UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Laurea Triennale in Ingegneria Informatica

Riconoscimento delle emozioni basato su EEG attraverso ensemble di alberi decisionali

Relatore:

Prof: Antonio Luca Alfeo

Prof: Mario G.C.A. Cimino

Candidato:

Gabriele Marcuccetti

ANNO ACCADEMICO 2021/2022

Indice

1 Abstract

Avere maggiori conoscenze dal punto di vista del funzionamento delle emozioni umane sarebbe vantaggioso da molti punti di vista, sia commerciali, per creare contenuti appositi a seconda del lato emotivo dei soggetti, sia scientifico, per comprendere meglio il funzionamento del cervello. Tuttavia, rilevare dei pattern tramite misure sperimentali per comprendere come le emozioni agiscano non è facile, né dal punto di vista dell'attrezzatura necessaria a compiere le varie misurazioni, né da quello della successiva processazione dei valori misurati. In questa tesi si utilizza un dataset contenente varie misurazioni relative a onde celebrali in risposta a stimoli visivi e uditivi. Si ricorre all'utilizzo del machine learning, in particolare di tecniche di regressione prima, e classificazione poi, per ricercare pattern fra il valore che viene registrato, relativo ad una grandezza fisica, e l'etichetta che l'utente ha assegnato a quello stimolo. Si tratta dunque con algoritmi di apprendimento supervisionato, fruiti attraverso varie librerie low-code, sulla piattaforma Google Colab. Google Colab è una piattaforma online, accessibile a tutti senza alcun iter particolare, la quale fornisce accesso ad alcune CPU e GPU di una macchina virtuale dedicata e permette di scrivere ed eseguire codice Python direttamente sul cloud, sfruttando la potenza di calcolo fornita da Google, il cui vantaggio è chiaramente molto significativo quando si ha a che fare con l'analisi di dataset grandi, come accade nel machine learning. Le librerie usate sono state PyCaret e Sklearn, dove la prima ha un livello di astrazione maggiore della seconda, per cui permette di svolgere le stesse funzioni in molte meno righe di codice. Dunque, l'utilità di Sklearn è stata quella di vedere ad un livello un pò più basso se le azioni effettuate dal modello fossero coerenti con l'output ottenuto, per avere maggiore sicurezza riguardo ai risultati ottenuti. Il lavoro si è svolto in tre fasi principali: una parte iniziale in cui sono stati cercati i metodi più appropriati per approcciare il problema, ed i più adatti in relazione alle esigenze del DataSet che è stato trattato. Successivamente, trovato il modello, si è potuti procedere con lo studio di vari casi: la classificazione, nello specifico adottando il modello dell'extraTreesRegressor, ha risposto in maniera positiva, con un'accuracy che si aggira intorno al 90%. Poi, è stata effettuata una riduzione di dimensionalità e un analisi delle principali caratteristiche che hanno influito nella creazione del modello e nelle predizioni. Dunque, inizialmente l'approccio è stato quello di cercare gli strumenti e lasciare agire loro, poi, una volta constatato che rispondevano nel modo corretto, le azioni successive sono state atte a comprendere come gli strumenti avevano agito nello specifico. Con l'analisi delle feature importance, si è potuto constatare come il modello si basi all'incirca sulle stesse feature per la predizione, sia nel caso dove al DataSet era stata effettuata la riduzione della dimensionalità, sia nel caso senza. Quest'ultima analisi è stata effettuata raccogliendo i valori delle varie onde celebrali a seconda dell'elettrodo usato per misurarle.

2 Introduzione e motivazioni

Sebbene l'esperienza emotiva umana giochi una parte centrale nella nostra vita, la nostra conoscenza scientifica sulle emozioni umane è ancora molto limitata. La causa principale è da ricercarsi nell'enorme complessità del cervello umano, la quale rende impossibili determinati approcci di studio, quali potrebbero essere approcci a forza bruta, o comunque studi che richiedano un numero limitato di tempo. Il problema nel suo insieme potrebbe apparentemente sembrare irrisolvibile. Un'idea potrebbe allora essere quella di utilizzare approcci d'intelligenza artificiale. Nel momento in cui si riscontra che un qualche problema è troppo complesso per essere affrontato con un preciso algoritmo di risoluzione, può venire in mente di lasciar trovare alla macchina determinate corrispondenze e legami, apparentemente non visibili da un essere umano. Il machine learning è una grande opportunità ma deve essere controllato. È conveniente utilizzare l'apprendimento automatico quando:

- È difficile formalizzare il problema
- Abbiamo dati incerti, rumorosi o incompleti, che ostacolano la formalizzazione di soluzioni
- È difficile scalare utilizzando altri metodi; infatti, le soluzioni di Machine Learning sono efficaci quando si tratta di gestire problemi su vasta scala.

Ma l'introduzione di metodi di apprendimento automatico porta con sé diversi altri problemi da risolvere e ricerche da effettuare. Per ottenere maggiori conoscenze in questo ambito, attraverso il riconoscimento automatico, è necessario che i ricercatori abbiano ricchi insiemi di dati o di esperimenti ripetibili. Tuttavia, ottenere dati di sensori multimodali è una sfida, in quanto diverse modalità di misurazione richiedono diverse apparecchiature, sviluppate e prodotte da diverse aziende all'interno delle quali vengono sfruttate competenze diverse. Il lavoro svolto in questa tesi si basa su un dataset chiamato MAHNOB-HCI . Dunque, MAHNOB-HCI nasce per contribuire a questa necessità di database emotivi ed etichettatura affettiva, i quali, vista l'interdisciplinarità richiesta per mettere insieme tutti i pezzi, scarseggiano. MAHNOB-HCI è un database multimodale. I valori numerici che lo compongono sono stati registrati in risposta a stimoli affettivi, con l'obiettivo di riconoscere le emozioni. È stata predisposta una configurazione multimodale per la registrazione sincronizzata di video di volti, segnali audio, dati sullo sguardo e segnali fisiologici del sistema nervoso periferico/centrale. Ventisette partecipanti di entrambi i sessi e di diverse culture hanno partecipato a due diversi esperimenti. Nel primo esperimento, i partecipanti hanno guardato 20 video, e durante la visione hanno dovuto esprimere il proprio stato emozionale. La scelta si componeva di: eccitazione, valenza, predominio e prevedibilità, e altre parole chiave riguardanti emozioni. Nel secondo esperimento, brevi video e immagini sono state mostrate una volta senza alcun tag e poi con tag corretti/ non corretti. I partecipanti hanno dovuto esprimere accordo o disaccordo con il tag relativo a ciò che veniva mostrato. I video registrati e le risposte corporee sono stati segmentati e archiviati in un database. [1] Il database è poi stato messo a disposizione della comunità accademica tramite un sistema

web-based. Sebbene il modo più semplice per rappresentare l'emozione consista nell'usare etichette discrete come "paura" o "gioia", le rappresentazioni basate sull'etichetta presentano alcuni svantaggi. Nello specifico, le etichette non sono interlinguistiche: le emozioni non hanno traduzioni esatte in diverse lingue, ad es. "disgusto" non ha una traduzione esatta in polacco. Gli psicologi, quindi, rappresentano spesso emozioni o sentimenti in un spazio n-dimensionale (generalmente 2 o 3D). Il più famoso fra tali spazi, utilizzato per il database MAHNOB-HCI, è la valenza 3D pleasure-arousal-dominance (PAD). La scala di valenza (indicata con valence o pleasure) varia da "sgradevole" a

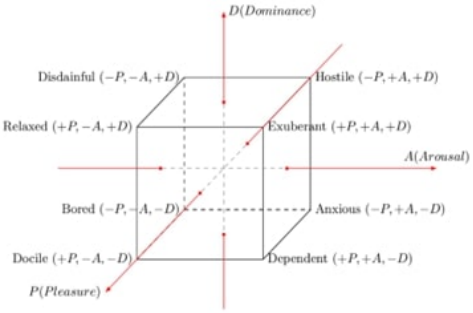


Figura 1: Modello PAD di Russel

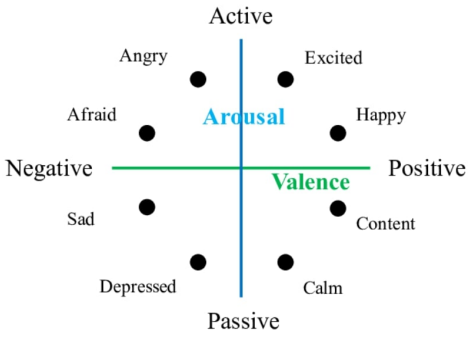


Figura 2: Piano 2-dimensionale

"piacevole". La scala dell'eccitazione (arousal) varia da passiva ad attiva o emozionato. La scala di dominanza (dominance) varia da "senza controllo" a "sotto controllo". [2] Durante l'esperimento, è stata usata come scala anche la predicibilità dell'evento a cui si assisteva. In questa tesi, in un primo momento verranno utilizzate tecniche di regressione, per risalire dai valori delle onde celebrali dei partecipanti agli esperimenti, alle emozioni provate sempre da quest'ultimi, dopodiché verranno utilizzate tecniche di classificazione. Dunque, l'idea iniziale dalla quale partiamo è: abbiamo le onde celebrali dei soggetti, relative ad una certa azione o contenuto visionato, ed il relativo tag dato dal soggetto, sempre relativo a quel contenuto. La prima cosa che cerchiamo di fare è risalire ad un qualche pattern, che leghi l'etichetta alle onde. Se, poi, associamo i valori delle onde celebrali misurate ad una qualche emozione, è come se avessimo delle corrispondenze: azione- emozione provata. Successivamente, si sposta il problema sulla classificazione, ovvero classificare una certa etichetta a seconda di come sono già state classificate etichette precedenti. Verrà poi effettuata una riduzione delle dimensionalità, per eliminare eventuali ridondanze e condizionamenti fra misurazioni di elettrodi che potrebbero interferire a vicenda. Infine, verranno valutate quali sono le feature più importanti nell'addestramento del modello, ovvero quali fra le varie colonne contenenti i valori registrati influisce maggiormente sulle predizioni finali del modello. Con approcci di machine learning come quelli utilizzati nel seguito, oltre al problema del reperimento dei dataset da utilizzare, successivamente sovviene anche la questione della potenza di calcolo necessario alla processazione dei dati. In questo aspetto, la soluzione è stata fornita dai servizi offerti da Google Colab. Mentre per i primi test può essere sufficiente affidarsi alla propria macchina, con l'aumentare della dimensione dei dataset l'esecuzio-

ne di algoritmi di addestramento complessi diventa rapidamente proibitiva, ed il dataset utilizzato contiene oltre 18 000 record, ed ogni record contiene valori in un centinaio di colonne. Per risolvere questo problema, esistono numerosi servizi cloud che offrono potenza di calcolo, spesso a pagamento o con varie limitazioni. In alternativa ad essi, esiste poi Google Colab, una piattaforma che, seppur con alcune limitazioni, ci permette di eseguire codice direttamente sul Cloud, sfruttando la potenza di calcolo fornita da Google. Per eseguire codice, Google Colab sfrutta i cosiddetti Jupyter Notebook. Questi non sono altro che documenti interattivi nei quali possiamo scrivere (e quindi eseguire) il nostro codice. Più precisamente, tali documenti permettono di suddividere il nostro codice in celle. Questo servizio di Google offre tre tipi di runtime per i notebook: CPU, GPU e TPU con un tempo di esecuzione continuo pari a 12 ore. Dopo questo tempo, la macchina virtuale viene "resettata" e bisogna ricominciare da capo. Anche ogni volta che viene chiuso il browser, la sessione viene resettata. Ci sono inoltre diverse librerie di Python preinstallate, come per esempio Pandas, NumPy, scikit-learn e, se necessario, è possibile installarne di altre. L'uso di librerie low-code facilita e snellisce di molto il lavoro, evitando di scrivere anche numerose righe di codice, dunque per accelerare il lavoro di analisi dei dataset. Nel sistema implementato in questa tesi è stato fatto uso della libreria PyCaret inizialmente, e successivamente di scikit-learn, PyCaret è una libreria Python progettata per consentire agli utenti di eseguire complesse attività di Machine Learning con poche righe di codice. Invece scikit-learn si trova ad un livello un po' più basso, anche se rimane comunque una libreria con la quale è possibile effettuare operazioni molto complesse con poche righe di codice, ma permette di espletare in più chiamate di funzioni e dichiarazioni quello che in PyCaret è fatto automaticamente con una sola chiamata. [3]

3 Related Works

All'interno del campo di studio delle emozioni umane sono presenti molti lavori, sia facenti uso del Machine Learning, sia di metodi più classici. Si può ragionevolmente considerare l'apprendimento automatico come più performante rispetto ad altri metodi, per quanto concerne i nostri scopi, in quanto è facilmente generalizzabile e scalabile, e non richiede soglie dettate da un esperto del dominio applicativo ma, al contrario, una volta individuato il modello, in ogni caso di studio o applicazione, non servirà un nuovo esperto e un nuovo studio. Per esempio, in "Predicting Students' Emotions Using Machine Learning Techniques" [12], si utilizza il Machine Learning, nello specifico la classificazione tramite i modelli Naive Bayes, Support Vector Machines e Random Forest, per indagare sulle emozioni provate da alcuni studenti.I dati dell'esperimento sono stati raccolti da lezioni tenute in alcune università giordane. Gli studenti hanno inviato feedback, opinioni e sentimenti sulla lezione tramite Twitter. Per ogni tweet, è stato chiesto loro di scegliere un'emozione da un insieme di emozioni fornite (etichette). Successivamente, è stato analizzato quali fossero le emozioni maggiormente provate, ed è stato constatato che alcune fossero più facilmente rilevabili di altre, pur non avendo ottenuto punteggi alti nell'accuracy.

