UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
GIOVANNI DEGLI ANTONI

Corso di Laurea triennale in Informatica

ANALISI DEI DATI PER PROBLEMI DI MEDICINA LEGALE

Relatore: Prof. Dario Malchiodi
Correlatore: Prof. Anna Maria Zanaboni

Tesi di Laurea di:
Alessandro Beranti
Matr. Nr. 855489

ANNO ACCADEMICO 2018-2019

Dedico questo mio lavoro alla mia famiglia

Ringraziamenti

Prima di procedere con la trattazione volevo dedicare qualche riga a chi mi ha aiutato e supportato in questo percorso.

Un grazie doveroso alla mia famiglia che ha sempre creduto in me. Grazie a tutti gli amici che mi sono sempre stati vicini, a tutti i miei compagni di corso conosciuti negli anni e in particolare a Andrea Pennati e Samuele Valente con cui ho fatto l'ultima rincorsa alla laurea. Infine un sincero ringraziamento al mio relatore Dario Malchiodi per la sua infinita disponibilità e il costante supporto sia durante il tirocinio che nella stesura della tesi.

Grazie a tutti.

Indice

Introduzione

Il tirocinio è stato svolto internamente all'Università. Il lavoro si è concentrato inizialmente sull'apprendere le basi del Machine Learning, e una volta che queste sono state consolidate si è proceduto ad affrontare il problema di classificare il mezzo che ha investito una persona a partire dalle caratteristiche di quest'ultima. Le ragioni che mi hanno spinto a svolgere un tirocinio su questo argomento, e di conseguenza una tesi, sono principalmente due: la necessità da parte di medici legali di conoscere il prima possibile che tipo di veicolo, vettura o mezzo pesante, avesse investito il malcapitato, ma sopratutto sono sempre stato affascinato dallo studio dei dati e dal Machine Learning in generale.

La tesi si compone di tre capitoli: nel Capitolo 1 viene introdotto il concetto di Machine Learning, in particolare viene spiegato come funziona e quali sono gli approcci esistenti. Successivamente vengono trattati i modelli che ho usato durante il tirocinio, spiegando come funzionano nello specifico.

Nel Capitolo 2 viene inizialmente spiegato il dataset che ho usato per prendere conoscenza e familiarità con le tecniche e gli algoritmi del Machine Learning. Successivamente viene illustrato il dataset che ho utilizzato durante il tirocinio, raccolto dai medici legali, ovvero una raccolta di dati legati a vittime di investimento, unitamente a una descrizione delle lesioni che hanno riportato. Vengono poi spiegate due tecniche usate per ridurre la dimensionalità, pratica utile nel processo di Machine Learning per migliorare le performance degli algoritmi, quando questi sono utilizzati su dataset di grandi dimensioni.

Infine nel Capitolo 3 si entra nello specifico del processo che ho usato per passare dal dataset a dei classificatori che predicono il tipo di veicolo che ha causato l'incidente, valutandoli in base all'accuratezza ottenuta e al variare dei diversi algoritmi di apprendimento discussi al Capitolo 1. Il lavoro si chiude con un'analisi dei risultati e un breve chiarimento delle tecnologie impiegate.

Capitolo 1

Machine Learning

Il termine Machine Learning, o apprendimento automatico in italiano, si riferisce alla capacità dei computer di apprendere e agire senza essere programmati esplicitamente. Gli strumenti di machine learning si occupano di dotare i programmi della capacità di "apprendere" e adattarsi agli input forniti. Al giorno d'oggi siamo circondati da tecnologie basate sull'apprendimento automatico:

- software che rilevano lo spam a partire dai nostri messaggi e-mail,
- motori di ricerca che imparano a ordinare i risultati di una query di ricerca al fine di mostrare prima quelli più rilevanti,
- transazioni con carta di credito protette da un software che impara a rilevare le frodi,
- fotocamere digitali che imparano a rilevare i volti,
- assistenti digitali che imparano a riconoscere i comandi vocali,
- veicoli autonomi addestrati a guidare senza intervento umano,
- applicazioni scientifiche come la bioinformatica, la medicina e l'astronomia.

1.1 Come funziona il Machine Learning

Nel Machine Learning vengono usati metodi statistici che applicano un procedimento di induzione a partire da dati che rappresentano istanze di un problema, accoppiate con una possibile soluzione, per migliorare le prestazioni di algoritmi o fare previsioni più accurate. La qualità e la dimensione dei dataset (collezione di dati), raccolti o resi disponibili e utilizzati nel processo sono fondamentali per il successo e l'accuratezza delle previsioni fatte.

Il processo parte da un domain set X, ovvero una raccolta arbitraria di dati, che rappresentano gli oggetti che si desidera etichettare. Essi possono essere già etichettati nel caso di apprendimento supervisionato come non esserlo nel caso non supervisionato. Questo set di dati rappresenta l'input dell'algoritmo di apprendimento che si è scelto di usare. Il risultato è un modello a cui è associato un errore di generalizzazione, ovvero la probabilità che non venga predetta l'etichetta corretta. Il domain set può essere diviso in due sottoinsiemi: il *training set*, insieme di dati che vengono scelti per compiere l'addestramento, e il *test set* che viene utilizzato per valutare le prestazioni del modello predittivo.

Una categorizzazione dei compiti del machine learning si ha quando si considera l'output desiderato del sistema che può essere di diversi tipi:

- classificazione, significa che il dato è categorico [1] e viene fatta una divisione dei dati in classi o etichette. Può essere una classificazione binaria, nel quale le etichette sono soltanto due, oppure multiclasse, etichette sono tre o più,
- regressione, usata per predire un valore assimilabile a una quantità che varia in un insieme continuo, per esempio il prezzo di una casa date la dimensione e la metratura [1],
- clustering, in cui un insieme di input viene diviso in gruppi in modo che i singoli elementi siano simili agli altri punti dello stesso insieme e diversi dagli elementi degli altri; diversamente da quanto accade per la classificazione, i gruppi non sono noti prima, rendendolo tipicamente un compito non supervisionato [2].

Gli algoritmi di apprendimento automatico vengono tipicamente organizzati in quattro categorie, a seconda della natura del "segnale" utilizzato per l'apprendimento o del "feedback" disponibile al sistema di apprendimento. Queste categorie, anche dette paradigmi, sono:

- Machine Learning supervisionato,
- Machine Learning non supervisionato,
- Machine Learning per rinforzo,
- Machine Learning semi-supervisionato.

1.1.1 Apprendimento supervisionato

Attraverso l'apprendimento supervisionato si cerca di costruire un modello partendo da dati di addestramento etichettati, a partire dai quali si tenta di fare previsioni

su dati non disponibili o futuri. Con il termine "supervisione" si intende quindi che nell'insieme dei campioni (o dataset), i segnali di output desiderati sono già noti poiché precedentemente etichettati. Nell'apprendimento supervisionato l'output può essere sia numerico che categorico, quindi può trattarsi rispettivamente di regressione e classificazione; siccome il tirocinio ha riguardato solo quest'ultima d'ora in avanti parleremo di classificazione.

Esistono molti algoritmi per svolgere apprendimento supervisionato, durante il tirocinio ho avuto modo di usare:

- Support Vector Machine,
- Decison Tree Classifier,
- Random Forest Classifier,
- Gaussian Naive Bayes,
- Linear Discriminant Analysis,
- Reti Neurali Multi-Strato.

1.1.1.1 Support Vector Machine

Il Support Vector Machine (SVM) è un algoritmo di apprendimento automatico supervisionato che può essere usato sia per scopi di classificazione che di regressione, si può applicare sia a problemi binari, sia a problemi multiclasse. Nel paragrafo seguente verrà descritto solo la versione per la classificazione binaria poichè è quella che ho usato durante il tirocinio.

L'SVM è basato sull'idea di riuscire a trovare un iperpiano che divida al meglio un set di elementi in due classi distinte [3]. Definiamo alcuni elementi che ci serviranno successivamente.

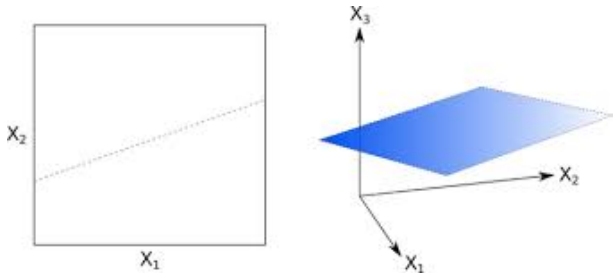

Figura 1: A sinistra grafico bidimensionale e destra tridimensionale

- Iperpiano: nel caso in cui si abbiano solo due dimensioni spaziali nel qualche si trovano le descrizioni degli oggetti da classificare x_1 e x_2, l'iperpiano è raffigurato come una linea che separa un insieme di dati [3]. Nel caso in cui le dimensioni siano 3, l'iperpiano è raffigurato come un piano, (cfr. Figura 1). Con più di 3 dimensioni viene definito "iperpiano".
- Support Vector: chiamati vettori di supporto in italiano, sono i punti che si trovano piu vicini all'iperpiano che divide i dati.
- Margine: è la distanza tra i vettori di supporto di due classi diverse. A metà di questa distanza viene tracciato l'iperpiano, (cfr. Figura 2).

