UNIVERSITY OF CALIFORNIA,
IRVINE

Building Intelligent IoT Agents Using Personality-Based Service Models

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

Zahraa Marafie

Dissertation Committee:
Professor Kwei-Jay Lin, Chair
Professor Nader Bagherzadeh
Professor Fadi Kurdahi

2021

DEDICATION

I dedicate this dissertation work to my advisor, Professor Kwei-Jay Lin, who has upheld me throughout the journey. Thank you for being there during the ups and downs that Ph.D. can bring. Words cannot express how thankful I am to have him as my advisor. People like Professor Lin inspire their students each and every day, empowering them to become great people in this world.

I also dedicate my dissertation work to my family and many friends. A special feeling of appreciation to my beloved husband, Ayoub, without whose constant support, this dissertation work was not possible. My thanks also go to my caring parents, dear siblings, and friends, who have supported this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am grateful to the following people, without whom I would not have completed this research and without whom I would not have made it through my Ph.D. degree! Firstly, I would like to give my sincere appreciation to my advisor, Prof. Kwei-Jay Lin, for his continuous support throughout this work and his patience, motivation, and immense knowledge.

Besides my advisor, I would like to express my gratitude to the rest of my thesis committee: Prof. Nader Bagherzadeh, and Prof. Fadi Kurdahi, for their insightful comments and encouragement, and also their question, which incented me to widen my research from diverse perspectives. I would like to send special thanks to Prof. Bagherzadeh, who's also been my MS advisor, for his guidance and support throughout those years.

I also would like to thank my colleagues, Yu Meng, Daben Wang, Haoyu Lyu, Yanan Lui, Zehua Wang, and Mason Ma, for their countless hours, long discussions, and hard work that lead to the success of this work.

I appreciate all those who have participated in this work's discussions, data collection phases, and experiments. Without their precious support, it would not be possible to conduct this research.

VITA

Zahraa Marafie

EDUCATION

Doctor of Philosophy in Electrical and Computer Engineering **2021**
University of California, Irvine *Irvine, California*

Masters of Science in Electrical and Computer Engineering **2016**
University of California, Irvine *Irvine, California*

Bachelor of Science in Information Science **2011**
Kuwait University *Kuwait*

RESEARCH EXPERIENCE

Graduate Research Assistant **2014–2021**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant **2011**
Kuwait University *Kuwait*

Data Structures
Computer Architecture

Grader **2018**
University of California *Irvine, California*

Real-time Systems

Grader **2019**
University of California *Irvine, California*

Real-time Systems

REFEREED JOURNAL PUBLICATIONS

AutoCoach: An Intelligent Driver Behavior Feedback Agent with Personality-Based Driver Models **June 2021**
Special Issue Applications of Next-Generation IoT, Electronics, MDPI

REFEREED CONFERENCE PUBLICATIONS

AutoCoach: Driving Behavior Management Using Intelligent IoT Services **November 2019**
IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)

ProActive FinTech: Using Intelligent IoT to Deliver Positive InsurTech Feedbacks **July 2018**
IEEE 20th Conference on Business Informatics (CBI)

SOFTWARE

AutoCoach Agent
An Android-based driving behavior management application designed and implemented to evaluate the personality-based concept we propose in this dissertation.

ABSTRACT OF THE DISSERTATION

Building Intelligent IoT Agents Using Personality-Based Service Models

By

Zahraa Marafie

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Irvine, 2021

Professor Kwei-Jay Lin, Chair

In the fast-paced world of technology, life is transforming into a connected universe of Internet-of-Things (IoT), assisting human needs and creating a better world. United with AI, IoT technologies that once could only be dreamed of have become a reality. This work introduces AutoCoach, an IoT-based intelligent agent categorizing drivers into various driving personalities and providing individualized feedback through machine learning models. AutoCoach is intended for improving automobile drivers' performance by applying persuasive technology. System models like Advanced driver-assistance (ADAS) and some Usage-based-Insurance (UBI) aim to increase road safety. However, most prior models do not consider the differences between driving habits. When feedback is offered through those systems, not everyone may respond to them.

While many of those systems look into driving behavior as a single event (e.g., lane-change), a driving behavior may be judged based on the historical data to create a personalized experience leading to better feedback. We created AutoCoach, Cloud-based Android software that collects, analyzes, and learns from a driver's previous driving data to deliver individualized, effective feedback. The AutoCoach design includes two unique components to build an effective persuasive system. The first component is the personality classification, which recognizes drivers' personalities by analyzing driving behavior patterns and habits. At the

same time, the second component is the rewarding system, which determines the current driving behavior's risk score based on some immediate past behavior. Once we have identified the driver's personality and risk scores, when to offer feedback to the driver becomes a critical question. We propose the memory factor model, which decides when to provide feedback to drivers based on their personalities. This memory factor identifies the most critical behaviors within a flexible period. AutoCoach then decides on feedback to maintain safe driving or improve awareness for risky driving habits. By applying means of persuasive technology, friendly feedback and suggestion services can be given to drivers to help them improve their behaviors. The system interacts with the users through its GUI interface that provides real-time user feedback for warnings and rewards through its simple yet effective design.

To test the success of the AutoCoach v1.0 design, we have created an event identification model using data collected from 10 drivers driving different car models. Moreover, we created the personality model based on data collected from 50 drivers driving in different cities. Each driver was assigned to a personality group based on his past driving behavior. Then, we conducted a user study to evaluate the feasibility and acceptability of AutoCoach by comparing a personality-based model with a non-personality model. The purpose of the study is to understand the drivers' level of acceptance of personalized personality-based driving behavior management systems compared to non-personalized systems by considering how feedback frequency plays a role in sending more effective feedback to drivers. Evaluations used the within-subject mixed design method where 8 participants (2 from each group) participated. The study found that the frequency and timeliness of feedback play a role in creating more perceived intelligence. Lowering feedback and giving it only when truly needed reduces drivers' stress caused by frequent feedback. When we have assigned different policies to drivers, we found that riskier drivers have noticed AutoCoach more and were more grateful for the feedback offered by AutoCoach. Our study findings show that our approach is feasible and beneficial in improving the user experience when utilizing a personality-based driving

agent, with 61% agreement that the personality-based model is more intelligent than the non-personality-based systems. We have upgraded the agent (AutoCoach v1.1) to improve the feedback and rewards engines when driving under different contexts. The improved version detects the road type and conditions and decides other feedback and rewarding strategies. The new improvement allowed assigning a more responsive policy to all drivers, allowing higher sensitivity to risk and more frequent feedback. We have conducted another user study requiring drivers to test different agent versions on local roads and highways to evaluate our system. From the user's point of view, v1.1 is 19% more accepted as an intelligent driving behavior management system than v1.0. Results indicate that AutoCoach successfully executes new AI algorithms and capabilities and is a promising concept that can be extended to other fields.

Chapter 1

Introduction

1.1 Motivation

Intelligent IoT (IIoT) systems are guiding the way in many fields like healthcare [56, 53], smart cities [2], smart living [22], and transportation [66]. From the business perspective, adopting IoT and AI brings new revenue sources from risk and non-risk-based information. Over time, AI learns from the environment and clients and might become more productive than humans. It also utilizes data assets and creates valuable information. Because of the increasing sophistication of its decision-making capabilities, AI Moreover, adopting technologies increases new product speed-to-market as AI techniques are speeding up decision-making. Computers can make decisions and give advice faster than a human can do. It also can support existing advisors and bring direct-to-consumer solutions. Furthermore, IoT provides remote access and data capture. The ability to collect and analyze vast amounts of data will allow businesses to shift from protection models (i.e., reactive) to more sophisticated prevention models by developing a more granular view of the risk (i.e., proactive), thus enabling personalization. Further, it gives new business opportunities as personalized

solutions. For example, connected health devices reduce the costs of future health complications happening because of patients neglecting regular checkups and proper healthcare. Medical advice builds trust between consumers and the enterprise. Health IoT-based services offer to support customers' health and prevent future health complications by alerting them when signs of illness are present. Furthermore, ride-sharing will offer different premium options where there will be self-driven and auto-driven cars, and this can bring more business opportunities, for instance, companies. Customers who show they take good care of themselves or showed good behavior are of higher value and can get rewards or premium discounts. AI improves the efficiency of customer interaction and conversion ratios, reducing quote-to-bind. Customers will be happy about having to pay only for how much they use.

One emerging IoT trend in the field of transportation is the automobile Usage-based Insurance (UBI) services [57, 70]. It is an IoT-based business innovation that aligns the driving behavior with the premium charges through data generated by the On-Board-Diagnostics (OBD) and telematics devices [6]. Pay-how-you-Drive (PHYD/PHUD) is the most promising UBI model [45] that is growing in demand due to the IoT revolution [48, 5]. However, currently available PHYD models mostly present feedback passively during or after the trip in the form of scores, issues, or monetary costs [13]. Services like PHYD can serve as driving behavior management systems and support drivers in improving their driving habits over time through persuasion. Improving behavior increases road safety and reduces traffic fatalities. According to the WHO, about 1.25 million traffic deaths occur worldwide every year [49], making the need for solutions to eliminate the road risks a necessity. We believe that by providing proper feedback through driving behavior management systems at the right time, we can support drivers' proactive response under high-risk situations, supporting drivers' actions and ultimately saving lives. Therefore, we design an agent who understands the differences between drivers' personalities and provides proper feedback based on their needs.

Designing such feedback systems requires collecting and analyzing motion sensed data in order to detect and analyze behaviors. With smartphones' mature sensing capabilities [9], researchers have been intensively studying their use to provide solutions to driving-related problems [16]. Although driving behavior detection has been studied, less attention has been paid to providing adequate feedback about driving behavior. In this work, we propose an innovative solution that understands people's driving style differences. Such style differences may be used to offer personality-based personalized feedback to improve driving safety.

Most smart driving behavior monitoring systems on the current market do not use personality to offer such personalized feedback that is more useful for drivers. On the other hand, some IoT systems try to monitor each user individually, but that may require very complex human behavior analysis to offer helpful feedback. Using the personality group models, we create a simple yet effective system's smartness that simplifies the human behavior modeling to give group-based feedback. Assigning a standard policy for each group creates an efficient system design that reduces processing time compared with complex system policies for many different drivers. It has been found that several driving behaviors are affected by emotions linking aggression to accidents [36]; this indicates that riskier drivers are usually those who are angry, anxious, or simply careless. This work presents AutoCoach, a personalized personality-based driving behavior management system to coach drivers to improve their driving behavior over time. We have implemented cloud-based systems that collect actual driving information to learn about user driving behavior styles, including accelerations, brakes, and turns, and use them to detect similarities and differences between drivers in a personality model. Our project has designed a driver-personality model to be used by machine learning algorithms to cluster drivers into different risk groups according to drivers' personalities. We then use a driver's personality attribute in a feedback policy model that decides how personal feedback should be given when aggression is detected. Our policy design also tries to give gradual positive-behavior support based on the drivers' differences and personality assignments.

In this work we propose several concepts applied in our project to enhance the currently available models by presenting personality into such systems. We summarize the concepts in the upcoming sections.

1.2 Improving Driving Behavior

An intelligent IoT system can be designed to discover the root cause of the score depletion and offer advice based on sensor-collected data. Advice may be as simple as requesting less over-speeding and improving braking habits. It may also decide if it is due to the road traffic condition or surrounding cars' speed. More elaborate consideration could be performed by Cloud Analytics using the current car's location and other context information such as weather, traffic conditions, accident reports, and others. An intelligent system should reduce its warning output or frequency if the causes were not within the driver's control.

Usually, drivers' scores tend to fluctuate within certain levels.By understanding the causes of score drops, the system can be designed to improve the driving habits on a gradual basis by offering positive feedback from time to time. With conditional and positive feedback, drivers can become more aware of their driving habits and may improve driving skills over time to reduce risks. This is the goal of our work; to make the system more friendly yet persuasive.

We believe that feedback should not only be given for showing current driving status but also provide feedback to protect drivers from unforeseen driving behavior issues. Drivers tend to show worse driving habits with longer trips. It may be because drivers are getting tired and no longer willing to perform better driving habits due to the length of the trip. Proactive FinTech may use this information to understand drivers and increase their scores by giving proactive feedback. The system may inform the driver of the additional risk when

the trip duration exceeds his normal range.

1.3 Abnormal Behavior Identification and Assessment

In our proactive feedback system design, drivers become aware of the reasons for risk levels to rise or decline, making it possible for the system to notify the driver about the conditions and situations affecting their risk levels. The on-board system communicates with the Cloud Analytics server reporting the current driver's behavior. Cloud Analytics can help decide if current drive reports are considered normal driving behaviors for the specific driver. If any behavior shows an abnormality, then the system becomes aware of it and points out that issue to the driver positively. We can design a proactive feedback system that uses historical data of the abnormal events to predict when similar conditions are present. Those certain inopportune situations and driving behaviors might occur; therefore, providing feedback allows drivers to respond to similar events proactively. Cloud Analytics considers learning from the combination of factors affecting driving behavior. During a trip, if it has been seen that some factors are close to causing similar situations that have happened in the past and the current drive is showing poor driving behavior, the system needs to send out recommendations. Those recommendations entitle the driver to be perceptive that certain conditions or behaviors affect his drive or trip and may positively respond to the event.

1.4 Designing Proactive User Feedback

Based on research, UBI policyholders showed improvement in drive score throughout the time when feedback is offered [55]. Also, it has shown that when drivers know that they are being monitored, they tend to drive better. Then, when drivers are confronted about their driving behavior after the event, there had been seen a 20% improvement in driving

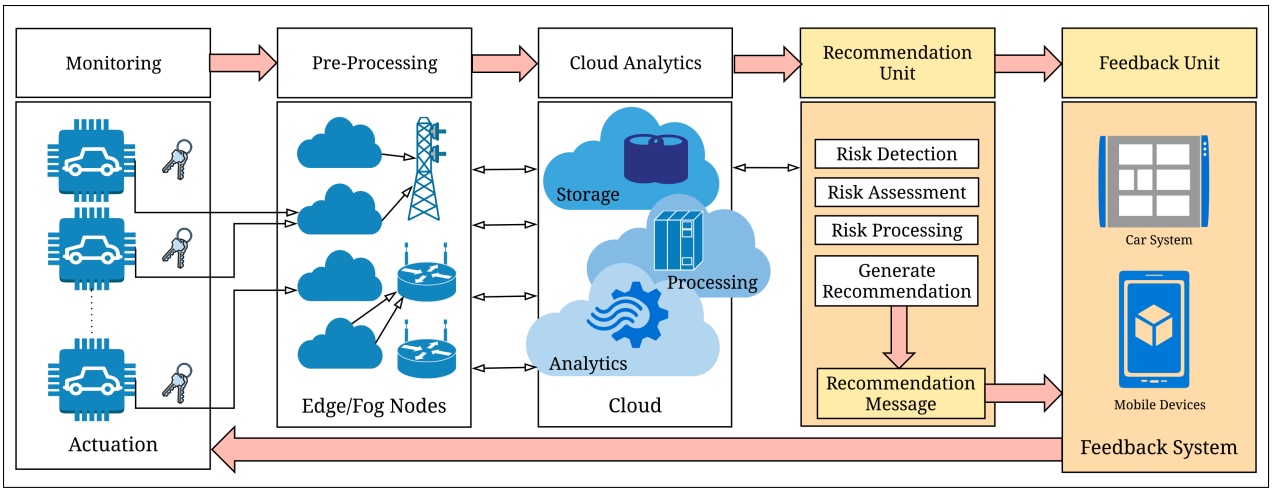


Figure 1.1: Components of Proactive FinTech Architecture

patterns [64]. Simulations to study different user interfaces showing rewards gained, driving behavior performance, and speed limits have been tested for comparison with non-UBI models [14]. Besides, the rational-economic model proposes that humans usually respond to cost minimization and rewards maximization [20].

According to those findings, we believe that presenting a proactive feedback model to lower risk levels can support drivers' behavior improvement over the long term and reduce road-associated risks. We propose a personality-based feedback system design that supports driving behavior-centric models and utilizes the use of sensor data for cyber-physical connections through IoT technologies.

In Figure 1.1, we show the overall architecture of the proactive feedback system. Through monitoring, IoT devices and actuators sense and collect driving behavior data, road conditions information. Then, the fog server receives the securely transmitted data and calculates the drive scores based on drivers personality characteristics. The cloud server acquires the data for storage and further analysis. Data analytics uses well-known machine learning techniques on the long-term historical data for assessing risk levels. The Recommendation Unit generates recommendations according to individual drivers risk levels. Recommendations are transmitted to the system's Feedback Unit to show drivers the suggested actions proactively.

We propose adriving behavior-centric model that includes the following components, and which will be described in further detail in the chapters of this dissertation:

1. Monitoring Unit: The car's installed telematics devices or mobile-phones are the primary data sources for behavior-centeric usage-based models. Data collection and analysis systems are installed in vehicles and set up (either by its built-in connection or via smartphones) to send the recorded data to the cloud database via mobile networks [62]. The collected data includes a wide range of variables that can be selected to be included in offering the Recommendation.

2. Event Manager: The Event Manager controls the monitoring and the feedback notification unit. When the system records a sufficient amount of data for processing, it sends them to the Fog Analytics Server for score calculation, personality analysis and recommendation policies decisions.

3. Proactive Feedback Unit: The Event Manager submits the calculated scores and Cloud analysis results to the Proactive Feedback component to define what measures should be followed to depress or keep acceptable scores and risk levels. IoT communication technologies can be used to communicate with the driver. The driver has the decision of whether to follow the given advice or not. Following recommendations results in earning rewards.

4. Fog Analytics Server: Since our system requires real-time data analysis to sustain a proactive system and the Fog Analytics technology was created to provide data analysis promptly, using the Fog as a component in our system provides a suitable environment for punctually offering services.

5. Scoring Engine: The Scoring Engine uses the recorded data to calculate scores [13]. The scores can be calculated based on single or multiple factors. For example, trip scores could use factors such as congestion rates, accidents rates, driven kilometers, road

types, time, and location information. Driving behavior scores could apply variables such as the number of times of hard accelerations, hard brakes, over-speeding, and aggressive lane changes. Our current scoring engine is based on the scoring model presented in [39]. Our system uses the speed score, brakes score, acceleration score, and swerve maneuver score. The scores are used to give an insight into the drivers' performance, and based on that, monetary rewards or penalties are given.

6. Cloud Computing Server: The Cloud Server receives the sensor collected data and the calculated scores from the Fog Server for storage and further analysis.

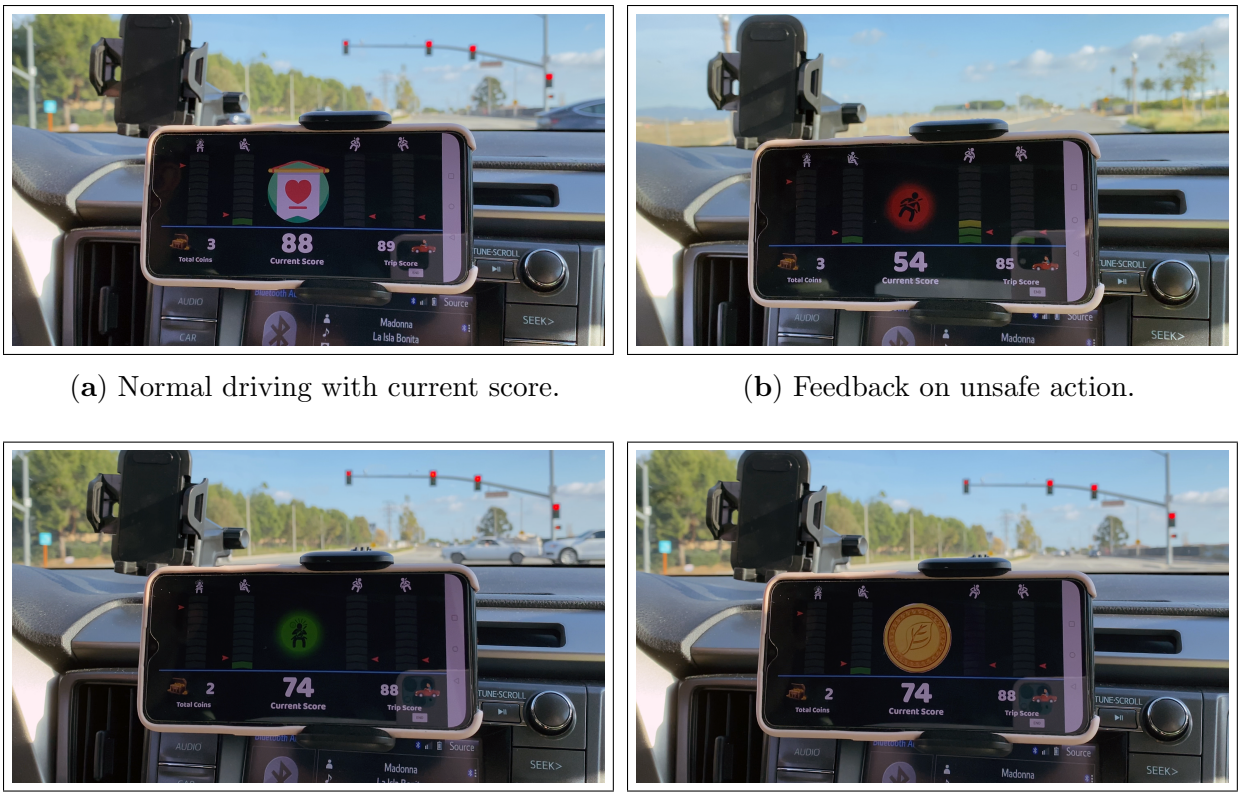
7. Risk Analysis Unit: Risk level information is available in the Cloud for risk analysis. The scores and other traditional and usage-based variables are used as inputs in the risk level formula to calculate the personalized risk levels for individual drivers.

1.5 AutoCoach Agent as a Proposed Solution

In this work, we propose AutoCoach, an IoT-based driving behavior management agent. AutoCoach is designed to improve drivers' behavior by understanding the differences between drivers to offer personalized feedback. AutoCoach classifies drivers into different personality groups based on their driving style. It then uses the classification to determine how to reward drivers and give valuable feedback as a means of persuasion.

Our agent software operates using the real-world collected and machine-learned personality models and a prototype feedback engine. This prototype has given us a useful platform to experiment with many machine-learning models, policy designs, and user experience studies. The experience of building, testing, and running IIoT in a real-world setting is essential for many future smart IoT trainer applications that aim to be user-specific and rewarding.

To our knowledge, our project is among the first to adopt machine learning capabilities to



(**a**) Normal driving with current score.

(**b**) Feedback on unsafe action.

(**c**) Behavior improvement notice.

(**d**) Reward coin issued.

Figure 1.2: AutoCoach feedback and rewards.

classify drivers into different risk groups to provide intelligent feedback supporting behavior adaptation. Our software architecture collects and analyzes real-time data between users' smartphones and a cloud-based server to update the user personality model and personal scores in real-time, allowing instant drivers' model updates. AutoCoach's feedback engine design includes a personalized, proactive feedback-triggering feature. AutoCoach understands the drivers' differences in this feedback model and identifies their needs for support and encouragement.

To test the success of AutoCoach, we conducted a pilot study allowing drivers to engage with AutoCoach through an informative GUI (Figure 1.2). We have used a within-subject design for this study, where all participants tested a personalized personality-based version and a non-personalized version of AutoCoach. We conducted a semi-structured interview right

after each participant completed the experiment. We analyzed qualitative and quantitative data to study several aspects from the participants' points of view, including timeliness, sensitivity to risky behavior, correctness, level of intelligence, and others. Based on the personality group design, the user study results have shown that 69.8% of test users report that the group-based version exhibits higher correctness in event detection, and 54.2% found that the group-based version is more timely in offering feedback. In terms of feedback sensitivity, 53.1% found that the group-based version is more sensitive to their driving events, 71.9% of participants found it is more accurate in identifying risk, and 81% found it more accurate in identifying safe driving. Participants also noticed that the group-based version is more interactive with their driving performance if they are in the higher risk groups, supporting our goal of more intense feedback for riskier drivers.

1.6 Summary and Dissertation Organization

In summary, this work presents AutoCoach, an Android-based mobile application that presents the personalization into machine intelligence for coaching drivers. It uses phone-embedded motion sensors, including accelerometers, gyroscopes, and magnetometers, to sense motion and identifies the events while driving. This information is then further analyzed and integrated with personality-based policies to offer more helpful feedback to drivers. Our contributions in this work are summarized as follows:

1. This project is one of the first works to improve IoT intelligence by classifying human users into different personality groups using machine learning models. Such intelligent personality grouping capability can be adopted in many smart IoT applications to offer more personalized service and more precise personal coaching.

2. We have built a working prototype that implements an intelligent real-time user feed-

back engine based on a user's past driving behavior and personality group association.The platform is used to propose and study new machine learning models and user policies, utilizing important user data in both time and space dimensions. The platform is very powerful to facilitate new AI algorithms and capabilities.

3. We have conducted an on-the-road user study involving 50 users in different cities to show the system's performance. Our user study has shown very positive responses from users. We have also conducted a repeated one-way ANOVA study to prove the driver's gradual behavior improvement over time.

4. We have improved our agent based on the results we obtained from the previous study to offer adjustments on the strategies assigned to drivers under different road types and conditions and conducted other studies to prove the success of the new version.

The rest of the dissertation is organized as follows: Chapter 2 shows some related work to our project. Chapter 3 discusses the basic system architecture of AutoCoach. In Chapter 4, we present details about the driving behavior detection and identification methods and models. Chapter 5 discuss details about the driving behavior risk analysis unit. In Chapter 6 we describe the personality group features, model design and analysis. Chapter 7 describes the rules and decisions models for feedback and scoring. The user interface for AutoCoach agent is discussed in Chapter 8. AutoCoach's performance evaluation through usability testing is reported in Chapter 9. Chapter 10 discusses the improvements made on AutoCoach and an assessment of it's performance through a user study. Conclusions and future works are given in Chapter 11.

Chapter 2

Background and Related Work

2.1 Usage-Based-Insurance Models

The advancements of IoT made it possible for automobile businesses and system developers to collect massive data that gives insight into drivers driving behavior. Sensors, like accelerometers, gyroscopes, magnetometers, GPS can teach us so much about how drivers behave. Similarly, On-board-diagnostics (OBD) devices communicate with the car's CAN-Bus to tell us even more about driving behavior when fused with IoT technologies.

Customer value is significantly improved when linking FinTech and InsurTech with IoT [47]. PwC [57] has investigated and shown that self-service and UBI models exhibit the highest business opportunities. Self-directed services provide simplicity, functionality, transparency, where the fulfillment of customer needs will be with ease and speed. In UBI, customers will be happy to have to pay only for how much they use. Many insurance service providers are now in the process of adapting to these business opportunities.

Traditional automobile insurance services rely on static data [70] collected from empirical

studies and statistics to define risk factors, such as age, gender, level of education, driving record, insurance records, and others [63]. However, one model does not fit all, and individuals' risk of loss is not equivalent. Traditional insurance strategies are inefficient because they do not consider the differences between individuals' driving behavior, trip context, and road situations [62]. Hence, many auto insurance providers are currently working on switching from traditional insurance business models to UBI models [52][3] to compete with the disruptive InsurTech startups [44].

Throughout the past decade, several UBI models were created [62]. Models started as simple as pay-per-mile models, which relied on offering premiums per number of distance units [38]. Nowadays, IoT-based behavior-centric models are gaining popularity [26][4], such as the Pay-How-You-Drive (PHYD) [11][45]. In those models, telematics devices are used to collect dynamic data about drivers' behavior and trip context [4][50][70]. This data is aggregated and analyzed either on the driver's mobile phone or the server-side of the service provider. Weighted scores, penalties, and drivers' risk of loss are calculated to be used with the premium rate formula to determine drivers insurance cost [63]. In PHYD, insured users pay a dynamic premium based on real-time data collected from telematics.

Most PHYD models provide analysis and feedback after events. Drivers may see their driving score information at the end of each trip [4][27][32]. Some models show driving behavior information for aggressive brakes, accelerations, swerve maneuvers, and over-speed in real-time on mobile phones [13], and others display alerts when specific driving behaviors exceed the acceptable limits [61]. Feedback can be presented in different models, such as scores, alerts, or monetary rewards.

However, we believe there remains a need to keep drivers aware of potential individual driving behavior improvements by offering personalized feedback about their specific driving styles. Proactive feedback also includes providing real-time knowledge about unexpected road situations or weather conditions. Under different contextual situations, the feedback

should be issued early enough to allow drivers to have the chance to respond and avoid falling into situations that could cause drivers to misbehave or get into high-risk situations.

2.2 Drivers Personality Characteristics

Research has shown that drivers carry different personalities according to [59]. Some drivers are considered angry drivers; others are reckless, anxious, or careful drivers. Anxious driving style: "has commonly been examined in studies on driver stress and reflects feelings of alertness and tension as well as ineffective engagement in relaxing activities during driving." [59] Angry and Hostile driving style: "refers to expressions of irritation, rage, and hostile attitudes and acts while driving, and reflects a tendency to act aggressively on the road, curse, blow the horn, or "flash" to other drivers." [59] Reckless and careless driving style: "reckless and careless driving style refers to deliberate violations of safe driving norms, and the seeking of sensations and thrill while driving. It characterizes persons who drive at high speeds, race in cars, pass other cars in no-passing zones, and drive while intoxicated, probably endangering themselves and others." [59] Careful and Patient driving style: "reflects a well-adjusted driving style that has received less attention in previous studies. This style refers to planning; attention, patience, politeness, and calmness while driving; and keeping the traffic rules." [59]

Drivers naturally combine those personalities but tend to bend more towards one or another, making it hard to specify a particular personality to each driver. Therefore, we rely on those personality definitions to extract the degree of risk-driving behavior patterns are showing. By this, we can monitor what happens during a drive in real-time and help drivers proactively respond to risky situations. Furthermore, understanding what represents each personality, we can learn each driver's personality to help support their different needs.

Researchers found that individual automobile drivers carry different personalities, and specific behaviors characterize each. Drivers' personalities have been summarized in four personalities: cautious, anxious, angry, and reckless drivers personalities [59]. Each of those personalities is described by showing some features or driving behavior patterns, which is discussed in detail in Section 6.

Many studies seek to classify drivers into different groups based on their driving styles using sensor readings. However, we do not find works that have clearly defined the different driver's personalities. For example, [28] has designed a system that classifies levels of hazardous driving , while [17] classifies driving behavior for safe/unsafe.

Angry drivers show more frequent and more intense aggressive behavior. Hard brakes and sudden acceleration reports higher G-force readings within a very short interval. On the other hand, reckless drivers are careless drivers. They have sudden, unpredictable behavior. They show sporadic, sudden brakes and accelerations within long intervals. Anxious drivers show much hesitation; hence, they are recognized by how many actions they take within short periods.

We use those personality characteristics to set the learning criteria for which behaviors should be considered risky or not. Eventually, this supports our goal of offering personalized feedback to drivers based on their habits and borders.

2.3 Driving Events Detection

Mobile phone built-in sensors, such as accelerometers, gyroscopes, and compasses, grant massive amounts of motion sensor data. This data is analyzed to obtain driving behavior information (e.g., [17, 37]). Some works incorporate data from CAN-Bus along with the mobile sensors data [30], Others only rely on the vehicle's electronic stability control system

Table 2.1: Accelerometer Data

Factor	Axis	G-force Range (g)
Hard Acceleration	g_x	0.3 - 0.4
Hard Brake	g_x	-0.6 - -0.7
Swerve Maneuver	$g_y\ left$	0.5 - 0.6
	$g_y\ right$	-0.5 - -0.6

to characterizes drivers based on driving style [41], and some depend on smartphones [42].

Driving behavior can be detected using in-vehicle accelerometer information [42]. Acceleration and brakes can be detected by using the lateral acceleration (x-axis), and the swerve maneuvers are detected by using the longitudinal acceleration (y-axis) readings [51]. Some thresholds have been defined in [15] to detect hard accelerations, hard brakes, hard swerve maneuvers (Table 2.1).

Let the dataset $A = \{r_1, r_2, r_3, \ldots, r_m\}$ be a set of continues data stream. and r_i is a record containing accelerometer data: $r_i = \{a_{xi}, a_{y_i}, sp_i\}$, where a_x is the lateral acceleration, a_y is the longitudinal acceleration, and sp is the driving speed. Let k be the size of the sampling window, then Simple Moving Average (SMA) is defined by:

$$SMA = \frac{a_x(i)^2 + a_x(i-1)^2 + \ldots + a_x(i-k-1)^2}{k} \tag{2.1}$$

Using SMA [30][67], we can find the numbers of hard accelerations, brakes, and swerve maneuvers.

When SMA for lateral acceleration a_x ranges between 0.3g and 0.4g, we count the event as hard acceleration, and when it is between -0.6g and -0.7g, we count the event like a hard brake. For Swerve Maneuver, when SMA for longitudinal acceleration a_y range is between 0.5g and 0.6g, it is counted as a left swerve maneuver and -0.5g to -0.6g for right maneuver.

Similar methodology is used to detect overspeeding events. Let n_{osp} be the number of events being in overspeed, then:

$$n_{Osp} = \sum_{sp_i \in A} \begin{cases} SMA_i > 120 & 1 \\ otherwise & 0 \end{cases} \tag{2.2}$$

When the average speed within the sampling window is greater than 120 km\h, then the event is counted as overspeeding.

2.4 Driving Behavior Feedback

There had been a discussion on when and how to offer drivers feedback, as presented by [12]. To our knowledge, many current UBI systems provide feedback passively. As in current models, latent feedback is provided at the end of the trip or day. Other schemes offer real-time feedback in the form of negative behavior counts displayed on a mobile phone application. These are some examples of some seen works. Some works use positive feedback by providing incentives that include reducing insurance premiums for the vehicle driver [23]. Others designed a system that detects driving behavior events using mobile-phone build-in sensors, where they score individual driving behavior events based on the intensity [29]. Some works allow users to access their driving behavior feedback and share it with other registered profiles [51]. Others score driver trips compared to other drivers using the trip's comfort level [69].

We observe that there is no insight given on how to improve and prevent certain behaviors from reoccurring. AutoCoach offers personalized feedback when drivers when hazardous actions are detected by understanding individuals' differences and needs.

