Titre: Title:	Analysis of User-Based Insurance Telematics data via Smartphone sensors in Driver Behavior and Safety
Auteur: Author:	Mohammad Ghavidel Gholamhosseinzadeh
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ghavidel Gholamhosseinzadeh, M. (2024). Analysis of User-Based Insurance Telematics data via Smartphone sensors in Driver Behavior and Safety [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/58344/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/58344/
Directeurs de recherche: Advisors:	Nicolas Saunier, & Aurélie Labbe
Programme: Program:	Génie civil

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Analysis of User-Based Insurance Telematics data via Smartphone sensors in Driver Behavior and Safety

MOHAMMAD GHAVIDEL GHOLAMHOSSEINZADEH

Département des génies civil, géologique et des mines

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie civil

Mai 2024

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé:

Analysis of User-Based Insurance Telematics data via Smartphone sensors in Driver Behavior and Safety

présenté par **Mohammad GHAVIDEL GHOLAMHOSSEINZADEH**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Martin TRÉPANIER, président
Nicolas SAUNIER, membre et directeur de recherche
Aurélie LABBE , membre et codirecteur de recherche
Bruno AGARD, membre

DEDICATION

Femmes, vie, liberté

Women, life, freedom.

ACKNOWLEDGEMENTS

First of all, I want to thank my supervisors, who supported me through the past two years. I really appreciate you accepting me as their student. I have learned a lot through my education. Thank you, Professor Nicolas Saunier and Professor Aurelie Labbe for being more meticulous in my work and understanding each step of the way.

Also, I want to say thank you to Mitacs for providing funding and financial support during my education, which facilitated the path. I have special thanks to Dr. Joshua Stipancic for his help, belief in my work, and guidance.

I came to Canada and Montreal from a country far away. I would like to thank my family, my mom, and dad, for their emotional and financial support during these 2 years. Also, I would like to thank my colleagues Bita Farokhian, Hamed Malekzadeh, and Xinyu Chen.

RÉSUMÉ

Les accidents de la route sont la troisième cause de mortalité au Canada, posant d'importants défis financiers et sanitaires pour les individus et les compagnies d'assurance. En réponse, l'industrie de l'assurance automobile a vu une croissance rapide de l'Assurance Basée sur l'Utilisation (UBI), un segment qui utilise les données de comportement de conduite pour des tarifs d'assurance personnalisés.

Historiquement, la recherche sur l'UBI s'est concentrée sur l'analyse de la variabilité de l'accélération, de la vitesse et des angles des véhicules à l'aide de capteurs embarqués et de données de positionnement global. Cependant, l'avènement des smartphones a révolutionné la collecte de données dans la sécurité routière, réduisant considérablement les coûts grâce à leur disponibilité généralisée et à leurs capacités de capteurs avancées. Malgré leur potentiel, les défis liés à la qualité des données, à l'incertitude et à la variabilité entravent leur adoption complète.

Cette recherche se concentre sur l'exploitation des données cinématiques issues des programmes UBI à Montréal et Ottawa, en utilisant des capteurs de smartphones tels que les accéléromètres, GNSS, magnétomètres et gyroscopes. Notre méthodologie s'inspire des algorithmes de l’Android pour l'estimation de l'azimut, adaptée pour le jeu de données UBI. Nous avons mené des expériences contrôlées avec un iPhone 13 pour assurer un alignement précis du téléphone avec le véhicule.

Un aspect clé de notre algorithme est la détermination de l'angle de lacet, alignant l'orientation du téléphone avec le mouvement du véhicule. Cela implique de comparer la direction du véhicule (à partir des données GNSS) avec l'orientation du smartphone et d'appliquer la méthode d'Euler pour ajuster les lectures de l'accéléromètre le long des axes x, y et z. L'objectif ultime est d'aligner les données de l'accéléromètre avec la vitesse GNSS, le freinage et l'écart de vitesse GNSS, en assurant la cohérence avec la direction du véhicule.

Le deuxième objectif de notre projet est de développer un modèle pour détecter les événements d'accélération et de freinage en utilisant les données GNSS et accéléromètre. Nous étiquetons les données en utilisant l'accélération GNSS, puis appliquons ces étiquettes aux données de l'accéléromètre. Notre approche teste divers modèles d'apprentissage automatique, y compris les arbres de décision, la forêt aléatoire, XGBoost et les réseaux LSTM, en se concentrant particulièrement sur les modèles performants avec des ensembles de données déséquilibrés.

Après des tests approfondis, nous avons constaté que le modèle de forêt aléatoire, avec un score F1 de 91%, surpasse les autres dans l'identification précise des événements d'accélération et de freinage. Ce score F1 élevé souligne son efficacité à gérer les complexités des données de conduite réelles.

Mots-clés:

sécurité routière, télématique, capteurs, accident, smartphone, assurance basée sur l'utilisateur.

ABSTRACT

In recent years, smartphones have become popular for collecting sensor measurements and traffic safety events, owing to their ability to reduce data collection costs substantially. The quality, uncertainty, and variability of these measurements pose barriers to widespread adoption. This research aims to leverage the kinematic data collected in the context of the Usage-based insurance (UBI) program from two cities in Canada, Montreal, and Ottawa.

The first goal of this thesis is to explore the alignment of smartphones with vehicles. Specifically, we utilize sensors such as the accelerometer, GNSS, and magnetometer. Our method is inspired by Android's algorithms for estimating azimuth from magnetic values combined with acceleration. This algorithm has been modified in response to the UBI dataset. We have also designed experiments using an iPhone 13 in a controlled environment. This ensures that the position of the phone in relation to the vehicle is known.

A crucial component of this algorithm is determining the yaw - the rotation of the phone so that it aligns with the motion of the vehicle. Our approach to determining this angle involves calculating the difference between the heading (obtained from GNSS recordings that indicate the direction of the car) and the smartphone's orientation. Subsequently, the Euler method is employed to rotate the accelerometer readings on the x, y, and z axes. The final step in our process is to verify that the new accelerometer values accurately represent the GNSS speed, braking, and deviation of GNSS speed, ensuring they are all aligned in the same direction as the vehicle.

The second objective of our project is to develop a model capable of detecting acceleration and deceleration events. This process begins with the utilization of GNSS and accelerometer data. Initially, we generate labels using GNSS acceleration data and then assign these labels to the instantaneous data from the accelerometer. Our approach involves experimenting with various machine learning models, including Decision Trees, Random Forest, XGBoost, and Long Short-Term Memory (LSTM) networks, to identify the most effective model for our purpose. Given that our dataset is imbalanced, we employ techniques to upscale the minority class. In doing so, we focus on optimizing the F1 score, which is a more appropriate metric for imbalanced datasets.

Upon evaluating the performance of these models, we conclude that the random forest model achieves the highest F1 score, at 91%. This indicates its superior capability in accurately categorizing acceleration and deceleration events compared to the other models.

Keywords:

traffic safety, telematics, sensors, crash, smartphone, user-based insurance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LISTE OF SYMBOLS AND ABBREVIATIONS

Avg	Average
AUC	Area Under Curve ROC
Acc_x	Acceleration in X axis
Acc_y	Acceleration in Y axis
Acc_z	Acceleration in Z axis
Acc_total	Total Acceleration
a_{GNSS}	Acceleration from GNSS
CV	Cross-Validation
DTW	Dynamic Time Warp
FN_i	Proportion of False Negative for class i
FP_i	Proportion of False Positive for class i
GBM	Gradient Boosted Machines
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
Hz	Hertz
HBE	Harsh Braking Events
ITS	Intelligent Transport System
LSTM	Long Short-Term Memory
Mx	Magnetometer in X axis
My	Magnetometer in Y axis
Mz	Magnetometer in Z axis
PAYD	Pay As You Drive
PAYG	Pay As You Go
PHYD	Pay How You Drive
$Racc_x$	Reorientated acceleration in x direction

$Racc_y$	Reorientated acceleration in y direction
$Racc_z$	Reorientated acceleration in z direction
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic Curve
SHAP	SHapley Additive exPlanations
SVM	Support Vector Machine
TN_i	Proportion of True Negative for class i
TP_i	Proportion of True Positive for class i
UBI	User-Based Insurance

CHAPTER 1 INTRODUCTION

A World Health Organization report [1] reveals that 1.5 million people lose their lives annually due to road collisions. It is estimated that traffic crashes cause most deaths among children and young adults aged 5 to 29 years old [1]. Traffic safety studies and their applications (real-time traffic updates, navigation assistance, emergency assistance, driver fatigue detection) have been conducted to enhance safety [2]. Vision Zero can be mentioned as an ambitious goal to diminish road accidents to zero within the next 30 years [3]. An accident or any severe injury could have a considerable impact on people's livelihood. Any countermeasure that could be taken to reduce the number of injuries and fatalities is necessary. Based on Transport Canada's National Collision Database, the number of fatalities was at an all-time low in 2020 with 1,745, which is down by 1% from the previous year. The number of collision injuries, 140,801, decreased to 101,572 in 2020, down 28% from 2019 [4]. However, if we break these numbers considering the number of severe and fatal crashes for car users, the number of severe collisions remains the same or, in some cases, increases. Road traffic crashes result in loss of life or physical disability due to sustained injuries. To avoid accidents, possible approaches can be taken: ensure that drivers are aware of safe driving behavior, collaborate with law enforcement agencies, and consider road geometry, thus we could incorporate new data sources into traffic safety. Researchers are increasingly interested in using telematics data to improve driving safety by providing feedback and diagnosing the causality of accidents.

Intelligent Transportation Systems (ITS) represent a comprehensive, integrated set of technology solutions designed to improve the safety, efficiency, and environmental friendliness of transportation systems [5]. The ITS applications include traffic control and management, vehicle tracking and management, traveler information systems, electronic toll collection, and advanced public transportation systems [6]. We aim to promote efficiency and safety by transmitting information from transport centers to travelers (users) [7]. Also, we want to take advantage of the collected data to diagnose safety issues. The rise in smartphone uses and the increasing adoption of sensors within these devices generate a significant flow of data. Combining these data with statistical learning and big data techniques can spawn novel opportunities in many industries, including insurance. Sensors on smartphones and dedicated communications systems in vehicles provide new perspectives on traffic data.

Based on the 2021 report by the National Safety Council of Canada, improper speed management or exceeding speed limits contributes to 3.62% of total accidents. These incidents result in a 5.59% death rate and a 3.34% injury rate [8]. Statistical studies have demonstrated a significant correlation between speed-related risk indices like acceleration and travel speed and historical accident data [9]. Unfortunately, most vehicles lack a detection system to report possible accidents. Accident warning systems have a lower penetration rate compared to smartphones. Smartphone-based driver monitoring, and smartphone-based telematics data sources are a prevalent trend, especially in the vehicle insurance industry. A smartphone can detect and report traffic accidents in vehicles lacking accident detection and notification systems. Mobile devices are particularly useful for driving monitoring because they provide a non-intrusive environment for continuously collecting rich information on the driving operation. Smartphone probes are more accessible than instrumented vehicles for naturalistic driving experiments. The installation and maintenance of instrumentation are challenging, especially when many vehicles are involved.

Telematics data is advantageous from two perspectives, reactive and proactive traffic safety diagnosis. Reactive traffic safety diagnosis refers to the process of identifying and addressing traffic safety issues after a traffic accident has occurred. This approach involves investigating the root cause of the accident and developing strategies to prevent similar accidents from happening in the future. Proactive traffic safety diagnosis involves identifying potential traffic safety hazards before they lead to accidents. This approach involves analyzing road networks, traffic patterns, and driver behavior to identify safety risks and implementing preventative measures to reduce the likelihood of accidents occurring [10].

Telematics data is collected from four sensors from cars or smartphones: the magnetometer, gyroscope, Global Navigation Satellite System (GNSS), and accelerometer. An accelerometer measures acceleration in all three directions. The accuracy of these measurements may be inferior compared to embedded sensors in the vehicle since these devices must be recalibrated frequently, and telematics data have shown that this calibration can be challenging [11]. It may, for instance, be affected by whether a car is parked on a steep road or on a flat plane during calibration. As well, it sometimes happens that the GNSS signal gets lost (or is imprecise) when, for instance, driving through a tunnel [12].

The GNSS, accelerometer, and gyroscope data can provide valuable insights into drivers' behavior before a crash. By analyzing the data, it is possible to identify patterns that may contribute to the likelihood of a crash occurring. Overall, collecting and analyzing crash data along with GNSS, accelerometer, and gyroscope data can provide valuable insights into driver behavior like braking and acceleration, or driver maneuvering and help improve safety on the roads. In addition, it may be possible to identify warning signs that could help prevent accidents in the future.

The utilization of accelerometer data plays an important role in ITS. However, the processing of acceleration data can be challenging, particularly in uncontrolled application scenarios, such as when collecting data in the background using a mobile phone while the user engages in everyday activities. In controlled scenarios, the smartphone's axes align with the vehicle's movement [13], [14]. Nevertheless, this alignment is not always applicable during regular smartphone usage, causing a mismatch between the axes and the frames of reference that accurately represent the user's experiences. In such cases, it becomes necessary to perform a series of rotations on the accelerometer's Cartesian axes to align them with a specific frame of reference. For example, if the smartphone is placed horizontally within a vehicle, and the goal is to capture events related to road irregularities from the shock absorbers, a reorientation of the accelerometer samples can be applied.

Detecting the orientation of smartphones poses a challenge in various domains, particularly when employing crowd-sensing or unsupervised experiments. The methods and strategies presented in existing literature exhibit variations based on the specific requirements and objectives of the experiment. Moreover, some approaches focus on orienting the smartphone in a specific situation and subsequently deducing different situations based on changes in orientation.

To illustrate the practical application of reorientation, consider the scenario of a vehicle involved in a crash. By employing the aforementioned techniques, the smartphone's position and orientation within the vehicle can be accurately determined. This information can be crucial for analyzing the impact forces experienced by the device during the crash and understanding the severity of the event [15]. Moreover, it enables researchers and engineers to gather valuable data for studying crash dynamics and developing enhanced safety measures.

Driver behavior monitoring systems that leverage smartphone sensor data to detect traffic events face significant challenges, particularly of data quality. These systems must effectively interpret

and analyze complex sensor data to accurately identify events such as hard braking, and rapid acceleration. One of the primary challenges in this domain is the extraction of meaningful features from raw sensor data, which can be highly variable and subject to noise. Machine learning algorithms play a crucial role in this process, as they can learn to recognize patterns and anomalies in the data that correlate with specific traffic events. However, the effectiveness of these algorithms heavily depends on the quality and preprocessing of the sensor data. This includes addressing issues like sensor alignment, data normalization, and the handling of varied sensor orientations and sensitivities. By focusing on advanced data processing techniques and robust machine learning models, it is possible to enhance the accuracy and reliability of traffic event detection using smartphone sensors, thereby contributing to more effective driver behavior monitoring systems.

The first objective of this thesis is to utilize and validate a method for the accurate reorientation of smartphone sensor data in vehicles. This research aims to address the challenges associated with the orientation and calibration of smartphone sensors in dynamic vehicular environments, thereby improving the reliability and effectiveness of driver behavior monitoring systems.

The second objective is to use telematics User-Based Insurance (UBI) data from smartphone sensors for driver behavior and safety. This involves using machine learning algorithms to interpret traffic-related events using data from accelerometers and GNSS, particularly focusing on acceleration and deceleration patterns. The main challenge is in developing a robust classification process for the data and addressing the issues arising from an imbalanced dataset. This thesis is structured to provide a coherent and detailed exploration of this subject. The introduction sets the stage, providing an overview of the criticality of traffic event detection for road safety and the emerging role of smartphones in monitoring driver behavior. It delves into the specific challenges faced when utilizing smartphone sensor data in traffic safety applications, particularly focusing on issues related to sensor reorientation and calibration.

Following the introduction, the literature review analyzes existing methodologies for collecting and interpreting smartphone sensor data in vehicles and the UBI concept. It reviews current approaches to traffic event detection, with a special emphasis on detecting harsh braking incidents, and critically examines the limitations and gaps in these existing methods.

In Chapter 3, the methodology section is divided into two sections to address the objectives. In the first subsection entitled phone reorientation, we describe the method proposed for reorienting

smartphone sensor data within vehicles. The second subsection, traffic event detection, details the development of algorithms aimed at effectively detecting traffic events, focusing on harsh deceleration. It outlines the experimental setup, including the strategies for data processing, labeling, and machine learning models.

In Chapter 4, data analysis and results section, we first illustrate the results and validate the smartphone reorientation process. Secondly, we present the machine learning models and evaluate the effectiveness of the algorithm for detecting traffic events. This section includes a comparative analysis with existing methods, highlighting the advancements in accuracy and reliability of our approach.

The discussion section, Chapter 5, interprets the results within the broader context of enhancing traffic safety. It discusses the strengths and limitations of the proposed methods and considers the implications of smartphone-based systems in driver behavior monitoring. The conclusion and future work section summarizes the key findings of the research and their significance in the field of traffic safety. It offers recommendations for the practical application of the research findings and proposes directions for future research to refine further and enhance smartphone-based traffic safety systems.

CHAPTER 2 LITERATURE REVIEW

Crash data are scarce in comparison with normal conditions (non-crash events); thus, any classification between crash and non-crash events would result in extreme imbalance. This issue can be seen in Figure 2.1, the hypothetical Heinrich's triangle for traffic safety.

In Figure 2.1, the probability of near crash event (for instance harsh brake) it is almost 10 to 100 times higher than for an accident. It is essential to consider the fact that most time spent driving is uneventful and that critical events are sparse. Therefore, it is essential to analyze the data in different ways, such as establishing distributions of driver performance metrics, categorizing drivers' behaviors into different driving styles [16] [17], and identifying risk factors in everyday driving [18]. This analysis can help in the development of effective driving safety measures, including infrastructure design, driver training, legislation, policymaking [19] [20], and intelligent safety systems [21].

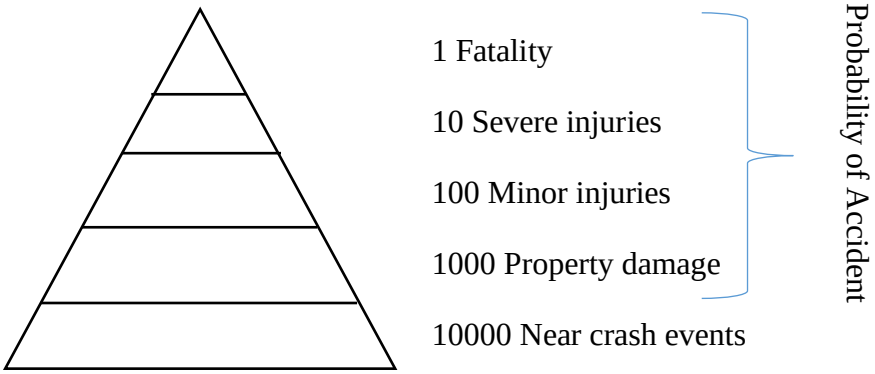


Figure 2.1 Heinrich's traffic safety triangle [22].

Several factors could influence the likelihood of a crash, and patterns in this data show that accidents are not random but rather tend to cluster by time and location. For instance, crashes are more likely to happen during rush hours a time of increased traffic volume [22]. Individual accidents still retain an element of randomness. For example, distractions or bad weather do not inevitably lead to a crash every time they occur. Urban intersections are hotspots for accidents, often labeled as "blackspots" due to their higher frequency of incidents [23]. These blackspots require particular attention and heightened safety measures to prevent crashes [24]. Meanwhile, the concept of "near-misses" or near crashes, which are incidents where an accident is narrowly avoided—like a driver braking suddenly to dodge a collision—led to a puzzle in domain traffic

safety analysis. It is an ongoing debate whether these near-misses, indicative of potential danger, could serve as reliable indicators or proxies for predicting crash sites, thus offering another avenue for improving road safety measures [25]. These factors, though statistically significant, do not guarantee an accident; they just raise the probability. It is this unpredictable interplay between deterministic and random factors that makes traffic accident prediction and prevention a complex task. This complexity is why analysis, and data-driven approaches are essential for improving road safety, as they help to understand and mitigate the risks where possible.

In conventional road safety analysis, the frequency and severity of crashes are the main factors for assessing traffic safety. In conventional traffic analysis, crash data determine the safety performance of a system. Surrogate measures of safety are indicators used to predict or estimate a transportation system's safety performance [26]. Surrogate measures of safety are becoming increasingly popular. These measures are derived from traffic data, such as near-crash events and harsh braking incidents and have the potential to enhance traditional traffic analysis methods by identifying crash precursors.

Some research studies have recognized that the number of atypical driving events, like harsh braking incidents, can serve as an indicator for safety models. Observable non-crash events, including vehicle maneuvers, near-crash occurrences, and driver behavior, represent spatial and temporal events that could potentially lead to collisions. By incorporating these alternative indicators, such as near-miss events, into traffic safety assessments, analysts can proactively identify and mitigate potential risks in the transportation network [27], [28]. As a result, safety can be assessed without having to wait for actual crashes [29]. Observing and identifying these near-crash events can occur at a higher frequency than analyzing data from traffic crash studies, which can often take several years.

Khattak et al. [33] designed a study to compare how various scenarios would affect the speed and acceleration of drivers' pre-event (before the crash). Volatility has been chosen as a surrogate measure to demonstrate the risks and driver comfort. Volatility refers to rapid and impulsive changes in speed or acceleration. Volatility is equal to the standard deviation of the acceleration multiplied by the square root of the number of periods in a specific time $\sigma_t = \sigma\sqrt{T}$. As a result of variations in driving behavior, different aspects of the volatility (reaction time) construct are captured and reflected in the variables of speeds, accelerations, and decelerations [30], [31].

2.1 User-Based Insurance

Traditionally, insurance premiums are determined by rating variables associated with the client and the vehicle, which means clients with similar characteristics (age, sex, location, and claims history) and driving similar vehicles pay similar premiums, regardless of driving behavior [32]. User-Based Insurance (UBI) is based on the concept that telematics and mobile applications give insurers detailed information about what the driver is doing and how the vehicle is used, allowing them to calculate premiums more accurately. UBI programs have emerged thanks to advances in insurance telematics, technologies for "sending, receiving, and storing information from and to road vehicles for insurance purposes" [33]. Drivers are encouraged to avoid risky behaviors in exchange for a discount. Afterward, a probabilistic risk model is developed to assign an individual risk profile for each driver based on their collected data [34]. Each driver's driving behavior can be analyzed to determine how likely one is to be involved in an accident. In traditional insurance programs, safer drivers essentially subsidize higher-liability drivers' insurance costs.

