Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

STUDIO E IMPLEMENTAZIONE DI UN FLUSSO OPERATIVO PER LA CRASH DETECTION TRAMITE DISPOSITIVI MOBILI

Relatore:
Dott.
Luca Sciullo

Presentata da:
Simone Folli

Correlatori:
Dott.
Alfonso Esposito,
Chiar.mo Prof.
Luca Bedogni

Sessione
Anno Accademico 2023-2024

Abstract

La tesi esplora l'applicazione di metodologie tipiche della Human Activity Recognition al problema della Crash Detection, impiegando i sensori integrati nei dispositivi mobili e algoritmi di apprendimento automatico per classificare eventi di schianto o frenata. Poiché non è stato possibile raccogliere dati reali su incidenti automobilistici, è stato realizzato un dataset simulando il comportamento di un'autovettura in un incidente mediante l'uso di un carretto. Il lavoro descrive in dettaglio la registrazione dei dati, la simulazione dell'incidente, il preprocessing dei dati, il calcolo delle features e le tecniche di bilanciamento adottate. Successivamente sono presentati gli algoritmi di apprendimento utilizzati e le metriche di valutazione. I risultati sperimentali dimostrano che i modelli scelti, addestrati sul nostro dataset, riescono efficacemente a classificare il tipo di evento; i migliori risultati sono stati ottenuti usando modelli basati su ensemble learning, con F1 score su Anomaly Detection di circa 98%, e con F1 score su Anomaly Classification di circa 95%.

Indice

Elenco delle figure

Capitolo 1

Introduzione

L'avanzamento tecnologico degli ultimi anni ha visto una proliferazione di sensori sempre più sofisticati e accessibili, come accelerometri e giroscopi, integrati in dispositivi mobili di uso quotidiano, in particolare negli smartphone. Questa onnipresenza apre nuovi scenari nel campo della sicurezza automobilistica. La natura pervasiva di tali dispositivi, che ormai accompagnano la vita di quasi ogni individuo, suggerisce la possibilità di sfruttarli come strumenti di monitoraggio e identificazione di anomalie durante la guida.

In questa tesi si propone di esplorare la possibilità di applicare approcci usati nel campo della *Human Activity Recognition*, per problemi nel campo della *Crash Detection*. Usando i sensori integrati nei dispositivi mobili insieme ad algoritmi di apprendimento automatico, andremo a vedere se è possibile classificare efficacemente eventi di schianto o frenata.

Dato che per ragioni ovvie è stato impossibile raccogliere dati su schianti in automobile, sufficienti per un'analisi rigorosa, si è pensato di creare un dataset utilizzando un carretto per simulare il comportamento di un'autovettura nel momento di un incidente. L'obiettivo è capire se questo approccio per la creazione del dataset può essere usato per l'addestramento di modelli nella classificazione di eventi anomali alla guida.

Nel secondo capitolo, dal titolo *Stato dell'arte*, viene discusso l'avanzamento scientifico nei campi di ricerca a cui ci siamo ispirati per questa tesi. I campi più importanti includono *Human Activity Recognition*, *Crash Detection* ed *Event Detection*. Più specificatamente vengono descritte tecnologie note su: sensori triassiali; tecniche di bilanciamento dei dataset; tecniche di Feature Engineering e algoritmi di apprendimento supervisionato adatti nel contesto di ricerca; per quest'ultimo punto è stato molto rilevante il lavoro svolto da Fernández Delgado in [9] per la scelta dei modelli da utilizzare.

Nel terzo capitolo, dal titolo *Flusso operativo*, una volta definito l'obiettivo, viene discusso l'approccio proposto spiegando nel dettaglio il procedimento adottato per registrare i dati in modo efficace con un dispositivo mobile; successivamente, vengono descritte la strategia per simulare un incidente in auto, le metodologie adottate per il preprocessing e il calcolo delle features. Infine, sono descritte le tecniche scelte per il bilanciamento del dataset, ispirate dal lavoro di Ramyachitra in [23]; e gli algoritmi di apprendimento automatico insieme alle metriche di valutazione selezionate. Nel quarto capitolo viene spiegato nel dettaglio come sono state implementate le tecniche descritte nel flusso operativo, andando nel dettaglio delle tecnologie utilizzate e dell'applicazione sviluppata.

Nel quinto ed ultimo capitolo sono presentati i risultati degli esperimenti svolti, con un'analisi su diversi aspetti del dataset e dei modelli utilizzati. È stata fatta un'analisi iniziale sulla segmentazione del dataset per il calcolo delle features; successivamente, è stata fatta una ricerca degli iperparametri sui modelli che ci ha mostrato come il Random Forest ottenga i risultati migliori su tutte le metriche di valutazione. Infine viene analizzato l'impatto del bilanciamento del dataset sulla capacità dei modelli di apprendere la classificazione. Le sperimentazioni fatte hanno dimostrato che i modelli scelti, addestrati sul nostro dataset, riescono efficacemente a classificare il tipo di evento; i migliori risultati sono stati ottenuti usando modelli basati su ensemble learning, con F1 score su Anomaly Detection di circa 98%, e con F1 score su Anomaly Classification di circa 95%.

Capitolo 2

Stato dell'arte

In questo capitolo si esamina lo stato dell'arte relativo agli ambiti di ricerca strettamente connessi al nostro studio. Verranno analizzate le principali linee di indagine e i contributi scientifici che hanno segnato l'evoluzione dei metodi e delle tecnologie adottate nel settore. Tale analisi ci consentirà di definire in maniera critica le basi teoriche e pratiche su cui si fonda l'approccio adottato.

2.1 Human Activity Recognition

Human Activity Recognition (HAR) è un problema ampiamente studiato; secondo [26], il feedback più vitale ed essenziale necessario per sviluppare applicazioni IoT (Internet of Things) intelligenti, è il risultato del processo di percezione dei comportamenti umani e della loro interazione fisica con l'ambiente circostante. Per ottenere un feedback accurato e utile sui comportamenti e sulle interazioni umane, è necessario integrare i fattori di inferenza e rilevamento utilizzando il campo di ricerca del riconoscimento dell'attività umana (HAR). Questo interesse deriva dalla necessità di ottenere dati contestualizzati, che a loro volta sono impiegati per fornire un supporto personale agli utenti su un'ampia varietà di set di applicazioni, come applicazioni di sicurezza, mediche, militari e di stile di vita [14]. Prenderemo ispirazione da questo ambito di ricerca applicandolo a quello della Crash detection.

2.2 Crash Detection

L'ambito di ricerca del Crash detection mediante l'utilizzo del sensore di accelerazione rappresenta un tema di notevole rilevanza in vari contesti applicativi, quali la sicurezza stradale, i sistemi di guida assistita e le soluzioni IoT per la smart mobility. L'obiettivo principale di questa linea di ricerca è rilevare automaticamente la presenza di impatti, riducendo così i tempi di reazione dei soccorsi e limitando la necessità di intervento manuale da parte degli utenti.

Gli studi condotti da Rani [24] e Ayyorgun [3] evidenziano che l'approccio più diffuso per il crash detection prevede l'impiego di hardware dedicato, come il sensore LSM9DS1 9-axis IMU, integrato con microcontrollori quali Raspberry Pi. In particolare, la configurazione proposta da Rani [24] prevede l'installazione di un sensore in grado di monitorare in tempo reale le accelerazioni lungo i tre assi. Quando il sensore registra uno scostamento superiore a una soglia prestabilita, viene attivata una procedura di allarme, evidenziando l'efficacia di un approccio IoT per il rilevamento immediato degli incidenti. Parallelamente, Ayyorgun [3] dimostra come anche una rete neurale di dimensioni contenute possa essere impiegata con successo per il rilevamento degli impatti.

È importante sottolineare che i sensori MEMS utilizzati in queste configurazioni rappresentano la stessa tipologia di dispositivi impiegati negli smartphone moderni. Considerando che i dispositivi mobili odierni dispongono di una notevole potenza computazionale e di sensori integrati capaci di rilevare in modo accurato le variazioni di accelerazione, si apre la prospettiva di sfruttare tali apparecchiature per applicazioni di crash detection e, più in generale, per sistemi di monitoraggio degli eventi. Questa convergenza tra tecnologia dedicata e dispositivi di largo consumo rende particolarmente interessante l'indagine sull'applicabilità degli smartphone in questo contesto.

2.3 Event Detection

L'argomento che esamineremo può essere inquadrato anche come un problema di Event Detection (ED). secondo Margineantu e Wong [20], event detection viene definito come il processo di monitoraggio (di dati regolarmente raccolti) finalizzato a individuare automaticamente eventi "interessanti", evitando l'analisi manuale da parte di esperti. In altre parole, è l'attività di sorvegliare e analizzare una varietà di dati (numerici, immagini, video, audio, testuali, ecc.) per rilevare automaticamente situazioni o fenomeni degni di nota. Gli algoritmi di apprendimento automatico, combinati con appro-

priate tecniche di feature engineering e strategie di data augmentation (ad esempio, l'oversampling), rappresentano alcuni tra gli approcci più solidi per affrontare problemi di event detection [19, 1].

2.4 Sensori

I sensori svolgono un ruolo fondamentale nel rilevamento di eventi e nel riconoscimento delle attività umane. I sensori sono il primo elemento del sistema per la raccolta di dati. Diversi sensori possono essere usati per il rilevamento di eventi, come accelerometro, giroscopio, magnetometro, videocamere, microfoni. Gli accelerometri sono dispositivi elettromeccanici in grado di misurare sia le forze di accelerazione statiche che dinamiche. Le forze statiche definiscono la gravità, mentre le forze dinamiche misurano il movimento e le vibrazioni. Gli accelerometri triassiali ottengono tutte le vibrazioni di una struttura prendendo simultaneamente misurazioni in tre direzioni ortogonali. L'accelerometro triassiale è uno dei sensori più importanti all'interno dei dispositivi mobili moderni per la HAR.

Secondo [16], un accelerometro può rilevare le misurazioni dell'attività per catturare sofisticati movimenti dell'utente come inclinazione, oscillazione o rotazione. L'accelerometro verrà affiancato ad altri due sensori: giroscopio e magnetometro. Questi ci possono dare una stima accurata della posizione del dispositivo mobile [17, 4].

