POLITECNICO DI TORINO

Corso di Laurea
in Matematica per l'Ingegneria

Tesi di Laurea

Metodi di ricampionamento per l'analisi dei dati

Relatore
prof. Francesco Vaccarino
firma dei relatore

. . . . . .
. . . . . .

Candidato
Filippo Grobbo
firma del candidato

. . . . . .

Anno Accademico 2020-2021

Sommario

I data set sono particolari matrici che servono per raccogliere dati di diverso interesse. Ad ogni osservazione è associata una diversa classe di appartenenza. Nelle applicazioni a casi reali spesso i dati sono sbilanciati, ossia le classi non sono equamente distribuite. Di fronte a questi problemi l'analisi dati risulta essere difficile in quanto molti algoritmi non riescono a predirre correttamente la classe di appartenenza delle osservazioni. A questo scopo sono stati definiti i metodi di ricampionamento. Essi permettono di generare dati mancanti in modo da pareggiare le classi all'interno di un data set. Questo è un aspetto fondamentale per gli algoritmi di previsione. Infatti, nella fase di addestramento di un algoritmo, esso deve essere in grado di poter riconoscere tutte le possibili classi. Soltanto in questo modo sarà possibile fare delle previsioni accurate su dati nuovi.

In questo elaborato vengono analizzate alcune delle principali strategie di ricampionamento: random oversampling, random undersampling, tomek's links, sythetic minority oversampling tecniques (smote) e adaptive syntethic sampling (adasyn). Inizialmente vengono presentati i metodi dal punto di vista teorico, spiegandone i vantaggi, limiti e possibili estensioni. Successivamente vi è una applicazione pratica su un data set reale riguardante lo studio dei financial ratios come indicatori dello stato di bancarotta. I risultati ottenuti applicando algoritmi predittivi al data set bilanciato evidenziano la validità di tali strategie di ricampionamento.

Ringraziamenti

Ringrazio la mia famiglia per avermi permesso di intraprendere questa carriera universitaria e chi mi è stato vicino anche al di fuori degli studi. Un ringraziamento anche al professor Vaccarino che mi ha concesso di sviluppare questo elaborato nonostante non ci fossimo mai conosciuti direttamente.

Indice

Elenco delle figure

Capitolo 1

Generalità sul bilanciamento dei dati

1.1 Introduzione al problema

Il machine learning è una branca dell'intelligenza artificiale che si occupa della costruzione di meccanismi che simulano il processo di apprendimento. Esistono diversi paradigmi di apprendimento a seconda di quale sia l'obiettivo finale.

Nell'apprendimento supervisionato lo scopo principale è quello di dedurre una funzione a partire dai dati a disposizione. Questi ultimi vengono collezionati in particolari matrici chiamate data set. Un data set D è un insieme formato da m oggetti, detti osservazoni, rappresentato sinteticamente come:

$$D = \{(x_i, y_i), i = 1,2, ..., m\} \tag{1.1}$$

dove x_i è un vettore (o matrice) di features che rappresentano le caratteristiche di una particolare osservazione mentre y_i rappresenta la classe di appartenenza dell'osservazione (spesso indicato come label). Ogni coppia (x_i,y_i) ha la stessa forma e dimensione per ogni i=1,2,...,m dove m rappresenta il numero totale di osservazioni. Il fine dell'analisi è quello di determinare una funzione ϕ tale che:

$$\phi(x_i) \simeq y_i \tag{1.2}$$

Questa mappa ϕ, spesso chiamata decision function, viene utilizzata per fare previsioni o elaborare informazioni. Il ruolo di D è fondamentale per poter determinare una decision function che faccia previsioni corrette. Infatti, negli algoritmi di machine learning, ϕ viene determinata a seguito di un addestramento eseguito su una porzione di D detta training set. Successivamente, si eseguono dei test sui dati restanti per verificare la bontà della ϕ precedentemente ottenuta. Le prestazioni della decision function possono essere valutate in diversi modi, solitamente in termini di accuratezza, cioè la percentuale di campioni predetti correttamente. Nella fase di addestramento ϕ deve essere in grado di poter riconoscere le diverse classi del campione in questione. Uno sbilanciamento nelle classi spesso porta dei problemi nella maggior parte degli algortimi di machine learning.

In Chawla et al. [2002] viene considerato ad esempio il problema di classificazione dei pixel in un'immagine di una mammografia possibilmente cancerogena. Un tipico dataset associato a queste immagini contiene circa il 98% di pixel normali e il 2% di pixel anormali. Una strategia molto semplice potrebbe essere quella di predirre sempre la classe di maggioranza, ossia i pixel normali. Costruendo una ϕ di questo genere si avrebbe un'accuratezza del 98%, cioè molto alta. Tuttavia, la natura del problema richiede che ci sia una percentuale di accuratezza maggiore nel predirre gli elementi della classe minoritaria che in questo caso sarebbe nulla. Quindi, di fronte a problemi che presentano classi sbilanciate, gli algoritimi di machine learning possono comportarsi erroneamente in termini predittivi dato che le decision function tendono a favorire le classi con il maggior numero di osservazioni. Lemaître et al. [2017].

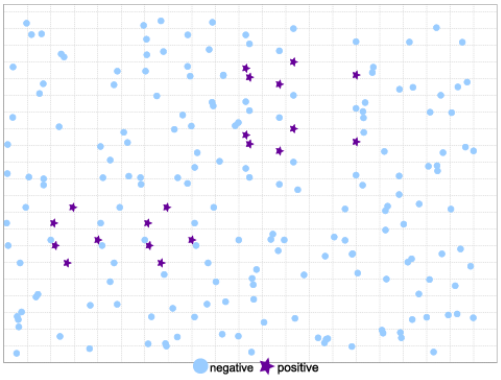
A questo proposito, il problema del bilanciamento dei dati si è sempre di più sviluppato col passar del tempo dato che nella maggior parte delle applicazioni a casi reali i data set presentano classi non equamente distribuite. Si è iniziata quindi a sviluppare un'area del machine learning nota da Fernández et al. [2018] come learning from imbalanced data, ossia l'apprendimento da dati sbilanciati.

Come già intuibile dall'esempio, i motivi principali per affrontare questo tipo di problemi sono: l'interesse per la classe minoritaria e le istanze rare. Lo studio della classe minoritaria sorge in domini dove queste osservazioni sono di grande interesse e quindi l'obiettivo degli algoritmi di machine learning diventa classificare la classe minoritaria nel modo più accurato possibile. Il problema delle istanze rare riguarda invece situazioni dove dati che rappresentano un tipico evento, come ad esempio la percentuale di pixel anormali, sono limitati in confronto ad altri. He et al. [2008]

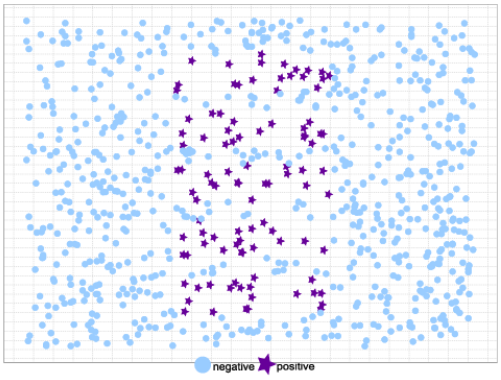
1.2 Cause del fallimento degli algoritimi

L'esempio precedente chiarifica come il corretto bilanciamento dei dati sia un punto critico per lo sviluppo di algoritmi nel machine learning. In generale uno sbilanciamento può portare ad una degradazione delle prestazioni in termini predittivi. Tuttavia, bisogna sottolineare che questa non è l'unica causa di fallimento degli algoritmi. La sua combinazione con caratteristiche intrinseche dei dati e del problema è ciò che porta allo sviluppo di modelli subottimali. Alcuni di questi problemi possono derivare ad esempio dalla presenza dei cosiddetti piccoli disgiunti, dalla mancanza di dati, dalla sovrapposizione delle classi oppure dalla diversa distribuzione interna delle classi. Fernández et al. [2018]

Spesso si fa riferimento ai piccoli disgiunti come small disjuncts. Un data set li contiene quando un concetto, a prescindere dalla classe di appartenenza, è rappresentato in piccoli gruppi. Nel caso di dati sbilanciati questo capita spesso dato che concetti sottorappresentati si trovano in piccole aree del data set. A titolo di esempio si possono osservare due situazioni diverse:



(a) Artificial dataset: small disjuncts for the minority classt



(b) Subclus dataset: small disjuncts for both classes

Figura 1.1. Esempio piccoli disgiunti

Nella Figura 1.1 (a) si possono osservare quattro small disjuncts per la classe minoritaria, mentre nella (b) sono presenti small disjuncts per entrambe le classi. In quest'ultimo caso infatti la classe maggioritaria presenta meno osservazioni nella regione rettangolare centrale dove è prevalente la presenza della minoritaria. Allo stesso tempo la classe minoritaria è presente solo in una piccola parte dell'intero data set. In queste situazioni potrebbe succedere ad esempio che un algoritmo di classificazione, detto anche classificatore, consideri alcune osservazioni di una certa classe come errori visto che sono situate nell'area di prevalenza delle classi opposte. Potenzialmente il modello potrebbe ignorare questi piccoli gruppi della classe minoritaria sebbene siano corretti e possano contere informazioni importanti.

La mancanza di dati è un concetto strettamente correlato con i small disjuncts. Solitamente infatti un'informazione è scarsamente rappresentata all'interno di un small disjunct. Di conseguenza, in fase di apprendimento, un classificatore non riuscirà a distinguere con grande certezza le classi di appartenenza delle osservazioni in queste aree del data set. Questo si riflette a sua volta in un peggioramento delle prestazioni.

La sovrapposizione delle classi invece è una caratteristica intrinseca dei dati molto importante. A titolo di esempio si può prendere un dataset D che presenti due classi fortemente sbilanciate ma linearmente separabili. Due classi si dicono linearmente separabili se esistono $w \in R^n$, $b \in R$ tali che $y_i(< w, x_i > +b) \geq 0 \; \forall i = 1,2...,m$, dove $(x_i, y_i) \in D$). Sotto queste ipotesi diventa facile per molti classificatori binari discriminare le diverse classi sebbene siano sbilanciate. Il problema sorge quindi quando si ha una sovrapposizione, come succede nella maggior parte delle applicazioni reali. A questo proposito si può definire una metrica per calcolare il grado di sovrapposizione di D. Per ogni feature si definsce il Fisher's Dicriminant Ratio, o semplicemente F1 come:

$$f = \frac{(\mu_1 - \mu_2)^2}{\delta_1^2 + \delta_2^2} \tag{1.3}$$

dove μ_1, μ_2, δ_1^2, δ_2^2 rappresentano le medie e le varianze delle due classi. F1 del dataset viene calcolato prendendo il massimo tra i valori precedentemente ottenuti. Minore è il suo valore maggiore sarà il grado di sovrapposizione delle classi, come si può osservare nelle seguenti immagini:

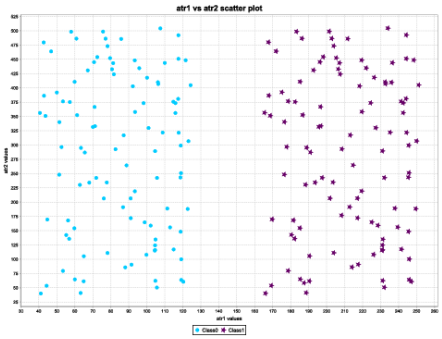


Figure 4: $F1 = 12.5683$

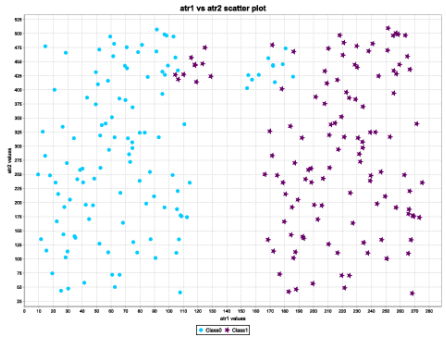
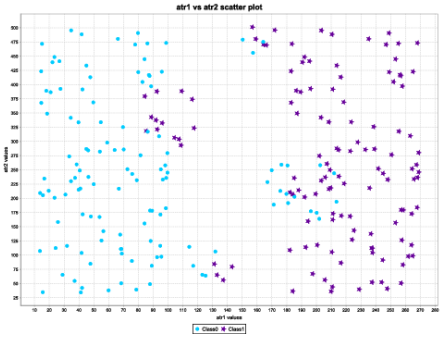


Figure 5: $F1 = 5.7263$



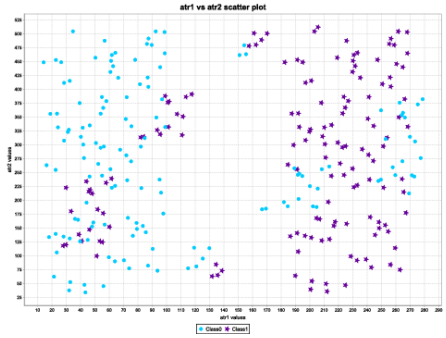


Figura 1.2. Sovrapposizione delle classi e F1

Il problema della sovrapposizione delle classi è inoltre collegato con il concetto di osservazioni sul bordo . Quest'ultime sono le osservazioni collocate nei confini tra le classi, cioè dove le classi si sovrappongono. In questi casi è fondamentale riuscire a distinguere quando osservazioni sul bordo rappresentano errori o informazioni utili. In questo modo si potranno scartare le osservazioni ingannevoli concentrandosi su quelle più difficili da classificare.

Infine, la diversa distribuzione interna delle classi può essere intesa in vari modi. Potrebbe succedere che la distribuzione delle classi sia diversa tra training set e test set. Analogamente potrebbe accadere con le features al posto delle classi. Nel caso invece di data set non stazionari, ossia insiemi dove le osservazioni arrivano in modo continuo e non sono fissate, le relazioni tra features e classi potrebbero variare all'aumentare dei dati a disposizione.

Capitolo 2

Metodi di ricampionamento

2.1 Come risolvere il problema del bilaciamento

Per trattare il problema del bilanciamento dei dati si possono utilizzare diversi approcci, i principali citati da He et al. [2008] sono:

1. strategie di ricampionamento: metodi che sviluppano tecniche di oversampling e undersampling per pareggiare la distribuzione iniziale del data set. L'oversampling permette di generare nuove osservazioni della classe che è sottorappresentata. L'undersampling invece permette di diminuire la proporzione della classe maggioritaria fino a che non ci sia un numero di osservazioni pari a quello della classe minoritaria.

2. generazione sintetica di dati: queste tecniche prevedono di superare lo sbilanciamento iniziale dei dati generando artificialmente le nuove osservazioni.

3. assegnazione di costi alle osservazioni: al posto di creare dei data set aventi classi bilanciate questi metodi si occupano di assegnare una matrice di costi per ogni errore di classificazione di una osservazione. Di conseguenza, la distribuzione iniziale dei dati non viene modificata.

2.2 Generalità sull'oversampling e undersampling

L'oversampling e l'undersampling rappresentano i concetti che sono la base delle strategie di ricampionamento.

