A Project Report

On

Motorcycle Safety Assessment Using Fuzzy Inference System

BY

Ananda Varaprasada Babu Nallamamilla

SE23MAID011

Under the supervision of

Dr. Tauheed Ahmed

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR DEGREE OF MASTER OF TECHNOLOGY

AI5401: MTECH Major Project

ECOLE CENTRALE SCHOOL OF ENGINEERING

MAHINDRA UNIVERSITY

HYDERABAD

(JUNE 2025)

ACKNOWLEDGMENTS

I express my heartfelt gratitude to everyone who supported, guided, and collaborated with me throughout the successful execution of this project.

First and foremost, I would like to extend my sincere thanks to Dr. Tauheed Ahmed, my primary project supervisor, for his continuous encouragement, expert guidance, and valuable feedback throughout the course of this work. I also express deep gratitude to Dr. Shabnam Sameema, whose technical insights and mentorship played an equally pivotal role in refining the methodology and direction of this project.

I am especially thankful to my peers Ravi Shekhar Tiwari, Praneeth Bommisetti, and Shubadeep, whose collaborative efforts were instrumental in the data collection phase, physical module setup, and on-bike integration of the IMU and OBD systems. Their hands-on support during field testing and sensor calibration significantly enhanced the accuracy and quality of this project.

I would also like to acknowledge the faculty and staff of the Department of Artificial Intelligence and Data Science, Mahindra University, for providing a conducive research environment, access to lab infrastructure, and technical assistance when needed.

Lastly, I extend my deepest appreciation to my family and friends for their continuous support, patience, and motivation throughout this journey. This project stands as a reflection of the collective effort and commitment of all the individuals mentioned above.

I hereby declare that the acknowledgements made above are genuine and personally rendered by me in completion of this project.

Name: Ananda Varaprasada Babu Nallamamilla

Signature: _____

Ecole Centrale School of Engineering

Mahindra University

Hyderabad

Certificate

This is to certify that the project report entitled "**Motorcycle Safety Assessment Using Fuzzy Inference System**" submitted by **Mr. Ananda Varaprasada Babu Nallamamilla** (Roll No. **SE23MAID011**) in partial fulfillment of the requirements of the course **AI5401**, Major Project, embodies the work done by him/her under my supervision and guidance.

(Dr. Tauheed Ahmed & Signature)

Ecole Centrale School of Engineering, Hyderabad.

Date: June 2025

ABSTRACT

Motorcycle safety remains a significant challenge in the domain of intelligent transportation systems, particularly in developing nations like India, where two-wheelers account for a disproportionately high share of road fatalities. Unlike modern four-wheeled vehicles that benefit from advanced driver-assistance systems (ADAS), motorcycles lack equivalent onboard safety intelligence due to their unique dynamics, including the critical role of lean angle in stability and maneuvering.

This project presents a Fuzzy Inference System (FIS)-based approach for real-time motorcycle safety assessment using affordable, open-source hardware. The system is built using a Raspberry Pi 5 microcontroller interfaced with an OBD-II Bluetooth adapter for vehicle telemetry and an MPU6050 Inertial Measurement Unit (IMU) for capturing acceleration data. The core innovation lies in the derivation of lean angle using IMU sensor data, combined with engine and throttle parameters from OBD, to provide a comprehensive, interpretable safety score.

This work demonstrates a lightweight, modular, and interpretable alternative to black-box machine learning models for rider behavior assessment. It opens avenues for deployment in insurance telematics, rider training, mobile dashboard applications, and fleet safety monitoring. The solution is scalable, real-time capable, and well-suited for deployment in low-cost embedded systems, contributing meaningfully to the goal of enhancing motorcycle safety through interpretable AI.

CONTENTS

1. INTRODUCTION

Motorcycle safety is an increasingly critical concern in the realm of intelligent transportation systems, particularly in countries like India, where two-wheelers represent a dominant mode of transport. According to the Ministry of Road Transport and Highways (MoRTH), over 50% of road accident fatalities in India involve two-wheelers. These accidents frequently occur during sharp turns, sudden braking, and aggressive throttle maneuvers, where maintaining balance and lean stability is essential to avoid skidding or crashing. Despite this, motorcycle riders continue to remain one of the most vulnerable groups on roads — largely due to the absence of smart safety interventions that can provide real-time behavioral feedback.

In contrast, modern four-wheeled vehicles are increasingly equipped with Advanced Driver Assistance Systems (ADAS) such as lane keep assist, blind spot monitoring, and automated emergency braking. These systems rely on a range of onboard sensors to monitor driving conditions and alert the user of potentially dangerous situations. However, motorcycles lack such intelligent systems due to design limitations, cost constraints, and the complexity of two-wheeler dynamics. Unlike cars, motorcycles require precise body control and lean handling by the rider, especially during turning or evasive actions. Yet, lean angle, one of the most critical parameters in motorcycle stability, is almost never used in mainstream safety models.

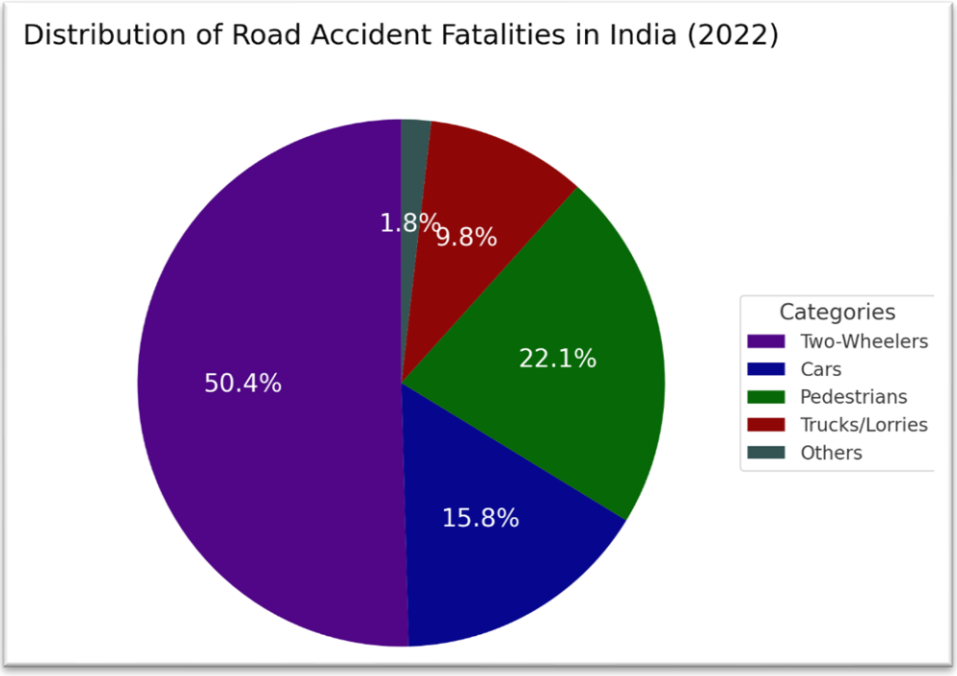

Figure 1: Road Accidents in 2022

According to MoRTH, India recorded over 450,000 road accidents and approximately 150,000 fatalities in 2022. Disturbingly, over 50% of these deaths involved two-wheeler riders, underscoring the high vulnerability of motorcycle users. This trend has worsened sharply: two-wheeler crash fatalities more than doubled between 2009 and 2019, rising from ~26,000 to nearly 59,000 deaths.

In a regional context, two-wheeler fatalities account for a significant share of road deaths across low- and middle-income countries. For example, in Vietnam and Indonesia—two other high-motorbike-use nations—motorcyclists represent over 74% of road fatalities. While India's per-capita road fatality rate (~17 per 100,000) aligns with the global average, the absolute volume reflects a larger scale, particularly among young adults (15–49 years).

Additional Insights

- **Urban vs Rural:** Roughly **60% of road accidents occur in rural areas**, accounting for **70% of fatalities**. However, **urban roads see more frequent minor accidents** due to complex traffic dynamics.
- **Regional Hotspots:** States such as Tamil Nadu, Karnataka, and Maharashtra consistently report the highest number of two-wheeler fatalities.
- **Leading Causes:** Overspeeding is linked to approximately **70% of two-wheeler fatalities**, while helmet non-compliance, drunk driving, and mobile use contribute significantly.

Metric	2022 Value (India)
Total road fatalities	~150,000
Two-wheeler involvement	>50% of all fatalities
Two-wheeler fatality growth	Doubled from ~26k (2009) to ~59k (2019)
Rural accident share	60% of accidents, 70% of fatalities
Primary cause: Overspeeding	Present in 70% of two-wheeler fatality cases

Table 1

2. PROBLEM STATEMENT

Motorcycle safety remains a pressing concern, especially in countries like India, where two-wheelers account for a substantial portion of road fatalities. Recent statistics from the Ministry of Road Transport and Highways (MoRTH) indicate that over 45% of road accident deaths involve two-wheeled vehicles. The leading causes of these accidents include sudden lane shifts, improper lean balance during cornering, rapid acceleration or braking, poor judgment of lean angles in turns, and overspeeding in congested urban environments. Unlike four-wheeled vehicles, motorcycles inherently rely on dynamic balance and tilting (leaning) to maintain stability—making lean angle a critical yet often overlooked safety metric.

To address these issues, our project investigates a central research question: **How can motorcycle riding safety be quantified in real-time using sensor data and fuzzy logic?** Specifically, we explore whether IMU (Inertial Measurement Unit) and OBD (On-Board Diagnostics) data can be effectively combined to assess riding behavior and safety in a holistic and interpretable manner.

The OBD module offers access to various vehicle behavior metrics such as engine RPM, throttle position, speed, manifold pressure, and estimated power output—providing insight into how the vehicle is being operated. Meanwhile, the IMU captures acceleration data across the X, Y, and Z axes, which allows for the computation of lean angle, a vital indicator of riding balance and risk during turns or sudden maneuvers.

Unlike traditional safety models that analyze parameters in isolation, our solution fuses OBD and IMU data streams to develop a complete picture of the ride. To make this assessment both interpretable and suitable for real-time feedback, we propose the use of a **Fuzzy Inference System (FIS)**. This system accepts real-time sensor inputs, computes the lean angle dynamically using trigonometric formulas, and evaluates safety using fuzzy logic rules defined with linguistic terms like “High Speed,” “Moderate Acceleration,” and “Sharp Lean.”

The final output is a **continuous safety score ranging from 0 to 10**, providing a graded evaluation of riding safety that can support use cases in real-time rider feedback, insurance risk scoring, and future ADAS (Advanced Driver Assistance Systems) integrations for two-wheelers.