2.5 Dataset

Nel contesto della ricerca sul rilevamento e sulla classificazione degli eventi, in particolare mediante l'impiego di algoritmi di apprendimento automatico supervisionato e non supervisionato, emerge una problematica rilevante legata alla presenza di dataset sbilanciati. Molto spesso, infatti, la distribuzione delle classi risulta fortemente asimmetrica, con un numero significativamente inferiore di istanze appartenenti ad alcune classi rispetto ad altre. Come evidenziato in [23], la principale difficoltà associata a questo problema consiste nel fatto che le classi meno rappresentate spesso rivestono un ruolo cruciale nelle applicazioni pratiche, mentre i classificatori standard tendono a essere influenzati dalla classe predominante, trascurando le classi minoritarie.

Per mitigare gli effetti dello sbilanciamento nei dataset, sono state proposte diverse strategie; quelle più rilevanti per il presente studio sono le seguenti:

Approcci basati sulla manipolazione dei dati

Questi metodi operano nella fase di pre-elaborazione dei dati con l'obiettivo di riequilibrare la distribuzione delle classi, migliorando così le prestazioni del modello. I metodi sono:

Sovracampionamento (Oversampling); Tecniche come *SMOTE (Synthetic Minority Over-sampling Technique)* e *ADASYN (Adaptive Synthetic Sampling Approach)* consentono di generare nuove istanze sintetiche della classe minoritaria, contribuendo a ridurre il divario numerico tra le classi.

Sottocampionamento (Undersampling); Questa tecnica riduce il numero di istanze della classe maggioritaria al fine di mitigare la tendenza del modello a favorire le classi più numerose.

Approcci basati sulla modifica degli algoritmi di apprendimento

Questi metodi agiscono direttamente sugli algoritmi di apprendimento per migliorarne la capacità di classificare correttamente le istanze appartenenti alla classe minoritaria. I metodi sono:

Support Vector Machine (SVM), introduce pesi differenziati per le classi al fine di contrastare l'effetto dello sbilanciamento e migliorare la capacità di generalizzazione del modello.

K-Nearest Neighbors (KNN), implementa strategie di ponderazione che riducono l'influenza della classe maggioritaria, favorendo una classificazione più equa delle classi minoritarie.

Random Forest, adotta strategie di campionamento con sostituzione per garantire una distribuzione più equilibrata dei dati durante il processo di apprendimento.

Queste strategie sono essenziali per affrontare il problema dello sbilanciamento, permettono di aumentare l'affidabilità e la robustezza delle previsioni.

2.6 Feature Engineering

Una feature è definita come ogni valore che può essere calcolato da dati grezzi con l'obiettivo di modellare il problema per l'algoritmo di ML. Distinguere i dati grezzi dalle feature rende esplicita la decisione di modellazione che riguarda la scelta e l'assemblaggio dei set di feature. Se i dati grezzi sono in formato tabellare, ogni riga può rappresentare un'istanza e potrebbe risultare naturale considerare ogni colonna come una feature. Tuttavia, decidere quali colonne siano effettivamente feature e quale tipo di preprocessing

(incluso il clustering, ecc.) debba essere applicato per ricavarle è un compito strettamente legato al problema che si vuole risolvere.

Quando si affronta un problema di apprendimento automatico (ML), le classi target e le istanze sono di solito definite in anticipo come parte della formulazione del problema. Esse rientrano in ciò che solitamente vengono definiti come dati grezzi (a volte riferiti anche come variabili o attributi). Tali dati grezzi sono normalmente il risultato di un'attività di raccolta dati, talvolta attraverso appositi "data collection hooks" in un sistema in produzione, e le classi target possono essere ricavate dal sistema stesso oppure tramite annotazione umana. Le feature in sé non sono così nettamente definite: per passare dai dati grezzi alle feature è necessario estrarle attraverso un processo di featurizzazione all'interno di una data pipeline. Questo processo va di pari passo con la pulizia e il potenziamento dei dati. La distinzione tra dati grezzi e features è chiave e ci apre la strada per definire come effettuare una Ingegnerizzazione delle features (FE).

Feature engineering è il processo di rappresentare un dominio di problema in modo da renderlo adatto alle tecniche di apprendimento. Questo processo include la scoperta iniziale delle feature e il loro miglioramento graduale basato sulle conoscenze di dominio e sulle prestazioni osservate di un determinato algoritmo di apprendimento automatico su dati di addestramento specifici [8].

2.7 Algoritmi di apprendimento

Esistono due tipi di algoritmi di apprendimento automatico per l'elaborazione e l'analisi dei dati HAR ampiamente noti: algoritmi di apprendimento automatico supervisionati e non supervisionati. Nell'apprendimento supervisionato, è necessario sostenere l'algoritmo utilizzato con conoscenze pregresse ed etichettare manualmente i dati. Per raggiungere questo obiettivo, i set di dati di training, che sono fondamentalmente i dati standard, devono essere creati e immessi nell'algoritmo utilizzato. In questo caso, l'algoritmo verrà addestrato per applicare i passaggi di elaborazione richiesti sui dati ricevuti dagli utenti, comunemente chiamati dati di test. D'altra parte, l'apprendimento non supervisionato non necessita di una conoscenza pregressa. Quindi, l'algoritmo non supervisionato è costruito per risolvere problemi complessi riconoscendo determinati modelli senza interferenza umana. Ci sono due fattori principali per scegliere quale tipo di algoritmi di apprendimento automatico sia adatto alla propria ricerca: il volume e la struttura dei tuoi dati. Poiché il volume di dati non è enorme e il livello di complessità non è ele-

vato, in questo lavoro andremo a testare diversi algoritmi di apprendimento supervisionato per valutarne le prestazioni e l'accuratezza [27].

Il primo modello che andremo a testare è la regressione logistica (LR) è un modello statistico ampiamente utilizzato per la classificazione binaria [18], in cui la variabile dipendente assume due possibili valori. In questo approccio, la probabilità che un'osservazione appartenga a una determinata classe viene modellata attraverso una funzione logistica, comunemente detta funzione sigmoide, che trasforma una combinazione lineare dei predittori in un intervallo compreso tra 0 e 1.

Zunash Zaki nel suo studio del 2020 [30], dimostra che LR unito alla tecnica: *Principle Component Analysis*, per ridurre la dimensionalità delle features, può ottenere dei buoni risultati nell'ambito della HAR. A differenza di modelli quali i Decision Trees o le Support Vector Machine (SVM), la regressione logistica si distingue per la sua solida interpretazione statistica e per la facilità con cui il modello può essere aggiornato per includere nuovi dati. Rispetto all'analisi discriminante, la regressione logistica offre il vantaggio di richiedere minori assunzioni sui dati: non è infatti necessario assumere una distribuzione normale per le variabili indipendenti, né è richiesta una relazione lineare diretta tra i predittori e la variabile target. Tuttavia, per ottenere stime stabili ed affidabili dei parametri, è spesso necessario disporre di campioni di dimensioni elevate, soprattutto in presenza di numerose variabili esplicative [18, 6].

Il secondo modello che andremo a testare è Decision Trees (DT), è un modello non parametrico che gestisce efficacemente le interazioni tra le variabili, riducendo l'impatto dei valori anomali e consentendo la classificazione anche in presenza di dati linearmente inseparabili. Algoritmi noti quali ID3, C4.5, C5.0 e CART si basano su criteri di suddivisione come il coefficiente di Gini e il guadagno di informazioni [25]. I DT sono in grado di gestire una varietà di tipi di dati (nominali, numerici, testuali), dati mancanti e attributi ridondanti; presentano una buona capacità di generalizzazione e sono robusti al rumore, il tutto con uno sforzo computazionale relativamente contenuto. Tuttavia, la gestione di dati ad alta dimensionalità può risultare problematica. L'approccio "dividi et impera" funziona bene in presenza di pochi attributi altamente rilevanti, ma può incontrare difficoltà se esistono numerose interazioni complesse, con errori che si propagano lungo l'albero, un problema che si aggrava con l'aumentare del numero di classi [29, 2].

Inoltre, con la crescita dell'albero, il numero di record nei nodi foglia può diventare troppo piccolo. Per garantire decisioni statisticamente significative, fenomeno noto come frammentazione dei dati. Tale problema può essere mitigato imponendo una soglia minima sul numero di record per ulteriori suddivisioni. Infine, senza un'adeguata potatura, i DT sono soggetti a sovra-

adattamento, motivo per cui è stata proposta l'adozione di modelli ensemble, come la Random Forest.

Random Forest (RF) è un ensemble di classificatori composto da alberi decisionali [18] che vengono generati utilizzando due diverse fonti di randomizzazione. Uno dei metodi di machine learning di maggior successo(Breiman [5]).

Basandosi sulla legge dei grandi numeri, si può dimostrare che l'errore di generalizzazione delle random forests converge a un limite man mano che il numero di alberi nella foresta aumenta [5]. L'implicazione di tale dimostrazione è che la dimensione dell'ensemble non rappresenta un iperparametro che necessiti di essere ottimizzato, poiché l'accuratezza di generalizzazione della random forest non peggiora in media al crescere del numero di classificatori inclusi. Più numerosi sono gli alberi nella foresta, maggiore è la probabilità che l'ensemble converga al suo errore asintotico. In effetti, uno dei principali vantaggi della random forest risiede nel fatto che è quasi priva di iperparametri, o quantomeno che l'impostazione predefinita degli iperparametri offra prestazioni notevoli [9]. Nurwulan, nella sua ricerca [21], evidenzia come Random Forest (RF) rappresenti frequentemente la scelta più appropriata per i problemi di Human Activity Recognition (HAR), considerando sia il compromesso tra costo computazionale e prestazioni del modello, sia la sua capacità di generalizzazione.

In particolare, lo studio dimostra che Random Forest offre un equilibrio ottimale tra accuratezza, interpretabilità e tempo di addestramento, rendendolo una soluzione competitiva rispetto ad altri algoritmi di apprendimento automatico. Inoltre, la sua robustezza rispetto ai dati rumorosi e la capacità di gestire feature non lineari lo rendono particolarmente efficace nell'ambito dell'HAR, dove la qualità dei dati sensoriali può variare significativamente.

