UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI ECONOMIA "GIORGIO FUÀ"

Corso di Laurea Magistrale in Data Science per l'Economia e le Imprese

"Tecniche di Machine Learning applicate alla previsione delle crisi finanziarie "

"Application of Machine Learning techniques to predict financial crises "

Relatore:
Prof. Domenico Potena
Correlatrice:
Prof.ssa Claudia Pigini

Tesi di Laurea di:
Damian Agachi Menna

Anno Accademico 2022 – 2023

INDICE

Introduzione

Le crisi bancarie[1] del 2007-2008 hanno avuto un impatto significativo sull'economia interna dei paesi inizialmente coinvolti, per poi diffondersi a livello globale. La loro identificazione tempestiva è fondamentale per preservare la stabilità finanziaria e mitigare il rischio sistemico. Le turbolenze del 2023 nel settore bancario con il fallimento negli Stati Uniti delle banche SVB e SBNY, in Svizzera della Credit Suisse hanno riacceso i riflettori sul settore cercando di interpretare se tali avvenimenti sono l'inizio di una possibile crisi oppure sono semplicemente degli eventi isolati. Disporre di sistemi di Early Warning Systems (EWS) efficienti risultano essere strumenti cruciali per monitorare la salute del sistema finanziario di un Paese e per prevedere potenziali crisi. Tuttavia, la previsione delle crisi bancarie rimane una sfida complessa. Gli approcci tradizionali spesso non riescono a catturare la complessità delle interazioni nel sistema finanziario globale, mentre la scarsa rappresentazione delle situazioni critiche compromette l'efficacia dei modelli predittivi.

I modelli predittivi tradizionalmente più diffusi si basano su approcci statistici quali probit e logit. Queste metodologie costituiscono due funzioni di regressione utilizzate per stimare la probabilità di verificarsi di un evento, rappresentato da una variabile dipendente binaria, in relazione ad un insieme di variabili esplicative indipendenti. Tuttavia, tali metodologie presentano sia punti di forza, come la facilità di interpretazione dei risultati e le solide basi statistiche, sia delle limitazioni che possono influenzare negativamente le loro prestazioni nella previsione delle crisi bancarie. Negli ultimi anni, vi è stato un crescente utilizzo di tecniche di Machine Learning, le quali non sono vincolate dalle assunzioni tipiche dei modelli econometrici, quali la linearità e la rilevanza delle variabili del fenomeno. Tale tendenza ha contribuito a rendere più efficienti le previsioni, in quanto le tecniche di Machine Learning possono adattarsi meglio a dati complessi e catturando gli

[1] Crisi bancaria (sistemica) si verifica quando molte banche si trovano contemporaneamente in seri problemi di solvibilità o liquidità, o perché sono colpite tutte dallo stesso shock esterno o perché il fallimento di una banca o di un gruppo di banche si diffonde ad altre banche nel sistema. (The World Bank, 2016)

effetti non lineari, migliorano così la capacità di anticipare eventi critici nel contesto bancario. Come si evidenzia in diversi articoli (Casabianca et al., 2019) e (P. Holopainene e P.Sarlin, 2017), i modelli di Machine Learning tendono a performare meglio dei modelli logit/ probit *in sample* in termini di F-measure. *Out of sample* invece, come dimostrato da (J. Beutel et al., 2019), i modelli logit tendono a generalizzare meglio rispetto ai precedenti, mettendo in evidenza una caratteristica tipica delle tecniche di apprendimento automatico che è l'*overfitting*[2]. In aggiunta, alcune tecniche sono delle vere e proprie "black box" di difficile interpretazione, oltre il fatto che necessitano di un numero elevato di dati affinché possano individuare i pattern del fenomeno osservato.

I dataset che descrivono eventi rari sono fortemente sbilanciati tra la situazione che normalmente accade e l'eccezionalità. Questo fattore distintivo dei dati porta ad avere la problematica in gergo chiamata *Class Imbalance*[3]. Questa rappresenta una sfida significativa, poiché le crisi bancarie sistemiche sono relativamente rare rispetto ai periodi di stabilità finanziaria, hanno un alto impatto economico ma sono difficili da prevedere, in virtù del fatto che numericamente sono poche e gli algoritmi fanno fatica a riconoscerli. Anzi, questo tratto distintivo favorisce la predizione della classe maggioritaria rispetto a quella più rara, che invece spesso e volentieri, data la sua eccezionalità, rappresenta il target del modello di apprendimento.

Per affrontare la scarsa rappresentazione delle situazioni critiche, in questo lavoro di tesi si propone una combinazione di tecniche volte a mitigare la scarsità di eventi della classe minoritaria. In particolare, per ridurre lo sbilanciamento tra le classi sono state sperimentate tecniche di *sampling bilanciato* e *cost sensitivity* nella fase di training del modello e poi confrontate fra di loro. Utilizzando tecniche di sampling bilanciate e l'integrazione della cost sensitivity, miriamo a migliorare l'accuratezza e l'affidabilità dei modelli di previsione. La combinazione di undersampling

[2] Si verifica quando un modello diventa troppo specializzato nell'apprendere i dettagli e il rumore dei dati di addestramento, con conseguente scarsa capacità di generalizzazione su nuovi dati non visti in precedenza.
[3] Situazione che si verifica quando il numero di campioni delle diverse classi di una variabile categorica è significativamente diverso

con k-Means e l'utilizzo di Metacost, permettono di focalizzare l'attenzione dei modelli sulla classe più rara in modo da creare un sistema di Early Warning System più efficace nel prevedere le crisi bancarie sistemiche a livello di Paese, rispetto all'uso dei dati sbilanciati. Seppur tecniche già note in letteratura, non vi sono casi del loro impiego per realizzare sistemi di EWS per la predizione di crisi bancarie sistemiche, facilitando la stabilità finanziaria e scongiurando i potenziali impatti negativi sull'economia globale. Inoltre, il problema è stato affrontato con le reti neurali di tipo Long Short-Term Memory (LSTM) che sono uno strumento molto più complesso ma in grado di catturare dipendenze a lungo termine nei dati. Il che è essenziale per rilevare segnali di crisi bancarie che possono svilupparsi nel corso di mesi o anni. Questo è uno dei vantaggi rispetto alle tecniche tradizionali di machine learning perché non tutte possono non essere in grado di gestire dipendenze a lungo termine.

Il testo è così organizzato: nel Capitolo 1 vi si trova una panoramica sulla letteratura degli EWS; nel secondo viene descritto il dataset e la metodologia utilizzata per svolgere gli esperimenti; nel terzo sono riportati i risultati; nel quarto vi sono le conclusioni.

Capitolo 1

Letteratura sugli Early Warning System

Gli Early Warning Systems sono strumenti proattivi che aiutano a identificare in anticipo i segnali di potenziali problemi nel sistema bancario, consentendo alle autorità di adottare misure preventive e di gestione del rischio (Guerra e Castelli, 2021). EWS economici possono avere diverse declinazioni, tra cui monitoraggio di crisi finanziarie, recessioni, instabilità bancarie, mercati immobiliari, debito pubblico, fluttuazioni valutarie e mercati delle materie prime. Cambia la tematica che mirano a monitorare, ma la mansione rimane quella di identificare segnali precoci di potenziali rischi e problemi nell'economia.

Prevenire tempestivamente o limitare gli effetti negativi delle crisi bancarie può aiutare le istituzioni ad evitare la diffusione su larga scala, la difficoltà di una singola banca o di un singolo Paese, evitando crisi globali come quella del 2007-2008. Gli effetti a livello sistemico di una situazione del genere possono avere conseguenze disastrose, che vanno oltre il fallimento della singola banca poiché il sistema economico finanziario moderno è altamente interconnesso con altri settori e con i mercati internazionali. La perdita di fiducia nel sistema bancario può portare al ritiro massiccio di fondi, innescando una spirale discendente in cui le banche sono costrette a limitare i prelievi e ridurre i prestiti. La difficoltà di accedere al credito da parte delle imprese e delle famiglie porta ad una contrazione economica. La possibilità di entrare in una recessione costringe le autorità a dover intervenire. Le operazioni messe in atto dalle istituzioni possono includere il salvataggio delle banche in difficoltà finanziarie, il sostegno finanziario di emergenza, garantire i risparmiatori ed evitare che il sistema creditizio si inceppi.

Negli anni ’90 sono iniziate ad avanzare proposte su come monitorare la salute dei sistemi finanziari e quali fossero le variabili che potessero maggiormente influenzarla. Uno studio di riferimento che ha contribuito significativamente alla nascita degli EWS è Demirgüç-Kunt e Detragiache (1998), in cui vengono proposti nuovi approcci utilizzando modelli econometrici come probit e logit, dove vengono considerati una vasta gamma di indicatori chiavi che possono segnalare l'imminenza di una crisi bancaria. Esaminando sia i fattori specifici del sistema bancario, come la liquidità delle banche, il livello di capitale e la redditività, sia i fattori macroeconomici come l'inflazione, il tasso di crescita del PIL e l'andamento del mercato azionario. I risultati dello studio indicano che le crisi bancarie sono spesso scatenate da una combinazione di fattori interni ed esterni. I fattori interni includono la debolezza delle banche in termini di insufficienza di capitale, cattiva gestione del rischio e liquidità inadeguata. I fattori esterni comprendono shock macroeconomici, come la riduzione del tasso di crescita economica, l'aumento dell'inflazione e i cali del mercato azionario. I modelli logit si sono dimostrati essere utili sia nell’individuare pattern nei dati e i fattori scatenanti, sia per scopi predittivi tanto da raggiungere il 70% di *classification accuracy* in sample. Altri studi, tra i quali (Caggiano et al., 2016) fanno notare che la durata media delle crisi sistemiche osservate storicamente è un determinante importante per discriminare tra i modelli alternativi. Nei campioni in cui la durata media delle crisi è relativamente lunga, il modello logit multinomiale, che distingue esplicitamente tra il primo anno della crisi e gli anni successivi alla crisi, migliora rispetto ai modelli logit binomiali più comunemente utilizzati. All'interno della classe di modelli logit binomiali, eliminare le osservazioni relative ai periodi post-crisi è empiricamente superiore al trattarle come periodi di tranquillità. Per quanto riguarda i modelli probit, (Antunes et al., 2018) dimostrano che aggiungendo una componente dinamica al modello per le crisi bancarie sistemiche, si migliora notevolmente l'accuratezza della previsione, sia in campione che fuori

campione. Inoltre, tenere conto del lag temporale delle variabili indipendenti durante gli eventi di crisi migliora anche la qualità degli strumenti di avviso precoce.

(Shularick e Taylor, 2012) hanno per primi proposto modelli logit ad effetti fissi che risultano avere prestazioni di forecast migliori dei *pooled logit*, in quanto catturano l'eterogeneità insita in fattori non osservabili specifici delle unità che potrebbero influenzare la relazione tra le variabili indipendenti e dipendenti. Tuttavia, i modelli pooled logit sono rimasti i più diffusi in quanto consentono di utilizzare l'intero campione del dataset, evitando una riduzione significativa dovuta al problema di perfetta previsione e ottenendo stime più robuste e rappresentative. La successiva proposta di Pigini (2021) è di stimare un modello ad effetti fissi utilizzando una log-verosimiglianza penalizzata, in modo da ottenere le previsioni anche per i paesi che non hanno mai registrato una crisi.

Un approccio alternativo è stato proposto da Kaminsky et al. 1998 noto come "approccio dei segnali". Questo approccio coinvolge il monitoraggio dell'evoluzione di vari indicatori economici che tendono a comportarsi in modo diverso prima di una crisi. Ogni volta che un indicatore supera un certo valore soglia, viene interpretato come un segnale di avvertimento che una crisi potrebbe verificarsi nei successivi mesi. Ogni indicatore viene considerato a sé stante, potrebbe accadere che un indicatore suggerisce la probabilità di una crisi, mentre un altro indicatore suggerisce l'assenza di una crisi, questo fa nascere il 'problema dei segnali contradditori'. Questa discrepanza può creare confusione e rendere difficile interpretare correttamente la situazione e adottare misure preventive da parte dei decisori politi. La soluzione proposta a questa criticità è di un sistema di allarme che considera una combinazione di indicatori, piuttosto che fare affidamento su un singolo indicatore.

Nell'ultimo decennio, il Machine Learning (ML) ha guadagnato una notevole attenzione come nuova tecnica per la costruzione degli Early Warning Systems nelle crisi bancarie sistemiche. I modelli costituiti con queste tecniche perdono di interpretabilità per la natura degli algoritmi stessi

ma acquisiscono la capacità di catturare gli effetti non lineari migliorando la previsione out of sample. Principalmente sono due i metodi di apprendimento usati in questo campo: supervisionato e rinforzato. I metodi di Machine Learning supervisionati sono ampiamente utilizzati nella costruzione degli EWS per le crisi bancarie sistemiche. Questi metodi si basano su un insieme di dati di addestramento che contiene etichette corrispondenti a determinati risultati della classe target, ad esempio l'occorrenza o meno di una crisi bancaria sistemica. Gli algoritmi di apprendimento supervisionato cercano di identificare i pattern nei dati attraverso l'analisi delle relazioni tra le variabili esplicative e le etichette di output nel dataset di addestramento. A seconda dell'algoritmo impiegato la costruzione del *bordo decisionale* avverrà in modo diverso. L'accuratezza delle previsioni dipende da tanti elementi, dalla qualità dei dati di addestramento dal tipo di dati, tipo di problema e capacità del modello di catturare la struttura interna del dataset (Guerra e Castelli, 2022). I metodi di Machine Learning rinforzati, noti anche come apprendimento con rinforzo, sono un altro approccio utilizzato nella costruzione degli EWS per le crisi bancarie sistemiche. Questi metodi implicano l'utilizzo di algoritmi che apprendono in un ambiente dinamico come, ad esempio, Deep Q-Network e Q-Learning. Essi vengono addestrati per massimizzare una funzione di ricompensa nel tempo, tenendo conto delle azioni intraprese e dell'output ottenuto (Athanasopoulos, 2018). I metodi supervisionati di Machine Learning offrono previsioni accurate e possono identificare pattern complessi nei dati superando in termini di previsione out of sample i modelli logit (Casabianca et al., 2019). Da (Holopainene e Sarlin, 2017) si evince che rispetto ai modelli di regressione più diffusi, quelli di ML riescono a performare meglio minimizzando l'errore della classe minoritaria. Gli studi di (Bluwstein et al., 2021) e (Beutel et al., 2019) ribadiscono la capacità degli algoritmi di machine learning di funzionare bene in out of sample, ma evidenziano anche le due principali criticità che li caratterizza; la tendenza di "overfittare" e la difficoltà di interpretazione dei modelli. Infatti, le tecniche di data mining cercano di massimizzare l'output del

modello rispetto al valore reale, per fare questo utilizzano le variabili del dataset e loro combinazioni, che potrebbero non avere un significato chiaro nel mondo reale. Le nuove variabili create vengono utilizzate per classificare le osservazioni spesso secondo logiche molto complesse, ciò rende i risultati complicati da interpretare. Per ovviare a queste problematiche, gli stessi (Bluwstein et al., 2021) propongono l'uso della teoria dei giochi in (Shapley, 1953) per migliorare la compressione delle black box costituite con l'apprendimento automatico. Recentemente (Torky et al., 2023) hanno proposto un nuovo modello di Intelligenza Artificiale Esplicabile (XAI) per riconoscere automaticamente le cause delle crisi finanziarie e interpretare le operazioni di selezione delle caratteristiche. Utilizzando un dataset di confronto, il modello XAI proposto utilizza il classificatore Gradient Boosting integrato nell'algoritmo Pigeon Inspired Optimizer (PIO), dove PIO viene usato per ottimizzare l'operazione di selezione delle caratteristiche. Successivamente, il classificatore Gradient Boosting aiuta a riconoscere le cause delle crisi finanziarie basandosi sulle caratteristiche più importanti ottenute raggiungendo performance migliori della Random Forest.

Un approccio più spinto rispetto alle tecniche tradizionali della machine learning è il Deep Learning che vede l'uso di reti neurali multistrato come LSTM o GRU per plasmare sistemi di EWS efficienti per la prevenzione delle crisi bancarie. Tölö (2020) ha fatto un tentativo in tal senso ed i risultati nella cross-validation sono coerenti con studi che dimostrano come i nuovi metodi di apprendimento automatico superino il modello di logit. Tuttavia, il modello di logit rimane uno strumento utile per la comunicazione semplice. Nell'analisi sequenziale invece non si è riscontrata una differenza sostanziale tra le performance delle reti neurali MLP e del modello di logit. Un ulteriore contributo alla letteratura viene da (Jin e Lin, 2023), che propongono un modello interpretabile di rete neurale ricorrente a memoria a lungo termine (IMV-LSTM) con focal loss (FL) per mitigare l'asimmetria tra le classi nel dataset. I risultati sono interessanti non solo per il tentativo di rendere comprensibile il modello ma anche per il 100% di Recall ottenuta sul set di test in cui oltre

alla presenza di variabili economico finanziarie, è stato aggiunto l'indicatore ESG per ciascuna nazione.

Tuttavia, uno dei principali problemi affrontati nella letteratura degli studi sulle crisi finanziarie è l'alto sbilanciamento[4] delle classi nei dati. Ciò causa problemi nella modellazione del problema e nel fare forecast. Le conseguenze di uno sbilanciamento di classe possono compromettere le prestazioni dei modelli di classificazione. Gli algoritmi sono influenzati dalla distribuzione non equa delle classi, tendono a concentrarsi sulla classe più popolosa del training set rispetto all'altra. Questo comportamento può portare a prestazioni distorte, con una precisione più bassa per la classe di minoranza e una precisione più alta per la classe di maggioranza. In molte applicazioni, come la diagnosi medica o la rilevazione delle frodi, è di particolare importanza identificare correttamente la classe di minoranza. Tuttavia, uno sbilanciamento di classe può portare a una maggiore propensione a classificare erroneamente un caso positivo come negativo (falsi negativi). Questo può avere conseguenze negative per la precisione e l'affidabilità del sistema. Un altro rischio dello sbilanciamento è l'overfitting. Gli algoritmi di machine learning possono sovra-adattarsi ai dati di addestramento, imparando dettagli specifici dal training set, ma non riuscendo a generalizzare altrettanto bene sui dati di test set. Di conseguenza, anche le metriche di valutazione tradizionali, come l'accuratezza, possono essere fuorvianti in presenza di classe sbilanciate. È necessario utilizzare metriche più appropriate, come l'area sotto la curva ROC (AUC-ROC), che è la metrica più usata nei paper citati, o la *Precision* e *Recall*, per valutare correttamente le prestazioni dei modelli su dataset sbilanciati (Satyasree e Murthy, 2013). Nella valutazione tra modelli diversi in situazione di forte asimmetria nei dati, si preferisce quello con una *Recall* elevata quanto più vicino al 100%. Ma tenere d'occhio solo la Recall potrebbe portare ad avere un modello che penalizza la classe maggioritaria, il che in situazioni particolari potrebbe essere quello che si cerca. Normalmente

[4]Situazione che si verifica quando la variabile target categorica non presenta un numero paritario di elementi per ciascuna categoria nel dataset

se invece si vuole avere un indice di quanto il modello ottenuto sia complessivamente migliore, in questo caso la scelta viene fatta sulla base del valore della *F Measure*, quanto più elevata è, tanto più il modello riesce a predire bene. Essa non è altro che una media ponderata tra le *Precision* e la *Recall*.

Per affrontare il problema delle classi sbilanciate, (Satyasree e Murthy, 2013) propongono due tecniche che sono diventate la pratica generalmente utilizzata nell'ambito analisi dati: *undersampling* cioè un sotto campionamento e *oversampling* cioè un sovra campionamento. La prima tecnica, si cerca di ridurre la quantità di campioni della classe maggioritaria per renderla più bilanciata rispetto alla classe minoritaria. Ciò può essere fatto rimuovendo casualmente campioni dalla classe più popolosa fino a raggiungere un rapporto desiderato tra le classi, metodo semplice da implementare, ma può comportare la perdita di informazioni importanti contenute nei campioni rimossi. In alternativa l'undersampling può essere fatto in modo da mantenere le caratteristiche iniziali della classe maggioritaria basato sull'algoritmo di clustering. In questo caso, si utilizza un algoritmo di clustering, come il K-Means, per identificare sottogruppi all'interno della classe maggioritaria. Successivamente, si esegue l'undersampling selezionando campioni da ciascun sottogruppo in modo da ottenere una classe con le stesse caratteristiche iniziali, ma meno popolosa. Questo approccio consente di mantenere una distribuzione simile di campioni rispetto al dataset originale, evitando la perdita di informazioni significative.

Invece l'oversampling è un metodo per incrementare la quantità di campioni della classe minoritaria per bilanciarla rispetto alla classe maggioritaria. Ciò può essere fatto replicando o sintetizzando campioni della classe minoritaria. Nel caso della replicazione, i campioni esistenti della classe minoritaria vengono duplicati fino a raggiungere il numero desiderato. Nel caso della sintesi, vengono creati nuovi campioni artificiali basati su quelli esistenti utilizzando tecniche come SMOTE (Synthetic Minority Over-sampling Technique) da Chawla N. V. et al (2002) che tiene conto della

struttura dei dati, oppure ROSE da Menardi G. e Torelli N. (2012) che li genera casualmente. L'oversampling può aiutare a mantenere più informazioni dalla classe minoritaria, ma può anche portare a un potenziale sovrapprendimento (overfitting) dei modelli, specialmente quando i campioni sintetici non rappresentano fedelmente la distribuzione dei dati reali. Sia l'undersampling che l'oversampling hanno i loro vantaggi e svantaggi e la scelta dipende dal contesto specifico e dalle caratteristiche del dataset. In alcuni casi, potrebbe essere utile combinare entrambe le tecniche per ottenere un migliore bilanciamento delle classi.

È importante notare che ci sono anche altre tecniche disponibili per affrontare il problema delle classi sbilanciate, come l'uso di pesi nelle funzioni di costo, un approccio cost-sensitive. Con questo metodo, si attribuiscono pesi diversi alle diverse classi durante il processo di apprendimento del modello. I pesi vengono assegnati in base all'importanza o alla rarità delle classi. Ad esempio, si può assegnare un peso maggiore alla classe minoritaria per dare più importanza alla sua classificazione corretta rispetto alla classe maggioritaria. Un algoritmo comunemente utilizzato per i classificatori cost-sensitive è il Metacost (Domingos, 1999). Il Metacost è un algoritmo di apprendimento supervisionato che adatta i pesi delle classi basandosi sui costi degli errori di classificazione. Esso tiene conto dei costi associati ai falsi positivi e falsi negativi per ciascuna classe, e cerca di minimizzare il costo totale dell'errore nel processo di classificazione.

L'approccio cost-sensitive e l'utilizzo di algoritmi come il Metacost sono utili quando le conseguenze degli errori di classificazione sono asimmetriche per le diverse classi. Ad esempio, in un problema di rilevamento di frodi finanziarie, gli errori che portano a classificare erroneamente una transazione legittima come frode possono avere costi molto più elevati rispetto agli errori che portano a classificare erroneamente una transazione fraudolenta come legittima. Tuttavia, è importante notare che questo metodo richiede una corretta specificazione dei costi associati agli errori di classificazione e può richiedere la raccolta di informazioni aggiuntive o l'analisi del dominio

per determinare tali costi in modo accurato. Bisogna considerare la complessità delle interazioni nel sistema finanziario, impiegando approcci più sofisticati e la combinazione simbiotica di più tecniche in modo da trarre a proprio vantaggio i pro di ciascuna di esse.

Capitolo 2

Dati e Metodologia

2.1 Il Dataset

Il presente lavoro riprende i dati di Pigini C. (2021) i quali sono pubblici e fruibili come IFS (International Financial Statistics) pubblicati dal Fondo Monetario Internazionale, oppure come WDI (World Development Indicators) pubblicati dalla Banca Mondiale. I dati utilizzati hanno una struttura panel contenente 129 paesi per un periodo temporale di 34 anni che va dal 1984 al 2017 dove come features si hanno: la variabile dipendente e le sette variabili esplicative, ripetute per ciascun anno e sono le seguenti:

- ***Current crisis***: è la variabile dipendente binaria che indica la presenza o meno della crisi nell'anno di riferimento secondo la definizione di Laeven and Valencia (2018), i quali affermano che affinché un evento possa essere considerato una crisi bancaria devono verificarsi una o entrambe delle seguenti condizioni: i) segni significativi di difficoltà finanziarie nel sistema bancario, come consistenti prelievi bancari, perdite nel sistema bancario e/o liquidazione di banche; ii) misure significative di intervento delle politiche bancarie in risposta a perdite significative nel sistema stesso;

Le variabili esplicative invece, sono state selezionate seguendo la letteratura pertinente sugli Early Warning Systems suddividendole in tre gruppi:

1. **Macroeconomico**, seguendo Demirgüç-Kunt and Detragiache (2005) e Davis and Karim (2008a), sono state incluse le variabili:
    - ***Real GDP growth***: riporta il tasso di crescita economica reale, o tasso di crescita del PIL reale, misura la crescita economica, espressa dal prodotto interno lordo (PIL), da un periodo all'altro, depurato dall'inflazione o dalla deflazione;

- ***Log GDP per capita***: è dato dalla divisione diretta del PIL totale per la popolazione su scala logaritmica, mira a catturare la ricchezza pro-capite;
- ***GDP deflator growth***: il deflatore del PIL è una misura dell'inflazione. È il rapporto tra il valore dei beni e servizi che un'economia produce in un determinato anno a prezzi correnti e quello dei prezzi prevalenti nell'anno di riferimento Thehindu, (2018).
- ***Real interest rate***: il tasso di interesse reale è il tasso di interesse sui prestiti che un investitore, risparmiatore o prestatore riceve (o si aspetta di ricevere) dopo aver tenuto conto dell'inflazione, un valore elevato di questo indice influenza negativamente la solvibilità dei debiti (Laeven and Valencia,2018);

2. **Condizioni monetarie**: ampiezza dell'offerta moneta (M2) coperta da riserve internazionali e crescita del rapporto credito/PIL
   - ***M2/Reserves***: rapporto tra M2 (misura dell'offerta di moneta che include contanti, depositi bancari e denaro vicino facilmente convertibile) e riserve valutario della Banca Centrale. Permette di cogliere la capacità del Paese di resistere a un'improvvisa interruzione e inversione degli afflussi di capitale. Più alto è il valore di questa variabile, maggiore è la vulnerabilità di incorrere in una crisi bancaria (Caggiano et al. ,2016).
   - ***Credit to GDP growth***: indicatore del tasso di crescita del rapporto tra credito privato interno reale e PIL. È stato adottato come punto di riferimento comune nell'ambito di Basilea III per guidare l'accumulo di buffer di capitale anticiclici (Drehmann et al., 2014). Una crescita repentina del credito può scatenare problemi di deterioramento degli asset bancari e/o riduzione della liquidità. L'incremento di tale rapporto si può associare ad una relazione positiva con la probabilità di una crisi.

3. **Fattori strutturali del sistema bancario**:

- ***Growth of net foreign assets to GDP***: indica il rapporto tra attività nette estere e PIL. La variabile riassume informazioni sulle vulnerabilità finanziarie del paese legate al rischio valuta, sostenibilità del debito estero e alla dipendenza dagli investimenti esteri.

2.1.1 Analisi Descrittiva

In questa sezione si riportano le informazioni essenziali circa le variabili nel dataset utilizzato. Nella Figura 2.1 viene mostrata la mappa con le nazioni colorate sulla base del numero di anni di crisi registrati.

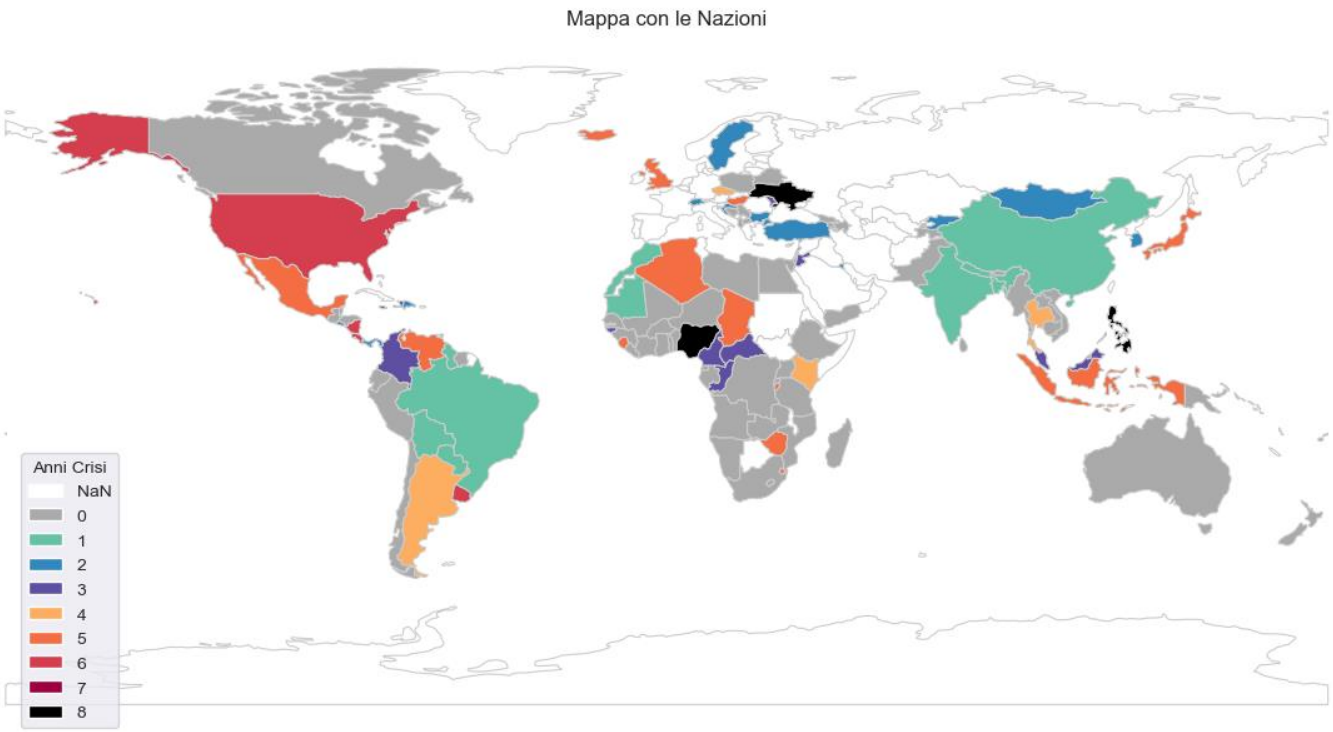

Figura 2.1

Attraverso il plot delle distribuzioni delle variabili, si vuole mettere in evidenza le differenze di ciascuna in relazione alla variabile dipendente. Per rendere più evidenti le differenze tra i boxplot, sono stati rimossi gli outlier più estremi, limitandoci a considerare i campioni compresi tra il primo

e il quarto quantile. Di seguito le figure 2.2, 2.3, 2.4, 2.5, 2.6, 2.7 e 2.8 riportano i *violinplot* delle 7 variabili indipendenti.