Un altro lavoro che si basa sulla predizione di emozioni tratte dal testo scritto è "Emotions from text: machine learning for text-based emotion prediction". Come negli altri articoli citati, anche in questo si fa uso della classificazione, e lo scopo che si prefiggono i ricercatori è quello di rendere la sintesi vocale il più naturale e coinvolgente possibile, capendo quale emozione descrive più appropriatamente un determinato passaggio di testo. Per il loro scopo è sufficiente considerare valence, prelevata da un data set in cui erano annotate frasi e relative etichette corrispondenti allo stato emozionale. Alla fine, è mostrato che l'accuracy è pari a 0.69. [15]

Si può citare anche "Supervised machine learning for audio emotion recognition" [13], nel quale si indaga sul riconoscimento di emozioni relative a risposte a stimoli auditivi. Così come il lavoro di questa tesi si basa su un Data Set contenente risposte a stimoli fisici, anche nel lavoro sopracitato i ricercatori si sono basati sul Data Set denominato "IADS", anch'esso basato sull'etichettatura di "arousal, valence e dominance". Anche in questo caso si esclude, inizialmente, un approccio tramite regressione, in quanto forniscono prestazioni peggiori (con R^2 pari a 0.28) di altri approcci tentati: nel caso in questione, il modello adottato si rivela basato su reti neurali. Tuttavia, pur ottenendo score positivi, viene riportato che la computazione è stata molto pesante, ed ha richiesto un totale di 16 ore per essere portata a termine.

Invece "Emotion Detection and Sentiment Analysis of Images" [14] analizza previsioni tramite apprendimento automatico di stimoli in risposta alla visione di immagini. Il lavoro esordisce affermando che al giorno d'oggi, le persone condividono molti contenuti sui social media in forma di immagini, dunque analizzare tali contenuti può fornire informazioni sul sentimento generale delle persone riguardo alle elezioni presidenziali, rimarcando quindi i fini sociologici della ricerca. Anche questo lavoro utilizza metodi di classificazione, nello specifico Support Vector Machine, ma stavolta la scala emozionale non è quella PAD, ma classifica le emozioni in Amore, Felicità, Violenza, Paura e tri-

stezza. I risultati finali di accuracy sono abbastanza elevati, ma molto più interessante è ciò che il modello ha imparato delle varie emozioni: per esempio associa "amore" a tramonti, cuori, fiori, il colore rosso, così come associa "Felicità" a colori accesi, volti, a "paura" immagini scure, foto di insetti, e così via.

4 Design e implementazione

4.1 Implementazione della regressione

Nel prosieguo, verrà utilizzata la regressione sia tramite PyCaret, sia tramite Sklearn. Le prime azioni sono comuni ad entrambe le regressioni: inizialmente, è stato installato PyCaret/Sklearn sulla piattaforma online Google Colab ed è stato importato il dataset, tramite una funzione della libreria Pandas, in un dataframe. Successivamente, sono stati preparati i dati eliminando diverse colonne dal data set, le quali non erano funzionali agli scopi da perseguire per l'individuazione dei pattern cercati. Le azioni successive sono state le seguenti:

- PyCaret: dal dataset sono stati ricavati due dataframe distinti: uno contenente i dati tramite i quali eseguire il fitting e l'altro contenente dei dati nuovi su cui effettuare le previsioni. La percentuale di dati da testare che è stata scelta è stata il 20% del dataset originario, mentre la restante parte è stata utilizzata per "allenare" il sistema. Poi, è stato configurato l'ambiente di lavoro, tramite la funzione setup(). Questa funzione deve essere chiamata prima di chiamare tutte le altre che seguiranno e ha un solo parametro obbligatorio, ovvero il dataframe. In questo caso viene passato quindi come argomento il dataframe che contiene i dati per il training del sistema, e viene scelto il label sul quale effettuare la regressione. Poi, sono stati confrontati vari modelli ed è stato scelto quello con le prestazioni migliori. Tramite la funzione create_model(), è stato creato il modello. Poi, il modello viene allenato e si fa una prima previsione delle prestazioni. Successivamente, viene ripetuta la previsione, ma stavolta sui dati che il modello non ha ancora mai visto (cioè il test set, quel 20% iniziale di dati portati fuori) per verificare che abbia acquisito la capacità di generalizzare correttamente le previsioni.

- Sklearn: dal dataset si ricavano due diverse porzioni di dati, una che corrisponde ai dati su cui allenare il sistema, l'altra che invece contiene i valori che il modello dovrà imparare a predire. Il passo successivo consiste nel dividere il dataset in training e test set, stavolta utilizzando il metodo della libreria "sklearn.model_selection" denominato "train_test_split", ancora usando il 20% come valore di riferimento per quanto concerne l'hold out. Successivamente, è stato creato il modello ed è stato allenato sui dati di training. Per fare ciò, prima è stata chiamata la funzione per creare il modello, poi è stato chiamato un metodo dell'istanza creata, detto "fit", al quale si passano i due array contenenti i dati di training relativi alle variabili dipendenti ed a quelle indipendenti. Infine, chiamando il metodo "predict", al quale stavolta si passano i dati di test, è stata valutata la accuratezza del modello, tramite diverse funzioni della libreria "sklearn.metrics".

4.2 La regressione

La regressione è un processo statistico, il quale permette di stimare le relazioni tra variabili. Determina il legame funzionale tra le variabili indipendenti e le variabili dipendenti. Se consideriamo la regressione lineare, si ha un'unica variabile dipendente (la y) ed un'unica indipendente (la x), più due parametri (α, β) detti parametri liberi, da apprendere. E' uno dei due principali problemi dell'apprendimento supervisionato, la branca del machine learning che vuole insegnare alla macchina a risolvere un determinato problema mostrandogli come questo problema sia stato già risolto in passato. La regressione corrisponde al valore medio atteso in base ai dati di input immessi nel modello. Grazie alla regressione si può parlare di previsione sui valori attesi: intuitivamente, consente di trovare la relazione tra input e output, in modo da predire output futuri avendo a disposizione solamente l'input. I valori di ingresso sono un vettore di dati n-dimensionale dalla quale deriva un unico valore di uscita; nel caso di regressione lineare, si ha n=1. [4] Ad esempio, avendo a disposizione un set di dati contenente le specifiche di diversi appartamenti e il relativo valore di mercato, potremmo utilizzare la regressione per predire a che prezzo potrà essere venduto un nuovo appartamento basandoci sulle sue specifiche. Matematicamente, la regressione può essere intesa come il trovare la funzione che meglio approssima la relazione tra la variabile dipendente X (l'input) e la variabile indipendente Y (l'output). Nel caso di una regressione lineare questa funzione è un polinomio:

$$f(x) = \alpha + \beta x$$

Graficamente, invece, questa relazione lineare può essere rappresentata come una retta che passa il più vicino possibile a tutti i punti costituiti da input X e output Y. Gli algoritmi supervisionati fanno uso di un insieme di campioni, ed a ognuno di essi è associato il relativo output corretto. Davanti a questo tipo di situazioni, in cui è presente un valore con la relativa etichetta (output corretto) abbiamo un problema supervisionato. Tra i problemi supervisionati abbiamo due categorie principali: una è la regressione, l'altra è la classificazione. La regressione consiste nello stimare un valore reale (continuo), mentre la classificazione genera un valore discreto. Dato un training set TS contenente n pattern:

$$x_1, y_1, ..., x_n, y_n$$

e la seguente dipendenza tra variabile indipendente e dipendente:

$$y_i = \alpha + \beta x_i + \epsilon_i$$

dove:

- ϵ_i l'errore di misura (incognito) del pattern x_i
- α, β parametri da determinare

La soluzione ai minimi quadrati (Least Square) del problema consiste nel determinare l'equazione della retta: f(x) = α + β x che minimizza la somma dei quadrati dei residui:

$$\alpha^*, \beta^* = \arg\min_{\alpha,\beta} \sum_{i=1}^{n} (f(x) - y_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$$

1) Regressione lineare - quando la relazione tra le variabili indipendenti è di tipo lineare, abbiamo una regressione lineare.

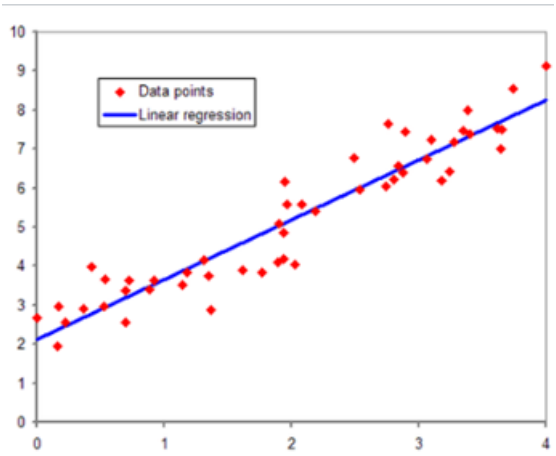


Figura 3: Esempio di regressione lineare con una variabile dipendente ed una indipendente: i punti rossi sono i valori del dataset, la retta blu è la retta di regressione che prova ad approssimarli ad una funzione.

2) Regressione polinomiale - la funzione che collega le variabili indipendenti dalle variabili dipendenti è un polinomio.

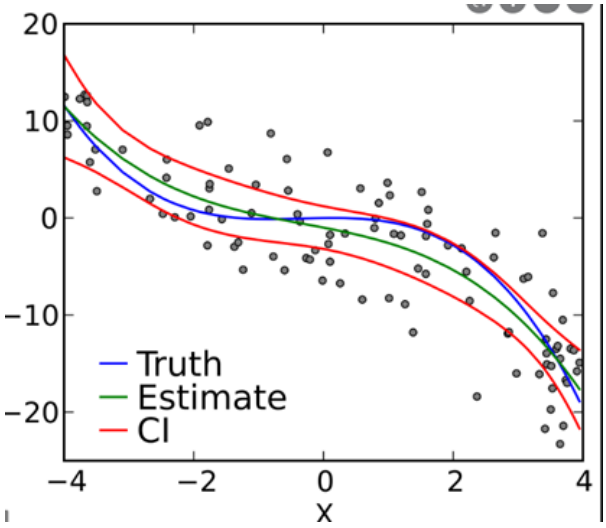


Figura 4: Esempio di regressione polinomiale

4.3 Decision tree

Nel machine learning un albero di decisione è un modello predittivo, in cui:

- ogni nodo interno rappresenta un test su un attributo
- ogni ramo verso un nodo figlio rappresenta un possibile valore per quell'attributo
- una foglia il valore predetto per la variabile obiettivo a partire dai valori delle altre proprietà, che nell'albero è rappresentato dal cammino (path) dal nodo radice (root) al nodo foglia.

E' quindi un sistema con n variabili in input e m variabili in output. Le variabili in input (attributi) sono derivate dall'osservazione dell'ambiente. Le ultime variabili in output, invece, identificano la decisione / azione da intraprendere. Ogni nodo verifica una condizione su una particolare proprietà dell'ambiente e ha due o più diramazioni verso il basso. Il processo consiste in una sequenza di test. Comincia sempre dal nodo radice, il nodo genitore situato più in alto nella struttura, e procede verso il basso. A seconda dei valori rilevati in ciascun nodo, il flusso prende una direzione oppure un'altra e procede progressivamente verso il basso. [5]

In un albero decisionale le variabili possono essere:

- Variabili discrete: Le variabili hanno valori numerici interi. In questi casi si parla di classificazione. La forma discreta più semplice è la classificazione booleana dove le variabili hanno soltanto due valori: zero (falso) o uno (vero).
- Variabili continue: Le variabili hanno valori numerici reali. Dati due numeri reali qualsiasi, c'è sempre un altro numero intermedio tra questi. In questi casi si parla di regressione.