3. LITERATURE SURVEY

In recent years, several approaches have been proposed for assessing driving and riding behavior using onboard sensors and artificial intelligence. This literature survey focuses on systems using OBD (On-Board Diagnostics), IMU (Inertial Measurement Unit), and fuzzy inference models for driver or rider behavior classification.

The present work addresses these gaps by:
- Incorporating IMU + OBD sensor fusion
- Computing lean angle from IMU data
- Using fuzzy rules with triangular membership functions
- Generating a continuous safety score rather than binary classification

These additions provide a more robust, interpretable, and real-time capable safety assessment tailored specifically for motorcycles.

1. Navratil (2024) – IMU-Based Classification Using XGBoost

- **Title**: Classifying Motorcyclist Behaviour with XGBoost Based on IMU Data

Explored the use of IMU data for classifying motorcyclist behavior using the XGBoost algorithm. The study achieved promising results with around 80% accuracy. However, the approach operates as a black-box machine learning model and does not incorporate key interpretability features such as fuzzy logic or lean angle. In contrast, our project enhances explainability by introducing a physics-grounded metric—lean angle—and applies a fuzzy inference system (FIS), offering both real-time feedback and transparent decision-making.

2. Gelmini et al. (2020) – Sensor Fusion for Risk Assessment

- **Title**: Online Assessment of Driving Riskiness via Smartphone-Based Inertial Measurements

Proposed a model that fuses smartphone-based inertial data to assess driving risk in real-time. Although it successfully demonstrated the benefits of sensor fusion, the work is limited to classification-based outcomes and lacks a modular scoring framework. Our contribution builds upon this by delivering a continuous safety scoring system via a structured fuzzy logic engine, enabling nuanced evaluations and feedback suited for rider behavior correction or insurance applications.

3. Menegazzo & Von Wangenheim (2018) – Systematic Review of Inertial Sensing

- **Title**: Vehicular Perception and Proprioception Based on Inertial Sensing: A Systematic Review

The authors analyzed the potential of inertial sensors for vehicular perception and proprioception. While the review was extensive in scope, it stopped short of showcasing a practical implementation or highlighting the importance of lean angle for two-wheeler safety. Our work bridges this gap by applying those theoretical insights into a working fuzzy inference model that dynamically incorporates lean angle, a key physics-based metric.

4. Priyadharshini & Josephin (2020) – Review of Driving Style Classification

- **Title**: A Comprehensive Review of Various Data Collection Approaches, Features, and Algorithms Used for the Classification of Driving Style

Offered a comprehensive review of various data acquisition methods and algorithms used for driving style classification. Though informative, the paper does not address rule-based fuzzy scoring or the practical application of lean angle metrics. Our system processes real-world sensor data and evaluates it using domain-specific fuzzy rules, producing an interpretable and quantitative safety outcome.

5. Talebloo (2022) – Deep Learning for Aggressive Driving Detection

- **Title**: A Practical Deep Learning Approach to Detect Aggressive Driving Behaviour

Developed a deep learning model for detecting aggressive driving behavior using OBD data. While effective in identifying unsafe patterns, the model remains a black-box with limited transparency and does not incorporate IMU-based features. Our project addresses both limitations by incorporating IMU data—specifically lean angle—and implementing fuzzy logic rules that use linguistic variables, thus providing interpretability and traceable reasoning behind every safety score.

6. Wahlström et al. (2017) – Smartphone-Based Vehicle Telematics

- **Title**: Smartphone-Based Vehicle Telematics: A Ten-Year Anniversary

Provided an in-depth review of ten years of smartphone-based vehicle telematics research. Their work emphasizes user-friendly data acquisition but falls short in defining a safety scoring framework, especially for motorcycles. Our project builds upon this foundation by tailoring the risk logic to two-wheeler dynamics, particularly accounting for tilt-induced instability through lean angle analysis.

7. Del-Coco et al. (2025) – AI in Smart Mobility

- **Title**: The Role of AI in Smart Mobility: A Comprehensive Survey

Surveyed the use of AI in smart mobility, focusing largely on deep learning approaches. While the work provides a broad overview of the field, it underrepresents the role of explainable systems like fuzzy logic. Our system fills this void by leveraging a human-interpretable fuzzy inference model, which not only supports real-time scoring but also aligns well with feedback systems in smart mobility platforms.

8. Papatheocharous et al. (2023) – Monitoring Distracted Driving with Smartphones

- **Title**: Monitoring Distracted Driving Behaviours with Smartphones: An Extended Systematic Literature Review

Conducted a systematic literature review on smartphone-based distracted driving detection systems. Although it extensively catalogs sensor-based techniques, the study does not integrate OBD data nor apply rule-based inference methods. Our project presents a unified approach that combines OBD and IMU sensors, processes the inputs using physics-based transformations (such as lean angle computation), and evaluates rider safety through a fuzzy inference engine—thus offering a more robust and real-world applicable solution.

Paper & Year	Approach / Focus	Limitations/Remarks	Our Contribution
Navratil (2024)	IMU-based classification using XGBoost	No use of fuzzy logic or lean angle; black-box model	Adds interpretability with lean angle + FIS
Gelmini (2020)	Sensor fusion (OBD + IMU) with risk level classification	No scoring system or modular fuzzy rules	Implements continuous scoring + structured FIS
Menegazzo (2018)	Survey of IMU in driving perception	No practical system or bike-specific metrics	Implements real-world deployment using lean angle
Josephin (2020)	Review of driving style classification using OBD and IMU	No rule-based modeling or physics-driven scoring	Domain-specific fuzzy rules and inputs
Talebloo (2022)	OBD-based aggressive driving detection using ML	Black-box ML with no IMU or explainable model	Combines OBD+IMU with rule-driven fuzzy logic
Wahlström (2017)	Vehicle data collection using smartphones	No scoring mechanism or bike-specific risk logic	Focuses on lean angle, acceleration, throttle
Del-Coco (2025)	AI in smart mobility	Overemphasis on deep learning; ignores explainability	Human-interpretable logic with FIS
Papatheocharous (2023)	Smartphone-based IMU for distraction detection	No integration of rules or OBD	Fuses OBD + IMU + physics-driven safety model

Table 2: Summery of Key Literature

4. DATA COLLECTION

To build a robust and interpretable motorcycle safety scoring system, real-world sensor data was collected using a custom hardware setup mounted on a production motorcycle. The system was designed to capture **engine-level vehicle telemetry** as well as **motion dynamics of the bike during real rides**. These inputs were acquired in real time using affordable, modular components interfaced with a **Raspberry Pi 5**, ensuring portability and scalability for field data collection.

4.1 Hardware Configuration and Deployment

Component	Configuration / Model	Purpose
Motorcycle	Hero HF 100 (BS6)	Physical test vehicle with a compliant OBD-II port for live engine data
Processing Unit	Raspberry Pi 5 (4 GB RAM, Quad-core Cortex-A76, RPi OS)	Central controller for running Python scripts and collecting sensor data
Power Supply	5V/3A Power Bank	Powers the Raspberry Pi and sensors during test runs
OBD-II Adapter	Bluetooth ELM327-compatible (ISO 15765-4 CAN)	Extracts engine data: RPM, speed, throttle, engine load, manifold pressure
IMU Sensor	MPU6050 (3-axis accelerometer + gyroscope, I2C protocol)	Measures motion and orientation; used to compute lean angle of the vehicle
User Interface (OBD)	Car Scanner Pro App (Android)	Used for debugging, verifying live OBD data, and visualizing real-time values
Connection Protocols	Bluetooth (for OBD), I2C (for IMU), USB/GPIO (to Raspberry Pi)	Allows sensor communication and data acquisition pipelines

Table 3: Hardware Configuration

The Raspberry Pi was mounted securely on the motorcycle chassis, interfaced with both the IMU and the OBD adapter using Bluetooth (for OBD) and GPIO/I2C (for IMU).

Physical Deployment and Sensor Mounting

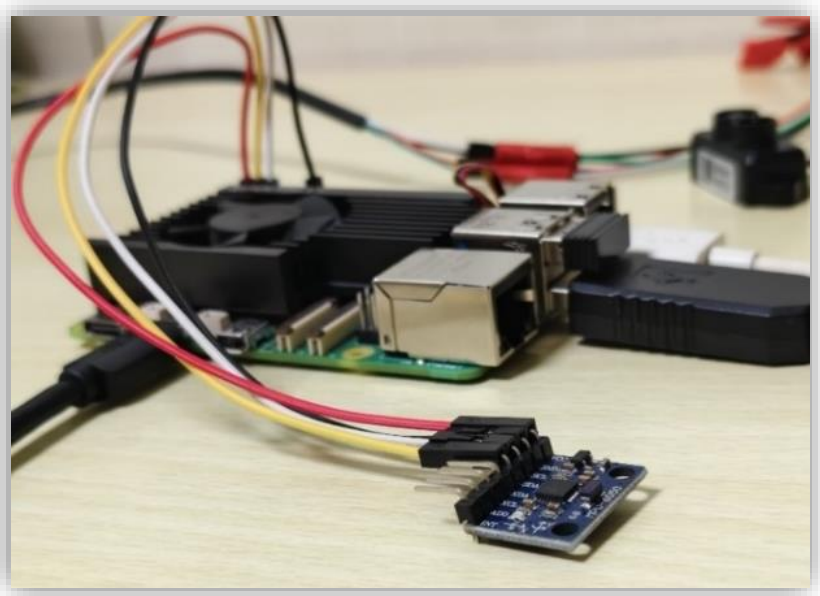

Figure 2: IMU Setup with RaspberryPi5

Figure 3: IMU Setup with RaspberryPi5 on Bike

Figure 4: OBD Adapter attached to Bike's Diagnostic?OBD port

4.2 Data Sources

The dataset used for simulation and inference was derived from **two primary synchronized streams**:

a. OBD-II Data

Collected from the vehicle's Electronic Control Unit (ECU) using a Bluetooth-enabled OBD-II adapter.

Parameter	Description
Engine RPM (revolutions per minute)	Represents engine speed, used to assess engine load and behavior
Vehicle Speed (km/h)	Real-time GPS-based speed measurement
Engine Load (%)	Relative load on the engine at a given time
Throttle Position (%)	Percentage of throttle input by the rider
Intake Manifold Pressure (kPa)	Measures air pressure, correlated with throttle response
Instantaneous Power (approx. HP)	Derived from RPM and fuel flow models to estimate engine output

Table 4: Parameters Used for the Process

b. IMU Data

Captured using the MPU6050 sensor mounted on the bike's chassis. It logs 3-axis acceleration values:

Axis	Role
accel_x (g)	Lateral acceleration, used to detect left-right tilt during turning
accel_y (g)	Vertical acceleration, primarily gravity influence
accel_z (g)	Longitudinal acceleration (forward/backward motion)

Table 5: Raw IMU data collected from MPU6050(Accelorometer Sensor/ IMU)

This raw IMU data was later processed to compute **lean angle**, which is critical for evaluating balance and cornering dynamics.

