TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Robotik, Künstliche Intelligenz und Echtzeitsysteme

Enabling Data-Driven Functions in Automotive E/E Architectures

Christoph Segler

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Thomas Neumann
Prüfer der Dissertation:	1. Prof. Dr.-Ing. habil. Alois Knoll
	2. Prof. Dr.-Ing. Markus Lienkamp

Die Dissertation wurde am 05.10.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 11.03.2021 angenommen.

Abstract

In the automotive domain, the prevalence of machine learning functions is increasing. This trend drives further development of automated driving, new vehicle functions, and integration of new sensors and results in more and more generated vehicle data. Current vehicles generate up to 25 GB of data per hour from up to 14 000 signals. This considerably rich amount of data already present in the vehicle enables the development of new functions, especially *data-driven functions*. These data-driven functions can range from raw sensor data pre-processing, system functions, context-aware functions, to anomaly detection.

Most of these data-driven functions are based on machine learning algorithms and represent a relationship between input and output. In the case of automotive data-driven functions, the input originates from a set of vehicle signals. The high number (i. e. high dimensionality) of signals accessible in the vehicle, leads to a challenge that is known as the *curse of dimensionality*. This *curse* relates to the challenge of having a too large amount of possible input signals for a data-driven function, which may contain irrelevant, redundant, or not suitable signals yielding in reduced performance and efficiency. This curse does not only affect algorithms for data-driven functions but also the *function developer* who is developing a data-driven function. In the development phase, the function developer has to select the appropriate input signals for the data-driven function and therefore might miss relevant signals which are essential for a good performance. Furthermore, current E/E architectures are not optimised for data-driven functions and raise additional challenges.

This work presents an approach to automatically select a signal subset out of all available vehicle signals, which should be used as input for the data-driven function. The proposed approach can either be deployed directly onboard the vehicle or in the back end. The approach is first evaluated on exemplary data-driven functions with vehicle data sets in a pure offline manner followed by an evaluation, demonstrating the streaming capabilities of the approach, meaning that no significant amount of data has to be stored onboard the vehicle or transferred to a back end. Finally, the real-world applicability of the approach is demonstrated with a proof of concept in a real vehicle.

Zusammenfassung

In der Automobilindustrie findet die Anwendung von Maschinellem Lernen eine immer weitere Verbreitung. Dies treibt unter anderem die Entwicklung von autonomen Fahren, neuen Fahrzeugfunktionen und die Integration von neuen Sensoren voran. Hierdurch erzeugen und verarbeiten heutige Fahrzeuge immer mehr Daten. Aktuell erzeugt ein Fahrzeug bis zu 25 GB pro Stunde bestehend aus bis zu 14 000 Signalen. Diese beträchtliche Menge an Daten ermöglicht die Entwicklung neuer Fahrzeugfunktionen, insbesondere datengetriebene Funktionen. Diese datengetriebenen Funktionen reichen von Sensorrohdatenverarbeitung, Systemfunktionen, kontextsensitiven Funktionen bis hin zur Anomalieerkennung.

Die meisten dieser Funktionen nutzen Maschinelles Lernen, um eine Beziehung zwischen dem Eingangswert und dem Funktionsausgang zu lernen und darzustellen. Im Falle von datengetriebenen Funktionen im Fahrzeug dienen Fahrzeugsignale als Eingangswert. Die hohe Anzahl (d.h. hohe Dimensionalität) an Signalen, die im Fahrzeug zur Verfügung steht, führt zu einer Herausforderung auch bekannt als *curse of dimensionality*. Dieser *Fluch* bezieht sich auf die Herausforderung eine zu große Menge an potentiellen Eingangssignalen für eine datengetriebene Funktion zu haben, welche u.a. irrelevant, redundant oder unpassend sein könnten. Dies kann zu einem schlechteren Ergebnis bzw. Effizienz des Algorithmus führen und beeinflusst nicht nur diesen, sondern auch den/die Funktionsentwickler/in. In der Entwicklungsphase werden u.a. die geeigneten Eingangssignale für die Funktion ausgewählt, was dazu führen kann, dass relevante Signale übersehen werden, die essenziell für ein gutes Ergebnis wären. Des Weiteren sind aktuelle E/E Architekturen nicht für datengetriebene Funktionen ausgelegt, was weitere Herausforderungen mit sich führt.

In dieser Arbeit wird ein Ansatz vorgestellt, der automatisch aus allen Fahrzeugsignalen die relevanten auswählt, die als Eingangswert für die datengetriebene Funktion genutzt werden sollten. Dieser Ansatz kann entweder direkt im Fahrzeug oder auch im Backend ausgeführt werden. Zuerst wird der Ansatz mittels mehrerer datengetriebener Funktionen und Fahrzeugdatensätzen bewertet, gefolgt von einer Evaluation, die die Streaming-Fähigkeiten des Ansatzes zeigt. D.h. es muss keine signifikante Datenmenge im Fahrzeug gespeichert oder in das Backend übertragen werden. Die praktische Anwendbarkeit dieses Ansatzes wird in einem Versuchsfahrzeug gezeigt.

"All models are wrong, but some models are useful.
So the question you need to ask is not *Is the model true?* (it never is)
but *Is the model good enough for this particular application?*"

George E. P. Box [1, p. 61]

Acknowledgements

First of all, I would like to express my sincere gratitude to Professor Alois Knoll for giving me the opportunity to write my thesis in this research field in cooperation with an industry partner and supporting me personally and professionally during this thesis. As well I would like to thank my industry supervisor Hans-Jörg Vögel for his constructive feedback and giving me the opportunity to work on my thesis at BMW Group Research.

Furthermore, I would like to thank all my colleagues at the Chair of Robotics, Artificial Intelligence and Real-Time Systems and the BMW Group for the fruitful discussions and critical input. Specifically, I would like to thank Stefan Kugele for the valuable discussions and as great co-author in many of my publications. Also, I would like to thank Sina Shafaei as a colleague at the chair, Morteza Hashemi Farzaneh as a mentor, Ute Lomp for helping me with all organisational matters at the university, and Alexander Lenz and Amy Bücherl for their great support at organising the IEEE International Conference on Cyborg and Bionic Systems (CBS) 2019. I also want to thank my BMW colleagues Philipp Obergfell, Thomas Hubregtsen, Florian Mirus, and Nico Epple for the enjoyable work environment, their new viewpoints on my work, the bug hunting in my code, the managing of our computing hardware, and for sharing their data sets.

Beyond that, I want to thank my partner Emely, for her support and patience throughout my work on this thesis. I also want to express my special thanks to my family for always supporting me. Without your steady support, this work would not have been possible.

Glossary

ACC Adaptive Cruise Control

ADAS Advanced Driver Assistance System

BMW Bayerische Motoren Werke AG

BNE BordNetz Engineer

CAN Controller Area Network

CARMEN CAR Measurement Environment

CC Check Control

CCPA California Consumer Privacy Act

CI/CD Continuous Integration and Continuous Deployment

CID Central Information Display

CIFE Conditional Infomax Feature Extraction

CMIM Conditional Mutual Information Maximization

CRC Cyclic Redundancy Check

DDF Data-Driven Function

DEC Driving Experience Control

DISR Double Input Symmetrical Relevance

DRM Digital Rights Management

ECU Electronic Control Unit

E/E Electric/Electronic

ExVe Extended Vehicle

Glossary

FCBF Fast Correlation Based Filter

FIBEX Field Bus Exchange Format

GDPR General Data Protection Regulation

GUI Graphical User Interface

HMI Human-Machine Interface

HVAC Heating Ventilation Air Conditioning

ICAP Interaction Capping

ICSA International Conference on Software Architecture

ICSE International Conference on Software Engineering

IP Intellectual Property

ITSC Intelligent Transportation Systems Conference

IV Intelligent Vehicles Symposium

JMI Joint Mutual Information

JSON JavaScript Object Notation

LIN Local Interconnect Network

MCC Matthew's Correlation Coefficient

MIFS Mutual Information Feature Selection

MIM Mutual Information Maximization

MRMR Minimum Redundancy Maximum Relevance

NaN Not a Number

OEM Original Equipment Manufacturer

PCA Principal Component Analysis

PDU Protocol Data Unit

RF Random Forest

RQ Research Question

SOA Service-Oriented Architecture

SOP Start of Production

SVM State Vector Machine

TSN Time-Sensitive Networking

UDP User Datagram Protocol

VDA Verband der Automobilindustrie e.V.

VIAS Vehicle Information API Specification

VIN Vehicle Identification Number

VISS Vehicle Information Service Specification

VSS Vehicle Signal Specification

VSSo Vehicle Signal Specification ontology

W3C World Wide Web Consortium

YAML YAML ain't markup language

Symbols

f_c Classically designed vehicle function $f_c : \boldsymbol{x} \rightarrow \boldsymbol{y}$

f_{dd} Data-driven vehicle function $f_{dd} : \boldsymbol{x} \rightarrow \boldsymbol{y}$

$\boldsymbol{x}$ Input vector $\boldsymbol{x}$ of the vehicle function corresponding to input $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$

$\boldsymbol{y}$ Output vector $\boldsymbol{y}$ of the vehicle function corresponding to output $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$

$\hat{\boldsymbol{y}}$ Label vector $\hat{\boldsymbol{y}}$ corresponding to the output $\hat{\boldsymbol{y}} = \{\hat{y_1}, \hat{y_2}, \ldots, \hat{y_n}\}$ of f_{dd}

C Context

U User/Driver

V Vehicle

C_U Context perceivable by the user

C_V Context perceivable by the vehicle

$C_U \setminus C_V$ Context only perceivable by the user

$C_V \setminus C_U$ Context only perceivable by the vehicle

$C_U \cap C_V$ Context perceivable by the user and the vehicle

S All vehicle signals $S = \{s_1, \ldots, s_i, \ldots, s_m\}$ with m signals, $1 \leq i \leq m, m \in \mathbb{N}^+$

S_{pre} All pre-processed vehicle signals $f_{pre} : S \rightarrow 2^{S_{pre}}$

S_{init} All signals in S_{pre}, S_E, S_G (input for the signal subset selection)

$f_{\mathcal{A}}$ The function of the algorithm $\mathcal{A}$ for the signal selection $f_{\mathcal{A}} : S \rightarrow 2^{S_{\mathcal{A}}} = \{s_1, \ldots, s_n\}$, where $S_{\mathcal{A}} \subseteq S$

$S_{\mathcal{A}}$ Vehicle signal subset selected by $f_{\mathcal{A}} : S \rightarrow 2^{S_{\mathcal{A}}} = \{s_1, \ldots, s_n\}$, where $S_{\mathcal{A}} \subseteq S$ and the algorithm $f_{\mathcal{A}}$

S_i Vehicle signal subset selected by $f_{\mathcal{A}} : S \rightarrow 2^{S_i} = \{s_1, \ldots, s_n\}$ on data set/user i

S_j Vehicle signal subset selected by $f_{\mathcal{A}} : S \rightarrow 2^{S_j} = \{s_1, \ldots, s_n\}$ on data set/user j

$S_{\mathcal{A}}^{I}$ Selected vehicle signal subset of type I (System)

$S_{\mathcal{A}}^{II}$ Selected vehicle signal subset of type II (Group)

$S_{\mathcal{A}}^{III}$ Selected vehicle signal subset of type III (User)

S_E All external signals $S_E = \{s_1, \ldots, s_e, \ldots, s_o\}$ with o external signals, $1 \leq e \leq o, o \in \mathbb{N}^+$

S_G All generated signals $S_G = \{s_1, \ldots, s_g, \ldots, s_p\}$ with p generated signals, $1 \leq g \leq p, p \in \mathbb{N}^+$

$S_{\hat{G}}$ All signals required for the generation of s_g: $S_{\hat{G}} = \{s_1, \ldots, s_{\hat{g}}, \ldots, s_q\}$

$S_{\hat{L}}$ All signals required for the generation of s_l: $S_{\hat{L}} = \{s_1, \ldots, s_{\hat{l}}, \ldots, s_r\}$

s_i Internal vehicle signal i

$\overline{s_i}$ Received value of signal s_i

s_{ij} Pre-processed internal vehicle signal ij based on original vehicle signal s_i

$\overline{s_{ij}}$ Value of pre-processed internal vehicle signal s_{ij} based on original vehicle signal s_i

s_e External signal e

s_g Generated signal g

$s_{\hat{g}}$ Signal $\hat{g}$ basis for generated signal s_g

$s_{\hat{l}}$ Signal $\hat{l}$ basis for label s_l

s_l Generated label s_l

$\overline{s_l}$ Value of generated label l

s_a Pre-processed vehicle signal $s_a \in S_{init}$

$\overline{s_a}$ Value of pre-processed vehicle signal s_a

$res(s_i)$ Specified resolution of signal s_i (only in the case of continous data)

$off(s_i)$ Specified offset of signal s_i (only in the case of continous data)

$sp(s_i, k)$ Specified special value k of signal s_i (only in the case of continous data)

$val(s_i, k)$ Specified value k of signal s_i (only in the case of enumerated data)

ϕ_{ik} Transformation from the received signal s_i with bit field k (only in the case of bit field encoded data)

$score_a$ Score of signal s_a

$rank_a$ Rank of signal s_a based on score $score_a$

$c(S_i)$ Classifier c trained on signal subset S_i

mcc_i Matthew's Correlation Coefficient of classifier $c(S_i)$ trained with signal set S_i

$mcc_{\Delta ij}$ Difference of Matthew's Correlation Coefficient of classifiers $c(S_i)$ and $c(S_j)$

J_{ij} Jaccard distance $J(S_i, S_j)$ between the signal sets S_i and S_j

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning is a highly emerging field, with a variety of applications. In the automotive domain, this trend drives further development of automated driving, new vehicle functions, and integration of new sensors. This development results in more and more generated and processed data in vehicles. Current vehicles generate up to 25 GB of data per hour [2] from up to 14 000 signals. This sensory data does not only include complex data such as input data for the environment perception of autonomous vehicles, but also less sophisticated data like temperatures, weather conditions, or the current road type. This considerably rich amount of data enables the development of new functions, especially *Data-Driven Functions (DDFs)*.

DDFs are differently designed compared to classical automotive functions. In the case of classical automotive functions, the functional logic is created by a developer based on the functional and non-functional requirements. In the case of DDFs, the input vector $\boldsymbol{x}$ and the output vector $\boldsymbol{y}$ of the function are defined, and the logic itself tries to represent relationships, which are solely based on training data and the underlying algorithm. These functions range from (i) raw sensor data pre-processing, (ii) system functions, (iii) context-aware functions, to (iv) anomaly detection.

Most of these DDFs are based on machine learning algorithms. In the case of automotive DDFs, the input $\boldsymbol{x}$ originates from a set of vehicle signals. The high number (i. e. high dimensionality) of signals accessible in the vehicle, leads to a challenge that is known as the *curse of dimensionality* [3]. This *curse* relates to the challenge of having a too large amount of possible input signals for a DDF, which may contain irrelevant, redundant, or not suitable signals yielding in reduced performance and efficiency. This curse does not only affect algorithms for DDFs but also the *function developer*. In the development phase, the function developer has to select the appropriate input signals for the DDF, and therefore, might miss relevant signals which are essential for a good performance. It would also not be reasonable to use all available signals as input for a single function, as the hardware resources within the vehicle are highly limited. Furthermore, current Electric/Electronic (E/E) architectures are not optimised for DDFs and

raise additional challenges. Besides, automotive data is mostly only available onboard the vehicle, in contrast to classical data mining or machine learning applications, where the data is available in the back end.

An example of such a DDF could be a function which is proactively switching the driving mode from *comfort mode* to *sport mode* and vice versa. This can be achieved by learning the user's behaviour based on data when the driver is changing the mode manually. A developer would not only have to design the algorithm and the output of the function, but also the input. Out of 14 000 vehicle signals (and even more internal signals)[1], the function developer has to select the appropriate signals representing the user's behaviour. In this case, the developer has to know which signals contain the correct information to learn the user's behaviour, which are measured by a sensor.

The main idea of the presented approach is to automatically select a signal subset out of the total available vehicle signals, which should be used as input for the DDF. The proposed approach can either be deployed directly onboard the vehicle or the back end.

1.1 Research Questions

For this work, we raise the following five Research Questions (RQs):

The first RQ relates to the *curse of dimensionality* mentioned earlier and the vast amount of vehicle signals available in current vehicles. For the development of DDFs in the automotive domain, it would not be reasonable to use all vehicle signals as input. To reduce the number of input signals, the appropriate vehicle signals have to be identified and selected. Therefore, we pose the following RQ:

Research Question 1 (Signal Subset Identification)
How to identify vehicle signals which are potentially relevant as input for a specific DDF?

The next RQ is related to the vehicles architecture and its impacts on DDFs. Current E/E architectures are historically grown and not designed for data mining or machine learning applications. The characteristics of E/E architectures have a significant impact on DDFs and pose additional challenges. For this work, we state the following additional RQ:

Research Question 2 (E/E Architecture Characteristics)
How to integrate the identification approach into the vehicle's architecture and consider the characteristics of automotive E/E architectures?

[1] Internal signals are signals which are only sent within an Electronic Control Unit (ECU) and not over the vehicle's communication networks. These often cannot be accessed from outside of this particular ECU and are therefore called ECU internal signals.

The developed DDFs will not only be deployed on one single vehicle. As an Original Equipment Manufacturer (OEM), the development of DDFs involves large vehicle fleets and a large number of users. As these functions can range from simple system-related functions which are similar for all vehicles, to highly personalised functions which are unique for each user, we extend the previous RQs with the additional requirement of scalability for large vehicle fleets:

Research Question 3 (Scalability)
How to scale this identification approach over large vehicle fleets from simple system-related DDFs to highly personalised DDFs?

Finally, the discovered vehicle signals have to be integrated as input for a DDF. This can either be done manually or automatically. Therefore we pose two RQs related to the integration of the discovered vehicle signals into the DDF. The first one considers the manual integration by the function developer and the second one, the automatic integration:

Research Question 4 (Manual Post-Processing)
How to assist the function developer in identifying potential information and signals which could be relevant as input for a specific DDF?

Research Question 5 (Automatic Post-Processing)
How to automatically use the discovered input signals for a specific DDF?

1.2 Structure

An overview of the structure of this thesis is depicted in Figure 1.1. Each box represents a section of this thesis and is vertically aligned to the related sections in the other chapters.

Chapter 1 introduces the topic, motivation for this work, and a list of contributions, which are part of this thesis or related to it.

In Chapter 2 the background and related work for this work are presented. The first section shows the state-of-the-art for DDFs within the automotive domain and is followed by an explanation of the impacts of the automotive E/E architecture on DDFs and their development. Finally, we introduce the related work and the state-of-the-art used in this work.

The approach itself is shown in Chapter 3. Section 3.1 gives an overview of the concept and the general approach of the presented method, which aims to answer the before mentioned RQs. The following sections describe the approach step by step. In Section 3.2, we introduce the *DDF specification*, which is the basis for each step. In the first step of the approach, the *vehicle signal pre-processing* (cf. Section 3.3) is specially designed for the characteristics of the automotive E/E architecture and available specifications. This step is followed by the *label and signal generation* presented in Section 3.4. In this step, the actual label information and additional input signals are generated. Section 3.5 introduces the *signal subset selection* process. These signal subsets contain the proposed set of signals, which should be used as DDF input. This step is mainly based on feature selection algorithms introduced in the related work. The

selected signal subset can then be post-processed in multiple ways. Section 3.6 presents the post-processing of the selected signal subset (i. e. how the set is used after the selection step). Finally, in Section 3.7, we introduce the deployment strategy of the proposed approach.

This complete approach aims to answer RQs 1 and 2. The outcome of the approach is the signal subset which contains potential relevant signals which can be used as input for a specific DDF. Also, the proposed approach is designed to consider the characteristics of automotive E/E architectures. RQ 3 is focused on the scalability over large vehicle fleets and is addressed with the proposed deployment strategy. The RQs related to the post-processing of the selected signals subset (RQ 4 and 5) are addressed by the proposed post-processing presented in Section 3.6.

For the evaluation of the proposed approach, multiple evaluations were conducted. The first evaluation is presented in Chapter 4. Here, we conduct an offline evaluation on data collected from customer vehicles. The first three sections (Sections 4.1 to 4.3) define the underlying 24 test cases, the used test setup, and the 101 data sets, followed by a separate presentation and discussion of the three evaluations. The first evaluation assesses the proposed vehicle signals pre-processing and compares its performance to the raw vehicle signals. Based on the pre-processed signals, the label for each test case is generated. This part of the approach cannot be evaluated because it is solely based on the specifications for each test case. In Section 4.5, we evaluate the actual signal subset selection step. The used algorithms highly impact the performance of the selection. To minimise the effect of the selected algorithm, the approach is evaluated with 15 different algorithms, on 24 test cases, using 101 data sets. In the last section of this chapter, we evaluate the deployment strategy on the same test cases and data sets.

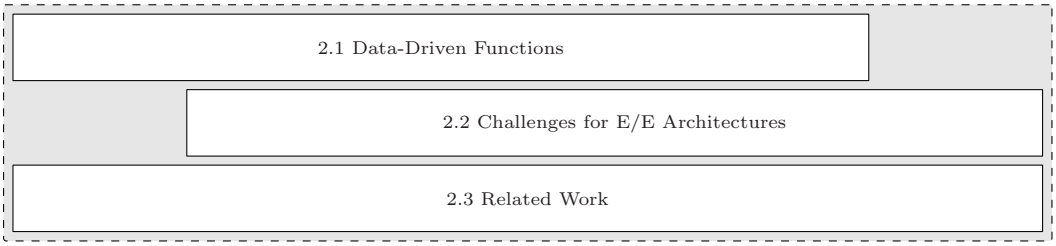
Chapter 5 presents the assessment of the streaming capabilities of the approach. With this evaluation, we want to show the ability of the approach to be executed without the need to store any significant amount of data. This is especially important for an onboard execution. The first two sections (Sections 5.1 and 5.2) show the test cases and setup for this evaluation. The data sets used in this evaluation are identical to the data sets shown in the previous chapter and are not further introduced in this chapter. The actual assessment, including its discussion, is presented in Section 5.3.

The streaming evaluation is followed by an onboard proof of concept of the approach, presented in Chapter 6. We integrated the proposed approach into a test vehicle and demonstrated its onboard execution capabilities in a real car, including live vehicle data.

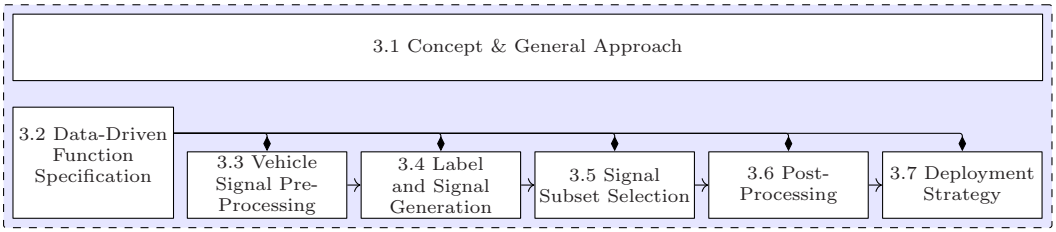
Chapter 7 summarises this work with a section for each chapter, concluded by the contributions of this work. Finally, Chapter 8 gives an outlook into future work and remaining challenges for DDFs and data analytics in the automotive domain.

1 Introduction

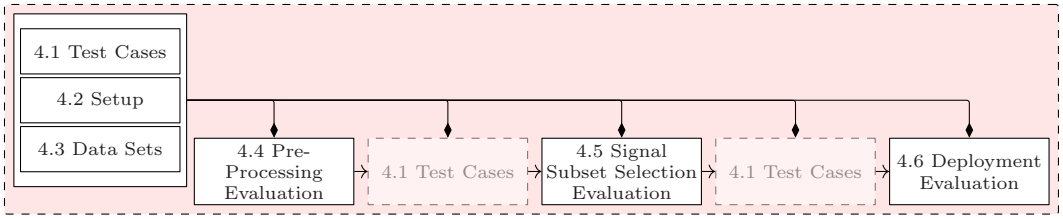
2 Background



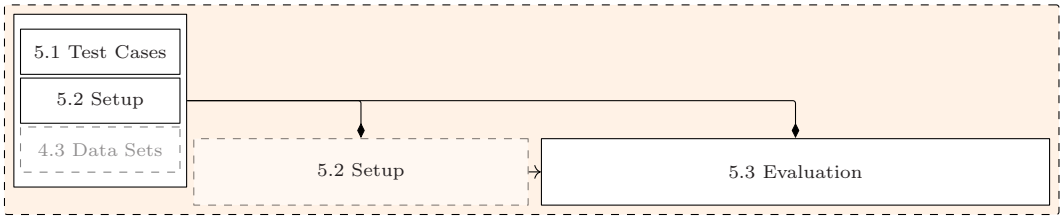
3 Approach



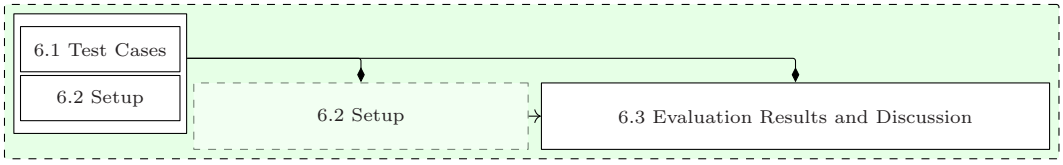
4 Offline Evaluation



5 Streaming Evaluation



6 Onboard Proof of Concept



7 Summary and Conclusion

8 Outlook

Figure 1.1: Thesis structure

1.3 Contributions

Parts of this thesis have been previously published at international peer-reviewed conferences, peer-reviewed workshops or have been filed as patent applications:

In the following work parts of the challenges for the development of DDFs posed by the automotive E/E architecture presented in Section 2.2 and the general concept and parts of the approach presented in Chapter 3 have been published. This includes a preliminary evaluation of a part of the approach similar to Section 4.5:

1. Christoph Segler, Stefan Kugele, and Alois Knoll. **Context Discovery for Personalised Automotive Functions**. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019. [4]

In addition to this publication, the main concept behind the approach has been filed in the following patent application:

2. Christoph Segler and Sina Shafaei. **Verfahren, Vorrichtung, Computerprogramm und Computerprogrammprodukt zur Datenbearbeitung für ein Fahrzeug**. *Patent Application DE 102018202348 A1*, Filed on 15th of February 2018. [5]

The initial concept behind the test case *anomaly detection*, including a concept for the assessment of anomalies at the OEM, has been published in the following work:

3. Philipp Obergfell*, Christoph Segler*, Eric Sax, and Alois Knoll (*contributed equally). **Synchronization between Run-Time and Design-Time View of Context-Aware Automotive System Architectures**. In *IEEE International Systems Engineering Symposium (ISSE)*, 2018. [6]

and has been filed in the following patent application:

4. Philipp Obergfell* and Christoph Segler* (*contributed equally). **Method Indicating Unexpected Behaviour and Vehicle, System, and Storage Medium Comprising the Same**. *Patent Application WO 2020020437 A1*, Filed on 24th of July 2018. [7]

The concept behind the streaming evaluation and its implementation presented in Chapter 5, including a preliminary evaluation on different data sets, has been published in the following works. Additionally, these publications presented parts of the challenges for the development of DDFs in automotive E/E architectures:

5. Christoph Segler, Stefan Kugele, Philipp Obergfell, Mohd Hefeez Osman, Sina Shafaei, Eric Sax, and Alois Knoll. **Evaluation of Feature Selection for Anomaly Detection in Automotive E/E Architectures**. In *ACM/IEEE International Conference on Software Engineering (ICSE): Companion Proceedings*, 2019. [8]

6. Christoph Segler, Stefan Kugele, Philipp Obergfell, Mohd Hafeez Osman, Sina Shafaei, Eric Sax and Alois Knoll. **Anomaly Detection for Advanced Driver Assistance Systems Using Online Feature Selection**. In *IEEE Intelligent Vehicles Symposium (IV)*, 2019. [9]

For the design of the DDFs and its implications to the software architecture, we have proposed a design pattern for DDFs, including the role of the presented approach. This includes the general concept for the automatic input selection which is part of the approach presented in Section 3.6.2. In the published work, we have also partly evaluated the presented approach and the interconnections between multiple DDFs:

7. Stefan Kugele*, Christoph Segler*, and Thomas Hubregtsen* (*contributed equally). **Architectural Patterns for Cross-Domain Personalised Automotive Functions**. In *IEEE International Conference on Software Architecture (ICSA)*, 2020. [10]

The following works use the proposed approach to enable and design DDFs in the automotive domain, including an evaluation:

8. Ilias Gerostathopoulos, Stefan Kugele, Christoph Segler, Tomas Bures, and Alois Knoll. **Automated Trainability Evaluation for Smart Software Functions**. In *IEEE/ ACM International Conference on Automated Software Engineering (ASE)*, 2019. [11]

9. Thomas Hubregtsen, Christoph Segler, Josef Pichlmeier, Aritra Sarkar, Thomas Gabor, and Koen Bertels. **Integration and Evaluation of Quantum Accelerators for Data-Driven User Functions**. In *International Symposium on Quality Electronic Design (ISQED)*, 2020. [12]

The following works are not directly part of this thesis, but further discuss the implications of DDFs and machine-learning functions and their impact on automotive E/E architectures:

10. Stefan Kugele, Vadim Cebotari, Mario Gleirscher, Morteza Hashemi, Christoph Segler, Sina Shafaei, Hans-Jörg Vögel, Fridolin Bauer, Alois Knoll, Diego Marmsoler, and Hans-Ulrich Michel. **Research Challenges for a Future-Proof E/E Architecture - A Project Statement**. In *GI Informatik*, 2017. [13]

11. Philipp Obergfell, Stefan Kugele, Christoph Segler, Alois Knoll, and Eric Sax. **Continuous Software Engineering of Innovative Automotive Functions: An Industrial Perspective**. In *IEEE International Conference on Software Architecture (ICSA): Companion Proceedings*, 2019. [14]

12. Lukas Heinzmann, Sina Shafaei, Mohd Hafeez Osman, Christoph Segler, and Alois Knoll. **A Framework for Safety Violation Identification and Assessment in Autonomous Driving**. In *Workshop on Artificial Intelligence Safety 2019 co-located with the 28th International Joint Conference on Artificial Intelligence (AISafety@IJCAI)*, 2019. [15]

The following patent applications are not directly part of the approach, but introduce concepts for the design and optimisation of automotive DDFs:

13. Christoph Segler. **Verfahren zum Steuern eines maschinellen Lernverfahrens einer Funktion eines Fahrzeugs, computerlesbares Medium, System, und Fahrzeug**. *Patent Application DE 102019119460 A1*, Filed on 18th of July 2019. [16]

14. Christoph Segler* and Thomas Hubregtsen* (*contributed equally). **System und Verfahren zur Bereitstellung einer Systemfunktion eines Fahrzeugs**. *Patent Application DE 102020104479 A1*, Filed on 20th of February 2020. [17]

15. Christoph Segler* and Hans-Jörg Vögel* (*contributed equally). **Verfahren, Vorrichtung, Computerprogramm und Computerprogrammprodukt zur Datenverarbeitung in einem Fahrzeug und Fahrzeug**. *Patent Application DE 102019117839 A1*, Filed on 2nd of July 2019. [18]

16. Christoph Segler, Thomas Hubregtsen, and Emmanuel Pollakis. **System und Verfahren für ein Fortbewegungsmittel**. *Patent Application DE 102019127802 A1*, Filed on 15th of October 2019. [19]

The following patent application and patent are not directly related to this work, but are associated with the work on this thesis:

17. Christoph Segler. **Verfahren zum Beheizen eines Kraftfahrzeugteils mit einer elektronischen Recheneinrichtung einer separaten Funktionseinheit sowie Kraftfahrzeug**. *Patent DE 102019114820 B3*, Issued on 23rd of July 2020. [20]

18. Mohsen Kaboli and Christoph Segler. **Method, System, and Computer Program Product for Controlling a Movement of a Vehicle**. *Patent Application EP 3702865 A1*, Filed on 28th of February 2019. [21]

Additionally, the following international workshops have impacted this work:

19. **1st International Workshop on Data Driven Intelligent Vehicle Applications (DDIVA) 2019**. Alois Knoll, Emec Ercelik, Esra Icer, Burcu Karadeniz, Christoph Segler, Sina Shafaei, and Julian Tatsch. Co-located with *IEEE Intelligent Vehicles Symposium (IV)*, 2019. [22]

20. **2nd International Workshop on Data Driven Intelligent Vehicle Applications (DDIVA) 2020**. Alois Knoll, Emec Ercelik, Esra Icer, Neslihan Kose, Burcu Karadeniz, Christoph Segler, Sina Shafaei, and Julian Tatsch. Co-located with *IEEE Intelligent Vehicles Symposium (IV)*, 2020. [23]

21. **International Workshop on Machines with Emotions: Affect Modeling, Evaluation, and Challenges in Intelligent Cars**. Alois Knoll, Sina Shafaei, Radoslaw Niewiadomski, Stefan Kugele, Christoph Segler, Morteza Hashemi Farzaneh. Co-located with *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019. [24]

Chapter 2

Background

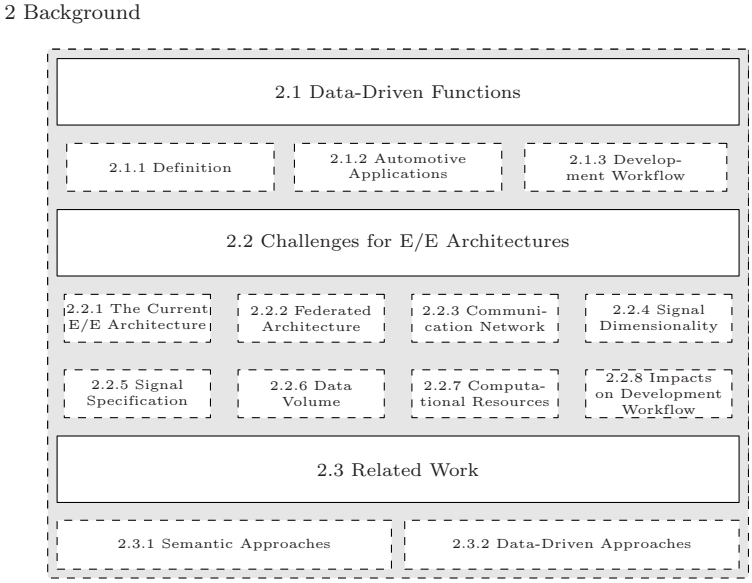


Figure 2.1: Chapter structure

This chapter presents the background and related work for the proposed approach (cf. Figure 2.1). The first section shows the background and the state-of-the-art of DDFs in the automotive domain. The first paragraph of this section introduces and defines DDFs, followed by an overview of automotive applications. The last paragraph presents the development workflow for DDFs. The second section presents the impacts of the automotive E/E architectures on DDFs and their development.[1] Here, the first paragraph introduces and defines automotive E/E architectures, followed by the impacts on DDFs. These impacts arise from (i) the highly

[1]Parts of the impacts of the automotive E/E architectures on DDFs and their development have been previously published at the 2019 IEEE ITSC (cf. [4]) and the 2019 IEEE IV (cf. [9]).

federated architecture, (ii) the limited access to signals, (iii) the used communication networks, (iv) the high dimensionality of signals, (v) the signal specification, (vi) the high volume of data generated by every vehicle, and (vii) the limited computational resources. The primary motivation of the presented approach is to minimise these impacts on DDFs and their development workflow. The last section of this chapter gives an introduction to the related work. For this work, we split the related work into *semantic approaches* and *data-driven approaches*. The approach in this work focuses on data-driven approaches and is also known as feature selection. The first paragraph defines the term *feature selection*, followed by the introduction of different feature selection methods. The last three paragraphs present three categories to cluster feature selection methods and the most applicable method in the case of automotive DDFs.

2.1 Data-Driven Functions

In the automotive domain, the term *function* relates to all functional characteristics of a vehicle. These functions either directly or indirectly affect the user. Each of these functions must provide an added value for the user and can either be realised as mechanical, hydraulic, electric, electronic or software system. [25, p. 2]

With the further integration of functions and more computational resources, more software-based functions are being used in current vehicles. Most of these software functions are still "hand-coded", but with the recent advances in machine learning, more and more functions become "data-driven".

2.1.1 Definition

In the automotive domain, functionalities follow the general pattern of measuring physical effects with sensors (e. g. force, button) processing this as input ($\boldsymbol{x}$), applying a functional logic ($f : \boldsymbol{x} \rightarrow \boldsymbol{y}$) on the input and controlling actuators based on the output ($\boldsymbol{y}$) from the logic [26, p. 16] (cf. Figure 2.2).

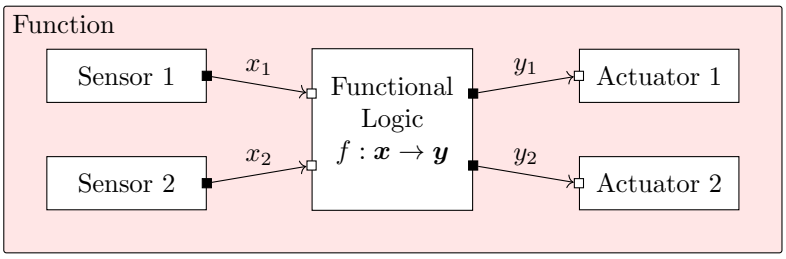


Figure 2.2: Function pattern in the automotive domain [26, p. 16]

The general pattern of classical automotive functions (f_c) and DDFs (f_{dd}) is similar to each other. Both measure an input $\boldsymbol{x}$, apply a function which is generating the output $\boldsymbol{y}$. In the case of classical automotive functions, the functional logic f_c is coded manually by a developer

based on the functional and non-functional requirements. However, in the case of DDFs the functional logic f_{dd} is "learned" using training data. Here, only the input $\boldsymbol{x}$ and the output $\boldsymbol{y}$ of the function is defined according to the functional and non-functional requirements and the logic f_{dd} is trained using a recorded data set. To "learn" f_{dd} and the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$, commonly machine learning algorithms are used.

2.1.2 Automotive Applications

With the advance of machine learning algorithms, data availability, and increasing functional complexity within the automotive domain, more and more DDFs are developed. The most prominent examples are autonomous driving functions [27]. Most of these functions process *raw* sensor data or more abstracted data, which is then used as input for another vehicle function (e. g. LIDAR based object detection [28], camera-based object detection [29], or map re-localisation using point clouds from LIDAR sensors [30]).

In contrast to the functions which are pre-processing raw sensor data, this work focuses on functions that use already processed vehicle data as input. In the automotive domain, there is a wide range of such applications. Most of these functions can be categorised into (i) system functions, (ii) context-aware functions, and (iii) anomaly detection. With the further development of DDFs, more types can evolve and gain further prevalence. The following paragraphs give an introduction into these types, including examples for each type.

System Functions

The first type of DDFs are *system functions*. These data-driven system functions are designed to model a part of the automotive system or a physical effect. In the automotive domain, a wide range of these functions have been developed, and with more computational resources available, the number will further increase. These can range from simple infotainment functions to hidden vehicle functions. For example these functions can be used for (i) data-driven battery management systems, which predict various battery parameters [31], (ii) air leak modelling and monitoring of combustion engines [32], (iii) intelligent active suspension systems [33], (iv) data-driven diagnosis of anti-lock braking systems [34], and (v) fault diagnosis for engines [35].

This work focuses on three exemplary use cases of these data-driven system functions for the evaluation of the proposed approach. One of these use cases is a prediction of the power consumption by all E/E components in the vehicle (cf. Chapter 4).

Context-Aware Functions

Another type of DDFs are *context-aware functions*. With the advance of intelligent systems, the customer demand for adaptive and smart functions increases. These functions learn according to the user's behaviour and capture hidden knowledge for continuously improving their capabilities and provide a highly personalised user experience and are also known as *context-aware*

systems [36].

In the automotive domain, these functions range from (i) intelligent window lifters [37], (ii) intelligent in-car infotainment systems [38], (iii) intelligent window wipers [39], (iv) proactive comfort functions [40], to (v) intelligent Adaptive Cruise Controls (ACCs) [41].

The principal functionality is already present in current vehicles. However, this basic functionality does not include any adaption to the user. As seen in the example of the intelligent ACC, many different data-driven approaches can be used to implement this functionality (cf. [41–45]). On the one hand, one can perform direct end-to-end learning of the output parameters of the vehicle's longitudinal positive and negative acceleration based on sensor data [41] or, on the other hand, one can perform learning of a single configuration value for a hand-crafted function which controls the longitudinal positive and negative acceleration of the vehicle [42].

In this work, we later use nine exemplary use cases of context-aware DDFs to evaluate the proposed approach. These functions range from Heating Ventilation Air Conditioning (HVAC) functions to driving dynamics functions. One of these use cases is to predict the desired temperature of the vehicle's interior and another use case is to predict the state of the ACC based on the user's preferences. More use cases used for the evaluation, and a further description of each use case can be found in Chapter 4.

Anomaly Detection

DDFs can also be used for *anomaly detection*. Anomaly detection is a process to find outliers on data by comparing with some predefined pattern or rules. In this work outliers are defined as "patterns in data that do not conform to a well-defined notion of normal behaviour" [46]. In recent years this problem receives significant attention and is researched in various fields as statistical analysis, and machine learning. *Time series anomaly detection* is studied and applied on different kinds of problems as detection of anomalous flight sequences using sensor data from an aircraft [47] or detection of outlier heartbeat pulses using ECG data [48].

In the automotive domain, anomaly detection is commonly used to recognise malicious attacks on the vehicle. Müter et al. [49] presented an attack recognition scheme for in-vehicle networks. This scheme is composed based on typical characteristics of automotive networks, which includes eight attack detection sensors. These predefined sensors serve as recognition criteria for automotive threats. Müter and Asaj [50] proposed an entropy-based anomaly detection approach. Baldoni et al. [51] utilised the network traffic data from the monitored systems to construct a failure prediction mechanism. They introduced failure prediction architecture (called CASPER) purposely to identify the abnormal behaviour of defined performance metrics. The approach by Taylor et al. [52] is explicitly designed to detect anomalies on Controller Area Network (CAN) buses with data-driven methods.

All these approaches only focus on the anomalies within the system, but not in the user's behaviour. For example, the exploit found by Hunt [53] allowed adversaries to set the temperature of the climate control remotely. From the car's perspective, the malicious requests looked

perfectly valid, although a driver would never set the temperature to a level, where they would freeze or sweat. Luckily, the safety impact of such a vulnerability is minimal. However, as we move towards higher levels of automation in autonomous driving, we see an increase in functionality that either assists or takes over in both normal and emergency scenarios. Attacks on such functionalities could have more severe impacts. For example, a driver can deactivate such functions manually (for personalisation). A deactivation per se is not a problem if intended by the driver. For instance, a driver might disable traction control when being stuck in an iced parking lot. However, if such a function is deactivated while driving at high speed due to a software or hardware fault, or an IT attack, this could lead to a severe problem. In addition to already applied methods at design time, we propose an anomaly detection function for these Advanced Driver Assistance System (ADAS).

This proposed anomaly detection for different ADAS functions are used as test cases later in this work. The anomaly detection is here used for twelve different ADAS functions, which are further described in Chapter 4.

2.1.3 Development Workflow

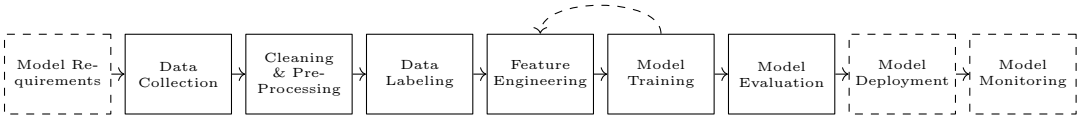


Figure 2.3: Development workflow for data-driven functions (cf. [54])

The development of these DDFs is similar to the development of machine learning applications. In the last decades, various development workflows for machine learning functions have been introduced (cf. [55–61]). Based on these, Amershi et al. [54] proposed a simplified version of this workflow with nine stages linearly arranged (cf. Figure 2.3). This workflow can also be easily adapted for the development of DDFs.

In the first stage, the developers specify the functionalities and additional requirements of the function. In the next step, the *data collection* for the training and design of the function is conducted, or already available data is searched. The next step is the *cleaning and preprocessing* of the data. This includes removing corrupt data points, decoding and resampling data. In the next step, the *data labeling*, the ground-truth labels for the training of the data-driven model are generated. These labels can either be manually generated by the developer or automatically generated by a set of rules. This step is not necessary for unsupervised models. During the stage of *feature engineering*, the input features for the model are selected, and/or new features are generated based on other features.[1] In the case of data-driven vehicle functions, the input features are mostly based on vehicle signals, and we consider the vehicle's signals as

[1] In data mining or machine learning, information which can be used as input for an algorithm is referred to as a *feature*

features. Based on the pre-processed data, generated labels, and selected features, a chosen machine learning model is trained.

In some cases, the feature engineering and model training have to be iterated multiple times as the features have to re-engineered after training. After the training, the model is evaluated based on the model requirements of the first step. If the model fulfils all conditions, the model is deployed and continuously monitored.

As this thesis focuses on the data collection and pre-processing during the development of DDFs, only the steps from *data collection* to *feature engineering* are further investigated. But in order to evaluate the proposed approach also the steps *model training* and *model evaluation* have to be implemented and executed.

2.2 Challenges for E/E Architectures

All the DDFs mentioned before rely on vehicle data which originate from the vehicle's E/E architecture. This architecture defines the technical realisation of vehicle functions and thereby has a significant impact on the development of DDFs and their input data. Today's E/E architectures are highly cost-optimised, federated, highly integrated, and not optimised for DDFs or functions realised using machine learning. This section introduces the current automotive E/E architecture and the major challenges for the development of automotive DDFs.

2.2.1 The Current E/E Architecture

The E/E architecture defines the physical topology, the network of the hardware components and the functional network of all electrical and electronic vehicle components [62, p. 19]. The main purpose of the automotive E/E architecture is to provide or to improve the functions of a vehicle. Here, the term *functions* refers to all functional characteristics which are part of the vehicle. These can either be directly or indirectly perceived by the user (or driver) and represent a benefit for the user. [25, p. 2]

In today's vehicles, most of these functions are realised as a mechatronic system which is executed on an ECU. In current E/E architectures, a domain-based clustering approach has been established [63, p. 9]. In this design approach, all vehicle functions of the same functional domain are clustered into segments [64, pp. 54ff]. Typically these domains are clustered in a hierarchical structure. Figure 2.4 shows an example of a domain structure. Here the vehicle's functionalities are clustered into the five main domains *drive-train*, *chassis*, *body*, *ADAS*, and *infotainment*. Each domain is then again split into multiple domains or functional clusters.

Based on these domains, the E/E architecture is then designed. Figure 2.5 shows a simplified example of such an architecture. Every dark-grey box represents an ECU on which vehicle functions are executed. With the advance of cross-domain functions and optimisation of the architectures, these domains are interconnected between each other [65, p. 12]. All ECUs of one domain are interconnected between each other, utilising a particular bus technology. Some

functions require cross-domain connections as they either need sensor data of another domain or control actuators in another domain. For the communication between the domains, each domain includes a gateway which is again connected to the gateways of the other domains.

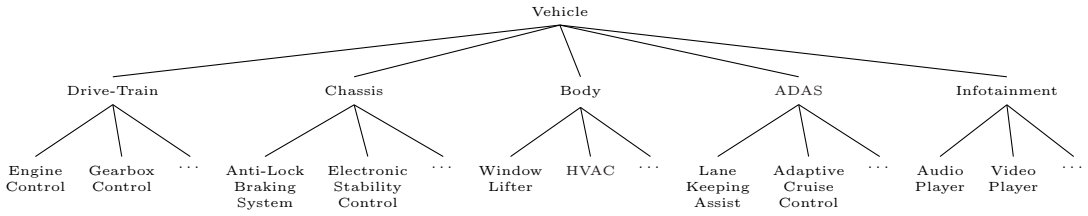


Figure 2.4: Example for domain clustering (cf. [66, p. 32])

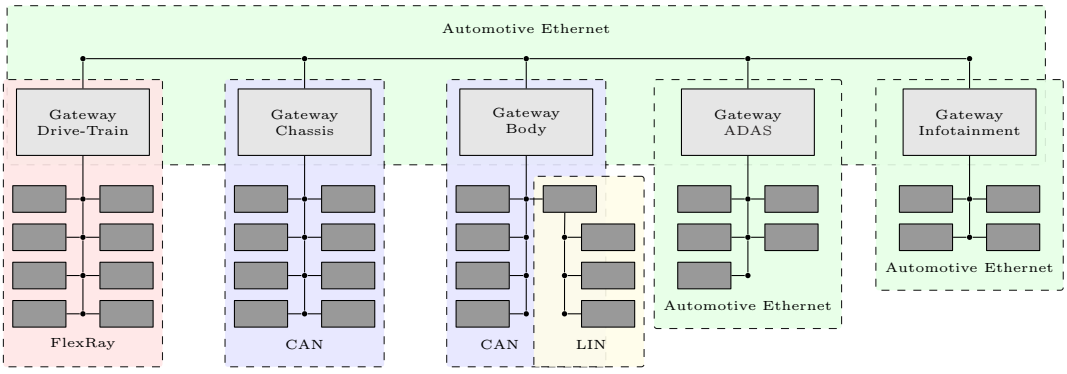


Figure 2.5: Exemplary domain based E/E architecture (based on [65, 67, 68])

2.2.2 Federated Architecture

Highly federated E/E architectures are a major challenge for DDFs. Current architectures consist out of more than 70 ECUs [69], where each ECU senses and generates different data. Due to the domain structure of the E/E architecture, there is no central point to retrieve or access all available data from multiple domains. However, most algorithms require all input signals present in one place, which is where the DDF is executed.

Besides, some signals are only processed within an ECU and not even sent out to another ECU. This is the case if the signal is solely required by a function partitioned on the same ECU and no other function on another ECU requires this signal. In this case signals are not sent over the vehicle's communication networks and cannot be easily accessed. Therefore these signals are also called ECU internal signals.

Forwarding all messages across the domains would lead to major communication overhead and is therefore not reasonable. This has an impact on the selection of input signals used by

the DDF and also on the selected algorithm. This leads to the following two challenges when designing an automotive DDF:

Challenge 1 (Federated Architecture)
Due to the highly federated structure of the E/E architecture, there is no central point for data retrieval or data analytics.

Challenge 2 (Internal Signals)
Some signals are only available as an internal signal on a single ECU and cannot be accessed from outside of this particular ECU, as it has not been designed for external access.

2.2.3 Communication Network

Another challenge arises from the communication networks. To exchange data between multiple ECUs current architectures mostly apply linear topologies, as seen in Figure 2.5 [25, p. 88], which are again connected over multiple gateways. As communication technology mostly serial bus technologies (e. g. CAN [70], Local Interconnect Network (LIN) [71], FlexRay [72]) are used. For the transmission of vehicle signals, multiple signals are clustered into one Protocol Data Unit (PDU) (cf. Figure 2.6). These PDUs are then clustered into a single frame and then send over the network, including the header specific for the communication technology [25, pp. 89f].

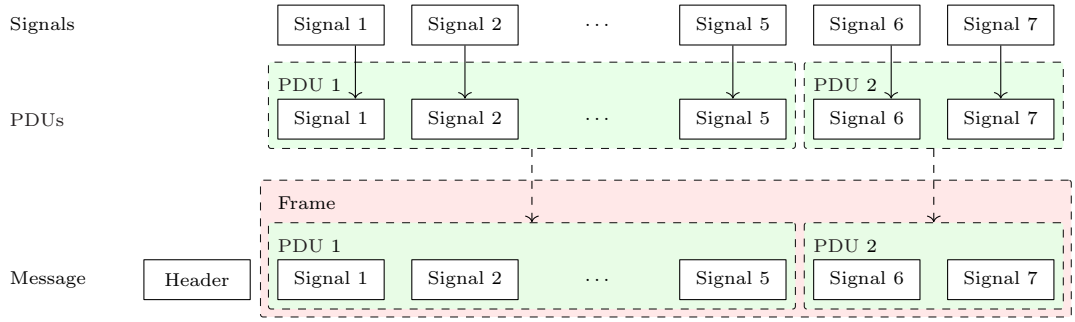


Figure 2.6: Communication Frame Structure [73, p. 268]

This clustering results in delays in communication, as all signals within one message might not be measured simultaneously. Additionally, these vehicle signals arrive asynchronously with different sampling rates, if not within the same PDU. Most machine learning algorithms, however, require synchronous input data, which becomes more complex, the more signals are being used as input. This leads to the following challenge:

Challenge 3 (Asynchronous Communication)
Vehicle signals arrive asynchronously with various sampling rates.

