



Università degli studi di Bari Aldo Moro

Dipartimento di Informatica
Corso di Laurea in Informatica

Tesi di Laurea in Informatica

Analisi espressioni facciali in contesto di e-learning

Professore
Prof. Stefano Ferilli
Relatrice
Prof. Berardina De Carolis

Laureando
Alessandro Carella

Anno Accademico 2022 - 2023

Dedica

Abstract

Il testo di tesi mira a identificare gli stati d'animo delle persone attraverso l'analisi di immagini o video che le riprendono attraverso l'utilizzo delle Action Units. Queste suddividono il volto della persona presente nelle immagini o nei video analizzati per effettuarne un'analisi.

Come attestato da diversi studi, gli stati d'animo incidono notevolmente sulle performance individuali, ed è per questo utile avere delle informazioni a riguardo col fine di rimodulare le attività e migliorare l'esperienza del singolo.

Lo studio trova applicazione sia in contesti di e-learning che di lavoro, o in qualsiasi altro contesto dove sia possibile ottenere delle riprese.

Un esempio pratico di un applicativo che usufruisce dei risultati del mio studio è l'utilizzo di questo, da parte di un docente, durante una lezione; l'insegnante, avendo cognizione dello stato d'animo dei suoi studenti, ha la possibilità di adattare lo stile di insegnamento e rendere più agevole la fruizione della lezione per i suoi alunni.

Indice

1	Stato dell'arte	3
1.1	Studi sull'engagement all'interno degli ambienti di studio	3
1.1.1	Riconoscimento delle emozioni cognitive in ambiente di e-learning	3
1.1.2	Le faccie dell'Engagement: Riconoscimento automatico dell'engagement degli studenti attraverso le espressioni facciali	4
1.1.3	Codifica facciale come mezzo per il monitoraggio continuo del comportamento degli studenti nell'e-learning	5
1.1.4	Predizione e localizzazione dell'engagement degli studenti in the wild	6
1.2	Cosa sono le AUs e il sistema FACS:	7
1.3	Studi sul riconoscimento delle emozioni FACS per scelta del modello da utilizzare	9
1.4	Studi che trattano campioni prelevati in scenari non controllati per una migliore precisione del modello	20
1.5	Metodologie di tagging delle immagini del dataset	21
1.6	Estrazione delle feature facciali dalle immagini e dai video dei dataset	21
1.7	Unione dei dataset ritrovati e relativa categorizzazione delle immagini all'interno di questo	23
2	Tecnologie, strumenti e librerie utilizzate	27
2.1	Tecnologie	27
2.1.1	Machine Learning	27
2.1.2	Computer Vision	28
2.1.3	Cuda	28
2.2	Strumenti	29
2.2.1	Visual Studio Code	29
2.3	Librerie	30
2.3.1	Pandas	30
2.3.2	Tqdm	30
2.3.3	Pickle	32
2.3.4	Torch	33
2.3.5	OpenCV	33

2.3.6	Scikit-learn	35
2.3.7	Py-Feat	36
3	Dataset	39
3.1	Generazione descrizione in linguaggio naturale	39
3.2	Estrazione delle Action Units utilizzando la libreria Py-feat	40
3.2.1	Dati ulteriori alle action units estratti da py-feat	41
3.2.2	Estrazione Action Units dalle immagini	42
3.2.3	Estrazione Action Units dai video	42
3.2.4	Pulizia dei dati	43
3.3	Resampling del dataset per migliorare la precisione delle predizioni	45
4	Modelli predittivi utilizzati	53
4.1	Algoritmi utilizzati per la creazione dei modelli predittivi	53
4.1.1	Random Forest	53
4.1.2	K-nearest neighbors	56
4.1.3	Naive Bayes Classifier	62
4.1.4	Support Vector Machine	65
4.1.5	Support Vector Regression	68
4.2	Random forest classifier	71
4.3	K-nearest Neighbors Classifier	73
4.4	Support vector machine Classifier	76
4.5	Naive Bayes Classifier	76
4.6	Support Vector Regression Classifier	76
4.7	Quale è il miglior modello predittivo?	77
4.7.1	Accuracy	77
4.7.2	Precision	78
4.7.3	Recall	78
4.7.4	Balanced Accuracy	79
4.7.5	K-fold Cross-Validation	79
4.7.6	Tabella dei risultati	81
5	Realizzazione interfaccia grafica	83
6	Il Framework WoMan per la Generazione dei Modelli Workflow dei Mood	87
6.1	Descrizione dei casi	87
6.2	Learning Workflow Structure and Weights	89
6.3	Realizzazione file per l'osservazione e la predizione dei mood attraverso il sistema WoMan	92
6.4	Generazione dei modelli workflow attraverso WoMan	95
7	Appendice	105

7.1	K-Nearest Neighbors classifier	106
7.2	Naive Bayes classifier	107
7.3	Random Forest classifier	108
7.4	Support Vector Machines classifier	109
7.5	Support Vector Regressor classifier	110

Prefazione

“The test of successful education is not the amount of knowledge that pupils take away from school, but their appetite to know and their capacity to learn.” -Sir Richard Livingstone 1941. [5]

La ricerca moderna, conseguentemente al fenomeno della pandemia da Covid-19, ha riservato attenzione sempre maggiore al tema della fruizione di contenuti, sia dal punto di vista dell’intrattenimento che dal punto di vista della produttività individuale.

La pandemia ha quindi consentito alla ricerca l’impiego di strumenti impattanti in questo campo, in quanto ormai possibile osservare e quindi analizzare, le attitudini comportamentali del singolo, per migliorare l’esperienza di utilizzo dei prodotti software.

Effettuando diverse ricerche a riguardo ho potuto constatare che la maggior parte degli studi precedentemente effettuati si concentra sull’analisi delle emozioni che vengono categorizzate come primarie dagli studi di Paul Ekman:

- Happiness,
- Anger,
- Sadness,
- Disgust,
- Fear,
- Neutral

Di contro argomenti quali gli stati d'animo (o mood), che influenzano l'engagement degli studenti durante l'apprendimento, sono stati ampiamente trascurati.

Diversi studi hanno dimostrato che gli stati d'animo positivi sono direttamente collegati al pensiero creativo e alla capacità dell'individuo di riflettere su ciò che sta compiendo, indi una maggiore agevolezza e beneficio nell'apprendimento che ne consegue; diversamente gli stati d'animo negativi sono correlati ad una maggiore difficoltà di esercitare queste caratteristiche portando ad un minore rendimento relativamente a questo ambito.

Per quanto le emozioni primarie forniscano sicuramente un indicatore dello stato d'animo, e del conseguente miglioramento dell'esperienza di apprendimento o di lavoro, non sono però generalmente manifestate in modo esplicito, soprattutto per quanto concerne le espressioni facciali, in contesti di ufficio e studio.

Coerentemente si è ritenuto fondamentale concentrare lo studio sulle espressioni facciali e i mood più frequenti all'interno di questi ambienti.

Nei prossimi paragrafi analizzerò degli studi riguardanti sia l'analisi degli stati d'animo e i relativi procedimenti adoperati per effettuarli, che vari studi inerenti al FACS (Facial Action Coding System).

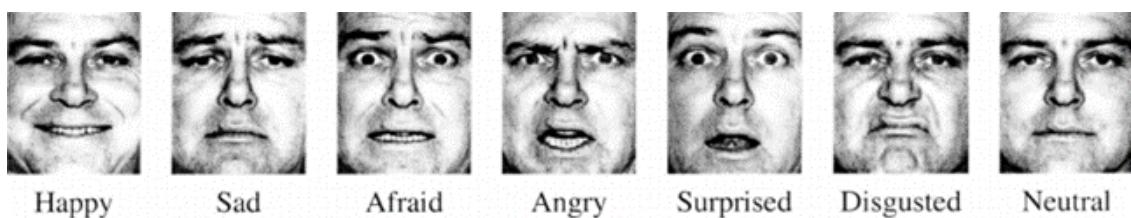


Figura 1: Emozioni categorizzate da Ekman

Capitolo 1

Stato dell'arte

1.1 Studi sull'engagement all'interno degli ambienti di studio

Gli studi ritrovati che effettuano quest'analisi o trattano un argomento simile sono:

1.1.1 Riconoscimento delle emozioni cognitive in ambiente di e-learning

In questo studio ([1]) gli stati di d'animo che vengono classificati dal loro sistema sono i seguenti:

- Entusiasmo,
- Interesse,
- Sorpresa,
- Noia,
- Perplessità,
- Frustrazione,
- Neutrale

Viene riportato che gli stati d'animo positivi (entusiasmo, interesse e sorpresa) sono spesso associati al raggiungimento del “flow state” da parte dello studente.

Più il singolo soggetto mantiene costante un mood positivo, tanto più permarrà in questo flow state, con esito un apprendimento più veloce ed efficace [1].

Il raggiungimento di questo mood è stato inoltre connesso alla capacità dei singoli studenti di percepirci come autosufficienti nel corso dell'attività di studio.

Questa percezione di sé stessi deriva dall'attitudine dello studente nel riuscire ad essere in controllo della sua personale situazione di studio, rispecchiandosi nella sua abilità di:

- pianificare,
- controllare,
- dirigere

l'attività di apprendimento [1].

È quindi necessario che gli studenti maturino una certa dimestichezza nello studio, e che vengano dunque supportati in vista dell'approccio ai problemi che vengono da loro riconosciuti in quanto difficili.

Naturalmente un sistema che permette di “leggere” con precisione lo stato d'animo di uno studente/studentessa o persino di un'intera classe, e quindi capire se questi si trovino nello stato di flow che possa permettere una migliore performance, è uno strumento utile per qualsiasi insegnante.

Un esempio circoscritto all'ambiente di lavoro potrebbe invece essere un'analisi del lavoratore nello svolgimento di un task e la possibilità, da parte di un capo progetto o di un tutor, di poter intervenire esclusivamente nel momento in cui il suo subordinato stia riscontrando dei problemi; in tal modo è permessa al dipendente una crescita professionale adeguata e non seguita al 100%, così da sfruttare al meglio l'impiego del tempo del tutor.

1.1.2 Le faccie dell'Engagement: Riconoscimento automatico dell'engagement degli studenti attraverso le espressioni facciali

In [4], gli stati d'animo organizzati in scala da meno attento/a a più attento/a, sono:

- Not engaged at all (Non coinvolto: che guarda da un'altra parte, che sta ovviamente non pensando al compito, occhi completamente chiusi)
- Nominally engaged (Formalmente coinvolto: occhi appena aperti, chiaramente non attento/a al task che sta svolgendo)
- Engaged in task (Coinvolto nel task: requisito che non richiede un'ammonizione per la progressione del task)
- Very engaged (Altamente coinvolto: lo/a studente/tessa potrebbe essere elogiato/a per il suo livello di coinvolgimento)
- Clip/frame non chiaro (l'immagine analizzata non contiene una persona o comunque non è possibile effettuare un'identificazione)

Lo studio si concentra sull’effettuare una stima dell’engagement degli studenti.

È stato inizialmente sviluppato un metodo per rilevare automaticamente l’engagement, si è poi indagato su quali segnali siano utilizzati nel riconoscimento automatico effettuato dal computer per poi individuare quali strumenti vengano adoperati dagli insegnanti per risolvere il medesimo task.

Infine si è investigato sulla correlazione effettiva fra i risultati di queste analisi e la qualità delle performance degli studenti.

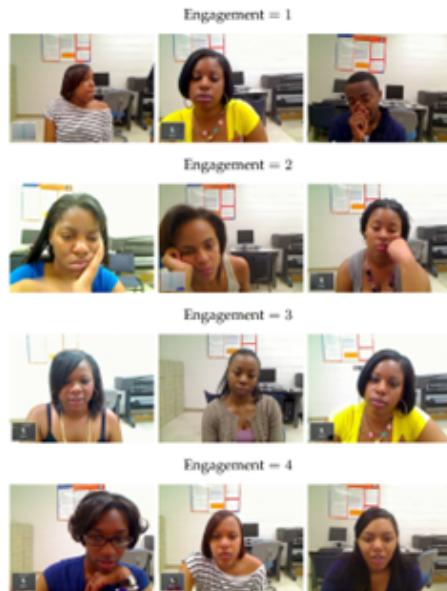


Figura 1.1: Esempio campioni dal dataset dello studio

1.1.3 Codifica facciale come mezzo per il monitoraggio continuo del comportamento degli studenti nell’e-learning

Il paper [6] si focalizza sul tracciamento continuo degli studenti, sia per quanto riguarda una vera e propria identificazione degli stessi attraverso il riconoscimento facciale, sia per calcolarne il livello di attenzione ed eventualmente stimarne le emozioni provate durante i corsi MOOCs (Massive Open Online Courses).

Per dirigere l’analisi del livello di attenzione, si è ricorso all’uso della libreria esterna Dlib, la quale consente di creare una mappatura delle caratteristiche facciali dello/a studente; per giunta, la piattaforma include anche un gaze tracker che lascia prevedere la direzione dello sguardo degli studenti durante lo svolgimento del corso.

Questi tre aspetti vengono successivamente congiunti al fine di creare un applicativo web per l’apprendimento, attraverso il quale, alla fine di ogni lezione, è possibile

visionare in quale percentuale della durata del corso le persone hanno adempito alle metriche citate nell'immagine 1.2.

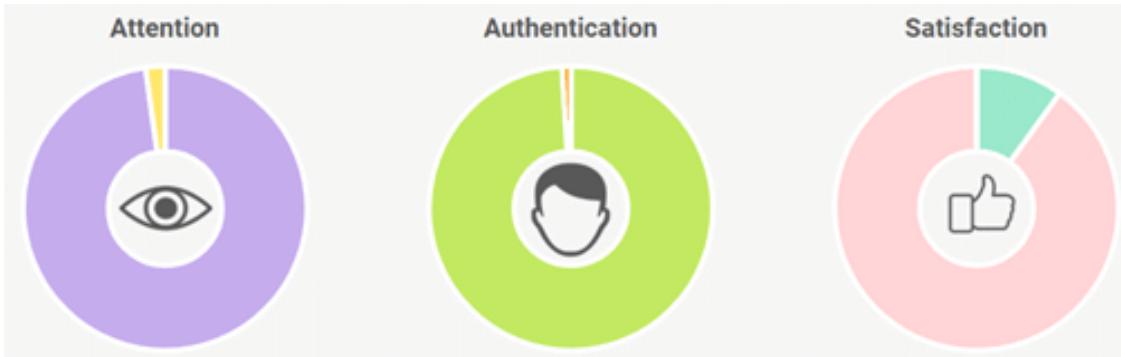


Figura 1.2: Esempio di output del programma

1.1.4 Predizione e localizzazione dell'engagement degli studenti in the wild

Lo studio [7], a differenza di altri, ha come premessa l'utilizzo delle immagini raccolte in ambienti non controllati per la creazione del modello che andrà successivamente ad effettuare la predizione per i nuovi campioni.

Per ambienti controllati, si intende setup di acquisizione dei video e delle immagini grazie ai quali non è possibile riscontrare problemi, quali scarsa illuminazione, occlusione ambientale, etc...

Per attuare ciò, sono stati sottoposti ad analisi molti studi precedentemente effettuati, per convenire al raccoglimento di campioni attraverso la fruizione, da parte dei soggetti, di video educazionali, categorizzando poi i vari video ed immagini ottenute in una scala, con valore da 0 a 3:

- 0 → per niente interessato (il soggetto non sembra interessato e guarda spesso al di fuori dello schermo)
- 1 → poco interesse (il soggetto apre a malapena gli occhi, si muove in modo irrequieto sulla sedia)
- 2 → interessato/a al contenuto (sembra che al soggetto il contenuto riprodotto risulti interessante ed esso interagisce con questo)
- 3 → altamente interessato/a (il soggetto ha “gli occhi attaccati allo schermo” e risulta concentrato/a)

Hanno poi sfruttato un framework che esegue il riconoscimento dell'engagement e della localizzazione degli studenti.

- Inizialmente vengono identificate la faccia e dei punti di riferimento all'interno di queste in ognuno dei frame analizzati
- Procedendo, i video vengono suddivisi in segmenti più piccoli e le feature vengono estratte, “effettuando una media” dei risultati di ognuno dei frame.
- Si passa poi alla sequenza di frame successiva per effettuare la stessa analisi.
- Una volta raccolti tutti i dati, questi vengono elaborati per calcolarne l'engagement e la localizzazione attraverso la deep MIL network, impiegando la media e la top-k pooling per calcolarne la regressione.

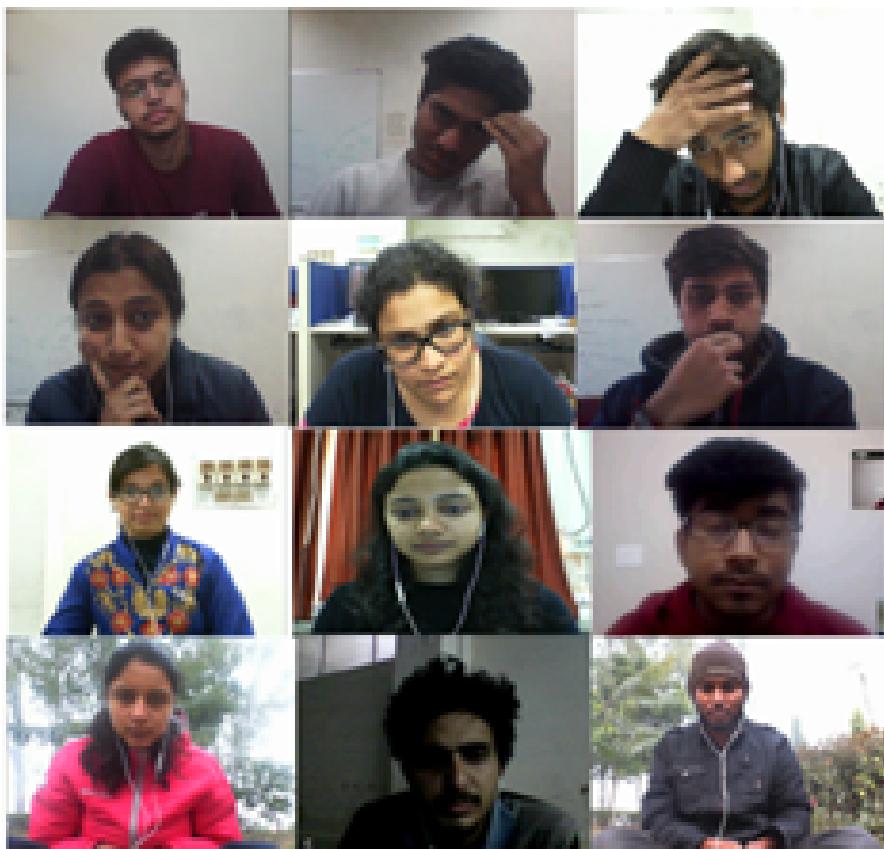


Figura 1.3: Esempio di campioni dal dataset utilizzato in [7]

1.2 Cosa sono le AUs e il sistema FACS:

FACS, acronimo di Facial Action Coding System, è un sistema di codifica delle espressioni facciali sviluppato dallo psicologo Paul Ekman e dal collega Wallace V. Friesen negli anni '70 [8].

Il sistema di codifica di FACS è composto da un insieme di codici numerici, o "Action Units" (AU), che rappresentano le azioni muscolari specifiche che avvengono durante le espressioni facciali. Ci sono 66 AUs, ciascuna delle quali rappresenta un'azione muscolare specifica, e ognuna di queste ha un codice numerico univoco.

La codifica delle FACS può essere utilizzata per identificare e descrivere le espressioni facciali in modo oggettivo e dettagliato. Ad esempio, un codificatore FACS identifica l'AU corrispondente al sollevamento di una delle due sopracciglia con codice 1, e l'AU corrispondente al sorriso con codice 12.

Questi codici possono essere utilizzati per creare una descrizione dettagliata dell'espressione facciale di un individuo.

Le FACS sono state utilizzate in una vasta gamma di contesti, tra cui la ricerca scientifica, la valutazione clinica e la produzione di effetti speciali per il cinema e la televisione.

Questo sistema è stato utilizzato con successo in molti contesti diversi ed è un metodo utile per descrivere le espressioni facciali in modo oggettivo e dettagliato.

Sulla documentazione della libreria che ho utilizzato per l'estrazione delle AUs dal dataset aggregato ed utilizzato, è presente una tabella dove viene mostrato: il numero associato alla singola Action Unit, il loro relativo nome FACS, a quali muscoli della faccia sono correlate, la loro categoria FACS, le espressioni che vengono spesso legate a determinati valori di esse e i modelli che le utilizzano. Ho reso disponibile un estratto di questa 1.1 dove vengono presentate solo le Action Units utilizzate dalla libreria [8] utilizzata per estrarre questi dati dalle immagini ma la tabella completa è consultabile sulla relativa pagina web.

Upper Face Action Units					
AU 1	AU 2	AU 4	AU 5	AU 6	AU 7
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser	Cheek Raiser	Lid Tightener
*AU 41	*AU 42	*AU 43	AU 44	AU 45	AU 46
Lid Droop	Slit	Eyes Closed	Squint	Blink	Wink
Lower Face Action Units					
AU 9	AU 10	AU 11	AU 12	AU 13	AU 14
Nose Wrinkler	Upper Lip Raiser	Nasolabial Deepener	Lip Corner Puller	Cheek Puffer	Dimpler
AU 15	AU 16	AU 17	AU 18	AU 20	AU 22
Lip Corner Depressor	Lower Lip Depressor	Chin Raiser	Lip Puckerer	Lip Stretcher	Lip Funneler
AU 23	AU 24	*AU 25	*AU 26	*AU 27	AU 28
Lip Tightener	Lip Pressor	Lips Part	Jaw Drop	Mouth Stretch	Lip Suck

Figura 1.4: Esempi di Action Units e le relativi sezioni facciali

1.3 Studi sul riconoscimento delle emozioni FACS per scelta del modello da utilizzare

Essendo l'ammontare di studi che trattano l'analisi delle emozioni FACS maggiore rispetto a quelle che cercano di creare sistemi di riconoscimento automatico per gli stati d'animo, i quali possono direttamente aiutare a identificare i problemi nell'apprendimento delle conoscenze, ho ritenuto opportuno studiare e selezionare fra i modelli da questi proposti per l'elaborazione delle informazioni per il mio caso di studio.

Il paper con i migliori risultati che ho trovato è stato [12], in questo studio viene utilizzata una random forest a due livelli:

- il primo livello è utilizzato per determinare le Action Units nelle sequenze di espressioni che vengono analizzate
- il secondo livello invece prende in input le Action Units estratte, ed effettua la classificazione delle espressioni finali

Il modello riesce ad ottenere una media di riconoscimento del 100% per le Action Units e del 96.38% per le espressioni facciali (emozioni) che risulta essere migliore degli studi successivamente riportati.

Prima ancora di arrivare al primo livello della random forest viene utilizzato il modello AAM che identifica i punti di riferimento sulla faccia del soggetto ripreso nel video analizzato (o nella sequenza di immagini analizzate) per poi ricorrere all'utilizzo del Lucas-Kanade optical flow tracker col fine di tracciare i punti di riferimento nei frame successivi.

Viene poi calcolato il vettore di spostamento fra i punti di riferimento trovati sul primo frame, dove il soggetto si presenta con un'espressione neutrale, e i punti di riferimento trovati nel frame finale dove l'espressione facciale è al suo picco espressivo. Questo vettore di spostamento viene utilizzato nel primo layer della random forest per l'estrazione delle Action Units.

Le espressioni facciali sono processi dinamici, conseguenza dell'attività muscolare di varie parti del volto:

- fronte,
- occhi,
- naso,
- bocca,
- mascella,
- contorno

Le caratteristiche dinamiche possono essere rappresentate dalla differenza tra i vettori estratti dai fotogrammi.

Le sei espressioni facciali di base:

- felicità,
- tristezza,
- rabbia,

- disgusto,
- paura,
- sorpresa

possono essere descritte utilizzando le AUs, codificate in base ai coinvolgimenti muscolari facciali.

Il processo di estrazione delle caratteristiche di movimento di un'espressione facciale è illustrato nella figura 1.5.

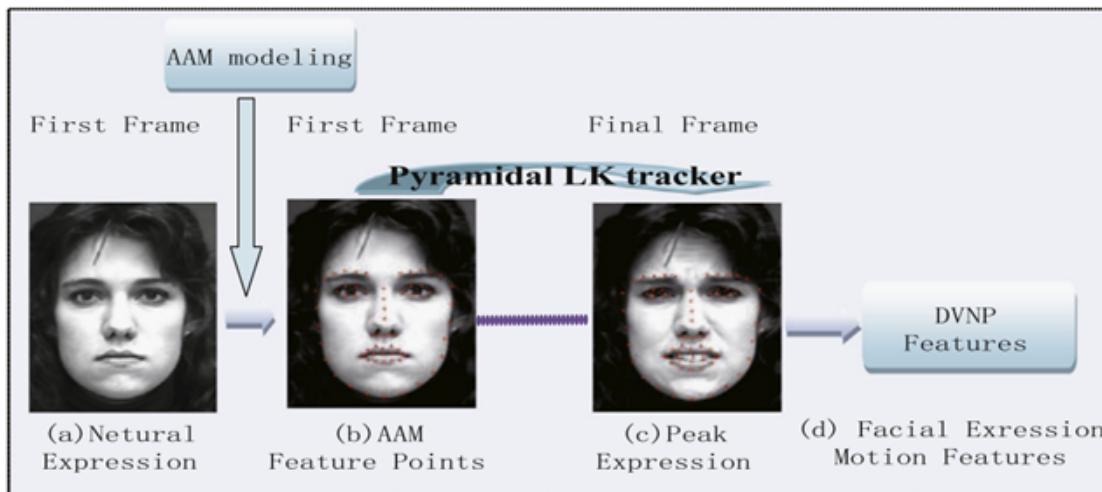


Figura 1.5: Processo di estrazione dinamica delle feature delle espressioni facciali

Random Forest è un algoritmo di machine learning che combina degli alberi decisionali, all'interno dei quali ognuno di questi dipende dai valori di un vettore casuale campionato in modo indipendente e con la medesima distribuzione.

Questo algoritmo ha la capacità di gestire grandi spazi di caratteristiche grazie a due fonti di casualità, ovvero input e feature casuali, che lo rendono robusto.

Random forest è da considerarsi affidabile per l'estimazione della posa, la rilevazione di oggetti e altre aree della computer vision grazie alla sua bassa complessità di calcolo, agevole implementazione, accuratezza nella classificazione e capacità di gestire grandi set di dati di addestramento.

L'algoritmo è costituito da un gran numero di alberi decisionali, dove ciascuno viene costruito ricorsivamente assegnando un test binario ad ogni nodo, non foglia, in base ai samples di formazione.

Per la classificazione combina i risultati degli alberi decisionali per fornire come risultato la classe più popolare.

Questo algoritmo ha la capacità di gestire errori di bilanciamento nella popolazione delle classi in dati sbilanciati.

Per validare i risultati, i ricercatori hanno condotto degli esperimenti sui datasets Cohn-Kanade [34] e Oulu-CAISA VIS [35] servendosi solo delle sequenze di immagini contenenti una vista frontale o girata a 30 gradi dei soggetti. Nel primo dataset sono categorizzate 7 espressioni (anger, contempt, disgust, fear, happy, sadness and surprise), con soggetti di età compresa fra i 18 e i 50 anni.

Il secondo dataset contiene samples ripresi in diverse condizioni di luce; nonostante ciò, sono state incluse solo le immagini con condizioni di illuminazione favorevoli (luce forte o buona illuminazione).

In entrambi i dataset, i video (e/o sequenze di immagini) riprendono delle persone che, in principio, mostrano un'espressione neutrale e, al termine, le espressioni che più esaltano l'emozione relativa al tag del video.

Nella fase finale del metodo proposto viene utilizzato l'ultimo layer della random forest per identificare l'etichetta finale dell'espressione facciale in base ai risultati della rilevazione delle AUs.

Tale metodo viene messo a confronto con una rete bayesiana; questo modello predittivo ottiene un tasso di riconoscimento delle caratteristiche facciali dell'86,3%.

Il metodo proposto dal paper, basato sulle AUs, può invece incrementare il tasso di riconoscimento medio all'89,37%.

I risultati riportati nella tabella dell'immagine 1.6 raffigurano il tasso di riconoscimento per la rilevazione dell'espressione facciale basata sulle AUs utilizzando il random forest.

	An	Co	Di	Fe	Ha	Sa	Su
An	92.86	3.57	3.57	0	0	0	0
Co	0	90.91	0	0	9.09	0	0
Di	2.70	0	97.30	0	0	0	0
Fe	0	0	0	93.75	6.25	0	0
Ha	0	0	0	0	100	0	0
Sa	0	0	0	0	0	100	0
Su	0	1.89	0	0	0	0	98.11
avg				97.10			

Figura 1.6: Matrice di confusione del riconoscimento delle espressioni basato su Action Units usando Random Forest

Confrontando i risultati della Tabella riportata nell'immagine 1.7, dove sono raffigurati i tassi di riconoscimento della rete bayesiana, le prestazioni migliorano significativamente.

Il classificatore con rete bayesiana ha un tasso di riconoscimento basso per le emozioni di tristezza e paura (rispettivamente 68,75% e 88,89%), mentre il random forest migliora queste prestazioni al 93,75% e al 100%.

Una possibile spiegazione è che la tristezza e la paura sono di solito poco evidenti e possono essere facilmente confuse con altre emozioni; tuttavia, il random forest ha maggiore capacità discriminativa e può trovare la differenza fra queste.

	An	Co	Di	Fe	Ha	Sa	Su
An	75.0	0	7.14	0	17.86	0	0
Co	0	90.91	0	0	9.09	0	0
Di	5.41	0	94.59	0	0	0	0
Fe	31.25	0	0	68.75	0	0	0
Ha	2.27	0	0	0	97.73	0	0
Sa	11.11	0	0	0	0	88.89	0
Su	1.89	1.89	0	1.89	1.89	0	92.45
avg				89.37			

Figura 1.7: Matrice di confusione del riconoscimento delle espressioni basato su Action Units usando Rete Bayesiana

Per giunta, i tassi di successo di emozioni come la rabbia, la sorpresa e il disgusto sono stati significativamente migliorati sfruttando il framework proposto.

Per confermare l’efficacia del metodo, sono stati selezionati casualmente un set di training ed un set di testing dal dataset, per poi ripetere l’esperimento nove volte. Tutti i risultati sono riportati qui di seguito nell’immagine 1.8.

Testing #	1	2	3	4	5	6	7	8	9
	95.91	97.72	96.80	95.43	97.25	95.87	95.39	97.22	95.83
avg	96.38								

Figura 1.8: Tasso di riconoscimento in percentuale per le nove selezioni randomiche del training e test set

È stato dimostrato che il sistema ha una prestazione stabile utilizzando set di training e testing diversi. Il tasso di riconoscimento medio è significativamente più alto del metodo di confronto [15].

Un altro studio utilizzato per la predizione delle emozioni attraverso l’analisi delle Action Units è [13]. In questo studio viene proposto un metodo per mappare le AUs a sei emozioni:

- anger,
- fear,
- happy,
- sad,
- surprise,

- disgust

Vengono selezionate delle Action Units, successivamente mappate alle emozioni attraverso relazioni statistiche e tecniche di matching.

Le relazioni fra le emozioni e le AUs sono collezionate come stringhe template che comprendono quelle più descrittive per ognuna delle emozioni trattate.

