1-1-2021

Accurate Threshold Estimation For Electronic Control Unit (ECU) Signatures In Vehicular Environments

Niroop Sugunaraj

ACCURATE THRESHOLD ESTIMATION FOR ELECTRONIC CONTROL UNIT (ECU) SIGNATURES IN VEHICULAR ENVIRONMENTS

A Thesis
Presented to
The Academic Faculty

By

Niroop Sugunaraj

In Partial Fulfillment
of the Requirements for the Degree
Masters of Electrical Engineering in the
School of Electrical Engineering and Computer Science
College of Engineering and Mines

Grand Forks, North Dakota
December
2021

Name: Niroop Sugunaraj

Degree: Master of Engineering

This document, submitted in partial fulfillment of the requirements for the degree from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

DocuSigned by:
Prakash Ranganathan
2E9D28B343204C8...
Dr. Prakash Ranganathan

DocuSigned by:
Hossein Salehfar
83DF30182E4944C...
Dr. Hossein Salehfar

DocuSigned by:
Saleh Faruque
6D0F996935C5443...
Dr. Saleh Faruque

This document is being submitted by the appointed advisory committee as having met all the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

DocuSigned by:
Chris Nelson
2E0AF088C733403...
Chris Nelson
Dean of the School of Graduate Studies

12/3/2021
Date

PERMISSION

Title	Accurate Threshold Estimation for Electronic Control Unit (ECU) Signatures in Vehicular Environments
Department	Electrical Engineering
Degree	Master of Science (MS)

In presenting this thesis in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis work or, in his absence, by the Chairperson of the department or the dean of the School of Graduate Studies. It is understood that any copying or publication or other use of this thesis or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my thesis.

Niroop Sugunaraj

2^{nd} December 2021

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Dr. Prakash Ranganathan for his constant support, patience, and encouragement. His advice and critique has consistently pushed me to my intellectual boundaries. Much of what I've learnt during my program can be attributed to your expertise. I'm grateful for our academic interactions and the knowledge I've gained from you.

I'd like to thank the members of my thesis committee for their help in preparation of this work – Dr. Hossein Salehfar, with his vast expertise in power systems gave me a different perspective to further my work and Dr. Saleh Faruque who has supported my endeavors from the beginning.

Thanks are due to my friends and colleagues who made this work possible and supported me in every way they could, small or significant. Special regards to my parents who have always been a sounding board for my ideas and a major support system in my life. I owe much to both of you, mom and dad.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Rapid digitization of modern vehicles using electronic control units (ECUs) has made the modern automobile to realize autonomous operations. ECUs within a vehicle are capable of handling multiple functions within the vehicle pertaining to vehicular control, infotainment system, or electronic control of mirrors, wipers, and seats. Such data are relayed through various communication buses within a vehicle that allows for wired communication between multiple ECUs when referring to in-vehicle communication or wireless communication between a vehicle's ECUs and other vehicles and/or roadside infrastructure. Though network traffic can be intercepted from these communication buses, identifying an ECU responsible for a particular function is an open problem and is proprietary to auto manufacturers.

Present day automobiles are equipped with a myriad of functionalities that allows for automation and sensing capabilities through on-board sensors and their respective sub-systems. Accurate and timely relay of such data in ideal ambient conditions such as adequate light, absence of fog/haze, and objects that do not interrupt with sensor inputs is critical to the safety of the passengers and passersby.

While sensors on-board the vehicle are known for having fairly high lifespans, especially on higher-end vehicles, the data relayed through these sensors via communication buses can be intercepted and analyzed in order to identify usage patterns and make recommendations based on the observations. To obtain this data from an in-vehicle communication bus, this research uses Linux open-source tools and a cost effective commercial off-the-shelf (COTS) hardware to read streaming data from the Controller Area Network (CAN). Analyzing multiple vehicle types from various makes is ideal for robust training data to identify similar patterns between automobile manufacturers.

Therefore, the collected CAN data included are from 3 Nissan sedans, 1 Honda sports utility vehicle (SUV), and 1 Toyota sedan. ECUs responsible for functions related to the powertrain system of the vehicle, namely speed, tachometer, and steering were identified and mapped to actual values in miles per hour (mph), revolutions per minute (RPM), and newton-metre (Nm) respectively based on available original equipment manufacturer (OEM) technical manuals. Using waveform patterns for the speed ECU signature from Nissan, threshold equations were designed for Honda and Nissan makes using unsupervised learning methods i.e. k-means++ and mean shift algorithms. The results from these methods show an agreement between the two clustering methods based on descriptive statistical parameters and the lowest errors were obtained for the speed and steering ECU signatures for the 10-minute and 5-minute driving datasets respectively.

Manually identifying ECUs and their signatures is possible for a certain number of test vehicles but cannot work at scale. To automate this process, three supervised machine learning algorithms were identified and compared based on their performances to solve a categorical or classification problem. Comparisons were done using evaluation metrics, namely accuracy, F1-score, and computation time for two cases. The first two metrics are essential for assessing the performance of classification algorithms while the third metric will play a significant role if such algorithms are to be deployed in a real-time streaming environment. Results from the classification algorithms show that the distance based k nearest neighbor (kNN) algorithm gives the highest performance followed by the Decision Tree algorithm and lastly the Gaussian Naive Bayes. Cross validation was used to identify whether the higher performing algorithms i.e. kNN and Decision Tree are prone to underfitting or overfitting; results show that using 5-fold cross validation, both these models generalize fairly well enough during their training sets to produce accuracy and F1-scores that are higher than 70% and 0.65 respectively. These findings indicate that the non-linear models like kNN and Decision Tree show potential for identifying ECU signatures at scale.

CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 Introduction

This decade has seen an unprecedented rise in automobiles. From electronic vehicles to self-driving cars, the feature-rich capabilities of these vehicles seek to ensure comfort, performance, reliability, and particularly safety. According to a report by Deloitte [1], safety features within a vehicle such as the lane keep assist (LKA) and the adaptive cruise control (ACC) are still in a growth phase as they are typically implemented in high-end vehicles. These additional functions within a vehicle increase the semiconductor footprint and the need for analog-to-digital converters (ADC). Previously used functions such as adjusting seat positions or driver/passenger windows through mechanical means are now handled by electronic control units (ECUs) which offer greater degrees of freedom and more precise, fine-tuned usage. It has to be noted that the acronym ECU is sometimes used to refer to an engine control unit so the context of usage of should be referred to before enabling further discussion.

In a typical modern vehicle that has no autonomous function, ECUs digitize analog data read by sensors and relay these data using communication buses within a vehicle. Automotive communication buses that are prevalent in the modern automobile are controller area network (CAN), local interconnect network (LIN), media oriented systems transport (MOST), and ethernet. These various buses have varying operating speeds (bit rates) depending on the function(s) they handle. For instance, the CAN bus works at low (10 kbps - 125 kbps) and high (125 kbps - 1 Mbps) speeds [2]. All these buses or protocols are encapsulated within a diagnostics interface called the second generation onboard diagnostics (OBD-II). The OBD-II interface is a vehicle's self-diagnostics tool that produces a set of diagnostic trouble codes (DTCs) when a OBD-II reader is used to help technicians to address issues such as those related to internal circuits or electronics.

Semi-autonomous vehicles typically use ethernet as their mode of communcation on domain controllers to aggregate sensor data received by multiple, less powerful ECUs [3]. Domain controllers handle highly specialized functions such as active safety to obtain data from radar, camera, or LiDAR sensors to model the environment around the vehicle in order to make real-time decisions such as to autonomously apply the brakes or inform the driver in the presence of a risk. Limitations in modern automobiles present key challenges as outlined by Sagstetter et al. [4] and Han and colleagues [5]:

1. **Limited computation resources:** ECUs within a vehicle are automotive industry-grade micro-controllers that can handle computational workloads over IoT devices installed in smart home but are still constrained by their ability to ensure any cryptographic or security mechanisms to secure data relayed by the ECUs on the CAN or LIN buses. For instance, data transmitted on the CAN bus obscure the IDs of the transmitting nodes so there is no authentication mechanism in place to ensure the legitimacy of relayed data.

2. **Single point of access:** The OBD-II port is the central point of access (PoA) within a vehicle that provides direct exposure to some of the internal networks (like CAN) within the vehicle [6]. Though physical access is required to hack this interface, a threat actor can relatively easily perform passive (insertion of a monitoring device) or active (command injections to ECUs) actions to eavesdrop or sabotage the vehicle [7].

3. **Increased connectivity:** Wireless networks such as dedicated short range communication (DSRC), cellular, or wireless fidelity (WiFi) enables vehicles to communication with one another and exchange dynamic state information. An external device such as a mobile phone that is malware-infected can connect to such vehicles and corrupt the vehicle. According to a report by the Ponemon Institute [8], over 50% of the manufacturers surveyed said they plan to implement over-the-air (OTA) updates; OTA updates are sent remotely to patch vulnerabilities but can be wireless intercepted and exploited [9]. Currently, there isn't an automotive standard to secure OTA updates which further expands the attack surface to attackers.

With these factors in mind, threat actors can exploit communications that have few (such as the implementation of security gateway modules in the newer FCA automobiles from 2018 onwards) or no security mechanisms in place to handle threats. Such data can be monitored passively and reverse engineered to map ECU IDs with their respective functions and can then be subject to injection attacks from an external device such as a portable computer or eavesdropping on data to obtain the vehicle's owner personal information or driving habits.

1.2 Related Works

One of the key characteristics of data transmitted on the CAN bus is that the receiver and sender nodes are obscured: the source and destination ECUs cannot be identified explicitly. As such, one way to identify the ECUs broadcasting their data on the CAN bus is through bit-level analysis based on timing. Zhou and colleagues [10] identify a fingerprint for each ECU based on its clock characteristics such as frequency, skew, and synchronicity to detect masquerade and cloaking attacks. Statistical parameters in the time domain such as mean, variance, standard deviation, etc. were extracted as features for the IDS. The IDS produces a classification output based on the multinomial logistics regression (MLR) model which identified each ECU as its own entity. 11 ECUs in total were tested by the model (8 ECUs built-in and 3 ECUs were custom made and externally connected to the OBD-II port) and the precision produced by the model in identifying each of these ECUs was upwards of 98%. There isn't a defense mechanism by the IDS after identification of attacks are done.

Ning et al. [11] present an intrusion detection system (IDS) based on a concept called as local outlier factor (LOF) in order to classify different samples of the dataset under factors. Factors clustered close to one another indicate normal behaviour whereas outliers have factors that deviate from the cluster. Three types of attacks, namely, spoofing, bus-off, and physical attack to the ECUs were performed. Bus-off attacks are denial-of-service (DoS)-based attacks that exploit the functionality of the CAN protocol: during arbitration, when two ECUs made to have the same ID transmit dominant (bit 0) and recessive (bit 1) data bits at the same time, the ECU interprets the

bit it sent as opposite to the one it sent prior to arbitration, thereby increasing the error counter and shutting the ECU off from the network if it reaches a threshold. The support vector machine (SVM) model achieved an average identification rate of 87.9% for all three attacks.

Choi and colleagues [12] address the issue of the CAN protocol to authenticate data by proposing a solution that takes its premise on the inconsistencies in digital and analog output signals produced by the ECUs. Recording and using these signals as characteristics of ECUs associated with a particular vehicle allows the identification of malicious ECUs by their corresponding and unique output signal behaviours. The monitoring of ECU signals is performed by a monitoring unit that receives the same input signals multiple times to train and classify signal characteristics from a particular ECU as unique to that ECU. There were 17 time and frequency domain features that were used (e.g., skewness, mean, flatness, etc.) to train classification algorithms like the support vector machine (SVM), neural network (NN), and bagged decision tree (BDT) deployed on a PC running Matlab 2016a. These classifiers showed accuracies of 90% and above in identifying two adversary models: 1) When data is injected into the CAN bus via the hardware OBD-II port, and 2) Malicious data being transmitted by a "hostage" or hijacked ECU.

Park and Choi [13] present a machine learning-based intrusion detection system (IDS) to stop malware from propagating through self-driving vehicles to smart infrastructure or other connected vehicles. Specifically, the vehicle-to-device (V2D) attack vector is explored wherein an Android-based smartphone connects to the in-vehicle infotainment (IVI) system through wireless means and as such, the dataset for training and testing was obtained from the Android "AdWare and General Malware" database. The IDS is installed at the gateway between the Android device and the internal controller area network (CAN) within the vehicle and is based on three steps; firstly, the data is pre-processed through feature selection to pick out only the most relevant attributes to classify the malware. Secondly, the data are modeled through cross validation and trained/tested through a 75%/25% split respectively. Finally, the IDS classifies the behaviour of the software as malware, benign, or adware. Six machine learning algorithms are tested for their F1 score (the harmonic mean of the precision and recall metrics) accuracies in multi-tier classifications (benign,

malware, or adware) and binary classifications (benign or adware). Performance results show that given the strict real-time requirement of detection in operating vehicles, the algorithms had an average elapsed detection time of 0.049s for detection which is suitable according to the authors. Though gradient boost (GB) was determined to be suitable for binary classification, Random forest (RF) was shown to be the ideal candidate for the IDS due to its computational time of 19.4s for learning time and high accuracy of 93% in detecting malware. The hardware aspect of this analysis is left unaddressed.

King [14] deploys cryptography in the form of a hashed message authentication code (HMAC) to authenticate ECUs sending messages over the CAN bus. The author acknowledges that the CAN frame cannot accommodate a standard HMAC as the data field within a CAN frame can have only 8 bytes of data while a secure HMAC is at least 20 bytes in length. As a workaround, the solution here was to send three CAN messages (24 bits) and a timestamp. The HMAC is a SHA-1 based digest that is produced by hashing the data fields of the 3 messages and their associated timestamp. The test setup used Arduino microcontrollers and CAN bus shields and showed acceptable performance in decreasing the success of DoS and replay attacks.

Tackling the message authentication flaw within the CAN bus networks, Matsumoto et al. [15] present a self-identification method for ECUs to transmit error messages (in the form of bits) should the ECU be hijacked and to combat unauthorized data transmission. A few of the evaluation criteria used for this method include cost of implementation, real-time response, and detection accuracy. An authorized ECU listens on the CAN bus for a data frame sent from any other ECU. If the ECU successfully transmits its ID field by setting a flag, the ECU is considered to be authorized. If the flag is off, the ECU self-identifies this anomaly, sends an error frame, and overrides data transmission. The authors project that there is a 100% detection accuracy as long as the sender nodes are equipped with this method and the cost of implementation is low as it requires a simple modification within CAN controllers.

1.3 Problem Statement

From a data-driven standpoint, there are four fundamental limitations of the CAN 2.0 standard and are open research areas:

1. The data transmitted on the CAN bus is unencrypted and can be intercepted by anyone with the appropriate hardware and software tools. Particularly, one can listen on the CAN bus at specified bit rates to obtain information about the message primarily through its data and arbitration fields. While several solutions to encrypting this data are proposed, appropriate standards to enable encryption of CAN message frames are not available and this functionality is not being seriously considered.

2. Vehicle specific information such as time reliant sensor inputs/outputs, enabled functions within a vehicle during idle/driving states, and driving behaviour can be obtained from the vehicle if there is a method to translate the hexadecimal data in the data fields to original equipment manufacturer (OEM)-specific physical values. The manual process of identifying these data will be tedious and cannot work at scale thereby warranting a need for an automated method to identify the ECUs and their corresponding data payloads within a vehicle.

3. Valuable data obtained through a vehicle's OBD-II port is currently not being used but for the purposes of troubleshooting. These data can be leveraged to create driver profiles in order to map users' driving behaviours.

4. In addition to creating driving "behavioral" profiles, historical data from the tachometer, speedometer, and function usages such as ACC, braking, and light settings can be used to forecast or predict future behavioral tendencies for a particular driver.

1.4 Research Objectives

The objectives of this thesis was to investigate the information within raw CAN data through the use of machine learning data to identify underlying patterns using economical and open source tools. The objectives are threefold:

1. Gather CAN data from multiple vehicle makes and integrate different ECU signatures. The makes investigated were Nissan and Honda from the years 2010, 2013, 2015, and 2016.

2. Characterize and automate identified ECU signatures by exploring byte positions in CAN data frame. Perform byte position and magnitude analyses to identify operating ranges and waveform patterns.

3. Investigate the applicability of unsupervised (clustering) and supervised machine learning methods to group ECU signatures. Verify performance of algorithms through distance tests and multi-scenario analyses.

1.5 Paper Organization

The introduction of the research area is stated in Chapter I, providing also the problem statement and open areas for research in automotive CAN. Chapter II expands on the the internal and external communication technologies used in the automotive industry. Chapter III provides the sensor systems widely used in the modern automobile and their relevance to the CAN bus. Chapter IV provides threshold equations that attempt to accurately estimate the typical driving behaviour of a user using mathematical equations that are based on two unsupervised learning methods. Chapter V highlights the use of supervised machine learning algorithms to classify ECU signatures based on specific functions such as brake, lighting, steering, etc. Chapter VI offers concluding remarks and future work.

CHAPTER 2

REVIEW OF VEHICULAR COMMUNICATION PROTOCOLS

2.1 Introduction

An automobile consists of various subsystems that communicate internally and externally, and allow for efficient and critical operations in real-time. Categorizing the functional elements within an automobile into their respective subsystems allows for a clearer understanding of each modular element when looked at holistically. For the purposes of this paper, the framework proposed by Zeinab et al. [16] termed "AutoVSCC" will be used. Following the bottom-up approach, the fundamental layer to the automobile is the sensing layer which handles the "sensory" functions of a vehicle (see Figure 2.1). This includes ambient sensors such as RADAR, ultrasonic, LiDAR, camera, temperature, global positioning system (GPS), etc. or those that are responsible for optimizing engine control by controlling air/fuel intake such as pulse and oxygen sensors. Sensors essential for the basic operation of a vehicle are characteristic to the vehicles of the past while ambient sensors allow for better control of the vehicle, occupant comfort, and safety to surrounding entities.

The next layer is called the communication layer which handles the critical tasks of relaying data collected by the sensory units of a vehicle to various subsystems for further processing and action. Depending on the vehicle and its features, various subsystems for lighting, seating/mirror adjustments, airbags, tire pressure monitoring system (TPMS), cruise control (CC), etc. require different operating speeds based on the criticality and time-constraints of information being relayed. This is possible by setting operating speeds for the most prevalent communication protocols in the modern automobile. If drawing parallels to the common paradigm used in the information technology (IT) sphere with the Open Systems Interconnect (OSI) model, the communication layer can be mapped to the data-link, network, and transport levels of the OSI model.

The final layer in the hierarchy is the control layer that puts the data communicated by the lower layers to handle actuator control. For instance, inputs from the braking and LiDAR sensory subsystems are used to dynamically control the vehicle when adaptive cruise control (ACC) is enabled; this application has highly dynamic operating requirements and is dependent on real-time constraints in the driving environment. Similarly, the anti-lock braking system (ABS) enables the driver of the vehicle to possess/regain control of the vehicle in case of an emergency brake so the wheels of the vehicle do not lock thereby shutting off inputs from the steering wheel. The functions in this layer can further be mapped to whether it is an open or closed loop system. An open loop system is one where a given input will usually produce a set, predefined output. For instance, applying the brake on older automobiles will result in the vehicle slowing down as a direct response to the demand from the driver's seat. A closed loop system produces outputs based on a feedback loop that serves as the input. A good example of this would be the ACC function whether the vehicle dynamically shifts its acceleration/braking magnitudes to safely maneuver the vehicle.

With the increase in the number of electronically-controlled systems such as traction control, engine management (engine oil pressure monitoring, throttle position checking, engine temperature sensing and others), active suspension (real-time adjustments of the mechanical controls between a vehicle's chassis and the wheel axles), and multimedia all rely on broadcasting their data on the high speed CAN bus [17]. Another way to address the communication mechanism taking place between these systems is through the concept of multiplexing. Multiplexing of data in in-vehicle networks allows multiple subsystems within a vehicle to communicate with one another over the same signal path and while these signals can be read by all the control systems connected to the signal path, only signals relevant to a particular sub-unit will be used to produce desired responses. How these signals are timed, arbitrated based on ECU priority, and error checking and correction mechanisms are specified in the CAN protocol. However, the CAN protocol is one of five in-vehicle communication buses that are used in automobiles today.

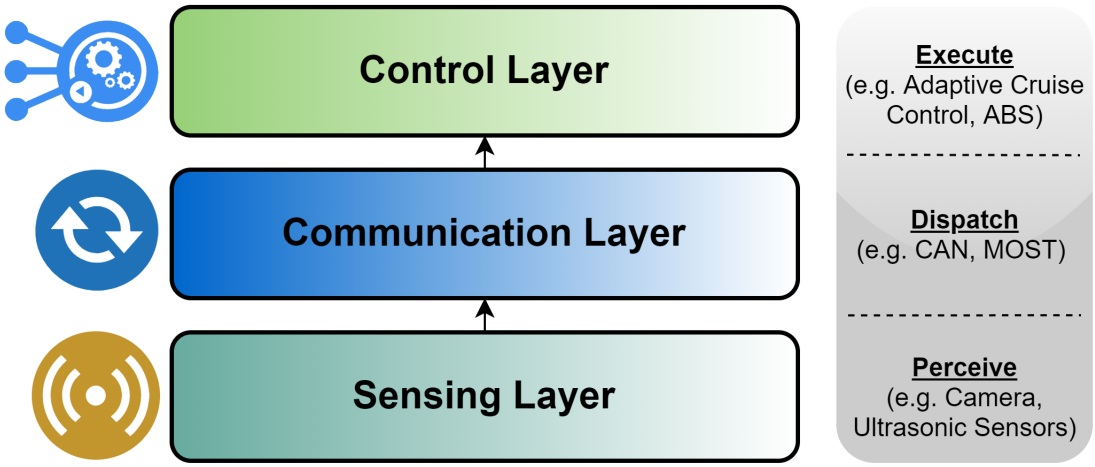


Figure 2.1: Three-layer operating framework for automobiles.

2.2 In-vehicle (Wired) Communication Technologies

In vehicle network (IVN) buses like LIN, MOST, FlexRay, and CAN work on what is commonly known as a central gateway architecture. This form of communication takes places when the aforementioned buses and their respective subsystems communicate with one another when they send their traffic to a gateway. There are two types of architectures for in-vehicle networks: centralized and backbone architectures. There are multiple methodologies to implement a centralized gateway [18]; for example, Seo et al.[19] categorize functions related to comfort, body control, real-time operations, and safety critical systems into 4 different classes which communicate with one another to enable information interchange between networks that are fundamentally heterogeneous in terms of the data they transmit and the functions they handle. The gateway proposed is based on the OSEK operating system (OS) and intuitively, places emphasis on fault tolerance (erroneous transmitting node due to electromagnetic interference (EMI)) and real-time operating benchmarks. A backbone architecture works with domain controllers that collect and relay data from domain-specific ECUs that are then sent to a main gateway to allow for connections to the external world such as vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications [20] and is a viable candidate for future vehicular applications.

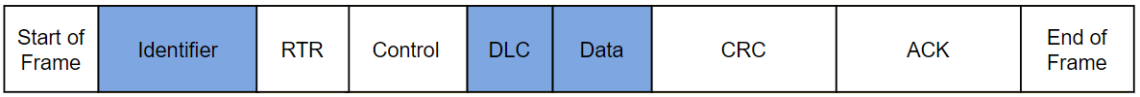


Figure 2.2: CAN frame structure.