Normalmente un albero di decisione viene costruito utilizzando tecniche di apprendimento a partire dall'insieme dei dati iniziali (data set), il quale può essere diviso in due sottoinsiemi: il training set sulla base del quale si crea la struttura dell'albero e il test set che viene utilizzato per testare l'accuratezza del modello predittivo così creato.

Pro e contro

- I vantaggi: Gli alberi logici hanno l'indiscusso vantaggio della semplicità. Sono facili da capire e da eseguire. Diversamente da altri modelli, l'albero decisionale è facilmente comprensibile. Pertanto, l'uomo può verificare come la macchina giunga alla decisione, in modo da constatare se il risultato sia coerente con quanto ci si debba aspettare. Dunque, essendo facilmente interpretabili, risulta più semplice anche trattarli attraverso codice di programmazione.
- Lo svantaggio principale è l'alto rischio di overfitting, visto che l'albero procede in una direzione o in un'altra a seconda del valore dell'attributo che sceglie volta per volta. Dunque rischia di escludere un sottoalbero che invece sarebbe potuto essere corretto, così come rischia di imparare sequenze precise che non generalizzano veramente i pattern che dovrebbe apprendere.

Vediamo un esempio:

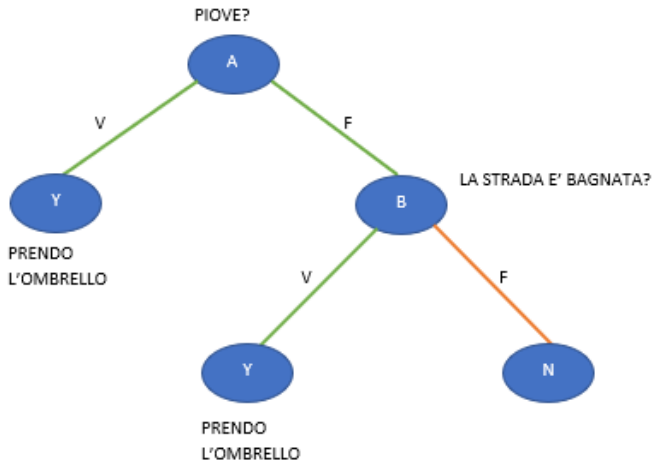


Figura 5: Albero di decisione

A	B	Y
V	V	1
V	F	1
F	V	1
F	F	0

Figura 6: Relativa tabella di verità

Potremmo anche rappresentare l'albero nel linguaggio proposizionale, ottenendo la seguente funzione di verità:

$$\forall s PrendoOmbrello(s) \Leftrightarrow (Piove(s)) \vee (\neg Piove(s) \wedge StradaBagnata(s))$$

Nel proseguire della tesi, il modello utilizzato inizialmente per la regressione sarà quello del decision tree.

4.4 Apprendimento supervisionato e non supervisionato

L'apprendimento supervisionato (supervised learning) è una tecnica di apprendimento automatico che mira a istruire un sistema informatico, in modo da consentirgli di elaborare automaticamente previsioni sui valori di uscita di un sistema rispetto ad un input sulla base di una serie di esempi ideali, costituiti da coppie di input e di output, che gli vengono inizialmente forniti. Quindi vengono dati dal supervisore esempi di training nella forma di coppie < input, output > = < x, y > (esempi etichettati) per la creazione di una funzione sconosciuta f(x). L'input è quindi una matrice con esempi etichettati, a partire dai quali la macchina elabora la funzione ipotesi h(x). Definiamo il valore target come il valore "y" che vogliamo ottenere come output (e che ci viene dato tramite gli esempi). Dunque la macchina deve trovare una funzione f(x) incognita che collega le variabili di input x a una variabile di output y. Il suo scopo è stimare una funzione ipotesi h(x) in grado di approssimare la f(x). Per capire se la funzione ipotesi è corretta, la macchina deve valutare se la funzione ipotesi h(x) approssima o meno la funzione f(x) e, nel caso, con quale grado di accuratezza. Per capirlo utilizza un altro insieme di dati, detto test set, fornito sempre dal supervisore. Pur essendo, l'insieme di training e di test, entrambi forniti da supervisiori, i due insiemi sono composti da esempi diversi,

altrimenti il modello non imparerebbe davvero, ma si limiterebbe ad "imparare a memoria" i dati forniti. [6] A questo punto, la macchina risponde agli N_T esempi del test set. Poi confronta ogni sua risposta R con la risposta corretta indicata dalle Y nel test set.

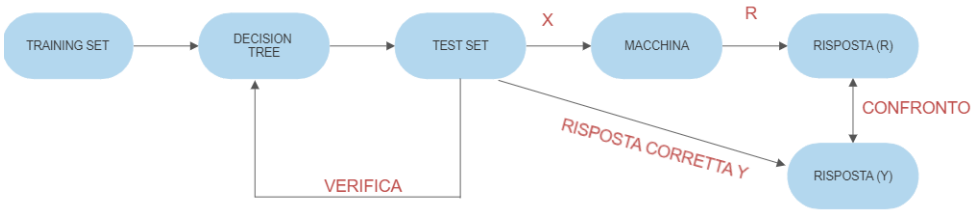


Figura 7: Schema funzionale apprendimento supervisionato

Le risposte coincidenti, ovvero tali per cui si verifichi R=Y, incrementano il numero delle risposte corrette R_c della macchina. Se la percentuale di risposte corrette C della macchina è soddisfacente, la funzione ipotesi h(x) supera l'esame e viene accolta.

$$C = \frac{R_c}{N_T} = \frac{Risposte\ corrette}{Numero\ test}$$

In caso contrario l'apprendimento supervisionato riparte con l'analisi di un ulteriore training set, facendo ricominciare il ciclo da capo. Le principali categorie dell'apprendimento supervisionato sono:

- La classificazione: La macchina viene addestrata alla classificazione dal supervisore tramite l'aggiunta di etichette sui dati in cui giudica il risultato. Ogni etichetta è una classe discreta che identifica il risultato atteso (prendendo come esempio le email: spam o no spam) oppure un giudizio di valore.
- La regressione: Nella regressione l'output è un valore continuo. Alla macchina spetta il compito di trovare una relazione tra i valori.

Una volta fatta una panoramica sull'apprendimento supervisionato, vediamo il suo speculare: l'apprendimento senza supervisione è una tecnica di machine learning in cui la macchina apprende dall'esperienza senza avere esempi e risposte di riferimento. Dunque, mentre nell'apprendimento supervisionato la macchina apprende dagli esempi e le soluzioni sono fornite dal supervisore, in quello non supervisionato i dati non sono etichettati e la struttura stessa dei dati non è predefinita. Per imparare, la macchina deve estrarre le informazioni rilevanti dai dati disponibili. Fra le principali tecniche di machine learning senza supervisione citiamo il clustering, in cui l'algoritmo ricerca delle regolatirà nei dati che ha a disposizione. [7]

4.5 Validazione del modello

Abbiamo inizialmente detto che, seppur da un lato gli algoritmi di apprendimento automatico abbiano le potenzialità per migliorare i processi computazionali, affinare le previsioni dei risultati e contenere i tempi di risoluzione, dall'altro lato presentano potenziali

criticità da tenere in considerazione nel processo di validazione del modello previsionale. Di seguito sono riportate alcune potenziali criticità comunemente associate a queste tecniche di apprendimento:

- Overfitting (eccessivo adattamento): si verifica quando un modello basa le sue previsioni su correlazioni fittizie all'interno di un campione di dati anziché su relazioni autentiche esistenti all'interno della popolazione nel suo complesso. Ovvero,l'algoritmo può (erroneamente) utilizzare ogni relazione, lineare e non, tra le variabili nei dati del campione di training, per eseguire raggruppamenti e/o fare previsioni. Il maggior rischio è che ogni singolo campione, su cui viene costruito il modello, abbia le proprie peculiarità che non riflettano puntualmente la vera popolazione, riducendo il potere di replicabilità del fenomeno osservato nella totalità dei dati. Gli alberi di Classificazione e di Regressione sono particolarmente sensibili a questa tipologia di criticità, in quanto possono suddividere i dati fino a raggiungere la classificazione perfetta o quasi perfetta, generando così alcune partizioni fuorvianti. Se un modello caratterizzato da overfitting viene applicato a nuovi dati appartenenti alla stessa popolazione, può, potenzialmente, produrre previsioni poco accurate e questa mancata accuratezza può avere effetti molto distorsivi. Problema speculare dell'overfitting è l'underfitting, il quale si verifica quando il modello non è riuscito ad apprendere correttamente le relazioni fra i dati. [8] Graficamente, vediamo un possibile esempio:

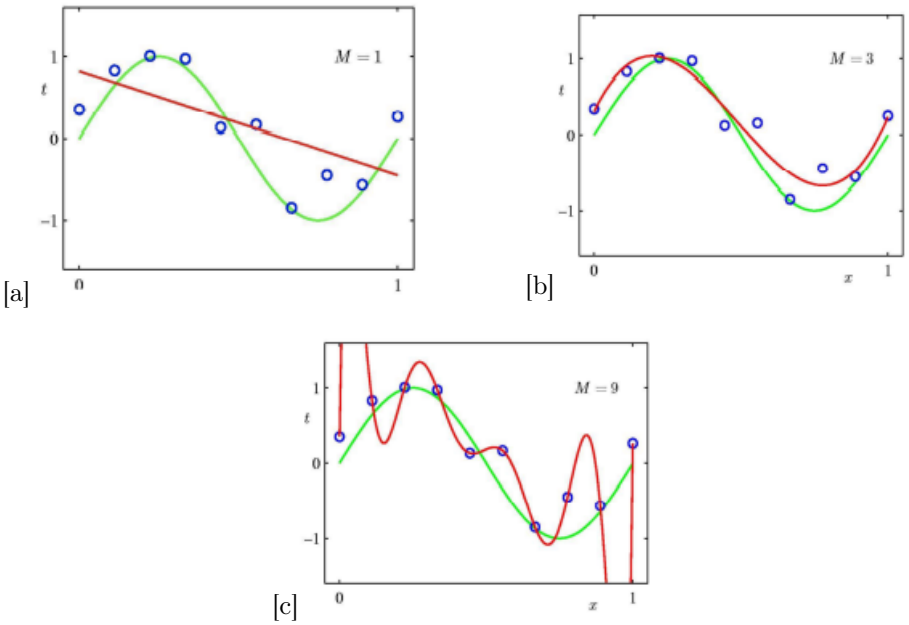


Figura 8: In [a] vediamo un esempio di underfitting: il polinomio è di grado 1, troppo basso per riuscire ad approssimare correttamente la funzione; in [b] vediamo un modello che ha appreso correttamente; in [c] vediamo un esempio di overfitting: il polinomio, di grado 9, passa esattamente per tutti i punti di test, ma un nuovo punto immesso finirebbe con molta probabilità fuori dalle linee rosse

- Riduzione della trasparenza: Le tecniche di Machine Learning riducono, inoltre, la trasparenza del processo interpretativo del modello previsionale. Con le tecniche di regressione più tradizionali, è più intuitivo vedere come interagiscono le variabili all'interno del modello. La maggior parte delle tecniche di apprendimento automatico non produce tuttavia risultati così facilmente interpretabili, ma anzi funziona come una specie di "scatola nera" diminuendo la trasparenza e l'interpretabilità della metodologia sottostante. Sebbene sia possibile esaminare i vari steps intermedi generati dall'algoritmo per capire come sia in grado di generare le previsioni, spesso è necessario ed opportuno possedere una conoscenza molto approfondita e consolidata di queste tecniche per poter valutare con trasparenza e giudizio critico l'intero processo. Altrimenti, gli utenti potrebbero ignorare la presenza di distorsioni latenti ed errori, i quali comporterebbero le possibilità di aumentare il potere predittivo. Il modello potrebbe essere stato adattato su dati non adeguati, scarsamente specificati o utilizzati in un contesto non corretto. Con le tecniche di Machine Learning, tuttavia, la mancanza di trasparenza nelle previsioni rende più difficoltosa l'individuazione di questi tipi di criticità.

- Altri problemi, per esempio risultati che si basano su dati campionari non adeguati: la qualità dei risultati ottenuti da un modello è direttamente proporzionale a quella dei dati su cui viene costruito. Tecniche di campionamento incomplete, o poco idonee al tipo di proiezione che si intende produrre, possono fornire dati non rappresentativi della popolazione nella sua interezza.