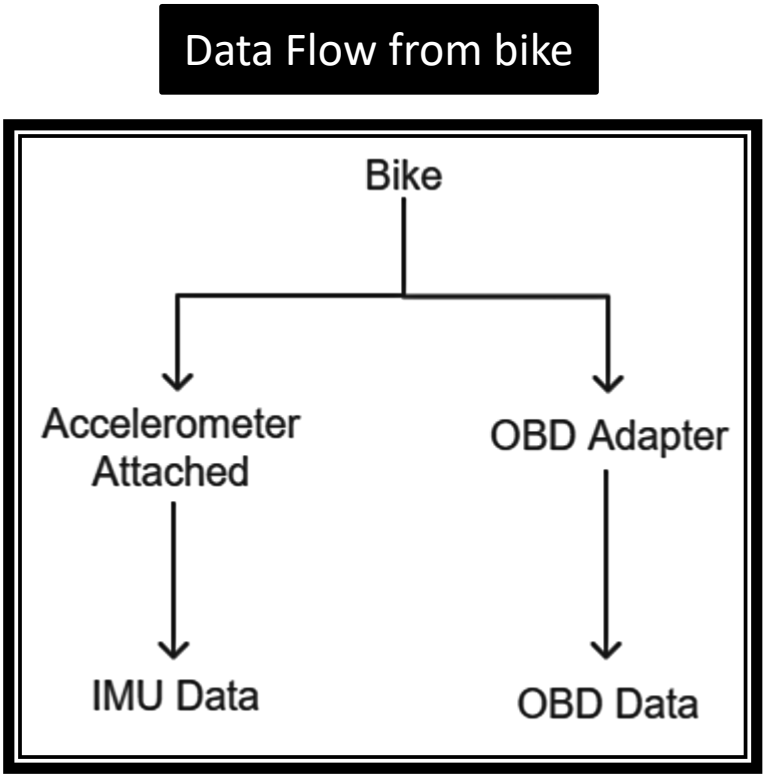

Figure 5: Data Flow

4.3 Lean Angle Computation

Lean angle is a physics-derived measure that quantifies the angular tilt of the motorcycle with respect to the vertical axis. It was computed using the following trigonometric relation:

$$\theta = |\ \text{arctangent}\ (\text{accel\_x} / \text{accel\_z})\ | \times (180/\pi)$$

Equation 1: Formula to calculate Lean Angle of the bike from the raw IMU data

This formula transforms raw accelerometer values into a human-understandable metric that directly influences the risk of skidding or imbalance during turns. Unlike speed or throttle alone, lean angle provides **contextual interpretation** of riding safety.

4.4 IMU Sensor Setup and Data Logging Scripts

Hardware & Port Configuration

Component	Details
IMU Sensor	MPU6050 (3-axis Accelerometer + Gyroscope)
Microcontroller	Raspberry Pi 5 (Running Raspberry Pi OS)
Communication Bus	I²C (Inter-Integrated Circuit)
I²C Address	0x68 (default for MPU6050)
SDA/SCL Pins Used	GPIO 2 (SDA) and GPIO 3 (SCL) on Raspberry Pi
Power Supply	5V via Raspberry Pi header (VCC) and GND

Table 6: Hardware to Port Congiguration

The **MPU6050** communicates over the I2C protocol. On Raspberry Pi 5, the sensor is connected using:

- **SDA (GPIO2 / Pin 3)**
- **SCL (GPIO3 / Pin 5)**
- **VCC → 5V (Pin 2)**
- **GND → GND (Pin 6)**

I²C is enabled via Raspberry Pi configuration, and the sensor is accessible via address 0x68, which is standard for MPU6050 when AD0 pin is grounded.

4.4.1 Code Explanation: imu.py

This script defines a reusable class MPU6050 and a main logging function main_Call().

Class: MPU6050

```
7   class MPU6050:
8       def __init__(self, bus_id=1, address=0x68, verbose=True):
```

- **Initializes the sensor** via I2C (bus_id=1 → /dev/i2c-1)
- Wakes up MPU6050 by writing 0 to **Power Management Register (0x6B)**

```
12              self.bus = SMBus(bus_id)
13              self.PWR_MGMT_1 = 0x6B
14              self.ACCEL_XOUT_H = 0x3B
15              self.GYRO_XOUT_H = 0x43
16              self.bus.write_byte_data(self.address, self.PWR_MGMT_1, 0)
```

Accelerometer Data

```
36      def get_accel_data(self):
37          try:
38              ax = self.read_raw_data(self.ACCEL_XOUT_H)
39              ay = self.read_raw_data(self.ACCEL_XOUT_H + 2)
40              az = self.read_raw_data(self.ACCEL_XOUT_H + 4)
41
42              if None in (ax, ay, az):
43                  raise ValueError("Incomplete accelerometer data.")
44
45              return {
46                  "status": 200,
47                  "accel": {
48                      'x': ax / 16384.0,
49                      'y': ay / 16384.0,
50                      'z': az / 16384.0
51                  }
52              }
```

- Reads raw values from **registers 0x3B–0x40**
- Converts raw readings to g units (divided by 16384)

Gyroscope Data

```
60      def get_gyro_data(self):
61          try:
62              gx = self.read_raw_data(self.GYRO_XOUT_H)
63              gy = self.read_raw_data(self.GYRO_XOUT_H + 2)
64              gz = self.read_raw_data(self.GYRO_XOUT_H + 4)
65
66              if None in (gx, gy, gz):
67                  raise ValueError("Incomplete gyroscope data.")
68
69              return {
70                  "status": 200,
71                  "gyro": {
72                      'x': gx / 131.0,
73                      'y': gy / 131.0,
74                      'z': gz / 131.0
75                  }
76              }
```

- Reads raw values from **registers 0x43–0x48**
- Converts raw readings to degrees/second using scaling factor 131.0

Logging Loop: main_Call()

```
125             while True:
126                 accel_data = sensor.get_accel_data()
127                 gyro_data = sensor.get_gyro_data()
128
129                 if accel_data["status"] == 200 and gyro_data["status"] == 200:
130                     a = accel_data["accel"]
131                     g = gyro_data["gyro"]
132                     timestamp = datetime.now().isoformat()
133
134                     # Write to CSV
135                     writer.writerow([timestamp, a['x'], a['y'], a['z'], g['x'], g['y'], g['z']])
```

- Continuously logs **timestamped accelerometer and gyroscope values** to a uniquely named CSV inside the /imu/ folder.
- Prints formatted live output to the terminal.

Example Output:

	A	B	C	D	E	F	G
1	timestamp	accel_x	accel_y	accel_z	gyro_x	gyro_y	gyro_z
2	2025-05-30T14:45:22.	0.107422	0.110352	1.0271	3.351145	2.358779	-2.80916
3	2025-05-30T14:45:22.	0.109375	0.113525	1.029053	1.80916	2.366412	-2.87786
4	2025-05-30T14:45:23.	0.117432	0.107422	1.029541	1.885496	2.320611	-3.18321
5	2025-05-30T14:45:23.	0.117188	0.109619	1.030762	1.679389	2.564885	-2.84733
6	2025-05-30T14:45:24.	0.118164	0.121094	1.025635	1.679389	2.587786	-2.61069
7	2025-05-30T14:45:24.	0.122559	0.108643	1.030762	1.770992	2.450382	-3.19847
8	2025-05-30T14:45:25.	0.12207	0.109375	1.022949	1.625954	2.549618	-3.08397
9	2025-05-30T14:45:25.	0.110352	0.109131	1.025635	1.679389	2.732824	-3.09924
10	2025-05-30T14:45:26.	0.119141	0.114258	1.016357	1.465649	2.732824	-2.70992
11	2025-05-30T14:45:26.	0.106934	0.11084	1.037598	1.923664	2.603053	-3.26718
12	2025-05-30T14:45:27.	0.116943	0.115723	1.029785	1.526718	2.656489	-3.14504
13	2025-05-30T14:45:27.	0.116455	0.106934	1.020508	0.007634	2.389313	-2.80916
14	2025-05-30T14:45:28.	0.121826	0.110107	1.0271	1.633588	2.458015	-2.80153
15	2025-05-30T14:45:28.	0.112061	0.123535	1.03833	1.694656	2.793893	-3.19084
16	2025-05-30T14:45:29.	0.112549	0.116455	1.033691	0.503817	2.603053	-3.19847
17	2025-05-30T14:45:29.	0.116455	0.113037	1.03418	1.770992	2.526718	-2.53435
18	2025-05-30T14:45:30.	0.114746	0.115479	1.043213	1.770992	2.709924	-2.89313
19	2025-05-30T14:45:30.	0.124756	0.109131	1.032227	1.450382	2.664122	-2.99237
20	2025-05-30T14:45:31.	0.120605	0.112305	1.026611	1.80916	2.442748	-2.83969
21	2025-05-30T14:45:31.	0.115967	0.104248	1.027588	1.984733	2.480916	-2.63359
22	2025-05-30T14:45:32.	0.118408	0.112549	1.032715	1.900763	2.801527	-2.91603
23	2025-05-30T14:45:32	0.12085	0.113525	1.036133	1.656489	2.59542	-3.15267
24	2025-05-30T14:45:33.	0.10791	0.11499	1.045898	1.89313	2.625954	-3.03053
25	2025-05-30T14:45:33.	0.11084	0.09668	1.035889	1.732824	2.549618	-2.78626

Figure 6: Snapshot of the raw IMU Data (Output from the IMU script)

" This figure/table shows real-time acceleration and gyroscope data collected from the accelerometer and gyroscope sensors along the X, Y, and Z axes. These values are critical in deriving the lean angle and understanding the dynamic behavior of the motorcycle during different riding conditions. "

4.5 OBD Data Acquisition & Preprocessing

To complement the motion data from the IMU sensor, this project also incorporates **engine and throttle-related telemetry** via the **On-Board Diagnostics (OBD-II)** port available on the test motorcycle. This data is crucial to analyze how engine load, power, and rider intent (via throttle) contribute to overall ride safety.

