ALMA MATER STUDIORUM

UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA IN

INGEGNERIA E SCIENZE INFORMATICHE

Sperimentazione di Metodi Predittivi per il Credit Scoring

Relatore:

Prof. Dr. Gianluca Moro

Presentata da:

Davide Tombari

Terza Sessione di Laurea

Anno accademico 2017/2018

Indice

Elenco delle figure

Elenco delle tabelle

Alla mia famiglia

che in questi lunghi anni mi ha sempre appoggiato.

In questa giornata i loro sacrifici e la loro fiducia
verranno ripagati...

Capitolo 1

Introduzione

A partire dal terzo trimestre del 2018 la Federal Reserve Bank ha stimato che il valore dei prestiti al consumo presso tutte le banche commerciali ammonta a 1,49 trilioni di dollari negli Stati Uniti.[1]

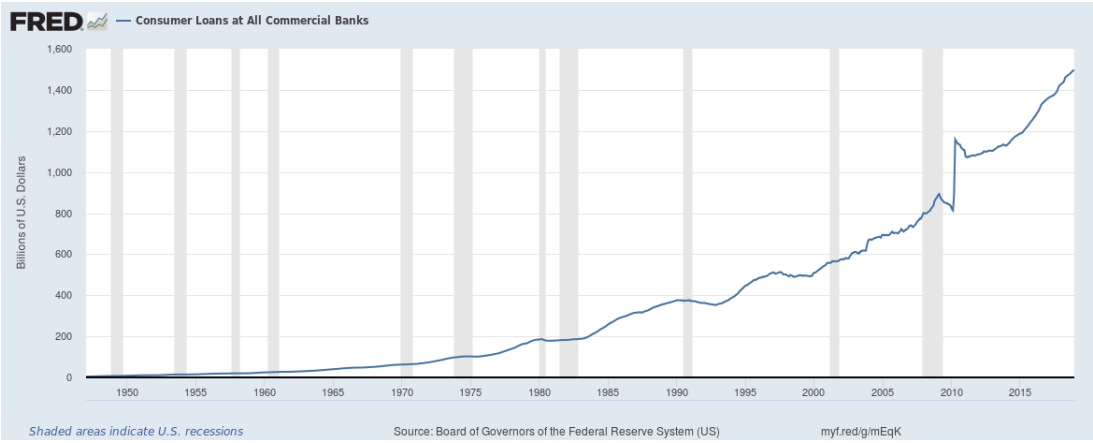

FIGURA 1.1: Prestiti al consumo presso tutte le banche commerciali statunitensi

A lungo sono state studiate tecniche di previsione di fallimento e di credit scoring nell'ambito di concessione di prestiti bancari. Recentemente nuove tecniche come il machine learning e le reti neurali hanno portato cambiamenti significativi in questa area. Fin dalle origini operatori del settore hanno dovuto prendere decisioni relative alla concessione o negazione di credito. Prima di elargire un prestito le banche valutano la possibilità che il soggetto possa restituire tale somma. Errate scelte comportano ingenti costi. Sulla base di precedenti studi scientifici si vogliono testare nuove

[1] Fonte: https://fred.stlouisfed.org/series/CONSUMER

tecniche finora poco utilizzate a causa della loro recente introduzione. Inizialmente verranno utilizzati metodi tradizionali già testati in precedenza, successivamente ci si concentrerà su modelli più recenti. A fronte dei risultati ottenuti saranno effettuate diverse considerazioni utilizzando un confronto con la letteratura, infine verranno tratte le opportune conclusioni.

Capitolo 2

Letteratura

Gran parte della letteratura si è occupata dell'applicazione e della conseguente valutazione di modelli predittivi nell'ambito della concessione del credito. Questi modelli stimano la solvibilità di un soggetto sulla base di alcune variabili. Relativamente ad una persona fisica[1] possiamo considerare l'ammontare del conto in banca, i membri del nucleo familiare, lo stato civile, stipendio mensile, la storia dei precedenti prestiti... mentre per una persona giuridica[2] vengono in aiuto indici finanziari/economici, bilancio... Il tutto è spesso tradotto in un problema di classificazione binaria o di multi-classificazione. Ad oggi, sono state utilizzate diverse tecniche tra cui semplici classificatori, ensemble, reti neurali. La letteratura ci insegna che l'utilizzo di un singolo dataset non è particolarmente indicato in questo ambito a causa del fatto che ogni istituto finanziario possiede diversi tipi di dati con le proprie variabili, inoltre ci possono essere fattori dipendenti dal luogo utili al riconoscimento di pattern di classificazione. Per i suddetti motivi in questo studio sono stati utilizzati 4 datasets provenienti da diversi continenti al fine di evidenziare il comportamento dei singoli modelli non sullo specifico dataset ma nel complesso, garantendo una visione più ampia nonché attendibile delle performances degli stessi. La qualità di un modello si verifica tramite l'ausilio degli opportuni indici statistici. Alcuni studi utilizzano un solo tipo di misura o misure dello stesso tipo il che risulta abbastanza limitante,

[1] In diritto, una persona fisica è un essere umano dotato di capacità giuridica, quindi soggetto di diritto, es: Mario Rossi. Fonte: https://it.wikipedia.org/wiki/Persona_fisica 24/01/2019

[2] La persona giuridica è un insieme di persone fisiche e di beni organizzato per il conseguimento di determinati obiettivi di interesse collettivo, es: Mario Rossi s.p.a. Fonte: https://www.regione.emilia-romagna.it/urp/faq/attivita-istituzionale/cosa-e-la-persona-giuridica 30/01/2019

per maggiori informazioni si consiglia la visione della tabella 1 dell'articolo [1] in bibliografia. Nell'ambito del Credit Scoring è opportuno utilizzare indici con caratteristiche diverse che permettono di verificare con maggiore meticolosità le performances dei modelli utilizzati. Particolare importanza assume il bilanciamento delle previsioni, aspetto che indici come l'accuratezza non possono tenere in considerazione ed è per questo motivo che verranno utilizzati diversi indicatori di performance 4.4. È possibile commettere due tipi di errori falsi positivi (FP) e falsi negativi (FN) vedi 4.2. Una caratteristica interessante di questi due tipi di errore è che non hanno lo stesso peso, infatti bisogna porre particolare attenzione ai falsi positivi perché concedere un prestito a soggetti che non possono restituirlo o che mostrano comportamenti indesiderati ha un costo maggiore rispetto a non elargire un prestito che poteva essere concesso. Tratteremo successivamente questo aspetto fondamentale.

Capitolo 3

Metodi Predittivi Utilizzati

Come si può facilmente intuire l'obiettivo di uno scorecard non è nient'altro che la classificazione. Nello specifico si tratta di classificazione binaria caratterizzata dalle classi "prestito buono", "prestito pessimo". Supponiamo di avere un insieme di $\boldsymbol{n}$ feature, possiamo definirle tramite un vettore $\boldsymbol{x}=[x_1,x_2,...x_n]$, infine definiamo la variabile binaria $\boldsymbol{y}\in[0\ ;\ 1]$, che assume il valore 1 quando si tratta di un buon prestito, oppure 0 nel caso opposto. Il modello utilizzato produce in output la probabilità di default del soggetto. Sulla base di questo valore il prestito viene concesso se tale probabilità è minore di una certa soglia $\boldsymbol{s}$. Esistono due grandi famiglie di algoritmi:

- Classificatori utilizzati in problemi di classificazione con due o più classi. Se si vuole distinguere un animale in base alla specie di appartenenza possiamo utilizzare questo tipo di algoritmi dove le specie sono appunto le diverse classi.

- Regressione stima di variabili nel continuo, per esempio la previsione di temperatura del giorno seguente.

Gli esempi successivi trattano alcuni modelli di regressione ed altri di classificazione per il semplice fatto che con piccole modifiche alcuni algoritmi di regressione possono essere utilizzati in problemi di classificazione perciò è stato descritto l'algoritmo nella sua forma base. Nella sezione 3.1 viene fornito un esempio su come è possibile trasformare una regressione lineare in un classificatore binario.

3.1 Linear Regression

Con la regressione lineare multivariata si stima una variabile dipendente $\hat{y}$ sulla base di più variabili indipendenti x_i. Si cerca di tracciare un iperpiano $\hat{y} = \theta^T x$ che meglio approssimi questo insieme. θ è il vettore dei pesi (per convenzione θ_0 è il termine noto con x_0=1). Per trovare l'iperpiano migliore è necessario minimizzare l'errore commesso nell'approssimazione. La metrica più comunemente utilizzata è la media dei quadrati $mse = \frac{1}{m}\sum_{k=1}^{m}(\hat{y}_i - y_i)^2$. Per risolvere questo tipo di problemi esistono due approcci possibili:

- metodo diretto è ottimo ma risolve solo problemi convessi, non è sempre applicabile con i big data a causa del suo costo computazionale [16].
- discesa del gradiente non garantisce soluzioni ottime, è eseguibile in parallelo anche per problemi non convessi ed è per questo largamente utilizzato [16].

Nella figura 3.1 a sinistra viene riportato un esempio di retta con una pessima approssimazione dei dati mentre a destra, dopo alcune iterazioni dell'algoritmo di discesa del gradiente, una retta con un'approssimazione decisamente migliore. Non è particolarmente utilizzata per problemi di classificazione tuttavia con qualche piccolo accorgimento è possibile utilizzarla: se il valore prodotto è al di sotto di una certa soglia allora il caso in esame viene assegnato alla classe positiva/negativa e viceversa.

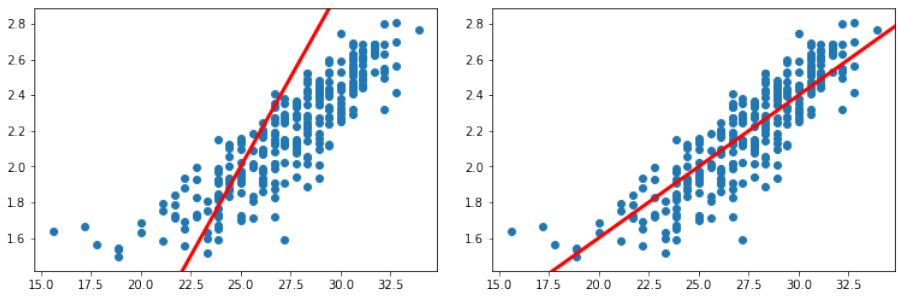

FIGURA 3.1: Linear Regession

Fonte: [16]

3.2 Gradiente

La derivata parziale di una funzione f con n variabili è la derivata della funzione secondo la variabile v, scelta tra le n disponibili. Per esempio nel caso di funzione a due variabili:

$$f(a,b) = 3a + b^2 \qquad \frac{\delta f}{\delta a} = 3 \qquad \frac{\delta f}{\delta b} = 2b$$

A questo punto è possibile definire il gradiente ∇f come il vettore contenente le derivate parziali della funzione f. Il significato geometrico del gradiente calcolato nel punto x rappresenta la pendenza della curva nel punto scelto, ciò permette di determinare la direzione in cui la curva sale o scende più rapidamente [16].