Per porre rimedio alle problematiche elencate, si utilizzano tecniche di validazione dei modelli. La Cross Validation è una tecnica frequentemente utilizzata per assicurarsi che un modello non produca un eccessivo livello di overfitting rispetto ai dati campionari sui quali è sviluppato. Con tale tecnica, un modello viene costruito ed adattato utilizzando solo una parte del campione di dati a disposizione, utilizzando la restante porzione per valutarne l'effettivo potere predittivo. In condizioni ideali, il modello funzionerà ugualmente bene su entrambe le porzioni di dati. In caso contrario, è probabile la presenza di un livello di overfitting non trascurabile. I dati campionari vengono solitamente suddivisi in proporzione, l'80% viene utilizzato per adattare o "addestrare" il modello (training set) e il restante 20% (validation set) viene impiegato per valutarne la capacità predittiva. Esistono anche approcci più rigorosi alla Cross Validation, tra cui la k-fold Validation, che prevede che il processo sopra esposto venga ripetuto k volte con diverse suddivisioni dei dati campionari. In pratica, si può addestrare ed allenare il modello con differenti tipi di dati campionari, e valutare sequenzialmente le previsioni prodotte al fine di determinare se il modello è eccessivamente sensibile ai dati utilizzati per la fase di training.

Viene ripetuta la stessa procedura per k volte, selezionando ogni volta un sottoinsieme diverso come validation set. A questo punto, calcolo la media dei valori predittivi ottenuti nei k esperimenti. L'albero selezionato è quello che risente meno del rumore causato dagli attributi irrilevanti presenti nel dataset. [9]

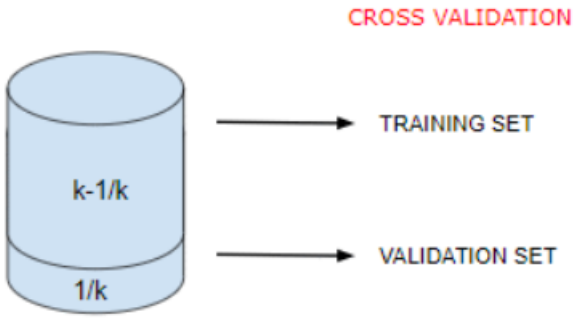


Figura 9: Cross validation

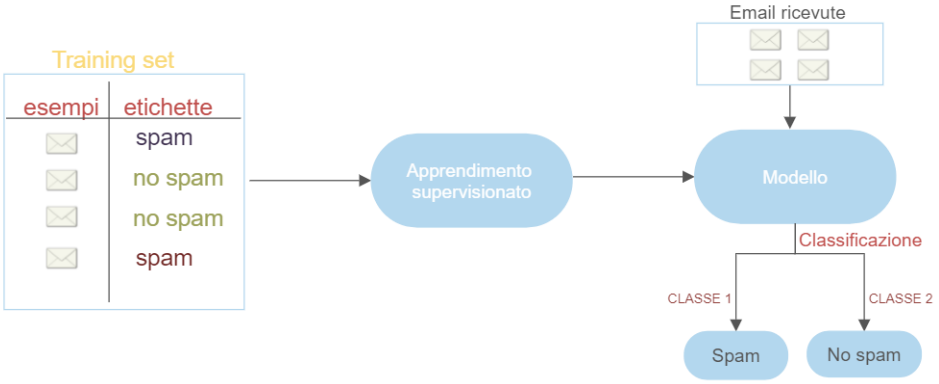
4.6 Implementazione della classificazione

All'inizio del capitolo sono state presentate le scelte relative all'implementazione della regressione, seguite da un excursus concernente la teoria dietro a ciò che è stato implementato col codice. Adesso, analogamente, verrà fatto un quadro generale riguardo a come è stata implementata la classificazione. Le librerie e i metodi utilizzate sono state, ancora una volta, quelle di Sklearn: inizialmente, è stato installato Sklearn sulla piattaforma online Google Colab ed è stato importato il dataset, tramite una funzione della libreria Pandas, in un dataframe. Successivamente, sono stati preparati i dati eliminando diverse colonne dal data set, le quali non erano funzionali agli scopi da perseguire per l'individuazione dei pattern cercati. Le azioni successive sono state le seguenti dal dataset si ricavano due diverse porzioni di dati, una che corrisponde ai dati su cui allenare il sistema, l'altra che invece contiene i valori che il modello dovrà imparare a predire. Stavolta è necessario chiamare il metodo "round(0)" sulle etichette, in quanto la classificazione prevede etichette intere. Il passo successivo consiste nel dividere il dataset in training e test set, stavolta utilizzando il metodo della libreria "sklearn.model_selection" denominato "train_test_split", ancora usando il 20% come valore di riferimento per quanto concerne l'hold out. Successivamente, è stato creato il modello ed è stato allenato sui dati di training. Per fare ciò, prima è stata chiamata la funzione per creare il modello, poi è stato chiamato un metodo dell'istanza creata, detto "fit", al quale si passano i due array contenenti i dati di training relativi alle variabili dipendenti ed a quelle indipendenti. Infine, tramite il metodo della libreria "sklearn.metrics" denominato "score", al quale sono stati passati i due array corrispondenti a input e output di test, è stata stampata a video l'accuratezza del modello.

4.7 La classificazione

La classificazione è un problema tipico del machine learning. Consiste nell'assegnare un dato a una categoria sulla base di un modello di classificazione appreso dalla macchina tramite l'intelligenza artificiale. Ad esempio, un algoritmo decide se un messaggio email in arrivo è spam oppure no. In questo caso le classi sono due: spam e no-spam. La decisione non dipende da un modello di classificazione deciso dal programmatore bensì

dall'esperienza della macchina. Un problema di classificazione può essere risolto tramite diverse tecniche e paradigmi di machine learning: nel caso Supervisionato, la macchina riceve in input degli esempi già classificati con alcuni messaggi spam e no spam (training set). [10] Sulla base di questi esempi la macchina costruisce un modello decisionale per classificare le future email.



Nel caso non supervisionato, la macchina non ha esempi già classificati. La decisione si basa sulla vicinanza/distanza dei dati. Se i dati di un'email sono simili a quelli di un'altra email spam, allora anche la prima è probabilmente spam, e viceversa. Una tecnica di questo tipo è il clustering. Nel caso di 2 sole classi si usa il termine binary classification, con più di due classi multi-class classification.

4.8 Riduzione della dimensionalità

La riduzione della dimensionalità è una tecnica di mapping dei dati. E' una delle operazioni di pre-elaborazione del dataset nell'apprendimento automatico non supervisionato. E' utile nel machine learning per eliminare dal dataset le informazioni ridondanti (correlate), meno o poco rilevanti per il problema da risolvere. E' senza dubbio più semplice e meno dispendioso addestrare un algoritmo con uno spazio dati di dimensione inferiore, quindi è una soluzione al problema della curse of dimensionality. La curse of dimensionality è un problema della dispersione dei pattern in un grande volume di dati. Nel machine learning si lavora con una grande mole di dati, a causa dell'elevata dimensionalità, i pattern sono dispersi nel dataset come in un oceano di rumore e dati insignificanti. In queste circostanze diventa complesso trovare uno schema. L'algoritmo impiega più tempo (complessità temporale) e più memoria (complessità spaziale). La riduzione della dimensionalità riduce il volume dei dati in cui cercare, senza perdere le informazioni più rilevanti. Ridurre la dimensionalità dei dati non vuol dire soltanto eliminare alcune dimensioni (rumore) ma soprattutto combinare tra loro le informazioni ridondanti e correlate. Gli obiettivi dell'algoritmo sono i seguenti: Eliminare il rumore dei dati, combinare le informazioni correlate. Un dataset di una dimensione iniziale R_n viene ridotto a uno spazio di dimensione inferiore R_k dove k<n. [11] Le principali tecniche di riduzione della dimensionalità dei dati sono le seguenti:

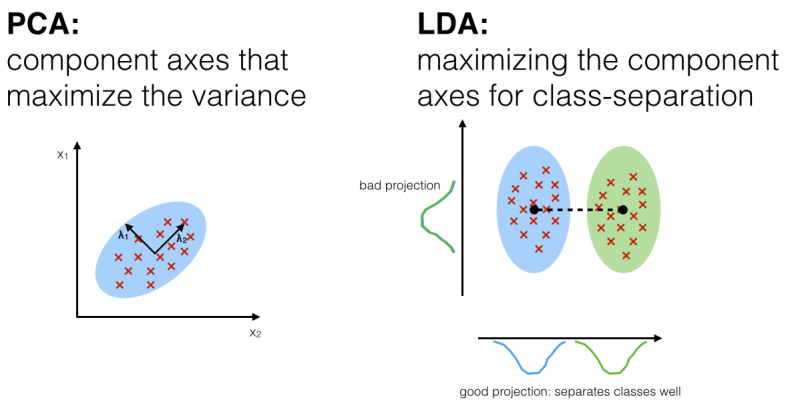


Figura 10: https://towardsdatascience.com/dimensionality-reduction-for-data-visualization-pca-vs-tsne-vs-umap-be4aa7b1cb29

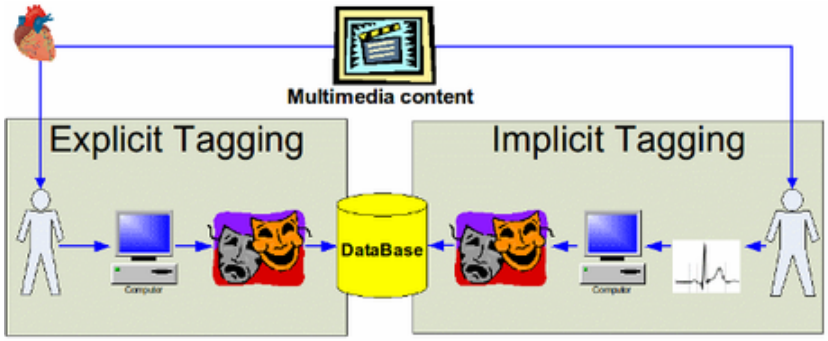
- Principal Component Analysis (PCA) La tecnica consiste in un mapping lineare dei dati non supervisionato. L'obiettivo della tecnica è individuare le dimensioni che rappresentano meglio i pattern.
- Linear Discriminant Anaylis (LDA) La tecnica LDA è un mapping lineare dei dati supervisionato. L'obiettivo della tecnica LDA è individuare le dimensioni che discriminano meglio i pattern.

La riduzione della dimensionalità il vantaggio di comprimere il volume dei dati, riducendo la complessità computazionale dell'algoritmo di apprendimento, ma d'altro lato può degradare le informazioni e le prestazioni predittive dell'algoritmo di apprendimento.

5 Case Study

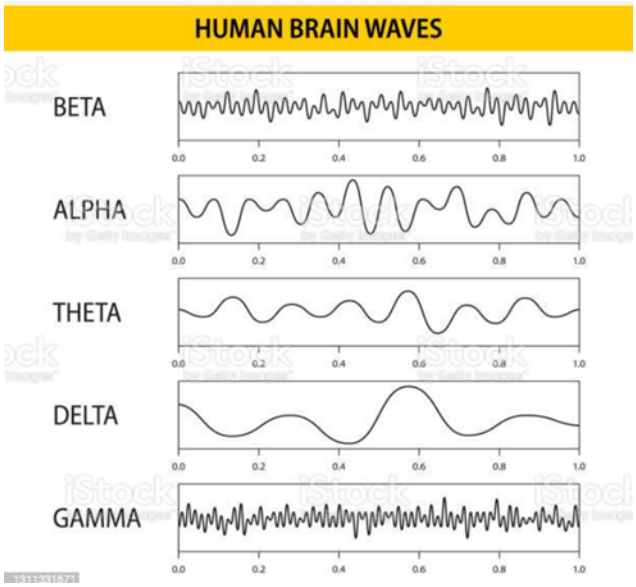
5.1 Il DataSet MAHNOB-HCI

Caratterizzare i contenuti multimediali con tag pertinenti, affidabili e discriminanti è vitale per il recupero delle informazioni multimediali. Con la rapida espansione dei contenuti multimediali digitali, sono necessari metodi alternativi alla codifica esplicita esistente per arricchire il pool di contenuti contrassegnati. Attualmente, i siti di social media incoraggiano gli utenti a etichettare i propri contenuti. Tuttavia, l'intento degli utenti nell'etichettare i contenuti multimediali non sempre corrisponde agli obiettivi di recupero delle informazioni. Gran parte dei tag definiti dall'utente sono motivati dall'aumento della popolarità e della reputazione di un utente in una comunità online o basati su giudizi individuali ed egoistici. Inoltre, gli utenti non valutano i contenuti multimediali in base agli stessi criteri. Alcuni potrebbero contrassegnare i contenuti multimediali con parole per esprimere le proprie emozioni, mentre altri potrebbero utilizzare i tag per descrivere il contenuto. Per esempio, il principio del tagging implicito consiste nel sostituire l'input dell'utente trovando automaticamente tag descrittivi per i contenuti multimediali, derivati dalla risposta naturale di un osservatore. Ad esempio, le emozioni che qualcuno mostra. Per facilitare la ricerca su questa nuova area del tagging multimediale, abbiamo registrato un database di risposte degli utenti ai contenuti multimediali.