2.2.4 Signal Dimensionality

Another challenge arises from the high number of signals. Current vehicles exchange up to 14 000 signals on the internal communication networks (not including any radar or camera data), and it is not reasonable to use all signals as input for a single DDF. This high number (i. e. high dimensionality) of signals accessible in the vehicle leads to the *curse of dimensionality* [3]. This *curse* relates to the problem of having a too large amount of data that may contain irrelevant, redundant, or not suitable data yielding a reduced accuracy and training efficiency. This does not only affect algorithms, but also the *function developer* who has to select the appropriate input signals for the DDF, and therefore, might miss relevant signals which are describing the system's or user's behaviour. This leads to the following challenge:

Challenge 4 (Signal Dimensionality)
High dimensionality of vehicle signals available within the vehicle.

2.2.5 Signal Specification

Even if all challenges mentioned earlier would be solved, function developers have difficulties in selecting and pre-processing signals due to their incomplete specification. These include missing data quality specification and missing semantic information.

In current E/E architectures, all vehicle signals are designed for a particular function and are not indented to be used by a third function. Each signal is manually specified by a developer to be used by the desired function, including the data quality. According to ISO 19157 [74] data quality can be specified by six attributes:

Temporal Quality
: Is the content temporal valid, consistent and time measurement accurate?

Logical consistency
: Is the information consistent to a conceptual schema, value domains, format and encoded topology?

Usability
: Is the content usable as required by function?

Accuracy
: Is the position of features accurate to true values, relative positions, or spatial positions?

Completeness
: Is the data incorrectly present or data missing in data set?

Thematic Accuracy
: Are quantitative attributes accurate and non-quantitative attributes correct?

These quality attributes are mostly only implicitly and not explicitly specified in the signal's documentation. This implicit specification raises an additional challenge for developers without any expert-knowledge regarding a particular vehicle signal.

Also, with a vast amount of sensors embedded in most modern cars, many of them are obscure to non-automotive experts and rely on non-standard units [75]. This approach worked well for classically designed vehicle functions. However, in the case of DDFs, input signals are not static, might change over time and might use data from a sensor initially implemented for a non-related function. In this case, current specifications are not sufficient for a developer seeing the signal for the first time to fully understand its content.

Besides, vehicle signals have diverse data structures. As an example, when looking at the specification of SOME/IP [76]—one of the major automotive Ethernet protocols—the following data types can be used for vehicle signals: Basic datatypes like boolean, uint8, uint16, float16 and also more complex datatypes as strings, arrays, enumerations, bit fields, and union of different data types. This shows the complex and diverse structure of vehicle signals. The incomplete specification and the diverse data structure leads to the following challenge:

Challenge 5 (Signal Specification)
Signals are not fully specified in a machine-readable format, and some attributes are only implicitly specified. In addition, the signals are specified in a diverse data structure.

2.2.6 Data Volume

Another main challenge derives from the high volume of data itself. Currently, each vehicle produces up to 25 GB of data per hour [2], and this amount will further increase. Vehicle data can be clustered in four different data usage categories, based on the *NEVADA*[1]*-Share & Secure* [77] concept by the Verband der Automobilindustrie e.V. (VDA):

Category 1 Vehicle data for improved traffic safety
(traffic safety relevant data),

Category 2 Vehicle data for cross brand services
(non-differentiating vehicle data),

Category 3a Vehicle data for brand specific services
(Data differentiating and Intellectual Property (IP) relevant for OEM),

Category 3b Vehicle data for component analysis and product improvement
(Data differentiating and IP relevant for OEM and supplier), and

Category 4 Personal vehicle data.

[1]Neutral Extended Vehicle for Advanced Data Access

The data of all categories are accessible through the OEM's back end defined by the Extended Vehicle (ExVe) concept in ISO 20077 [78] and ISO 20078 [79] (e. g. *BMW CarData* [80] or *Mercedes-Benz Data* [81]). This allows not only the OEM to use the vast amount of vehicle data without altering the security and safety of the vehicle, but also third parties.

Data of categories 1 and 2 must be readable and usable by different parties (e. g. governmental institutions) and must be well-specified. This categories only include a fraction of the actual available vehicle data. This work will focus on the data category 4 for the personalisation of vehicle functions. These categories hold the most information about the vehicle, the user and its context. Also, this data is OEM-specific and, therefore, might not be entirely specified.

This high data volume raises a major challenge not only for the development of DDF but also for other data mining applications. On the one hand, the vehicle has only minimal resources for storing data, and it would also not be reasonable. On the other hand, transmitting all data would not be cost-efficient or desired by the customer, even if technically possible with current cellular networks (e. g. 5G with 1.2 GiB/s [82]). Due to the high amount, not all data can be transferred to the back end, and it has to be pre-selected which data should be transferred. Currently, this pre-selection must be performed based on expert knowledge or based on requests for certain data points either by internal developers or external requirements. These limited resources for storing and transmitting data shows the need for an efficient onboard data selection strategy and leads to the following challenge:

Challenge 6 (Data Volume)
The high data volume produced by a vehicle cannot be easily transferred or stored onboard the vehicle.

2.2.7 Computational Resources

Additionally, current E/E architectures are highly optimised, not only in regards to data but also regarding computational resources. All extensive computational executions would not only add cost for additional hardware but would also have a significant impact on the power consumption of the onboard electronics. As the only power supply in the vehicle is the battery, current architectures try to minimise the used computational resources and try to offload heavy computations into the back end (e. g. *Control as a Service* concept [83]). This restricts the deployment of extensive computational algorithms onboard the vehicle, even if they would lead to the best performance. Therefore, the E/E architecture raises the following challenge for the development of DDFs:

Challenge 7 (Computational Resources)
The computational resources for data processing within the vehicle are highly limited.

2.2.8 Impacts on Development Workflow

These challenges for the development of DDFs have a significant impact on the development workflow (cf. Section 2.1.3). Figure 2.7 gives an overview of the challenges, and on which step of the workflow, they have the biggest impact on.

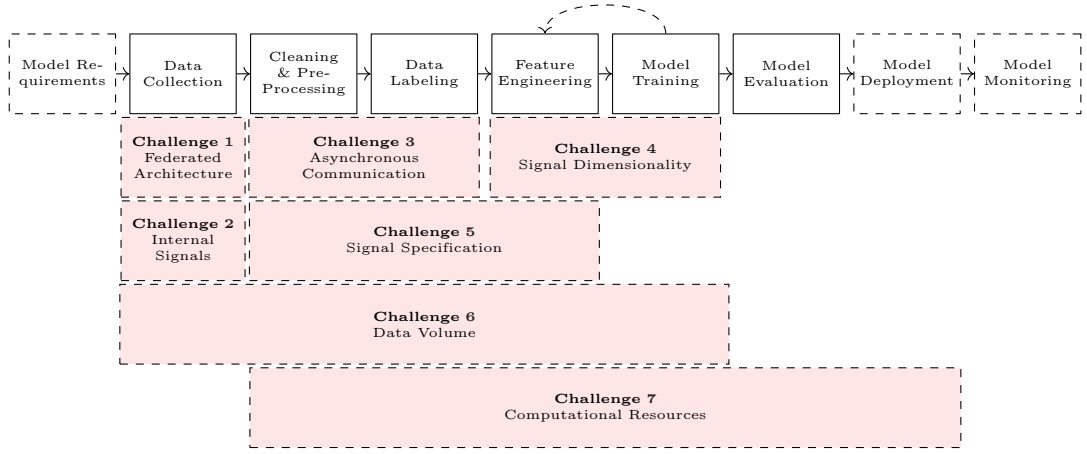


Figure 2.7: Impacts by challenges from E/E architecture on development workflow

The *federated architecture* (cf. Challenge 1), as well as the *internal signals* (cf. Challenge 2), have a direct impact on the data collection step in the development workflow. Here it is challenging to collect all available data, or it might even be impossible to do so, as there is no central point for data collection, and there might be signals not accessible.

The third challenge raised by *asynchronous communication* (cf. Challenge 3) has a direct impact on the data cleaning and data labeling step. During the data cleaning and pre-processing step, all incorrect data have to be removed and resampled. As the signals arrive asynchronously, it is challenging to know the temporal quality (i. e. age) of the present data point and by this it is not always clear if a data point needs to be removed or not. In the case of the data labeling, the asynchronicity raises a challenge if rules directly generate the label. As the label represents the ground-truth for the training of the model, this information should always be correct. But for example, if the label has been changed, but this information arrives after another signal, the ground-truth is incorrect in relation to the other signal. This would then lead to a false ground-truth and might have a significant impact on the performance of the trained model.

The high *signal dimensionality* (cf. Challenge 4) has a significant impact on the feature engineering and model training step. In the feature engineering step, the developer has to select the appropriate input signals for the DDF, and therefore, might miss relevant signals which are necessary to represent the behaviour the function should represent. During the model training, the high dimensionality of signals accessible in the vehicle leads to the *curse of dimensionality*, mentioned before.

On the other side, the challenge of missing machine-readable *signal specifications* (cf. Chal-

lenge 5) has a direct impact on the data cleaning, data labeling, and feature engineering step. For the data cleaning, the current specifications of signals are lacking the information which values of a signal contain actual information and which values should be only used for debugging. For the cleaning step, it is mandatory to have this information for all processed signals. During the data labeling step, this information is only needed for signals used to generate the label. In most cases, only a little number of signals are used here, and the missing specification has only a limited impact. In the case of feature engineering, the missing information about the data quality has a significant effect. All used input signals for the training of the model have to be selected and specified. The not fully defined data quality raise the challenge for the developer to decide if the quality of used signals fulfils all requirements of the DDF or not.

However, the biggest impact derives from the high *volume of data* (cf. Challenge 6). The high volume of data produced by every single car not only makes it challenging to transfer all data to a back end or the cloud but also to store all data (even within the vehicle). Even if it were possible to store or transfer all data, this would not scale in the next workflow steps as this massive volume of data must be processed.

In the case of onboard processing, the *computational resources* are limited and require lightweight algorithms (cf. Challenge 7). The highly limited computational resources affect not only the data processing but also the model training and deployment. All these limited resources for storing, transmitting, and processing data show the need for an efficient onboard data selection strategy during the whole workflow.

As this thesis focuses on the data collection and pre-processing during the development of DDFs, only the steps from *data collection* to *feature engineering* are further investigated. As seen in Figure 2.7, all challenges have a direct impact on these steps. In the later shown approach, we try to assess these challenges while helping the developer to identify the needed signals for a particular DDF.

2.3 Related Work

The works cited in Section 2.1 solely focus on the implementation of a specific DDF and rely on a hand-picked set of input signals. As seen in the example of an intelligent ACC, many different data-driven approaches can be used to implement this functionality [41–45]. On the one hand, one can perform direct end-to-end learning of the output parameters of the vehicle's longitudinal (positive and negative) acceleration based on sensor data [41] and, on the other hand, one can perform learning of a single configuration value for a hand-crafted function which controls the longitudinal acceleration of the vehicle [42]. In all cases, the input signals have been fixed and hand-picked in the development phase. This work focuses on a general approach for the identification and discovery of vehicle signals which are already present within the vehicle and can directly be used as input for the DDF.

In the state-of-the-art in data mining and machine learning, we can differentiate between two

methods for identification and discovery of such signals. The first method relies on semantic annotation of all vehicle signals and thereby can ease the discovery. The other method is purely data-driven and mostly uses the present data to identify correlations and thus gives the developer insight on potential input signals. To the best of our knowledge, we are not aware of any related work that considers the characteristics of automotive E/E architectures and focuses on discovering relevant vehicle signals for a variety of DDFs (or classical functions).

2.3.1 Semantic Approaches

For the semantic annotation of vehicle signals, the de facto state-of-the-art is the Vehicle Signal Specification (VSS) [84]. This structure was originally introduced and specified by the World Wide Web Consortium (W3C) and the GENIVI Alliance and is one of the building blocks of the Vehicle Information API Specification (VIAS) [85] and Vehicle Information Service Specification (VISS) [86] which are still under development. In addition to the VSS many other semantic representations of vehicle data have been introduced (e. g. [87–91]), and are mostly focused on ADAS. [75]

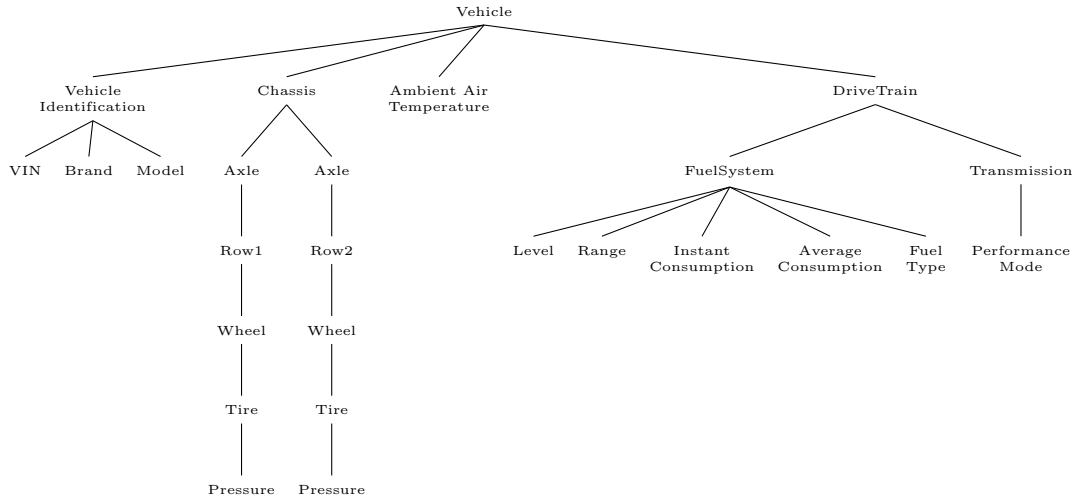


Figure 2.8: Exemplary snapshot of a VSS tree [84]

Figure 2.8 depicts an exemplary snapshot of the VSS tree. The root-node is representing the *vehicle* with linked nodes for different domains and vehicle parts. For example, in the case of the drive-train, it consists of the fuel system and transmission. Finally, these components contain data points as last "leaf". For example, the fuel system contains the following data points: *level*, *range*, *instant consumption*, *average consumption*, and *fuel type*. Each of these data points can be addressed using the structure of the VSS tree. For example, the information about the fuel type can be obtained with *Vehicle.DriveTrain.FuelSystem.FuelType* and it will return the current value and how the information is generated, the data type, the unit, the minimum/maximum value, and a short description.

Klotz et al. [92] proposed in their work an ontology-based on the VSS, called Vehicle Signal Specification ontology (VSSo). The semantic annotation of vehicle data modelled with this ontology allows the developer, for example, to query for data required as input for a DDF. By this, the developer can find appropriate vehicle signals if they know what to search for. But especially in the case of DDFs mimicking a user, this information is often hidden. For example, if the developer has to develop a function which learns if the driver wants to drive in *sport mode* or *comfort mode*, it is not easy to determine which vehicle signals are the appropriate input to describe this behaviour, even with expert knowledge. Later in this work, we implemented such a DDF and we identified the following vehicle signals as the most correlating: The weekday, road-type, temperature, and the information about the presence of a co-driver. Such correlations might be obvious when seeing them, but in the first place, one might not consider these signals. Such correlations can only be found based on studies or using data-driven methods.

Besides, the current specification of vehicle signals does not allow automatic semantic annotation of all vehicle signals and the integration into a semantic structure like the VSS. This process still would require manual annotation of each signal and would not scale with the current amount of around 14 000 signals. Even if all vehicle signals or data points would be modelled in a structure like the VSS, the amount of data is still massive and requires a lot of effort to identify the desired signals.

2.3.2 Data-Driven Approaches

The alternative to the semantic approaches are data-driven approaches. These approaches purely rely on the content of each signal and do not need semantic information on each signal, which is the case for automotive data (cf. Challenge 5).

In data mining, *feature selection* algorithms are widely used to resolve the curse of dimensionality and identify potential input signals/features. Hall gives one of the most intuitive definitions for feature selection: "Feature subset selection is the process of identifying and removing as much irrelevant and redundant information as possible. This reduces the dimensionality of the data and may allow learning algorithms to operate faster and more effectively" [93]. A wide range of applications apply this method to reduce a large amount of irrelevant or redundant information. In early applications, only a few domains used more than 40 features [94]. There was a rapid change, and nowadays, data mining applications deal with up to hundreds of thousands of features [95]. Another typical application of feature selection besides machine learning is the gene subset selection in medical applications. Here, algorithms find a subset of marker genes for the identification of diseases (e. g. [96]). Other applications use feature selection in a variety of domains: Distributed P2P networks [97], data mining for vehicle maintenance [98], or context reasoning [99]. In this work, this method plays a key role in the proposed approach.

In the last years, a variety of algorithms have been developed. These algorithms can be categorised based on four different characteristics: This can either be the general methods of the feature selection, the type of the input labels, the type of input data, or type of the

features [100]. In the following chapter, we further describe this categorisation and compare the applicability to automotive applications, including the characteristics of the E/E architecture.

Feature Selection Methods

One way of categorising feature selection algorithms is based on the general method. Here, algorithms can be categorised into *filter*, *wrapper*, or *embedded methods* [100, 101]. Figure 2.9 gives an overview of the underlying methods. In the figure, the top-down arrows represent data/signal flows and the bold arrows the transition and exchanged information to the step/iteration of the algorithm.

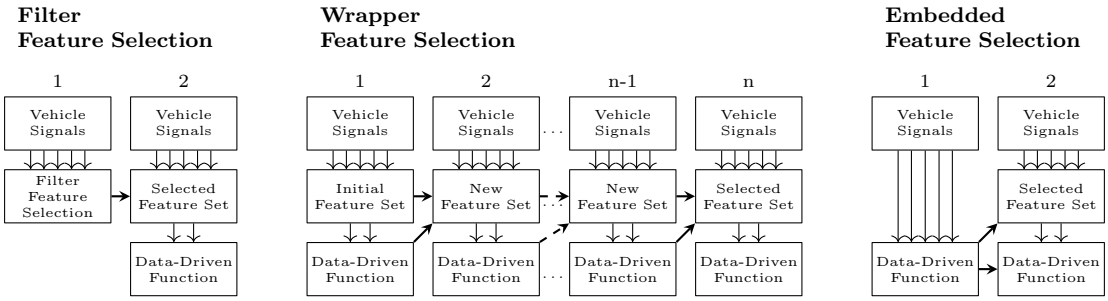


Figure 2.9: Overview of feature selection methods on vehicle data

Filter feature selection methods rely only on the input data and are independent of any DDFs. Typically filter feature selection methods are executed in two steps (cf. Figure 2.9). In the first step, all features (in our case vehicle signals) are ranked based on a feature evaluation criteria. Based on this ranking, the most important features can be used for the DDFs and low ranked features are filtered out. The feature evaluation criteria can vary from algorithm to algorithm. Li et al. [100] categorised the used criteria into feature discriminative ability (e. g. [102]), feature correlation (e. g. [95, 103]), mutual information (e. g. [104]), feature ability to preserve data manifold structure (e. g. [105–107]), and feature ability to reconstruct the original data (e. g. [108, 109]). A major advantage of these algorithms is the independence to the existence of a DDF and thereby can be executed at any time; even if no function has been developed yet.

Wrapper feature selection algorithms follow a different approach. In general, these algorithms select an initial feature set in the first step (cf. Figure 2.9). This feature set is used to create and evaluate the DDF. In the next step, another feature set is selected, and again the performance of the function is evaluated with the newly selected feature set. This step is repeated until a predefined stop criterion has been reached. As a final step, the last selected features are used as input for the final DDF. For the selection of the new feature set, many different strategies have been implemented, such as sequential search (e. g. [110,111]), or genetic algorithms (e. g. [112]). A major advantage of these algorithms is the integration of the DDF

into the feature selection step. However, this can only be applied if the function is already developed. According to Li et al. [100] these methods are rarely used due to the large search space in high dimensional data sets.

In contrast to the methods mentioned above, *embedded feature selection* methods are directly embedded in the underlying algorithm of the DDF. The methods use all features as training input for the function, and the algorithm automatically selects the appropriate features during/after the training process. A typical example is the use of deep neural networks, where input edges below a certain threshold are removed (pruned) after the training step and thereby, features are removed [113]. This incorporates the advantages of filter feature selection and wrapper feature selection methods but needs all available features during the training.

When applying feature selection onboard the vehicle, the computational resources are highly limited (cf. Challenge 7). Wrapper feature selection methods cannot be used in the here presented environment, as they tend to be computational inefficient, due to their the large search space. Filter and embedded feature selection methods require less computational resources. However, in our case, a DDF is not developed yet, and the function developer is still in its design process. This also includes the definition of the appropriate inputs. Here, filter feature selection methods are most suitable, as they only rely on the present data and have no need for an already implemented function. These methods can run without any DDF generating output and provide a subset of relevant signals as a result. This subset can be directly read by a function developer and allows integration of these specific signals into the function. In the case of embedded feature selection methods, the algorithm requires full access to all vehicle signals during the selection and training process. This is not possible within current E/E architectures (cf. Challenge 1), and the result can hardly be read by a function developer as the selection happened within the trained model. This work will therefore focus solely on filter feature selection methods.

Input Labels

Feature selection algorithms can also be classified based on the use of label information. Label information holds the information of the prediction the model should make, in our case the output of the DDF. Depending on the use of this label information, feature selection algorithms can, in general, be clustered in *unsupervised feature selection*, *supervised feature selection*, and *semi-supervised feature selection* [100].

Unsupervised feature selection does not use any label information for the selection step. The most common examples of unsupervised feature selection are Principal Component Analysis (PCA) (e. g. [114,115]), and auto-encoders (e. g. [116]). These algorithms are mainly used when no label information is available, or where labeling is very costly.

In contrast, supervised feature selection use the label information for the selection by usually calculating the correlation of the features to the label, or in the case of wrapper algorithms the prediction performance of the model. These algorithms require for all collected data points the

full label information, which in some cases is not possible to collect or generate. Semi-supervised feature selection algorithms are designed to run on partly labelled data. This is often the case on sparse data or when labeling is costly and cannot be performed for all data points.

In the case of automotive DDF, the use case of the function is already specified and based on the high amount of vehicle data, it is easy to create the label information for the desired function automatically. Therefore, we focus in this work on supervised feature selection algorithms, which use label information automatically generated based on vehicle data.

Input Data

For all feature selection algorithms, the input data is crucial. The type of input data for feature selection algorithms can be clustered into *static data* and *streaming data* [100].

In classical data mining and machine learning, static data is used as input. Static data does not change over time and is entirely available during the complete feature selection and model training process. This is true for most data-driven applications. However, with the advance of big data, there is a steady flow of newly arriving data, which cannot be completely stored. This is related to as streaming data. Streaming data raises additional challenges to feature selection algorithms, as the input data cannot be loaded entirely into memory nor stored. In this case, the algorithm has to process all incoming data and discard all values afterwards and still should be able to select the most appropriate features.

In the case of E/E architecture, most of the vehicle data cannot be stored and have to be processed on the fly (cf. Challenge 6). In our work, we consider the input data as *streaming* in the case of onboard data and as static if we consider already stored data in the back end.

Input Features

Similar to the input data, the input features can also be classified as *static features* and *streaming features* [100].

Again in classical feature selection, only static features are considered; this means the feature set which is evaluated does not change over time and is completely known at the beginning of the feature selection process. In other applications, it is common that streaming features have to be evaluated, which increase or change over time (e. g. [117]). These applications raise additional challenges for the feature selection algorithm. In contrast to the input data, input features can have a structure among each other (e. g. groups [118,119], graphs [120], or trees [121]). Feature selection algorithm can use this knowledge as additional information for the selection process.

In the case of the E/E architecture, all vehicle signals are defined during vehicle development and do not change over time. Moreover, no explicit structure between the signals is modelled and thereby cannot be used as additional information (cf. Challenge 5). Therefore, we consider the input features in our case as *static*.

Algorithms Used in this Work

This work will focus on supervised filter feature selection algorithms, with static features. These algorithms are the most suitable feature selection algorithms in consideration of the challenges raised by the E/E architecture and can be executed when no DDF is implemented yet. In addition, we will consider input data as *streaming* in the case of onboard data, and as *static* when the data is already stored data in the back end.

The presented approach heavily relies on the performance of the used feature selection algorithm. To minimise this effect in the evaluation of the approach, we will use 15 different state-of-the-art feature selection algorithms. The selection of the algorithms is based on the extensive survey paper by Li et al. [100]. The used algorithms are listed in Table 2.1, including its reference. By using different algorithms, we can also identify the best-suited algorithms for automotive signals—at least for the presented test cases.

Table 2.1: Feature selection algorithms used in this work

Algorithm	Full Name	Reference
CIFE	*(Conditional Infomax Feature Extraction)*	[122]
CMIM	*(Conditional Mutual Information Maximization)*	[123]
Chi^2	-	[124]
DISR	*(Double Input Symmetrical Relevance)*	[125]
FCBF	*(Fast Correlation Based Filter)*	[126]
FScore	-	[127]
Fisher Score	-	[128]
Gini Index	-	[129]
ICAP	*(Interaction Capping)*	[130]
JMI	*(Joint Mutual Information)*	[131]
MIFS	*(Mutual Information Feature Selection)*	[132]
MIM	*(Mutual Information Maximization)*	[133]
MRMR	*(Minimum Redundancy Maximum Relevance)*	[134]
ReliefF	-	[135]
Trace Ratio Fisher	-	[136]

Chapter 3

Approach

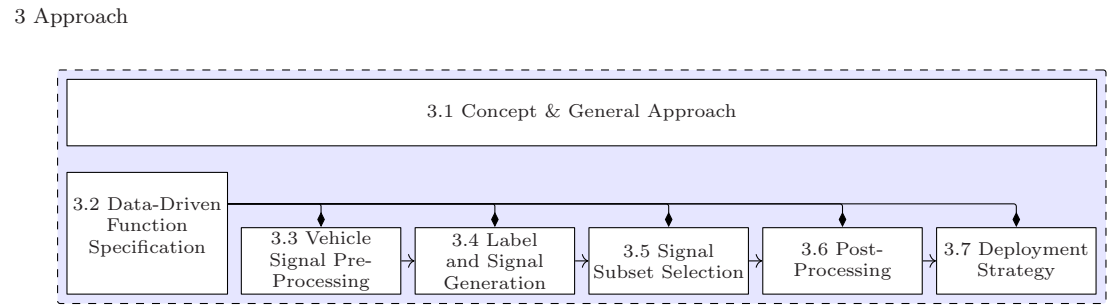


Figure 3.1: Chapter structure

The approach, which aims to answer the RQs, is presented in this chapter. The first section gives an overview of the concept and the general approach of the presented method.[1] The following sections describe the approach step by step (cf. Figure 3.1).

3.1 Concept & General Approach

The general idea of the approach is depicted in Figure 3.2. The *perceivable context* is at the core of the approach: both the *user* U and the *vehicle* V can perceive information from the context C. In this work, context is defined according to the definition of Dey: "Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves" [137, p. 4].

We distinguish between different *types* of perceivable context: On the one hand, the user can perceive a specific set of information from the context upon which actions are taken (C_U). For

[1] The general concept of the approach has been previously published at the 2019 IEEE ITSC (cf. [4]).

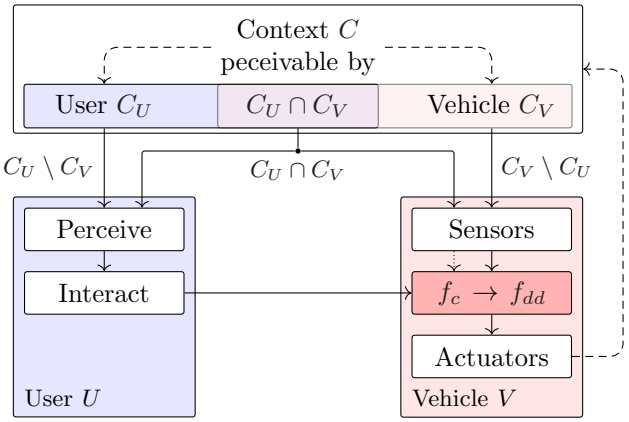


Figure 3.2: General concept of the presented approach*

*Figure previously published at 2019 IEEE ITSC (cf. [4]) ©IEEE 2019

example, the user can perceive the current type of road the vehicle is travelling on. On the other hand, the vehicle can perceive a set of information from the context with its built-in sensors (C_V). For example, the vehicle can measure the current speed of the vehicle. This context information does not only include environmental information but also information about the vehicle's interior, including vehicle function states. The context perceived by the user is not necessarily similar to the set of context information perceivable by the vehicle and vice versa. For example, the emotions of the co-driver can only be perceived by another passenger and not by the vehicle ($C_U \setminus C_V$). Furthermore, for example, the night vision systems in vehicles can detect persons at night, which the user cannot "see" ($C_V \setminus C_U$). However, there is also context, $C_U \cap C_V$, which both the user and the vehicle can perceive/measure. For example, the user can "feel" the temperature within the vehicle and the vehicle itself can measure it with a built-in sensor, even though in different units or scales.

This approach focuses on interactions of the user with a specific classically designed vehicle function f_c based on information the user perceives from the context (C_U). The vehicle function f_c then controls specific actuators based on user interactions and vehicle sensors, which, in turn, changes the vehicle's context and the environment.

In the initial state of the discovery phase, the function f_c only uses sensor input predefined by the function developer. Hence, only this particular part of the context is being used as input. However, it is not necessarily overlapping with the context the user is perceiving. The idea of the approach is to identify the *subset* $S_\mathcal{A}$ (i. e. subset of all vehicle signals) of context information $C_U \cap C_V$, on which users are making their decisions and coincidentally is already measured by built-in vehicle sensors from another function and is not yet used by the corresponding function f_c. This signal *subset* $S_\mathcal{A}$ out of all vehicle signals S describing $C_U \cap C_V$ is then used as an additional input for the function f_c and allows this function to learn the user's behaviour and proactively act based on the learned behaviour. This function is then referred to as *DDF* f_{dd}. Vice versa, when discovering signals describing a system's behaviour, we observe the interaction

of the system with a vehicle function instead of the user's interaction.

In order to discover the afore-mentioned context $C_U \cap C_V$ and the signal subsets describing this context, we use the following approach: First of all, let S be the set of all available vehicle signals s_i (i. e. $S = \{s_1, \ldots, s_i, \ldots, s_m\}, 1 \leq i \leq m$), where m represents the total number of vehicle signals. Let $S_\mathcal{A}$ be the subset of vehicle signals out of S, which represent $C_U \cap C_V$ for a specific classically designed function f_c. This function f_c is the basis for a new DDF f_{dd}. The goal of the approach ($f_\mathcal{A}$) is to select this signal subset $S_\mathcal{A}$ of size n (i. e. $f_\mathcal{A} : S \to 2^{S_\mathcal{A}}, S_\mathcal{A} \subseteq S, |S_\mathcal{A}| = n, |S| = m, m \in \mathbb{N}^+, n \in \mathbb{N}^+$), which then can be used as the appropriate input for the new DDF f_{dd} and represents the context $C_U \cap C_V$. An overview of this proposed approach is depicted in Figure 3.3.

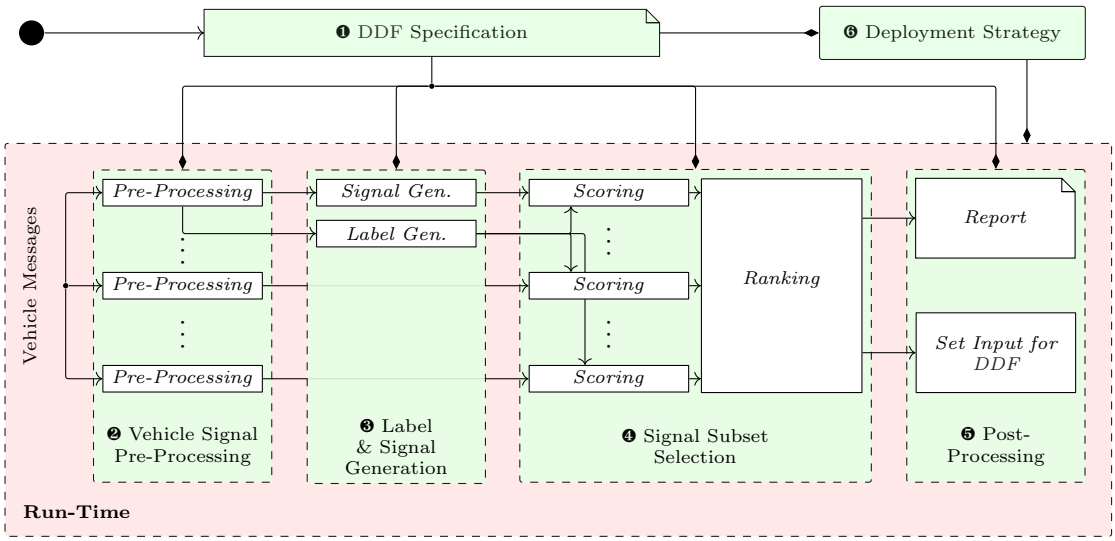


Figure 3.3: Flowchart of general approach

In the first step (❶), the *DDF specification*, the developer has to specify various attributes of the function. This includes the label generation (i. e. the output vector $\boldsymbol{y}$ of the DDF), a whitelist/blacklist of possible input signals (i. e. the input vector $\boldsymbol{x}$ of the DDF), the number of maximum input signals $|S_\mathcal{A}| = |\boldsymbol{x}|$, and the post-processing of the selected signal subset $S_\mathcal{A}$. This specification is then used to configure each step of the approach (represented by the diamond-shaped arrows in the figure).

In the second step of the approach (❷), all vehicle signals S are pre-processed based on their existing signal specification by the OEM. Followed by the creation of the label for the DDF (❸). This label is corresponding to the output $\hat{\boldsymbol{y}}$ the DDF should later predict (i. e. $\sum_{t=1}^{n} \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t = \boldsymbol{0}$). This label can either be generated based on signals already measured within the vehicle (e. g. when predicting the use of a function which is already part of the vehicle) or by an additional measurement. Additionally, new signals can be generated based on a set of rules.

In the next step (❹) *signal subset selection* a supervised filter feature selection is performed on all vehicle signals S based on the beforehand created label information $\hat{\boldsymbol{y}}$ to select the most appropriate signal subset $S_{\mathcal{A}}$ for the specified DDF. Until this step, each step is done independently for each signal and can easily be distributed over the E/E architecture or even deployed at the origin of each signal. Each signal is scored, and the score is collected and ranked in one central component. The scores and ranking of the top n selected signals are then forwarded to the *post-processing* (❺) as a proposed signal subset $S_{\mathcal{A}}$. This subset $S_{\mathcal{A}}$ can then either be automatically integrated as input for a DDF or can be reported to the developer for further investigation.

This overall process can either be deployed within the vehicle or in the back end. Additionally, we have to differentiate between customer vehicles or development vehicles. The exact deployment is decided by the *deployment strategy component* (❻) and the specification by the developer. The following sections further describe each element of the proposed approach.

3.2 Data-Driven Function Specification

In the first step of the approach (❶), the developer of the DDF has to specify multiple attributes of the to be developed DDF and configuration parameters. Listing 3.1 shows an exemplary specification for a DDF for the use case of a proactive seat heating. The purpose of this function is to predict the preferred state of the seat heating (i. e. on or off) based on the user's preferences and behaviour. The specification is done through a YAML-file [138], but could also be done through any other data interchange format (e. g. JavaScript Object Notation (JSON) [139]). The following paragraphs will introduce each attribute step by step.

3.2.1 Data-Driven Function

In the first part of the specification, the DDF has to be specified (`function`). This step is related to the model requirement step in a classical development workflow (cf. Section 2.1.3). In addition to the name of the function, the structure of the desired input $\boldsymbol{x}$ of the DDF has to be specified. This includes a threshold for the signal subset size, a whitelist of signals which by all means should be included, and a blacklist of signals which should be excluded.

First, the size of the selected signal subset used as input for a DDF has to be limited. This has to be done due to the reasons mentioned in the previous chapter (i. e. *the curse of dimensionality* and the impacts of the E/E architecture). The used algorithms score each signal and rank them according to their score. By this, the optimal number of feature is not computed. Therefore, the number of signals have to be limited by an absolute number of signals or a score threshold. This depends on the type of DDF and computational resources for the execution of this function. In some cases, it can be sufficient to use two signals as input, but in some cases, more can be necessary to achieve a good performance. This has to be specified by the

```
# ----------------------------------------------------------
# Specification for Proactive Seat Heating for the Driver
# ----------------------------------------------------------

meta_info:
  author:  Christoph Segler
  date:    2019-12-01
  version: 1.0
  comments: Signalnames changed due to confidential information

function:
  name:           seatheating_driver  # name of function
  max_inputs:     10                  # maximum number of selected signals

  whitelist:                          # whitelisted signals
  - vehicle_vin

  blacklist:                          # blacklisted signals
  - status_seatheating_driver
  - temperature_seat_driver
  - button_seatheating_driver

deployment_class:
  function_class: group               # system, group, or user

pre_processing:
  error-values:     # inclusion of error-values
    setting: no

  scaling:          # scaling of all signals in [0,1]
    setting: on
    min:     0
    max:     1

  discretization:  # binning of signal values into discrete bins
    setting: off

add_signals:
  - offset_temperature: temperature_outside - temperature_inside  # generation of new signal

labeling:
  - 0: status_seatheating_driver == 0       # rule for label 0: seatheating off
  - 1: 3 >= status_seatheating_driver >= 1  # rule for label 1: seatheating on

post_processing:
  report:  # settings for report
    setting: yes
    file: /raid/csegler/seatheating_driver.yaml

  auto:    # setting for automatic input selection
    setting: no
```

Listing 3.1: Exemplary Specification for the use case *Proactive Seat Heating*[*]

[*]The vehicle signal names in this document have been changed due to confidential information.

developer, and for further insight into the ranking, the manual post-processing can be chosen. In this example, a maximum number of $|S_{\mathcal{A}}| = 10$ signals is specified.

In the next attribute, the whitelist, vehicle signals can be defined, which would show low correlations to the label information, but are required by the DDF. For example, the signal containing the unique Vehicle Identification Number (VIN) would not correlate with the label information if run on the same vehicle. In this case, this signal would be ranked low due to its static value and thereby would not be included in the selected signal subset $S_{\mathcal{A}}$. However, in some cases, such a signal could be crucial as input for a DDF and must be included in the signal subset $S_{\mathcal{A}}$.

The opposite is specified in the blacklist. Here, signals can be determined, which should, by all means, not be included in the signal subset. This could either be the reason due to privacy regulation (e. g. signal containing the current location of the vehicle) or due to correlations of signals to the label information which would lead to undesired behaviour of a DDF. A simple example of such correlation are the signals on which the label information is generated. In the case of the proactive seat heating, the signal containing the level of the seat heating would show a very high correlation to the label information. This would lead to a high rank and would thereby be included in the selected signal subset. If this signal would now be used as input for the DDF, it would lead to an undesired behaviour as it is directly connected to the output of the DDF (i. e. the level of the seat heating). Also the signal containing the current temperature of the seat would show such a high correlation. This signal is not used for the label generation, but would also show a high correlation to the level of the seat heating as this function has a direct influence on the temperature of the seat. These signals are solely known by the expert of the desired function and consequently have to be specified by the developer. Additionally, due to computational optimisation or access rights management, other signals can also be excluded.

3.2.2 Deployment Class

In the next part of the specification, the deployment class has to be specified. This deployment class configures the later shown deployment strategy. In this approach, we differentiate three types of $S_{\mathcal{A}}$. These subset types range from intuitive correlations of a system to very user-specific correlations. In the following, we will briefly introduce each type of $S_{\mathcal{A}}$. Section 3.7.2 will introduce each type in detail.

The first signal subset type I ($S_{\mathcal{A}}^{I}$) includes all signal subsets that hold for *all* users or *all* systems and contain similar signals. For example, the steering torque of the steering wheel is only related to the physics of the driving dynamics and not to any user behaviour. These correlations are similar among all users and lead to the same selected signal subset.

The second type II ($S_{\mathcal{A}}^{II}$) includes all signal subsets that hold for a *group of users* and are at least similar for two users. For example in one of the use cases in the later shown evaluation, a group of users showed a correlation between the driving experience control (i. e. *sport*, *comfort*, or *eco pro*) and whether a co-driver is present or not. These kind of signal subsets are similar

for a particular group of users, but not all users.

The third type III ($S_{\mathcal{A}}^{III}$) includes all signal subsets that hold for only *one specific user* and are unique for every user. These correlations are very user-specific and at the core of user-specific personalisation. An example is parallel usage of several functions at the start-up of the vehicle (i. e. which functions are triggered when the user is commuting from home to work).

The developer has to specify which type of signal subset they suspect as results. This classification can only be done with prior knowledge from the expert developing the DDF and is used for the deployment strategy of the approach. If the type is unclear, it is also possible to iteratively classify it from type I ($S_{\mathcal{A}}^{I}$) to type III ($S_{\mathcal{A}}^{III}$).

3.2.3 Pre-Processing

In the next part of the specification, the pre-processing of the vehicle signals has to be configured. We distinguish between three different configuration parameters.

The first parameter (cf. `error-values`) configures if error-values of signals should be discarded or not. Current vehicle signals often contain values which are used for debugging or logging of unexpected events. Besides, if a value is not measured yet (e. g. at vehicle start-up), vehicle signals often contain a value referring to an empty value. In most cases, these values should be discarded for the development of a DDF, as most functions do not need this information. But in some cases (e. g. vehicle analytics functions) these values have to be considered for the development of the DDF.

The next parameter *scaling* configures the scaling of the values from each vehicle signal. Most DDFs are being developed with machine learning algorithms. Some of these algorithms are sensitive to different scales within the used input signals [140]. As the ranges of each signal are solely known in the specification of each signal, a rescaling must be directly performed in the pre-processing step. In the shown example (cf. Listing 3.1), a simple scaling of all signals in the range of $[0, 1]$ is specified.

Some machine learning algorithms need discrete values for all input values [141], and therefore, a discretisation can be configured in the pre-processing step. In the last parameter, the discretisation of signal values during the pre-processing can be configured. The data types of vehicle signals are highly diverse (cf. Section 2.2). These include continuous and discrete data types. In some cases, the algorithm used in the signal subset selection step needs a discretisation of signals and will be mentioned later. Many different methods for the discretisation have been developed and can be applied during the pre-processing step [142]. In the case shown in Listing 3.1, no discretisation is configured.

All three parameters have either to be configured according to the use case or to the used machine learning algorithm. Only the developer has this information and therefore has to set each parameter by themself.

3.2.4 Signal Generation

In this part of the specification, rules for the generation of new signals can be specified. In machine learning, this step is often referred to as *feature generation*. Here, in consideration of the term signal instead of feature, we will refer to as *signal generation*. Based on already present vehicle signals, the developer can specify rules for the generation of a new signal which could be possibly relevant for the DDF. The specification of such additional signals is optional and must be based on already present vehicle signals. This step leads to more signals which are evaluated by the signal subset selection but might also enrich the possible input for the DDF. For example, the developer—according to their expert knowledge—has the hypothesis that the duration of the current drive could be a useful input for the proactive seat heating. This information might not be part of the current set of vehicle signals. Still, it can be easily computed from a signal containing the current time and a signal containing the state of ignition. As part of the signal subset selection, the developer would receive feedback whether the hypothesis holds true and additionally is ranked among the other signals which might have a higher ranking. In the example shown in Listing 3.1, a new signal for the offset temperature of the outside and inside temperature is generated.

3.2.5 Labeling

In the next part of the specification, the label generation has to be specified (i. e. how the label for the supervised feature selection and supervised training of the DDF is generated). The label information $\hat{\boldsymbol{y}}$ corresponds to the desired output $\boldsymbol{y}$ of the later developed DDF. In the case of a perfectly designed and developed DDF the label information and output of the function should be identical (i. e. $\sum_{t=1}^{m} \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t = \mathbf{0}$) at all discrete time steps t with $1 \leq t \leq m, m \in \mathbb{N}^+$.

The label information $\hat{\boldsymbol{y}}$ can either be defined based on one or multiple vehicle signals, external signals, or a custom rule/function. In the case of the generation based on vehicle signals, the developer has to specify a rule manually. For example, if the developer wants to create a proactive seat heating (i. e. a seat heating which can predict the future state and thereby the preference of the user), the label can be directly created on the vehicle signals providing the current state of the seat heating. In this case, the DDF should only provide a prediction whether the seat heating should be turned on or off. The vehicle signal provides one value for off and various values for on. So a custom mapping has to be specified (cf. Listing 3.1). But in some cases, the generation of the label information can be more complex and requires an additional function or even manual labeling through an external signal. For example, if the DDF should predict the emotional state of the driver and this information is not available as a vehicle signal, it is necessary that an additional function is implemented generating this information or has to be provided from an external source.

This label information $\hat{\boldsymbol{y}}$ is then generated based on the specified rules, additional function, or manually in the labeling step of the proposed approach. This information is later used in

the signal subset selection and can also be directly used for the training of the DDF itself.

3.2.6 Post-Processing

In the last part of the specification, the post-processing of the signal subset $S_{\mathcal{A}}$ is configured. Here, $S_{\mathcal{A}}$ can be used in three different ways: (i) As report providing insights for the developer of a non-data-driven classical function, (ii) as report providing insight for the developer for a manual development of the DDF, or (iii) as automatically selected input for the DDF.

In the first case, a report of the ranking of each signal can give the developer of a classical function (i. e. non-data-driven) further insight how to optimise the function by adding signals. In the case of manual development, the signal subset is manually integrated into the DDF based on the report. Additional expert knowledge can be used during this manual step to optimise the selected input further; however, this does not scale for signal subsets varying for each user. In the automatic case, no manual assessment of the selected signal subset is required. This allows user-specific personalisation, but lacking the manual verification the input of the resulting function is not fully transparent during the development. Therefore, it may require additional verification steps within the DDF. Which one of the two different post-processing methods is preferred, has to be configured by the developer. Also, both at the same time are possible. In our example, (cf. Listing 3.1) a report has been configured. The resulting report will later be shown in Section 3.6.1.

3.3 Vehicle Signal Pre-Processing

This section introduces the next step of the approach, the *signal pre-processing* (❷). The basis for the approach are all vehicle messages which are sent within the vehicle's communications networks. The approach also allows it to directly use each signal within each ECU, where the signals are generated. However, this is not the case here as current E/E architectures often do not allow direct access to data within an ECU (cf. Challenge 2). As vehicle messages cannot be directly used for feature selection algorithms or machine learning algorithms, each message and vehicle signal has to pass multiple pre-processing steps. These include the decoding from messages to vehicle signals, filtering of signals, pre-processing of each signal, and more optional steps like scaling or discretisation.

Each message, PDU, and vehicle signal has a different structure. As shown in Section 2.2.5, current specifications are not sufficient for DDFs (cf. Challenge 5). In our approach, the existing specifications are automatically processed and enhanced with expert knowledge to automatically pre-process each vehicle signal for the usage with feature selection and machine learning algorithms. In the case of the BMW Group, all vehicle signals are specified in the company-wide tool *BordNetz Engineer (BNE)*. This tool supports various import and export formats (e. g. Field Bus Exchange Format (FIBEX) [143]) [144, p. 33]. Based on the information available in this tool, the vehicle message, PDUs, and vehicle signal are pre-processed in our approach.

Tables 3.1 to 3.3 show examples of three different vehicle signals and their specification, which are later introduced in detail. These examples contain information about the current vehicle speed (cf. Table 3.1), the current vehicle status (cf. Table 3.2), or the current road segment (cf. Table 3.3). At first, each specification contains a *longname* and a *shortname* to name the vehicle signal. In the second part of every specification, the purpose of each signal and content is briefly described. In most cases, these descriptions are very concise and require expert knowledge to be understood [75]. In the next part, the data type is specified. These can be either basic data types like boolean, uint8, uint16, float16 or more complex data types like strings, arrays, enumerations, bit fields, and union of different data types [76]. In the next part, it is specified how the data is encoded. This varies depending on the data type and is further described in the following paragraphs of this section. Finally, the specification contains information which PDU contains the signals and which message frames contain the corresponding PDU. Another part of the specification is the timing and/or events when each frame is sent. Based on this information, each frame is processed into pre-processed vehicle signals which can be directly used by the feature selection or the machine learning algorithm.

An overview of the pre-processing approach is depicted in Figure 3.4. In the initial step, the vehicle message frame arrives and is decoded into each PDU. In this example the frame contains the PDUs *VEH* (PDU 1) and *RD* (PDU 2). From each PDU, each vehicle signal is then extracted. In this case: *AliveCounter*, *Vehicle_Speed*, and *Vehicle_Status* for *VEH* (PDU 1) and *CRC* and *Road_Info* for *RD* (PDU 2). In current vehicles, some signals only contain operational information as countermeasures for error detections within the communication (e. g. Cyclic Redundancy Checks (CRCs), alive counters) as specified by the AUTOSAR standard [145]. These signals do not contain any valuable information as input for most DDFs and are thereby filtered out. In our example, both signals *AliveCounter* and *CRC* are discarded as they only contain operational information. Only in the case of detected errors, all effected signals are also discarded. Within this step, also the blacklist from the specification is applied (cf. Section 3.2.1). Next, the further pre-processing differs depending on the data type of each signal. In this approach, we automatically classify all vehicle signals into three different types: continuous data (e. g. vehicle speed), enumerated data (e. g. vehicle status), and bit field encoded data (e. g. street information). Signals containing continuous data typically contain environmental information which can be described in physical units (e. g. m/s, km/h). Signals with enumerated data typically contain distinct enumerated states of a system or environmental states. In this case, only one state can occur at the same time (e. g. vehicle status, road type). Signals encoded as bit field are either used when multiple signals are merged into one signal or where multiple states can occur at the same time (e. g. road information). In our approach, the classification of each signal is based on the data type specified in the available signal specification.