UBI is an umbrella term for insurance programs that use telematics or other technology to track the usage of a vehicle and customize insurance premiums based on the actual usage of the vehicle. This can include Pay-As-You-Drive (PAYD) programs [35], but it also includes other programs such as Pay-How-You-Drive (PHYD) [32] and Pay-As-You-Go (PAYG) programs [36]. The PAYD system charges premiums based on total exposure factors like mileage and road network, whereas the PHYD system is based on individual driving behavior factors like speed, harsh acceleration, and hard braking [37]. These plans provide a fair option for all users to pay for insurance based on their driving hours by gathering data while they are driving.

Using surrogate measures of safety at the driver level, UBI programs monitor drivers and adjust their premiums accordingly [38]. Surrogate measures of safety enable UBI to shift from reactive pricing (premiums only increase if a claim is filed) to proactive pricing (high-risk clients are identified before a claim is filed). Castignani et al. [39] proposed to collect telematics data within an application that works with the fuzzy logic model. This fuzzy logic model demonstrates a way to categorize drivers based on their driving behavior. This application would use accelerometer, gravity, and magnetic sensory data fused with GNSS data as input data to extract the following characteristics: acceleration, harsh braking, speeding, and aggressive steering events. In another study, researchers operated with similar datasets, including accelerometer, gyroscope, and

magnetometer data, to identify risky driving events such as sharp turning, aggressive acceleration and lane changing, and sudden braking. In this research, the dynamic time-warping technique is used on sensor data to distinguish risky and non-risky behavior [40]. Another application that can be mentioned is MobiDriveScore which detects risky events. In this application, advanced discriminative and generative modeling is utilized [41].

It remains a challenge for insurance companies to extract meaningful and explainable features from sensor data to distinguish risky drivers (a driver who make rapid movement during driving like harsh brake and plenty of overtaking) and provide fairer policies accurately. Most risk models currently available in the literature do not incorporate telematics data into modeling the drivers' behavior. According to the literature, research is concentrated on predicting the frequency of insurance claims and organizing crash data by the likelihood of injury. The input features are divided into seven different categories [42]. The driver's accident is based on the history of driving trajectories, driving events, and exposure (how much, when, and where someone drives), road geometry, the demographic of the driver, vehicle features, and speeding.

Indeed, the use of telematics data in risk modeling is rapidly gaining acceptance and popularity within the industry. As the field of telematics continues to evolve and mature, it is expected that more and more risk models will incorporate this valuable data source. An analysis of UBI programs was conducted by Ma et al. [45]. Furthermore, they gathered and examined GNSS data to demonstrate how it can be integrated into auto insurance pricing structures. The researchers found that mileage, peak travel times, and driving behavior, such as braking and vehicle maneuvering, strongly correlated with accident rates. In addition, they discovered that contextual factors, such as speeding and relative speed (velocity of the vehicle compared to traffic or speed limit), were significant risk factors [42], [43].

2.2 Vehicle Maneuver Detection

Several studies have linked aggressive driving maneuvers with traffic accidents, which is an essential topic for traffic, particularly insurance [44], [45]. It is generally accepted that speeding and harsh braking are dangerous driving behaviors that account for up to 10% of all traffic accidents [46]. Thus, identifying and detecting car maneuvers are essential for traffic safety analysis. Detecting car maneuvers can be done using various sensors and algorithms, depending on the specific maneuver to detect.

In general, detecting car maneuvers involves analyzing sensor data and comparing it to pre-defined patterns of behavior to determine if a maneuver is taking place. The following research is noteworthy regarding vehicle maneuver identification. In the research implemented by Ferreira et al., they utilized the sliding window to acquire the mean, median, and standard deviation from the sensors [47]. Likewise, Yu et al. [48] extracted standard deviation, mean, maximum, and minimum from the sensor data as extracted features. Constantinescu, et al. [48] employed a data mining approach to analyze several vehicle-based driving parameters, including hierarchical cluster analysis and principal component analysis, and identified four distinct driving styles categorized as aggression, speed, acceleration, and braking. These three studies used predefined thresholds and criteria to detect vehicle maneuvers.

Johnson and Trivedi [49] affirm that detecting a vehicle maneuver like a U-turn with only an accelerometer or gyroscope would be challenging. Sensor fusion could be a solution and dramatically enhance the accuracy of U-turn detection from 23% to 77%. Similarly, the GNSS and accelerometer sensors can be fused to enhance the accuracy of vehicle speed. Mumcuoglu et al. [50], applied a long short-term memory (LSTM) algorithm to classify distinct drivers based on longitudinal and lateral acceleration. The experiment was designed for six distinct drivers. Li [51] designed a vehicle maneuver detection system based on smartphone sensors. Vehicle maneuvers are detected by applying the stacked-LSTM model to the accelerometer and gyroscope data. The following movements are detected with this algorithm: straight, left turn, right turn, and U-turn.

The other approach is to develop applications to aid drivers while driving. Das et al. [52], in 2020 developed a dynamic lane change detection to assist drivers and improve their performance. A wrapper-based algorithm called Boruta was employed to select relevant features. These features are vehicle kinematics, machine vision features like lane position offset, roadway characteristics, and driver demographics under different weather conditions. The MODLEM algorithm (machine learning classifier) is used in another study, to find the minimal set of decision rules using sequential covering algorithm to detect harsh accelerations, braking, and cornering [53].

Liu et al. [54] designed an algorithm to detect hard braking via a transformer-based model using concurrent telematics sensors from smartphones and vehicle sensors. In this model, they employ the following features accelerometer, gyroscope, linear acceleration, and vehicle speed. This algorithm was found to be superior to GNSS speed-based and accelerometer-based heuristic

models, establishing a new benchmark in accuracy for detecting harsh braking events through sensor fusion.

In Table 2.1, we summarised the previous research on detecting vehicle maneuvers via smartphone sensors. The table outlines various research studies that utilize different combinations of sensors and analytical methods to detect specific driving behaviors and road conditions. These studies range from 2008 to 2021 and employ sensors such as accelerometers, GNSS, gyroscopes, and magnetometers to capture data related to vehicle dynamics and environmental interactions.

It shows that one of the necessary steps is to stabilize the reorientation of the phone during the trip. As can be inferred from the table, the accelerometer, along with GNSS sensors, are the predominant choice for feature selection. Moreover, machine learning techniques have been the preferable choice for detecting vehicle maneuvers in recent years.

Analytical approaches include threshold values, complex pattern matching, machine learning techniques like SVM and Hidden Markov Models, and advanced deep learning with Stacked LSTM. The detection purposes vary from identifying driving behaviors like braking, honking, and drunk driving to assessing road conditions such as potholes. Gradually, the methods have evolved from basic pattern recognition to sophisticated data-driven models, reflecting technological advancements in data processing and sensor capabilities.

Table 2.1 Studies on car maneuver data collected via smartphone.

Authors	Method	Sensors				Detection purpose
		Accelerometer	GNSS	Gyroscope	Magnetometer	
Mohan et al. 2008 [55]	Re-orientation, Pattern matching using threshold	✓	✓			Braking, honking, bumps
Dai et al. 2010 [56]	Re-orientation, Pattern matching using threshold	✓		✓		Drunk driving detection
White et al. 2011 [57]	Threshold value	✓	✓			Accidents
Bhoraskar et al. 2012 [58]	Support Vector Machine classification	✓	✓		✓	Bumps and braking
Saiprasert 2013 [59]	Threshold values, Dynamic Time Warping	✓	✓		✓	Lateral and longitudinal maneuver
Dang et al. 2014 [60]	Pattern matching	✓	✓			Braking
Daptardar et al. 2015 [61]	Hidden Markov models	✓		✓		Lateral and longitudinal maneuver, driver profiling
Singh et al. 2017 [60]	Crowdsensing, Dynamic Time Warping	✓	✓			Braking, lateral maneuver
Zang et al. 2018 [62]	Pattern matching using threshold	✓	✓			Road surface roughness potholes/humps
Vlahogianni et al. 2019 [53]	Re-orientation, Pattern matching using MODLEM	✓	✓	✓		Braking, acceleration, left cornering and right cornering
Li et al. 2020 [63]	Deep learning (Stacked LSTM)	✓	✓	✓	✓	Lateral and longitudinal maneuver
Gelmini et al. 2021 [64]	Re-orientation, Pattern matching using threshold	✓	✓	✓		Abnormal longitudinal decelerations, crash detection

CHAPTER 3 METHODOLOGY

As mentioned in Chapter 1, two main research objectives have been outlined in this thesis: the reorientation of smartphones to align with vehicles and traffic event classification. In the upcoming Sections 3.1 and 3.2, the methods used for each research objective are elaborated.

3.1 Phone Reorientation

Smartphones have accelerometers, gyroscopes, and magnetometers that use a coordinate system with three orthogonal axes x, y, and z, following a convention. This system adheres to a standard convention and is used to track the phone's orientation. Although the phone's position can change, this coordinate system remains constant in relation to the device's structure. The sensors on a smartphone, such as the accelerometer and gyroscope, provide measurements in a coordinate system that is attached to the phone itself. This is different from the phone's position data, which can be measured in an external x, y, z coordinate system, typically using a GNSS sensor. It is important to note that the phone's position is not measured in the same system as the accelerometer and gyroscope. The magnetometer, which measures deviations from the magnetic north, may also operate in a distinct system, given its unique functionality. This distinction is crucial for accurately interpreting and integrating data from these sensors. A vehicle's movement can also be described using a three-axis system. In this system, each axis is assigned a specific orientation from the driver's point of view. This is detailed in Figure 3.1 and Figure 3.2 which illustrate the vehicle's and phone's axes. We aim to align the smartphone's axes with those of the vehicle.

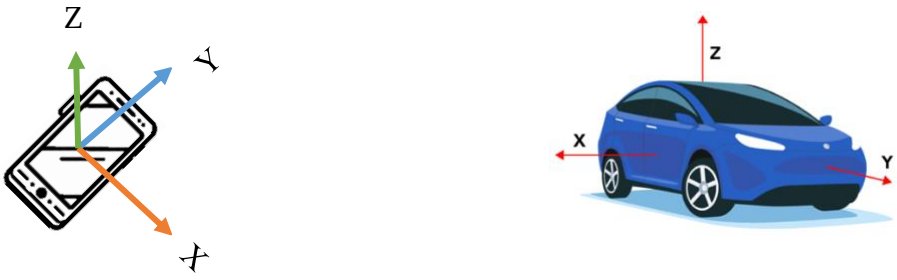


Figure 3.1 Coordinate the system of the vehicle and the smartphone.

The research explores the use of smartphones as inertial measurement units to acquire data about a vehicle's motion. Inertial measurement units include accelerometer, gyroscope, and magnetometer. However, it is challenging to interpret the collected data due to the various positions

of the smartphone in the car and the user's movement while traveling. To address this challenge, it is best practice to align the collected data along the car axes automatically. This is done by rotating the data points in 3D space until the orientation between each sensor axis and the corresponding vehicle axis is zero, indicating alignment.

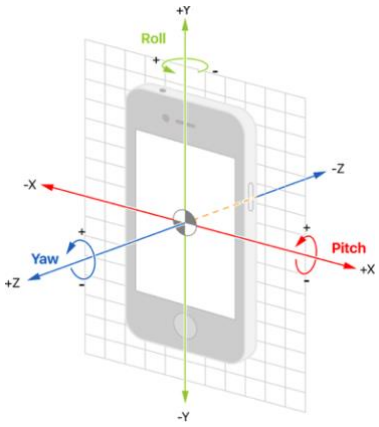


Figure 3.2 Orientation upon the phone axes roll, pitch, and yaw.

During a trip, smartphones' orientation may change, which creates a challenge for motion recognition. For this reason, based on existing research, we used another sensor that would not be affected by the reorientation of the phone as a benchmark to meet this challenge. The approach employs a combination of three sensors, namely accelerometers, magnetometers, and GNSS data. Data from the accelerometer and magnetometer are fused to create a compass, which provides the angle between the smartphone and the magnetic north, called the azimuth [65]. This angle is then compared with the heading, or direction of travel, obtained from GNSS, to determine the orientation of the smartphone relative to the direction of the vehicle it is traveling in [66]. Prior to testing the reorientation approach, the reliability of the sensor reading of the magnetometer was assessed due to potential electromagnetic noise from the car's electrical systems and the susceptibility of smartphone magnetometers to environmental magnetic interference [67]. Yaw is the difference between the GNSS-reported heading and azimuth see Figure 3.2. Understanding this discrepancy is crucial for tackling the challenges in reorienting UBI data, and it sheds light on the difficulties we encountered in applying the same reorientation method to UBI data.

3.1.1 Euler Theorem

Euler angles are a set of three angles used to describe the orientation of a rigid body with respect to a fixed coordinate system. They are commonly represented by symbols such as α, β, and φ, or

respectively by the terms roll, pitch, and yaw. The concept of Euler angles is defined as follows: Consider two right-handed Cartesian reference frames in 3D space, one referred to as the fixed frame and the other as the mobile frame. Initially, these frames coincide with each other [68].

The Euler's rotation theorem demonstrates that reorientation (taking values from one frame of reference and expressing them in another) is possible through a linear transformation. Euler's theorem states that any three-dimensional orientation can be achieved by rotation about a fixed axis. This means that any coordinate system rotation can end at the desired orientation, regardless of how the initial values are rotated (see equation 3.4) [68]. The transformation can be expressed as three successive rotations [69]. As a result, different rotation matrices can be used to accomplish the same transformation in different orders [70].

- Rotate the mobile frame around the x, y, or z axis of the fixed frame, or around the x', y' or z' axis of the mobile frame, by an angle α.
- Rotate the mobile frame again around the x, y, or z axis of the fixed frame, or around the x', y' or z'axis of the mobile frame, by an angle β.
- Finally, rotate the mobile frame around the x, y, or z axis of the fixed frame, or around the x', y' or z'axis of the mobile frame, by an angle φ.

The specific sequence in which rotations are applied is critical. Although there are theoretically 216 possible ways to order these rotations, such as rotating around the x-axis followed by the y-axis and then the z-axis (notated as x→y→z), or other variations like y→x→z, z→y→x, many of these are redundant or do not represent a general spatial orientation. This redundancy occurs because rotations around the same axis are not meaningful consecutively. Additionally, before any rotation is applied, each axis of the original system aligns with its corresponding primed axis—x with x', y with y', and z with z'. Therefore, there are only six unique sequences that lead to distinct orientations without repeating any axis in the rotation sequence. These are the Tait-Bryan angles, which do not iterate any of the axes in their rotation sequence. The notations like XYZ refer to a rotation first around the x-axis, then around the y-axis, and finally around the z-axis, respectively. Other unique sequences include YXZ, ZXY, XZY, YZX, and ZYX, as cited in reference [71].

Accelerometers, when stationary, primarily detect gravitational acceleration. In a steady state, an accelerometer will register a force equal to 1g on the Z-axis (if Z axis is oriented along the gravitational force). The orientation concerning this Z-axis can be altered by tilt, which can be described using: $\alpha = \arcsin\left(\frac{\mathrm{Acc_x}}{\mathrm{g}}\right)$, where Acc_x represents the acceleration in the x direction. Pre-rotation followed by tilt would introduce non-zero values for $\mathrm{Acc_x}$ and $\mathrm{Acc_y}$ due to the influence of gravity. The relationship between β and acceleration is presented as follow: $\beta = \arcsin\left(\frac{\mathrm{Acc_y}}{\mathrm{g}}\right)$, where $\mathrm{Acc_y}$ represent the acceleration in the y direction.

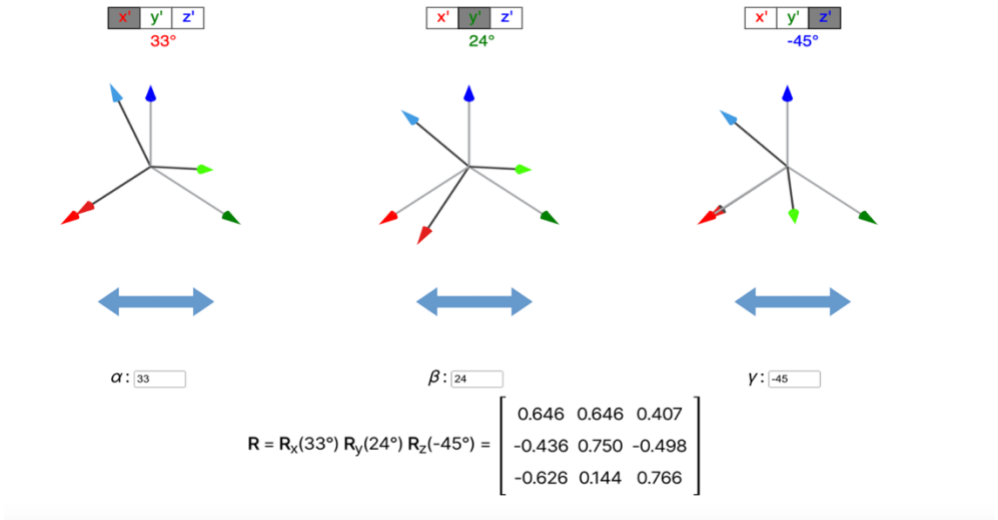


Figure 3.3 The plausible orientation of the vector after applying rotation in x, y, and z [72].

To estimate α and β, it is possible to identify periods when the phone is stationary (at a traffic light) or undergoing steady motion, with GNSS speed equals to zero. However, a simpler approach, which has proven effective in practice, is to utilize the median values of acceleration in (x, y, z) over a 10-second window. Remarkably stable, the median value over this duration remains unaffected even during a bumpy drive. This stability can be attributed to the temporary nature of acceleration spikes, which tend to settle back around the median within a few seconds or less. For instance, if the vehicle experiences a bump causing a spike in the g-force, it will subsequently descend, resulting in a spike in the opposite direction. However, the median value remains largely unchanged. By reading acceleration in (x, y, z) over short time windows, it becomes possible to estimate α and β continuously.

Any significant alterations in α and β indicate notable changes in the phone's orientation. However, the reverse is not always true. If the phone is deliberately rotated about the Z-axis, α and β would remain unaffected. Figure 3.3 illustrates how changing each angle affects the orientation of the phone. In equation 3.1 using acceleration in x, y and z direction, α and β, the yaw φ can be estimated.

$$\varphi = \text{Arctan}\left(\frac{\text{Acc}_x * \cos\alpha - \text{Acc}_z * \sin\alpha}{\text{Acc}_x * \sin\alpha * \sin\beta - \text{Acc}_y * \cos\beta + \text{Acc}_z * \cos\alpha * \sin\beta}\right) \quad (3.1)$$

3.1.2 Application of Euler Theorem to smartphone Reorientation

Input data are accelerometer $\begin{bmatrix} Acc_x \\ Acc_y \\ Acc_z \end{bmatrix}$ and magnetometer $\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}$ and heading from GNSS. It is important to check the constraints before running any algorithm that involves sensor data processing. Constraints can include limitations on the range or magnitude of the sensor readings, as well as other physical or environmental factors that can affect the sensor readings. If any of the constraints are not met, it can lead to incorrect or inaccurate results. For example, if a sensor reading is affected by external factors such as picking up the phone or falling, this can result in a distorted or unreliable signal. Similarly, for magnetic sensors, the presence of other magnetic sources in the vicinity can influence the readings and result in errors. Therefore, it is important to account for these constraints and take measures to mitigate their effects. This may involve preprocessing the sensor data to remove unwanted noise to account for the influence of external factors on the sensor readings [73]. The following constraints should be applied to the data to ensure we would not collect error (noise) with values.

$$\text{Acc}_x{}^2 + \text{Acc}_y{}^2 + \text{Acc}_z{}^2 < 9.81^2 * 0.1 \text{ (Acceleration should be less than g)} \quad (3.2)$$

$$\begin{aligned} \text{H}_x &= \text{m}_y * \text{Acc}_z - \text{m}_z * \text{Acc}_y \\ \text{H}_y &= \text{m}_z * \text{Acc}_x - \text{m}_x * \text{Acc}_z \\ \text{H}_z &= \text{m}_x * \text{Acc}_y - \text{m}_y * \text{Acc}_x \end{aligned} \quad (3.3)$$

$$\text{Norm(H)} = \sqrt{\text{H}_x^2 + \text{H}_y^2 + \text{H}_z^2} < 0.1 \text{ (Device is close to free fall or close to the magnetic north pole)} \quad (3.4)$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (3.5)$$

The following is a description of one such convention. Equation 3.5 shows the reoriented value with Euler theorem namely x', y' and z'. To specify the orientation of the second frame, which shares the same origin as the first frame, the mobile frame is successively rotated.

R is the rotation matrices representing the rotation of the sensor data around the corresponding axes. Four of the rotation sequences' corresponding rotation matrices are unsuitable for determining smartphone orientation. This is because the accelerometer output, which has three components, only has two degrees of freedom since the vector magnitude always equals 1g in the absence of linear acceleration. The accelerometer vector lies on the surface of a sphere with a radius of 1g. It is conventional, therefore, to select either the rotation sequence R_{xyz} or R_{yxz}.. Lastly, we need to use Euler angles as stated below to calculate R_y, R_z, and R_x.

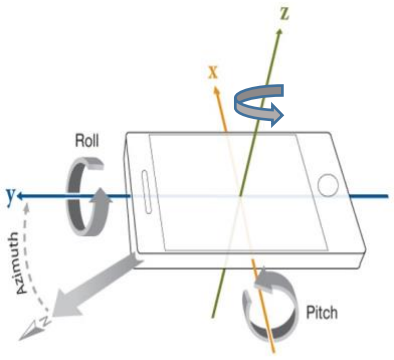


Figure 3.4 Azimuth and orientation of the phone.

$$. R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\beta & -sin\beta \\ 0 & sin\beta & cos\beta \end{bmatrix} R_y = \begin{bmatrix} cos\alpha & 0 & -sin\alpha \\ 0 & 1 & 0 \\ sin\alpha & 0 & cos\alpha \end{bmatrix} R_z = \begin{bmatrix} cos\varphi & sin\varphi & 0 \\ -sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\alpha = \arcsin\left(\frac{\mathrm{Acc_x}}{\mathrm{g}}\right) \text{ and } \beta = \arcsin\left(\frac{\mathrm{Acc_y}}{\mathrm{g}}\right) \quad (3.6)$$

$$\varphi = \mathrm{Arctan}\left(\frac{\mathrm{Acc_x} * \cos\alpha - \mathrm{Acc_z} * \sin\alpha}{\mathrm{Acc_x} * \sin\alpha * \sin\beta - \mathrm{Acc_y} * \cos\beta + \mathrm{Acc_z} * \cos\alpha * \sin\beta}\right) \quad (3.7)$$

With known angles for each direction, by doing matrix multiplication, X_i, Y_i, and Z_i for i represent i-th instant for reading of acceleration in each direction, the new orientation of the phone is calculated as follows:

$$X_i = x_i(\cos\alpha\cos\varphi - \sin\alpha\sin\beta) + y_i\cos\beta\sin\varphi - z_i(\cos\varphi\sin\alpha + \cos\alpha\cos\varphi\sin\beta) \quad (3.8)$$

$$Y_i = -x_i(\sin\alpha\sin\beta\cos\varphi - \cos\alpha\sin\varphi) + y_i\cos\beta\cos\varphi + z_i(\sin\varphi\sin\alpha - \cos\alpha\cos\varphi\sin\beta) \quad (3.9)$$

$$Z_i = x_i(\sin\alpha\cos\beta) + y_i\sin\beta + z_i(\sin\varphi\sin\alpha - \cos\alpha\cos\varphi\sin\beta) \quad (3.10)$$

The formula provided above demonstrates how to estimate new orientation based on the phone's angle in each direction. In these equations, x_i, y_i and z_i are the initial readings of acceleration in the respective X_i, Y_i and Z_i. The angles α, β, and ϕ represent the phone's orientation in terms of roll, pitch, and yaw, respectively.

