essere applicati ai dati finanziari. Attualmente le società che operano nel settore finanziario utilizzano queste tecnologie per l'analisi preventiva delle frodi finanziarie, per la suddivisione dei suoi clienti in gruppi target per fornire dei prodotti mirati, e per analizzare il rischio individuando i cattivi pagatori.

Secondo il World Bank Group, il "Bank nonperforming loans to total gross loans" per l'Italia nel 2017 si aggira intorno al 14.4%, cioè, il rapporto tra la cifra lorda percentuale a rischio di insolvenza sul totale lordo delle cifre concesse in prestito per cento, è 14.4% [1]. Tenuto conto della criticità di tale argomento, i vari istituti finanziari si sono dotati di uffici appositi per la valutazione della bontà del cliente come pagatore, e di conseguenza dell'aggiornamento costante delle variabili che

possono condizionare tale valutazione, supportandosi anche con istituti privati che permettono l'accesso a differenti dati finanziari legati ai singoli individui, quali il CRIF.

Per indagare sullo stato attuale della valutazione dei rischi nel concedere i prestiti, con l'aiuto di Blue Reply, sono stati consultati degli esperti del dominio che lavorano presso gli uffici rischio dell'area Consumer Credit. Da questa indagine risulta che alcuni uffici rischio hanno un *motore di regole* costruito negli anni in modo programmatico che fornisce una stima sul rischio nel concedere il prestito ad una persona.

Questo motore delle regole che viene utilizzato per valutare i rischi ha due svantaggi principali:

1. Scarsa flessibilità, le regole vengono inserite via codice utilizzando le soglie che devono essere fornite dall'analisi di un esperto dei rischi.

2. Impossibilità di individuare le nuove tendenze dei clienti a partire dai dati. Il motore delle regole non consente di individuare i nuovi pattern nel comportamento dei clienti. Potrebbe essere che col passare degli degli anni una regola non sarà più valida.

Visti i limiti di questo strumento si vuole realizzare un sistema che utilizzi l'apprendimento automatico per trovare i nuovi pattern comportamentali dei clienti e che costruisca autonomamente le regole a partire dalle osservazioni sui dati. Questo nuovo sistema non deve sostituire i vecchi meccanismi di valutazione dei rischi ormai collaudati e funzionanti, ma sarà un ulteriore strumento che aiuta l'addetto alla concezione dei prestiti nel valutare la sua decisione.

2.3 Creazione di un modello predittivo

Considerando il contesto descritto precedentemente si deduce che il motore delle regole utilizzato è poco flessibile, inoltre non riesce ad individuare autonomamente gli attributi che influenzano la futura chiusura del prestito. Per risolvere questi problemi si vuole costruire un sistema che con l'impiego degli algoritmi di machine learning riesca a stabilire le regole imparando dai dati e fornendo il supporto all'addetto ai finanziamenti.

Prima di passare alla concreta implementazione viene descritto il processo di creazione di un modello predittivo che apprende dai dati per poter riconoscere e classificare i record non ancora incontrati:

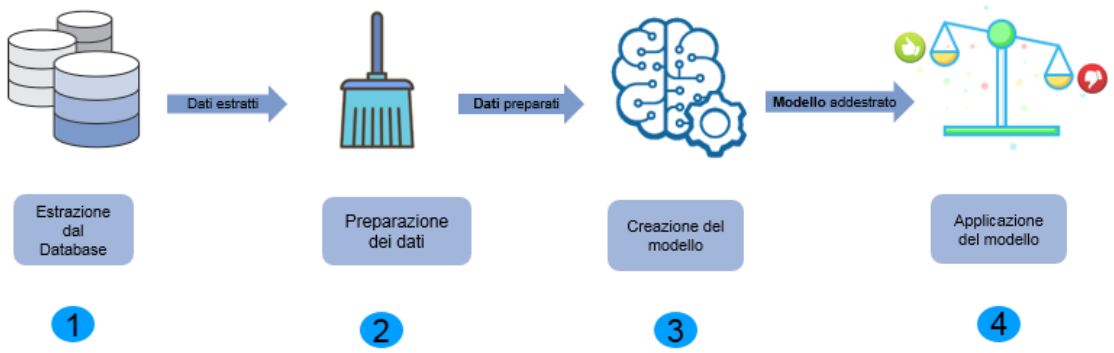


Figura 2.1: Flusso generale

1. Nella prima fase vengono individuati le fonti dei dati di interesse, poi viene effettuata l'estrazione da questi fonti che possono essere un database, file *csv* o altro. L'output di questa fase è un insieme di dati detto anche **dataset** composto da due inseme distinti: *trainig set* - inseme di record che servono per l'addestramento di un algoritmo di machine learning e *test set* che contiene i record sui quali viene valutata la precisione del algoritmo.

2. Nella seconda fase i dati estratti sono processati, i valori mancanti vengono sostituiti con quelli opportuni, viene effettuata la normalizzazione. Vengono selezionati gli attributi utili ai fini della costruzione del modello utilizzando le tecniche di *feature selection*, oppure vengono creati delle nuove features a partire da quelle esistenti. L'output di questa fase è un dataset pronto per essere utilizzato nella fase di creazione del modello.

3. Nella terza fase viene creato il modello predittivo utilizzando degli algoritmi di apprendimento automatico. In questo lavoro saranno utilizzati e confrontati 4 algoritmi *logistic regression*, *decision tree*, *random forest* e una *rete neurale (MLP)*. Alla fine viene selezionato un modello che consiste in un algoritmo addestrato con determinati parametri della sua configurazione.

4. Infine al modello addestrato vengono forniti dei record sconosciuti e per ciascuno di essi il modello fornisce la sua predizione: in questo caso per ogni record del test set viene associata un'etichetta della classe di appartenenza, così il sistema viene testato sui dati non ancora incontrati.

   Successivamente si valuta la precisione del modello: la classe reale di appartenenza e la classe predetta. Si applicando le misure di accuratezza, errore o delle misure definite appositamente. Inoltre se è possibile il modello può essere visualizzato per fornire un'ulteriore informazione . L'output dell'intero processo è un sistema software capace di apprendere dalla storia passata e fornire le previsioni future.

2.4 Architettura del modello realizzato

Dopo questa breve panoramica sul processo di creazione di un modello predittivo possiamo passare alla concreta implementazione del sistema che vuole essere realizzato, descrivendo quali strumenti sono stati utilizzati e perché sono stati scelti.

Nella Figura 2.2 sono stati descritti i tool utilizzati per ogni fase del processo di creazione del modello:

Figura 2.2: Architettura dell'applicazione e tool utilizzati

1. Nella fase di estrazione dei dati è stato utilizzato uno strumento di ETL **IBM InfoSphere DataStage** [6], un software molto versatile che consente di estrarre i dati da sorgenti eterogenei, metterli inseme e creare un file chiamato dataset che sarà utilizzato nelle successive fasi del processo. Questa fase viene trattata in dettaglio nel prossimo capitolo intitolato "Estrazione dei dati".

2. I dati estratti nella fase precedente possono contenere i valori mancanti, alcuni tipi e formati del dato possono essere non rispettati. Questi per poter essere utilizzati da un algoritmo di machine learning, devono essere processati correggendo tutte le problematiche derivanti dalla prima fase. Per processare il dataset prodotto e correggere eventuali errori è stato scelto di utilizzare una piattaforma open source **KNIME** [29], che consente di effettuare delle analisi di tipo statistico sui dati, manipolarli in modo opportuno sostituendo e correggendo i valori degli attributi. Tutti i dettagli su KNIME possono seno stati trattati nel prossimo capitolo.

3. Il terzo modulo prende i dati prodotti da KNIME ed utilizzando un algoritmo di machine learning costruisce il modello predittivo. Data la grande quantità di record si è scelto di utilizzare architetture che sfruttano il calcolo distribuito. **IBM Watson Studio** [11] e **Apache Spark** [12] risolvono questa esigenza. Il modello è stato costruito con l'utilizzo di Apache Spark, mentre IBM Watson Studio offre un *endpoint* attraverso il quale il modello può fornire la sua predizione. L'output di questa fase è un modello addestrato capace di fare la predizione di concedere o meno il prestito richiesto dal cliente.

4. L'ultimo modulo offre una interfaccia grafica per interrogare il modello sfruttando le REST API offerte da IBM Watson Studio. L'utente finale inserisce i dati relativi al prestito e le informazioni anagrafiche del cliente ed il sistema risponde con una predizione futura della chiusura del prestito: saldato o meno. Gli ultimi due moduli sono descritti nel capitolo 4 "Costruzione del modello".

Capitolo 3

Estrazione dei dati

In questo capitolo saranno descritte le fonti dei dati e gli strumenti necessari per la loro estrazione. Per realizzare ciò dobbiamo dobbiamo individuare di che tipo di dati abbiamo bisogno e dove procurarli.

3.1 Fonti delle informazioni

L'area Consumer Credit si occupa di fornire ai privati i prestiti a breve e medio termine utilizzati per finanziare l'acquisto di merci o servizi per il consumo personale o per rifinanziare i debiti [3].

Dal contesto applicativo sappiamo che siamo di fronte a dei dati di tipo finanziario che trattano i finanziamenti e i dati relativi alla anagrafica dei clienti.

Per poter costruire il dataset di esempi sono stati analizzate le fonti forniti dall'azienda dove è stato svolto il lavoro di tesi. Queste fonti consistono in un insieme di database contenenti le informazioni che riguardano l'area Consumer Credit.

In questo lavoro vengono usati gli algoritmi di classificazione. Essi apprendono a partire dagli *esempi* forniti creando una regola generale che associ l'input all'output corretto. Un esempio consiste in una coppia di input e output atteso per quel dato. Quindi per poter creare una regola generale e apprendere dai dati la macchina ha bisognino di moltissimi esempi. Per prima cosa dobbiamo individuare quali dati sono utili e dove prenderli.

Per poter costruire i record di esempio dobbiamo individuare l'insieme di attributi che lo caratterizzano. Quindi prima dobbiamo analizzare i database selezionando tutte le tabelle contenenti le informazioni che riguardano i prestiti e l'anagrafica dei clienti.

Successivamente le tabelle individuate e i rispettivi attributi devono essere presentati all'esperto del dominio per la valutazione. Ogni attributo deve fornire un informazione significativa.

Sono state individuate tre aree nel database fornito: La prima contiene le informazioni relativi ai **contratti**, quest'area ha tutte le informazioni riguardanti le pratiche aperte internamente. La seconda area tratta tutte le informazioni riguardanti **l'anagrafica** dei clienti presenti: anno di nascita, luogo di nascita, stato civile ecc. La terza area (CRIF) conserva la storia dei finanziamenti degli ultimi 3 anni per ogni cliente presso gli **istituti finanziari esterni**:

1. **Area Contratti**, qui possiamo trovare le informazioni riguardanti i prestiti dei clienti. Ogni singolo record rappresenta un prestito che contiene il valore complessivo del finanziamento, la rata mensile, tipo di pagamento ecc.

Tabella 3.1: Informazioni presenti sull'area Contratti

Attributo	**Possibile valore**
Data Caricamento DWH	2017-11-30
Codice cliente	XXXXX
Numero pratica	YYYYY
Data ultima salita SSR	2015-07-13
Data di scadenza del rapporto di lavoro del cliente	null
Importo totale previsto del premio di assicurazione	5000
Contributo casa madre (es. Citroen)	2000
Prezzo bene finanziato	10000
Importo del saldo attuale del cliente	4000
Tipo di pagamento	addebito diretto
Codice agenzia di competenza	RR5
Codice agenzia di gestione	FP4
Data finanziamento	2016-07-15
Flag nuovo cliente	0
Codice agenzia finanziamento	EEE
Importo richiesto	F444
Data anzianità lavoro del cliente	2010-07-15
Data anzianità lavoro del coniuge	2009-02-25
Età del cliente all'atto del finanziamento	31
Anno inizio professione cliente	2008-01-19
Codice posizione attuale	**SALDATA**
Data dell'ultimo finanziamento	2014-10-26
Flag assicurazione	1
Importo finanziato	8000
Importo mensilità	250.70
Numero di mensilità	36
...	...

2. **Anagrafica**, in questa area si trovano le informazioni riguardanti il luogo di residenza, stato civile e reddito dei clienti.

Tabella 3.2: Informazioni presenti sull'area Anagrafica

Attributo	**Possibile valore**
Data Caricamento DWH	2017-11-30
Codice cliente	XXXXX
Numero di Cambiamento di Indirizzi	2
Data Nascita Cliente	1970-12-01
Numero Pratiche Finanziate	2
Numero Pratiche Incorso	1
Numero Salite al Recupero	0
Numero Pratiche Rifiutate	0
Numero figli	0
Nazionalità Cliente	ITAL
Residenza Secondaria	
Stato Civile	Celibe
Codice Professione	lp07
Anno Professionale	1999
Reddito Cliente	1450.00
Reddito Coniuge	0.00
Livello d'istruzione	B
...	...

3. **Prestiti esterni**, nella terza area possiamo trovare le informazioni riguardanti i prestiti che i clienti hanno aperto presso altri istituti finanziari. Questi record provengono dalla fonte esterna gestita da CRIF.

Tabella 3.3: Informazioni presenti sull'area CRIF

Attributo	Possibile valore
Data Caricamento DWH	2017-11-30
Codice cliente	XXXXX
Tipo Pagamento	C
Numero di mensilità	24
Importo mensilità	120.60
Data Finanziamento	2015-07-13
Stato pratica	**SALDATA**
Data ultimo pagamento	2017-07-20
Codice Ente	F444
Tipo Cliente	F
...	...

Ora che abbiamo individuato le aree di nostro interesse, possiamo passare alla fase di creazione del dataset. Questa fase prevede di mettere insieme le informazioni da fonti differenti e collegarli opportunamente.

3.2 Data warehouse di interesse

Le fonti individuate si trovano all'interno di un *data warehouse*. Per poter mettere insieme le informazioni individuate dobbiamo innanzitutto capire la natura di un data warehouse.

Data warehouse nasce dall'esigenda delle aziende di per trasformale i dati aziendali in informazioni utilizzabili per la generazione dei report e per fare delle analisi in funzione del tempo. Esso è una base di dati per il supporto delle decisioni che esiste separatamente dalla basi di dati operativa dell'azienda. William H. Inmon, definisce un data warehouse, come una collezzione di dati *orientata al soggetto, che varia nel tempo, integrata e non volatile*[8]. Ora approfondiamo questi quattro termini:

- **integrata**, visto che all'interno del data warehouse confluiscono i dati provenienti da sistemi transazionali diversi, essi hanno bisogno di essere integrati. Questo scopo può essere raggiunto applicando una codifica uniforme o applicando le stesse unità di misura.

- **orientata al soggetto**, lo scopo del data warehouse è quello di fornire il supporto alle aziende nel analizzare il loro stato intraprendendo una decisione rapida. I dati in un DW devono essere facilmente leggibili, per questo l'obiettivo non è più minimizzare la ridondanza, ma di fornire una visione multidimensionale degli dati stessi.

- **variabile nel tempo**, una delle caratteristiche principale di un DW è la presenza della variabile temporale piuttosto estesa. Può capitare che l'area di interesse rappresentata non sia aggiornata fino ad una certa data, ma nel compenso viene fornita una visione sul evoluzione temporale molto più ampia rispetto ad un sistema transazionale.

- **non volatile**, questo vuol dire che i dati all'interno di un DW non possono essere modificati, gli accessi sono in solo lettura. Questo consente una progettazione più facile visto che non si presentano le anomalie di aggiornamenti. Inoltre non occorre tenere i sistemi sofisticati per bloccare i record durante l'aggiornamento.

Il data warehouse è un sistema OLAP (On-Line Analytical Processing) progettato per analizzare in modo interattivo e veloce le grosse quantità di dati. Per poter eseguire le interrogazioni veloci esso introduce la *denormalizzazione dei dati*. Invece un databese tradizionale è un sistema OLTP (On-Line Transaction Processing) orientato alla transazione. A differenza del sistema precedente esso mira di eseguire le letture e scritture concorrenti garantendo atomicità, coerenza, isolamento e non volatilità dei dati [9].

Ora vediamo su un esempio concreto come viene caricata la tabella anagrafica trovata nell'area esaminata. Come abbiamo visto sopra nel data warehose viene introdotta la dimensione temporale. Dalla data di caricamento indicata come **Data Caricamento DWH** si nota che la periodicità è mesinsile, vuol dire a fine di ogni mese la tabella Anagrafica dal tradizionale database transazionale viene caricata nel data warehouse. Grazie alla dimensione temporale possiamo vedere l'evoluzione dello stipendio nella figura sotto.

Possiamo osservare in Figura 3.1 a sinistra l'evoluzione della tabella relazionale dove lo stipendio del cliente con idCliente = 100 cresce da 1400 nel mese 01/2011 a 1500 nel mese 02/2011. A destra è rappresentato il data warehouse con la tabella anagrafica "storicizzata" dove per ogni mese è stata caricata la fotografia della tabella del sistema transazionale.

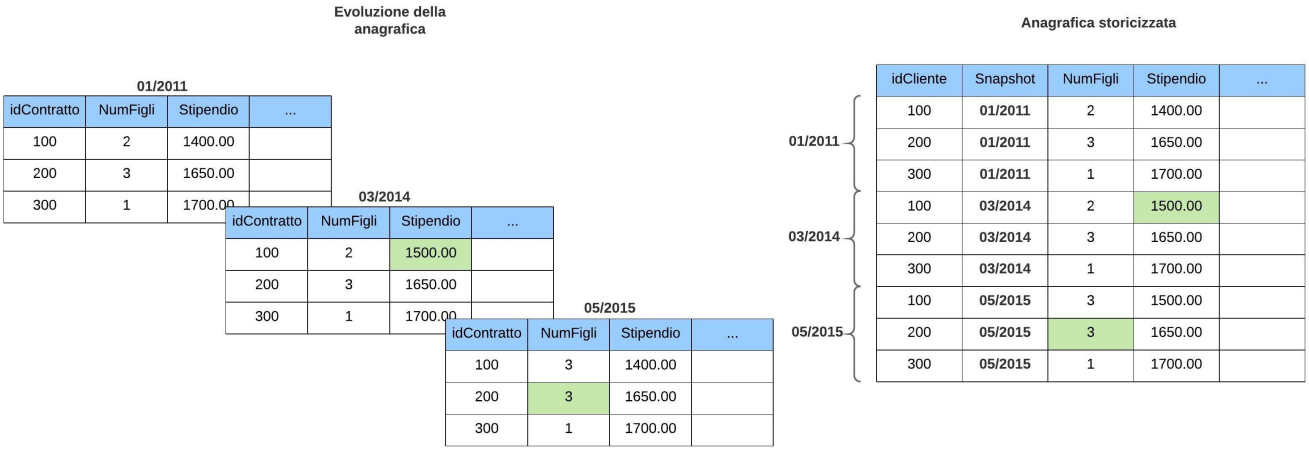


Figura 3.1: Alimentazione dell'anagrafica nel data warehouse

3.3 Processo ETL

Vista la necessità di tenere traccia dell'andamento temporale dei dati sotto differenti punti di vista per poter trarre le decisioni di business viene creato il data warehouse (DW).

Le informazioni contenuti nei database tradizionali orientati alla transazione, devono essere copiate con una certa periodicità nell'area data warehouse calcolando le informazioni aggregati.

I dati alla sorgente possono essere eterogenei quindi occorre tradurre, normalizzare e decodificarli in modo da poter caricare le informazione consistenti in DW.

Dalla necessità di alimentazione di un DW nascono degli strumenti dedicati chiamati tool ETL, dal inglese Extract, Transform, Load. Extract si riferisce al processo di estrazione, Transform alla trasformazione dei dati e Load al caricamento di essi in un DW.

Visto che il processo di estrazione occupa molto tempo, in generale le tre operazioni si fanno in parallelo per migliorare la velocità di caricamento. Mentre i dati vengono estratti il percosso di trasformazione li manipola passando i dati già modificati al processo di caricamento senza completare la prima fase. Approfondiamo ciascuna di questi tre fasi [10]:

1. **Estrazione**: La prima parte del processo, essa consiste nella lettura dei dati dai sistemi sorgenti. I sistemi sorgenti possono essere di carattere eterogeneo: database relazionali, file csv o altro. Una parte intrinseca di questo processo è la validazione dei dati - controllare se il formato delle sorgenti corrisponde a

quelli attesi in dato domino. L'obiettivo di questa fase è quello di estrarre i dati in in un unico formato appropriato per l'elaborazione della trasformazione.