3.2.1 Discesa Del Gradiente

La discesa del gradiente è utilizzata per trovare un minimo locale della funzione f seguendo il suo gradiente. Partendo da un punto iniziale x_0 si valuta il valore della funzione e si calcola il suo gradiente, ci si sposta nel punto successivo x_1 sottraendo dal punto x_o un vettore proporzionale al gradiente della funzione calcolato nel punto x_0. Questa procedura si ripete fino alla convergenza ad un minimo secondo la seguente formula:

$$x_{k+1} = x_k - \eta \cdot \nabla f(x_k)$$

dove η rappresenta il passo di discesa (step size). Se troppo basso aumenta il numero di iterazioni necessarie alla convergenza, viceversa il tempo impiegato potrebbe essere alto a causa di salti troppo ampi [16]. Di seguito viene riportato un esempio nel caso di regressione lineare univariata.

$$E(\alpha,\beta) = \frac{1}{m}\sum_{i=1}^{m}(\alpha x_i + \beta - y_i)^2$$

Le derivate parziali rispetto ad α e β sono rispettivamente:

$$\frac{\delta E}{\delta \alpha} = \frac{2}{m}\sum_{i=1}^{m} x_i(\alpha x_i + \beta - y_i) \qquad \frac{\delta E}{\delta \beta} = \frac{2}{m}\sum_{i=1}^{m}(\alpha x_i + \beta - y_i)$$

Ad ogni iterazione i parametri sono aggiornati nel modo seguente:

$$\alpha_{k+1} = \alpha_k - \frac{2\eta}{m}\sum_{i=1}^{m} x_i(\alpha_k x_i + \beta_k - y_i) \qquad \beta_{k+1} = \beta_k - \frac{2\eta}{m}\sum_{i=1}^{m}(\alpha_k x_i + \beta_k - y_i)$$

3.3 Logistic Regression

Viene comunemente utilizzata per determinare la probabilità che un'istanza appartenga ad una determinata classe. Per esempio se la probabilità di appartenenza alla classe positiva è maggiore del 50% allora viene assegnata alla classe positiva altrimenti a quella negativa. È definita nel seguente modo:

$$\hat{p} = h_\theta(x) = \sigma(\theta^T \cdot x)$$

dove $\sigma(t) = \frac{1}{1+exp(-t)}$ è una funzione sigmoidale che genera valori tra 0 ed 1 [3]. Una volta determinata la probabilità che un'istanza appartenga alla classe positiva è facile predire $\hat{y}$.

$$\hat{y} = \begin{cases} 0 & if\ \hat{p} < 0.5 \\ 1 & if\ \hat{p} \geq 0.5 \end{cases} \tag{3.1}$$

Da notare che $\sigma(t) < 0.5$ quando $t < 0$, e $\sigma(t) \geq 0.5$ quando $t \geq 0$, così la regressione logistica predice 1 se $\theta \cdot x$ è positivo, 0 altrimenti [3].

Per addestrare il modello è necessario definire una funzione di costo

$$c(\theta) = \begin{cases} -\log(\hat{p}) & if\ y = 1 \\ -\log(1-\hat{p}) & if\ y = 0 \end{cases} \tag{3.2}$$

Questa funzione di costo ha senso perché se $y = 1$ con $\hat{p}$ vicino allo zero e se $y = 0$ con $\hat{p}$ vicino ad 1 il costo è alto, viceversa il costo risulta essere basso ed è esattamente quanto richiesto perché con il crescere dell'errore il costo cresce mentre più ci si avvicina alla previsione corretta più il costo risulta basso [3]. Ora è possibile ricavare la funzione di costo su tutto il training set chiamata log loss [3]:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)]$$

Non esiste una forma chiusa per risolvere questa equazione tuttavia è possibile utilizzare la discesa del gradiente come per la regressione lineare.

3.4 Ridge Regression

La ridge regression è un'estensione della linear regression con l'aggiunta di regolarizzazione $\alpha \sum_{i=1}^{n} \theta_i^2$ il che forza l'algoritmo a rendere i pesi più bassi possibile [3]. Da notare che questo termine è utilizzato solo ed esclusivamente durante le fasi di training e non durante la predizione. Il parametro α controlla il grado di regolarizzazione, più è alto più la regolarizzazione è forte e i pesi tenderanno allo zero. Se α è zero la formula è la stessa della linear regression. La nuova funzione di costo è:

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} \theta_i^2$$

il termine θ_0 non è soggetto a regolarizzazione, infatti il valore iniziale dell'indice i è 1 [3].

3.5 Support Vector Machine

Le SVMs sono particolarmente indicate per la classificazione di datasets di piccole-medie dimensioni. Supponiamo che le due classi siano linearmente separabili, si cerca di individuare un iperpiano che divida le due classi scelto in modo tale che lo spazio tra i punti di confine (support-vectors) delle due classi sia massimizzato. Questa tecnica è comunemente detta large margin classification [3]. Da notare che aggiungere istanze al di fuori del margine non ha effetti sul decision boundary (la linea continua in figura 3.2). Se si impone che tutte le istanze stiano al di fuori del margine si tratta di hard margin classification 3.2, tuttavia questa tecnica non è sempre applicabile, per esempio nel caso in cui le due classi non siano linearmente separabili [3]. Per ovviare a questo problema risulta necessario eliminare alcuni vincoli. L'obiettivo è un trade-off tra ottenere un margine più ampio possibile e limitare le violazioni di margine (soft margin classification 3.3).

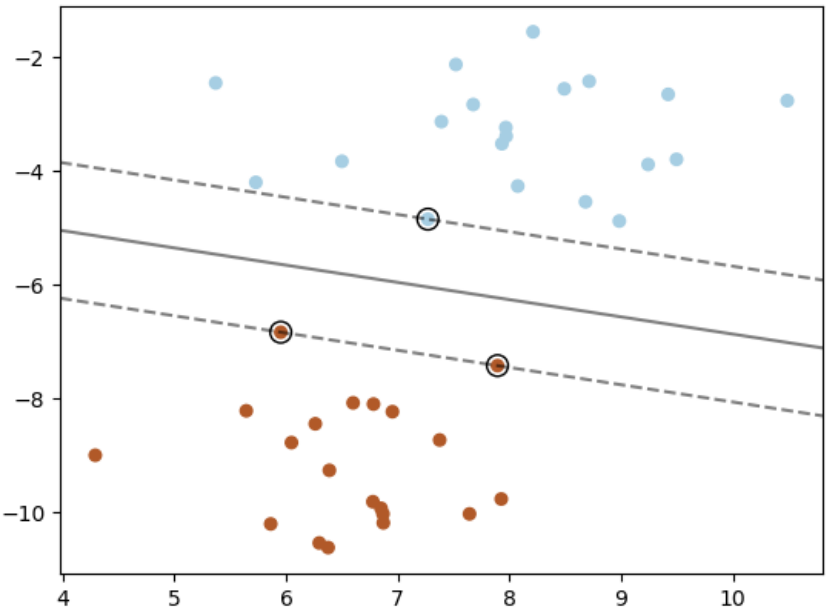

FIGURA 3.2: SVMs Hard Margin Classification

Fonte: https://scikit-learn.org/stable/modules/svm.html [14]

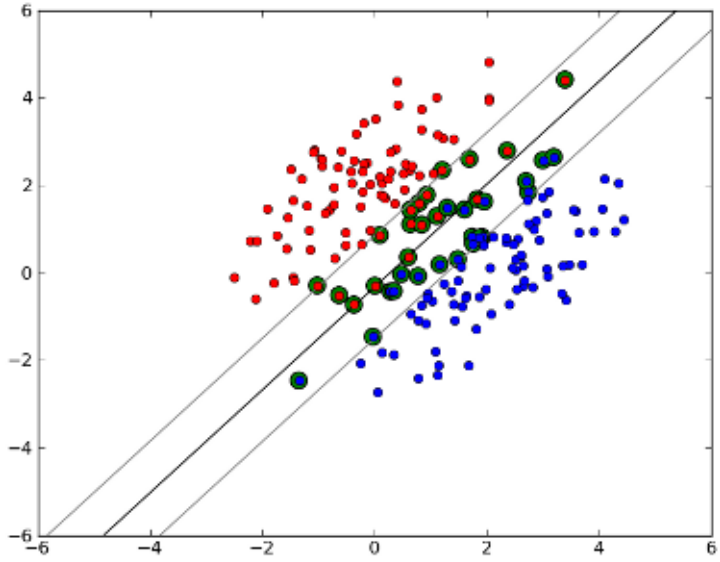

FIGURA 3.3: SVMs Soft Margin Classification

Fonte: https://stackoverflow.com/questions/9480605/what-is-the-relation-between-the-number-of-support-vectors-and-training-data-and

Definita x come nuova istanza, w il vettore dei pesi, un classificatore SVM lineare predice la classe di appartenenza di x computando la seguente funzione [3]:

$$\hat{y} = \begin{cases} 0 & if \quad w^T \cdot x + b < 0 \\ 1 & if \quad w^T \cdot x + b \geq 0 \end{cases} \tag{3.3}$$

Addestrare una svm significa trovare i valori di w e b che rendono il margine più ampio possibile senza violazioni nel caso di hard margin classification oppure limitandolo in presenza di soft margin classification.

3.6 Kernel Polinomiale

Al contrario di quanto visto finora supponiamo che le classi non siano linearmente separabili, in questo caso c'è bisogno di una maggiore complessità del modello aggiungendo per esempio termini di grado superiore. Più il grado aumenta più la funzione diventa complessa. Attenzione però all'esplosione della dimensionalità dovuta all'introduzione di nuove variabili. Un polinomio di grado 10 con 10 variabili genera 184 756 termini prima ancora di effettuare la classificazione. Un polinomio di grado g con n variabili genera $\binom{n+g}{g}$ termini [16].