Il random oversampling duplica le osservazioni della classe minoritaria scegliendole in modo casuale e non singolarmente una alla volta. Una sua variante permette anche di avere un parametro di dispersione che consente di non generare le stesse identiche osservazioni. Lemaître et al. [2017]

Per quanto riguarda l'undersampling in generale vengono distinte due varianti. La prima fa riferimento ad algoritmi generativi, dove a partire da un data set D sbilanciato si genera un nuovo set D' tale che $|D| < |D|$ e $D' \not\subset D$. In altre parole vengono ridotte le osservazioni della classe maggioritaria ma queste vengono create nuovamente. La seconda

variante invece fa riferimento ad algoritmi selettivi: si genera un data set D' tale che $|D| < |D|$ e $D' \subset D$. In questo caso le osservazioni vengono quindi selezionate e non ricreate. All'interno degli algoritmi selettivi si possono distinguere a sua volta altri due gruppi: le tecniche di undersampling controllato e le tecniche di undersampling di pulizia. Le prime fanno riferimento a metodi dove la cardinalità di D' è specificata. Rientra in questa categoria ad esempio il random undersampling che rimuove in modo casuale un sottoinsieme dei dati dalla classe di riferimento. Sia il random undersampling che il random oversampling permettono di campionare dati eterogenei, contenenti ad esempio stringhe, e nel caso di un problema multiclasse il metodo è applicato a ciascuna classe considerandole indipendenti. Nelle tecniche di undersampling di pulizia invece non è possibile specificare il numero di osservazioni per ogni classe dato che l'obiettivo pricipale è quello di ripulire lo spazio delle features. Fanno parte di questo gruppo i Tomek's links che verranno approfonditi meglio in seguito.

Nonostante queste tecniche siano state le prime soluzioni proposte per affrontare il problema del bilanciamento dei dati esse presentano dei limiti. Come analizzato in Fernández et al. [2018] il random undersampling ad esempio potrebbe rimuovere delle osservazioni importanti. Inoltre, se il tasso di sbilanciamento dei dati risulta essere molto alto, verrebbero rimosse molte osservazioni con una conseguente una mancanza di dati. Questo potrebbe influenzare le performance degli algoritmi in termini predittivi. D'altro canto, per quanto riguarda il random oversampling, la generazione di nuovi dati implica un aumento dei costi e dei pesi della classe minoritaria. A livello di prestazioni questo si riflette nel fenomeno dell'overfitting, ovvero un eccessivo addestramento dell'algoritmo rispetto al training set con conseguente scarsa capacità di generalizzazione delle nuove osservazioni Dirk P. Kroese [2019]. Un interpretazione intuitiva di questo fenomeno può essere vista in termini di regioni di decisione, come mostrato nella figura 2.1.

Le immagini sono plot dei dati estratti dal dataset sulla classificazione dei pixel di una mammografia possibilmente cancerogena. Nella figura 2.1 (a) si osserva che la regione di decisione rettangolare corrisponde ad osservazioni della classe maggioritaria sebbene al suo interno ci siano dei dati della classe minoritaria (i cosiddetti falsi negativi). A seguito dell'oversampling le regioni di decisione diventano molto più specifiche e piccole (figura 2.1 (b)). Durante la fase di addestramento infatti gli algoritmi si focalizzeranno tanto sulle osservazioni duplicate non essendo poi in grado di prevederne altre.

Esistono infine altre varianti delle tecniche sopra presentate che sono state introdotte in precedenti lavori sul trattamento di dataset sbilanciati Chawla [2005]. Ad esempio, il campionamento focalizzato (focused sampling) consiste nell'applicare le strategie di ricampionamento con particolari eccezioni. Il focused oversampling concentra la generazione di dati nel confine tra la classe minoritaria e quella maggioritaria, mentre il focused undersampling rimuove le osservazioni più distanti interpretandole come di minore importanza. Oltre a questo sono state provate anche combinazioni di oversampling della classe minoritaria parallelamente all'undersampling della classe maggioritaria. Tutti questi metodi sono stati testati su diversi data set e non hanno riscosso risutati particolarmente significativi in termini di migliori performace. Tuttavia sono stati il punto di partenza per lo sviluppo di nuove tecniche rivelatesi successivamente molto efficaci.

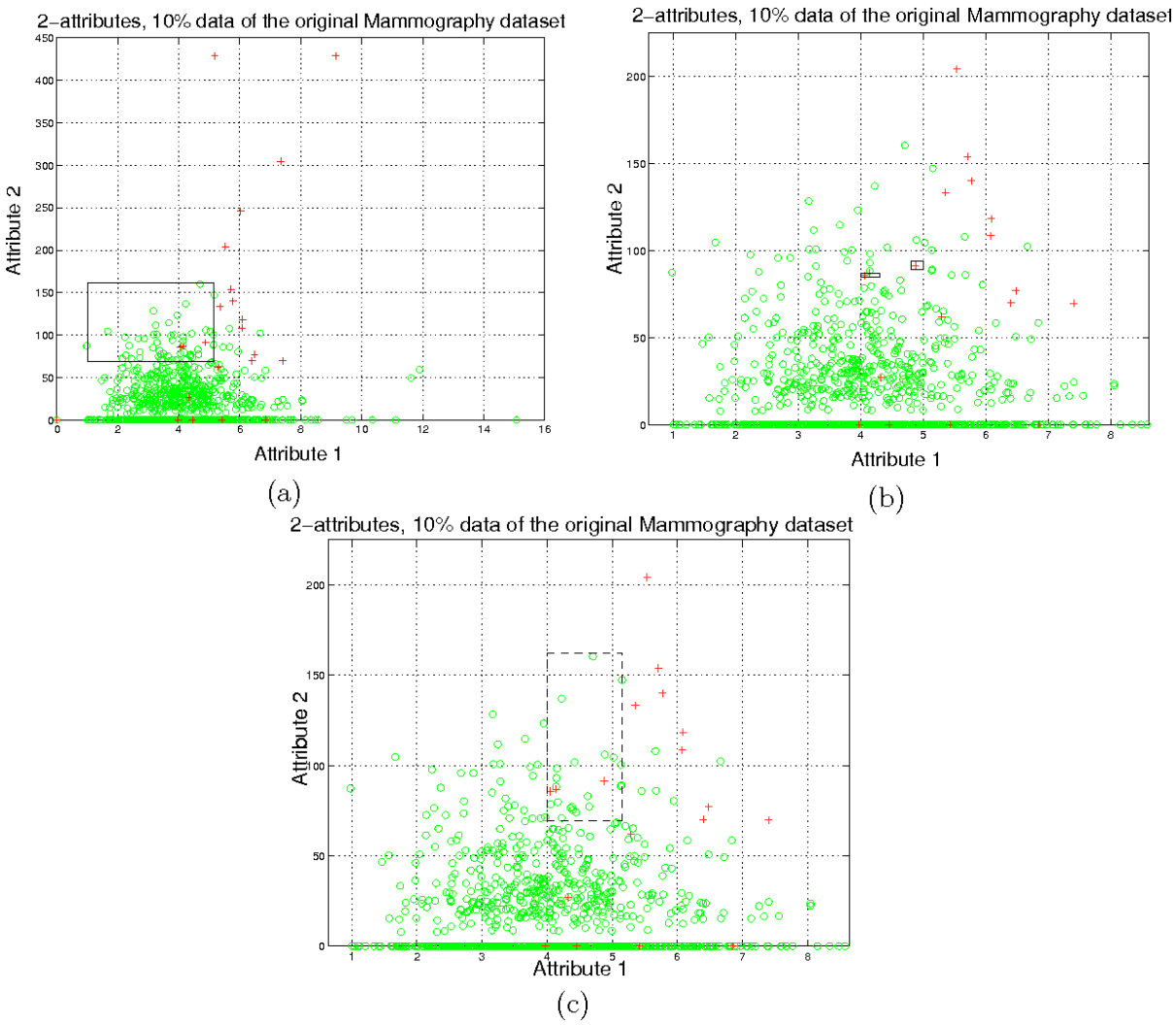


Figura 2.1. Regioni di decisione

2.3 Tomek's Links

I Tomek's Links sono una tecnica di undersampling selettivo di pulizia. Essi permettono di rimuover elementi che si sovrappongono tra le diverse classi. Più precisamente, un Tomek's link è una coppia di osservazioni x e y appartenenti a classi differenti tale che per qualsiasi altra osservazione z valga $d(x, y) < d(x, z)$ e $d(x, y) < d(y, z)$ dove $d(.)$ è la distanza tra due osservazioni. L'esistenza di queste coppie è legata alla presenza di osservazioni che sono rispettivamente l'una la più vicina dell'altra. Una volta trovato un Tomek's link si prosegue eliminando l'osservazione corrispondente alla classe maggioritaria. Esistono varianti del metodo che prevedono la rimozione di entrambe le osservazioni. Lemaître et al. [2017]

Risulta chiaro come l'esistenza dei Tomek's link sia indice di sovrapposizione tra classi. Rimuovendo questi dati all'interno del training set si possono creare dei gruppi tra osservazioni appartenenti alla stessa classe e permettere quindi un migliore addestramento degli algoritmi con conseguente incremento delle performance. A titolo illustrativo si può osservare nelle seguenti figure il principio di funzionamento dei Tomek's links:

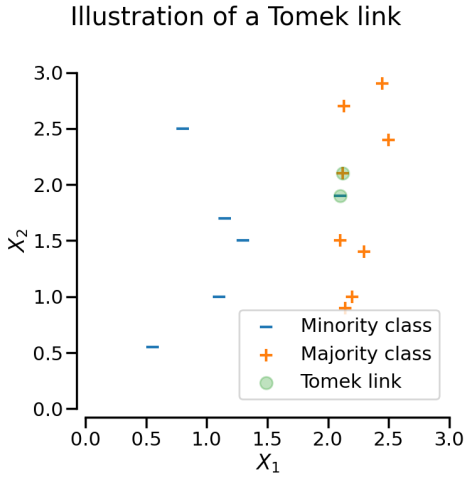


Figura 2.2. Ricerca di Tomek's links

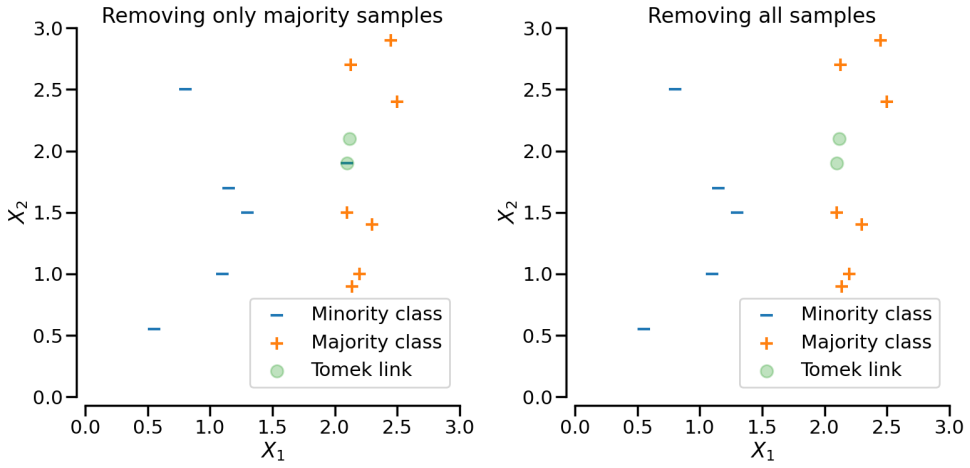


Figura 2.3. Rimozione delle osservazioni associate ai Tomek's links

2.4 Synthetic Minority Oversampling Technique

2.4.1 K-nearest neighbors

Molti algoritmi di classificazione supervisionata si basano sul concetto di nearest neighbors, ossia dati vicini tra loro. Nel dettaglio, dato un insieme di punti ciascuno appartenente alla propria classe, è possibile classificare un nuovo punto calcolando quale sia la classe più rappresentata nel suo vicinato. Quest'ultima viene definita sulla base delle K osservazioni più vicine al nuovo dato, dove il concetto di vicininanza viene inteso basandosi sulla definizione di una metrica. La classe maggiormente rappresentata dai K vicini sarà quella assegnata alla nuova osservazione. In questo modo è possibile definire delle regioni che separano osservazioni appartenenti a classi differenti in modo più o meno accurato a seconda della scelta del valore di K.

2.4.2 Smote

Smote è una tecnica di oversampling basata sulla generazione sintetica di dati. A differenza delle strategie di oversampling citate fino ad adesso, dove si opera nello spazio dei dati duplicando le osservazioni, l'algoritmo sviluppato da Chawla et al. [2002] si concentra sullo spazio delle features. A partire da una osservazione della classe minoritaria si cercano i suoi K-nearest neighbors della stessa classe di appartenenza. Successivamente, si generano nuovi dati lungo il segmento che collega le coppie delle K osservazioni più vicine. La scelta di K è qualcosa che dipende dalle richieste del problema. Solitamente viene anche fissata una quantità N che rappresenta la quantità totale di oversampling che si vuole fare. Nel particolare, la generazione dei dati viene eseguita prendendo la differenza tra due vettori delle features associati a due delle K osservazioni più vicine Fernández et al. [2018]. Questo passaggio viene eseguito su ogni coppia. Si moltiplica quindi la differenza per un numero casuale compreso tra zero e uno e la si somma ad uno dei vettori della coppia in considerazione. Questo porta alla creazione di una osservazione lungo il segmento che unisce i vettori delle features. Infine, di tutte le osservazioni generate ne vengono scelte N in modo casuale. Una diretta conseguenza di questa costruzione è che Smote non fa alcun tipo di distinzione tra le osservazioni difficili o facili da apprendere Lemaître et al. [2017]. Viene infatti utilizzata la tecnica dei K-nearest neighbors per la scelta delle osservazioni senza indirizzare la generazione dei dati verso particolari aree del data set.