Many researchers have been actively studying possible solutions for better estimating driving risks. Some tackled the event detection problems, and others went with driving behavior analysis, while some proposed feedback models. In [18], authors studied the detection of the events, where they compared the performance of the event detection of aggressive and safe events when using different ML algorithms on different mobile phones. They concluded that some combinations of algorithms and phone models perform better. Ref. [65] presented a novel driving behavior recognition system that can detect brakes, accelerations, turns, and lane change with high accuracy. Health Driving [37] is a mobile-based application that detects driving events and conditions when accelerometer axis values pass predefined thresholds. It assigns classes of safe/unsafe to detected events. Another paper proposes a strategy based on bag-of-words to model accelerometer information associated with aggressive driving maneuvers [7]. D_3 [10] is another mobile application that uses its built-in sensors to detect and classify driving behaviors, such as swerving, side-sliding, weaving, and sudden braking. Other research projects focus on the studies of the classification of driving behavior patterns described by the sequence of events [31]. Additional studies seek to classify drivers into different groups based on their driving styles using sensor readings. For example, [29] has designed a system that classifies hazardous driving levels, while [16] classifies driving behavior drivers to be either safe and unsafe. DrivingSense [42] identifies dangerous behaviors, including speeding and failing to control the car properly. SenseFleet [8] is another mobile-based application that identifies risky maneuvers, accelerating, and braking and provides a score reflecting the driver's overall score. All of the above provide a good foundation for our personality-based user model.

Some studies proposed ideas for scoring and providing feedback to drivers. The authors of [33] provide an application to monitor, analyze, and offer recommendations to drivers based on detected unsafe behaviors. Join Driving [69] is another smartphone-based driving behavior evaluation system. Join has two components: driving event detection and the evaluation part. It passes the detected events information to a scoring mechanism to quantitatively

evaluate the events and comfort levels in real-time. The identified behavior can be used to provide feedback and recommendations to drivers. Our system architecture has a similar structure but supports more sophisticated user models.

Behavior adaptation (BA) occurs in many different ways as people become more mindful of the risks associated with their driving [58]. While BA is possible, user experience (UX) design is the top way to BA. How people feel while using a product and how well it serves their purpose defines UX [1]. Factors like trust are one of the main components of reliable UX design. Trust leads to customer satisfaction, loyalty, and patience [35] conceiving a successful product. Our project aims to build a smart driver behavior management agent to enable BA in drivers.

Chapter 3

AutoCoach System Architecture

The accelerating growth of intelligent technologies is leading the way for smart innovations in diverse fields. That is making the appearance of a smart world more evident every day. We believe that intelligent IoT models can greatly enfold the idea of penalization by recognizing the characteristics of humans' responses in different environments and settings. This behavioral understanding allows us to investigate computer-based approaches for identifying particular personalities represented by habits, emotions, or lifestyle choices. By acknowledging human differences, we can provide more dependable, trustworthy services. We present AutoCoach as an innovative Intelligent personality-based solution for driving behavior management. In this chapter we discuss the details of AutoCoach's system architecture. Our current work uses built-in mobile phone sensors to detect driving behavior events and send them for analysis and feedback on-board and on-cloud. In Figure 3.1 we explain AutoCoach's design.

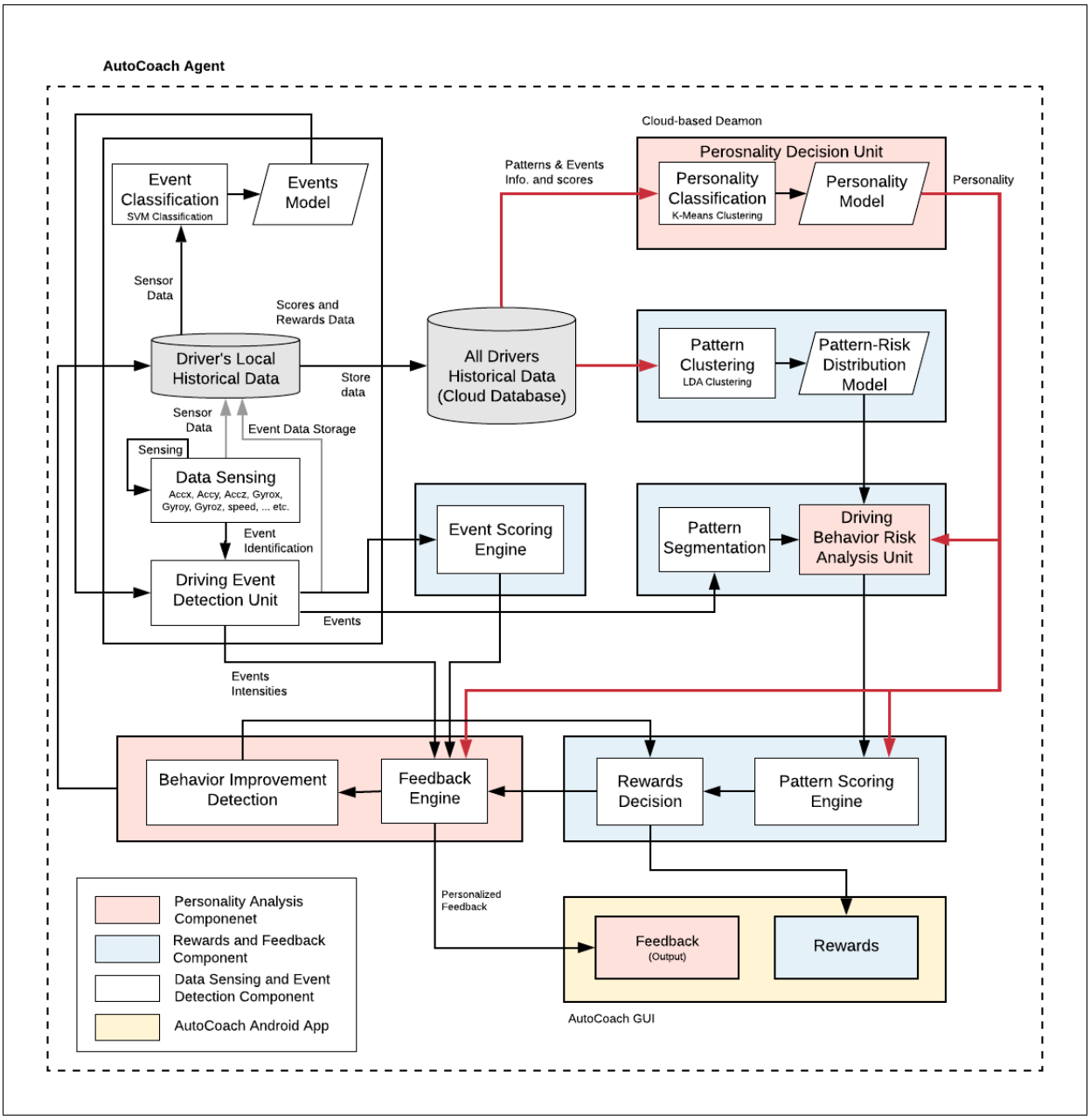


Figure 3.1: AutoCoach System Architecture

3.1 Data Sensing and Event Detection Component

Mobile phone built-in sensors, such as accelerometers, gyroscopes, and compasses, grant access to massive amounts of motion sensor data. This data is analyzed to obtain driving behavior information (e.g., [17, 37, 42]). Some works incorporate more data from CAN-

Bus [30], while others only rely on the on it to characterizes drivers based on driving style [41].

AutoCoach starts once the driver clicks the start button. The mobile phone continuously senses the driving data in the *Data Sensing Unit* as shown in Figure 3.1. Some of the data is collected from the phone's accelerometer, gyroscope, magnetometer, and GPS. The data is then sent to the *Driving Event Detection Unit (EDU)*.

EDU holds a model identifying driving behavior classifications (e.g., average acceleration, hard brake). The events are continuously detected, and the consecutive sets of events are segmented into windows. The window w is a fixed-time window of size n events. A window w represents what we call a driving behavior pattern. Those segmented patterns are passed to the *Driving Risk Analysis Unit (RAU)*. Further information about the EDU are discussed in Chapter 4.

3.2 Rewards and Feedback Component

RAU receives those sets of events. By using Latent Dirichlet topic modeling, we derive a model representing the pattern's risk levels. The model holds the pattern-risk probabilistic distribution for how much each pattern (i.e., a word in the LDA's original representation) holds risk (i.e., a topic in the traditional LDA representation). The different risk-clusters are ranging from safe to risky. The details of RAU are presented in Chapter 5.

RAU decides for each pattern its probabilistic risk level. When a pattern is detected in real-time, its risk probabilities are sent to the *Scoring Engine (SE)*. SE calculates the pattern's score and and event scores. The scores are used as threshold values to define drivers personality characteristics in order to make feedback and rewards decisions. Furthermore, the system scores the driving events (e.g., brakes) to identify whether the event is within

the driver's normal threshold. If not, then feedback might be required based on some rules and decisions made by the system. The scores are sent to the *Rewards and Penalty Decision Unit (RPD)* to determine feedback, rewards or penalties for the driver. The mechanism is presented in more detail in Chapter 7.

3.3 Personality Analysis Component

The *Personality Decision Unit (PDU)* defines the driver's personality risk group using K-Means clustering. Each group will carry a risk level ranking from safe to high-risk. When drivers use the system for the first time, they are placed into the safest risk group. Later, when the system learns more about the driver, he will be put into a more representing group. We discuss more details on the personality model and design in Chapter 6

The personality classifications are used to adjust the risk scores in the RAU. It is also used to improve the rewarding and penalization criteria in the RPD Unit. Further, based on the personality, the feedback offered by the *Feedback Engine* is defined. The system provides personalized feedback and recommendation using our innovative idea of *Memory Factor (MF)* (Details in Section 7.3).

The system architecture is implemented in AutoCoach, an android-based intelligent driving behavior management system, that applies the concepts of personalized personality-based feedback and rewards. Details on the design and implementation of the user interface and the GUI is discussed in Chapter 8.

The AutoCoach agent supports driving behavior management and behavior improvement over time. It may also support actuarial services in insurance contexts by providing them with driving behavior habits. It may also assist the car control system by sending signals to the brakes system and other parts to prevent accidents and reduce risks.

Chapter 4

Driving Event Detection

The *EDU* receives the sensor readings to detect and identify events. It analyzes the data, finds the type of the event, how aggressive it is, and then records the frequencies of events occurrences. We define four types of events: brakes, accelerations, right- or left-turns, and side-slides/lane-changes. Lane-changing, swerving, and driving on a curved road are all detected as side-slides in the current design. The risk level of each of those events depends on its aggressiveness. It can be either a safe, medium, or high-risk event. We use Support Vector Machine (SVM) supervised learning to analyze the collected raw data and generate a model to classify each event into one of the twelve classifications.

4.1 Sensor Calibration

Sensor calibration is an essential step for sensor data accuracy as the mobile phone will be moving and placed at different angles and in different positions. It becomes difficult to collect consistent data. For this purpose, we ask the drivers to set the mobile phone on a phone holder in a landscape view. Once the phone is placed, the drivers' car had to remain

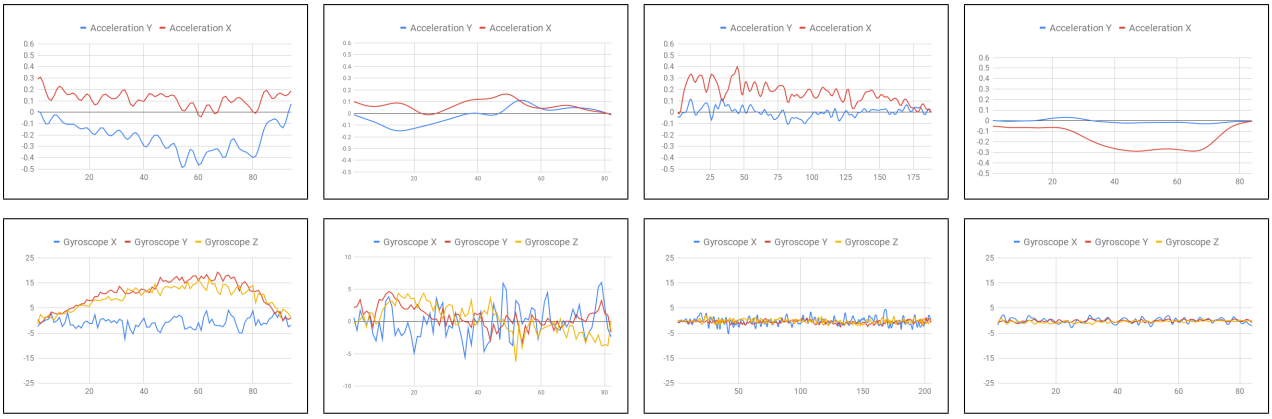
in zero motion before starting the application. Once the start button is clicked, then the sensors will be calibrated.

We use the automatic calibration method [46] to align the sensor's Cartesian frame of reference with the vehicle's Cartesian frame of reference. We generate a transformation matrix to convert the accelerometer reference frame axes to the car reference frame axes. The accelerometer is very sensitive; a running car can create much noise making the readings inaccurate. Therefore, we also use a Low Pass Butter-Worth filter with cutoff = 0.4 to filter out the noise as shown in the examples in Figure 4.1. The blue lines are the unfiltered signals. We can see many signal fluctuations due to noise. When smoothing out the signal (red line), we get a cleaner signal that we can use to determine the severity of the driving behavior event.

4.2 SVM Ground Truth Labeling

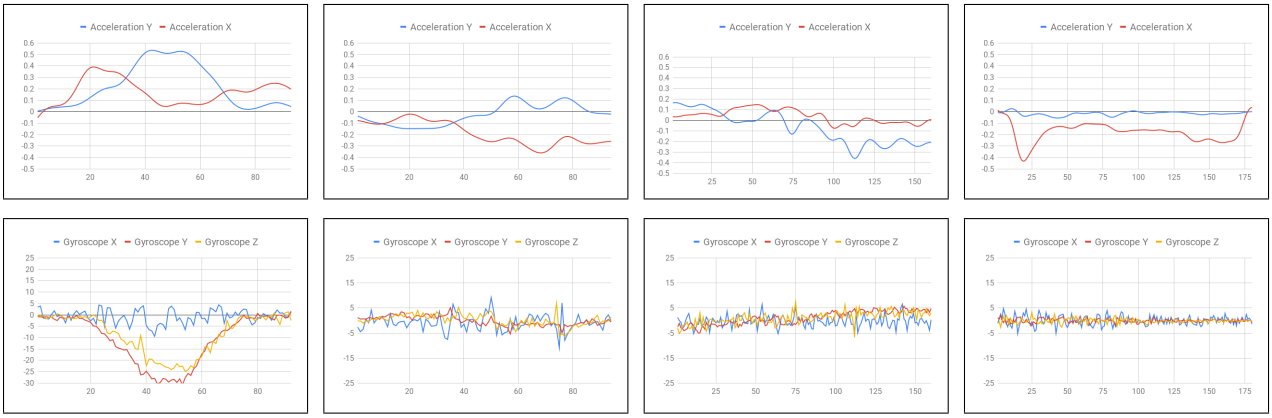
We collect ground truth data to define the event classes. We rely on human perception of force to determine the ground truth based on how much lateral and longitudinal force the driver or passenger senses for different events. The aggressiveness levels are labeled as normal-risk, low-risk, medium-risk, and high-risk levels.

As we progressed through the project, our data set kept growing, and we kept retraining the model to improve it. We collected the ground truth data from many trips on a Toyota Rav4 and a BMW M2 for our initial model. During the ride, we start recording a video by stating the date and time of the trip. Then, at the occurrence of events, the driver indicates the event type and his perception of the aggressiveness level. Once the trip has ended, the driver says the time of the trip. In the lab, the data labels recorded in the video are matched with the detected beginnings and ends of events. We defined the combination of event types and



(a) Safe turn. (b) Safe lane-change. (c) Safe acceleration. (d) Safe brake.

(e) Mid-risk turn. (f) Mid-risk lane-change. (g) Mid-risk acceleration. (h) Mid-risk brake.



(i) High-risk turn. (j) High-risk lane-change. (k) High-risk acceleration. (l) High-risk brake.

Figure 4.1: Events Risk Levels Examples

labeled it with a letter representation, as shown in Table 5.1.

4.3 Events Model Training

We follow the work in D3 [54] to build the detection mechanism as we found it to be of the best accuracy among other works. We have selected 12 features from their list of features and defined seven more features, as shown in Table 4.1, to help us identify different risk levels.

Table 4.1: SVM Features

	Features	**Description**
D3	Acc_x Std	Accelerometer X_{axis} Std
	Acc_y Std	Accelerometer Y_{axis} Std
	Acc_x Range	Accelerometer Max_x - Min_x
	Acc_y Range	Accelerometer Max_y - Min_y
	$Gyro_x$ Std	Std of Gyroscope X_{axis} values
	$Gyro_y$ Std	Std of Gyroscope Y_{axis} values
	Acc_x Mean	Mean of Accelerometer X_{axis} values
	Acc_y Mean	Mean of Accelerometer Y_{axis} values
	$Gyro_x$ Mean	Mean of Gyroscope X_{axis} values
	$Gyro_y$ Mean	Mean of Gyroscope Y_{axis} values
	Speed Mean	Mean of Speed
	Speed Std	Std of speed
AutoCoach	$Gyro_{Max}$	Max Gyroscope value
	Acc x_{Max}	Maximum Accelerometer X_{axis} value
	Acc y_{Max}	Maximum Accelerometer Y_{axis} value
	Acc x_{Min}	Minimum Accelerometer X_{axis} value
	Acc y_{Min}	Minimum Accelerometer Y_{axis} value
	Speed Difference	Difference between start & end speed
	Time	End of event - Start of event

4.4 Initial Event Detection Model

For the first version of the SVM model, We trained three different models, one for each of the cars we used in our study. The initial model was to classify the events to be either safe or risky using the 12 features we selected from the D3 paper and seven other features we added to satisfy our goal of detection, as shown in Table 4.1. Training data are described in Table 4.2. For this model, we collected a total of 186 events divided as 44 safe accelerations, 14 unsafe accelerations, 52 safe brakes, 17 unsafe brakes, 41 normal turns, seven unsafe turns, 12 safe swerves, and six unsafe swerves on the BMW car. We split the data into 80% training data and 20% testing data. The training accuracy for the 2-levels model was 100%, while the test accuracy was 92.1%.

Table 4.2: SVM Training for Two Risk Levels

Car Model	**Risk**	**Brakes**	**Acceleration**	**Turns**	**Side-slide**
BMW	Safe	51	42	41	11
	Risky	16	13	6	5
Accuracy	**Training**	100%		Test	92.1%

Then, we further improved our model to classify events into three risk levels. In Table 4.3, we labeled some trips taken on the Toyota Rav4 car. We had 56 brakes, 33 accelerations, 43 turns, and eight side-sliding of normal risk. We also recorded seven brakes, eight accelerations, four turns, and one side-sliding of medium risk. Moreover, there were six brakes, 13 accelerations, nine turns, and three side-slides of high risk. We tested the model and got a training accuracy of 96% and test accuracy of 76.9%.

We have also trained based on four risk levels using data for several trips on the Rav4 car. The training accuracy was 90.1%, and the test accuracy was only 61.5%. We noticed that the more risk levels we have, the lower the accuracy is. We realized that having small amounts of data is one of the biggest reasons for failing to get better results. Therefore, we used the

Table 4.3: SVM Training for Three Risk Levels

Car Model	**Risk**	**Brakes**	**Acceleration**	**Turns**	**Side-slide**
Rav4	Normal	56	33	43	8
	Medium	7	8	4	1
	High	6	13	9	3
Accuracy	**Training**	96%		Test	76.9%
BMW	Normal	89	108	59	12
	Medium	3	12	1	0
	High	14	17	6	5
Accuracy	**Training**	100%		Test	86.36%

oversampling method to balance the data on the Rav4 3-levels data and retrained the model. The training and test accuracy improved to 98% and 89%, respectively.

4.5 Generalized Detection Model

To expand our work for this study and allow many drivers to use our application, we create a generalized model that could work on different SUVs and Sedans. To train the model, we explored a series of ML algorithms. We found that SVM and KNN, which are applied to many areas, such as pattern recognition and regression, are the most suitable ML algorithms to classify signals. Since SVM and KNN are supervised algorithms, the input data requires labeling. We collected raw sensor data from ten cars of different models and sizes and under different road conditions (Table 4.4). Our team has manually labeled 1350 driving events into the twelve classifications. The data was collected and labeled at several stages throughout the phases of our work.

Then, we further investigated the addition and removal of several features displayed in Table 4.6. We tested non-weighted features 1–13 and 18–23, which resulted in 83.6% for SVM and 72.62% for KNN. We identified that many of the false classifications were falling

Table 4.4: SVM Training Data for the Generalized Model.

Risk	**Brakes**	**Acceleration**	**Turns**	**Lane-changes**
Safe	322	278	183	77
Medium	113	50	54	35
High	72	72	57	43
Sum.	400	507	294	155

in the minority classes. Therefore, we examined using weighted SVM [25] to solve the problem of unbalanced data. The accuracy of SVM successfully improved to 85.2%. We further investigated new features that could help get even better results. We found that the addition of Feature 14 significantly increased the classification accuracy to 93.4% (SVM) and 81.4% (KNN). This feature uses the sensor data to detect when an event occurs when the signal exceeds a certain threshold. When the accelerometer x-axis signal is responsible for triggering the detection, the event is likely to be a turn or a lane change; otherwise, it is an acceleration or a brake.

Table 4.5: SVM and KNN accuracy results.

Features	**SVM Accuracy**	**KNN Accuracy**
Weighted Features 1–13, 18–23	85.20%	
Non-weighted Features 1–13, 18–23	83.60%	72.62%
Features 1–14,16–23	85.20%	78.26%
Features 1–14,18–23	93.40%	81.40%
Features 1–17	93.40%	83.70%
Features 1–17 with Relief weight [34]	86.80%	79.10%
PCA-3D, Features 1–17	75.40%	82.00%
PCA-4D, Features 1–17	80.30%	85.32%

Table 4.5 shows a summary of our event detection experimental results. We concluded that SVM performed best in identifying twelve driving events with the features 1–17, and we got similar results when using the 19 features 1–14 and 16–23 with an accuracy of 93.4%. We used the model with 17 features because it is of lower dimensionality.

Table 4.6: SVM features.

	No.	Feature	Description
v1.0	1	Max $accel_x$	The maximum value of the accelerometer's x-axis
	2	Min $accel_x$	The minimum value of the accelerometer's x-axis
	3	Max $gyro_x$	The maximum value of gyroscope's x-axis
	4	Speed difference	The subtraction of the event's end speed and start speed
	5	Time duration	The time duration of the whole event
v1.1	6	Range of $accel_x$	Subtraction of max. and min. values of the accelerometer's x-axis
	7	Range of $accel_y$	Subtraction of max. and min. values of the accelerometer's y-axis
	8	Std $accel_x$	The standard deviation of the accelerometer's x-axis
	9	Std $accel_y$	The standard deviation of the accelerometer's y-axis
	10	Mean $accel_x$	The mean value of the accelerometer's x-axis
	11	Mean $accel_y$	The mean value of the accelerometer's y-axis
	12	Mean $gyro_x$	The mean value of the gyroscope's x-axis
	13	Speed mean	The mean value of speed within the event window
v2.0	14	Axis direction	States whether it is an event occurring on the x-axis or the y-axis
	15	Max $accel_y$	The maximum absolute value of the accelerometer's y-axis
	16	$StartEndAccx$	The sum of the start and end values of the accelerometer's x-axis
	17	$StartEndAccy$	The sum of the start and end values of the accelerometer's y-axis
v1.1 excluded	18	Std $gyro_x$	The standard deviation of the gyroscope's x-axis
	19	Std $gyro_y$	The standard deviation of the gyroscope's x-axis
	20	Mean $gyro_y$	The mean value of the gyroscope's y-axis
	21	Min $accel_y$	The minimum value of accelerometer's y-axis
	22	speed Std	The standard deviation of the speed
	23	Max $accel_y$	The maximum value of the accelerometer's y-axis

Chapter 5

Driving Behavior Pattern Analysis

In our model, a set of events happening within a fixed time window defines a *driving behavior pattern*. Some of those patterns frequently occur through a trip driven by people with particular habits. For instance, one driver may often have stronger brakes, accelerations, and lane changes within short periods. Another driver who drives carefully is more likely to drive carefully every time. It does not mean that a riskier driver will never drive safely, or a safe driver will never drive risky. Nevertheless, we can see similar patterns from different people.

We can use this information to understand and to offer the drivers feedback. In this model, we look for those patterns with similar features and group them.

Our study applies Latent Dirichlet Allocation (LDA) topic modeling to classify the patterns into their matched risk groups the same way words are matched with topics. LDA is an unsupervised machine learning algorithm that uses probabilistic modeling to output two models: the word-topic and document-topic models. In traditional LDA, a set of documents defines the corpus.

LDA's word-topic model defines how much a word is related to a topic, and the document-topic model defines how much a topic represents a document. Because LDA is a text-based model, we have transformed our input patterns into a form of text. The events and intensities model defined in Table 5.1 represents the letters of the words. The combination of letters becomes the words of a document, and in our case, we call it a *pattern*. Our topics are the risk groups.

5.1 Driving Behavior Patterns Definition

Table 5.1: LDA Driving Behavior Letters

Event	**Normal**	**Medium-Risk**	**High-Risk**
Brake	a	b	c
Acceleration	h	i	j
Turn	o	p	q
Lane-change	v	w	x

We have 12 classes of events, each represented by the combination of an event type and its associated risk level (e.g., brake: low risk, turn: high risk). In our definition, a pattern may contain a series of events or a "no events" pattern happening within a fixed-time window. A trip is a combination of many patterns happening within it. The input to the LDA model is many trips, and each trip is input, such as shown in the following example:

$ahhva, aav, oavov, va, vaav, ho, o, h, vooi, hxvvvq, oa, oha, va, oovoohv, ohh, vo,$
$hhhva, h, av, hoh, hvav, c, vaaaaa, vh, aahaa, hqh, ah, vvaav, ooo, vvovv, hv, hv,$
$oovhvo, hh, hhhooaa, vaaao, vhv, oho, av, avhv, ovv, vooo, oao, o, hh, av, joha, hahoo,$
$vho, vav, oh, hvh, oo, hvaho, ohv, av, vhaov, va, ooh, hho, va, o, hahv, avaaohh, oooav,$
$hoa, v, oh, oahah, h, ahvh, haaho, avh, va, avvha, ao, h, va, vo, ahhv, ooavva, aa,$

This trip was on a local road with lots of turns, stop-lights, and intersections. It shows many events within each of its patterns. Each pattern happens within a window of 20 seconds. As shown in the example in Figure 5.1, if we look at pattern *oavov*, it means that within a window, the driver has performed: a normal turn, followed by a brake, normal lane-change, normal brake, and another normal lane-change. This pattern can tell us that this driving behavior is safe and that the agent will not worry about it.

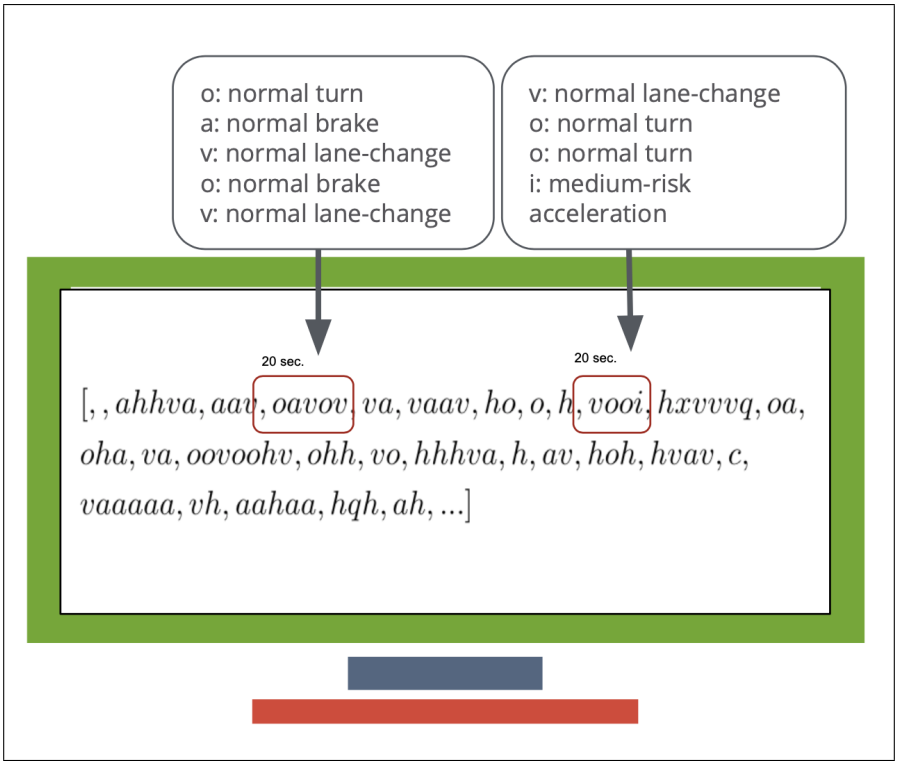


Figure 5.1: Trip Example

5.2 LDA Patterns Model

In order to train the LDA model, we needed at least a few hundred trips datasets to represent different trips. We collected data from actual trips driven with different driving styles; however, it was not easy to collect many datasets. Therefore, we used 200 simulated trips created concerning the actual trips we have taken. The data is used to classify the patterns

into four risk groups, where each group has similarities in driving behavior. For example, aggressive drivers tend to have more aggressiveness in their driving behavior, and thus, those aggressive patterns tend to appear together. Events of specific intensities highly coexist in some patterns, and those patterns coexist in the same class. Figure 5.2 shows the final output of the trained LDA model.

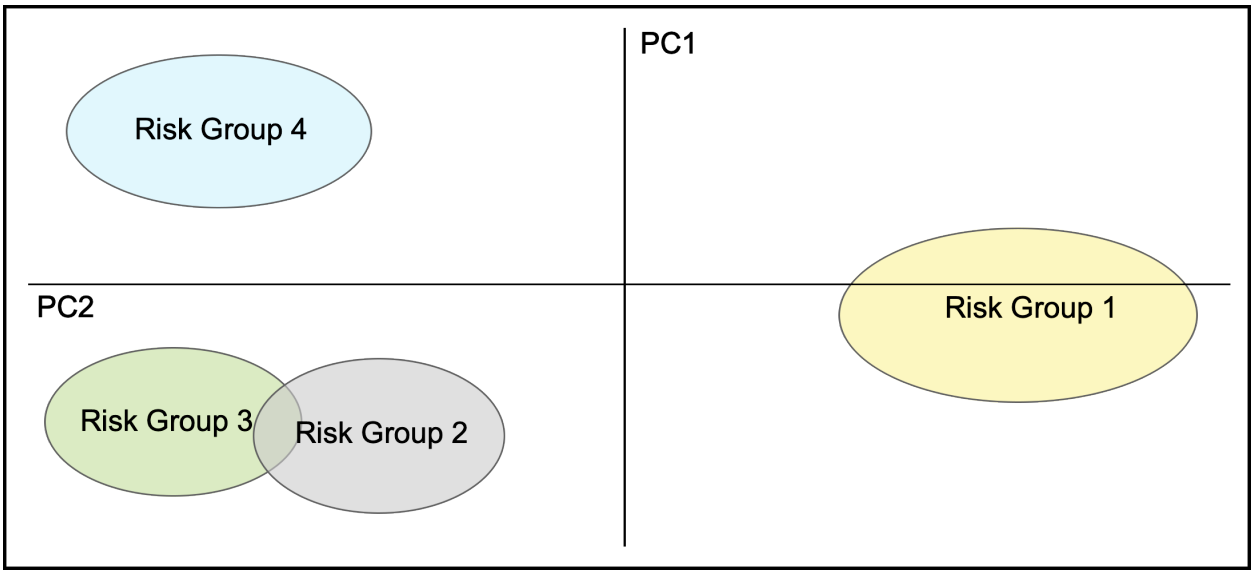


Figure 5.2: Patterns Distribution

5.3 Patterns Distributions

For each pattern, the LDA gives the probabilities for belonging to each of the four risk levels. For example, the system detected the pattern "*hcpah*" and sent it to the RAU; the output is:

$$Result = (0, 0.1254097), (1, 0.6228426), (2, 0.1262365), (3, 0.1255111) \tag{5.1}$$

where $(1, 0.6228426)$ belongs to "1: Low-Risk Group" with probability 0.62284267. We conclude that the pattern has a higher probability of posing "Low Risk" with a probability

of 0.62.

5.3.1 LDA Patterns Model Evaluation

Looking at the individual events risk classifications shown in Table 5.1, we know that if there are multiple events from the same risk group within a pattern, then this pattern fully represents that specific risk-group. Based on this concept, we selected the top 30 most relevant patterns to be judged by a human to tell whether each one of those 30 patterns represents the same risk group or not. Table 5.2 shows a binary decision whether a pattern perfectly represents Risk Group 3 or not.

Table 5.2: Risk Group 3 (Cluster 2) Evaluation

P	**y/n**	**P**	**y/n**	**P**	**y/n**	**P**	**y/n**	**P**	**y/n**
q	y	jc	y	qq	y	qc	y	qx	y
c	y	w	n	cj	y	qqj	y	jqj	y
x	y	xj	y	cc	y	qj	y	xcq	y
j	y	i	n	xx	y	xc	y	b	n
cq	y	jq	y	jj	y	cx	y	jcj	y
xq	y	p	n	jx	y	qqq	y	qxx	y

We found 29 relevant patterns out of 30 classified patterns in Cluster 1 (Risk Group 1), 26 out of 30 in Cluster 2 (Risk Group 3), and 45 out of 60 in Clusters 3 and 4 (Risk Group 2). Then, we applied Recall, Precision, and F-score measures to the top 30 most relevant results to each of the 3 risk groups. Results are shown in Table 5.3

Precision requires looking at the false negatives, which could be patterns that strongly represent a specific cluster but are represented falsely in other clusters. In Clusters 2, we found false positives: "i," "p," and "b," which strongly represent Clusters 3 and 4, but they are also true positives in Clusters 4 and 3. We also found that in Cluster 4, there exist a pattern "vba," which partially represents Cluster 1. However, as mentioned previously, we exclude those patterns with partial representation. Therefore, by looking at all 120 patterns, we

found 29 relevant patterns to Cluster 1, 27 are relevant to Cluster 2, and 45 are relevant to Cluster 3 and 4. Results in Table 5.3 show reasonable results for the pattern classification.