3.6.1 Kernel Trick

Il kernel trick permette di lavorare in uno spazio ad n dimensioni senza introdurre nuove variabili [16]. Sviluppando il polinomio $(1+x_1z_1+x_2z_2)^2$ dove $x=(x_1,x_2)$, $z=(z_1,z_2)$ si ottiene:

$$(1+x_1z_1+x_2z_2)^2 = x_1^2z_1^2+2x_1x_2z_1z_2+x_2^2z_2^2+2x_1z_1+2x_2z_2+1$$

Da notare che corrisponde esattamente al prodotto scalare dei due vettori seguenti:

$$(x_1^2, x_1x_2\sqrt{2}, x_2^2, x_1\sqrt{2}, x_2\sqrt{2}, 1)\cdot(z_1^2, z_1z_2\sqrt{2}, z_2^2, z_1\sqrt{2}, z_2\sqrt{2}, 1)$$

In questo modo è stato effettuato un mapping in 6 dimensioni dei vettori x e z triplicando le dimensioni iniziali.

Sviluppando interamente il polinomio di prima si ottiene:

$$\begin{aligned}(1+x_1z_1+x_2z_2)^2 &= x_1^2z_1^2+2x_1x_2z_1z_2+x_2^2z_2^2+2x_1z_1+2x_2z_2+1\\ &= (x_1z_1+x_2z_2)^2+2(x_1z_1+x_2z_2)+1\\ &= (x^Tz)^2+2(x^Tz)+1\\ &= (x^Tz+1)^2\end{aligned}$$

dunque il quadrato del prodotto scalare dei valori iniziali x ed y in 2 dimensioni + 1 è uguale al prodotto scalare dei vettori trasformati in 6 dimensioni.

$$((x_1,x_2)\cdot(z_1,z_2)+1)^2 = (x_1^2,x_1x_2\sqrt{2},x_2^2,x_1\sqrt{2},x_2\sqrt{2},1)\cdot(z_1^2,z_1z_2\sqrt{2},z_2^2,z_1\sqrt{2},z_2\sqrt{2},1)$$

Ciò dimostra che $((x_1,x_2)\cdot(z_1,z_2)+1)^2$ corrisponde a lavorare in uno spazio a 6 dimensioni senza creare nuove variabili: 2 moltiplicazioni producono lo stesso effetto di 6 moltiplicazioni [16]. Il metodo kernel trick vale per ogni grado g e numero di dimensioni n:

$$x=[x_1...x_n],\quad z=[z_1...z_n]\quad Kernel(x,z)=(x\cdot z+1)^g$$

3.7 Ensemble

Ensemble learning è una tecnica che utilizza un insieme di modelli detto ensemble allo scopo di superare le performances del singolo. Supponiamo di porre una domanda ad un gruppo di persone e di aggregare le loro risposte, scopriremo che la risposta normalmente è migliore di quella di un esperto in materia, tale fenomeno è detto saggezza della folla (wisdom of the crowd) [3]. Questo concetto applicato al machine learning dà origine a nuovi algoritmi detti ensemble methods. Esistono diverse tecniche per determinare l'output di un ensemble, per esempio una tra le più semplici è la tecnica di hard-voting dove la classe predetta è la classe che ha ottenuto il maggior numero di voti. In caso di classificazione binaria è opportuno utilizzare un numero dispari di modelli per evitare casi come il pareggio. Un altro metodo è il soft-voting che utilizza la probabilità, l'idea è di dare più importanza ai modelli che danno in output probabilità più alte. Per utilizzarla ogni modello che compone l'ensemble deve generare delle probabilità e non delle classi di appartenenza. In genere quest'ultima ottiene risultati migliori rispetto alla precedente. Esistono tantissime altre tecniche avanzate di ensemble tuttavia sono difficilmente reperibili nei vari linguaggi. In questo studio sono stati effettuati diversi test con le tecniche citate ma i test iniziali non hanno prodotto i risultati sperati. Per questo motivo è stato abbandonato questo campo per concentrarsi su modelli più promettenti.

3.7.1 Ensemble Omogenei

Gli ensemble omogenei utilizzano un insieme di classificatori dello stesso tipo. Tra le tecniche più potenti attualmente disponibili troviamo gli alberi di decisione. XGBoost (Extreeme Gradient Boosting) è costituito da un insieme di classificatori deboli (decision trees). Agli errori viene dato un peso maggiore (boosting=potenziamento). Successivamente questi classificatori deboli vengono utilizzati per generare un unico forte classificatore. Non è scopo di questo studio trattare nel dettaglio questo argomento vista la sua complessità, si consiglia la lettura del capitolo 6 di [3] e di

consultare [15].

3.7.2 Ensemble Eterogenei

Con ensemble eterogenei si intende un insieme di classificatori con caratteristiche totalmente diverse. Per esempio è possibile utilizzare XGBoost, Logistic Regression, e Ridge Classifier e creare un ensemble per ottenere dei risultati migliori di quanto ottenuto dai singoli classificatori. L'idea di utilizzare modelli con caratteristiche totalmente diverse deriva dal fatto che le lacune di un modello possono essere colmate da un altro e viceversa. È possibile inoltre utilizzare dati di addestramento diversi per ogni algoritmo al fine di ottenere un forte grado di indipendenza.

3.8 Reti Neurali

Esistono diversi tipi di reti e non è scopo di questo studio trattarle, si cerca di fare una sintesi degli aspetti comuni. Possiamo immaginare una rete neurale come un grafo suddiviso in livelli. Ogni livello è composto da nodi chiamati neuroni che sono collegati ai neuroni del livello successivo. Il livello iniziale è chiamato input layer a cui vengono dati in pasto i dati. L'ultimo livello è chiamato output layer e ci fornisce l'output della rete. I livelli intermedi prendono il nome di hidden layer (la maggior parte delle reti utilizza un numero inferiore a 10 livelli). In ogni nodo viene calcolata una funzione $y = f(x)$ il cui risultato viene inviato ad uno o più nodi del livello successivo a seconda dell'architettura della rete. Nel caso l'output venga inviato a tutti i nodi del livello successivo questa rete prende il nome di rete densa. L'output di un nodo j di un livello, dato il vettore di input $x \in R$, è il seguente:

$$y_j = \sigma(w \cdot x + b_j)$$

w e b sono i parametri che devono essere appresi dalla rete, w rappresenta il vettore dei pesi, b il termine noto. Per ottimizzare i parametri è necessaria una funzione di

loss, per esempio *MSE*. Anche in questo caso si utilizza la discesa del gradiente. La funzione di attivazione σ va scelta in base alle caratteristiche del problema. Ne esistono diversi tipi, ad esempio Relu (Rectified Linear Unit) è definita come:

$$f(x) = \begin{cases} 0 & for\, x \leq 0 \\ x & for\, x > 0 \end{cases} \tag{3.4}$$

Il processo di addestramento è composto da due fasi principali [3]:

- forward pass le istanze di training vengono passate alla rete che genera il suo output e misura l'errore commesso.
- reverse pass si procede all'indietro per verificare l'errore in ogni connessione e si modificano i pesi di conseguenza.

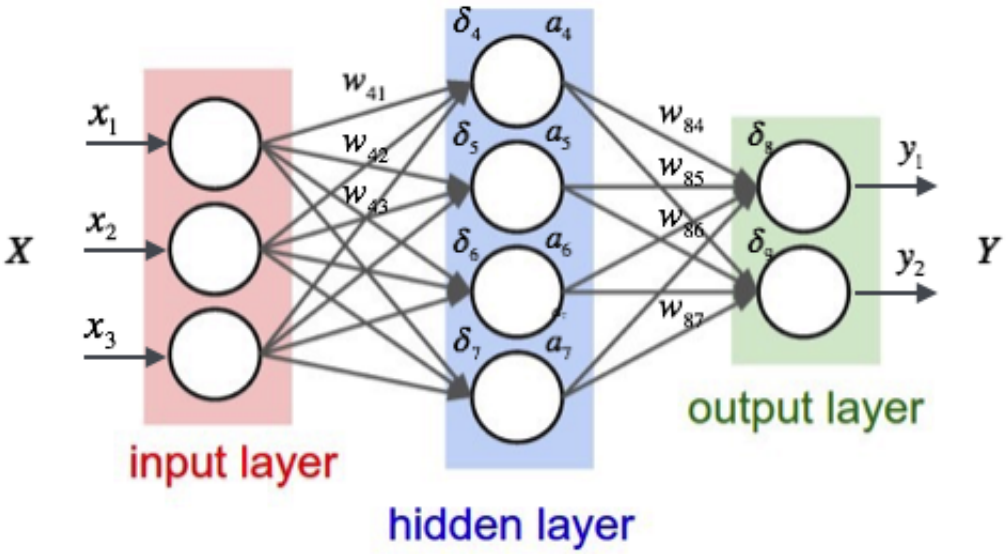

FIGURA 3.4: Neural Network

Fonte: https://medium.com/@curiousily/tensorflow-for-hackers-part-iv-neural-network-from-scratch-1a4f504dfa8

Capitolo 4

Esperimenti

4.1 Datasets

Nome	Nome esteso	Area Geografica	Casi	Variabili	Perc. di default
AC	australian	Australia	690	14	44.5
GC	german	Germania	1000	20	30
PAK	pak	Asia	50000	37	2.6
GMC	kaggle	Non fornita	150000	12	6.7

TABELLA 4.1: Datasets

Nella tabella 4.1 sono riportati i datasets utilizzati e le loro caratteristiche. Le colonne area geografica, casi e variabili sono autoesplicative e rappresentano rispettivamente il luogo di provenienza dei dati, il totale delle istanze e il numero di variabili indipendenti, mentre "Perc. di default" indica, in percentuale, quante istanze sono da attribuire al caso pessimo. Per esempio AC contiene 690 record, 14 variabili indipendenti e 44,5% di istanze negative. Sono stati utilizzati diversi datasets con diverse caratteristiche al fine di variare lo studio e renderlo più attendibile. I datasets AC e GC contengono un esiguo numero di record rispetto a PAK e GMC. Il range del numero di variabili varia da 12 a 37. La percentuale di default indica che il dataset AC è bilanciato in quanto contiene circa un egual numero di istanze per ciascuna classe,

mentre PAK è caratterizzato da un forte sbilanciamento che tratteremo nella sezione successiva.

