

Università degli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate (DiSTA)
Corso di Laurea Triennale in Informatica



“Applicazione di Tecniche di Machine Learning per la Determinazione dello Stato di Stress nei Conducenti”

Relatore: Prof. Silvia Corchs

Tesi di Laurea di
Cristian Citterio
Matricola 745297

Anno Accademico 2023-2024

"Se il settore dell'automobile si fosse sviluppato come l'industria informatica, oggi avremmo veicoli che costano 25 dollari e fanno 500 Km con un litro-(Bill Gates)"

Indice

1 Capitolo 1	1
1.1 INTRODUZIONE	1
1.2 Dataset	2
1.2.1 Raccolta dati	2
1.3 Tipi di guida	2
1.3.1 Fasi della guida:	3
1.3.2 Punti fondamentali della ricerca del dataset:	4
1.4 STATO DELL'ARTE	5
2 Capitolo 2	8
2.1 SEGNALI MULTIMODALI	8
2.2 Strumenti e metodi di misura	8
2.3 Segnali fisiologici e di performance	9
2.3.1 Variabili fisiologiche	9
2.3.2 Variabili di performance	11
3 Capitolo 3	13
3.1 FASE DI PREPROCESSING	13
3.1.1 Ambiente di sviluppo	13
3.1.2 Parte 1: Pre-elaborazione dei dati	13
3.1.3 Pulizia e sistemazione dei dati:	14
3.1.4 Esplorazione dei dati prima parte	15
3.1.5 Esplorazione dei dati seconda parte	17
4 Capitolo 4	24
4.1 ESTRAZIONE DELLE FEATURES	24
4.2 Creazione colonna stress, per classificazione binaria	24
4.3 Features segnali fisiologici	25
4.3.1 Features aggiuntive Heart Rate	26
4.3.2 Features aggiuntive Breathing Rate	27
4.3.3 Features aggiuntive EDA palmare	28
4.3.4 Features EDA perinasale	28
4.4 Features segnali telemetrici	29
4.5 Applicazione delle features ai sottogruppi di interesse	30
4.6 Post estrazione e filtraggio dati	32

5 Capitolo 5	34
5.1 CLASSIFICATORI UTILIZZATI	34
5.1.1 Classificatori base	34
5.1.2 Classificatori ottimizzati	36
5.1.3 Gridsearch	39
6 Capitolo 6	41
6.1 TECNICHE DI APPRENDIMENTO E METODI DI VALIDAZIONE	41
6.1.1 Definizione delle caratteristiche ed etichette	41
6.1.2 Divisione dei dati in set di addestramento e di test	42
6.1.3 Metodi di validazione incrociata	43
6.1.4 Standardizzazione delle features	45
6.1.5 Altri metodi utilizzati	45
6.1.6 Matrice di confusione:	47
6.1.7 Synthetic Minority Over-sampling Technique (SMOTE)	48
7 Capitolo 7	49
7.1 RISULTATI E DISCUSSIONI	49
7.2 Risultati classificazione binaria:	50
7.2.1 StratifiedShuffleSplit con matrice di confusione a due etichette, classificatori ottimizzati:	50
7.2.2 StratifiedShuffleSplit (SMOTE) con matrice di confusione a due etichette, classificatori ottimizzati:	53
7.2.3 Principal Component Analysis (PCA)	59
7.2.4 Receiver Operating Characteristic (ROC)	61
7.3 Risultati classificazione multiclasse:	63
7.3.1 StratifiedShuffleSplit con matrice di confusione a quattro etichette, classificatori ottimizzati:	63
7.3.2 StratifiedShuffleSplit (SMOTE) con matrice di confusione a quattro etichette, classificatori ottimizzati:	66
7.3.3 StratifiedShuffleSplit con matrice di confusione a quattro etichette, classificatori base:	68
7.3.4 StratifiedShuffleSplit (SMOTE) con matrice di confusione a quattro etichette, classificatori base:	70
7.4 Metodi di validazione incrociata più rigorosi	71
7.4.1 'Five Fold Cross Validation' a 2 etichette	72
7.4.2 'Five Fold Cross Validation' a 4 etichette	74
7.4.3 'Leave One Subject Out' a 2 etichette	76
7.4.4 'Leave One Subject Out' a 4 etichette	77
7.5 Considerazioni finali	77
8 Capitolo 8	79
8.1 CONCLUSIONI	79

Elenco delle figure

2.1	Un esempio di sensori utilizzati durante la simulazione	9
2.2	Heart Rate variability	10
2.3	Respiratory Rate chart	10
2.4	EDA Funzionamento	11
3.1	Creazione colonna Fase nel DataFrame	14
3.2	Segnale filtrato Breathing Rate	16
3.3	Segnale filtrato Heart Rate	16
3.4	Segnale filtrato Palm EDA	16
3.5	Segnale filtrato Perinasal EDA	16
3.6	Segnale filtrato Speed	16
3.7	Segnale filtrato Acceleration	16
3.8	Segnale filtrato Braking	17
3.9	Risultati 1	19
3.10	Risultati 2	19
3.11	Risultati 3	20
3.12	Codice creazione boxplots	21
3.13	Codice creazione boxplots	21
3.14	Boxplots Drive 5 'PP'	22
3.15	Boxplots Drive 6 'PP'	22
3.16	Boxplots Drive 7 'PP'	22
3.17	Boxplots Drive 5 'HR'	22
3.19	Boxplots Drive 7 'HR'	22
3.18	Boxplots Drive 6 'HR'	23
3.20	Boxplots Drive 5 'BR'	23
3.21	Boxplots Drive 6 'BR'	23
3.22	Boxplots Drive 7 'BR'	23
4.1	Features Heart Rate	25
4.2	Features Accelerazione	30
4.3	Applicazione delle features ai sottogruppi di interesse	31
4.4	Filtraggio dati post estrazione features	33
5.1	K-Nearest Neighbors (K-NN)	35
5.2	Support Vector Machine (SVM)	35
5.3	Artificial Neural Network	36
5.4	Classificatori base	36

5.5	Naive Bayes (NB)	37
5.6	Logistic Regression (LR)	37
5.7	Decision Tree	38
5.8	Random Forest	39
5.9	1. Classificatori ottimizzati	40
5.10	2. Classificatori ottimizzati	40
6.1	Five Fold Cross Validation	44
6.2	Leave One Subject Out	45
6.3	Create Signal Boxplot	46
6.4	Confusion Matrix	48
7.1	1. Confusion Matrix	50
7.2	2. Confusion Matrix	53
7.3	3. Confusion Matrix	56
7.4	4. Confusion Matrix	58
7.5	PCA 95% di varianza	60
7.6	PCA 99% di varianza	61
7.7	ROC classificatori ottimizzati	62
7.8	ROC classificatori base	62
7.9	5. Confusion Matrix	63
7.10	6. Confusion Matrix	66
7.11	7. Confusion Matrix	68
7.12	8. Confusion Matrix	70
7.13	Boxplot dei punteggi di validazione incrociata ‘Five Fold Cross Validation’ a 2 etichette per diversi segnali utilizzando i classificatori KNN, SVM e ANN. L’asse y rappresenta il punteggio ‘ f1_micro ’	72
7.14	Boxplot dei punteggi di validazione incrociata ‘Five Fold Cross Validation’ a 2 etichette per i tre classificatori utilizzati: KNN, SVM, e ANN. L’asse y rappresenta il punteggio ‘ f1_micro ’	73
7.15	Boxplot dei punteggi di validazione incrociata ‘Five Fold Cross Validation’ a 4 etichette per diversi segnali utilizzando i classificatori KNN, SVM e ANN. L’asse y rappresenta il punteggio ‘ f1_micro ’	74
7.16	Boxplot dei punteggi di validazione incrociata ‘Five Fold Cross Validation’ a 4 etichette per i tre classificatori utilizzati: KNN, SVM, e ANN. L’asse y rappresenta il punteggio ‘ f1_micro ’	75
7.17	Boxplot dei punteggi di validazione incrociata ‘Leave One Subject Out’ a 2 etichette per diversi segnali utilizzando i classificatori KNN, SVM e ANN. L’asse y rappresenta il punteggio ‘ f1_micro ’	76
7.18	Boxplot dei punteggi di validazione incrociata ‘Leave One Subject Out’ a 2 etichette per i tre classificatori utilizzati: KNN, SVM, e ANN. L’asse y rappresenta il punteggio ‘ f1_micro ’	76
7.19	Boxplot dei punteggi di validazione incrociata ‘Leave One Subject Out’ a 4 etichette per diversi segnali utilizzando i classificatori KNN, SVM e ANN. L’asse y rappresenta il punteggio ‘ f1_micro ’	77

- 7.20 Boxplot dei punteggi di validazione incrociata ‘Leave One Subject Out’ a 4 etichette per i tre classificatori utilizzati: KNN, SVM, e ANN. L’asse y rappresenta il punteggio ‘**f1_micro**’ 77

Elenco delle tabelle

7.1 Rapporto del classificatore Naive Bayes	50
7.2 Rapporto del classificatore Logistic Regression	50
7.3 Rapporto del classificatore K-Nearest Neighbors (KN)	51
7.4 Rapporto del classificatore Support Vector Machine (SVM)	51
7.5 Rapporto del classificatore Decision Tree	51
7.6 Rapporto del classificatore Random Forest	51
7.7 Rapporto del classificatore Naive Bayes	53
7.8 Rapporto del classificatore Logistic Regression	53
7.9 Rapporto del classificatore K-Nearest Neighbors (KN)	54
7.10 Rapporto del classificatore Support Vector Machine (SVM)	54
7.11 Rapporto del classificatore Decision Tree	54
7.12 Rapporto del classificatore Random Forest	54
7.13 Rapporto del classificatore K-Nearest Neighbors (KN) semplice	56
7.14 Rapporto del classificatore Support Vector Machine (SVM) semplice	56
7.15 Rapporto del classificatore Artificial Neural Network (AN) semplice	57
7.16 Rapporto del classificatore K-Nearest Neighbors (KN) semplice	58
7.17 Rapporto del classificatore Support Vector Machine (SVM) semplice	58
7.18 Rapporto del classificatore Artificial Neural Network (AN) semplice	59
7.19 Rapporto del classificatore Naive Bayes	63
7.20 Rapporto del classificatore Logistic Regression	63
7.21 Rapporto del classificatore K-Nearest Neighbors (KN)	64
7.22 Rapporto del classificatore Support Vector Machine (SVM)	64
7.23 Rapporto del classificatore Decision Tree	64
7.24 Rapporto del classificatore Random Forest	64
7.25 Rapporto del classificatore Naive Bayes	66
7.26 Rapporto del classificatore Logistic Regression	66
7.27 Rapporto del classificatore K-Nearest Neighbors (KN)	67
7.28 Rapporto del classificatore Support Vector Machine (SVM)	67
7.29 Rapporto del classificatore Decision Tree	67
7.30 Rapporto del classificatore Random Forest	67
7.31 Rapporto del classificatore K-Nearest Neighbors (KN) semplice	69
7.32 Rapporto del classificatore Support Vector Machine (SVM) semplice	69
7.33 Rapporto del classificatore Artificial Neural Network (AN) semplice	69
7.34 Rapporto del classificatore K-Nearest Neighbors (KN) semplice	70
7.35 Rapporto del classificatore Support Vector Machine (SVM) semplice	71
7.36 Rapporto del classificatore Artificial Neural Network (AN) semplice	71

1

Capitolo 1

1.1 INTRODUZIONE

La guida distratta è un problema di sicurezza stradale che causa molti incidenti e vittime ogni anno. Questo problema può essere causato da diversi tipi di distrazioni, come l'uso del cellulare, la conversazione con i passeggeri, l'ascolto della musica, o la gestione di eventi stressanti. Per studiare gli effetti delle distrazioni sulla guida, è fondamentale avere a disposizione un dataset che contenga dati oggettivi e soggettivi sui conducenti, sulle loro reazioni fisiologiche e comportamentali, e sulle loro prestazioni di guida.

Nel contesto di questa tesi, il dataset “A multimodal dataset for various forms of distracted driving” (pubblicato su Scientific Data nel 2017 dagli autori Salah Taamneh, Panagiotis Tsiamyrtzis, Malcolm Dcosta, Pradeep Buddharaju, Ashik Khatri, Michael Manser, Thomas Ferris, Robert Wunderlich e Ioannis Pavlidis, disponibile online sulla piattaforma OSF [ref1]) è stato utilizzato come riferimento principale. Questo dataset offre una panoramica completa delle varie forme di distrazione alla guida, fornendo sia dati oggettivi che soggettivi sui conducenti, le loro reazioni fisiologiche e comportamentali, e le loro prestazioni di guida.

Dopo una fase iniziale di pre-elaborazione e filtraggio dei dati, è stato valutato lo stato di stress dei soggetti che hanno partecipato al simulatore di guida. Questa valutazione è stata effettuata utilizzando vari classificatori e tecniche di machine learning, che hanno permesso di analizzare e comprendere meglio l'impatto delle distrazioni sulla guida.

Gli obiettivi principali di questo lavoro sono multipli. Prima di tutto, si vuole comprendere in modo più approfondito come le distrazioni influenzano la guida e contribuiscono agli incidenti stradali. In secondo luogo, si mira a sviluppare modelli predittivi efficaci che possano identificare i conducenti distratti e valutare il loro livello di stress. Si spera

che i risultati di questo studio possano contribuire a sviluppare strategie e tecnologie per mitigare gli effetti delle distrazioni alla guida, migliorando così la sicurezza stradale.

In conclusione, la guida distratta è un problema significativo che richiede ulteriori ricerche e interventi mirati. Attraverso l'analisi dei dati e l'applicazione del machine learning, questa tesi si propone di contribuire a questa importante area di studio.

1.2 Dataset

Il set di dati utilizzato per gli esperimenti di questo progetto rappresenta uno dei primi a fornire informazioni multimodali su vari tipi di guida distratta. Questo è stato possibile grazie all'uso di un simulatore di guida realistico e una serie di sensori non invasivi.

Il dataset include dati termici, elettro-dermici, cardiaci, respiratori, oculari, e di performance di guida di 68 soggetti che hanno guidato in sei scenari diversi, con o senza distrazioni cognitive, emotive o sensorimotorie.

La validità di queste informazioni è stata confermata attraverso vari metodi statistici e analitici. Questi metodi hanno dimostrato che alcune variabili fisiologiche, come l'EDA perinasale (in riferimento alla zona intorno al naso) e la frequenza cardiaca, sono indicatori affidabili di stress durante la guida distratta. Queste informazioni sono ora disponibili per la comunità scientifica e possono essere utilizzate per ulteriori ricerche sulla sicurezza stradale e sull'interazione uomo-macchina. Questo rappresenta un passo importante verso la comprensione e la prevenzione degli incidenti stradali causati dalla guida distratta.

1.2.1 Raccolta dati

Il dataset è organizzato per soggetto in tre cartelle principali:

- **(1) Raw Thermal Data:** che contiene le sequenze termiche del viso per tutte le sessioni sperimentali.
- **(2) Structured Study Data:** che contiene i dati quantitativi estratti dalle diverse modalità di misurazione, in un formato gerarchico che riflette il disegno sperimentale.
- **(3) R-Friendly Study Data,:** che contiene una copia dei dati quantitativi (fisiologici e prestazionali) in un formato piatto, adatto per l'analisi statistica.

1.3 Tipi di guida

Lo studio ha esaminato l'effetto di stressori cognitivi, emotivi e sensorimotori sul comportamento di guida in condizioni tipiche. Per fare questo, ha sottoposto i soggetti a sette sessioni di guida:

- **Baseline (B):** La prima sessione dell'Esperimento I era una sessione di base in cui i soggetti sedevano tranquillamente in una stanza con luce soffusa, ascoltando musica rilassante per 5 minuti. Dopo questa sessione di base, i soggetti hanno effettuato sei sessioni di guida al simulatore.
- **Practice Drive (PD):** In cui soggetti hanno familiarizzato con il simulatore di guida guidando su un tratto rettilineo di 8 km di una strada a quattro corsie a velocità stabile.
- **Relaxing Drive (RD):** I soggetti dovevano guidare su un tratto rettilineo di 10,9 km di una strada a quattro corsie con limite di velocità di 70 km/h; due corsie erano dedicate al traffico in ciascuna direzione, con l'auto del soggetto che viaggiava nella corsia di destra (R); c'era un leggero traffico sulle corsie opposte (3 veicoli per km). I soggetti erano costretti a cambiare corsia (R a L) dopo 5,2 km di guida. Dovevano rimanere nella corsia di sinistra (L) per 1,2 km, prima di avere l'opportunità di tornare alla corsia di destra (R), se lo desideravano. La motivazione per questo cambio di corsia era ridurre la monotonia della guida.
- **Unità caricate (Loaded-Drive):**
 - **Normal Drive-Loaded Drive (LDØ = ND):** In questa condizione, il conducente non doveva fare nessuna attività secondaria, ma solo guidare normalmente. Questa condizione serviva come termine di confronto per le altre condizioni, per valutare l'effetto delle distrazioni sul comportamento di guida.
 - **Cognitive Drive(LDC = CD):** In questa condizione, il conducente doveva rispondere a domande matematiche o analitiche, come calcolare il 15% di 120 o dire il nome di un paese che inizia con la lettera A. Queste domande venivano poste tramite un altoparlante e il conducente doveva rispondere a voce alta. Lo scopo era di creare una situazione di carico cognitivo, che richiede attenzione, memoria e ragionamento.
 - **Emotional Drive((LDE = ED):** In questa condizione, il conducente doveva rispondere a domande emotivamente stimolanti, come raccontare un episodio in cui si è sentito felice, triste o arrabbiato. Queste domande venivano poste tramite un altoparlante e il conducente doveva rispondere a voce alta. Lo scopo era di creare una situazione di carico emotivo, che richiede empatia ed espressione.
 - **Sensorimotor Drive(LDM = MD):** In questa condizione, il conducente doveva inviare messaggi di testo con lo smartphone, seguendo le istruzioni che apparivano sullo schermo. Queste istruzioni potevano essere di scrivere una parola, una frase o un numero. Lo scopo era di creare una situazione di carico sensorimotorio, che richiede coordinazione, destrezza e percezione.

1.3.1 Fasi della guida:

La disposizione delle fasi all'interno di ogni guida stressante LD_j (_j che appartiene[C,E,M]) era la seguente:

- **Fase P1LDj:** Guida senza distrazioni per 80 s.
- **Fase P2LDj:** Guida mentre si svolge un'attività secondaria j per 160 s.
- **Fase P3LDj:** Guida senza distrazioni per 240 s.
- **Fase P4LDj:** Guida mentre si svolge un'attività secondaria j per 160 s.
- **Fase P5LDj:** Guida senza distrazioni per 120 s.

Nel DataFrame scaricato, le diverse sessioni di guida vengono etichettate nell'apposita colonna 'Drive' per ogni soggetto che ha partecipato all'esperimento in questo modo:

- **Baseline:** etichettata con 1
- **Practice Drive:** etichettata con 2
- **Relaxing Drive:** etichettata con 3
- **Normal Drive:** etichettata con 4
- **Cognitive Drive:** etichettata con 5
- **Emotional Drive:** etichettata con 6
- **Sensorimotor Drive:** etichettata con 7

1.3.2 Punti fondamentali della ricerca del dataset:

- **Obiettivo dello studio:** Questa ricerca si propone di esaminare l'impatto di diversi tipi di distrazioni (cognitive, emotive e sensomotorie) sullo stato interno dei conducenti, sul loro comportamento osservabile e sulle loro prestazioni di guida. L'analisi si estende a situazioni di guida tipiche e alla reazione dei conducenti a eventi inaspettati.
- **Metodologia di raccolta dei dati:** Le informazioni sono state acquisite in modo non invasivo, utilizzando telecamere termiche e visive o sensori indossabili. Questo ha garantito che i conducenti mostrassero comportamenti naturali, non disturbati dalla meccanica degli esperimenti.
- **Analisi dei dati:** Le informazioni raccolte sono state analizzate utilizzando metodi statistici e di apprendimento automatico per estrarre segnali fisiologici, emozionali e di prestazione. I dati sono stati anche valutati soggettivamente dai conducenti tramite il NASA Task Load Index (TLX), uno strumento psicométrico che misura il carico di lavoro percepito.
- **Valore del dataset:** Queste informazioni offrono una ricca risorsa per la comunità scientifica per svolgere indagini esplorative e ipotetiche riguardanti l'effetto di vari tipi di stress sulla guida. Le informazioni possono anche essere utilizzate per studi comparativi realistici di metodi di misurazione affettivi, come EDA perinasale versus EDA palmare, o per il riconoscimento facciale multi-spettrale, grazie all'uso di modalità di imaging termiche e visive per i volti dei soggetti.

1.4 STATO DELL'ARTE

Gli articoli citati esplorano vari aspetti delle distrazioni alla guida e utilizzano tecniche avanzate per rilevare e analizzare lo stress del conducente. Questi studi utilizzano segnali fisiologici, come la frequenza cardiaca e la frequenza respiratoria che possono fornire informazioni preziose sul livello di stress del conducente, permettendo di prevedere e, in alcuni casi, di mitigare i comportamenti di guida pericolosi.

Vengono utilizzate tecniche di machine learning per analizzare i dati raccolti e fare previsioni sullo stato del conducente. Questi algoritmi possono identificare modelli nei dati che possono non essere immediatamente evidenti, fornendo una comprensione più profonda del comportamento dei soggetti.

La ricerca sullo stress alla guida è un campo importante che può avere un impatto significativo sulla sicurezza stradale. Gli studi citati rappresentano un contributo importante a questo campo, fornendo nuove intuizioni e metodi per la rilevazione e l'analisi dello stress del conducente.

- Taamneh et al. 2017 ([1]): Questo è l'articolo originale che descrive il dataset multimodale acquisito in un esperimento controllato su un simulatore di guida. Il set include dati per n=68 volontari che hanno guidato in quattro diverse condizioni: nessuna distrazione, distrazione cognitiva, distrazione emotiva e distrazione sensorimotoria.
- Bianco et al. 2019([2]): Questo articolo presenta uno studio sulla rilevazione dello stress del conducente di un'auto multimodale. Gli autori hanno utilizzato quattro diversi segnali: frequenza cardiaca, frequenza respiratoria, EDA del palmo e sudorazione perinasale. Il problema di riconoscimento è stato formulato come una classificazione binaria stress vs non-stress.
- Perry e Baldwin 2000 ([3]): Questo studio fornisce ulteriori prove delle associazioni tra i punteggi della personalità di tipo A e gli atteggiamenti e comportamenti legati alla guida. Gli autori hanno scoperto che la personalità di tipo A era significativamente correlata a più incidenti stradali, maggiore frequenza di violazione delle norme stradali, maggiore impazienza durante la guida, maggiori manifestazioni di aggressività sulla strada e comportamenti di guida più rischiosi. La personalità di tipo A è un termine usato per descrivere un insieme di caratteristiche psicologiche che includono competitività, impazienza, ostilità, senso di urgenza e forte motivazione al successo. Le persone con personalità di tipo A tendono ad essere più stressate, ansiose e suscettibili a malattie cardiache rispetto alle persone con personalità di tipo B, che sono più rilassate, calme e flessibili.
- Napoletano e Rossi 2018 ([4]): Questo articolo presenta un metodo per riconoscere lo stress del conducente di un'auto combinando la frequenza cardiaca e la frequenza respiratoria. Il problema è definito mediante una classificazione binaria, cioè stress vs non-stress, ed è affrontato utilizzando la strategia di classificazione delle macchine a vettori di supporto (SVM). Gli autori propongono l'uso di una combinazione di caratteristiche tradizionali dello stato dell'arte e valori grezzi dei segnali acquisiti come rappresentazione dei dati.

- Zangróniz et al. 2017 ([5]): Questo articolo introduce un sensore di attività elettrodermica per la classificazione della condizione di calma/stress. Gli autori propongono un dispositivo indossabile in grado di acquisire l'EDA di un soggetto al fine di rilevare la sua condizione di calma/stress dai segnali fisiologici acquisiti.
- Healey e Picard 2005 ([6]): Questo articolo presenta metodi per raccogliere e analizzare dati fisiologici durante compiti di guida nel mondo reale per determinare il livello di stress di un conducente. Elettrocardiogramma, elettromiogramma, conductanza cutanea e respirazione sono stati registrati continuamente mentre i conducenti seguivano un percorso stabilito su strade aperte nella zona di Boston. L'analisi ha utilizzato caratteristiche da intervalli di dati di 5 minuti durante le condizioni di riposo, guida in autostrada e guida in città per distinguere tre livelli di stress.
- Alibeigi et al. 2023 ([7]): Questo articolo introduce il Zenseact Open Dataset (ZOD), un dataset multimodale su larga scala raccolto in vari paesi europei nel corso di due anni, coprendo un'area 9 volte superiore rispetto ai dataset esistenti. ZOD vanta i sensori con la più alta portata e risoluzione tra i dataset comparabili, abbinati a dettagliate annotazioni di keyframe per oggetti 2D e 3D (fino a 245m), segmentazione semantica/istanza della strada, riconoscimento dei segnali stradali e classificazione della strada.
- Choi et al. 2016 ([8]): Questo articolo discute i sistemi di monitoraggio dello stato del conducente per veicoli intelligenti utilizzando sensori fisiologici. Gli autori evidenziano il rapido aumento del numero di automobili e i seri problemi che hanno creato, come incidenti, congestione del traffico e inquinamento atmosferico. Propongono diversi modelli di apprendimento basati su CNN e CNN-LSTM utilizzando segnali fisiologici per i livelli di stress del conducente.
- Cooper et al. 2009 ([9]): Questa ricerca esplora l'interrelazione tra la distrazione del conducente e le caratteristiche del comportamento del conducente associate a una ridotta efficienza del traffico stradale. Gli autori hanno scoperto che sia la distrazione del conducente che la congestione del traffico influenzavano significativamente la frequenza dei cambi di corsia, la velocità media e la probabilità di rimanere dietro a un veicolo più lento.
- Pavlidis et al. 2016 ([10]): In questo articolo, gli autori sezionano i comportamenti del conducente sotto stress cognitivo, emotivo, sensorimotorio e misto. Hanno scoperto che tutti gli stress applicati comportano aumenti significativi nell'arousal simpatico medio accompagnati da aumenti significativi nella sterzata assoluta media. L'arousal simpatico è una condizione che indica l'attivazione del sistema nervoso simpatico, una parte del sistema nervoso autonomo che regola le funzioni involontarie del corpo. L'arousal simpatico si manifesta con una serie di cambiamenti fisiologici, come l'aumento della frequenza cardiaca, della pressione arteriosa, della sudorazione e della glicemia.
- Tsiamyrtzis et al. 2016 ([11]): Questo articolo studia come misurare l'attività elettrodermica (EDA) su diverse parti del corpo. Di solito, l'EDA si misura sulle dita o sul palmo della mano, ma in questo articolo vengono utilizzati dei sensori portatili

che si possono mettere anche su altre parti del corpo. Gli autori hanno spaventato diversi soggetti con dei rumori improvvisi misurando la loro EDA. Hanno scoperto che i sensori portatili funzionano bene sulle dita e sul palmo, ma non sul collo o sul piede. Questo significa che bisogna stare attenti a dove posizionare correttamente i sensori per calcolare in maniera ottimale l'attività elettrodermica cutanea.