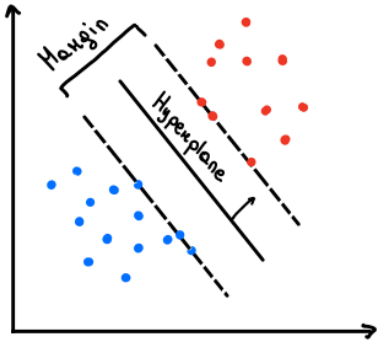

Figura 2: Iperpiano e margine

Il Support Vector Machine ha l'obiettivo di identificare l'iperpiano che divide meglio i vettori di supporto in classi. Esistono due algoritmi diversi a seconda se esista o meno l'iperpiano.

- Nel caso in cui esista e in particolare ce ne sia più di uno cerca quello con il margine più alto tra i vettori di supporto in modo da evidenziare meglio la divisione tra i dati. La massimizzazione del margine porta ad avere degli iperpiani che tendono a minimizzare le probabilità di errore quando classificano nuovi dati.
- Se l'iperpiano cercato non esiste, Support Vector Machine usa una mappatura non lineare per trasformare i dati di training X, in uno spazio di dimensione superiore rispetto a quello dei dati originali in modo che le immagini dei dati di due classi siano separati da un iperpiano, che sarà scelto per la suddivisione dei dati.

In dati linearmente separabili è possibile individuare un iperpiano in cui si possono distinguere due semispazi. Nella figura 3 è visibile come sia possibile disegnare un numero infinito di linee rette per separare i diversi elementi. Il problema è trovare quale tra le infinite rette risulti ottimale, ossia quella che, a fronte di una generica nuova osservazione, classifichi nel modo corretto l'osservazione.

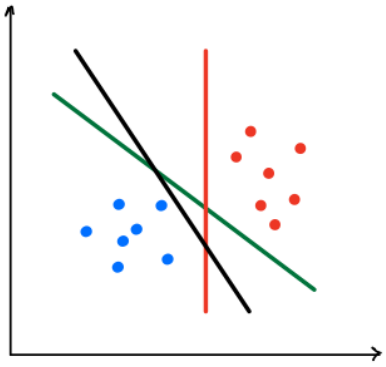

Figura 3: Infinite rette possono separare gli elementi

Dato un training set etichettato:

$$(x_1, y_1), ..., (x_n, y_n) \; \forall i \; x_i \in R^d, \; y_i \in \{-1, +1\}$$

dove x_i sono i punti da classificare e y_i sono le etichette.

Un iperpiano è definito come

$$w_0 + w_1 z_1 + w_2 z_2 + ... + w_m z_m = 0,$$

dove ω è il vettore di peso, z è il vettore di caratteristiche di input e w_0 è il bias. In sostanza in m dimensioni un iperpiano di separazione è una combinazione lineare di tutte le dimensioni uguagliate a 0. Ragionando a due dimensioni per semplificare il problema abbiamo che

$$w_0 + w_1 z_1 + w_2 z_2 = 0.$$

I punti che stanno sopra l'iperpiano e che rappresentano un classe soddisfano la seguente condizione:

$$w_0 + w_1 z_1 + w_2 z_2 > 0,$$

mentre qualsiasi punto che si trova sotto l'iperpiano, appartiene all'altra classe, che è soddisfatta dalla seguente condizione :

$$w_0 + w_1 z_1 + w_2 z_2 < 0,$$

L'algoritmo di apprendimento per SVM, in realtà, riscrive queste condizioni in modo più stringente:

$$w_0 + w_1 z_{i,1} + w_2 z_{i,2} \geq 1, \ \forall i \ t.c. \ y_i = 1,$$

$$w_0 + w_1 z_{i,1} + w_2 z_{i,2} \leq -1 \ \forall i \ t.c. \ y_i = -1,$$

dove $z_{i,1}$ rappresenta la prima componente dell'i-esimo vettore e $z_{i,2}$ la seconda componente dell'i-esimo vettore.

Se il vettore dei pesi è indicato da w e $\| \ w \ \|$ è la sua norma, allora la dimensione del margine massimo è

$$\frac{1}{\|w\|} + \frac{1}{\|w\|} = \frac{2}{\|w\|},$$

ciò significa che minimizzando la norma del vettore peso w, avremo margine massimo che determina l'iperpiano ottimale.

Non è però sempre possibile dividere i dati tramite un iperpiano, come esemplificato in Figura 4.

Per utilizzare la classificazione tramite iperpiani anche per dati che avrebbero bisogno di funzioni non lineari per essere separati, è necessario ricorrere alla tecnica degli spazi immagine ($feature \ spaces$). Questo metodo, che sta alla base della teoria delle SVM, consiste nel mappare i dati iniziali in uno spazio di dimensione superiore. Presupponendo quindi $m > n$, per la mappa si utilizza una funzione

$$\phi : \mathbb{R}^n \to \mathbb{R}^m, \tag{1}$$

attraverso la funzione ϕ i dati vengono mappati in uno spazio a più dimensioni, ciò comporta che i dati potrebbero essere linearmente separabili e quindi sarebbe possibile trovare un iperpiano che li separi [3].

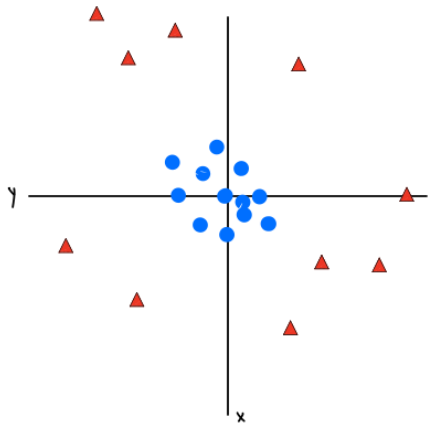

Figura 4: Non sempre è possibile dividere i dati linearmente

La tecnica degli spazi immagine è particolarmente interessante per algoritmi che utilizzano i dati di training x_i solo attraverso prodotti scalari $x_i \cdot x_j$. In questo caso nello spazio $\mathbb{R}^m$ non si devono trovare esplicitamente $\phi(x_i)$ e $\phi(x_j)$ ma basta calcolare il loro prodotto scalare $\phi(x_i) \cdot \phi(x_j)$. Per rendere semplice questo ultimo calcolo, che in spazi di dimensioni elevate diventa molto complicato, si utilizza una funzione detta *kernel* che restituisce direttamente il prodotto scalare delle immagini:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j), \tag{2}$$

Esistono svariati *kernel*, i più utilizzati sono:

- il kernel lineare: $K(x, y) = x \cdot y$,
- il kernel polinomiale: $K(x, y) = (x \cdot y)^d$ oppure $K(x, y) = (1 + x \cdot y)^d$,
- il kernel gaussiano: $K(x, y) = \exp(-||x - y|\,|^2)/(2\sigma 2)$,
- il kernel sigmoide: $K(x, y) = \tanh(kx \cdot y - \delta)$.

1.1.1.2 Decision Tree Classifier

Il Decision Tree Classifier è un metodo di apprendimento supervisionato usato sia per scopi di classificazione che di regressione. Utilizza un albero decisionale, *decision tree*, composto da [4]:

- nodi non terminali: rappresentano un test su uno o più caratteristiche,
- ramo: rappresenta un esito del test,

- foglia: rappresenta una possibile classe.

Dato un insieme di dati è possibile costruire un numero esponenziale di alberi di decisione. Alcuni alberi sono più efficienti di altri ma trovare l'albero ottimo è una cosa computazionalmente infattibile a causa dello spazio esponenziale nel quale bisogna cercare. Tuttavia esistono algoritmi che riescono, in un tempo ragionevole, a trovare un albero efficiente anche se non corrisponde all'ottimo. Questi algoritmi spesso implementano una strategia greedy che crea l'albero facendo una serie di decisioni localmente ottime su quale caratteristica dei dati usare per partizionare i dati [2].

