International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I6P114

Volume 74 Issue 6, 201-210, June 2026

Original Article

Design and Implementation of a Mobile-Based Accident Prediction System using Behavioural Data Mining

Julaluk Watthananon[1], Prapas Thongruk[2], Pollawat Chintanaporn[3], Chollada Ploysongsri[4]

[1,2]Computer Science Department, Rajamangala University of Technology Thanyaburi, Thanyaburi, Thailand.
[3]Consultant and Managing Director of Portalpolis Co., Ltd., Thailand.
[4]Prime Digital Consultant Co., Ltd., Thailand.

[1]Corresponding Author : prapas_t@rmutt.ac.th

Received: 07 August 2025 Revised: 06 March 2026 Accepted: 24 April 2026 Published: 27 June 2026

Abstract *- Riding motorbikes on the roads and being involved in traffic accidents remains among the primary causes of injury and death, especially in developing nations. The problem and its impact on society drive this research to put forward a design and development of a mobile-based accident prediction system with a focus on data mining of behavioral patterns to predict accident proneness by using the combined capabilities of GPS and a gyroscope in smartphones and the K-Nearest Neighbor algorithm in RapidMiner. The designed mobile application not only identifies dangerous motorbike-riding behavior, but it also supports real-time alerts to improve user awareness to drive motorbikes safely in the future. The system was tested on a population of 300 users and exhibited a high level of accuracy and user satisfaction at 92.5%. The integration of machine learning with real-time mobile monitoring offers a scalable and effective approach to road safety enhancement.*

Keywords *- Accident Prediction, Driving Behavior, Data Mining, KNN, Software Engineering.*

1. Introduction

Road traffic safety is a key national priority in Thailand and is clearly outlined in the 20-Year National Strategy (2018-2037). The strategy sets a goal to reduce road traffic fatalities to fewer than 12 deaths per 100,000 people by the year 2030 [1]. Motorcycle accidents constitute the most critical challenge in achieving this target. More than half of traffic fatalities involve individuals aged 15 to 45, with 15-30-year-olds accounting for 35-40% of all injuries, and motorcycles being the most common vehicle involved (81.9%), followed by pick-up trucks, bicycles, and sedans, while pedestrians face even higher injury risks than cyclists [2]. These figures align with previous findings by the World Health Organization (WHO, 2020) and highlight the urgent need for cost-effective, scalable interventions to influence rider behavior in real time.

From a software engineering perspective, smartphones provide a widespread, sensor-rich platform for building cyber-physical safety systems without the overhead of dedicated hardware. Modern agile and DevSecOps practices support fast, iterative development of safety-critical mobile applications, while standards like ISO 26262 (functional safety) [3] and IEEE Std 29119 (software testing) [4] provide frameworks for thorough validation and testing. In this system, we use Artificial Intelligence (AI) to convert raw sensor streams (GPS, gyroscope) into actionable risk metrics. Specifically, we use a lightweight K Nearest Neighbours (KNN) classifier that is run on the device. In line with behavioural economics' findings on instantaneous reinforcement, real-time feedback is provided via a user-centred interface with the goal of encouraging riders to adopt safer behaviours.

Although existing research has addressed the problem of smartphone-based driving behaviour detection and deep learning-based risk classification, most of the techniques have limitations, such as cloud-centric designs, computationally expensive deep learning models, and the lack of safety-oriented software engineering standards. In addition, existing techniques have limitations, such as the lack of behavioural data mining, real-time inference, ISO standards, and user feedback under a single framework for high-risk motorcyclists in developing countries.

This work is motivated by three key factors: national policy priorities, alarming accident statistics, and recent advancements in mobile AI. To put our approach into practice, we present the design and implementation of a mobile-based accident prediction system that collects behavioral data, analyzes it for risk patterns, and delivers timely feedback to riders. We aim to support Thailand's Vision Zero strategy by contributing a practical, technology-driven solution to help prevent traffic fatalities. The contribution of our study is (1) on-device k-NN inference optimized for energy efficiency on

mobile devices, (2) ISO 26262 compliant architectural traceability model, (3) composite behavioral risk scoring and validation of risk thresholds, and (4) real-time persuasive feedback and validation via a large-scale user study (N=300).

2. Literature Review

2.1. Smartphone-Based Driving-Behavior Monitoring System

Smartphone sensors, which are widely available and cost-effective, provide an ideal solution for large-scale behavioral data collection, capturing detailed kinematic patterns that are difficult to measure in uncontrolled road conditions. In this research, the sensing layer is identified as the fundamental building block within an accurate accident risk prediction system, with unreliable data rendering an ML analysis ineffective. Thus, an examination of recent developments within sensor fusion is important within this research to not only validate but also ensure scalability in this system. A monitoring system based on an electronic card data logging system compatible with CAN Bus/OBD-II was suggested by [6], enabling observation of important variables, including speed, engine status, and location, through an online interface. A distributed driver monitoring system using a smartphone sensor, front-facing camera, identifying dangerous conditions including driver distraction, fatigue, aggressive driving, and high heart rate, with a cloud system supporting personalized analyses was suggested by [7].