- Shastri et al. 2012 ([12]): Questo articolo parla di un metodo per rilevare lo stress fisiologico utilizzando una telecamera termica in grado vedere il calore che emana il nostro corpo. Gli autori hanno usato questa telecamera per analizzare il naso delle persone durante i periodi di stress, osservando che in queste situazioni diventava più caldo. Hanno anche scoperto che i soggetti che usavano il cellulare mentre guidavano erano più stressati e registravano le prestazioni di guida peggiori.

2

Capitolo 2

2.1 SEGNALI MULTIMODALI

Secondo quanto descritto nel dataset utilizzato in questo progetto, nel corso degli esperimenti sono stati presi in considerazione diversi segnali per determinare lo stato di 'stress' dei conducenti durante la simulazione di guida. In tale sezione vengono presentati i segnali utilizzati ed i diversi metodi di misura impiegati per calcolarli.

2.2 Strumenti e metodi di misura

- **Imaging termico e visivo del viso:** Questi dati sono stati usati per estrarre i segnali di sudorazione perinasale e di espressione facciale, che sono indicatori di stress emotivo e cognitivo.
- **Sensori indossabili:** Questi dispositivi hanno misurato il battito cardiaco, il ritmo respiratorio e l'attività elettrodermica (EDA) della mano, che sono indicatori di attivazione simpatica.
- **Eye tracking:** Questo sistema ha registrato la direzione dello sguardo del conducente, che è un indicatore di attenzione e distrazione.
- **Simulatore di guida:** Questo software ha salvato un record dei parametri di guida, come la velocità, l'accelerazione, la forza del freno, l'angolo dello sterzo e la posizione

della corsia, che sono indicatori di prestazione e reattività.

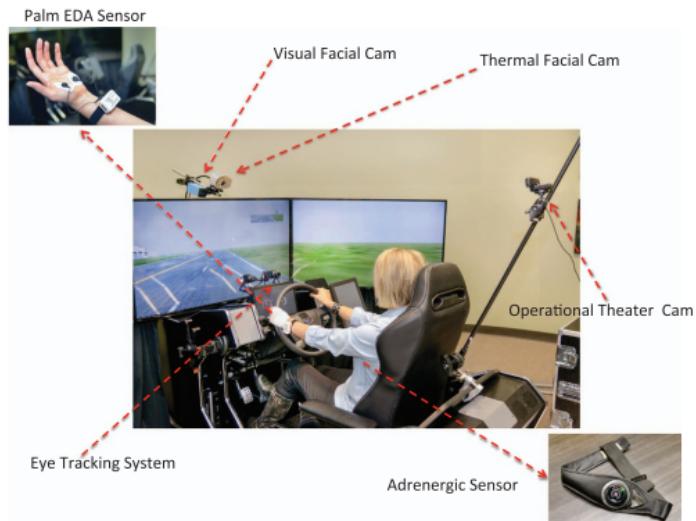


Figura 2.1: Un esempio di sensori utilizzati durante la simulazione

2.3 Segnali fisiologici e di performance

Le variabili fisiologiche e di performance sono due gruppi di variabili che sono state utilizzate nello studio “Data Descriptor: A multimodal dataset for various forms of distracted driving” per analizzare l’effetto delle distrazioni sulla guida. Di seguito vi è una descrizione completa per ciascuna variabile appartenente a questi gruppi.

2.3.1 Variabili fisiologiche

Le variabili fisiologiche sono variabili che registrano i segnali del sistema nervoso simpatico dei soggetti, come il battito cardiaco, il ritmo respiratorio, l’attività elettrodermica (EDA) del palmo e della regione perinasale, usando sensori indossabili o telecamere termiche e visive. Queste variabili sono state raccolte continuamente durante le sessioni di guida, per monitorare la risposta fisiologica del conducente alle diverse condizioni di distrazione. Le variabili fisiologiche sono le seguenti:

- **Battito cardiaco:** è il numero di battiti del cuore al minuto (bpm). Indica il livello di attivazione cardiaca del conducente e può essere influenzato da fattori emotivi, cognitivi e ambientali. Un battito cardiaco elevato può indicare uno stato di stress, ansia o paura. Il battito cardiaco è stato misurato usando un sensore ottico indossabile al polso, che rileva le variazioni del flusso sanguigno attraverso la pelle.

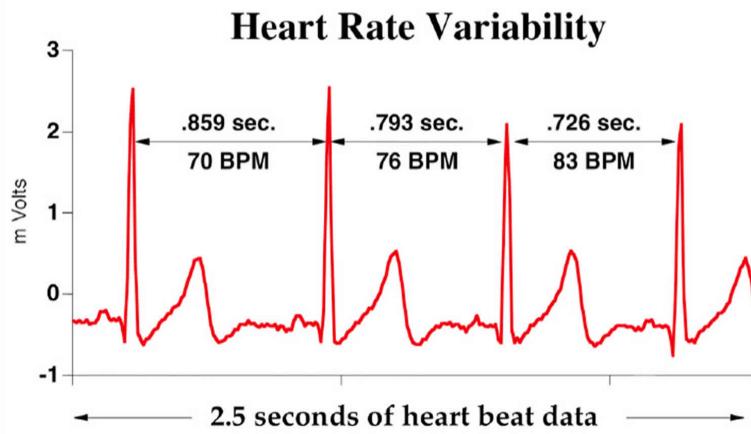


Figura 2.2: Heart Rate variability

- **Ritmo respiratorio:** è il numero di atti respiratori al minuto. Indica il livello di attivazione respiratoria del conducente e può essere influenzato da fattori emotivi, cognitivi e ambientali. Un ritmo respiratorio elevato può indicare uno stato di stress, ansia o paura. Il ritmo respiratorio è stato misurato usando un sensore inerziale indossabile al torace, che rileva i movimenti del petto durante la respirazione.

Acceptable Range of Respiratory Rates for Age	
Age	Rate (Breaths per Minute)
Newborn	30-40
Infant (6 months)	20-40
Toddler (2 years)	25-32
Child	20-30
Adolescent	16-19
Adult	12-20

Figura 2.3: Respiratory Rate chart

- **Attività elettrodermica (EDA) del palmo:** è la variazione della conduttanza elettrica della pelle della mano, causata dalla secrezione di sudore da parte delle ghiandole sudoripare. Indica il livello di attivazione elettrodermica del conducente e può essere influenzato da fattori emotivi, cognitivi e ambientali. Un'attività elettrodermica elevata può indicare uno stato di stress, ansia o paura. L'attività elettrodermica del palmo è stata misurata usando un sensore elettrodermico indossabile al dito, che applica una piccola corrente elettrica alla pelle e ne misura la resistenza.

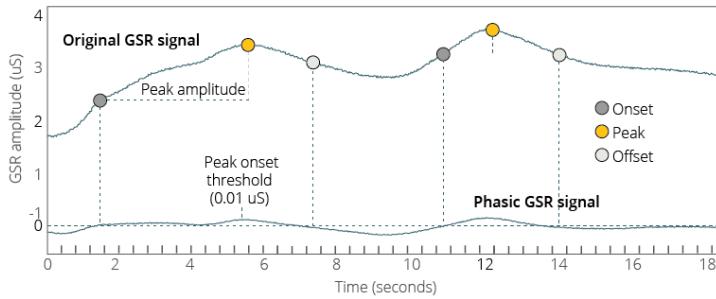


Figura 2.4: EDA Funzionamento

Questa immagine illustra il funzionamento dell'Attività Elettrodermica (EDA). La soglia del segnale agisce come un confine per l'incremento massimo del picco. Qualsiasi valore che supera questa soglia (ad esempio, 0,1 μ S) da un campione all'altro è considerato come un aumento troppo rapido per riflettere un vero processo fisiologico e quindi viene scartato. Con questi limiti impostati, un conteggio dei punti dati dovrebbe rispecchiare il numero di picchi GSR presenti nei dati.

- **Attività elettrodermica (EDA) perinasale:** Analogamente all'EDA del palmo, questa variabile indica il livello di attivazione elettrodermica del conducente nella regione perinasale. L'attività elettrodermica perinasale è stata misurata usando una telecamera termica, che rileva le variazioni di temperatura della pelle dovute alla sudorazione.

2.3.2 Variabili di performance

Sono variabili che, usando il simulatore di guida, rilevano i parametri di guida dei soggetti, come la velocità, l'accelerazione, la forza del freno, l'angolo dello sterzo e la posizione nella corsia. Queste variabili sono state registrate continuamente durante le diverse sessioni di guida, per valutare il comportamento e la reattività del conducente alle diverse condizioni di distrazione. Le variabili di performance sono le seguenti:

- **Velocità:** è la rapidità con cui il veicolo si muove. Si misura in metri al secondo (m/s) o in chilometri all'ora (km/h). Indica il livello di controllo e di adattamento del conducente alle condizioni della strada e del traffico. La velocità è stata registrata usando il simulatore di guida, che calcola la distanza percorsa dal veicolo nel tempo.
- **Accelerazione:** è la variazione della velocità del veicolo nel tempo. Si misura in metri al secondo quadrato (m/s^2). Indica il livello di controllo e di adattamento del conducente alle condizioni della strada e del traffico. L'accelerazione è stata registrata usando il simulatore di guida, che calcola la variazione di velocità del veicolo nell'intervallo di tempo.
- **Forza frenante:** è la pressione applicata al pedale del freno per rallentare o fermare il veicolo. Si misura in newton (N). Indica il livello di controllo e di reattività del

conducente alle situazioni di emergenza o di pericolo. Una forza del freno elevata può indicare uno stato di stress, ansia o paura. La forza del freno è stata registrata usando il simulatore di guida, che rileva la pressione esercitata sul pedale del freno.

- **Angolo dello sterzo:** è la rotazione del volante per cambiare la direzione del veicolo. Si misura in gradi ($^{\circ}$). Indica il livello di controllo e di adattamento del conducente alle curve e agli ostacoli. L'angolo dello sterzo è stato registrato usando il simulatore di guida, che rileva la rotazione del volante.
- **Posizione nella corsia:** è la distanza tra il centro del veicolo e il centro della corsia. Si misura in metri (m). Indica il livello di controllo e di attenzione del conducente alla guida. Una posizione nella corsia elevata può indicare uno stato di distrazione, stanchezza o confusione. La posizione nella corsia è stata registrata usando il simulatore di guida, che calcola la distanza tra il centro del veicolo e il centro della corsia.

3

Capitolo 3

3.1 FASE DI PREPROCESSING

In questa sezione della tesi, verrà presentato il processo di pre-elaborazione dei dati raccolti durante l'esperimento di guida. Questi dati, provenienti da diversi driver tester, sono stati raccolti in vari scenari di guida e registrati in file CSV. Il processo di pre-elaborazione include la lettura di questi file, l'identificazione dei "test drivers", la categorizzazione delle fasi sperimentali e l'unificazione di tutti i dati in un unico DataFrame. Questo processo è fondamentale per preparare i dati per le analisi successive. Nella sezione seguente, verranno descritti in dettaglio ciascuno di questi passaggi.

3.1.1 Ambiente di sviluppo

Il progetto è stato sviluppato su un portatile ASUS X541U con le seguenti specifiche:

- Sistema Operativo: Sistema operativo a 64 bit, processore basato su x64
- Processore: Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 1.99 GHz
- RAM: 4,00 GB

E' stato utilizzato il linguaggio di programmazione Python nella versione 3.11, per analizzare e interpretare i dati. Python è noto per la sua sintassi leggibile e per la sua vasta gamma di librerie scientifiche e di analisi dei dati, che lo rendono uno strumento ideale per la ricerca accademica.

3.1.2 Parte 1: Pre-elaborazione dei dati

I file CSV scaricati dall'articolo contengono i dati dei diversi driver tester che hanno partecipato all'esperimento. Ogni file CSV corrisponde a un test driver e contiene le variabili di risposta e le variabili esplicative registrate durante le sessioni di guida. Ogni

file CSV è stato caricato in un DataFrame utilizzando la libreria pandas ed è stata aggiunta una colonna identificativa del driver tester, utilizzando il nome del file come identificativo.

Successivamente, è stata creata una nuova colonna ‘Fase’ nel DataFrame, che indica la fase sperimentale in cui si trova il test driver. Questo è stato realizzato utilizzando un contatore delle fasi e la colonna ‘Drive’ per determinare la fase corrente, assegnando un nome specifico a ogni fase in base al numero della Drive e allo stimolo presente.

```
48     def create_phase(df):
49         """Crea una nuova colonna 'Fase' nel DataFrame"""
50         # Inizializza il contatore delle fasi e la Drive corrente
51         phase_counter = 1
52         current_drive = df.loc[0, 'Drive']
53
54         # Itera su tutte le righe del dataframe
55         for i in range(len(df)):
56             # Se la Drive cambia, resetta il contatore delle fasi
57             if df.loc[i, 'Drive'] != current_drive:
58                 phase_counter = 1
59                 current_drive = df.loc[i, 'Drive']
60
61             # Altrimenti, se lo stimolo cambia e la Drive è tra 5 e 7, incrementa il contatore delle fasi
62             elif i > 0 and df.loc[i, 'Stimulus'] != df.loc[i - 1, 'Stimulus'] and df.loc[i, 'Drive'] in [5, 6, 7]:
63                 phase_counter += 1
64
65             # Se la Drive è tra 0, 1, 2, 3 o 8 assegna il nome specifico della Drive alla colonna 'Fase'
66             if df.loc[i, 'Drive'] == 1:
67                 df.loc[i, 'Fase'] = 'baseline'
68             elif df.loc[i, 'Drive'] == 2:
69                 df.loc[i, 'Fase'] = 'practice_drive'
70             elif df.loc[i, 'Drive'] == 3:
71                 df.loc[i, 'Fase'] = 'relaxing_drive'
72             elif df.loc[i, 'Drive'] == 4:
73                 df.loc[i, 'Fase'] = 'loaded_drive_NO'
74             elif df.loc[i, 'Drive'] == 8:
75                 df.loc[i, 'Fase'] = 'failure_drive'
76
77             # Assegna il valore corrente del contatore delle fasi alla colonna 'Fase'
78             else:
79                 df.loc[i, 'Fase'] = f'fase{phase_counter}'
80
81         # Aggiunge l'identificativo della Drive alla colonna 'Fase' per distinguerle meglio
82         df.loc[i, 'Fase'] += f'_Drive{df.loc[i, "Drive"]}'
83
84     return df
```

Figura 3.1: Creazione colonna Fase nel DataFrame

Infine, tutti i DataFrame sono stati uniti in un’unica struttura dati, utilizzando la funzione pd.concat e ignorando l’indice originale. Il DataFrame totale è stato salvato come un nuovo file CSV nella directory corrente. Questo DataFrame contiene tutti i dati dei ”test drivers” in tutte le fasi sperimentali e ha permesso di preparare i dati per le fasi successive del lavoro.

3.1.3 Pulizia e sistemazione dei dati:

Per preparare i dati in vista delle successive analisi, sono state eseguite alcune operazioni di pulizia e sistemazione dei dati, seguendo questi passaggi:

Selezione delle righe del DataFrame che corrispondono alle Drive 4, 5, 6 e 7, che sono le ”Loaded-Drive” in cui i driver tester hanno sperimentato le diverse condizioni di distrazione. Questo perché il lavoro su cui si è sviluppata la tesi si concentra sull’esperimento I dell’articolo, che riguarda l’effetto di stressori cognitivi, emotivi e sensorimotori sul comportamento di guida in condizioni tipiche.

Modifica dei valori nella colonna ‘Stimulus’, che indica il tipo di distrazione presente (stress associato in base alla drive e fase specifica in cui il soggetto si trova), usando un

dizionario che mappa i valori originali con i valori desiderati. Questo perché per una stessa guida, ad esempio la guida con stressore cognitivo, venivano applicati due tipi di stimoli sempre di categoria stress cognitivo ma etichettati con numeri diversi (1 e 2). Per semplificare l’analisi, questi valori vengono sostituiti con un numero univoco per ogni tipo di stress. Inoltre, è stato assegnato il valore 0 alla colonna ‘Stimulus’ per i periodi in cui non c’era nessuna distrazione, che si alternano ai periodi in cui c’è una distrazione, in tutte le drive. In questo modo, si ottiene nel DataFrame una colonna ‘Stimulus’ che assume i seguenti valori: 0 per i periodi di guida senza distrazioni (dunque la drive 4 e per i periodi nelle drive 5, 6 e 7 in cui non veniva applicato nessun tipo di stress), 1 per la guida con stress cognitivo, 3 per la guida con stress emotivo e 5 per la guida con stress sensorimotorio.

I valori di alcune variabili fisiologiche e di performance, che possono contenere errori o anomalie, vengono filtrati usando gli intervalli di validità descritti nel paper. Sono state anche effettuate alcune operazioni di pulizia e correzione dei dati, come la sostituzione dei valori di velocità vicini a zero con zero, il trattamento dei valori di velocità negativi come mancanti, la sostituzione dei valori negativi dell’accelerazione con NaN e la limitazione dei valori della forza di frenata a 300.

Il DataFrame modificato viene poi salvato come un nuovo file CSV nella directory corrente. Questo DataFrame contiene tutti i dati dei driver tester nelle quattro condizioni di distrazione.”

3.1.4 Esplorazione dei dati prima parte

Nella fase finale della pre-elaborazione, sono stati creati dei grafici per esplorare i dati raccolti dai diversi soggetti, cercando di riprodurre il più fedelmente possibile quelli presenti nel documento di riferimento da cui è stato svolto il lavoro della tesi. La libreria matplotlib di Python è stata utilizzata per tracciare i grafici di alcune variabili di interesse nel tempo, per ogni driver e per ogni drive. Sono stati selezionati 33 driver tra quelli disponibili e sono state considerate solo una parte delle variabili totali: la velocità, l’accelerazione, la forza frenante, la frequenza respiratoria, la frequenza cardiaca, l’EDA del pamo della mano e della regione perinasale. La scelta dei soggetti e delle variabili da tenere in considerazione è stata fatta dopo aver scaricato il file excel ‘Dataset-Table-Index’, che cataloga tutti i documenti del dataset nelle directory Raw Thermal Data e Structured Study Data. Questo indice non solo elenca i documenti presenti nelle due directory, ma anche quelli non presenti per motivi tecnici, quelli che non dovrebbero esserci a causa della progettazione sperimentale, quelli che sono stati redatti a causa di restrizioni IRB, quelli associati a variabili termiche derivate che non hanno potuto essere estratti a causa della presenza di peli sul viso (soggetti maschili), quelli che, nonostante siano presenti, sono stati rilevati come compromessi dal rumore durante il processo di convalida tecnica. Successivamente sono state definite delle funzioni appropriate per ogni variabile, che prendono in input il DataFrame e restituiscono dei grafici con titolo, etichette e legenda. Prima della creazione dei grafici, sono stati applicati dei filtri ai dati per eliminare o correggere dei valori anomali o errati, utilizzando la funzione filter_data definita nella seconda parte della fase di pre-elaborazione. Questa funzione utilizza gli stessi criteri di filtraggio usati nella

seconda parte, basati sugli intervalli di validità descritti nel paper e su alcune operazioni di pulizia e correzione dei dati.” Ecco un esempio dei grafici ottenuti per la ‘DRIVE 7’,

ovvero la guida in cui veniva applicato uno stress di tipo sensorimotorio:

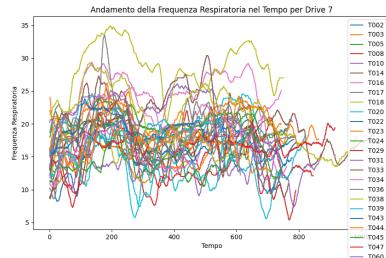


Figura 3.2: Segnale filtrato Breathing Rate

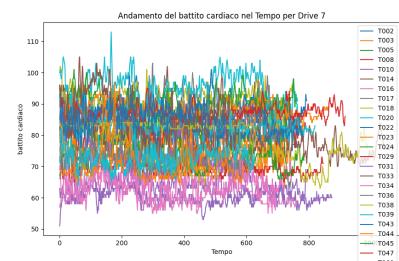


Figura 3.3: Segnale filtrato Heart Rate

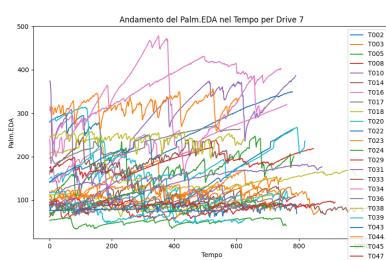


Figura 3.4: Segnale filtrato Palm EDA

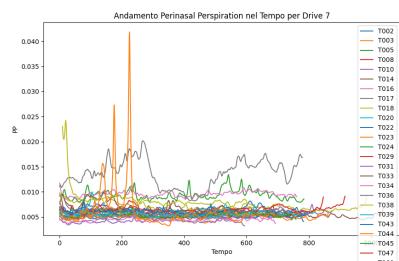


Figura 3.5: Segnale filtrato Perinasal EDA

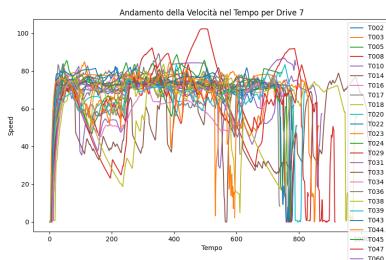


Figura 3.6: Segnale filtrato Speed

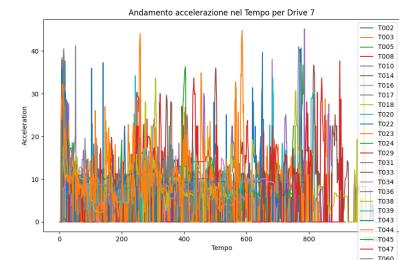


Figura 3.7: Segnale filtrato Acceleration

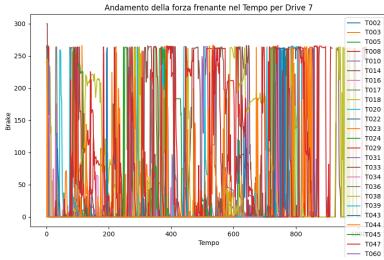


Figura 3.8: Segnale filtrato Braking

3.1.5 Esplorazione dei dati seconda parte

Nel paper di riferimento, per dimostrare che le variabili fisiologiche scelte sono valide per misurare lo stress dei conducenti in situazioni di distrazione, vengono utilizzate le seguenti formule:

$$\begin{aligned}\bar{X}_{4i} &= \frac{1}{n_{4i}} \sum_{j=1}^{n_{4i}} X_{4ij} \\ \bar{X}_{kli} &= \frac{1}{n_{kli}} \sum_{j=1}^{n_{kli}} X_{klij} \\ R_{kli} &= \bar{X}_{kli} - \bar{X}_{4i}\end{aligned}$$

Dove:

- X_{4ij} è il valore della variabile fisiologica per il driver i nella Drive 4 ('no stress added') al tempo j .
- X_{klij} è il valore della variabile fisiologica per il driver i nella Drive k ($k \in \{5, 6, 7\}$, cioè cognitive, emotional, sensorimotor) nella Fase l ($l \in \{1, 2, 3, 4, 5\}$) al tempo j .
- \bar{X}_{4i} è la media della variabile fisiologica per il driver i nella Drive 4 ('no stress added').
- \bar{X}_{kli} è la media della variabile fisiologica per il driver i nella Drive k nella Fase l .
- R_{kli} è il risultato della variabile fisiologica per il driver i nella Drive k nella Fase l , calcolato come la differenza tra la media della Drive k nella Fase l e la media della Drive 4 ('no stress added').

Le variabili fisiologiche utilizzate per questa analisi sono quattro: perinasal EDA, palm EDA, heart rate e breathing rate. Per ognuna di queste variabili, gli autori confrontano i valori medi nelle fasi con distrazioni (P2 e P4) con i valori medi nelle fasi senza distrazioni (P1 e P3). Usano dei test t accoppiati per verificare se le differenze sono statisticamente significative. I risultati mostrano che la perinasal EDA, che misura la sudorazione nella zona del naso, è la variabile più sensibile a tutti e tre i tipi di distrazioni, perché i suoi valori medi sono più alti nelle fasi con distrazioni rispetto a quelle senza. Questo significa che la perinasal EDA è un buon indicatore dello stress dei conducenti. Questi risultati vengono

mostrati con dei grafici a scatola, che rappresentano la distribuzione delle differenze tra i valori medi delle variabili fisiologiche nelle varie fasi.