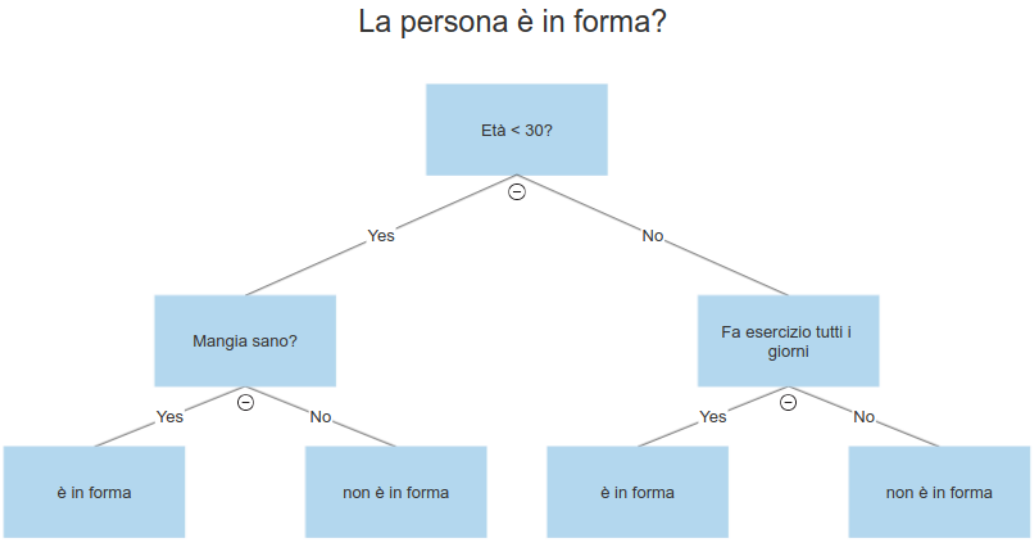

Figura 5: Visualizzazione di un albero di decisione

Uno di questi algoritmi è l'algoritmo di Hunt nel quale l'albero di decisione viene costruito ricorsivamente partizionando il training set in sottoinsiemi sempre più piccoli. Sia D_t l'insieme dei dati di training che sono associati al nodo t e sia $y = y_1, y_2, ..., y_c$ le etichette. La definizione ricorsiva dell'algoritmo di Hunt è la seguente [2].

- se tutti i dati in D_t appartengono alla stessa classe y_t, allora t è una foglia etichettata come y_t,
- se D_t contiene dati che appartengono a più di una classe, viene selezionata una caratteristica per eseguire un test per partizionare i dati in sottoinsiemi più piccoli. Viene creato un nodo figlio per ogni risultato della condizione

del test e i dati in D_t vengono divisi in base al risultato del test. L'algoritmo viene applicato in modo ricorsivo ad ogni nodo figlio.

La misura di selezione degli attributi è un'euristica per selezionare il criterio di suddivisione che divide i dati in modo da massimizzare l'omogeneità dei sottoinsiemi di dati. I più famosi sono l'entropia e l'indice di Gini [4].

1.1.1.3 Random Forest Classifier

Il Random Forest Classifier è un algoritmo di apprendimento supervisionato. Esso considera più alberi di decisione che operano come un insieme (vedi. Figura 6). Ogni albero della foresta genera una previsione e quella che compare più frequentemente diventa la previsione del modello [5].

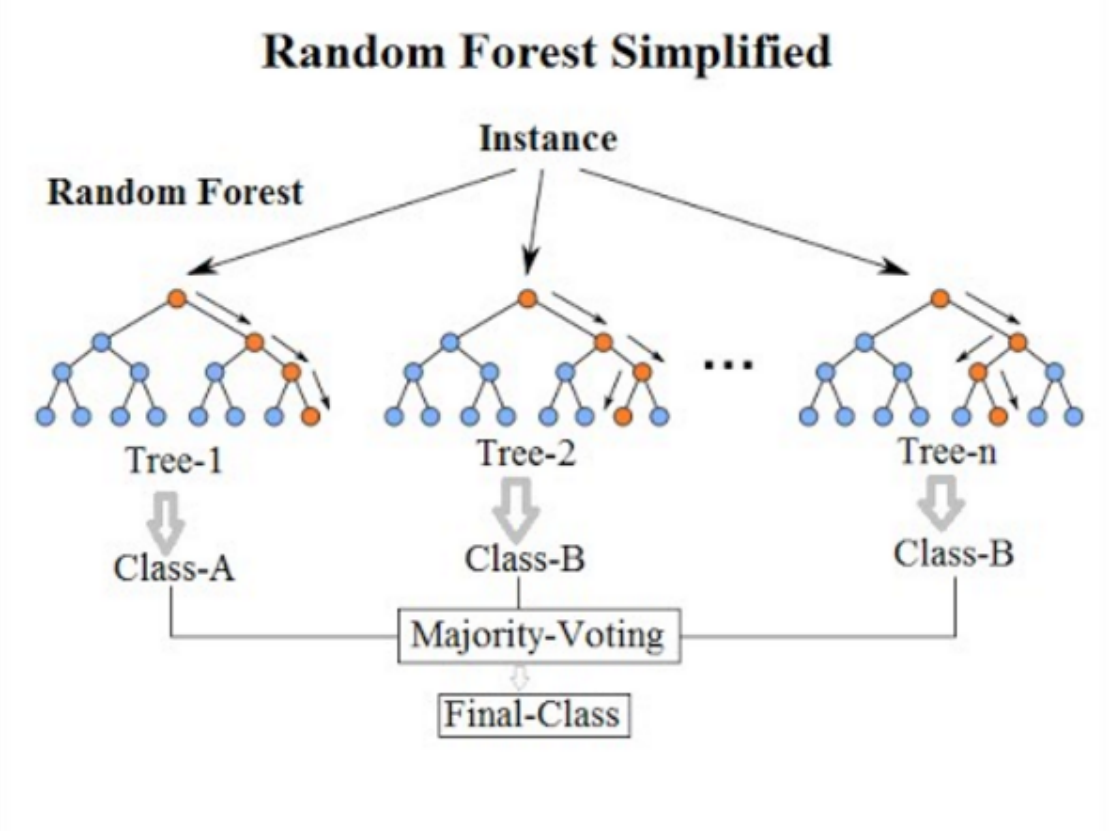

Figura 6: Random Forest composto da più alberi di decisione

Il concetto fondamentale dietro una foresta casuale risiede nel fatto che un grande numero di alberi non correlati tra loro che operano insieme tendono a essere più efficienti di un singolo albero.

La bassa correlazione è la chiave, modelli non correlati possono produrre previsioni d'insieme più accurate di qualsiasi singola previsione. La ragione risiede nel fatto che, anche se alcuni alberi potrebbero sbagliare, molti altri avranno ottenuto la previsione corretta.

1.1.1.4 Gaussian Naive Bayes

Il Gaussian Naive Bayes è uno dei più semplici algoritmi di apprendimento supervisionato, e si basa sul teorema di Bayes. L'algoritmo assume che l'effetto di una particolare caratteristica in una classe sia indipendente dalle altre caratteristiche. Anche se le caratteristiche sono interdipendenti, esse vengono comunque considerate in modo indipendente [6]. Questa ipotesi, a cui ci si riferisce come indipendenza condizionale di classe, semplifica il calcolo. Applicando il teorema di Bayes si ha

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}, \tag{3}$$

dove y è un'etichetta e $X = (z_1, z_2, z_3, ..., z_n)$ è il corrispondente vettore di caratteristiche.

Siccome abbiamo assunto che le caratteristiche siano indipendenti possiamo dire che:

$$P(y|z_1, ..., z_n) = \frac{P(z_1|y)P(z_2|y)...P(z_n|y)P(y)}{P(z_1)P(z_2)...P(z_n)}. \tag{4}$$

Poiché il denominatore rimane costante per un determinato input possiamo rimuoverlo e scrivere:

$$P(y|z_1, ..., z_n) \propto P(y) \prod_{i=1}^{n} P(z_i|y). \tag{5}$$

Ora dobbiamo creare un modello per classificare. Lo facciamo trovando la probabilità di un dato insieme di input per tutti i possibili valori di y e prendendo l'output con la massima probabilità:

$$y = \arg\max P(y) \prod_{i=1}^{n} P(z_i|y). \tag{6}$$

Rimane solo da calcolare $P(y)$ e $P(z_i|y)$, che sono ricavabili dal dataset dato in input al sistema.

Nel Gaussian Naive Bayes si presume che i valori continui associati a ciascuna caratteristica siano distribuiti secondo un modello gaussiano.

Sulla base di questa assunzione è quindi possibile scrivere le probabilità condizionate che compaiono in (6) nel modo seguente

$$P(z_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(z_i-\mu_y)^2}{2\sigma_y^2}}, \tag{7}$$

dove μ_y è la media e σ_y la deviazione standard.

1.1.1.5 Linear Discriminat Analysis

Il Linear Discriminat Analysis è un algoritmo di apprendimento supervisionato il cui scopo è quello di trovare una combinazione lineare di caratteristiche che caratterizza o separa due o più classi di oggetti. La combinazione risultante può essere utilizzata come classificatore o anche per la riduzione della dimensionalità (vedi Paragrafo 2.3).

Il processo prevede di proiettare i dati in input su un sottospazio lineare dalle direzioni che massimizzano la divisione tra le classi. La dimensione dell'output è necessariamente inferiore al numero di classi, quindi questa è, in generale, una riduzione della dimensionalità piuttosto forte, e ha senso solo in un ambiente multiclasse.