L'ultimo algoritmo che andremo a testare è Gradient Boosting (GB), un algoritmo di boosting per la regressione [12]. Gli algoritmi di boosting combinano weak learners, ovvero learners leggermente migliori di quelli random, in un learner forte in modo iterativo [11]. S.Rahman nella sua ricerca [22] mostra come gli algoritmi di boosting possano essere applicati al problema delle HAR, ottenendo risultati competitivi. Questo algoritmo può soffrire di overfitting se il processo iterativo non è correttamente regolato [12]. Contrariamente a Random Forest, i valori predefiniti per questi iperparametri nel GB vengono impostati per limitarne il potere espressivo; ad esempio, la profondità è generalmente limitata a $\approx 3-5$. Un'altra famiglia di iperparametri presenti in diverse versioni di GB, riguarda la randomizzazione dei learner di base, il che può migliorare ulteriormente la capacità di generalizzazione dell'ensemble [13], come il sottocampionamento casuale senza sostituzione.

Capitolo 3

Flusso operativo

In questa sezione, viene presentato l'approccio proposto per il fine ultimo descritto sotto. L'approccio include le seguenti fasi: creazione di un'applicazione ad hoc per dispositivi mobili; raccolta dei dati con l'utilizzo del dispositivo mobile, del carretto e di un ostacolo per l'impatto; l'elaborazione del dataset; tecniche di estrazione delle caratteristiche (features); addestramento di diversi modelli e valutazione dei risultati.

3.1 Obiettivo

L'obiettivo della presente tesi consiste nella creazione di un dataset finalizzato al riconoscimento degli impatti su veicoli in movimento. Poiché la sperimentazione diretta su veicoli reali risulta impraticabile, si è scelto di adottare un carrello, seguendo l'ipotesi che la rilevazione di eventi su un oggetto soggetto a impatti di entità inferiore possa essere traslata al contesto di un veicolo reale. Con il dataset realizzato è stata condotta un'analisi preliminare atta a verificare la correttezza delle ipotesi formulate. È importante precisare che questo studio rappresenta una fase iniziale e che, in futuri sviluppi, si intende approfondire l'applicazione dei risultati al caso dei veicoli reali.

3.2 Creazione dell'applicazione

In primo luogo, è stata sviluppata un'applicazione Android per la raccolta delle misurazioni. L'app consente di acquisire i dati da tre sensori:

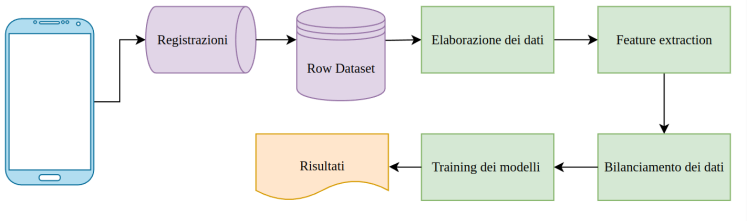


Figura 3.1: Pipeline proposta

accelerometro, giroscopio e magnetometro triassiali, per un totale di 9 valori grezzi.

Per garantire una maggiore precisione nella fase di etichettatura dei dati, l'applicazione implementa un protocollo di comunicazione Bluetooth che consente la connessione tra due dispositivi. Questo approccio offre due vantaggi significativi: il primo è che abbiamo una riduzione delle interferenze sui dati sensoristici; poiché l'etichettatura degli impatti viene effettuata da un dispositivo remoto, si evita la contaminazione dei dati dovuta al contatto diretto con il touchscreen del dispositivo che esegue la registrazione. Il secondo vantaggio è una migliore precisione nella marcatura temporale degli impatti: dato che il dispositivo di raccolta dati è in movimento, la connessione remota consente di registrare con maggiore accuratezza l'istante esatto dell'impatto, senza ritardi dovuti all'interazione manuale.

Questa soluzione migliora l'affidabilità del dataset, garantendo dati più puliti e una sincronizzazione più accurata tra le misurazioni sensoristiche e gli eventi di impatto o frenata.

3.3 Creazione dei dataset

In uno scenario reale, in cui si verifica un impatto tra il veicolo del soggetto e un altro oggetto, il telefono potrebbe trovarsi in diverse posizioni, ad esempio fissato al cruscotto tramite un supporto, nella tasca o alloggiato nel porta bicchieri. Durante la raccolta dei dati, abbiamo considerato alcune di queste possibilità, simulando diversi scenari. Progetteremo questo esperimento per emulare il comportamento di un'autovettura.

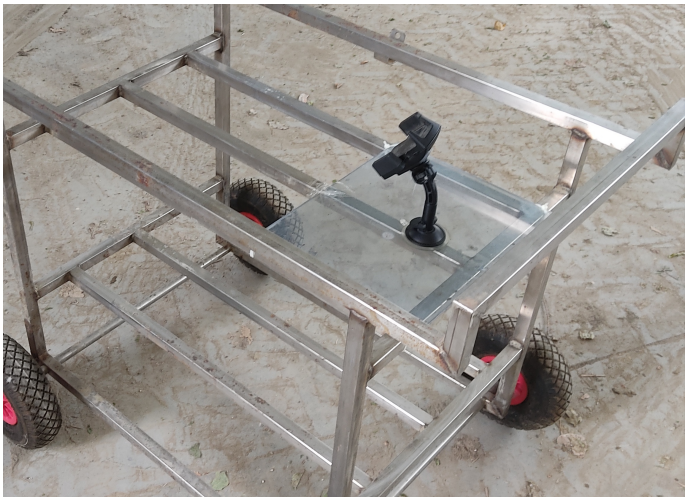


Figura 3.2: Supporto per telefono con ventosa montato sul carrello

Raccolta dei dati

Per raccogliere i dati dei sensori, oltre all'applicazione sopra descritta, è stato usato un carrello in acciaio con quattro ruote. Sul carrello sono stati montati tre supporti diversi per il dispositivo; il primo è un supporto in plastica con ventosa (fig. 3.2), scelto perché spesso viene utilizzato per tenere il telefono in macchina; il secondo è una scatola in cartone per simulare un portaoggetti; il terzo è un pantalone in cotone con all'interno imbottitura, per simulare, durante un impatto, l'ammortizzazione che il telefono può subire stando in una tasca.

La rampa per il carrello è una superficie piana lunga circa 10 metri che termina su un palo in cemento armato coperto con assi in legno, per evitare di rovinare la struttura e il carrello. I primi cinque metri della rampa verranno usati per accelerare il veicolo; solo dopo, quando viene raggiunta la velocità, si avvia la registrazione dei sensori, che termina dopo pochi secondi dall'impatto o dalla frenata.

Per avere una maggiore varianza nei dati, l'esperimento è stato registrato su diversi angoli d'impatto, con il telefono in diversi orientamenti e con una velocità che variava dai $12km/h$ ai $16km/h$. Ogni registrazione contiene quindi dati raccolti durante un andamento regolare e dati raccolti durante una fase di impatto o frenata.

Per le registrazioni relative allo schianto, il carrello viene spinto sulla rampa per i primi cinque metri e lasciato andare per i successivi cinque; nel momento in cui avviene l'impatto contro il pannello in legno, l'istante viene

(a) Scatola in cartone montata sul carrello per la simulazione su un porta oggetti

(b) Pantalone il stoffa, riempito con imbottitura per simulare il comportamento di una tasca di pantalone

Figura 3.3

segnato manualmente utilizzando il dispositivo collegato da remoto. Per le registrazioni relative alla frenata, il carrello è stato fissato tramite una corda, in materiale sintetico, ad un attacco posizionato dalla parte opposta alla direzione di lancio. La lunghezza della corda è di quindici metri circa, ed è stata calcolata in modo tale che, in uno stato di distensione senza tensione, il carrello si appoggi alla linea dei dieci metri. L'allungamento della corda è di circa il 7% della lunghezza, perciò lo spazio totale di frenata è di un metro. Come nel test precedentemente descritto, dopo i primi cinque metri di accelerazione viene avviata la registrazione; raggiunta la linea di 10 metri viene segnato l'istante temporale, e la registrazione termina quando la corda raggiunge la massima estensione e il carrello si ferma prima di essere spinto indietro. Le registrazioni vengono successivamente scritte in formato *csv* dall'applicazione, catalogate e caricate su un server locale per le fasi successive.

Le registrazioni relative alla frenata sono state effettuate anche con un'autovettura, montando prima il supporto con la ventosa, successivamente po-

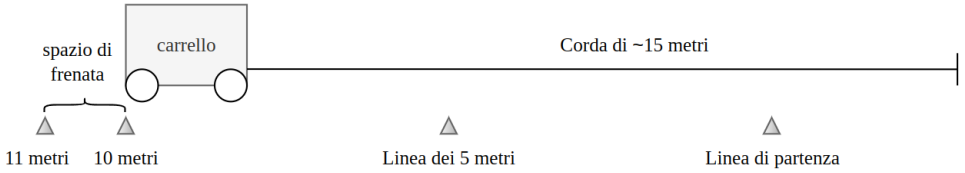


Figura 3.4: Schema per le registrazioni relative alla frenata

sizionandolo all'interno del portaoggetti, ed infine nella tasca dei pantaloni. La velocità mantenuta durante questo esperimento è stata ridotta rispetto al carrello per ragioni di sicurezza. A differenza dell'esperimento precedente, dove la velocità è stata calcolata, qui viene impostata direttamente usando la vettura.

Per ogni combinazione tra veicoli, supporti e fermate, ad eccezione, ovviamente, di schianti con autovettura, sono state effettuate 50 registrazioni, per un totale complessivo di 450.

Preprocessing

Dopo aver raccolto tutte le registrazioni sul server locale, è stata fatta una pulitura delle etichette errate, i file *csv* sono stati organizzati secondo la loro catalogazione e successivamente aggregati in sottocartelle per creare un unico dataset. Questo dataset è stato sottoposto a diverse tecniche di segmentazione per il calcolo delle features. Per valutare l'impatto della segmentazione temporale sulle prestazioni dei modelli abbiamo creato dataset derivati utilizzando diverse finestre temporali. Le finestre temporali sono state scelte linearmente tra 100 e 500 millisecondi, è stata fatta una segmentazione con finestre non sovrapposte. Per le finestre di 500 millisecondi, è stata usata anche una diversa tecnica di segmentazione con finestre sovrapposte, in questo modo abbiamo creato sei dataset derivati. Faremo riferimento al dataset derivato con finestre di 500 millisecondi non sovrapposte come *dataset E1*, mentre il dataset derivato con finestre sovrapposte di 500 millisecondi sarà *dataset E2*.