Le stringhe di template vengono poi computate usando un concetto chiamato discriminative power:

l'LCS, un metodo per approssimare il grado di coincidenza delle sottostringhe alla stringa di template, con l'applicazione del calcolo della prossimità fra le stringhe di test e la stringa di template della singola emozione.

Lo studio ha trovato che LCS è un metodo efficiente nella gestione di particolari problemi come il rilevamento errato delle AUs e aiuta a ridurre le predizioni false.

Il metodo proposto è stato testato su vari datasets:

- CK+,
- ISL,
- FACS,
- JAFFE,
- Mind Reading,
- altri frame ripresi in condizioni in-the-wild

è stata poi effettuata una comparazione fra il metodo proposto e altri precedentemente presentati e si sono notati dei miglioramenti, sia sui datasets di benchmark che per quelli in-the-wild.

La procedura proposta basa il rilevamento delle AUs sul metodo offerto da [16] e a seguire le Action Units vengono mappate con il metodo indicato nel paper, ed è visibile nell'immagine 1.9.

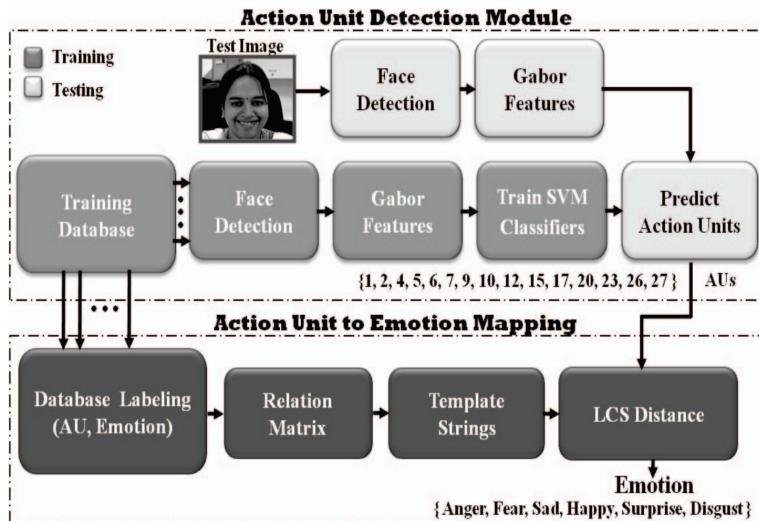


Figura 1.9: Metodo di mappatura delle Action Units di [16]

Per l'allenamento del modello volto al riconoscimento delle AUs sono state utilizzate 580 immagini dal CK+ dataset, e, per ognuna di queste, sono state effettuate varie modifiche alle immagini in modo da renderle adatte all'impiego; è stato applicato l'AdaBoost per la selezione delle feature al fine di ridurre la dimensione del vettore delle features, ed è stato utilizzato il Support Vector Machines (SVM) per il modello delle AUs.

Dopo un'attenta analisi del sistema FACS sono state scelte dai ricercatori quindici AUs, sufficienti per la rappresentazione delle sei emozioni che si sono proposti di identificare.

Le AUs rilevate vengono poi processate attraverso il modulo di mappatura per predire lo stato d'animo combaciante.

L'idea sulla quale il modello sviluppato in questo paper si fonda è che diversi metodi utilizzati per la rilevazione delle emozioni si basano su input (AUs) inaffidabili, in quanto sono presenti inevitabili errori nella localizzazione dei volti all'interno delle immagini.

È per questo che il loro metodo si concentra maggiormente sul trovare la correlazione fra le singole AUs e le emozioni che sulla diretta predizione di queste.

La relazione fra le Action Units e le sei emozioni si ottiene mediante un'analisi statistica del dataset di riferimento (CK+), etichettato sia per le emozioni che per le AUs. Le relazioni vengono acquisite sotto forma di una matrice relazionale derivata utilizzando un concetto chiamato discriminative power [17].

Il discriminative power è definito come:

$$H = P(Y_j|X_i) - P(Y_j|\overline{H_i})$$

dove $P(Y_j|X_i)$ è la probabilità dell'azione Y_j , dopo che X_i è avvenuta, e $P(Y_j|\overline{H_i})$ è la probabilità dell'azione Y_j , dopo che l'emozione H_i non è avvenuta. La dimensione di H rappresenta il potere discriminante di un'AU rispetto ad un'emozione.

La matrice relazionale viene ottenuta normalizzando H su tutte le AU per ciascuna delle emozioni. Pertanto, si ottengono pesi di associazione non lineari per ciascuna delle Action Unit, in base alla loro rilevanza per le emozioni calcolate servendosi dell'equazione sopra riportata.

La figura 1.10 riporta la matrice di relazione calcolata per le sei emozioni. Qui, i valori positivi, rappresentati in bianco, indicano l'alta probabilità per un'AU di appartenere ad un'emozione; d'altra parte, i valori negativi, rappresentati in nero, indicano l'alta probabilità per un'AU di non essere associata ad un'emozione.

Ad esempio, l'emozione happy è associata positivamente alle AU6, AU7, AU12, AU26 e negativamente alle AU1, AU2, AU5, AU9.

I risultati sperimentali presentati dallo studio dimostrano che la matrice di relazione derivata è efficiente nell'identificazione delle azioni facciali altamente rilevanti e i loro pesi di associazione per le varie emozioni, e può quindi prevedere correttamente le emozioni.

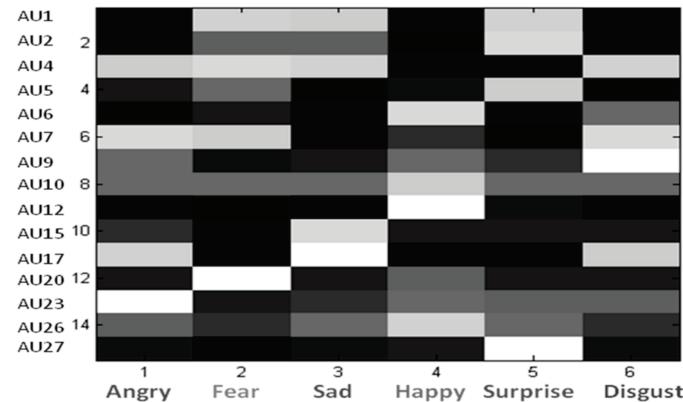


Figura 1.10: Matrice di relazione calcolata sulle sei emozioni

Per ciascuna emozione sono state selezionate le prime N entrate da AU altamente discriminanti e queste sono state salvate come stringhe di template per i futuri abbinamenti delle Action Units.

La lunghezza delle stringhe di template utilizzata è di cinque per gli esperimenti condotti dai ricercatori dello studio.

Emotions	Action Units				
Angry	23	7	17	4	2
Fear	20	4	1	5	7
Sad	15	1	4	17	10
Happy	12	6	26	10	23
Surprise	27	2	1	5	26
Disgust	9	7	4	17	6

Figura 1.11: Esempio di stringa template

Date le stringhe con le Action Units per le immagini analizzate, queste vengono confrontate con le stringhe di template per trovare l'emozione corrispondente.

Si utilizza la sotto sequenza comune più lunga (LCS) [18] per misurare la similarità tra le stringhe. LCS è un metodo per trovare la sotto sequenza più lunga fra tutte le sequenze date.

In questo caso, una sotto sequenza è definita come una sequenza in cui le unità appaiono nello stesso ordine una rispetto all'altra, ma non necessariamente in modo contiguo.

Ad esempio, nella stringa "ACTTGCG", "ACT", "ATTC" e "ACTTGC" sono tutte sotto sequenze.

LCS permette "inserzioni", "eliminazioni" e "sostituzioni" delle singole AUs tra le stringhe. Questa caratteristica di LCS si è rivelata idonea per la predizione delle emozioni da stringhe di Action Units.

Per fare un esempio, data un'immagine di test, possono essere predette le combinazioni di AUs riportate nell'immagine 1.12.

In questo caso, la proprietà di eliminazione diventa importante per la mappatura delle diverse stringhe di Action Units come AU12 o AU6, AU12 o AU6, AU12, AU26 all'emozione happy, poiché tutte indicano l'emozione di felicità pur mancando, alcune delle AU, nelle stringhe.

Allo stesso modo, l'inserzione ha un ruolo nell'eliminazione delle AU rilevate erroneamente, come AU1, AU4, che comunque mappano l'emozione nella figura alla stringa di template che rappresenta l'emozione happy.

Inoltre, permettere le sostituzioni aiuta a correggere gli errori di rilevazione che coinvolgono AU visivamente simili come AU12, AU20, AU23, e a predire l'emozione con un alto valore di confidenza.

Viene evitata anche la mappatura errata di molte emozioni quando la stringa di test presenta errori considerevoli.

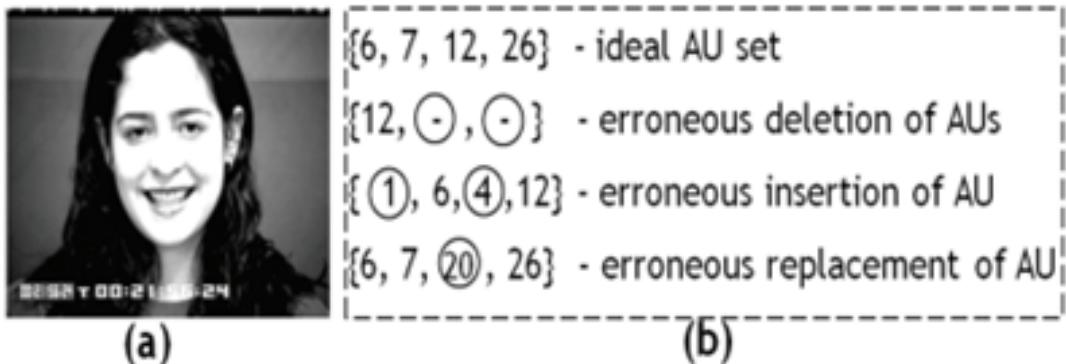


Figura 1.12: Esempio di rilevamento delle Action Units

Ad esempio, in sostituzioni non condizionali, una stringa di test come $\{AU4, AU6, AU12, AU17, AU23\}$, può essere mappata a $\{AU1, AU4, AU10, AU15, AU17\}$ (tristezza) e $\{AU4, AU6, AU7, AU9, AU17\}$ (disgusto) con un costo di sostituzione di "3".

I risultati e l'accuratezza ottenuta dal metodo proposto dallo studio sui vari dataset utilizzati per valutarlo sono esposti nella tabella all'immagine 1.13:

*	Correct Detection(%)		
Database	Valstar	Moon	Proposed
<i>CK+ (125)</i>	72.0	80.5	97.0
<i>FACS (136)</i>	75.0	79.0	94.0
<i>ISL (100)</i>	58.0	63.5	77.5
<i>JAFFE (213)</i>	70.5	76.5	87.5
<i>MindReading (200)</i>	62.0	64.0	82.0
Average	67.5	72.7	87.6

Figura 1.13: Accuracy del metodo

Analizzando altri papers sul soggetto, nessuno di questi mi è apparso particolarmente rilevante, in quanto carenti di metodologie che aggiungevano informazioni e metodi utili da unire ai lavori già ritrovati; inoltre i risultati di accuratezza e precisione ottenuti da questi sono inferiori rispetto agli studi già analizzati o insufficientemente migliori e necessitano di più risorse (tempo, quantità di dati e potere computazionale).

Ad esempio, in [14] vengono confrontate le due principali tecniche di previsione delle emozioni facciali, ovvero le tecniche di apprendimento automatico (ML) e di deep learning (DL), in termini di accuratezza e livello di accountability.

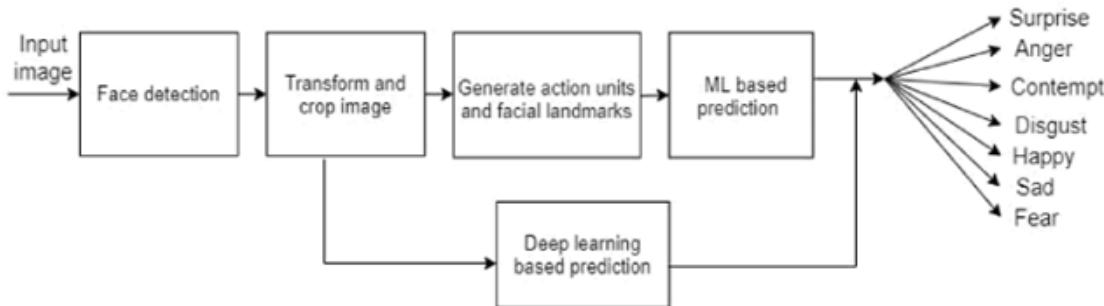


Figura 1.14: Entrambi i process flow del riconoscimento delle emozioni facciali in [14]

Analizzando le due tecniche, i ricercatori hanno dimostrato che i modelli ML possono raggiungere risultati comparabili rispetto ai modelli DL.

Nonostante la maggior parte delle ricerche recenti si siano focalizzate sulla costruzione di tecniche di Deep Learning per la predizione delle emozioni, si è scoperto che i modelli DL raggiungono prestazioni migliori a fronte di un costo elevato in termini di potenza di calcolo, di set di dati più grandi e di tempi di elaborazione più lunghi.

Negli esperimenti sono stati conseguiti livelli di accuratezza del 86,66% e dell'80% rispettivamente per gli approcci DL e ML in termini di previsione delle emozioni; nonostante ciò, le tecniche di DL necessitano del supporto di una GPU ad alte prestazioni, mentre le tecniche ML possono ricoprire il proprio ruolo più velocemente anche senza il supporto di una GPU.

Il beneficio più importante nell'utilizzo delle tecniche ML con le Action Units è la capacità di questi ultimi di poter giustificare l'emozione prevista in termini di contributo di ciascuna AU.

1.4 Studi che trattano campioni prelevati in scenari non controllati per una migliore precisione del modello

Un altro problema trattato solo da [7] (fra i paper ritrovati a tema engagement e stati d'animo) e che invece ho notato più prominente all'interno della ricerca riguardo le emozioni FACS, è quello dei campioni denominati come “in-the-wild”.

Molti degli studi effettuati si concentrano su campioni prodotti in ambienti controllati, con nessun tipo di interferenza per quanto concerne i problemi che abitualmente si possono riscontrare quando invece si lavora con riprese che potrebbero avere una valenza anche quando le migliori condizioni non si presentano.

Come riporta [3]: “The videos are recorded under controlled conditions, e.g. illumination is uniform, background is static, and there is a limited amount of head pose variation and occlusion. Although a range of affective states are displayed and recorded, emotions are elicited by a limited number of tasks, e.g., in [36], all subjects underwent exactly the same tasks.”

Quindi i video creati in condizioni verificate, di luce uniforme, sfondo statico, con insufficienti variazioni nella posa delle persone, poca occlusione e dove le persone svolgono lo stesso compito non sono validi per creare un modello che possa effettuare delle predizioni veritieri da applicare ad ogni ambiente.

Per quanto riguarda la raccolta di immagini, negli studi “in the wild” è stato messo in atto il prelievo e la categorizzazione delle immagini da film, serie tv, e altri media che non rispettano gli standard imposti durante la creazione dei dataset prelevati invece in ambienti controllati [3].

Per l'annotazione delle immagini sono stati consultati due annotatori esperti FACS AU che hanno individuato lo stato d'animo della/e persone presenti all'interno della scena presentata attraverso una piattaforma di tagging delle immagini.

Entrambi gli annotatori erano sempre presenti durante l'analisi dei singoli campioni ed hanno quindi potuto discutere ognuno dei casi incerti insieme; una volta concluse le prime analisi, sono stati inoltre ripresentati alcuni dei frame già analizzati agli stessi annotatori in modo da effettuare un'ulteriore verifica della loro consistenza, ed essi hanno dato la stessa annotazione l'87% delle volte.

Per costruire il modello hanno poi utilizzato l'estrazione delle AU attraverso un sistema semi automatico, ed hanno utilizzato questi dati estratti per la costruzione del modello.

I risultati di queste analisi hanno portato alla conclusione che i modelli sviluppati attraverso raccolte di immagini prelevate in scenari controllati non portano necessariamente a risultati puntuali (o accurati) quando vengono applicati al mondo reale. Una volta analizzati i sistemi utilizzati per l'elaborazione dei dati raccolti dai ricercatori che hanno redatto gli studi riportati precedentemente, ritengo ora necessario illustrare gli strumenti di cui si sono avvalsi.

1.5 Metodologie di tagging delle immagini del dataset

Per quanto riguarda la classificazione delle immagini dei dataset, vari studi hanno fatto ricorso a tecniche diverse:

- Self report sul proprio stato emotivo da parte delle persone da cui vengono prelevati i campioni per le analisi, attraverso questionari o in modo arbitrario
- Valutazione da parte di esperti, spesso più di uno, per assegnare uno stato emotivo alle singole immagini o video; o scelto in modo interamente arbitrario da parte del/dei valutatore/i o utilizzando come riferimento lo stato emotivo riportato dalla persona
- Classificazione attraverso metodi di clustering [9]

Strumenti utilizzati per effettuare la classificazione:

- Questionari a cui sottoporre i soggetti che vengono ripresi per la creazione del dataset
- In [11] è stata adoperata una tecnica di crowdsourcing: mediante la piattaforma Amazon Mechanical Turk (o MTurk) i ricercatori hanno specificato il task che si erano proposti di risolvere e hanno fornito i frame da analizzare. Questi sono stati poi suddivisi dalla piattaforma e categorizzati dai singoli individui (tale servizio è ovviamente retribuito)



1.6 Estrazione delle feature facciali dalle immagini e dai video dei dataset

Per quanto concerne l'estrazione dei dati dalle immagini, i paper analizzati hanno adoperato diverse librerie e strumenti fra cui:

- OpenCV: libreria open-source di computer vision di cui ci si può avvalere per il riconoscimento delle espressioni facciali. OpenCV contiene diverse funzioni per l'estrazione delle AUs, come la rilevazione delle linee facciali e la stima dei parametri di deformazione.
- dlib: libreria di machine learning open-source che somministra funzioni per la rilevazione e l'analisi dei volti nelle immagini. Dlib è in grado di rilevare le AUs attraverso l'utilizzo di una rete neurale convoluzionale (CNN) addestrata su un grande dataset di espressioni facciali.

- FaceReader: software commerciale sviluppato dalla società olandese Noldus Information Technology che ricorre ad algoritmi di analisi dell'emozione per registrare e classificare le AUs nelle immagini facciali. FaceReader è capace di riscontrare fino a 20 AUs e di classificarle in base alle emozioni ad essi associate.
- OpenFace: framework open-source di computer vision sviluppato dall'Università di Carnegie Mellon che offre funzioni per l'estrazione delle AUs e la rilevazione delle espressioni facciali. OpenFace sfrutta una combinazione di tecniche di machine learning e di analisi geometrica per rilevare le AUs.
- FACET: software commerciale sviluppato dalla società americana Emotient che impiega una combinazione di tecniche di computer vision e di analisi dell'emozione per rilevare e classificare le AUs nelle immagini facciali. FACET è in grado di rilevare fino a 20 AUs e di classificarle in base alle emozioni associate.
- DeepFaceLab: software open-source che utilizza le deep neural networks per l'analisi dell'immagine e la manipolazione dei lineamenti. Può essere strumentalizzato per l'estrazione delle AUs da immagini facciali.
- Per il progetto di tesi ho invece deciso di utilizzare: Pyfeat: libreria open-source sviluppata in Python per l'estrazione delle feature facciali dalle immagini. Pyfeat si basa sulla fruizione di un insieme di funzioni matematiche, dette funzioni di base, per descrivere la forma e l'aspetto delle suddette espressioni

Dopo aver analizzato i sistemi impiegati per l'elaborazione delle informazioni ottenute dalle immagini ed aver illustrato il processo di estrappolazione di questi dati, reputo possibile la creazione di un dataset ricavato dall'unione di alcuni dei datasets citati precedentemente per lo studio di questa tesi.



Figura 1.15: Logo py-feat

1.7 Unione dei dataset ritrovati e relativa categorizzazione delle immagini all'interno di questo

I dataset utili ritrovati sono il DAiSEE [11] e il Student-engagement-dataset [10].

- DAiSEE presenta i seguenti mood:
 - Boredom
 - Engagement
 - Confusion
 - Frustration
- Tutte con valore compreso fra 0 e 3:
 - Inizialmente è stato chiesto ai labelers di annotare i frame a loro presentati con uno di questi tre tag:
 - * Looking at their paper
 - * Looking at their screen
 - * Wandering
- Student engagement dataset, invece, presenta i seguenti mood:
 - Confused (macro categoria: engaged)
 - Engaged (macro categoria: engaged)
 - Frustrated (macro categoria: engaged)
 - Bored (macro categoria: not engaged)
 - Drowsy (macro categoria: not engaged)
 - Looking away (macro categoria: not engaged)

Per la realizzazione del mio dataset ho ritenuto giusto attenermi alle classi proposte dallo Student Engagement dataset, in quanto propone più labels di cui usufruire.

Per mettere in atto ciò, è stato necessario unire i samples all'interno di DAiSEE ai samples dello student engagement dataset, per mezzo delle classi corrispondenti.

DAiSEE è provvisto di un maggior numero di samples, in quanto non solo racchiude molti più file, ma questi stessi sono video e non immagini, come nello student engagement dataset, naturalmente suddivisi in frame (2 immagini per ogni secondo di video).

Le classi che non trovano corrispondenza fra i due dataset (Drowsy e Looking away) potrebbero risultare meno frequenti e accurate nel riconoscimento; ho comunque preferito conservarle in quanto permettono una classificazione più ampia.

DAiSEE è un dataset realizzato tenendo in considerazione la questione “in-the-wild” e risulta quindi perfetto per lo studio proposto da questa tesi.

Lo student engagement dataset non è stato creato da una fonte autorevole e certificata come il DAiSEE ma, sfogliando le immagini presenti in esso, è possibile notare come gli scenari ripresi in questo dataset (ambienti di studio casalinghi) le immagini al suo interno risultino simili a quelle presentate nel DAiSEE e quindi della tipologia interessata.

Tabella 1.1: estratto tabella AUs

AU1	Inner Brow Raiser	Frontalis (medial)	sadness, surprise, fear
AU2	Outer Brow Raiser	Frontalis (lateral)	surprise, fear
AU4	Brow Lowerer	Procerus, Depressor Supercilii, Corrugator Supercilii	sadness, fear, anger
AU5	Upper Lid Raiser	Levator Palpebrae Superioris, Superior Tarsal Muscle	surprise, fear, anger
AU6	Cheek Raiser	Orbicularis Oculi (orbital)	happiness, disgust, contempt
AU7	Lid Tightener	Orbicularis Oculi (palpebral)	fear, anger
AU9	Nose Wrinkler	Levator Labii Superioris Alaeque Nasi	disgust
AU10	Upper Lip Raiser	Levator Labii Superioris	disgust, fear
AU11	Nasolabial Deepener	Zygomaticus Minor	happiness, contempt
AU12	Lip Corner Puller	Zygomaticus Major	contempt
AU14	Dimpler	Buccinator	sadness, disgust
AU15	Lip Corner Depressor	Depressor Anguli Oris	disgust
AU17	Chin Raiser	Mentalis	fear
AU20	Lip Stretcher	Risorius, Platysma	anger
AU23	Lip Tightener	Orbicularis Oris	
AU24	Lip Pressor	Orbicularis Oris	
AU25	Lip Part	Depressor Labii Inferioris	happiness, surprise, fear
AU26	Jaw Drop	Masseter, Temporalis, Medial Pterygoid	fear, surprise
AU28	Lip Suck	Orbicularis Oris	
AU43	Eyes Closed	Levator Palpebrae Superioris (relaxation)	behavioral

Capitolo 2

Tecnologie, strumenti e librerie utilizzate

Questo capitolo tratterà le tecnologie, gli strumenti, le librerie e gli algoritmi utilizzati per la realizzazione dei modelli predittivi e dell'interfaccia successivamente creata.

2.1 Tecnologie

2.1.1 Machine Learning

Il Machine Learning, o apprendimento automatico, è un campo di studio che si occupa di sviluppare algoritmi in grado di perfezionarsi automaticamente mediante l'esperienza acquisita tramite l'utilizzo dei dati.

Gli algoritmi di Machine Learning creano modelli matematici a partire da dati di esempio chiamati "dati di training", in modo da poter attuare predizioni o prendere decisioni senza essere esplicitamente programmati per farlo.

Esistono tre categorie di approcci di Machine Learning: l'apprendimento supervisionato, l'apprendimento non supervisionato e l'apprendimento per rinforzo.

- L'apprendimento supervisionato consiste nell'indicare al calcolatore una regola generale che mappi gli input e gli output desiderati.

L'algoritmo di apprendimento è provvisto di dati di input di esempio e degli output corrispondenti e, attraverso successive iterazioni, viene successivamente costruito un modello matematico che può essere sfruttato per predire l'output associato ad un nuovo input.

- L'apprendimento non supervisionato invece, non fornisce all'algoritmo di apprendimento le etichette desiderate.

In questo caso, l'algoritmo di apprendimento deve necessariamente estrapolare le informazioni significative dai dati di input pur non essendo a conoscenza dell'output desiderato.

- L'apprendimento per rinforzo prevede che un programma interagisca con un ambiente dinamico con lo scopo di raggiungere un obiettivo specifico, ad esempio guidare un veicolo o vincere un gioco contro un avversario.

Durante le varie iterazioni il programma riceve un feedback sotto forma di premio e cerca di massimizzare il suo “punteggio”, in modo da imparare e raggiungere l'obiettivo prefissato.

2.1.2 Computer Vision

La Computer Vision è un campo interdisciplinare che si occupa della capacità dei computer di acquisire conoscenza da immagini o video, cercando di replicare il complesso meccanismo alla base dell'apparato visivo umano.

Questa utilizza metodi per l'acquisizione e l'analisi di immagini digitali in modo da estrarre dati multidimensionali dal mondo reale e produrre informazioni numeriche o simboliche.

Si avvale pertanto anche di conoscenze di geometria, fisica, statistica e della teoria dell'apprendimento per descrivere il mondo in modo sensato, producendo risultati che possono indirizzarci alla corretta linea d'azione.

2.1.3 Cuda

CUDA® [19] è una piattaforma di calcolo parallelo, un modello di programmazione sviluppato da NVIDIA per il calcolo generale su unità di elaborazione grafica (GPU).

Per mezzo di CUDA, gli sviluppatori possono ampliare significativamente la velocità delle applicazioni di calcolo sfruttando la potenza delle GPU.

Nelle applicazioni dotate di accelerazione GPU, la parte sequenziale del carico di lavoro viene eseguita sulla CPU, ottimizzata per le prestazioni single-threaded, mentre la parte computazionalmente intensiva dell'applicazione viene realizzata su migliaia di core GPU in parallelo.

Quando ci si avvale di CUDA, gli sviluppatori hanno la possibilità di programmare in linguaggi popolari come C, C++, Fortran, Python e MATLAB, esprimendo il parallelismo attraverso estensioni sotto forma di poche parole chiave di base. Il toolkit CUDA di NVIDIA fornisce tutto il necessario per sviluppare applicazioni con accelerazione GPU.

Il toolkit CUDA include librerie accelerate su GPU, un compilatore, strumenti di sviluppo e il runtime CUDA.



Figura 2.1: CUDA

2.2 Strumenti

2.2.1 Visual Studio Code

Visual Studio Code è un editor di testo sviluppato da Microsoft per Windows, Linux e macOS, che supporta il debugging, il controllo Git integrato, la Syntax Highlighting, l'IntelliSense, lo Snippet e il refactoring del codice.

Visual Studio Code supporta molteplici linguaggi e funzionalità aggiuntive grazie alla possibilità di installare dei plugin disponibili attraverso un repository centrale che presenta, oltre a diverse estensioni fornite direttamente da Microsoft, innumerevoli estensioni rese disponibili dalla community.

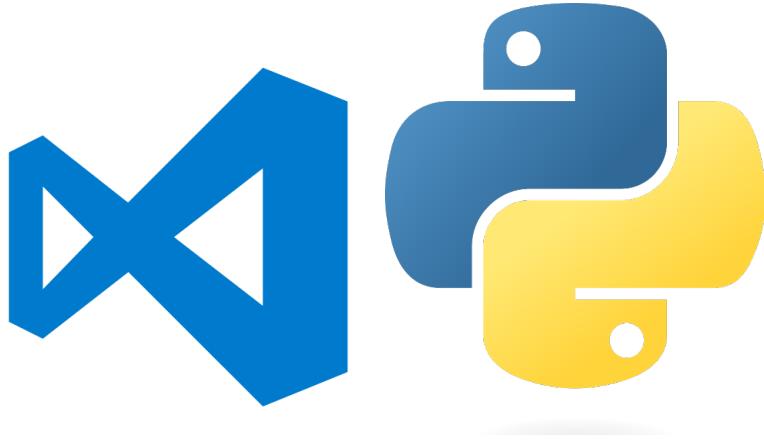


Figura 2.2: Visual studio code e python

2.3 Librerie

2.3.1 Pandas

La libreria software open source Pandas è stata sviluppata per il linguaggio di programmazione Python, ed è utilizzata per la manipolazione e l'analisi dei dati. Con Pandas è possibile effettuare operazioni su tabelle numeriche e serie temporali mediante le sue strutture dati.

Il termine "Pandas" deriva dal termine econometrico "Panel Data", appellativo attribuito a un insieme di dati contenenti osservazioni sugli stessi individui durante più periodi di tempo.



Figura 2.3: Pandas

2.3.2 Tqdm

La libreria Python tqdm [20] è uno strumento molto utile per la visualizzazione di barre di avanzamento durante i cicli di elaborazione nel codice.

Il nome "tqdm" deriva dall'unione della parola araba "taqaddum" che significa "progresso" ed è l'abbreviazione di "te quiero demasiado" (ti amo troppo) in spagnolo.

Per servirsi della libreria basta inserire qualsiasi iterabile (liste, dizionari, tuple e set) nel metodo della libreria, in questo modo `tqdm(iterable)`.

La libreria funziona su qualsiasi piattaforma ed è completamente indipendente dalle dipendenze.

Durante l'estrazione delle Action Units la libreria tqdm ha fornito una stima precisa del tempo necessario per completare l'elaborazione, consentendo di monitorare l'avanzamento del processo in tempo reale.

Nonostante l'utilizzo della tecnologia CUDA per accelerare le analisi, la grande quantità di immagini da elaborare ha richiesto molto tempo.

La presenza di tqdm è stata quindi di fondamentale importanza, in quanto ha permesso di gestire efficacemente l'elaborazione dei dati, evitando eventuali problemi tecnici, garantendo risultati accurati ed affidabili e portandomi a riconsiderare delle scelte algoritmiche, non efficientissime, inizialmente intraprese.

In sintesi, la libreria Python tqdm è uno strumento prezioso per semplificare l'elaborazione di grandi quantità di dati, fornendo una stima del tempo rimanente e consentendo di pianificare il lavoro in modo efficiente.