2.2.1 CAN

The CAN network is a multi-master multiplexing bus that enables the ECUs within a vehicle to simultaneously broadcast their data on the bus. An important part of CAN's networks consists of ECU that handle critical functions within a vehicle based on a principle of event-driven communication: a priori commmunication between ECUs is not required and access to the bus is given based on event occurrence. The CAN uses a method of transmission called differential signalling with two signals called CAN-high (CAN-H) and CAN-low (CAN-L). During transmission, the bus is either at the "recessive" voltage (typically 2.5V) to transmit a binary 1 or at what is known as the "dominant" state when the CAN-H signal is pulled to 3.5V while the CAN-L is brought down to 1.5V. This differential in voltage represents a binary 0 and message frames starting with this dominant state are given a preference. This method of data transmission gives the CAN its key attributes of high integrity or robustness to noise and an arbitration-based mechanism that allows higher priority ECUs (or nodes) to transmit data first. A typical structure of a CAN message (or data frame) is shown in Figure 2.2. The fields highlighted in blue will be most relevant to analyses in the following chapters.

When a node begins transmission on the bus, it sends a dominant bit on the bus to initiate message relay (frame initiation). According to the CAN 2.0a standard, the arbitration field (also known as the identifier which is in hexadecimal) is a 11 bit segment that represents the unique identifier for the node. Arbitration or message priority is assigned based on this field. A remote tranmission request (RTR) field has a dominant bit (bit '0') if the frame is a data frame and a recessive bit (bit '1') if the frame is a remote frame based on the type of control field; the control field is a 6-bit identifier that specifies the type of message being transmitted: data frame (a typical type that transmits valuable data to multiple receivers), remote frame (solicits a data frame response

from a corresponding identifier), error frame (flag used when a node detects an error on the bus), and overload frame (which enables smooth data flow by requesting additional time delay before the transmission of a data or remote frame). The data field relays 0 - 64 bits of hexadecimal data that other nodes on the network can use to perform responsive actions. The next field is the cyclic redundancy check (CRC) field which is 15 bits in length and is used a checksum used when there error checking is required. This CRC is obtained as the remainder when all the bits from the start of frame till the end of the data field is added with 15 zeros and is divided by a generator as specified by the Bosch specification [21]. The fifth field called as acknowledgment (ACK) consists of two bits: the first bit is used by receivers to acknowledge that they have received the message by sending a dominant bit. The second bit is a delimiter that separates this field from the last field which is the end of frame that consists of 7 bits. There is also the extended CAN frame standard as specified by CAN2.0b that supports an extended (higher payload) identifier of 29 bits but is otherwise similar to the CAN2.0a. The classical CAN standard used in the modern automobile typically uses an operating baud rate of 1 Mbps which poses as a limitation to the increasing functionalities in the modern vehicle. As a workaround to this, the CAN-FD (flexible data rate) was introduced in 2012 and allows for higher operating baud rates of upto 8 Mbps and higher data payloads of upto 64 bytes.

2.2.2 LIN

The LIN protocol was originally developed the LIN consortium that consisted of manufacturers from Europe, namely Volkswagen, DaimlerChrysler, BMW, Audi, and Volvo. The consortium's objective with LIN was to identify a low cost universal asynchronous receiver-transmitter (UART) based communication bus that handles low-speed (typically 5 to 20 kbps) communication. According to Volvo [22], LIN serves as an alternative to CAN and is made to primarily handle non-critical operations such as the electronic control of windows, mirrors, or windshield wipers. In contrast to CAN, the LIN protocol follows a single-master and multi-slave (with a maximum of 16 nodes) architecture that allows for bidirectional communication and a physical interface that consists of

a single wire that is referenced to a common ground, and uses data payloads of 2-8 bytes. An important feature of the bus is the elimination of high accuracy clock sources to be used on member nodes [23]. This negates the expense of an external crystal or ceramic oscillator that serve well in circuits that are sensitive to timing requirements. Additionally, the slave ECUs on the LIN network regress to a "sleep" mode if not broadcasting any data for a specified period of time. The data sent, received, and used by these nodes are managed by the master node. The master node also maintains a schedule table [24] that provides information about which nodes are connected to the LIN bus, what messages can be sent, and the respective transmission time slots for each node on the bus. Like CAN, a LIN frame has two key elements in a data packet which are the header and message segments.

Master nodes synchronize the data received by these ECUs using a synchronize field within a LIN frame's header segment so that messages are received correctly. A synchronization interrupt (also called as a break) within the header is used to wake sleeping ECUs and an identifier is broadcasted to provide information about the contents of the message. The message segment in a LIN frame consists of a data field that carries bit-level information about the data payload and a checksum that is used to perform error checks in case of erroneous data transmission.

2.2.3 FlexRay

Similar to CAN in terms of its relevance in high-speed and critical applications, FlexRay is projected to be the next successor to CAN. It is most useful in high-speed vehicle control applications related to body control, chassis, or power train. The FlexRay protocol was jointly spearheaded by DaimlerChrysler and BMW to allow for a protocol that is deterministic and highly fault tolerant. FlexRay follows a dual channel architecture where each channel handles either an event driven response or a time driven response. An event driven response is when a node's response is expected prior to the happening of an event, such as gear shifting. A time driven response is expected when a message from the node has to be transmitted before a a specific time [24] i.e. each ECU has a time slot within which a response is required. These dual channels support data rates of 10 Mbps

each.

A FlexRay data frame consists of three distinct elements: header, trailer, and the payload. The header contains sub-elements of a 11-bit frame ID, a 7-bit payload length field, a header-specific CRC, and a cycle count. A FlexRay data frame can carry a payload of 254 bytes. The trailer element consists of 24-bit CRC that is frame-specific. Additionally, an unique transmission characteristic of the FlexRay protocol is what is known as a communication segment for each communication cycle. Nodes on a FlexRay network can follow a static segment or a dynamic segment. A static segment pre-allocates time slots for transmitting nodes hence giving the aspect of determinism to communicating nodes while a dynamic segment allows for more flexibility in transmission in terms of relay period(s) and payload length. There is another component called the sliding window which is used only when the FlexRay network needs to be managed. An important feature of the static segment is its ability to accommodate components from OEMs and their suppliers under subsystems that will continue to function effectively due to the timing requirements of the static segment [25]. However, the complexity and high costs related to the implementation of a FlexRay network should be weighed with its benefits prior to implementation.

2.2.4 MOST

A less critical but significant feature of the modern automobile is its ability to provide entertainment and/or more broadly infotainment for passenger comfort. The in-vehicle protocol well known for this is MOST founded by BMW and DaimlerChrysler. The MOST protocol is the de-facto communication bus for automobiles and allows for the relay of audio, video, and navigation data that fit into user-targeted applications like music players, video displays, and in-vehicle GPS respectively. Depending on the version of MOST being used (MOST25, MOST50, or MOST150), such data can be relayed at data rates of 25, 50, or 150 Mbps. In addition to being low cost and easy to implement, the MOST protocol's physical uses a plastic fibre optic (PFO) cable that is highly resistant to electromagnetic interference (EMI). A maximum of 64 nodes can be connected to the MOST bus and these nodes report a master node that needs to be determined beforehand. The master node is

called the “Timing Master” and controls the synchronization of messages between different nodes. Similarly, “Network”, “Connection”, and “Power” Masters handle network requirements, set up of synchronization channels, and the powering up and powering down of nodes respectively.[26]. Additionally, communication channels in MOST can handle three data types: synchronous (e.g. for real-time applications that rely on audio/video), control data (e.g. pressing of control buttons on the multimedia unit to switch on/off system) or asynchronous that handle data with varying bandwidths. These types also define a MOST data frame.

2.2.5 Ethernet

The last wired network that is in use today in modern automobiles albeit to a smaller degree is the automotive ethernet. Driven again by increasing complexity in today’s vehicles, the ethernet protocol was used not only to deliver adequate data rates to applications, but to also lower the costs of wiring harnesses used to implement the physical layer requirements of the network. As of 2021, the automotive ethernet physical layer works on 5 implementations: 100-BASET1, 1000-BASET1, 10-BASET1, and xGBASE-T1 where ‘x’ can be 2.5, 5, or 10 gigabits. The first three channels work on 100, 1000, and 10 megabit respectively. Depending on the channel, there are stringent design requirements that should be taken into consideration. Though Porter [27] suggests that 100-BASET1 can be implemented only using an unshielded twisted pair, the physical layer implementing 100-BASET1 is highly susceptible to EMI and requires proper shielding; shielding may not be necessary across the entire length of the bus but only for regions of high EMI such as those near an antenna subsystem. Additionally, the wiring harnesses should be tested in a realistic environment that brings parameters such as temperature, voltage, cable material, insulation, etc. to optimize the higher level data being relayed through the physical medium when deployed. An ethernet frame is much larger in size than its other counterparts with a maximum size of 1522 bytes. An ethernet frame has 6 fields: media access control (MAC) addresses for receiver and sender addresses, ether type, vlan tag, payload, and a CRC checksum.

2.3 External (Wireless) Communication Technologies

One of the key benefits of using wireless technologies not just intra-vehicle but also inter-vehicle is that it can meet highly mobile and dynamic requirements for real-time applications. For instance, vehicles moving at various speeds on roadways can function as sensors that relay traffic information and driving/road conditions to alert other drivers on the motorway [28] or communicate with roadside infrastructure such as tollbooths to wireless initiate payments. These applications are broadly defined under vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication respectively. Based on the use-case, applications such as traffic management or collision avoidance will have different requirements and considerations for the optimal technologies to implement have to be taken into account [29].

2.3.1 DSRC/Wave

Dedicated short range communication (DSRC) is a wireless communication technology derived from WiFi and provides low latency and high reliability to traffic safety or emergency services [30] primarily used in vehicle to infrastructure (V2I) applications. A dedicated bandwidth of 75MHz has been allocated for DSRC applications in the 5.850 - 5.925 GHz frequency. Apart from providing high data rates and requiring low power, there is flexibility in the data rates that can be used. Depending on the width of the channels (10 MHz or 20 MHz), a maximum of 27 and 54 Mbps respectively may be allocated. According to Cash [31], DSRC can be used to notify drivers in situations of public hazard such as work zone and road condition warnings, traffic information, optimal speed advisory, etc. in addition to private use cases such as toll collection, parking lot payments, and rental car processing. The U.S. Department of Transporation [32] provides an illustration of DSRC's interoperability with the other subsystems in the modern vehicle where a DSRC radio, cables, and antennae are needed to interface with the vehicle's GPS receiver and safety system. This safety system may connect with the vehicle's internal networks such as CAN or FlexRay to meet real-time and high speed requirements of DSRC.

Wireless access in vehicular environments (WAVE) is a technology that is based on 802.11p and serves as the foundation for DSRC. Developed by the Institute of Electrical and Electronics Engineers (IEEE), 802.11p defines the requirements of the physical and medium access control (MAC) layers [33] and its typically applications are network modeling for connected vehicles, routing decisions, network congestion monitoring, etc. It provides a maximum capacity of 27 Mbps, with a range of 10 - 300m, at a frequency of 5.850 - 5.925 GHz.

2.3.2 Cellular (4G LTE/5G)

Cellular technologies like 4G LTE and its evolved counterpart (5G) are becoming increasingly prominent in vehicle to everything (V2X) communications. A major benefit to using cellular communications for V2X applications like road safety and traffic management is that no existing infrastructure needs to be replaced. However, unlike technologies that can work with an internal clock for synchronization, LTE relies on synchronous communication with stringent clock requirements that may need expensive hardware. Though LTE is touted for having low latency and a peak data rates of 1 Gbps, it may not be appropriate for critical safety applications due to its speculated inability to accommodate resources to work at scale [34].

5G adds to the advantages of 4G LTE by providing higher data rates of 20 Gbps, low latencies of under 1ms, and can work at scale with 1 million devices per square kilometer [34]. It builds on existing multi-input-multi-output (MIMO) technology by adding more antennas (in the magnitudes of 10 or 100) within each MIMO system to enable increased capacity [35].

2.3.3 Zigbee

Zigbee is a low power (1 mW or less), lost cost technology developed by the Zigbee alliance and is based on the IEEE 802.15.4 standard. A Zigbee system consists of routers, end devices, and a coordinator where end devices can broadcast their data periodically in what is known as a beacon mode, or can broadcast their data at any point as long as the channel is free. Despite its low power requirements, it is able to transmit data at a distance of 150 m at data rates of 20, 40, and 250 kbps

[36]. This is due to its use of direct sequence spread spectrum (DSSS) which can negate the effects of interference and improve performance by increasing the bandwidth [37] in the Shannon-Hartley Theorem:

$$C = B \times Log_2(1 + \frac{S}{N}) \tag{2.1}$$

where C is the channel capacity (in bps), B is the bandwidth, and S/N is the signal-noise-ratio (SNR).

Zigbee can be used in non-critical applications and in V2I communications such as for identifying vacant parking spaces in high density parking lots [38] or for more dynamic applications such like speed control as proposed by Kochar and Supriya [39].

2.3.4 WiFi/WiMAX

Wireless Fidelity (WiFi) is a well established standard whose specifications are given in IEEE's 802.11 set of standards. Its ease of implementation and ubiquity in terms of hardware makes it a viable candidate for vehicular ad hoc networks (VANETs). Tufail and colleagues [40] assess the possibility of using WiFi in fast moving vehicles. Preliminary finds shows that WiFi data transfer rates decrease as relative speed between the vehicles increases and careful considerations should be made for the type of applications at different speeds.

The Worldwide Interoperability for Microwave Access (WiMAX) is a technology specified in IEEE 802.16 and is thought to have better throughput and a greater communication range than WiFi in most settings. Some findings indicate that though WiFi may have a higher throughput for a range of 200m, WiMAX supersedes its performance by making data transfer possible at a rate of 1 Mbps even at ranges of 1000m [41].

CHAPTER 3

VEHICULAR SENSORS

3.1 Introduction

This chapter will introduce the sensing technologies used in the modern automobile. These technologies enable a vehicle to operate with more safety and precision either autonomously/semi-autonomously and allow drivers to derive more information about the environment around them in addition to improving the occupants' comfort. To have a clearer conceptualization, the sensing technologies in this chapter will be divided according to the classification method proposed by Zeinab and colleagues [42] (see Figure 3.1):

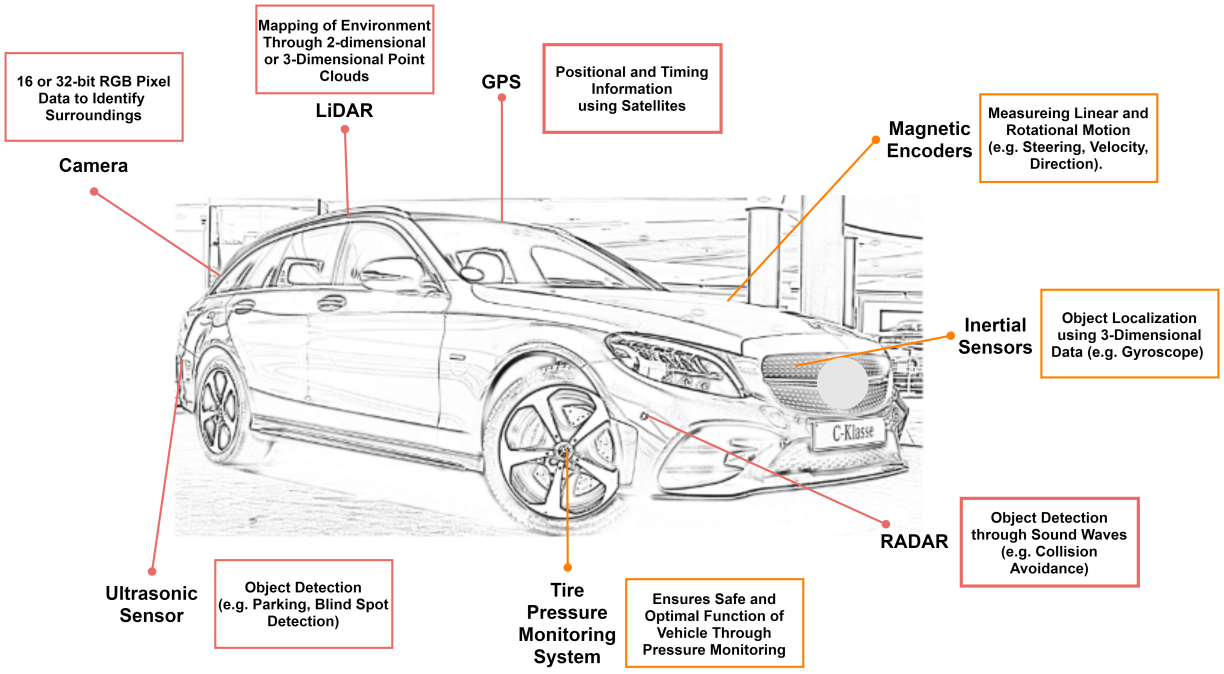


Figure 3.1: Sensor technologies onboard the modern automobile.

1. **Environment Sensors**: Sensors that observe and respond accordingly to changing conditions in a vehicle's immediate surroundings during static or dynamic operations.

2. **Vehicle Dynamics Sensors**: Sensors that provide information about the vehicle's current operating state.

Alternatively, sensors onboard vehicles can be categorized further based on the functions they handle. For instance, Hamid et al. [43] classify sensors based on safety (responsible for decreasing fatal incidents), diagnostics (data that can be used to troubleshoot subsystems), convenience (passenger comfort), and monitoring functions (ambience monitoring). With no exception, autonomous vehicles rely extensively on these sensor systems to solve the key problems in autonomous driving (AD) which are localization, object detection and tracking, and mission planning [44]. Localization in the context of AD requires sensor fusion [45], which is an amalgamation of multiple sensors' data, and a sensor system like 3D light detection and ranging (LiDAR) that creates a 3D point cloud; this can be used to create a 3D map that can be frequently updated based on the vehicle's position [46]. Sensory data gathered from these sensors serve as the basis for other services such as wireless data relay between vehicles or infrastructure, and cloud computing and storage.

3.2 Environment Sensors

3.2.1 LIDAR

LIDAR (previously known as airborne laser scanning sensors) was initially developed in the 1990s for the primary purpose of mapping terrain [47]. LIDAR sensors generate precise spatial mapping of an operating environment's by sensing its surface and shape characteristics by rapidly emitting near infrared (NIR) pulses to the surrounding environment and measuring the time taken for the reflections to be observed at the source. These reflections produce 3D coordinates (in the x, y, and z axes or latitude, longitude, and elevation respectively). A collection of these coordinates is referred to as a "point cloud" and can be used to accurately gauge an environment's surface and features [48]. LIDAR can have ranges up to 60 meters [49] and are of two types: mechanical and solid-state. Mechanical LIDAR sensor systems rely on a gimbal to steer the direction of emitted

and subsequently received pulses and are the systems primarily in use today. Solid state LIDAR do not have moving parts, are compact, provide greater degrees of freedom and higher fields of view (FoV), and will be the future of LIDAR systems used for AD and features such as advanced driver assistance system (ADAS), ACC, and collision avoidance systems [50].

3.2.2 RADAR

Similar to the working principle for LiDAR, radio detection and ranging (RADAR) systems use electromagnetic pulses periodically transmitted to and reflected from objects in the environment. Unlike LIDAR, RADAR sensors can penetrate foggy or cloud ambiences to determine the presence of objects such as pedestrians, buildings, or other vehicles which is otherwise obscured due to factors that limit clear vision. Today, RADAR sensors in vehicles operate in the millimeter wave (mmWave) and use the frequency modulated continuous wave (FMCW) technology as opposed to continuous wave (CW) technology which allows for dynamic adjustment and more efficient usage of the frequency ranges which can vary from 900 MHz to 80 GHz [51]; however, the two frequency bands used by RADARs are 24 GHz and 77 GHz. The types of RADAR sensors in use are either short-range systems, long-range systems, or ultra short-range systems. Short-range RADAR sensors are typically installed on the sides or at the "B Pillars" of a vehicle for collision avoidance and blind spot detection, long-range sensors for ACC at a range of 200 meters and above [52], and ultra short-range sensors are used for parking and lane change assistance at ranges under 40 meters [53][54].

3.2.3 Ultrasonic

Ultrasonic (also known as sonar) sensors measure the distance between the transceiver and surrounding objects using ultrasonic sound waves at frequencies beyond the range of human audibility at approximately 50 kHz. Unlike RADAR or LIDAR, ultrasonic sensors are low cost and have detection ranges upto 2.5 meters [55]. Ultrasonic sensors' accuracies are affected by surface texture and temperature but unlike LIDAR, work well in low lighting conditions and poor visibil-

ity. Ultrasonic systems are used for tasks that will typically be considered non-critical tasks at low speeds, such as when a driver is reversing (object detection to avoid collision) or parking and are typically mounted on a vehicle's bumper [56]. Wagh and colleagues [57] propose a five-tier safety measure system where the first tier is the ultrasonic sensing subsystem that uses CAN due to its robust error handling and fault confinement capabilities.

3.2.4 Camera

Cameras are now ubiquitous and essential components within a vehicle and are used for parking assistance, blind spot detection, traffic sign detection, ACC, etc. depending on where the cameras are installed. Cameras can be installed front-facing, rear-facing, or in locations that provide a 360° surround view when the vehicle is typically operating under parking conditions or under 15 miles per hour [58]. More recently, manufacturers are exploring the option of replacing side wing mirrors with camera monitoring systems (CMS) in consumer automobiles which will allow for greater fields of view and improve the aerodynamic performance of the vehicle thus leading to higher fuel savings. Unlike traditional mirrors, the driver has little control over the adjustment of the CMS apart from brightness adjustment. However, strict guidelines need to be set with regards to lighting and operation in all weather conditions [59], and regulatory approvals need to be sought before implementation. A "context-aware" benchtop setup using CAN that mimics the embedded systems in an automobile is presented by [60] where a camera system is used in tandem with a tire pressure monitoring system (TPMS) to alert the driver of the proximity of the vehicle with an object in addition to wirelessly relaying such data to other vehicles through Zigbee and WLAN.

3.2.5 GPS

Global positioning system (GPS) is a navigational technology that is based on the principle of trilateration and is used extensively for aviation, marine, railroads, roads and highways, and agricultural applications. Trilateration is the principle that a GPS receiver on the earth's surface can be located given that there are atleast 3 satellites that it relays data with. The GPS receiver uses the

time difference between the time it broadcasted its position and the time it received a response from the satellite to estimate the distance (or range) between itself and the satellite. If there are three satellites, the receiver can identify its 3D location. A fourth satellite is used in order to synchronize the received GPS signals to the atomic clocks onboard the satellites; this way, the latitude, longitude, altitude, and time can be obtained by the GPS receiver. Given how ubiquitous GPS-enabled devices are, it is the defacto navigational tool used within automobiles to travel to/from destinations. From an in-vehicle network perspective, GPS data can be derived from CAN traffic and are classified as sensitive data as they can be used to identify vehicle routes and operators' driving patterns [61].

3.3 Vehicle Dynamics Sensors

3.3.1 Magnetic Encoders

Magnetic encoders are used for automobile engine control and associated functions such as the measurement of wheel speed, direction, and angular position. Optical encoders were more commonly used in the past either through a linear (encoding motion along a particular axis) or rotary (encoding rotational motion about an axis) mechanism in high precision devices such as medical equipment or in less demanding hardware such as computer mice. Magnetic encoders are the superior counterparts to the optical encoder due to their low power dissipation, high response, resistance to high working temperatures, and compact form factor [62]. The other significant distinction between magnetic encoders and optical encoders is how they generate a feedback signal: magnetic encoders produce signals based on the change in the resistance of its individual components which depends on magnitude of the ambient magnetic field [62], while optical encoders produce a feedback signal based on the interruption of light from a source such as an light emitting diode (LED). One use case of magnetic encoders is to measure torque [63](force that causes rotation about an axis) which directly influences how powerful an automobile is to undertake tasks such as climbing steep grades.