Nel dataset sono presenti i valori riguardanti le sensazioni che i partecipanti all'esperimento hanno riportato, uniti ai parametri registrati dalle varie apparecchiature, in particolare le onde alpha e theta. I partecipanti dovevano assegnare un valore a: un' etichetta emotiva, arousal, valence, dominance, predictability. Le etichette emotive includevano: neutro, ansia, divertimento, tristezza, gioia, disgusto, rabbia, sorpresa e paura. Le onde cerebrali sono suddivise in 5 categorie, a seconda dello spettro di frequenza: theta (4 Hz <f< 8 Hz), slow alpha (8 Hz <f< 10 Hz), alpha (8 Hz <f< 12 Hz), beta (12 Hz <f< 30 Hz), and gamma (30 Hz < f).

A 30 partecipanti sono stati mostrati frammenti di filmati e immagini, mentre li monitoravano con 6 videocamere, un microfono indossato sulla testa, un tracker dello sguardo oculare, nonché sensori fisiologici che misuravano ECG, EEG (32 canali), ampiezza della respirazione e temperatura cutanea. Ogni esperimento consisteva in due parti. Nella prima parte sono stati mostrati frammenti di film e ad un partecipante è



stato chiesto di annotare il proprio stato emotivo dopo ogni frammento su una scala di valenza ed eccitazione, come mostrato a destra. Nella seconda parte dell'esperimento, sono state mostrate immagini o frammenti di video insieme a un tag nella parte inferiore dello schermo. In alcuni casi, il tag descriveva correttamente qualcosa sulla situazione. Tuttavia, in altri casi il tag non si applicava effettivamente all'elemento multimediale. Di seguito sono riportati due esempi, sovrapposti alle misurazioni del tracciamento oculare come punti rossi e linee rispettivamente per posizioni e spostamenti dello sguardo. Dopo ogni elemento, a un partecipante è stato chiesto di premere un pulsante verde se era d'accordo con il tag applicabile all'elemento multimediale, o di premere un pulsante rosso in caso contrario. Durante l'intero esperimento, audio, video, dati sullo sguardo e dati fisiologici sono stati registrati simultaneamente con un'accurata sincronizzazione tra i sensori. Il database è disponibile gratuitamente per la comunità di ricerca.

6 Risultati sperimentali

6.1 Una prima regressione sul DataSet

Vogliamo effettuare una regressione sul parametro “valence” il quale indica se un'emozione è piacevole o sgradevole, e successivamente “arousal”, il quale indica se un'emozione è forte o debole. Ciò, in modo da poter capire a quali frequenze di onde cerebrali sono associati determinati punti sul piano PAD. La regressione corrisponde al valore medio atteso in base ai dati di input che noi immettiamo. Grazie alla regressione si può parlare di previsione sui valori attesi. Intuitivamente la regressione consente di trovare la relazione tra input e output, in modo da predire output futuri avendo a disposizione solamente l'input. E' buona pratica nel machine learning eseguire l'addestramento su un set di dati per poi verificare i risultati delle sue predizioni con un altro set contente dati non visti durante l'addestramento. Questo garantisce che il nostro modello sia in grado di generalizzare su dati sconosciuti e quindi ha realmente "imparato" dai dati, piuttosto che limitarsi a memorizzare il set di addestramento, condizione conosciuta come overfitting. Mettiamo, allora, una parte del nostro dataset nella variabile “data unseen”, che useremo in seguito. Quello che facciamo allora è dividere il nostro unico dataset in due insiemi che chiamiamo:

- training set, sarà usato nella fase di training per consentire al modello di apprendere la relazione nascosta tra i dati. Presente nella variabile “data” del codice;
- testing set, sarà usato nella fase di testing per valutare le prestazioni di generalizzazione del modello, calcolando l'errore tra i risultati predetti e quelli reali. Presente nella variabile “data unseen”;

Usiamo il 90% del dataset come insieme di training. La parte restante è chiamata hold out e, per completezza d'informazione, costituisce il 20%. Dopo aver effettuato la regressione, otterremo dei valori restituiti dal modello, i quali andranno interpretati per verificare che il modello sia in grado di eseguire un “buon fitting” sul nostro dataset. I valori, contestualizzando, saranno i seguenti, detto y = valore originale, $\hat{y}$ = valore predetto, $\bar{y}$ valore medio di y

- Mean absolute error, rappresenta la differenza fra il valore originale e quello predetto, facendo la media fra la somma delle differenze fra i due valori in valore assoluto:
$$MAE = \frac{1}{N} \cdot \sum_{i=1}^{n} |y_i - \hat{y}|$$
- Means quare error, rappresenta la differenza fra il valore originale e quello predetto, facendo la media fra la somma delle differenze fra i due valori elevata al quadrato:
$$MSE = \frac{1}{N} \cdot \sum_{i=1}^{n} (y_i - \hat{y})^2$$

- RMSE è la radice quadrata dell'indice sopra, è il tasso di errore:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{n} (y_i - \hat{y})^2}$$

- R^2 Rappresenta il coefficiente di adattamento dei valori predetti rispetto ai valori originali. Il valore è compreso fra 0 e 1 e va interpretato come una percentuale. Più alto è il valore, migliore è il modello:

$$R^2 = 1 - \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2}$$

- RMSLE Root Mean Squared Log Error: stavolta la differenza è fra il logaritmo del valore originale rispetto a quello del valore predetto:

$$RMSLE = \sqrt{\frac{1}{N} \sum_{i=1}^{n} (log(x_i + 1) - log(y_i + 1))^2}$$

- MAPE Mean absolute percentage error: F è il valore predetto, A quello attuale:

$$MAPE = \frac{1}{N} \sum_{i=1}^{n} \frac{|A_i - F_i|}{|A_i|}$$

6.1.1 caso 'valence'

Prima di tutto abbiamo bisogno di trovare il modello più adatto al nostro scopo, e per far ciò utiliziamo la "compare_models()". Il modello che restituisce le migliori prestazioni è il decision tree, come vediamo nella tabella dal valore di R2

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
dt	Decision Tree Regressor	0.1780	3.842000e-01	0.6176	0.9308	0.1397	0.0596	6.084
br	Bayesian Ridge	0.9146	1.482900e+00	1.2153	0.7323	0.2671	0.3279	1.255
omp	Orthogonal Matching Pursuit	0.9285	1.490200e+00	1.2203	0.7312	0.2725	0.3335	0.132
ridge	Ridge Regression	0.9223	1.516600e+00	1.2280	0.7262	0.2686	0.3293	0.679
lr	Linear Regression	0.9350	1.542700e+00	1.2397	0.7215	0.2748	0.3325	0.593
knn	K Neighbors Regressor	0.6993	1.619400e+00	1.2714	0.7082	0.2686	0.2633	7.471
en	Elastic Net	1.8452	4.815700e+00	2.1940	0.1319	0.4529	0.6973	2.628
lasso	Lasso Regression	1.8617	4.854700e+00	2.2029	0.1250	0.4563	0.7077	2.218
llar	Lasso Least Angle Regression	1.9899	5.554900e+00	2.3566	-0.0014	0.4870	0.7760	0.129
huber	Huber Regressor	2.4405	1.095360e+01	3.3007	-0.9746	0.5869	0.8127	5.287
par	Passive Aggressive Regressor	11.0678	5.620949e+03	67.5762	-1023.7519	0.9250	3.9111	0.394
lar	Least Angle Regression	279.5854	1.060842e+06	624.3973	-194343.5671	2.7070	95.3689	0.342

Dunque andiamo a creare il nostro modello, il quale restituisce i seguenti valori:

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	0.2273	0.5445	0.7379	0.9055	0.1683	0.0845
1	0.162	0.3401	0.5832	0.9409	0.1218	0.0437
2	0.179	0.4504	0.6711	0.9177	0.1469	0.0562
3	0.1866	0.3732	0.6109	0.9321	0.1421	0.0666
4	0.1637	0.3316	0.5759	0.9393	0.1356	0.0538
5	0.1552	0.2841	0.533	0.9478	0.1218	0.0518
6	0.1791	0.3642	0.6035	0.9335	0.1355	0.0574
7	0.1825	0.3846	0.6201	0.9343	0.1443	0.0566
8	0.174	0.3676	0.6063	0.9324	0.1304	0.0575
9	0.1706	0.4015	0.6337	0.9245	0.1505	0.0679
Mean	0.178	0.3842	0.6176	0.9308	0.1397	0.0596
Std	0.0189	0.0678	0.053	0.0116	0.0133	0.0106

Di tutti questi valori, nel momento in cui andiamo a comparare i modelli, viene mostrata la media. Adesso guardiamo i risultati di predizione del modello:

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Decision Tree Regressor	0.1631	0.3563	0.5969	0.9366	0.1294	0.0529

L'R2 sul test/hold-out è 0,9366 rispetto allo 0,9308 ottenuto sui risultati della create_model(). Questa non è una differenza significativa. Se c'è una grande variazione tra i risultati del test/hold-out e quelli della create_model(), questo è normalmente dovuto all' over-fitting, come potrebbe essere dovuto a diversi altri fattori e richiederebbe ulteriori indagini. In questo caso, andremo avanti con la finalizzazione del modello e la previsione su dati invisibili (la parte del dataset inserita nella variabile "data_unseen" che avevamo separato all'inizio e non abbiamo utilizzato durante la scelta del modello). Andiamo dunque a finalizzare il modello e vedere le prestazioni sui dati non ancora visti:

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Decision Tree Regressor	0.0695	0.1325	0.3641	0.9616	0.0787	0.0189

6.1.2 caso 'arousal'

Ripetiamo gli stessi passaggi cambiando il target in "arousal". Anche questa volta il modello scelto è il decision tree regressor, per via delle migliori prestazioni sul parametro R2:

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
dt	Decision Tree Regressor	0.2814	7.406000e-01	0.8594	0.8523	0.1730	0.0814	6.547
knn	K Neighbors Regressor	0.6745	1.488100e+00	1.2179	0.7029	0.2672	0.2678	7.489
br	Bayesian Ridge	1.1636	2.262300e+00	1.5011	0.5476	0.3156	0.4080	1.303
ridge	Ridge Regression	1.1618	2.285000e+00	1.5081	0.5430	0.3163	0.4053	0.684
lr	Linear Regression	1.1690	2.318400e+00	1.5187	0.5362	0.3192	0.4074	0.623
omp	Orthogonal Matching Pursuit	1.2299	2.390700e+00	1.5457	0.5230	0.3316	0.4368	0.132
en	Elastic Net	1.7573	4.312800e+00	2.0763	0.1399	0.4506	0.7037	3.070
lasso	Lasso Regression	1.7990	4.449500e+00	2.1091	0.1126	0.4587	0.7261	2.622
llar	Lasso Least Angle Regression	1.9396	5.018600e+00	2.2400	-0.0009	0.4875	0.8032	0.134
huber	Huber Regressor	2.3974	9.404000e+00	3.0648	-0.8750	0.6088	0.8450	5.324
par	Passive Aggressive Regressor	10.9424	3.181534e+03	54.0521	-631.0284	0.9981	4.0770	0.404
lar	Least Angle Regression	301.7409	1.282681e+06	458.0102	-257284.9667	2.5078	109.8223	0.338

Ottenendo risultati leggermente peggiori rispetto al caso precedente. I valori delle altre colonne si riferiscono al valore medio misurato rispetto a quei parametri. I risultati di predizione del modello:

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Decision Tree Regressor	0.2937	0.768	0.8764	0.8489	0.1777	0.0917

Mentre quelli ottenuti dopo aver finalizzato il modello, sui dati non ancora visti, sono:

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Decision Tree Regressor	0.0203	0.0492	0.2217	0.9887	0.0484	0.0075