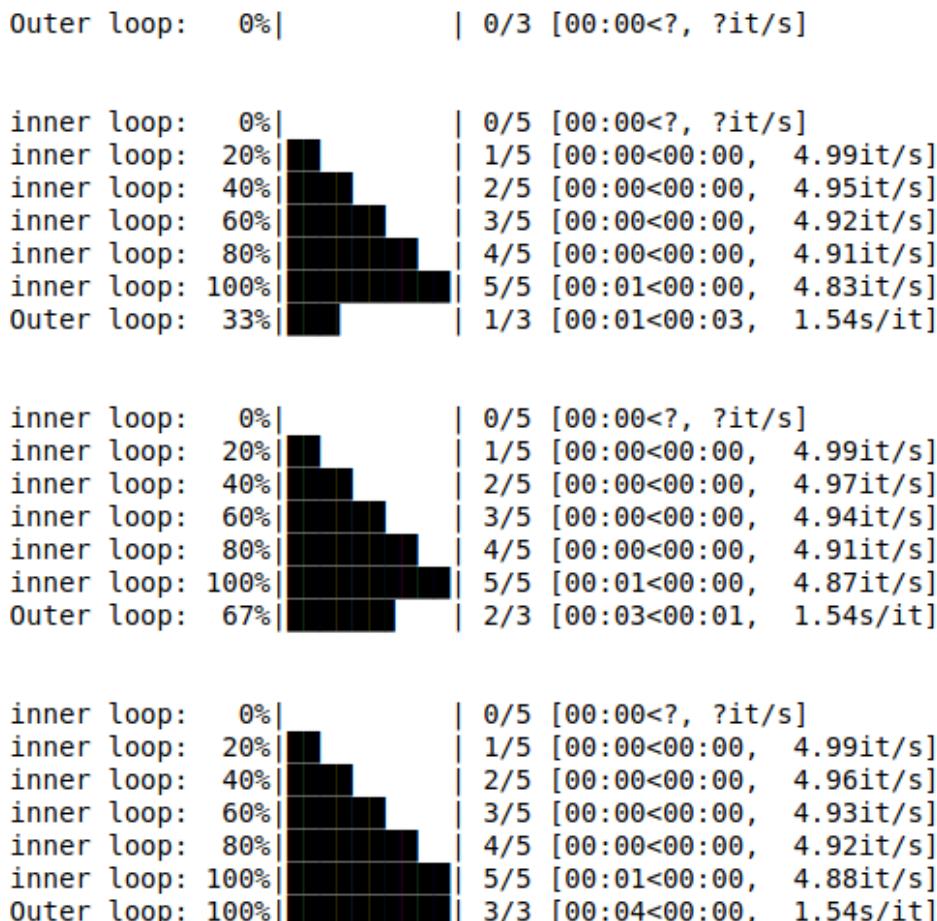


Figura 2.4: Esempio di utilizzo di tqdm

2.3.3 Pickle

La libreria pickle è molto versatile e può essere utilizzata per salvare e ripristinare qualsiasi tipo di oggetto Python, inclusi dizionari, liste, tuple, classi e istanze di oggetti personalizzati.

Inoltre, pickle supporta anche la serializzazione di oggetti multipli in un unico file, rendendo più semplice l'organizzazione dei dati.

La libreria offre anche diverse opzioni per controllare il comportamento della serializzazione, come la scelta del protocollo di serializzazione e la possibilità di escludere alcuni attributi dall'oggetto da serializzare.

Una caratteristica importante della libreria pickle è che gli oggetti serializzati possono essere utilizzati su diverse piattaforme e versioni di Python, purché il protocollo di serializzazione utilizzato sia compatibile.

Ciò significa che un oggetto serializzato su un computer Windows con Python 3.9 potrebbe essere deserializzato su un computer Linux con Python 2.7, ad esempio.

Tuttavia, è importante notare che non tutti gli oggetti possono essere serializzati correttamente, come ad esempio le funzioni e le istanze di oggetti di alcune librerie Python.

Inoltre, la compatibilità tra diverse versioni di Python non è garantita in tutti i casi; quindi, è necessario prestare attenzione a eventuali incompatibilità.

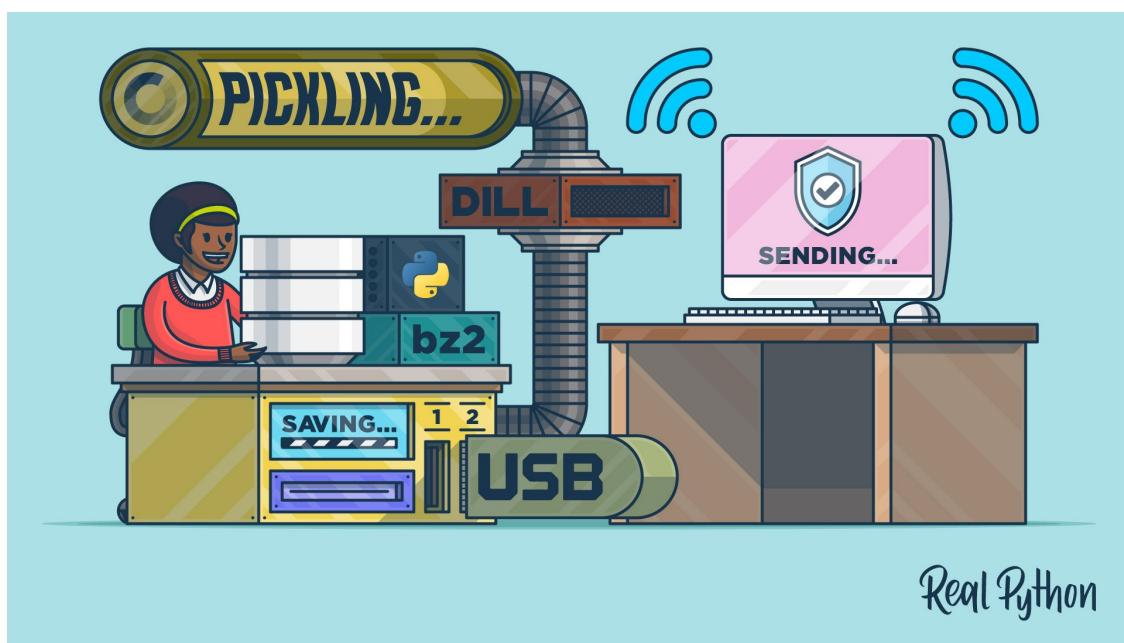


Figura 2.5: Pickle

2.3.4 Torch

PyTorch è un popolare framework open-source di deep learning che consente agli sviluppatori di creare modelli di intelligenza artificiale in modo rapido ed efficiente. È stato sviluppato originariamente da Facebook AI Research, guadagnandosi una grande popolarità per la sua facilità d'utilizzo, la sua flessibilità e la sua scalabilità.

Una delle sue principali caratteristiche è la sua architettura a flusso di dati (data flow), la quale rende il framework particolarmente adatto per le applicazioni di deep learning.

Inoltre, PyTorch è dotato di un'ampia gamma di librerie e strumenti come PyTorch Lightning, che mirano a semplificare lo sviluppo di modelli di intelligenza artificiale.

Questa libreria è anche conosciuta per la sua flessibilità e scalabilità, in quanto permette di creare modelli di deep learning sia per computer singoli che per cluster di computer.

In aggiunta, supporta una vasta gamma di piattaforme hardware, come CPU, GPU e TPU, il che la rende adatta per le applicazioni in ambiti come il machine learning, la visione artificiale e il linguaggio naturale.



Figura 2.6: PyTorch

2.3.5 OpenCV

OpenCV (Open Source Computer Vision Library) [21] è una libreria software open source finalizzata all'ambito della computer vision e del machine learning.

È stata ideata per fornire un'infrastruttura comune per le applicazioni di computer vision e per accelerare l'uso della percezione automatica nei prodotti commerciali.

In quanto prodotto con licenza Apache 2, OpenCV facilita l'utilizzo e la modifica del codice da parte delle aziende.

La libreria contiene più di 2500 algoritmi ottimizzati, che includono un insieme completo di algoritmi di computer vision e machine learning sia classici che all'avanguardia.

Questi algoritmi possono essere utilizzati per:

- rilevare e riconoscere volti,
- identificare oggetti,
- classificare azioni umane nei video,
- tracciare il movimento della telecamera,
- tracciare oggetti in movimento,
- estrarre modelli 3D di oggetti,
- produrre cluster di punti 3D da telecamere stereo,
- unire immagini per produrre un'immagine ad alta risoluzione di un'intera scena,
- trovare immagini simili da un database di immagini,
- rimuovere gli occhi rossi dalle immagini scattate con il flash,
- seguire i movimenti degli occhi,
- riconoscere paesaggi,
- creare marker per sovrapporli alla realtà aumentata, ecc.

OpenCV ha più di 47mila utenti nella sua comunità e un numero stimato di download superiore a 18 milioni.

Oltre alle aziende ben consolidate come Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda e Toyota, esistono molte startup come Applied Minds, VideoSurf e Zeitera che ne fanno un uso intensivo.

Esso è impiegato per molteplici applicazioni, tra cui unire immagini di Street View, rilevare intrusioni in video di sorveglianza in Israele, monitorare l'equipaggiamento minerario in Cina, aiutare i robot a navigare e raccogliere oggetti presso Willow Garage, rilevare gli incidenti di annegamento in piscina in Europa, eseguire arte interattiva in Spagna e New York, controllare le piste di atterraggio per rilevare detriti in Turchia, ispezionare le etichette sui prodotti nelle fabbriche di tutto il mondo e per la rapida rilevazione dei volti in Giappone.

È provvisto di interfacce per C++, Python, Java e MATLAB e supporta Windows, Linux, Android e Mac OS.

Ci sono oltre 500 algoritmi e circa 10 volte tante funzioni che compongono o supportano questi ultimi.

OpenCV è scritto nativamente in C++ e ha un’interfaccia template che funziona perfettamente con i contenitori STL.



Figura 2.7: OpenCV

2.3.6 Scikit-learn

Scikit-learn [22] (precedentemente conosciuto come scikits.learn e anche noto come sklearn) è una libreria di machine learning gratuita per il linguaggio di programmazione Python.

Essa include vari algoritmi di classificazione, regressione e clustering, tra cui support-vector machine, random forest, gradient boosting, k-means e DBSCAN, ed è progettata per funzionare in combinazione con le librerie numeriche e scientifiche di Python, come NumPy e SciPy.

Il progetto scikit-learn è nato come progetto Google Summer of Code dal data scientist francese David Cournapeau, originariamente chiamato scikits.learn. Il nome del progetto deriva dal concetto di "SciKit" (SciPy Toolkit), un'estensione di terze parti separata e distribuita per SciPy.

Il codice originale è stato successivamente riscritto da altri sviluppatori. Nel 2010, i contribuenti Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort e Vincent Michel dall'Istituto francese per la ricerca in informatica e automazione a Saclay, hanno preso il comando del progetto rilasciando in seguito la prima versione pubblica della libreria il 1 febbraio 2010. Nel novembre 2012, scikit-learn e scikit-image sono stati descritti come due delle "scikits library" meglio conservate e popolari. Nel 2019 si è poi stimato che scikit-learn fosse una delle librerie di machine learning più popolari su GitHub.

Scikit-learn è principalmente scritto in Python e utilizza ampiamente NumPy per l'algebra lineare ad alta prestazione e le operazioni sugli array.

Inoltre, alcuni algoritmi core sono scritti in Cython per migliorare le prestazioni. Support vector machine è implementato da un wrapper Cython intorno a LIBSVM; la regressione logistica e le macchine a vettori di supporto lineari da un wrapper simile intorno a LIBLINEAR. In tali casi, estendere questi metodi con Python potrebbe non essere possibile.

Scikit-learn si integra bene con molte altre librerie di Python come Matplotlib e Plotly per la visualizzazione dei dati, NumPy per la vettorizzazione degli array, Pandas dataframes, SciPy e molte altre.



Figura 2.8: SciKit learn

2.3.7 Py-Feat

Per il progetto di tesi ho invece deciso di utilizzare:

Py-Feat: libreria open-source sviluppata in Python per l'estrazione delle feature facciali dalle immagini. Pyfeat si basa sulla fruizione di un insieme di funzioni matematiche, dette funzioni di base, per descrivere la forma e l'aspetto delle suddette espressioni.

Tali funzioni di base sono rappresentate da immagini di capacità espressiva standardizzate, quali sorriso o rabbia, create e validate da esperti di psicologia e neuroscienze.

La libreria Pyfeat fa uso di queste funzioni di base per il calcolo di un insieme di feature facciali, fra cui la forma del viso, la posizione degli occhi, la sagoma delle sopracciglia, la posizione della bocca e le Action Units.

Pyfeat è in grado di estrarre un insieme completo di 280 feature facciali, le 2D FACS (Facial Action Coding System), anch'esse validate da esperti di psicologia e neuroscienze.

Pyfeat può essere impiegato per diverse applicazioni di analisi facciale, come la rilevazione delle emozioni, l'analisi del comportamento non verbale e la diagnostica medica.

Ho ritenuto opportuno ricorrere a questa libreria per l'estrazione delle AUs in quanto affidabile e agevole nell'utilizzo, soprattutto attraverso le funzioni CUDA della libreria che mi hanno permesso di velocizzare di molto il processing delle immagini contenute nei dataset ritrovati; inoltre, nessuno degli studi analizzati ha adoperato questa libreria.

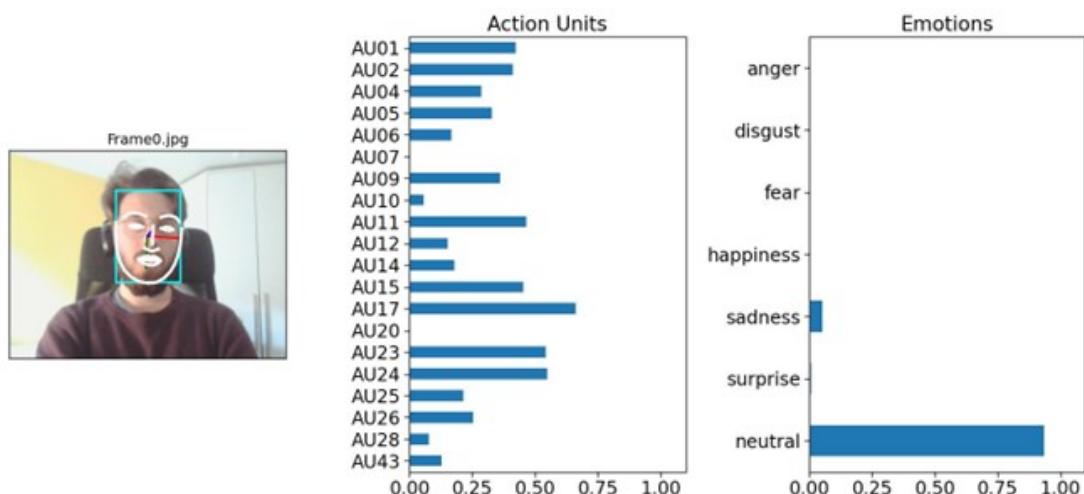


Figura 2.9: Esempio di estrazione AUs tramite py-feat

Capitolo 3

Dataset

Come anticipato nel primo capitolo, il dataset di immagini utilizzato per il mio caso di studio è il risultato dell'unione dei due dataset Student engagement dataset[10] e DAiSEE[11].

In principio le immagini e i video al loro interno sono state elaborate attraverso la libreria py-feat per ottenere le misure delle Action Units.

Le labels risultanti e il numero di sample per ognuna di queste sono:

- engaged con 55707 samples
- bored con 16086 samples
- confused con 1041 samples
- looking away con 409 samples
- frustrated con 893 samples
- drowsy con 240 samples

con un totale di 74322 immagini (o frame estratti da video) per le quali sono stati generati i dati relativi alle Action Units.

3.1 Generazione descrizione in linguaggio naturale

Ho aggiunto una descrizione in linguaggio naturale di ogni immagine utilizzando il seguente codice:

```
if value and value >= 0.5:  
    return outAU.get ("FACS Name") + ", using the muscles: " +  
        outAU.get ("Muscles") + ", with a value of " + str (value) + ";"  
else:  
    return ""
```

Questo algoritmo verifica inizialmente che il valore dell'Action Unit passato in input al metodo (non riportato interamente in quanto prevede azioni preliminari trascurabili) sia presente e, successivamente, in caso fosse provvista di un valore maggiore o uguale a 0.5 (il range di valori è fra 0 e 1) restituisce la stringa che ho descritto nello spazio sottostante; in caso contrario restituirà una stringa vuota.

La frase restituita dall'algoritmo, nel caso in cui il valore sia maggiore o uguale a 0.5, è composta dal nome FACS della relativa Action Unit, con la successiva aggiunta del muscolo analizzato da questa Action Unit e il valore che è stato prelevato.

La frase presente nel dataset per ognuno dei samples è il risultato del concatenamento delle frasi generate per ogni Action Unit e separate da un “;”, ad esempio:

Upper Lip Raiser, using the muscles: Levator Labii Superioris, with a value of 0.6412415504; Dimpler, using the muscles: Buccinator, with a value of 0.6336596608; Chin Raiser, using the muscles: Mentalis, with a value of 0.6474888921; Lip Pressor, using the muscles: Orbicularis Oris, with a value of 0.582298696;

3.2 Estrazione delle Action Units utilizzando la libreria Py-feat

Prima di poter effettuare delle predizioni è necessaria la creazione di un oggetto Detector fornito dalla libreria.

```
device = "cuda" if torch.cuda.is_available() else "cpu"
return Detector(
    device=device,
    face_model="retinaface",
    landmark_model="mobilefacenet",
    au_model="xgb",
    emotion_model="resmasknet",
    facepose_model="img2pose",
)
```

Come è possibile notare nel codice, durante la creazione dell'oggetto Detector è possibile specificare il parametro `device`, permettendo l'esecuzione delle operazioni utilizzando la tecnologia cuda.

Per controllare che sia effettivamente possibile utilizzare questa funzionalità è stata usata la libreria torch per python.

Il parametro `face_model` imposta il modello di rilevamento del viso da utilizzare. Ho ritenuto opportuno impostarlo su `"retinaface"`, popolare modello di rilevamento del viso che utilizza una CNN (Convolutional Neural Networks).

Il parametro `landmark_model` imposta il modello di rilevamento dei landmark facciali da utilizzare. Qui è impostato su `"mobilefacenet"`, un Single-stage dense face localisation in the wild ottenuto dagli autori della libreria da [23].

Il parametro `au_model` prepara il modello utilizzato alla rilevazione automatica dell’unità d’azione (AU) facciale. Il modello è un classificatore Extreme Gradient Boosting (`"xgb"`) estratto, dagli autori di py-feat da i datasets BP4D, DISFA, CK+, UNBC-McMaster shoulder pain, e AFF-Wild2 e basato sul lavoro di [24].

Il parametro `emotion_model` avvia il modello utilizzato per la rilevazione delle emozioni dalle espressioni facciali, ovvero `"resmasknet"`, implementato utilizzando il lavoro di [25].

Il parametro `facepose_model` stabilisce il modello utilizzato per la stima della posa della testa `"img2pose"`, implementato utilizzando il lavoro di [26].

3.2.1 Dati ulteriori alle action units estratti da py-feat

Py-feat permette di estrarre i valori delle Action units attraverso il metodo `detector.detect_image(imagePath)` che prende in input il percorso di un’immagine e restituisce i valori estratti; i valori estratti da questo metodo non si limitano alle Action Units da me utilizzate poiché vengono calcolati anche altri valori, quali:

- FaceRectX: la coordinata X dell’angolo in alto a sinistra del rettangolo del viso rilevato nell’immagine di input
- FaceRectY: la coordinata Y dell’angolo in alto a sinistra del rettangolo del viso rilevato nell’immagine di input
- FaceRectWidth: la larghezza del rettangolo del viso rilevato
- FaceRectHeight: l’altezza del rettangolo del viso rilevato
- FaceScore: un punteggio che indica il livello di fiducia del modello di rilevamento del viso nella regione del viso rilevata
- x_0 a x_67: le coordinate X dei 68 punti landmark facciali rilevati dal modello di landmark
- y_0 a y_67: le coordinate Y dei 68 punti landmark facciali rilevati dal modello di landmark
- Pitch: l’angolo di inclinazione del volto (inclinazione su o giù) rilevato dal modello di posizione del volto

- Roll: l'angolo di rollio del volto (inclinazione a sinistra o destra) rilevato dal modello di posizione del volto
- Yaw: l'angolo di imbardata del volto (girare a sinistra o destra) rilevato dal modello di posizione del volto
- anger, disgust, fear, happiness, sadness, surprise, neutral: i punteggi di probabilità delle classi di emozioni rilevate come previsto dal modello interno alla libreria
- input: il percorso dell'immagine di input
- frame: l'indice del frame elaborato (se si stanno elaborando più di un frame)

3.2.2 Estrazione Action Units dalle immagini

I risultati ottenuti sono poi stati convertiti nel formato json mediante il metodo `detector.detect_image(imagePath).to_json()`, aggregati e salvati su un file, sempre in questo formato, così da poterli mostrare più chiaramente; successivamente questo file è stato trasformato in formato csv per una lettura più veloce da parte della libreria pandas.

3.2.3 Estrazione Action Units dai video

Per quanto riguarda i video analizzati dal dataset DAiSEE la libreria offre il metodo `detector.detect_video(videoPath, skip_frames)`.

Il parametro `videoPath` fa riferimento al percorso del video dal quale estrarre i dati, mentre il parametro `skip_frames` è un intero che determina ogni quanti frame estrapolare l'immagine per calcolarne i relativi valori.

Ho optato per l'estrazione di un'immagine per ogni secondo di video, scrivendo un metodo attraverso il quale estrarre il framerate di ognuno dei video:

```
def getFPS (videoPath):  
    cap = cv2.VideoCapture(videoPath)  
    fps = cap.get(cv2.CAP_PROP_FPS)  
    cap.release()  
    return fps
```

Il risultato di questo metodo è stato poi dato in input al metodo per effettuare l'analisi del video.

Le analisi dei video sono organizzate in modo diverso rispetto alle analisi per le immagini, in quanto ognuno dei campi citati prima (FaceRectX, FaceRectY, ...) contengono i campi per i singoli frame, esempio:

```
"FaceRectX": {
    "0.0": 334.3970982143,
    "30.0": 325.8671875,
    "60.0": 319.8182291667,
    "90.0": 314.8222470238,
    "120.0": 313.5849330357,
    "150.0": 312.7389136905,
    "180.0": 312.5695684524,
    "210.0": 307.6665178571,
    "240.0": 310.235639881,
    "270.0": 312.9242931548
},
...
...
```

3.2.4 Pulizia dei dati

È quindi stato necessario effettuare una rielaborazione dei file ottenuti per portare ognuno dei dati estratti nello stesso formato delle immagini:

```
{
    "FaceRectX": 2.4332027435,
    "FaceRectY": 1.9402399063,
    "FaceRectWidth": 39.422876358,
    "FaceRectHeight": 42.0940465927,
    "FaceScore": 0.6566667557,
    "x_0": 6.6779442048,
    "x_1": 5.354107498,
    "x_2": 4.4593806637,
    ...
}
```

Una volta ottenuti tutti i dati in un singolo file json (e parallelamente nel file csv) questi ultimi sono stati puliti eseguendo queste operazioni:

- pulizia dei valori nulli:

– sono state rimosse le righe dei datasets risultanti dalle analisi, attraverso il codice: `df = df.dropna(subset=['AU01'])`.

Il codice presentato rimuove ogni riga dove il valore della colonna `AU01` è nullo.

In rari casi, py-feat ha riscontrato difficoltà nel riconoscere il volto della persona presente nel video, o questa non era presente all'interno dell'immagine; ciò ha portato al mancato riconoscimento di tutte le AUs

e degli altri dati. Filtrando le righe vuote per una sola colonna (la prima delle AUs) ottengo la rimozione di tutte le righe del tutto vuote in modo efficiente.

Per verificare la mancanza di righe vuote ho eseguito il seguente codice:

```
nullVals = df.isnull()  
  
print("Numero di valori nulli per ogni colonna:")  
print(nullVals.sum())
```

l'output è riportato all'immagine 3.1:

```
Numero di valori nulli per ogni colonna:  
AU01          0  
AU02          0  
AU04          0  
AU05          0  
AU06          0  
AU07          0  
AU09          0  
AU10          0  
AU11          0  
AU12          0  
AU14          0  
AU15          0  
AU17          0  
AU20          0  
AU23          0  
AU24          0  
AU25          0  
AU26          0  
AU28          0  
AU43          0  
input          0  
naturalLanguageDescription 0  
label          0  
numLabel       0
```

Figura 3.1: Numero di valori nulli ancora presenti nel dataset risultato

Come è possibile osservare dall'immagine 3.1, il dataset risultato non presenta valori nulli.

- aggiunta del valore di frame per ognuna delle analisi dei video:

- ogni analisi dei singoli frame di un video presentava lo stesso percorso di input (il file video associato); di conseguenza ho aggiunto alla fine del valore della colonna di input il frame dal quale sono state estratte le analisi
- rimozione delle colonne che non riguardano le Action Units
- aggiunta colonne al dataset
 - label:
 - * le analisi inizialmente estrapolate non presentavano già le relative label; è stato quindi opportuno aggiungere
 - numLabel
 - * ho associato ad ognuna delle label presenti un numero da 0 a 5:
 - 0 → confused
 - 1 → engaged
 - 2 → frustrated
 - 3 → bored
 - 4 → drowsy
 - 5 → looking away
 - descrizione in linguaggio naturale descritta precedentemente

3.3 Resampling del dataset per migliorare la precisione delle predizioni

Il dataset risultato da me realizzato presenta una notevole complicazione: se decidessi di effettuare delle predizioni attraverso uno dei classificatori generati direttamente dal dataset as-his, il numero di campioni (samples) per ognuno dei valori della colonna labels risulterebbe sbilanciato.

Per sbilanciato intendo il fatto che sono presenti molti valori per alcune delle classi (mood), e troppi pochi, a confronto, per altri, come possibile notare nell’immagine 3.2.

label	
engaged	55707
bored	16086
confused	1041
frustrated	839
looking away	409
drowsy	240

Figura 3.2: Numero di sample per ogni mood nel dataset iniziale

Dataset

Nei grafici 3.3 3.4 è possibile riscontrare la differenza fra il numero di elementi per ogni valore unico nella colonna label:

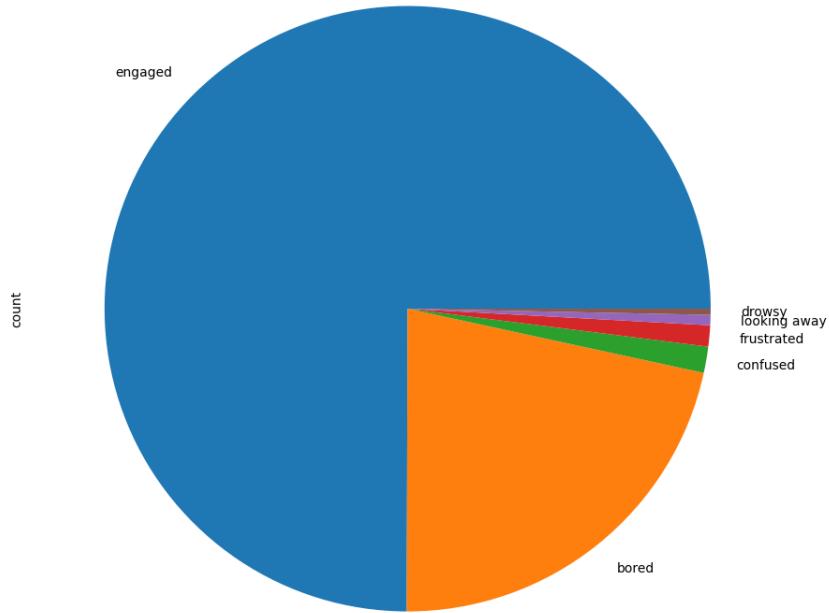


Figura 3.3: Grafico a torta del numero di samples per ogni mood

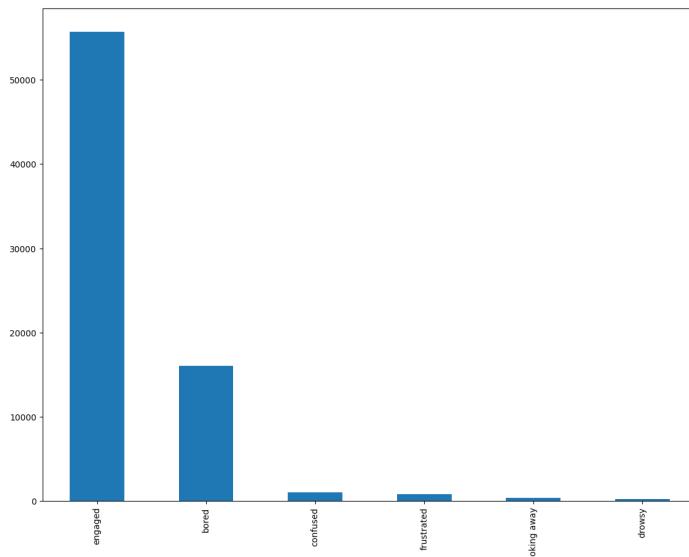


Figura 3.4: Grafico a barre del numero di samples per ogni mood

Il resampling, che prevede una modifica della distribuzione dei dati di un dataset mediante la rimozione o l'aggiunta, in modo casuale, di valori simili a quelli delle sue

istanze, è una tecnica comune per bilanciare dataset sbilanciati o per migliorare le prestazioni di modelli di machine learning. Esistono due tipi principali di resampling: undersampling e oversampling:

- L’undersampling, come suggerisce il nome, consiste nel rimuovere alcune delle istanze della classe maggioritaria (ovvero quella con un maggior numero di campioni) in modo da bilanciare la distribuzione delle classi nel dataset.

Ciò può essere effettuato in modo casuale, pur essendo possibile utilizzare tecniche più sofisticate, come l’eliminazione degli esempi più vicini (nearest neighbor deletion) o la selezione degli esempi più rappresentativi (prototype selection).

Il codice utilizzato per effettuare l’undersampling dei sample per ognuna delle labels è il seguente:

```
import pandas as pd
def undersampleDataset(df, columnName, valueToDownSample, number_of_samples_after):
    if len(df[df[columnName] == valueToDownSample]) > number_of_samples_after:
        engagedIndices = df[df[columnName] == valueToDownSample].index
        tempDf = df.loc[engagedIndices]
        tempDfUndersampled = tempDf.sample(n=number_of_samples_after, random_state=42)
        return pd.concat([df.drop(engagedIndices), tempDfUndersampled])
    return df
```

- L’oversampling, d’altra parte, consiste nell’aumentare il numero di istanze della classe minoritaria (ovvero quella con un minor numero di campioni) in modo da bilanciare la distribuzione delle classi nel dataset.

Sussistono molte tecniche per l’oversampling, tra cui la duplicazione casuale degli esempi esistenti, la generazione di nuovi esempi sintetici attraverso tecniche come la Synthetic Minority Over-sampling Technique (SMOTE), e la duplicazione degli esempi esistenti con una variazione minore (data augmentation).

Il codice utilizzato per effettuare l’oversampling dei sample per ognuna delle labels è il seguente:

```
import pandas as pd
from sklearn.utils import resample
def oversampleDataset(df, columnName, value_to_oversample, number_of_samples_after):
    if len(df[df[columnName] == value_to_oversample]) < number_of_samples_after:
        engagedIndices = df[df[columnName] == value_to_oversample].index
        tempDf = df.loc[engagedIndices]
        tempDfOversampled = resample(
            tempDf,
            replace=True,
            n_samples=number_of_samples_after,
            random_state=42
        )
        return pd.concat([df.drop(engagedIndices), tempDfOversampled])
    return df
```

In sintesi, il resampling è una tecnica utile per bilanciare dataset e migliorare le prestazioni dei modelli di machine learning.