3.3.2 Inertial Sensors

Inertial sensors typically consists of an accelerometer, which measures the vibrations or rate of change of velocity of a particular object, and gyroscopes which measure the orientation and angular velocity of that particular object in a three dimensional frame. Accelerometers currently function through an electro-optical (EO) principle and are in wide use due to EO's low cost. For instance, they are used in airbags so that quick deployment is made in case of a sudden shock to the vehicle such as in the case of an accident. However, inertial sensors are prone to what is known as drift errors [64] which are deviant and erroneous measurements and thus require information from complementary sensors through sensor fusion in order to relay data of higher accuracy.

3.3.3 Tire Pressure Monitoring System (TPMS)

TPMS is a subsystem within the modern automobile that is made to recognize pre-specified pressure value and notifies the user through the vehicle's instrument cluster (IC) or infotainment system if the measured value is above or below this threshold. There are three types of TPMS present in the modern automobile: the indirect type, where the TPMS gathers information either through software that reads the differences in air pressure by measuring the rotational speed at each wheel, the direct type where pressure sensors are deployed at each tire which relay data to the TPMS through the CAN bus [65], or the hybrid type which combines the benefits of the direct and indirect types.

Please see Table 3.1 for a summary of the sensing systems used in automobiles. The 'usage' attribute refers to whether the corresponding sensor is used in high range (H), mid range (M), or low range (L) vehicles. It's generally observed that vehicle dynamics sensors are common to all modern automobiles. 'Constraint' refers to the operating limitations the corresponding sensor is subject to.

Table 3.1: Sensing systems used in automobiles.

Sensor	Type	Usage	Function	Constraint
LiDAR	Environment	M-H	2D/3D Ambient Mapping	0 - 250 Meters
RADAR		M-H	Object Detection by Sound	30 - 250 Meters
Ultrasonic		M-H	Object Detection by Sound	0 - 2.5 Meters
GPS		L-M-H	Localization	Satellite Coverage
Camera		L-M-H	Ambience Imaging	Fog/Haze & Obstructions
Magnetic Encoders	Vehicle Dynamics	L-M-H	Control Through Detection of Mechanical Motion	Sensitive to Magnetic and Radio Interference
Inertial Sensors		L-M-H	ABS, Air Bag System, Electronic Static Control (ESC)	Drift Errors
TPMS		L-M-H	-	Inaccurate or False Readings

CHAPTER 4

THRESHOLD RANGE ESTIMATION FOR ECU SIGNATURES

To identify underlying patterns in ECU signatures to estimate thresholds, two specific clustering algorithms were studied. The ECU signatures had a high speed (HS) CAN bus operating at a bit rate of 500 kbps. The ECU signatures analyzed were those were for speed (mph), steering (Nm), and tachometer (RPM).

ECU signatures for IDs 280, 1F9, and 300 were captured for 1-minute (idle state), 5-minute (driving), 10-minute (driving), and 15-minute intervals (driving), analyses were broken down on a bin-wise basis to understand the transmission frequencies of the ECUs. There were 10 bins for each of these signatures where each bin was 6 seconds, 30 seconds, 60 seconds, and 90 seconds in duration respectively for the datasets.

Based on histogram and box plots, two of the ECUs transmitted periodically on the network regardless if there were any actions relevant to the ECUs were being applied. The lighting ECU (60D), however, had a different range of transmission frequencies when a function associated to them such as blinkers, hazards, headlights, and high beam were applied. Preliminary findings indicate that the more safety-critical and body control ECUs such as those for steering, tachometer, and speedometer seem to be transmitting consistently and periodically regardless of any changes in their usage and thus will be the focus of this chapter.

Input features to clustering algorithms such as k-means++ ideally need to be normalized as the two features (time along the x-axis and speed, steering, or tachometer along the y-axis) in the feature vector are of different units (such as time or tire pressure) and magnitudes. This is done to prevent the clustering algorithm from biasing its selection towards data points with higher magnitudes. For the purposes of this research work, minimum-maximum scaling was utilized and is given by:

$$X_i = (X_{S.D} \times (X_{max} - X_{min})) + X_{min}$$

X_i represents the feature sample, $X_{S.D}$ is the standard deviation for that feature, X_{min} is the minimum value for that feature, X_{max} is the maximum value for that feature, and $i = 1...N$ where 'N' is the size of the feature vector.

4.1 k-means++ Clustering

Given a set of points 'p' in a 'd' dimensional space, the objective of the k-means problem is to find 'k' number of centroids or centers that minimizes the sum of squared distance (SSD) between each point in a cluster (also known as distortion) and its respective center. The classical k-means algorithm computes these centers randomly from a data space and keeps iterating through the entire space until the algorithm has converged to a reasonable solution. This random selection of centers implies that the results produced for a given dataset might vary if made to run multiple times i.e. the SSD cannot be defined by an interval even when fixed values are specified for the number of data points and number of centroids 'k'. Moreover, though this approach works fairly well with its level of simplicity and speed, it fails to perform accurately as there is no pre-initiation procedure that is followed prior to assigning cluster centroids. For this reason, the algorithm used in this paper is based on the k-means++ rendition of the original k-means algorithm.

Algorithm 1 shows the functional basis of the k-means++ algorithm as proposed by Arthur and Vassilvitskii [66].

Algorithm 1 k-means++

Require: Specifying cluster space 'D', data points x_i, and 'N' number of data points.

1: Randomly choose a centroid C_1
2: **for** x_i where i $= 1, 2,,\ldots, N$ in cluster space 'D' **do**
3: Compute probability distance function from centroid C_1 using equation $\frac{(d_{x_i})^2}{\sum_i^N (d_{x_i})^2}$
4: Choose a new center C_k based on the largest probability
5: **end for**

4.2 Mean Shift Clustering

Mean shift clustering is a non-parametric (i.e. no specified parameters at run-time besides bandwidth of kernel) clustering technique that is based on a fundamentally distinct framework called hierarchical clustering when compared to the k-means++ algorithm. This is a form of clustering that creates a hierarchy from an overarching cluster or "parent" cluster is broken down further at each iteration into sub-clusters that are distinct.

There are distinct kernels such as the Gaussian or flat kernels which are functions that map the defined feature space using weights for each data point in the feature space. This is performed using a kernel density estimation (KDE) using a specified kernel. Algorithm 2 shows the working principle behind the mean shift algorithm. The mean shift algorithm is repeated for all elements in the finite set until it gets closer to the sample means [67] or to the regions of higher density until it reaches convergence.

Algorithm 2 Mean Shift

Require: Finite set 'S' in Euclidean space 'X' where $x \in X$, $s \in S$, K is a defined kernel

1: **for** $s \in S$ **do**
2: $m(x) = \frac{\sum_{s \in S} K \times (s-x) s}{\sum_{s \in S} K \times (s-x)}$
3: Compute mean shift $m(x) - x$ based on sample set
4: Create centroids and shift centroid closer to mean location
5: **end for**

4.3 Pre-processing

The original raw CAN datasets needed to be pre-processed before clustering algorithms could be carried out (please see Figure 4.4). The pre-processing step consisted of 4 sub-steps, namely:

1. Packet timestamping and slotting: Each sample (or packet) within the original raw CAN dataset needed to be identified by a timestamp that started from the 0th second. Since the raw packets captured 32-bit system timestamps which are floating point values that had 10 integer digits and 6 decimal digits, these system timestamps had to be truncated to derive

meaningful time instants. The system timestamp of the first very packet was used as the initiation instant and was subtracted from every subsequent system timestamp (please see Figure 4.1). This produced datasets with time instants $0..n$, where $n = 60, 300,$ and 900. Additionally, time instances were slotted to allow "bin-wise" analysis where each bin was 1/10th of the total duration for the 5, 10, and 15 minute datasets.

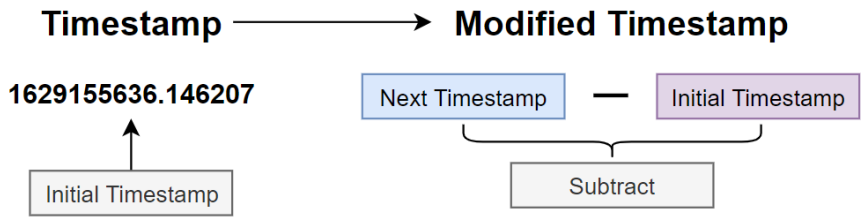


Figure 4.1: Timestamp conversion.

2. Decimal conversion: Raw packets captured for the "ID" and "Data" features are in standard hexadecimal format as specified by the CAN protocol and need to be converted to a decimal format to serve as inputs to supervised learning algorithms which cannot work with alphanumeric strings.

3. Support computation: In order to identify the packets which had high, medium, and low frequencies of transmission on the CAN bus, the support i.e. the ratio of transmitted packets for ID 'A' to the total number of packets for all the transmitting nodes was computed. Support is computed as follows:

$$Support(\%) = \frac{F_i}{T} \times 100$$

where F_i is the data packets corresponding to ECU ID i and T is the total number of data packets. See Figure 4.2: the x-axis represents the ECUs detected on the CAN bus for Nissan Versa 2010 and the y-axis represents the support.

4. Physical value mapping: Standard hexadecimal data for each ECU gives the total length of the transmitting payload (1 - 8 bytes). Further analyses can be done to graph each byte

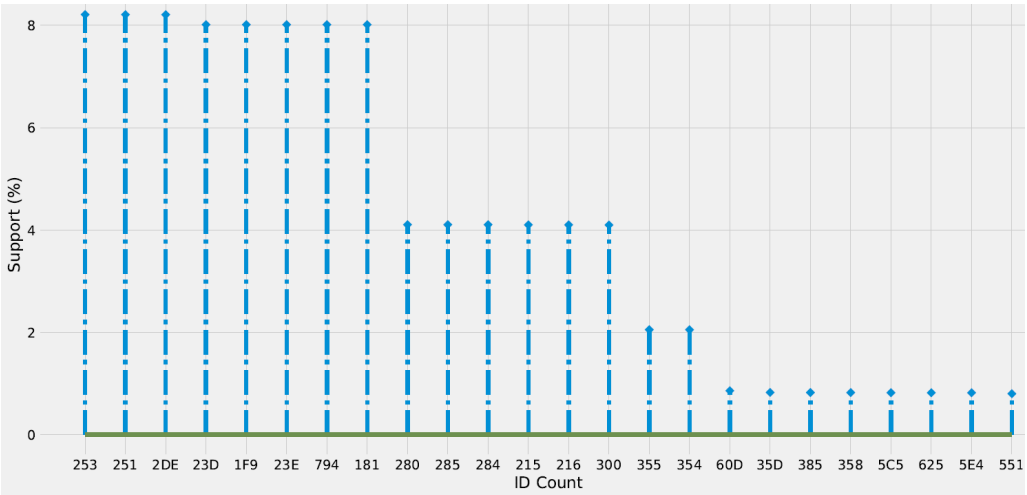


Figure 4.2: Stem plot - Support for Nissan Versa 2010.

in order to observe its behaviour in a time-series. However, physical values i.e. the actual operating values for certain ECUs like engine speed or tachometer can only be obtained after using a OEM-defined equation to convert specific bytes in the hexadecimal data to real-world physical values. The equations below are used in order to obtain the physical values.

$$Torque = \textbf{DEC}(B)$$

$$Tachometer = \frac{(\textbf{DEC}(HB) \times 256) + \textbf{DEC}(LB)}{10}$$

$$Speed = \frac{\textbf{DEC}(HB + LB)}{10}$$

where B, LB, and HB represent a single byte input, low byte, and high byte inputs respectively. Please see Figure 4.3 for the speed, tachometer, and steering torque outputs.

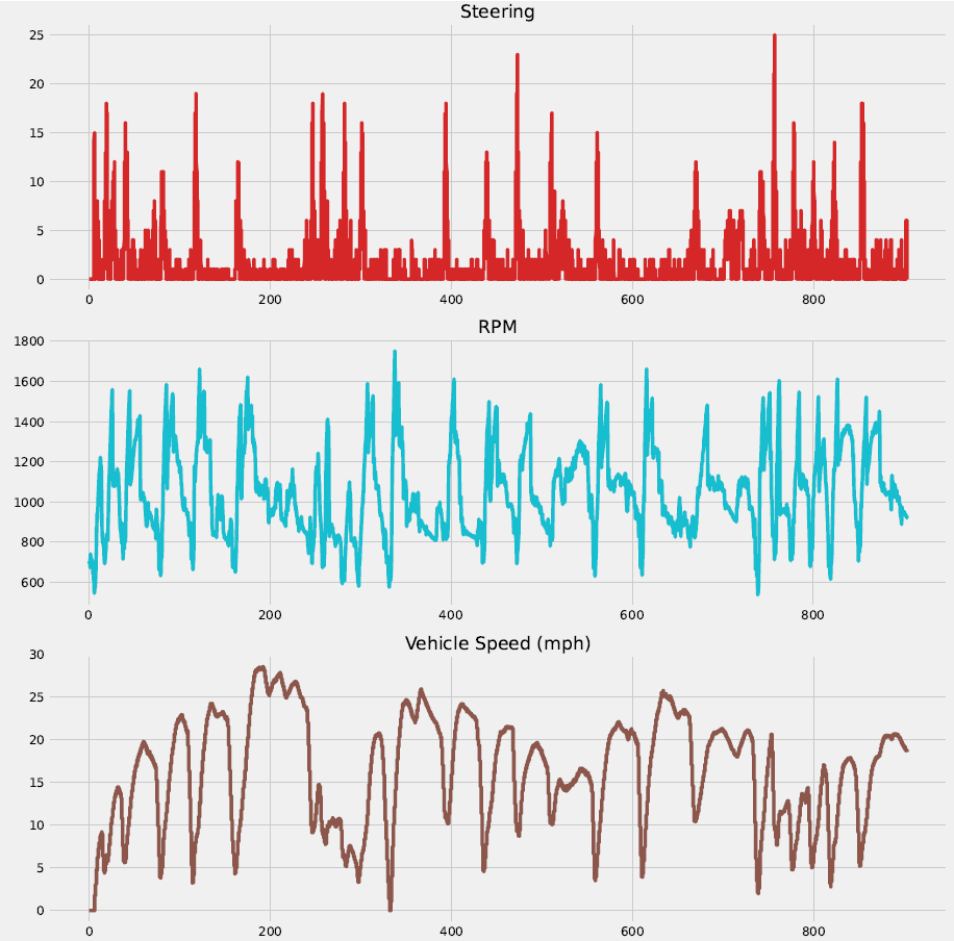


Figure 4.3: Steering, RPM, and vehicle speeds for Nissan Versa 2010.

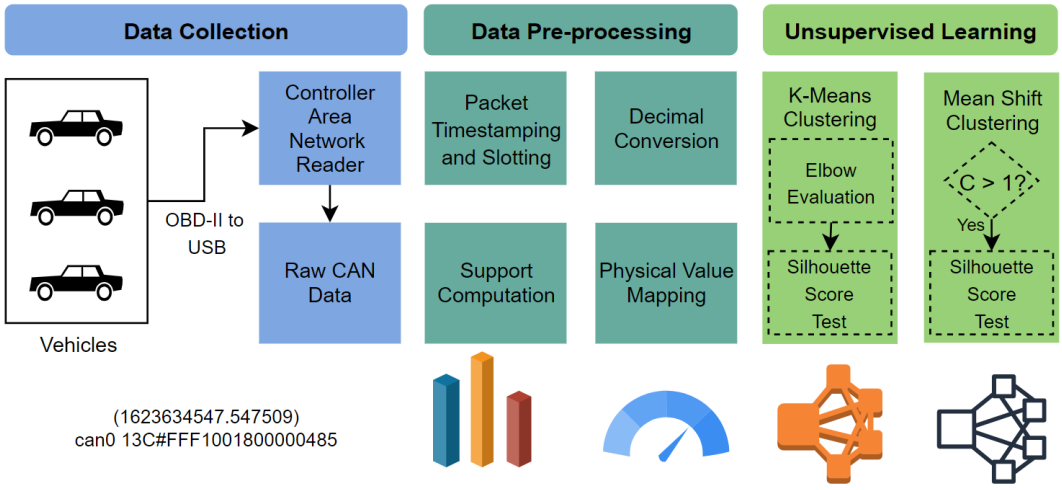


Figure 4.4: Operating framework for estimating thresholds.

4.4 Application of Clustering Methods

Once these pre-processing steps are complete, we obtain a dataset as shown in Table 4.1 which represents the first and last 5 samples of the 10-minute driving dataset for Nissan Altima 2015. The column values for ECUs 300, 1F9, and 280 (steering torque, tachometer, and speed respectively) represent physical values. The columns "ModID" and "ModData" represent the decimal equivalents of the captured hexadecimal data. It can be observed that the column values for ECUs 300 and 280 (steering torque and speed respectively) have negative values. This was a pre-processing step added to suggest that at that particular time instant, that ECU was not active. It can also be inferred that at a particular time instant, say 0, there are a minimum of 500 packets or being transmitted by 49 different nodes. Nodes with lower arbitration IDs tend to transmit more frequently than nodes with greater IDs and tend to be responsible for handling safety critical and body control functions.

Table 4.1: Sample dataset for Nissan Altima 2015.

ModTime	**DataSize**	**ModID**	**ModData**	**ECU300**	**ECU1F9**	**ECU280**
0	8	386	1.84E+19	-1	-1	-1
0	8	505	7.49E+18	-1	545	-1
0	6	533	2.81E+14	-1	-1	-1
0	8	534	4349984	-1	-1	-1
0	8	2	7.21E+18	-1	-1	-1
....						
....						
605	7	352	1.72E+16	-1	-1	-1
605	8	384	2.68E+18	-1	-1	-1
605	8	372	9.1E+18	-1	-1	-1
605	8	386	1.84E+19	-1	-1	-1
605	8	375	1.83E+19	-1	-1	-1

It was deemed appropriate to evaluate the k-means++ clustering algorithm for each signature separately. Additionally, the k-means++ algorithm requires a 2 feature input for each signature and cannot function if a high dimensional input is passed. For instance, k-means++ clustering for the speed ECU signature requires only the timestamp ("ModTime") and physical value ("ECU280")

features. The other requirement for the k-means++ algorithm is to specify the number of clusters. Rather than working with a range of clusters to evaluate the clustering efficiencies of the method, a 2-step process was used to specify 'k' i.e. the number of clusters.

The first step was to evaluate the optimal number of clusters for a given dataset 'D'. This step is well known and is called the elbow evaluation method. The objective of this method is to find the optimal number of clusters at which the sum of squared error (SSE) of specified number of centroids and the data points belonging to those centroids is minimized. This simple method works intuitively to minimize the distance between points belong to a particular centroid or cluster. To keep a consistent basis for choosing the right number of clusters, the error difference (ED) was calculated as the % difference between the error at cluster 'k' and cluster 'k+1' and was set to be under 15%.

$$ED_i(\%) = \frac{e_j - e_{j+1}}{e_j + e_{j+1}} \times 2 \tag{4.1}$$

where $i = 1, 2, 3, ..(N-1)$ and $j = 1, 2, 3, ..N$ where 'N' is the number of clusters being evaluated.

To verify whether the appropriate number of clusters inferred by the elbow evaluation method is optimal, the silhouette coefficient (SC) is defined by:

$$SC = \frac{b_i - a_i}{max(a_i, b_i)} \tag{4.2}$$

where b is the average inter-cluster distance and a is the average intra-cluster distance. Silhouette scores are measured at three significant markers: a score of -1 implies that the clusters are incorrectly assigned, a score of 0 suggests that the inter-cluster distance is not significant and that too many or too clusters are assigned, and a score of 1 shows that the clusters are well separated.

In order to validate the clustering equations for speed and steering, the error % was obtained for each duration and was calculated by the following equation:

$$Error = \frac{Actual_t - Calculated_t}{Actual_t} \times 100\% \tag{4.3}$$

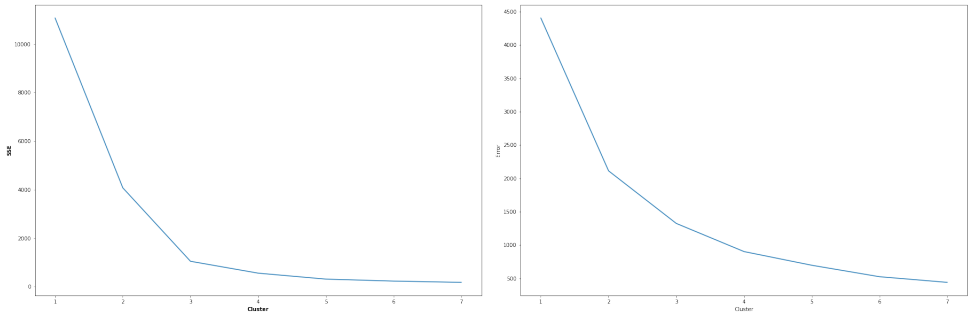
where $Actual_t = \mu_H$ and $Calculated_t$ is the value obtained from Equation 4.6, Equation 4.7, or Equation 4.8 (when we consider the speed signature for make 'A') based on duration 't' where $t = 5$, 10, or 15.

4.4.1 Speed Signature (Make A)

Figure 4.5 shows elbow plots for 5, 10, and 15-minute durations for the speed signature of make 'A' (Nissan). The SSE gets lower progressively as expected from an elbow plot but to identify the optimal number of clusters can be ambiguous in certain cases simply by inspecting the elbow plot. To validate the chosen number of clusters to be the optimal value for the number of clusters when using the elbow method, silhouette coefficients for increasing number of clusters starting from $k = 2$ to $k = 4$ is shown in Figure 4.6. In all these cases, it can be observed that the optimal value for $'k' = 4$; though the differences between each incremental cluster is negligible, the optimal silhouette scores was chosen to be $k = 4$.

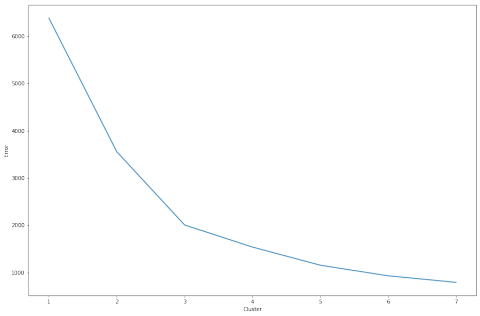
Table 4.2 indexes the error differences for the speed signature across multiple durations. Specifying the number of clusters > 4 for this signature yielded no significant difference. As mentioned in the previous section, a difference of under 15% was kept as a reference and it can be observed that the ED is under this reference percentage when comparing the difference between cluster numbers 3 & 4 and 4 & 5. For instance, the difference is 3.7% when comparing the difference for cluster numbers 3 & 4 and 4 & 5. Overlaying the ED for multiple durations as displayed in Figure 4.7 shows that the speed signature has different ED line plots across the 5, 10, and 15 minute durations but they all give a difference of under 15% at indexes or rows 3 and 4 for each duration.

Following the optimal number of clusters for the speed signature (ECU280) to be $k = 4$, clustering plots for each of the durations are shown in Figure 4.8. It was observed that, as opposed to clustering when the data is not normalized and the clustering process simply segments the three clusters into roughly even intervals when referring to the 'x' axis (duration), the clustering process



(a) Optimal 'k' for speed signature for 5 minute duration.

(b) Optimal 'k' for speed signature for 10 minute duration.



(c) Optimal 'k' for speed signature for 15 minute duration.

Figure 4.5: Elbow plots for make A's speed signature.