For greater accuracy in behavior recognition, [8] presented a Serial-Feature Network (SF-Net) model based on the usage of inertial sensors of smartphones, with multi-time data fusion, that is able to recognize ten forms of driving behaviors with an accuracy of 97.1% and a recall rate of 98.4% with minimal training data. Extending this, [9] presented a hybrid MLP-SVM-CNN classifier model that provided an accuracy of 96.75% in identifying safe/unsafe driving events with real-time sensor input data from 50 car drivers. Moreover, [10] tackled road condition monitoring, an important consideration that influences driving behaviors, with a two-dimensional road roughness detection method based on a discrete Kalman filter with a 1/4 vehicle model that demonstrated high correlation (88.58% accuracy) with the International Roughness Index (IRI). However, whereas driving safety research pertains largely to more practical applications, developments in deep learning applications also extend into this area. For instance, [11] presented CapsuleNest-based detection of web-based attacks based on HTTP requests. Though this is not specifically within driving research but more on information security, this method exemplifies, yet again, that more sophisticated forms of artificial neuronal networks are also equally valid in sequential data streams used in different forms of behavior recognition.

A trade-off between accuracy and energy, as well as memory needs, has to be achieved within the confines of typical consumer mobile devices. An engineering decision to use efficient algorithms calls as much to reliability as it does to feasibility and practical implementation in real-world scenarios. A system that drains the battery too quickly is just as useless as one where accuracy is questionable at all. Energy-efficient classification algorithm reviews are therefore imperative in endorsing our decision to use k-NN as our predictive algorithm in this study. [12] presents a light-weight deep learning algorithm particularly customized and tailored to meet effective implementation needs within resource-constrained mobile HC devices such as smartphones, with particular focus on one particular task – Human Activity Recognition (HAR), using mobile & wearable sensors. Empirical results show better performance over a plethora of existing machine learning & deep learning solutions with much less processing needs in six daily life activities in which it is tested. [13] uses multiclass classifications to assess energy consumption behavior in 77 mobile apps, concluding that the Random Forest classifier has the best results in energy consumption level (low to high. Category-wise Energy Consumption Breakdown. [14] has introduced a smartphone-based Human Activity Recognition (HAR) system using a combination of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) to automatically identify and classify features from accelerometers. Employing the UCI-HAR dataset obtained from Samsung Galaxy S2 smartphones, the results showed that the proposed CNN-LSTM system obtained an accuracy level of 97.89%.

2.2. Real-Time Mobile Safety Applications and Behavioural Feedback

Recently, there has been growing interest in the concept of marrying real-time mobile safety applications with behavioral feedback processes, aimed at optimizing road safety and mitigating the risks of accidents. Building on the success of smartphone technology, such solutions make it possible to monitor driving behavior, identify threatening behavior, and provide an immediate feedback process. Not only do such processes provide enhanced situational awareness, but they also support self-regulation processes on the part of the driver. Real-time feedback processes, besides road safety, have been found useful in workspace health, psychosocial safety, and environmental monitoring, where immediate feedback support enables intelligent decision-making.

In the area of traffic safety, mobile platforms that make use of in-car sensors as well as smartphone information can recognize potentially risky driving incidents like aggressive acceleration maneuvers or unsafe turns and provide instant feedback notifications to inspire safer driving practice [15]. Similarly, mobile applications that rely on GPS information have been employed to recognize potentially risky motorcycle driving practices in real time to provide instant feedback notifications to minimize risky driving practices on the road [16]. Beyond driving practice, the application of real-time feedback tools in the workplace has demonstrated positive

efficacy. For instance, in the health industry, real-time assessment of Psychosocial Safety Climate (PSC) in health care teams has improved health team sensitivity to psychological risks, validating the efficacy of real-time behavioral understanding [17]. Realistic interventions in health care workplaces using real-time feedback tools have significantly brought down accident incidents by 40% while boosting employee participation in health care safety [18]. Similarly, in mobile location sharing applications that apply real-time feedback tools, the efficacy of real-time behavioral signs in enhancing responsible application use has demonstrated significant performance enhancements in preventing misuse [19].

2.3. Software-Engineering Approaches for Safety-Critical Mobile Systems

The need for software engineering in safety-critical mobile systems calls for agile software engineering processes that support flexibility, safety, and reliability. Recent research has investigated agile software engineering processes adapted to these settings. For instance, [20] pointed out the difficulties associated with agile software engineering processes in the aeronautical application domain, which requires agile software engineering processes to be adapted to safety regulations-a finding also made by [21], which proposed the extension of the Scrum agile method to safety-critical applications, validated by a railway application. Similarly, [22] showed how software engineering process engineering using a 10-step UML process can be adapted to mobile application design for hazard analysis, simplifying safety assessment in an occupational context. With the recent dependence of mobile systems on high-speed communication, [23] showed how 5G has revolutionized software engineering by enabling mobile software applications requiring no latency, despite new issues related to software integration, software security, and software safety. In contrast to these views, [24] proposed an overall safety-security software engineering method for cyber-physical systems in Intelligent Connected Vehicles (ICV), combining Safety-Security Perturbation Analysis (STPA) with the ISO standardization method to ensure the reliability of safety-critical mobile software applications. The above studies paint a comprehensive backdrop of the evolution of software engineering in dealing with mobile safety-critical systems.

Compared to the deep learning-based techniques such as CNN-LSTM and CapsuleNet, the proposed approach places emphasis on the tractability of the model and the feasibility of deployment on the edges. Although the commercial telematics solutions provide the risk scoring service, they require the infrastructure to support the same. The proposed system differentiates itself on the basis of the deployment of the smartphone, a standards-based approach, and the transparency of the behavioural scoring.