É possibile infine riassumere quanto appena detto attraverso lo pseudocodice riportato in 1 e 2. Fernández et al. [2018]

2.4.3 Vantaggi di Smote

La nuova visione del bilanciamento dei dati proposta da Chawla et al. [2002] attraverso Smote ha permesso di superare diversi problemi citati precedentemente. In particolare, facendo riferimento all'esempio del data set sui pixel di una mammografia, le regioni di decisione ottenute tramite Smote sono più larghe e meno specifiche, come mostrato in figura 2.1 (c). Gli algoritmi infatti in fase di apprendimento non focalizzeranno la loro attenzione solo su certe osservazoni determinando quindi regioni più generali per la classe

Algoritmo 1 Smote

funzione Smote(T,N,K)
Input T: osservazioni classe minoritaria;
N: quantità totale di oversampling;
K: parametro per la scelta delle osservazioni più vicine.
Output (N/100)*T osservazioni della classe minoritaria
if N $<$ 100 **then**
considera solo la percentuale che N rappresenta di tutto T
T=(N/100)*T
N=100
end if
N=(int)N/100 La quantità di oversampling da eseguire è un multiplo intero di 100
for $i = 1 : T$ **do**
calcola i K-nearest neighbors di i e salvali in *nnarray*
Populate(N,i,*nnarray*)
end for
end funzione

Algoritmo 2 Funzione per generare osservazioni

funzione Populate(N,i,nnarray)
Input N: numero di osservazioni da creare;
i: indice dell'osservazione originale;
nnarray: array contenente le k osservazioni più vicine di i.
Output N osservazioni nuove contenute nell'array *synthetic*
Variabili *sample*: array contenente le osservazioni originali della classe minoritaria;
newindex: conteggio dati generati (inizializzato a 0);
synthetic: array contenente le osservazioni generate;
numattrs: numero di attributi presenti in ogni osservazione originale.
while $N \neq 0$ **do**
nn=random(1,k)
for $attr = 1 : numattrs$ **do**
dif=*sample*[nnarray[nn]][attr]-*sample*[i][attr]
gap=random(0,1)
synthetic[*newindex*][attr]=*sample*[i][attr] + $gap \cdot dif$
end for
newindex++
N- -
end while
end funzione

minoritaria. Questo permette di fornire predizioni più accurate. Oltre a ciò, per come è stato costruito, Smote permette di risolvere parallelamente il problema dello sbilanciamento insieme agli small disjuncts e alla mancanza di dati già discussi precedentemente.

Creando nuove osservazioni verranno riempite maggiormente le aree del dataset favorendo la presenza di gruppi di dati con caratteristiche analoghe. Ovviamente il presupposto alla base di questo ragionamento è che i K-nearest neighbors siano contenuti all'interno dell'area dove vi è mancanza di dati. Diventa ancora più cruciale la scelta del parametro K in base a come sono caratterizzati gli small disjuncts.

Per quanto riguarda invece i problemi sulla sovrapposizione tra le classi e la diversa distribuzione interna, sono state create delle estensioni di Smote. La ragione principale è dovuta al fatto che Smote non è in grado di riconoscere all'interno dei K-nearest neighbors quali appartengano alla stessa classe dell'osservazione in considerazione. In questi casi diventa necessario combinare Smote con altre tecniche ad esempio di pulizia dei dati.

2.4.4 Estensioni di Smote

Per determinare possibili varianti di Smote è necessario analizzare quali componenti dell'algoritmo possano essere integrate con differenti metodologie. Fernández et al. [2018] hanno elaborato estensioni che prevedono:

1. Differente selezione iniziale delle istanze da campionare. Come già accennato precedentemente potrebbe essere utile selezionare le osservazioni più importanti a partire dalle quali generarne di nuove. Questa scelta deve essere fatta in modo tale da ridurre la sovrapposizione e le osservazioni rumorose, ossia quelle affette da possibili errori.

2. Integrazione con undersampling e/o tecniche di pulizia. Esistono procedure che prevedono la combinazione di oversampling della classe minoritaria parallelamente all'undersampling della classe maggioritaria. In questi casi gli algoritmi forzano ad apprendere maggiormente la classe minoritaria. Inoltre, nel caso particolare di Smote, la generazione di osservazioni a partire dal vicinato di un dato iniziale potrebbe portare alla creazione di istanze rumorose. Infatti, i K-nearest neighbors del dato iniziale non necessariamente appartegono alla sua stessa classe. Interpolando lungo il segmento che collega osservazioni appartenenti a classi differenti si aumenterebbe solo il grado di sovrapposizione tra le classi. Per questo scopo vengono introdotti metodi che prevedono ad esempio l'utilizzo di Smote e Tomek's links insieme.

3. Diverso tipo di interpolazione. La versione base di Smote genera nuovi dati lungo il segmento che collega un'istanza con i suoi K-nearest neighbors. Una possibile variante potrebbe essere quella di tenere conto solamente dei vicini appartenenti alla stessa classe. In maniera diemetralmente opposta potrebbero invece essere solamente tenuti in considerazione i vicini appartenenti alla classe opposta. In questo caso tuttavia la generazione di nuovi dati dovrebbe portare alla creazione di osservazioni più vicine all'istanza di riferimento rispetto al suo vicinato. Infine, un'altra possibile alternativa potrebbe prevedere nessun tipo di interpolazione ma semplicemente una variazione di alcune features in modo tale da generare osservazioni leggermente perturbate rispetto a quella originale. Questo procedimento viene chiamato jittering.

4. Operazioni con la riduzione di dimensionalità. É prevista la combinazione di Smote con tecniche di feature selection o feature extraction, come la Principal Component Analysis. Questo punto verrà ripreso successivamente.

Nel seguito vengono riportati metodi pensati come estensioni di Smote. Alcuni sono stati sviluppati a partire dalle osservazioni appena fatte, altri invece sono legati alla natura dei problemi e dei dati a disposizione.

Smote-NC

Smote-NC, ossia Synthetic minority oversampling technique-nominal continuos, è la versione di Smote per data set che presentano features continue e nominali Chawla et al. [2002]. Il principio dell'algoritmo è lo stesso dell'originale ma con qualche accorgimento. Viene definita una nuova variabile, la mediana delle deviazioni standard di tutte le features continue della classe minoritaria. Questo termine viene utilizzato per tenere conto delle features nominali, penalizzando le osservazioni che ne presentano valori differenti. Infatti, il criterio con cui viene calcolata la distanza tra due osservazioni, prevede di sommare la mediana ogni volta che i valori nominali delle due istanze non coincidono. In questo modo vengono calcolati i K-nearest neighbors. Infine, quando vengono create nuove osservazioni, viene assegnato come valore nominale quello meggiormente presente nei K-nearest neighbors.

Smote-N

Smote-N è l'estensione di Smote per data set con sole features nominali Chawla et al. [2002]. Viene definita una nuova metrica, detta Value difference Metric (VDM), per calcolare la distanza tra i valori di due features x_1, y_1:

$$\delta(x_1, y_1) = \sum_{i=1}^{n} \left| \frac{C_{x_1 i}}{C_{x_1}} - \frac{C_{y_1 i}}{C_{y_1}} \right|^k \tag{2.1}$$

dove $C_{x_1 i}$ rappresenta il numero di occorrenze di x_1 nella classe i e C_{x_1} è il numero totale di occorrenze di x_1. Analgamente per y_1, $C_{y_1 i}$, C_{y_1} mentre k è una costante solitamente posta pari a 1. Va osservato che x_1 e y_1 rappresentano la generica componente di due osservazioni differenti. La distanza effettiva tra due osservazioni è data da:

$$\Delta(X, Y) = w_x w_y \sum_{i=1}^{N} \delta(x_i, y_i)^r \tag{2.2}$$

dove r è un coefficiente che dipende dal tipo di distanza che si vuole calcolare e w_x, w_y sono dei pesi. In particolare, r=2 è la distanza euclidea mentre solitamente w_y=1 e w_x è un peso che da maggiore importanza alle osservazioni più corrette e meno rumorose. Per generare nuove osservazioni della classe minoritaria viene creato infine un nuovo set di features corrispondenti alle features più ricorrenti del vicinato. Si supponga di avere ad esempio il dato iniziale F1=A B C D E e i suoi 2 vicini F2=A F C G N, F3=H B C D N (2-nearest neighbors). Il risultato restituito da Smote-N sarà l'osservazione FS=A B C D N.

Borderline-Smote

Borderline-Smote è una variante di Smote pensata per i casi dove i maggiori errori di classificazione avvengono ai confini tra le regioni di decisione, dette anche decision boundary, Wijaya [2020]. Viene quindi assunto che le osservazioni distanti dal bordo possano contribuire meno alle performance finali di classificazione Fernández et al. [2018]. Diversamente da Smote, che prevede la generazione di una nuova osservazione tra due dati scelti in modo casuale, Borderline-Smote genera nuovi dati solamente lungo i decision boundary. Per identificare le osservazioni sul bordo in una classificazione binaria, viene calcolato il rapporto tra numero di elementi della classe maggioritaria rispetto a quella minoritaria presenti nel vicinato di un osservazione. Nel dettaglio, Lemaître et al. [2017] classifica le istanze in tre modi differenti: rumorose, in pericolo o sicure. Le prime sono caratterizzate per avere tutto il vicinato appartenente alla classe opposta. Le seconde presentano una metà del vicinato omologo e l'altra metà eterologo, mentre le sicure hanno tutti i vicini della loro stessa classe. Vengono quindi utilizzate le istanze in pericolo per generarne di nuove. In questi casi è possibile inoltre specificare quali vicini dei K-nearest neighbors scegliere per la creazione di un nuovo dato. Esistono due versioni a riguardo: la prima prevede di selezionare i vicini appartenti alla stessa classe mentre la seconda permette di selezionare i vicini anche della classe opposta.

Safe-Level-Smote

Come riportato in Fernández et al. [2018] questo algoritmo concentra la generazione di osservazioni all'interno delle regioni di dati apppartenenti alla stessa classe. Questo è permesso grazie alla definizione di un safe level per ogni istanza della classe minoritaria che corrisponde al numero di vicini appartenententenenti alla stessa classe. Viene quindi calcolato il safe level ratio ottenuto come il rapporto fra il safe level di una osservazione minoritaria e il safe level di un suo vicino. L'interpolazione di nuovi dati verrà effettuata tenendo conto di questo rapporto posizionando i nuovi dati vicino a quelli più sicuri. In questo modo si verranno a creare delle safe regions che permetteranno un migliore distinzione tra le classi, con conseguente incremento delle performance.

Metodi multiclasse

Fino a questo momento nella maggior parte dei casi si è parlato sempre di classificazione binaria. É bene tenere presente che le tecniche di ricampionamento per duplicazione sono applicabili identicamente nel caso di dataset che presentino più classi. In questi casi infatti il ricampionamento verrà fatto in maniera indipendente tra le classi. Per quanto riguarda Smote invece, l'utilizzo della tecnica dei K-nearest neighbors in generale non permette di selezionare istanze della classe di appartenenza, come già evidenziato in precedenza. In quesi casi viene utilizzato un approccio one-vs-rest, ossia viene applicata una classifazione binaria considerando come due classi quella minoritaria e quella comprendente tutte le altre. Lemaître et al. [2017]

2.5 Adaptive Synthetic Sampling

Adasyn è un'altra una tecnica di oversampling basata sulla generazione sintetica di dati. Spesso viene consderata un'estensione di Smote per il semplice fatto che utilizza lo stesso principio per la generazione di nuove osservazioni. Tuttavia, il numero di osservazioni minoritarie generate è inversamente proporzionale alla loro densità Wijaya [2020]. Verranno quindi creati più dati in corrispondenza delle aree che ne presentano meno. Bisogna tenere presente comunque che spesso i dati a minore densità potrebbero essere degli outlier, ossia delle osservazioni rumorose, errate. In questi casi Adasyn dedicherebbe troppa attenzione a questi dati portando ad un calo delle prestazioni degli algoritmi in termini predittivi. Di conseguenza, potrebbe essere utile rimuovere gli outlier da un data set prima di applicare Adasyn.

Nel dettaglio, analogamente a Smote, è possibile descrivere Adasyn attraverso lo pseudocodice tratto da He et al. [2008] e riportato in 3.

Algoritmo 3 ADASYN

funzione Adasyn(D_{tr}, m_s, m_l)

Input D_{tr}: training set con m osservazioni $\{x_i, y_i\}$ con $x_i \in X$ e $y_i \in Y = \{-1,1\}$;
m_s: numero osservazioni della classe minoritaria;
m_l: numero osservazioni della classe maggioritaria ($m_s \leq m_l$ e $m_s + m_l = m$);

Output g nuove osservazioni della classe minoritaria ($g = \sum_{i=1}^{m_s} g_i$)

Variabili d_{th}: massimo grado tollerato di sbilanciamento;
β: parametro per specificare il grado di sbilanciamento ($\beta \in [0,1]$);
Δ_i numero di osservazioni tra i K vicini che appartengono alla classe maggioritaria

d=$\frac{m_s}{m_l}$ grado sbilanciamento

if d $< d_{th}$ **then**

G=$(m_l - m_s) \cdot \beta$ numero di osservazioni da generare per la classe minoritaria

for i=1:m_s **do**

calcola i K-nearest neighbors di x_i

r_i=$\frac{\Delta_i}{K}$, i=1,...,m_s proporzione di osservazioni maggioritarie nel vicinato di x_i

$\hat{r}_i$=$\frac{r_i}{\sum_{i=1}^{m_s} r_i}$ proporzione normalizzata

g_i=$\hat{r}_i \cdot G$ numero di osservazioni che devono essere generate per x_i

for j=1:g_i **do**

scegli in modo casuale una osservazione x_{zi} tra i K vicini di x_i

s_i=$x_i + (x_{zi} - x_i) \cdot \lambda$ nuova osservazione generata ($\lambda \in [0,1]$ numero casuale)

end for

end for

end if

end funzione

L'introduzione della distribuzione di densità $\hat{r}$ è ciò che differenzia principalmente Adasyn da Smote. Essa può essere interpretata come una misura del livello di difficoltà nell'apprendere un certo campione all'inteno del data set. Infatti, maggiore sarà $\hat{r}_i$ maggiore sarà la quantità di osservazioni generate nell'intorno di x_i. In questo modo gli algoritmi di classificazione focalizzeranno maggiormente la loro attenzione nelle aree del data set difficili da apprendere. Infine, un altro punto di forza di Adasyn è sicuramente l'aggiornamento degli $\hat{r}_i$ che ad ogni iterazione viene fatto adattandosi alla nuova distribuzione dei dati, contenente quindi anche quelli appena generati.

2.5.1 Possibili estensioni di Adasyn

Analogamente a Smote è possibile determinare varianti di Adasyn. In particolare, la struttura dell'algoritmo si presta bene a gestire sia problemi multiclasse che data set non stazionari He et al. [2008].

Nel caso di problemi multiclasse è sufficiente fare piccoli accorgimenti a partire dall'algoritmo iniziale. Innanzitutto bisognerebbe calcolare i gradi di sbilanciamento per ogni classe rispetto a quella che presenta il maggior numero di istanze. Successivamente, per tutte le classi i che soddisfano la condizione $d_i < d_{th}$ verrebbe eseguita la funzione Adasyn bilanciando ogni classe in accordo con la propria distribuzione di densità dei dati. Nel dettaglio, supponendo $Y = \{1, ..., C\}$ il vettore delle classi possibili e $y_s \in Y$ la classe con il maggior numero di istanze, il calcolo di Δ_i potrebbe essere ridefinito come il numero di osservazioni tra i K-nearest neighbors che appartengono alla classe y_s. Un'altra possibile variante potrebbe essere di procedere con un approccio one-vs-rest simile a quello di Smote. In questo caso Δ_i conterebbe il numero di osservazioni tra i vicini che appartengono a tutte le classi eccetto quella di x_i. A prescindere dalla scelta fatta, il coefficiente r_i verrebbe definito diversamente determinando quindi una diversa distribuzione di densità rispetto al caso binario.