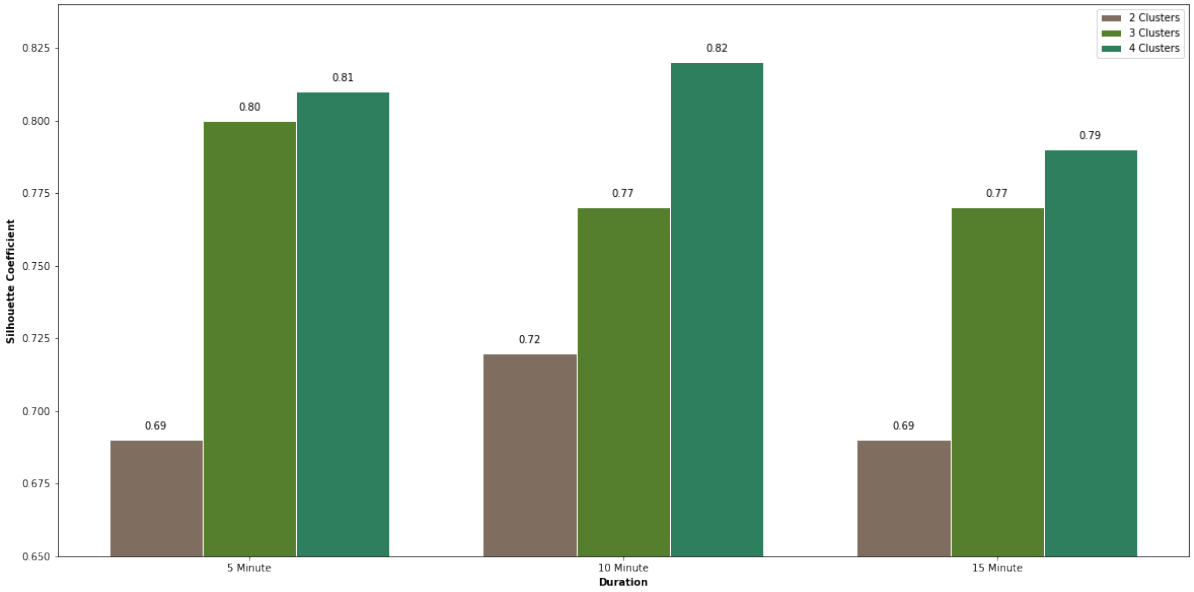


Figure 4.6: Silhouette scores for 2, 3, and 4 clusters for different durations (speed).

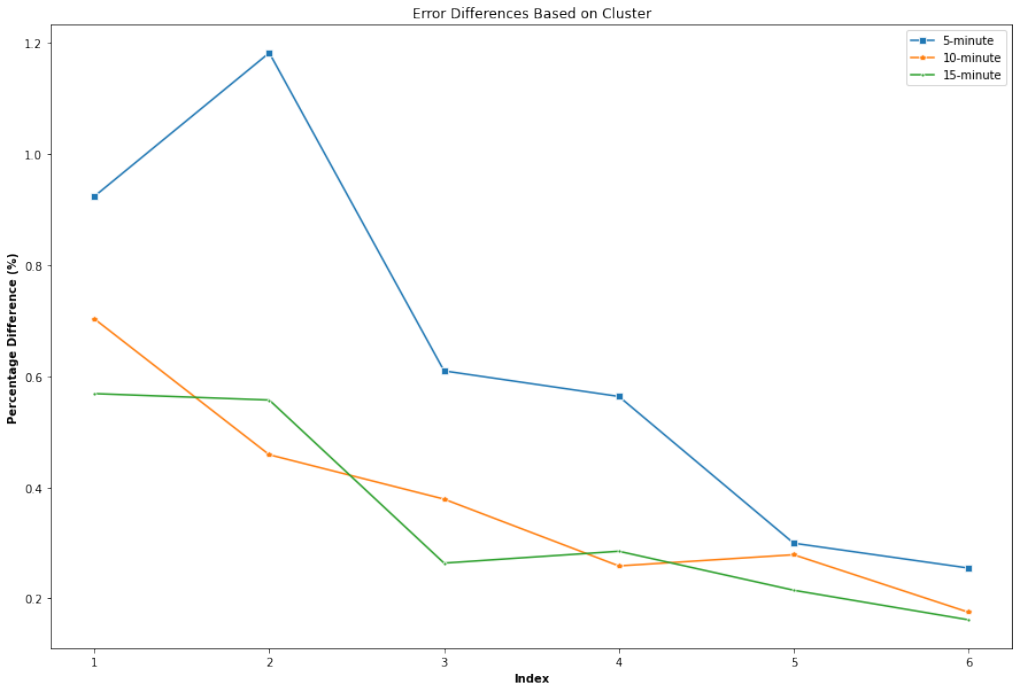
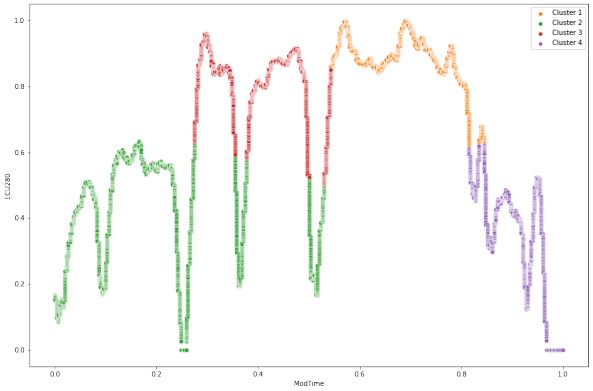


Figure 4.7: Error differences (%) for multiple durations for speed signature.

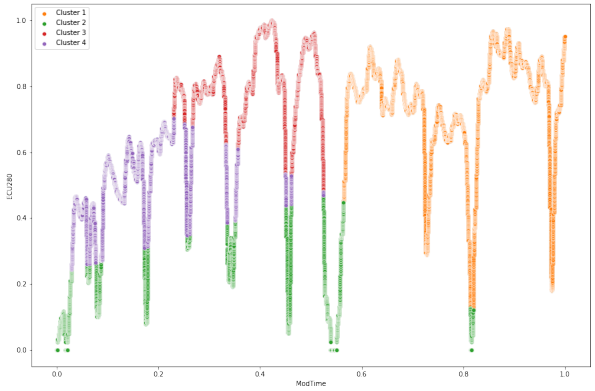
Table 4.2: Inter-cluster percentage error differences for increasing number of clusters (speed).

Cluster Numbers	**Error Difference (%)**	**Duration**
1 & 2	92.3	5 Minutes
2 & 3	118.2	
3 & 4	60.0	
4 & 5	56.3	
5 & 6	29.9	
6 & 7	25.4	
1 & 2	70.3	10 Minutes
2 & 3	45.8	
3 & 4	37.8	
4 & 5	25.8	
5 & 6	27.8	
6 & 7	17.5	
1 & 2	56.8	15 Minutes
2 & 3	55.7	
3 & 4	26.3	
4 & 5	28.4	
5 & 6	21.4	
6 & 7	16.1	

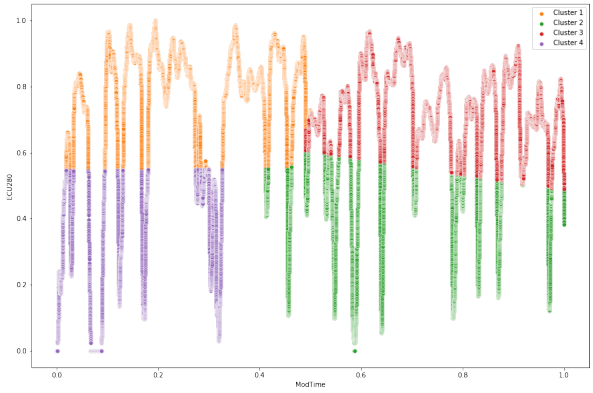
post-normalization yields cluster centers that are statistically different from one another for each duration (please see Table 4.3 and Table 4.4).



(a) k-means++ clustering for speed signature for 5 minute duration.



(b) k-means++ clustering for speed signature for 10 minute duration.



(c) k-means++ clustering for speed signature for 15 minute duration.

Figure 4.8: k-means++ clustering results for 5, 10, and 15 minute durations (speed).

The clusters and their relative statistics identified by k-means++ and mean shift algorithms are shown in Table 4.3 and Table 4.4 respectively. The numbers of clusters identified by each of the algorithms differ for the 5 and 15 minute durations and hence, the requirement of a color scheme to identify similar clusters was defined. The color schemes in these tables indicate which clusters between the two clustering methods match well in terms of their average values. For instance, for the 5 minute duration, we see that clusters 1 and 2 of Table 4.3 when combined with cluster 1 of Table 4.4, equate closely in terms of their averages i.e. within one S.D. Similarly, cluster 1 and cluster 1 of the k-means++ and mean shift algorithms respectively for the 10 minute duration match

well despite varying in the number of data points. It can be consistently observed that the average values for each cluster and for each duration (left most column) are within one standard deviation of the mean standard deviation (mean S.D.) for each of the durations; this can be considered to be within reasonable bounds for the formulation of threshold equations for this signature.

Table 4.3: k-means++ cluster analysis for multiple durations for speed signature.

Duration Average	Cluster No.	Minimum	Maximum	Average	No. of Data Points	Duration	Mean S.D.
15.7	1	12.9	24.6	21.3	3218	5 Minutes (k = 4)	6.6
	2	0	16.2	10.5	4978		
	3	15.7	25.6	22.7	4236		
	4	0	15.9	8.35	2625		
14.5	1	3.2	25.1	19.4	12998	10 Minutes (k = 4)	
	2	0	12.0	3.9	3800		
	3	12.4	25.8	21.4	7389		
	4	6.3	18.1	13.4	6039		
14.6	1	3.2	26.6	22.0	12998	15 Minutes (k = 4)	
	2	0	16.1	9.35	5228		
	3	13.0	25.7	20.17	18297		
	4	0	14.6	7.2	5499		

Table 4.4: Mean Shift cluster analysis for multiple durations for speed signature.

Duration Average	Cluster No.	Minimum	Maximum	Average	No. of Data Points	Duration	Mean S.D.
15.8	1	0.0	25.6	16.6	7597	5 Minutes (C = 2)	6.6
	2	0.0	24.6	15.1	7460		
16.9	1	0.0	25.1	18.6	9454	10 Minutes (C = 3)	
	2	0	25.8	15.1	13376		
	3	0.0	24.8	17.2	7396		
18.2	1	0.0	25.7	18.3	9445	15 Minutes (C = 4)	
	2	2.6	24.6	17.8	13882		
	3	1.2	26.2	20.4	7896		
	4	0.0	26.6	16.6	14433		

The threshold equation for the speed signature can be formulated as follows:

$$Tspeed_t = \mu_L \pm (m \times S.D_t) \tag{4.4}$$

where

$$m = \frac{\mu_H - \mu_L}{\mu_{S.D}} \tag{4.5}$$

and $\mu_H, \mu_L, \mu_{S.D.}$ represent the highest mean for a cluster, the lowest mean for a cluster, and the mean of the standard deviations for the different durations respectively. $S.D_t$ represents the standard deviations for durations $'t' = 5, 10, 15$.

Using Equation 4.4, mathematical representations for these thresholds of the speed signature are shown below using Equation 4.6, Equation 4.7, and Equation 4.8 for the 5, 10, and 15 minute durations respectivel for vehicle of make 'A' and year 2015. These equations estimate the thresholds with an error of under 3% when the % difference between $S.D_H$ and $S.D_L$ is below 11%, where $S.D_H$ is the highest standard duration among the 3 durations and $S.D_L$ is the lowest standard deviation.

$$Tspeed_5 = \mu_L \pm (2.17 \times S.D_5) \tag{4.6}$$

$$Tspeed_{10} = \mu_L \pm (2.65 \times S.D_{10}) \tag{4.7}$$

$$Tspeed_{15} = \mu_L \pm (2.24 \times S.D_{15}) \tag{4.8}$$

To validate the equations shown above, two different vehicles from years 2010 and 2013 but the same make were used for the same durations as shown in Table 4.5. The "S.D. Difference" is defined as

$$S.D.Difference = \frac{S.D_H - S.D_L}{S.D_H + S.D_L} \times 2$$

where $S.D_H$ is the highest standard deviation among the 5, 10, and 15 minute duration and $S.D_L$ is the lowest standard deviation among the three durations. The percentage error for each duration was obtained by calculating the error from the result of the threshold equation for that duration and the μ_H using both the clustering methods as shown in Equation 4.3. The lowest errors obtained for each duration are reported. As expected, we can see errors of under 6% for the 2015 vehicle when using the threshold equations for speed whereas higher errors for the older 2010 vehicle. The lowest error reported by the equations for the 2010 and 2013 vehicles is for the 10-minute duration for the 2013 vehicle which is at 25%. The errors for the 2010 and 2013

vehicles for the 5 and 10 minute durations are similar; this is because the highest means obtained by clustering are similar in magnitude for both these vehicles and their respective durations. Higher S.D. differences are not effective indicators of lower validation errors and can be appropriately disregarded.

Table 4.5: Validation percentage errors using speed threshold equations for 2 different vehicles of the same make.

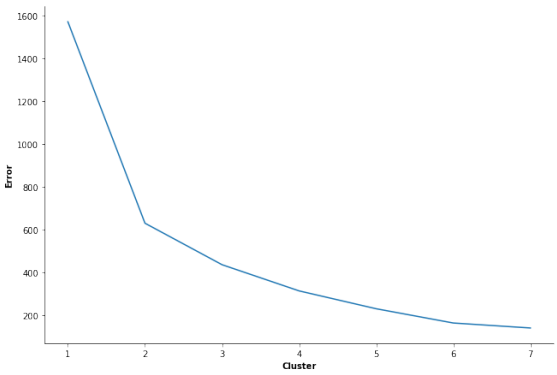
Vehicle Year	$\mathbf{T}_{speed_5}$	$\mathbf{T}_{speed_{10}}$	$\mathbf{T}_{speed_{15}}$	S.D. Difference
2015	4.8%	2.0%	2.1%	11.0%
2010	35.9%	35.3%	48.9%	20.6%
2013	29.9%	25.0%	39.4%	37.5%

4.4.2 Steering Signature (Make A)

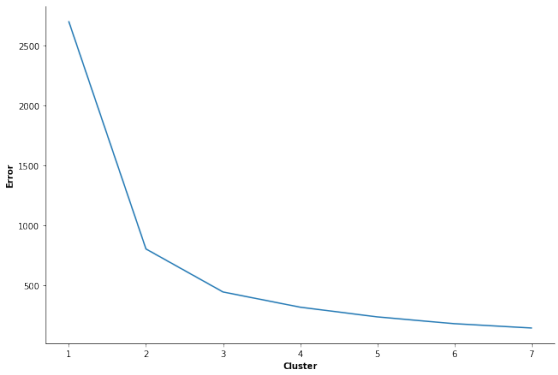
Evaluating the ideal number of clusters, Figure 4.9 shows elbow plots for 5, 10, and 15-minute durations for the speed signature. Similar to the speed signature, the elbow is not particularly obvious from the graphs but '$k' = 4$ is chosen to be the optimal number of clusters.

Table 4.6 indexes the distance errors for the steering signature across multiple durations. Figure 4.10 shows the silhouette coefficients for cluster $k = 2$ to $k = 4$. Plotting the DE for multiple durations as shown in Figure 4.11 shows that the steering signature percentage differences are similar to the speed signature in that the % difference falls below the benchmark of 15% for $k = 4$. Performing the silhouette coefficient test, it is evident that following the optimal number of clusters for the steering signature (ECU300) is $k = 4$. Clustering plots for each of the durations based on $k = 4$ are shown in Figure 4.12.

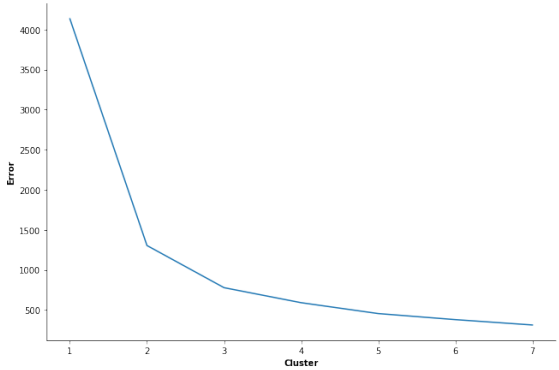
The validation percentage errors for the steering signature are shown in Table 4.7. Like the validation for the speed equations, the percentage error for each duration was obtained by calculating the error from the result of the threshold equation for that duration and the μ_H using clustering as shown in Equation 4.3. The lowest error obtained for each duration is reported. Similar to the speed threshold, the S.D. difference (measure of spread in the data) is not an effective indicator of validation error; though the 2010 vehicle has a lower S.D. difference, the average validation error



(a) Optimal ‘k’ for steering signature for 5 minute duration.



(b) Optimal ‘k’ for steering signature for 10 minute duration.



(c) Optimal ‘k’ for steering signature for 15 minute duration.

Figure 4.9: Elbow plots for make A’s steering signature.

across the 3 durations is slightly higher than that of the 2013 vehicle. The 5 minute duration for the 2013 vehicle reports the relative lowest validation error of 45.9%.

The clusters and their relative statistics identified by k-means++ and mean shift algorithms are shown in Table 4.8 and Table 4.9 respectively. Like for the speed signature, a color scheme was used to match the varying number of clusters identified by the two algorithms. The duration average values for each duration are within one mean standard deviation for each of the multiple durations except for the 5 minute duration of the mean shift cluster analysis.

Based on Equation 4.4, the threshold equations (with an error under 1.2%) for the steering signature for vehicle 1 of make ‘A’ are as follows:

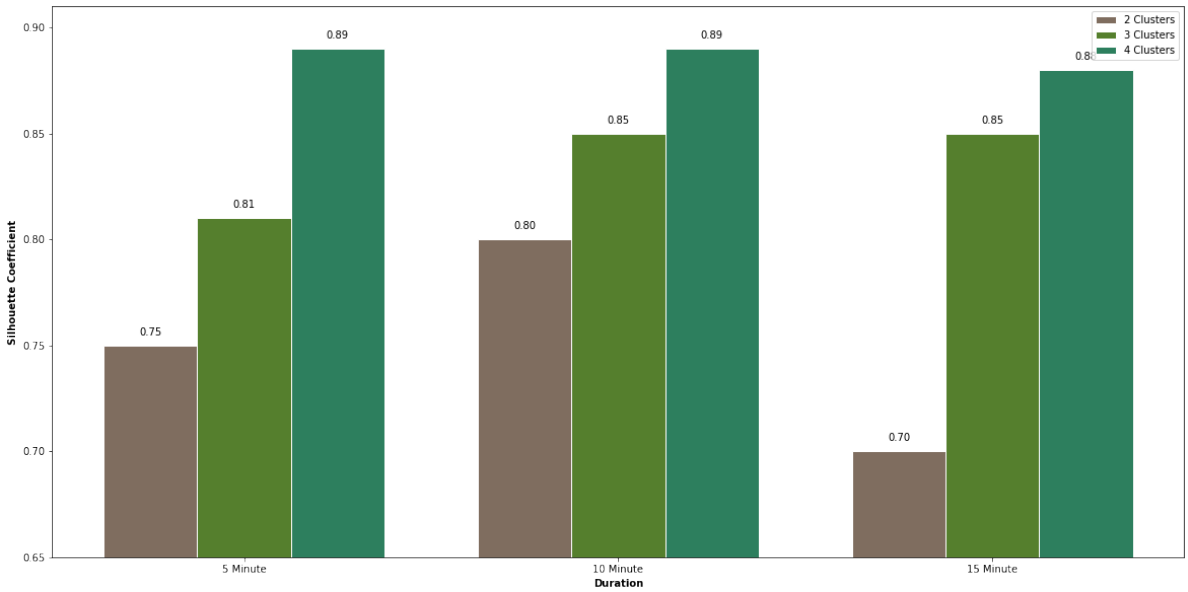


Figure 4.10: Silhouette scores for 2, 3, and 4 clusters for different durations (steering).

Table 4.6: Inter-cluster percentage error differences for increasing number of clusters (steering).

Cluster Numbers	**Error Difference (%)**	**Duration**
1 & 2	85.4	5 Minutes
2 & 3	36.3	
3 & 4	32.5	
4 & 5	30.5	
5 & 6	33.6	
6 & 7	15.2	
1 & 2	108.0	10 Minutes
2 & 3	57.2	
3 & 4	32.9	
4 & 5	28.8	
5 & 6	26.8	
6 & 7	21.6	
1 & 2	104.1	15 Minutes
2 & 3	50.5	
3 & 4	27.4	
4 & 5	26.0	
5 & 6	18.1	
6 & 7	19.3	

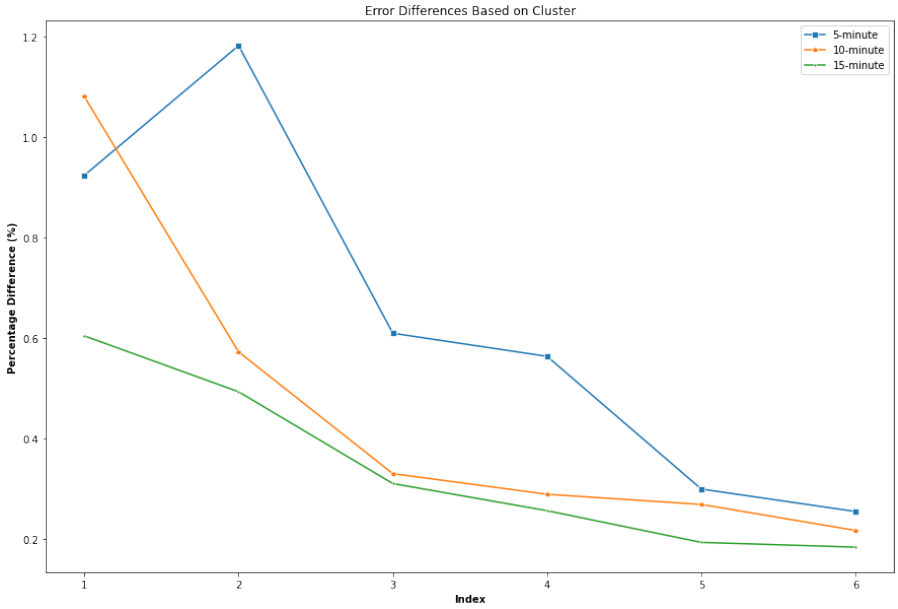


Figure 4.11: Error differences (%) for multiple durations for steering signature.

Table 4.7: Validation percentage errors using steering threshold equations for 2 different vehicles of the same make.

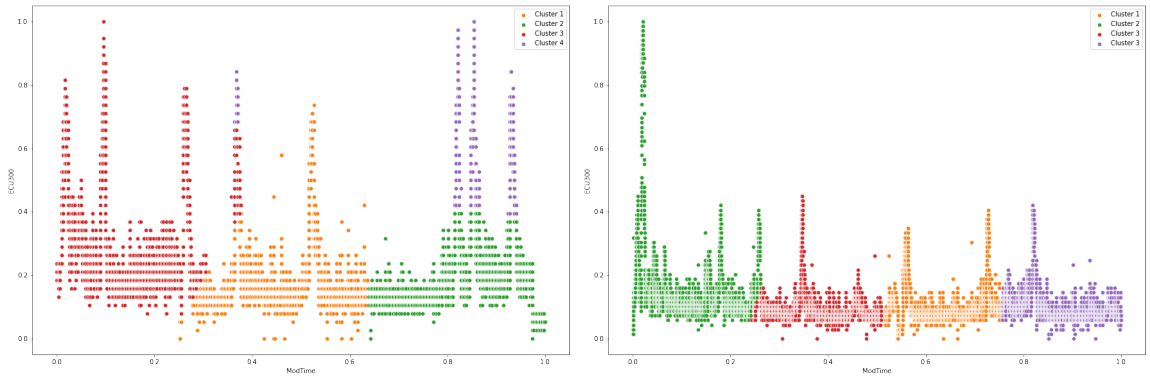
Vehicle Year	$T_{steering_5}$	$T_{steering_{10}}$	$T_{steering_{15}}$	S.D. Difference
2015	3.5%	0.07%	5.3%	7.5%
2010	54.8%	89.5%	95.4%	17.7%
2013	45.9%	91.6%	96%	44.7%

Table 4.8: k-means++ cluster analysis for multiple durations for steering signature.