To reorient sensor data collected from smartphones in vehicles, it is crucial to adjust each data observation according to the magnetometer value. A straightforward method for reorienting the data is to align the readings from the accelerometer with those from the magnetometer, modifying each data point based on the smartphone's sensors and the vehicle's heading as provided by GNSS.

By applying a Euler rotation matrix, the phone's orientation and the vehicle's motion in three-dimensional space can be precisely calculated and represented. Reorienting each data point in this manner can enhance the accuracy and reliability of sensor readings, which in turn enables more effective driver behavior monitoring. Nevertheless, this process has its challenges, such as sensor drift, interference from other magnetic fields, and changes in orientation, all of which can impact the accuracy and reliability of the sensor data. Magnetic declination, the angular difference between geographic north and the direction a compass needle points, is a navigational challenge resulting from the Earth's complex magnetic field. This misalignment, such as the -13.5-degree declination observed in Montreal, affects not only traditional navigation but also the accuracy of digital compasses in smartphones used for capturing vehicle sensor data.

To address these challenges, it is crucial to develop strategies that mitigate their effects. For instance, one can use ground truth data to validate sensor readings and adjust the reorientation process as needed. We recommend using a braking event as the reference (ground truth), which is

expected to exhibit higher acceleration values in the y-axis direction. The primary assumption is that the phone remains in the same direction after braking.

In essence, the phone undergoes two reorientation processes: first, by using the magnetometer value to align with the compass direction, and second, based on the ground truth to maintain the same direction as the car. By taking these steps, the sensor data can be made more accurate and reliable.

3.1.3 Reorientation Algorithm

When the data obtained from smartphone sensors exhibits significant errors, any conclusions drawn from such data become inherently unreliable. This issue becomes particularly prominent in smartphone studies compared to earlier research endeavors where researchers provided participants with measurement devices [74]. In previous studies, researchers had the ability to carefully select appropriate devices, preconfigure all necessary components, and verify the data's reliability. Although some errors might still exist, they could be examined and potentially mitigated. Furthermore, the error was generally consistent across the participant sample.

In contrast to previous research studies where researchers provided participants with specific measurement devices, most smartphone studies rely on participants using their personal devices. This introduces several factors that can influence the data quality and reliability.

Firstly, there is a wide variety of phone brands and models used by participants in smartphone studies. Different brands may employ different sensor technologies, resulting in variations in data accuracy and reliability. For example, some high-end smartphones may incorporate advanced sensors that offer more precise measurements, while lower-end or older models may have less accurate sensors or lack certain types of sensors altogether.

Secondly, the frequency of data sampling (in our dataset is 10 per second), also known as the sample rate, can vary among different smartphones. The sample rate determines how often data is captured by the sensors. Some smartphones may have higher sample rates, allowing for more frequent and detailed data collection, while others may have lower sample rates, leading to less granular or intermittent data. Additionally, the choice of operating system (iOS or Android) can also impact the quality and availability of sensors in smartphones. While both operating systems offer a range of sensor capabilities, there may be differences in sensor availability and performance

between iOS and Android devices. Researchers conducting smartphone studies need to consider these variations when interpreting and analyzing sensor data. The methodology leverages trigonometric calculations to derive orientation angles from the rotation matrix. In the following, we will present in detail the algorithm objectives, inputs, steps for processing of each algorithm and lastly the outputs are presented:

Algorithm 1: Rotation Matrix.

Objective	To compute the rotation matrix from gravity and geomagnetic field vectors.
Input	Gravity vector (gravity) and geomagnetic field vector (geomagnetic).
Process	1. Validate the magnitude of the gravity vector to avoid free fall conditions. 2. Compute cross product of gravity and geomagnetic vectors. 3. Normalize vectors and compute the rotation matrix R and inclination matrix I. 4. Return R and I if conditions (eq. 3.2-3.5) are met, else return False.
Output	Rotation R and inclination I matrices.

Algorithm 2: Azimuth

Objective	This algorithm calculates the azimuth angle using accelerometer and magnetometer data, which is crucial for determining the orientation of the smartphone in a vehicle.
Input	Accelerometer readings (Acc_x, Acc_y, Acc_z) and magnetometer readings (m_x, m_y, m_z).
Process	For each set of accelerometer and magnetometer readings: 1. Calculate rotation matrices using algorithm 1. 2. Compute the azimuth angle and adjust with declination related to location. 3. Return the list of azimuth angles.
Output	A list of azimuth angles corresponding to each set of sensor readings.

Algorithm 3: Reorientation Matrix

Objective	To reorient accelerometer data based on the computed azimuth and heading angle.
Input	Accelerometer readings (X, Y, Z) and yaw angle (phi).
Process	1. Calculate the median of X, Y, Z to find the gravitational component. 2. Compute the angles alpha and beta eq. 3.6. 3. Phi is estimated by deducing azimuth from the heading (output algorithm 2). 4. Apply trigonometric transformations to reorient the accelerometer data (eq. 3.8- 3.10). 5. Return the reoriented data.
Output	Reoriented accelerometer data.

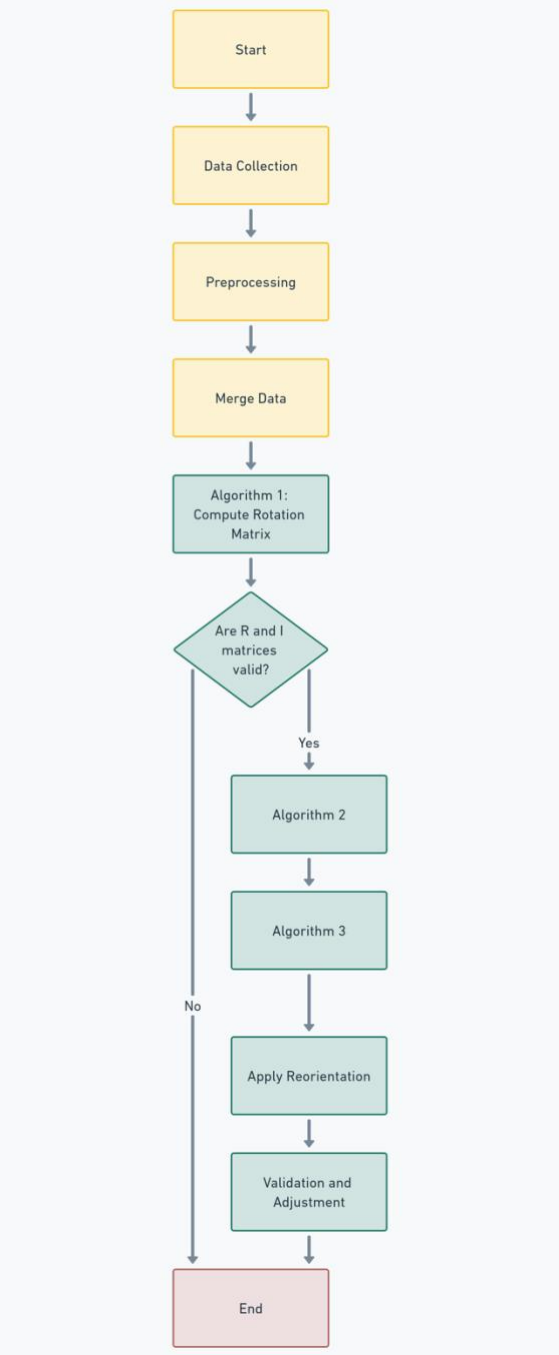


Figure 3.5 Flowchart outline for smartphone reorientation.

3.2 Traffic Event Classification

To detect the driver's behavior maneuver, two main approaches have been proposed [35]:

- Threshold-based approach
- Machine learning

Threshold-based and machine learning approaches. While traditionally threshold-based methods involve manually setting thresholds and are straightforward to implement, they can also be viewed under the machine learning umbrella when thresholds are determined algorithmically, similar to a decision tree with a depth of one. This can make threshold-based methods inflexible, as the manually fixed thresholds might not be appropriate for all drivers or driving conditions. Furthermore, they can suffer from high false positive rates, as they might classify normal maneuvers as sharp ones if the thresholds are set too low.

Machine-learning methods can learn from the input data and be adjusted to different drivers and driving conditions. They can be more accurate than threshold-based methods, and they can also detect maneuvers that might not be captured by simple threshold-based approaches. However, they require more computational resources and data to train, and their performance heavily depends on the quality and quantity of the data used for training. Among the two mentioned approaches, the machine learning approach is widely used in the detection of anomalies.

In summary, both threshold-based and machine-learning methods can be used for driver maneuver and driver behavior classification, and the choice depends on the specific application (type of maneuver) and available resources. Threshold-based methods are easy to use, while machine-learning methods can learn more complex data structures.

In this thesis, based on a dataset extracted from smartphone sensors, both methods have been implemented; the threshold-based method has been utilized for the segmentation of time series data, whereas machine learning tools have been used in the classification of the telematics data (time series data).

Traditional data sources for traffic crashes often provide limited information, which hinders detailed analysis. The data is typically sparse, meaning it lacks the fine-grained detail necessary to draw in-depth conclusions about crash causes, patterns, and trends. As a surrogate for actual crashes, hard-braking events (HBEs) have been studied. These events are more frequent and can

be detected with vehicle sensors. The paper [75] delves into the challenge of understanding pattern of time series data for traffic event. This paper introduces a novel method to detect HBEs using smartphone sensors. Their primary aim is to develop a scalable approach for detecting hard-braking events using kinematics data obtained from smartphone sensors. This is advantageous because it does not rely on fixed vehicle sensors and is hence more widely applicable. Several machine learning models and heuristic algorithms were tested. The Transformer model outperformed others, including Convolution Neural Networks (CNN), Long Short-Term Memory (LSTM), and fully connected models. Two heuristic detectors based on accelerometer and GNSS speed were used as baselines. Model performance was evaluated using precision-recall and ROC curves, with the Transformer-based model achieving a Precision-AUC or Area under the Curve ROC of 0.83. We aim to replicate a similar procedure with GNSS speed and the accelerometer sensor.

3.2.1 Manoeuvre Detection Preprocess

Sensor selection and fusion are crucial factors affecting vehicle maneuver recognition. Sensor fusion has shown promising results in improving performance but using too many sensors can increase battery consumption and computation costs [76]. Among different types of sensors, the gyroscope detects the motion of the vehicle while turning, whereas the accelerometer is more suitable for lane change and braking detection. Over the past two decades, various methods have been proposed to detect real-time vehicle maneuvers based on the data extracted from smartphone sensors mentioned in Table 2.1.

Detecting vehicle maneuvers using smartphone sensors presents several challenges that must be addressed. While previous studies have fixed the smartphone on a mount, our research uses reoriented smartphone data such that Z direction is fixed and not aligned to the vehicle direction, which adds a new level of complexity. Moreover, data preprocessing techniques or extracted feature utilization employed by some studies to improve model performance may vary in effectiveness depending on the specific dataset and context. Furthermore, this research aims to extract vehicle maneuvers from an unlabeled dataset, without any control over the type of smartphone used or the frequency of records. This presents a unique challenge, as variations in sensor quality and sampling rates within the dataset could affect the accuracy of the maneuver detection algorithm.

Sensor fusion is implemented to combine data from multiple sensors, thereby obtaining a more accurate and reliable output. Timestamps are used to synchronize data from different sensors, and the data recording frequency can vary depending on the sensor type. GNSS data typically has a lower frequency due to satellite positioning update limitations. A timestamp of 1 second (1 Hz) is a standard rate for GNSS data, while accelerometer readings are more frequent, approximately 9-10 Hz or every 0.1 s.

When fusing sensor data with different timestamps and frequencies, it is important to consider time misalignment and interpolation techniques for proper data synchronization. This may involve resampling or interpolating data from lower-frequency sensors to match higher-frequency data. After preprocessing the data, the objective is to detect possible driver actions (maneuvers).

The underlying hypothesis is that specific maneuvers can be identified based on their attributes and patterns. Every maneuver exhibits distinct features, and if similar patterns reoccur, the same maneuver is expected to take place. For example, rapid acceleration typically occurs at intersections or on straight roads when the driver slams on the accelerator pedal to quickly accelerate the vehicle. Harsh braking occurs when the driver feels the vehicle is too close to the one in front and firmly presses the brake pedal to rapidly reduce speed, potentially resulting in a rear-end collision. Therefore, a change in the pattern of acceleration is expected.

To effectively manage a nine-instant gap in GNSS data, linear or constant interpolation methods were considered; however, these methods typically yield a correlation of only 0.3 with acceleration because they fail to take into account the curvature of the speed-time graph, simply averaging speed over time and neglecting acceleration variability. Consequently, our approach shifted to employing second-order polynomial interpolation, which not only factors in this curvature but has also demonstrated a higher correlation—approximately 0.7—between GNSS speed and acceleration a_{GNSS}. This correlation aligns with the expected relationship between these two variables from a single sensor and has proven, through testing, to provide superior results. Following the successful interpolation of the speed data, we integrated the enhanced speed data with the acceleration data from the GNSS and other sensors for a more comprehensive analysis.

In our analysis, we categorize GNSS instantaneous readings based on a_{GNSS}. We have established two predetermined thresholds at ±2.5 m/s². For any instantaneous reading where a_{GNSS} exceeds 2.5 m/s², indicating rapid acceleration, we assign a label of 1. Conversely, when a_{GNSS} is less than -

2.5 m/s², suggesting significant deceleration, we label it as 2. Instantaneous readings where a_{GNSS} falls within the range of -2.5 to 2.5 m/s² are considered to represent normal driving conditions and are thus labeled as 0.

To ensure that each event is comprehensively represented and to account for the temporal aspect of driving behavior, we have introduced a buffer of three seconds, equivalent to 30 instantaneous readings, around each labeled event. This decision is based on the recommendations by Liu [77], who both suggest a 3-second boundary for accurate event representation. This principle is evident in Figure 3.6, where a noticeable spike in acceleration surpasses a set threshold. Following these steps (depicted in Figure 3.7), we have created a database where each instantaneous reading is appropriately labeled, providing a clear indication of normal driving, acceleration, and deceleration events. This refined dataset is now ready for classification. However, it is crucial to note that our dataset is unbalanced, a factor that should be carefully considered during any analytical procedures.

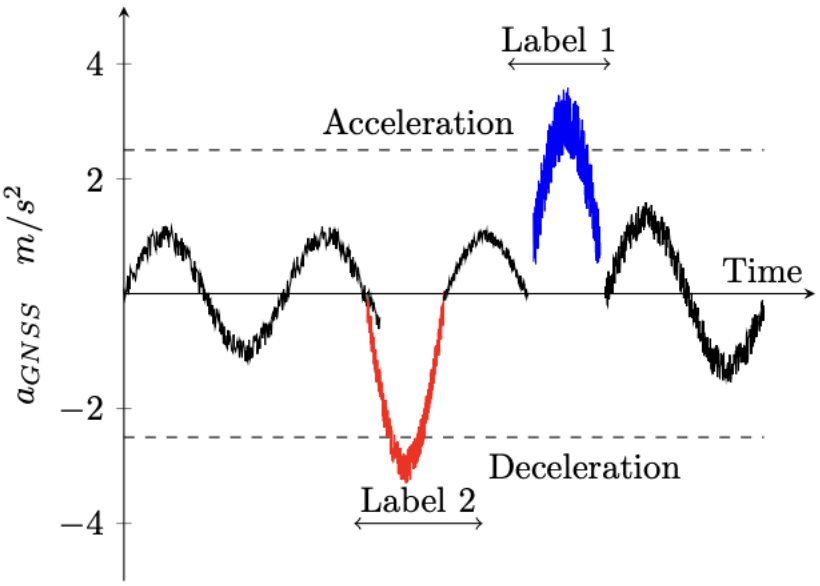


Figure 3.6 Traffic event window before and after the event.

Figure 3.7 Flowchart outline for traffic event classification.

3.2.2 Dataset for Classification

In this study, we utilize two sets of data. The first set is from a single trip by a specific driver, with data collected from two sensors: GNSS and an accelerometer that includes three axes: x, y, and z. The GNSS provides data on the trip's length, speed, heading, and the longitude and latitude of the phone. Additionally, we compute the total acceleration. For this trip, the duration is approximately 20,000 s, or 5.5 h of driving.

In examining the dataset, we observe a significant imbalance among the labels. Most of the data, approximately 99.35%, belong to normal driving (represented by 79,632 instantaneous data points). In contrast, acceleration and braking are sparsely represented, accounting for only 0.31% (with a count of 246 instants corresponding to four acceleration event) and 0.34% (with a count of 274 instants corresponding to five braking event), respectively. Such disparities in representation can lead to biased models and compromise the generalization capabilities in real-world scenarios. To mitigate this, we incorporated weights to account for the size disparities, ensuring that the model is adequately sensitive to the nuances of the less frequent labels.

In our dataset, the input dimension is four, including the spatial readings from the 3-D accelerometer (x, y, z) and the total acceleration. It is essential to note that no overlap will occur during the labeling process because we first identify acceleration and braking and then categorize the rest as normal driving. This approach ensures that critical events, such as braking, are distinctly identified without interference from the more common normal driving segments.

These input features for the LSTM slightly differ with consideration of time interval as the input value with time window of 30 instants and overlap of 50%. To clarify, the size of the input refers to the number of sequential data points fed into the model at each time, and the overlap indicates the proportion of data points in one time step that are re-used in the next step. Labels are assigned to each input sequence based on the desired output of the model for that specific period, ensuring the temporal relevance of the data to the predicted variable.

Post-labeling, the GNSS variables were eliminated, leaving only data from the accelerometer sensor for subsequent processes. This pruned dataset, comprising the accelerometer readings as inputs and the corresponding labels as outputs, was then partitioned into training and testing subsets at a ratio of 80/20. The test set (20% of the data) was reserved exclusively for testing purposes. For hyperparameter tuning, the train dataset was used in a 5-fold cross-validation process to ensure that the optimization was robust and not biased toward a specific subset of the data.

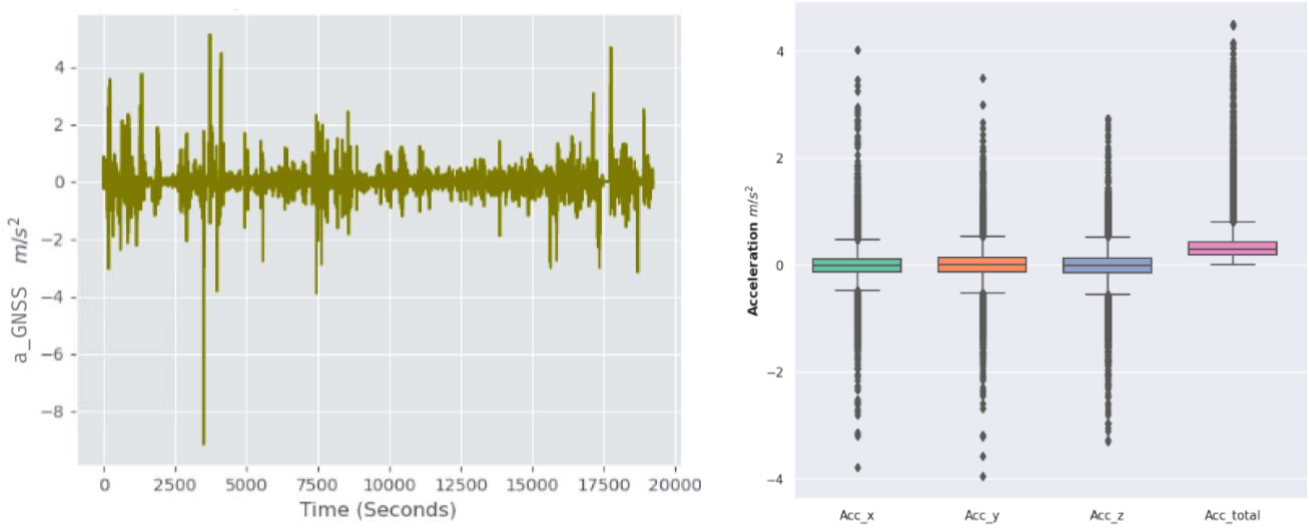
The second dataset is a compilation of 100 trips from 100 distinct drivers, containing more than 2 million data points. After processing the second dataset analogously to the initial one, we applied the best-trained model from the inaugural 5.5-hour trip dataset to it. The primary motivation behind this approach was to validate the model's scalability and generalization capabilities on a larger dataset. The anticipated benefits include assessing the model's efficiency across diverse datasets,

negating the need for frequent retraining, and saving computational resources. This method provides a benchmark for performance, ensuring the model's robustness.

As shown in

Figure 3.8, the chart on the left side is a_{GNSS} of the acceleration drive from GNSS. The standard deviation for this a_{GNSS} is 1.51 and There are peaks that go up to about 4 m/s² and down to about -8 m/s². This chart shows the vehicle acceleration that corresponded for labelling the traffic event into acceleration, deceleration, and normal driving. The right-side box plot represents Acc_x, Acc_y, Acc_z and Acc_{Total} as the input values (before allocating to train and test sets) illustrating key statistical such as the median around zero, the range between the maximum 4 and minimum -4 values, and outliers.

The bottom figure presents a temporal analysis of the acceleration components $Acc_x, Acc_y,$ and Acc_z delineated by color coding for each respective axis. The data demonstrate considerable variability with standard deviations of 0.225 for Acc_x, 0.432 for Acc_y, , and 0.378 for Acc_z, while maintaining an approximate mean of zero, indicative of an oscillatory pattern centered on the origin over the observed time.