Table 5.3: LDA Results Evaluation

Risk Group	**Recall**	**Precision**	**F-Score**
R_1	0.966	1	0.9827
R_2	0.733	1	0.8459
R_3	0.766	0.963	0.8532
R_4	0.866	0.963	0.9119

We use 200 simulated trips to train the LDA model to classify the patterns into four risk groups. The simulated trips were created concerning the actual trips we have taken. When training the LDA model, the output has given out four clusters, as shown in Figure 5.2.

Our system must match the patterns detected in real-time with patterns in the patterns dictionary to find their risk level. Not all patterns could exist in the trained dictionary; therefore, the closest match represents the pattern risk level. We use an Edit Distance pattern matching algorithm to find the closest pattern. When a pattern is detected and is not found in the dictionary to directly find the risk level, Levenshtein Distance Algorithm, a pattern matching algorithm, is used to match the patterns to the closest available pattern in the model and use that closest pattern to represent the detected pattern.

5.3.2 Pattern Analysis Unit

The pattern analysis unit is responsible for assessing the behavior risk. Each pattern is represented by letters, where a letter represents a combination of event type and risk level (Table 5.1). Our model uses 12 letters and a pattern length ranging from 1 to approximately ten letters, creating a great number of possibilities. An example of a trip pattern would look like: [a, bah, hhbp, aoaaib, a, bh, aihb, bxaah...] , where a pattern is a window of 40 s, and the many patterns represent a trip.

Using this data, we would like to create a cluster model based on the similarities in risk between the patterns to group them. The model should create a well skewed scored data set with high variance to represent many possible combinations of events scores. We assume that the more high-risk events present in a pattern, the higher the risk is.

In order to find relationships between patterns, we considered several semi-supervised and unsupervised algorithms. Because unsupervised learning draws the inference from a dataset without labels, it could work for our dataset of unknown similarities between patterns. On the other hand, semi-supervised learning allows introducing a hint of knowledge about the dataset, which can be even more meaningful. We believe that, for example, the pattern "a" cannot be together with "b", and either cannot be with "c" as each belongs to a different risk group. However, when it comes to combinations of patterns, decisions become more challenging, and this is when machine-learning can figure out such complicated decisions.

5.4 Further Improvements on Patterns Model

We explored K-means and mean-shift from the list of the unsupervised clustering algorithms. We first converted the collected patterns to a fixed-length vector as the input data should be of equal length. The vectors include 12 features concerning the number of safe, medium-risk, and high-risk event types. We trained models using K-means and mean-shift. In K-means, we strictly assign numbers of clusters, while mean-shift decides the number of clusters the output will have. We found it could not show meaningful relevance for within-clusters data points when we looked through the results obtained. We found patterns such as "a" closest to "x", for example, which does not imply our purpose. As when converting the pattern into a vector, the sense of sequence and risk levels were gone.

Using a semi-supervised algorithm for such data format is more meaningful as we can, as

humans, provide insightful information about the degrees of risk patterns represent. We used the same features presented in K-Means to upgrade it to the COP K-means algorithm. COP K-means allows semi-supervision by adding must-link and cannot-link pairs of data, which helps direct the algorithm to represent the clusters based on initial decisions. Results from COP K-means were not successful either.

Therefore, we thought of moving back to our initial probabilistic LDA model and extended it to configure a Guided-LDA model to create a semi-supervised version of the LDA. LDA is a probabilistic model that views driving behavior patterns as words assigned to topics. The topics are viewed as groups of patterns with similar risk levels. Each pattern is given a combination of probabilistic values belonging to four defined risk groups.

Table 5.4: Semi-supervised algorithm statistics.

Algorithm	**Mean**	**Median**	**Std**	**Max**	**Min**	**Variance**	**Kurtosis**	**Skewness**
COP-K	83.202	89.86	19.892	99.96	40.12	395.693	0.286	−1.206
LDA	63.956	58.19	22.795	97.04	28.39	519.630	−1.091	−0.006
Guided LDA	61.008	59.405	12.780	96.63	37.49	163.330	2.623	1.145

On the contrary, in Guided-LDA, we set seeds for each cluster and allow the model to revolve around them. A seed is a pattern we believe should belong to a specific topic. The probabilities derived from the LDA model are used to generate scores for the detected patterns to determine the risk level the driver is showing at that moment. We scored the clustered patterns using the scoring engine presented in [43] where higher risk probabilities are multiplied by lower weights, while lower-risk probabilities are multiplied by higher weights. Statistical comparison was conducted on randomly selected sample data and is shown in Table 5.4. Results show that LDA and Guided-LDA are closer means to the data medians than COP K-means. LDA showed the highest standard deviation and variance (Std = 22.79 and variance = 519.62) and most asymmetric data based on skewness measures (skewness = −0.0056). Furthermore, the minimum and maximum values are widely separated in LDA (minimum =

28.39 and maximum = 97.04). We concluded that LDA results are most significant for the patterns analysis model and for scoring the behavior patterns.

Chapter 6

Personality-Based Driver Model

6.1 K-Means Clustering for Personality Groups

It is known that from different perspectives, groups of people may share similar habits and behaviors. In the same way, some drivers have similar driving habits. Unsupervised ML algorithms can help us identify the similarities of risks shown by drivers to classify them into multiple ranked risk groups. Such ranking allows us to set practical feedback methods based on how risky a driver is. We have explored the possible algorithms and found that the best-known algorithms for such a problem are K-means clustering. Therefore, we use K-Means to model different personalities groups. K-means is an unsupervised learning algorithm used to partition n observations into k clusters where observation belongs to the cluster with the nearest mean. As mentioned previously, drivers hold a combination of all those personalities, and therefore, cannot be given a specific title to a group. K-Means allows us to discover rules that describe large portions of driver's behaviors and habits. Thus, we can create personality groups based on the features we assume that represent specific behaviors.

Personality features can include information about habits, behaviors and emotional aspects.

Using events and risk levels detected in the Event Detection Unit (Section 4), we can define habit-related features, while patterns discussed in Section 5 can define behavior-related features. Emotionally, as defined by [59], authors propose that driving style personalities can be divided into four main types: reckless and careless, anxious, angry and hostile, patient and careful. All drivers have a combination of those emotion-induced personalities, but they tend to show one more than the others. We follow all those habitual, behavioral, and emotional characteristics to define features for classifying drivers into different personality groups.

In order to classify drivers into different clusters, we need to collect large amounts of data. The data includes events types and risk levels, patterns, scores, and time duration of patterns and events, and emotional-induced patterns. Our initial study collected about 300 minutes of actual driving under different personality characteristics. We created from those trips 400 simulated driver profiles, where each record contains $100 - 200$ patterns, and each pattern is a window of 40 seconds. To test if the data is suitable for representing several groups, we check the similarities and differences to make sure many clusters are created. We used the heat map for the euclidean distances to confirm present differences. Figure 6.1 calculates the Euclidean Distance between the data records and illustrates the similarities differences, where red is for significant differences and blue is for somewhat similar data records. We found from the results that the data we have has significant amount of similarities and differences, making it a well-balanced data set to create multiple clusters.

6.1.1 Personality Group Feature Selection

In the first model we created for the system prototype, we defined eight features to represent the characteristics of the drivers and used the previously created data for training the model. Suppose a driver has historical event sequence $E = \{e_1, e_2, ...e_n\}$, which are divided into

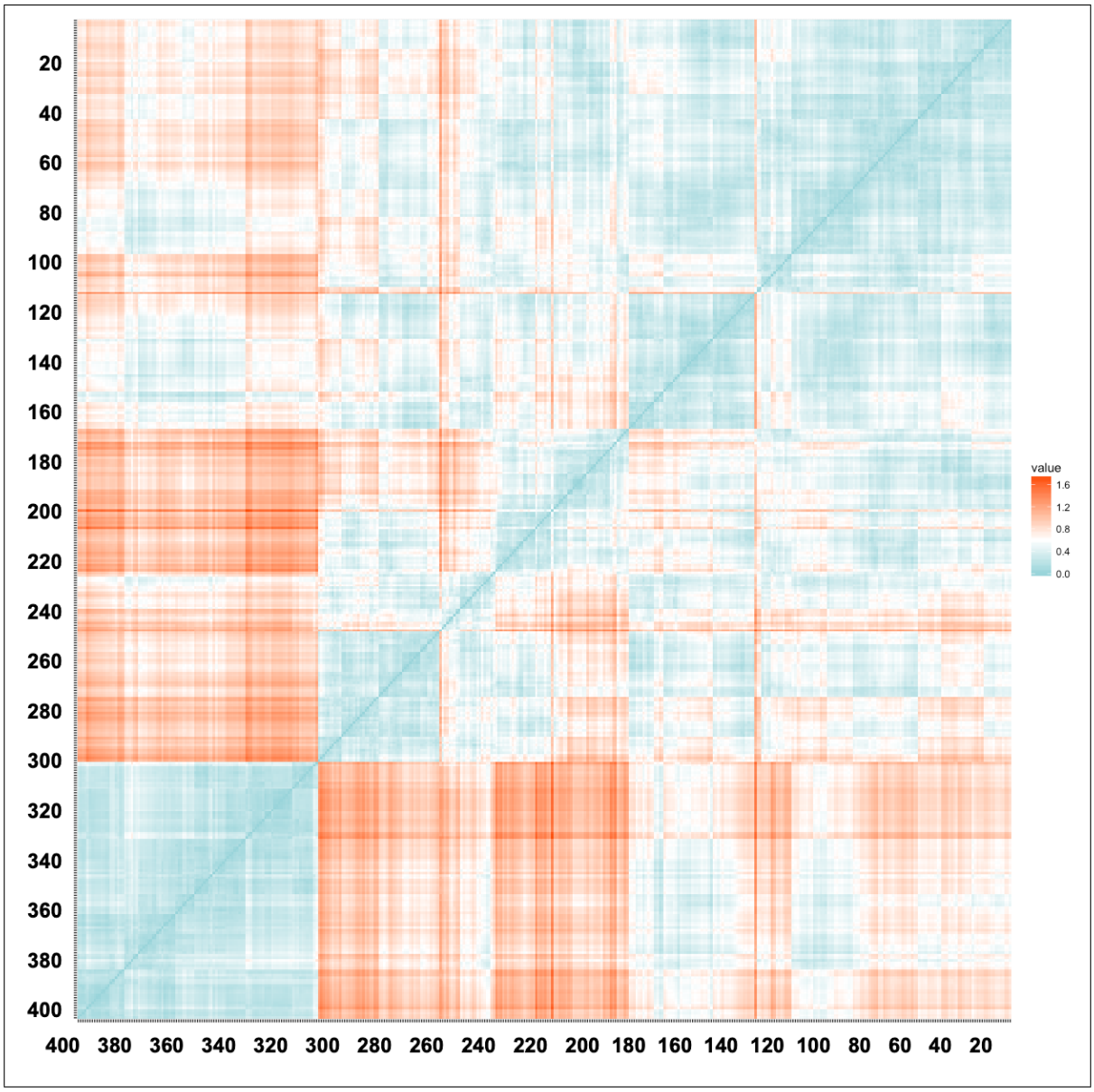


Figure 6.1: Euclidean Distance

window patterns $C = \{c_1, c_2, ...c_m\}$ with window scores $S = \{s_1, s_2, ...s_m\}$. We use the event-level classifications from the EDU to decide which event is safe-, medium-, or high-

risk. Each event $e_i \in E$ is assigned a value:

$$e_i = \begin{cases} 1 & \text{safe event} \\ 2 & \text{medium risk} \\ 3 & \text{high risk} \end{cases} \tag{6.1}$$

We use $\lambda = 75$ as a threshold to consider a window pattern score s_i as a high score. Using Iverson Brackets Formula, [P] is given the value of 1 or 0 depending on proposition P:

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true;} \\ 0 & \text{otherwise;} \end{cases} \tag{6.2}$$

The features for driver personality are defined as follows:

1. Percentage of Safe Events (E_s): Careful drivers show more safe events than other types of drivers. The percentage of safe events E_s for a driver with a history E of n events is given by:

$$E_s(E) = \frac{\left(\sum_{i=1}^{n} [e_i = 1]\right)}{n} \times 100 \tag{6.3}$$

2. Medium-Risk Events Percentage (E_m): Reckless and anxious drivers show worse than normal patterns but better than completely aggressive patterns. We can get the medium-risk events percentage E_m by:

$$E_m(E) = \frac{\left(\sum_{i=1}^{n} [e_i = 2]\right)}{n} \times 100 \tag{6.4}$$

3. High-Risk Events Percentage (E_h):

$$E_h(E) = \frac{\left(\sum_{i=1}^{n} [e_i = 3]\right)}{n} \times 100 \tag{6.5}$$

4. High Score Patterns Percentage (S_h): This feature detects safe driving behavior since safe drivers tend to have a high percentage of safe driving patterns. We can get the High Score Pattern Percentage by:

$$S_h(S) = \frac{\left(\sum_{j=1}^{m} [s_j \geq \lambda]\right)}{m} \times 100 \tag{6.6}$$

5. Low Score Percentage (S_l):

$$S_l(S) = \frac{\left(\sum_{j=1}^{m} [s_j < 75]\right)}{m} \times 100 \tag{6.7}$$

6. Aggressive Behavior Density (D_a): As shown in Figure 6.2, for angry drivers, we may observe a cluster of consecutive risky events in a fixed window. The *density* of aggressive behavior can be decided by the length of time a driver has a certain type of behavior repeatedly. We can check the duration l of an aggressive cluster from those consecutive events with a high risk of longer than t_a. In addition, we check if such clusters make the window to have a low score ($s_{c_k} < \lambda$).

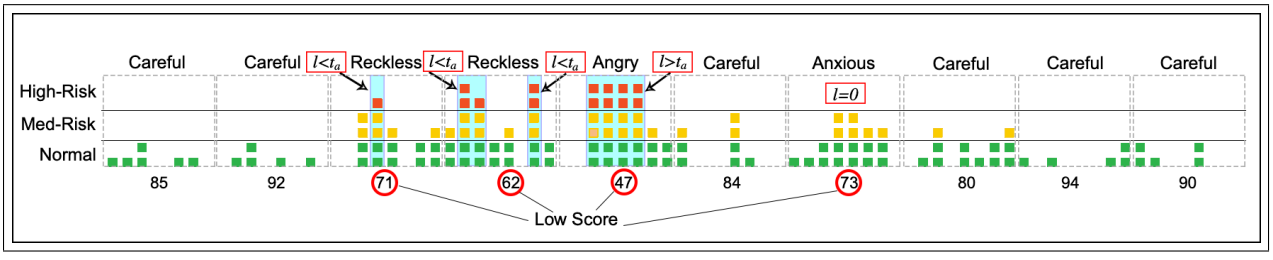


Figure 6.2: Personality Analysis Example

Then the density of aggressive behavior D_a is:

$$D_a(C) = \frac{\Big(\sum_{k=1}^{m}[s_{c_k} < \lambda \text{ and } t_a < l_{c_k}]\Big)}{m} \times 100 \tag{6.8}$$

7. Reckless Behavior Density (D_r): We observe reckless drivers have lower density risky events. Reckless driving behaviors show shorter and sparse aggressive patterns and are determined by checking if $t_a < l$.

$$D_r(C) = \frac{\Big(\sum_{k=1}^{m}[t_a > l_{c_k} \text{ and } s_{c_k} < \lambda]\Big)}{m} \times 100 \tag{6.9}$$

8. Anxious Behavior Density (D_x): Anxious drivers are those drivers who show hesitation and have low scores due to the high number of occurring normal events within a window:

$$D_x(C) = \frac{\Big(\sum_{k=1}^{m}[l_{c_k} = 0 \text{ and } s_{c_k} < \lambda]\Big)}{m} \times 100 \tag{6.10}$$

The features are summarized to which personality classifications they mainly target in Table 6.1. Each of those features generally tells us something about drivers' aggressiveness. The density of risky events tells us how long the driver remains in higher aggressiveness patterns periods, giving us insight about angry drivers. While, high and medium risk events frequency represent how often one would go into reckless manners, which in return is telling us the driver could be careless. Those drivers who would be doing all sorts of risky and non-risky behaviors might be anxious.

Table 6.1: K-Means Features and Personality Representation

No.	**Feature**	**Personality Characteristics**
1	Percentage of Safe Events (E_s)	Careful
2	Medium-Risk Events Percentage (E_m)	Reckless and Anxious
3	High-Risk Events Percentage (E_h)	Angry and Reckless
4	High Score Patterns Percentage (S_h)	Careful
5	Low Score Percentage (S_l)	Anxious, Reckless and Angry
6	Aggressive Behavior Density (D_a)	Angry and Reckless
7	Reckless Behavior Density (D_r)	Reckless
8	Anxious Behavior Density (D_x)	Anxious

6.1.2 Personality Groups Clusters Model

In order to find that these are the optimal clusters for our data, we have evaluated the K-Means clustering results for different k values using three statistical measures: Average Silhouette, Gap Statistics, and Total-Within Sum of Squares (TWSS). We have run the clustering for $k = 2$ up to $k = 10$ to find the optimal number of clusters. The final results of the clustering we have settled on are shown in Figure 6.3.

In the Average Silhouette, we try to find the cluster where an elbow is created, and we can clearly see in Figure 6.4 that at 4 clusters, this is the optimal number of clusters based on this method. However, for Gap Statistics, the optimal number of clusters is at the point where the curve stops rising, and we can see from the results in Figure 6.5 that 5 clusters are the optimal number of clusters. Based on the TWSS results shown in Figure 6.6, the optimal number of clusters is 3. The results from those methods are not exact, and we can go a little lower or higher around the results ranges. Therefore, we take the average of the three results and decide that 4 is our optimal number of clusters (Figure 6.7).

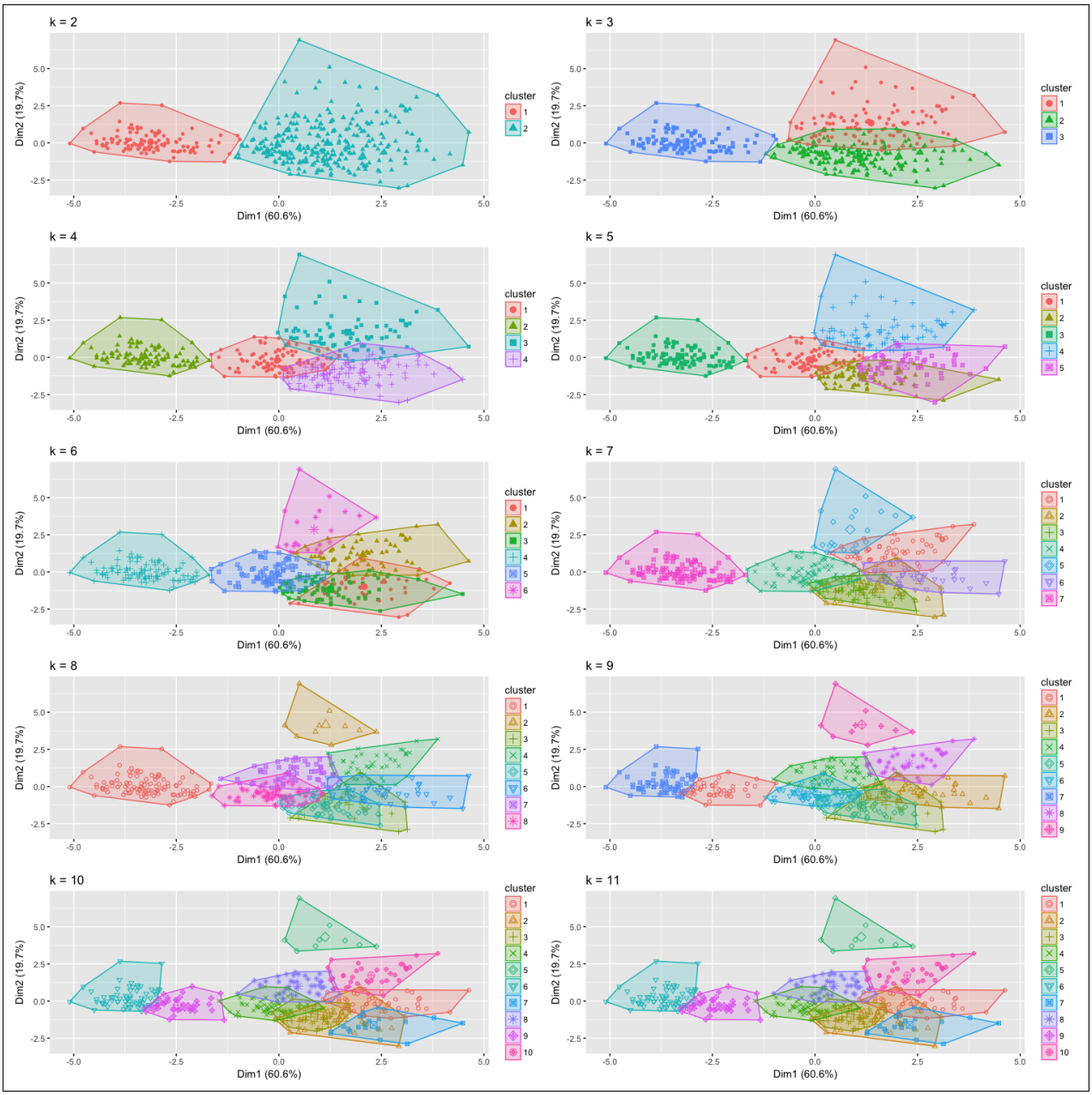


Figure 6.3: Clusters k=2 to k=11

6.2 Improved Personality-Based Driver Model

Later in the study, after we have completed setting up the whole system, we had to initiate a user study to test the success of our intelligent agent. Therefore, the system had to be personalized for different actual groups of people, and in order to get that, we had to collect

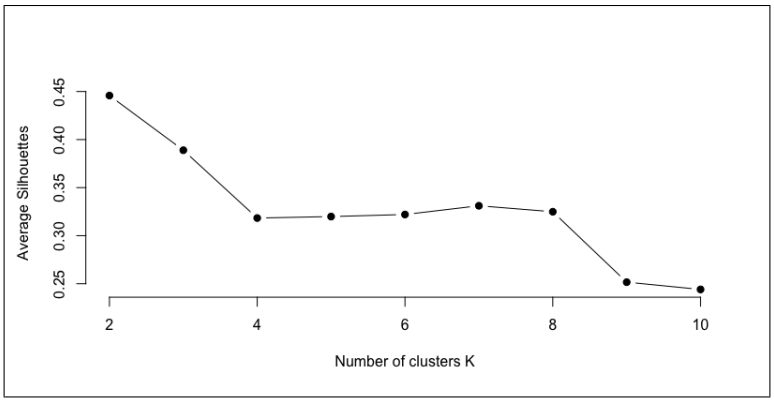


Figure 6.4: Average Silhouette

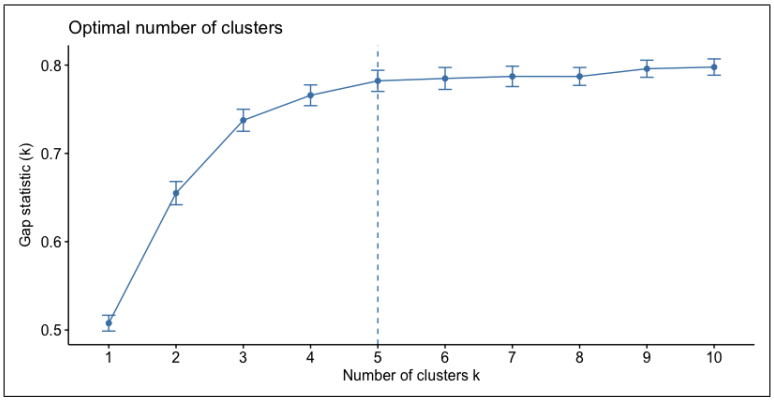


Figure 6.5: Gap Statistics

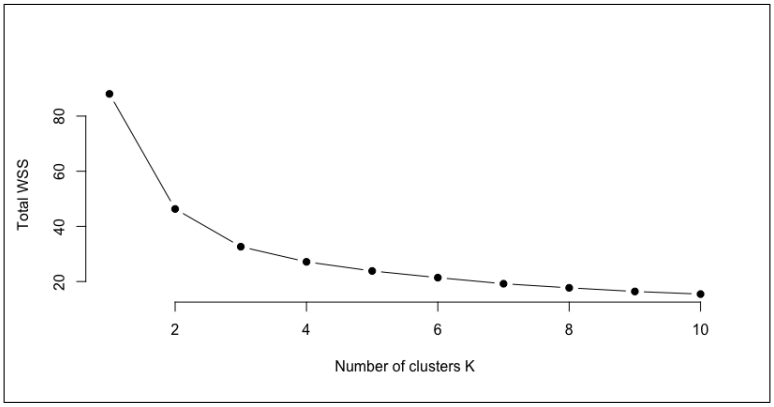


Figure 6.6: WSS

data from real drivers.

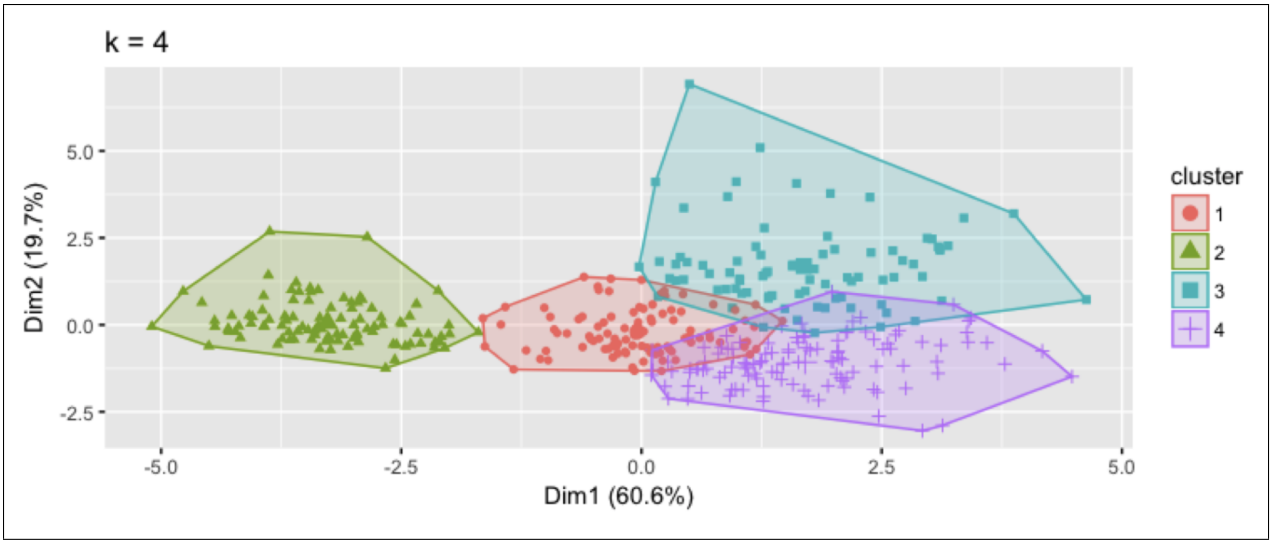


Figure 6.7: Personality Clusters

6.2.1 Real Users Data Collection

We have recruited 50 android users (29 males and 21 females, mean = 33.8, Std = 5.3) in several cities to collect data at two stages. In the first stage, we recruited ten users to collect events data to train the SVM model. They were assigned a task to run a simple version of AutoCoach to collect 200 events each. The GUI displays a counter, and every time an event is detected, the counter is incremented. It took about 1.5–2 h to collect 200 events when driving on local roads.

We needed more real users to build the personality model using the K-means clustering algorithm in the second stage of our project to let actual users test it. We recruited 40 more drivers from different cities to drive under different road conditions and different rules. We have introduced the drivers to the project, and this time drivers got to experience the actual GUI. We explained the setup directions (e.g., how to run the application and what to expect) and requirements (e.g., phone holder, android version). We tasked the participants to run the application whenever they could drive to collect at least two hours of driving data. While we closely monitored the collected data to make sure it was correct and valid. The collected dataset includes over 350 trips, 12K events, and 7K patterns collected by 50 users

over two months. We used the K-Means clustering algorithm to train the personality model. For accuracy reasons, we selected only 36 drivers who had more than 2 hours of driving data to train the model.

6.2.2 Extended Personality Features Selection

Many features could be included to be part of a personal risk-assessment model. This study considers features associated with habits and emotions to be part of the personality model. In this context, habits are defined by all those features associated with driving events and behaviors. Emotions, on the other hand, are those feelings contributing to the aggression or recklessness of drivers. In [59], authors propose that driving style personalities can be divided into four main types: reckless and careless, anxious, angry and hostile, patient and careful. We believe that drivers have a combination of those personalities. We follow habits and personality characteristics to define features for classifying drivers into different personality groups. We assume that the groups we would like to study are distinguished by how much risk they present. The collected data included information regarding the following:

- Events types;
- Events risk levels;
- Events time duration;
- Events Scores;
- Behavior patterns;
- Behavior patterns scores;
- Angry, reckless, and anxious behaviors.

Table 6.2 describes the features we have selected to train our model using the previously mentioned data. Eight of those features (1–8) were used in our earlier study of AutoCoach. We collected and analyzed more data for this current study, which helped us better estimate drivers' personalities and created more features (9–21).

Features 6, 7, and 8 summarize the emotion-related features. In density of angry behavior (feature 6), we detect a cluster of sequential risky events in a fixed time window. The amount of time a driver exhibits a particular behavior repeatedly determines the *density* of angry behavior. We recognize that reckless drivers have a lower density of risky events, where less risky events are found in a pattern (feature 7). In contrast, for density of anxious behavior (feature 8), we comprehend that anxious drivers are those drivers who present hesitation and have low scores due to the high number of occurring safe events within a window. Other features presented in Table 6.2 represent the habits of drivers.

Because we are considering the risks, we collected information about driving habits, which are defined by how aggressive brakes, accelerations, turns, and lane changes are. Drivers respond to some situations with emotions, and those emotional aspects of driving pose a risk. We use anger, recklessness, and anxiousness as measures in defining the personality groups. Other characteristics for the groups are how their overall driving is seen from the scoring engine's perspective. This feature represents the level of safety of the drivers.

6.2.3 Personality-Based Cluster Analysis

In the personality model, we would like to create multiple clusters to define our customer groups. An unsupervised learning algorithm would work best because of the data's nature and the uncertainty about the probable classifications. Because drivers combine many characteristics represented by anger, anxiousness, recklessness, and other driving habits, a cluster cannot be given a particular name. The K-means algorithm allows us to discover

rules that describe large portions of driver's behaviors and habits. Consequently, it can create personality groups based on the features we assume that represent specific behaviors.

AutoCoach has a daemon on its servers, and it is responsible for automatically retraining the K-means model every time 200 events are uploaded to the cloud-based database. The Personality Cloud Daemon provides the necessary functionality to support the personality analysis of the drivers. It does not directly interact with the AutoCoach Android app. Instead, it gathers information from the Firebase database and writes the results back to it. Decoupling the Daemon with the app allows more sophisticated analytical models and algorithms while reducing the workload of mobile devices. Asynchronous inference also means that the personality analysis happens in the background so that the users do not have to wait for it to be done. The daemon listener knows when new data is added to the database and re-clusters the groups every now and then makes sure the groups are up-to-date. When retraining is triggered, AutoCoach considers adding only those users with sufficient data in the trained model. Users with no data for specific events will be excluded, for accuracy purposes, from the model and will remain in their initial group assignment.