4.2 Bilanciamento

È stato dimostrato che lo sbilanciamento costituisce un forte ostacolo per una corretta classificazione. In questi casi il classificatore utilizzato potrebbe prestare particolare attenzione alla classe di maggioranza a discapito di quella di minoranza. Esistono package come *smoote*[1] che nascono per risolvere queste casistiche utilizzando tecniche di under/over sampling, tuttavia non verrà utilizzato. È vero che lo sbilanciamento può inficiare sulle prestazioni dell'algoritmo,[2] tuttavia se lo sbilanciamento colpisce tutti i classificatori allo stesso modo allora viene influenzata la prestazione assoluta del classificatore ma non quella relativa; in base a quanto detto, ottenere un buon risultato in queste condizioni deriva dalla robustezza del classificatore rispetto allo sbilanciamento [1].

4.3 Indicatori di Qualità del Modello

Ogni modello deve poter verificare la validità del suo output, per fare ciò sono stati utilizzati 4 diversi indici: accuracy, precision, recall, f1. Gli ultimi tre menzionati sono stati calcolati per ogni classe. Prima di elencarne le proprietà è necessario definire una matrice detta matrice di confusione[3]:

[1] https://imbalanced-learn.readthedocs.io/en/stable/index.html

[2] non inteso in termini di performance tempistiche ma accuratezza, precision, recall..

[3] Fonte:https://it.quora.com/Come-%C3%A8-possibile-valutare-laccuratezza-di-uno-stimatore-basato-su-Neural-Networks 15/02/2019

		Predicted Class		
		p	**n**	**total**
Actual Class	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

TABELLA 4.2: Matrice di confusione

Fonte: [22]

- True Positives (TP) sono gli esempi classificati come appartenenti alla classe positiva che effettivamente appartenevano alla classe positiva [22].
- True Negatives (TN) sono gli esempi classificati come appartenenti alla classe negativa che effettivamente appartenevano alla classe negativa [22].
- False Positives (FP): sono gli esempi classificati come appartenenti alla classe positiva che in realtà appartenevano alla classe negativa [22].
- False Negatives (FN): sono gli esempi classificati come appartenenti alla classe negativa che in realtà appartenevano alla classe positiva [22].

Dunque ora possiamo definire:

- Accuracy è la percentuale di previsioni indovinate rispetto al totale [22].

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

- Precision indica quante delle classificazioni sono state erroneamente assegnate alla classe positiva [22].

$$Precision = \frac{TP}{TP + FP}$$

- Recall indica quanti esempi appartenenti alla classe positiva il classificatore ha mancato [22].

$$Recall = \frac{TP}{TP + FN}$$

- F1 score si calcola come la media armonica delle due precedenti [22].

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

4.4 Set-up

Per GC e AC innanzitutto sono stati rimpiazzati i valori mancanti con media/moda per attributi continui/categorici. Successivamente è stata applicata la standardizzazione. Si è deciso di procedere con la standardizzazione al posto della normalizzazione (in python StandardScaler/MinMaxScaler) per le seguenti ragioni: a seguito della normalizzazione tutti i valori assumono un range tra 0 e 1 tramite la seguente formula:

$$\widetilde{x} = \frac{x - min}{max - min}$$

Questo significa che in caso di outliers i valori vengono notevolmente "ammassati" e ciò può costituire un problema per certi modelli. La standardizzazione non soffre di quest'ultimo fenomeno e non vincola i risultati ad assumere uno specifico range:

$$\widetilde{x} = \frac{x - \mu}{\sigma}$$

Per quanto riguarda gli altri due datasets, è stata applicata solo la standardizzazione in quanto forniti dallo stesso autore dell'articolo Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research [1]. Questo ci porta ad avere un raffronto più preciso dei risultati ottenuti perché la fase di preparazione dei dati è la stessa. Per AC e GC è stata utilizzata una suddivisione 70/30 training/test con $k\_fold = 10$. La percentuale utilizzata per il testing in Pak e Kaggle è pari al 90% con $k\_fold = 7$. La scelta è motivata dal fatto che la fase di addestramento con datasets di queste dimensioni avrebbe richiesto tempi biblici non compatibili con questo studio. Facendo un esempio concreto Kaggle contiene 150 000 istanze, se si utilizzasse la prima suddivisione allora le istanze per il training sarebbero 105 000. Durante la cross validation con k=10 il training set viene diviso in k fold ognuna con 10 500 istanze. A turno k-1 vengono utilizzate per il training ed 1 per il testing, questo significa che dovremmo addestrare un dataset con $\frac{105\,000 \cdot 9}{10} = 94\,500$ istanze per 10 volte solo per una configurazione di parametri. Si pensi cosa succederebbe se si volessero testare un centinaio di configurazioni diverse: è evidente che non è plausibile a meno che non si disponga di un cluster di server. Tutto si complica nel caso di utilizzo delle reti neurali che accentuano ancora di più questa criticità. Per tagliare i tempi di addestramento senza sacrificare la robustezza dei risultati si è deciso di ridurre k da 10 a 7 e di compensare questa minore robustezza dei risultati con un aumento della dimensione del test set. Questo approccio ha permesso di tagliare notevolmente i costi di apprendimento perché in un singolo colpo è stata ridotta la dimensione del training set e sono stati diminuiti i tempi della cross-validation. È possibile che con una quantità maggiore di dati destinati al training si sarebbero potuti ottenere risultati migliori, tuttavia con le macchine che si avevano a disposizione è stata necessaria questo tipo di scelta. Durante le fasi conclusive è stato possibile evidenziare che la decisione presa non ha costituito un ostacolo al raggiungimento degli obiettivi prefissati nonché all'ottenimento di discreti risultati.

4.5 Trainig, Validation, Test sets

Nel machine learning e deep learning è bene suddividere il proprio set di dati in: training, validation, test. Il training set è utilizzato per la fase di addestramento, il validation set per verificare come si comporta il modello su nuovi dati. Nella fase finale, quando si ha ragione di credere di disporre di un buon modello, questo viene testato sul test set. Vi starete chiedendo per quale motivo in questo studio non è presente il validation set, è una domanda più che legittima. Prendere decisioni in base alle prestazioni del modello sul test set equivale, in un certo senso, ad un addestamento su quest'ultimo. È anche vero che quando si trattano datasets di piccole dimensioni non è sempre possibile una corretta suddivisione, il modello deve pur avere dati su cui essere addestrato. Qualora i dati lo permettano, è bene procedere alla suddivisione riportata nel titolo di questa sezione. Questo splitting sarebbe stato possibile per i datasets più numerosi PAK e Kaggle, ma avrebbe costituito un ostacolo all'apprendimento per i datasets con pochi record, in particolar modo su quello australiano (AC), causa le sue 690 istanze totali. Al fine di utilizzare lo stesso approccio con tutti i datasets la cross validation viene in nostro aiuto in quanto ci consente di avere dei risultati di testing senza utilizzare effettivamente il test set vero e proprio, oltretutto di una certa robustezza. Si consiglia di non utilizzare mai il test set fino alla fase finale per evitare quanto detto.

4.6 Tuning

Inizialmente è stata utilizzata la RandomSearchCv fissando il numero massimo di permutazioni al fine di esplorare le prestazioni del modello con diverse configurazioni. Le proprietà della cross-fold-validation con $k = 10$ garantiscono una certa robustezza dei risultati grazie al fatto che il training set viene diviso in k parti, a turno k-1 vengono utilizzate per il training, la restante per il testing. Il risultato è una media dei k test effettuati. È molto importante in questa fase provare più configurazioni

possibili, ciò non significa fare migliaia di permutazioni con una quantità esorbitante di parametri perché non è sempre possibile. L'idea è di scegliere i parametri più largamente utilizzati e mandare alcuni test. In base ai risultati ottenuti in questa fase iniziale si possono eliminare dei parametri, modificarne il range e aggiungerne di nuovi. Si consiglia di non stravolgere il set dei parametri tra un test e l'altro. In linea generale è meglio modificare un parametro per volta in modo tale che un miglioramento/peggioramento possa essere imputato direttamente ad esso. Modificare tanti parametri in una volta sola col fine di eliminare delle fasi di test può rivelarsi controproducente: qualora si ottenesse un peggioramento, ipotesi tutt'altro che remota, si dovrebbe tornare sui propri passi ripartendo dal test precedente vista la difficoltà di identificazione della causa. Risulta particolarmente complicato, soprattutto nelle fasi iniziali, riconoscere quale valore dei parametri risulti più appropriato perché in questa fase non si testa un parametro alla volta ma diverse combinazioni, non a caso è definita esplorazione. In queste circostanze non bisogna soffermarsi sull'andamento del singolo test ma è opportuno avere una visione globale per non intraprendere scelte sbagliate. Parlando in modo concreto se si utilizza la cross validation è bene salvare su file tutti i risultati dei test effettuati con i relativi parametri (in python è veramente semplice farlo). L'errore più comune che si può commettere è focalizzarsi solo sul modello migliore ed i suoi parametri: questo può portarci totalmente fuori strada perché, come preannunciato, diversi parametri variano. È vero che la cv porta risultati robusti ma questo non basta. Si è cercato di limitare il più possibile questo comportamento usando un metodo basato sulla *"media delle medie"*. I risultati riportati dalla cross validation sono già in formato tabellare ed è possibile caricarli in un dataframe pandas. Nelle righe vengono riportati i singoli test effettuati con le varie configurazioni dei parametri, nelle colonne il valore dei parametri del modello ed i risultati ottenuti (ricordo che i risultati sono delle medie). La tecnica utilizzata in fase di tuning è basata sul group-by, ovvero se si vuole sapere realmente quanto lo specifico parametro sia rilevante, si raggruppano i dati per la colonna del parametro

di interesse e si calcola la media per gruppo. Ogni gruppo è formato dai valori distinti del parametro scelto al variare degli altri parametri. In questo modo facendo la media per gruppo si può realmente scoprire quanto il valore del singolo parametro è buono/pessimo in generale perché permette di verificare il suo andamento con configurazione diverse di altri parametri. Questo tipo di risultato sarà tanto più robusto quanti più elementi compongono ogni gruppo. Per questo motivo è stato definito *"media delle medie"*. Ciò ha permesso di velocizzare i tempi di tuning in quanto ottiene risultati migliori rispetto al focalizzarsi sul singolo risultato migliore. Nella pagina successiva è riportato un piccolo esempio. Il metodo di scelta del modello nella cross validation inizialmente è stato *accuracy* ma ben presto si è rivelata totalmente inadeguata perché risulta troppo sensibile a fattori come lo sbilanciamento, perciò si è optato per l'utilizzo di un'altra metrica chiamata *balanced_accuracy* definita come la media della recall ottenuta in ogni classe.[4]