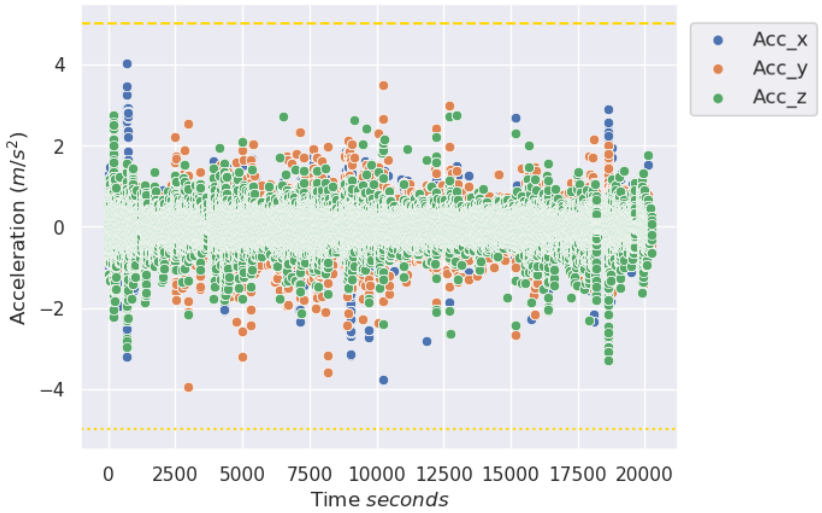


Figure 3.8 Data description for the entire data.

3.2.3 Machine Learning Classification

In the following sections, we discuss four machine learning models: Decision Tree, Random Forest, XGBoost, and LSTM. We address the issue of class imbalance for the acceleration and deceleration classes, which we attempt to mitigate by tuning the models and selecting appropriate evaluation metrics.

3.2.3.1 Decision Tree

Decision trees segment the data space into homogenous regions based on feature thresholds. decision trees stand out as one of the most intuitive and visually appealing algorithms. Designed to mimic human decision-making processes, these tree-like structures systematically divide datasets into smaller and more homogenous subsets based on feature values. By doing so, they transform the complex landscape of data into a series of straightforward decisions, providing clear insights into the underlying patterns and relationships. At the heart of decision trees lies the concept of node splitting.

The algorithm initiates the decision-making process by selecting the feature that provides the greatest decrease in node impurity. This initial step is crucial for the success of the model, particularly in classification tasks where the goal is to create as pure a node as possible. Gini impurity, for example, assesses the chance of a random element being mislabeled, striving for a score of zero which would imply perfect classification. Entropy, another measure, evaluates the

level of disorder within the data, with higher values signifying more chaos and uncertainty. Instead of these, one might consider the classification error, which reflects the proportion of incorrect predictions in relation to the total number of instances. In regression contexts, variance reduction is key—it ensures the model's predictions are as close to the actual outcomes as possible, aiming for minimal prediction error [78], [79]. The following terminologies are used to describe the model analysis:

1. Node Splitting: For each feature, the algorithm determines the best feature by evaluating certain criteria.

- For Classification: The criteria can be Gini impurity, entropy, or classification error.
- For Regression: The criteria are typically the reduction in variance.

2. Tree Growth: After determining the best split, the data is partitioned, and the process is repeated recursively for each subset.

3. Stopping Criteria: The tree stops growing when a predefined depth is reached, when the node contains fewer samples than a predefined threshold, or when it cannot achieve a reduction in the splitting criterion. Also, the process will stop if the purity of a node exceeds a certain predefined level, indicating a high degree of homogeneity in the data within that node. Lastly, the splitting process will also halt if all the records within a node have identical values for the predictor variables, leaving no basis for further meaningful division. These criteria collectively ensure that the decision tree remains efficient, relevant, and prevents overfitting.

4. Pruning: To avoid overfitting, branches of the tree that have little power in predicting the target variable can be pruned.

The measures of Gini impurity, entropy, and classification error are introduced after pruning to explain the criteria by which the decision tree decides whether a branch should be pruned. The p_i represent the probability of class i. The lower the impurity, the less likely the tree is to benefit from further splitting that node, and the more likely that node or branch can be pruned. Each of these measures provides a different way to quantify impurity:

$$Gini(t) = 1 - \sum_{i=1}^{c} p_i^2 \tag{3.11}$$

$$Entropy(t) = -\sum_{i=1}^{c} p_i \, log_2(p_i) \tag{3.12}$$

$$Classification\ Error(t) = 1 - max(p_i|t) \tag{3.13}$$

The $p_i|t$ denote the proportion of samples of class i in node t and c is the number of classes.

3.2.3.2 Random Forest

Random forests aggregate predictions from an ensemble of decision trees. The concept of "wisdom of the crowd" finds a remarkable representation in the Random Forest algorithm. Random Forests employ an ensemble learning method for classification and regression by constructing a multitude of decision trees at training time. The principle underlying the algorithm is the bootstrap sampling method, wherein each tree is trained on a random sample of data drawn with replacement, normally representing about two-thirds of the total data. This strategy ensures that each tree develops its predictive model based on a different sample, which introduces diversity in the model ensemble.

In addition to bootstrap sampling, Random Forests utilize feature bagging to select a random subset of features at each split point within a tree. When a Random Forest grows each tree, at every split in the tree, it does not consider all features. It selects a random subset of features and only considers these for splitting. This number is typically much smaller than the total number of features. This randomness has dual benefits: it mitigates the potential over-dominance of any single feature and promotes heterogeneity amongst the trees, enhancing the ensemble's resilience to overfitting. In contrast to single decision trees which are often pruned to prevent overfitting, trees within a Random Forest are allowed to grow to their maximum size, banking on the ensemble approach to average out idiosyncrasies and thus balance any individual tree's tendency to overfit.

Finally, prediction in Random Forests involves aggregating the decisions of all trees. For classification, this is typically done by majority voting, while for regression, the average of all tree predictions is calculated. Such an ensemble approach leverages the collective decision-making of multiple models, which tends to be more accurate and robust than individual models [80], [81].

In this model, Y represents the output we are trying to predict. We construct B decision trees, each based on a separate bootstrapped sample of the data, with each tree providing an output Y_b. For a classification task, we determine the final predicted outcome, Y_{pred}, by taking the mode of all the individual tree predictions $(Y_1, Y_2, \dots, Y_B)$. The mode is the most frequently occurring prediction across all B trees for a given input. For classification mode we would have:

$$Y_{pred} = mode\ (Y_1, Y_2, \dots, Y_B) \tag{3.15}$$

3.2.3.3 Gradient Boosted Machines (GBM)

GBMs, including tools like XGBoost and LightGBM, sequentially build decision trees where each tree corrects the predecessor's errors. The main parameters consist of learning rate, number of estimators, and tree depth. For imbalanced datasets, the scale weight parameter or under sampling methods are often favored [82]. In XGBoost, we define a parameter to scale up the class with fewer samples. Thus, we use scale weight, which is the ratio of the size of the largest class to the size of the smaller class, chosen as the weight to balance the data. The XGBoost algorithm's objective function is as follows:

$$F_{Obj}(\theta) = L(\theta) + \Omega(\theta) \tag{3.16}$$

L(θ) is a differentiable convex loss function that measures the discrepancy between the predicted value and the actual target. This function assesses the model's data fit. Common loss functions like mean square loss and logistic loss can be applied. The term Ω(θ), which includes the number of leaves in the tree (T), the learning rate (γ), and the leaf weights (w), acts as a regularization term to avoid overfitting.

The optimization process in XGBoost involves constructing a tree structure in each iteration that minimizes the objective function. This process is based on learning from the previous tree's outcomes and residuals. The objective function is allowing it to handle various loss functions efficiently. This relies on the first and second derivatives of the loss function at each data point, enhancing the optimization speed. XGBoost employs a greedy algorithm to iteratively find the optimal split point for each tree, aiming to minimize the objective function. It ranks data by feature values, then iteratively evaluates each feature as a potential split point by calculating the gain loss. The feature value offering the highest gain loss becomes the chosen split point. This process involves traversing all feature values and iterating through all samples to determine the most effective split [83]. In essence, XGBoost is a powerful and efficient algorithm that builds upon the principles of gradient boosting, adding features like regularization and optimized tree pruning to improve performance and prevent overfitting.

3.2.3.4 Long Short-Term Memory (LSTM)

LSTM, or Long Short-Term Memory, is a deep learning model, a variant of the Recurrent Neural Network (RNN) architecture. LSTM offers a key advantage over traditional RNNs: it effectively handles long data sequences without losing early information. This is due to the unique structure of LSTM cells, designed to overcome the limitations commonly faced by RNNs. Each LSTM cell includes additional parameters compared to a Neural Network and a system of gates that regulate information flow. The cell state, often referred to as the long-term memory, retains information from previous time steps within the cell. The current cell state (c(t)) is updated by combining the previous cell state (c(t−1)) with the output of the forget gate (f), and then integrating new information through the input gates (i) and the input modulation gate (g), in Equation (3.17).

$$C(t) = C(t-1) * f + g * i \tag{3.17}$$

The forget gate (f) decides what information from the past should be discarded. The input gate (i) and the input modulation gate (g) collectively determine what new information is added to the cell state. The output gate (o) decides what portion of the current state is outputted. Equations (3.18) – (4.21) detail the calculations for the forget gate, input gate, input modulation gate, and output gate. These equations involve the previous interval's output (h(t−1)), the current input (x(t)), and specific weights and biases (W_f, W_i, W_g, W_o, b_f, b_i, b_g, b_o) for each gate. The final output of the LSTM unit (h(t)) is given by Equation (3.22), which combines the output gate's result with the current cell state. The structure of an LSTM unit is illustrated in Figure 3.9, providing a visual representation of these processes and interactions within the cell [84].

$$f, i, g, o = \sigma\left(W_{f,i,g,o}\,[h(t-1), x(t)]\right) + b_{f,i,g,o} \tag{3.18 – 3.21}$$

$$h(t) = o * \tanh\,(c(t)) \tag{3.22}$$

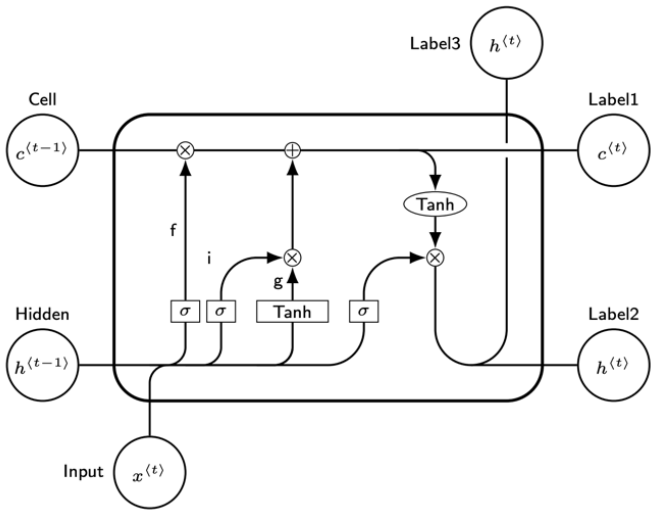


Figure 3.9 Schematic of LSTM model [85].

3.2.4 Evaluation Measures

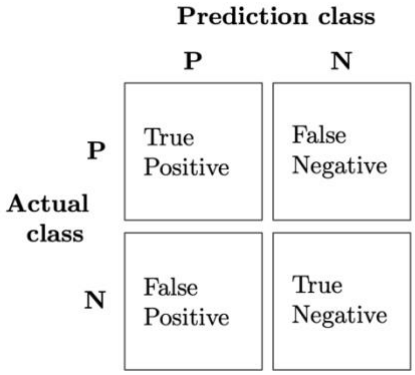
There are four main evaluation measure for calculating the performance of classification in machine learning namely, accuracy, precision, recall and F1 score. Accuracy is the total of correctly predicted data divided by the total data [86].

A confusion matrix is a table that shows the performance of a classification model. The matrix shows the actual target values with those predicted by the model, providing a comprehensive picture of its performance and shortcomings. Each element m_{ij} at row i and column j represent the number of instances from the actual class i.

TP_i (True Positive): This is the count of instances where the actual class is i and the model correctly predicts class i, found on the confusion matrix's diagonal at cell m_{ii}. True Negatives TN_i are all the instances that are correctly identified as not belonging to class i, calculated. TN_i is the sum of all cells that are true negatives for class i, accounting for correct predictions of class j where $i \neq j$, which are not part of the false positives or false negatives for class i.

FP_i (False Positive): This is the total count of instances where the model incorrectly predicts class i, while the actual class is not i. These are the off-diagonal elements in the column of class i.

FN_i (False Negative): This is the total count of instances where the actual class is i, but the model predicts a different class.



Precision $_i$ is the ratio of correctly predicted positive observations to the total predicted positives. (equation 3.23), It is a measure of the accuracy of the model in classifying a sample as positive for class i.

Recall $_i$ is the ratio of correctly predicted positive observations to all observations in actual class i (equation 3.24).

$F1_i$-score is a measure used to evaluate the performance of a classification model, and it is particularly beneficial when the portion of the class of i (equation 3.25).

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} \tag{3.23}$$

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i} \tag{3.24}$$

$$F1_i = 2 * \frac{\text{Precision}_i * \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{4.15}$$

The F1-score, a harmonized measure, maintains equilibrium between precision and recall, rendering it a valuable metric in scenarios characterized by imbalanced class distribution. In equation 3.26, we present the F1-weighted score to mitigate the imbalance in the dataset. Where n_i is the number of samples of class i in the dataset.

The macro average approach computes the metric for each class separately and then calculates their average. This method treats every class equally, irrespective of their frequency in the dataset. By assigning equal weight to each class, the macro average is particularly useful when the objective is to ensure equal representation and treatment of all classes, regardless of their size.

Macro precision is calculated by averaging the precision scores of all classes without weighing for their frequency in the dataset. Macro recall is calculated by averaging the recall scores of all classes, also without weighing for class frequency. The formula for macro precision and recall are:

$$\text{Macro Precision} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i+FP_i} \tag{3.27}$$

$$\text{Macro Recall} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i+FN_i} \tag{3.28}$$

Contrastingly, the weighted average takes the class frequency (the number of actual instances for each label) into consideration while averaging. It assigns greater significance to metrics from classes with a higher number of instances. This approach is more relevant when the aim is to emphasize classes that are more prevalent in the dataset.

3.2.5 Explainable Artificial Intelligence

While working with machine learning models, interpretation of the features is vital due to their intricate structure and the need for overt relationships between input features and predictions. To enhance interpretability, one can utilize methods such as LIME (Local Interpretable Model-agnostic Explanations) [87] and SHAP (SHapley Additive exPlanations) [90]. These techniques elucidate individual predictions and spotlight feature significance.

SHAP values provide a unified measure of feature importance by allocating an importance value for each feature for a particular prediction. It is based on a game theoretic approach to explain the output of any machine learning model. The core idea behind SHAP is the concept of a "Shapley value" which comes from cooperative game theory. The Shapley value allocates the prediction among the features by averaging the marginal contributions of each feature over all possible combinations [94].

1. Local explanations: SHAP can explain the model prediction for a specific instance.

2. Consistent: If we change the model such that it relies more on a particular feature for making predictions, then the SHAP value of that feature should not decrease.

3. Global explanations: Aggregated SHAP values can give a global perspective of the model.

$$f(x) = E[f(x)] + \text{SHAP Value} \tag{3.29}$$

In equation 3.29, f(x) represents the probability of a certain class given the input x, as predicted by the model. For a binary classification, this would typically be the probability of the positive class.

E[f(x)] is the expected value of the model's output, which, in classification, can be interpreted as the baseline probability of predicting the positive class. This baseline is calculated over the dataset and serves as a reference for understanding individual predictions.

SHAP values explain how each feature in the input contributes to the difference between the model's prediction for that specific input and the baseline prediction. Each SHAP value indicates the magnitude and direction (positive or negative) of a feature's impact on the probability of predicting a particular class.

Using SHAP values with Random Forest and Decision tree:

- When applied to tree-based models like Random Forest, SHAP uses TreeSHAP, a fast and exact method to estimate SHAP values for tree ensembles.
- After the best hyperparameters are chosen for the Random Forest and Decision tree, we can compute the SHAP values for each feature and each observation to understand how the features influence the model's predictions.

In summary, we provide a table that presents the advantages and disadvantages of each machine learning classification model in Table 3.1 [89], [90].

Table 3.1 Machine learning classification comparison.

Model	Advantages	Disadvantages
Decision Tree [83]	- Simple to understand and visualize. - Can handle both numerical and categorical data. - Inherently handles feature interactions.	- Prone to overfitting, especially on small datasets. - Biased towards dominant classes in unbalanced datasets
Random Forest [86]	- Can achieve higher accuracy than decision trees. - Less prone to overfitting due to the ensemble method. - Can handle class imbalance by adjusting class weights as a feature.	- Can be computationally intensive. - Predictive latency due to multiple trees.
LSTM [84]	- Effective in learning long-term dependencies in sequence data - Automatically learn and extract features	- Large sample size to prevent over fitting. - Struggle to effectively capture and learn from rare events (minority class)
XGBoost [83]	- Gradient boosting framework that can handle class imbalance. - Regularization helps in preventing overfitting.	- Requires careful tuning of hyperparameters. - Can be computationally intensive for large datasets.

CHAPTER 4 RESULTS

In section 4.1, we present the results of phone reorientation, which is our initial objective. The content of this section is developed based on section 3.1. In the following section 4.2, we will present the results from traffic event classification, the second objective, as outlined in section 3.2.

4.1 Phone Reorientation

As presented in Figure 4.1, data was collected from a personal iOS smartphone over a duration of 685 s. The experiment was conducted with the device moving at an average velocity of 20 m/s. To ensure the accuracy and reliability of the data, the smartphone was strategically positioned within a cup holder to negate extraneous noise and maintain a consistent orientation with respect to the vehicle – at an angle of 57 degrees to the plane x-y that starts from point A and ends at point B.

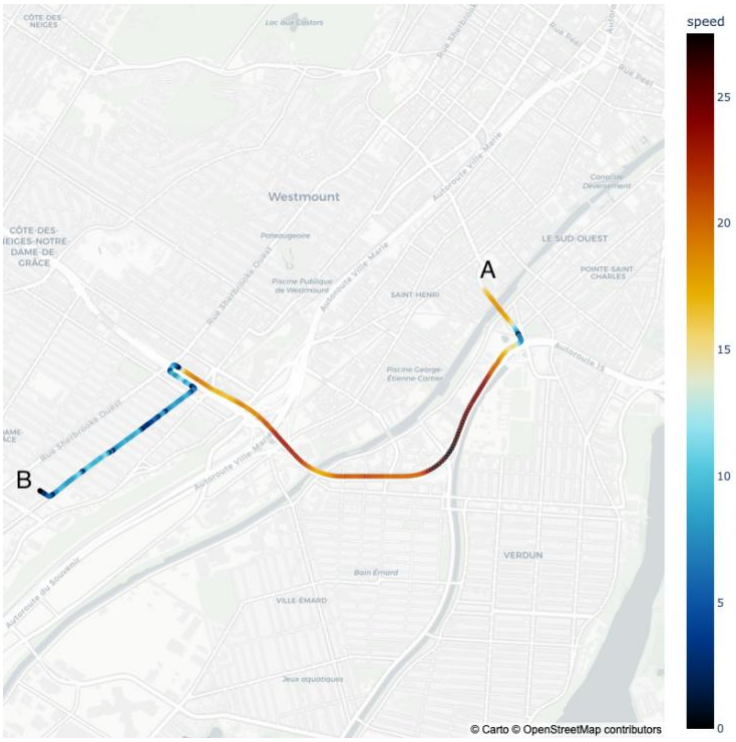
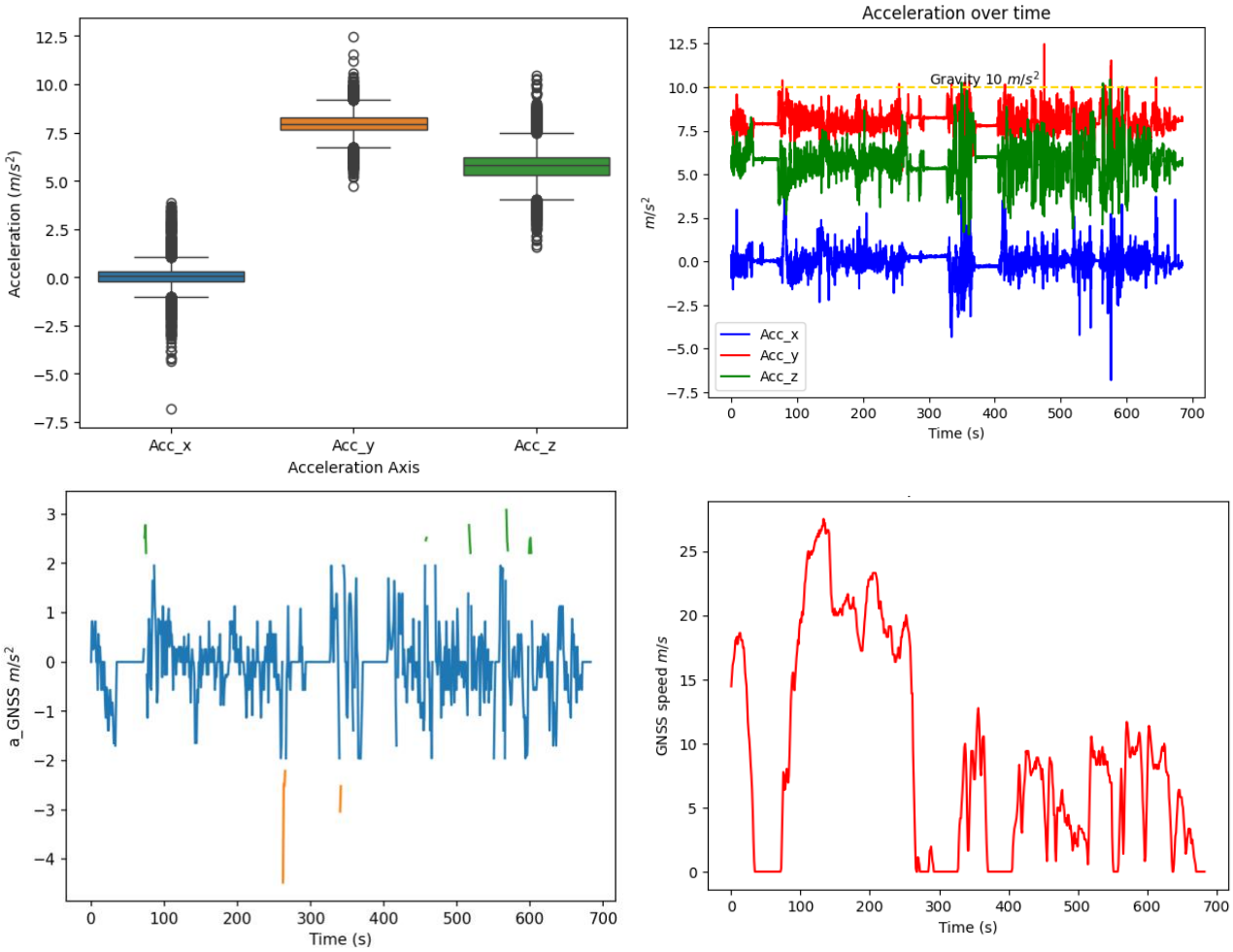


Figure 4.1 Map of the route passed by the vehicle.