Duration Average	Cluster No.	Minimum	Maximum	Average	No. of Data Points	Duration	Mean S.D.
11.8	1	0	28	6.3	5179	5 Minutes (k = 4)	5.3
	2	0	15	6.1	4957		
	3	3	38	10.5	4364		
	4	15	38	24.6	563		
7.4	1	0	28	6.6	7460	10 Minutes (k = 4)	
	2	1	69	10.5	7778		
	3	0	31	6.2	7647		
	4	0	29	6.69	7358		
7.9	1	0	40	7.3	11176	15 Minutes (k = 4)	
	2	0	57	8.0	10977		
	3	3	35	9.4	11562		
	4	0	40	7.1	11458		

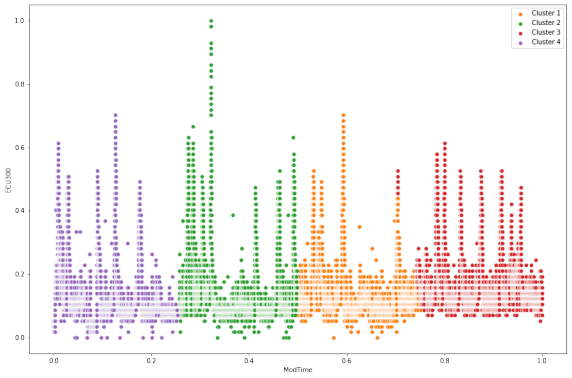
$$Tsteering_5 = \mu_L \pm (3.49 \times S.D_5) \tag{4.9}$$

$$Tsteering_{10} = \mu_L \pm (0.86 \times S.D_{10}) \tag{4.10}$$



(a) k-means++ clustering for steering signature for 5 minute duration.

(b) k-means++ clustering for steering signature for 10 minute duration.



(c) k-means++ clustering for steering signature for 15 minute duration.

Figure 4.12: k-means++ clustering results for 5, 10, and 15 minute durations (steering torque).

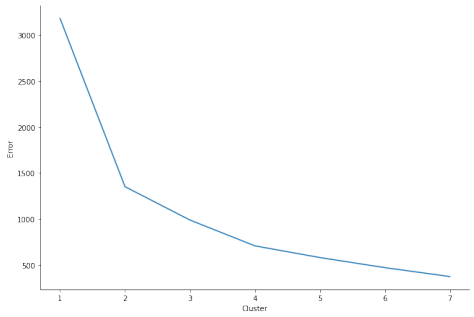
Table 4.9: Mean shift cluster analysis for multiple durations for steering signature.

Duration Average	Cluster No.	Minimum	Maximum	Average	No. of Data Points	Duration	Mean S.D.
24.1	1	0	34	6.5	5859	5 Minutes (C = 3)	5.3
	2	0	36	13.4	8546		
	3	37	75	52.4	666		
11.8	1	0	53	10.6	15846	10 Minutes (C = 2)	
	2	0	103	13.15	14234		
7.9	1	3	35	9.3	11838	15 Minutes (C = 4)	
	2	0	40	6.9	7908		
	3	0	36	8.0	8072		
	4	0	57	7.5	17355		

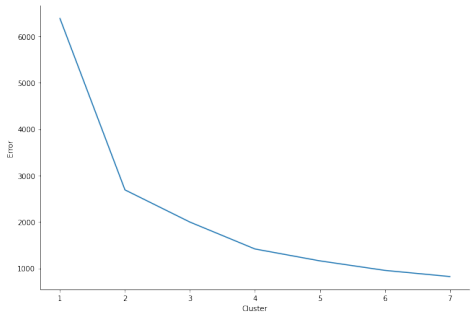
$$Tsteering_{15} = \mu_L \pm (0.39 \times S.D_{15}) \tag{4.11}$$

4.4.3 Tachometer Signature (Make A)

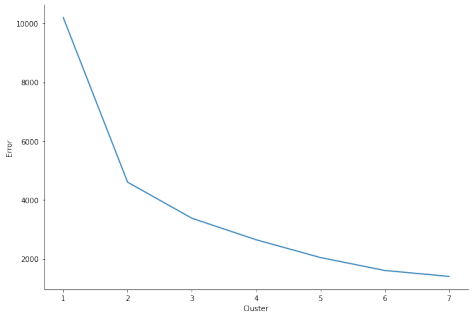
Figure 4.13 shows elbow plots for 5, 10, and 15-minute durations for the tachometer signature. The optimal 'k' for this signature could not be coherently determined by the elbow evaluation plots in Figure 4.13 and required silhouette coefficients for clusters $k = 2$ to $k = 4$ as shown in Figure 4.14. The highest silhouette coefficient scores are for when $k = 4$; thus, we can conclude from these graphs that the optimal $k = 4$ for the 5, 10, and 15 minute durations.



(a) Optimal 'k' for steering signature for 5 minute duration.



(b) Optimal 'k' for steering signature for 10 minute duration.



(c) Optimal 'k' for steering signature for 15 minute duration.

Figure 4.13: Elbow plots for make A's tachometer signature.

Table 4.10 indexes the distance errors for the steering signature across multiple durations. It is observed, particularly in the case for the 10 minute duration, that though the ED % between each cluster may be large, it may not indicate that the optimal cluster is the greater cluster number of the two clusters being differenced. Plotting the DE for multiple durations as shown in unlike the signatures for steering and speed, Figure 4.16 shows that the tachometer signature percentage differences vary between each cluster is different and there is little to no agreement between mul-

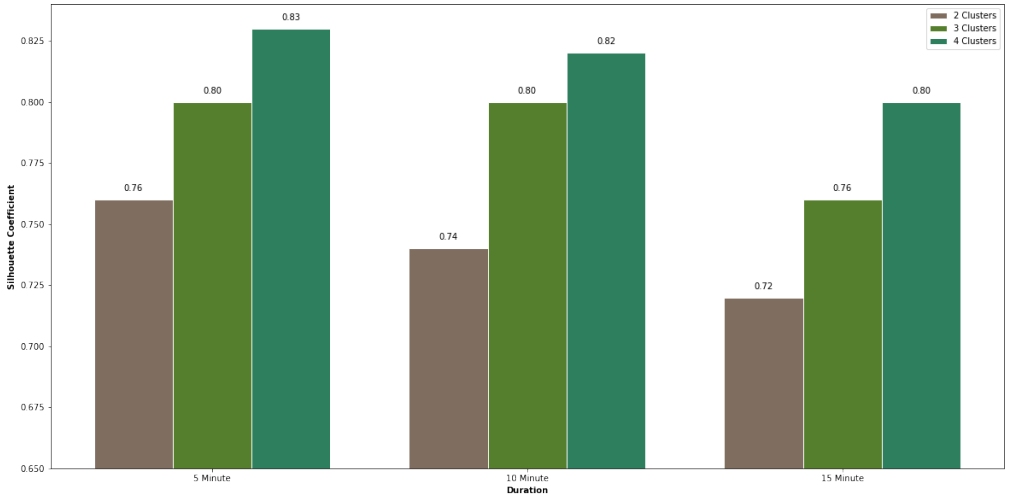
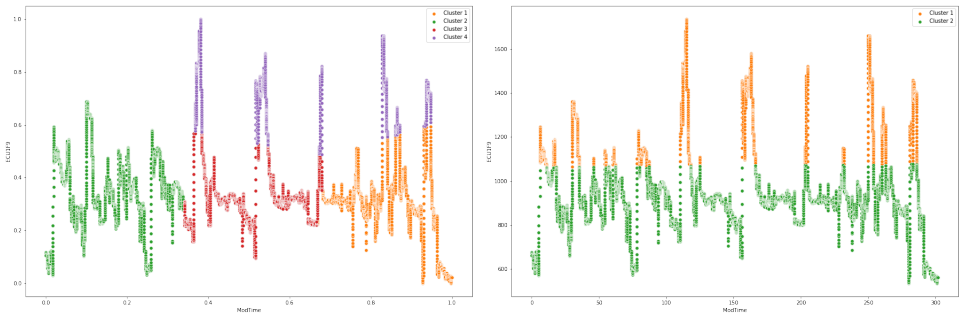


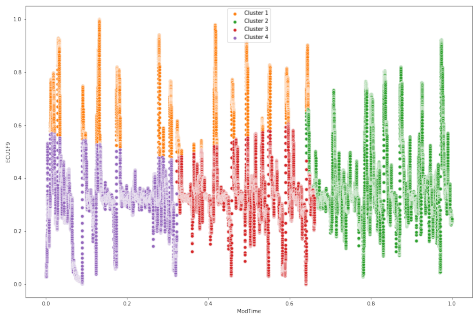
Figure 4.14: Silhouette scores for 2, 3, and 4 clusters for different durations (tachometer).

tiple durations. Clustering plots for each of the durations are shown based on $k = 4$ are shown in Figure 4.15.



(a) k-means++ clustering for tachometer signature for 5 minute duration.

(b) k-means++ clustering for tachometer signature for 10 minute duration.



(c) k-means++ clustering for tachometer signature for 15 minute duration.

Figure 4.15: k-means++ clustering results for 5, 10, and 15 minute durations (tachometer).

Table 4.10: Inter-cluster percentage error differences for increasing number of clusters (tachometer).

Cluster Numbers	Error Difference (%)	Duration
1 & 2	80.7	5 Minutes
2 & 3	31.0	
3 & 4	33.1	
4 & 5	19.7	
5 & 6	20.9	
6 & 7	22.9	
1 & 2	81.3	10 Minutes
2 & 3	29.5	
3 & 4	33.8	
4 & 5	19.8	
5 & 6	19.3	
6 & 7	14.8	
1 & 2	104.1	15 Minutes
2 & 3	50.5	
3 & 4	27.4	
4 & 5	26.0	
5 & 6	18.1	
6 & 7	19.3	

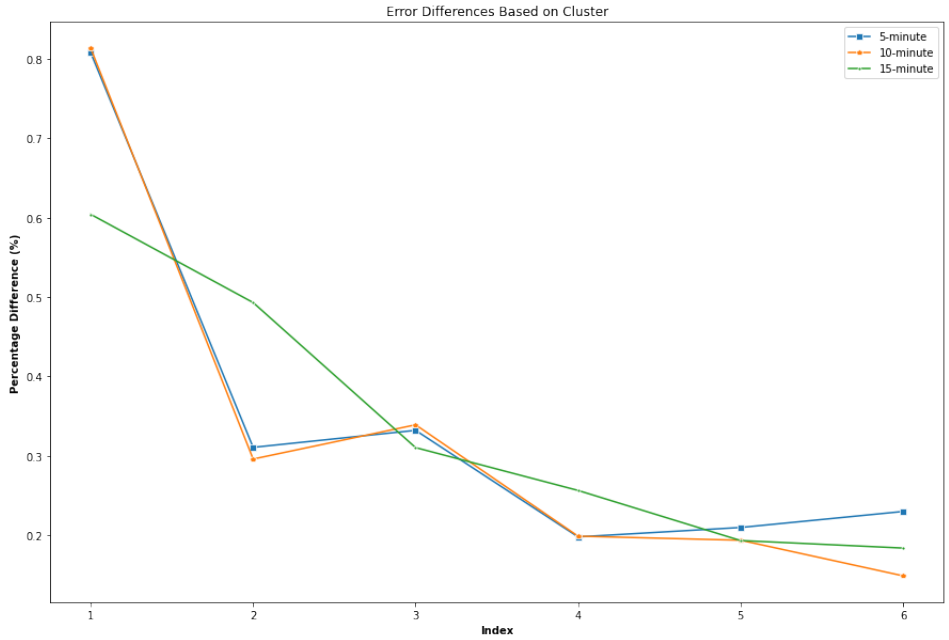


Figure 4.16: Error differences (%) for multiple durations for tachometer signature.

Table 4.11: k-means++ cluster analysis for multiple durations for tachometer signature.

Duration Average	Cluster No.	Minimum	Maximum	Average	No. of Data Points	Duration	Mean S.D.
1028.4	1	572.5	1360	936.2	9988	5 Minutes	188.7
	2	535	1247.5	869	8607		
	3	650	1217.5	903	8520		
	4	1120	1735	1405.4	2309		
1006.3	1	510	1165	871.4	9988	10 Minutes	
	2	527.5	1215	877.7	17750		
	3	595	1670	973.5	18867		
	4	1040	1692.5	1302.8	5490		
1093.9	1	545	1127.5	874.8	29114	15 Minutes	
	2	537.5	1612.5	1039.1	26324		
	3	790	1662.5	1159.8	20210		
	4	1022.5	1752.5	1302	12569		

4.4.4 Speed Signature (Make B)

To compare the threshold equation obtained for make 'A' with another vehicle manufacturer, a 10-minute dataset was obtained for a vehicle of make 'B' (Honda) and the signature analyzed was the speed signature. Figure 4.17 shows the elbow plot and the corresponding silhouette scores for 2, 3, and 4 clusters. The optimal k for this signature using the elbow plot and the silhouette scores is $k = 4$. The corresponding clustered plot for the speed signature is shown in Figure 4.18.

The corresponding threshold equation for this signature is a special case of Equation 4.4 where $\mu_{S.D} = \mu_{S.D_t}$ where $t = 10$. So, we modify the equation to

$$Tspeed_{10} = \mu_L \pm (m \times \mu_c) \tag{4.12}$$

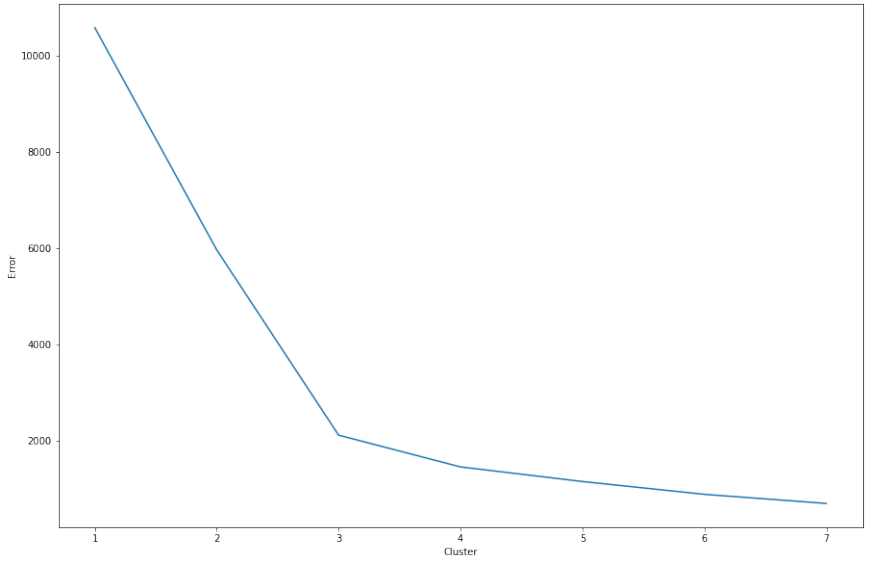
where μ_c is the average of the means of the clusters identified by k-means++ within the 10 minute speed dataset. The threshold equation thus obtained for the speed signature of make 'B' is

$$Tspeed_{10} = \mu_L \pm (2.59 \times \mu_c) \tag{4.13}$$

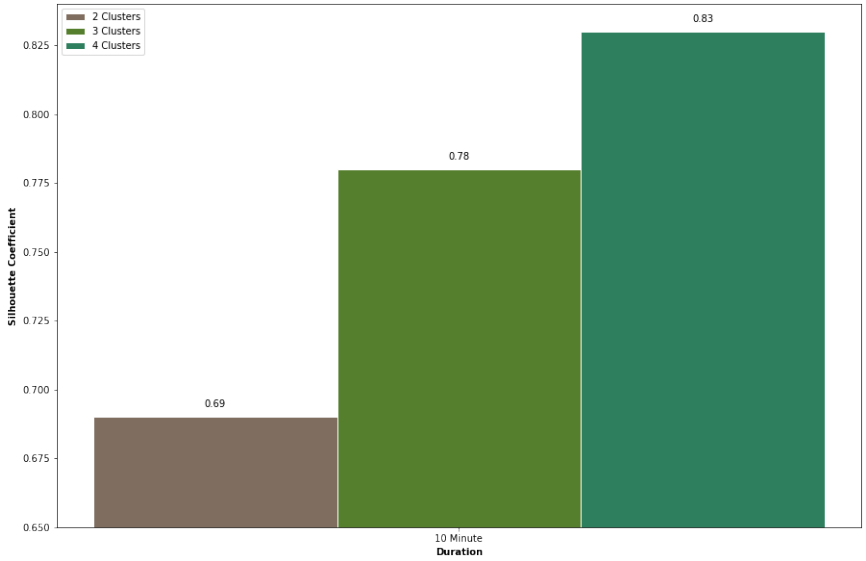
Table 4.12: Validation percentage errors using speed threshold equations for 2 sub-datasets of the same make.

Vehicle Year	k-means++	S.D. Difference
2016_d	13.5%	12.7%
2016_{sd1}	13.6%	13.4%
2016_{sd2}	79%	10.8%

To verify this equation, the 10-minute dataset (2016_d) was split into two 5-minute sub-datasets 2016_{sd1} and 2016_{sd2}. The validation errors using the speed threshold equation for make 'B' is shown in Table 4.12. Unlike the case for the previous signatures of speed for make 'A', there is an increase in % error as shown by the k-means++ algorithm though the S.D. difference is low. The high error of 26.2% for the main dataset suggests that the dependent variables in Equation 4.13 need to be reevaluated to accommodate for lower errors atleast on the original dataset.



(a) k-means++ clustering for speedometer signature for 10 minute duration.



(b) Silhouette scores for 2, 3, and 4 clusters for 10 minute duration

Figure 4.17: k-means++ elbow plot and silhouette scores for Honda (speed signature).

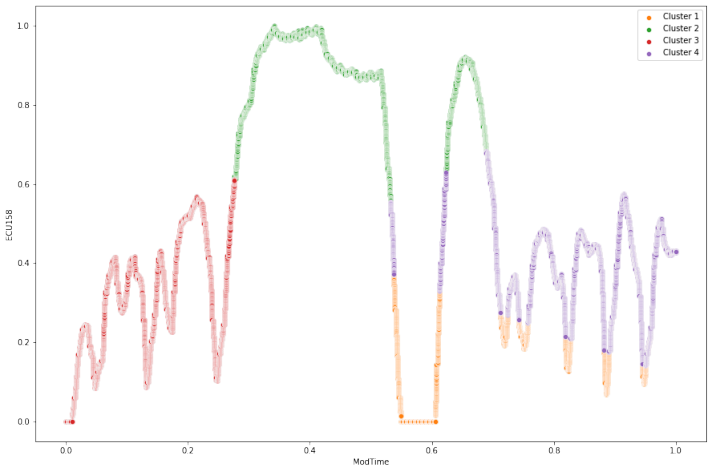


Figure 4.18: k-means++ clustering results and silhouette coefficients for speed signature of make ‘B’.

CHAPTER 5

ECU CLASSIFICATION USING MACHINE LEARNING

5.1 Introduction

Supervised and unsupervised machine learning algorithms have been used for binary and multi-class classification problems in literature to identify different ECUs within a vehicle. For instance, Young and colleagues [68] use a hybrid supervised and unsupervised machine learning approach to identify ECUs based on vehicular function (driving or idle states). Three CAN datasets were collected from simulation, Oak Ridge National Laboratory traffic logs, and a John Deere tractor. The vehicle setup was on a dynamometer to simulate driving conditions. This data was preprocessed to label the raw CAN data into headers such as ID, timestamp, data, and vehicular function. Agglomerative clustering is used in order to make this classification by grouping together ECU IDs which are similar in function. However, no supervised learning algorithm was used and the ECU classification broadly classifies similar ECUs into driving and idle states without individually identifying the ECUs responsible for specific functions such as lighting or seatbelt. Similarly, Avatefipour et al. [69] posit that each electronic control unit (ECU) and the physical medium (CAN bus) that it uses to transmit the data produces unique time and spectral "artifacts" that can be classified to correctly identify the source ECU based on power characteristics of the ECU from different channels on the CAN bus. Features in the time and frequency domain were used as inputs to the machine learning classifier used i.e. a multi-layer perceptron (ML). Accuracies of 95% and upwards were observed in identifying the channel and the respective ECUs within that channel. However, only one classifier was used to assess the performance which gives little to no information about how other algorithms not based on a neural network architecture will perform. Additionally, the number of samples per ECU isn't mentioned i.e. whether the ECUs are evenly sampled or not; this is an important observation in such datasets as CAN samples will be unbalanced for each ECU.

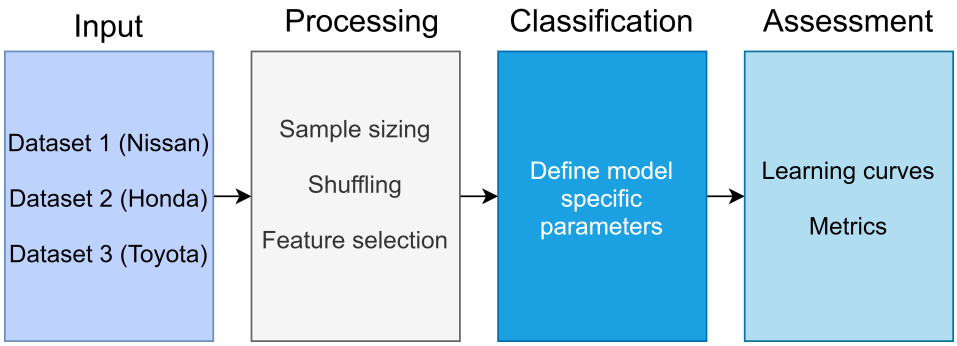


Figure 5.1: Machine-learning workflow for ECU identification.

In this section, three machine learning classifiers are presented in order to identify ECUs from homogeneous (single make) and heterogeneous (multi-make) datasets (please see Figure 5.1). The second case is unique in that the same classifiers trained on identifying ECUs from Nissan will be tested to identify ECUs from the Honda and Toyota datasets. The datasets are first shuffled sample-wise, and then sampled based on the number of frames per ECU to test for both unbalanced and balanced classes. The features selected are shown in Table 5.1. While the dimensionality problem (where accuracy progressively decreases with the increase in features) is not well known for datasets used for ECU classification, minimizing the number of input features to the machine learning classifiers was one of the objectives in order to assess how the performance of the models are when limited but significant features were given for the classification task. Model specific parameters such as number of neighbors, distance metric (Euclidean/Manhattan), maximum depth of each tree, and variance smoothing are chosen to maximize accuracy. In order to assess the models in terms of their learning (overfitting or underfitting), learning curves based on a 5-fold cross validation is performed. Depending on whether the datasets used are balanced or unbalanced, the metrics used were accuracy, balanced accuracy, and weighted F1-score.

Table 5.1: Features and characteristics.

Feature	**Size**	**Description**
Time	10-bit floating point	Timestamp of transmitted frame
Size	3-bit floating point	Size of transmitted frame in bytes
Data	32-bit floating point	Data payload of transmitted frame

5.2 Machine Learning Models and Evaluation

5.2.1 Decision Tree

The Decision Tree (DT) model is a machine learning algorithm that can be used for classification and regression tasks and is based on dividing a dataset and progressively narrowing the input sample to a specific output based on a decision function. The general architecture of of a DT model is comprised of a root node (where the input is given), interior nodes (where feature tests are performed based on a decision function), and leaf nodes which provide the final output of the task. Figure 5.2 shows the general (classification or regression) architecture for a DT model. Secondly, the internal (or decision) node splits the input into further sub-nodes based on the features (or attributes) selected for splitting that node. This feature selection is done using criteria such as entropy, information gain, or Gini index. For each internal node, the node should be split in a way that ensures that each descending nodes are "purer" than their parents nodes.