Lo stesso tipo di analisi è stato fatto anche per le altre tre variabili fisiologiche: palm EDA, heart rate e breathing rate. I risultati mostrano che queste variabili sono meno sensibili alle distrazioni, perché i loro valori medi non cambiano molto tra le fasi con e senza distrazioni ma contribuiscono comunque a fornire informazioni sullo stato di stress dei conducenti. D'altra parte si può osservare che il segnale palm EDA non è un segnale significativo per rilevare lo stress dei conducenti, perché non rispecchia i cambiamenti nella conduttanza elettrica della pelle causati dall'attivazione del sistema nervoso simpatico. Questo sistema è responsabile della secrezione di sudore da parte delle ghiandole sudoripare eccrine, che sono presenti in tutto il corpo, ma in particolare nella zona del naso e delle dita. Gli autori ipotizzano che il motivo di questa scarsa performance sia dovuto all'attrito costante del sensore, che viene schiacciato tra il palmo del soggetto e il volante; questo potrebbe alterare la qualità del segnale e ridurne l'informazione. Per questo motivo, il segnale palm EDA non è affidabile come indicatore dello stress dei conducenti.

Per cercare di riprodurre nel modo più attendibile possibile questi risultati è stato realizzato un codice che agisce direttamente sul dataframe filtrato (creato nelle fasi precedenti del progetto); e fa quanto segue: Per ogni variabile fisiologica (Heart.Rate, Perina-sal.Perspiration e Breathing.Rate), calcola la media per il driver i nella Drive 4 ('no stress added') utilizzando la prima formula.

$$\bar{X}_{4i} = \frac{1}{n_{4i}} \sum_{j=1}^{n_{4i}} X_{4ij} \quad (3.1)$$

Per ogni variabile fisiologica, calcola la media per il driver i nella Drive k nella Fase l utilizzando la seconda formula.

$$\bar{X}_{kli} = \frac{1}{n_{kli}} \sum_{j=1}^{n_{kli}} X_{klij} \quad (3.2)$$

Per ogni variabile fisiologica, calcola il risultato per il driver i nella Drive k nella Fase l utilizzando la terza formula.

$$R_{kli} = \bar{X}_{kli} - \bar{X}_{4i} \quad (3.3)$$

Salva i risultati in un DataFrame e poi in un file CSV.

```

1 import pandas as pd
2 import numpy as np
3 from datetime import timedelta
4
5 # Carica il database
6 df = pd.read_csv( filepath_or_buffer: 'df_sistema_valori_esperimento1.csv', dtype=[14: str])
7
8 # Definisci le colonne che vuoi considerare
9 colonne_da_considerare = ['Failure', 'Palm.EDA', 'Heart.Rate', 'Breathing.Rate',
10                             'Perinasal.Perspiration', 'Speed', 'Acceleration', 'Brake', 'Steering']
11 # Elimina le righe con valori NA solo per le colonne specificate
12 df.dropna(subset=colonne_da_considerare, inplace=True)
13
14 # Crea una nuova colonna 'stimolo_id' che identifica univocamente ogni sequenza di stimoli
15 df['stimolo_id'] = (df['Stimulus'] != df['Stimulus'].shift()).cumsum()
16
17 # Raggruppa i dati per periodo di guida, Drive, stimolo_id, Failure
18 gruppi = df.groupby(['driver', 'Drive', 'stimolo_id', 'Failure'])
19
20 # Calcola le statistiche per ogni gruppo
21 df_stats = gruppi.agg({
22     'Heart.Rate': ['mean'],
23     'Time': ['max'], # Calcola il tempo massimo
24     'Palm.EDA': ['mean'],
25     'Breathing.Rate': ['mean'],
26     'Perinasal.Perspiration': ['mean'],
27 })

```

Figura 3.9: Risultati 1

```

29 # Rinomina le colonne
30 df_stats.columns = ['_'.join(col).strip() for col in df_stats.columns.values]
31
32 # Converti il tempo in secondi nel formato "ore:minuti:secondi"
33 df_stats['Time_max'] = df_stats['Time_max'].apply(lambda x: str(timedelta(seconds=int(x))))
34
35 df_stats = df_stats.reset_index().merge(df[['driver', 'Drive', 'stimolo_id', 'Failure', 'Stimulus', 'Fase']].drop_duplicates(),
36                                         on=['driver', 'Drive', 'stimolo_id', 'Failure'])
37
38 # Rimuovi la colonna stimolo_id e Failure
39 df_stats.drop(['stimolo_id', 'Failure'], axis=1, inplace=True)
40
41 # Definisce l'ordine delle colonne
42 colonne_ordinate = ['driver', 'Drive', 'Stimulus', 'Fase',
43                      'Heart.Rate_mean', 'Perinasal.Perspiration_mean',
44                      'Breathing.Rate_mean',
45                      'Time_max',
46                      'Palm.EDA_mean'] # Sostituisce con l'ordine delle colonne desiderato
47
48 # Riordina le colonne
49 df_stats = df_stats[colonne_ordinate]
50 """calcolo delle formule presenti nel report"""
51 # Calcola la media dell'Heart.Rate, Perinasal.Perspiration e Breathing.Rate per la Drive 4 per ogni driver
52 media_drive_4_hr = df[df['Drive'] == 4].groupby('driver')[['Heart.Rate'].mean()]
53 media_drive_4_pp = df[df['Drive'] == 4].groupby('driver')[['Perinasal.Perspiration'].mean()]
54 media_drive_4_br = df[df['Drive'] == 4].groupby('driver')[['Breathing.Rate'].mean()]
55
56 # Calcola la media dell'Heart.Rate, Perinasal.Perspiration e Breathing.Rate per le Drive 5, 6, 7 per ogni fase
57 media_drive_567_hr = df[df['Drive'].isin([5, 6, 7])].groupby(['driver',
58                                         'Drive',
59                                         'Fase'])[['Heart.Rate'].mean()]
60 media_drive_567_pp = df[df['Drive'].isin([5, 6, 7])].groupby(['driver',
61                                         'Drive',
62                                         'Fase'])[['Perinasal.Perspiration'].mean()]
63 media_drive_567_br = df[df['Drive'].isin([5, 6, 7])].groupby(['driver',
64                                         'Drive',
65                                         'Fase'])[['Breathing.Rate'].mean()]

```

Figura 3.10: Risultati 2

```

00
01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
```

```

def calcola_risultati(media_drive_4, media_drive_567):
    risultati = []
    for idx in media_drive_567.index:
        driver, drive, fase = idx
        if driver in media_drive_4.index:
            risultato = media_drive_567.loc[idx] - media_drive_4.loc[driver]
            risultati.append({'Driver': driver,
                             'Drive': drive,
                             'Fase': fase,
                             'Risultato': risultato})
    return risultati

risultati_hr = calcola_risultati(media_drive_4_hr, media_drive_567_hr)
risultati_pp = calcola_risultati(media_drive_4_pp, media_drive_567_pp)
risultati_br = calcola_risultati(media_drive_4_br, media_drive_567_br)

for res in risultati_hr:
    df_stats.loc[(df_stats['driver'] == res['Driver']) &
                 (df_stats['Drive'] == res['Drive']) &
                 (df_stats['Fase'] == res['Fase']),
                 "Risultato_HR"] = res["Risultato"]

for res in risultati_pp:
    df_stats.loc[(df_stats['driver'] == res['Driver']) &
                 (df_stats['Drive'] == res['Drive']) &
                 (df_stats['Fase'] == res['Fase']),
                 "Risultato_PP"] = res["Risultato"]

for res in risultati_br:
    df_stats.loc[(df_stats['driver'] == res['Driver']) &
                 (df_stats['Drive'] == res['Drive']) &
                 (df_stats['Fase'] == res['Fase']),
                 "Risultato_BR"] = res["Risultato"]

# Salva il DataFrame totale come un nuovo file CSV
df_stats.to_csv('df_extr_feat_ristretto_exp1.csv', index=False)
```

Figura 3.11: Risultati 3

Nella fase finale della pre-elaborazione, l'analisi esplorativa dei dati è stata condotta attraverso la creazione di boxplot per le variabili di interesse specificate precedentemente. Le funzioni implementate utilizzano la libreria matplotlib di Python e accettano come input un DataFrame contenente i dati raccolti dai driver tester. Prima della creazione dei boxplot, i dati sono stati filtrati per eliminare o correggere eventuali valori anomali o errati mediante l'applicazione della funzione filter_data, precedentemente definita nella seconda parte della fase di pre-elaborazione. Questa procedura ha garantito l'affidabilità dei risultati mostrati nei boxplot.

I **boxplot** sono dei grafici che mostrano la distribuzione dei valori di una variabile per ogni gruppo di interesse. In questo caso, i gruppi sono le diverse condizioni di guida (normal, cognitive, emotional, sensorimotor) e le diverse fasi all'interno di ogni condizione (P1, P2, P3, P4, P5).

Ogni boxplot è composto da una scatola che contiene il 50% dei valori centrali (dal 25° al 75° percentile), una linea che indica il valore mediano, due baffi che si estendono fino ai valori minimi e massimi che non sono considerati anomali, e dei punti che rappresentano i valori anomali (outliers).

```

32     def crea_boxplots_hr(data, save_dir):
33         # Crea la directory se non esiste
34         if not os.path.exists(save_dir):
35             os.makedirs(save_dir)
36         # Ottieni l'elenco unico delle Drive
37         drives = data['Drive'].unique()
38
39         # Definisci le proprietà personalizzate per il boxplot
40         boxprops = dict(linestyle='-', linewidth=1, color='black')
41         flierprops = dict(marker='o', markerfacecolor='black', markersize=3, linestyle='none')
42         medianprops = dict(linestyle='-', linewidth=2, color='red') # Cambia il colore qui
43         whiskerprops = dict(linestyle='--', linewidth=1.5, color='black')
44         capprops = dict(linestyle='--', linewidth=1.5, color='black')

```

Figura 3.12: Codice creazione boxplots

```

45     # Per ogni Drive
46     for drive in drives:
47         # Filtra i dati per la Drive corrente
48         data_drive = data[data['Drive'] == drive]
49
50         # Ottieni l'elenco unico delle Fasi per la Drive corrente
51         fasi = data_drive['Fase'].unique()
52
53         # Per ogni Fase
54         for fase in fasi:
55             # Filtra i dati per la Fase corrente
56             data_fase = data_drive[data_drive['Fase'] == fase]
57
58             # Crea un boxplot personalizzato per la Fase corrente
59             plt.figure()
60             data_fase[[ Ristultato_MR']].boxplot(boxprops=boxprops, flierprops=flierprops, medianprops=medianprops,
61                                         whiskerprops=whiskerprops, capprops=capprops)
62
63             plt.title(f'Drive {drive}, Fase {fase}')
64             plt.xlabel('Drive')
65             plt.ylabel('Ristultato HR')
66             plt.grid(axis='y', linestyle='--')
67             plt.show()
68             # Salva il grafico nella directory specificata
69             plt.savefig(os.path.join(save_dir, f'Drive_{drive}_Fase_{fase}.png'))
70
71             plt.close()
72

```

Figura 3.13: Codice creazione boxplots

Questi grafici servono a confrontare le differenze tra i gruppi in termini di posizione (media o mediana), dispersione (varianza o deviazione standard), simmetria (skewness) e presenza di valori anomali. In generale, si può dire che:

- Se le scatole sono vicine, significa che i gruppi hanno valori simili.
- Se le scatole sono lontane, significa che i gruppi hanno valori diversi.
- Se le scatole sono larghe, significa che i gruppi hanno una grande variabilità.
- Se le scatole sono strette, significa che i gruppi hanno una piccola variabilità.
- Se le scatole sono asimmetriche, significa che i gruppi hanno una distribuzione distorta verso destra o verso sinistra.
- Se ci sono molti punti, significa che i gruppi hanno molti valori anomali.

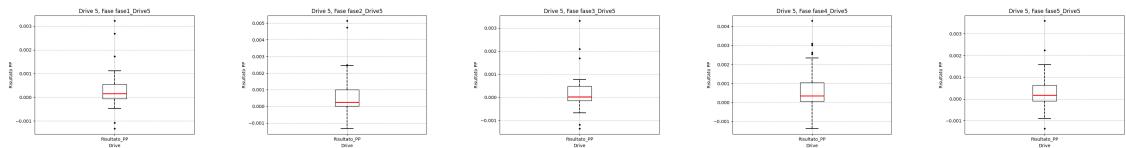


Figura 3.14: Boxplots Drive 5 'PP'

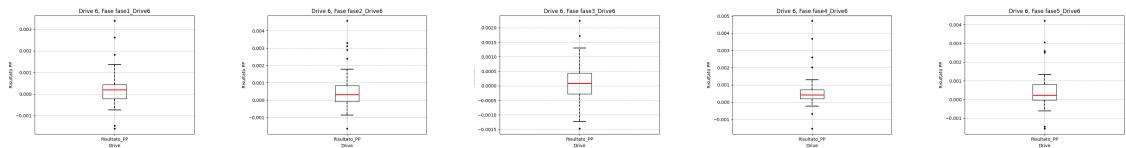


Figura 3.15: Boxplots Drive 6 'PP'

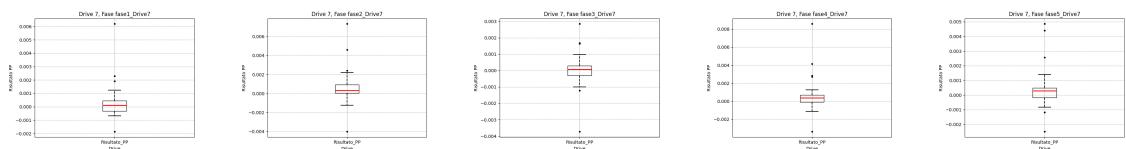


Figura 3.16: Boxplots Drive 7 'PP'

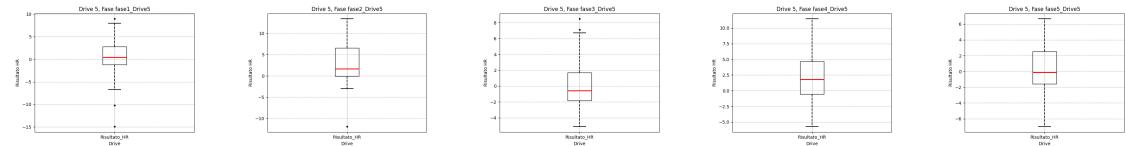


Figura 3.17: Boxplots Drive 5 'HR'

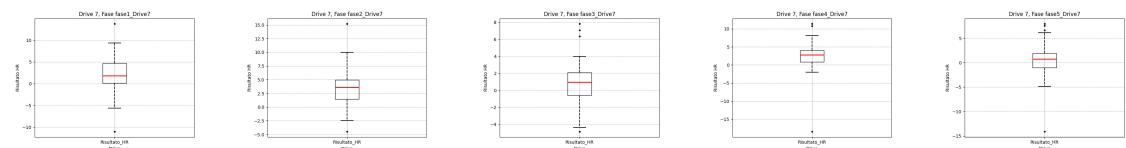


Figura 3.19: Boxplots Drive 7 'HR'

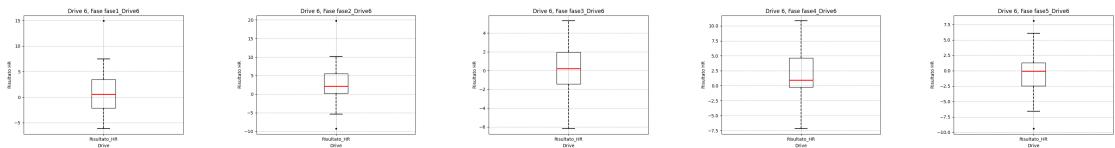


Figura 3.18: Boxplots Drive 6 'HR'

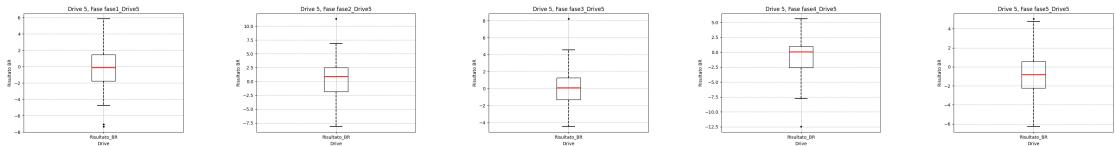


Figura 3.20: Boxplots Drive 5 'BR'

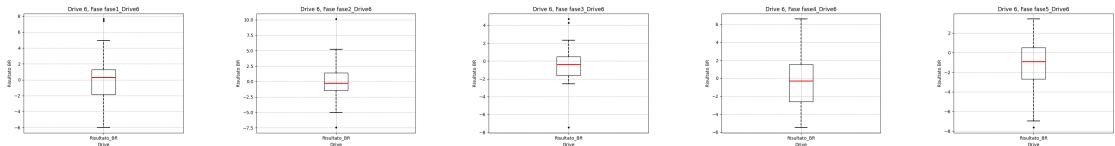


Figura 3.21: Boxplots Drive 6 'BR'

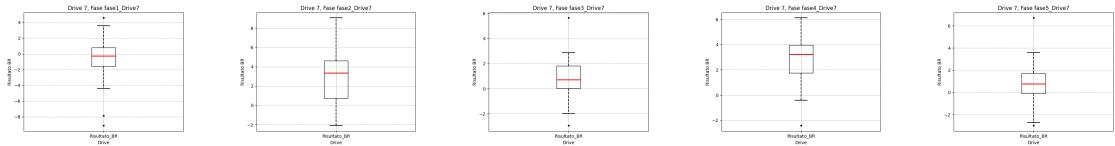


Figura 3.22: Boxplots Drive 7 'BR'

4

Capitolo 4

4.1 ESTRAZIONE DELLE FEATURES

Nella fase successiva di questo lavoro, l'attenzione si è spostata verso l'estrazione di caratteristiche significative dai dati raccolti. Questo processo è fondamentale per trasformare le misurazioni grezze in informazioni utili che possono essere utilizzate per trarre conclusioni significative. Le caratteristiche estratte includono vari descrittori sia nel dominio del tempo che nel dominio della frequenza, fornendo una visione completa dell'attività fisiologica dei conducenti durante l'esperimento. Inoltre, sono state calcolate diverse caratteristiche aggiuntive per ciascun segnale, basate su vari metodi di analisi. Queste caratteristiche aggiuntive forniscono informazioni sullo stato dei conducenti durante l'esperimento. Di seguito, vengono descritti in dettaglio i passaggi seguiti per l'estrazione delle caratteristiche.

4.2 Creazione colonna stress, per classificazione binaria

Per effettuare la classificazione binaria vengono effettuate le seguenti operazioni:

- Viene aggiunta una nuova colonna chiamata ‘stress’ al DataFrame. Questa colonna viene creata applicando una funzione lambda alla colonna ‘Stimulus’. La funzione lambda assegna il valore 1 (per rappresentare lo stress in modo generico) se il valore in ‘Stimulus’ è uno dei numeri dispari [1, 3, 5] (che rappresentano le tre tipologie di stress: cognitivo, emotivo, sensorimotorio), altrimenti assegna 0 (per rappresentare le fasi in cui i soggetti guidano in assenza di stress). Questo passaggio è utile per preparare il Dataframe alle fase successiva del progetto in cui verranno utilizzati diversi metodi di apprendimento automatico per classificare lo stato di stress dei conducenti. Generalizzando le tre diverse fasi di stress come un'unica fase è possibile mostrare le differenze tra una classificazione binaria (stress, non stress) ed una classificazione multiclass a quattro labels (non stress, stress cognitivo, emotivo, sensorimotorio).

- Viene definita una funzione replace_with_mean che sostituisce i valori nulli in un array con la media dei valori non nulli dell'array.
- Questa funzione viene in seguito applicata a diverse colonne di interesse nel DataFrame. Queste colonne sono raggruppate per ‘driver’ e ‘stress’, il che significa che la media viene calcolata separatamente per ogni combinazione delle due colonne specificate.

4.3 Features segnali fisiologici

Per ogni segnale fisiologico sono state estratte diverse features da ogni finestra osservativa di 10 secondi. Le features includono sia descrittori nel dominio del tempo che nel dominio della frequenza. Queste tecniche di estrazione delle features sono ampiamente utilizzate nell'analisi dei segnali fisiologici. Per l'estrazione delle features è stato consultato il lavoro di Bianco et al. [ref2].

Ecco un esempio che mostra il calcolo delle features per il segnale del battito cardiaco:

```

124 def calculate_heart_rate_features(group):
125     hr_array = group['Heart.Rate'].to_numpy()
126     fft = np.fft.rfft(hr_array)
127     psd = np.abs(fft) ** 2
128     peak_diffs = np.diff(signal.argrelextrema(hr_array)[0])
129     hr_diffs = np.diff(hr_array)
130
131     return pd.Series({
132         'Heart.Rate_ratio': np.mean(np.diff(hr_array) / hr_array[:-1]),
133         'Heart.Rate_HRV': create_hrv_feature(hr_array),
134         'Heart.Rate_mean': hr_array.mean(),
135         'Heart.Rate_median': np.median(hr_array),
136         'Heart.Rate_std': hr_array.std(),
137         'Heart.Rate_var': hr_array.var(),
138         'Heart.Rate_sum': hr_array.sum(),
139         'Heart.Rate_range': hr_array.ptp(),
140         'Heart.Rate_max': hr_array.max(),
141         'Heart.Rate_min': hr_array.min(),
142         'Heart.Rate_ms': np.sqrt(np.mean(hr_array**2)),
143         'Heart.Rate_entropy': stats.entropy(hr_array),
144         'Heart.Rate_iqr': stats.iqr(hr_array),
145         'Heart.Rate_psd': psd.sum(),
146         'Heart.Rate_psd_mean': psd.mean(),
147         'Heart.Rate_psd_median': np.median(psd),
148         'Heart.Rate_mean_peak_diff': np.sqrt(np.mean(np.square(peak_diffs))) if len(peak_diffs) > 0 else np.nan,
149         'Heart.Rate_std_peak_diff': peak_diffs.std() if len(peak_diffs) > 0 else np.nan,
150         'Heart.Rate_num_peak_diff_gt_50': (peak_diffs > 50).sum() if len(peak_diffs) > 0 else np.nan,
151
152         'Heart.Rate_mean_diff_sqrt': np.mean(np.sqrt(np.abs(hr_diffs))) if len(hr_diffs) > 0 else np.nan,
153         'Heart.Rate_std_diff': hr_diffs.std() if len(hr_diffs) > 0 else np.nan,
154         'Heart.Rate_num_diff_gt_50': (hr_diffs > 50).sum() if len(hr_diffs) > 0 else np.nan,
155     })

```

Figura 4.1: Features Heart Rate

- Heart.Rate_ratio: Il rapporto medio tra la differenza consecutiva e il valore del battito cardiaco.
- Heart.Rate_HRV: L'indice di variabilità del ritmo cardiaco.
- Heart.Rate_mean: La media del battito cardiaco.
- Heart.Rate_median: La mediana del battito cardiaco.
- Heart.Rate_std: La deviazione standard del battito cardiaco.
- Heart.Rate_var: La varianza del battito cardiaco.

- Heart.Rate_sum: La somma del battito cardiaco.
- Heart.Rate_range: L'intervallo del battito cardiaco.
- Heart.Rate_max: Il valore massimo del battito cardiaco.
- Heart.Rate_min: Il valore minimo del battito cardiaco.
- Heart.Rate_rms: La radice quadrata media del battito cardiaco.
- Heart.Rate_entropy: L'entropia del battito cardiaco.
- Heart.Rate_iqr: L'intervallo interquartile del battito cardiaco.
- Heart.Rate_psd: La densità spettrale di potenza del battito cardiaco.
- Heart.Rate_psd_mean: La media della densità spettrale di potenza del battito cardiaco.
- Heart.Rate_psd_median: La mediana della densità spettrale di potenza del battito cardiaco.
- Heart.Rate_mean_peak_diff: La media delle differenze tra i picchi del battito cardiaco.
- Heart.Rate_std_peak_diff: La deviazione standard delle differenze tra i picchi del battito cardiaco.
- Heart.Rate_num_peak_diff_gt_50: Il numero di differenze tra i picchi del battito cardiaco che sono maggiori di 50.
- Heart.Rate_mean_diff_sqrt: La media della radice quadrata delle differenze assolute del battito cardiaco.
- Heart.Rate_std_diff: La deviazione standard delle differenze del battito cardiaco.
- Heart.Rate_num_diff_gt_50: Il numero di differenze del battito cardiaco che sono maggiori di 50.

4.3.1 Features aggiuntive Heart Rate

Per il segnale HR è stata calcolata una feature aggiuntiva basata sull'analisi di Lomb-Scargle, che rappresenta il rapporto tra la potenza spettrale nelle bande di frequenza ulf (0.0-0.1 Hz) e hf (0.3-0.4 Hz). Questa caratteristica è stata proposta come indice di varianabilità della frequenza cardiaca (HRV). Il periodogramma di Lomb-Scargle è uno strumento statistico utilizzato per identificare la presenza di segnali periodici (cioè segnali che si ripetono a intervalli regolari) in un insieme di dati che potrebbe non essere uniformemente spaziato. Questo è particolarmente utile in situazioni in cui i dati sono raccolti in modo irregolare nel tempo, il che è comune in molte applicazioni reali.

In termini più tecnici, il periodogramma di Lomb-Scargle è un metodo per stimare la potenza del segnale a diverse frequenze, aiutando ad identificare la frequenza (o le frequenze) in cui il segnale è più forte. Questo è simile a come la trasformata di Fourier può

essere utilizzata per analizzare i segnali, ma il periodogramma di Lomb-Scargle è progettato specificamente per gestire i dati non uniformemente spaziati.