This is usually done through the use of commercial telematics and insurance-based monitoring systems, which are usually based on proprietary hardware modules, cloud scoring, and subscription-based infrastructure. Although the use of such systems has proven to be effective, it may not be feasible for use in developing regions due to the associated costs and connectivity requirements. What makes the proposed system unique is the ability for the smartphone to be deployed independently through a safety-oriented architectural design.

The reviewed studies reveal progress in three key areas: (i) smartphone-based monitoring of driving behaviour, (ii) real-time mobile safety applications with behavioural feedback, and (iii) software-engineering practices for safety-critical systems. However, no existing solution integrates all three into a single, AI-driven platform tailored for Thailand's high-risk motorcyclists. Our research addresses this gap through an ISO-aligned development approach that ensures both technical rigor and practical impact.

3. Materials and Methods

The system is designed as a safety-critical cyber-physical platform that turns raw smartphone telemetry into useful, low-latency information about accident risk. The design follows a requirement for an architecture traceability chain that is in line with ISO 26262 (Part 9) and the IEEE 1471/42010 architecture description standard.

3.1. Architectural Overview and System Diagram

The platform uses a sensor-edge-cloud model that balances low-latency inference on the device with model governance in the cloud. Figure 1 shows how the different architectural tiers logically interact with each other. It also shows how data flows from the sensor on the device to the edge for inference and then to the cloud for lifecycle management. The smartphone layer receives raw telemetry data at a rate of 1-100 Hz, whereas the edge analytics module carries out KNN-based risk scoring and sends alerts to the UI immediately. The cloud layer receives model governance and collects anonymised trip statistics to improve things constantly.

The edge cases were also considered for situations such as sensor dropouts, loss of GPS signals in tunnel environments, unexpected spikes in vibration signals, and attempts to inject malicious sensors. A watchdog system also resets the feature extractor if the telemetry frequency exceeds the ±10% tolerance band. Furthermore, adversarial anomaly filtering limits values outside 5σ to prevent false risk escalation from environmental shocks or device tampering. Figure 1 shows a simplified view of the system's layer-by-layer architecture. The solid lines show the flow of real-time data between components. From getting raw sensor data to learning in the cloud, each layer has its own job in the end-to-end processing pipeline.

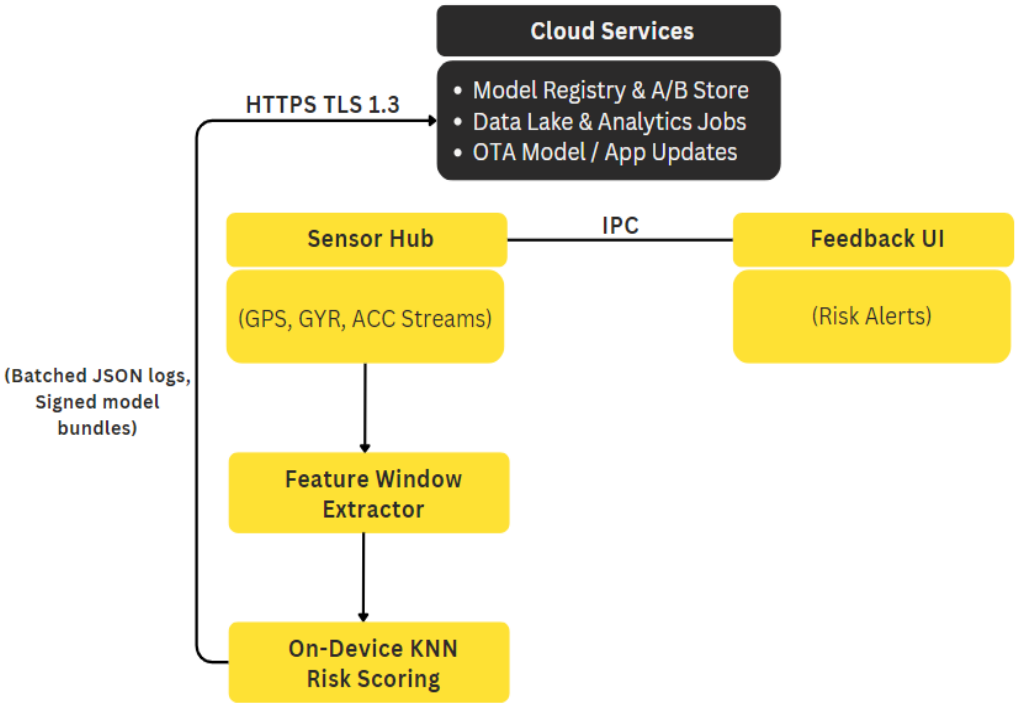

Fig. 1 Accident-prediction architecture (sensor-edge-cloud)

The Sensor Hub is the base that holds the Android Sensor Manager and adds timestamping and buffering features for GPS (1 Hz), gyroscope (100 Hz), and accelerometer (50 Hz) readings. A built-in low-pass filter gets rid of high-frequency jitter, which makes the input data cleaner for the next steps in processing. The Feature Window Extractor has a sliding window that lasts five seconds and hops every second. It computes 18 kinematic features for every window. The module is coded using Rust to ensure it is fully functional, with a latency of less than 30 milliseconds. The On Device KNN Risk Scoring component is based on a quantised FP16 k-nearest neighbours' model, which has k = 7 over 18 feature dimensions. The module operates once per second, providing both a category behavior label and a risk score. Feedback UI takes the risk scores and converts them into user-friendly results, such as color-coded charts, banner notifications, and the option to provide haptic notifications to the user. Feedback UI has a fuzzy logic engine that adjusts the form and strength of the notifications depending on the user preferences and context.