Utilizing the sensors available through the iOS framework, a comprehensive dataset was acquired. The employed sensors encompassed the GNSS, gyroscope, accelerometer, and magnetometer. After the refinement of the algorithm, its efficacy will be put to the test in real-world reorientation scenarios.



(a) (b)

Figure 4.2 Before reorientation, the box plot and charts for Acc_x, Acc_y, Acc_z and a_{GNSS}.

Figure 4.2 illustrates the acceleration upon applying constraints and executing the reorientation algorithm. Braking events in the dataset are derived from GNSS data. For a comprehensive analysis, accelerometer data was employed, upon examining various trips and drivers. In Figure 4.2 (a) bottom, a consistently high and rapid oscillation pattern in the acceleration magnitude was observed, correlating with the a_{GNSS}. In Figure 4.2 (b), we can observe fluctuations at the bottom,

which represent variations in speed due to braking and acceleration. The standard deviation of the GNSS speed is recorded at 8.16, indicating variability in the speed data. Additionally, the high coefficient value of 0.91 supports the shape of the chart, suggesting a strong correlation between the observed data and the measured speed. We are looking for a sharp upward trend from 80 s to 150 s, the vehicle starts to move. In Figure 4.2 (b), on the top with three distinct colors, we can see the Acc_x, Acc_y, Acc_z. Figure 4.2 (a) top, represent boxplot of Acc_x, Acc_y, Acc_z, it is noticeable that gravity have not been remove and it influence two of directions Acc_y, Acc_z.

To align the accelerometer data with the car's direction, the algorithm 1-3 from section 3.1 was applied. This adjustment ensures that the accelerometer data mirrors the car accelerations during braking events, offering a reliable reference frame for analysis. The heading is derived from GNSS readings throughout the trip. The azimuths are the output of the reorientation algorithm. In the next step, using the heading and azimuth estimates, we can calculate the yaw. Considering the noise in the readings, the patterns of azimuth and heading are quite similar. Thus, the median of yaw is approximately, -5.88 degrees (See Figure 4.3 Estimated azimuth and heading.

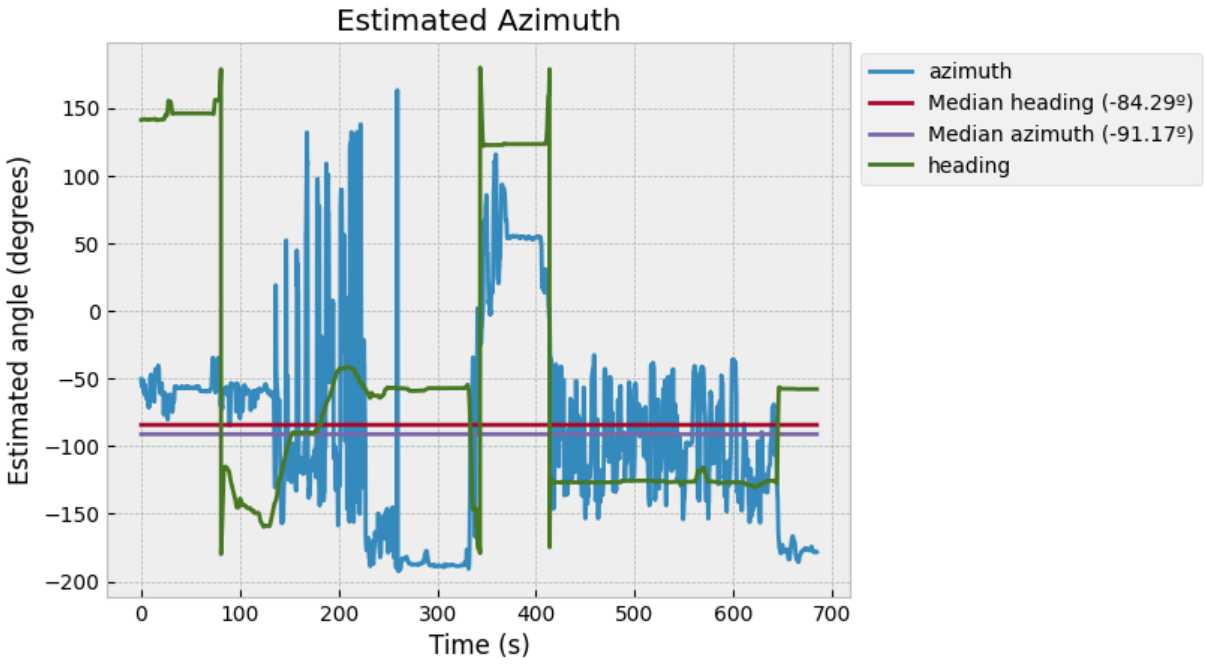


Figure 4.3 Estimated azimuth and heading.

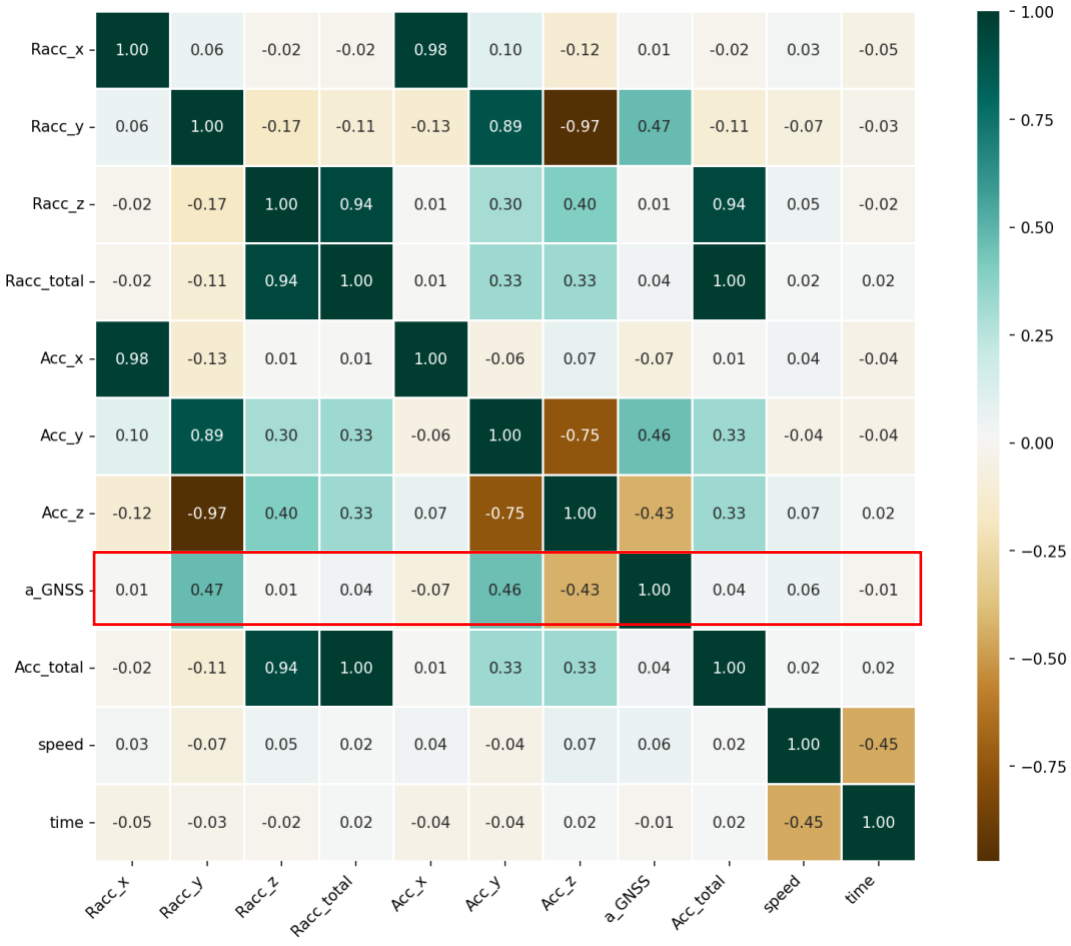


Figure 4.4 Correlation variables between raw sensor data and reoriented acceleration $Racc_x$, $Racc_y$, and $Racc_z$.

As presented in Figure 4.4, upon reorientation, it is evident that the adjusted acceleration only correlates with the y direction with 0.47. In the correlation heatmap, we can see a clear change in the relationship between a_{GNSS} and the vertical acceleration Acc_z. Before we adjusted the data, these two were somewhat inversely related with a correlation of -0.43, which meant they tended to move in opposite directions. However, after we used the algorithm to refine the data, their relationship became negligible, with a correlation of 0.01. This suggests that vertical movement does not really affect a_{GNSS} after we correct the data. As delineated in Figure 3.1, during vehicular braking, a pronounced acceleration along the Y-axis is anticipated.

4.1.1 Validation with traffic event

As previously discussed, one method to diagnose and validate the algorithm involves the utilization of traffic events. Within a controlled environment and given knowledge of GNSS speed during trips, two significant event periods are evident: firstly, between 0-100 seconds and secondly, between 200-400 seconds in Figure 4.5. During these intervals, both braking and acceleration events transpire. Our criteria for verifying traffic events incorporation of GNSS data and a_{GNSS} which means reading from GNSS determines the acceleration and deceleration. The objective is to determine whether, after reorienting the phone, the y-axis displays a pattern analogous to that of a_{GNSS}. This observation illustrates further underscores the strong correlation between these two variables.

As depicted in Figure 4.5 and Figure 4.6, the first plot represents the acceleration in the y direction. The graph showcases notable oscillations, with sharp peaks and troughs. Between the time intervals 0-200 and 800-1000 s from 15 to 0 $^{m}/_{s}$, and 5 to 20 $^{m}/_{s}$, we notice significant variations, implying instances of braking and acceleration. The plot flattens around the 200-800 s range and reaches zero, indicating a relatively stable acceleration phase. The second plot demonstrates the GNSS speed and a_{GNSS}. The speed presents an inverse bell-shaped curve, commencing with a steep decline, reaching a minimum around the 400 s mark, and then exhibiting a gradual increase. a_{GNSS}, contrastingly, follows a slightly ascending trajectory, showing a gradual change in speed during this period. The third plot visualizes the acceleration in the x and z directions. The x acceleration remains relatively constant, with minor perturbations, while the z acceleration indicates an abrupt spike around the 800 s mark, followed by a descent.

In conclusion, we can see a similar pattern in GNSS acceleration and acceleration y-axis after reorientation was applied. The acceleration and deceleration pattern depicted in the event data indicates a notable similarity between the y-direction acceleration Acc_y and a_{GNSS}, with a high correlation coefficient of 0.61. The extremely low p-value of 0.00159 further reinforces the strength of this relationship, highlighting the patterns' similarity. Furthermore, the Dynamic Time Warping (DTW) analysis yields a value of 7955, which suggests a certain level of alignment between the two datasets over time. According to Figure 4.4 and Figure 4.6, we conclude that the method is successful. We have applied a similar methodology on ten different trips with various phone positions and came to the same conclusion.

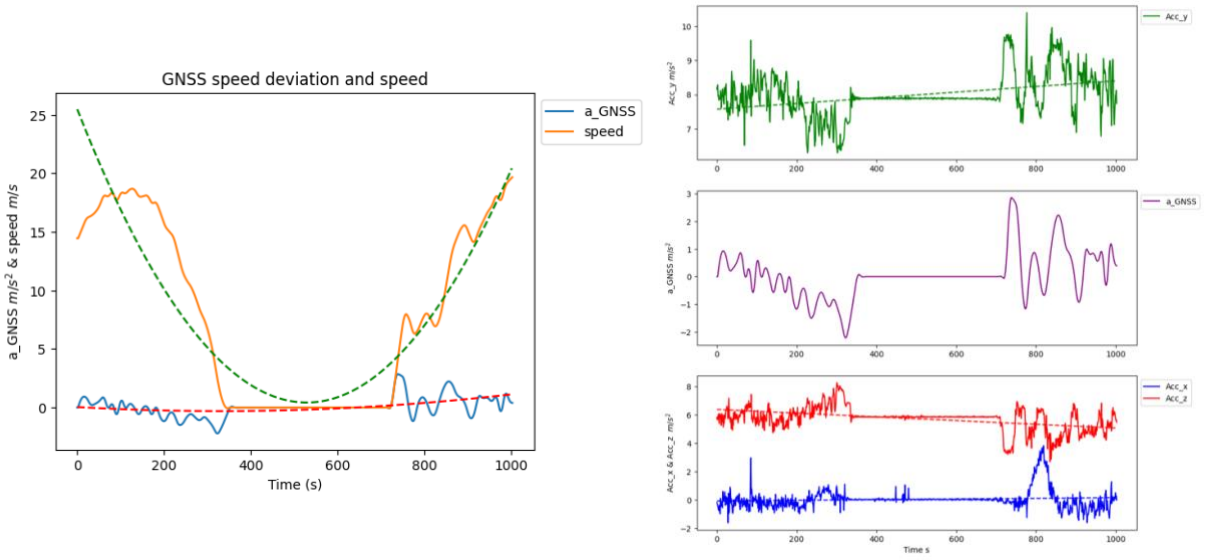
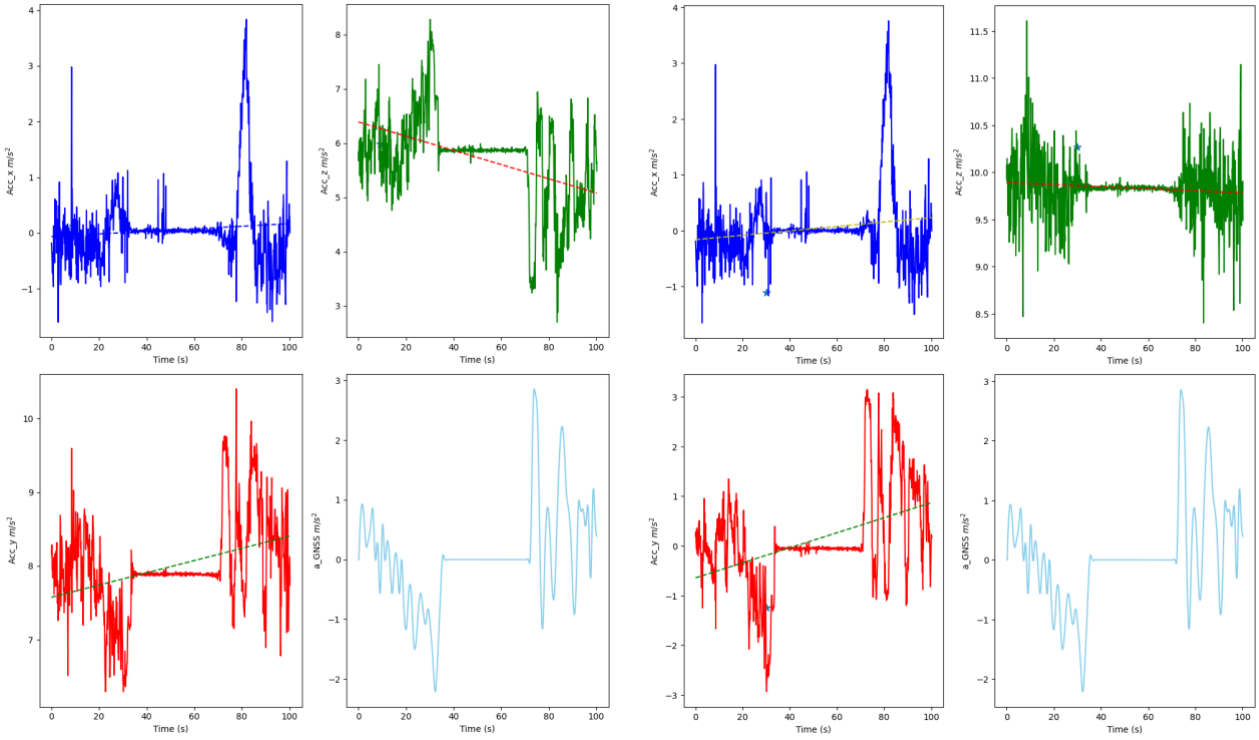


Figure 4.5 The speed and a_{GNSS} change over the braking event.



(a) Acceleration pattern before reorientation. (b) Acceleration pattern after reorientation.

Figure 4.6 Acceleration plot for before and after reorientation.

In summary, we undertook the collection of our own raw data, and all the tests were applied to ensure the model's functionality with varied data sources. It is noteworthy that these models were originally proposed by Google and written in Java [88]. We took the initiative to rewrite them in Python for our specific needs. Moreover, in 2016, an article was published that evaluated the accuracy of these reorientation models using real-world data [77]. We reached the same conclusion based our own data results. It is essential to highlight a pivotal constraint for these algorithms: the phone must remain in a stationary position for most of the trip.

4.1.2 Reorientation on UBI dataset

Challenges frequently arose when working with the real data collected by an insurance company. Although we applied similar algorithms 1-3 to reorient the phone, all were rigorously tested on collected trips, indicating the potential efficacy of the reorientation method. Unfortunately, there were unforeseen complications. Within the data gathered from the insurance application, external variables such as the car model and phone operating system introduced unexpected complexity and noise. Central to our difficulties was with pre-processed data. Their data processing presented a black box issue. While they asserted that they had rotated the phone such that the z-axis pointed upwards, problems persisted with the other axes; the orientations of x and y remained indeterminate. In the absence of ground truth and labeled data, it became imperative to devise a strategy to identify validation points.

Three distinct validation methods were employed to ascertain model functionality. Initially, we estimated a_{GNSS}. Theoretically, a_{GNSS} should exhibit acceleration in the direction of the vehicle; thus, we can investigate the reoriented acceleration correlation with a_{GNSS} to make sure the phone to align with one of its axes. Secondly, if gravity was not factored out from the acceleration, the acceleration in the z-axis should approximate 10 m/s^2. Our final validation criterion involved examining specific acceleration and deceleration patterns using GNSS speed. During instances of stationary periods when vehicle is not moving or rapid acceleration, we anticipated observing a corresponding pattern in the y-axis. If the data, post reorientation algorithm application, met all these criteria, it was deemed to have been successfully modified.

One intrinsic issue with the insurance data is its prior manipulation, which manipulates of its unbiased positioning because they have been reoriented once. With gravity removed and the position of the z-axis stabilized, the z-axis could not be employed as a validator. This left us with a 2D rotation challenge. Further complicating matters, the values reported for other sensors were also affected. Our primary benchmark for calculating yaw rotations in the x and y axes was the magnetometer reading. Naturally, Earth's magnetic density and power can only generate values ranging from 25 to 65 $^{\mu T}/_{s}$. With each phone rotation, these readings are expected to change due to orientation relative to the North Pole. However, the data from the insurance company consistently revealed a magnetic power substantially above the 65 $^{\mu T}/_{s}$ threshold, often averaging between 100 to 200 $^{\mu T}/_{s}$ (see Figure 4.7). The peak observed in the readings of the magnetometer may result from interference caused by other magnetic field sources, such as the magnets commonly found in car cup holders.

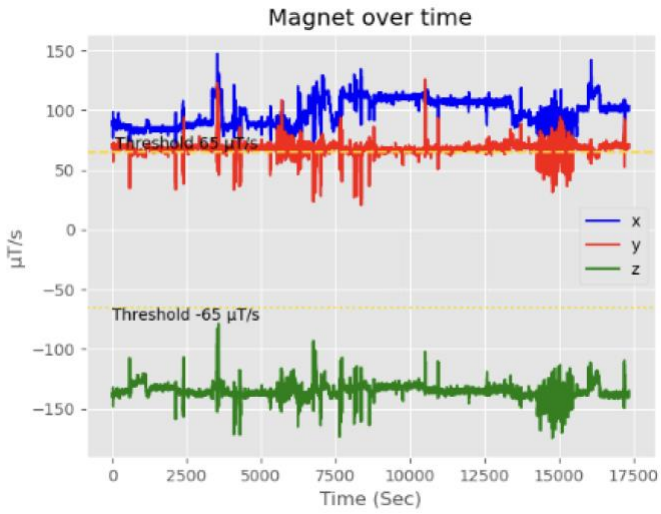


Figure 4.7 Magnetometer reading from Insurance data.

Despite these challenges, we made multiple refinements to the data. As an example, we present data from one of the trips extracted from insurance data. This particular trip occurred in Montreal and spanned 5.5 h This is the same dataset utilized for traffic event classification, which is explained in more detail in Dataset for Classification.

The charts and results derived from the UBI data, shown in Figure 4.8 and Figure 4.7, highlight these adaptations. Discrepancies such as anomalously high magnetometer readings—well above the expected 65 μT/s mark—and a low correlation between GNSS acceleration and reoriented data

(a maximum of 0.23) suggested that our findings might differ from those based on raw, unmanipulated data. These and other issues faced when working with UBI data underscore the complexity of applying reorientation techniques and affirm the need for careful validation in data-driven studies.

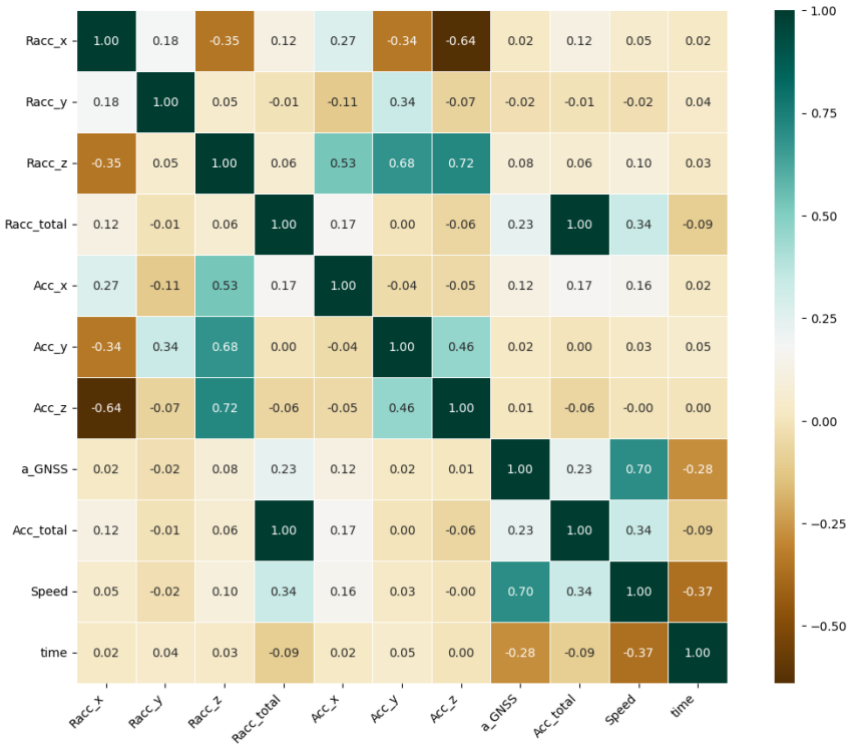


Figure 4.8 Results after reorientation from Insurance data.