2. **Trasformazione**: In questa fase ai dati estratti vengono applicati una serie funzioni o regole per preparali per la successiva fase di caricamento nel sistema target. Può succedere che i dati non hanno bisogno di alcuna trasformazione, in quel caso essi possono essere caricati direttamente. Di seguito alcuni esempi di questi trasformazioni:

   - Unire i dati da sorgenti differenti utilizzando join, lookup, merge.
   - Selezionare solo i record che hanno con gli attributi valorizzati.
   - Derivare i nuovi attributi a partire da quelli esistenti: tolte delle vendite = prezzo * numero unità .
   - I dati possono essere raggruppati calcolando le informazioni aggregate su di essi. Ad esempio importi delle vendite dei negozi raggruppati per città.

3. **Caricamento**: In questa fase i dati trasformati vengono caricati nel data warehouse. Quindi è importante sapere lo schema del database di target. La periodicità di questi caricamenti può essere giornaliera, settimanale o mensile.

L'itero processo di ETL estrae i dati dai sorgenti eterogenei, aumenta la loro qualità e consistenza e alla fine fornisce i dati in un formato pronto per l'interrogazione da parte degli utenti di business.

3.4 IBM InfoSphere DataStage

Dalle tre are individuate nel paragrafo 3.1 nasce la necessita di estrarre ed integrare i dati da sorgenti differenti: Anagrafica, Contratti, CRIF. Successivamente questi dati devono essere raggruppati calcolando i valori aggregati e alla fine deve essere creato un dataset pronto per l'utilizzo dall'algoritmo di classificazione.

Per effettuare l'estrazione dalle tre aree descritte sopra per creare un unico dataset è stato scelto IBM InfoSphere DataStage. Tale scelta è dovuta dalla necessità di collegarsi alle fonti tramite delle reti privati aziendali utilizzati in Blue Reply. Inoltre IBM DataStage consente di effettuare questa estrazione ed applicare ai dati tutte le trasformazioni necessarie.

Figura 3.2: Logo IBM InfoSphere DataStage [31]

IBM InfoSphere DataStage è una piattaforma ETL che integra i dati di più sistemi eterogenei. Esso sfrutta un'architettura parallela ad alte prestazioni, disponibile in locale o sul cloud. La piattaforma è scalabile ed offre una gestione estesa dei metadati.

Con l'utilizzo di IBM InfoSphere DataStage possiamo integrare i dati eterogenei sia *at rest* che *in motion*. Con il termine data at rest si intendono i dati salvati in modo persistente in database, file, nastri ecc. Mentre i data in motion sono i dati in face di elaborazione dalla CPU o salvari in memoria RAM [5].

DataStage può essere eseguito sia sulle piattaforme distribuite che sui mainframe. Il software utilizza un'architettura client/server[6]:

- il server gestisce informazioni necessarie per eseguire i processi.
- il client, invece, permette mediante l'interfaccia grafica di definire i processi di ETL.

InfoSphere DataStage consente agli utenti di integrare, trasformare e caricare i dati dai sistemi di origine sui sistemi di destinazione sia in modalità batch e in tempo reale. Esso mette a disposizione le funzioni di trasformazione predefinite che consentono l'integrazione dei dati.

Per realizzage i processi ETL si creano dei flussi riutilizzabili in modo che la logica possa essere progettata una sola volta ed eseguita su qualsiasi sistema che utilizza InfoSphere DataStage. Questo tool ETL consente di importare metadati da tabelle e viste.

Dopo questa breve introduzione possiamo introdurre i suoi principali componenti. Il programma eseguibile elementare prende il nome di **job**. Il job permette di fare l'estrazione, trasformazione e caricamento dei dati. Esso è composto al suo interno dagli **stage** che eseguono una funzione dedicata. Per esempio lo stage di lettura di file consente di leggere varie tipologie di file di dati.

Job

I job sono costituiti dagli stage e hanno la forma di un flusso di dati. Il flusso comincia da una lettura della sorgente di interesse, poi i dati letti vengono messi inseme, per poter passare alla fase successiva di caricamento. Per migliorare le prestazioni i dati appena letti vengono elaborati dagli stage successivi senza completare l'intera lettura.

Di seguito viene riportato un job di che legge dal databese, effettua delle trasformazioni per ciascun record letto, e alla fine crea un dataset contenente tutti i record.

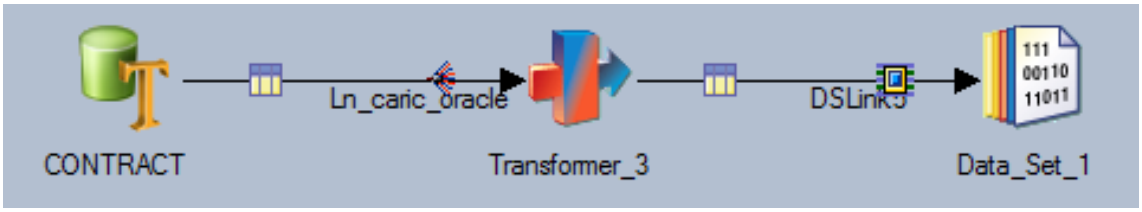


Figura 3.3: Job DataStage contenente 3 stage

Stage

Ogni job è composto dai componenti elementari chiamati stage. Lo scopo dello stage è quello di eseguire una funzione specifica. Lo stage riceve il dato e lo mappa in uscita per il prossimo stage.

Possiamo suddividere gli stage in due gruppi. Il primo gruppo sono gli stage di input-output, chiamati anche *passivi*, in quanto svolgono soltanto la funzione di lettura/scrittura. Questi stage implementano l'operazione di Estrazione e Caricamento in un processo ETL.

Il secondo gruppo è composto dagli stage *attivi* essi effettuano le operazioni di modifica dei dati, conversione, join o altri tipi di trasformazioni.

Nella Figura 3.3 viene riportato un esempio di un job che ha 3 stage: da sinistra a destra:

- Connettore a Teradata: Esegue una query fornita nella configurazione effettuando la connessione al database di Teradata.
- Transformer: esegue una trasformazione sui valori o semplicemente propaga i record in avanti.
- DataSet: stage che scrive tutti i record estratti nel dataset sotto forma di file in un particolare formato proprietario.

Con l'utilizzo di questo tool è stato progetto un flusso di estrazione che crea il dataset utilizzato nella fase di Creazione del modello descritto nel Capitolo 4.

3.5 Query di estrazione

Dopo aver presentato le fonti dei dati e gli strumenti con i quali essi possono essere estratti descriviamo con maggior dettaglio il dataset che serve per la creazione del modello predittivo.

Il record per il dataset è stato costruito andando a prendere per ogni pratica della tabella di contratti il record corrispondente nella tabella di Anagrafica usando come chiave **idCliente** e la **DataSnapshot** (data di caricamento).

In questo modo si ottiene lo *stato del cliente* nel momento di prestito. Allo stesso modo al record del prestito si aggiungono le informazioni sui finanziamenti presso le altre società finanziarie presenti sull'area CRIF:

- Quanti figli aveva il cliente al momento del prestito.
- L'età del cliente al momento di prestito.
- Il saldo al momento di prestito.
- IL reddito al momento di prestito.
- Stato Civile.
- Il salario del coniuge.
- La professione del cliente al momento del prestito.
- La sua residenza (Città).
- Il livello d'istruzione del cliente al momento del prestito.

Ad ogni record è stata associata una etichetta che indica se in prestito è stato chiuso con successo (etichetta "saldata") oppure il prestito è andato al recupero crediti e successivamente in contenzioso (etichetta "contenzioso"). Questo attributo rappresenta la classe di appartenenza di ogni record e l'output per il sistema.

Si definisce:

- **SALDATA**: L'etichetta che rappresenta che il finanziamento è stato saldato con successo.
- **CONTENZIOSO**: L'etichetta che rappresenta che il finanziamento *non è stato saldato*, in questo caso il cliente non ha pagato le rate prestabilite, la pratica è salita al recupero crediti e successivamente è andata in contenzioso.

Inoltre per ogni record della tabella dei contratti al momento dell'apertura del prestito si calcolano dei dati aggregati per costruire delle **features aggiuntivi** che potrebbero influenzare la precisione del futuro modello predittivo.

Possiamo calcolare per ogni prestito il numero, la somma degli importi finanziati dei prestiti con la data di finanziamento **antecedente**, al variare dello stato di chiusura (saldato,contenzioso)

- numero dei prestiti chiusi con successo (saldato)
- importo dei prestiti chiusi con successo (saldato)
- numero dei prestiti chiusi con insuccesso (contenzioso)
- importo dei prestiti chiusi con insuccesso (contenzioso)

Di seguito viene riportata la query utilizzata:

```
SELECT
        MAIN.idCliente,
        MAIN.idContratto,
        MIN(MAIN.data_snapshot)AS data_snapshot,
        MIN(MAIN.data_finanziamento) AS data_finanziamento,

        COUNT(CASE WHEN SLAVE.stato=SALDATA THEN SLAVE.idContratto ELSE null END)
        as NUM_PRA_SALDATA,
        COALESCE(SUM(CASE WHEN SLAVE.stato=SALDATA  THEN  SLAVE.IMP_MEN ELSE 0.00 END),0.00)
        AS IMP_TOT_MES_SALDATA,
        COALESCE(SUM(CASE WHEN SLAVE.stato=SALDATA  THEN  SLAVE.IMP_FIN ELSE 0 END ),0)
        IMP_TOT_SALDATA,
        COALESCE(SUM(CASE WHEN SLAVE.stato=SALDATA  THEN SLAVE.NUM_MEN ELSE 0.00 END),0.00)
        AS NUM_MES_SALDATA,

        COUNT(CASE WHEN SLAVE.stato=CONTENZIOSO THEN SLAVE.idContratto  ELSE null END)
        as NUM_PRA_CONTENZIOSO,
        COALESCE(SUM(CASE WHEN SLAVE.stato=CONTENZIOSO  THEN SLAVE.IMP_MEN ELSE 0.00 END),0.00)
        AS IMP_TOT_MES_CONTENZIOSO,
        COALESCE(SUM(CASE WHEN SLAVE.stato=CONTENZIOSO  THEN SLAVE.IMP_FIN ELSE 0 END),0)
        AS IMP_TOT_CONTENZIOSO,
        COALESCE(SUM(CASE WHEN SLAVE.stato=CONTENZIOSO  THEN SLAVE.NUM_MEN ELSE 0.00 END),0.00)
        AS NUM_MES_CONTENZIOSO,

        COUNT(CASE WHEN SLAVE.stato=CHIUSE_ANTICIPATAMENTE THEN SLAVE.idContratto  ELSE null END)
        as NUM_PRA_CHIUSE_ANTICIPATAMENTE,
        COALESCE(SUM(CASE WHEN SLAVE.stato=CHIUSE_ANTICIPATAMENTE  THEN SLAVE.IMP_MEN ELSE 0.00 END),0.00)
        AS IMP_TOT_MES_CHIUSE_ANTICIPATAMENTE,
        COALESCE(SUM(CASE WHEN SLAVE.stato=CHIUSE_ANTICIPATAMENTE  THEN SLAVE.IMP_FIN ELSE 0 END),0)
        AS IMP_TOT_CHIUSE_ANTICIPATAMENTE,
        COALESCE(SUM(CASE WHEN SLAVE.stato=CHIUSE_ANTICIPATAMENTE  THEN SLAVE.NUM_MEN ELSE 0.00 END),0.00)
        AS NUM_MES_CHIUSE_ANTICIPATAMENTE

        FROM (
        SELECT
          idCliente AS idCliente,idContratto AS idContratto,MIN(data_snapshot)
          AS data_snapshot, MIN(data_finanziamento)
          AS data_finanziamento  from CONTRATTI
        WHERE   stato=ATTIVO  GROUP BY idCliente,idContratto) MAIN

          left outer  JOIN
                        (SELECT
                                SALDATA as stato,
                                idCliente AS idCliente,
                                idContratto AS idContratto,
                                MIN(data_finanziamento) AS data_finanziamento,
                                MAX(CONTRATTI.IMP_FIN) as  IMP_FIN,
                                MAX(CONTRATTI.IMP_MEN) as IMP_MEN,
                                MAX(CONTRATTI.NUM_MEN) as NUM_MEN
                                FROM CONTRATTIT   CONTRATTI
                                WHERE   stato=SALDATA   GROUP BY idCliente,idContratto

                                union all

                                SELECT
```

```
                                CONTENZIOSO as stato,
                                idCliente AS idCliente,
                                idContratto AS idContratto,
                                MIN(data_finanziamento) AS data_finanziamento,
                                MAX(CONTRATTI.IMP_FIN) as IMP_FIN,
                                MAX(CONTRATTI.IMP_MEN) as IMP_MEN,
                                MAX(CONTRATTI.NUM_MEN) as NUM_MEN
                                FROM CONTRATTIT CONTRATTI
                                WHERE stato=CONTENZIOSO GROUP BY idCliente,idContratto

                                union all

                                SELECT
                                CHIUSE_ANTICIPATAMENTE as stato,
                                idCliente AS idCliente,
                                idContratto AS idContratto,
                                MIN(data_finanziamento) AS data_finanziamento,
                                MAX(CONTRATTI.IMP_FIN) as IMP_FIN,
                                MAX(CONTRATTI.IMP_MEN) as IMP_MEN,
                                MAX(CONTRATTI.NUM_MEN) as NUM_MEN
                                FROM CONTRATTIT CONTRATTI
                                WHERE stato=CHIUSE_ANTICIPATAMENTE GROUP BY idCliente,idContratto

                                ) SLAVE
        ON MAIN.idCliente= SLAVE.idCliente
        AND SLAVE.data_finanziamento<MAIN.data_finanziamento

    GROUP BY MAIN.idCliente,MAIN.idContratto,SLAVE.stato
```

Nonostante la sua versatilità nel estrarre e trasformare i dati DataStage non offre gli strumenti per l'analisi dei dati.

Il dataset appena estratto necessita di essere analizzato e ulteriormente trasformato per aumentare la qualità dei dati. Una buona qualità di dati aumenta la precisione di un algoritmo di apprendimento automatico.

3.6 KNIME

Vista la necessità di analizzare e preparare i dati per la fase di addestramento del modello predittivo dobbiamo scegliere uno strumento adatto a questo scopo. Dalle ricerche dei tool risulta che **KNIME** si presta meglio a questa esigenza.

KNIME o Konstanz Information Miner è una piattaforma open source con licenza GPLv3 di analisi dati, reportistica e integrazione. Ha anche al suo interno i componenti di machine learning e data mining. La sua interfaccia grafica permette di collegare i nodi per le operazioni di ETL e di effettuare l'analisi dei dati e la loro visualizzazione [29].

Figura 3.4: Logo KNIME [29]

Questo strumento è stato scelto perché consente di visualizzare la distribuzione dei valori degli attributi e normalizzare i dati. Inoltre esso ha una licenza open source. KNIME implementa una architettura modulare che consente di installare le esenzioni come **Weka** [30].

Weka è un plugin molto utilizzato nella comunità per gli scopi di ricerca e per la didattica. Esso consiste in una serie di tool per il pre-processamento dei dati e data mining. Weka implementa gli algoritmi di classificazione come decsion tree, svm e mlp. Inoltre sono disponibili il clustering e le regole associative [32].

Ora facciamo una breve panoramica sull'architettura di KNIME. Esso è scritto in Java e basato su Eclipse per utilizzare il meccanismo di estensione per aggiungere i plugin che forniscono funzionalità aggiuntive, Weka ne è un esempio.

La versione core include i moduli per l'integrazione dei dati come file, connettori JDBC o connettori nativi PostrgreSQL, SQL Server, SQLite, MySQL. Invece per la trasformazione si possono usare filtri, convertitori, join o altri metodi comunemente applicati in statistica, data mining, e analisi del testo.

KNIME è implementato in Java, ma consente grazie ai wrapper di chiamare altro codice eseguire come Python e Perl [7].

Questo software dal punto di vista di interfaccia grafica è molto simile al DataStage descritto precedentemente. Anche qui si può creare dei flussi di estrazione e manipolazione dei dati.

In particolare KNIME è molto utile per la visualizzazione delle statistiche sui dati (distribuzioni, qualità dei dati) e per la preparazione degli attributi per la fase *addestramento*. Qui di seguito una parte del flusso utilizzata per l'elaborazione dei dati descritti nel paragrafo 3.5:

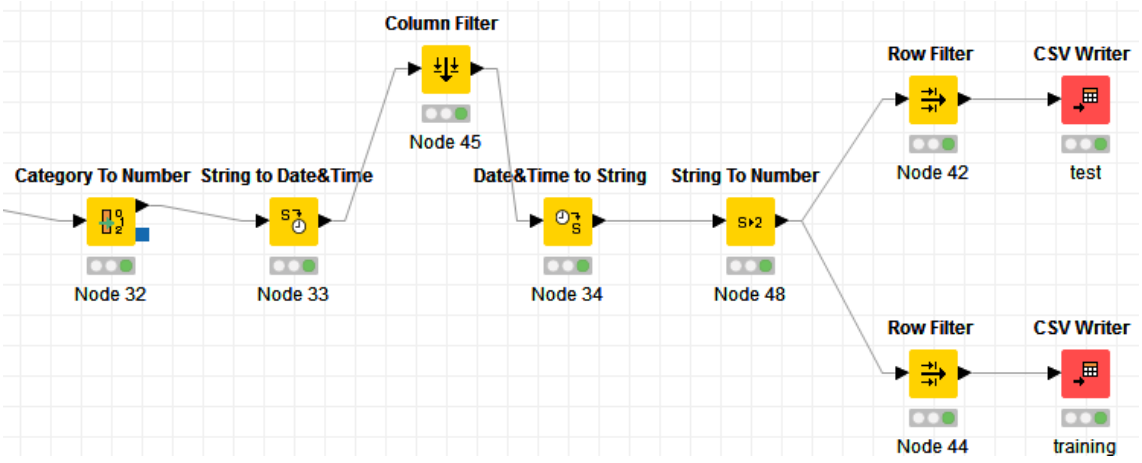


Figura 3.5: Parte di flusso KNIME

Di seguito vengono descritti alcuni dei nodi fondamentali per la preparazione dei dati, che saranno utilizzati nella fase di costruzione del modello [33]:

1. Il nodo **Sampling**: consente di prendere solamente alcuni record da secondo varie strategie: Campionamento stratificato, record presi in modo casuale, oppure una percentuale del intero dataset.

Figura 3.6: Nodo Row Sampling

2. **Category To Number** : Questo nodo prende le colonne con dati nominali e mappa ogni categoria in un numero intero. Per comodità, è possibile elaborare più colonne con questo nodo. Tuttavia, queste colonne vengono elaborate separatamente come se si utilizzasse un singolo nodo Category To Number per ogni colonna. Utile per convertire gli attributi categorici in un range numerico finito.

Figura 3.7: Nodo Category To Number

3. **String to Number**: Converte stringhe in una colonna (o una serie di colonne) in numeri. Se il nodo non riesce ad analizzare una stringa, genererà una cella vuota e aggiungerà un messaggio di avviso sulla console KNIME con informazioni dettagliate.

Figura 3.8: Nodo String To Number

A questo punto abbiamo estratto i dati di interesse in un unico dataset. Poi abbiamo scelto i tool che possono essere utilizzati per preparali per la successiva fase di addestramento che viene descritta nel prossimo capitolo.

Capitolo 4

Creazione del modello

Lo scopo di questo capitolo è quello di creare un modello capace di apprendere a partire dal dataset di esempi descritto nel Capitolo 3.

4.1 Apprendimento automatico

Imparare è una abilità di tutti gli esseri viventi e in particolar modo degli umani. L'apprendimento di solito è costituito da due parti: il processo ed il risultato. Il processo di apprendimento consiste in concentrarsi su un determinato oggetto eliminando le caratteristiche superflue.

Il risultato viene fornito da un insegnante che stabilisce la bontà dei risultati, quindi esso è un processo dinamico.

Possiamo definire l'apprendimento anche come un processo che deduce le regole a partire dagli esempi. Gli esempi appartengono allo spazio di input mentre le regole consistono nelle osservazioni generali dedotti dalla struttura di questo spazio o possono avere la forma di dipendenza tra lo spazio di input e quello di output [35].

Lo scopo dell'apprendimento è quello di: dato un **Trainig set** insieme di N coppie input-output $(x_1, y_1), (x_2, y_2), ...(x_N, y_N)$ dove ogni y_j è stata generata da una funzione sconosciuta $y = f(x)$, trovare una funzione h che approssima la funzione f [17].

Qui x ed y possono essere qualsiasi valore, non necessariamente dei numeri. La funzione h è detta **ipotesi**. L'addestramento consiste nella ricerca all'interno dello spazio delle possibili ipotesi per trovare quella che approssima meglio f.

Per misurare l'accuratezza della funzione ipotesi deve essere fornito un **Test set** insieme di esempi differenti da Trainig set. Si dice che la hunzione h **generalizza** bene se essa predice correttamente il valore di y per gli esempi non ancora incontrati.

Quando il valore di y è compreso in un insieme finito di valori per esempio {sole, pioggia, vento} questo problema di apprendimento automatico prende il nome di **Classificazione**, in questo caso l'algoritmo associa un etichetta al dato di esempio classificandolo.