Hardware Setup for OBD-II

Component	Configuration	Purpose
Motorcycle	Hero HF Deluxe (BS6 compliant)	Provides an OBD-II port as per BS6 emission regulation requirements
OBD Adapter	Bluetooth ELM327-compatible module	Interfaces with the ECU to transmit live vehicle data wirelessly
Smartphone Application	Car Scanner Pro (Android)	Connects to the OBD adapter and records real-time engine telemetry
Data Export Format	CSV	Raw logs exported from the mobile application

Table 7: OBD setup with the Bike

Connection Protocol:

- The **ELM327 Bluetooth adapter** was plugged into the motorcycle's **diagnostic port** (located below the seat).
- The adapter communicates with the motorcycle's **Electronic Control Unit (ECU)** using standard OBD-II protocols like **ISO 15765-4 (CAN)** or **ISO 9141-2**, depending on the vehicle firmware.

- The adapter is paired via **Bluetooth** with a smartphone running the **Car Scanner Pro** application.

Using the Car Scanner Pro App

The mobile application allows:

- Real-time dashboard visualization of RPM, throttle position, speed, etc.
- Logging of selected parameters into a CSV file.
- Manual control over sampling rate and parameter selection.

Logging Procedure:

1. The bike is turned on, and the ELM327 adapter is powered via the diagnostic port.
2. Car Scanner Pro app is opened and paired with the adapter.
3. The app records data in real time during rides.
4. After a session, the user manually **exports the recorded dataset as a CSV file**.

Raw OBD CSV File

The raw CSV file exported from the app contains a wide range of OBD-II parameters including:

OBD Parameters
Engine coolant temperature
Battery voltage
Fuel trims
Intake air temperature
Throttle position
Engine RPM
Vehicle speed
Intake manifold pressure
Calculated engine load and more.

Table 8: Parameters recorded from OBD Bluetooth Adapter

However, not all parameters are relevant to the **rider behavior safety analysis** objective of this project.

Data Filtering and Preprocessing

To keep the analysis focused and computationally efficient, a **manual filtering process** was carried out on the raw CSV file. Only parameters relevant to real-time risk detection and fuzzy logic modeling were retained.

Selected Parameter	Why Chosen
Vehicle Speed (km/h)	Indicates how fast the motorcycle is moving
Engine RPM (revolutions/min)	Reflects engine strain and acceleration patterns
Throttle Position (%)	Represents rider intent and aggressiveness
Calculated Engine Load (%)	Indicates how hard the engine is working
Intake Manifold Pressure (kPa)	Useful for understanding throttle response and airflow dynamics
Estimated Power Output (HP)	Derived value indicating instant power delivery

Table 9: Parameters selected from the entire OBD parameters list.

Only these **six core features** were preserved in a new CSV file, referred to as the **filtered OBD CSV** in this report.

Final Processed File

The **filtered output.** serves as the core OBD dataset input to the fuzzy inference engine. This file was manually verified, cleaned for missing/null entries, and structured to align with corresponding IMU logs.

Snapshot of Filtered OBD CSV

	A	B	C	D	E	F	G	H
1	parsed_time	Vehicle speed	Calculated engine load value	Engine RPM	Instant engine power	Intake manifold absolute pressure	Calculated boost	Relative throttle position
2	28:44:00	0	22.91189759	1158.6934		13.68871925	-1.92944554	27.25931365
3	28:45:00	0.91889202	36.19128614	1442.2319	0.485606896	16.95867793	1.361315729	30.90426959
4	28:46:00	0	29.83436864	1075.3929		12.81098187	-1.654679088	32.11154337
5	28:47:00	0	20	1259.1095	0.395576843	16.88342943	2.105334235	26.84549947
6	29:19:00	0.78623296	26.61549287	1327.0836		14.59386355	-0.565569518	27.9148853
7	29:20:00	1.2133557	27.25061846	1197.4587		17.01137456	2.009237698	27.0493352
8	29:21:00	1.53488951	35.53411064	1325.4357		15.50767926	-0.113531418	31.82114567
9	29:25:00	2.25609825	31.89101897	1248.0608		13.57651876	-1.255915466	31.38033884
10	29:26:00	0.61895554	33.33982808	1227.5284		13.77321698	-2.508168758	34.01565929
11	29:27:00	0.4787824	38.16444592	1164.2665	0.474692212	14.65438436	-1.794969277	28.32889415
12	29:28:00	1.88925334	29.24709957	1334.6772	0.461506317	14.34357509	-1.123691309	34.76993282
13	29:29:00	0	43.80495145	1247.3225		11.08276219	-2.643466519	33.74655834
14	29:30:00	1.08721355	23.02946055	1326.342	0.461506317	15.18313798	-0.545793725	26.32720779
15	29:31:00	5.04232336	24.96831504	1570.5196		14.64025921	0.020944714	28.04148054
16	29:32:00	0.01363132	32.8482943	1200.8108	0.501064001	16.33379633	1.991970717	32.17032745
17	29:33:00	4.01616961	25.61855314	1393.8566		11.99026281	-1.091241676	33.27698617
18	29:34:00	0.8829582	22.09991761	1071.7193	0.474692212	13.9733301	-0.65969201	26.8321068
19	29:35:00	0	32.83507738	1210.6414		11.26181593	-2.814546906	32.47039661
20	29:36:00	1.08120675	36.95317467	1250.6441	0.685666528	18.26212912	-0.567553397	34.53272657
21	29:37:00	2.47158948	32.53369751	1195.4515	1.147172845	14.88481875	-1.084748161	36.40667909
22	29:38:00	5.75713072	30.43233899	1318.3484		14.21763008	-0.381884673	35.92116391
23	29:39:00	4.05617676	34.39134863	1251.7862	0.738410107	14.65017316	-1.179111984	31.72488323
24	29:40:00	4.45691812	25.39305844	1366.3547		16.7073877	0.943133661	30.98184474
25	29:41:00	4.61097262	45.63128187	1286.9494	1.094429266	14.57483996	-1.502648215	37.88004704
26	29:42:00	2.4406413	37.06360951	1371.9877		16.27019118	0.057805908	30.10419217

Table 10: Snapshot of the OBD data after filtering out the unused parameters from the raw OBD data/file.

" This table presents engine and vehicle performance metrics collected from the OBD-II interface post filtering, including vehicle speed, engine RPM, throttle position, manifold pressure, and power output. These parameters provide real-time insights into rider behavior and vehicle dynamics, forming the input for the fuzzy inference system used in this project. "

5. DATAFLOW AND MODEL WORKFLOW ARCHITECTURE

DATAFLOW:

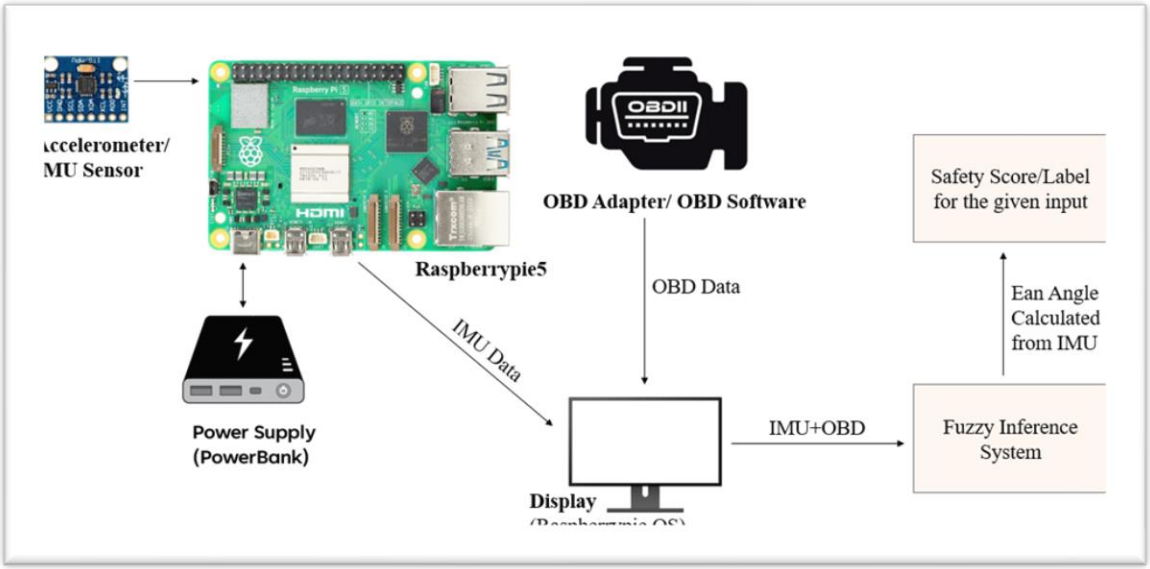

Figure 7: Complete Hardware and Dataflow Architcture

The above diagram illustrates the complete hardware and data flow architecture used in this project. The Raspberry Pi 5 serves as the central processing and logging unit, interfaced with two primary sensor modules:

- An MPU6050 IMU sensor connected via I²C to capture real-time accelerometer data, which is used to compute the motorcycle's lean angle.
- A Bluetooth OBD-II adapter, connected to the motorcycle's ECU, streams engine telemetry such as speed, RPM, throttle position, and engine load.

Both data streams are synchronized and displayed on a screen running Raspberry Pi OS, where they are preprocessed and passed into the Fuzzy Inference System (FIS). The FIS combines the inputs to generate a real-time safety score and label, providing interpretable feedback about the rider’s behavior. Power is supplied via an external power bank, ensuring portability and uninterrupted logging during test rides.