Calcolo dell'indice di variabilità della frequenza cardiaca:

- **Preparazione dei dati:** Viene creato un array di periodi e calcolate le corrispondenti frequenze angolari. Viene anche creato un array di timestamp che rappresenta i momenti in cui ogni misurazione del battito cardiaco è stata presa.
- **Calcolo del periodogramma di Lomb-Scargle:** Utilizzo della funzione `signal.lombscargle()` per calcolare il periodogramma di Lomb-Scargle del battito cardiaco. Questo permette di ottenere un array che rappresenta la potenza del segnale a ciascuna delle frequenze angolari specificate.
- **Calcolo del rapporto di potenza:** Viene calcolato il rapporto tra la somma delle potenze alle prime 8 frequenze e la somma delle potenze alle frequenze da 15 in poi. Questo rapporto è una misura della variabilità della frequenza cardiaca (HRV), che è un indicatore importante della salute del cuore.
- **Gestione degli errori:** Se si verifica un errore di divisione per zero (che potrebbe accadere se tutte le potenze sono zero, ad esempio), viene stampato un messaggio di errore e viene restituita la media del battito cardiaco come risultato.

4.3.2 Features aggiuntive Breathing Rate

Per il segnale BR sono state calcolate quattro features aggiuntive basate sulla potenza spettrale nelle bande di frequenza ulf, vlf, lf e hf. La potenza spettrale nelle diverse bande di frequenza può fornire informazioni preziose sulla regolazione autonoma della respirazione. Il metodo di Welch è un approccio utilizzato per stimare la potenza di un segnale a diverse frequenze, che è particolarmente utile quando i dati sono raccolti in modo non uniforme nel tempo.

Ecco una spiegazione del metodo di Welch:

- **Divisione del segnale:** Il segnale viene diviso in segmenti più piccoli, che possono sovrapporsi. Questo aiuta a ridurre il rumore nelle stime della potenza del segnale.
- **Applicazione della finestra:** Ogni segmento viene moltiplicato per una “finestra”, che è una funzione che vale 1 al centro e si riduce gradualmente a 0 ai bordi. Questo riduce l'effetto dei bordi del segmento sulle stime della potenza del segnale.
- **Calcolo del periodogramma:** Per ogni segmento, si calcola il periodogramma, che è una stima della potenza del segnale a diverse frequenze. Questo si ottiene calcolando la trasformata di Fourier del segmento (che converte il segnale dal dominio del tempo al dominio della frequenza) e poi prendendo il modulo quadro del risultato.
- **Media dei periodogrammi:** Infine, si calcola la media dei periodogrammi di tutti i segmenti. Questo riduce ulteriormente il rumore nelle stime della potenza del segnale, ma riduce anche la risoluzione in frequenza (cioè la capacità di distinguere tra frequenze vicine).

In sostanza, il metodo di Welch è un compromesso tra la riduzione del rumore nelle stime della potenza del segnale e una minore risoluzione in frequenza. Il calcolo del periodogramma di Welch è spiegato nelle seguenti fasi:

Calcolo del periodogramma di Welch:

- Utilizzo della funzione `signal.welch()` per calcolare il periodogramma di Welch del segnale. Questo permette di ottenere due array: `frequencies`, che rappresenta le diverse frequenze a cui il segnale ha potenza, e `powers`, che rappresenta la potenza del segnale a ciascuna di queste frequenze.
- Definizione delle bande di frequenza: Definisce un dizionario `fbands` che mappa i nomi delle bande di frequenza (`ulf`, `vlf`, `lf`, `hf`) ai loro rispettivi intervalli di frequenza.
- Calcolo della potenza spettrale in una specifica banda di frequenza: Infine, si somma la potenza del segnale a tutte le frequenze che rientrano nella banda di frequenza specificata (ad esempio, `ulf`, `vlf`, `lf`, `hf`). Questo viene fatto utilizzando l'operatore `np.where()` per selezionare solo le potenze che corrispondono alle frequenze nella banda di frequenza specificata.

4.3.3 Features aggiuntive EDA palmare

Per il segnale P-EDA, sono state calcolate quattro features aggiuntive basate sull'analisi dei picchi di sudorazione. Queste features sono: la frequenza dei picchi, la magnitudine dei picchi, la durata dei picchi e l'area dei picchi.

- Frequenza dei picchi: Questa feature viene calcolata come il numero totale di picchi nel segnale P-EDA. Si ottiene semplicemente contando il numero di picchi trovati nel segnale.
- Magnitudine dei picchi: Questa feature viene calcolata sommando le altezze di tutti i picchi nel segnale P-EDA. L'altezza di un picco è la differenza tra il valore del picco e l'altezza a metà larghezza del picco.
- Durata dei picchi: Questa feature viene calcolata sommando le larghezze di tutti i picchi nel segnale P-EDA. La larghezza di un picco è la differenza tra la posizione del picco e la posizione del bordo sinistro del picco.
- Area dei picchi: Questa feature viene calcolata sommando le aree di tutti i picchi nel segnale P-EDA. L'area di un picco è la metà del prodotto dell'altezza del picco (calcolata come sopra) e la larghezza del picco.

L'analisi dei picchi di sudorazione può fornire informazioni preziose sull'attivazione del sistema nervoso simpatico in risposta allo stress.

4.3.4 Features EDA perinasale

Per i segnali PER-EDA, sono state utilizzate le stesse features estratte per il segnale HR, senza la feature aggiuntiva basata sull'analisi di Lomb-Scargle. Questa scelta è stata fatta in quanto i segnali PER-EDA hanno una banda di frequenza inferiore a quella del segnale

HR, e quindi l'analisi di Lomb-Scargle non è applicabile.

Per il calcolo delle features aggiuntive relative alla frequenza respiratoria (BR), all'attività elettrodermica della palma (P-EDA) e alla variabilità della frequenza cardiaca (HRV) è stato utilizzato come riferimento il codice sorgente di un progetto che ha affrontato il problema del riconoscimento delle emozioni del conducente di un'auto, utilizzando un database basato sulle guide ma diverso da quello su cui è sviluppato questo progetto. Il progetto "Predicting Driver Stress Using Deep Learning" (<https://github.com/KryeKuzhinieri/predicting-driver-stress-using-deep-learning>), sviluppato da KryeKuzhinieri mira a prevedere i livelli di stress del conducente utilizzando il dataset SRAD (drivedb) di Physionet con metodi come LSTM (Long Short-Term Memory), RNN (Recurrent Neural Networks) e CNN (Convolutional Neural Networks).

Il progetto si basa principalmente sul lavoro di Healey, intitolato "Detecting Stress During Real-World Driving Tasks Using Physiological Sensors". Il dataset utilizzato contiene input raccolti da segnali fisiologici come l'elettrocardiogramma, l'elettromiogramma e la risposta galvanica cutanea.

Il progetto include diverse fasi, tra cui la conversione del dataset, la visualizzazione dei dati di base per ogni guida, la pre-elaborazione dei dati, la selezione delle caratteristiche e l'applicazione di vari modelli di apprendimento.

Infine, il progetto mira a prevedere i livelli di stress utilizzando varie tecniche di riduzione con l'aiuto di modelli di apprendimento come LSTM, CNN e RNN.

4.4 Features segnali telemetrici

In questo studio, sono state anche estratte delle features dai segnali telemetrici relativi alla guida, come velocità, sterzata, frenata e accelerazione. Queste features sono state calcolate utilizzando delle funzioni definite in Python, che operano su gruppi di dati. Di seguito viene fornita una descrizione dettagliata delle caratteristiche per ogni segnale.

- **Velocità:** Per il segnale di velocità, sono state calcolate le seguenti features: media, deviazione standard, minimo, massimo, range, varianza e differenza tra il massimo e il minimo (indicata come "Speed_change").
- **Sterzata:** Per il segnale di sterzata, sono state calcolate le stesse features della velocità, con l'aggiunta della differenza tra il massimo e il minimo (indicata come "Steering_change").
- **Frenata:** Per il segnale di frenata, oltre alle features calcolate per la velocità, sono state calcolate la somma totale dei valori di frenata (indicata come "Breaking_sum") e il numero di volte che la forza di frenata supera la media del segmento (indicata come "Breaking_freq").
- **Accelerazione:** Per il segnale di accelerazione, sono state calcolate le stesse features della velocità, con l'aggiunta di features relative al jerk (la derivata dell'accelerazione rispetto al tempo), che includono: media del jerk, deviazione standard del jerk, minimo del jerk, massimo del jerk, range del jerk, varianza del jerk.

Un esempio di come sono state calcolate le features per i segnali telemetrici:

```
106 def calculate_acceleration_features(group):
107     array = group['Acceleration'].to_numpy()
108     jerk = np.diff(array) if len(array) > 1 else np.array([])
109     return pd.Series({
110         'Acceleration_mean': array.mean() if len(array) > 0 else np.nan,
111         'Acceleration_std': array.std() if len(array) > 0 else np.nan,
112         'Acceleration_min': array.min() if len(array) > 0 else np.nan,
113         'Acceleration_max': array.max() if len(array) > 0 else np.nan,
114         'Acceleration_range': array.ptp() if len(array) > 0 else np.nan,
115         'Acceleration_var': array.var() if len(array) > 0 else np.nan,
116         'Acceleration_Jerk_mean': jerk.mean() if len(jerk) > 0 else np.nan,
117         'Acceleration_Jerk_std': jerk.std() if len(jerk) > 0 else np.nan,
118         'Acceleration_Jerk_min': jerk.min() if len(jerk) > 0 else np.nan,
119         'Acceleration_Jerk_max': jerk.max() if len(jerk) > 0 else np.nan,
120         'Acceleration_Jerk_range': jerk.ptp() if len(jerk) > 0 else np.nan,
121         'Acceleration_Jerk_var': jerk.var() if len(jerk) > 0 else np.nan,
122     })
```

Figura 4.2: Features Accelerazione

Queste caratteristiche possono essere utilizzate per caratterizzare il comportamento dei guidatori e per identificare eventuali correlazioni tra le features e le condizioni di guida. Tuttavia, l'elaborazione dei segnali telemetrici presenta delle sfide, come la presenza di rumore, di outlier, di dati mancanti dovuti alla registrazione o alla trasmissione dei dati. Inoltre queste variabili estratte dai segnali telemetrici, possono non riflettere direttamente la condizione di stress del conducente. Questi segnali sono influenzati da molti fattori, tra cui il tratto di strada percorso nel simulatore, le azioni del conducente come la partenza del veicolo o la pressione sul pedale che si riflette sulla forza frenante quando il veicolo è fermo. Questo rende più difficile l'interpretazione di queste features rispetto a quelle estratte dai segnali fisiologici, che sono più direttamente correlate allo stress del conducente.

4.5 Applicazione delle features ai sottogruppi di interesse

Per raggruppare i dati in base a diversi criteri e applicare le funzioni di calcolo delle features definite nel codice, è stato utilizzato il metodo groupby della libreria pandas. Il metodo groupby ha permesso di dividere il DataFrame in sottogruppi in base a uno o più elementi dell'indice, e di applicare una funzione di aggregazione a ciascun sottogruppo. I dati sono stati raggruppati in base a tre elementi dell'indice: il driver, lo stress e il Time_Interval. Questa scelta è stata fatta per poter analizzare le differenze tra i driver, tra le condizioni di stress e non-stress, e tra gli intervalli di tempo. Il Time_Interval è stato utilizzato come elemento di raggruppamento per avere una finestra temporale fissa di 10 secondi per ogni sottogruppo, in modo da poter confrontare le features estratte in modo omogeneo. Per creare il Time_Interval, sono state create prima due colonne, il Group_ID e il Time_Reset. Il Group_ID è una colonna che identifica il numero progressivo di ogni Drive, cioè ogni sessione di guida di un driver. Il Time_Reset è una colonna che azzera il tempo all'inizio di ogni Drive, in modo da poter calcolare il Time_Interval in base al tempo trascorso dall'inizio della sessione di guida. Queste due colonne sono state create utilizzando il metodo diff, che calcola la differenza tra i valori consecutivi di una colonna, il metodo ne, che verifica se i valori sono diversi da zero, il metodo any, che verifica se ci sono

valori diversi da zero lungo un asse, il metodo cumsum, che calcola la somma cumulativa dei valori, e il metodo transform, che applica una funzione a ciascun gruppo e restituisce un oggetto con lo stesso indice del DataFrame originale.

Per creare il Time_Interval, il Time_Reset è stato diviso per 10, il risultato è stato arrotondato all'intero inferiore, moltiplicato per 10, e convertito in un formato orario. Questo è stato fatto utilizzando il metodo apply, che applica una funzione a ciascun valore di una colonna, e una funzione lambda, che definisce una funzione anonima in una sola riga. La funzione lambda utilizzata prende in input un valore x, che rappresenta il numero di secondi trascorsi dall'inizio della sessione di guida, e restituisce una stringa che rappresenta il formato orario di x, usando la funzione format per inserire i valori delle ore, dei minuti e dei secondi.

Dopo aver creato il Time_Interval, i dati sono stati raggruppati in base ai tre elementi dell'indice: driver, stress e Time_Interval. Questo è stato fatto utilizzando il metodo groupby, passando la lista di colonne scelte come argomento. Questo ha restituito un oggetto GroupBy, che rappresenta una collezione di sottogruppi. A questo oggetto è stato applicato il metodo apply, passando come argomento la funzione calculate_features definita nel codice. Questa funzione calcola le features per ciascun sottogruppo, utilizzando le funzioni specifiche per ogni segnale fisiologico e telemetrico.

Infine, l'indice del DataFrame è stato resettato, in modo da avere le colonne dell'indice come colonne normali, e il DataFrame è stato salvato in un file csv, utilizzando il metodo to_csv. Questo file contiene le features che saranno utilizzate per la classificazione dello stress dei "test drivers".

```

251     def calculate_features(group):
252         heart_rate_features = calculate_heart_rate_features(group)
253         breathing_rate_features = calculate_breathing_rate_features(group)
254         palm_edo_features = calculate_palm_edo_features(group)
255         perinasal_perspiration_features = calculate_perinasal_perspiration_features(group)
256         speed_features = calculate_speed_features(group)
257         acceleration_features = calculate_acceleration_features(group)
258         brake_features = calculate_brake_features(group)
259         steering_features = calculate_steering_features(group)
260
261         return pd.concat([
262             [heart_rate_features, breathing_rate_features, palm_edo_features, perinasal_perspiration_features,
263              speed_features, acceleration_features, brake_features, steering_features
264             ])
265
266 df = pd.read_csv( filepath_or_buffer='df_sistema_valori_esperimento2.csv', dtype={14: str})
267 df['Group_ID'] = (df[['Drive']].diff().ne(0).any(axis=1)).cumsum()
268 df['Time_Reset'] = df.groupby('Group_ID')[ 'Time'].transform(lambda x: x - x.min())
269 df['Time_Interval'] = (df['Time_Reset'] // 10).astype(int) * 10
270 df['Time_Interval'] = df['Time_Interval'].apply(
271     lambda x: '{:0d} {:0d} {:0d}'.format(*args: x // 3600, (x // 60) % 60, x % 60))
272 ...
273
274 df['Group_ID'] = (df[['Drive']].diff().ne(0).any(axis=1)).cumsum()
275 df['Time_Reset'] = df.groupby('Group_ID')[ 'Time'].transform(lambda x: x - x.min())
276 df['Time_Start'] = (df['Time_Reset'] // 15).astype(int) * 15
277 df['Time_Start'] = df['Time_Start'].apply(
278     lambda x: '{:0d} {:0d} {:0d}'.format(x // 3600, (x // 60) % 60, x % 60))
279 df['Time_End'] = ((df['Time_Reset'] // 15).astype(int) * 15) + 30
280 df['Time_End'] = df['Time_End'].apply(
281     lambda x: '{:0d} {:0d} {:0d}'.format(x // 3600, (x // 60) % 60, x % 60))'''
282
283 gruppi = df.groupby(['driver', 'stress', 'Time_Interval'])
284
285 df_stats = gruppi.apply(calculate_features)
286 df_stats = df_stats.reset_index()
287 df_stats.to_csv( path_or_buf='driver_stress_Time2.csv', index=False),

```

Figura 4.3: Applicazione delle features ai sottogruppi di interesse

4.6 Post estrazione e filtraggio dati

Nel lavoro di questa tesi, dopo aver calcolato le features per ciascun sottogruppo, è stata effettuata un’ulteriore operazione di filtraggio dei dati. Questa operazione è stata necessaria per gestire i valori nulli che si erano generati in alcune righe del DataFrame, a causa di errori di acquisizione o di calcolo delle features. I valori nulli possono compromettere la qualità dei dati e compromettere la classificazione dello stress, quindi la decisione è stata quella di sostituirli con una media particolare.

Per fare questo, è stata utilizzata la funzione `replace_with_mean` definita nel codice. Questa funzione prende in input una serie di valori e restituisce la stessa serie con i valori nulli sostituiti con la media dei valori non nulli. La media è stata calcolata escludendo i valori nulli, in modo da non alterare il valore reale. Questa funzione è stata applicata a tutte le colonne del DataFrame che contenevano le features, tranne che per le colonne che contenevano le informazioni sul driver, sulla Drive, sullo stress e sugli intervalli di tempo. Queste colonne sono state escluse perché non contenevano valori rilevanti per il calcolo della media.

La funzione `replace_with_mean` è stata applicata utilizzando il metodo `transform`, che applica una funzione a ciascun gruppo e restituisce un oggetto con lo stesso indice del DataFrame originale. I dati sono stati raggruppati in base al driver e allo stress, in modo da calcolare la media in modo specifico per ciascuna combinazione di driver e condizione di stress. Questa scelta è stata fatta per preservare la variabilità dei dati tra i diversi driver e tra le diverse condizioni di stress e inoltre per evitare di introdurre valori non coerenti con la realtà.

Dopo aver sostituito i valori nulli con la media, sono state eliminate le colonne che contenevano ancora valori nulli, utilizzando il metodo `dropna`. Queste colonne erano quelle relative alle features basate sui picchi di sudorazione, che non erano state calcolate per alcuni sottogruppi a causa della mancanza di picchi nel segnale P-EDA. Queste colonne sono state eliminate perché non erano significative per la classificazione dello stress, e perché avrebbero potuto introdurre rumore o distorsione nel modello.

Infine, il DataFrame modificato è stato salvato in un nuovo file csv, utilizzando il metodo `to_csv`. Questo file contiene le features che saranno utilizzate per la classificazione dello stress dei driver, senza valori nulli.

```
39 def replace_with_mean(x):
40     mean = x[(x.notna())].mean() # Calcola la media escludendo i valori nulli
41     x[(x.isna())] = mean # Sostituisce i valori nulli o zero con la media
42     return x
43
44 # Crea una lista delle colonne da escludere
45 exclude_columns = ['driver', 'Time_Interval', 'stress']
46 # Ottieni una lista di tutte le colonne nel DataFrame
47 all_columns = df.columns.tolist()
48 # Usa la differenza tra le liste per ottenere le colonne su cui operare
49 columns_to_include = [col for col in all_columns if col not in exclude_columns]
50 # Ora columns_to_include può essere incluso nel ciclo for
51 for column in columns_to_include:
52     df[column] = df.groupby(['driver', 'stress'])[column].transform(replace_with_mean)
53
54 df = df.dropna(axis=1)
55
56
57
58 # Salva il DataFrame modificato in un nuovo file CSV
59 df.to_csv( path_or_buf: 'driver_stress_Time_train2.csv', index=False)
```

Figura 4.4: Filtraggio dati post estrazione features

5

Capitolo 5

5.1 CLASSIFICATORI UTILIZZATI

Nel lavoro di questa tesi, sono stati utilizzati diversi classificatori per assegnare la classe di stress o non-stress a ciascun sottogruppo di dati, in base alle features estratte dai segnali fisiologici e telemetrici. Sono stati definiti tre classificatori 'base' nel codice, basati su singoli algoritmi di apprendimento, e sei classificatori 'ottimizzati' nel codice, basati su algoritmi di apprendimento che richiedono l'ottimizzazione di alcuni parametri. Di seguito vengono descritti i classificatori appena citati, spiegando cosa fa ciascuno di essi e quali parametri sono stati scelti o ottimizzati.

5.1.1 Classificatori base

- **KNN:** Questo classificatore si basa sul metodo k-NN (k-vicini), assegnando a ciascun campione la classe più comune tra i suoi k vicini. In questo caso, è stato scelto $k=1$, cioè il classificatore si basa sul solo vicino più prossimo di ogni campione. È stata anche scelta la distanza euclidea come misura di similarità tra i campioni, e i vicini sono stati pesati in base alla loro distanza inversa, in modo da dare più importanza ai vicini più prossimi.

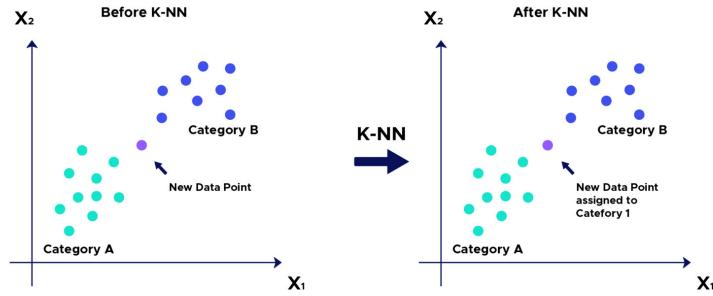


Figura 5.1: K-Nearest Neighbors (K-NN)

- **SVM:** Questo è un classificatore basato sul metodo delle macchine a vettori di supporto (SVM), che cerca di separare le classi con un iperpiano che massimizza il margine tra i campioni più vicini all'iperpiano stesso, detti vettori di supporto. In questo caso viene utilizzato un kernel Radial Basis Function (RBF), che trasforma lo spazio delle caratteristiche in uno spazio di dimensione maggiore, in cui è più facile trovare una separazione lineare tra le classi. Sono stati anche scelti due parametri, C e gamma, che controllano rispettivamente il compromesso tra la larghezza del margine, il numero di errori di classificazione e l'ampiezza della funzione radiale. I valori ottimali di questi parametri devono essere trovati tramite una ricerca nella griglia, cioè provando diverse combinazioni di valori e scegliendo quella che ottiene il miglior risultato sul set di validazione.

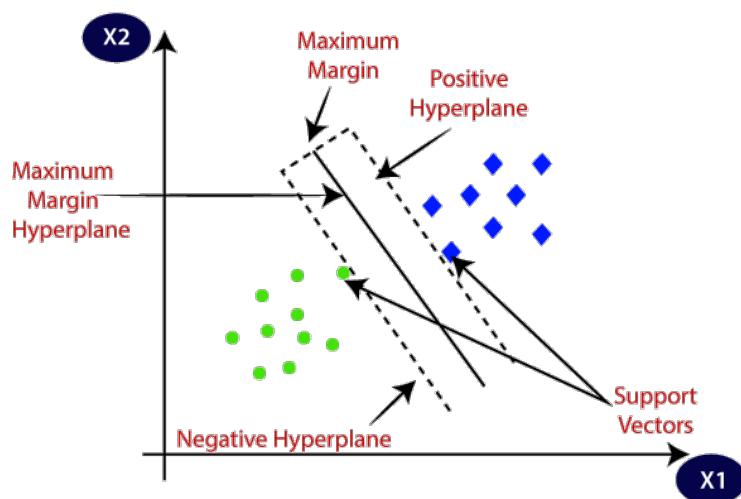


Figura 5.2: Support Vector Machine (SVM)

- **ANN:** Questo è un classificatore basato sul metodo delle reti neurali artificiali (ANN), che apprende una funzione non lineare che mappa le caratteristiche di ingresso con le classi di uscita tramite una serie di strati di neuroni artificiali che

trasmettono segnali tra loro. In questo caso è stata scelta un'architettura superficiale, ossia con pochi strati nascosti che vengono allenati in modo non supervisionato come autoencoder, cioè come modelli che cercano di ricostruire i dati di ingresso minimizzando l'errore di ricostruzione. Il numero di strati nascosti e il numero di neuroni in ciascuno di essi vengono selezionati mediante una ricerca a griglia, cioè provando diverse configurazioni e scegliendo quella che ottiene il miglior risultato sul set di validazione. Lo strato finale è uno strato softmax, che assegna una probabilità a ogni classe di uscita in base al segnale ricevuto dagli strati precedenti.