4.2 Traffic Event Classification

Hyperparameter tuning is a crucial step in the design of machine learning algorithms. It is possible to improve the performance of a model by selecting optimal settings for the hyperparameters. This document presents an in-depth analysis of hyperparameter tuning for Decision tree, XGBoost and Random Forest classifiers using the GridSearchCV approach. Grid Search Cross-Validation (GridSearchCV) is a systematic way to explore multiple combinations of hyperparameter values. It trains an algorithm with each combination of hyperparameters in the specified grid and uses cross-validation to evaluate the performance of each model [89]. We utilized 5 folds for cross-validation on the same dataset described on subsection 3.2.2.

Moreover, we define a scorer leveraging F1 score with a weighted average to accommodate potential class imbalances. We compute the metrics for each individual label and then calculate their average, weighted according to the support, which is the number of actual instances for each label. This measure has been utilized for all presented models and explained in more details in subsection 3.2.4.

After iterating through all combinations, the best model is selected based on the specified performance metric. We selected the models based on two evaluation metrics: the F1 weighted score and macro average. These measures are better suited for use with imbalanced datasets. Considering that the majority vote — which is the most common label or outcome in the data, chosen randomly if the model does not learn — for this dataset stands at 99.3%, it is essential to direct our evaluation towards the F1 weighted score and perform a macro analysis, with a special emphasis on the minority classes. This approach ensures that we are not just measuring the performance based on the predominant class but also considering the accuracy and recall of the less represented classes.

4.2.1 Decision Tree

We first run the base model of the decision tree on this dataset. The base model has the following attributes: depth to none (nodes are then expanded until either every leaf is homogeneous or until each leaf has fewer samples than the specified minimum number needed to split), and max features are equal to the number of features equals to 4 for this model result presented in Table 4.1. As can be inferred from Table 4.1, the model was capable of fitting to the dataset and identifying the pattern of each label with good performance. As mentioned in the previous section, our evaluation metric for comparing models is the F1 score, and the macro average for this model is 0.81. This model has understood the trend for normal driving, but it struggles to accurately identify deceleration and acceleration patterns. The grid search technique has been employed here to find the optimal hyperparameters for the decision tree model. The model achieved an accuracy score of approximately 0.9958 using these parameters including, max depth, min sample spilt, min sample leaf, max feature, and class weight. The following hyper-parameters are considered for decision tree models.

Table 4.1 Result for base Decision tree model.

	Precision	**Recall**	**F1 score**	**Observation**
Normal driving	1.00	1.00	1.00	23899
Acceleration	0.62	0.81	0.70	72
Deceleration	0.67	0.81	0.73	75
Macro average	0.76	0.87	0.81	24046

- Max depth: None, 10, 20, 30, **40**
- Min samples split: **2**, 5, 10
- Min samples leaf: **1**, 2, 4
- Max features: auto, sqrt, **log2**
- Class weight: **balanced**, none

1. Class weight: This configuration parameter is critical when working with datasets that exhibit class imbalance, meaning some classes are underrepresented compared to others. By setting the class weight to balanced, the algorithm dynamically assigns a higher weight to the underrepresented or minority classes, inversely proportional to their frequencies in the input data. This adjustment compels the model to pay more attention to the minority class during training, thereby fostering a more balanced and fair approach in the learning process. If class weights are not balanced, the model may bias its predictions toward the majority class. To counter this, the class weight parameter can be modified to ensure that the tree learning algorithm compensates for class imbalance by treating instances of the minority class as more significant, which is reflected in the weight adjustment formula as shown in equation 4.1. In summary, model class weight balanced, means that the number of samples from each class are equals in depth zero.

$$\text{Weight}_i = \frac{\text{Total number of samples}}{\text{The number of sample in class i} * \text{number of class}} \quad (4.1)$$

2. Max depth: It specifies the maximum depth of the tree. A tree depth of 40 means the model can make up to 40 consecutive decisions (or have 40 levels) based on the input features. A deeper tree can capture more intricate patterns but can also lead to overfitting if not monitored.

3. Max features: The number of features to consider when looking for the best split. When set to log2, the model will consider the base 2 logarithm of the number of features at each split. This can speed up training and add a bit of randomness which might help in generalization.

4. Min samples leaf: The minimum number of samples required to be at a leaf node. This means that each leaf (the end nodes of the tree where decisions are made) must have at least one sample. This is the smallest value you can have for this hyperparameter, ensuring that the tree can create leaves even for very small groups of samples.

5. Min samples split: The minimum number of samples required to split an internal node. An internal node will split if it has at least two samples. This is also the smallest possible value for this hyperparameter. It ensures maximum flexibility in splitting nodes but, if not coupled with other regularization hyperparameters, it can lead to overfitting.

Table 4.2 Decision tree best model

	Precision	**Recall**	**F1 score**	**Observation**
Normal driving	1.00	1.00	1.00	23899
Acceleration	0.76	0.81	0.78	72
Deceleration	0.84	0.81	0.82	75
Macro average	0.87	0.87	0.87	24046

As illustrated in Table 4.2 and Figure 4.9, the decision tree model demonstrates a high degree of accuracy in classifying acceleration and deceleration events, achieving an F1 score of 78% for acceleration and 82% for deceleration: it correctly labeled 58 instances of acceleration and 61 instances of deceleration without any mislabeling. This performance indicates a high degree of better fitting model for acceleration and deceleration, which is our objective, even with a relatively small sample size. This effectiveness in classification underscores the decision tree model's capability to discern the nuanced differences between various driving behaviors. The high F1 scores reflect its precision and recall balance.

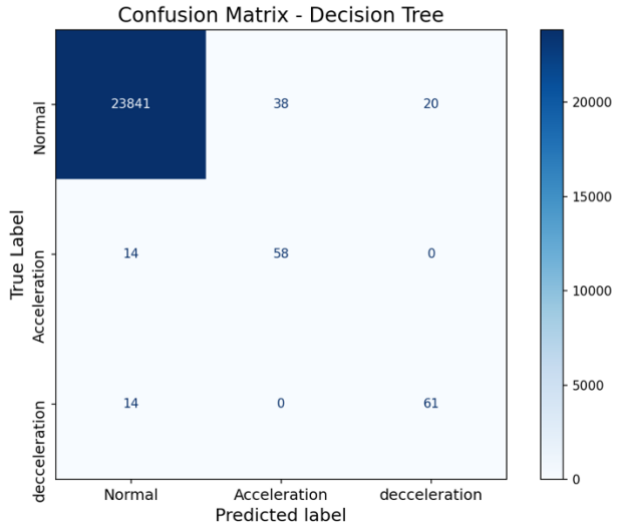


Figure 4.9 Confusion matrix for the hyper-tuned Decision Tree model.

4.2.2 Random Forest

As depicted in Table 4.3, we used the default settings of the Random Forest, which include 100 estimators, the Gini criterion, and a max depth of zero (the same principle as decision tree applies here too). Interestingly, by only changing the weight from none to balanced, the macro average improved from 81% to 85%.

Table 4.3 Result random forest model.

	Precision	**Recall**	**F1 score**	**Observation**
Normal driving	1.00	1.00	1.00	23899
Acceleration	0.68	0.79	0.75	72
Deceleration	0.72	0.82	0.70	75
Macro average	0.77	0.86	0.85	24046

This analysis aims to find the optimal hyperparameters for the Random Forest algorithm as provided and offer insights into their selection. The parameter grid for this model is stated below with the best parameters shown in bold.

1. Number of estimators: 10, 50, 100, **200**
2. Max features: auto, sqrt, **log2**
3. Max depth tree: None, 10, 20, **30**, 40
4. Min samples split: **2**, 5, 10
5. Min samples leaf: **1**, 2, 4
6. bootstrap: **True**, False
7. Class weight: **balanced**, None

In the following, each hyper-parameter is explained, including how it will affect the model.

1. Bootstrap: True;

By setting bootstrap to True, each tree in the ensemble is trained on a different subset of data. This bootstrapping introduces diversity among the trees, ensuring that the overall model is less likely to overfit to the training data. It also allows the model to capture a wider range of patterns and correlations in the data, contributing to its overall robustness.

2. Class weight: balanced; it adheres to a rule similar to the one explained above for decision tree.

3. Max depth: 30; A deeper tree captures more detailed patterns and correlations in the data. With a max depth of 30, the trees in the forest are allowed to grow deep. This might indicate the presence of complex interactions in the dataset that shallow trees might miss. However, care must be taken as very deep trees might risk overfitting, but when used in a Random Forest setting, this risk is often mitigated by the ensemble nature of the model.

In Figure 4.10, The influence of the maximum depth parameter on the F1 score has been analyzed, and it can be deduced that beyond a maximum depth of 20, the score remains constant. The main reason we use grid search is that, in this method, we might encounter a plateau in one parameter, but the effect of this parameter on others is also important.

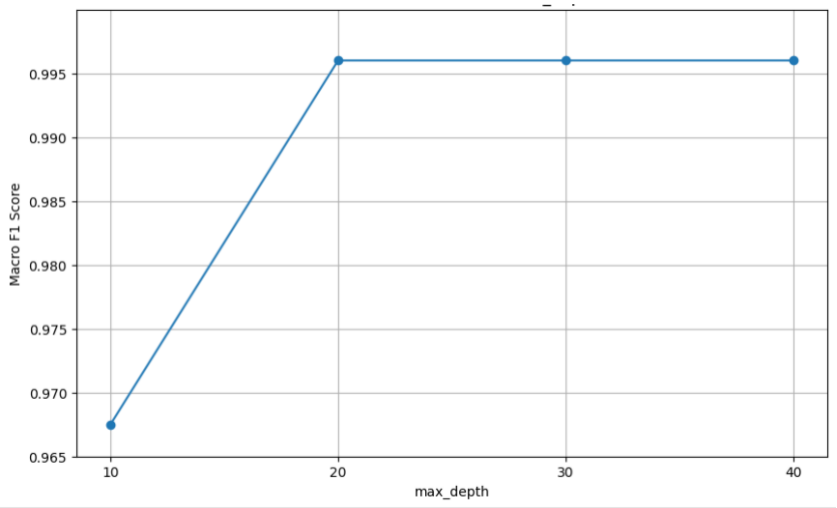


Figure 4.10 Max depth with macro average F1 for Random Forest.

4. Min samples leaf: 1 and min samples split: 2

Setting min samples leaf to 1 and min samples split to 2; it adheres to a rule similar to the one explained above for decision trees. However, in the context of Random Forest, the ensemble nature can counteract some of this risk.

5. Number of estimators: 200

A value of 200 suggests a comprehensive model that harnesses the power of multiple decision trees for a robust aggregated output. This parameter controls the number of trees randomly created for the Random Forest model; the default value is 100. As shown in Figure 4.11, The analysis indicates an increase in the F1 score as the number of estimators in the model increases from 10 to 50. After this initial increase, the F1 score continues to rise, albeit at a more gradual pace, displaying a smooth upward trend with the increase in the number of estimators. As the number of trees continues to grow, the incremental improvements in the F1 score suggest diminishing returns, yet the overall trend still points towards better performance with more estimators.

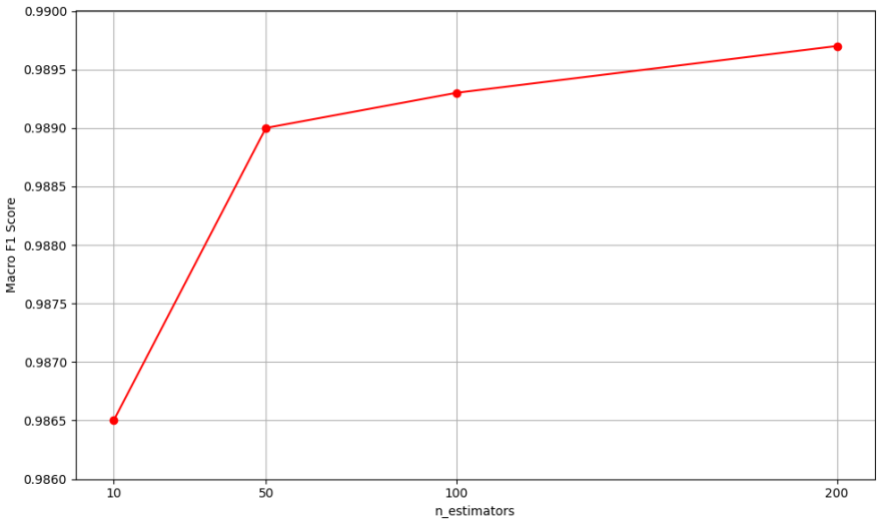


Figure 4.11 Number of the estimator for Random Forest

Utilizing a Random Forest classifier, we trained the model on the designated training dataset and subsequently assessed its efficacy on the test data. Upon an examination of the model's outcomes, several interpretations can be drawn in Table 4.4; the model demonstrates proficiency in predicting normal driving. The model's predictions are undeniably precise for accelerations (boasting a Precision of 1.00). However, it encompasses merely 81% of the actual braking occurrences (as indicated by a Recall of 0.81).

Table 4.4 Result of the best parameters of Random Forest.

	Precision	**Recall**	**F1-score**	**Support**
Normal Driving	1.00	1.00	1.00	23899
Acceleration	1.00	0.81	0.89	72
Deceleration	1.00	0.81	0.90	75
Macro Average	1.00	0.87	0.91	24045

In Figure 4.12, the confusion matrix is presented. The random forest model perfectly identifies normal driving, and the true positives for acceleration and deceleration are higher compared to those of the decision tree model. However, when the model evaluates acceleration, it accurately

identified 58 instances but misclassified 14 of them as normal driving. Similarly, for the deceleration category, the model correctly discerned 61 instances, yet erroneously classified 14.

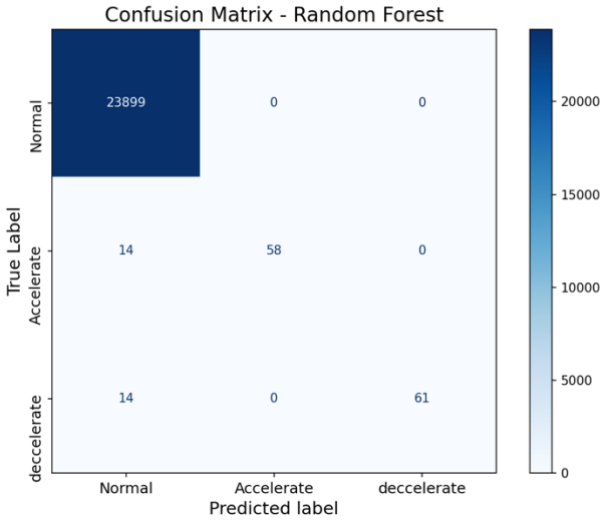


Figure 4.12 Confusion matrix for Random Forest model.

In relation to acceleration, the classifier's performance mirrors that observed for deceleration, accurately discerning 89-90% of acceleration instances (F1 score). In summation, while the Random Forest classifier excels in discerning the normal state with impeccable precision, it occasionally falters in its identification of deceleration and acceleration events, sporadically mislabeling them as normal.

4.2.3 Gradient Boosted Machines

For this analysis, we have used the following parameters to find the best combination. It is worth mentioning that both grid search and random search techniques have been employed on the XGBoost model. As previously stated, this model is more computationally intensive, requiring a significant amount of time for each model iteration. On average, it takes between 23 to 28 minutes per model fit. Due to this computational expense, we adopted a combination of grid and random search approaches. We utilized grid search to optimize primary hyperparameters such as learning rate, number of estimators, and max depth. Subsequently, the random search was employed to tune the remaining hyperparameters. This modification of the macro approach adjusts for label imbalance and may lead to an F-score that does not necessarily fall between precision and recall.
In Table 4.5 the result for XGBoosted model with the following parameters are shown, number of estimators of 100, learning rate of 0.1, subsample size of 1.0, and Colsample bytree of 1.0.

Table 4.5 Result for XGBoosted model.

	Precision	**Recall**	**F1 score**	**Observation**
Normal driving	1.00	1.00	1.00	23899
Acceleration	0.86	0.08	0.15	72
Deceleration	0.00	0.00	0.00	75
Macro average	0.62	0.32	0.38	24046

Hyperparameters explored for hyper parameter tuning as follows:

- Number of estimators: 50, 100, **200**
- Max depth of tree: 3, 6, **10**
- Learning rate: 0.001, 0.01, **0.1**
- Subsample size: 0.5, **0.8**, 1.0
- Colsample bytree: 0.5, 0.8, **1.0**

1. Learning rate: 0.1; The learning rate, also known as the step size or shrinkage, determines the contribution of each tree to the final prediction. A smaller learning rate signifies that each tree's contribution is diminished, thus requiring more trees (number of estimators) to achieve comparable performance. While this might yield more robust models by giving less importance to individual trees, it can be computationally costly. Conversely, a larger learning rate may result in a model that learns quickly but risks overfitting. A value of 0.1 is a middle ground, balancing between model robustness and training speed. This choice often provides a good trade-off between preventing overfitting and ensuring a decent model performance [90].

2. Max depth: 10; The depth of the trees influences the model's complexity. A max depth of 10 suggests moderately deep trees, which allow the model to capture nuanced patterns in the data without severely risking overfitting [91].

3. Number of estimators: 200; More trees typically lead to better model performance as the model gets more chances to correct its mistakes from prior rounds. However, there's a point of diminishing returns. Beyond a certain number of trees, the model might not gain significant improvements, and it might even begin to overfit. Moreover, computational costs and training time increase. A value of 200 indicates a preference for capturing complex patterns in the dataset and an investment in

computational resources to achieve a high model performance. As Table 4.6 and Figure 4.13, XGBoosted is not a proper model to detect the pattern of deceleration.

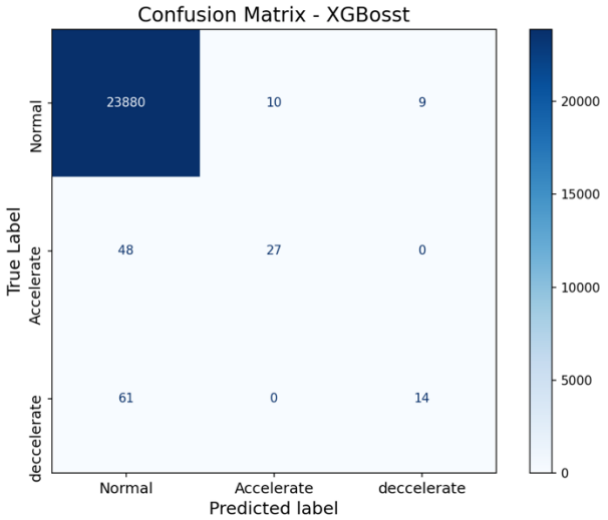


Figure 4.13 Confusion matrix for XGBoosted model.

As can be seen from Figure 4.13 and Table 4.6, the XGBoost model demonstrates exceptional performance in identifying normal driving conditions, achieving perfect precision and recall. However, it struggles significantly with recognizing acceleration and deceleration events. This difficulty is evidenced by a high number of false negatives for both acceleration and deceleration, indicating that the model frequently mislabels these driving behaviors.

The F1 score for deceleration, standing at 0.31, and for acceleration, at 0.55, further underscore the model's inadequacy for this dataset. Despite its flawless detection of normal driving conditions, the model's inability to accurately identify acceleration and deceleration events suggests it is not suitable for this specific dataset.

Table 4.6 Result for best XGBoosted model.

	Precision	**Recall**	**F1 score**	**Observation**
Normal driving	1.00	1.00	1.00	23899
Acceleration	1.00	0.38	0.55	72
Deceleration	1.00	0.19	0.31	75
Macro average	1.00	0.52	0.62	24046

4.2.4 Long Short-Term Memory

In our final experiment, we aimed to evaluate a model more adept at handling time series data. To this end, we applied the same dataset, used initially for classification problems, to a time series model. Interestingly, during this process, the model encountered difficulties in identifying the optimal loss function. The most favorable results were achieved around the 40-epoch mark. For this purpose, we utilized a Long Short-Term Memory (LSTM) model in PyTorch, which offers several adjustable parameters to fine-tune its performance. These parameters include number of layers, which determines the number of recurrent layers; batch size, which specifies the format of input and output tensors; dropout, defining the dropout probability to prevent overfitting; and bidirectional, which indicates whether the LSTM should be bidirectional. To optimize the LSTM model, I conducted a grid search focusing on three key parameters: the dropout rate (dropout), the batch size (batch size), and the number of units in each layer (units). This approach was chosen to systematically explore various combinations of these parameters, aiming to enhance the model's ability to process our time series data effectively.

The number of LSTM units directly affects the model's ability to capture information in sequences. More units can enable the model to learn complex patterns, but too many can lead to overfitting. Batch size impacts how the model updates its weights during training. Smaller batches often lead to more noise in the updating process, which can contribute to better generalization. Larger batches provide a more accurate estimate of the gradient but can also lead to overfitting if not managed correctly. The number of epochs determines how many times the model will work through the entire dataset. More epochs give the model more opportunities to learn and adjust its weights, but also run the risk of overfitting if too many are used. Dropout rates offer a method of regularization by randomly deactivating a portion of the network's neurons during training, which helps to prevent co-adaptation of neurons and thus overfitting.