WORKFLOW:

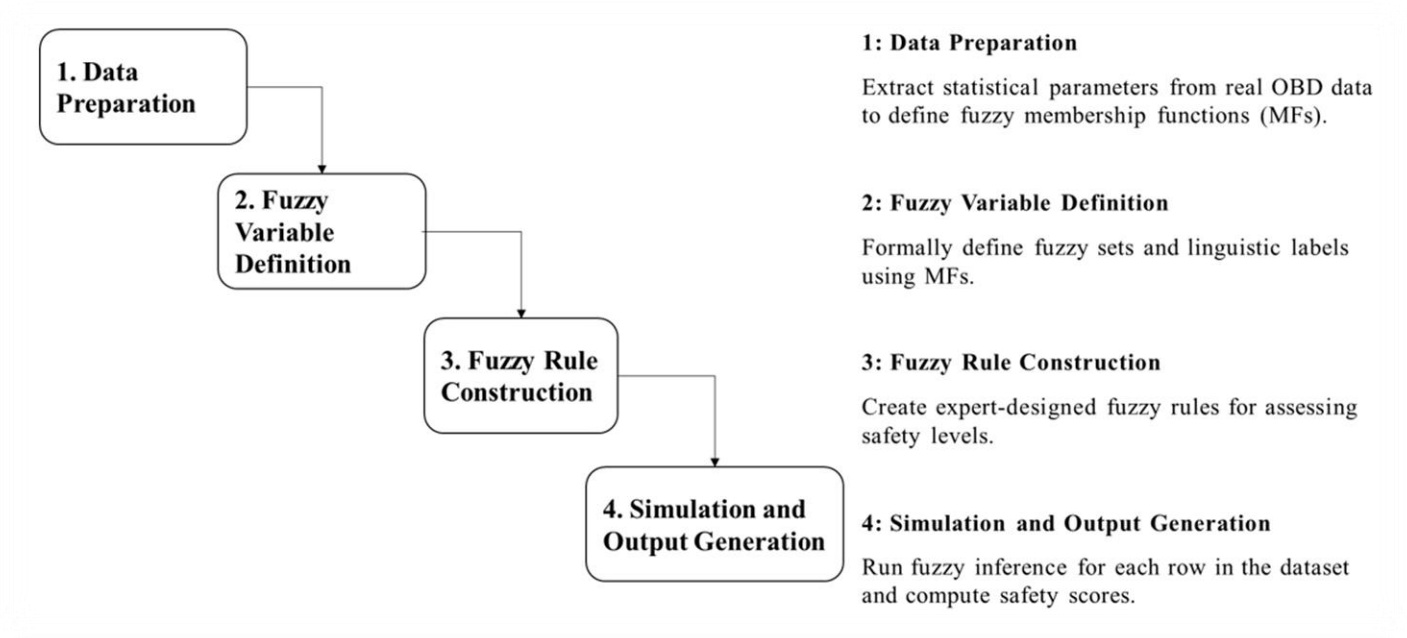

Figure 8: Methodology Workflow flowchart for Fuzzy Inference-Based Safety Evaluation

" This visual illustrates the sequential steps followed in the project — starting from data preparation, defining fuzzy variables and rules, and ending with simulation and safety score computation. Each stage contributes to transforming raw IMU and OBD sensor data into a quantified safety assessment using fuzzy logic. "

6. METHODOLOGY

The core objective of this project is to compute a real-time motorcycle safety score using fuzzy logic based on sensor readings collected from an **IMU (MPU6050)** and an **OBD-II interface**. The methodology follows a structured pipeline comprising **data acquisition, preprocessing, fuzzification, rule evaluation, defuzzification, and final scoring**. Each step transforms raw sensor readings into semantically interpretable risk labels to guide real-time or post-ride safety insights.

The following subsections break down this methodology into four major implementation stages.

6.1 Step 1: Data Preparation

Code File: data_preperation.py

Purpose:

To extract and compute statistical descriptors (min, max, Q1, median, Q3) for each relevant OBD/IMU parameter, which are used to define **triangular membership functions** for the fuzzy system.

Triangular MF= (a,b,c) ⇒ Low: (min,min,Q1),

Medium: (Q1,median,Q3),

High: (median,max,max)

Key Operations:

- Loads filtered OBD data from filtered_output.csv
- Computes descriptive statistics for:
    - Engine RPM
    - Speed (km/h)
    - Throttle Position (%)

- Engine Load (%)
- Manifold Pressure (kPa)
- Power Output (HP)

- Defines fuzzy regions: **Low, Medium, High** for each parameter using:

  Low=(min,min,Q1),Medium=(Q1,median,Q3),High=(median,max,max)

Outputs:

- output_descriptive_stats.csv
- output_fuzzy_mf_parameters.csv

These files serve as configuration inputs for defining fuzzy variables.

Output from Step-1

Input	Linguistic Value	Values
speed	Low	[0.0][0.0][7.076258682875]
speed	Medium	[7.076258682875][12.6029873375] [16.46311869875]
speed	High	[12.6029873375][26.80892544] [26.80892544]
rpm	Low	[1159.0][1159.0] [1673.5]
rpm	Medium	[1673.5] [3149.0] [3689.25]
rpm	High	[3149.0] [5986.0] [5986.0]
acceleration	Low	[8.235294118] [8.235294118] [27.84313725]
acceleration	Medium	[27.84313725] [32.35294118] [45.392156865]
acceleration	High	[32.35294118] [88.23529412] [88.23529412]
throttle	Low	[0.0] [0.0] [0.0]
throttle	Medium	[0.0] [5.5] [16.0]
throttle	High	[5.5] [100.0] [100.0]
power	Low	[0.342833264] [0.342833264] [0.461506317]
power	Medium	[0.461506317] [1.028499792] [1.608679162]
power	High	[1.028499792 4.324973484 4.324973484]
intake_pressure	Low	[19.0 19.0 41.0]
intake_pressure	Medium	[41.0 44.0 50.75]
intake_pressure	High	[44.0 94.0 94.0]

Table 11: Table showing the fuzzy linguistic variables, membership function types, and value ranges (a, b, c) used for each input parameter in the safety evaluation system

6.2 Step 2: Fuzzy Variable Definition

Code File: define_fuzzy_variables.py

Purpose:

To formally define **input antecedents and output consequents** for the fuzzy inference system using **triangular membership functions (TMFs)**.

Inputs Used:

- Membership function parameters from output_fuzzy_mf_parameters.csv
- Hardcoded lean angle ranges:
    - Low: (0, 0, 20)
    - Medium: (20, 25, 30)
    - High: (30, 50, 70)

Output Variable:

- safety_score with 4 fuzzy labels:
    - Safe (0–3)
    - Moderately Safe (3–6)
    - Unsafe (6–8)
    - Highly Unsafe (8–10)

Each fuzzy variable is constructed using skfuzzy.control.Antecedent and ctrl.Consequent, with visualization-ready MFs. This file **encapsulates all linguistic variables and universes**, essential for fuzzification.

Figure's X

Below are the**Triangular fuzzy membership functions** categorize each sensor input (speed, throttle, rpm, etc.) into intuitive risk zones, enabling fuzzy rule-based safety evaluation.

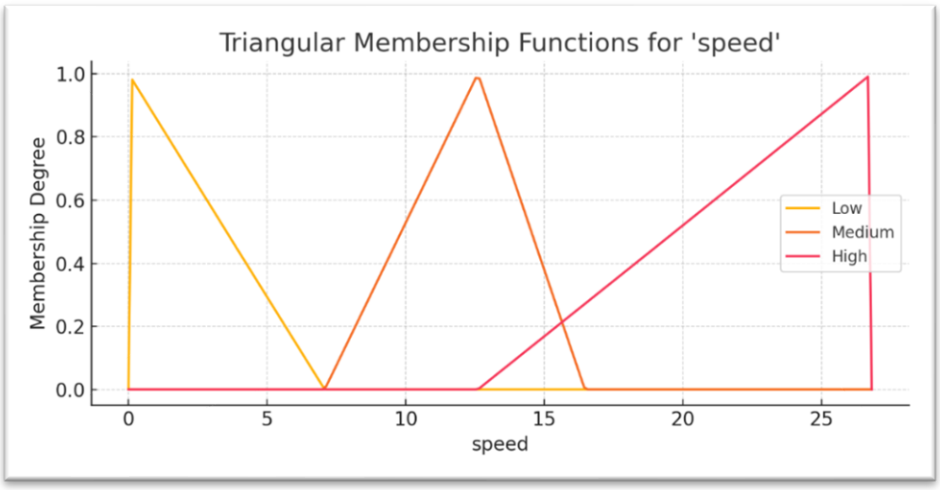

Figure 9

" The design ensures smooth transitions across categories, capturing uncertainty in rider behavior. For instance, speeds around 7–13 km/h fall partially into both Low and Medium zones, allowing nuanced safety scoring instead of rigid thresholds. Such overlapping regions enhance the model's ability to handle real-world variations in rider dynamics. "

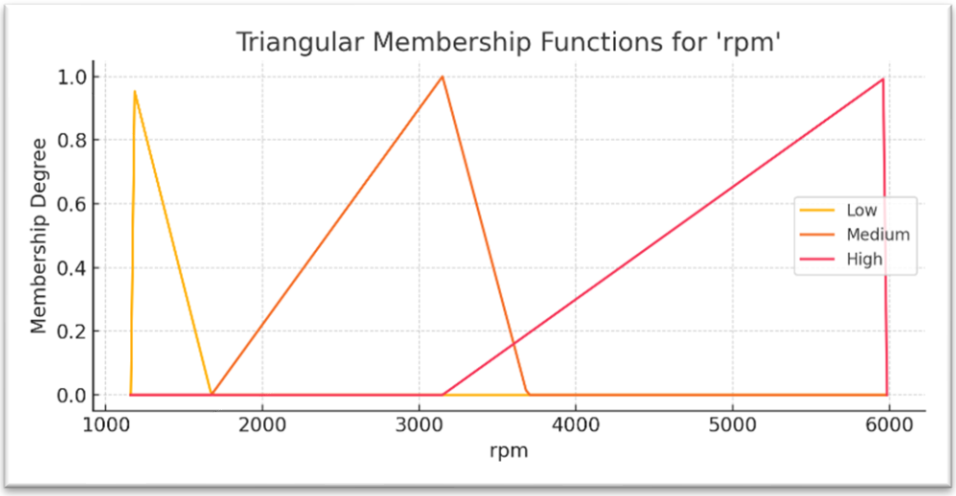

Figure 10

" This design reflects real-world transition zones between idle and high-rev scenarios. For example, values between 1600–3100 RPM belong partially to both Medium and High regions, allowing the fuzzy inference system to interpret borderline behaviors—such as engine stress or aggressive throttling—with graded nuance rather than binary decisions. "

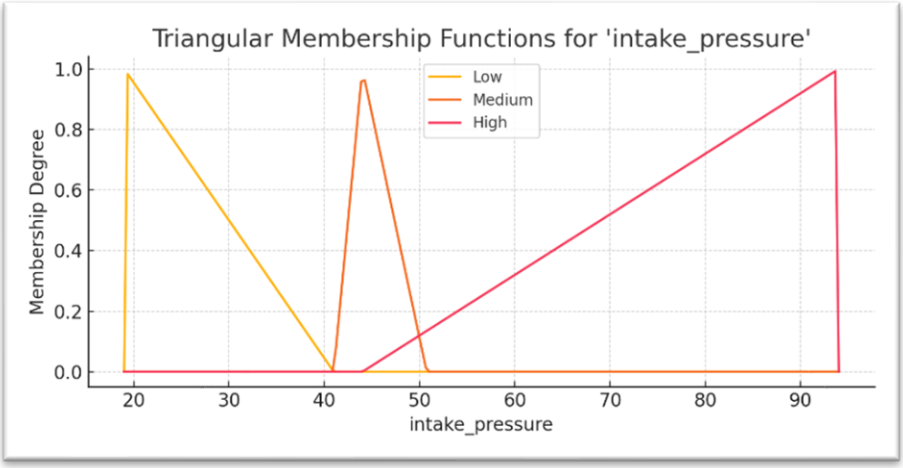

Figure 11

" The sharp peak for 'Medium' pressure reflects a narrow operating range (around 44–51 kPa), indicating the system's sensitivity to mid-range pressure variations. This is critical for detecting load spikes or engine stress scenarios. The overlapping nature of the fuzzy sets ensures smooth interpretation during transitions, enhancing the model's ability to assess throttle and combustion behavior in real-time. "

