Wissenschaftliche Reihe
Fahrzeugtechnik Universität Stuttgart

RESEARCH

Philipp Bergmeir

Enhanced Machine Learning and Data Mining Methods for Analysing Large Hybrid Electric Vehicle Fleets based on Load Spectrum Data

IVK

Springer Vieweg

Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart

Reihe herausgegeben von
M. Bargende, Stuttgart, Deutschland
H.-C. Reuss, Stuttgart, Deutschland
J. Wiedemann, Stuttgart, Deutschland

Das Institut für Verbrennungsmotoren und Kraftfahrwesen (IVK) an der Universität Stuttgart erforscht, entwickelt, appliziert und erprobt, in enger Zusammenarbeit mit der Industrie, Elemente bzw. Technologien aus dem Bereich moderner Fahrzeugkonzepte. Das Institut gliedert sich in die drei Bereiche Kraftfahrwesen, Fahrzeugantriebe und Kraftfahrzeug-Mechatronik. Aufgabe dieser Bereiche ist die Ausarbeitung des Themengebietes im Prüfstandsbetrieb, in Theorie und Simulation.

Schwerpunkte des Kraftfahrwesens sind hierbei die Aerodynamik, Akustik (NVH), Fahrdynamik und Fahrermodellierung, Leichtbau, Sicherheit, Kraftübertragung sowie Energie und Thermomanagement – auch in Verbindung mit hybriden und batterieelektrischen Fahrzeugkonzepten.

Der Bereich Fahrzeugantriebe widmet sich den Themen Brennverfahrensentwicklung einschließlich Regelungs- und Steuerungskonzeptionen bei zugleich minimierten Emissionen, komplexe Abgasnachbehandlung, Aufladesysteme und -strategien, Hybridsysteme und Betriebsstrategien sowie mechanisch-akustischen Fragestellungen.

Themen der Kraftfahrzeug-Mechatronik sind die Antriebsstrangregelung/Hybride, Elektromobilität, Bordnetz und Energiemanagement, Funktions- und Softwareentwicklung sowie Test und Diagnose.

Die Erfüllung dieser Aufgaben wird prüfstandsseitig neben vielem anderen unterstützt durch 19 Motorenprüfstände, zwei Rollenprüfstände, einen 1:1-Fahrsimulator, einen Antriebsstrangprüfstand, einen Thermowindkanal sowie einen 1:1-Aeroakustikwindkanal.

Die wissenschaftliche Reihe „Fahrzeugtechnik Universität Stuttgart“ präsentiert über die am Institut entstandenen Promotionen die hervorragenden Arbeitsergebnisse der Forschungstätigkeiten am IVK.

Reihe herausgegeben von

Prof. Dr.-Ing. Michael Bargende
Lehrstuhl Fahrzeugantriebe,
Institut für Verbrennungsmotoren und
Kraftfahrwesen, Universität Stuttgart
Stuttgart, Deutschland

Prof. Dr.-Ing. Hans-Christian Reuss
Lehrstuhl Kraftfahrzeugmechatronik,
Institut für Verbrennungsmotoren und
Kraftfahrwesen, Universität Stuttgart
Stuttgart, Deutschland

Prof. Dr.-Ing. Jochen Wiedemann
Lehrstuhl Kraftfahrwesen,
Institut für Verbrennungsmotoren und
Kraftfahrwesen, Universität Stuttgart
Stuttgart, Deutschland

Weitere Bände in der Reihe http://www.springer.com/series/13535

Philipp Bergmeir

Enhanced Machine Learning and Data Mining Methods for Analysing Large Hybrid Electric Vehicle Fleets based on Load Spectrum Data

Springer Vieweg

Philipp Bergmeir
Stuttgart, Germany

Zugl.: Dissertation Universität Stuttgart, 2017

D93

Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart
ISBN 978-3-658-20366-5 ISBN 978-3-658-20367-2 (eBook)
https://doi.org/10.1007/978-3-658-20367-2

Library of Congress Control Number: 2017961103

Springer Vieweg

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer Vieweg imprint is published by Springer Nature
The registered company is Springer Fachmedien Wiesbaden GmbH
The registered company address is: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

„For Tina.“

Acknowledgements

I would like to thank all those people who made this thesis possible and an unforgettable experience for me.

First, I would like to acknowledge my gratitude to Prof. Dr.-Ing. Michael Bargende, my advisor at the University of Stuttgart, for his great support and his permanent commitment in the doctoral program “Promotionskolleg HYBRID” in which I was participating. Without him, I probably would not have been given the chance to write this thesis as part of a collaboration between the University of Stuttgart, the Esslingen University of Applied Sciences, and the Daimler AG.

I also owe Prof. Dipl.-Ing. Jürgen Nonnast, my advisor at the Esslingen University of Applied Sciences, a great debt of gratitude: He was always willing to listen to me and helped me with technical as well as organizational questions. He also gave me the opportunity to design and to give my own lectures about Data Mining at the Department of Information Technology.

Next, I would like to thank Prof. Dr.-Ing. Oliver Sawodny for his interest in this work and for joining the doctoral commitee.

I would like to express my deep sense of gratitude to Dr. Christof Nitsche, my advisor at Daimler AG, for his continuous advice and encouragement he gave to me throughout the four years I spent on this thesis. I thank him for his patience, motivation, and enthusiasm. His systematic guidance and experience helped me in all the time of research and writing of this thesis.

My very sincere thanks also goes to the whole department RD/PGH at Daimler AG for offering me the opportunity to work on a thrilling industrial research project regarding hybrid electric vehicles. Namely, I want to thank Peter Antony, my team leader at Daimler AG, and the heads of the department RD/PGH during my research time, Dr. Uwe Keller and Jochen Strenkert, for their support, especially in organizational matters.

For many fruitful discussions and for supporting me in programming tasks, my special thanks goes to my colleagues Gautham Raju, Vishal Ratra, and Kirankumar Reddy from Mercedes-Benz Research and Development India.

Furthermore, I also want to express my gratitude to the Ministry for Science, Research and Arts Baden-Württemberg for funding the doctoral program “Promotionskolleg HYBRID”.

I am thankful to Dr. Andreas Theissler for being my co-lecturer in the two courses “Introduction to Data Mining” and “Intelligent Data Analytics”. For sure, I am going to miss our excellent collaboration.

For their great technical support regarding the usage of the computer cluster of the Esslingen University of Applied Sciences, I want to thank Dr. Adrian Reber and Alexandru Saramet.

Moreover, I thank my friends and former, fellow Ph.D. students Dr. Daniel Görke and Dr. Andreas Haag for all the stimulating technical and non-technical discussions we had as well as for the athleticism we performed in our leisure times.

I am indebted to my cousin, Dr. Christoph Bergmeir, for his helpful advices, in particular regarding the process of publishing scientific papers.

None of this would have been possible without the love and patience of my family. That is why I take this opportunity to express the profound gratitude from my deep heart to my parents, my grandparents, my sister, my brother-in-law, my niece Luisa, my nephew Lukas, my sister-in-law, and my parents-in-law for their unconditional support throughout my whole life. They never doubted any of my decisions, they helped me to clear my mind in stressful times, and I am very glad to know that they will always stand by me, in good as well as in bad times.

Finally, but most importantly, I would like to thank my beloved wife Tina. Without her endless support, it would have been impossible for me to master the challenge of writing this thesis. Dear Tina, thank you for understanding all of my decisions and for sticking by me, in spite of the geographical distance between us during the last few years. Thank you for loving me just the way I am.

Augsburg Philipp Bergmeir

Contents

List of Figures

List of Tables

Acronyms

ACC	Accuracy
BAC	Balanced Accuracy
BER	Balanced Error Rate
BMS	Battery Management System
CAN	Controller Area Network
CART	Classification and Regression Trees
cf.	compare
CFS	Correlation-based Feature Selection
CRF	Combined Rule Extraction and Feature Elimination
CV	Cross-validation
DL	Description Length
ECU	Electronic Control Unit
GPS	Global Positioning System
HEV	Hybrid Electric Vehicle
ICE	Internal Combustion Engine
IREP	Incremental Reduced Error Pruning
Isomap	Isometric feature mapping
k-NN	k-Nearest Neighbours
KKT	Karush-Kuhn-Tucker
KL divergence	Kullback-Leibler divergence
Li-ion	Lithium-ion
linearEnet	Linear Regression with an Elastic Net Penalty
linearLasso	Linear Regression with a Lasso Penalty
linearRidge	Linear Regression with a Ridge Penalty

LLE	Locally Linear Embedding
log	(Unregularized) Logistic Regression
logEnet	Logistic Regression with an Elastic Net Penalty
logLasso	Logistic Regression with a Lasso Penalty
logRidge	Logistic Regression with a Ridge Penalty
MDL	Minimum Description Length
MDS	Multidimensional Scaling
OEM	Original Equipment Manufacturer
OOB	Out-of-bag
ORF	Oblique Random Forest
PCA	Principal Components Analysis
pls	Partial Least Squares Regression
PPV	Positive Predictive Value
rbf	Radial Basis Function
RF	Random Forest
RF-t-SNE	t-Distributed Stochastic Neighbour Embedding employing a Random Forest based dissimilarity
RFE	Recursive Feature Elimination
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
SoC	State of Charge
SVDD	Support Vector Data Description
SVM	Support Vector Machine
SVM-RFE	Recursive Feature Elimination for Support Vector Machines
t-SNE	t-Distributed Stochastic Neighbour Embedding
TNR	True Negative Rate
TPR	True Positive Rate

Symbols

	Calligraphic letters
$\mathscr{D}_B$	binary encoded and labelled dataset
$\mathscr{D}$	labelled dataset
$\mathscr{G}$	set of possible class labels
$\mathscr{H}$	high-dimensional space with an inner product
$\mathscr{I}_\tau$	index set of randomly chosen variables at node τ in an (O)RF model
$\mathscr{M}$	low-dimensional data representation, a map
$\mathscr{N}_k\left(x_i\right)$	set of k nearest neighbours of i^{th} instance
$\mathscr{O}\left(N\right)$	Big O notation for efficiency of algorithms
$\mathscr{P}$	probability distribution that is defined over pairs of points in t-SNE and RF-t-SNE, respectively
$\mathscr{Q}_k\left(\mathbf{x}_j\right)$	$\left(100 \cdot k\right)$%-quantile value of the empirical distribution of the j^{th} variable
$\mathscr{R}_g$	rule set for predicting class g
$\mathscr{S}_t\left(\tau\right)$	Shannon entropy at node τ of tree t
$\mathscr{T}$	training data

	Greek letters
α	parameter that regulates the mixture between a lasso and a ridge penalty in an elastic net regularized regression
α_i	Lagrange-multiplier of i^{th} constraint
β_0	bias term, offset
β_j	weight of j^{th} variable in CRF; j^{th} split direction in multivariate classification trees
$\Delta G_t\left(\tau, q_j\right)$	decrease of Gini Index at node τ of tree t resulting from split point q_j
$\Lambda_{\beta_\tau, \mathscr{I}_\tau, \theta_\tau}\left(x_i\right)$	node split function of multivariate trees
Φ	non-linear mapping in kernels, stress function of several dimensionality reduction techniques
ρ	parameter that regulates the influence of precision and recall in the F-measure

$\Delta\mathscr{S}_t\,(\tau, q_j)$	decrease of Shannon entropy at node τ of tree t
σ_i	bandwidth of Gaussian kernel
σ_{rbf}	kernel parameter of SVM with a radial basis function kernel
τ	node in a decision or classification tree
$\tau_{q_j,L}$	left child node of node τ resulting from split point q_j
$\tau_{q_j,R}$	right child node of node τ resulting from split point q_j
θ_τ	split point at node τ
ξ	slack variable
λ	complexity parameter of elastic net penalty, eigenvalues in PCA

Indices

batt	hybrid car battery
f	index of a random forest model in a list of such models
g	class label
i	index of an instance, object, or sample in a dataset; index of a constraint in an optimization problem
j	index of a variable, attribute, or feature in a dataset
t	index of a tree in a random forest model

Numbers

$\mathbf{1}_N$	N-dimensional vector with all components being one

Roman letters

A	set of instances that are covered by a rule after adding a new condition in RIPPER
a	set of instances of class g that are covered by a rule after adding a new condition in RIPPER
$\arg\min$	arguments of the minimum function
r	number of top-ranked features

B	set of instances that fulfil all rule conditions; set of instances that are covered by a rule before adding a new condition in RIPPER
b	set of instances of class g that are covered by a rule before adding a new condition in RIPPER
$BPI\left(X_j\right)$	Balanced Permutation Importance Index of variable X_j
$\mathbf{B}$	binary encoded input matrix
$\overline{B}$	complement set of B
b_i	binary encoding of i^{th} instance
$b_{i,j}$	j^{th} element of binary encoding of i^{th} instance
B_j	j^{th} variable of binary encoded input matrix
C	cost, regularization parameter of SVM
CF	confidence level of pessimistic pruning employed by C5.0 and C5.0rules, respectively
C_g	penalty for misclassification of a class g sample
C_{linear}	cost, regularization parameter of SVM with a linear kernel
$conf(R)$	confidence of a rule R
cov	covariance
$cov(R)$	coverage of a rule
c_p	cost-complexity pruning parameter of CART
C_{rbf}	cost, regularization parameter of SVM with a radial basis function kernel
$cutoff_g$	parameter that adjusts the final voting scheme of a random forest model for class g predictions
$\mathbf{D}$	dissimilarity or distance matrix
d_{ih}	dissimilarity between i^{th} and j^{th} instance
d_E	Euclidean distance
$dist_{RF}\left(x_i, x_h\right)$	random forest dissimilarity between i^{th} and h^{th} instance
$d^{(W)}$	distance measure in space W
err	error function
$err\left(OOB_t\right)$	misclassification error on OOB_t samples
$err\left(\overline{OOB}_t^j\right)$	misclassification error on $\overline{OOB}_t^j$ samples
$errorCheck$	boolean parameter specifying if RIPPER stopping criterion is checked
$\exp$	exponential function

F	set of available, survived features	
f^*	least important feature in RFE	
$f(x)$	decision function of a classification model	
FN	number of false negatives	
FP	number of false positives	
FT_1	synthetic failure type 1	
FT_2	synthetic failure type 2	
$GI(X_j)$	Gini Importance of variable X_j	
$G_t(\tau)$	Gini Index at node τ of tree t	
H	hyperplane; set of instances that fulfil the rule consequent	
$\overline{H}$	complement set of H	
I_{batt}	current of a hybrid car battery	[A]
K	kernel function	
k	number of cross-validation folds; number of neighbours	
$KL(\mathscr{P} \parallel \mathscr{Q})$	Kullback-Leibler divergence between the distributions $\mathscr{P}$ and $\mathscr{Q}$	
L	Lagrange function	
log	logarithm function	
$\ell(\beta_0, \beta)$	log-likelihood function for a node split model	
max	maximum function	
m_i	map point, low-dimensional data representation of i^{th} instance	
min	minimum function	
minCases	minimum number of training samples that must be put in at least two of the splits in C5.0 and C5.0rules, respectively	
trials	number of boosting iterations performed by C5.0 and C5.0rules, respectively	
minSplit	minimum number of training samples that must reach a node in CART in order for a split to be attempted	
minWeights	minimum number of instances that have to be covered by a rule in RIPPER	
m_{try}	number of variables that are randomly selected as candidates for a node split in a random forest model	

N	number of instances in a dataset
n_{comp}	number of components or latent variables in partial least squares regression
n_{iter}	number of iterations
n_{forest}	number of random forest models
$n(S)$	cardinality of a set S
N_τ	number of instances at node τ
$N_{\tau,g}$	number of class g instances at node τ
n_{tree}	number of trees in a random forest model
numFolds	number of folds used in RIPPER
numOpt	number of optimization runs in RIPPER
OOB_t	out-of-bag samples of tree t
$\overline{OOB}_t^j$	out-of-bag samples of tree t with randomly permuted values for variables X_j
P	partition (of a space)
p	number of variables
$P_\alpha(\beta)$	elastic net penalty
$PI(X_j)$	Permutation Importance Index of variable X_j
$PI_g(X_j)$	Permutation Importance Index of variable X_j for class g instances
$Perp$	perplexity
perplexity	perplexity parameter of t-SNE and RF-t-SNE, respectively
$\hat{p}_{\tau,g}$	proportion of class g instances at node τ
$\hat{p}_{i,g}$	fraction of trees in a random forest that predict class g for the i^{th} instance
$\hat{p}(S)$	proportion of instances in a set S
$p_{i\|h}$	conditional probability
$prox_f(x_i, x_h)$	proximity between i^{th} and j^{th} instance, determined with f^{th} RF model
prune	boolean parameter specifying if RIPPER pruning strategy is performed
q_j	split point for j^{th} variable in a univariate classification tree
R	list of eliminated features in RFE; classification rule
Body	antecedent of a rule
Head	consequent of a rule

S	set of support vectors	
s	support vector	
$sampsize_g$	random forest parameter that specifies the bootstrap size that is drawn from class g	
$sign$	sign or signum function	
SoC_{batt}	state-of-charge of a hybrid car battery	$[\%]$
$supp(R)$	support of a rule R	
T_{batt}	temperature of a hybrid car battery	$[°C]$
TN	number of true negatives	
TP	number of true positives	
u	number of dimensions of map point m_i	
w_g	weight of class g	
$\mathbf{X}$	input data matrix	
x_i	vector of observed values of i^{th} instance, object, or sample	
x_{ij}	observed value for j^{th} variable of i^{th} instance	
X_j	j^{th} variable, attribute, or feature	
$\mathbf{x_j}$	vector of observed values of j^{th} variable, attribute, or feature	
$\bar{x}$	mean value of vector x	
Y	response variable	
$\mathbf{y}$	vector of true class labels	
y_i	true class label of i^{th} instance	

Abstract

In order to optimize the dimensioning of current and future hybrid power-trains and their components with respect to their reliability as well as to economic reasons, it is obligatory to identify relationships between the usage and stress of a vehicle and failures of these elements. Hence, modern control units are installed in vehicles, nowadays, to record many diverse load spectra of different dimensions, which have to be exploited for this purpose. Since it has become almost impossible to analyse manually and simultaneously this huge amount of data that is transferred from each vehicle of a large fleet to databases, which are owned by the car manufacturers, there is an urgent need for methods that allow for an automated analysis of this kind of data.

Therefore, the first part of this work assesses the usability of modern classification algorithms, from the field of Machine Learning, to distinguish between vehicles with and those without a particular component failure, if these methods are only applied to load spectrum data. Additionally, a new approach is proposed that does not only achieve a higher classification accuracy than the other studied methods on two real-world datasets that have been recorded for large hybrid electric vehicle fleets, but also requires for this purpose only a small subset of load spectrum classes, which are selected autonomously. Furthermore, it is demonstrated that the extracted load spectrum classes represent stress factors that are known to be related to the considered component failure.