With a large number of features, we face the problem of high dimensionality. The higher the dimensionality, K-means becomes less effective at distinguishing between examples [21]. To eliminate the problem, we apply the rule stating that the correlation of magnitude between 0.7 and 0.9 is considered high [68]. Therefore, we decide that highly correlated features with coefficients greater than 0.7 will be excluded. This exclusion will also eliminate the possibility that those features with similarities give more weight than other features. (Figure 6.8)

We use the same statistical measures to examine and find the optimal number of clusters for our current data set. As shown in Figure 6.9, we have tested the average silhouette, the Total Within-Cluster Sum of Square (WSS), and the gap statistics. Each method suggested a different optimal number of clusters (6, 4, and 3). We use the average of the three methods to find that four clusters are about a good number of clusters for the current model. We

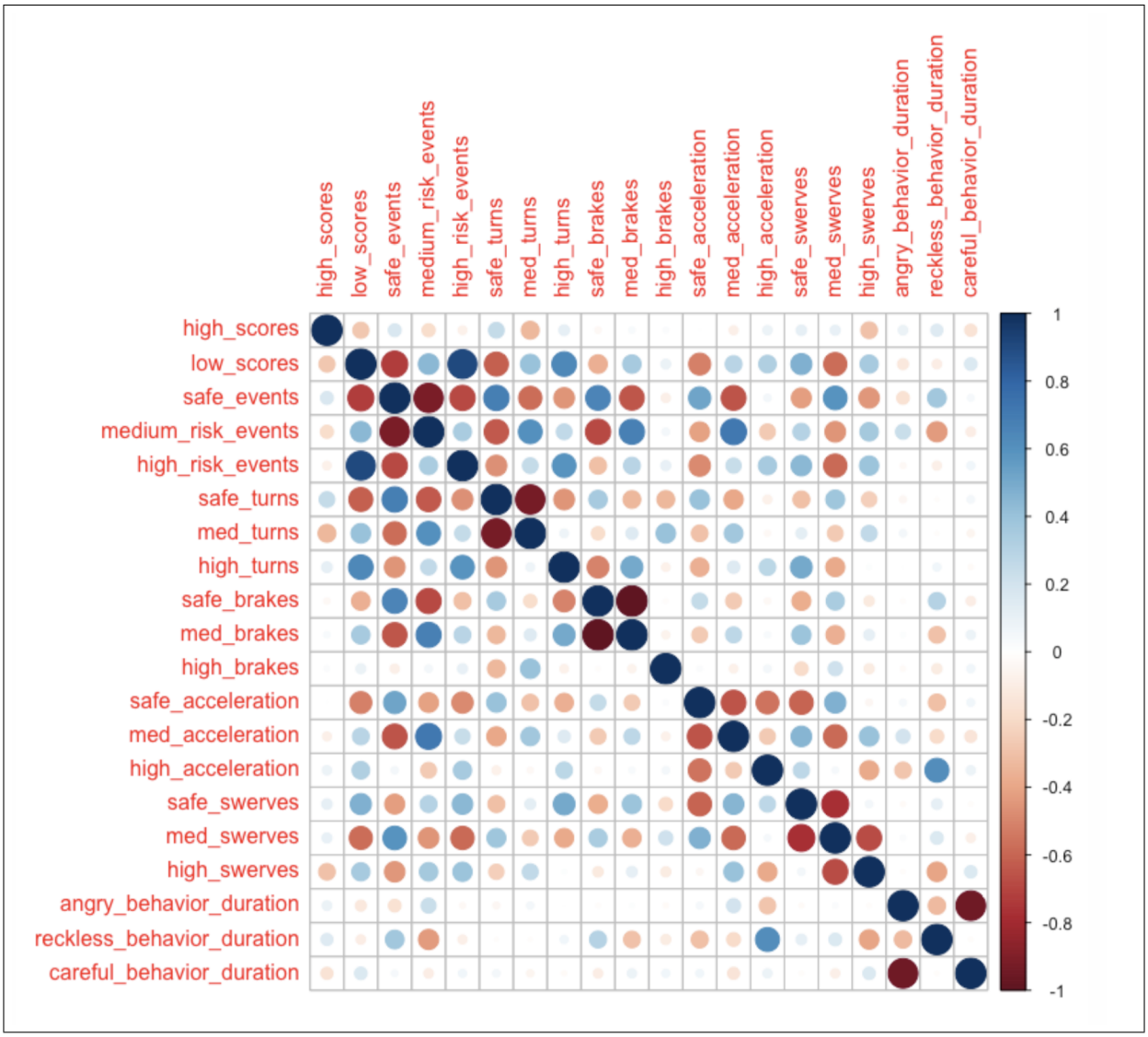
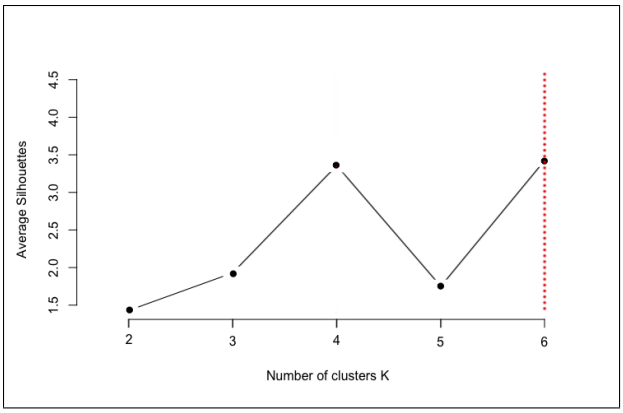


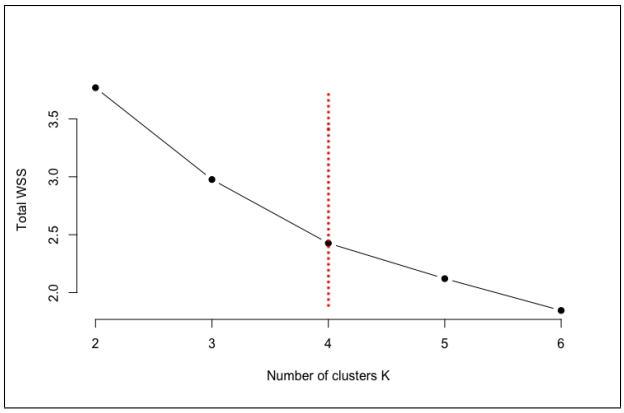
Figure 6.8: Correlation Matrix

label the four clusters in order of safety as follows:

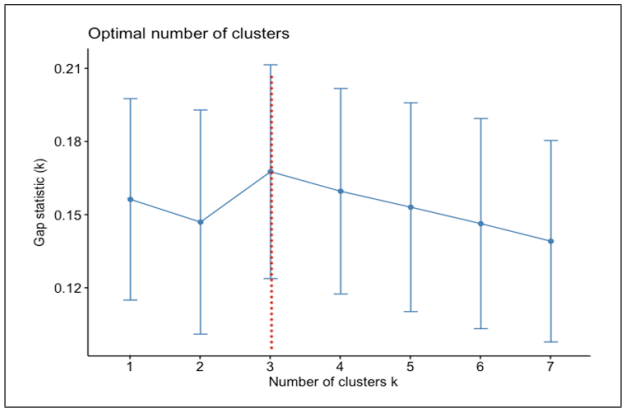
1. Best Drivers: Group A (G_A);
2. Good Drivers: Group B (G_B);
3. Average Drivers: Group C (G_C);
4. Worst Drivers: Group D (G_D).



(**a**) Silhouette.



(**b**) Total WSS.



(**c**) Gap statistics.

Figure 6.9: Statistical methods to determine the optimal number of Kmeans clusters.

When a new model is trained, the Daemon automatically labels the clusters from the safest to the riskiest using the silhouette factor. Silhouette refers to a method of interpretation and validation of data consistency within clusters. We calculate a silhouette score for each feature as if the dataset has only one single feature. We do this for all features selected for training. Then, AutoCoach compares the silhouette scores. The feature with the most significant silhouette score will be chosen to rank and name the clusters. It means that the same cluster samples are similar to each other, and thus, this is a well-separated feature that could create solid boundaries between the clusters. When a driver is assigned to a group, the system will automatically know which associated policies and rules apply to this driver. Every time the model is retrained, the driver will move or stay within the same cluster. Moving from a safe group to a riskier group will result in stricter rules, while the opposite

results in more relaxed rules.

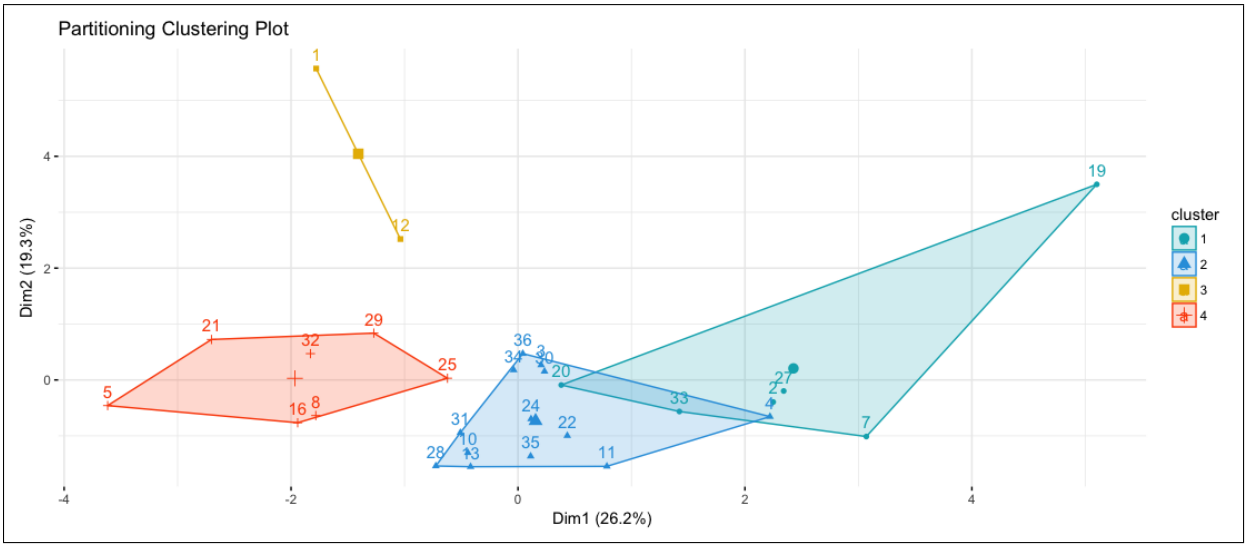


Figure 6.10: Drivers classified into groups using 20 features. Groups are in order from safer to riskier: 1, 2, 4, 3.

Figure 6.10 shows 36 drivers with sufficient data clustered into four personality groups. The numbers on the figure match the following group labels: Best (1), Good (2), Average (4), and Worst (3) drivers. The groups were labeled in this model based on the highest risk turns feature, which had the most significant silhouette score in this trained set. Based on the high-risk turns' cluster values, they will be ranked from safest to riskiest. Those clusters with higher high-turns will be put in the more dangerous group, and as it goes down, they will be labeled as safer. This model was used in our usability study to offer personalized feedback to drivers, which will be explained in further detail in Section 9.

Table 6.2: Personality groups K-means features.

No.	Feature	Description
1	Percentage of safe events	The number of safe events divided by the total number of all recorded events.
2	Percentage of medium-risk events	The number of medium-risk events divided by the total number of all recorded events.
3	Percentage of high-risk events	The number of high-risk events divided by the total number of all recorded events.
4	Percentage of high score patterns	The number of high score patterns where $scores \geq 75$ divided by the total number of all recorded patterns.
5	Percentage of low score patterns	The number of low score patterns where $scores \leq 40$ divided by the total number of all recorded patterns.
6	Density of angry behavior	The number of patterns a driver has been in angry behavior divided by the total number of patterns.
7	Density of reckless behavior	The number of patterns a driver has been in reckless behavior divided by the total number of patterns.
8	Density of anxious behavior	The number of patterns a driver has been in anxious behavior divided by the total number of patterns.
9	Density of Carefulness	The number of patterns a driver has shown safety based on patterns scores divided by the total number of patterns.
10	Percentage of safe brakes	Number of safe brakes divided by the number of all brakes.
11	Percentage of medium-risk brakes	The number of medium-risk brakes divided by the total number of all brakes.
12	Percentage of high-risk brakes	The number of high-risk brakes divided by the total number of all brakes.
13	Percentage of safe acceleration	The number of safe accelerations divided by the total number of all accelerations.
14	Percentage of medium-risk acceleration	The number of medium-risk accelerations divided by the total number of all accelerations.
15	Percentage of high-risk acceleration	The number of high-risk accelerations divided by the total number of all accelerations.
16	Percentage of safe turns	Number of safe turns divided by the total number of all turns.
17	Percentage of medium-risk turns	The number of medium-risk turns divided by the total number of all turns.
18	Percentage of high-risk turns	Number of safe turns divided by the total number of all turns.
19	Percentage of safe lane-changes	The number of safe lane-changes divided by the total number of all lane-changes.
20	Percentage of medium-risk lane-changes	The number of medium-risk lane-changes divided by the total number of all lane-changes.
21	Percentage of high-risk lane-changes	The number of high-risk lane-changes divided by the total number of all lane-changes.

Chapter 7

Feedback and Scoring Engines

AutoCoach is designed to offer feedback, rewards, and penalties to drivers to persuade them to follow the feedback and recommendation given by the system. There are several factors to be considered when designing how to offer those rewards and penalties. How often, how much, what form, and when should they be given are big questions we consider to support our goal of providing positive feedback to support better-driving behavior.

AutoCoach applies persuasive technology [19] to change drivers' behavior by employing means of influence. AutoCoach acts as a coach. It understands if certain personality-related aggression is happening by looking into the current driving behavior pattern score and the driver's personality group. Then, it applies effective measures (i.e., feedback) to direct drivers towards better choices. We implement several concepts of persuasion represented in the *Rewards and Feedback Component* shown in Figure 3.1. It includes:

1. Real-time behavior scoring
2. Personalized reward and penalty decisions
3. Visual and audible feedback

4. Interactive feedback

In our system definition, we have drivers of various personalities, and those personalities reflect varying risk levels. Hence, we may define distinct functions to determine the rewards, penalties and decide when it is essential to trigger feedback. AutoCoach, at this point, understands the differences between drivers and knows the risks different behavior patterns pose. Therefore, those patterns (20 seconds period of driving behavior) and driving events (e.g., brakes or turns) are given scores ranging from 0 to 100 and based on that risk level.

7.1 Patterns Scoring Engine

Many works score drivers based on the frequency (e.g., [40]) and the aggressiveness (e.g., [29]) of driving behavior events. We designed our *Scoring Engine* to score the driving behavior based on the historical data. According to the recorded driving behavior pattern and the driver personality, a score is calculated and sent to the *Feedback Engine* to decide the driver's feedback needs.

The Patterns Scoring Engine has four *Scoring Factors*. Those factors are parameters in the scoring formula. When the Engine receives the patterns probabilistic distribution from the RAU, it calculates the score for the pattern. We have set our system to have four driving behavior risk levels with scoring factors: $R1 = 100$, $R2 = 75$, $R3 = 50$, $R4 = 25$, respectively, from the safest to highest risk. Using those definitions, the higher the pattern probability of occurring within a safer risk group is, the higher the score is. For example, pattern "$hcpah$", the pattern probabilities are multiplied by the scoring factors:

$$\begin{aligned} S_i &= 100 \times 0.12540978 + 75 \times 0.62284267 + 50 \times 0.12623651 + 25 \times 0.12551107 \\ &= 68.7 \end{aligned}$$

where S_i is the score of current pattern i. The result represents that the driver has shown more of a low-risk behavior with a score of 68.7. If a time window has detected no events at all, then a "no events" pattern is considered "perfect behavior."

7.2 Events Scoring Engine

We have defined scores to identify risks associated with driving patterns. Nevertheless, when we want to give the driver recommendations about his driving behavior, we would like to specify which specific event (e.g., brake, turn, acceleration, or lane change) is where he needs improvement at that time. Therefore, we create a simple event scoring engine that defines a score ranging from 0-100 for every performed driving event, and the value of the score depends on the event's risk level and duration.

Table 7.1: Driving Behavior Scoring Factors

Risk Level	**Minimum Score**	**Maximum Score**
Safe events	75	100
Medium-risk events	74.9	50
High-risk events	49.9	0

Given that the maximum scoring threshold is T, where $T = MAX$ is the highest time duration for scoring, anything greater than T will be given the value of T. The score variables are determined based on the event's risk level obtained from the Event Detection Unit. Table 7.1 shows the minimum and maximum score variables used for the scoring equations. Once the risk level is determined, if the event is safe, then the score F:

$$F = (100 - 75) * (1 - min(duration, T)/T) + 75 \tag{7.1}$$

If it's a medium-risk event, then:

$$F = (75 - 50) * (1 - min(duration, T)/T) + 50 \tag{7.2}$$

For high-risk events:

$$F = 50 * (1 - (min(duration, T)/T)) \tag{7.3}$$

This way, we allow scoring safer events to be between 75 and 100, medium-risk events between 50 and 75, and higher risk events to be between 0 and 50. This will create a scoring range from 0 to 100 for any detected events to propose a reliable and fair scoring engine.

7.2.1 Average Event Time Duration

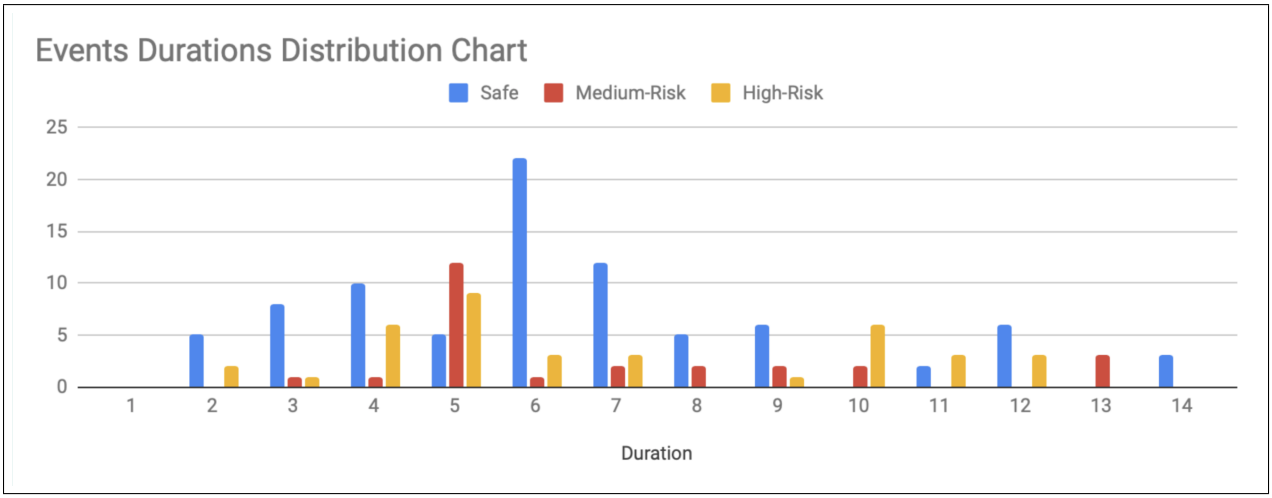


Figure 7.1: Event scoring window size decision data.

When scoring the events, we have set a maximum duration for variable T to set scoring thresholds based on the risk associated with each event (based on its risk level). Any event time duration beyond that maximum threshold will keep the final score the same. We have

decided the threshold window by looking at a sample of randomly selected 150 events from several trips (Figure 7.1). We calculated the median of the events data set, which turned out to be 6 seconds. The average event time duration is 6.77 seconds. We decided to set the scoring window size to be 10 seconds, given that the average event would last for 7 seconds and could go even longer.

7.3 Memory Factor

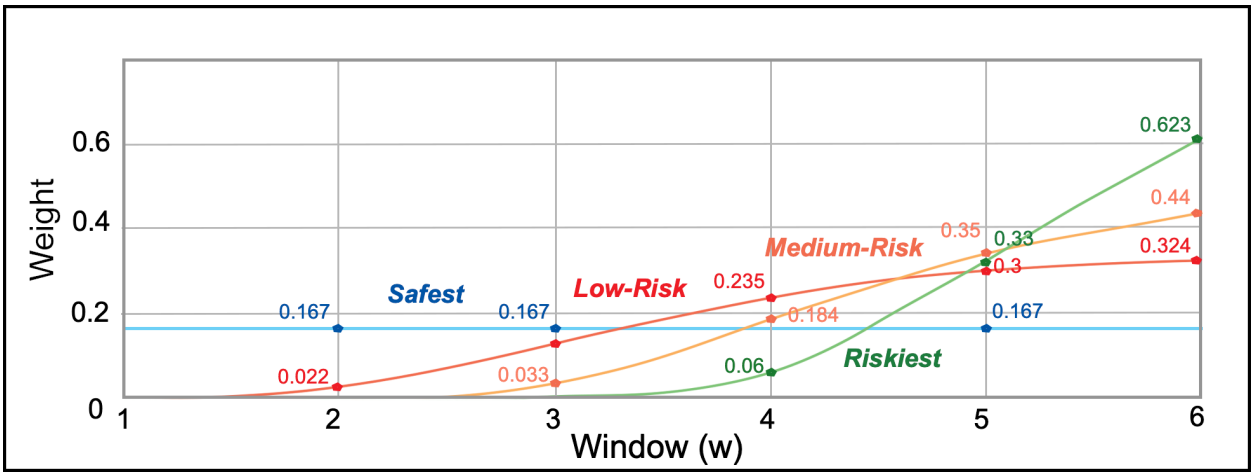


Figure 7.2: Memory Factor

The main contribution to the personalized feedback is what we call a *Memory Factor (MF)*. This factor helps in the decision of when to offer feedback based on the different drivers' personalities. The MF remembers the driving behavior scores recorded from previous patterns. Then it decides how the current score is affected by what's been done in the past. In the MF, we would like to use the idea that the most recent data point is weighing more than the older data. For example, for an angry driver who shows short episodes of anger, the most recent behavior represents aggressiveness. If we consider older data to weigh as much as the current data, then the aggressiveness will be diluted. The S-shaped curve of the Gompertz model is sufficient to be used as a filter on the driving behavior scores. The higher side of the curve gives higher weight to the most recent behaviors, and the lower side of the curve

is used to neglect older data.

We selected Modified Gompertz Equation [60] because it has clear set parameters to describe its curve. Three main characteristics are representing the function. β represents the initial lag. γ represents the period of rapid exponential growth. δ is the period of reduced growth-rate. The Modified Gompertz Equation is as follows:

$$f(x) = \alpha e^{(-e^{\beta-\gamma\times x}+\delta\times x)} \tag{7.4}$$

The model is applied to the dataset $[X : x_1, X_2, .., x_6]$, where X is a function of time. Since δ represents the later reduced growth rate period, it counts as insignificant for our model. Since we have a small data set consisting of six windows, and we would like to have a considerable difference in the weights for each time window w_i, δ's value is negligible. Similarly, γ represents the period of rapid exponential growth, and with a small dataset, we find that setting it to 1 is sufficient. Therefore, the equation is modified to the following:

$$\sum_{x=1}^{6} \alpha e^{-e^{\beta-x}} = 1 \tag{7.5}$$

where $x \in X : \{1, 2, 3, 4, 5, 6\}$ and x is data point of time, then:

$$\alpha[e^{-e^{\beta-6}} + e^{-e^{\beta-5}} + ... + e^{-e^{\beta-2}} + e^{-e^{\beta-1}}] = 1 \tag{7.6}$$

In Table 7.2, we show the variables we used to set four MF functions. We set the initial lag $\beta = 5$ for the riskiest drivers, $\beta = 4$ for the medium-risk drivers, and $\beta = 3$ for the low-risk drivers. For the safe drivers, we set β, γ to equal 0, as we are looking for a linear average.

As an example, we calculate the value of α that satisfies the equation for Personality risk

group 4 by solving 7.6 as a weighted moving average filter for $\beta = 5$:

$$\alpha[e^{-0.37} + e^{-1} + e^{-2.72} + e^{-7.4} + e^{-20.07} + e^{-54.6}] = 1 \tag{7.7}$$

$$\alpha[0.69 + 0.37 + 0.06 + 0.0008 + 10^{9} \times 1.9 + 10^{24} \times 1.9] = 1 \tag{7.8}$$

$$\alpha = 0.88756 \tag{7.9}$$

Those MFs decide how much historical data to neglect and how much to remember. We have set different functions to be distinctive for different personalities. Those functions represent how much weight past information is affecting the current state for four risk groups. Figure 7.2 shows a separate curve for each diverse personality risk group, respectively, from safest to riskiest: blue, red, yellow, green.

Table 7.2: Memory Factor Equations

Equation	(α)	(β)	(γ)	(δ)
$\alpha[6/e] = 1$	0.16667	0	0	0
$\alpha[1/e + e^{-1/e^3} + e^{-1/e^2} + e^{-1/e} + e^{-e} + e^{-e^2}] = 1$	0.33881	3	1	0
$\alpha[1/e + e^{-1/e^2} + e^{-1/e} + e^{-e} + e^{-e^2} + e^{-e^3}] = 1$	0.49997	4	1	0
$\alpha[1/e + e^{-1/e} + e^{-e} + e^{-e^2} + e^{-e^3} + e^{-e^4}] = 1$	0.88756	5	1	0

The MF functions represented by the points shown in Table 7.3 are applied to the pattern scores as a moving weighted average window [24] for every six historical scores. In order to decide the actual feedback score FS_i for pattern i:

$$FS_i = w_k \times s_{i-5} + ... + w_{k-5} \times s_i \tag{7.10}$$

Table 7.3: Memory Factor Functions

Age (k)	Safe (R1)	Low (R2)	Medium (R3)	High (R4)
1	0.167	0	0	0
2	0.167	0.011	0	0
3	0.167	0.134	0.014	0
4	0.167	0.251	0.201	0.036
5	0.167	0.295	0.362	0.394
6	0.167	0.31	0.424	0.57

where k is the number of the weights representing MF (i.e., $k = 6$) and s_i is the score of pattern at time window i. For example, by referring to Table 7.3, a safe driver's current feedback score would be the average of the past six pattern scores (linear average):

$$FS_i = 0.167 \times s_{i-5} + ... + 0.167 \times s_i \tag{7.11}$$

A high risk driver's current feedback score is based on MF's weighted average, where at $k = 1, 2, 3$, $w_i = 0$:

$$FS_i = 0.57 \times s_{i-2} + 0.394 \times s_{i-1} + 0.036 \times s_i \tag{7.12}$$

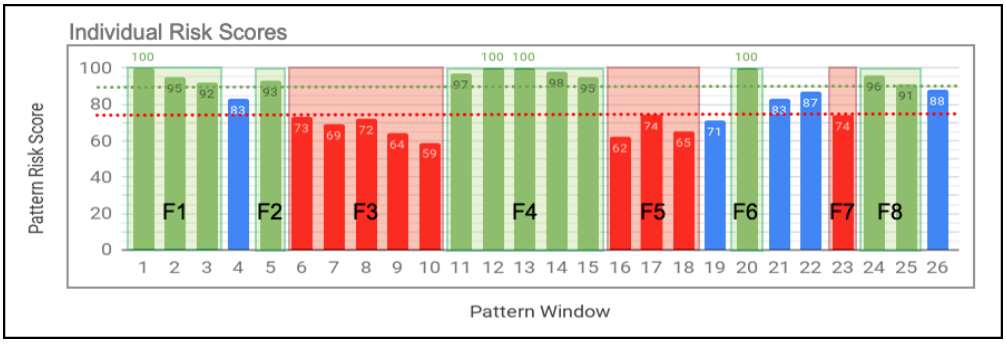
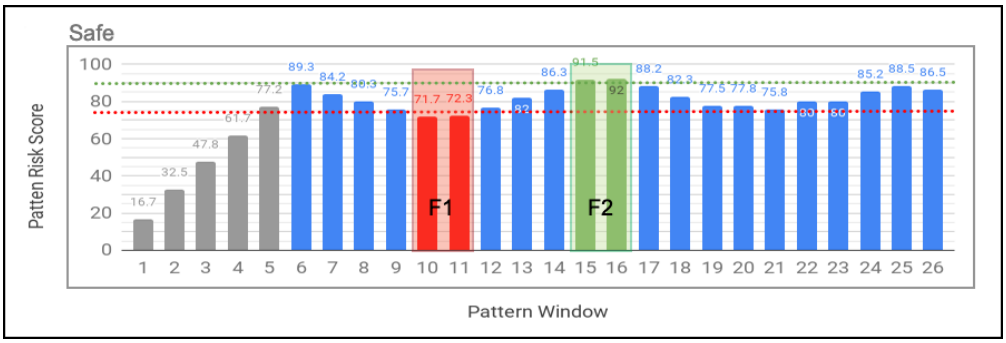
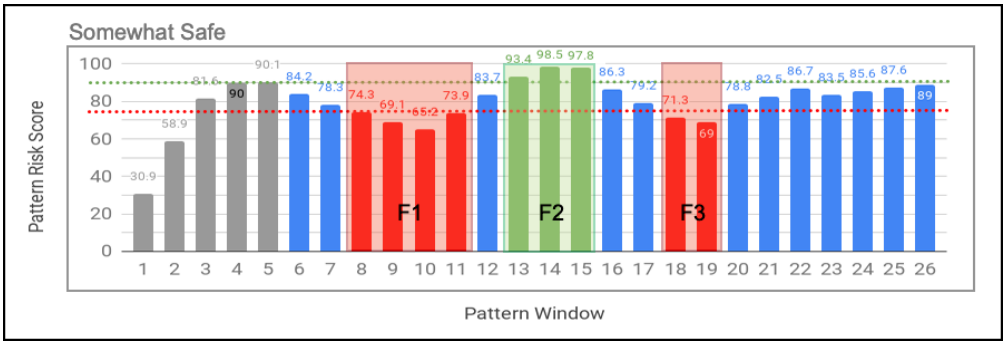


Figure 7.3: Individual Pattern Scores History

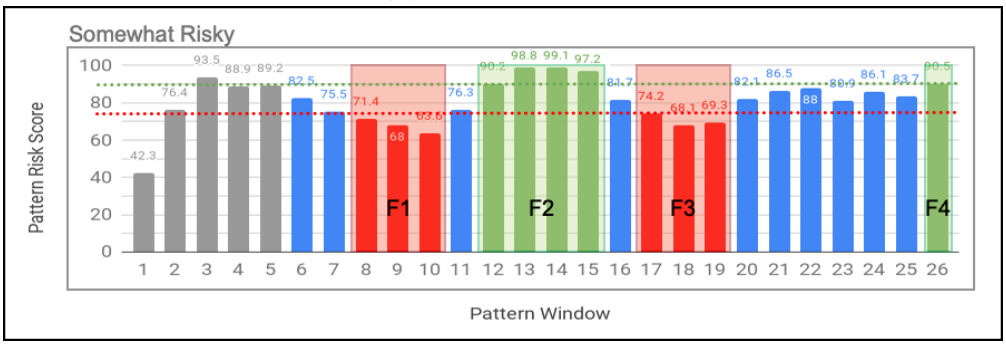
In Figure 7.3, we give an example of 26 patterns score history for a trip. The green bars show the triggered "Good" Behavior feedback when scores are higher than 90, and the red bars are for the triggered "Bad" Behavior feedback when scores are lower than 75. During



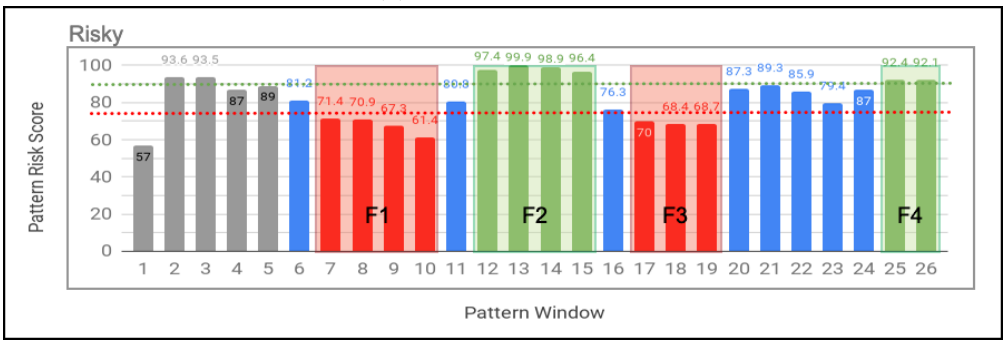
(a) Safest



(b) Somewhat Safe



(c) Somewhat Risky



(d) Riskiest

Figure 7.4: Feedback Score History

time windows 6 to 10, the driver has shown some poor driving behavior and thus received low pattern scores. By applying the R1 to R4 MF functions, in Figure 7.4, we can see how older scores are affecting the current state of the feedback score. The riskier drivers will be given feedback starting at window seven, while the safest drivers do not get feedback until window 10.

Feedback is delayed giving the driver more chances to perform better. If he/she is a safe driver, sometimes some situations can cause the driver to drive badly. Safe drivers are generally aware of their driving behavior and tend to fix their mistakes after the event of danger. While riskier drivers might be reckless or angry and thus, might show more aggressive behavior and needs to be offered some feedback sooner than later. Therefore the idea of MF allows us to weigh newer patterns more for the riskier driver so that the feedback to be triggered earlier than for safer drivers under the same circumstances.

7.4 Feedback Engine

Feedback is essential to deliver what issues the drivers need help with. We defined several components in the Feedback Engine to trigger proper and significant feedback. AutoCoach applies persuasive technology [19] to change drivers' behavior by employing means of influence. Designed to act as a coach, AutoCoach detects if certain personality-related aggression is happening by looking into the current driving pattern score and the driver's personality group from his past behavior. Then, it applies measures (i.e., feedback) to direct drivers towards better actions. We have designed several concepts of persuasion represented in the *Rewards and Feedback Component* shown in Figure 3.1. It includes:

1. Real-time behavior scoring

2. Personalized reward and penalty decisions

3. Visual and audible feedback

4. Interactive feedback

Due to the differences among all individual's unique driving habits, one feedback model does not fit all. AutoCoach offers a personalized personality-based model to address this problem. Personality-based models group people with similarities and then decide when and how feedback should be given to drivers in different groups. In the current design of AutoCoach, two levels of personalization are used. The first level looks at personal habits, and the second level considers each personality group's common issues. Those two aspects together are used to determine drivers' feedback needs.

Specifically, the first level of personalization looks at a driver's history to identify how the driver usually brakes, accelerates, turns, or changes lanes. The historical data of events are analyzed and given scores ranging from 0 to 100. Historical event scores are averaged to develop a final score representing the drivers' typical behavior for each event. Accordingly, each driver has four personal scores (P-scores):

1. Brakes P-score;

2. Acceleration P-score;

3. Turns P-score;

4. Lane-changes P-score.

At the second level, we rely on the Personality Model described in Section 6 to associate a driver to some group of people exhibiting similar risks and use a common strategy to help them. Using this model, group assignments, policies, and memory factors [43] applied to different personality groups, group policies decide the overall feedback strategy, while the

individual P-scores decide the personal adjustment. The group assignments could show us what issues are seen within the specific groups and can be used to adjust the policies and what to focus on. The P-scores scores are used through the feedback policies to adjust feedback triggering thresholds for each event type.

7.5 Feedback Strategies and Policies

We believe drivers should be trained gradually to improve their driving habits. As discussed in [58], drivers can show behavior adaptation after being exposed to feedback, recommendations, and support—there are many possible mathematical and practical models to create successful policies that can influence behavior adaptation.

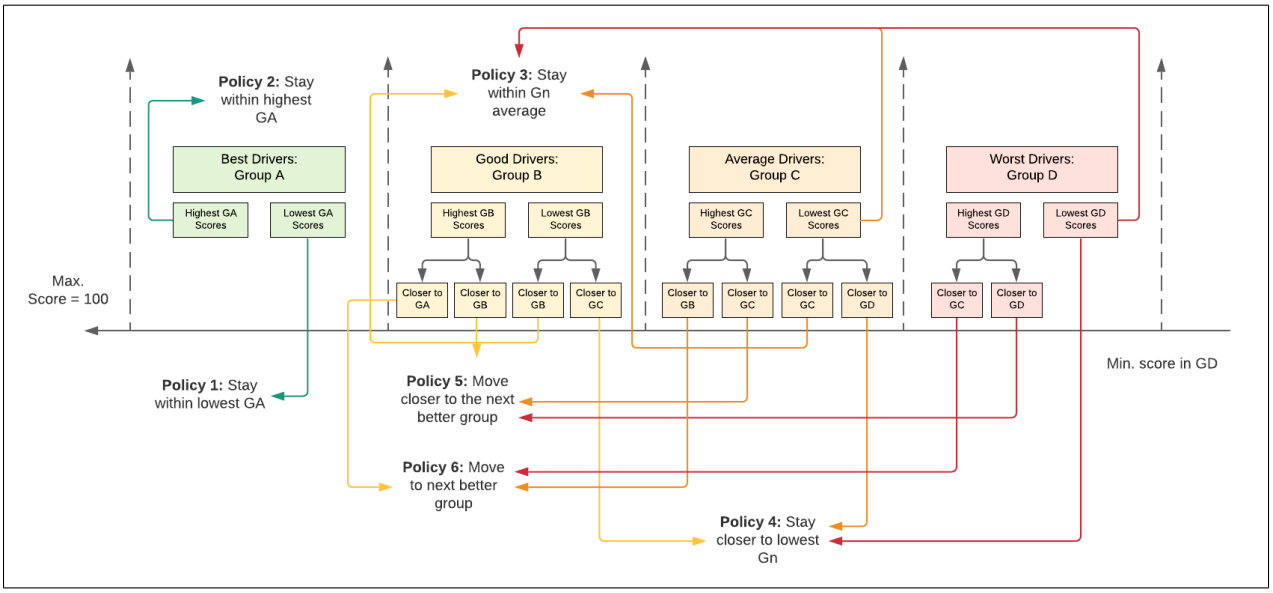


Figure 7.5: Policies Model

In AutoCoach, we offer a deeper level of personalization to the current personality model defined in 6. As described previously, we have clustered drivers into four personality groups: Best (G_A), Good (G_B), Average (G_C), and Worst drivers (G_D). Each driver is assigned to a group, but the driver scores could be closer to the group's mean or farther within the

group. We use those within-group differences to define policies to supporting the following rationale:

1. Best drivers are given more relaxed checking, as long as they will self-correct their driving behavior.
2. Average drivers are encouraged to improve by setting a higher standard above their current and group norm.
3. Bad drivers are monitored closely and given warnings to prevent them from falling even further.

