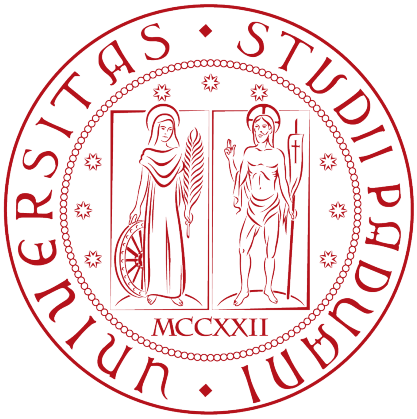
Università degli studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Magistrale in

Scienze Statistiche



Tesi di laurea

Modelli statistici per la previsione della popolarità di contenuti Social Media: analisi su dati reali relativi a profili Facebook

Relatore Prof. Livio Finos
Dipartimento di Psicologia dello
Sviluppo e della Socializzazione - DPSS
Correlatore Dott. Dario Solari

Laureando Ciro Lista
Matricola 1132870

Anno Accademico 2016/2017

Alla mia famiglia

Indice

Introduzione

Negli ultimi anni la diffusione di Social Media come Facebook, Instagram e Twitter ha completamente rivoluzionato il modo di interagire delle persone e ha contribuito in maniera significativa al fenomeno della globalizzazione. Inoltre, in questo mondo così connesso, gli individui cercano sempre più di acquisire popolarità attraverso la pubblicazione continua di contenuti, soprattutto di immagini. Ma quali sono le caratteristiche che rendono una foto virale sui Social Media? Capire se esistano delle componenti che rendano apprezzata una foto è di notevole interesse da un punto di vista scientifico, ma anche da un punto di vista del marketing questo potrebbe avere tante applicazioni utili.

Lo scopo di questa tesi è quindi quello di prevedere la popolarità di contenuti Social Media utilizzando diversi modelli statistici. In particolare, i dati di riferimento sono relativi ai profili Facebook di un campione di 240 studenti dell'Università degli Studi di Padova. Di tali profili, oltre alle foto pubblicate sono disponibili alcune informazioni demografiche ed altre relative alla rete di amici dell'utente. La letteratura riguardante quello che di fatto è un problema di statistica sociale è in realtà molto recente, e va di pari passo con gli straordinari sviluppi ottenuti negli ultimi anni nel campo della *computer vision*. Infatti, la possibilità di processare in maniera sempre più dettagliata un'immagine ha ispirato diversi lavori, che avevano sì obiettivi diversi ma che da un punto di vista metodologico hanno utilizzato di fatto gli stessi strumenti. L'anno zero di quest'area di ricerca è databile al 2014, dove sono stati pubblicati ben tre lavori. Quelli probabilmente più rilevanti e sicuramente più vicini a quello che è l'obiettivo di questa tesi sono quelli di Khosla et al., (2014) e McParlane et al., (2014). Questi lavori volevano sostanzialmente individuare quali fossero le $features$ che contribuissero maggiormente al numero di visualizzazioni e al numero di commenti ricevuti da un'immagine postata sulla community Flickr. In particolare, il primo lavoro si è fermato solo al numero di visualizzazioni mentre il secondo utilizzava come indicatore della popolarità di un'immagine anche il numero di commenti ricevuti da altri utenti, proprio come in questa tesi. L'ultimo contributo significativo è stato dato da Totti et al., (2014), il cui *task* era quello di individuare i *pattern* che favorissero la diffusione di una foto sul web, per cui la variabile di risposta era rappresentata proprio dal numero di condivisioni di un'immagine.

Ora, ciò che vale la pena sottolineare e che rappresenta un elemento di innovazione rispetto ai lavori precedenti, è che tra i dati a disposizione oltre alle informazioni descritte in precedenza sono disponibili anche i profili psicologici degli utenti. Di interesse sarà quindi valutare se sog-

getti con particolari tratti caratteriali sono anche più popolari sui Social Media.
Nel primo capitolo si descrivono i modelli statistici utilizzati per prevedere la variabile di risposta, rappresentata dal numero di commenti ricevuti da una foto escludendo quelli effettuati dall'utente stesso.
Nel secondo capitolo si approfondiscono le reti neurali, e la motivazione di questo sta nel fatto che oltre ad averlo utilizzato come modello previsivo per la variabile di risposta, questi modelli sono stati largamente impiegati nella costruzione del dataset. Come si vedrà, si è fatto riferimento a reti allenate per l'identificazione del contenuto dell'immagine.
Nel terzo capitolo ci si sofferma sulla costruzione del dataset e in particolar modo delle variabili esplicative. Dal momento che buona parte di queste sono state definite non solo ispirandosi alla letteratura ma anche in base a considerazioni di natura personale, è importante tenere a mente come si è deciso di operativizzare i diversi aspetti che descrivono l'immagine.
Nel quarto capitolo si passa alla fase di analisi, divisibile in due parti. In particolare, nella prima ci si è concentrati soprattutto sul lato descrittivo e si cerca di scomporre il problema cercando di estrarre delle prime indicazioni sul ruolo delle diverse $features$. Successivamente, nella seconda parte si passa al *tuning* dei parametri dei modelli utilizzati e ad un confronto delle loro performance.
Infine, al quinto capitolo sono dedicati i risultati ottenuti e una breve discussione relativa ai punti critici della metodologia utilizzata.

Capitolo 1

Modelli di Previsione

Come accennato nell'introduzione l'obiettivo di questa tesi è quello di stimare diversi modelli allo scopo di prevedere in maniera più accurata possibile sia la presenza/assenza di almeno un commento che il numero di commenti stessi. Da un punto di vista statistico si tratta quindi di affrontare da un lato un problema di classificazione binaria e dall'altro un problema di regressione 0-inflated. L'alta dimensionalità del dataset, resa tale non solo dal numero di unità statistiche ma anche da quello delle variabili in gioco, incoraggia all'utilizzo di modelli che oltre a fornire delle buone performance in termini di previsione siano anche efficienti da un punto di vista computazionale. Tra quelli che rispondono a queste esigenze e che attualmente rappresentano uno status quo nella *Data Science*, sono stati scelti i seguenti:

- Bagging (con alberi)
- Random Forest
- Gradient Boosting
- XGBoosting
- Light Gradient Boosting
- SVM/SVR
- Multi-layer neural network
- Regressione Ridge (come benchmark)

1.0.1 Bagging

L'idea generale del bagging è quella di combinare tanti modelli caratterizzati da molta instabilità e poca distorsione allo scopo di ottenere un modello con varianza minore. Il concetto che sta alla base del bagging è che il valore atteso di una media di variabili aleatorie $i.d.$ è uguale al valore atteso della singola variabile aleatoria, per cui l'obiettivo è far diminuire la varianza sfruttando quest'operazione di media. Tipicamente il candidato migliore per questo approccio è l'albero di classificazione perché a dispetto della sua semplicità riesce a cogliere molto bene le relazioni non lineari e le interazioni tra le variabili, e quindi è stato utilizzato anche per questa tesi. Il parametro di regolazione del bagging è rappresentato da B, il numero di alberi; tuttavia, studi di simulazione ed evidenze empiriche hanno mostrato che le performance di tale modello non sono molto sensibili alla scelta del parametro di regolazione, per cui è sufficiente scegliere un numero di alberi sufficientemente elevato per ottenere performance soddisfacenti. In questo lavoro il valore di B è stato considerato pari a 1000.

1.0.2 Random Forest

Le Random Forest, che rappresentano uno dei modelli fondati sul concetto dell'*ensamble* più famosi, hanno come obiettivo quello di ridurre la varianza del modello agendo sulla correlazione tra gli alberi. Questo ha senso se pensiamo che data una successione di variabili aleatorie $i.d$, la varianza della media di queste è pari a $\rho\sigma^2 + (1-\rho)\sigma^2$, dove ρ è la correlazione tra coppie di variabili aleatorie. Questo da un punto di vista pratico significa che se ρ è relativamente grande i vantaggi del fare bagging possono ridursi di molto. Le Random Forest tentano di risolvere questo problema selezionando ad ogni split di un albero un numero di variabili pari a $m \ll p$, con l'idea che in questo modo i singoli alberi saranno più diversi tra loro e quindi meno correlati. Il parametro di regolazione della Random Forest è rappresentato quindi da m, che se troppo piccolo porterà delle performance non buone soprattutto laddove il rapporto segnale rumore non è molto alto. D'altra parte, come nel caso del bagging, il numero di alberi non rappresenta un parametro di regolazione rilevante e per questo è stato posto fisso pari a 300.

1.0.3 Gradient Boosting

L'idea che sta alla base del boosting nella sua prima versione AdaBoost è quella di costruire un buon classificatore a partire da una sequenza di classificatori deboli (tipicamente stumps o comunque alberi caratterizzati da un numero di split d piccolo). Quindi, sotto questo punto di vista il boosting ha abbastanza a che vedere con il Bagging e la Random Forest. Tuttavia, se escludiamo questo aspetto, esso è molto diverso dagli altri due metodi in quanto tale successione di classificatori deboli viene costruita in maniera adattiva. In particolare, quello che fa è modificare la probabilità di estrazione di un soggetto ad ogni singola iterazione assegnando una probabilità maggiore a quelli mal classificati nel passo precedente. L'idea è che in questo modo ci si concentra maggiormente su quel sottinsieme di dati che non riusciamo a classificare correttamente. Questo significa che dal punto di vista del trade-off tra varianza e distorsione il numero di alberi che si considera rappresenta un parametro di regolazione importante perché nel caso in cui questo sia troppo alto c'è il rischio che il modello si sovradatti ai dati.
Successivamente sono state proposte nuove versioni di questo metodo, tutte basate sulla discesa del gradiente, per cui l'idea è scegliere lo split che massimizza la discesa del gradiente e che quindi favorisce l'avvicinamento al punto di minimo della funzione obiettivo.
In particolare, se si adotta la tipica funzione di perdita quadratica, il Gradient boosting da un punto di vista operativo non fa altro che stimare iterativamente alberi sui residui ottenuti nel passo precedente e aggiornare in maniera adattiva le stime (ALGORITMO 1).

ALGORITMO 1: Gradient Boosting con funzione di perdita quadratica

---

1. Dato un campione di training $d = (X, y)$ il primo passo è quello di fissare i parametri di regolazione, rappresentati dal numero di alberi B, da un fattore ϵ che rappresenta il tasso di apprendimento, e dal numero di split d. Successivamente si inizializza il valore della stima $\hat{G}_0$ a 0 e il vettore dei residui r ponendoli pari alle osservazioni y del training set.

2. Per b=1,2..B ripetere le seguenti operazioni:

   a Stimare un albero di regressione $\tilde{g}_b$ avente numero di split d sui dati (X, r). Ad ogni passo viene scelto lo split che massimizza la riduzione della funzione di perdita.

   b Modificare la stima del modello con una sua versione regolarizzata in modo da evitare di modificare la stima in maniera troppo ottimistica. $\tilde{g}_b : \hat{G}_b = \hat{G}_{b-1} + \hat{g}_b \Rightarrow \hat{g}_b = \epsilon \cdot \tilde{g}_b$

   c Aggiornare i residui utilizzando le stime ottenute al punto b: $r_i = r_i - \hat{g}_b(x_i), i = 1, .., n$.

Quello riportato sopra è esattamente l'algoritmo che è stato utilizzato per la previsione del numero di commenti.
Tuttavia, come detto in precedenza, di interesse non è solo prevedere il numero di commenti ma anche prevedere se una foto riceva almeno un commento. In quest'ottica, la variabile di interesse è una variabile di Bernoulli, ed in questo caso il Gradient Boosting modella la probabilità π attraverso una successione di alberi, come descritto nella Formula (1.1) riportata di seguito:

$$\pi(x) = G_B(x) = \sum_{b=1}^{B} g_b(x; \gamma_b) \tag{1.1}$$

In questo caso, l'unica differenza rispetto al caso precedente è che la funzione di perdita non è più la canonica funzione di perdita quadratica ma è l'opposto della log-verosimiglianza Bernoulliana. Ad ogni modo, la procedura può essere agevolmente generalizzata per qualsiasi funzione di perdita, e può essere espressa attraverso l'ALGORITMO 2:

ALGORITMO 2: Gradient Boosting con qualsiasi funzione di perdita

---

1. Inizializzare la stima della funzione $\hat{G}_0$ a 0, il numero di alberi B e il learning rate $\epsilon : \epsilon > 0$

2. Per b=1,...,B ripetere i seguenti passi:

   a Calcolare il gradiente della funzione di perdita per ogni osservazione dell'insieme di training: $r_i = \frac{\delta L(y_i, \pi_i)}{\delta \pi_i} \mid_{\pi_i = \hat{G}_{b-1}(x_i)}, 1 = 1, ...n$

   b Dal momento che l'obiettivo è quello di fare previsione dei valori di y anche al di fuori del training set, il passo successivo è quello di approssimare il gradiente con un albero avente un numero di split pari a d risolvendo il seguente problema di minimo: $min_\gamma \sum_{i=1}^{n} (r_i - g(x_i; \gamma))^2$

   c Utilizzando l'albero stimato al passo precedente a questo punto è possibile modificare la stima:
   $\hat{G}_b(x) = \hat{G}_{b-1}(x) + \hat{g}_b(x), con\ \hat{g}_b = \epsilon \cdot g(x_i; \hat{\gamma}_b)$

3. A partire dalla sequenza di alberi $\hat{G}_b(x), b = 1, ..., B$ è possibile ricavare le previsioni sfruttando il principio del voto di maggioranza.

1.0.4 XGBoost

L'XGBoost rappresenta una particolare implementazione del Gradient boosting sempre basato sulla discesa del gradiente. Essi si differenziano principalmente per due aspetti, il primo inerente alla formulazione del modello e il secondo invece legato al metodo di stima. Per quanto riguarda il primo aspetto, si può dire che l'XGBoost utilizzi una funzione obiettivo regolarizzata in modo da ridurre il rischio di sovradattamento del modello ai dati, cosa che non accade con il Gradient boosting e che rende quest'ultimo più sensibile alla scelta dei parametri di regolazione. In particolare, la funzione di perdita dell'XGBoost può essere rappresentata tramite l'equazione (1.2) :

$$L(\gamma, \lambda, \tau) = \sum_{i=1}^{n} L(y_i, \hat{G}_{B-1}(x_i) + g(x_i; \gamma)) + \sum_{b=1}^{B} \Omega(g_b) \; con \; \Omega(g) = \tau T + \frac{1}{2}\lambda \|w\|^2 \qquad (1.2)$$

dove:

- λ e τ rappresentano i parametri di regolazione che gestiscono il trade-off tra varianza e distorsione
- T il numero di foglie nell'albero
- w una funzione punteggio per ogni foglia che sarà tanto più elevata quanto più i soggetti saranno predetti accuratamente

Chiaramente valori elevati di λ e τ assegnano una penalità maggiore a modelli più complessi e questo si traduce in una funzione stimata più liscia ma che segue meno l'andamento dei dati. Viceversa, valori di λ vicini a zero implicano una penalità molto bassa con la conseguenza che il modello avrà un rischio di sovradattamento maggiore.
Per quanto riguarda il secondo aspetto invece,si può dire che il guadagno in termini di costi computazionali sia il motivo principale per cui l'XGBoost sia attualmente uno dei metodi di previsione più utilizzati e quasi sempre tra i primi posti nelle maggiori competizioni di Data Science. Tipicamente, un algoritmo di tipo euristico come quello utilizzato per gli alberi seleziona ad ogni passo lo split e la variabile che consentano di ottenere la maggior riduzione della funzione di perdita. Gli algoritmi di natura euristica, che rappresentano una soluzione sub-ottima ad un problema di minimizzazione, lavorano molto bene anche quando la dimensionalità delle variabili esplicative è alta; tuttavia, quando questa diventa nell'ordine delle migliaia e il numero di osservazioni è elevato come nei dati che si stanno analizzando in questa tesi, il tempo computazionale aumenta inesorabilmente e la soluzione ottenuta oltre ad essere sub-ottimale è anche lunga da ottenere. Per questo motivo è necessario utilizzare un approccio algoritmico 'approssimato', dove il termine approssimato si riferisce al fatto che la ricerca delle variabili e dei relativi split non viene effettuata in tutto lo spazio $\mathbb{R}^P$ ma in particolari punti che vengono proposti. Più nel dettaglio, ad ogni passo vengono definiti una serie di vettori di

proposte della dimensione del numero di nodi presenti a quel passo basandosi principalmente sulla distribuzione delle X stesse, selezionando poi il vettore che massimizza la riduzione della funzione di perdita. Il fatto di proporre vettori di valori di split porta il modello ad avere una struttura piramidale, come mostrato in Figura 1.1, e questo in taluni casi può portare ad un peggioramento delle performance del modello perché ad ogni passo ciascun nodo è costretto a splittarsi.

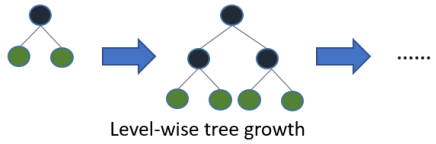


Figura 1.1: *Esempio di split nell'XGBoost*

In generale vi sono due varianti di quest'algoritmo, una denominata globale e l'altra locale. Queste differiscono principalmente per il fatto che nel primo caso le proposte ad ogni passo rimangono immutate, mentre nel secondo caso cambiano. La conseguenza di questo è che il metodo glabale pur essendo leggermente più lento può fornire talvolta dei risultati più accurati perché esplora in maniera più completa lo spazio $\mathbb{R}^P$ delle variabili esplicative.

1.0.5 Light Gradient Boosting

Il Light Gradient boosting rappresenta un'ulteriore versione del Gradient boosting estremamente efficiente dal punto di vista computazionale, da cui l'accezione 'light'. Esso è di fatto molto simile all'XGBoost dal quale si differenza solo per il modo nel quale viene fatto crescere ciascun albero. Infatti, come descritto in precedenza, l'XGBoost definisce alberi con una struttura piramidale, come mostrato in Figura 1.2. D'altra parte, il Light Gradient boosting incoraggia una crescita non necessariamente orizzontale ma che può essere anche verticale , nel senso che ad ogni passo viene semplicemente scelto come nodo di split quello che determina la maggior riduzione della funzione di perdita.

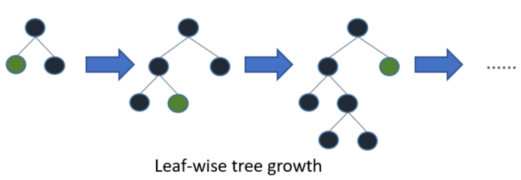


Figura 1.2: *Esempio di split nel LightGBM*

Questo ha come conseguenza che gli split non siano mai forzati, come accade nell'XGBoost, e ciò determina una miglior accuratezza del modello. D'altra parte, il fatto che non vi sia

alcuna costrizione sul modo in cui avviene lo split aumenta il rischio di sovradattamento del modello ai dati, ed è per questo che viene tipicamente sconsigliato l'utilizzo di questo modello in quei contesti in cui il numero di osservazioni è minore di 100.000. Vista la dimensionalità dei dati analizzati in questa tesi, l'utilizzo del LightGBM viste le sue performance è stato ritenuto opportuno.

1.0.6 Support Vector Machines per classificazione

Le Support Vector Machine sono state introdotte da Vapnik (1998) e nascono prevalentemente in un contesto di classificazione. L'idea che sta alla base di questo metodo è quella di costruire degli iperpiani in uno spazio trasformato delle variabili esplicative con l'obiettivo di separare il più possibile due o più classi. Chiaramente, il fatto di lavorare in uno spazio trasformato rende le SVM molto flessibili ed efficaci in tanti contesti. Da un punto di vista pratico, esse altro non sono che una generalizzazione degli iperpiani separatori ottimali, i quali partono dal presupposto che, immaginando di osservare N coppie di valori (x_i, y_i) $i = 1..N, x_i \in \mathbb{R}^P$ *ed* $y_i \in \{-1, 1\}$, sia possibile definire un iperpiano che possa separare completamente le due classi:

$$x : f(x) = x^T\beta + \beta_0 = 0$$

dove β è un vettore a norma unitaria.
La regola di classificazione sarà quindi ragionevolmente del tipo: $G(x) = sign\left[x^T\beta + \beta_0\right]$. Ora, dal momento che le classi sono linearmente separabili da più di un iperpiano, si preferisce quello in grado di massimizzare il *margine*, ovverosia la distanza tra i punti appartenenti a classi diverse, e l'iperpiano. Il problema è un problema di ottimizzazione quadratica convesso con vincoli di diseguaglianza lineare, e viene descritto Formula (1.3):

$$\begin{aligned} &\underset{\beta,\beta_0,\|\beta\|=1}{\text{minimize}} \quad M \\ &\text{subject to} \quad y_i(x_i)(x_i^T\beta + \beta_0) \geq M;\ i = 1, \ldots, N. \end{aligned} \tag{1.3}$$

Ora, se le classi si sovrappongono in $\mathbb{R}^P$ il problema di minimo scritto nella (1.3) non ha chiaramente soluzione. Dunque, il modo più naturale per tener conto di quest'aspetto è continuare a massimizzare M, inserendo però delle variabili $\xi = \xi_1, .., \xi_N$ che contemplino la possibilità che alcuni punti possano trovarsi sul lato sbagliato del margine. In questo modo, i vincoli assumono una forma del tipo:

$$y_i(x_i)(x_i^T\beta + \beta_0) \geq M - \xi_i\ ; \qquad \xi_i \geq 0\ i = 1, \ldots, N,\ \sum_{i=1}^{N} \xi_i \leq C \tag{1.4}$$

Oppure, se si vuole ottenere un problema di ottimizzazione convesso, i vincoli possono esser scritti come segue:

$$y_i(x_i)(x_i^T\beta + \beta_0) \geq M(1 - \xi_i); \qquad \xi_i \geq 0 i = 1, \ldots, N,\ \sum_{i=1}^{N} \xi_i \leq C \tag{1.5}$$

Osservando la natura dei vincoli (1.4 e 1.5) si può intuire una delle motivazioni per le quali le SVM funzionano bene in molti contesti. Infatti, quello che accade è che le osservazioni che sono lontane dalla frontiera e quindi facilmente classificabili non assumono un ruolo rilevante nella definizione dell'iperpiano, cosa che non accade ad esempio nell'LDA dove la matrice di varianza e covarianza dei singoli gruppi viene costruita a partire da tutte le osservazioni.

In un certo senso, l'SVM riesce a concentrarsi sulle osservazioni più complesse da classificare in maniera del tutto autonoma, senza ricorrere ad esempio a tecniche di ricampionamento come accade per l'algoritmo AdaBoost. In definitiva, il problema di minimo può essere espresso attraverso i moltiplicatori di Lagrange (1.6):

$$\begin{aligned} \underset{\beta,\beta_0}{\text{minimize}} \quad & \frac{1}{2} + \|\beta\|^2 + C \cdot \sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & \xi_i \geq 0, y_i(x_i)(x_i^T\beta + \beta_0) \ \geq 1 - \xi_i \ \forall i = 1, \ldots, N. \end{aligned} \tag{1.6}$$

dove C è un parametro di regolazione. Valori elevati di C rispecchiano una penalità elevata alle osservazioni per essere lontane dal margine, per cui gli iperpiani verranno costruiti adattandosi il più possibile ai dati e cercando di separare al meglio le classi. D'altra parte, valori piccoli di C tollerano maggiormente che alcuni punti siano sul lato sbagliato del margine. Da un punto di vista pratico, all'aumentare di C aumenta la varianza del modello mentre al diminuire di C ad aumentare sarà la distorsione, per cui la scelta di tale parametro deve essere effettuata in maniera opportuna. L'aspetto interessante è che la funzione stimata che deriva dalla soluzione del problema di minimo può essere espressa come una combinazione lineare delle y pesate con un'opportuna trasformazione delle variabili esplicative:

$$\hat{f}(x) = \textstyle\sum_{i=1}^{N} \hat{\alpha_i} y_i K(x, x_i) + \hat{\beta}_0$$

dove $K(x, x') = \langle h(x), h(x^T) \rangle$ viene detto $Kernel$ e altro non è che il prodotto interno tra le variabili indipendenti. In particolare, esistono diverse tipologie di $Kernel$, come ad esempio quello polinomiale o radiale, ed esso può esser visto come un iperparametro da scegliere.

1.0.7 Support Vector Machine per regressione

L'idea sottostante alle Support Vector Machine applicate ad un contesto di regressione è quello di individuare una funzione $f(x) = x^T\beta + \beta_0$ che si discosti di un valore non più grande di ϵ dalle y osservate per ciascun valore di x appartenente al training set. Come nel caso della classificazione, si va alla ricerca di un iperpiano che si adatti più o meno bene a tutti i punti senza sovradattarsi ai dati. Il problema di ottimizzazione convessa da risolvere è espresso nella Formula (1.7):

$$\begin{aligned} &\underset{\beta,\beta_0,\|\beta\|=1}{\text{minimize}} \quad \frac{1}{2}\beta'\beta \\ &\text{subject to} \quad |y_i - (x_i^T\beta + \beta_0)| \leq \epsilon \;\; \forall i = 1,\ldots,N. \end{aligned} \tag{1.7}$$

Chiaramente è difficile trovarsi in un contesto nel quale esista una soluzione a tale problema di minimo, per cui quello che si fa è introdurre anche qui delle variabili ξ che consentono alla funzione stimata di avere degli scarti dalle rispettive y maggiori della soglia fissata. Analiticamente il problema di minimo diventa quello descritto nella Formula (1.8):

$$\begin{aligned} &\underset{\beta,\beta_0,\|\beta\|=1}{\text{minimize}} \quad \frac{1}{2}\beta'\beta + C\sum_{n=1}^{N}(\xi_i + \xi_i^*) \\ &\text{subject to} \quad y_i - (x_i^T\beta + \beta_0) \leq \epsilon + \xi_i \quad \forall i = 1,\ldots,N. \\ &\qquad\qquad (x_i^T\beta + \beta_0) - y_i \leq \epsilon + \xi_i^* \quad \forall i = 1,\ldots,N. \\ &\qquad\qquad \xi_i \geq 0 \quad \forall i = 1,\ldots,N. \\ &\qquad\qquad \xi_i^* \geq 0 \quad \forall i = 1,\ldots,N. \end{aligned} \tag{1.8}$$

Come è intuibile, ϵ è un parametro di regolazione che gestisce il $trade-off$ tra varianza e distorsione e riflette la tolleranza verso deviazioni della funzione stimata dai valori osservati. Una volta riscritto il problema attraverso i moltiplicatori di Lagrange, si può mostrare che la funzione derivante dalla minimizzazione della funzione obiettivo può esser scritta anche in questo caso come una combinazione lineare delle y.

$$\hat{f}(x) = \sum_{i=1}^{N}(\hat{\alpha}_i - \alpha_i^*)y_iK(x,x_i) + \hat{\beta}_0$$

dove:

- $K(x,x)$ è un generico Kernel tipicamente lineare,Gaussiano o polinomiale
- α ed α^* rappresentano i moltiplicatori di Lagrange non negativi tali che $0 < \alpha_i \leq C$ e $0 < \alpha_i^* \leq C \; \forall i = 1,\ldots,N$.

Un aspetto da tenere in considerazione e che rappresenta un elemento che contraddistingue le SVMr ha a che vedere con la funzione di perdita utilizzata. Infatti, quella a cui si fa riferimento è la funzione $\epsilon - insentitive$, la quale ha la seguente espressione:

$$L_\epsilon = \begin{cases} 0 & se\ |y - f(x)| \leq \epsilon \\ |y - f(x)| - \epsilon & altrimenti \end{cases} \tag{1.9}$$

Come si può notare, la $\epsilon - insensitive$ (1.9) oltre ad essere robusta rispetto a valori estremi fa si che il modello si concentri maggiormente sulle osservazioni più difficili da prevedere e quindi riprende in un certo senso lo stesso principio della SVMc.