In practise, there is usually no information available whether a particular component failed in all affected vehicles due to the same stress or not. Thus, in the second part of this thesis a dimensionality reduction technique, from the field of Data Mining, is developed that makes it possible to visualize all load spectra, which have been recorded for a large vehicle fleet, simultaneously. Two case studies are performed to compare this new approach with common competitors and to demonstrate its potential to detect distinct stress and usage patterns.

In the third and last part of this work it is investigated if rule learning methods are applicable to load spectrum data to identify and to describe patterns in the data that are symptomatic for vehicles suffering from a particular component failure. For this purpose, new approaches are proposed and it is shown by means of a synthetic as well as a real-world dataset that these methods are able to extract failure-related patterns from huge load spectrum datasets. Moreover, they significantly outperform the other studied rule learning techniques with respect to the confidence value that is achieved by the best created rule. Hence, these algorithms may support engineers to find out which stress patterns are harmful for particular components.

Kurzfassung

Um die Auslegung aktueller und zukünftiger Hybridantriebsstränge und deren Komponenten hinsichtlich Zuverlässigkeit sowie wirtschaftlicher Aspekte optimieren zu können, müssen Zusammenhänge zwischen Fahrzeugnutzung beziehungsweise -belastung und Komponentenausfällen identifiziert werden. Deshalb werden heutzutage moderne Steuergeräte in den Fahrzeugen verbaut, um eine Vielzahl an unterschiedlich dimensionalen und gearteten Belastungskollektiven aufzuzeichnen, die es zu diesem Zwecke auszuwerten gilt. Da eine manuelle und simultane Analyse dieser Daten, die für jedes einzelne Fahrzeug einer großen Fahrzeugflotte in die Datenbanken der Automobilhersteller übertragen werden, nahezu unmöglich geworden ist, müssen zwingend Methoden entwickelt werden, die eine automatisierte Auswertung dieser Daten ermöglichen.

Im ersten Teil dieser Arbeit erfolgt deshalb eine Bewertung moderner Klassifikationsverfahren aus dem Bereich des Maschinellen Lernens hinsichtlich ihres jeweiligen Potentials, Fahrzeuge mit spezifischem Komponentenausfall von solchen ohne zu unterscheiden und zwar rein anhand ihrer Belastungskollektive. Zusätzlich wird ein neues Verfahren vorgestellt, das auf den Realdatensätzen zweier großer Hybridfahrzeugflotten die jeweils höchste Klassifikationsgüte erzielt und hierfür zugleich nur eine geringe Anzahl an Belastungskollektivklassen benötigt, die automatisiert ausgewählt werden. Außerdem wird gezeigt, dass diese selektierten Belastungskollektivklassen Stressfaktoren repräsentieren, die bekanntermaßen mit dem untersuchten Komponentenausfall zusammenhängen.

In der Praxis ist die Information, ob der Ausfall einer bestimmten Komponente bei allen betroffenen Fahrzeugen durch die gleiche Belastung zustande kam, in der Regel nicht verfügbar. Deshalb wird im zweiten Teil dieser Arbeit ein Verfahren zur Dimensionsreduktion aus dem Bereich des Data Minings entwickelt, das eine simultane Visualisierung aller Belastungskollektive erlaubt, die für zwei große Hybridfahrzeugflotten aufgezeichnet wurden. Anhand zweier Fallstudien, in denen je ein Vergleich mit gängigen Konkurrenz-Algorithmen erfolgt, wird der große Nutzen dieser Methode zur Detektion unterschiedlicher Fahrzeugbelastungs- und -nutzungsmuster demonstriert.

Der dritte und letzte Teil dieser Arbeit untersucht die Anwendbarkeit von Regellernverfahren zur Identifikation und Beschreibung der oben genannten Muster in den Belastungskollektivdaten, die charakteristisch für Fahrzeuge mit Komponentenausfall sind. Hierfür werden neue Methoden entwickelt und anhand eines synthetisch generierten sowie eines Realdatensatzes gezeigt, dass diese in der Lage sind, Ausfall relevante Belastungsmuster aus großen Belastungskollektivdatensätzen zu extrahieren. Des Weiteren übertreffen diese die untersuchten, bestehenden Verfahren hinsichtlich der erzielten Konfidenz der Muster beschreibenden Regeln deutlich. Daher können diese Algorithmen Ingenieure dabei unterstützen zu identifizieren, welche Belastungsmuster schädlich für bestimmte Fahrzeugkomponenten sind.

1 Introduction

Nowadays, modern vehicles, like hybrid electric vehicles (HEV), are equipped with many electronic control units (ECUs) such as an engine control unit or a battery management system (BMS). In general, among the main tasks of such a unit are monitoring the state of a component, controlling various of its functions, and protecting it from abnormal use. For this purpose, an ECU reads data that are measured by sensors and calculates control values and statistics that describe the component's load, e.g., *load spectrum data* [61], based on the sensor input. Afterwards, it communicates these values over several bus systems to actuators or further ECUs [79, 113].

In order to optimize the dimensioning of a vehicle's component or the entire power-train, engineers have to assess the vehicle's stress, i.e., amongst others, they have to analyse the on-board data that are computed and recorded on ECUs. On the one hand, optimizing the design of the components of a vehicle, e.g., determining the adequate engine size, is a crucial step in a vehicle's development process. Due to challenging emission requirements given by governments, engineers have to reduce the weight and to improve the efficiency of a vehicle and its components as much as possible, nowadays [70]. On the other hand, a vehicle's performance does not have to suffer from these modifications to still achieve a high customer satisfaction. For the same reason and in order to avoid high warranty costs, also the reliability of cars and their elements should not be affected by the optimized dimensioning.

However, the analysis of the above mentioned on-board data becomes more and more complex, because the increasing number of ECUs in a vehicle as well as the fast development of cheap but powerful data collection and storage tools leads to a huge amount of data that is recorded on-board of a modern car. Hence, extracting valuable information from these data, e.g., load patterns that are related to a failure of a particular component, and converting it to structured knowledge is getting more and more complicated [49]. In particular, analysing these data manually by a human expert, which has been a common way of exploiting the data in the past, becomes very time consuming and exhaustive.

Moreover, it is very difficult to explore and understand all the interdependencies that may occur in such a complex mechanical and electrical system,

P. Bergmeir, *Enhanced Machine Learning and Data Mining Methods for Analysing Large Hybrid Electric Vehicle Fleets based on Load Spectrum Data*, Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart, https://doi.org/10.1007/978-3-658-20367-2_1

like a hybrid power-train, and that may provoke component failures. As a consequence, automatic, fast, and less human-resource demanding methods have to be developed that allow to uncover these relationships between loads and failures of components as well as hidden structure, like different types of vehicle usage, in the mass of data.

1.1 Aims

The main goal of this thesis is to research ways to autonomously analyse a huge amount of *load spectrum data*, which is recorded for large vehicle fleets and that will be described in Chapter 2 in detail. For this purpose, this thesis aims at developing Data Mining and Machine Learning approaches that allow engineers to tackle the following issues by only using the mentioned type of automotive data:

- Building models that are able to distinguish between vehicles that suffer from a particular component failure and those that do not by exploiting exclusively their load spectrum data, which describe the stress and usage of these cars;
- Assessing if there are different patterns of stress or usage, like distinct usage types, within a large vehicle fleet;
- Uncovering and describing stress patterns that are related to particular component failures.

In summary, it is investigated how a plenty of load spectrum data can be analysed concurrently to get a better understanding of to what stress patterns modern vehicles, like HEVs, are exposed to under real-world operating conditions.

1.2 Related work

Data Mining, Machine Learning, and Artificial Intelligence techniques have been successfully applied to several kinds of automotive data. However, there are only few publications about using these methods to extract knowledge from aggregated logged on-board data such as *load spectrum data*.

Furthermore, most of these few publications propose approaches for building Machine Learning methods based on load spectrum data in combination with workshop data in order to predict maintenance needs of vehicles. Frisk et al. [40], for example, use this kind of data to learn *random survival forest* [56] models to compute prognostic information on the remaining useful life (RUL) of lead-acid batteries in Scania heavy-duty trucks. In [94], the applicability of the Machine Learning algorithms *k-Nearest Neighbours (k-NN)* [37], *C5.0* [96], and *random forest* [16] is studied for predicting compressor faults of Volvo trucks. Although the quality of their results is not great, the authors conclude that using Data Mining techniques is a viable approach for solving this task. The extension of this work is the recent publication [95], where a combination of random forest models and two feature selection methods are applied to logged on-board data for predicting the need for repairs of air compressors in heavy duty trucks.

On contrary, a lot of research has been done on analysing other types of automotive data, e.g., time series or warranty data. Theissler, for example, developed in his PhD thesis [113] a semi-autonomous approach employing an enhanced variant of *support vector data description (SVDD)* [111] and user-driven Visual Data Mining tools that can support engineers to detect anomalies in a large set of measurement data that is recorded during test drives. Thus, stress on the vehicle is given by multivariate time series data, but not in the aggregated form of load spectrum data, as in this work.

In [23], a sequential pattern mining algorithm is presented that extracts patterns and relationships among occurrences of warranty claims over time in a large automotive warranty database. These patterns are depicted as sequential rules, which are represented as simple IF-THEN clauses. Moreover, the proposed algorithm tries to filter out insignificant patterns by using rule strength parameters. The remaining significant rules reveal relations between past and current product failures and future ones. However, in this work there is no information about the usage or loads of the vehicles available.

Bishop [11] provides a comprehensive overview of "intelligent vehicle systems" that are installed in modern vehicles, e.g., vehicle safety systems that help to avoid collisions by monitoring nearby traffic and by warning the driver. In [45], in particular artificial intelligence applications in the automotive industry are surveyed. Therein, various domains, such as manufacturing, onboard fault diagnostics and prognostics, and after-market service are considered.

A substantial review of the literature talking about Data Mining applications in the wide domain of manufacturing can be found in [28]. Among the discussed topics are the use of classification methods for quality control and fault diagnostic as well as for predicting maintenance needs of components.

In [98], an ontology based text mining system is proposed to identify best-practice repairs by analysing millions of unstructured repair verbatim. Thus, the provided decision-support system can help to improve the fault diagnostic and allows to perform root-cause investigations of faults. In order to automatically assign diagnostic categories, like engine or brake, to textual problem descriptions, in [55] a text document classification system is introduced.

In summary, analysing aggregated logged on-board vehicle data, e.g., load spectrum data, by using Data Mining or closely related techniques seems to be a new research field. However, since this kind of data is already recorded in mass by modern ECUs and since it describes the different types of stress on a vehicle and its components, research on this topic becomes more and more important.

1.3 Own publications

Some parts of this thesis have already been published in the following papers, with various researches co-authoring:

- **2014:** In the paper *"Using Balanced Random Forests on Load Spectrum Data for Classifying Component Failures of a Hybrid Electric Vehicle Fleet"* [6], a novel random forest based classification system is proposed to distinguish between HEVs suffering from a failure of a hybrid component and non-faulty ones by only exploiting load spectrum and workshop data. Moreover, the usability of this system to select a small number of failure related load spectrum classes is studied.

- **2014:** The technical paper *"Klassifikationsverfahren zur Identifikation von Korrelationen zwischen Antriebsstrangbelastungen und Hybridkomponentenfehlern einer Hybridfahrzeugflotte"* [5] compares the approach that is introduced in [6] to a popular feature elimination method that employs a support vector machine.

- **2015:** In the technical paper *“Methoden des Data Mining zur Visualisierung unterschiedlicher Belastungsmuster einer Hybridfahrzeugflotte auf Basis von Lastkollektivdaten”*, the subsequently published paper [8] is extended with an additional case study where the data of another HEV fleet is exploited.
- **2016:** In the paper *“Classifying component failures of a hybrid electric vehicle fleet based on load spectrum data – Balanced random forest approaches employing uni- and multivariate decision trees”* [9], oblique random forest models are employed in the framework, which is proposed in [6], and additionally compared to the original models.
- **2016:** The paper *“A Load Spectrum Data based Data Mining System for Identifying Different Types of Vehicle Usage of a Hybrid Electric Vehicle Fleet”* [8] proposes a visualization technique for identifying different patterns of vehicle usage by using all the available load spectrum data that has been recorded for a real-world HEV fleet.

1.4 Outline

The structure of this thesis is organised in seven chapters. Each chapter starts with a brief motivation and overview such that the reader is able to get rapidly an idea of what the chapter is all about. Next, fundamentals are provided, if required for understanding the chapter’s content, before the enhancements and new approaches are explained, in detail. Thereby, references to literature are given continuously in this thesis, whenever this work is based on previous publications.

Besides this introductory part, the thesis includes the following chapters:

2. **Data foundation:** The second chapter explains what kind of data is analysed during this thesis and how the data are generated in modern vehicles such as HEV. Moreover, the data preprocessing steps are described that are required to ascertain that the data quality is sufficiently high enough.
3. **Classifying component failures of a vehicle fleet:** The terms “classification” and “feature selection” as well as corresponding state-of-the-art algorithms are introduced in the third chapter. Furthermore, newly developed classification approaches are proposed with the goal to build models that

are able to distinguish between vehicles with and those without a failure of a component of the hybrid power-train by exploiting only load spectrum data. Moreover, it is investigated whether these methods are able to select load spectrum classes that are related to the studied component failures. Finally, the performance of the existing and new models is assessed in two big case studies that are conducted using the load spectrum data that has been recorded for two large HEV fleets.

4. **Visualizing different kinds of vehicle stress and usage:** This chapter explains several state-of-the-art dimensionality reduction techniques and introduces a new approach called RF-t-SNE. It investigates the applicability of the discussed methods to visualize simultaneously the entire load spectrum dataset of a large HEV fleet. Therefore, a first case study is carried out that shows that RF-t-SNE allows to detect visually if there are different types of usage and stress patterns in the dataset, on the one hand. Afterwards, it is demonstrated in a second case study that this new approach has the potential to identify vehicles that suffer from a component failure visually and hence possibly allows for a visual distinction of different failure types of a particular component.

5. **Identifying usage and stress patterns in a vehicle fleet:** State-of-the-art rule learning methods as well as newly developed RF based rule learners are proposed in this chapter. Their ability to identify and to describe stress patterns in load spectrum data, which are related to component failures of HEV, is evaluated on both an artificially created and a real-world dataset. Thereby, their potential to support engineers in finding out stress patterns that are harmful for particular components is demonstrated empirically.

6. **Conclusion:** A summary of the thesis is given and the main contributions are identified in this chapter. Additionally, limitations and benefits of the proposed approaches are discussed.

7. **Outlook:** The final chapter provides an outlook for possible, future research directions and discusses potential enhancements of the proposed approaches.

2 Data foundation

The success of each Data Mining task strongly depends on the quality of the data that are used. Thus, assuring a high data quality before performing any data analysis is a crucial, maybe the most important step of any data analysis project. This is also pointed out by the popular saying “garbage in, garbage out” [10]. In other words, the results of any data analysis can not be better than the quality of the data.

Therefore, this chapter gives insights into the characteristics of the data that are studied in this work. It starts with the description of the data sources, i.e., it explains what data is analysed in this work and how it is generated. Hence, it introduces a special kind of data that is called *load spectrum data* and illustrates in detail how this data is derived from measurement signals. Then, it discusses how the data are enriched with additional information from the workshops.

Afterwards, it is explicated what preprocessing steps are performed to improve the data quality and to transform the data into an appropriate structure such that the studied Data Mining and Machine Learning algorithms can be applied to it.

At the end of this chapter, the most important properties of the two real-world datasets are provided that are analysed thoroughly in several distinct case studies in this thesis.

2.1 Data sources

During its lifetime, a modern vehicle produces a huge amount of data, such as signals from several sensors and from different ECUs, e.g., the engine control unit, which are communicated within the car through a controller area network (CAN). However, in-vehicle storage of data, i.e., data acquisition that takes place on-board of the vehicles, is still expensive. Thus, the memory capacity of modern vehicles is still limited severely. Moreover, equipping each vehicle with large scale on-board logging solutions would result in high development

P. Bergmeir, *Enhanced Machine Learning and Data Mining Methods for Analysing Large Hybrid Electric Vehicle Fleets based on Load Spectrum Data*, Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart, https://doi.org/10.1007/978-3-658-20367-2_2

costs, since these data loggers have to be provided an intelligent software, they have to be tested thoroughly and they have to be produced in mass, finally [93]. Amongst others, it is still not possible to record the complete data streams in modern customers' cars because of these reasons. Another issue is data confidentiality. It is prohibited by law to continuously log signals which would enable the vehicle manufactures to create individual, customer related driving profiles if the customer does not explicitly permit such a recording. For example, the GPS signal is among these critical signals.

Hence, only development cars and vehicles that are part of a preproduction test fleet are usually equipped with expensive data loggers to perform continuous signal recordings, while this is impossible for mass-production cars. However, the car manufacturers also require the knowledge about how their vehicles are driven and how the components of their cars are stressed under real-world conditions. Otherwise, it would be very difficult for them to improve the vehicles or their components in terms of several aspects, such as emission requirements that are given by the governments. Another example of such an aspect is the reliability of the vehicles, because it affects directly the satisfaction of the customers.

As a consequence, data are nowadays aggregated directly on-board in a customer's vehicle to account for the memory limitations and the restrictions given by data confidentiality reasons, as mentioned before. Then, there are two common ways how the vehicle manufactures extract this kind of aggregated data from each car: Either modern telematics services are used to transmit the data wirelessly to the companies' databases or the data are downloaded during a workshop visit and are sent to these databases, afterwards. This kind of logged and aggregated on-board data is called *load spectrum data* in this work. It is discussed in detail in the following subchapter.

2.1.1 On-board data: load spectrum data

Load spectrum data originate from Fatigue Analysis, where it is still the state-of-the-art data employed for calculating the fatigue life of components. Since the fatigue life of a component depends on the magnitude of the amplitudes of alternating stress and the frequency of occurrences, many counting methods have been developed to transform stress-time functions, such as signals from the ECU, to frequency distributions. However, these data reduction methods

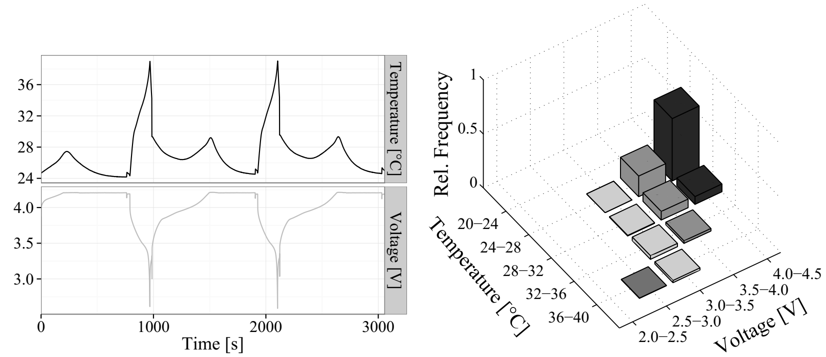


Figure 2.1: Measurements of the Li-ion battery signals *battery temperature* and *terminal voltage* and the corresponding relative load spectrum resulting from a two-parameter level distribution counting (cf. [9])

result in a loss of information, since some intrinsic characteristics of the signal data, like the chronology of certain events, are discarded. Thus, the responsibility lies with the analyst to decide for each use case individually whether such a data reduction is valid or not [61].