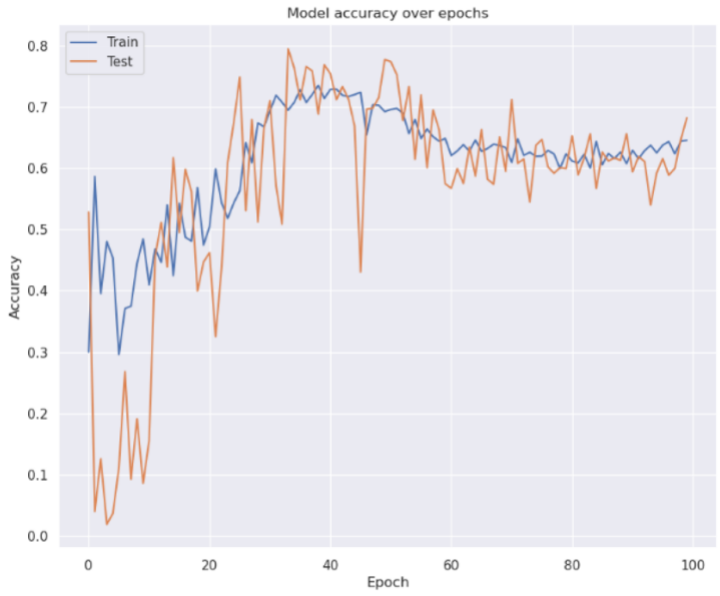


Figure 4.14 Accuracy of train and test for LSTM model.

A comprehensive hyperparameter tuning process was undertaken for the LSTM model to ensure its optimal performance on the dataset. The tested hyperparameters encompassed a range of values across four key dimensions:

- LSTM Units: The model was tested with three different settings for the number of LSTM units - 50, 100, and 150. This range was selected to explore the effects of varying model complexity, from a relatively simpler model with 50 units to a more complex one with 150 units.
- Batch Sizes: Batch sizes of 32, 64, and 128 were included in the tests. These sizes were chosen to understand how different levels of data batching affect the model's learning process, from smaller batches potentially offering more robust convergence to larger batches aimed at enhancing computational efficiency.
- Epochs: The number of training epochs was varied between 50 and 100.
- Dropout Rates: Dropout rates tested were 0.0, 0.2, and 0.5. These rates were selected to evaluate the impact of different levels of regularization on the model's ability to generalize, thereby preventing overfitting.

- Time steps: In the methodology for our time series analysis, we tested the input shape by varying the number of time instants from 5, 15 and 30 to assist the model in learning the pattern. We determined that a time step of 15 with a 50% overlap between successive steps provides the best results.

In this study, the optimal performance of the LSTM model was achieved with a specific configuration of hyperparameters. The best-performing model featured 50 units, indicating a moderate level of complexity that effectively captured the data's patterns without overfitting. A dropout rate of 0.5 was employed, providing a substantial level of regularization to prevent overfitting, a common challenge in neural network training. The chosen batch size was 64, striking a balance between computational efficiency and the benefits of stochastic gradient descent. Finally, the model was trained for approximately 50 epochs, a duration that was found to be sufficient for the model to converge to its highest accuracy without the drawbacks of excessive training. This configuration of hyperparameters demonstrates a carefully balanced approach, optimizing the LSTM model for the specific characteristics and challenges of the dataset in question. The best LSTM model has reached a 71% macro F1 score and accuracy of 83%. We utilize a time step of 15 with a 50% overlap, and the presence of minority classes penalizes the model due to the reduced sample size. This model should be tested on a larger sample size to capture the patterns of traffic events more effectively.

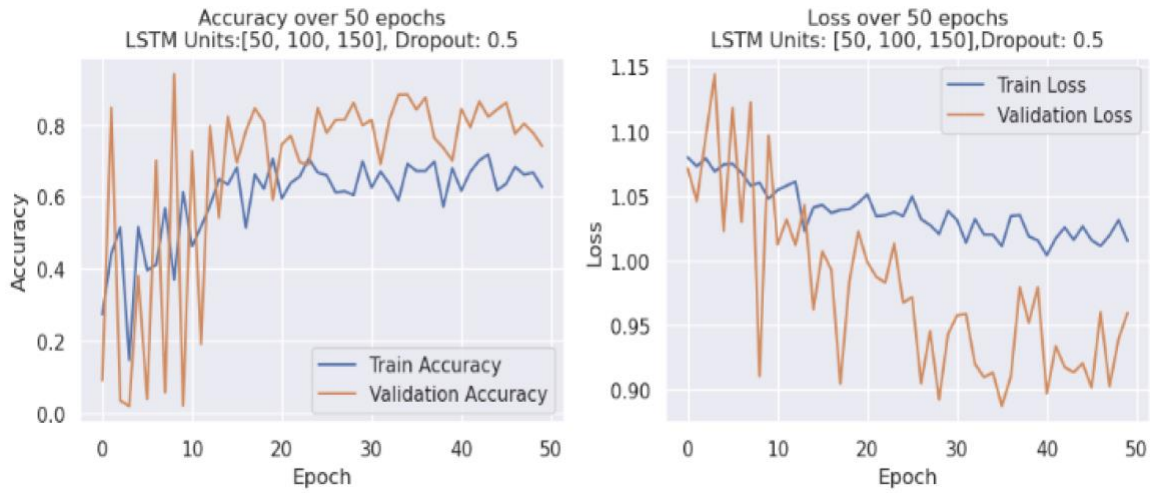


Figure 4.15 The grid search result for LSTM model.

Finally, we present the classification report for each class to examine the F1 score and macro average for each label. As shown in Table 4.7, it even struggles to predict normal driving with a 75% F1 score, which is the lowest among the four models. A macro-average of 0.28 indicates that if we increase the sample size, we could see an improvement from nearly 0 to 0.28 in predicting acceleration and braking. Generally, this model is not a suitable choice for identifying traffic events. The underperformance of the LSTM model may be largely attributed to the traffic events lacking clear temporal patterns or possessing temporal dependencies that exceed the LSTM's capability to capture effectively. Consequently, the model struggles to make accurate predictions.

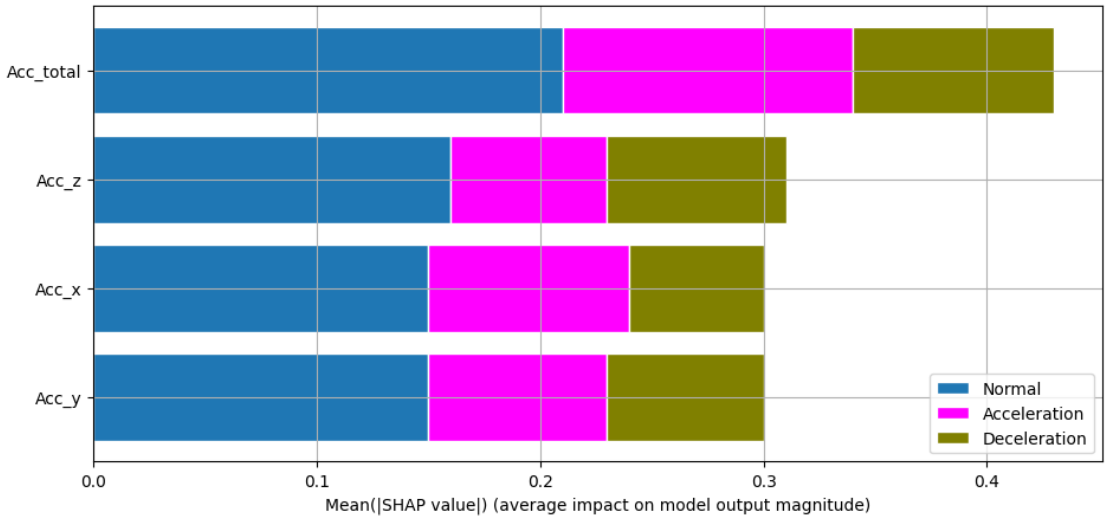
Table 4.7 Result for the LSTM model.

	Precision	**Recall**	**F1 score**	**Observation**
Normal driving	0.99	0.78	0.75	3413
Acceleration	0.3	0.32	0.24	9
Deceleration	0.25	0.44	0.53	9
Macro average	0.83	0.55	0.71	3434

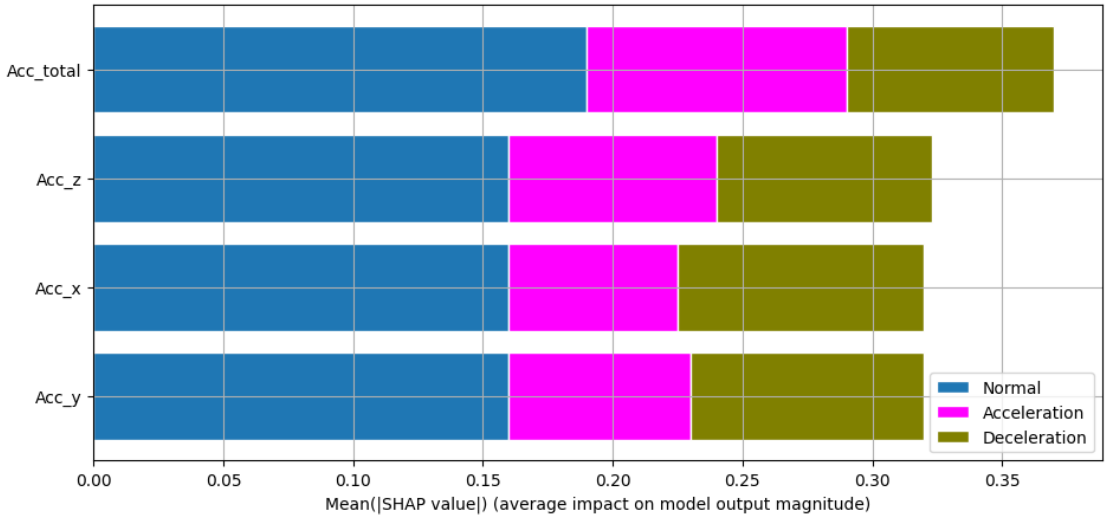
4.2.5 Explainable Artificial Intelligence

Explainable artificial intelligence guide researcher to better understand the influence of each feature on the machine learning model.

Positive SHAP values indicate that the presence of a feature pushes the model output higher, while negative values push the output lower. The color of the bars indicates the class to which the SHAP value belongs. Figure 4.16 indicates the SHAP values of all features of the test set. Each bar represents a specific feature's SHAP value for the test set. The position on the x-axis indicates the magnitude and direction of the feature's impact on the prediction [87].



(a)



(b)

Figure 4.16 Result based on their significance on Decision Tree (a) and Random Forest (b).

The features Acc_{Total}, Acc_x, Acc_z, and Acc_y have varying importance for the three classes (Normal driving, Acceleration, and Deceleration). The Figure 4.16 delineates the significant role Acc_{Total} plays in determining the appropriate class with approximately 19% for Random Forest and 35% for decision tree magnitude value. Interestingly, other axes of the sensor contribute equitably to each class's definition. In order to test our hypothesis, we create a new feature for total acceleration x and y axis. The SHAP value for this has higher significant compared to Acc_x, Acc_z, and Acc_y by 15 %.

A noteworthy observation is the potential enhancement in classification accuracy if the sensor is aligned with the vehicle's direction. Currently, the z-direction acceleration shows the correct alignment to the z direction; we test a summation of x and y, with a similar high SHAP value seen for total acceleration in the x and y directions. Looking ahead, if advancements permit the alignment of the sensor with the vehicle's orientation, we can foresee a more pronounced correlation and potentially improved classification outcomes.

4.2.6 Scalability

Working with large datasets, there are several challenges and benefits to consider. On the challenge side, the main issues involve the increased need for computing power and data storage. Large datasets can be computationally expensive to process and analyze, requiring advanced infrastructure, often in the form of cloud computing resources. Additionally, there is a risk of introducing noise and irrelevant information, which can lead to a model that performs well on training data but poorly on new, unseen data, a problem known as overfitting.

On the opportunity side, large datasets can provide a more comprehensive foundation for training machine learning models. With more data, models can identify and learn from a wider range of examples, which can improve their ability to recognize patterns and make accurate predictions. This is particularly true in scenarios where the model needs to learn driver behaviors such as deceleration and acceleration patterns.

In the specific study mentioned, a dataset of approximately four million instances is used to assess the impact of dataset size on the accuracy of a machine learning model. The steps followed in processing and analyzing this data are the same as those previously mentioned, ensuring consistency in the methodology while scaling up to handle the larger dataset.

Given the high computational demands of running models on large datasets, we have applied the hyperparameter tuning results from the previous section as a foundational step. This approach allows us to leverage insights gained from prior tuning to optimize our model's performance without incurring excessive computational costs [92].

Furthermore, it is crucial to ensure that our model is robust and generalizable. To achieve this, we report performance using cross-validation techniques. Cross-validation helps in assessing the model's effectiveness on different subsets of the dataset, which reduces the risk of overfitting and

verifies that the data maintains a consistent distribution. By doing so, we strive to maintain the reliability of the model across the entire dataset, affirming its predictive stability and accuracy.

This dataset approximately 4 million instances (aggregation of 100 trips) are used: the challenge will be handling this volume efficiently, ensuring that the computational resources namely cloud computing power and paralleled model Spark are effectively utilized. The opportunity is that with such a sizable pool of data, the models have a robust training ground to improve their prediction accuracy regarding braking and acceleration patterns. The identical steps mentioned previously would need to be scaled to accommodate this larger dataset, ensuring that the model's performance is improved for accuracy while managing the model training time.

The confusion matrices and classification reports for the Decision Tree and Random Forest models reveal several insights into the performance of these classification algorithms on the dataset. It can be inferred from Figure 4.17 and Table 4.8, that the decision tree's evaluation measures, such as precision, recall, and F1-score for normal driving, are near perfect (0.99). However, a more informative metric for assessing the model's performance is its ability to predict the less represented classes, specifically acceleration and deceleration.

In evaluating the performance for acceleration and braking, the decision tree model yields high scores, with both precision and recall at 0.97. This indicates a strong ability to correctly identify these specific driving behaviors, despite them being less represented in the dataset. The overall accuracy of the decision Tree model is exceptional, reflecting its proficiency in classifying different driving modes.

When considering the Random Forest model, we observe a slight improvement in the precision and recall for the same acceleration and braking classes, underscoring a marginally enhanced performance in distinguishing these classes. The Random Forest's accuracy parallels that of the decision tree, which implies that both models exhibit comparable effectiveness in their classification capabilities. However, the slight edge in precision and recall for the Random Forest may suggest it has a better handle on managing the intricacies and variability within the data.

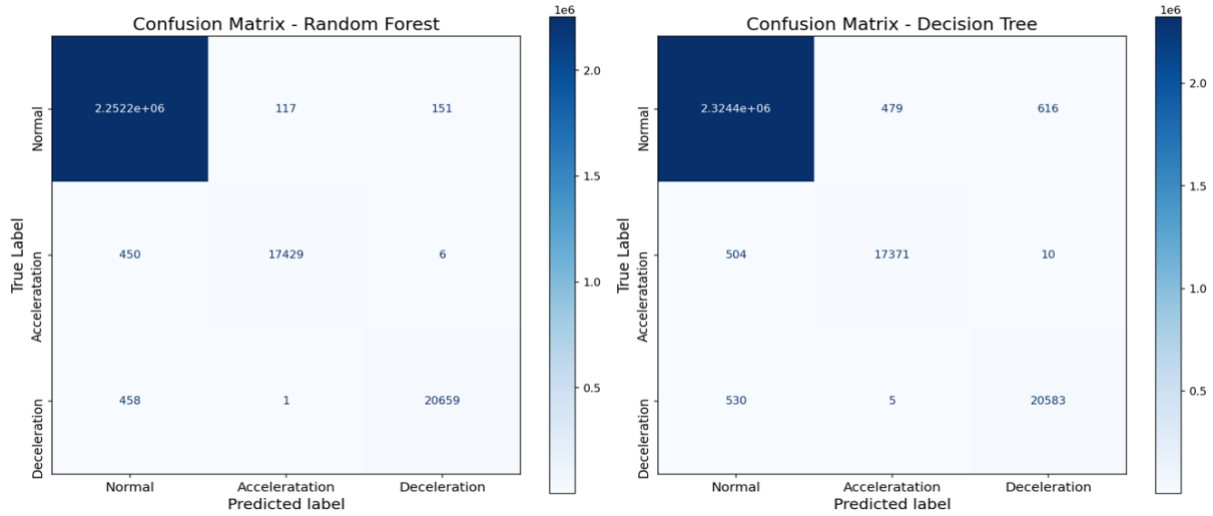


Figure 4.17 Confusion matrix Random Forest and Decision Tree.

Based on the result in subsections 4.2.3 and 4.2.4, we skipped the XGBoost and LSTM models due to poor performance. The results from the Random Forest and decision tree validate the efficacy of using our tuned model as a pre-trained model on a larger dataset. This approach is beneficial, as both datasets exhibit similar distributions. The results demonstrate a high number of correctly predicted instances within the minority class for both acceleration and deceleration [93].

Table 4.8 Result for each model.

Model	**Type of event**	**Precision**	**Recall**	**F1 score**
Random Forest	Normal driving	0.999	0.999	0.999
	Acceleration	0.993	0.974	0.983
	Deceleration	0.992	0.978	0.985
Decision Tree	Normal driving	0.999	0.999	0.999
	Acceleration	0.972	0.971	0.972
	Deceleration	0.970	0.974	0.972

CHAPTER 5 CONCLUSION AND RECOMMENDATIONS

In the era of digital transformation, the integration of smartphone sensors into the realm of auto insurance offers a promising avenue for enhancing road safety and insurance accuracy. Our research not only underscores the potential of smartphone kinematic data in the UBI landscape but also sheds light on the challenges and solutions associated with data quality and alignment. The modified algorithm (including reorientation on 2D and validation method), tailored specifically for the UBI dataset, exemplifies the adaptability of existing technologies to specific datasets and conditions. As the ubiquity of smartphones continues to grow, their role in data-driven sectors like auto insurance is poised to become increasingly significant. Future research might delve deeper into refining these algorithms, exploring multiple trips, ensuring a holistic understanding and application of these methodologies in real-world scenarios.

5.1 Summary

In conclusion, after meticulous testing and analysis, we find merit in Android algorithm's suggested method for reorienting smartphones to align with a vehicle's direction. Due to the absence of labeled data and unknown smartphone positions, we devised three distinct validation methods to assess the efficacy of any proposed approach namely, correlation test, gravity influence, and deceleration events. Upon applying our validation methods to a collected data sample, our findings lent support to our initial hypothesis. Specifically, during harsh deceleration events, there was a notable correlation between the acceleration/deceleration of the GNSS speed and the acceleration after the application of the reorientation algorithm, with a value reaching as high as 0.47.

Moving forward, for the second objective, we explore the potential of smartphone accelerometers for predicting deceleration and acceleration events. By integrating data from GNSS and accelerometers, we plan to use GNSS speed as a benchmark for labeling and assessing accuracy with machine learning classifiers. We tested four models: Decision Tree, Random Forest, LSTM, and XGBoost. The Random Forest model showed the most promise, achieving a macro F-1 score of 91%, compared to 87% for the Decision Tree, 71% for LSTM, and 62% for XGBoost. These results suggest that accelerometers could be effective for detecting braking events, particularly where GNSS sensors are not available.

Additionally, we aggregated data from 100 trips to address the issues of imbalanced data and small sample sizes that could lead to a high false positive rate. Ultimately, we applied explainable AI techniques to identify the most influential parameters in our model. Notably, total acceleration had the highest SHAP value for identifying the deceleration and acceleration classes. This suggests that in the future, if we can reorient and align the phone with respect to the vehicle, the Y-axis would exhibit the highest SHAP value. The application of traffic event classification using an accelerometer would be particularly useful in downtown areas affected by GNSS noise due to skyscrapers, or in the absence of GNSS readings, where acceleration and braking behavior could be inferred solely from accelerometer data and deceleration.

5.2 Limitations

The limitations of our study for both objectives are as follows: firstly, we applied the reorientation algorithm to only one trip at a time. Additionally, the phone must remain stationary; removing it from the cupholder can interfere with our analysis.

Secondly, the effectiveness of our proposed method for detecting traffic events is also dependent on the quality of our labeling and the accuracy of sensor readings. In this study, we had access to GNSS data, which we used to assign acceleration and braking events. However, for both reorientation and traffic event detection, the lowest reading frequency in our case, which is GNSS at 1Hz or 1 sample per second, could result in missing observations.

5.3 Future Work

In future research, enhancing the alignment of smartphone sensors with a vehicle's coordinate system is anticipated to refine model accuracy and influence SHAP values, which could more precisely indicate the sensor axis most aligned with the vehicle's motion. This can be augmented by integrating advanced deep learning models, particularly transformers, which show promise for their adept handling of time-series data. Additionally, expanding our methods to fuse smartphone sensor data with other vehicular and environmental information sources could further refine event detection accuracy. We envision employing sophisticated techniques to dynamically reorient sensors, even as the smartphone's position changes, utilizing machine learning to adapt in real-time.

The implementation of more intricate behavioral and contextual models may also yield a finer understanding of driver habits and the varying impacts of external conditions on driving patterns. Addressing data gaps due to sensor reading frequencies, possibly through advanced data imputation techniques, could ensure a richer, more reliable dataset for analysis.

Practical application and rigorous testing in a real-world setting, with a diverse array of vehicles and environments, will be crucial in evaluating the robustness of these models. Moreover, crafting an interactive interface could offer drivers actionable feedback, potentially improving road safety through behavioral modification.

REFERENCES

[1] World Health Organization. "Global status report on road safety 2023," 13 December 2023, ISBN: 978-92-4-008651-7, Available: https://www.who.int/publications-detail-redirect/9789240086517.

[2] H. Noura and W. Znaidi, “Security of Cooperative Intelligent Transport Systems: Standards, Threats Analysis and Cryptographic Countermeasures,” *Electronics (Basel)*, vol. 4, Mar. 2015, doi: 10.3390/electronics4030380.

[3] “City of Montrèal,” *Plan d’action Vision Zéro décès et blessé grave 2019-2021*, 2021.

[4] “Canadian Motor Vehicle Traffic Collision Statistics,” 2020, [Online], Available: https://tc.canada.ca/en/road-transportation/statistics-data/canadian-motor-vehicle-traffic-collision-statistics-2020.

[5] S. A. Shaheen and R. Finson, “Intelligent Transportation Systems,” in *Encyclopedia of Energy*, C. J. Cleveland, Ed., New York: Elsevier, 2004, pp. 487–496. doi: 10.1016/B0-12-176480-X/00191-1.

[6] U. Fugiglando et al., "Driving Behavior Analysis through CAN Bus Data in an Uncontrolled Environment," in IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 2, pp. 737-748, Feb. 2019, doi: 10.1109/TITS.2018.2836308.

[7] A. Varhelyi, A. Laureshyn, and C. Johnsson, “Surrogate measures of safety and traffic conflict observations,” *the Transport System and Transport Policy*, pp. 93–126, 2018.

[8] “Motor Vehicle Safety Issues.” Mar. 21, 2023, [Online], Available: https://injuryfacts.nsc.org/motor-vehicle/motor-vehicle-safety-issues/speeding/data-details/.