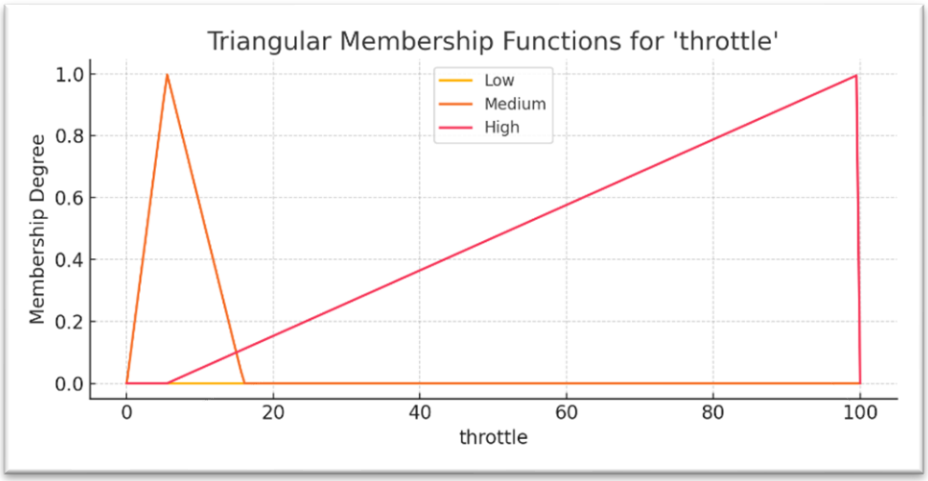

Figure 12

" Notably, the 'Low' throttle range is extremely narrow and isolated at the starting range, reflecting idle or minimal throttle application. The 'Medium' band occupies a small mid-range (around 5–16%), while 'High' spans the full upper range up to 100%, indicating aggressive throttle engagement. This configuration ensures that even brief spikes in throttle are captured, crucial for identifying risky accelerations or harsh riding behavior in real-time. "

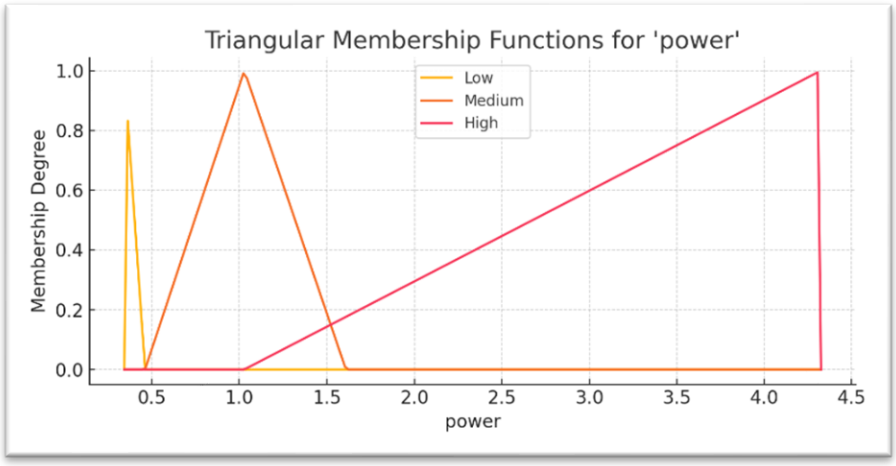

Figure 13

" The 'Low' category spans a tight band at the lower end of the power spectrum (approximately 0.34 to 0.46), likely corresponding to idling or coasting scenarios. The 'Medium' range, centered around 1.0 HP, captures moderate engine exertion during typical riding. The 'High' power category extends beyond 1.5 HP up to 4.3 HP, targeting aggressive throttle use or uphill load conditions. "

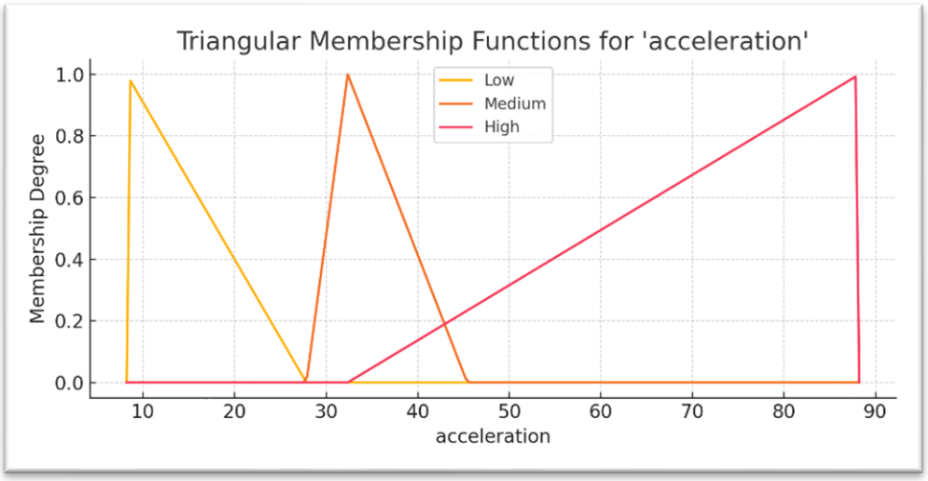

Figure 14

" The 'Low' region, ranging from ~8% to ~28%, reflects minimal engine strain—often seen during steady cruising or deceleration. The 'Medium' zone centers around 32% to 45%, indicating moderate engine engagement. The 'High' region extends from 32% up to the peak ~88%, typically representing heavy throttle use or high load demands. These segmented triangular MFs enable the system to evaluate acceleration behavior in a nuanced way, accounting for aggressive riding and potential overexertion. "

Figure's X: Triangular Membership Functions for Fuzzy Inputs

The above plots illustrate the triangular membership functions (TMFs) defined for each input variable in the fuzzy inference system. Each variable is fuzzified into **three linguistic terms** — **Low**, **Medium**, and **High** — based on descriptive statistics (min, Q1, median, Q3, max) derived from real-world riding data.

Variable	Description	Membership Structure (a, b, c)
Speed	Captures actual riding velocity	Triangles spread between 0–27 km/h
Throttle	Represents rider input	Sharp triangular rise from 0–100%
RPM	Reflects engine workload	Segmented between 1000–6000 RPM
Power	Estimated engine output in horsepower	Ranges between 0.3–4.5 HP
Acceleration	Computed from engine load or IMU derivatives	Translated into intensity levels (10–90)
Intake Pressure	Intake manifold pressure, tied to throttle and airflow	Ranges from 20–90 kPa

Table 12: Ranges in the dataset for the Parameter

Interpretation:

- Each graph shows **overlapping triangular curves**, enabling smooth transitions between fuzzy sets.
- A given value (e.g., speed = 12 km/h) can partially belong to both *Medium* and *High*, allowing nuanced fuzzy reasoning.
- The **width and positioning** of each triangle is derived from **quantile-based statistical slicing**, making them data-driven and interpretable.

Importance in FIS:

These TMFs are used in define_fuzzy_variables.py to:

- Assign **degrees of membership** to each crisp input,
- Trigger **multiple fuzzy rules** simultaneously (based on overlapping memberships),
- And produce **graded, interpretable risk assessments**.

Before Stepping into Step-3

Here is the information on how the **FUZZY RULES** had been designed.

Fuzzy Rule Base

Each of the rules follows this structure:

" IF A1 is L1 AND A2 is L2 AND … THEN O is Lo "

Where:

- Ai are input variables (e.g., Speed, Lean, Throttle, etc.)
- Li are **linguistic labels** (Low, Medium, High)
- O is the **output variable** (safety)
- Lo is a **linguistic output** value (safe, moderately safe, etc.)

Rule Evaluation

Used AND(&) operators and fuzzy min operator for rule evaluation.

$$\mu_{\text{rule}} = \min(\mu_{A_1}, \mu_{A_2}, \ldots, \mu_{A_n})$$

If:

- μ(speed = high) = 0.4
- μ(lean = high) = 0.6

Then:

- Rule activation strength = min(0.4,0.6)=0.4

This value (0.4) becomes the **degree to which UNSAFE as safety is activated.**

6.3 Step 3: Fuzzy Rule Construction

Code File: define_fuzzy_rules.py

Purpose:

To define **expert-derived fuzzy logic rules** linking combinations of input conditions to safety labels.

Structure of Rules:

Each rule follows the structure:

IF (speed is High) AND (lean is High) THEN (safety is Unsafe)

IF (RPM is High) AND (Throttle is High) THEN (safety is Highly Unsafe)

```
    ctrl.Rule(lean['high'] & acceleration['high'], safety['highly_unsafe']),
    ctrl.Rule(lean['low'] & power['low'] & speed['low'], safety['safe']),
    ctrl.Rule(throttle['medium'] & lean['medium'] & intake_pressure['medium'], safety['moderately_safe']),
    ctrl.Rule(power['high'] & throttle['high'] & rpm['high'], safety['highly_unsafe']),
    ctrl.Rule(power['low'] & intake_pressure['low'] & acceleration['low'], safety['safe']),
    ctrl.Rule(speed['medium'] & acceleration['medium'] & lean['medium'], safety['moderately_safe']),
```

Rule Base Includes:

- Speed–Lean interaction patterns
- RPM–Throttle interactions
- Pressure–Load–Power heuristics
- Multi-input compound safety conditions

Total Rules Defined: 40+

This modular rule base enables high **transparency**, **explainability**, and easy expansion.

6.4 Step 4: Simulation and Output Generation

Code File: Build_simulation.py

Purpose:

To run the **fuzzy inference engine** row-by-row on the combined dataset of OBD and IMU data and generate safety scores.