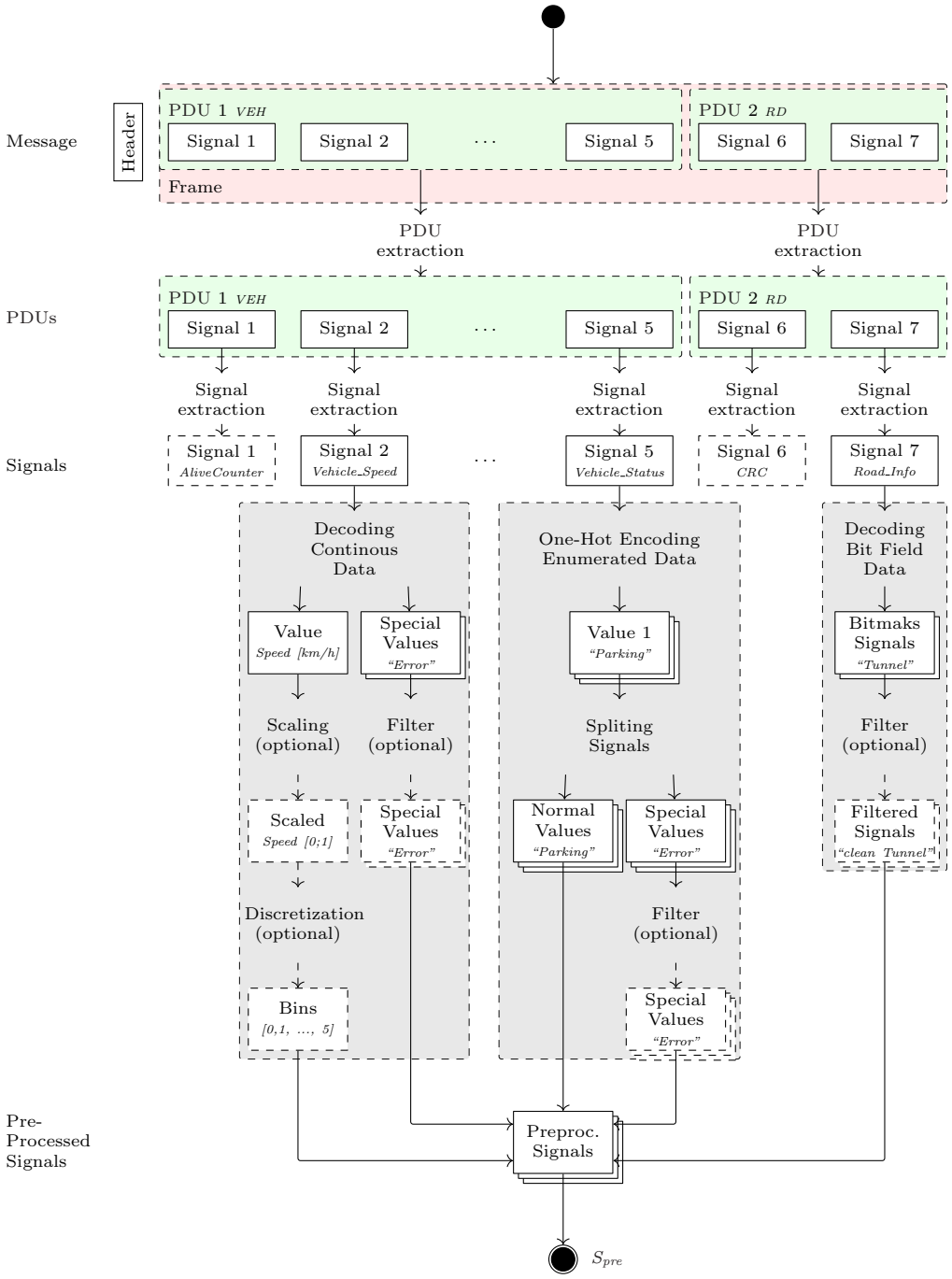


Figure 3.4: Vehicle signal pre-processing

3.3.1 Signals containing Continuous Data

Table 3.1: Specification of the signal *vehicle speed*[*]

[*]Names and values in this specification have been changed due to confidential information and only represents the general structure of such a vehicle signal.

Longname:	Vehicle_Speed
Shortname:	VEH_V
Description:	*Absolute speed of the vehicle's centre of gravity.* *This is a combination of the vehicles lateral and longitudinal speed*
Type:	16bit unsigned integer
Offset $off(s_i)$**:**	0
Resolution $res(s_i)$**:**	0.01
Range:	[0, 400]
Unit:	km/h
Sp. Values $sp(s_i, k)$**:**	Description:
0xFFFF	Error
PDU:	VEH
Frames:	0x123 A-CAN 12.3.4 FlexRay
Timing:	cyclic 50ms

First, we will introduce the pre-processing for signals containing continuous data. Let s_i be the signal i out of all vehicle signals S, with $1 \leq i \leq m = |S|, m \in \mathbb{N}^+$. All signals s_i of the data types integer or float (unsigned or signed) are classified as continuous signals, as these data types contain continuous data, even if encoded as an integer. In the here shown example, the signal vehicle speed (cf. Table 3.1) is classified as a continuous signal as its data type is unsigned integer. The signal specification is unique for signals with the data type integer or float. Let $\overline{s_i}$ be the received value of signals s_i. Also, let $res(s_i)$ be the specified resolution and $off(s_i)$ the specified offset for signal s_i and let $sp(s_i, k)$ be the special value k of signal s_i with $1 \leq k \leq n, n \in \mathbb{N}^+$, where n is the number of special values of s_i. For each signal, one or multiple special values $sp(s_i, k)$ can be defined which contain additional information and are not part of the specified range. In case of the vehicle speed, one special value $k = 1$ is specified. If the received signal contains this value, an error has occurred, and the signal of the vehicle speed contains an invalid value. Also, let s_{ij} be the new pre-processed signal based on the received signal s_i, with $1 \leq j \leq n+1, n \in \mathbb{N}^+$ and with n the number of possible special values of the signal s_i and let $\overline{s_{ij}}$ be the pre-processed value of s_{ij}.

The first pre-processed signal s_{i1} contains the pre-processed value in the specified unit and range (e. g. vehicle speed in km/h). The pre-processed value of s_{i1} can be obtained by converting the received value $\overline{s_i}$ according to the specification. This conversion only holds if the received values $\overline{s_i}$ does not contain any special values $sp(s_i, k)$. If this is the case, s_{i1} will

not be forwarded. The pre-processed signal s_{i1} is generated based on the following equation:

$$\overline{s_{i1}} = \begin{cases} (\overline{s_i} \cdot res(s_i)) + off(s_i) & \text{if } \overline{s_i} \notin \{sp(s_i,k) \mid 1 \leq k \leq n, n \in \mathbb{N}^+\} \\ - & \text{if } \overline{s_i} \in \{sp(s_i,k) \mid 1 \leq k \leq n, n \in \mathbb{N}^+\} \end{cases} \tag{3.1}$$

For example in the case of the vehicle speed a received value of $\overline{s_i} = 1000$ will result in a speed of $\overline{s_{i1}} = (1000 \cdot 0.010) + 0 = 10$ in the unit *km/h*.

All other pre-processed signals s_{ij} contain the information about any special values. The idea is to encode all special values in a new signal to split valid values from special values (e. g. invalid values). Each s_{ij}, where $2 \leq j \leq n+1$ represents one of the possible values $sp(s_i, k)$, which can either be 0 or 1.[1] These pre-processed signals are generated based on the following equation:

$$\overline{s_{ij}} = \begin{cases} 0 & \text{if } j-1 = k \text{ and } \overline{s_i} \neq sp(s_i,k) \\ 1 & \text{if } j-1 = k \text{ and } \overline{s_i} = sp(s_i,k) \end{cases} \quad | \quad \forall k \in \{1,\ldots,n\}, j \in \{2,\ldots,n+1\} \tag{3.2}$$

For example, if the value $\overline{s_i} = sp(s_i, k=1) = \mathit{0xFFFF}$ is received an error for the vehicle speed has occurred and only s_{i2} with $\overline{s_{i2}} = 1$ will be forwarded.

This splitting helps the feature selection algorithm or machine learning algorithm to distinguish between the actual values and special values. In the case of the converted value, optional pre-processing steps can be applied (e. g. scaling and discretisation) which are required by some feature selection or machine learning algorithms. These steps require the range of the signals, which is specified for every signal containing continuous data and can be directly used from the available specification.

3.3.2 Signals containing Enumerated Data

All signals s_i of the data type *enumeration* are classified as enumerated data. Table 3.2 shows the specification of an exemplary signal containing enumerated data. Each value of the signal corresponds to a certain state, special states (e. g. "error"), or non-defined states. Signals with enumerated data typically contain distinct enumerated states of a system or environmental states, where only one state can occur at the same time. The here shown signals, states the current status of the vehicle, where only one state can occur at the same time.

Let again $\overline{s_i}$ be the received value of signals s_i out of S. Let $val(s_i, k)$ be the specified value k of the signal s_i, with $1 \leq k \leq n, n \in \mathbb{N}^+$, where n is the number of possible values of s_i. Let s_{ij} be the pre-processed signals where each signal represents one of the possible values $val(s_i, k)$. The received value $\overline{s_i}$ of signal s_i will be split into n signals where each represents one of the possible values $val(s_i, k)$, which can either be 0 or 1:

[1] In machine learning, this conversion is also known as *one-hot encoding* (cf. [146, p. 129])

Table 3.2: Specification of the signal *vehicle status*[*]

[*]Names and values in this specification have been changed due to confidential information and only represents the general structure of such a vehicle signal.

Longname:	Vehicle_Status
Shortname:	VEH_ST
Description:	*This signal describes the vehicle status from the view of the user/driver*
Type:	4 bit enumeration
Value $val(s_i, k)$**:**	Description:
0x0	reserved
0x1	Parking
0x2	Parking Error
0x3	On / Driver not present
0x4	-
0x5	On / Driver present
0x6	-
0x7	Debugging
0x8	Ignition
0x9	-
0xA	Driving
0xB	-
0xC	Switching off
0xD	reserved
0xE	reserved
0xF	empty signal
PDU:	VEH
Frames:	0x123 A-CAN 12.3.4 FlexRay
Timing:	cyclic 50ms

$$\overline{s_{ij}} = \begin{cases} 0 & \text{if } \overline{s_i} \neq val(s_i, k) \\ 1 & \text{if } \overline{s_i} = val(s_i, k) \end{cases} \quad | \quad \forall j, k \in \{1, \ldots, n\}, j = k \tag{3.3}$$

After the splitting each s_i which is containing enumerated data into multiple signals s_{ij}, each signal is either classified as *normal values* which contain information of normal vehicle states (e. g. "Parking") or into *special values* which contain debug or error information (e. g. "Parking Error", or "reserved"). As the specification of a vehicle signal does not contain this information, the classification is performed based on the description of each value. Through a keyword search within the description, the states can be classified as *special values* (e. g. "error", "empty", or "reserved") or as *normal values*.[1]

This splitting helps the feature selection algorithm or machine learning algorithm to distinguish between the actual values and special values. Additionally, some algorithms require this so-called *one-hot encoding* to perform well, as the distance (i. e. order) of the values do not contain any information and should not be used by the algorithm. This holds especially for data containing categories (mostly enumerated data).

[1]The complete list of keywords used for this classification cannot be disclosed due to confidential information. This keyword list is highly depending on the OEM and has to be designed based on expert knowledge.

3.3.3 Signals containing Bit Field Encoded Data

Table 3.3: Specification of the signal *road information*[*]

[*]Names and values in this specification have been changed due to confidential information and only represents the general structure of such a vehicle signal.

Longname:	Road_Info	
Shortname:	RD_INFO	
Description:	*This signal contains information on the current road segment*	
Type:	8 bit bit field	
Bitmask ϕ_{ik}:	Value:	Description:
0000 0011b	**** **00b	no bridge
0000 0011b	**** **01b	bridge
0000 0011b	**** **10b	bridge information not available
0000 0011b	**** **11b	-
0000 1100b	**** 00**b	no tunnel
0000 1100b	**** 01**b	tunnel
0000 1100b	**** 10**b	tunnel information not available
0000 1100b	**** 11**b	-
0011 0000b	**00 ****b	no trees
0011 0000b	**01 ****b	trees
0011 0000b	**10 ****b	trees information not available
0011 0000b	**11 ****b	-
0100 0000b	*0** ****b	right-hand drive
0100 0000b	*1** ****b	left-hand drive
0011 1111b	0010 1010b	information not available
1111 1111b	1111 1111b	invalid signal
PDU:	RD	
Frames:	0x123 A-CAN	
	0x345 B-CAN	
Timing:	cyclic 50ms	

All signals of the data type bit field are classified as bit field encoded data. Table 3.3 shows the specification of an exemplary signal containing bit field encoded information of the current road segment. The specification shows the encoded values, the used bitmasks and a short description of each value. In this example, a received value $\overline{s_i} = 33 = 0100\,0001b$ holds the information "bridge", "no tunnel", "no trees", and "left hand drive" for the current road segment.

In this approach, each received signal s_i containing bit field encoded data is split into multiple signals s_{ij} based on the bit fields. Let ϕ_{ik} be the transformation from the received signal value $\overline{s_i}$ with bit field k, the binary length m of the received signal s_i, with $1 \leq k \leq n, n \in \mathbb{N}^+$, where n is the number of possible bitmasks of s_i:

$$\phi_{ik} : \mathbb{F}_2^m \to \mathbb{F}_2^m, (b_1, \ldots, b_m) \to (\tilde{b_1}, \ldots, \tilde{b_m}), \mathbb{F}_2 = \{0, 1\} \tag{3.4}$$

If only one value is specified for the bitmask k (e. g. the "invalid signal" value in this example), the value of the received signal $\overline{s_i}$ has to be equal to the defined value or the pre-

processed value will be $\overline{s_{ij}} = 0$.

Let s_{ij} be the pre-processed signals where each signal represents one of the possible bit fields k. The received value $\overline{s_i}$ of signal s_i will be split into multiple signals s_{ij}, applying all defined bitmasks k:

$$\overline{s_{ij}} = \phi_{ik}(s_i) \quad | \quad \forall j, k \in \{1, \ldots, n\}, j = k \tag{3.5}$$

In the example of the signal for the road information a received value $\overline{s_i} = 33 = 0100\,0001b$ will result in one signal $\overline{s_{i1}} = 0000\,0001b = 1$ for the "bridge" value, one signal $\overline{s_{i2}} = 0000\,0000b = 0$ for the "tunnel" value, one signal $\overline{s_{i3}} = 0000\,0000b = 0$ for the "tree" value, one signal $\overline{s_{i4}} = 0100\,0000b = 64$ for the "left-hand drive" value, one signal $\overline{s_{i5}} = 0000\,0000b = 0$ for the "information not available" value, and one signal $\overline{s_{i6}} = 0000\,0000b = 0$ for the "invalid signal" value.

This splitting helps the feature selection algorithm or machine learning algorithm to distinguish between each bitmask, which each encodes different information even though it is encoded within the same signal. Each signal also contains information about special states as in this case "tunnel information not available". This is done with the same keyword search list as in the case of the enumerated data.

3.3.4 Pre-Processed Signals

This pre-processing approach results in a reduction of signals (e. g. filtering of CRCs) and an increase of signals by splitting the vehicle signals into additional signals. Each signal s_{ij} is individually forwarded and processed in the next steps of this approach and is further referred to as $s_{ij} \in S_{pre}$. The signal name of each signal s_{ij} holds information about the original signal and a suffix of the pre-processing step. It can thereby easily be identified and traced back to the original specification. In the later shown data set a total of around 4 000 signals s_i are converted (without CRCs, and Alive Counters), which results in around 19 900 pre-processed signals s_{ij} including special values, or around 14 000 signals s_{ij} without special values. At this point in the approach, all vehicle messages are still processed asynchronously as data streams. Thereby the challenge of asynchronous communication (cf. Challenge 3) and the lack of resources to store the data (cf. Challenge 6), does not affect the proposed approach. More pre-processing and data enhancement steps would be possible, but based on the currently available vehicle specifications (cf. Challenge 5), only these steps can be performed in an automated way.

3.4 Label and Signal Generation

In the next step of the approach, the *label and signal generation* (❸), all additionally defined signals and the label are generated. An overview of this step is given in Figure 3.5.

In the initial state of this step, the pre-processed signals $s_{ij} \in S_{pre}$ of the previous step are received. Optionally external signals s_e can be added here. For the definition of the external

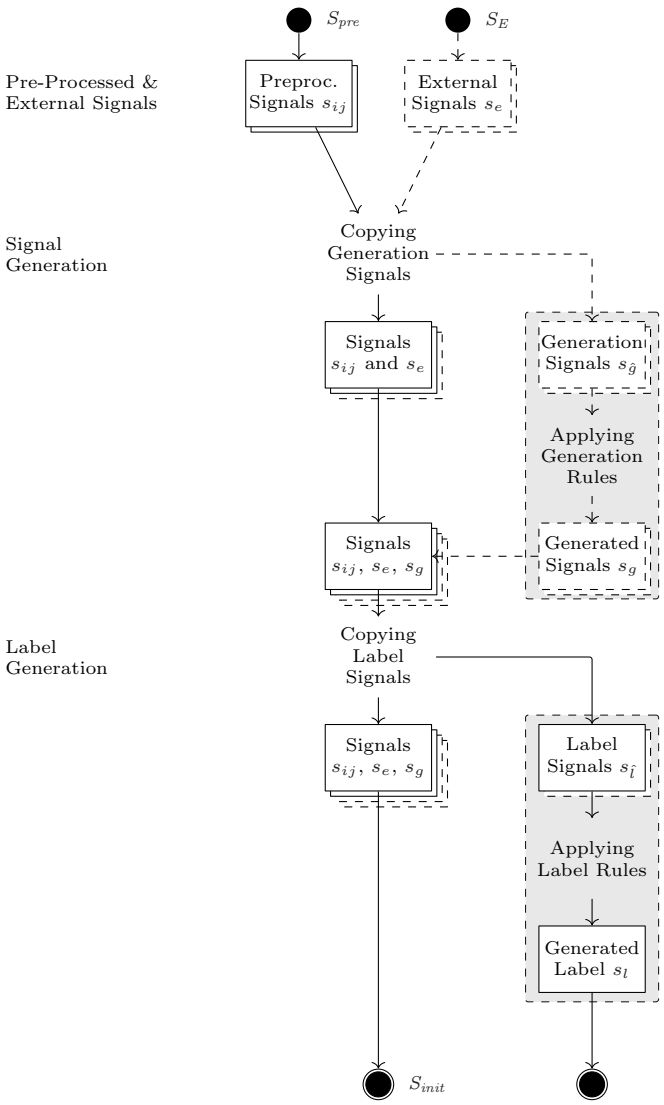


Figure 3.5: Overview of the label and signal generation

signals, let s_e be the external signal e with $1 \leq e \leq o, o \in \mathbb{N}^+$, where o is the number of external signals. Let S_E be the set of all external signals $S_E = \{s_1, \ldots, s_e, \ldots, s_o\}$. For example, the developer could define external weather data as a new signal. For the next steps, S_{pre} and S_E are merged (i. e. $S_{pre} \cup S_E$).

In the first step (the signal generation), all additional signals are generated, which have been defined by the developer in the DDF specification (cf. Section 3.2). These signals hold additional information which could be beneficial as input for the DDF. These signals are not yet part of the vehicle signals S_{pre} or the external signals S_E and have to be specified by the developer. The basis for the generation of these signals can either be already existing vehicle signals out of S_{pre} or external signals out of S_E. In the example of the proactive seat heating (cf. Listing 3.1) a new signal is defined which holds the offset temperature between the outside and inside temperature of the vehicle, based on already existing vehicle signals. Let $s_g \in S_G$ be the generated signal g, with $1 \leq g \leq p, p \in \mathbb{N}^+$, where p represents the number of generated signals. Let $s_{\hat{g}} \in (S_{pre} \cup S_E)$ be a signal specified in the rule for the generation of a signal s_g, with $1 \leq \hat{g} \leq q, q \in \mathbb{N}^+$, where q represents the number of required signals. Furthermore, let all signals $s_{\hat{g}}$ required for the generation of s_g be $\hat{G} = \{s_1, \ldots, s_{\hat{g}}, \ldots, s_q\}$.

In the first step of the signal generation, all signals out of $S_{\hat{G}}$ are extracted from the data stream. In the example of the proactive seat heating (cf. Listing 3.1) this would only include the signals `temperature_outside` and `temperature_inside`. Based on the extracted signal, the specified rules are applied to generate the new signals S_G. In this example, only one rule is defined. Here, the rule is a simple subtraction of both signals and results in the signal `offset_temperature`. In the end, the generated signals S_G are merged with the other signals S_{pre} and S_E (i. e. $S_{pre} \cup S_E \cup S_G$). This step is still completely asynchronous. If all signals required for the generation of a new signal are received within the same vehicle message, no synchronisation is needed at all, as all signals arrive at the same point in time. If this is not the case, the last value of each signal $s_{\hat{g}}$ used for the generation of a new signal s_g has to be buffered. The same can be applied for the distributed execution of this approach. Until this step, the approach is completely distributed over the vehicle's E/E architecture. In this case, only the required signals $s_{\hat{g}}$ for the generation of a new signal s_g have to be available at the same point. As the number of signals $|S_{\hat{G}}|$ is minor, this can easily be achieved.

Based on the pre-processed signals S_{pre}, external signals S_E, and newly generated signals S_G the label signal is generated. The label signal is crucial for the supervised signal subset selection in the following step and the whole approach. This generated label signal holds the information $\hat{\boldsymbol{y}}$ of the desired output $\boldsymbol{y}$ of the DDF. In the case of a perfectly designed and developed DDF the label information and output of the function should be identical at all discrete time steps t with $1 \leq t \leq n, n \in \mathbb{N}^+$ (i. e. $\sum_{t=1}^{n} \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t = \boldsymbol{0}$). This label is generated based on the DDF specification (cf. Section 3.2). In the example of the proactive seat heating, this label corresponds with the status of the driver's seat heating which should later be predicted. Let s_l be the generated label and let $s_{\hat{l}} \in (S_{pre} \cup S_E \cup S_G)$ be the signals specified in the rule for the generation of the label

s_l. Also, let $\overline{s_l}$ be the value of s_l. Furthermore let $S_{\hat{L}}$ be the set of signals $s_{\hat{l}} \in (S_{pre} \cup S_E \cup S_G)$ required for the generation of s_l, with $1 \leq \hat{l} \leq r, r \in \mathbb{N}^+$, where r represents the number of required signals. In the first step of the label generation, all signals out of $S_{\hat{L}}$ are extracted which are required to create the label specified by the developer. Based on these signals $s_{\hat{l}}$, the specified labeling rule is applied to generate the label s_l. In the example of the proactive seat heating, this label s_l holds the information if the seat heating of the driver is switched on or off. In this case, only one signal $s_{\hat{l}}$ (i. e. $|S_{\hat{L}}| = 1$) is required for the generation of the label s_l. The required signal `status_seatheating_driver` contains the information of the current seat heating level. As the label should only hold the information if the seat heating is switched on or off and not the actual level of the seat heating, a simple transformation is necessary. All values 0 are mapped to 0, and all values between 1 and 3 are mapped to 1. Similar to the generation of the signals, this can be executed asynchronously and distributed, depending on the signals in $S_{\hat{L}}$.

Finally, the pre-processed signals S_{pre}, external signals S_E, and newly generated signals S_G are forwarded, here further denoted as $S_{init} \stackrel{\text{def}}{=} S_{pre} \cup S_E \cup S_G$. The label information s_l is forwarded separately to the next step.

3.5 Signal Subset Selection

In the signal subset selection (❹), the actual signal subset $S_{\mathcal{A}}$ for the input of the DDF is selected. An overview of this step is depicted in Figure 3.6. In the initial state of this step the label information s_l and the signals of the previous step are received S_{init} (i. e. $S_{pre} \cup S_E \cup S_G$).

Let s_a be all signals with $1 \leq a \leq m, m \in \mathbb{N}^+$, where m is the total number of pre-processed, external and generated signals (i. e. $m = |S_{init}|$). Furthermore, let $\overline{s_a}$ be the value of signals s_a. For each signal s_a a score $score_a$ is calculated. The calculation of the score is based on a supervised filter feature selection algorithm (cf. Section 2.3.2) and uses the current value $\overline{s_a}$ of the signal s_a and the current value $\overline{s_l}$ of the label s_l to calculate a score describing the importance of the signal. Depending on the used algorithm, this scoring can be completely independent for each signal s_a or not. If the algorithms work independently for each signals s_a, this step only needs the current value $\overline{s_l}$ of the label s_l and each signal-value (i. e. $\overline{s_a}$ of s_a) independently. In this case, this step can easily be distributed over the E/E architecture (cf. Challenge 1) and no synchronisation of all signals is required (cf. Challenge 3). When executed onboard the vehicle, the calculation of the score can even be performed where each signal is generated. The only communication within the architecture would be the broadcast of the label s_l consisting out of one value. If an algorithm is used, which requires multiple signals s_a for the computation, these have to be present at the same place, and further synchronisation is required.

Next, the scores $score_a$ are collected and ranked among each other. Depending on the algorithm, a higher score describes more important signals or vice versa. As a result, a ranking

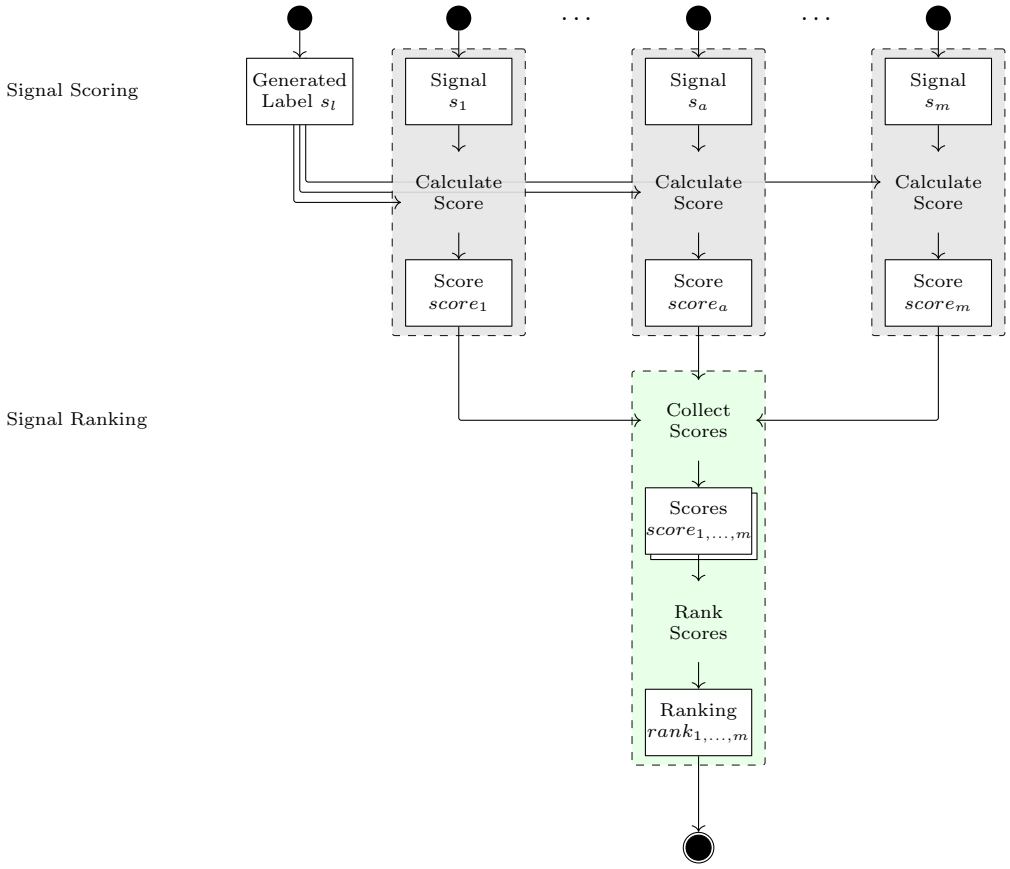


Figure 3.6: Overview of the signal subset selection

$rank_a$ of each signal s_a is obtained, and this is the basis for the next step the *post-processing*. This is the first step in the proposed approach, which has to be executed in a central place. This step does not produce a substantial overhead in communication or is contradictory to the federated E/E architecture (cf. Challenge 1), as only the score of each signal, is collected and no actual signal values have to be sent. Depending on the DDF, the collection has to be performed once within multiple seconds, minutes, hours, or even days.

Based on the ranking $rank_a$ the top n signals are selected. The value of n has to be specified as a threshold in the specification of the DDF. These top n signals out of S_{init} are denoted as the set S_A with the size $|S_A| = n$. This selected signal subset S_A is then forwarded to the next step, the *post-processing*.

3.6 Post-Processing

After the selection of the signal subset S_A it can be post-processed (❺) in three different ways: (i) As report providing insight for the developer of a non-data-driven classical function, (ii) as report providing insight for the developer for a manual development of the DDF, or (iii) as automatically selected input for the DDF.

In the first case, the report of the selected signal subset and the ranking of each signal can give the developer of a classical function (i. e. non-data-driven) further insight how to optimise the function by adding new signals. In the case of manual development, the signal subsets are manually integrated into the DDF. Additional expert knowledge can be used during this manual process to optimise the selected input further; however, this does not scale for signal subsets varying for each user. In the automatic case, no manual assessment of the selected signal subset is required, and the signal subset is automatically used as input for DDF. This allows very user-specific personalisation. However, without any manual verification, the input of the resulting function is not fully transparent during the development phase, as it could differ for every user. It, therefore, may require additional verification steps within the DDF. For every DDF, the post-processing can be configured in the DDF specification (cf. Section 3.2). Which one of the two different post-processing methods is preferred, has to be decided by the developer. Also, both at the same time would be possible. In the following two paragraphs, each post-processing method is further presented.

3.6.1 Report

To gain further insights or to manually develop the DDF, a report of the results can be configured. In the case of the report, all ranking results are collected from the vehicle(s) data and merged into one report for the developer. An exemplary report for the use case *proactive seat heating* is shown in Listing 3.2. This report is generated either once after a specific execution time of the approach or continuously. The report contains meta-information from the specification, the initial specification, and the results of the signal subset selection. Similar to the

specification, the results are exported to a YAML-file, but could also be exported to similar data interchange formats (e. g. JSON).

The first part of the report contains a direct copy of the metadata and the configuration from the specification. In Listing 3.2, this information is not further mentioned and would be similar to Listing 3.1. The second part contains the actual information of the results and the selected subset of vehicle signals. The results are listed for each unique user the approach has been executed on. Each user is pseudonymised with a hashed id in order to preserve the user's privacy. In case the approach is executed on test vehicles owned by the OEM, the hashing could also be omitted for the identification of the exact vehicle.

First, the timestamp information of the generated results is stated. The timestamp is encoded in a human-readable format as well as *seconds since the epoch* (cf. [147]). In the next part, the information on the evaluated data is stated. As the communication within the E/E architecture is asynchronous, no exact number of samples can be counted as one normally would do in the field of machine learning. In this case, the exact time of the evaluated data (i. e. the time the car was driven) is stated. The next section contains the information on how long each label has been observed, which is crucial information for the developer to evaluate the results. For example, this information can be used to validate how long a function is used, and if the result should be omitted. In some cases, the rules for the label generation lead to missing values of the label (e. g. if a function's state is not valid, or during the start-up of the vehicle). The duration of missing label information is encoded as Not a Number (NaN) (cf. [148, p. 26]) and during this time, no signals s_a are processed. The ranking section contains the actually selected signal subset $S_{\mathcal{A}}$ in the order of the $rank_a$, stating the score $score_a$ of each selected signal s_a. In our example, only the top $n = 10$ ranked signals are listed, as configured in the specification. The score for each signal helps the developer to select the appropriate signals and to assess the relevance of each signal. The method and range of the scores depends on the used algorithm.

The section with all whitelisted signals follows the ranking section. For each whitelisted signal the score is additionally stated. In the here shown example the signal containing the VIN has a score of zero. As the value of this signal never changes within the same vehicle, and the user only used the same vehicle, the algorithm calculates for this static signal a score of zero. Whitelisted signals are never excluded from the evaluation or report, as the developer can configure them explicitly. In addition, the scores give the developer the possibility to evaluate their hypothesis on the signals. For example, if the customer uses different vehicles (i. e. different VINs) and if this correlates to the usage of the function. In the last part of the report, the execution time of the algorithm is stated, and the used algorithm.

```
# ------------------------------------------------------
# Report for Proactive Seat Heating for the Driver
# ------------------------------------------------------


meta_info:
  # meta information from specification
  author:   Christoph Segler
  date:     2019-12-01
  version:  1.0
  comments: Signalnames changed due to confidential information

specification:
  # configuration from specification (cf. Data-Driven Function Specifiaction)
  function:
    name:           seatheating_driver
    max_inputs:     10
    ...
  pre_processing:
    ...
  add_signals:
    ...
  labeling:
    - 0: status_seatheating_driver == 0       # rule for label 0: seatheating off
    - 1: 3 >= status_seatheating_driver >= 1  # rule for label 1: seatheating on

results:
  # results from the proposed approach on actual data (cf. Evaluation)
  - user_id:   99c5e07b4d5de9d18c350cdf64c5aa3d # hashed user id

    timestamp: 1575203018  # timestamp of the report
    date:      2019-12-01  # date of the report
    time:      13:23:38    # time of the report

    signals:   14045       # number of evaluated signals
    duration:  50290.0s    # duration of data/drive

    labels:
      # observed duration of each label
      - 0:      20400.0s    # seatheating off
      - 1:      29440.0s    # seatheating on
      - NaN:      450.0s    # non specified label

    ranking:
      # selected signals by algorithm including score
      - temperature_evaporator:     0.7713567316753018
      - hvac_heatflow:              0.7005797593649095
      - hvac_ventilation_control:   0.4827692038819273
      - temperature_axle_front:     0.4465333673434153
      - temperature_ecu1_sensor2:   0.4448788330681288
      - temperature_ecu1_sensor1:   0.4446271999459923
      - temperature_axle_front_amp: 0.4367590224275681
      - temperature_axle_rear:      0.3381573220801821
      - temperature_outside:        0.3240877182274780
      - presence_passenger:         0.1105681040988211

    whitelist:
      # selected signals by specification including score
      - vehicle_vin:                0.0

    execution_time: 9.17s  # actual execution time of algorithm
    algorithm: FisherScore # used algorithm

  - user_id:   dd458505749b2941217ddd59394240e8
    ...
    ...
```

Listing 3.2: Exemplary report for the use case *Proactive Seat Heating* based on evaluation data*

*The vehicle signal names in this document have been changed due to confidential information.

3.6.2 Automatic Input

The previously selected signal subset $S_{\mathcal{A}}$ can also be directly used as input for the DDF without any further manual processing. Figure 3.7 shows an overview of this step.[1] The input of the DDF is split into two different types of input, the *dynamic input* and the *static input*.

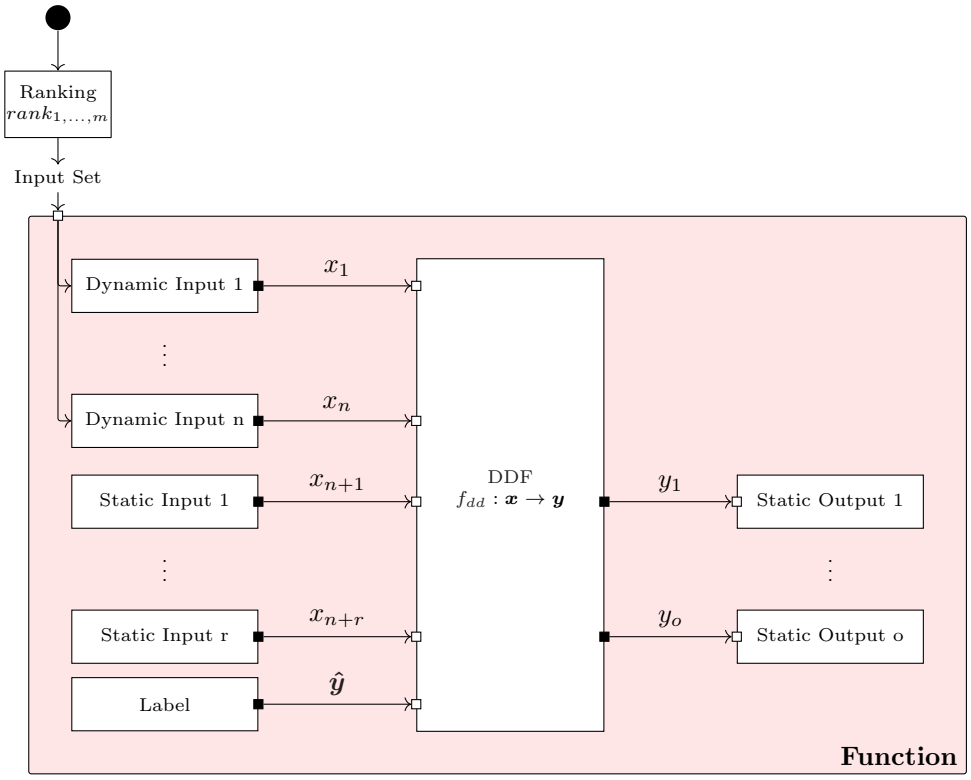


Figure 3.7: Automatic input selection for data-driven functions

The dynamic input is defined after the selection of the signal subset $S_{\mathcal{A}}$. This leads to a dynamic definition of this input at run-time. Depending on the type of DDF, this dynamic input can vary between each user or each vehicle. In contrast to the dynamic input, the static input is predefined during the deployment.

Let x_i be the input signal of the input vector $\boldsymbol{x}$ of the DDF f_{dd} with $1 \leq i \leq n+r, n \in \mathbb{N}^+, r \in \mathbb{N}^+$, with n dynamic inputs and r static inputs. The dynamic input is automatically defined through the selected signal subset $S_{\mathcal{A}}$. Each input signal x_i is linked to the corresponding signals s_a, where $rank_a = i$ (i. e. the first input corresponds to the highest-ranked signal and so on). The size of the dynamic input corresponds to the size of the selected signal subset (i. e. $n = |S_{\mathcal{A}}|$).

The number of signals n is always fixed, which makes it easier for the developer to create the

[1] The general concept for the automatic input selection has been previously published at the 2020 IEEE ICSA (cf. [10]).

actual function and do not have to care about changing input sizes. The disadvantage of the fixed input size is that redundant or irrelevant data might be used as input of the DDF. Here, we can neglect this as n is typically a lower two-digit number, which is a significant improvement compared to a total number of up to many thousand signals (cf. Challenge 4) available within the vehicle. In the case of the example of the proactive seat heating, the number of selected input signals is $n = 10$, and the selected input signals would be the signals shown in Listing 3.2. The static input has to be manually specified by the developer and can also be omitted.

For the training of the DDF, the label vector $\hat{\boldsymbol{y}}$ is required, which can either be identical to the label information s_l from the previous steps or can be defined independently. In most cases, the label vector is equal to the generated label in the label generation step and can be directly used. In some cases, the developer requires different or additional label information for the training. For example, in the case of the proactive seat heating, the developer might want to use a more precise input, which has all seat heating levels encoded.

The DDF (f_{dd}) itself has to be created by the developer. This function will learn the representation of the input $\boldsymbol{x}$ to the $\boldsymbol{y}$ and trying to fit the output vector $\boldsymbol{y}$ to the label vector $\hat{\boldsymbol{y}}$. Let y_i be one output signal of the output vector $\boldsymbol{y}$ of f_{dd} with $1 \leq i \leq o, o \in \mathbb{N}^+$, with o the number of outputs generated by f_{dd}. These outputs are only defined by the developer. In the example of the proactive seat heating, this could either be the control of the seat heating level, the temperature the seat heating should reach, or even the current/voltage used to heat the seat. Depending on the implementation in most cases, the output directly relates to the label $\hat{\boldsymbol{y}}$, but it could also differ. Parts of this DDF can also be pre-trained on the static input labels and then later be deployed including the dynamic input.

The automatic input has the vital advantage of directly selecting the appropriate input for a DDF onboard of the vehicle and therefore scales very well for highly personalised functions. However, this also comes with the trade-off of in-transparency of the input signals during the development of the DDF.

3.7 Deployment Strategy

The previously presented approach can be deployed on different targets (e. g. onboard, off-board). The deployment strategy component (❻) controls this deployment. The approach can be deployed in the OEM's back end infrastructure on recorded or streamed data or can be deployed directly onboard the vehicle. The following paragraphs introduce the different deployment targets and strategy.[1]

[1] The general idea behind the deployment strategy has been previously published at the 2019 IEEE ITSC (cf. [4]).

3.7.1 Deployment Targets

We differentiate between four different deployment targets. An overview of the different deployment targets and their attributes is given in the upper part of Table 3.4.

Table 3.4: Overview on deployment targets and strategy*
*Parts of table previously published at 2019 IEEE ITSC (cf. [4]) ©IEEE 2019

Deployment Data	Back End Test Fleet	Back End Set of users	Onboard Set of users	Onboard All users
Data Collection Cost	● ● ●	○ ○ ●	○ ● ●	○ ● ●
E/E Architecture Impact	● ● ●	○ ● ●	○ ○ ●	○ ○ ●
Algorithm complexity	● ● ●	● ● ●	○ ○ ●	○ ○ ●
Privacy Preservation	● ● ●	○ ○ ●	○ ● ●	○ ● ●
Result Delay	● ● ●	○ ○ ○	○ ○ ○	○ ○ ○
Report Generation	✔	✔	✔	✖
Automatic Input Selection	✖	✖	✖	✔
Type S_A^I (System)	✔	✔	✔	✔
Type S_A^{II} (Group)	✖	✔	✔	✔
Type S_A^{III} (User)	✖	✖	✖	✔
Legend:	Good: ● ● ●, notable: ○ ● ●, moderate: ○ ○ ●, poor: ○ ○ ○ Covered by deployment strategy: ✔, not covered: ✖			

The most straightforward deployment is execution on recorded test fleet traces in the back end. These traces are already present in the OEM's back end and are collected from test fleet vehicles which are equipped with data loggers. This data is usually used for the development and testing of prototypes, but can also be used for this approach. By using this data, no additional cost for the collection is added, and the impact of the E/E architecture is relatively small as the logging hardware is already integrated into the test vehicles. Any algorithm can be deployed in this case, as computing resources in the back end can be considered as nearly unlimited. Running the approach on already recorded data leads to almost immediate results and a fast development of the DDF. However, this data originates from test vehicles which are driven by test drivers. Test drivers behave differently from a normal user, as they verify the proper function of the vehicle and its functionalities. Therefore no user-specific behaviour can be covered, but is highly privacy-preserving, as no customer-related data is used.

To cover any user-specific behaviour, the approach has to be run on actual user data. In case of a back end-deployment, user data needs to be transferred to the back end. The transmission of all vehicle signals of a complete vehicle fleet would be highly inefficient and very costly, even if technically possible (cf. Challenge 6). The only way to reduce this amount is to either lower the number of vehicles or number of vehicle signals. As this approach tries to discover appropriate vehicle signals for a specific DDF, only a reduction of the number of vehicles is reasonable. By this, the transferred amount of data can be minimised by only collecting data of a subset of users, but this also leads to less accurate results by not capturing the behaviour of all users. This transfer of vehicle signals also adds hardware cost for the vehicle, which are

selected for the data transfer. In the case of production vehicles, the hardware has to either be retrofitted to the chosen vehicles or built-in for all vehicles at the production phase, as it is not clear from the beginning which vehicles will be selected. On top, the characteristics of the E/E architecture also impacts this deployment. As the data has to be collected from real users, the collection can earliest start after the Start of Production (SOP) and therefore delays the receipt of the first results. As actual user-related data of a group of users is collected, this is not as privacy-preserving as using data from test vehicles.[1]

By deploying the algorithms directly onboard the vehicle, no actual user data needs to be transmitted to the back end, and therefore the data collection is much more efficient. However, when deploying onboard, additional computational resources have to be added to the vehicle and thereby restrict the applicable algorithms. With this deployment, we also need to consider all characteristics of the E/E architecture. The design of this approach allows an easy, distributed deployment within the vehicle. A significant advantage of this deployment is its privacy-preserving nature by not transmitting any user-related data and only transmitting the resulting signal subsets. When trying to identify highly personalised signal subsets, deployment on all vehicles is inevitable, as the subsets vary for every user.

If we consider the different post-processing methods and the different deployment targets, not all methods can be deployed everywhere. The middle part of Table 3.4 gives an overview of the possible options. In the case of report generation, only the first three deployment targets can be used (i. e. back end on test fleet data, back end for a set of users, and onboard for a set of users). We do not consider the report generation for an onboard deployment for all users. This would lead to a vast amount of reports, and this would not necessarily help the developer. Even if so, the purpose of the report is to develop a DDF valid for all users. This can already be done with the report of a smaller set of users. In the case of an automatic selection for the input, only an onboard deployment on every vehicle would scale. Here an implementation in the back end would also add a significant communication overhead by transmitting data back and forth. An onboard-deployment for a set of users does not make sense as the DDF requires the selected input within the vehicle.

3.7.2 Deployment Classification and Strategy

For the deployment strategy of this approach, we distinguish between three different types of signal subsets $S_{\mathcal{A}}$. These subset types have been briefly presented in Section 3.2. The developer has to specify which type of signal subset they expect as result. This classification can only be done with prior knowledge from the expert developing the DDF. In case it is unclear which type to use, an iterative classification is possible. These subset types range from intuitive correlations of a system to very user-specific correlations. We categorised the signal subsets $S_{\mathcal{A}}$ into the following three types:

[1] In case of deployment on user data, the approach has to be deployed in compliance with the user's preferences and local laws and regulations (e. g. General Data Protection Regulation (GDPR) [149], California Consumer Privacy Act (CCPA) [150]).

Let S_i with $1 \leq i \leq k, k \in \mathbb{N}^+$ be a selected signal subset $S_\mathcal{A}$ for user i and a total number of users k. The first signal subset type, *type I* ($S_\mathcal{A}^I$), includes all signal subsets S_i which hold true for *all* users or *all* systems and contain a similar signal subset:

$$S_\mathcal{A}^I \stackrel{\text{def}}{=} \{S_i \mid \forall i, j \in \{1, \ldots, k\}, S_i = S_j\} \tag{3.6}$$

For example, the steering torque of the steering wheel is only related to the physics of the driving dynamics and not to any user behaviour. These correlations are similar among all users and lead to the same selected signal subset.

The second type, *type II* ($S_\mathcal{A}^{II}$), includes all signal subsets S_i which hold true for a *group of users* and are at least similar for two users:

$$S_\mathcal{A}^{II} \stackrel{\text{def}}{=} \{S_i \mid \exists i, j \in \{1, \ldots, k\}, i \neq j \wedge S_i = S_j\} \tag{3.7}$$

For example in one of the test cases in the later shown evaluation, a group of users showed a correlation between the driving dynamics control (i. e. *sport*, *comfort*, or *eco*) and whether a co-driver is present or not. This kind of signal subsets are similar for a certain group of users, but not all users.

The third type, *type III* ($S_\mathcal{A}^{III}$), includes all signal subsets S_i, which hold true for only *one specific user* and is unique for every user:

$$S_\mathcal{A}^{III} \stackrel{\text{def}}{=} \{S_i \mid \forall i, j \in \{1, \ldots, k\}, i \neq j \wedge S_i \neq S_j\} \tag{3.8}$$

These correlations are very user-specific and at the core of user-specific personalisation. An example is parallel usage of several functions at the start-up of the vehicle (i. e. which functions are triggered when the user is commuting from home to work).

For every DDF, the developer has to configure this type in the specification. This configuration is then used for the deployment strategy. The approach is either deployed (i) in the back end on test fleet data, (ii) in the back end on user data, (iii) onboard a group of vehicles, or (iv) onboard all vehicles. The lower part of Table 3.4 shows which deployment target can be used to identify which type of signal subset. Also, an iterative classification, from *type I* over *type II* to *type III* is possible if the developer is unsure about the classification.

If the DDF is categorised as *type I* the required signals can be easily selected on test vehicle data, and the approach is deployed in the back end on test vehicle data. In the case, the DDF is categorised as *type II* actual user data is required as test vehicle data contain interactions of test drivers with vehicle functions and do not cover any user behaviour. Here, the approach is then either deployed on user data already present in the back end or if no data is yet available deployed onboard a group of vehicles. If the DDF is highly user-specific and is categorised as *type III*, the approach is directly executed onboard every vehicle where the DDF will be deployed.

In the case, the automatic input selection is configured, the approach must always be deployed onboard every vehicle, as every vehicle needs the input information for the deployment of the DDF. This strategy can also be integrated into existing Continuous Integration and Continuous Deployment (CI/CD) pipelines as proposed in our work on an automated CI/CD pipeline for DDFs (cf. [11]).

Chapter 4

Offline Evaluation

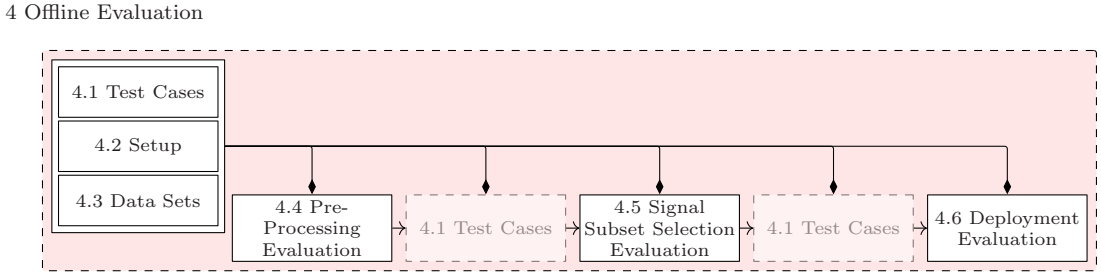


Figure 4.1: Chapter structure

For the assessment of the approach, we conduct multiple evaluations. This chapter presents the first three evaluations. Figure 4.1 depicts the structure of this chapter.

The first evaluation assesses the proposed vehicle signals pre-processing step and compares its performance to a baseline without any pre-processing. Next, based on these pre-processed signals, the labels needed for the next evaluation are generated. This part of the approach cannot be evaluated because it is only based on the specification for each test case. Next, we evaluate the signal subset selection step of the proposed approach.[1] The used algorithm profoundly impacts the performance of this step. To minimise the effect of the selected algorithm, we use 15 different algorithms for each test case and data set. In the last section, we evaluate the deployment strategy by comparing the different selected signal subsets out of the signal subset selection step before. All these evaluations are based on the same 24 test sets and 101 data sets. These evaluations are purely done offline without any streaming data or any onboard deployment. This will be evaluated in Chapters 5 and 6.

[1] A preliminary evaluation of the signal subset selection step using other data sets has been previously published at the 2019 IEEE ITSC (cf. [4]).

4.1 Test Cases

For all three evaluations conducted in this chapter, we use 24 different DDFs as test cases (cf. Table 4.1). We categorise these test cases into the classification presented in Section 2.1. First, we will present the test cases of the type *system function*, followed by *context-aware function*, and finally, all test cases of the type *anomaly detection*.

Table 4.1: Test case overview

No.	Test Case	Type
(1)	Day/Night Mode	System Function
(2)	Power Consumption	System Function
(3)	Valid Lane Markings	System Function
(4)	HVAC Driver	Context-Aware Function
(5)	HVAC Co-Driver	Context-Aware Function
(6)	Proactive ACC	Context-Aware Function
(7)	Proactive ACC Gap	Context-Aware Function
(8)	Proactive Window Driver	Context-Aware Function
(9)	Proactive Window Co-Driver	Context-Aware Function
(10)	Proactive Seat Heating Driver	Context-Aware Function
(11)	Proactive Seat Heating Co-Driver	Context-Aware Function
(12)	Proactive DEC	Context-Aware Function
(13)	Frontend Collision Warning	Anomaly Detection
(14)	Cross Traffic Alert	Anomaly Detection
(15)	Lane Departure Warning	Anomaly Detection
(16)	Lane Departure Sensitivity	Anomaly Detection
(17)	Lane Departure Intervention	Anomaly Detection
(18)	Lane Change Warning	Anomaly Detection
(19)	Lane Change Sensitivity	Anomaly Detection
(20)	Lane Change Intervention	Anomaly Detection
(21)	Side Collision Warning	Anomaly Detection
(22)	Speed Limit Assist	Anomaly Detection
(23)	Speed Limit Assist Offset	Anomaly Detection
(24)	Steering Wheel Vibration	Anomaly Detection

4.1.1 System Functions

Three out of the 24 test cases are pure system functions. These data-driven system functions are designed to model a part of the automotive system or a physical effect within the vehicle. In the automotive domain, a wide range of these functions have been developed, and with the advance of available computational resources, the number will further increase. These functions can range from data-driven battery management systems [31], air leak modelling and monitoring of combustion engines [32], intelligent active suspension systems [33], diagnosis of antilock braking systems [34], to fault diagnosis for automotive engines [35]. For the evaluation of our approach, we use the following three test cases of the type system function:

Day/Night Mode

For the first test case, we use the day and night mode function of the vehicle's Human-Machine Interface (HMI). This function is already part of today's vehicles and is based on an illumination sensor in the windscreen (cf. [151]). The prediction of this state should be easily achievable by any algorithm. In our case, we use this use case as a first "sanity check" of the algorithm's results. The label for the training is directly generated from the signal stating the current mode of the HMI. Therefore, this signal is needed to be added to the blacklist for the input of the DDF. If this signal would be included into the signal subset selection, it would identify this signal as input for the DDF, and it would directly learn from the label. This would not lead to any meaningful results. For each test case, the threshold $|S_{\mathcal{A}}| = n$ has to be set as a trade-off between accuracy and computational complexity. The higher the number n, the more possible signals can describe the context for a particular state, but could also lead to the before-mentioned *curse of dimensionality* and increased calculation time in the next steps. For all test cases in this evaluation, we choose the maximum number of $n = 30$ for the selected input size. As the first test case, we specify the following:

Test Case 1 (Day/Night Mode)
Prediction if the vehicle's HMI should be switched to day mode or night mode.