1.1.1.6 Reti Neurali Multi-Strato

Le Reti Neurali Multi-Strato sono un modello che utilizza l'algoritmo "backpropagation", basato sulla minimizzazione dell'errore, per compiere apprendimento supervisionato. L'idea di base è quella del neurone umano e della rete di neuroni che compone il nostro cervello. Il componente principale di una rete neurale è detto neurone o percettrone. Esso è identificato da n pesi reali $w_1, w_2, ..., w_n$ e quando riceve una serie di input $x_1, x_2, ..., x_n$ li moltiplica per i pesi corrispondenti, viene così prodotto un valore v sommando i prodotti ottenuti a cui viene sommato un termine di bias [7]. L'output della rete è ottenuto calcolando una particolare funzione, detta funzione di attivazione, usando v come argomento.

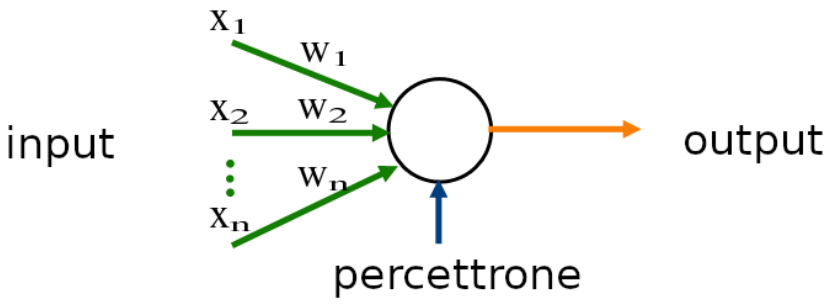

Figura 7: Visualizzazione di un percettrone

Una rete con un solo percettrone è chiamata a singolo livello. Esistono poi le reti multistrato, sono reti i cui percettroni sono disposti su più livelli, come indicato in Figura 8. Esistono tre tipi di livelli:

- livello input: costituito da un insieme di neuroni che rappresentano le caratteristiche in input,
- livello output: riceve i valori dall'ultimo livello hidden presente e li trasforma in valori di output,
- livello hidden: livelli intermedi tra input e output.

In questa rete ogni nodo, esclusi quelli di input, usa una funzione di attivazione non lineare per modellare il comportamento.

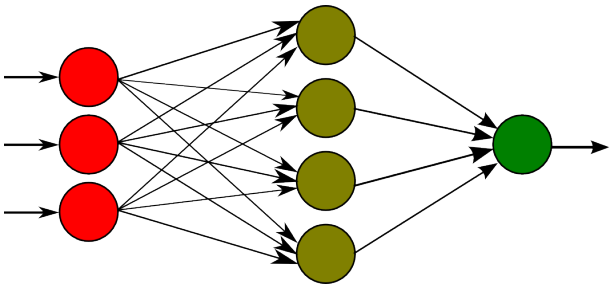

Figura 8: Visualizzazione di una rete multistrato; in rosso il livello di input, al centro il livello hidden, in verde a destra il livello di output

Esistono diverse funzioni di attivazione:

- Identity: $f(x) = x$,
- Logistic: $f(x) = \frac{1}{1+e^{-x}}$,
- Tanh: $f(x) = \tanh(x)$,
- Relu: $f(x) = \max(0, x)$.

L'output y viene calcolato nel seguente modo:

$$y = \Phi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \Phi(w^T x + b), \tag{8}$$

dove w è il vettore di pesi, x è il vettore di input, b è il bias e Φ è la funzione di attivazione.

Il processo di addestramento del Multi-Layer Perceptron avviene mediante una continua regolazione dei pesi delle connessioni dopo l'elaborazione di ogni oggetti nel dataset a disposizione. Questa regolazione si basa sull'errore nell'output e dà vita a un processo di apprendimento chiamato "backpropagation". Tale processo continua fino a che non si verifica un criterio di fine apprendimento (per esempio, dopo un numero finito di iterazioni, o quando l'errore non scende sotto una soglia prefissata). Il processo continua fino a quando l'errore non raggiunge il valore più basso [7].

1.1.2 Apprendimento non supervisionato

Nell'apprendimento non supervisionato, al contrario di quello supervisionato si hanno dei dati senza etichetta. Le tecniche di apprendimento non supervisionato mirano a estrarre, in modo automatico, della conoscenza a partire da basi di dati, e questo avviene senza una specifica conoscenza dei contenuti da analizzare. Un esempio tipico di questi algoritmi lo si ha nei motori di ricerca. Questi programmi, data una o più parole chiave, sono in grado di creare una lista di link rimandanti alle pagine che l'algoritmo di ricerca ritiene attinenti alla ricerca effettuata [8]. I principali algoritmi utilizzati in ambito non supervisionato fanno riferimento a tecniche di clustering e a regole di associazione.

1.1.3 Apprendimento per rinforzo

Il terzo tipo di apprendimento automatico è l'apprendimento per rinforzo. L'obiettivo di questo tipo di apprendimento è quello di costruire un sistema che attraverso le interazioni con l'ambiente migliori le proprie performance [8].

Per poter migliorare le funzionalità del sistema vengono introdotti dei rinforzi, ovvero segnali di ricompensa. Questo rinforzo non è dato dalle etichette, ma è una misurazione sulla qualità delle azioni intraprese dal sistema. Per questo motivo non può essere assimilato all'apprendimento supervisionato. Potremmo trovare questo tipo di apprendimento ad esempio nell'addestramento di un sistema per il gioco degli scacchi.

Inizialmente le mosse saranno del tutto casuali e senza una logica. Dal momento in cui il sistema riceverà dei feedback positivi, come ad esempio nel caso in cui mangi una pedina avversaria, allora riceverà un peso maggiore e conseguentemente un rinforzo positivo su quell'azione. Contrariamente in caso di azione negativa, il valore dei pesi su quell'azione andrà in decremento.

Conseguentemente a questi rinforzi, il sistema darà maggior peso alle mosse che gli hanno portato maggiori benefici e tenderà a replicare lo stesso comportamento su nuove mosse future.

1.1.4 Apprendimento semi supervisionato

Può essere visto come un quarto tipo di apprendimento automatico. In questo caso, al contrario dell'apprendimento non supervisionato, abbiamo che di tutti i dati presenti nel training set, solo pochi di essi sono stati etichettati [9].

Capitolo 2

Dataset

Un dataset è una collezione di dati, comunemente corrisponde al contenuto di una singola tabella nella quale una colonna rappresenta una caratteristica e ogni riga una singola osservazione o istanza. Gli algoritmi di Machine Learning utilizzano i dataset per "apprendere", tanto più un dataset è ricco di osservazioni tanto più l'algoritmo che lo utilizza sarà in grado di fornire prestazioni e accuratezza in output.

2.1 Iris

Per prendere familiarità con gli strumenti e gli algoritmi da utilizzare, ho fatto riferimento al dataset Iris. Esso è un dataset multivariato introdotto da Ronald Fisher nel 1936. Consiste in 150 istanze di fiori iris classificate secondo tre specie: Setosa, Virginica e Versicolor. Le variabili considerate sono lunghezza e larghezza di sepalo e petalo (cfr. Figura 9).

	sepal.length	sepal.width	petal.length	petal.width	variety
2	5.1	3.5	1.4	.2	Setosa
3	4.9	3	1.4	.2	Setosa
4	4.7	3.2	1.3	.2	Setosa
5	4.6	3.1	1.5	.2	Setosa
6	5	3.6	1.4	.2	Setosa
7	5.4	3.9	1.7	.4	Setosa

Figura 9: Visualizzazione di un estratto del dataset Iris

2.2 Incidenti Stradali

Apprese le basi del Machine Learning ho iniziato a lavorare con il dataset "Incidenti Stradali", questo dataset è stato fornito da medici legali e raccoglie, tristemente, i decessi causati appunto da incidenti stradali. Sono state raccolte 131 istanze e ognuna rappresenta una persona deceduta. Il dataset è organizzato in vari livelli di dettaglio. Un primo livello è composto dalle caratteristiche basilari:

- numero del verbale, verrà usato come indice univoco,
- data del decesso,
- sesso,
- anni,
- peso,
- altezza,
- BMI, indice di massa corporea.

Successivamente sono descritti i totali delle rotture avvenute nei distretti di:

- testa,
- torace,
- addome,
- scheletro.

Ogni distretto è poi diviso più nello specifico in singole ossa (cfr. Figura 10), ognuna con etichetta che va da 0, nessuna lesione, a 4, lesione massima .