In dependence of the number of stress-time functions that are transformed simultaneously, the two main groups, in which these data reduction methods are frequently categorized in, are called the *one-* and *two-parameter counting methods*, respectively [104]. While in practise these are the most common groups, it has to be noted that in theory the majority of these counting methods is not limited by the number of signals they can process at the same time, i.e., they can be applied to even more than two signals. Regardless of the quantity of the processed signals, the outcome of a counting method is called a *load spectrum* in this work. However, it has to be mentioned that sometimes only the result of one-parameter counting is named load spectrum, whereas two-parameter counting leads to a *load matrix*. Since this distinction is irrelevant for the analysis presented in this study, the only term that is used in the following is *load spectrum*.

Since there are a plenty of different counting methods, the focus lies on the explanation of those techniques which are used to create the load spectrum data that are analysed in this work. The majority of them results from the *level distribution counting*, which is illustrated by Figure 2.1. The left chart presents the curves of the two signals *temperature* and *terminal voltage* of a hybrid car battery, while the right one visualizes the corresponding normalized load spectrum that results from a two-parameter level distribution counting. Thereby, the measurements of these two signals are extracted from the publicly available *Battery Data Set* [102] by randomly selecting five runs of the lithium-ion (Li-ion) battery through the alternating operational profiles *charge* and *discharge* at room temperature. Moreover, breaks between these five runs are eliminated and it is additionally assumed that the signal values are measured at every second.

In general, the classes of interest, e.g. intervals, in which each of the two signals shall be grouped in have to be specified first, i.e., before the level distribution counting can be applied to these signals. Thereby, in the case of intervals, the widths do not necessarily have to be of the same size, but overlapping is not allowed. In the example shown in Figure 2.1, the five intervals that are used for partitioning the range of values of the battery's terminal voltage are $2.0 - 2.5V$, $2.5 - 3.0V$, $3.0 - 3.5V$, $3.5 - 4.0V$, and $4.0 - 4.5V$. Furthermore, the battery temperature is divided into the intervals 20 – 24°C, 24 – 28°C, 28 – 32°C, 32 – 36°C, and 36 – 40°C. Afterwards, the total time of measurement is cut into equidistant time intervals. Then, the value of each signal is queried successively at each of the resulting points in time. In the example, the values are queried at every second. At the same time, the counter of the class formed by the corresponding intervals, to which the current values of the two signals belong to, is increased by one. Thus, the frequency counts of each class of interest are also a measure of the signals' operating time within the corresponding classes. Finally, the *normalized* or *relative load spectrum*, which is shown in the right chart of Figure 2.1, is obtained by dividing the frequency counts of each class by the whole measurement time, i.e., by the total sum of counts.

Another important counting method, which is underlying some of the studied load spectra, is the *rainflow-counting* algorithm. This method has been designed to account for cycle fatigue and is proposed in [81] for the first time, whereas [35] is the first publication in English about it. It appeared only a few years later.

The name of this method is based on its analogy with rainwater that is dripping down a pagoda roof. Figure 2.2 illustrates the basic principle of the original variant of this method. First, the (non-linear) time-series of recorded stress

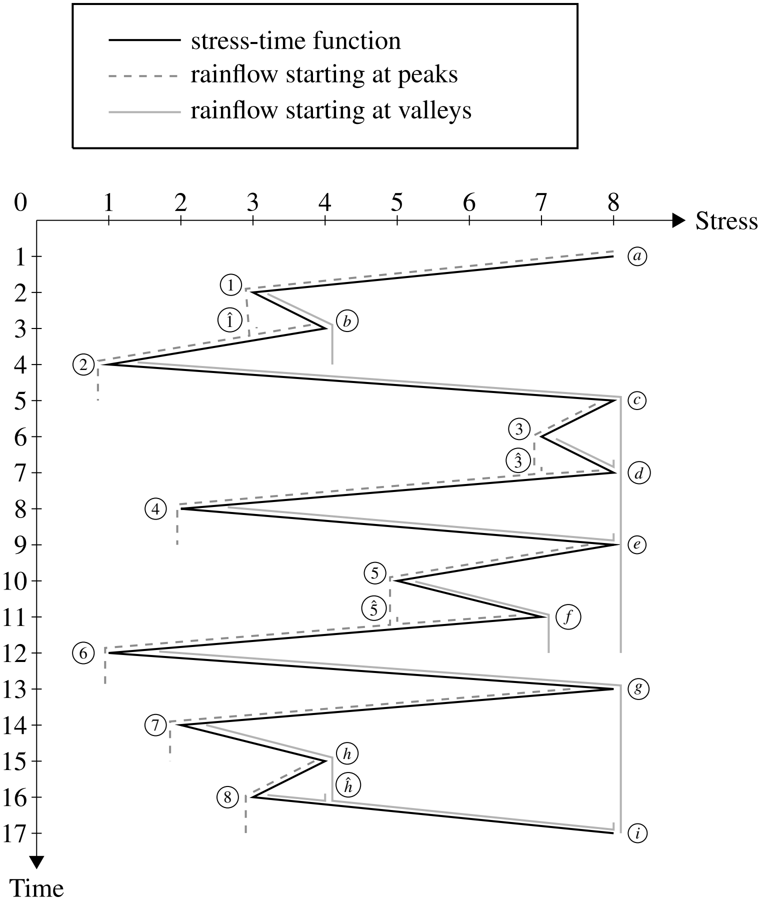


Figure 2.2: Basic principle of the rainflow-counting algorithm with peaks of the stress-time function being tagged with letters, while valleys are labelled with numbers

is reduced to a point process of peaks and valleys. Then, the stress history is rotated clockwise by 90 degrees to symbolize the pagoda roofs. Next, it is assumed that the rainwater flows down the pagoda, where each peak and each valley of the process is imagined as a source of water. In Figure 2.2 peaks are labelled with letters, while valleys are tagged with numbers to be able to distinguish easily between them. Furthermore, each path of rain flow stops if one of the following conditions is satisfied [123]:

- If its source is a valley/peak, if its flow of water reaches a tip of a roof, and if it merges with another flow of rainwater that originates in a valley/peak that has a lower/higher stress value than its source; *or*
- If it merges with another flow that originates in a valley/peak of a higher roof level; *or*
- If it reaches the end of the time history.

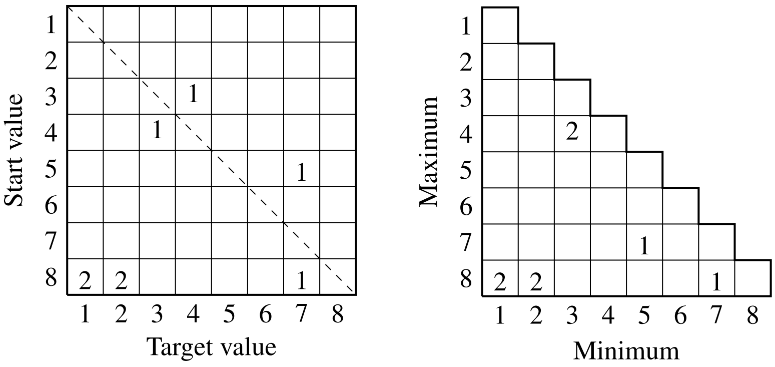
In Figure 2.1, merging points of the water are indicated by letters or figures that are marked with the hat symbol, respectively. Here, the first rule applies, for example, for path $1-b$, whereas the second one holds for $d-\hat{3}$.

Next, each resulting path of rain flow is counted as a half cycle and gets assigned a magnitude that equals the difference in stress between its starting and endpoint. Thus, the path $a-2$ in Figure 2.2 gets assigned a stress value of 7, the path $5-f$ a value of 2, and so forth. Finally, half cycles of the same magnitude and of the same location, i.e., if they have identical maximum and minimum values, are merged to a full cycle if their flow directions oppose each other. In Figure 2.2, the half cycles $a-2$ and $2-c$ build a full cycle, for example.

In order to obtain a load spectrum, the resulting counts are usually presented in form of a matrix or a triangular matrix. A full matrix is used to list the full cycles, their maxima and minima as well as their flow directions. However, if the information about the flow directions is not relevant, the full cycles can be stored in a triangular matrix that accounts for the maxima and minima. Figure 2.3 shows both variants of the load spectrum resulting from applying the rainflow-counting method to the stress history, that is illustrated by Figure 2.2.

Finally, it has to be mentioned that there are more than two possibilities to obtain the final load spectra. In the two representations that are shown in Figure 2.3, only full cycles are counted. However, if the stress-time function leads to half cycles that can not be merged to a full cycle, it would also be possible to count those separately.

In summary, load spectrum data contain aggregated information about the usage and stresses of a vehicle or its components. They are frequently computed



Start value \ Target value	1	2	3	4	5	6	7	8
1								
2								
3				1				
4			1					
5							1	
6								
7								
8	2	2					1	

Maximum \ Minimum	1	2	3	4	5	6	7	8
1								
2								
3								
4			2					
5								
6								
7					1			
8	2	2					1	

Figure 2.3: Two forms of representation of a load spectrum resulting from applying rainflow-counting to the stress-time function shown in Figure 2.2

directly on-board of each car with the help of a control unit. In this work, whenever a vehicle visits a workshop for service and repair that is authorized by an original equipment manufacturer (OEM), the current counts are downloaded and uploaded afterwards to a database that is managed by the OEM. However, the readouts do not provide any information about conducted repairs or replacements of any of the components of the vehicle. This information is collected in a different database which is described in the next subchapter.

2.1.2 Off-board data: workshop data

In this work, there is a database used which stores repair and maintenance information from OEM authorized workshops that facilitates the identification of vehicles suffering from a failure of a hybrid component. Table 2.1 shows a notional excerpt of such a database. It does not only contain information about which vehicle and what element was erroneous, but also provides some interesting facts about the date and type of repairing as well as the fault diagnosis that is made by the mechanist or by a control unit. Additionally, it shows the mileage of the vehicle at the point in time, when the car came into the workshop.

However, it has to be mentioned that the workshop data is mainly used for warranty issues and customer invoicing. Hence, it has unfortunately not been

Table 2.1: Conceptual illustration of the workshop information data

Car ID	**Mileage [km]**	**Date of repair**	**Diagnosis**	**Failure location**	**Repair work**
22	183450	2016-02-17	electric error	BMS	replacement of battery
815	17782	2014-10-03	contact fault	power electronics	cleaning of contacts
1103	64202	2015-08-13	brake wear	brakes	renewal of brake pads
⋮	⋮	⋮	⋮	⋮	⋮

designed for being merged with the database storing the load spectrum records. At least, both databases contain the attributes *CarID*, *Mileage*, and a time-stamp. In the database that stores the load spectrum data, the latter attribute stores the date, when the load spectrum data is read out from the control units of the vehicle. In the workshop database, this time-stamp refers to the manually entered date of repair. Since, the repair work does not necessarily have to take place on the same day as when the error memories of the vehicle are read out, there may be a mismatch of a few days between these to time-stamps of the two discussed databases.

The attribute mileage is also manually inserted by the mechanist in the workshop database, while this attribute stores the value that is revealed by the ECU in the load spectrum database. Therefore, typos may also lead to discrepancies in this attribute between the two databases.

All these issues have to be handled carefully, when the information stored in these two databases have to be merged. Corresponding dates are matched using the attributes *CarID*, *Mileage*, and the mentioned time-stamps in this work. In that way, individual readouts of the load spectrum data of a vehicle can be labelled with the information about the health status of the component of interest.

Table 2.2: Conceptual illustration of the dataset resulting from merging the load spectrum with the workshop information data

Car ID	Battery SoC [%]				Speed [km/h]				# ICE starts	Label
	0–25	25–50	50–75	75–100	0–10	10–20	…	240–250		
1	0.0	0.3	0.4	0.3	0.3	0.1	···	0.0	9832	faulty
2	0.1	0.4	0.4	0.1	0.2	0.2	···	0.1	78920	healthy
3	0.0	0.3	0.5	0.2	0.1	0.1	···	0.0	46010	healthy
⋮	⋮	⋮	⋮	⋮	⋮	⋮	…	⋮	⋮	⋮

2.2 Preprocessing of data

After having merged the load spectrum data and the workshop information, several preprocessing steps are performed to achieve a high quality of the data. First, vehicles with a driven distance less than 1000 km are filtered out, because failures of components of these vehicles belong to the category “early failures” and are usually caused by manufacturing fault and not by loads of the vehicle or its components.

Moreover, sometimes the control units of a vehicle have to be flashed to install a new release of the operating software. As part of this process, the current values of the load spectra are downloaded and then reset to zero. Thus, the data before such a flash of a control unit have to be merged with the load spectrum data that are recorded after this event. However, if the readouts before and after a flash are not properly stored in the database, gaps can occur in the recordings of the load spectrum during the lifetime of a vehicle. Therefore, in order to ensure that the load spectrum data reflect the real usage and load patterns, vehicles are removed from the datasets if the corresponding load spectrum records do not cover at least 75% of their total mileage.

Since all the algorithms, which are studied in this work, require that the input is given in a matrix form, it has to be explained how such a data matrix is obtained. Hence, Table 2.2 presents the arrangement of the observed load spectrum data. All the recorded load spectra of each vehicle are concatenated

Table 2.3: Characteristics of the studied versions a) and b) of two real-world datasets

	Dataset 1		Dataset 2	
Characteristic	a)	b)	a)	b)
Number of vehicles	6848	6670	8131	7576
Number of distinct operating countries	irrelevant	12	irrelevant	11
Minimum number of vehicles per country	irrelevant	25	irrelevant	100
Total number of load spectrum classes	737	737	823	823
Number of vehicles with a failure of the hybrid car battery	195	irrelevant	47	irrelevant

in a such a way that each row stores the observed load spectra values of an individual vehicle, while each column contains the observed values of a particular class of a certain load spectrum. For example, the SoC of the hybrid car battery of vehicle 1 has never been lower than 25%, while it has been in the range from 50% to 75% for 40% of the operating time. Moreover, the ICE of vehicle 1 has been started 9832 times.

2.3 Real-world datasets

Table 2.3 shows the main characteristics of the two datasets that are studied intensively in this work. They result from two large HEV fleets, where the versions a) and b) are created for each of them. Version b) is derived from a) by filtering out vehicles that are driven in countries where the total number of vehicles which are operated in the same country is less than 25 and 100 in dataset 1 and 2, respectively. These reduced datasets are necessary for the visualization purposes of the analyses conducted in Chapter 4.

Each fleet contains only vehicles of the same type, whereas the car type is different in these two main datasets, i.e., the datasets 1 and 2 do not have any vehicle in common. Moreover, individual characteristics which are not used for the studies carried out on the dataset, are flagged as “irrelevant”.

Dataset 1 contains less vehicles and each of them is described by less features compared to those of dataset 2. Furthermore, it is known which vehicles included in the versions a) of both datasets suffer from a failure of the hybrid car battery. However, the types of defect leading to the failure of this component are manifold in dataset 1, whereas there is only a single error type predominant in dataset 2. Nevertheless, it is not differentiated between the distinct kinds of failures in dataset 1, i.e., all faulty vehicles are only labelled as “faulty”. The reason is that there is not enough information available about the different types of failures that are contained in dataset 1.

Finally, there is a severe imbalance between the number of “healthy” and “faulty” vehicles in each of the two datasets. The imbalance ratio between these two classes is approximately 35:1 in dataset 1a), whereas it is 173:1 in dataset 2a). This special property may have a strong influence on the outcome of the algorithms that are studied in this work, as will be explained in Chapter 3.1.2 in detail.

2.4 Conclusion

In this work, the focus lies on performing classification, visualization, and rule learning on load spectrum data that have been enriched by information about repairs of particular components. Since the considered data sources already have been created for other purposes, there is no need for spending any additional money for collecting the data, for buying and setting up the required IT-infrastructure, or for installing new hardware within the vehicles of interest. However, a drawback is that the databases storing the load spectrum data and the workshop information have not been designed for being merged. Hence, merging these two databases is not straightforward.

Like in all data analysis tasks, the quality of the results strongly depends on the data quality. Thus, several pre-processing steps have been performed to create two main datasets where two variants are created from each of them. These two datasets will be studied thoroughly in several distinct case studies in the upcoming chapters.

3 Classifying component failures of a vehicle fleet

In this chapter, the applicability of several state-of-the-art classification algorithms such as random forests and support vector machines are studied for the purpose of distinguishing non-faulty HEV from those suffering from a failure of a particular component of the hybrid power-train, when these algorithms are fed with load spectrum data. Furthermore, it is analysed whether these classifiers can be combined with feature selection approaches to not only improve the classification performance of the models, but also to select a small set of failure related features.

Hence, first the required fundamentals of classification as a functionality of Machine Learning are provided. Moreover, the basics of the studied classification and feature selection methods are surveyed in detail. Afterwards, the practicability of the described techniques is demonstrated by a case study using two real-world datasets. Additionally, a new framework employing random forest models is proposed that is able to autonomously tune the parameters of the incorporated model as well as to filter out irrelevant features.

The presented results have been published partly by the author of this Thesis in [5], [6], and [9].

3.1 Fundamentals of classification

Before the Data Mining functionality *classification* can be explained, some notation has to be provided first. This work mainly follows the notation used in [50]. Mathematically, an $N \times p$ input matrix $\mathbf{X}$ of predictors is given by

$$\mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = (\mathbf{x}_1 \mid \mathbf{x}_2 \mid \ldots \mid \mathbf{x}_p) = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{pmatrix}, \qquad \text{Eq. 3.1}$$

P. Bergmeir, *Enhanced Machine Learning and Data Mining Methods for Analysing Large Hybrid Electric Vehicle Fleets based on Load Spectrum Data*, Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart, https://doi.org/10.1007/978-3-658-20367-2_3

where row $x_i \in \mathbb{R}^p$ denotes the vector containing the observed values of *sample* $i \in \{1, 2, \dots, N\}$, i.e., the load spectra of vehicle i. In addition, column $\mathbf{x}_j \in \mathbb{R}^N$ contains the recorded values of the j^{th} *variable* X_j, $j = 1, 2, \dots, p$, i.e., the values of all the vehicles in a specific load spectrum class. Moreover, the scalar $x_{ij} \in \mathbb{R}$ denominates the entry of the j^{th} variable for the i^{th} sample. Thus, lowercase letters are used for observed values, while uppercase ones, like X_j refer to the generic aspect of a variable. Similarly, the response variable is denoted by $Y \in \mathbb{R}^N$, its observed values by $\mathbf{y}$ and the known class of sample i by y_i. Bold uppercase letters represent matrices such as $\mathbf{X}$, while bold lowercase letters signify vectors with components that are related to several samples, such as $\mathbf{y}$.