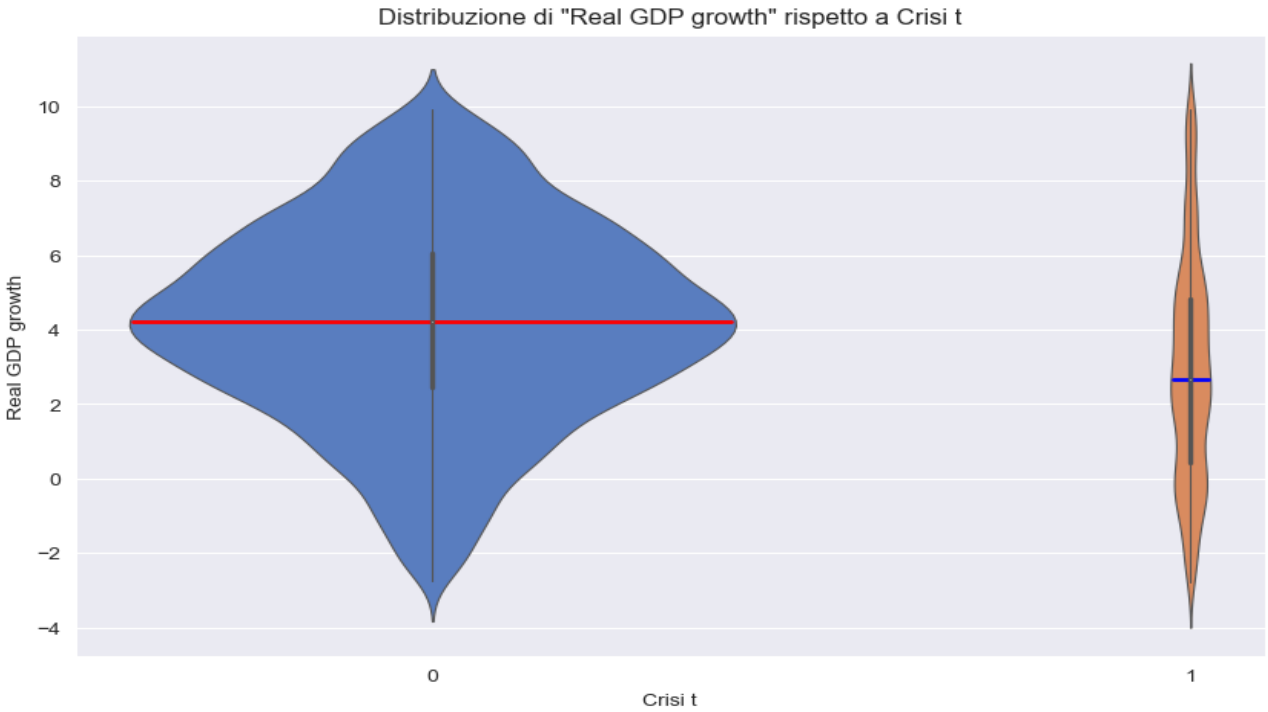

Figura 2.2

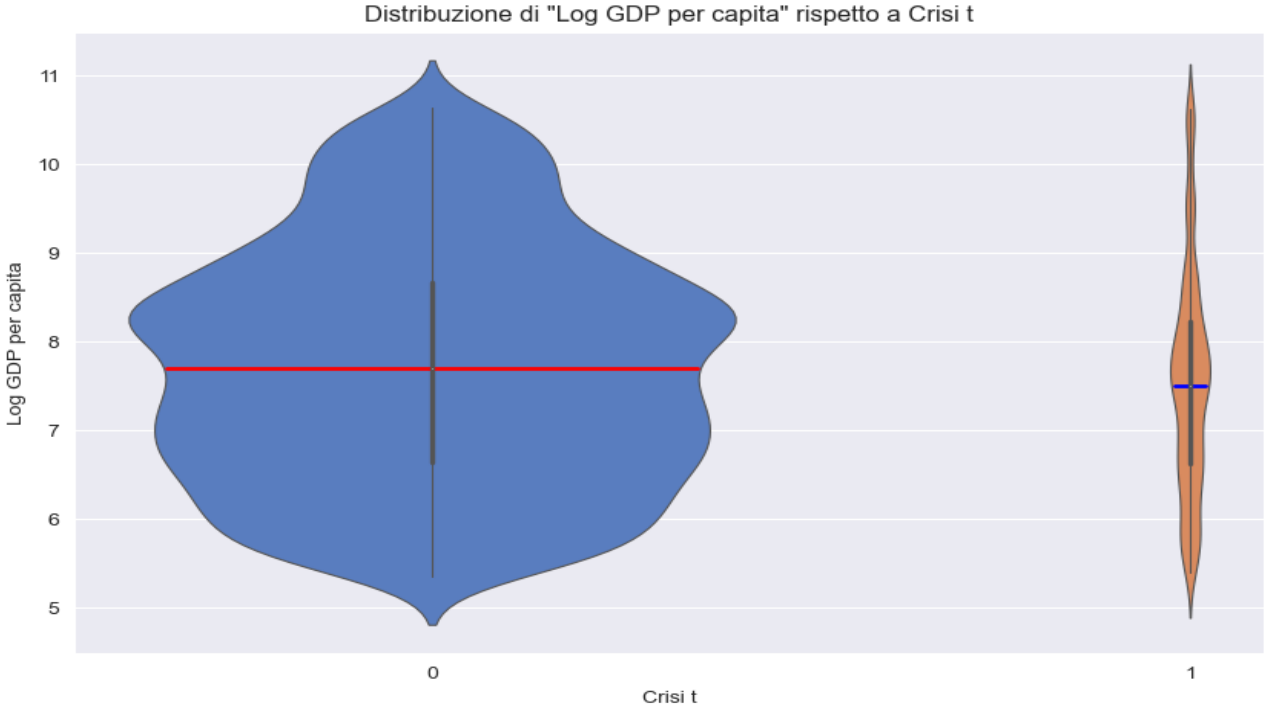

Figura 2.3

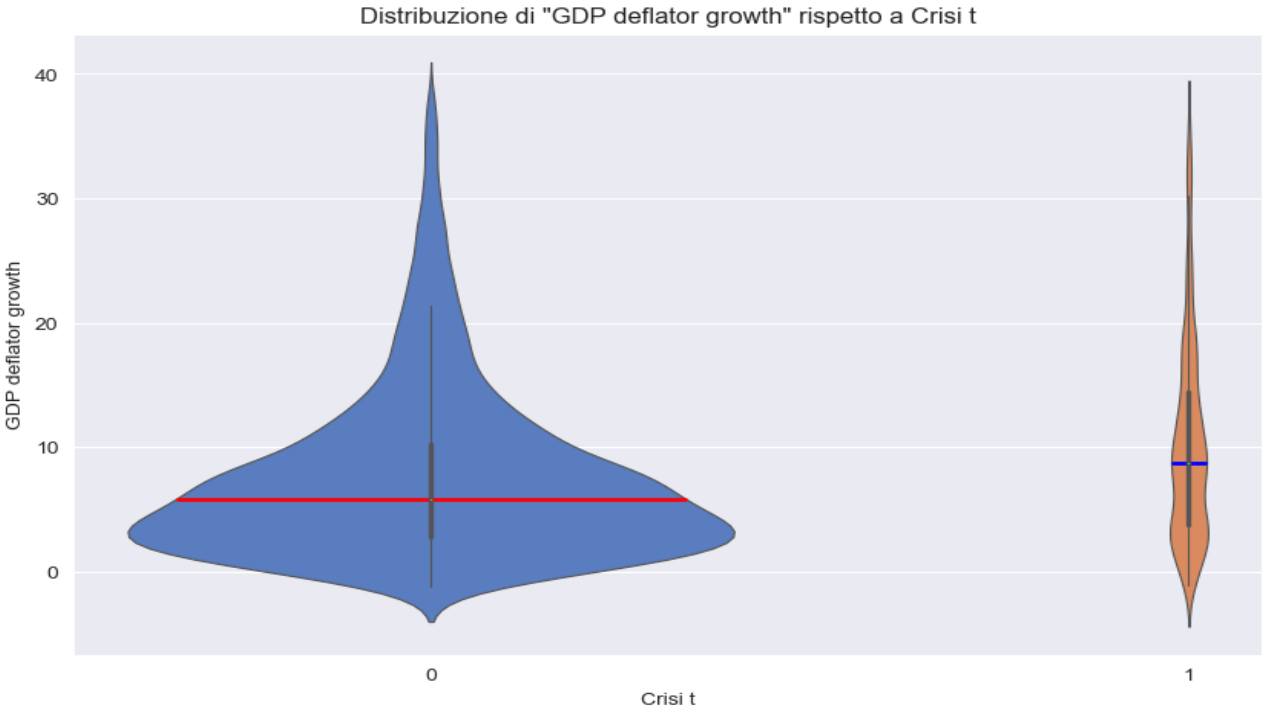

Figura 2.4

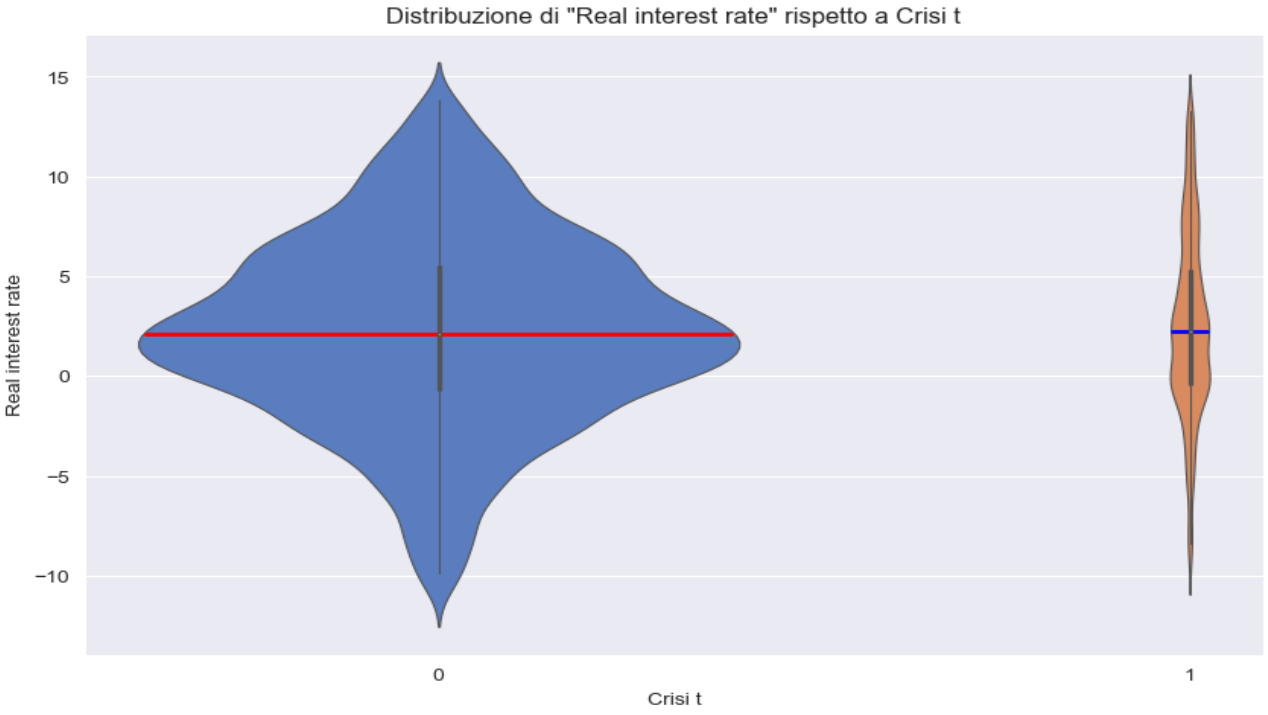

Figura 2.5

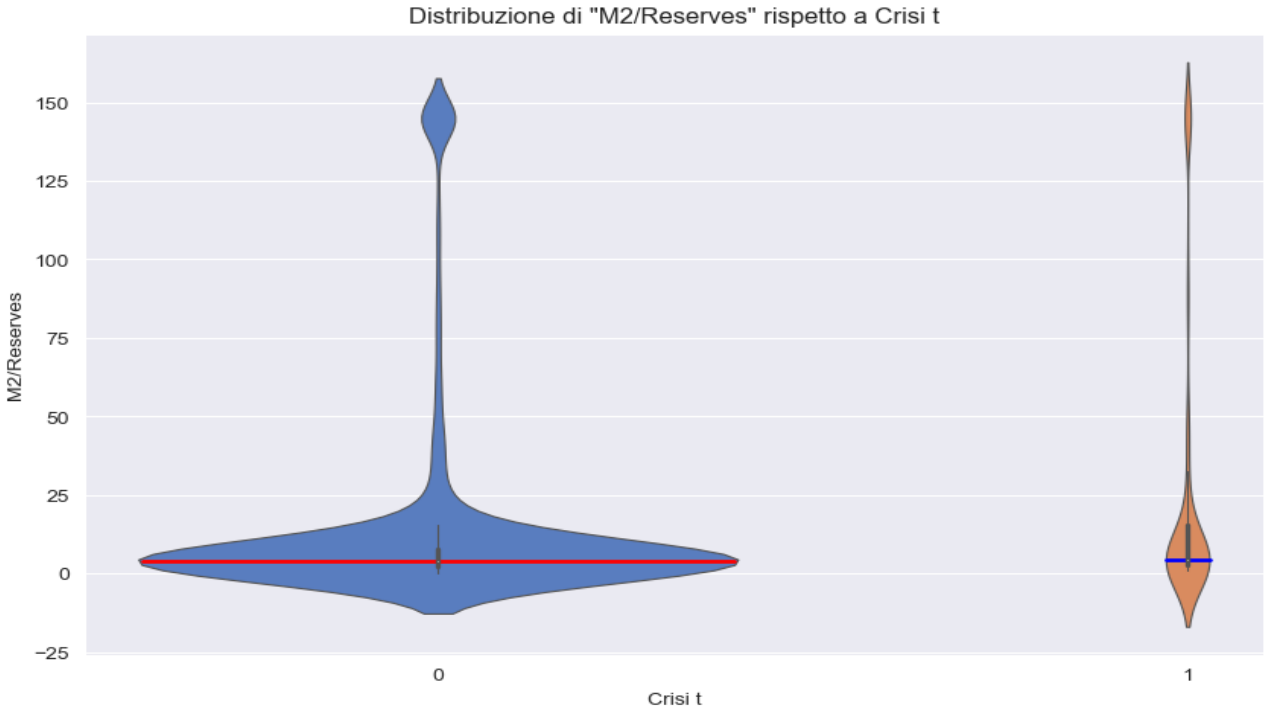

Figura 2.6

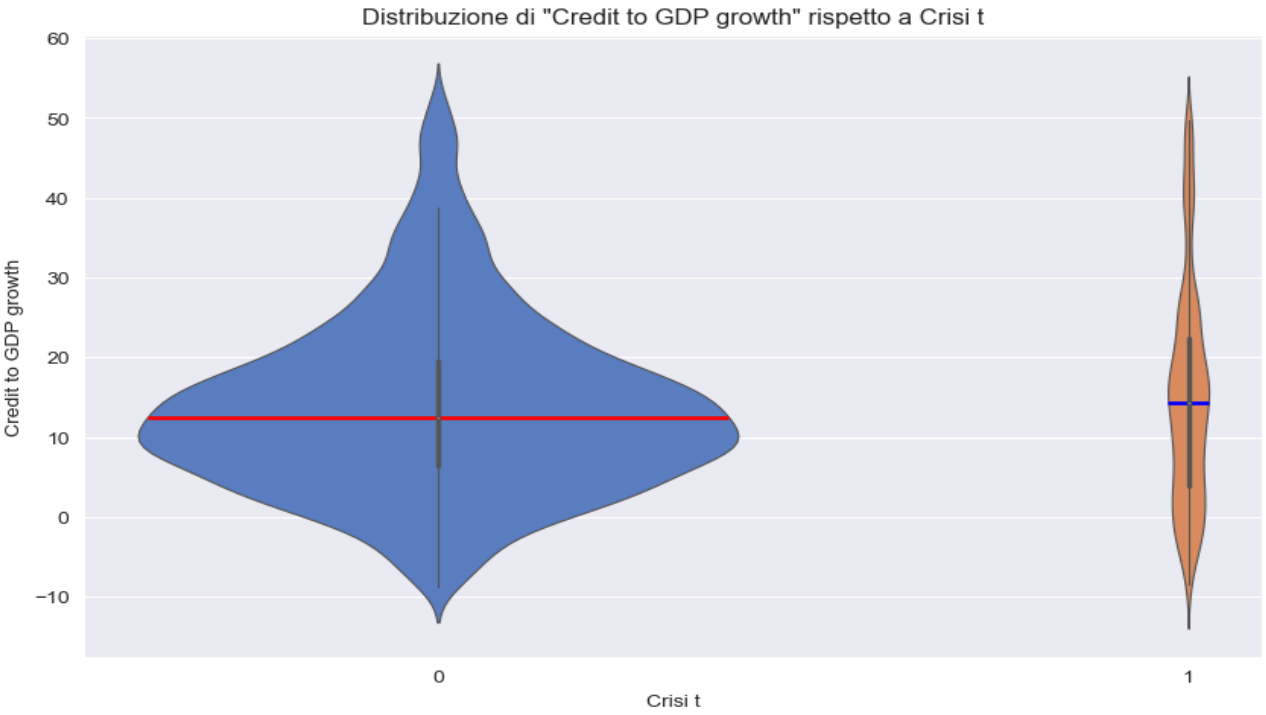

Figura 2.7

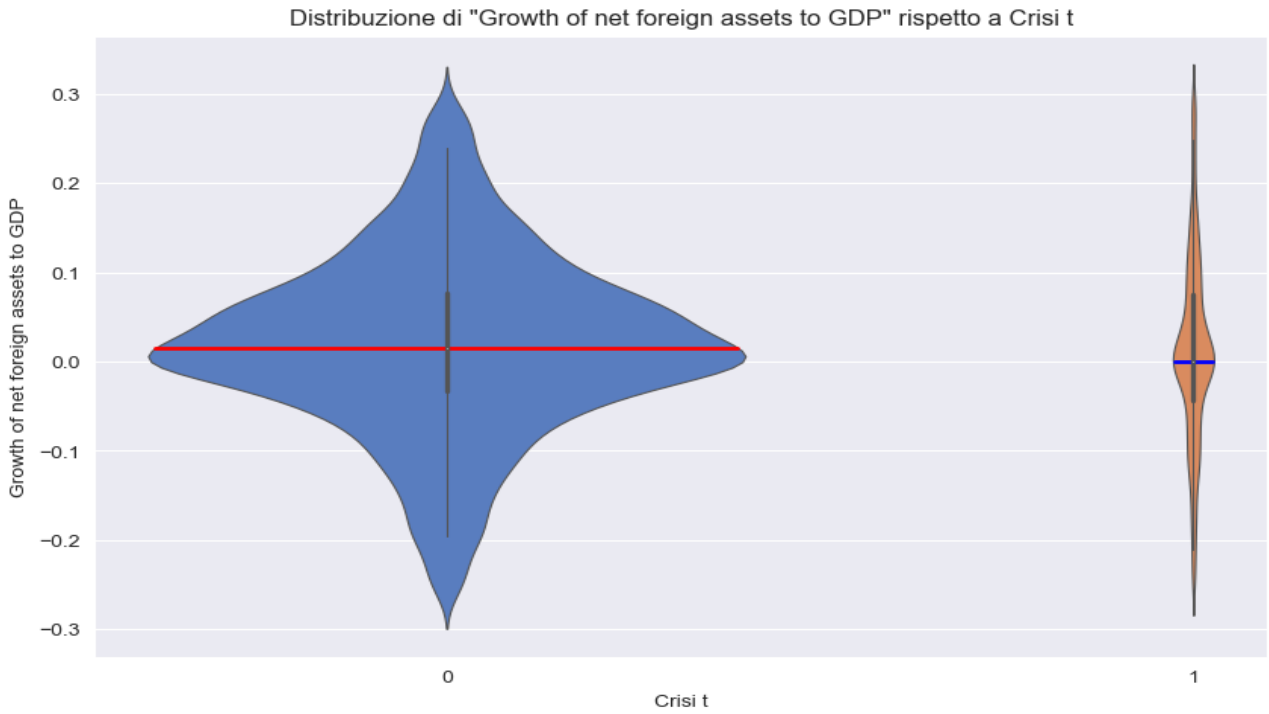

Figura 2.8

Si nota che seppure con forme diverse dovuto alla numerosità dei campioni, le due classi (0 e 1) hanno distribuzioni che non sono nettamente distinguibili secondo i parametri statistici, lo si osserva anche dai valori mediani evidenziati nei grafici. La Tabella 1 riporta le statistiche descrittive per ciascuna caratteristica nelle due classi.

Analisi statistica

Variabili	min 0	min 1	mean 0	mean 1	median 0	median 1	max 0	max 1
Real GDP growth	-13.8230	-13.8230	4.1466	1.8531	4.2367	2.3638	16.4383	18.1800
Log GDP per capita	4.7128	4.9688	7.7293	7.6825	7.6833	7.5403	11.3898	11.1335
GDP deflator growth	-31.5659	-12.9066	14.4246	96.5562	5.8010	9.8219	6261.2400	5016.1080
Real interest rate	-46.1111	-46.1111	2.3325	2.3715	2.1082	2.1826	66.1519	66.1519
M2/Reserves	0.0294	0.8873	14.0210	26.0440	3.9494	4.4105	144.9974	144.9974
Credit to GDP growth	-237.3343	-20.8898	15.0106	24.1458	12.3092	15.2021	405.9631	518.0811
Growth of net foreign assets to GDP	-2.4395	-1.0749	0.0184	0.0118	0.0150	0.0063	2.2972	1.3356

Tabella 1

2.1.2 Data Preprocessing

Questa fase è caratterizzata principalmente da due principali operazioni: data cleaning e data transformation. Nella prima, sono state eliminate le colonne “countryname” e “imfcode” poiché non necessarie, così come le osservazioni contenenti missing value. Nella seconda, si è proceduto con la trasformazione dei dati dal formato wide a quello cross section. Inoltre, è stata creata la variabile target dell’anno successivo ‘Crisis t+1’, oggetto della previsione basata sulle variabili al tempo t. La struttura del nuovo dataset è riportata nella Figura 2.9.

	country	Crisis t	gdp_growth	lgdp_per_capital	defl_gdp_growth	realirate	m2	domcrpriv_growth	ld_nfassets	Crisis t+1	year
0	USA	0.0	4.528648	9.652551	3.948367	4.628473	144.99740	10.064420	0.016643	0.0	1984
1	USA	0.0	7.007542	9.748837	3.548237	4.882592	144.99740	12.876350	0.023814	0.0	1985
2	USA	0.0	4.151125	9.812984	3.199612	4.350076	144.99740	9.563841	0.022628	0.0	1986
3	USA	0.0	3.451809	9.858232	2.017624	3.503928	144.99740	9.830336	0.043825	1.0	1987
4	USA	1.0	3.403046	9.908518	2.551295	3.479758	144.99740	6.911492	0.009882	0.0	1988
5	USA	0.0	4.118098	9.975028	3.500904	3.059316	144.99740	6.490453	-0.010683	0.0	1989
6	USA	0.0	3.614057	10.039870	3.888020	3.185107	92.84618	4.447268	-0.029817	0.0	1990
7	USA	0.0	1.901260	10.083910	3.699157	3.158056	81.37415	1.936560	-0.035966	0.0	1991

Figura 2.9

Inoltre, come visibile nella figura precedente i dati sono stati scalati utilizzando la funzione di normalizzazione MinMaxScaler portando i valori nel range 0-1. Ciò uniforma la scala delle variabili riducendo il rischio di rendere meno rilevanti o addirittura ignorare le features con valori più piccoli, anche se importanti ai fini predittivi. Per di più, l'assenza di normalizzazione può rendere difficile l'interpretazione dei coefficienti del modello o la comprensione dell'importanza dei diversi attributi, proprio perché i valori sono espressi su scale diverse.

La trasformazione deve tener conto del problema dimensionale dei dati (curse dimensionality), in una situazione di numerose variabili e poche osservazioni si rischia di imbattersi in complicazioni come la sparsità dei dati o overfitting. Avendo pochi dati gli algoritmi di apprendimento tendono a catturare i rumori o le fluttuazioni casuali portando ad avere delle pessime prestazioni nelle previsioni out of sample. Per evitare ciò, all’aumentare del numero degli attributi il numero di osservazioni deve crescere esponenzialmente. Con la trasformazione precedente siamo passati dall’avere un dataset di 129x275 ad averne uno con dimensioni 2888x11,

come effetto indiretto si è ottenuto una riduzione del problema della dimensionalità, perdendo però informazione sulla dinamica temporale delle variabili.

2.2 Metodologia

2.2.1 Tecniche impiegate

Il presente lavoro riprende la metodologia del lavoro di Rocchi (2022) in termini di bilanciamento delle classi, confrontando tali risultati con i nuovi approcci proposti. L'idea di base è, a partire da un dataset fortemente sbilanciato 6,7:100 con 2888 elementi totali, splittarlo in training e test folds attraverso il k-fold cross validation[5] e poi addestrare i modelli con dati bilanciati, oppure applicando tecniche di cost sensitivity. Bilanciare i dati è necessario per le ragioni dette nel Capitolo 1. Lo si effettua sul training set in modo che il modello venga artificiosamente addestrato su un campione equilibrato in numero di elementi per le singole classi, ma poi la fase di test avviene sui dati reali sbilanciati. Al fine di bilanciare il training set sono state applicate due tecniche di campionamento: undersampling della classe maggioritaria con il k-means e oversampling della classe minoritaria con lo SMOTE. Oltre al working flow ripreso da Rocchi (2022), per incrementare la corretta classificazione della classe 'No crisi' sono stati introdotti dei costi per penalizzare gli elementi classificati erratamente attraverso il meta-algoritmo di MetaCost. Inoltre, sono state provate diverse combinazione tra le tre tecniche e testato il funzionamento delle reti ricorrenti di tipo LSTM al problema affrontato.

Il flusso di lavoro per gli approccio *sampling* e *cost sensitive* sono stati svolti principalmente sul software RapidMiner con alcune integrazioni implementate con Python, soprattutto per le fasi di preprocessing, analisi descrittiva e visualizzazione dei risultati. Mentre per l'approccio alle reti, tutte le fasi compreso l'addestramento sono state svolte su Python. Di seguito vengono descritte in

[5] È una tecnica di sampling utilizzata per la validazione del modello poiché divide il dataset in k folds, usa k-1 fold come dati di training ed il fold restante come test per k volte ogni volta cambiando i fold.

maniera dettagliata le tecniche impiegate, i parametri e le metriche adoperate per valutare i modelli e scegliere i migliori.

Come prima operazione è stato effettuato un undersampling con il K-Means della classe 'Crisi' in modo da ridurre proporzionalmente il numero di osservazioni dai k gruppi individuati limitando la perdita di informazione. Il K-Means è un algoritmo di clustering unsupervised utilizzato per suddividere in K sottoinsieme i dati, dove K è un iperparametro che bisogna specificare in partenza. Ogni cluster viene rappresentato da un centroide, che è la media delle osservazioni appartenenti al cluster. L'algoritmo cerca di minimizzare la somma dei quadrati delle distanze tra ogni punto e il centroide del cluster a cui appartiene, assegnando gli esempi ai cluster più vicini in modo da ottimizzare la compattazione e allo stesso tempo la separazione dei cluster. Le fasi costituenti dell'algoritmo sono le seguenti:

1. **Inizializzazione**: fase in cui si sceglie il numero predefinito di cluster K e si selezionano casualmente K punti dal dataset come centroidi iniziali.
2. **Assegnazione**: Per ciascun punto del dataset, si calcola la distanza euclidea tra il punto e i centroidi di tutti i cluster. Si assegna il punto al cluster il cui centroide è più vicino.
3. **Ricalcolo dei centroidi**: Viene ricalcolato la media di ciascun cluster tra i punti assegnati al medesimo gruppo, il valore medio trovato viene indicato come il nuovo centroide. Questo fa sì che il centroide si sposta verso il nuovo centro dei punti assegnati mentre la forma e la posizione dei cluster vengono modificate.
4. **Riassegnazione**: In questa fase vengono ripetuti i passaggi 2 e 3 finché i centroidi dei cluster non vengono più aggiornati, oppure fino a quando non viene raggiunto un criterio di stop (come il numero massimo di iterazioni).

Alla fine, l'algoritmo fornisce i K cluster con i rispettivi centroidi ed ogni punto del dataset sarà stato assegnato a uno di essi. Per valutare se i gruppi sono stati individuati correttamente e se contengono

elementi quanto più omogenei al loro interno, si usano metriche come la *coesione* e *separazione* della Silhouette Analysis. Le quali, oltre a fornirci un'indicazione sull'ottimalità dei gruppi individuati possono essere adoperate per determinare il numero ottimale di cluster iniziale. La Silhouette valuta la coesione e la separazione dei cluster assegnando un valore di silhouette a ciascun esempio, misurando quanto è simile agli altri esempi nel proprio cluster rispetto a quelli degli altri cluster. Un valore di silhouette vicino a 1 indica una buona separazione dei cluster, mentre valori vicini a 0 indicano sovrapposizioni o cluster mal definiti. Il K ottimale è associato al valore di silhouette più alto.

Un altro è l'Elbow Method, che calcola la somma dei quadrati delle distanze intra-cluster per diversi valori di K fornito dall'utente. L'idea è di trovare il punto in cui l'aumento del numero di cluster smetta di avere un effetto significativo sulla riduzione della somma dei quadrati delle distanze, formando un "gomito" nel grafico. Il K associato al "gomito" è spesso considerato il numero ideale di cluster. Come è possibile notare nella Figura 10 e Figura 11, entrambi i metodi hanno indicato come numero di cluster ottimale k=4.

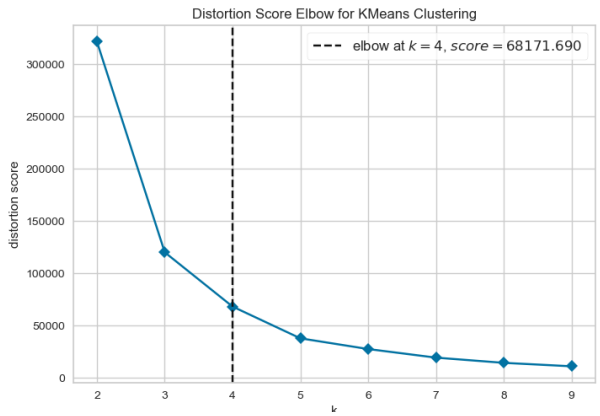

Figura 2.10

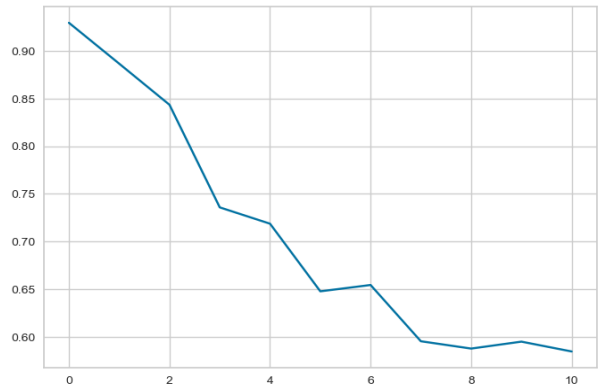

Figura 2.11 Silhouette score

La seconda tecnica impiegata per ridurre il *Class Imbalance* è l'oversampling tramite lo SMOTE. Mentre con undersampling si eliminavano campioni, con questa tecnica si vuole crearne di nuovi per le classi minoritarie, in modo da bilanciare la distribuzione delle classi. Il processo si basa

sull'identificazione dei dati della classe minoritaria e sulla creazione di esempi sintetici interpolando le caratteristiche tra i punti vicini. In pratica, per ogni punto della classe minoritaria, SMOTE seleziona k punti vicini, calcola la differenza tra il punto corrente e i suoi vicini, e crea nuovi punti lungo queste direzioni. Tuttavia, è importante notare che l'uso di questa tecnica può anche introdurre rumore o sovrapposizioni nei dati, quindi va utilizzato con cautela. Le fasi che lo caratterizzano sono le seguenti fasi:

1. **Identificazione dei punti della classe minoritaria**: La prima fase consiste nell'identificare i punti di dati appartenenti alla classe minoritaria nel dataset sbilanciato.
2. **Selezione dei punti vicini**: Per ogni punto della classe minoritaria, vengono selezionati k punti vicini nel dataset. Il valore di k è un iperparametro specificato in anticipo e rappresenta il numero di vicini da selezionare. Tipicamente k è scelto come un numero tra 5 e 10.
3. **Creazione di campioni sintetici**: Per ogni punto della classe minoritaria, si calcola la differenza tra il punto corrente e i suoi k punti vicini selezionati. Successivamente, si generano nuovi campioni sintetici interpolando le caratteristiche tra il punto corrente e i punti vicini, utilizzando un valore compreso tra 0 e 1. In altre parole, si creano nuovi punti lungo le linee che collegano il punto corrente con i punti vicini, e la posizione di questi nuovi punti dipende da un parametro di sovrapposizione.
4. **Aggiunta dei campioni sintetici**: I nuovi campioni sintetici vengono aggiunti al dataset, aumentando così la rappresentazione della classe minoritaria e bilanciando la distribuzione delle classi.

Il modo per individuare il numero ottimale di nuovi elementi da generare è quello empirico, valutando di volta in volta le prestazioni del modello su diversi livelli di oversampling. Come parametri di customer si possono far variare il valore di k (il numero di vicini) e il parametro di sovrapposizione, monitorando le prestazioni del modello utilizzando metriche di valutazione come

l'accuratezza, la Precision, il Recall o la F measure. L'obiettivo è trovare un equilibrio tra il miglioramento della performance del modello e la generazione di dati sintetici realistici e rappresentativi della classe minoritaria.

La terza tecnica è il meta-algoritmo MetaCost che mira a migliorare le performance del modello introducendo una penalizzazione diversa per gli errori di classificazione in base alla classe di appartenenza. In altre parole, attribuisce un peso diverso agli errori commessi su ciascuna classe, in modo da "incoraggiare" il modello a prestare maggiore attenzione alle classi con costo di errore più elevato migliorando così la loro performance previsiva. Le diverse fasi dell'algoritmo MetaCost vengono descritte da Domingos (1999):

Inputs:
S is the training set,
L is a classification learning algorithm,
C is a cost matrix,
m is the number of resamples to generate,
n is the number of examples in each resample,
p is True iff L produces class probabilities,
q is True iff all resamples are to be used for each example.

Procedure MetaCost (S, L, C, m, n,p, q)

For i = 1 to *m*
  Let S_i be a resample of *S* with *n* examples.
  Let M_i = Model produced by applying *L* to S_i.

For each example *x* in *S*
  For each class *j*
    Let $P(j|x) = 1/\sum 1 * \sum P(j|x, M_i)$
    Where
      If *p* then $P(j|x, M_i)$ is produced by M_i
      Else $P(j|z, M_i) = 1$ for the class predicted by M_i for *x*, and 0 for all others.
      If *q* then *i* ranges over all M_i
      Else *i* ranges over all M_i such that $x \notin S_i$.
  Let *x*'s class = $\text{argmin}_i \sum P(j|x)C(i, j)$

Let *M* = Model produced by applying *L* to *S*.
Return *M*

Quindi, in seguito, all'addestramento di un classificatore base (ad esempio, un algoritmo di machine learning come decision tree) sul dataset sbilanciato senza considerare il bilanciamento delle classi, vengono valutati gli errori di classificazione su ciascuna classe e le probabilità a priori $Pi = n_i/n_{samples}$, dove $n_{samples}$ è il numero totale di campioni nel dataset e n_i è il numero di campioni della classe i. Secondo la logica di ICF (Inverse Class Frequency) a ciascuna classe viene assegnato come peso l'inverso della probabilità a priori della stessa $Wi = 1/P_i$, dove W_i è quindi il peso assegnato agli errori per la classe i. I pesi degli errori calcolati vengono quindi utilizzati per aggiornare il classificatore base. Il modello viene riaddestrato prendendo in considerazione i nuovi pesi, in modo da penalizzare di più gli errori sulle classi minoritarie e ridurre la tendenza del modello a predire in modo sbilanciato. Questo processo di aggiornamento dei pesi e di riaddestramento del modello può essere iterato per ottenere una convergenza migliore e una performance ottimale.

L'ultima tecnica impiegata, sono le reti neurali ricorrenti (RNN) di tipo LSTM. Le reti neurali in generale sono un tipo di modello di machine learning ispirato dal funzionamento del cervello umano. Sono composte da unità di calcolo chiamate "neuroni artificiali" o "perceptroni" e da tre diversi stratti: ***input layer*** è lo stratto iniziale che riceve i dati e passa l'informazione agli strati successivi; ***hidden layer*** contengono i neuroni intermedi e rappresentano il cuore della rete, ogni neurone è connesso a tutti i neuroni negli strati precedenti e successivi con dei pesi associati; ***output layer*** è lo stratto finale produce l'output finale della rete, che può essere una classificazione, una previsione numerica o un'altra forma di risultato desiderato. Nella Figura 2.12 possiamo vedere una rete multistrato con 3 hidden layer.