Questo invece è il codice che gestisce le chiamate per entrambi i metodi riportati sopra:

```
def resampleDataset():
    df = pd.read_csv(oldDatasetPath)
    visualizeDataFrameChart(df)

    numberOfRowsForEachLabel = 2000

    labelsList = df["label"].unique()
    for label in labelsList:
        df = undersampleDataset (
            df,
            "label",
            label,
            numberOfRowsForEachLabel
        )
    for label in labelsList:
        df = oversampleDataset (
            df,
            "label",
            label,
            numberOfRowsForEachLabel
        )

    df.to_csv(newDatasetPath, index=False)
```

1. `sklearn.utils.resample` è una funzione fornita dalla libreria scikit-learn che viene utilizzata al fine di generare esempi sintetici per l'oversampling di dataset. In particolare, la funzione `resample` prende in input un insieme di campioni e genera un nuovo insieme di campioni sintetici.

La funzione `resample` prende in input i seguenti parametri:

- X: un array o un dataframe che rappresenta le feature dei campioni
- y: (non valorizzato) un array o un dataframe che rappresenta le label dei campioni
- replace: un valore booleano che indica se l'oversampling deve essere fatto con o senza sostituzione (ovvero se gli esempi sintetici possono essere duplicati)
- n_samples: il numero di esempi sintetici da generare
- random_state: un valore intero che rappresenta il seed per la generazione casuale degli esempi sintetici

La funzione `resample` restituisce due oggetti:

- X_resampled: un array o un dataframe contenente le feature dei campioni originali e dei nuovi esempi sintetici generati (unico output utilizzato)

- `y_resampled`: un array o un dataframe contenente le label dei campioni originali e dei nuovi esempi sintetici generati

In sostanza, la funzione resample genera nuovi esempi sintetici aggiungendo variazioni minime ai campioni esistenti, in modo da produrre una distribuzione bilanciata delle classi nel dataset.

Questi nuovi esempi sintetici vengono poi utilizzati insieme ai campioni esistenti per addestrare i modelli di machine learning.

2. `pandas.DataFrame.sample` è una funzione fornita dalla libreria pandas che viene utilizzata per estrarre casualmente un sottinsieme di righe da un dataframe.

In particolare, la funzione sample prende in input un dataframe e restituisce un nuovo dataframe contenente solo un sottinsieme delle righe del dataframe originale.

- `n`: il numero di righe da estrarre casualmente
- `frac`: (non valorizzato) la frazione di righe da estrarre casualmente (ad esempio, 0.5 per estrarre il 50)
- `replace`: (non valorizzato) un valore booleano che puntualizza se le righe estratte devono essere selezionate con o senza sostituzione (ovvero se una stessa riga può essere selezionata più volte)
- `weights`: (non valorizzato) un array di pesi per ogni riga, utilizzato per selezionare le righe in modo ponderato
- `random_state`: un valore intero che rappresenta il seed per la generazione casuale degli indici delle righe da selezionare

La funzione sample restituisce un nuovo dataframe contenente solo il sottinsieme delle righe selezionate casualmente dal dataframe originale. Fondamentalmente, la funzione sample è utilizzata per ridurre la dimensione del dataframe originale, selezionando solo un sottinsieme casuale delle righe. Ciò può dimostrarsi utile per ridurre i tempi di calcolo durante l'addestramento dei modelli di machine learning, in particolare quando il dataset originale è molto ampio e non è necessario utilizzare tutte le righe per ottenere un buon modello.

Per l'appunto, prima di effettuare il resampling del dataset, ho provato ad effettuare delle analisi su nuove immagini utilizzando un classificatore generato dal dataset as-his e i risultati si sono rivelati quelli aspettati, ovvero:

Quasi sempre veniva rilevato che la persona ripresa nell'immagine aveva delle Action Units che portavano alla predizione “engaged” e le altre volte veniva rilevato

lo stato d'animo “bored”, ignorando del tutto gli altri valori presenti per la colonna label.

Dopo aver messo in pratica vari test, sono giunto alla conclusione di eseguire un resample al fine di ottenere 2000 istanze per ogni label, in quanto questo sembra il numero di sample in grado di far conseguire risultati più coreenti nel tempo; i risultati relativi ottenuti sono riportati nell'ultimo capitolo.

Di seguito riporto i grafici 3.5 e 3.6 riguardanti il dataset presentati all'inizio, generati dopo aver effettuato il resampling:

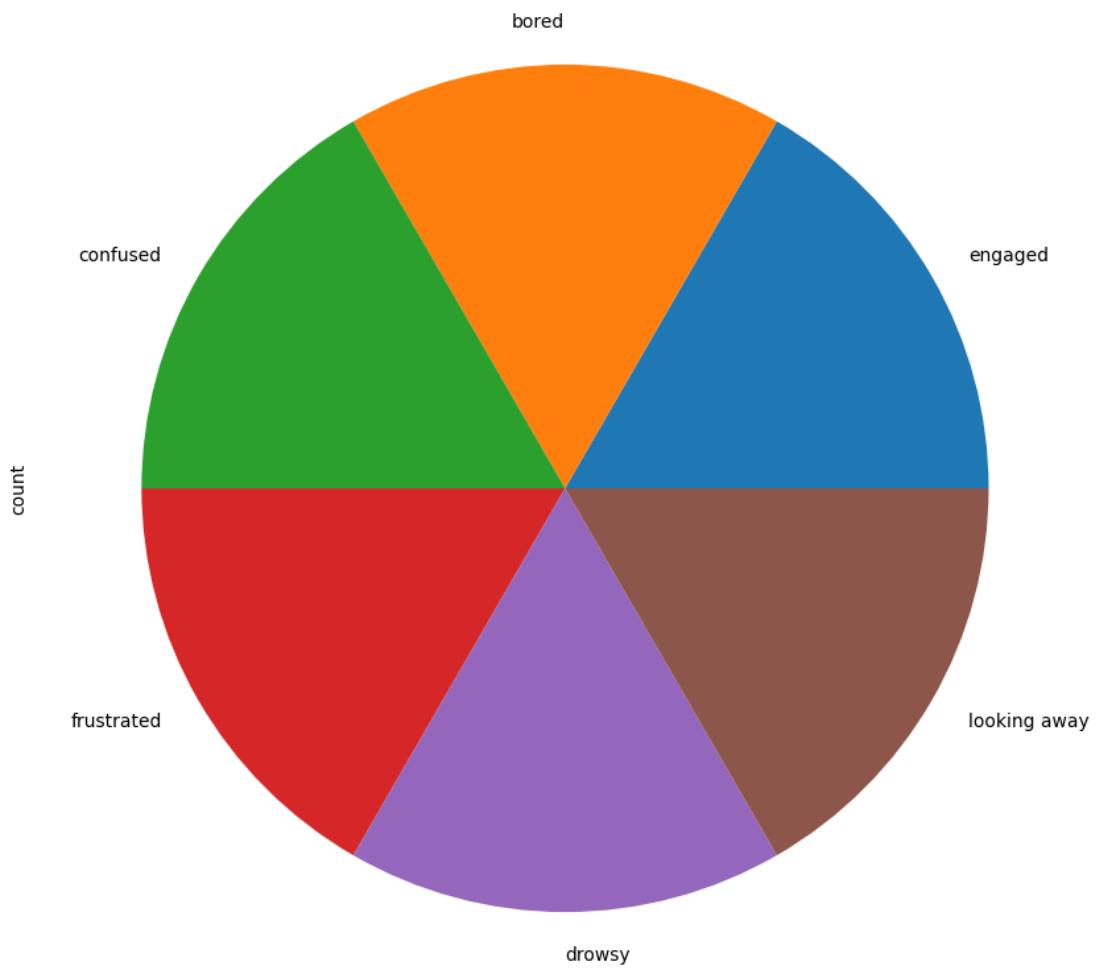


Figura 3.5: Grafico a torta del numero di samples per ogni mood post resampling

3.3 – Resampling del dataset per migliorare la precisione delle predizioni

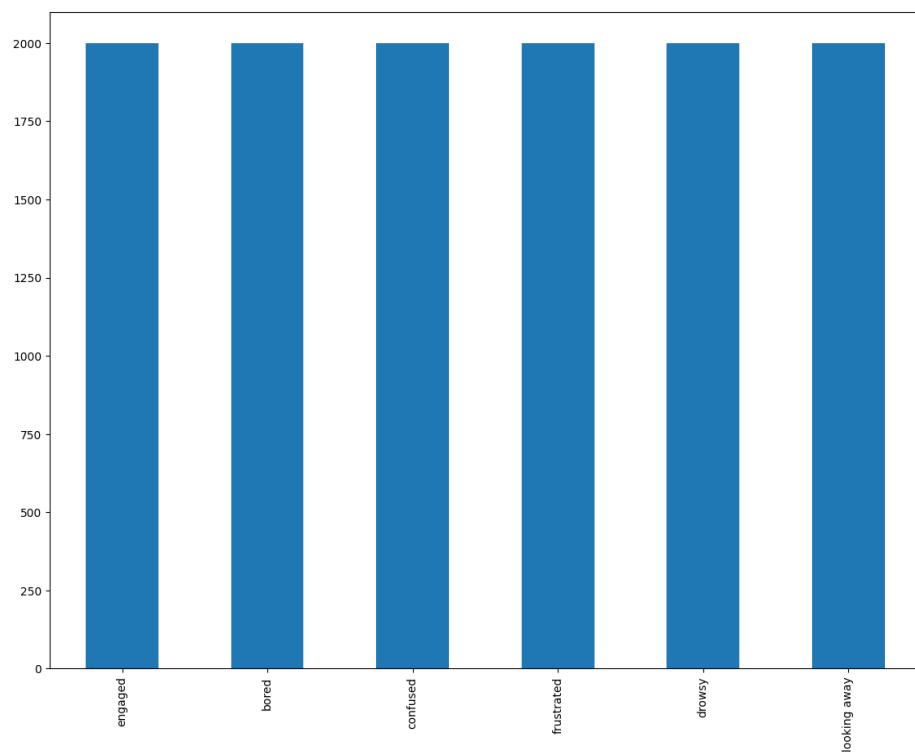


Figura 3.6: Grafico a barre del numero di samples per ogni mood post resampling

Capitolo 4

Modelli predittivi utilizzati

4.1 Algoritmi utilizzati per la creazione dei modelli predittivi

4.1.1 Random Forest

[27]

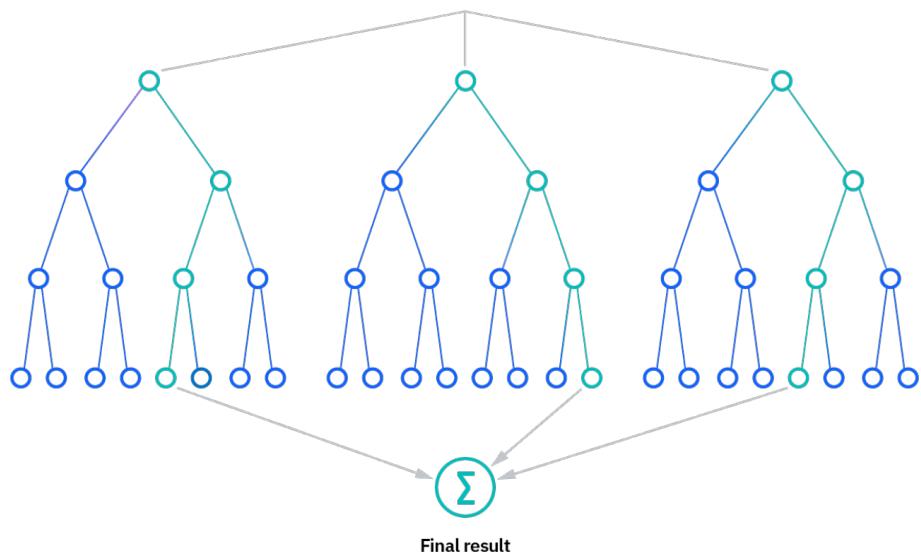


Figura 4.1: Rappresentazione di Random Forest

La random forest è un algoritmo di apprendimento automatico creato da Leo Breiman e Adele Cutler, che combina l'output di più alberi decisionali col fine di raggiungere un singolo risultato.

La sua predisposizione intuitiva e la sua flessibilità hanno nutrito l'aumento della sua adozione, in quanto gestisce sia problemi di classificazione che di regressione.

Alberi decisionali

Dato che il modello di random forest è composto da più alberi decisionali, è utile descrivere brevemente l'algoritmo dell'albero decisionale.

Gli alberi decisionali partono da una domanda di base, ad esempio "Dovrei fare surf?".

A partire da ciò è possibile porre una serie di domande per determinare una risposta, come "C'è un'onda di lungo periodo?" o "Il vento soffia a riva?".

Queste domande costituiscono i nodi decisionali dell'albero, agendo come mezzo di ripartizione dei dati.

Ogni domanda aiuta l'albero a giungere a una decisione finale, indicata dal nodo foglia raggiunto.

Le osservazioni che soddisfano i criteri seguiranno il ramo "Sì", mentre quelle che non li soddisfano seguiranno il percorso alternativo.

Gli alberi decisionali cercano di trovare la miglior suddivisione per i dati, e vengono tipicamente addestrati attraverso l'algoritmo Classification and Regression Tree (CART).

Metriche come l'impurità di Gini, il guadagno di informazione o l'errore quadratico medio (MSE) possono essere utilizzati per valutare la qualità della suddivisione.

Sebbene gli alberi decisionali siano comuni algoritmi di apprendimento supervisionato, possono essere soggetti a problemi come il bias e l'overfitting.

Tuttavia, quando più alberi decisionali formano un insieme nell'algoritmo di random forest, predicono risultati più accurati, in particolar modo quando i singoli alberi non sono correlati tra loro.

L'algoritmo della random forest

L'algoritmo della random forest è un'estensione del metodo di bagging, in quanto utilizza sia il bagging che la casualità delle features per creare una foresta di alberi decisionali non correlati.

La casualità delle features, altrimenti nota come bagging delle caratteristiche o "metodo del sottospazio casuale", genera un sottoinsieme casuale di caratteristiche che assicura una bassa correlazione tra gli alberi decisionali.

Questa è una differenza chiave tra gli alberi decisionali e le random forest, poiché mentre gli alberi decisionali considerano tutte le possibili suddivisioni delle caratteristiche, le random forest selezionano solo un sottoinsieme di quelle caratteristiche.

Tornando all'esempio "dovrei fare surf?", le domande che potrei porre per determinare la previsione potrebbero non essere così esaustive come il set di domande di un altro utente.

Tenendo conto di tutta la potenziale variabilità dei dati, possiamo ridurre il rischio di overfitting, di bias e di varianza complessiva, ottenendo previsioni via via più precise.

Come funziona

Gli algoritmi delle random forest hanno tre iperparametri principali da impostare prima dell'allenamento. Questi includono:

- la dimensione del nodo,
- il numero di alberi,
- il numero di caratteristiche campionate

Da lì, il classificatore della random forest può essere utilizzato per risolvere problemi di regressione o di classificazione.

L'algoritmo della random forest è costituito da una collezione di alberi decisionali, dove ogni albero nell'insieme è costituito da un campione di dati tratto da un set di allenamento con sostituzione, chiamato campione di bootstrap.

Di quel campione di allenamento, ad esempio, un terzo viene messo da parte come dati di test, noti come campione fuori dalla borsa (oob), su cui ci soffermeremo in seguito.

Un'altra istanza di casualità viene quindi iniettata attraverso il bagging delle caratteristiche, incrementando la diversità nel dataset e riducendo la correlazione tra gli alberi decisionali.

A seconda del tipo di problema, la determinazione della previsione subirà variazioni; per un compito di regressione gli alberi decisionali individuali verranno mediati, mentre per un compito di classificazione una maggioranza di voti, ossia la variabile categorica più frequente, darà come risultato la classe prevista.

Infine, il campione oob viene utilizzato per la convalida incrociata, finalizzando quella previsione.

Benefici e sfide del random forest

Ci sono diversi vantaggi e sfide che l'algoritmo random forest presenta quando utilizzato per problemi di classificazione o regressione. Alcuni di essi includono:

- Principali vantaggi

- Riduzione del rischio di overfitting: gli alberi di decisione corrono il rischio di overfitting poiché tendono ad adattarsi strettamente a tutti i campioni all'interno dei dati di formazione. Tuttavia, quando è presente un largo numero di alberi di decisione in un random forest, il classificatore non sovrastimerà il modello, poiché la media di alberi non correlati fra loro riduce la variazione complessiva e l'errore di previsione.
- Flessibilità: il random forest può gestire sia compiti di regressione che di classificazione con un elevato grado di precisione, distinguendosi in quanto metodo popolare tra i data scientist. Inoltre, la feature bagging rende il classificatore random forest uno strumento adeguato per la stima dei valori mancanti, in quanto mantiene l'accuratezza quando una parte dei dati è irreperibile.
- Facile determinazione dell'importanza delle feature: il random forest rende facile valutare l'importanza delle variabili, o del contributo, al modello. Sussistono alcuni modi per valutare l'importanza della feature; l'importanza di Gini e la diminuzione media dell'impurità (MDI) vengono solitamente utilizzati per misurare quanto diminuisce l'accuratezza del modello quando una determinata variabile viene esclusa.

- Principali sfide

- È un processo che richiede tempo: gli algoritmi random forest possono gestire grandi set di dati e fornire previsioni più accurate; tuttavia il processo è rallentato dalla computazione dei dati per ogni singolo albero decisionale.
- Richiede più risorse: le random forest, elaborando set di dati più grandi, richiedono più risorse per elaborare questi ultimi.
- Più complesso: la previsione di un singolo albero decisionale è più facile da interpretare rispetto a una foresta di questi.

4.1.2 K-nearest neighbors

[28]

L'algoritmo k-nearest neighbors, noto anche come KNN o k-NN, è un classificatore di apprendimento supervisionato non parametrico.

Esso sfrutta la prossimità per effettuare classificazioni o previsioni sul raggruppamento di un singolo punto dati.

Sebbene possa essere utilizzato per problemi di regressione o classificazione, viene generalmente impiegato in quanto algoritmo di classificazione, basandosi sul presupposto che dati simili, se analizzati o rappresentati nella giusta maniera, possono essere trovati l'uno vicino all'altro.



Figura 4.2: Rappresentazione di K-nearest neighbors

Per i problemi di classificazione un’etichetta di classe viene assegnata sulla base di un voto a maggioranza(ad es. viene utilizzata l’etichetta più frequentemente rappresentata attorno a un determinato punto dati).

I problemi di regressione utilizzano un concetto simile al problema di classificazione; in questo caso però viene presa la media dei k elementi vicini più vicini per effettuare una previsione su una classificazione.

Qui, ciò che si discosta notevolmente, è il fatto che la classificazione viene utilizzata per i valori discreti, mentre la regressione viene utilizzata per quelli continui.

Tuttavia, prima di poter effettuare una classificazione, è necessario definire il concetto di distanza.

La distanza euclidea, altra metrica di distanza popolare, misura il valore assoluto tra due punti.

Vale la pena notare che l’algoritmo KNN fa anche parte di una famiglia di modelli di "apprendimento pigro", il che significa che memorizza solo un set di dati di addestramento nella fase di addestramento.

Perviene quindi che tutto il calcolo avvenga quando si compie una classificazione o una previsione.

Poiché fa ampiamente affidamento sulla memoria per archiviare tutti i suoi dati di addestramento, viene anche definito metodo di apprendimento basato su istanze o basato sulla memoria.

Le idee iniziali sul modello KNN sono attribuite a Evelyn Fix e Joseph Hodges in questo articolo [29] del 1951, mentre Thomas Cover ne amplia il concetto nella sua ricerca, “Nearest Neighbor Pattern Classification”[30].

Pur non riscuotendo un tale successo come in precedenza, è ancora uno dei primi algoritmi che si affronta nello studio della data science per la sua semplicità ed accuratezza.

Tuttavia, al crescere di un set di dati KNN diventa sempre più inefficiente, compromettendo le prestazioni del modello.

Viene riscontrato il suo impiego per semplici sistemi di raccomandazione, riconoscimento di modelli, data mining, previsioni dei mercati finanziari, rilevamento delle intrusioni e altro ancora.

Calcolare il KNN: metriche di distanza

Ricapitolando, l'obiettivo dell'algoritmo k-nearest neighbor è identificare i vicini più prossimi di un dato punto di query, in modo da poter assegnare un'etichetta di classe a quel punto.

Per determinare quali punti dati sono più attigui ad un determinato punto di query, sarà necessario calcolare la distanza tra il punto di interrogazione e gli altri punti dati.

Le metriche di distanza aiutano a formare confini decisionali che suddividono i punti di query in regioni diverse.

Sebbene sussistano diverse misure di distanza, tra cui è possibile scegliere, l'articolo sul sito di IBM tratta esclusivamente quelle a seguire:

- Distanza euclidea ($p=2$): la misura della distanza più comunemente adottata; essa è limitata ai vettori con valori reali.

Utilizzando la formula 4.1 traccia una linea retta tra il punto di query e l'altro punto che si vuole misurare.

$$d(x, y) = \sqrt{\sum_{i=1}^n (y_i - x_i)^2} \quad (4.1)$$

- Distanza di Manhattan ($p=1$): trascitta a 4.2, è un'altra metrica di distanza particolarmente nota che si propone di misurare il valore assoluto tra due punti.

È anche riconosciuta come distanza del taxi o distanza del blocco cittadino, poiché è comunemente visualizzata come una griglia che illustra come si

potrebbe percorrere il tragitto da un indirizzo all’altro attraverso le strade della città.

$$d(x, y) = \sum_{i=1}^m |x_i - y_i| \quad (4.2)$$

- Distanza di Minkowski: questa misura della distanza è la forma generalizzata delle metriche di distanza euclidea e di Manhattan.

La distanza Euclidea è rappresentata dalla formula 4.3 quando p è uguale a due e la distanza di Manhattan è indicata con p uguale a uno.

$$\left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}} \quad (4.3)$$

- Distanza di Hamming: riportata a 4.4, questa tecnica viene utilizzata tipicamente con vettori booleani o stringa, identificando i punti in cui i vettori non trovano corrispondenza.

Di conseguenza, è stata anche definita metrica di sovrapposizione.

$$d_H(x, y) = \sum_{i=1}^k |(x_i - y_i)| \quad (4.4)$$

Ad esempio, se avessi le seguenti stringhe, la distanza di Hamming sarebbe due poiché solo due dei valori differiscono.

Vector 1	1	0	1	0	0	0	1	1
Vector 2	1	0	0	0	0	0	0	1

Figura 4.3: Esempio per la distanza di Hamming

Calcolare KNN: definizione di k

Il valore k nell’algoritmo k-NN definisce quanti vicini verranno controllati per determinare la classificazione di un punto di query specifico.

Di fatti, se k=1, l’istanza verrà assegnata alla stessa classe del suo singolo neighbor più vicino.

Definire k può essere un atto di bilanciamento, in quanto valori diversi possono portare a overfitting o underfitting.

Valori inferiori a k possono essere caratterizzati da una variabilità elevata ma una bassa distorsione, mentre valori maggiori di k possono portare a una distorsione elevata e una variabilità inferiore.

La scelta di k dipenderà in particolar modo dai dati di input, dal momento che i dati con più valori anomali o rumore probabilmente opereranno in modo più proficuo tanto più elevati sono i valori di k.

In generale si consiglia di adoperare un numero dispari per k, col fine di evitare pareggi nella classificazione.

Applicazioni di k-NN nell'apprendimento automatico

L'algoritmo k-NN è stato utilizzato all'interno di una varietà di applicazioni, in maggior misura all'interno della classificazione. Alcuni di questi casi d'uso includono:

- Pre-elaborazione dei dati: poichè i dataset presentano spesso valori mancanti, l'algoritmo KNN può stimare tali valori in un processo noto come imputazione dei dati mancanti.
- Recommender systems: adoperando i dati del flusso di clic dai siti web, l'algoritmo KNN è stato utilizzato per fornire consigli automatici agli utenti su contenuti aggiuntivi. Da tale ricerca emerge che un utente è assegnato a un particolare gruppo e, sulla base del comportamento di quest'ultimo, riceve un consiglio. Tuttavia, dati i problemi di scalabilità con KNN, questo approccio potrebbe non risultare ottimale in caso di impiego di dataset più grandi.
- Finanza: è stato utilizzato anche in una varietà di casi di utilizzo finanziari ed economici. Ipoteticamente, un articolo mostra in che modo l'utilizzo di KNN sui dati di credito può aiutare le banche nella valutazione dei rischi su un prestito a un'organizzazione o a un individuo. Viene quindi sfruttato per determinare l'affidabilità creditizia del richiedente di tale prestito. Un altro articolo ne sottolinea l'uso nelle previsioni del mercato azionario, nei tassi di cambio, nel trading di futures e nelle analisi sul riciclaggio di denaro.
- Assistenza sanitaria: KNN ha riscontrato applicazioni anche nel settore dell'assistenza sanitaria, effettuando previsioni sul rischio di infarto e cancro alla prostata. L'algoritmo funziona calcolando le espressioni geniche più probabili.
- Riconoscimento dei pattern: KNN ha anche aiutato a identificare i pattern, come nel testo e nella classificazione digitale. Ciò è stato particolarmente utile per identificare i numeri scritti a mano in cui ci si potrebbe imbattere su moduli o buste postali.

Vantaggi e svantaggi dell'algoritmo KNN

K-NN possiede i suoi punti di forza e di debolezza. A seconda del progetto e dell'applicazione, potrebbe rivelarsi o meno la scelta giusta.

- Vantaggi

- Facile da implementare: data la semplicità e l'accuratezza dell'algoritmo, è uno dei primi classificatori che un data scientist alle prime armi apprenderà.
- Si adatta facilmente all'aggiunta di nuovi campioni di addestramento, l'algoritmo si adatta per tenere conto di eventuali nuovi dati, a fronte dell'archivio di tutti i dati di addestramento in memoria.
- Pochi iperparametri: KNN necessita solo di un valore k e una metrica di distanza, il che, a confronto con altri algoritmi di machine learning, è minore in quantità.

- Svantaggi

- Non è provvisto di una buona scalabilità: essendo KNN un algoritmo cosiddetto pigro, occupa più memoria e spazio di storage dei dati rispetto ad altri classificatori. Sebbene diverse strutture di dati, come Ball-Tree, siano state create per affrontare le inefficienze computazionali, un classificatore diverso potrebbe dimostrarsi ideale.
- Maledizione della dimensionalità: l'algoritmo tende a cadere vittima della "maledizione della dimensionalità"; ciò significa che non ricopre adeguatamente il proprio ruolo con input di dati ad alta dimensionalità. Questo è a volte indicato anche come il fenomeno del picco, nel quale, dopo che l'algoritmo raggiunge l'ottimale numero di funzioni, le funzioni aggiuntive aumentano la quantità di errori di classificazione, soprattutto quando la dimensione del campione è inferiore.
- Propensione al sovrardimensionamento dei dati: a causa della "maledizione della dimensionalità", KNN è anche più propenso al sovrardimensionamento dei dati. Sebbene le tecniche di selezione delle caratteristiche e di riduzione della dimensionalità vengano sfruttate per evitare che ciò accada, il valore di k può inevitabilmente influire sul comportamento del modello. Valori più bassi di k possono sovra-alimentare i dati, mentre valori più alti di k tendono a "smussare" i valori di previsione, poiché si sta attuando la media dei valori su un'area o un neighborhood più grande. Tuttavia, se il valore di k è troppo alto, può essere inferiore ai dati.

4.1.3 Naive Bayes Classifier

[31] Un classificatore di Bayes è un modello di apprendimento automatico probabilistico utilizzato per compiti di classificazione. Il fulcro del classificatore si basa sul teorema di Bayes.

Teorema di Bayes

Utilizzando il teorema di Bayes, di cui formula 4.5, possiamo trovare la probabilità che A accada, conseguentemente all'avvenimento di B. Qui, B è la prova e A è l'ipotesi.

L'assunzione alla base del teorema è che i predittori e le caratteristiche siano indipendenti: in altre parole, la presenza di una particolare caratteristica non influisce sull'altra.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (4.5)$$

Volendo offrire un esempio al fine di una migliore comprensione, consideriamo il problema di giocare a golf. Il dataset è rappresentato in 4.4.

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

Figura 4.4: Dataset esempio

Classifichiamo se la giornata si configura adatta per una partita di golf, basandoci sugli attributi di tale giornata. Le colonne rappresentano questi attributi, mentre le righe rappresentano le singole voci.

Prendendo a campione la prima riga del dataset, possiamo osservare che non è adatta per giocare a golf se il cielo è nuvoloso, la temperatura è calda, l'umidità è alta e non c'è vento. Qui facciamo due assunzioni:

come già detto, consideriamo che questi predittori siano indipendenti (ovvero, se la temperatura è calda, non significa necessariamente che l'umidità sia alta). Un'altra assunzione fatta qui è che tutti i predittori abbiano un effetto uguale sul risultato. Cioè, il fatto che ci sia vento non ha una maggiore importanza nella decisione di giocare a golf o meno.

Rifacendosi al secondo esempio, il teorema di Bayes può essere quindi riscritto come in 4.6.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \quad (4.6)$$

La variabile y è la variabile di classe (giocare a golf), la quale rappresenta quanto sia adatta la giornata o meno per giocare a golf. La variabile X rappresenta i parametri/caratteristiche.

X è dato come 4.7:

$$X = (x_1, x_2, x_3, \dots, x_n) \quad (4.7)$$

Qui x_1, x_2, \dots, x_n raffigurano le caratteristiche, cioè possono essere mappati su cielo, temperatura, umidità e vento. Sostituendo X ed espandendo utilizzando la regola della catena otteniamo 4.8.

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)} \quad (4.8)$$

Ora è possibile ottenere i valori per ognuno guardando il dataset e sostituendoli nell'equazione. Per tutte le voci nel dataset, il denominatore non cambia, rimane statico. Pertanto, il denominatore può essere rimosso e può essere introdotta una proporzionalità. Formula 4.9.

$$P(y|x_1, \dots, x_n) = P(y) \prod_{i=1}^n P(x_i|y) \quad (4.9)$$

Nel nostro caso, la variabile di classe (y) ha solo due risultati, sì o no. Potrebbero esserci casi in cui la classificazione potrebbe rivelarsi multivariata. Pertanto, sarà necessario trovare la classe y con la massima probabilità. Formula 4.10.

$$y = \arg \max_y P(y) \prod_{i=1}^n P(x_i|y) \quad (4.10)$$

Utilizzando 4.10 ci è possibile ottenere la classe, dato il predittore.

Tipi di Classificatori Bayesiani

- Classificatore Bayesiano Multinomiale:
 - Esso è principalmente utilizzato per problemi di classificazione dei documenti; ad esempio, se un documento appartiene alla categoria di sport, politica, tecnologia, ecc. Le caratteristiche/predittori impiegati dal classificatore saranno la frequenza delle parole presenti nel documento.
- Classificatore Bayesiano di Bernoulli:
 - Questo classificatore è simile al precedentemente citato, pur essendo i predittori variabili booleane. I parametri che usiamo per prevedere la variabile di classe assumono solo valori sì o no (se una parola compare nel testo o meno).
- Classificatore Bayesiano Gaussiano:
 - Quando i predittori assumono un valore continuo e non discreto, assumiamo che questi valori siano estratti da una distribuzione gaussiana.