6.1.3 Prime considerazioni ed uso di sklearn

Guardando i risultati ottenuti nelle pagine precedenti, si nota che R2 del modello sui dati non visti è superiore a quello sui dati di training. Per verificare se il modello dà buoni risultati anche utilizzando altre librerie, testiamolo con sklearn, in quanto Pycaret opera ad un livello di astrazione troppo alto per riuscire a capire se sia stata eseguita una qualche azione di processing dei dati che non risulta dall'esposizione dei risultati sopra elencati. Allora andiamo ad importare di nuovo il nostro dataset, insieme a diverse librerie:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import r2_score
```

Andiamo dunque a constatare se, agendo ad un livello di astrazione un po' più basso, otteniamo risultati diversi o ci accorgiamo di qualche trasformazione compiuta dalla libreria a nostra insaputa. Il dubbio nasce dal fatto che il modello si addestra su determinati dati, sui quali, dunque, sarà necessariamente più preparato. Quando, però, andiamo a stampare a video il risultato dell'addestramento sui dati di training e su quelli di test, quest'ultimi vengono più alti, contraddicendo il comportamento che dovrebbe normalmente avere un modello. Ripetiamo il fitting, partendo dal caso di 'arousal'. Come fatto in precedenza con Pycaret, non consideriamo le colonne "video", "window" e "subject". Dunque, 'arousal' diventa il parametro della prima colonna, e per selezionarlo utilizziamo "iloc[:,0:1]". Nel caso in questione, stiamo eseguendo il fit sull'intero dataset, dunque quando andiamo ad eseguire la predizione sui dati di test, questi sono tutti dati già visti ed il modello sa come rispondere. Perciò, i risultati ottenuti sono molto positivi:

```
Mean Absolute Error: 0.251
Mean Squared Error: 0.413
Root Mean Squared Error: 0.643
R2: 0.919
```

La situazione cambia nel momento in cui andiamo ad utilizzare solamente i dati di training per effettuare il fit, dunque "regr.fit(X_train, y_train)" al posto di "regr.fit(X,y)". In questo caso, quando poi andiamo ad eseguire la predizone sui dati di test (stavolta non sono dati che il modello ha già visto, quindi deve dimostrare di saper apprendere anche su dati non ancora visti), i risultati sono inferiori rispetto al caso precedente, ed anche rispetto a quelli ottenuti utilizzando Pycaret:

```
Mean Absolute Error: 0.251
Mean Squared Error: 0.413
Root Mean Squared Error: 0.643
R2: 0.674
```

Risultati molto simili si ottengono effettuando la regressione su "valence".

6.1.4 Utilizzo di altri modelli su Sklearn, prova di ExtraTreesRegressor

Dunque, l'uso di Sklearn è stato introdotto per provare a risolvere il problema iniziale della differenza di punteggio R2 fra dati di training e test, ed ha invece introdotto un nuovo problema: il rendimento del modello decision tree è significativamente inferiore rispetto a quello ottenuto con Pycaret. Ci chiediamo allora, adesso, se questo è un problema soltanto del decision tree o di tutti gli altri modelli che, per qualche motivo, anche se ciò sarebbe molto strano, rendono diversamente fra le due diverse librerie. Se andiamo ad effettuare, con sklearn, una regressione usando altri modelli, i risultati ottenuti sono molto simili a quelli indicati da Pycaret, sono qui riportati tre esempi:

- Con KNN: "regr = KNeighborsRegressor(n_neighbors=2)" importando "from sklearn.neighbors import KNeighborsRegressor" otteniamo:

```
Mean Absolute Error: 0.701
Mean Squared Error: 1.869
Root Mean Squared Error: 1.363
R2: 0.690
```

- Con lasso regression: "regr = linear_model.Lasso(alpha= 0.1)" importando "from sklearn import linear_model" otteniamo:

```
Mean Absolute Error: 1.895
Mean Squared Error: 4.870
Root Mean Squared Error: 2.210
R2: 0.115
```

- - Con Baesyan Ridge: "regr = linear_model.BayesianRidge()" importando "from sklearn import linear_model" otteniamo:

```
Mean Absolute Error: 1.895
Mean Squared Error: 4.870
Root Mean Squared Error: 2.210
R2: 0.115
```

Dunque il problema è da ricercarsi altrove, visto che altri modelli funzionano con prestazioni molto simili fra le due librerie. Proviamo allora a vediamo col modello potenziato dei decision tree, ovvero ExtraTreesRegressor, sempre effettuando una regressione. Nel caso di "arousal", del codice in figura, il risultato che otteniamo per lo score, che nel caso del metodo in questione rappresenta il valore di R2, è:

0.629

Otteniamo quindi prestazioni nettamente superiori a prima, ma che rimangono comunque inferiori a quelle utilizzando Pycaret. Anche effettuando la regressione su "valence", dunque cambiando la riga "y=dataset.iloc[:,0]" in y=dataset.iloc[:,1], il risultato non è molto diverso:

0.605

6.2 Utilizzo della classificazione

Proviamo allora a spostare il problema su di una classificazione: non prevediamo più numeri reali, ma un numero intero. Ciò ed atto a verificare se i modelli rispondono meglio per il tipo di problema che dobbiamo affrontare, in quanto la regressione si è dimostrata poco efficiente ai nostri scopi.

6.2.1 Uso del classificatore con decision tree

Il codice è: In cui prendiamo la colonna "arousal" e la trasformiamo in valori interi tramite "round(0)", visto che per il classificatore ci servono valori in quella forma. Il

risultato finale è:

0.481

Ovvero il modello classifica correttamente circa il 48% dei campioni che si trova davanti. Similmente, nel caso di "valence" i risultati ottenuti di predizione si aggirano intorno al 50%:

0.502

Proviamo allora ad utilizzare il modello potenziato del classificatore con decision tree, ovvero l'ExtraTreesClassifier, e stavolta i risultati ottenuti sono migliori: sia nel caso 'arousal' che nel caso 'valence' la predizione si aggira attorno al 90%.

6.3 PCA sul nostro dataset

Adesso che, con la classificazione, siamo riusciti ad ottenere risultati positivi, proviamo ad indagare un altro po' su possibili cause del problema riguardante la regressione. Proviamo a verificare se l'R2 nella regressione effettuata in precedenza viene più alto in test piuttosto che in training per un qualche problema di dimensionalità. Una possibile causa potrebbe essere che la stessa informazione venga ripetuta più volte nei campioni misurati all'interno dell'esperimento dal quale vengono presi i dati per effettuare la regressione. Ciò potrebbe essere causato, per esempio, da prossimità spaziali o da prossimità di banda, ovvero:

- prossimità di due elettrodi, troppo vicini, che quindi raccolgono informazioni non indipendenti fra loro
- prossimità della banda frequenziale delle onde celebrali, in quanto le onde che vengono misurate durante l'esperimento rientrano in un intervallo di frequenze molto vicine fra loro.

Torniamo ad utilizzare Pycaret, ancora con il classificatore extraTree. Proviamo, però, ad effettuare una riduzione della dimensionalità, per vedere se in qualche modo possa influire sui risultati delle elaborazioni fatte. Abbiamo effettuato una riduzione della dimensionalità. Il dataset partiva da 32 colonne per le onde theta, 32 per le onde gamma, 32 per le onde beta e 32 per le onde alpha. Abbiamo preso la prima colonna di ognuno di questi gruppi e le abbiamo riunite in una nuova colonna. Dunque, inizialmente avevamo:

theta_1	theta_2	... theta_32
gamma_1	gamma_2	... gamma_32
beta_1	beta_2	... beta_32
alpha_1	alpha_2	... alpha_32

E le colonne sono state riunite come:

theta_1	gamma_1	beta_1	alpha_1
theta_2	gamma_2	beta_2	alpha_2
...	...		...
theta_32	gamma_32	beta_32	alpha_32

Effettuando quindi un "cambio di base", come previsto nelle PCA. La nuova disposizione è stata scelta in quanto ogni numero indica un diverso elettrodo. Quindi, per esempio, "theta_1" indica le onde di frequenza theta registrate dall'elettrodo 1, "theta_2" indica le onde di frequenza theta registrate dall'elettrodo 2, e così via, mentre "beta_1" indica le onde di frequenza beta registrate con l'elettrodo 1, "beta_2" indica le onde di frequenza beta registrate con l'elettrodo 2 e così via. Quindi, il nuovo raggruppamento è per elettrodo e non più per diverse onde. Otteniamo i seguenti risultati per "valence":

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Extra Trees Classifier	0.8951	0.9839	0.854	0.899	0.8942	0.8584	0.8595

```
ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, max_samples=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1,
                     oob_score=False, random_state=123, verbose=0,
                     warm_start=False)
```

Mentre i seguenti per "arousal":

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Extra Trees Classifier	0.8821	0.9837	0.8542	0.8881	0.8814	0.8457	0.8472