[4]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html#sklearn.metrics.balanced_accuracy_score

```
In [2]: #...impostazione dei sid e dell'ambiente...
        #TEST 5 RETI NEURALI
        #cartella contenente i pesi, parametri e risultati del test
        path='../test-new/australian/KerasClassifier/30-01-2019-11:35:03/'
        #utilizzo una classe apposita per il caricamento
        results5=Results(path,australianDataset)

Loaded model from disk

In [3]: #ordino i risultati in base alla  colonna "mean_test_balanced_accuracy"
        results5.cv_results.sort_values('mean_test_balanced_accuracy',
                                        ascending=False).head()
```

Out[3]:

	mean_test_accuracy	mean_test_balanced_accuracy	mean_train_accuracy	\
99	0.877847	0.878048	0.884518	
67	0.873706	0.874983	0.887738	
57	0.871636	0.874393	0.888658	
27	0.877847	0.873501	0.890038	
59	0.873706	0.871421	0.921328	

	mean_train_balanced_accuracy	param_activation	param_batch_size	\
99	0.884472	elu	32	
67	0.888224	elu	32	
57	0.890021	elu	128	
27	0.890209	elu	16	
59	0.920535	relu	128	

	param_decay	param_dropout_rate	param_epochs	param_hidden_layers	\
99	0.100	0.3	50	3	
67	0.010	0.4	100	2	
57	0.010	0.2	100	2	
27	0.001	0.4	50	1	
59	0.010	0.4	100	1	

	param_kernel_initializer	param_last_activation	param_lr	param_neurons	\
99	glorot_uniform	sigmoid	0.001	64	
67	lecun_uniform	sigmoid	0.001	32	
57	lecun_uniform	sigmoid	0.100	32	
27	lecun_uniform	sigmoid	0.001	16	
59	glorot_uniform	sigmoid	0.010	8	

	param_optimizer	std_test_accuracy	std_test_balanced_accuracy	\
99	RMSprop	0.053562	0.060537	
67	Adam	0.052013	0.060999	
57	RMSprop	0.040228	0.046395	
27	Adam	0.049407	0.055817	
59	RMSprop	0.056663	0.061025	

```
    std_train_accuracy  std_train_balanced_accuracy
99            0.006563                     0.006240
67            0.008163                     0.007796
57            0.016810                     0.017405
27            0.008419                     0.008103
59            0.005944                     0.005885
```

```
In [4]: #ordino i risultati in base alla colonna "mean_test_accuracy"
        results5.cv_results.sort_values('mean_test_accuracy', ascending=False).head()
```

```
Out[4]:     mean_test_accuracy  mean_test_balanced_accuracy  mean_train_accuracy  \
        27            0.877847                     0.873501             0.890038
        99            0.877847                     0.878048             0.884518
        59            0.873706                     0.871421             0.921328
        67            0.873706                     0.874983             0.887738
        15            0.871636                     0.869012             0.906831

            mean_train_balanced_accuracy param_activation  param_batch_size  \
        27                      0.890209              elu                16
        99                      0.884472              elu                32
        59                      0.920535             relu               128
        67                      0.888224              elu                32
        15                      0.905683             relu                16

            param_decay  param_dropout_rate  param_epochs  param_hidden_layers  \
        27        0.001                 0.4            50                    1
        99        0.100                 0.3            50                    3
        59        0.010                 0.4           100                    1
        67        0.010                 0.4           100                    2
        15        0.100                 0.4           100                    2

           param_kernel_initializer param_last_activation  param_lr  param_neurons  \
        27             lecun_uniform               sigmoid     0.001             16
        99           glorot_uniform               sigmoid     0.001             64
        59           glorot_uniform               sigmoid     0.010              8
        67             lecun_uniform               sigmoid     0.001             32
        15             lecun_uniform               sigmoid     0.010             32

           param_optimizer  std_test_accuracy  std_test_balanced_accuracy  \
        27            Adam           0.049407                    0.055817
        99         RMSprop           0.053562                    0.060537
        59         RMSprop           0.056663                    0.061025
        67            Adam           0.052013                    0.060999
        15         RMSprop           0.055731                    0.063912
```

```
    std_train_accuracy  std_train_balanced_accuracy
27            0.008419                     0.008103
99            0.006563                     0.006240
59            0.005944                     0.005885
67            0.008163                     0.007796
15            0.004411                     0.004880
```

```
In [5]: #calcola la media delle medie
        results5.getMean('param_neurons')
```

```
Out[5]:               mean_mean_test_accuracy  mean_mean_test_balanced_accuracy  \
        param_neurons
        8                            0.797707                          0.792867
        16                           0.777831                          0.770936
        32                           0.824323                          0.818540
        64                           0.741056                          0.733653

                      mean_mean_train_accuracy  mean_mean_train_balanced_accuracy  \
        param_neurons
        8                             0.854894                           0.850765
        16                            0.833631                           0.825613
        32                            0.882909                           0.880970
        64                            0.787808                           0.782079

                      mean_std_test_accuracy  mean_std_test_balanced_accuracy  \
        param_neurons
        8                           0.080338                         0.084038
        16                          0.077726                         0.077292
        32                          0.066509                         0.070532
        64                          0.097232                         0.090869

                      mean_std_train_accuracy  mean_std_train_balanced_accuracy
        param_neurons
        8                            0.046425                          0.049195
        16                           0.044794                          0.045835
        32                           0.033551                          0.034228
        64                           0.062304                          0.059111
```

Come si può notare nella riga "Out [3]", i valori 64 e 32 del numero di neuroni ottengono rispettivamente la prima e seconda posizione se si ordinano i risultati in base alla colonna *mean_test_balanced_accuracy*, mentre le prime due posizioni considerando la colonna *mean_test_accuracy* se le aggiudicano 16 e 64 "Out [4]". Si evidenzia quanto preannunciato, ovvero sembra che 64 sia un buon candidato numero di neuroni dato che si aggiudica la prima e seconda posizione in base alle due metriche scelte, tuttavia se si confrontano i risultati medi ottenuti dal valore 64 scopriamo che si trova in ultima posizione sia per quanto riguarda *mean_test_balanced_accuracy* sia *mean_test_accuracy*. Senza questa tecnica si sarebbero potute compiere scelte inappropriate optando per un aumento del numero di neuroni quando, in realtà, il test dice tutt'altro: ecco cosa significa visione globale.

4.7 Early Stopping

Non è stato adottato early stopping per la sua inadeguatezza se utilizzato con la cross-validation. In fase di splitting in k fold infatti un modello potrebbe essere addestrato per un numero di epoche diverso rispetto agli altri. Dato che i risultati sono delle medie, questi valori perderebbero di significato. La cross-validation è utile quando tutti i modelli usati nello splitting utilizzano gli stessi parametri.

4.8 Feature Selection

Sono stati effettuati dei piccoli test di eliminazione di feature con metodo ricorsivo: si sono rivelati particolarmente promettenti, tuttavia per i motivi che verranno elencati non si è proseguito oltre. La tecnica enunciata elimina ricorsivamente la feature di minor importanza e successivamente riaddestra il modello sul subset di features rimanenti. Sebbene sia una tecnica teoricamente semplice, in realtà comporta aspetti critici dal punto di vista della riproducibilità e del tuning. In una regressione lineare l'importanza delle features la si può ricavare dal valore dei pesi θ_i (in python attributo $coef\_$), analogamente modelli più complessi offrono lo stesso tipo di interfaccia o metodi simili. Il problema è che l'importanza delle features dipende fortemente dall'algoritmo e dai parametri utilizzati. Risulta quindi difficile poter utilizzare questa tecnica anche con un insieme relativamente piccolo di datasets. Esistono altre tecniche come l'eliminazione delle features che hanno una varianza minore di una certa soglia. Questo metodo non dipende in alcun modo dall'algoritmo e dai suoi parametri quindi sembrerebbe più facilmente utilizzabile, ma si è scelto di non proseguire a causa del fatto che le reti neurali tramite il dropout e gli alberi utilizzati da xgboost adottano intrinsecamente meccanismi simili all'eliminazione delle features.

4.9 Risultati

SIGLA	DESCRIZIONE
xgb	xgboost
svc	support vector classifier
lg	logistic regression
ln	linear regression
nn	neural network
rg	ridge classifier

TABELLA 4.3: Algoritmi

SIGLA	DESCRIZIONE
acc	accuratezza
pg	precisione classe positiva
rg	recupero classe positiva
fg	f1 classe positiva
pb	precisione classe negativa
rb	recupero classe negativa
fb	f1 classe negativa

TABELLA 4.4: Indici

- La dicitura "*precision_test_good*" è la precisione sul dataset di testing della classe positiva. La sua abbreviazione è "*pg*" come riportato sopra 4.4, analogamente per gli altri indici.
- I risultati seguenti sono divisi per dataset. Ogni tabella rappresenta il miglior risultato ottenuto da ciascun algoritmo rispetto alla metrica scelta. Le righe sono ordinate in ordine decrescente rispetto al miglior risultato.