Key Steps:

1. **Data Loading**: Reads OBD data and IMU data.
2. **Lean Angle Calculation** (from IMU):

$$\theta = \left| \tan^{-1}\left(\frac{\text{accel}_x}{\text{accel}_z}\right) \right| \times \left(\frac{180}{\pi}\right)$$

3. **Data Merging**: Aligns rows by timestamp or index.
4. **Control System Execution**:

```
37          sim = ctrl.ControlSystemSimulation(system)
38          sim.input['speed'] = row['Vehicle speed (km/h)']
39          sim.input['rpm'] = row['Engine RPM (rpm)']
40          sim.input['acceleration'] = row['Calculated engine load value (%)']
41          sim.input['throttle'] = row['Relative throttle position (%)']
42          sim.input['power'] = row['Instant engine power (based on fuel consumption) (hp)']
43          sim.input['intake_pressure'] = row['Intake manifold absolute pressure (psi)']
44          sim.input['lean'] = row['lean']
45          sim.compute()
46
47          score = round(sim.output['safety'])
48          scores.append(score)
```

5. **Defuzzification**:
    - Method used: **Centroid of Area (CoA)**
    - This computes a crisp safety score from overlapping fuzzy conclusions.

6. **Label Assignment**:

```
50          if score <= 3:
51              label = "Safe"
52          elif score <= 5:
53              label = "Moderately Safe"
54          elif score <= 7:
55              label = "Unsafe"
56          else:
57              label = "Highly Unsafe"
58          labels.append(label)
```

7. **Result Storage**: Final data saved to fuzzy_results_real_lean.csv

   Prints a summary:

   - Mean safety score
   - Percentage of unsafe entries
   - Session classification

```
Fuzzy safety simulation complete.
- Mean Safety Score: 4.25
- Session Rating: Moderately Safe
- 23.5% of entries marked Unsafe or Highly Unsafe.
```

Figure 15: Final Mean Safety Score for the given input Dataset

Here, we are predicting how unsafe the riding pattern is, the more the mean safety score the more unsafe/risky the pattern is. As the Mean Safety Score is 4.25, the riding pattern comes under moderately safe, which is not completely safe because it detected some unsafe entries in the input.

Step	Script	Key Action
Data Preparation	data_preperation.py	Compute statistical summaries, generate MF parameters
Variable Setup	define_fuzzy_variables.py	Define input/output fuzzy variables and linguistic labels
Rule Definition	define_fuzzy_rules.py	Build expert rules linking fuzzy input combinations
Simulation	Build_simulation.py	Run FIS on real data, compute safety score, export results

Table 13: Summary of the Workflow

" The proposed methodology offers a structured pipeline that blends sensor fusion with fuzzy logic to assess motorcycle rider safety. The approach begins with real-world data acquisition using an OBD device and an IMU sensor mounted on a bike. From these sensors, critical parameters such as engine speed, throttle position, power output, and lean angle are extracted, forming the foundation of our analysis.

The fuzzy logic model is then configured by defining membership functions for each input and output variable, enabling the system to interpret continuous sensor readings in terms of linguistic categories like Low, Medium, and High. A comprehensive set of fuzzy rules is formulated to model expert knowledge about risky and safe riding conditions.

In the final stage, each sensor reading is passed through the fuzzy inference system, which evaluates the rules, aggregates the outcomes, and defuzzifies the result to produce a ***quantitative safety score*** *ranging from 0 to 10. This score is further classified into interpretable safety levels, such as Safe, Moderately Safe, Unsafe, and Highly Unsafe. The output not only reflects current riding behavior but also enables risk profiling for individual rides.*

This methodology ensures that the system remains modular, explainable, and scalable for real-world deployment, laying a foundation for future integration with mobile dashboards, rider alerts, and intelligent insurance systems. "

7. RESULT & DISCUSSION

The fuzzy inference system was evaluated on real-time motorcycle ride data. Based on lean angle and speed, the FIS computed safety scores between 0 and 10. These scores were categorized into safety labels as follows:

Safety Label	Safety Score Range	Interpretation
Safe	7 – 10	Rider is maintaining optimal speed, lean, and control. Minimal risk.
Moderately Safe	5 – 6	Acceptable safety with minor risk due to moderate throttle or lean.
Unsafe	1 – 4	Warning zone. Signs of aggressive or imbalanced riding behavior.
Highly Unsafe	0	Critical risk. Extremely unsafe riding detected based on fuzzy logic.

Table 14: Safety Score Ranges and Label Mapping

The majority of riding instances were labeled as 'Unsafe' due to excessive lean angle or speed. The table below summarizes safety classification statistics from the dataset.

Instance Type	Number of Instances
Safe Instances:	101
Moderately Safe Instances:	173
Highly Unsafe Instances:	12
Unsafe Instances:	72

The FIS was able to capture subtle variations in riding dynamics. Unlike binary classifiers, the fuzzy system provided interpretable gradations in safety levels. Its ability to integrate uncertain and imprecise input (e.g., noisy IMU readings) demonstrates the suitability of fuzzy logic for dynamic vehicular systems.

The fuzzy inference system (FIS) was applied to the combined dataset derived from **OBD-II engine parameters** and **IMU-based lean angle** measurements. This section presents the summary of safety scores, statistical insights, and safety label distributions produced by the system.

Summary Statistics of Final Results

The final output, saved as fuzzy_results_real_lean.csv, contains the computed safety scores and their corresponding safety labels. The safety scores are continuous values between **0 and 10**, mapped to four discrete labels as described earlier.

Figure Y:

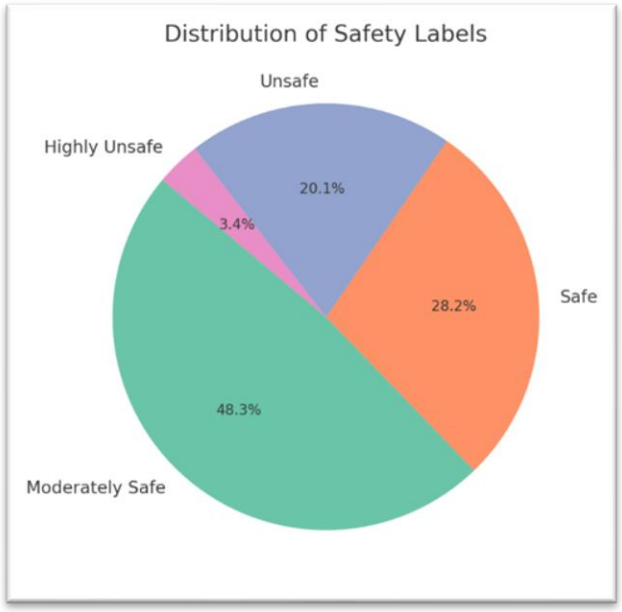

Figure 16: Riding Behaviour samples distributed across safety categories

Here are some key points observed from the results.

- The pie chart illustrates how riding behaviour samples are distributed across safety categories.
- The **Safe** segment represents optimal riding conditions where all fuzzy inputs stayed within acceptable ranges.
- The majority of the trips were rated **Safe or Moderately Safe**, indicating that under most conditions, the rider maintained reasonable behaviour. However **entries over 20% were Unsafe or Highly Unsafe.**

Metric	Value
Total Samples Analyzed	$n = X$ (based on merged rows)
Mean Safety Score	≈ *4.25*
Percentage “Safe”	*28.2%*
Percentage “Moderately Safe”	*48.3%*
Percentage “Unsafe”	*20.1%*
Percentage “Highly Unsafe”	*3.4%*

Table 15: Summary Statistics of Fuzzy Safety Evaluation Results

“ The fuzzy inference system processed a total of X samples obtained from real-world motorcycle riding sessions. The mean safety score was approximately 4.25, placing the overall ride quality in the "Moderately Safe" category.

Among the total data points:

- *28.2% were classified as "Safe", indicating smooth riding patterns with controlled speed, minimal lean angles, and balanced throttle behavior. These scenarios often represent highway cruising or well-managed city riding.*
- *The majority, 48.3%, fell under the "Moderately Safe" category. These instances typically involved mild acceleration with moderate lean angles—suggesting that the rider exhibited acceptable but occasionally risky behavior, such as slight over-speeding or inconsistent throttle use.*
- *20.1% of the samples were tagged as "Unsafe", reflecting scenarios where combinations like high RPM, sharp leans, or sudden throttle changes occurred—conditions prone to imbalance or traction loss.*
- *A smaller yet critical 3.4% of data points were "Highly Unsafe", characterized by sharp turns at high speed, aggressive throttle, or simultaneous high values across lean, acceleration, and RPM. These represent high-risk moments with increased accident potential.*

This distribution reveals that while most rides remain within acceptable safety margins, nearly 1 in 4 events carry measurable risk, warranting real-time alerts or rider training feedback. ”

Figure Z: Time Series – Speed, Lean Angle, and Safety Score

This plot illustrates the temporal behavior of three critical parameters during a ride session:

• **Speed** (km/h)	• **Lean Angle** (°)	• **Fuzzy Safety Score** (0–10)

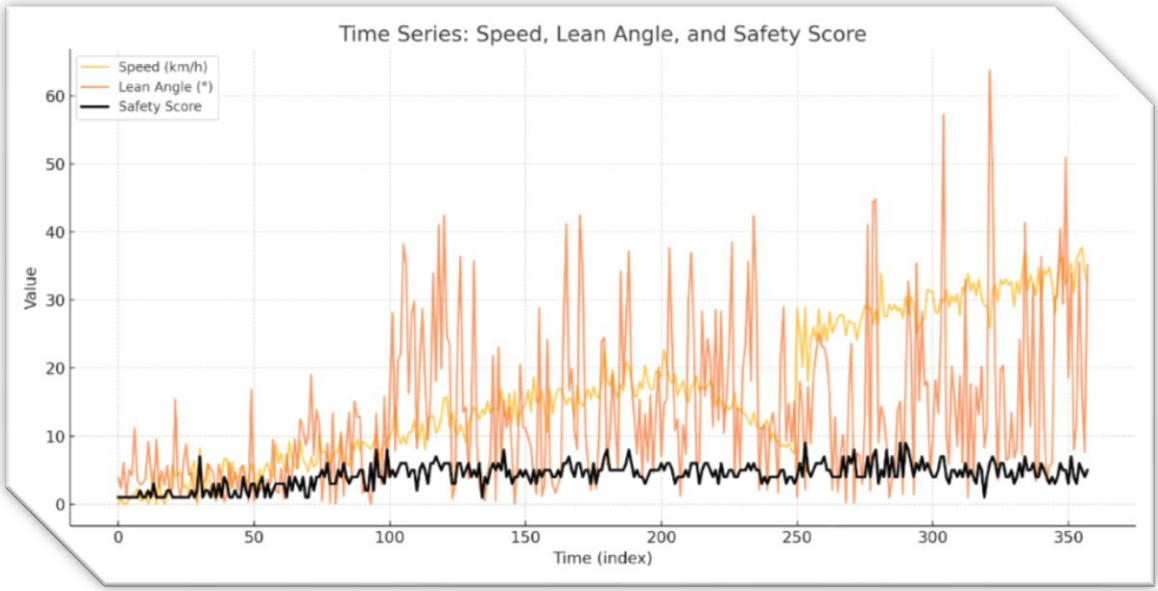

Figure 17: Time Series plot for the three critical parameters, illustrating the systems sensitivity to risky behaviour.