Listing A.1 (cf. Appendix A) shows the specification of this DDF. The other parts of the specification are similar for all test cases in this evaluation and are mentioned later.

Power Consumption Prediction

The second test case is also of the type system functions. The purpose of this DDF is to predict the power consumption by all E/E systems in the vehicle. This kind of functions are already proposed in different works and are partly already integrated into current vehicles (e. g. [152–154]). Here we use this test case to evaluate if the proposed approach can identify appropriate signals for the input of such a function. As a result of this, we specify the following test case:

Test Case 2 (Power Consumption)
Prediction of the electrical power consumption of all E/E systems in the vehicle.

In Listing A.2, the specification is shown for this DDF. As in the previous test case, the label can be directly created from an existing vehicle signal stating the current power consumption. In this case, we want to identify 15 different equally sized classes of power consumption, based on the current consumption. Additionally, all signals of already implemented functions of this kind are excluded as potential input signals.

Valid Lane Marking Prediction

The third test case of the type system function is a DDF to predict whether the detected lane markings are valid or not. The function lane marking detection is already part of current vehicles, and many different implementation methods have been proposed (mostly camera-based, e. g. [155–158]). However, we do not want to detect a lane marking, but we want to predict whether a lane marking is valid or not, based on the context information posed by the vehicle signals. As a result of this, we specify the following:

Test Case 3 (Valid Lane Markings)
Prediction if valid lane markings are present and can be used by other vehicle systems.

Listing A.3 shows the specification for this DDF. The label is created based on the signals representing the measurement of the current lane width. In case the lane marking cannot be identified, the signal has a special value. In our case $\overline{s_i} = 15$. As in the other cases, the signal which is used for the label generation is added to the backlist.

4.1.2 Context-Aware Functions

Another type of DDFs are *context-aware functions* (cf. Section 2.1). With the advance of intelligent systems, the customer demands adaptive and intelligent functions which learn according to their behaviour. These DDFs capture hidden knowledge for continuously improving their capabilities and providing a highly personalised user experience (also known as *context-aware systems* [36]). In the automotive domain, these functions range from intelligent window lifters [37], intelligent in-car infotainment systems [38], intelligent window wipers [39], proactive comfort functions [40], to intelligent cruise controls [41]. The principal functionality of these functions is already present in current vehicles. However, this basic functionality does not include any adaption to the user. For this evaluation, we used nine test cases of the type of context-aware DDFs. These functions range from HVAC functions to driving dynamics functions and are introduced in the following.

HVAC Temperature

The basis for the first two DDFs is the automatic HVAC system built in current vehicles (cf. [159, pp. 293ff]). The here used DDFs tries to predict the desired temperature of the vehicle's interior, for the driver and the co-driver separately. The current implemented automatic HVAC systems can automatically reach and hold the temperature defined by the user, but the user still has to select the preferred temperature manually. As the preferred interior temperature highly depends on the user's preferences and other contextual information, we try to predict the preferred temperature with a DDF. The idea behind this DDF is to learn the user's preferences and proactively set the temperature for the HVAC system according to past behaviour. HVAC systems in premium vehicles differentiate between the temperature for the driver and the co-driver (or even for every single passenger). Here, we only differentiate between the driver and

the co-driver, which results in two separate DDFs and test cases. The following two test cases are specified:

Test Case 4 (HVAC Driver)
Prediction of the desired temperature for the driver maintained by the HVAC system.

Test Case 5 (HVAC Co-Driver)
Prediction of the desired temperature for the co-driver maintained by the HVAC system.

Listings A.4 and A.5 show the specification for both test cases. The label in each test case is created based on the signals containing the current setting of the preferred temperature within bins of 1.5°C. In both cases, the signal used for the label generation is added to the blacklist to avoid any undesired correlations.

Adaptive Cruise Control

The next two test cases for a context-aware DDF are based on the ACC. The ACC itself is already part of the ADAS of current generation vehicles (cf. [160, pp. 478-521]). According to ISO 15622 (cf. [161]), the ACC is "an enhancement to conventional cruise control systems, which allows the subject vehicle to follow a forward vehicle at an appropriate distance by controlling the engine and/or power train and potentially the brake" [161]. The purpose of these two DDFs is to predict if the users want to activate/deactivate the system and at which distance the system should follow the vehicle in front.

The first DDF only predicts whether the function should be on or off and thereby can proactively engage/disengage the ACC. The purpose of the second DDF is to predict the following distance, which can be defined by the driver and which highly depends on the context and the driver's preferences. In ISO 15622 (cf. [161]) this time gap is defined as: "Time interval for travelling a distance, which is the clearance d between consecutive vehicles. The time gap τ is related to vehicle speed v and clearance d by: $\tau = d/v$" [161]. Current ACC systems do not allow the driver to exactly set this time gap, but gives the option of different predefined settings. In our case, the driver can choose between 1, 2, 3, or 4, where 1 is the closest setting and 4 the furthest setting (cf. [159, pp. 242ff]). Here the purpose of the second DDF is to predict this setting. Thereby, we define the following two test cases:

Test Case 6 (Proactive ACC)
Prediction of the activation/deactivation of the ACC.

Test Case 7 (Proactive ACC Gap)
Prediction of the preferred gap-distance for the ACC.

Listings A.6 and A.7 show the specification for both DDFs. In the first case, the label is directly created from the signal containing the current state of the ACC and in the second case based on the signal containing the current time gap setting. In both cases, the signals indicating

the status of the system, the current gap setting, and the maximum speed are added to the blacklist, as they contain the systems status and should not be used for the prediction within the DDF. Also, additional signals used for the internal configuration of the ACC are added to the blacklist. However, they are not further mentioned due to confidential information.

Windows

The basis for the next two test cases are the window lifters of the driver's and co-driver's window (cf. [159, pp. 106ff]). The purpose of these two DDFs is to predict the state of the driver's window, respectively the state of the co-driver's window. By this, the DDF can proactively control the window based on the user's preferences and context. The following two test cases are specified:

Test Case 8 (Proactive Window Driver)
Prediction of the preferred state of the driver's window.

Test Case 9 (Proactive Window Co-Driver)
Prediction of the preferred state of the co-driver's window.

Listings A.8 and A.9 show the specification for both test cases. In each test case, the label is directly created from the signal containing the window's state. The states for partly opened and fully opened are combined into one label. In both cases, the blacklist contains the signals for the window's state, the control of the window lifter and the button's state.

Seat Heatings

The basis for the next two DDFs is the seat heating of the driver's and the co-driver's seat (cf. [159, pp. 130f]), which was introduced in the last chapter. The purpose of these two test cases is to predict the preferred state of the driver's or co-driver's seat heating. By learning the user's behaviour, the DDFs can act proactively and switch on/off the seat heating:

Test Case 10 (Proactive Seat Heating Driver)
Prediction of the desired state of the driver's seat heating.

Test Case 11 (Proactive Seat Heating Co-Driver)
Prediction of the desired state of the co-driver's seat heating.

Listings A.10 and A.11 show the specification for both test cases. The label is directly created from the observed state of the seat heating used by the user. To eliminate any correlations with the function's output, the state signals, the control signals, and the temperature of the seat heating are added to the blacklist.

Driving Experience Control

The basis for the last context-aware DDF is the Driving Experience Control (DEC) system (cf. [159, pp. 145ff]). This function adjusts various driving dynamic properties (e. g. suspension, engine) and can be manually set by the driver. The driver has the choice between the modes, *comfort*, *sport*, or *eco pro*. Here, the purpose of the DDF is to predict the chosen driving mode in similar situations where the driver preferred a certain driving mode. For this DDF, we specify the following test case:

Test Case 12 (Proactive DEC)
Prediction of the preferred driving mode controlled by the DEC.

Listing A.12 shows the specification of this DDF. The label is directly created from the signal containing the current state of the DEC. To avoid that any outputs of this function being used as input, all output signals of the DEC are added to the blacklist. This includes not only the status of the DEC but also all signals which configure different vehicle components according to the driving mode.

4.1.3 Anomaly Detection

DDFs can also be used for *anomaly detection*. Anomaly detection is a process to find outliers on data by comparing with some predefined pattern or rules. Here, an outlier is defined as "patterns in data that do not conform to a well-defined notion of normal behaviour" [46]. In the automotive domain, anomaly detection is commonly used to detect anomalies in the system to recognise malicious attacks on the vehicle. Here, we aim to identify mistimed or unintended deactivation of vehicle functions, in particular, ADAS, at run-time by DDFs. In current vehicles, more and more ADAS are integrated. For personalisation purposes, these functions can be configured by the driver. For example, a driver can deactivate such systems manually. A deactivation per se is not a problem if intended by the driver. For instance, a driver might disable traction control when being stuck in an iced parking lot. However, if such a function is deactivated while driving at high speed due to a software or hardware fault, or an IT attack, this could lead to a severe problem. We evaluate our approach based on twelve ADAS functions. In each test case, the purpose of the DDF is to predict the configuration of the ADAS function based on similar situations seen before and thereby identify potential anomalies in the current configuration/state and inform the driver about the current configuration. By this, the driver is informed about potential deactivated ADAS functions and take countermeasures (e. g. change the driving style). Here we use the following twelve ADAS functions as the basis for the proposed DDFs.

Frontend Collision Warning

The first ADAS function for which we want to detect anomalies by using a DDF is the *frontend collision warning* system. According to ISO 15623 (cf. [162]) the purpose of the frontend

collision warning system "is to warn the driver when the subject vehicle encounters the situation of a forward vehicle in the subject vehicle's trajectory becoming a potential hazard" [162]. This system can be switched on and off by the driver, for personalisation purposes. For our evaluation, we define the following test case:

Test Case 13 (Frontend Collision Warning)
Anomaly detection for the status of the frontend collision warning system.

Listing A.13 shows the specification for this test case. The label is directly created from the configuration signals which can either be on or off. The blacklist contains the status and configuration signals of the frontend collision warning system.

Cross Traffic Alert

The second ADAS which can be configured by the driver and where we want to detect anomalies is the *cross traffic alert* system. "A rear cross traffic alert system for a vehicle is utilized to generate and display rear cross-traffic warning symbols to the driver during reverse gear manoeuvres" [163]. In our case, the system can be switched on or off (cf. [159, pp. 287ff]) and we specify the following test case:

Test Case 14 (Cross Traffic Alert)
Anomaly detection for the status of the cross traffic alert system.

Listing A.14 shows the specification for this test case. Similar to the test case before, the label is based on the status signal, and the blacklist contains the status and configuration signal.

Lane Departure Warning

The next ADAS function for which we want to identify anomalies in the configuration is the *lane departure warning*. The purpose of the lane departure warning system is to warn the driver if the driver unintentionally crossed the lane marking on the road (cf. [164–166]). Here the system has three configuration parameters which can be changed by the driver. The system itself can be activated or deactivated, the sensitivity of the system can be configured, and it can be configured if the car should automatically intervene in case of a lane departure [159, pp. 210ff]. For each of the parameters which can be configured, a separate DDF is implemented. We specify the following three test cases:

Test Case 15 (Lane Departure Warning)
Anomaly detection for the status of the lane departure warning system.

Test Case 16 (Lane Departure Sensitivity)
Anomaly detection for the sensitivity setting of the lane departure warning system.

Test Case 17 (Lane Departure Intervention)
Anomaly detection for the setting if the lane departure warning system should intervene by actively steering in the opposite direction.

Listings A.15 to A.17 show the specification of each test case. The labels are based on the signal containing the current configuration setting for which anomalies should be detected. In all test cases, the three signals containing the three configuration settings are added to the blacklist. The signal containing the status of the lane departure warning system is added to the blacklist, to avoid any dependencies on the status of the function.

Lane Change Warning

The next test case for which we want to identify anomalies is based on the ADAS function *lane change warning*. The purpose of this function is to warn the driver if they want to change the lane, and the lane is occupied by a different vehicle. In ISO 17387 (cf. [167]), three different types of lane change warning systems—also known as lane-change-decision-aid-systems—are defined. In our case, the system is a type three system which can detect vehicles in the blind spot, or approaching vehicles from behind in the adjacent lanes (cf. [159, pp. 213ff]). Similar to the lane departure warning system, the driver can configure this function with three different settings. The system can be activated or deactivated, the sensitivity of the system can be configured, and it can be configured if the car should automatically intervene in case of a lane change onto a lane occupied by another vehicle. For our evaluation, we specify the following three test cases:

Test Case 18 (Lane Change Warning)
Anomaly detection for the status of the lane change warning system.

Test Case 19 (Lane Change Sensitivity)
Anomaly detection for the sensitivity setting of the lane change warning system.

Test Case 20 (Lane Change Intervention)
Anomaly detection for the setting if the lane change warning system should intervene by actively steering in the opposite direction.

Listings A.18 to A.20 show the specification for each test case. In each test case, the label is based on the signal containing the current configuration setting for which anomalies should be detected. In all test cases, the three signals containing the three configuration settings are added to the blacklist. Also, the signal containing the status of the lane departure warning system is added to the blacklist, to avoid any dependencies on the status of the function.

Side Collision Warning

The next test case is based on the ADAS function *side collision warning*. This system helps the driver to avoid side collisions with vehicles on the adjacent lanes by a warning and steering intervention (cf. [159, pp. 217ff]). This system can be either activated or switched off by the driver. For this function, we specify the following test case:

Test Case 21 (Side Collision Warning)
Anomaly detection for the status of the side collision warning system.

Listing A.21 shows the specification of this test case. Similar to the other test cases, the label of the DDF is created based on the status signal, and the blacklist contains the status and configuration signal.

Speed Limit Assist

The basis for the next test case is the *speed limit assist* system. The system uses the traffic sign detection and other information sources to detect the speed limit on the current road. It can automatically transfer this information, for example, to the ACC system as a new speed setting [159, pp. 251ff]. This system can either be activated or deactivated, and the offset to the actual speed limit can be configured. Here we consider the activation/deactivation and the offset as a separate test case. Thereby we specify the following two test cases for this function:

Test Case 22 (Speed Limit Assist)
Anomaly detection for the status of the speed limit assist.

Test Case 23 (Speed Limit Assist Offset)
Anomaly detection for configuration of the offset speed for the speed limit assist.

Listings A.22 and A.23 show the specification for both test cases. In the first test case, the label is directly created from the status signal. In the second test case, the label is created based on the signal containing the offset setting in bins of 2 km/h each, leading a total of 15 classes. In both test cases, the signals containing the status, configuration and offset are added to the blacklist.

Steering Wheel Vibration

Many of those functions mentioned above can use tactile feedback on the steering wheel to inform or warn the driver (cf. [168]). Here this is achieved through a vibration of the steering wheel, which can be configured globally for all ADAS in the modes: *light*, *medium*, and *strong* [159, pp. 212,215,219]. For this function, we implement the following test case:

Test Case 24 (Steering Wheel Vibration)
Anomaly detection for the steering wheel vibration strength for the aforementioned ADAS.

Listing A.24 shows the specification for this test case. Similar to the other test cases, the label is directly created from the status signals. In this case, the blacklist contains the status and configuration signal.

4.2 Setup

In the foregoing section, we have only defined the function and the labeling. In this section, we specify the missing configuration parameters of the DDF specification for the previously introduced test cases. This includes the configuration and setup of the deployment class, pre-processing, additional signals generation rules and the post-processing. These configuration parameters are similar for all test cases, but some can differ based on the type of the test case and the used algorithms.

4.2.1 Deployment Class

The first configuration parameter is the deployment class. In the case of the deployment class, we differentiate between the test cases of the type system function and the other two (context-aware function and anomaly detection). All functions specified as system functions are specified in the deployment class system (i. e. type I) (cf. Listing B.1 in Appendix B). These functions only correlate to physical input and will not correlate to any user behaviour. The signal subset can be identified based on data of only one user or one test vehicle. All other test cases of the type of context-aware functions and anomaly detection for ADAS are defined as deployment class group (i. e. type II) (cf. Listing B.2). For these functions, we assume that if enough vehicles are selected, a representative of every group is selected. By this, the appropriate signal subset for each function can be found. These assumptions will be later evaluated in the deployment evaluation.

4.2.2 Pre-Processing

The next configuration parameter is the pre-processing setup. In the following offline evaluations, we differentiate between three different pre-processing setups.

For the first evaluation of the pre-processing step, the specification is shown in Listing C.1 (cf. Appendix C). Here, we include all error-values, as we compare the raw signals to the pre-processing signals, and both data sets should contain the same information. Also, we scale all signals in the range of $[0, 1]$, which is common for machine learning tasks. In this case, we do not discretise any signals as we would like to contain as much information as possible.

For the second evaluation, the signal subset evaluation, we differentiate between two different pre-processing setups (cf. Listings C.2 and C.3). In both setups, all error-values are removed, as we want to learn on contextual information and not on any vehicle specific error messages. In both cases, all signals are again scaled within the range of $[0, 1]$. The only difference between both setups is the binning of the signal values in discrete bins. As some feature selection algorithms can only handle discrete data, and some of the vehicle signals contain continuous data, we have to discretise these signals in case such an algorithm is used. Table 4.5 states which algorithms requires which pre-processing, and will be later introduced.

4.2.3 Additional Signals

In all test cases and for the following evaluations, we do not specify any additional signals. Here, the evaluations focus on the assessment of the proposed approach and its capability of identifying the appropriate vehicle signals, and we do not focus on the development of a special DDF. Therefore, in all specifications, the configuration for additional signals (`add_signals`) is left empty.

4.2.4 Post-Processing

To evaluate the selected signal subset, we directly use the resulting signals as input for the DDF. In this case, the selected signal subset $S_{\mathcal{A}}$ is automatically used for the dynamic input of the DDF. The static input will be left empty as we want only to evaluate the signal subsets. As a machine learning component of the DDF, we will use two different machine learning algorithms. This component is learning the representation between the input $\boldsymbol{x}$ and the output $\boldsymbol{y}$ and is trained with the same label input $\hat{\boldsymbol{y}}$ as the label information used for the selection step (i. e. $\hat{\boldsymbol{y}} = s_l$). We then evaluate the trained function on a test data set from the same user and assess the performance of the DDF. For all test cases, we will use the same implementation and algorithms. Here, we do not use the report for the evaluation of the performance of the approach. This evaluation would have to be performed manually, which would be highly subjective.

Based on the ranking (i. e. ordered list of scored signals) of the output of the proposed approach, we trained a *State Vector Machine (SVM)* classifier (cf. [169]) with the top $|S_{\mathcal{A}}| = 30$ ranked signals to predict the function's state. We have selected this type of classifier, because of its robustness to the *curse of dimensionality* [170] and simple replication for similar evaluations. As a kernel of the SVM we used a *radial basis function kernel* with a *kernel coefficient* $\gamma = 1/|S_{\mathcal{A}}|$ and *shrinking heuristic*.

As a second algorithm, we trained a *Random Forest (RF) classifier* (cf. [171]), with again the top $|S_{\mathcal{A}}| = 30$ ranked signals identified by the proposed approach. We have selected this type of classifier, because of its different mechanism compared to a SVM classifier and its simple replication. We used 200 trees with a minimum number of samples required to split an internal node of 60 and a minimum number of samples required per leaf node of 30.

In order to evaluate the performance of each DDF, we use the Matthew's Correlation Coefficient (MCC) [172], also known as phi coefficient [173, pp. 282f]. In the prediction of K classes (i. e. possible values of s_l) by the classifier c the MCC is defined as:

$$MCC(c) = \frac{ts - \sum_{k=1}^{K} p_k o_k}{\sqrt{(s^2 - \sum_{k=1}^{K} p_k^2)(s^2 - \sum_{k=1}^{K} o_k^2)}} \tag{4.1}$$

where o_k represents the number of occurrences of class k, p_k the number of predictions of class k, t the total number of correct predictions, and s the total number of samples. [174]

The value ranges between $MCC(c) \in [-1; 1]$, where $MCC(c) = 1$ represents the perfect

classifier and $MCC(c) = -1$ the worst classifier. In the case of $MCC(c) = 0$ the classifier performs similar to a static classifier always predicting the same value. In the case of predicted classes $K > 2$, the minimum value of $MCC(c)$ ranges between -1 and 0. [174]

We further denote the classifier c trained on the signal subset S_i as $c(S_i)$ and the MCC of this classifier as:

$$mcc_i \stackrel{\text{def}}{=} MCC(c(S_i)) \tag{4.2}$$

The implementation of the SVM classifier, RF classifier and the calculation of the MCC is based on the Python package *scikit-learn* [175]. All other configuration parameters not mentioned before are kept in the default setting based on version 0.20 of scikit-learn.[1]

4.3 Data Sets

For the evaluation of the approach, we have used real vehicle data from current-generation BMWs. In this section, we first introduce the data sets and how they have been collected, followed by a breakdown of each data set and the usage of the different test cases.

4.3.1 Data Set Overview

Figure 4.2: BMW 7 Series generation used for data collection [178]

The used vehicle data sets were collected during a field study which has been conducted at the BMW Group. This study was conducted in multiple countries also to cover different behaviour depending on the geographic region and cultural characteristics. These data sets originate from vehicle traces logged from current-generation BMW 7 Series [178] driven by real customers (cf. Figure 4.2). All data have been collected compliant to local laws and regulations and the consent of each driver/customer, which have explicitly participated in this field study. Each vehicle trace contains all vehicle data frames which were sent over the major internal vehicle communication networks (e. g. CAN, FlexRay, Ethernet). From these traces, all signals

[1] *Scikit-learn* v0.20 documentation: SVM classifier [176] and RF classifier [177]

Table 4.2: Data sets used for evaluation

(a) Number of signals in each data set

Raw Signals *Pre-filtered*	**Pre-Processed Signals** *Error-values included*	**Pre-Processed Signals** *Error-values filtered*
4294	19890	14047

(b) Samples and length of each data set

Data Set	**Samples**	**Length**	**Data Set**	**Samples**	**Length**	**Data Set**	**Samples**	**Length**
1	50935	141.5h	**35**	6438	17.9h	**69**	1227	3.4h
2	29475	81.9h	**36**	4888	13.6h	**70**	1177	3.3h
3	27991	77.8h	**37**	4764	13.2h	**71**	1037	2.9h
4	26494	73.6h	**38**	4626	12.8h	**72**	973	2.7h
5	26077	72.4h	**39**	4586	12.7h	**73**	871	2.4h
6	22792	63.3h	**40**	4498	12.5h	**74**	766	2.1h
7	21552	59.9h	**41**	4210	11.7h	**75**	612	1.7h
8	21139	58.7h	**42**	4154	11.5h	**76**	545	1.5h
9	19532	54.3h	**43**	3983	11.1h	**77**	498	1.4h
10	17757	49.3h	**44**	3760	10.4h	**78**	463	1.3h
11	17545	48.7h	**45**	3597	10.0h	**79**	463	1.3h
12	17070	47.4h	**46**	3426	9.5h	**80**	458	1.3h
13	16841	46.8h	**47**	3336	9.3h	**81**	450	1.2h
14	15823	44.0h	**48**	3306	9.2h	**82**	444	1.2h
15	13791	38.3h	**49**	2958	8.2h	**83**	423	1.2h
16	13096	36.4h	**50**	2853	7.9h	**84**	342	0.9h
17	12532	34.8h	**51**	2250	6.2h	**85**	331	0.9h
18	12220	33.9h	**52**	2217	6.2h	**86**	309	0.9h
19	11879	33.0h	**53**	2182	6.1h	**87**	274	0.8h
20	10599	29.4h	**54**	2090	5.8h	**88**	251	0.7h
21	10423	29.0h	**55**	2034	5.7h	**89**	240	0.7h
22	9573	26.6h	**56**	2026	5.6h	**90**	219	0.6h
23	9565	26.6h	**57**	1911	5.3h	**91**	176	0.5h
24	9338	25.9h	**58**	1900	5.3h	**92**	174	0.5h
25	9219	25.6h	**59**	1836	5.1h	**93**	165	0.5h
26	8237	22.9h	**60**	1738	4.8h	**94**	158	0.4h
27	7706	21.4h	**61**	1593	4.4h	**95**	139	0.4h
28	7506	20.9h	**62**	1564	4.3h	**96**	82	0.2h
29	7489	20.8h	**63**	1549	4.3h	**97**	59	0.2h
30	7173	19.9h	**64**	1493	4.1h	**98**	56	0.2h
31	7083	19.7h	**65**	1343	3.7h	**99**	45	0.1h
32	6917	19.2h	**66**	1309	3.6h	**100**	27	0.1h
33	6886	19.1h	**67**	1286	3.6h	**101**	12	0.1h
34	6880	19.1h	**68**	1252	3.5h			

containing information for error detections (e. g. alive counters, CRCs) have been filtered. The vehicle data from each driver was extracted into a separate data set. This splitting is based on the identification of each driver using an interior camera part of the study setup. To optimise the data handling all vehicle signals have been resampled with a sampling rate of $10s$. As the DDFs of the test cases only represent behaviour within this time range (e. g. seat heating), the resampling does not affect the evaluation.

Table 4.2 gives an overview of the used data sets, the number of signals, and length of each data set. Table 4.2a shows the number of signals in each data set. The first number represents the number of vehicle signals after the first pre-filtering of operational signals (i. e. $|S|$). The second one represents the number of signals after the pre-processing step chapter without any filtering active (i. e. $|S_{pre}|$). The last number, the number of pre-processed signal with filtering of all error-values (i. e. $|S_{pre}|$). The increase of vehicle signals after pre-processing shows the high amount of encoded vehicle signals within other vehicle signals. Without any pre-processing, this information would be inaccessible for the algorithms used for the signals subset selection and the training of the DDF.

Table 4.2b gives an overview of each data set and the recorded samples and total length. The period of each data variates due to the actual time the driver was driving the vehicle. The full length of the data sets vary between a few minutes up to $141.5h$.

4.3.2 Test Cases in Data Sets

For this evaluation, each data set has to be divided into a training set and test set. The first 66% of each recorded data set is used to run the proposed approach, and the last 34% of each data set is used to test and evaluate the approach. The purpose of all test cases is to predict a specific state of a function. If the function does not change in the data set, the function is not able to learn any behaviour of a function as the output is static. For example, if we try to learn if whether the seat heating should be switched on or off and we train the DDF on data where the seat heating is never switched on, the DDF would never switch on the seat heating either. If we evaluate the approach on these data sets, the evaluation results would not contain any meaningful data. Therefore, we only use a data set for a test case where the function contains at least two different states in the training set and in the test set. Table 4.3 gives an overview of each test case and on how many data sets we can evaluate the approach. In total, we have data for 2424 evaluation runs ($101 \times 24 = 2424$). In 680 runs the function basis for the test case have been in at least two different states in the training and test set. In 302 cases, only in the training set or the test set at least two different states have been observed. In 1442 cases, the function which is the basis for the test cases was static. Therefore, we can in total evaluate the approach with 680 runs, in this offline evaluations.

The four test cases *Frontend Collision Warning*, *Cross Traffic Alert*, *Side Collision Warning*, and *Speed Limit Assist* have never been observed in at least two different states/configurations in the training and test set for any of the data set. In this case, these four test cases cannot be

used for the following evaluations as no behaviour can be learned. All other test cases have at least one data set where at least two labels where observed within the training and test data set.

Table 4.3: Test cases and usage over data sets

Test Case	**Train and Test**	**Train or Test**	**Never**
(1) Day/Night Mode	69	18	14
(2) Power Consumption	96	4	1
(3) Valid Lane Markings	82	11	8
(4) HVAC Driver	49	30	22
(5) HVAC Co-Driver	48	31	22
(6) Proactive ACC	58	11	32
(7) Proactive ACC Gap	46	16	39
(8) Proactive Window Driver	74	14	13
(9) Proactive Window Co-Driver	43	27	31
(10) Proactive Seat Heating Driver	40	20	41
(11) Proactive Seat Heating Co-Driver	22	29	50
(12) Proactive DEC	32	19	50
(13) Frontend Collision Warning	-	-	101
(14) Cross Traffic Alert	-	1	100
(15) Lane Departure Warning	1	4	96
(16) Lane Departure Sensitivity	3	3	95
(17) Lane Departure Intervention	2	5	94
(18) Lane Change Warning	1	3	97
(19) Lane Change Sensitivity	3	12	86
(20) Lane Change Intervention	2	4	95
(21) Side Collision Warning	-	-	101
(22) Speed Limit Assist	-	4	97
(23) Speed Limit Assist Offset	8	18	75
(24) Steering Wheel Vibration	1	18	82
Total	**680**	**302**	**1442**
			2424

4.4 Pre-Processing Evaluation

In this first evaluation, we only evaluate the pre-processing step of the approach. For this evaluation, we compare the performance of a DDF using all raw signals (i. e. S) as an input versus a DDF using all pre-processed signals (i. e. S_{pre}). In this section, we first introduce the evaluation setup, followed by the used metrics. The next paragraph states the gained results and their discussion. Finally, we assess the threats to validity for this particular evaluation.

4.4.1 Setup

For this evaluation, we compare the performance of both machine learning algorithms SVM and RF on the raw vehicle signals and a pre-processed signals. The specification for the pre-

processing is shown in Listing C.1. We include all error-values, as we compare the raw signals to the pre-processed signals, and both signals should contain the same information. In this case, we do not discretise any signal as we would like to preserve as much information as possible. Each classifier is then trained with all signals as input for each test case on the corresponding training set.

4.4.2 Metrics

We evaluate the performance of each trained classifier on the corresponding test data set with the following metrics. Let $c(S_i)$ be the classifier trained and evaluated on with the signal subset S_i. The MCC (i. e. performance) of the classifier $c(S_i)$ trained with the signal subset S_i is denoted by mcc_i.

In the case, the classifier c was trained on the pre-processed signal set S_{pre} we denote the MCC as mcc_p and in the case of the raw signal set S as mcc_r:

$$mcc_p \stackrel{\text{def}}{=} mcc(c(S_{pre})) \tag{4.3}$$

$$mcc_r \stackrel{\text{def}}{=} mcc(c(S)) \tag{4.4}$$

By comparing both MCCs, we can compare the performance of the classifier on the raw signals and the pre-processed signals and evaluate the benefits/drawbacks of the pre-processing step.

To compare the difference of a classifier c trained on the same data set, but with different signal sets, we introduce the difference of the MCCs between two different sets S_i and S_j. We denote this difference as $mcc_{\Delta ij}$:

$$mcc_{\Delta ij} \stackrel{\text{def}}{=} mcc(c(S_i)) - mcc(c(S_j)) \tag{4.5}$$

In the case the classifier $c(S_i)$ outperforms the classifier $c(S_j)$, the value is positive and vice versa for negative values. Furthermore, we denote the difference between the MCCs on the raw signal subset and pre-processed signal subset as $mcc_{\Delta pr}$:

$$mcc_{\Delta pr} = mcc_p - mcc_r = mcc(c(S_{pre})) - mcc(c(S)) \tag{4.6}$$

If this value is positive, the classifier trained on the pre-processed signals outperforms the classifier trained on the raw signals and vice versa for a negative value.

4.4.3 Results and Discussion

Table 4.4 gives an overview of the evaluation results. First, we state the number of runs for each test case. In the case of four test cases, the function has not been used in the test and

Table 4.4: Results of pre-processing evaluation

Test Case	Runs	RF Cl.			SVM Cl.		
		mcc_p	mcc_r	$mcc_{\Delta pr}$	mcc_p	mcc_r	$mcc_{\Delta pr}$
(1) Day/Night Mode	69	0.60	0.67	-0.06	0.49	0.00	0.49
(2) Power Consumption	96	0.05	0.16	-0.11	0.00	0.00	0.00
(3) Valid Lane Markings	82	0.01	0.01	-0.00	0.00	0.00	0.00
(4) HVAC Driver	49	0.07	0.08	-0.02	0.06	0.00	0.06
(5) HVAC Co-Driver	48	0.08	0.10	-0.02	0.07	0.00	0.07
(6) Proactive ACC	58	0.38	0.43	-0.05	0.30	0.00	0.30
(7) Proactive ACC Gap	46	0.23	0.25	-0.02	0.19	0.00	0.19
(8) Proactive Window Driver	74	0.03	0.03	-0.00	0.01	0.00	0.01
(9) Proactive Window Co-Driver	43	0.02	0.02	0.00	0.02	0.00	0.02
(10) Proactive Seat Heating Driver	40	0.16	0.22	-0.06	0.10	0.00	0.10
(11) Proactive Seat Heating Co-Driver	22	0.12	0.14	-0.02	0.13	0.00	0.13
(12) Proactive DEC	32	0.10	0.14	-0.03	0.06	0.00	0.06
(13) Frontend Collision Warning	-	-	-	-	-	-	-
(14) Cross Traffic Alert	-	-	-	-	-	-	-
(15) Lane Departure Warning	1	0.00	0.00	0.00	0.00	0.00	0.00
(16) Lane Departure Sensitivity	3	0.28	0.37	-0.10	0.03	0.00	0.03
(17) Lane Departure Intervention	2	0.35	0.31	0.04	0.87	0.00	0.87
(18) Lane Change Warning	1	0.32	0.15	0.17	0.54	0.00	0.54
(19) Lane Change Sensitivity	3	0.20	0.43	-0.23	0.00	0.00	0.00
(20) Lane Change Intervention	2	0.35	0.31	0.04	0.46	0.00	0.46
(21) Side Collision Warning	-	-	-	-	-	-	-
(22) Speed Limit Assist	-	-	-	-	-	-	-
(23) Speed Limit Assist Offset	8	0.03	0.12	-0.10	0.07	0.00	0.07
(24) Steering Wheel Vibration	1	0.00	0.00	0.00	0.00	0.00	0.00
***Total / Average** (over all runs)*	**680**	**0.16**	**0.19**	**-0.04**	**0.12**	**0.00**	**0.12**

training at least once, and therefore we cannot train and evaluated a classifier on these test cases. In the table, the results for each type of classifier are separately stated. In each first column, the average of the MCC over all runs of each test case is reported. This is done once for the classifier trained on the preprocessed signals (i. e. mcc_p) and once for the classifier trained on the raw signals (i. e. mcc_r). In the third column, the average of the difference of the MCCs (i. e. $mcc_{\Delta pr}$) over all runs of each test case is reported. In the last row, the total number of runs is reported, and the average of the MCC of all runs for each column.

When we compare mcc_p and mcc_r for the RF classifier (i. e. $mcc_{\Delta pr}$), we can see that the classifier trained on the raw signals outperforms the classifier trained on the pre-processed signals in twelve out of 24 test cases. In all of these twelve test cases, we can only see in two test cases a significant outperformance of the classifier trained on the raw signals (i. e. $mcc_{\Delta pr} = -0.23$ and -0.11). However, this is only the case for test cases with a low number of runs. In all other test cases, the classifier trained on the raw signals does only slightly outperform or even perform worse than the classifier trained on the pre-processed signals. In average over all runs the difference of the MCCs between both classifiers is relatively low (i. e. $mcc_{\Delta pr} = -0.04$).

When comparing the performance of the SVM classifier trained on the different signal sub-

sets, we can see that the classifier trained on the raw signals performed as good as a classifier always predicting the same value (i. e. static classifier). This results from the linear nature of this type of classifier and its incapability of splitting signals at a particular value and scaling signals during training. However, in the case of automotive signals, this step is required for raw vehicle signals due to its structure. Here, the classifier trained on the pre-processed data clearly outperforms the classifier trained on the raw signals.

This evaluation shows the importance of the pre-processing of signals for machine learning tasks. Often this step is manually designed for each use case. In the case of automotive data and the vast amount of vehicle signals, this step has to be automatically performed, based on the already present signal specifications. When using machine learning algorithms which are capable of splitting and scaling signals during the training phase (e. g. RF, decision trees), the proposed pre-processing step is not necessarily required. Nevertheless, it can be used to filter and reduce the amount of data and does not have a significant impact on the performance of the classifier. In the case of linear machine learning algorithms (e. g. SVM, deep neural networks), we can see that a pre-processing step is mandatory for the training of the classifier. With the current trend of deep learning (i. e. deep neural networks), the correct pre-processing of vehicle signals becomes more important. Additionally, the pre-processing step is agnostic of the actual function. It can be reused, even without applying the following steps of the approach and can be used for other data analytics or machine learning tasks based on vehicle signals.

4.4.4 Threats to the Validity

A significant threat to the validity of this evaluation results from the selection of the data set and test cases. There is a vast amount of in-house test vehicle data; however, we purposely used real user data to demonstrate the approach's practical feasibility. As a premium car manufacturer, the privacy of its customers is highly considered. Hence, it is challenging—even as an OEM—to get a large number of full customer car traces. Due to limited functions accessible in the used vehicles, we had to limit this evaluation to a selection of test cases. Therefore, the results of the evaluation are only valid for these test cases. We tried to select a wide range of user-functions to generalise the results for other use cases, but the results may vary for different use cases and users.

Another threat to validity results from the selection of the machine learning algorithm used for the classifier and the used metric to evaluate the classifier. By selecting the machine learning algorithms RF and SVM, we tried to capture the performance of different types of classifiers. Due to limited computational resources, we had to limit this evaluation on these two types of algorithms, but this could be easily extended. We selected the MCC as the primary metric because it can be compared between test cases with a different number of labels. Additionally, unbalanced data sets have none to little impact, which is the case here. Also, other metrics could be used, but as we directly compare the metrics on each other, the impact should be negligible.

4.5 Signal Subset Selection Evaluation

In this section, we will evaluate the signal subsets selected by the approach. Here, we first introduce the evaluation setup, followed by the metrics used. Next, we present the gained results and discuss these. Finally, we assess the threats to validity for this evaluation.

4.5.1 Setup

The signal subset selection step is the third step of the approach. As the basis for this step, we use the same data sets and test cases presented at the beginning of this chapter. To run this step, we also perform the pre-processing step and the label generation step beforehand. On these pre-processed signals and generated label information, we run the actual selection step.

Table 4.5: Feature selection algorithms used for evaluation

Algorithm	Full Name	Input Data	Ref.
CIFE	*(Conditional Infomax Feature Extraction)*	discrete	[122]
CMIM	*(Conditional Mutual Information Maximization)*	discrete.	[123]
Chi^2	-	discrete	[124]
DISR	*(Double Input Symmetrical Relevance)*	discrete	[125]
FCBF	*(Fast Correlation Based Filter)*	discrete	[126]
FScore	-	disc. & cont.	[127]
Fisher Score	-	disc. & cont.	[128]
Gini Index	-	discrete	[129]
ICAP	*(Interaction Capping)*	discrete	[130]
JMI	*(Joint Mutual Information)*	discrete	[131]
MIFS	*(Mutual Information Feature Selection)*	discrete	[132]
MIM	*(Mutual Information Maximization)*	discrete	[133]
MRMR	*(Minimum Redundancy Maximum Relevance)*	discrete	[134]
ReliefF	-	disc. & cont.	[135]
Trace Ratio Fisher	-	disc. & cont.	[136]

The performance of this step heavily relies on the performance of the used feature selection algorithm. To exclude this impact on our evaluation, we evaluate the presented approach with 15 state-of-the-art feature selection algorithms. The selection of the algorithms is based on the extensive survey paper by Li et al. [100]. The here used algorithms are listed in Table 4.5. This table also states the required input data type of each algorithm. Here, we only differentiate between algorithms capable of processing only discrete data or algorithms capable of processing discrete and continuous data. In case the algorithm requires discrete input data, we extend the pre-processing step with a discretisation step for each vehicle signal (cf. Section 4.2.2). In this case, each signal with continuous data is discretised into 20 equally sized bins, based on the defined range of the signals. By evaluating our approach with multiple algorithms, we can also identify the best-suited algorithms for automotive signals—at least for the presented test cases. The implementation of the feature selection algorithms is based on the Python package *scikit-feature* [100].

Based on the ranking (i. e. ordered list of scored signals) of each feature selection algorithm, we trained a SVM classifier and a RF classifier with the top $|S_{\mathcal{A}}| = 30$ ranked signals to predict the DDF's state. This is similar to the last evaluation, but now we only use the top 30 signals instead of all. Also, in this case, we remove all error-values in the pre-processing step.

4.5.2 Metrics

To assess the results of the evaluation, we introduce four different metrics based on the MCC:

First, let the signals selected by each algorithm be denoted as $S_{\mathcal{A}}$, where $S_{\mathcal{A}} \subseteq S_{pre}$. Each classifier c is then trained on this selected signal subset $S_{\mathcal{A}}$. The classifier's MCC using the signal subset $S_{\mathcal{A}}$ is denoted as:

$$mcc_{\mathcal{A}} \stackrel{\text{def}}{=} mcc(c(S_{\mathcal{A}})) \tag{4.7}$$

We calculate the mean of $mcc_{\mathcal{A}}$ for each algorithm on each test case over all data sets. Let $|S_{\mathcal{A}}|$ be defined by the corresponding maximal input of the DDF as specified in the specification (here $|S_{\mathcal{A}}| = 30$).

Next, we compare each trained $c(S_{\mathcal{A}})$ with a trained classifier using all signals S_{pre} as training input (i. e. $c(S_{pre})$). We compare the $mcc_{\mathcal{A}}$ with the classifier trained with all signals $S_{\mathcal{A}}$, by computing the difference, denoted as $mcc_{\Delta\mathcal{A}p}$:

$$mcc_{\Delta\mathcal{A}p} = mcc_{\mathcal{A}} - mcc_{p} \tag{4.8}$$

We determine the mean of the differences for each algorithm and test case independently. This indicator shows whether the selected signal subset represents the given information adequately and whether $c(S_{\mathcal{A}})$ and $c(S_{pre})$ performed similarly. If this value is positive, the classifier trained on the selected signals outperforms the classifier trained on all signals and vice versa for a negative value.

Next, we count the number of runs k where $c(S_{\mathcal{A}})$ was at least performing as good as $c(S_{pre})$, i. e. $mcc_{\Delta\mathcal{A}p} \geq 0$. By this, we want to show how often the classifier trained on the selection signals is at least as good as the classifier trained on all signals or even outperforming the other classifier.

To illustrate the execution time of each algorithm, we measured the execution time t of each algorithm per run. This execution time heavily depends on the test case, the size of the data set, and the current workload on the hardware the algorithm is executed. To exclude outliers and to assess a rough estimate of the execution time, we will only use the median execution time of all runs per algorithm, i. e. $\tilde{t}$. When assessing this execution time, we aim at getting a feeling of the applicability of the respective algorithms in an onboard execution scenario. These values are subject to correction, due to the high dependability to the implementation of each algorithm. The evaluation was performed on a workstation with an Intel Core i7-5930K and 64 GiB RAM [179].

4.5.3 Results and Discussion

The results for both types of classifiers and the four metrics can be found in Tables 4.6 to 4.12. Tables 4.6 and 4.7 give a comprehensive overview of the MCCs of the 10 200 runs (680 test cases and data sets × 15 algorithms) for each type of classifier. Tables 4.8 and 4.9 shows the results for $mcc_{\Delta\mathcal{A}p}$ and Tables 4.10 and 4.11 the results for the number of runs k, where $mcc_{\Delta\mathcal{A}p} \geq 0$. Finally, Table 4.12 shows the results of the time measurements.

When comparing the MCCs (cf. Tables 4.6 and 4.7), we observe different values throughout all algorithms and test cases. The values compared between the two different types of classifiers are almost identical to each other. The test cases 1, 6, 7, 12, 17, 18, and 20 show the best results, whereas the test cases 3, 8, 9, 15, 23, and 14 we can see MCCs close to 0. The test cases with the best values are not limited to one type of test case and are equally distributed between the three types. Almost all MCCs are $mcc_{\Delta\mathcal{A}p} \geq 0$, which means in almost all cases, the classifier was at least as good as a static classifier would be.

When comparing the performance of the classifier trained on the selected signal subset $mcc_{\mathcal{A}}$ with the performance of the classifier trained on all signals mcc_A (cf. Tables 4.8 and 4.9), we see that the algorithms Chi^2, DISR, FScore, Fisher Score, Gini Index, MRMR, and Trace Ratio show the least difference $mcc_{\Delta\mathcal{A}p}$ when evaluated with a SVM classifier. All these seven algorithms were able to select signal subsets as input for a SVM classifier which improved its performance compared to a classifier using all signals as input. Only in four test cases (11, 12, 17, and 20), the performance was worse. When using the selected signal subset to train a RF classifier, the same seven algorithms showed the best performance compared to a classifier trained on all signals.

Considering the number of runs k in which $mcc_{\Delta\mathcal{A}p} \geq 0$ (cf. Tables 4.10 and 4.11), we see that the algorithms FScore, Fisher Score, and Gini Index show the highest number of runs. Besides, the algorithm Trace Ratio indicates a high number in the case of the SVM classifier and the algorithm MRMR in the case of the RF classifier. The number of runs k between the SVM and RF classifier are similar for each of the algorithms.

Comparing the execution times (cf. Table 4.12) only the algorithms Chi^2, FScore, Fisher Score, ReliefF, and Trace Ratio had a median execution time lower than one minute. The other algorithms show execution times up to many hours per data set and test case. We can also see a difference between the test case when comparing the same algorithm. This difference results from the different data sets used for the evaluation of each test case.

Additionally, to the here shown results where we trained each classifier with the $30 = |S_A|$ top signals, we evaluated the results when changing the number of selected signals. When varying the size of the selected signal subset and comparing the resulting MCC of the respective classifier, we observed that over all runs the best performance is achieved with a various number of selected signals and the results are close to each other. This results from the used machine learning algorithms and their robustness against redundant or irrelevant inputs; here redundant or irrelevant signals in the selected signals subset.

To sum up, the algorithms Chi^2, DISR, FScore, Fisher Score, Gini Index, MRMR, and Trace Ratio provided signal subsets that achieved good and robust results for the trained classifier over all test cases independent of the used type of classifier. Additionally, from these algorithms, Chi^2, FScore, Fisher Score, and TraceRatio Fischer are distinguished by their fast execution. Despite their fast execution, the results are similar or even better than more complex algorithms with much higher execution times.

The overall good performance of the trained classifiers based on signals selected, showed the applicability of the proposed approach and also, that in most cases enough signals are already present to achieve good results for a DDF. When using the signal subset selected by seven out of the 15 algorithms, the performance of the classifier even increased in comparison to a classifier trained on all signals. This also shows the dependability of the approach to the performance of the used algorithm for selecting the signals.

4.5.4 Threats to the Validity

Besides the threats to the validity mentioned for the previous evaluation, the selection of the used algorithms might be a threat to the validity of this evaluation, as any supervised filter feature selection algorithm can be used as the basis for the selection. We tried to select the most widely used algorithms, but we had to limit the selection due to limited computational resources. The selection of algorithms can be easily extended as the approach is independent of the used algorithm. Another threat could also be seen in the selected metrics. In the field of machine learning, many different metrics for the performance-evaluation have been introduced. Here, we directly compared classifier against each other using the same metric to reduce the influence of the used metric.