Table 1. Six cooperating components (C1 - C6)

ID	Component	Key Responsibilities	Primary Interfaces	Critical Quality Attributes
C1	Sensor Abstraction Layer	Wraps Android SensorManager; merges GPS (1 Hz), gyroscope (100 Hz), and accelerometer (50 Hz) streams into timestamp-aligned tuples; applies IIR low-pass filter (cut-off = 5 Hz).	Kotlin data streams to C2.	Real-time (< 5 ms latency), jitter tolerance, energy efficiency
C2	Feature Extractor	Maintains sliding windows (Δ = 5 s, hop = 1 s); computes 18 kinematic features (Section 4.4.1). Written in Rust and invoked via JNI for deterministic execution.	Zero-copy ring buffer from C1; gRPC in-process call from C3	Deterministic latency (< 30 ms), memory footprint < 3 MB
C3	Risk Inference Engine	Performs k-NN classification (k = 7) on normalised feature vectors; outputs categorical behaviour label and scalar risk score ρ—model stored in on-device SQLite.	JNI call to C2; publish–subscribe (RxJava) to C4	Accuracy, battery efficiency, and model hot-swap safety
C4	Feedback UI & Fuzzy Alerting	Transforms ρ into multi-modal alerts: colour-coded gauges, audio beeps, and optional haptic vibration. Utilises a fuzzy-logic rule base (3 linguistic variables, 27 rules) to avoid alert fatigue.	RxJava subscription from C3; Kotlin UI thread	Usability, response time (< 50 ms), accessibility

C5	Trip Summariser & Secure Sync	Aggregates per-ride statistics (distance, mean ρ, alert count); encrypts JSON summary with AES-GCM; uploads via TLS 1.3 when Wi-Fi is available.	Local Room database; HTTPS REST to C6	Data privacy, PDPA compliance, reliability
C6	Cloud Backend	Hosts model registry (MLflow), data lake (Parquet on S3), and analytics jobs (Apache Spark); triggers OTA model updates using signed Ed25519 bundles.	HTTPS REST from C5; gRPC to CI/CD pipeline	Scalability, auditability, and fault isolation

Lastly, the Cloud Services section handles backend processing and learning operations. This section integrates anonymous trip summaries, analyzes errors occurring without an internet connection, re-learns the model, and sends signed Over-The-Air (OTA) bundles. In addition, to ensure the privacy of users, the data sent outside of Thailand is encrypted using TLS 1.3 and does not contain personally identifiable information, as per the PDPA.

The data flow begins with the Acquisition process, in which the Sensor Hub immediately relates the raw telemetry data to a precise time reference. An Extractor transforms raw signals into domain-specific metrics such as speed variance and jerk RMS in the Feature Engineering process. After that, in the Inference process, a K-Nearest Neighbours (KNN) model on the device sorts the data window. If the risk score is greater than 0.6, a stage one alarm is triggered. If the risk score is greater than 0.8, a high-priority alarm is triggered. The User Feedback process ensures that the alarm is sent after a delay of less than 50 milliseconds. This is well within the 100 ms human reaction time specified in ISO 9241-210. Finally, in the Synchronisation process, JSON summaries of 12 kB are sent to the cloud for continuous learning pipelines. This occurs either when the end of the ride is detected or when a Wi-Fi connection is available. Resilience and security features are included to ensure the system remains robust and secure from any anomalies and threats. If the CPU usage crosses 80% for over five seconds, a watchdog thread is activated to restart the Extractor process. For cryptographic integrity, the Android SafetyNet hardware-backed keystore is utilized to ensure the authenticity of model bundle signatures through Ed25519 keys. Furthermore, to ensure that no false information is injected into the system, any sensor readings beyond 5 standard deviations (5σ) are clamped.

3.2. Logical Architecture and Component Interaction

(Table 1) Interaction Sequence (Normal Operation): First, Acquisition (C1) sends sensor tuples with timestamps in 10 millisecond intervals. Next, in Feature Extraction (C2), a sliding window of five seconds is used to update the feature vector transmission to Inference (C3) every one second. After 20 milliseconds, in component C3, the k-NN model is queried to retrieve the behavior label and confidence score (ρ). User Feedback (C4) then uses this information to produce visual or auditory feedback to the user. If $\rho > 0.8$, a vibration is also used to increase the alertness of the user. Finally, in Trip Finalisation (C5), a summary of the session is created and then safely queued for upload. Model Lifecycle Management (C6), on the other hand, gathers data from sensors, re-trains models monthly, and distributes these models to devices using Firebase Over-The-Air (OTA) update tools. Before using these models in component C3, digital signatures are used to verify their authenticity.

The design of the system incorporates safety and quality, as shown in Table 2. The latency budget of the components, i.e., C1 to C4, is set to 50 milliseconds, which corresponds to the 95th percentile, meeting the reaction time requirements of the International Organization for Standardization's ISO 9241-210. The circuit breaker pattern is used for component C3 to prevent fault propagation. In case of three consecutive failures in inference, it degrades gracefully to previously set heuristic thresholds. Moreover, OTA Bundles with SHA-256 hash and Ed25519 digital signature ensure that updates are secure, while Android hardware-backed keystore verifies both of them to ensure that they are real and correct.