Furthermore, the terms *object*, *instance*, *observation*, and *record* are used as synonyms for *sample* in this work. For simplicity, the same terminology will be used for the vector of observed values x_i of sample i if the verbal distinction is not necessary. *Feature* and *attribute*, respectively, are alternatives for the term *variable*. Moreover, the slightly informal notation $x_i \in \mathbf{X}$ is used to indicate that sample x_i belongs to the input matrix $\mathbf{X}$.

Classification is the task of assigning one *class*, *category*, or *group* g to a given object. Thereby, g belongs to a set of possible classes $\mathscr{G}$ that has a finite number of discrete categories, i.e., if $|\mathscr{G}| < \infty$, and that is known beforehand. If $|\mathscr{G}| = 2$, then the problem is called a *binary* or *two-class classification problem* [10, 50]. Without loss of generality, the two possible categories are named the *negative* and *positive* class in this work, respectively, and $\mathscr{G}$ is set to $\{-1, +1\}$.

In the context of Machine Learning, the goal of classification is to learn a model that is able to predict the class label of a given instance on the basis of its observed attribute values. Thereby, the model is learned from a dataset that contains instances for which the classes are already known. However, usually a classification model does not aim to predict the class labels of old data, i.e., of samples for which the corresponding categories are already known. It rather should be capable to correctly predict the classes of new, unseen data, i.e., of instances that have not been used for training the model. Therefore, it does not make sense to assess the performance or quality of a classification model using the training data that has been used to build the model. This would lead to overoptimistic performance estimates. However, in order to be able to decide whether the predictions of a model are correct, the true categories of the instances which serve as its input have to be known [124].

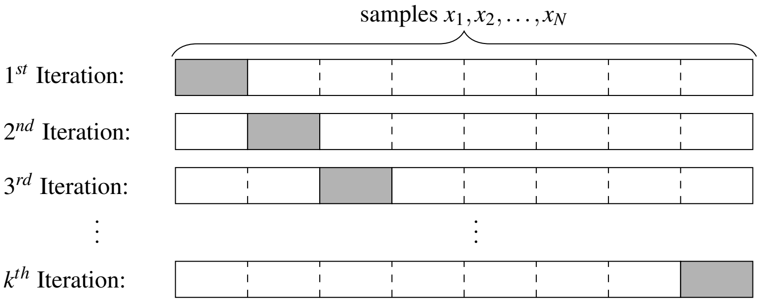


Figure 3.1: The principle of k-fold cross-validation

Hence, a common strategy is to split a given dataset into two disjoint set which are called a *training set* and a *test set*. The first set is used to build the classification model, while the second one is held back and used later to assess the performance of the model. Thereby, the assumption is that both datasets are representative of the whole problem. In a data-rich situation this holdout procedure is fine, since there is enough data to train the model and sufficient data to yield good performance estimates.

If there is not a huge amount of relevant data available, as it is often the case in real world problems, then putting aside a certain fraction of the data for testing leads to the following dilemma: On the one hand, as much of the available data as possible should be used to train the model to get a good classifier; but on the other hand, as much of this data as possible has to be used for testing to also obtain good performance estimates [124].

A common approach which addresses this problem is *k-fold cross-validation (CV)* [50]. Its basic principle is illustrated by Figure 3.1. First, the whole set of observed, labelled data $\mathscr{D} = (\mathbf{X}, \mathbf{y})$ is randomly partitioned into k chunks of (approximately) equal size, where $\mathbf{y}$ contains the known class labels of each instance $x_i \in \mathbf{X}$. Afterwards, a classification model is repeatedly trained k times with each of these k chunks being held out for testing exactly once, while the remaining $k-1$ chunks form the training set. Thus, during the k iterations each chunk serves exactly once as the test set. In Figure 3.1 these test chunks are highlighted in grey, while the remaining training partitions are visually separated by dashed lines. Finally, the performance of the final classification model that is trained on the whole dataset $\mathscr{D}$ is estimated by averaging the results

Table 3.1: Confusion matrix for a binary classification model [124]

		predicted class	
		positive	negative
true class	positive	*true positive (TP)*	*false negative (FN)*
	negative	*false positive (FP)*	*true negative (TN)*

achieved on the k test sets. Common choices for k are 5 and 10 [50]. If the distribution of the class variable Y is respected additionally, e.g., if it is ensured that the distribution of the classes in each fold is approximately the same as in the whole dataset $\mathscr{D}$, the cross-validation is called *stratified*.

3.1.1 Performance Measures

So far, it has been explained that the performance of a classifier has to be evaluated on a test set consisting of instances that have not been used to create the model. Thus, now it has to be clarified, how the performance can be measured.

First, the classification result on the test set is expressed with a *confusion matrix*, sometimes also called *contingency table* [124]. In the particular case of a binary classification problem, Table 3.1 depicts such a result table. Here, the prediction of a single sample has four possible outcomes: A correct prediction is either counted as *true positive (TP)* or *true negative (TN)*, depending on the true class of the considered object, while a wrong class assignment is called a *false negative (FN)* or a *false positive (FP)*.

Amongst others, the following performance measures can be deduced [49]:

- *True positive rate (TPR)*, *sensitivity*, or *recall*: This measure computes the fraction of positive samples that are classified correctly. Thereby, the number of negative instances that are falsely predicted as members of the positive class is not considered. Formally, the TPR is given by

$$TPR = sensitivity = recall = \frac{TP}{TP+FN}. \qquad \text{Eq. 3.2}$$

  A high TPR value signifies that the model is able to classify the vast majority of positive instances correctly. However, it is not quantified whether this outcome is achieved at the expense of misclassifying many objects of the negative class.

- *True negative rate (TNR)* or *specificity*: Similarly, this measure computes the fraction of correctly classified negative objects, while neglecting the correctness of the predictions of the positive class instances:

$$TNR = specificity = \frac{TN}{TN+FP}. \qquad \text{Eq. 3.3}$$

- *Precision* or *positive predictive value (PPV)*: In contrast to the quantities above, this performance measure incorporates the outcome of samples from both classes. It determines the fraction of positive instances in the set of objects which are classified as positive:

$$precision = PPV = \frac{TP}{TP+FP}. \qquad \text{Eq. 3.4}$$

  A high precision value indicates that if a sample is predicted as positive it is very probable that the prediction is correct.

- *Accuracy (ACC)*: The overall success rate is quantified by this performance measure. Therefore, the number of correct is divided by the total number of predictions:

$$ACC = \frac{TP+TN}{TP+FN+FP+TN}. \qquad \text{Eq. 3.5}$$

  Thus, a perfect model achieves an ACC value of 1.

Unfortunately however, all these measures are not suitable for estimating the performance of the models developed in this work, because the studied datasets have a special characteristic that necessarily has to be taken into account. This kind of data property is discussed in the next section.

3.1.2 The “class imbalance problem”

Real-world datasets, like the ones introduced in Section 2.3, suffer often from the fact that one of the two classes is heavily underrepresented in a two-class classification problem. To complicate matters further, the samples of the rare class are frequently the interesting ones. Without loss of generality, the minority class is called the *positive class* and the majority class the *negative class* in the following. The phenomenon that there is a strong imbalance between the number of objects of the two classes which are present in a dataset, is called the *class imbalance problem* [72].

Datasets with this property require a special treatment, when it comes to the task of learning classification models from them, e.g., adequate performance measures. The accuracy measure, e.g., is biased in that case, because a model can achieve a high accuracy value by simply assigning all instances to the majority class. Then, however, the prediction performance of the model is obviously bad for samples of the minority class.

That is why, amongst others, the following performance measures have been developed for this kind of problem:

- *F-measure* [49]: This measure can be interpreted as the weighted average of precision and recall. Thereby, the parameter ρ regulates the influence of the latter two quantities. Formally, this measure is given by

  $$F_\rho = \frac{\left(1+\rho^2\right)(precision \cdot recall)}{\rho^2 \cdot precision + recall}, \qquad \text{Eq. 3.6}$$

  where $\rho > 0$. For $\rho = 1$ it is equal to the harmonic mean of precision and recall. Higher values of F_ρ signify a better classification model.

- *Balanced accuracy (BAC)* [21]: It is another weighted performance measure that is equal to the arithmetic mean of TPR and TNR:

  $$BAC = \frac{TPR + TNR}{2}. \qquad \text{Eq. 3.7}$$

  A BAC value of 1 indicates a perfect classification.

Since the BAC incorporates both the model performance on the positive and on the negative class and because of its straightforward interpretation, it is chosen as the main performance measure for the experiments conducted in this work.

3.2 Classification methods

In this section the studied classification algorithms are introduced. These are *support vector machines*, *classification trees* and *random forests*. Since the later performed case study is basically a binary classification problem, the explanations are restricted to this kind of problem.

3.2.1 Support vector machine (SVM)

In 1995, *support vector machines (SVM)* were introduced by Vapnik [118]. Since then, they have gained a lot of popularity and belong to the state-of-the-art classification and regression algorithms, nowadays. They have been used successfully in many distinct research fields such as pattern recognition [12, 120] and bioinformatics [126].

In a binary classification problem, the main goal of a SVM is to determine a hyperplane that separates the given two classes as good as possible [24]. Since it can not be assumed generally that the two classes can be separated perfectly, both the *separable* and *non-separable (overlap)* case will be discussed in the following. Moreover, some classes may be separable with a *linear* decision boundary, while a *non-linear* one may be necessary for others. Thus, also these two cases are considered in the upcoming sections. Finally, it is explained how a SVM can be modified to be able to handle the class imbalance problem which is inherent to the datasets studied in this work, as mentioned earlier.

Linear hard-margin support vector machine

First, it is assumed that the two classes are linearly separable, i.e., there is a hyperplane that can be placed in between the instances of the two classes in such a way that all samples of the positive class lie on one side, while all the negative class examples can be found on the other side of that hyperplane.

Following the notation in [24], let $\beta \in \mathbb{R}^p$ be the normal vector of such a hyperplane. In addition, let d_+ and d_- be the shortest distance between the hyperplane and the nearest object of the positive and negative class, respectively. Then, the *margin* of this hyperplane is given by $d_+ + d_-$. Now, a SVM aims at finding a hyperplane that maximizes the margin between the two classes. Thus, SVMs belong to the group of *maximum-margin classifiers* [118].

In the linearly separable case, each instance $i \in \{1, \ldots, N\}$ satisfies exactly one of the following two conditions:

$$\text{if } y_i = +1: \qquad x_i^T \beta + \beta_0 \geq +1, \qquad \text{Eq. 3.8}$$

$$\text{if } y_i = -1: \qquad x_i^T \beta + \beta_0 \leq -1, \qquad \text{Eq. 3.9}$$

where β_0 is called the *bias*, i.e., the offset of the hyperplane.

Due to using opposite signs for the two classes, these two inequalities can be combined easily into a single one, as follows:

$$\forall i \in \{1, \ldots, N\}: \qquad y_i \left(x_i^T \beta + \beta_0\right) - 1 \geq 0. \qquad \text{Eq. 3.10}$$

Now, the maximum-margin hyperplane can be determined by solving the following *primal* convex optimization problem:

$$\min_{\beta, \beta_0} \frac{1}{2} ||\beta||^2 \qquad \text{Op. 3.1}$$

subject to:

$$y_i \left(x_i^T \beta + \beta_0\right) - 1 \geq 0, \qquad \forall i \in \{1, \ldots, N\}.$$

However, for reasons of computational speed, usually the corresponding *dual* optimization problem is solved. The Lagrangian dual [14] of problem Op. 3.1 is defined, as follows:

$$\begin{aligned} L(\beta, \beta_0, \alpha) &= \frac{1}{2} ||\beta||^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(x_i^T \beta + \beta_0\right) - 1 \right] \qquad \text{Eq. 3.11} \\ &= \frac{1}{2} ||\beta||^2 - \sum_{i=1}^{N} \alpha_i y_i \left(x_i^T \beta + \beta_0\right) + \sum_{i=1}^{N} \alpha_i, \end{aligned}$$

where α_i denotes the *Lagrange multiplier* of the i^{th} constraint of the primal problem.

Applying the so-called *Karush-Kuhn-Tucker (KKT)* [14] conditions to the Lagrangian function, $L(\beta, \beta_0, \alpha)$, results in the following four equations:

$$\frac{\partial L(\beta, \beta_0, \alpha)}{\partial \beta} = \beta - \sum_{i=1}^{N} \alpha_i y_i x_i = 0, \qquad \text{Eq. 3.12}$$

$$\frac{\partial L(\beta, \beta_0, \alpha)}{\partial \beta_0} = - \sum_{i=1}^{N} \alpha_i y_i = 0, \qquad \text{Eq. 3.13}$$

$$\alpha_i \geq 0, \qquad \forall i \in \{1, \ldots, N\}, \qquad \text{Eq. 3.14}$$

$$\alpha_i \left[y_i \left(x_i^T \beta + \beta_0 \right) - 1 \right] = 0, \qquad \forall i \in \{1, \ldots, N\}, \qquad \text{Eq. 3.15}$$

where Eq. 3.15 is also known as *complementary slackness*.

In sum, the dual optimization problem can be formulated, as follows:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{h=1}^{N} \alpha_i \alpha_h y_i y_h x_i^T x_h \qquad \text{Op. 3.2}$$

subject to:

$$\sum_{i=1}^{N} \alpha_i y_i = 0,$$

$$\alpha_i \geq 0, \qquad \forall i \in \{1, \ldots, N\}.$$

Obviously, there is a Lagrange multiplier α_i for each instance-label pair (x_i, y_i). Therefore, let the optimal solution of the dual problem Op. 3.2 be designated by vector $\alpha^* = (\alpha_1^*, \ldots, \alpha_N^*)$. Then, each sample that satisfies $\alpha_i^* > 0$ is called a *support vector*. Geometrically, these instances are the closest to the determined optimal hyperplane.

Furthermore, let the set of all support vectors be denoted by S. Due to Eq. 3.15, S contains all samples which lie either on the hyperplanes H_1 that is given by $x_i^T \beta + \beta_0 = +1$ or on H_2 which is determined by $x_i^T \beta + \beta_0 = -1$.

The Lagrange multipliers of the remaining observations $i \notin S$ satisfy $\alpha_i^* = 0$. Again, due to Eq. 3.15 the corresponding instances either also lie on H_1, H_2, or on one side of H_1 or H_2.

The optimal bias β_0^* of the separating hyperplane can be computed with the help of an arbitrary support vector $s \in S$ and condition Eq. 3.15:

$$\beta_0^* = y_i - \beta_*^T x_s, \qquad \text{Eq. 3.16}$$

where the optimal normal vector β_* can be determined using Eq. 3.12:

$$\beta_* = \sum_{i=1}^{N} \alpha_i y_i x_i = \sum_{s \in S} \alpha_s y_s x_s. \qquad \text{Eq. 3.17}$$

Finally, the following linear decision function is obtained that enables to predict the class of an unknown instance $x \in \mathbb{R}^p$:

$$f(x) = \operatorname{sign}\left(\beta_*^T x + \beta_0^*\right) = \operatorname{sign}\left(\sum_{i=1}^{N} y_i \alpha_i^* \left(x^T x_i\right) + \beta_0^*\right). \qquad \text{Eq. 3.18}$$

In summary, this kind of linear SVM does not allow any misclassification, i.e., none of the training samples is permitted to lie on the wrong side of the determined hyperplane. That is why, this model is called a *linear hard-margin SVM* [118]. Since, this constraint usually is too restrictive, it has to be softened up to be able to learn SVM models from more complex datasets.

Linear soft-margin support vector machine

Now, it is assumed that the two classes of the given dataset are overlapping, i.e., they are not completely linearly separable. This happens frequently in real-world problems. In that case, the two conditions given by Eq. 3.8 and Eq. 3.9 do not hold for all instances any more. Nevertheless, a hyperplane is searched which separates the two classes at its best.

Hence, the strategy of a *linear soft-margin SVM* is to allow a non-fulfilment of these two constraints, while penalizing each violation by an additional cost term at the same time. This cost is therefore added to the objective function of the corresponding optimization problem. Thus, it is ensured that the smallest possible number of constraints is violated.

Formally, the primal problem is extended with non-negative, so-called *slack variables* $\xi_1, \xi_2, \ldots, \xi_N$, as follows:

$$\begin{aligned}
&\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \qquad \text{Op. 3.3}\\
&\text{subject to:}\\
&y_i\left(x_i^T\beta+\beta_0\right) \geq 1-\xi_i, \qquad \forall i \in \{1,\ldots,N\},\\
&\xi_i \geq 0, \qquad \forall i \in \{1,\ldots,N\},
\end{aligned}$$

where the cost parameter C has to be chosen properly by the user. The higher the value of this regularization parameter C is, the larger becomes the penalty for a misclassification, i.e., for allowing an instance to lie on the wrong side of the hyperplane.

However, the beauty of this modification is that neither the newly introduced slack variables ξ_i nor their Lagrange multipliers do occur in the corresponding dual problem which is given by:

$$\begin{aligned}
&\max_{\alpha} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{h=1}^{N}\alpha_i\alpha_h y_i y_h x_i^T x_h \qquad \text{Op. 3.4}\\
&\text{subject to:}\\
&\sum_{i=1}^{N}\alpha_i y_i = 0,\\
&0 \leq \alpha_i \leq C, \qquad \forall i \in \{1,\ldots,N\}.
\end{aligned}$$

Hence, compared to the dual problem of the separable case which is given by program Op. 3.2, the only difference lies in the fact that the Lagrange multipliers now have an upper bound given by C.

The normal vector that specifies the optimal hyperplane can be computed, as before:

$$\beta_* = \sum_{s\in S}\alpha_s y_s x_s. \qquad \text{Eq. 3.19}$$

Moreover, the linear decision function is still given by:

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{N} y_i \alpha_i^* \left(x^T x_i\right) + \beta_0^*\right), \qquad \text{Eq. 3.20}$$

where the condition $0 \leq \alpha_i^* \leq C$ holds, as mentioned above [77].

Non-linear support vector machine

So far, a SVM is only able to determine a *linear* decision function that separates the two classes as good as possible. Since this is obviously a serious limitation, which would make the algorithm unsuitable for many real-world problems, a SVM has to be enhanced to be also capable to create highly flexible, *non-linear* decision boundaries. This leads to *non-linear SVM* models.

The basic idea behind this modification is based on *Cover's theorem* [121] which states that if a given dataset is not linearly separable in the original feature space, then it can be projected non-linearly into a higher space $\mathscr{H}$ where the transformed data becomes linearly separable with a high probability. Hence, such a non-linear mapping Φ has to be found.