Invece se y assume un valore numerico come ad esempio tempreratura, il problema di apprendimento prende il nome di **Regressione**. Questo approccio invece fa una previsione esatta e puntuale del dato.

Possiamo suddividere l'apprendimento automatico in tre categorie in funzione del **feedback** disponibile fornito dal sistema di apprendimento [17]:

1. **Apprendimento non supervisionato**: Il sistema impara le regole (o *pattern*) dai dati di input senza alcun feedback fornito dall'esterno. I dati non sono etichettati. L'esempio più famoso dell'apprendimento non supervisionato è **clustering**.

   Clustering può essere definito come processo di organizzazione in gruppi di oggetti simili tra loro. Mentre per Cluster si intende una collezione di oggetti simili tra loro ma allo stesso tempo diversi dagli oggetti appartenenti ad altro cluster[36].

2. **Apprendimento per rinforzo**: Il computer impara da una serie di rinforzi: ricompense o punizioni, interagendo con l'ambiente circostante. In questo approccio occorre di avere un insegnante esterno che gli comunica se ha raggiunto o meno il suo obiettivo. Un esempio di apprendimento per rinforzo può essere la guida di un veicolo o imparare di giocare senza sapere le regole imparandoli dal avversario [37].

3. **Apprendimento supervisionato**: Il sistema impara osservando le *coppie (input,output)* e cerca di dedurre la funzione che mappa l'input all'output. Gli algoritmi che utilizzano questo approccio funzionano bene se si tratta di dati lineari, ingressi simili corrispondono a uscite simili. Nell'apprendimento supervisionato si utilizzano algoritmi come: *Decision Tree, Rete Neurale, Naive Bayes* .

Dai precedenti capitoli sappiamo che siamo di fronte a dei record etichettati, quindi si tratta di apprendimento supervisionato. In esso sono previsti in due insiemi distinti: training e test set.

4.2 Training e Test set

Dalla definizione di apprendimento supervisionato il training e test set sono due insiemi di dati distinti. L'algoritmo costruisce il modello basandosi sui record del training set e valuta la sua accuratezza facendo le predizioni su quelli di test set.

Riprendendo il caso d'uso, si dovrebbe predire l'esito di un finanziamento. Visto che i dati hanno la natura temporale, il training set non può contenere i finanziamenti futuri. L'addestramento deve essere effettuato soltanto con i record aventi la data della chiusura in un certo intervallo [inizio,T].

Mentre le previsioni devono essere effettuati sui record che hanno la data di apertura maggiore della data chiusura dei record di training.

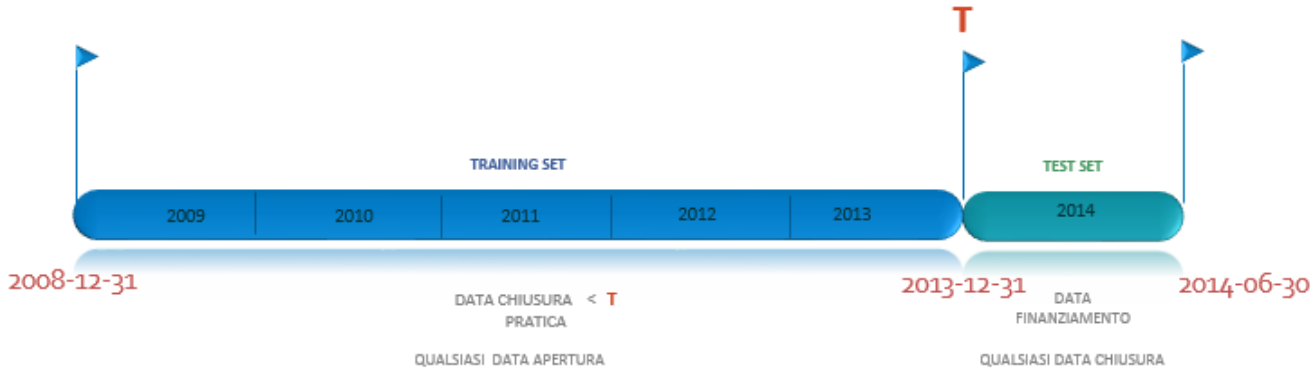


Figura 4.1: Separazione tra il training e test set

La cardinalità dei insiemi ottenuti è la seguente:

- training set: contiene 3858629 record.
- test set: contiene 310214 record.

Prima di passare al addestramento del modello i dati devono essere analizzate allo scopo di creare un training set pronto per essere dato in input al modello predittivo.

4.3 Preparazione dei dati

Dalle analisi effettuati con l'utilizzo di KNIME su training e test set risulta che:

1. le numeriche delle classi nel training set sono sbilanciate.
2. occorre sostituire i valori mancanti con dei opportuni valori.
3. occorre convertire gli attributi data in valori numerici.
4. convertire gli attributi categorici, come provincia di residenza (Torino, Napoli ecc.) in attributi numerici.

Bilanciamento del training set

La prima problematica che deve essere risolta è quella di bilanciare il training set. Questa esigenza nasce dal fatto che le classi target hanno le cardinalità completamente diverse: 3.45% con l'etichetta "contenzioso" e 96.55% "saldata". Quindi per evitare che il modello si addestri a riconoscere soltanto i record con l'etichetta "saldata" il training set deve essere bilanciato. Di seguito in figura sono rappresentati ne numeriche delle due classi.

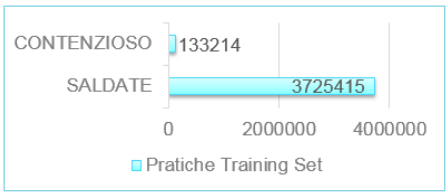


Figura 4.2: Classi sbilanciate nel tarining set

Il problema sopra citato può essere formulato come un problema di classificazione binaria cost-sensitive, dove il costo di commettere un errore nella predizione è più elevato se non viene identificato un buon pagatore rispetto al caso in cui non viene identificato un cattivo pagatore. Il costo maggiore è dovuto al fatto che non riconoscere un buon pagatore porterebbe alla negazione di un prestito a una persona che avrebbe in realtà saldato il finanziamento.

Per esempio, l'errore di un classificatore che deve predire una diagnosi medica (malato, non malato) è un problema cost-sensitive. In questo caso classificare in modo errato una persona malata ha un costo più elevato perché essa non riceverebbe le cure necessarie a causa di una predizione sbagliata.

Anche nel caso dei prestiti finanziari negare il prestito ad una persona che avrebbe pagato ha un costo più elevato di concedere il prestito a un cattivo pagatore. Questo problema può essere risolto con oversempling [15] della classe "saldata" servirci del *campionamento stratificato*.

Il *campionamento* è un processo di selezione da una popolazione statistica di un sottoinsieme di individui capace di stimare le caratteristiche dell'itera popolazione.

In particolare il *campionamento stratificato* crea dei gruppi omogenei prima di effettuare il campionamento[38]. Dopo viene preso una parte uguale di individui dal ciascun gruppo.

Knime fornisce un implementazione pratica del campionamento stratificato nel nodo "Row Sampling".

L'attributo in base al quale si effetua il campionamento stratificato deve essere discriminativo ed avere un chiaro significato. Esso deve creare un certo numero di classi distinti.

Le principali features di un finanziamento sono importo, durata in mesi, data finanziamento.

Consultando l'esperto del dominio e valutando le features specificate prima si è scelto la **durata del finanziamento** come attributo in base al quale effettuare il campionamento stratificato. Per capire la distribuzione delle pratiche in base al numero di mensilità si potrebbe tracciare un istogramma che per il range di durate associa il numero di occorrenze.

Qui sotto viene riportato l'istogramma della classe "saldata" prima di effettuare il campionamento.

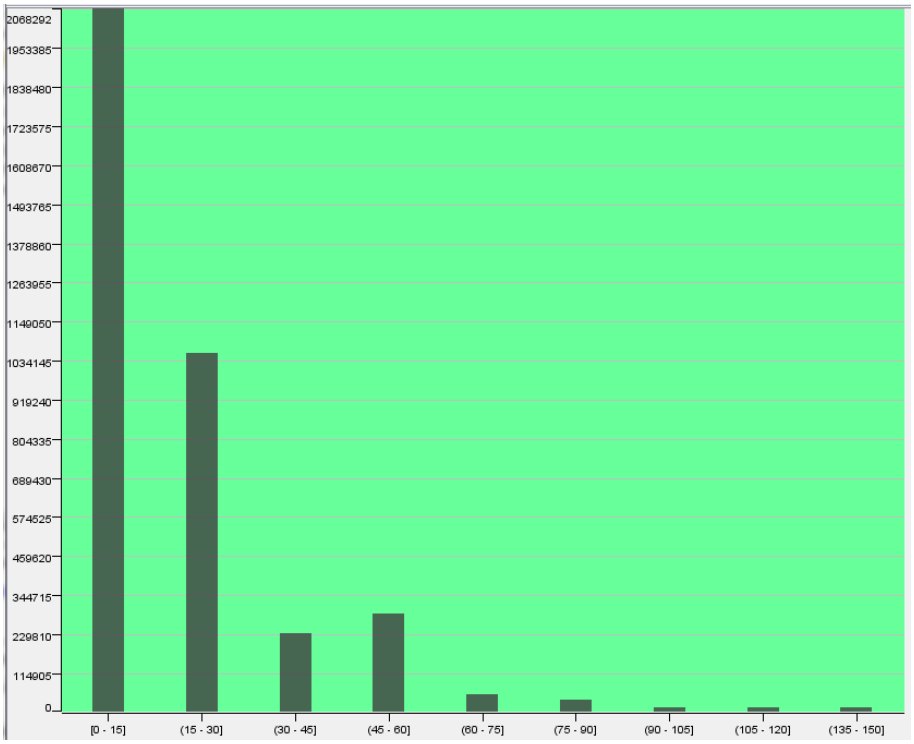


Figura 4.3: Distribuzione della durata della classe "saldata" prima del sampling

Sulle ascisse dell'istogramma è rappresentata la durata del finanziamento, mentre sulle ordinate il numero di occorrenze per ciascuna durata. Si osserva che la maggior parte dei prestiti hanno la durata fino a 15 mesi.

Per effettuare il campionamento stratificato è stato utilizzato il nodo *Row Sampling* configurando la percentuale della classe "saldata" a 8%, 10%, 14% e 18%.

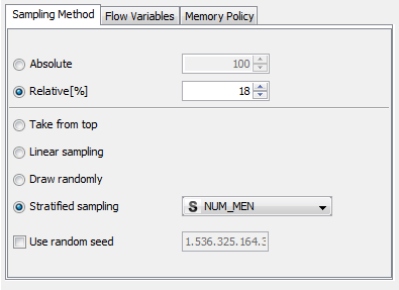


Figura 4.4: Configurazione del nodo *Row Sampling*

Di seguito viene riportata l'istogramma dopo il campionamento, si nota che la distribuzione della popolazione non cambia. Questo suggerisce che il numero delle pratiche "saldate" è stato ridotto conservando la distribuzione iniziale dei dati.

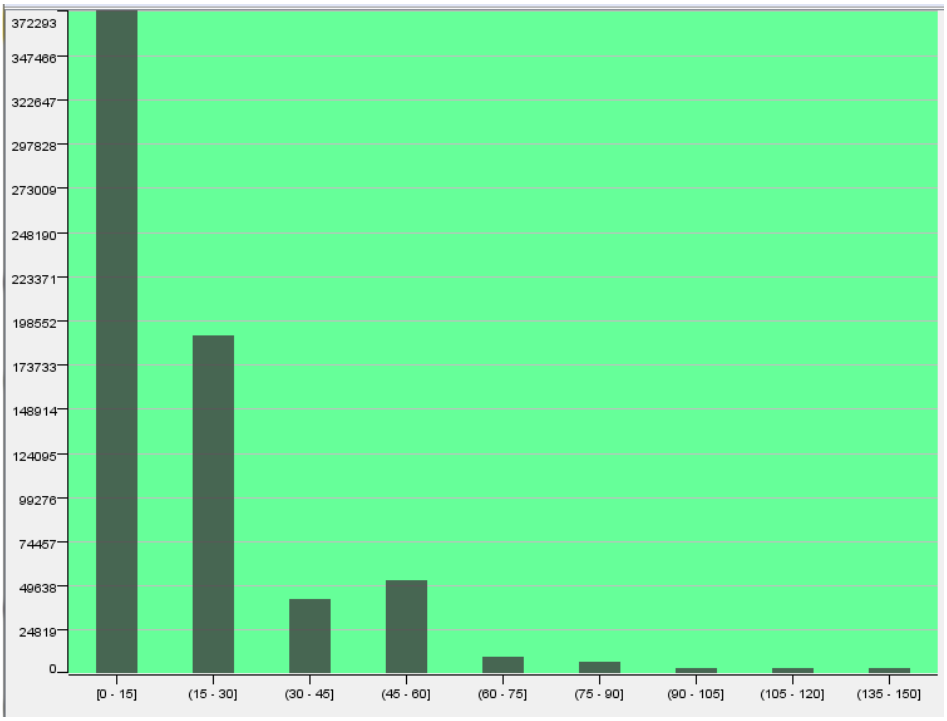
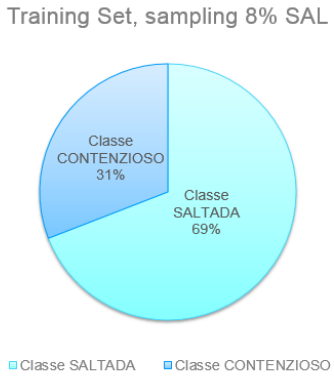


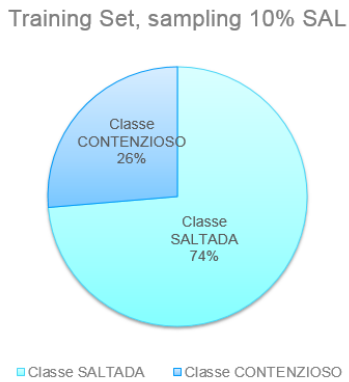
Figura 4.5: Distribuzione della durata della classe "saldata" dopo il sampling

La classe dei prestiti chiusi con insuccesso "contenzioso" non è stata campionata perché è una classe di minoranza. Alla fine le due classi sono stati messi insieme formando il nuovo training set bilanciato. Il numero dei prestiti "saldati" è stato ridotto di un ordine di grandezza.

Vista l'importanza della distribuzione dei dati in trainig set alla classe dei prestiti saldati sono stati applicati quattro campionamenti differenti al 8, 10, 14 e 18 percento.

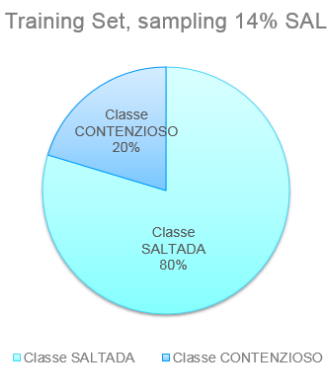


(a) *Training set, sampling della classe positiva 8%, 431247 record.*

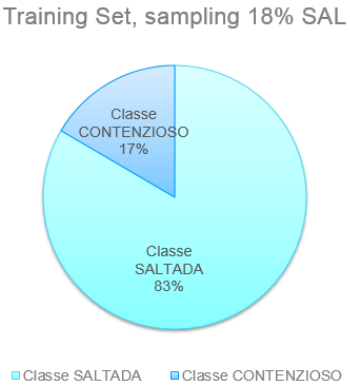


(b) *Training set, sampling della classe positiva 10%, 505755 record.*

Figura 4.6: Training set dopo il sampling 8% e 10%



(a) *Training set, sampling della classe positiva 14%, 654772 record.*



(b) *Training set, sampling della classe positiva 18%, 803788 record.*

Figura 4.7: Training set dopo il sampling 14% e 18%

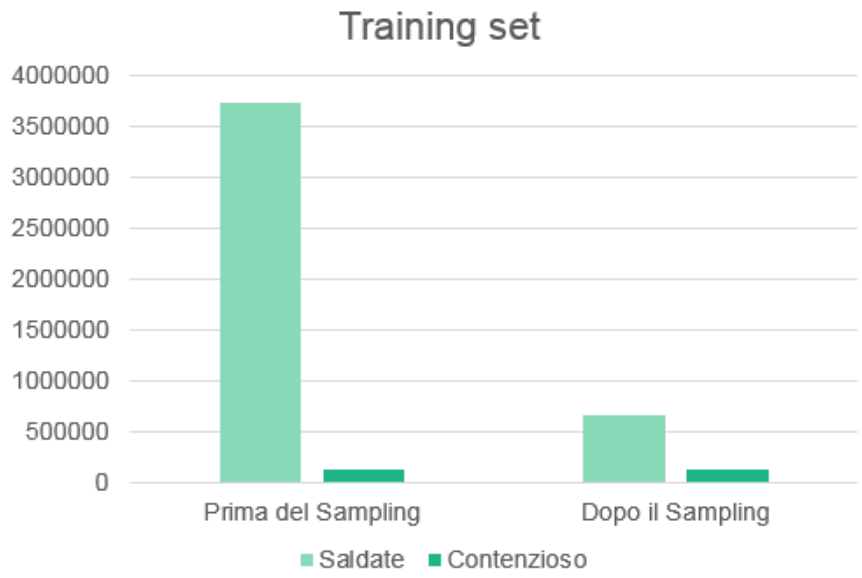


Figura 4.8: Cardinalità delle classi prima e dopo il campionamento

Nella Figura 4.8 sull'asse delle ordinate sono rappresentati le numeriche delle classi "saldata" e "contenzioso". Per fare il confronto a sinistra è stato messo il training set prima del campionamento, si può osservare che le pratiche saldate sono in netta maggioranza. Mentre a destra è rappresentato il training set dopo il campionamento dove il numero dei finanziamenti "saldate" è diminuito, ottenendo così una collezione di esempi che non penalizzerà le pratiche contenziose in fase di addestramento.

Le problematiche riguardanti i valori mancanti sono stati risolte con l'aiuto di KNIME. Ogni valore mancante è stato sostituito con un valore non presente nel dominio del attributo per non alterare in alcun modo i dati.

Tutte le date sono state rappresentate come differenza in giorni dalla data di finanziamento. Per esempio l'attributo "anno professione", che rappresenta la data della assunzione del cliente, è stato sostituito con la differenza in giorni : data finanziamento - anno professione.

Tutti gli attributi categorici sono stati sostituiti con i numeri, per aumentare le prestazioni in fase di addestramento. Per risolvere questo problema è stato utilizzato il nodo *category to number*.

4.4 Algoritmi applicati

Dato il problema di costruire il modello capace di fare le predizioni sull'esito di un finanziamento dobbiamo scegliere un algoritmo di classificazione adatto al nostro caso.

Di seguito verranno esaminati 4 algoritmi di classificazione capaci di addestrarsi su un insieme di esempi e fornire la predizione.

4.4.1 Logistic regression

Detto anche Regressione Logistica è un metodo popolare per prevedere una risposta categorica. È un caso speciale dei modelli lineari generalizzati che predice la probabilità dei risultati. La regressione logistica può essere utilizzata per prevedere un risultato binario utilizzando la regressione logistica binomiale, oppure può essere utilizzata per prevedere un risultato multiclasse utilizzando la regressione logistica multinomiale.[39]

Come descritto nel paragrafo 4.1 lo scopo di un classificatore è quello di trovare la la funzione h che approssima $y = f(x)$. Supponiamo di avere gli esempi di due classi 0 e 1. Ogni esempio è definito da due valori di input x_1 e x_2.

La funzione h prende in input un nuovo punto (x_1, x_2) e ritorna una etichetta 0 o 1.

Un modo più semplice di separare i dati in due classi è quello di creare una frontiera di decisione lineare : retta o superficie in caso multidimensionale. Di seguito viene riportato un esempio di una possibile funzione di separazione, dove x è un vettore delle features e w è un vettore dei pesi associati [17].

$$h_w(x) = \begin{cases} 1 & w \cdot x \geq 0 \\ 0 & altrimenti \end{cases} \tag{4.1}$$

Possiamo subito notare che questa non è una funzione continua che associa una predizione completamente confidente anche per i punti che sono vicini alla frontiera.

Questo problema può essere risolto rilassando il vincolo di classificazione con una funzione continua:

Per i problemi di classificazione binaria, l'algoritmo genera un modello di regressione logistica binaria. Dato un nuovo punto dati x, il modello effettua previsioni applicando la **funzione logistica**

$$h_w(x) = \frac{1}{1 + e^{-z}} \tag{4.2}$$

dove $z = w \cdot x$. L'output del modello di regressione logistica, $f(z)$, ha un'interpretazione probabilistica (cioè, la probabilità che x sia positiva). Per definizione, se $h(z) > 0.5$, l'otput è positivo, altrimenti è negativo.