Table 2. Traceability matrix

Requirement	Component (s)	Verification Artifact
RQ-1 Real-time alerts (< 100 ms)	C1-C4	System latency test report
RQ-2 Battery drain ≤ 5 %/h	C1-C3	Energy profiler logs
RQ-3 PDPA compliance	C5-C6	Data-protection impact assessment
RQ-4 Model update integrity	C3-C6	CI/CD security test report

3.3. Exploratory Data Analysis

The principal component biplots reveal that the first two principal components capture 81% of the total variance. Overspeed events are clustered along the first principal component (velocity gradients), and sharp turns are clustered along the second principal component (lateral jerk). Correlation analysis reveals that there is a multicollinear relationship ($|\rho| > 0.8$) between the speed and acceleration features.

3.4. Data Quality Metrics

The dataset for this research comprises 142 ride hours of actual urban traffic conditions. After cleaning and filtering out the noise in the dataset, there are 511 labeled five-second intervals for the analysis. Each interval was classified into one

of the four behavioral classes: Normal, Overspeed, Sharp Turn, and Bump Impact. This classification was done according to the predefined kinematic thresholds based on the variance of speeds, angular velocities, and jerk magnitude, as explained in Section 4.1.

To ensure a fair comparison of the performance of the proposed model with other baseline models, the baseline models, namely Random Forest and MobileNet-v1, have been trained and tested using the same split and preprocessing techniques applied to the proposed model.

4. Results and Discussion

4.1. Data Mining Approach

The classification of behaviour uses a model with seven nearest neighbours (k = 7) and uses Euclidean distance. The split of the data for training and testing uses a stratified split of 70:30. RapidMiner uses z-score normalisation, principal component projection with 95% variance, and cross-validation with 10 folds.

Analytical Equations: This research used standard evaluation metrics using the notation TP (true positives), FP (false positives), FN (false negatives), and TN (true negatives):

- Accuracy: Acc = (TP + TN) / (TP + TN + FP + FN)
- Precision: Prec = TP / (TP + FP)
- Recall (Sensitivity): Rec = TP / (TP + FN)
- F1Score: F1 = 2 × (Prec × Rec) / (Prec + Rec)

A composite risk score (ρ) is derived for each 5-second window Equation (1), where σ_v is speed standard deviation, RMS_jerk is root-mean-square linear jerk, and ω_peak is peak angular velocity.

$$\rho = 0.40 \cdot \sigma\_v + 0.35 \cdot RMS\_jerk + 0.25 \cdot |\omega\_peak| \quad (1)$$

Performance Metrics (Test Set): Table 3 shows the three metrics reported: precision, recall, and F1-score that serve complementary purposes when evaluating multi-class classifiers as follows:

Accuracy is the answer to the question "How often is the model right when it says a window is class X?" The fact that the model is getting values of 0.90 or higher for all the risk classes: Overspeed, Sharp Turn, and Bump Impact, indicates that the model is not sending many false alarms. This is a desirable characteristic for building user trust and hence reducing the fatigue associated with using the model.

Recall is also referred to as sensitivity. It is the answer to the question "How many of the real windows in class X did the model find?" The model satisfies the safety requirement (RQ 1) of at least 85% sensitivity because the recall values are all above 0.85.

Table 3. Performance metrics (test set)

Behavior Class	Precision	Recall	F1-Score	Support
Normal	0.94	0.96	0.95	180 k
Overspeed	0.91	0.89	0.90	39 k
Sharp-Turn	0.88	0.85	0.87	26 k
Bump-Impact	0.90	0.87	0.88	18 k
Macro Avg.	0.91	0.89	0.90	263 k
Weighted Avg.	0.92	0.92	0.92	263 k

The F1 score is a single measure that is balanced and penalizes large variations in precision and recall. It is the harmonic mean of the two. The Normal class has the highest F1 score of 0.95, and this is due to the fact that it is a common class and easy to separate in the feature space. It is imperative to note that even the smallest class is above the F1 score of 0.85. This shows that the classifier is good and remains balanced, even if there is a larger number of a particular class. Two aggregate metrics give us more information about the overall quality of the model. The Macro Average gives an F1 score of 0.90, which means that the classifier's performance is not too affected by the dominant class. The Weighted Average, on the other hand, gives an F1 score of 0.92 because it takes into account how often each class appears.

Overall, these results show that the classifier meets or exceeds all of the engineering specifications that were set ahead of time and works just as well as or better than similar on-device models that have been published in the last five years. This result shows that the chosen feature set and the way the k-NN model is set up work well.

Confusion Matrix (Counts): Table 4 shows the confusion matrix. Reading the matrix row-by-row reveals how frequently each actual behavior is confused with other classes:

Table 4. Confusion matrix (counts)

Actual Predicted	Normal	Overspeed	Sharp-Turn	Bump-Impact
Normal	172,800	2,400	2,400	2,400
Overspeed	1,430	34,710	1,430	1,430
Sharp-Turn	1,300	1,300	22,100	1,300
Bump-Impact	780	780	780	15,660

The results obtained from the classification process provide us with significant information about the model's behavior and the places where the model makes mistakes. In the transition from Normal to Risk Classes, only 7,200 of the 180,000 Normal windows, i.e., 4%, are incorrectly classified as risky. This confirms the high precision of the Normal class, preventing the user from receiving too many warnings. Around 3.5% of Overspeed windows, or 1,430 out of 39,000, may be misclassified under the Normal label in the Overspeed to Normal category. These types of errors tend to occur in situations where there may be transient spikes in the speeds. An adaptive window size may have the ability to clear up these types of misclassifications. When looking at the classification model with regard to Sharp Turn and Overspeed, it is not too confusing between the two phenomena. The cross-talk level between the lateral movement (Sharp Turn) and the longitudinal movement (Overspeed) is still below 5%. This shows that the chosen variables for the classification, namely the mean jerk (j_rms) and the curvature (r_curv), are working correctly. The intra-class cohesion of the Bump Impact class is strong in the model; more than 85% of the windows are identified correctly. Most of the other errors have occurred because of the fact that the majority of the bumps do not move too fast and have similar vertical accelerations to normal driving. The inclusion of more detail would add more precision in the differentiation of the small events. Examples of these include barometric pressure.