Capitolo 2

Zoom: Reti Neurali

Cos'è una rete neurale? Dare una risposta univoca a questa domanda non è cosa semplice. Infatti, si può dire che essa cambi a seconda della figura professionale che risponde, che sia un informatico, un fisico, o uno statistico. Dal momento che questa è una tesi di laurea in statistica, una possibilità è definire senza perdita di generalità le reti neurali come '*una classe di modelli parametrici altamente parametrizzati che attraverso una stima algoritmica riescono a scoprire relazioni non lineari tra variabili esplicative e una o più variabili di risposta*'. Tale classe di modelli è applicabile sia in contesti di classificazione che di regressione, e prende il nome di rete neurale perché in più di un aspetto riprende il modo in cui si pensava agisse il cervello umano. Per capire il perché di quest'affermazione, è necessario partire dall'architettura di un neurone, che rappresenta l'unità fondamentale in un modello di questo tipo.

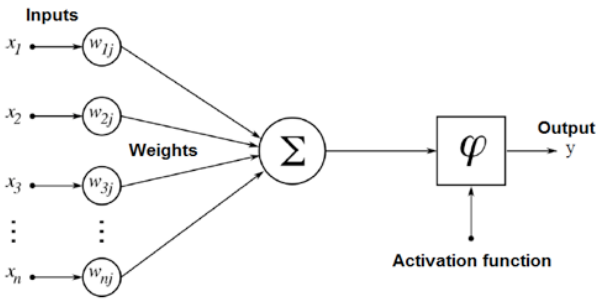


Figura 2.1: *Architettura di un neurone*

Come mostra la Figura 2.1, il primo elemento rilevante è rappresentato dalle sinapsi, che al contrario di come accade per il cervello possono qui assumere valori sia positivi che negativi. Esse rappresentano un fattore di scala degli *input*, e il valore assunto dipende di fatto dall'importanza che questi hanno per svolgere un certo compito, nel nostro caso minimizzare una funzione obiettivo. Successivamente, di questi input pesati ne viene fatta una combinazione lineare, sulla quale viene poi applicata una funzione tipicamente non lineare. Tale funzione, detta *di attivazione*, fornisce maggiore flessibilità al modello.

Visto che la sua scelta rappresenta un aspetto rilevante per le performance di quest'ultimo, ai diversi tipi di funzioni di attivazione verrà dedicata la sottosezione 2.1.5. A partire dal concetto di neurone, è possibile a questo punto fornire un'altra definizione di rete neurale, ossia quella di *modelli costituiti da una serie di neuroni che sono tra loro interconnessi seguendo lo schema di un grafo aciclico*. Ciò ha come conseguenza il fatto che alcuni neuroni possano rappresentare degli input per altri neuroni, e la motivazione per la quale il grafo sia aciclico riguarda la fase di stima; infatti, se così non fosse, avremmo nel passo in avanti dell'algoritmo di stima un loop infinito per cui non potremo stimare i parametri del modello sfruttando il principio del *back-propagation*. Una volta chiarita l'architettura di un neurone comincia ad essere evidente il perché questi modelli prendano il nome di reti neurali: infatti, quello che fanno è acquisire informazioni attraverso un processo di apprendimento, che è un pò quello che accade con l'esperienza per l'uomo. Inoltre, dal momento che veicolano l'informazione rilevante, i parametri del modello possono essere associati alle sinapsi, che nell'uomo misurano l'impulso presente tra due cellule comunicanti. Nelle prossime sottosezioni verranno discussi diversi tipi di rete neurale, il metodo di stima con annessa regolarizzazione, e come detto in precedenza le diverse funzioni di attivazione.

2.1 Architettura di una rete neurale

2.1.1 Reti Neurali a Singolo Strato

Le reti neurali a singolo strato rappresentano il caso più semplice di questa classe di modelli, tuttavia sono utili per comprenderne il funzionamento generale.

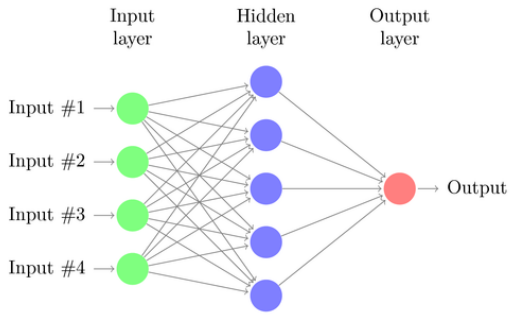


Figura 2.2: *Architettura di una rete neurale a Singolo strato*

Da un punto di vista statistico le reti neurali a singolo strato non sono altro che una regressione (o classificazione) a due stadi; in particolare, nel primo stadio viene effettuata una regressione non lineare per non far collassare tutto il modello in un modello lineare, mentre nel secondo stadio la linearità della regressione dipende sostanzialmente dal tipo di problema che stiamo affrontando. La Figura 2.2 mostra un esempio chiarificatore. Qui abbiamo una generica rete neurale a singolo strato composta da quattro variabili esplicative, uno strato nascosto

composto da cinque neuroni, ed una output unidimensionale. Senza perdita di generalità, è possibile descrivere ciascuno degli elementi in maniera analitica attraverso l'equazione (2.1):

$$f(x) = h(w_0^{(2)} + \sum_{l=1}^{5} w_l^{(2)} a_l) \tag{2.1}$$

dove:

- $a_l = g(w_{l0}^{(1)} + \sum_{j=1}^{4} w_{lj}^{(1)} x_j)$ rappresenta l'l-esimo neurone

- $w_{lj}^{(1)}$ rappresenta il peso che ha la j-esima variabile esplicativa nel prevedere l'l-esimo neurone dello strato latente

- w_{l0}^{1}, comunemente detto bias, coglie la componente di livello e altro non è che un'intercetta

- g è una funzione non lineare ed è quella che in precedenza è stata definita come funzione di attivazione

2.1.2 Reti Neurali Multistrato

Nella precedente sottosezione è stata definita una rete neurale a singolo strato che, come abbiamo visto, ha una chiave di lettura dal punto di vista statistico molto chiara; tuttavia, come è intuibile, un modello così "semplice" difficilmente riesce a cogliere tutti gli elementi di non linearità necessari a descrivere le relazioni tra le variabili, per cui nella pratica le reti utilizzate sono sempre a più strati. L'utilizzo di reti neurali multistrato consente di aggiungere enorme flessibilità al modello, tant'è che si dimostra che una rete neurale di questo tipo è *un approssimatore universale*, ossia riesce ad approssimare qualsiasi funzione sufficientemente liscia. D'altra parte, le ottime performance in termini di capacità predittiva compensano le difficoltà che questi modelli incontrano nel momento in cui l'obiettivo primario diventa quello di estrarre le $features$ più importanti, come ad esempio un indice di importanza di variabili. A tal proposito, se si esclude qualche proposta isolata (Garson), poco è stato fatto in questa direzione, per cui se l'obiettivo dell'analisi non è strettamente previsivo si può pensare di preferire altri modelli. Tornando alle reti neurali multistrato, osservando la Figura 2.3 si può notare come una rete di questo tipo mantenga la stessa architettura di una rete neurale a singolo strato e differisca da questa soltanto per il numero di strati latenti.

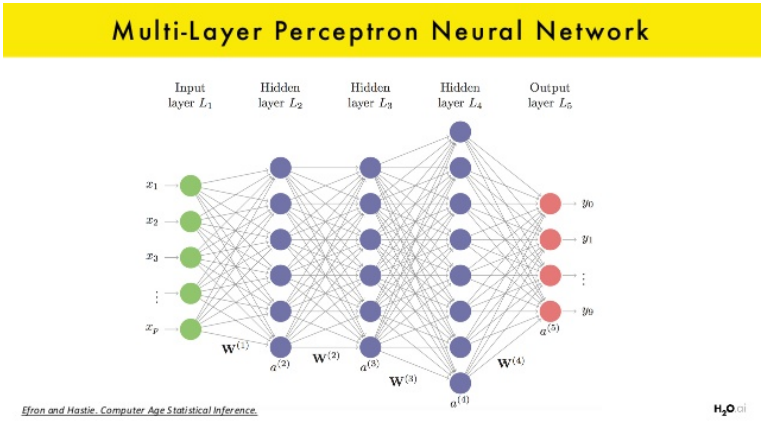


Figura 2.3: *Architettura di una rete neurale multi-strato*

Il fatto di avere un numero di strati maggiore ha come conseguenza che il modello sia altamente sovraparametrizzato. Questo, da un lato ha come conseguenza che il costo computazionale per stimarlo sarà elevato e dall'altro un elevato rischio di sovradattamento. Alle tecniche più utilizzate per regolarizzare una rete neurale verrà dedicata la sottosezione 2.1.4. Ora, da un punto di vista analitico descrivere una rete neurale multistrato è senz'altro più complesso rispetto alla sua controparte a singolo strato; tuttavia, se si fa riferimento alla notazione utilizzata in Efron, Hastie e al., (2016), la generalizzazione del modello diventa senz'altro più semplice. In particolare, passando alla descrizione analitica, possiamo indicare con:

- $z_l^{(2)} = w_{lo}^{(1)} + \sum_{j=1}^{p} w_{lj}^{(1)} x_j$ la combinazione lineare delle variabili esplicative che faranno da input per l'l-esimo neurone del secondo strato.

- $a_l^{(2)} = g^{(2)}(z_l^{(2)})$ l'l-esimo neurone del secondo strato, con g generica funzione tipicamente non lineare. Ricordiamo che tale funzione g mantiene la stessa struttura dal primo al (K-1)-esimo strato, mentre nell'ultimo essa è tipicamente una funzione identità in contesti di regressione e una funzione $softmax$ in quelli di classificazione. Nel secondo caso, questa altro non è che l'inversa della funzione g utilizzata nel GLM multinomiale, per cui la formulazione è quella espressa nell'equazione (2.2) :

$$f(z(x)) = g^{(K)}(z_m^{(K)}; z^{(K)}) = \frac{e^{z_m(K)}}{\sum_{l=1}^{M} e^{z_l(K)}} \tag{2.2}$$

Tale notazione, valida per un numero di strati pari a due, può essere facilmente generalizzata al caso di una rete Fully-connected a K strati. Quindi, per il generico strato K avremo:

- $z_l^{(k)} + \sum_{j=1}^{p_{k-1}} w_{lj}^{(k-1)} a_j^{(k-1)}$ l'l-esima combinazione lineare dei neuroni (o delle variabili esplicative se k=2)

- $a_l^k = g^{(k)}(z_l^{(k)})$ l'l-esimo neurone del k-esimo strato

2.1.3 Stima di una rete neurale (cenni)

Una volte stabilita l'architettura della rete neurale scegliendo il numero di strati, il numero di neuroni per ciascuno strato, e non per ultimo il tipo di connessione che si vuol dare tra i neuroni, il passo successivo è quello della stima. Chiaramente, l'obiettivo è trovare dei valori per i pesi W tali che questi massimizzino l'adattamento del modello ai dati. Senza perdita di generalità, è possibile definire la funzione di perdita attraverso l'equazione (2.3):

$$\min_{W} \quad \frac{1}{n}\sum_{i=1}^{n} L[y_i, f(x_i, W)] + \lambda J(W) \tag{2.3}$$

dove la componente di penalizzazione $\lambda J(W)$ è presente solo se la regolarizzazione è di tipo *weight decay*. D'altra parte, la funzione $L[y_i, f(x_i; W)]$ rappresenta il fulcro della funzione di perdita ed è una funzione convessa in f ma non in $\hat{W}$, che tipicamente presenta tanti minimi locali di cui alcuni molto lontani dal minimo assoluto. Questo è uno dei punti deboli delle reti neurali a cui si tenta di sopperire in diversi modi, ad esempio attraverso la tecnica del *batch-learning*, in cui si utilizzano i dati poco alla volta aggiornando più volte le stime in una singola *epoca*. Ciò consente un'esplorazione migliore dello spazio parametrico. Ora, a seconda di che si stia trattando un problema di classificazione o di regressione, la componente $L[y_i, f(x_i; W)]$ può assumere diverse forme. Infatti, se stiamo affrontando un problema di regressione, dove il vettore aleatorio $Y = (y_1, y_l..y_n) \in \mathbb{R}^P$, le più conosciute sono:

- l'errore quadratico medio di previsione, o eventualmente una sua trasformazione monotona come la radice quadrata: MSE(W)$= \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i; W))^2$
- l'errore medio assoluto, tipicamente meno utilizzato per i problemi che causa in termini di differenziabilità: MAE(W)$= \frac{1}{n}\sum_{i=1}^{n}|y_i - f(x_i; W)|$
- l'errore medio $\epsilon - insensitive$. Esso ha il vantaggio di essere robusto alla presenza di valori anomali, presenta ϵ come parametro di regolazione ed ha la seguente forma: $IL(W, \epsilon) = \sum_{i=1}^{n} max(|y_i - f(x_i, W)| - \epsilon, 0)$

D'altra parte, se il problema che stiamo affrontando è un problema di classificazione, la variabile di risposta assume valori in un insieme discreto. In questo caso, le funzioni a cui si fa maggiormente riferimento sono:

- indice di Gini: $GI(W) = \sum_{k=1}^{K} y_{ik}(1 - y_{ik})$
- la cross-entropia, che ha anche un'interpretazione frequentista in quanto è legata alla verosimiglianza multinomiale. Infatti, $CH(W) = -\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} log f_k(x_i; W)$
- tasso di errata classificazione, $ER(W) = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq f(x_i; W))$

Ora, quale che sia il problema che si affronta, la soluzione al problema di minimo espresso nella formulazione precedente va trovata chiaramente per via numerica attraverso un algoritmo iterativo. Il problema principale è che molti algoritmi numerici sono basati sul calcolo del gradiente

primo e secondo, come quelli Newton-Raphson, utilizzati in una loro variante nella stima dei GLM (dove prende il nome di *Scoring di Fisher*). In contesti come quello delle reti neurali un approccio di questo tipo non è sostenibile da un punto di vista computazionale, visto l'alto grado di parametrizzazione che caratterizza tali modelli. Infatti, se si pensa che ad ogni passo di un algoritmo Newthon-Raphson il numero di operazioni richieste per calcolare il gradiente secondo cresce col quadrato del numero dei parametri, appare evidente come sia necessario utilizzare un algoritmo che consenta di diminuire il carico computazionale. Ad oggi, l'algoritmo maggiormente utilizzato per stimare una rete neurale è l'algoritmo di *back-propagation*, il quale oltre a non richiedere il computo del gradiente secondo non necessita nemmeno il calcolo del gradiente primo in tutti gli strati. Infatti, a partire da pesi inizializzati vicino a zero, corrispondenti ad una soluzione altamente regolarizzata, quello che quest'algoritmo fa è calcolare il gradiente solo nell'ultimo strato; successivamente, il gradiente viene propagato all'indietro per ottenere i gradienti negli altri strati e quindi l'aggiornamento dei parametri. Da un punto di vista pratico, l'idea è che questi gradienti $\delta_l^{(k)}$ rappresentino il contributo del generico neurone nell'errore di previsione del vero valore della variabile dipendente y. In particolare, per l'ultimo strato di attivazione $a_l^{(K)}$ questi errori hanno un'interpretazione semplice e sono associabili ai residui parziali del modello. D'altra parte, per gli strati latenti interni l'errore $\delta_l^{(k)}$ rappresenta una media pesata dei termini d'errore che hanno proprio $a_l^{(k)}$ come input. Andando più nel dettaglio, l'ALGORITMO 3 mostra le fasi di una generica iterazione del *back-propagation*, considerando un caso generale e quindi non utilizzando una penalizzazione di tipo *weight decay*.

ALGORITMO 3: BACK PROPAGATION

---

1. Considerando l'esempio di un vettore aleatorio (x, y), partendo dai pesi correnti viene effettuato il passo in avanti dell'algoritmo. Esso si basa sostanzialmente sul computo dei nodi $a_l^{(k)}$ in ognuno degli strati $L_2, L_3, ..., L_K$ e successivamente nella stima di $f(x; W)$, salvando tutte le quantità che verranno utilizzate nei passi successivi.

2. Il passo successivo è calcolare per ciascun neurone presente nell'ultimo strato L_K il termine d'errore δ_l^k mediante la *regola della catena*.

$$\delta_l^{(K)} = \frac{\partial L[y, f(x, W)]}{\partial z_l^K} \delta_l^{(K)} = \frac{\partial L[y, f(x; W)]}{\partial a_l^{(K)}} \cdot g'^K(z_l^K) \tag{2.4}$$

3. Dopo aver calcolato il gradiente nell'ultimo strato, questo viene propagato all'indietro per ottenere quello del generico strato k:

$$\delta_l^{(k)} = (\sum_{j=1}^{p_{k+1}} w_{jl}^{(k)} \cdot \delta_j^{(k+1)}) \cdot g'^k(z_l^k) \tag{2.5}$$

4. Infine, a partire dai termini d'errore δ_l^k, è possibile ricavare le derivate parziali rispetto ai pesi ed utilizzarle per aggiornare i pesi stessi.

$$\frac{\partial L[y, f(x; W)]}{\partial w_{lj}^{(k)}} = a_j^{(k)} \cdot \delta_l^{(k+1)} \tag{2.6}$$

Una volta ottenute le derivate parziali (2.6), aggiornare i pesi è un passaggio immediato. Ci sono diversi modi per far questo: quello più classico è il metodo del Gradient Descent, che sostanzialmente in un'epoca aggiorna i parametri una volta soltanto considerando tutte le osservazioni contemporaneamente. Passando ad una notazione matriciale, mettendoci nel caso di utilizzare una penalizzazione L_2, l'aggiornamento può esser descritto analiticamente dall'espressione (2.7):

$$\begin{aligned} \Delta W^{(k)} &= \frac{1}{n} \sum_{i=1}^{n} \frac{\partial L[y_i, f(x_i; W)]}{\partial W^{(k)}} \\ W^{(k)} &\Leftarrow W^{(k)} - \alpha(\Delta W^{(k)} + \lambda W^{(k)}), k = 1..., K-1 \end{aligned} \tag{2.7}$$

Il metodo del Gradient Descent (2.7) è probabilmente il più conosciuto ma non è il più efficiente dal punto di vista computazionale. Un'alternativa proposta di recente è quella dell'Accelerated Gradient Descent (2.8), il quale definisce una dipendenza tra l'aggiornamento al passo t+1 e quelli precedenti. La struttura di dipendenza tra gli aggiornamenti è definita dalle equazioni nella (2.8):

$$\begin{aligned} V_{t+1} &= \mu V_t - \alpha(\Delta W_t + \lambda W_t) \\ W_{t+1} &= W_t + V_{t+1} \end{aligned} \tag{2.8}$$

dove V_t è un vettore che veicola tutta l'informazione sui pesi nelle iterazioni precedenti, mentre μ è un parametro che in sostanza definisce il peso che i precedenti aggiornamenti hanno sull'ultimo. In generale, scegliere in maniera appropriata il parametro μ può avere un costo computazionale oneroso, tuttavia se scelto in maniera opportuna l'algoritmo può avere dei tassi di convergenza decisamente migliori rispetto al Gradient Descent nella sua versione classica.

2.1.4 Regolarizzazione di una Rete Neurale

Come accennato in precedenza, la fase di regolarizzazione di una rete neurale assume un ruolo cruciale quando il numero di strati latenti è elevato. Infatti, più strati latenti significa più parametri e più parametri significa maggiori rischi di sovradattamento del modello ai dati. Inoltre, seppur sia vero che quando abbiamo tante osservazioni a disposizione è meno probabile che il modello colga delle variabilità locali, le reti neurali multistrato hanno cosi tanti parametri che questa eventualità è sempre dietro l'angolo. Per questa ragione, visto che ridurre la dimensione della rete non è una strada perseguibile dal momento che è proprio quella che ci consente di cogliere anche i *pattern* più complessi, è necessario effettuare una qualche forma di regolarizzazione per migliorare l'errore di previsione. Un metodo molto utilizzato a questo scopo anche al di fuori del contesto delle rete neurali per prevenire il sovradattamento è quello di far ricorso ad un *ensamble* di modelli, come ad esempio Bagging, Boosting e Random Forest. La stessa logica può essere quindi esportata alle reti neurali attraverso la tecnica del *dropout* (Srivastava et al., 2014). In breve, in fase di training ad ogni passo dell'algoritmo di stima un neurone viene disattivato con una probabilità fissata pari a ϕ, e con esso vengono chiaramente eliminate tutte le connessioni che il neurone ha con gli altri neuroni. Chiaramente ϕ è un parametro di regolazione, tuttavia in tante applicazioni un valore pari a 0.5 consente di ottenere delle ottime performance ed evitare la fase di *tuning* che può essere computazionalmente onerosa. Completata la fase di *training*, in fase di previsione si utilizzerà la rete con la sua architettura completa andando semplicemente a moltiplicare per ϕ ciascuna unità, approssimandone in questo modo il valore atteso. (Figura 2.4).

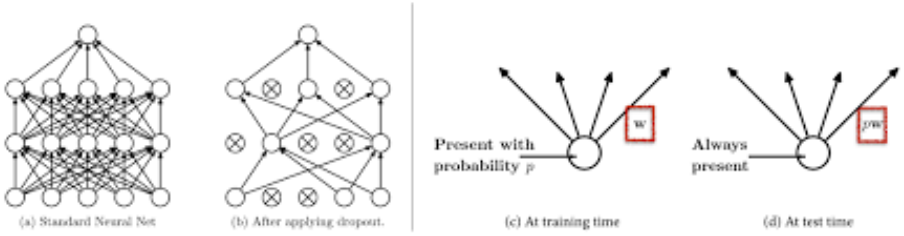


Figura 2.4: *Esempio di applicazione di dropout*

Nella sua semplicità la tecnica del dropout lavora estremamente bene, e come detto, riprende l'idea di ricorrere ad un ensamble di modelli stimandone alla fine uno soltanto. Infatti, allenare diverse reti separatamente, magari con un'architettura diversa, richiederebbe dei tempi di calcolo eccessivi mentre in questo modo riusciamo ad ottimizzare tutto il processo di stima.
Un'alternativa estremamente innovativa proposta negli ultimi anni è quella della Dataset augmentation (Wong et al., 2016). Essa è particolarmente utilizzata in tutti quei contesti nei quali la dimensione dei dati non è molto ampia e quindi un qualsiasi metodo di machine learning farebbe fatica a fornire delle buone performance. L'idea è quindi quella di aumentare la dimensione del dataset in maniera artificiale a partire dal training set, e nella pratica vi sono diversi

modi per farlo. Il primo, intuitivo, è quello di creare diverse versioni delle immagini iniziali attraverso operazioni di rotazione, traslazione, o rimpicciolimento, ed incrementare in questo modo la dimensione del training set. Il secondo approccio, introdotto da Ian Goodfellow (2016) e trattate in Zhu et al., (2017), si basa invece sulle cosidette GAN (Generative Adversarial Newtorks) e poggiano sulla seguente idea: invece di utilizzare una singola rete neurale, vengono costruite due reti neurali, in cui la prima sostanzialmente si allena a riprodurre fedelmente un'immagine, mentre la seconda si allena a riconoscere l'immagine reale dalla sua riproduzione.

Figura 2.5: *Esempio di GAN*

In un certo senso, le due reti sono in competizione l'una con l'altra, e al termine del processo idealmente avremo da un lato diverse riproduzioni di una certa immagine, di cui si riporta un esempio in Figura 2.5, e dall'altra una rete che avrà un tasso di falsi positivi molto basso, visto che si è allenata su delle imitazioni che sono via via sempre più indistinguibili dall'immagine originale.

Il *dropout* e ancor più la Dataset Augmentation rappresentano l'ultima frontiera della regolarizzazione di una rete neurale, tuttavia in precedenza venivano utilizzate delle tecniche diverse che descriviamo brevemente per completezza, ossia l'*early stopping* e il *weight decay*. Per quanto riguarda la prima, essa si basa sostanzialmente sul non allenare la rete fino a convergenza fermandosi un certo numero di iterazioni prima. L'idea che sta alla base di questo approccio è che all'inizio della fase di training di una rete, se i pesi sono stati inizializzati con valori attorno allo 0, essa si avvicina molto ad un modello lineare visto che sfruttiamo solo la parte lineare della funzione di attivazione (Hastie et al., 2009 per dettagli). Di conseguenza, la rete parte da un modello altamente regolarizzato ed iterazione dopo iterazione coglie relazioni più complesse, per cui fermando l'algoritmo prima della convergenza evitiamo che il modello colga dei *pattern* che riflettano variabilità stocastica. Passando invece alla regolarizzazione *weight decay*, essa agisce allo stesso modo in un modello di regressione o classificazione standard, e quello che fa è modificare la funzione di perdita $L(W)$ introducendo una penalizzazione sulla dimensione dei pesi, passando quindi ad una formulazione del tipo $L(W) + \lambda J(W)$.

In particolare, ci sono due possibilità per quanto riguarda la norma utilizzabile nella penalizzazione:

$$L_2 = \sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{j=1}^{p}(w_{jl}^{(k)})^2 \tag{2.9}$$

$$L_1 = \sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{j=1}^{p}(|w_{jl}^{(k)}|) \tag{2.10}$$

La (2.9) è esattamente la penalizzazione utilizzata nella regressione ridge e schiaccia verso 0 i pesi alleviando l'incremento della varianza delle stime dato dalla correlazione tra i neuroni. D'altra parte, la (2.10) è equivalente a quella utilizzata nel LASSO, ed è dal punto di vista pratico più vicina al *dropout* visto che ha come effetto la "disattivazione" di alcuni neuroni.

2.1.5 Funzioni di attivazione

L'ultima sottosezione di questo capitolo è dedicata alle funzioni di attivazione, che, come detto in precedenza, rappresentano la motivazione principale per la quale tali modelli sono cosi flessibili ed ottengono spesso delle performance soddisfacenti. Andando più nel dettaglio, possiamo dire che vi sono sostanzialmente cinque tipologie di funzioni di attivazione:

- La funzione *sigmoide* mappa i valori in input dall'asse reale in (0,1), e si avvicina agli estremi del codominio per valori grandi in valore assoluto, come si osserva in Figura 2.6.

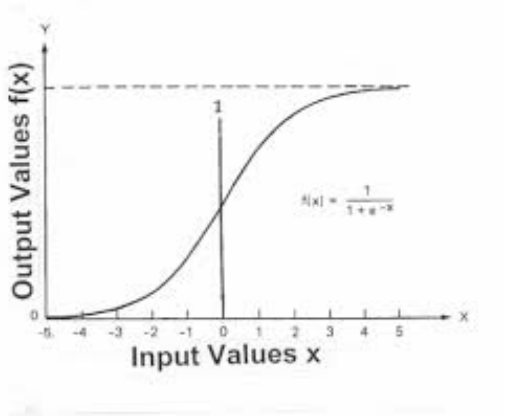


Figura 2.6: *Funzione Sigmoide*

Essa è stata ampiamente utilizzata inizialmente, salvo poi essere sostituita da altre funzioni. Le motivazioni che stanno alla base di questo sono essenzialmente due:

  - Una proprietà assolutamente indesiderabile e che porta a dei risultati poco affidabili in termini di stima è la cosiddetta saturazione del gradiente. In breve quando la funzione di attivazione assume dei valori vicini agli estremi del codominio, il gradiente della funzione obiettivo tende ad annullarsi. Questo ha come conseguenza che la rete fa fatica ad apprendere e l'algoritmo si arresta bruscamente.