Come proposto da David Jardim [15], data una sequenza di eventi continui è possibile dividere le sequenze in ciò che possiamo chiamare segmenti temporali o finestre di tempo. Una finestra temporale è un sottoinsieme di una sequenza che corrisponde a una classe specifica. Le nostre finestre tem-

porali possono corrispondere a 3 diverse classi. La segmentazione del dataset è stata fatta creando finestre temporali che appartenessero interamente a una delle classi.

Prima del calcolo delle feature, è stata determinata la magnitudo dei tre sensori, portando il numero totale di variabili iniziali da nove a dodici per ciascun segmento. Le caratteristiche verranno calcolate sulle dodici colonne corrispondenti ai dati di ogni segmento, le features scelte sono: minimo, massimo, media, mediana, primo e terzo quartile, energia e deviazione standard.

A questo punto ogni segmento che era composto da dodici colonne, oltre ai metadati, è stato trasformato in un tipo di dato complesso con più di cento features che rappresentano un istante appartenente a una delle classi. Questi dati possono anche essere interpretati come punti in uno spazio n-dimensionale, dove n è il numero delle features e le coordinate corrispondono ai valori delle features stesse.

Bilanciamento

Dato che i dataset derivati sono sbilanciati verso la classe negativa, è necessario applicare tecniche di bilanciamento per evitare che i modelli abbiano un Bias verso la classe maggioritaria. Per questo, è stata adottata una strategia combinata che impiega SMOTE (Synthetic Minority Over-sampling Technique) per la creazione di dati sintetici in sinergia con Tomek Links che permette di ripulire il dataset dai dati più rumorosi. Nei risultati sperimentali, l'efficacia di questa tecnica di bilanciamento verrà confrontata con due approcci alternativi: un metodo basato sull'undersampling della classe maggioritaria e l'analisi del dataset nella sua forma originaria, ovvero non bilanciato.

SMOTE, introdotto da Nitesh Chawla et al. [7], è una tecnica di oversampling che genera nuovi esempi sintetici per la classe minoritaria. Il metodo opera selezionando campioni vicini nello spazio delle feature, tracciando una linea tra di essi e generando un nuovo punto lungo questo segmento. Tale procedura consente di creare un numero arbitrario di esempi sintetici della classe minoritaria, migliorando così l'equilibrio del dataset. L'efficacia di SMOTE risiede nel fatto che i nuovi campioni sintetici risultano plausibili, in quanto vengono generati in prossimità di esempi esistenti della classe minoritaria, preservando così la distribuzione originale dei dati senza introdurre duplicazioni dirette.

Ivan Tomek nel suo articolo del 1976 [28]. Propose una modifica alle procedure CNN, una regola che individua coppie di esempi, uno per ciascuna classe, che insieme hanno la minima distanza euclidea l'uno dall'altro nello

spazio delle caratteristiche. Questo significa che in un problema di classificazione binaria, una coppia sarebbe composta da un esempio per ciascuna classe e questi sarebbero i vicini più prossimi nell'intero dataset. Queste coppie tra classi differenti sono ora generalmente indicate come "Tomek Links" e sono importanti perché definiscono il confine tra le classi. Applicheremo questo metodo, dopo aver ribilanciato il dataset, per pulirlo dai dati più rumorosi e distinguere meglio le classi.

3.4 Algoritmi di apprendimento automatico

Per la valutazione del nostro dataset sono stati selezionati algoritmi di apprendimento automatico provenienti dalle quattro principali famiglie di classificatori impiegati nei problemi di Human Activity Recognition e Event Detection.

Il primo classificatore adottato è la regressione logistica, appartenente alla famiglia dei modelli lineari generalizzati (Generalized Linear Models, GLM), scelta motivata dalla sua capacità di modellare relazioni lineari tra le variabili e dalla sua interpretabilità. Il secondo classificatore utilizzato è l'albero decisionale (Decision Tree), un modello non parametrico che consente di rappresentare in modo intuitivo le decisioni. Il terzo classificatore selezionato è la Random Forest, un metodo ensemble basato sul bagging, ritenuto spesso la soluzione ottimale sia per il rapporto tra costo computazionale e prestazioni, sia per la sua elevata capacità di generalizzazione. Infine, è stato impiegato un classificatore basato sul gradient boosting, un'altra tecnica ensemble appartenente alla categoria del boosting, ampiamente utilizzata con successo in problemi analoghi al nostro.

3.5 Metriche di valutazione

Definiamo le metriche impiegate per la valutazione delle prestazioni dei modelli di classificazione, espresse in termini di vero positivo (TP), vero negativo (TN), falso positivo (FP) e falso negativo (FN).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F1\ \text{Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Queste metriche sono state selezionate in quanto offrono una prospettiva completa sulla capacità dei modelli di prevedere correttamente le classi, permettendo di analizzare sia la precisione che la sensibilità delle previsioni. Per una valutazione rigorosa delle prestazioni è stato usato Stratified K-Fold Cross Validation con cinque fold, applicando una randomizzazione degli elementi e fissando il random state a 14.

3.6 Test sui dataset

La sperimentazione è stata articolata in più fasi, con l'obiettivo di valutare differenti aspetti dei dataset derivati e dei modelli di classificazione adottati.

In una prima fase è stata esaminata la segmentazione temporale ottimale, compresa tra 100 e 500 millisecondi, al fine di individuare l'intervallo che garantisse, in media, le migliori prestazioni sui modelli. Per questa analisi è stata utilizzata la metrica F1 score, in quanto essa rappresenta la media tra precision e recall, risultando pertanto più indicativa dell'efficacia predittiva in presenza di dataset sbilanciati rispetto all'accuratezza. I risultati, che verranno approfonditi nel capitolo dedicato, hanno evidenziato che il dataset ottenuto mediante una segmentazione di 500 millisecondi offre prestazioni superiori; segmentazioni con intervalli superiori a 500 millisecondi non determinano incrementi significativi e, al contempo, riducono eccessivamente la dimensione del dataset, compromettendone l'efficacia nell'analisi.

Nella fase successiva, per ciascuno dei quattro modelli selezionati, è stata effettuata un'ottimizzazione degli iperparametri mediante Bayesian Optimization insieme a Stratified K-Fold Cross Validation, applicata al dataset da 500 millisecondi opportunamente ribilanciato mediante SMOTE e Tomek Links. Tale procedura ha permesso di individuare quale modello offrisse le migliori performance, sia nel contesto di anomaly detection, in cui si distingue tra andamento regolare e anomalo, sia in quello di anomaly classification, in cui l'andamento anomalo viene ulteriormente differenziato in "schianto" e "frenata".

Infine, in una terza fase della sperimentazione, è stato valutato l'impatto del bilanciamento del dataset sulle prestazioni, adottando il Random Forest come modello di riferimento. La scelta di questo modello è motivata dal basso sforzo richiesto per la ricerca degli iperparametri e dall'assenza della necessità di una rigorosa normalizzazione o scaling delle feature, a differenza di metodi lineari o reti neurali. Generalmente, il Random Forest viene impiegato come baseline per verificare se le modifiche apportate al dataset producano benefici

reali. I risultati ottenuti, attraverso l'utilizzo della Stratified K-Fold Cross Validation, sono stati confrontati sulle metriche sopra descritte. I dataset analizzati, indicati come *E1* ed *E2*, sono stati valutati sia nella loro versione sbilanciata che in quella bilanciata.

Capitolo 4

Implementazione

In questo capitolo si analizza l'implementazione delle diverse fasi descritte nel flusso operativo. In particolare, verranno illustrati in dettaglio lo sviluppo dell'applicazione Android, il processo di creazione ed elaborazione del dataset, il calcolo delle feature, il bilanciamento dei nuovi dataset generati e l'implementazione dei test. Per questo sono state adottate tecnologie mobile avanzate, tecniche di ottimizzazione e strumenti per l'analisi dei dati.

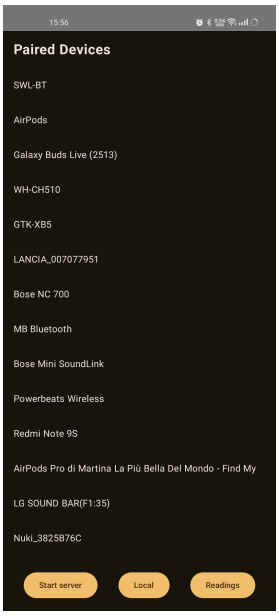
4.1 Applicazione Android

Per la costruzione del dataset è stata sviluppata un'applicazione Android dedicata, progettata per la raccolta simultanea dei dati provenienti dai sensori. L'applicazione [10] effettua campionamenti a una frequenza di 50 Hz (ossia una lettura ogni 20 millisecondi), registrando le misure simultaneamente con il relativo timestamp e l'indicatore di impatto.

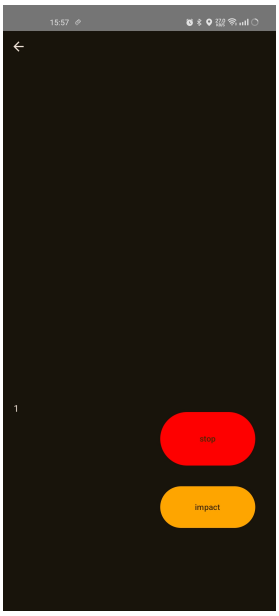
L'applicazione è stata sviluppata in Kotlin, il linguaggio di programmazione nativamente supportato da Android. L'impiego di Kotlin ha permesso di sfruttare pienamente le funzionalità della piattaforma, l'integrazione con Android Studio e l'accesso alle API moderne hanno facilitato l'implementazione di pattern di programmazione asincrona e reattiva, migliorando la gestione degli eventi e ottimizzando le prestazioni dell'applicazione.