Dataset: german
BEST accuracy_test

	id	acc	pg	rg	fg	pb	rb	fb
xgb	1971	0.787	0.807	0.914	0.857	0.710	0.489	0.579
svc	997	0.777	0.807	0.895	0.849	0.672	0.500	0.573
lg	907	0.777	0.799	0.910	0.851	0.689	0.467	0.556
ln	904	0.777	0.797	0.914	0.851	0.695	0.456	0.550
nn	1918	0.773	0.823	0.862	0.842	0.638	0.567	0.600
rg	1180	0.773	0.793	0.914	0.850	0.690	0.444	0.541

BEST precision_test_good

	id	acc	pg	rg	fg	pb	rb	fb
svc	1750	0.720	0.880	0.695	0.777	0.522	0.778	0.625
rg	1084	0.733	0.878	0.719	0.791	0.539	0.767	0.633
nn	1922	0.707	0.877	0.676	0.763	0.507	0.778	0.614
lg	1888	0.730	0.869	0.724	0.790	0.536	0.744	0.623
xgb	1971	0.787	0.807	0.914	0.857	0.710	0.489	0.579
ln	904	0.777	0.797	0.914	0.851	0.695	0.456	0.550

BEST recall_test_good

	id	acc	pg	rg	fg	pb	rb	fb
svc	1231	0.705	0.708	0.986	0.824	0.600	0.050	0.092
rg	913	0.763	0.764	0.957	0.850	0.757	0.311	0.441
xgb	916	0.767	0.776	0.938	0.849	0.717	0.367	0.485
ln	939	0.773	0.791	0.919	0.850	0.696	0.433	0.534
lg	907	0.777	0.799	0.910	0.851	0.689	0.467	0.556
nn	1919	0.763	0.806	0.871	0.838	0.630	0.511	0.564

BEST f_score_test_good

	id	acc	pg	rg	fg	pb	rb	fb
xgb	1971	0.787	0.807	0.914	0.857	0.710	0.489	0.579
ln	1087	0.777	0.797	0.914	0.851	0.695	0.456	0.550
lg	907	0.777	0.799	0.910	0.851	0.689	0.467	0.556
rg	1140	0.763	0.764	0.957	0.850	0.757	0.311	0.441
svc	1748	0.777	0.804	0.900	0.849	0.677	0.489	0.568
nn	1918	0.773	0.823	0.862	0.842	0.638	0.567	0.600

BEST precision_test_bad

	id	acc	pg	rg	fg	pb	rb	fb
rg	913	0.763	0.764	0.957	0.850	0.757	0.311	0.441
xgb	1775	0.773	0.784	0.933	0.852	0.720	0.400	0.514
ln	939	0.773	0.791	0.919	0.850	0.696	0.433	0.534
lg	907	0.777	0.799	0.910	0.851	0.689	0.467	0.556
svc	1748	0.777	0.804	0.900	0.849	0.677	0.489	0.568
nn	1918	0.773	0.823	0.862	0.842	0.638	0.567	0.600

BEST recall_test_bad

	id	acc	pg	rg	fg	pb	rb	fb
svc	1750	0.720	0.880	0.695	0.777	0.522	0.778	0.625
nn	1922	0.707	0.877	0.676	0.763	0.507	0.778	0.614
rg	1084	0.733	0.878	0.719	0.791	0.539	0.767	0.633
lg	1081	0.727	0.868	0.719	0.786	0.532	0.744	0.620
xgb	1307	0.780	0.805	0.905	0.852	0.688	0.489	0.571
ln	904	0.777	0.797	0.914	0.851	0.695	0.456	0.550

BEST f_score_test_bad

	id	acc	pg	rg	fg	pb	rb	fb
nn	1926	0.750	0.869	0.757	0.809	0.564	0.733	0.638
rg	1084	0.733	0.878	0.719	0.791	0.539	0.767	0.633
svc	1750	0.720	0.880	0.695	0.777	0.522	0.778	0.625
lg	1888	0.730	0.869	0.724	0.790	0.536	0.744	0.623
xgb	1971	0.787	0.807	0.914	0.857	0.710	0.489	0.579
ln	904	0.777	0.797	0.914	0.851	0.695	0.456	0.550

```
Dataset: australian
BEST accuracy_test
```

	id	acc	pg	rg	fg	pb	rb	fb
nn	1930	0.913	0.929	0.913	0.921	0.894	0.913	0.903
xgb	1873	0.884	0.910	0.878	0.894	0.854	0.891	0.872
lg	1889	0.879	0.895	0.887	0.891	0.860	0.870	0.865
svc	998	0.874	0.932	0.835	0.881	0.817	0.924	0.867
rg	1901	0.874	0.850	0.939	0.893	0.912	0.793	0.849
ln	1872	0.874	0.868	0.913	0.890	0.884	0.826	0.854

```
BEST precision_test_good
```

	id	acc	pg	rg	fg	pb	rb	fb
ln	905	0.874	0.949	0.817	0.879	0.806	0.946	0.870
svc	1520	0.845	0.946	0.765	0.846	0.763	0.946	0.845
rg	1560	0.841	0.946	0.757	0.841	0.757	0.946	0.841
lg	1441	0.841	0.936	0.765	0.842	0.761	0.935	0.839
xgb	1606	0.831	0.935	0.748	0.831	0.748	0.935	0.831
nn	1941	0.860	0.930	0.809	0.865	0.794	0.924	0.854

```
BEST recall_test_good
```

	id	acc	pg	rg	fg	pb	rb	fb
svc	1557	0.638	0.605	1.000	0.754	1.000	0.185	0.312
rg	1588	0.647	0.612	1.000	0.759	1.000	0.207	0.342
nn	1930	0.913	0.929	0.913	0.921	0.894	0.913	0.903
ln	1871	0.865	0.854	0.913	0.882	0.881	0.804	0.841
lg	1889	0.879	0.895	0.887	0.891	0.860	0.870	0.865
xgb	1873	0.884	0.910	0.878	0.894	0.854	0.891	0.872

```
BEST f_score_test_good
```

	id	acc	pg	rg	fg	pb	rb	fb
nn	1930	0.913	0.929	0.913	0.921	0.894	0.913	0.903
xgb	1873	0.884	0.910	0.878	0.894	0.854	0.891	0.872
rg	1901	0.874	0.850	0.939	0.893	0.912	0.793	0.849
lg	1889	0.879	0.895	0.887	0.891	0.860	0.870	0.865
ln	1002	0.874	0.868	0.913	0.890	0.884	0.826	0.854
svc	998	0.874	0.932	0.835	0.881	0.817	0.924	0.867

BEST precision_test_bad

	id	acc	pg	rg	fg	pb	rb	fb
svc	1557	0.638	0.605	1.000	0.754	1.000	0.185	0.312
rg	1588	0.647	0.612	1.000	0.759	1.000	0.207	0.342
nn	1930	0.913	0.929	0.913	0.921	0.894	0.913	0.903
ln	1002	0.874	0.868	0.913	0.890	0.884	0.826	0.854
lg	1889	0.879	0.895	0.887	0.891	0.860	0.870	0.865
xgb	1873	0.884	0.910	0.878	0.894	0.854	0.891	0.872

BEST recall_test_bad

	id	acc	pg	rg	fg	pb	rb	fb
svc	1520	0.845	0.946	0.765	0.846	0.763	0.946	0.845
rg	1560	0.841	0.946	0.757	0.841	0.757	0.946	0.841
ln	892	0.841	0.946	0.757	0.841	0.757	0.946	0.841
xgb	1606	0.831	0.935	0.748	0.831	0.748	0.935	0.831
lg	1441	0.841	0.936	0.765	0.842	0.761	0.935	0.839
nn	1941	0.860	0.930	0.809	0.865	0.794	0.924	0.854

BEST f_score_test_bad

	id	acc	pg	rg	fg	pb	rb	fb
nn	1930	0.913	0.929	0.913	0.921	0.894	0.913	0.903
xgb	1873	0.884	0.910	0.878	0.894	0.854	0.891	0.872
ln	905	0.874	0.949	0.817	0.879	0.806	0.946	0.870
svc	998	0.874	0.932	0.835	0.881	0.817	0.924	0.867
lg	909	0.879	0.902	0.878	0.890	0.853	0.880	0.866
rg	1901	0.874	0.850	0.939	0.893	0.912	0.793	0.849

```
Dataset: kaggle
BEST accuracy_test
```

	id	acc	pg	rg	fg	pb	rb	fb
nn	1957	0.936	0.945	0.989	0.966	0.558	0.195	0.289
xgb	1857	0.936	0.941	0.994	0.967	0.592	0.129	0.212
lg	1830	0.934	0.936	0.998	0.966	0.588	0.044	0.083
ln	1827	0.934	0.934	0.999	0.966	0.569	0.022	0.043
rg	1846	0.934	0.934	0.999	0.966	0.569	0.022	0.042
svc	1834	0.888	0.936	0.944	0.940	0.115	0.102	0.108

```
BEST precision_test_good
```

	id	acc	pg	rg	fg	pb	rb	fb
lg	1831	0.809	0.970	0.821	0.889	0.205	0.641	0.310
svc	1845	0.841	0.970	0.857	0.910	0.238	0.625	0.345
rg	1848	0.648	0.961	0.650	0.775	0.115	0.633	0.194
xgb	1852	0.932	0.945	0.984	0.964	0.475	0.206	0.287
nn	1957	0.936	0.945	0.989	0.966	0.558	0.195	0.289
ln	1828	0.934	0.934	0.999	0.966	0.569	0.022	0.043

```
BEST recall_test_good
```

	id	acc	pg	rg	fg	pb	rb	fb
ln	1875	0.933	0.934	0.999	0.966	0.563	0.021	0.041
rg	1846	0.934	0.934	0.999	0.966	0.569	0.022	0.042
lg	1830	0.934	0.936	0.998	0.966	0.588	0.044	0.083
xgb	1857	0.936	0.941	0.994	0.967	0.592	0.129	0.212
nn	1957	0.936	0.945	0.989	0.966	0.558	0.195	0.289
svc	1834	0.888	0.936	0.944	0.940	0.115	0.102	0.108

```
BEST f_score_test_good
```

	id	acc	pg	rg	fg	pb	rb	fb
xgb	1857	0.936	0.941	0.994	0.967	0.592	0.129	0.212
nn	1957	0.936	0.945	0.989	0.966	0.558	0.195	0.289
lg	1830	0.934	0.936	0.998	0.966	0.588	0.044	0.083
ln	1827	0.934	0.934	0.999	0.966	0.569	0.022	0.043
rg	1846	0.934	0.934	0.999	0.966	0.569	0.022	0.042
svc	1834	0.888	0.936	0.944	0.940	0.115	0.102	0.108

```
BEST precision_test_bad
```

	id	acc	pg	rg	fg	pb	rb	fb
xgb	1857	0.936	0.941	0.994	0.967	0.592	0.129	0.212
lg	1830	0.934	0.936	0.998	0.966	0.588	0.044	0.083
ln	1827	0.934	0.934	0.999	0.966	0.569	0.022	0.043
rg	1846	0.934	0.934	0.999	0.966	0.569	0.022	0.042
nn	1957	0.936	0.945	0.989	0.966	0.558	0.195	0.289
svc	1835	0.858	0.968	0.877	0.920	0.259	0.601	0.362

```
BEST recall_test_bad
```