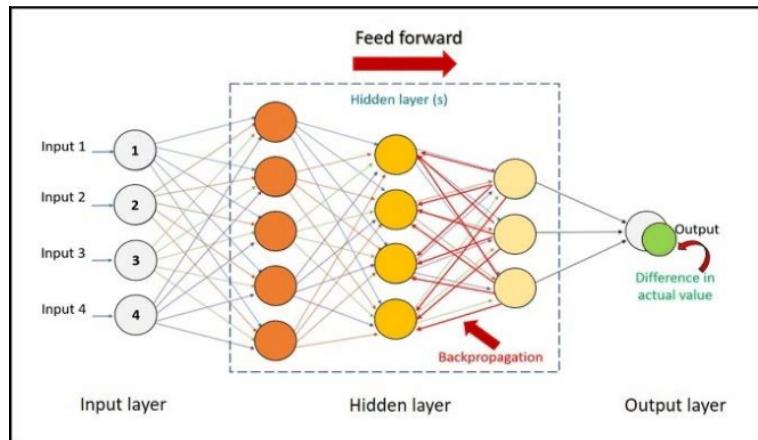


Figura 5.3: Artificial Neural Network

```

106 def define_ilknn():
107     # ILK-NN con distanza euclidea e K=1
108     ilknn = KNeighborsClassifier(n_neighbors=1, metric='euclidean', weights='distance')
109     return ilknn
110
111 def define_svm():
112     # SVM con kernel Radial Basis Function (RBF)
113     # I valori ottimali di C e gamma devono essere trovati tramite la ricerca nella griglia
114     svm = SVC(kernel='rbf', C=1.0, gamma='scale') # Valori di C e gamma sono esempi
115     return svm
116
117 def define_ann():
118     # ANN con architettura superficiale
119     # Il numero di strati nascosti e il numero di neuroni in ciascuno di essi vengono selezionati mediante ricerca a griglia
120     # Se M1 = 0 o M2 = 0, la ANN risulta ha un solo strato nascosto. Se M1 = M2 = 0, la ANN risultante non ha livelli nascosti
121     ann = MLPClassifier(hidden_layer_sizes=(50,)) # Il valore 50 è un esempio
122     return ann

```

Figura 5.4: Classificatori base

5.1.2 Classificatori ottimizzati

I classificatori ottimizzati che sono stati definiti nel codice sono:

- **Naive Bayes:** Questo è un classificatore basato sul teorema di Bayes, che calcola la probabilità di appartenenza a una classe in base alla frequenza delle caratteristiche nei dati di allenamento. È stato scelto di ottimizzare il parametro var_smoothing, che aggiunge una piccola quantità alla varianza dei dati per evitare calcoli instabili. È stato utilizzato gridsearch per trovare il valore ottimale di questo parametro, provando i valori [1e-9, 1e-8, 1e-7, 1e-6, 1e-5].

Naive Bayes

thatware.co

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

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

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

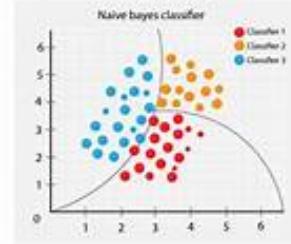


Figura 5.5: Naive Bayes (NB))

- **Logistic Regression:** Questo è un classificatore basato su una funzione logistica, che apprende i pesi da assegnare alle caratteristiche in modo da minimizzare l'errore di classificazione. È stato scelto di ottimizzare i parametri C e penalty, che controllano rispettivamente il grado di regolarizzazione e il tipo di norma da applicare ai pesi per evitare il sovraccarico. È stato utilizzato gridsearch per trovare i valori ottimali di questi parametri, provando le combinazioni di $C=[0.001, 0.01, 0.1, 1, 10, 100]$ e $\text{penalty}=['l1', 'l2']$.

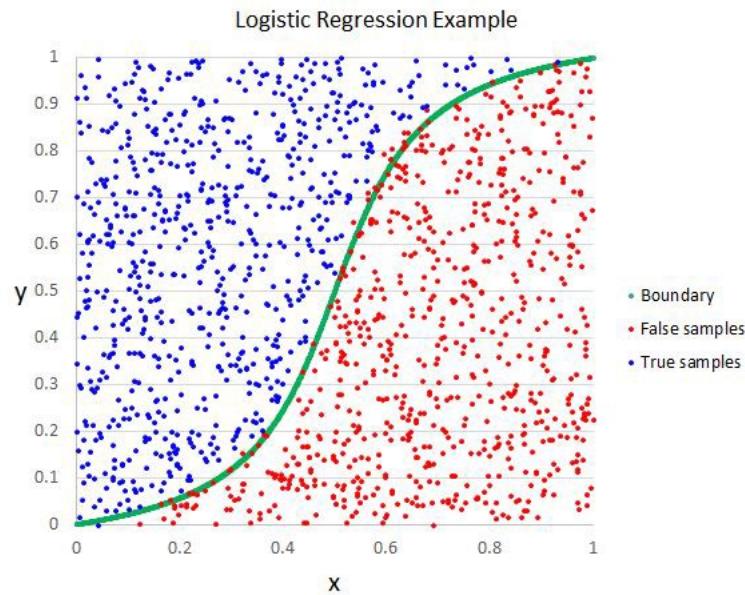


Figura 5.6: Logistic Regression (LR)

- **KNN 'ottimizzato':** Questo è un classificatore basato sul metodo dei k-vicini più prossimi (k-NN), che assegna a ogni campione la classe più frequente tra i suoi k vicini più prossimi. È stato scelto di ottimizzare i parametri n_neighbors e weights, che controllano rispettivamente il numero di vicini da considerare e il modo di pesare

i vicini in base alla loro distanza. È stato utilizzato gridsearch per trovare i valori ottimali di questi parametri, provando le combinazioni di `n_neighbors=[3, 5, 7, 9]` e `weights=['uniform', 'distance']`. È stata anche scelta la metrica ‘manhattan’ come misura di similarità tra i campioni, che calcola la somma delle differenze assolute tra le coordinate.

- **SVM ‘ottimizzato’:** Questo è un classificatore basato sul metodo delle macchine a vettori di supporto (SVM), che cerca di separare le classi con un iperpiano che massimizza il margine tra i campioni più vicini all’iperpiano stesso, detti vettori di supporto. È stato scelto di ottimizzare i parametri `C` e `gamma`, che controllano rispettivamente il compromesso tra la larghezza del margine e il numero di errori di classificazione, e l’ampiezza della funzione radiale. È stato utilizzato gridsearch per trovare i valori ottimali di questi parametri, provando le combinazioni di `C=[0.01, 0.1, 1, 10]` e `gamma=[1, 0.1, 0.01, 0.001]`. È stato anche scelto di usare il kernel ‘rbf’, che trasforma lo spazio delle caratteristiche in uno spazio di dimensione maggiore, in cui è più facile trovare una separazione lineare tra le classi.
- **Decision Tree:** Questo è un classificatore basato su una struttura ad albero, che divide ricorsivamente i dati in base a dei criteri di scelta delle caratteristiche, fino a raggiungere dei nodi foglia che assegnano la classe di uscita. È stato scelto di ottimizzare i parametri `max_depth` e `min_samples_split`, che controllano rispettivamente la profondità massima dell’albero e il numero minimo di campioni necessari per effettuare una divisione. È stato utilizzato gridsearch per trovare i valori ottimali di questi parametri, provando le combinazioni di `max_depth=[None, 2, 4, 6]` e `min_samples_split=[2, 3]`. È stato anche scelto di usare il criterio ‘entropy’ per scegliere le caratteristiche, che misura il grado di disordine dei dati.

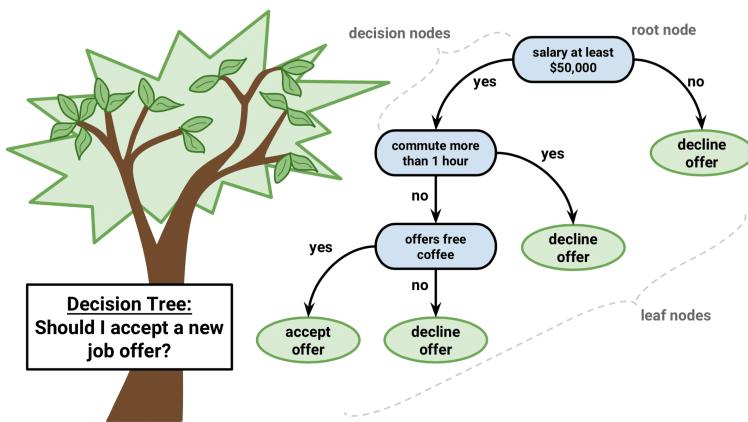


Figura 5.7: Decision Tree

- **Random Forest:** Questo è un classificatore basato su un insieme di alberi decisionali, che genera una predizione media a partire dalle predizioni dei singoli alberi. È stato scelto di ottimizzare il parametro `n_estimators`, che controlla il numero di alberi da generare. È stato utilizzato gridsearch per trovare il valore ottimale di questo parametro, provando i valori [10, 20, 30, 40]. È stato anche scelto di usare

il criterio ‘entropy’ per scegliere le caratteristiche, e di usare la radice quadrata del numero di caratteristiche come numero massimo di caratteristiche da considerare per ogni albero.

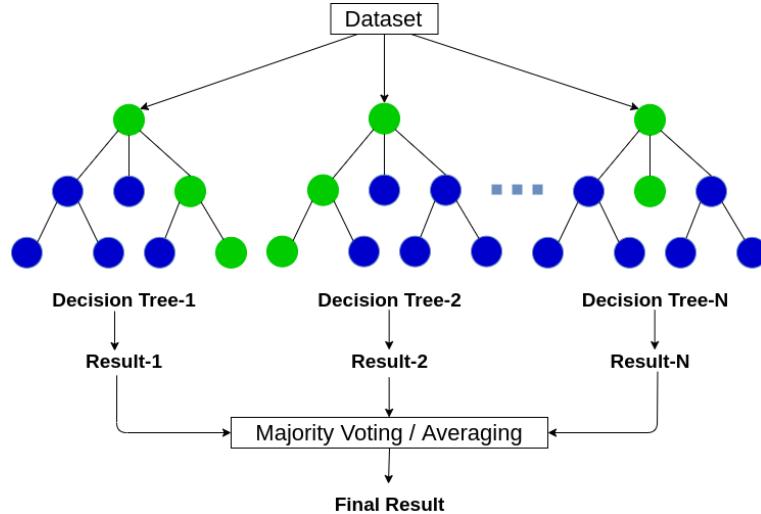


Figura 5.8: Random Forest

5.1.3 Gridsearch

Per trovare i valori ottimali dei parametri dei classificatori ‘ottimizzati’, è stata utilizzata la tecnica del gridsearch, che consiste nel provare diverse combinazioni di valori e scegliere quella che ottiene il miglior risultato sul set di validazione. Per usare il gridsearch è stato definito un dizionario di parametri da ottimizzare per ogni classificatore, contenente i possibili valori da testare per ogni parametro, e poi è stato creato un oggetto GridSearchCV, passando come argomenti il classificatore da ottimizzare e il dizionario di parametri. L’oggetto GridSearchCV si occupa di eseguire la ricerca nella griglia e di restituire il classificatore con i parametri ottimali.

```

32     def naive_bayes():
33         # Definisci i parametri da ottimizzare per Naive Bayes
34         parametri_nb = {'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5]}
35         # Crea un classificatore Naive Bayes
36         nb = GaussianNB()
37         # Crea l'oggetto GridSearchCV per Naive Bayes
38         grid_nb = GridSearchCV(nb, parametri_nb, refit=True)
39         return grid_nb
40
41     def logistic_regression():
42         # Definisci i parametri da ottimizzare per Logistic Regression
43         parametri_lr = {'C': [0.001, 0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}
44         # Crea un classificatore Logistic Regression
45         lr = LogisticRegression(solver='liblinear')
46         # Crea l'oggetto GridSearchCV per Logistic Regression
47         grid_lr = GridSearchCV(lr, parametri_lr, refit=True)
48         return grid_lr
49
50
51
52     def kn():
53         # Definisci i parametri da ottimizzare per K Neighbors Classifier
54         parametri_kn = {'n_neighbors': [3, 5, 7, 9], 'weights': ['uniform', 'distance']}
55         # Crea un classificatore K Neighbors Classifier
56         kn = KNeighborsClassifier(leaf_size=1, metric='manhattan')
57         # Crea l'oggetto GridSearchCV per K Neighbors Classifier
58         grid_kn = GridSearchCV(kn, parametri_kn, refit=True)
59         return grid_kn
60
61

```

Figura 5.9: 1. Classificatori ottimizzati

```

63     def svm():
64         # Definisci i parametri da ottimizzare per SVM
65         parametri_svm = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001]}
66         # Crea un classificatore SVM con kernel RBF
67         svm = SVC(kernel='rbf', probability=True)
68         # Crea l'oggetto GridSearchCV per SVM
69         grid_svm = GridSearchCV(svm, parametri_svm, refit=True)
70         return grid_svm
71
72
73     def decision_tree():
74         # Definisci i parametri da ottimizzare per Decision Tree Classifier
75         parametri_dt = {'max_depth': [None, 2, 4, 6], 'min_samples_split': [2, 3]}
76         # Crea un classificatore Decision Tree Classifier
77         dt = DecisionTreeClassifier(criterion='entropy', splitter='best')
78         # Crea l'oggetto GridSearchCV per Decision Tree Classifier
79         grid_dt = GridSearchCV(dt, parametri_dt, refit=True)
80         return grid_dt
81
82
83     def random_forest():
84         # Definisci i parametri da ottimizzare per Random Forest Classifier
85         parametri_rf = {'n_estimators': [10, 20, 30, 40]}
86         # Crea un classificatore Random Forest Classifier
87         rf = RandomForestClassifier(max_features='sqrt', criterion='entropy', max_depth=None,
88                                     min_samples_split=2, min_samples_leaf=1)
89         # Crea l'oggetto GridSearchCV per Random Forest Classifier
90         grid_rf = GridSearchCV(rf, parametri_rf, refit=True)
91         return grid_rf
92

```

Figura 5.10: 2. Classificatori ottimizzati

6

Capitolo 6

6.1 TECNICHE DI APPRENDIMENTO E METODI DI VALIDAZIONE

La seguente sezione si concentra sull'applicazione di vari metodi di classificazione per prevedere lo stato di stress dei conducenti. Questi metodi sono stati applicati su un dataset multimodale che include diverse caratteristiche rilevanti per la guida, come la frequenza cardiaca, la frequenza respiratoria, la sudorazione perinasale, l'EDA del palmo, la velocità, l'accelerazione, la frenata e lo sterzo.

Sono stati implementati diversi classificatori, tra cui Naive Bayes, la Regressione Logistica, il K-Nearest Neighbors (KNN), il Support Vector Machine (SVM), il Decision Tree e il Random Forest. Ogni classificatore è stato addestrato e testato utilizzando diverse tecniche di validazione incrociata, tra cui la validazione incrociata a 5 fold, la validazione incrociata 'StratifiedShuffleSplit' e la validazione incrociata leave-one-subject-out.

Inoltre, è stata applicata la tecnica di sovra campionamento SMOTE per bilanciare le classi nel set di addestramento ed è stata utilizzata l'Analisi delle Componenti Principali (PCA) per ridurre la dimensionalità dei dati. Infine, sono state calcolate e visualizzate le matrici di confusione per ciascun classificatore e sono state tracciate le curve ROC per confrontare le prestazioni dei diversi classificatori.

Questa sezione fornisce una panoramica dettagliata di come sono stati applicati e valutati questi metodi di classificazione per prevedere lo stato di stress dei conducenti. Attraverso questa analisi, si mira a identificare i metodi di classificazione più efficaci per questo compito specifico.

6.1.1 Definizione delle caratteristiche ed etichette

Nel codice presentato, viene utilizzata la funzione 'pd.read_csv()' per leggere il dataset completo con le features calcolate e filtrate con le tecniche definite nelle sezioni precedenti.

Successivamente vengono definite le caratteristiche e le etichette che saranno utilizzate per la classificazione. Le caratteristiche, indicate con X, includono tutte le features calcolate, ad eccezione delle colonne ‘driver’, ‘Time_Interval’ e ‘stress’ che non contengono dati significativi per la classificazione. Queste caratteristiche rappresentano un insieme complessivo di dati che saranno utilizzati per addestrare i modelli di classificazione.

Vengono inoltre create diverse viste filtrate del set di caratteristiche, ognuna delle quali contiene solo le colonne che corrispondono a un particolare tipo di segnale. Ad esempio, X_HR contiene solo le features estratte per il segnale della frequenza cardiaca (Heart Rate), X_BR per il segnale della frequenza respiratoria (Breathing Rate), e così via. Queste viste filtrate permettono di esaminare l’effetto di ciascun segnale individualmente quando si esegue la classificazione.

Le etichette, indicate con y, sono prese dalla colonna ‘stress’ del dataframe. Queste etichette rappresentano lo stato di stress del conducente, che è l’obiettivo della predizione in questo caso.

Viene creato un array ‘groups’ che codifica i nomi dei driver utilizzando un LabelEncoder. Questo array sarà utilizzato per dividere i dati in gruppi durante la validazione incrociata.

6.1.2 Divisione dei dati in set di addestramento e di test

Nell’ambito dell’apprendimento automatico, la divisione dei dati in set di addestramento e di test è una pratica fondamentale. Questo processo coinvolge la suddivisione del dataset originale in due sottoinsiemi distinti:

- Il **set di addestramento**, utilizzato per addestrare il modello. Durante l’addestramento, il modello cerca di apprendere le relazioni tra le caratteristiche e l’etichetta.
- Il **set di test**, utilizzato per valutare le prestazioni del modello. Questo set non viene utilizzato durante l’addestramento e serve per testare quanto bene il modello ha appreso dalle caratteristiche per prevedere l’etichetta.

La divisione dei dati in set di addestramento e di test è cruciale per prevenire il sovraddattamento, un problema comune nell’apprendimento automatico e riguardante il fatto che il modello si adatta troppo bene ai dati di addestramento e non riesce a generalizzare bene su nuovi dati.

Un metodo comune per eseguire questa divisione è **StratifiedShuffleSplit**, che restituisce fold casuali stratificati, preservando la percentuale di campioni per ciascuna classe. Questo garantisce che la distribuzione delle etichette sia la stessa nei set di addestramento e di test come nel set di dati originale.

Ecco un esempio di come viene utilizzato **StratifiedShuffleSplit** nel codice:

```
from sklearn.model_selection import StratifiedShuffleSplit
sss = StratifiedShuffleSplit(n_splits=5, test_size=0.1, random_state=42)
for train_index, test_index in sss.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

In questo frammento di codice, `StratifiedShuffleSplit` viene inizialmente istanziato per creare un oggetto `sss`. Successivamente, il metodo `split` viene chiamato sull'oggetto `sss` per generare gli indici di addestramento e di test. Il risultato è un set di addestramento e un set di test che mantengono la stessa distribuzione di etichette del set di dati originale.

6.1.3 Metodi di validazione incrociata

Il codice implementa due metodi di validazione incrociata tra cui: la validazione incrociata a 5 fold e la validazione incrociata leave-one-out (LOO). In entrambi i metodi, i dati vengono divisi in set di addestramento e di test, e vengono calcolati i punteggi di validazione incrociata per ciascuno dei classificatori forniti. I punteggi vengono poi salvati in un dizionario per un'analisi successiva.

Ecco una panoramica dei metodi di validazione che sono stati testati all'interno del codice:

- **1. Train-Test Split:** In questa tecnica, i dati vengono suddivisi in un set di addestramento e un set di test. Il modello viene addestrato sul set di addestramento e poi valutato sul set di test. Questa è una tecnica semplice e veloce, ma ha alcuni svantaggi. Ad esempio, i risultati possono variare a seconda di come vengono suddivisi i dati, e il modello potrebbe non essere addestrato o testato su tutti i dati disponibili. Per questo motivo è stata preferita la tecnica dello `StratifiedShuffleSplit` per la suddivisione dei dati.
- **2. StratifiedShuffleSplit:** è una tecnica di validazione incrociata. Questo oggetto di validazione è una fusione di `StratifiedKFold` e `ShuffleSplit`, che restituisce fold casuali stratificati. I fold sono creati preservando la percentuale di campioni per ciascuna classe. Questo garantisce che la distribuzione delle etichette sia la stessa nei set di addestramento e di test come nel set di dati originale
- **3. 5-Fold Cross Validation:** In questa tecnica, i dati vengono suddivisi in 5 parti uguali o “fold”. Il modello viene addestrato su 4 fold e poi valutato sul fold rimanente. Questo processo viene ripetuto 5 volte, ogni volta con un fold diverso usato come set di test. Alla fine, si ottiene una stima dell'accuratezza del modello basata su tutti i dati. Questa tecnica è più robusta rispetto al semplice train-test split perché il modello viene addestrato e testato su tutti i dati. Tuttavia, richiede più tempo perché il modello deve essere addestrato 5 volte.

```

196     """5 fold cross validation"""
197     def five_fold_cross_validation(X, y, groups, classifiers):
198         #cv = KFold(n_splits=5, shuffle=True, random_state=42)
199         cv = GroupKFold(n_splits=5)
200
201         scores_dict = {}
202
203         for train_index, test_index in cv.split(X, y, groups):
204             X_train, X_test = X.iloc[train_index], X.iloc[test_index]
205             y_train, y_test = y.iloc[train_index], y.iloc[test_index]
206
207             print(f"Driver unici per l'addestramento: {np.unique(groups[train_index])}")
208             print(f"Driver unici per il test: {np.unique(groups[test_index])}")
209
210             # Stampa i valori di 'driver' per le righe del set di addestramento e del set di test
211             print("Driver nel set di addestramento:", df['driver'].iloc[train_index].unique())
212             print("Driver nel set di test:", df['driver'].iloc[test_index].unique())
213
214             # Stampa alcuni dei tuoi dati di addestramento e di test
215             print("Dati di addestramento:", X_train.head())
216             print("Dati di test:", X_test.head())
217
218             for name, classifier in classifiers.items():
219                 pipe = Pipeline([('scaler', StandardScaler()), ('clf', classifier)])
220                 scores = cross_val_score(pipe, X_train, y_train, cv=cv, scoring='f1_micro', groups=groups[train_index])
221                 scores_dict[name] = scores
222                 print(f"Punteggi di validazione incrociata per {name}: {scores}")
223                 print(f"Media dei punteggi di validazione incrociata per {name}: {scores.mean()}")
224                 print(f"Deviazione standard dei punteggi di validazione incrociata per {name}: {np.std(scores)}")
225
226         return scores_dict

```

Figura 6.1: Five Fold Cross Validation

- **4. Leave-One-Subject-Out Cross Validation:** Questa tecnica è simile alla 5-fold cross validation, ma invece di dividere i dati in 5 parti uguali, i dati vengono divisi in base al soggetto (in questo contesto, il guidatore). Il modello viene addestrato sui dati di tutti i drivers tranne uno, e poi valutato sui dati del driver escluso. Questo processo viene ripetuto per ogni test driver. Questa tecnica è particolarmente utile quando i dati sono raggruppati per entità, perché assicura che il modello sia in grado di generalizzare a nuovi soggetti. Tuttavia, richiede ancora più tempo rispetto alla 5-fold cross validation perché il modello deve essere addestrato tante volte quanti sono i soggetti.

```

227     '''leave one subject out (LOO)'''
228
229     def leave_one_subject_out_validation(X, y, groups, classifiers):
230         loo = LeaveOneGroupOut()
231
232         scores_dict = {}
233
234         for train_index, test_index in loo.split(X, y, groups):
235             X_train, X_test = X.iloc[train_index], X.iloc[test_index]
236             y_train, y_test = y.iloc[train_index], y.iloc[test_index]
237
238             # Stampa i valori di 'driver' per le righe del set di addestramento e del set di test
239             print("Driver nel set di addestramento:", df['driver'].iloc[train_index].unique())
240             print("Driver nel set di test:", df['driver'].iloc[test_index].unique())
241
242             # Stampa alcuni dei tuoi dati di addestramento e di test
243             print("Dati di addestramento:", X_train.head())
244             print("Dati di test:", X_test.head())
245
246
247
248         for name, classifier in classifiers.items():
249             pipe = Pipeline([('scaler', StandardScaler()), ('clf', classifier)])
250             scores = cross_val_score(pipe, X_train, y_train, cv=loo, scoring='f1_micro', groups=groups[train_index])
251             scores_dict[name] = scores
252             print(f"Punteggi di validazione incrociata per {name}: {scores}")
253             print(f"Media dei punteggi di validazione incrociata per {name}: {scores.mean()}")
254             print(f"Deviazione standard dei punteggi di validazione incrociata per {name}: {np.std(scores)}")
255
256
257     return scores_dict

```

Figura 6.2: Leave One Subject Out

In entrambi i metodi, prima dell’addestramento del modello, i dati vengono standardizzati utilizzando un ‘StandardScaler’.

6.1.4 Standardizzazione delle features

La standardizzazione dei dati è un metodo di riduzione che trasforma una serie di valori in una distribuzione normale standard con media uguale a zero e deviazione standard uguale a uno. Questo processo viene eseguito per ciascuna caratteristica indipendentemente, calcolando le statistiche rilevanti sui campioni nel set di addestramento. La standardizzazione viene eseguita sottraendo la media (μ) da ogni valore (x) del vettore e dividendo la differenza per la devianza standard (σ).

All’interno del codice ‘StandardScaler’ viene utilizzato per standardizzare i dati prima di addestrare i classificatori. Questo è importante perché molti algoritmi di apprendimento automatico, come SVM e KNN, non funzionano bene se le caratteristiche non sono su una scala simile. Inoltre, la standardizzazione dei dati può migliorare le prestazioni del processo di apprendimento.

6.1.5 Altri metodi utilizzati

- Nel codice presentato, viene definita una funzione ‘`leave_one_subject_out_single_driver_test()`’, che implementa un metodo di validazione incrociata leave-one-out (LOO) per un singolo driver di test. Questa funzione prende come input le caratteristiche (X), le etichette (y), i gruppi, i classificatori e il driver di test. Utilizza la funzione ‘`LeaveOneGroupOut()`’ per creare un oggetto LOO e divide i dati in set di addestramento e di test in base al driver di test specificato. Successivamente, per ciascun classificatore, viene creato un pipeline che

standardizza i dati e addestra il classificatore. I punteggi di validazione incrociata vengono calcolati e salvati in un dizionario.

- La funzione ‘`create_signal_boxplot()`’ crea dei boxplot per visualizzare i punteggi di validazione incrociata ottenuti con la validazione incrociata a 5 fold o con la leave-one-subject-out per diversi segnali e classificatori. Questa funzione prende come input i gruppi e per ciascun segnale, calcola i punteggi di validazione incrociata, ne calcola la media e salva i risultati in un dizionario. Questi risultati vengono poi convertiti in un dataframe e visualizzati come boxplot utilizzando la libreria seaborn. Viene creato un boxplot per ogni segnale e un boxplot per ogni classificatore. Questi grafici forniscono una rappresentazione visiva della distribuzione dei punteggi di validazione incrociata per ciascun segnale e classificatore, facilitando la comparazione dei risultati.