We define two general feedback strategies. The first strategy is outlined for the best drivers in the group G_A, where those drivers are in the best possible group and do not need further improvement; therefore, thresholds are more relaxed. The second strategy applies to the drivers who need improvement. The strategy is set to trigger feedback at slightly higher scores than their standard behavior scores. When the driver shows progress over time, he will be pushed slightly more again. Those strategies push drivers for gradual improvement by helping them either stay within their group range or move up to better groups.

Based on the drivers' group assignments and personal scores, we consider six options in order to create flexible feedback threshold adjustments to push drivers gradually through behavior adaptation:

1. Driver's group assignment is GA and P-score is within the average of the assigned group
2. Driver's group assignment is GA and P-score higher than the average
3. P-score is at the lower edge of the assigned group and closer to the assigned group than the worse group

4. P-score is at the lower edge of the assigned group and closer to the worse group

5. P-score is at the higher edge of the assigned group and closer to the group than the better group

6. P-score is at, the higher edge of the assigned group and closer to the next better group.

According to the options above, we define the personalized feedback policies. The policies define when and how often feedback should be administered according to the group assignments and P-scores. Those thresholds are set differently for drivers based on their P-scores and their group scores.

The two feedback strategies are shown in Figure 7.5. The first strategy is outlined for the best drivers in the group G_A, using two policies (Policy 1 and 2), where those drivers are in the best possible group and do not need further improvement. The second applies to the drivers in the rest of the groups using four policies (Policies 3, 4, 5, and 6). Table 7.4 describes the mathematical model behinds the policies.

Table 7.4: Personality Groups Policies

Policy	Condition	Rules	Threshold Equation (δ)
Policy 1	$P < G_A$	$\delta <= P > G_B$	$\delta = \max[(G_A + G_B)/2, P]$ (7.13)
Policy 2	$P >= G_A$	$G_A < \delta < P$	$\delta = \min[(G_A + \max)/2, P]$ (7.14)
Policy 3	$P < G_n$, $\lvert P - G_n \rvert < Y$	$X = 1$, $Y = [[G_{n+1} - G_n]/2]$,	$\delta = G_n$ (7.15)
Policy 4	$P < G_n$, $\lvert P - G_n \rvert > Y$	$0 < X < 1$, $P < \delta < G_n$, $X = [[G_A - G_n]/[G_A - G_D]]$, $Y = [[G_{n+1} - G_n]/2]$	$\delta = P + [G_n - P] * X$ (7.16)
Policy 5	$P >= G_n$, $\lvert P - G_n \rvert < Y$	$0 < X < 1$, $P < \delta < G_{n-1}$, $X = [[G_A - G_n/[G_A - G_D]]$, $Y = [G_{n-1} - G_n]/2$	$\delta = P + [G_{n-1} - P] * X$ (7.17)
Policy 6	$P >= G_n$, $\lvert P - G_n \rvert > Y$	$\delta = G_{n-1}$, $X = 1$, $Y = [G_{n-1} - G_n]/2$	$\delta = G_{n-1}$ (7.18)

7.5.1 Best Drivers Strategy

We want drivers in G_A to stay within the group score ranges and be given a more flexible threshold when triggering feedback when they are better thanG_A's average. Because drivers in this group have moderately high scores, the feedback threshold should not be a solid boundary but must be more flexible as long as the behavior scores do not get too low to reach group G_B. We define two policies for this strategy.

1. Policy 1: Threshold is set to keep drivers within-group range. If the driver's P-score (P) is below G_A, then the feedback triggering threshold (δ) should be either $\delta = P$ or $\delta = (G_A + G_B) \,/\, 2$; whichever is greater. This allows keeping the driver closer to G_A's average. This is defined by using Equation 7.13.

2. Policy 2: It is designed for those best drivers in G_A, whose scores are higher than average. Thus, no more improvement is needed. Threshold is set using Equation 7.14 to be either $\delta = P$ or $\delta = (G_A + MAX) \,/\, 2$; whichever is smaller. Since this driver's performance is greater than the best group's average, scores are more flexible and should remain where he is.

7.5.2 Drivers Improvement Strategy

For drivers in (G_B, G_C, G_D), we want drivers to keep up within the group if the driver's P score is below the group's average; otherwise, try to improve to the higher group if the score is above the current group's average. In Figure 7.5, G_n represents the driver's assigned group average score, and G_{n-1} is the higher group's average score (E.g. if $G_n = G_B$, then $G_{n-1} = G_A$). The value of Y is a measure of the difference between the driver's assigned group and the "better" or "worse" group averages. It is used to compare how close $P - score$ to the assigned group score is compared to another neighboring group. We use

$X = [[G_A - G_n/[G_A - G_D]]$, where $0 < X < 1$ as a factor to slightly adjust scores.

1. Policy 3: In this policy we help drivers get closer to group average by setting the threshold strictly equal to the assigned group average. If $P < G_n$ and $Y = [G_{n+1} - G_n]/2$ indicates that P is closer to G_n than the worse group, then, we want the driver to stay within his group average G_n, so we try to push him closer to G_n. We use Equation $\delta = G_n$ 7.15 to set the threshold closer to G_n.

2. Policy 4: In this policy the system finds drivers fallen behind and got to be closer to the worse group, therefore, threshold is set slightly higher to issue more proactive feedback. If $P < G_n$ and $Y = [G_{n-1} - G_n]/2$ indicates that P is closer to the worse group, then we want to give the driver a little push closer to G_n by applying Equation $\delta = P + [G_n - P] * X$. 7.16.

3. Policy 5: This policy is used when a driver is performing better than the average and thus, can be pushed to do event better by setting a slightly higher threshold than group average. If $P >= G_n$ and $Y = [G_{n-1} - G_n]/2$ indicates that P is closer to G_n than the better group, we expect the driver to do slightly better than his current behavior by setting the threshold closer to G_{n-1} via using applying the equation $\delta = P + [G_{n-1} - P] * X$ 7.17.

4. Policy 6: This policy is used when a driver is performing better that the group average, and is found to be closer to the next better group, therefore, the new threshold is set to be equal to the new group threshold. if $P > G_n$, $Y = [G_{n-1} - G_n]/2$ indicates that P is closer to the better group, then we set $\delta = G_{n-1}$. as shown in Equation 7.18.

Those policies push drivers for gradual improvement by helping them either stay within their group range or move up. When drivers are ready, they get a slight adjustment in the feedback threshold to create a more sensitive feedback engine.

7.5.3 Rewards Engine

AutoCoach could be used as an application in the Auto Insurance business, where collected coins can be applied towards rewards, coupons, and discounts in real life. On the human side, we follow gamification concepts in non-game contexts to reward drivers as a positive reinforcement measure. Rewards are represented by golden coins popping on the screen with sound effects as a definition of achievement. Much like many current smart fitness tracker products (e.g., Fitbit), such positive feedback does help individuals make even more efforts to improve their behaviors.

We have designed the Rewards Engine to reward drivers in two conditions:

1. Rewards for behavior improvement
2. Rewards for good behavior

In our model, the "Rewards for Good Behavior" is the main scoring mechanism for safe drivers. Safe drivers can earn many coins as long as they are driving safely (e.g., every 40 seconds). On the other hand, risky drivers can also earn coins but only when it is clear that they have improved. Risky drivers may not receive as many coins, yet they can receive "Rewards for Improvement" coins as means of encouragement.

For the first mechanism: Rewards for Good Behavior, the behavior score is calculated every 40 seconds by the Pattern Scoring Engine as described in Figure 7.6. The current pattern score is adjusting using the memory factor (6-window moving average) to consider the history from the previous five windows [43]. Whenever this score is greater or equal to the personal threshold, the driver is given a reward coin. Safe drivers can easily earn good behavior coins.

For the second condition: rewards for behavior improvement, the reward engine is triggered when an event is detected, scored by the Events Scoring Engine, and adjusted by the memory

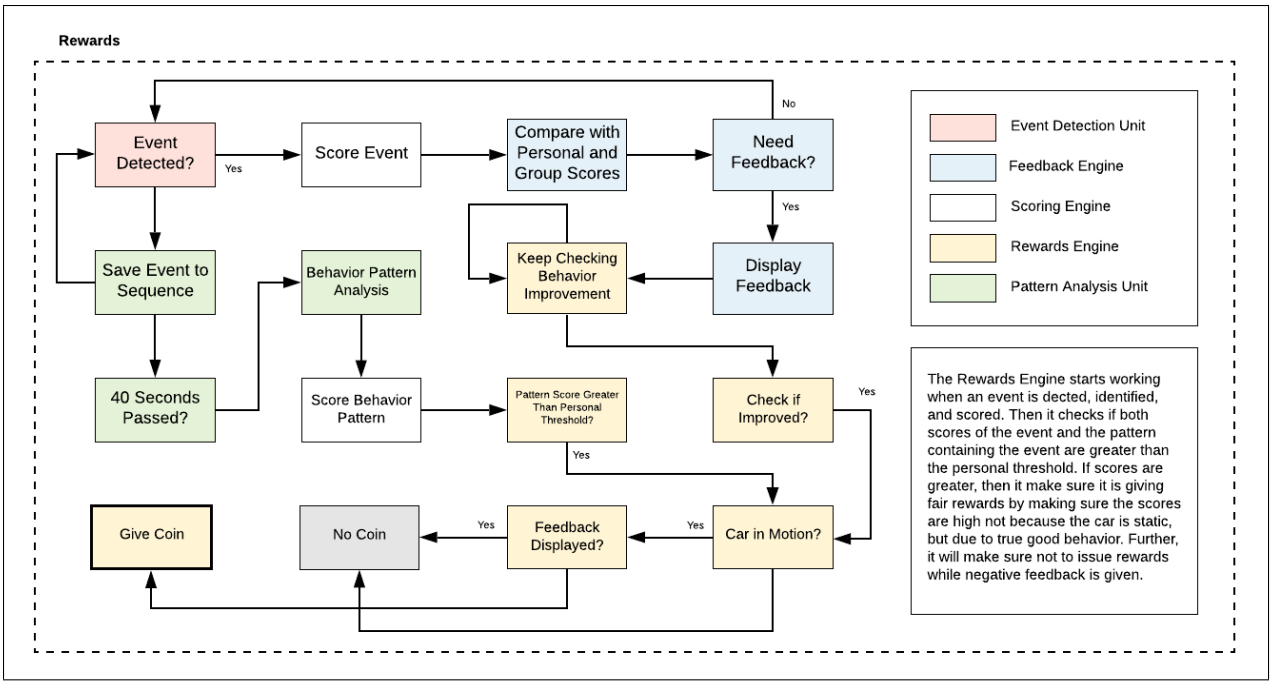


Figure 7.6: Rewards Engine Diagram

factor. The event score is compared with the event's p-score to decide whether the feedback is needed or not. If feedback is required, then AutoCoach issues the feedback, stores the feedback status, and waits until the score is improved. The improvement is time-based. As long as the driver does not make another risky event of the same type, they will notice the event bar will decline every 10 seconds until it reaches below the personal threshold arrow. This is when rewards should be deserved.

Algorithm 1 shows the Pseudocode for how the Feedback and Rewards Engines are working together in AutoCoach.

Algorithm 1: Feedback/Rewards Engine Algorithm

```
Result: Display Feedback on GUI
// Initialize driver personality group and personal information to
   match current driver feedback engine setup
FeedbackEngine = new FeedbackEngine();
while Feedback Engine Thread is Active do
    if Event is detected then
        Calculate event score;
        Adjust score based on the assigned strategy;
        Update event bar on the GUI;
        Check if driving behavior requires recommendation;
        if Recommendation is suggested then
            Save recommendation information for later behavior improvement check;
            // If there is feedback running, and driving behavior is
               improved or been safe for a while, do not reward right
               away, wait some time for recommendation box to disappear
            feedbackTimer = 0; // Rest FeedabckTimer
            getRecommendationEvent();
            if There is a running recommendation then
                if new recommendationEvent = current recommendationEvent then
                    if Current running RecommendationEvent score > New
                      RecommendationEvent score then
                        Replace recommendation with new;
                    else
                        Ignore new recommendation;
                    end
                end
                else if new Recommendation != current Recommendation then
                    Display Glow on event Bar
                end
            end
            else if There is NO running recommendation then
                Set new Recommendation on GUI;
            end
        end
        else if No Recommendation is suggested then
            Check if the driver deserves rewards
        end
    end
    Check behavior improvement based on previously given feedback for rewards;
end
```

Chapter 8

AutoCoach 1.0

8.1 AutoCoach GUI Components

Figure 8.1: AutoCoach Application

We have implemented AutoCoach, an Android-based application designed with a friendly GUI with minimal text but more color variations to reduce distraction and allow easy tracking of driving performance. AutoCoach GUI has several components on the screen as described in Figure 8.2:

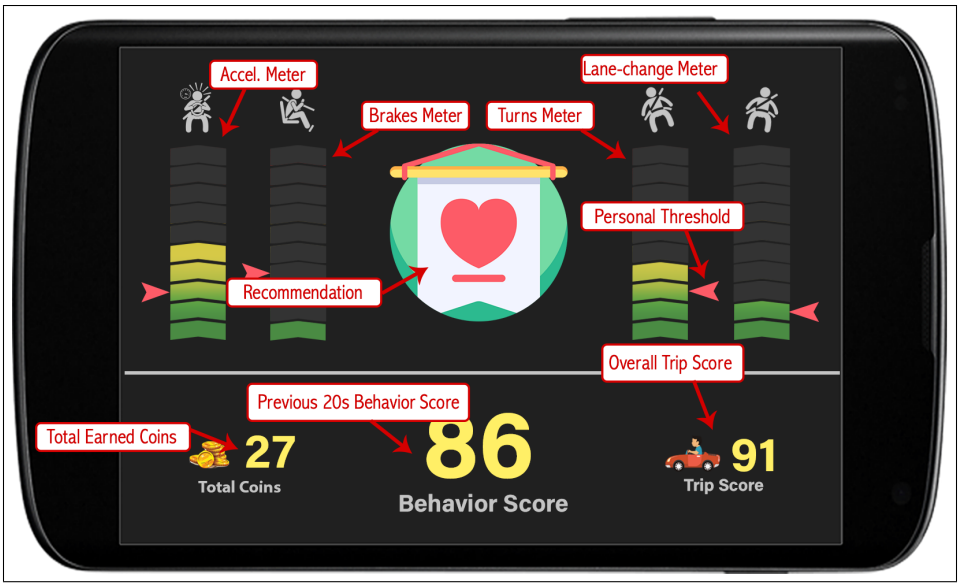


Figure 8.2: Main screen components.

1. Event meters: those meters represent the scores of the individual events (accelerations, brakes turns, and lane changes) within the last 10 seconds moving average window.

2. Personal event thresholds: are red arrows next to each bar. Those represent the driver's personalized thresholds calculated based on the drivers' habits and personality group scores.

3. Recommendation component: This is where recommendations, improvements, and rewards icons are presented in the center of the GUI.

4. Total earned coins: The total earned coins for this specific trip.

5. Behavior score: The score for the driver's behavior in the last 40 s window.

6. Trip score: The average score of all behavior scores from the beginning of this specific trip.

The majority of the previously summarized components have been described in detail in the previous chapters. The recommendations component and the Events bar components will be discussed in further detail in this chapter.

8.1.1 Recommendations Component

The recommendation component is where the drivers receive feedback regarding their driving behavior. As described in Figure 8.4, once AutoCoach starts, the heart badge will always be displayed in the recommendation box. As events and patterns are detected and scored, the system makes decisions whether to issue a recommendation or not. There will be recommendations for brakes, accelerations, turns, and lane changes, and they will be identified by the icons shown in Figure 8.3.

Figure 8.3: Events Icons on AutoCoach's GUI

If a recommendation is needed, then it will show an icon with the event type, like a brake, as shown in the figure. The background of the event image will glow either red or yellow. Red means high-risk, while yellow means low-risk. If the behavior is improved, then a green glow will show along with the icon of the improved event, followed by an Earned Coin icon.

Coins are earned in two ways, as discussed in Section 7.5.3. They can be either behavior improvement coins or good behavior coins. Behavior improvement coins are given when behavior is improved following a given recommendation. Drivers who are issued recommendations will no longer earn coins as long as the recommendation is on. The good behavior coins are given when the following conditions are true:

1. Current speed > 0 or the car is not idle.

2. Pattern (window) score $\geq$ personal threshold.

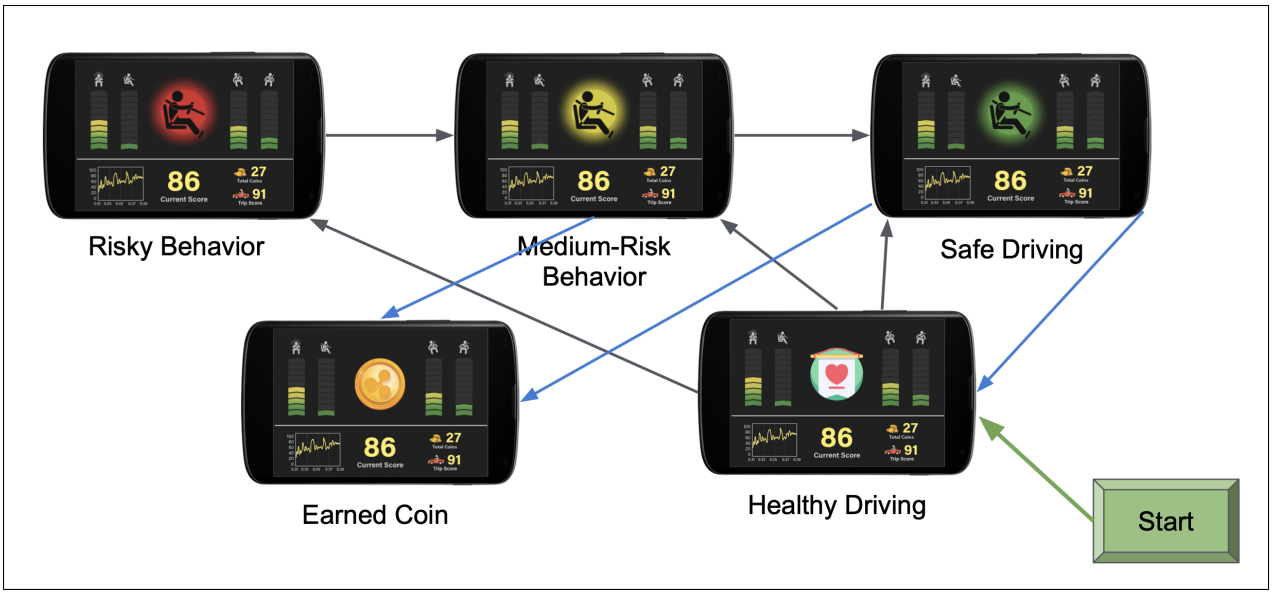


Figure 8.4: Feedback Options

3. No recommendation is displayed (no outstanding warnings).

8.1.2 Event Bar Component

Each event bar component is used to display the score of the detected events in real-time. The events scores are in the format shown in Figure 8.5. In our context, these scores are calculated based on the risk level of the detected event. The higher the risk, the lower the score. Then, the score is compared to the driver's event P-score, and when the score $<$ P-score, feedback is triggered. We use a weighted moving-average window to calculate a new score every 10 seconds. The decision was to use a 10 s window because we found from the collected data that the average event duration is 6 seconds and to make sure at least one event is recorded every window, we set the window size to 10. Whenever no event occurs in a window, then the window score $= 100$.

Given that the maximum score $s = 100$, the maximum events detection window size $w = 10$ s and the system can detect three-risk-levels (safe, medium-risk, high-risk) for each event

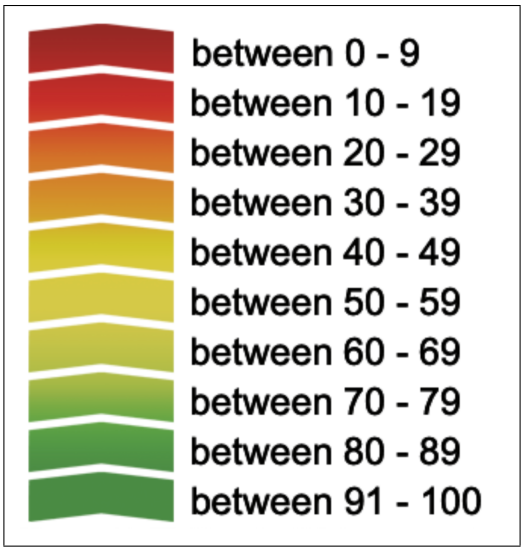


Figure 8.5: Events scoring.

type. We have segmented scoring rules into three divisions. For safer events, score ≥ 74, for medium-risk events, $74 >$ score ≥ 49, and for high-risk events, score < 49. We selected these parameters according to the users' scores of single-event patterns obtained by the patterns scoring engine. We learned that medium-risk single-event patterns are scored on average 74, while high-risk single-event patterns are scored 49.

If the Event Detection Unit detects a safe event with time duration t, then, The Events Scoring Engine scores it as follows:

$$s = (100 - 74) \times (1 - min(t, w)/w) + 74 \tag{8.1}$$

If the detected event was classified as a medium-risk event, then:

$$s = (73 - 49) \times (1 - min(t, w)/w) + 49 \tag{8.2}$$

Otherwise, if it is a high-risk event, then:

$$s = 48 \times (1 - min(t, w)/w) \tag{8.3}$$

These equations allow scores to vary between 0 and 100, giving safer events higher scores while riskier events lower scores. Each bar on the GUI is personalized based on the policies and memory factors assigned to the drivers based on their personality group.

There are many possible ways to design the scoring model to trigger feedback. The focus can target improving personal habits, group habits, current trip performance, or driving history-related performance. We could build a weakness-based scoring engine, where it checks the worst among brakes, accelerations, turns, or change lanes, and then use this information to focus on that one specific event for personal improvement. We also could build a history-based model that monitors the p-score trends (e.g., from the past ten trips). Then, classify a driver to whether he is improving or worsening and set policies accordingly; i.e., if a driver is already improving, we would not set thresholds to push them so much anymore. We also consider scoring under different road and weather conditions; i.e., rainy, we care more when the driver drives more aggressively.

8.2 Application Screens

In Figure 8.6, we describe AutoCoach application screens. When AutoCoach is installed and run, users will be asked to log in. The users should create a username and password if they were first-time users; otherwise, they can log in with their previously created credentials. Once logged in, they will be taken to screen with the "Start AutoCoach" button. This screen allows users to place their phones on the car phone holder in a landscape view. Once the driver starts a trip, AutoCoach will calibrate the sensors right away and continuously

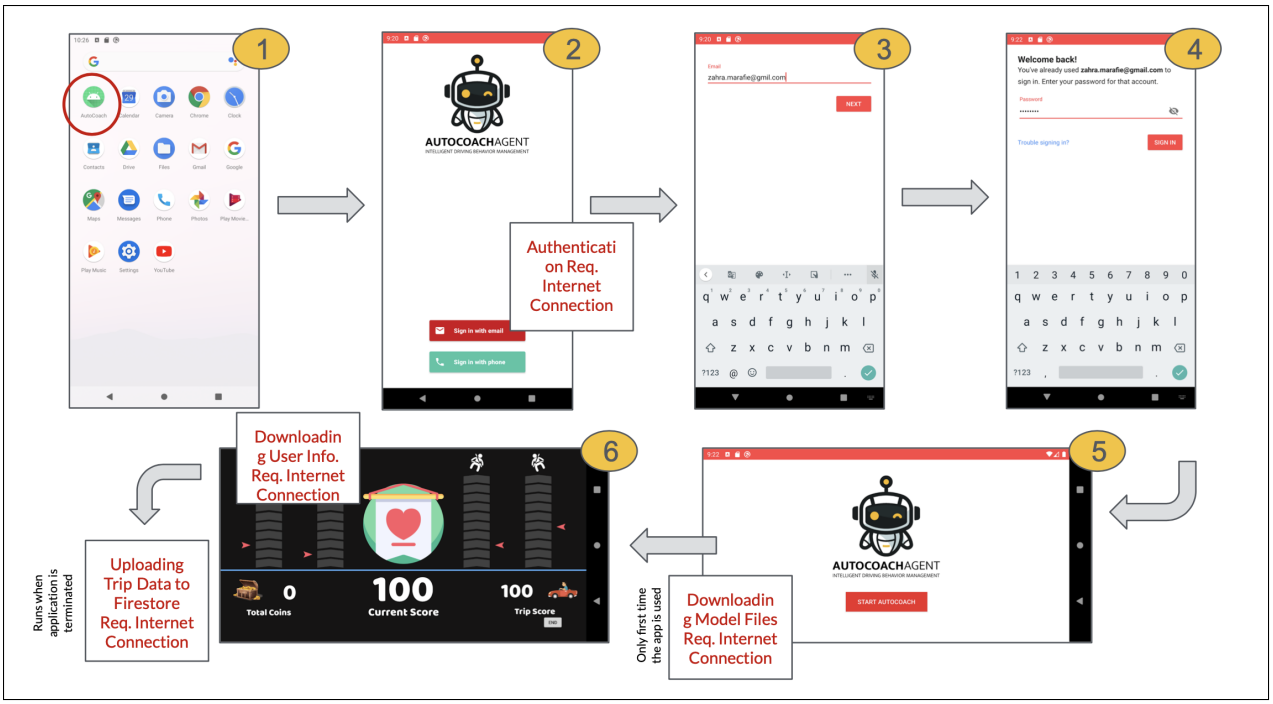


Figure 8.6: GUI screens.

collect and analyze sensor data to generate intelligent, personalized decisions about real-time driving behavior.

8.3 System Components

AutoCoach consists of three integrated components as described in Figure 8.7: AutoCoach App, Google Firebase, and Personality Analysis Daemon. AutoCoach application is the user interface where driving behavior is detected, analyzed, and responses to driving behavior are shown. Once the user opens the AutoCoach application, he will be prompted to enter a user name and password or create one. This is used for authentication and associating stored data with each user to create a personalized experience. Once logged in, the user will be taken to the start AutoCoach screen (Screen (5) in Figure 8.6). The purpose of this screen is to allow the driver to place the phone on the phone holder in a landscape view for calibration as we have the axis set for calibration in that position. If this was the first time

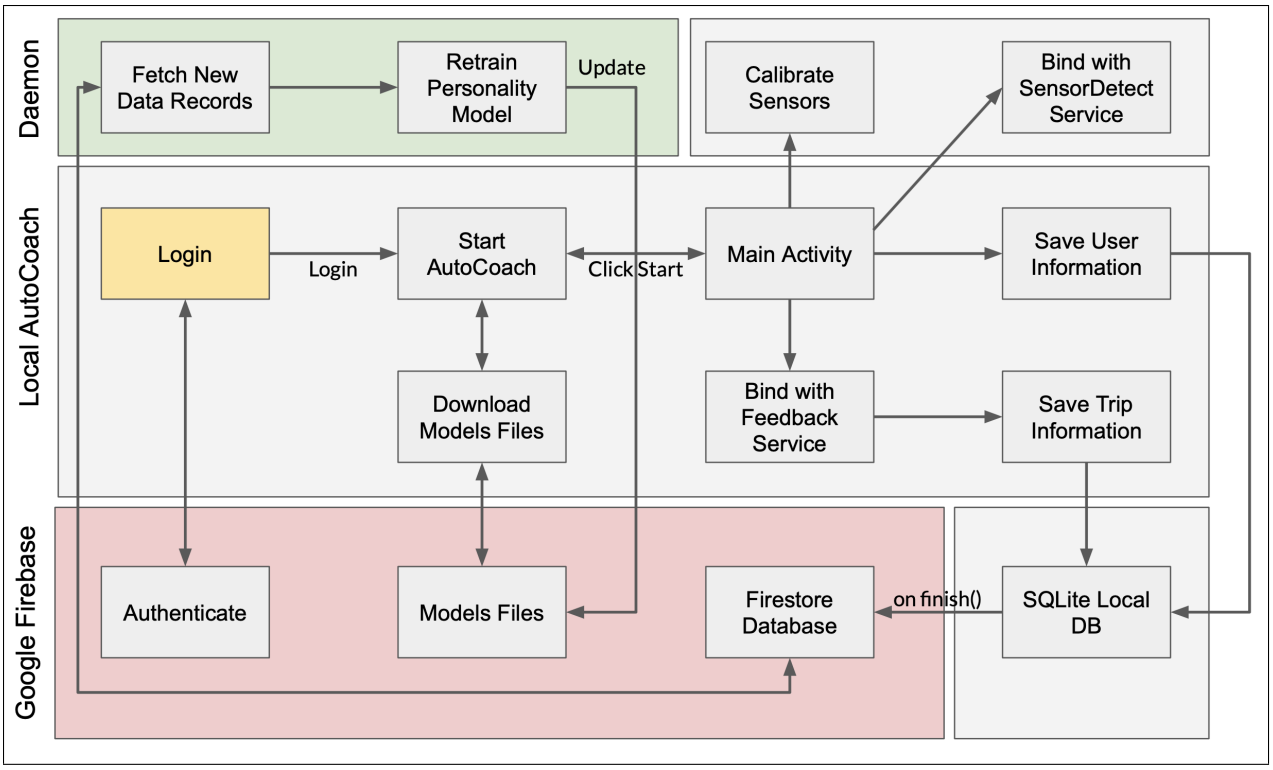


Figure 8.7: GUI functional diagram

the user installed AutoCoach, the system will start downloading the model files from the Firebase server. There files that will be downloaded:

1. SVM model file
2. LDA model files
3. K-Means model file
4. Personality groups data file

Google Firebase is where all the model files are stored and where the collected data is sent after every trip. Once the driver is read, he will click "Start AutoCoach." Then, the sensors will immediately calibrate, and the system is ready for use. We will discuss the main screen further in the following sections. Whenever a driver clicks the "End" button, the locally

stored data is segmented into small portions and sent to Google Firebase servers in the background whenever no other processes are running. The Personality Analysis Daemon is a background application that runs on a server. It remains asleep and awakens whenever new data is added to the Google Firebase database. The daemon waits for at least 200 events to be added; then, it will pull the newly added data, retrain the personality model, and send a new version to the Firebase server. Therefore, whenever a user logs in for his/her next trip, the new model will be downloaded, and the personality assignment decision will be updated.

8.4 Feedback Risk-Level Decision

(**a**) (**b**) (**c**)

Figure 8.8: Feedback event risk level compared to driver's personal threshold. (**a**) Risky–Red glow. (**b**) Medium-risk–Yellow glow. (**c**) Safe–Green glow.

In this section, we look at how we decide the risk level of the feedback:

1. High-Risk (Red Glow).
2. Medium-Risk (Yellow Glow).
3. Safe (Green Glow).

The background glow represents how risky the feedback event is compared to the driver's personal threshold.

When feedback is triggered, and a decision is made on which event is responsible for the given feedback. Then, given T, a value that represents the maximum score difference between the current event score C_x and the average of personal score P_x and group score G_x for the feedback event, the decision will be made based on Algorithm 2. In the current model, we set $T = 10$.

Algorithm 2: Feedback Risk Level Decision

```
Result: Decide feedback event's risk-level
// Determine the behavior risk-level to decide glow color
// P_x is the driver's personal score for the feedback event
// G_x is the driver's group score for the feedback event
// C_x is the current event's score
// T is the maximum difference value to switch from one risk level to
   the other
if Feedback is triggered then
    if Feedback event is decided then
        Difference = |(Avg (P_x,G_x) - C_x)|;
        if Difference ≥ T then
            Display red glow;
        end
        else
            Display yellow glow;
        end
    end
end
```

8.5 Behavior Improvement Decision

Rewards is one of the important pieces of AutoCoach. Determining when to offer drivers rewards is a critical task. As mentioned in Section 7.5.3, there are two ways to earn rewards; one of them is through behavior improvement. Figure 8.9 describes the behavior improvements model. As long as the system is running, it will be checking whether feedback displaying is triggered. If there is feedback that needs to be shown to drivers, then the system should decide how risky this behavior is compared to the driver's normal driving

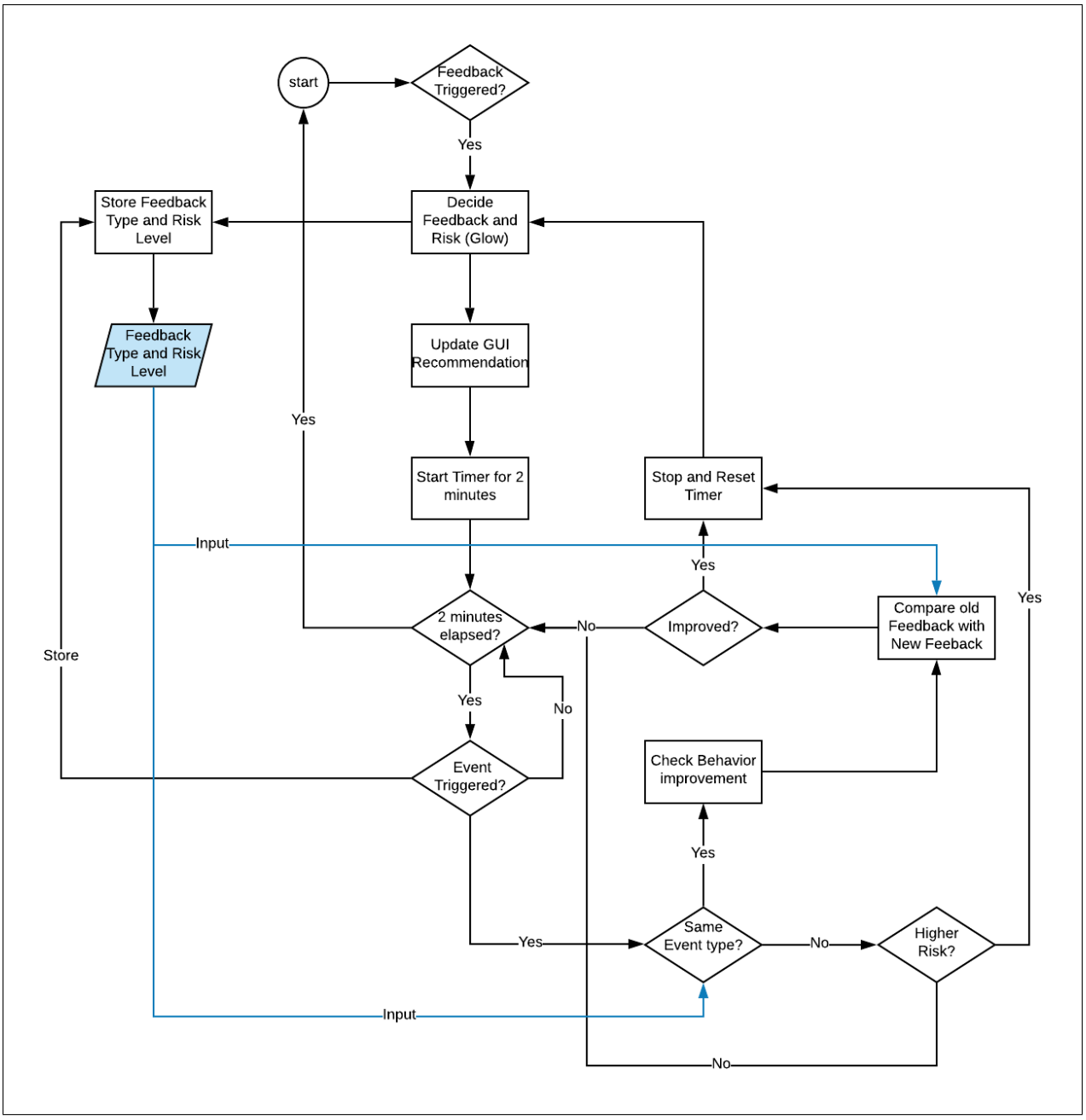


Figure 8.9: Behavior Improvement Design

according to Algorithm 3.