– Un altro aspetto negativo che si riscontra utilizzando una funzione di attivazione di questo tipo entra in gioco nel caso in cui gli input siano tutti positivi. Infatti, se questo accade, il fatto che la funzione di attivazione sigmoide non sia centrata in 0 causa il fatto che il gradiente sia fatto o di elementi tutti positivi o di elementi tutti negativi. Questo comporta un rallentamento nella convergenza dell'algoritmo qualora il minimo della funzione obiettivo preveda dei pesi con segni alterni. Tuttavia, c'è da dire che tale problema può essere attenuato semplicemente attraverso un'operazione di *batch learning*.

- Un'altra funzione di attivazione talvolta utilizzata è la *tangente iperabolica*, che mappa i valori in input su un codominio [-1,1], come è possibile vedere in Figura 2.7. Tale funzione, dal momento che non è centrata in 0 viene sempre più preferita alla funzione sigmoide, dal momento che evita i problemi discussi in precedenza.

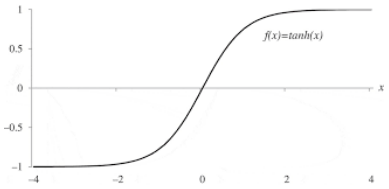


Figura 2.7: *Funzione Tangente Iperbolica*

- La funzione di attivazione più largamente utilizzata nella pratica è sicuramente la *ReLU*, che altro non è che una funzione lineare a tratti $f(x) = max(0, x)$ (Figura 2.8).

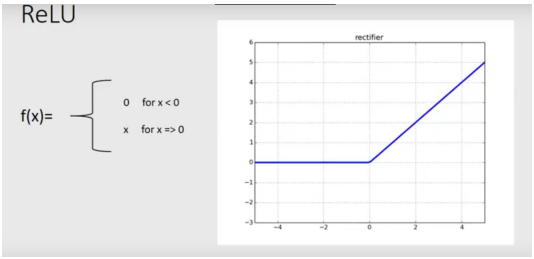


Figura 2.8: *Funzione ReLU*

Tale funzione di attivazione presenta sicuramente dei vantaggi enormi, primo tra questi il fatto che il suo computo non necessita di operazioni esose come negli altri due casi ma tutto ciò che bisogna fare è semplicemente trasformare in zeri i valori negativi. Inoltre, da un punto di vista computazionale essa accelera la convergenza dell'algoritmo di stima, e questo è dovuto al fatto che trattandosi di una funzione lineare a tratti il gradiente è essenzialmente costante. Tuttavia, anche la funzione *ReLU* presenta degli aspetti negativi, e quello che ha un impatto maggiore è senz'altro la "morte" del neurone. Questo accade sostanzialmente quando abbiamo dei gradienti molto elevati e i pesi vengono modificati

in maniera tale che quel neurone non sia più attivo per la restante fase di *training*. Uno dei modi per contrastare questo problema è quello di definire un tasso di apprendimento più basso, in modo da rendere meno forte l'impatto del gradiente nell'aggiornamento dei parametri.

- Per ovviare al problema riscontrabile con la *ReLU*, è stata introdotta la *Leaky ReLU* (Figura 2.9), la quale invece di trasformare i valori negativi in zeri li riscala verso valori negativi molto vicini a zero. La funzione *Leaky ReLU* è descritta di seguito dall'espressione (2.11):

$$I(x \leq 0) \cdot \alpha \cdot x + I(x \geq 0) \cdot x \tag{2.11}$$

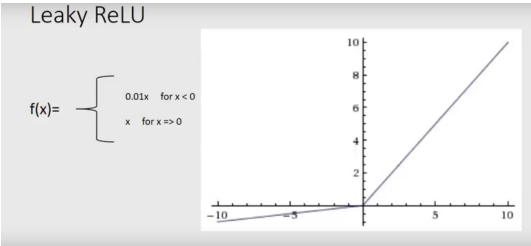


Figura 2.9: *Funzione Leaky-ReLU*

- Un'altra funzione introdotta recentemente e che generalizza la *ReLU* per evitare il problema della "morte" del neurone è la funzione di attivazione *maxout* (2.12). Questa fa un'operazione diversa rispetto alle altre, nel senso che non applica una funzione non lineare ad una combinazione lineare degli input, ma un'operazione di massimo tra diversi elementi. In particolare, la sua espressione è descritta nella (2.12):

$$max(w_1^T x + b_1, w_2^T x + b_2) \tag{2.12}$$

Da notare che per w_1^T e b_1 pari a zero abbiamo la funzione *ReLU* (Figura 2.8). Essendo una generalizzazione della *ReLU*, la funzione *maxout* ne condivide i vantaggi ed evita quindi anche la saturazione del gradiente.

2.1.6 Convolutional Neural Network

Le reti neurali convoluzionali rappresentano una tipologia di rete neurale ampiamente utilizzata nell'ambito del riconoscimento d'immagini. Infatti, esse vengono applicate con buoni risultati per identificare volti, oggetti, o anche segnali stradali come accade per le self driving cars. Allo stesso modo delle reti neurali standard, esse sono costituite da neuroni, le cui sinapsi (pesi e intercette), vengono definite attraverso una fase di apprendimento. La differenza sostanziale tra una rete neurale classica e una CNN sta nel fatto che quest'ultima esplicita l'assunzione che l'input sia rappresentato da un'immagine, e ciò si traduce in un'architettura della rete ben precisa. Ma quali sono le motivazioni che spingono all'utilizzo di una rete neurale diversa rispetto a quelle tradizionali?
Ricordiamo che tipicamente una rete neurale non fa altro che ricevere dei dati in input e trasformarli attraverso una serie di strati nascosti per cogliere relazioni complesse tra gli input stessi e le variabili di risposta. Ora, immaginiamo di avere un'immagine di dimensione estremamente ridotta, ad esempio $32 \times 32 \times 3$ (dove questi valori si riferiscono ad altezza, larghezza e numerosità dei canali); se volessimo costruire una rete *Fully Connected* per classificare il contenuto dell'immagine, considerando soltanto un neurone avremo già 3072 pesi. Chiaramente, una rete neurale con un singolo neurone non è molto utile, per cui è evidente come il numero di parametri tenda a crescere in maniera estremamente elevata all'aumentare del numero di neuroni e di strati, causando un problema di sovradattamento. Inoltre, tutto questo discorso è stato fatto per un'immagine dalle dimensioni ridotte, caso molto raro nella realtà. Detto questo, è importante sottolineare che se da un lato il problema del sovradattamento sia estremamente rilevante, non è comunque quello più problematico. Infatti, trattando il pixel di ogni canale come una variabile a sé stante, il modello non tiene conto in alcun modo della disposizione spaziale degli stessi, rendendo in questo modo più difficile l'estrazione delle *features* necessarie per riconoscere l'immagine. D'altro canto, le reti neurali convoluzionali, come si vedrà a breve, non sviluppano le operazioni su una ma su tre dimensioni, ossia larghezza, altezza e profondità, dove quest'ultima fa sempre riferimento al numero di canali.
Nelle successive sottosezioni verrà fornita una breve descrizione dei diversi strati di CNN, cercando di chiarire il perché questa architettura consenta di ottenere delle performance spesso buone. Per facilità di comprensione, si riportano di seguito i diversi strati di una rete neurale convoluzionale descritti di seguito:

- Convoluzione
- Relu.
- Pooling, detta anche sub-Sampling
- Classificazione

2.1.7 Strato di convoluzione

L'operazione di convoluzione consiste fondamentalmente nell'applicazione di filtri, i cui valori vengono appresi durante la fase di *training* della rete. Tali filtri si estendono su una porzione ristretta delle prime due dimensioni e su tutta la terza dimensione, per cui vi è un'asimmetria nel modo in cui vengono trattate le tre dimensioni nel senso che il numero di canali rimane quasi sempre immutato. Un esempio di convoluzione è mostrato in Figura 2.10.

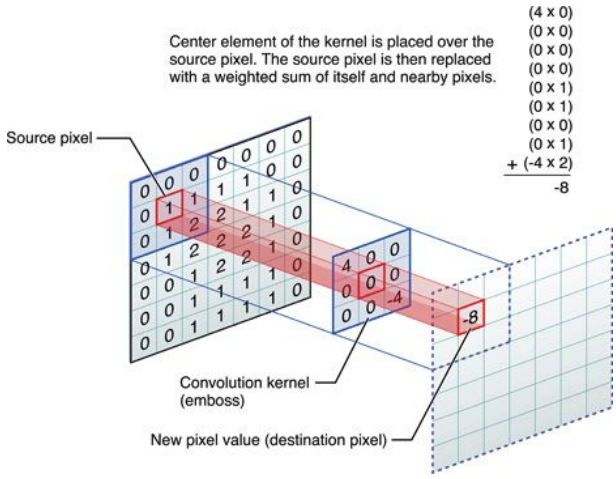


Figura 2.10: *Esempio di Convoluzione*

In generale l'estensione spaziale del filtro rappresenta un iperparametro, e sempre in un'ottica di confronto con il cervello umano viene chiamato campo recettivo, in quanto definisce la regione nella quale lo stimolo viene effettuato. Da un punto di vista pratico, il filtro viene applicato facendolo slittare lungo le prime due dimensioni e calcolando ad ogni passaggio il prodotto interno tra gli elementi del filtro e quelli dell'input che si trovano in una certa posizione spaziale e per questo coinvolti nell'operazione di convoluzione. La dimensione dell'ouptut dipende di fatto da tre iperparametri, che sono profondità, *stride* e $zero - padding$. Partendo dal primo, esso corrisponde essenzialmente al numero di filtri che vorremmo utilizzare, tenendo a mente che ciascuno di questi può apprendere aspetti differenti dell'immagine. Ad esempio, partendo dall'immagine grezza ed immaginando di considerare uno strato convoluzionale come primo strato, dei neuroni potrebbero attivarsi in presenza di particolari archi, colori e blobs, ossia da regioni di pixel caratterizzati da intensità comuni. In seguito, gli strati di convoluzione successivi potrebbero attivarsi per cogliere features ad un livello più alto, come una determinata parte del corpo se stiamo classificando persone.
Per quanto riguarda il secondo iperparametro, lo *stride*, esso definisce di fatto di quanti pixel il filtro debba muoversi ad ogni passo. Sempre a titolo di esempio, diciamo che quando abbiamo un passo pari a 3, questo significa che il filtro si sposta di 3 pixel lungo la dimensione orizzontale. Chiaramente il passo di un filtro ha un peso rilevante su quella che è la dimensione dell'output, visto che filtri con *stride* elevati saranno caratterizzati da dimensioni dell'ouput minori e quindi

comporteranno un risparmio in termini di parametri. Un aspetto da tenere in considerazione è che questo iperparametro non può essere definito in maniera completamente arbitraria, ma è ragionevolmente soggetto a dei vincoli.
Infine, il terzo iperparametro entra in gioco in quei contesti nei quali per qualche motivo vogliamo preservare la dimensione spaziale dell'output, e, da un punto di vista pratico non fa altro che ricoprire la matrice in input con uno strato di zeri. Questo può essere utile soprattutto quando abbiamo immagini di dimensioni non molto elevate, perché in questi casi l'applicazione ripetuta di filtri tendenzialmente ha come conseguenza un'eccessiva riduzione dell'output e una perdita eccessiva di informazione. Inoltre, vi è un'altra motivazione non meno importante che spinge all'introduzione di questi ricoprimenti: infatti, facendo questo possiamo applicare il filtro ad ogni elemento della matrice degli input, senza tralasciarne alcuno. Infatti, in linea generale non è possibile applicare un filtro ai pixel che non hanno elementi vicini in tutto l'intorno.

2.1.8 Strato di Pooling

In molte applicazioni delle reti neurali convoluzionali è comune inserire uno strato di Pooling tra due strati di convoluzione adiacenti. Da un punto di vista sostanziale, la funzione di uno strato di pooling è quella di ridurre in maniera progressiva la dimensione spaziale dello strato latente per ridurre la numerosità di parametri e quindi diminuire il rischio di sovradattamento del modello ai dati. Come quello di convoluzione, esso preserva la terza dimensione dell'input lavorando solo sulle prime due. La forma più comune di strato di pooling consiste in filtri di dimensione 2x2 (andare oltre porterebbe ad una perdita di informazione troppo elevata) che, fissato un certo stride, effettuano un'operazione di massimo. In Figura 2.11 si riporta un esempio di questo tipo.

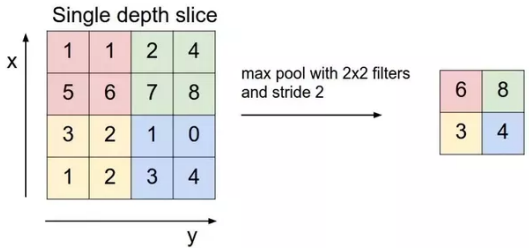


Figura 2.11: *Pooling Layer*

Naturalmente, la scelta della funzione da applicare al filtro è arbitraria e uno può scegliere di fare operazioni di media, preferita in un primo momento dai più, o ancora la norma. Tuttavia, si è riscontrato da un punto di vista pratico che l'operazione di massimo lavora meglio rispetto alle altre due ed è per questo preferibile.

Per il modo in cui agisce, a volte questo strato viene definito strato di sottocampionamento, e da un punto di vista applicativo ci sono principalmente due motivazioni per usarlo:

- L'operazione di pooling riduce la dimensione della matrice iniziale cercando di mantenere l'informazione rilevante. Pensando ad un'applicazione nel Natural Language Processing, si può immaginare che ogni filtro serva ad identificare una particolare parola o una sequenza di parole. Quando il filtro viene fatto scorrere, esso assumerà valori elevati presumibilmente dove la parola è presente e valori bassi laddove questa non lo è. Quindi, l'operazione di pooling per come è costruita fa perdere intuitivamente l'informazione su dove si trovi effettivamente la parola, concentrandosi soltanto sulla sua presenza.

- Un'altra proprietà, più inerente al riconoscimento di immagini, è relativa al fatto che si dimostra che il pooling determini un'invarianza dell'output rispetto a traslazioni, spostamenti e rotazioni dell'immagine. L'idea è che se ad esempio ruotiamo l'immagine di qualche pixel, o la trasliamo, l'operazione di pooling farà si che l'output sia approssimativamente lo stesso e questo grazie all'operazione di massimo che restituirà lo stesso valore.

Nella pratica, a differenza dello strato di convoluzione, quello di pooling non è cosi rilevante in termini di informazione utile per la previsione, anzi per alcuni è addirittura deleterio visto che porta ad una compressione dello strato, e in un certo senso ad un suo snaturamento. Per questa ragione molti dei modelli più recenti si basano principalmente su strati di convoluzione ripetuti, magari con uno *stride* maggiore, in modo da diminuire il rischio di sovradattamento senza perdere eccessiva informazione.

2.1.9 ReLU

Molto spesso dopo ogni operazione di convoluzione ne viene effettuata una ulteriore chiamata ReLU, che sta per Rectified Linear Unit ed è un'operazione non lineare. La ReLU è un'operazione che viene applicata elemento per elemento, o, visto che siamo in un contesto di analisi di immagine, pixel per pixel. Essa di fatto sostituisce tutti i valori di pixel negativi con il valore 0. L'obiettivo della ReLU è quello di introdurre elementi di non linearità nella rete neurale convoluzionale, visto che le relazioni più complesse da cogliere sono proprio di quel tipo. Inoltre, bisogna tenere a mente che la convoluzione è un'operazione lineare, per cui introdurre delle funzioni che modellino le relazioni più complesse è in qualche modo necessario. Come per l'operazione di pooling, è possibile utilizzare delle alternative rispetto a quella ReLU, ad esempio una di quelle delineate nella sottosezione 2.1.5; tuttavia, da un punto di vista applicativo si è visto che la ReLU ottiene tipicamente buone performance e per questo motivo viene largamente utilizzata.

2.1.10 Fully Connected Layer

Lo strato Fully Connected rappresenta un elemento essenziale in ogni CNN, e l'espressione implica che ogni neurone nello strato precedente è connesso a tutti i neuroni dello strato successivo. Dal momento che gli output derivanti degli strati descritti precedenti rappresentano degli aspetti importanti dell'immagine, in un certo senso l'obiettivo di questo strato è quello di utilizzare quest'ultimi per classificare le immagini. La funzione di attivazione utilizzata nell'ultimo strato è la funzione *softmax*, la quale assicura che la somma delle probabilità sia pari ad uno. Ma non solo, infatti, data la sua forma essa consente anche di cogliere elementi di non linearità.

Capitolo 3

Costruzione del dataset

I dati utilizzati sono relativi a 240 studenti dell'Università degli studi di Padova. Di questi sono disponibili i profili Facebook, dai quali sono state estratte le foto e alcune variabili relative alla sfera "social" degli utenti. Inoltre, sono a disposizione una serie di variabili che descrivono il loro profilo psicologico.
Dal momento che buona parte delle variabili utilizzate nei modelli di previsione sono state create *ex-novo*, si è ritenuto necessario dedicare diverse sezioni del capitolo al modo in cui queste sono state costruite, in modo da avere anche una visione più chiara delle quantità che entrano in gioco nell'analisi.

3.1 Variabili che descrivono il profilo psicologico dell'utente

La presenza di questo gruppo di variabili rappresenta un elemento di innovazione non trascurabile che rende in qualche modo unico questo dataset. Infatti, in nessuno dei lavori svolti in precedenza si era riuscito a tener conto di informazioni legate alla sfera psicologica dell'individuo, e questo semplicemente per mancanza di dati. Tuttavia, si ritiene che tali variabili possano essere legate in qualche modo al processo generatore dei dati, in quanto è verosimile immaginare che i contenuti pubblicati da particolari tipi di personalità acquisiscano maggior popolarità rispetto ad altri.
Andando nel dettaglio, i dati per la ricerca sono stati raccolti nel periodo compreso tra novembre 2015 e aprile 2016, attraverso un questionario condiviso carta e matita compilato dagli studenti durante un'ora di lezione alla presenza dei ricercatori. Lo scopo dello studio era indagare il ruolo di alcune caratteristiche individuali nel predire l'uso problematico di internet e di Facebook, in particolare tra gli studenti universitari. Il comitato Etico della Ricerca Psicologica dell'Università di Padova ha fornito l'approvazione del protocollo. Per questa tesi non sono state prese in considerazione le risposte relative a tutti gli item del questionario, bensì solo quelle ritenute più importanti rispetto al problema in analisi. In particolare, sono stati utilizzati innanzitutto quelli relativi ai Big-Five, utilizzando come riferimento la scala contenuta nei lavori di Caprara et al., (1993) e Caprara et al., (1994). Essi rappresentano le cinque grandi dimensioni della personalità, e possono essere descritte dalle seguenti categorie:

- livello di espansività
- cortesia
- coscienziosità e precisione
- nevroticismo, inteso come poca stabilità emotiva e come incapacità di convivere con le proprie emozioni
- apertura mentale e interesse verso persone di altre culture

Un altro insieme di item che si è ritenuto utile utilizzare è relativo al concetto di *covitality*, intendendo con questo un benessere emotivo e sociale derivante dall'interazione di diversi aspetti rientranti nella sfera psicologica e sociale dell'individuo. In particolare, il concetto di *covitality* è stato operativizzato seguendo il lavoro di Furlong, (2014), costruendo 36 item (riportati per intero con le espressioni originali) che da un punto di vista semantico possono esser tuttavia riassunti in quattro classi.

- credere in se stessi (Belief-In-Self=Self-Efficacy+Self-Awareness+Persistence)
- credere negli altri (Belief-In-Others=School Support + Family Coherence + Peer Support)

- competenze emotive (Emotional Competence= Emotional Regulation + Empathy + Behavorial Self-Control)
- felicità (Engaged Living=Optimism+Gratitude+Zest)

Inoltre, è stato misurato uno score totale dei 36 item che rappresenta un indice di benessere psico-sociale generale.
Infine, l'ultimo item estratto riprende il concetto di autostima ed è stato operativizzato utilizzando la scala di Rosemberg, successivamente validata da Prezza et al., (1997).

3.2 Variabili legate alla sfera "social" dell'utente

Questo gruppo di variabili comprende quelle caratteristiche dell'utente che riflettono il suo grado di socialità e che si è ritenuto opportuno inserire in quanto esplicative della popolarità di un'immagine pubblicata; la loro scelta è stata condotta sia in base a considerazioni di natura personale che in base ai lavori precedenti presenti in letteratura. Andando nel dettagli, le variabili considerate sono:

- Sesso dell'utente, utilizzato sia da Totti et al., (2014) che da McParlane et al., (2014).
- Numero di amici nella rete. Questa rappresenta una variabile dimensionale di cui non si può non tener conto, visto che a parità di altre variabili una foto di un utente con più amici avrà ragionevolmente più commenti. Tale variabile è stata utilizzata allo stesso modo da Khosla et al., (2014), e in una versione discretizzata da McParlane et al., (2014).
- Proporzione di amici femmine nella rete Facebook dell'utente

3.3 Variabili legate alla foto

Passando infine all'ultimo gruppo di variabili, diciamo che queste riguardano quella che è la vera unità statistica di questo lavoro, ossia l'immagine. Gli indicatori considerati possono essere a loro volta suddivisi in 3 gruppi:

- Low-level features
- High-level features
- Other image features

3.3.1 Low-level features

Con Low-level features si intendono tutte quelle variabili costruite a livello di pixel e successivamente aggregate mediante una misura di sintesi. Per cominciare, sono state estratte le proporzioni di 23 colori principali presenti nell'immagine, con l'idea che particolari colori possano essere più attraenti di altri. Questo è stato fatto assegnando ad ogni pixel il colore che minimizza la distanza di Manhattan tra i valori del pixel sui canali RGB e i valori RGB dei 23 colori; in seguito, si è semplicemente contato il numero di pixel assegnati a ciascun colore calcolandone poi la proporzione per svincolarsi dalla dimensione dell'immagine. Per far questo sono state utilizzate le formule di conversione riportate in Tabella 1 a partire dai canali RGB per poi aggregare tali misure su tutti i pixel attraverso un'operazione di media.L'approccio utilizzato in questa tesi è lo stesso utilizzato da Khosla et al., (2014) per quanto riguarda la norma utilizzata, quella l_1; tuttavia rispetto al loro lavoro si è preferito considerare una numerosità di colori minore per garantire una maggior robustezza. Una metrica diversa è stata invece utilizzata in McParlane et al., (2014), dove si è preferita una norma l_2 e concentrarsi sui colori principali tralasciando le sfumature.
Dopo aver stimato la proporzione dei colori, si è passati ad estrarre delle misure di sintesi legate alla luminosità, al contrasto, e alla saturazione dell'immagine. Per far questo, sono state semplicemente utilizzate le formule di conversione a partire dai canali RGB (riportate in Tabella 3.1), effettuando tale operazione pixel per pixel per poi fornire una misura globale tramite una semplice operazione di media. Tali misure sono state ottenute in maniera analoga da Totti et al., (2014) per quanto riguarda la saturazione e la la luminosità, mentre per il contrasto questi hanno utilizzato un approccio più fine; brevemente, quello che facevano era definire un istogramma normalizzato della luminosità dei pixels, e prendevano come indice l'ampiezza dell'intervallo necessario per ricoprire una percentuale importante dell'immagine. L'idea è quindi che un intervallo grande riflette un'alta variabilità della luminosità dei pixels e quindi un contrasto elevato nell'immagine.

	Formula di conversione
Luminosità	$L = 0.5 \cdot max(\frac{R}{255}, \frac{G}{255}, \frac{B}{255}) + 0.5 \cdot min(\frac{R}{255}, \frac{G}{255}, \frac{B}{255})$
Saturazione	$S = \begin{cases} \frac{\Delta}{1-\lvert 2L-1 \rvert} \; se \; \Delta \not\equiv 0 \\ 0 \; altrimenti \end{cases}$ dove $\Delta = max(\frac{R}{255}, \frac{G}{255}, \frac{B}{255}) - min(\frac{R}{255}, \frac{G}{255}, \frac{B}{255})$
Contrasto	$C = 0.3 \cdot (\frac{R}{255}) + 0.3 \cdot \frac{G}{255} + 0.3 \cdot \frac{B}{255}$

Tabella 3.1: *Formule di conversione a partire dai canali RGB*

Andando avanti, le prossime misure partono dal presupposto che foto caratterizzate da pixel con un gradiente elevato possano riscontrare un apprezzamento maggiore rispetto a quelle che non lo hanno. Per tener conto di quest'aspetto, sono state estratte due misure di variabilità dei pixels sulla scala dei grigi: in particolare, il primo indicatore è stato costruito in maniera *naive* semplicemente calcolando la deviazione standard dei pixels sulla scala dei grigi. Tuttavia, poiché in questo modo non teniamo in alcun modo conto della struttura dell'immagine, è stato costruito un ulteriore indice che agisce in maniera locale. In pratica per ciascun pixel viene calcolata la deviazione standard rispetto ai pixels del suo intorno di Moore (Nord,Sud,Est,Ovest,Nord-Est,Nord-Ovest,Sud-Est,Sud-Ovest), e poi queste misure vengono aggregate semplicemente facendone una media. Per la sua natura locale, questo indice è in qualche modo simile all'Histogram of Oriented Gradient (HGO) introdotto da Tsai, (2010), il quale ha il vantaggio di restituire dei risultati più accurati e lo svantaggio di richiedere un maggior tempo computazionale. Data la numerosità del campione si è preferito quindi optare per una soluzione meno accurata ma più veloce da ottenere.
L'ultima low-level features ha a che fare col fatto che quando guardiamo un'immagine, uno dei fattori che è ritenibile essere parte integrante del giudizio che daremo su quest'ultima è dato dal suo grado di nitidezza; infatti, immagini nitide saranno probabilmente più apprezzate rispetto ad immagini offuscate e per questo motivo si è deciso di tener conto di questo aspetto. Uno dei metodi maggiormente utilizzati fino a qualche anno fa consisteva nell'effettuare la trasformata di Fourier sull'immagine per eliminare tutto il rumore e successivamente analizzare la distribuzione delle alte frequenze, ossia quelle frequenze che contribuiscono ad un cambiamento evidente nell'intensità del segnale. Da qui è immediato dedurre che un'immagine con poche frequenze alte possa essere considerata offuscata, perché da un punto di vista pratico mancano i bordi. Tuttavia, il problema con questa misura sta nel fatto che stabilire quanto debba essere bassa la frequenza delle alte frequenze è una scelta del tutto arbitraria, per cui si è preferito utilizzare una misura sintetica che si svincolasse da questo tipo di decisioni, vista la scarsa conoscenza sulla teoria dei segnali.

L'indicatore utilizzato deriva dunque da un lavoro di A.Robembrock (*Blur detection with OpenCV*), ideatore di PyImageSearch.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Figura 3.1: *Laplacian Kernel*

L'idea è quella di applicare un particolare Kernel all'immagine e successivamente considerare la varianza dei pixel sull'immagine trasformata. Fatto questo, se tale valore cade al di sotto di una certa soglia l'immagine può essere considerata offuscata, altrimenti no; il motivo per il quale questo metodo apparentemente semplice funziona è che il Kernel Laplaciano (Figura 3.1) per come è costruito riesce a mettere in evidenza le regioni di un'immagine che ne caratterizzano i contorni. Per cui, da un punto di vista pratico, si può concludere che se dopo tale trasformazione la varianza dei pixel è bassa l'immagine è offuscata.