La regressione logistica binaria può essere generalizzata in regressione logistica multinomiale per addestrare e prevedere problemi di classificazione multiclasse. Ad esempio, per K possibili risultati, uno dei risultati può essere scelto come un "pivot", e gli altri $K-1$ risultati possono essere separatamente regrediti rispetto al risultato del pivot.

Per i problemi di classificazione multiclasse, l'algoritmo produrrà un modello di regressione logistica multinomiale, che contiene $K-1$ modelli di regressione logistica binaria regrediti contro la prima classe. Dati nuovi esempi, $K-1$ i modelli verranno eseguiti e la classe con maggiore probabilità verrà scelta come classe prevista [40].

4.4.2 Decision tree

Denominato anche Albero decisionale è uno dei metodi molto diffusi per la classificazione. Gli alberi decisionali sono ampiamente utilizzati poiché sono *facili da interpretare*, gestiscono le features categoriali, non richiedono il ridimensionamento delle features e sono in grado di lavorare con i dati non lineari[46].

Algoritmo base

L'albero delle decisioni è un algoritmo greedy (avido) che esegue un partizionamento binario ricorsivo dello spazio delle features. L'albero predice la stessa etichetta per ogni partizione più bassa (foglia). Ogni partizione viene scelta in modo greedy selezionando la migliore divisione da un insieme di possibili divisioni, al fine di massimizzare il guadagno di informazioni su un nodo dell'albero [17].

In altre parole, la divisione scelta su ciascun nodo dell'albero viene presa dal set

$$\max_{s} IG(D, s) \tag{4.3}$$

dove $IG(D, s)$ è il guadagno di informazioni quando una divisione s viene applicata a un set di dati D.

Impurità del nodo e guadagno

L'impurità del nodo è una misura dell'omogeneità delle etichette sul nodo. Si usano delle misure di impurità per la classificazione :

1. **Varianza**: Si utilizza per la regressione. Con y_i si indica l'etichetta per un istanza, N numero di istanze e μ la media data da $\frac{1}{N}\sum_{i=1}^{N} y_i$

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2 \tag{4.4}$$

2. **Gini index**: Si utilizza per la classificazione. Con f_i si indica la frequenza dell'etichetta i su un nodo e C è il numero di etichette univoche [43].

$$\sum_{i=1}^{C} f_i(1 - f_i) \tag{4.5}$$

3. **Entropy**: Si utilizza per la classificazione. Con f_i si indica la frequenza dell'etichetta i su un nodo e C è il numero di etichette univoche [44].

$$\sum_{i=1}^{C} -f_i \log(f_i) \tag{4.6}$$

Invece il **guadagno** di informazioni è la differenza tra l'impurità del nodo genitore e la somma ponderata delle due impurità del nodo figlio. Supponendo che le partizioni di una divisione divengano il set di dati D di dimensione N in due serie di dati D_{left} e D_{right} di dimensioni N_{left} e N_{right}, rispettivamente, il guadagno di informazioni è [45]:

$$IG(D, s) = Impurity(D) - \frac{N_{left}}{N} Impurity(D_{left}) - \frac{N_{right}}{N} Impurity(D_{right}) \tag{4.7}$$

Condizioni di Split

1. **Attributi Continui**: Alcune implementazioni ordinano i valori delle feature e quindi usano i valori univoci ordinati come candidati split per il calcolo dell'albero. Questo approccio funziona solo per i dataset di piccoli dimensioni.

   Ordinare i valori delle feature è costoso per i data set distribuiti. Per ovviare a questo problema, altre implementazioni calcolano un insieme approssimativo di candidati split eseguendo il calcolo su una frazione campionata dei dati [46].

2. **Attributi Categorici**: Per gli attributi categorici con M possibili valori (categorie), i candidati split potrebbero diventare $2^{M-1} - 1$. Per la classificazione e la regressione binarie (0/1), possiamo ridurre il numero di candidati split a $M - 1$ ordinando i valori delle feature categoriali in base all'etichetta media.

Nella classificazione multiclasse, tutti i $2^{M-1}-1$ possibili divisioni vengono utilizzate ogni volta che è possibile. Quando $2^{M-1}-1$ è maggiore del parametro *maxBins*, utilizziamo un metodo (euristico) simile al metodo utilizzato per la classificazione e la regressione binarie. I valori della feature categoriale M sono ordinati per impurità e vengono considerati i candidati allo split $M-1$ risultanti [46].

Regola di arresto dell'algoritmo

La struttura ad albero ricorsiva viene arrestata su un nodo quando si verifica una delle seguenti condizioni:

- La profondità del nodo è uguale al parametro di training *maxDepth*.
- Nessun candidato split porta a un guadagno di informazioni superiore a *minInfoGain*.

La particolare implementazione e significato di questi parametri saranno descritti in seguito.

4.4.3 Random forest

Random forest sono insiemi di alberi decisionali[47]. Le Random forest sono uno dei modelli di apprendimento automatico di maggior successo per la classificazione e la regressione. Come gli alberi decisionali, le Random forest gestiscono le features categoriali, permettono la classificazione multiclasse, non richiedono il ridimensionamento delle funzioni e sono in grado di lavorare con i dati non lineari.

Algoritmo di base

Le Random forest addestrano separatamente una serie di alberi decisionali , quindi l'addestramento può essere fatto in parallelo. L'algoritmo inietta la casualità nel processo di addestramento in modo che ogni albero decisionale sia leggermente diverso. La combinazione delle predizioni di ogni albero riduce la varianza, migliorando le prestazioni dei dati di test[48].

Training
La casualità introdotta nel processo di creazione include:

1. Sottocampionamento del set di dati originale su ogni iterazione per ottenere un set di addestramento diverso (a.k.a. bootstrap).

2. Considerando diversi sottoinsiemi casuali di features da suddividere su ciascun nodo dell'albero.

Oltre a queste randomizzazioni, l'addestramento dell'albero decisionale viene eseguito allo stesso modo dei singoli alberi decisionali.

Predizione

Per fare una previsione su una nuova istanza, una l'algoritmo deve aggregare le previsioni dal suo insieme di alberi decisionali. Questa aggregazione è fatta in modo diverso per la classificazione e la regressione.

Classificazione: voto di maggioranza. La previsione di ogni albero viene considerata come un voto per una classe. Si prevede che l'etichetta sia la classe che riceve il maggior numero di voti.

Regressione: media. Ogni albero predice un valore reale. Si prevede che l'etichetta sia la media delle previsioni dell'albero.

4.4.4 Multilayer Perceptron

Multilayer Perceptron è un classificatore basato sulla rete neurale *artificiale*. Come mostrato in figura sotto una rete neurale è composta da dei neuroni connessi tra di loro con dei link con i vari pesi associati.

Questa struttura descrive una rete neurale in generale.

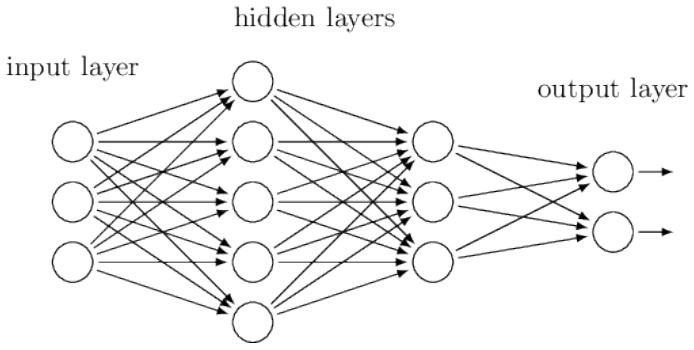


Figura 4.9: Rete neurale [18]

La differenza tra una rete neurale ed un'altra è dovuta dalla struttura interna dei neuroni, dalle funzioni utilizzate per determinare l'output e dai meccanismi utilizzati per l'apprendimento. La figura precedente mostra appunta una possibile struttura di una rete neurale.[17].

Nell'immagine si può osservare che ogni nodo è collegato ad un altro attraverso i *link*. Un link da i a j serve a propagare *l'attivazione* a_i. Ogni link inoltre ha un peso numerico $w_{i,j}$ che serve a stabilire l'importanza e segno della connessione.

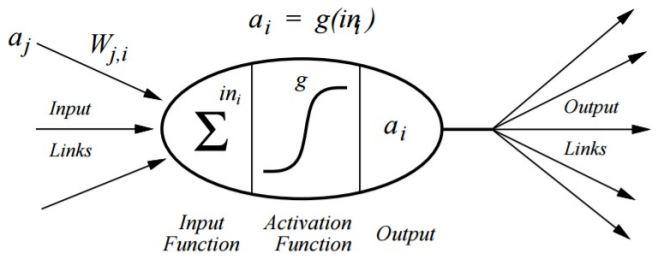
Ogni nodo j per prima calcola la somma pesata dei suoi input, a_0 può essere scelto a piacere.

$$in_j = \sum_{i=0}^{n} w_{i,j} a_i \tag{4.8}$$

Successivamente il nodo applica una funzione di attivazione g a questa somma per determinare l'output:

$$a_j = g(in_j) = g(\sum_{i=0}^{n} w_{i,j} a_i) \tag{4.9}$$

In questo caso il nodo prende il nome di precettrone, schematicamente rappresentato nella figura sotto.



$$a_i = g(\sum_j W_{j,i} a_j)$$

Figura 4.10: Schema generale di un precettrone [19]

Activaction function - una funzione che definisce l'output del neurone a partire da un input. Qui potrebbe essere utilizzata qualsiasi funzione. Le più utilizzate sono la funzione sigmoid, o la tangente iperbolica[50].

4.5 Strumenti per la creazione del modello predittivo

Dopo aver preparato i dati ed esaminato gli algoritmi di classificazione possiamo dedicarci alla fase successiva che si occupa di addestramento dei algoritmi descritti sopra.

Vista la grande quantità di dati, per poter ottenere le performance accettabili durante l'addestramento si è scelto di utilizzare un prodotto che sfrutta il calcolo distribuito sul cloud - *Watson Data Platform IBM* [11]. Watson Data Platform al suo interno utilizza Apache Spark [12] piattaforma ad alte prestazioni studiata appositamente per algoritmi di apprendimento automatico [13].

Watson Data Platform IBM

Watson Data Platform è una piattaforma dasata sul cloud per la costruzione dei modelli predittivi. Essa offre un insieme di servizi cloud integrati, che sfruttano gli algoritmi di intelligenza artificiale per l'analisi dei dati di business.

Figura 4.11: IBM Watson Logo [41]

Al suo interno Watson Data Platform utilizza l'architettura **Apache Spark** [12]. Apache Spark è una piattaforma open source di calcolo distribuito.

Spark estende il popolare modello MapReduce per supportare le query interattivi e streaming. Nella versione 2.1 utilizzata in questo lavoro, Spark sfrutta i DataFrame che è una evoluzione del RDD (Resilient Distributed Datasets). RDD è una collezione distribuita di elementi suddivisa in varie partizioni che possono essere calcolati sui nodi differenti del cluster [51].

DataFrame a sua volta fornisce un interfaccia più facile da usare per lo sviluppatore introducendo differenti linguaggi come Java, Python, Scala.

Ad alto livello , ogni applicazione Spark è costituita da un programma driver che esegue varie operazioni parallele su un cluster. L'astrazione principale che Spark

Figura 4.12: Apache Spark Logo [42]

fornisce è un resilient distributed dataset RDD), che è una raccolta di elementi partizionati attraverso i nodi del cluster che possono essere utilizzati in parallelo.

Gli RDD vengono creati inizializzando un file nel file system Hadoop (o qualsiasi altro file system supportato da Hadoop) o con una collezione Scala esistente nel programma del driver. Gli utenti possono salvare un RDD in memoria, consentendone il riutilizzo in modo efficiente attraverso operazioni parallele. Infine, gli RDD si ripristinano automaticamente dai guasti del nodo.

Una seconda astrazione in Spark è costituita da variabili condivise che possono essere utilizzate in operazioni parallele. Per impostazione predefinita, quando Spark esegue una funzione in parallelo come un insieme di attività su nodi diversi, invia una copia di ciascuna variabile utilizzata nella funzione a ciascuna attività. A volte, una variabile deve essere condivisa tra le attività o tra le attività e il programma del driver.

Spark supporta due tipi di variabili condivise: le variabili di trasmissione, che possono essere utilizzate per memorizzare un valore in memoria su tutti i nodi e gli accumulatori, che sono solo variabili "aggiunte", come contatori e somme.

I DataFrame come le Pipeline appartengono alla libreria **Machine Learning Library (MLlib)** [52]. La Pipeline definisce delle API per combinare più algoritmi di machine learning in un singolo flusso.

I principali componenti della pipeline sono:

- **Transformer**: un astrazione che include le funzionalità di feature trasforming e di addestramento dei modelli. Il modello prende in input un DataFrame, legge le colonne contenenti le features, predice una etichetta per ogni record, e restituisce in output un altro DataFrame con le predizioni associate in una nuova colonna.
- **Estimator**: un astrazione del concetto di un algoritmo di addestramento, per esempio Logistic Regression o Decision tree.

Pipeline è una sequenza di nodi che possono essere Transformer o Estimator. Questi nodi vengono eseguiti in sequenza, il DataFrame di input viene elaborato passando attraverso ogni nodo. Per ogni nodo Transformer viene invocato il metodo transform() sul DataFrame. Per ogni nodo Estimator viene invocato il motodo fit() per creare Transformer che farà parte del PipelineModel a questo punto viene invocato il metodo transform() sul DataFrame del Transformer creato.

4.5.1 Creazione della Pipeline

A questo punto abbiamo preparato i dati per l'addestramento e abbiamo selezionato quattro algoritmi di classificazione. Ora passiamo alla creazione dei modello utilizzando questi algoritmi in Apache Spark.

Lettura dei dati

Il Traning set e il Test set sono stati caricati sulla piattaforma Watson Data Platfor all'interno del DB2 database. Qui vengono creati due DataFrame *test_data* e *test_data* leggendo dalle tabelle precedentemente caricate in DB2.

```
test_data=spark.read.jdbc(db2_credentials['jdbcurl'], table=loan_test, properties=db2_credentials)
train_data=spark.read.jdbc(db2_credentials['jdbcurl'], table=loan_train, properties=db2_credentials)
```

Creazione del vettore delle features

Il vettore delle features consiste in un inseme delle colonne che sono l'input per il modello. Si ricorda che tutti gli attributi che compongono il vettore delle features sono stati descritti nel catitolo 3.

```
features_in=[
'COD_MOD_PAG_CAT',
'REDD_CON',
'IMP_SAL_CLT',
'IMP_TOT_MES_17',
'IMP_TOT_17',
```

```
'NUM_MES_17',
'NUM_PRA_10',
'IMP_TOT_MES_10',
'IMP_TOT_10',
'NUM_MES_10',
'NUM_PRA_50',
'IMP_TOT_MES_50',
'IMP_TOT_50',
'NUM_MES_50',
'NUM_PRA_ES',
'IMP_TOT_MES_ES',
'IMP_TOT_ES',
'NUM_MES_ES',
'IND_POS',
'CAMB_IND',
'ANN_BAN',
'ANN_MAT',
'ANN_NAS_CON',
'ANN_PROF_CON',
'ANN_RES',
'COD_PROF',
'COD_PROF_CON',
'IMP_ASS_TOT',
'IMP_CNTR',
'IMP_MAT',
'IMP_MAXI',
'TIP_PAG',
'CAP',
'COD_AGE_COMP',
,...]
```

Infine viene creato il vettore delle features

```
# create vector of features
vectorAssembler_features = VectorAssembler(inputCols=features_in, outputCol="features")
vdf = vectorAssembler_features.transform(train_data)
```

Creazione degli estimator

Allo scopo di creare un modello predittivo saranno confrontati tra di loro 4 estimator differenti utilizzando gli algoritmi descritti precedentemente:

1. **LogisticRegressionModel** [53]

   ```
   pyspark.mllib.classification.LogisticRegressionModel
   (weights, intercept, numFeatures, numClasses)
   ```

   Con parametri :

   - weights - pesi calcolati per ogni feature.
   - intercept: intercetta calcolata per questo modello. (Utilizzato solo nella regressione logistica binaria. Nella regressione logistica multinomiale, le intercettazioni non saranno considerate come valore singolo, quindi le intercettazioni faranno parte dei pesi).
   - numFeatures: la dimensione delle features.
   - numClasses: il numero di possibili risultati per il problema di classificazione delle classi k nella regressione logistica multinomiale. Di default, è una regressione logistica binaria, quindi numClass sarà impostato su 2.

2. **DecisionTreeClassifier** [46]

   `pyspark.ml.classification.DecisionTreeClassifier`

   I parametri utilizzati dall'algoritmo sono:

   - *maxDepth*: profondità massima di un albero. Gli alberi più profondi sono più espressivi (potenzialmente permettono una maggiore accuratezza), ma sono anche più costosi da addestrare in termini di tempo.
   - *minInstancesPerNode*: affinché un nodo sia ulteriormente diviso, ciascuno dei suoi figli deve ricevere almeno questo numero di istanze di addestramento. Questo è comunemente usato con RandomForest dal momento che quelli sono spesso addestrati più in profondità rispetto ai singoli alberi.
   - *minInfoGain*: per dividere ulteriormente un nodo, la divisione deve migliorare almeno in termini di guadagno di informazioni.
   - *maxBins*: L'aumento dei maxbins consente all'algoritmo di prendere in considerazione più candidati divisi e di prendere decisioni divise a grana fine. Si noti che il parametro maxBins deve essere almeno il numero massimo di categorie M per qualsiasi feature categoriale.
   - *maxMemoryInMB*: quantità di memoria da utilizzare per la raccolta di statistiche. Il valore predefinito viene scelto di 256 MB per consentire all'algoritmo decisionale di funzionare nella maggior parte degli scenari. Aumentare maxMemoryInMB può portare ad un allenamento più veloce (se la memoria è disponibile) consentendo un minor numero di passaggi sui dati. Tuttavia, potrebbero esserci rendimenti decrescenti con il crescere di maxMemoryInMB poiché la quantità di comunicazione su ciascuna iterazione può essere proporzionale a maxMemoryInMB.
   - *impurità*: misura dell'impurezza (discussa sopra) utilizzata per scegliere tra le divisioni candidate.

3. **Random forest classifier** [48]

   `pyspark.ml.classification.RandomForestClassifier`

   I parametri utilizzati dall'algoritmo sono:

   - *numTrees*: numero di alberi. Aumentando il numero di alberi diminuirà la varianza nelle previsioni. Il tempo di allenamento aumenta approssimativamente in modo lineare nel numero di alberi.

- *maxDepth*: profondità massima di ogni albero. Aumentare la profondità rende il modello più espressivo e potente. Tuttavia, gli alberi profondi impiegano più tempo ad addestrarsi. In generale, è accettabile addestrare alberi più profondi quando si usano Random forest rispetto a quando si utilizza un singolo albero decisionale. È più probabile che un albero decisionale vada oltre il limite di un Random forest (a causa della riduzione della varianza dalla media di più alberi).
- *subsamplingRate*: questo parametro specifica la dimensione dell'insieme di dati utilizzato per addestrare ciascun albero, come una frazione della dimensione del set di dati originale. L'impostazione predefinita (1.0) è consigliata, ma la riduzione di questa frazione può accelerare l'allenamento.
- *featureSubsetStrategy*: numero di funzioni da utilizzare come candidati per la suddivisione in ciascun nodo dell'albero. Il numero viene specificato come frazione o funzione del numero totale di funzioni. La riduzione di questo numero accelera la formazione, ma a volte può influire sulle prestazioni se troppo bassa.

4. **MultilayerPerceptronClassifier** [49]

   `pyspark.ml.classification.MultilayerPerceptronClassifier`

   I parametri utilizzati dall'algoritmo sono:

   - *layers* parametro che rappresenta la configurazione della rete in termini di numero di nodi. Per esempio il vettore layers = [4, 5, 8, 3] specifica 4 nodi nello strato di input, 3 nodi nello strato di output, mentre 5 e 8 sono numeri dei nodi nei livelli intermedi.

     Inoltre il numero dei nodi in input deve essere uguale alla dimensione del vettore delle features, mentre il numero dei nodi in uscita deve essere uguale al numero delle classi.
   - *maxiter* numero massimo di iterazioni.

4.5.2 Creazione del modello in Spark

```
#classificatore
dt = DecisionTreeClassifier(labelCol="label", featuresCol="features",maxDepth=5)
# indexed labels back to original labels.
labelConverter=IndexToString(inputCol="prediction",
outputCol="predictedLabel",labels=stringIndexer_label.labels)
#build the pipeline
pipeline_dt=Pipeline(stages=[stringIndexer_label,vectorAssembler_features, dt,labelConverter])
model = pipeline_dt.fit(train_data)
```

A questo punto possiamo vedere la precisione del modello creato:

```
predictions = model.transform(test_data)
```

Capitolo 5

Tuning dei modelli

In questo capitolo saranno confrontati i modelli costruiti in precedenza per determinare la *migliore* funzione ipotesi h sui *nuovi dati*. Prima di tutto dobbiamo definire cosa vuol dire "migliore" e "nuovi dati" [17].