	id	acc	pg	rg	fg	pb	rb	fb
lg	1052	0.779	0.969	0.788	0.869	0.179	0.644	0.280
rg	1120	0.634	0.960	0.634	0.764	0.111	0.636	0.189
svc	1845	0.841	0.970	0.857	0.910	0.238	0.625	0.345
xgb	1852	0.932	0.945	0.984	0.964	0.475	0.206	0.287
nn	1957	0.936	0.945	0.989	0.966	0.558	0.195	0.289
ln	1828	0.934	0.934	0.999	0.966	0.569	0.022	0.043

```
BEST f_score_test_bad
```

	id	acc	pg	rg	fg	pb	rb	fb
svc	1835	0.858	0.968	0.877	0.920	0.259	0.601	0.362
lg	1833	0.815	0.969	0.827	0.893	0.209	0.635	0.314
nn	1957	0.936	0.945	0.989	0.966	0.558	0.195	0.289
xgb	1852	0.932	0.945	0.984	0.964	0.475	0.206	0.287
rg	1851	0.652	0.961	0.654	0.778	0.115	0.631	0.195
ln	1828	0.934	0.934	0.999	0.966	0.569	0.022	0.043

```
Dataset: pak
BEST accuracy_test


       id    acc     pg     rg     fg     pb     rb     fb
xgb  1038  0.740  0.742  0.995  0.850  0.531  0.017  0.034
ln   1785  0.740  0.740  0.999  0.850  0.625  0.004  0.008
rg   1809  0.739  0.740  0.999  0.850  0.531  0.004  0.009
lg   1019  0.739  0.741  0.993  0.849  0.480  0.018  0.035
nn   1951  0.722  0.750  0.936  0.833  0.392  0.117  0.180
svc  1805  0.570  0.822  0.534  0.647  0.337  0.671  0.449


BEST precision_test_good


       id    acc     pg     rg     fg     pb     rb     fb
svc  1805  0.570  0.822  0.534  0.647  0.337  0.671  0.449
rg   1810  0.590  0.817  0.575  0.675  0.345  0.635  0.447
lg   1786  0.592  0.816  0.578  0.677  0.345  0.631  0.446
nn   1950  0.639  0.760  0.748  0.754  0.316  0.329  0.322
xgb  1818  0.738  0.743  0.987  0.848  0.465  0.031  0.058
ln   1877  0.739  0.740  0.999  0.850  0.529  0.005  0.009


BEST recall_test_good


       id    acc     pg     rg     fg     pb     rb     fb
xgb  1043  0.739  0.739  1.000  0.850  1.000  0.000  0.001
ln   1785  0.740  0.740  0.999  0.850  0.625  0.004  0.008
rg   1031  0.739  0.739  0.999  0.850  0.389  0.002  0.004
lg   1788  0.739  0.741  0.994  0.849  0.472  0.015  0.030
nn   1949  0.719  0.747  0.936  0.831  0.360  0.102  0.159
svc  1805  0.570  0.822  0.534  0.647  0.337  0.671  0.449


BEST f_score_test_good


       id    acc     pg     rg     fg     pb     rb     fb
xgb  1823  0.740  0.740  0.999  0.850  0.625  0.004  0.008
ln   1785  0.740  0.740  0.999  0.850  0.625  0.004  0.008
rg   1809  0.739  0.740  0.999  0.850  0.531  0.004  0.009
lg   1788  0.739  0.741  0.994  0.849  0.472  0.015  0.030
nn   1951  0.722  0.750  0.936  0.833  0.392  0.117  0.180
svc  1805  0.570  0.822  0.534  0.647  0.337  0.671  0.449
```

```
BEST precision_test_bad


       id    acc     pg     rg     fg     pb     rb     fb
xgb  1043  0.739  0.739  1.000  0.850  1.000  0.000  0.001
ln   1785  0.740  0.740  0.999  0.850  0.625  0.004  0.008
rg   1809  0.739  0.740  0.999  0.850  0.531  0.004  0.009
lg   1019  0.739  0.741  0.993  0.849  0.480  0.018  0.035
nn   1951  0.722  0.750  0.936  0.833  0.392  0.117  0.180
svc  1805  0.570  0.822  0.534  0.647  0.337  0.671  0.449


BEST recall_test_bad


       id    acc     pg     rg     fg     pb     rb     fb
svc  1805  0.570  0.822  0.534  0.647  0.337  0.671  0.449
rg   1810  0.590  0.817  0.575  0.675  0.345  0.635  0.447
lg   1786  0.592  0.816  0.578  0.677  0.345  0.631  0.446
nn   1950  0.639  0.760  0.748  0.754  0.316  0.329  0.322
xgb  1818  0.738  0.743  0.987  0.848  0.465  0.031  0.058
ln   1877  0.739  0.740  0.999  0.850  0.529  0.005  0.009


BEST f_score_test_bad


       id    acc     pg     rg     fg     pb     rb     fb
svc  1805  0.570  0.822  0.534  0.647  0.337  0.671  0.449
rg   1810  0.590  0.817  0.575  0.675  0.345  0.635  0.447
lg   1786  0.592  0.816  0.578  0.677  0.345  0.631  0.446
nn   1950  0.639  0.760  0.748  0.754  0.316  0.329  0.322
xgb  1818  0.738  0.743  0.987  0.848  0.465  0.031  0.058
ln   1877  0.739  0.740  0.999  0.850  0.529  0.005  0.009
```

4.10 Interpretazione dei Risultati

Nessun algoritmo si è particolarmente distinto rispetto agli altri. Algoritmi più complessi come xgboost e le reti neurali ottengono globalmente accuratezze migliori, tuttavia questo potrebbe portarci fuori strada. Non è difficile ottenere accuratezze elevate in presenza di forte sbilanciamento. Per esempio, nel caso dei datasets kaggle o pak un algoritmo che genera in output predizioni della sola classe positiva ottiene un'accuratezza sopra il 90%. Come vedremo è meglio concentrarsi su indici che premiano il bilanciameto piuttosto che l'accuratezza. Per favorire il bilanciamento è bene utilizzare attributi del modello che assegnano un peso alle varie classi come ad esempio "*class_weight*". In genere diversi modelli del package scikit-learn offrono questo tipo di attributi o qualcosa di analogo. È stato possibile riscontrare che il loro utilizzo di norma porta un notevole miglioramento dal punto di vista del bilanciamento, tuttavia viene quasi sempre penalizzata l'accuratezza. Questo non deve spaventarci perché è essenziale disporre di un algoritmo bilanciato. Un altro aspetto che è emerso è la difficoltà di manipolare le reti neurali per favorire il bilanciamento. Mentre gli altri modelli forniscono attributi allo scopo, le reti neurali soffrono di questa mancanza e si rivelano particolarmente sofferenti allo sbilanciamento. La linear regression si è rilevata inadatta soprattutto in presenza di sbilanciamento. Durante il tuning dei datasets con meno record si sono evidenziati alcuni comportamenti indesiderati: i risultati della cross-validation e i risultati sul test set erano tutt'altro che simili. Nello specifico questo comportamento è più evidente nel dataset australiano dove risultati medi dell'88% di accuratezza con una bassa deviazione standard, producevano risultati nettamente inferiori nel test set vero e proprio (si parla di uno scarto a volte superiore ai 5 punti percentuali). È chiaro che disporre di risultati medi di una certa robustezza con una bassa deviazione standard poi non garantisce effettivamente che questi risultati siano rispettati anche sul test set, ci dà una buona probabilità che lo siano, ma non la certezza. È stato possibile riscontrare che questa anomalia colpiva tutti i diversi classificatori. A questo punto sembrano esserci troppe

coincidenze per essere ignorate. Una possibile spiegazione potrebbe essere che con datasets di piccole dimensioni sia molto influente come viene scelto il test set. Con meno istanze ogni singolo errore ha un peso influente sul risultato percentuale. Con datasets numerosi questo comportamento non dovrebbe verificarsi. Una soluzione plausibile è mischiare il dataset australiano, ripetere la suddivisione in training e test e poi verificare l'output dei modelli. A questo punto dello studio non è stato possibile verificarlo.

4.11 Risultati Letteratura

Le successive tabelle contengono un confronto dei risultati ottenuti in letteratura e i risultati ottenuti in questo studio. La prima tabella effettua un confronto dell'accuratezza mentre la seconda un confronto su f1 con gli studi riportati in bibliografia.

studio/dataset	AC	GC	PAK	KAGGLE
Lessman 2015 [1]	0.868	0.761	0.682	0.925
questo studio	0.913	0.787	0.740	0.936
Tsai 2008 [2]	0.973	0.834	x	x
Xia 2017 [5]	0.883	0.783	x	x
Munkhdalai 2018 [7]	0.868	0.771	x	x
Ha 2016 [8]	0.862	0.747	x	x
Ala'raj 2016 [9]	0.880	0.777	x	x
Xia 2017 [10]	0.879	0.773	x	x

TABELLA 4.5: Confronto Accuratezza

studio/dataset	AC	GC	PAK	KAGGLE
He 2018 [4]	0.854	0.850	x	x
questo studio	0.921	0.853	0.850	0.967

TABELLA 4.6: Confronto F1 Classe Positiva

Solo uno studio forniva F1 ecco perché la tabella contiene una sola riga

Per quanto riguarda l'accuratezza i risultati ottenuti in questo studio sono stati superati solo dallo studio [2] pubblicato nell'anno 2008. Lo studio utilizza le reti neurali e ensemble di reti neurali. Il fatto che uno studio non recentissimo abbia surclassato i risultati di tutti gli studi più recenti consultati che utilizzano tecniche ed algoritmi all'epoca non disponibili è motivo di stupore, tuttavia è doveroso fare alcune considerazioni al riguardo: prima di tutto non è chiaro se sia stata utilizzata la cross-validation per ottenere risultati robusti, mancano aspetti importanti riguardo la fase di tuning ecc... C'è da sottolineare che con datasets di piccole dimensioni i risultati dei test sono facilmente influenzabili da fattori non derivanti dalla bontà del modello. Inoltre riportare dati di accuratezza senza sapere quale sia la varianza e la media non permette di verificare quale sia la robustezza dei risultati. Oltretutto cosa si può dire sulla fase di apprendimento della rete? Siamo realmente a conoscenza su quali fattori si basi il suo apprendimento? Qualora fosse estremamente importante capire su quali aspetti principali si basi l'output di una rete sarebbe particolarmente difficoltoso risalirne. Nel caso di riconoscimento di tumori o altre patologie scoprire quali feature sono maggiormente rilevanti costituisce un obiettivo essenziale. A seguire un altro esempio: in un problema di riconoscimento di pietanze/ingredienti attraverso le immagini possono verificarsi aspetti indesiderati come il fatto che il modello si basi su oggetti al di fuori del piatto per il riconoscimento dello stesso come oggetti da cucina. Chiaramente un modello del genere non è utilizzabile. In questo studio si sono ottenuti buoni risultati anche con algoritmi più semplici che non soffrono di queste problematiche, anzi nella maggior parte dei casi (vedi tabelle a inizio capitolo) le reti neurali non occupano la prima posizione.