[9] T. Toledo and T. Lotan, “In-Vehicle Data Recorder for Evaluation of Driving Behavior and Safety,” *Transport Research Record*, vol. 1953, no. 1, pp. 112–119, Jan. 2006, doi: 10.1177/0361198106195300113.

[10] R. Eenink, M. Reurings, R. Elvik, J. Cardoso, S. Wichert, and C. Stefan, “Accident prediction models and road safety impact assessment: recommendations for using these tools,” *Institute for Road Safety Research, Leidschendam*, 2008.

[11] A. Bittel, A. Elazzazi, and D. Bittel, “Accuracy and Precision of an Accelerometer-Based Smartphone App Designed to Monitor and Record Angular Movement over Time,” *Telemedicine Journal E-Health*, vol. 22, Oct. 2015, doi: 10.1089/tmj.2015.0063.

[12] C. Jing, G. Huang, Q. Zhang, X. Li, Z. Bai, and Y. Du, “GNSS/Accelerometer Adaptive Coupled Landslide Deformation Monitoring Technology,” *Remote Sensing (Basel)*, vol. 14, no. 15, 2022, doi: 10.3390/rs14153537.

[13] M. Tundo, E. Lemaire, and N. Baddour, "Correcting Smartphone Orientation for Accelerometer-Based Analysis", *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2013, doi: 10.1109/MeMeA.2013.6549706.

[14] M. Ghavidel, N. Khademi, E. B. Samani, and L.-M. Kieu, “A Random Effects Model for Travel-Time Variability Analysis Using Wi-Fi and Bluetooth Data,” *Journal Transport Engineering A System*, vol. 148, no. 2, 2022, doi: 10.1061/JTEPBS.0000624.

[15] Astarita Vittorio, Vaiana Rosolino, Iuele Teresa, Caruso Maria Vittoria, P. Giofrè Vincenzo, De Masi Francesco, “A Mobile Application for Road Surface Quality Control: UNIquALroad,” *Procedia Society Behavior Science*, vol. 54, pp. 1135–1144, Oct. 2012, doi: 10.1016/j.sbspro.2012.09.828.

[16] D. Leblanc, S. Bao, J. R. Sayer, and S. Bogard, “Longitudinal Driving Behavior with Integrated Crash-Warning System,” *Transport Research Record*, vol. 2365, pp. 17–21, 2013.

[17] Y. Peng, L. N. Boyle, and J. D. Lee, “Reading, typing, and driving: How interactions with in-vehicle systems degrade driving performance,” *Transport Research Part F Traffic Psychology Behavior*, vol. 27, pp. 182–191, 2014, doi: 10.1016/j.trf.2014.06.001.

[18] Y. Barnard, S. Innamaa, S. Koskinen, H. Gellerman, E. Svanberg, and H. Chen, “Methodology for Field Operational Tests of Automated Vehicles,” *Transportation Research Procedia*, pp. 2188–2196, 2015, doi: 10.1016/j.trpro.2016.05.234.

[19] Transportation, subcommittee on surface, and merchant marine infrastructure, "Oversight of motor carrier safety efforts," 2010.

[20] National highway traffic safety administration, “Visual-Manual NHTSA Driver Distraction Guidelines for In-Vehicle Electronic Devices: Notice of Proposed Federal Guidelines,” *Federal Register 77: 37*, pp. 11199–11250, 2012.

[21] M. Distner, M. Bengtsson, T. Broberg, and L. Jakobsson, “City safety—a system addressing rear-end collisions at low speeds,” *21st International Technical Conference on the Enhanced Safety of Vehicles*, 2009.

[22] M. Dozza, R. Schindler, G. Bianchi-Piccinini, and J. Karlsson, “How do drivers overtake cyclists?,” *Accident Analysis Prevention*, vol. 88, pp. 29–36, 2016, doi: 10.1016/j.aap.2015.12.008.

[23] Y. Wang and N. L. Nihan, “Estimating the risk of collisions between bicycles and motor vehicles at signalized intersections,” *Accident Analysis Prevention*, vol. 36, no. 3, pp. 313–321, 2004.

[24] H. H. Nguyen, P. Taneerananon, C. Koren, and P. Luathep, “The evolution of criteria for identifying black spots and recommendations for developing countries,” Journal of the Eastern Asia Society for Transportation Studies, 2014.

[25] E. Hauer, “Identification of sites with promise,” *Transport Research Record*, vol. 1542, no. 1, pp. 54–60, 1996.

[26] J. W. Joubert, D. De Beer, and N. De Koker, “Combining accelerometer data and contextual variables to evaluate the risk of driver behaviour,” *Transport Research Part F Traffic Psychol Behav*, vol. 41, pp. 80–96, 2016.

[27] S. G. Klauer, T. A. Dingus, V. L. Neale, J. D. Sudweeks, and D. J. Ramsey, “The Impact of Driver Inattention on Near-Crash/Crash Risk: An Analysis Using the 100-Car Naturalistic Driving Study Data,” *National Highway Traffic Safety Administaration,* 2006.

[28] F. Guo, S. G. Klauer, J. M. Hankey, and T. A. Dingus, “Near Crashes as Crash Surrogate for Naturalistic Driving Studies,” *Transport Research Record*, vol. 2147, no. 1, pp. 66–74, Jan. 2010, doi: 10.3141/2147-09.

[29] G. A. Davis, J. Hourdos, H. Xiong, and I. Chatterjee, “Outline for a causal model of traffic conflicts and crashes,” *Accident Analysis Prevention*, vol. 43, no. 6, pp. 1907–1919, 2011, doi: 10.1016/j.aap.2011.05.001.

[30] R. Arvin and A. J. Khattak, “Driving impairments and duration of distractions: assessing crash risk by harnessing microscopic naturalistic driving data,” *Accident Analysis Prevention,*, vol. 146, 2020.

[31] B. Wali and A. J. Khattak, “Harnessing ambient sensing & naturalistic driving systems to understand links between driving volatility and crash propensity in school zones–A generalized hierarchical mixed logit framework,” *Transport Research Part C Emerg Technol*, vol. 114, pp. 405–424, 2020.

[32] D. I. Tselentis, G. Yannis, and E. I. Vlahogianni, “Innovative Insurance Schemes: Pay as/how You Drive,” *Transportation Research Procedia*, vol. 14, pp. 362–371, 2016, doi: 10.1016/j.trpro.2016.05.088.

[33] F. Bruneteau and F. Bruneteau, “Why insurance telematics mattersOverview of a future EUR 50 billion market,” *Telematics Munich*, 2013.

[34] P. Händel, J. Ohlsson, M. Ohlsson, I. Skog, and E. Nygren, “Smartphone-Based Measurement Systems for Road Vehicle Traffic Monitoring and Usage-Based Insurance,” *Systems Journal, IEEE*, vol. 8, pp. 1238–1248, Dec. 2014, doi: 10.1109/JSYST.2013.2292721.

[35] J.-P. Boucher, A. M. Peârez-Marõân, and M. Santolino, “Pay-As-You-Drive Insurance: The Effect of The Kilometers on the Risk of Accident,” *Anales Del Instituto De Actuarios EspanÄoles*, vol. 19, pp. 135–154, Jan. 2013.

[36] T. Kuhlmann, P. Garaizar, and U.-D. Reips, “Smartphone sensor accuracy varies from device to device in mobile research: The case of spatial orientation,” *Behavior Research Methods*, vol. 53, no. 1, pp. 22–33, 2021, doi: 10.3758/s13428-020-01404-5.

[37] J. Ferreira *et al.*, “Driver behavior profiling: An investigation with different smartphone sensors and machine learning,” *PLoS One*, vol. 12, pp. 1–16, Apr. 2017, doi: 10.1371/journal.pone.0174959.

[38] A. Ziakopoulos, V. Petraki, A. Kontaxi, and G. Yannis, “The transformation of the insurance industry and road safety by driver safety behaviour telematics,” *Case Stud Transport Policy*, vol. 10, no. 4, pp. 2271–2279, 2022.

[39] G. Castignani, T. Derrmann, R. Frank, and T. Engel, “Driver behavior profiling using smartphones: A low-cost platform for driver monitoring,” *IEEE Intelligent transportation systems magazine*, vol. 7, no. 1, pp. 91–102, 2015.

[40] H. Eren, S. Makinist, E. Akin, and A. Yilmaz, “Estimating driving behavior by a smartphone,” *IEEE Intelligent Vehicles Symposium*, IEEE, 2012, pp. 234–239.

[41] A. Chakravarty, R. Grewal, and V. Sambamurthy, “Information technology competencies, organizational agility, and firm performance: Enabling and facilitating roles,” *Information systems research*, vol. 24, no. 4, pp. 976–997, 2013.

[42] Z. Liu, Q. Shen, and J. Ma, “A driving behavior model evaluation for UBI,” *International Journal of Crowd Science*, vol. 1, no. 3, pp. 223–236, 2017.

[43] C. Ma, W. Hao, W. Xiang, and W. Yan, “The impact of aggressive driving behavior on driver-injury severity at highway-rail grade crossings accidents,” *Journal Advance Transport*, 2018.

[44] S. G. Klauer, T. A. Dingus, V. L. Neale, J. D. Sudweeks, and D. J. Ramsey, “Comparing real-world behaviors of drivers with high versus low rates of crashes and near crashes,” *National Highway traffic safety administration*, 2009.

[45] T. Osafune, T. Takahashi, N. Kiyama, T. Sobue, H. Yamaguchi, and T. Higashino, “Analysis of accident risks from driving behaviors,” *International journal of intelligent transportation systems research*, vol. 15, pp. 192–202, 2017.

[46] J. D. Lee, “Fifty years of driving safety research,” *Human Factors*, vol. 50, no. 3, pp. 521–528, 2008.

[47] J. J. Ferreira, V. Ratten, and L.-P. Dana, “Knowledge spillover-based strategic entrepreneurship,” *International Entrepreneurship and Management Journal*, vol. 13, pp. 161–167, 2017.

[48] Z. Constantinescu, C. Marinoiu, and M. Vladoiu, “Driving style analysis using data mining techniques,” *International Journal of Computers Communications & Control*, vol. 5, no. 5, pp. 654–663, 2010.

[49] D. A. Johnson and M. M. Trivedi, “Driving style recognition using a smartphone as a sensor platform,” *14th International IEEE Conference on Intelligent Transportation Systems (ITSC), IEEE*, pp. 1609–1615, 2011.

[50] M. E. Mumcuoglu *et al.*, “Driving behavior classification using long short term memory networks,” *AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*, IEEE, 2019, pp. 1–6, 2019.

[51] P. Li, M. Abdel-Aty, Q. Cai, and Z. Islam, “A deep learning approach to detect real-time vehicle maneuvers based on smartphone sensors,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 4, pp. 3148–3157, 2020.

[52] A. Das, M. N. Khan, and M. M. Ahmed, “Detecting lane change maneuvers using SHRP2 naturalistic driving data: A comparative study machine learning techniques,” *Accident Analysis Prevention*, vol. 142, 2020.

[53] E. I. Vlahogianni and E. N. Barmpounakis, “Driving analytics using smartphones: Algorithms, comparisons and challenges,” *Transport Research Part C Emergency Technology*, vol. 79, pp. 196–206, 2017.

[54] L. Liu *et al.*, “Smartphone-based hard-braking event detection at scale for road safety services,” *Transport Research Part C Emergency Technology*, vol. 146, p. 103949, 2023.

[55] P. Mohan, V. N. Padmanabhan, and R. Ramjee, “Nericell: using mobile smartphones for rich monitoring of road and traffic conditions,” in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 357-358, 2008.

[56] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan, “Mobile phone based drunk driving detection,” in *2010 4th International Conference on Pervasive Computing Technologies for Healthcare*, IEEE, pp. 1-8, 2010.

[57] J. White, C. Thompson, H. Turner, B. Dougherty, and D. C. Schmidt, “Wreckwatch: Automatic traffic accident detection and notification with smartphones,” *Mobile Networks and Applications*, vol. 16, pp. 285–303, 2011.

[58] R. Bhoraskar, N. Vankadhara, B. Raman, and P. Kulkarni, “Wolverine: Traffic and road condition estimation using smartphone sensors,” *fourth international conference on communication systems and networks*, IEEE, pp. 1-6, 2012.

[59] C. Saiprasert and W. Pattara-Atikom, “Smartphone enabled dangerous driving report system,” *46th Hawaii international conference on system sciences*, IEEE, pp. 1231–1237, 2013.

[60] G. Singh, D. Bansal, S. Sofat, and N. Aggarwal, “Smart patrolling: An efficient road surface monitoring using smartphone sensors and crowdsourcing,” *Pervasive Mobility Computation*, vol. 40, pp. 71–88, 2017.

[61] S. Daptardar, V. Lakshminarayanan, S. Reddy, S. Nair, S. Sahoo, and P. Sinha, “Hidden Markov model based driving event detection and driver profiling from mobile inertial sensor data,” *IEEE SENSORS*, pp. 1–4, 2015.

[62] K. Zang, J. Shen, H. Huang, M. Wan, and J. Shi, “Assessing and mapping of road surface roughness based on GPS and accelerometer sensors on bicycle-mounted smartphones,” *Sensors*, vol. 18, no. 3, 2018.

[63] P. Li, M. Abdel-Aty, Q. Cai, and Z. Islam, “A deep learning approach to detect real-time vehicle maneuvers based on smartphone sensors,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 4, pp. 3148–3157, 2020.

[64] S. Gelmini, S. C. Strada, M. Tanelli, S. M. Savaresi, and V. Biase, “Online assessment of driving riskiness via smartphone-based inertial measurements,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5555–5565, 2020.

[65] T. Ozyagcilar, “Accuracy of angle estimation in ecompass and 3d pointer applications,” *Freescale Semiconductor*, 2015.

[66] N. Promwongsa and T. Sanguankotchakorn, “Packet size optimization for energy-efficient 2-hop in multipath fading for WBAN,” *22nd Asia-Pacific Conference on Communications (APCC)*, IEEE, pp. 445–450, 2016.

[67] S. H. Kang and T. Nguyen, “Distance based thresholds for cluster head selection in wireless sensor networks,” *IEEE Communications Letters*, vol. 16, no. 9, pp. 1396–1399, 2012.

[68] I. Y. Bar-Itzhack, “Extension of Euler’s theorem to n-dimensional spaces,” in *Flight Mechanics/Estimation Theory Symposium*, 1989.

[69] J. B. Kuipers, "Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality", *Princeton University press*, 1999.

[70] M. E. Goldstein, “An exact form of Lilley’s equation with a velocity quadrupole/temperature dipole source term,” *Journal Fluid Mechanics*, vol. 443, pp. 231–236, 2001.

[71] D. M. Henderson, “Euler angles, quaternions, and transformation matrices for space shuttle analysis,” 1977, [Online], Available: https://api.semanticscholar.org/CorpusID:118382631

[72] S. Miura, L.-T. Hsu, F. Chen, and S. Kamijo, “GPS error correction with pseudorange evaluation using three-dimensional maps,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3104–3115, 2015.

[73] M. Pedley, “Tilt sensing using a three-axis accelerometer,” *Freescale semiconductor application note*, vol. 1, pp. 2012–2013, 2013.

[74] A. M. Walker, D. P. Miller, and C. Ling, “Spatial orientation aware smartphones for tele-operated robot control in military environments: a usability experiment,” in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, SAGE Publications Sage CA: Los Angeles, CA, pp. 2027–2031, 2013.

[75] L. Liu *et al.*, “Smartphone-based hard-braking event detection at scale for road safety services,” *Transport Research Part C Emergency Technology*, vol. 146, 2023, doi: 10.1016/j.trc.2022.103949.

[76] W. Liu, J. Wei, M. Liang, Y. Cao, and I. Hwang, “Multi-sensor fusion and fault detection using hybrid estimation for air traffic surveillance,” *IEEE Transport Aerospace Electronics System*, vol. 49, no. 4, pp. 2323–2339, 2013.

[77] M. R. Carlos, M. E. Aragón, L. C. González, H. J. Escalante, and F. Martínez, “Evaluation of detection approaches for road anomalies based on accelerometer readings—Addressing who’s who,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3334–3343, 2018.

[78] L. Rokach and O. Maimon, “Feature set decomposition for decision trees,” *Intelligent Data Analysis*, vol. 9, no. 2, pp. 131–158, 2005.

[79] Y.-Y. Song and L. U. Ying, “Decision tree methods: applications for classification and prediction,” *Shanghai Arch Psychiatry*, vol. 27, no. 2, 2015.

[80] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, pp. 5–32, 2001.

[81] M. Pal, “Random forest classifier for remote sensing classification,” *Intelligent Journal Remote Sensing*, vol. 26, no. 1, pp. 217–222, 2005.

[82] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” *the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[83] P. Zhang, Y. Jia, and Y. Shang, “Research and application of XGBoost in imbalanced data,” *Intelligent Journal Distribution Sensing Networks*, vol. 18, no. 6, Jun. 2022, doi: 10.1177/15501329221106935.

[84] L. S.-T. Memory, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 2010.

[85] C. Rattanapoka, M. Duangdoaw, and N. Phetponpun, “The Comparison of Thai Speech Emotional Features for LSTM Classifier,” *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, vol. 17, no. 4, pp. 500–509, 2023.

[86] C. Lever, R. Walls, Y. Nadji, D. Dagon, P. McDaniel, and M. Antonakakis, “Domain-z: 28 registrations later measuring the exploitation of residual trust in domains,” *IEEE symposium on security and privacy (SP)*, IEEE, pp. 691–706, 2016.

[87] G. Visani, E. Bagli, F. Chesani, A. Poluzzi, and D. Capuzzo, “Statistical stability indices for LIME: Obtaining reliable explanations for machine learning models,” *Journal of the Operational Research Society*, vol. 73, no. 1, pp. 91–101, 2022.

[88] “core/java/android/hardware/SensorManager.java - platform/frameworks/base - Git at Google.”

[89] S. Yadav and S. Shukla, “Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification,” *IEEE 6th International conference on advanced computing (IACC)*, IEEE, pp. 78–83, 2016

[90] H. Li, J. Li, X. Guan, B. Liang, Y. Lai, and X. Luo, “Research on Overfitting of Deep Learning,” *15th International Conference on Computational Intelligence and Security (CIS)*, pp. 78–81, 2019, doi: 10.1109/CIS.2019.00025.

[91] X. Hu, L. Chu, J. Pei, W. Liu, and J. Bian, “Model complexity of deep learning: a survey,” *Knowledge Information System*, vol. 63, no. 10, pp. 2585–2619, 2021, doi: 10.1007/s10115-021-01605-0.

[92] D. Bzdok, T. E. Nichols, and S. M. Smith, “Towards algorithmic analytics for large-scale datasets,” *Nature Machine Intelligent*, vol. 1, no. 7, pp. 296–306, 2019, doi: 10.1038/s42256-019-0069-5.

[93] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD

’16. New York, NY, USA: Association for Computing Machinery, pp. 785–794, 2016, doi: 10.1145/2939672.2939785.

APPENDIX A REORIENTATION ALGORITHM

In this section, we have presented the function we have created to reorient the smartphone.

1. Get Azimuth: to find the angle of the phone to magnetic north.

```
def get_azimuth(acc_x, acc_y, acc_z, mag_x, mag_y, mag_z):
    declination = -13.5 #based on location for Montreal
    azimuth = []
    R = []
    I = []
    for i in range(0, len(acc_x)):
        accs=(acc_x[i], acc_y[i], acc_z[i])
        mags=(mag_x[i], mag_y[i], mag_z[i])
        R,I = getRotationMatrix(accs,mags)
        azimuth.append(np.degrees(np.arctan2(R[1], R[5])) + declination)

    return azimuth
```

2. Reorientation matrix: to normalize and find the Euler matrix based on the accelerometer and magnetometer input.

```
def getRotationMatrix(gravity, geomagnetic):
        Ax = gravity[0]
        Ay = gravity[1]
        Az = gravity[2]
        normsqA = (Ax * Ax + Ay * Ay + Az * Az)
        g = 9.81
        freeFallGravitySquared = 0.01 * g * g

        if normsqA < freeFallGravitySquared:
                # gravity less than 10% of normal value
               return False

        Ex = geomagnetic[0]
        Ey = geomagnetic[1]
        Ez = geomagnetic[2]
        Hx = Ey * Az - Ez * Ay
        Hy = Ez * Ax - Ex * Az
        Hz = Ex * Ay - Ey * Ax

        normH = np.sqrt(Hx * Hx + Hy * Hy + Hz * Hz)
```

```
if normH < 0.1:
   # device is close to free fall, or close to magnetic north
   pole.
        return False

invH = 1.0 / normH
Hx *= invH
Hy *= invH
Hz *= invH

invA = 1.0 / np.sqrt(Ax * Ax + Ay * Ay + Az * Az)
Ax *= invA
Ay *= invA
Az *= invA

Mx = Ay * Hz - Az * Hy
My = Az * Hx - Ax * Hz
Mz = Ax * Hy - Ay * Hx

R = np.zeros(9)

R[0] = Hx;     R[1] = Hy;     R[2] = Hz
R[3] = Mx;     R[4] = My;     R[5] = Mz
R[6] = Ax;     R[7] = Ay;     R[8] = Az

invE = 1.0 / np.sqrt(Ex * Ex + Ey * Ey + Ez * Ez)
c = (Ex * Mx + Ey * My + Ez * Mz) * invE
s = (Ex * Ax + Ey * Ay + Ez * Az) * invE

I = np.zeros(9)

I[0] = 1;     I[1] = 0;     I[2] = 0;
I[3] = 0;     I[4] = c;     I[5] = s;
I[6] = 0;     I[7] = -s;    I[8] = c;

return R, I
```

3. Reorientation matrix: based on the azimuth and the heading angle calculate the reoriented accelerometer value for the smartphone.

```
def reorientation(X,Y,Z, phi=0 ):

        xm = np.median(X)
        ym = np.median(Y)
        zm = np.median(Z)

        g = np.sqrt(xm**2 + ym**2 + zm**2)


       alpha = np.arcsin(xm/g)
       beta  = np.arcsin(ym/g)

        phi *= -1

        ca = np.cos(alpha)
        sa = np.sin(alpha)
        cb = np.cos(beta)
        sb = np.sin(beta)
        cp = np.cos(phi)
        sp = np.sin(phi)

        l = list(map(
                lambda x,y,z: [
                y*cb*sp + z*(-cp*sa -ca*sb*sp) + x*(ca*cp -sa*sb*sp),
                y*cb*cp + x*(-cp*sa*sb -ca*sp) + z*(sa*sp -ca*cp*sb),
                z*ca*cb + x*sa*cb + y*sb
                ]
                X,Y,Z
        ))

        return np.array(l)
```