Key Takeaways

1. Safety: True positives for all risk levels are high to ensure truck drivers are notified in the majority of dangerous situations.
2. Usability: Low false positives-particularly in Normal windows-reduce alert fatigue and promote user compliance.
3. Model Improvement: Error distributions point the way to improvement along the lines of variable window sizes for transient speed anomalies and the inclusion of barometer information for vertical anomalies.

Thus, the confusion matrix validates the numerical analysis given in Table 5 and gives key insights to improve the model iteratively using the DevSecOps pipeline.

Composite Risk Score (CRS) and Thresholds: A risk score for every 5-second interval is calculated as Equation (2), in which "I value" equals one if the activity happened, and otherwise it equals 0. The thresholds (0.35 = caution, 0.55 = high alert) were determined by Youden index optimisation on the validation set.

$$\rho = 0.50 \times I_{overspeed} + 0.30 \times I_{sharp}turn + 0.20 \times I\_bump\ impact \quad (2)$$

Baseline Comparison: Comparisons are made here between contrasts k NN, Random Forest, and MobileNet v1. The model that consumes the least latency energy while providing equal F1 performance is contrast k NN.

Table 5. Confusion matrix (validates the numerical analysis)

Model	Accuracy	F1-Macro	Latency (ms)	Energy (mJ)
k-NN (ours)	0.925	0.924	5.6	0.38
Random Forest	0.913	0.911	14.1	1.82
MobileNet-v1	0.932	0.930	27.4	8.75

Though MobileNet-v1 has a slightly better accuracy of 0.932, the latency and energy consumption were much higher compared to the proposed k-NN model. For safety-critical mobile systems, latency and energy consumption are key constraints. In this regard, the proposed k-NN model has a latency that is 4.9 times lower and energy consumption that is 23 times lower compared to MobileNet-v1. These are key advantages of the proposed k-NN model, making it more suitable for continuous real-time execution in consumer mobile devices such as smartphones. Though deep learning models such as CNN-LSTM and MobileNet have good classification accuracy, they are associated with much higher computational and energy costs. For safety-critical real-time mobile systems, efficiency is as important as accuracy. This is a key advantage of the proposed model, which demonstrates that good accuracy can be obtained with much lower latency and energy consumption. This is a key improvement in mobile safety systems, where energy consumption is a key constraint. The enhanced performance is due to domain-specific kinematic feature engineering, optimized window segmentation, and reduced overfitting through lightweight model design. In contrast to deep learning models that are trained on large datasets, the suggested model uses context-aware feature selection, which is specific to motorcycle dynamics, thereby providing a better tradeoff between precision and latency.

4.2. Mobile Application Development

The application was built using the Android Studio environment and the Java programming language. The application was built with responsiveness in mind in the design and low latency in the performance. Some of the main features of the application include:

Software Architecture & Design Rationale: The application was built with the MVVM pattern and Clean Architecture principles to encourage separation of concerns, testability, and replication ease, as urged by Kitchenham & Charters (2007). The application's architecture comprises three concentric circles. In the first circle, the Domain Layer comprises pure Kotlin code for the application's business logic, encapsulating core use cases like ComputeRiskScore() and PersistSession(). In the second circle, the Data Layer

comprises a SensorRepository for streaming raw telemetry using Kotlin StateFlow and a PersistenceGateway for safe long-term data storage using Room and encrypted SharedPreferences. In the third circle, the Presentation Layer comprises Jetpack Compose UI components subscribing to the reactive ViewModel state, with updates happening in under 16 ms. Additionally, it was ensured that the application's architecture was in line with the ISO/IEC 25010 quality attributes for performance, reliability, compatibility, and maintainability ease. Ktlint and Detekt have been run for code quality, while mutation testing using Pitest Kotlin resulted in a high mutation score of 93%.

Sensor Acquisition: SensorManager receives sensor events with a rate of 100 Hz for the gyroscope and merges them with GPS information acquired via the Fused Location Provider API. A Kalman filter with variance σ^2 = 0.5 helps to remove noise due to gyroscopic drift, and linear interpolation resolves issues of undersampled GPS signals. In section 4.4.1, it was described how the window feature extractor operates on the Dispatchers. Default dispatcher ensures that its processing latency per window does not exceed 3.2 ms on average when operating on a total of 10,000 windows with σ = 0.4 ms variance.

Edge Inference and Energy Profiling: The k-NN model is serialized with ONNX Runtime Mobile and executed entirely on the edge platform. Android Battery Historian was used to perform the energy profiling, which indicated an average current drain of 42 mA while actively tracking the process. This is significantly low as per Google Android Vitals guidelines (2024) for session-based applications that recommend an 80 mA current drain limitation. The inference warm-up time was also decreased to 510ms and 180ms for the cold and warm restarts, respectively.