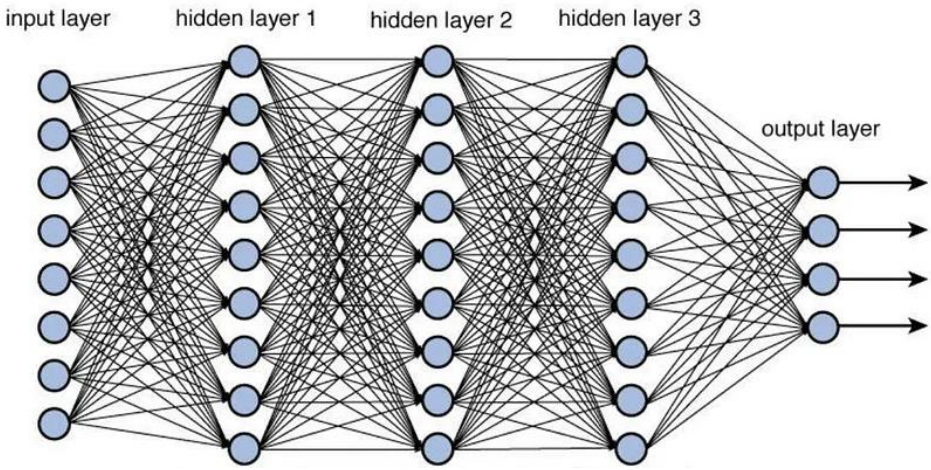

Figura 2.12

L'addestramento delle reti neurali è un processo che permette alla rete stessa di imparare dai dati e migliorare le sue capacità di risolvere problemi specifici, in particolare cerca di produrre un output che si avvicini quanto più all'output dato. Questo processo coinvolge principalmente due fasi: il forward pass e il backpropagation.

- **Forward Pass:**

  1. I dati di addestramento ricevuti nello strato di input percorrono la rete neurale, passando da uno strato all'altro fino all'output layer.
  2. Ogni neurone in uno strato nascosto calcola il potenziale di attivazione, che è la somma ponderata delle connessioni in ingresso dallo strato precedente, compreso un termine bias. Questo potenziale di attivazione per il neurone j nello strato nascosto h può essere calcolato come segue: $Z_j^{(h)} = \sum_{(h)}\left(w_{ij}^{(h)} x_i\right) + b_j^{(h)}$

     Dove $w_{ij}^{(h)}$ rappresenta il peso della connessione tra il neurone i nello strato precedente e il neurone j nello strato nascosto h, x_i è l'output del neurone i nello

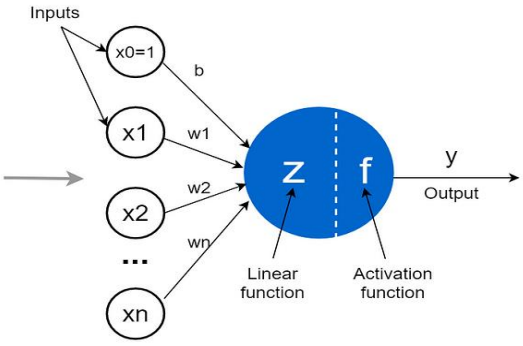

Figura 2.13

strato precedente, $b_j^{(h)}$ è il termine bias del neurone j nello strato nascosto h.

3. La funzione di attivazione non lineare, come la funzione ReLU (Rectified Linear Unit) o la funzione sigmoide, viene quindi applicata al potenziale di attivazione per ottenere l'output del neurone nello strato nascosto. $a_j^{(h)} = f(z_j^{(h)})$, dove f è la funzione di attivazione.
4. L'output dell'ultimo strato nascosto viene passato allo strato di output, dove viene calcolato l'output finale della rete.
5. L'errore tra l'output previsto e il target (output desiderato) viene calcolato utilizzando una funzione di costo, ad esempio, l'errore quadratico medio (MSE) per problemi di regressione o l'entropia incrociata per problemi di classificazione.

$$E = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Dove N è il numero di esempi di addestramento, y_i è l'output desiderato e $\hat{y}_i$è l'output previsto per l'esempio i.

- **Backpropagation:**

6. I gradienti dell'errore rispetto agli output dell'ultimo strato nascosto e agli output finali vengono calcolati utilizzando la regola della catena: $\frac{\delta E}{\delta a_j^{(L)}}$, dove L rappresenta l'ultimo strato di output.
7. I gradienti vengono quindi retropropagati attraverso la rete, calcolando i gradienti dei pesi e dei bias in ogni neurone in base ai gradienti dell'errore rispetto agli output.
8. Gli ottimizzatori, come l'Algoritmo di Gradiente Stocastico (SGD) o Adam, vengono utilizzati per aggiornare gradualmente i pesi delle connessioni nella rete, utilizzando i gradienti calcolati. Il tasso di apprendimento (α) controlla la dimensione degli aggiornamenti dei pesi.

$$w_{ij}^{(h)} \leftarrow w_{ij}^{(h)} - \propto \frac{\delta E}{\delta w_{ij}^{(h)}}$$

Questi passaggi vengono ripetuti per molti esempi di addestramento attraverso diverse iterazioni o epoche fino a quando la rete neurale raggiunge una prestazione desiderata sui dati di addestramento. In questo modo, la rete neurali multistrato apprende a modellare complesse relazioni nei dati e a fare previsioni accurate. Le reti più adatte al tema affrontato sono però le RNN (Recurrent Neural Network). Le quali sono strumenti potenti per affrontare il complesso compito di costruire modelli di Early Warning System (EWS) per rilevare crisi bancarie sistemiche. Questo perché le RNN sono intrinsecamente progettate per gestire dati sequenziali e catturare dipendenze temporali nei dati. A differenza delle reti neurali classiche, che non riescono a considerare l'ordine o il contesto temporale dei dati, le RNN sono in grado di analizzare serie storiche apprendendo relazioni non lineari tra indicatori. Inoltre, hanno la capacità di mantenere uno stato interno, che si evolve con le iterazioni temporali, che consente loro di catturare dipendenze a lungo termine nei dati, un aspetto cruciale nell'identificazione delle crisi bancarie sistemiche. Tuttavia, va notato che le RNN tradizionali possono incontrare il problema del *'vanishing gradient'*, in cui i gradienti degli errori diminuiscono notevolmente durante il processo di apprendimento a causa delle iterazioni temporali. Per superare questa sfida, sono state introdotte le Long Short-Term Memory (LSTM), che utilizzano specifiche unità con porte per gestire in modo più efficace lo stato di memoria e il flusso delle informazioni. Le LSTM riescono così a catturare dipendenze a lungo termine nei dati, identificando correlazioni cruciali e combinando informazioni recenti e passate per migliorare la capacità di riconoscere modelli di comportamento distintivi nei dati analizzati. Poiché siamo in presenza di una situazione di analisi di eventi rari, la scelta delle metriche per misurare l'efficienza dei modelli è stata di usare la Precision, Recall e F measure, piuttosto che le più tradizionali Accuracy

e ROC curve. Le formule per calcolare gli indicatori precedenti sono esplicitate di seguito nella Figure 2.14 e Figura 2.15.

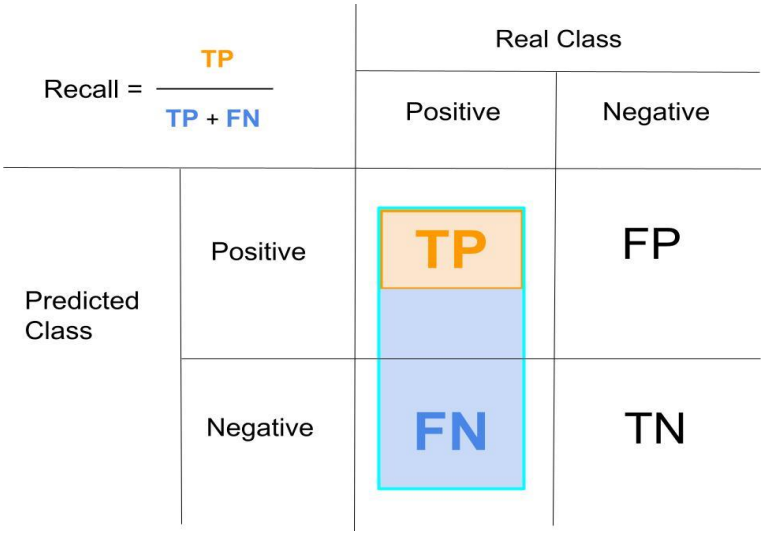

Figura 2.14 riporta la formula della *Recall*

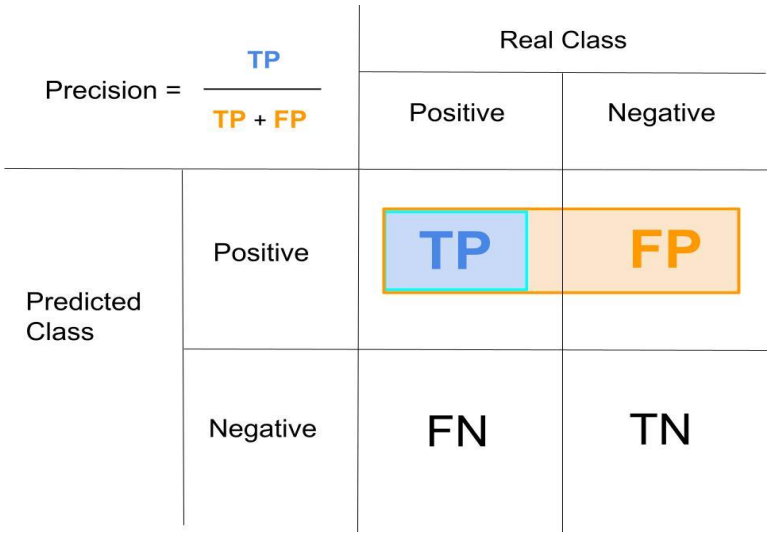

Figura 2.15 riporta la formula della *Precision*

Dove:

- *True Positive* (TP): numero di crisi predette correttamente come crisi;
- *True Negative* (TN): numero di non crisi predette correttamente come non crisi;
- *False Positive* (FP): numero di non crisi predette come crisi;
- *False Negative* (FN): numero di crisi predette come non crisi;

Recall e Precision sono due metriche di valutazione molto importanti utilizzate per valutare le prestazioni di un modello di classificazione. Mentre Recall si concentra sulla capacità del modello di individuare correttamente gli esempi positivi, e un valore alto indica che il modello riesce a individuare la maggior parte dei casi positivi. Precision, d'altra parte, si concentra sulla correttezza delle classificazioni positive effettuate dal modello stesso. Un valore alto di Precision indica che il modello indica pochi falsi positivi, ma potrebbe perdere alcuni dei casi positivi reali. Dato il contesto dell'analisi e la natura delle misure, idealmente si preferisce un modello con un alto valore di Recall poiché in grado di ridurre i falsi negativi, individuando così tempestivamente le crisi bancarie. Tuttavia, questo porta ad incrementare il numero di falsi positivi cioè a segnalare false crisi bancarie. Quindi, per avere una misura che indichi quanto complessivamente un modello sia migliore di un altro si usa la F Measure, che non è altro che una media ponderata tra le precedenti.

$$FMeasure = \frac{2 * Precision * Recall}{Precision + Recall}$$

Mentre l'Accuracy indica complessivamente di veri positivi (TP) e veri negativi (TN), quanti ne vengono predetti correttamente rispetto al totale di osservazioni.

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

2.2.2 Classificatori

In questa sezione vengono presentati i classificatori impiegati per implementare i modelli di previsione e le griglie per la scelta degli iperparametri.

- **Decision Tree**: è un algoritmo di machine learning che svolge sia compiti di classificazione che di regressione. Il processo di costruzione dell'albero decisionale inizia con un nodo radice contenente tutti i dati di addestramento. L'algoritmo cerca la caratteristica che meglio separa i dati in base a una metrica di *impurità* indicata. Suddivide quindi il nodo in due o più figli corrispondenti alle diverse opzioni della caratteristica scelta. Questo processo di suddivisione continua ricorsivamente per i nodi figli fino a quando si verifica una delle seguenti condizioni: la profondità massima dell'albero è raggiunta, il numero minimo di campioni per foglia è soddisfatto o la suddivisione non migliora significativamente la purezza delle foglie. I decision tree sono apprezzati per la loro interpretabilità e flessibilità, ma possono essere inclini all'overfitting su dati complessi e per alberi molto profondi. Per mitigare questo problema, possono essere utilizzate tecniche come il *prunning* degli alberi, l'uso di ensemble di alberi, o la regolazione degli iperparametri. Nella Tabella 2 sono elencati gli iperparametri e i valori che sono stati oggetto di iterazione per il Decision Tree al fine di selezionare la combinazione che presentasse la F-measure più elevata.

Criterio di split	Gini index; Accuracy; Information gain; Gain ratio
Profondità massima	[1; 16; 31; 46; 61; 76; 90; 105; 120; 135; 150]
Pruning e pre-pruning	True

Tabella 2

- **Random Forest:** è una tecnica di ML che funziona creando un insieme di alberi decisionali, ognuno addestrato su un sottoinsieme casuale di dati e di attributi. Durante la fase di predizione, ciascun albero fornisce una previsione e la classe (o valore) finale viene determinata dalla votazione o media delle predizioni dei singoli alberi. La Random Forest è apprezzata per la sua robustezza, la capacità di gestire grandi quantità di dati e la caratteristica di essere meno incline

all'overfitting rispetto a un singolo albero decisionale. Tuttavia, potrebbe non essere la scelta migliore in tutte le situazioni, specialmente se l'interpretabilità è una priorità o se si dispone di risorse computazionali limitate. Tabella 3 indica gli iperparametri iterati per trovare il modello migliore.

Criterio di split	Gini index; Accuracy; Information gain; Gain ratio
Profondità massima	[1; 31; 61; 90; 120; 150]
Numero di alberi	[1; 81; 161; 240; 320; 400]
Pruning e pre-pruning	True

Tabella 3

- **Logistic Regression**: è un algoritmo per problemi di classificazione binaria. Nella fase di addestramento, l'algoritmo cerca di stimare i coefficienti per massimizzare la funziona log-Likelihood, che rappresenta la probabilità di ottenere i dati di addestramento così come sono stati osservati sulla base delle variabili esplicative. In altre parole, cerca i coefficienti che rendono più probabile ottenere un output dal modello che sia quanto più vicino ai dati osservati. Per trovare i beta ottimali, vengono utilizzati ottimizzatori come il Gradient Descent, L-BFGS o altri. La regressione logistica è apprezzata per la sua semplicità e interpretabilità, ma può essere limitata in problemi con dati non linearmente separabili.

Ottimizzatore	AUTO; IRLSM; L_BFGS; COORDINATE_ DESCENT; COORDINATE_ DESCENT_NAIVE
Ad intercept e remove collinear column	True

Tabella 4

Le **Support Vector Machines** (SVM) sono algoritmi di machine learning supervised utilizzati principalmente per problemi di classificazione, anche se possono essere estesi a problemi di regressione. Gli SVM sono noti per la loro capacità di trattare efficacemente dati complessi e non separabili linearmente. L'obiettivo principale degli SVM di classificazione è di trovare un iperpiano che separi al meglio l'insieme degli elementi delle classi, massimizzando la distanza (margine) tra i punti più vicini delle due classi, chiamati "vettori di supporto". Se i dati non possono essere separati linearmente nello spazio originale, si possono utilizzare i kernel (come il kernel RBF o polinomiale) per mappare i dati in uno spazio di dimensione superiore. Questa mappatura può rendere i dati linearmente separabili nello spazio trasformato. I kernel impiegati negli esperimenti di tesi sono: *Linear, Radial* e *Polynomial*.

- Gli **SVM lineari** cercano di separare le classi con un iperpiano lineare nello spazio delle caratteristiche. L'obiettivo è massimizzare la larghezza del margine tra le classi poiché questa separazione massimizza la generalizzazione del modello. Il parametro da indicare per tale algoritmo è *C*, noto come parametro di regolarizzazione, controlla la penalizzazione degli errori di classificazione nel modello SVM lineare. Un valore grande di *C* implica una penalizzazione più elevata per gli errori, rendendo il modello più "rigido" e potenzialmente più incline all'overfitting sui dati di addestramento, poiché il margine costruito è ampio. Al contrario, un valore più piccolo di *C* permette errori di classificazione più alti, rendendo il modello più "flessibile" e riducendo il rischio di overfitting.
- Le **SVM Radial** sono una variante degli SVM che utilizzano la funzione kernel RBF per mappare i dati in uno spazio delle caratteristiche ad alta dimensionalità, dove le classi potrebbero essere linearmente separabili. La funzione RBF genera un iperpiano non lineare nello spazio trasformato, consentendo di gestire dati non linearmente separabili. La scelta dell'iperparametro *gamma* oltre al parametro *C* influisce sulla flessibilità del modello.

Gamma controlla quanto è sensibile il kernel alla distanza tra i due punti. Un valore più grande di esso rende il kernel più sensibile alla distanza producendo un modello più conservativo. Mentre un valore più piccolo di *gamma* rende il kernel meno sensibile alla distanza costruendo un modello più aggressivo, dove sarà più propenso a classificare i punti come appartenenti a classi diverse anche se sono vicini l'uno all'altro.

- Le **SVM polinomiali** sono un altro tipo di SVM non lineare che utilizzano una funzione kernel polinomiale per trasformare lo spazio originario dei dati. La complessità del modello dipende dall'iperparametro *grado*, che determina il numero di termini polinomiali utilizzati per costruire il modello. Un valore alto come *grado* produrrà un modello più complesso, in quanto sarà in grado di apprendere relazioni più complesse tra i dati ma anche più suscettibile a overfitting. Al contrario, un valore basso di *grado* produrrà un modello più semplice, in quanto sarà in grado di apprendere solo relazioni relativamente semplici tra i dati.

Nella Tabella 5 si possono osservare i valori dei parametri usati nella fase del loop.

Linear SVM	C: [0.001; 0.01; 0.1; 0.4; 0.8; 1; 10; 100; 1000]
Radial basis function SVM	C: [0.001; 0.01; 0.1; 0.4; 0.8; 1; 10; 100; 1000]; Gamma: [0.0001; 0.001; 0.01; 0.1; 0.2; 0.4; 0.6; 0.8; 1; 10]
Polynomial SVM	C: [0.001; 0.01; 0.1; 0.4; 0.8; 1; 10; 100; 1000] Degree of the polynomial: [2; 3; 4; 5; 6]

Tabella 5

- **Le reti LSTM**: Il passaggio da reti neurali ricorrenti (RNN) a celle LSTM è stato un importante sviluppo per superare il problema del "*vanishing gradient*" delle RNN tradizionali come anticipato precedentemente. Le LSTM sono state progettate per migliorare il flusso dei gradienti

durante l'addestramento, consentendo alle reti neurali di catturare dipendenze di lungo e breve termine nei dati sequenziali. La struttura della cella LSTM è composta da tre parti:

1. **Stato di Memoria** (Cell State): lo stato di memoria *C* è la componente principale di una cella LSTM e rappresenta la memoria a lungo termine. Lo stato di memoria viene aggiornato e gestito attraverso le porte
2. **Stato Nascosto** (Hidden State): l'output *h* della cella LSTM è noto come stato nascosto e viene utilizzato per la previsione e può contenere informazioni rilevanti per la previsione immediata di breve periodo.
3. **Porte della Cella LSTM**: la quale a sua volta è composta da altre tre porte ciascuna con la sua funzione. *Porta di Dimenticanza* (Forget Gate) questa porta determina quali informazioni dello stato di memoria passato debbano essere dimenticate o mantenute. Utilizza una funzione sigmoide f_t per produrre valori tra 0 (dimentica completamente) e 1 (mantieni completamente). *Porta di Input* (Input Gate) questa determina quali nuove informazioni debbano essere aggiunte allo stato di memoria. Utilizza una funzione sigmoide i_t per determinare quali informazioni incorporare. Infine, la *Porta di Output* (Output Gate) che cosa verrà restituito come output basato sullo stato di memoria corrente. Utilizza una funzione sigmoide o_t per il controllo e una funzione tangente iperbolica (tanh) per produrre l'output. Nella Figura 2.16 vediamo la raffigurazione dell'architettura di una cella LSTM.

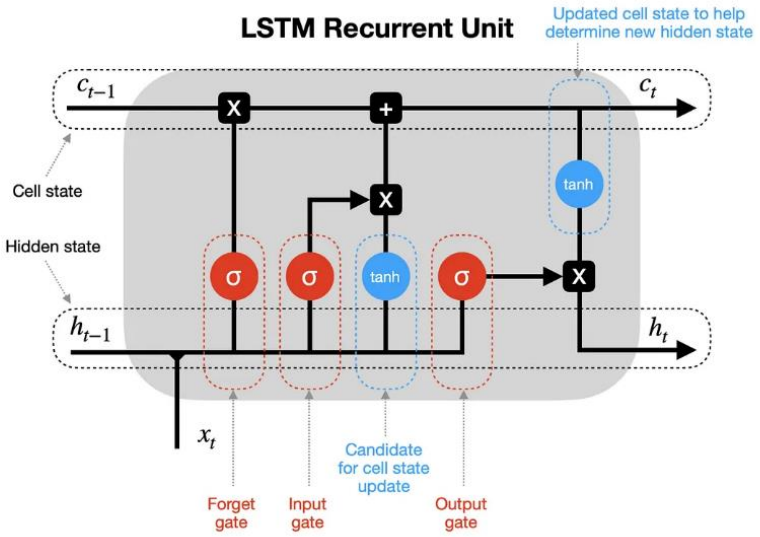

Figura 2.16

La fase di addestramento e all'aggiornamento dei pesi è simile a quella per le reti classiche vista precedentemente con l'aggiunta di qualche passaggio nuovo caratteristico della loro struttura. Le fasi sono così composte:

1. **Inizializzazione**: All'inizio dell'addestramento, i pesi delle porte (come W_f, W_i, W_o) e i bias (come b_f, b_i, b_o) all'interno della cella LSTM vengono inizializzati casualmente.
2. **Forward Pass**: Durante la fase di forward, l'input corrente x_t e lo stato nascosto precedente h_{t-1} vengono utilizzati per calcolare le uscite delle porte forget gate f_t, input gate i_t, output gate o_t e la nuova informazione proposta $\widetilde{C}_t$ come segue.

   Calcolo della Porta di Dimenticanza: $f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$

   Calcolo della Porta di Input: $i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$

   Calcolo della Porta di Output: $o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$

   Calcolo della Nuova Informazione Proposta: $\widetilde{C}_t = \tanh(W_c * [h_{t-1}, x_t] + b_c)$
3. **Aggiornamento dello Stato di Memoria**: lo stato di memoria precedente C_{t-1} viene aggiornato utilizzando le informazioni delle porte di dimenticanza, input e la nuova informazione proposta: $C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$

4. **Calcolo dell'Uscita**: l'uscita corrente h_t viene calcolata utilizzando il nuovo stato di memoria C_t e l'uscita della porta di output o_t: $h_t = o_t * \tanh(C)$
5. **Calcolo dell'Errore e Backpropagation**: viene calcolato l'errore tra l'uscita prevista e l'uscita desiderata. Quindi, i gradienti vengono calcolati risalendo attraverso la rete utilizzando la retropropagazione dell'errore (backpropagation).
6. **Aggiornamento dei Pesi**: gli aggiornamenti dei pesi *W* e dei bias *b* all'interno della cella LSTM avvengono tramite un algoritmo di ottimizzazione come la discesa del gradiente. I gradienti calcolati vengono utilizzati per regolare i pesi in modo da minimizzare l'errore durante l'addestramento.

Questo processo di forward pass, calcolo dell'errore e aggiornamento dei pesi viene ripetuto attraverso molte iterazioni (epoche) fino a quando la rete raggiunge una prestazione desiderata sui dati di addestramento. La modellazione con le LSTM parte dal definire i parametri chiave, tra cui l'ottimizzatore, la dimensione del batch, il tasso di apprendimento e l'architettura della rete neurale. Questi parametri sono selezionati in base alle esigenze specifiche del problema e alla natura dei dati.

L'addestramento inizia con la compilazione del modello, specificando l'ottimizzatore (Adam o SGD sono i più diffusi) e la funzione di costo da minimizzare (“binary_crossentropy”). Per i dati di addestramento viene usata una slide windows che considera informazioni passate rispetto all’attuale, aggiornando iterativamente i pesi della rete attraverso il processo di backpropagation durante un numero fisso di epoche. In questa fase, il modello apprende a modellare le relazioni nei dati e a fare previsioni accurate. Durante ciascun’epoca dell'addestramento, un sottoinsieme separato dei dati noto come set di validazione viene utilizzato per valutare le prestazioni del modello. Questi dati sono indipendenti dai dati di addestramento principali ed è utilizzato per misurare metriche di valutazione come l'F1-score o l’Accuracy. L'obiettivo è monitorare le

prestazioni del modello durante l'addestramento per identificare segnali di overfitting. Infine, per valutare le prestazioni reali del modello, viene eseguito un test finale su un set di dati completamente separato, noto come set di test. Questo set di test rappresenta situazioni reali o nuovi dati e non è stato utilizzato in precedenza per l'addestramento o la validazione. Il modello viene utilizzato per fare previsioni su questo set, e le prestazioni finali del modello vengono valutate sulla base delle performance ottenute.

Tuttavia, la costituzione del modello finale è un processo molto lungo e complesso poiché richiede tempi di calcolo elevati per la fase di training-validation-test ma soprattutto di difficile scelta la definizione dell'architettura ottimale. Poiché il numero di parametri da definire e da calibrare è molto elevato rispetto agli altri classificatori precedentemente visti. Nella tabella a seguire vengono riportati gli iperparametri ed i valori usati per cercale la migliore configurazione.

numero di neuroni	[8, 16, 32, 64, 128]
numero di strati	[1, 2, 3, 4]
ottilizatore	['Adam', 'SGD']
tasso di apprendimento	[0.001, 0.01, 0.1]
batch size	[8, 16, 32, 64]
epoche	[50, 200, 500, 600, 800, 1000]
sliding window	[3, 5, 7]
TimeSerieSplit	[3, 5]

Tabella 6

2.2.3 Esperimenti svolti

Considerando le tecniche e i classificatori precedentemente presentati, il presente lavoro è stato incentrato sul testare diverse combinazioni di tecniche di campionamento che potessero migliorare il potere predittivo dei modelli rispetto a quelli ottenuti con la tecnica del *Sampling*. Le diverse combinazioni testate nei vari esperimenti hanno contribuito a creare un set di dati bilanciato o comunque meno sbilanciato rispetto a quelli originali. Utilizzando tale campione come il training set

per gli algoritmi di apprendimento e sviluppare modelli di classificazione 'customizzati' secondo i parametri che permettono di ottenere performance di F Measure migliori. In modo specifico per i modelli ML, tutto questo è stato ripetuto per ciascun esperimento 6 volte, che è il numero dei diversi algoritmi testati utilizzando una cross-validation con k=10. La divisione del campione in fold aiuta a mitigare il problema della variabilità dei dati nel dataset e a fornire una valutazione più robusta delle prestazioni del modello. Inoltre, aiuta a utilizzare al meglio i dati disponibili per l'addestramento e la valutazione, soprattutto in presenza di dataset di piccole dimensioni.

Esperimento 1

Il primo esperimento è stato effettuato per testare il funzionamento del *cost sensitive learning* applicato al problema delle crisi bancarie, che è stato messo in contrapposizione alla prova di bilanciamento delle classi proposto nel lavoro di Rocchi (2022). La validazione del modello è sempre avvenuta attraverso una k-fold cross validation con k=10, all'interno del quale sui dati di training vi è stato applicato il MetaCost, a sua volta contenente l'algoritmo di apprendimento. Prima però di costituire il modello, attraverso una ricerca a griglia sono stati trovati i valori dei parametri per i quali si ottiene il miglior indice di F Measure. La matrice dei costi risulta essere un iperparametro, pertanto la ricerca a griglia descritta precedentemente è stata effettuata tenendo conto anche i diversi possibili pesi assegnati ai valori classificati erratamente. Uno schema complessivo si può vedere nella Figura 14 riassuntivo del processo sviluppato in RapidMiner. I risultati dei singoli esperimenti possono essere consultati in Appendice [1].

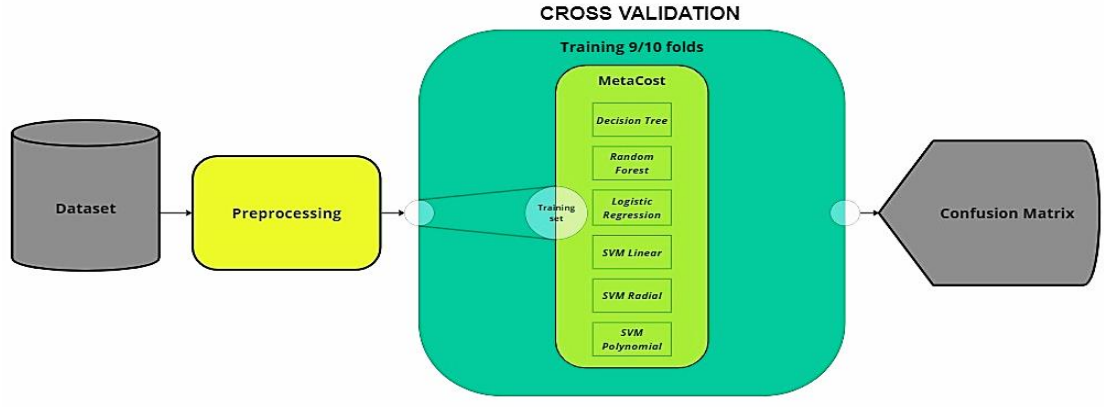

Figura 2.17

Esperimento 2

Il secondo esperimento è stato pensato tentando un approccio ibrido tra il metodo cost sensitive e il bilanciamento paritario dei dati. In particolare, trascurando la parte di oversampling dove vengono generati nuovi elementi che possono essere fonte di rumore per la fase di addestramento del modello. È stato provato un undersampling della classe maggioritaria del 25%, mentre il numero della minoritaria è rimasto invariato, per poi applicare il meta-algoritmo MetaCost. Questa combinazione di tecniche è stata applicata alla fase di training per poi effettuare il test del modello sui dati nella forma originaria. Come per l'esperimento precedente, è stata effettuata una ricerca a griglia dei valori per gli iperparametri, matrice dei costi compresa. La Figura 15 riporta il flusso del lavoro effettuato, mentre i risultati delle singole matrici di confusione sono consultabili in Appendice [2].

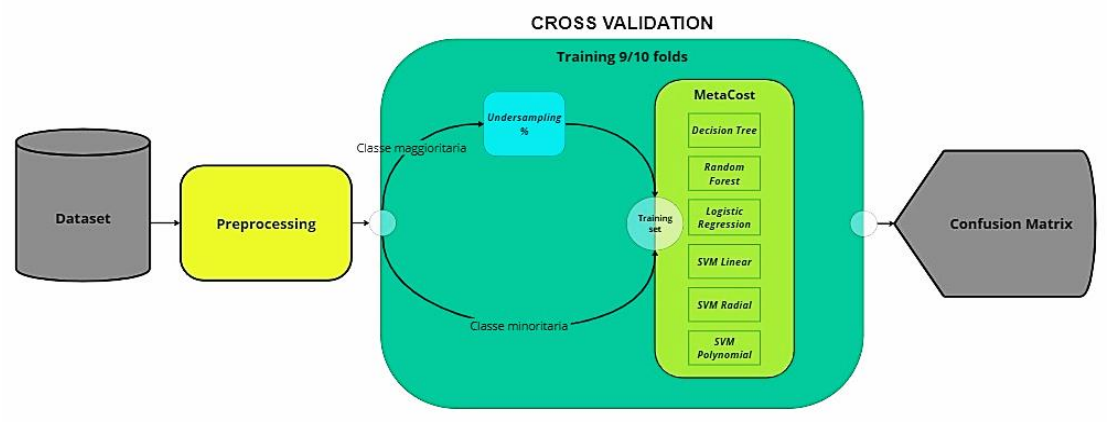

Figura 2.18

Esperimento 3

Nella terza prova è stato ripetuto l'Esperimento 2 cambiando la percentuale di sottocampionamento, usando una percentuale del 15% anziché del 25%. L'obiettivo è di ridurre ulteriormente lo squilibrio tra il numero di osservazioni delle due classi, ma senza generarne di nuovi, riapplicando il MetaCost. Risultati consultabili in Appendice [3].

Esperimento 4

In un set di dati con 129 paesi diversi, gli algoritmi possono faticare a catturare i segnali predittivi, poiché le singole nazioni hanno caratteristiche diverse fra loro e si possono trovare in fasi del ciclo economico e di sviluppo diversi. Formare gruppi omogenei basati su misure di similarità tra essi può aiutare gli algoritmi nella previsione e ad individuare pattern e segnali predittivi più rilevanti e specifici, riducendo il rumore nei dati. Per cui in questo esperimento si propone una filosofia diversa rispetto ai precedenti, suddividendo il dataset sulla base della durata delle crisi precedenti verificatesi nelle singole nazioni. Secondo Caggiano (2016), questo è un approccio che risulta essere un elemento determinante per le prestazioni dei diversi modelli logit nel costruire gli EWS. Sulla base del suo lavoro, le singole nazioni sono state suddivise in tre gruppi, nel gruppo '*sotto*' sono state inserite quei paesi che nel periodo 1984-2017 hanno registrato una durata media di crisi bancaria inferiore alla media del dataset che è di 2,86 anni. Il gruppo '*media*' contiene coloro che hanno in media il valore che si discosta al massimo di 4 mesi dalla media dell'intero campione. In fine il gruppo '*sopra*', contiene tutte quelle nazioni che hanno registrato nell'arco temporale considerato mediamente più di 2,86 anni di crisi. Possiamo vedere la distribuzione dei gruppi nella Figura 16 dove per ragioni di leggibilità sono stati plottati i nomi delle nazioni in modo alternato.

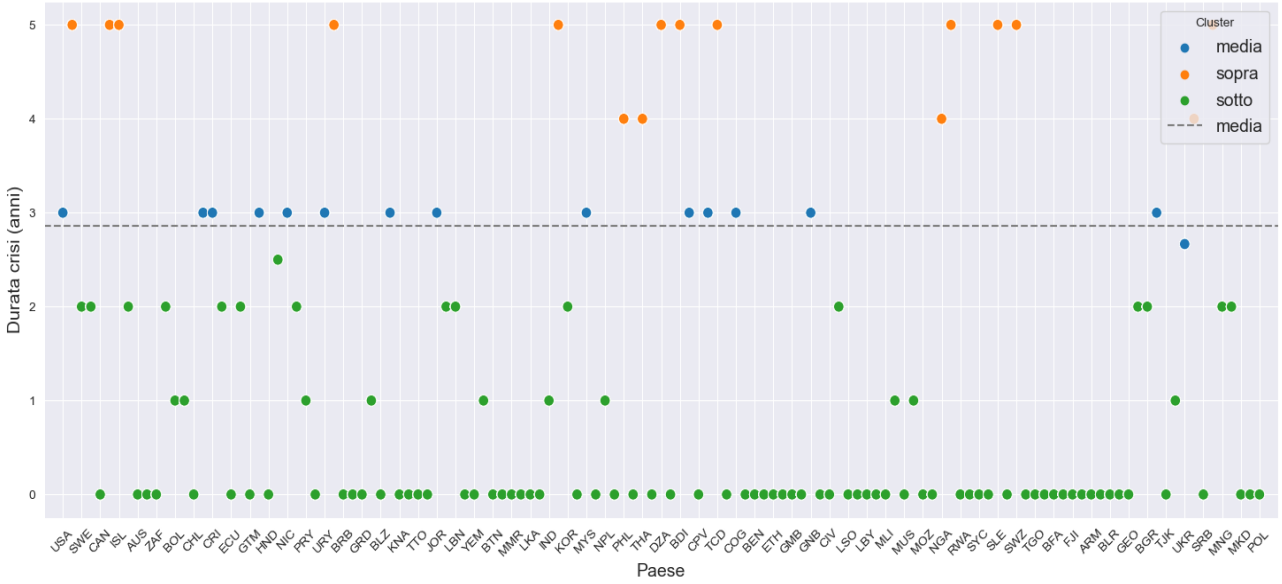

Figura 2.19

Fatto ciò, per ciascun cluster è stata applicata la procedura degli altri esperimenti, la ricerca dei valori per i parametri e la costruzione dei modelli per ciascuno dei sei classificatori proposti, considerando il MetaCost come tecnica per la regolazione del costo di misclassificazione. In questo modo si sono ottenuti 18 modelli (per il *Decision Tree* tre modelli uno per ciascun gruppo, e così anche per gli altri algoritmi), è stata effettuata una media pesata tra le matrici dei risultati dei tre gruppi, in modo da ottenere un'unica *Confusion Matrix* complessiva. In modo da poterla confrontare con i risultati degli altri esperimenti. Risultati consultabili in Appendice [4].