5.1 Metodi di validazione

Facciamo un assunzione sulla *stazionarità* dei nostri esempi. Questo vuol dire che la distribuzione di probabilità sui esempi rimane stazionaria nel tempo. Ogni esempio che soddisfa questa assunzione viene chiamato *indipendente e distribuito identicamente*.

Questa assunzione collega il passato con il futuro, senza di essa il futuro può essere qualsiasi cosa.

Ora dobbiamo definire cosa vuol dire l'ipotesi "migliore". Definiamo il *tasso di errore* della funzione ipotesi la proporzione dei errori che essa fa, la proporzione quando $h(x) \neq f(x)$ per un esempio (x, y). Però se la funzione h ha un tasso di errore basso non vuol dire che essa generalizza bene. Per esempio un professore sa che valutare gli studenti sulle prove già visti non produce un risultato accurato.

Holdout cross-validation

Per ottenere una misura accurata della funzione ipotesi occorre testarla sui dati non ancora incontrati. L'approccio più semplice può essere suddividere il dataset a disposizione in **training set** da cui l'algoritmo di addestramento produce h, e **test set** insieme di esempi per la valutare la sua accuratezza. Di solito questo approccio prende il nome di holdout cross-validation.

Lo svantaggio di questo metodo che esso non utilizza tutti i dati disponibili, utilizzando la metà dei dati per il test per l'addestramento rimane solo l'altra metà. Dall'altro lato se lasciamo pochi dati per il test, per esempio 10%, la stima

di accuratezza viene fatta solo su alcuni esempi e non sulla maggior parte della distribuzione.

k-fold cross-validation

Questo approccio può essere la soluzione al problema precedente. L'idea è quella di separare i dati in k sottoinsiemi. Si effettuano k addestramenti dove ogni volta $1/k$ di dati è utilizzato come test, mentre i dati rimanenti sono utilizzati per l'addestramento. I sottoinsiemi non devono essere sovrapposti. Alla fine si prende la media dei risultati della accuratezza, che sarà più precisa di una singola misura.

I popolari valori di k sono 5 e 10. Tuttavia il miglioramento dell'accuratezza ha un costo computazionale maggiore, 5 o 10 volte più grande. Il caso particolare quando k è uguale a n prende il nome di **leave-one-out cross-validation**, con n la cardinalità del dataset.

Validation set

Consideriamo un altro problema di addestramento: overfitting. Questo fenomeno accade quando il modello si aderisce perfettamente ai dati utilizzati in fase di addestramento. Infatti non appena il modello addestrato incontra un esempio leggermente diverso da quelli incontrati in precedenza sbaglia la predizione, e non generalizza bene. Per ovviare al fenomeno di overfitting dobbiamo introdurre un terzo insieme di esempi chiamato **validation set**. Questo insieme si tiene da parte e viene utilizzato come prova finale della accuratezza, valutando se l'algoritmo generalizza bene riuscendo a classificare correttamente anche gli esempi non ancora incontrati.

5.2 Misure

Per valutare l'accuratezza di un modello dobbiamo utilizzare alcune misure. Prima di parlare della precisione e accuratezza, dobbiamo trovare una possibile rappresentazione dell'accuratezza. Una di queste rappresentazioni può essere la **matrice di confusione**.

Nelle colonne di questa matrice troviamo i valori predetti dall'algoritmo. Invece le righe rappresentano i valori effettivi. L'elemento della matrice $x_{i,j}$ rappresenta il numero delle volte in cui il classificatore ha identificato la classe "vera" i come appartenente alla classe j [54].

Di seguito una possibile rappresentazione della matrice di confusione di una classificatore binario. La predizione viene fatta sulla classe positiva e sulla classe negativa.

- **Veri Positivi (TP)**: l'etichetta è positiva e anche la previsione è positiva
- **Veri Negativi (TN)**: l'etichetta è negativa e anche la previsione è negativa
- **Falsi Positivi (FP)**: l'etichetta è negativa ma la previsione è positiva
- **False Negativi (FN)**: l'etichetta è positiva ma la previsione è negativa

		Valori predetti		
		p′	**n′**	**totale**
Valori reali	**p**	Veri Positivi	Falsi Negativi	P
	n	Falsi Positivi	Veri Negativi	N
	totale	P′	N′	

Gli elementi della matrice applicata ai dati di addestramento selezionati in precedenza assume il seguente significato:

- **Veri Positivi (TP)**: pratica classificata come saldata, l'algoritmo suggerisce di concedere il prestito. Anche l'etichetta della pratica è saldata il che vuol dire la persona è riuscita a pagare il prestito. Da questi soldi l'istituto finanziario trae il guadagno tramite interessi prestabiliti.
- **Veri Negativi (TN)**: pratica classificata come contenzioso, l'algoritmo suggerisce di NON concere il prestito. Senza la previsione il prestito sarebbe stato concesso, quindi questi soldi andrebbero perso o recuperati solo in parte tramite i meccanismi di recupero credito.
- **Falsi Positivi (FP)**: pratica classificata come saldata, l'algoritmo suggerisce di concedere il prestito. Mentre l'etichetta della pratica è contenzioso. In realtà questo non è un errore significante visto che anche senza la predizione la pratica sarebbe andata in contenzioso.

- **False Negativi (FN)**: pratica classificata come contenzioso, l'algoritmo suggerisce di NON concere il prestito. Mentre l'etichetta della pratica è saldata. Questo è un errore rilevante, visto che l'algoritmo ha negato il finanziamento ad una persona che avrebbe saldato il prestito. Questo errore fa diminuire i guadagni e deve essere il più basso possibile.

Questi quattro numeri sono gli elementi costitutivi della maggior parte delle metriche di valutazione del classificatore. Un punto fondamentale quando si considera la valutazione del classificatore è che l'accuratezza pura (cioè la previsione è corretta o errata) non è generalmente una buona metrica. La ragione di ciò è dovuta al fatto che un set di dati potrebbe essere fortemente sbilanciato.

Ad esempio, se un modello è progettato per prevedere la frode da un set di dati in cui il 95% dei punti dati non è una frode e il 5% dei punti dati è una frode, un classificatore ingenuo che prevede non una frode, indipendentemente dall'input, essere accurato al 95%. Per questo motivo, le metriche come precisione e richiamo vengono generalmente utilizzate perché tengono conto del tipo di errore.

Di seguito la classe "salata" vine chiamata **positiva**, mentre la classe dei prestiti non restituiti ("contenzioso") viene denominata **negativa**.

Vista la natura del dataset dei prestiti finanziari, classificare erroneamente un positivo come negativo ha un costo elevato. Per qeusto l'errore sui falsi negativi è molto rilevante.

Definizione delle misure

L'obiettivo è costruire un modello capace di concedere i prestiti ai buoni pagatori e negarli ai cattivi pagatori, ovvero individuare una parte dei finanziamenti insoluti continuando a concedere tutti i prestiti saldati.

Una possibile misura per valutare un classificatore può essere l'accuratezza:

$$ACC = \frac{TP + TN}{P + N} \tag{5.1}$$

Visto che la classe dei positivi supera per il numero di record quella negativa l'accuratezza darebbe un valore poco significativo.

Con l'aiuto degli esperti dell'ufficio rischi si è scelto di utilizzare i seguenti parateti nella valutazione dei modelli [55]:

$$TPR = \frac{TP}{T} \tag{5.2}$$

Il rapporto misura la proporzione dei prestiti saldati classificati correttamente sul totale dei saldati. L'obiettivo è concedere i prestiti solo ai "buoni" pagatori, quindi si vuole avere il TP/P possibilmente vicino 1.

$$TNR = \frac{TN}{N} \tag{5.3}$$

Questo rapporto valuta la proporzione dei prestiti insoluti classificati correttamente sul totale dei insoluti. Lo scopo del algoritmo è di negare i prestiti ai "cattivi" pagatori, TN/N anche del 0.15 significa individuare il 15% dei cattivi pagatori ed evitare la perdita dei soldi.

I parametri TP/P e TN/N parametri rappresentano la precisione dell'algoritmo sulla classe positiva e sulla classe negativa. Essi devono essere valutati inseme visto l'obiettivo di concedere i prestiti ai buoni pagatori e negarli ai cattivi pagatori.

$$PERR = \frac{FP}{P} \tag{5.4}$$

Questo rapporto evidenzia il numero dei prestiti insoluti classificati come saldati. FP/P non è un errore significativo dell'algoritmo perché i falsi positivi sono i prestiti concessi che comunque sarebbero diventati insoluti anche in assenza della predizione.

$$NERR = \frac{FN}{N} \tag{5.5}$$

Il rapporto FN/N, che rappresenta i falsi negativi sul totale dei negativi. I falsi negativi rappresentano i prestiti negati dall'algoritmo che in realtà sarebbero stati saldati. Questo è un errore molto significativo visto che fa diminuire il numero dei finanziamenti concessi abbassando i ricavi dell'azienda. Quindi il rate dei falsi negativi deve essere più basso possibile.

5.3 Risultati

Di seguito saranno applicati i metodi di validazione come Holdout cross-validation e k-fold cross-validation ai quattro modelli selezionati.

5.3.1 Logistic regression

Viene valutato l'algoritmo Logistic regression variando i parameteri: *maxIter* che rappresenta il massimo numero di iterazioni. Totale delle classi usate nella fase di test: $positivi = 303048$, $negativi = 7166$.

Sampling della classe positiva 18%, $maxIter = 15$

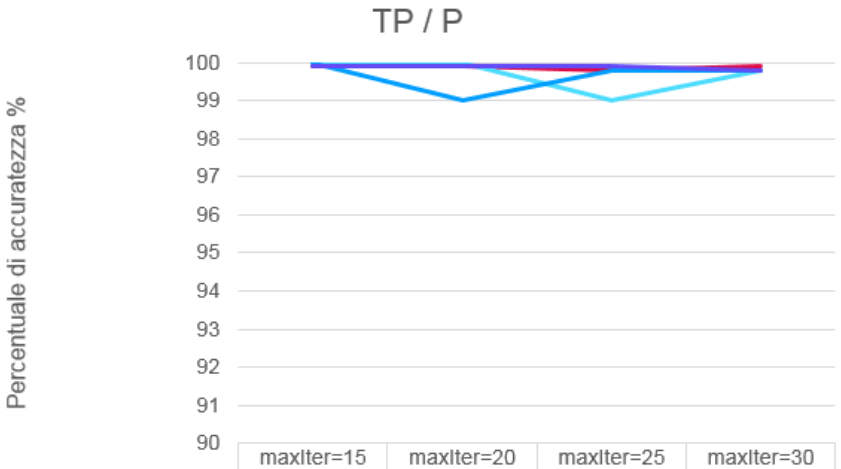
		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **302847**	Falsi Negativi **201**
	n	Falsi Positivi **6755**	Veri Negativi **411**

Sampling della classe positiva 14%, $maxIter = 25$

		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **302849**	Falsi Negativi **199**
	n	Falsi Positivi **6824**	Veri Negativi **342**

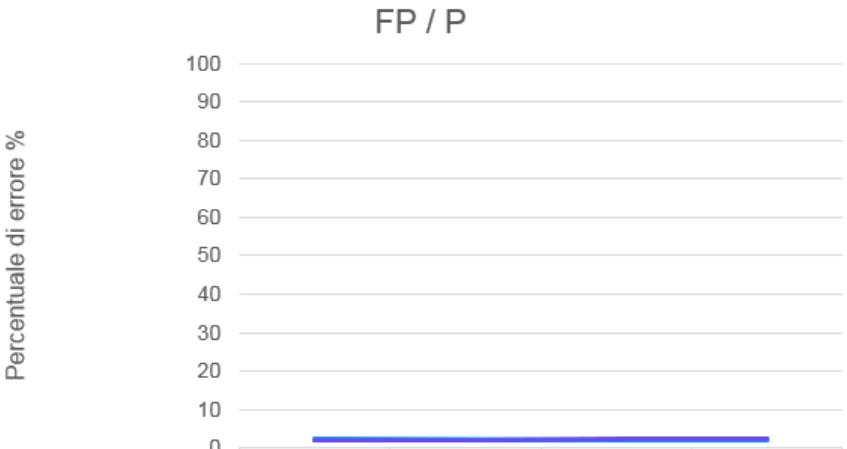
Di seguito viene rappresentato l'andamento della precisione dell'algoritmo prima nella classe positiva e poi in quella negativa. Sull'asse delle ordinate viene rappresentato il rapporto **TP/P** e **FP/P** in percentuale. Mentre sull'asse delle ascisse il numero di iterazioni utilizzati nell'addestramento.

Classe positiva, maxIter variabile



	maxIter=15	maxIter=20	maxIter=25	maxIter=30
sampling 18% SAL	99,9	99,9	99,8	99,9
sampling 8% SAL	100	100	99	99,8
sampling 10% SAL	100	99	99,8	99,8
sampling 14% SAL	99,9	99,9	99,9	99,8

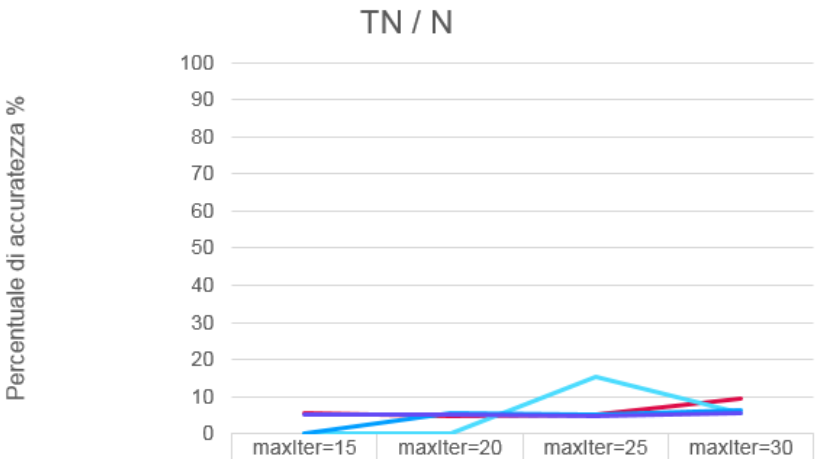
Figura 5.1: Veri positivi - Logistic regression



	maxIter=15	maxIter=20	maxIter=25	maxIter=30
sampling 18% SAL	2,21	2,22	2,23	2,28
sampling 8% SAL	2,36	2,36	2	2,22
sampling 10% SAL	2,36	2,23	2,23	2,21
sampling 14% SAL	2,22	2,23	2,25	2,26

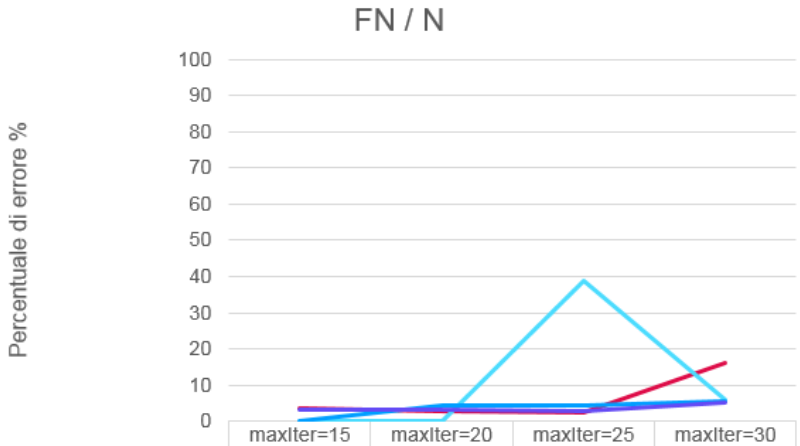
Figura 5.2: Falsi positivi - Logistic regression

Classe negativa, maxIter variabile



	maxIter=15	maxIter=20	maxIter=25	maxIter=30
sampling 18% SAL	5,73	4,7	5	9,61
sampling 8% SAL	0	0	15,16	5,7
sampling 10% SAL	0	5,48	5,31	6,18
sampling 14% SAL	5,3	5,17	4,77	5,44

Figura 5.3: Veri negativi - Logistic regression



	maxIter=15	maxIter=20	maxIter=25	maxIter=30
sampling 18% SAL	3,56	2,8	2,3	16,3
sampling 8% SAL	0	0	38,89	5,95
sampling 10% SAL	0	4,56	4,2	5,5
sampling 14% SAL	3,2	3,16	2,7	5,2

Figura 5.4: Falsi negativi - Logistic regression

Precedentemente sono stati creati quattro training set per consentire la valutazione della precisione del modello al variare della percentuale dei esempi positivi nel training set. Ciascun grafico riporta sotto una tabella con i relativi percentuali delle misure per training set utilizzato.

Valutazioni del modello Logistic regression

Nei grafici possimo osservare che il modello addestrato con il training set campionato a 8% classifica molti esempi come classe negativa. Questo aumenta il numero dei veri negativi ma allo stesso tempo fa crescere anche il numero dei falsi negativi. Questo comportamento fa capire che con *maxIter* pari a venticinque nel training set la porzione della classe positiva deve essere più alta del 8%.

Logistic regression ha una buona precisione nella classe positiva, il rapporto TP/P i positivi classificati correttamente sul totale di positivi e prossimo a 1. Mentre la precisione sulla classe negativa TN/N è 0.15 nella migliore configurazione. Inoltre non è possibile visualizzare il modello costruito in forma comprensibile.

Per questo modello non è stato possibile applicare la k-fold cross-validation essendo i tempi di esecuzione dell'algoritmo troppo lunghi.

5.3.2 Random Forest

Viene valutato l'algoritmo Random Forest utilizzando i parametri disponibili nella sua implementazione: *numTrees* che rappresenta il numero di alberi e *maxDepth* che rappresenta la massima profondità di ciascun albero.

Per valutare le prestazioni dell'algoritmo variando i suoi parametri per prima fissiamo il paramento *numTrees* variando *maxDepth*. Una volta trovato il *maxDepth* opportuno esso vine fissato e si ripete la valutazione variando questa volta *numTrees*.

Parametro *numTrees*

Di seguito il parametro *maxDepth* è fissato a 3, mentre il nunmero di alberi varia *numTrees* varia. Negli sperimenti vengono utilizzati 3 training set differenti con la classe positiva campionata a 8%,14% e 18%.

Totale delle classi usate nella fase di test: $positivi = 303048$, $negativi = 7166$. Seguono alcune delle configurazioni utilizzate.

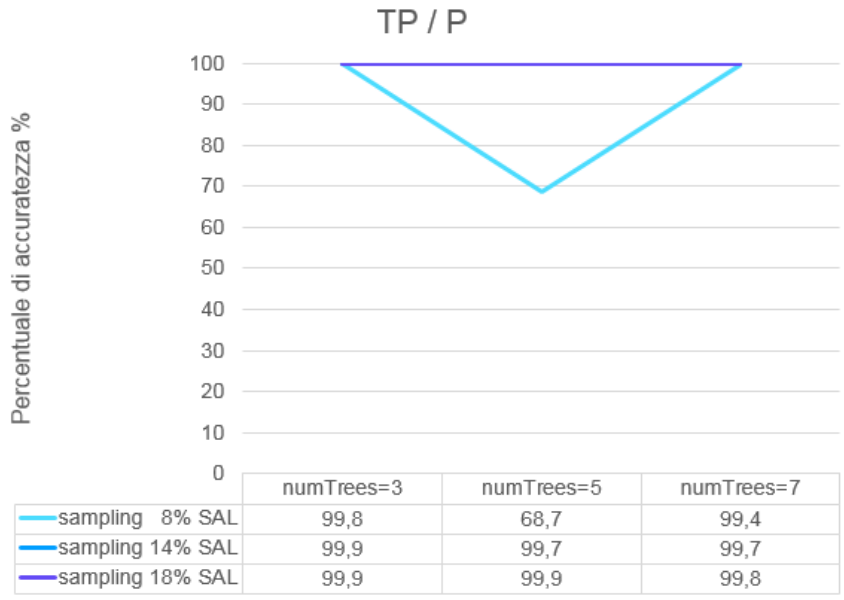
- Sampling della classe positiva 14%, $numTrees = 7$

		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **302286**	Falsi Negativi **762**
	n	Falsi Positivi **6227**	Veri Negativi **939**

Successivamente viene rappresentato l'andamento della precisione dell'algoritmo prima nella classe positiva e poi in quella negativa. Sull'asse delle ordinate viene rappresentato il rapporto **TP/P** in percentuale. Mentre sull'asse delle ascisse il numero di iterazioni utilizzati nell'addestramento.