Dal punto di vista hardware, il sistema sfrutta i sensori presenti su dispositivi di ogni fascia, in particolare quelli basati sulla tecnologia Micro-Electro-Mechanical Systems (MEMS). Questi sensori, che integrano strutture meccaniche in microchip, sono caratterizzati da un basso consumo energetico e da prestazioni elevate, garantendo una rilevazione accurata dei movimenti. Il fatto che siano ormai standard in tutti i dispositivi mobili moderni li rende

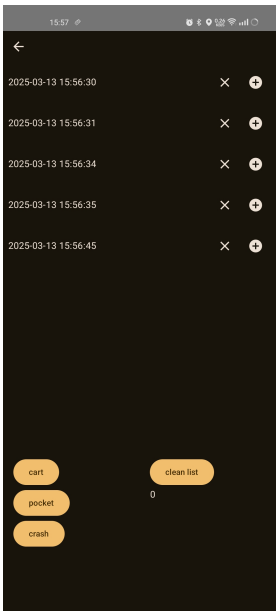
particolarmente adatti per applicazioni di raccolta dati in ambito Human Activity Recognition ed Event Detection. Il dispositivo scelto è un telefono cellulare di fascia bassa con Android 12, una CPU Octa-core 2.32GHz con 6 GB di RAM, il sensore montato è il modello *BMI160*. Questo telefono è stato scelto perché può rappresentare le performance medie dei dispositivi attualmente sul mercato.



(a) Screenshot della home dell'Applicazione android, con la lista di dispositivi collegabili



(b) Screenshot della facciata di controllo delle registrazioni



(c) Screenshot delle registrazioni effettuate dal dispositivo sul carrello

4.2 Dataset

Il dataset grezzo è composto dai valori dei sensori triassiali: accelerometro, giroscopio, magnetometro. Prima delle feature viene calcolata la magnitudo dei tre sensori per ogni istante registrato, la magnitudo di un sensore triassiale viene calcolata come:

$$|\text{magnitude}| = \sqrt{x^2 + y^2 + z^2} \tag{4.1}$$

dove x, y, z rappresenta i valori del sensore sui rispettivi assi. La magnitudo rappresenta l'intensità complessiva della grandezza misurata (ad esempio

accelerazione, campo magnetico, vibrazione) indipendentemente dalla direzione, quindi è spesso usata per il rilevamento di anomalie o impatti in sistemi di monitoraggio. A questi dodici valori è associato un Timestamp, un identificatore di schianto e le caratteristiche della registrazione. Nella tabella 4.1 si possono vedere tutti i setup per le registrazioni.

Tabella 4.1: Setup per le registrazioni

Veicolo	Supporto	Fermata
carrello	braccio con ventosa	schianto
carrello	contenitore, porta oggetti	schianto
carrello	tasca dei pantaloni	schianto
carrello	braccio con ventosa	frenata
carrello	contenitore, porta oggetti	frenata
carrello	tasca dei pantaloni	frenata
macchina	braccio con ventosa	frenata
macchina	contenitore, porta oggetti	frenata
macchina	tasca dei pantaloni	frenata

Le registrazioni sono state poi unite per formare un unico dataset diviso in cartelle rappresentative di ogni specifico setup.

4.2.1 Elaborazione del dataset

Come detto nel capitolo precedente, le finestre scelte sono: 100, 200, 300, 400, 500 millisecondi; con 500 millisecondi sono state fatte due diverse elaborazioni per vedere come il calcolo di queste finestre impatti l'addestramento dei modelli. Nei risultati le andremo a testare per vedere le differenze.

Nella prima elaborazione, i dati sono stati suddivisi in finestre temporali di 500 millisecondi, senza sovrapposizione tra di esse. Questo approccio permette di analizzare le informazioni in intervalli ben distinti, facilitando l'interpretazione e la gestione dei dati. Tuttavia, potrebbe comportare una perdita di dettagli nelle transizioni tra le finestre.

Inoltre, considerando che il nostro dispositivo mobile registra i valori dei sensori con una frequenza di 20 millisecondi, la suddivisione in finestre da 500 millisecondi implica una riduzione della cardinalità del dataset elaborato di un fattore 25 rispetto al dataset grezzo. Questo comporta un minor numero

di istanze disponibili per l'analisi. Faremo riferimento a questa elaborazione del dataset come: *dataset E1* .

Nella seconda elaborazione, invece, è stato adottato un approccio a finestre mobili (sliding windows), in cui le finestre di 500 millisecondi vengono fatte scorrere lungo il dataset con un passo di 100 millisecondi. Questo metodo consente di catturare in modo più continuo le variazioni nei dati e di preservare maggiori informazioni temporali, riducendo il rischio di perdita di pattern rilevanti.

Rispetto alla prima elaborazione, questo approccio comporta una minore perdita di dati, con una riduzione della cardinalità del dataset di circa 4 volte rispetto al dataset grezzo. Di conseguenza, otteniamo un numero significativamente maggiore di istanze disponibili per l'addestramento del modello, migliorando il potenziale di generalizzazione. Faremo riferimento a questa elaborazione del dataset come: *dataset E2*.

4.2.2 Calcolo delle features

Come specificato nel capitolo precedente, le caratteristiche verranno calcolate sulle dodici colonne corrispondenti ai dati di ogni segmento. Le features scelte sono : la media, il minimo, il massimo, la mediana che è particolarmente utile in dati distribuiti in modo asimmetrico, dove la media potrebbe essere distorta da valori estremi.

Altre features sono: 25° percentile (o primo quartile, Q1) è una misura di posizione statistica che rappresenta il valore sotto il quale si trova il 25% dei dati in un insieme ordinato. Dato un insieme $\{x_1, x_2, ..., x_n\}$ ordinato in modo non decrescente: $x_1 \leq x_2 \leq ... \leq x_n$ il primo quartile è definito come: $\text{Q1} = \frac{1}{4}(n-1)$, 75° percentile (o terzo quartile, Q3) viene calcolato come $\text{Q3} = \frac{3}{4}(n-1)$, usato insieme al 25° percentile per calcolare l'Interquartile Range (IQR) definito come: $\text{IQR} = \text{Q3} - \text{Q1}$, L'interquartile range misura la dispersione dei dati ignorando gli estremi e concentrandosi sulla variabilità centrale. Viene spesso usato per identificare valori anomali.

In fine calcoliamo l'energia come: $E = \sum_{i=1}^{n} x_i^2$, e la deviazione standard, o scarto quadratico medio, un indice di dispersione statistica, vale a dire un indicatore usato per fornire una stima sintetica della variabilità di una popolazione di dati o di una variabile casuale. La deviazione standard per un insieme finito di n valori $X = \{x_1, x_2, \ldots, x_n\}$ è definita come:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

dove:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

è la **media aritmetica** dell'insieme.

L'implementazione in Python si è avvalsa della libreria pandas, attraverso l'uso di metodi integrati, è stato possibile calcolare in maniera diretta e ottimizzata le misure di posizione e di dispersione, nonché i percentili e l'interquartile range.

4.2.3 Bilanciamento

SMOTE per il bilanciamento del dataset è stato implementato in questo modo:

Dato X l'insieme della classe minoritaria, N la percentuale di oversampling desiderata e k il numero di vicini da considerare, questi sono i passaggi dell'algoritmo:

for *ogni* x_i *elemento della classe minoritaria* **do**
| trova i k punti più vicini utilizzando la distanza Euclidea.
end
Determina il numero T numero di campioni da generare, in base a N
for *ogni campione da generare* **do**
| seleziono casualmente un $x_{neighbor}$;
| calcolo $x_{new} = x + \lambda * (x_{neighbor} - x)$
end

dove:
$x_{neighbor}$ è un punto vicino a x
λ è un valore casuale tra 0 e 1
x_{new} sono i nuovi punti sintetici.

Insieme a SMOTE è stato implementato anche Tomek Links. La procedura per individuare i Tomek Links può essere utilizzata per localizzare tutti i vicini più prossimi di classi differenti. Se gli esempi della classe minoritaria vengono mantenuti costanti, la procedura può essere impiegata per trovare tutti quegli esempi della classe maggioritaria che sono più vicini alla classe minoritaria, per poi essere rimossi.

4.3 Algoritmi di apprendimento

Tutti i test sui dataset e sui modelli sono stati eseguiti in ambiente Python (versione 3.11.11), avvalendosi delle librerie pandas, numpy, csv, os e

matplotlib.

L'ottimizzazione degli iperparametri è stata effettuata mediante Bayesian Optimization utilizzando BayesSearchCV, implementato nella libreria scikit-optimize. Tale procedura si basa su un algoritmo di ottimizzazione bayesiana sequenziale (Sequential Model-Based Optimization, SMBO) e, per determinare il prossimo punto da valutare, viene impiegata la funzione di acquisizione Expected Improvement (EI).

Per valutare le prestazioni dei modelli è stata adottata la tecnica di Stratified K-Fold Cross Validation, implementata tramite il metodo StratifiedK-Fold della libreria sklearn.

Capitolo 5

Risultati

In questo capitolo verranno esaminati gli esiti ottenuti dall'applicazione delle metodologie e degli algoritmi descritti nei capitoli precedenti. L'obiettivo primario è quello di fornire una chiara comprensione delle performance raggiunte, evidenziando i punti di forza e le eventuali limitazioni delle soluzioni implementate.

Inizialmente, verrà condotta un'analisi approfondita del dataset impiegato, al fine di contestualizzare i risultati ottenuti e fornire elementi utili alla loro interpretazione. Successivamente, si procederà con la presentazione della sperimentazione sui modelli di apprendimento automatico. Quest'ultima fase è stata strutturata in tre sezioni distinte, ciascuna dedicata a specifici aspetti della valutazione e confronto delle diverse tecniche adottate.

5.1 Analisi del Dataset

Come verrà illustrato nelle sezioni successive, l'algoritmo di apprendimento che ha ottenuto i migliori risultati è Random Forest. Pertanto, prima di presentare i risultati dei test, si procederà con un'analisi approfondita del dataset.