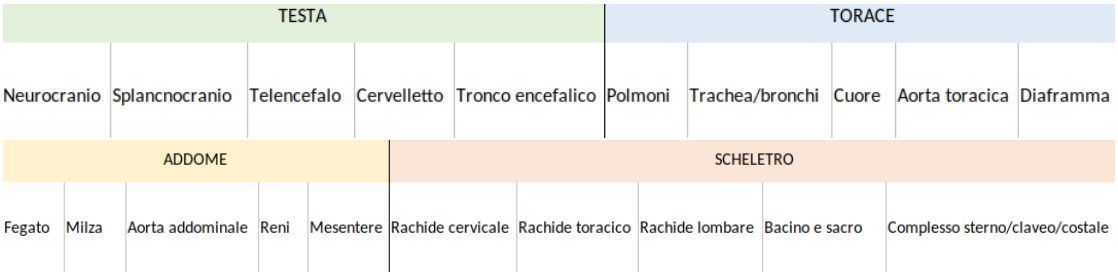

Figura 10: Suddivisione caratteristiche dettagliate

Il dataset ha inoltre due ulteriori livelli di dettaglio. Esso contiene infatti anche dati più specifici riguardanti la frattura o meno di singole ossa raggruppate in:

- cranio,
- rachide,
- torace - gabbia toracica,
- bacino,
- arti superiori,
- arti inferiori.

In ultimo è registrato il mezzo che ha investito la persona. Questo è il label set, la y dei modelli descritti al capitolo precedente. Tutti gli esperimenti effettuati in fase di studio sono incentrati sul cercare di classificare il tipo di mezzo, leggero ovvero auto, etichettato con 0, o pesante, con etichetta 1, che ha investito l'individuo.

Durante il tirocinio mi sono limitato a considerare le caratteristiche basilari, i totali e quelle riportate in Figura 10.

2.3 Riduzione della Dimensionalità

Nei problemi di classificazione di Machine Learning ci sono spesso molte caratteristiche da tenere in considerazione, basti pensare che il dataset "Incidenti Stradali" al completo ha circa 350 caratteristiche. Per dataset con molte dimensioni la riduzione della dimensionalità è un pratica importante svolta prima di applicare algoritmi di Machine Learning per evitare l'effetto chiamato *curse of dimensionality*, "maledizione della dimensionalità"[10]. Il problema sorge nel momento in cui le caratteristiche sono molte e lo spazio aumenta così rapidamente che i dati disponibili diventano radi, sparsi. In ambito statistico questa scarsità è problematica in quanto i dati necessari a supportare il risultato aumentano in modo esponenziale. Inoltre dati con molte dimensioni possono causare problemi di *overfitting*, ovvero il sistema in qualche modo impara e memorizza il risultato, si adatta (*fitting*) troppo bene (*over*) ai dati di training perdendo di generalità. Il modello sembra perfetto per i dati di training ma quando si prova ad applicarlo ai dati di test si verificano molti errori. Per questo motivo viene effettuata una riduzione della dimensionalità in modo da eliminare le caratteristiche più irrilevanti e lasciare spazio a quelle rilevanti.

Gli algoritmi presi in considerazione durante il tirocinio sono PCA e t-SNE, come descritto nei paragrafi che seguono.

2.3.1 PCA

L'idea alla base del pricipal component analysis (PCA), o analisi delle componenti principali, è di ridurre la dimensione di un dataset che contiene un grande numero di caratteristiche correlate, mantenendo il più possibile la varianza presente nel dataset. Quello che si fa è selezionare le componenti principali, che non sono correlate tra di loro e che sono ordinate in modo tale che le prime mantengano la maggior parte di varianza presente in tutte le caratteristiche iniziali [11].

L'idea è di trattare le caratteristiche come una matrice A ottenuta mettendo uno sopra l'altro tutti i vettori nello spazio originale, e trovare gli autovettori per AA^T o A^TA.

La matrice di questi autovettori può essere vista come una rotazione nello spazio.

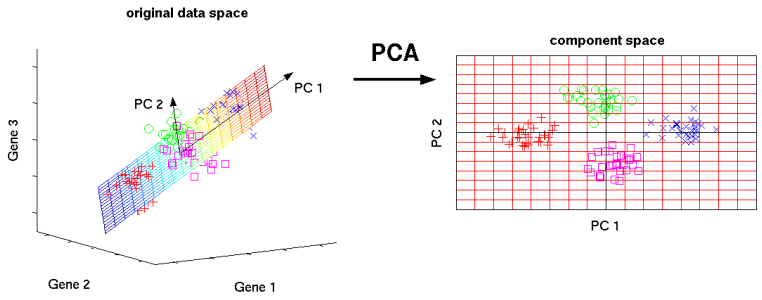

Figura 11: Dimensioni trovate tramite PCA [12]

Quando si applica questa trasformazione dei dati, l'asse corrispondente all'autovettore principale è quello nel quale i dati sono più sparsi, ovvero l'asse nel quale la varianza dei dati è massimizzata.

Detto in altro modo, i punti possono essere osservati meglio se distesi lungo questo asse, con piccole deviazioni da esso. Allo stesso modo, l'asse corrispondente al secondo autovettore (l'autovettore corrispondente al secondo autovalore più grande) è l'asse lungo il quale la varianza delle distanze dal primo asse è maggiore, e così via fino al numero di dimensioni che si è scelto di mantenere.

Ma come funziona esattamente? Per prima cosa si costruisce la matrice dei dati A nel quale ogni colonna corrisponde ad una caratteristica e ogni riga ad una istanza. Successivamente si calcola la media di ogni colonna e la si sottrae ad ogni elemento della matrice, ottenendo così la matrice B.

A questo punto viene calcolata la matrice di covarianza. Ogni valore di covarianza assume un segno: se positivo significa che c'è una correlazione positiva, al contrario, se negativa, c'è una correlazione inversa.

La matrice appena calcolata serve per poter calcolare gli autovalori, operatori che si basano sul fatto che moltiplicando o dividendo un vettore cambia solo la sua

lunghezza e non la direzione. Un autovalore λ è tale se risolve

$$Av = \lambda v, \tag{9}$$

con A matrice quadrata e v vettore che viene definito autovettore.

Se per esempio

$$A = \begin{bmatrix} 1 & 1 \\ 8 & 1 \end{bmatrix} \text{ e } v = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \tag{10}$$

è facile verificare che $\lambda = 5$ rappresenta un autovalore in quanto risolve l'equazione (9).

Per calcolare gli autovalori di una matrice è possibile risolvere la seguente equazione

$$\det(A - \lambda I) = 0, \tag{11}$$

dove I è la matrice identità e det rappresenta il determinante della matrice.

Avendo trovato il valore di λ è possibile calcolare gli autovettori corrispondenti agli autovalori risolvendo l'equazione (9) e trovando v:

Gli autovettori trovati formeranno gli assi del nuovo sistema di riferimento di dimensioni più piccole rispetto a quello iniziale. Tuttavia gli autovettori definiscono solo le direzioni degli assi, di conseguenza per decidere quale autovettore vogliamo eliminare per il nostro sottospazio di dimensione inferiore dobbiamo vedere gli autovalori. Quello che faremo sarà eliminare gli autovettori con autovalori più bassi in quanto hanno il minor numero di informazioni sulla distribuzione dei dati al loro interno. Quello che viene fatto è ordinare gli autovalori dal più grande al più piccolo e prendere i primi k autovettori corrispondenti.

In ultimo dobbiamo formare le componenti principali, per farlo calcoliamo:

$$CP = W^T \cdot B^T, \tag{12}$$

dove CP è la matrice costituita dalle componenti principali, W^T è la matrice formata usando gli autovettori che abbiamo scelto di mantenere precedentementee di cui consideriamo la trasposta, B^T è la trasposta della matrice B precedentemente calcolata. Quello che si ottiene sono i dati originali ma inseriti nello spazio considerando come assi di riferimento gli autovettori calcolati.

Sebbene PCA sia uno degli algoritmi più famosi e usati poiché è veloce, facile da usare e intuitivo ha un problema: è una tecnica lineare, presuppone cioè che le componenti principali siano una combinazione lineare delle caratteristiche originali. Nel caso non sia così non fornirà risultati efficienti. Il paragrafo seguente illustra una tecnica non lineare di riduzione della dimensionalità che tende ad adattarsi meglio ai dataset disponibili.

2.3.2 t-SNE

t-SNE, o t-Distributed Stochastic Neighbor Embedding, è una tecnica di riduzione della dimensionalità non lineare [13]. Utilizza una struttura locale nella quale punti aventi molte dimensioni vengono mappati nello spazio usando un numero inferiore di dimensioni, ciò avviene in modo che punti vicini nello spazio originale vengano trasformati in punti vicini nello spazio con dimensione minore.

L'algoritmo prevede tre passi.

- Il primo punto calcola una misura di similarità tra i punti nello spazio a molte dimensioni, e centra una distribuzione gaussiana sopra a ogni punto x_i. Viene poi calcolato il valore della densità corrispondente per tutti gli altri punti x_j. Dopo un processo di normalizzazione, si ottiene un insieme di valori di probabilità P_{ij} che sono che sono proporzionali ai valori di similarità. La distribuzione può essere manipolata usando quella che viene chiamata perplessità, la quale influenza la varianza della distribuzione, ovvero quanto è ampia la curva, che a sua volta influenza il numero dei vicini di ogni punto. Il normale range della perplessità è tra 5 e 50.