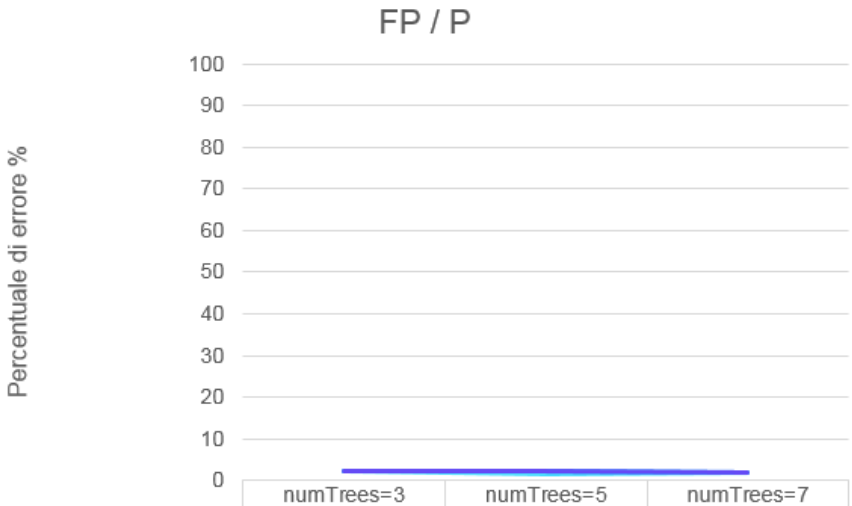
Nei grafici riportati sotto il parametro *maxDepth* è stato fissato a 3, mentre *numTrees* varia tra 3, 5 e 7. Possiamo notare dai risultati riportati sotto che in corrispondenza al valore 5 il modello dipende molto dai dati di addestramento, infatti l'errore del modello presenta i picchi in quei punti. Invece il valore *numTrees* = 7 sembra opportuno, offre una buona precisione nella classe positiva, e un errore basso nella classe negativa .

Classe Positiva , *numTrees* variabile



	numTrees=3	numTrees=5	numTrees=7
sampling 8% SAL	99,8	68,7	99,4
sampling 14% SAL	99,9	99,7	99,7
sampling 18% SAL	99,9	99,9	99,8

Figura 5.5: Veri positivi - Random Forest



	numTrees=3	numTrees=5	numTrees=7
sampling 8% SAL	2,3	1,5	1,9
sampling 14% SAL	2,3	2,1	2
sampling 18% SAL	2,3	2,1	2

Figura 5.6: Falsi positivi - Random Forest

Classe Negativa, *numTrees* variabile

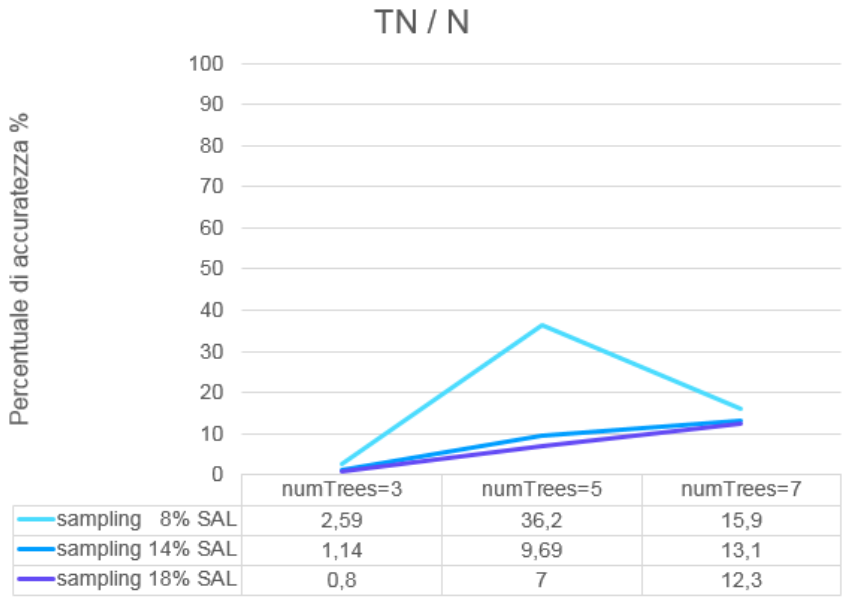


Figura 5.7: Veri negativi - Random Forest

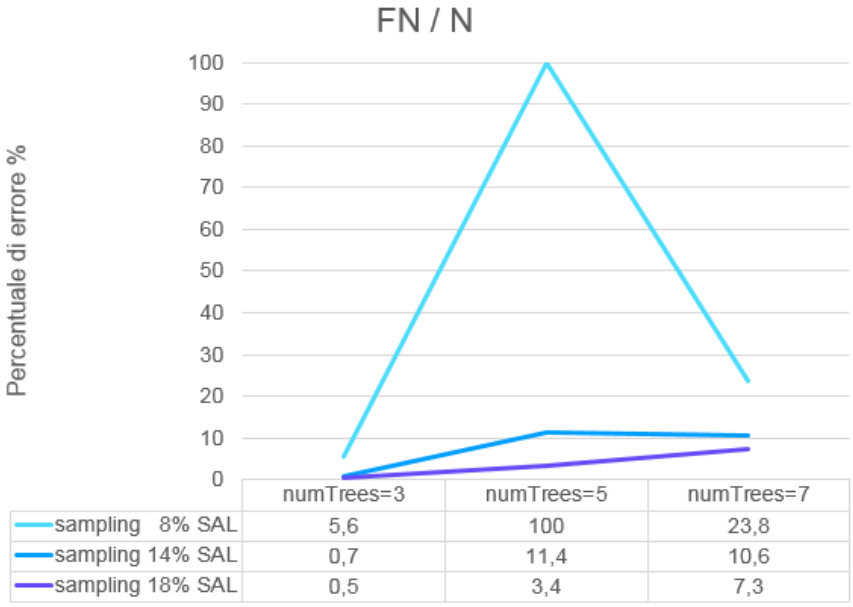


Figura 5.8: Falsi negativi - Random Forest

Parametro *maxDepth*

Dai risultati precedenti il parametro *numTrees* = 7 sembra opportuno. Quindi il passo successivo è andare a variare *maxDepth* tra 4,5 e 6. Negli sperimenti vengono utilizzati 3 training set differenti con la classe positiva campionata a 8%,14% e 18%.

Totale delle classi usate nella fase di test: *positivi* = 303048, *negativi* = 7166. Seguono alcune delle configurazioni utilizzate.

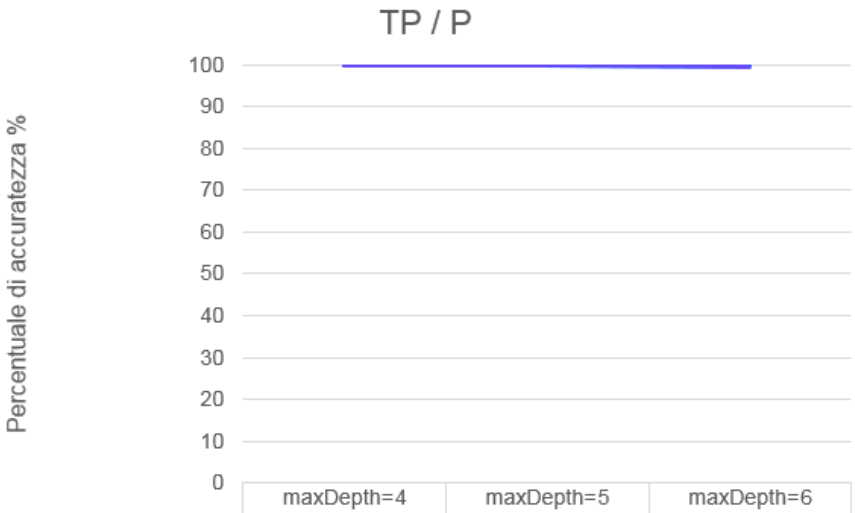
- Sampling della classe positiva 14%, *maxDepth*=6, *numTrees* = 7

		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **302661**	Falsi Negativi **387**
	n	Falsi Positivi **6271**	Veri Negativi **895**

- Sampling della classe positiva 14%, *maxDepth*=5, *numTrees* = 7

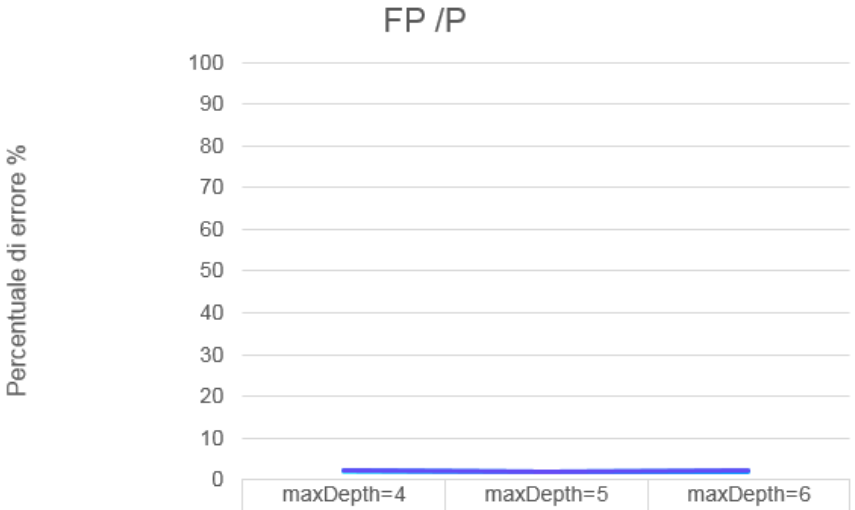
		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **302492**	Falsi Negativi **556**
	n	Falsi Positivi **6248**	Veri Negativi **918**

Classe Positiva, *maxDepth* variabile



	maxDepth=4	maxDepth=5	maxDepth=6
sampling 8% SAL	99,8	99,7	99,5
sampling 14% SAL	99,8	99,7	99,7
sampling 18% SAL	99,8	99,8	99,5

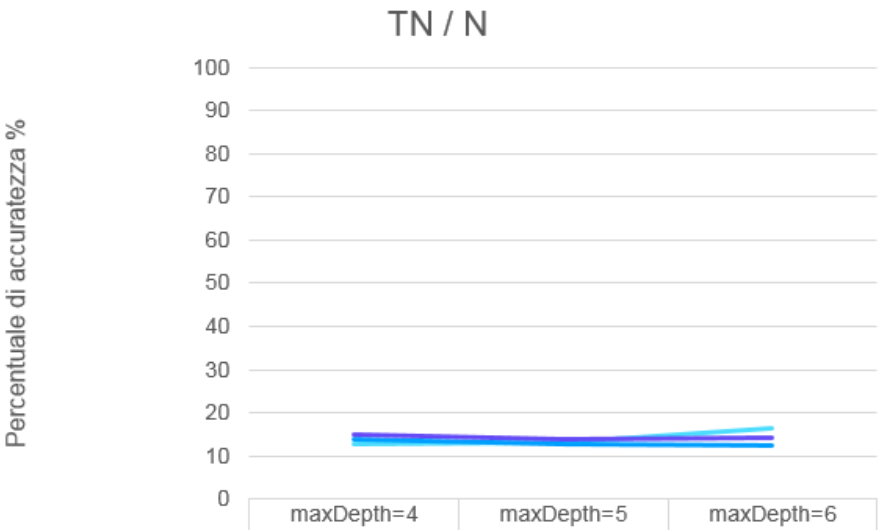
Figura 5.9: Veri positivi - Random Forest



	maxDepth=4	maxDepth=5	maxDepth=6
sampling 8% SAL	2	2	2,1
sampling 14% SAL	2,3	2	2
sampling 18% SAL	2,2	2	2,1

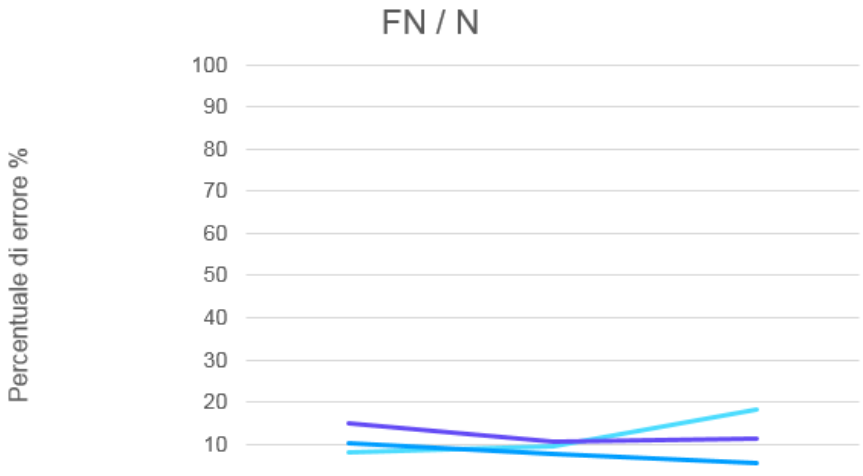
Figura 5.10: Falsi positivi - Random Forest

Classe Negativa, *maxDepth* variabile



	maxDepth=4	maxDepth=5	maxDepth=6
sampling 8% SAL	12,8	13	16,4
sampling 14% SAL	13,9	12,8	12,4
sampling 18% SAL	14,8	13,8	14,1

Figura 5.11: Veri negativi - Random Forest



	maxDepth=4	maxDepth=5	maxDepth=6
sampling 8% SAL	8,1	9,4	18,2
sampling 14% SAL	10,1	7,7	5,4
sampling 18% SAL	15,1	10,5	11,3

Figura 5.12: Falsi negativi - Random Forest

Valutando i grafici possiamo notare che l'aumento del parametro *maxDepth* fa migliorare la precisione del algoritmo sulla classe negativa.

Cross-Validation

Ora proviamo ad applicare la tecnica di validazione k-fold cross-validation. Sono stati utilizzati k=3 e k=10.

Qui di seguito vengono riportati le matrici di confusione del modello random forest nella configurazione *maxDepth=5*, *numTrees= 7*

- Sampling della classe positiva 14%, k=3

		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **301913**	Falsi Negativi **1135**
	n	Falsi Positivi **5774**	Veri Negativi **1392**

  Il true posite rate espresso come rapporto TP/P è uguale 0.99. Mentre il true negative rate rappresentato dal rapporto TN/N è uguale a 0.15 con l'errore FN/N pari a 0.16.

- Sampling della classe positiva 14%, k=10

		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **302293**	Falsi Negativi **755**
	n	Falsi Positivi **6145**	Veri Negativi **1021**

  Dalla seconda matrice di confusione, si può osservare che aumentando il k la precisione sulla classe negativa dell'algoritmo cresce leggermente, infatti per il k=10 il rapporto TN/N rimane invariato, mentre l'errore sulla classe negativa FN/N diminuisce diventando pari a 0.10

5.3.3 Decision tree

Di seguito viene valutato l'algoritmo Decision tree variando il parametero: *maxDepth* che rappresenta la massima profondità dell'albero. Negli sperimenti vengono utilizzati 3 training set differenti con la classe positiva campionata a 8%,14% e 18%. Viene utilizzato come misura di impurità dei nodi **GINI index**.

Totale delle classi usate nella fase di test: *positivi* = 303048, *negativi* = 7166. Seguono alcune delle configurazioni utilizzate.

- Sampling della classe positiva 14%, *maxDepth*=4

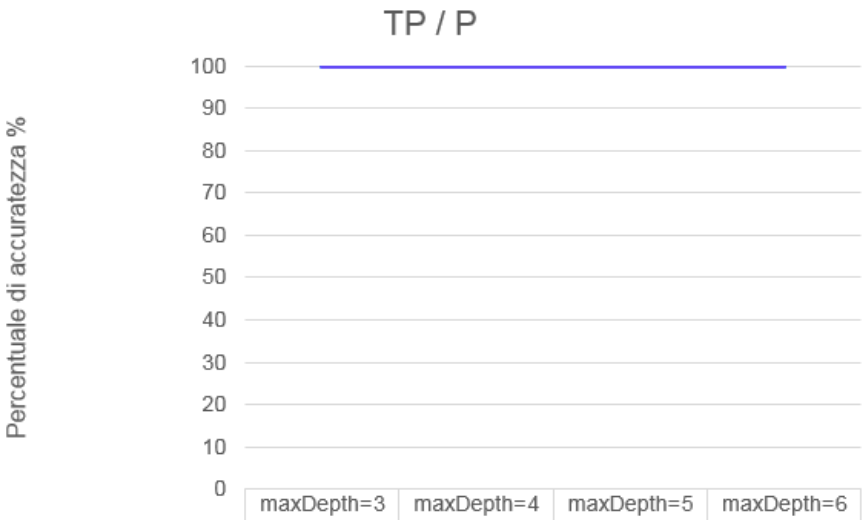
		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **302543**	Falsi Negativi **505**
	n	Falsi Positivi **6262**	Veri Negativi **904**

- Sampling della classe positiva 18%, *maxDepth*=3

		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **302793**	Falsi Negativi **255**
	n	Falsi Positivi **6342**	Veri Negativi **824**

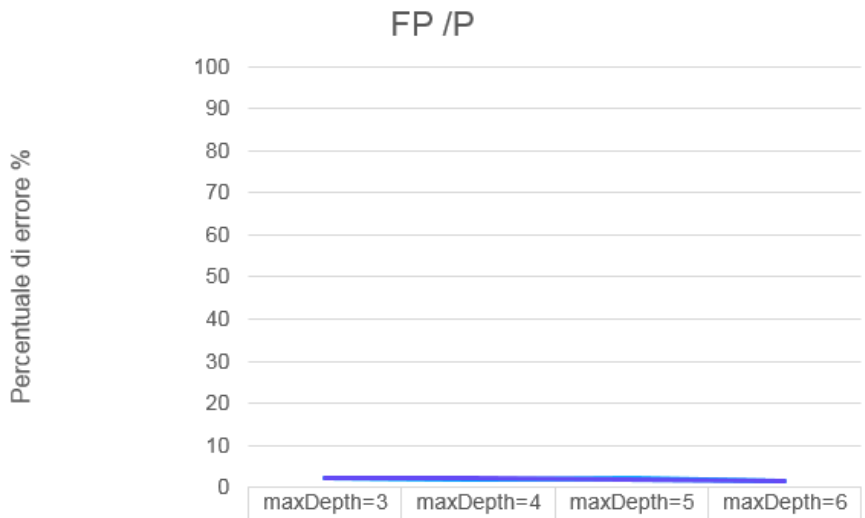
I grafici riportati sotto hanno la stessa leggenda di prima, il parametro variabile in questo caso è *maxDepth*. In Figura 5.8 si osserva che in corrispondenza al valore *maxDepth*=6 l'algoritmo ha un notevole errore sulla classe negativa.