Utilizzando Random Forest, è stata analizzata l'importanza delle diverse feature impiegate per la classificazione. Come evidenziato in Figura 5.1, l'accelerometro riveste un ruolo fondamentale nel processo classificatorio, in particolare la magnitudo (4.1) calcolata sull'accelerometro. Questo risultato è dovuto al fatto che la magnitudo è sensibile alle oscillazioni lungo tutti e tre gli assi del sensore. Le caratteristiche più rilevanti, calcolate sulla magnitudo dell'accelerometro, risultano essere le seguenti:

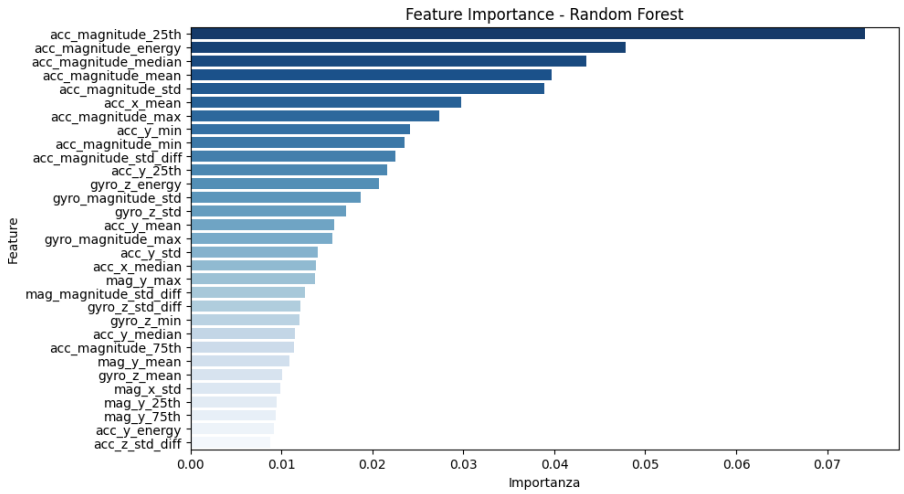


Figura 5.1: Il grafico rappresenta le features piu importanti sopra la media. L'insieme dei valori di importanza è stato normalizzato tra 0 e 1. La feature piu rilevante, il venticinquesimo percentile sulla magnitudo, ha una importanza del $7,5\%$ circa

- *acc_magnitude_25th*: venticinquesimo percentile sulla magnitudo
- *acc_magnitude_energy*: somma dei quadrati
- *acc_magnitude_median*: mediana
- *acc_magnitude_mean*: media
- *acc_magnitude_std*: deviazione standard
- *acc_magnitude_max*: massimo

L'importanza delle features è mediamente distribuita, non è presente una caratteristica completamente dominante. Le dieci features più rilevanti insieme hanno un'importanza del 35% circa.

Una volta identificate le caratteristiche più rilevanti, si procede con il calcolo della loro correlazione. La matrice di correlazione, riportata in Figura 5.2, fornisce una panoramica delle relazioni esistenti tra le variabili selezionate. Il valore medio della correlazione è pari a 0.4119. Questo valore suggerisce che esiste una certa dipendenza tra alcune feature, ma non abbastanza elevata da generare problemi di collinearità, il che è positivo per il modello.

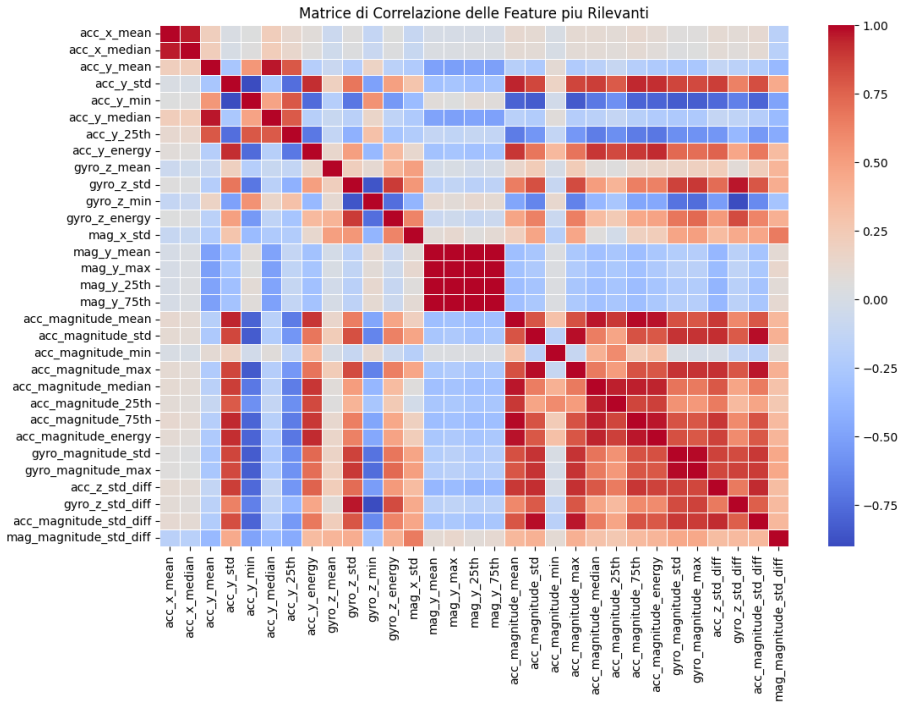


Figura 5.2: Matrice di correlazione tra le features più rilevanti

5.2 Experimental setup

Durante tutti i test per valutare le prestazioni useremo Stratified K-Fold Cross Validation con cinque fold, applicando una randomizzazione degli elementi e fissando il random state a 14. Sui quattro modelli scelti faremo tuning degli iperparametri usando Bayesian Optimization insieme a Stratified K-Fold Cross Validation. Gli spazi di ricerca per ogni modello sono descritti nelle tabelle: 5.1, 5.2, 5.3, 5.4. La funzione di scoring scelta per l'ottimizzazione è F1 score con average: macro, utile per avere una visione globale e non lasciare "nascoste" le performance della classe minoritaria del dataset sbilanciato.

5.3 Confronto tra dataset derivati

Nella prima fase abbiamo testato quale segmentazione fornisse mediamente i migliori risultati. Abbiamo quindi testato i quattro modelli scelti sui cinque dataset derivati, da 100 a 500 millisecondi.

Tabella 5.1: Spazio dei parametri per Logistic Regression

Parametro	Tipo	Range/Valori
C	Real	$[10^{-6}, 10^{3}, Prior : log - uniform]$
penalty	Categorical	{'l1','l2','elasticnet', None}
l1_ratio	Real	[0.0, 1.0]
solver	Categorical	{'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}
max_iter	Integer	[500, 1000]

Tabella 5.2: Spazio dei parametri per Decision Tree

Parametro	Tipo	Range/Valori
max_depth	Integer	[1, 50]
min_samples_split	Integer	[2, 20]
min_samples_leaf	Integer	[1, 20]
max_features	Categorical	{'sqrt', 'log2', None}

5.3.1 Anomaly detection

In un primo test, è stato fatto l'addestramento degli algoritmi per Anomaly detection utilizzando Stratified K-Fold Cross Validation, per la distinzione tra andamento regolare del veicolo e un comportamento anomalo. In figura 5.3 vediamo l'F1 score medio sulle cinque istanze del dataset.

I risultati ottenuti indicano che l'aumento nella durata della finestra temporale comporta un miglioramento delle performance dei modelli di apprendimento. Analizzando le caratteristiche più rilevanti, si osserva che un intervallo più lungo contribuisce a ridurre l'impatto del rumore sulle feature estratte.

La riduzione del rumore influisce positivamente sulla separabilità delle classi, facilitando il compito dei modelli di apprendimento automatico nel distinguere le diverse categorie. Questo aspetto risulta cruciale per il miglioramento dell'accuratezza e della capacità predittiva degli algoritmi.

Tra i modelli analizzati, gli approcci basati su ensemble learning, in particolare *Random Forest*, hanno dimostrato elevate prestazioni sia nella classificazione binaria che in quella multiclasse, confermandosi come una soluzione efficace per il problema in esame.

Tabella 5.3: Spazio dei parametri per Random Forest

Parametro	Tipo	Range/Valori
n_estimators	Integer	[50, 200]
max_depth	Integer	[1, 50]
min_samples_split	Integer	[2, 20]
min_samples_leaf	Integer	[1, 20]
max_features	Categorical	{'sqrt', 'log2', None}

Tabella 5.4: Spazio dei parametri per Gradient Boosting

Parametro	Tipo	Range/Valori
n_estimators	Integer	[5, 200]
learning_rate	Real	$[0.01, 1.0, Prior : log - uniform]$
max_depth	Integer	[1, 20]
min_samples_split	Integer	[2, 20]
min_samples_leaf	Integer	[1, 20]
subsample	Real	$[0.5, 1.0, Prior : uniform]$
max_features	Categorical	{'sqrt', 'log2', None}

5.3.2 Anomaly classification

Nel secondo test,la classe "andamento anomalo" è stata divisa tra: "brake" che identifica una frenata brusca, e "crash" che identifica lo schianto del veicolo contro un oggetto stazionario. Si è dunque ripetuta la stessa procedura sperimentale, con la differenza che ora distinguiamo tra le classi, addestrando gli stessi modelli sui cinque dataset derivati. Anche in questo caso i risultati indicano che una finestra temporale maggiore diminuisce il rumore, migliorando la separabilità delle classi.

5.4 Confronto tra i modelli

Nella seconda fase lo scopo è capire, tra i modelli presi in esame, quale performi meglio utilizzando il *dataset E2*, per questo faremo un tuning de-

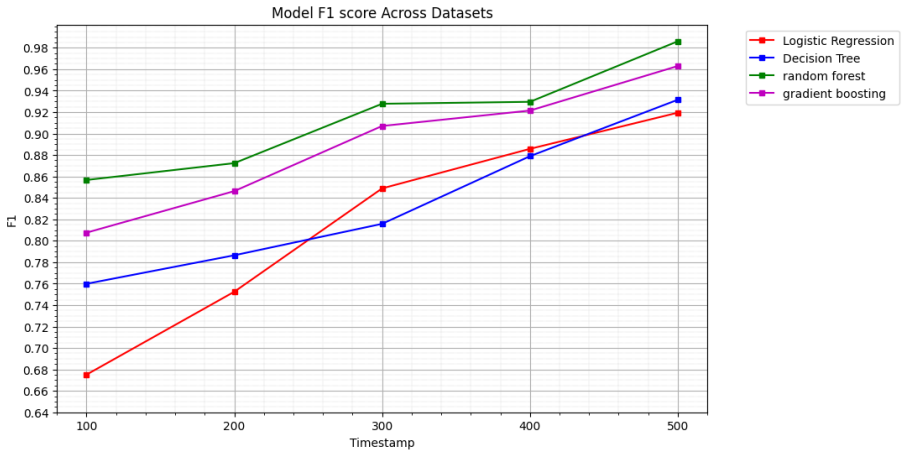


Figura 5.3: F1 score dei modelli su timeframe da 100 a 500

gli iperparametri usando Bayesian Optimization insieme a Stratified K-Fold Cross Validation.