- Il secondo punto è simile al primo, ma al posto di una distribuzione gaussiana fa riferimento ad una distribuzione t-Student. Essa ci fornisce un secondo set di probabilità Q_{ij} in uno spazio ridotto. La distribuzione t-Student ha code più lunghe rispetto alla gaussiana(cfr. Figura 12), e questo permette una migliore visualizzazione delle distanze tra i punti.

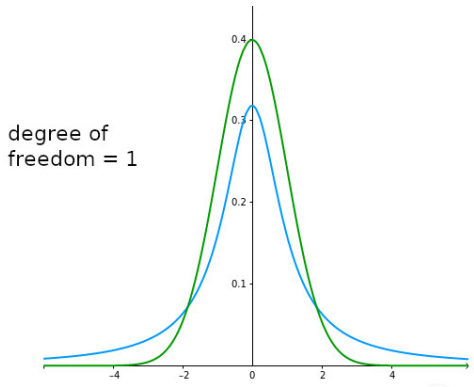

Figura 12: Grafici della densità delle distribuzioni gaussiana (curva verde), e t-Student (curva blu). La forma della distribuzione t-Student è data dai gradi di libertà che si riferiscono al numero di osservazioni indipendenti all'interno dei dati

- Nel terzo passo si finalizza il calcolo delle probabilità Q_{ij} nello spazio ridotto in modo che riflettano il meglio possibile le probabilità P_{ij} relative allo spazio con più dimensioni. Ciò viene effettuato misurando la differenza tra le due distribuzioni di probabilità usando la divergenza di Kullback-Liebler (KL) che confronta efficacemente valori di P_{ij} e Q_{ij} di grandi dimensioni. Per minimizzare la divergenza di KL tra le due dimensioni viene utilizzata la discesa del gradiente [13].

Nel Capitolo 3 verranno applicate PCA e t-SNE per cercare di aumentare le prestazioni di classificazione dei modelli usati.

Capitolo 3

Esperimenti

Durante il tirocinio ho effettuato diversi esperimenti a partire dal dataset descritto al capitolo precedente prendendo in considerazione i diversi livelli di dettaglio descritti nel Paragrafo 2.2. In questo capitolo verranno discussi i passaggi necessari per passare dal dataset ad una previsione dell'etichetta corrispondente il più precisa possibile.

3.1 Preprocessing

La prima azione da svolgere su dati da analizzare consiste in un processo di preprocessing. I dati vengono trasformati o codificati in modo tale che sia più facile per l'algoritmo, che verrà applicato in un secondo momento, analizzarli e interpretarli. Il preprocessing richiede normalmente diversi passaggi.

- E' necessario gestire i valori nulli prima di fornire i dati all'algoritmo poiché nessun modello è in grado di gestirli autonomamente. Il pacchetto *scikit-learn* mette a disposizione la classe *Imputer* per sostituire i valori mancanti.
- Standardizzazione dei valori. Questo viene fatto poiché all'interno del dataset potrebbero esserci caratteristiche inserite con scale diverse. Prendiamo per esempio un dataset con all'interno due caratteristiche numeriche, "metratura" e "costo": non sono sulla stessa scala, "metratura" si misura in metri quadrati, mentre "costo" in euro.
- Gestire le variabili categoriche, variabili che indicano l'appartenenza a una specifica categoria.

Le operazioni di preprocessing svolte durante il tirocinio hanno richiesto di controllare che il numero del verbale di ogni istanza fosse univoco e che le caratteristiche

basilari descritte nel Paragrafo 2.2 avessero un valore sensato con quello che stavano rappresentando, nello specifico "Anni" fosse compreso tra 1 e 95, estremi inclusi, "Peso" tra 30 e 120, "Altezza" tra 1 metro e 2.10 metri, "BMI" tra 10.0, estrema magrezza, e 50.0, obesità gravissima e infine che tutti i valori che rappresentavano la gravità della frattura avessero valori compresi tra 0 e 4. Successivamente ho scalato i dati usando tre diversi scaler del pacchetto *preprocessing* di *scikit-learn*:

- *StandardScaler*,
- *MinMaxScaler*,
- *RobustScaler*.

3.1.1 Scalare i dati

Il processo di scalatura consente di standardizzare i dati. Questa pratica è importante, non solo per riuscire a confrontare caratteristiche con unità di misura diverse, come detto nel Paragrafo 3.1, ma anche perché molti algoritmi lavorano meglio con dati impostati in questo modo. Analizziamo ora le tre diverse tecniche usate.

StandardScaler standardizza i dati sottraendo la media μ e dividendo per la deviazione standard σ [14], ovvero:

$$z_i = \frac{x_i - \mu}{\sigma}. \tag{13}$$

Questo viene fatto poiché molti algoritmi assumono che tutte le caratteristiche siano centrate attorno a 0 e abbiano varianza dello stesso ordine. Se una caratteristica presenta una varianza di ordini di grandezza superiore rispetto ad altre, potrebbe influenzare troppo la funzione obiettivo dell'algoritmo usato e rendere lo stimatore incapace di apprendere correttamente da altre caratteristiche [14].

MinMaxScaler ridimensiona singolarmente ogni caratteristica in un determinato intervallo [14], per esempio tra 0 e 1. Si applica la seguente formula:

$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{norm}}}. \tag{14}$$

RobustScaler scala le caratteristiche usando statistiche robuste rispetto agli outlier [14]. Questo scaler rimuove la mediana e scala i dati in base all'intervallo interquantile; è utile perché i valori anomali possono spesso influenzare la media e la varianza del campione in modo negativo.

3.2 Model Selection

Il processo di *model selection* è la ricerca del modello più performante dato un set di modelli prodotti da diverse scelte di iperparametri. [15]. Durante il tirocinio ho avuto modo di lavorare con i modelli riportati nel Paragrafo 1.1.1, dei quali molti presentano diversi iperparametri con cui è possibile modificare il comportamento del modello.

3.2.1 Scelta degli iperparametri

In ogni modello sono presenti diversi iperparametri, ognuno dei quali ha scopi ben precisi. Iniziamo con gli iperparametri di SVC: per un valore grande di C è assegnato un grande peso agli errori di classificazione, quindi diminuisce il bias e aumenta la varianza. Per valori di C piccoli invece si ha un bias più grande e una minore varianza. Un altro iperparametro di SVC è il *Kernel* che può essere di diversi tipi: *linear*, *polynomial*, *radial basis function*. Associato al kernel di tipo polinomiale c'è inoltre un altro iperparametro chiamato *degree*. Oni kernel può inoltre avere un proprio specifico parametro, come per esempio il grado relativo al kernel polinomiale o la deviazione standard relativa a quello gaussiano. Di seguito viene riportata la griglia con gli iperparametri valutati, unitamente ai valori presi in considerazione.

```
c_space = np.logspace(-4, 3, 10)
gamma_space = np.logspace(-4, 3, 10)

model_selection_grid_SVC = [
{'C': c_space, 'kernel': ['linear']},
{'C': c_space, 'gamma': gamma_space, 'kernel': ['rbf']},
{'C': c_space, 'gamma': ['auto', 'scale'], 'kernel': ['rbf']},
{'C': c_space, 'degree': [2, 3, 5, 9], 'kernel': ['poly'],
'gamma': ['auto']},
]
```

Per quanto riguarda Decision Tree abbiamo diversi iperparametri: *criterion*, funzione che viene usata per misurare la qualità della divisione nell'albero e può essere di due tipi ("gini" o "entropy") [14]; *max_depth*, ovvero la profondità massima dell'albero [14]; *max_features*, il numero di caratteristiche da considerare quando si cerca la migliore divisione [14]; *max_leaf_nodes*, imposta un limite massimo ai nodi foglia.

```
model_selection_grid_DT = {'criterion': ['gini', 'entropy'],
'max_leaf_nodes': [None, 2, 5, 10, 50, 100],
'max_features': [None, 'sqrt', 'log2'],
'max_depth': [None, 2, 5, 10]}
```

In Random Forest troviamo gli stessi iperparametri di Decision Tree con l'aggiunta di *n_estimators* che rappresenta il numero di alberi nella foresta.

```
model_selection_grid_RF = {'n_estimators': [5, 10, 50, 100, 200],
'criterion': ['gini', 'entropy'],
'max_leaf_nodes': [None, 2, 5, 10, 50, 100],
'max_features': [None, 'sqrt', 'log2'],
'max_depth': [None, 2, 5, 10]}
```

In Guaussian Naive Bayes e Linear Discriminant Analysis non ho effettuato model selection.