After revising the dual problem Op. 3.4 of the linear soft-margin SVM, it can be noticed that the feature vectors are only contained in the inner product $x_i^T x_h$. Thus, applying a non-linear mapping Φ on the training data would only result in a replacement of the inner product $x_i^T x_h$ by $\Phi(x_i)^T \Phi(x_h)$ [24].

However, this leads implicitly to two new problems: On the one hand, it has to be ensured that there exists an inner product in $\mathscr{H}$ at all. On the other hand, the choice of Φ remains an open question.

In [49] it is shown the *kernel trick* solves both problems at the same time. First, a *kernel function*

$$K(x_i, x_h) = \Phi(x_i)^T \Phi(x_h) \qquad \text{Eq. 3.21}$$

is needed which implicitly computes the inner product in $\mathscr{H}$ without calculating explicitly the coordinates of the data in the latter space. However, according to [25] and as mentioned before, it has to be ensured that $K(x_i, x_h)$ really corresponds with an inner product in $\mathscr{H}$.

Vapnik [118] shows that this is true for all kernel function satisfying *Mercer's condition*. The latter theorem states that each positive semi-definite and symmetric function corresponds to an inner product in $\mathscr{H}$ [25].

Using the kernel trick, the dual optimization problem of a non-linear SVM can be now formulated, as follows:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{h=1}^{N} \alpha_i \alpha_h y_i y_h K(x_i, x_h) \qquad \text{Op. 3.5}$$

subject to:

$$\sum_{i=1}^{N} \alpha_i y_i = 0,$$

$$0 \leq \alpha_i \leq C, \qquad \forall i \in \{1, \ldots, N\}.$$

An unknown instance x can consequently be classified using the resulting non-linear decision function:

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{N} y_i \alpha_i^* K(x, x_i) + \beta_0^* \right). \qquad \text{Eq. 3.22}$$

Ultimately, according to [127], popular choices of the kernel function are

$$K_{linear}(x, \tilde{x}) = x^T \tilde{x}, \qquad \text{Eq. 3.23}$$

$$K_{poly}(x, \tilde{x}) = \left(x^T \tilde{x} + r \right)^d, \qquad \text{Eq. 3.24}$$

$$K_{sigmoid}(x, \tilde{x}) = \tanh\left(\gamma \left(x^T \tilde{x} \right) + r \right), \qquad \text{Eq. 3.25}$$

$$K_{RBF}(x, \tilde{x}) = \exp\left(- \frac{\|x - \tilde{x}\|^2}{2\sigma^2} \right), \qquad \text{Eq. 3.26}$$

where r, d, γ, and σ are kernel specific parameters that have to be tuned by the user. In this work, the application of SVM employing one of the probably most common kernels, i.e., K_{linear} and K_{RBF}, to load spectrum data is studied, exclusively.

Weighted support vector machine

In order to respect the unequal proportion of instances between the two classes datasets suffering from the class imbalance problem, a SVM can be extended to a so-called *weighted SVM* [90]. The new feature of this modification is that it facilitates to set different costs for erroneous predictions of samples from distinct classes.

Thereby, the primal optimization problem can be formulated, as follows:

$$\begin{aligned}
&\min_{\beta,\beta_0,\xi} \frac{1}{2}||\beta||^2 + C_{+1} \sum_{\{i|y_i=+1\}} \xi_i + C_{-1} \sum_{\{i|y_i=-1\}} \xi_i && \text{Op. 3.6}\\
&\text{subject to:}\\
&y_i\left(\Phi(x_i)\cdot\beta+\beta_0\right) \geq 1-\xi_i, \qquad \forall i \in \{1,\ldots,N\},\\
&\xi_i \geq 0, \qquad \forall i \in \{1,\ldots,N\},
\end{aligned}$$

where the two positive constants C_{+1} and C_{-1} denote the penalties for misclassifications of positive and negative class samples, respectively.

According to [26], the corresponding dual problem is given by:

$$\begin{aligned}
&\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{h=1}^{N} \alpha_i\alpha_h y_i y_h K(x_i,x_h) && \text{Op. 3.7}\\
&\text{subject to:}\\
&\sum_{i=1}^{N} \alpha_i y_i = 0,\\
&0 \leq \alpha_i \leq C_{+1}, \quad \text{if } y_i = +1,\\
&0 \leq \alpha_i \leq C_{-1}, \quad \text{if } y_i = -1.
\end{aligned}$$

Thus, the only difference to problem Op. 3.5 is that the upper bounds of the Lagrange multipliers α_i are now dependent on the class labels of samples. The non-linear decision function remains the same. However, the parameters C_{+1} and C_{-1} have to be tuned additionally to the kernel parameters by the user to get a well performing classification model. An major advantage of this *cost-sensitive* [124] approach is that it does not require any further manipulation of the dataset such as oversampling the minority class.

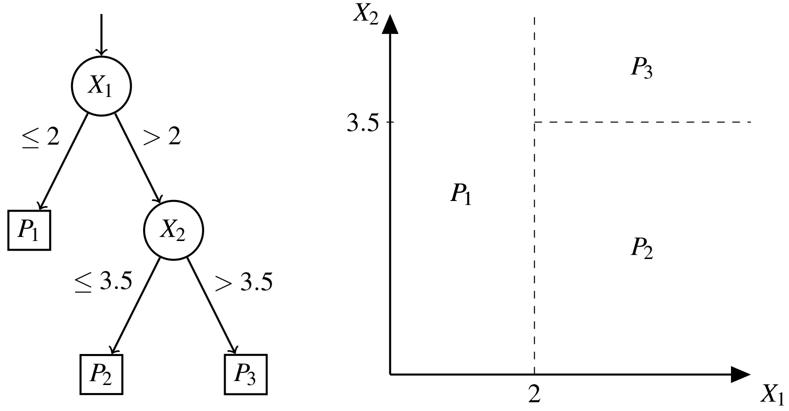


Figure 3.2: Partition of a two-dimensional feature space by means of a binary classification tree (similarly to [110])

Whenever the term SVM is used throughout the rest of this work, a weighted SVM is meant.

3.2.2 Classification tree

Classification trees such as *CART* [20], *C4.5*, or its newer version *C5.0* [96], are a simple but effective way to recursively partition the feature space into a set of non-overlapping, rectangular areas. Each of these areas is then assigned a class label. This label is finally used as prediction for samples falling into this area [110].

Figure 3.2 illustrates the partitioning process of a binary classification tree. It presents the structure of a binary tree which partitions a two-dimensional feature space that is given by the two attributes X_1 and X_2. The partitions created by the tree are named P_1, P_2 and P_3, while the computed *split points*, *cutpoints*, or *thresholds* are 2 for X_1 and 3.5 for X_2. The graphical representation of the tree reveals that the feature space is split at a cutpoint value of 2 in variable X_1. Thereby, all samples i with $x_{i1} \leq 2$ fall into partition P_1, while those with $x_{i1} > 2$ are further divided by a condition on variable X_2 in the next step. The second split is given at a split point of 3.5 in variable X_2. Thus, all observations satisfying the constraint $x_{i1} > 2$ and $x_{i2} \leq 3.5$ are assigned to area P_2, while the remaining samples end up in partition P_3. In a binary classification problem, each area P_j, i.e., each *terminal* or *leaf node* is labelled by a single class $g \in \mathscr{G}$ with $|\mathscr{G}| = 2$. The category g is finally assigned to all the samples falling into area P_j.

In general, it is possible to split in the same variable several times and the order of the variables in which a split is performed is not predefined, i.e., it is possible to split the feature space first in another variable than the first one. Rather the “optimal” pair of variable and split point depends on a so called *impurity measure* [50] that is used to compute the cutpoint values. Moreover, sometimes there are variables in which no split is performed at all.

Next, it is explained how a classification tree is deduced from a dataset. Since this work does not consider any regression problems, the focus lies exclusively on classification trees. At each inner node τ of such a tree, is has to be found out, which variable and which corresponding split point lead to the best separation of the training data $\mathscr{T}_\tau$ that is available at this node. Thereby, a variable and a cutpoint are considered as being optimal for a split if they lead to the highest empirical impurity reduction, e.g., the maximal decrease of the *Gini Index* [50] or the highest *Information Gain* [64].

Let $N_{\tau,c}$ be the number of samples at node τ which belong to class $g \in \mathscr{G} = \{-1,+1\}$. Moreover, let N_τ be the total number of instances that have reached node τ, i.e., $N_\tau = |\mathscr{T}_\tau|$. Then, the proportion of class g instances at node τ is given by

$$\hat{p}_{\tau,g} = \frac{N_{\tau,g}}{N_\tau}. \qquad \text{Eq. 3.27}$$

Now, the impurity measure called *Gini Index* at node τ of tree t can be defined, as follows:

$$G_t(\tau) = 1 - \hat{p}^2_{\tau,-1} - \hat{p}^2_{\tau,+1}. \qquad \text{Eq. 3.28}$$

Furthermore, let q_j be a candidate split point of variable X_j which would lead to the left and right child nodes $\tau_{q_j,L}$ and $\tau_{q_j,R}$ of node τ, respectively. Then, the decrease of the Gini Index that results from this split is given by

$$\Delta G_t(\tau, q_j) = G_t(\tau) - \left(\frac{N_{\tau_{q_j,L}}}{N_\tau} G_t\left(\tau_{q_j,L}\right) + \frac{N_{\tau_{q_j,R}}}{N_\tau} G_t\left(\tau_{q_j,R}\right) \right). \qquad \text{Eq. 3.29}$$

Thus, the impurity reduction measured by the decrease of the Gini Index is equal to the difference of the Gini Index of node τ before splitting minus the weighted mean of the two Gini Indices calculated on the child nodes resulting from splitting τ. Finally, a node τ is split in that the cutpoint which leads to the maximal decrease of the Gini Index. This impurity reduction measure is used in the standard implementation of CART.

On the contrary, the other popular classification tree algorithm called C5.0 creates splits using the Information Gain which is based on the *Shannon entropy*[110]. There are also variants of CART employing this impurity measure. For a binary classification problem the empirical Shannon entropy is defined, as follows:

$$\mathscr{S}_t(\tau) = -\hat{p}_{\tau,-1}\log_2(\hat{p}_{\tau,-1}) - \hat{p}_{\tau,+1}\log_2(\hat{p}_{\tau,+1}). \qquad \text{Eq. 3.30}$$

The corresponding impurity reduction measure is given by

$$\Delta\mathscr{S}_t(\tau, q_j) = \mathscr{S}_t(\tau) - \left(\frac{N_{\tau_{q_j,L}}}{N_\tau}\mathscr{S}_t\left(\tau_{q_j,L}\right) + \frac{N_{\tau_{q_j,R}}}{N_\tau}\mathscr{S}_t\left(\tau_{q_j,R}\right)\right). \qquad \text{Eq. 3.31}$$

Hence, it is possible to grow a tree as long as it is able to classify perfectly each instance of the training set. However, this usually results in a tree that has learned all the details including spurious characteristics that are inherent to the training data, but that is not able to generalize well on unseen data, i.e., the tree is *overfitted* [124] to the training data.

This phenomenon has to be avoided, since a good classifier has to perform well on data that has not been used to create the model. Therefore, there are several strategies for classification trees to avoid overfitting which basically can be categorized into *stopping*, sometimes also known as *prepruning*, and *(post)pruning* methods, respectively [124].

While the tree growing process ends in stopping approaches when a predefined stopping condition is satisfied, the tree is fully grown and reduced afterwards, i.e., pruned back, in the alternative strategies. Thus the methods of the first category prevent the model from becoming to complex, while the algorithms of the second one try to simplify a complex model.

On the one hand, common stopping criteria allow to recursively split the data until

- all leaf nodes contain only samples of the same class *or*
- a constraint on the predefined minimum number of observations that have to reach a leaf is violated *or*
- a constraint on the minimum amount of impurity reduction is violated by all candidate variables [110].

On the other hand, popular operations for pruning are *subtree replacement* and *subtree raising* [124]. In the first operation a selected subtree is replaced by a single leaf node, while in the latter one an inner node is replaced by one of its child nodes, i.e., by the nodes below it. In order to decide whether such a pruning strategy is useful, the expected error rate has to be estimated at both inner and leaf nodes. Thus, an independent validation set, i.e., a set of objects that have not been used to grow the tree could be used to calculate the classification errors at a particular node before and after pruning the tree. This procedure is called *reduced-error pruning* [124] and each inner node is a candidate for pruning. Furthermore, nodes are pruned iteratively beginning with the elimination of the node that leads to the highest increase in classification performance over the validation set [86].

Another possibility is to estimate the error using the training set itself by applying a heuristic that is based on statistical testing [124] and sometimes called *pessimistic pruning* [64]. Thereby, an upper bound for the true error rate at an inner node is estimated for a given confidence level CF. This pessimistic estimate is done before and after pruning. If the estimated error is less after than before pruning, the pruning step is conducted. Usually, the smaller CF, the more pessimistic the estimated error is and the heavier the pruning. Although this procedure is statistically not really valid because of making assumptions on the distribution of the error rate which are not always true, it works quite well in practise [86].

The pruning strategy which is proposed by Breiman et al. [20] is called *cost-complexity pruning*. In this variant, the error rate of the tree is penalized with the term $c_p \cdot (\textit{\# leaf nodes})$, where the user-defined parameter c_p is called the *complexity parameter*. Then, the goal is to find the pruned tree with the lowest penalized error rate [64]. Hence, the higher the values of c_p are set, the more aggressive the pruning is.

The CART implementation employs cost-complexity pruning, while C5.0 uses the pessimistic pruning approach. Moreover, both types of classification trees offer the possibility to incorporate different costs for misclassifications of observations of distinct classes. Thus, the model can be biased towards less frequent classes during the model training [64]. In this way, the class imbalance problem can be respected.

A major difference between CART and C5.0 is that only C5.0 offers *adaptive boosting* [50]. The main idea of this approach is to learn iteratively several trees from the training data, where every new tree focuses on predicting those instances correctly which are misclassified by the previous tree. Therefore, a single decision tree is constructed, first. Then, a second classifier is learned which pays more attention to the objects which are misclassified by the first tree in order to get them right. As a consequence, the second classification tree will generally be distinct from the first one. However, it also will make mistakes on some training objects. Hence, these cases become the focus of attention during construction of the third tree, and so forth. This process stops when a predefined number of *trials* is reached or if the most recent tree is either extremely accurate or inaccurate. Finally, the trees are combined by voting to determine the final class of an object, i.e., each tree votes for its predicted class and from the aggregation of these votes for each class the class assignment is deduced [97].

A big advantage of classification trees is that they can be interpreted easily and that the variables can have domains of different scales, i.e., they are invariant under monotone transformations of the predictor variables [110]. A path from the root of the tree to any of its leaf nodes describes a simple and understandable IF-THEN rule. Hence, the predictions made by a tree can be retraced without problems. However, a major drawback of these trees is that they are very sensitive to small changes in the training data, i.e., they have a high variance. In other words, small changes in the training data can lead to different splits and therefore to a distinct shaped tree. Thus, this instability provokes some uncertainty whether the rules derived from a tree are trustful. This is due to the low complexity of this kind of models [50].

An approach to reduce the high variance of a single tree is to build classifiers as ensembles of many trees, as it is done in a *random forest* [16]. However, the price that has to be paid for the improved stability of this kind of models is their loss of interpretability.

3.2.3 Random forest (RF)

Since Breiman [16] introduced the *random forest (RF)* algorithm in 2001, it has become a state-of-the-art classification and regression method for high-dimensional data. Is has been applied successfully to different kinds of data, such as microarray gene-expression data [58], mental fatigue electroencephalogram time series data [106], or video data [78]. According to [19], the main benefits of this method are, amongst others, its efficient scalability to huge datasets, its excellent performance regarding accuracy compared to current algorithms, its implicit estimation of variable importance in classification tasks, and its capability to handle imbalanced datasets.

The main idea of a random forest is to form an ensemble by pooling many univariate decision trees such as CART or C5.0. Thereby, it makes predictions by averaging the decisions made by each individual tree. Thus, it aims at reducing the variance that is inherent to each base learner, i.e., each individual decision tree.

Since this work only deals with binary classification tasks, the basic steps for fulfilling this kind of analysis are explained exclusively. Figure 3.3 illustrates the standard process of training a random forest model, where the building process of a single tree is shown inside a grey dashed rectangle. Thus, this part of the algorithm has to be repeated consecutively n_{tree} times if a random forest model is desired that is formed by n_{tree} decision trees.

In order to build a single tree, a *bootstrap sample* is drawn from the given dataset $\mathscr{D} = (\mathbf{X}, \mathbf{y}) \in \mathbb{R}^{N \times p} \times \{-1, +1\}^N$ first, where $\mathbf{y}$ contains the class labels $g \in \mathscr{G} = \{-1, +1\}$ for each of the N samples. “Drawing a bootstrap sample” means randomly choosing N records from $\mathscr{D}$ with replacement. The objects in such a bootstrap sample build the training data, while the remaining ones are used to assess the performance of the decision tree. They are called *out-of-bag (OOB)* samples.

Next, a classification tree is learned iteratively from the training data in the following way: At each node, a fixed number of $m_{try} \leq p$ attributes is selected randomly from the set of all explanatory variables $\{X_1, X_2 \ldots, X_p\}$ (cf. step “Variable selection” in Figure 3.3). Then, the Gini Index is used to calculate the best univariate split for the current step, only within these m_{try} variables (cf. step “Grow tree” in Figure 3.3). This learning process is repeated until a predefined stopping criterion is fulfilled or until the tree is maximal. The latter

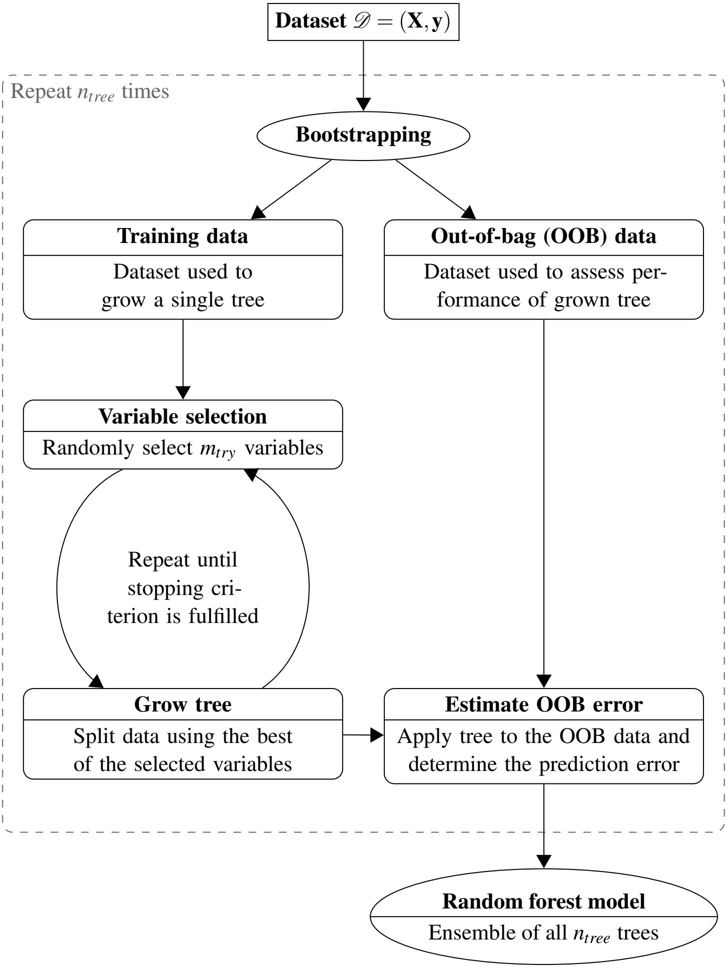

Dataset $\mathscr{D} = (\mathbf{X}, \mathbf{y})$
Repeat n_{tree} times
Bootstrapping
Training data
Dataset used to grow a single tree
Out-of-bag (OOB) data
Dataset used to assess performance of grown tree
Variable selection
Randomly select m_{try} variables
Repeat until stopping criterion is fulfilled
Grow tree
Split data using the best of the selected variables
Estimate OOB error
Apply tree to the OOB data and determine the prediction error
Random forest model
Ensemble of all n_{tree} trees

Figure 3.3: The basic concept of the standard random forest algorithm (inspired by Figure 1 in [15]).

is the case if every sample of the training data can be predicted correctly by the tree. In its standard implementation all trees are grown to the maximum, i.e., no pruning or early stopping steps are performed.