Per quanto riguarda invece la gestione di data set sbilanciati non stazionari, ossia dove i dati di training sono in continuo aumento, l'adattabilità di Adasyn gioca un ruolo fondamentale. Infatti, a livello intuitivo, gli algoritmi che si approcciano a questo tipo di situazioni devono essere in grado di fare esperienza su ciò che hanno a disposizione per poter dedurre nuove informazioni e aiutare la predizione futura. In questo senso l'elasticità del ricampionamento tramite Adasyn potrebbe aiutare gli algoritmi durante l'apprendimento. Potenzialmente infatti gli r_i potrebbero essere aggiornati ogni volta che si riceve una nuova porzione di dati.

Capitolo 3

Applicazione pratica

3.1 Financial ratios per la previsione di bancarotta

In questa sezione vengono applicati i metodi precedentemente descritti ad un caso pratico. In particolare, vengono fatte analisi dati in seguito al bilanciamento del dataset in questione. L'obiettivo è quello di analizzare i diversi risultati ottenuti al variare della strategia di ricampionamento utilizzata e confrontarli. Il tema di questo data set sono i financial fatios per la previsione dello stato di bancarotta. Questo è un tema molto importante che viene trattato con particolare attenzione all'interno degli isituti finanziari. Il fine principale è quello di prendere decisioni appropriate per finanziare prestiti che non abbiano un impatto negativo sugli istituti stessi. I financial ratios in letteratura sono conosciuti come fattori importanti che influenzano la bancarotta. Essi possono essere classificati in 7 categorie: solvency, profitability, cash flow ratios, capital structure ratios, turnover ratios, growth e others. Oltre a quanto già detto un altro fine di questo lavoro è quello di testare algoritmi per capire quali features hanno un maggior peso rispetto alle altre nella predizione della bancarotta.

3.2 Descrizione del dataset nel dettaglio

Questo dataset presenta dati che sono stati raccolti dal Taiwain Economic Journal tra il 1999 e il 2009. Lo stato di bancarotta viene definito in base alla regolamentazione del mercato azionario taiwanese.
Il dataset presenta 6819 righe e 96 colonne così suddivise:
- stato attuale: colonna 1, 0=no bancarotta, 1=si bancarotta
- solvency: colonne 14, 32-37, 40-43, 54, 67, 84-85, 95
- capital structure ratios: colonne 38-39, 76-79, 91-93
- profitability: colonne 1-10, 19, 23, 68-70, 6, 89-90, 94
- turnover ratios: colonne 44-50, 71-75, 88
- cahs flow ratios: colonne 13, 80-83
- growth: colonne 24-31
- others: colonne 11-12, 15-18, 20-22, 51-53, 87

```
0    6599
1     220
Name: Bankrupt?, dtype: int64
```

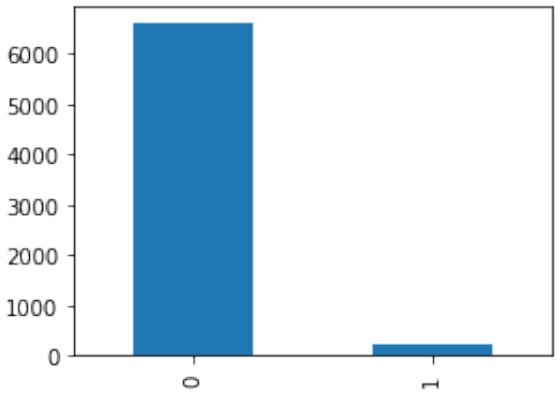


Figura 3.1. Distribuzione delle classi

Il barplot mostra che le classi sono altamente sbilanciate e questo potrebbe generare diversi problemi nell'applicazione di metodi predittivi.

3.3 Random oversampling & random undersampling

Suddivisione training (50%) e test (50%) set per PCA e LDA

[4]:
```
X=frs.iloc[:,1:96]
y=frs.iloc[:,0]
random_state = 20210528
test_p = 0.5

indices = np.arange(X.shape[0])
Xtrain, Xtest, ytrain, ytest  = train_test_split(X, y, test_size=test_p,␣
 ↪random_state=random_state, shuffle=True,stratify=y)
```

Il pacchetto train_test_split permette di dividere il dataset iniziale in 2 sottoinsiemi da poter utilizzare separatamente. L'opzione stratify=y permette di dividere i dati in training e test mantenendo al loro interno le proporzioni esatte rispetto al bilanciamento iniziale delle classi. Senza questa opzione si potrebbero generare dei training set pari al 50% dei dati totali ma con al loro interno solo 0 o solo 1. All'interno del training set sbilanciato si va ad effettuare il pareggio delle classi utilizzando random oversampling, random undersampling, Smote, Adasyn e Tomek's links. Per questa prima parte verranno utilizzate solamente le prime due tecniche tra quelle appena citate.

	train pca-lda	train over pca-lda	train under pca-lda	test pca-lda
0	3299	3299	110	3300
1	110	3299	110	110

Nella seguente tabella sono presenti le dimensioni dei train e test set che verranno utilizzati per applicare la PCA e la LDA al variare della strategia di ricampionamento scelta.

3.4 Principal Component Analysis (PCA)

La PCA è un metodo che viene definito di apprendimento non supervisionato. Non vi è infatti presente alcuna funzione che associa alle osservazioni iniziali un target ma l'obiettivo principale è quello di capire come sono distribuiti i punti nello spazio. La PCA si occupa quindi di dare una rappresentazione dei dati in dimensione bassa, ad esempio in $\mathbb{R}^2$ o $\mathbb{R}^3$ attraverso degli score graph, per avere un'idea di come possano essere distribuite le osservazioni nello spazio.
A partire dal nostro data set si proiettano i dati lungo le direzioni che massimizzano la varianza e preservano la distibuzione delle distanze. Minimizzare l'errore legato alla proiezione dei dati coincide con minimizzare il seguente funzionale:

$$J(e_1, ..., e_k, \alpha_{11}, ..., \alpha_{nk}) = \sum_{j=1}^{n} \|x_j - \sum_{i=1}^{k} \alpha_{ji} e_i\|^2$$

dove $\{e_k\}_{k\geq 1}$ sono gli elementi della base ortonormale del sottospazio in cui voglio proiettare e gli $\{\alpha_{ij}\}_{i,j\geq 1}$ sono le componenti rispetto alla base. Attraverso il calcolo delle derivate ci si riconduce a risolvere il problema agli autovalori:

$$Se_m = \lambda_m e_m$$

Dove S=$\sum_{i=1}^{k} x_j^\top x_j$ è la Scatter Matrix ossia la matrice di varianza-covarianza campionaria moltiplicata per n-1 dove n è il numero totale di osservazioni. La soluzione del problema di minimo che massimizza la varianza è data dai primi k più grandi autovalori di S. Vengono quindi definite le PC come le direzioni ortogonali lungo le quali proiettare, ossia i primi k autovettori della matrice S.

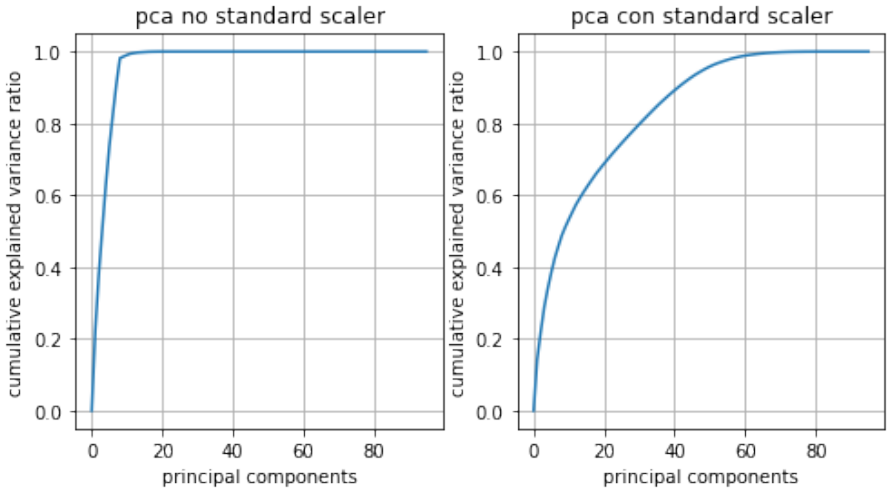


Figura 3.2. Proporzione varianza spiegata dalle componenti principali nella PCA con e senza standard scaler

	n° PC	var spiegata
explvar_no_scaled=0.9	7	1.00000
explvar_scaled=0.9	41	0.90043

I grafici e la tabella mostrano che la PCA senza dati standardizzati è poco affidabile. Per capire questo è necessario osservare le statistiche base dei dati. Esse evidenziano che esistono features con range di intervalli diversi e sproporzionati. Non essendo inoltre note le unità di misura dei dati non è possibile capire quanto gli intervalli di valori siano dello stesso ordine di grandezza. Questo giustifica che le colonne con maggior variabilità vadano ad inglobare anche quelle con variabilità più bassa ottenendo che con sole 7 features su 95 si spieghi più del 90% della varianza. Considereremo quindi dati standardizzati.

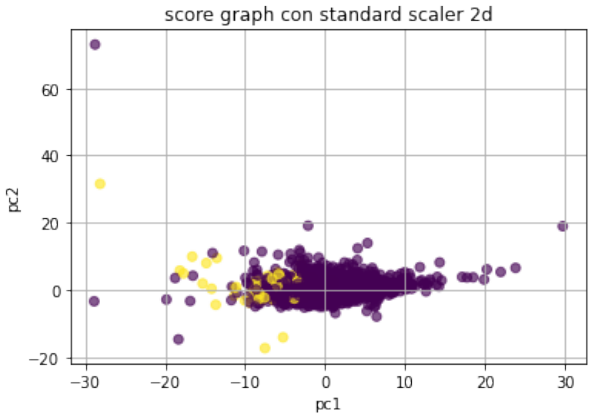


Figura 3.3. Score graph 2d PCA

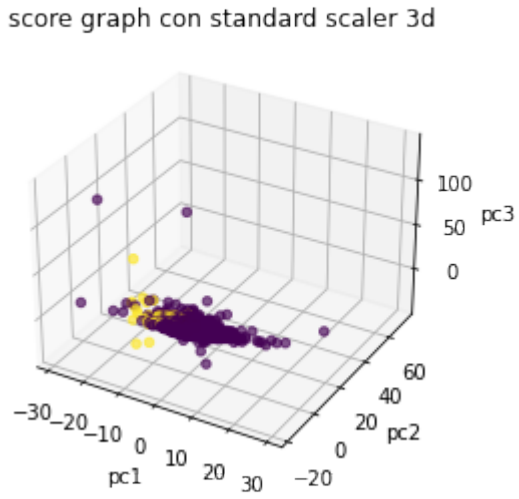


Figura 3.4. Score graph 3d PCA

Il forte sbilanciamento delle classi non permette una visualizzazione ottimale negli score graph. Per questo motivo scegliamo di pareggiare le classi con tecniche di ricampionamento.

3.5 PCA random oversampling

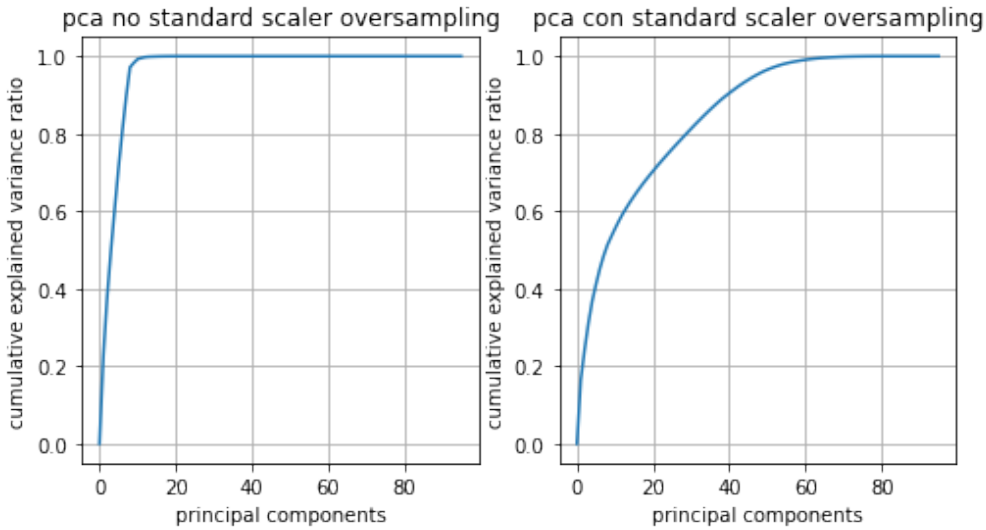


Figura 3.5. Proporzione varianza spiegata dalle componenti principali nella PCA con e senza standard scaler nel caso random oversampling

[13]:

	n° PC	var spiegata
explvar_scaled_over=0.9	40	0.904692
explvar_no_scaled_over=0.9	8	0.971304

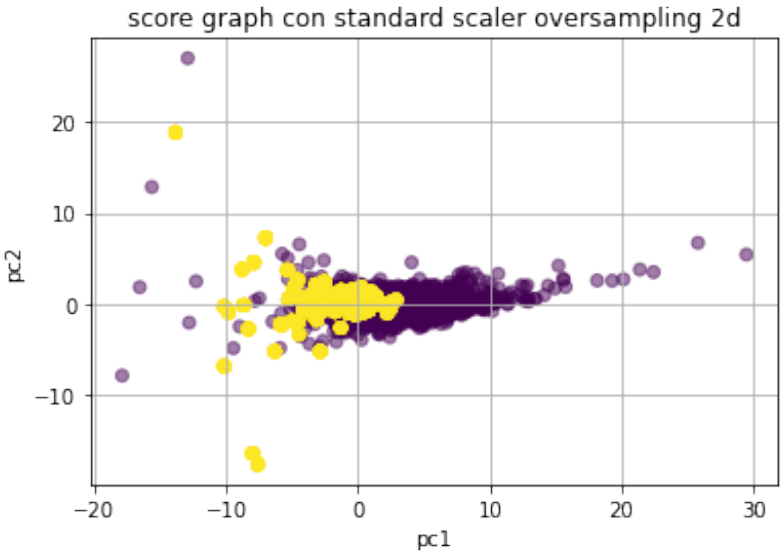


Figura 3.6. Score graph 2d PCA random oversampling

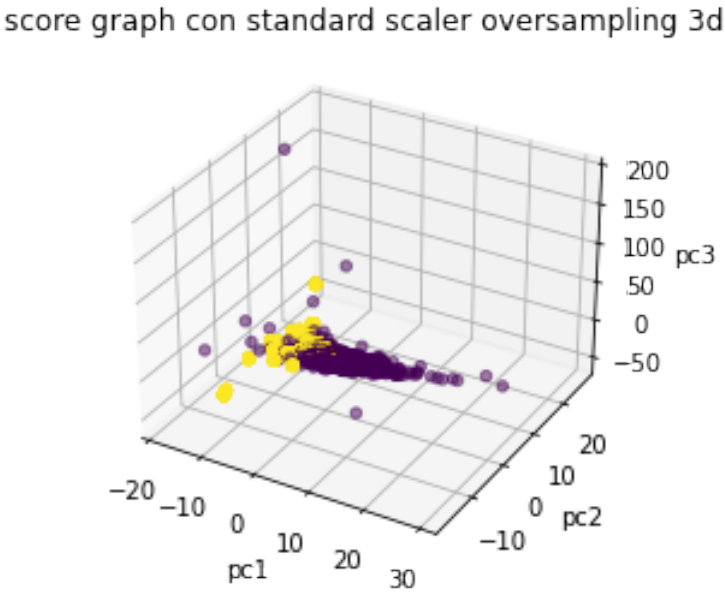


Figura 3.7. Score graph 3d PCA random oversampling

3.6 PCA random undersampling

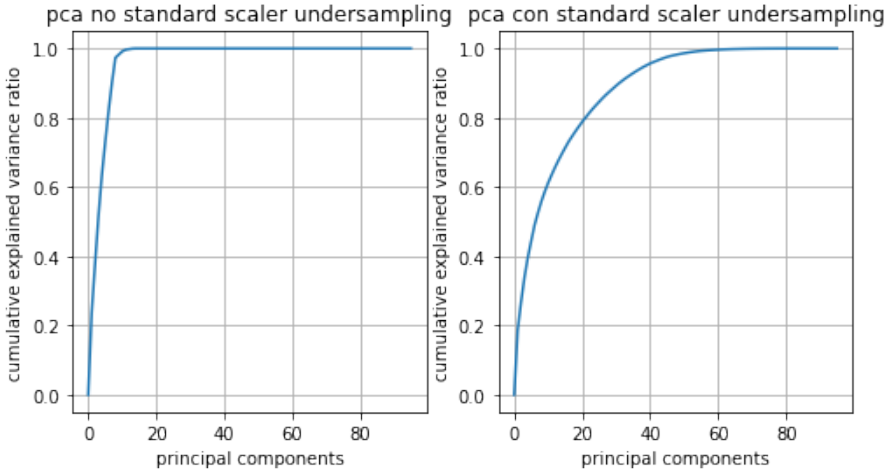


Figura 3.8. Proporzione varianza spiegata dalle componenti principali nella PCA con e senza standard scaler nel caso random undersampling