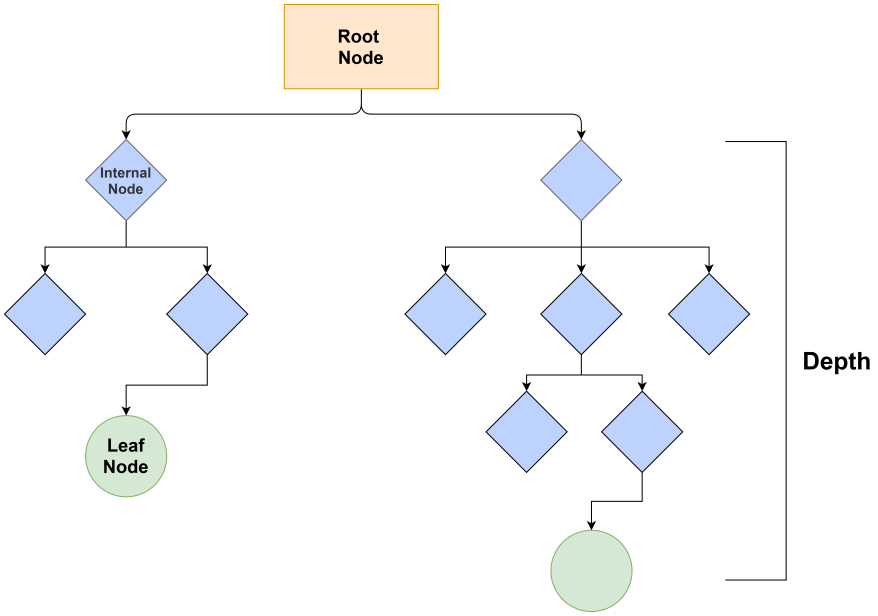


Figure 5.2: Architecture of a Decision Tree model.

This purity represents the goodness of the split. Lastly, the leaf (or terminal) nodes represent the final outcome of the decision tree. In a classification task, leaf nodes are pure if all the input samples belong to the same class. The maximum depth refers to the number of layers created for

the decision-making process. In Figure 5.2, the depth is 4.

For this classification task, the Gini index was used. Consider that a training sample has a probability $p(c_i|T)$ of being assigned to a class c_i at a given node T. Then, the Gini index is given by:

$$G(T) = 1 - \sum_{i=1}^{n} p(c_i|T)$$

The Gini index varies from 0 to 1 where 0 indicates purity and is usually found at a leaf node.

5.2.2 k-nearest Neighbor

k-nearest Neighbor (kNN) is a simple but effective supervised learning classification algorithm that works on the premise of finding k "neighbors" for an input test sample from a distribution of training samples.

Algorithm 3 kNN algorithm

Require: A training set 'S', training set samples S_i, and test set samples T_j, $i = 0...N$, $j = 0...M$
'N' is the size of the training set, 'k' is the number of neighbors, and 'M' is the size of the test set

1: **for** $T_j \in T$ **do**
2: Compute Euclidean or Manhattan distance from T_j to 'k' neighbors
3: Consider neighbors based on the Euclidean distance from T_j
4: Count each neighbor belonging to each class
5: Assign T_j to class with max. no. of neighbors
6: **end for**

The primary sorting criterion used by kNN is based on distance and the two most commonly used distance metrics for kNN are the Euclidean and Manhattan distances. The choice of 'k' is non-trivial and there isn't a specific number that can be identified based on statistical methods. Rather, it is advised to start off with a lower value of 'k' and slowly increase the 'k' value if the performance is poor. Increasing the value of 'k' leads to smoother decision boundaries that allows for better classification. For the classification problem in this research, $k = 3$ was chosen as a starting point to benchmark model performance.

Since the kNN algorithm performs its classification based on distance, appropriate scaling had to be performed so that the classification output would not be biased for features with higher magnitudes. For the following cases, a standard scaler was used to set the variance for each feature to be of unit magntiude and the mean was set to 0. The standard scaler is given by the following equation:

$$S = \frac{I - \mu}{\sigma}$$

where S is the standardized output, I represents the input sample, μ is the mean of the data, and σ is the standard deviation.

5.2.3 Gaussian Naive Bayes

The Gaussian Naive Bayes algorithm is a variation of the Naive Bayes algorithm that is suited for data which follow a normal (or Gaussian) distribution and that have continuous variables in its feature space. The Naive Bayes algorithm is based on the probabilistic Bayes theorem. Used for a variety of classification tasks such as spam filtering and detecting fraudulent activity on a credit card, it is specifically called "Naive" due to the algorithm considering each feature in the feature space to be independent of one another.

The Bayes theorem is given by:

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

where $P(A)$ is the probability of occurrence for event 'A' (variable to be predicted), $P(B)$ is the probability of occurrence for event 'B' (known variable), $P(A|B)$ is the probability of occurrence of event 'A' given event 'B', $P(B|A)$ is the probability of occurrence of event 'B' given event 'A', and $P(A \cap B)$ is the probability of both events happening.

The Gaussian Naive Bayes algorithm uses the following probability density function:

$$P(A|B) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \times e^{\frac{-x-\mu_c}{2\sigma_c^2}}$$

where $sigma_c^2$ and μ_c are the variance and the mean respectively of a continuous variable 'B' for a particular class 'c' for a test sample 'A'.

5.2.4 Evaluation Metrics

In order to evaluate the classification performance of the models, two metrics were chosen. The first evaluation metric is the F1-score [70] (or F-measure) and is given by:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.1}$$

where Precision represents the fraction of actually positive classes given the number of positive predictions, and is given by

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

and Recall represents the fraction of actually positive classes given the number of positive predictions, and is given by

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

where TP is true positive, FP is false positive, and FN is false negative.

Since the F1-score is the harmonic mean of the precision and recall scores and a high F1-score implies high precision and recall scores, the precision and recall metrics have not been directly evaluated on the models.

The second evaluation metric is accuracy [70] and is given by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.4}$$

Accuracy is only used in classification problems and specifically in datasets where the no. of samples in the dataset for each class is balanced. If the no. of samples was unbalanced, the accuracy score would be biased and show an overall higher score that will represent the dominant class.

5.3 Results

5.3.1 Case 1 - Balanced Dataset (Nissan)

The first case for ECU identification was considered for the homogeneous Nissan dataset. The details of this dataset are shown in Table 5.2. The 'Class' represents the numerical class for each function as done during input, 'No. of Samples' represents the number of rows identified for that respective class, 'Class Function' represents the vehicular functions identified for the numerical class, and 'Features' represents the three features considered as inputs: time, data size, and data payload. The models were trained and tested on a 60/40 split.

Table 5.2: Original input Nissan dataset.

Class	No. of Samples	Class Function	Features	Split
Class 0	173,433	Brake	Time, Size, Data	60% and 40%
Class 1	346,785	Steering		
Class 2	346,918	Speed		
Class 3	677,763	Tachometer		
Class 4	72,296	Lighting		
Class 5	8,351,625	Other		

As mentioned in Table 5.1, the input features needed to be converted to appropriate floating point types to serve as valid inputs to each classifier. Please see Figure 5.3 for a pairwise plot that represents each of the features in the dataset as pairwise combinations. The diagonal plots show the distribution plot for feature (univariate). The significant inferences that can be made from the figure are that the distribution for the size of the data follows a right skewed distribution and that the classes (designated as "function" on the right hand side of the figure) are distributed non-linearly when considering the plots for time ("ModTime")/size ("DataSize"), time/data ("ModData"), and

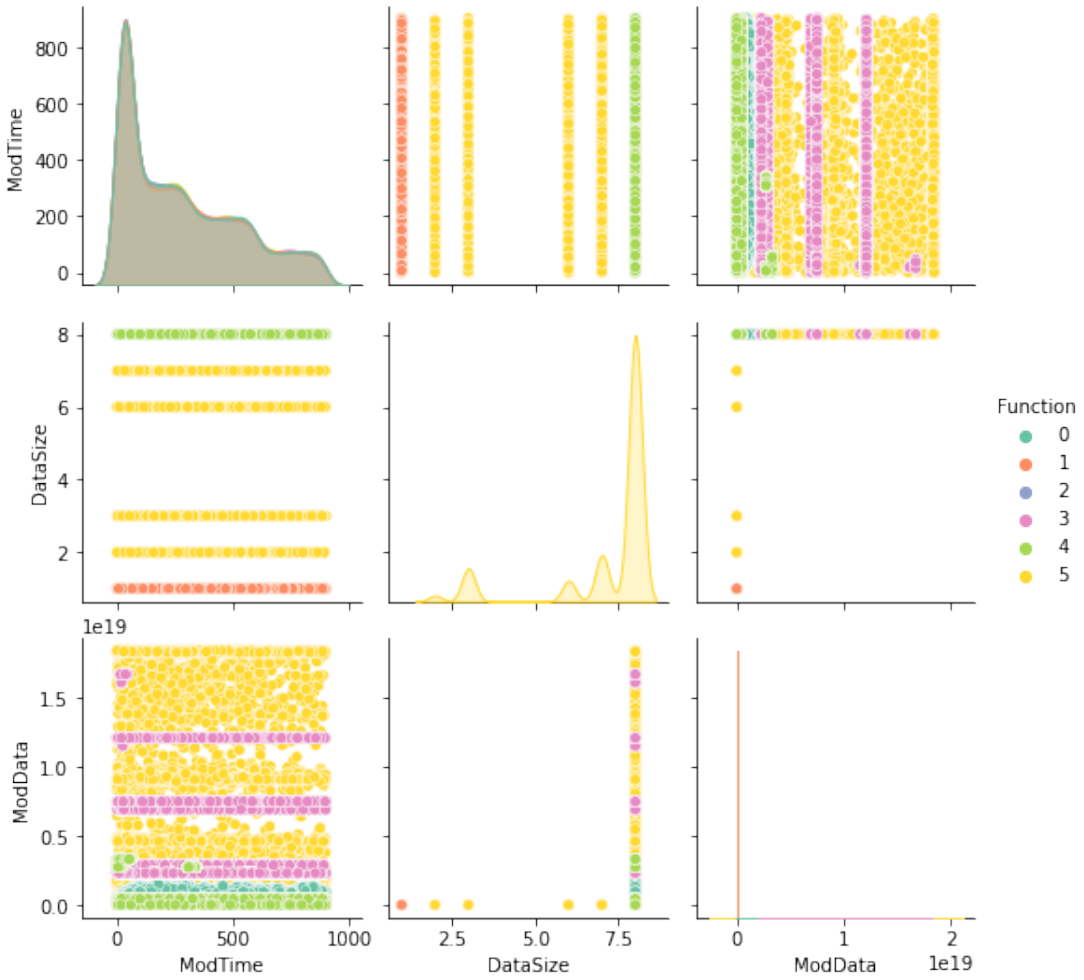


Figure 5.3: Feature-wise pair and distribution plots for Nissan dataset.

size/data.

It can be seen from Table 5.2 that the sample distribution for each class is uneven and providing these as inputs to the machine learning algorithm would bias the classifier performance in favor of the dominant class (class 5). Class 5 represents other ECU IDs that could not be mapped to a function and is the dominant class for this research work. Thus, to assess the models' performance on a balanced dataset, 30,000 samples were chosen for each class as shown in Figure 5.4. Figure 5.5 shows the first five samples of the balanced dataset. The first three attributes represents the three features used for the classification task as mentioned in Table 5.1: "ModTime" is the modified timestamp, "DataSize" is the size of the data frame in bytes, and "ModData" is the data payload for that data frame. "Function" is the class label that represents the data frame is the target

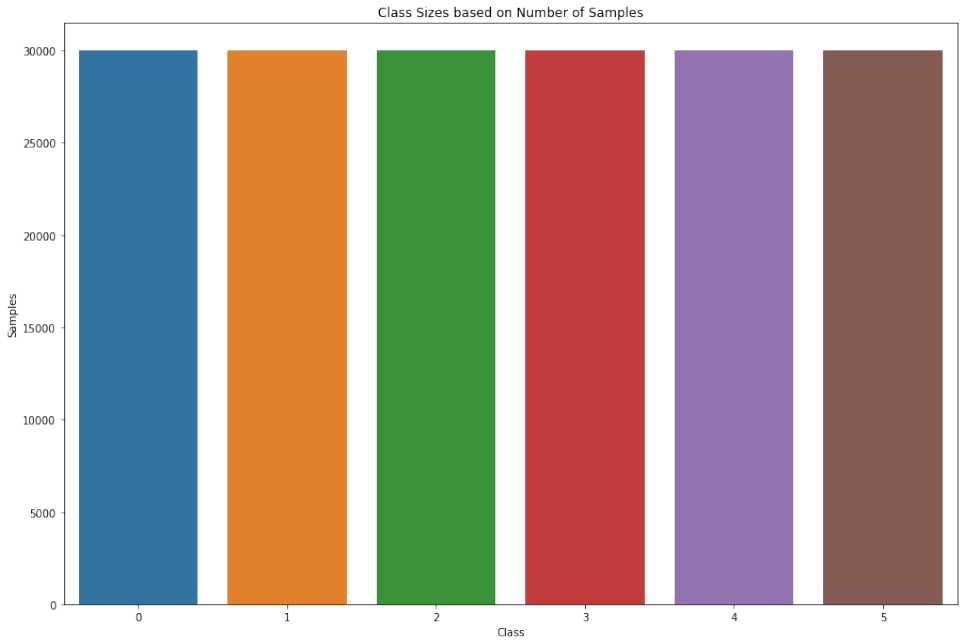


Figure 5.4: Balanced classes for Nissan dataset.

variable.

ModTime	Data Size	ModData	Function
543.00	8.0	72339103382771033	5
493.00	8.0	408021927531	5
264.00	8.0	18374756849254662144	5
5.00	8.0	4900222161891622912	5
544.00	8.0	9726975881377719042	5

Figure 5.5: Snapshot of the first five samples of the Nissan dataset.

Decision Tree Performance

The performance for the DT algorithm was assessed using the metrics of F1-score and accuracy. Additionally, a clearer representation of DT's performance on the Nissan dataset can be seen in Figure 5.6. The x-axis represents the predicted labels whereas the y-axis represents the ground truth. Decision Tree identifies classes 1, 2, and 3 (steering, speed, and tachometer respectively) well whereas the lowest performance is shown for classes 4 and 5 (lighting and other ECUs respectively). The accuracy, F1-score, and processing times are shown in Table 5.3.

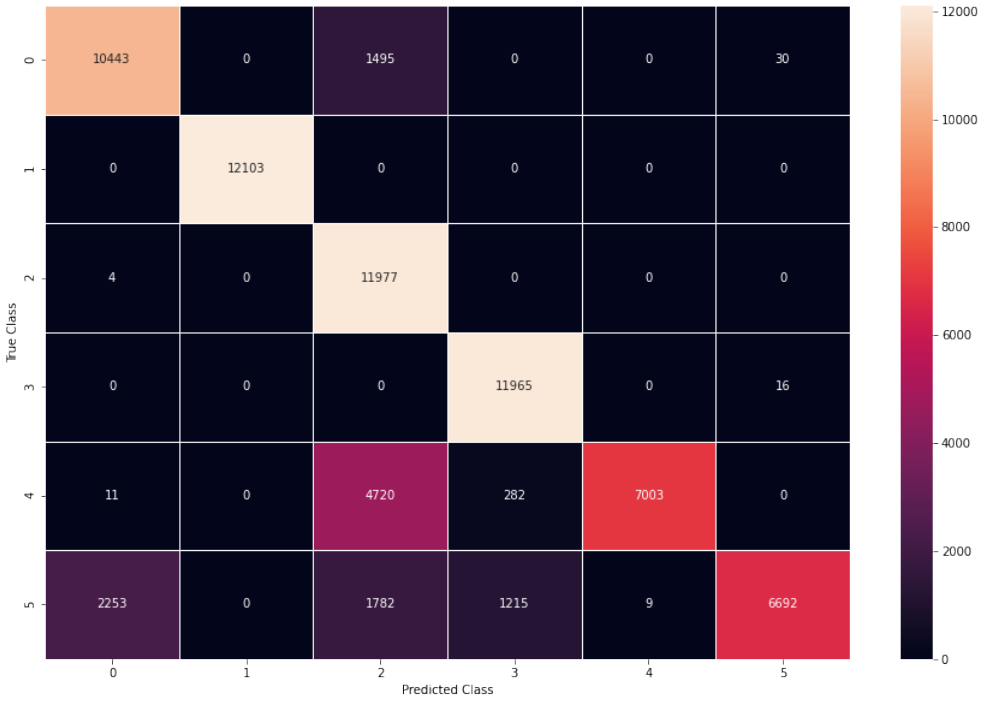


Figure 5.6: Confusion matrix for Decision Tree (Nissan dataset).

Table 5.3: Performance for Decision Tree (Nissan dataset).

Accuracy	F1-score	Training Time	Testing Time
83.1 %	83.5	76.7 ms	3.9 ms

kNN Performance

The confusion matrix for kNN is shown in Figure 5.7. Unlike DT, the kNN algorithm shows near perfect scores for classes 1, 3, and 4 while still performing well in identifying classes 0, 2, and 5. The accuracy, F1-score, and processing times are shown in Table 5.4.

Table 5.4: Performance for kNN (Nissan dataset).

Accuracy	F1-score	Training Time	Testing Time
94.3 %	94.3	219 ms	1580 ms

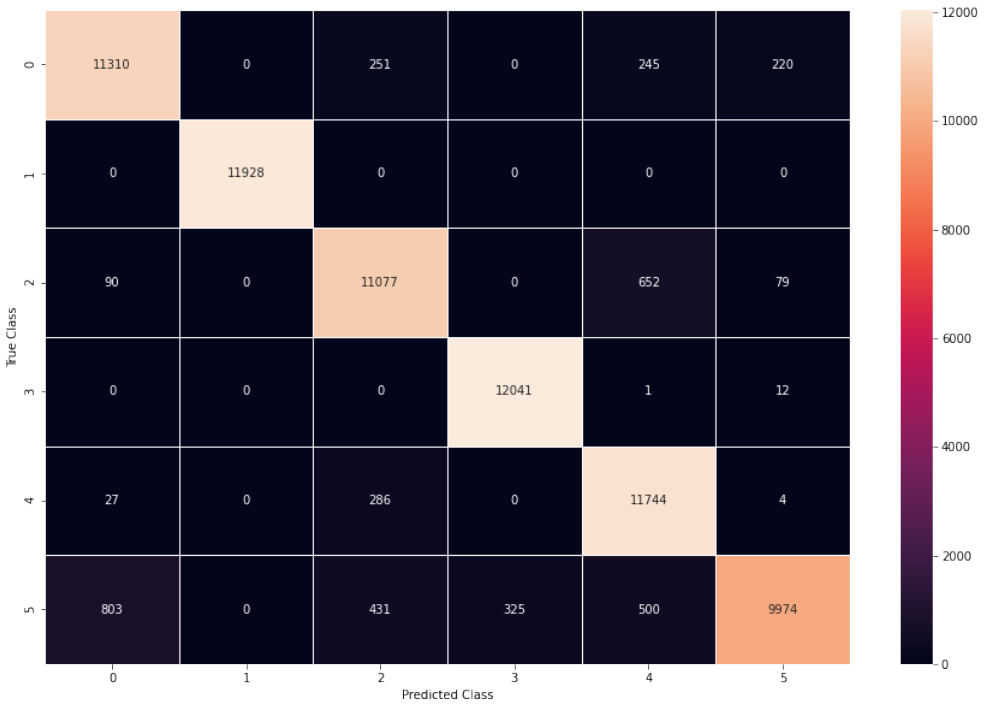


Figure 5.7: Confusion matrix for kNN (Nissan dataset).

Gaussian Naive Bayes Performance

It can be observed from Figure 5.8 that the Gaussian Naive Bayes classifier identifies classes 1 and 2 well but fails to identify the other classes. The reason for this may be due to the distribution of the 3-dimensional data. Since Gaussian Naive Bayes works well for datasets that have a PDF that is represented by a Gaussian distribution, it may perform poorly when the dataset follows other distributions such as binomial or skewed distributions. The accuracy, F1-score, and processing times are shown in Table 5.5.

Table 5.5: Performance for Gaussian Naive Bayes (Nissan dataset).

Accuracy	F1-score	Training Time	Testing Time
54.9 %	61.5	28.9 ms	27.9 ms

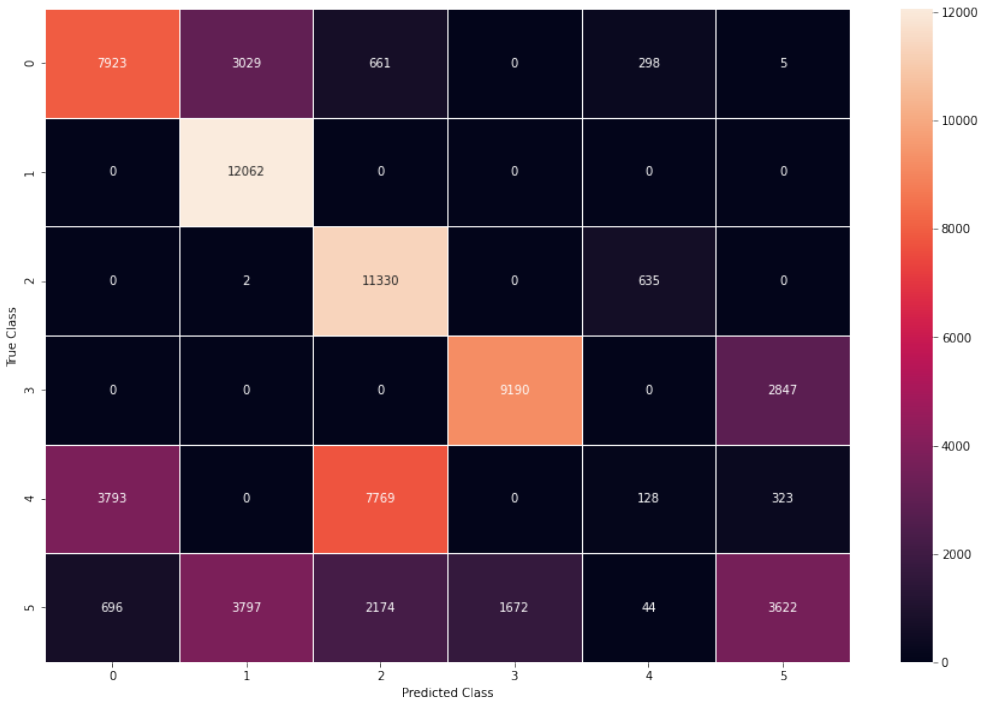


Figure 5.8: Confusion matrix for Gaussian Naive Bayes (Nissan dataset).

5.3.2 Case 2 - Balanced Dataset (Nissan, Honda, and Toyota)

The second case for ECU identification used datasets from the three makes of Nissan, Honda, and Toyota in order to create one heterogenous dataset. The details of this dataset are shown in Table 5.6.

Table 5.6: Original input Nissan, Honda, and Toyota dataset.

Class	No. of Samples	Class Function	Features	Split
Class 0	173,433	Brake	Time, Size, Data	60% and 40%
Class 1	346,785	Steering		
Class 2	346,918	Speed (Nissan)		
Class 3	677,763	Tachometer		
Class 4	72,296	Lighting		
Class 5	8,351,625	Other		
Class 6	60,036	Speed (Honda)		
Class 7	30,057	Speed (Toyota)		

Similar to Case 1, 30,000 samples were chosen for each class as shown in Figure 5.10. Please see Figure 5.9 for a pairwise plot that represents the each of the features in the dataset. The diagonal plots show the distribution plot for feature (univariate). The significant inferences that can be made

from the figure are that the distributions for the size and time features follow a skewed distribution. Unlike Case 1, when looked at closely, there is another distribution that is represented for classes 6 and 7 which does not follow a skewed pattern unlike the distributions for classes 0 through 5. Similar to Case 1, the classes (designated as "function" on the right hand side of the figure) are distributed non-linearly when considering the plots for time ("ModTime")/size ("DataSize"), time/data ("ModData"), and size/data.