4.12 Riproducibilità

Per ottenere risultati riproducibili sono stati utilizzati salvataggi su file in aggiunta a salvataggi su database, tuttavia sono emersi diversi problemi che ora elencheremo. Definire gli opportuni seed può non essere sufficiente, in particolar modo quando si utilizzano le reti neurali. È emerso che il parallelismo costituisce un ostacolo all'ottenimento di dati riproducibili.[5] Risulta opportuno quindi salvare pesi e configurazione dei modelli. Un altro aspetto rilevante è che il parallelismo di Tensorflow e della Random/GridSearchCv vanno in conflitto se utilizzati in contemporanea: consiglio di configuare la sessione di tensorflow seguendo le istruzioni precedentemente riportate ed in aggiunta di consultare questo materiale.[6]

[5] dati riproducibili in keras.
[6] multiprocessing

Capitolo 5

Conclusioni

È difficile decretare un modello vincitore in quanto 4 datasets non sono sufficienti ad elargire una sentenza definitiva, tuttavia si possono trarre diverse considerazioni. Modelli complessi come le reti neurali ed ensemble non hanno surclassato le performance dei modelli più semplici. Da tenere in considerazione che l'addestramento e il tuning delle reti comporta una notevole quantità di lavoro sia per l'uomo che per la macchina (non a caso è stata definita una "black art"), perciò si consiglia il loro utilizzo se i risultati finora ottenuti non risultino soddisfacenti; inoltre i modelli più semplici soffrono meno del fenomeno overfitting. Quando non è possibile risolverlo non si deve commettere l'errore di pensare che il nostro modello sia da scartare: ogni caso va valutato attentamente. In generale l'overfitting è tollerabile in alcune condizioni per esempio se i risultati della cross-validation mostrano una deviazione standard bassa e dei buoni risultati medi. In questo tipo di problemi si consiglia di dimenticarsi di aspetti come l'accuratezza e focalizzarsi su indici più adatti al caso. Risulta molto importante la scelta della funzione utilizzata per decretare il modello migliore, la cosa migliore è creare una funzione apposita definita in collaborazione con l'istituto di credito che commissiona il progetto, nelle sezioni successive verrà trattato in maggiore dettaglio questo aspetto.

5.1 Dataset con Elevato Numero di Istanze

Con datasets di notevoli dimensioni, qualora non si disponga di risorse a sufficienza o qualora risulti troppo lunga la fase di addestramento, si consiglia di ricavare una piccola parte del dataset e destinarlo alla fase di training: al posto del 70% utilizzare un 10% per il dataset kaggle. In questo caso il numero di istanze utilizzate in fase di training ammonta a 15 000 (contro 105 000 suddivisione 70/30), che si sono dimostrate più che sufficienti. Successivamente quando il modello è pronto per essere testato si utilizzano i parametri del modello più promettente sulla suddivisione originaria 70/30. Tutto questo viene fatto per tagliare i tempi di addestramento, ma bisogna fare alcune precisazioni: non è detto che i risultati ottenuti sul 10% siano poi mantenuti. Per ovviare o limitare il più possibile questo inconveniente è opportuno mischiare e poi ricavare il 10% mantenendo la stessa distribuzione delle classi positive e negative del dataset originario. Se così non fosse il modello sarà più soggetto a comportamenti indesiderati dato che è stato addestrato e testato su datasets con diverso bilanciamento. Prestare particolare attenzione a come viene ricavato il 10% per non incorrere in strane anomalie dovute ad esempio ad intersezioni con il 30% del test set ricavato in seguito.

5.2 Consigli sul Confronto dei Risultati

Per quanto riguarda gli indici di accuratezza, si consiglia di non prendere il numero puro come riferimento, non c'è nulla di più sbagliato. Risulta doveroso verificare da quale circostanze è stato partorito tale risultato: è stata utilizzata una CV? se sì qual è il valore di k? la dimensione del dataset, i seed ecc... Si sconsiglia quindi la comparazione morbosa dei risultati tra uno studio ed un altro perché un 78% di accuratezza dello studio 1 può risultare migliore dell'80% ottenuto dallo studio 2.

5.3 Definire la Propria Funzione

Fin da subito nasce la necessità di individuare gli indici opportuni che conferiscono una visione globale sulla validità del nostro algoritmo. Questa operazione risulta tutt'altro che semplice anzi, c'è stato un acceso dibattito in passato e sono state fatte diverse critiche e studi a tal proposito. La domanda è sempre la stessa: quale modello è migliore? Come annunciato nella sezione precedente non basta confrontare il risultato nudo e crudo nemmeno nella migliore delle ipotesi dove vengono utilizzati gli stessi dati con la stessa fase di preprocessing, gli stessi seed, la stessa CV... insomma nelle stesse condizioni di un altro studio con cui ci vogliamo confrontare. In questo caso il confronto tra risultati sarebbe sensato, tuttavia non ci porta a rispondere correttamente alla domanda che ci siamo posti. Ciò che possiamo dire è che il nostro modello raggiunge risultati superiori in accuratezza, il nostro modello è superiore in recall ecc... Ma la domanda è un'altra: qual è il migliore? Ebbene, è necessario ragionare in termini bancari per rispondere a questa domanda. Ciò che interessa alla banca è il profitto, in questo e diversi altri studi non vengono utilizzati questo tipo di indicatori, causa la loro difficile reperibilità. Attenzione non sto dicendo che gli indici utilizzati siano di scarsa importanza, sto semplicemente dicendo che essi potrebbero passare in secondo piano di fronte ad indicatori di profitto. Le banche ricavano interessi dai prestiti concessi ma conseguono delle perdite nel caso di insolvenza del cliente. È quindi possibile formulare una banale funzione di profitto che chiameremo P definita nel seguente modo:

$$P = \frac{p_1 \cdot TP - c_1 \cdot FN - c_2 \cdot FP}{p_1 \cdot totPos}$$

dove p_1 è un coefficiente di profitto, c_1 e c_2 sono coefficienti di costo, per TP, FN, FP vedi Matrice di confusione, totPos è il totale dei casi positivi. Questa funzione deve essere massimizzata. Nel caso in cui alcuni algoritmi non permettano la massimizzazione è possibile cambiare di segno la funzione e procedere con la sua

minimizzazione come se fosse una normale funzione di loss.

5.4 Lavori Futuri

- Verificare se esiste un modello con una particolare configurazione che ottiene risultati discreti su diversi datasets.
- Utilizzare tecniche di feature selection per cercare di migliorare i risultati ottenuti.
- Provare tecniche di ensemble avanzate.
- Ripetere il tuning utilizzando una funzione di profitto definita in collaborazione con istituti di credito come quella definita nella sezione 5.3
- Sperimentare modelli e tecniche non utilizzati in questo studio.

Ringraziamenti

- Si ringrazia Prof. Dr. Stefan Lessmann per i brillanti studi effettuati e per aver contribuito alla redazione di questa tesi fornendo materiale ausiliario.

- Si ringrazia infinitamente Prof. Dr. Gianluca Moro per la disponibilità ed il supporto fornito.

Bibliografia

[1] Stefan Lessmann, Bart Baesens, Hsin-Vonn Seow, Lyn C. Thomas. *Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research* .

[2] Chih-Fong Tsai, Jhen-Wei Wu. *Using neural network ensembles for bankruptcy prediction and credit scoring*.

[3] Aurélien Géron. *Hands-On Machine Learning With Scikit-Learn and Tensorflow*.

[4] Hongliang He, Wenyu Zhang, Shuai Zhang. *A novel ensemble method for credit scoring: Adaption of different imbalance ratios*.

[5] Yufei Xia, Chuanzhe Liu, Bowen Da, Fangming Xie. *A novel heterogeneous ensemble credit scoring model based on bstacking approach*.

[6] Arindam Chaudhuri, Soumya K. Ghosh. *Bankruptcy Prediction through Soft Computing based Deep Learning Technique*.

[7] Lkhagvadorj Munkhdalai, Oyun-Erdene Namsrai, Keun Ho Ryu1. *Credit Scoring With Deep Learning*

[8] Van-Sang Ha, Ha-Nam Nguyen. *Credit scoring with a feature selection approach based deep learning*

[9] Maher Alaraj, Maysam F Abbod. *Classifiers consensus system approach for credit scoring*

[10] Yufei Xia, Chuanzhe Liu, YuYing Li, Nana Liu. *A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring*

[11] Chi Guotai , Mohammad Zoynul Abedin, Fahmida–E–Moula. *Modeling credit approval data with neural networks: an experimental investigation and optimization*

[12] *Libreria keras*

[13] *Libreria tensorflow*

[14] *Libreria scikit-learn*

[15] *Libreria xgboost*

[16] Gianluca Moro, Roberto Pasolini. *Slide corso data intensive*

[17] Jason Brownlee. *Understand the Impact of Learning Rate on Model Performance With Deep Learning Neural Networks*

[18] Tarang Shah. *About Train, Validation and Test Sets in Machine Learning*

[19] Jason Brownlee. *How to Create a Random-Split, Cross-Validation, and Bagging Ensemble for Deep Learning in Keras*

[20] Aarshay Jain. *Complete Guide to Parameter Tuning in XGBoost (with codes in Python)*

[21] Jason Brownlee. *How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras*

[22] Giuseppe Gullo. *Come è possibile valutare l'accuratezza di uno stimatore basato su Neural Networks?*

[23] Mohtadi Ben Fraj. *In Depth: Parameter tuning for Gradient Boosting*