```
ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, max_samples=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1,
                     oob_score=False, random_state=123, verbose=0,
                     warm_start=False)
```

6.3.1 Interpretazione dei risultati

Abbiamo dunque riscontrato che l'accuratezza rimane all'incirca uguale a quella ottenuta nelle precedenti prove. L'accuratezza corrisponde al numero di elementi classificati correttamente diviso per il numero totale di elementi nel set di test. È compresa tra 0 (meno accurato) e 1 (più accurato). È la metrica più semplice e facile da capire, però non è sufficiente a valutare le performance di un modello perché non è in grado di dirci le cause del malfunzionamento. Con sklearn si aggirava intorno al 90%, stesso valore si ottiene, centesimo più centesimo meno, anche con Pycaret. Essa viene calcolata tramite la funzione accuracy_score(), che riceve come argomenti un array che contiene le etichette corrette e un array che contiene le etichette previste. Questa funzione confronta

le etichette considerate esatte con le etichette previste e ritorna la frazione di campioni classificati correttamente. Chiariamo adesso il significato di alcuni termini:

- True Positive: rappresenta il caso in cui il modello prevede correttamente la classe positiva.
- True Negative: rappresenta il caso in cui il modello prevede correttamente la classe negativa.
- False Positive: rappresenta il caso in cui il modello prevede in modo scorretto la classe positiva.
- False Negative: rappresenta il caso in cui il modello prevede in modo scorretto la classe negativa

Nel caso, per esempio, di classificazione di email in "spam" oppure "no spam", se il modello classifica un'email come "spam", ed effettivamente lo è, siamo nel caso di vero positivo, così come se classificasse un'email come "no spam" e questa si rivelasse effettivamente non di spam. Viceversa quando parliamo di falsi positivi e falsi negativi. Per quanto riguarda gli altri parametri ottenuti, anch'essi appartenenti all'intervallo [0,1] :

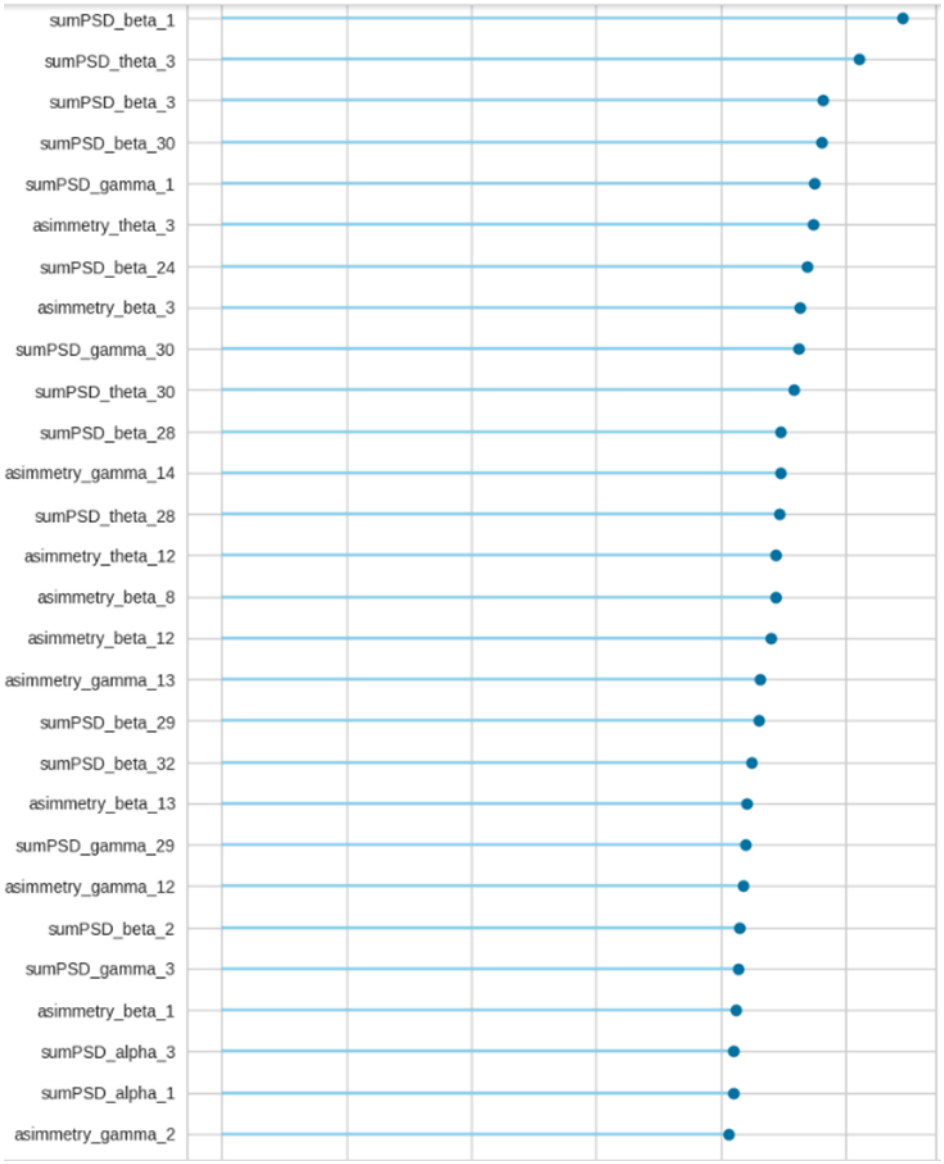
- La precisione è il rapporto tra il numero delle previsioni corrette di un evento sul totale delle volte che il modello lo prevede, quindi il modello prevede in modo leggermente peggiore dopo aver ridimensionato il dataset. Quando un modello è preciso per una classe, ogni volta che prevede l'evento sbaglia raramente. Potrebbe però non prevedere tutti gli eventi ossia non essere selettivo/sensibile. Nel nostro caso la precisione si aggira intorno al 88%.
- Il richiamo (recall) misura la sensibilità del modello. E' il rapporto tra le previsioni corrette per una classe sul totale dei casi in cui si verifica effettivamente. F1 consente di bilanciare la precisione e il richiamo. Il valore di recall si ottiene tramite la funzione recall_score(), che riceve come argomenti due array (uno con i valori corretti e l'altro con le previsioni). Essa restituisce il recall della classe positiva nella classificazione binaria o la media ponderata del recall di ciascuna classe per attività multiclass. Nel nostro caso recall si aggira intorno al 85%.
- AUC, area under the curve, ovvero l'area sotto la curva che traccia il tasso di veri positivi rispetto al tasso di falsi positivi. Nel caso in questione i valori di AUC risultano molto elevati, intorno al 98%.

L'accuratezza rimane dunque circa la stessa, anche effettuando la riduzione di dimensionalità, comunque elevata, ottenendo prestazioni molto simili col modello rispetto al precedente uso.

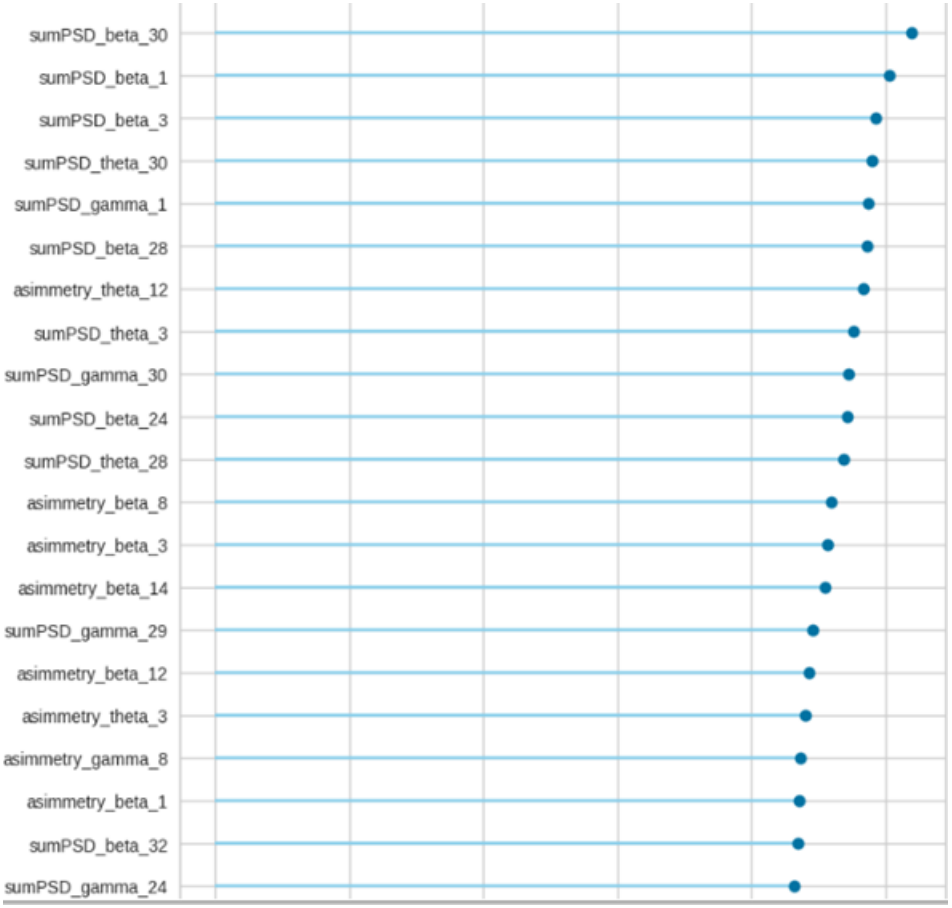
6.4 Analisi delle feature importance

Riportiamo i grafici delle feature importance, nel caso con e senza principal component analysis. Più un attributo è in alto in classifica, più è significativo il suo utilizzo per le predizioni. Riportiamo adesso la parte alta della tabella nei due casi senza riduzione di dimensionalità.

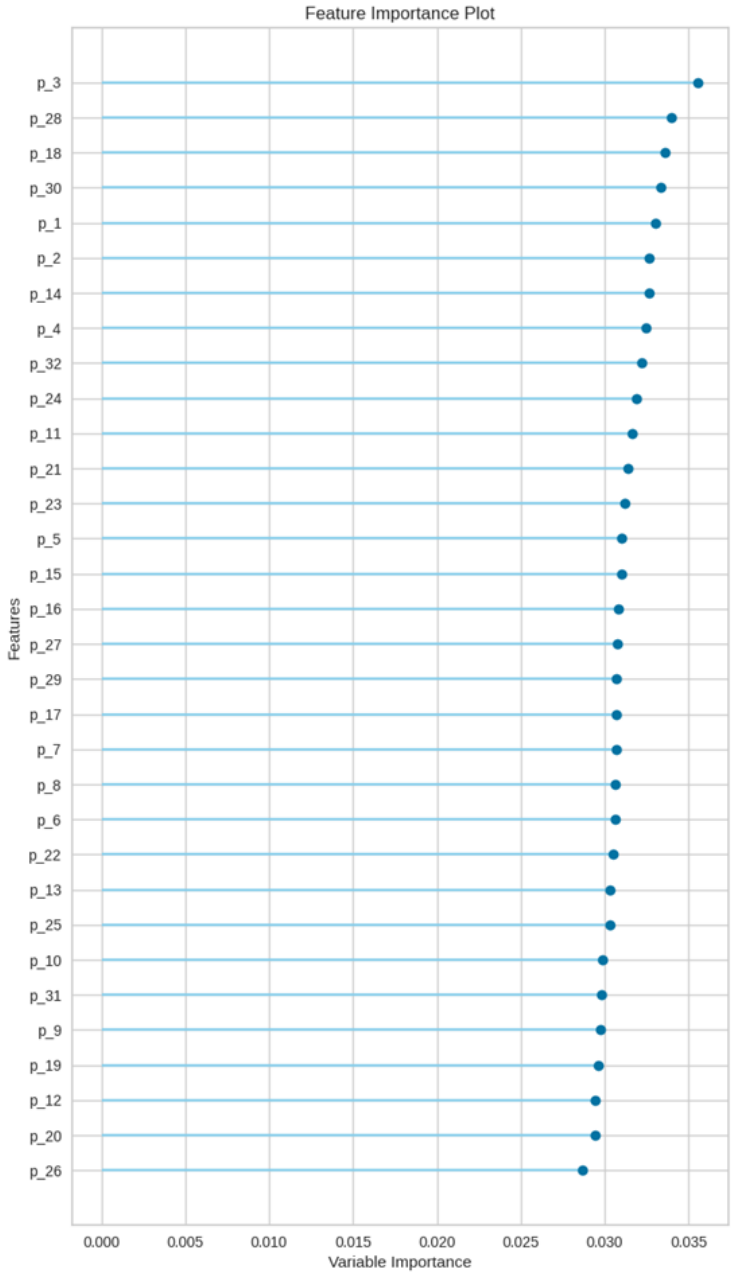
Senza PCA, caso arousal:



Senza PCA, caso valence:



Con PCA otteniamo:



Dove "P_3" indica il terzo elettrodo, "p_28" indica l'elettrodo numero 28 e così via.

6.4.1 Importanza canale

Per fare un confronto equo tra i due approcci, con e senza PCA, sarebbe utile capire quanto spesso le feature di un certo canale appaiono in alto sulla classifica della feature importance (nel caso senza PCA, in quanto il caso con PCA è già stato rappresentato in precedenza). Con "canale" intendiamo un determinato elettrodo: per esempio, nel

grafico prima citato del caso con PCA, al primo posto si ha il canale “3”, al secondo il canale “28” e cosi via. Questo ci permetterà di mettere a confronto i due grafici in maniera più diretta, avendo in entrambi i casi un ordinamento basato sull' importanza di ciascun canale. Potremmo ottenere facilmente questa misura come:

$$IMPORTANZACANALEN = \frac{1}{P} . \sum_{i=1}^{P} \frac{(totale\_features - posizione\_ordinamento)}{totale\_features}$$

Esempio: dati 3 soli canali e 2 sole features (alpha e beta) un ordinamento di feature importance del tipo

- Posizione 1) alpha_canale_2
- Posizione 2) beta_canale_3
- Posizione 3) beta_canale_2
- Posizione 4) alpha_canale_1
- Posizione 5) alpha_canale_3
- Posizione 6) beta_canale_1

genera le seguenti importanze per canale

- IMPORTANZA CANALE 1 = (((6 - 4)/6) + ((6 - 6)/6)) / 2 = 0.16
- IMPORTANZA CANALE 2 = (((6 - 1)/6) + ((6 - 3)/6)) / 2 = 0.66
- IMPORTANZA CANALE 3 = (((6 - 2)/6) + ((6 - 5)/6)) / 2 = 0.41

In questo modo il canale 3, che appare più spesso in alto nella classifica, ha un valore di importanza maggiore. Al contrario il canale 1. Eseguendo il codice, notiamo che la classifica viene all'incirca identica a quella del caso con PCA stilata precedentemente. La similarità della spiegazione potrebbe essere proprio relativa al dataset in questione: vuol dire che le dinamiche che osservate sono abbastanza astabili e che tutti i metodi di spiegazione sono concordi su cosa sia più o meno importante. L'array restituito è:

```
[ 22 7 31 12 25  6  4  8 10 20 17  9 26 16 13 19 23 24 15 5  27 29 1  21
 30  2 11 32 18 14 28 3]
```

Riassumiamo in una tabella i due risultati, con e senza PCA:

Con PCA	Senza PCA
3	3
28	28
18	14
30	18
1	32
2	11
14	2
4	30
32	21
24	1
11	29
21	27
23	5
5	15
15	24
16	23
27	19
29	13
17	16
7	26
8	9
6	17
22	20
13	10
25	8
10	4
31	6
9	25
19	12
12	31
20	7
26	2

7 Conclusioni

Per concludere, è stato implementato un sistema di classificazione di un dataset, usando un approccio basato su cloud e facendo uso di librerie low-code di Python, in particolare di PyCaret e di Sklearn. Infine, è stata effettuata una rielaborazione dei dati, tramite una riduzione di dimensionalità, ed è stata condotta un'analisi delle performance ed un'interpretazione dei modelli utilizzati per mezzo di determinate metriche e un'indagine delle feature importance. Visti i risultati ottenuti nei capitoli precedenti:

- La regressione non ha risposto in modo positivo alle varie elaborazioni, in quanto PyCaret effettuava delle elaborazioni interne non visibili dall'esterno, ed è stata infatti sostituita dalla classificazione come metodo per creare il modello. La classificazione ha, infatti, risposto positivamente alle varie prove. Il modello del decision tree, il quale si è rivelato come modello più performante nel caso della regressione, non ha però ottenuto gli stessi risultati con la classificazione: pur non avendo scarsissime prestazioni, l'accuratezza si aggirava intorno al 60%, comunque troppo poco per un modello di machine learning. Infatti, pensiamo al caso di classificazione in "0" e "1": se usassimo una moneta perfetta per classificare le istanze nelle due classi, otterremmo (coi grandi numeri) il 50% di accuratezza. Dunque, non ci si può ritener soddisfatti del precedente risultato, visti li sforzi necessari a raccogliere i dati e determinare ed addestrare i modelli. Si è allora optato per l'Extra Trees Classifier, il quale sia nel caso di "valence" come parola chiave, sia nel caso di "arousal", ottiene un'accuratezza superiore o uguale al 90%, di cui ci possiamo dunque ritenere soddisfatti.

- Successivamente si è ricorsi alla riduzione di dimensionalità, per provare a rilevare e correggere possibili interferenze fra valori degli elettrodi vicini (nello spazio o nella frequenza). Il valore di accuratezza non cambia però molto nella classificazione successiva alla PCA, pur migliorando lievemente gli altri parametri. Comunque, essendo 0.92 anche prima di effettuare curse of dimensionality, anche di questo risultato si possono cogliere gli aspetti positivi.

- L'ultima operazione compiuta è stata quella dell'analisi delle feature più importanti per le predizioni del modello. E' stata compiuta una prima analisi considerando il data set prima della PCA, e questa mostra che nel caso "arousal" e nel caso "valence", i parametri più importanti si discostano significativamente, dunque i metodi non sono concordi su quali siano i valori più importanti nell'uno e nell'altro caso. Invece, prendendo la tabella nel caso con PCA, se andiamo a calcolare l'importanza di ogni canale del precedente caso, questa coincide all'incirca con la classifica ottenuta. Dunque, in questo caso vengono ritenute come più importanti gli stessi valori.

Riassumendo, attraverso varie librerie e la potenza di calcolo fornita da Google Colab, è stato possibile realizzare in modo rapido e semplice un sistema in grado di classificare in modo molto accurato valori a partire da un set di dati non ancora visualizzato dal modello, con risultati abbastanza positivi. Utilizzando inoltre PyCaret è stato possibile

ridurre i tempi e le righe di codice che un lavoro simile avrebbe richiesto. Dunque, così come Python permette anche a programmatori alle prime armi di compiere elaborazioni molto potenti col minimo sforzo, questo lavoro potrebbe permettere anche a coloro che non hanno molte competenze nell'ambito del Machine Learning di effettuare classificazioni di dati, con tutte le comodità che ne conseguono.

8 Appendice A

Manuali utilizzati:

- Pandas: https://pandas.pydata.org/docs/user_guide/index.html
- Sklearn: https://scikit-learn.org/stable/user_guide.html
- Numpy: https://numpy.org/doc/stable/user/
- Matplotlib: https://matplotlib.org/
- Scikitplot: https://scikit-plot.readthedocs.io/en/stable/
- PyCaret: https://pycaret.org/guide/

Software utilizzati:

- https://www.overleaf.com
- https://www.smartdraw.com
- https://colab.research.google.com/
- https://drawio-app.com/

Comune a tutti i pezzi di codice è l'importazione del dataset:

```
from google.colab import files

uploaded = files.upload()
```

Vediamo la regressione con PyCaret:

```
# installazione pacchetti PyCaret
!pip install --use-deprecated=legacy-resolver pycaret[full]
!pip install pandas-profiling==3.1.0
!pip uninstall -y pyyaml
!pip install pyyaml==5.4.1

import io
import pandas as pd

dataset = pd.read_csv(io.BytesIO(uploaded['EEG_MANHOB_w8_s2.csv']))

# elimino le colonne che non mi servono
dataset = dataset.drop('video', axis=1)
dataset = dataset.drop('window', axis=1)
dataset = dataset.drop('subject', axis=1)
dataset = dataset.drop('dominance', axis=1)
dataset = dataset.drop('valence', axis=1)
dataset = dataset.drop('emotion', axis=1)
dataset = dataset.drop('predictance', axis=1)

# il 20% del dataset va nell'hold out
data = dataset.sample(frac=0.8)
data_unseen = dataset.drop(data.index)
data.reset_index(drop=True)
data_unseen.reset_index(drop=True)

from pycaret.regression import *
# eseguo la regressione su "valence", oppure "arousal"
exp_reg101 = setup(data = data, target = 'valence', session_id=123)

# eseguo la scelta del modello
best = compare_models()
regr = create_model(best)

# valuto il modello
evaluate_model(regr)
predict_model(regr);

final_model = finalize_model(regr)
print(final_model)

# guardo le prestazioni sui dati non ancora visti
unseen_predictions = predict_model(final_model, data=data_unseen)
unseen_predictions.head()

save_model(regr,'Final dt Model 2842022')
```

Regressione con Sklearn:

```
import pandas as pd
import io
import numpy as np
from sklearn.tree import ExtraTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import r2_score

# importo il dataset
dataset = pd.read_csv(io.BytesIO(uploaded['EEG_MANHOB_w8_s2.csv']))

# elimino le colonne che non mi servono
dataset = dataset.drop('video', axis=1)
dataset = dataset.drop('window', axis=1)
dataset = dataset.drop('subject', axis=1)
dataset = dataset.drop('dominance', axis=1)
dataset = dataset.drop('valence', axis=1)
dataset = dataset.drop('emotion', axis=1)
dataset = dataset.drop('predictance', axis=1)

# prendo arousal, in questo caso. Per avere valence, bisogna selezionare [:,1:2]
y = dataset.iloc[:,0:1]
X = dataset.iloc[:,5:133]

# splitto il dataset in train e test, con 20% di hold out
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)

regr = DecisionTreeRegressor()
regr.fit(X_train,y_train)

y_pred = regr.predict(X_test)

print('Mean Absolute Error:', metrics.mean_absoulute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print('R2:', r2_score(y_test, y_pred))
```

Per avere la regressione con l'ExtraTrees, nel codice precedente bisogna sostituire alla chiamata a DecisionTreeRegressor la seguente chiamata:

```
from sklearn.ensemble import ExtraTreesRegressor
regr = ExtraTreesRegressor()
```

Per la classificazione:

```
import pandas as pd
import io
import numpy as np
from sklearn.ensemble import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import r2_score

# importo il dataset
dataset = pd.read_csv(io.BytesIO(uploaded['EEG_MANHOB_w8_s2.csv']))

# elimino le colonne che non mi servono
dataset = dataset.drop('video', axis=1)
dataset = dataset.drop('window', axis=1)
dataset = dataset.drop('subject', axis=1)
dataset = dataset.drop('dominance', axis=1)
dataset = dataset.drop('valence', axis=1)
dataset = dataset.drop('emotion', axis=1)
dataset = dataset.drop('predictance', axis=1)

# prendo arousal, in questo caso. Per avere valence, bisogna selezionare [:,1:2]
y = dataset.iloc[:,0:1].round(0)
X = dataset.iloc[:,5:133]

# splitto il dataset in train e test, con 20% di hold out
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)

clas = DecisionTreeClassifier()
clas.fit(X_train,y_train)

test_score = clas.score(X_test, y_test)
print(test_score)
```

Come per la regressione, l'uso dell'ExtraTrees si ottiene modificando la chiamata al modello nel seguente modo:

```
from sklearn.ensemble import DecisionTreesClassifier
clas = ExtraTreesClassifier()
```

Per effettuare la PCA:

```
from pycaret.classification import *
import pandas as pd
import io
import numpy as np
from sklearn.preprocessing import StandardScaler

# importo il dataset
dataset = pd.read_csv(io.BytesIO(uploaded['EEG_MAHNOB_w8_s2.csv']))

# split of dataset, in order to locate the features
aux1 = "activity_theta_"
aux2 = "activity_alpha_"
aux3 = "activity_beta_"
aux4 = "activity_gamma_"

for i in range(1,33):
  globals()[f"f_{i}"] = []      # dichiaro gli array

for i in range(1,33):
  for j in range(1,5):
    globals()[f"f_{i}"] += [globals()[f"aux{j}"] + str(i)]

f_target = ['arousal']

# separating out the features
for i in range(1,33):
  globals()[f"a_{i}"] = []      # dichiaro gli array

for i in range(1,33):
  globals()[f"a_{i}"] = dataset.loc[:, globals()[f"f_{i}"]].values

# Separating out the target
a_target = dataset.loc[:, f_target].values

# standardizing the features
for i in range(1,33):
  globals()[f"n_{i}"] = []      # dichiaro gli array

for i in range(1,33):
  globals()[f"n_{i}"] = StandardScaler().fit_transform( globals()[f"a_{i}"] )

# Standardizing the target
```

```
n_target = StandardScaler().fit_transform(a_target)

#  iniziamo la PCA effettiva
from sklearn.decomposition import PCA

pca = PCA(n_components=1)

# dichiaro gli array
for i in range(1,33):
  globals()[f"p_{i}"] = []

for i in range(1,33):
    globals()[f"p_{i}"] = pca.fit_transform( globals()[f"n_{i}"] )

target = pca.fit_transform(n_target)

#ricomponiamo il dataset
for i in range(1,33)
  aux = globals()[f"p_{i}"]
  globals()[f"df_{i}"] = pd.DataFrame(data = aux, columns = [ aux ] )

# trasformo i valori da continui ad interi
df_target = pd.DataFrame(data = target, columns = ['target']).round(0)

data = pd.concat([df_1, df_2, df_3, df_4, df_5,
                  df_6, df_7, df_8, df_9, df_10,
                  df_11, df_12, df_13, df_14, df_15,
                  df_16, df_17, df_18, df_19, df_20,
                  df_21, df_22, df_23, df_24, df_25,
                  df_26, df_27, df_28, df_29, df_30,
                  df_31, df_32, df_target], axis = 1)

# eseguo la classificazione su "valence", oppure "arousal"
exp_reg101 = setup(data = data, target = 'target', session_id=123)

# uso extraTreeClassifier
et = create_model('et')

# valuto il modello
evaluate_model(et)
predict_model(et);
final_et = finalize_model(et)
print(final_et)
```

Per il calcolo dell'importanza di canale, basta che nel codice dell'ExtraTreesClassifier si aggiunga il seguente codice:

```
# Computing the importance of each feature
feature_importance = etc.feature_importances_

importanza_canale = [None]*32

for i in range(0,32):
  importanza_canale[i] = 0
  for j in range(1,4):
    importanza_canale[i] += ( (128 - feature_importance[4*i + j] )/128 )
  importanza_canale[i] = importanza_canale[i]/4

print(np.argsort(importanza_canale) + 1)
```

Riferimenti bibliografici

[1] Mohammad Soleymani, Jeroen Lichtenauer, Thierry Pun, and Maja Pantic, "A Multimodal Database for Affect Recognition and Implicit Tagging", January-March 2012

[2] Albert Mehrabian, " Pleasure-Arousal.Dominance: A General Framework for Describing and Measuring Individual Differences in Temperament", University of California, Los Angeles, Winter 1996

[3] https://pycaret.gitbook.io/docs/

[4] Dastan Hussen Maulud1, Adnan Mohsin Abdulazeez, "A Review on Linear Regression Comprehensive in Machine Learning", Duhok Polytechnic University, Duhok, Kurdistan Region, Iraq, December 29th, 2020

[5] Yan-yan Song, Ying Lu, "Decision tree methods: applications for classification and prediction", Department of Pharmacology and Biostatistics, Shanghai Jiao Tong University School of Medicine, Shanghai, China, 2015 Apr 9

[6] Cunningham, P., Cord, M., Delany, S.J. (2008). Supervised Learning. In: Cord, M., Cunningham, P. (eds) Machine Learning Techniques for Multimedia. Cognitive Technologies. Springer, Berlin, Heidelberg.

[7] Ghahramani, Z. (2004). Unsupervised Learning. In: Bousquet, O., von Luxburg, U., Rätsch, G. (eds) Advanced Lectures on Machine Learning. ML 2003. Lecture Notes in Computer Science(), vol 3176. Springer, Berlin, Heidelberg

[8] Douglas M. Hawkins, "The problem of overfitting", School of statistics, University of Minnesota, Minneapolis, October 30, 2003

[9] Daniel Berrar, "Cross-validation", Data Science Laboratory, Tokyo Institute of Technology, January 2018

[10] https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

[11] https://www.andreaminini.com/ai/machine-learning/riduzione-dimensionalita-dati

[12] "Predicting Students' Emotions Using Machine Learning Techniques", Nabeela Altrabsheh, Mihaela Cocea, and Sanaz Fallahkhair, School of Computing, University of Portsmouth, 2015

[13] Cunningham, S., Ridley, H., Weinel, J. et al. Supervised machine learning for audio emotion recognition. Pers Ubiquit Comput 25, 637–650 (2021)

[14] " Emotion Detection and Sentiment Analysis of Images" , Vasavi Gajarla, Aditi Gupta, Georgia Institute of Technology, 2020

[15] "Emotions from text: machine learning for text-based emotion prediction" , Cecilia Ovesdotter Alm,Dept. of Linguistics, Illinois, USA, Dan Roth, Dept. of Computer Science, Richard Sproat, 2005