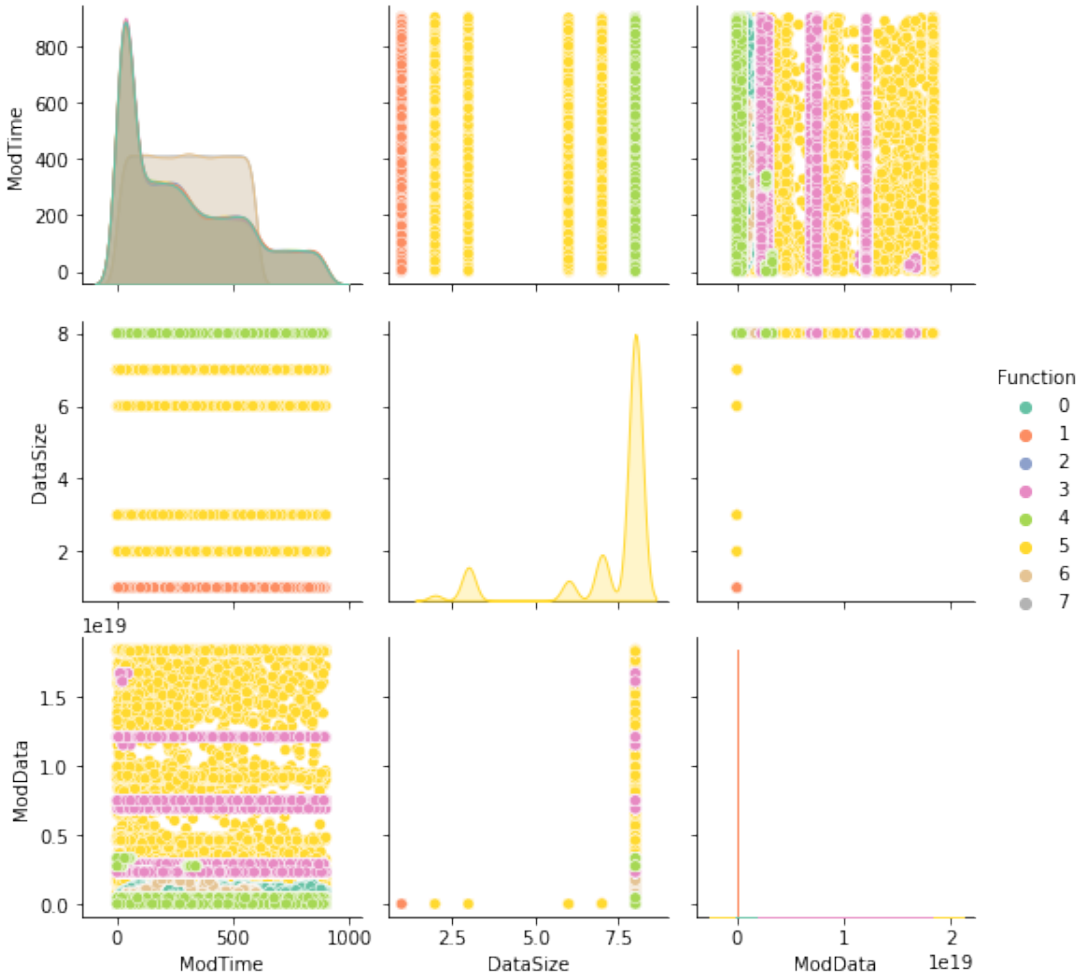


Figure 5.9: Feature-wise pair and distribution plots for Nissan, Honda, and Toyota dataset.

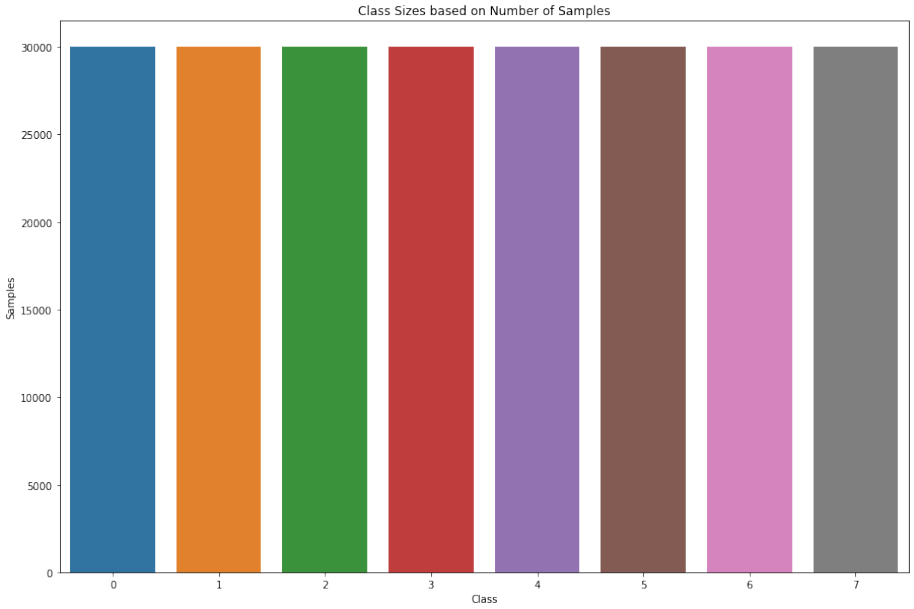


Figure 5.10: Balanced classes for Nissan, Honda, and Toyota dataset.

Decision Tree Performance

As can be observed from Figure 5.11, the performance for DT is consistent in terms of identifying classes 1, 2, 3, and even class 7 which represents the speed ECU for the Toyota dataset. It is also inferred from Table 5.7 that there was a drop of 7.8% and 9.4% respectively in accuracy and F1-score. This can be considered negligible as it is under 10% and the model still continues to show potential for improvement through hyperparameter optimization.

Table 5.7: Performance for Decision Tree (Nissan, Honda, and Toyota dataset).

Accuracy	**F1-score**	**Training Time**	**Testing Time**
75.3 %	74.1	128.6 ms	6.9 ms

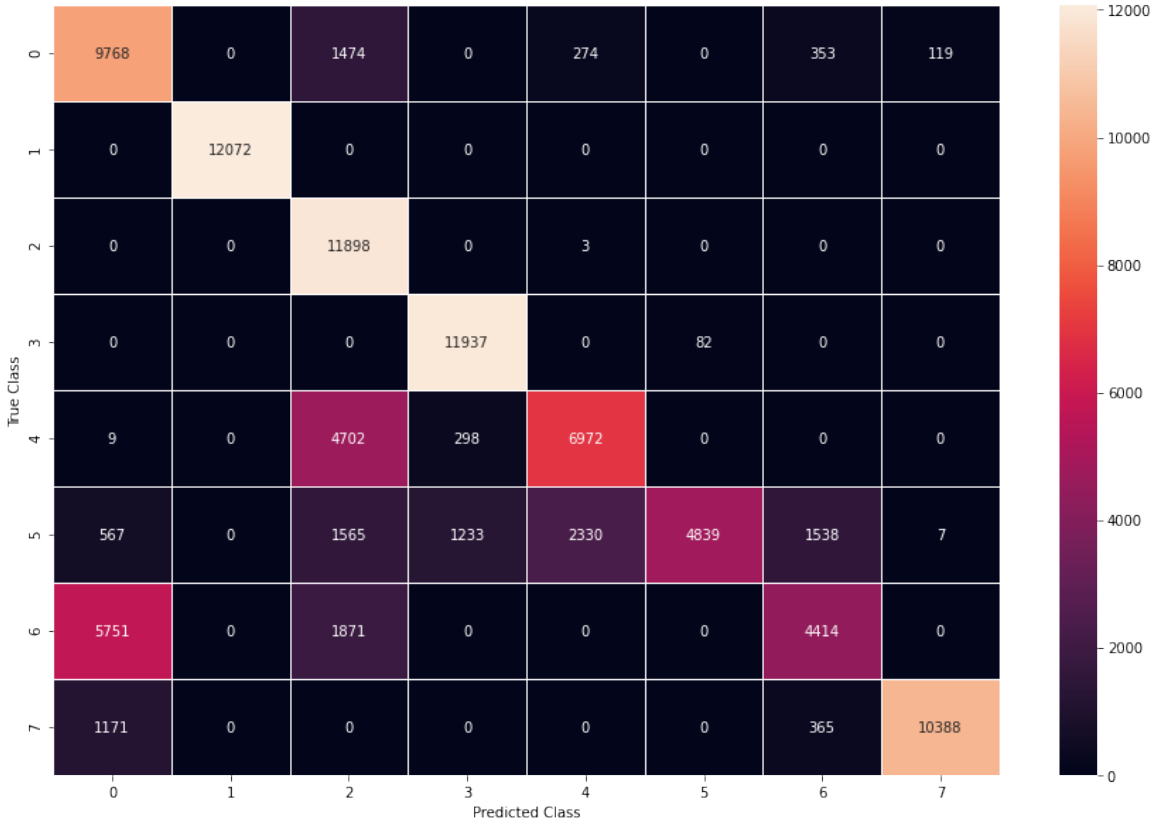


Figure 5.11: Confusion matrix for Decision Tree (Nissan, Honda, and Toyota dataset).

kNN Performance

The kNN algorithm performs well in identifying all classes and has near identical performance when compared to Case 1. Please see Figure 5.12 for the confusion matrix for kNN. The tradeoff with high accuracy and F1-score is the high computation time of kNN as shown in Table 5.8.

Table 5.8: Performance for kNN (Nissan, Honda, and Toyota dataset).

Accuracy	F1-score	Training Time	Testing Time
94.5 %	94.4	740 ms	2317 ms

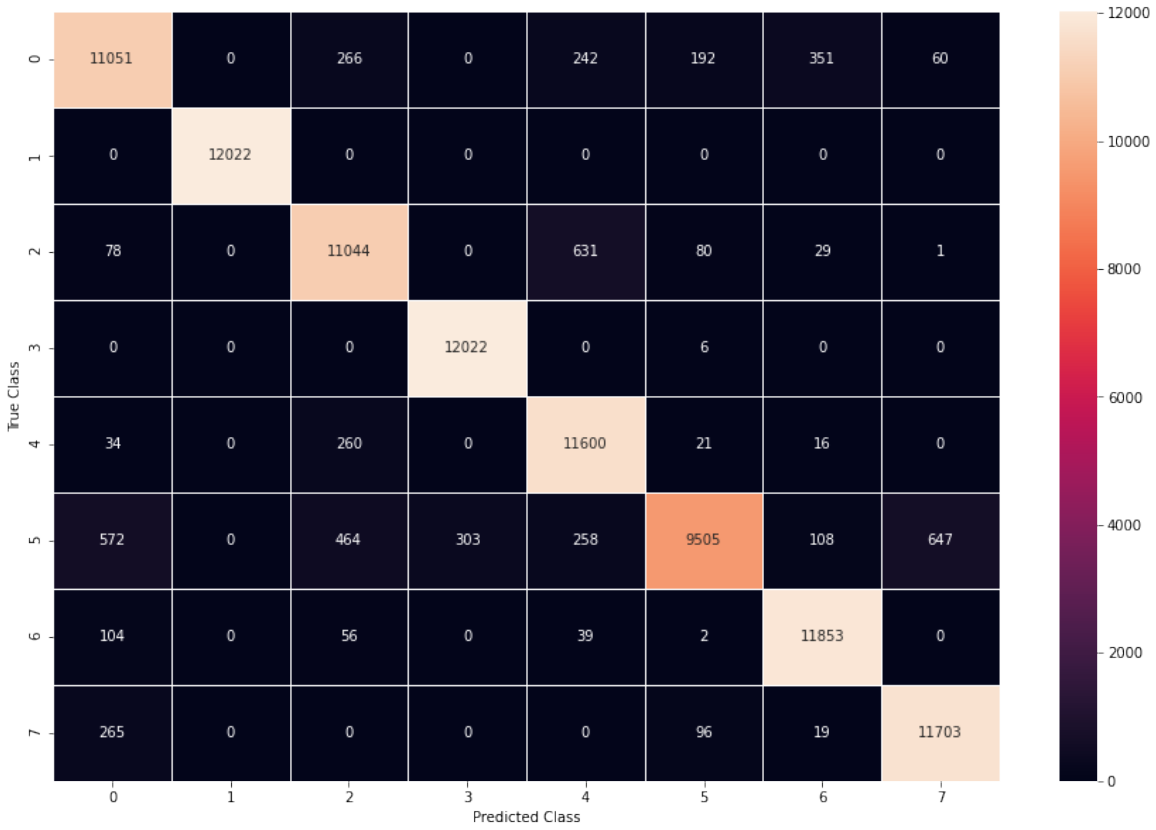


Figure 5.12: Confusion matrix for kNN (Nissan, Honda, and Toyota dataset).

Gaussian Naive Bayes Performance

Gaussian Naive Bayes continues to identify classes 1 and 2 well while performing fairly well in identifying class 3. However, there is a significant drop of 22.2% in the F1-score when compared to Case 1. The training and test times have seen an increase due to the increased number of samples under classes 6 and 7.

Table 5.9: Performance for Gaussian Naive Bayes (Nissan, Honda, and Toyota dataset).

Accuracy	F1-score	Training Time	Testing Time
46.5 %	0.39	740 ms	2317 ms

Figure 5.13: Confusion matrix for Gaussian Naive Bayes (Nissan, Honda, and Toyota dataset).

Discussion

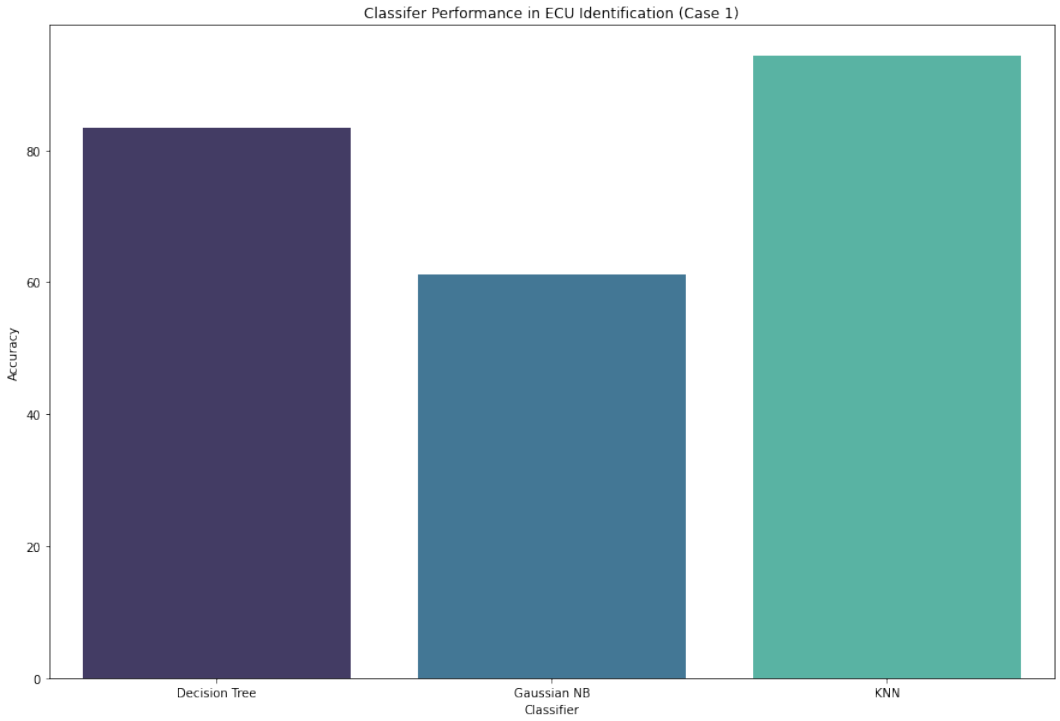
Figure 5.14 and Figure 5.15 show the classifier performance for Case 1 (homogeneous dataset) and Case 2 (heterogeneous dataset) respectively. The highest performing model was identified to be kNN, followed by DT, and Gaussian Naive Bayes. kNN remains the highest performing model due to its ability to perform well in identifying non-linear relationships between features albeit at a higher computation cost as shown in Table 5.10. It is thus implied by these preliminary results that kNN's distance-based approach to classifying each sample shows higher accuracy in contrast to the independency assumption of the Gaussian Naive Bayes model where each feature is assumed to be independent of one another. Though the Naive Bayes classifier has been shown to perform well in many domains, and sometimes even better than kNN [71][72], kNN and Decision Tree hint at being better choices for this classification task. It has to be noted, however, that the Gaussian Naive Bayes performance is not necessarily a reflection of strong feature-dependence in this dataset

and that measuring information loss [73] (i.e. the difference in class information when Naive Bayes assumption is made when compared to a benchmark that measures the mutual information between the input features and the categorical variable) is a better step in understanding the latent relationships within the dataset and improving the Gaussian Naive Bayes performance.

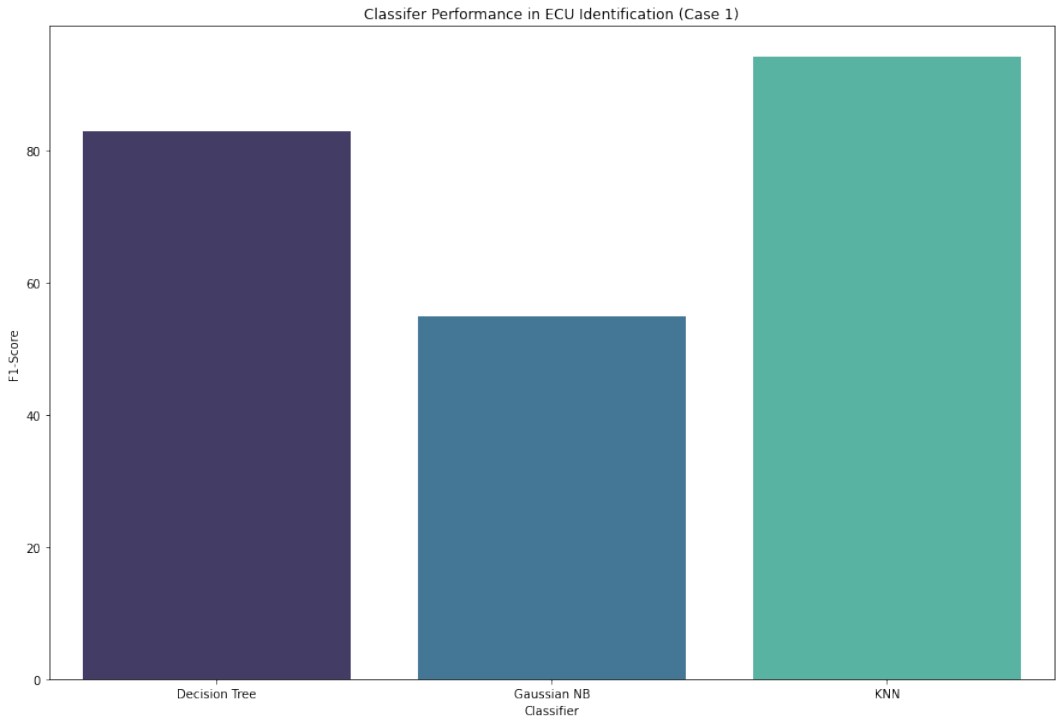
In order to assess whether kNN and Decision Tree models are overfitting, 5-fold cross validation was performed using accuracy and F1-score as metrics. The results for the accuracy and F1-scores using cross validation are shown in Figure 5.16. The results for accuracy and F1-score obtained are the average scores from the 5 folds. Decision Tree gave scores of 75% and 0.66 for the accuracy and F1-scores respectively, while kNN showed high performance again with 95.2% and 0.95 for the accuracy and F1-scores respectively. Since these scores are relatively similar to the ones obtained in both the cases, it can be reasonably inferred that the results obtained for kNN and Decision Tree Cases 1 and 2 are representative of a learning method that is neither underfitting nor overfitting [74] and are the first step in concluding that the models are generalizing well to the training data [75]. DT and kNN may exhibit similar or better performance when cross validation is used in tandem with after hyperparameter optimization methods such as GridSearch or RandomizedSearch to identify the optimal tunable parameters that lowers error and/or increases accuracy/F1-scores.

Table 5.10: Performance for Decision Tree, kNN, and Gaussian Naive Bayes for Cases 1 and 2.

Accuracy	F1-score	Training Time	Testing Time	Model	Case
83.1%	0.83	76.7 ms	3.9 ms	Decision Tree	Case 1
94.3%	0.94	219 ms	1580 ms	kNN	
54.9%	0.61	28.9 ms	27.9 ms	Gaussian Naive Bayes	
75.4%	0.74	128.6 ms	6.9 ms	Decision Tree	Case 2
94.5%	0.94	740 ms	2317 ms	kNN	
46.5 %	0.39	89.4 ms	98.6 ms	Gaussian Naive Bayes	

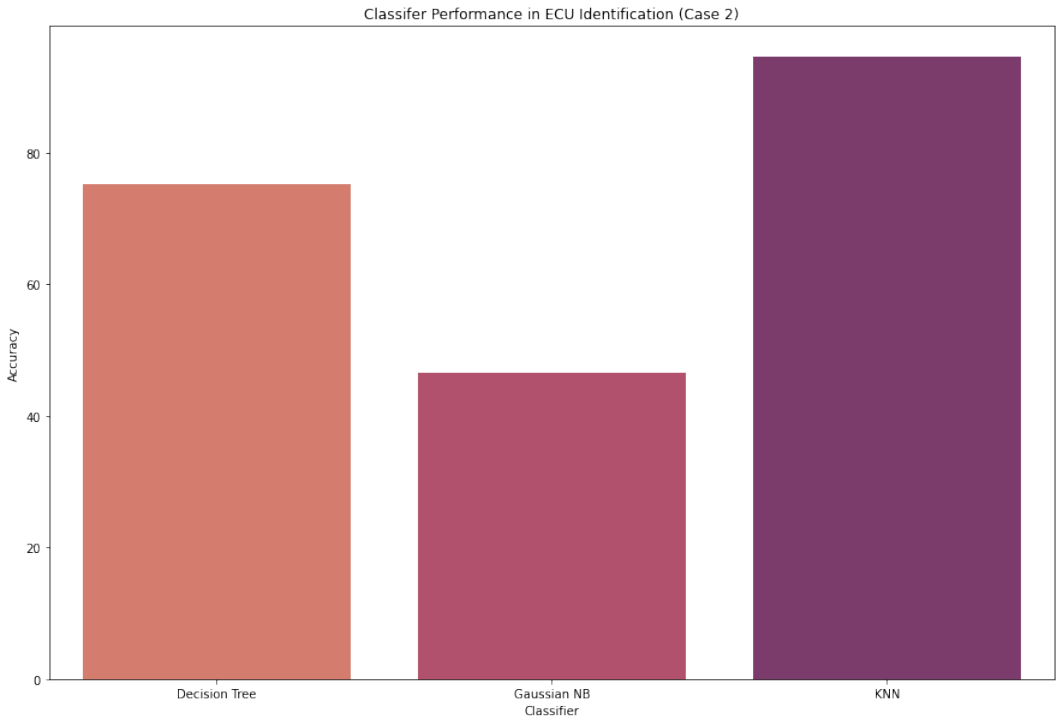


(a) Accuracy scores for Decision Tree, Gaussian Naive Bayes, and kNN for classification (Nissan dataset).