Afterwards, an unbiased estimation of the classification error of the current tree is obtained by calculating the prediction error on the OOB data. Finally,

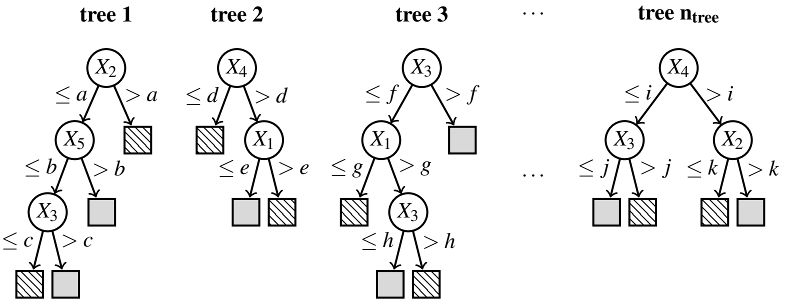


Figure 3.4: A random forest model for a binary classification problem, where the colours of the leaf nodes indicate the class assignment

the resulting n_{tree} trees are pooled and new data is predicted by aggregating their predictions, i.e., each sample is assigned the class the majority of the trees votes for.

In order to take the class imbalance problem into account, one may modify the bootstrapping procedure, which is required for learning each individual tree of a RF, such that this sampling step is performed only for the minority class instances, first, instead of drawing randomly a bootstrap sample from all training objects. Afterwards, a bootstrap sample of the same or at least of a comparable size is drawn from the majority class samples of the training set. Thus, it is ensured that each tree of the RF is learned from an approximately balanced subsample of the training set. This modification leads to a so called *balanced RF* [27]. All the RF models used in this work apply this adapted sampling strategy.

Another possibility is to tweak the voting scheme which is used for deducing the final class assignment from the predictions of the individual trees. Thereby, the RF can be enabled to already predict an instance as a member of the minority class if only a small fraction of the trees of the forest votes for this class [31].

Figure 3.4 illustrates how a random forest model could look like for a dataset containing five variables $X_1, X_2, \ldots, X_5$. The pattern of the quadratic leaf nodes indicates which class is assigned to which decision path. The thresholds for the values of the variables that are checked at each node level are designated by the letters $a, b, \ldots, k$. Having a look at tree 1, it can be seen that each sample

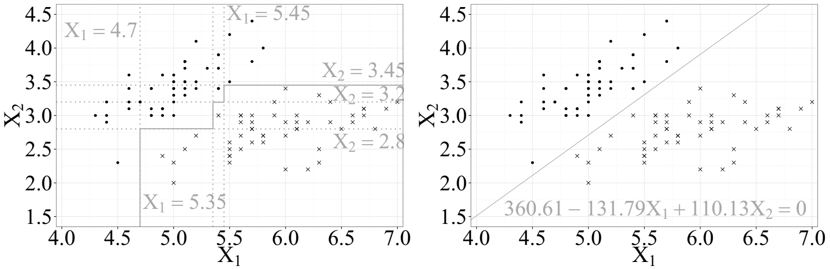
$i \in \{1 \dots N\}$ satisfying the condition $x_{i,2} \leq a \ \wedge \ x_{i,5} > b$ gets assigned the grey class, whereas each sample ℓ with $x_{\ell,2} \leq a \ \wedge \ x_{\ell,5} \leq b \ \wedge \ x_{\ell,3} \leq c$ is predicted as a member of the shaded class. However, the final class assignment depends on the predictions of all trees forming the forest.

3.2.4 Oblique random forest (ORF)

In the previous section, it has already been explained that the original random forest implementation is an ensemble using *univariate* decision trees which separate the variable space by learning hyperplanes that are *orthogonal* to single feature axes. The decision surfaces for these classifiers can be described as "stair-" or "box-like". Therefore, they can be inappropriate for datasets that reside in subspaces lying between the coordinate axes, e.g., collinear data with correlated features. The limitation of testing just a single feature at each non-terminal node can lead in that case to deeply nested decision trees that are not able to separate the classes appropriately [84].

Thus, enabling decision trees to test linear combinations of multiple attributes at each internal node [87] may produce decision boundaries that allow a better separation of these kind of class distributions [84]. Trees of this form are known as *multivariate decision trees* [22]. Since the multivariate tests performed at each inner node are equivalent to separating hyperplanes at an oblique orientation to the feature axes, these kind of trees are also called *oblique decision trees* [4, 87]. By using this nomenclature, it is additionally emphasized that non-linear feature combinations are not considered for testing at non-terminal nodes, whereas this can not be deduced from the term *multivariate* decision trees. Finally, the corresponding ensemble methods are also known as *oblique random forests (ORF)* [84].

Figures 3.5a and 3.5b show scatter plots of the first 100 samples and first two variables of Fisher's famous *Iris Dataset* [36] which describes flowers of two species. Flowers of different species are visualized by distinct shapes, i.e., by filled circles and crosses. Figure 3.5a shows the "stair-like" decision boundary (cf. solid, grey line) of a univariate decision tree that allow a distinction of the instances of the two species. A total of six hyperplanes that are oriented orthogonally to the feature axes (cf. dotted, grey lines) are necessary to form this boundary. Figure 3.5b, however, shows that the same data can be separated more easily by using just a single hyperplane at an oblique orientation to the



(a) Orthogonal decision boundary. **(b)** Oblique decision boundary.

Figure 3.5: Visualization of orthogonal and oblique split directions resulting from (a) uni- and (b) multivariate classification trees

feature axes (cf. solid, grey line). Thus, for some datasets it might be useful to allow multivariate splits at a non-terminal node of a decision tree in order to get a simpler tree and an improved classification result.

In general, RF and ORF models have the same building process which is illustrated in Figure 3.3. However, the main and only difference lies in the way they grow each individual tree (cf. step “Grow tree” in Figure 3.3), i.e., how they split the data. While RF models determine the best univariate split within the set of the m_{try} randomly chosen variables at each inner node level, node splits in ORF models are based on the information that is stored in several, possibly all these m_{try} features. Referring to [91], potential interactions between variables can be captured better by these oblique splits and thus lead to improvements regarding the classification performance.

In a two-class problem, these binary splits are determined by employing multivariate models, such as linear discriminative models, at each inner node τ of a tree. Mathematically, at each inner node τ the parameters $\beta_\tau \in \mathbb{R}^{m_{try}}$ and $\theta_\tau \in \mathbb{R}$ of the function

$$\Lambda_{\beta_\tau,\mathscr{I}_\tau,\theta_\tau}(x_i) = \begin{cases} 0 & \text{, if } \beta_\tau^T \left[x_{ij}\right]_{j\in\mathscr{I}_\tau} < \theta_\tau \\ 1 & \text{, otherwise} \end{cases}, \qquad \text{Eq. 3.32}$$

have to be fitted at node τ, while $\mathscr{I}_\tau \subseteq \{1,2,\ldots,p\}$ with $|\mathscr{I}_\tau| = m_{try}$ is the set of indices of the m_{try} variables that are randomly chosen at this node. Thereby, β_τ is the vector of the coefficients which define the projection for the split and θ_τ is the cutpoint of the split at the internal node τ [84, 105]. The split directions β_τ are determined by applying multivariate models to the training data $\mathscr{T}_\tau \subseteq \mathscr{D}$ that is available at node τ, while the optimal threshold θ_τ is computed using an impurity measure, like in a standard RF. In this work, the Gini Index is employed to calculate the optimal value of θ_τ based on the scores $s_\tau = \beta_\tau^T \left[x_{ij}\right]_{i \in \mathscr{T}_\tau, j \in \mathscr{I}_\tau}$, where the slightly misused notation $i \in \mathscr{T}_\tau$ signifies that only instances of $\mathscr{T}_\tau$ are used.

In the upcoming sections, different variants for the required multivariate node models are discussed. Thereby, it is focused on the approaches used in this study. Moreover, the notation is simplified by omitting the index τ, i.e., β is used for β_τ, θ_τ in place of θ, $\mathscr{I}$ instead of $\mathscr{I}_\tau$, and $\mathscr{T}$ signifies $\mathscr{T}_\tau$.

Elastic net regularized linear regression

In [128], the regularization and variable selection technique named *elastic net* is proposed for the first time. It is motivated by regression problems, but can also be used for solving classification tasks without problems. Its most important characteristic is that it employs a penalty term that is a combination of the *ridge regression* [53] and the *lasso* penalty [115].

According to its inventors, a major benefit of this procedure is that it creates well performing, but sparse models, i.e., models that reduce their input space to only a few variables. Furthermore, it tends to either incorporate strongly correlated features altogether or to leave them all out. This is called the *grouping effect* [128].

In [105], such a linear regression model with an elastic net penalty *(linEnet)* is employed to learn the multivariate split directions $\beta \in \mathbb{R}^{m_{try}}$ at each node τ of a tree in the random forest. Mathematically, the elastic net penalty $P_\alpha(\beta)$ is given by

$$\begin{aligned} P_\alpha(\beta) &= (1-\alpha)\frac{1}{2}\left\|\beta\right\|_{\ell_2}^2 + \alpha\left\|\beta\right\|_{\ell_1} \qquad \text{Eq. 3.33} \\ &= \sum_{j=1}^{m_{try}}\left[\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha\left|\beta_j\right|\right], \end{aligned}$$

where $\alpha \in [0,1]$ specifies the proportion of mixture between the ridge regression and the lasso penalty. If α is fixed to zero, only a ridge regression penalty *(linRidge)* is used, whereas setting α to one results in a pure lasso penalty *(linLasso)*. Usually, the parameter α is fixed by the user beforehand, i.e., it is not optimized.

In other words, $P_\alpha(\beta)$ can be interpreted as an α-weighted trade-off between the above mentioned penalty terms. The optimal values of $\beta \in \mathbb{R}^{m_{try}}$, i.e., the optimal split directions, at an inner node using an elastic net penalized linear regression as node model are the solution of the following optimization problem [105]:

$$\min_{\beta_0,\beta} \left\{ \frac{1}{2|\mathscr{T}|} \sum_{(x,y)\in\mathscr{T}} \left(y - \beta_0 - \beta^T \left[\mathbf{x}_j\right]_{j\in\mathscr{I}} \right)^2 + \lambda P_\alpha(\beta) \right\}, \qquad \text{Eq. 3.34}$$

where β_0 is the (unpenalized) intercept, λ is a complexity parameter and $\mathscr{T} \subseteq \mathscr{D}$ is a subset of the training set containing only the labelled instances that have reached the considered node.

Hence, the elastic net extends the standard least squares regression with an additional penalty on the size of the regression coefficients [50]. Moreover it enhances standard ridge regression which has been employed as node model of oblique random forests in [84] in such a way that it does not only result in small values of β_j, but also shrinks pretty small β_j to zero with the help of the additional lasso term.

Thus, it implicitly selects only relevant features leading to a reduction of the dimensionality of the split space. It is also interesting, that if $\alpha = 1$ and if the values of λ are large, only a single non-zero coefficient β_j is obtained. This corresponds to an orthogonal split like in the standard random forest algorithm [105]. In contrast to a pure lasso regularization, the elastic net penalty term also enables the method to perform well in situations where $p >> N$, i.e., where there are many more variables available than instances. According to [84], this phenomenon can occur in deep split nodes, far away from the root.

(Unregularized) logistic regression

(Unregularized) *logistic regression (log)* is a classification method which is related to multiresponse linear regression [124]. Among its advantages are its robustness against outliers, the fact that it only makes weak assumptions on the data and that there are no parameters that have to be optimized. Because of these reasons, it is proposed in [117] to use this technique for growing oblique decision trees.

In the case of a binomial logistic regression model, i.e., if there are only two possible outcomes for the dependent variable Y, like in a binary classification problem, it is supposed that the posterior log-odds of the first against the second class are of the following linear form:

$$\log \frac{P(Y=+1|X=x)}{P(Y=-1|X=x)} = \beta_0 + \beta^T x. \qquad \text{Eq. 3.35}$$

Usually, *maximum likelihood estimation* [50] is applied to fit these models. Thereby, the corresponding log-likelihood function for a node split model is given by

$$\begin{aligned}\ell(\beta_0,\beta) = \sum_{(x,y)\in\mathscr{T}_\tau} \Big[y \left(\beta_0 + \beta^T \left[\mathbf{x}_j\right]_{j\in\mathscr{I}}\right) - \qquad \text{Eq. 3.36}\\ -\log\left(1+\exp\left(\beta_0+\beta^T\left[\mathbf{x}_j\right]_{j\in\mathscr{I}}\right)\right)\Big].\end{aligned}$$

This function has to be maximized to obtain the optimal values of the intercept $\beta_0 \in \mathbb{R}$ and the split directions $\beta \in \mathbb{R}^{m_{try}}$.

If the condition $p >> N$ holds for $\mathscr{T}$ at some node, unregularized logistic regression is prone to overfitting, though. Since this phenomenon can occur in deep split nodes, as has already been mentioned in the previous section, this characteristic is a major drawback of this technique. However, this problem can be overcome by imposing an additional penalty term in the objective function of logistic regression such as a ridge, lasso, or elastic net penalty.

Elastic net regularized logistic regression

Imposing an elastic net penalty in the objective function of the logistic regression leads to an *elastic net regularized logistic regression (logEnet)*. Formally, this regularized technique is given by

$$\min_{\beta_0,\beta} -\frac{1}{|\mathscr{T}|}\ell(\beta_0,\beta) + \lambda P_\alpha(\beta). \qquad \text{Eq. 3.37}$$

Analogue to the elastic net-regularized linear regression is a *logistic regression with a ridge penalty (logRidge)* obtained by setting α to zero, while fixing $\alpha = 1$ results in a *lasso-penalized logistic regression (logLasso)*.

Partial least squares regression

The idea behind *partial least squares regression (pls)* [80, 125] is to compute derived directions that have a strong correlation with the response variable, i.e., the class vector in classification, as well as a high variance [124]. These directions can be calculated iteratively by only using dot product operations, as follows [50]:

1. Standardize each variable $\mathbf{x}_j$ to zero mean and unit variance.
2. Set $\hat{\mathbf{y}}^{(0)} = \bar{y}\mathbf{1}_N$, where $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$ and $\mathbf{1}_N = (1, 1, \ldots, 1)^T \in \mathbb{R}^N$. Additionally, set $\mathbf{x}_j^{(0)} = \mathbf{x}_j$ for all $j = 1, 2, \ldots, p$.
3. For $m = 1, 2, \ldots, n_{comp}$
   a) Compute for each observed variable $\mathbf{x}_j$ the coefficient ϕ_{mj} for the m^{th} partial least-squares direction, as follows:

$$\phi_{mj} = \mathbf{y}^T \mathbf{x}_j^{(m-1)}. \qquad \text{Eq. 3.38}$$

   b) Construct the m^{th} partial least-squares direction, as follows:

$$\mathbf{z}_m = \sum_{j=1}^{p} \phi_{mj} \mathbf{x}_j^{(m-1)}. \qquad \text{Eq. 3.39}$$

   c) Regress $\mathbf{y}$ on $\mathbf{z}_m$:

$$\hat{\mathbf{y}}^{(m)} = \hat{\mathbf{y}}^{(m-1)} + \hat{\theta}_m \mathbf{z}_m, \qquad \text{Eq. 3.40}$$

   where the coefficients $\hat{\theta}_m$ are given by

$$\hat{\theta}_m = \frac{\mathbf{y}^T \mathbf{z}_m}{\mathbf{z}_m^T \mathbf{z}_m}. \qquad \text{Eq. 3.41}$$

d) Orthogonalize each $\mathbf{x}_j^{(m-1)}$ with respect to $\mathbf{z}_m$:

$$\mathbf{x}_j^{(m)} = \mathbf{x}_j^{(m-1)} - \left[\frac{\mathbf{z}_m^T \mathbf{x}_j^{(m-1)}}{\mathbf{z}_m^T \mathbf{z}_m} \right] \mathbf{z}_m. \qquad \text{Eq. 3.42}$$

4. Finally, for $n_{comp} \leq p$ partial least-squares directions, the desired linear coefficient β are recovered from

$$\hat{\mathbf{y}}^{(n_{comp})} = \mathbf{X}\beta. \qquad \text{Eq. 3.43}$$

This method is popular in many research fields, since it is able to confront the situation $p << N$, i.e., when there are many, possibly correlated predictor variables, and comparatively few samples [85]. Thus, it is another appropriate method for computing the split directions in an oblique decision tree, where this situation can occur in deep split nodes. It has to be noted, that the number of components *ncomp*, sometimes also call *latent variables*, has to be provided by the user.

Linear support vector machine

In Section 3.2.1, a linear soft-margin SVM *(svm)* has been introduced. Do et al. [34] successfully apply this classifier to compute oblique splits at the internal nodes of an oblique decision tree. In particular, they use a so called *linear proximal SVM* [41] for reasons of computational cost. However, this method is computationally expensive because a quadratic programming problem has to be solved to determine the optimal split directions.

3.3 Fundamentals of feature selection

Referring to [46], *variable selection*, also known as *feature* or *attribute selection*, has become more and more important in the past few decades, because the dimensionality of the studied datasets has steadily increased during the same period. However, usually many of the recorded attributes are irrelevant or redundant with respect to the Data Mining or Machine Learning task. While more features may provide more information on the one hand, they may also confuse an algorithm and thus can have a negative impact on the quality of the results of this algorithm.