3.4 High-level features

Con il termine High-level features si vogliono considerare tutte quelle informazioni che vengono ottenute guardando un'immagine nel suo complesso e quindi non più a livello di pixel. Per cogliere questo tipo di *pattern* sono state utilizzate delle reti neurali convoluzionali allenate che differiscono tra loro non solo nell'architettura ma anche per l'obiettivo che si prefissano. Infatti, le reti Vgg16 e Place365 rientrano nell'ambito della *classificazione d'immagini*, mentre la rete Yolo si occupa di *object detection*. Quest'ultima appare essere particolarmente utile dal momento che in un'immagine vi può essere più di un elemento e quindi si adatta bene al problema in esame.

3.4.1 Rete Vgg16

Negli ultimi anni sono state indette diverse competizioni che premiavano sostanzalmente il team che riusciva ad ottenere le migliori performance in termini di classificazione di immagini su larga scala. Una di queste, la ImageNet Challange 2014 (*http:/www.image-net.org/challenges/LSVRC/2014/*), ha visto trionfare un team che ha utilizzato una rete con un'architettura molto diversa da quelle vittoriose negli anni precedenti; infatti, il team Vgg (Simonyan, Zisserman e al., 2014) ha introdotto un'architettura più profonda, basata su un numero di strati maggiori e un notevole risparmio computazionale. Uno dei principali elementi di differenziazione è infatti rappresentato dalla dimensione dei filtri, che passano da una dimensione più elevata (tipicamente 7x7) ad una dimensione 3x3. Inoltre, tali filtri vengono fatti convolvere pixel per pixel, utilizzando dunque uno *stride* unitario. Ora, da un punto di vista analitico si può mostrare che due strati di convoluzione con un filtro di dimensione 3x3 corrispondono ad uno strato di convoluzione di dimensione 5x5, mentre tre strati di convoluzione sempre con un filtro di dimensione 3x3 corrispondono ad uno strato di convoluzione con un filtro di dimensione 7x7. Date queste equivalenze, può sembrare poco utile utilizzare un'architettura di questo tipo. In realtà, questa porta da un punto di vista pratico due tipi di guadagno, il primo computazionale e il secondo in termini di capacità predittiva del modello. Per quanto riguarda quest'ultima, bisogna ricordare che ad ogni strato di convoluzione segue uno ReLU, come mostrato in Figura 3.2, quindi aumentando il numero di strati aumenta anche la possibilità di cogliere relazioni più complesse.

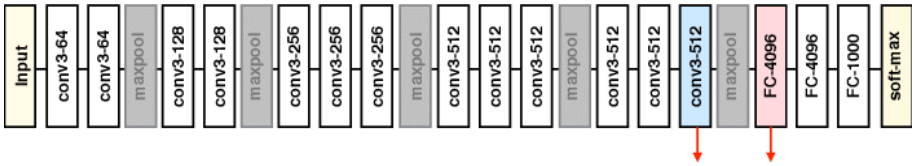


Figura 3.2: *Architettura Rete Vgg16*

Per quanto riguarda il primo discorso invece, tutto si risolve in un decremento del numero di parametri. Infatti, sempre considerando l'esempio di una serie di tre filtri di dimensione 3x3 e un filtro di dimensione 7x7, abbiamo che nel primo caso il numero di parametri è nell'ordine di $3 \cdot (3 \cdot C \cdot 2) = 27 \cdot C \cdot 2$; nel secondo caso invece, il numero di parametri è nell'ordine di $72 \cdot C \cdot 2 = 49 \cdot C \cdot 2$. Si tratta quindi di un guadagno che è circa dell'80%, un valore assolutamente non trascurabile visto l'alto grado di parametrizzazione delle reti neurali.
Un altro elemento caratteristico dell'architettura Vgg è rappresentato dall'introduzione di filtri di convoluzione di dimensione 1x1, che in generale vengono utilizzati per effettuare una riduzione nello spazio dei canali mentre qui lasciano inalterata la dimensione dell'output; il senso di quest'operazione sta nel fatto che dal momento che ogni strato di convoluzione è seguito da uno strato ReLU, l'inserimento di questi filtri consente di modellare ulteriori effetti non lineari.
Tra i diversi modelli proposti dal team "Vgg" è stato utilizzato il Vgg-16, il cui nome deriva dal numero degli strati di convoluzione utilizzati. Data la ovvia impossibilità di ottenere delle performance migliori del team allenando *ex-novo* la rete, sono stati utilizzati i pesi ottenuti allenando il modello su un sottinsieme del dataset Imagenet, che è stato utilizzato nell'ImageNet Large-Scale Visual Recognition Challange (ILSVRC). Da un punto di vista operativo si è deciso di non considerare soltanto la classe prevista come features, ma le probabilità derivanti dall'ultimo strato softmax. Sotto questo aspetto l'approccio utilizzato si differenzia rispetto a quello utilizzato in (Khosla et al., 2014), dal momento che in quest'ultimo si è utilizzata come *feature* il penultimo strato Fully-Connected della rete.
Dal momento che è di interesse valutare quali siano gli elementi di una foto che incidano sulla popolarità di quest'ultima, si è deciso di utilizzare delle $features$ che favoriscano l'interpretabilità.

3.4.2 Centralità

Con centralità si intende sostanzialmente la presenza di un elemento in primo piano nell'immagine. Si è deciso di tenere in considerazione di quest'aspetto perché si ipotizza che la presenza di un elemento principale nell'immagine possa attirare l'attenzione dell'utente portandolo ad un commento. Il modo attraverso il quale si è deciso di operativizzare questo concetto parte dal presupposto che empiricamente se la rete Vgg16 attribuisce un'immagine ad una classe con una probabilità elevata, il più delle volte è perché un elemento è in primo piano e quindi facilmente riconoscibile dal modello. Per questo motivo, dal punto di vista operativo si considera l'immagine un "primo piano" se la probabilità più alta assegnata ad una classe è maggiore di 0.90. Questo valore è stato scelto in maniera assolutamente arbitraria basandosi sull'esperienza avuta con le reti utilizzate. La ragione per quale non è stata effettuata una scelta rigorosa ricade nel tempo eccessivo che si sarebbe dovuto spendere per etichettare le immagini.

3.4.3 Place365

Negli ultimi anni il *task* della *Scene Recognition* è diventato sempre più importante alla luce le sue remunerative applicazioni in ambito commerciale. Un esempio classico è quello dei Social Media che potrebbero essere interessati ad individuare i luoghi in cui gli utenti si trovano con più frequenza, per poi rivendere tali informazioni a chi è interessato a fare pubblicità. Un aspetto chiave dello Scene Recognition è dunque quello di identificare il luogo nel quale gli elementi di un'immagine sono situati, che si tratti di un campo da calcio o di una cucina. Bisogna dire che questo non è l'unico approccio utilizzabile, perché in alternativa si potrebbe fare l'elenco degli elementi presenti nell'immagine e delle loro relazioni spaziali, o ancora semplicemente una lista. Tuttavia, l'approccio che viene usato nell'ambito della Scene Recognition ha il vantaggio innanzitutto di fornire una descrizione sintetica di un'immagine collocandosi su un adeguato livello di astrazione; inoltre, per come è costruita, elimina l'ambiguità derivante dalla mera elencazione di oggetti. A titolo di esempio, si può pensare ad un tavolo, una sedia e una lampada che possono far parte di un soggiorno, ma anche di un ufficio, o addirittura di un ristorante se ci soffermiamo solo sulle prime due.
Vista la specificità del problema, per gli sviluppatori è stato necessario costruire un dataset di training *ad hoc*, chiamato Places; esso è composto da quasi 10 milioni di immagini, ognuna di queste etichettate con una delle 434 categorie disponibili, le quali ricoprono circa il 98% dei luoghi in cui un uomo possa trovarsi. Successivamente, a partire da questo dataset sono stati estrapolati dataset minori, tra cui Place365 (Zhou et al., 2017) che è quello che è stato utilizzato in questa tesi. Come è intuibile dal nome, esso è composto non più da 434 ma da 365 categorie, che sono state scelte in base alla numerosità delle immagini in ciascuna classe, privilegiando quelle che potevano garantire un'accuratezza maggiore. Per quanto riguarda l'architettura della rete, gli sviluppatori hanno scelto le tre che in termini di performance hanno raggiunto i risultati migliori negli ultimi anni, ossia AlexNet, GoogLeNet e Vgg-16; in particolare, per questa tesi è stata utilizzata l'architettura Vgg-16 descritta in precedenza.
Operativamente, si è deciso di non considerare il vettore di probabilità come nel caso precedente ma semplicemente assegnare un'immagine alla classe per la quale risulta più alta la probabilità stimata, trattando quindi la variabile come un semplice fattore. Tuttavia, bisogna anche considerare che non tutte le immagini sono caratterizzate da un paesaggio (basti pensare ai ritratti) per cui si è deciso di assegnare un'immagine ad un luogo solo se la probabilità massima stimata è maggiore di 0.30; in caso contrario, l'immagine verrà considerata senza ambientazione. Una piccola parentesi va spesa per il criterio di scelta della soglia, che non è assolutamente rigoroso ma come nel caso precedente molto *naive*. La ragione che ha spinto in questa direzione ricade nel fatto che tali reti sono così ben allenate che se un'immagine ha effettivamente un'ambientazione la probabilità più alta sarà sempre elevata. Al contrario, se la rete è "indecisa" e quindi associa diverse probabilità non nulle a varie ambientazioni,tanto spesso è perché nella foto non c'è una reale ambientazione. D'altronde, per come è costruita essa è obbligata a collocare l'im-

magine in una delle categorie, per cui si è ritenuto opportuno effettuare una scelta di questo tipo.
Dal punto di vista della letteratura sono stati diversi i tentativi di tener conto dell'ambientazione dell'immagine per prevederne la popolarità; ad ogni modo, si ritiene che la rete Place365 rappresenti un passo in avanti significativo per questo scopo, soprattutto rispetto ai lavori precedenti. Infatti, McParlane et al., (2014) riducono il problema ad una classificazione di 5 classi, portando ad una perdita di informazione non trascurabile. D'altra parte, Khosla et al., (2014) nemmeno classificano l'ambientazione dell'immagine ma utilizzano un metodo molto più grezzo, le GIST. Queste descrivono un'immagine attraverso la definizione di contorni e forme, e sono quindi meno informative.

3.4.4 Indoor/Outdoor

Per determinare se una foto sia stata scattata Indoor o Outdoor si è utilizzato una regola euristica a partire dalla rete Place365. Quest'ultima come detto in precedenza riconosce la location di una foto ed oltre questo la classifica come Indoor/Outdoor. Ora, per avere maggior robustezza rispetto ad errori di classificazione, si è deciso di assegnare all'immagine l'ambientazione Indoor/Outdoor attraverso un voto di maggioranza dato dalle 10 classi con probabilità più alta, per cui se più di 5 classi vengono classificate come Outdoor l'immagine verrà etichettata come tale.

3.4.5 Yolo: You only look once

Il modello Yolo (Redmon et al., 2016) rappresenta al momento uno dei migliori in termini di performance per quanto attiene l'*object detection*. Esso si basa sulle reti neurali convoluzionali ed utilizza un approccio molto diverso (e in un certo senso più semplice oltre che veloce) rispetto ad altri che sono stati introdotti precedentemente. La Figura 3.3 fornisce un'idea del suo funzionamento.

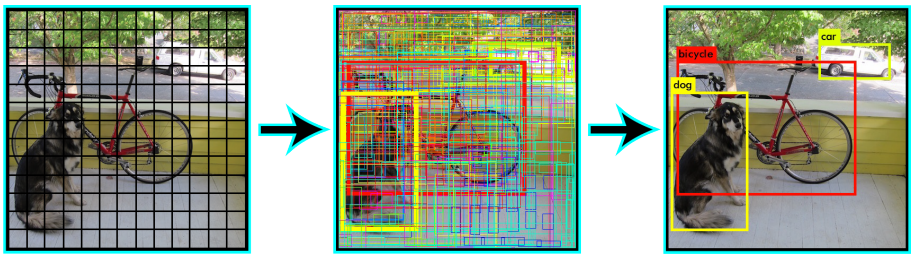


Figura 3.3: *Esempio di un'applicazione del modello YOLO*

Quello che tutti questi metodi hanno in comune è il fatto che non operano su tutta l'immagine ma utilizzano dei classificatori che agiscono localmente, ed è questo che consente sostanzialmente l'individuazione di più elementi nella stessa immagine. Il primo approccio proposto

per fare object detection è stato il modello DPM (Felzenszwalb et al., 2010) in cui abbiamo una finestra mobile che viene fatta slittare lungo tutta l'immagine, e successivamente in ognuna di queste finestre viene utilizzato un classificatore qualsiasi. Successivamente questo approccio è stato messo da parte, e questo perché sebbene i risultati fossero nella maggior parte dei casi accurati il costo computazionale era cosi esoso da rendere impossibile una sua applicazione nella pratica. La ragione per la quale il modello DPM è troppo lento ha a che fare col fatto che considera un numero di finestre troppo elevato.

Per questo motivo la DPM, così come altri metodi basati sullo stesso principio, ha ceduto il posto ad altri approcci, tra cui quello utilizzato nella R-CNN. Quello che differenzia la R-CNN dagli altri metodi è che qui le finestre vengono selezionate in maniera euristica, attraverso un algoritmo che d'altra parte oltre a restituire dei risultati accurati è estremamente efficiente da un punto di vista computazionale. Tale algoritmo è stato introdotto da Felzenszwalb, Huttenlocher e al., (2004) e affronta il problema di segmentazione dell'immagine impostandolo come un problema di grafo; in breve, all'inizio dell'algoritmo ciascun pixel rappresenta un nodo e tutti formano un grafo indiretto pieno, in cui il peso tra due archi rappresenta una misura di similarità tra i pixel. Questa misura tiene conto di diverse variabili quali intensità, colore, posizione ecc., per cui quello che l'algoritmo fa è da un lato aggregare progressivamente pixels (o gruppi di pixels) che hanno una similarità molto alta, e dall'altro definire dei contorni dell'immagine in modo che la similarità tra i pixels che si trovano in due aree diverse della frontiera sia molto bassa. Tale metodo è completamente data-driven, ed è questa la ragione per cui è estremamente efficace: infatti, se nell'immagine le componenti sono poche l'algoritmo restituirà poche componenti, con la conseguenza che le reti Neurali convoluzionali operarano su un numero di finestre minore, riducendo drasticamente il costo computazionale complessivo. Ora, malgrado le R-CNN abbiano rappresentato un netto miglioramento nel contesto dell'*object detection* grazie al loro approccio basato sulla segmentazione, c'è da dire che queste erano ancora troppo lente rispetto alla mole di immagini di cui ha bisogno una CNN per essere allenata, e il fatto che ogni immagine avesse bisogno fino a 40 secondi per essere analizzata ha portato all'esigenza di considerare altre strade.

Il modello YOLO differisce dalle R-CNN non solo per il modo nel quale vengono costruite le finestre, ma soprattutto perchè il modello lavora per costruire simultaneamente le probabilità di appartenenza alla classe e le finestre, in un'ottica che non è più sequenziale come nel caso precedente: infatti le fasi che compongono l'implementazione di una R-CNN, dall'estrazione delle finestre fino ad arrivare alla classificazione mediante SVM, avvengono una dopo l'altra, per cui l'obiettivo di Yolo è sostanzialmente ottimizzare tutto congiuntamente.

Da un punto di vista pratico significa che questa rete neurale ragiona in maniera globale su tutta l'immagine, ed identifica a partire da questo sia il contenuto che la finestra in cui giace ogni elemento riconosciuto. Operativamente, quello che il modello fa è dividere l'immagine in una griglia di dimensione fissata, tipicamente 7x7, e per ognuna delle celle definite viene applicata una rete neurale che ha un architettura molto simile a quella utilizzata da GoogleNet

(Figura 3.4).

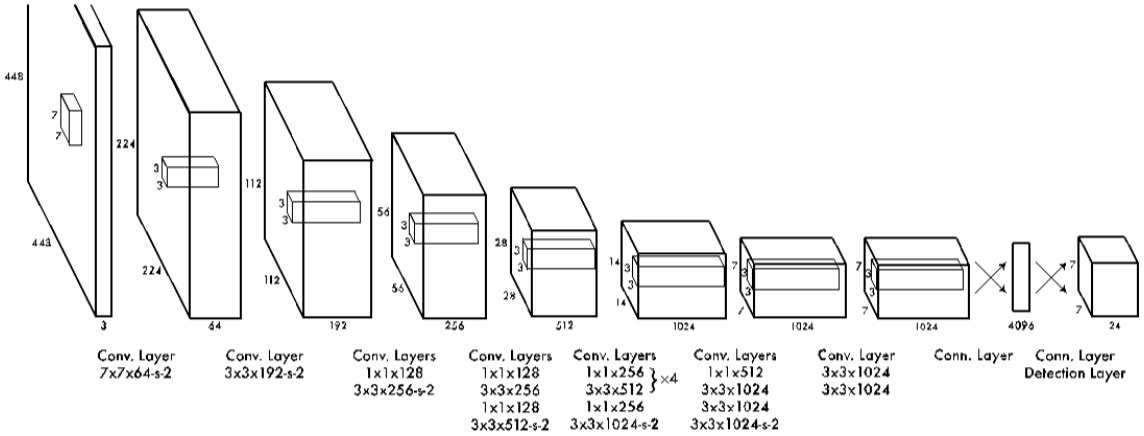


Figura 3.4: *Architettura del modello Yolo*

Per comprendere meglio il contesto nel quale ci stiamo muovendo e il modo nel quale questa rete lavora, riportiamo la funzione di perdita per una singola immagine (3.1):

$$\sum_{m=1}^{48} \lambda \times 1_i^{obj}((x_i - \hat{x})^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 + \sum_{c \in classes} ((p_i(c) - \hat{p}_i(c))^2) \quad (3.1)$$

dove:

- x_i e y_i rappresentano le coordinate del centro dell'elemento da identificare, mentre $\hat{x}_i$ e $\hat{y}_i$ le loro stime.
- w_i e h_i rappresentano invece la larghezza e l'altezza di tali elementi, e allo stesso modo $\hat{w}_i$ e $\hat{h}_i$ le loro stime.
- infine, p_i e la sua stima $\hat{p}_i$ definiscono le probabilità osservate e stimate di appartenenza a ciascuna classe.

Come quasi sempre accade, la funzione obiettivo è espressa in forma quadratica per facilitarne la massimizzazione, mentre è opportuno soffermarsi su due elementi, ossia la presenza di un fattore di scala λ e sul perché l'altezza e la larghezza delle scatole vengano minimizzate sotto radice.

Per quanto riguarda il primo aspetto, diciamo che questo fattore serve per dare maggior peso agli errori di localizzazione degli elementi e questo probabilmente perché in molti contesti non si è solamente interessati a che la rete identifichi gli oggetti correttamente ma anche al fatto che li identifichi nel posto corretto (basti pensare alle self driving car, identificare uno stop nel posto sbagliato potrebbe essere disastroso!). Per quanto riguarda il secondo discorso, questo ha più a che fare con la dimensione che gli elementi occupano nell'immagine, nel senso che è plausibile pensare di essere più tolleranti verso errori commessi quando l'elemento da riconoscere è grande invece che nel caso contrario. Chiarito questo, è immediato che effettuare una trasformazione in radice ha lo scopo di diminuire il peso dato agli elementi con delle finestre più ampie.
Ora, per ogni elemento della griglia oltre alle coordinate dell'elemento individuato e alla probabilità di appartenenza ad una certa classe, viene associata una misura di affidabilità del riconoscimento. Questa è data dalla probabilità che un elemento venga riconosciuto nella foto,riscalata per l'IoU, che in questo tipo di modelli assume un ruolo cruciale. L'IoU, riportato graficamente in Figura 3.5 e che sta per "Intersection over Union", rappresenta l'area di sovrapposizione tra la finestra previsto dal modello per un elemento e la quella vera, riscalata per l'unione delle due aree. Ora, se si pensa al modo nel quale è costruito il modello, è intuitivo

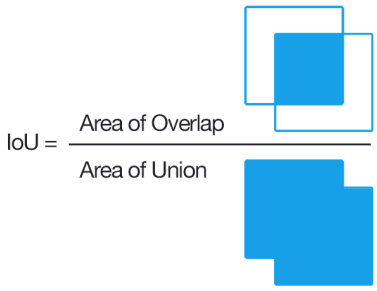


Figura 3.5: *IoU-Intersection over Union*

che lo stesso elemento possa essere identificato da più di una finestra; questo significa, in altre parole, che è necessario introdurre un modo attraverso il quale stabilire se un'identificazione debba essere considerata come falso positivo oppure no.

Questo avviene mediante una procedura in più step, che viene ben descritta in Figura 3.6:

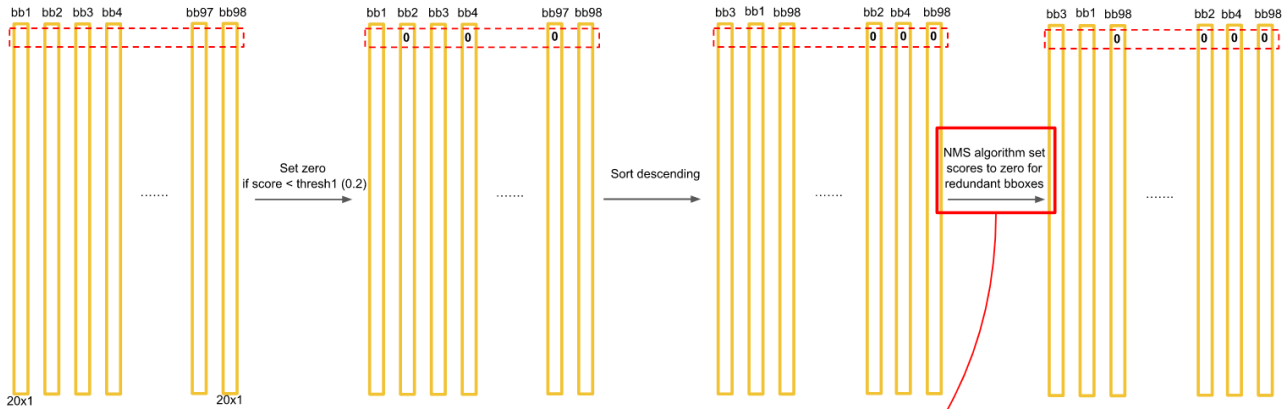


Figura 3.6: *Fase di pruning degli elementi individuati*

Come è possibile vedere, poichè a partire da ogni cella della griglia vengono definite due finestre, l'output di un modello di questo tipo sarà costituito da 98 vettori c-dimensionali, con c numero di classi. Ora, quello che si fa riga per riga, ossia classe per classe, è innanzitutto ordinare gli score in ordine decrescente, e mandare a 0 tutti quegli score che sono inferiori rispetto alla soglia prefissata. E' opportuno ricordare che tale soglia rappresenti un parametro di regolazione che gestisce il compromesso tra falsi positivi e falsi negativi; chiaramente, per valori elevati di questa soglia avremo meno falsi positivi e più falsi negativi, e viceversa, ma comunque la scelta del parametro ricade innanzitutto su quelli che sono gli obiettivi per i quali viene utilizzata la rete. A questo punto, sebbene la maggior parte dei punteggi sia adesso pari a 0, è necessario effettuare ulteriori operazioni sul vettore per limitare il più possibile la presenza di falsi positivi. Per far questo viene utilizzato l'algoritmo *Non-Maximum Suppression*, che sostanzialmente effettua un confronto ricorsivo a coppie tra la finestra a cui viene associato lo score più alto e tutti quelli non nulli. In particolare, si confronta il loro IoU, e se questo è >0.5 allora lo score associato alla finestra con score minore viene annullato.

3.5 Other image features

In questa categoria rientrano tutte le variabili che non vengono estratte direttamente a partire dal contenuto della foto, ma che riportano altri tipi di informazione. Esse sono illustrate di seguito:

- Stagione in cui è stata scattata l'immagine, come è stato fatto da Philip J. McParlane *et.Al* (2014).
- Città di provenienza
- Dimensione dell'immagine. Tale indicatore è stato costruito semplicemente andando a moltiplicare l'altezza e la larghezza dell'immagine. L'idea è che se una foto ha una risoluzione più alta potrebbe riscontrare maggiore popolarità tra gli utenti. Inoltre, per ridurre la scala della variabile, è stata effettuata una trasformazione logaritmica della stessa. La dimensione dell'immagine è stata tenuta in considerazione anche da Totti et al., (2014) e in una versione discretizzata anche da McParlane et al., (2014).
- Orientamento dell'immagine. Prendendo spunto da McParlane et al., (2014), anche tale variabile è stata costruita a partire da altezza e larghezza. In pratica queste vengono confrontate, e a seconda della loro relazione l'immagine viene assegnata alla classe "landscape","portrait" e "square". Il criterio di assegnazione è stato il seguente:
  - Se la larghezza è maggiore dell'altezza l'immagine viene classificata come "landscape"
  - Se l'altezza è maggiore della larghezza l'immagine viene classicata come "portrait"
  - Se altezza e larghezza hanno egual valore l'immagine viene classificata come "square"

Capitolo 4

Fase di analisi

Una volta costruito il dataset il passo successivo è stato procedere con l'analisi vera e propria, la quale, è stata indirizzata prevalentemente sul prevedere la presenza di almeno un commento nella foto più che il numero di commenti stessi. Dovendo scegliere su quale problema dedicarsi maggiormente per motivi di tempo, si è deciso di focalizzarsi sulla classificazione anche perché vi sono a disposizione misure quantitative che descrivono le performance dei modelli in maniera più immediata. In altre parole, è più semplice stabilire se un modello funziona bene o meno guardando l'*accuracy* piuttosto che il *Mean Square Error*. Come schema di lavoro si è deciso di suddividere il dataset in due gruppi, il primo di dimensione pari a 150000 e il secondo pari a 50000; in particolare, il sottocampione più grande è stato utilizzato per le analisi esplorative, la *feature selection* e la scelta dei parametri di regolazione dei diversi modelli di previsione, mentre il secondo soltanto come insieme di verifica per confrontare le performance di quest'ultimi.
Le successive sezioni sono organizzate nel seguente modo: la (4.1),(4.2) e (4.3) sono dedicate tutte alla classificazione e trattano rispettivamente le fasi di *analisi esplorativa*,*feature selection* e *tuning dei parametri*.
Infine, nella (4.4) si passa al secondo problema statistico affrontato in questa tesi, la previsione del numero di commenti. In particolare, oltre ai modelli utilizzati per la classificazione, vengono introdotti dei modelli particolarmente adatti dal punto di vista teorico al problema in esame, ossia i modelli "ZIP" e gli "Hurdle models".

4.1 Analisi Esplorative

Il primo passo di ogni analisi esplorativa è chiaramente quello di estrarre informazioni circa la distribuzione della variabile risposta. Come si può osservare dall'istogramma riportato in Figura 4.1 il numero di commenti alle foto ha una distribuzione decisamente asimmetrica, dal momento che la maggior parte delle immagini pubblicate non ricevono alcuna reazione dagli altri contatti in termini di commenti.