Table 4.6: Signal subset evaluation results for $mcc_{\mathcal{A}}$ of the SVM classifier - Values highlighted where $mcc_{\mathcal{A}} \geq mcc_p$

Test	CIFE	CMIM	Chi^2	DISR	FCBF	FSc.	Fisher	Gini I.	ICAP	JMI	MIFS	MIM	MRMR	ReliefF	T Rat.	-
(1)	0.08	0.15	0.75	0.74	0.66	0.77	0.77	0.76	0.15	0.66	0.69	0.73	0.75	0.74	0.77	0.52
(2)	0.00	0.00	0.16	0.05	0.01	0.16	0.16	0.11	0.00	0.00	0.11	0.00	0.17	0.03	0.15	0.00
(3)	0.00	0.00	0.03	0.02	0.01	0.02	0.03	0.02	0.00	0.00	0.00	0.00	0.02	0.01	0.02	0.00
(4)	0.00	0.02	0.07	0.06	0.06	0.10	0.11	0.09	0.02	0.02	0.04	0.05	0.08	0.05	0.09	0.05
(5)	0.00	0.02	0.08	0.11	0.09	0.07	0.08	0.10	0.02	0.03	0.02	0.01	0.10	0.07	0.08	0.05
(6)	0.05	0.13	0.52	0.51	0.37	0.49	0.50	0.47	0.13	0.11	0.38	0.23	0.52	0.49	0.49	0.44
(7)	0.02	0.05	0.32	0.37	0.29	0.36	0.35	0.35	0.05	0.05	0.26	0.10	0.38	0.20	0.34	0.28
(8)	0.01	0.00	0.06	0.04	0.04	0.06	0.06	0.04	0.00	0.00	0.00	0.01	0.07	0.03	0.06	0.01
(9)	0.01	-0.01	0.04	0.01	0.02	0.03	0.03	0.03	-0.01	0.00	0.00	0.02	0.03	0.02	0.03	0.02
(10)	0.01	0.08	0.12	0.09	0.11	0.16	0.16	0.15	0.08	0.10	0.01	0.06	0.16	0.18	0.18	0.14
(11)	0.01	0.02	0.13	0.14	0.11	0.11	0.12	0.15	0.02	0.09	0.01	0.09	0.24	0.18	0.16	0.14
(12)	0.01	0.00	0.28	0.29	0.26	0.22	0.23	0.25	0.00	0.07	0.11	0.08	0.32	0.17	0.26	0.53
(13)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(14)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(15)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
(16)	0.28	0.33	0.36	0.04	0.64	0.04	0.04	0.05	0.33	0.64	0.64	0.30	0.04	0.04	0.04	0.03
(17)	0.03	0.17	0.52	1.00	0.52	0.52	0.52	0.52	0.17	0.52	0.52	0.52	1.00	0.64	0.52	0.88
(18)	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	0.56
(19)	-0.15	0.38	0.36	0.56	0.33	0.50	0.50	0.66	0.38	-0.04	0.31	0.00	0.50	0.23	0.50	0.00
(20)	0.00	0.00	0.57	0.95	0.52	0.52	0.52	0.52	0.00	0.07	0.57	0.52	0.95	0.69	0.52	0.65
(21)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(22)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(23)	0.00	0.03	0.00	0.12	0.12	0.12	0.12	0.12	0.03	0.12	0.00	0.00	0.12	0.12	0.12	0.07
(24)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00
Sum	0.02	0.04	0.22	0.21	0.17	0.22	0.23	0.21	0.04	0.10	0.15	0.12	0.24	0.18	0.22	0.16

(1) Day/Night Mode
(2) Power Consumption
(3) Valid Lane Markings
(4) HVAC Driver
(5) HVAC Co-Driver
(6) Proactive ACC
(7) Proactive ACC Gap
(8) Proactive Window Driver
(9) Proactive Window Co-Driver
(10) Proactive Seat Heating Driver
(11) Proactive Seat Heating Co-Driver
(12) Proactive DEC
(13) Frontend Collision Warning
(14) Cross Traffic Alert
(15) Lane Departure Warning
(16) Lane Departure Sensitivity
(17) Lane Departure Intervention
(18) Lane Change Warning
(19) Lane Change Sensitivity
(20) Lane Change Intervention
(21) Side Collision Warning
(22) Speed Limit Assist
(23) Speed Limit Assist Offset
(24) Steering Wheel Vibration

Table 4.7: Signal subset evaluation results for $mcc_{\mathcal{A}}$ of the RF classifier - Values highlighted where $mcc_{\mathcal{A}} \geq mcc_p$

Test	CIFE	CMIM	Chi^2	DISR	FCBF	FSc.	Fisher	Gini I.	ICAP	JMI	MIFS	MIM	MRMR	ReliefF	T Rat.	-
(1)	0.10	0.19	0.70	0.70	0.66	0.75	0.74	0.76	0.19	0.75	0.15	0.74	0.75	0.70	0.73	0.63
(2)	0.02	0.05	0.08	0.25	0.03	0.15	0.16	0.35	0.06	0.24	0.00	0.12	0.21	0.03	0.16	0.13
(3)	0.00	0.00	0.01	0.03	0.00	0.03	0.03	0.03	0.00	0.01	0.00	0.01	0.00	0.00	0.02	0.01
(4)	0.11	0.10	0.07	0.07	0.08	0.09	0.07	0.11	0.10	0.12	0.10	0.09	0.12	0.03	0.11	0.08
(5)	0.08	0.12	0.08	0.10	0.11	0.09	0.07	0.08	0.12	0.14	0.06	0.10	0.12	0.05	0.06	0.08
(6)	0.11	0.16	0.50	0.49	0.41	0.50	0.50	0.50	0.16	0.16	0.10	0.24	0.49	0.45	0.50	0.50
(7)	0.07	0.11	0.33	0.33	0.31	0.32	0.31	0.31	0.11	0.11	0.06	0.13	0.36	0.17	0.30	0.31
(8)	0.00	0.01	0.03	0.01	0.00	0.02	0.02	0.04	0.01	0.01	0.00	0.02	0.01	0.00	0.03	0.02
(9)	0.01	-0.01	0.01	0.00	0.03	0.03	0.02	0.02	-0.01	0.01	0.00	0.01	0.01	0.02	0.03	0.02
(10)	0.09	0.11	0.13	0.16	0.10	0.20	0.19	0.20	0.11	0.17	0.00	0.15	0.13	0.15	0.21	0.18
(11)	0.11	0.12	0.08	0.14	0.11	0.14	0.12	0.15	0.12	0.12	-0.01	0.08	0.15	0.21	0.13	0.10
(12)	0.12	0.07	0.19	0.43	0.43	0.22	0.21	0.38	0.07	0.22	0.02	0.26	0.50	0.16	0.22	0.57
(13)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(14)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(15)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
(16)	0.27	0.43	0.37	0.02	0.64	0.04	0.04	0.04	0.43	0.63	0.33	0.04	0.04	0.04	0.04	0.29
(17)	0.13	0.08	0.88	0.52	0.52	0.52	0.52	0.51	0.08	0.52	0.45	0.52	0.52	0.52	0.52	0.49
(18)	0.00	0.78	0.00	1.00	1.00	1.00	1.00	1.00	0.78	1.00	0.00	1.00	1.00	1.00	1.00	0.25
(19)	0.51	0.61	0.65	0.59	0.22	0.50	0.50	0.16	0.61	0.47	0.67	0.40	0.45	0.19	0.50	0.18
(20)	0.12	0.08	0.88	0.52	0.52	0.52	0.52	0.51	0.08	0.07	0.00	0.07	0.52	0.52	0.52	0.49
(21)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(22)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(23)	0.01	0.11	0.12	0.12	0.04	0.12	0.12	0.12	0.11	0.12	0.04	0.11	0.12	0.12	0.12	0.04
(24)	0.37	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sum	0.06	0.09	0.19	0.23	0.18	0.22	0.21	0.26	0.09	0.19	0.05	0.17	0.24	0.16	0.21	0.21

(1) Day/Night Mode
(2) Power Consumption
(3) Valid Lane Markings
(4) HVAC Driver
(5) HVAC Co-Driver
(6) Proactive ACC
(7) Proactive ACC Gap
(8) Proactive Window Driver
(9) Proactive Window Co-Driver
(10) Proactive Seat Heating Driver
(11) Proactive Seat Heating Co-Driver
(12) Proactive DEC
(13) Frontend Collision Warning
(14) Cross Traffic Alert
(15) Lane Departure Warning
(16) Lane Departure Sensitivity
(17) Lane Departure Intervention
(18) Lane Change Warning
(19) Lane Change Sensitivity
(20) Lane Change Intervention
(21) Side Collision Warning
(22) Speed Limit Assist
(23) Speed Limit Assist Offset
(24) Steering Wheel Vibration

Table 4.8: Signal subset evaluation results for $mcc_{\Delta Ap}$ of the SVM classifier - Values highlighted where $mcc_{\Delta Ap} \geq 0$

Test	CIFE	CMIM	Chi^2	DISR	FCBF	FSc.	Fisher	Gini I.	ICAP	JMI	MIFS	MIM	MRMR	ReliefF	T Rat.
(1)	-0.43	-0.37	0.23	0.23	0.14	0.25	0.26	0.24	-0.37	0.14	0.17	0.21	0.23	0.23	0.25
(2)	0.00	0.00	0.16	0.05	0.01	0.16	0.16	0.11	0.00	0.00	0.11	0.00	0.17	0.03	0.15
(3)	0.00	0.00	0.03	0.02	0.01	0.02	0.03	0.02	0.00	0.00	0.00	0.00	0.02	0.01	0.02
(4)	-0.05	-0.03	0.02	0.01	0.00	0.05	0.06	0.04	-0.03	-0.03	-0.01	0.00	0.02	0.00	0.04
(5)	-0.05	-0.03	0.03	0.06	0.04	0.02	0.02	0.05	-0.03	-0.02	-0.04	-0.05	0.05	0.02	0.03
(6)	-0.39	-0.31	0.08	0.07	-0.07	0.05	0.06	0.03	-0.31	-0.33	-0.07	-0.21	0.08	0.04	0.05
(7)	-0.26	-0.23	0.04	0.09	0.01	0.08	0.07	0.06	-0.23	-0.23	-0.02	-0.18	0.09	-0.09	0.06
(8)	0.00	0.00	0.06	0.03	0.03	0.05	0.06	0.03	0.00	-0.01	0.00	0.00	0.07	0.03	0.06
(9)	-0.02	-0.03	0.01	-0.01	0.00	0.01	0.01	0.01	-0.03	-0.02	-0.02	0.00	0.00	0.00	0.01
(10)	-0.13	-0.06	-0.01	-0.05	-0.03	0.02	0.02	0.01	-0.06	-0.04	-0.13	-0.08	0.02	0.04	0.04
(11)	-0.13	-0.11	-0.01	0.01	-0.03	-0.02	-0.02	0.01	-0.11	-0.04	-0.13	-0.05	0.10	0.04	0.02
(12)	-0.52	-0.52	-0.25	-0.23	-0.26	-0.30	-0.29	-0.28	-0.52	-0.45	-0.42	-0.45	-0.21	-0.35	-0.26
(13)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(14)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(15)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
(16)	0.25	0.30	0.33	0.01	0.60	0.01	0.01	0.02	0.30	0.60	0.60	0.27	0.01	0.01	0.01
(17)	-0.85	-0.71	-0.36	0.12	-0.36	-0.36	-0.36	-0.36	-0.71	-0.36	-0.36	-0.36	0.12	-0.24	-0.36
(18)	-0.56	-0.56	0.44	0.44	0.44	0.44	0.44	0.44	-0.56	0.44	0.44	0.44	0.44	0.44	0.44
(19)	-0.15	0.38	0.36	0.56	0.33	0.50	0.50	0.66	0.38	-0.04	0.31	0.00	0.50	0.23	0.50
(20)	-0.65	-0.65	-0.09	0.30	-0.14	-0.13	-0.13	-0.13	-0.65	-0.59	-0.09	-0.13	0.30	0.03	-0.13
(21)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(22)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(23)	-0.07	-0.05	-0.08	0.05	0.05	0.05	0.05	0.05	-0.05	0.05	-0.07	-0.07	0.05	0.05	0.05
(24)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00
Sum	-0.14	-0.12	0.06	0.04	0.01	0.06	0.06	0.05	-0.12	-0.06	-0.01	-0.04	0.08	0.02	0.06

(1) Day/Night Mode
(2) Power Consumption
(3) Valid Lane Markings
(4) HVAC Driver
(5) HVAC Co-Driver
(6) Proactive ACC
(7) Proactive ACC Gap
(8) Proactive Window Driver
(9) Proactive Window Co-Driver
(10) Proactive Seat Heating Driver
(11) Proactive Seat Heating Co-Driver
(12) Proactive DEC
(13) Frontend Collision Warning
(14) Cross Traffic Alert
(15) Lane Departure Warning
(16) Lane Departure Sensitivity
(17) Lane Departure Intervention
(18) Lane Change Warning
(19) Lane Change Sensitivity
(20) Lane Change Intervention
(21) Side Collision Warning
(22) Speed Limit Assist
(23) Speed Limit Assist Offset
(24) Steering Wheel Vibration

Table 4.9: Signal subset evaluation results for $mcc_{\Delta Ap}$ of the RF classifier - Values highlighted where $mcc_{\Delta Ap} \geq 0$

Test	CIFE	CMIM	Chi^2	DISR	FCBF	FSc.	Fisher	Gini I.	ICAP	JMI	MIFS	MIM	MRMR	ReliefF	T Rat.
(1)	-0.53	-0.44	0.07	0.07	0.03	0.11	0.11	0.13	-0.44	0.11	-0.48	0.11	0.12	0.07	0.10
(2)	-0.11	-0.08	-0.05	0.12	-0.10	0.02	0.02	0.22	-0.08	0.11	-0.13	-0.01	0.08	-0.10	0.03
(3)	-0.01	0.00	0.01	0.03	-0.01	0.02	0.02	0.03	0.00	0.00	-0.01	0.00	0.00	-0.01	0.01
(4)	0.03	0.03	0.00	-0.01	0.00	0.01	0.00	0.03	0.03	0.04	0.02	0.02	0.04	-0.04	0.03
(5)	-0.01	0.04	-0.01	0.02	0.03	0.00	-0.02	0.00	0.04	0.06	-0.02	0.02	0.04	-0.03	-0.03
(6)	-0.39	-0.34	0.00	-0.01	-0.09	0.00	0.00	0.00	-0.34	-0.34	-0.39	-0.26	-0.01	-0.05	0.00
(7)	-0.24	-0.20	0.02	0.02	0.01	0.01	0.01	0.00	-0.20	-0.20	-0.25	-0.18	0.05	-0.14	-0.01
(8)	-0.02	-0.01	0.01	-0.01	-0.02	0.00	0.00	0.02	-0.01	-0.01	-0.02	0.00	-0.01	-0.03	0.01
(9)	-0.01	-0.03	-0.01	-0.02	0.01	0.00	0.00	0.00	-0.03	-0.01	-0.02	-0.01	-0.01	-0.01	0.01
(10)	-0.09	-0.07	-0.05	-0.02	-0.08	0.02	0.01	0.02	-0.07	-0.01	-0.18	-0.02	-0.05	-0.03	0.03
(11)	0.01	0.02	-0.02	0.04	0.01	0.04	0.02	0.05	0.02	0.02	-0.11	-0.02	0.06	0.11	0.03
(12)	-0.45	-0.50	-0.38	-0.15	-0.14	-0.35	-0.36	-0.19	-0.50	-0.35	-0.55	-0.31	-0.08	-0.41	-0.35
(13)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(14)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(15)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
(16)	-0.01	0.15	0.09	-0.27	0.35	-0.25	-0.25	-0.25	0.15	0.34	0.05	-0.25	-0.24	-0.25	-0.25
(17)	-0.36	-0.41	0.39	0.03	0.03	0.03	0.03	0.02	-0.41	0.03	-0.04	0.03	0.03	0.03	0.03
(18)	-0.25	0.53	-0.25	0.75	0.75	0.75	0.75	0.75	0.53	0.75	-0.25	0.75	0.75	0.75	0.75
(19)	0.33	0.43	0.47	0.41	0.04	0.32	0.32	-0.02	0.43	0.29	0.48	0.22	0.27	0.01	0.32
(20)	-0.37	-0.41	0.39	0.03	0.03	0.03	0.03	0.02	-0.41	-0.42	-0.49	-0.42	0.03	0.03	0.03
(21)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(22)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(23)	-0.03	0.07	0.08	0.08	0.00	0.08	0.08	0.08	0.07	0.08	0.00	0.07	0.08	0.08	0.08
(24)	0.37	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00
Sum	-0.15	-0.12	-0.01	0.02	-0.03	0.01	0.00	0.05	-0.12	-0.02	-0.16	-0.04	0.03	-0.05	0.01

(1) Day/Night Mode
(2) Power Consumption
(3) Valid Lane Markings
(4) HVAC Driver
(5) HVAC Co-Driver
(6) Proactive ACC
(7) Proactive ACC Gap
(8) Proactive Window Driver
(9) Proactive Window Co-Driver
(10) Proactive Seat Heating Driver
(11) Proactive Seat Heating Co-Driver
(12) Proactive DEC
(13) Frontend Collision Warning
(14) Cross Traffic Alert
(15) Lane Departure Warning
(16) Lane Departure Sensitivity
(17) Lane Departure Intervention
(18) Lane Change Warning
(19) Lane Change Sensitivity
(20) Lane Change Intervention
(21) Side Collision Warning
(22) Speed Limit Assist
(23) Speed Limit Assist Offset
(24) Steering Wheel Vibration

Table 4.10: Signal subset evaluation results for the number of runs k for the SVM classifier, where $mcc_{\Delta Ap} \geq 0$ - Values in the upper quartile per row highlighted

Test	CIFE	CMIM	Chi^2	DISR	FCBF	FSc.	Fisher	Gini I.	ICAP	JMI	MIFS	MIM	MRMR	ReliefF	T Rat.
(1)	23	24	62	61	55	66	66	66	24	59	60	62	60	60	66
(2)	96	96	92	95	83	94	94	95	96	96	96	96	95	94	93
(3)	82	81	79	79	77	79	79	77	81	82	82	82	77	82	78
(4)	28	27	32	33	25	36	37	33	27	22	33	24	31	31	35
(5)	31	24	35	35	32	36	36	34	24	30	29	29	35	31	36
(6)	21	22	33	34	23	33	35	32	22	21	31	27	36	32	32
(7)	21	21	30	29	29	33	34	28	21	21	29	22	27	24	31
(8)	72	70	63	64	61	66	66	63	70	70	71	72	62	61	67
(9)	40	39	37	36	36	36	36	37	39	40	39	40	36	38	36
(10)	21	25	19	18	25	28	27	27	25	27	20	21	26	25	28
(11)	16	13	16	16	10	14	15	15	13	17	15	17	18	15	15
(12)	10	10	14	12	14	10	10	12	10	10	12	11	13	12	11
(13)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(14)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(15)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
(16)	3	2	3	3	3	3	3	3	2	3	3	3	3	3	3
(17)	0	0	1	2	1	1	1	1	0	1	1	1	2	1	1
(18)	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1
(19)	2	3	3	3	3	3	3	3	3	2	3	2	3	3	3
(20)	0	0	1	2	1	1	1	1	0	0	1	1	2	1	1
(21)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(22)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(23)	7	6	6	8	8	8	8	8	6	8	7	7	8	8	8
(24)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Sum	475	465	529	533	489	550	554	538	465	512	535	520	537	524	547

(1) Day/Night Mode
(2) Power Consumption
(3) Valid Lane Markings
(4) HVAC Driver
(5) HVAC Co-Driver
(6) Proactive ACC
(7) Proactive ACC Gap
(8) Proactive Window Driver
(9) Proactive Window Co-Driver
(10) Proactive Seat Heating Driver
(11) Proactive Seat Heating Co-Driver
(12) Proactive DEC
(13) Frontend Collision Warning
(14) Cross Traffic Alert
(15) Lane Departure Warning
(16) Lane Departure Sensitivity
(17) Lane Departure Intervention
(18) Lane Change Warning
(19) Lane Change Sensitivity
(20) Lane Change Intervention
(21) Side Collision Warning
(22) Speed Limit Assist
(23) Speed Limit Assist Offset
(24) Steering Wheel Vibration

Table 4.11: Signal subset evaluation results for the number of runs k for the RF classifier, where $mcc_{\Delta Ap} \geq 0$ - Values in the upper quartile per row highlighted

Test	CIFE	CMIM	Chi^2	DISR	FCBF	FSc.	Fisher	Gini I.	ICAP	JMI	MIFS	MIM	MRMR	ReliefF	T Rat.
(1)	17	19	45	50	46	61	61	63	20	59	22	59	60	48	60
(2)	46	49	58	88	43	77	77	94	50	85	44	65	89	48	75
(3)	81	81	81	81	76	80	79	82	81	81	81	81	81	81	78
(4)	33	35	31	31	28	35	33	33	35	32	35	30	35	25	35
(5)	30	32	34	37	31	36	33	35	32	35	36	36	36	29	29
(6)	18	21	29	26	24	28	30	28	21	20	22	23	29	28	28
(7)	21	22	31	30	31	31	31	23	22	21	23	23	30	23	30
(8)	66	65	71	65	66	65	67	68	65	65	68	67	64	59	64
(9)	38	38	38	39	39	38	38	39	38	40	39	40	40	39	39
(10)	21	23	21	21	21	26	26	25	23	23	20	22	22	19	29
(11)	16	19	14	17	12	17	16	17	19	17	16	16	15	17	16
(12)	9	9	11	12	13	9	8	10	9	10	9	8	13	8	9
(13)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(14)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(15)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
(16)	1	3	3	1	3	2	2	2	3	3	2	2	2	2	2
(17)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
(18)	0	1	0	1	1	1	1	1	1	1	0	1	1	1	1
(19)	3	3	3	3	3	2	2	2	3	2	3	2	2	3	2
(20)	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
(21)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(22)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(23)	6	8	8	8	8	8	8	8	8	8	8	8	8	8	8
(24)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Sum	409	432	482	514	449	520	516	534	434	506	432	487	531	442	509

(1) Day/Night Mode
(2) Power Consumption
(3) Valid Lane Markings
(4) HVAC Driver
(5) HVAC Co-Driver
(6) Proactive ACC
(7) Proactive ACC Gap
(8) Proactive Window Driver
(9) Proactive Window Co-Driver
(10) Proactive Seat Heating Driver
(11) Proactive Seat Heating Co-Driver
(12) Proactive DEC
(13) Frontend Collision Warning
(14) Cross Traffic Alert
(15) Lane Departure Warning
(16) Lane Departure Sensitivity
(17) Lane Departure Intervention
(18) Lane Change Warning
(19) Lane Change Sensitivity
(20) Lane Change Intervention
(21) Side Collision Warning
(22) Speed Limit Assist
(23) Speed Limit Assist Offset
(24) Steering Wheel Vibration

Table 4.12: Signal subset evaluation results for median run-time of feature selection algorithms per use case - Values below 1.0m highlighted

Test	CIFE	CMIM	Chi^2	DISR	FCBF	FSc.	Fisher	Gini I.	ICAP	JMI	MIFS	MIM	MRMR	ReliefF	T Rat.
(1)	2.6 h	3.9 h	0.2 s	7.8 h	12.0 m	0.2 s	8.0 s	1.3 m	3.9 h	4.1 h	3.8 h	4.1 h	3.9 h	2.1 m	11.3 s
(2)	1.0 h	1.6 h	0.1 s	3.6 h	5.2 m	0.1 s	2.3 s	0.9 m	1.6 h	1.7 h	1.6 h	1.7 h	1.6 h	0.8 m	4.6 s
(3)	1.5 h	2.3 h	0.1 s	4.6 h	6.9 m	0.1 s	4.2 s	22.3 m	2.3 h	2.4 h	2.2 h	2.4 h	2.2 h	0.7 m	6.5 s
(4)	2.7 h	4.3 h	0.2 s	10.1 h	19.4 m	0.3 s	9.4 s	1.7 m	5.0 h	4.4 h	4.9 h	4.6 h	4.7 h	4.6 m	16.0 s
(5)	2.7 h	4.0 h	0.2 s	8.1 h	16.0 m	0.3 s	10.7 s	1.6 m	4.1 h	4.2 h	4.1 h	4.2 h	4.1 h	3.5 m	15.0 s
(6)	2.1 h	3.1 h	0.2 s	6.7 h	11.8 m	0.2 s	6.8 s	0.9 m	3.2 h	3.3 h	3.1 h	3.2 h	3.0 h	1.1 m	8.5 s
(7)	1.8 h	3.0 h	0.2 s	6.4 h	9.8 m	0.2 s	5.3 s	1.0 m	3.0 h	3.2 h	2.9 h	3.3 h	2.9 h	1.7 m	8.7 s
(8)	0.7 h	1.0 h	0.1 s	2.3 h	3.8 m	0.1 s	1.4 s	0.3 m	1.1 h	1.1 h	1.0 h	1.1 h	1.1 h	0.2 m	3.1 s
(9)	1.0 h	1.4 h	0.1 s	3.1 h	4.3 m	0.1 s	3.3 s	0.5 m	1.4 h	1.6 h	1.5 h	1.6 h	1.5 h	0.5 m	4.7 s
(10)	3.4 h	5.2 h	0.3 s	10.4 h	22.5 m	0.3 s	17.2 s	2.0 m	5.1 h	5.5 h	5.2 h	5.4 h	5.0 h	4.7 m	21.2 s
(11)	4.7 h	7.3 h	0.5 s	15.0 h	24.9 m	0.3 s	36.5 s	3.6 m	7.4 h	7.6 h	7.3 h	7.5 h	7.0 h	8.8 m	40.9 s
(12)	3.1 h	4.4 h	0.2 s	9.2 h	19.3 m	0.3 s	10.2 s	1.5 m	4.5 h	4.6 h	4.6 h	4.6 h	4.4 h	4.0 m	15.0 s
(13)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(14)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(15)	7.3 h	12.0 h	0.4 s	24.4 h	30.0 m	0.7 s	78.0 s	5.9 m	11.7 h	12.1 h	11.7 h	12.1 h	11.6 h	18.4 m	102.0 s
(16)	5.4 h	8.6 h	0.6 s	20.2 h	22.5 m	0.4 s	46.0 s	4.8 m	8.5 h	8.8 h	8.5 h	8.6 h	8.5 h	16.0 m	66.0 s
(17)	4.3 h	7.0 h	0.4 s	14.1 h	18.7 m	0.3 s	26.3 s	2.8 m	7.1 h	7.3 h	7.0 h	7.3 h	7.2 h	7.8 m	41.8 s
(18)	8.3 h	14.1 h	0.4 s	25.5 h	35.8 m	0.8 s	72.0 s	5.0 m	13.7 h	14.3 h	13.7 h	14.2 h	14.1 h	18.0 m	96.0 s
(19)	2.4 h	4.6 h	0.2 s	10.4 h	11.6 m	0.4 s	13.5 s	1.3 m	5.0 h	5.2 h	4.9 h	5.1 h	4.7 h	4.2 m	17.5 s
(20)	4.5 h	6.8 h	0.3 s	14.3 h	19.6 m	0.3 s	32.4 s	2.8 m	6.9 h	7.1 h	6.9 h	7.0 h	6.9 h	8.9 m	46.7 s
(21)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(22)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
(23)	1.7 h	2.5 h	0.1 s	5.5 h	8.1 m	0.1 s	5.9 s	0.7 m	2.6 h	2.6 h	2.5 h	2.7 h	2.5 h	1.2 m	8.4 s
(24)	0.8 h	1.2 h	0.1 s	2.8 h	2.8 m	0.1 s	2.0 s	0.4 m	1.2 h	1.2 h	1.2 h	1.2 h	1.2 h	0.4 m	4.0 s
Median	1.7 h	2.6 h	0.1 s	5.2 h	9.6 m	0.2 s	5.1 s	1.4 m	2.7 h	2.8 h	2.6 h	2.8 h	2.6 h	0.9 m	8.1 s

(1) Day/Night Mode
(2) Power Consumption
(3) Valid Lane Markings
(4) HVAC Driver
(5) HVAC Co-Driver
(6) Proactive ACC
(7) Proactive ACC Gap
(8) Proactive Window Driver
(9) Proactive Window Co-Driver
(10) Proactive Seat Heating Driver
(11) Proactive Seat Heating Co-Driver
(12) Proactive DEC
(13) Frontend Collision Warning
(14) Cross Traffic Alert
(15) Lane Departure Warning
(16) Lane Departure Sensitivity
(17) Lane Departure Intervention
(18) Lane Change Warning
(19) Lane Change Sensitivity
(20) Lane Change Intervention
(21) Side Collision Warning
(22) Speed Limit Assist
(23) Speed Limit Assist Offset
(24) Steering Wheel Vibration

4.6 Deployment Evaluation

In the last two evaluations, we only evaluated the approach for each user separately. In this section, we will evaluate the proposed deployment strategy and how users differ in each test case. For this comparison, we compare the selected signal subset on different data sets within the same test case.

4.6.1 Setup

To compare the deployment evaluation, we compare the selected signal subsets $S_{\mathcal{A}}$ of the different users for each test case. The selection itself is based on the selection step in the last evaluation. We denoted the signals which have been selected by each algorithm as $S_{\mathcal{A}}$, where $S_{\mathcal{A}} \subseteq S_{pre}$. In this evaluation, we do not use the selected signal subset $S_{\mathcal{A}}$ to train a classifier. We directly compare these signal subsets between each user (i. e. data set). By this, we will see if our assumption of three different types of signal subsets will hold (cf. Section 3.7). To limit the number of comparisons, we will only use the selected signal subset by the feature selection algorithm Fisher Score. This algorithm performed very well in the last evaluation, and therefore the signals selected by this algorithm should be the most descriptive. For the other algorithms which also performed well (Chi^2, DISR, FScore, Gini Index, and MRMR), we depict the results in Appendix D, but will not further discuss these results as they are similar.

4.6.2 Metric

We compare each selected signal subset by using the *Jaccard distance* [180]. This metric shows the difference between two different sets S_i and S_j and is defined as:

$$J_{ij} \stackrel{\text{def}}{=} J(S_i, S_j) = \frac{|S_i \cup S_j| - |S_i \cap S_j|}{|S_i \cup S_j|}, (S_i \cup S_j) \neq \emptyset \tag{4.9}$$

A value of $J_{ij} = 1$ states that the sets are entirely different to each other and a value of $J_{ij} = 0$ shows that the sets are identical. With this metric, we can evaluate if the selected signal subset of a user set fits for a different user and if the signal subset types introduced before hold.

In the following, we will briefly recap each signal subset type: The first signal subset type we introduced (type I) includes all signal subsets S_i/S_j, which hold true for *all* users or *all* systems k and contain similar signals (cf. Eq. (3.6)):

$$S_{\mathcal{A}}^{I} = \{S_i \mid \forall i, j \in \{1, \ldots, k\}, S_i = S_j\}$$

For example, the steering torque of the steering wheel is only related to the physics of the driving dynamics and not to any user behaviour. These correlations are similar among all users k and lead to the same selected signal subset.

The second type (type II) includes all signal subsets S_i/S_j which hold true for a *group of*

users and are at least similar for two users (cf. Eq. (3.7)):

$$S_{\mathcal{A}}^{II} = \{S_i \mid \exists i, j \in \{1, \ldots, k\}, i \neq j \wedge S_i = S_j\}$$

For example, a group of users show a correlation between the driving dynamics control (i. e. *sport*, *comfort*, or *eco pro*) and whether a co-driver is present or not. This kind of signal subsets are similar for a particular group of users, but not all users.

The third type (type III) includes all signal subsets S_i/S_j which hold true for only *one specific user* and are unique for every user (cf. Eq. (3.8)):

$$S_{\mathcal{A}}^{III} = \{S_i \mid \forall i, j \in \{1, \ldots, k\}, i \neq j \wedge S_i \neq S_j\}$$

These correlations are very user-specific and at the core of user-specific personalisation. An example is parallel usage of several functions at the start-up of the vehicle, i. e. which functions are triggered when the user is commuting from home to work.

To evaluate which type of subset we observe for each test case, we calculate the J_{ij} for the top $|S_{\mathcal{A}}| = 10$ signals selected by the signal selection. These values are then visualised using heat maps for each test case.

4.6.3 Results and Discussion

The results of this evaluation are shown in heat maps for each test case in Figure 4.3. As test cases, we only use those where we have more than ten valid data sets (i. e. users). In our case, this excludes all test cases of the type anomaly detection. Each heat map shows on the y-axis the index i of the used data set for the selection of S_i and on the x-axis, the index j of the used data set used for the selection of S_j. The number i and j correspond to the numbers of the data sets introduced in Section 4.3. The colour of each box represents the Jaccard distance J_{ij} of the signal subsets S_i and S_j. The diagonal value in the heat map must always be equal to $J_{ij} = 0$, because we compare the subsets of the same user (i. e. $S_j = S_i$).

In all figures, we see different patterns which are, in some cases, similar to the results seen in another test case. The first three test cases (cf. Figures 4.3a to 4.3c) show low values for J_{ij} for the most data sets with lower indexes (i. e. the upper left part of the heat map). When comparing data sets with higher indexes, we see higher values up to $J_{ij} = 1$. In Figures 4.3f and 4.3g, we see a similar distribution of the values, but with harder edges between the values. These high values for the data set with high indexes result from the enumeration of the data sets. When introducing the data sets in Section 4.3, we sorted all data sets based on their length in descending order. Meaning data sets with a low index are longer than data sets with a high index. Short data sets (here up to $0.1h$) lead to signal subsets which are different to all other data sets. This shows that the user's behaviour is either different or an inappropriate subset has been selected. In the first three test cases we can see low values for the data sets with lower indexes (i. e. longer data sets in the upper left part). This shows that an inappropriate subset

has been selected due to the short length of the data set. This demonstrates the need for long enough data sets to identify the correct signals.

When not including the data sets with high indexes, we see relatively similar values across all data sets in the case of *Day/Night Mode*, *Power Consumption*, and *Valid Lane Markings*. These functions only correlate to the system's behaviour and do not correlate to the user's behaviour, and therefore, all subsets should be similar to each other. But here we see values $J_{ij} > 0$ for most of the cases, which shows differences between the sets. But at the same time, almost all values are $J_{ij} < 1$, showing that at least one signal in the subsets is identical. Depending on the function, which is the basis for the test case, we either only have one or a few signals the function is correlating to. Since we compare the top $|S_i| = |S_j| = 10$ ranked signals, this leads to values $J_{ij} > 0$ but also to values $J_{ij} < 1$, if less than 10 signals are correlating. In summary, the results for the first three test cases support our assumption of type I for these test cases.

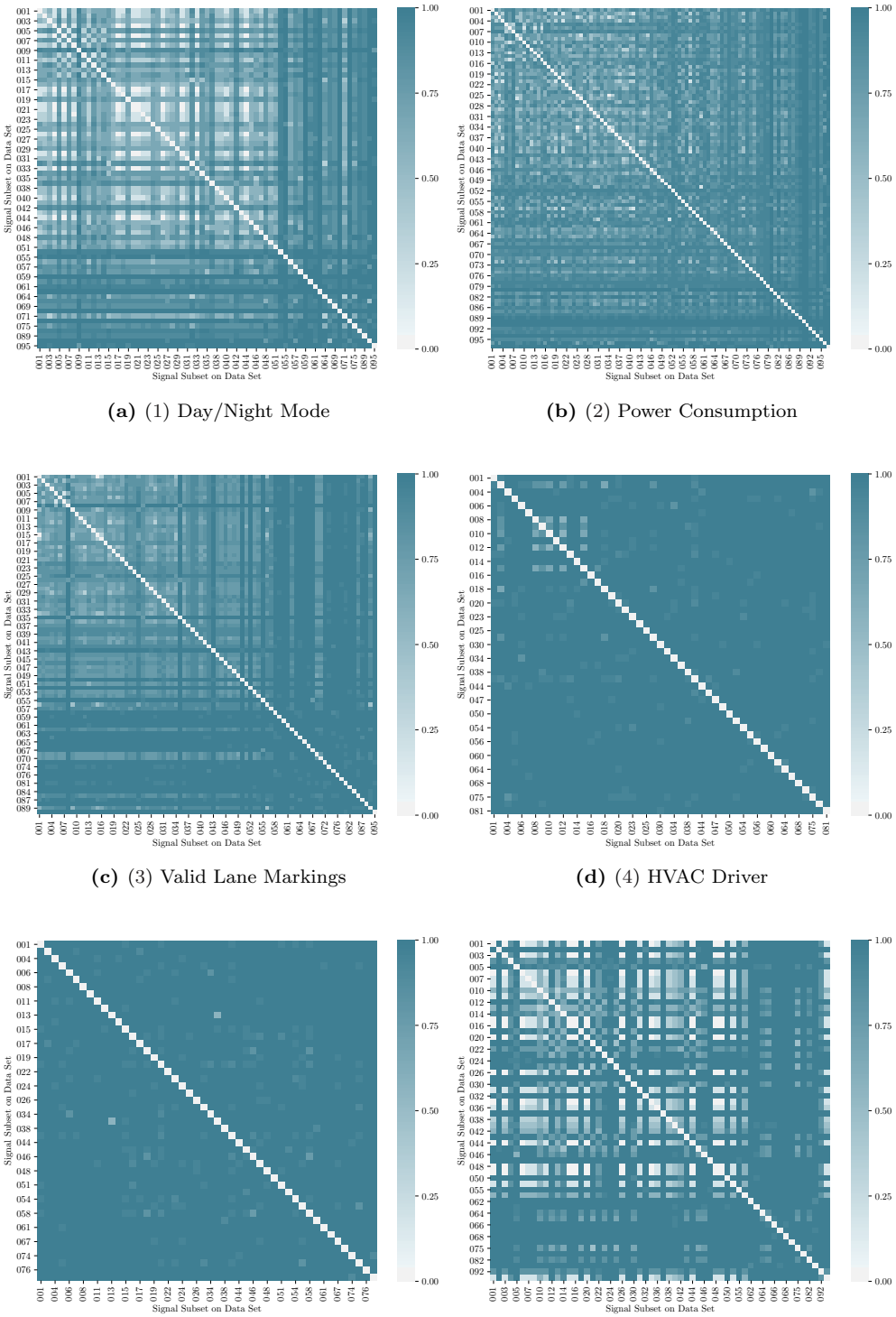
In contrast to this, the test cases *Proactive ACC* and *Proactive ACC Gap* also show low values for the lower indexed data sets but also include some with high values up to $J_{ij} = 1$ in between. The basis for these two test cases is the ACC system, which is highly dependent on the user's preferences and behaviour. We can see that a *group* of users show a similar correlation to a set of signals. This supports our assumption of type II, where the signal subsets of a group of users are similar. Nevertheless, we can again see the influence of the length of the data sets and the need for a sufficient amount of data.

All other test cases show high values throughout all data sets. This indicates that the signal subsets are different for each user they were selected on. We also cannot see a clear border after which index the data sets were to short to select an appropriate signal subset. All these test cases show the demand for a highly personalised selection of proper input signals for a DDF, as we suggested by the subset type III. We can also see in the last evaluation, that the classifiers trained on these subsets achieved good performance, even if the subset are highly diverse.

In summary, we showed that we observed all three types of signal subsets $S_{\mathcal{A}}^{I}$, $S_{\mathcal{A}}^{II}$, and $S_{\mathcal{A}}^{III}$, and that, depending on the test case, there is a need for a highly personalised selection of appropriate input signals for a DDF. At the same time, we were also able to see that a minimum amount of data from each user is required to select the appropriate subset and that this varies from test case to test case.

4.6.4 Threats to the Validity

Besides the threats to the validity mentioned for the previous evaluation, the selection of the Jaccard distance as a metric and the used visualisation for this evaluation might affect the validity of the results. However, we purposely used this metric as we do not want to compare the actual performance of the machine learning algorithms on different data sets, but the difference between selected signals. By the visualisation in heat maps, the similarity of the different selected signals throughout the data sets can be easily visualised.



(a) (1) Day/Night Mode

(b) (2) Power Consumption

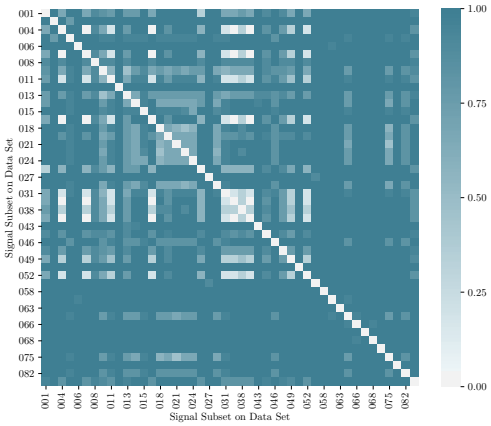
(c) (3) Valid Lane Markings

(d) (4) HVAC Driver

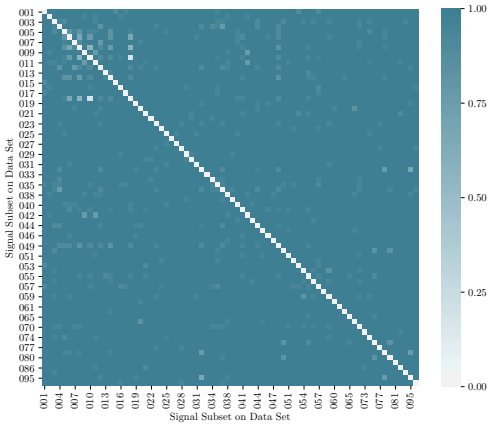
(e) (5) HVAC Co-Driver

(f) (6) Proactive ACC

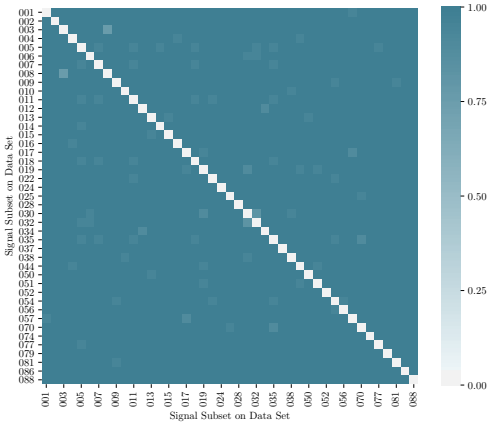
Figure 4.3: Results of deployment evaluation (Fisher Score) *(cont.)*



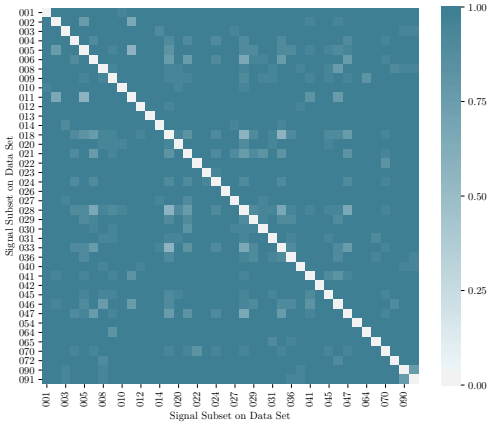
(g) (7) Proactive ACC Gap



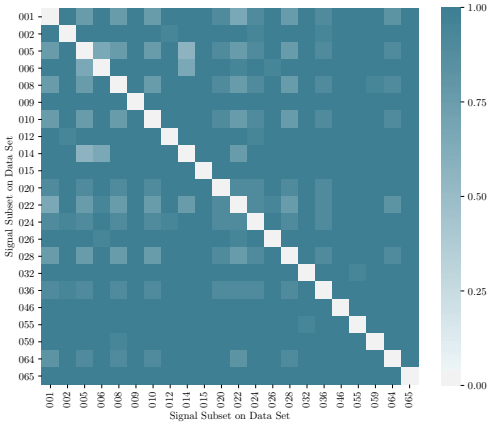
(h) (8) Proactive Window Driver



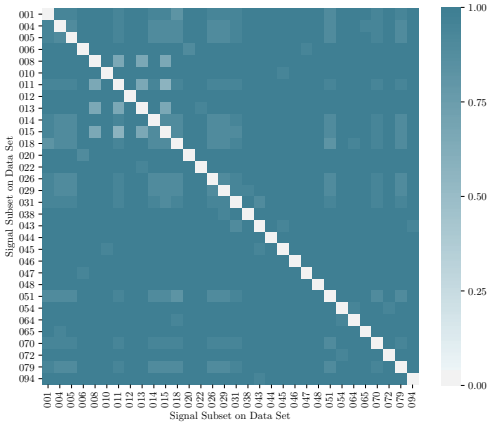
(i) (9) Proactive Window Co-Driver



(j) (10) Proactive Seat Heating Driver



(k) (11) Proactive Seat Heating Co-Driver



(l) (12) Proactive DEC

Figure 4.3: Results of deployment evaluation (Fisher Score)

Chapter 5

Streaming Evaluation

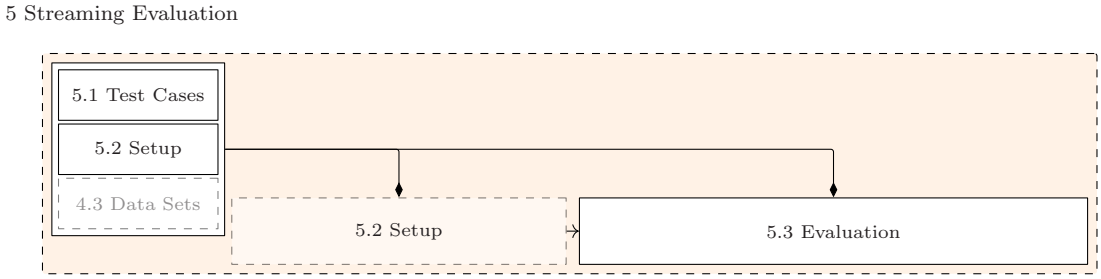


Figure 5.1: Chapter structure

One of the main challenges of E/E architectures is the vast amount of vehicle data and the limited resources for storing or transferring this data (cf. Challenge 6). The idea of this approach is use the capability of processing streamed data without storing a significant amount of data. In this chapter, we evaluate the approach on streaming data, meaning no vehicle data has to be stored within the vehicle or the back end.[1] We will use the data sets and the test cases of the type anomaly detection, both presented in the last chapter. An overview of this chapter's structure can be found in Figure 5.1.

5.1 Test Cases

For these test cases, we consider the deactivation of ADAS functions such as side collision warning, lane change warning, and cross-traffic alert. A driver can deactivate such functions manually for personalisation. A deactivation per se is not a problem if intended by the driver: for instance, a driver might deactivate traction control when being stuck in an iced parking

[1] The streaming evaluation and its implementation including a preliminary evaluation using other data sets has been previously published at the 2019 ACM/IEEE ICSE (cf. [8]) and the 2019 IEEE IV (cf. [9]).

lot. However, suppose such functions have been deactivated due to a software or hardware fault, or even after an IT attack by an intruder. In that case, this is a severe problem which we try to identify and to mitigate. In addition to already applied methods at design time, we need to learn a personalised driver model that contains information about the situations (i. e. context) in which drivers intentionally deactivate functions. We refer to this as *nominal behaviour*. Here we will consider all presented ADAS functions from the previous chapter and implement a data-driven anomaly detection for all of these test cases.

Based on nominal behaviour, we consider several reasons for anomalies or deviations from this behaviour. Gleirscher and Kugele [181] presented different reasons for a function deactivation and possible actions as countermeasures taken by the involved parties *vehicle*, *driver*, and *OEM*. For all areas of deactivation, i. e. the vehicle itself (e. g. software and hardware faults), the driver (e. g. intentionally or by maloperation), or the environment (e. g. IT attack, intruder) we can take countermeasures. The concept is to inform the driver about a potential anomaly, who then can take the appropriate countermeasures. In case of a detected anomaly, the driver is informed about the potential anomaly by showing a Check Control (CC) message. Thus, the driver is informed about the *anomalous behaviour* and can confirm it as intended or as not intended, yielding a more precise personalised driver model. By receiving this notification, the driver can then mitigate the potentially hazardous driving situation by stopping the car and bringing it into a safe state.

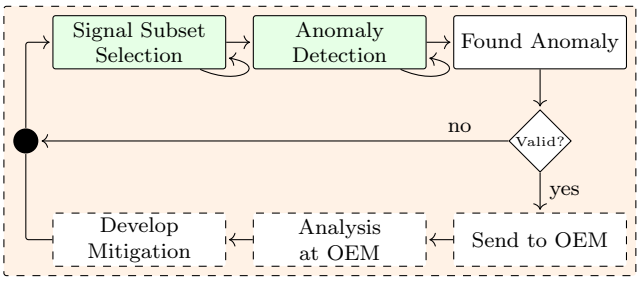


Figure 5.2: General concept of the anomaly detection test case*

*Figure based on previously published figure at the 2019 ACM/IEEE ICSE (cf. [8]) and the 2019 IEEE IV (cf. [9]) ©IEEE 2019

The basic concept of the proposed anomaly detection function (i. e. a DDF for anomaly detection) is depicted in Figure 5.2. In the first step, the approach presented in this work preprocesses all vehicle signals and identifies the most descriptive vehicle signals (i. e. the signal subset selection step). These signals are then used for the anomaly detection step. For each of the signals, the nominal behaviour is learned by the DDF. These both steps are executed continuously (cf. green boxes).

If the current state of a vehicle's signal deviates from its nominal behaviour, a potential anomaly is detected for these particular signals. If a certain number of signals above a threshold show potential anomalies an anomaly is detected and the driver is informed. The correctness of each detection must be validated by the driver, by presenting it as a CC message on the

main vehicle display with a warning explaining the target function and its possible abnormal behaviour. If the detected anomaly gets validated by the driver, then it is sent to the OEM for further analysis and investigation.

At the OEM, the anomaly is compared with existing anomalies that have been identified from other vehicles of the OEM's fleet. In the case of identical or similar anomalies, i. e. anomalies that relate to the same signal(s), we propose to cluster them. Based on this clustering, causes for the anomaly as well as impacts on the relationship between vehicle, passengers, and the environment need to be derived. Upon classification of anomalies, the impact assessment on the interrelation between vehicle, passengers, and the environment is supported by our approach. As a basic set of guidelines, we include methodologies and tools from the field of functional safety according to the ISO 26262 (cf. [182]) into our proceeding since any possible harm on humans that roots from an anomaly needs to be captured. ISO 26262 proposes to embed the assessment of potential harm on humans within the safety assurance process of a V-model approach. Here, anomalies that potentially result in human injury can be captured accordingly to their identification at run-time. The assessment at the OEM is not part of this evaluation (cf. dashed boxes).

5.2 Setup

In this section, we will introduce the setup used for the implementation and evaluation of the proposed DDFs. For the assessment of the presented streaming anomaly detection for ADAS functionalities, we have chosen an offline setup in order to reuse the already collected data sets. This offline setup does not differ to an onboard streaming implementation. Here, the collected data will be replayed as a stream, similar to an onboard scenario. We also have chosen this type of setup to have a controlled setup which we can rerun multiple times and which does not pose any harm to a test engineer. First, we will introduce briefly the DDF specification, followed by the used data sets and the signal pre-processing configuration. Then we will present the streaming feature selection algorithm and the used streaming anomaly detection algorithm.

5.2.1 Data-Driven Function Specification

For the specification of each anomaly detection test case, we reuse the specification for the anomaly detection test case out of the offline evaluations (cf. Listings A.13 to A.24 in Appendix A). These ADAS functions and specification have been discussed in detail in Section 4.1.3, and we will not reintroduce these functions or specifications. In total, we used twelve test cases for this evaluation.

5.2.2 Data Sets

As input data, we use the same data sets used in the last chapter (cf. Section 4.3). We have used vehicle data sets which were collected during a field study conducted at the BMW Group. These data sets originate from vehicle traces logged from current-generation BMW 7 Series (cf. [178]) driven by real customers. In contrast to the last evaluation, we do not split the data sets into a training and test set. The proposed DDF is learning continuously and can, therefore, be trained and evaluated continuously on the complete data set.

In this evaluation, the experimental setting requires an ADAS function that changed its state at least once in a data set. This is needed in order to have at least two classes for the signals subset selection and to be able to learn the nominal behaviour for the anomaly detection. If this were not the case, the signal subset selection would not select any signals as no change in the label was observed, and it cannot compare the signals to another state. An overview of the number of reconfigured ADAS functions per test case can be found in Table 5.1.

Table 5.1: Test cases and usage over data sets

Test Case	**Reconfigured**	**Not Reconfigured**
(13) Frontend Collision Warning	-	101
(14) Cross Traffic Alert	1	100
(15) Lane Departure Warning	5	96
(16) Lane Departure Sensitivity	6	95
(17) Lane Departure Intervention	7	94
(18) Lane Change Warning	4	97
(19) Lane Change Sensitivity	15	86
(20) Lane Change Intervention	6	95
(21) Side Collision Warning	-	101
(22) Speed Limit Assist	3	98
(23) Speed Limit Assist Offset	25	76
(24) Steering Wheel Vibration	19	82
Total	**91**	**1121**
		1212

As we can see, all ADAS functions, except the frontend collision warning and the side collision warning, have been reconfigured at least once. This leads to a total number of 91 runs where we can evaluate the DDF from 39 different data sets (i. e. users). In Table E.1 (cf. Appendix E), a more detailed version of the table is shown. In 16 out of all 101 data sets exactly one ADAS function has been reconfigured at least once, and in 23 data sets more than one ADAS function has been reconfigured. As we can see, these functions are rarely being reconfigured, but for at least 39 users out of 101 users, there is the need to at least reconfigure one of these functions.

5.2.3 Pre-Processing

For the pre-processing of the vehicle signals, we reuse the pre-processing specification of the offline evaluation for the pre-processing step (cf. Listing C.1 in Appendix C). In this case, we want to include all error-values, as we also want to capture potential error states of sensors and functions. Besides, we scale all signals in the range of $[0, 1]$, and we do not discretise any signals as we would like to contain as much information as possible. Also, the used algorithm, which is introduced later is able to process continuous and discrete values. Also, we do not specify any additional signals for the signal subset section. Here, we want to focus on the evaluation of the proposed approach and its capability of identifying the appropriate vehicle signals and not on the development of a special DDF.

5.2.4 Streaming Feature Selection Algorithm

In this streaming evaluation, we will only use one feature selection algorithm, as we already evaluated different algorithms in the last chapter. We have chosen as the basis for the evaluation the Fisher Score algorithm [128], which showed high performance and short execution times. We have modified this supervised feature selection algorithm to support streaming processing and federated execution (cf. Algorithm 1).

Algorithm 1: Statistics calculation for each signal s_a *

Data: Value of signal s_a denoted by $\overline{s_a}$ and class k of label $\overline{s_l}$
Result: Statistics matrices $\boldsymbol{\mu}$ and S
$o_{ak} \leftarrow 0;\ \forall 1 \leq a \leq m$ and $\forall 1 \leq k \leq K$;
while $(\overline{s_a} \neq \perp)$ **do**
  if $(o_{ak} = 0)$ **then**
    $\mu_{ak} \leftarrow \overline{s_a}$; $S_{ak} \leftarrow 0$; $o_{ak} \leftarrow 1$;
  else
    $\mu_{ak} \leftarrow \mu_{ak} \oplus (\overline{s_a} \ominus \mu_{ak}) \oslash o_{ak}$;
    $S_{ak} \leftarrow S_{ak}(\overline{s_a} \ominus \mu_{ak}) \otimes (\overline{s_a} \ominus \mu_{ak})$;
    $o_{ak} \leftarrow o_{ak} + 1$;

* This algorithm has been previously published at the 2019 IEEE IV (cf. [9])

In the first step, the statistical distribution of all car signals is calculated. Let s_a with $1 \leq a \leq m, m \in \mathbb{N}^+$ be a pre-processed signal out of S_{init}, where $m = |S_{init}|$. Moreover, let k, $1 \leq k \leq K, K \in \mathbb{N}^+$ be a class and K be the number of possible classes (i. e. values) of the label s_l. A class k represents the function's state which can have K possible states (e. g. the traction control can be switched on or off) and is identical to the value of the label information $\overline{s_l}$. With $\overline{s_a}$ we denote the current value of signal s_a. The fundamental idea of Algorithm 1 is to *update continuously* (as long as new signal observations are received, i. e. $\overline{s_a} \neq \perp$) two $m \times K$

matrices $\boldsymbol{\mu} = (\mu_{ak}) \in \mathbb{R}^{m \times K}$ (for the mean) and $S = (S_{ak}) \in \mathbb{R}^{m \times K}$ (interim value for the standard deviation). Hence, we can assess each signal s_a in each class k. With o_{ak} we denote the number of observations of signal s_a in class k. This calculation is based on the semi-numerical calculation of Knuth [183] and Welford [184]. With Equation (5.1) we can calculate the mean μ_a of signal s_a independent of the class k and with Equation (5.2) the standard deviation σ_{ak} for each signal s_a in class k.

$$\mu_a = \frac{\sum_{k=1}^{K} \mu_{ak} o_{ak}}{\sum_{k=1}^{K} o_{ak}}, \sum_{k=1}^{K} o_{ak} > 0 \tag{5.1}$$

$$\sigma_{ak} = \sqrt{S_{ak}/(o_{ak} - 1)} \tag{5.2}$$

For all $m = |S_{init}|$ signals that are accessible in the vehicles, we use the current value of the signal s_a and the generated class k of $\overline{s_l}$ as input for the algorithm. Since the calculation is independent for each signal, we do not have to consider the synchronisation of all signals, and there is no overhead in sending the data to a central point. This could lead to a suboptimal signal subset because we do not consider the relationship of signals to each other. However, we accept this drawback due to the raised challenges by the E/E architecture. Furthermore, in the calculation itself, there is only a single dependency to the last state of μ_{ak}, S_{ak}, and o_{ak} and no additional data of previous states needs to be stored. This step is repeatedly executed whenever a new observation of a signal s_a is received/generated.