5.4.1 Anomaly detection

Per ogni modello abbiamo fatto un tuning degli iperparametri usando Bayesian Optimization insieme a Stratified K-Fold Cross Validation per Anomaly detection. Nel grafico 5.4 abbiamo riportato i risultati ottenuti dal miglior set di iperparametri per ogni modello. Come possiamo osservare, sia Random Forest che Gradient Boosting mostrano prestazioni complessivamente superiori rispetto agli altri due modelli su tutte le metriche, mentre la Logistic Regression, con un valore basso nella Precision, indica una maggiore frequenza di falsi positivi.

5.4.2 Anomaly classification

Anche per questo esperimento abbiamo fatto un tuning degli iperparametri usando Bayesian Optimization insieme a Stratified K-Fold Cross Validation, per Anomaly classification. Possiamo vedere i risultati nel grafico 5.5, per i valori di precision, recall, e F1 score è stato usato average='macro', che calcola la media non ponderata delle metriche per ogni etichetta. Questo non tiene conto dello squilibrio delle etichette; per questo, in un dataset sbilanciato come il nostro, è più significativa di 'weighted'. Possiamo osservare una importante performance della Regressione Logistica; questo può signifi-

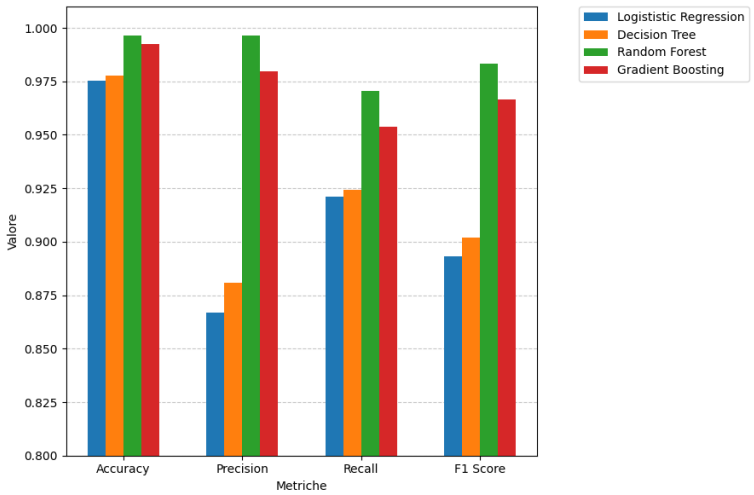


Figura 5.4: Grafico a barre dei risultati per i modelli ottimizzati con dataset E2 su anomaly detecton

care che in un problema binario abbiamo una classe "mista" in cui campioni eterogenei vengono raggruppati insieme, mentre passando a un problema a tre classi, questi campioni sono suddivisi in categorie più omogenee.

5.5 Impatto del Bilanciamento dei Dati

Nella terza fase vogliamo valutare quanto il bilanciamento del dataset sia importante per il miglioramento delle performance di classificazione. A tal fine useremo il Random Forest con i dataset derivati *E1* ed *E2*, descritti nella sezione 4.2.1. Gli iperparametri del RF sono stati mantenuti invariati per tutti i test.

5.5.1 Anomaly detection

per confrontare le metriche di performance tra le versioni bilanciate e non bilanciate dei dataset abbiamo scelto di usare il Random Forest. Il primo motivo per cui è stato scelto è che, come visto nel capitolo Stato dell'Arte, viene generalmente indicato come uno dei modelli più performanti [9], il secondo motivo è che il RF ha pochi iperparametri da ottimizzare

Il grafico 5.6 presenta i risultati su Accuracy, Precision, Recall, F1 Score, relativi al classificatore Random Forest addestrato per fare Anomaly detec-

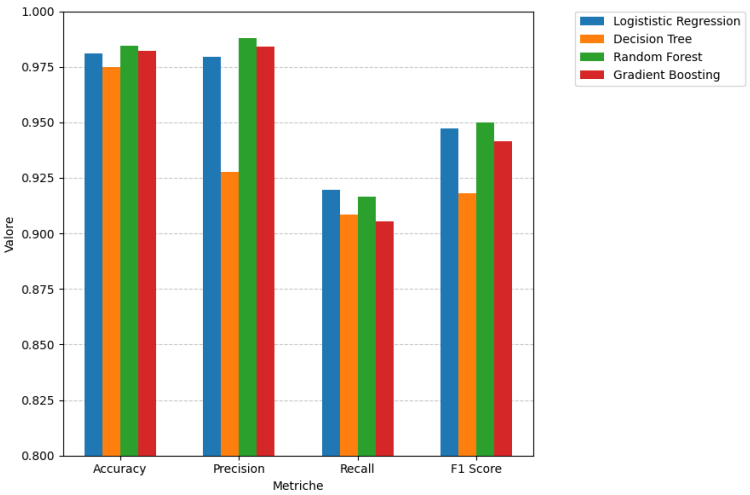


Figura 5.5: Grafico a barre dei risultati per i modelli ottimizzati con dataset E2 su anomaly classification

tion, sulle diverse elaborazioni del dataset. Nello specifico, sono state testate le configurazioni: *dataset E1* sbilanciato, *dataset E1* bilanciato con SMOTE e Tomek links, *dataset E2* sbilanciato, *dataset E2* bilanciato con undersampling della classe maggioritaria, *dataset E2* bilanciato con SMOTE e Tomek links.

L'analisi raffigurata nel grafico 5.6 evidenzia come la configurazione ottimale sia stata ottenuta combinando sliding windows con il ribilanciamento del dataset tramite SMOTE e una pulizia mediante Tomek Links. Sotto sono riportate le confusion matrix ottenute (5.7 5.8 5.9 5.10 5.11).

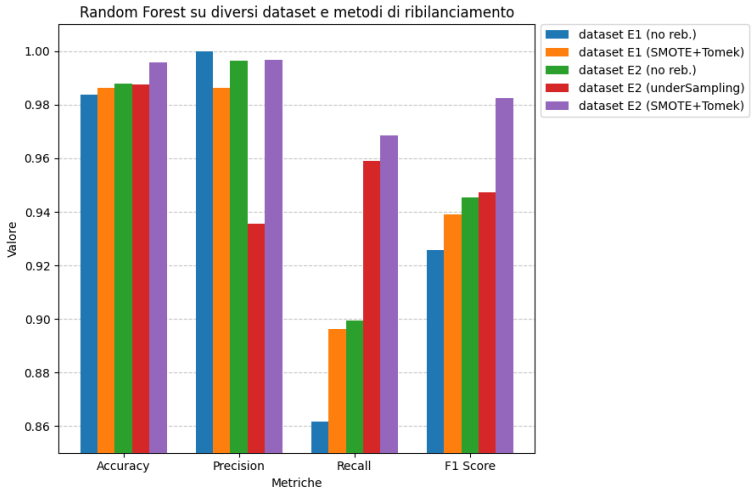


Figura 5.6: Confronto tra metriche e tecniche di ribilanciamento dei dataset

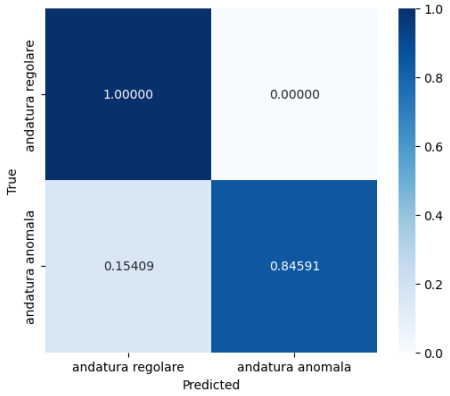


Figura 5.7: Confusion matrix ottenuta dall'addestramento del Random Forest sul dataset E1 sbilanciato, si vede come lo sbilanciamento influisce negativamente sulle performance per la classe meno rappresentata

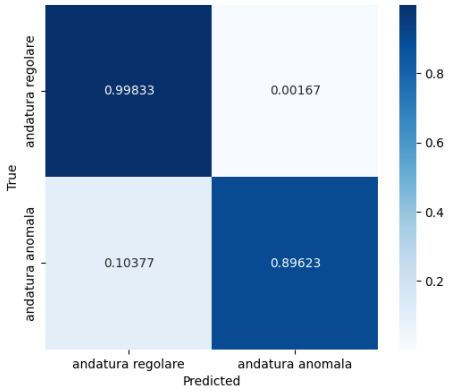


Figura 5.8: Confusion matrix ottenuta dall'addestramento del Random Forest sul dataset E1 bilanciato con SMOTE e Tomek links

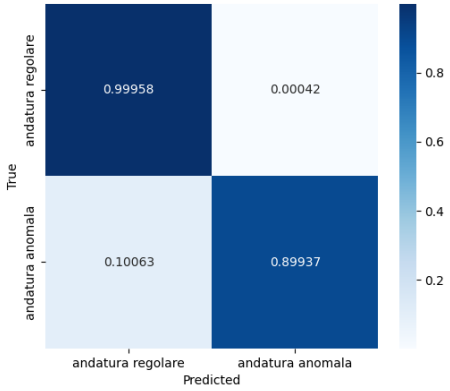


Figura 5.9: Confusion matrix ottenuta dall'addestramento del Random Forest su dataset E2 sbilanciato, anche in questo caso si vede come lo sbilanciamento influisce negativamente sulle performance per la classe meno rappresentata

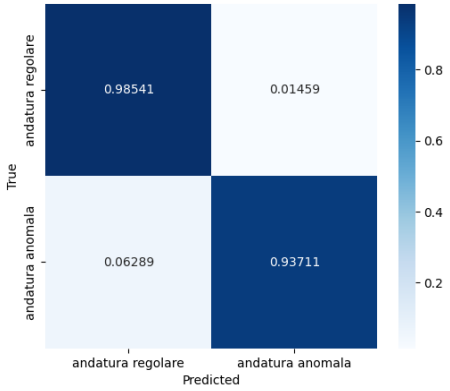


Figura 5.10: Confusion matrix ottenuta dall'addestramento del Random Forest su dataset E2 bilanciato con undersampling della classe maggioritaria, si osserva un miglior equilibrio nelle predizioni e una maggiore capacità di riconoscimento delle istanze appartenenti alla classe minoritaria

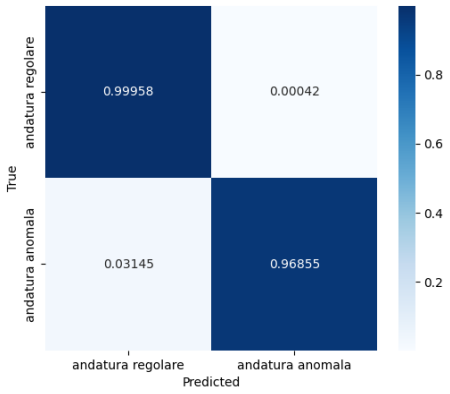


Figura 5.11: Confusion matrix ottenuta dall'addestramento del Random Forest su dataset E2 bilanciato con SMOTE e Tomek links, si vede come la sintetizzazione di nuovi dati per classe meno rappresentata a migliorare la capacità del modello di riconoscere correttamente le istanze di tale classe, riducendo in modo significativo l'incidenza dei falsi negativi e garantendo un equilibrio più marcato nelle predizioni.