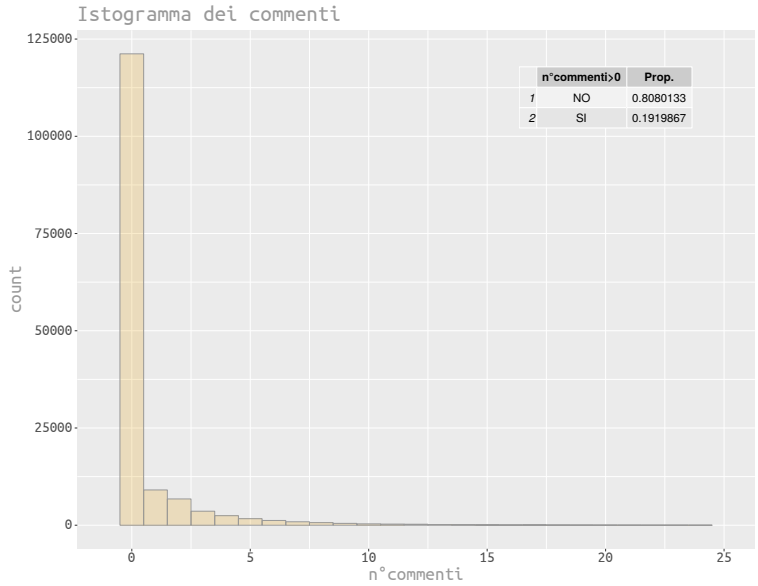


Figura 4.1: *Istogramma del numero di commenti*

Il fatto che la variabile di risposta sia di fatto una mistura tra una variabile degenere in 0 ed una di conteggio rende più problematica sia la stima dei modelli di classificazione che di regressione. Infatti, per quanto riguarda il primo contesto, se è vero che è piuttosto semplice ottenere un classificatore con un errore di classificazione globale basso (si pensi a un caso estremo di un modello che assegni banalmente ogni immagine alla classe '0 commenti'), dall'altro questo comporta il fatto di avere sempre dei falsi negativi troppo elevati. Questo è chiaramente un elemento di complicazione perché l'obiettivo è proprio quello di identificare i *pattern* che portano a commentare una foto, per cui detta in altri termini, siamo più interessati ad avere un errore di secondo tipo basso. Per tener conto di quest'aspetto si è deciso di considerare una valore soglia pari alla proporzione delle foto con commenti nel *training set*, che è pari a 0.18.
Descritta la variabile di risposta, si è poi passati ad analizzare le relazioni tra questa, ossia il numero di commenti in versione binaria e non, e le variabili esplicative. Come accennato in precedenza, il fatto di lavorare con una dimensionalità cosi elevata sia per quanto riguarda il numero di osservazioni che il numero di variabili rende complesso effettuare una canonica analisi esplorativa fatta di scatter plots e box plots. Questo sia perché si richiederebbe troppo tempo (il numero di variabili esplicative è pari circa a 1500) sia perché con 150000 osservazioni tutti gli scatter plots non sarebbero altro che nuvole di punti senza forma da cui è impossibile estrarre

qualsivoglia relazione. Per queste ragioni, si è cercato di effettuare un'analisi esplorativa mirata all'interpretabilità e ad un discreto grado di sintesi.
Da un punto di vista pratico, si è quindi deciso di concentrarsi prima sulle variabili categoriali derivate in fase di costruzione del dataset, poi su quelle continue, e infine sul numero di persone presenti nella foto individuate dalla rete Yolo.

4.1.1 Analisi esplorativa delle variabili categoriali

Le variabili categoriali derivate in fase di esplorazione del dataset sono il sesso, l'orientamento dell'immagine, la stagione in cui è stata scattata la foto, la presenza di un elemento di centralità dell'immagine, e infine l'appartenenza della foto alla categoria Indoor/Outdoor. Per descrivere la relazione intercorrente tra queste variabili e la variabile di risposta nella sua forma binaria si è fatto riferimento al mosaic plot. Di seguito si riportano alcuni pattern riscontrati e ritenuti interessanti:

- ***Sesso***

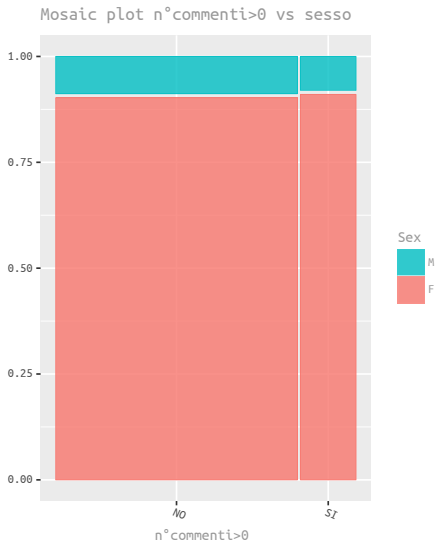


Figura 4.2: *Mosaic Plot n°commenti* > 0 *vs Sesso*

Osservando il mosaic plot in Figura 4.2 si può notare come le foto con commenti siano in proporzione leggermente più frequenti per le donne piuttosto che per uomini. Questo aspetto non è rilevante per quelli che sono gli obiettivi veri e propri dell'analisi, ma lo è perché è assolutamente importante cercare di contenere l'eterogeneità non osservata.

- ***Orientamento Immagine***

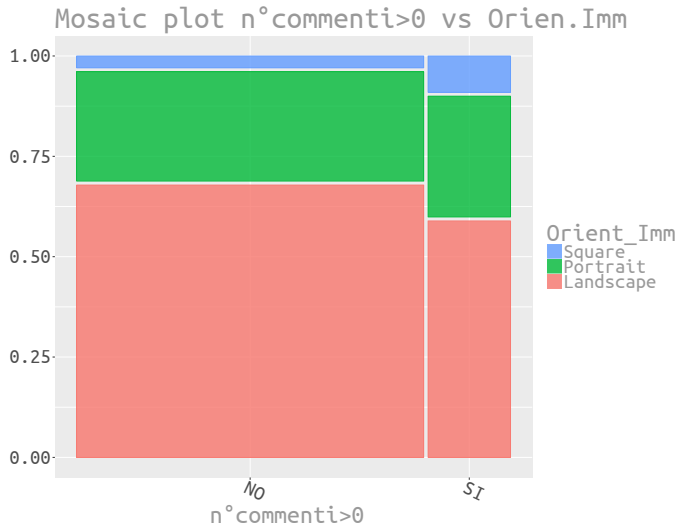


Figura 4.3: *Mosaic Plot n°commenti* > 0 *vs Orientamento Immagine*

Il mosaic plot riportato in Figura 4.3 mostra come un elemento che pare essere rilevante per la generazione di un commento sia dato dall'orientamento dell'immagine. In particolare, le foto di dimensione quadrata sembrano riscuotere un commento con maggior probabilità rispetto alle altre foto. Questo da un punto di vista interpretativo può essere legato al fatto che le foto modificate vengono spesso salvate con questo formato, per cui l'importanza che ha tale variabile sembra riflettere la presenza di particolari filtri applicate all'immagine.

4.1.2 Analisi esplorativa delle variabili continue

L'analisi esplorativa delle variabili continue è stata effettuata facendo riferimento a box plots in cui la variabile su cui si stratifica è la presenza di almeno un commento mentre la variabile dipendente è rappresentata da una delle variabili esplicative continue. Questo approccio, che magari da un punto di vista concettuale può sembrare scorretto in quanto stiamo invertendo il verso della relazione tra le variabili, porta con sé il vantaggio di portare a dei risultati interpretabili, cosa che non sarebbe possibile come detto in precedenza se facessimo riferimento agli scatter plots. Come nel caso precedente, riportiamo di seguito quelli che sono i pattern ritenuti più interessanti:

- ***Livello di Saturazione***

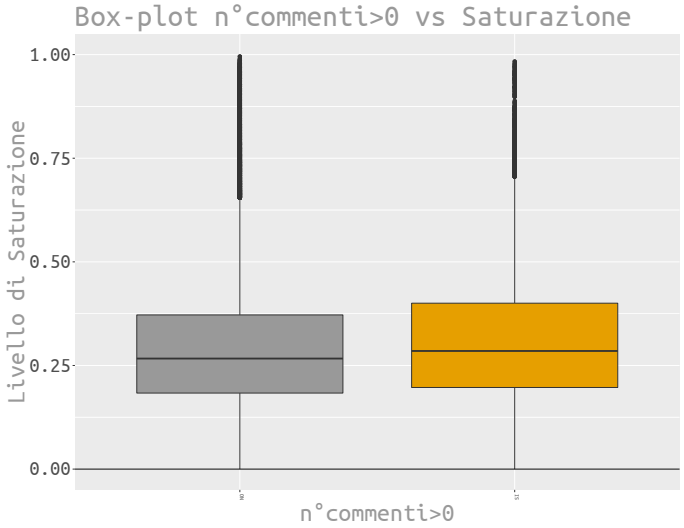


Figura 4.4: *Box Plot n°commenti* > 0 *vs Liv.Saturazione*

Dall'osservazione del Box-plot (Figura 4.4) appare che la mediana del livello di saturazione dell'immagine sia leggermente più alta per le immagini che ricevono commenti. Inoltre, si nota come il range di variazione del livello di saturazione sia traslato verso alto per le immagini che ricevono almeno un commento rispetto a quelle che non lo ricevono. Ciò da un punto di vista pratico significa che le immagini con colori più accesi sono mediamente preferite.

- ***Livello di Varianza sulla scala dei grigi***

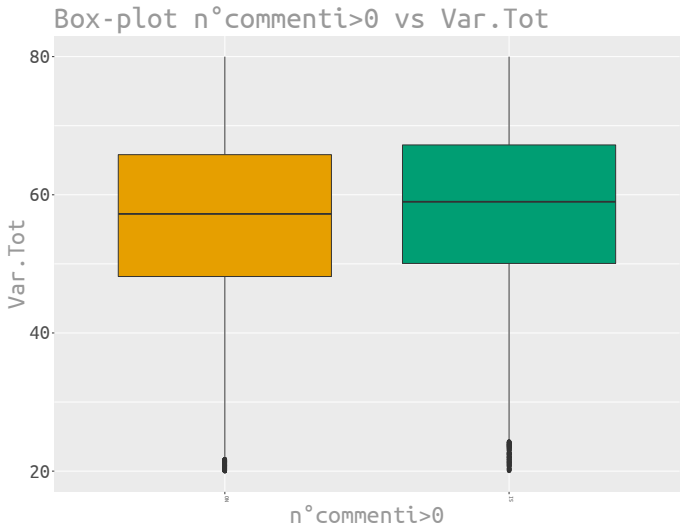


Figura 4.5: *Box Plot n°commenti* > 0 *vs Var.Pixels*

Il Box-plot in Figura 4.5, che descrive la variabilità di un'immagine sulla scala dei grigi a seconda che vi sia o meno un commento, suggerisce come le foto che ricevono un commento siano caratterizzate in mediana da una dispersione maggiore in termini di pixel. Se si ricorda il modo nel quale è stato costruito l'indice, questo da un punto di vista pratico significa sostanzialmente che le foto con particolari filtri o foto senza filtri che riproducano particolari paesaggi abbiano una propensione maggiore a ricevere un commento.

- ***Corrplot*** *n°commenti* ***e colori***

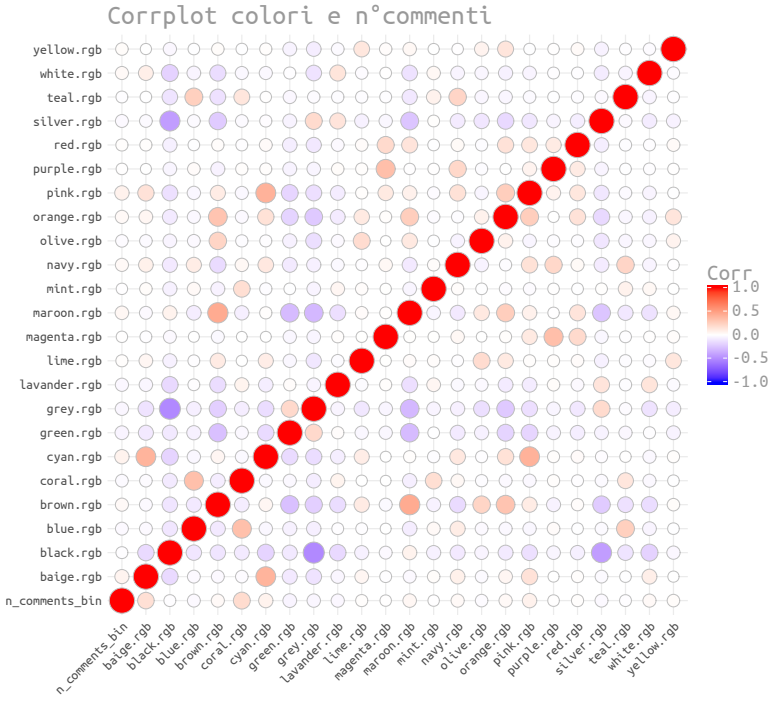


Figura 4.6: *Heat map di correlazione tra colori e n°commenti* > 0

La matrice di correlazione riportata in Figura 4.6 esplora le relazioni intercorrenti tra la presenza di particolari colori in un'immagine e il numero dei commenti. In particolare, si osserva una correlazione discretamente alta tra colori vicini nella scala RGB, come accade per le tonalità di marrone e di blu. Per quanto riguarda invece le relazioni tra i colori e la variabile di risposta, non si osservano pattern interessanti tranne che valori di ρ intorno allo 0.15 per quanto riguarda il ciano e il rosa. Naturalmente come detto in precedenza potrebbe tranquillamente trattarsi di una relazione spuria, dovuta, per esempio, al fatto che il ciano è il colore del mari più belli, per cui potrebbe essere l'ambientazione più che il colore a generare il commento.

4.1.3 Analisi esplorativa di alcune variabili high-features

La fase finale delle analisi esplorative si è concentrata sulle relazioni intercorrenti tra la variabile di risposta e i contenuti dell'immagine estratti a partire dalle reti Vgg16, Place365 e Yolo. Chiaramente, dal momento che le variabili in questione sono più di 1000, si è deciso di concentrarsi solo sul numero di persone individuate nelle immagine dalla rete Yolo e lasciare che sia la stima dei modelli a dare una visione completa del resto grazie all'indice di importanza delle variabili che diversi modelli forniscono.

- *n°**persone nella foto***

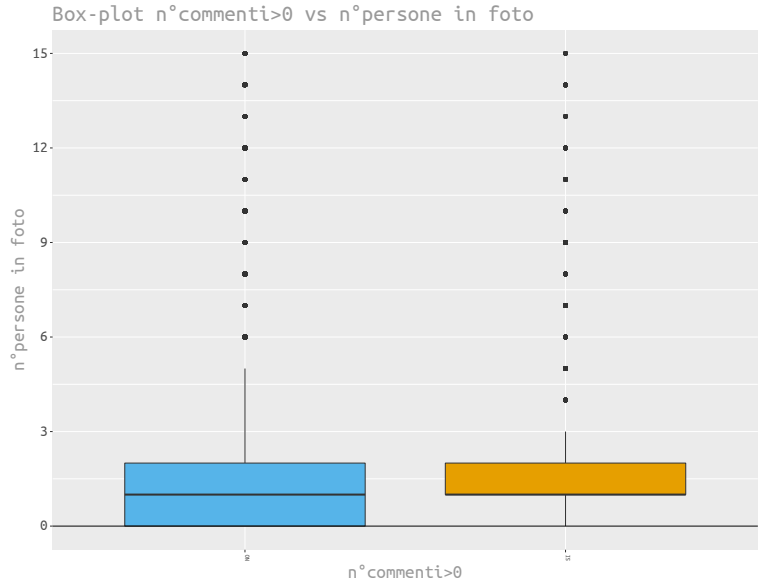


Figura 4.7: *Box Plot n°commenti* > 0 *vs n°persone in foto*

Dall'osservazione del boxplot in Figura 4.7 si osserva che la mediana delle persone sia pari ad uno sia per le foto che non ricevono commenti sia per le foto che li ricevono. Tuttavia, se ci condizionamo alle foto che ricevono commenti, vediamo che la distribuzione delle persone presenti in foto sia asimmetrica a sinistra nel senso che se vi è assenza di persone nella foto più probabilmente si tratta di una foto senza commenti.

4.1.4 Trattamento delle variabili categoriali

Un aspetto sul quale si ritiene valga la pena soffermarsi riguarda la codifica delle variabili categoriali. Tale fase, banale se si utilizza un modello lineare dal momento che bisogna semplicemente scegliere la codifica in base all'interpretazione che si vuol dare ai parametri, può diventare invece rilevante quando si utilizzano altri modelli. Infatti, il primo aspetto da tenere a mente è che i modelli appartenenti alla famiglia del Gradient boosting, l'SVM e anche le reti neurali multistrato lavorano solo con variabili quantitative come esplicative; d'altra parte, anche i modelli basati sul principio dell'*ensamble* di alberi fanno molta fatica se il numero di livelli delle variabili categoriali è elevato. Questo perché da un punto di vista operativo gli alberi 'splittano' le variabili qualitative andando a cercare quella partizione che determina la maggior riduzione della funzione di perdita. Quindi, è evidente che il numero di possibili split cominci a diventare ingestibile al crescere del numero di livelli e questo è il motivo per il quale diversi pacchetti tendono a porre un veto fissandone un valore massimo ammissibile. Alla luce di quanto detto, si è deciso di utilizzare per la maggior parte delle variabili un tipo di codifica che viene detta *one-hot encoding*, che semplicemente definisce un numero di colonne pari al numero di livelli, assegnando un unico valore pari ad uno in corrispondenza del livello registrato e ponendo il resto dei valori pari a zero. Questo tipo di codifica è evidentemente una codifica che rende il rango della matrice delle esplicative non pieno, ed è il motivo per il quale tale codifica viene sostituita nei modelli lineari classici da altri tipi di codifiche come quella ad angolo, additiva, ecc.
Un'altra tipologia di codifica è stata applicata alla variabile che descrive la città natale dell'utente, ed è stata implementata attraverso il pacchetto R "*text2vec*". Dato l'elevato numero di livelli, l'idea è quella di trasformare tale variaible in una variabile quantitativa sostituendo alle varie modalità le frequenze con cui queste si manifestano. Questo tipo di codifica è ampiamente utilizzata nelle competizioni Kaggle dove il numero di livelli di una variabile categoriale può andare ben oltre le migliaia; tipicamente essa funziona bene nei modelli basati sul principio dell'*ensamble* perché quello che fa è una sorta di aggregazione supervisionata delle diverse modalità. In questo caso, eventuali split dividerebbero probabilmente utenti del Veneto e utenti provenienti da altre regioni o Stati.

4.2 Feature Selection

La fase di Feature Selection assume un ruolo rilevante in tutti i problemi di previsione. La motivazione di questo non è soltanto quella filosofica legata al principio del rasoio di Occam, secondo la quale sia preferibile un modello più parsimonioso a parità di capacità esplicativa. Infatti, da un punto di vista statistico la presenza di variabili *noisy* aumenta il rischio di sovradattamento del modello ai dati, con la conseguenza che quest'ultimo avrà una performance previsiva peggiore, che sia un contesto di classificazione o di regressione. Naturalmente ci sono dei modelli più sensibili ad altri, nel senso che i modelli che fanno selezione automatica non forzando tutte le variabili ad esser dentro il modello riescono a gestire meglio il problema (si pensi agli alberi a titolo di esempio).
Ad ogni modo, quando la dimensionalità del problema è cosi elevata anche modelli di questo tipo fanno fatica, per cui si è ritenuto necessario implementare una procedura di selezione di variabili ad hoc. Il primo passo parte dalla considerazione che le variabili esplicative ottenute tramite le reti allenate descritte in precedenza (1397) rappresentino in buona parte rumore di fondo e quindi non esplicative nel prevedere se una foto riceverà o meno un commento. D'altra parte, è ragionevole pensare che le altre variabili siano legate in qualche modo al processo generatore dei dati, visto che o sono già state utilizzate precedentemente in letteratura oppure partono da considerazioni teoriche che a questo stadio possono esser ritenute valide. Alla luce di quanto detto, si è deciso di effettuare una prima procedura di *screening* solo sulle variabili ottenute a partire dalle reti allenate. Da un punto di vista pratico ci sono diverse strade per far questo, ad esempio guardare la correlazione tra la variabile di risposta e le altre e definire una soglia al di sotto della quale una variabile viene eliminata. In questa tesi si è deciso di considerare un'altra strada, ossia quella di effettuare semplicemente test $t - Student$ (con valore soglia del $p - value$ pari a 0.05) univariati in cui la variabile di stratificazione è proprio la presenza o meno di un commento. L'idea che sta dietro a questo primo grezzo step di selezione di variabili è che con un numero di osservazioni cosi elevate la potenza del test t è cosi elevata che anche effetti piccolissimi vengono individuati. Quindi, quel che stiamo facendo a questo stadio è semplicemente eliminare quelle variabili che rappresentano sicuramente rumore di fondo e quindi oltre a non essere utili per la previsione dei commenti addirittura peggiorano le performance dei modelli che verranno utilizzati successivamente. Questa prima fase di screening ha ridotto circa del 25% la numerosità delle variabili di rete, e questo rimarca il fatto che in generale ad un aumento consistente delle variabili a disposizione non corrisponde quasi mai un aumento dell'informazione utile. Ora, fatta questa operazione il passo successivo è stato quello di utilizzare una strategia di *feature selection* che si potrebbe definire "Dividi et Impera", dal momento che la selezione di variabili non viene fatta considerandole tutte contemporaneamente ma dividendole rispetto al loro contenuto semantico. Un approccio di questo tipo ha il difetto di non tener conto di possibili interazioni tra variabili, d'altra parte cosi come viene fatto per il MARS è plausibile ipotizzare che se una variabile non ha un effetto marginale sulla variabile di

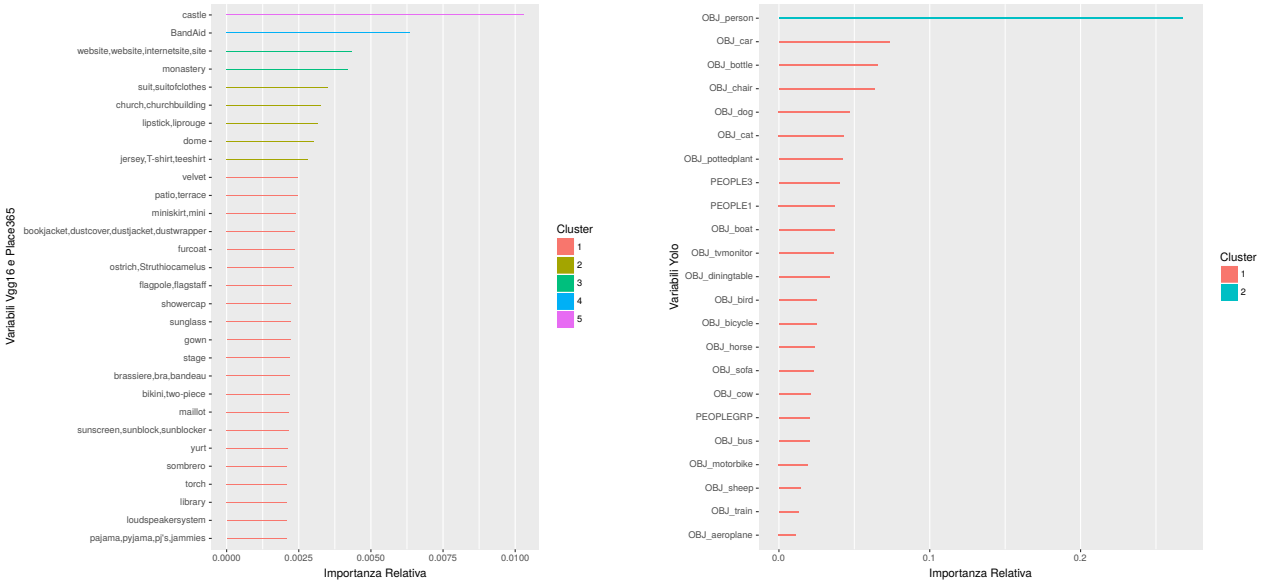
risposta, non avrà nemmeno un effetto di interazione con un'altra variabile esplicativa. Il fatto di utilizzare quest'approccio consente anche di avere maggior interpretabilità e di farsi un'idea più chiara dell'importanza relativa che ciascuna variabile assume tra i suoi "competitor". In particolar modo, le categorie definite sono:

- Variabili derivanti dalle reti neurali Vgg16 e Place 365, sia perché entrambe sono reti di classificazione sia perché presentano delle categorie in comune. Variabili derivanti dalla rete neurale di object detection Yolo.
- Variabili demografiche e variabili legate alla sfera social dell'utente.
- Variabili ottenute a partire dall'immagine e che descrivono caratteristiche di quest'ultime diverse dal contenuto.
- Variabili che descrivono il profilo psicologico dell'utente

Da un punto di vista metodologico, per fare selezione di variabili è stato utilizzato un XGBoost, che non nasce prettamente per questo scopo e non gode nemmeno delle proprietà teoriche di altri modelli come il LASSO. Tuttavia, la complessità del problema fa escludere che il modello generatore dei dati possa essere un modello lineare, e si preferisce sfruttare il fatto che come tutti i modelli basati su alberi anche l'XGBoost fornisca in maniera diretta una misura dell'importanza di ciascuna variabile. L'idea sulla quale si basa la costruzione dell'indice è che se una generica variabile viene coinvolta in molti split è perché questa ha verosimilmente una forte predittività rispetto alla variabile di risposta, per cui banalmente quello che si fa è andare a sommare tutti i *gain* (ossia riduzioni della funzione di perdita) che otteniamo quando questa viene coinvolta in uno split. Questo modo di procedere consente di definire un ranking delle variabili in termini di importanza relativa nella spiegazione della variabile di risposta, e di scartare le variabili poco rilevanti. In questa analisi si è deciso di eliminare una variabile se la sua importanza relativa è minore dell'1% per le variabili ottenute a partire da una rete neurale e del 2% per tutte le altre categorie, per tener conto dell'eterogeneità in termini di dimensione presente tra i gruppi.
Un appunto che vale la pena fare riguarda la scelta dei parametri di regolazione. Dato che a questo stadio si sta valutando quali possano essere le variabili rilevanti mettendo in secondo piano le performance previsive del modello, questi sono stati scelti secondo il buon senso con l'idea che le variabili da inserire del modello siano sostanzialmente le stesse al variare dei parametri di regolazione, a patto che questi vengano scelti in maniera ragionevole. Per questa tesi, è stato considerato un *learning rate* pari a 0.3, albero poco profondi come è comune nel boosting con una numerosità di nodi pari a 6. Inoltre, si è deciso di non considerare come candidate ad ogni split tutte le variabili ma solo il 25%, per esplorare in maniera esaustiva lo spazio delle variabili. Nelle seguenti sottosezioni verrà trattata nel dettaglio la *features selection* per ciascun gruppo di variabili esplicative, riportando l'importance plot e descrivendo le variabili più rilevanti.

4.2.1 Variabili ottenute a partire dalle Reti Neurali Allenate

Cosi come nello step precedente le variabili ottenute tramite le reti Vgg16,Yolov2 e Place365 sono sicuramente quelle maggiormente coinvolte in questa seconda fase di *Feature Selection*; la motivazione alla base di questo sta sostanzialmente nel fatto che tali reti sono allenate per prevedere moltissime classi, ed è ragionevole pensare che non tutte siano legate al processo generatore dei dati. Inoltre, ci si può aspettare che i $test - t$ univariati non abbiano eliminato tutte le variabili Noisy. La parziale inefficacia dell'operazione svolta ha a che vedere con la numerosità campionaria estremamente elevata che accentua la tendenza a rifiutare l'ipotesi nulla, e ciò rende necessario un'ulteriore fase di selezione di variabili.