Algorithm 3: Behavior Improvement Decision

```
Result: Decide if behavior has improved and issue rewards for improvement
// Determine the behavior risk-level to decide glow color
// currentEventScore = the event type score in the current window
// timer = 2 minutes duration for behavior improvement in order to
   receive rewards coin
if Feedback is decided then
    while Wait 2 minutes for behavior improvement do
        if new pattern is received then
            Diff = (Avg(P_x, G_x) - C_x);
            if currentEventScore ≥ personalThreshold then
                Display green glow;
                Wait(2 seconds);
                Display rewards coin;
                Increment rewards coins counter;
            end
        end
    end
end
```

8.6 Feedback and Rewards Engine Functional Diagram Description

Feedback and rewards solely depend on the detected events and patterns. Based on the model shown in Figure 8.10, AutoCoach waits for an event to be detected. When an event is detected, there are two paths that the system will take. The first path is related to pattern analysis and good behavior rewards. The event will be added to the patterns list. Once the 20-second window is complete, the behavior pattern is sent to the Patterns Analysis Unit (Section 5) to analyze the behavior, then send to the Scoring Engine to find the score for the behavior. The score is adjusted based on the memory factor assigned to the driver (Section 7.3). The Rewards function checks whether the adjusted behavior score is greater or less than the personal threshold. If it is greater, then the driver deserves a reward. However, the reward will not be issued if the car is not in motion. If the car has stopped, then issuing the reward will seem confusing. Also, if there was a recommendation displayed on the screen

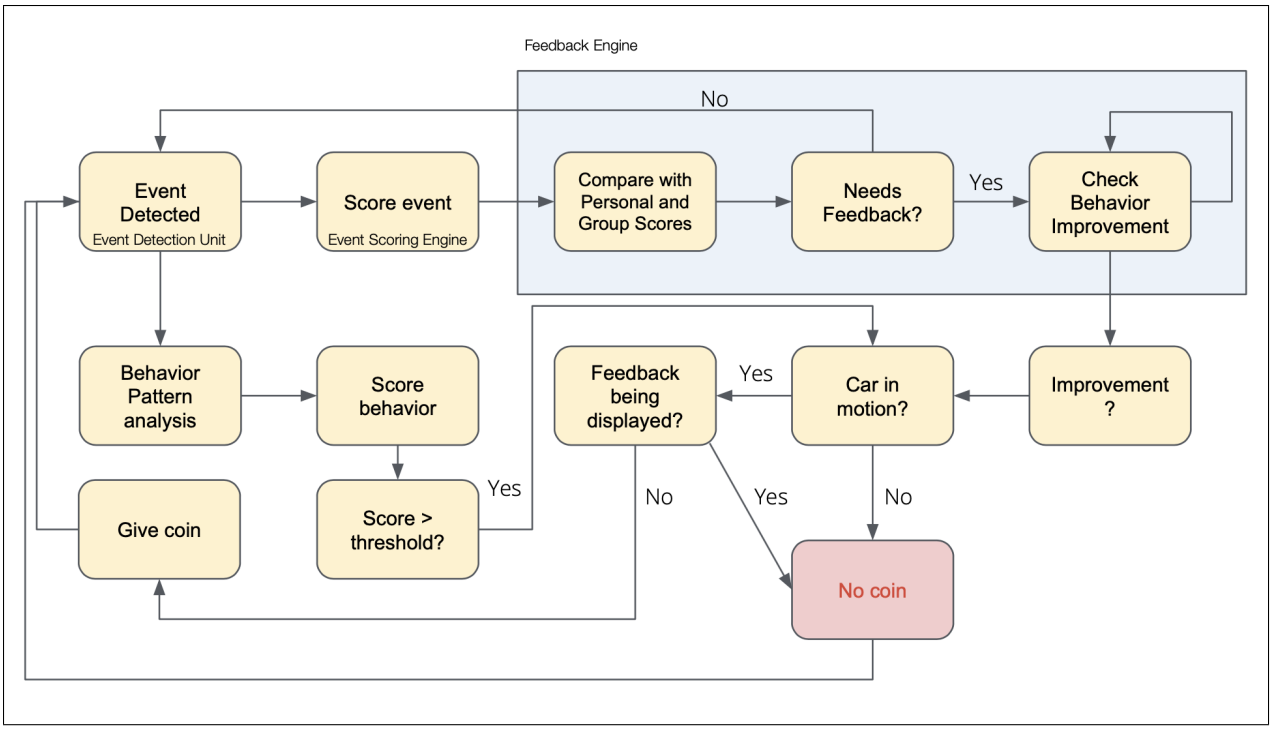


Figure 8.10: Feedback Engine

regarding some risky behavior, then issuing the reward, in this case, is not logical; therefore, in both cases, the system will wait for the next window, where the behavior score is greater than the threshold to offer a new reward.

The second path the system will take when an event is detected is associated with the feedback engine and behavior improvement rewards. If an event is detected, the event will be scored and adjusted based on the memory factor. Then, the score will be compared with the driver's personal scores (e.g., Brakes personal score). The system will decide whether feedback should be issued or not based on the decisions made (Algorithm 1). Then, the system will wait for behavior improvement. If improvement is detected, check whether the car is in motion or any recommendations are displayed; if not, then give the driver a rewards coin.

8.7 AutoCoach Use Case Model

A use case (UC) diagram is a graphical view of all the possible interactions between the system components and the user. In Figure 8.11 we explain in detail the interactions between the users and the system. In AutoCoach, several actors interact with the system:

1. Google Firebase Authentication
2. System's Local Database
3. Google Firestore Database
4. User (Driver)

The system initially waits for the user to Create an Account (UC1) if he was a first-time user and logs in (UC2). If a user has forgotten his password (UC3), he may reset it. Once logged in and "Start AutoCoach" button is clicked (UC7) (Further details in Section 8.2). Then it waits for actions from the user in order to make decisions. User actions may include:

- Perform a brake
- Perform an acceleration
- Perform a turn
- Perform a lane-change

The system then detects the event and assigns a type and risk level to make other decisions (UC9). The events bar will change to reflect the score of the identified event (UC20) on the screen. The event information will be stored in the local database (UC8). The event details will also be sent to the Pattern Analysis Unit (UC10). Then, feedback decisions will be made

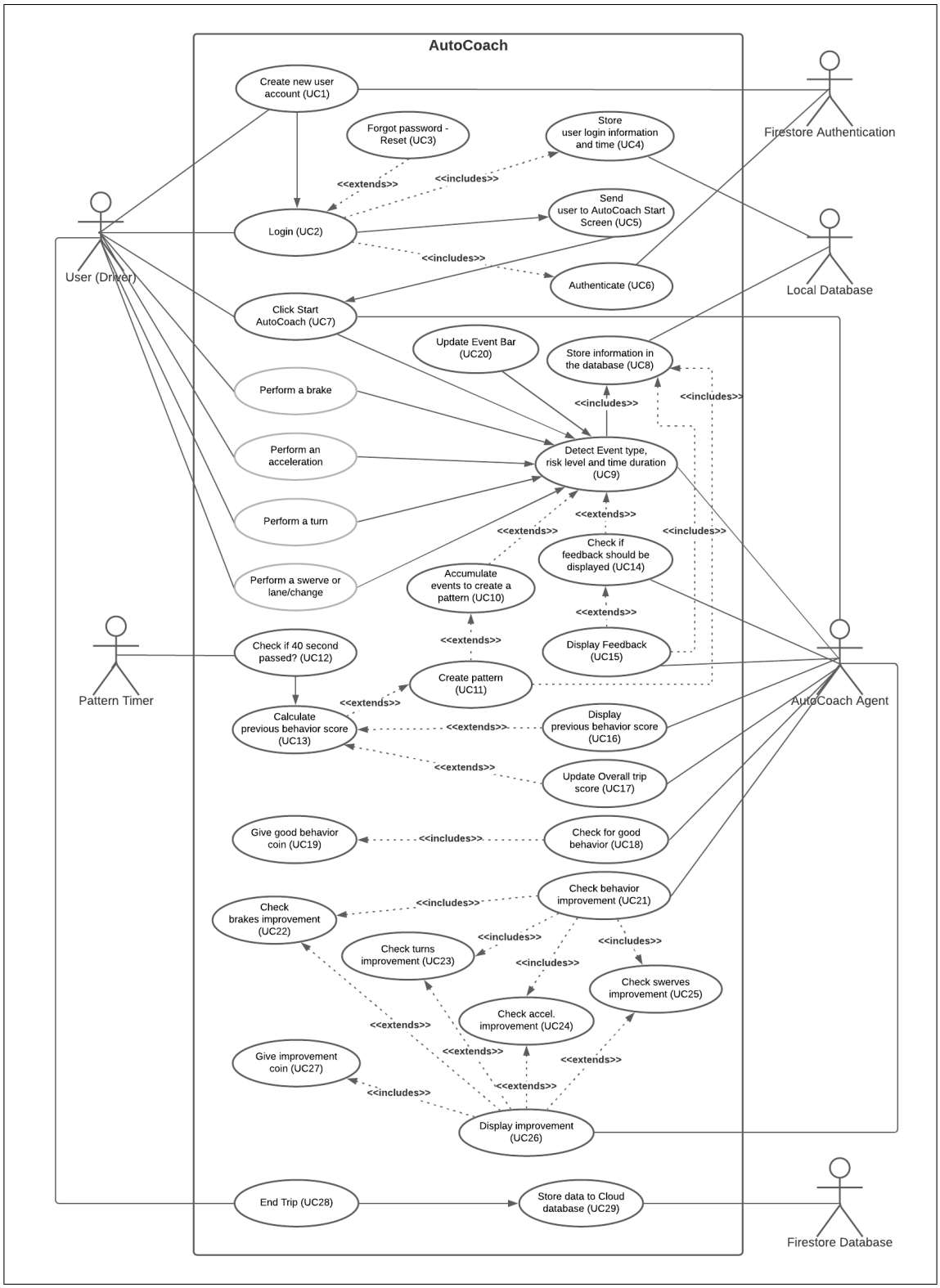


Figure 8.11: AutoCoach use case diagram.

(UC14), and if there is a feedback that should be given, it will be displayed on the screen (UC15) according to the decisions that were made as discussed in the previous sections.

In the Pattern Analysis Unit, once the pattern window duration has elapsed (UC12), create a pattern (UC11) and store its details into the local database (UC8). Then, the pattern score will be calculated in the Scoring Engine (UC13), and the "Previous Behavior" score and the "Trip Score" will be updated on the screen (UC16, UC17). AutoCoach will check for good behavior in order to issue rewards (UC18). Also, if there had been a feedback issued, AutoCoach will check for behavior improvement (UC21). Whether it was an improvement in brakes, accelerations, turns, or lane-changes (UC22, UC23, UC24, UC25), the system will display the improvement notification on the screen (UC26) and will reward the driver by given a rewards coin (UC27) (Further details in Section 8.5).

Once the driver has arrived at his destination, he will click on the "End Trip" button (UC28), and the locally stored data will be sent to the Firebase Database.

8.8 Database Design

AutoCoach's local and Cloud databases are one of the most important pieces that holds all driver's information locally and universally. Figure 8.12 shows the Entity Relationship Diagram for AutoCoach's local database which it's data is also sent to the Cloud where all driver's data are collected. We have designed AutoCoach's database to collect information about:

- Driver
- Trip
- Event

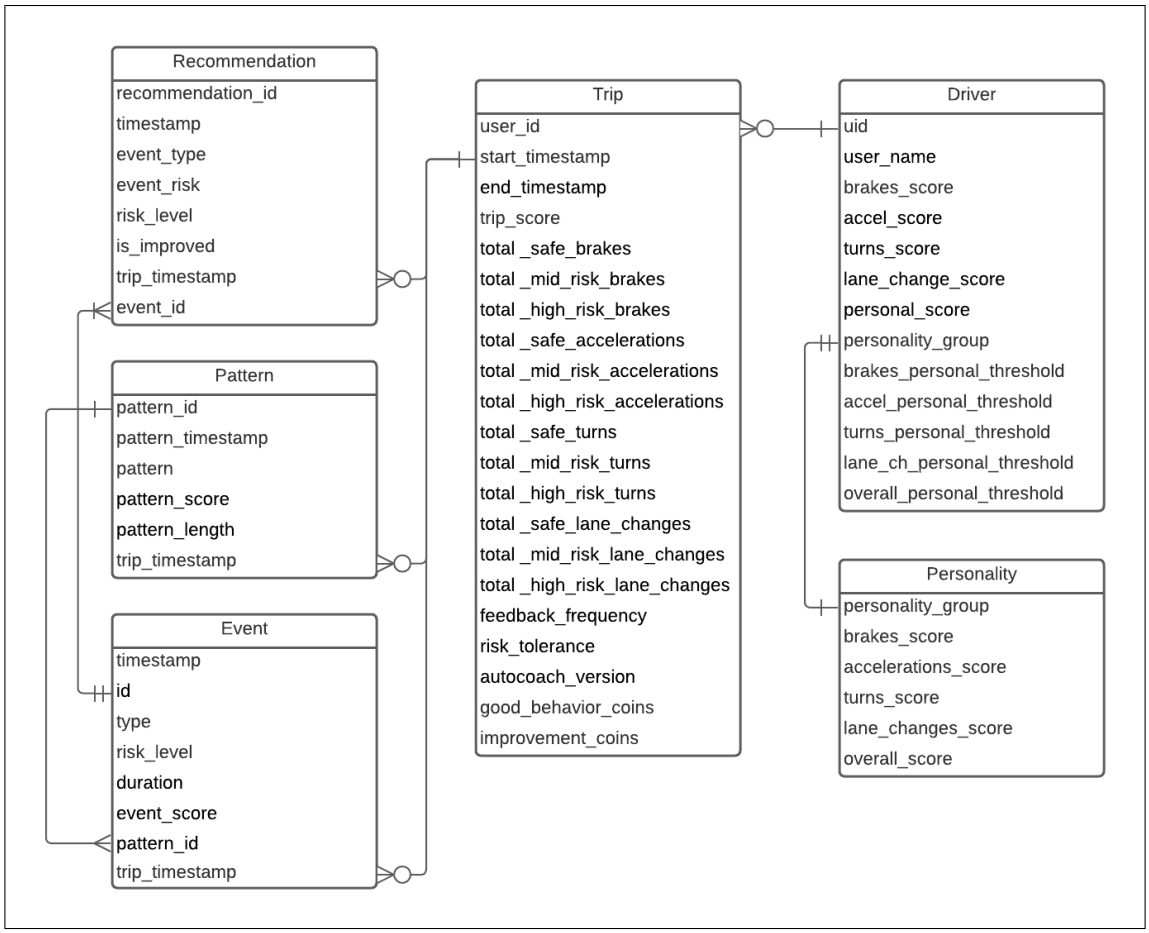


Figure 8.12: Entity Relationship Diagram

- Pattern
- Recommendation
- Personality

Every time the driver logs in, a record is created for the new trip storing its start timestamp and the user's id. The driver's personality information is also pulled from the Cloud database and is stored locally. Because every time the daemon retrains the model, there is a possibility of changing the driver's personality group. Hence, we update this information in real-time to make sure the system is up-to-date. The driver's personal scores are also calculated by the daemon and are updated at every run of the system. When AutoCoach is running, it keeps

track of all detected events, patterns, recommendations. At the end of the trip, AutoCoach calculates the trip scores and total numbers of events, frequencies, risk tolerances, and total numbers of earned coins as listed in Entity "Trip in the Figure. The data is sent to Firebase Cloud Database, where AutoCoach's Daemon wakes up when listening to new data added. The daemon recalculates the driver's personal scores and re-clusters the model if required, and updates the Firebase Database with the new data. Whenever AutoCoach runs again, it pulls the new data from Firebase servers for an updated model.

Chapter 9

AutoCoach v1.0 User Study

In this section, we present a pilot study to evaluate the feasibility and acceptability of AutoCoach. Evaluations used the within-subject mixed design method. This study aims to understand the drivers' level of acceptance of personalized personality-based driving behavior management systems compared to non-personalized systems. Considering drivers' differences and needs, AutoCoach identifies those differences and offers personalized feedback that matches their needs. For the user study, we have two purposes. First, we would like to understand how feedback frequency plays a role in sending more effective feedback to drivers. Second, we would like to identify the strengths and weaknesses of AutoCoach for further improvements.

9.1 Participants Selection

For selecting participants, we use convenience sampling due to time constraints. We have randomly selected three users from each personality group to participate in this study out of the 50 participants who participated in the data collection phases. Those users have

already used the system previously to collect SVM and Personality Analysis data but have not seen the GUI before. This collected data establishes our initial personality model shown in Section 6.2.3. All drivers participating in this study have accumulated at least two hours of driving data. This rule allows us to make sure all drivers are ready for the Personalized Feedback User Study.

Two users did not complete the study, and another two failed to place the phone in the car adequately, thus collected false data. We had them removed from our participants' list. Eight participants are left in the usability study (five males and three females, mean = 36, Std = 3.12). Figure 9.1a displays the differences between the participants, where drivers are arranged from safest to riskiest personality groups, and where User 1 is the safest driver and User 7 is the most dangerous.

(**a**) (**b**)

Figure 9.1: Test participants and groups scores. (**a**) Participants P-scores from safest to riskiest, where User 1 is the safest and User 7 is the riskiest driver. (**b**) Personality groups average scores: extracted from the Personality Model Groups. Details about the personality group model are in Section 6.

9.2 Experimental Setup

For this user study, we have set up AutoCoach to have two different versions to test the acceptance and feasibility of the personalized AutoCoach compared to a version running no personalization. On the user GUI screen, we show two buttons: Version NP and Version P.

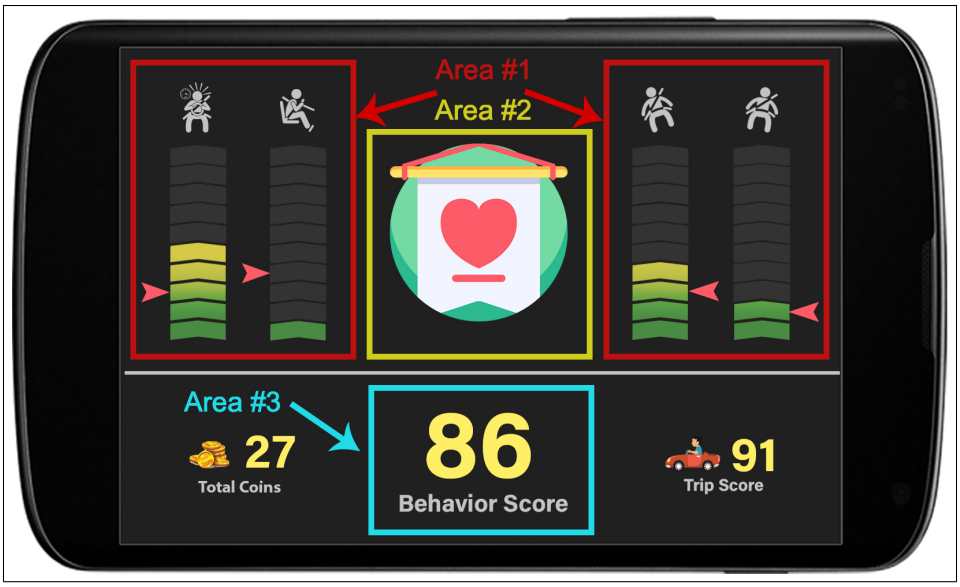


Figure 9.2: Usability test GUI components.

To validate that the drivers have selected the correct model when experimenting, we save the version information with the driver's data on the servers at the end of the trip. In this test, we asked each driver to experiment with both versions. The first version (NP) has the personalization feature turned off. All drivers have equal thresholds for all feedback components ($\delta = 80$). The second version (P) uses the personalized model. Each driver will be treated based on their personal scores (P-score) and assigned group scores. In this experiment, participants were introduced to AutoCoach days before the test to give them time to explore it. We provided the drivers with detailed information about what AutoCoach does and what each component represents. We also provided them with the test task details. Then, we scheduled a time for the experiment for each driver individually. Before starting the test, we verbally explained to drivers the test tasks shown in Table 9.1. Throughout this chapter, we will be referring to the non-personalized version as Version NP and the personalized version as Version P.

Table 9.1: User study task list.

No.	Task List
1	Assume you are going out on a short trip around your neighborhood, drive for 2 minutes safely
2	Observe how the 3 areas are changing shown in Figure 9.2
3	Drive for 2 minutes more aggressive than your normal as if there is some risky event around you or you are avoiding some danger
4	Observe how the three areas are changing while driving
5	Drive for 2 minutes safely again drive as you normally do again as if the danger is gone
6	Observe how the three areas are changing while driving
7	Stop driving as if you reached your destination, and click "End Trip"
8	Repeat the same steps for Version P

9.3 Quantitative Results

9.3.1 Recommendations vs. Improvement Coins

From the results shown in Figure 9.3a,d, we notice that drivers were given much more recommendations in the NP version than in the P version. The riskier the driver, the more feedback offers he seemed to get. When comparing the number of recommendations proposed with the number of improvement coins earned, we find that drivers could not improve their behavior for all given recommendations in the NP version. On the other hand, when we look at the P version results, we can see that the offered recommendations seem pretty consistent among all drivers. Therefore, safer and riskier drivers were offered relatively similar feedback as the system created a personal model for each of them. Furthermore, most drivers can catch up with the number of recommendations offered and earn behavior improvement coins.

Figure 9.3: User study statistics. (**a**) Version NP—recommendations vs. improvement coins. (**b**) Highways—recommendations. (**c**) Highways—earned coins. (**d**) Version P—recommendations vs. improvement coins. (**e**) Local roads—recommendations. (**f**) Local roads—earned coins.

9.3.2 Highways vs. Local Roads

We asked drivers to drive around local roads and repeat the same trip for both versions; however, participants 4, 5, and 7 had to go through a highway for part of their trip. From the

collected data shown in Figure 9.3b,e, we noticed that when drivers went through highways, they were issued more recommendations in the NP version compared to drivers who remained within local roads. This increase is because the higher the speed, the more likely an event is to be considered risky. However, in the P version, the number of recommendations seemed to be consistent. As the P version uses the two levels of personalization, it can decide whether this is a true emergency to offer a recommendation or not.

9.4 Qualitative Results

When a participant completes his experiment, we conduct a semi-structured interview right after to make sure the information is still fresh. We ask the participants 24 questions, followed by a discussion comparing the two versions. The questions covered many aspects, including:

1. Event detection correctness: tests if the system correctly identifies or misses events.
2. Feedback timeliness: tests how timely the were the proposed recommendations.
3. Feedback sensitivity: tests how sensitive the system is in identifying risky behaviors.
4. Risk identification accuracy: are the identified risky events hazardous?
5. Safety identification accuracy: are the detected safe events truly safe?
6. Level of intelligence: asks questions on how a driver rates the intelligence is AutoCoach.

9.4.1 Feedback Timeliness and Sensitivity

As mentioned earlier, we assume riskier drivers need to be offered feedback sooner rather than later while ensuring that those more dangerous drivers' "normal behavior" usually is

worse than those of safer drivers. We asked drivers questions regarding the timeliness of each version. As shown in Figure 9.4a, 67.5% of drivers found that version P was more timely in offering the feedback for brakes, accelerations, and turns, while 37.5% found those were neutral for both versions. Because version P is set, most drivers belonging to groups G_B, G_C, G_D will be getting faster feedback, and the higher the risk group, the quicker the feedback.

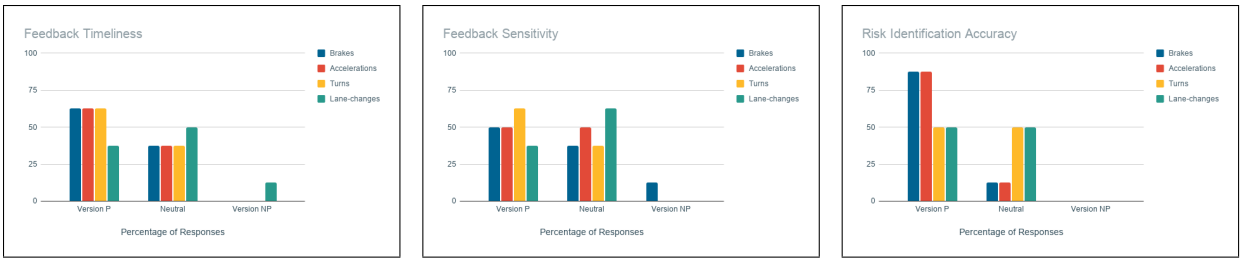
In regards to sensitivity, the same concepts of timeliness apply. Results have shown that participants found that version P is more sensitive than version NP when detecting riskier behavior.

9.4.2 Level of Intelligence

Through the interview, we asked drivers to rate the level of intelligence of version NP and Version P. Results are shown in Figure 9.4e exhibits that all participants highly believe that Version P is more intelligent than NP. We notice that the riskier the drivers, the less they believe Version NP is more intelligent. This belief is because the memory factor is more proactive for higher-risk drivers. Results are shown in Figure 9.4c,d state that drivers also find participant agrees that Version P is better in estimating the risks and safety of their driving behavior.

9.5 Overall Level of Acceptance of Version NP vs. Version P

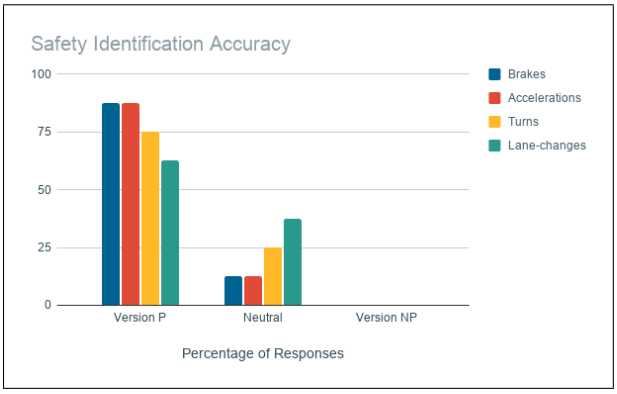
The acceptance level indicates how much a participant accepts AutoCoach as an intelligent driving behavior management application in terms of frequency, timeliness, correctness, sen-



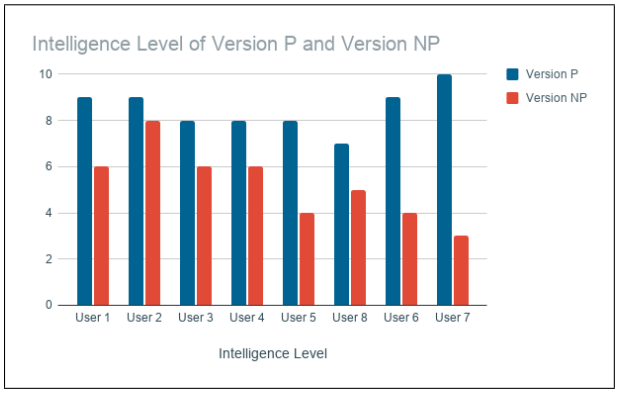
(a)

(b)

(c)



(d)



(e)

Figure 9.4: (**a**) Feedback timeliness. (**b**) Feedback sensitivity. (**c**) Risk identification accuracy. (**d**) Safety identification accuracy. (**e**) Level of intelligence.

sitivity, accuracy, and intelligence overall, and that is considering the feedback components presented in the three areas shown in Figure 9.2. Out of the 24 interview questions, we included the eleven 10-point Likert-scale question ratings, where question (q)'s ratings are averaged and converted to a percentage:

$$Acceptance\ level = \sum_{q=1}^{n} \frac{rating}{n} * 100 \tag{9.1}$$

Questions compare between Version NP and Version P, where "1" indicates that a participant strongly agrees that Version P is more accurate than version NP. In contrast, "10" indicates the opposite. Results in Figure 9.5 show the means of Likert-scale ratings of the 8 participants.

The results have shown that the personality-based model is found to be 61% more accepted

as an intelligent model compared to non-personality based models. Results indicated high acceptance for our assumption that Version P is more intelligent and helpful.

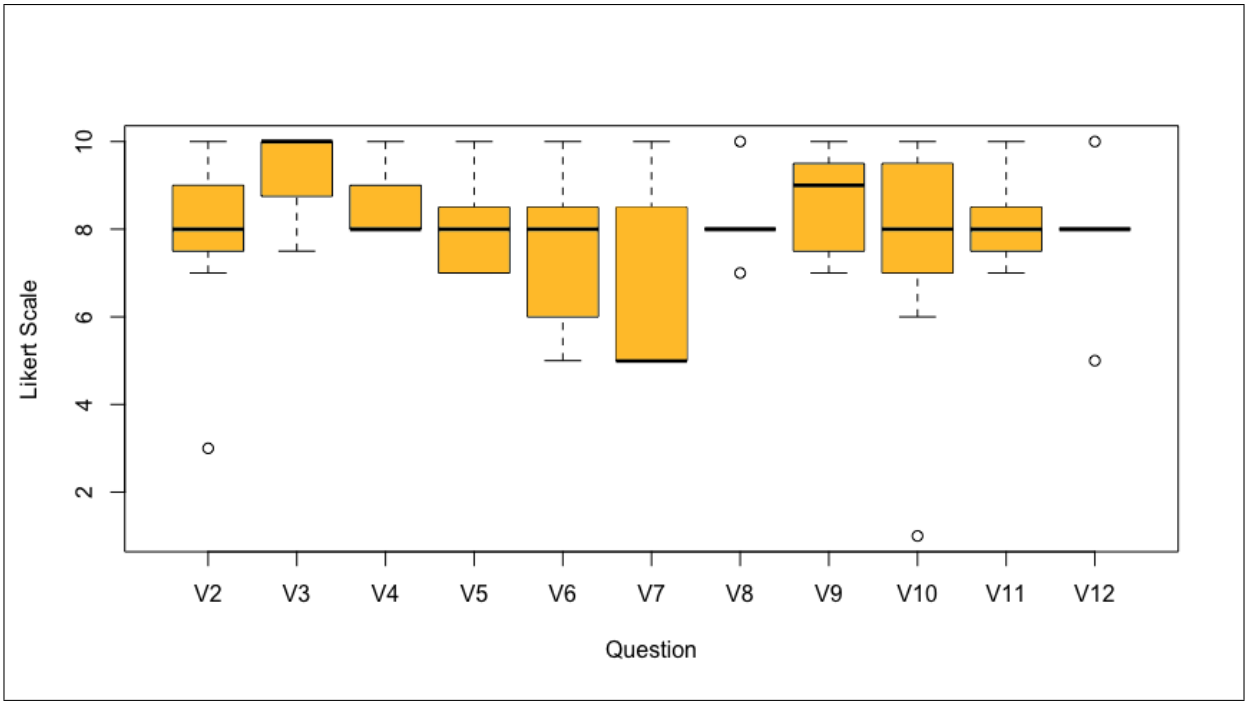


Figure 9.5: Results from questions about the overall level of acceptance of Version P compared to Version NP.

9.6 Success Assessment

We designed six policies to help drivers in better behavior adaptation. The policies allow threshold adjustments based on the driver's driving performance. During the phases of data collection, we exposed some of our drivers to the AutoCoach system without prior knowledge that we are monitoring their driving behavior changes. We have re-clustered the drivers throughout the study to evaluate the historical change in drivers' groups. We select results from three points of time over eight weeks to identify whether behavioral change is significant.

Figure 9.6 shows the 21 drivers' group placements at three different points of time. We notice

that many drives in higher-risk groups have transitioned to less risky groups over time.

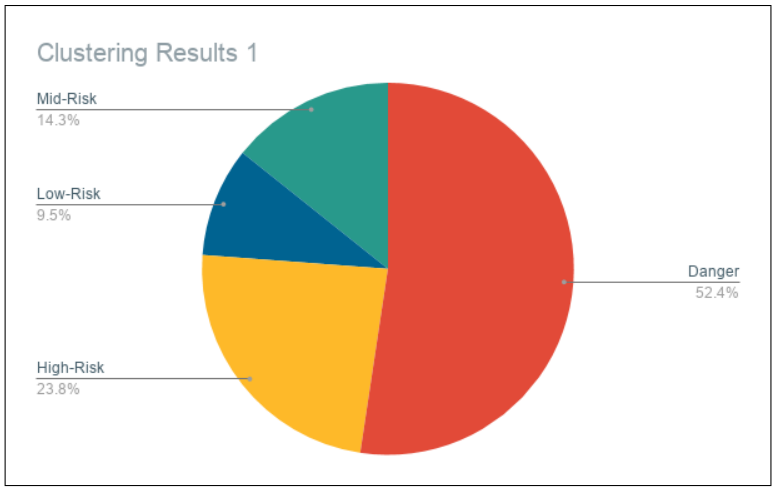
Given the null hypothesis that people will not show behavioral changes at any time, we believe that the policies support drivers to improve their driving behavior under significance level $\alpha = 0.05$. The dataset is structured as a Likert scale ranging from 1 to 4 where drivers' groups are represented as follows: "Worst Drivers" = 4, "Average Drivers" = 3, "Good Drivers" = 2, and "Best Drivers" = 1).