Esperimento 5

A partire dall'esperimento precedente, anziché suddividere le osservazioni secondo la variabile intrinseca (la durata delle crisi bancaria) presente nel campione, è stato provato un clustering non supervisionato mediante K-Means. Come per il metodo del sottocampionamento, la scelta del valore di k è stato determinato dalla prova del metodo Elbow e Silhouette sull'intero campione. Da tale verifica, il numero di cluster ottimale è stato scelto k=4. Dato che ogni paese appare più di una volta nel dataset, per decidere a quale cluster assegnarlo è stata usata la logica della "majority voting", come conseguenza da quattro gruppi formati inizialmente sono rimasti in tre, con medie di: 12,21 cluster0, 50,65 cluster1 e 58,34 cluster3. Nella Figura 17 vediamo il plot rispetto alla covariata M2/Reserves che risulta essere la più discriminante. In seguito, le fasi di scelta dei parametri e la costituzione del modello finale sono state identiche a quelle dell'Esperimento 4. Risultati consultabili in Appendice [5].

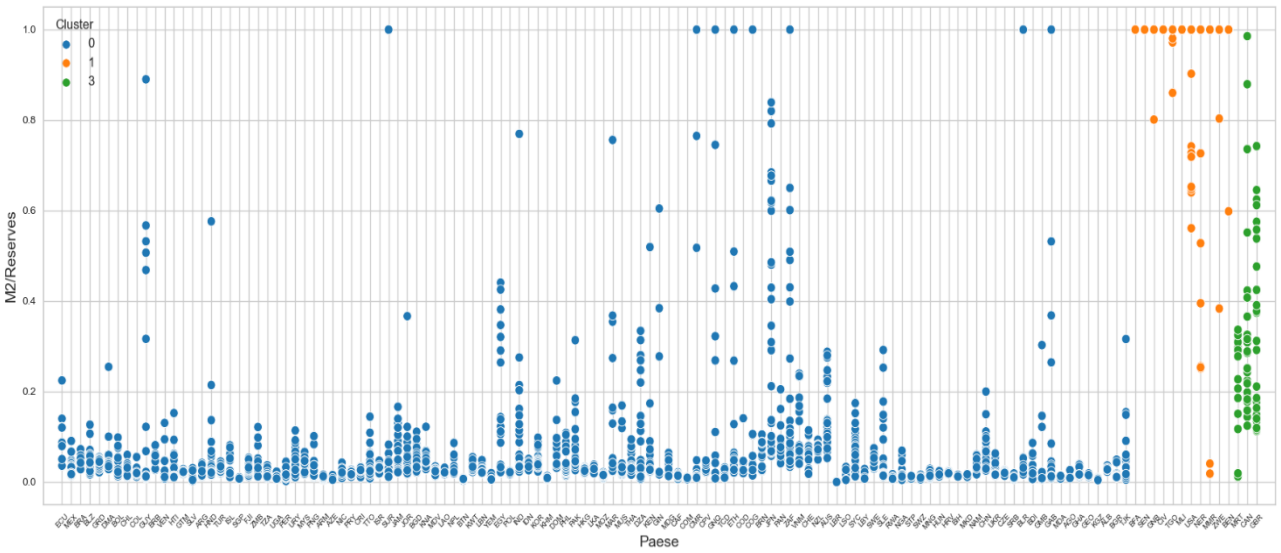

Figura 2.20

Esperimento 6

Dal correlogramma si è notato che la variabile crisi al tempo t, è correlata del 0.67 con la target. Per sfruttare l'informazione antecedente a essa sono stati aggiunti dei lag temporali. La trasformazione da wide a cross section ha comportato la perdita di informazione poiché è stata spezzata la variabile temporale considerando l'osservazione solo nell'istante t. La nuova struttura del dataset invece, prevede l'aggiunta di due lag temporali per tutte le variabili escluso '*Year*' (Figura 18) cercando di colmare il bias introdotto. Con questo nuovo set di dati sono stati ripetuti gli esperimenti **1**, **2** e **3** con le medesime procedure di addestramento dei modelli. Risultati consultabili in Appendice [6][6.1][6.2].

1-10 2464 rows × 27 columns CSV

Country	Crisis t_...	Crisis t_lag1	Crisis t_lag2	Target	gdp_growth_...	gdp_growth_lag1	gdp_growth_lag2	defl_gdp_growth_lag0	defl_gdp_growth
USA	0.0	0.0	0.0	0.0	0.594837	0.688355	0.573435	0.005525	0
USA	0.0	0.0	0.0	1.0	0.571694	0.593963	0.650893	0.005337	0
USA	1.0	0.0	0.0	0.0	0.570080	0.570854	0.561639	0.005422	0
USA	0.0	1.0	0.0	0.0	0.593744	0.569243	0.539787	0.005573	0
USA	0.0	0.0	1.0	0.0	0.577064	0.592872	0.538263	0.005634	0
USA	0.0	0.0	0.0	0.0	0.520380	0.576215	0.560607	0.005604	0
USA	0.0	0.0	0.0	0.0	0.455020	0.519615	0.544857	0.005545	0
USA	0.0	0.0	0.0	0.0	0.573069	0.454351	0.491337	0.005378	0
USA	0.0	0.0	0.0	0.0	0.547107	0.572227	0.429624	0.005394	0

Figura 2.21 schema del dataset in Python

Esperimento 7

Nel settimo esperimento, è stata valutata l'efficacia delle reti neurali ricorrenti LSTM per l'analisi di dati temporali, una scelta particolarmente idonea in questo contesto. Come nel sesto esperimento, è stata adottata una diversa struttura dei dati, considerando solo le variabili normalizzate al tempo t. Di conseguenza, il numero totale di righe è stato ridotto a 2951, con un totale di 8 colonne, e queste sono state suddivise in un set di allenamento e un set di test, con una proporzione del 75% nel primo e il 25% nel secondo. Per adattare i dati all'input del modello LSTM, è stato necessario effettuare un ridimensionamento in una forma tridimensionale (2951, dimensione finestra, 8), in cui la "dimensione finestra" rappresenta un iperparametro che verrà definito in seguito. Questa rappresentazione tiene conto della finestra di informazioni storiche fornita come input alle celle LSTM, un aspetto cruciale nell'analisi di serie temporali. È stato utilizzato un approccio di cross-validation specifico per dati sequenziali, noto come TimeSeriesSplit. A differenza del k-fold validation, il TimeSeriesSplit rispetta l'ordine temporale dei dati, suddividendo la serie in blocchi sequenziali. Questo è fondamentale quando si desidera testare un modello su dati futuri, simulando il comportamento del mondo reale. Il parametro ‘tscv’ rappresenta il numero di blocchi in cui i dati sono stati divisi ed è anch'esso un iperparametro poiché la scelta del numero di blocchi può influenzare le prestazioni del modello. Prima di procedere con la fase di addestramento del modello LSTM, sono state impostate alcune variabili chiave. La funzione di attivazione per gli strati nascosti è stata selezionata come Rectified Linear Unit (ReLU), mentre per l'ultimo strato è stata utilizzata la funzione di attivazione sigmoidale. La funzione di costo scelta è la “binary cross-entropy”, considerata appropriata per il contesto di classificazione. La metrica di valutazione del modello più comunemente utilizzata è l'accuracy, ma sono stati effettuati tentativi anche con la F1-score. Successivamente, attraverso cicli for, sono stati individuati i valori ottimali per gli iperparametri rimanenti, valutando le performance in base ai punteggi di F-measure ottenuti per ciascuna

combinazione di parametri. Una volta individuata la combinazione ottimale, è stata calcolata la matrice di confusione come indicatore per il confronto con gli altri esperimenti. La matrice di confusione offre una chiara rappresentazione delle prestazioni del modello nella classificazione, consentendo una valutazione dettagliata dei risultati ottenuti. Il tutto è stato svolto in codice Python.

Capitolo 3

Risultati

In questo capitolo vengono illustrati i risultati degli esperimenti in termini di valori dei parametri, metriche e matrici di confusione per tutti i modelli addestrati. L'obiettivo primario è di individuare il modello che meglio riesce a prevedere le due classi 'No Crisi' e 'Crisi', avendo un occhio di riguardo per quest'ultima. Per cui la scelta dei valori degli iperparametri in tutti gli esperimenti, ha seguito la seguente scala gerarchica d'importanza: 1) F Measure, 2) Recall, 3) Accuracy; tenendo sempre in considerazione i valori delle variazioni di ciascuno. A parità di F Measure sono stati preferiti i modelli più parsimoniosi.

Un elemento importante di customizzazione per il processo di sotto campionamento con il K-Means, rispetto al lavoro di Rocchi (2022) dove è stato impiegato il numero di cluster k = 2, è stato di utilizzare k = 4 poiché in alcune situazioni la capacità previsionale del modello si è rivelata migliore. Pertanto, è stato scelto come valore comune per tutti gli esperimenti che prevedono un undersampling. Un ulteriore valore custom impostato per la fase di training è il numero di iterazioni dell'algoritmo MetaCost che dal valore default di 10 è stato fissato a 200. Questo ha comportato un incremento di 20 volte il numero delle iterazioni, che da un lato aumenta la precisione e la stabilità del modello, dall'altro richiede maggiori risorse computazionale, che si traduce in tempi di addestramento più lunghi. La scelta di 200 iterazioni deriva dalle prove effettuate sulla prestazione dei modelli, si è visto che un valore superiore a questo fa incrementare notevolmente i tempi di addestramento a fronte di prestazione pressoché invariate.

Il parametro principale da dover settare per il MetaCost è la matrice dei costi, nella Tabella 7 vengono riportate le combinazioni provate per i singoli esperimenti. Considerando il dispendio computazionale richiesto per effettuare la ricerca degli iperparametri, sono state valutate caso per

caso in base ai risultati delle precedenti se le combinazioni di costo potessero migliorare le performance predittive oppure no, andandole eventualmente ad escludere. Per questo motivo non compaiono sempre tutte le combinazioni nei diversi esperimenti riportati nelle "*Tabella delle combinazioni*" in Appendice.

Matrice di costo	$\begin{bmatrix} 0 & a \\ b & 0 \end{bmatrix}$
a costo per misclassification della classe 1	[0.95; 0.9; 0.85; 0.8; 0.75; 0.7; 0.65; 0.6; 0.55; 0.5]
b costo per misclassification della classe 0	[0.05; 0.1; 0.15; 0.2; 0.25; 0.3; 0.35; 0.4; 0.45; 0.5]
numero di iterazioni	200

Tabella 7

Le figure mostrate negli esperimenti riportano i valori degli indicatori: Recall, Precision e F Measure; e si riferiscono alla classe minoritaria 'Crisi'. Lo schema con il quale è stata costruita la matrice di confusione è presente nell'Appendice [Confusion Matrix].

Esperimento 1

Come già anticipato nel capitolo precedente, nel primo esperimento è stato usato il MetaCost nella fase di training, per tener conto dello squilibrio tra le classi. Tenendo in considerazione uno sbilanciamento del 93,80% verso la classe maggioritaria, 2709 elementi per la classe 'No Crisi' (0) e 179 per quella 'Crisi' (1). Si può osservare il confronto tra le performance dei modelli migliori ottenuti con il campionamento e il MetaCost per ciascun classificatore.

Decision Tree

Numero totale di combinazioni provate è di 264 ed i valori dei parametri migliori si possono consultare in Appendice [1]. Dalla Figura 3.1.1 si evince un miglioramento complessivo del modello in particolare la Recall è incrementata di 2,24%. Mediamente 4 elementi in più vengono classificati come veri negativi dal MetaCost rispetto al Sampling.

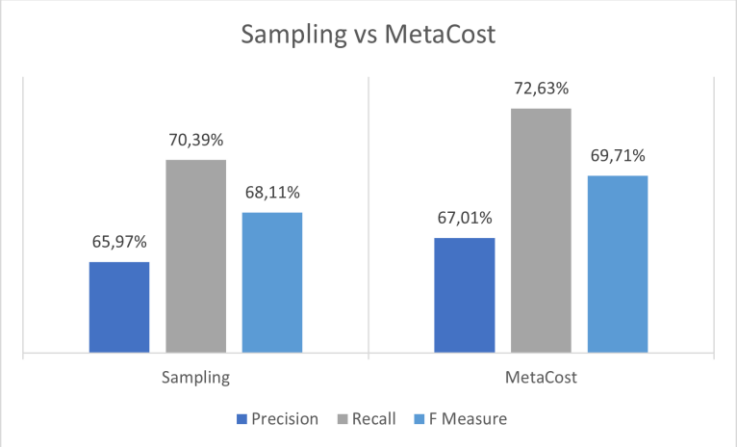

Figura 3.1.1

Random Forest

Per la RF il numero totale di combinazioni provate è stato di 864 e le migliori sei sono riportate in Appendice [1]. Rispetto alla tecnica di Sampling, il MetaCost non sembra poter far la differenza ottenendo pressoché gli stessi risultati in termini di predizione.

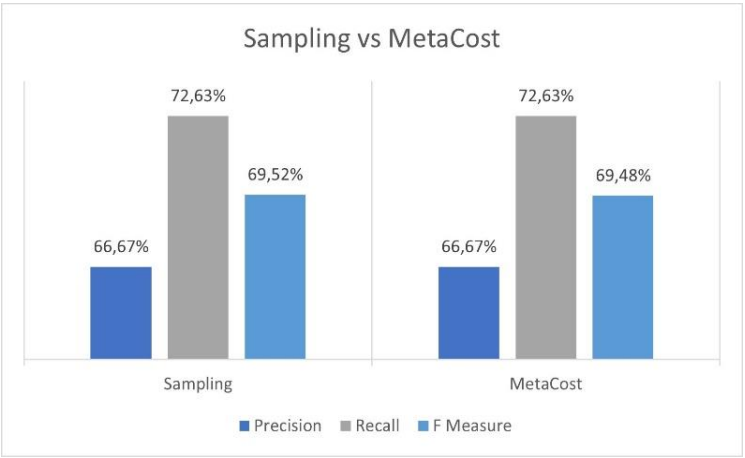

Figura 3.1.2

Logistic Regression

L'algoritmo di regressione logistica è decisamente più semplice e con meno parametri per cui le combinazioni totali provate sono state 30. In questo caso il MetaCost sembra performare meglio dell'approccio Sampling, registrando una Recall più bassa ma complessivamente un F Measure più elevata di 6%.

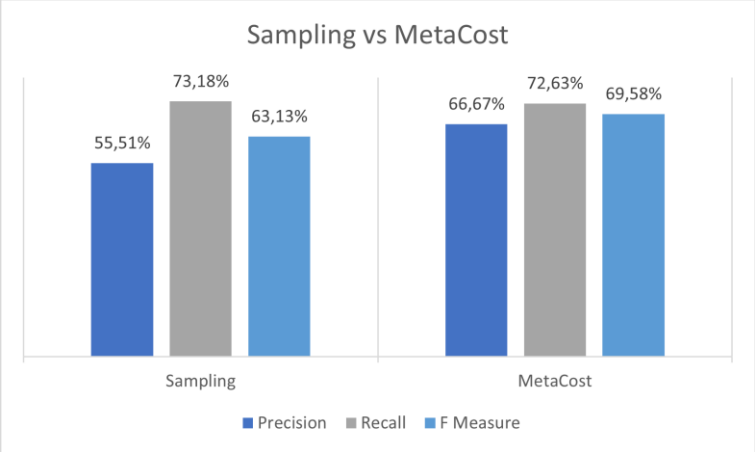

Figura 3.1.3

SVM Linear

La SVM con kernel lineare è quella più semplice tra le Support Vector Machine, per cui anche dal punto di vista del numero dei parametri/combinazioni e tempo di esecuzione è meno dispendiosa rispetto ad una con il kernel polinomiale. Pertanto, il numero di combinazioni totali provate è stato di 54 e le migliori sei sono consultabili in Appendice [1]. Il kernel lineare sembra comportarsi meglio con il *cost sensitive learning* piuttosto che con *sampling*, riportando incrementi per tutte e tre le metriche di confronto.

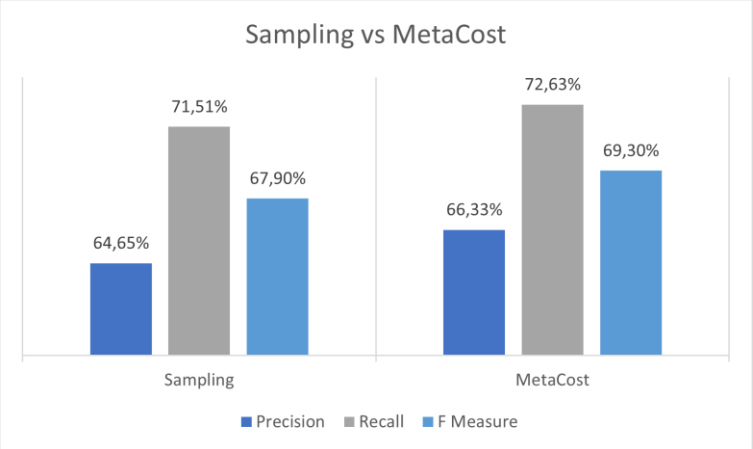

Figura 3.1.4

SVM Radial

Il kernel radiale (RBF) ha un parametro in più, quello dello "spazio di somiglianza" *gamma*, che porta il numero delle combinazioni tra cui selezionare la migliore a 540. Così come si è verificato con la logistica, anche in questa situazione il MetaCost è meno performante per quanto riguarda la Recall ma complessivamente costruisce un modello più performante.

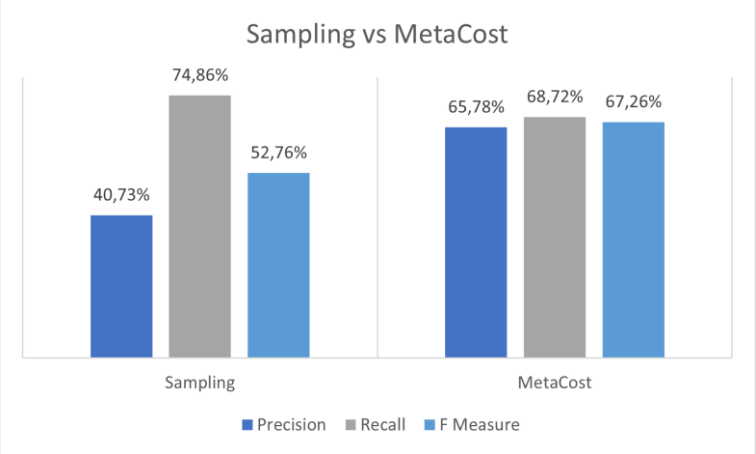

Figura 3.1.5

SVM Polynomial

Nel caso di kernel polinomiale si cerca di incrementare la dimensione dello spazio delle features per cercare un separatore lineare, per cui si ha bisogno di definire il grado del polinomio del nuovo spazio. Questo rende la Polynomial più lenta delle altre SVM anche se con un numero totale di combinazioni pari a 270. Anche in questa situazione il MetaCost si comporta meglio del Sampling costruendo un modello più stabile nella previsione.

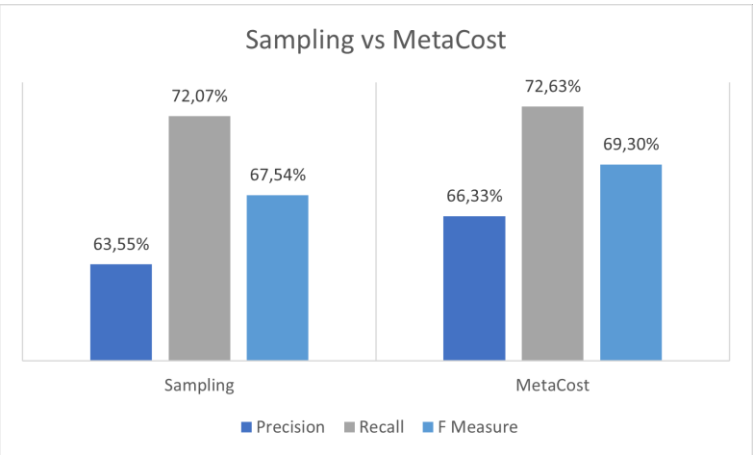

Figura 3.1.6

Da questo primo esperimento si può estrapolare che il MetaCost performa in tutti i casi, tranne che per la Random Forest, meglio della tecnica di Sampling sperimentata. Nella Figura 3.1.7 si possono osservare le prestazioni medie in comparazione. Si nota che seppur una leggera riduzione per le Recall, i modelli hanno registrando valori medi di F Measure del 69,11%, circa il 5% in più rispetto ai modelli addestrati con la tecnica del Sampling. Mentre la matrice dei costi è piuttosto sbilanciata, ottenendo i risultati migliori in situazioni di (08 – 0.2) e (0.7 – 0.3).

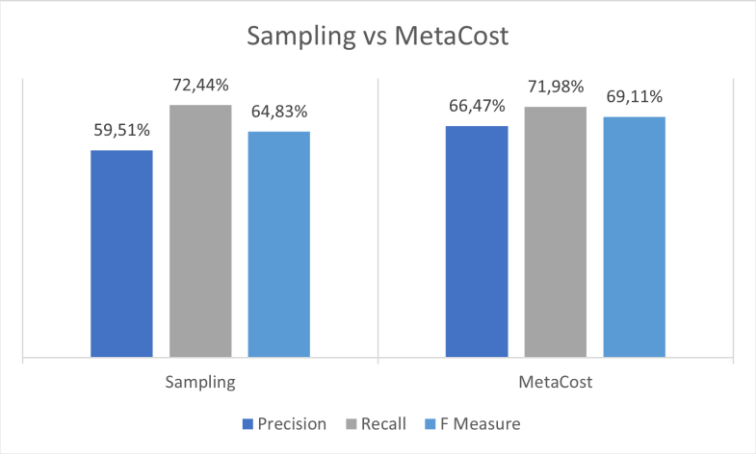

Figura 3.1.7

La ragione per cui i modelli costruiti con il MetaCost risultano essere migliori di quelli addestrati con le tecniche di Sampling potrebbe essere dovuta a diversi fattoti: il meta-algoritmo è costruito propriamente per gestire dati sbilanciati per cui penalizzando gli errori commessi nel classificare la classe minoritaria spinge l'algoritmo a comprendere meglio i dati. Nella fase di addestramento il modello cerca di minimizzare la funzione di costo, che tiene conto dei costi specifici associati a ciascuna classe. Questo influenza la decision boundary del modello durante l'addestramento avvicinandolo verso la classe minoritaria. Invece, la tecnica di Sampling incide solamente sulla numerosità delle classi, introducendo del rumore che non giova alla classificazione perché vengono modificate le distribuzioni erratamente.

Esperimento 2

Nel secondo esperimento si è provato ad unire il meglio delle due tecniche messe a confronto nell'Esperimento 1, cercando di ottenere un quasi bilanciamento tra le classi eliminando lo SMOTE e facendo un Undersampling del 25%. Ottenendo uno sbilanciamento del 79% distribuito su 677 elementi di classe 'No Crisi' e 179 di classe 'Crisi'.

Il numero totale di combinazioni provate per ciascuna tecnica è uguale all'esperimento precedente per cui non è stato riportato.

Decision Tree

Il modello costruito con l'approccio ibrido risulta più performante per ciascuna metrica rispetto al semplice Sampling.

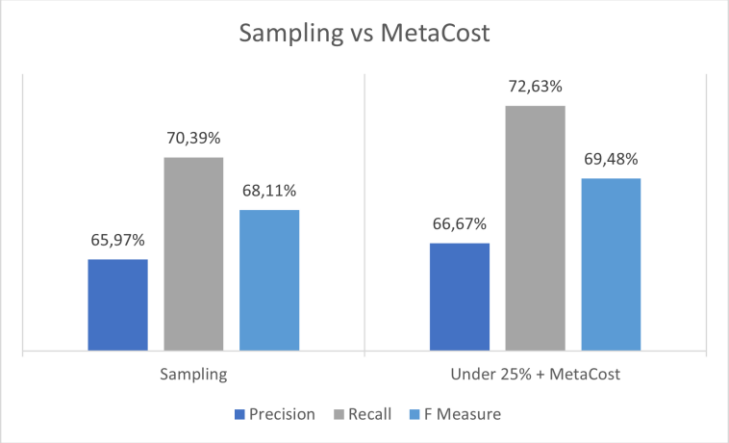

Figura 3.2.1

Random Forest

Con questa combinazione si riottengono gli stessi risultati che con le tecniche di campionamento, come lo si può notare nella Figura 3.2.2.

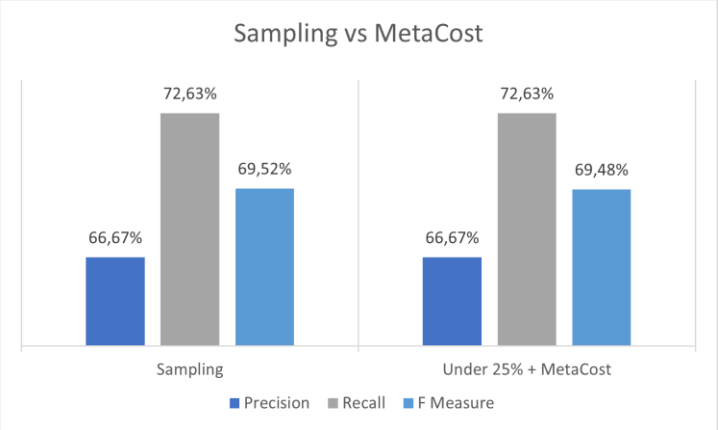

Figura 3.2.2

Logistic Regression

Nel caso della logistica, la tecnica ibrida sperimentata risulta essere migliore, seppur perdendo un po' di capacità nel prevedere la classe minoritaria.

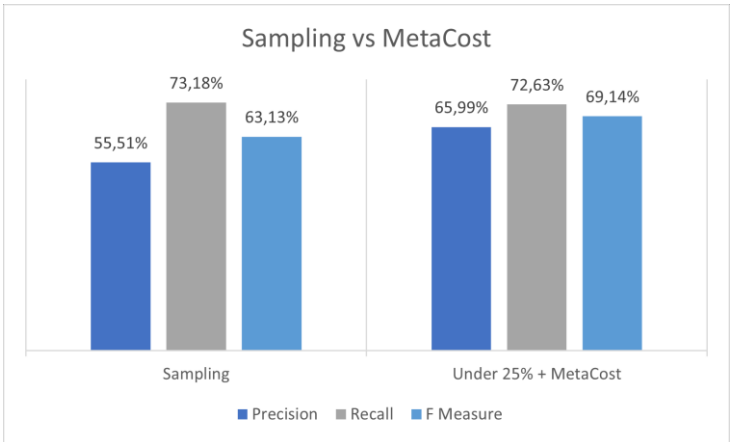

Figura 3.2.3

SVM Linear

Nel caso della SVM lineare, un Undersampling del 25% della maggioritaria e il successivo uso del MetaCost, risulta più efficace che usare lo SMOTE per ridurre l'effetto dello sbilanciamento dei dati.

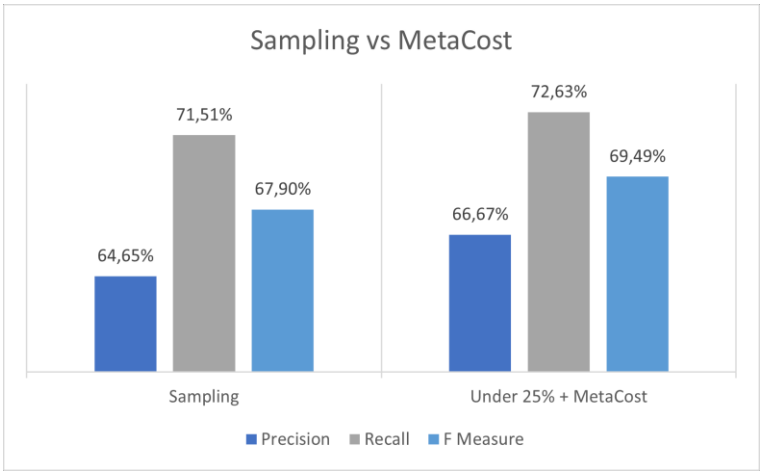

Figura 3.2.4

SVM Radial

Con il kernel radiale si perde in Recall ma il modello complessivamente risulta migliorato come si può notare nella Figura 3.2.5.

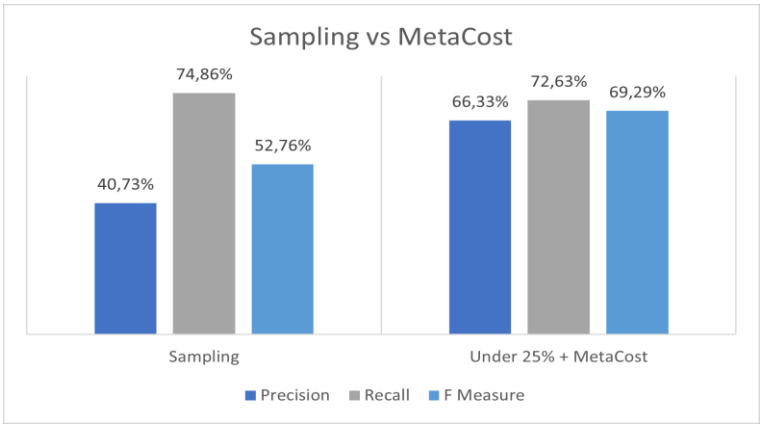

Figura 3.2.5

SVM Polynomial

Invece il kernel polinomiale nonostante sia quello con il quale si registrano prestazioni peggiori tra tutte le SVM, si nota un leggero miglioramento rispetto alla situazione di confronto.

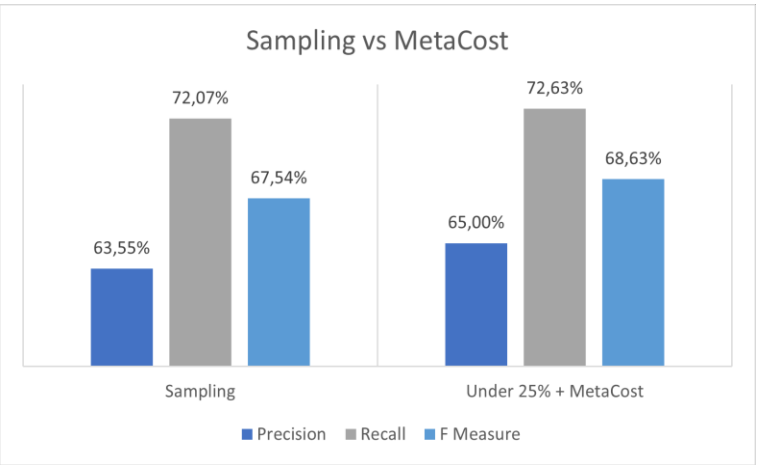

Figura 3.2.6

In conclusione, come si può notare dalla Figura 3.2.7 nella media con l'approccio ibrido si migliora complessivamente in tutte le tecniche provate raggiungendo una F Measure del 69,25%. Confermando che l'uso del MetaCost rende più affidabili i modelli ma ha mostrato rendimenti di poco inferiore rispetto all'esperimento attuale. Vista la riduzione dello squilibrio tra il numero di elementi delle classi, la matrice dei costi selezionati hanno valori inferiori rispetto al caso precedente. Infatti, i risultati migliori si ottengono con pesi (0.7 – 0.3), (0.6- 0.4) e addirittura anche (0.55 – 0.45) per la RF e la Logistica.

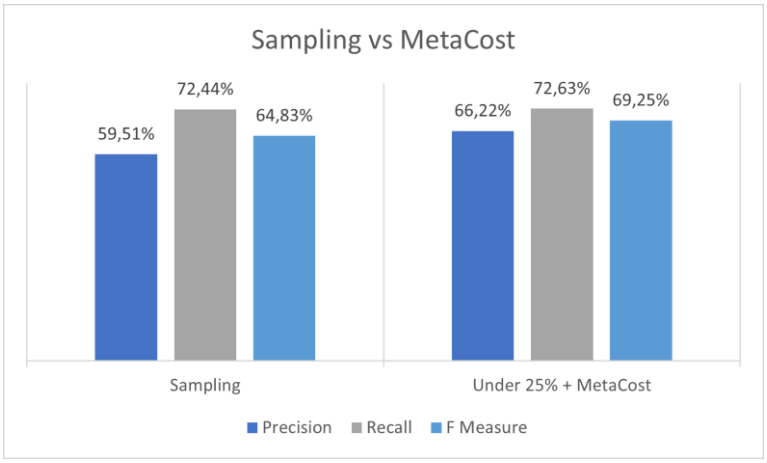

Figura 3.2.7

Esperimento 3

Nel terzo esperimento, attraverso un sotto campionamento della maggioritaria del 15% rispetto al numero totale di osservazioni, si è tentato di ridurre ulteriormente il gap rispetto alla classe minoritaria. Ottenendo uno sbilanciamento dello 69,4% distribuito fra i 406 elementi della classe 'No Crisi' ed i soliti 179 della classe 'Crisi'. Anche per questo esperimento il numero totale delle combinazioni testate per ciascuna tecnica è uguale al primo esperimento per cui non sono state riportate.