(a) Variabili Vgg16 e Place365

(b) Variabili Yolo

Figura 4.8: *Importance Plot per le variabili relative al contenuto dell'immagine*

Osservando il primo importance plot riportato in Figura 4.8 e relativo alle reti neurali Vgg16 e Place365 si nota ai primi posti la presenza di particolari ambientazioni dell'immagine. In particolare, le dieci variabili con importanza relativa più alta comprendono luoghi legati alla sfera religiosa come chiese e monasteri ed altre principalmente connesse al lusso come attici e suit. D'altra parte, è possibile identificare un altro cluster di elementi rilevanti associabili all'abbigliamento, soprattutto femminile; in questo senso, si evidenzia la presenza di minigonne, bikini, e abbigliamento intimo.

Passando al secondo grafico, si osserva che un elemento forte che evoca un commento è rappresentato chiaramente dalle persone. Selfie, foto di gruppo, o semplici scatti rubati sembrano assumere un ruolo di primo rilievo in questo contesto. Seguono due elementi riconducibili in qualche modo alla sfera del lusso, ossia macchine e bottiglie (difficile ipotizzare si tratti di bottiglie d'acqua!). Infine, sempre tra le prime posizioni si trovano animali domestici come cani e gatti, non a caso presenti spesso in pubblicità di ogni tipo.

Da un punto di vista della *feature selection*, utilizzato il criterio definito in precedenza sono state eliminate 289 variabili delle 712 presenti all'inizio di questa fase.

4.2.2 Variabili demografiche e variabili "Social"

Questo rappresenta da un punto di vista quantitativo il gruppo più piccolo di covariate ma che al cui interno vi è comunque informazione utile per il problema in esame.

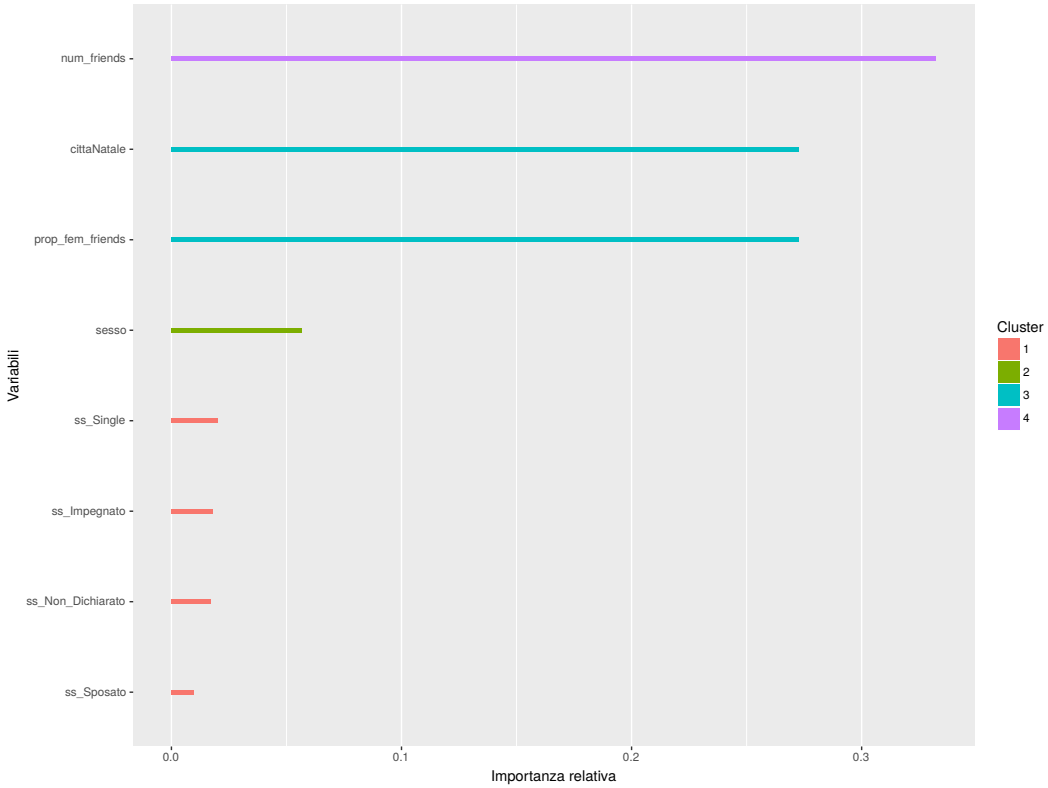


Figura 4.9: *Importance Plot per variabili legate all'immagine*

In particolare, come si può osservare dal grafico riportato in Figura 4.9 la variabile più rilevante è rappresentata dal numero di amici presenti nella rete, e questo ha senso dal momento che al netto delle altre variabili le foto pubblicate da un utente con tanti amici riceveranno più commenti. Considerazioni interessanti possono esser fatte anche rispetto alla seconda variabile, ossia la città natale dell'utente. Infatti, ricordando che è stata codificata assegnando ad un livello la frequenza con la quale questo si manifesta, si potrebbe ipotizzare che le foto pubblicate da utenti che nascono e vivono nelle città principali del Veneto abbiano un comportamento diverso rispetto agli altri; questo da un punto di vista sociologico potrebbe esser legato ad un discorso di integrazione. Infine, tra le variabili legate allo stato sentimentale, che comunque nel complesso sembrano esser le meno rilevanti rispetto alle altre, spicca lo status di single. Questo potrebbe riflettere la propensione degli individui, in particolar modo degli uomini, a manifestare apprezzamenti verso persone non impegnate sentimentalmente.

4.2.3 Variabili legate al profilo psicologico

La caratteristiche individuali che sono maggiormente importanti in questo contesto sembrano essere alcuni tratti di personalità relativi agli stili di interazione con gli altri e con il mondo (estroversione, coscienziosità, e apertura mentale) e alcuni tratti e competenze relativi alla sfera emotiva. In particolare, sembra che essere stabili emotivamente (scala personalità) ed essere in grado di gestire le proprie emozioni siano variabili che contribuiscono a spiegare variabilità.

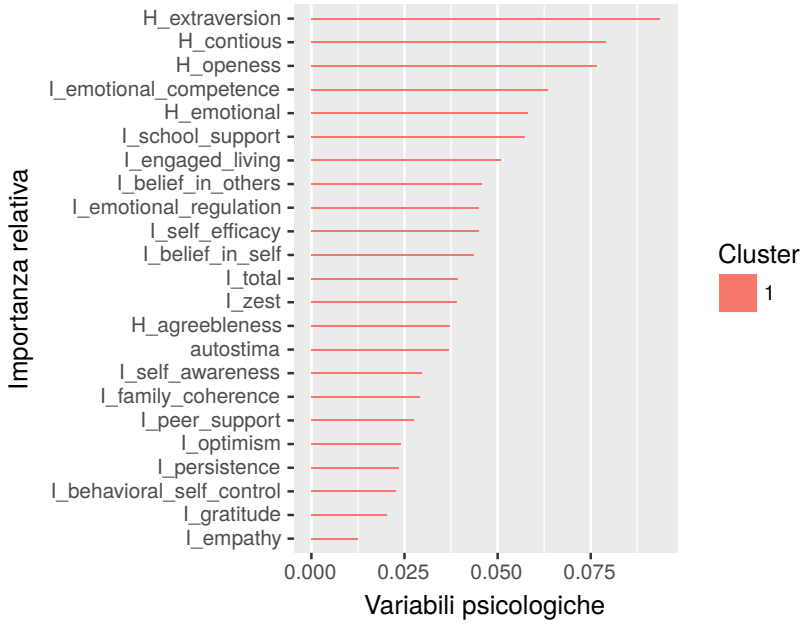


Figura 4.10: *Importance Plot per variabili legate al profilo psicologico*

Altri risultati interessanti sono la relativa minore importanza dell'autostima (che in letteratura è una delle variabili più studiate con Facebook) e delle variabili relative alla sfera delle relazioni sociali (ad esempio il supporto dei pari).

4.2.4 Altre caratteristiche dell'immagine

Come accennato in precedenza tutte le variabili appartenenti a questo gruppo sono state definite o in base a considerazioni di natura teorica oppure perché sono state utilizzate precedentemente in letteratura. Ed infatti, come ci si poteva aspettare,si riscontra che praticamente tutte le variabili hanno un'importanza relativa maggiore della soglia prefissata. Dall'importance plot riportato di seguito in Figura 4.11 si può osservare come sostanzialmente tali variabili siano suddivisibili rispetto alla loro importanza relativa in tre clusters.

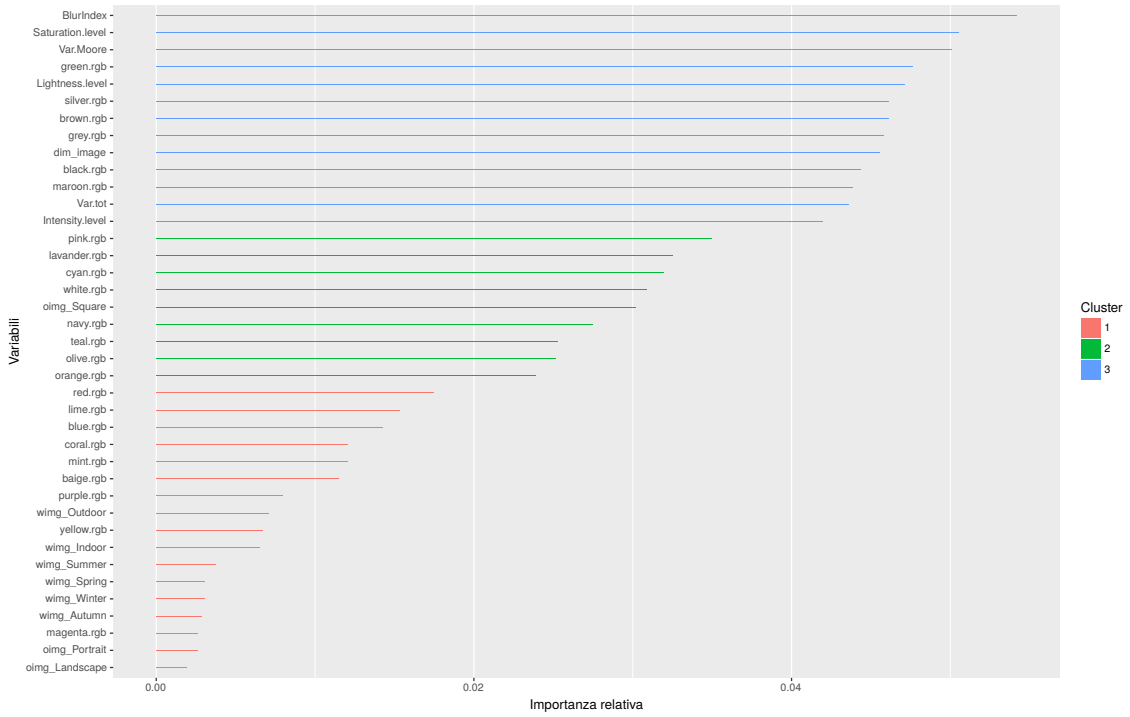


Figura 4.11: *Importance Plot per variabili legate all'immagine*

In particolare, il primo comprende luminosità, saturazione, intensità, nitidezza e varianza dei pixels, quindi tutte variabili che assumono valori elevati o in presenza di particolari ambientazioni (si pensi alle luci di un tramonto) o in presenza di un filtro artificiale applicato all'immagine. Il secondo gruppo di variabili comprende di fatto la proporzione di colori presente nell'immagine, con il rosa che spicca su tutti; si può pensare che ciò sia dovuto ad una correlazione spuria visto che è legata alla presenza di persone della foto. Interessante anche la presenza dell'azzurro che viene etichettato in diversi studi come un colore contemplativo, cioè che si presta ad essere osservato per un tempo elevato.
Infine, nel terzo cluster, quello in termini relativi meno importante, notiamo la presenza di variabili che non esprimono direttamente delle caratteristiche legate alla foto, come la stagione in cui è stata scattata la foto e il suo orientamento; rispetto a quest'ultima variabile soltanto le foto di dimensione quadrata sembrano avere un importanza discretamente elevata.

4.2.5 Confronto tra i modelli parziali

In questa breve sottosezione si effettua un confronto tra i modelli parziali, ossia ai modelli stimati per ciascuno dei gruppi definiti in precedenza. Nelle precedenti sottosezioni si è rivolta l'importanza principalmente ad aspetti descrittivi per avere delle prime indicazioni sul ruolo che hanno le singole variabili dei vari gruppi all'interno del modello; tuttavia, è importante fornire anche una misura legata alla capacità previsiva dei vari gruppi di variabili. Questo *in primis* per avere un'indicazione su quello che è il punto di partenza, dal momento che, vale la pena ricordarlo, nessun tuning è stato effettuato sull'XGBoost e che il modello utilizzato è stato scelto in base a considerazioni di natura qualitativa. Inoltre, è utile capire anche quali siano i contributi marginali dei diversi tipi di variabile nello spiegare la variabile di risposta.

	$1 - accuracy$	fpr	fnr
mod_car_imm	0.41	0.77	0.14
mod_demo_soc	0.38	0.73	0.12
mod_psi	0.37	0.73	0.12
mod_reti	0.41	0.75	0.12

Tabella 4.1: *Tabella di confronto modelli parziali*

Osservando la Tabella 4.1 appare intanto evidente come nessuno dei modelli abbia delle performance soddisfacenti non soltanto in termini di errore di classificazione totale ma anche di errori di primo e di secondo tipo. Confrontando poi i diversi modelli si osserva che quelli che spiegano in maniera migliore la variabile di risposta siano quelli basati da un lato sulle caratteristiche psicologiche dell'utente e dall'altro sulle sue caratteristiche socio-demografiche. Questo da un punto di vista concettuale non stupisce più di tanto, ed inoltre tali considerazioni trovano riscontro anche nei lavori di Khosla et al., (2014) e Totti et al., (2014) in cui queste $features$ assumono un ruolo rilevante.

4.3 Tuning dei parametri

Questa sezione è dedicata al metodo di scelta dei parametri di regolazione in ciascun modello. Tale fase è particolarmente importante dal momento che ognuno dei modelli utilizzati (tranne il Bagging) ha al suo interno uno o più parametri che governano il *trade-off* tra varianza e distorsione.
Dal momento che il campione è sbilanciato il criterio di scelta dei modelli non si è basato soltanto sull'errore di classificazione globale, bensì si è tenuto conto anche degli errori di primo e di secondo tipo.

4.3.1 Random Forest

Il parametro di regolazione della Random Forest è rappresentato dal numero di variabili che fanno da *competitors* in ogni split, mentre il numero di alberi utilizzati come accennato in precedenza non rappresenta un parametro rilevante.

Mtry	1-accuracy	fpr	fnr
79	0.35	0.29	0.36
61	0.35	0.29	0.37
512	0.35	0.32	0.36
392	0.35	0.32	0.36
46	0.36	0.28	0.37
135	0.36	0.30	0.37
300	0.36	0.32	0.36
176	0.36	0.31	0.37
36	0.36	0.28	0.38
103	0.36	0.29	0.37
230	0.36	0.30	0.37
27	0.37	0.28	0.39
21	0.38	0.27	0.40
16	0.39	0.26	0.42

Tabella 4.2: *Tuning Random Forest*

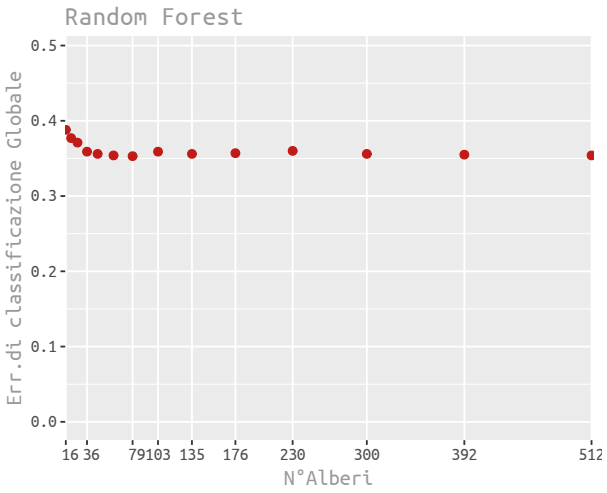


Figura 4.12: *Tuning Random Forest*

Osservando la Figura 4.12 che descrive come varia l'errore di classificazione al variare del numero di variabili candidate ad ogni split, si nota un andamento decrescente di tale errore, che tende a stabilizzarsi già al valore 61. Il fatto di non dover utilizzare una proporzione delle variabili troppo elevata riflette il fatto che non vi siano variabili portatrici di gran parte dall'informazione ma che questa si distribuisca in maniera omogenea tra le variabili.
In particolare, guardando la Tabella 4.2, si sceglie come valore del parametro di regolazione 79

dal momento che fornisce l'accuracy più elevata e il miglior bilanciamento tra errore di primo e di secondo tipo.

4.3.2 Ridge

Il GLM regolarizzato a norma l_2 è stato introdotto principalmente come *benchmark* per avere un'idea di quanto i modelli selezionati per questa tesi facciano meglio rispetto ad un modello semplice come il modello lineare. La scelta della penalizzazione è stata dettata dalla natura delle variabili esplicative che rendevano la matrice X a rango non pieno e che quindi rendeva indeterminata la stima dei coefficienti del modello. Il parametro di regolazione λ è stato scelto tra 100 valori uniformemente distribuiti tra 0 e 0.3.

Lambda	1-accuracy	fpr	fnr
0.001	0.404	0.313	0.424
0.003	0.404	0.313	0.424
0.007	0.404	0.313	0.425
0.010	0.406	0.311	0.427
0.013	0.406	0.312	0.427
0.023	0.407	0.310	0.429
0.017	0.408	0.310	0.430
0.020	0.408	0.310	0.429
0.027	0.408	0.310	0.430
0.033	0.408	0.309	0.431
0.030	0.409	0.307	0.432
0.040	0.409	0.308	0.431
0.043	0.410	0.307	0.433
0.050	0.410	0.307	0.433
0.037	0.411	0.305	0.434

Tabella 4.3: *Tuning Ridge*

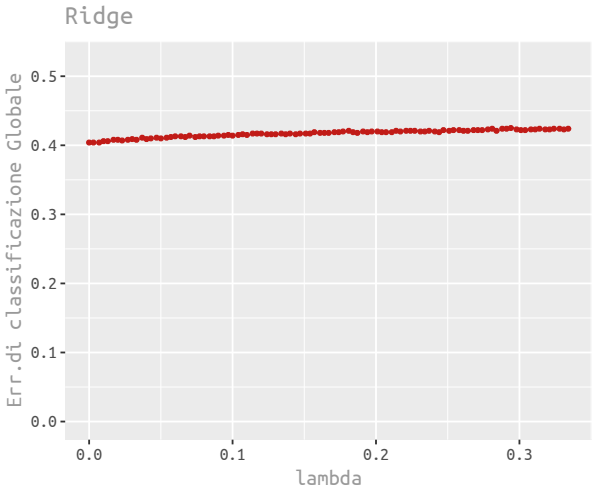


Figura 4.13: *Tuning Ridge*

La Tabella 4.3 (di cui si riporta uno stralcio), e il grafico in Figura 4.13 suggeriscono che all'aumentare del valore della penalità aumenta l'errore di classificazione globale mentre errori di primo e di secondo tipo restano sostanzialmente costanti. Ad ogni modo, anche i valori di λ migliori non forniscono come ci si poteva aspettare delle performance accettabili.

4.3.3 Gradient Boosting

Il Gradient Boosting come la Random Forest appartiene a quell'insieme di modelli che sfrutta il principio dell'*ensamble* e tra quelli utilizzati è quello che meglio riesce a cogliere le interazioni tra le variabili. Ciò è particolarmente interessante nel contesto di questa tesi perché le interazioni sono spesso associate a co-presenze di elementi e paesaggi nella foto.
I parametri di regolazione del Gradient Boosting sono la profondità dell'albero e il *learning rate*, e i valori dei candidati sono stati scelti seguendo una crescita esponenziale. In particolare:

- *learning rate* $\in \{0.001, 0.003, 0.007, 0.014, 0.032, 0.069, 0.152, 0.333\}$
- *max depth* $\in \{2, 5, 8, 10, 12, 15, 20\}$

Come mostrano la Tabella 4.4 (vedere Appendice A Tabella 4.12 per quella completa) e il contour plot in Figura 4.14 modelli con alberi più profondi tendono ad avere un'accuratezza maggiore ma non riescono a gestire bene errori di primo e di secondo tipo. D'altra parte, Gradient Boosting con *learning rate* maggiore fanno tendenzialmente meglio.

learning rate	max depth	1-accuracy	fpr	fnr
0.15	20	0.19	0.77	0.07
0.33	20	0.20	0.76	0.08
0.07	20	0.21	0.69	0.10
0.33	15	0.21	0.74	0.09
0.15	15	0.22	0.68	0.11
0.33	12	0.23	0.69	0.12
0.03	20	0.23	0.58	0.15
0.07	15	0.24	0.59	0.18
0.33	10	0.24	0.64	0.15
0.15	10	0.26	0.55	0.20
0.01	20	0.26	0.50	0.20
0.33	8	0.26	0.58	0.19
0.07	12	0.27	0.50	0.21
0.15	12	0.27	0.52	0.21
0.03	15	0.27	0.50	0.20
0.15	8	0.29	0.48	0.24

Tabella 4.4: *Tuning GB*

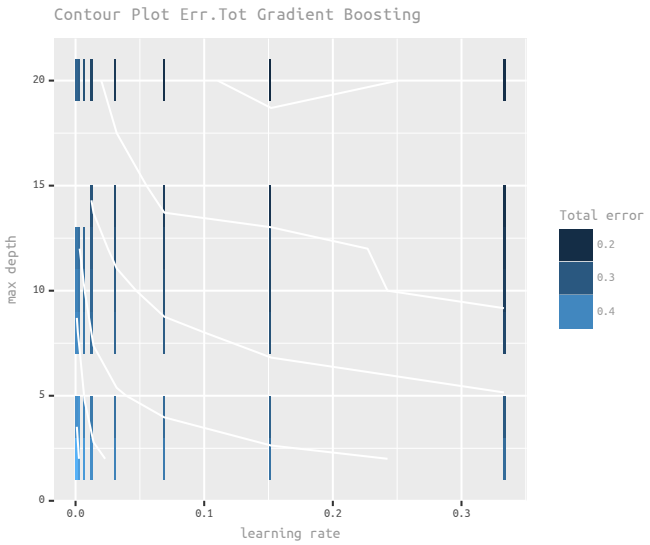


Figura 4.14: *Tuning GB*

Per quanto riguarda il set di parametri scelti, quello che meglio sembra bilanciare i tre tipi di errore è il Gradient Boosting con *learning rate* pari a 0.152 e alberi profondi 10.

4.3.4 XGBoosting

Dal punto di vista dei parametri di regolazione l'XGBoost differisce dal Gradient Boosting soltanto per λ, che penalizza il modello in base alla complessità degli alberi che costruisce ed è quindi coinvolto nel $trade - off$ tra varianza e distorsione.

$maxleaves$	$learning\ rate$	$lambda$	$1 - accuracy$	fpr	fnr
15	0.04	0.10	0.28	0.66	0.11
15	0.01	0.10	0.29	0.67	0.11
15	0.01	0.00	0.29	0.67	0.11
15	0.00	0.10	0.30	0.67	0.11
15	0.00	0.00	0.30	0.68	0.11
15	0.04	0.00	0.30	0.68	0.11
12	0.01	0.00	0.30	0.67	0.10
12	0.01	0.10	0.30	0.67	0.10
15	0.11	0.10	0.30	0.69	0.12
12	0.04	0.00	0.31	0.68	0.10
15	0.11	0.00	0.31	0.69	0.12
12	0.04	0.10	0.31	0.68	0.10
12	0.11	0.00	0.31	0.68	0.11
10	0.04	0.00	0.32	0.68	0.10
12	0.11	0.10	0.32	0.69	0.11

Tabella 4.5: Tuning XGBoost

Guardando la Tabella 4.5 dei migliori 15 modelli (vedere la 4.13 in Appendice A per quella completa) e in particolare l'andamento marginale dei tipi di errore al variare dei diversi parametri, si nota che anche qui alberi più profondi raggiungono tendenzialmente performance migliori. D'altra parte, *learning rate* bassi sembrano comportarsi meglio rispetto a quelli alti, mente non si apprezzano particolari differenze tra i due valori del parametro λ, che di fatto riflettono le situazioni di assenza di penalizzazione e di una penalizzazione moderata. Rispetto alla sua versione di base, l'XGBoosting si comporta mediamente peggio sia in termini di accuratezza che di errori di primo e secondo tipo, e questo enfatizza il fatto che l'ottimizzazione approssimata descritta nella sottosezione 1.0.4 garantisce delle performance migliori in termini computazionali ma non necessariamente previsivi. Per quanto riguarda la scelta del modello, la Tabella mostra che il modello migliore tra quelli proposti è quello con *learning rate* pari a 0.037, alberi di profondità pari a 15 e una penalizzazione moderata pari a 0.1

4.3.5 Light Gradient Boosting

Il Light Gradient Boosting ha essenzialmente gli stessi parametri di regolazione dell'XGBoosting e come spiegato nella sottosezione 1.0.4 differisce da questo per la struttura data all'albero. Il set di valori considerati per tali parametri è il seguente:

- *learning rate* $\in \{0.010, 0.107, 0.203, 0.250, 0.300\}$
- *max depth* $\in \{15, 35, 75, 127\}$
- *lambda* $\in \{0.1, 0.5, 2\}$

Dal punto di vista della performance, la Tabella 4.6 dei migliori 15 modelli (vedere la Tabella 4.14 in Appendice A per quella completa) mostra come l'LGB sia in questo caso il peggior modello tra quelli appartenenti alla famiglia del Boosting, dal momento che riesce a raggiungere al massimo un'accuratezza del 68%.

learningrate	*lambda*	*max leaves*	$1 - accuracy$	fpr	fnr
0.01	0.50	127	0.32	0.68	0.10
0.01	2.00	127	0.32	0.68	0.10
0.01	0.10	127	0.33	0.68	0.10
0.11	2.00	127	0.33	0.69	0.10
0.11	0.50	127	0.33	0.69	0.10
0.20	2.00	127	0.33	0.69	0.10
0.11	0.10	127	0.33	0.69	0.10
0.25	2.00	127	0.33	0.69	0.10
0.30	2.00	127	0.34	0.70	0.11
0.01	0.10	75	0.34	0.69	0.09
0.25	0.10	127	0.34	0.70	0.01
0.01	0.50	75	0.34	0.69	0.09
0.11	0.50	75	0.34	0.69	0.10
0.20	0.10	127	0.34	0.69	0.10
0.20	0.50	127	0.34	0.69	0.10

Tabella 4.6: Tuning LGB

Una plausibile motivazione potrebbe essere la tendenza di questo modello a sovradattarsi ai dati, dal momento che a differenza dell'XGBoost gli alberi non sono forzati ad avere una struttura piramidale. Ad ogni modo, è stato scelto il modello con *learning rate* pari a 0.01, un λ pari a 0.5 e un numero di foglie massimo pari a 127.