In order to test that the policies have a positive effect on influencing behavior adaptation, a one-way repeated measures ANOVA was conducted to determine whether there was a statistically significant difference in behavior adaptation using AutoCoach's proposed policies over a period of 8 weeks. There were no outliers, and the data violated the normality of distribution at each time point, as assessed by boxplot and Shapiro–Wilk test (p ¡ 0.001, 0.001, 0.01), respectively.

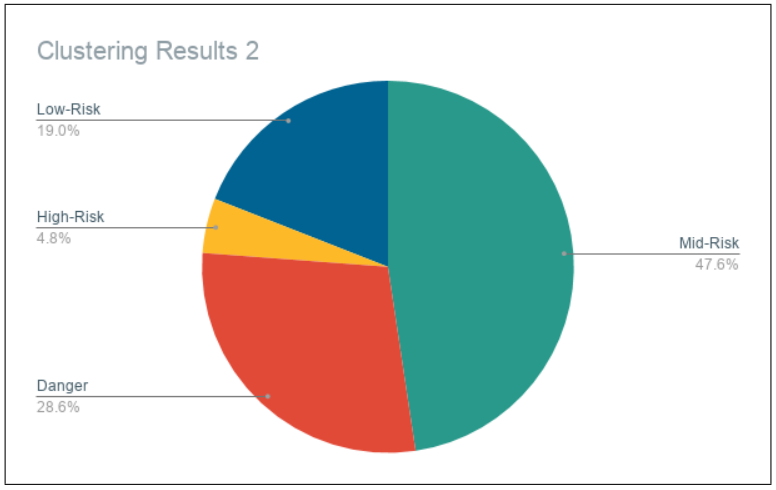
Mauchly's test of sphericity indicated that the assumption of sphericity had been violated, $x^2(2) = 13.103$, $p = 0.001$. Epsilon (ϵ) was 0.664, as calculated according to Greenhouse and Geisser, and was used to correct the one-way repeated measures ANOVA. There has been statistically significant election for behavioral change at the different time points during the study, $F(1.335, 26.698) = 5.231$, $p = 0.022$, $\eta_p^2 = .207$, with group ranks decreased from 3.05 $\pm$ 0.973 at the first point of time to 2.13 $\pm$ 1.121 into the second point of time to 2.14 $\pm$ 0.854 at the third point of time. Results indicate successful behavior improvement.

9.7 User Experience Discussion

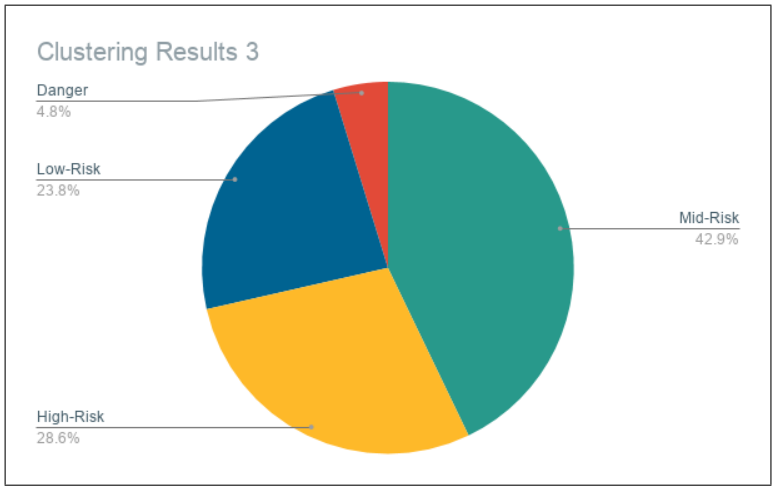
AutoCoach proposed the first feedback-model design that considers the idea of penalization to driving behavior feedback engines. We built a mobile application with our innovative personality-based model and feedback design. Then, we allowed drivers to experience it



(**a**)



(**b**)



(**c**)

Figure 9.6: (**a**) Clustering results 1. (**b**) Clustering results 2. (**c**) Clustering results 3.

and compare it with the same application with the personality option turned off. Results discussed in Section 9.4 show that participants noticed that a personalized version could understand their driving style. During our after-driving discussion, User 4 remarked:

> "When I used the P version, it gave me "behavior improvement" coins after I improved my driving when I was somewhat braking or swerving harsher than usual. It knows it is my driving style. However, in the NP version, it gave me coins all through it."

The "behavior improvement" coins, as discussed in Section 7.5.3, are given to drivers after feedback is initiated and improvement is detected. What User 4 was saying proves that the personalized version knew the driver's standard driving style and only issued feedback and improvement coins when he truly needed the feedback. On the other hand, Version NP has given feedback throughout the trip, not knowing that aggressiveness is part of the driver's typical driving style. As seen in the results shown in Section 9.3.1, the personalization has given much less feedback because it knew what is genuinely a risky behavior and what is not the driver's habit and personality group assignment.

In regards to the feedback sensitivity and frequency, User 2 stated:

> "Version P has a less aggressive screen with fewer feedback and outputs. Version NP does give too much information. It feels like version P does not consider everything, and it understands how I usually drive."

In our design, sensitivity is defined by what the system considers risky. In version NP, the system knew events are dangerous when they exceeded the pre-set threshold. However, participants disagreed that this is an actual risk. While in the personalized version, some events could have been seen as risky, but they were the drivers' normal as part of the individual behavior. As User 2 said, the system understood how they usually drive.

In terms of feedback timeliness, we designed the feedback to be more proactive for higher-risk drivers. User 7, assigned to the riskiest personality group, sensed these features and stated:

> "Version P was faster in offering recommendations. It offered coins in a more timely fashion. I felt it more accurate and was able to understand my driving better."

In AutoCoach, we purposely delay feedback for safer drivers while sending it much quicker to riskier drivers assuming that they exhibit a high risk and need to be notified. In comparison, safer drivers should be treated less aggressively. From the interview and discussion with drivers, we found that drivers sensed the personalization by stating that Version P knew who they are as drivers.

As we have mentioned earlier, this study's primary purpose was to understand the level of acceptance of personalized feedback compared to non-personalized ones. AutoCoach has successfully gained the trust of drivers as we hoped to achieve. The frequency and timeliness of feedback play a role in creating more perceived intelligence. Lowering the amounts of feedback and giving it only when truly needed reduces drivers' stress caused by frequent feedback. When we have assigned different policies to drivers, we found that riskier drivers have noticed AutoCoach more and were more grateful for the feedback offered by AutoCoach. This acceptance could help them follow the feedback and adapt to newer and safer habits with time. AutoCoach has been shown to give a successful feedback design. We, therefore, believe that the personality-based feedback design can be applied in many other fields to provide a more pleasant UX.

Chapter 10

AutoCoach v1.1 and User Study

After conducting the first study, we identified some problems we believe improving them will enhance the performance and usability of AutoCoach. We found that scoring on local roads was lower than expected, while unnecessary feedback was issued to drivers. On the other hand, some drivers experience low or no feedback when driving on highways, while we believe that AutoCoach should be more responsive because of how quickly risky events happened there.

10.1 Improved AutoCoach Design

Therefore, we decided to improve the version of AutoCoach, by allowing AutoCoach to identify the road type and road condition and change the feedback strategy accordingly. The improved version includes the design model shown in Figure 10.1. AutoCoach continuously detects the speed through the GPS readings, where reading is recorded every 5 seconds. We use a moving average window that passes through every 24 readings. The purpose of this study is to compare v1.0 of AutoCoach with the upgraded v1.1, wherein v1.1 we have added

new features that automatically change the applied policy based on the road type and road conditions.

There are four possible scenarios in our model:

1. Clear Local Road
2. Clear Highway
3. Busy Local Road
4. Busy Highway

We assume that local roads are those roads where drivers could drive at lower speeds and could commit more brakes, accelerations, turns and lane changes than they would on highways. The initial state when AutoCoach starts is set to Clear Local Roads.

In our original version of AutoCoach v1.0, each driver has a policy assignment based on his personality group decision. In this version (v1.1) we assume that assigning a safer strategy for the drivers will allow improved scoring and less responsive feedback; therefore, we set the strategic decision to the following:

1. Clear highway: apply Strategy 2 - policy 5 or 6 (high proactivity) - the drivers are most likely braking and accelerating lightly. The drivers who brake harder or accelerate harder tend to either be reckless or angry. Therefore applying memory factor 4 will proactively respond to aggression.
2. Clear local road: apply Strategy 1 - Policy 1 or 2, because, in this model, we already had seen issues when drivers complained of too much feedback when they did not really drive risky, but due to too many events, the system was highly responsive to driving events. Also, update the pattern scoring engine- because long patterns are scored low, but this is the normal driving for some local roads.

3. There are no changes for busy highways and busy local roads. Apply the same policy in terms of responsiveness and sensitivity. However, the scoring engine should be updated for busy and free local roads due to the high number of events in short periods.

For selecting the participants, we have requested the same users who have participated in the previous study to join this study. We have sufficient information about each participant allowing the personality-based model to run for each of them.

In this study, we would like to achieve the following:

1. Increase number of earned coins for safer drivers when driving on clear local roads.
2. Decrease the number of recommendations the drivers receive on clear local roads.
3. Improve the scoring when drivers drive on clear local roads.
4. Higher responsiveness to risk and increase the number of recommendations the driver receives on a clear highways.

The scoring engine factors described in Section 7.1 are adjusted whenever the system detects a clear local road. We have set the four scoring factors to: $R1 = 100$, $R2 = 80$, $R3 = 60$, $R4 = 40$, respectively, from the safest to highest risk behaviors. This adjustments results in higher scoring results for patterns to solve the problem drivers have mentioned about lower scores than expected.

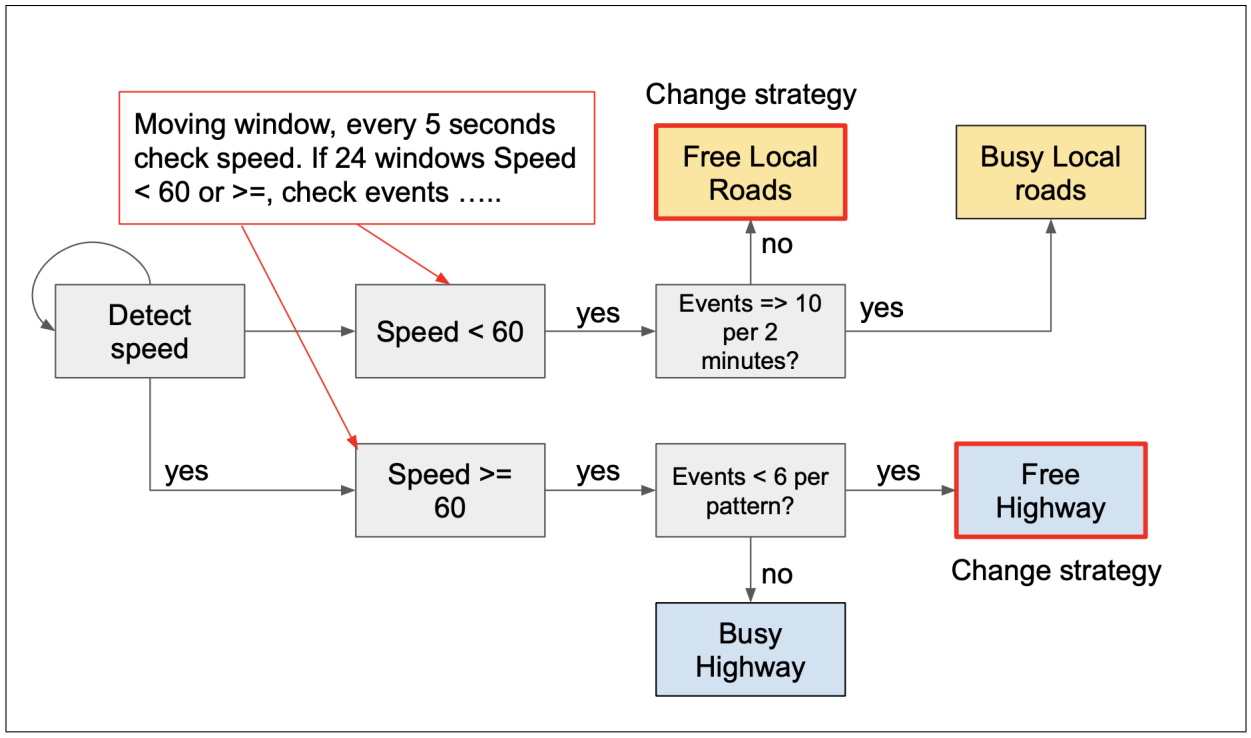


Figure 10.1: Improved AutoCoach Design

10.2 Experimental Setup

10.2.1 Experiment Task List

We have verbally explained the following task list to the participants and sent them a copy. We informed the users that in this study, we would like to compare the old version of Personalized AutoCoach with an upgraded version of it. There are four components to this experiment. The participants were required to test two versions of the AutoCoach application (Version 1.0 and Version 1.1). Participants have to test each application twice: two times on local roads and another two times on highways. For local roads: each test will take approximately 6 minutes. After that, participants will be asked some interview questions to be answered and hear more about their experience. Then, participants can do the two highways tasks with each, followed by an interview. We set the time duration to complete the tasks to be one week. The tasks are listed as follows:

Task 1: Driving on a local road using Version 1.0

We would like you to drive on Local Road (somewhere around where you live or the main roads of your city where turns, bumps, or traffic lights exist). Run Version 1.0 and drive for a total of 6 minutes: 2 minutes safely, 2 minutes risky, and 2 minutes safely again. Notice the feedback and scores while driving only when you feel it is safe to look at the screen. Right after completing the task, please fill out the interview questions for Task 1.

Task 2: Driving on a local road using Version 1.1

We would like you to drive on Local Road (somewhere around where you live or main roads of your city where turns, bumps or traffic lights exist). Run Version 1.1 and drive for a total of 8 minutes: drive for 2 minutes until the text shown in the top center of the screen changes from "Road condition" to either "busy/clear highway or local road, then, 2 minutes safely, 2 minutes risky, and 2 minutes safely again. Notice the feedback and scores while driving only when you feel it is safe to look at the screen. Right after completing the task, please fill out the interview questions for Task 2 from the usability test form.

Task 3: Driving on a highway using Version 1.0

We would like you to drive on a Highway (Maybe sometime when you are going to work or any other trip as long as you get to drive on a highway for at least 10 minutes. Run the application, and drive as you normally do, and then about halfway, drive a bit more aggressively than you normally do while maintaining safety. Notice the changes of feedback and scores on the screen. Please fill out the interview questions for Task 3 right after completing the task.

Task 4: Driving on a local road using Version 1.1

We would like you to drive on a Highway (Maybe sometime when you are going to work or any other trip as long as you get to drive on a highway for at least 10 minutes. Run Version 1.1, for the first two minutes, you should notice the top center text showing "Road Conditions". After driving for about two minutes, the road conditions will change to either "bust/clear local road or highway. Then you may drive as you normally do. Sometimes halfway through, please drive a bit more aggressively than you normally do for a couple of minutes only when you feel it's safe to do so. We just want to see how the system responds to your change in driving habits. Notice the feedback and scores while driving if you feel it is safe to look at the screen only. Right after the trip, please fill the Task 4 page from the usability test form.

Task 5: Interview

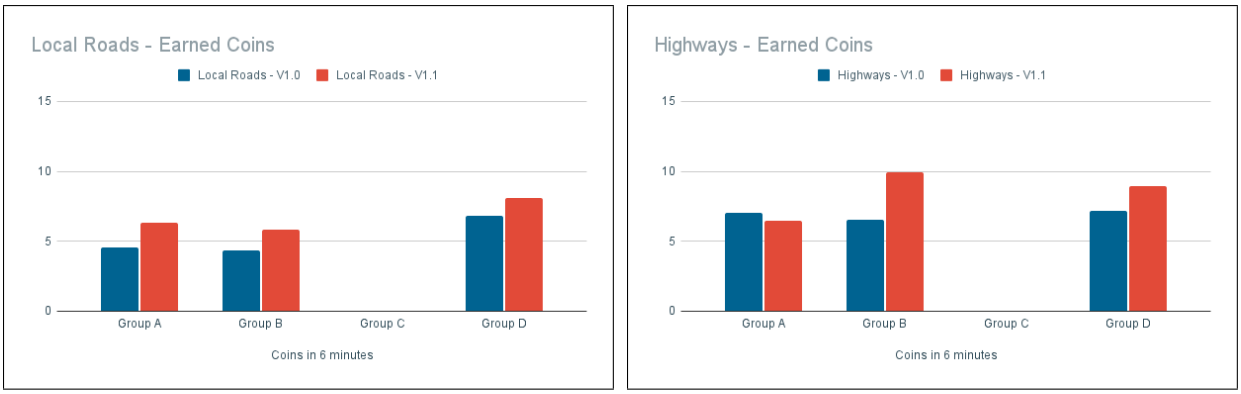
After you are done with the experiment, please meet me so I can ask you some questions about your overall experience. During the interview, you will be asked questions to compare the two versions of AutoCoach. This information will help us test the usability of AutoCoach and allow us to improve it to understand the user needs better.

10.3 Quantitative Data Analysis

In this dataset, some participants failed to complete the assigned tasks correctly, therefore, we do not have data for participants in Group C. All data shown in this study will compare drivers belonging to groups GA, GB, and GD only.

10.3.1 Earned Coins Results

Participants have driven on the same road for both version 1.0 and version 1.1 for a fair comparison in the local roads experiment. Due to the differences between drivers and the locations where they have done the tasks, between group comparison is not a reliable measure, therefore, comparisons will be made based on the change within group (e.g., Group A v1.0 and v.1.1 experiments). For the local roads experiment, results shown in Figure 10.2(a) have shown an increase of 37.52%, 33.35%, and 18.71% in earned coins for groups GA, GB, and GD, respectively. This increase confirms that AutoCoach has successfully improved the rewarding mechanism on local roads, which seemed to be a problem the participants mentioned in the previous version. On the other hand, drivers also have shown an increase of 51.12% and 23.98% for groups GB and GD when driving on highways (Figure 10.2(b)).



(**a**) (**b**)

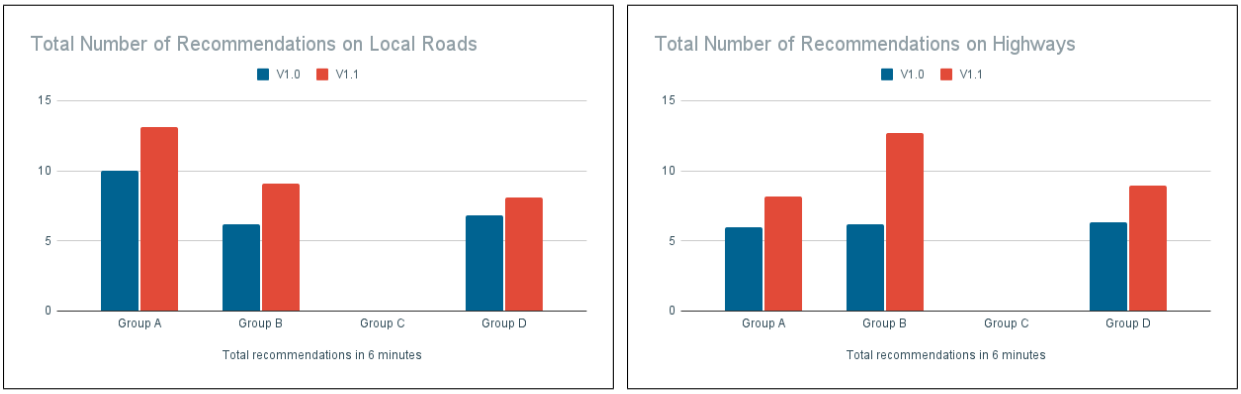
Figure 10.2: Total number of earned coins in each group within 6 minutes trip. (**a**) Local Roads. (**b**) Highways.

10.3.2 Recommendations

Drivers' trips on highways varied in length. Therefore, to fairly compare the data, we calculated how many recommendations on average the driver got per minute. We multiplied it by six minutes to get the number of recommendations drivers received in a six-minute

trip. Referring to Figure 10.3(a) results indicate an increase in recommendations of 31.13%, 46.08%, and 18.71% for groups GA, GB, GD, respectively, in recommendations when driving on local roads. When driving on highways, there was also an increase of 36.35%, 104.46%, and 41.7%, as shown in Figure 10.3(b). Based on the improved model we designed, we expected a decrease in the number of recommendations when driving on local roads. However, the outcome did not meet our expectations. We found out that due to the adjustments in the strategy, where safer driving policies were applied, it did not work with riskier drivers resulting in further recommendations. While for highways, the upgrade has successfully met the expectations. We will discuss the qualitative results to back up the results we have obtained in the Section 9.4.

When driving on local roads and highways, we found rewards coins average increase of 29.8% and 25.03% in v1.1 compared to v1.0. We also found an average increase of 26.4% in the number of given recommendations when driving on highways.

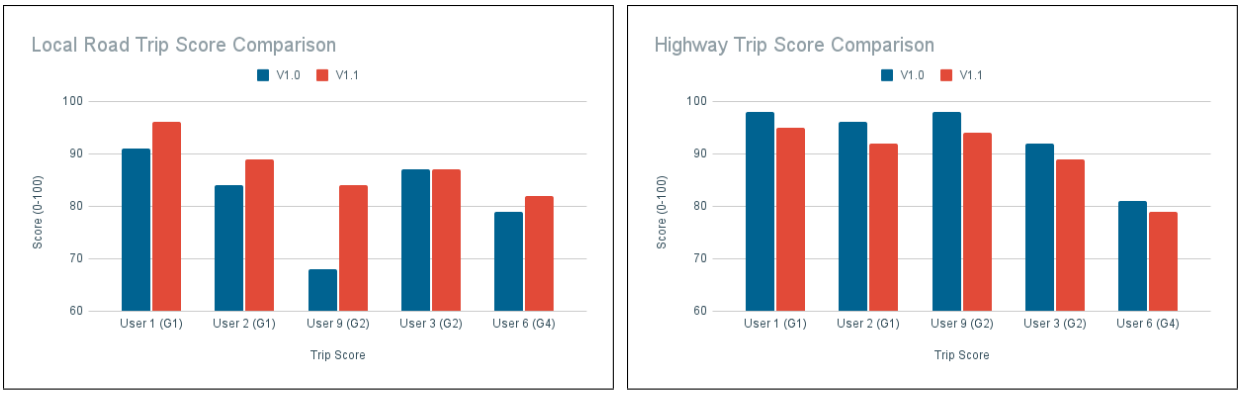


(**a**) (**b**)

Figure 10.3: Total number of recommendations in each group within 6 minutes trip. (**a**) Local Roads. (**b**) Highways.

10.3.3 Scores Results

According to our previous study, drivers indicated that the scores did not seem fair when driving on local roads, and safer drivers were getting fairly low scores for their safe driving.



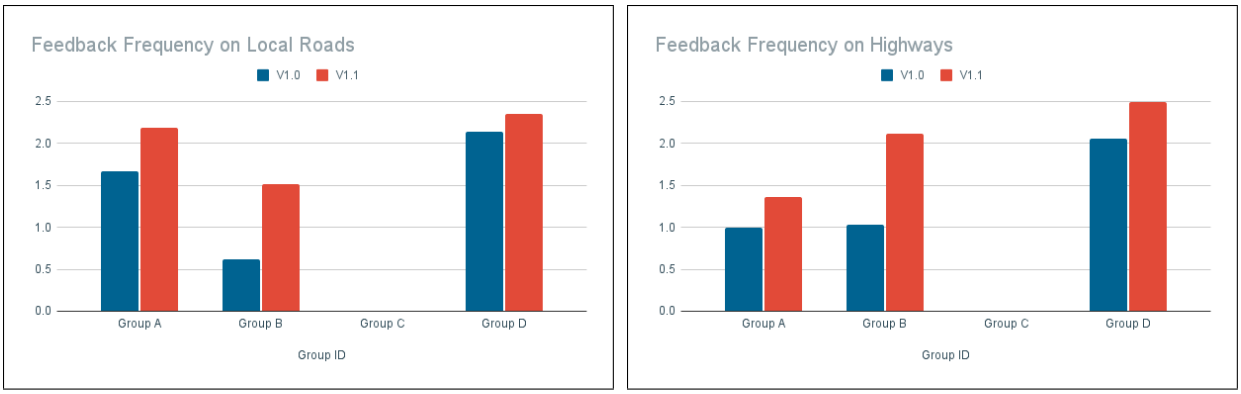
(**a**) (**b**)

Figure 10.4: Trip scores for participants when driving on local roads and highways (**a**) Version 1.0. (**b**) Version 1.1.

Therefore, in version v1.1, we have adjusted the scoring engine parameters. Whenever drivers are driving on clear local roads, the strategy is adjusted. The scoring engine parameters will be adjusted. According to the collected trip scores shown in Figure 10.4(a), local roads scores showed an overall increase of 7.4% in v1.1 compared to v1.0. While on the other hand, scores have a 3.1% decrease in v1.1 comparing to v1.0 when driving on highways. This is a healthy decrease in scores as we want the system to be more responsive to risk and thus, increase recommendations and decrease scores.

10.3.4 Feedback Frequency

Feedback frequency is a measure used to indicate the risk associated with driving behavior; the more feedback means, the more risky behavior is. In our previous study, drivers indicated that they received too much feedback on local roads and too little on highways. Therefore, this version is meant to fix this problem. According to the study results displayed in Figure 10.5(a), there shows an increase in the amount of recommendations. Based on the improved model, drivers who drove on clear local roads, whether they were safe drivers or risky, will all be assigned to the safest group. Therefore, scoring criteria will be adjusted to offer higher scores. With that, we should expect less recommendations, however, there had



(**a**) (**b**)

Figure 10.5: Feedback frequency (**a**) Local Roads. (**b**) Highways.

been an increase in number of recommendations (shown as light blue dots on the figures). When we look further into the example presented in Figure 10.6, which shows the series of scored driving behavior pattern in one of the recorded trips, where the red line represents the memory factor-based scores. We find that the threshold (Yellow line) that is responsible for triggering the feedback was adjusted along with the scoring engine memory factor. This resulted in higher expectations, and some behaviors which would not be considered risky in v1.0 started to be considered as risky in v1.1 resulting in more feedback. AutoCoach did not reduce the feedback amount in this case but increased it, while it still improved the scoring based on the results discussed previously in Section 10.3.3.

On the other hand, AutoCoach v1.1 has successfully increased the number of issued recommendations when driving on highways and resulted in a more responsive feedback system in all personality groups. The new improvement allowed assigning a more responsive policy to all drivers, allowing higher sensitivity to risk and more frequent feedback. Figure 10.7 shows an examples of another trip, where the blue line represents the memory factor-based scores. We can compare the right column of figures to the left ones and see that, for each group, the number of recommendations has increased. We have completed the analysis of our results by hearing from participants. Further supporting data on the feedback frequency is discussed in Section 9.4.

Figure 10.6: Local Roads Feedback and Scoring Example. (**a**) Group B–v1.0. (**b**) Group B–v1.1. (**c**) Group C–v1.0. (**d**) Group C–v1.1. (**e**) Group D–v1.0. (**f**) Group D–v1.1.

10.3.5 Risk Tolerance

Risk Tolerance is a variable representing the system's willingness to tolerate risky behaviors. It is calculated by the average differences between the personal threshold and the original

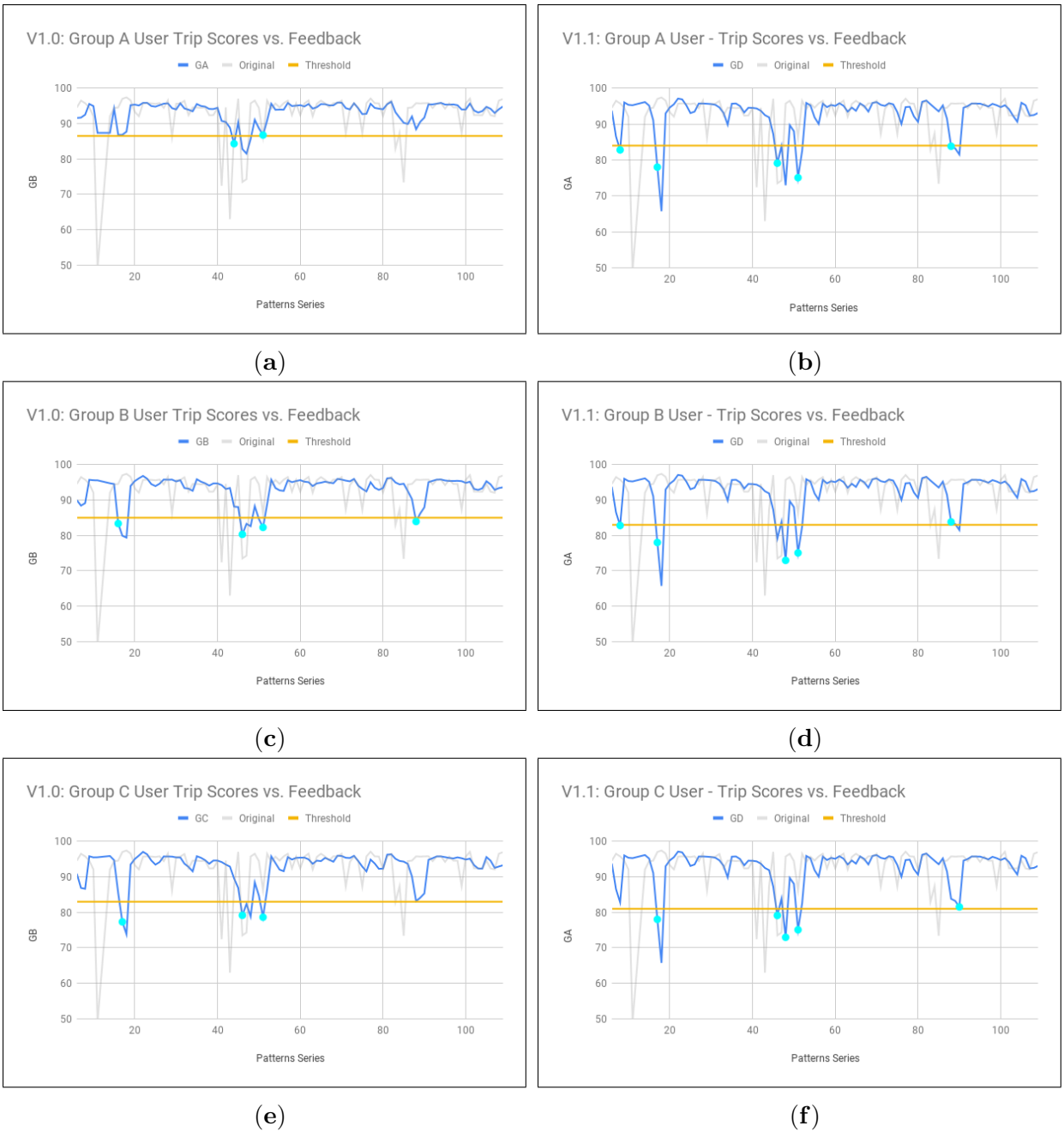
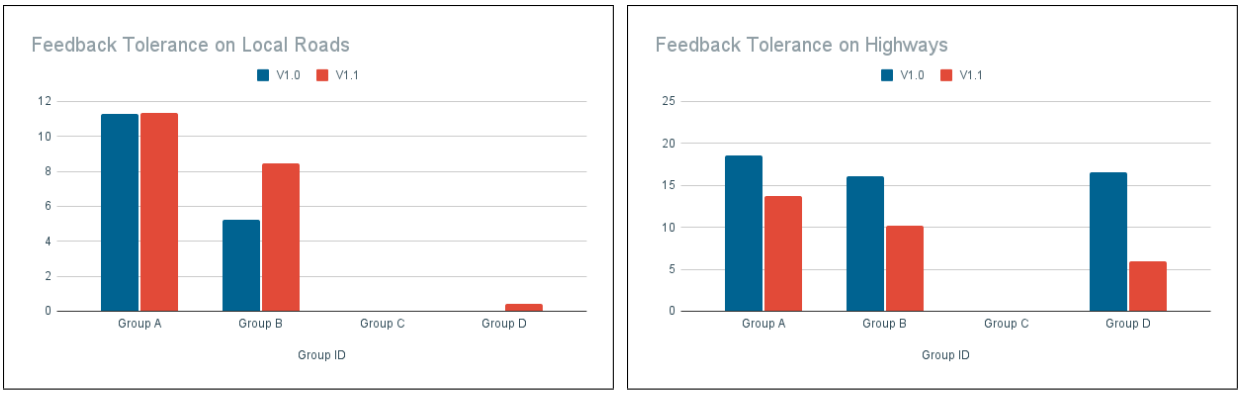


Figure 10.7: Highways Feedback and Scoring Example. (**a**) Group B–v1.0. (**b**) Group B–v1.1. (**c**) Group C–v1.0. (**d**) Group C–v1.1. (**e**) Group D–v1.0. (**f**) Group D–v1.1.

score at which events have triggered feedback. The higher the value, the more tolerant the system is on local roads. For example, if the personal threshold $P - Score = 90$ and *Event Score = 80*, then, *Risk Tolerance = P-Score - Event Score = 10*. If a policy is adjusted during a trip due to a detected highway, P-Score could be lower (e.g., P-Score=80), resulting



(**a**) (**b**)

Figure 10.8: Risk Tolerance (**a**) Local Roads. (**b**) Highways.

in Risk tolerance $= 5 < 10$. This means that the currently applied policy allows the system to tolerate risky behavior less than previously. When driving on highways, the system becomes more proactive and more responsive to risky driving. Risk tolerance allows us to identify the success of the new adjustments by comparing its results on v1.0 and v1.1.