```

330     """creazione boxplot per loo e 5 fold cross validation"""
331     def create_signal_boxplot(groups):
332         results = {}
333         for signal_name, X_resampled in [('ALL', X), ('HR', X_HR), ('BR', X_BR), ('PP', X_PP),
334                                         ('Palm_EDA', X_Palm_EDA), ('SPEED', X_SPEED), ('ACC', X_ACC),
335                                         ('BRA', X_BRA), ('STEE', X_STEE)]:
336             scores = leave_one_subject_out_validation(X_resampled, y, groups, classifiers)
337             mean_scores = {name: np.mean(scores[name]) for name in classifiers.keys() if name in scores}
338             results[signal_name] = mean_scores
339             df_results = pd.DataFrame(results)
340
341             # Crea il boxplot
342             plt.figure(figsize=(10, 5))
343             sns.boxplot(data=df_results)
344             plt.title('Leave one out')
345             plt.ylabel('Punteggio loo')
346             plt.xlabel('Segnale')
347             plt.show()
348
349             # Crea il boxplot
350             plt.figure(figsize=(10, 5))
351             sns.boxplot(data=df_results.T)
352             plt.title('Leave one out')
353             plt.ylabel('Punteggio loo')
354             plt.xlabel('Classificatore')
355             plt.show()

```

Figura 6.3: Create Signal Boxplot

- Nel codice presentato viene definita una funzione ‘`matrice_di_confusione()`’, che implementa la divisione del set di dati utilizzando ‘StratifiedShuffleSplit’ e calcola la matrice di confusione per ciascun classificatore.

La funzione ‘`StratifiedShuffleSplit`’ è utilizzata per dividere il set di dati in set di addestramento e di test. Questa funzione mantiene la stessa proporzione di etichette nel set di addestramento e nel set di test come nel set di dati originale. Il numero di suddivisioni è impostato a 5 e la dimensione del set di test è impostata al 10% del set di dati originale. Per ciascuna suddivisione vengono stampati i valori unici del driver nel set di addestramento e nel set di test. Successivamente, per ciascun classificatore, viene creata una pipeline che standardizza i dati e addestra il classificatore. Il classificatore addestrato viene poi utilizzato per prevedere le etichette del set di test.

La matrice di confusione viene calcolata utilizzando le etichette previste e le etichette reali del set di test. Viene creato un grafico per ciascun classificatore che mostra la matrice di confusione e include il titolo del classificatore, il rapporto di classificazione e l'accuratezza del modello.

6.1.6 Matrice di confusione:

Una matrice di confusione è uno strumento utilizzato per analizzare le prestazioni di un modello di classificazione. Questa matrice confronta i valori effettivi con i valori previsti dal modello.

Ogni riga della matrice rappresenta le classi effettive, mentre ogni colonna rappresenta le classi previste. L'elemento sulla riga i e sulla colonna j è il numero di casi in cui il modello ha classificato la classe "vera" i come classe j . In questo modo, la matrice mette in evidenza dove il modello commette errori.

La matrice di confusione può essere utilizzata per calcolare diverse metriche, tra cui:

- True positive (TP): se la classe prevista è positiva ed è uguale alla classe effettiva.
- True negative (TN): se la classe prevista è negativa ed è uguale alla classe effettiva.
- False positive (FP): se la classe prevista è positiva ma è diversa dalla classe effettiva.
- False negative (FN): se la classe prevista è negativa ma è diversa dalla classe effettiva.

Queste metriche possono essere utilizzate per calcolare l'accuracy, la precisione, il recall e il punteggio F1 del modello.

- Precisione: La precisione è il rapporto tra le istanze correttamente classificate positivamente e il totale delle istanze classificate positivamente. Ad esempio, una precisione del 0.83 per la classe 0 nel K-NN significa che l'83% delle istanze che il modello ha classificato come classe 0 erano effettivamente della classe 0.
- Recall: Il recall è il rapporto tra le istanze correttamente classificate positivamente e il totale delle istanze effettivamente positive. Ad esempio, un recall del 0.90 per la classe 0 nel K-NN significa che il modello ha identificato correttamente il 90% delle istanze totali della classe 0.
- F1-Score: L'F1-Score è una media armonica tra precisione e recall. Un F1-Score più alto indica un modello migliore. Ad esempio, un F1-Score di 0.86 per la classe 0 nel K-NN indica che il modello ha una buona precisione e recall per questa classe.
- L'accuracy è la frazione delle predizioni che il modello ha ottenuto correttamente. Ad esempio, un'accuracy del 0.74 nel K-NN significa che il modello ha predetto correttamente il 74% delle istanze. Le medie macro avg e weighted avg calcolano la media delle metriche per tutte le classi. La media macro calcola la media non ponderata delle metriche per ciascuna classe, mentre la media ponderata tiene conto del numero di istanze in ciascuna classe.

Ecco un esempio di una matrice di confusione per un problema di classificazione binaria, rappresentante rispettivamente i veri positivi, i falsi positivi, i falsi negativi e i veri negativi:

		Actual Values	
		Yes	No
Predicted Values	Yes	True Positive	False Positive
	No	False Negative	True Negative

Figura 6.4: Confusion Matrix

6.1.7 Synthetic Minority Over-sampling Technique (SMOTE)

Nel codice fornito è stata implementata anche una funzione chiamata ‘matrice_di_confusione_smote()’. Questa funzione utilizza la tecnica di sovraccampionamento SMOTE (Synthetic Minority Over-sampling Technique) per bilanciare le classi nel set di addestramento. SMOTE è una tecnica di sovraccampionamento che genera nuovi esempi sintetici per la classe minoritaria. Questa tecnica è particolarmente utile quando si lavora con set di dati sbilanciati, in cui una classe ha molti meno esempi rispetto alle altre.

In questa funzione, i dati vengono prima divisi in set di addestramento e di test utilizzando ‘StratifiedShuffleSplit’. Successivamente, SMOTE viene applicato solo al set di addestramento. Questo garantisce che i nuovi esempi sintetici generati da SMOTE non siano presenti nel set di test, evitando così il rischio di sovraddattamento.

Dopo l’applicazione di SMOTE, per ciascun classificatore, viene creata una pipeline che standardizza i dati e addestra il classificatore sui dati di addestramento sovraccampionati. Il classificatore addestrato viene poi utilizzato per predirre le etichette del set di test.

7

Capitolo 7

7.1 RISULTATI E DISCUSSIONI

In questa sezione saranno mostrati i risultati ottenuti grazie alle diverse metriche di validazione e classificatori utilizzati.

7.2 Risultati classificazione binaria:

7.2.1 StratifiedShuffleSplit con matrice di confusione a due etichette, classificatori ottimizzati:

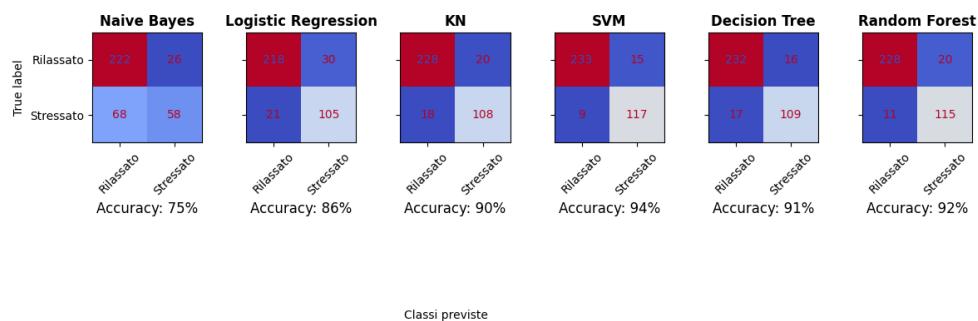


Figura 7.1: 1. Confusion Matrix

	Precisione	Recall	F1-score
Classe 0	0.77	0.90	0.83
Classe 1	0.69	0.46	0.55
Accuracy			0.75
Macro avg	0.73	0.68	0.69
Weighted avg	0.74	0.75	0.73

Tabella 7.1: Rapporto del classificatore Naive Bayes

	Precisione	Recall	F1-score
Classe 0	0.91	0.88	0.90
Classe 1	0.78	0.83	0.80
Accuracy			0.86
Macro avg	0.84	0.86	0.85
Weighted avg	0.87	0.86	0.86

Tabella 7.2: Rapporto del classificatore Logistic Regression

	Precisione	Recall	F1-score
Classe 0	0.93	0.92	0.92
Classe 1	0.84	0.86	0.85
Accuracy			0.90
Macro avg	0.89	0.89	0.89
Weighted avg	0.90	0.90	0.90

Tabella 7.3: Rapporto del classificatore K-Nearest Neighbors (KN)

	Precisione	Recall	F1-score
Classe 0	0.96	0.94	0.95
Classe 1	0.89	0.93	0.91
Accuracy			0.94
Macro avg	0.92	0.93	0.93
Weighted avg	0.94	0.94	0.94

Tabella 7.4: Rapporto del classificatore Support Vector Machine (SVM)

	Precisione	Recall	F1-score
Classe 0	0.93	0.94	0.93
Classe 1	0.87	0.87	0.87
Accuracy			0.91
Macro avg	0.90	0.90	0.90
Weighted avg	0.91	0.91	0.91

Tabella 7.5: Rapporto del classificatore Decision Tree

	Precisione	Recall	F1-score
Classe 0	0.95	0.92	0.94
Classe 1	0.85	0.91	0.88
Accuracy			0.92
Macro avg	0.90	0.92	0.91
Weighted avg	0.92	0.92	0.92

Tabella 7.6: Rapporto del classificatore Random Forest

La Figura 7.1 mostra le matrici di confusione per diversi classificatori utilizzati per prevedere lo stato di stress dei conducenti. Ogni matrice è accompagnata dalla percentuale di accuratezza del modello corrispondente, che rappresenta la proporzione delle previsioni corrette sul totale delle predizioni effettuate. Nelle matrici, le righe rappresentano le etichette vere ("relaxed" e "stressed"), mentre le colonne rappresentano le classi previste dai modelli.

Dal grafico si può osservare quanto segue:

- Il classificatore Naive Bayes ha una precisione dell'75%. Si può vedere che questo classificatore ha una tendenza a classificare erroneamente i casi di "non stress" come

“stressati” (68 FP), rispetto al numero di “stress” erroneamente classificati come “non stressati” (26 FN). Questo potrebbe indicare che il modello è più sensibile nel rilevare lo “stress”, ma potrebbe portare a un numero maggiore di falsi allarmi.

- La Regressione Logistica mostra una migliore prestazione con un’accuratezza dell’86%. Ha meno FN e falsi positivi (FP) rispetto al Naive Bayes, suggerendo un equilibrio migliore nella capacità di prevedere correttamente entrambi gli stati di stress e non stress.
- Il classificatore K-Nearest Neighbors (KNN) presenta un’accuratezza dell’90%. Mostra una distribuzione equilibrata delle previsioni tra gli stati di stress e non stress, con un numero simile di FN e FP.
- Il Support Vector Machine (SVM) ha l’accuratezza più alta tra tutti i classificatori, con il 94%. Ha il minor numero di FN e FP, indicando una maggiore precisione nella predizione degli stati di stress e non stress dei conducenti.
- L’albero decisionale presenta un’accuratezza dell’ 89%, con una maggioranza di FN rispetto ai VP. Questo indica che tende a predire erroneamente lo stato di non stress più spesso dello stato di stress.
- Il Random Forest ha un’accuratezza del 92%. Sembra essere in grado di bilanciare meglio la previsione degli stati di stress e non stress rispetto all’albero decisionale singolo.

In generale, l’SVM sembra essere il classificatore più efficace per prevedere lo stato di stress dei conducenti in questa tipologia di classificazione, avendo l’accuratezza più alta tra tutti i metodi testati. Tuttavia, va notato che ogni classificatore ha i suoi punti di forza e di debolezza e potrebbe essere più adatto a determinati scenari o tipi di dati. È importante considerare anche altri fattori come la complessità del modello, il tempo di addestramento e il numero di features utilizzate quando si sceglie un classificatore per una specifica applicazione.

In conclusione, la scelta del miglior classificatore dipende da vari fattori come l’accuratezza, l’equilibrio tra FN e FP, la complessità del modello e le esigenze specifiche dell’applicazione.

7.2.2 StratifiedShuffleSplit (SMOTE) con matrice di confusione a due etichette, classificatori ottimizzati:

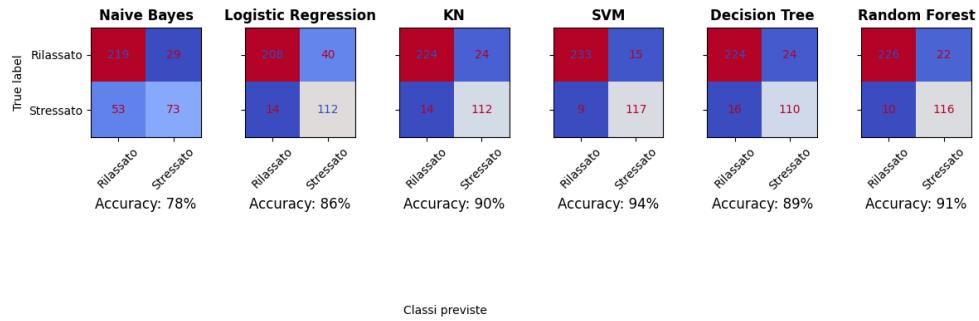


Figura 7.2: 2. Confusion Matrix

	Precisione	Recall	F1-score
Classe 0	0.81	0.88	0.84
Classe 1	0.72	0.58	0.64
Accuracy			0.78
Macro avg	0.76	0.73	0.74
Weighted avg	0.78	0.78	0.77

Tabella 7.7: Rapporto del classificatore Naive Bayes

	Precisione	Recall	F1-score
Classe 0	0.94	0.84	0.89
Classe 1	0.74	0.89	0.81
Accuracy			0.86
Macro avg	0.84	0.86	0.85
Weighted avg	0.87	0.86	0.86

Tabella 7.8: Rapporto del classificatore Logistic Regression

	Precisione	Recall	F1-score
Classe 0	0.94	0.90	0.92
Classe 1	0.82	0.89	0.85
Accuracy			0.90
Macro avg	0.88	0.90	0.89
Weighted avg	0.90	0.90	0.90

Tabella 7.9: Rapporto del classificatore K-Nearest Neighbors (KN)

	Precisione	Recall	F1-score
Classe 0	0.96	0.94	0.95
Classe 1	0.89	0.93	0.91
Accuracy			0.94
Macro avg	0.92	0.93	0.93
Weighted avg	0.94	0.94	0.94

Tabella 7.10: Rapporto del classificatore Support Vector Machine (SVM)

	Precisione	Recall	F1-score
Classe 0	0.93	0.90	0.92
Classe 1	0.82	0.87	0.85
Accuracy			0.89
Macro avg	0.88	0.89	0.88
Weighted avg	0.90	0.89	0.89

Tabella 7.11: Rapporto del classificatore Decision Tree

	Precisione	Recall	F1-score
Classe 0	0.96	0.91	0.93
Classe 1	0.84	0.92	0.88
Accuracy			0.91
Macro avg	0.90	0.92	0.91
Weighted avg	0.92	0.91	0.92

Tabella 7.12: Rapporto del classificatore Random Forest

Dai risultati ottenuti, si osservano alcune differenze rispetto ai risultati precedenti in cui non era stata utilizzata la tecnica SMOTE.

- Naive Bayes: Questo algoritmo ha mostrato una precisione dell'81% per la classe 0 (non stress) e del 72% per la classe 1 (stress). Il recall per la classe 1 è aumentato al 58%, il che indica un miglioramento nel rilevare lo stress rispetto all'analisi precedente senza SMOTE. Tuttavia, l'accuracy complessiva è stata del 78%, il che indica che c'è spazio per miglioramenti.
- Logistic Regression: Questo algoritmo ha mostrato una precisione dell'94% per la classe 0 e del 74% per la classe 1. Il recall per la classe 1 è aumentato notevolmente

all'89%, il che indica un significativo miglioramento nel rilevare lo stress rispetto all'analisi precedente senza SMOTE. L'accuracy complessiva è aumentata all'86%.

- K-Nearest Neighbors (KN): Questo algoritmo ha mostrato una precisione dell'94% per la classe 0 e dell'82% per la classe 1. Il recall per la classe 1 è aumentato all'89%. L'accuracy complessiva è rimasta stabile al 90%.
- Support Vector Machine (SVM): Questo algoritmo ha mostrato una precisione dell'96% per la classe 0 e dell'89% per la classe 1. Il recall per la classe 1 è rimasto stabile al 93%. L'accuracy complessiva è rimasta stabile al 94%.
- Decision Tree: Questo algoritmo ha mostrato una precisione dell'93% per la classe 0 e dell'82% per la classe 1. Il recall per la classe 1 è sceso all'82%. L'accuracy complessiva è stata del 89%.
- Random Forest: Questo algoritmo ha mostrato una precisione dell'96% per la classe 0 e dell'84% per la classe 1. Il recall per la classe 1 è aumentato al 92%, il che indica un miglioramento nel rilevare lo stress rispetto all'analisi precedente senza SMOTE. L'accuracy complessiva è stata del 91%.

L'applicazione della tecnica SMOTE sembra aver migliorato la capacità di alcuni algoritmi nel rilevare lo stress, come indicato dai punteggi di recall più alti per la classe 1. In particolare, Logistic Regression e KNN hanno mostrato i miglioramenti più significativi, nonostante l'accuracy di alcuni classificatori sembra essere diminuita rispetto all'analisi precedente. Tuttavia, è importante notare che l'accuratezza non è l'unico indicatore delle prestazioni di un modello di classificazione, specialmente quando si lavora con set di dati sbilanciati.

In questo caso è stata applicata la tecnica SMOTE per bilanciare le classi. Ciò può portare a una diminuzione dell'accuracy, ma può anche migliorare altre metriche importanti come il recall, specialmente per la classe minoritaria (in questo caso, la classe 1 - stress). Come si può osservare il recall per la classe 1 è aumentato per la maggior parte dei classificatori rispetto all'analisi precedente. Questo indica che questi classificatori sono ora più efficaci nel rilevare lo stress (classe 1), che potrebbe essere un risultato desiderabile per analisi future.

Inoltre, l'uso di SMOTE può aiutare a prevenire l'overfitting sui dati di training, migliorando la capacità del modello di generalizzare su nuovi dati.

StratifiedShuffleSplit con matrice di confusione a due etichette, classificatori base:

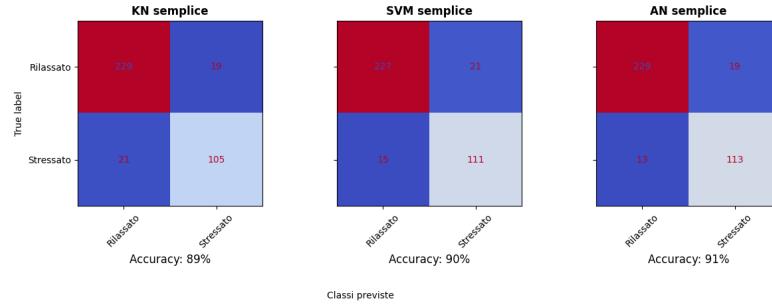


Figura 7.3: 3. Confusion Matrix

	Precisione	Recall	F1-score
Classe 0	0.92	0.92	0.92
Classe 1	0.85	0.83	0.84
Accuracy			0.89
Macro avg	0.88	0.88	0.88
Weighted avg	0.89	0.89	0.89

Tabella 7.13: Rapporto del classificatore K-Nearest Neighbors (KN) semplice

	Precisione	Recall	F1-score
Classe 0	0.94	0.92	0.93
Classe 1	0.84	0.88	0.86
Accuracy			0.90
Macro avg	0.89	0.90	0.89
Weighted avg	0.91	0.90	0.90

Tabella 7.14: Rapporto del classificatore Support Vector Machine (SVM) semplice

	Precisione	Recall	F1-score
Classe 0	0.95	0.92	0.93
Classe 1	0.86	0.90	0.88
Accuracy			0.91
Macro avg	0.90	0.91	0.91
Weighted avg	0.92	0.91	0.91

Tabella 7.15: Rapporto del classificatore Artificial Neural Network (AN) semplice

- K-Nearest Neighbors (KNN): ha mostrato una precisione dell'92% per la classe 0 (non stress) e dell'85% per la classe 1 (stress). Il recall per entrambe le classi è stato relativamente alto (92% per la classe 0 e 83% per la classe 1), il che indica che l'algoritmo è stato efficace nel rilevare sia lo stress che il non stress. Tuttavia, l'accuracy complessiva è stata del 89%, il che indica che c'è spazio per miglioramenti.
- Support Vector Machine (SVM): ha mostrato una precisione dell'94% per la classe 0 e dell'84% per la classe 1. Il recall per la classe 1 è aumentato all'88%, il che indica che l'algoritmo è stato più efficace nel rilevare lo stress rispetto al KN semplice. L'accuracy complessiva è stata del 90%, il che indica un miglioramento rispetto al KNN.
- Artificial Neural Network (ANN): ha mostrato la precisione e il recall più alti tra tutti gli algoritmi testati in questa analisi. In particolare, il recall per la classe 1 è aumentato al 90%, il che indica che l'algoritmo è stato molto efficace nel rilevare lo stress. L'accuracy complessiva è stata del 91%, il che indica un ulteriore miglioramento rispetto a SVM e KNN.

Le reti neurali artificiali, come il modello AN semplice, sono in grado di apprendere relazioni non lineari tra le variabili. Se la relazione tra le caratteristiche del set di dati e lo stato di stress del conducente non è lineare, allora questo potrebbe spiegare perché il modello AN semplice ha superato gli altri classificatori. Inoltre, siccome le reti neurali possono catturare interazioni complesse tra le caratteristiche, questo potrebbe fornire un ulteriore vantaggio al modello in questione.