4.3.6 Multilayer Neural Network

Dopo aver utilizzato modelli basati sulle reti neurali per costruire le variabili esplicative, si è ritenuto opportuno utilizzarle anche come modello di previsione della variabile di risposta. Ora, se da un lato è vero che le reti neurali multistrato siano in grado di fornire dei buoni risultati in termini di performance, è vero anche che questi risultati sono condizionati (più che negli altri modelli) ad una scelta corretta dei parametri di regolazione. In particolare, si è deciso di concentrarsi sui tre parametri ritenuti più rilevanti, ovverosia:

- *Dimensione della rete* $\in (32, 64, 128, 256, 512, (32, 32), (64, 64), (128, 128), (256, 256), (32, 32, 32), (64, 64, 64), (128, 128, 128), (16, 16, 16, 16), (32, 32, 32, 32))$
- *Funzione di attivazione* : ReLU e Tangente iperbolica
- *Dropout ratio sugli input* $\in \{0.2, 0.8\}$

Per quanto riguarda gli altri parametri, si è deciso di utilizzare un valore di ϕ (descritto nella sottosezione 2.1.4) pari a 0.5 ed un *learning rate* adattivo.

hlsize	actfun	input_dropout_ratio	1-accuracy	fpr	fng
(128, 128)	TanhWithDropout	0.20	0.28	0.53	0.29
(32, 32, 32)	TanhWithDropout	0.20	0.31	0.45	0.34
(32, 32)	TanhWithDropout	0.20	0.32	0.47	0.35
(64, 64)	TanhWithDropout	0.20	0.33	0.44	0.36
512	RectifierWithDropout	0.20	0.33	0.35	0.38
128	RectifierWithDropout	0.20	0.33	0.36	0.39
32	RectifierWithDropout	0.20	0.34	0.37	0.39
512	TanhWithDropout	0.20	0.34	0.41	0.39
256	RectifierWithDropout	0.20	0.35	0.31	0.43
(16, 16, 16, 16)	TanhWithDropout	0.20	0.36	0.37	0.42
(256, 256)	TanhWithDropout	0.20	0.36	0.39	0.42
(16, 16, 16, 16)	RectifierWithDropout	0.20	0.36	0.39	0.42
128	TanhWithDropout	0.20	0.38	0.35	0.45
64	RectifierWithDropout	0.20	0.38	0.29	0.46
64	TanhWithDropout	0.20	0.39	0.32	0.46

Tabella 4.7: Tuning MLNNs

Dalla Tabella 4.7 (vedere la 4.15 in Appendice per quella completa) si osserva che pur avendo utilizzato una griglia relativamente poco fitta,le performance del modello sono estremamente sensibili al variare dei parametri di regolazione. Questo rimarca il fatto che fare *tuning* su una rete neurale è un'operazione tutt'altro che semplice e che richiede un elevato costo computazionale. Andando nel dettaglio, si vede che i modelli con un *input dropout ratio* elevato performano mediamente meglio con un valore pari a 0.2 piuttosto che 0.8, mentre la funzione di attivazione non pare assumere un ruolo rilevante. Inoltre, il fatto di utilizzare un'architettura più complessa non garantisce sempre performance migliori, al contrario diversi modelli con tre e quattro strati performano peggio degli altri.

Ad ogni modo, il modello che fornisce un miglior bilanciamento tra errore di classificazione globale e errori di primo e secondo tipo è un modello con due strati latenti, un *input dropout ratio* pari a 0.2 e una funzione di attivazione tangente iperbolica.

4.3.7 Support Vector Machine

Tra i metodi proposti l'SVM è sicuramente da un punto di vista quello meno appropriato dato l'alto numero di $features$ presenti e la sua difficoltà nello gestire il rumore. Infatti, nonostante siano stati effettuati ben due step di $feature\ selection$ è verosimile immaginare che vi siano ancora alcune variabili *noisy* che non portano informazione rilevante.

C	$Gamma$	$Kernel$	$1 - accuracy$	fpr	fnr
0.71	8.00	polynomial	0.38	0.61	0.46
4.00	8.00	radial	0.39	0.48	0.43
0.71	0.03	polynomial	0.39	0.56	0.31
0.71	8.00	radial	0.41	0.61	0.46
0.12	0.03	polynomial	0.42	0.57	0.33
0.12	8.00	polynomial	0.42	0.50	0.41
4.00	0.03	radial	0.42	0.54	0.37
0.12	8.00	radial	0.44	0.52	0.33
4.00	0.03	polynomial	0.44	0.58	0.46
0.71	0.03	radial	0.45	0.50	0.44
0.12	0.03	radial	0.45	0.59	0.43
4.00	8.00	polynomial	0.45	0.49	0.46

Tabella 4.8: Tuning SVM

La Tabella 4.8 enfatizza quest'aspetto e si nota che al variare dei valori dei parametri di regolazione (C, γ e il Kernel) l'errore di classificazione globale non va mai al di sotto dello 0.38, valore decisamente più elevato rispetto ad altri modelli utilizzati.

4.3.8 Confronto tra i modelli selezionati

Dopo aver completato la fase di tuning dei parametri e scelti i migliori modelli sulla base dei criteri descritti in precedenza, il passo successivo è stato quello di ristimare i modelli finali e validarli sulle 50000 osservazioni non ancora utilizzate allo scopo di evitare problemi di sovradattamento.

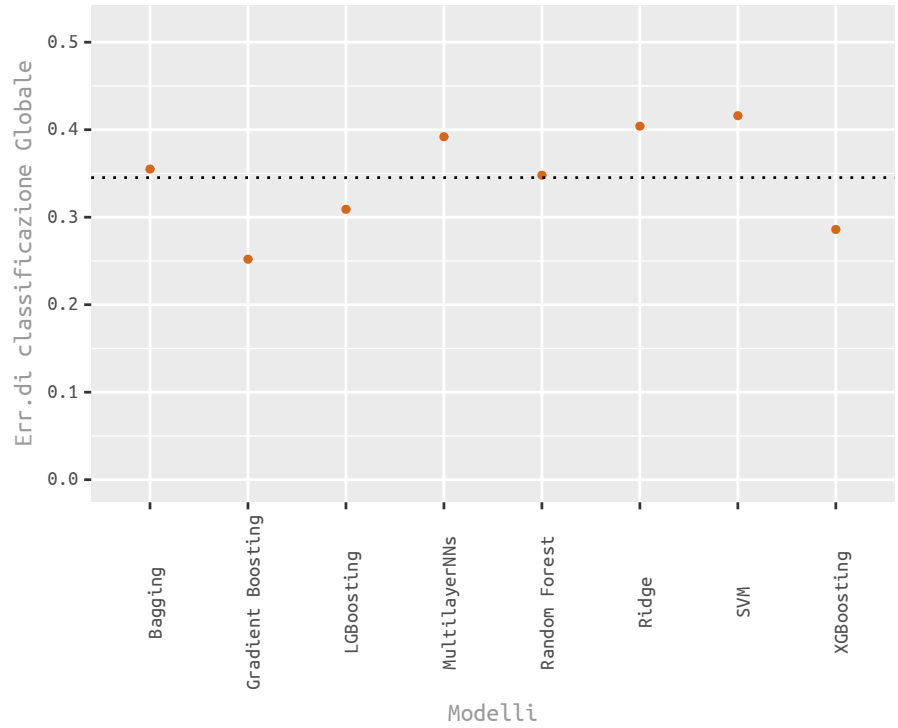


Figura 4.15: *Confronto tra i modelli finali*

La Figura 4.15 descrive l'errore di classificazione globale per ognuno dei modelli finali, ed in realtà quello che si nota è la presenza di 3 clusters. Infatti, i modelli appartenenti alla famiglia del Boosting ottengono delle buone performance in termini di accuracy tant'è che il Gradient Boosting addirittura raggiunge una percentuale di soggetti classificati correttamente del 75%. Il secondo cluster è costituito da altri due modelli basati sul principio dell'*ensamble*, Random Forest e Bagging, mentre il terzo vede rispettivamente la Ridge, l'SVM, e le MLNNs. Chiaramente, l'accuracy è soltanto uno degli indicatori di cui bisogna tener conto, soprattutto in contesti come questo dove le classi sono sbilanciate.

Modello	1-accuracy	fpr	fnr
Bagging	0.35	0.33	0.29
Gradient Boosting	0.25	0.20	0.53
LGBoosting	0.31	0.59	0.09
MultilayerNNs	0.39	0.62	0.17
Random Forest	0.35	0.36	0.28
Ridge	0.40	0.43	0.30
SVM	0.42	0.65	0.15
XGBoosting	0.29	0.58	0.11

Tabella 4.9: Performance dei modelli finali

La Tabella 4.9 investiga ulteriormente quest'aspetto e riporta oltre all'*accuracy* anche le stime delle probabilità di errore di primo e di secondo tipo. Di fatto le considerazioni fatte a partire dalla Figura 4.15 restano immutate, nel senso che il Gradient Boosting ha una performance globale migliore delle altre ed in generale i modelli ad albero fanno meglio rispetto ai restanti. Questo riflette la presenza di molte interazioni all'interno del fenomeno in analisi, ed è coerente se pensiamo ad esempio a come la compresenza di due elementi in una foto possa essere associata ad un commento. Rispetto agli altri modelli, stupisce un pò la performance decisamente non soddisfacente delle MultilayerNNs, addirittura peggiore rispetto alla Ridge che era stata utilizzata come *benchmark*. Le motivazioni che stanno alla base di questo sono diverse, e vanno ricercate in parte nella struttura di questo modello. Infatti, rispetto alle reti neurali a singolo strato queste sono caratterizzate da un'esplosione nel numero di parametri, per cui probabilmente il set trovato nella fase di *tuning* era semplicemente quello che meglio si adattava all'insieme di verifica, e non quello che descrivesse effettivamente bene il fenomeno di interesse. Un'altra motivazione potrebbe essere che questi modelli, così come le SVM, fanno più fatica in presenza di *variabili noisy*, che comunque non possono mai essere eliminate definitivamente. Ad ogni modo, un *tuning* su una griglia molto più fitta avrebbe probabilmente migliorato le cose, tuttavia l'obiettivo di questa tesi esulava dall'allenare in maniera rigorosa una rete neurale e quindi il tuning è stato effettuato in maniera più *naive*.

4.3.9 Interpretazione delle Features

Come detto a più riprese nel corso dei vari capitoli, l'obiettivo di questo elaborato non è solo esclusivamente legato alla previsione. Infatti, è di interesse anche valutare quale sia la relazione tra le $features$ (o gruppi di $features$) e la variabile di risposta, almeno in termini di direzione. Nella sezione 4.2 è stato effettuata un'analisi descrittiva in cui sono state individuate le variabili potenzialmente più rilevanti, senza però discutere se queste aumentassero o diminuissero la probabilità di ricevere un commento. Questo semplicemente perché gli indici di importanza delle variabili costruiti dai modelli ad albero nulla dicono in tal senso. D'altra parte, si è consapevoli che fermarsi a questo stadio porterebbe ad un'informazione parziale e relativamente poco utile. Per raggiungere lo scopo si è deciso di percorrere una strada alternativa, che si basa sempre sui modelli ad albero. L'idea è quella di stimare nuovamente un XGBoost, dal momento che anche con alberi poco profondi fornisce performance discrete, ed utilizzare i percorsi definiti dai vari alberi come variabili esplicative in un modello LASSO. Da un punto di vista pratico, questo equivale a stimare un modello in cui le variabili esplicative non sono tutte le interazioni possibili, che sarebbero praticamente infinite, ma le interazioni stimate al passo precedente dall'XGBoost, e che sono potenzialmente quelle realmente esplicative. Successivamente, sui relativi coefficienti viene applicato uno *shrinkage* in modo da effettuare un'ulteriore selezione. Da un punto di vista metodologico si è deciso di concentrarsi solo sulle variabili estratte a partire dalle reti allenate e su quelle relative ai profili psicologici degli utenti, ritenendo questi due di maggior interesse rispetto alle altre. Per quanto riguarda il primo, operativamente le variabili continue sono state ricodificate in dummy indicanti presenza/assenza allo scopo di avere un'interpretazione immediata delle interazioni. Dato l'elevato numero delle classi della rete Vgg16, è stata scelta una soglia pari a 0.25. Per quanto riguarda invece il secondo gruppo, le variabili sono state discretizzate in tre livelli rispetto al primo e al terzo quartile.
Andando nel dettaglio, rispetto ai contenuti dell'immagine sono emerse le seguenti considerazioni:

- Elementi che richiamano all'estate, come occhiali da sole, costumi da bagno e spiagge hanno un impatto positivo sulla generazione dei commenti.

- Particolari tipologie di abbigliamento sono maggiormente apprezzate. Tra queste, sottolineiamo le minigonne (risultato riscontrato peraltro anche in Khosla et al., 2014) e l'abaya. A tal proposito si può dire che se da un lato la prima rappresenta una $features$ "scontata", la seconda rimarca un apprezzamento verso le altre culture.

- Luoghi legati alla sfera sacra come cupole, chiese e monasteri hanno un impatto negativo sulla generazione dei commenti. Tuttavia, se la foto è scattata all'interno della chiesa e vi sono tante persone, la relazione diventa positiva. Ciò probabilmente accade in corrispondenza di foto che immortalano sacramenti come matrimoni,battesimi ecc.

- Animali domestici attirano commenti, con i gatti (quello soriano in particolare) che hanno un ruolo addirittura più importante rispetto ai cani.

- Libri e Televisori, oggetti che richiamano due idee diverse che l'individuo vuol dare di sé, rispettivamente intellettualità e lusso, aumentano la probabilità che la foto riceva un commento.

Passando invece alle variabili che descrivono i profili psicologici degli utenti, due sono gli aspetti interessanti che vale la pena segnalare. Il primo, è che individui felici ed in grado di interagire in maniera adeguata nel proprio contesto sociale ricevono mediamente più commenti rispetto agli altri. D'altra parte, tratti della personalità come eccessiva sicurezza in se stessi e scarsa empatia verso gli altri hanno un effetto negativo sulla sulla probabilità che la foto riceva un commento.

4.4 Previsione del numero di commenti

Finora ci si è concentrati esclusivamente nel prevedere se una foto riceve almeno un commento, affrontando un problema di classificazione. In questa sezione si affronta in maniera più breve il secondo problema statistico di questa tesi, ossia quello di prevedere il numero di commenti, utilizzando oltre ai modelli introdotti nel Capitolo 1 anche i modelli "ZIP" ed "Hurdle". Da un punto di vista metodologico si sottolinea che per motivazioni legate al tempo non si è seguito lo stesso rigore avuto in precedenza, e si è deciso sostanzialmente di utilizzare le stesse $features$ selezionate nei modelli di classificazione e anche gli stessi parametri di regolazione, consapevoli che non sia chiaramente la scelta ottimale.

4.4.1 Introduzione agli Hurdle Models e motivazioni teoriche del metodo

Per comprendere meglio le motivazioni che hanno spinto all'utilizzo di questi modelli, partiamo dal presupposto che la variabile di risposta osservata altro non è che un conteggio. Tipicamente, ciò che si fa è modellare la media di tale variabile attraverso un GLM Poisson, stimandolo in maniera iterativa con l'algoritmo *Scoring di Fisher*. Ora, il problema è che l'assunzione principale di un GLM Poisson è che la varianza della variabile di risposta cresca con la media, ipotesi disattesa in contesti come questo dal momento che la maggior parte delle risposte non riceve alcun commento. Questo causa quella che viene comunemente definita "sovradispersione", che si manifesta quando la varianza ipotizzata dal modello è minore di quella osservata nei dati. Tale fenomeno può avere delle conseguenze drammatiche sia in termini previsivi dal momento che il numero di zeri viene sempre sottostimato, sia in termini inferenziali. Infatti, ipotizzare una varianza minore di quella reale porta ad una sottostima degli standard error e quindi un aumento dell'errore di primo tipo. In generale, per risolvere il problema della sovradispersione viene specificato un modello binomiale negativo, tuttavia se tale fenomeno è dovuto principalmente agli zeri, sono preferibili altri approcci. In particolare, tre sono le strade che tipicamente vengono seguite:

- $Finite\ Mixture\ model$
- $Zero-inflated\ model$
- $Hurdle\ model$

Nei modelli a mistura finita si assume che i dati non provengano da una bensì da k popolazioni, le quali possono avere eventualmente una relazione diversa con le variabili esplicative. Da un punto di vista analitico, immaginando di osservare $y_i \sim Poisson(\lambda_j(z))\ \forall\ i = 1..N$, il modello può essere espresso dall'equazione (4.1) riportata di seguito:

$$P(y_i = q|x, z) = \sum_{j=1}^{k} \frac{\pi_j \lambda_j(z) e^{-\lambda_j z}}{q!} \tag{4.1}$$

dove k indica il numero di componenti mentre z ed x sono le matrici delle variabili esplicative. Le probabilità di appartenenza al $k - esimo$ gruppo di mistura (π_j) e le medie della variabile di risposta (λ_j) sono legate alle variabili esplicative attraverso le seguenti funzioni legame:

$$logit(\pi_j(x)) = log(\frac{\pi_j(x)}{\pi_j(x)}) = x_j^T \beta_j$$
$$log(\lambda_j(z)) = z_j^T \alpha_j$$

Dal punto di vista della stima dei parametri del modello β_j e α_j, queste vengono ottenute attraverso una combinazione dell'algoritmo EM e dell'algoritmo quasi-Newton. Il secondo modello comunemente utilizzato per tener conto della presenza di molti zeri è lo zero-inflated (Lambert, 1992). Da un punto di vista concettuale questo assume che gli zeri abbiano due cause, una strutturale ed una aleatoria. Per comprendere la differenza tra questi, si pensi ad un esempio in cui la variabile di risposta è il numero di sigarette fumate nell'ultima settimana. Da un lato, è possibile avere individui che non hanno fumato sigarette in quanto non fumatori, e dall'altro individui fumatori che per qualche motivo non hanno comunque fumato. I primi sono dunque zeri strutturali ed i secondi zeri dovuti al caso. Per gestire le due diverse fonti di zeri, gli zero-inflated modellano la variabile di risposta come una mistura tra una variabile di Bernoulli con massa di probabilità pari ad 1 in corrispondenza del valore 0, e tipicamente una Poisson (potrebbe essere anche una Binomiale negativa, per esempio). In questo caso, la specificazione del modello (4.2) sarà quindi la seguente:

$$\begin{aligned} P(y_i = 0|x, z) &= 1 - \pi(x) + \pi(x)e^{-\lambda(z)} \\ P(y_i = q|x, z) &= \pi(x)e^{(-\lambda(z))}\lambda(z)^q,\ q = 1..N \end{aligned} \tag{4.2}$$

dove:

- z ed x sono le matrici delle esplicative
- $\pi(x)$ è la probabilità di trovarsi nell'elemento della mistura che ha distribuzione di Poisson, ed è modellata come nel modello a mistura finita
- $\lambda(z)$ è la media della variabile di risposta condizionatamente al gruppo di mistura a cui l'osservazione appartiene, ed è modellata come nel modello a mistura finita

La stima dei parametri viene ottenuta attraverso un algoritmo EM implementato *ad hoc* da Lambert, (1992), che si basa sulla massimizzazione di una log-verosimiglianza che coinvolge un

modello logistico ed un modello di Poisson.
L'ultimo modello introdotto più di ricente rispetto agli altri per gestire i fenomeni caratterizzati da tanti zeri è l'Hurdle model (Welsh *et.al*, (1996)). La differenza sostanziale tra quest'ultimo e lo Zero-Inflated sta nel modo in cui vengono trattati gli zeri. Infatti, se negli Zero-Inflated si distingue tra zeri strutturali e zeri dovuti al caso, negli Hurdle model si assume che tutti gli zeri siano strutturali. Rispetto al problema affrontato in questa tesi, ciò significa che esistono foto con particolari caratteristiche che a prescindere non avranno mai un commento, il ché è verosimile. A partire da questa considerazione è immediato capire che la specificazione del modello non sarà più una mistura tra una Bernoulli degenere e una variabile di conteggio come ad esempio la Poisson ma tra una Bernoulli e una Poisson troncata. Analiticamente, il modello ha quindi la forma descritta nell'equazione (4.3):

$$\begin{aligned} P(y_i = 0|x) &= 1 - \pi(x) \\ P(y_i = q|x, z) &= \frac{\pi(x)e^{-\lambda(z)\lambda(z)^q}}{q!(1 - e^{-\lambda(z)})}, \ q = 1, 2..N \end{aligned} \tag{4.3}$$

dove x e z assumono la stessa formulazione vista nei due precedenti casi. Un aspetto estremamente interessante degli Hurdle model oltre alla loro semplicità concettuale sta sicuramente nella stima.

$$\begin{aligned} LL_{Hurdle} &= \sum_{y_i>0} x_i\beta + \sum_{i=1}^{n} log(1 + e^{x_i\beta}) + \sum_{y_i>0} y_i z_i \alpha - e^{-(z_i\alpha)} - log(1 - e^{-e^{z_i\alpha}}) - log(y_i!) \\ LL_{Hurdle} &= LL(\beta) + LL(\alpha) \end{aligned} \tag{4.4}$$

Il fatto di modellare in maniera disgiunta la probabilità di osservare uno zero e la media della Poisson troncata, fa si che la verosimiglianza possa essere separata in due componenti: di queste, la prima dipende da β ed è nient'altro che la verosimiglianza di un modello logistico, mentre la seconda dipende da α. Essendo le due componenti indipendenti, possono chiaramente essere stimate separatamente, ed infatti la prima viene ottenuta mediante l'algoritmo *Scoring di Fisher* mentre la seconda attraverso un algoritmo $Nelder - Mead$.

4.4.2 Confronto tra i modelli stimati

Come accennato all'inizio di questa sezione, i modelli a cui si è fatto riferimento sono sostanzialmente gli stessi utilizzati per affrontare il problema di classificare la presenza di almeno un commento in un'immagine. Inoltre, è stato ritenuto opportuno utilizzare anche modelli "Zip" e "Hurdle", motivando tale scelta con la presenza eccessiva di zeri che rappresenta esattamente il contesto in cui tali modelli nascono. Le metriche di confronto tra i modelli sono stati il canonico MSE e il MAE, per avere un indice di bontà della previsione più robusto alla presenza di foto con un altissimo numero di commenti.

	Modello	MSE	MAE
1	Ridge	2.74	0.68
2	Bagging	2.77	0.62
3	Random Forest	2.65	0.62
4	Gradient Boosting	2.71	0.58
5	XGBoosting	2.69	0.65
6	LGBoosting	2.61	0.62
7	MultilayerNNs	2.77	0.64
8	SVR	2.89	0.70
9	HURDLE (BN)	2.90	0.77
10	ZIP (BN)	2.98	0.76

Tabella 4.10: Performance dei modelli finali

Come si può osservare in Tabella 4.10, la gerarchia delle performance dei modelli di regressione è praticamente la stessa di quella osservata per i modelli di classificazione. Infatti, i modelli appartenenti alla famiglia del Boosting raggiungono tendenzialmente il miglior compromesso tra le due metriche, mentre non si apprezzano particolari differenze tra gli altri modelli, ad esclusione della SVR che non raggiunge buone performance.
Soffermandoci poi sui modelli ZIP e HURDLE, si sottolinea intanto che in virtù di problemi di singolarità della matrice delle esplicative è stata utilizzata una decomposizione in valori singolari parziale, facendo tuning sul numero di vettori da stimare (17 e 22, rispettivamente). Guardando la Tabella 4.10 si nota come tali modelli siano addirittura i peggiori in termini di performance. Una delle possibili motivazioni ha a che vedere col fatto che sebbene tali modelli nascano da un punto di vista teorico proprio per questa tipologia di dati, essi prevedono una forma funzionale troppo restrittiva e quindi siano poco flessibili. D'altra parte, anche se i diversi tipi di Boosting stimano il modello su una metrica sbagliata, la flessibilità consente loro di ottenere performance migliori rispetto agli altri modelli.

Discussione finale

Questa tesi offre una metodologia per prevedere la popolarità di tutti quei contenuti mediatici caratterizzati da immagini. Infatti, anche se in questo contesto ci si è rivolti esclusivamente all'analisi di dati Social Media, in particolar modo di Facebook, tale approccio ha tanti potenziali campi di applicazione. Il principale è sicuramente quello del marketing, basti pensare a quanto sarebbe utile per un'azienda produttrice di beni sapere che tipo di contenuti pubblicitari possano attirare maggiormente l'attenzione dei consumatori. Stesso discorso vale anche per le aziende produttrici di servizi, come alberghi, B&B, o anche ristoranti. Che struttura dovrebbe avere in termini di colori,ambientazione e contenuti la locandina proposta sui siti di booking? Insomma, c'è un mondo da scoprire e con i passi da gigante compiuti recentemente nell'ambito del *machine learning* questo approccio potrebbe dare contributi importanti.
Tuttavia, non bisogna pensare che questi performanti algoritmi bastino a raggiungere lo scopo: per cominciare, una fase preliminare fondamentale è quella del *web scraping*. Infatti, il dato di partenza è un dato altamente destrutturato, lontano dalla canonica matrice individui per variabili, ed è rappresentato dalle pagine html. Successivamente, si passa al processamento dell'immagine ed in questa fase bisogna prendere delle decisioni sulla costruzione delle $features$. A questo stadio diventa inoltre importante il supporto di modelli basati sulle reti neurali, che sono implementate in diversi linguaggi di programmazione, tra cui R, Julia e Python, che è stato utilizzato in questa tesi. Solo allora si passa alla fase di analisi, resa complessa dalla dimensionalità del dato che si analizza. Infatti, come si è visto non tutti i modelli sono adatti allo scopo, per cui è importante selezionare quelli giusti. Il messaggio di fondo è il seguente: un approccio basato sul principio "*push the button*" e sulla "$brute\ force$" non è un approccio vincente, ma sono necessarie scelte di buon senso in ognuna delle fasi descritte.
Da un punto di vista metodologico, vale la pena discutere su quali siano i punti deboli di questo elaborato. Il primo è di natura concettuale ed è legato alla definizione della variabile di risposta. Come nel lavoro di Philip J. McParlane *et.Al* (2014) il concetto di popolarità di un'immagine è stato operativizzato mediante il numero di commenti ricevuto da altri utenti alla foto, immaginando che i commenti suscitino soltanto reazioni positive. In uno sviluppo successivo sarebbe quindi interessante o effettuare una *sentiment analysis* sui commenti e poi utilizzare questa come punto di partenza per le fasi successive, oppure semplicemente considerare come indicatore il numero di like, che purtroppo non erano a disposizione. Il secondo punto da discutere ha a che fare con la costruzione delle $features$ ed in particolar modo di quelle relative al contenuto

dell'immagine. Infatti, sia la rete Vgg16 che la rete Place365 sono reti di classificazione e in quanto tali assegnano necessariamente ogni foto ad una classe. Questo è un problema soprattutto per la rete che identifica i le ambientazioni, dal momento che non è verosimile pensare che ogni foto ne abbia una. Si è cercato di tener conto di quest'aspetto considerando una soglia al di sotto della quale l'immagine viene classificata come "senza ambientazione", ma questa non è stata fissata in maniera rigorosa.
Tralasciando quest' aspetto , un punto di criticità maggiore ha a che fare col fatto che queste reti fanno fatica quando nell'immagine vi sono più elementi, ed è per questo che è stata introdotta la rete Yolo. Tuttavia, questa rete è allenata attualmente per riconoscere poco meno di 100 elementi diversi in un'immagine, per cui in un'analisi successiva sarebbe interessante utilizzare Yolo9000, che è ancora in fase di perfezionamento e che è in grado di riconoscere fino a 9000 elementi diversi.
Ad ogni modo, nonostante queste limitazioni si è riusciti ad ottenere comunque dei modelli con una buona performance predittiva e ad allo stesso tempo individuare dei *pattern* qualitativi interessanti, relativi non solo alla relazione tra il contenuto della foto e popolarità della stessa ma anche tra quest'ultima e i tratti psicologici degli individui.
Si conclude quest'elaborato dicendo che tutto il codice scritto sia per l'estrazione delle *features* che per il *data scraping* è a disposizione su *github* al seguente indirizzo:
https://github.com/xlxlcyRuSReXxlxl/FacebookAnalysis.git.