[16]:

	n° PC	var spiegata
explvar_scaled_under=0.9	31	0.900813
explvar_no_scaled_under=0.9	7	0.902652

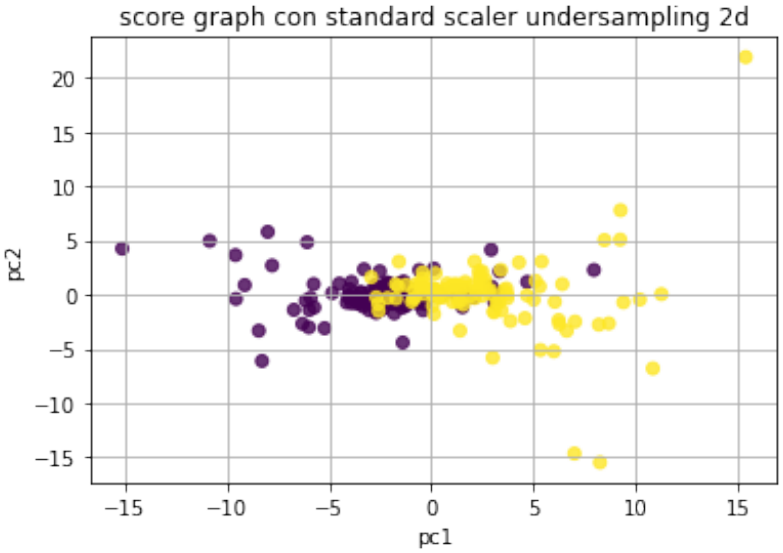


Figura 3.9. Score graph 2d PCA random undersampling

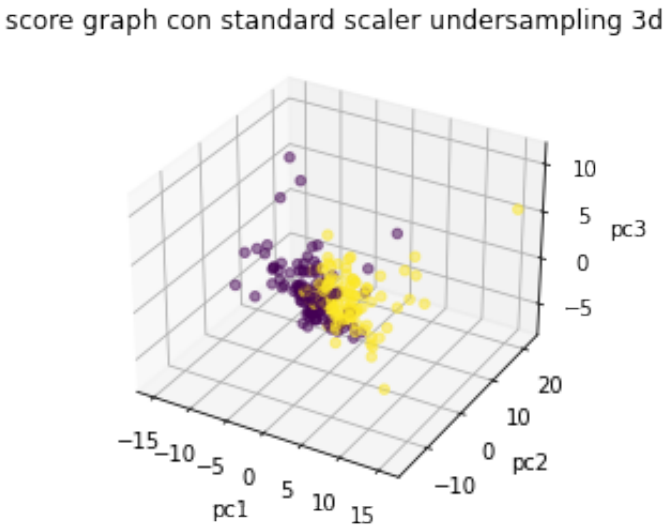


Figura 3.10. Score graph 3d PCA random undersampling

Sia col random oversampling che col random undersampling si ottengono migliori visualizzazioni. Sebbene la distribuzione del set di dati sia stata alterata, la PCA evidenzia che nel caso del random undersampling le proiezioni lungo le prime tre PC permettono una migliore separazione dei dati rispetto al random oversampling come si vede confrontando le figure 3.10 e 3.7.

3.7 Linear Discriminant Analysis (LDA) random oversampling

La LDA è un metodo di apprendimento supervisionato con l'obiettivo di stimare la probabilità di appartenenza ad una certa classe. Il principio su cui si basa la LDA è il teorema di Bayes che può essere riscritto nel seguente modo:

$$P(Y = k|X = x) = \frac{P(X = x|Y = k)P(Y = K)}{P(X = x)} = \frac{\Pi_k f_k(x)}{\sum_{l=1}^{k} \Pi_l f_l(x)}$$

dove Π_k = densità marginale della classe k, $f_k(x)$ =densità di x nella classe k. La classificazione dei punti viene effettuata in base alla densità della classe k più alta che i punti stessi sottendono. Sebbene la LDA possa essere fatta a prescindere dalla distribuzione che seguono i punti, sotto le ipotesi che i dati abbiano distribuzione normale omoschedastica ($\sigma_k = \sigma, \forall k \in Y$) si ha che $P(Y = k|X = x)$ può essere approssimato con $\delta_k(x)$ detto discriminant score. Vale infatti la relazione:

$$P(Y = k|X = x) = \frac{e^{\delta_k(x)}}{\sum_{l=1}^{k} e^{\delta_l(x)}}$$

$$\delta_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\Pi_k) = a_k x + b_k$$

dove μ_k è la media riferita alla distribuzione normale della classe k. Infine, è possibile visualizzare i risultati anche a livello grafico. La classificazione effettuata dalla LDA infatti è equivalente ad una classificazione fatta proiettando i dati sullo spazio generato dai vettori medi delle classi (di dimensione pari al numero di classi meno uno) e identificando il centroide più vicino.

[18]:
```
# test normalità dati
k2,p=stats.normaltest(X)
#p
```

Quanto appena fatto è il test sulla normalità dei dati. Il pacchetto stats di scipy restituisce due valori associati al data set in questione. Il primo è k2 e contiene statistiche eseguite sulle osservazioni a disposizione. Il secondo più importante è un array contenente tanti pvalue quanti sono gli attributi del data set. Per ogni attributo viene eseguito un test d'ipotesi dove l'ipotesi nulla è che i dati siano distribuiti normalmente. Osservando p otteniamo tutti valori minori di α=0.05 e di conseguenza rifiutiamo l'ipotesi nulla. Segue che i dati non sono distribuiti normalmente e non possono essere fatte le approssimazioni precedentemente descritte. Decidiamo comunque di applicare la LDA come algoritmo per predirre lo stato di bancarotta.

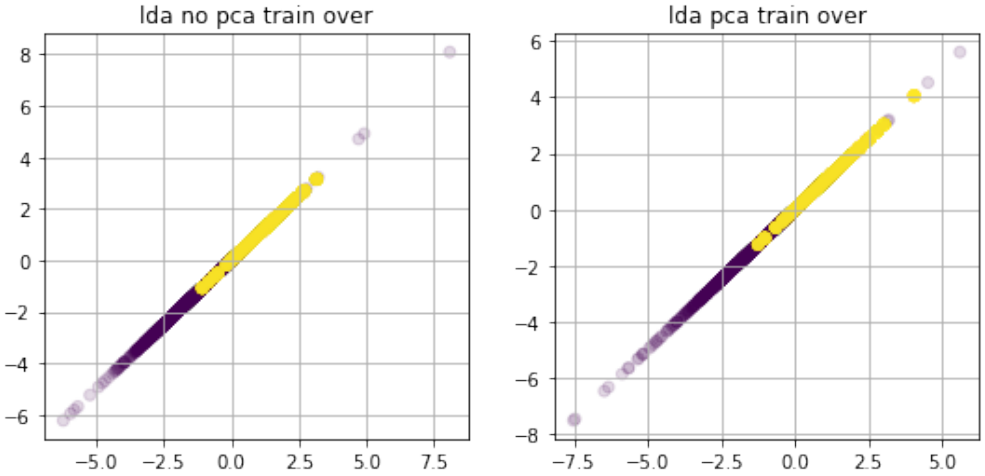


Figura 3.11. Rappresentazione grafica predizioni della LDA sul training set nel caso random oversampling

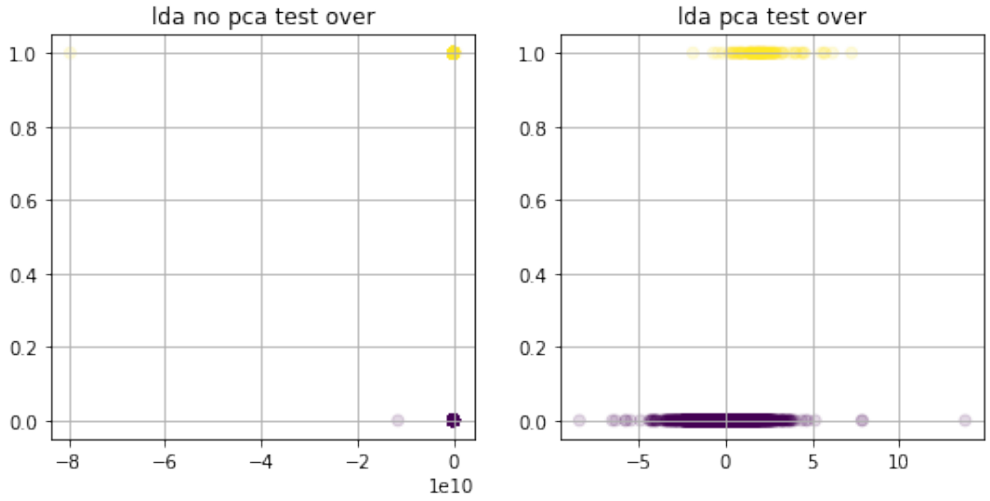


Figura 3.12. Rappresentazione grafica predizioni della LDA sul test set nel caso random oversampling

	actual class	pred class no pca over	pred class pca over
0	3300	2909	1768
1	110	501	1642

Le figure 3.11 e 3.12 rappresentano graficamente i risultati ottenuti. Avendo a disposizione due classi è possibile proiettare i dati solamente lungo una retta. Di conseguenza, non è possibile apprezzare pienamente i risultati sul test set ma è necessario ricorrere a valori numerici. Dalla tabella osserviamo che 391 osservazioni sono state predette non correttamente senza PCA, mentre con la PCA il numero sale a 1532. Le predizioni sbagliate sono riferite a casi di non bancarotta che in realtà lo sono. Tuttavia, da questa tabella si può osservare solo il numero finale di casi sbagliati quindi non si può affermare a priori che tra i 501 1 non ci sia qualche 0 che è stato predetto come 1. Per capire questo si utilizza la confusion matrix e l'F1 score. Ciò che si può affermare con certezza fino a questo momento è che la LDA nel caso del random oversampling tende a predirre casi di non bancarotta come bancarotta visto che predice più 1 del dovuto.

`Performance LDA senza PCA random oversampling`

	acc	prec	rec	f1
train lda no pca over	0.898909	0.899909	0.898909	0.898846
test lda no pca over	0.870088	0.964501	0.870088	0.907565

Confusion matrix LDA senza PCA random oversampling

	no bancarotta	bancarotta
no bancarotta	2883	417
bancarotta	26	84

Performance LDA con PCA random oversampling

	acc	prec	rec	f1
train lda pca over	0.873143	0.873733	0.873143	0.873093
test lda pca over	0.548387	0.967635	0.548387	0.677580

Confusion matrix LDA con PCA random oversampling

	no bancarotta	bancarotta
no bancarotta	1764	1536
bancarotta	4	106

Dalle precedenti tabelle è possibile osservare che la LDA senza PCA lavora meglio rispetto alla LDA con PCA nel caso del random oversampling. Infatti il misclassification rate della LDA random oversampling senza PCA è pari al 13% mentre con la PCA sale al 45%. Questo indicatore viene calcolato a partire dalla confusion matrix come la percentuale di classificazioni errate, a prescindere dalla classe di appartenenza, rispetto a tutto il data set. Invece, le altre statistiche presenti nelle tabelle vengono calcolate nel seguente modo:

1. Accuratezza, il numero totale di predizioni corrette:
$$\text{acc}(\mathcal{P}) = \frac{\text{numero predizioni corrette su } \mathcal{P}}{|\mathcal{P}|}, \tag{3.1}$$
2. Matrice di confusione:
$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix} \tag{3.2}$$
dove TP=veri positivi, FN=falsi neagativi, FP=falsi positivi, TN=veri negativi.
3. Precision, la precisione del modello nel classificare i campioni che appartengono alla classe C_i:
$$\begin{aligned} \text{prec}(C_i; \mathcal{P}) &= \frac{\text{veri } C_i}{\text{veri } C_i + \text{falsi } C_i} = \frac{\text{veri } C_i}{\text{numero di elem. in } \mathcal{P} \text{ pred. } C_i} \\ &= \frac{a_{ii}}{\sum_{k=1}^{c} a_{ki}} = \frac{a_{ii}}{\text{somma elem. in col. } i\text{-esima}} \end{aligned}$$

4. Recall, abilità dell'algoritmo nel trovare tutti i campioni della classe C_i:

$$\text{rec}(C_i;\mathcal{P}) = \frac{\text{veri } C_i}{\text{numero di } C_i \text{ in } \mathcal{P}} = \frac{a_{ii}}{\sum_{k=1}^{c} a_{ik}} = \frac{a_{ii}}{\text{somma elem. in riga } i\text{-esima}} \tag{3.3}$$

5. F1 score, media armonica di precision e recall:

$$F_1(C_i;\mathcal{P}) = 2\,\frac{\text{prec}(C_i;\mathcal{P})\cdot\text{rec}(C_i;\mathcal{P})}{\text{prec}(C_i;\mathcal{P})+\text{rec}(C_i;\mathcal{P})} \tag{3.4}$$