Decision Tree

Anche con un sotto campionamento del 15% il modello ottenuto risulta essere più performante di quello ottenuto con il bilanciamento.

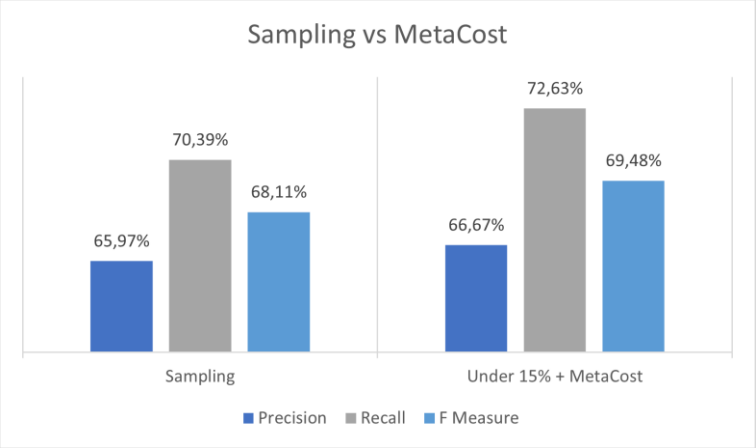

Figura 3.3.1

Random Forest

Non si può dire la stessa cosa per la RF, la quale già con l'approccio Sampling ha raggiunto risultati notevoli e con l'Under del 15% li conferma.

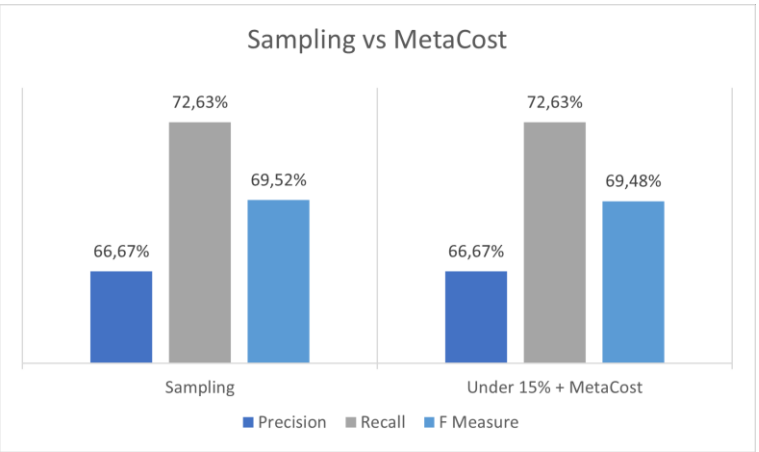

Figura 3.3.2

Logistic Regression

Nonostante in termine di sensitivity il modello Sampling avesse registrato valori buoni confrontandoli con gli altri modelli, non si può dire altrettanto per la precisione. Nel secondo caso, la situazione si conferma pressoché con gli stessi valori per la richiamo, ma migliora decisamente per la Precision.

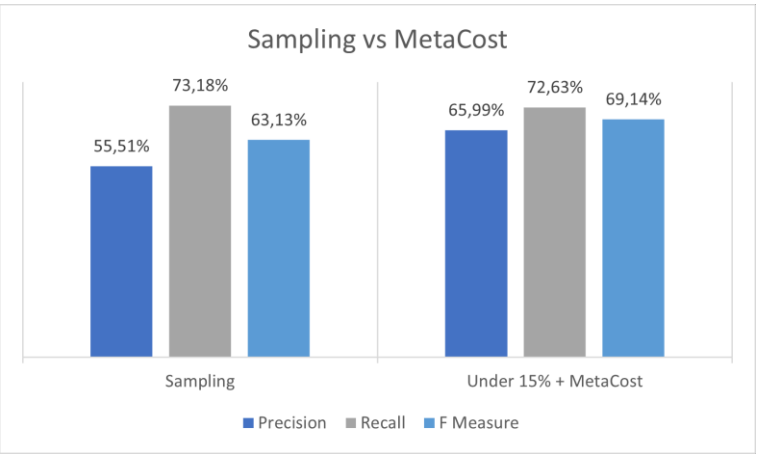

Figura 3.3.3

SVM Linear

Nel caso della SVM lineare un approccio Undersampling del 15% sembri funzionare meglio rispetto al Sampling, come si può notare nella Figura 3.3.4.

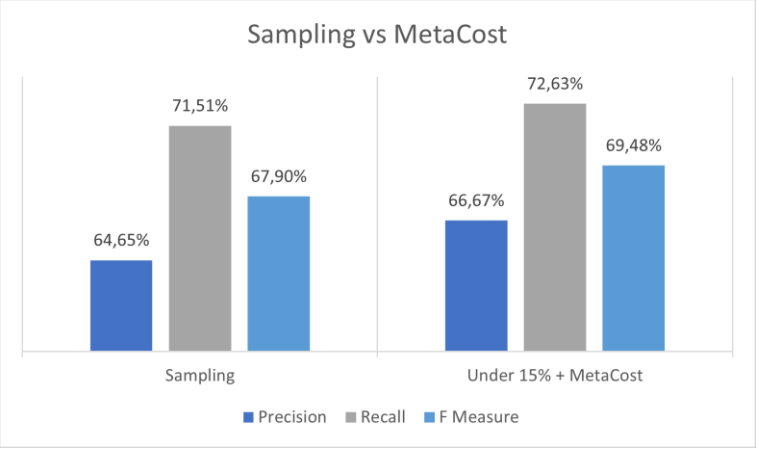

Figura 3.3.4

SVM Radial

Simile a quanto detto per la regressione logistica, nel caso di Sampling il kernel radiale registra la migliore performance in termini di richiamo 74,86%, ma non si può dire altrettanto per la precisione. Mentre con un approccio Under al 15%, il modello perde in termini di Recall, ma migliora complessivamente.

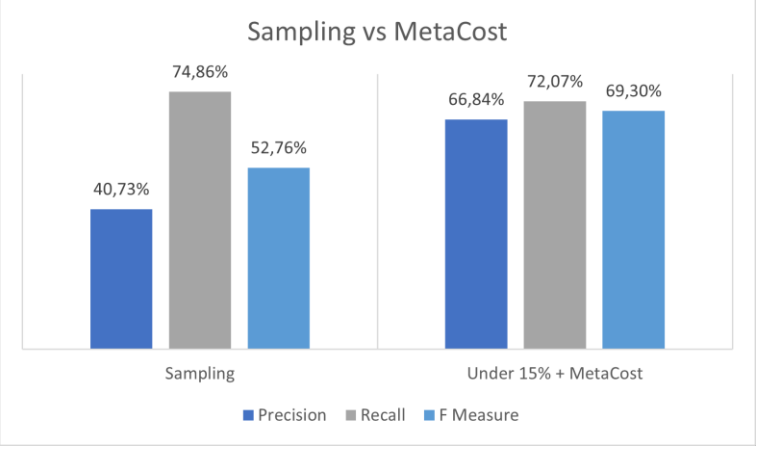

Figura 3.3.5

SVM Polynomial

Il kernel polinomiale risulta essere quello meno performante fra tutti i kernel provati e seppur ottenendo una piccola miglioria in termini di sensitivity, peggiora invece, per quanto riguarda la precisione.

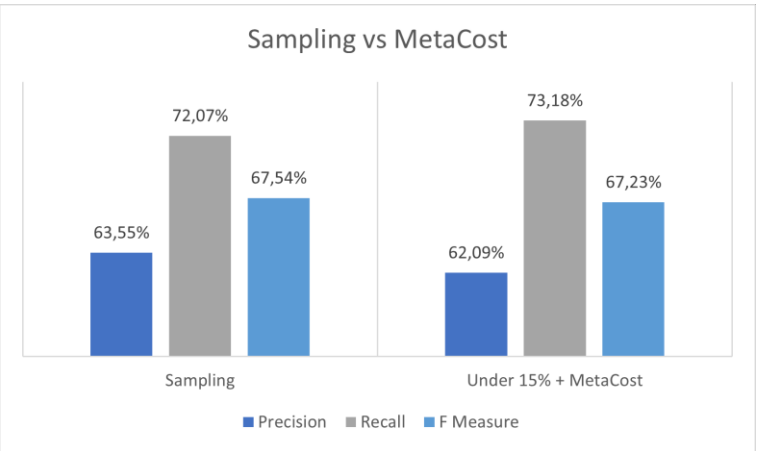

Figura 3.3.6

In sintesi, nella metà dei modelli si è verificato un leggero miglioramento della Recall, contemporaneamente sono stati ottenuti progressi importanti in termini di precisone nell'ordine del 6%. In altri termini, mediamente 10 elementi in più vengono classificati correttamente come classe 1 anziché 0. Tutto questo in un contesto sbilanciato al 69,4% e un campione ridotto composto da solamente 585 osservazioni. L'effetto del MetaCost in questa situazione sembra più mitigato poiché usa matrici di costo (0.55 – 0.45) in tre casi su sei.

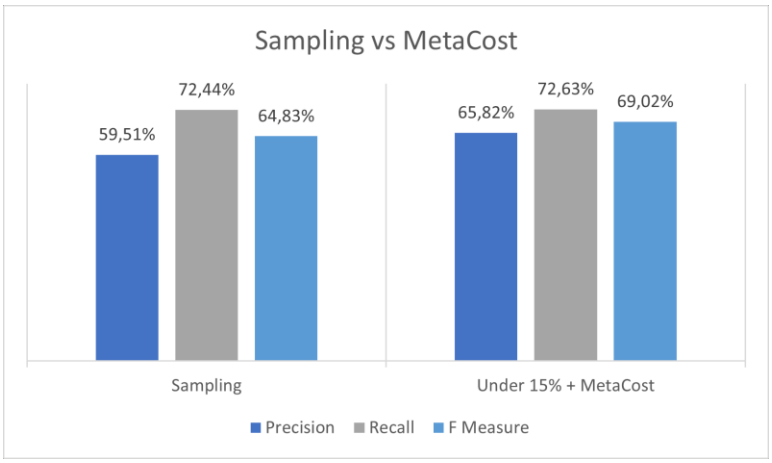

Figura 3.3.7

Esperimento 4

Nel quarto esperimento è stato provato un cluster dei dati come descritto da Caggiano (2016), secondo la media della crisi registrata da ciascun paese si ottengono tre gruppi. Con la suddivisione in gruppi, si cerca di individuare i paesi simili in modo da costruirvi dei modelli che si possano adattare meglio alle loro caratteristiche, migliorando così le previsioni di crisi o non crisi.

I tre gruppi formati hanno dimensioni e tasso di equilibrio tra le classi diversi, per cui anche i risultati e i parametri dei modelli stessi sono diversi. Nel caso del cluster 'sotto', è composto da 2085 elementi di cui 2036 di classe 0 e 38 di classe 1, complessivamente lo sbilanciamento è di 97,65%. Il secondo cluster 'media' è composto da 382 elementi (321 (0) – 61 (1)) ed uno sbilanciamento del 84,03%. Mentre l'ultimo gruppo 'sopra' è composto solamente da 421 elementi, distribuiti in 321 di classe 'No Crisi' e 80 di 'Crisi' ed uno squilibrio di 80,99%. Per questo esperimento vengono riportati anche i parametri che danno vita al modello migliore di ciascun gruppo.

Decision Tree

A partire dalle matrici di confusioni dei singoli cluster ne è stata composta una complessiva riassuntiva e ricalcolate le metriche di valutazione. Nella Figura 3.4.1 si evince che l'approccio proposto, permette di ottenere un modello con performance predittive migliori, rispetto al modello in contrapposizione.

Cluster '*sotto*'

Costi	Criterio	Profondità
0.97 – 0.03	Gini	135

Cluster '*media*'

Costi	Criterio	Profondità
0.8 – 0.2	Gini	46

Cluster '*sopra*'

Costi	Criterio	Profondità
0.5– 0.5	Gain	31

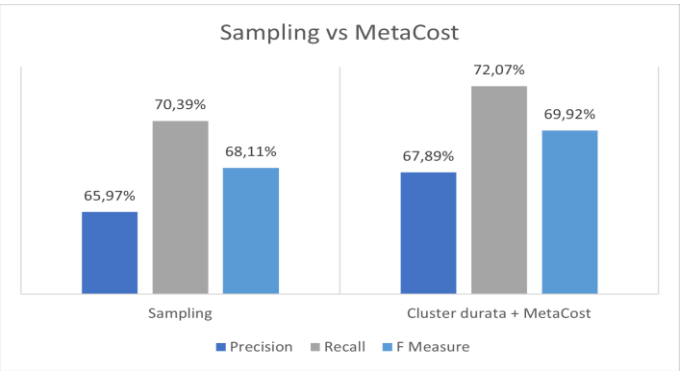

Figura 3.4.1

Dai valori dei parametri si nota che per i gruppi con un forte sbilanciamento, anche la matrice dei costi assume valori lontani tra loro. Al contrario, ad esempio nel cluster '*sopra*' dove le classi sono più paritarie in numero rispetto al gruppo '*sotto*', l'effetto del MetaCost sembra essere superfluo per la fase di addestramento del modello.

Random Forest

Cluster '*sotto*'

Costi	Alberi	Criterio	Profondità
0.55 – 0.45	240	Accuracy	61

Cluster '*media*'

Costi	Alberi	Criterio	Profondità
0.75 – 0.25	90	Accuracy	61

Cluster '*sopra*'

Costi	Alberi	Criterio	Profondità
0.55 – 0.45	400	Information	61

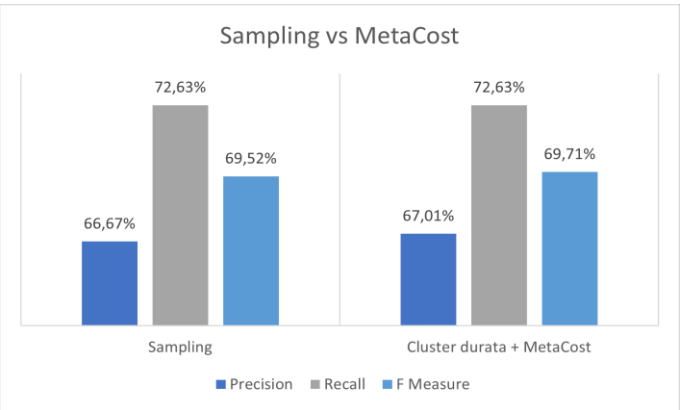

Figura 3.4.2La Random Forest si dimostra essere un algoritmo performante in tutte le situazioni, confermando la sua capacità di catturare la struttura dei dati in maniera pressoché equiparabile in entrambi i casi riportati nella Figura 3.4.2.

Logistic Regression

Cluster '*sotto*'

Costi	Criterio
0.7 – 0.3	COORDINATE_DESCENT NAIVE

Cluster '*media*'

Costi	Criterio
0.8 – 0.2	IRLS

Cluster '*sopra*'

Costi	Criterio
0.6 – 0.4	L_BFGS

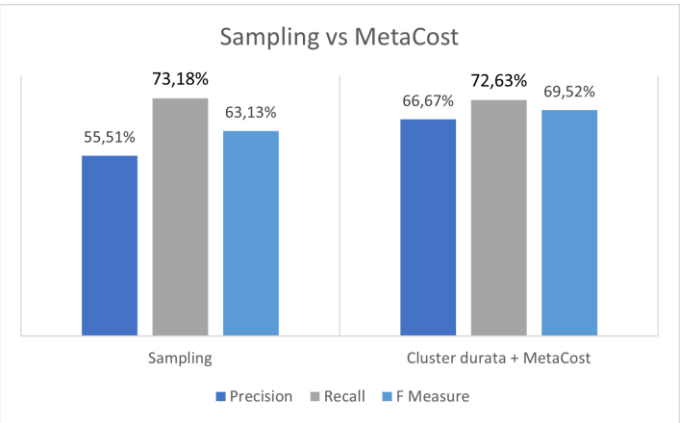

Figura 3.4.3

La regressione logistica si comporta molto bene nell'individuare la classe minoritaria, individuando oltre il 70% dei casi di 'Crisi' in ambe due le situazioni. Ma il secondo metodo permette di incrementare notevolmente la percentuale di Precision, riducendo quindi il numero di falsi positivi

(falso allarme di crisi). Mediamente sono 21 gli elementi che vengono correttamente classificati come 'No Crisi'.

SVM Linear

Cluster '*sotto*'

Costi	C
0.7 – 0.3	0.01

Cluster '*media*'

Costi	C
0.7 – 0.3	0.001

Cluster '*sopra*'

Costi	C
0.55 – 0.45	1

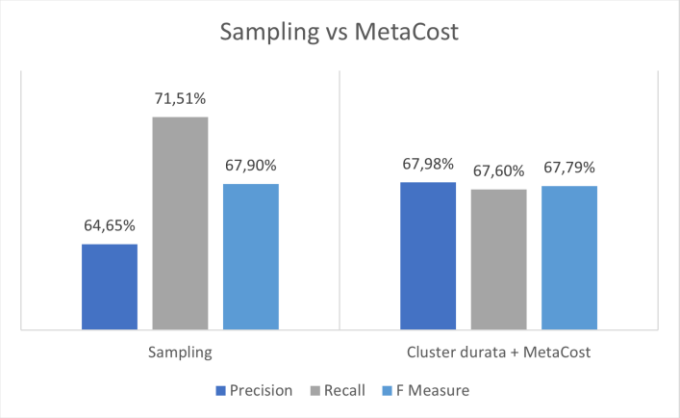

Figura 3.4.4

Il kernel lineare nella situazione di clustering iniziale e la successiva applicazione del MetaCost, non migliora la capacità previsionale complessiva. Quello che succede, lo si può vedere nella Figura 3.4.4, cioè una diminuzione del numero dei falsi positivi, ma un contemporaneo aumento dei falsi negativi. Complessivamente le due situazioni, una migliorativa e l'altra peggiorativa, si controbilanciano ottenendo una prestazione simile a quella al modello Sampling.

SVM Radial

Cluster '*sotto*'

Costi	C	gamma
0.7 – 0.3	0.4	0.2

Cluster '*media*'

Costi	C	gamma
0.55 – 0.45	1	0.1

Cluster '*sopra*'

Costi	C	gamma
0.55 – 0.45	1	0.1

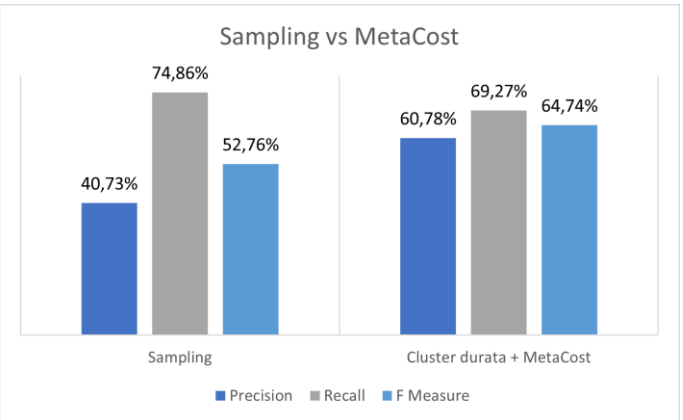

Figura 3.4.5

Con il kernel radiale si è partiti da una prestazione modesta, per cui con il secondo metodo si migliora la prestazione generale, ma non si raggiungono quelle del kernel lineare.

SVM Polynomial

Cluster *'sotto'*

Costi	C	degree
0.6 – 0.4	1000	6

Cluster *'media'*

Costi	C	degree
0.65 – 0.35	0.01	2

Cluster *'sopra'*

Costi	C	degree
0.5 – 0.5	0.01	2

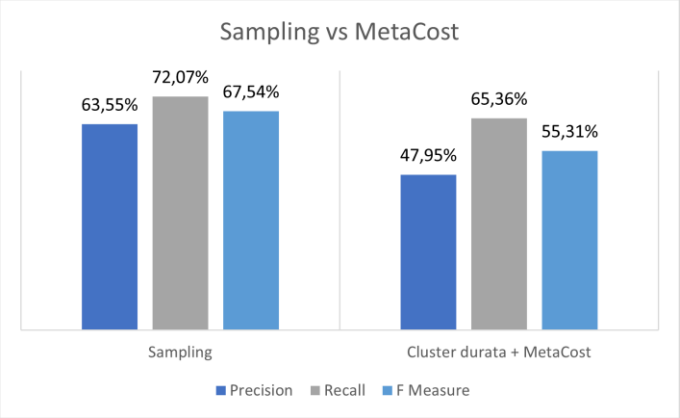

Figura 3.4.6

Invece, il kernel multidimensionale non sembra funzionare meglio, rispetto al modello di confronto per nessun indicatore, è anche il meno performante tra gli SVM.

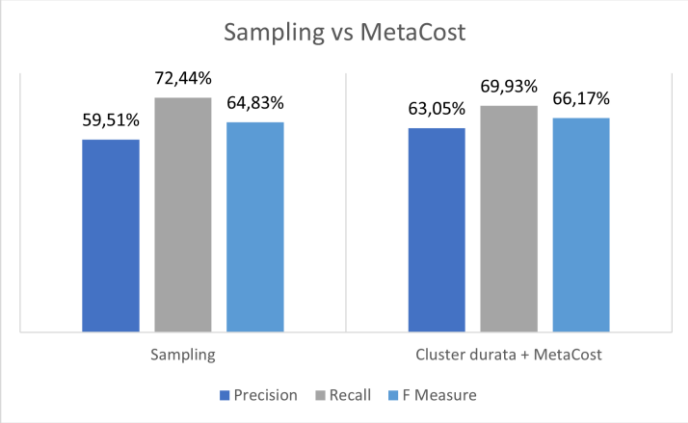

Figura 3.4.7

Concludendo, gli algoritmi SVM in questo esperimento sembrano performare meglio del caso bilanciato 50%-50%, ma non si può dire altrettanto rispetto agli Esperimenti 1,2 e 3. Contrariamente, la *logistica* così come *il decision tree* e la *random forest,* si confermano funzionare molto bene anche con un clustering iniziale, registrando performance in linea con i primi tre esperimenti.

Esperimento 5

Con il quinto esperimento si prova un confronto tra il modello con il metodo Sampling ed un clustering mediante K-Means e successivo impiego del MetaCost. Il numero di gruppi ideale, come anticipato nel capitolo precedente, è quattro, ma nella fase di assegnazione dei cluster alle nazioni, diventano solamente tre. Il gruppo numero due viene assorbito dal Cluster 1 e Cluster 3 essendo molto simile a essi, in particolare registra valori di crisi per nazione molto più vicino ai due piuttosto che al Cluster 0. Rinominati come: *Cluster 0* contiene 2679 elementi distribuiti in 2520 di classe 0 e 159 di classe 1; *Cluster 1* contiene 140 elementi distribuiti in 126 di classe 0 e 14 di classe 1 e, *Cluster 2* ne contiene 69 elementi distribuiti in 63 di classe 0 e 6 di classe 1. Lo sbilanciamento dei gruppi ottenuti è di: 94% per il *Cluster* 0; 90% per il *Cluster 1* e di 91,3% per il *Cluster* 2. Come per l'esperimento precedente, a partire dal miglior modello individuato per ciascun gruppo, sono state costituite le matrici di confusione riassuntiva per i sei algoritmi provati. Oltre alle metriche di confronto vengono riportati i valori degli iperparametri per ciascun cluster.

Decision Tree

Cluster 0

Costi	Criterio	Profondità
0.8 – 0.2	Gini	16

Cluster 1

Costi	Criterio	Profondità
0.7 – 0.3	Accuracy	105

Cluster 2

Costi	Criterio	Profondità
0.5 – 0.5	Gini	76

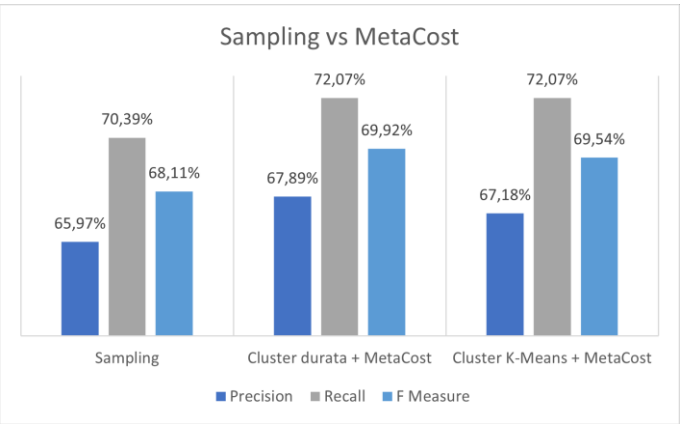

Figura 3.5.1

La "clusterizzazione" con il K-Means sembra funzionare meno rispetto a quella fatta nell'Esperimento 4, anche se la differenza è sottilissima. Rimane il miglioramento complessivo del modello addestrato con il MetaCost, rispetto a quello ottenuto con il bilanciamento delle classi.

Random Forest

Cluster 0

Costi	Alberi	Criterio	Profondità
0.7 – 0.3	240	Information	90

Cluster 1

Costi	Alberi	Criterio	Profondità
0.7 – 0.3	1	Accuracy	61

Cluster 2

Costi	Alberi	Criterio	Profondità
0.7 – 0.3	161	Gain	31

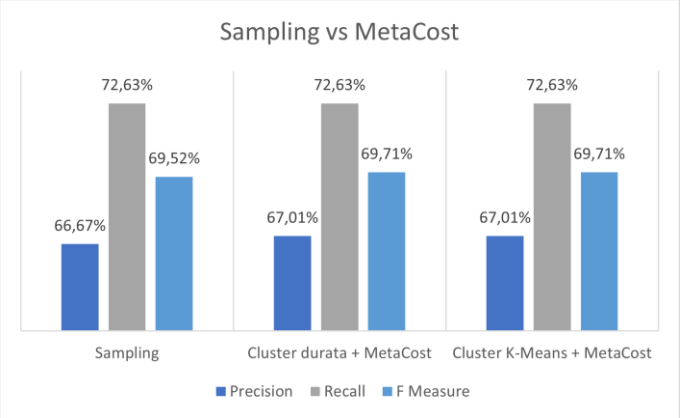

Figura 3.5.2

Anche nel caso della Random Forest si osserva una prestazione pressoché identica rispetto all'Esperimento 4 e al metodo Sampling, come si può notare nella Figura 3.5.2.

Logistic Regression

Cluster 0

Costi	Criterio
0.7 – 0.3	L_BFGS

Cluster 1

Costi	Criterio
0.6 – 0.4	COORDINATE_DESCENT_NAIVE

Cluster 2

Costi	Criterio
0.5 – 0.5	COORDINATE_DESCENT

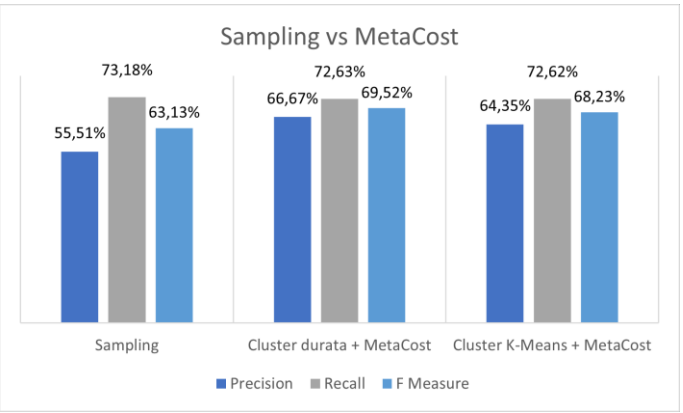

Figura 3.5.3

Nel caso della regressione logistica, la suddivisione in gruppi proposta aiuta a migliorare la classificazione, ma la performance migliore rimane del clustering basato sulla durata della crisi dei singoli paesi.

SVM Linear

Cluster 0

Costi	C
0.6 – 0.4	0.01

Cluster 1

Costi	C
0.7 – 0.3	0.01

Cluster 2

Costi	C
0.55 – 0.45	0.4

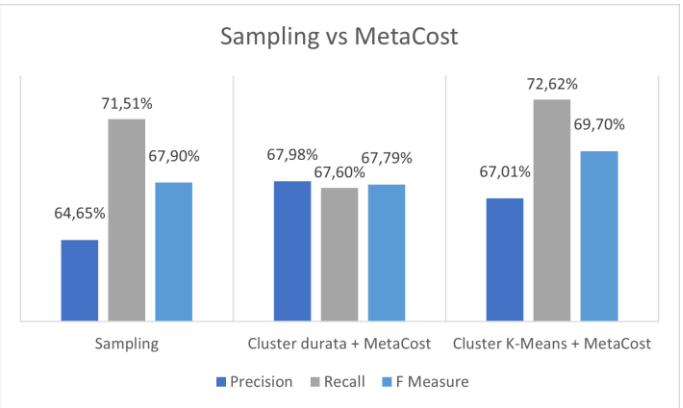

Figura 3.5.4

Nel caso della SVM con kernel lineare si misurano prestazioni migliori rispetto ai modelli messi a confronto nella Figura 3.5.4. In media si classificano 21 falsi crisi in più correttamente come 'No Crisi'.

SVM Radial

Cluster 0

Costi	C	gamma
0.7 – 0.3	0.8	0.001

Cluster 1

Costi	C	gamma
0.65 – 0.35	0.4	0.1

Cluster 2

Costi	C	gamma
0.7 – 0.3	0.1	0.1

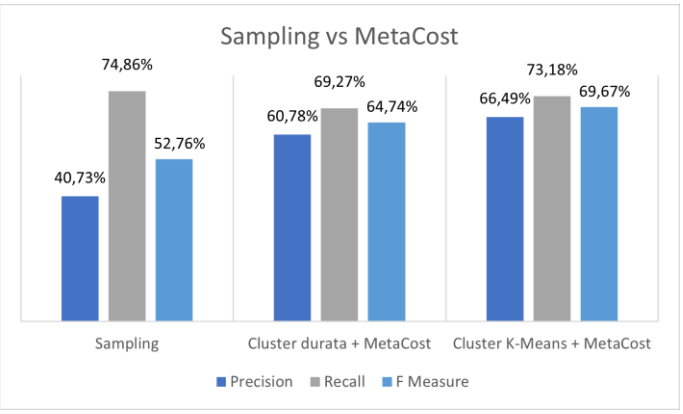

Figura 3.5.5

Il kernel radiale invece, con questa suddivisione dei dati migliora nettamente rispetto al Cluster durata, seppure non raggiunga un valore alto della Recall, come verificatosi nel caso di Sampling.

SVM Polynomial

Cluster 0

Costi	C	degree
0.6 – 0.4	0.001	2

Cluster 1

Costi	C	degree
0.65 – 0.35	0.001	2

Cluster 2

Costi	C	degree
0.6 – 0.4	0.001	2

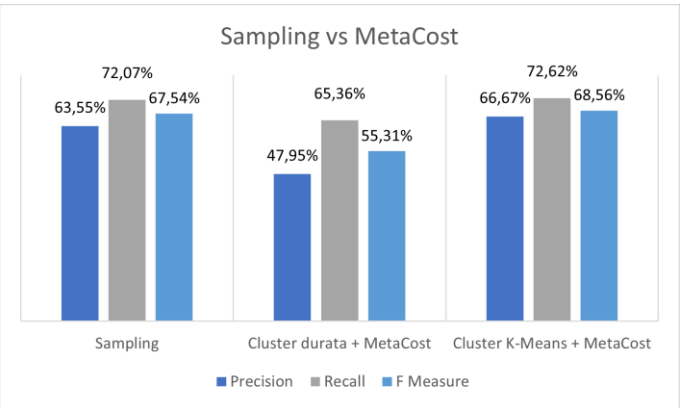

Figura 3.5.6

Per quanto cerne la svm polinomiale con il clustering proposto e applicando il MetaCost, si riesce a pareggiare la richiamata del modello Sampling. Si migliora rispetto all'altra proposta di suddivisione dei dati di oltre 7%.

Concludendo, si può affermare che il clustering dei dati iniziali come proposto aiuta a migliorare la predizione della classe correttamente, soprattutto per gli algoritmi SVM. Mentre per gli altri, si nota una conferma della bontà dei modelli secondo i valori di Precision, Recall e F Measure. Per quanto riguarda la matrice dei costi del Cluster 2, la combinazione migliore corrisponde a costi (0.5 -0.5), che in altri termini significa non penalizzare nessuna classe. In tre casi su sei si è verificata questa situazione, in cui in pratica il MetaCost non ha avuto nessuna influenza sull'addestramento del modello. Dato il numero irrisorio di campioni, l'algoritmo fa fatica a comprendere il pattern che governa i dati, nel Cluster 2 specialmente. Per cui aver ottenuto risultati migliori anche dell'esperimento precedente, nonostante i gruppi avessero all'incirca lo stesso tasso di sbilanciamento, può essere sintomo di overfitting.

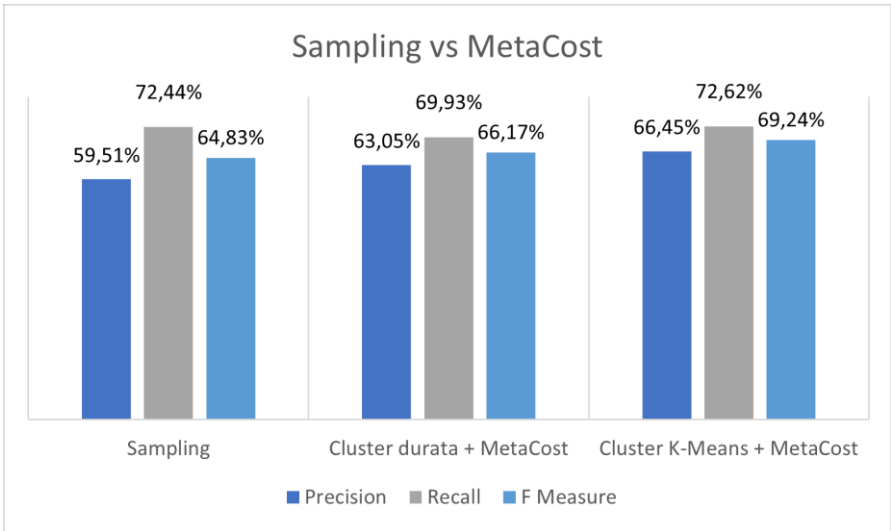

Figura 3.5.7

Esperimento 6

Fino a questo punto è stata utilizzata l'informazione dell'anno corrente per prevedere la variabile target 'Crisi' dell'anno successivo. In questo esperimento sono stati usati i dati di tre anni per prevedere la variabile target ad un anno di distanza, con questa trasformazione però non tutte le 2888 osservazioni hanno uno storico di due lag temporali. Per cui il numero totale di osservazioni è passato a 2464 di cui di 2313 classe 'No Crisi' e 151 di classe 'Crisi'.