Un altro aspetto da considerare è l'adattabilità delle reti neurali. Questi modelli sono altamente adattabili e possono apprendere da una vasta gamma di forme di dati.

StratifiedShuffleSplit (SMOTE) con matrice di confusione a due etichette, classificatori base:

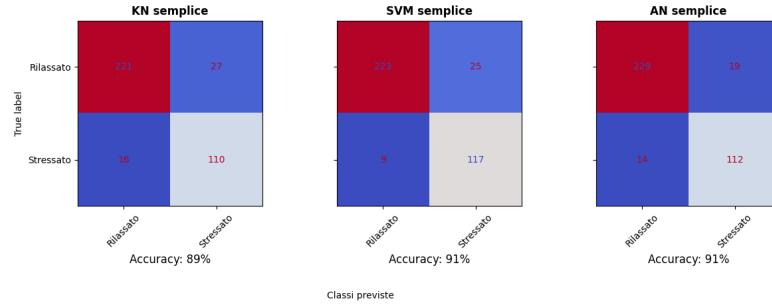


Figura 7.4: 4. Confusion Matrix

	Precisione	Recall	F1-score
Classe 0	0.93	0.89	0.91
Classe 1	0.80	0.87	0.84
Accuracy			0.89
Macro avg	0.87	0.88	0.88
Weighted avg	0.89	0.89	0.89

Tabella 7.16: Rapporto del classificatore K-Nearest Neighbors (KN) semplice

	Precisione	Recall	F1-score
Classe 0	0.96	0.90	0.93
Classe 1	0.82	0.93	0.87
Accuracy			0.91
Macro avg	0.89	0.91	0.90
Weighted avg	0.91	0.91	0.91

Tabella 7.17: Rapporto del classificatore Support Vector Machine (SVM) semplice

	Precisione	Recall	F1-score
Classe 0	0.94	0.92	0.93
Classe 1	0.85	0.89	0.87
Accuracy			0.91
Macro avg	0.90	0.91	0.90
Weighted avg	0.91	0.91	0.91

Tabella 7.18: Rapporto del classificatore Artificial Neural Network (AN) semplice

Dai risultati ottenuti utilizzando la tecnica SMOTE, si osservano alcune differenze rispetto ai risultati precedenti.

- K-Nearest Neighbors (KNN): Questo algoritmo ha mostrato una precisione dell'93% per la classe 0 (non stress) e dell'80% per la classe 1 (stress). Il recall per la classe 1 è aumentato all'87%, il che indica un miglioramento nel rilevare lo stress rispetto all'analisi precedente senza SMOTE. L'accuracy complessiva è rimasta stabile al 89%.
- Support Vector Machine (SVM) semplice: Questo algoritmo ha mostrato una precisione dell'96% per la classe 0 e dell'82% per la classe 1. Il recall per la classe 1 è aumentato notevolmente al 93%, il che indica un significativo miglioramento nel rilevare lo stress rispetto all'analisi precedente senza SMOTE. L'accuracy complessiva è aumentata al 91%.
- Artificial Neural Network (ANN): Questo algoritmo ha mostrato una precisione dell'94% per la classe 0 e dell'85% per la classe 1. Il recall per la classe 1 è rimasto stabile all'89%. L'accuracy complessiva è rimasta stabile al 91%.

In conclusione, l'applicazione della tecnica SMOTE sembra aver migliorato la capacità degli algoritmi di rilevare lo stress, come indicato dai punteggi di recall più alti per la classe 1.

7.2.3 Principal Component Analysis (PCA)

L'Analisi delle Componenti Principali (PCA, dall'inglese Principal Component Analysis) è una tecnica di riduzione dell'estensione, particolarmente utile quando si lavora con set di dati di grande entità, dove la PCA può aiutare a ridurne la dimensionalità mantenendo la maggior parte delle informazioni rilevanti.

La PCA funziona selezionando esempi che sono vicini nello spazio delle caratteristiche, tracciando una linea tra gli esempi in quello spazio e generando nuovi esempi lungo quella linea. In questo modo, la PCA è in grado di bilanciare la distribuzione di classe senza perdere informazioni importanti.

Nel corso della ricerca, è stata eseguita un'analisi delle componenti principali (PCA) sui dati di input. Inizialmente, è stato visualizzato un grafico della varianza cumulativa spiegata in funzione del numero di componenti. Questo ha aiutato a determinare il numero ottimale di componenti da mantenere per spiegare una certa percentuale della varianza totale nei dati. Nel grafico viene tracciata una linea rossa per indicare la soglia della

varianza spiegata (in questo progetto si è scelto di fare due esperimenti con la varianza al 99% e poi al 95%).

Successivamente, la PCA è stata eseguita sui dati, riducendo la loro dimensionalità a un numero specificato di componenti.

Infine, i dati sono stati standardizzati ed è stata visualizzata la matrice di confusione.

In sintesi, queste funzioni permettono di eseguire un'analisi PCA sui dati, visualizzare i risultati e determinare il numero ottimale di componenti da mantenere per spiegare una certa percentuale della varianza nei dati.

Inizialmente, nel codice che genera la matrice di confusione, i dati vengono standardizzati utilizzando ‘StandardScaler()’. Successivamente, viene applicata la PCA ai dati standardizzati per ridurre la dimensionalità dei dati in base al numero di componenti calcolati.

I dati trasformati dalla PCA vengono poi divisi in set di addestramento e di test utilizzando ‘StratifiedShuffleSplit’. Questa funzione mantiene la stessa proporzione di etichette nel set di addestramento e nel set di test come nel set di dati originale.

Per ciascun classificatore, viene creato un pipeline che standardizza i dati e addestra il classificatore sui dati di addestramento. Il classificatore addestrato viene poi utilizzato per prevedere le etichette del set di test.

La matrice di confusione viene calcolata utilizzando le etichette previste e le etichette reali del set di test. La matrice di confusione viene poi visualizzata in un grafico, permettendo così di confrontare le prestazioni dei diversi classificatori. Ogni grafico mostra la matrice di confusione e include il titolo del classificatore, il rapporto di classificazione e l'accuratezza del modello.

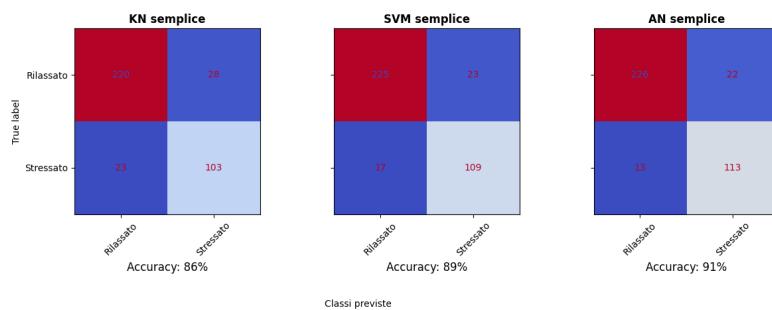


Figura 7.5: PCA 95% di varianza

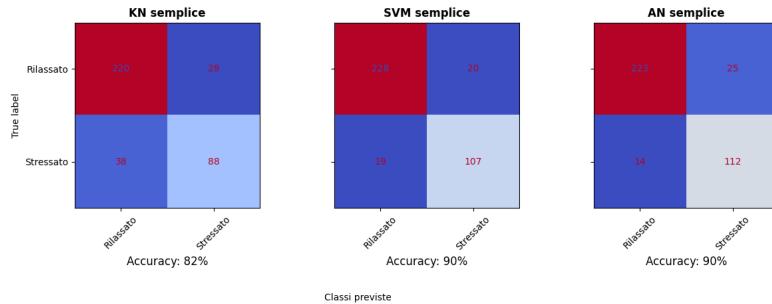


Figura 7.6: PCA 99% di varianza

7.2.4 Receiver Operating Characteristic (ROC)

La curva ROC (Receiver Operating Characteristic) è un grafico che mostra le prestazioni di un modello di classificazione a tutte le soglie di classificazione possibili¹. Questa curva traccia due parametri¹: - Il tasso di veri positivi (TPR, True Positive Rate), noto anche come sensibilità o recall¹. - Il tasso di falsi positivi (FPR, False Positive Rate), noto anche come fall-out¹.

La curva ROC viene creata tracciando il valore del TPR rispetto al FPR a varie impostazioni di soglia¹. Abbassando la soglia di classificazione si classificano più elementi come positivi, aumentando sia i falsi positivi che i veri positivi¹.

L'AUC (Area Under the ROC Curve) misura l'intera area bidimensionale sotto l'intera curva ROC (da (0,0) a (1,1)). L'AUC fornisce una misura aggregata delle prestazioni attraverso tutte le possibili soglie di classificazione. Un modo di interpretare l'AUC è come la probabilità che il modello classifichi un esempio positivo casuale più in alto di un esempio negativo casuale.

Per disegnare la curva è stata implementata una funzione, che calcola e visualizza la curva ROC per diversi classificatori. Per ciascun classificatore, viene creato un pipeline che standardizza i dati e addestra il classificatore sui dati di addestramento. Il classificatore addestrato viene poi utilizzato per prevedere le probabilità delle etichette del set di test. La curva ROC viene calcolata utilizzando le probabilità previste e le etichette reali del set di test; infine, la viene visualizzata in un grafico, che permette di confrontare le prestazioni dei diversi classificatori.

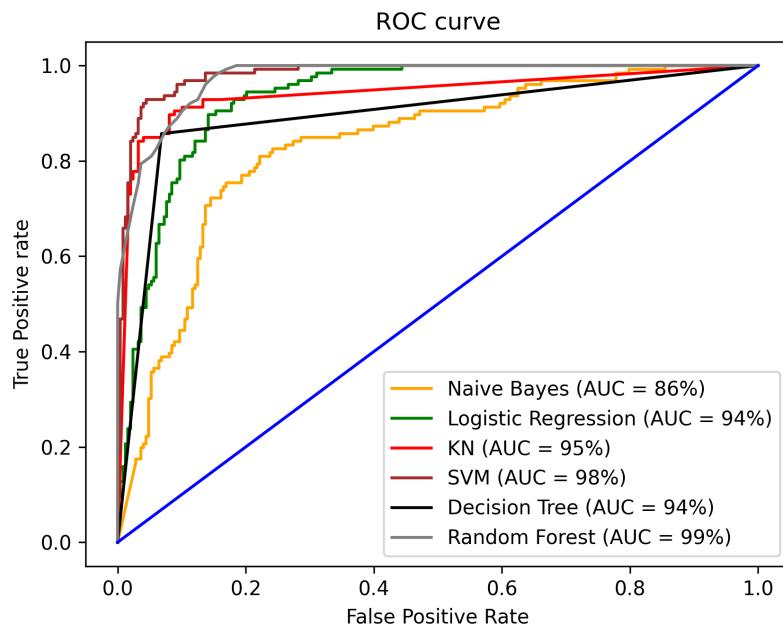


Figura 7.7: ROC classificatori ottimizzati

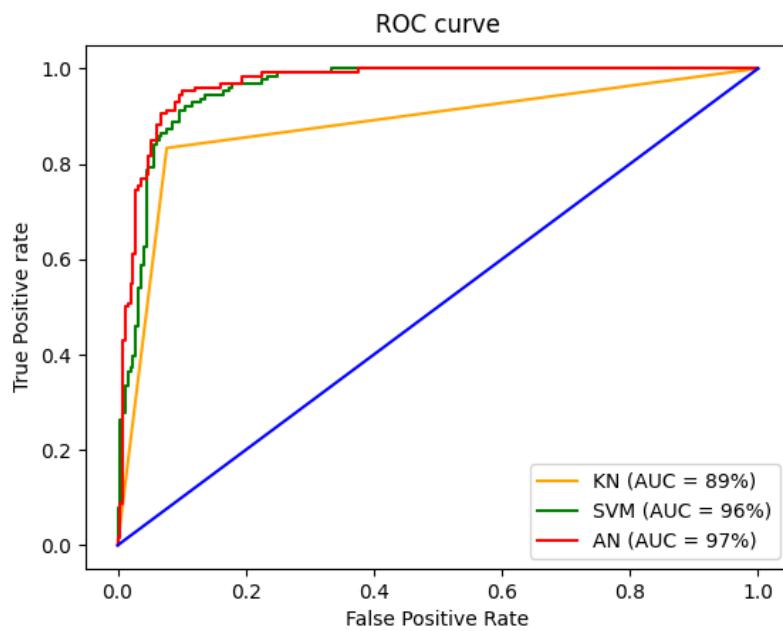


Figura 7.8: ROC classificatori base

7.3 Risultati classificazione multclasse:

7.3.1 StratifiedShuffleSplit con matrice di confusione a quattro etichette, classificatori ottimizzati:

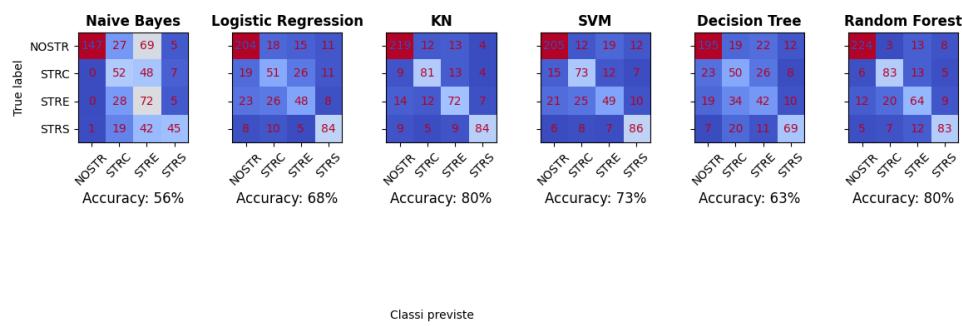


Figura 7.9: 5. Confusion Matrix

	Precisione	Recall	F1-score
Non stress	0.99	0.59	0.74
Stress cognitivo	0.41	0.49	0.45
Stress emotivo	0.31	0.69	0.43
Stress sensorimotorio	0.73	0.42	0.53
Accuracy			0.56
Macro avg	0.61	0.55	0.54
Weighted avg	0.71	0.56	0.59

Tabella 7.19: Rapporto del classificatore Naive Bayes

	Precisione	Recall	F1-score
Non stress	0.80	0.82	0.81
Stress cognitivo	0.49	0.48	0.48
Stress emotivo	0.51	0.46	0.48
Stress sensorimotorio	0.74	0.79	0.76
Accuracy			0.68
Macro avg	0.63	0.64	0.63
Weighted avg	0.68	0.68	0.68

Tabella 7.20: Rapporto del classificatore Logistic Regression

	Precisione	Recall	F1-score
Non stress	0.87	0.88	0.88
Stress cognitivo	0.74	0.76	0.75
Stress emotivo	0.67	0.69	0.68
Stress sensorimotorio	0.85	0.79	0.82
Accuracy			0.80
Macro avg	0.78	0.78	0.78
Weighted avg	0.81	0.80	0.80

Tabella 7.21: Rapporto del classificatore K-Nearest Neighbors (KN)

	Precisione	Recall	F1-score
Non stress	0.96	0.90	0.93
Stress cognitivo	0.82	0.93	0.87
Stress emotivo	0.56	0.47	0.51
Stress sensorimotorio	0.75	0.80	0.77
Accuracy			0.73
Macro avg	0.89	0.91	0.90
Weighted avg	0.91	0.91	0.91

Tabella 7.22: Rapporto del classificatore Support Vector Machine (SVM)

	Precisione	Recall	F1-score
Non stress	0.80	0.79	0.79
Stress cognitivo	0.41	0.47	0.43
Stress emotivo	0.42	0.40	0.41
Stress sensorimotorio	0.70	0.64	0.67
Accuracy			0.63
Macro avg	0.58	0.57	0.58
Weighted avg	0.63	0.63	0.63

Tabella 7.23: Rapporto del classificatore Decision Tree

	Precisione	Recall	F1-score
Non stress	0.91	0.90	0.91
Stress cognitivo	0.73	0.78	0.75
Stress emotivo	0.63	0.61	0.62
Stress sensorimotorio	0.79	0.78	0.78
Accuracy			0.80
Macro avg	0.90	0.92	0.91
Weighted avg	0.92	0.91	0.92

Tabella 7.24: Rapporto del classificatore Random Forest

Nell’analisi presentata, si è passati da una classificazione binaria, utilizzando la colonna ‘stress’, a una classificazione multiclasse, utilizzando la colonna ‘Stimulus’ del dataframe

originale. Questo cambiamento ha introdotto una maggiore complessità nel compito di classificazione, poiché ora ci sono quattro possibili etichette: normal drive (no stress), cognitive drive, emotional drive e sensorimotor drive.

- Naive Bayes: Questo algoritmo ha mostrato una precisione molto alta per la classe 0 (Non stress), ma una precisione relativamente bassa per le altre classi. Il recall per la classe 3 (Stress sensorimotorio) è stato sorprendentemente alto, il che indica che l'algoritmo è stato efficace nel rilevare questo tipo di stress. Tuttavia, l'accuracy complessiva è stata del 56%, il che indica che c'è spazio per miglioramenti.
- Logistic Regression: Questo algoritmo ha mostrato una precisione e un recall relativamente equilibrati per tutte le classi, con la classe 0 (Non stress) che ha ottenuto i punteggi più alti. L'accuracy complessiva è stata del 68%.
- K-Nearest Neighbors (KN): Questo algoritmo ha mostrato una precisione e un recall relativamente alti per tutte le classi, con la classe 0 (Non stress) che ha ottenuto i punteggi più alti. L'accuracy complessiva è stata del 80%, il che indica un miglioramento rispetto a Naive Bayes e Logistic Regression.
- Support Vector Machine (SVM): Questo algoritmo ha mostrato una precisione e un recall relativamente alti per tutte le classi, con la classe 0 (Non stress) che ha ottenuto i punteggi più alti. L'accuracy complessiva è stata del 73%.
- Decision Tree: Questo algoritmo ha mostrato una precisione e un recall relativamente bassi per tutte le classi, con la classe 0 (Non stress) che ha ottenuto i punteggi più alti. L'accuracy complessiva è stata del 63%, il che indica che c'è spazio per miglioramenti.
- Random Forest: Questo algoritmo ha mostrato una precisione e un recall relativamente alti per tutte le classi, con la classe 0 (Non stress) che ha ottenuto i punteggi più alti. L'accuracy complessiva è stata del 80%, il che indica un miglioramento rispetto a Decision Tree.

Dai risultati ottenuti, si osserva che l'accuratezza dei classificatori è generalmente diminuita quando si è passati alla classificazione multclasse. Questo è dovuto al fatto che distinguere tra quattro classi è intrinsecamente più difficile che distinguere tra due classi. In particolare, potrebbe essere difficile per i classificatori distinguere tra i diversi tipi di stress (cognitive drive, emotional drive e sensorimotor drive), poiché questi stati potrebbero presentare caratteristiche simili o sovrapposte.

7.3.2 StratifiedShuffleSplit (SMOTE) con matrice di confusione a quattro etichette, classificatori ottimizzati:

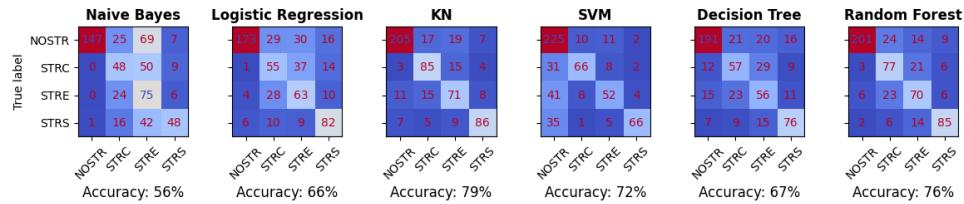


Figura 7.10: 6. Confusion Matrix

	Precisione	Recall	F1-score
Non stress	0.99	0.59	0.74
Stress cognitivo	0.42	0.45	0.44
Stress emotivo	0.32	0.71	0.44
Stress sensorimotorio	0.69	0.45	0.54
Accuracy			0.56
Macro avg	0.61	0.55	0.54
Weighted avg	0.70	0.56	0.59

Tabella 7.25: Rapporto del classificatore Naive Bayes

	Precisione	Recall	F1-score
Non stress	0.94	0.70	0.80
Stress cognitivo	0.45	0.51	0.48
Stress emotivo	0.45	0.60	0.52
Stress sensorimotorio	0.67	0.77	0.72
Accuracy			0.66
Macro avg	0.63	0.64	0.63
Weighted avg	0.71	0.66	0.67

Tabella 7.26: Rapporto del classificatore Logistic Regression

	Precisione	Recall	F1-score
Non stress	0.91	0.83	0.86
Stress cognitivo	0.70	0.79	0.74
Stress emotivo	0.62	0.68	0.65
Stress sensorimotorio	0.82	0.80	0.81
Accuracy			0.79
Macro avg	0.76	0.78	0.77
Weighted avg	0.80	0.79	0.79

Tabella 7.27: Rapporto del classificatore K-Nearest Neighbors (KN)

	Precisione	Recall	F1-score
Non stress	0.68	0.91	0.78
Stress cognitivo	0.78	0.62	0.69
Stress emotivo	0.68	0.50	0.57
Stress sensorimotorio	0.89	0.62	0.73
Accuracy			0.72
Macro avg	0.76	0.66	0.69
Weighted avg	0.74	0.72	0.71

Tabella 7.28: Rapporto del classificatore Support Vector Machine (SVM)

	Precisione	Recall	F1-score
Non stress	0.85	0.77	0.81
Stress cognitivo	0.52	0.53	0.53
Stress emotivo	0.47	0.53	0.50
Stress sensorimotorio	0.68	0.71	0.69
Accuracy			0.67
Macro avg	0.63	0.64	0.63
Weighted avg	0.68	0.67	0.68

Tabella 7.29: Rapporto del classificatore Decision Tree

	Precisione	Recall	F1-score
Non stress	0.95	0.81	0.87
Stress cognitivo	0.59	0.72	0.65
Stress emotivo	0.59	0.67	0.62
Stress sensorimotorio	0.80	0.79	0.80
Accuracy			0.76
Macro avg	0.73	0.75	0.74
Weighted avg	0.79	0.76	0.77

Tabella 7.30: Rapporto del classificatore Random Forest

- Naive Bayes: mostra un'accuratezza del 56%. Questo è il più basso tra i cinque classificatori, suggerendo che potrebbe non essere il più adatto per questo compito specifico.
- Logistic Regression: mostra un'accuratezza del 66%. Questo è un miglioramento significativo rispetto al Naive Bayes, ma è ancora inferiore rispetto ad altri classificatori.
- KNN: mostra un'accuratezza del 79%. Questo è un ulteriore miglioramento rispetto alla Logistic Regression.
- SVM: mostra un'accuratezza del 72%, inferiore rispetto al KNN.
- Decision Tree: mostra un'accuratezza del 64%, che è inferiore rispetto al KNN e all'SVM.
- Random forest: mostra un'accuratezza del 76%.

In termini di falsi positivi e falsi negativi, si può dire che ci sono alcune differenze tra i classificatori. Ad esempio, il classificatore KNN sembra avere un numero relativamente equilibrato di falsi positivi e falsi negativi per ciascuna classe. Al contrario, il classificatore SVM sembra avere un numero maggiore di falsi positivi per la classe “normal drive” e un numero maggiore di falsi negativi per le altre classi. Infine, il classificatore Decision Tree sembra avere un numero relativamente equilibrato di falsi positivi e falsi negativi per ciascuna classe, simile al classificatore KNN.

7.3.3 StratifiedShuffleSplit con matrice di confusione a quattro etichette, classificatori base:

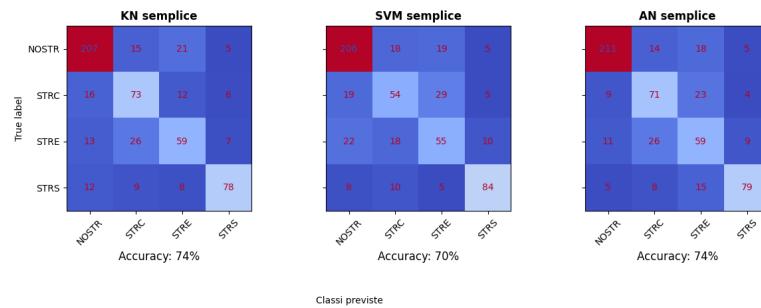


Figura 7.11: 7. Confusion Matrix

	Precisione	Recall	F1-score
Non stress	0.83	0.83	0.83
Stress cognitivo	0.59	0.68	0.63
Stress emotivo	0.59	0.56	0.58
Stress sensorimotorio	0.81	0.73	0.77
Accuracy			0.74
Macro avg	0.71	0.70	0.70
Weighted avg	0.74	0.74	0.74

Tabella 7.31: Rapporto del classificatore K-Nearest Neighbors (KN) semplice

	Precisione	Recall	F1-score
Non stress	0.81	0.83	0.82
Stress cognitivo	0.54	0.50	0.52
Stress emotivo	0.51	0.52	0.52
Stress sensorimotorio	0.81	0.79	0.80
Accuracy			0.70
Macro avg	0.67	0.66	0.66
Weighted avg	0.70	0.70	0.70

Tabella 7.32: Rapporto del classificatore Support Vector Machine (SVM) semplice

	Precisione	Recall	F1-score
Non stress	0.89	0.85	0.87
Stress cognitivo	0.60	0.66	0.63
Stress emotivo	0.51	0.56	0.54
Stress sensorimotorio	0.81	0.74	0.77
Accuracy			0.74
Macro avg	0.70	0.70	0.70
Weighted avg	0.75	0.74	0.75

Tabella 7.33: Rapporto del classificatore Artificial Neural Network (AN) semplice

- K-Nearest Neighbors (KNN): Questo classificatore ha ottenuto una precisione e un recall relativamente alti per tutte le classi, con la classe “Non stress” che ha ottenuto i punteggi più alti. L’accuracy complessiva è stata del 74%, il che indica un buon equilibrio tra precisione e recall per tutte le classi.
- Support Vector Machine (SVM): Questo classificatore ha ottenuto una precisione e un recall relativamente alti per le classi “Non stress” e “Stress sensorimotorio”, ma più bassi per le classi “Stress cognitivo” e “Stress emotivo”. L’accuracy complessiva è stata del 70%, il che indica che c’è spazio per miglioramenti, in particolare per le classi “Stress cognitivo” e “Stress emotivo”.
- Artificial Neural Network (ANN): Questo classificatore ha ottenuto una precisione e un recall relativamente alti per le classi “Non stress” e “Stress sensorimotorio”,

ma più bassi per le classi “Stress cognitivo” e “Stress emotivo”. Tuttavia, l’accuracy complessiva è stata del 74%, il che indica un miglioramento rispetto al classificatore SVM semplice.

In generale, sembra che il classificatore KNN e il classificatore ANN abbiano avuto le prestazioni migliori in termini di accuracy complessiva.

7.3.4 StratifiedShuffleSplit (SMOTE) con matrice di confusione a quattro etichette, classificatori base:

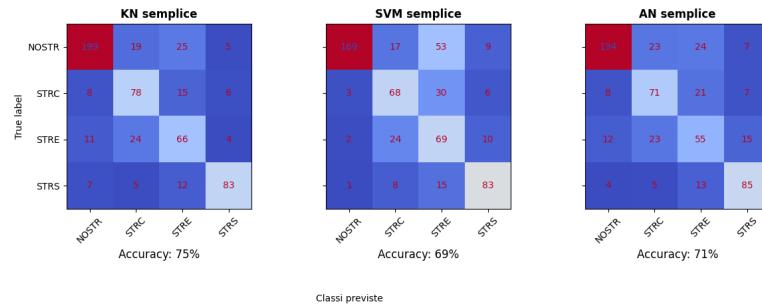


Figura 7.12: 8. Confusion Matrix

	Precisione	Recall	F1-score
Non stress	0.88	0.80	0.84
Stress cognitivo	0.62	0.73	0.67
Stress emotivo	0.56	0.63	0.59
Stress sensorimotorio	0.85	0.78	0.81
Accuracy			0.75
Macro avg	0.73	0.73	0.73
Weighted avg	0.77	0.75	0.76

Tabella 7.34: Rapporto del classificatore K-Nearest Neighbors (KNN) semplice

	Precisione	Recall	F1-score
Non stress	0.97	0.68	0.80
Stress cognitivo	0.58	0.64	0.61
Stress emotivo	0.41	0.66	0.51
Stress sensorimotorio	0.77	0.78	0.77
Accuracy			0.69
Macro avg	0.68	0.69	0.67
Weighted avg	0.75	0.69	0.70

Tabella 7.35: Rapporto del classificatore Support Vector Machine (SVM) semplice

	Precisione	Recall	F1-score
Non stress	0.89	0.78	0.83
Stress cognitivo	0.58	0.66	0.62
Stress emotivo	0.49	0.52	0.50
Stress sensorimotorio	0.75	0.79	0.77