3.8 LDA random undersampling

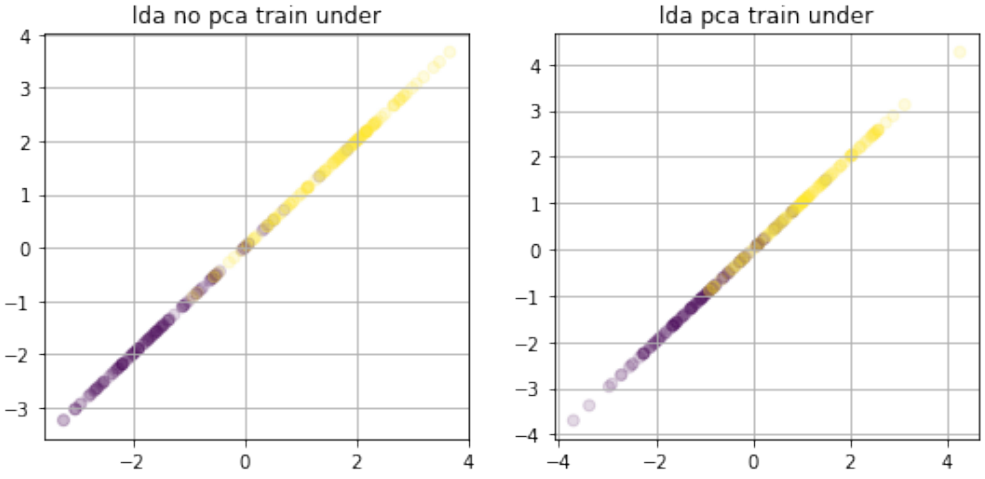


Figura 3.13. Rappresentazione grafica predizioni della LDA sul training set nel caso random undersampling

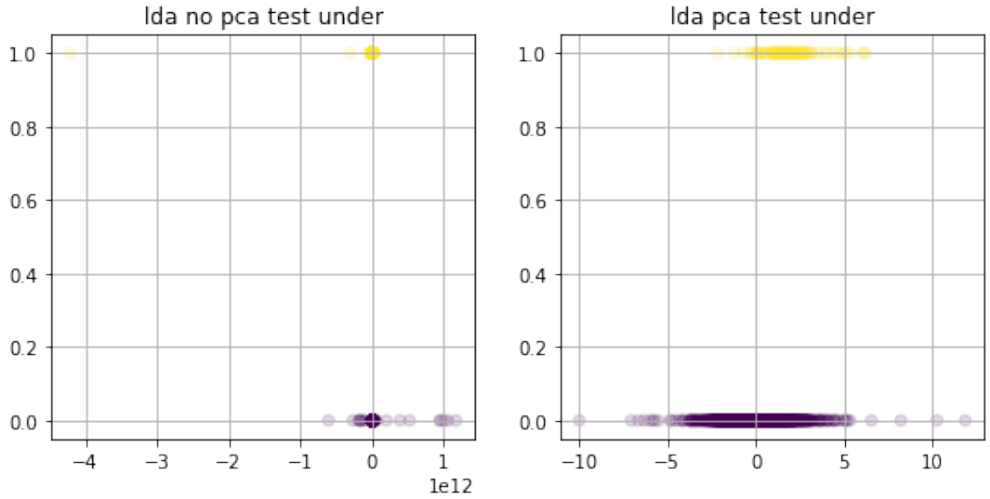


Figura 3.14. Rappresentazione grafica predizioni della LDA sul test set nel caso random undersampling

Performance LDA senza PCA random undersampling

	actual class	pred class lda no pca under	pred class lda pca under
0	3300	2671	1778
1	110	739	1632

Performance LDA senza PCA random undersampling

	acc	prec	rec	f1
training lda no pca under	0.918182	0.918320	0.918182	0.918175
test lda no pca under	0.797361	0.959959	0.797361	0.861752

Confusion matrix LDA senza PCA random undersampling

	no bancarotta	bancarotta
no bancarotta	2640	660
bancarotta	31	79

Performance LDA con PCA random undersampling

```
                            acc      prec       rec        f1
training lda pca under  0.840909  0.840937  0.840909  0.840906
test lda pca under      0.549560  0.965968  0.549560  0.678833


Confusion matrix LDA con PCA random undersampling


               no bancarotta  bancarotta
no bancarotta           1771        1529
bancarotta                 7         103
```

Analogamente al caso precedente i risultati non sono apprezzabili dal punto di vista grafico. Tuttavia nel grafico del training set la separazione tra le classi è molto più evidente rispetto al grafico del test set. Questo si può riscontrare anche a livello numerico. Infatti, i dati riguardanti l'accuratezza, la precision, la recall e l'F1 score sono molto buoni nel training set mentre peggiorano per il test set. Ricalcolando il misclassification rate rispetto alla LDA random undersampling senza e con PCA otteniamo rispettivamente 27% e 48%. In generale si può quindi concludere che la LDA con PCA e il random undersampling lavorano peggio rispetto alla LDA senza PCA e il random oversampling.

3.9 Support Vector Machine (SVM) non lineari random oversampling

A differenza della PCA che mantiene la struttura metrico-statistica dei dati, nelle SVM si cerca l'iperpiano che massimizza la separazione tra punti. Questi ultimi quindi sono fissati, immobili, non vengono proiettati lungo particolari direzioni ma si cerca un separatore lineare che distacchi le classi nel miglior modo possibile. Sotto le ipotesi di lineare separabilità (ovvero esistono $w \in R^n$, $b \in R$ tali che $y_i(< w, x_i > +b) \geq 0 \ \forall i = 1,2..., m$, dove $\{(x_i, y_i)|x_i \in R^n, y_i \in R, i = 1,2..., m\}$ è il data set di riferimento), l'obiettivo delle SVM è massimizzare il margine $margin = \{min\ d(\Pi_{w,b}, x)|x \in dataset\}$. $\Pi_{w,b}$ rappresenta l'iperpiano separatore identificato dalla coppia (w,b) mentre d è la distanza euclidea tra l'iperpiano e i punti. In generale possiamo distinguere due tipi di SVM lineari: le prime sono le SVM hard a cui corrisponde il problema di minimo:

$$\begin{cases} \min_{\boldsymbol{w}} \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{w} \\ y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1\,, \quad \forall\ i = 1, \ldots, T \end{cases} \tag{3.5}$$

Il secondo tipo sono le soft SVM che vengono utilizzate quando i dati non sono perfettamente separabili. In questo caso vengono introdotte delle slack variable $\xi_i = \delta_i ||w||$ e dei parametri di costo $C \in \mathbb{R}^+$ che permettono di poter violare il margine di una certa quantità δ_i. Parametri di costo elevati implicano piccole violazioni mentre parametri di costo bassi permettono rilassamenti maggiori visto che il problema associato riguarda sempre la minimizzazione di un funzionale. Il problema di minimo infatti viene riformulato nel seguente modo:

$$\begin{cases} \min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_{i=1}^{T}\xi_i \\ y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i\,, & \forall\ i = 1,\dots,T\ , \\ \xi_i \geq 0\,, & \forall\ i = 1,\dots,T \end{cases} \tag{3.6}$$

Nel caso di dati non linearmente separabili si utilizzano le SVM non lineari. I dati vengono proiettati in dimensione più alta grazie ad una mappa $\phi : \mathbb{R}^n \to \mathbb{R}^m$ (con $m > n$) così da poterli separare con una SVM lineare. Le criticità legate a questo tipo di SVM sono il costo computazionale e il problema della dimensionalità. Per definire il problema di minimo delle SVM non lineari partiamo dal problema duale associato ad una SVM lineare:

$$\begin{cases} \min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^\top Q\boldsymbol{\alpha} - \sum_{i=1}^{T}\alpha_i \\ \sum_{i=1}^{T}\alpha_i y_i = 0 \\ \alpha_i \geq 0\,, \quad \forall\ i = 1,\dots,T \end{cases}\ , \tag{3.7}$$

dove $\alpha \in \mathbb{R}^T$ e la matrice $Q \in \mathbb{R}^{T\times T}$ è tale che

$$Q = (q_{i,j})_{i,j=1,\dots,T} = \left(y_i y_j x_i^\top x_j\right)_{i,j=1,\dots,T}\ . \tag{3.8}$$

Applicando la trasformazione ϕ ricaviamo il problema duale associato alla SVM non lineare:

$$\begin{cases} \min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^\top Q'\boldsymbol{\alpha} - \sum_{i=1}^{T}\alpha_i \\ \sum_{i=1}^{T}\alpha_i y_i = 0 \\ \alpha_i \geq 0\,, \quad \forall\ i = 1,\dots,T \end{cases}\ , \tag{3.9}$$

dove $\alpha \in \mathbb{R}^T$ e la matrice $Q' \in \mathbb{R}^{T\times T}$ è tale che

$$Q' = (q_{i,j})_{i,j=1,\dots,T} = \left(y_i y_j \phi(x)_i^\top \phi(x)_j\right)_{i,j=1,\dots,T}\ . \tag{3.10}$$

Le difficoltà di quest'ultimo problema di minimo stanno nel calcolo del prodotto scalare $\langle \phi_i, \phi\rangle = \phi(x)_i^\top \phi(x)$. Vengono definiti quindi i Kernel k che sono funzioni $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ per cui esiste un'unica mappa $\phi : \mathcal{X} \to \mathcal{H} \subseteq \mathbb{R}^m$, con $m > n$ ed $\mathcal{H}$ spazio di Hilbert, tale che $\langle \phi_i, \phi_j\rangle_{\mathcal{H}} = k(x_i, x_j)$. Il teorema di Moore assicura l'esistenza e l'unicità di questi Kernel. Nel seguito vengono elencati i principali Kernel utilizzzati nelle applicazioni :
- Kernel Lineare: corrisponde al caso in cui ϕ sia l'identità (cioè è il kernel delle SVM lineari).

$$\langle \phi_i, \phi_j\rangle_{\mathcal{H}} = \langle x_i, x_j\rangle_{\mathcal{X}}\,, \tag{3.11}$$

- Kernel Polinomiale: caratterizzato da un'espressione di tipo polinomiale.

$$\langle \phi_i, \phi_j\rangle_{\mathcal{H}} = \left(\gamma\,\langle x_i, x_j\rangle_{\mathcal{X}} + c_0\right)^d\ ; \tag{3.12}$$

- Radial Basis Function (RBF) Kernel: caratterizzato da un'espressione di tipo esponenziale.

$$\langle \phi_i, \phi_j \rangle_{\mathcal{H}} = e^{-\gamma \langle x_i, x_j \rangle_{\mathcal{X}}^2} ; \tag{3.13}$$

- Kernel Sigmoidale: caratterizzato da un'espressione di tipo sigmoidale

$$\langle \phi_i, \phi_j \rangle_{\mathcal{H}} = \tanh\left(\gamma \langle x_i, x_j \rangle_{\mathcal{X}} + c_0\right) ; \tag{3.14}$$

I parametri γ e c_0 dei Kernel influenzano l'addestramento delle SVM. Ad esempio, più grande è γ, maggiore è l'influenza che hanno i singoli elementi del training set per l'addestramento. Per quanto riguarda c_0 invece, il suo comportamento è molto simile a quello γ.

Prima di applicare le SVM scegliamo di ridividere il data set secondo le seguenti proporzioni: training set 30%, validation set 20% e test set 50%. Le percentuali sono da intendere come frazione dell'intero data set. Nella ricerca a griglia degli iper-parametri ottimali per un generico problema risulta utile dividere il dataset non in due, ma in tre sottoinsiemi: training set, validation set e test set. Il validation set serve da pre-test set. Viene utilizzato ad esempio per la ricerca dei migliori iper-parametri, generalmente misurando le performance di modelli addestrati sul training set per avere una sottostima delle possibili performance finali sul test set. Normalmente il validation set è il sottoinsieme di dimensione più piccola, per questo motivo le performance sono una sottostima di quelle che si dovrebbero avere nel test set.

```
   train svm  test svm
0       1989      3294
1         56       116
```

```
   train over svm  train under svm  val over svm  val under svm  test svm
0            1989               56          1316             48      3294
1            1989               56          1316             48       116
```

Per cercare la miglior combinazione di iperparametri per un modello, si effettua una ricerca a griglia. Dato un intervallo discreto di valori I_h per ogni iperparametro p_h, $h = 1, \ldots, H$, si considera la griglia di punti generata dal prodotto cartesiano degli intervalli, cioè $G = I_1 \times \cdots \times I_H$. Ogni punto di G rappresenta una possibile combinazione degli iper-parametri del modello. In totale si considerano quindi i $K = |G|$ modelli caratterizzati dalle K combinazioni di iper-parametri differenti e si cerca quello con le migliori performance sul validation set.

[27]:
```
C_list = [i for i in range(1,6)]
kernel_list =['rbf','poly','sigmoid']
gamma_list = ['scale','auto']
hparameters = {'C': C_list, 'kernel': kernel_list}
svm = SVC(class_weight='balanced')
svm_gs = GridSearchCV(estimator=svm,
                      param_grid=hparameters,
                      scoring='f1_weighted',
                      return_train_score=True,
```

```
                       cv=zip([ind_train_svm], [ind_val_svm]))
svm_gs.fit(X, y)
```

[27]:
```
GridSearchCV(cv=<zip object at 0x000001D86DA47900>,
             estimator=SVC(class_weight='balanced'),
             param_grid={'C': [1, 2, 3, 4, 5],
                         'kernel': ['rbf', 'poly', 'sigmoid']},
             return_train_score=True, scoring='f1_weighted')
```

D'ora in avanti tutte le analisi riportate faranno riferimento alla miglior SVM trovata dalla GridSearchCV.

```
Performance SVM senza PCA random oversampling
```

	acc	prec	rec	f1
train svm no pca over	0.908245	0.910938	0.908245	0.908095
val svm no pca over	0.516717	0.530018	0.516717	0.456513
test svm no pca over	0.824340	0.942613	0.824340	0.875767

```
Confusion matrix SVM senza PCA random oversampling
```

	no bancarotta	bancarotta
no bancarotta	2771	523
bancarotta	76	40

```
Performance SVM con PCA radom oversampling
```

	acc	prec	rec	f1
train svm over pca	0.976370	0.977436	0.976370	0.976357
val svm over pca	0.716185	0.769425	0.716185	0.701436
test svm over pca	0.754252	0.954507	0.754252	0.832376

```
Confusion matrix SVM con PCA random oversampling
```

	no bancarotta	bancarotta
no bancarotta	2494	800
bancarotta	38	78

I risultati ottenuti evidenziano che non esite una SVM (con o senza pca) nettamente migliore dell'altra. La scelta del modello migliore si baserà anche su quanta accuratezza si vuole avere nel predirre la classe bancarotta come tale. I misclassification rate dell'SVM random oversampling senza e con PCA sono rispettivamente 17% e 24%.