6.1 Metacost + lag

In questa sezione vengono riportati i risultati dell'esperimento effettuato con il MetaCost, come algoritmo di cost sensitivity, a differenza dell'Esperimento 1 la struttura del dataset è diversa.

Decision Tree

Nella figura a fianco si nota che questo approccio non raggiunge i risultati aspettati, poiché non c'è nessun incremento delle metriche di performance.

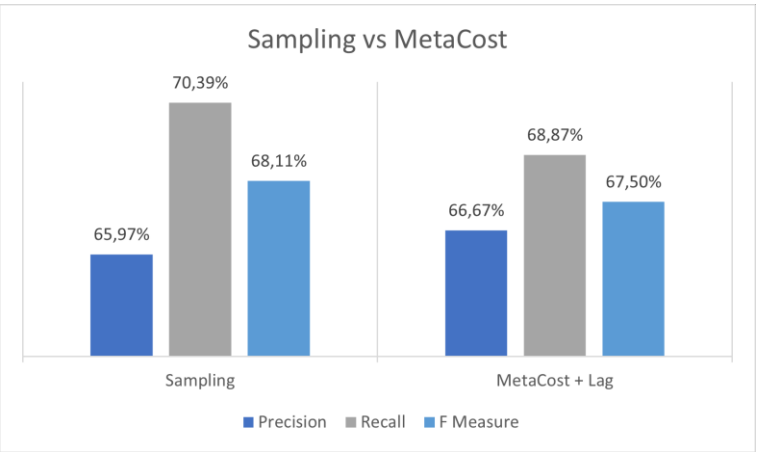

Figura 3.6.1

Random Forest

La Random Forest, che fino ad ora si è dimostrato essere l'algoritmo più performante in quasi tutte le situazioni testate, peggiora la sua capacità predittiva rispetto al modello Sampling.

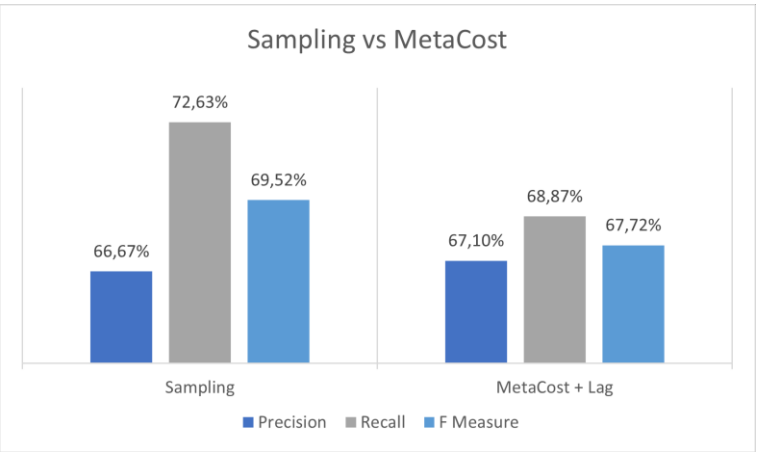

Figura 3.6.2

Logistic Regression

Nella Figura 3.6.3 si nota che nel caso analizzato con il MetaCost "*laggato*" si incrementa il valore della Precision, complessivamente però possiamo dire che è in linea con i risultati ottenuti negli esperimenti precedenti. Questo miglioramento è anche dovuto al basso livello iniziale di performance del modello Sampling.

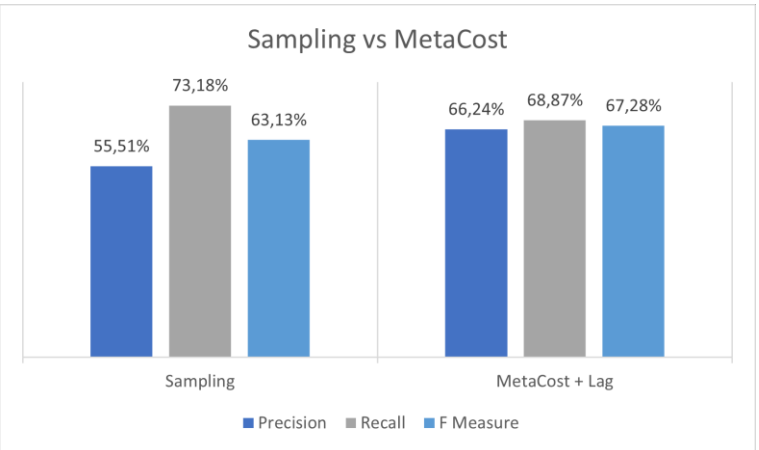

Figura 3.6.3

SVM Linear

L'utilizzo del Support Vector Machine con kernel lineare in sostanza, permette di avere valori di F Measure pressoché uguali per i due modelli. Non indicando una netta preferenza per uno in particolare.

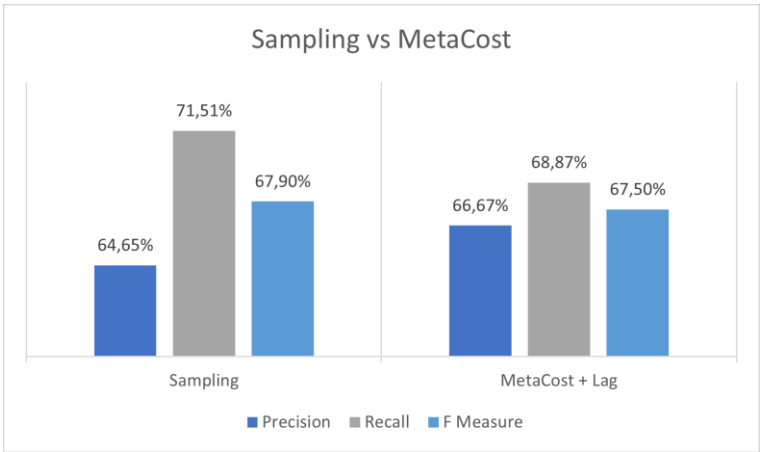

Figura 3.6.4

SVM Radial

Con il kernel radiale si registra un incremento netto della Precision e della F Measure rispetto al primo metodo. Propriamente per il loro valore molto basso, il secondo metodo proposto, seppur riduca la richiamata è un modello migliore.

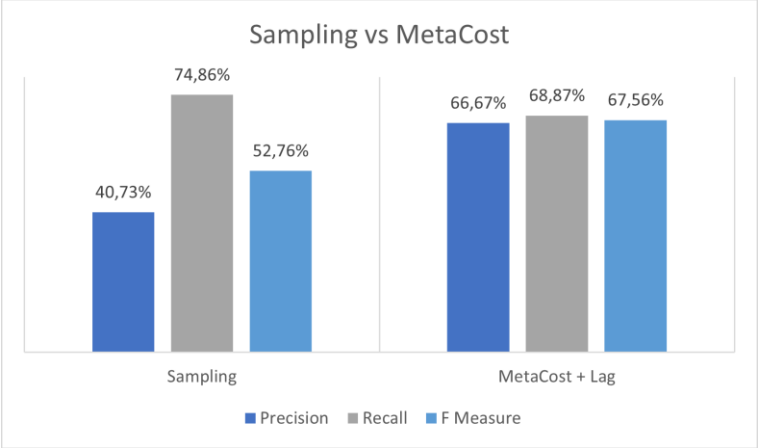

Figura 3.6.5

SVM Polynomial

Situazione analogo alla SVM Linear si verifica con il kernel polinomiale, Figura 3.6.6.

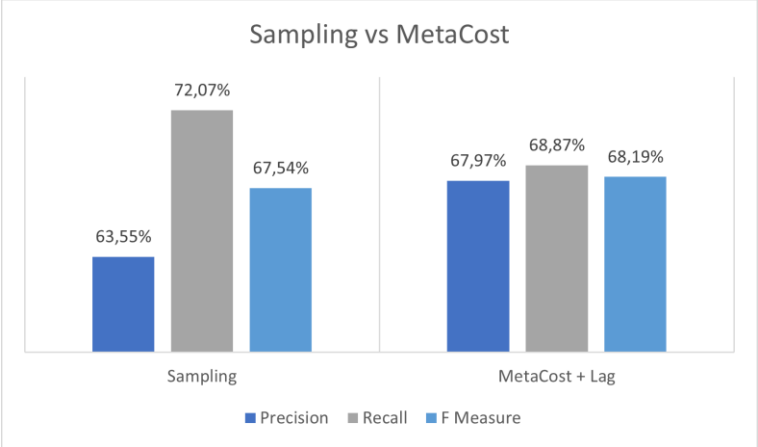

Figura 3.6.6

Mediamente, aggiungendo i lag temporali alle features, rispetto al modello addestrato con il Sampling, il guadagno in precisione essendo maggiore della perdita in sensitività, si ottengono valori di F Measure più elevate. Tuttavia, rispetto agli esperimenti precedenti, non sono così migliorativi. Questo potrebbe essere dovuto al fatto che il numero già non elevatissimo di dati per l'addestramento si sono ridotti ulteriormente mentre il numero delle variabili esplicative sono triplicati. Un segnale possibile per il problema di *curse dimensionality*.

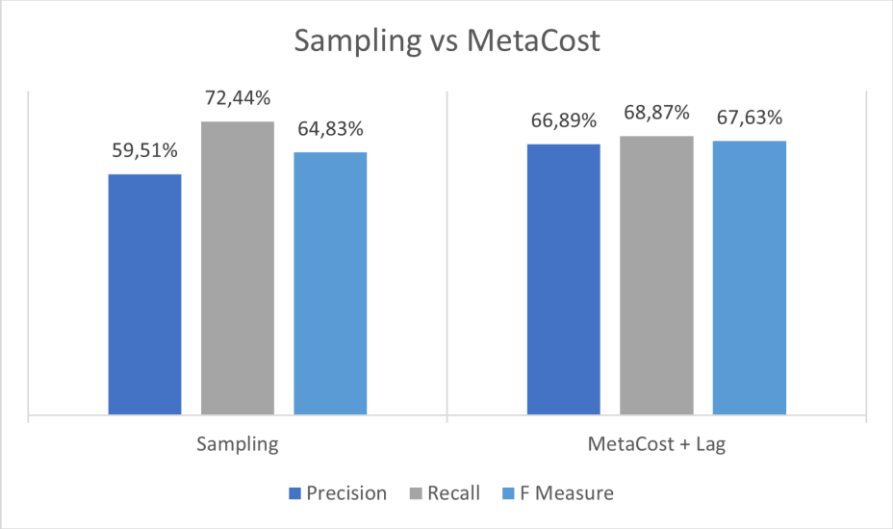

Figura 3.6.17

6.2 Undersampling + Metacost + lag

In questa sezione, a partire dal dataset trasformato sono stati aggiunti i lag temporali alle 7 variabili iniziali, sono stati ripetuti gli Esperimenti 2 e 3 effettuando un sottocampionamento della classe maggioritaria con due diverse percentuali 0,15 e 0,25. Il rapporto di equilibrio tra le classi è di 100:230 per Under 15% e di 100:383 per Under 25%.

Decision Tree

Dal grafico a fianco si nota che il modello con sottocampionamento del 15% seppur di poco, risulta essere migliore di quello a 25%. Ma entrambi sottoperformano rispetto al modello costruito con il Sampling.

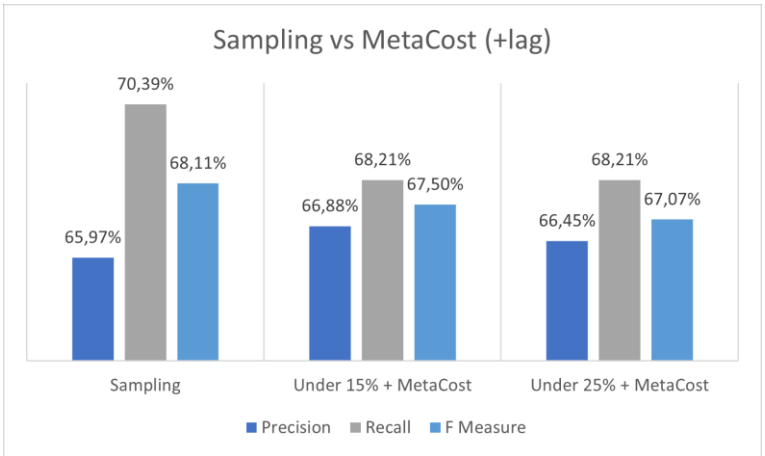

Figura 3.6.21

Random Forest

Per la random forest non si segnala una differenza tra i due modelli Under, ma in media si classificano erroneamente circa otto elementi in più della classe minoritaria rispetto al primo modello.

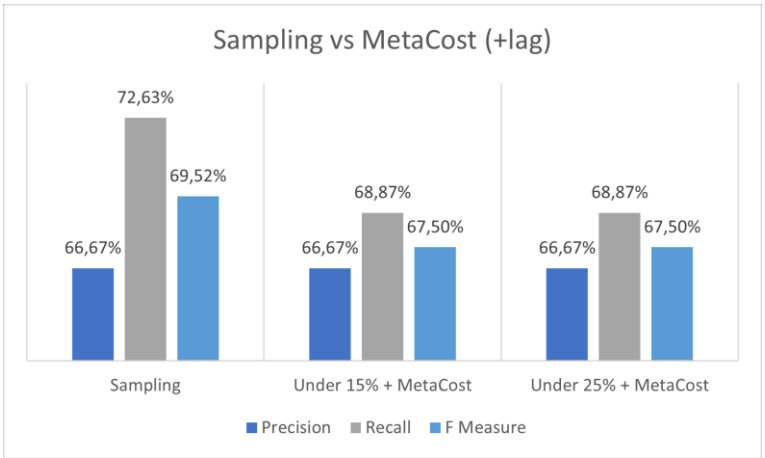

Figura 3.6.22

Logistic Regression

L'addestramento con la logistica, in particolare per l'undersampling del 25% registra una F Measure più elevata rispetto agli altri due modelli. Segnando una sofferenza più contenuta alla riduzione del numero delle osservazioni.

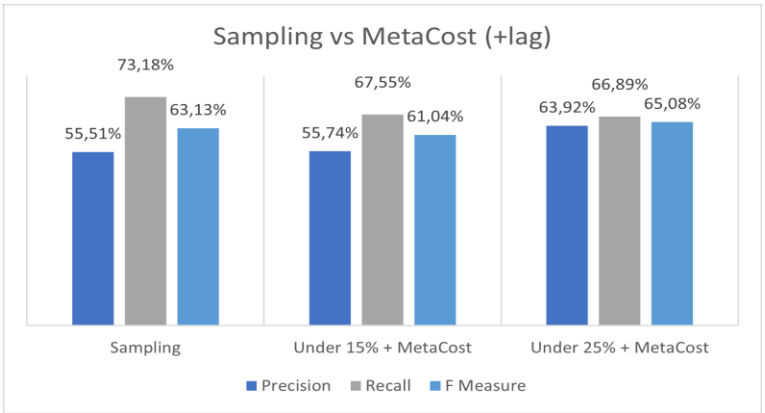

Figura 3.6.23

SVM Linear

La svm lineare sottoperforma in entrambi i casi di sotto campionamento rispetto al caso di confronto, nonostante l'incremento del valore della Precision.

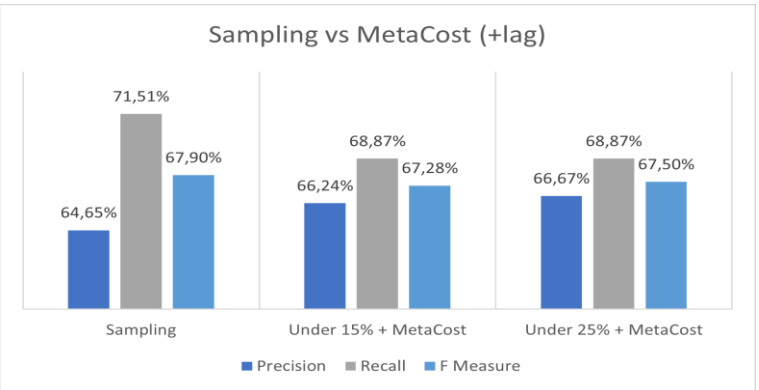

Figura 3.6.24

SVM Radial

Il kernel radiale mostra un particolare appiattimento tra i valori delle metriche di precisione e di sensitività del modello. Risulta migliore rispetto al primo, ma non in termini assoluti.

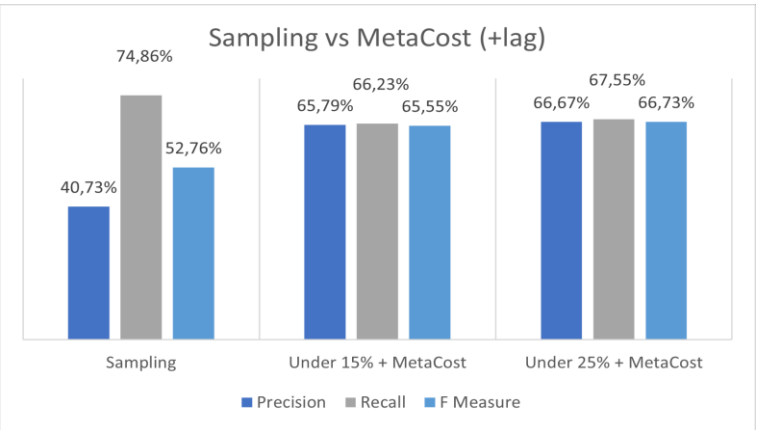

Figura 3.6.25

SVM Polynomial

Il kernel multidimensionale ancor più di quanto visto con il *decision tree* e la *random forest*, perde la sua capacità di classificare efficacemente le osservazioni rispetto al modello di confronto. Sottolineando il fatto che trasformare lo spazio delle features non è fruttuoso al fine di questo lavoro.

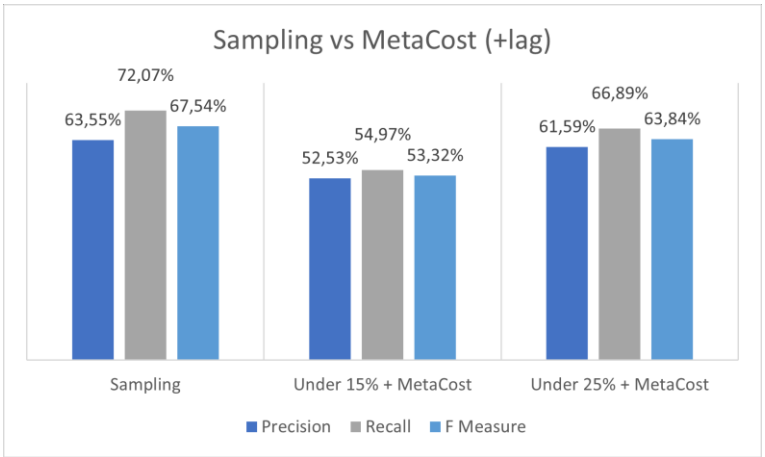

Figura 3.6.26

Concludendo, dalla Figura 3.6.27 si evince che il metodo proposto in questa sezione non è risultato utile al fine di incrementare la capacità predittiva dei modelli. Ridurre lo sbilanciamento delle classi attraverso il sottocampionamento e l'addestramento successivo del modello attraverso il *cost sensitive learning* non produce gli effetti aspettati. Inoltre, la riduzione del numero delle osservazioni ed il contemporaneo aumento delle features da 'spiegare' comporta un accentuato problema di curse dimensionality, soprattutto le tecniche più tradizionali di Machine Learning sembrano soffrirlo maggiormente. Mentre la regressione logistica riesce a gestire meglio questa situazione, incrementando addirittura il valore di F Measure nel caso di Undersampling del 25%.

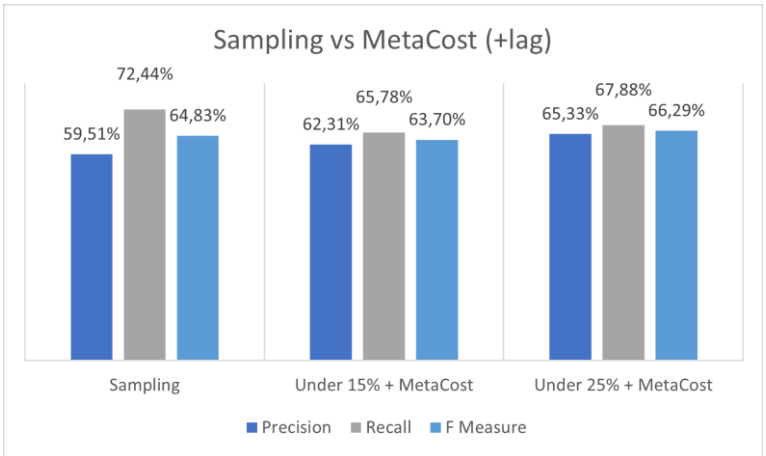

Figura 3.6.27

Esperimento 7

L'esperimento con le reti neurali dal punto di vista computazionale e delle combinazioni da testare per trovare l'ottimale è risultata essere molto dispendiosa. Infatti, il numero totale di combinazioni provate è stato di 17280, tra queste quella che ha mostrato una maggiore capacità nel classificare più correttamente le osservazioni è così formata:

- numero di neuroni = 64;
- numero di strati = 2;
- ottimizzatore = "Adam";

- tasso di apprendimento = 0.001;
- batch size = 16;
- epoche = 550;
- sliding window size = 7;
- TimeSerieSplit = 3.

L'ottimizzatore Adam si è dimostrato di grand lunga il più efficace per l'architettura in parte definita ed in parte (iperparametri) calibrata per il problema affrontato. In particolare, se la F Measure parziale è risultata essere di 0.14, quella complessiva ha raggiunto il valore di 0.45. Dalla Figura 2.23 si possono vedere i modelli più performanti sviluppati in questo lavoro.

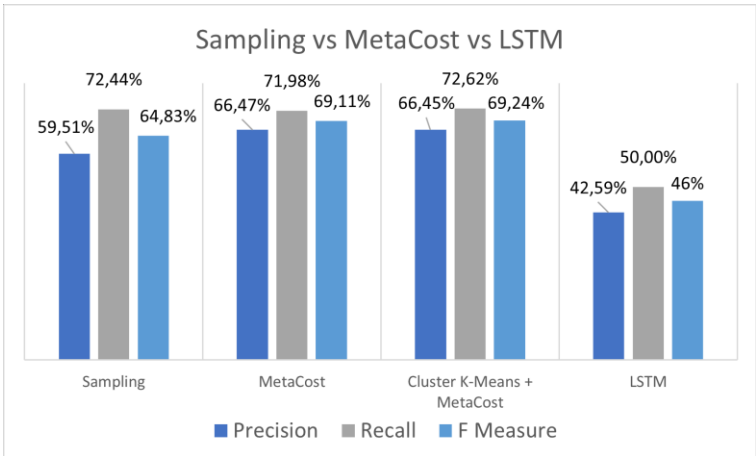

Figura 2.23

Dopo l'ottimizzazione dei circa 70.000 pesi coinvolti, il modello basato su reti Short-Term Memory (STM) ha mostrato prestazioni notevolmente inferiori rispetto agli esperimenti precedenti. Ulteriori combinazioni di iperparametri sono state esplorate, incluso l'uso della funzione di perdita "focal loss", come suggerito da Lin S. L. e Jin X. (2023), nota per gestire situazioni di class imbalance, ma anche questi tentativi non hanno prodotto risultati soddisfacenti. Per aumentare il numero di osservazioni e migliorare la capacità predittiva del modello, è stata adottata una strategia di sostituzione dei valori mancanti mediante l'interpolazione lineare e polinomiale di terzo grado. Questa operazione ha portato il numero totale di righe nel dataset a 3247, ma questo incremento

di dati non ha portato a miglioramenti significativi nelle prestazioni. Le ragioni di questa sottoperformance rispetto ad altre tecniche potrebbero derivare da diversi fattori. Uno di essi potrebbe essere l'incapacità del modello di superare un minimo locale nelle fasi di ottimizzazione. Inoltre, la limitata quantità di dati a disposizione potrebbe non essere sufficiente per consentire al modello di catturare la complessa struttura dei dati e fornire previsioni accurate.

Tabella delle combinazioni

Nella tabella a seguire sono state riportate le combinazioni vincenti per ciascun esperimento svolto per il relativo algoritmo impiegato.

Esperimento	*Undersampling / Oversampling*	*Metacost*	*Algoritmo*	*Parametri*	*F measure*
0	U. & O.; k=2	-	Decision Tree	Information; 16	69,21%
0	U. & O.; k=2	-	Random Forest	81; Accuracy; 61	69,57%
0	U. & O.; k=2	-	Logistic	COORNINATE_DESCENT_NAIVE	63,13%
0	U. & O.; k=2	-	SVM Linear	1	67,90%
0	U. & O.; k=2	-	SVM RBF	0.001, 1	52,76%
0	U. & O.; k=2	-	SVM Polynomial	0.001; 2	67,54
1	-	0.8 – 0.2	Decision Tree	Gini; 46	69.48% +/- 9.11%
1	-	0.7 – 0.3	Random Forest	400; Gini; 61	69.48% +/- 9.11%
1	-	0.7 – 0.3	Logistic	IRLSM	69.29% +/- 9.29%
1	-	0.7 – 0.3	SVM Linear	0.001	69.48% +/- 9.11%
1	-	0.8 – 0.2	SVM RBF	100; 0.01	69.29% +/- 9.29%
1	-	0.8 – 0.2	SVM Polynomial	0.1; 2	69.30% +/- 8.48%
2	U. 25%; k=4	0.8 – 0.2	Decision Tree	Accuracy; 90	69.48% +/- 9.11%
2	U. 25%; k=4	0.55 – 0.45	Random Forest	61; Gini; 31	69.48% +/- 9.11%
2	U. 25%; k=4	0.6 – 0.4	Logistic	L_BFGS	69.14% +/- 9.35%
2	U. 25%; k=4	0.55 – 0.45	SVM Linear	0.01	69.49% +/- 9.11%
2	U. 25%; k=4	0.7 – 0.3	SVM RBF	0; 0.001	69.29% +/- 9.29%
2	U. 25%; k=4	0.5 – 0.5	SVM Polynomial	0; 2	68.63% +/- 9.98%
3	U. 15%; k=4	0.75 – 0.25	Decision Tree	Accuracy; 76	69.48% +/- 9.11%
3	U. 15%; k=4	0.55 – 0.45	Random Forest	240; Accuracy; 61	69.48% +/- 9.11%
3	U. 15%; k=4	0.55 – 0.45	Logistic	L_BFGS	69.14% +/- 9.35%
3	U. 15%; k=4	0.7 – 0.3	SVM Linear	0.001	69.48% +/- 9.11%
3	U. 15%; k=4	0.5 – 0.5	SVM RBF	0.1; 0.01	69.30% +/- 8.48%

3	U. 15%; k=4	0.6 – 0.4	SVM Polynomial	0.01; 2	67.23% +/- 10.14%
4	-	Cluster	Decision Tree	Cluster	69.92%
4	-	Cluster	Random Forest	Cluster	69.71%
4	-	Cluster	Logistic	Cluster	69.52%
4	-	Cluster	SVM Linear	Cluster	67.79%
4	-	Cluster	SVM RBF	Cluster	64.74%
4	-	Cluster	SVM Polynomial	Cluster	55.31%
5	-	Cluster	Decision Tree	Cluster	69.54%
5	-	Cluster	Random Forest	Cluster	69.71%
5	-	Cluster	Logistic	Cluster	68.23%
5	-	Cluster	SVM Linear	Cluster	69.70%
5	-	Cluster	SVM RBF	Cluster	69.67%
5	-	Cluster	SVM Polynomial	Cluster	68.56%
6.1	-	0.8 – 0.2 (+ Lag)	Decision Tree	Gini; 105	67.50% +/- 7.92%
6.1	-	0.65 – 0.35 (+ Lag)	Random Forest	161; Accuracy; 31	67.72% +/- 8.15%
6.1	-	0.8 – 0.2 (+ Lag)	Logistic	AUTO	67.28% +/- 7.79%
6.1	-	0.7 – 0.3 (+ Lag)	SVM Linear	0.01	67.50% +/- 7.92%
6.1	-	0.7 – 0.3 (+ Lag)	SVM RBF	10; 0.01	67.56% +/- 7.85%
6.1	-	0.6 – 0.4 (+ Lag)	SVM Polynomial	0.001; 2	68.19% +/- 8.34%
6.2	U. 25%; k=4	0.8 – 0.2 (+ Lag)	Decision Tree	Gini; 105	67.07% +/- 7.83%
6.2	U. 25%; k=4	0.8 – 0.8 (+ Lag)	Random Forest	240; Gini; 31	67.50% +/- 7.92%
6.2	U. 25%; k=4	0.5 – 0.5 (+ Lag)	Logistic	COORDINATE_DESCENT	65.08% +/- 7.28%
6.2	U. 25%; k=4	0.55 – 0.45 (+ Lag)	SVM Linear	0.01	67.50% +/- 7.92%
6.2	U. 25%; k=4	0.5 – 0.5 (+ Lag)	SVM RBF	0.4; 0.01	66.73% +/- 9.02%
6.2	U. 25%; k=4	0.6 – 0.4 (+ Lag)	SVM Polynomial	0; 2	63.84% +/- 7.22%
6.2	U. 15%; k=4	0.5 – 0.5 (+ Lag)	Decision Tree	Gini; 76	67.25% +/- 7.39%
6.2	U. 15%; k=4	0.55 – 0.45 (+ Lag)	Random Forest	81; Accuracy; 31	67.50% +/- 7.92%
6.2	U. 15%; k=4	0.5 – 0.5 (+ Lag)	Logistic	L_BFGS	61.04% +/- 8.70%
6.2	U. 15%; k=4	0.6 – 0.4 (+ Lag)	SVM Linear	0.01	67.28% +/- 7.79%
6.2	U. 15%; k=4	0.6 – 0.4 (+ Lag)	SVM RBF	100; 0.01	65.55% +/- 9.24%
6.2	U. 15%; k=4	0.55 – 0.45 (+ Lag)	SVM Polynomial	0.001; 2	53.32% +/- 12.95%
7	-	-	LSTM	64; 2; Adam; 0.001; 16; 7; 3	46.00%

Per gli esperimenti che prevedono il clustering, i valori degli iperparametri non sono stati riportati per questioni di spazio, ma sono presenti in Appendice.

Capitolo 4

Conclusioni

L'obiettivo di questo lavoro è stato quello di testare metodi di cost sensitive learning per l'addestramento di modelli con lo scopo di fare previsioni sulle crisi bancarie sistemiche. Oltre al MetaCost che fornisce la possibilità di assegnare costi in maniera distinta per ciascuna classe, sono stati provati altri approcci ibridi in contrapposizione a quello più classico di campionamento proposta da Rocchi (2022).

Dai risultati ottenuti si può notare che generalmente gli algoritmi che hanno performato meglio sono il Decision Tree, Random Forest e Logistic Regression. Mentre le metodologie che hanno permesso valori più alti in termini di F Measure diffusa sono l'Esperimento 4 e 1. In particolare nel quarto il decision tree è risultato particolarmente efficace per tutti e tre i gruppi. Invece nel primo esperimento in cui è stato applicato il MetaCost puro, tutti i modelli si sono dimostrati capaci di supperare il 69% di F Measure. Il quale sembrerebbe essere un valore soglia, oltre il quale nessuna combinazione è riuscita a spingersi, con il presente dataset e i due diversi formati sperimentati.

Come divulgato da Tatsuyuki T. et al (2016) e molti altri studi, si confermano le spiccate capacità previsive "out of sample" dei modelli di Random Forest, in quanto riescono a delineare bordi non lineari ottenendo risultati accurati e robusti. Questa caratteristica è dovuta al fatto che, rispetto al Decision Tree, costruisce un gran numero di alberi catturando cosi molta più varianza dai dati, rispetto a quanto può fare un solo albero. Vista però la dimensione (ristretta) del dataset impiegato, questa differenza non è completamente visibile poiché la RF, così come tutte le tecniche di ML, riesce a essere più efficace quando hanno a disposizione grandi collezioni di dati. In molti esperimenti hanno ottenuto prestazioni uguali, anzi, nel caso di suddivisione in gruppi con pochi dati si è dimostrato essere seppur di poco, migliore la RF rispetto al DT. Il merito può essere

attribuito in parte a come lo stesso algoritmo costituisce il dataset di training cioè con il bagging, ma anche al bias provocato dall'overfitting. Il modello in presenza di pochi dati si adatto bene a essi, ma il rischio è che generalizzi male. Quindi un elemento in più correttamente classificato incide molto sul valore finale delle metriche, ma il valore della stessa non può essere attendibile perché in presenza di tanti dati il modello può dimostrarsi instabile rispetto alle performance attese.

Rispetto all' uso del campionamento come metodo per la riduzione dello sbilanciamento dei dati, l'uso del MetaCost ottiene risultati medi di F Measure più elevati del 5%. L'aggiunta di un sotto campionamento della classe 'No Crisi' in entrambi i casi trattati, ha portato ad una flessione della medesima metrica. Tuttavia, il metodo cost sensitive ha dimostrato essere mediamente più sensibile, in confronto con i medesimi algoritmi dell'Esperimento 0. Questo ha portato anche le performance dei modelli ottenuti con la SVM e Logistic Regression su livelli del Decision Tree e Random Forest. Evidenziando però un limite intrinseco ai dati analizzati, cioé che nessun modello è riuscito ad oltrepassare il valore di 73% di sensitività e di 67% per la precision.

La proposta di aggiungere più informazioni sulla storia di ciascuna variabile (lag), non ha migliorato l'abilità previsionale dei modelli. Probabilmente è dovuto a due fattori, una scarsa disponibilità di dati soprattutto per la classe minoritaria, e dall'incremento contemporaneo del numero di variabili esplicative.

Per quanto cerne il tentativo effettuato con le reti LSTM non hanno portato grandi risultati nonstate le svariati combinazioni tentate. Questo a dimostrazione del fatto che esse sono uno strumento molto potente ma altrettanto difficili da calibrare e necessitano di tanto potere computazionale.

Le tecniche di Machine Learning per funzionare bene necessitano di tanti dati per riuscire a catturare la struttura dei dati ed individuare eventuali pattern. Questa rappresenta una forte limitazione per il presente lavoro, poiché il dataset impiegato è popolato solamente da 2888

elementi. Per ovviare parzialmente a questa problematica per futuri sviluppi, si potrebbe cambiare la frequenza del campione, ad esempio mensile anziché annuale. Come ulteriore tentativi si sugeriscono due tecniche interessanti. La prima è di impiegare le reti GRU (Gated Recurrent Units), le quali potrebbe migliorare la capacità predittiva degli EWS, perché particolarmente adatta quando la disponibilità di dati è limitata. La secondo tecnica, è l'impiego di simulazioni di Monte Carlo per generare dati aggiuntivi per ampliare il dataset, includendo scenari finanziari diversi e migliorando la diversità e la quantità dei dati di addestramento. Entrambe le proposte sono indirizzate nel risolvere la ridotta disponobilità di dati essenziale per le tecniche soffidsticate come le reti.