Next, each signal is ranked by the supervised feature selection algorithm Fisher Score [128]:

$$score_a \stackrel{\text{def}}{=} fisherscore(s_a) = \frac{\sum_{k=1}^{K} o_{ak}(\mu_{ak} - \mu_a)^2}{\sum_{k=1}^{K} o_{ak}\sigma_{ak}^2} \tag{5.3}$$

This algorithm calculates $score_a$ for each signal s_a. This calculation is based on the mean μ_a and standard deviation σ_a of each signal s_a, and the mean μ_{ak}, standard deviation σ_{ak}, and number observations o_{ak} of each signal s_a in class k. All these values have already been calculated in the previous step (cf. Algorithm 1) and can be directly used to calculate the score of s_a (i. e. $score_a$). This score is then collected in a central place to rank the signal according to their correlation to the label and thereby calculate the rank $rank_a$. This is the only step which is conducted in a non distributed manner but will only lead to a minimal amount of communication due to the small number of parameters. Only the highest correlating (i. e. ranked) signals are included in the selected signal subset.

The main advantage of this scoring method is that it uses data that was calculated from data streams and can be applied for discrete and continuous data. Another advantage—which can also be considered a drawback—is the evaluation of each signal individually. While performing signal selection and evaluating each signal on its own, the correlation between signals cannot be considered in the collection, leading to a suboptimal signal subset. Referring to the challenges stated in Section 2.2, this drawback is accepted in favour of resolving the issues of the highly

distributed architecture and highly heterogeneous data. Hence, all signals can be evaluated in a distributed manner, and minimal overhead in communication is achieved.

5.2.5 Streaming Anomaly Detection

To detect anomalies, our approach is based on finding outliers for each of the previously selected top $n = |S_{\mathcal{A}}|$ signals $s_a \in S_{\mathcal{A}}$ based on the rank $rank_a$. An outlier is defined as "patterns in data that do not conform to a well-defined notion of normal behaviour" [46]. Anomaly detection is a process to find these outliers in data by comparing with some predefined patterns or rules. Based on Chandola et al. [46] there are three different types of anomalies: *Point anomaly*, which happens when a single instance of data is too far away from the rest, *contextual anomaly*, which is considered when a single point or a sequence of the data instance is considered as an anomaly in a specific context, and the *collective anomaly* which refers to a set of data collectively helping to find out a sequence of anomalies. For the anomaly detection in our approach, we are trying to detect *contextual anomalies*.

In order to detect these contextual anomalies we apply the Grubbs' test [185–187] for each signal $s_a \in S_{\mathcal{A}}$ (cf. Eq. (5.4)):

$$\frac{|\overline{s_a} - \mu_{ak}|}{\sigma_{ak}} = z_a > \frac{o_{ak}}{\sqrt{o_{ak}}} \sqrt{\frac{t^2_{\alpha/(2o_{ak}),o_{ak}-2}}{o_{ak} - 2 + t^2_{\alpha/(2o_{ak}),o_{ak}-2}}} \tag{5.4}$$

In this test, the current value $\overline{s_a}$ of each signal $s_a \in S_{\mathcal{A}}$ in the currently observed class k is compared to the previously calculated mean μ_{ak} and standard deviation σ_{ak}. The current observation is considered as anomalous if the value z_a exceeds a threshold. This threshold is calculated using the number of observations o_{ak} of the signal s_a and the value of the t-distribution with a significance level of $\alpha/(2o_{ak})$ and a degree of freedom of $o_{ak} - 2$ referred to as $t_{\alpha/(2o_{ak}),o_{ak}-2}$. The value of α indirectly influences the sensitivity of this anomaly detection and has to be chosen function-specific.

For every signal in the selected subset, the calculation is done for each new observation of the signal. Since this anomaly detection is signal-independent, it can be distributed in the architecture and calculated directly at the source of the signal. Each of the used values for the Grubbs' test is already calculated by Algorithm 1 and can be directly reused. As the basis for this anomaly detection also any other anomaly detection algorithm could be used which is capable of processing streamed data.

For this evaluation we set the parameters as follows: For anomaly detection, we consider the top $|S_{\mathcal{A}}| = n = 30$ signals and set a threshold of $\alpha = 0.1$ for the t-distribution in the Grubbs' test. For further investigation, different parameters can be evaluated. Depending on the criticality of the observed function, the parameters should be set accordingly.

5.3 Evaluation

In this section, we will present the actual evaluation of the proposed *anomaly detection* and demonstrate the streaming capabilities of the proposed approach. In the first paragraph, we will introduce the metrics used for this evaluation, followed by the gained results and its discussion. Finally, we assess the threats to validity of this evaluation.

5.3.1 Metrics

We use two different metrics to evaluate the runs: For the evaluation of the signal selection, we use the Jaccard distance [180] of the top $|S_A| = n = 30$ signals between consecutive time steps of 300 seconds. As already introduced in Section 4.6.2, the Jaccard distance shows the difference between two different sets S_i and S_j (cf. Eq. (4.9)):

$$J_{ij} = J(S_i, S_j) = \frac{|S_i \cup S_j| - |S_i \cap S_j|}{|S_i \cup S_j|}, (S_i \cup S_j) \neq \emptyset$$

where $J_{ij} = 1$ if the sets are entirely different to each other and $J_{ij} = 0$ if similar to each other. This allows visualising the difference in the signal sets between two observed time steps. The n-sized signal sets are fundamental for the anomaly detection approach. The optimal curve for this metric should be a high Jaccard distance for the initialisation at the beginning, and it should subside—meaning no more changes in the signal subset.

For the evaluation of the anomaly detection, we consider all recorded actions in the trace as nominal behaviour of the user. To describe the anomaly detection, we use the ratio of signals for which an anomaly is detected. In this ratio, we only consider signals within the n-sized signal set. This ratio is calculated for two classes:

(1) Once for the currently active class of the function in the trace. This ratio corresponds to the *false positives* of the anomaly detection and represents a detected anomaly in the recorded trace, which we consider as nominal behaviour. This ratio should always be as low as possible.

(2) The second ratio considers the opposite state of the function recorded in the trace. This state was not recorded in the trace, and we consider this hypothetical function state as not intended by the user, and we refer to this as an anomaly, due to not being present in the real trace. This ratio corresponds to the *true positive* rate of the anomaly detection and should always be as high as possible.

The *true negative* and *false negative* ratio can be calculated with the other two values and is not further mentioned, due to simplicity.

5.3.2 Results and Discussion

For the analysis of our results, we classify the results of the anomaly detection into seven patterns. Each of these patterns will be presented in the following:

Pattern 1

For the first changes in the state of the function, no false positives are detected. After these changes, the true positive rate increases and, changes not triggered by the user, will lead to detected anomalies. In this pattern, all potential anomalies can be detected without triggering any false alarms.

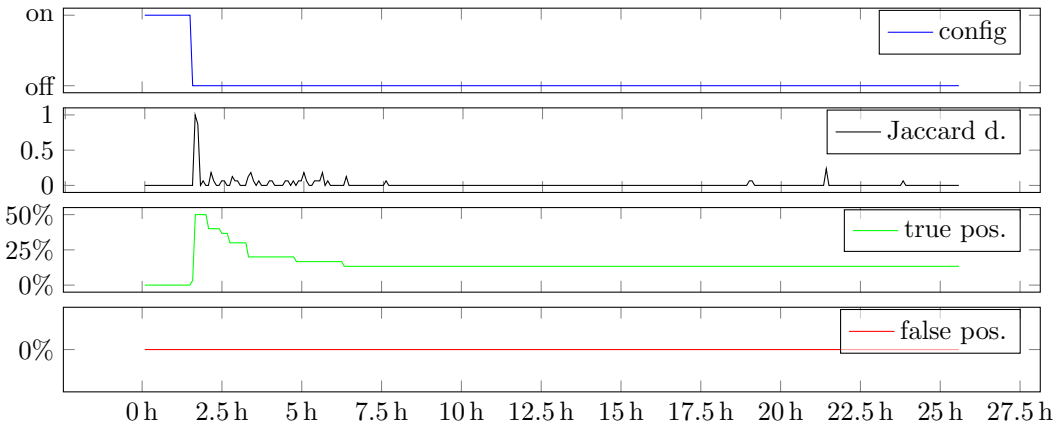


Figure 5.3: Exemplary result with pattern 1: Lane Departure Intervention in data set 25

In Figure 5.3 an exemplary result of pattern 1 is shown. As depicted in the upper plot, the *Lane Departure Intervention* is switched off after 1.5 hours of the trace. Right after this change, the signal subset initialises, due to the first observation of another state. With the initialisation of the signal subset, true positives are detected for ≈50% of the signals. The percentage declines over time, but still, within the full trace, we would always detect potential anomalies. The signal subset changes only slightly over time and rarely changes in the last half of the trace. Within the full trace, no false positives are detected, and we would not trigger any false alarms. In total, 20/91 runs are similar to this pattern 1. This pattern represents a *perfect* function of the anomaly DDF. All potential anomalies would have been detected, and no false notifications would have been triggered.

Pattern 2

Within the first changes in the function's state, a false positive is detected. This false positive will subside, and we observe a high true positive rate. In this case, the user is informed at

the first change of the setting. Afterwards, a change which is not triggered by the user will be detected as an anomaly.

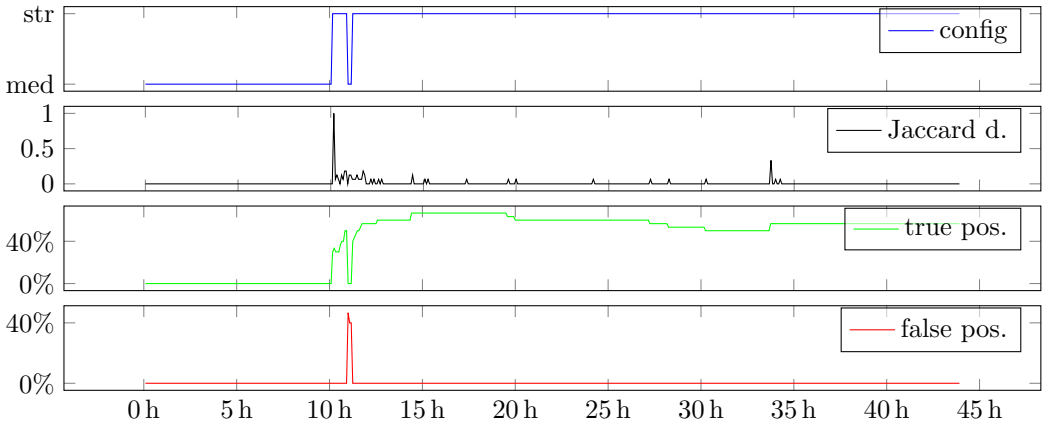


Figure 5.4: Exemplary result with pattern 2: Steering Wheel Vibration in data set 14

An example of pattern 2 is depicted in Figure 5.4. After around ten hours, the *Steering Wheel Vibration* is changed from *medium* to *strong*. After this first change in the configuration, the signal subset initialises, as seen in the Jaccard distance. This leads to the detection of potential anomalies. Shortly after this change, the configuration is changed back to the setting *medium*, and a false alarm would have been triggered. After a short period, this setting is changed back; the false-positive rate drops to 0%. The true positive rate increases and anomalous changes in the configuration would trigger anomalies. This pattern 2 represents an almost perfect anomaly detection with only one false alarm right at the beginning of the first reconfiguration. In total 8/91 runs are similar to this patter. Together with the pattern 1, the approach was able to detect in 31% of the runs all potential anomalies and at maximum triggered one false alarm.

Pattern 3

With a change in the function's state, a true positive is detected. It will subside, and no false positives or true positives will be detected any more. In this case, a potential anomaly would be detected after the first change, but no anomalies will be detected afterwards.

Figure 5.5 shows an example of pattern 3. In this case, the *Lane Change Sensitivity* is reconfigured from *medium* to *late* after around 25 minutes of the trace. After this reconfiguration, the signal subset initialises and potential anomalies would have been detected within the next hour after the change. Afterwards, no more anomalies would have been detected any more. Over the full trace, no false notifications would have been triggered. Here, 4/91 runs are similar to this pattern. In these runs, only initial anomalies were detected.

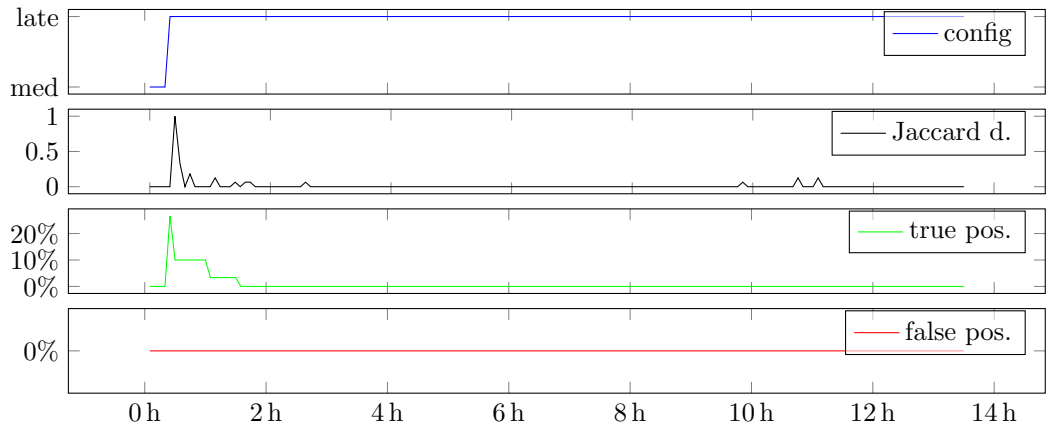


Figure 5.5: Exemplary result with pattern 3: Lane Change Sensitivity in data set 36

Pattern 4

Here, true positives and false positives were observed almost at all times. In this case, the user will receive a notification; regardless if intended or not.

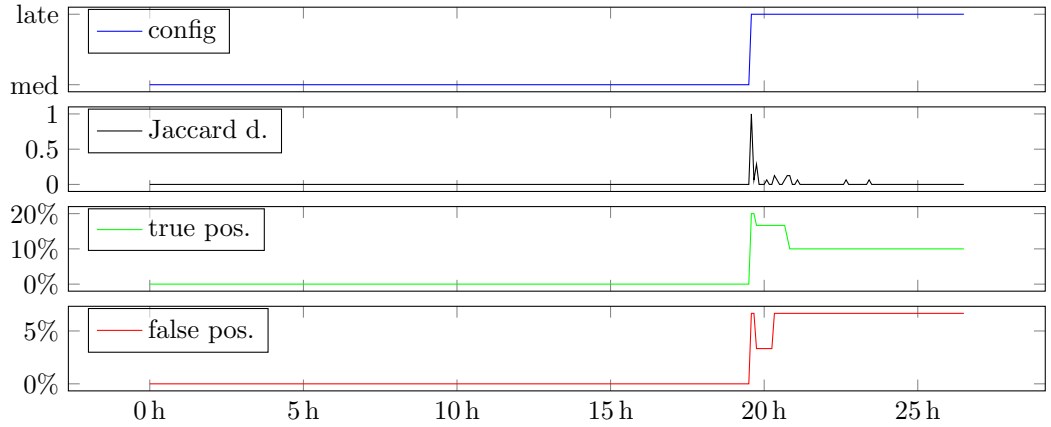


Figure 5.6: Exemplary result with pattern 4: Lane Change Sensitivity in data set 36

In Figure 5.6, an example of pattern 4 is shown. In this case, the configuration of the *Lane Change Sensitivity* is changed from *late* to *medium* after around 19.5 hours. With this first reconfiguration, the selected signal subset initialises and true positives and false positives are detected for the rest of the trace. This means potential anomalies would have been detected, but every change would also lead to a notification to the user. We observed this pattern in 3 out of 91 runs.

Pattern 5

Pattern 5 is only observed, when the function is in a specific state for a longer duration than in the other states. Each time the function is in the less observed state (i. e. less used state), we can see a high false-positive rate and a low true positive rate. Vice versa in the more observed state, a low false-positive rate and a high true positive rate can be seen. This leads to a message for the user, each time the user is changing to the less observed state of the function.

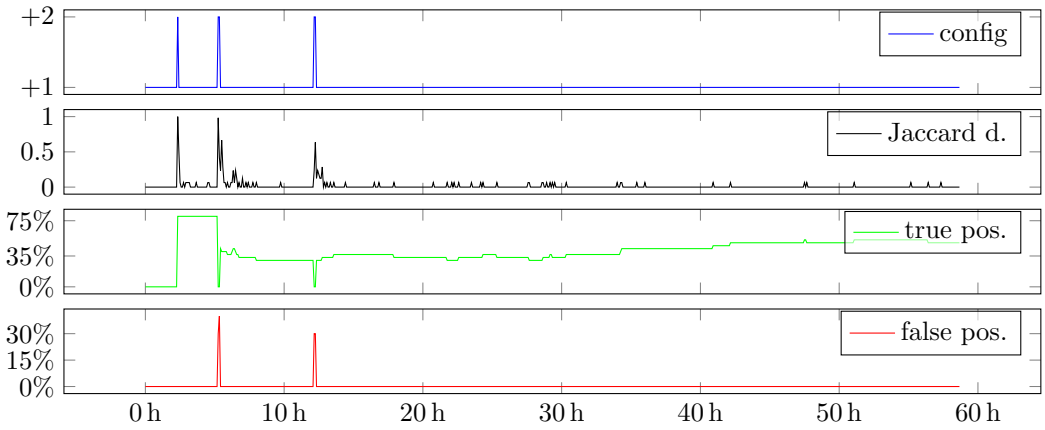


Figure 5.7: Exemplary result with pattern 5: Speed Limit Assist Offset in data set 8

Figure 5.7 depicts an example of pattern 5. Here, the *Speed Limit Assist Offset* is changed three times within the trace. After the first reconfiguration after around 2.3 hours from an offset of +1 km/h to +2 km/h, the signal subset initialises and potential anomalies would have been detected. Throughout the rest of the trace, we would always detect potential anomalies in the configuration +1 km/h, which is longer observed by the DDF. In both short changes to +2 km/h, a false alarm would have been triggered, and in this state, we also would not detect any potential anomalies. In 8/91 runs, we observed this type of pattern and we would detect anomalies in the longer observed state and trigger false positives in the shorter observed state.

In all of the before mentioned patterns (Patterns 1 to 5) we were in total able to detect potential anomalies in 47% of the runs, even though in some cases it leads to false positives.

Pattern 6

Here, no true positives or false positives are observed at all. In this case, the user will not receive any message, and no anomalies can be detected. This behaviour represents the current state-of-the-art in vehicles, as nothing will be triggered.

Figure 5.8 shows an exemplary trace for pattern 6. Here, the *Lane Departure Sensitivity* has been reconfigured three times from *always* to *reduced* and back. After each change the selected signal subset changes, but no true positives or false positives had been detected. This means no potential anomalies could have been detected, but also, no false alarm would have

been triggered. This is similar if the DDF would not run at all, and thereby has no negative impact on the user. This kind of pattern was observed for 43/91 runs.

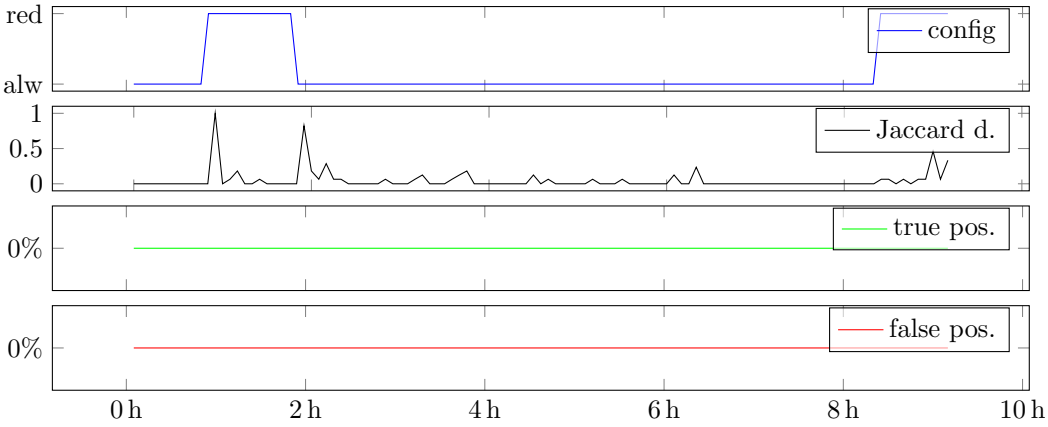


Figure 5.8: Exemplary result with pattern 6: Lane Departure Sensitivity in data set 48

Pattern 7

In this pattern, false positives and no true positives are observed almost at all times. In this case, the user will receive a notification; regardless if the change was intended or not.

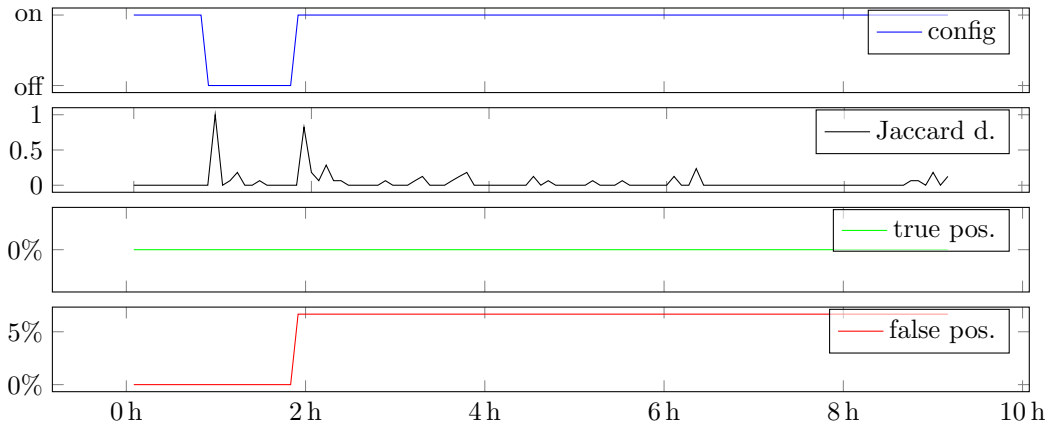


Figure 5.9: Exemplary result with pattern 7: Lane Departure Warning in data set 48

In Figure 5.9, an example of pattern 7 is shown. In this case, the *Lane Departure Warning* switched off and on once each. We see initialisation of the selected signal subset after each reconfiguration. With the second reconfiguration, a false positive for a small amount of selected signals is detected (6%). Throughout the rest of the trace, this does not change—besides, no potential anomalies where detected. In 5/91 of the runs, we can see similar results.

Discussion

To sum up, function changes that were not triggered by users led to a detected anomaly in 43/91 (47%) runs (Pattern 1 to 5). In 24/91 (26%) runs (Patterns 2, 3, 5, and 7), the user was notified at least once when they changed the setting of the function, and in 48/91 (53%) runs (Patterns 6 and 7), no anomalies were detected if the user would not have triggered the change. In 43/91 (47%) runs (Pattern 6), no false notifications but also no anomalies would have been triggered.

This means that in 47% of the runs, we would have detected potential anomalies, and in 47% no notifications would have been triggered at all, similar to current systems. Only in 5% of the runs, we would have only triggered false notification without detecting any anomalies. This overall detection rate can be improved by using more sophisticated anomaly detection algorithms. In this evaluation, we showed that the proposed signal subset selection approach is suitable for streaming configurations. It can be used to efficiently implement DDF as anomaly detection algorithms on the already present vehicle data, and not the actual capabilities of the anomaly detection algorithm.

5.3.3 Threats to the Validity

Similar to the evaluations in the previous chapter, a significant threat to validity concerns the selection of vehicles. There is a vast amount of in-house test vehicles data, but in our case, the approach has to be evaluated on real customer data, due to the behaviour we want to learn from the user. We tried to prevent this by using traces at least from different regions of the world. Another threat is the missing anomalous traces. We decided to evaluate our approach on nominal traces and perform a hypothetical switch to the different state, not to damage a car or even endanger ourself or someone else by creating anomalous traces with a real car.

Chapter 6

Onboard Proof of Concept

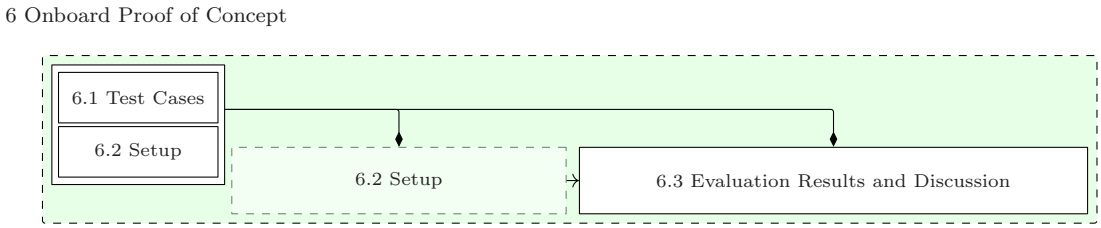


Figure 6.1: Chapter structure

In the previous chapters, we evaluated the approach on pre-recorded vehicle signals, meaning no requirements for an onboard execution have been posed to the approach yet. In this chapter, we conduct an additional evaluation to proof the onboard capabilities of the proposed approach in a real vehicle setup. An overview of this chapter's structure can be found in Figure 6.1. First, the test cases for this evaluation are presented, followed by the setup of the test vehicle and its implementation. Finally, the results of this evaluation are presented and discussed.

6.1 Test Cases

As a basis for the onboard evaluation, we only use the test cases of the type *system function*, which have been presented in Section 4.1.1 and have also been used in the previous evaluations. Namely, these test cases are *Day/Night Mode*, *Power Consumption*, and *Valid Lane Markings*. We have chosen to conduct the onboard evaluation solely on these test cases, as this evaluation is conducted on an internal test vehicle which is driven by different test drivers and not by a real customer. Additionally, due to the changes made to the vehicle, the group of employees allowed to drive this vehicle was limited. Also, no information about the driver's identity was collected by the test setup. In this case, we cannot collect any user-specific behaviour or data.

The purpose of this evaluation is only to show the applicability of the approach for an onboard deployment. The actual proof of the performance of the approach has been presented in the last chapters.

6.2 Setup

As the basis for this evaluation, we used a BMW X5 [188] (cf. Figure 6.2). This vehicle has similar functionalities as the BMW 7 Series used in the previous evaluations (cf. [189, pp. 127,156f]). Figure 6.3 depicts the architecture of the test vehicle setup. This setup can be split into three parts. First, the vehicle's E/E architecture, including the data-access is shown, which is further introduced in the first paragraph. This includes all steps from the top of the figure to the *CAN Interface*. The second part represents the approach and is further introduced in the second paragraph of this section. The last part is the post-processing of the selected signals, which is presented in the last paragraph.

6.2.1 Vehicle

Figure 6.2: BMW X5 generation used for test setup [190]

In this evaluation, we use a production BMW X5. This means the vehicle is similar to a vehicle sold to customers, and all test hardware had to be retrofitted. Production vehicles only provide limited access to the vehicle's ECUs and communication networks. This thereby also limits our access to vehicle signals (cf. Challenges 1 and 2). To gain access to the vehicle signals, the vehicle has been modified with additional hardware. The upper part of Figure 6.3 shows an abstraction of the vehicle's E/E architecture, similar to Figure 2.5 introduced in Chapter 2. Please note that the figure does not represent the actual architecture of the vehicle's communication networks and only illustrates the hardware access to the vehicle signals. Here, the vehicle communication network consists of five major CANs, a FlexRay network, multiple LINs, and multiple Ethernet networks. We physically access all five major CANs via a custom-built vehicle network interface shown in Figure 6.4a. These five networks are then connected to

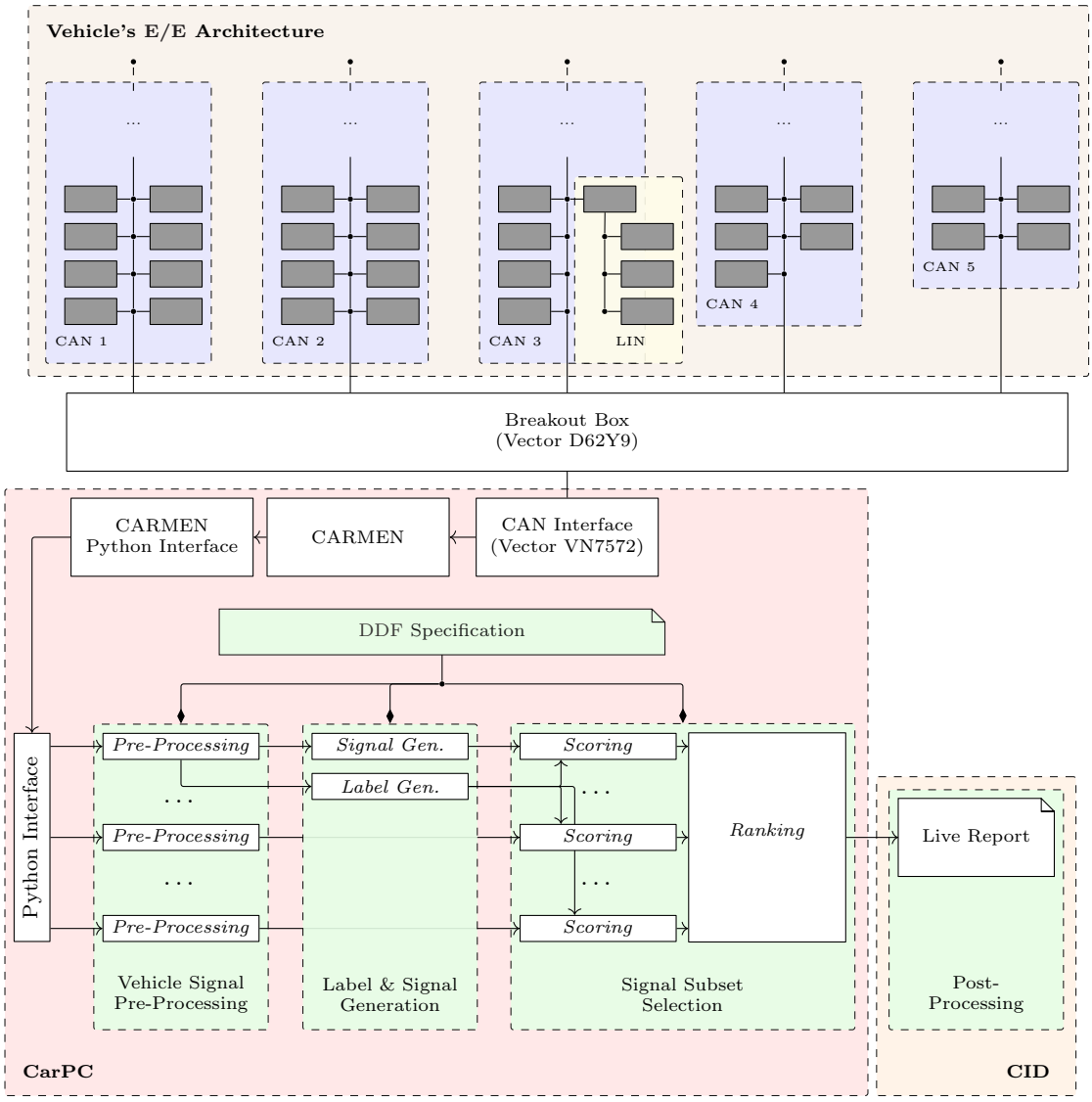
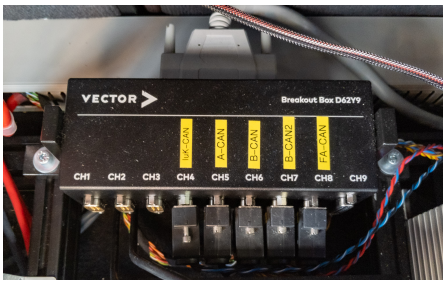


Figure 6.3: Proof of concept architecture

a breakout box (Vector D62Y9 [191]) shown in Figure 6.4b. The breakout box collects all CAN connections for the following CAN interface (Vector VN7572 [192]). The CAN interface itself is part of a retrofitted *CarPC*. This CarPC is similar to an ordinary PC with additional capabilities for automotive applications (e. g. shock-resistant, wider temperature operation range). Here, we retrofitted a *Fanless Box-PC 7902* from Delta Components with an Intel Xeon E3-1268L and 32GiB RAM (cf. [193]). In Figure 6.4c, the whole test setup is depicted. The vehicle interfaces can be seen on the right side of the figure and the CarPC is the silver box on the right side of the boot. All the equipment left from the CarPC is either required for the power supply of the test setup or to access the vehicle's camera system (which is not used in this work).

(a) Vehicle network access



(b) Breakout box for CAN interface

(c) Overview of the test setup

Figure 6.4: Test vehicle setup

6.2.2 Signal Data Processing and Subset Selection

The CAN messages were received by the CAN interface, which is accessed by the software tool CAR Measurement Environment (CARMEN). CARMEN is a BMW internal development tool for access and analysis of internal vehicle communication and is mainly used for the development of new vehicles. This tool already provides the corresponding device drivers for various CAN interfaces and can directly decode vehicle message frames into the corresponding PDUs and vehicle signals (cf. Figure 2.6). All decoded vehicle signals are exported via *localhost* User Datagram Protocol (UDP) messages by using a custom-built Python script and the provided Python interface of CARMEN. A second Python interface, which is part of the approach, receives the exported vehicle signals. All signals are then being processed, labelled, and ranked according to the DDF specification and the proposed approach. The specifications of the test cases are identical to the last evaluations (cf. Listings A.1 to A.3). For the pre-processing, we use the same setup as in the offline signal subset evaluation, including continuous data out of the last chapter (cf. Listing C.1). For the ranking and selection of the signal subsets, we use the modified version of the Fisher Score algorithm, including streaming processing and federated execution, presented in the last chapter. In this test setup, also any other supervised feature selection with streaming processing capabilities could be used (e. g. [117] or [194]).

6.2.3 Live Report

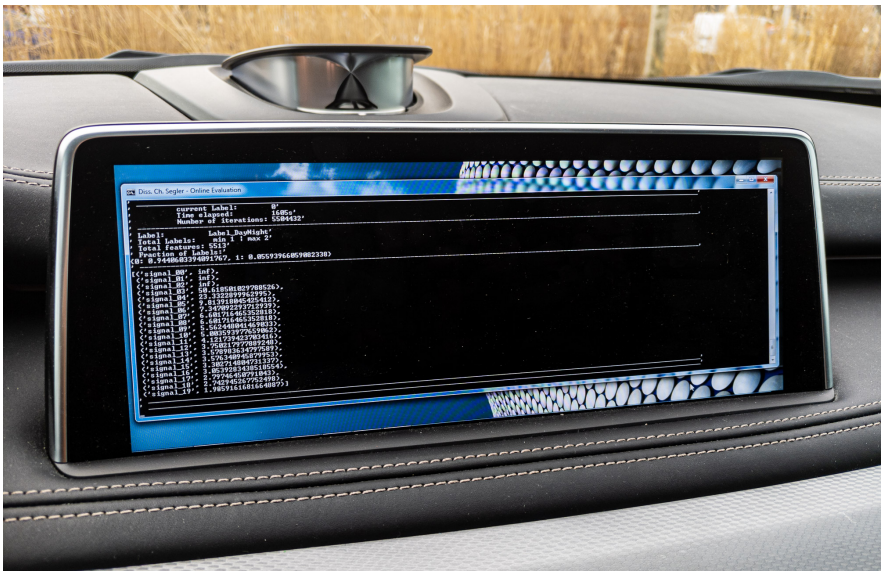


Figure 6.5: Live report on CID

The selected signal subset is then displayed on the vehicle's Central Information Display (CID) on the dashboard of the vehicle. In Figure 6.5, a photo of this live report is shown. The live report shows the current active label and the elapsed time of the current test drive,

followed by the active test cases, the total number of features and the fraction of each distinct label. The lower part shows the live ranking of the top $|S_{\mathcal{A}}| = 20$ signals and their current score. Unfortunately, the signal names in the report had to be omitted due to confidential information, but this would not differ to the report shown previously (cf. Listing 3.2).

6.3 Evaluation Results and Discussion

We first evaluate the maximum number of signals which can be processed on-the-fly onboard a vehicle and the required computational resources of our setup. For this evaluation, we configured the maximum sampling rate at which a vehicle signal is forwarded to the Python interface of the implemented approach.

We started with a maximum sampling rate of $5.0s$ and reduced the sampling rate in steps of $0.1s$. With a minimum sampling rate of $0.5s$, all vehicle signals, we had access to, can be forwarded to the approach and processed without loosing any signal values, or generating any buffer overflows. When setting the sampling rate to $0.4s$, we started loosing signal values after $42s$ of starting the test setup. When further decreasing the sampling rate, the time when loosing the first values decreased. These values were lost, due to a bottleneck between CARMEN and the CARMEN Python interface. Unfortunately, the source code of the CARMEN Python interface is not disclosed, and we cannot further investigate this bottleneck. However, all steps from our approach did not show any bottlenecks or even lost values, and we able to processed all values which were received from the CARMEN Python interface.

During the testing, CARMEN on average used around 15% to 17% of the CPU cycles and around $530MiB$ of RAM. The implementation of the approach itself only used around 1% to 2% of the CPU cycles and around $60MiB$ of RAM. Here, we used CARMEN solely for the connection to the CAN interface and the decoding of the received CAN frames. The actual approach only used minimal computational resources to decode and process all available signals and could be easily executed on embedded hardware.

Figure 6.5 shows a live report of a conducted test drive. Here, we used a sampling rate of $0.5s$ with the test case *Day/Night Mode*, as seen in the upper part of the live report. The total length of this drive was around 30 minutes, and in total 5677 decoded vehicle signals have been processed. This resulted in a total of about 5.9 million processed values. Please note that this number of signals is smaller than in the data sets in the previous evaluations. The reason is that we only have physical access to some of the major CANs and therefore, did not have access to all vehicle signals. Additionally, the used test vehicle has an older E/E architecture generation, as in the previous evaluations, which has a lower total number of signals. Here, we also only scored signals which have been received at least once. For example, if in the test drive, the sunroof was never used, we also would never receive such a signal. As this test drive has only the length of 30 minutes, not all functions have been used, and not all signals were observed.

The lower part of the live report shows the ranking of the signals according to the used test case. The first three signals show a perfect correlation to the signal (i. e. an infinite score), as these signals either are directly used for the generation of the label *day mode* or *night mode* or contain this information. The fourth signal with a score around 50 is the actual signal of the brightness sensor controlling this function, which shows a high correlation. This is also the signal we would expect as the most top-ranked signal.

In all test drives and the three used test cases, the approach was able to decode all arriving signal values and identify the corresponding signals. Only when using sampling rates below 0.5s, we could not forward all signal values to the approach, due to constraints in CARMEN. We only use CARMEN for implementation of the test setup and to show the onboard capabilities of the approach. In case of integration of the proposed approach in the vehicles E/E architecture, we would have direct access to the vehicle signals without any testing tool like CARMEN.

The here shown evaluation is very limited, and the major threat to validity concerns the used test vehicle and the accessed CANs, as new vehicle generations generate more signals. Here, we have only tested an online implementation of the approach on a current-generation vehicle and only on a part of the vehicle's networks. However, we also showed the only bottleneck was the connection to these networks and not the actual approach. If we would have direct access to the vehicle signals, the implemented approach could be executed on all received signals. As this approach was implemented in Python and did not include any optimisation for the execution platform, the used resources and performance can be even further improved. Here, we only wanted to show the onboard capabilities of the approach and proof the execution in a "real world" example.

To sum up, we were able to show that our approach can be executed onboard a vehicle, although we had a bottleneck in the connection to the car. This bottleneck is not part of our approach, and could easily be solved with direct access to the vehicle signals. Additionally, we showed the low computational requirements of our approach.

Chapter 7

Summary and Conclusion

The considerably rich amount of signals and data already present in current vehicles enables the development of new functions, especially DDFs. This work presented an approach to automatically select a signal subset out of all available vehicle signals, which should be used as input for a particular DDF. It, thereby, helps the function developer in the design phase of DDFs, especially to handle the vast amount of potential input data.

Figure 1.1 in the first chapter depicts the structure of this work. At first, we gave a motivation for this work, introduced the posed research questions (RQs). The second chapter introduced the background for the approach and the impact of the automotive E/E architecture on DDFs. Chapter 3 presented the proposed approach step by step. Next, we evaluated the approach. The first three evaluations, presented in Chapter 4, were conducted in a purely offline manner on static data, and here we evaluated each step of the approach step by step. In the next evaluation, presented in Chapter 5, we demonstrated the streaming capabilities of the approach, meaning that no significant amount of data had to be stored or transferred. Finally, we showed the real-world applicability with a proof of concept in a real vehicle. The following sections will conclude each chapter in more detail and will be concluded by the contributions of this work.

7.1 Background

In Chapter 2, we presented the background and related work for the proposed approach. First, we defined the term of DDF in the automotive domain, followed by an overview of applications. These applications range from system functions, to context-aware functions, and to anomaly detection systems. Based on the development workflow from machine learning functions, we presented a workflow for DDFs.

Next, we introduced the term E/E architecture and the purpose of this architecture. We gave an overview of current E/E architectures and their impact on the development of DDFs. We

identified seven challenges impacting DDFs in E/E architectures: (i) The federated architecture, (ii) internal signals, (iii) asynchronous communication, (iv) signal dimensionality, (v) signal specification, (vi) data volume, and (vii) limited computational resources. We linked these challenges imposed by the E/E architecture to the steps of the presented development workflow.

Finally, we gave an overview of the related work in this field. In general, we divided the related work into semantic approaches and data-driven approaches. We introduced the state-of-the-art in semantic approaches with annotation of vehicle signals and showed the limited applicability of these approaches, as they require manual annotation. With the high dimensionality of currently up to 14 000 signals, this would not scale well. We also introduced data-driven methods for the identification of potential input signals for a DDF. We gave an overview of the currently existing methods and clustered them based on four categories.

7.2 Approach

Chapter 3 presented the primary approach of this work. At first, we gave an overview of the underlying concept and briefly introduced each step of the approach. Next, we presented each step in detail.

First, we presented the *DDF specification* in Section 3.2, which is the basis for each step. In the first step of the approach, the *vehicle signal pre-processing*, we presented the data access and pre-processing, which is the basis for all following steps. The presented data access and pre-processing is specially designed for the characteristics of the automotive E/E architecture. The pre-processing is followed by the *label and signal generation* step presented in Section 3.4. The pre-processed signals combined with the generated labels and signals, the actual signal subset is selected by the next step, presented in Section 3.5.

This signal subset selection process identifies the proposed signals, which should be used as input for a specific DDF out of all available vehicle signals. This step is mainly based on feature selection algorithms introduced in Chapter 2. The selected signal subset can then be post-processed in multiple ways. Section 3.6 presented the possible post-processing steps (i. e. how these signals are used after the selection step). This can be done manually or automatically.

Next, we presented the deployment strategy of the approach regarding the onboard and back end execution and selected data sets. This deployment strategy is based on the type of the selected signal subset, which we also introduced in the deployment strategy.

7.3 Offline Evaluation

For the evaluation of the proposed approach, we conducted multiple evaluations. Chapter 4 presented the first three evaluations. These were only conducted offline on data collected from customer vehicles. First, we presented the underlying 24 test cases, the test setup and the 101 data sets used for the conducted evaluations. Next, we presented and discussed these three

evaluations separately.

In the first evaluation, we assessed the proposed vehicle signal pre-processing step and compared its performance to a baseline without any pre-processing. Here, we compared two different types of machine learning algorithms as the basis for each test cases. In the case of the RF classifier, the DDF performed similarly on the raw vehicle signals and the pre-processed signals. However, in the case of the SVM classifier, only the classifier using the pre-processed signals was able to achieve a good performance.

This shows the importance of pre-processing of signals for machine learning tasks. Often this step is manually designed for each use case. In the case of automotive data and the vast amount of vehicle signals, this step has to be automated, based on the already present signal specifications. When using machine learning algorithms which are capable of splitting and scaling signals during the training phase (e. g. RF, decision trees), the proposed pre-processing step is not necessarily required. Nevertheless, it can be used to filter and reduce the amount of data and does not have a significant impact on the performance of the classifier. In the case of linear machine learning algorithms (e. g. SVM, deep neural networks), we can observe that a pre-processing step is mandatory to train these type of classifiers. With the current trend to deep learning, correct pre-processing of vehicle signals becomes inevitable.

Next, we evaluated the signal subset selection step of the proposed approach. To minimise the effect of the selected algorithm, we evaluated 15 different algorithms on 24 test cases, using 101 data sets. In total, we conducted 10 200 evaluation runs (680 test cases and data sets $\times$ 15 algorithms). Here, the algorithms Chi^2, DISR, FScore, Fisher Score, Gini Index, MRMR, and Trace Ratio provided signal subsets that achieved good and robust results for the trained classifier over all test cases independent of the used type of classifier. Additionally, from these algorithms, Chi^2, FScore, Fisher Score, and TraceRatio Fischer are distinguished by their fast execution. Despite their fast execution, the results are similar or even better than more complex algorithms with much higher execution times. The overall good performance of the trained classifiers based on signals selected, showed the applicability of the proposed approach and also, that in most cases enough signals are already present to achieve good results. When using the signal subset selected by seven out of the 15 algorithms, the performance of the classifier was even increased in comparison to a classifier trained on all signals. This also shows the dependability of the approach on the performance of the used algorithm.

In the third evaluation, we assessed the deployment strategy by comparing the different selected signal subsets between the data sets (i. e. users). All these evaluations were based on the same 24 test sets and 101 data sets. In summary, we showed that we observed all three types of signal subsets $S_{\mathcal{A}}^{I}$, $S_{\mathcal{A}}^{II}$, and $S_{\mathcal{A}}^{III}$, and that, depending on the test case, there is a need for a highly personalised selection of appropriate input signals for a DDF. At the same time, we were also able to see that a minimum amount of data from each user is required to select the appropriate subset and that this varies from test case to test case.

7.4 Streaming Evaluation

One of the main challenges of current E/E architectures is the vast amount of vehicle data and limited resources for storing or transferring this data (cf. Challenge 6). In the evaluation presented in Chapter 5, we assessed the approach on streaming data, meaning no data has to be stored within the vehicle or the back end. This also can be directly performed onboard the vehicle. As the basis for this evaluation, we used the same data sets as presented in the offline evaluation and the test cases of the type *anomaly detection*. First, we introduced the test case of the streaming anomaly detection and its basic concept and the used test setup.

We demonstrated the streaming capabilities of our approach and proposed a novel DDF for providing proactive safety management for customer functions in automotive systems at run-time. This DDF can detect safety violations for personalisable functions and support the software engineering process by gathering run-time data from the vehicles already running in the field. We have modified the Fisher Score feature selection algorithm and performed on the reduced data the statistical Grubbs' outliers detection test pointing to anomalous system behaviour. With this modified algorithms, we showed the full capabilities of our approach and minimised the impact of the E/E architecture on DDFs.

Based on the 101 data sets and twelve ADAS functions, we evaluated our approach on 91 runs, where the user reconfigured a function at least once. For over 47% (43/91 runs) of these runs, we were able to detect and flag possible anomalies. In 47% (43/91 runs) of the runs no anomalies, but also no false alarms have been triggered, which is similar to the current systems without any anomaly detection. Only in 5% (5/91 runs) of the runs, we would have only triggered false notification without detecting any anomalies. This overall detection rate can be improved by using more sophisticated anomaly detection algorithms. However, we wanted to demonstrate that the proposed signal subset selection approach is suitable for streaming configurations.

7.5 Onboard Proof of Concept

In the previous evaluations, we assessed the approach on pre-recorded vehicle signals, meaning no requirements for an onboard execution have been posed to the approach. With a proof of concept presented in Chapter 6, we demonstrated the onboard capabilities of the proposed approach in a real vehicle setup.

With the test cases *Day/Night Mode*, *Power Consumption*, and *Valid Lane Markings* and test setup in a real vehicle, we demonstrated the applicability of the proposed approach for an onboard execution. As the basis for this evaluation, we have used a BMW X5 with similar functionalities as the BMW 7 Series used in the other evaluations.

In all test drives and the three used test cases, the approach was able to decode all arriving signal values and to identify the corresponding signals. Only when using sampling rates below 0.5s, we were not able to forward all signal values to the approach, due to constraints in the

software CARMEN. Here, we only used CARMEN for implementation of the test setup and to show the onboard capabilities of the approach. In case of integration of the proposed approach in the vehicle's E/E architecture, we would have direct access to the vehicle signals without any testing tool like CARMEN.

We were able to show that our approach can be executed onboard a vehicle, although we had a bottleneck in the connection to the car, which was not part of our approach, and could be easily solved with a direct access to the vehicle signals. Additionally, we showed the low computational requirements of our approach, although the implementation was not fully optimised.

7.6 Research Questions and Contributions

For this work, we posed a total of five RQs. These have already been introduced in the first chapter, but we will briefly recap these and show the contribution of this work:

RQ 1 (Signal Subset Identification)
How to identify vehicle signals which are potentially relevant as input for a specific DDF?

The first RQ was related to the *curse of dimensionality* and the vast amount of vehicle signals available in current vehicles. For the development of DDF in the automotive domain, it would not be reasonable to use all vehicle signals as input. To reduce the number of input signals, the appropriate vehicle signals have to be identified and selected. The presented approach showed in the evaluation that it was able to choose appropriate signals for the training of a DDF on multiple test cases and different users and thereby gives an answer to this RQ.

RQ 2 (E/E Architecture Characteristics)
How to integrate the identification approach into the vehicle's architecture and consider the characteristics of automotive E/E architectures?

The second RQ was related to the vehicles architecture and the challenges posed by it. Current automotive E/E architectures are historically grown and not designed for data mining or machine learning applications. The characteristics of these E/E architectures have a significant impact on DDFs and pose additional challenges. The presented approach can entirely be executed in a distributed manner until the last step, where each score is collected. Additionally, the approach supports an onboard execution without storing any signal data and works on the available signal specifications. This minimises the impact of the current E/E architectures on data-driven applications. In the evaluation, we were able also show that the approach works on real vehicle data and even in an actual vehicle at run-time.

RQ 3 (Scalability)
How to scale this identification approach over large vehicle fleets from simple system-related DDFs to highly personalised DDFs?

The third RQ was related to deployment and scalability. The developed DDFs will not only be deployed on one single vehicle. As an OEM, this will involve large vehicle fleets. These functions can range from simple system-related DDFs which are similar for all vehicles, to highly personalised DDFs which are unique for every user or vehicle. Therefore we extended the previous RQs with the additional requirement of scaling the identification approach over large vehicle fleets. In the deployment strategy, we proposed a classification of signals subsets and their dependability on the system's or user's behaviour. With this classification, we introduced a deployment strategy answering the question about the scalability of this approach (even for highly personalised DDFs). Additionally, we were able to show in the evaluation that this personalisation is needed.

RQ 4 (Manual Post-Processing)
How to assist the function developer in identifying potential information and signals which could be relevant as input for a specific DDF?

RQ 5 (Automatic Post-Processing)
How to automatically use the discovered input signals for a specific DDF?

The fourth and fifth RQ were related to the actual integration of the resulting signal subsets and their post-processing. This can either be done by the developer manually or automatically. Therefore we posed two RQs related to the integration of the discovered vehicle signals into the DDF. The first one considers the manual integration by the function developer into the DDF. The second one considers the automatic integration into the DDF. In this work, we consider both the manual integration of the signal subsets into the DDF and the automatic integration in the case of highly personalised DDFs. In the post-processing of our approach, we presented a solution for the manual integration and the automatic integration. By this, we presented a possible answer to both of these RQs. Additionally, we showed in the evaluation that—at least with the automatic integration—the DDFs performed well.

In addition to these five RQs, this work also contributed to the state of practice. As part of the presented approach, we proposed an automatic pre-processing of vehicle signals for data mining and machine learning purposes, entirely based on the currently available specification of vehicle signals defined by the AUTOSAR standard. This part of the approach can be easily used not only for DDFs but for any other data mining, data analytics, or machine learning tasks. In one of the evaluations, we also compared 15 state-of-the-art feature selection algorithms on automotive data. The shown results help future developers to decide on the best algorithm for their use case.