Appendice A: Tabelle relative alla fase di tuning dei parametri

Lambda	1-accuracy	fpr	fnr	Lambda	1-accuracy	fpr	fnr
0.001	0.404	0.313	0.424	0.187	0.418	0.299	0.445
0.003	0.404	0.313	0.424	0.157	0.419	0.297	0.446
0.007	0.404	0.313	0.425	0.171	0.419	0.296	0.447
0.010	0.406	0.311	0.427	0.174	0.419	0.297	0.447
0.013	0.406	0.312	0.427	0.184	0.419	0.297	0.447
0.023	0.407	0.310	0.429	0.194	0.419	0.298	0.446
0.017	0.408	0.310	0.430	0.204	0.419	0.298	0.446
0.020	0.408	0.310	0.429	0.207	0.419	0.297	0.447
0.027	0.408	0.310	0.430	0.211	0.419	0.297	0.447
0.033	0.408	0.309	0.431	0.244	0.419	0.298	0.446
0.030	0.409	0.307	0.432	0.177	0.420	0.296	0.447
0.040	0.409	0.308	0.431	0.191	0.420	0.296	0.447
0.043	0.410	0.307	0.433	0.197	0.420	0.297	0.447
0.050	0.410	0.307	0.433	0.201	0.420	0.296	0.447
0.037	0.411	0.305	0.434	0.217	0.420	0.296	0.448
0.047	0.411	0.305	0.435	0.231	0.420	0.297	0.448
0.054	0.411	0.305	0.434	0.234	0.420	0.297	0.447
0.057	0.412	0.303	0.436	0.241	0.420	0.298	0.447
0.067	0.412	0.303	0.436	0.181	0.421	0.295	0.449
0.074	0.412	0.303	0.437	0.214	0.421	0.295	0.448
0.060	0.413	0.301	0.437	0.221	0.421	0.296	0.449
0.064	0.413	0.301	0.438	0.224	0.421	0.296	0.449
0.077	0.413	0.303	0.438	0.227	0.421	0.296	0.449
0.080	0.413	0.303	0.437	0.237	0.421	0.297	0.448
0.084	0.413	0.302	0.437	0.251	0.421	0.296	0.449
0.087	0.413	0.302	0.438	0.261	0.421	0.296	0.449
0.070	0.414	0.300	0.439	0.264	0.421	0.297	0.449
0.090	0.414	0.301	0.440	0.284	0.421	0.297	0.449
0.100	0.414	0.302	0.439	0.254	0.422	0.296	0.450
0.097	0.415	0.300	0.441	0.258	0.422	0.296	0.450
0.104	0.415	0.301	0.440	0.268	0.422	0.295	0.450
0.110	0.415	0.303	0.439	0.271	0.422	0.295	0.451
0.107	0.416	0.301	0.441	0.274	0.422	0.295	0.450
0.124	0.416	0.301	0.441	0.301	0.422	0.296	0.450
0.127	0.416	0.301	0.441	0.304	0.422	0.296	0.451
0.130	0.416	0.301	0.442	0.278	0.423	0.294	0.451
0.137	0.416	0.301	0.442	0.298	0.423	0.294	0.452
0.144	0.416	0.301	0.442	0.308	0.423	0.295	0.451
0.114	0.417	0.299	0.444	0.311	0.423	0.296	0.451
0.117	0.417	0.299	0.443	0.318	0.423	0.296	0.451
0.120	0.417	0.301	0.443	0.321	0.423	0.295	0.452
0.134	0.417	0.298	0.443	0.331	0.423	0.296	0.452
0.140	0.417	0.299	0.444	0.281	0.424	0.293	0.453
0.147	0.417	0.300	0.443	0.288	0.424	0.294	0.452
0.151	0.417	0.300	0.443	0.291	0.424	0.293	0.453
0.154	0.417	0.299	0.444	0.314	0.424	0.294	0.453
0.161	0.418	0.298	0.444	0.324	0.424	0.295	0.453
0.164	0.418	0.298	0.445	0.328	0.424	0.294	0.453
0.167	0.418	0.298	0.445	0.334	0.424	0.295	0.452

Tabella 4.11: Tuning Ridge

learning rate	*max depth*	$1 - accuracy$	*fpr*	*fnr*
0.15	20	0.19	0.77	0.07
0.33	20	0.20	0.76	0.08
0.07	20	0.21	0.69	0.10
0.33	15	0.21	0.74	0.09
0.15	15	0.22	0.68	0.11
0.33	12	0.23	0.69	0.12
0.03	20	0.23	0.58	0.15
0.07	15	0.24	0.59	0.18
0.33	10	0.24	0.64	0.15
0.15	10	0.26	0.55	0.20
0.01	20	0.26	0.50	0.20
0.33	8	0.26	0.58	0.19
0.07	12	0.27	0.50	0.21
0.15	12	0.27	0.52	0.21
0.03	15	0.27	0.50	0.20
0.15	8	0.29	0.48	0.24
0.07	10	0.29	0.45	0.25
0.01	20	0.29	0.44	0.25
0.01	15	0.29	0.55	0.23
0.03	12	0.29	0.42	0.26
0.33	5	0.30	0.49	0.26
0.07	8	0.31	0.40	0.29
0.03	10	0.31	0.39	0.29
0.00	20	0.31	0.40	0.29
0.01	12	0.31	0.38	0.30
0.15	5	0.32	0.40	0.30
0.00	20	0.32	0.40	0.31
0.03	8	0.33	0.36	0.32
0.01	10	0.33	0.35	0.32
0.01	12	0.33	0.35	0.33
0.07	5	0.34	0.35	0.33
0.33	2	0.34	0.36	0.34
0.01	8	0.34	0.34	0.35
0.01	10	0.35	0.33	0.35
0.00	12	0.35	0.35	0.35
0.03	5	0.35	0.33	0.36
0.15	2	0.36	0.32	0.37
0.00	12	0.36	0.35	0.37
0.01	8	0.37	0.32	0.38
0.01	5	0.37	0.31	0.38
0.00	10	0.37	0.32	0.38
0.07	2	0.37	0.31	0.39
0.03	2	0.39	0.30	0.41
0.00	10	0.39	0.34	0.40
0.00	8	0.39	0.31	0.41
0.01	5	0.40	0.30	0.42
0.00	8	0.41	0.34	0.42
0.01	2	0.41	0.29	0.44
0.00	5	0.42	0.34	0.43
0.00	5	0.43	0.29	0.46
0.01	2	0.44	0.28	0.47
0.00	2	0.44	0.29	0.48
0.00	2	0.48	0.24	0.54

Tabella 4.12: Tuning Gradient Boosting

maxdepth	*learning rate*	*lambda*	$1 - accuracy$	*fpr*	*fnr*
15	0.04	0.10	0.28	0.66	0.11
15	0.01	0.10	0.29	0.67	0.11
15	0.01	0.00	0.29	0.67	0.11
15	0.00	0.10	0.30	0.67	0.11
15	0.00	0.00	0.30	0.68	0.11
15	0.04	0.00	0.30	0.68	0.11
12	0.01	0.00	0.30	0.67	0.10
12	0.01	0.10	0.30	0.67	0.10
15	0.11	0.10	0.30	0.69	0.12
12	0.04	0.00	0.31	0.68	0.10
15	0.11	0.00	0.31	0.69	0.12
12	0.04	0.10	0.31	0.68	0.10
12	0.11	0.00	0.31	0.68	0.11
10	0.04	0.00	0.32	0.68	0.10
12	0.11	0.10	0.32	0.69	0.11
10	0.11	0.10	0.32	0.69	0.10
10	0.01	0.00	0.33	0.69	0.10
10	0.04	0.10	0.33	0.69	0.10
15	0.25	0.10	0.33	0.71	0.13
10	0.01	0.10	0.33	0.69	0.10
10	0.11	0.00	0.34	0.69	0.10
8	0.04	0.10	0.34	0.69	0.09
15	0.25	0.00	0.34	0.72	0.13
8	0.11	0.10	0.34	0.69	0.10
12	0.00	0.10	0.34	0.69	0.10
10	0.25	0.10	0.34	0.70	0.11
8	0.11	0.00	0.34	0.69	0.10
8	0.04	0.00	0.34	0.69	0.10
12	0.00	0.00	0.34	0.69	0.10
8	0.25	0.00	0.34	0.70	0.10
15	0.33	0.10	0.34	0.72	0.13
12	0.25	0.00	0.34	0.71	0.11
12	0.25	0.10	0.34	0.71	0.11
10	0.25	0.00	0.34	0.70	0.11
10	0.33	0.10	0.34	0.71	0.11
8	0.25	0.10	0.34	0.70	0.10
8	0.33	0.00	0.35	0.70	0.11
10	0.33	0.00	0.35	0.71	0.11
12	0.33	0.00	0.35	0.72	0.12
12	0.33	0.10	0.35	0.72	0.12
8	0.01	0.00	0.35	0.70	0.09
5	0.11	0.10	0.35	0.70	0.10
8	0.01	0.10	0.35	0.70	0.09
5	0.11	0.00	0.35	0.70	0.10
5	0.25	0.00	0.35	0.70	0.10
8	0.33	0.10	0.35	0.71	0.10
10	0.00	0.10	0.35	0.70	0.10
15	0.33	0.00	0.36	0.74	0.14
10	0.00	0.00	0.36	0.70	0.10
5	0.25	0.10	0.36	0.70	0.10
5	0.33	0.00	0.36	0.71	0.10
5	0.04	0.00	0.36	0.70	0.09
5	0.04	0.10	0.36	0.70	0.09
5	0.33	0.10	0.36	0.71	0.10
8	0.00	0.00	0.38	0.71	0.09
8	0.00	0.10	0.38	0.71	0.09
5	0.01	0.00	0.38	0.71	0.09
5	0.01	0.10	0.38	0.71	0.09
5	0.00	0.00	0.43	0.74	0.09
5	0.00	0.10	0.43	0.74	0.09

Tabella 4.13: Tuning XGBoost

learning_rate	*lambda*	*max_leaves*	$1 - accuracy$	*fpr*	*fnr*
0.01	0.50	127	0.32	0.68	0.10
0.01	2.00	127	0.32	0.68	0.10
0.01	0.10	127	0.33	0.68	0.10
0.11	2.00	127	0.33	0.69	0.10
0.11	0.50	127	0.33	0.69	0.10
0.20	2.00	127	0.33	0.69	0.10
0.11	0.10	127	0.33	0.69	0.10
0.25	2.00	127	0.33	0.69	0.10
0.30	2.00	127	0.34	0.70	0.11
0.01	0.10	75	0.34	0.69	0.09
0.25	0.10	127	0.34	0.70	0.01
0.01	0.50	75	0.34	0.69	0.09
0.11	0.50	75	0.34	0.69	0.10
0.20	0.10	127	0.34	0.69	0.10
0.20	0.50	127	0.34	0.69	0.10
0.25	0.10	75	0.34	0.69	0.10
0.11	2.00	75	0.34	0.69	0.10
0.30	0.50	127	0.34	0.70	0.10
0.11	2.00	35	0.34	0.69	0.10
0.30	2.00	75	0.34	0.70	0.10
0.25	0.50	75	0.34	0.69	0.10
0.11	2.00	15	0.34	0.69	0.10
0.25	0.50	127	0.34	0.70	0.10
0.30	0.10	75	0.34	0.70	0.10
0.20	0.50	75	0.34	0.70	0.10
0.20	2.00	75	0.34	0.70	0.10
0.30	0.50	75	0.34	0.70	0.10
0.11	0.10	35	0.34	0.69	0.10
0.11	0.10	75	0.34	0.69	0.10
0.20	0.10	75	0.34	0.69	0.10
0.30	0.10	35	0.34	0.70	0.10
0.25	2.00	35	0.34	0.70	0.10
0.11	0.50	15	0.35	0.70	0.10
0.30	0.10	127	0.35	0.70	0.01
0.11	0.50	35	0.35	0.70	0.10
0.25	0.50	35	0.35	0.70	0.10
0.30	2.00	35	0.35	0.70	0.10
0.25	2.00	75	0.35	0.70	0.10
0.20	0.50	15	0.35	0.70	0.10
0.25	0.10	35	0.35	0.70	0.10
0.20	0.50	35	0.35	0.70	0.10
0.01	2.00	35	0.35	0.69	0.09
0.11	0.10	15	0.35	0.70	0.10
0.20	2.00	15	0.35	0.70	0.10
0.01	0.10	35	0.35	0.70	0.09
0.20	0.10	35	0.35	0.70	0.10
0.01	0.50	35	0.35	0.70	0.09
0.20	2.00	35	0.35	0.70	0.09
0.01	2.00	75	0.35	0.70	0.09
0.20	0.10	15	0.35	0.70	0.10
0.25	0.10	15	0.35	0.70	0.10
0.25	2.00	15	0.35	0.70	0.10
0.30	2.00	15	0.35	0.70	0.10
0.30	0.50	35	0.36	0.70	0.10
0.25	0.50	15	0.36	0.70	0.10
0.30	0.50	15	0.36	0.71	0.10
0.01	2.00	15	0.36	0.70	0.09
0.01	0.50	15	0.36	0.70	0.09
0.01	0.10	15	0.36	0.70	0.09
0.30	0.10	15	0.36	0.71	0.10

Tabella 4.14: Tuning LGB

hlsize	actfun	input_dropout_ratio	1-accuracy	fpr	fng
(128, 128)	TanhWithDropout	0.20	0.28	0.53	0.29
(32, 32, 32)	TanhWithDropout	0.20	0.31	0.45	0.34
(32, 32)	TanhWithDropout	0.20	0.32	0.47	0.35
(64, 64)	TanhWithDropout	0.20	0.33	0.44	0.36
512	RectifierWithDropout	0.20	0.33	0.35	0.38
128	RectifierWithDropout	0.20	0.33	0.36	0.39
32	RectifierWithDropout	0.20	0.34	0.37	0.39
512	TanhWithDropout	0.20	0.34	0.41	0.39
256	RectifierWithDropout	0.20	0.35	0.31	0.43
(16, 16, 16, 16)	TanhWithDropout	0.20	0.36	0.37	0.42
(256, 256)	TanhWithDropout	0.20	0.36	0.39	0.42
(16, 16, 16, 16)	RectifierWithDropout	0.20	0.36	0.39	0.42
128	TanhWithDropout	0.20	0.38	0.35	0.45
64	RectifierWithDropout	0.20	0.38	0.29	0.46
64	TanhWithDropout	0.20	0.39	0.32	0.46
32	TanhWithDropout	0.20	0.39	0.32	0.47
256	TanhWithDropout	0.20	0.40	0.30	0.48
(64, 64, 64)	TanhWithDropout	0.20	0.40	0.32	0.47
(128, 128,128)	TanhWithDropout	0.20	0.40	0.30	0.49
(32, 32, 32, 32)	TanhWithDropout	0.20	0.41	0.30	0.50
(64, 64)	TanhWithDropout	0.80	0.42	0.27	0.51
256	TanhWithDropout	0.80	0.42	0.25	0.53
(256, 256)	TanhWithDropout	0.80	0.42	0.27	0.52
(128, 128, 128)	RectifierWithDropout	0.20	0.43	0.26	0.53
(256, 256)	RectifierWithDropout	0.20	0.43	0.24	0.53
(32, 32, 32)	RectifierWithDropout	0.80	0.43	0.26	0.53
(128, 128)	TanhWithDropout	0.80	0.43	0.25	0.53
(32, 32)	RectifierWithDropout	0.20	0.43	0.26	0.53
(32, 32, 32, 32)	TanhWithDropout	0.80	0.43	0.26	0.53
(32, 32, 32)	RectifierWithDropout	0.20	0.43	0.27	0.53
512	RectifierWithDropout	0.80	0.44	0.22	0.55
512	TanhWithDropout	0.80	0.44	0.26	0.54
128	TanhWithDropout	0.80	0.44	0.23	0.54
(32, 32, 32)	TanhWithDropout	0.80	0.44	0.25	0.54
64	TanhWithDropout	0.80	0.44	0.23	0.55
(128, 128, 128)	TanhWithDropout	0.80	0.44	0.24	0.55
32	TanhWithDropout	0.80	0.45	0.22	0.56
(64, 64)	RectifierWithDropout	0.20	0.45	0.22	0.56
(64, 64, 64)	RectifierWithDropout	0.20	0.45	0.24	0.56
(32, 32, 32, 32)	RectifierWithDropout	0.20	0.45	0.24	0.56
(32, 32)	TanhWithDropout	0.80	0.45	0.23	0.56
(128, 128)	RectifierWithDropout	0.20	0.45	0.22	0.57
256	RectifierWithDropout	0.80	0.46	0.21	0.57
(32, 32)	RectifierWithDropout	0.80	0.46	0.21	0.58
32	RectifierWithDropout	0.80	0.46	0.20	0.58
(64, 64)	RectifierWithDropout	0.80	0.46	0.20	0.58
(64, 64, 64)	TanhWithDropout	0.80	0.46	0.21	0.58
64	RectifierWithDropout	0.80	0.46	0.20	0.58
(16, 16, 16, 16)	TanhWithDropout	0.80	0.47	0.21	0.58
(128, 128, 128)	RectifierWithDropout	0.80	0.47	0.20	0.59
(128, 128)	RectifierWithDropout	0.80	0.47	0.19	0.60
(64, 64, 64)	RectifierWithDropout	0.80	0.48	0.18	0.61
128	RectifierWithDropout	0.80	0.48	0.17	0.61
(16, 16, 16, 16)	RectifierWithDropout	0.80	0.48	0.18	0.62
(256, 256)	RectifierWithDropout	0.80	0.49	0.14	0.65
(32, 32, 32, 32)	RectifierWithDropout	0.80	0.49	0.06	0.81

Tabella 4.15: Tuning Multilayer Neural Networks

Bibliografia

An Intuitive Explanation of Convolutional Neural Networks. URL: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

Azzalini, A., B. Scarpa e et al. (2012). *Data analysis and data mining: An introduction*. A cura di OUP USA.

Building Regression Models in R using Support Vector Regression. URL: http://www.kdnuggets.com/2017/03/building-regression-models-support-vector-\\regression.html.

Caprara, G.V. et al. (1993). “"The big five questionnaire: A new questionnaire for the measurement of the five factor model"”. In: *Personality and Individual Differences,15,281-288*.

Caprara, G.V. et al. (1994). “"Mapping personality dimensions in the big five model. "”. In: *European Journal of Applied Psychology,44,9-16*.

Chen, T., Guestrin C. e et al. (2016). “Xgboost: A scalable tree boosting system.” In: *arXiv preprint arXiv:1603.02754*.

Chen, Y. Y. et al. (2014). “Predicting Viewer Affective Comments Based on Image Content in Social Media”. In: *ICMR ’14 Proceedings of International Conference on Multimedia Retrieval (Pages 233)*.

Chih-Wei Hsu Chih-Chung Chang, Chih-Jen Lin. *A pratical Guide to Support Vector Classification*. URL: http://www.csie.ntu.edu.tw~cjlin/papers/guide/guide.pdf.

COMPSCI 697L Deep Learning. URL: https://compsci697l.github.io/notes/neural-networks-1/.

CS231n Convolutional Neural Networks for Visual Recognition. URL: http://cs231n.github.io/convolutional-networks/.

Dallago, F., M. Roccato e et al. (2010). “"Right-Wing Authoritarianism, Big Five and Perceived Threat to Safety."”. In: *European Journal of Personality. Eur. J. Pers. 24: 106–122 (2010). DOI: 10.1002/per.745*.

Dalrymple, M. L. et al. (2003). “"Finite mixture, zero-inflated Poisson and hurdle models with application to SIDS."”. In: *Computational Statistics and Data Analysis 41.3-4 (2003): 491-504*.

Deep Learning with H2O. URL: https://h2o-release.s3.amazonaws.com/h2o/rel-slater/9/docs-website/h2o-docs/booklets/DeepLearning_Vignette.pdf.

Dhar, S. et al. (2011). “High Level Describable Attributes for Predicting Aesthetics and Interestingness”. In: *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on (pp. 1657-1664). IEEE*.

Efron, B., T. Hastie e et al. (2016). *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. A cura di Cambridge University Press.

Felzenszwalb, P. F., D. P. Huttenlocher e et al. (2004). "Efficient Graph-Based Image Segmentation". In: *International journal of computer vision.*

Felzenszwalb, P. F. et al. (2010). "Object Detection with Discriminatively Trained Part Based Models". In: *IEEE transactions on pattern analysis and machine intelligence, 32(9), 1627-1645.*

Furlong, M. J. (2014). ""Preliminary development and validation of the Social and Emotional Health Survey for secondary school students."". In: *Social Indicators Research 117.3: 1011-1032.*

Garson, G.D. ""Interpreting neural network connection weights."". In: *Artificial Intelligence Expert. 6(4):46–51.*

Gelli, F. et al. (2015). "Image Popularity Prediction in Social Media Using Sentiment and Context Features". In: *In Proceedings of the 23rd ACM international conference on Multimedia (pp. 907-910).ACM.*

Goodfellow, I. et al. (2016). *Deep Learning*. A cura di MIT Press.

Gurney, K. (1997). *An Introduction to Neural Networks*. A cura di University College London (UCL) Press.

Hastie, T. et al. (2009). *The elements of statistical learning*. A cura di Springer New York.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. A cura di Inc Prentice Hall International.

ImageNet: VGGNet, ResNet, Inception, and Xception with Keras. URL: `https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-\\xception-keras/`.

Khosla, A et al. (2014). "What Makes an Image Popular?" In: *WWW '14 Proceedings of the 23rd international conference on World wide web,Pages 867-876.*

Lambert, D. (1992). ""Zero-inflated Poisson regression, with an application to defects in manufacturing."". In: *Technometrics 34.1 (1992): 1-14.*

Large Scale Visual Recognition Challenge 2014 (ILSVRC2014). URL: `http://www.image-net.org/challenges/LSVRC/2014/`.

Laurae++:xgboost/LightGBM. URL: `https://sites.google.com/view/lauraepp/parameters`.

McParlane, P. J. et al. (2014). ""Nobody comes here anymore, it's too crowded"; Predicting Image Popularity on Flickr". In: *Proceedings of International Conference on Multimedia Retrieval (p. 385).ACM.*

One by One [1x1] Convolution-counter-intuitively useful. URL: `http://iamaaditya.github.io/2016/03/one-by-one-convolution/`.

Prezza, M. et al. (1997). ""La scala dell'autostima di Rosenberg: traduzione e validazione italiana [Rosenberg's self-esteem scale: Italian translation and validation]."". In: *Social Indicators Research 117.3: 1011-1032.*

Redmon, J. et al. (2016). “You Only Look Once: Unified, Real-Time Object Detection”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788.*

Rosebrock, Adrian. *Blur detection with OpenCV*. URL: https://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/.

Siersdorfer, S. et al. (2010). “Analyzing and predicting sentiment of images on the social web.” In: *In Proceedings of the 18th ACM international conference on Multimedia (pp. 715-718). ACM.*

Simonyan, K., A. Zisserman e et al. (2014). “Very Deep Convolutional Networks For Large Scale Image Recognition”. In: *arXiv preprint arXiv:1409.1556.*

Srivastava, N. et al. (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research 15 1929-1958.*

The Definitive Performance Tuning Guide for H2O Deep Learning. URL: https://blog.h2o.ai/2015/02/deep-learning-performance/.

Totti, L. et al. (2014). “The Impact of Visual Attributes on Online Image Diffusion”. In: *Proceedings of the 2014 ACM conference on Web science (pp. 42-51).ACM.*

Tsai, G. (2010). “"Histogram of oriented gradients."”. In: *University of Michigan.*

Understanding Convolutional Neural Networks for NLP. URL: http://www.wildml.com/2015/11/understanding-convolutional-neural-\\networks-for-nlp/.

Understanding Support Vector Machine Regression. URL: https://it.mathworks.com/help/stats/understanding-support-vector-machine-\\regression.html.

Understanding Support Vector Machine via Examples. URL: https://sadanand-singh.github.io/posts/svmpython/.

What is LightGBM, How to implement it? How to fine tune the parameters? URL: https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-\\lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc.

Wong, S.C. et al. (2016). “Understanding data augmentation for classification: when to warp?” In: *Digital Image Computing: Techniques and Applications (DICTA), International Conference on (pp. 1-6). IEEE.*

Yolo: You only look once,How it works(Video+Slides). URL: https://www.youtube.com/watch?v=L0tzmv--CGY.

Zhou, B. et al. (2017). “Places: A 10 million Image Database for Scene Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Zhu, J. Y. et al. (2017). “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *arXiv preprint arXiv:1703.10593.*

Ringraziamenti

Roma non è stata costruita in un solo giorno, e soprattutto non da un solo uomo. Ed ecco che, ancora una volta, la fortuna di avere al mio fianco delle persone meravigliose ha reso possibile un altro grande traguardo.
Ringrazio innanzitutto tutta la mia famiglia, a cui dedico questa tesi, per il supporto che mi ha dato ogni giorno e per avermi sostenuto totalmente nella scelta di venire a Padova. Mia madre, mio padre e mia sorella sono senza dubbio le persone più importanti nella mia vita, e che amo incondizionatamente.
Ringrazio gli amici di sempre, in particolare Francesco, Ciro, Roberto e Giulia, con cui il rapporto persiste nonostante le distanze e che spero non si perda mai.
Ringrazio poi Alessandro, Mattia e Paolo, che non mi hanno fatto sentire mai nostalgia di casa e con cui ho condiviso dei momenti meravigliosi sotto lo stesso tetto.
Ringrazio poi tutte le persone conosciute qui e con cui ho vissuto degli anni splendidi, in particolare voglio ringraziare i Soliti, Alice e Giulia, che è stata la persona più rappresentativa di questo percorso.
Ringrazio Padova, perché al di là delle chiacchiere da bar la verità è che mi ha accolto in maniera stupenda sin dal primo giorno. Spero di tornarci, un giorno.
Ringrazio Silvia, che mi ha pregato in tutti di scrivere i ringraziamenti dicendo che alla fine l'avrei apprezzato, ed aveva ragione. La ringrazio anche per l'aiuto che mi ha dato per raggiungere quest'obiettivo, fino all'ultimo giorno.
Infine, ringrazio il Prof.Livio Finos e il Dott.Dario Solari per avermi dato la possibilità di mettermi alla prova con questa tesi e che hanno avuto tanta pazienza nel sopportare le mie paranoie e le mie ansie. Adesso mi sento veramente pronto per trovare il mio posto nel mondo e dimostrare chi sono.
Concludo i ringraziamenti con questa frase tratta dal mio film preferito, "A beautiful mind": *"Ho sempre creduto nei numeri. Nelle equazioni e nella logica che conduce al ragionamento. Dopo una vita vissuta in questi studi, io mi chiedo: cos'è veramente la logica? Chi decide la ragione? La mia ricerca mi ha spinto attraverso la fisica la metafisica, mi ha illuso e mi ha riportato indietro. Ed ho fatto la mia più importante scoperta della mia carriera. La più importante scoperta della mia vita. È soltanto nelle misteriose equazioni dell'amore chi si può trovare ogni ragione logica."*