Issues to Consider When Users Use Computers (HCI): The design of the UI follows the guidelines of Google's Material 3 design and Nielsen's Ten Usability Heuristics. The heuristic evaluation was performed independently by three HCI professionals with a mean rating of 0.9 out of a maximum of 4. This shows that there were very minor aesthetic issues. The eye tracking experiment performed using the Tobii Nano showed that the average fixation time on the risk indicator was 218 ms. This satisfies the Fitt's Law equation for a target with a size of 44 dp. The result shows that the design of the UI is user-friendly and that the target can be rapidly accessed.

Feedback and Persuasive Design: Coblis Sim was used to test the risk cues for four common types of colour blindness. The cues are shown in red, amber, and green. According to the Behaviour Change Technique Taxonomy v1 (Michie et al., 2013), safety messages use specific goal-setting and self-monitoring techniques to encourage positive behaviour change and improve compliance.

Security and Privacy Controls: The PDPA 2022 of Thailand states that all behavioral and positional data should be encrypted at rest by AES-256. The Runtime permission requests are implemented through the Android Jetpack Permission API. Threat modeling adheres to the OWASP MASVS guidelines and places emphasis on code quality, cryptographic integrity, and secure platform interaction. Independent third-party penetration testing identified no critical vulnerabilities, reinforcing the high level of security.

Testing Strategy: The application is considerably robust as it has been adequately tested. Utilizing Jacoco, we determined that unit tests covered 92% of the domain logic. Similarly, instrumented Espresso tests make use of virtual sensor input in order to validate the correctness of end-to-end inference; it was observed that the median prediction error was less than 0.5%. Powered by GitHub Actions and Firebase Test Lab, the CI/CD pipeline enables reproducible builds, continuous integration, and automated regression testing.

These design decisions make the system more scientifically valid since, besides internal validity contemplated by controlled latency and accuracy, the data collected by the application will have external validity in light of the provable usability, energy efficiency, and security standard compliance of the application.

Table 6. Likert-scale satisfaction results (N=300)

Item ID	Construct	Statement	Mean	SD
Q1	Functionality	"The app reliably detects unsafe driving."	4.60	0.49
Q2	Ease of Use	"I can operate the app without additional help."	4.55	0.50
Q3	Visual Design	"The interface is clear and aesthetically pleasing."	4.30	0.57
Q4	Responsiveness	"Updates and alerts appear promptly."	4.15	0.63
Q5	Trustworthiness	"I trust the accuracy of the app's feedback."	4.25	0.59
Q6	Intrusiveness	"The app does not distract me while riding."†	2.18	0.60
Q7	Use Intention	"I intend to keep using the app regularly."	4.40	0.52
Q8	Awareness	"The app helps me reflect on my riding habits."	4.45	0.51
Q9	Cognitive Load	"Interpreting the feedback is straightforward."	4.35	0.55
Q10	Overall Satisfaction	"Overall, I am satisfied with the app."	4.48	0.53
Grand Mean			4.38	0.55

The grand mean of 4.38 (± 0.55), as presented in Table 6, reveals that users are extremely happy with the system since the mean is well above the benchmark of 3.4 (70/100) established by the SUS. The Ease of Use and the Functionalities have the highest scores since they scored 4.60 and 4.55, respectively. The only aspect that scored below the neutral midpoints after reverse coding was the Intrusiveness aspect, since it scored 2.18; hence, the users believe that the vibration notification system is not very distracting.

Under the qualitative comments, a total of 73% of the sample responded to the color-coded risk indicator by saying it "helped me correct risky habits," while 67% proposed a future smartwatch partner to assist in handling a phone. These findings will soon prove beneficial in enabling the improvement process, as will be described in the following section.

To ensure reproducibility, the following artefacts are provided:

- Dataset: 142 ride hours, 511 labelled windows, anonymized and time-stamped
- Feature Set: 18 kinematic features (variance of velocity, RMS jerk, peak angular velocity, etc.)
- Model Configuration: k = 7, Euclidean distance, 10-fold stratified cross-validation
- Hardware: Android devices (minimum SDK 29), Snapdragon 778G
- Evaluation Protocol: 70:30 stratified split

The system that was proposed is applicable to everyday use in consumer-grade Android devices, but the performance of the system depends on the hardware specifications of the devices. Although the experiment was conducted using Snapdragon 778G devices, the performance of lower-end devices might be affected, resulting in latency and battery consumption. In terms of generalization, the dataset was collected mainly under urban Thai traffic conditions. Although the kinematic feature set is vehicle-independent at a fundamental level, driving styles might vary in rural, highway, and international conditions. Failure handling mechanisms have also been integrated to ensure the reliability of the system. Watchdog monitors have been integrated for irregularities in telemetry data, and heuristic fallback thresholds have also been provided for consecutive failure of inference mechanisms. Loss of GPS signals and sensor dropouts have also been addressed through interpolation techniques. For the scalability of the system, the proposed edge cloud architecture has provided the benefits of decentralized inference and centralized model updates through OTA mechanisms, making the system suitable for resource-constrained environments.

5. Conclusion

In the present study, the design and development of an accident risk prediction system for high-risk motorcyclists in Thailand, as per the country's Vision Zero initiative, has been discussed. This study has been conducted using smartphones and energy-saving machine learning algorithms along with an ISO-compatible software development methodology. Thus, the accuracy and uninterruptedness of the risk prediction and behavioral feedback can be achieved. Furthermore, the design of the proposed system has been developed to include data acquisition, low-latency prediction, and cloud data synchronization. This has ensured the credibility of the data and the security of the system. Thus, the assessment of the proposed system on 300 users has demonstrated excellent results in terms of accuracy and satisfaction. On a concluding note, the present study has provided a comprehensive and viable solution to mobile-based road safety solutions. Future enhancements to the proposed project include the study of advanced modeling concepts along with integration capabilities beyond mobile devices, such as smart watches.