7.7 Conclusion

In this work, we presented an approach to discover vehicle signals, which represent a system's or user's behaviour when interacting with functionalities. These signals can directly be used as

the basis for DDFs, which are often hidden in the high amount of vehicle data. In multiple evaluations, we demonstrated the capabilities of the approach, which can, in addition, be executed onboard a vehicle without having to store large amounts of data onboard. Also, the real-world applicability of the approach was demonstrated with a proof of concept in a real vehicle.

By using the resulting signal subsets, we were able to train DDFs, which achieved a good performance in predicting the behaviour of the system or user. These results show that current vehicles are already equipped with enough sensors and provide a solid basis to predict a system's or user's behaviour. Still, this data has to be discovered first of all. Considering the vast amount of data as well as the highly diverse behaviour of every user, the selection of the appropriate signals cannot be easily performed manually. In order to use feature selection algorithms for this task, an automatic pre-processing method for vehicle signals has to be used, which has been presented as part of the approach. This method is solely based on the specification of vehicle signals by the widely used AUTOSAR standard and is thereby independent of the OEM.

The evaluations of 15 different feature selection algorithms also showed that simple algorithms provide solid results and even outperform more complex algorithms, if sufficient data samples can be obtained onboard the vehicle. These simple algorithms can be executed on embedded hardware onboard the vehicle and thereby provide the function developer fast results, without the need of any vehicle data transfer to the OEM's back end. This also has the significant advantage that the privacy of the driver/customer is affected as little as possible and still allows the developer to design personalised functions.

Chapter 8

Outlook

In this work, we presented and evaluated an approach to ease the development of DDFs in automotive E/E architectures, by proposing relevant input signals for such functions. By identifying the relevant input signals for a DDF, the function developer will easier find the appropriate signals representing the user's behaviour. However, this is only one challenge the developer is facing in the design phase of automotive DDFs.

8.1 Signal Subset Selection

First of all, we would like to mention a point in the signal subset selection, which is not fully addressed in this work. Here, we contemplate our approach as a single step in the development of DDFs, which is not yet part of a whole process within the design phase. CI/CD pipelines are established in many software domains, and this trend also affects automotive software, including DDFs [14]. However, in the case of DDFs, especially the training of the machine learning component, poses additional challenges (cf. [54]). In one of our publications, we proposed to integrate the here presented approach into a CI/CD pipeline for personalised DDFs (cf. [11]). This direct integration can ease and accelerate the development of DDFs. In the current implementation of the approach, we used YAML-files for the configuration and the report of the results. This human-readable markup language might be sufficient for a trained data scientist. However, in the case of a business analyst, this might not be optimal. Therefore, we propose to extend our approach with a Graphical User Interface (GUI) to facilitate the use of the approach integrated into a CI/CD pipeline.

8.2 E/E Architecture

In this work, we showed the major impacts of the E/E architecture on designing DDFs and the selection of the input data. Not only the selection of the input is significantly affected by

the architecture, also the actual deployment of the DDF. In one of our publications (cf. [13]), we introduced and assessed more challenges of E/E architectures on DDFs. One of these challenges is the static communication within the vehicle. Once a vehicle is built, it is nearly impossible to rearrange vehicle signals or even create new signals dynamically. For example, if the developer of a DDF identifies an essential signal with our approach, but this signal is not sent to the deployment target of their DDF, this signal cannot be used. IP-based networks (e. g. Time-Sensitive Networking (TSN)) and new software architectures (e. g. Service-Oriented Architecture (SOA)) are a step forward to dynamic communication within the vehicle. Even if we would have a dynamic communication in current E/E architectures, we still not have a central point in the architecture to access data, nor a data broker for this data. On top, currently, only few methods for Digital Rights Management (DRM) are implemented, which becomes more crucial in regards to privacy regulations.

Besides, more and more onboard analytics platforms will be integrated into the vehicle (e. g. [195] or [196]). The here presented approach was only one example of such an onboard analytic platform. All these additional computations will further increase the required computational resources. With the predominance of electric vehicles, the power consumption by these platforms will have a direct effect on the range of the vehicle. This can only be minimised with specialised hardware optimised for data analytics. This is not the case in current E/E architectures and has to be further investigated. In [12], we proposed a preliminary architecture for such hardware accelerators and in [20] a management concept for the produced heat, by distributing the computations between the components. These topics have to be further investigated.

8.3 Data Architecture

Not only the E/E architecture effects DDFs but also the data architecture of current vehicles. Current architectures are mostly focused on the functions. In this case, signals are only the means to an end. With the further advance of DDFs, this has to change. In the background chapter, we introduced the VSS as a data architecture for vehicles. However, the annotation of each data point still has to be performed manually and will not scale for e. g. 14 000 vehicle signals. We, therefore, propose to develop a data architecture which can be directly created from the function's specifications which generates the data. Architectures like SOA can provide sufficient annotations to directly describe the data each service is providing. The advantage of such an inheritance is that the developer has only to maintain one specification, which is also the primary purpose of the vehicle's architecture; providing functions for the customer. In this case, this specification should also be the single point of truth and not a separated data architecture. In the automotive domain, this is not yet fully investigated, and with the advance of DDFs this becomes more important.

8.4 Data-Driven Functions

Also, the way DDFs work has a high impact on the development process. With more such functions being implemented, this will gain further importance. These functions have additional requirements compared to classical functions. For example, the dynamic nature of these functions will require new design patterns (e. g. [10]) and safety assessments, as these functions will not any more be deterministic and different for every user. This will require new methods for the safety assessment of vehicle functions (e. g. [15]). Also, these DDFs directly learn from the user and directly interact with the user. The user can easily understand a good designed classical function as these are mostly deterministic. However, with functions directly learning from the user and interacting with him, new forms of HMIs are required (e. g. [16] or [17]). This research and development is only at the beginning and must be further investigated.

Appendix A

Test Case Specifications

In the following specifications, the vehicle signal names and values have been changed or partly not mentioned due to confidential information.

```
function:
  name:           day_night           # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - day_night_mode

labeling:
  - 0: day_night_mode == 0            # rule for label 0: day mode
  - 1: day_night_mode == 1            # rule for label 1: night mode
```

Listing A.1: Specification for the Test Case *Day/Night Mode*

```
function:
  name:           power_consumption   # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - power_demand
  - power_supply_1
  - power_supply_2
  - power_prediction

labeling:
  -  0:   10 > power_demand >=   0    # rule for label  0:  1st power consumption class
  -  1:   20 > power_demand >=  10    # rule for label  1:  2nd power consumption class
  -  2:   30 > power_demand >=  20    # rule for label  2:  3rd power consumption class
  -  3:   40 > power_demand >=  30    # rule for label  3:  4th power consumption class
  ...
  ...
  - 13: 140 > power_demand >= 130     # rule for label 13: 14th power consumption class
  - 14: 150 > power_demand >= 140     # rule for label 14: 15th power consumption class
```

Listing A.2: Specification for the Test Case *Power Consumption*

```
function:
  name:           lane_marking_valid  # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - lane_width

labeling:
  - 0:  lane_width  < 15              # rule for label  0:  lane marking valid
  - 1:  lane_width == 15              # rule for label  1:  lane marking invalid
```

Listing A.3: Specification for the Test Case *Lane Marking Valid*

```
function:
  name:           set_temp_driver     # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - hvac_temp_driver

labeling:
  - 0: 16 < hvac_temp_driver <= 18    # rule for label 0
  - 1: 18 < hvac_temp_driver <= 20    # rule for label 1
  - 2: 20 < hvac_temp_driver <= 22    # rule for label 2
  - 3: 22 < hvac_temp_driver <= 24    # rule for label 3
  - 4: 24 < hvac_temp_driver <= 26    # rule for label 4
  - 5: 26 < hvac_temp_driver <= 28    # rule for label 5
```

Listing A.4: Specification for the Test Case *HVAC Driver*

```
function:
  name:           set_temp_codriver   # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - hvac_temp_codriver

labeling:
  - 0: 16 < hvac_temp_codriver <= 18  # rule for label 0
  - 1: 18 < hvac_temp_codriver <= 20  # rule for label 1
  - 2: 20 < hvac_temp_codriver <= 22  # rule for label 2
  - 3: 22 < hvac_temp_codriver <= 24  # rule for label 3
  - 4: 24 < hvac_temp_codriver <= 26  # rule for label 4
  - 5: 26 < hvac_temp_codriver <= 28  # rule for label 5
```

Listing A.5: Specification for the Test Case *HVAC Co-Driver*

```
function:
  name:           acc_onoff           # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - acc_status
  - acc_gap
  - acc_speed

labeling:
  - 0: acc_status == 0                # rule for label 0: acc off
  - 1: acc_status == 1                # rule for label 1: acc on
```

Listing A.6: Specification for the Test Case *Proactive ACC*

```
function:
  name:           acc_gap             # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - acc_status
  - acc_gap
  - acc_speed

labeling:
  - 0: acc_gap == 1                   # rule for label 0: acc gap 1
  - 1: acc_gap == 2                   # rule for label 0: acc gap 2
  - 2: acc_gap == 3                   # rule for label 0: acc gap 3
  - 3: acc_gap == 4                   # rule for label 0: acc gap 4
```

Listing A.7: Specification for the Test Case *Proactive ACC Gap*

```
function:
  name:           window_driver       # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - window_position_driver
  - window_control_driver
  - window_button_driver

labeling:
  - 0: window_position_driver == 0    # rule for label 0: window closed
  - 1: window_position_driver == 1    # rule for label 1: window partly opended
  - 1: window_position_driver == 2    # rule for label 1: window fully opended
```

Listing A.8: Specification for the Test Case *Proactive Window Driver*

```
function:
  name:           window_codriver     # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - window_position_codriver
  - window_control_codriver
  - window_button_codriver

labeling:
  - 0: window_position_codriver == 0  # rule for label 0: window closed
  - 1: window_position_codriver == 1  # rule for label 1: window partly opended
  - 1: window_position_codriver == 2  # rule for label 1:window fully opended
```

Listing A.9: Specification for the Test Case *Proactive Window Co-Driver*

```
function:
  name:           seatheating_driver       # name of function
  max_inputs:     30                       # maximum number of selected signals

  blacklist:                               # blacklisted signals
  - status_seatheating_driver
  - temperature_seat_driver
  - button_seatheating_driver

labeling:
  - 0: status_seatheating_driver == 0      # rule for label 0: seatheating off
  - 1: 3 >= status_seatheating_driver >= 1 # rule for label 1: seatheating on
```

Listing A.10: Specification for the Test Case *Proactive Seatheating Driver*

```
function:
  name:           seatheating_codriver       # name of function
  max_inputs:     30                         # maximum number of selected signals

  blacklist:                                 # blacklisted signals
  - status_seatheating_codriver
  - temperature_seat_codriver
  - button_seatheating_codriver

labeling:
  - 0: status_seatheating_codriver == 0      # rule for label 0: seatheating off
  - 1: 3 >= status_seatheating_codriver >= 1 # rule for label 1: seatheating on
```

Listing A.11: Specification for the Test Case *Proactive Seatheating Co-Driver*

```
function:
  name:           proactive_dec       # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - status_ddc
  - control_ddc
  - config_drivetrain
  - config_engine
  - config_suspension
  - config_steering
  - config_display
  - config_gearbox

labeling:
  - 0: status_ddc == 0                # rule for label 0: comfort
  - 1: status_ddc == 1                # rule for label 1: eco pro
  - 2: status_ddc == 2                # rule for label 2: sport
```

Listing A.12: Specification for the Test Case *Proactive DEC*

```
function:
  name:           frontend_collision_warning  # name of function
  max_inputs:     30                          # maximum number of selected signals

  blacklist:                                  # blacklisted signals
  - status_frontend_collision_warning
  - config_frontend_collision_warning

labeling:
  - 0: status_frontend_collision_warning == 0 # rule for label 0: off
  - 1: status_frontend_collision_warning == 1 # rule for label 1: on
```

Listing A.13: Specification for the Test Case *Frontend Collision Warning*

```
function:
  name:           cross_traffic_alert  # name of function
  max_inputs:     30                   # maximum number of selected signals

  blacklist:                           # blacklisted signals
  - status_cross_traffic_alert
  - config_cross_traffic_alert

labeling:
  - 0: status_cross_traffic_alert == 0 # rule for label 0: off
  - 1: status_cross_traffic_alert == 1 # rule for label 1: on
```

Listing A.14: Specification for the Test Case *Cross Traffic Alert*

```
function:
  name:           lane_departure_warning  # name of function
  max_inputs:     30                      # maximum number of selected signals

  blacklist:                              # blacklisted signals
  - status_lane_departure_warning
  - config_lane_departure_warning
  - sensitivity_lane_departure_warning
  - intervention_lane_departure_warning

labeling:
  - 0: status_lane_departure_warning == 0 # rule for label 0: off
  - 1: status_lane_departure_warning == 1 # rule for label 1: on
```

Listing A.15: Specification for the Test Case *Lane Departure Warning*

```
function:
  name:           lane_departure_warning_sen    # name of function
  max_inputs:     30                            # maximum number of selected signals

  blacklist:                                    # blacklisted signals
  - status_lane_departure_warning
  - config_lane_departure_warning
  - sensitivity_lane_departure_warning
  - intervention_lane_departure_warning

labeling:
  - 0: sensitivity_lane_departure_warning == 0 # rule for label 0: low
  - 1: sensitivity_lane_departure_warning == 1 # rule for label 1: medium
  - 2: sensitivity_lane_departure_warning == 2 # rule for label 2: high
```

Listing A.16: Specification for the Test Case *Lane Departure Warning Sensitivity*

```
function:
  name:           lane_departure_warning_inter  # name of function
  max_inputs:     30                            # maximum number of selected signals

  blacklist:                                    # blacklisted signals
  - status_lane_departure_warning
  - config_lane_departure_warning
  - sensitivity_lane_departure_warning
  - intervention_lane_departure_warning

labeling:
  - 0: intervention_lane_departure_warning == 0 # rule for label 0: on
  - 1: intervention_lane_departure_warning == 1 # rule for label 1: off
```

Listing A.17: Specification for the Test Case *Lane Departure Warning Intervention*

```
function:
  name:           lane_change_warning  # name of function
  max_inputs:     30                   # maximum number of selected signals

  blacklist:                           # blacklisted signals
  - status_lane_change_warning
  - config_lane_change_warning
  - sensitivity_lane_change_warning
  - intervention_lane_change_warning

labeling:
  - 0: status_lane_change_warning == 0 # rule for label 0: off
  - 1: status_lane_change_warning == 1 # rule for label 1: on
```

Listing A.18: Specification for the Test Case *Lane Change Warning*

```
function:
  name:           lane_change_warning_sen   # name of function
  max_inputs:     30                        # maximum number of selected signals

  blacklist:                                # blacklisted signals
  - status_lane_change_warning
  - config_lane_change_warning
  - sensitivity_lane_change_warning
  - intervention_lane_change_warning

labeling:
  - 0: sensitivity_lane_change_warning == 0 # rule for label 0: low
  - 1: sensitivity_lane_change_warning == 1 # rule for label 1: medium
  - 2: sensitivity_lane_change_warning == 2 # rule for label 1: high
```

Listing A.19: Specification for the Test Case *Lane Change Warning Sensitivity*

```
function:
  name:           lane_change_warning_inter  # name of function
  max_inputs:     30                         # maximum number of selected signals

  blacklist:                                 # blacklisted signals
  - status_lane_change_warning
  - config_lane_change_warning
  - sensitivity_lane_change_warning
  - intervention_lane_change_warning

labeling:
  - 0: intervention_lane_change_warning == 0 # rule for label 0: off
  - 1: intervention_lane_change_warning == 1 # rule for label 1: on
```

Listing A.20: Specification for the Test Case *Lane Change Warning Intervention*

```
function:
  name:           side_collision_warning  # name of function
  max_inputs:     30                      # maximum number of selected signals

  blacklist:                              # blacklisted signals
  - status_side_collision_warning
  - config_side_collision_warning

labeling:
  - 0: status_side_collision_warning == 0 # rule for label 0: off
  - 1: status_side_collision_warning == 1 # rule for label 1: on
```

Listing A.21: Specification for the Test Case *Side Collision Warning*

```
function:
  name:           speed_limit_assist  # name of function
  max_inputs:     30                  # maximum number of selected signals

  blacklist:                          # blacklisted signals
  - status_speed_limit_assist
  - config_speed_limit_assist
  - offset_speed_limit_assist

labeling:
  - 0: status_speed_limit_assist == 0 # rule for label 0: off
  - 1: status_speed_limit_assist == 1 # rule for label 1: on
```

Listing A.22: Specification for the Test Case *Speed Limit Assist*

```
function:
  name:           speed_limit_assist_offset    # name of function
  max_inputs:     30                           # maximum number of selected signals

  blacklist:                                   # blacklisted signals
  - status_speed_limit_assist
  - config_speed_limit_assist
  - offset_speed_limit_assist

labeling:
  -  0: -15 =< offset_speed_limit_assist < -13 # rule for label 0
  -  1: -13 =< offset_speed_limit_assist < -11 # rule for label 1
  -  2: -11 =< offset_speed_limit_assist <  -9 # rule for label 2
  -  3:  -9 =< offset_speed_limit_assist <  -7 # rule for label 3
  -  4:  -7 =< offset_speed_limit_assist <  -5 # rule for label 4
  -  5:  -5 =< offset_speed_limit_assist <  -3 # rule for label 5
  -  6:  -3 =< offset_speed_limit_assist <  -1 # rule for label 6
  -  7:  -1 =< offset_speed_limit_assist <=  1 # rule for label 7
  -  8:   1  < offset_speed_limit_assist <=  3 # rule for label 8
  -  9:   3  < offset_speed_limit_assist <=  5 # rule for label 9
  - 10:   5  < offset_speed_limit_assist <=  7 # rule for label 10
  - 11:   7  < offset_speed_limit_assist <=  9 # rule for label 11
  - 12:   9  < offset_speed_limit_assist <= 11 # rule for label 12
  - 13:  11  < offset_speed_limit_assist <= 13 # rule for label 13
  - 14:  13  < offset_speed_limit_assist <= 15 # rule for label 14
```

Listing A.23: Specification for the Test Case *Speed Limit Assist Offset*

```
function:
  name:           steering_wheel_vibration  # name of function
  max_inputs:     30                        # maximum number of selected signals

  blacklist:                                # blacklisted signals
  - status_steering_wheel_vibration
  - config_steering_wheel_vibration

labeling:
  - 0: status_steering_wheel_vibration == 0 # rule for label 0: light
  - 1: status_steering_wheel_vibration == 1 # rule for label 1: medium
  - 2: status_steering_wheel_vibration == 2 # rule for label 2: strong
```

Listing A.24: Specification for the Test Case *Steering Wheel Vibration*

Appendix B

Deployment Class Specifications

```
# Test Cases of Type System Functions

deployment_class:
  function_class: system
```

Listing B.1: Deployment Class Test Cases in Section System Functions

```
# Test Cases of Type Context-Aware Functions and Anomaly Detection

deployment_class:
  function_class: group
```

Listing B.2: Deployment Class Test Cases in Context-Aware Functions and Anomaly Detection

Appendix C

Pre-Processing Specifications

```
# pre-processing setup for pre-processing and streaming evaluation

pre_processing:
  error-values:    # inclusion of error-values
    setting: yes

  scaling:         # scaling of all signals in [0,1]
    setting: on
    min:     0
    max:     1

  discretization:  # binning of signal values into discrete bins
    setting: off
```

Listing C.1: Pre-processing specification for pre-processing evaluation and streaming evaluation

```
# pre-processing setup for signal subset evaluation
# if feature selection algorithm can handle continuous and discrete data

pre_processing:
  error-values:    # inclusion of error-values
    setting: no

  scaling:         # scaling of all signals in [0,1]
    setting: on
    min:     0
    max:     1

  discretization:  # binning of signal values into discrete bins
    setting: off
```

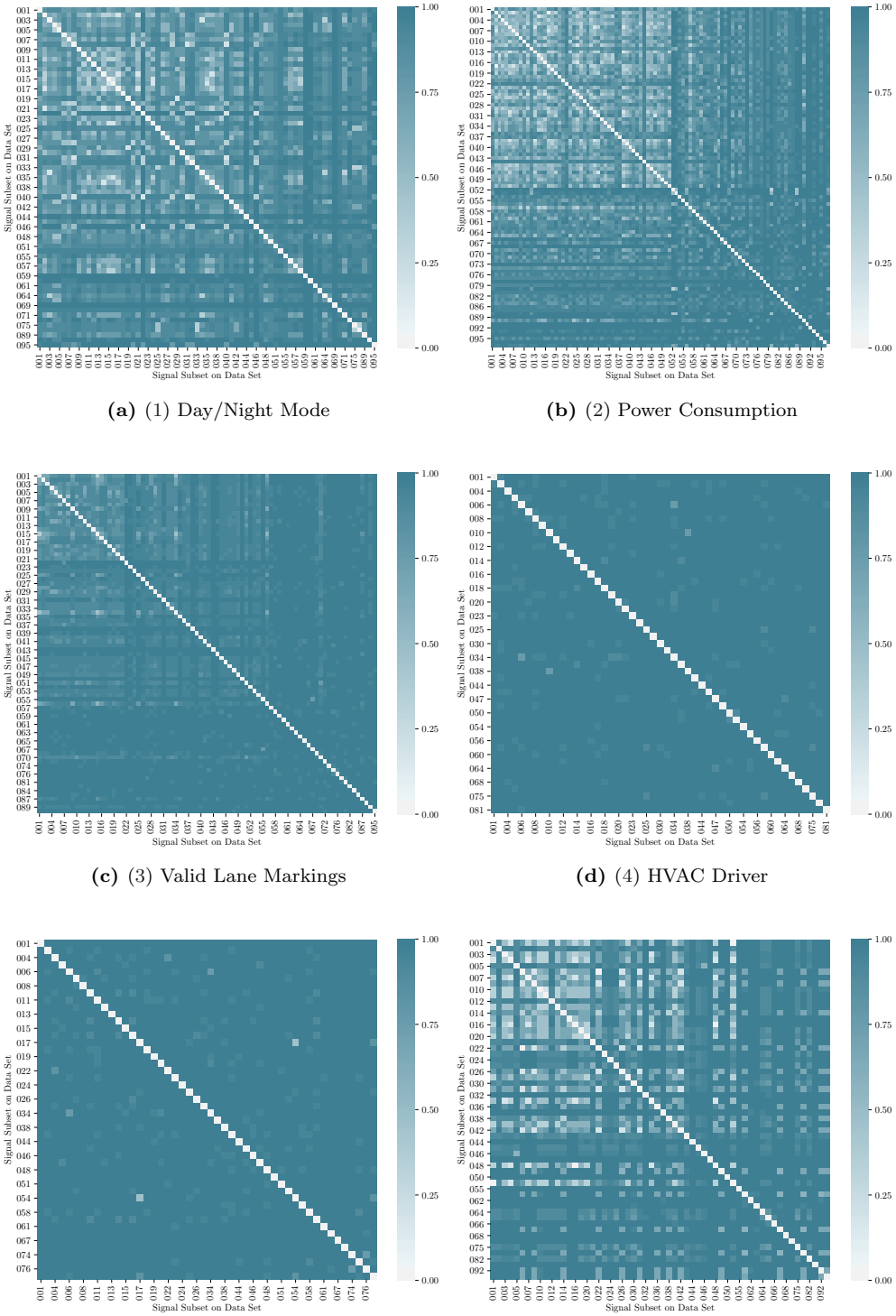
Listing C.2: Pre-processing specification for signal subset evaluation including continuous data

```
# pre-processing setup for signal subset evaluation
# if feature selection algorithm can only handle discrete data

pre_processing:
  error-values:    # inclusion of error-values
    setting: no

  scaling:         # scaling of all signals in [0,1]
    setting: on
    min:     0
    max:     1

  discretization:  # binning of signal values into discrete bins
    setting: on
    bins:    20
```

Listing C.3: Pre-processing specification for signal subset evaluation with only discrete data

Appendix D

Offline Evaluation



(a) (1) Day/Night Mode

(b) (2) Power Consumption

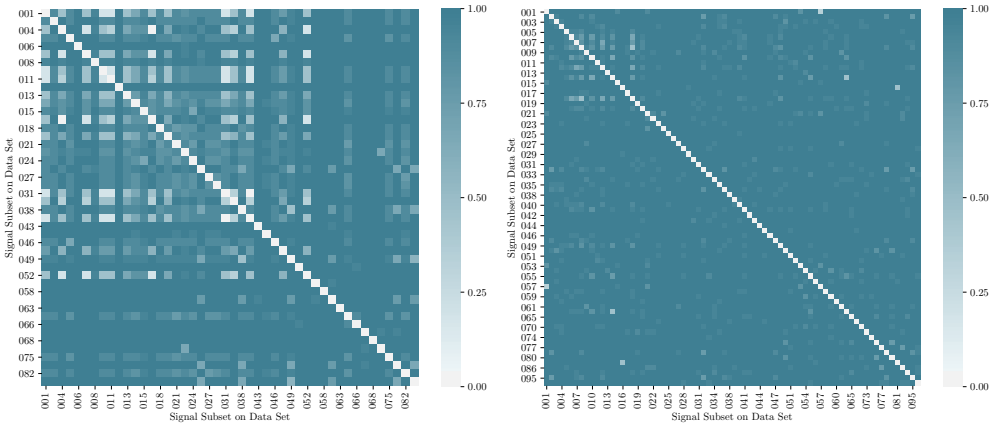
(c) (3) Valid Lane Markings

(d) (4) HVAC Driver

(e) (5) HVAC Co-Driver

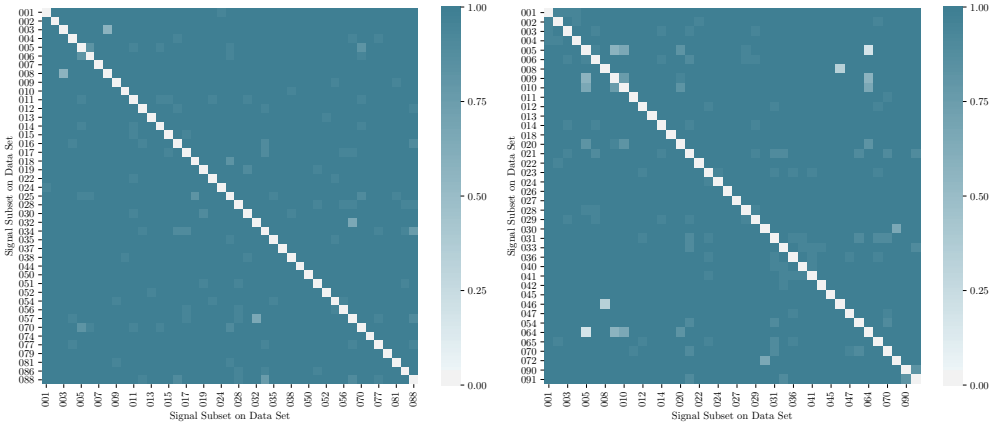
(f) (6) Proactive ACC

Figure D.1: Results of deployment evaluation (CHI^2) *(cont.)*



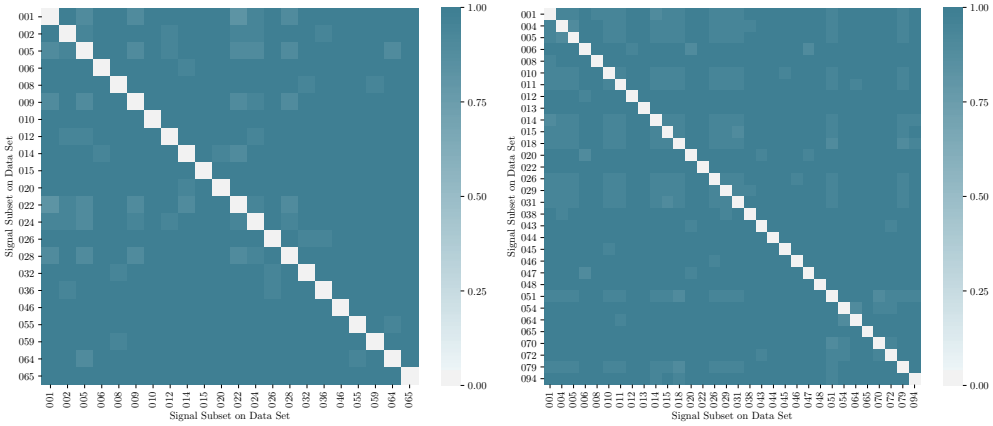
(g) (7) Proactive ACC Gap

(h) (8) Proactive Window Driver



(i) (9) Proactive Window Co-Driver

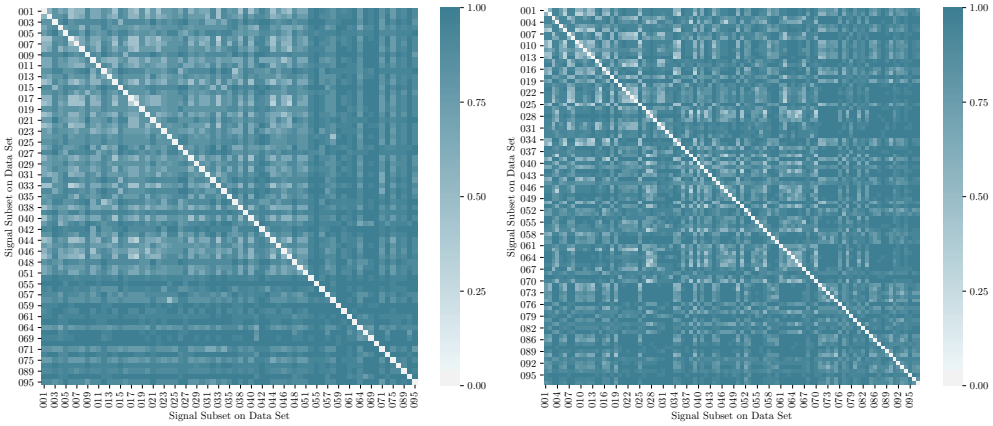
(j) (10) Proactive Seat Heating Driver



(k) (11) Proactive Seat Heating Co-Driver

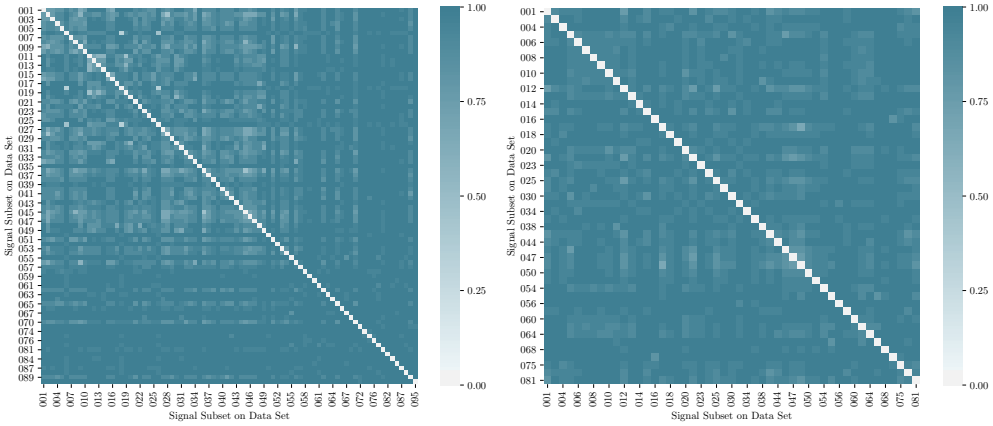
(l) (12) Proactive DEC

Figure D.1: Results of deployment evaluation (CHI^2)



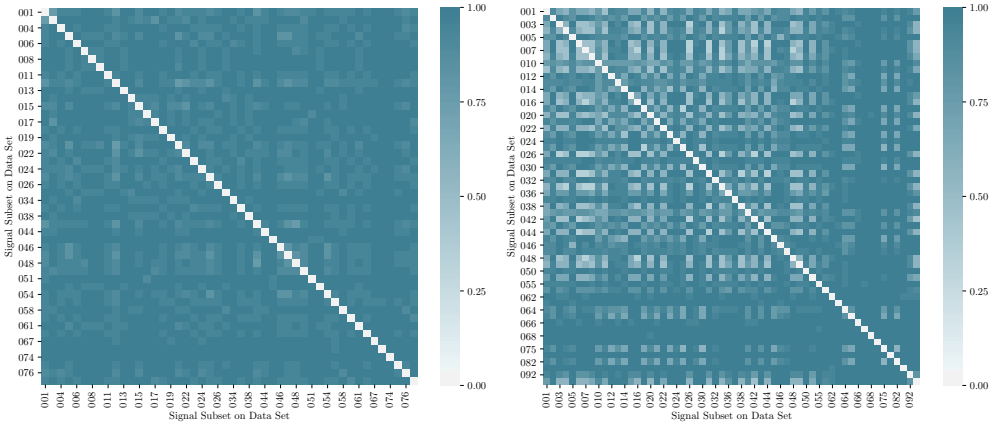
(a) (1) Day/Night Mode

(b) (2) Power Consumption



(c) (3) Valid Lane Markings

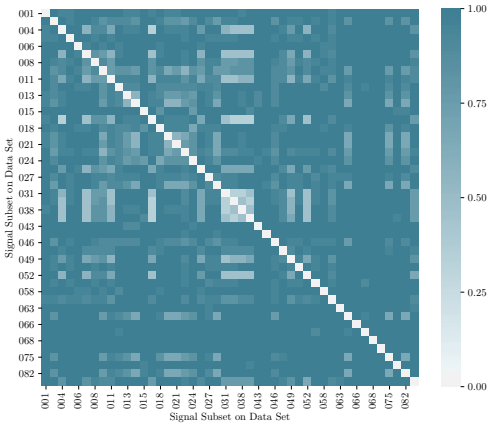
(d) (4) HVAC Driver



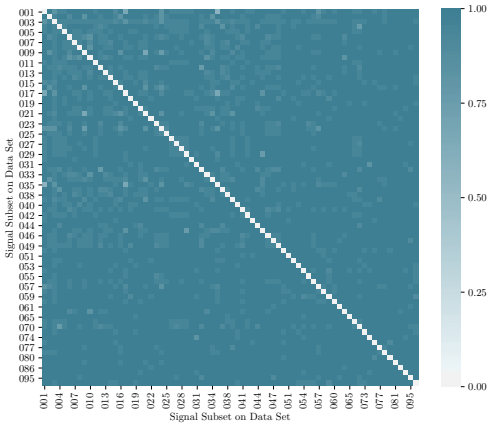
(e) (5) HVAC Co-Driver

(f) (6) Proactive ACC

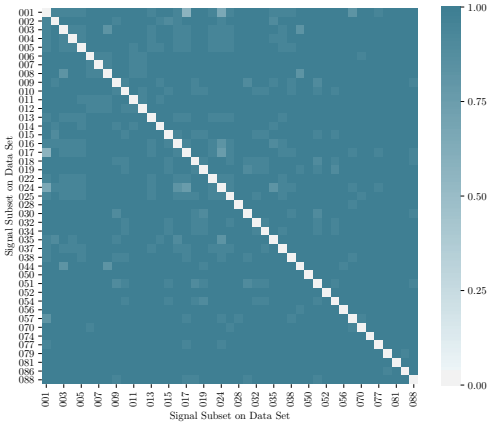
Figure D.2: Results of deployment evaluation (DISR) *(cont.)*



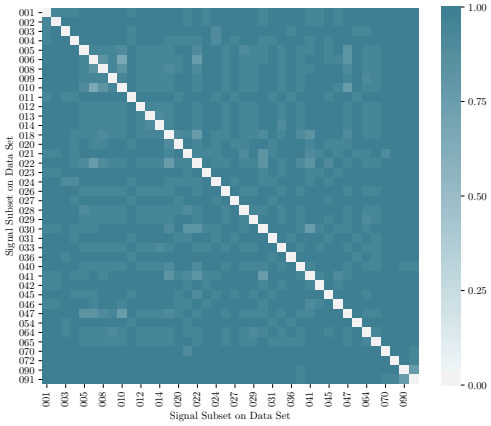
(g) (7) Proactive ACC Gap



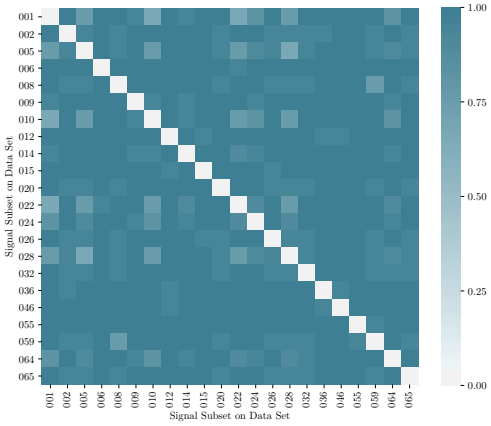
(h) (8) Proactive Window Driver



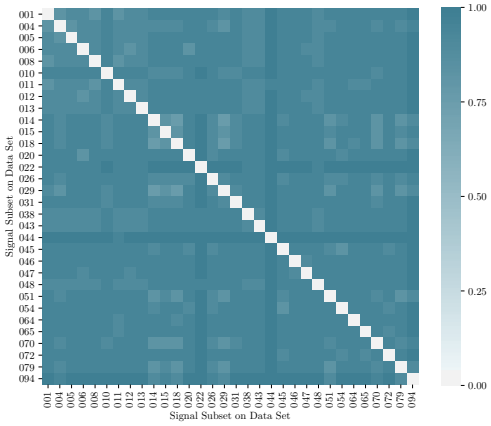
(i) (9) Proactive Window Co-Driver



(j) (10) Proactive Seat Heating Driver

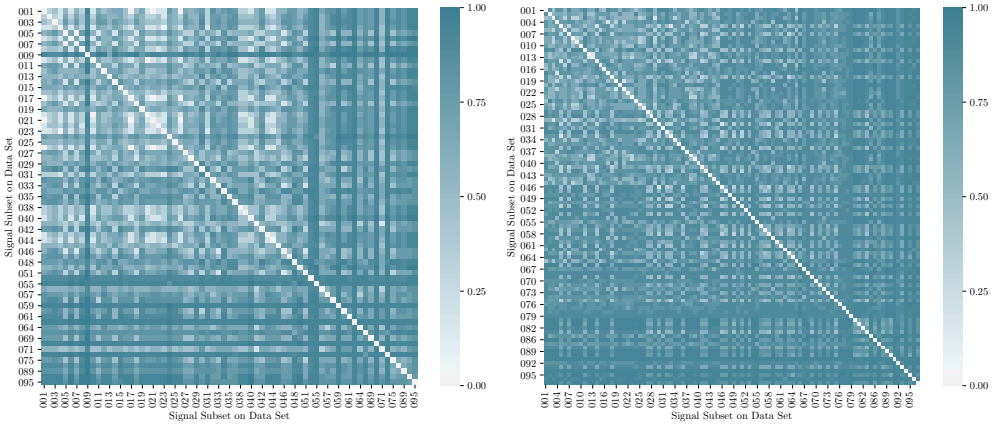


(k) (11) Proactive Seat Heating Co-Driver



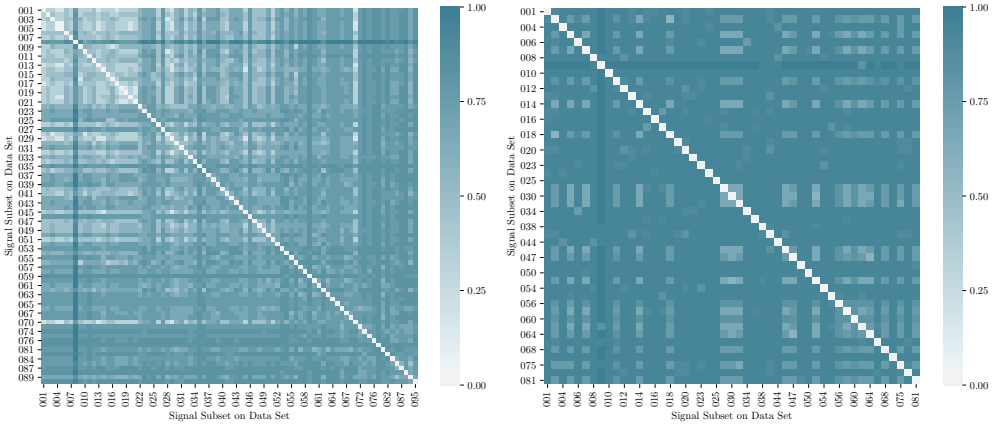
(l) (12) Proactive DEC

Figure D.2: Results of deployment evaluation (DISR)



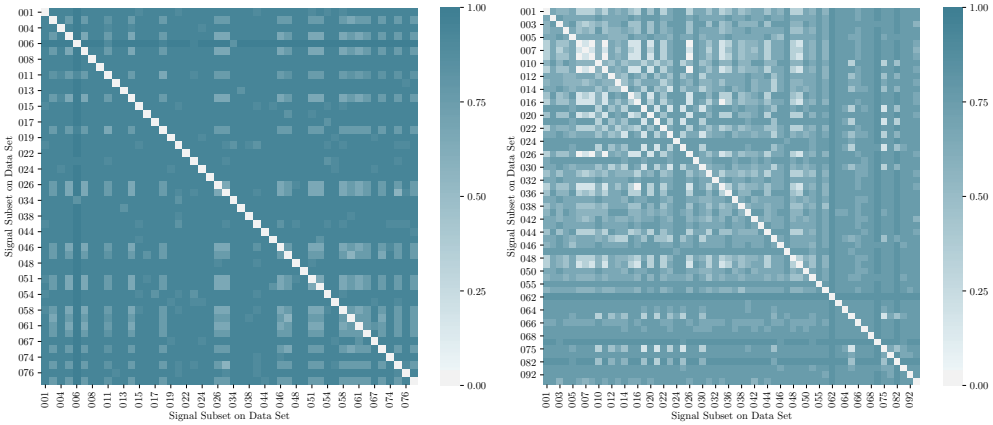
(a) (1) Day/Night Mode

(b) (2) Power Consumption



(c) (3) Valid Lane Markings

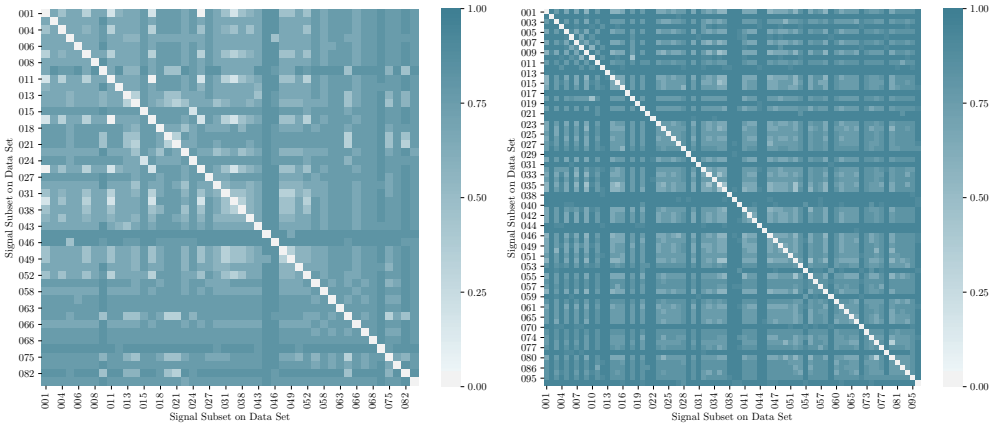
(d) (4) HVAC Driver



(e) (5) HVAC Co-Driver

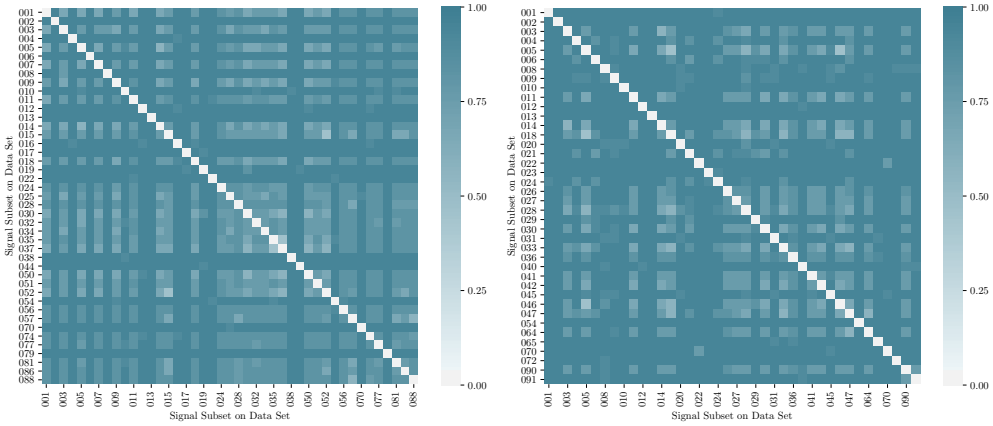
(f) (6) Proactive ACC

Figure D.3: Results of deployment evaluation (FScore) *(cont.)*



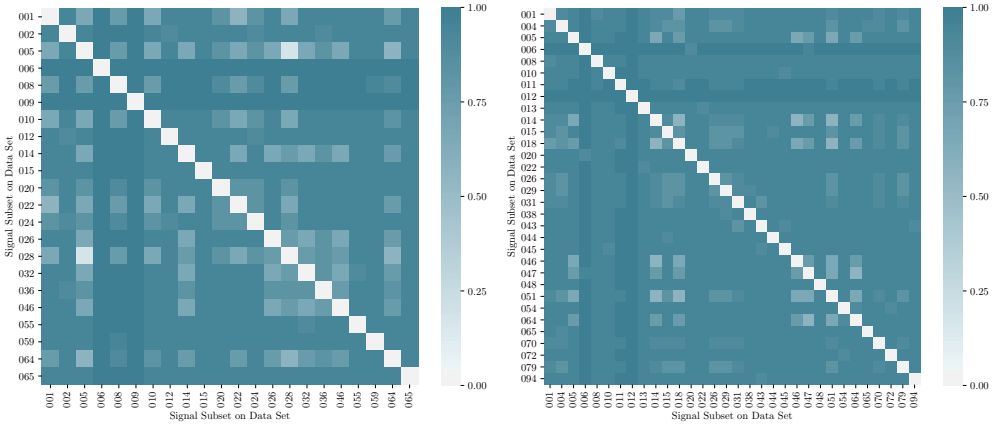
(g) (7) Proactive ACC Gap

(h) (8) Proactive Window Driver



(i) (9) Proactive Window Co-Driver

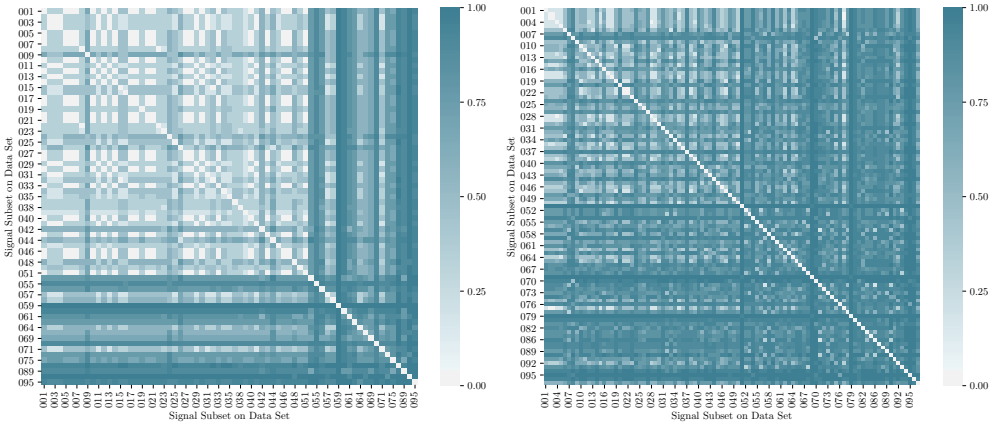
(j) (10) Proactive Seat Heating Driver



(k) (11) Proactive Seat Heating Co-Driver

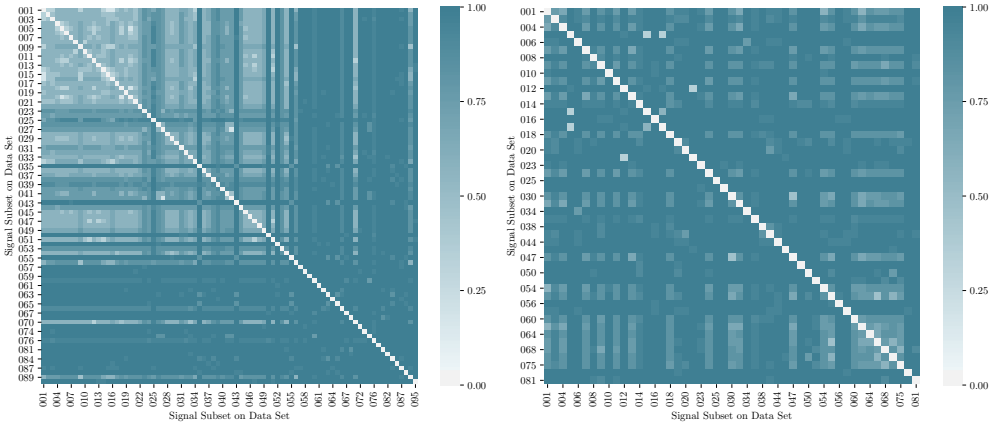
(l) (12) Proactive DEC

Figure D.3: Results of deployment evaluation (FScore)



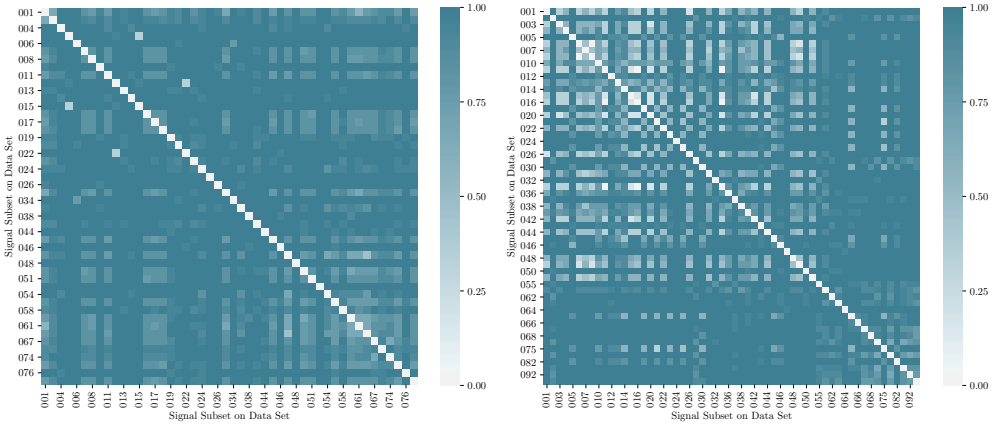
(a) (1) Day/Night Mode

(b) (2) Power Consumption



(c) (3) Valid Lane Markings

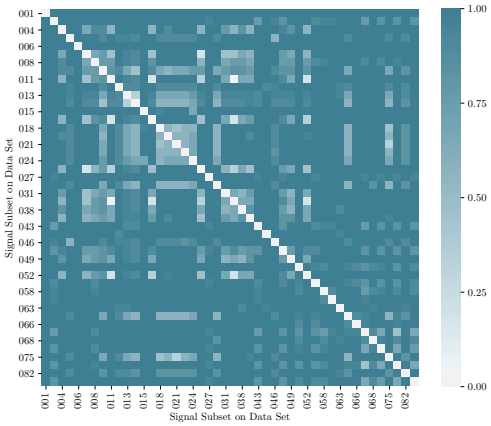
(d) (4) HVAC Driver



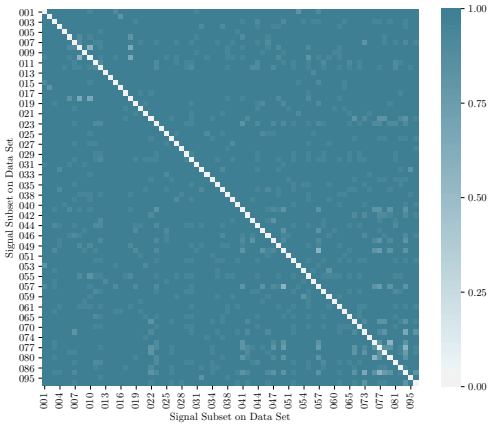
(e) (5) HVAC Co-Driver

(f) (6) Proactive ACC

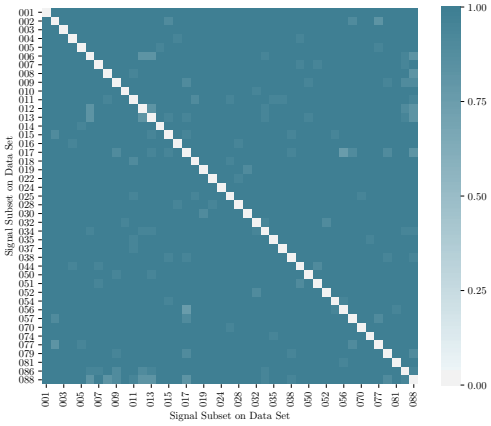
Figure D.4: Results of deployment evaluation (Gini Index) *(cont.)*