Appendice

Confusion Matrix

Accuracy			
F Measure			
Predict \ Actual	Actual False (0)	Actual True (1)	Precision
Predict False (0)	TN	FN	$\frac{TN}{TN + FN}$
Predict True (1)	FP	TP	$\frac{TP}{TP + FP}$
Recall	$\frac{TN}{TN + FP}$	$\frac{TP}{TP + FN}$	

Lo schema del grafico precedente anticipa la struttura che viene usata per mostrare i risultati, nella quale sono presenti le metriche di confronto e la matrice di confusione.

[0] Under + Oversampling

- [2709 (0) – 179(1)], sbilanciamento 93,34%
- Bilanciato 1:1

Decision Tree

Accuracy	95,91%
F measure	68,11%

Predicted	**true false**	**true true**	**precision**
pred false	2644	53	98,03%
pred true	65	126	65,97%
Recall	97,60%	70,39%	

Random Forest

Accuracy	96,05%
F measure	69,52%

Predicted	**true false**	**true true**	**precision**
pred false	2644	49	98,18%
pred true	65	130	66,67%

recall	97,60%	72,63%	

Logistic Regression

Accuracy 94,70%

F measure 63,13%

Predicted	true false	true true	precision
pred false	2604	48	98,19%
pred true	105	131	55,51%
recall	96,12%	73,18%	

SVM linear

Accuracy 95,81%

F measure 67,90%

Predicted	true false	true true	precision
pred false	2639	51	98,10%
pred true	70	128	64,65%
recall	97,42%	71,51%	

SVM radial

Accuracy 91,69%

F measure 52,76%

Predicted	true false	true true	precision
pred false	2514	45	98,24%
pred true	195	134	40,73%
recall	92,80%	74,86%	

SVM Polynomial

Accuracy 95,71%

F measure 67,54%

Predicted	true false	true true	precision
pred false	2635	50	98,14%
pred true	74	129	63,55%
recall	97,27%	72,07%	

[1] Metacost

- [2709 (0) – 179 (1)]; sbilanciamento 93,34%
- MetaCost 200 iterazioni

F Measure (Loop): il valore F score ottenuto dal loop effettuato per la ricerca degli iperparametri, sulla base di questo valore sono state scelte le combinazioni che vengono riportate per ogni classificatore.
F Measure (Cross Validation): il valore ottenuto dal processo con Cross Validation e senza loop, la combinazione in assoluto migliore è stata scelta sulla base di questa metrica.

Decision Tree

Tabella delle combinazioni

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Gini	31	0.699	69.48% +/- 9.11
0.6 – 0.4	Gini	76	0.697	69.48% +/- 9.11
0.7 – 0.3	Gini	76	0.697	69.48% +/- 9.11
0.75 – 0.25	Gini	76	0.698	69.48% +/- 9.11
0.8 – 0.2	Gini	46	0.698	69.48% +/- 9.11
0.9 – 0.1	Accuracy	46	0.697	69.48% +/- 9.11

Parametri migliori

Costi	Criterio	Profondità
0.8 – 0.2	Gini	46

* 1000 iterazioni

0.8 – 0.2	Accuracy	105	0.698	69.48% +/- 9.11

Confusion Matrix

Accuracy	96,05% +/- 1,24%
F measure	69,48% +/- 9,11%

predicted	true false	true true	precision
pred false	2644	49	98,18
pred true	65	130	66,67%
recall	97,60%	72,63	

Random Forest

Tabella delle combinazioni

Costi	Alberi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	81	Information	31	0.661	62.55% +/- 9.14
0.55 – 0.45	240	Information	90	0.693	66.25% +/- 10.93
0.6 – 0.4	320	Information	90	0.693	69.33% +/- 9.55
0.7 – 0.3	400	Information	61	0.669	69.48% +/- 9.11
0.8 – 0.2	400	Accuracy	31	0.700	69.48% +/- 9.11
0.9 – 0.1	320	Accuracy	31	0.695	68.55% +/- 9.24

Parametri migliori

Costi	Alberi	Criterio	Profondità
0.7 – 0.3	400	Gini	61

Confusion Matrix

Accuracy	96,05% +/- 1,24%
F measure	69,48% +/- 9,11%

predicted	true false	true true	precision
pred false	2644	49	98,18%
pred true	65	130	66,67%

recall	97,60%	72,63%	

Logistica

Tabella delle combinazioni

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	L_BFGS	0.661	66.36% +/- 10.37
0.55 – 0.45	L_BFGS	0.696	68.29% +/- 10.02
0.6 – 0.4	IRLSM	0.696	69.14% +/- 8.67
0.7 – 0.3	IRLSM	0.696	69.48% +/- 9.11
0.8 – 0.2	L_BFGS	0.695	69.29% +/- 9.29
0.9 – 0.1	COORDINATE_DESCENT	0.697	69.14% +/- 8.67

Parametri migliori

Costi	Criterio
0.7– 0.3	IRLSM

Confusion Matrix

Accuracy	96,02% +/- 1,26%
F measure	69,29% +/- 9,29%

predicted	true false	true true	precision
pred false	2643	49	98,18%
pred true	66	130	66,33%
recall	97,56%	72,63%	

SVM LINEAR

Tabella delle combinazioni

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.1	0.696	69.48% +/- 9.11
0.55 – 0.45	0.01	0.697	69.48% +/- 9.11
0.6 – 0.4	0.001	0.696	69.48% +/- 9.11
0.7 – 0.3	0.001	0.696	69.48% +/- 9.11
0.8 – 0.2	1	0.150	11.67% +/- 0.34
0.9 – 0.1	0.001	0.117	11.67% +/- 0.34

Parametri migliori

Costi	C
0.7 – 0.3	0.001

Confusion Matrix

Accuracy	96,05% +/- 1,24%
F measure	69,48% +/- 9,11%

predicted	true false	true true	precision
pred false	2644	49	98,18%
pred true	65	130	66,67%
recall	97,60%	72,63%	

SVM RBF

Tabella delle combinazioni

Costi	C	gamma	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	1	0.01	0.698	69.68% +/- 8.96
0.55 – 0.45	0.001	0.01	0.699	69.68% +/- 8.96
0.6 – 0.4	0.001	0.01	0.702	69.48% +/- 9.11
0.7 – 0.3	100	0.01	0.697	69.48% +/- 9.11
0.8 – 0.2	100	0.01	0.693	69.48% +/- 9.11
0.9 – 0.1	100	0.01	0.689	69.11% +/- 9.36

Parametri migliori

Costi	C	gamma
0.8 – 0.2	100	0.01

Confusion Matrix

Accuracy	96,05% +/- 1,24%
F measure	69,29% +/- 9,29%

predicted	true false	true true	precision
pred false	2644	49	98,18%
pred true	65	130	66,67%
recall	97,60%	72,63%	

SVM Polynomial

Tabella delle combinazioni

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	2	0.700	69.29% +/- 9.29
0.55 – 0.45	0.001	2	0.698	69.29% +/- 9.29
0.6 – 0.4	0.001	2	0.698	69.29% +/- 9.29
0.7 – 0.3	0.001	2	0.685	68.13% +/- 10.12
0.8 – 0.2	0.1	2	0.678	69.30% +/- 8.48
0.9 – 0.1	0.1	2	0.425	50.64% +/- 13.26

Parametri migliori

Costi	C	degree
0.8 – 0.2	0.1	2

Confusion Matrix

Accuracy	96,05% +/- 1,19%
F measure	69,30% +/- 8,48%

predicted	true false	true true	precision
pred false	2645	50	98,14%
pred true	64	129	66,84%
recall	97,64%	72,07%	

[2] Undersampling + Metacost

- Undersampling 25%
- [673 (0) – 179 (1)], sbilanciamento 77,5%
- K = 4
- MetaCost 200 iterazioni

Decision Tree

Tabella delle combinazioni

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Information	135	0.697	68.73% +/- 12.56
0.6 – 0.4	Information	31	0.699	69.48% +/- 9.11
0.7 – 0.3	Accuracy	90	0.698	69.48% +/- 9.11
0.75 – 0.25	Accuracy	90	0.698	69.48% +/- 9.11
0.8 – 0.2	Accuracy	90	0.698	69.48% +/- 9.11
0.9 – 0.1	Gini	150	0.697	68,79% +/- 9.93

Parametri migliori

Costi	Criterio	Profondità
0.8 – 0.2	Accuracy	90

* 1000 iterazioni

0.8 – 0.2	Gini	150	0.697	55.25% +/- 8.54%

Confusion Matrix

Accuracy	96,05% +/- 1,24%
F measure	69,48% +/- 9,11%

predicted	true false	true true	precision
pred false	2644	49	98,18
pred true	65	130	66,67%
recall	97,60%	72,63	

Random Forest

Tabella delle combinazioni

Costi	Alberi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	161	Accuracy	61	0.696	69.48% +/- 9.11
0.55 – 0.45	81	Gini	31	0.697	69.48% +/- 9.11
0.6 – 0.4	161	Information	61	0.697	69.32% +/- 9.18
0.7 – 0.3	1	Information	120	0.669	64.78% +/- 7.67
0.8 – 0.2	1	Accuracy	31	0.643	62.89% +/- 9.29
0.9 – 0.1	1	Accuracy	120	0.307	25.87% +/- 10.77

Parametri migliori

Costi	Alberi	Criterio	Profondità
0.55 – 0.45	81	Gini	31

Confusion Matrix

Accuracy	96,05% +/- 1,24%
F measure	69,48% +/- 9,11%

predicted	true false	true true	precision
pred false	2644	49	98,18%
pred true	65	130	66,67%
recall	97,60%	72,63%	

Logistica

Tabella delle combinazioni

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	L_BFGS	0.695	69.14% +/- 9.35
0.55 – 0.45	L_BFGS	0.695	69.14% +/- 9.35
0.6 – 0.4	L_BFGS	0.695	69.14% +/- 9.35
0.7 – 0.3	L_BFGS	0.689	68.64% +/- 9.83
0.8 – 0.2	COORDINATE_DESCENT	0.636	64.98% +/- 9.84
0.9 – 0.1	COORDINATE_DESCENT	0.339	33.10% +/- 3.96

Parametri migliori

Costi	Criterio
0.6 – 0.4	L_BFGS

Confusion Matrix

Accuracy	95,98% +/- 1,29%
F measure	69,14% +/- 9,35%

predicted	true false	true true	precision
pred false	2642	49	98,18%
pred true	67	130	65,99%
recall	97,53%	72,63%	

SVM LINEAR

Tabella delle combinazioni

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.01	0.697	69.49% +/- 9.11
0.55 – 0.45	0.01	0.697	69.49% +/- 9.11
0.6 – 0.4	0.01	0.696	69.29% +/- 9.29
0.7 – 0.3	0.01	0.697	68.63% +/- 9.98
0.8 – 0.2	100	0.156	13.20% +/- 2.82
0.9 – 0.1	0.01	0.117	11.67% +/- 0.34

Parametri migliori

Costi	C
0.55 – 0.45	0.01

Confusion Matrix

Accuracy	96,05% +/- 1,24%
F measure	69,49% +/- 9,11%

predicted	true false	true true	precision
pred false	2644	49	98,18%

pred true	65	130	66,67%
recall	97,60%	72,63%	

SVM RBF

Tabella delle combinazioni

Costi	C	gamma	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	0.001	0.697	69.48% +/- 9.11
0.55 – 0.45	0.001	0.001	0.697	68.79% +/- 9.93
0.6 – 0.4	0.4	0.01	0.698	68.40% +/- 9.46
0.7 – 0.3	0.001	0.001	0.696	69.29% +/- 9.29
0.8 – 0.2	10	0.01	0.548	58.95% +/- 11.74
0.9 – 0.1	10	0.01	0.136	16.17% +/- 6.21

Parametri migliori

Costi	C	gamma
0.7 – 0.3	0.001	0.001

Confusion Matrix

Accuracy	96,02% +/- 1,26%
F measure	69,29% +/- 9,29%

predicted	true false	true true	precision
pred false	2643	49	98,18%
pred true	66	130	66,33%
recall	97,56%	72,63%	

SVM Polynomial

Tabella delle combinazioni

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	2	0.694	68.63% +/- 9.98
0.55 – 0.45	0.001	2	0.691	67.84% +/- 10.41
0.6 – 0.4	0.001	2	0.684	67.28% +/- 10.31
0.7 – 0.3	0.4	2	0.680	67.14% +/- 9.54
0.8 – 0.2	0.4	2	0.656	64.95% +/- 11.15
0.9 – 0.1	0.4	2	0.374	33.36% +/- 11.23

Parametri migliori

Costi	C	degree
0.5 – 0.5	0.001	2

Confusion Matrix

Accuracy	95,88% +/- 1,37%
F measure	68,63% +/- 9,98%

predicted	true false	true true	precision
pred false	2639	49	98,18%
pred true	70	130	65,00%
recall	97,42%	72,63%	

[3] Undersampling + Metacost

- Undersampling 15%
- [403 (0) – 179 (1)], sbilanciamento 67,39%
- K = 4
- MetaCost 200 iterazioni

Decision Tree

Tabella delle combinazioni

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Gini	76	0.697	69.48% +/- 9.11
0.6 – 0.4	Accuracy	76	0.698	69.48% +/- 9.11
0.7 – 0.3	Accuracy	76	0.698	69.48% +/- 9.11
0.75 – 0.25	Accuracy	76	0.698	69.48% +/- 9.11
0.8 – 0.2	Accuracy	76	0.698	69.32% +/- 9.18
0.9 – 0.1	Gain	31	0.278	27.62% +/- 4.64

Parametri migliori

Costi	Criterio	Profondità
0.75 – 0.25	Accuracy	76

Confusion Matrix

Accuracy	96,05% +/- 1,24%
F measure	69,48% +/- 9,11%

predicted	true false	true true	precision
pred false	2644	49	98,18
pred true	65	130	66,67%
recall	97,60%	72,63	

Random Forest

Tabella delle combinazioni

Costi	Alberi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	161	Accuracy	90	0.698	69.48% +/- 9.11
0.55 – 0.45	240	Accuracy	61	0.699	69.48% +/- 9.11
0.6 – 0.4	240	Accuracy	90	0.696	69.32% +/- 9.15
0.7 – 0.3	81	Accuracy	61	0.683	66.26% +/- 9.25
0.8 – 0.2	161	Accuracy	61	0.522	47.83% +/- 5.77
0.9 – 0.1	161	Information	31	0.229	21.78% +/- 1.25

Parametri migliori

Costi	Alberi	Criterio	Profondità
0.55 – 0.45	240	Accuracy	61

Confusion Matrix

Accuracy	96.05% +/- 1,24%
F measure	69.48% +/- 9.11%

predicted	true false	true true	precision
pred false	2644	49	98.18%
pred true	65	130	66,67%
recall	97,60%	72,63%	

Logistica

Tabella delle combinazioni

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	IRLSM	0.689	68.65% +/- 9.65
0.55 – 0.45	L_BFGS	0.694	69.14% +/- 9.35
0.6 – 0.4	L_BFGS	0.690	68.37% +/- 10.24
0.7 – 0.3	COORDINATE_DESCENT_NAIVE	0.624	64.27% +/- 11.01
0.8 – 0.2	AUTO	0.424	42.56% +/- 7.65
0.9 – 0.1	L_BFGS	0.216	21.01% +/- 1.29

Parametri migliori

Costi	Criterio
0.55 – 0.45	L_BFGS

Confusion Matrix

Accuracy	95,98% +/- 1,29%
F measure	69,14% +/- 9,35%

predicted	true false	true true	precision
pred false	2642	49	98,18%
pred true	67	130	65,99%
recall	97,53%	72,63%	

SVM LINEAR

Tabella delle combinazioni

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.01	0.696	69.48% +/- 9.11
0.55 – 0.45	0.01	0.696	69.48% +/- 9.11
0.6 – 0.4	0.01	0.696	69.14% +/- 9.35
0.7 – 0.3	0.001	0.691	69.48% +/- 9.11
0.8 – 0.2	0.1	0.681	67.58% +/- 9.96
0.9 – 0.1	10	0.117	11.67% +/- 0.34

Parametri migliori

Costi	C
0.7 – 0.3	0.001

Confusion Matrix

Accuracy	96,05% +/- 1,24%
F measure	69,48% +/- 9,11%

predicted	true false	true true	precision
pred false	2644	49	98,18%
pred true	65	130	66,67%
recall	97,60%	72,63%	

SVM RBF

Tabella delle combinazioni

Costi	C	gamma	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.1	0.01	0.695	69.30% +/- 8.48
0.55 – 0.45	0.4	0.01	0.700	68.25% +/- 9.69
0.6 – 0.4	0.1	0.01	0.696	68.97% +/- 8.65
0.65 – 0.35	100	0.01	0.693	69.11% +/- 9.49
0.7 – 0.3	0.01	0.1	0.690	68.79% +/- 8.80
0.8 – 0.2	100	0.01	0.574	60.53% +/- 11.74
0.9 – 0.1	100	0.1	0.246	36.62% +/- 7.48

Parametri migliori

Costi	C	gamma
0.5 – 0.5	0.1	0.01

Confusion Matrix

Accuracy	96,05% +/- 1,19%
F measure	69,30% +/- 8,48%

predicted	true false	true true	precision
pred false	2645	50	98,14%
pred true	64	129	66,84%
recall	97,64%	72,07%	

SVM Polynomial

Tabella delle combinazioni

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.01	2	0.682	67.29% +/- 10.22
0.55 – 0.45	0.01	2	0.679	66.91% +/- 10.04
0.6 – 0.4	0.01	2	0.674	67.23% +/- 10.14
0.7 – 0.3	1	2	0.673	63.46% +/- 10.67
0.8 – 0.2	0.4	2	0.536	53.67% +/- 10.66
0.9 – 0.1	0.4	2	0.233	21.12% +/- 4.95

Parametri migliori

Costi	C	degree
0.6 – 0.4	0.01	2

Confusion Matrix

Accuracy	95,57% +/- 1,45%
F measure	67,23% +/- 10,14%

predicted	true false	true true	precision

pred false	2629	48	98,21%
pred true	80	131	62,09%
recall	97,05%	73,18%	

[4] Cluster durata + MetaCost

- '*sotto*': 2085 elementi, 2036 (0) – 38 (1), sbilanciamento del 97,65%
- '*media*': 382 elementi, 321 (0) – 61 (1), sbilanciamento del 84,03%
- '*sopra*': 421 elementi, 341 (0) – 80 (1), sbilanciamento del 80,99%

Decision Tree

- **Cluster sotto**

Parametri migliori

Costi	Criterio	Profondità
0.97 – 0.03	Gini	135

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Gain	105	0.276	0.2546
0.6 – 0.4	Gain	61	0.355	0.3218
0.7 – 0.3	Gain	61	0.390	0.3514
0.8 – 0.2	Gain	90	0.429	0.3704
0.9 – 0.1	Gain	135	0.460	0.4419
0.97 – 0.03	Gini	135	0.463	0.4419

Confusion Matrix

Accuracy	97,70% +/- 0,78%
F measure	44,19% +/- 12%

predicted	true false	true true	precision
pred false	2018	19	99,07%
pred true	29	19	39,58%
recall	98,58%	50,00%	

- **Cluster media**

Parametri migliori

Costi	Criterio	Profondità
0.8 – 0.2	Gini	46

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Gini	1	0.691	69.46
0.6 – 0.4	Accuracy	150	0.711	68.14
0.7 – 0.3	Information	120	0.714	69.06
0.8 – 0.2	Gini	46	0.706	70.25
0.85 – 0.15	Gain	46	0.706	69.74
0.9 – 0.1	Accuracy	31	0.700	69

Confusion Matrix

Accuracy	90,82% +/- 5,73%
F measure	71,24% +/- 18,34%

predicted	true false	true true	precision
pred false	302	16	94,97%
pred true	19	45	70,31%
recall	94,08%	73,77%	

- **Cluster sopra**

Parametri migliori

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Gain	31	0.824	82.36
0.55 – 0.45	Accuracy	120	0.823	81.94

Costi	Criterio	Profondità
0.5– 0.5	Gain	31

0.6– 0.4	Accuracy	46	0.821	81.35
0.7– 0.3	Accuracy	120	0.812	80.77
0.8– 0.2	Gain	105	0.796	79.50
0.9– 0.1	Accuracy	61	0.785	77.90

Confusion Matrix

Accuracy	93,36% +/- 4,43%
F measure	82,36% +/- 11,23%

predicted	true false	true true	precision
pred false	328	15	95,63%
pred true	13	65	83,33%
recall	96,19%	81,25%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

Decision Tree

Accuracy	96,16% +/- 2,49%
F measure	69,92%

Predicted	true false	true true	precision
pred false	2648	50	98,15%
pred true	61	129	67,89%
recall	97,75%	72,07%	

Random Forest

- ## Cluster sotto

Costi	Alberi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	81	Accuracy	31	0.185	18.50
0.6 – 0.4	1	Accuracy	90	0.246	24.60
0.7 – 0.3	161	Accuracy	150	0.329	32.90
0.8 – 0.2	240	Accuracy	61	0.48	44.19
0.9 – 0.1	81	Gain	31	0.472	44.94
0.97 – 0.03	240	Accuracy	61	0.474	44.19

Parametri migliori

Costi	Alberi	Criterio	Profondità
0.8 – 0.2	240	Accuracy	61

Confusion Matrix

Accuracy	97,70% +/- 0,78%
F measure	45,45% +/- 12,14%

Predicted	true false	true true	precision
pred false	2017	18	99,12%
pred true	30	20	40,00%
recall	98,53%	52,63%	

- ## Cluster media

Costi	Alberi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	400	Gini	31	0.709	69.89
0.6 – 0.4	81	Gini	90	0.728	70.35
0.7 – 0.3	320	Accuracy	31	0.733	71.62
0.75 – 0.25	81	Accuracy	90	0.735	71.62
0.8 – 0.2	320	Information	61	0.738	69.46
0.9 – 0.1	400	Accuracy	90	0.679	67.90

Parametri migliori

Costi	Alberi	Criterio	Profondità
0.75 – 0.25	81	Accuracy	90

Confusion Matrix

Accuracy	90,08% +/- 5,31%
F measure	71,62% +/- 17,78%

Predicted	**true false**	**true true**	**precision**
pred false	303	16	94,98%
pred true	18	45	71,43%
recall	94,39%	73,77%	

- **Cluster sopra**

Parametri migliori

Costi	Alberi	Criterio	Profondità
0.55 – 0.45	400	Information	61

Costi	Alberi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	161	Gini	61	0.87	79.96
0.55 – 0.45	400	Information	61	0.824	81.02
0.6 – 0.4	400	Information	61	0.821	81.04
0.7 – 0.3	400	Information	1	0.816	80.46
0.8 – 0.2	400	Information	150	0.811	80.18
0.9 – 0.1	1	Accuracy	61	0.723	72.30

Confusion Matrix

Accuracy	92,65% +/- 5,17%
F measure	81,02% +/- 12,70%

Predicted	**true false**	**true true**	**precision**
pred false	325	15	95,59%
pred true	16	65	80,25%
recall	95,31%	81,25%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

Random Forest

Accuracy	96,09% +/- 2,82%
F measure	69,71%

Predicted	**true false**	**true true**	**precision**
pred false	2645	49	98,18%
pred true	64	130	67,01%
recall	97,64%	72,63%	

Logistic Regression

- **Cluster sotto**

Parametri migliori

Costi	Criterio
0.7 – 0.3	COORDINATE_DESCENT NAIVE

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	COORDINATE_DESCENT NAIVE	0.446	45.98%
0.6 – 0.4	COORDINATE_DESCENT NAIVE	0.446	45.98%
0.7 – 0.3	COORDINATE_DESCENT NAIVE	0.446	45.98%
0.8 – 0.2	COORDINATE_DESCENT NAIVE	0.442	45.45%
0.9 – 0.1	COORDINATE_DESCENT NAIVE	0.460	41.24%
0.97 – 0.03	IRLSM	0.269	28.84%

Confusion Matrix

Accuracy	97,75% +/- 0,75%
F measure	45,98% +/- 12,14%

Predicted	**true false**	**true true**	**precision**
pred false	2018	18	99,12%
pred true	29	20	40,82%

recall	98,58%	52,63%	

- **Cluster media**

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	L_BFGS DESCENT	0.685	68.50
0.6 – 0.4	L_BFGS DESCENT	0.710	69.74
0.7 – 0.3	L_BFGS DESCENT	0.725	69.74
0.75– 0.25	L_BFGS DESCENT	0.723	69.26
0.8 – 0.2	IRLS	0.727	71.07
0.9 – 0.1	L_BFGS DESCENT	0.627	68.99

Parametri migliori

Costi	Criterio
0.8 – 0.2	IRLS

Confusion Matrix

Accuracy	90,82% +/- 5,31%
F measure	70,35% +/- 18,15%

Predicted	true false	true true	precision
pred false	303	17	94,69%
pred true	18	44	70,97%
recall	94,39%	72,13%	

- **Cluster sopra**

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	L_BFGS DESCENT	0.795	79.50
0.6 – 0.4	L_BFGS DESCENT	0.808	80.46
0.65 – 0.35	L_BFGS DESCENT	0.808	80.46
0.7 – 0.3	L_BFGS DESCENT	0.808	80.50
0.8 – 0.2	L_BFGS DESCENT	0.797	79.70
0.9 – 0.1	AUTO	0.716	71.60

Parametri migliori

Costi	Criterio
0.6 – 0.4	L_BFGS DESCENT

Confusion Matrix

Accuracy	92,41% +/- 5,09%
F measure	80,46% +/- 12,82%

Predicted	true false	true true	precision
pred false	323	14	95,85%
pred true	18	66	78,57%
recall	94,72%	82,50%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

Logistic Regression

Accuracy	96,05% +/- 2,73%
F measure	69,52%

Predicted	true false	true true	precision
pred false	2644	49	98,18%
pred true	65	130	66,67%
recall	97,60%	72,63%	

SVM LINEAR

- **Cluster sotto**

Parametri migliori

Costi	C
0.7 – 0.3	0.01

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	0.011	1.0
0.6 – 0.4	100	0.192	19.0
0.65 – 0.35	1000	0.155	15.0
0.7 – 0.3	0.01	0.384	31.40
0.8 – 0.2	100	0.036	3.50
0.9 – 0.1	0.001	0.036	3.10

Confusion Matrix

Accuracy	97,70% +/- 0,54%
F measure	31,43%

Predicted	true false	true true	precision
pred false	2026	27	98,68%
pred true	21	11	34,38%
recall	98,97%	28,95%	

- **Cluster media**

Parametri migliori

Costi	C
0.7 – 0.3	0.001

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	0.722	71.0
0.6 – 0.4	1	0.730	70.0
0.65 – 0.35	1	0.723	71.20
0.7 – 0.3	0.001	0.722	71.50
0.8 – 0.2	1	0.594	60.10
0.9 – 0.1	10	0.036	3.50

Confusion Matrix

Accuracy	90,82% +/- 5,31%
F measure	71,50% +/- 18,15%

Predicted	true false	true true	precision
pred false	303	17	94,69%
pred true	18	44	70,97%
recall	94,39%	72,13%	

- **Cluster sopra**

Parametri migliori

Costi	C
0.55 – 0.45	1

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	0.802	76.50
0.55 – 0.45	1	0.803	80.40
0.6 – 0.4	1	0.803	78.90
0.7 – 0.3	0.001	0.808	80
0.8 – 0.2	0.001	0.676	67.60
0.9 – 0.1	10	0.417	41.70

Confusion Matrix

Accuracy	92,41% +/- 5,09%
F measure	80,46% +/- 12,82%

Predicted	true false	true true	precision
pred false	323	14	95,85%
pred true	18	66	78,57%
recall	94,72%	82,50%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

SVM linear

Accuracy 96,02% +/- 2,70%

F measure 67,79%

Predicted	true false	true true	precision
pred false	2652	58	97,86%
pred true	57	121	67,98%
recall	97,90%	67,60%	

SVM RBF

- ## Cluster sotto

Parametri migliori

Costi	C	gamma
0.7 – 0.3	0.4	0.2

Costi	C	gamma	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	0.1	0.857	0.115
0.6 – 0.4	10	0.1	0.347	0.298
0.65 – 0.35	1	0.1	0.361	0.373
0.7 – 0.3	0.4	0.2	0.390	0.387
0.8 – 0.2	1000	0.1	0.301	0.294
0.9 – 0.1	1000	0.1	0.225	0.357

Confusion Matrix

Accuracy 97,27% +/- 1,45%

F measure 38,71%

Predicted	true false	true true	precision
pred false	2010	20	99,01%
pred true	37	18	32,73%
recall	98,19%	47,37%	

- ## Cluster media

Parametri migliori

Costi	C	gamma
0.55 – 0.45	1	0.1

Costi	C	gamma	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	1	0.1	0.699	0.654
0.55 – 0.45	1	0.1	0.707	0.679
0.6 – 0.4	1	0.1	0.709	0.675
0.7 – 0.3	10	0.1	0.589	0.640
0.8 – 0.2	100	0.1	0.531	0.624
0.9 – 0.1	100	0.1	0.456	0.487

Confusion Matrix

Accuracy 90,04% +/- 2,44%

F measure 67,92% +/- 8,93%

Predicted	true false	true true	precision
pred false	302	19	94,08%
pred true	19	42	68,85%
recall	94,08%	68,85%	

- ## Cluster sopra

Parametri migliori

Costi	C	gamma	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	1	0.1	0.804	0.749
0.55 – 0.45	1	0.1	0.796	0.763
0.6 – 0.4	1	0.1	0.805	0.755
0.7 – 0.3	100	0.1	0.698	0.744

Costi	C	gamma
0.55 – 0.45	1	0.1

0.8 – 0.2	100	0.1	0.604	0.710
0.9 – 0.1	100	0.1	0.439	0.486

Confusion Matrix

Accuracy	90,51% +/- 3,12%
F measure	76,30% +/- 7,07%

Predicted	true false	true true	precision
pred false	317	16	95,20%
pred true	24	64	72,73%
recall	92,96%	80,00%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

SVM radial

Accuracy	95,32% +/- 3,67%
F measure	64,74%

Predicted	true false	true true	precision
pred false	2629	55	97,95%
pred true	80	124	60,78%
recall	97,04%	69,27%	

SVM Polynomial

- **Cluster sotto**

Parametri migliori

Costi	C	degree
0.6 – 0.4	1000	6

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	4	0.100	0.081
0.55 – 0.45	1000	2	0.324	0.083
0.6 – 0.4	1000	6	0.036	0.142
0.7 – 0.3	0.8	4	0.174	0.034
0.8 – 0.2	100	3	0.036	0.051
0.9 – 0.1	1000	2	0.038	0.038

Confusion Matrix

Accuracy	94,44% +/- 3,33%
F measure	13,43%

Predicted	true false	true true	precision
pred false	1960	29	98,54%
pred true	87	9	9,38%
recall	95,75%	23,68%	

- **Cluster media**

Parametri migliori

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	3	0.667	0.647
0.55 – 0.45	0.001	3	0.661	0.663
0.6 – 0.4	0.001	3	0.713	0.674
0.65 – 0.35	0.01	2	0.661	0.680
0.7 – 0.3	0.001	2	0.677	0.677
0.8 – 0.2	0.4	2	0.605	0.630

Costi	C	degree
0.65 – 0.35	0.01	2

0.9 – 0.1	10	2	0.476	0.460

Confusion Matrix

Accuracy	89,78% +/- 2,61%
F measure	68,07% +/- 8,79%

Predicted	**true false**	**true true**	**precision**
pred false	300	18	94,34%
pred true	21	43	67,19%
recall	93,46%	70,49%	

- **Cluster sopra**

Parametri migliori

Costi	C	degree
0.5 – 0.5	0.01	2

Costi	**C**	**degree**	**F measure (Loop)**	**F measure (Crossvalidation)**
0.5 – 0.5	0.01	2	0.784	0.781
0.55 – 0.45	0.01	2	0.797	0.783
0.6 – 0.4	0.01	2	0.784	0.778
0.7 – 0.3	0.1	2	0.748	0.744
0.8 – 0.2	0.8	2	0.693	0.663
0.9 – 0.1	1	2	0.574	0.567

Confusion Matrix

Accuracy	91,93% +/- 2,97%
F measure	78,83% +/- 8,73%

Predicted	**true false**	**true true**	**precision**
pred false	322	15	95,55%
pred true	19	65	77,38%
recall	94,43%	81,25%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

SVM Polynomial

Accuracy	93,45% +/- 8,6%
F measure	55,31%

Predicted	**true false**	**true true**	**precision**
pred false	2582	62	97,65%
pred true	127	117	47,95%
recall	95,31%	65,36%	

[5] Clustering usando il K-Means

- Cluster 0 -> 2679 di cui 2520 (0) - 159 (1), sbilanciamento 94,06%
- Cluster 1 -> 140 di cui 126 (0) - 14 (1), sbilanciamento 90%
- Cluster 2 -> 69 di cui 63 (0) - 6 (1), sbilanciamento 91,30%

Decision Tree

- **Cluster 0**

Parametri migliori

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
.5 – 0.5	Gini	105	0.692	0.6882 +/- 8.95
0.55 – 0.45	Gini	16	0.692	0.6882 +/- 8.95
0.6 – 0.4	Accuracy	90	0.692	0.6882 +/- 8.95
0.7 – 0.3	Gini	16	0.692	0.6882 +/- 8.95
0.8 – 0.2	Gini	16	0.692	0.6882 +/- 8.95
0.9 – 0.1	Accuracy	90	0.692	0.6882 +/- 8.95

Costi	Criterio	Profondità
0.8 – 0.2	Gini	16

Confusion Matrix

Accuracy	96,12% +/- 1,09
F measure	68,82% +/- 8,95

predicted	true false	true true	precision
pred false	2460	44	98,24%
pred true	60	115	65,71%
recall	97,62%	72,33%	

- **Cluster 1**

Parametri migliori

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Accuracy	31	0.759	0.667
0.6 – 0.4	Information	61	0.800	0.6667
0.65 – 0.35	Accuracy	105	0.797	0.7586
0.7 – 0.3	Accuracy	105	0.797	0.7586
0.8 – 0.2	Information	61	0.773	0.7097
0.9 – 0.1	Gini	76	0.629	0.6286