3.10 SVM random undersampling

```
Performance SVM senza PCA random undersampling
```

	acc	prec	rec	f1
train svm no pca under	0.785714	0.789032	0.785714	0.785098
val svm no pca under	0.562500	0.565336	0.562500	0.557701
test svm no pca under	0.537830	0.935150	0.537830	0.671671

```
Confusion matrix SVM senza PCA random undersampling
```

	no bancarotta	bancarotta
no bancarotta	1779	1515
bancarotta	61	55

```
Performance SVM con PCA random undersampling
```

	acc	prec	rec	f1
train svm under pca	0.991071	0.991228	0.991071	0.991071
val svm under pca	0.947917	0.952830	0.947917	0.947775
test svm under pca	0.416129	0.966348	0.416129	0.551466

```
Confusion matrix SVM con PCA random undersampling
```

	no bancarotta	bancarotta
no bancarotta	1305	1989
bancarotta	2	114

Le SVM nel caso del random undersampling non presentano ottimi risultati. Questo era intuibile da come sono stati bilanciati i dati inizialmente. Infatti, i dati di training nel caso del random undersampling sono molto piccoli (#{1}=56, #{0}=56) e questo non permette alle SVM di addestrarsi correttamente nel riconoscimento delle classi. Non a caso i misclassification rate delle SVM random undersampling con e senza PCA risultano essere rispettivamente 58% e 46%, ossia all'incirca metà delle predizioni sono errate.

3.11 Multi-Layer Perceptron (MLP) random oversampling

Gli MLP sono grafi multipartiti diretti. La loro struttura è composta da una serie di strati completamente connessi detti FC layers.

Un FC layer è un'applicazione $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}^m$ tale che

$$\mathcal{L}(x) := \sigma\left(Wx + b\right) , \tag{3.15}$$

dove: $W \in \mathbb{R}^{m \times n}$ è la matrice dei pesi, $b \in \mathbb{R}^m$ è il vettore dei bias e $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ è una funzione vettoriale che applica elemento per elemento la funzione di attivazione $\sigma : \mathbb{R} \to \mathbb{R}$ (σ non necessariamente lineare). Un generico MLP quindi può essere rappresentato dalla funzione $F : \mathbb{R}^n \to \mathbb{R}^m$ dove:

$$F(x) : \mathbb{R}^n \xrightarrow{\mathcal{L}^{(1)}} \mathbb{R}^{n_1} \xrightarrow{\mathcal{L}^{(2)}} \cdots \xrightarrow{\mathcal{L}^{(H)}} \mathbb{R}^{n_H} \xrightarrow{\mathcal{L}^{(H+1)}} \mathbb{R}^m \tag{3.16}$$

$$F(x) = \sigma^{(H+1)}\left(W^{(H+1)}\sigma^{(H)}\left(\cdots\left(W^{(2)}\sigma^{(1)}\left(W^{(1)}x + b^{(1)}\right) + b^2\right)\cdots\right) + b^{(H+1)}\right) , \quad \forall\, x \in \mathbb{R}^n . \tag{3.17}$$

$F(x)$ è una funzione parametrica con parametri $W^{(h)}$ e $b^{(h)}$. L'addestramento della MLP con una funzione parametrica $\hat{F}_w$ ha come fine quello di far apprendere F e quindi risolvere il problema $\min_w \mathrm{Loss}(w)$.
Esistono 3 approcci diversi a questo problema:
- approccio niente stocasticità:

$$\mathrm{Loss}_{\mathcal{T}}(w) := \sum_{i=1}^{T} |y_i - \hat{F}(x_i)| \ , \tag{3.18}$$

dove $y_i = F(x)$. Viene quindi fissata una loss e dei pesi all'MLP, finchè non si raggiunge un criterio di arresto i pesi vengono aggiornati con un generico metodo di discesa del gradiente.
- approccio stocasticità pura:

$$\mathrm{Loss}_{(x,y)}(w) := |y - \hat{F}(x)| \tag{3.19}$$

In questo caso la loss è parametrica per ogni coppia di valori nel training set. Dopo aver assegnato i pesi iniziali finchè non si raggiunge un criterio di arresto si mescolano le coppie e si aggiornano i pesi per ogni coppia con un generico metodo di discesa del gradiente.
- metodo minibatch:

$$\mathrm{Loss}_{\mathcal{B}}(w) := \sum_{(x,y)\in\mathcal{B}} |y - \hat{F}(x)| \tag{3.20}$$

Quest'ultimo è l'approccio più utilizzato perchè si definisce ancora una loss parametrica ma per un sottoinsieme B di punti e non per ogni coppia di punti del training set. Anche in questo caso dopo aver assegnato i pesi si fissa un numero $K \in \mathbb{N}$ di minibatch in cui dividere il training set e si procede analogamente agli altri metodi per ciascun minibatch.

Un generico metodo di discesa del gradiente viene definito nel seguente modo:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) , \quad \forall\ k \geq 0 , \tag{3.21}$$

dove $\alpha_k \in \mathbb{R}^+$ è un fattore di moltiplicazione del passo di discesa e $-\nabla f(x_k)$ è la direzione di più ripida discesa per f nel punto x_k. Sotto opportune ipotesi per f e nella scelta degli $\{\alpha_k\}_{k\in\mathbb{N}}$ si ha che il metodo del gradiente converge al punto di minimo assoluto di f.
Nel caso dell'MLP vengono quindi definiti due parametri: l'epoca di addestramento e il learning rate α. Il primo fa riferimento al numero di volte che vengono riaggiornati i pesi con un generico metodo di discesa del gradiente. α viene definito invece come il tasso di apprendimento, può variare da epoca a epoca e rappresenta il coefficiente per cui moltiplicare la direzione di massima discesa.
Infine, i principali criteri di arresto per una rete neurale possono essere fissare un numero massimo $e_{\max} \in \mathbb{N}$ di epoche da eseguire oppure fissare una numero massimo $p \in \mathbb{N}$ di epoche di tolleranza rispetto al quale posso accettare che la loss sul validation set cresca invece di diminuire. Questo permette di evitare il fenomeno dell'overfitting e allo stesso tempo di ridurre i tempi di addestramento.

	trainval mlp	trainval over mlp	trainval under mlp	test mlp
0	3305	3305	104	3294
1	104	3305	104	116

Anche in questo caso viene risuddiviso il data set nello stesso modo, ossia training set 30%, validation set 20% e test set 50%. Tuttavia, attraverso il nome trainval viene definito l'insieme dei dati che comprende sia il training che il validation set. Questa operazione è stata fatta dato che il validation set viene creato internamente dalla classe MLPClassifier.
Successivamente, eseguiamo una ricerca a griglia dei migliori iperparametri riguardanti l'MLP. Le analisi riportate fanno riferimento al miglior MLP trovato, analogamente a quanto fatto per le SVM.

Performance MLP senza PCA random oversampling

	acc	prec	rec	f1
trainval mlp no pca over	0.893797	0.894103	0.893797	0.893777
test mlp no pca over	0.840762	0.937422	0.840762	0.884538

Confusion matrix MLP senza PCA random oversampling

	no bancarotta	bancarotta
no bancarotta	2843	451
bancarotta	92	24

Performance MLP con PCA random oversampling

	acc	prec	rec	f1
train mlp over pca	0.995159	0.995205	0.995159	0.995159
test mlp over pca	0.842815	0.958386	0.842815	0.889255

Confusion matrix MLP con PCA random oversampling

	no bancarotta	bancarotta
no bancarotta	2794	500
bancarotta	36	80

Anche in questo caso valgono osservazioni simili a quelle già fatte nel caso delle SVM random oversampling con o senza PCA per quanto riguarda la scelta del metodo migliore. Tuttavia gli score con PCA risultano leggermente migliori rispetto a quelli senza PCA in questo caso, al contrario di quanto accadeva nelle SVM. Infine, i misclassification rate per l'MLP random oversampling senza e con PCA risultano essere entrambi circa del 16%.

3.12 MLP random undersampling

Performance MLP senza PCA random undersampling

	acc	prec	rec	f1
trainval mlp no pca under	0.807692	0.811844	0.807692	0.807050
test mlp no pca under	0.508504	0.938102	0.508504	0.645632

Confusion matrix MLP senza PCA random undersampling

	no bancarotta	bancarotta
no bancarotta	1670	1624
bancarotta	52	64

Performance MLP con PCA random undersampling

	acc	prec	rec	f1
train mlp under pca	0.985577	0.985622	0.985577	0.985577
test mlp under pca	0.432551	0.967182	0.432551	0.568055

Confusion matrix MLP con PCA random undersampling

	no bancarotta	bancarotta
no bancarotta	1360	1934
bancarotta	1	115

Sebbene il numero dei dati di training più validation nel caso dell' MLP random undersampling sia il doppio di quelli di training dell'SVM random undersampling le conclusioni sono sempre le stesse visto chhe in entrambi i casi si tratta di numeri piccoli (SVM random undersampling: #{1}=56, #{0}=56, MLP random undersampling: #{1}=104, #{0}=104). Infine, i misclassification rate dell'MLP random oversampling senza e con PCA risultano essere rispettivamente 49% e 57%.

3.13 Confronto dei metodi utilizzati random oversapling & random undersampling

In generale si può osservare che i modelli più corretti hanno ottimi score grazie al maggior numero di predizioni corrette dei casi di non bancarotta rispetto a quelli di bancarotta. Questo è dovuto all'iniziale squilibrio delle classi presenti nel dataset.

Confrontando le diverse tabelle emerge che la LDA random oversampling senza PCA sia il metodo migliore in termini predittivi. Oltre a questo metodo anche l'MLP random oversampling con e senza pca, l'SVM random oversampling con e senza PCA e la LDA random undersampling senza PCA funzionano molto bene. In generale il random oversampling funziona meglio del random undersampling mentre la PCA non sempre è superflua. Come già detto, il motivo principale dello scarso funzionamento del random undersampling potrebbe esser dovuto all'eccessivo sbilanciamento iniziale delle classi che porta ad ottenere data set con un numero molto basso di osservazioni, rendendo quindi difficili le previsioni per gli algoritmi.

3.14 Analisi delle features fondamentali

Nel seguito della trattazione viene effettuata una analisi delle categorie di Financial ratios fondamentali. L'obiettivo è testare il miglior metodo trovato fino a questo punto (LDA random oversampling senza PCA) applicandolo al data set senza una categoria in particolare. Facendo così si potrà osservare quali categorie sono più critiche e quindi utili nella predizione della bancarotta.

3.14.1 Data set senza solvency

```
LDA random oversampling no solvency
```

	acc	prec	rec	f1
train lda no pca no solvency over	0.891028	0.891455	0.891028	0.890998
test lda no pca no solvency over	0.867155	0.965847	0.867155	0.905958

	no bancarotta	bancarotta
no bancarotta	2869	431
bancarotta	22	88

```
LDA random oversampling all

                           acc      prec       rec        f1
train lda no pca over  0.898909  0.899909  0.898909  0.898846
test lda no pca over   0.870088  0.964501  0.870088  0.907565

               no bancarotta   bancarotta
no bancarotta            2883         417
bancarotta                 26          84
```

Gli score restano circa invariati.

3.14.2 Data set senza profitability

```
LDA random oversampling no profitability

                                        acc      prec       rec        f1
train lda no pca no profitability over  0.886329  0.886841  0.886329  0.886292
test lda no pca no profitability over   0.855425  0.961996  0.855425  0.898233

               no bancarotta  bancarotta
no bancarotta           2838         462
bancarotta                31          79


LDA random oversampling all

                           acc      prec       rec        f1
train lda no pca over  0.898909  0.899909  0.898909  0.898846
test lda no pca over   0.870088  0.964501  0.870088  0.907565

               no bancarotta   bancarotta
no bancarotta            2883         417
bancarotta                 26          84
```

Gli score peggiorano anche se di poco.

3.14.3 Data set senza turnover ratios

```
LDA random oversampling no turnover ratios

                           acc      prec       rec     f1
train lda no pca over  0.876629  0.877179  0.876629     0.876584
test lda no pca over   0.859531  0.964402  0.859531     0.901074
```

```
               no bancarotta  bancarotta
no bancarotta           2846         454
bancarotta                25          85


LDA random oversampling all

                            acc      prec       rec        f1
train lda no pca over  0.898909  0.899909  0.898909  0.898846
test lda no pca over   0.870088  0.964501  0.870088  0.907565

               no bancarotta   bancarotta
no bancarotta            2883         417
bancarotta                 26          84
```

Gli score restano circa invariati.

3.14.4 Data set senza capital structure ratios

```
LDA random oversampling no capital structure ratios

                            acc      prec       rec     f1
train lda no pca over  0.896332  0.897027  0.896332     0.896287
test lda no pca over    0.031672  0.323585  0.031672    0.003117


               no bancarotta  bancarotta
no bancarotta              2        3298
bancarotta                 4         106


LDA random oversampling all

                            acc      prec       rec        f1
train lda no pca over  0.898909  0.899909  0.898909  0.898846
test lda no pca over   0.870088  0.964501  0.870088  0.907565

               no bancarotta   bancarotta
no bancarotta            2883         417
bancarotta                 26          84
```

Gli score peggiorano nettamente.

3.14.5 Data set senza cash flow ratios

```
LDA random oversampling no cash flow ratios
```

```
                            acc      prec       rec     f1
train lda no pca over  0.898303  0.899351  0.898303     0.898236
test lda no pca over    0.869208  0.965200  0.869208    0.907127


               no bancarotta  bancarotta
no bancarotta           2878         422
bancarotta                24          86


LDA random oversampling all

                            acc      prec       rec        f1
train lda no pca over  0.898909  0.899909  0.898909  0.898846
test lda no pca over   0.870088  0.964501  0.870088  0.907565

               no bancarotta   bancarotta
no bancarotta            2883         417
bancarotta                 26          84
```

Gli score restano circa invariati.

3.14.6 Data set senza growth

```
LDA random oversampling no growth

                                    acc      prec       rec        f1
train lda no pca no growth over  0.899060  0.900049  0.899060  0.898998
test lda no pca no growth over   0.872141  0.965337  0.872141  0.908945

               no bancarotta  bancarotta
no bancarotta           2888         412
bancarotta                24          86


LDA random oversampling all

                            acc      prec       rec        f1
train lda no pca over  0.898909  0.899909  0.898909  0.898846
test lda no pca over   0.870088  0.964501  0.870088  0.907565

               no bancarotta   bancarotta
no bancarotta            2883         417
bancarotta                 26          84
```

Gli score migliorano anche se di poco.

3.14.7 Data set senza others

```
LDA random oversampling no others

                                    acc      prec       rec        f1
train lda no pca no others over  0.898606  0.899654  0.898606  0.898539
test lda no pca no others over   0.653079  0.937139  0.653079  0.763871

               no bancarotta  bancarotta
no bancarotta           2191        1109
bancarotta                74          36


LDA random oversampling all

                             acc      prec       rec        f1
train lda no pca over  0.898909  0.899909  0.898909  0.898846
test lda no pca over   0.870088  0.964501  0.870088  0.907565

               no bancarotta   bancarotta
no bancarotta            2883         417
bancarotta                 26          84
```

Gli score peggiorano.