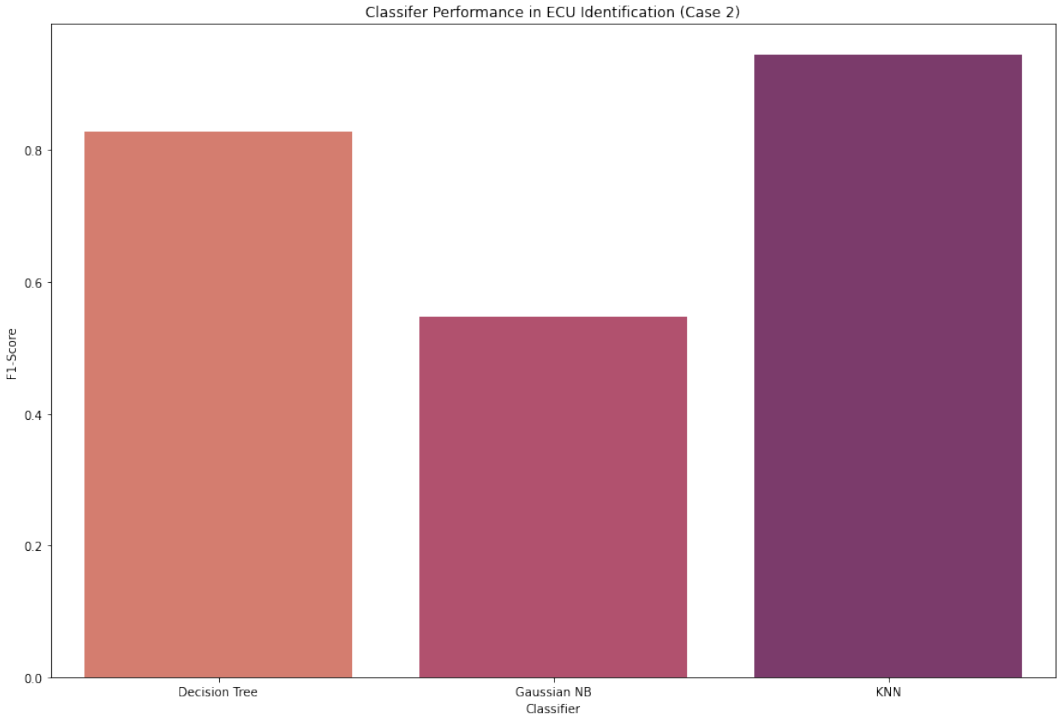


(b) F1-Scores for Decision Tree, Gaussian Naive Bayes, and kNN for classification (Nissan dataset).

Figure 5.14: Model performance for Decision Tree, Gaussian Naive Bayes, and kNN (Case 1).

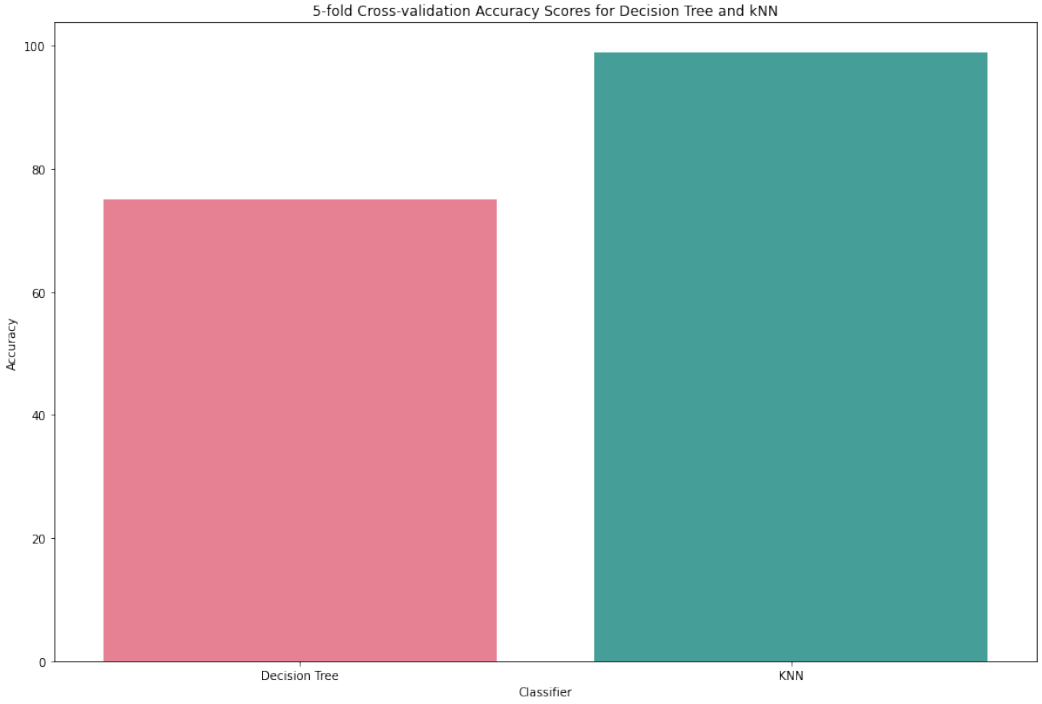


(a) Accuracy scores for Decision Tree, Gaussian Naive Bayes, and kNN for classification (Nissan, Honda, and Toyota dataset).

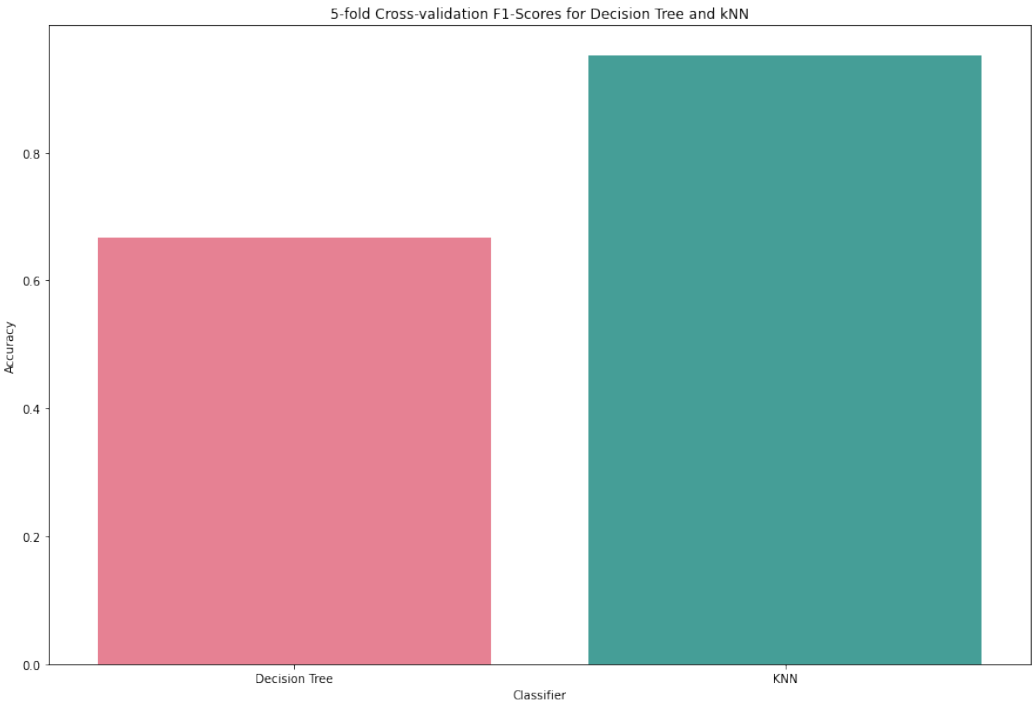


(b) F1-Scores for Decision Tree, Gaussian Naive Bayes, and kNN for classification (Nissan, Honda, and Toyota dataset).

Figure 5.15: Model performance for Decision Tree, Gaussian Naive Bayes, and kNN (Case 2).



(a) 5-fold cross-validation accuracy scores for Decision Tree and kNN.



(b) 5-fold cross-validation F1-scores for Decision Tree and kNN.

Figure 5.16: Model performance for Decision Tree, Gaussian Naive Bayes, and kNN using cross validation.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

A vehicle can have multiple CAN buses that typically operate at bus speeds from 125 kbps to 1 Mbps. This research work analyzed the CAN bus to gather ECU data from multiple makes. The main contributions of this research work are as follows:

1. **Data acquisition and processing:** Analyzing and drawing meaningful conclusions from the CAN data required a methodology. The first step was data collection where data were acquired for three vehicles of Nissan, one vehicle of Honda, and one vehicle of Toyota using COTS hardware and Linux open-source tools to handle the software interface. The 1-, 5-, 10-, and 15-minute datasets were pre-processed to enhance readability, processed to gather ECU ID support, display byte-level waveforms, and remove incompatible data (negative integers and alphanumeric data) to make it feasible for further processing in machine learning tasks such as classification and forecasting.

2. **Application of unsupervised learning:** Two clustering algorithms were analyzed (namely k-means++ and mean shift) for three ECU signatures (speed, steering, and RPM) for two different makes. This two-step approach provides a way for multiple ECUs to be distinguished from one another in order to automate and optimize ECU operations. The ideal number of clusters for the speed and steering ECU signatures was $k = 4$ for the Nissan and Honda makes. There was close agreement on the ranges for the maximum, minimum, and average values (under 1 S.D.) for all but one duration for both the ECU signatures i.e. the 5 minute duration for the steering signature. Average validation errors of 35.7% and 78.8% were observed for speed and steering ECU signatures respectively while the average errors for the 2015 vehicle (based on which the threshold equations were created) were 2.95% and 2.96% for the speed and steering signatures respectively. The lowest errors of 29.9% and 45.9%

for the speed and steering signatures respectively were observed for the 2013 vehicle of the Nissan make at the 5 minute duration. Threshold equations for the RPM signature could not be formulated due to the inability of the mean shift algorithm to identify multiple clusters.

3. **Application of supervised learning:** Three machine algorithms (kNN, Gaussian Naive Bayes, and Decision Tree) were analyzed to classify 5 different ECU signatures (braking, steering, speed, tachometer, and lighting) from various makes based on two cases. The highest performance was shown by the kNN model with average accuracy and F1-scores of 94.4% and 0.94 respectively, followed by Decision Tree with 79.2% and 0.78 respectively, and finally the Gaussian Naive Bayes with 50.7% and 0.50 respectively. A cross validation check was performed to recognize early signs of bias or variance for the kNN and Decision Tree models. The preliminary results show that the models have generalized well based on 3 feature inputs (time, size, and data payload).

Formulating and empirically verifying the effectiveness of the given threshold equations that may vary depending on automobiles' year and make is a complex problem and warrants further research. The limitations of this works are: optimal usage of ECU resources such as energy, time, and latency from the results presented warrants further insight. Secondly, this study only analyzed a limited dataset of five vehicles but preliminary results indicate that there is a positive consensus of similar threshold ranges to some degree of confidence. Thirdly, a robust vehicular CAN dataset needs to be developed to validate the claims in this research work and for large scale analyses for optimal ECU operation and deducing approximate thresholds based on various ECU signatures. Lastly, the cases studied for the classification task assumed that the dataset(s) contain a balanced number of samples for each class which is less likely to be reflective of real-time CAN traffic.

REFERENCES

[1] Deloitte, "Trends and Outlook of the Auto Electronics Industry," *Deloitte China Automotive Practice*, no. October, 2013.

[2] S. Jadhav and D. Kshirsagar, "A survey on security in automotive networks," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, IEEE, 2018, pp. 1–6.

[3] L. van Dijk, "Future vehicle networks and ecus architecture and technology considerations," NXP Semiconductors, Tech. Rep., 2021.

[4] F. Sagstetter *et al.*, "Security challenges in automotive hardware/software architecture design," in *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2013, pp. 458–463.

[5] K. Han, A. Weimerskirch, and K. G. Shin, "Automotive cybersecurity for in-vehicle communication," *IQT QUARTERLY*, vol. 6, no. 1, pp. 22–25, 2014.

[6] W. Zeng, M. A. Khalid, and S. Chowdhury, "In-vehicle networks outlook: Achievements and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1552–1571, 2016.

[7] D. Klinedinst and C. King, "On board diagnostics: Risks and vulnerabilities of the connected vehicle," *CERT Coordination Center, Tech. Rep*, 2016.

[8] S. International and Synopsys, "Securing the Modern Vehicle: A Study of Automotive Industry Cybersecurity Practices," Ponemon Institute, Traverse City, Michigan, Tech. Rep., 2019.

[9] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, 2015.

[10] J. Zhou, P. Joshi, H. Zeng, and R. Li, "Btmonitor: Bit-time-based intrusion detection and attacker identification in controller area network," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 6, pp. 1–23, 2019.

[11] J. Ning, J. Wang, J. Liu, and N. Kato, "Attacker identification and intrusion detection for in-vehicle networks," *IEEE Communications Letters*, vol. 23, no. 11, pp. 1927–1930, 2019.

[12] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, "Identifying ecus using inimitable characteristics of signals in controller area networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 4757–4770, 2018.

[13] S. Park and J.-Y. Choi, "Malware detection in self-driving vehicles using machine learning algorithms," *Journal of advanced transportation*, vol. 2020, 2020.

[14] Z. King and S. Yu, "Investigating and securing communications in the controller area network (can)," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, 2017, pp. 814–818.

[15] T. Matsumoto, M. Hata, M. Tanabe, K. Yoshioka, and K. Oishi, "A method of preventing unauthorized data transmission in controller area network," in *2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*, IEEE, 2012, pp. 1–5.

[16] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, "Cybersecurity challenges in vehicular communications," *Vehicular Communications*, vol. 23, p. 100 214, 2020.

[17] C. University, *Active suspension systems*, https://cecas.clemson.edu/cvel/auto/systems/active_suspension.html, 2021.

[18] T.-Y. Lee, I.-A. Lin, and R.-H. Liao, "Design of a flexray/ethernet gateway and security mechanism for in-vehicle networks," *Sensors*, vol. 20, no. 3, p. 641, 2020.

[19] S.-H. Seo, J.-H. Kim, S.-H. Hwang, K. H. Kwon, and J. W. Jeon, "A reliable gateway for in-vehicle networks based on lin, can, and flexray," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 11, no. 1, pp. 1–24, 2012.

[20] D. Zelle, C. Krauß, H. Strauß, and K. Schmidt, "On using tls to secure in-vehicle networks," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 2017, pp. 1–10.

[21] R. Bosch, "CAN Specification Version 2.0," *Rober Bousch GmbH, Postfach*, vol. 300240, p. 72, 1991.

[22] Volvo, "LIN - Local Interconnect Network," in vol. 60, 2012, pp. 1–9.

[23] M. Ruff, "Evolution of local interconnect network (lin) solutions," in *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484)*, IEEE, vol. 5, 2003, pp. 3382–3389.

[24] A. Umair and M. G. Khan, "Communication technologies and network protocols of automotive systems," *Advances in Networks, SciencePG*, vol. 6, no. 1, pp. 58–65, 2018.

[25] N. Navet and F. Simonot-Lion, *Automotive embedded systems handbook*. CRC press, 2017.

[26] L. E. Frenzel, "Media Oriented Systems Transport (MOST)," *Handbook of Serial Communications Interfaces*, pp. 141–142, 2016.

[27] D. Porter, "100BASE-T1 Ethernet: the evolution of automotive networking," pp. 1–11, 2018.

[28] H. Gharavi, K. V. Prasad, and P. Ioannou, "Scanning advanced automobile technology," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 328–333, 2007.

[29] M. L. Sichitiu and M. Kihl, "Inter-vehicle communication systems: A survey," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 2, pp. 88–105, 2008.

[30] K. Dar, M. Bakhouya, J. Gaber, M. Wack, and P. Lorenz, "Wireless communication technologies for its applications [topics in automotive networking]," *IEEE Communications Magazine*, vol. 48, no. 5, pp. 156–162, 2010.

[31] B. Cash, "GHz DEDICATED SHORT RANGE COMMUNICATION (DSRC) OVERVIEW," *University of British Columbia*, no. Tutorial, 2005.

[32] C. Bettisworth *et al.*, "Status of the Dedicated Short-Range Communications Technology and Applications Report to Congress," *United States Department of Transportation. Intelligent Transportation Systems Joint Program Office*, no. July, 2015.

[33] B. Shrestha, "Wireless Access in Vehicular Environments (WAVE) Channel Coordination," *Thesis*, 2008.

[34] S. Zeadally, J. Guerrero, and J. Contreras, "A tutorial survey on vehicle-to-vehicle communications," *Telecommunication Systems*, vol. 73, no. 3, pp. 469–489, 2020.

[35] D. Kombate *et al.*, "The internet of vehicles based on 5g communications," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, IEEE, 2016, pp. 445–448.

[36] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, "Study on zigbee technology," in *2011 3rd International Conference on Electronics Computer Technology*, IEEE, vol. 6, 2011, pp. 297–301.

[37] Dallas Semiconductor, "An Introduction to Direct-Sequence Spread-Spectrum Communications," vol. 2, 2003.

[38] Y. Hirakata, A. Nakamura, K. Ohno, and M. Itami, "Navigation system using zigbee wireless sensor network for parking," in *2012 12th International Conference on ITS Telecommunications*, 2012, pp. 605–609.

[39] P. Kochar and M. Supriya, "Vehicle speed control using zigbee and gps," in *International Conference on Smart Trends for Information Technology and Computer Communications*, Springer, 2016, pp. 847–854.

[40] A. Tufail, M. Fraser, A. Hammad, K. K. Hyung, and S.-W. Yoo, "An empirical study to analyze the feasibility of wifi for vanets," in *2008 12th international conference on computer supported cooperative work in design*, IEEE, 2008, pp. 553–558.

[41] C.-M. Chou, C.-Y. Li, W.-M. Chien, and K.-c. Lan, "A feasibility study on vehicle-to-infrastructure communication: Wifi vs. wimax," in *2009 tenth international conference on mobile data management: systems, services and middleware*, IEEE, 2009, pp. 397–398.

[42] Z. El-Rewini, K. Sadatsharan, N. Sugunaraj, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, "Cybersecurity attacks in vehicular sensors," *IEEE Sensors Journal*, vol. 20, no. 22, pp. 13 752–13 767, 2020.

[43] S. Abdelhamid, H. S. Hassanein, and G. Takahara, "Vehicle as a mobile sensor," *Procedia Computer Science*, vol. 34, pp. 286–295, 2014.

[44] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.

[45] T. G. Reid *et al.*, "Localization requirements for autonomous vehicles," *arXiv preprint arXiv:1906.01061*, 2019.

[46] S. Campbell *et al.*, "Sensor technology in autonomous vehicles: A review," in *2018 29th Irish Signals and Systems Conference (ISSC)*, IEEE, 2018, pp. 1–4.

[47] S. E. Reutebuch, H.-E. Andersen, and R. J. McGaughey, "Light detection and ranging (lidar): An emerging tool for multiple resource inventory," *Journal of forestry*, vol. 103, no. 6, pp. 286–292, 2005.

[48] N. O. Center and A. A. N. C. Services, "Lidar 101 : An Introduction to Lidar Technology , Data , and Applications," *NOAA Coastal Services Center*, no. November, p. 76, 2012.

[49] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, "Advanced driver-assistance systems: A path toward autonomous vehicles," *IEEE Consumer Electronics Magazine*, vol. 7, no. 5, pp. 18–25, 2018.

[50] M. Rajasekhar and A. K. Jaswal, "Autonomous vehicles: The future of automobiles," in *2015 IEEE International Transportation Electrification Conference (ITEC)*, IEEE, 2015, pp. 1–6.

[51] Y. Wang, *Frequency modulated continuous wave radar system at ism band for short range indoor positioning*, 2017.

[52] A. Ascher, "Automotive Radar Performance Characteristics," *IEEE*, 2019.

[53] F. Jansen, "Automotive radar sensor for ultra short range applications," in *2017 18th International Radar Symposium (IRS)*, 2017, pp. 1–6.

[54] G. Duggal, S. Vishwakarma, K. V. Mishra, and S. S. Ram, "Doppler-resilient 802.11 ad-based ultra-short range automotive radar," *arXiv preprint arXiv:1902.01306*, 2019.

[55] H. Hatano, T. Yamazato, and M. Katayama, "Automotive ultrasonic array emitter for short-range targets detection," in *2007 4th International Symposium on Wireless Communication Systems*, IEEE, 2007, pp. 355–359.

[56] W. J. Fleming, "New automotive sensors—a review," *IEEE Sensors Journal*, vol. 8, no. 11, pp. 1900–1921, 2008.

[57] P. A. Wagh, R. R. Pawar, and S. Nalbalwar, "Vehicle speed control and safety prototype using controller area network," in *2017 International Conference on Computational Intelligence in Data Science (ICCIDS)*, IEEE, 2017, pp. 1–5.

[58] S. Dabral, S. Kamath, V. Appia, M. Mody, B. Zhang, and U. Batur, "Trends in camera based automotive driver assistance systems (adas)," in *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)*, IEEE, 2014, pp. 1110–1115.

[59] L. Young, "Digital Eye," *Bus and Coach Engineering*, no. June 2016, pp. 35–36, 2019.

[60] S. Cai, M. Becherif, and M. Wack, "Context system using pervasive controller area network bus system to improve driving safety," in *Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, IEEE, 2010, pp. 560–565.

[61] S. Hartzell, C. Stubel, and T. Bonaci, "Security analysis of an automobile controller area network bus," *IEEE Potentials*, vol. 39, no. 3, pp. 19–24, 2020.

[62] K. Miyashita, T. Takahashi, and M. Yamanaka, "Features of a magnetic rotary encoder," *IEEE Transactions on Magnetics*, vol. 23, no. 5, pp. 2182–2184, 1987.

[63] K. Miyashita, T. Takahashi, S. Kawamata, S. Morinaga, and Y. Hoshi, "Noncontact magnetic torque sensor," *IEEE Transactions on Magnetics*, vol. 26, no. 5, pp. 1560–1562, 1990.

[64] T. Seel, M. Kok, and R. S. McGinnis, *Inertial sensors—applications and challenges in a nutshell*, 2020.

[65] I. Rouf *et al.*, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study.," in *USENIX Security Symposium*, vol. 10, 2010.

[66] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[67] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Proceedings of the seventh IEEE international conference on computer vision*, IEEE, vol. 2, 1999, pp. 1197–1203.

[68] C. Young, J. Svoboda, and J. Zambreno, "Towards reverse engineering controller area network messages using machine learning," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, IEEE, 2020, pp. 1–6.

[69] O. Avatefipour, A. Hafeez, M. Tayyab, and H. Malik, "Linking received packet to the transmitter through physical-fingerprinting of controller area network," in *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, IEEE, 2017, pp. 1–6.

[70] H. Dalianis, "Evaluation metrics and evaluation," in *Clinical Text Mining*, Springer, 2018, pp. 45–53.

[71] L. Dey, S. Chakraborty, A. Biswas, B. Bose, and S. Tiwari, "Sentiment analysis of review datasets using naive bayes and k-nn classifier," *arXiv preprint arXiv:1610.09982*, 2016.

[72] M. J. Islam, Q. J. Wu, M. Ahmadi, and M. A. Sid-Ahmed, "Investigating the performance of naive-bayes classifiers and k-nearest neighbor classifiers," in *2007 International Conference on Convergence Information Technology (ICCIT 2007)*, IEEE, 2007, pp. 1541–1546.

[73] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, 2001, pp. 41–46.

[74] D. Berrar, *Cross-validation.* 2019.

[75] R. B. Rao, G. Fung, and R. Rosales, "On the dangers of cross-validation. an experimental evaluation," in *Proceedings of the 2008 SIAM international conference on data mining*, SIAM, 2008, pp. 588–596.