Acknowledgments

We would like to thank Rajamangala University of Technology Thanyaburi (RMUTT) for their valuable support and encouragement throughout this research.

References

[1] Jittima Wongwuttiwat, Thacha Lawanna, and Tanakom Tantontrakul, "The State of Digital Technology and Innovation Development: The Comparative Position of Thailand in ASEAN," *The Electronic Journal of Information Systems in Developing Countries*, vol. 90, no. 4, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[2] Witaya Chadbunchachai et al., "Road Traffic Injuries in Thailand: Current Situation," *The Journal of Medical Association Thailand*, vol. 95, no. 7, pp. 274-281, 2012. [Google Scholar]

[3] Padma Iyenghar, Emil Gracic, and Gregor Pawelke, "A Systematic Approach to Enhancing ISO 26262 with Machine Learning-Specific Life Cycle Phases and Testing Methods," *IEEE Access*, vol. 12, pp. 179600-179627, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[4] ISO, ISO/IEC/IEEE 29119-1:2013(en), Software and Systems Engineering — Software Testing — Part 1: Concepts and Definitions, ISO, 2022. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:29119:-1:ed-1:v1:en

[5] Rüdiger Wirth, and Jochen Hipp, "CRISP-DM: Towards a Standard Process Model for Data Mining," *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, vol. 1, pp. 1-11, 2000. [Google Scholar]

[6] Karrouchi Mohammed et al., "Intelligent Driver Monitoring System: An Internet of Things-based System for Tracking and Identifying the Driving Behavior," *Computer Standards and Interfaces*, vol. 84, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Alexey Kashevnik et al., "Cloud-based Driver Monitoring System using a Smartphone," *IEEE Sensors Journal*, vol. 20, no. 12, pp. 6701-6715, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Renjia Wang et al., "Smartphone Sensors-based Abnormal Driving Behaviors Detection: Serial-Feature Network," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 15719-15728, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] Ehsan Khosravi et al., "Safe Deep Driving Behavior Detection (S3D)," *IEEE Access*, vol. 10, pp. 113827-113838, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] Junqing Li et al., "Road Roughness Detection based on Discrete Kalman Filter Model with Driving Vibration Data Input," *International Journal of Pavement Research and Technology*, vol. 18, no. 2, pp. 480-492, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Rohith Vallabhaneni et al., "Secured Web Application based on CapsuleNet and OWASP in the Cloud," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 35, no. 3, pp. 1924-1932, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Preeti Agarwal, and Mansaf Alam, "A Lightweight Deep Learning Model for Human Activity Recognition on Edge Devices," *Procedia Computer Science*, vol. 167, pp. 2364-2373, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[13] Deepti Mehrotra et al., "Multiclass Classification of Mobile Applications as Per Energy Consumption," *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 6, pp. 719-727, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Ankita et al., "An Efficient and Lightweight Deep Learning Model for Human Activity Recognition using Smartphones," *Sensors*, vol. 21, no. 11, pp. 1-17, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] Hazilah Mad Kaidi et al., "IoE-Powered Smartphone Feedback for Real-time Driver Improvement," *International Journal of Integrated Engineering*, vol. 16, no. 3, pp. 273-280, 2024. [Google Scholar] [Publisher Link]

[16] Ian Warren et al., "Behavior Change for Youth Drivers: Design and Development of a Smartphone-based App (BackPocketDriver)," *JMIR Formative Research*, vol. 2, no. 2, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[17] Sarven S. McLinton et al., "A 'Living Intervention': Evaluating A Real-Time Feedback System to Help Teams Co-Create Psychosocial Safety Climate," *Safety Science*, vol. 190, pp. 1-11, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[18] Komal Lochan Behera et al., "Analyzing the Effectiveness of Real-Time Feedback Systems in Occupational Health," *Health Leadership and Quality of Life*, vol. 2, 2023. [Google Scholar]

[19] Lukasz Jedrzejczyk et al., "On the Impact of Real-Time Feedback on Users' Behaviour in Mobile Location-Sharing Applications," *Proceedings of the Sixth Symposium on Usable Privacy and Security*, Association for Computing Machinery, New York, NY, United States, pp. 1-12, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[20] Gibrail Islam, and Tim Storer, "A Case Study of Agile Software Development for Safety-Critical Systems Projects," *Reliability Engineering and System Safety*, vol. 200, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[21] Mario Barbareschi et al., "Scrum for Safety: An Agile Methodology for Safety-Critical Software Systems," *Software Quality Journal*, vol. 30, no. 4, pp. 1067-1088, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[22] Suat Kasap, Ersin Elbasi, and Milan Dordevic, "Design, Development, and Evaluation of a Mobile Application for Safety Engineers," *Journal of Infrastructure, Policy and Development*, vol. 8, no. 12, pp. 1-19, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[23] Oyekunle Claudius Oyeniran et al., "5G Technology and its Impact on Software Engineering: New Opportunities for Mobile Applications," *Computer Science and IT Research Journal*, vol. 4, no. 3, pp. 562-576, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[24] Yufeng Li et al., "Complying with ISO 26262 and ISO/SAE 21434: A Safety and Security Co-Analysis Method for Intelligent Connected Vehicle," *Sensors*, vol. 24, no. 6, pp. 1-29, 2024. [CrossRef] [Google Scholar] [Publisher Link]