Costi	Criterio	Profondità
0.7 – 0.3	Accuracy	105

Confusion Matrix

Accuracy	95,00% +/- 4,82
F measure	78,86%

predicted	true false	true true	precision
pred false	122	4	96,83%
pred true	3	11	78,57%
recall	97,60%	73,33%	

- **Cluster 2**

Parametri migliori

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Gini	76	0.750	0.7500
0.55 – 0.45	Gain	105	0.750	0.6667
0.6 – 0.4	Gain	105	0.750	0.6667
0.7 – 0.3	Information	76	0.667	0.5455
0.8 – 0.2	Gini	61	0.615	0.4000
0.9 – 0.1	Accuracy	46	0.615	0.4000

Costi	Criterio	Profondità
0.5 – 0.5	Gini	76

Confusion Matrix

Accuracy	97,14% +/- 6,02
F measure	75,00%

predicted	true false	true true	precision
pred false	64	2	96,97%
pred true	0	3	100%
recall	100%	60%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

Decision Tree

Accuracy	**96,08% +/- 3,97%**
F measure	**69,54%**

Predicted	**true false**	**true true**	**precision**
pred false	2646	50	98,15%
pred true	63	129	67,18%
recall	97,67%	72,07%	

Random Forest

- **Cluster 0**

Parametri migliori

Costi	Alberi	Criterio	Profondità
0.7 – 0.3	240	Information	90

Costi	**Alberi**	**Criterio**	**Profondità**	**F measure (Loop)**	**F measure (Crossvalidation)**
0.5 – 0.5	240	Information	31	0.652	0.632
0.6 – 0.4	400	Gini	120	0.690	0.684
0.7 – 0.3	240	Information	90	0.693	0.688
0.8 – 0.2	240	Information	61	0.690	0.686
0.9 – 0.1	400	Accuracy	150	0.688	0.681
0.97 – 0.03	1	Gain	31	0.476	0.375

Confusion Matrix

Accuracy	96,12% +/- 1,09%
F measure	68,82% +/- 8,95

Predicted	**true false**	**true true**	**precision**
pred false	2460	44	98,24%
pred true	60	115	65,71%
recall	97,62%	72,33%	

- **Cluster 1**

Parametri migliori

Costi	Alberi	Criterio	Profondità
0.7 – 0.3	1	Accuracy	61

Costi	**Alberi**	**Criterio**	**Profondità**	**F measure (Loop)**	**F measure (Crossvalidation)**
0.5 – 0.5	81	Gini	120	0.759	0.615
0.6 – 0.4	400	Accuracy	90	0.797	0.714
0.7 – 0.3	1	Accuracy	61	0.800	0.758
0.75 – 0.25	320	Information	31	0.800	0.758
0.8 – 0.2	320	Accuracy	90	0.840	0.733
0.9 – 0.1	240	Gini	120	0.657	0.558

Confusion Matrix

Accuracy	95,00% +/- 4,82%
F measure	75,86%

Predicted	**true false**	**true true**	**precision**
pred false	122	4	96,83%
pred true	3	11	78,57%
recall	97,60%	73,33%	

- **Cluster 2**

Parametri migliori

Costi	**Alberi**	**Criterio**	**Profondità**	**F measure (Loop)**	**F measure (Crossvalidation)**
0.5 – 0.5	81	Gini	120	0.750	0.750
0.6 – 0.4	81	Gain	31	0.889	0.750

Costi	Alberi	Criterio	Profondità
0.7 – 0.3	161	Gain	31

0.65 – 0.35	320	Information	31	0.889	0.667
0.7 – 0.3	161	Gain	31	0.889	0.800
0.8 – 0.2	240	Gini	90	0.800	0.571
0.9 – 0.1	1	Accuracy	120	0.615	0.470

Confusion Matrix

Accuracy	97,14% +/- 6,02%
F measure	80,00%

Predicted	true false	true true	precision
pred false	63	1	98,44%
pred true	1	4	80,00%
recall	98,44%	80,00%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

Random Forest

Accuracy	96,09% +/- 2,82%
F measure	69,71%

Predicted	true false	true true	precision
pred false	2645	49	98,18%
pred true	64	130	67,01%
recall	97,64%	72,63%	

Logistica

- **Cluster 0**

Parametri migliori

Costi	Criterio
0.7 – 0.3	L_BFGS

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	IRLSM	0.593	0.662
0.55 – 0.45	AUTO	0.615	0.672
0.6 – 0.4	IRLSM	0.629	0.688
0.7 – 0.3	L_BFGS	0.632	0.688
0.8 – 0.2	AUTO	0.623	0.684
0.9 – 0.1	L_BFGS	0.523	0.674

Confusion Matrix

Accuracy	97,75% +/- 1,12%
F measure	67,41% +/- 9,20%

Predicted	true false	true true	precision
pred false	2453	44	98,24%
pred true	67	115	63,19%
recall	97,34%	72,33%	

- **Cluster 1**

Parametri migliori

Costi	Criterio
0.6 – 0.4	COORDINATE_DESCENT_NAIVE

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	COORDINATE_DESCENT_NAIVE	0.688	0.733
0.55 – 0.45	COORDINATE_DESCENT_NAIVE	0.688	0.733
0.6 – 0.4	COORDINATE_DESCENT_NAIVE	0.667	0.733
0.7 – 0.3	COORDINATE_DESCENT	0.643	0.727
0.8 – 0.2	L_BFGS	0.611	0.685
0.9 – 0.1	L_BFGS	0.531	0.558

Confusion Matrix

Accuracy	94,29% +/- 5,63%
F measure	73,33%

Predicted	true false	true true	precision
pred false	121	4	96,80%
pred true	4	11	73,33%
recall	96,80 %	73,33%	

- **Cluster 2**

Parametri migliori

Costi	Criterio
0.5 – 0.5	COORDINATE_DESCENT

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	COORDINATE_DESCENT	0.800	0.800
0.55 –0.45	AUTO	0.727	0.667
0.6 – 0.4	AUTO	0.727	0.667
0.7 – 0.3	AUTO	0.615	0.533
0.8 – 0.2	AUTO	0.615	0.444
0.9 – 0.1	AUTO	0.533	0.363

Confusion Matrix

Accuracy	97,14% +/- 6,02%
F measure	80,00%

Predicted	true false	true true	precision
pred false	63	1	98,44%
pred true	1	4	80,00%
recall	98,44%	80,00%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

Logistic Regression

Accuracy	95,81% +/- 4,25%
F measure	68,23%

Predicted	true false	true true	precision
pred false	2637	49	98,17%
pred true	72	130	64,35%
recall	97,34%	72,62%	

SVM LINEAR

- **Cluster 0**

Parametri migliori

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.01	0.688	0.688

Costi	C
0.6 – 0.4	0.01

0.55 – 0.45	0.01	0.688	0.688
0.6 – 0.4	0.01	0.688	0.688
0.7 – 0.3	0.001	0.686	0.684
0.8 – 0.2	0.4	0.315	0.158
0.9 – 0.1	10	0.114	0.112

Confusion Matrix

Accuracy	96,12% +/- 1,09%
F measure	68,82% +/- 8,95%

Predicted	true false	true true	precision
pred false	2460	44	98,24%
pred true	60	115	65,71%
recall	97,62%	72,33%	

- **Cluster 1**

Parametri migliori

Costi	C
0.7 – 0.3	0.01

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.1	0.759	0.758
0.55 – 0.45	0.1	0.759	0.758
0.6 – 0.4	0.1	0.759	0.758
0.7 – 0.3	0.01	0.733	0.758
0.8 – 0.2	1000	0.595	0.615
0.9 – 0.1	100	0.369	0.424

Confusion Matrix

Accuracy	95,00% +/- 4,82%
F measure	75,86%

Predicted	true false	true true	precision
pred false	122	4	96,83%
pred true	3	11	78,57%
recall	97,60%	73,33%	

- **Cluster 2**

Parametri migliori

Costi	C
0.55 – 0.45	0.4

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.8	0.800	0.800
0.55 – 0.45	0.4	0.800	0.800
0.6 – 0.4	0.4	0.800	0.727
0.7 – 0.3	0.01	0.727	0.727
0.8 – 0.2	0.8	0.455	0.416
0.9 – 0.1	100	0.357	0.333

Confusion Matrix

Accuracy	97,14% +/- 6,02%
F measure	80,00%

Predicted	true false	true true	precision
pred false	63	1	98,44%
pred true	1	4	80,00%
recall	98,44%	80,00%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

SVM linear

Accuracy	96,08% +/- 1,97%
F measure	69,70%

Predicted	**true false**	**true true**	**precision**
pred false	2645	49	98,18%
pred true	64	130	67,01%
recall	97,63%	72,62%	

SVM RBF

- **Cluster 0**

Parametri migliori

Costi	C	gamma
0.7 – 0.3	0.8	0.001

Costi	**C**	**gamma**	**F measure (Loop)**	**F measure (Crossvalidation)**
0.5 – 0.5	0.8	0.01	0.681	0.676
0.6 – 0.4	0.8	0.01	0.686	0.679
0.65 – 0.35	1	0.01	0.686	0.684
0.7 – 0.3	0.8	0.001	0.688	0.686
0.8 – 0.2	1000	0.2	0.606	0.597
0.9 – 0.1	1000	0.2	0.548	0.542

Confusion Matrix

Accuracy	96,08% +/- 1,10%
F measure	68,62% +/- 9,02%

Predicted	**true false**	**true true**	**precision**
pred false	2459	44	98,24%
pred true	61	115	65,34%
recall	97,58%	72,33%	

- **Cluster 1**

Parametri migliori

Costi	C	gamma
0.65 – 0.35	0.4	0.1

Costi	**C**	**gamma**	**F measure (Loop)**	**F measure (Crossvalidation)**
0.5 – 0.5	1	0.1	0.692	0.692
0.55 – 0.45	1000	0.001	0.800	0.667
0.6 – 0.4	0.8	0.1	0.810	0.727
0.65 – 0.35	0.4	0.1	0.800	0.774
0.7 – 0.3	1000	0.001	0.774	0.709
0.8 – 0.2	1000	0.001	0.661	0.521
0.9 – 0.1	1000	0.01	0.241	0.265

Confusion Matrix

Accuracy	95,00% +/- 6,78%
F measure	77,42%

Predicted	**true false**	**true true**	**precision**
pred false	121	3	97,58%
pred true	4	12	75,00%
recall	96,80%	80,00%	

- **Cluster 2**

Parametri migliori

Costi	C	gamma
0.7 – 0.3	0.1	0.1

Costi	C	gamma	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	1000	0.01	0.800	0.667
0.55 – 0.45	10	0.1	0.800	0.667
0.6 – 0.4	100	0.0001	0.800	0.800
0.65 – 0.35	0.8	0.01	0.800	0.727
0.7 – 0.3	0.1	0.1	0.800	0.800
0.8 – 0.2	1000	0.001	0.278	0.256
0.9 – 0.1	1000	0.01	0.164	0.144

Confusion Matrix

Accuracy	97,14% +/- 6,02%
F measure	80,00%

Predicted	true false	true true	precision
pred false	63	1	98,44%
pred true	1	4	80,00%
recall	98,44%	80,00%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

SVM radial

Accuracy	96,05% +/- 4,63%
F measure	69,67%

Predicted	true false	true true	precision
pred false	2643	48	98,21%
pred true	66	131	66,49%
recall	97,56%	73,18%	

SVM Polynomial

- **Cluster 0**

Parametri migliori

Costi	C	degree
0.6 – 0.4	0.001	2

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	2	0.685	0.681
0.55 – 0.45	0.001	2	0.684	0.681
0.6 – 0.4	0.001	2	0.682	0.681
0.7 – 0.3	0.001	2	0.670	0.678
0.8 – 0.2	0.1	2	0.606	0.642
0.9 – 0.1	0.1	2	0.449	0.548

Confusion Matrix

Accuracy	96,04% +/- 1,06%
F measure	68,18% +/- 8,73%

Predicted	true false	true true	precision
pred false	2459	45	98,20%
pred true	61	114	65,14%
recall	97,58%	71,70%	

- **Cluster 1**

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.01	2	0.759	0.714
0.55 – 0.45	0.001	2	0.710	0.285
0.6 – 0.4	0.1	2	0.690	0.709
0.65 – 0.35	0.001	2	0.759	0.800
0.7 – 0.3	0.001	2	0.706	0.705
0.8 – 0.2	1000	2	0.462	0.423
0.9 – 0.1	1000	2	0.260	0.225

Parametri migliori

Costi	C	degree
0.65 – 0.35	0.001	2

Confusion Matrix

Accuracy	95,71% +/- 4,99%
F measure	80,00%

Predicted	true false	true true	precision
pred false	122	3	97,60%
pred true	3	12	80,00%
recall	97,60%	80,00%	

- **Cluster 2**

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.01	3	0.800	0.800
0.55 – 0.45	0.1	2	0.800	0.800
0.6 – 0.4	0.001	2	0.800	0.800
0.7 – 0.3	0.001	2	0.727	0.727
0.8 – 0.2	10	2	0.303	0.303
0.9 – 0.1	1000	2	0.143	0.144

Parametri migliori

Costi	C	degree
0.6 – 0.4	0.001	2

Confusion Matrix

Accuracy	97,14% +/- 6,02%
F measure	80,00%

Predicted	true false	true true	precision
pred false	63	1	98,44%
pred true	1	4	80,00%
recall	98,44%	80,00%	

Facendo la media delle prestazioni dei singoli gruppi si ottiene la seguente matrice di confusione:

SVM Polynomial

Accuracy	96,16% +/- 2,86%
F measure	68,56%

Predicted	true false	true true	precision
pred false	2644	49	98,18%
pred true	65	130	66,67%
recall	97,60%	72,62%	

[6] *Under + Metacost + lag*

[6.1] Metacost + lag

- Lag temporale 2 anni
- [2313 (0) – 151 (1)], sbilanciamento 93,8%
- MetaCost 200 iterazioni

Decision Tree

Tabella delle combinazioni

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Gini	135	0.654	65.67% +/- 8.03
0.6 – 0.4	Information	16	0.676	67.74% +/- 8.07
0.7 – 0.3	Gini	90	0.683	67.50% +/- 7.92
0.75 – 0.25	Accuracy	46	0.682	67.50% +/- 7.92
0.8 – 0.2	Gini	105	0.682	67.50% +/- 7.92
0.9 – 0.1	Accuracy	150	0.679	67.50% +/- 7.92

Parametri migliori

Costi	Criterio	Profondità
0.8 – 0.2	Gini	105

Confusion Matrix

Accuracy	95.98% +/-0,84%
F measure	67.50% +/- 7.92%

predicted	true false	true true	precision
pred false	2261	47	97.96%
pred true	52	104	66,67%
recall	97,75%	68,87%	

Random Forest

Tabella delle combinazioni

Costi	Alberi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	161	Gain	61	0.635	59.34% +/- 8.70
0.6 – 0.4	81	Information	90	0.684	67.08% +/- 8.61
0.65 – 0.35	161	Information	31	0.686	67.72% +/- 8.15
0.7 – 0.3	81	Information	90	0.685	67.50%+/- 7.92
0.8 – 0.2	240	Gini	31	0.682	67.50%+/- 7.92
0.9 – 0.1	161	Accuracy	90	0.680	67.04% +/- 7.67

Parametri migliori

Costi	Criterio	Profondità	Profondità
0.65 – 0.35	161	Accuracy	31

Confusion Matrix

Accuracy	96,02% +/- 0,87%
F measure	67,72% +/- 8,15%

predicted	true false	true true	precision
pred false	2262	47	97,96%
pred true	51	104	67,10%
recall	97,80%	68,87%	

Logistica

Tabella delle combinazioni

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	IRLSM	0.629	61.38% +/- 8.12
0.6 – 0.4	AUTO	0.677	66.25% +/- 9.30
0.7 – 0.3	IRLMS	0.671	66.60% +/- 8.27
0.8 – 0.2	AUTO	0.678	67.28% +/- 7.79

Parametri migliori

Costi	Criterio
0.8 – 0.2	Auto

0.9 – 0.1	L_BFGS	0.646	64.18% +/- 7.90

Confusion Matrix

Accuracy	95.94% +/- 0,83%
F measure	67.28% +/- 7,79%

predicted	true false	true true	precision
pred false	2260	47	97.96
pred true	53	104	66.24%
recall	97,71%	68,87%	

SVM LINEAR

Tabella delle combinazioni

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.01	0.681	67.50% +/- 7.92
0.6 – 0.4	0.01	0.681	67.50% +/- 7.92
0.7 – 0.3	0.01	0.679	67.50% +/- 7.92
0.8 – 0.2	1	0.465	61.20 +/- 5.79
0.9 – 0.1	1	0.181	15.53% +/- 4.88

Parametri migliori

Costi	C
0.7 – 0.3	0.01

Confusion Matrix

Accuracy	95,98% +/- 0,84%
F measure	67,50% +/- 7,92%

predicted	true false	true true	precision
pred false	2261	47	97,96%
pred true	52	104	66,67%
recall	97,75%	68,87%	

SVM RBF

Tabella delle combinazioni

Costi	C	gamma	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	1	0.01	0.653	65.99% +/- 9.49
0.6 – 0.4	1	0.01	0.676	66.85% +/- 7.44
0.7 – 0.3	10	0.01	0.670	67.56% +/- 7.85
0.8 – 0.2	100	0.01	0.631	67.51% +/- 7.77
0.9 – 0.1	100	0.01	0.441	53.45% +/- 17.08

Parametri migliori

Costi	C	gamma
0.7 – 0.3	10	0.01

Confusion Matrix

Accuracy	95,98% +/- 0,88%
F measure	67,56% +/- 7,85%

predicted	true false	true true	precision
pred false	2261	47	97.96%
pred true	52	104	66.67%
recall	97,75%	68,87%	

SVM Polynomial

Tabella delle combinazioni

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	2	0.645	67.79% +/- 8.59
0.6 – 0.4	0.001	2	0.666	68.19%+/- 8.34
0.7 – 0.3	0.01	2	0.598	64.52% +/- 8.11
0.8 – 0.2	0.01	2	0.470	53.55% +/- 9.33

Parametri migliori

Costi	C	degree
0.6 – 0.4	0.001	2

0.9 – 0.1	10	2	0.225	22.81% +/- 2.79

Confusion Matrix

Accuracy	96,10% +/- 0,90%
F measure	68,19% +/- 8,34%

predicted	true false	true true	precision
pred false	2264	47	97,97%
pred true	49	104	67,97%
recall	97,88%	68,87%	

[6.2] Under + Metacost + lag

- Undersampling 15%
- Lag temporale di due anni
- [346 (0) – 151 (1)], sbilanciamento 69,6%
- K = 4
- MetaCost 200 iterazioni

Decision Tree

Tabella delle combinazioni

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Gini	76	0.680	67.25% +/- 7.39
0.55 – 0.45	Accuracy	16	0.680	66.63% +/- 7.39
0.6 – 0.4	Accuracy	90	0.682	67.07% +/- 7.83
0.7 – 0.3	Accuracy	76	0.677	66.64% +/- 7.68
0.8 – 0.2	Accuracy	90	0.653	62.47% +/- 10.21
0.9 – 0.1	Gini	46	0.245	25.43% +/- 3.68

Parametri migliori

Costi	Criterio	Profondità
0.5 – 0.5	Gini	76

Confusion Matrix

Accuracy	95.98% +/-0,75%
F measure	67.25% +/- 7.39%

predicted	true false	true true	precision
pred false	2262	48	97.92%
pred true	51	103	66,88%
recall	97,80%	68,21%	

Random Forest

Tabella delle combinazioni

Costi	Alberi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	320	Accuracy	31	0.682	67.50% +/- 7.92
0.55 – 0.45	81	Accuracy	31	0.679	67.50% +/- 7.92
0.6 – 0.4	81	Accuracy	61	0.666	66.07% +/- 8.11
0.7 – 0.3	81	Accuracy	31	0.541	52.69% +/- 7.35
0.8 – 0.2	161	Accuracy	61	0.318	30.17% +/- 8.97
0.9 – 0.1	161	Gaino	61	0.169	16.40% +/- 1.41

Parametri migliori

Costi	Alberi	Criterio	Profondità
0.55 – 0.45	81	Accuracy	31

Confusion Matrix

Accuracy	95.98% +/-0,84%
F measure	67.50% +/- 7.92%

predicted	true false	true true	precision

pred false	2261	47	97.96%
pred true	52	104	66,67%
recall	97,75%	68,87%	

Logistica

Tabella delle combinazioni

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	L_BFGS	0.694	61.04% +/- 8.70
0.55 – 0.45	L_BFGS	0.688	57.83% +/- 8.31
0.6 – 0.4	L_BFGS	0.686	54.42% +/- 8.09
0.7 – 0.3	Coordinate_Descent_Naive	0.622	42.54% +/- 5.53
0.8 – 0.2	AUTO	0.431	30.34% +/- 3.45
0.9 – 0.1	L_BFGS	0.221	20.89% +/- 2.06

Parametri migliori

Costi	Criterio
0.5 – 0.5	L_BFGS

Confusion Matrix

Accuracy	94,72% +/- 1,21%
F measure	61,04% +/- 8,70%

predicted	true false	true true	precision
pred false	2232	49	97,85%
pred true	81	102	55,74%
recall	96,50%	67,55%	

SVM LINEAR

Tabella delle combinazioni

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	1	0.674	66.40% +/- 7.52
0.55 – 0.45	0.01	0.677	67.28% +/- 7.79
0.6 – 0.4	0.01	0.672	67.28% +/- 7.79
0.65 – 0.35	0.01	0.672	67.05% +/- 7.62
0.7 – 0.3	0.1	0.674	65.83% +/- 8.14
0.8 – 0.2	0.1	0.6	59.98% +/- 8.24
0.9 – 0.1	1	0.192	20.49% +/- 5.42

Parametri migliori

Costi	C
0.6 – 0.4	0.01

Confusion Matrix

Accuracy	95,94% +/- 0,83%
F measure	67,28% +/- 7,79%

predicted	true false	true true	precision
pred false	2260	47	97,96%
pred true	53	104	66,24%
recall	97,71%	68,87%	

SVM RBF

Tabella delle combinazioni

Costi	C	gamma	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	1	0.01	0.648	65.17% +/- 8.55
0.55 – 0.45	100	0.01	0.634	63.88% +/- 10.24
0.6 – 0.4	100	0.01	0.635	65.55% +/- 9.24
0.7 – 0.3	100	0.01	0.531	57.98% +/- 10.21
0.8 – 0.2	100	0.01	0.353	29.76% +/- 8.36
0.9 – 0.1	100	0.01	0.212	14.66% +/- 2.32

Parametri migliori

Costi	C	gamma
0.6 – 0.4	100	0.01

Confusion Matrix

Accuracy	95,82% +/- 0,89%
F measure	65,55% +/- 9,24%

predicted	true false	true true	precision
pred false	2261	51	97,79%
pred true	52	100	65,79%
recall	97,75%	66,23%	

SVM Polynomial

Tabella delle combinazioni

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.001	2	0.509	51.16% +/- 11.35
0.55 – 0.45	0.001	2	0.541	53.32% +/- 12.95
0.6 – 0.4	0.001	2	0.491	48.13% +/- 8.72
0.7 – 0.3	0	4	0.332	33.73% +/- 9.60
0.8 – 0.2	0.1	2	0.145	14.11% +/- 1.48
0.9 – 0.1	100	2	0.122	12.12% +/- 0.77

Parametri migliori

Costi	C	degree
0.55 – 0.45	0.001	2

Confusion Matrix

Accuracy	94,20% +/- 1,47%
F measure	53,32% +/- 12,95%

predicted	true false	true true	precision
pred false	2238	68	97,05%
pred true	75	83	52,53%
recall	96,76%	54,97%	

Under + Metacost + lag

- Undersampling 25%
- [578 (0) – 151 (1)], sbilanciamento 79,2%
- K = 4
- MetaCost 200 iterazioni
- Lag temporale di due anni

Decision Tree

Tabella delle combinazioni

Costi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Accuracy	76	0.680	66.63% +/- 7.39
0.6 – 0.4	Accuracy	31	0.680	67.07% +/- 7.83
0.7 – 0.3	Gini	135	0.682	67.07% +/- 7.83
0.75 – 0.25	Accuracy	61	0.679	67.07% +/- 7.83
0.8 – 0.2	Accuracy	61	0.682	67.07% +/- 7.83
0.9 – 0.1	Accuracy	120	0.652	64.07% +/- 11.47

Parametri migliori

Costi	Criterio	Profondità
0.7 – 0.3	Gini	135

Confusion Matrix

Accuracy	95.94% +/-0,83%

F measure	67.07% +/- 7.83%		
predicted	**true false**	**true true**	**precision**
pred false	2261	48	97.92%
pred true	52	103	66,45%
recall	97,75%	68,21%	

Random Forest

Tabella delle combinazioni

Costi	Alberi	Criterio	Profondità	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	240	Gini	31	0.682	67.50% +/- 7.92
0.55 – 0.45	240	Gini	31	0.682	67.50% +/- 7.92
0.6 – 0.4	240	Gini	31	0.680	67.31% +/- 8.01
0.7 – 0.3	240	Accuracy	90	0.674	67.29% +/- 8.10
0.8 – 0.2	320	Accuracy	61	0.500	50.04% +/- 9.31
0.9 – 0.1	161	Information	90	0.237	11.80% +/- 1.11

Parametri migliori

Costi	Alberi	Criterio	Profondità
0.55 – 0.45	240	Gini	31

Confusion Matrix

Accuracy	95.98% +/-0,84%		
F measure	67,50% +/- 7,92%		
predicted	**true false**	**true true**	**precision**
pred false	2261	47	97.96%
pred true	52	104	66,67%
recall	97,75%	68,87%	

Logistica

Tabella delle combinazioni

Costi	Criterio	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	Coordinate_Descent	0.672	65.08% +/- 7.28
0.55 – 0.45	Coordinate_Descent	0.659	64.06% +/- 6.75
0.6 – 0.4	L_BFGS	0.648	63.46% +/- 8.50
0.7 – 0.3	L_BFGS	0.597	58.44% +/- 7.76
0.8 – 0.2	Coordinate_Descent	0.474	44.48% +/- 5.47
0.9 – 0.1	Coordinate_Descent	0.279	27.9% +/- 2.60

Parametri migliori

Costi	Criterio
0.5 – 0.5	Coordinate_Descent

Confusion Matrix

Accuracy	95,66% +/-0,76%		
F measure	65,08% +/- 7,28%		
predicted	**true false**	**true true**	**precision**
pred false	2256	50	97,83%
pred true	57	101	63,92%
recall	97,54%	66,89%	

SVM LINEAR

Tabella delle combinazioni

Costi	C	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.01	0.677	67.50% +/- 7.92
0.55 – 0.45	0.01	0.677	67.50% +/- 7.92
0.6 – 0.4	0.01	0.675	67.26% +/- 7.76
0.7 – 0.3	0.01	0.669	66.80% +/- 7.33
0.8 – 0.2	0.8	0.537	57.00% +/- 6.34

Parametri migliori

Costi	C
0.55 – 0.45	0.01

0.9 – 0.1	10	0.124	11.57% +/- 0.52

Confusion Matrix

Accuracy	95,98% +/-0,84%
F measure	67,50% +/- 7,92%

predicted	true false	true true	precision
pred false	2261	47	97,96%
pred true	52	104	66,67%
recall	97,75%	68,87%	

SVM RBF

Tabella delle combinazioni

Costi	C	gamma	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.4	0.01	0.673	66.67% +/- 9.02
0.55 – 0.45	0.4	0.01	0.670	66.30% +/- 8.90
0.6 – 0.4	10	0.01	0.662	65.73% +/- 7.24
0.7 – 0.3	100	0.01	0.661	63.30% +/- 7.65
0.8 – 0.2	100	0.01	0.558	37.86% +/- 5.98
0.9 – 0.1	100	0.01	0.273	23.16% +/- 4.61

Parametri migliori

Costi	C	gamma
0.5 – 0.5	0.4	0.01

Confusion Matrix

Accuracy	95,94% +/-0,89%
F measure	66,73% +/- 9,02%

predicted	true false	true true	precision
pred false	2262	49	97,88%
pred true	51	102	66,67%
recall	97,80%	67,55%	

SVM Polynomial

Tabella delle combinazioni

Costi	C	degree	F measure (Loop)	F measure (Crossvalidation)
0.5 – 0.5	0.01	2	0.612	58.16% +/- 8.79
0.55 – 0.45	0.001	2	0.589	63.35% +/- 7.87
0.6 – 0.4	0.001	2	0.641	63.84% +/- 7.22
0.7 – 0.3	0.001	3	0.393	40.01% +/- 5.07
0.8 – 0.2	0.4	2	0.203	18.34% +/- 1.79
0.9 – 0.1	0.4	2	0.142	13.14% +/- 0.88

Parametri migliori

Costi	C	degree
0.6 – 0.4	0.001	2

Confusion Matrix

Accuracy	95,41% +/-0,74%
F measure	63,84% +/- 7,22%

predicted	true false	true true	precision
pred false	2250	50	97,83%
pred true	63	101	61,59%
recall	97,28%	66,89%	

[7] Reti LSTM

Confusion Matrix

Accuracy	95,41% +/-0,74%
F measure	63,84% +/- 7,22%

predicted	true false	true true	precision
pred false	2250	50	97,83%
pred true	63	101	61,59%
recall	97,28%	66,89%	

Ringraziamenti

"Desidero esprimere la mia sincera gratitudine a tutti coloro che hanno contribuito al mio percorso accademico e alla realizzazione di questa tesi di laurea. Ringrazio i miei professori per la loro guida, il supporto e la conoscenza che hanno condiviso con me. Un ringraziamento ai miei genitori per il loro amore incondizionato e il costante sostegno durante tutto il mio percorso di studi.

Un ringraziamento speciale va alla mia ragazza Francesca, che è stata sempre al mio fianco, incoraggiandomi e spingendomi a superare le sfide con resilienza, tenacia e a non mollare mai.

Grazie ai miei amici e colleghi per le esperienze condivise, le discussioni stimolanti e il supporto reciproco.

Come un ingegnere che unisce diverse parti per creare un meccanismo funzionante, così ogni "ingranaggio" delle vostre azioni e supporto hanno contribuito a rendere questo giorno reale. Grazie di cuore per rendere questo momento speciale. "

Bibliografia

Alessi L. and Detken C., Identifying excessive credit growth and leverage. *Journal of Financial Stability*, (2018).

Antunes A., Bonfim D. et al., Forecasting banking crises with dynamic panel probit models. *International Journal of Forecasting*, (2018).

Beutel J., List S. and von Schweinitz G., Does machine learning help us predict banking crises? *Journal of Financial Stability*, (2019).

Bluwstein K., Buckmann M. et al., Credit growth, the yield curve and financial crises prediction: ecidence from a machine learning approach. *Elsevier*, (2023).

Brownlee K., Buckmann M. et al., Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning. *Machine Learning Mastery*, (2020).

Caggiano G., Calice P. et al., Comparing logit-based early warning systems: Does the duration of systemic banking crises matter? *Journal of Empirical finance*, (2016).

Casabianca E. J., Catalano M. et al., An early warning system for banking crises: From regression-based analysis to machine learning techniques. *EconPapers. Orebro: Orebro University* , (2019).

Chawla N. V. et al., *SMOTE: sythetic minority over-sampling technique.* Journal of artificial intelligent research, (2002).

Davis E. P. and Karim D., Comparing early warning systems for banking crises. *Journal of Financial stability*, (2008).

Demirgüç-Kunt A. and Detragiache E., The determinants of banking crises in developing and developed countrues. *Staff Papers*, (1998).

Domingos P., Metacost: A general method for making classifiers cost-sensitive. *In Proceedings of the fith ACM SIGKDD international conference on Knowledge discovery and data mining,* (1999).

Drehmann M. and Tsatsaronis K., The Credit-to-GDP and Countercyclical Capital Buffers: Questions and Answers. *SSRN,* (2014).

Guerra G. and Castelli M., Machine Learning Applied to Banking Supervision a Literature Review. *MDPI*, (2021).

Guerra G., Castelli M. and Côrte-Real N., Machine learning for liquidity risk modelling: A supervisory perspective. *Economic Analysis and Policy*, (2022).

Holopainen M. and Sarlin P., Toward robust early-warnings models: A horse race, ensembles and model uncertainty. *Quantitative Finance*, (2017).

Hyndman R. J. and Athanasopoulos G., Forecasting: principles and practice. *OTexts*, (2018).

Kaminsky G., Lizondo S. and Reinhart C. M., Leading indicators of currency crises. *Staff Papers*, (1998).

Kim J., Choi K. and Kim G., Classification cost: An empirical comparison amont traditional classifier, Cost-Sensitive Classifier, and MetaCost. *Elsevier*, (2012).

Laeven K. L. and Valencia M. F., Systemic banking crises revisited. *International Monetary Fund*, (2018).

Lin S. L. and Jin X., Does ESG Predict Systemic Banking Crises? A computational economics model of Early Warning Systems with interpretable multivariable LSTM based on Mixture attention. *MDPI*, (2023).

Menardi G. and Torelli N., Training and assessing classification rules with imbalanced data. *Springer Link,* (2012).

Pigini C., Penalized maximum likelihood estimation of logit-based early warning systems. *International Journal of Forecasting*, (2021).

Rocchi R., Classificatori sampling based per eventi rari: un'applicazione agli early warning systems per crisi finanziarie. *univpm.it,* (2022).

Satyasree K. and Murthy J., An exhaustive literature review on class imbalance problem. *International Journal of Emerging Trends of tecnology in computer science*, (2013).

Schularick M. and Taylor A. M., Creit booms gone bust: Monetary policy, levarage cycles, and financial crises. *American Economic Review*, (2012).

Shapley L. S., Stochastic games. *Proceedings of the national academy of science*, (1953).

Katsuyuki T., Takuji K. and Shiheyuki H., Random forest-based early warning system for bank failures. *Elsevier*, (2016).

Thehindu B., What is the gdp deflator? *www.thehindu.com/*, 2018.

Tölö E., Predicting systemic financial crises with recurrent neural networks. *Journal of Financial Stability*, (2020).

Torky M. and Gad A., Explainable AI model for recognizing Financial Crises roots based on Pigeon Optimization and gradient boosting model. *International Journal of Computational Intelligence Systems*, (2023).

LSTM Recurrent Neural Networks, www.towardsdatascience.com/ , *Towardsdatascience*, 2022

Wongvorachan T, He S. and Bulut O., A comparison of Undersampling, Oversampling, and SMOTE methods for dealing with Imbalanced Classification in Educational data mining. *MDPI*, (2023).

www.worldbank.org/en/publication/gfdr/gfdr-2016/background/banking-crisis, *The World Bank*, (2016).