Poiché il modo in cui i valori sono presenti nel dataset cambia, la formula per la probabilità condizionata cambia come in 4.11

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \quad (4.11)$$

Conclusione

Gli algoritmi Bayesiani sono principalmente utilizzati nell'analisi del sentiment, nel filtraggio dello spam, nei recommender systems, ecc.

Sono veloci e facili da implementare, ma il loro più grande svantaggio è la necessità che i predittori siano indipendenti.

Nella maggior parte dei casi reali, i predittori sono dipendenti, il che ostacola le prestazioni del classificatore.

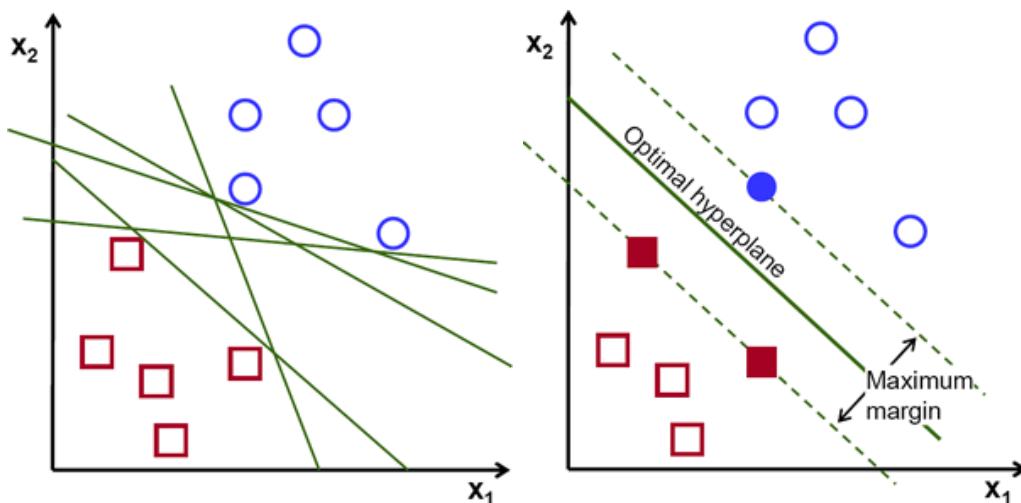
4.1.4 Support Vector Machine

Il support vector machine è favorito maggiormente data la produzione di una significativa accuratezza mediante una minor potenza di calcolo.

Il Support Vector Machine, abbreviato come SVM, può essere utilizzato sia per compiti di regressione che di classificazione. Tuttavia, viene ampiamente sfruttato negli obiettivi di classificazione.

Cosa è il Support Vector Machine?

L’obiettivo dell’algoritmo support vector machine è quello di trovare un iperpiano in uno spazio N-dimensionale (N - il numero di caratteristiche) che classifichi distintamente i punti dati.



Per separare le due classi di punti dati sono presenti molti iperpiani possibili da cui poter scegliere. Il nostro obiettivo è trovare un piano che possegga il margine massimo, ovvero la massima distanza tra i punti dati di entrambe le classi. Massimizzare la distanza del margine fornisce un rinforzo in modo che i futuri punti dati possano essere classificati con maggiore fiducia.

Iperpiani e support vector

Gli iperpiani sono i confini decisionali che aiutano a classificare i punti dati.

I punti dati che cadono su entrambi i lati dell’iperpiano possono essere attribuiti a diverse classi.

Inoltre, la dimensione dell’iperpiano dipende dal numero di caratteristiche.

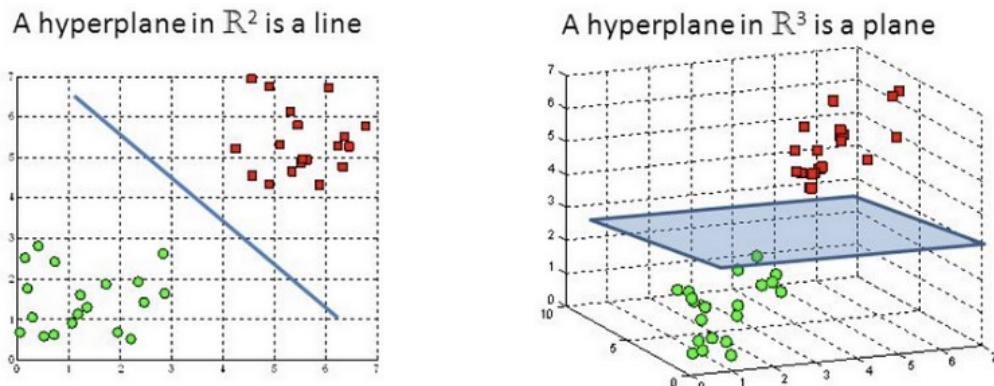


Figura 4.6: Rappresentazione iperpiani

Se il numero di caratteristiche di input è 2, l'iperpiano è solo una linea. Se il numero di caratteristiche di input è 3, l'iperpiano diventa un piano bidimensionale. Diventa difficile immaginare quando il numero di caratteristiche supera 3.

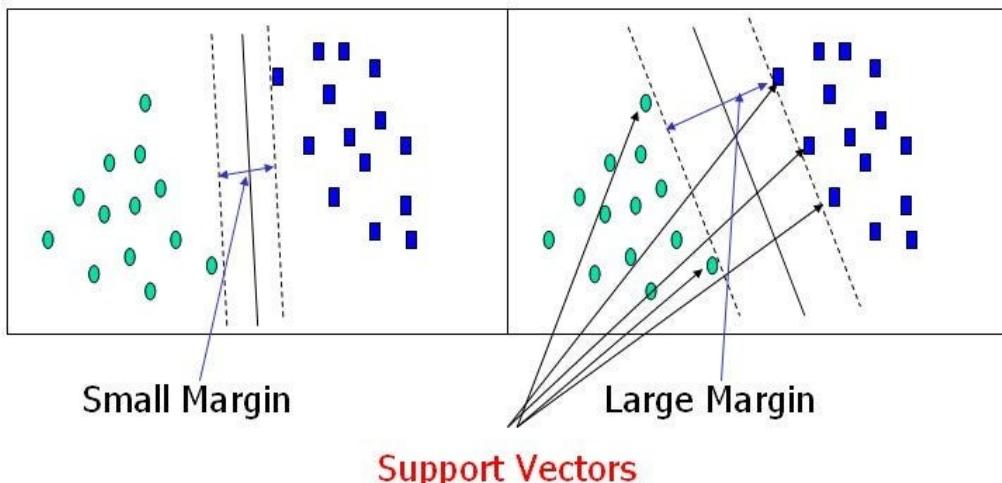


Figura 4.7: Vettori di supporto

I support vector sono i punti dati più vicini all'iperpiano e influenzano la posizione e l'orientamento di quest'ultimo. Impiegando i support vector, massimizziamo il margine del classificatore. L'eliminazione dei support vector cambierà la posizione dell'iperpiano.

Intuizione del grande margine

Nella regressione logistica, prendiamo l'output della funzione lineare e normalizziamo il valore nell'intervallo [0,1].

Se il valore normalizzato è maggiore di un valore soglia (ad es. 0,5), gli assegniamo una etichetta 1, altrimenti gli assegniamo un’etichetta 0.

Nell’SVM, prendiamo l’output della funzione lineare e, se quell’output è maggiore di 1, lo identifichiamo con una classe, mentre se l’output è -1, lo identifichiamo con un’altra classe.

Poiché i valori della soglia sono cambiati in 1 e -1 nell’SVM, otteniamo questo intervallo ($[-1,1]$) che viene utilizzato come margine.

Funzione di costo e aggiornamenti del gradiente

Nell’algoritmo SVM, cerchiamo di massimizzare la distanza tra i punti dati e l’iperpiano. La funzione di perdita che ci aiuta a massimizzare la distanza è la hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases} \quad c(x, y, f(x)) = (1 - y * f(x))_+$$

Figura 4.8: Funzione di costo e di aggiornamento del gradiente

Il costo è 0 se il valore previsto e il valore effettivo hanno lo stesso segno. In caso contrario, calcoliamo il valore di perdita. Aggiungiamo anche un parametro di regolarizzazione alla funzione di costo. L’obiettivo del parametro di regolarizzazione è di bilanciare la massimizzazione della distanza con la perdita.

Dopo aver aggiunto il parametro di regolarizzazione, la funzione di costo appare come in 4.12.

$$\min_w \lambda \|w\|^2 + \sum_{i=1}^n (1 - y_i \langle x_i, w \rangle)_+ \quad (4.12)$$

Ora che abbiamo ottenuto la funzione di perdita, utilizziamo le derivate parziali rispetto ai pesi per trovare i gradienti.

Così facendo, possiamo aggiornare i pesi.

$$\frac{\delta}{\delta w_k} \lambda \|w\|^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} (1 - y_i \langle x_i, w \rangle)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Figura 4.9: Aggiornamento dei pesi

Quando non vi è alcuna errata classificazione, ovvero il nostro modello prevede correttamente la classe del nostro punto dati, sarà necessario aggiornare solo il

gradiente dal parametro di regolarizzazione come in 4.13.

$$w = w - \alpha \cdot (2\lambda w) \quad (4.13)$$

Quando ci si trova di fronte ad una errata classificazione, ossia il nostro modello commette un errore sulla previsione della classe del nostro punto dati, includiamo la perdita insieme al parametro di regolarizzazione per eseguire l'aggiornamento del gradiente, come in 4.14.

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w) \quad (4.14)$$

4.1.5 Support Vector Regression

Le Support Vector Machines (SVMs) vengono spesso adoperate per i problemi di classificazione; tuttavia, il loro utilizzo nella regressione non è altrettanto documentato. Questo tipo di modelli è noto come Support Vector Regression (SVR).

La Regressione Lineare Semplice

Nella maggior parte dei modelli di regressione lineare l'obiettivo è minimizzare la somma degli errori quadratici.

Prendiamo ad esempio il metodo dei minimi quadrati ordinari (OLS).

La funzione obiettivo per OLS con un singolo predittore (feature) è riportata a 4.15:

$$\min \left(\sum_{i=1}^n (y_i - w_i x_i)^2 \right) \quad (4.15)$$

dove y_i rappresenta il valore target, w_i è il coefficiente e x_i è il predittore (feature).

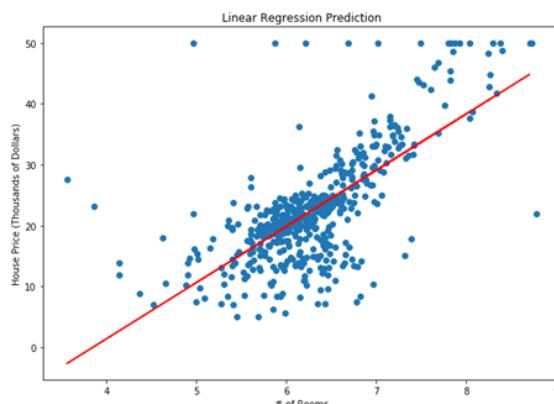


Figura 4.10: Grafico relativo

Lasso, Ridge e ElasticNet sono tutte estensioni di questa semplice equazione, con un parametro di penalità aggiuntivo che mira a minimizzare la complessità e/o ridurre il numero di feature utilizzate nel modello finale.

Tuttavia, l’obiettivo, come per molti modelli, è quello di ridurre l’errore sul set di test.

Ciononostante cosa succede se ci preoccupiamo solo di ridurre l’errore entro un certo grado? Cosa succede se non ci importa quanto siano grandi i nostri errori, purché rientrino in un intervallo accettabile?

Prendiamo ad esempio i prezzi delle case. Cosa succede se decidiamo che la previsione debba essere entro una determinata somma di denaro, ad esempio \$5.000?

Possiamo quindi dare al nostro modello una certa flessibilità nel trovare i valori previsti, purché l’errore rientri in quel range.

E’ proprio in questo momento che entra in gioco la Support Vector Regression. SVR ci offre la flessibilità di definire quanto margine di errore è accettabile nel nostro modello e troverà una linea appropriata (o iperpiano in dimensioni superiori) per adattarsi ai dati.

La funzione obiettivo degli SVR è quella di minimizzare i coefficienti, più precisamente la norma l2 del vettore dei coefficienti, non l’errore quadratico.

Il termine di errore viene invece gestito nei vincoli, dove viene impostato l’errore assoluto come minore o uguale a una determinata soglia, chiamata errore massimo, ϵ . Possiamo tarare epsilon per ottenere la precisione desiderata del nostro modello. La nostra nuova funzione obiettivo e i vincoli sono i seguenti:

Funzione obiettivo:

$$\min \left(\frac{1}{2} \|w\|^2 \right) \quad (4.16)$$

Vincolo:

$$|y_i - w_i x_i| \leq \epsilon \quad (4.17)$$

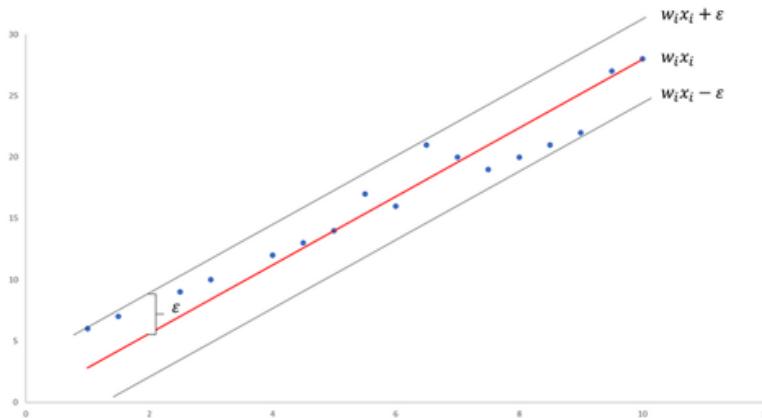


Figura 4.11: Esempio illustrativo

Il grafico 4.12 mostra i risultati di un modello SVR addestrato sul dataset dei prezzi delle case a Boston. La linea rossa rappresenta la linea di miglior adattamento e le linee nere rappresentano la fascia di errore, ϵ , che abbiamo impostato a 5 (\$5.000).

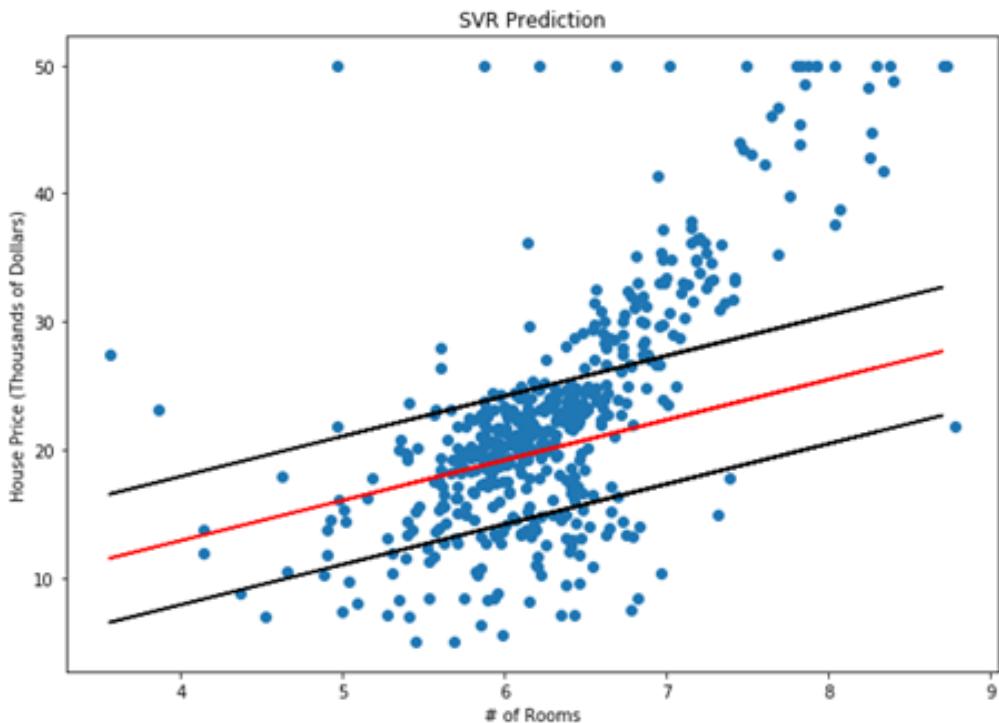


Figura 4.12: Risultati di un modello SVR addestrato sul dataset dei prezzi delle case a Boston

Si può rapidamente notare che questo algoritmo non funziona per tutti i data points.

L'algoritmo ha risolto la funzione obiettivo nel miglior modo possibile, ma alcuni punti si trovano ancora al di fuori delle fasce di errore.

Pertanto, dobbiamo considerare la possibilità di errori che sono più grandi di ϵ . E' possibile effettuare ciò utilizzando le variabili di slack.

Slack (e un altro iperparametro)

Il concetto di variabili di slack è semplice: per ogni valore che cade al di fuori di ϵ , possiamo indicare la sua deviazione dalla fascia come ξ .

Sappiamo che queste deviazioni possono esistere, ma vorremmo comunque ridurle al minimo. Pertanto, possiamo aggiungere queste deviazioni alla funzione obiettivo (4.18 e 4.19).

Funzione obiettivo:

$$\min \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^n |\xi_i| \right) \quad (4.18)$$

Vincolo:

$$|y_i - w_i x_i| \leq \epsilon + |\xi_i| \quad (4.19)$$

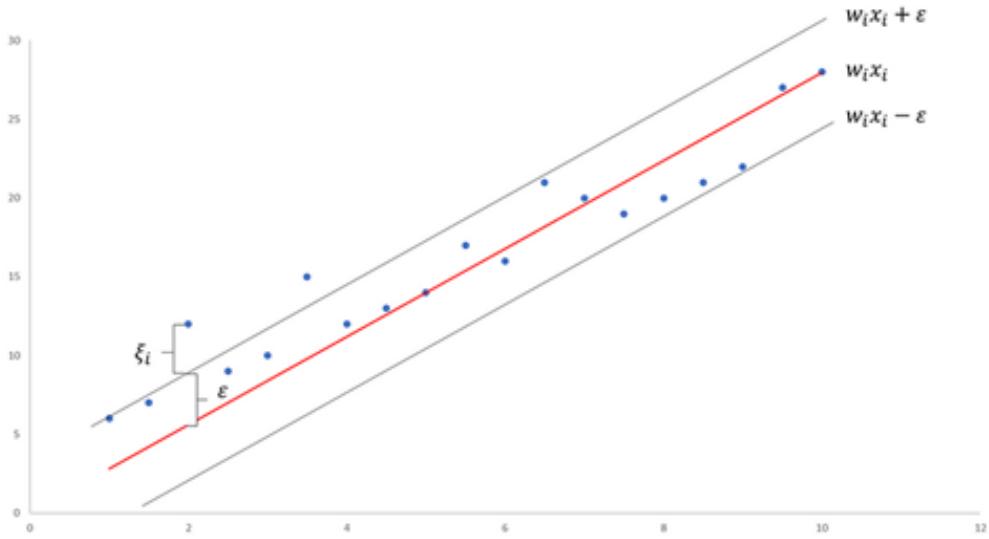


Figura 4.13: Esempio illustrativo

Ora abbiamo un ulteriore iperparametro, C , che possiamo tarare. All'aumentare di C , aumenta anche la tolleranza per i punti al di fuori di ϵ .

4.2 Random forest classifier

Per effettuare delle predizioni sul dataset ho realizzato un classificatore random forest sul quale effettuare delle query, fornendogli i dati riguardanti le Action Units da nuove immagini, sempre attraverso la libreria py-feat.

Per la creazione del classificatore ho in primis letto il file csv contenente il dataset pre-elaborato (modifiche precedentemente trattate e sulle quali aggiungerò considerazioni, in quanto è risultato efficace applicarne ulteriori per migliorare la precisione del predittore creato), rimosso le colonne non necessarie e, infine, ho diviso il dataset in set di addestramento e di test usando la funzione `train_test_split` della libreria `sklearn.model_selection`; tramite l'output di questo metodo ho ricavato i pandas's dataframes `Xtrain, Xtest, yTrain, yTest`.

Questo è il metodo relativo:

```
def getXtrainYTrainXtestYTest():
    pd.read_csv(join(dirname(
        abspath(__file__)),
        "../final analysis/" +
        "DAiSEE and student engagement dataset clean sampled.csv"
    ))
    )
    y = df['label']
    X = df.drop(
        ["input", "naturalLanguageDescription", "label", "numLabel"],
        axis=1
    )

    Xtrain, Xtest, yTrain, yTest = train_test_split(
        X,
        y,
        test_size=0.2,
        random_state=42
    )

    return Xtrain, yTrain
```

Una volta ottenuti questi dataset ho creato il classificatore utilizzando l'oggetto a disposizione fornito dalla libreria `sklearn.ensable`.

Il RandomForestClassifier viene generato con 100 alberi di decisione e viene addestrato con l'utilizzo dei due dataframe `Xtrain` e `yTrain` restituiti dalla funzione `getXtrainYTrainXtestYTest()`.

Il metodo è il seguente:

```
randomForestClassifier = RandomForestClassifier(
    n_estimators=100,
    verbose=True,
    random_state=42
)
Xtrain, yTrain = getXtrainYTrainXtestYTest()
randomForestClassifier.fit(Xtrain, yTrain)
return randomForestClassifier
```

Eseguendo dei test attraverso il metodo `score(Xtest, yTest)` offerto dal random forest classifier generato dalla libreria `sklearn.ensemble` ho potuto calcolare l'accuracy del modello da me generato.

I risultati di questa analisi sono che il random forest classifier ottiene una precisione dell'82,03% sul dataset di test generato nel metodo `getXtrainYTrainXtestYTest`.

Oltre alla produzione del classificatore viene anche generato un grafico per la visualizzazione dell'influenza di ognuna delle label sulla predizione, riportato all'immagine 4.14.

Come è possibile constatare in 4.14, quasi tutte le label influenzano la predizione in modo simile (in un range fra il 7,2% e l'11,9%) tranne per la AU43 (Eyes Closed) che, ovviamente, influenza largamente la predizione effettuata dal modello.

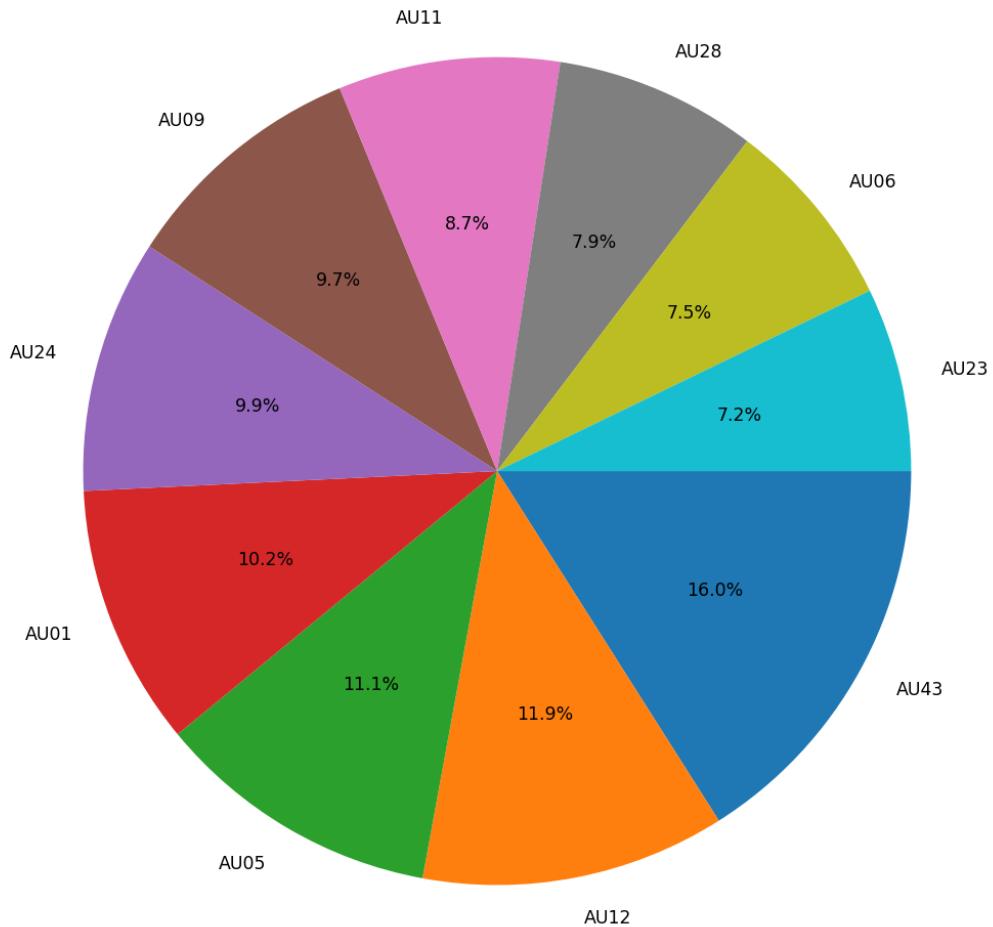


Figura 4.14: Feature con maggiore importanza per il Random Forest

4.3 K-nearest Neighbors Classifier

Un ulteriore classificatore realizzato per effettuare predizioni sul dataset è il K-nearest Neighbors Classifier, attraverso il quale ho effettuato delle query fornendo i dati delle Action Units da nuove immagini.

Come per il Random Forest classifier, per la creazione del classificatore ho in primis letto il file csv contenente il dataset pre-elaborato, rimosso le colonne non necessarie e, infine, ho diviso il dataset in set di addestramento e di test usando la funzione `train_test_split` della libreria `sklearn.model_selection`; tramite l'output di questo metodo ho ricavato i pandas's dataframes `Xtrain`, `Xtest`, `yTrain`, `yTest`.

Il metodo sopracitato per l'estrazione di questi dataframe è lo stesso riportato in pochi paragrafi precedenti.

Una volta ottenuti questi dataset ho creato il classificatore utilizzando l'oggetto a disposizione fornito dalla libreria `sklearn.neighbors`.

Il K-nearest Neighbors Classifier viene creato impostando il parametro relativo al numero di elementi vicini da utilizzare settato a 1 e viene addestrato con l'utilizzo dei due dataframe `Xtrain` e `yTrain` restituiti dalla funzione `getXtrainYTrainXtestYTest()`.

Il metodo è il seguente

```
KnnClassifier = KNeighborsClassifier(n_neighbors=1)
Xtrain, yTrain = getXtrainYTrainXtestYTest()
KnnClassifier.fit(Xtrain, yTrain)
return KnnClassifier
```

Ho ritenuto opportuno valorizzare il campo con 1, in quanto è risultato il quantitativo necessario per non ottenere delle rilevazioni “ballerine” all'interno dell'interfaccia grafica da me realizzata e, allo stesso tempo, avere il miglior valore di accuracy possibile, calcolato come mostrato successivamente.

Ho poi eseguito il calcolo dell'accuracy delle predizioni sul dataframe di test realizzato attraverso il seguente codice:

```
from sklearn.metrics import accuracy_score
yPred = KnnClassifier.predict(Xtest)
accuracy = accuracy_score(yTest, yPred)
print("Accuracy: ", accuracy)
```

ed ho ottenuto un'accuracy del 78,54%.

Il classificatore K-nearest neighbors (KNN) determina l'importanza delle caratteristiche basandosi sulla loro influenza sulla metrica di distanza utilizzata per calcolare la vicinanza tra le istanze.

Più le istanze sono vicine, più sono simili. Pertanto, le caratteristiche più importanti sono quelle che contribuiscono maggiormente al calcolo della distanza.

Un modo per visualizzare l'importanza delle caratteristiche per un classificatore KNN è quello di utilizzare una heatmap della matrice di correlazione a coppie delle caratteristiche.

Le caratteristiche con correlazioni elevate avranno un impatto minore sul calcolo della distanza, mentre le caratteristiche non correlate riporteranno l'effetto contrario.

Il seguente codice viene utilizzato per disegnare una heatmap della matrice di correlazione delle caratteristiche:

```
def visualizeHeatMapCorrelationMatrix(Xtrain):
    corr = Xtrain.corr()
    sns.heatmap(corr, cmap='coolwarm')
    plt.show()
```

Il colore di ogni quadrato nella heatmap 4.15 rappresenta la correlazione tra due caratteristiche.

Le caratteristiche altamente correlate saranno vicine tra loro nella heatmap, mentre le caratteristiche non correlate risulteranno distanti.

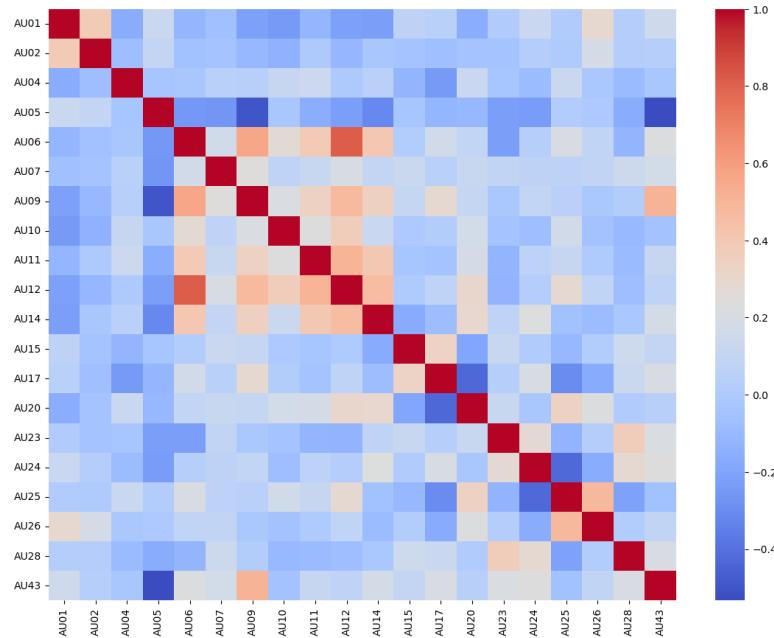


Figura 4.15: Heatmap modello KNN generato

Dalla heatmap riportata è possibile dedurre che, oltre all'ovvia correlazione di ogni Action Units a se stessa, ad esempio, la AU1 e la AU2 sono strettamente correlate, o ancora, AU12 e AU6 sono ancora più strettamente correlate, indi, influenzano particolarmente le predizioni risultanti.

4.4 Support vector machine Classifier

Un altro classificatore adoperato per effettuare delle predizioni sul dataset è basato sull'algoritmo Support vector machine.

La modalità di realizzazione è molto simile a quella per i precedenti classificatori, con modalità di utilizzo identica al Random Forest classifier.

Viene sempre letto in memoria il dataset pre elaborato e resamplato, con successiva rimozione delle colonne non necessarie e viene suddiviso in set di addestramento e set di test.

L'oggetto che viene creato per effettuare le predizioni proviene sempre dalla libreria sklearn, in questo caso il modulo `sklearn.svm` e la sua relativa classe `SVC`.

Eseguendo dei test attraverso il metodo `score(Xtest, yTest)` offerto dalla classe SVC ho potuto calcolare l'accuracy del modello da me generato che risulta essere del 55%.

4.5 Naive Bayes Classifier

Un ulteriore classificatore di cui mi servo per effettuare delle predizioni sul dataset è basato sull'algoritmo Naive Bayes.

La modalità di realizzazione è molto simile a quella per i precedenti classificatori e la modalità di utilizzo è pari a quella del Random Forest classifier.