Classe Positiva, *maxDepth* variabile



	maxDepth=3	maxDepth=4	maxDepth=5	maxDepth=6
sampling 8% SAL	99,9	99,7	99,7	99,7
sampling 14% SAL	99,9	99,8	99,8	99,8
sampling 18% SAL	99,9	99,9	99,8	99,8

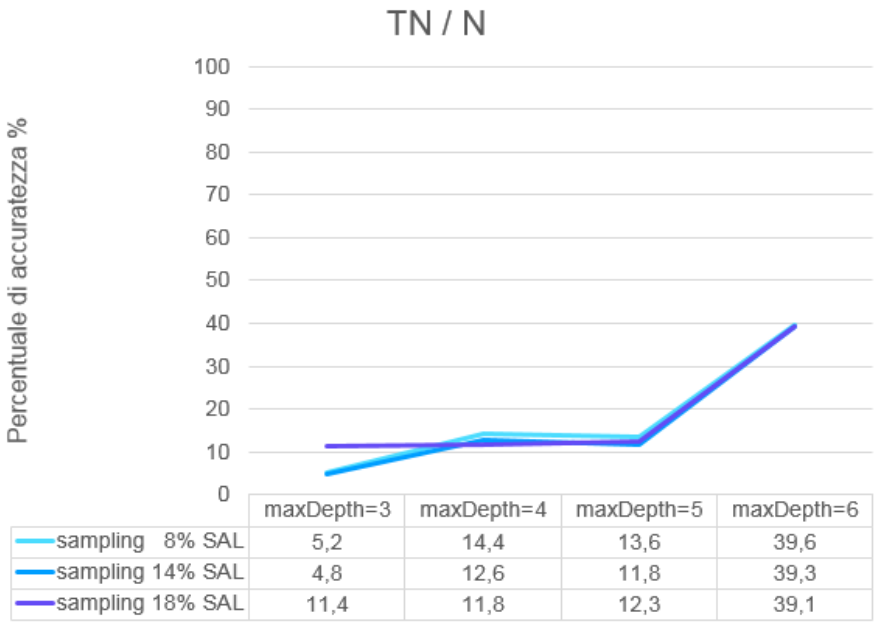
Figura 5.13: Veri positivi - Decision tree



	maxDepth=3	maxDepth=4	maxDepth=5	maxDepth=6
sampling 8% SAL	2,24	2,02	2,04	1,4
sampling 14% SAL	2,25	2,06	2,08	1,4
sampling 18% SAL	2,09	2,08	2,07	1,43

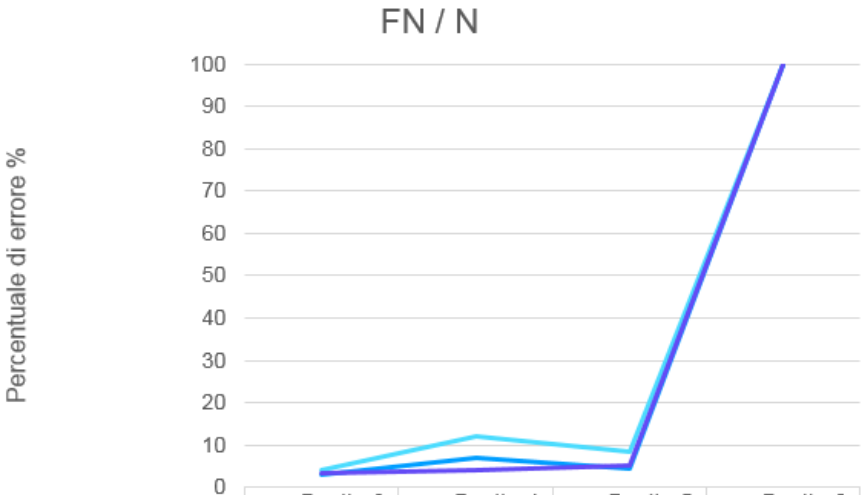
Figura 5.14: Falsi positivi - Decision tree

Classe Negativa, *maxDepth* variabile



	maxDepth=3	maxDepth=4	maxDepth=5	maxDepth=6
sampling 8% SAL	5,2	14,4	13,6	39,6
sampling 14% SAL	4,8	12,6	11,8	39,3
sampling 18% SAL	11,4	11,8	12,3	39,1

Figura 5.15: Veri negativi - Decision tree



	maxDepth=3	maxDepth=4	maxDepth=5	maxDepth=6
sampling 8% SAL	3,96	12,2	8,6	100
sampling 14% SAL	3,07	7,04	4,5	100
sampling 18% SAL	3,5	4,17	5	100

Figura 5.16: Falsi negativi - Decision tree

Dai andamenti dei grafici possiamo osservare che all'aumentare del parametro *maxDepth* l'algoritmo tende a classificare gli esempi come classe negativa. Osservando le Figure 5.15 e 5.16 risulta che il parametro più appropriato di *maxDepth* è 4. In quel punto l'algoritmo riesce a individuare i "cattivi pagatori" mantenendo basso l'errore.

Cross-Validation

Ora proviamo ad applicare la tecnica di validazione k-fold cross-validation. Per k=3 e k=10 l'algoritmo ha l'errore nella classe negativa pari al 80% e 100% rispettivamente. Questo metodo non da grandi vantaggi perché utilizza un training set abbastanza grande circa 800000 record. Questo è dovuto dal fatto che k-fold cross-validation crea un modello che generalizza male aderendo troppo ai dati di training.

Rappresentazione grafica dell'albero decisionale

L'albero decisionale a differenza di altri algoritmi esaminati offre la possibilità di visualizzare il modello. Infatti, alla radice dell'albero si trovano gli attributi più discriminanti che influenzano maggiormente la decisione. I nodi finale del albero, denominate foglie, rappresentano la decisione presa dal algoritmo.

Nella configurazione dell'albero decisionale con il parametro *maxDepth* = 4 possiamo notare che la feature IMP_SAL_CLT ha un forte potere discriminate. Esso rappresenta il saldo del cliente al momento del prestito. Questa informazione potrebbe servire al esperto che si occupa dei rischi nella valutazione.

La possibilità di interpretare il modello è molto importante in questo caso, gli attributi alla radice dell'albero possono essere valutata dal ufficio rischi per determinare i nuovi attributi che influenzano l'esito del finanziamento. Questo modello che utilizza decision tree è un buon candidato nella scelta di algoritmo di predizione.

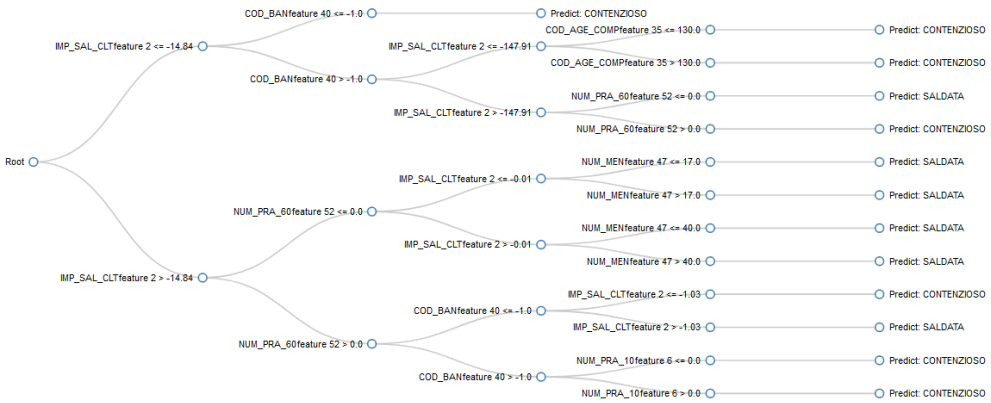


Figura 5.17: Albero decisionale con *maxDepth* = 4

5.3.4 Multilayer perceptron

Viene valutato l'algoritmo Multilayer perceptron variando la configurazione interna della rete neurale. In particolare verrà cambiato il numero dei nodi nei livelli intermedi, mentre il livello di input e di output rimane fisso.

Il numero di nodi in input corrisponde al numero delle features, in questo caso è uguale a 74 invece il numero di nodi di output è fissato a 2 che corrispondono alle due classi "saldata" e "contenzioso". Negli sperimenti vengono utilizzati 2 training set differenti con la classe positiva campionata a 14% e 18%.

Totale delle classi usate nella fase di test: $positivi = 303048$, $negativi = 7166$. Seguono alcune delle configurazioni utilizzate.

- Sampling della classe positiva 18%, $layers = [74,10,2,2]$

		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **261847**	Falsi Negativi **41201**
	n	Falsi Positivi **2756**	Veri Negativi **4410**

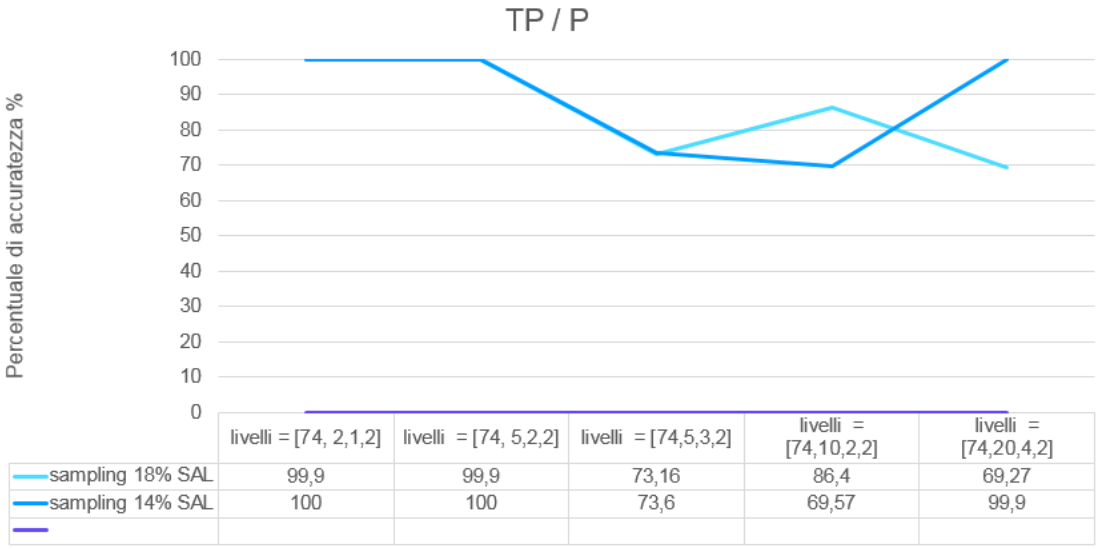
- Sampling della classe positiva 14%, $layers = [74,20,4,2]$

		Valori predetti	
		p′	**n′**
Valori reali	**p**	Veri Positivi **303034**	Falsi Negativi **14**
	n	Falsi Positivi **7163**	Veri Negativi **3**

I grafici riportati sotto hanno la stessa leggenda di prima, il parametro variabile in questo caso è la configurazione interna della rete. Il vettore "layers" è fatto in questo modo: per ogni posizione si ha il numero di nodi da utilizzare nel livello corrispondente.

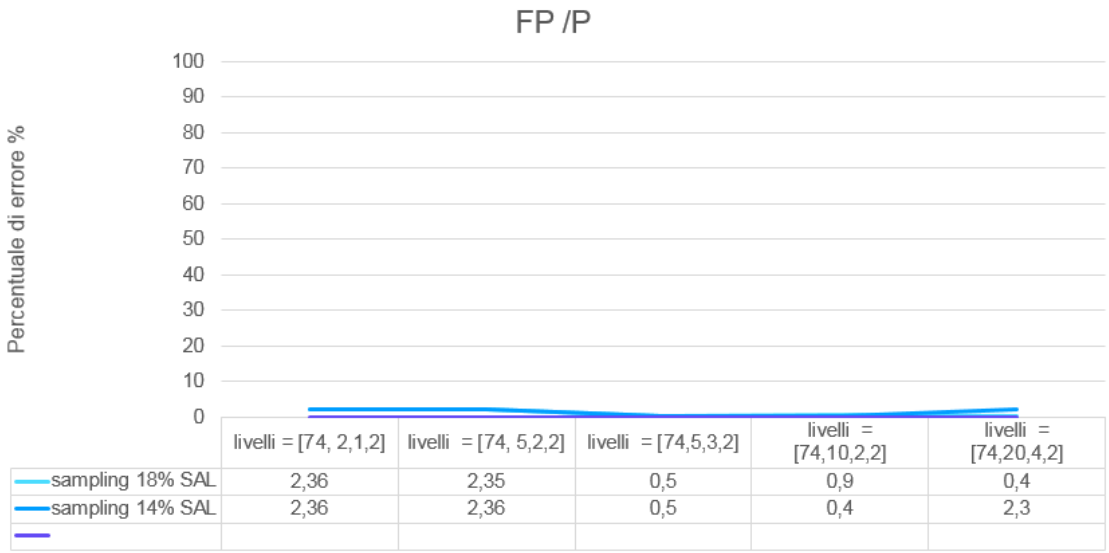
[input,livelloUno,livelloDue,output]

Classe Positiva, configurazione interna variabile



	livelli = [74, 2,1,2]	livelli = [74, 5,2,2]	livelli = [74,5,3,2]	livelli = [74,10,2,2]	livelli = [74,20,4,2]
sampling 18% SAL	99,9	99,9	73,16	86,4	69,27
sampling 14% SAL	100	100	73,6	69,57	99,9

Figura 5.18: Veri positivi - Multilayer perceptron



	livelli = [74, 2,1,2]	livelli = [74, 5,2,2]	livelli = [74,5,3,2]	livelli = [74,10,2,2]	livelli = [74,20,4,2]
sampling 18% SAL	2,36	2,35	0,5	0,9	0,4
sampling 14% SAL	2,36	2,36	0,5	0,4	2,3

Figura 5.19: Falsi positivi - Multilayer perceptron

La precisione della classe positiva decresce all'aumentare del numero di nodi nel primo strato intermedio.

Classe Negativa, configurazione interna variabile

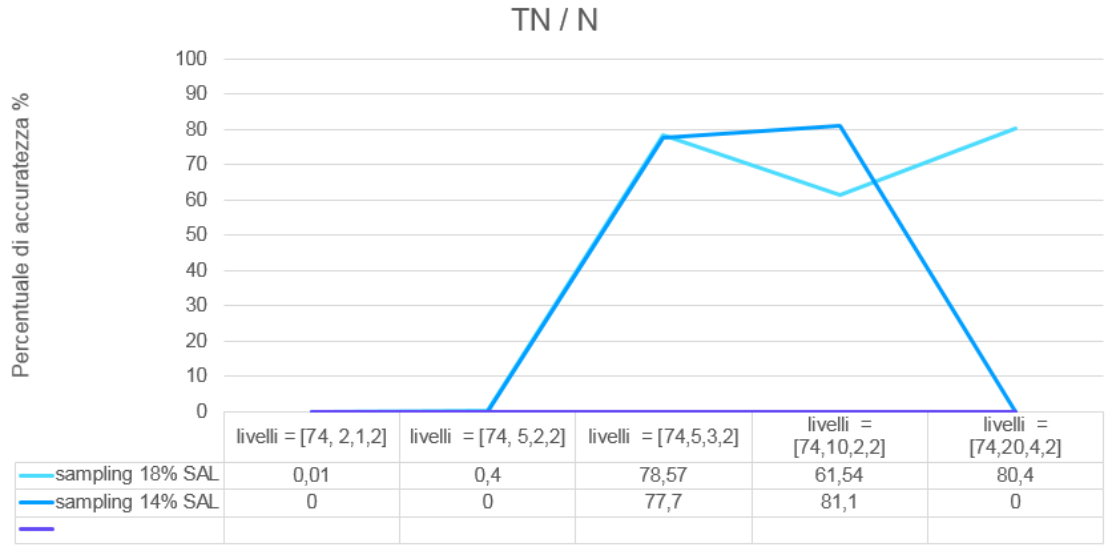


Figura 5.20: Veri negativi - Multilayer perceptron

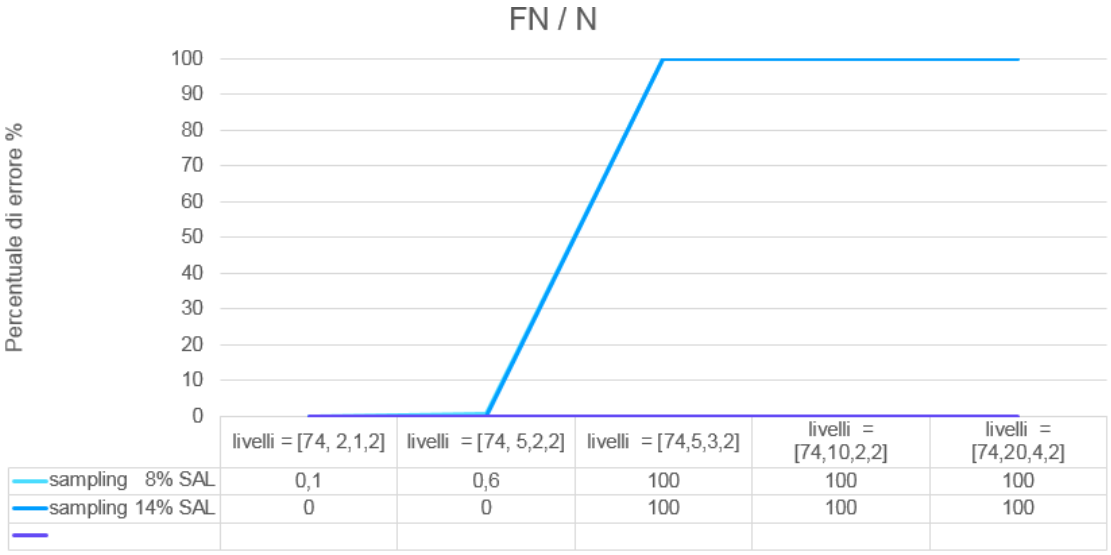


Figura 5.21: Falsi negativi - Multilayer perceptron

Questo modello presente notevole errore nella classe negativa che è molto importante nel contesto di valutazione di un finanziamento. Vuol dire che l'algoritmo Multilayer perceptron tende a negare i prestiti alle persone che l'avrebbero chiuso con successo e in questo modo fa perdere i guadagni all'azienda.

5.4 Selezione del modello

Riassumiamo in breve i quattro algoritmi utilizzati:

- **Logistic regression**: Questo algoritmo riesce ad avere il true positive rate pari a 99.8% individuare soltanto il 5% dei prestiti insoluti e non ha un modello interpretabile.
- **Random forest**: Random forest offre una precisione più alta nella classe negativa. Quindi riesce ad individuare i prestiti contenziosi. Il modello costruito presenta più alberi decisionali messi insieme che non è facile da interpretare al crescere del parametro *numTrees*.
- **Decision tree**: Questo algoritmo individua la quasi totalità dei finanziamenti saldati ed ha una buona precisione nella classe negativa individuando il 15% dei cattivi pagatori. Inoltre esso offre un modello interpretabile che può essere utilizzato dagli esperti per individuare i nuovi attributi che influenzano l'esito del prestito.
- **Multilayer Percepron**: Dai grafici si può osservare che la rete neurale neurale ha un comportamento piuttosto instabile sui dati utilizzati. Esso non é in grado costruire un modello capace di classificare correttamente la classe negativa.

Dagli esperimenti effettuati il modello che si adatti meglio alle esigenze risulta **decision tree**. Esso ha una precisione simile sulla classe positiva al random forest, si comporta in modo abbastanza lineare al variare dei parametri ed ha un modello interpretabile.

Capitolo 6

Applicazione del modello ad un dato di test

Una volta selezionato il modello esso deve essere ulteriormente valutato per evirate il fenomeno di *overfitting*. Questa valutazione avviene utilizzando il validation set introdotto nel capitolo precedente.

6.1 Validation set

Visto che i dati finanziari hanno una natura temporale, per simulare il funzionamento del sistema sviluppato in un contesto reale dobbiamo ripetere l'addestramento e test a distanza di un mese.

In pratica il traing è stato effettuato con i prestiti chiusi entro il 2013-12-31, quindi a questi dobbiamo aggiungere tutte le pratiche chiuse nel mese successivo entro il 2014-01-31. Anche la finestra temporale di test set deve essere spostata a desta di un mese. I dati del test saranno tutte le pratiche aperte dal 2014-01-31 al 2014-07-31. Denominiamo questo nuovo insieme di dati *validation set*.

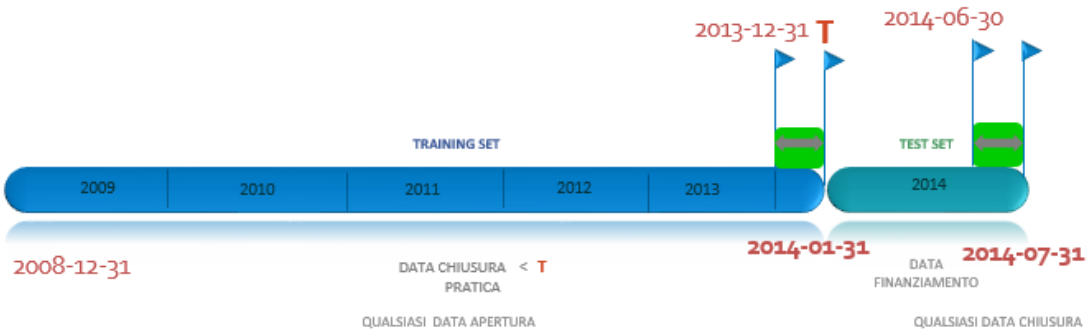


Figura 6.1: Training e test set a trascorrere di un mese

Di seguito viene confrontata la precisione dell'algoritmo **decision tree** tra training e validation set per entrambi le classi.