Results in Figure 10.8 show the average risk tolerance for all triggered feedback. When driving on Local Roads as shown in Figure 10.8(a), Group A drivers have not shown much difference due to no change in policy, while group B shows that v1.1 created higher tolerance compared to v1.0. For Group D, fewer recommendations were triggered due to driving at low events road; however, v1.1 slightly increased risk tolerance. We noticed that for riskier groups, risk tolerance is generally tolerated less than safer groups. While when driving on highways, we expect the system to tolerate risky behavior less. Results in Figure 10.8(b) show that Risk tolerance has decreased for all risk groups in v1.1 compared to v1.0, while risk tolerance was gradually decreased as we move from safer to riskier groups (from GA to GD) in general.

The new version of AutoCoach has successfully given increased risk tolerance when driving on local roads, reducing the amount of produced feedback. While it also has successfully reduced the risk tolerance when driving on highways creating a more responsive system to

detect risky situations quickly.

10.4 Qualitative Data Analysis

We have interviewed each of the participants after experimenting with both versions on local roads and highways to collect qualitative data. The qualitative data is then analyzed to back up the system-collected quantitative data.

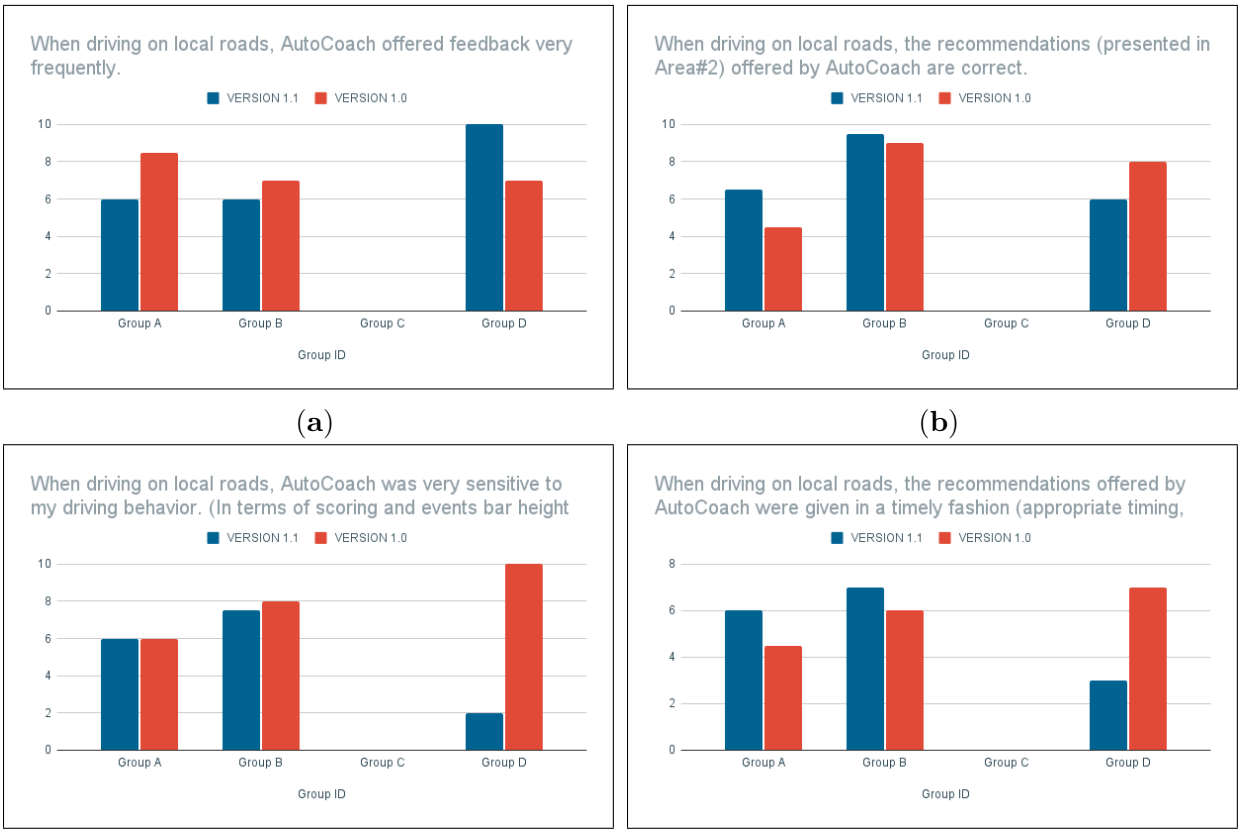
10.4.1 Local Roads Experiment

We asked the drivers to rate the system components on a scale from 1-10, where 1 indicates totally disagree, and 10 indicates totally agree. They were covering the following four components:

1. Frequency of feedback: when driving on local roads, AutoCoach offered feedback very frequently.
2. Correctness of feedback: when driving on local roads, the recommendations offered by AutoCoach were correct.
3. Sensitivity of the system: when driving on local roads, AutoCoach was very sensitive to your driving behavior in terms of scores and events bar changes (events scores).
4. Timeliness of feedback: when driving on local roads, the recommendations offered by AutoCoach were given in a timely fashion (not late or too early).

Throughout this study, we will refer to drivers in groups GA and GB as "safe drivers" and drivers in groups GC and GD as "risky drivers".

Feedback Frequency



(**a**) (**b**)

(**c**) (**d**)

Figure 10.9: Local roads: personality-based users point of view. (**a**) Frequency. (**b**) Correctness. (**c**) Sensitivity. (**d**) Timeliness.

According to the results shown in Figure 10.9(a), safer drivers found that version 1.1 offers less frequent feedback than v1.0. When we look back at our results from the system, we find it contradicting (Section 10.3.4). When we asked drivers about why they believe so; the answers from drivers who belonged to group GA tells us they must not have noticed the change as they had the same policy running in both experiment:

> "The score is barely changing. I don't see difference."

Participant 9 from group *GB* said:

"I find v1.1 the same as V1.0 or a little bit faster."

While participant 6, who believed that v.1.1 offered more feedback said:

"I noticed that version 1.1 gives feedback faster especially on safe driving."

From the interview results, it seems that drivers in GB and GD see that v1.1 offered more frequent feedback, while GA did not notice the difference, which supports the system generated results (10.3(a)) with a possible error rate from the human's perspective when comparing two similar versions.

Feedback Correctness

In terms of correctness (Figure 10.9(b), safer drivers found AutoCoach more correct in v1.1 that in v1.0, while riskier drivers said that v1.0 is more correct that v1.1. When looking deeply, when we ask driver 6 from group D, why he believes v1.0 is more correct said:

"Version 1.1 is slower and soft on giving coins."

And that is true for a driver who's been using Policy 5 or 6, and getting a more responsive service, switching down to GA's policy is definitely a big shift. While other drivers agreeing that v1.1 is more correct said:

"Because what I saw on the screen matched my driving pattern more."

Another participant said:

"When I drove risky, it immediately recognized my risky behavior and once I switched to safety, it identified that and rewarded me."

Risk Sensitivity

Regarding sensitivity, majority of participants found that AutoCoach v1.0 and v1.1 behaved about the same. Participant 1 said:

> "Sensitivity barely responds to change in driving behaviour but its more accurate than version 1.0 which is overly sensitive especially to turns and brakes."

Sensitivity to what specific events vary from one driver to the other depending on how risky that specific behavior is considered for that specific user. Therefore, some users might notice the system is more sensitive to a driving event but not another. However, participant 6 indicated that the identifying behaviors sounds more correct, where he said:

> "Version 1.0 was better at detecting risky/safe behavior."

For riskier drivers driving on local roads, the system will become less responsive due to the higher risk tolerance based on the new strategy update. For specific behavior it sounds like, riskier drivers preferred the more responsive system, while overall behavior and scoring improved from their perspective.

Feedback Timeliness

Results shown in Figure 10.9(c) state that safer drivers find version 1.0 is less timely in issuing feedback. In theory, allowing setting a safer memory factor results in higher risk tolerance and less timely feedback. However, with the memory factor adjustments and personal scores adjusted too, results in higher risk tolerance allowing taking a bit less time to issue feedback. When we further asked drivers about what they think about it. A driver from group GB said:

"The feedback timing sounds little but better than 1.0 in my perception."

While another driver from group GB mentioned:

"Because I noticed the change on the screen with each movement on the road."

This supports the system results and the claim that the feedback timeliness is higher for version 1.1. On the contrary, a driver from group GD said:

"Version 1.0 is a bit faster with feedback."

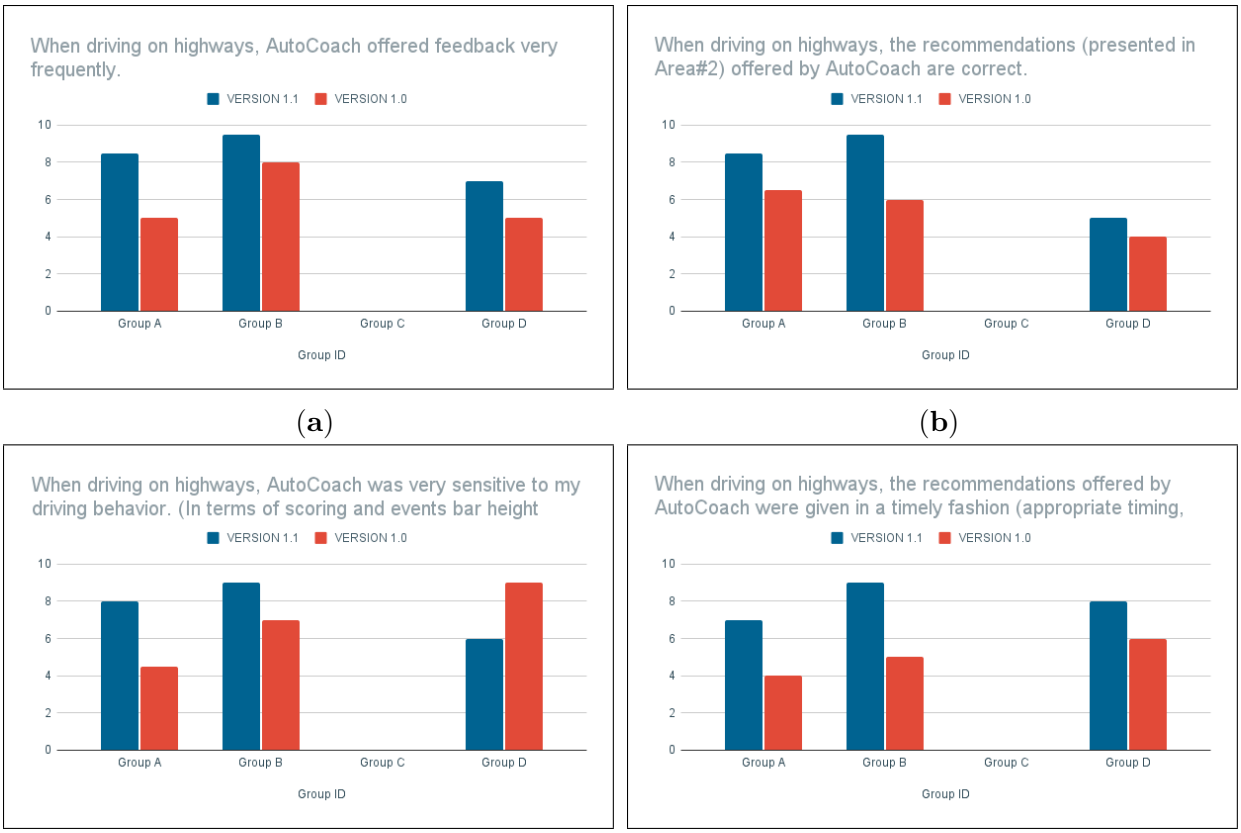
And this is true for riskier drivers, v1.0 will sound more timely as it will issue more feedback because drivers in GD are naturally riskier than others. Drivers in GD though were not happy about the more feedback, while they acknowledge better scoring and understanding of the overall behavior.

10.4.2 Highways Experiment

We have interviewed drivers about their highway experience with AutoCoach and asked about the same components we used in the Local Roads Experiment (Section 10.4.1). Interview results from the highways experiment sound even more successful that in the local road experiment. In this section will will look into the results regarding the frequency, timeliness, correctness and sensitivity.

Feedback Frequency

According to the quantitative results obtained from the system, there has been shown an increase in the feedback frequency for all personality groups when driving on highways (Fig-



(**a**) (**b**)

(**c**) (**d**)

Figure 10.10: Highways: personality-based users point of view. (**a**) Frequency. (**b**) Correctness. (**c**) Sensitivity. (**d**) Timeliness.

ure10.5(b)). We backup this information by testing the usability of the system through this experiment and have got some more feedback from drivers regarding those measures. We asked drivers to rate the feedback frequency of the system for both v1.0 and v1.1. Interview results shown in Figure 10.10(a) state that there had been an increase in feedback frequency for all personality groups. Drivers from different personality groups supported their claims by stating the following:

> Participant 1: "I noticed that v1.1 offered more feedback compared to v1.0. and it makes it feel more intelligent."
>
> Participant 4: "Both versions are about the same but v1.1 slightly better in terms of how much feedback it's offered."

Participant 9: "It gave me feedback with even the slight movements."

Participant 6: "Gives feedback faster."

Feedback frequency have actually increase as we have hoped in the improved version. Drivers found v1.1 system more responsive to their driving behavior and felt it is more intelligent as they said. Both qualitative and quantitative results prove that feedback frequency have been noticed and drivers have found it to be better than v1.0.

Feedback Correctness

In terms of feedback correctness, Figure 10.10(b) shows that v1.1 sounded more correct to them in terms of the recommendations issued for their risky driving behavior. Drivers in safer groups said:

"Version 1.1 is more responsive, better feedback, better scoring."

"It is more sensitive to my driving and gave better recommendations."

The safer drivers in v1.0 have had much less feedback and very high scores even for their risky behavior. Which switching to the new strategy that makes the system more responsive and have stricter rules when driving on highway, safer drivers found it to be a better system.

Participant 6 fro the riskier group mentioned that AutoCoach has given them more feedback but they assume the scoring should have been much worse from their perspective. However, the scoring criteria did not change for highway driving. Based on the results, the overall feedback correctness of AutoCoach v1.1 outperforms v1.o.

Risk Sensitivity

Results shown in Figure 10.10(c) indicate that safer drivers have seen the system to be more sensitive to their driving behavior, while riskier drivers ranked the sensitivity of v1.1 to be lower. In actual system setup, we think the sensitivity should be about the same as Group D drivers don't really have much change in their policies and feedback rules. When we further asked about it, the participant said:

> "They both felt of similar sensitivity."

Which support our claim that the system must not change for riskier drivers as they are already being treated with the most responsive policy. Safer drivers mentioned that v1.1 responds better to their risky driving:

> "Score responded better to aggressive driving but I feel its too sensitive to turns."

Feedback Timeliness

When looking into the interview results about timeliness, Figure 10.10(d) shows that all drivers found v1.1 to be more timely than v1.0, while safer drivers found it highly more timely than v1.0.

A driver from the group GA said:

> "The first version barely gave any feedback, while the second version offered accurate feedback but a little delayed."

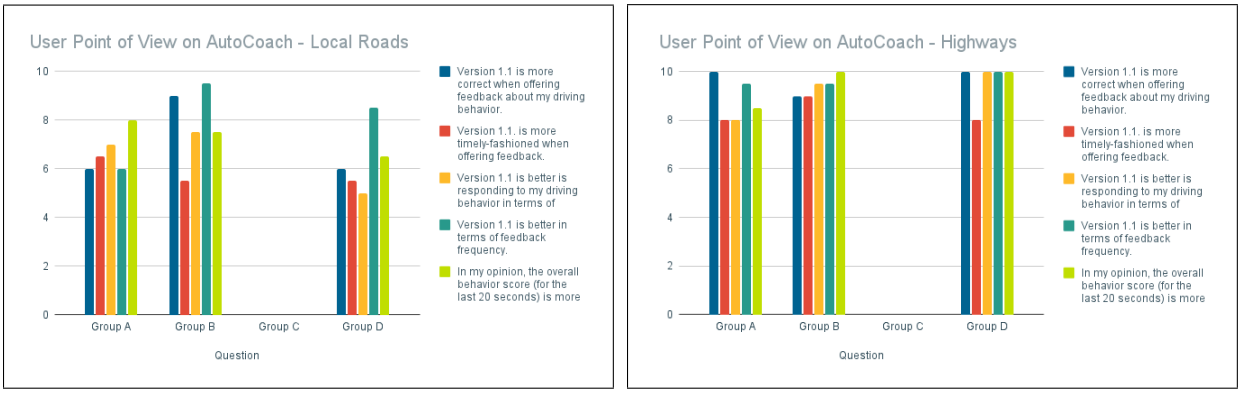
Due to v1.0's setup, some feedback could be ignored by the system assuming this is a safer driver and not all feedback is as important when driving risky. On the other hand, v1.1

assumes most risky behavior requires feedback and therefore, it issues more feedback. The delay is related to the system setup where feedback is not issued only every 10 second window passes. This parameter can be adjusted in later studies to test the usability of adjusting timing parameter and user experience to it.

Another driver from group GB said:

> Another driver from group *GA* said: "I noticed faster feedback."
> Another driver from group *GD* said: "I was able to sense the sense the timeliness when I hear the sound of earning coins."

10.5 Results and Discussion



(**a**) (**b**)

Figure 10.11: Success Assessment (**a**) Local Roads. (**b**) Highways.

This study's main intention is to understand the level of acceptance of the improved version of AutoCoach (v.1.1), which uses an adjusted scoring and feedback engines. In v1.1, a lighter memory factor model is assigned to drivers when the system detects a clear local road, while assigned a more responsive memory factor when detecting a clear highway. In this user study, we wish to compare v1.1 to the original version of AutoCoach (v1.0). Drivers need to accept the feedback offered by a system they are using in order to follow the recommendations

and reduce risk; consequently, studying the acceptability level of our system is an essential block. Based on our findings, AutoCoach has successfully gained an increased trust in the new version of AutoCoach. When driving on highways, feedback frequency, timeliness, and sensitivity have been noticed to have increased, building a more responsive and trustworthy system. While on local roads, the system has responded with more frequent feedback; however, it's been found to be also accepted by safer drivers. With the different personality characteristics and personality group assignments, drivers, this adjustment has successfully gained the driver's trust. We have learned from this study that the overall policy setup for each personality group does not entirely work for all groups. However, certain parameters have to be adjusted for each group. Some policies improved frequency but affected timeliness, while others improved other parameters and affected something else. However, the user study concluding results shown in Figure 10.11 indicated a 90.5% acceptance level of v1.1 compared to v1.0, while a 71.5% acceptance level of v1.1 compared to v1.0. We conclude that policy setup adjustments are essential under different contexts and such changes improve the overall UX of a system.

Chapter 11

Conclusions

Personalizing using human behavioral information is a prominent aspect of creating a better living. IoT's availability in our surrounding contexts makes it a great tool to identify behavioral issues. With its integration with AI, we can better persuade people with more precise supporting information.

In this work, we study the design of an intelligent agent for human drivers as a propitious example of new ML and AI applications. The AutoCoach agent aims to detect undesired user driving behaviors and provide personality-based feedback to improve them. The purpose of the feedback is to enhance the level of awareness for certain types of driving habits. By applying persuasive technologies, friendly feedback is delivered to drivers to respond to risky situations proactively. We have designed different memory factor models to encourage drivers to maintain their good behaviors for as long as possible. Good drivers are forgiven for minor mistakes along this path, and bad drivers are encouraged to improve as soon as possible. In this work, we differentiate between drivers based on personalized information rather than considering traditional statistics.

This project is one of the first automobile agents to classify users into different driver-

personality groups using machine learning algorithms. AutoCoach has implemented an intelligent real-time user feedback engine based on a driver's past typical behavior and personality group assessment. In this dissertation, we conducted several systematic pilot studies to test AutoCoach. Our studies have attempted to demonstrate the success of an innovative personality-based driver behavior management and feedback agent. We were exploring concepts as well as abstracting metrics when conducting this study. The validity of these concepts and the dependability of the metrics have yet to be shown. However, we conclude that by presenting our results in this manner, we have demonstrated how quantitative and qualitative approaches effectively evaluate the feasibility and acceptability of our system. Clearly, further research is required to discover the total utility of our model.

We observe that the participants' interaction with the personalized feedback system was more effective than the non-personalized system. We found the majority of participants agreeing that the personalization shows more intelligent and adequate real-time feedback. We gathered sufficient quantitative and qualitative data, which indicated that the intervention was acceptable and showed a decrease in risky driving behaviors. We further improved AutoCoach's model to adjust the feedback and scoring strategies under different road conditions, and we conducted another study to test the success of the new improvement. We observed that adjustment of feedback strategies under different road conditions outperformed the original model, which remains the same on different roads.

We learned that the timing and frequency of feedback are factors we could enhance in future research. We also learned that the test delivery method could be segmented into smaller portions to make sure participants could focus on each component individually. We want to conduct a study on a larger scale to test the behavior adaptation and factors affecting this adaptation in the future. We also would like to provide a more complex rewarding and scoring engine and reach the grand goal of behavior adaptation.

We have further improved our project to adjust the feedback frequency, sensitivity, timeliness

of the scoring engine, and we conducted another pilot study to test the success of our system. Results have indicated a more intelligent design according to the qualitative and quantitative data we collected. We learned from this study that different road conditions and road types affect how users interpret the received feedback and recommendations, and it is crucial to consider different contexts. AutoCoach's study results are auspicious and show a powerful example of facilitating new AI algorithms and capabilities.

For future work, many possibilities are available to enhance AutoCoach's learning components and improve its feedback delivery by incorporating different measures of persuasive technology. Personalizing using human behavioral information is a prominent aspect of creating a better living. IoT's availability in our surrounding contexts makes it a great tool to identify behavioral issues. With its integration with AI, we can better persuade people with more precise supporting information.

Bibliography

[1] L. Alben. Quality of experience: defining the criteria for effective interaction design. *interactions*, 3(3):11–15, 1996.

[2] Z. Allam and Z. A. Dhunny. On big data, artificial intelligence and smart cities. *Cities*, 89:80–91, 2019.

[3] Allstate Insurance Company. https://www.allstate.com.

[4] Y. Bian, C. Yang, J. L. Zhao, and L. Liang. Good drivers pay less: a study of usage-based vehicle insurance models. *Transportation Research Part A: Policy and Practice*, 107:20 – 34, 2018.

[5] S. Birnbaum. What's next for usage-based insurance?, April 2017. [Online; accessed 10-March-2018].

[6] A. Brandenburg. Usage based insurance and telematics. *National Association for Insurance Commissioners*, 2018. [Online; accessed 2-April-2018].

[7] M. R. Carlos, L. C. González, J. Wahlström, G. Ramírez, F. Martínez, and G. Runger. How smartphone accelerometers reveal aggressive driving behavior?—the key is the representation. *IEEE Transactions on Intelligent Transportation Systems*, 21(8):3377–3387, 2019.

[8] G. Castignani, T. Derrmann, R. Frank, and T. Engel. Driver behavior profiling using smartphones: A low-cost platform for driver monitoring. *IEEE Intelligent Transportation Systems Magazine*, 7(1):91–102, 2015.

[9] T. K. Chan, C. S. Chin, H. Chen, and X. Zhong. A comprehensive review of driver behavior analysis utilizing smartphones. *IEEE Transactions on Intelligent Transportation Systems*, 21(10):4444–4475, 2020.

[10] Z. Chen, J. Yu, Y. Zhu, Y. Chen, and M. Li. D 3: Abnormal driving behaviors detection and identification using smartphone sensors. In *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 524–532. IEEE, 2015.

[11] P. Desyllas and M. Sako. Profiting from business model innovation: Evidence from pay-as-you-drive auto insurance. *Research Policy*, 42(1):101 – 116, 2013.

[12] C. Dijksterhuis, B. Lewis-Evans, B. Jelijs, D. de Waard, K. Brookhuis, and O. Tucha. The impact of immediate or delayed feedback on driving behaviour in a simulated pay-as-you-drive system. *Accident Analysis & Prevention*, 75:93–104, 2015.

[13] C. Dijksterhuis, B. Lewis-Evans, B. Jelijs, O. Tucha, D. de Waard, and K. Brookhuis. In-car usage-based insurance feedback strategies. a comparative driving simulator study. *Ergonomics*, 59(9):1158–1170, 2016.

[14] C. Dijksterhuis, B. Lewis-Evans, B. Jelijs, O. Tucha, D. de Waard, and K. Brookhuis. In-car usage-based insurance feedback strategies. a comparative driving simulator study. *Ergonomics*, 59(9):1158–1170, 2016. PMID: 26653393.

[15] J. P. Ehsani, F. O'Brien, and B. Simmons-Morton. Comparing g-force measurement between a smartphone app and an in-vehicle accelerometer. *2017 Driving Assessment Conference*, 2017.

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

[17] H. Eren, S. Makinist, E. Akin, and A. Yilmaz. Estimating driving behavior by a smartphone. In *2012 IEEE Intelligent Vehicles Symposium*, pages 234–239, June 2012.

[18] J. Ferreira, E. Carvalho, B. V. Ferreira, C. de Souza, Y. Suhara, A. Pentland, and G. Pessin. Driver behavior profiling: An investigation with different smartphone sensors and machine learning. *PLoS one*, 12(4):e0174959, 2017.

[19] B. J. Fogg. Persuasive technology: Using computers to change what we think and do. *Ubiquity*, 2002(December), Dec. 2002.

[20] J. Froehlich, L. Findlater, and J. Landay. The design of eco-feedback technology. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1999–2008, New York, NY, USA, 2010. ACM.

[21] Google. K-means advantages and disadvantages: Clustering in machine learning.

[22] X. Guo, Z. Shen, Y. Zhang, and T. Wu. Review on the application of artificial intelligence in smart homes. *Smart Cities*, 2(3):402–420, 2019.

[23] A. Gupte. System and method for monitoring driving behavior with feedback, Oct. 2 2008. US Patent App. 12/079,837.

[24] C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004.

[25] Y.-M. Huang and S.-X. Du. Weighted support vector machine for classification with uneven training class sizes. In *2005 International Conference on Machine Learning and Cybernetics*, volume 7, pages 4365–4369. IEEE, 2005.

[26] S. Husnjak, D. Peraković, I. Forenbacher, and M. Mumdziev. Telematics system in usage based motor insurance. *Procedia Engineering*, 100:816 – 825, 2015. 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014.

[27] P. Händel, J. Ohlsson, M. Ohlsson, I. Skog, and E. Nygren. Smartphone-based measurement systems for road vehicle traffic monitoring and usage-based insurance. *IEEE Systems Journal*, 8(4):1238–1248, Dec 2014.

[28] T. Imkamon, P. Saensom, P. Tangamchit, and P. Pongpaibool. Detection of hazardous driving behavior using fuzzy logic. In *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, volume 2, pages 657–660, May 2008.

[29] T. Imkamon, P. Saensom, P. Tangamchit, and P. Pongpaibool. Detection of hazardous driving behavior using fuzzy logic. In *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, volume 2, pages 657–660. IEEE, 2008.

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

[31] S. Kanarachos, S.-R. G. Christopoulos, and A. Chroneos. Smartphones as an integrated platform for monitoring driver behaviour: The role of sensor fusion and connectivity. *Transportation Research Part C: Emerging Technologies*, 95:867–882, 2018.

[32] D. Karapiperis, A. Obersteadt, A. Brandenburg, S. Castagna, B. Birnbaum, A. Greenberg, and R. Harbage. Usage-based insurance and vehicle telematics: insurance market and regulatory implications. *CIPR Study Series*, 1:1–79, 2015.

[33] A. Kashevnik, I. Lashkov, and A. Gurtov. Methodology and mobile application for driver behavior analysis and accident prevention. *IEEE transactions on intelligent transportation systems*, 21(6):2427–2436, 2019.

[34] I. Kononenko. Estimating attributes: Analysis and extensions of relief. In *European conference on machine learning*, pages 171–182. Springer, 1994.

[35] M. Kuniavsky. *Observing the user experience: a practitioner's guide to user research*. Elsevier, 2003.

[36] T. Lajunen and D. Parker. Are aggressive people aggressive drivers? a study of the relationship between self-reported general aggressiveness, driver anger and aggressive driving. *Accident Analysis & Prevention*, 33(2):243–255, 2001.

[37] Y. Li, F. Xue, L. Feng, and Z. Qu. A driving behavior detection system based on a smartphone's built-in sensor. *International Journal of Communication Systems*, 30(8), 2017.

[38] T. Litman. Distance-based vehicle insurance as a tdm strategy. *Transportation Quarterly*, 51:119–137, 1997.

[39] Z. Liu, Q. Shen, H. Li, and J. Ma. A risky driving behavior scoring model for the personalized automobile insurance pricing. In *Proceedings of the 2Nd International Conference on Crowd Science and Engineering*, ICCSE'17, pages 61–67, New York, NY, USA, 2017. ACM.

[40] Z. Liu, Q. Shen, and J. Ma. A driving behavior model evaluation for ubi. *International Journal of Crowd Science*, 1(3):223–236, 2017.

[41] J. Lu, D. Filev, K. Prakah-Asante, F. Tseng, and I. V. Kolmanovsky. From vehicle stability control to intelligent personal minder: Real-time vehicle handling limit warning and driver style characterization. In *2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*, pages 43–50. IEEE, 2009.

[42] C. Ma, X. Dai, J. Zhu, N. Liu, H. Sun, and M. Liu. Drivingsense: Dangerous driving behavior identification based on smartphone autocalibration. *Mobile Information Systems*, 2017, 2017.

[43] Z. Marafie, K.-J. Lin, D. Wang, H. Lyu, Y. Meng, and T. Ito. AutoCoach: Driving Behavior Management Using Intelligent IoT Services. In *2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)*, pages 103–110, Kaohsiung, Taiwan, 11 2019. IEEE.

[44] Metromile Inc. https://www.metromile.com.

[45] W. Nai, W. Qin, F. Zhang, Y. Yu, and D. Dong. Contract strategy design in early stage of implementing pay-how-you-drive policy with traditional commercial vehicle insurance coexisted. In *2015 12th International Conference on Service Systems and Service Management (ICSSSM)*, pages 1–4, June 2015.

[46] A. Nicolae. Method of transforming coordinates of a vehicle-mounted accelerometer: Example application of calculating lean angle of a motorcycle. *International Journal of Scientific and Engineering Research*, 7(10):891–921, 2016.

[47] B. Nicoletti. *Fintech Innovation*, pages 81–159. Springer International Publishing, Cham, 2017.

[48] S. Oh. Usage-based insurance (ubi) expected to grow to 142m subscribers globally by 2023, May 2016. [Online; accessed 28-August-2019].

[49] W. H. Organization. Global health observatory data: Road traffic deaths, 2019.

[50] J. Paefgen, F. Kehr, Y. Zhai, and F. Michahelles. Driving behavior analysis with smartphones: Insights from a controlled field study. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, MUM '12, pages 36:1–36:8, New York, NY, USA, 2012. ACM.

[51] J. Paefgen, F. Kehr, Y. Zhai, and F. Michahelles. Driving behavior analysis with smartphones: insights from a controlled field study. In *Proceedings of the 11th International Conference on mobile and ubiquitous multimedia*, page 36. ACM, 2012.

[52] Progressive Casualty Insurance Company.

[53] Y. A. Qadri, A. Nauman, Y. B. Zikria, A. V. Vasilakos, and S. W. Kim. The future of healthcare internet of things: a survey of emerging technologies. *IEEE Communications Surveys & Tutorials*, 22(2):1121–1167, 2020.

[54] P. Ranjitkar, T. Nakatsuji, Y. Azuta, and G. S. Gurusinghe. Stability analysis based on instantaneous driving behavior using car-following data. *Transportation Research Record*, 1852(1):140–151, 2003.

[55] M. Soleymanian, C. B. Weinberg, and T. Zhu. Sensor data and behavioral tracking: Does usage-based auto insurance benefit drivers? *Marketing Science*, 38(1):21–43, 2019.

[56] T. G. Stavropoulos, A. Papastergiou, L. Mpaltadoros, S. Nikolopoulos, and I. Kompatsiaris. Iot wearable sensors and devices in elderly care: a literature review. *Sensors*, 20(10):2826, 2020.

[57] J. Y. Stephen O'Hearn. Oppertunities await: how insurtech is reshaping insurance, June 2016.

[58] J. M. Sullivan. 12 behavioral adaptation. *Handbook of Human Factors for Automated, Connected, and Intelligent Vehicles*, 2020.

[59] O. Taubman-Ben-Ari, M. Mikulincer, and O. Gillath. The multidimensional driving style inventory—scale construct and validation. *Accident Analysis & Prevention*, 36(3):323–332, 2004.

[60] K. M. Tjørve and E. Tjørve. The use of gompertz models in growth analyses, and new gompertz-model approach: An addition to the unified-richards family. *PloS one*, 12(6):e0178691, 2017.

[61] T. Toledo, O. Musicant, and T. Lotan. In-vehicle data recorders for monitoring and feedback on drivers behavior. *Transportation Research Part C: Emerging Technologies*, 16(3):320 – 331, 2008. Emerging Commercial Technologies.

[62] D. I. Tselentis, G. Yannis, and E. I. Vlahogianni. Innovative motor insurance schemes: A review of current practices and emerging challenges. *Accident Analysis and Prevention*, 98:139 – 148, 2017.

[63] G. Warren and M. Greenlee. Calculation of driver score based on vehicle operation, 3 2006. US Patent App. 10/936,293.

[64] P. I. Wouters and J. M. Bos. Traffic accident reduction by monitoring driver behaviour with in-car data recorders. *Accident Analysis and Prevention*, 32(5):643 – 650, 2000.

[65] M. Wu, S. Zhang, and Y. Dong. A novel model-based driving behavior recognition system using motion sensors. *Sensors*, 16(10):1746, 2016.

[66] F. Zantalis, G. Koulouras, S. Karabetsos, and D. Kandris. A review of machine learning and iot in smart transportation. *Future Internet*, 11(4):94, 2019.

[67] A. S. Zeeman and M. J. Booysen. Combining speed and acceleration to detect reckless driving in the informal public transport industry. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 756–761. IEEE, 2013.

[68] S. L. Zeger and K.-Y. Liang. Longitudinal data analysis for discrete and continuous outcomes. *Biometrics*, 42(1):121–130, 1986.

[69] H. Zhao, H. Zhou, C. Chen, and J. Chen. Join driving: A smart phone-based driving behavior evaluation system. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 48–53. IEEE, 2013.

[70] C. H. D. Y. S. Zheng. The automobile insurance pricing model, combining static premium with dynamic premium——based on the generalized linear models. *World Risk and Insurance Economics Congress (WRIEC)*, 2015.