Viene sempre letto in memoria il dataset pre elaborato e resamplato, vengono rimosse le colonne non necessarie e viene suddiviso in set di addestramento e set di test.

L'oggetto che viene creato per effettuare le predizioni proviene sempre dalla libreria sklearn, in questo caso il modulo `sklearn.naive_bayes` e la sua relativa classe `MultinomialNB`.

Eseguendo dei test attraverso il metodo `score(Xtest, yTest)` offerto dalla classe `MultinomialNB` ho potuto calcolare l'accuracy del modello da me generato che risulta essere del 40%

Il 40% di accuracy è il valore più alto che sono riuscito ad ottenere modificando i diversi valori presi in input dal costruttore di `MultinomialNB`.

4.6 Support Vector Regression Classifier

Un altro classificatore adoperato per effettuare delle predizioni sul dataset è basato sull'algoritmo Support Vector Regression.

La modalità di realizzazione è molto simile a quella per i precedenti classificatori, con utilizzo identico al K-nearest Neighbors Classifier.

Viene sempre letto in memoria il dataset pre elaborato e resamplato, con successiva rimozione delle colonne non necessarie e viene suddiviso in set di addestramento e set di test.

L'oggetto che viene creato per effettuare le predizioni proviene sempre dalla libreria sklearn, in questo caso il modulo `sklearn.svm` e la sua relativa classe `SVR`.

Eseguendo dei test attraverso il metodo `score(Xtest, yTest)` offerto dalla classe SVC ho potuto calcolare l'accuracy del modello da me generato che risulta essere del 34%.

4.7 Quale è il miglior modello predittivo?

Ho collezionato diverse metriche per valutare quale dei modelli predittivi sviluppati fosse il migliore:

4.7.1 Accuracy

La metrica dell'accuracy è una delle metriche più comuni utilizzate per valutare la performance dei modelli predittivi. Essa misura la percentuale di predizioni corrette rispetto al numero totale di predizioni effettuate. Ad esempio, se un modello ha effettuato 100 predizioni e 80 di queste sono corrette, l'accuracy è del 80%.

L'accuracy è una metrica particolarmente utile in caso, similmente al mio, di classi bilanciate. Ciò significa che il numero di esempi per ogni classe di output è simile o pari.

In questo contesto, può fornire una valutazione equilibrata della capacità del modello di predire entrambe le classi in modo corretto.

Tuttavia, se le classi non dovessero essere bilanciate, tale metrica potrebbe risultare fuorviante, dal momento che il modello potrebbe essere molto accurato nella predizione della classe maggioritaria ma non essere in grado di predire la classe di minoranza in modo corretto.

Un'altra considerazione importante quando si utilizza l'accuracy è il tipo di errore che si vuole minimizzare. Ad esempio, se il costo di un falso positivo è molto alto, allora un modello che massimizza l'accuracy potrebbe non essere la scelta migliore.

In formule, l'accuracy si calcola come in 4.20

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \quad (4.20)$$

dove TP indica il numero di veri positivi, TN indica il numero di veri negativi, FP indica il numero di falsi positivi e FN indica il numero di falsi negativi.

4.7.2 Precision

La metrica di precisione è una delle metriche più comuni utilizzate per valutare le prestazioni di un modello predittivo. La precisione è definita come la proporzione di predizioni corrette rispetto al totale delle predizioni fatte dal modello. In altre parole, la precisione misura quanto spesso il modello predittivo fornisce una risposta corretta rispetto al totale delle risposte fornite. È una misura di quanto preciso sia il modello nel predire le classi corrette.

Per calcolare la precisione di un modello predittivo occorre confrontare le predizioni effettuate dal modello con le etichette di classe corrette. La precisione è calcolata dividendo il numero di predizioni corrette per il totale delle predizioni effettuate dal modello. Ad esempio, se un modello predittivo ha effettuato 100 predizioni e 80 di queste predizioni sono corrette, la precisione del modello è del 80%.

La metrica precision è definita come il rapporto tra il numero di veri positivi (TP) e il numero di tutti i casi predetti positivi (TP + FP) a 4.21. In altre parole, la precisione è la percentuale di volte in cui il modello ha predetto correttamente una classe positiva rispetto a tutte le volte in cui ha predetto quella classe.

Pur considerandosi utile, presenta alcune limitazioni. In particolare, la precisione non tiene conto dei falsi negativi, ovvero dei casi in cui il modello ha sbagliato a predire una classe positiva. Inoltre può essere influenzata dal numero di casi positivi e negativi presenti nel set di dati.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (4.21)$$

4.7.3 Recall

La metrica recall è una misura impiegata per la valutazione della capacità di un modello predittivo di identificare correttamente i veri positivi tra tutti i positivi effettivi. In altre parole, la recall comunica quanti dei casi positivi presenti in un insieme sono stati correttamente individuati dal modello.

Per calcolare la recall si divide il numero di veri positivi per la somma dei veri positivi e dei falsi negativi come in 4.22. In questo modo otteniamo una percentuale che fa emergere quanti dei casi positivi sono stati individuati dal modello.

Una delle principali ragioni dell'importanza di tale metrica è che ci permette di valutare la capacità del modello di individuare correttamente i casi positivi, anche se questo può comportare l'avere un alto numero di falsi positivi.

Un altro aspetto da considerare quando si utilizza la recall come metrica è che può essere influenzata dal bilanciamento delle classi. Ad esempio, se abbiamo un insieme di dati in cui la maggior parte dei casi sono negativi, potrebbe essere difficile ottenere una buona recall anche con un modello altamente performante.

Infine, è importante notare che la recall non ci comunica nulla sulla capacità del modello di classificare correttamente i casi negativi. Pertanto, è sempre importante considerare anche altre metriche come la precisione e l'accuratezza complessiva del modello quando si valutano le prestazioni complessive.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4.22)$$

4.7.4 Balanced Accuracy

La metrica di Balanced Accuracy (o accuratezza bilanciata), in formula a 4.23, è una misura di valutazione della performance di un modello predittivo che tiene conto della distribuzione bilanciata delle classi target all'interno del dataset di test. In altre parole, è una misura che considera l'accuratezza del modello in modo equilibrato per entrambe le classi target, evitando così di favorire una classe rispetto all'altra.

Per calcolare la metrica di Balanced Accuracy, si prende in considerazione la media aritmetica tra l'accuratezza della classe positiva (TPR) e quella della classe negativa (TNR), ovvero: $\text{Balanced Accuracy} = (\text{TPR} + \text{TNR}) / 2$

Dove TPR (True Positive Rate) rappresenta la proporzione di veri positivi rispetto a tutti i casi positivi e TNR (True Negative Rate) rappresenta la proporzione di veri negativi rispetto a tutti i casi negativi.

In caso di dataset sbilanciati, in cui una classe target è più rappresentata dell'altra, la metrica di Balanced Accuracy risulta essere più informativa rispetto alla semplice accuratezza (Accuracy) in quanto fornisce una valutazione più equilibrata delle performance del modello per entrambe le classi target.

$$\text{Balanced Accuracy} = \frac{1}{2} \left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}} \right) \quad (4.23)$$

4.7.5 K-fold Cross-Validation

La k-fold cross validation è una tecnica comunemente utilizzata nell'apprendimento automatico e nella valutazione dei modelli predittivi; quest'ultima mira a suddividere il set di dati disponibile in k sottoinsiemi (fold) pari in termini di dimensione e simili in termini di distribuzione dei dati.

Questo metodo si propone di valutare le prestazioni di un modello predittivo servendosi di un set di dati limitato, consentendo di giudicare il comportamento del modello proponendogli nuovi dati e aiutandolo a stimare l'accuratezza delle previsioni sulle nuove istanze.

Nel processo di validazione il set di dati viene diviso casualmente in k sottoinsiemi di dimensioni simili: solitamente k viene scelto come un numero intero positivo come 5 o 10, ma può variare a seconda del caso; ad esempio, per il mio specifico caso di studio, è stata effettuata una 33-fold validation.

Durante ogni iterazione della cross validation, uno dei fold viene selezionato come set di test, mentre i restanti $k-1$ fold vengono adottati come set di addestramento.

Questa procedura viene ripetuta k volte, durante la quale ciascun fold viene utilizzato esclusivamente una volta come set di test. Al termine delle k iterazioni, si ottengono k misure di valutazione delle prestazioni del modello.

Una volta completate, tutti i risultati delle k iterazioni possono essere combinati per ottenere una stima più robusta delle prestazioni del modello. Le misure comunemente utilizzate includono l'accuratezza, la precisione, il recall e l'F1-score.

La k -fold cross validation risulta particolarmente utile quando si dispone di un set di dati limitato e si desidera massimizzare l'utilizzo dei dati disponibili per la valutazione del modello, consentendo così di sfruttare al meglio i dati in modo da ottenere stime più affidabili delle prestazioni del modello.

Un vantaggio importante che la tecnica offre è che ogni istanza del set di dati viene sfruttata sia per l'addestramento che per la valutazione. Ciò permette di ridurre la varianza nella stima delle prestazioni del modello rispetto a una singola divisione di addestramento/test.

La scelta del valore di k può influenzare le prestazioni della cross validation; valori più grandi di k possono ridurre la varianza nella stima delle prestazioni, contemporaneamente aumentando il costo computazionale. Valori più piccoli di k possono portare invece a stime più instabili.

È importante ricordare che essa non può compensare le limitazioni intrinseche del set di dati. Se il set di dati è di scarsa qualità o contiene un forte bias, la cross validation non può garantire prestazioni migliori; è sempre consigliabile quindi eseguire una pulizia dei dati annessa ad una pre-elaborazione adeguata prima di applicare la k -fold cross validation.

La k -fold cross validation trova ampio utilizzo nella pratica dell'apprendimento automatico per valutare e confrontare diversi modelli. È particolarmente utile quando si tratta di selezionare il miglior modello tra diverse opzioni o quando si desidera confrontare le prestazioni di modelli con diverse configurazioni.

Un'altra considerazione importante da fare quando ci si serve di questa tecnica è la gestione degli insiemi di dati sbilanciati; qualora il set di dati avesse una distribuzione sbilanciata tra le classi, potrebbe essere necessario utilizzare metodi di campionamento stratificato o pesatura delle classi per garantire una valutazione accurata delle prestazioni del modello su tutte le classi.

Inoltre, può essere utilizzata per identificare eventuali problemi di overfitting o underfitting del modello: se le prestazioni sul set di addestramento sono significativamente migliori rispetto quelle del test, potrebbe essere un segnale di overfitting. D'altra parte, se le prestazioni su entrambi i set sono scarse, potrebbe indicare underfitting.

In conclusione, la k -fold cross validation fornisce una stima robusta delle prestazioni del modello utilizzando tutti i dati disponibili e aiuta a selezionare il modello

migliore o ad identificare problemi di overfitting o underfitting. È un’importante pratica da seguire durante lo sviluppo e la valutazione dei modelli predittivi.

4.7.6 Tabella dei risultati

Ho effettuato il calcolo delle metriche riportate nella tabella più sotto dopo aver suddiviso il dataset, utilizzando la 33-cross fold validation. Per effettuare la suddivisione del dataset e il calcolo di queste metriche per ognuno dei dataset generati ho utilizzato il metodo `cross_validate` fornito dalla libreria `sklearn.model_selection`. Ho quindi calcolato e salvato su file i risultati delle analisi per ognuna delle metriche sopracitate:

- accuracy,
- precision weighted,
- recall weighted,
- balanced accuracy

le medie dei risultati ottenuti da ognuno dei classificatori realizzati per ciascuna di queste metriche sono riportate nella tabella 4.1.

Mentre i risultati per ognuno dei dataset sono riportati nell’appendice (capitolo 7).

Come già evidenziato nel primo capitolo riguardante lo stato dell’arte, dalle analisi effettuate in [12] l’algoritmo di random forest risulta essere il più affidabile per effettuare le previsioni su dataset di Action Units col fine di un rilevamento delle emozioni FACS e, come si evince dalla tabella sopra riportata, lo stesso è vero per gli stati d’animo (o moods) che vengono trattati nel mio caso di studio.

Tabella 4.1: Tabella media risultati metriche

	Accuracy	Precision	Recall	Balanced Accuracy
K-Nearest Neighbors	77.87493%	76.43947%	77.87493%	77.87465%
Random forest	82.25838%	81.86107%	82.25838%	82.25838%
Naive bayes	39.16669%	35.96379%	39.16669%	39.16736%
Support vector machine	54.37561%	53.82087%	54.37561%	54.38249%
Support vector regressor	16.62504%	18.92075%	16.62504%	16.6416%

Capitolo 5

Realizzazione interfaccia grafica

Una volta generati i modelli predittivi per effettuare nuove analisi, ho pensato che creare questo modello senza poi poterne usufruire come un applicativo vero e proprio sarebbe risultato fine a sé stesso.

Ho quindi scelto di creare una semplice interfaccia grafica attraverso la quale poter effettuare delle predizioni sullo stato d'animo della persona ripresa da una webcam.

Una volta avviato il programma per l'interfaccia grafica, si presenta una schermata che dà la possibilità di scegliere fra i modelli predittivi creati 5.1

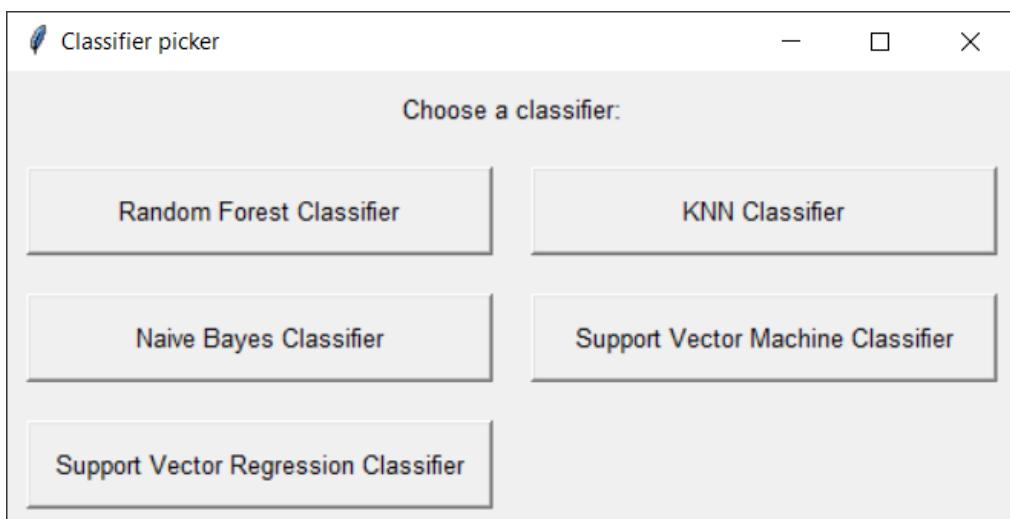


Figura 5.1: Prima schermata interfaccia

Una volta scelto l'algoritmo da utilizzare, questo viene, se precedentemente utilizzato, prelevato dalla memoria, altrimenti viene generato al momento e poi salvato su file binario attraverso i metodi disponibili nella libreria pickle (utilizzata sia per la scrittura che per la lettura di questi file contenenti i modelli).

```
def getRandomForestClassifier():
    filePathRandomForestClassifier = join(
        dirname(abspath(__file__)),
        "randomForestClassifier.pickle"
    )

    if isfile(filePathRandomForestClassifier):
        with open(filePathRandomForestClassifier, "rb") as f:
            randomForestClassifier = pickle.load(f)
    else:
        print("Creazione random forest classifier")
        randomForestClassifier = RandomForestClassifier(
            n_estimators=100,
            verbose=True,
            random_state=42
        )
        Xtrain, yTrain, Xtest, yTest = getXtrainYTrainXtestYTest()
        randomForestClassifier.fit(Xtrain, yTrain)

        with open(filePathRandomForestClassifier, "wb") as f:
            pickle.dump(randomForestClassifier, f)

    relativeTestResult = randomForestClassifier.score(Xtest, yTest)
    print("Relative test result:", relativeTestResult)

    return randomForestClassifier
```

La schermata che si presenta successivamente è riportata a 5.2.

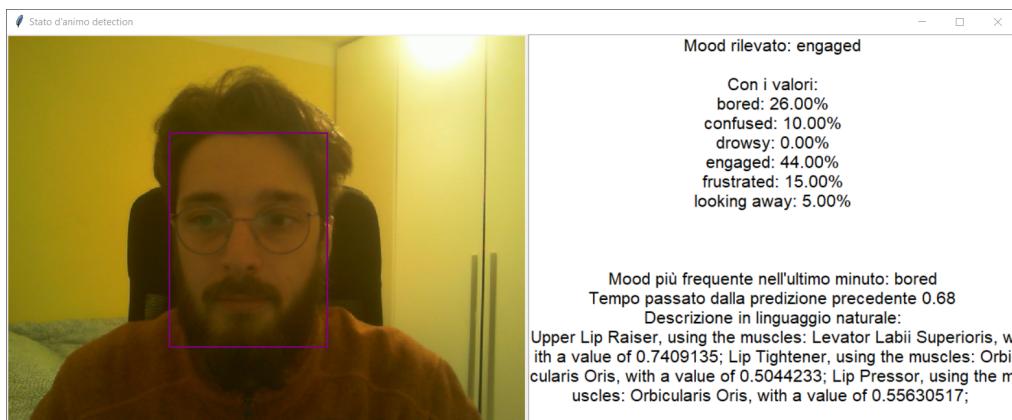


Figura 5.2: Schermata rilevazione mood

Sulla sinistra è possibile notare la webcam dalla quale vengono estratti i frame per

effettuare le analisi e che riprende, ovviamente, il soggetto di fronte alla fotocamera.

Sulla destra è invece possibile osservare il testo, suddiviso in:

La prima riga, dove è presente il mood rilevato, ossia la label con la percentuale di predizione più alta, secondo il modello predittivo precedentemente scelto, sull'immagine che è stata catturata in quel momento.

Subito sotto sono presenti tutte le label e le relative percentuali di predizione calcolate.

È poi riportato il mood più frequente nell'ultimo minuto; per decidere quale label fra quelle estratte vada riportata qui ho immagazzinato in una struttura dati dizionario (dict di python) ognuna delle label, con la relativa percentuale di predizione più alta, raccolte nell'ultimo minuto, e le ho utilizzate come chiave; come valori ho invece immagazzinato il timestamp nel quale è stata effettuata la predizione.

Ogni minuto questo dizionario viene aggiornato rimuovendo le coppie chiave valore prelevate più di un minuto prima.

```
def addToBestClassesLastMinute (bestClass):
    bestClassesLastMinute[time.time()] = bestClass

def removeOldKeys(bestClassesLastMinute):
    currentTime = time.time()
    oneMinuteAgo = currentTime - 60
    return {
        k:v for k,v in bestClassesLastMinute.items() if k > oneMinuteAgo
    }
```

Viene inoltre mostrato quanto tempo è passato fra una predizione e l'altra, così da fornire un'idea delle prestazioni del programma.

Infine, viene mostrata la descrizione in linguaggio naturale generata dall'immagine.

Ovviamente le prestazioni dell'interfaccia variano in base alla macchina sulla quale viene compilata:

il personal computer che ho utilizzato per eseguire il programma, dotato di queste specifiche:

- Ryzen 7 5800H
- 16GB ram DDR4
- GeForce RTX 3060 6GB (mobile)
- SSD m.2

mi ha permesso di ottenere intorno alle 100 predizioni al minuto e, di fatti, il video mostrato nella schermata risulta andare a scatti.

È importante mettere in luce che per effettuare le predizioni l’immagine mostrata sullo schermo viene salvata sul disco, e che le prestazioni dipendono anche dal tipo di disco sul quale viene eseguita l’analisi; ho difatti riscontrato un decremento notevole delle performance nel momento in cui ho provato ad eseguire il programma su un hard disk classico rispetto ad un SSD, tipologia di disco utilizzata in entrambe le macchine sopraccitate.

Un’altra differenza importante è data dal fatto che la macchina presentata è dotata di una scheda video Nvidia che può quindi sfruttare la tecnologia CUDA per effettuare l’estrazione delle Action Units dall’immagine prelevata, il che migliora esaurientemente le prestazioni.

Effettuando delle predizioni impostando il parametro del costruttore della classe Detector, offerta da py-feat, a “cpu” ho rilevato un decremento di performance, che porta il delta tempo fra una predizione e l’altra da poco meno di un secondo ($\sim 0.6/0.7\text{s}$) a $\sim 1.6/1.7$ secondi.

Capitolo 6

Il Framework WoMan per la Generazione dei Modelli Workflow dei Mood

[32] [33] WoMan è un sistema dichiarativo di process mining, in quanto capace di adottare rappresentazioni FOL (First-Order Logic). La FOL fornisce un'ottima espressività, utile per gestire domini complessi che coinvolgono una variabile quantità di oggetti con la possibilità che si verifichino diverse interazioni tra di essi; inoltre offre un unico framework in cui è possibile esprimere e combinare descrizioni di casi, modelli di processo e condizioni sulle attività, sia ai fini dell'apprendimento che dell'attuazione.

Nella FOL, un predicato, scritto con la notazione usuale p/n , esprime una proprietà o una relazione p su n oggetti (chiamati i suoi argomenti).

Un atomo è un predicato p/n applicato a n oggetti t_1, \dots, t_n , scritto come $p(t_1, \dots, t_n)$; esso afferma che la proprietà o la relazione p si limita a quegli oggetti. Qui adotterò la rappresentazione classica della programmazione logica, in cui una virgola tra due atomi è indice della loro congiunzione, e le implicazioni possono essere espresse nella forma $l_0 :- l_1, \dots, l_m$, dove l_0 è la conclusione e l_1, \dots, l_m è la congiunzione delle premesse. Le sezioni seguenti mostrano come WoMan rappresenta e gestisce le informazioni necessarie per applicare le tecniche di process mining a un ambiente intelligente.

6.1 Descrizione dei casi

Il flusso di attività di un caso è espresso in WoMan come una congiunzione di atomi costruiti sui seguenti predicati:

- $\text{activity}(S, T)$: al passo S viene eseguita l'attività T ;

- $\text{next}(S, S')$: il passo S segue il passo S' .

L'argomento T del predicato $\text{activity}/2$ è estrapolato da un insieme (fisso e dipendente dal contesto) di costanti che rappresentano le attività consentite.

I passi, indicati da identificatori univoci, sono associati agli eventi e possono essere implementati come timestamp che indicano gli eventi associati.

Il predicato $\text{next}/2$ consente di rappresentare esplicitamente le esecuzioni parallele nel flusso di attività; ciò evita la necessità di individuare il parallelismo tra le attività mediante considerazioni statistiche.

Di fatti quest'approccio potrebbe risultare fuorviante e quindi indurre in errore il processo di apprendimento del flusso di lavoro.

A partire dal formato a 6 tuple precedentemente introdotto, ogni attività può essere automaticamente tradotta in questo formato.

Ad esempio, l'attività campione per la routine del "pomeriggio" vista in [32] sarebbe espressa come segue:

```
activity(s0,start), next(s0,s1), activity(s1,housekeeping), next(s1,s2), activity(s2,toilet),
next(s2,s3), activity(s3,relax), next(s3,s4), activity(s4,tea), next(s4,s5),
activity(s5,prepare_snack), next(s5,s6), activity(s6,eat_snack), next(s5,s7),
activity(s7,radio),
next(s5,s8), activity(s8,magazine), next(s6,s9), next(s7,s9), next(s8,s9),
activity(s9,stop).
```

Lo step $s0$ è associato all'attività "start", indicativa dell'inizio dell'attività. La prima attività ("housekeeping") è associata allo step $s1$. L'attività "toilet" è associata allo step $s2$ e ha una relazione "next" con "housekeeping" come attività più recentemente completata. Allo stesso modo, una sequenza di attività "relax", "tea" e "prepare_snack" segue, portando allo step $s5$. Quindi, l'attività "eat_snack", associata allo step $s6$, ha una relazione "next" con "prepare_snack". Anche l'attività "radio", associata allo step $s7$, ha una relazione "next" con "prepare_snack" ma non con "eat_snack", che è ancora in esecuzione. Lo stesso vale per "magazine", associato allo step $s8$, che genera complessivamente tre attività concorrenti. Il caso termina al compimento delle tre attività, come indicato dalle relazioni "next" tra i rispettivi passi e lo step $s9$, associato all'attività fittizia "stop" che indica la fine del caso.

Utilizzando ulteriori prediciati dipendenti dal dominio, è possibile aggiungere scorrevolmente ulteriori informazioni di contesto e relazioni relative agli step e alle attività alla descrizione, sostando all'interno del framework FOL.

Mentre i prediciati $\text{activity}/2$ e $\text{next}/2$ sono riservati, è possibile definire e utilizzare qualsiasi altro predicato da parte dell'ingegnere della conoscenza, responsabile di garantire la coerenza nella definizione e nell'uso di questi prediciati, per esprire informazioni di contesto senza richiedere modifiche al sistema; ciò consente un'applicazione generale dell'approccio nei più svariati domini e compiti.

Consideriamo, ad esempio, l'estensione seguente del nostro caso di esempio, la quale esprime le relative informazioni di contesto:

```

early_afternoon(s1), sunny(s1), air_conditioning_on(s1),
early_afternoon(s2), sunny(s2), short_duration(s2), short_interval(s2,s3),
early_afternoon(s3), sunny(s3), air_conditioning_on(s3), long_duration(s3),
mid_afternoon(s4), cloudy(s4), humidity_high(s4),
mid_afternoon(s5), rainy(s5), microwave_on(s5),
late_afternoon(s6), rainy(s6),
late_afternoon(s7), rainy(s7), radio_on(s7), news(s2,n'), about(n',s), sports(s),
bad(n'),
late_afternoon(s8), rainy(s8),

```

Si desume che fosse presto, il tempo fosse soleggiato e l'aria condizionata accesa mentre l'utente svolgeva l'attività di "housekeeping". Era presto e soleggiato, ma l'aria condizionata era spenta anche quando è andato in bagno, dove è rimasto per breve tempo. Dopo un breve intervallo ha iniziato a rilassarsi, con l'aria condizionata accesa e il tempo ancora soleggiato. Il relax è durato a lungo e quando l'utente ha preso il tè nel pomeriggio il tempo era nuvoloso e umido. Ha iniziato a piovere mentre stava preparando uno spuntino nel forno a microonde, dopodiché, nel tardo pomeriggio, la pioggia non accennava a fermarsi e la radio ha dato notizie negative sullo sport.

6.2 Learning Workflow Structure and Weights

Nel sistema WoMan, una struttura di flusso di lavoro è descritta come una congiunzione di atomi costruiti sui seguenti predicati:

- task(t,C): il task t si verifica nei casi C;
- transition(I,O,p,C): la transizione p, che si verifica nei casi C, consiste nel terminare tutti i task in I (che devono essere in esecuzione) e avviare l'esecuzione di nuove istanze di tutti i task in O.

L'argomento C dei due predicati è uno storico che riporta gli identificatori dei casi in cui si è incontrato il task/transizione.

È un multi insieme perché un task/transizione può verificarsi più volte nello stesso caso; esso può essere sfruttato per calcolare statistiche sul loro utilizzo e per decrementare la ridondanza, imponendo che la sequenza di transizioni osservata durante il funzionamento del modello si sia verificata almeno in un caso di addestramento.

Un modello è espresso come un insieme (da interpretare come una congiunzione) di atomi costruiti su questi predicati e viene costruito come riportato nell'Algoritmo all'immagine 6.1.

```

Require:  $\mathcal{W}$ : workflow model
Require:  $c$ : case having FOL description  $D$ 
for all activity  $(s, t) \in c$  do
    if  $\exists$  task  $(t, C) \in \mathcal{W}$  then
         $\mathcal{W} \leftarrow (\mathcal{W} \setminus \text{task}(t, C)) \cup \{ \text{task}(t, C \cup \{c\}) \}$  /* update statistics on task  $t$  */
    else
         $\mathcal{W} \leftarrow \mathcal{W} \cup \{ \text{task}(t, \{c\}) \}$  /* insert new task and initialize statistics */
    end if
    refine_precondition  $(\mathcal{W}, t(s))$ : -  $D|_s$ 
    refine_postcondition  $(\mathcal{W}, t(s))$ : -  $D$ 
end for
for all next  $(s', s'') \in c$  do
     $I \leftarrow \{t' | \text{activity}(s', t') \in c\}$ 
     $O \leftarrow \{t'' | \text{activity}(s'', t'') \in c\}$ 
    if  $\exists$  transition  $(I, O, p, C) \in \mathcal{W}$  then
         $\mathcal{W} \leftarrow (\mathcal{W} \setminus \text{transition}(I, O, p, C)) \cup \{ \text{transition}(I, O, p, C \cup \{c\}) \}$ 
        /* update statistics on transition  $p$  */
    else
         $p \leftarrow \text{generate_jresh.transition\_identifier}()$ 
         $\mathcal{W} \leftarrow \mathcal{W} \cup \{ \text{transition}(I, O, p, \{c\}) \}$ 
        /* insert new transition and initialize statistics */
    end if
end for

```

Figura 6.1: Pseudocodice WoMan

WoMan funziona nel seguente modo: preleva un nuovo caso di addestramento c e lo utilizza per aggiornare un modello di flusso di lavoro esistente, se presente, o per costruire un nuovo modello dal principio, se è il primo caso considerato. Indi, è completamente incrementale. Durante la gestione del caso, crea/aggiorna prima i task nel modello per poi creare/aggiornare le transizioni.

Per quanto riguarda i task, per ogni atomo $\text{activity}(s, t)$ in c , se un atomo $\text{task}(t, C)$ è già presente nel modello corrente, viene sostituito con $\text{task}(t, C \cup c)$, altrimenti viene aggiunto un nuovo atomo $\text{task}(t, c)$ al modello corrente. Per quanto riguarda le transizioni, ogni atomo $\text{next}(s, s')$ in c deve appartenere a una transizione con I e O tali che $s \in I$ e $s' \in O$.

Pertanto, devi selezionare ed eliminare da c gli altri atomi $\text{next}/2$ che si riferiscono alla stessa transizione, ottenendo I e O , e quindi sostituire (se presente) l'atomo $\text{transition}(I, O, p, C)$ con $\text{transition}(I, O, p, C \cup c)$, o creare un nuovo atomo $\text{transition}(I, O, p, c)$, dove p è un nuovo identificatore di transizione, nel caso in cui non esista.

Per quanto riguarda la selezione degli atomi $\text{next}/2$, molti atomi $\text{next}(s, \text{si})$ indicano che l'esecuzione del task associato allo step s è seguita dall'esecuzione parallela dei task associati agli step si , quindi i task associati ai si devono essere inclusi in O .

Allo stesso modo, molti atomi $\text{next}(s_j, s)$ indicano che l'esecuzione di vari task paralleli associati agli step s_j converge nell'esecuzione del task associato allo step s ,

quindi i task associati ai s_j devono essere inclusi in I .

Per ciascun s_i (rispettivamente, s_j), la stessa procedura deve essere iterata per trovare altri atomi $\text{next}(s_h, s_i)$ ($\text{next}(s_j, s_k)$, rispettivamente), e così via fino alla convergenza.

Ciò accade quando è necessario interrompere più task di input paralleli e/o avviare più task di output nello stesso passaggio, una situazione che richiederebbe modelli complessi per essere rappresentati utilizzando le reti di Petri.