Infine nelle Reti Neurali Multi-Strato ho considerato tre iperparametri: *max_iter* che specifica il numero massimo di iterazioni nell'algoritmo di apprendimento; *hidden_layer_sizes* che rappresenta il numero di neuroni in un livello hidden (ad esempio, $hidden\_layer\_sizes = [2]$ indica che c'è un livello hidden con due neuroni, mentre $hidden\_layer\_sizes = [4, 4]$ significa che ci sono due livelli hidden ognuno composto da quattro neuroni); *activation*, che indica la funzione di attivazione (vedi Paragrafo 1.1.1.6).

```
model_selection_grid_MLP = {'max_iter': [5000],
'hidden_layer_sizes': [[2], [4], [6], [10], [20], [4, 4], [10, 10]],
'activation': ['identity', 'logistic', 'tanh', 'relu']}
```

Per effettuare la model selection ho utilizzato *GridSearchCV* fornito dal pacchetto *scikit-learn*, esso effettua una ricerca esaustiva dei valori degli iperparametri del modello, selezionato tramite il parametro *estimator*, specificati all'interno del parametro *param_grid*. La ricerca degli iperparametri del modello può essere ottimizzata effettuando una cross-validation (vedi Paragrafo 3.4) sulla griglia degli iperparametri passata in input.

3.3 Errore di Generalizzazione

L'errore di generalizzazione è una misura di quanto accuratamente un algoritmo è in grado di prevedere i valori delle etichette per nuovi dati. Un modello viene "allenato" usando degli esempi di cui è nota l'etichetta che si vuole predire. L'obiettivo è raggiungere uno stato in cui l'algoritmo sia in grado di predire le etichette per altri esempi che non ha avuto in input durante la fase di training. Si dice che così facendo l'algoritmo sarà in grado di *generalizzare*. L'errore di generalizzazione può essere minimizzato evitando l'"overfitting" nell'algoritmo di apprendimento, ovvero il caso in cui l'algoritmo si adatta a caratteristiche che sono specifiche solo del training set e che non hanno riscontro nel resto dei dati. Questo fenomeno si verifica quando i dati di training non sono sufficienti, non sono rappresentativi della realtà oppure quando l'algoritmo prende in considerazione delle caratteristiche del training set irrilevanti con l'etichetta. Perciò, in presenza di "overfitting",

le prestazioni sui dati di allenamento aumenteranno, mentre quelle sui dati non visionati saranno peggiori.

Per valutare se i risultati di un algoritmo di apprendimento soffrano o meno di overfitting si può utilizzare la tecnica di cross-validation, descritta nel Paragrafo seguente.

3.4 Cross Validation

La cross-validation, o k-fold validation, è una tecnica che utilizza parte dei dati disponibili per addestrare il modello e una parte per testarlo [16]. Viene specificato il numero di "fold", ovvero in quante parti dividere il training set. Se per esempio consideriamo tre fold, otterremo una divisione del training set come quella illustrata in Figura 13.

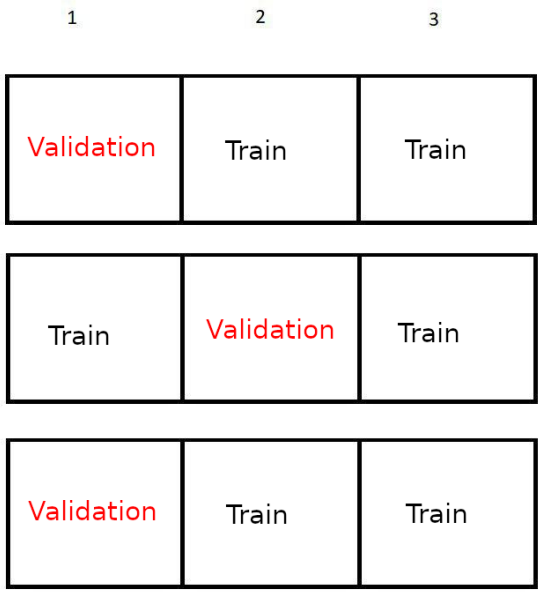

Figura 13: Cross validation con 3 fold

Un set viene assegnato a "validation test", mentre i restanti 2 compongono il "training set". L'algoritmo viene addestrato usando il training set così composto e testato sul validation. Questo procedimento viene eseguito per tutte le combinazioni possibili dei set. Negli esperimenti descritti nel Paragrafo 3.5 ho usato una cross-validation annidata per stimare l'errore di generalizzazione e fare la model selection, usando in entrambi i casi 9 come numero di fold per tutti gli algoritmi nel quale erano presenti degli iperparametri 3.2, quindi per Gaussian Naive Bayes

e Linear Discriminant Analysis ho effettuato solo una cross-validation singola per stimare l'errore di generalizzazione.

3.5 Risultati Ottenuti

Visualizziamo ora i risultati ottenuti: nelle tabelle che seguono, ogni riga corrisponde ad un esperimento effettuato usando un algoritmo di apprendimento diverso, mentre ogni colonna corrisponde al tipo di dataset usato per effettuare l'addestramento. Il dataset "Details" ha una dimensione iniziale di 20. Il numero affianco a "PCA" e "t-SNE" indica il numero di dimensioni dopo l'applicazione della relativa tecnica di riduzione della dimensionalità. I valori della tabella indicano l'accuratezza del classificatore: se per esempio troviamo 0.77 significa che nel 77% dei casi il classificatore ha prodotto l'etichetta corretta. In azzurro vengono evidenziati gli score massimi del singolo modello mentre in arancione lo score massimo della tabella.

	Totali	Totali_with_BMI	Totali_with_DATA	Totali_with_DATA_and_BMI
SVC	0.66	0.66	0.77	0.73
DT	0.56	0.56	0.61	0.63
RF	0.64	0.63	0.57	0.59
NB	0.69	0.68	0.72	0.75
LD	0.71	0.68	0.72	0.72
MLP	0.69	0.69	0.74	0.72

	Details	Details PCA_5	Details PCA_10	Details PCA_13	Details PCA_15	Details TSNE_13	Details TSNE_15
SVC	0.68	0.63	0.68	0.67	0.71	0.51	0.53
DT	0.68	0.60	0.67	0.64	0.57	0.43	0.50
RF	0.67	0.59	0.69	0.70	0.63	0.51	0.55
NB	0.63	0.61	0.66	0.67	0.65	0.56	0.57
LD	0.63	0.64	0.60	0.59	0.59	0.48	0.49
MLP	0.63	0.62	0.61	0.62	0.65	0.39	0.47

Tabella 1: Risultati ottenuti StandardScaler, in azzurro vengono evidenziati gli score massimi ottenuti uno specifico modello usato come indice delle righe mentre in arancione lo score massimo assoluto

	StandardScaler
Totali	0.71
Totali_with_BMI	0.69
Totali_with_DATA	0.77
Totali_with_DATA_and_BMI	0.75
Details	0.68
Details_reduce_PCA_5_dim	0.64
Details_reduce_PCA_10_dim	0.69
Details_reduce_PCA_13_dim	0.70
Details_reduce_PCA_15_dim	0.71
Details_reduce_TSNE_13_dim	0.56
Details_reduce_TSNE_15_dim	0.57

Tabella 2: Risultati massimi, tra tutti i modelli utilizzati, ottenuti dai dataset usati come indice delle righe

	Totali	Totali_BMI	Totali_DATA	Totali_DATA_and_BMI
SVC	0.66	0.62	0.73	0.74
DT	0.61	0.55	0.65	0.63
RF	0.60	0.62	0.62	0.61
NB	0.69	0.68	0.72	0.75
LD	0.71	0.68	0.72	0.72
MLP	0.62	0.64	0.71	0.76

	Details	Details PCA_5	Details PCA_10	Details PCA_13	Details PCA_15	Details TSNE_13	Details TSNE_15
SVC	0.74	0.70	0.64	0.67	0.70	0.58	0.42
DT	0.70	0.61	0.60	0.56	0.59	0.51	0.56
RF	0.67	0.65	0.66	0.71	0.67	0.46	0.52
NB	0.63	0.67	0.65	0.67	0.67	0.57	0.58
LD	0.63	0.65	0.61	0.60	0.62	0.43	0.39
MLP	0.65	0.64	0.64	0.60	0.63	0.51	0.43

Tabella 3: Risultati ottenuti usando MinMaxScaler, in azzurro vengono evidenziati gli score massimi ottenuti uno specifico modello usato come indice delle righe mentre in arancione lo score massimo assoluto

	MinMaxScaler
Totali	0.71
Totali_with_BMI	0.68
Totali_with_DATA	0.73
Totali_with_DATA_and_BMI	0.76
Details	0.74
Details_reduce_PCA_5_dim	0.70
Details_reduce_PCA_10_dim	0.66
Details_reduce_PCA_13_dim	0.71
Details_reduce_PCA_15_dim	0.70
Details_reduce_TSNE_13_dim	0.58
Details_reduce_TSNE_15_dim	0.58

Tabella 4: Risultati massimi, tra tutti i modelli utilizzati, ottenuti dai dataset usati come indice delle righe