Accuracy			0.71
Macro avg	0.68	0.69	0.68
Weighted avg	0.73	0.71	0.72

Tabella 7.36: Rapporto del classificatore Artificial Neural Network (AN) semplice

- K-Nearest Neighbors (KNN): Questo classificatore ha mostrato una precisione e un recall relativamente alti per tutte le classi, con la classe “Non stress” che ha ottenuto i punteggi più alti. L’accuracy complessiva è stata del 75%, il che indica un miglioramento rispetto all’analisi precedente senza SMOTE.
- Support Vector Machine (SVM): Questo classificatore ha mostrato una precisione e un recall relativamente alti per le classi “Non stress” e “Stress sensorimotorio”, ma più bassi per le classi “Stress cognitivo” e “Stress emotivo”. L’accuracy complessiva è stata del 69%, il che indica che c’è spazio per miglioramenti, in particolare per le classi “Stress cognitivo” e “Stress emotivo”. Tuttavia, rispetto all’analisi precedente senza SMOTE, il recall per la classe “Stress emotivo” è aumentato notevolmente.
- Artificial Neural Network (AN) semplice: Questo classificatore ha mostrato una precisione e un recall relativamente alti per le classi “Non stress” e “Stress sensorimotorio”, ma più bassi per le classi “Stress cognitivo” e “Stress emotivo”. Tuttavia, l’accuracy complessiva è stata del 71%.

7.4 Metodi di validazione incrociata più rigorosi

Nell’ultima analisi, sono state utilizzate due diverse tecniche di validazione incrociata: la Five Fold Cross Validation e la Leave One Subject Out. Queste tecniche sono state impiegate per valutare l’efficacia dei classificatori e per garantire che i risultati non fossero dovuti al sovrardattamento.

Tuttavia, a causa delle limitazioni computazionali, sono stati utilizzati solo i tre classificatori base (KNN, SVM e ANN) per questa analisi. Se fossero stati utilizzati tutti e cinque i classificatori ottimizzati, il tempo di elaborazione sarebbe stato eccessivamente lungo, date anche le prestazioni del pc. Pertanto, si è scelto di concentrarsi sui tre classificatori base per mantenere l'analisi gestibile e efficiente dal punto di vista temporale.

È importante notare che, sebbene l'uso di classificatori più sofisticati possa potenzialmente migliorare l'accuratezza del modello, l'efficienza computazionale è un fattore critico da considerare, soprattutto quando si lavora con grandi set di dati o con risorse di calcolo limitate.

7.4.1 'Five Fold Cross Validation' a 2 etichette

Sono state dunque applicate le due tecniche di validazione incrociata a ciascun segnale e calcolati i punteggi per ciascun classificatore. Questi punteggi, rappresentati sull'asse y del boxplot nella prossima figura, sono calcolati utilizzando la metrica '*f1_micro*', una misura di accuratezza che considera sia la precisione che il richiamo, rendendola una scelta appropriata per il set dati in questione che presenta classi sbilanciate. Il punteggio mostrato per ciascun segnale è la media dei punteggi ottenuti da tutti i classificatori ('KNN', 'SVM', 'ANN'). Quindi, per esempio, il punteggio finale del battito cardiaco, è la media delle valutazioni ottenute dal segnale 'HR' per ciascuno dei tre classificatori. Questo permette di avere una visione generale delle prestazioni di ciascun segnale indipendentemente dal classificatore utilizzato.

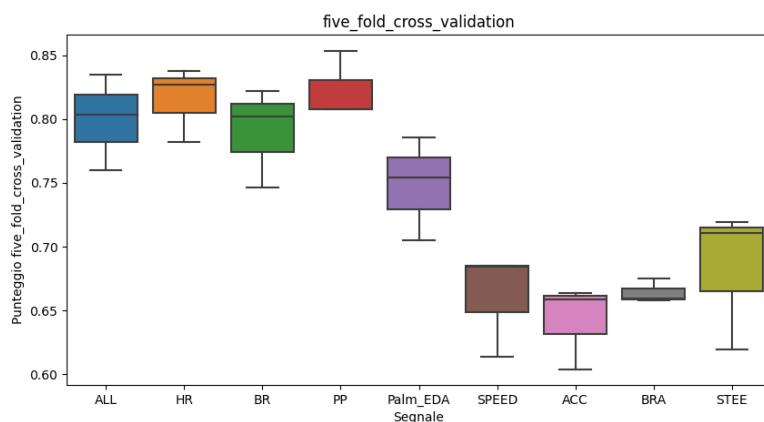


Figura 7.13: Boxplot dei punteggi di validazione incrociata 'Five Fold Cross Validation' a 2 etichette per diversi segnali utilizzando i classificatori KNN, SVM e ANN. L'asse y rappresenta il punteggio '*f1_micro*'

- ALL (insieme di tutte le features): L'uso di tutte le features ha portato ad un'accuratezza mediana che sembra essere inferiore rispetto ad alcuni singoli segnali. Questo suggerisce che non sempre la concatenazione di tutte le caratteristiche potrebbe portare a migliori prestazioni.
- HR (Heart Rate): Il classificatore basato sul battito cardiaco ha una mediana di accuratezza inferiore rispetto a PP, ma superiore rispetto a BR. Tuttavia, mostra

una variabilità maggiore nei risultati, indicando una consistenza minore nelle sue prestazioni.

- BR (Breathing Rate): Il classificatore basato sul respiro mostra una mediana di accuratezza inferiore rispetto a HR e PP, con una variabilità ancora maggiore nei risultati.
- PP: Questo classificatore ha una mediana di accuratezza superiore rispetto a HR e BR, e una variabilità minore rispetto a HR e BR, suggerendo prestazioni più consistenti e più accurate.
- Palm_EDA: Mostra un calo significativo nella mediana di accuratezza rispetto a PP, HR e BR, con una variabilità moderata nei risultati.
- SPEED: Questo classificatore mostra una delle mediane di accuratezza più basse, intorno a 0,65, e mostra anche una variabilità significativa nelle sue prestazioni.
- ACC (Acceleration): Il classificatore basato sull'accelerazione mostra prestazioni simili a SPEED, con una mediana di accuratezza bassa e una variabilità alta.
- BRA e STEE (Braking e Steering): hanno mediane molto basse, indicando prestazioni sia scarse che altamente variabili.

Nella Figura sottostante, sono riportati invece i punteggi di validazione incrociata ‘Five Fold Cross Validation’ per i tre classificatori utilizzati: KNN, SVM, e ANN. L’asse y rappresenta anche in questo il risultato ottenuto dalla metrica di accuratezza ‘f1_micro’. Il punteggio mostrato per ciascun classificatore è la media dei punteggi ottenuti da tutti i segnali. Dunque, per esempio, la valutazione finale ottenuta per il ‘KNN’ è la media dei punteggi ottenuti dal classificatore in questione per ciascuno dei segnali. Questo permette di avere una visione generale delle prestazioni di ciascun classificatore indipendentemente dal segnale utilizzato.

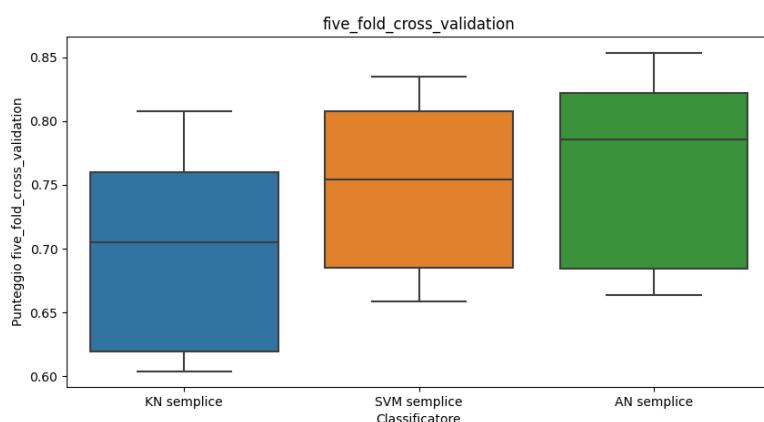


Figura 7.14: Boxplot dei punteggi di validazione incrociata ‘Five Fold Cross Validation’ a 2 etichette per i tre classificatori utilizzati: KNN, SVM, e ANN. L’asse y rappresenta il punteggio ‘f1_micro’

- KNN: Questo classificatore ha ottenuto un'accuratezza mediana di circa 0,75 con una variazione minima. Questo suggerisce che il modello KNN è relativamente stabile e consistente nelle sue previsioni.
- SVM: Questo classificatore ha ottenuto un'accuratezza mediana leggermente superiore a 0,75. Tuttavia, mostra una variazione più ampia rispetto al KNN, indicando che le prestazioni del modello SVM possono variare in modo più significativo a seconda del set di dati di addestramento e di test utilizzato.
- ANN: Questo classificatore ha ottenuto l'accuratezza mediana più alta, vicino a 0,8. Tuttavia, come l'SVM, anche l'ANN mostra una variazione significativa nei suoi punteggi di validazione incrociata, indicando che le sue prestazioni possono variare a seconda del set di dati di addestramento e di test utilizzato.

7.4.2 'Five Fold Cross Validation' a 4 etichette

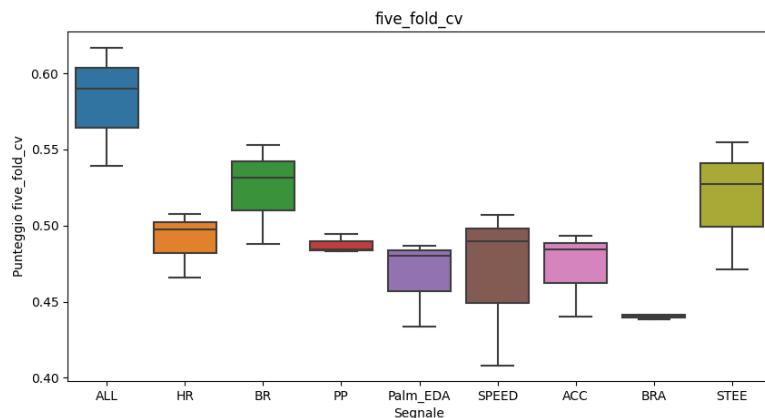


Figura 7.15: Boxplot dei punteggi di validazione incrociata 'Five Fold Cross Validation' a 4 etichette per diversi segnali utilizzando i classificatori KNN, SVM e ANN. L'asse y rappresenta il punteggio 'f1_micro'

Dall'analisi dei boxplot si può notare che l'accuratezza dei classificatori varia a seconda del tipo di segnale utilizzato. In particolare, l'uso di tutte le features (ALL) sembra portare a un'accuratezza più alta rispetto all'uso di singoli segnali. Questo suggerisce che la combinazione di diverse features può fornire informazioni più complete e quindi migliorare le prestazioni del modello per quanto riguarda la classificazione multclasse.

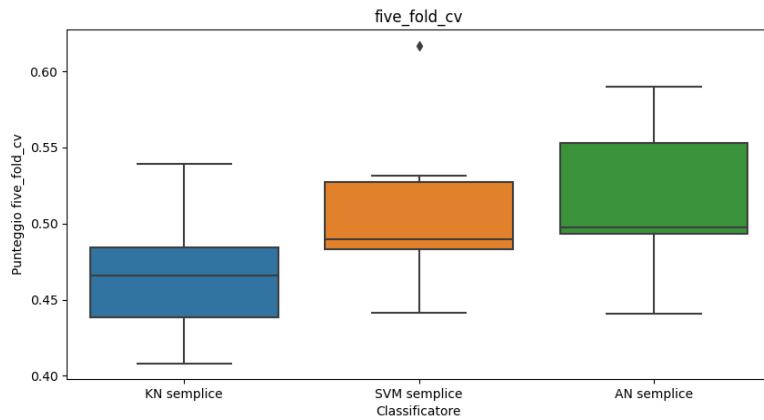


Figura 7.16: Boxplot dei punteggi di validazione incrociata ‘Five Fold Cross Validation’ a 4 etichette per i tre classificatori utilizzati: KNN, SVM, e ANN. L’asse y rappresenta il punteggio ‘f1_micro’

Per quanto riguarda i classificatori invece:

- KNN: Questo classificatore ha una mediana di accuratezza intorno a 0,45, con una variabilità moderata nei risultati come indicato dalla lunghezza dei baffi nel boxplot.
- SVM: Questo classificatore ha una mediana di accuratezza leggermente superiore rispetto a KN semplice, intorno a 0,52. Tuttavia, mostra una variabilità maggiore nei risultati, indicando una consistenza minore nelle sue prestazioni.
- ANN: Questo classificatore ha la mediana di accuratezza più alta, intorno a 0,55, e una variabilità minore rispetto a SVM semplice.

Anche in questo caso, come si può osservare, nonostante l’uso di tecniche avanzate come la cross-validation e l’uso di diversi classificatori, è molto difficile ottenere un’alta accuratezza nella classificazione multiclass dello stress.

7.4.3 'Leave One Subject Out' a 2 etichette

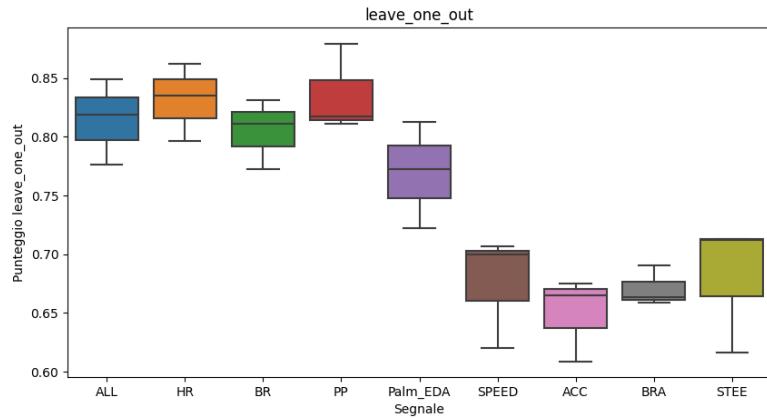


Figura 7.17: Boxplot dei punteggi di validazione incrociata 'Leave One Subject Out' a 2 etichette per diversi segnali utilizzando i classificatori KNN, SVM e ANN. L'asse y rappresenta il punteggio 'f1_micro'

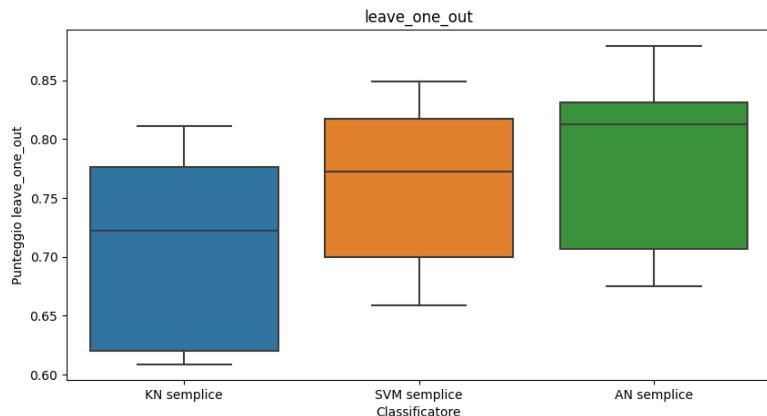


Figura 7.18: Boxplot dei punteggi di validazione incrociata 'Leave One Subject Out' a 2 etichette per i tre classificatori utilizzati: KNN, SVM, e ANN. L'asse y rappresenta il punteggio 'f1_micro'

Per quanto riguarda la classificazione "leave One Out", si può notare un andamento simile a quanto mostrato prima per la tecnica di validazione incrociata a cinque fold. I segnali fisiologici hanno ottenuto in generale punteggi più alti rispetto ai segnali telemetrici, ed il classificatore ANN risulta nuovamente quello con la mediana di accuratezza più alta.

7.4.4 'Leave One Subject Out' a 4 etichette

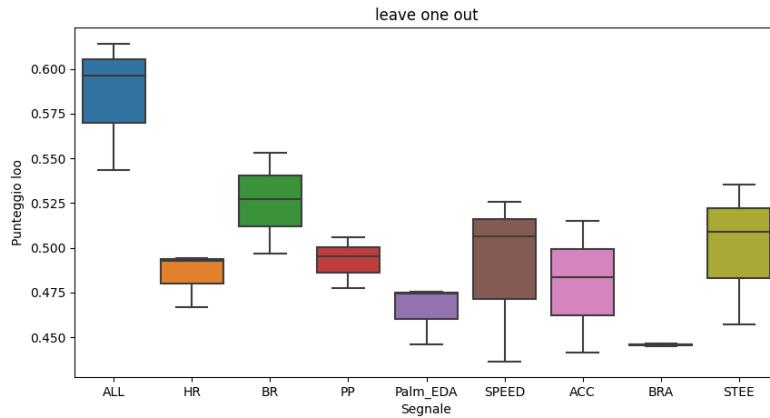


Figura 7.19: Boxplot dei punteggi di validazione incrociata 'Leave One Subject Out' a 4 etichette per diversi segnali utilizzando i classificatori KNN, SVM e ANN. L'asse y rappresenta il punteggio 'f1_micro'

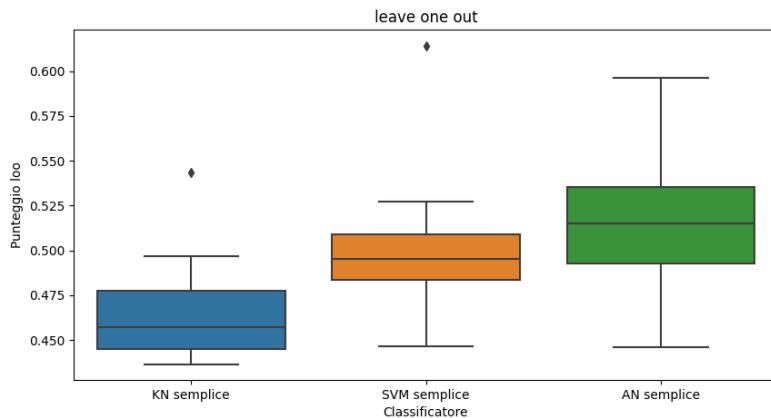


Figura 7.20: Boxplot dei punteggi di validazione incrociata 'Leave One Subject Out' a 4 etichette per i tre classificatori utilizzati: KNN, SVM, e ANN. L'asse y rappresenta il punteggio 'f1_micro'

7.5 Considerazioni finali

In sintesi, la '*Five Fold Cross Validation*' e la '*Leave One Out*' sono metodi di validazione più rigorosi che assicurano che il modello sia in grado di generalizzare bene su nuovi gruppi. Questi metodi utilizzano `cross_val_score`, una funzione di Scikit-learn utilizzata per eseguire la validazione incrociata su un modello, al fine di valutarne le prestazioni basandosi su diverse suddivisioni dei dati. Questo può aiutare a prevenire l'overfitting, che si verifica quando un modello è troppo complesso e si adatta troppo bene ai dati di

addestramento, non generalizzando bene sui nuovi.

D'altra parte, lo *Stratified Shuffle Split* può a volte produrre punteggi di validazione più alti perché mantiene la stessa percentuale di campioni per ciascuna classe come nel set di dati completo. Questo può essere particolarmente utile se le classi sono sbilanciate. Tuttavia, poiché Stratified Shuffle Split utilizza una singola suddivisione casuale dei dati, potrebbe non essere così robusto quanto la validazione incrociata per prevenire l'overfitting. Dunque, mentre Stratified Shuffle Split può a volte dare punteggi di validazione più alti, la validazione incrociata che utilizza la metrica del `cross_val_score` può fornire una stima più accurata di come il modello si comporterà su nuovi dati.

8

Capitolo 8

8.1 CONCLUSIONI

Nel corso della ricerca, è stato esplorato l'uso di diversi classificatori e segnali per prevedere lo stress del conducente, un problema importante dato il ruolo che lo stress può giocare negli incidenti stradali. Gli incidenti stradali rappresentano una delle principali cause di morte e lesioni in tutto il mondo, e la ricerca ha dimostrato che lo stress del conducente può contribuire significativamente a questo problema.

Sono stati utilizzati diversi classificatori in questo progetto. Dall'analisi dei risultati, sembra che il classificatore ANN abbia ottenuto i punteggi più alti in termini di accuratezza per quanto riguarda la sperimentazione effettuata con i "classificatori base", suggerendo che potrebbe essere il classificatore più efficace per questa specifica applicazione. Lo stesso discorso vale per l'SVM ed il Random Forest nell'analisi effettuata con i classificatori in cui è stata svolta una ricerca per ottimizzarne i parametri.

È stato anche esplorato l'uso di diversi tipi di segnali; i risultati indicano che i segnali fisiologici tendono ad avere un'accuratezza più alta rispetto ai segnali telemetrici. Questo suggerisce che i primi possono essere un indicatore più affidabile per il rilevamento dello stress. E' importante ricordare anche che l'esperimento si è svolto in un ambiente controllato utilizzando un simulatore di guida. Questo significa che le condizioni di stress simulate non possono essere paragonate direttamente a quelle che si verificherebbero in uno scenario reale. Pertanto, il carico di stress percepito dai partecipanti durante l'esperimento potrebbe differire da quello che si proverebbe in una vera situazione di guida su strada.

Tuttavia, c'è ancora spazio per migliorare e ottimizzare questo lavoro. Ad esempio, potrebbe essere utile:

- esplorare ulteriormente l'uso di tecniche di selezione delle features per identificare caratteristiche più informative.

- Valutare diverse tecniche di ottimizzazione dei parametri per migliorare le prestazioni dei classificatori.
- Utilizzare tecniche di apprendimento profondo, che si concentrano sull'uso di reti neurali a più strati e che hanno dimostrato di essere particolarmente efficaci in molte applicazioni nel campo del machine learning . Questi strati permettono al modello di apprendere automaticamente le features dai dati, eliminando la necessità di selezionarle manualmente. Le tecniche di apprendimento profondo sono valide anche per una vasta gamma di applicazioni, tra cui la classificazione delle immagini, il riconoscimento vocale, e la traduzione automatica.

Nel contesto di questa ricerca, l'uso di tecniche di apprendimento profondo potrebbe implicare l'utilizzo di reti neurali per classificare i diversi stati di stress basandosi sui segnali fisiologici e telemetrici. Ad esempio, si potrebbe utilizzare una rete neurale convoluzionale (CNN) per analizzare i segnali temporali, o una rete neurale ricorrente (RNN) per analizzare i segnali che cambiano nel tempo.

In conclusione, la ricerca ha fornito diverse intuizioni sul problema della previsione dello stress del conducente. I risultati suggeriscono che l'uso di classificatori avanzati e dei segnali fisiologici possono essere un approccio promettente per affrontare questo problema. Le informazioni raccolte potrebbero essere utili per chiunque sia interessato a questa importante area di studio.

Colophon

La tesi è stata scritta utilizzando il linguaggio LaTeX.

Il template grafico è stato sviluppato da Carullo Moreno modifiche di Nicola Landro e Ignazio Gallo.

Il lavoro è stato svolto utilizzando il linguaggio di programmazione Java sul framework Android utilizzando l'ide Android Studio.

Le immagini sono state create appositamente con Gimp e Drawio, oppure sono degli screenshot di un dispositivo o emulatore.

Bibliografia

- [1] S. Taamneh, P. Tsiamyrtzis, M. Dcosta, P. Buddharaju, A. Khatri, M. Manser, T. Ferris, R. Wunderlich, and I. Pavlidis, “A multimodal dataset for various forms of distracted driving,” *Scientific data*, vol. 4, no. 1, pp. 1–21, 2017.
- [2] S. Bianco, P. Napoletano, and R. Schettini, “Multimodal car driver stress recognition,” in *Proceedings of the 13th EAI International Conference on Pervasive Computing Technologies for Healthcare*, pp. 302–307, 2019.
- [3] A. R. Perry and D. A. Baldwin, “Further evidence of associations of type a personality scores and driving-related attitudes and behaviors,” *Perceptual and motor skills*, vol. 91, no. 1, pp. 147–154, 2000.
- [4] P. Napoletano and S. Rossi, “Combining heart and breathing rate for car driver stress recognition,” in *2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*, pp. 1–5, IEEE, 2018.
- [5] R. Zangróniz, A. Martínez-Rodrigo, J. M. Pastor, M. T. López, and A. Fernández-Caballero, “Electrodermal activity sensor for classification of calm/distress condition,” *Sensors*, vol. 17, no. 10, p. 2324, 2017.
- [6] J. A. Healey and R. W. Picard, “Detecting stress during real-world driving tasks using physiological sensors,” *IEEE Transactions on intelligent transportation systems*, vol. 6, no. 2, pp. 156–166, 2005.
- [7] M. Alibeigi, W. Ljungbergh, A. Tonderski, G. Hess, A. Lilja, C. Lindström, D. Motorniuk, J. Fu, J. Widahl, and C. Petersson, “Zenseact open dataset: A large-scale and diverse multimodal dataset for autonomous driving,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20178–20188, 2023.
- [8] Y. Choi, S. I. Han, S.-H. Kong, and H. Ko, “Driver status monitoring systems for smart vehicles using physiological sensors: A safety enhancement system from automobile manufacturers,” *IEEE Signal Processing Magazine*, vol. 33, no. 6, pp. 22–34, 2016.
- [9] J. M. Cooper, I. Vladisljevic, N. Medeiros-Ward, P. T. Martin, and D. L. Strayer, “An investigation of driver distraction near the tipping point of traffic flow stability,” *Human factors*, vol. 51, no. 2, pp. 261–268, 2009.
- [10] I. Pavlidis, M. Dcosta, S. Taamneh, M. Manser, T. Ferris, R. Wunderlich, E. Akleman, and P. Tsiamyrtzis, “Dissecting driver behaviors under cognitive, emotional, sensorimotor, and mixed stressors,” *Scientific reports*, vol. 6, no. 1, p. 25651, 2016.

- [11] P. Tsiamyrtzis, M. Dcosta, D. Shastri, E. Prasad, and I. Pavlidis, “Delineating the operational envelope of mobile and conventional eda sensing on key body locations,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 5665–5674, 2016.
- [12] D. Shastri, M. Papadakis, P. Tsiamyrtzis, B. Bass, and I. Pavlidis, “Perinasal imaging of physiological stress and its affective potential,” *IEEE Transactions on Affective Computing*, vol. 3, no. 3, pp. 366–378, 2012.