Esecuzioni alternative possono emergere dall'analisi di diversi casi che selezionano diverse opzioni di routing nello schema di flusso di lavoro, rappresentate in seguito da diversi atomi $\text{transition}/2$. La rappresentazione proposta implica una gestione fluida e naturale dei casi complessi o problematici che coinvolgono nodi di task invisibili o duplicati, che solitamente sono problematici da rappresentare nelle reti di Petri e da apprendere con le altre tecniche disponibili in letteratura.

Infatti, diversi nodi $\text{transition}/2$ possono combinare un dato task in modi diversi con altri task o ignorare un task quando non è obbligatorio per un passaggio specifico.

Nel complesso, i modelli appresi da WoMan sono un superset stretto di quelli che possono essere espressi dalle reti di Petri; tuttavia, quando un modello WoMan rientra nell'intervallo esprimibile dalle reti di Petri, può essere facilmente e automaticamente tradotto in tale formalismo.

WoMan ha la facoltà di gestire anche il rumore; di fatti, le probabilità delle transizioni sono proporzionali al numero di casi in cui si sono effettivamente verificate, ovvero alla cardinalità del multiset associato di casi: quando viene elaborato un nuovo caso di addestramento, l'aggiornamento di questi multisets comporta anche l'aggiornamento dei pesi come effetto collaterale.

Pertanto, sebbene i casi rumorosi siano effettivamente incorporati nel modello, impostare una tolleranza al rumore $N \in [0,1]$ significa semplicemente ignorare tutte le transizioni la cui frequenza è inferiore a N .

WoMan apprende dagli esempi costituiti da descrizioni complete dei casi, ma è in grado di applicare il modello appreso in modo progressivo finché vengono generati gli eventi di log. La procedura, la cui versione formale dell'algoritmo è riportata in [33], può essere riassunta come segue: le 6-tuple delineanti gli eventi che si verificano nell'ambiente vengono inviate al sistema man mano che vengono generate; il sistema le sfrutta per navigare nel modello, tenendo traccia in ogni momento delle attività ancora in corso.

Quando si verifica un evento “begin activity”, il sistema verifica se una transizione nel modello consente di passare dallo stato corrente alla nuova attività; quando invece si verifica un evento “end activity”, il sistema aggiorna lo stato corrente chiudendo quella attività.

Sulla base di questi controlli, il sistema può rispondere immediatamente in corso di routine , restituendo per ogni evento uno dei seguenti esiti: "ok" (se l'evento è conforme al modello nello stato corrente del flusso di lavoro), "warning" (se l'evento non è conforme al modello nello stato corrente del flusso di lavoro, ma potrebbe

comunque indicare un nuovo ramo della routine da poter utilizzare per perfezionare il modello (a caso concluso), "errore" (se l'evento non è accettabile nello stato corrente del flusso di lavoro - ad esempio, terminare un'attività che non era mai iniziata o chiudere il caso mentre alcune attività sono ancora in corso).

Nel caso di un warning, può specificare se il problema è dovuto a un'attività o transizione inaspettata, consentendo all'ambiente intelligente di intraprendere le opportune contromisure e di mantenere statistiche sui problemi che si sono verificati durante l'esecuzione del flusso di lavoro/routine.

Ciò assicura che WoMan venga, in teoria, applicato in tempo reale e che possa verificare progressivamente, finché gli eventi e le attività si concludono, la loro conformità al modello della routine.

Il sistema può gestire più routine non separate temporalmente, alimentando gli eventi e le attività che si verificano a tutti i modelli di routine attivi. Pertanto, il sistema non distingue a priori quali eventi/attività appartengono a quale routine, ma ogni modello può sfruttare o ignorare questi eventi/attività secondo necessità.

Le routine sono considerate indipendenti l'una dall'altra, indi per cui il sistema non è in grado di imporre alcun tipo di sincronizzazione tra di loro; se tale sincronizzazione è significativa, è necessario apprendere una routine composta che coinvolga esplicitamente tutte le attività delle singole routine.

Anche nella fase di apprendimento, è possibile gestire più routine contemporaneamente, ma in questo caso è necessaria la presenza di un modo per il sistema di capire quali attività devono essere alimentate a quale modello.

6.3 Realizzazione file per l'osservazione e la predizione dei mood attraverso il sistema WoMan

Per la realizzazione di questi file ho proseguito nel seguente modo:

- Ho inizialmente ridotto la pull di mood di due in quanto solo confused, engaged, bored e frustrated sono presenti all'interno del dataset DAiSEE [11] che, a differenza dello student-engagement-dataset [10], il quale presenta solo immagini, è composto solo da video.
- Successivamente ho scomposto il video utilizzando sempre il metodo `detector.detect_video(videoPath)`; come è possibile notare, il parametro `skip_frames` non è valorizzato. Quando questo parametro non è valorizzato il valore diventa `None`, ovvero non viene saltato nessuno dei frame del video. Effettuo quindi un'analisi su ognuno dei frame del video analizzati, rielaboro l'output di queste ed ottengo i file di input per il sistema WoMan.

- Ognuna delle linee dei file di input prodotti è così strutturata: (T, E, W, P, A, O), dove:
 - T è suddiviso in frame in cui avviene l’attivazione della singola action units e numero di utilizzo di questo frame,
 - E è il tipo di evento che sta avvenendo in quel frame (begin of process, end of process, begin of activity, end of activity),
 - W è il mood del video relativo,
 - P è il video che sto adoperando per l’analisi,
 - A è l’Action Unit che è stata attivata o che si è spenta,
 - O è il numero progressivo relativo a quante volte la singola Action Unit si è accesa e spenta (il numero progressivo di occorrenze di A in P)

Genero quindi, ad esempio, stringhe come queste:

```
entry(000 00 , begin_of_process , confused , video987736015 , start , 1) .
entry(00100,begin_of_activity,confused,video987736015,au02,1) .
entry(00101,begin_of_activity,confused,video987736015,au24,1) .
entry(00200,end_of_activity,confused,video987736015,au02,1) .
entry(00201,begin_of_activity,confused,video987736015,au14,1) .
entry(00202,begin_of_activity,confused,video987736015,au23,1) .
entry(00203,end_of_activity,confused,video987736015,au24,1) .
entry(00300,begin_of_activity,confused,video987736015,au07,1) .
entry(00301,end_of_activity,confused,video987736015,au14,1) .
entry(00302,end_of_activity,confused,video987736015,au23,1) .
entry(00303,begin_of_activity,confused,video987736015,au24,2) .
```

Una volta realizzati questi file li ho consegnati al professore di modo che potesse darli in input al sistema.

Ho analizzato 50 file per ognuna delle label all’interno di DAiSEE e il dataset risultante, in formato json, è così strutturato:

```
[  
  {  
    "AU01": {"dizionario con i valori frame per frame"},  
    "AU02": {"dizionario con i valori frame per frame"},  
    "...": "tutte le Action Units utilizzate da py-feat e  
           precedentemente citate"  
    "AU28": {"dizionario con i valori frame per frame"},  
    "AU43": {"dizionario con i valori frame per frame"},  
    "input": {"dizionario con i valori frame per frame"},  
    "label": "la label relativa al video"  
  },  
  "...": {"gli altri valori"}  
]
```

La label per ognuno dei video è stata estratta attraverso il file “AllLabels” contenuto all’interno del dataset DAiSEE; il file è strutturato come segue:

ClipID	Boredom	Engagement	Confusion	Frustration
1100011002.avi	0	2	0	0
1100011003.avi	0	2	0	0
1100011004.avi	0	3	0	0
1100011005.avi	0	3	0	0
1100011006.avi	0	3	0	0
1100011007.avi	1	2	0	0
1100011008.avi	0	3	0	0
1100011009.avi	0	2	1	0
1100011010.avi	0	3	0	0
1100011011.avi	0	3	0	0

È un file con estensione csv che presenta cinque colonne:

- l’identificativo del video, riportato poi anche nella colonna input del mio dataset derivato,
- un valore fra 0 e 3 per la label bored,
- un valore fra 0 e 3 per la label engaged,
- un valore fra 0 e 3 per la label confused,
- un valore fra 0 e 3 per la label frustrated,

Ai fini del mio studio ho deciso di assegnare una singola label ad ognuno dei video in base al valore più alto fra quelli presenti.

Il numero di 50 file per ognuno dei mood è stato scelto in quanto rappresentava il numero migliore in merito al rapporto numero di dati da utilizzare e tempo per generare questi dati:

ogni video di quelli presenti all'interno del dataset [11] è lungo circa 5/6 secondi, i quali diventano quindi fra i 270 e i 300 frame da analizzare; effettuando un test sommario sulla durata dell'analisi, per un video di 300 frames sono stati necessari 192,60754 secondi, sostituendo la media del numero di frame nei video (285) a quelli del video analizzato ottengo un tempo di analisi medio di 182,97716 secondi.

Moltiplicando questo valore per 200 ottengo 10,16539 ore alle quali vanno aggiunte le ore per scrivere su disco ognuna delle analisi e per generare i file con le entry riportate sopra.

C'è poi da considerare la questione di surriscaldamento e relativo thermal throttling; il numero sopra riportato è stato ottenuto dall'esecuzione dell'analisi di un singolo video, è ragionevole pensare che, dopo aver raggiunto una temperatura abbastanza alta, il tempo di esecuzione delle singole analisi aumenti.

Viene da sé che l'analisi di molti altri video non è sostenibile, sia per quanto riguarda il tempo da dedicare a questa operazione sia a livello di costi, come si è poi rivelato necessario e che, da quanto si evince da [33] invece, non sarebbero dovuti essere necessari.

Il tempo effettivo necessario per l'elaborazione dei video è stato di ~ 36 ore.

6.4 Generazione dei modelli workflow attraverso WoMan

Una volta generati i file con le entry, ho fornito questi al professore che ha eseguito il framework dando in input i file risultanti per i video dei singoli mood uniti in singoli file, uno per ogni mood.

Mi sono stati forniti due file di output per ognuno dei mood:

- il primo contiene il numero di task e di transition per ognuno dei casi (o activity) analizzati dal sistema che stanno a segnalare il numero di nuovi task e di transizioni effettuate ad ogni nuovo video analizzato dal sistema per lo specifico workflow (o mood).
- nel secondo è presente invece il modello generato per ogni mood.

Riporto i grafici generati per il primo tipo di file alle immagini 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8, 6.9.

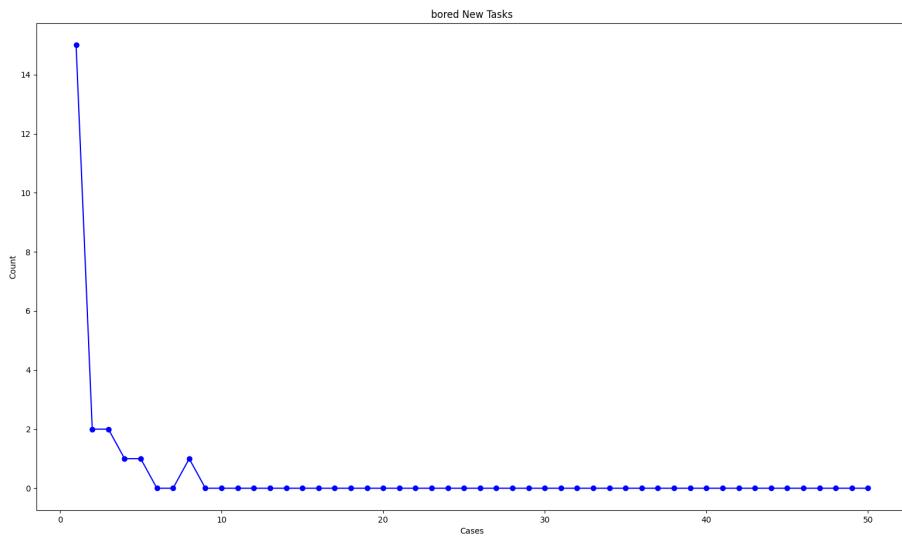


Figura 6.2: Bored New Tasks

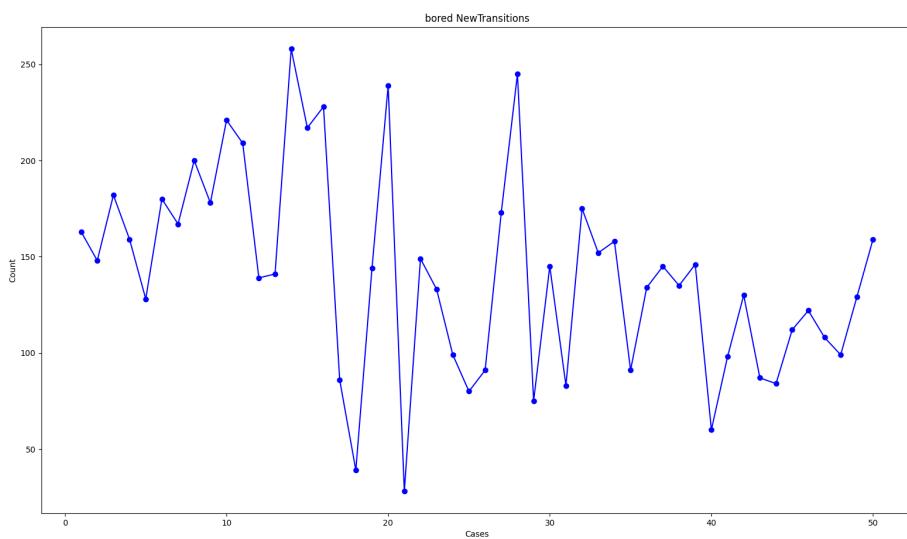


Figura 6.3: Bored New Transitions

6.4 – Generazione dei modelli workflow attraverso WoMan

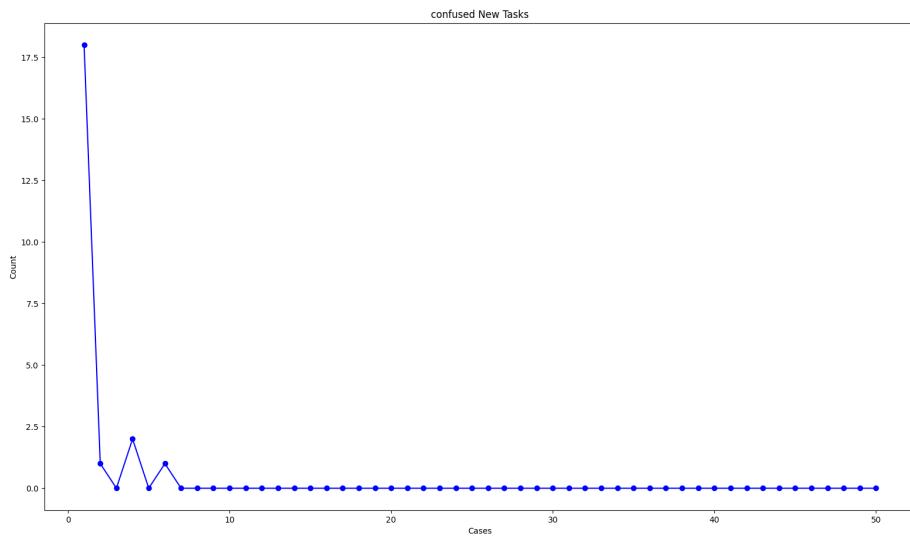


Figura 6.4: Confused New Tasks

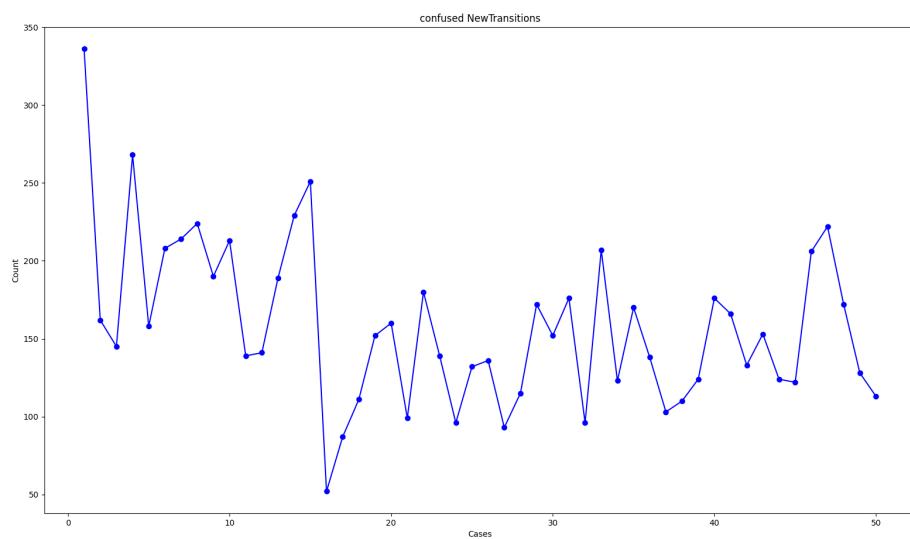


Figura 6.5: Confused New Transitions

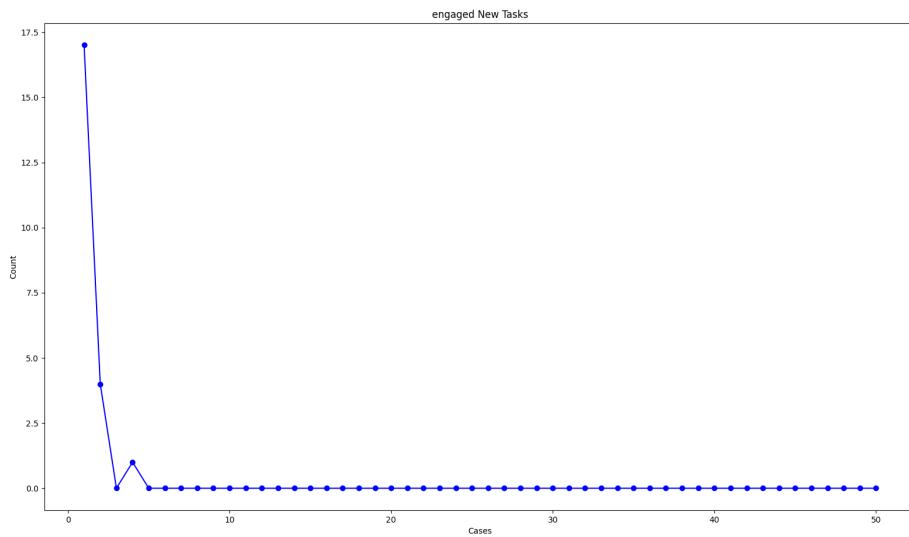


Figura 6.6: Engaged New Tasks

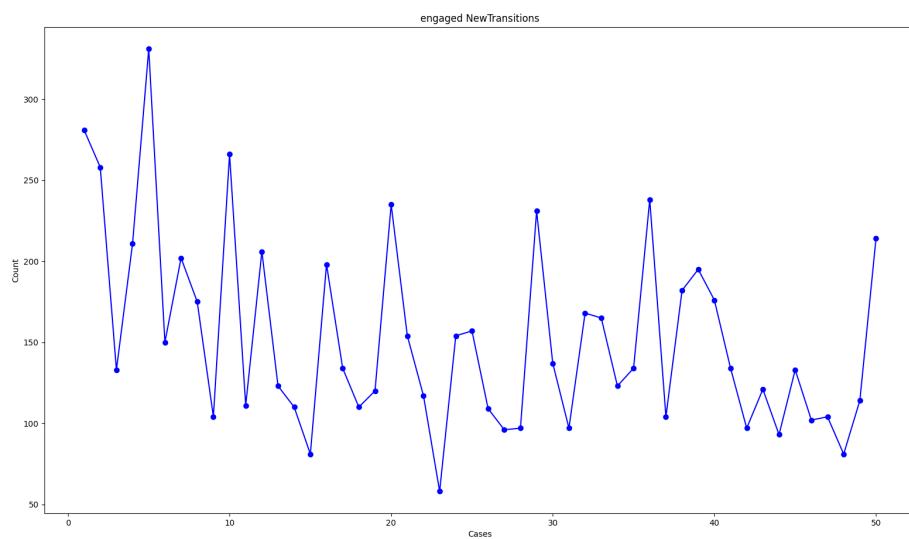


Figura 6.7: Engaged New Transitions

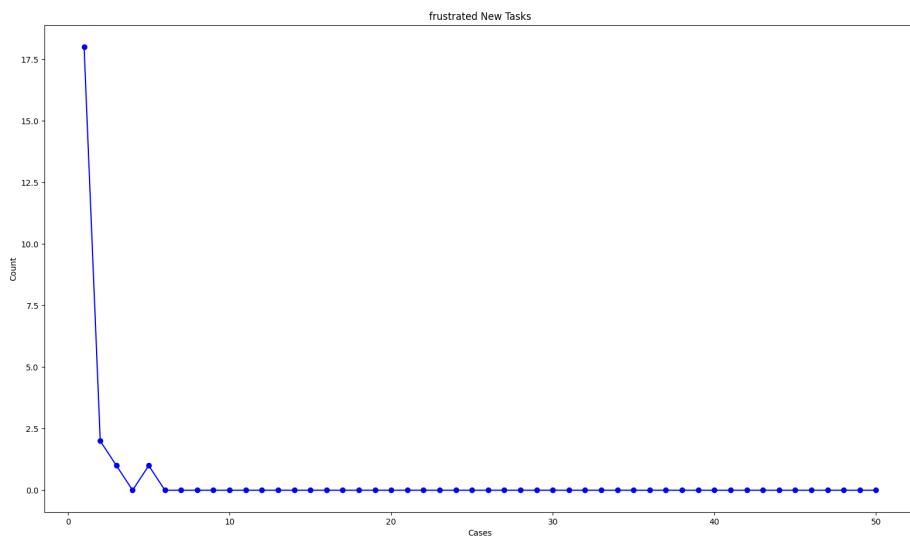


Figura 6.8: Frustrated New Tasks

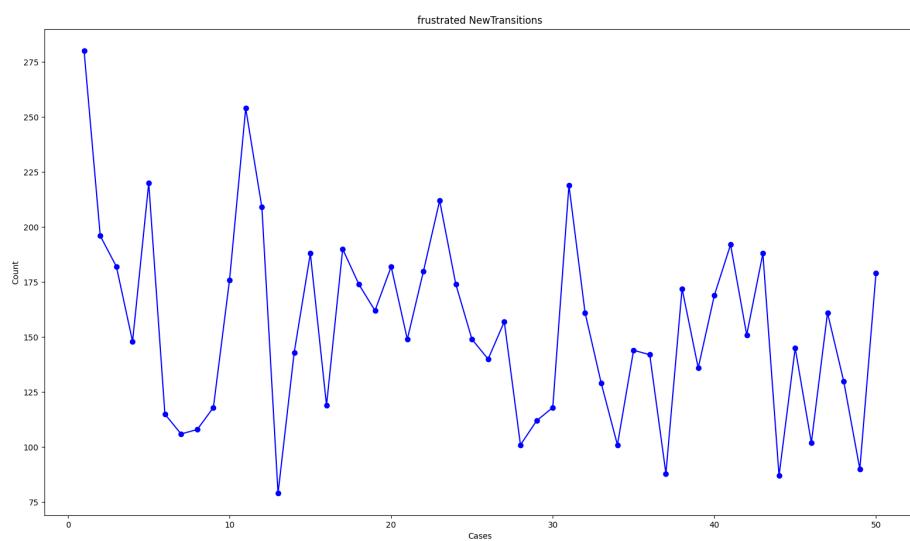


Figura 6.9: Frustrated New Transitions

Come è possibile notare dai grafici:

per quanto riguarda i task, dopo pochi video il numero di nuovi task trovati scende a 0.

Mentre per quanto riguarda le transizioni non c'è una convergenza vera e propria con 50 video ma è presente un accenno di un progressivo abbassamento dei picchi.

Vengono in mente più possibili soluzioni da provare per risolvere questa problematica:

- cambiare dataset per averne uno che contenga dei video nei quali sono presenti solo frame molto specifici in relazione al singolo mood e non anche frame che presentano altre espressioni facciali (che comunque sfociano nel mood categorizzato) come in DAiSEE. Per fare ciò, oltre a trovare un altro dataset con questi dati, si potrebbe, con l'aiuto di un esperto, modificare i video già presenti per ottenere dei video con all'interno solo i frame interessati.
- un altro approccio potrebbe essere fornire un numero maggiore di dati fino ad arrivare a 0 nuove transizioni; per quanto riguarda questo modus operandi ho provato a trovare, attraverso la regressione lineare, una linea che descrivesse l'andamento dei vari grafici tracciati utilizzando, sull'asse delle ascisse, il conto delle transizioni e, sull'asse delle ordinate, il numero di transizioni in quel momento (quindi come nei grafici riportati sopra).

Ho trovato la linea sopracitata e ho generato anche i valori successivi fino ad arrivare al primo valore pari o minore a 0.

Essendo stato rilevato, dal metodo utilizzando, un trend negativo, non sono presenti valori maggiori di 0 dopo il primo valore minore o uguale a 0. È ovvio che questo valore, in uno scenario reale, potrebbe rialzarsi per poi, gradualmente, arrivare a zero e che i dati riportati di seguito non sono affidabili al 100% ma danno solo una vaga indicazione di quanti altri video potrebbero essere necessari.

Il numero di passaggi successivi calcolato è riportato nell'immagine 6.10

```
Per il mood bored è stato calcolato che potrebbero essere necessari altri 96 video o più.  
Per il mood confused è stato calcolato che potrebbero essere necessari altri 75 video o più.  
Per il mood engaged è stato calcolato che potrebbero essere necessari altri 75 video o più.  
Per il mood frustrated è stato calcolato che potrebbero essere necessari altri 185 video o più.
```

Figura 6.10: Numero di video aggiuntivi necessari per ogni mood

Riporto i grafici generati con i nuovi possibili valori trovati nelle immagini 6.11, 6.12, 6.13, 6.14.

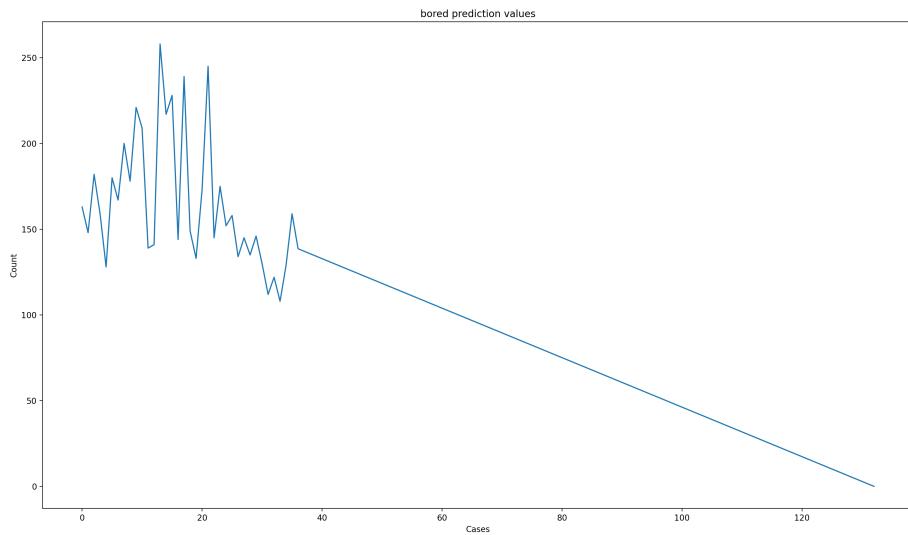


Figura 6.11: Valori predetti per il mood Bored

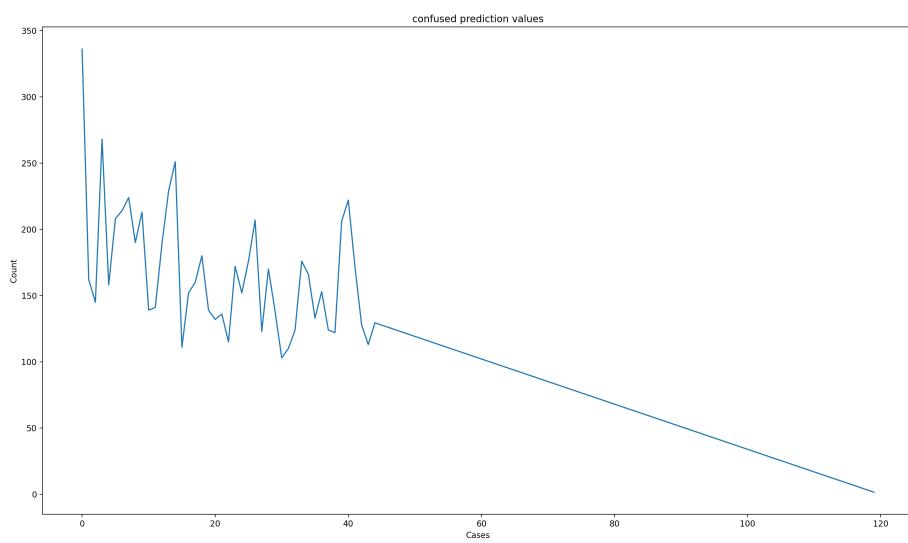


Figura 6.12: Valori predetti per il mood Confused

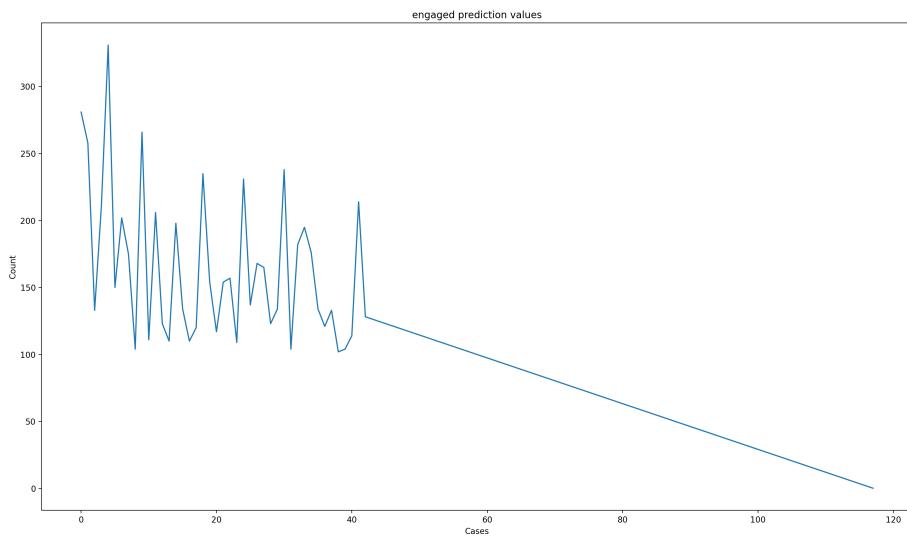


Figura 6.13: Valori predetti per il mood Engaged

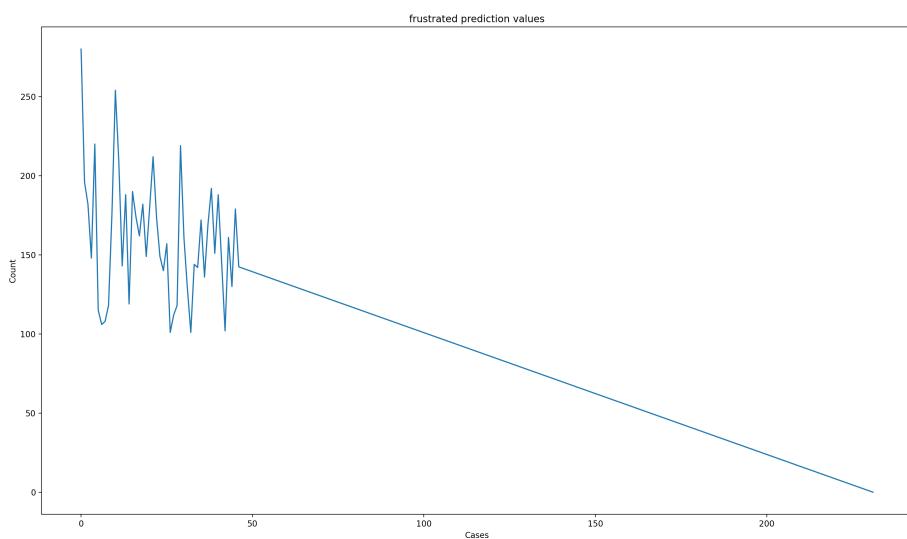


Figura 6.14: Valori predetti per il mood Frustrated

Dalla seconda categoria di file con i modelli generati per ogni mood e dai file da me generati con all'interno le estrazioni create dai video ho estratto invece i dati alla tabella 6.1.