Hence, among the most important objectives of variable selection are

- improving the computational performance of a model;
- facilitating a better understanding of the dataset;
- improving the quality of a model;
- reducing the requirements for data storage and measurement.

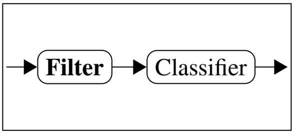
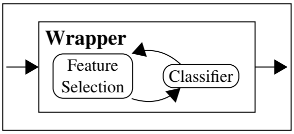
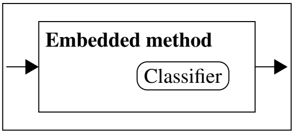
A common way to categorize variable selection methods is to divide them into *filters*, *wrappers*, and *embedded methods*. According to [13], these three groups of algorithms can be described, as follows:

- *Filters* exploit the characteristics of the data independent of the ultimate choice of the induction algorithm. Thus, feature selection is performed as a preprocessing step. Popular representatives are *correlation-based feature selection (CFS)* [48], and *ReliefF* [62].
- *Wrappers* evaluate the usefulness of variables with respect to a given learning machine, e.g., a classifier. Hence, they tune the final induction algorithm during the feature selection process. A famous wrapper is *Recursive Feature Elimination for Support Vector Machines (SVM-RFE)* [47].
- *Embedded methods* perform feature selection implicitly during the model training process. Therefore, they are specific to certain machine learning techniques. Common techniques are decision trees such as CART and C5.0.

Table 3.2 summarizes the main advantages and disadvantages of the three categories of feature selection methods. While filters are among the fastest attribute selection methods, they often do not lead to the best classification performance because they work independently of the choice of the classifier. On the contrary, wrappers optimize the set of chosen variables for a specific classification algorithm and thus often lead to better predictions.

Wrappers can be further subdivided, dependent on the search strategy they use to exploit the attribute space. Either they employ a *forward selection* or a *backward elimination* strategy [46]. The former technique starts with an empty set of variables and consecutively increases this set by adding features to it based on their relevance for the classification performance. In contrast, backward

Table 3.2: Major categories of feature selection techniques (in the context of classification) (cf.[13])

Category	**Advantages**	**Disadvantages**
	• Independent of the classifier • Computationally fast • Good generalization ability	• No interaction with the classifier
	• Interaction with classifier • Captures feature dependencies	• Computationally expensive • Risk of overfitting • Classifier-dependent selection
	• Interaction with classifier • Computationally faster than wrappers • Captures feature dependencies	• Classifier-dependent selection

elimination sequentially removes the least important variables from the set of all attributes. Both variants suffer from the fact that once a variable has been incorporated into or eliminated from the set of variables, it can not be removed from or added to this set in a later step, respectively. Moreover, the price that has usually to be paid for the higher prediction accuracy, is a strong increase in computational time of these techniques.

Finally, embedded methods can be seen as a trade-off between filters and wrappers. On the one hand, they are faster than wrappers, but still interact with the induction algorithm. On the other hand, they are usually slower than filters and do not provide a generic variable selection, but a classifier-specific one.

In this work, no filter methods are considered because the focus lies on achieving the best classification performance. From the set of embedded methods, CART, C5.0, and the random forest algorithm are studied, whereas a popular recursive feature elimination process that employs a SVM has been selected as representative of the wrapper methods. In the following section, approaches of variable selection in classification trees and random forests are introduced, followed by the explanation of the studied wrapper approach, i.e., of SVM-RFE.

3.3.1 Variable importance in tree-based classifiers

Tree-based classifiers have the nice property that they implicitly perform feature selection. During the learning process only the most informative variables are kept using impurity reduction methods, as explained in Sections 3.2.2 and 3.2.3. Thus, only a small subset of "strong" features is kept, i.e., noise variables are not used in the final models [17].

Ensembles of classification trees such as random forests, additionally offer several measures of *variable importance* which help to rank the selected variables from the most to the least relevant ones. The variable importance measures, which are studied in this work, are briefly explained in the following.

Gini importance

The Gini Index is the default criterion for splitting nodes during the growing process of a random forest. Its definition has already been given in Section 3.2.2. On the basis of the computed decrease of the Gini Index resulting from the determined optimal splits, a fast measure of variable importance can be obtained, as follows [83]:

$$GI\left(X_j\right) = \sum_t \sum_\tau \Delta G_t\left(\tau, q_j^*\right), \qquad \text{Eq. 3.44}$$

where q_j^* denote the optimal split points of variable X_j that are contained in the trees t which form the forest. In other words, the Gini importance can be calculated by accumulating for all nodes τ of all trees t in the forest, the decreases of the Gini Index individually for all variables X_j. Thus, it measures for each variable its overall discriminative power for the studied classification task [83].

Permutation importance

The variable relevance measure called the *Permutation Importance Index* [44] takes advantage of the OOB samples of each tree t in the forest. The set of these instances is denoted by OOB_t.

First, the misclassification rate on OOB_t is determined by predicting all the corresponding feature vectors $(x_i)_{i\in OOB_t}$ using tree t. This error is designated by $err(OOB_t)$. Afterwards, the observed values for variable X_j of the instances $i \in OOB_t$ are permuted randomly. For simplicity, this permutation sample is denoted by $\overline{OOB}_t^j$. Ultimately, also the misclassification rate of $\overline{OOB}_t^j$ is calculated, i.e., $err\left(\overline{OOB}_t^j\right)$. In sum, the permutation importance of variable X_j is given by

$$PI(X_j) = \frac{1}{n_{tree}} \sum_t \left(err\left(\overline{OOB}_t^j\right) - err(OOB_t) \right). \qquad \text{Eq. 3.45}$$

Balanced permutation importance

In order to take the class imbalance problem also in the variable importance measure into account, it is possible to calculate the above described Permutation Importance Index individually for each class $g \in \{-1,+1\}$. Thereby, only OOB samples belonging to class g are considered. For class g and variable X_j this class-specific index is formally given by

$$PI_g(X_j) = \frac{1}{n_{tree}} \sum_t \left(err\left(\overline{OOB}_{g,t}^j\right) - err(OOB_{g,t}) \right), \qquad \text{Eq. 3.46}$$

where $OOB_{g,t}$ and $\overline{OOB}_{g,t}^j$ denote the OOB samples of class g of tree t before and after randomly permuting the observed values for variable X_j, respectively. Inspired by the definition of the BAC measure, a new variable importance measure is defined based on these class-specific Permutation Importance Indices, as follows:

$$BPI(X_j) = 0.5 \cdot \left[PI_{-1}(X_j) + PI_{+1}(X_j)\right]. \qquad \text{Eq. 3.47}$$

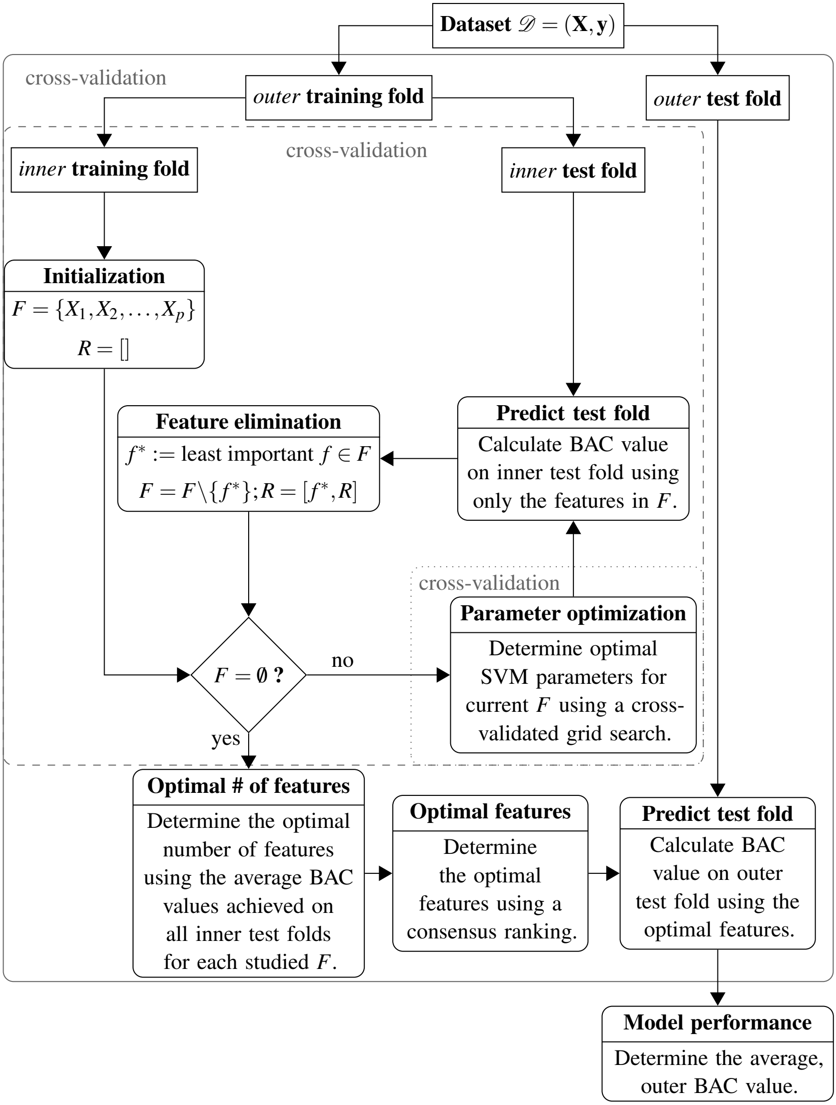

Dataset $\mathscr{D} = (\mathbf{X}, \mathbf{y})$
cross-validation
outer training fold
outer test fold
cross-validation
inner training fold
inner test fold
Initialization
$F = \{X_1, X_2, \dots, X_p\}$
$R = []$
Feature elimination
$f^* :=$ least important $f \in F$
$F = F \setminus \{f^*\}; R = [f^*, R]$
Predict test fold
Calculate BAC value on inner test fold using only the features in F.
cross-validation
Parameter optimization
Determine optimal SVM parameters for current F using a cross-validated grid search.
$F = \emptyset$?
no
yes
Optimal # of features
Determine the optimal number of features using the average BAC values achieved on all inner test folds for each studied F.
Optimal features
Determine the optimal features using a consensus ranking.
Predict test fold
Calculate BAC value on outer test fold using the optimal features.
Model performance
Determine the average, outer BAC value.

Figure 3.6: Workflow for determining the optimal set of variables and for assessing the performance of the final SVM model using RFE

3.3.2 Recursive feature elimination (RFE)

Figure 3.6 presents the individual steps of the *recursive feature elimination (RFE)* process as it is applied in this work in combination with SVM models. This procedure will be denoted by *SVM-RFE* in the following. The studied variant of SVM-RFE differs slightly from the original one that is proposed by Guyon et al. [47]. Here, the version that can be found in [64] is utilized because it is shown in [3] that Guyon's approach may lead to biased performance estimates.

The grey-coloured solid, dashed, and dotted rectangles, respectively, in Figure 3.6 highlight which steps of the algorithm are executed using stratified CV in order to avoid biased results. Since the CV loops are nested, it can already be seen that this algorithm is computationally very expensive.

The outermost CV (cf. grey, solid rectangle) is used to asses the performance of the final model, while the one, higlighted by a grey, dashed rectangle is used to determine the optimal number of features that are selected for building the final model. Finally, the innermost CV (cf. grey, dotted rectangle) optimizes the SVM parameters on a predefined parameter grid.

After pointing out the three main parts of the algorithm, the heart of this approach, i.e., the recursive feature elimination process that is embedded in the grey, dashed rectangle is explained next. Thereby, the set of features that currently survived the elimination process is denoted by F, whereas R designates a list that collects the eliminated features. Therefore, R ultimately contains all features, which are ordered according to their variable importance. The most important feature is inserted last and thus will be the leftmost element of R.

In the beginning of the feature elimination process, R is empty and F contains all available features, i.e., $F = \{X_1, X_2, \ldots, X_p\}$. Then, each iteration of the subsequent elimination procedure starts with a check if the set of survived features F is empty. If this is true, the elimination process is stopped. Otherwise, the optimal SVM parameters are determined for the current set F performing a CV grid search over a predefined parameter grid. The parameters that achieve the highest cross-validated BAC value are used to fit the SVM model in this work. In general, however, other performance measures could be used without limitations. Then, the inner test fold which has not been used to determine the optimal parameters is predicted to assess the performance of the SVM using the current features in F. The resulting performance value is recorded for later use, afterwards. The process continues with the determination of the least important feature $f^* \in F$ which has the following property [77]:

$$f^* = \arg\min_{f \in F} \left| \sum_{i,h=1}^{N} \alpha_i \alpha_h y_i y_h K(x_i, x_h) - \sum_{i,h=1}^{N} \alpha_i \alpha_h y_i y_h K\left(x_{i,F\setminus\{f\}} - x_{h,F\setminus\{f\}}\right) \right|, \qquad \text{Eq. 3.48}$$

where $x_{i,F\setminus\{f\}}$ denotes the feature vector of instance i containing the observed values for all features in F with exception of the one for variable f. Hence, f^* is the variable that leads to the minimum change in the objective function of the SVM.

Next, f^* is eliminated from set F and inserted in list R, i.e., $F = F\setminus\{F^*\}$ and $R = [f^*, R]$. Finally, the current iteration ends with the repeated test whether the reduced feature set F is empty. As long as this stopping condition does not hold, the whole parameter optimization and feature elimination process is repeated using the reduced feature set $F\setminus\{f^*\}$.

As soon as the feature elimination process has been performed for all folds of the inner CV, the optimal number of features that has to be used for building the final model is determined. For this purpose, the mean values of the recorded BAC values that have been achieved on the inner test folds are computed for each size $s = |F|$ that has been occurred during the elimination process. The number of features s that maximizes the average BAC value, is declared as being optimal.

It has to be noted, that until now, it is not known which s features have to be selected. However, during each inner CV loop a list R has been created that contains the features sorted by their importance. These ranking and the corresponding importance values of the features are ultimately used to calculate a consensus ranking leading to the set of optimal features. In this work, the best features are determined by the mean importance values that the s top ranked features achieved for each fold. Obviously, the intersection of these lists R must not necessarily contain exactly the same features. In that case, features that are not contained in a certain list R, but in any of the others, are assigned an importance value of zero. Thereby, features that are in all lists R have a higher chance to be in the final list of optimal features.

Finally, a SVM is fit based on the optimal s features using the same parameter grid as before. Then, the outer test fold is predicted. The mean values of the obtained BAC values estimate the performance of the final SVM model, i.e., when it is trained on the whole dataset $\mathscr{D}$.

3.4 A new RF based classification and feature selection framework

Figure 3.7 presents the workflow of the newly proposed classification and feature selection framework that is based on a random forest. It employs stratified CV to asses the final model performance which is measured by the average BAC value in this work. However, it is not limited to this performance measure. All steps within this *outer* CV loop are surrounded by a grey-coloured, dashed rectangle.

For each outer training fold, it is searched for the optimal parameters of the RF model using a predefined parameter grid. Another stratified, *inner* CV is employed for this search. In Figure 3.7, this CV loop is depicted by a grey, solid rectangle. In this work, a balanced RF model is used to take the class imbalance problem of the studied datasets into account. Since this only leads to more parameters that have to be tuned, the proposed framework is not restricted to this special type of RF model. Moreover, also other variants such as ORF models can be used as long as they offer a possibility to calculate variable importances implicitly.

As soon as the optimal parameters among the values provided by the grid have been found, 500 RF models are built using these parameters. Then, for each of the 500 RF models, an importance index, e.g., the Permutation Importance Index, is used to measure the importance of each available variable. Next, the median value of the obtained 500 importance index values is calculated for each variable X_j to get stable importance values. Finally, all the variables are ranked using these median importance values. This step is inspired by [44].

Afterwards, a computationally demanding forward feature selection strategy is applied to each outer training fold to determine the optimal number of the most important variables that have to be kept for building the final model. Thereby, a RF model is trained successively on the r top-ranked features with

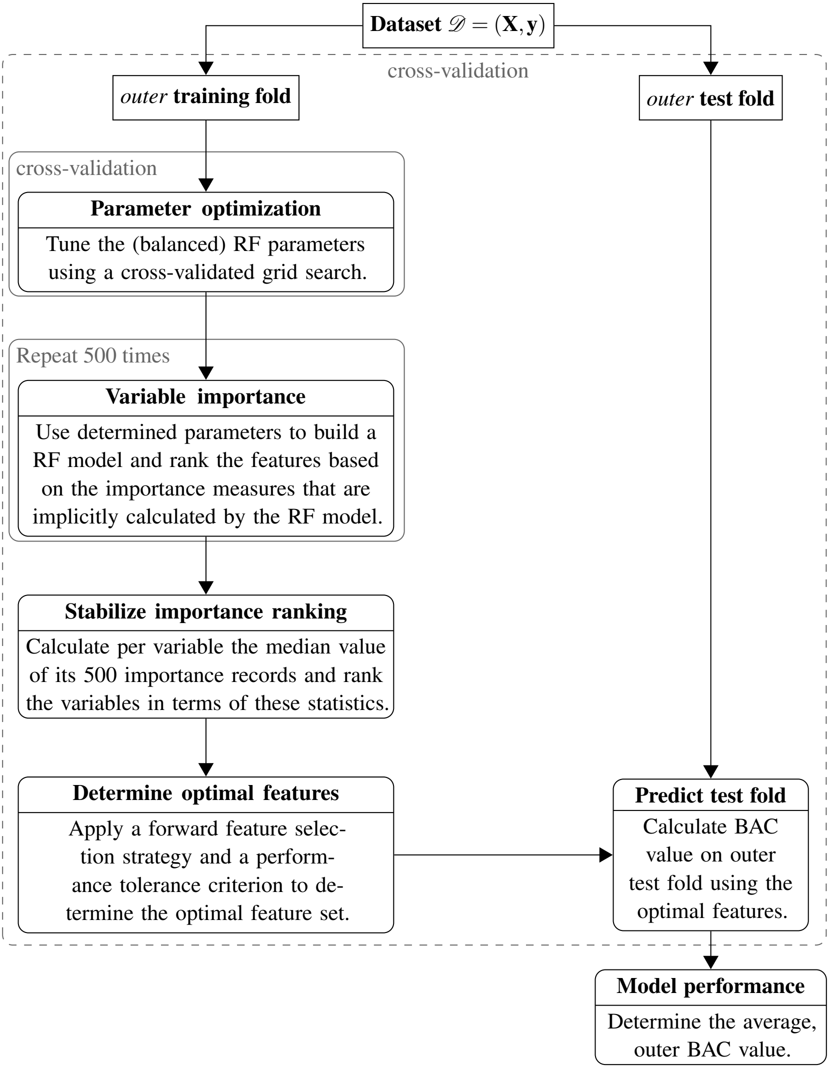

Dataset $\mathscr{D} = (\mathbf{X}, \mathbf{y})$
cross-validation
outer training fold
outer test fold
cross-validation
Parameter optimization
Tune the (balanced) RF parameters using a cross-validated grid search.
Repeat 500 times
Variable importance
Use determined parameters to build a RF model and rank the features based on the importance measures that are implicitly calculated by the RF model.
Stabilize importance ranking
Calculate per variable the median value of its 500 importance records and rank the variables in terms of these statistics.
Determine optimal features
Apply a forward feature selection strategy and a performance tolerance criterion to determine the optimal feature set.
Predict test fold
Calculate BAC value on outer test fold using the optimal features.
Model performance
Determine the average, outer BAC value.