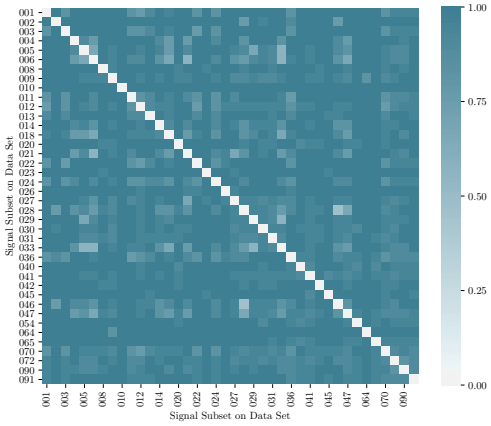
(g) (7) Proactive ACC Gap



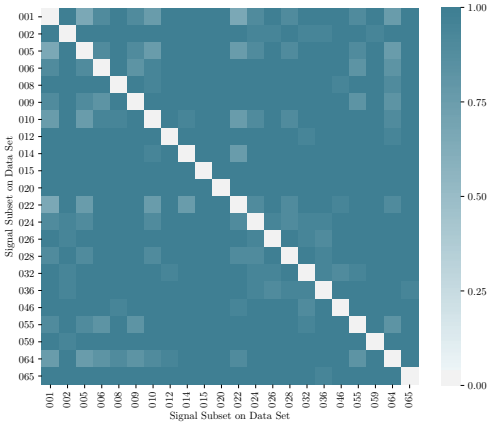
(h) (8) Proactive Window Driver



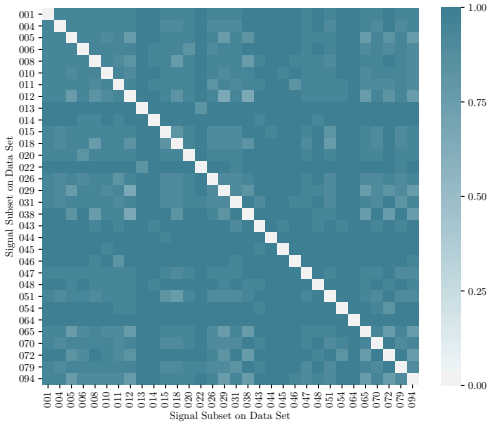
(i) (9) Proactive Window Co-Driver



(j) (10) Proactive Seat Heating Driver

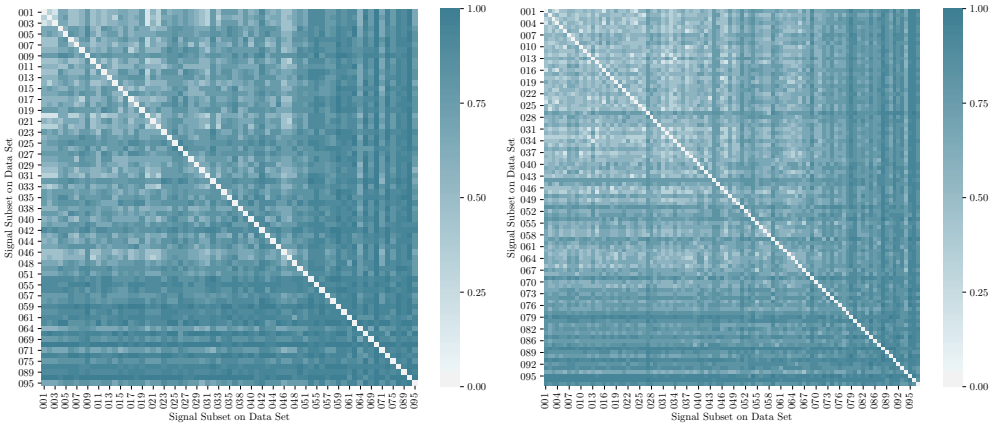


(k) (11) Proactive Seat Heating Co-Driver



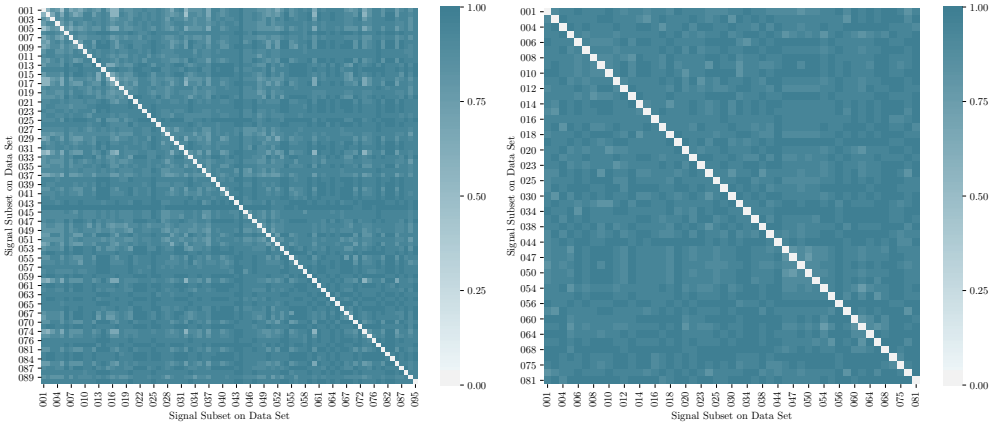
(l) (12) Proactive DEC

Figure D.4: Results of deployment evaluation (Gini Index)



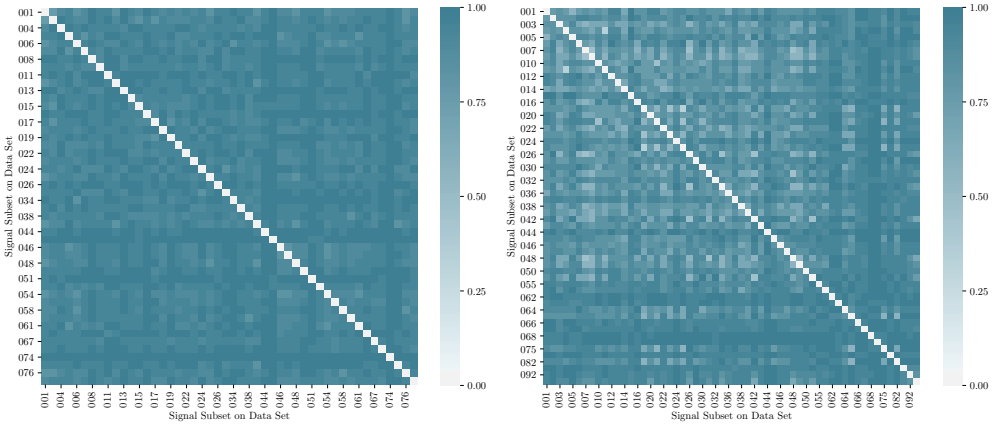
(a) (1) Day/Night Mode

(b) (2) Power Consumption



(c) (3) Valid Lane Markings

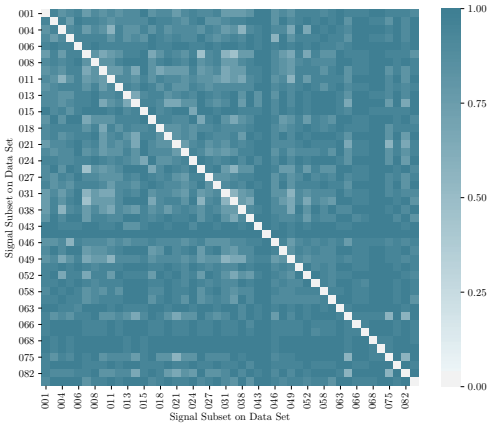
(d) (4) HVAC Driver



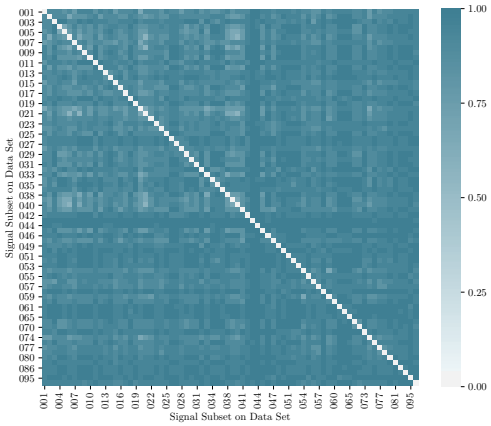
(e) (5) HVAC Co-Driver

(f) (6) Proactive ACC

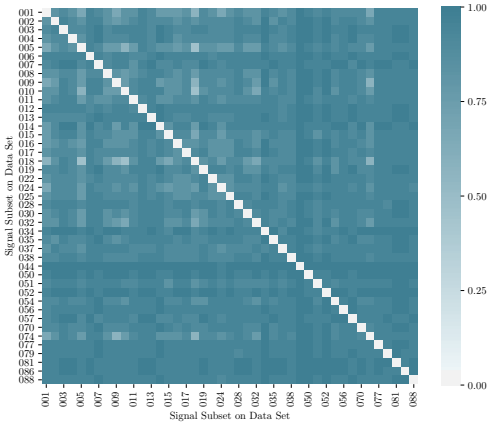
Figure D.5: Results of deployment evaluation (MRMR) *(cont.)*



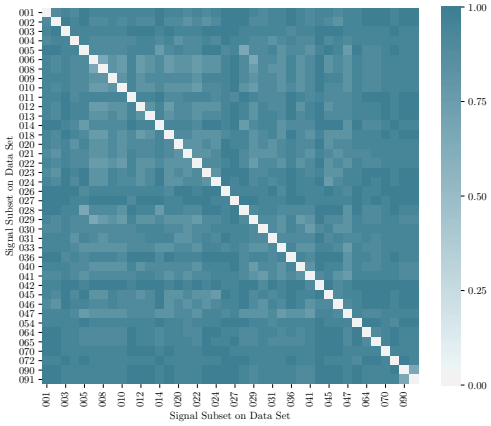
(g) (7) Proactive ACC Gap



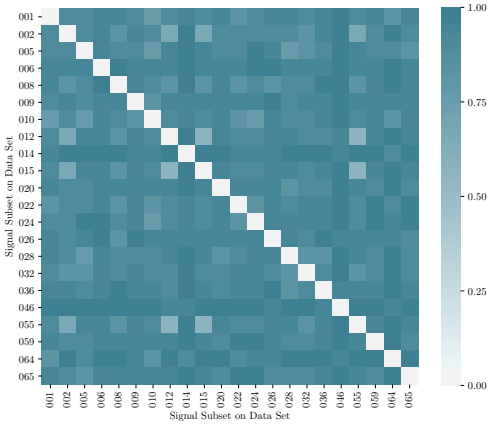
(h) (8) Proactive Window Driver



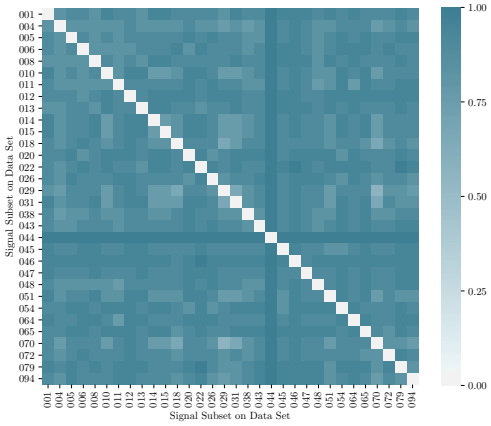
(i) (9) Proactive Window Co-Driver



(j) (10) Proactive Seat Heating Driver



(k) (11) Proactive Seat Heating Co-Driver



(l) (12) Proactive DEC

Figure D.5: Results of deployment evaluation (MRMR)

Appendix E

Streaming Evaluation

Table E.1: Usage of ADAS functions over different data sets

Data Set	Not Rec.	Rec.	Data Set	Not Rec.	Rec.	Data Set	Not Rec.	Rec.
1	11	1	**35**	12	-	**69**	9	3
2	10	2	**36**	9	3	**70**	12	-
3	11	1	**37**	9	3	**71**	10	2
4	12	-	**38**	12	-	**72**	12	-
5	12	-	**39**	12	-	**73**	12	-
6	11	1	**40**	12	-	**74**	12	-
7	9	3	**41**	12	-	**75**	9	3
8	11	1	**42**	11	1	**76**	12	-
9	12	-	**43**	9	3	**77**	12	-
10	12	-	**44**	12	-	**78**	12	-
11	11	1	**45**	9	3	**79**	11	1
12	11	1	**46**	12	-	**80**	12	-
13	6	6	**47**	12	-	**81**	12	-
14	6	6	**48**	7	5	**82**	12	-
15	12	-	**49**	11	1	**83**	12	-
16	12	-	**50**	10	2	**84**	12	-
17	12	-	**51**	12	-	**85**	12	-
18	10	2	**52**	11	1	**86**	12	-
19	9	3	**53**	12	-	**87**	12	-
20	12	-	**54**	12	-	**88**	12	-
21	12	-	**55**	11	1	**89**	12	-
22	11	1	**56**	12	-	**90**	12	-
23	10	2	**57**	12	-	**91**	10	2
24	12	-	**58**	10	2	**92**	12	-
25	9	3	**59**	12	-	**93**	12	-
26	9	3	**60**	12	-	**94**	12	-
27	12	-	**61**	12	-	**95**	12	-
28	12	-	**62**	12	-	**96**	12	-
29	12	-	**63**	11	1	**97**	12	-
30	12	-	**64**	12	-	**98**	12	-
31	5	7	**65**	12	-	**99**	12	-
32	11	1	**66**	8	4	**100**	12	-
33	11	1	**67**	9	3	**101**	12	-
34	11	1	**68**	12	-			

References

[1] George E. P. Box, Alberto Luceño, and Maria del Carmen Paniagua-Quinones. *Statistical Control by Monitoring and Adjustment*. Wiley series in probability and statistics. John Wiley & Sons, Hoboken, 2nd ed. edition, 2011.

[2] McKinsey & Company. **Ready for Inspection - The Aftermarket in 2030**, 2018.

[3] Richard Bellman. *Dynamic Programming*. Dover Books on Computer Science. Princeton University Press and Dover Publications, Princeton, NJ, USA, 1 edition, 1957.

[4] Christoph Segler, Stefan Kugele, and Alois Knoll. **Context Discovery for Personalised Automotive Functions**. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2470–2476. IEEE, 2019.

[5] Christoph Segler and Sina Shafaei. **Verfahren, Vorrichtung, Computerprogramm und Computerprogrammprodukt zur Datenbearbeitung für ein Fahrzeug**, Patent, Application DE 102018202348 A1, 2019.

[6] Philipp Obergfell, Christoph Segler, Eric Sax, and Alois Knoll. **Synchronization between Run-Time and Design-Time View of Context-Aware Automotive System Architectures**. In *2018 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–3. IEEE, 2018.

[7] Philipp Obergfell and Christoph Segler. **Method Indicating Unexpected Behaviour and Vehicle, System, and Storage Medium Comprising the Same**, Patent, Application WO 2020020437 A1, 2020.

[8] Christoph Segler, Stefan Kugele, Philipp Obergfell, Mohd Hefeez Osman, Sina Shafaei, Eric Sax, and Alois Knoll. **Evaluation of feature selection for anomaly detection in automotive E/E architectures**. In *ICSE '19 Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*, pages 260–261. ACM / IEEE, 2019.

[9] Christoph Segler, Stefan Kugele, Philipp Obergfell, Mohd Hafeez Osman, Sina Shafaci, Eric Sax, and Alois Knoll. **Anomaly Detection for Advanced**

Driver Assistance Systems Using Online Feature Selection. In *2019 IEEE Intelligent Vehicles Symposium*, pages 578–585. IEEE, 2019.

[10] Stefan Kugele, Christoph Segler, and Thomas Hubregtsen. **Architectural Patterns for Cross-Domain Personalised Automotive Functions**. In *IEEE International Conference on Software Architecture (ICSA 2020)*, pages 191–201. IEEE, 2020.

[11] Ilias Gerostathopoulos, Stefan Kugele, Christoph Segler, Tomas Bures, and Alois Knoll. **Automated Trainability Evaluation for Smart Software Functions**. In *IEEE/ACM International Conference on Automated Software Engineering (ASE 2019)*, pages 998–1001. IEEE, 2019.

[12] Thomas Hubregtsen, Christoph Segler, Josef Pichlmeier, Aritra Sarkar, Thomas Gabor, and Koen Bertels. **Integration and Evaluation of Quantum Accelerators for Data-Driven User Functions**. In *21st International Symposium on Quality Electronic Design (ISQED 2020)*, pages 329–334. IEEE, 2020.

[13] Stefan Kugele, Vadim Cebotari, Mario Gleirscher, Morteza Hashemi, Christoph Segler, Sina Shafaei, Hans-Jörg Vögel, Fridolin Bauer, Alois Knoll, Diego Marmsoler, and Hans-Ulrich Michel. **Research Challenges for a Future-Proof E/E Architecture - A Project Statement**. In *Informatik 2017*, GI-Edition Lecture Notes in Informatics Proceedings, pages 1463–1474. Gesellschaft für Informatik, Bonn, 2017.

[14] Philipp Obergfell, Stefan Kugele, Christoph Segler, Alois Knoll, and Eric Sax. **Continuous Software Engineering of Innovative Automotive Functions: An Industrial Perspective**. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 127–128. IEEE, 2019.

[15] Lukas Heinzmann, Sina Shafaei, Mohd Hafeez Osman, Christoph Segler, and Alois Knoll. **A Framework for Safety Violation Identification and Assessment in Autonomous Driving**. In *Proceedings of the Workshop on Artificial Intelligence Safety 2019 co-located with the 28th International Joint Conference on Artificial Intelligence, AISafety@IJCAI*. International Joint Conferences on Artificial Intelligence Organization, 2019.

[16] Christoph Segler. **Verfahren zum Steuern eines maschinellen Lernverfahrens einer Funktion eines Fahrzeugs, computerlesbares Medium, System, und Fahrzeug**, Patent, Application DE 102019119460 A1, 2021.

[17] Christoph Segler and Thomas Hubregtsen. **System und Verfahren zur Bereitstellung einer Systemfunktion eines Fahrzeugs**, Patent, Application DE 102020104479 A1, 2021.

[18] Christoph Segler and Hans-Jörg Vögel. **Verfahren, Vorrichtung, Computerprogramm und Computerprogrammprodukt zur Datenverarbeitung in einem Fahrzeug und Fahrzeug**, Patent, Application DE 102019117839 A1, 2021.

[19] Christoph Segler, Thomas Hubregtsen, and Emmanuel Pollakis. **System und Verfahren für ein Fortbewegungsmittel**, Patent, Application DE 102019127802 A1, 2021.

[20] Christoph Segler. **Verfahren zum Beheizen eines Kraftfahrzeugteils mit einer elektronischen Recheneinrichtung einer separaten Funktionseinheit sowie Kraftfahrzeug**, Patent, DE 102019114820 B3, 2020.

[21] Mohsen Kaboli and Christoph Segler. **Method, System, and Computer Program Product for Controlling a Movement of a Vehicle**, Patent, Application EP 3702865 A1, 2020.

[22] Alois Knoll, Emec Ercelik, Esra Icer, Burcu Karadeniz, Christoph Segler, Sina Shafaei, and Julian Tatsch. **1st International Workshop on Data Driven Intelligent Vehicle Applications (DDIVA) 2019: Co-located with 2019 IEEE Intelligent Vehicles Symposium (IV)**, 2019.

[23] Alois Knoll, Emec Ercelik, Esra Icer, Neslihan Kose, Burcu Karadeniz, Christoph Segler, Sina Shafaei, and Julian Tatsch. **2nd International Workshop on Data Driven Intelligent Vehicle Applications (DDIVA) 2020: Co-located with 2020 IEEE Intelligent Vehicles Symposium (IV)**, 2020.

[24] Alois Knoll, Sina Shafaei, Radoslaw Niewiadomski, Stefan Kugele, Christoph Segler, and Morteza Hashemi Farzaneh. **International Workshop on Machines with Emotions: Affect Modeling, Evaluation, and Challenges in Intelligent Cars: Co-located with 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, 2019.

[25] Jörg Schäuffele and Thomas Zurawka. *Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen.* ATZ / MTZ-Fachbuch. Springer Fachmedien Wiesbaden, Wiesbaden, 6. edition, 2016.

[26] T. Streichert and M. Traub. *Elektrik/Elektronik-Architekturen im Kraftfahrzeug: Modellierung und Bewertung von Echtzeitsystemen.* VDI-Buch. Springer Berlin Heidelberg, 1. edition, 2012.

[27] BMW Group. **The new BMW Group High Performance D3 platform: Data-Driven Development for Autonomous Driving**, 2019-03-27.

[28] Bo Li, Tianlei Zhang, and Tian Xia. **Vehicle Detection from 3D Lidar Using Fully Convolutional Network**. In *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016.

[29] Qijie Zhao, Tao Sheng, Yongtao Wang, Feng Ni, and Ling Cai. **CFENet: An Accurate and Efficient Single-Shot Object Detector for Autonomous Driving**. In *arXiv.org*. 2018.

[30] Victor Vaquero, Kai Fischer, Francesc Moreno-Noguer, Alberto Sanfeliu, and Stefan Milz. **Improving Map Re-localization with Deep 'Movable' Objects Segmentation on 3D LiDAR Point Clouds**. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 942–949. IEEE, 2019.

[31] Bharath Pattipati, Krishna Pattipati, Jon P. Christopherson, Setu Madhavi Namburu, Danil V. Prokhorov, and Liu Qiao. **Automotive battery management systems**. In *2008 IEEE AUTOTESTCON*, pages 581–586. IEEE, 2008.

[32] David Antory. **Application of a data-driven monitoring technique to diagnose air leaks in an automotive diesel engine: A case study**. *Mechanical Systems and Signal Processing*, **21**(2):795–808, 2007.

[33] J.-S. Chiou and M.-T. Liu. **Using fuzzy logic controller and evolutionary genetic algorithm for automotive active suspension system**. *International Journal of Automotive Technology*, **10**(6):703–710, 2009.

[34] Jianhui Luo, Madhavi Namburu, Krishna R. Pattipati, Liu Qiao, and Shunsuke Chigusa. **Integrated Model-Based and Data-Driven Diagnosis of Automotive Antilock Braking Systems**. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, **40**(2):321–336, 2010.

[35] Setu Madhavi Namburu, Shunsuke Chigusa, Danil Prokhorov, Liu Qiao, Kihoon Choi, and Krishna Pattipati. **Application of an Effective Data-Driven Approach to Real-time time Fault Diagnosis in Automotive Engines**. In *2007 IEEE Aerospace Conference*, pages 1–9. IEEE, 2007.

[36] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. **Towards a Better Understanding of Context and Context-Awareness**. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Hans-W Gellersen, editors, *Handheld and Ubiquitous Computing*, **1707** of *Lecture Notes in Computer Science*, pages 304–307. Springer, Berlin, Heidelberg, 1999.

[37] Hee Eon Byun and Keith Chaverst. **Utilizing Context History to Provide Dynamic Adaptations**. *Applied Artificial Intelligence*, **18**(6):533–548, 2004.

[38] Sandro Rodriguez Garzon. **Intelligent In-Car-Infotainment Systems: A Contextual Personalized Approach**. In *2012 8th International Conference on Intelligent Environments (IE)*, pages 315–318, 2012.

[39] Deborah Höltje. *Developing a Smart Wiper System Adaptive to the Drivers' Behavior based on Support Vector Machines (SVM).* Bachelor's thesis, Technische Universität München, München, 2018.

[40] Nikolaos Tzioras. *A Case Study of Reinforcement Learning-based Approaches for Proactive Comfort Functions in Cars.* Master's thesis, Technische Universität München, München, 2018.

[41] M. Canale, S. Malan, and V. Murdocco. **Personalization of ACC Stop and Go Task Based on Human Driver Behaviour Analysis**. *IFAC Proceedings Volumes*, **35**(1):357–362, 2002.

[42] Avi Rosenfeld, Zevi Bareket, Claudia V. Goldman, Sarit Kraus, David J. LeBlanc, and Omer Tsimhoni. **Learning Driver's Behavior to Improve the Acceptance of Adaptive Cruise Control**. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 2317–2322, 2012.

[43] Stéphanie Lefèvre, Ashwin Carvalho, Yiqi Gao, H. Eric Tseng, and Francesco Borrelli. **Driver models for personalised driving assistance**. *Vehicle System Dynamics*, **53**(12):1705–1720, 2015.

[44] Yoshihiro Nishiwaki, Chiyomi Miyajima, Norihide Kitaoka, Katsunobu Itou, and Kazuya Takeda. **Generation of Pedal Operation Patterns of Individual Drivers in Car-Following for Personalized Cruise Control**. In *2007 IEEE Intelligent Vehicles Symposium*, pages 823–827, 2007.

[45] H. Tan and B. J. Tedesco. **Personalized driver assistance system for vehicle**, Patent, US9623878 B2, 2017.

[46] Varun Chandola, Arindam Banerjee, and Vipin Kumar. **Anomaly detection**. *ACM Computing Surveys*, **41**(3):1–58, 2009.

[47] David L. Iverson. **Inductive system health monitoring**. In *International Conference on Artificial Intelligence*, 2004.

[48] Mooi Choo Chuah and Fen Fu. **ECG Anomaly Detection via Time Series Analysis**. In *Proceedings of the 2007 International Conference on Frontiers of High Performance Computing and Networking*, ISPA'07, pages 123–135, Berlin, Heidelberg, 2007. Springer-Verlag.

[49] Michael Müter, André Groll, and Felix C. Freiling. **A structured approach to anomaly detection for in-vehicle networks**. In *6th International Conference on Information Assurance and Security, IAS 2010*, pages 92–98. IEEE, 2010.

[50] Michael Müter and Naim Asaj. **Entropy-based anomaly detection for in-vehicle networks**. In *Intelligent Vehicles Symposium (IV) 2011*, pages 1110–1115. IEEE, 2011.

[51] Roberto Baldoni, Luca Montanari, and Marco Rizzuto. **On-line failure prediction in safety-critical systems**. *Future Generation Computer Systems*, **45**:123–132, 2015.

[52] Adrian Taylor, Nathalie Japkowicz, and Sylvain Leblanc. **Frequency-based anomaly detection for the automotive CAN bus**. In *2015 World Congress on Industrial Control Systems Security, WCICSS 2015*, pages 45–49. IEEE, 2015.

[53] Troy Hunt. **Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs**, 2016 [Accessed 2020-09-24].

[54] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. **Software Engineering for Machine Learning: A Case Study**. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300. IEEE, 2019.

[55] Mathew Salvaris, Danielle Dean, and Wee Hyong Tok. **Microsoft AI Platform**. In Mathew Salvaris, Danielle Dean, and Wee Hyong Tok, editors, *Deep Learning with Azure*, pages 79–98. Apress, Berkeley, CA, 2018.

[56] Charles Hill, Rachel Bellamy, Thomas Erickson, and Margaret Burnett. **Trials and tribulations of developers of intelligent systems: A field study**. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 162–170. IEEE, 2016.

[57] Google Cloud. **Machine learning workflow**, 2020 [Accessed 2019-09-27].

[58] Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. **Investigating statistical machine learning as a tool for software development**. In Mary Czerwinski, Arnie Lund, and Desney Tan, editors, *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 667, New York, New York, USA, 2008. ACM Press.

[59] Microsoft Azure. **The Team Data Science Process**, 2020 [Accessed 2020-02-03].

[60] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. **The KDD process for extracting useful knowledge from volumes of data**. *Communications of the ACM*, **39**(11):27–34, 1996.

[61] Rüdiger Wirth and Jochen Hipp. **Crisp-dm: towards a standard process modell for data mining**. In *Proc. 4th Intl. Conference on Practical Applications of Knowledge Discovery and Data mining*, 2000.

[62] Matthias Traub. *Durchgängige Timing-Bewertung von Vernetzungsarchitekturen und Gateway-Systemen im Kraftfahrzeug*. PhD thesis, Karlsruher Institut für Technologie, Karlsruhe, 2010.

[63] Lisa Braun. *Modellbasierte Design-Space-Exploration nicht-funktionaler Auslegungskriterien des Fahrzeugenergiebordnetzes*. PhD thesis, Karlsruher Institut für Technologie, 2018.

[64] Julian Weber. *Automotive Development Processes: Processes for Successful Customer Oriented Vehicle Development*. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2009.

[65] Johannes Bach. *Methoden und Ansätze für die Entwicklung und den Test prädiktiver Fahrzeugregelungsfunktionen*. PhD thesis, Karlsruher Institut für Technologie, Karlsruhe, 2018.

[66] Julian Broy. *Modellbasierte Entwicklung und Optimierung flexibler zeitgesteuerter Architekturen im Fahrzeugserienbereich*. PhD thesis, Karlsruher Institut für Technologie, Karlsruhe, 2010.

[67] Waldemar Haas and P. Langjahr. **Cross-domain vehicle control units in modern E/E architectures**. In Michael Bargende, Hans-Christian Reuss, and Jochen Wiedemann, editors, *16. Internationales Stuttgarter Symposium*, Proceedings, pages 1619–1627. Springer Fachmedien Wiesbaden, Wiesbaden, 2016.

[68] Dominik Reinhardt and Markus Kucera. **Domain Controlled Architecture - A New Approach for Large Scale Software Integrated Automotive Systems**. In *Proceedings of the 3rd International Conference on Pervasive Embedded Computing and Communication Systems*, pages 221–226. SciTePress - Science and and Technology Publications, 2013.

[69] Manfred Broy. **Challenges in automotive software engineering**. In Leon J. Osterweil, Dieter Rombach, and Mary Lou Soffa, editors, *Proceeding of the 28th international conference on Software engineering - ICSE '06*, page 33, New York, New York, USA, 2006. ACM Press.

[70] ISO 11898. **Road Vehilces - Controller Area Network (CAN)**, 2015.

[71] ISO 17987. **Road Vehicles - Local Interconnect Network (LIN)**, 2016.

[72] ISO 17458. **Road Vehicles - FlexRay Communication System**, 2013.

[73] Werner Zimmermann and Ralf Schmidgall. *Bussysteme in der Fahrzeugtechnik*. Springer Fachmedien, Wiesbaden, 2014.

[74] ISO 19157. **Geographic information – Data quality**, 2013.

[75] Benjamin Klotz, Raphaël Troncy, Daniel Wilms, and Christian Bonnet. **VSSo: The Vehicle Signal and Attribute Ontology**. In *9th International Semantic Sensor Networks Workshop @ International Semantic Web Conference*, pages 56–63, 2018.

[76] AUTOSAR. **SOME/IP Protocol Specification: Release 1.0.0**, 2016-11-30.

[77] Verband der Automobilindustrie. **Access to the vehicle and vehicle generated data - NEVADA Share and Secure Concept**, 2017 [Accessed 2019-11-19].

[78] ISO 20077. **Road Vehicles — Extended vehicle (ExVe) methodology**, 2017.

[79] ISO 20078. **Road vehicles — Extended vehicle (ExVe) web services**, 2019.

[80] BMW Group. **CarData**, 2017 [Accessed 2020-08-14].

[81] Daimler AG. **Mercedes-Benz Kunden profitieren von neuen Datendiensten**, 2018-12-14 [Accessed 2020-07-12].

[82] Javier Gozalvez. **Samsung Electronics Sets 5G Speed Record at 7.5 Gb/s**. *IEEE Vehicular Technology Magazine*, **10**(1):12–16, 2015.

[83] Hasan Esen, Masakazu Adachi, Daniele Bernardini, Alberto Bemporad, Dominik Rost, and Jens Knodel. **Control as a service (CaaS)**. In Alberto Sangiovanni-Vincentelli, editor, *Proceedings of the Second International Workshop on the Swarm at the Edge of the Cloud - SWEC '15*, pages 13–18, New York, New York, USA, 2015. ACM Press.

[84] GENIVI. **Vehicle Signal Specification: Release 1.0**, 2020.

[85] W3C. **Vehicle Information Service Specification: W3C Candidate Recommendation**, 2018-02-13.

[86] W3C. **Vehicle Information API Specification: W3C Working Group Note**, 2018-06-26.

[87] Alexandre Armand, David Filliat, and Javier Ibanez-Guzman. **Ontology-based context awareness for driving assistance systems**. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 227–233. IEEE, 2014.

[88] Daniel Lüddecke, Nina Bergmann, and Ina Schaefer. **Ontology-Based Modeling of Context-Aware Systems**. In Juergen Dingel, Wolfram Schulte, Isidro Ramos, Silvia Abrahão, and Emilio Insfran, editors, *Model-Driven Engineering Languages and Systems*, **8767** of *Lecture Notes in Computer Science*, pages 484–500. Springer International Publishing, Cham, 2014.

[89] Saravanan Kannan, Arunkumar Thangavelu, and Rameshbabu Kalivaradhan. **An Intelligent Driver Assistance System (I-DAS) for Vehicle Safety Modelling using Ontology Approach**. In *International Journal of UbiComp (IJU)*. UbiComp, 2010.

[90] Zhitao Xiong, Vinayak V. Dixit, and S. Travis Waller. **The development of an Ontology for driving Context Modelling and reasoning**. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 13–18. IEEE, 2016.

[91] Lihua Zhao, Ryutaro Ichise, Seiichi Mita, and Yutaka Sasaki. **Core Ontologies for Safe Autonomous Driving**. In Serena Villata, Jeff Z. Pan, and Mauro Dragoni, editors, *Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015)*, **1486** of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[92] Benjamin Klotz, Soumya Kanti Datta, Daniel Wilms, Raphael Troncy, and Christian Bonnet. **A Car as a Semantic Web Thing: Motivation and Demonstration**. In *2018 Global Internet of Things Summit (GIoTS)*, pages 1–6. IEEE, 2018.

[93] Mark Andrew Hall. *Correlation-based feature selection for machine learning*. PhD thesis, University of Waikato, Hamilton, New Zealand, 1999.

[94] Avrim L. Blum and Pat Langley. **Selection of relevant features and examples in machine learning**. *Artificial Intelligence*, **97**(1-2):245–271, 1997.

[95] Isabelle Guyon and André Elisseeff. **An introduction to variable and feature selection**. *J. Mach. Learn. Res.*, **3**(7-8):1157–1182, 2003.

[96] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. **Gene Selection for Cancer Classification using Support Vector Machines**. *Machine Learning*, **46**(1/3):389–422, 2002.

[97] Kamalika Das, Kanishka Bhaduri, and Hillol Kargupta. **A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks**. *Knowledge and Information Systems*, **24**(3):341–367, 2010.

[98] Bernhard Schlegel and Bernhard Sick. **Design and optimization of an autonomous feature selection pipeline for high dimensional, heterogeneous feature spaces**. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–9, 2016.

[99] Ville Könönen, Jani Mäntyjärvi, Heidi Similä, Juha Pärkkä, and Miikka Ermes. **Automatic feature selection for context recognition in mobile devices**. *Pervasive and Mobile Computing*, **6**(2):181–197, 2010.

[100] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. **Feature Selection: A Data Perspective**. *ACM Computing Surveys*, **50**(6):1–45, 2018.

[101] Girish Chandrashekar and Ferat Sahin. **A survey on feature selection methods**. *Computers & Electrical Engineering*, **40**(1):16–28, 2014.

[102] Kenji Kira and Larry A. Rendell. **A Practical Approach to Feature Selection**. In *Machine Learning Proceedings 1992*, pages 249–256. Elsevier, 1992.

[103] Daphne Koller and Mehran Sahami. **Toward Optimal Feature Selection**. In Lorenza Saitta, editor, *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 284–292. Morgan Kaufmann, 1996.

[104] N. Kwak and Chong-Ho Choi. **Input feature selection by mutual information based on Parzen window**. *IEEE transactions on pattern analysis and machine intelligence*, **24**(12):1667–1671, 2002.

[105] Xiaofei He, Deng Cai, and Partha Niyogi. **Laplacian Score for Feature Selection**. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, pages 507–514, Cambridge, MA, USA, 2005. MIT Press.

[106] Quanquan Gu, Zhenhui Li, and Jiawei Han. **Generalized Fisher Score for Feature Selection**. In *arxiv.org*. 2012.

[107] Zheng Zhao and Huan Liu. **Spectral feature selection for supervised and unsupervised learning**. In Zoubin Ghahramani, editor, *Proceedings of the 24th international conference on Machine learning - ICML '07*, pages 1151–1157, New York, New York, USA, 2007. ACM Press.

[108] Mahdokht Masaeli, Yan Yan, Ying Cui, Glenn Fung, and Jennifer G. Dy. **Convex Principal Feature Selection**. In Srinivasan Parthasarathy, Bing Liu, Bart Goethals, Jian Pei, and Chandrika Kamath, editors, *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 619–628. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2010.

[109] Ahmed K. Farahat, Ali Ghodsi, and Mohamed S. Kamel. **An Efficient Greedy Method for Unsupervised Feature Selection**. In *2011 IEEE 11th International Conference on Data Mining*, pages 161–170. IEEE, 2011.

[110] Iñaki Inza, Basilio Sierra, and Rosa Blanco. **Gene selection by sequential search wrapper approaches in microarray cancer class prediction**. *Journal of Intelligent and Fuzzy Systems*, **12**(1):25–33, 2002.

[111] Yonghong Peng, Zhiqing Wu, and Jianmin Jiang. **A novel feature selection approach for biomedical data classification**. *Journal of biomedical informatics*, **43**(1):15–23, 2010.

[112] Jinjie Huang, Yunze Cai, and Xiaoming Xu. **A hybrid genetic algorithm for feature selection wrapper based on mutual information**. *Pattern Recognition Letters*, **28**(13):1825–1844, 2007.

[113] A. Verikas and M. Bacauskiene. **Feature selection with neural networks**. *Pattern Recognition Letters*, **23**(11):1323–1335, 2002.

[114] Claude Sammut and Geoffrey I. Webb. **Principal Component Analysis**. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, page 1006. Springer US, Boston, MA, 2017.

[115] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 2002.

[116] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. **Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion**. *Journal of Machine Learning Research*, **2010**(11):3371–3408, 2010.

[117] Xindong Wu, Kui Yu, Wei Ding, Hao Wang, and Xingquan Zhu. **Online feature selection with streaming features**. *IEEE transactions on pattern analysis and machine intelligence*, **35**(5):1178–1192, 2013.

[118] Ming Yuan and Yi Lin. **Model selection and estimation in regression with grouped variables**. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**(1):49–67, 2006.

[119] Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. **Structured Variable Selection with Sparsity-Inducing Norms**. *Journal of Machine Learning Research*, **2011**(12):2777–2824, 2011.

[120] Junzhou Huang, Tong Zhang, and Dimitri Metaxas. **Learning with structured sparsity**. In *Proceedings of the 26th International Conference On Machine Learning, ICML 2009*, pages 417–424, 2009.

[121] Seyoung Kim and Eric P. Xing. **Tree-guided Group Lasso for Multi-task Regression with Structured Sparsity**. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 543–550, USA, 2010. Omnipress.

[122] Dahua Lin and Xiaoou Tang. **Conditional Infomax Learning: An Integrated Framework for Feature Extraction and Fusion**. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 68–82, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[123] François Fleuret. **Fast Binary Feature Selection with Conditional Mutual Information**. *Journal of Machine Learning Research*, **2004**(5):1531–1555, 2004.

[124] Huan Liu and Rudy Setiono. **Chi2: feature selection and discretization of numeric attributes**. In Anon, editor, *Proceedings of the International Conference on Tools with Artificial Intelligence*, pages 388–391. IEEE, 1995.

[125] David Hutchison. **On the Use of Variable Complementarity for Feature Selection in Cancer Classification**. In *Applications of Evolutionary Computing*, **3907** of *Lecture Notes in Computer Science*, pages 91–102. Springer, Berlin, Heidelberg, 2006.

[126] Lei Yu and Huan Liu. **Feature selection for high-dimensional data: A fast correlation-based filter solution**. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.

[127] Sewall Wright. **The Interpretation of Population Structure by F-Statistics with Special Regard to Systems of Mating**. *Evolution*, **19**(3):395, 1965.

[128] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. A Wiley-Interscience publication. Wiley, New York NY u.a., 2. ed. edition, 2001.

[129] C. W. Gini. **Variability and mutability, contribution to the study of statistical distribution and relaitons**. *Studi Economico-Giuricici della R*, 1912.

[130] Aleks Jakulin. *Machine learning based on attribute interactions*. PhD thesis, Univerza v Ljubljani, 2005.

[131] Howard Hua Yang and John Moody. **Data visualization and feature selection: New algorithms for nongaussian data**. In *Advances in Neural Information Processing Systems*, pages 687–693, 2000.

[132] R. Battiti. **Using mutual information for selecting features in supervised neural net learning**. *IEEE transactions on neural networks*, **5**(4):537–550, 1994.

[133] David D. Lewis. **Feature selection and feature extraction for text categorization**. In *Proceedings of the workshop on Speech and Natural Language - HLT '91*, page 212, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[134] Hanchuan Peng, Fuhui Long, and Chris Ding. **Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy**. *IEEE transactions on pattern analysis and machine intelligence*, **27**(8):1226–1238, 2005.

[135] Marko Robnik-Šikonja and Igor Kononenko. **Theoretical and Empirical Analysis of ReliefF and RReliefF**. *Machine Learning*, **53**(1/2):23–69, 2003.

[136] Feiping Nie, Shiming Xiang, Yangqing Jia, Changshui Zhang, and Shuicheng Yan. **Trace ratio criterion for feature selection**. In *AAAI*, pages 671–676, 2008.

[137] Anind K. Dey. *Providing architectural support for building context-aware applications*. PhD thesis, College of Computing, Georgia Institute of Technology, 2000.

[138] Oren Ben-Kiki, Clark Evans, and Ingy döt Net. **YAML Ain't Markup Language (YAML) - Version 1.2: 3rd Edition**, 2009-10-01.

[139] ISO/IEC 21778. **Information technology — The JSON data interchange syntax**, 2017.

[140] Piotr Juszczak, D. Tax, and Robert P. W. Duin. **Feature scaling in support vector data description**. In *Proceedings of ASCI*, 2002.

[141] Sotiris Kotsiantis and Dimitris Kanellopoulos. **Discretization Techniques: A recent survey**. In *GESTS International Transactions on Computer Science and Engineering, Vol.32 (1)*. GESTS, 2006.

[142] James Dougherty, Ron Kohavi, and Mehran Sahami. **Supervised and Unsupervised Discretization of Continuous Features**. In *Machine Learning Proceedings 1995*, pages 194–202. Elsevier, 1995.

[143] Association for Standardisation of Automation and Measuring Systems. **ASAM MCD-2 NET (FIBEX)**, 2017.

[144] Michael Sedlmair. *Visual Analysis of In-Car Communication Networks*. PhD thesis, Ludwig-Maximilians-Universität, München, 2010.

[145] AUTOSAR. **E2E Protocol Specification: Release 1.3.0**, 2017-12-08.

[146] David Money Harris and Sarah L. Harris. *Digital design and computer architecture*. Morgan Kaufmann, Waltham, MA, 2nd ed. edition, 2013.

[147] IEEE. **IEEE Standard for Information Technology - Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7**, 2017.

[148] Kenneth P. Bowman. *An introduction to programming with IDL: Interactive Data Language*. Elsevier Academic Press, Amsterdam and Boston, 2006.

[149] EU. **General Data Protection Regulation: 2016/679**, 2016.

[150] California State Legislature. **California Consumer Privacy Act: AB-375**, 2018.

[151] Cédric Bray, Christopher D. Moore, Patrick S. Piemonte, Emanuele Vulcano, Marcel van Os, Billy P. Chen, Seejo K. Pylappan, and Justin O'Beirne. **Night mode**, Patent, US9536325B2, 2014.

[152] Xuewei Qi, Yadan Luo, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew J. Barth. **Deep reinforcement learning-based vehicle energy efficiency autonomous learning system**. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1228–1233. IEEE, 2017.

[153] Markus Herzog, Andreas W. Ebentheuer, Michael Winter, Julian Taube, Joachim Froeschl, and Hans-Georg Herzog. **Applications of the Viable System Model in automotive and battery storage systems**. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1747–1752. IEEE, 2016.

[154] Tom Kohler, Joachim Froeschl, Christiane Bertram, Dominik Buecherl, and Hans-Georg Herzog. **Approach of a Predictive, Cybernetic Power Distribution Management**. *World Electric Vehicle Journal*, **4**(1):22–30, 2010.

[155] Christian Lipski, Bjorn Scholz, Kai Berger, Christian Linz, Timo Stich, and Marcus Magnor. **A Fast and Robust Approach to Lane Marking Detection and Lane Tracking**. In *2008 IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 57–60. IEEE, 2008.

[156] Wei Liu, Hongliang Zhang, Bobo Duan, Huai Yuan, and Hong Zhao. **Vision-Based Real-Time Lane Marking Detection and Tracking**. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 49–54. IEEE, 2008.

[157] Tsung-Ying Sun, Shang-Jeng Tsai, and V. Chan. **HSI color model based lane-marking detection**. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 1168–1172. IEEE, 2006.

[158] J. C. McCall and M. M. Trivedi. **An integrated, robust approach to lane marking detection and lane tracking**. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 533–537. IEEE, 2004.

[159] BMW Group. **Owner's Handbook: The BMW 7 Series: Part no. 01402667335 - VI/19**, 2019.

[160] Hermann Winner, Bernd Danner, and Joachim Steinle. **Adaptive Cruise Control**. In Hermann Winner, Stephan Hakuli, and Gabriele Wolf, editors, *Handbuch Fahrerassistenzsysteme*, pages 478–521. Vieweg+Teubner, Wiesbaden, 2009.

[161] ISO 15622. **Intelligent transport systems — Adaptive cruise control systems — Performance requirements and test procedures**, 2018.

[162] ISO 15623. **Intelligent transport systems — Forward vehicle collision warning systems — Performance requirements and test procedures**, 2013.

[163] Charles A. Green and Uzmaa H. Balbale. **Cross traffic alert system for a vehicle, and related alert display method**, Patent, US 2010/0201508 A1, 2010.

[164] Parag H. Batavia. *Driver-adaptive lane departure warning systems*. PhD Thesis, Carnegie Mellon University, 1999.

[165] Sam-Yong Kim and Se-Young Oh. **A driver adaptive lane departure warning system based on image processing and a fuzzy evolutionary technique**. In *2003 IEEE Intelligent Vehicles Symposium*, pages 361–365. IEEE, 2003.

[166] Carsten Schmitz. *Adaptiver Spurverlassenswarner mit fahrerabsichts- und fahrerzustandsabhängiger Warnstrategie*. PhD Thesis, Karlsruher Institut für Technologie, Karlsruhe, 2004.

[167] ISO 17387. **Intelligent transport systems — Lane change decision aid systems (LCDAS) — Performance requirements and test procedures**, 2008.

[168] Frank Beruscha, Klaus Augsburg, and Dietrich Manstetten. **Haptic warning signals at the steering wheel: A literature survey regarding lane departure warning systems (short paper)**. *Haptics-e, The electronic journal of haptics research*, 2011.

[169] Corinna Cortes and Vladimir Vapnik. **Support-vector networks**. *Machine Learning*, **20**(3):273–297, 1995.

[170] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. **The curse of dimensionality for local kernel machines**. *Techn. Rep*, **1258**, 2005.

[171] Tin Kam Ho. **Random decision forests**. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, pages 278–282. IEEE Comput. Soc. Press, 1995.

[172] B. W. Matthews. **Comparison of the predicted and observed secondary structure of T4 phage lysozyme**. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, **405**(2):442–451, 1975.

[173] Harald Cramér. *Mathematical methods of statistics*. Princeton paperbacks. Princeton Univ. Press, Princeton, 19. printing edition, 1999.

[174] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. **Scikit-learn documentation - v0.20: Module sklearn.metrics.matthews_corrcoef** [Accessed 2020-03-02].

[175] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. **Scikit-learn: Machine Learning in Python**. *Journal of Machine Learning Research*, **12**:2825–2830, 2011.

[176] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. **Scikit-learn documentation - v0.20: Module sklearn.svm.SVC** [Accessed 2020-03-02].

[177] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. **Scikit-learn documentation - v0.20: Module sklearn.ensemble.RandomForestClassifier** [Accessed 2020-03-02].

[178] BMW Group. **The new BMW 7 Series**, 2015-06-10 [Accessed 2020-08-15].

[179] Nvidia. **DIGITS™ DevBox**, 2020-02-03 [Accessed 2020-02-03].

[180] Paul Jaccard. **Étude comparative de la distribution florale dans une portion des Alpes et des Jura**. *Bulletin de la Société Vaudoise des Sciences Naturelles*, **37**:547–579, 1901.

[181] Mario Gleirscher and Stefan Kugele. **From Hazard Analysis to Hazard Mitigation Planning: The Automated Driving Case**. In Clark Barrett, Misty Davies, and Temesghen Kahsai, editors, *NASA Formal Methods*, **10227** of *Lecture Notes in Computer Science*, pages 310–326. Springer International Publishing, Cham, 2017.

[182] ISO 26262. **Road vehicles - Functional safety**, 2011.

[183] Donald Ervin Knuth. *Seminumerical algorithms*, **Donald E. Knuth ; vol. 2** of *The art of computer programming*. Addison-Wesley, Boston, 3rd edition, 33rd printing edition, 2016.

[184] B. P. Welford. **Note on a Method for Calculating Corrected Sums of Squares and Products**. *Technometrics*, **4**(3):419–420, 1962.

[185] Frank E. Grubbs. **Procedures for Detecting Outlying Observations in Samples**. *Technometrics*, **11**(1):1–21, 1969.

[186] Wilhelmine Stefansky. **Rejecting Outliers in Factorial Designs**. *Technometrics*, **14**(2):469–479, 1972.

[187] F. J. Anscombe. **Rejection of Outliers**. *Technometrics*, **2**(2):123–146, 1960.

[188] BMW Group. **The BMW X5 xDrive40e**, 2015-03-15 [Accessed 2020-08-15].

[189] BMW Group. **Owner's Handbook: The BMW X5 with eDrive: Part no. 01402915553 - X/16**, 2016.

[190] BMW Group. **The new BMW X5: Photo Attachments**, 2013-05-30 [Accessed 2020-08-15].

[191] Vector Informatik GmbH. **Breakout Box D62Y9 and VNcable D62Y9**, 2020 [Accessed 2020-02-03].

[192] Vector Informatik GmbH. **VN7572 FlexRay/CAN/LIN/IO Interface**, 2020 [Accessed 2020-02-03].

[193] Delta Components GmbH. **Fanless Box-PC 790x Series**, 2018-05-18 [Accessed 2020-03-10].

[194] Mohsen Jafari Asbagh and Hassan Abolhassani. **Feature-Based Data Stream Clustering**. In *2009 Eighth IEEE/ACIS International Conference on Computer and Information Science*, pages 363–368, 2009.

[195] Romaric Duvignau, Bastian Havers, Vincenzo Gulisano, and Marina Papatriantafilou. **Querying Large Vehicular Networks: How to Balance On-Board Workload and Queries Response Time?** In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2604–2611. IEEE, 2019.

[196] Daniel Alvarez-Coello, Benjamin Klotz, Daniel Wilms, Sofien Fejji, Jorge Marx Gomez, and Raphael Troncy. **Modeling dangerous driving events based on in-vehicle data using Random Forest and Recurrent Neural Network**. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 165–170. IEEE, 2019.