Tabella 6.1: Tabella media risultati metriche

Mood	Numero di attività	Numero di eventi	Numero di task	Numero di transizioni
bored	33103	33153	50	7051
confused	36350	36400	50	7905
engaged	36601	36651	50	7624
frustrated	33407	33457	50	7727

Capitolo 7

Appendice

7.1 K-Nearest Neighbors classifier

Tabella 7.1: Tabella media risultati metriche per K-Nearest Neighbors classifier

Dataset	Accuracy	Precision	Recall	Balanced Accuracy
Dataset 1	79.94505%	78.6269%	79.94505%	80.03188%
Dataset 2	78.57143%	77.05335%	78.57143%	78.65209%
Dataset 3	79.67033%	78.08867%	79.67033%	79.7541%
Dataset 4	79.3956%	78.18508%	79.3956%	79.47632%
Dataset 5	75.54945%	73.73556%	75.54945%	75.62386%
Dataset 6	75.27473%	74.02603%	75.27473%	75.35974%
Dataset 7	77.1978%	76.25371%	77.1978%	77.26776%
Dataset 8	76.37363%	74.56964%	76.37363%	76.30692%
Dataset 9	78.2967%	77.21308%	78.2967%	78.25137%
Dataset 10	77.47253%	76.02751%	77.47253%	77.44991%
Dataset 11	78.2967%	76.63085%	78.2967%	78.28324%
Dataset 12	77.74725%	76.89128%	77.74725%	77.70492%
Dataset 13	78.2967%	76.2142%	78.2967%	78.2377%
Dataset 14	78.02198%	76.48645%	78.02198%	77.9827%
Dataset 15	77.1978%	76.02367%	77.1978%	77.16302%
Dataset 16	78.2967%	76.53138%	78.2967%	78.2377%
Dataset 17	76.64835%	74.81263%	76.64835%	76.57104%
Dataset 18	79.67033%	78.9761%	79.67033%	79.64026%
Dataset 19	78.57143%	76.94415%	78.57143%	78.5337%
Dataset 20	76.92308%	74.85723%	76.92308%	76.86248%
Dataset 21	78.84615%	77.62683%	78.84615%	78.81148%
Dataset 22	77.13499%	75.69965%	77.13499%	77.11749%
Dataset 23	76.8595%	75.28026%	76.8595%	76.86248%
Dataset 24	78.23691%	76.51369%	78.23691%	78.26047%
Dataset 25	78.5124%	78.20728%	78.5124%	78.50182%
Dataset 26	74.10468%	71.6482%	74.10468%	74.09836%
Dataset 27	77.41047%	75.46258%	77.41047%	77.42714%
Dataset 28	77.13499%	75.45398%	77.13499%	77.14481%
Dataset 29	81.26722%	80.7337%	81.26722%	81.27505%
Dataset 30	80.71625%	79.38591%	80.71625%	80.72404%
Dataset 31	78.78788%	77.31627%	78.78788%	78.80237%
Dataset 32	77.13499%	75.2633%	77.13499%	77.14026%
Dataset 33	76.30854%	75.76339%	76.30854%	76.30692%

7.2 Naive Bayes classifier

Tabella 7.2: Tabella media risultati metriche per Naive Bayes classifier

Dataset	Accuracy	Precision	Recall	Balanced Accuracy
Dataset 1	40.65934%	35.66242%	40.65934%	40.69672%
Dataset 2	38.18681%	34.82528%	38.18681%	38.2286%
Dataset 3	36.26374%	33.17577%	36.26374%	36.31148%
Dataset 4	36.81319%	34.33651%	36.81319%	36.86248%
Dataset 5	38.46154%	35.42212%	38.46154%	38.53825%
Dataset 6	35.71429%	30.58777%	35.71429%	35.75592%
Dataset 7	37.91209%	34.2044%	37.91209%	37.94627%
Dataset 8	40.93407%	38.77032%	40.93407%	40.94262%
Dataset 9	41.20879%	42.38325%	41.20879%	41.25683%
Dataset 10	38.73626%	37.00685%	38.73626%	38.73862%
Dataset 11	39.56044%	36.76121%	39.56044%	39.59472%
Dataset 12	43.40659%	42.6477%	43.40659%	43.42896%
Dataset 13	39.83516%	35.08367%	39.83516%	39.86794%
Dataset 14	36.26374%	32.28899%	36.26374%	36.30692%
Dataset 15	36.26374%	32.49623%	36.26374%	36.19763%
Dataset 16	40.65934%	34.99976%	40.65934%	40.56011%
Dataset 17	40.38462%	36.27428%	40.38462%	40.31876%
Dataset 18	39.83516%	36.77725%	39.83516%	39.74044%
Dataset 19	38.46154%	35.827%	38.46154%	38.36521%
Dataset 20	42.58242%	38.644%	42.58242%	42.51821%
Dataset 21	40.10989%	37.90881%	40.10989%	40.02732%
Dataset 22	38.29201%	36.90668%	38.29201%	38.30601%
Dataset 23	39.9449%	35.1532%	39.9449%	39.98179%
Dataset 24	41.32231%	40.29759%	41.32231%	41.35246%
Dataset 25	37.46556%	33.65189%	37.46556%	37.47268%
Dataset 26	34.71074%	29.89534%	34.71074%	34.75865%
Dataset 27	39.9449%	35.78917%	39.9449%	39.96812%
Dataset 28	36.63912%	35.06274%	36.63912%	36.63934%
Dataset 29	38.29201%	34.31324%	38.29201%	38.2878%
Dataset 30	41.87328%	39.03951%	41.87328%	41.87614%
Dataset 31	40.77135%	35.88619%	40.77135%	40.74226%
Dataset 32	43.80165%	40.73727%	43.80165%	43.75228%
Dataset 33	37.19008%	33.98868%	37.19008%	37.18124%

7.3 Random Forest classifier

Tabella 7.3: Tabella media risultati metriche per Random Forest classifier

Dataset	Accuracy	Precision	Recall	Balanced Accuracy
Dataset 1	84.06593%	83.405%	84.06593%	84.13479%
Dataset 2	79.67033%	79.06962%	79.67033%	79.7541%
Dataset 3	85.16484%	85.13501%	85.16484%	85.22313%
Dataset 4	83.51648%	82.49349%	83.51648%	83.58834%
Dataset 5	79.3956%	78.10591%	79.3956%	79.48087%
Dataset 6	79.94505%	79.73896%	79.94505%	80.02732%
Dataset 7	82.96703%	82.83479%	82.96703%	83.02368%
Dataset 8	84.06593%	83.7696%	84.06593%	84.03916%
Dataset 9	84.34066%	84.74718%	84.34066%	84.32149%
Dataset 10	81.31868%	80.65453%	81.31868%	81.28415%
Dataset 11	80.76923%	80.7341%	80.76923%	80.74681%
Dataset 12	83.24176%	82.72689%	83.24176%	83.21494%
Dataset 13	80.21978%	79.80737%	80.21978%	80.16849%
Dataset 14	83.24176%	83.07325%	83.24176%	83.2377%
Dataset 15	81.04396%	81.11757%	81.04396%	80.99271%
Dataset 16	81.59341%	80.64924%	81.59341%	81.54827%
Dataset 17	80.21978%	79.30411%	80.21978%	80.14117%
Dataset 18	84.61538%	84.83731%	84.61538%	84.5765%
Dataset 19	82.96703%	82.73634%	82.96703%	82.92805%
Dataset 20	81.86813%	81.0957%	81.86813%	81.82605%
Dataset 21	82.69231%	82.08141%	82.69231%	82.66393%
Dataset 22	81.5427%	81.48118%	81.5427%	81.55282%
Dataset 23	80.71625%	80.33785%	80.71625%	80.71038%
Dataset 24	84.29752%	84.01015%	84.29752%	84.30328%
Dataset 25	80.16529%	79.68935%	80.16529%	80.17304%
Dataset 26	83.47107%	82.68765%	83.47107%	83.46084%
Dataset 27	81.26722%	81.13433%	81.26722%	81.26138%
Dataset 28	82.92011%	82.60102%	82.92011%	82.90528%
Dataset 29	82.64463%	81.79966%	82.64463%	82.65938%
Dataset 30	82.92011%	82.74064%	82.92011%	82.92805%
Dataset 31	83.47107%	83.78035%	83.47107%	83.48361%
Dataset 32	81.26722%	80.65173%	81.26722%	81.27505%
Dataset 33	82.92011%	82.38394%	82.92011%	82.89162%

7.4 Support Vector Machines classifier

Tabella 7.4: Tabella media risultati metriche per Support Vector Machines classifier

Dataset	Accuracy	Precision	Recall	Balanced Accuracy
Dataset 1	54.67033%	53.76671%	54.67033%	54.72678%
Dataset 2	51.37363%	51.25092%	51.37363%	51.42987%
Dataset 3	52.74725%	51.93782%	52.74725%	52.82332%
Dataset 4	54.12088%	52.96413%	54.12088%	54.19399%
Dataset 5	51.37363%	52.94898%	51.37363%	51.44353%
Dataset 6	53.2967%	51.23573%	53.2967%	53.34244%
Dataset 7	53.84615%	53.29907%	53.84615%	53.87523%
Dataset 8	53.2967%	52.01672%	53.2967%	53.30601%
Dataset 9	54.67033%	54.64264%	54.67033%	54.71767%
Dataset 10	52.1978%	51.30095%	52.1978%	52.24499%
Dataset 11	56.86813%	57.02421%	56.86813%	56.90801%
Dataset 12	57.41758%	57.81782%	57.41758%	57.45902%
Dataset 13	54.67033%	52.73567%	54.67033%	54.68579%
Dataset 14	51.0989%	50.25467%	51.0989%	51.13843%
Dataset 15	51.0989%	50.78627%	51.0989%	50.99271%
Dataset 16	57.69231%	57.06125%	57.69231%	57.60929%
Dataset 17	54.12088%	53.54436%	54.12088%	53.99362%
Dataset 18	57.14286%	55.43765%	57.14286%	57.02641%
Dataset 19	54.12088%	54.46172%	54.12088%	53.99818%
Dataset 20	53.57143%	53.43615%	53.57143%	53.4745%
Dataset 21	55.21978%	55.54726%	55.21978%	55.16393%
Dataset 22	53.16804%	52.19609%	53.16804%	53.1694%
Dataset 23	56.47383%	56.38213%	56.47383%	56.49362%
Dataset 24	58.67769%	58.1095%	58.67769%	58.69308%
Dataset 25	49.03581%	48.74524%	49.03581%	49.04827%
Dataset 26	51.79063%	51.08096%	51.79063%	51.82149%
Dataset 27	53.99449%	53.59481%	53.99449%	54.0255%
Dataset 28	52.61708%	52.70613%	52.61708%	52.62295%
Dataset 29	55.09642%	55.31257%	55.09642%	55.13206%
Dataset 30	57.02479%	56.21099%	57.02479%	57.03097%
Dataset 31	60.88154%	59.53682%	60.88154%	60.93352%
Dataset 32	55.3719%	54.07576%	55.3719%	55.41439%
Dataset 33	55.64738%	54.66697%	55.64738%	55.68306%

7.5 Support Vector Regressor classifier

Tabella 7.5: Tabella media risultati metriche per Support Vector Regressor classifier

Dataset	Accuracy	Precision	Recall	Balanced Accuracy
Dataset 1	100.0%	100.0%	100.0%	100.0%
Dataset 2	100.0%	100.0%	100.0%	100.0%
Dataset 3	100.0%	100.0%	100.0%	100.0%
Dataset 4	99.72527%	100.0%	99.72527%	99.72527%
Dataset 5	99.45055%	100.0%	99.45055%	99.45055%
Dataset 6	49.17582%	24.38471%	49.17582%	49.72222%
Dataset 7	0.0%	0.0%	0.0%	0.0%
Dataset 8	0.0%	0.0%	0.0%	0.0%
Dataset 9	0.0%	0.0%	0.0%	0.0%
Dataset 10	0.0%	0.0%	0.0%	0.0%
Dataset 11	0.0%	0.0%	0.0%	0.0%
Dataset 12	0.0%	0.0%	0.0%	0.0%
Dataset 13	0.0%	0.0%	0.0%	0.0%
Dataset 14	0.0%	0.0%	0.0%	0.0%
Dataset 15	0.27473%	100.0%	0.27473%	0.27473%
Dataset 16	0.0%	0.0%	0.0%	0.0%
Dataset 17	0.0%	0.0%	0.0%	0.0%
Dataset 18	0.0%	0.0%	0.0%	0.0%
Dataset 19	0.0%	0.0%	0.0%	0.0%
Dataset 20	0.0%	0.0%	0.0%	0.0%
Dataset 21	0.0%	0.0%	0.0%	0.0%
Dataset 22	0.0%	0.0%	0.0%	0.0%
Dataset 23	0.0%	0.0%	0.0%	0.0%
Dataset 24	0.0%	0.0%	0.0%	0.0%
Dataset 25	0.0%	0.0%	0.0%	0.0%
Dataset 26	0.0%	0.0%	0.0%	0.0%
Dataset 27	0.0%	0.0%	0.0%	0.0%
Dataset 28	0.0%	0.0%	0.0%	0.0%
Dataset 29	0.0%	0.0%	0.0%	0.0%
Dataset 30	0.0%	0.0%	0.0%	0.0%
Dataset 31	0.0%	0.0%	0.0%	0.0%
Dataset 32	0.0%	0.0%	0.0%	0.0%
Dataset 33	0.0%	0.0%	0.0%	0.0%

Conclusione

Il fine ultimo del lavoro di tesi era quello di riuscire a realizzare dei modelli di workflow per comprenderne la struttura e come possono essere rilevati gli stati d'animo, o mood, attraverso delle immagini del volto.

Ci si era preposti di effettuare questa analisi attraverso il framework WoMan che però non ha raggiunto l'obbiettivo prefissato in quanto, come è risultato evidente dall'analisi effettuata, vi era necessità di diversi altri punti dati per effettuare questo.

Ho proposto diverse soluzioni per risolvere questo problema: la prima è quella di cambiare il dataset utilizzato o di modificare quello già presente in modo da risultare maggiormente congiunto allo scopo che si voleva raggiungere; la seconda soluzione proposta è quella di fornire altri dati al sistema in modo che possa raggiungere i risultati sperati.

Non ho intrapreso nessuna delle due strade in quanto, la prima sarebbe stata troppo dispendiosa in termini di tempo e non sono qualificato alla modifica dei video presenti, mentre, per quanto riguarda la seconda: l'estrapolazione dei dati per i 200 video utilizzati è risultata molto dispendiosa di tempo e di energia elettrica, ho optato quindi per mantenere il numero di sample a 50 per ognuno dei mood utilizzati.

Ho però calcolato, attraverso la regressione lineare, quanti altri sample servirebbero, vagamente, per arrivare al risultato sperato e riporto i risultati ottenuti nelle figure 7.1 7.2 7.3 7.4 7.5

Per il mood bored è stato calcolato che potrebbero essere necessari altri 96 video o più.
Per il mood confused è stato calcolato che potrebbero essere necessari altri 75 video o più.
Per il mood engaged è stato calcolato che potrebbero essere necessari altri 75 video o più.
Per il mood frustrated è stato calcolato che potrebbero essere necessari altri 185 video o più.

Figura 7.1: Numero di video aggiuntivi necessari per ogni mood

Conclusione

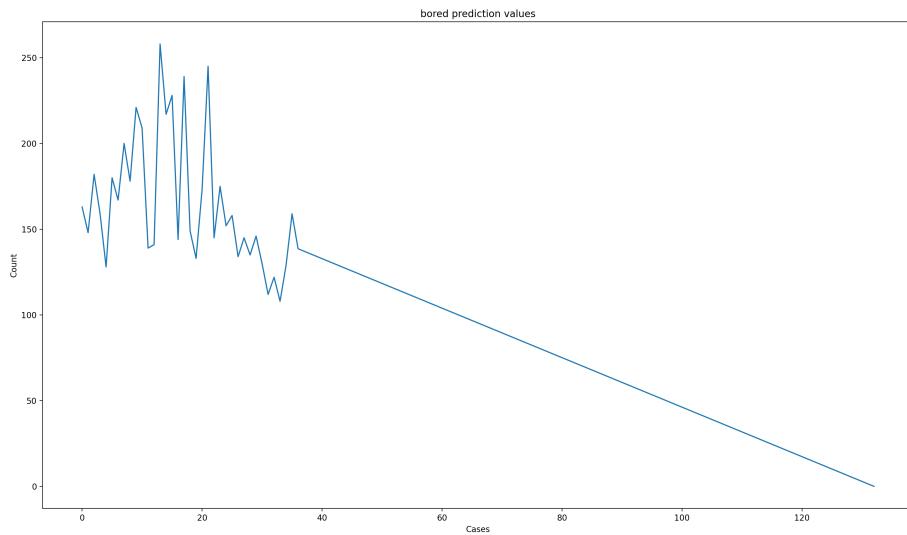


Figura 7.2: Valori predetti per il mood Bored

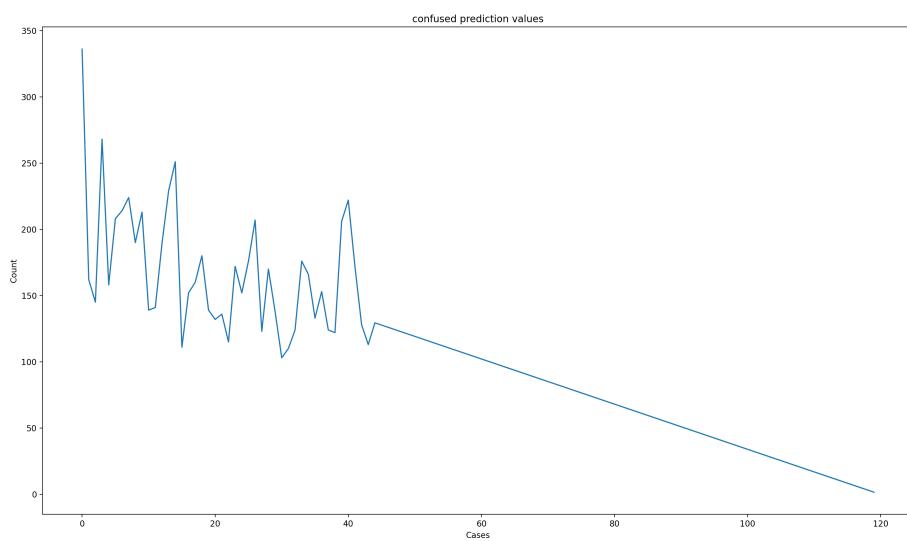


Figura 7.3: Valori predetti per il mood Confused

Conclusione

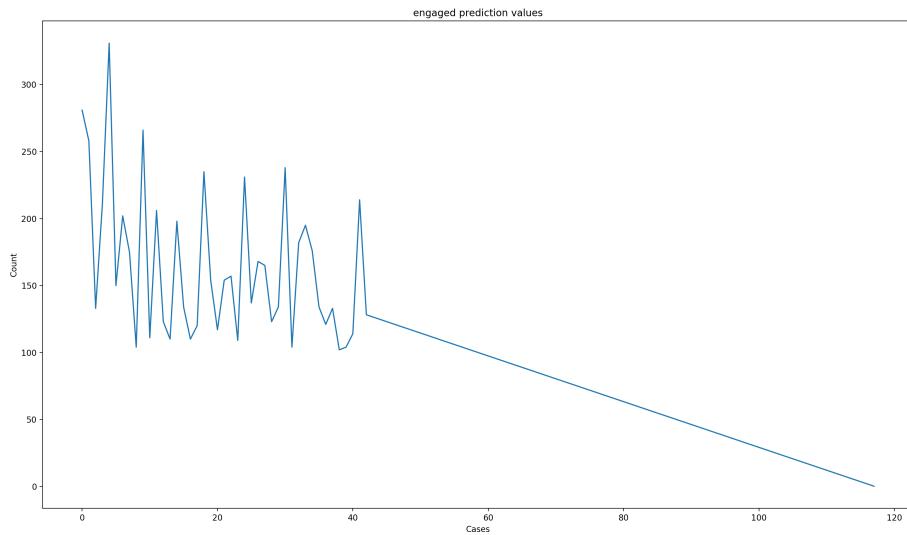


Figura 7.4: Valori predetti per il mood Engaged

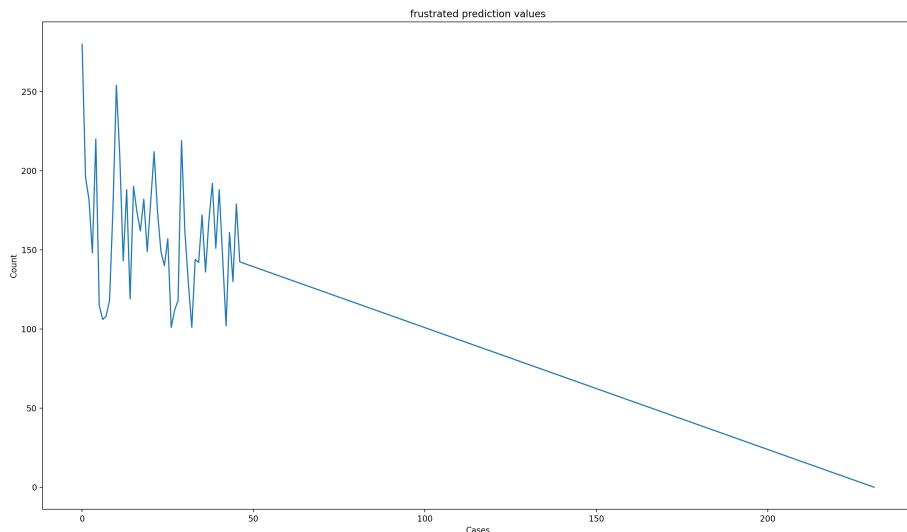


Figura 7.5: Valori predetti per il mood Frustrated

Conclusione

In aggiunta a questo studio, come proposto dalla professoressa De Carolis, si è presa una strada diversa per la rilevazione dei mood attraverso le espressioni facciali.

Ho quindi analizzato i diversi studi presenti per l'analisi delle emozioni FACS per poi procedere all'applicazione di questi metodi allo studio dei mood.

Questa strada ha portato a risultati migliori ed è anche stato possibile realizzare un applicativo per effettuare delle predizioni in tempo reale utilizzando i modelli predittivi creati (vedi 7.6).

Difatti, due dei modelli creati sono capaci di effettuare delle predizioni con una accuracy intorno all'80%, nello specifico il Random Forest classifier e il KNN classifier (vedi 7.6).

Tabella 7.6: Tabella media risultati metriche

	Accuracy	Precision	Recall	Balanced Accuracy
K-Nearest Neighbors	77.87493%	76.43947%	77.87493%	77.87465%
Random forest	82.25838%	81.86107%	82.25838%	82.25838%
Naive bayes	39.16669%	35.96379%	39.16669%	39.16736%
Support vector machine	54.37561%	53.82087%	54.37561%	54.38249%
Support vector regressor	16.62504%	18.92075%	16.62504%	16.6416%

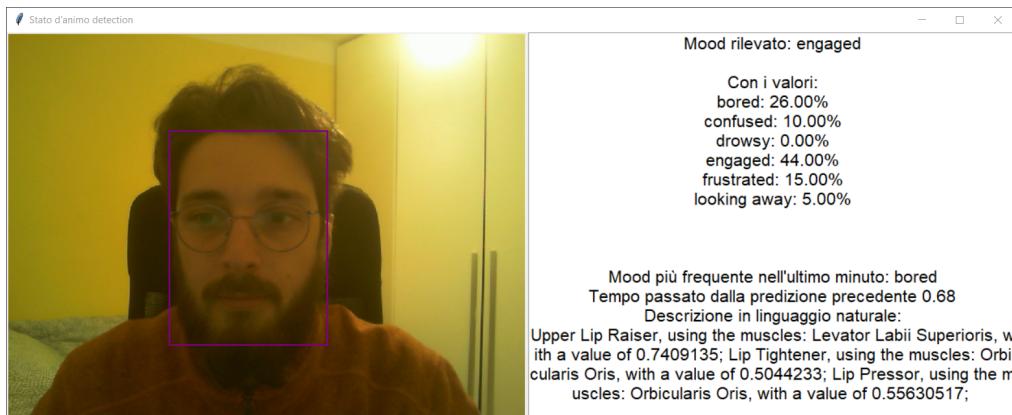


Figura 7.6: Interfaccia per la rilevazione dei mood realizzata

Bibliografia

- [1] Berardina De Carolis, Francesca D'Errico, Nicola Macchiarulo, Marinella Paciello, and Giuseppe Palestra. "Recognizing Cognitive Emotions in E-Learning Environment". In: *International Workshop on Higher Education Learning Methodologies and Technologies Online*. Springer, 2021, pp. 17–27.
- [2] Paolo Buono, Berardina De Carolis, Francesca D'Errico, and Giuseppe Palestra. "Assessing student engagement from facial behavior in on-line learning." *Journal Title* Volume, no. Number (2022): Pages.
- [3] Jean Kossaifi, Georgios Tzimiropoulos, Sinisa Todorovic, and Maja Pantic. AFEW-VA database for valence and arousal estimation in-the-wild. *Image and Vision Computing*, 65:23–36, 2017. Elsevier.
- [4] Jacob Whitehill, Zewelanji N Serpell, Yi-Ching Lin, Amy Foster, and Javier R Movellan. "The Faces of Engagement: Automatic Recognition of Student Engagement from Facial Expressions." *IEEE Transactions on Affective Computing* 5, no. 1 (2014): 86–98.
- [5] Richard Winn Livingstone. *The future in education*. University Press, 1941.
- [6] S. Ceccaccia, A. Generosi, G. Cimini, S. Faggiano, L. Giraldi, and M. Mengoni, "Facial coding as a mean to enable continuous monitoring of student's behavior in e-Learning," in *Proceedings of the 2nd International Workshop on Higher Education Learning Methodologies and Technologies Online*, 2021, pp. 1–10.
- [7] Kardan, S., Kaghazgaran, M., Roshanfekr, P., Sayyad, A., Javidi, M. (2019). Prediction and Localization of student engagement in the wild. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, year=2019.
- [8] Py-feat. (n.d.). Action Unit Reference. Retrieved from https://py-feat.org/pages/au_reference.html
- [9] Taheri, S.M., Daneshvar, S., Chen, Q. (2016). Facial Expression Recognition Based on Local Binary Patterns and Kernel Discriminant Isomap. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.

BIBLIOGRAFIA

- [10] Student Engagement Dataset. Retrieved from <https://www.kaggle.com/datasets/joyee19/studentengagement>
- [11] DAiSEE dataset. Retrieved from <https://people.iith.ac.in/vineethnb/resources/daisee/index.html>
- [12] X. Pu, K. Fan, X. Chen, L. Ji, and Z. Zhou, "Facial expression recognition from image sequences using twofold random forest classifier," *Neurocomputing*, vol. 168, pp. 1173-1180, 2015. <https://doi.org/10.1016/j.neucom.2015.05.005>
- [13] S. Velusamy, H. Kannan, B. Anand, A. Sharma, and B. Navathe, "A method to infer emotions from facial Action Units," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 2028-2031. doi: 10.1109/ICASSP.2011.5946910.
- [14] M. Nadeeshani, A. Jayaweera, and P. Samarasinghe, "Facial Emotion Prediction through Action Units and Deep Learning," in *2020 2nd International Conference on Advancements in Computing (ICAC)*, Malabe, Sri Lanka, 2020, pp. 293-298. doi: 10.1109/ICAC51239.2020.9357138.
- [15] M. Liu, S. Shan, R. Wang, and X. Chen, "Learning expressionlets on spatio-temporal manifold for dynamic facial expression recognition," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1749-1756.
- [16] M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, "Recognizing facial expression: Machine learning and application to spontaneous behavior," in *Proc. of IEEE Conf. on Computer Vision and Pat. Recog. (CVPR)*, 2005, pp. 568-573.
- [17] R. el Kaliouby, "Mind-reading machines: the automated inference of complex mental states from video," Ph.D. Thesis, University of Cambridge, 2005.
- [18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 3rd ed., MIT Press, 2009.
- [19] <https://developer.nvidia.com/cuda-zone>
- [20] <https://tqdm.github.io/>
- [21] Da <https://opencv.org/about/>
- [22] <https://en.wikipedia.org/wiki/Scikit-learn>
- [23] S. Chen, Y. Liu, X. Gao, and Z. Han, "MobileFaceNets: Efficient CNNs for Accurate Real-time Face Verification on Mobile Devices," arXiv preprint arXiv:1804.07573, Apr. 2018.

BIBLIOGRAFIA

- [24] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. *CoRR*, volume abs/1603.02754, 2016. <http://arxiv.org/abs/1603.02754>.
- [25] L. Pham, T. H. Vu and T. A. Tran, "Facial Expression Recognition Using Residual Masking Network," 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 2021, pp. 4513-4519, doi: 10.1109/ICPR48806.2021.9411919.
- [26] V. Albiero, X. Chen, X. Yin, G. Pang and T. Hassner, "img2pose: Face Alignment and Detection via 6DoF, Face Pose Estimation," 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2021, pp. 7613-7623, doi: 10.1109/CVPR46437.2021.00753.
- [27] <https://www.ibm.com/topics/random-forest>
- [28] <https://www.ibm.com/it-it/topics/knn>
- [29] <https://apps.dtic.mil/sti/pdfs/ADA800276.pdf>
- [30] T. Cover and P. Hart, "Nearest neighbor pattern classification," in IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, January 1967, doi: 10.1109/TIT.1967.1053964.
- [31] <https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c>
- [32] Carolis, Berardina De, Stefano Ferilli and Domenico Redavid. "Incremental Learning of Daily Routines as Workflows in a Smart Home Environment." ACM Transactions on Interactive Intelligent Systems (TiiS) 4 (2015): 1 - 23.
- [33] S. Ferilli, "WoMan: Logic-Based Workflow Learning and Management," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 44, no. 6, pp. 744-756, June 2014, doi: 10.1109/TSMC.2013.2273310.
- [34] <https://www.ri.cmu.edu/project/cohn-kanade-au-coded-facial-expression-database/>
- [35] <https://paperswithcode.com/dataset/oulu-casia>
- [36] I. Sneddon, M. McRorie, G. McKeown and J. Hanratty, "The Belfast Induced Natural Emotion Database," in IEEE Transactions on Affective Computing, vol. 3, no. 1, pp. 32-41, Jan.-March 2012, doi: 10.1109/T-AFFC.2011.26.