3.15 Conclusioni del ricampionamento random oversampling e random undersampling

Lo studio dei Finacial ratios come metodo di predizione effettiva di bancarotta evidenzia che le tecniche applicate sono efficaci. In particolare il pareggio delle classi effettuato tramite il random over e under sampling permette una migliore rappresentazione dei dati tramite PCA. Nel caso del random undersampling inoltre si ha una buona separazione delle classi in R^3 a livello grafico. Dal punto di vista numerico invece l'LDA risulta essere il miglior metodo predittivo. Essa lavora meglio nel caso senza PCA sia per il random oversampling che per il random undersampling: tra queste 2 tecniche di ricampionamento è da preferire la prima per qualità di risultati. Non si può dire lo stesso per quanto riguarda le SVM e l'MLP. Entrambi infatti lavorano tendenzialmente allo stesso modo con il random oversampling sia con che senza PCA mentre col random undersampling i risultati sono decisamente poco precisi. In generale, come già detto, la scelta del metodo più efficace da utilizzare si baserà su quanta accuratezza si vuole avere nel predirre la classe bancarotta come tale. Confrontando tutti metodi utilizzati risulta tuttavia che, oltre alla LDA, anche l'MLP e le SVM, entrambi nel caso random oversampling con e senza PCA, sono buoni classificatori. Il ruolo della PCA nella riduzione della dimensionalità quindi non è da escludere a priori come strumento di appoggio per una migliore predizione. Infatti, in certi casi (vedi MLP random oversampling con e senza PCA) può risultare anche leggermente più conveniente utilizzarla. In conclusione, dall'analisi delle features fondamentali

emerge chiaramente come le categorie capital structure ratios e others siano indicatori critici per la predizione della bancarotta. Infatti, è molto più netto il peggioramento delle prestazioni del miglior metodo senza di essi rispetto a quando vengono esclusi altri. Allo stesso tempo si può osservare come l'indicatore growth abbia un influenza negativa nella predizione della bancarotta; senza di esso infatti gli score migliorano. Per quanto riguarda l'assenza dell'indicatore profitability gli score peggiorano leggermente ma i dati a disposizione non sono sufficienti per poter affermare con certezza che sia un indicatore critico. Analogamente si può pensare per gli indicatori restanti dato che non presentano grandi variazioni negli score.

3.16 Smote e Tomek's links

In quest'ultima parte vogliamo testare le nuove tecniche di ricampionamento presentate a inizio elaborato per confrontarle con quelle appena utilizzate. Riapplichiamo quindi gli stessi algoritmi presentati fino ad ora ma nel caso di Smote, Tomek's links e loro combinazioni. I risultati ottenuti sono mostrati nel seguito.

```
Performance LDA senza PCA Smote
```

	acc	prec	rec	f1
train lda no pca	0.909366	0.909822	0.909366	0.909341
test lda no pca	0.878006	0.963418	0.878006	0.912228

```
Confusion matrix LDA senza PCA Smote
```

	no bancarotta	bancarotta
no bancarotta	2914	386
bancarotta	30	80

```
Performance LDA con PCA Smote
```

	acc	prec	rec	f1
train lda pca	0.893604	0.894398	0.893604	0.893551
test lda pca	0.557771	0.967715	0.557771	0.685570

```
Confusion matrix LDA con PCA Smote
```

	no bancarotta	bancarotta
no bancarotta	1796	1504
bancarotta	4	106

Performance LDA senza PCA Smote+Tomek

	acc	prec	rec	f1
training lda no pca	0.911602	0.912199	0.911602	0.911570
test lda no pca	0.878299	0.964168	0.878299	0.912528

Confusion matrix LDA senza PCA Smote+Tomek

	no bancarotta	bancarotta
no bancarotta	2913	387
bancarotta	28	82

Performance LDA con PCA Smote+Tomek

	acc	prec	rec	f1
training lda pca	0.891651	0.892439	0.891651	0.891597
test lda pca	0.560411	0.967738	0.560411	0.687800

Confusion matrix LDA con PCA Smote+Tomek

	no bancarotta	bancarotta
no bancarotta	1805	1495
bancarotta	4	106

Performance SVM senza PCA Smote

	acc	prec	rec	f1
train svm no pca	0.925842	0.929241	0.925842	0.925695
val svm no pca	0.605623	0.648306	0.605623	0.575047
test svm no pca	0.848974	0.943775	0.848974	0.890640

Confusion matrix SVM senza PCA Smote

	no bancarotta	bancarotta
no bancarotta	2855	439
bancarotta	76	40

Performance SVM con PCA Smote

	acc	prec	rec	f1
train svm pca	0.976119	0.977207	0.976119	0.976105
val svm pca	0.810790	0.827588	0.810790	0.808333
test svm pca	0.702346	0.957614	0.702346	0.796814

Confusion matrix SVM con PCA Smote

	no bancarotta	bancarotta
no bancarotta	2306	988
bancarotta	27	89

Performance SVM senza PCA Smote+Tomek

	acc	prec	rec	f1
train svm no pca	0.934130	0.937061	0.934130	0.934020
val svm no pca	0.604022	0.650025	0.604022	0.571146
test svm no pca	0.850440	0.943848	0.850440	0.891518

Confusion matrix SVM senza PCA Smote+Tomek

	no bancarotta	bancarotta
no bancarotta	2860	434
bancarotta	76	40

Performance SVM con PCA Smote+Tomek

	acc	prec	rec	f1
train svm pca	0.974568	0.975799	0.974568	0.974551
val svm pca	0.816319	0.832234	0.816319	0.814092
test svm pca	0.690909	0.959078	0.690909	0.788622

Confusion matrix SVM con PCA Smote+Tomek

	no bancarotta	bancarotta
no bancarotta	2263	1031
bancarotta	23	93

Performance MLP senza PCA Smote

	acc	prec	rec	f1
trainval mlp no pca	0.878517	0.879440	0.878517	0.878443
test mlp no pca	0.814663	0.942176	0.814663	0.869854

Confusion matrix MLP senza PCA Smote

	no bancarotta	bancarotta
no bancarotta	2738	556
bancarotta	76	40

Performance MLP con PCA Smote

	acc	prec	rec	f1
train mlp pca	0.995008	0.995041	0.995008	0.995007
test mlp pca	0.828446	0.958941	0.828446	0.880389

Confusion matrix MLP con PCA Smote

	no bancarotta	bancarotta
no bancarotta	2742	552
bancarotta	33	83

Performance MLP senza PCA Smote+Tomek

	acc	prec	rec	f1
trainval mlp no pca	0.864695	0.865320	0.864695	0.864637
test mlp no pca	0.816129	0.941868	0.816129	0.870699

Confusion matrix MLP senza PCA Smote+Tomek

	no bancarotta	bancarotta
no bancarotta	2744	550
bancarotta	77	39

Performance MLP con PCA Smote+Tomek

	acc	prec	rec	f1
train mlp pca	0.993871	0.993924	0.993871	0.993871
test mlp pca	0.778006	0.959216	0.778006	0.848185

Confusion matrix MLP con PCA Smote+Tomek

	no bancarotta	bancarotta
no bancarotta	2565	729
bancarotta	28	88

3.17 Adasyn e Tomek's links

Replichiamo quanto appena fatto nel paragrafo precedente con Adasyn al posto di Smote. I risultati ottenuti sono i seguenti:

Performance LDA senza PCA Adasyn

	acc	prec	rec	f1
train lda no pca	0.908345	0.908754	0.908345	0.908326
test lda no pca	0.876540	0.963711	0.876540	0.911383

Confusion matrix LDA senza PCA Adasyn

	no bancarotta	bancarotta
no bancarotta	2908	392
bancarottar	29	81

Performance LDA con PCA Adasyn

	acc	prec	rec	f1
train lda pca	0.891473	0.892488	0.891473	0.891412
test lda pca	0.559824	0.967733	0.559824	0.687305

Confusion matrix LDA con PCA Adasyn

	no bancarotta	bancarotta
no bancarotta	1803	1497
bancarotta	4	106

Performance LDA senza PCA Adasyn+Tomek

	acc	prec	rec	f1
training lda no pca	0.909691	0.910289	0.909691	0.909660
test lda no pca	0.878592	0.964184	0.878592	0.912709

Confusion matrix LDA senza PCA Adasyn+Tomek

	no bancarotta	bancarotta
no bancarotta	2914	386
bancarotta	28	82

Performance LDA con PCA Adasyn+Tomek

	acc	prec	rec	f1
training lda pca	0.894640	0.896080	0.894640	0.894547
test lda pca	0.559824	0.967733	0.559824	0.687305

Confusion matrix LDA con PCA Adasyn+Tomek

	no bancarotta	bancarotta
no bancarotta	1803	1497
bancarotta	4	106

Performance SVM senza PCA Adasyn

	acc	prec	rec	f1
train svm no pca	0.924547	0.927604	0.924547	0.924414
val svm no pca	0.603824	0.645680	0.603824	0.571415
test svm no pca	0.850733	0.943863	0.850733	0.891693

Confusion matrix SVM senza PCA Adasyn

	no bancarotta	bancarotta
no bancarotta	2861	433
bancarotta	76	40

Performance SVM con PCA Adasyn

	acc	prec	rec	f1
train svm pca	0.973089	0.974464	0.973089	0.973069
val svm pca	0.799618	0.815542	0.799618	0.796863
test svm pca	0.686804	0.957662	0.686804	0.785736

Confusion matrix SVM con PCA Adasyn

	no bancarotta	bancarotta
no bancarotta	2252	1042
bancarotta	26	90

Performance SVM senza PCA Adasyn+Tomek

	acc	prec	rec	f1
train svm no pca	0.928915	0.932044	0.928915	0.928768
val svm no pca	0.606354	0.649526	0.606354	0.575227
test svm no pca	0.843695	0.943518	0.843695	0.887472

Confusion matrix SVM senza PCA Adasyn+Tomek

	no bancarotta	bancarotta
no bancarotta	2837	457
bancarotta	76	40

Performance SVM con PCA Adasyn+Tomek

	acc	prec	rec	f1
train svm pca	0.972932	0.974315	0.972932	0.972908
val svm pca	0.815188	0.831280	0.815188	0.812866
test svm pca	0.693548	0.959139	0.693548	0.790508

Confusion matrix SVM con PCA Adasyn+Tomek

	no bancarotta	bancarotta
no bancarotta	2272	1022
bancarotta	23	93

Performance MLP senza PCA Adasyn

	acc	prec	rec	f1
trainval mlp no pca	0.864132	0.864198	0.864132	0.864125
test mlp no pca	0.833431	0.943031	0.833431	0.881284

Confusion matrix MLP senza PCA Adasyn

	no bancarotta	bancarotta
no bancarotta	2802	492
bancarotta	76	40

Performance MLP con PCA Adasyn

	acc	prec	rec	f1
train mlp pca	0.994093	0.994148	0.994093	0.994093
test mlp pca	0.810850	0.956768	0.810850	0.869126

Confusion matrix MLP con PCA Adasyn

	no bancarotta	bancarotta
no bancarotta	2686	608
bancarotta	37	79

Performance MLP senza PCA Adasyn+Tomek

	acc	prec	rec	f1
trainval mlp no pca	0.871826	0.876733	0.871826	0.871392
test mlp no pca	0.777419	0.943216	0.777419	0.846859

Confusion matrix MLP senza PCA Adasyn+Tomek

	no bancarotta	bancarotta
no bancarotta	2604	690
bancarotta	69	47

Performance MLP con PCA Adasyn+Tomek

	acc	prec	rec	f1
train mlp pca	0.996023	0.996050	0.996023	0.996023
test mlp pca	0.773021	0.961072	0.773021	0.844985

Confusion matrix MLP con PCA Adasyn+Tomek

	no bancarotta	bancarotta
no bancarotta	2543	751
bancarotta	23	93

3.18 Conclusioni del ricampionamento Smote, Adasyn, Tomek's Links e loro combinazioni

Riapplicando gli stessi algoritmi di apprendimento usati precedentemente ma con altre tecniche di ricampionamento otteniamo risultati che evidenziano alcuni lati positivi di Smote, Adasyn e Tomek's Links rispetto al semplice ricampionamento casuale. Osservando i principali score analizzati possiamo affermare che l'utilizzo di Smote o la combinazione di esso assieme ai Tomek's Links permetta un leggero, ma non netto, incremento delle prestazioni rispetto al random oversampling. Il confronto invece rispetto al random undersampling evidenzia chiaramente come la combinazione di oversampling (Smote) e undersampling (Tomek's Links) restituiscano migliori risultati rispetto al semplice undersampling (random undersampling). Questo era facilmente intuibile dato il basso numero di osservazioni per la classe minoritaria. Le tecniche di ricampionamento presentate risultano inoltre non essere soggette ad alterazioni nel caso di riduzione della dimensionalità. In effetti i ragionamenti fatti nel caso senza PCA valgono anche con l'applicazione della PCA ad eccezione delle SVM dove l'utilizzo della PCA insieme a Smote peggiora le prestazioni rispetto al caso del random oversampling. Tuttavia, questa eccezione va sempre contestualizzata a quanto si diceva già nel caso del ricampionamento casuale. Se viene preferito predirre i casi di bancarotta come tali si avrà un incremento degli score mentre peggioreranno se viene preferita la scelta di predirre i casi di non bancarotta come tali. Ancora una volta, come già detto, questo è dovuto alla conformazione del data set che presenta pochi casi di bancarotta ma che potrebbero essere di grande interesse per molti istituti finanziari. Quanto detto fino ora vale anche nel caso di Adasyn, essendo esso stesso una variante di Smote. Infine, confrontando tra di loro quete tre tecniche e le loro combinazioni emerge come non sia possibile stabilire un metodo migliore di altri. Ad esempio, le SVM con PCA risultano essere migliori nel caso di Smote rispetto a Adasyn mentre nell'MLP con e senza PCA vale l'opposto. In generale, è possibile osservare che, rispetto a questo data set, Adasyn insieme ai Tomek's Links sia da preferire al solo Adasyn mentre Smote singolarmente è più preciso rispetto a quando combinato con i Tomek's links.

Bibliografia

N. Chawla. Data mining for imbalanced datasets: An overview. In *The Data Mining and Knowledge Discovery Handbook*, 2005.

N. Chawla, K. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357, 2002.

A. Aldo Faisal Cheng Soon Ong Deisenroth, Marc Peter. *Mathematics for machine learning*. Cambridge University Press, 2020.

Thomas Taimre Radislav Vaisman Dirk P. Kroese, Zdravko I. Botev. *Data Science and Machine Learning: Mathematical and Statistical Methods*. CRC Press, 2019.

Alberto Fernández, Salvador García, Francisco Herrera, and N. Chawla. Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *J. Artif. Intell. Res.*, 61:863–905, 2018.

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

Hasite Trevor Tibshirani Robert James Gareth, Witten Daniela. *An Introduction to Statistical Learning with Applications in R*. Springer.

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. URL http://jmlr.org/papers/v18/16-365.html.

P.Norvig S.Russell. *Artificial Intelligence: A Modern Approach*. Pearson, 2020.

Cornellius Yudha Wijaya. 5 smote techniques for oversampling your imbalance data, 2020. URL https://towardsdatascience.com/5-smote-techniques-for-oversampling-your-imbalance-data-b8155bdbe2b5.