Classe Positiva, *maxDepth* variabile

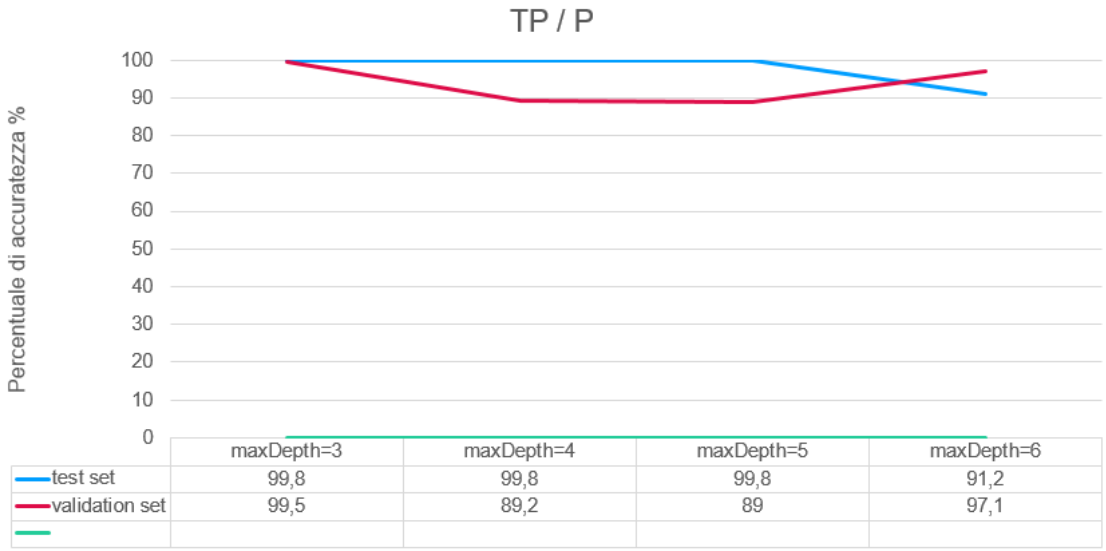


Figura 6.2: Veri positivi - Decision tree

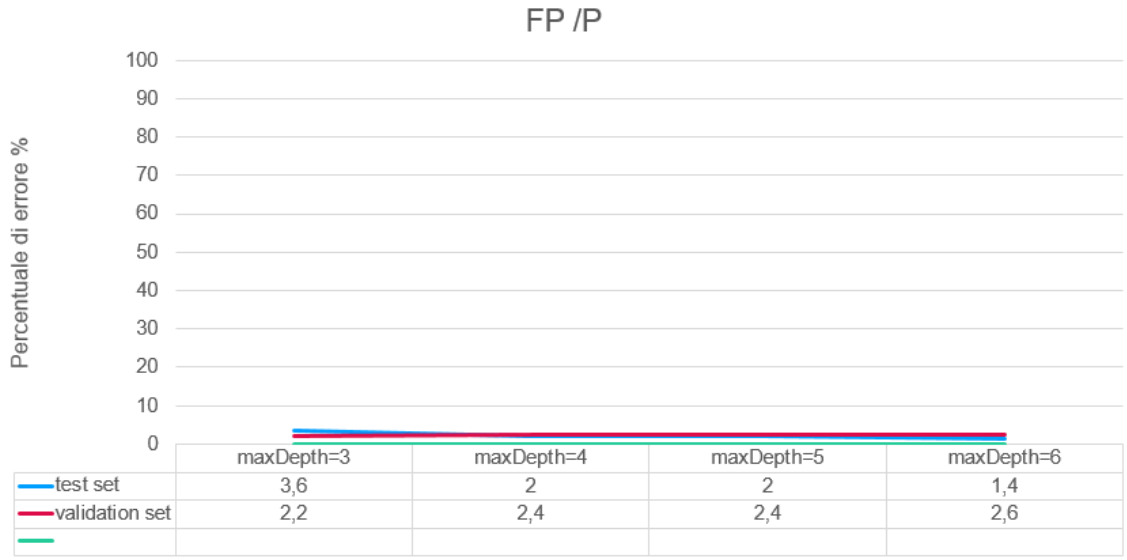


Figura 6.3: Falsi positivi - Decision treen

Classe Negativa, *maxDepth* variabile

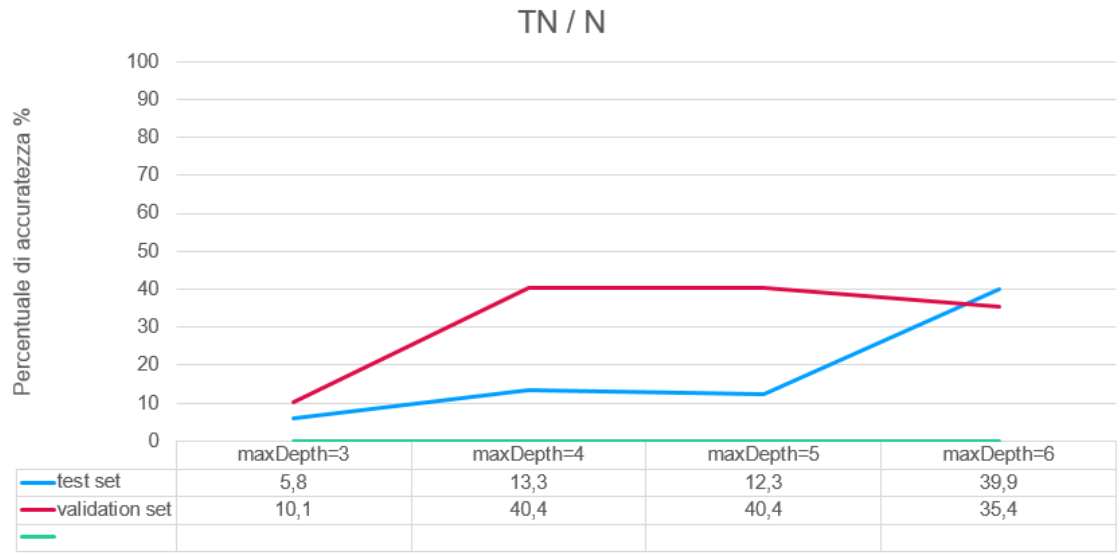


Figura 6.4: Veri negati - Decision tree

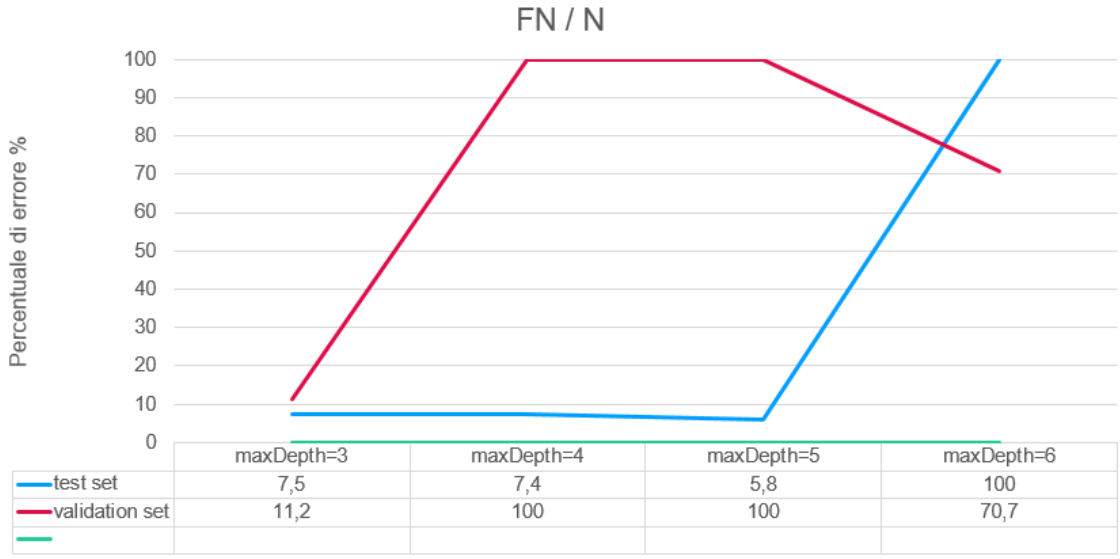


Figura 6.5: Falsi negativi - Decision treen

Dai grafici possiamo osservare il fenomeno di overfitting all'aumentare della profondità dell'albero. Infatti la linea rossa che rappresenta l'errore dell'algoritmo nel validation set cresce per valori di *maxDepth* uguali a 4 e 5. Questo accade perché l'algoritmo aderisce troppo ai dati in fase di addestramento, infatti nella fase di test

quando incontra gli esempi provenienti da una distribuzione diversa non riesce a classificarli correttamente.

Capitolo 7

Futuri Sviluppi

Per migliorare ulteriormente la precisione del modello e rendere l'addestramento più automatico si potrebbe fare dei futuri sviluppi su due aspetti:

- **Finestra Temporale**: Si potrebbe fare degli esperimenti variando la finestra temporale nel traing set prendendo un intervallo più grande di 5 anni considerati in questo lavoro. Ciò comporta un training set di notevole dimensione.
- **Continuous Learning System**: IBM Watson Studio mette a disposizione la possibilità di implementare un sistema capace di riaddestrasi in modo automatico [56].

7.1 Finestra Temporale

Dai risultati descritti in capitolo precedente si nota che la finestra temporale del **traning** e **test** set, cioè il range della data della chiusura della pratica influenza molto la precisione del modello. Questo potrebbe essere dovuto alle tendenze degli utenti di restituire o meno il prestito in un determinato periodo temporale: una crisi economica o viceversa un aumento del benessere.

Ampliando la finestra temporale in **trainig** e **test** comporta di avere il data set con molto più dati. Questo influenzerà le performance in fase di addestramento.

Figura 7.1: Ampliare il range temporale in data set

7.2 Continuous Learning System

Visto che la consegna in produzione del modello addestrato si ripete più volte, è possibile utilizzare IBM Watson Studio per riaddestrare il modello con nuovi dati. Per fare questo, si utilizza il sistema di **apprendimento continuo** IBM Watson Machine Learning, che fornisce il monitoraggio automatizzato delle prestazioni del modello, il riaddestramento e la ridistribuzione per garantire la qualità della previsione.

7.2.1 Configurazione

Per configurare un sistema di apprendimento continuo in IBM Watson Studio occorre:

1. Preparare il *data set di feedback*. *Data set di feedback* consiste in una tabella di training data e una tabella di dati di test creati e caricati su Db2 Warehouse (IBM Cloud).

2. Creare una pipeline e addestrare un modello in python MLlib. In questa fase per l'addestramento vengono utilizzati i dati differenti da *data set di feedback*.

3. Salvare il modello su IBM Watson Studio cloud. Per salvare il modello si può utilizzare **watsonMachineLearningClient** importato dalla libreria WatsonMachineLearningAPIClient.

4. Configurare il sistema di continuos learning. Si utilizzano le REST API messe a disposizione da IBM Watson [57].

```
system_config = {
    client.learning_system.ConfigurationMetaNames.FEEDBACK_DATA_REFERENCE: feedback_data_reference,
    client.learning_system.ConfigurationMetaNames.MIN_FEEDBACK_DATA_SIZE: 10,
    client.learning_system.ConfigurationMetaNames.SPARK_REFERENCE: spark_credentials,
    client.learning_system.ConfigurationMetaNames.AUTO_RETRAIN: "conditionally",
    client.learning_system.ConfigurationMetaNames.AUTO_REDEPLOY: "always"
}
```

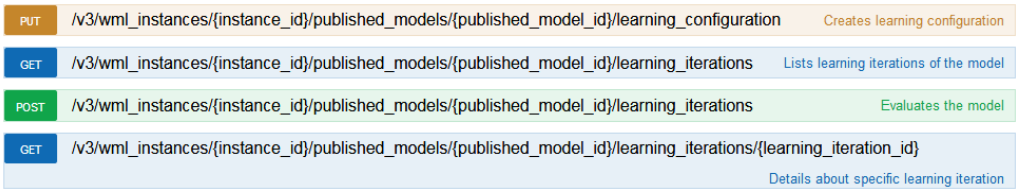


Figura 7.2: REST API IBM Watson

Parametri utilizzati:

- min_feedback_data_size - questo è il numero minimo di record nel set di dati di feedback per avviare l'iterazione del sistema di apprendimento continuo
- auto_retrain [never, always, conditionally] - questo parametro specifica se il processo di riaddestramento deve essere attivato ("condizionatamente" attiverà il processo di riaddestramento quando il risultato della valutazione è al di sotto del valore di soglia specificato)
- auto_redeploy [never, always, conditionally] - questo parametro specifica se il modello di riaddestramento deve essere distribuito ("condizionatamente" attiverà la ridistribuzione quando la qualità del modello di nuova formazione è migliore)

5. Caricare nuovi record nel *feedback data set*. Per poter fare un nuovo addestramento del modello è necessario inserire dei nuovi record nelle tabelle di feedback creati precedentemente.

6. Valutare il modello. Una volta configurato il sistema di addestramento continuo è possibile far partire un addestramento del modello con i dati caricati in *feedback data set*. Se la precisione del nuovo modello supera la soglia specificata nella configurazione allora il modello viene salvato e reso disponibile per la consegna in produzione.

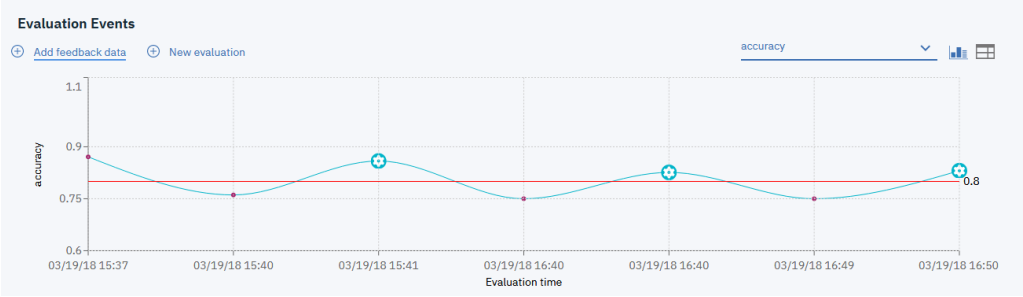


Figura 7.3: Valutazione del modello

Capitolo 8

Conclusioni

Questo lavoro di tesi ha permesso la realizzazione di una architettura software contenente più soluzioni per la predizione dell'esito di un finanziamento. Essa è stata sviluppata in Python su ambiente Spark utilizzando principalmente la libreria Spark MLlib.

Il sistema prende in input un insieme di pratiche dei prestiti finanziari e restituisce una etichetta che indica se il prestito in questione sarà saldato o meno.

Come visto nel capitolo due la soluzione è basata su varie blocchi sofware ognuno con una specifica funzione: estrazione, preparazione dei dati e addestramento del modello. Le principali difficoltà sono state riscontrate nella fase di estrazione dei dati e di preparazione (data processing). Dal lavoro svolto possiamo dedurre che la fase di preparazione dei dati determina maggiormente la futura precisione del modello costruito.

Nel capitolo 5 è stata confrontata la precisione nella classe positiva e negativa di 4 algoritmi al variare dei parametri specifici di ciascuno di essi. Tenendo conto della natura cost-sensitive del problema, dagli esperimenti risulta che l'algoritmo **decision tree** offre la migliore precisione in termini del richiamo sulla classe positiva. Inoltre il modello costruito decision tree può essere *interpretato*.

Nella configurazione con 3 livelli di profondità l'algoritmo ha 99.9% di precisione della classe "saldata", questo vuol dire riesce a prevedere correttamente quasi sempre l'esito dei finanziamenti che finiscono saldati.

Come mostra l'esperimento sul validation set, questa configurazione dell'algoritmo generalizza bene riuscendo ad individuare i record non ancora incontrati.

Il sistema implementato sarà inserito come supporto negli uffici rischio per fornire un aiuto agli utenti che devono valutare il rischio nel concedere un prestito finanziario. Il nuovo sistema grazie all'apprendimento automatico impara direttamente dai dati offre il modello più flessibile e dinamico rispetto a quello attualmente utilizzato negli uffici rischio trattati in questo lavoro.

Il sistema consente agli istituti finanziari, grazie all'**interpretabilità** del modello basato sul decision tree, di individuare i nuovi attributi che influenzano l'esito di un finanziamento.

Bibliografia

[1] World Bank Group, il "Bank nonperforming loans to total gross loans (%)", https://data.worldbank.org/indicator/FB.AST.NPER.ZS?locations=IT

[2] CRIF - Centrale Rischi Finanziari S.p.A. è una azienda italiana che fornisce aiuto nell'erogazione e nella gestione del credito al consumo, essa offre le informazioni di referenza creditizia per prevenire e controllare i rischi finanziari. https://www.crif.it/

[3] Definizione Consumer credit, https://www.britannica.com/topic/consumer-credit

[4] China's New Tool for Social Control: A Credit Rating for Everything https://www.wsj.com/articles/chinas-new-tool-for-social-control-a-credit-rating-for-everything-1480351590

[5] Understanding big data: Analytics for enterprise class hadoop and streaming data, P. Zikopoulos, C. Eaton - 2011

[6] IBM InfoSphere DataStage https://www.ibm.com/us-en/marketplace/datastage

[7] https://en.wikipedia.org/wiki/KNIME
KNIME Analytics Platform Article

[8] W. H. Inmon, Building the data warehouse, 1992

[9] data warehousing, data mining, olap and oltp technologies are essential elements to support decision-making process in industries - G.Satyanarayana Reddy, Rallabandi Srinivasu, M. Poorna, Chander Rao, Srikanth Reddy Rikkula

[10] An overview of data warehousing and OLAP technology. S. Chaudhuri, U. Dayal

[11] IBM Watson Data Platform - Watson is IBM's suite of enterprise-ready AI services, applications and tooling.

[12] Spark 2.1 - Apache Spark is a fast and general-purpose cluster computing system.

[13] Matei Zaharia, Spark: In-Memory Cluster Computing for Iterative and Interactive Applications, https://www.youtube.com/watch?v=qLvLg-sqxKc

[14] Eugene Kleinberg, On the Algorithmic Implementation of Stochastic Discrimination.

[15] Cost-Sensitive Classification: Algorithms and Advances, Hsuan-Tien Lin

[16] Berry, Michael J.A (1997). Data Mining Techniques For Marketing, Sales and Customer Support.
[17] Russell, Stuart; Norvig, Peter (2003) [2010]. Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall
[18] www.ce.unipr.it
[19] http://www.sisef.it/forest@/papers//no07/Scrinzi_349@image009.gif
[20] Machine learning - SAS https://www.sas.com/it_it/insights/analytics/machine-learning.html
[21] Alan Turing, COMPUTING MACHINERY AND INTELLIGENCE, in MIND, vol. 59, nº 236, ottobre 1950
[22] https://support.google.com/a/answer/2368132?hl=it
[23] Bishop, Christopher M. (2006), "Pattern Recognition and Machine Learning"
[24] Czech, Tomasz. "Deep learning: the next frontier for money laundering detection" https://www.globalbankingandfinance.com/deep-learning-the-next-frontier-for-money-laundering-detection/
[25] David B. Searls, "Using bioinformatics in gene and drug discovery" https://pdfs.semanticscholar.org/2bcc/e180aff078e53f442d986c2d4b96adad293a.pdf
[26] Isabella Chiari, Introduzione alla linguistica computazionale, Bari, Laterza, 2007
[27] Siri - assistente digitale, Apple Inc. https://www.apple.com/it/ios/siri/
[28] Alexa Voice Service https://developer.amazon.com/it/alexa-voice-service/
[29] https://www.knime.com/ KNIME Analytics Platform
[30] https://www.cs.waikato.ac.nz/ml/weka/
[31] http://greenstechnologysoftwares.blogspot.com/2016/01/best-datastage-training-institute-in.html
[32] http://www.uet.edu.pk/Conferences/icosst2009/presentations_2009/OSSW_Presentations/4_WEKA_x_KNIME
[33] https://nodepit.com/category/ KNIME Analytics Platform Nodes
[34] https://archive.ics.uci.edu/ml/index.html
[35] Piotr Juszczak (2006), "Learning to recognise"
[36] J. B. MacQueen (1967): Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press.
[37] Machine learning and pattern recognition "can be viewed as two facets of the same field." http://www.iascgroup.it/en/machine-learning-en.html
[38] Lance, P. & Hattori, A. (2016). Sampling and Evaluation
[39] Applied logistic regression. DW Hosmer Jr, S Lemeshow, RX Sturdivant
[40] https://spark.apache.org/docs/2.1.0/mllib-linear-methods.html Logistic regression
[41] IBM Watson logo https://houseofgeek.in/ibm-watson-taking-world/
[42] http://spark.apache.org/

[43] Gini, Corrado (1912). Variabilità e mutabilità
[44] F. Aiolli (2008) - Sistemi Informativi.
[45] J.R. QUINLAN 2007, Induction of Decision Trees.
[46] https://spark.apache.org/docs/2.1.0/mllib-decision-tree.html Decision tree classifier Spark MLlib
[47] Random forests, L. Breiman - Machine learning, 2001
[48] https://spark.apache.org/docs/2.1.0/mllib-ensembles.html Random forest classifier
[49] https://spark.apache.org/docs/2.1.0/ml-classification-regression.htmlmultilayer-perceptron-classifier
[50] Rectifier nonlinearities improve neural network acoustic models AL Maas, AY Hannun, AY Ng
[51] Holden Karau, Andy Konwinski, Patrick Wendell & Matei Zaharia, in "Learning Spark"
[52] MLlib Libreria di Machine Learning di Spark.
[53] https://spark.apache.org/docs/2.1.0/api/python/pyspark.mllib.html Logistic regression Spark MLlib
[54] Stehman, Stephen V. (1997). Selecting and interpreting measures of thematic classification accuracy.
[55] The relationship between Precision-Recall and ROC curves J Davis, M Goadrich
[56] https://dataplatform.cloud.ibm.com/docs/content/analyze-data/ml-continuous-learning.html
[57] http://watson-ml-api.mybluemix.net/