	Totali	Totali_BMI	Totali_DATA	Totali_DATA_and_BMI
SVC	0.66	0.68	0.77	0.72
DT	0.59	0.60	0.62	0.65
RF	0.62	0.63	0.56	0.61
NB	0.69	0.68	0.72	0.75
LD	0.71	0.68	0.72	0.72
MLP	0.70	0.67	0.70	0.66

	Details	Details PCA_5	Details PCA_10	Details PCA_13	Details PCA_15	Details TSNE_13	Details TSNE_15
SVC	0.68	0.63	0.63	0.66	0.66	0.56	0.47
DT	0.65	0.63	0.61	0.64	0.63	0.45	0.56
RF	0.65	0.65	0.64	0.64	0.66	0.52	0.49
NB	0.63	0.67	0.67	0.67	0.65	0.54	0.56
LD	0.63	0.63	0.60	0.64	0.67	0.43	0.42
MLP	0.56	0.63	0.60	0.65	0.60	0.49	0.46

Tabella 5: Risultati ottenuti usando RobustScaler, in azzurro vengono evidenziati gli score massimi ottenuti uno specifico modello usato come indice delle righe mentre in arancione lo score massimo assoluto

	RobustScaler
Totali	0.71
Totali_with_BMI	0.68
Totali_with_DATA	0.77
Totali_with_DATA_and_BMI	0.75
Details	0.68
Details_reduce_PCA_5_dim	0.67
Details_reduce_PCA_10_dim	0.67
Details_reduce_PCA_13_dim	0.67
Details_reduce_PCA_15_dim	0.67
Details_reduce_TSNE_13_dim	0.56
Details_reduce_TSNE_15_dim	0.56

Tabella 6: Risultati massimi, tra tutti i modelli utilizzati, ottenuti dai dataset usati come indice delle righe

	Details PCA_5	Details PCA_10	Details PCA_13	Details PCA_15	Details TSNE_13	Details TSNE_15
SVC	0.65	0.66	0.68	0.66	0.57	0.54
DT	0.60	0.55	0.59	0.51	0.50	0.52
RF	0.66	0.64	0.60	0.64	0.53	0.55
NB	0.67	0.64	0.66	0.65	0.52	0.55
LD	0.64	0.62	0.58	0.62	0.47	0.51
MLP	0.66	0.65	0.60	0.58	0.43	0.53

Tabella 7: Risultati ottenuti senza usare uno scaler di dati, in azzurro vengono evidenziati gli score massimi ottenuti uno specifico modello usato come indice delle righe mentre in arancione lo score massimo assoluto

	No_Scaler
Details_reduce_PCA_5_dim	0.67
Details_reduce_PCA_10_dim	0.66
Details_reduce_PCA_13_dim	0.68
Details_reduce_PCA_15_dim	0.66
Details_reduce_TSNE_13_dim	0.57
Details_reduce_TSNE_15_dim	0.55

Tabella 8: Risultati massimi, tra tutti i modelli utilizzati, ottenuti dai dataset usati come indice delle righe

	StandardScaler	MinMaxScaler	RobustScaler	No_Scaler
Totali	0.71	0.71	0.71	NaN
Totali_with_BMI	0.69	0.68	0.68	NaN
Totali_with_DATA	0.77	0.73	0.77	NaN
Totali_with_DATA_and_BMI	0.75	0.76	0.75	NaN
Details	0.68	0.74	0.68	NaN
Details_reduce_PCA_5_dim	0.64	0.70	0.67	0.67
Details_reduce_PCA_10_dim	0.69	0.66	0.67	0.66
Details_reduce_PCA_13_dim	0.70	0.71	0.67	0.68
Details_reduce_PCA_15_dim	0.71	0.70	0.67	0.66
Details_reduce_TSNE_13_dim	0.56	0.58	0.56	0.57
Details_reduce_TSNE_15_dim	0.57	0.58	0.56	0.55

Tabella 9: Visualizzazione generale dei risultati massimi ottenuti tra tutti i dataset con i vari scaler

	StandardScaler	MinMaxScaler	RobustScaler	NoScaler	Media
SVC	0.77	0.74	0.77	0.68	0.74
DT	0.68	0.70	0.65	0.60	0.65
RF	0.70	0.71	0.66	0.66	0.68
NB	0.75	0.75	0.75	0.67	0.73
LD	0.72	0.72	0.72	0.64	0.70
MLP	0.74	0.76	0.70	0.66	0.71

Tabella 10: Visualizzazione generale dei risultati massimi ottenuti tra tutti i modelli con i vari scaler e la loro media

3.6 Analisi dei Risultati

Osservando i risultati ci si accorge di come non ci sia uno scaler che si comporta meglio in maniera assoluta. Mentre si vede come SVC ha l'accuratezza media, tra tutti gli scaler, più alta con 0.74, segue NB con 0.73 e a seguire gli altri con accuratezza compresa tra 0.71 e 0.65. La riduzione della dimensionalità eseguita con PCA e t-SNE non comporta un aumento delle prestazioni, anzi molto spesso genera un risultato meno prestante rispetto al dataset di partenza "Details". Si può inoltre notare che, all'interno delle tabelle, non sono presenti esperimenti effettuati usando "Details" con riduzione della dimensionalità a 5 e 10 dimensioni tramite t-SNE. Questo perché durante l'addestramento t-SNE non è riuscito a convergere e quindi l'addestramento non è riuscito. In ogni caso l'accuratezza dei risultati ottenuti usando t-SNE non supera mai lo 0.58. PCA si comporta meglio, generando accuratezze nel range 0.64 - 0.71. Si ottengono i risultati migliori considerando i dataset "Totali", in particolare i dataset "Totali_with_DATA" e "Totali_with_DATA_and_BMI" sono quelli che generano le accuratezze più elevate. In particolare l'accuratezza più alta in assoluto è 0.77, generata usando "Totali_with_DATA" e indifferentemente StandardScaler o RobustScaler.

3.7 Tecnologie Utilizzate

Il codice per eseguire gli esperimenti descritti in questo documento è stato scritto usando il linguaggio *Python* 3.7.4, unitamente ai seguenti pacchetti:

- *skikit-learn* [14], versione 0.22.1
- *numpy*, versione 1.18.1
- *pandas*, versione 0.25.3

Ho inoltre utlizzato *Jupyter* come ambiente di lavoro [17].

Conclusioni

Lo scopo del tirocinio era quello di applicare il paradigma del Machine Learning per riuscire a classificare che tipo di mezzo avesse investito una persona deceduta a causa dell'impatto con un veicolo. Il lavoro si è inizialmente concentrato sull'apprendere le basi del Machine Learning, per poi passare all'applicazione pratica usando un dataset fornito da medici legali. L'accuratezza ottenuta dai modelli non è altissima ma ho tenuto conto solo dei livelli di dettaglio meno specifici, come descritto nel Paragrafo 2.2. Un possibile sviluppo futuro potrebbe essere quello di includere tutti i livelli di dettaglio che rappresentano la rottura o meno di ossa del corpo molto specifiche in modo da estendere il numero delle dimensioni da 30, quelle usate durante il tirocinio, a tutte le 350 dimensioni presenti nel dataset fornito.

Un altro possibile sviluppo potrebbe essere quello di estendere la griglia di iperparametri dal quale *GridSearchCV* possa ricercare quelli che meglio si adattano ai dati. Entrambi questi sviluppi richiederebbero un aumento notevole del tempo di calcolo, che però potrebbe portare a un aumento nell'accuratezza dei modelli ottenuti.

Bibliografia

[1] Fredrik Lindsten Thomas B. Schön Andreas Lindholm, Niklas Wahlström. *Supervised Machine Learning*. Cambridge University Press, 2020.

[2] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining (2nd Edition)*. Pearson, 2nd edition, 2018.

[3] Andrew Ng. Cs229 lecture notes. url http://cs229.stanford.edu/notes/cs229-notes3.pdf/, October 2018.

[4] Oded Maimon and Lior Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag, Berlin, Heidelberg, 2005.

[5] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[7] K.-L Du and M.N.s Swamy. *Neural Networks and Statistical Learning*. 01 2019.

[8] Zoubin Ghahramani. Unsupervised learning. In *Advanced Lectures on Machine Learning*, pages 72–112. Springer-Verlag, 2004.

[9] Philippe Thomas. Semi-supervised learning by olivier chapelle, bernhard schölkopf, and alexander zien (review). *IEEE Trans. Neural Networks*, 20(3):542, 2009.

[10] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning : from theory to algorithms*. 2014.

[11] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[12] Matthias Scholz. *Approaches to analyse and interpret biological profile data.* Ph.d. thesis, University of Potsdam, Germany, 2006.

[13] Geoffrey Hinton Laurens van der Maaten. Visualizing data using t-sne. *Journal of Machine Learning Research 9*, 2008.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] Sebastian Raschka. Model evaluation, model selection, and algorithm selection in machine learning. *ArXiv*, abs/1811.12808, 2018.

[16] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning*. Springer-Verlag New York, 2009.

[17] Jupyter. https://jupyter.org/.