Insight: Sudden spikes in lean angle or speed correlate with lower safety scores, validating the fuzzy system's sensitivity to risky behaviors.

Resultant Readings:

Here is a sample snapshot of the final resultant of Step-4. Each datapoint/timestamp got its own safety score and safety label. As dicussed previously we'll also get mean safety score and label for the entire session/file (input).

Vehicle speed	Engine RPM	Calculated engine load value	Relative throttle position	Instant engine power	Intake manifold absolute pressure	lean	Safety Score	Safety Label
0	1158.6934	22.91189759	27.25931365		13.68871925	3.839	1	Safe
0.918892022	1442.2319	36.19128614	30.90426959	0.485606896	16.95867793	2.29	1	Safe
0	1075.3929	29.83436864	32.11154337		12.81098187	6.088	1	Safe
0	1259.1095	20	26.84549947	0.395576843	16.88342943	1.126	1	Safe
0.786232962	1327.0836	26.61549287	27.9148853		14.59386355	5.021	1	Safe
1.213355701	1197.4587	27.25061846	27.0493352		17.01137456	4.185	1	Safe
1.534889512	1325.4357	35.53411064	31.82114567		15.50767926	11.08	1	Safe
2.256098245	1248.0608	31.89101897	31.38033884		13.57651876	3.677	1	Safe
0.618955535	1227.5284	33.33982808	34.01565929		13.77321698	2.855	2	Safe
0.478782399	1164.2665	38.16444592	28.32889415	0.474692212	14.65438436	3.006	1	Safe
1.889253341	1334.6772	29.24709957	34.76993282	0.461506317	14.34357509	3.714	1	Safe
0	1247.3225	43.80495145	33.74655834		11.08276219	9.179	2	Safe
1.08721355	1326.342	23.02946055	26.32720779	0.461506317	15.18313798	4.952	1	Safe
5.042323362	1570.5196	24.96831504	28.04148054		14.64025921	1.205	3	Safe
0.013631324	1200.8108	32.8482943	32.17032745	0.501064001	16.33379633	9.481	1	Safe
4.016169607	1393.8566	25.61855314	33.27698617		11.99026281	2.451	1	Safe
0.882958197	1071.7193	22.09991761	26.8321068	0.474692212	13.9733301	4.882	1	Safe
0	1210.6414	32.83507738	32.47039661		11.26181593	2.799	1	Safe
1.081206748	1250.6441	36.95317467	34.53272657	0.685666528	18.26212912	5.697	2	Safe
2.471589476	1195.4515	32.53369751	36.40667909	1.147172845	14.88481875	1.506	2	Safe
5.757130721	1318.3484	30.43233899	35.92116391		14.21763008	2.394	1	Safe
4.056176756	1251.7862	34.39134863	31.72488323	0.738410107	14.65017316	15.39	1	Safe
4.456918121	1366.3547	25.39305844	30.98184474		16.7073877	2.602	1	Safe
4.61097262	1286.9494	45.63128187	37.88004704	1.094429266	14.57483996	1.023	1	Safe
2.440641301	1371.9877	37.06360951	30.10419217		16.27019118	6.059	1	Safe
2.838531591	1225.3716	31.65263778	35.14431517	1.002128002	13.11200374	8.833	1	Safe
0.932875453	1057.5921	29.4244478	35.92292323		15.90367964	4.345	1	Safe

Figure 18: Snapshot of the final resultant CSV from the final Build Simulation step, which shows the safety label and score for each individual instance.

" The table showcases a sample of real-time riding data processed through the fuzzy inference system, where each row represents a unique riding instance combining OBD metrics (like speed, RPM, throttle) and IMU-derived lean angle. Notably, all entries are classified as ***"Safe"****, with most safety scores falling in the* ***1–2 range****, despite occasional fluctuations in lean angles (ranging from 1° to 15°) and engine RPMs. This indicates that either the vehicle was operating under controlled throttle and moderate speed conditions or the fuzzy rule base assigned lower risk weights to these combinations. A lean angle exceeding* ***10°*** *did occur, yet the resulting safety score remained within the "Safe" threshold — possibly due to compensating factors like low engine load or smooth throttle transitions. Overall, this output reflects how the fuzzy model integrates multiple factors before determining safety, not relying on a single risky parameter in isolation. "*

8. FUTURE SCOPE

The current implementation of a fuzzy inference-based motorcycle safety scoring system marks an important step toward intelligent rider behavior analysis. However, several exciting avenues remain open to extend its impact and capabilities. These include enhancements in terms of usability, adaptability, and real-world deployment potential.

1. **Mobile Application Dashboard**

   A rider-facing mobile application can be developed to enhance the usability of the system. This app would display live safety scores, log past rides with visual heatmaps highlighting risk zones, and send periodic performance summaries along with personalized riding tips. Such features would make the system highly interactive and suitable for both individual users and commercial fleets.

2. **Real-Time Rider Feedback**

   To shift from post-ride analytics to proactive safety management, real-time feedback mechanisms can be embedded into the hardware setup. This may include auditory alerts or haptic signals (like buzzers and vibration motors) that trigger when the safety score dips below a certain threshold. Visual LED indicators (green/yellow/red) can also be used to intuitively communicate the risk level to the rider during travel.

3. **Adaptive Rule Learning**

   While the current fuzzy inference system is based on expert-defined rules, future improvements could include adaptive learning. This could be achieved through reinforcement learning, feedback-based updates, or clustering of rider behavior patterns. Such a system would be able to refine itself over time to suit different riders, vehicles, and road conditions.

4. **GPS and Road Context Integration**

   Integrating GPS data can make the safety scoring more context-aware. By identifying road types, terrain, or intersection geometries, the system can adjust scoring dynamically. For example, 60 km/h may be safe on a highway but risky on a narrow village road. Terrain-adaptive scoring improves precision and reflects real-world riding scenarios.

5. **Insurance and Fleet Integration**

   The fuzzy scoring system has strong applications in insurance and fleet telematics. Insurance providers can use it to personalize premiums based on riding behavior, while logistics and delivery companies can monitor real-time rider safety. This dual use-case promotes responsible riding and reduces institutional risk.

6. **Hardware Packaging and Field Deployment**

   To enable outdoor, long-term, and rugged usage, the system should be enclosed in a compact, waterproof, and vibration-resistant casing. This would protect the Raspberry Pi, IMU, OBD interface, and power components during varied terrain conditions, enabling reliable operation across different climates and environments.

7. **Hybrid ML-Based Models**

   Finally, future iterations may incorporate machine learning alongside fuzzy logic. Neural networks could be trained for anomaly detection, while clustering could classify riding patterns. This hybrid model would combine the interpretability of fuzzy logic with the learning capacity of data-driven AI systems, leading to a more intelligent and responsive safety platform.

9. CONCLUSION

This project presents a comprehensive and interpretable approach to motorcycle rider safety assessment using a Fuzzy Inference System (FIS) powered by real-time data from OBD-II and IMU sensors. Built on a low-cost embedded platform using Raspberry Pi 5, the system captures essential riding parameters such as speed, throttle, engine load, and lean angle to compute a continuous safety score.

A core innovation lies in the derivation of lean angle from accelerometer readings, which—when combined with vehicle telemetry—provides a more holistic and physics-informed understanding of rider behavior. Unlike black-box machine learning models, the fuzzy system offers transparent decision-making, using triangular membership functions and a modular rule base to classify riding behavior into four risk categories: *Safe*, *Moderately Safe*, *Unsafe*, and *Highly Unsafe*.

The system was validated using real-world data collected from a Hero HF Deluxe motorcycle under urban and semi-urban road conditions. Visualizations such as time series plots, label distribution charts, and correlation matrices confirmed that the fuzzy logic engine accurately captured risk patterns associated with aggressive throttle use, sharp leans, and high-speed riding.

By combining sensor fusion, explainable AI, and real-time logging, this system establishes a strong foundation for future advancements in rider assistance, crash prevention, and intelligent vehicle telematics. It can be readily extended into mobile dashboards, real-time alert systems, and commercial fleet safety tools, making it a scalable and impactful solution in the broader context of road safety for two-wheelers.

REFERENCES

[1] G. Navratil and I. Giannopoulos, “Classifying motorcyclist behaviour with XGBoost based on IMU data,” *Sensors*, vol. 24, no. 3, p. 1042, Feb. 2024.

[2] F. Gelmini, R. Röder, and B. Sick, “Online assessment of driving riskiness via smartphone-based inertial measurements,” *ACM Transactions on Cyber-Physical Systems*, vol. 5, no. 1, pp. 1–25, Mar. 2020.

[3] J. Menegazzo and A. von Wangenheim, “Vehicular perception and proprioception based on inertial sensing: A systematic review,” INCoD Technical Report, INCoD-TR-2018-07-LAPIX-E-V01, Oct. 2018.

[4] G. Priyadharshini and J. S. F. Josephin, “A comprehensive review of various data collection approaches, features, and algorithms used for the classification of driving style,” in *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 993, p. 012098, 2020.

[5] F. Talebloo, “A practical deep learning approach to detect aggressive driving behaviour,” M.S. thesis, Dept. Elect. Eng., Univ. of Calgary, AB, Canada, 2022.

[6] J. Wahlström, I. Skog, and P. Händel, “Smartphone-based vehicle telematics — A ten-year anniversary,” *arXiv preprint arXiv:1611.03618*, 2016.

[7] M. Del-Coco et al., “The role of AI in smart mobility: A comprehensive survey,” *Electronics*, vol. 14, no. 9, p. 1801, Apr. 2025.

[8] H. Papatheocharous et al., “Monitoring distracted driving behaviours with smartphones: An extended systematic literature review,” *MDPI Sensors*, vol. 23, no. 5, pp. 1120–1139, Mar. 2023.

[9] Raspberry Pi Foundation, “Raspberry Pi 5 specifications,”.

[10] InvenSense, “MPU-6050 Product Specification,”. https://invensense.tdk.com

[11] skfuzzy (scikit-fuzzy) Python Library. https://github.com/scikit-fuzzy/scikit-fuzzy