Conclusioni

Il presente lavoro si proponeva di creare un dataset finalizzato al riconoscimento degli impatti su veicoli in movimento, adottando un approccio innovativo che, data l'impraticabilità di sperimentare direttamente su veicoli reali, ha utilizzato un carrello come surrogato. In questo modo, si è ipotizzato che la rilevazione di eventi su un oggetto di dimensioni inferiori possa essere tradotta nel contesto reale.

L'analisi condotta sul dataset ha permesso di verificare, in via preliminare, la validità dell'ipotesi formulata. La segmentazione temporale dei dati, effettuata in finestre ottimali, insieme a tecniche di feature engineering, ha garantito una migliore cattura delle variazioni del segnale, riducendo l'effetto del rumore. Inoltre, l'applicazione di tecniche di bilanciamento (come SMOTE associato a Tomek Links) ha migliorato l'efficacia dei modelli di apprendimento automatico, in particolare per quanto riguarda l'identificazione degli eventi anomali.

Tra i vari algoritmi testati, le soluzioni basate su ensemble, quali Random Forest e Gradient Boosting, hanno mostrato prestazioni generalmente superiori. Nel caso specifico dell'Anomaly classification con un'opportuna ottimizzazione degli iperparametri, la Regressione Logistica ha dimostrato di essere una soluzione valida, soprattutto considerando i tempi di addestramento. Questi risultati, pur essendo da considerarsi iniziali, forniscono una solida base per ulteriori sviluppi e perfezionamenti.

Limiti e prospettive future

Nonostante i risultati incoraggianti, lo studio presenta alcune limitazioni. La sperimentazione è stata condotta in ambiente simulato e controllato, il che potrebbe non riprodurre appieno la complessità dei contesti reali. Per questo motivo è necessario proseguire il lavoro di sperimentazione includendo dati registrati in casi reali, provando a collaborare con enti specializzati nella raccolta di dati in questo contesto come le agenzie di assicurazione o gli enti di gestione degli autodromi. In sviluppi futuri si possono testare altri

modelli ensemble, con particolare attenzione al costo computazionale e con le adeguate ottimizzazioni, capire se è possibile implementare, in un'applicazione mobile, uno dei modelli di apprendimento automatico testati, per creare un'applicazione di assistenza in casi reali di incidente.

Considerazioni finali

Questo studio rappresenta un primo passo verso l'implementazione di sistemi intelligenti per la sicurezza stradale. La creazione di un dataset dedicato e la sperimentazione con diversi modelli di machine learning hanno dimostrato la fattibilità dell'approccio proposto, aprendo la strada a ulteriori ricerche e applicazioni. L'adozione del carrello come modello sperimentale ha permesso di superare le difficoltà inerenti alla raccolta dati su veicoli reali, fornendo al contempo una solida base per future integrazioni e applicazioni pratiche. In definitiva, questo lavoro non solo contribuisce all'avanzamento della ricerca in ambito di rilevamento degli impatti, ma costituisce anche un valido punto di partenza per lo sviluppo di sistemi di monitoraggio e sicurezza sempre più sofisticati nel settore dei trasporti.

Bibliografia

[1] Yahaya Idris Abubakar et al. “A Systematic Review of Rare Events Detection Across Modalities Using Machine Learning and Deep Learning”. In: *IEEE Access* 12 (2024), pp. 47091–47109. DOI: 10.1109/ACCESS.2024.3382140. URL: https://doi.org/10.1109/ACCESS.2024.3382140.

[2] Nahla Ben Amor, Salem Benferhat e Zied Elouedi. “Naive Bayes vs decision trees in intrusion detection systems”. In: *Proceedings of the 2004 ACM Symposium on Applied Computing*. SAC ’04. Nicosia, Cyprus: Association for Computing Machinery, 2004, pp. 420–424. ISBN: 1581138121. DOI: 10.1145/967900.967989. URL: https://doi.org/10.1145/967900.967989.

[3] Burak Ayyorgun. “Developing Ultra-Lite ML Models for Crash Detection in Noisy Environments”. In: (feb. 2025). DOI: 10.36227/techrxiv.173933098.86669621/v1. URL: http://dx.doi.org/10.36227/techrxiv.173933098.86669621/v1.

[4] B. Barshan e H.F. Durrant-Whyte. “Orientation estimate for mobile robots using gyroscopic information”. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’94)*. Vol. 3. 1994, 1867–1874 vol.3. DOI: 10.1109/IROS.1994.407605.

[5] Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–32.

[6] Rich Caruana e Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 161–168.

[7] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (giu. 2002), pp. 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953. URL: http://dx.doi.org/10.1613/jair.953.

[8] Pablo Duboue. *The art of feature engineering: essentials for machine learning*. Cambridge University Press, 2020.

[9] Manuel Fernández-Delgado et al. "Do we need hundreds of classifiers to solve real world classification problems?" In: *The journal of machine learning research* 15.1 (2014), pp. 3133–3181.

[10] Simone Folli. *Bt-data-collector Android application for data collection*. 2024. URL: https://github.com/Follisim/Bt-data-collector.

[11] Yoav Freund, Robert Schapire e Naoki Abe. "A short introduction to boosting". In: *Journal-Japanese Society For Artificial Intelligence* 14.771-780 (1999), p. 1612.

[12] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[13] Jerome H Friedman. "Stochastic gradient boosting". In: *Computational statistics & data analysis* 38.4 (2002), pp. 367–378.

[14] Syahirah Jamian et al. "Human Activity and Posture Classification using Smartphone Sensors and Matlab Mobile". In: *IEEE International Instrumentation and Measurement Technology Conference, I2MTC 2022, Ottawa, ON, Canada, May 16-19, 2022*. 2022, pp. 1–6. DOI: 10.1109/I2MTC48687.2022.9806551. URL: https://doi.org/10.1109/I2MTC48687.2022.9806551.

[15] David Jardim, Luís Nunes e Miguel Dias. "Human Activity Recognition from automatically labeled data in RGB-D videos". In: *2016 8th Computer Science and Electronic Engineering (CEEC)*. 2016, pp. 89–94. DOI: 10.1109/CEEC.2016.7835894.

[16] Abdul Rehman Javed et al. "Analyzing the Effectiveness and Contribution of Each Axis of Tri-Axial Accelerometer Sensor for Accurate Activity Recognition". In: *Sensors* 20.8 (2020), p. 2216. DOI: 10.3390/S20082216. URL: https://doi.org/10.3390/s20082216.

[17] Bhoram Lee et al. "Orientation estimation in mobile virtual environments with inertial sensors". In: *IEEE Transactions on Consumer Electronics* 57.2 (2011), pp. 802–810. DOI: 10.1109/TCE.2011.5955225.

[18] Ana C Lorena et al. "Comparing machine learning classifiers in potential distribution modelling". In: *Expert Systems with Applications* 38.5 (2011), pp. 5268–5275.

[19] "Machine learning algorithms for event detection: A special issue of machine learning." In: *Machine learning*. 79.3 (20100601). ISSN: 0885-6125.

[20] Dragos Margineantu, Weng-Keen Wong e Denver Dash. "Machine learning algorithms for event detection." In: *Machine Learning* 79.3 (2010).

[21] Nurul Retno Nurwulan e Gjergji Selamaj. "Random Forest for Human Daily Activity Recognition". In: *Journal of Physics: Conference Series* 1655.1 (ott. 2020), p. 012087. DOI: 10.1088/1742-6596/1655/1/012087. URL: https://dx.doi.org/10.1088/1742-6596/1655/1/012087.

[22] Saifur Rahman et al. "Performance analysis of boosting classifiers in recognizing activities of daily living". In: *International journal of environmental research and public health* 17.3 (2020), p. 1082.

[23] Duraisamy Ramyachitra e Parasuraman Manikandan. "Imbalanced dataset classification and solutions: a review". In: *International Journal of Computing and Business Research (IJCBR)* 5.4 (2014), pp. 1–29.

[24] B Rani, R Praveen Sam e Govardhan Reddy Kamatam. "A review on vehicle tracking and accident detection system using accelerometer". In: *International Journal of Applied Engineering Research* 13.11 (2018), pp. 9215–9217.

[25] L. Rokach e O. Maimon. "Top-down induction of decision trees classifiers - a survey". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 35.4 (2005), pp. 476–487. DOI: 10.1109/TSMCC.2004.843247.

[26] AY Shdefat, AA Halimeh e HC Kim. "Human activities recognition via smartphones using supervised machine learning classifiers". In: *Prim Health Care* 8.289 (2018), pp. 2167–1079. URL: https://www.researchgate.net/profile/Ahmed-Shdefat/publication/324355530_Human_Activities_Recognition_Via_Smartphones_Using_Supervised_Machine_Learning_Classifiers/links/5b27a27aaca2723fbef03e10/Human-Activities-Recognition-Via-Smartphones-Using-Supervised-Machine-Learning-Classifiers.pdf.

[27] Amanpreet Singh, Narina Thakur e Aakanksha Sharma. "A review of supervised machine learning algorithms". In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. Mar. 2016, pp. 1310–1315.

[28] Ivan Tomek. "Two Modifications of CNN". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-6.11 (1976), pp. 769–772. DOI: 10.1109/TSMC.1976.4309452.

[29] Daniela Xhemali, Chris J Hinde e Roger Stone. "Naive bayes vs. decision trees vs. neural networks in the classification of training web pages". In: (2009).

[30] Zunash Zaki et al. "Logistic regression based human activities recognition". In: *J. Mech. Contin. Math. Sci* 15.4 (2020), pp. 228–246.