Figure 3.7: Workflow for determining the optimal set of variables and for assessing the performance of the final RF model using the newly proposed RF based classification and feature selection framework

$r = 2, 3, \ldots, 200$, while the parameter m_{try} of the RF model is optimized again over the set of values $\{1, 2, \ldots, r\}$. In other words, an exhaustive search is performed for optimizing this RF parameter for each value of r, where the *out-of-bag balanced error rate (OOB-BER)* [31] is chosen as performance criterion. The strong dependency of the classification performance of RF models on this parameter is the reason for tuning it again during this feature selection step. On the contrary, the remaining RF parameters are not re-adapted. Moreover, the forward feature selection process is stopped after having exploited only the 200 top-ranked variables. This is due to the fact that in this work the focus lies on selecting only a small number of important variables. Another reason is that this exhaustive feature selection strategy is computationally burdensome.

Finally, the parameter pair (r, m_{try}) is considered as optimal if it leads to an OOB-BER value, denoted by $\mathit{OOB\text{-}BER}(r, m_{try})$, that satisfies the following "tolerance" condition [64]:

$$\frac{\mathit{OOB\text{-}BER}\,(r, m_{try}) - \mathit{OOB\text{-}BER}_{best}}{\mathit{OOB\text{-}BER}_{best}} \leq 3\%, \qquad \text{Eq. 3.49}$$

where $\mathit{OOB\text{-}BER}_{best} = \min\limits_{r, m_{try}} \left\{ \mathit{OOB\text{-}BER}\,(r, m_{try}) \right\}$.

In other words, the parameter pair (r, m_{try}) is declared as optimal if its achieved OOB-BER value differs from the best one by only three percent. This can be seen as a trade-off between the performance and the complexity of the model, where complexity of the model is regarded as being proportional to the number of features used for building the model. Thereby, it is also intended to avoid overfitting.

3.5 Case study: Classifying component failures of a hybrid car battery

Based on the two real-world datasets 1a) and 2a), a huge case study is conducted to investigate which of the presented classification approaches works best for distinguishing healthy vehicles from those suffering from a failure of a component of the hybrid power-train, i.e., the hybrid car battery in this study. Furthermore, the discussed feature selection techniques are employed to reduce noise and irrelevant information in the data. On the one hand, the classification performance is likely to be boosted, thereby. On the other hand, it is analysed whether the obtained variable set contains attributes that are relevant for the considered component failure from an engineering point of view.

All required implementations are based on several `R` packages with `caret` [64, 65] building the main framework. The latter package is mainly exploited to perform stratified CV, to apply RFE, to optimize the algorithm-specific parameters and execute the code in parallel. The studied SVM models rely on the package `kernlab`[59], while for building CART and C5.0 models the packages `rpart` [114] and `C50` [66] are used. For RF and ORF the packages `randomForest`[69] and `obliqueRF`[82] are utilized, where the latter one is enhanced to be able to draw stratified bootstrap samples of a defined size. Moreover, it is extended with additional logistic and linear node models such as an elastic net regularized linear regression model. Additionally, some of its functionalities are re-implemented in `C++` to improve computational speed.

In the following, the univariate and multivariate random forest approaches are denoted by *rf* and *orf*, respectively. Moreover, subscripts are used to specify the considered variant of each algorithm. Table 3.3 shows all used acronyms for the studied algorithms, a brief description as well as the chapters of this Thesis, where these methods are introduced and explained. Finally, it has to be noted that all ORF variants use the Permutation Importance Index, when it comes to the feature selection part of the newly proposed classification framework.

3.5.1 Parameter optimization

The parameters of all studied classification algorithms are optimized over predefined parameter grids which are given by Tables 3.4, 3.5, and 3.6, respectively. Thereby, nested stratified CV is used to determine the optimal parameter settings, on the one hand, and to assess the performance of each of the resulting models, on the other hand. More precisely, an inner, stratified 5-fold CV is used for optimizing the parameters, while an outer stratified 5-fold CV is utilized to estimate the performance of each model.

Additionally, seeds are used to make the results repeatable and comparable in the best possible way. Both CV loops try to maximize the BAC measure. This performance measure is employed because it respects the class imbalance in-

Table 3.3: The used acronyms for the studied algorithms and the corresponding chapters of this work, where the methods are explained in detail

Acronym	Description	Chapter(s)
rf	balanced random forest	3.2.3
rf_{Gini}	balanced random forest using the Gini Index as variable importance measure	3.2.3 , 3.3.1 , 3.4
rf_{PI}	balanced random forest using the Permutation Importance Index as variable importance measure	3.2.3 , 3.3.1 , 3.4
rf_{BPI}	balanced random forest using the Balanced Permutation Importance Index as variable importance measure	3.2.3 , 3.3.1 , 3.4
$orf_{linEnet}$	balanced oblique random forest using an elastic net regularized linear regression as node model	3.2.4
$orf_{linRidge}$	balanced oblique random forest using ridge regression as node model	3.2.4
$orf_{linLasso}$	balanced oblique random forest using a lasso regularized linear regression as node model	3.2.4
orf_{log}	balanced oblique random forest using unregularized logistic regression as node model	3.2.4
$orf_{logEnet}$	balanced oblique random forest using an elastic net regularized logistic regression as node model	3.2.4
$orf_{logRidge}$	balanced oblique random forest using an logistic ridge regression as node model	3.2.4
$orf_{logLasso}$	balanced oblique random forest using a lasso regularized logistic regression as node model	3.2.4
orf_{pls}	balanced oblique random forest using partial least squares regression as node model	3.2.4
orf_{svm}	balanced oblique random forest using a linear soft-margin SVM as node model	3.2.4
SVM_{linear}	weighted, linear soft-margin SVM	3.2.1
SVM_{rbf}	weighted, non-linear SVM employing a rbf-kernel	3.2.1
$SVM\text{-}RFE_{linear}$	recursive feature elimination using a weighted, linear soft-margin SVM	3.2.1 , 3.3.2
$SVM\text{-}RFE_{rbf}$	recursive feature elimination using a weighted, non-linear SVM employing a rbf-kernel	3.2.1 , 3.3.2
$CART_{gini}$	CART classification tree using the Gini Index as impurity measure	3.2.2
$CART_{info}$	CART classification tree using the Information Gain as impurity measure	3.2.2
$C5.0$	C5.0 classification tree	3.2.2

Table 3.4: Predefined parameter grid for optimizing the SVM based methods *SVM*$_{linear}$, *SVM-RFE*$_{linear}$, *SVM*$_{rbf}$, and *SVM-RFE*$_{rbf}$

Parameter	**Values**
w_{+1}	1
C_{linear}	$10^{-7}, 10^{-6.75}, 10^{-6.5}, \ldots, 10^{1}, 2^{1}, 2^{1.25}, 2^{1.5}, \ldots, 2^{10}$
C_{rbf}	$10^{-5}, 10^{-4}, 10^{-3}, \ldots, 10^{1}, 2^{1}, 2^{2}, 2^{3}, \ldots, 2^{10}$
σ_{rbf}	$2^{-7}, 2^{-6}, 2^{-5}, \ldots, 2^{-1}$

herent to the studied datasets and has a straightforward interpretation at the same time.

First, the potential parameter values that are utilized for tuning the SVM based classifiers are described. They are presented by Table 3.4. Both for *SVM*$_{linear}$, *SVM*$_{rbf}$ and their corresponding RFE approaches, the regularization parameter C_{linear} and C_{rbf} has to be optimized, respectively. Due to the severe imbalance between the samples from the two classes of the studied datasets, a weighted SVM is used in all cases. Without loss of generality, it is only explained how the class-specific penalties C_{+1} and C_{-1} are set in the case of the linear SVM.

Mathematically, the class-specific regularization parameters are specified, as follows:

$$C_{+1} = w_{+1} \cdot \frac{N_{-1}}{N_{+1}} \cdot C_{linear}, \qquad \text{Eq. 3.50}$$

$$C_{-1} = C_{linear}, \qquad \text{Eq. 3.51}$$

where N_{-1} and N_{+1} denote the number of majority and minority class samples in the current training set, respectively. In other words, the penalty for a misclassification of a positive class instance is $w_{+1} \cdot \frac{N_{-1}}{N_{+1}}$ times as high as for an erroneous prediction of a negative class example. This setting is inspired by the “rule of thumb” proposed in [1]. The same approach is used for *SVM*$_{rbf}$ and *SVM-RFE*$_{rbf}$. However, the latter approaches require an additional optimization of the kernel parameter σ.

Next, Table 3.5 shows the parameter values which are used to optimize the classification tree models *CART*$_{gini}$, *CART*$_{info}$, and *C5.0*. For the two variants of *CART*, the value of the complexity parameter c_p, which is used for pruning

Table 3.5: Predefined parameter grid for optimizing the classification tree methods $CART_{gini}$, $CART_{info}$, and *C5.0*

Parameter	Values
w_{+1}	$0.1, 0.2, 0.3, \dots, 2$
c_p	$10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.02, 0.03, \dots, 0.50$
minSplit	$2, 4, 6, \dots, 30$
trials	$1, 2, 3, \dots, 10$
minCases	$1, 2, 5, 10, 20, 25, 50$
CF	$10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.05, 0.10, 0.25, 0.50$

back the tree, is varied in the set $\{10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.02, 0.03, \dots, 0.50\}$. On the contrary, the optimal value of the pruning parameter *CF* in *C5.0* is searched within the values $10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.05, 0.10, 0.25, 0.50$.

Furthermore, in a *CART* a node is only considered as a split if it is reached by at least *minSplit* training samples. Thus, this parameter regulates the complexity of the tree during the growing process. For this parameter, the values $2, 4, 6, \dots, 30$ are examined.

The similar parameter *minCases* that specifies the minimum number of cases that have to be in a terminal node in *C5.0* is adapted among the values 1, 2, 5, 10, 20, 25, and 50. Moreover, 1 to 10 boosting iterations are tried in *C5.0*.

In Chapter 3.2.2 it is explained that the classification methods *CART* as well as *C5.0* offer the possibility to use different penalty factors for punishing misclassifications of instances from distinct classes. In order to tackle the class imbalance problem, misclassifications of positive samples are penalized in this study with a higher cost compared to those of a negative instances, like in the SVM based approaches. While a cost of one is used for false positives here, the cost C_{+1} for a false negative is set to $w_{+1} \cdot \frac{N_{-1}}{N_{+1}}$, where the factor w_{+1} is optimized on the values $0.1, 0.2, 0.3, \dots, 2$.

Finally, Table 3.6 lists the studied parameter values for the RF and ORF variants. While almost each of the studied algorithms has its own, model-specific parameters which have to be optimized, all these ensemble classifiers also share a few adaptable parameters. Thus, the parameters that all RF and ORF variants have in common are discussed, first. They are tuned in the same way.

Table 3.6: Predefined parameter grid for optimizing the different RF and ORF models using univariate and multivariate decision trees (cf. [9])

Parameter	Values	RF/ORF variant
m_{try}	$\frac{\sqrt{p}}{2}$, $\sqrt{p}, 50, 100, 150, \ldots, p$	All
n_{tree}	300	All
$cutoff_{+1}$	$0.10, 0.15, \ldots, 0.50$	All
$cutoff_{-1}$	$1 - cutoff_{+1}$	All
$sampsize_{+1}$	number of minority samples in current cv-training fold	All
$sampsize_{-1}$	$sampsize_{+1}$, $2 \cdot sampsize_{+1}$, $3 \cdot sampsize_{+1}$	All
λ	cf. Section 2.5 in [39]	$orf_{linEnet}$
		$orf_{linLasso}$
		$orf_{linRidge}$
		$orf_{logEnet}$
		$orf_{logLasso}$
		$orf_{logRidge}$
α	0	$orf_{linRidge}$
	0	$orf_{logRidge}$
	0.5	$orf_{linEnet}$
	0.5	$orf_{logEnet}$
	1	$orf_{linLasso}$
	1	$orf_{logLasso}$
n_{comp}	$1, \ldots, \min\{m_{try}, 25\}$	orf_{pls}
C	10^{-3}, 10^{-1}, 1, 10	orf_{svm}

At each internal node of a tree of the ensemble, there are m_{try} out of the p variables randomly selected as candidates for the current node split. However, the functionality of this parameter is slightly different between RF and ORF models, as explained in Chapters 3.2.3 and 3.2.4.

In a RF, a single variable is determined among these m_{try} features that leads to the best (univariate) split, i.e., the remaining $m_{try} - 1$ attributes are not used for that split. In contrast, in an ORF the (multivariate) best split is given by the optimal separating linear hyperplane in the subspace that is defined by these m_{try}

variables, i.e., it is described by a linear combination of these attributes and a cutpoint. According to [19], the parameter m_{try} has the biggest influence on the classification performance of RF models, since it regulates the correlation between the trees as well as the predictive performance of each individual tree in the forest. Thus, it can be seen as the most important parameter of this kind of ensemble classifiers. Besides using the recommended value of $\sqrt{p}$ [69], it is additionally searched for the optimal value of this parameter within the set $\left\{\frac{\sqrt{p}}{2}, 50, 100, 150, \ldots, p\right\}$.

Moreover, the newly proposed classification and feature selection framework incorporates two strategies which try to overcome the class imbalance problem that is inherent to the studied datasets. They have already been explained in Chapters 3.2.3. First, each tree in the forest is trained on an almost balanced bootstrap sample, i.e., the number of minority and majority samples is almost the same in this set, respectively. Secondly, the quorum that regulates the final class assignment of the whole ensemble is tuned, i.e., not only a simple majority voting scheme is considered. These two strategies are accomplished by adapting the class-specific parameters $sampsize_g$ and $cutoff_g$.

The parameters $sampsize_g$ are used to draw stratified and almost balanced bootstraps from the current training set. For each class $g \in \mathscr{G} = \{-1, +1\}$ it specifies the size of the bootstrap that is drawn from this class to grow an individual tree of the forest. For the positive class, the parameter $sampsize_{+1}$ is fixed to the number of positive training samples that are available in the current training fold, whereas $sampsize_{-1}$ can be equal to, twice, or three times the value of $sampsize_{+1}$. Thereby, the class imbalance ratio within each bootstrap lies between approximately 1 and 3.

The class assignment process of the RF or ORF model is altered by the parameters $cutoff_g$. While a simple majority voting scheme is applied in the default case, these parameters can be tweaked to enable the model to assign class g to an instance if less than 50 percent of the n_{tree} trees predict this class as the right one. Without loss of generality, the forest assigns class $+1$ to an object x_i if the following conditions holds:

$$\frac{\hat{p}_{i,+1}}{cutoff_{+1}} > \frac{\hat{p}_{i,-1}}{cutoff_{-1}}, \qquad \text{Eq. 3.52}$$

where $\hat{p}_{i,g}$ is the fraction of the n_{tree} trees in the forest that predict class g for instance x_i. Thus, $\hat{p}_{i,g}$ can be interpreted as the estimated probability that the i^{th} sample belongs to class g [31]. The optimal value for $cutoff_{+1}$ is searched among the values $0.10, 0.15, \ldots, 0.50$, while $cutoff_{-1}$ is set to $1 - cutoff_{+1}$.

Inspired by [84] and for computational reasons the number of trees, n_{tree}, is fixed to 300 in this case study. Moreover, increasing the value of n_{tree} did not lead to significant improvements regarding the classification performance of the RF and ORF models, respectively.

These are the parameters both the RF and ORF models have in common. Next, the parameters which are specific for each ORF variant are discussed.

For the ORF variants which employ regularized linear or logistic regression as node models, i.e., for $orf_{linEnet}$, $orf_{linRidge}$, $orf_{linLasso}$, $orf_{logEnet}$, $orf_{logRidge}$, and $orf_{logLasso}$, the complexity parameter λ has to be provided at each inner node of a tree. Following the proposal made in [39], possible values are directly derived from the training data that is available at the considered node. In this procedure, first the smallest value is computed which leads to a coefficient vector $\beta \in \mathbb{R}^{m_{try}}$ which contains only zeros. This value is denoted by λ_{max}. In [39], it is shown that there is an analytical solution for determining λ_{max}:

$$\lambda_{max} = \frac{\max_i \left|x_i^T y\right|}{N\alpha}, \qquad \text{Eq. 3.53}$$

where α is the user-defined parameter of the elastic net penalty $P_\alpha(\beta)$ that regulates the trade-off between a lasso and a ridge penalty. Afterwards, λ_{min} is set to $0.001\lambda_{max}$. Then, a decreasing sequence of 100 values is calculated on a logarithmic scale, which starts at λ_{max} and ends at λ_{min}.

For the variant orf_{pls} the number of components, n_{comp}, has to be chosen at each internal node. The value of this parameter is optimized on the set $\{1, 2, \ldots, \sqrt{p}\}$.

Finally, for the ORF variant orf_{svm} the regularization parameter C has to be adapted. For this purpose, a small, coarse grid is used which is given by the values $10^{-3}, 10^{-1}, 1$ and 10. It can be seen as a trade-off between computational speed and classification performance.

Table 3.7: Optimal determined parameters and corresponding CV$_{out}$-BAC values achieved on the *outer* stratified 5-fold CV for each of the studied algorithms, **before** applying additionally feature selection

Method	**Parameter**										
	m_{try}	*cut-off*$_{+1}$	*samp-size*$_{-1}$	C	σ	w_{+1}	c_p	*CF*	*tri-als*	*min-Ca-ses/ Split*	**CV$_{out}$-BAC**
rf	100	0.30	2	–	–	–	–	–	–	–	**0.844**
orf$_{linEnet}$	400	0.25	3	–	–	–	–	–	–	–	0.836
orf$_{linRidge}$	150	0.25	3	–	–	–	–	–	–	–	0.821
orf$_{linLasso}$	500	0.25	3	–	–	–	–	–	–	–	0.830
orf$_{log}$	50	0.25	3	–	–	–	–	–	–	–	0.789
orf$_{logEnet}$	550	0.15	3	–	–	–	–	–	–	–	0.829
orf$_{logRidge}$	150	0.20	3	–	–	–	–	–	–	–	0.804
orf$_{logLasso}$	550	0.30	2	–	–	–	–	–	–	–	0.835
orf$_{pls}$	250	0.30	2	–	–	–	–	–	–	–	0.820
orf$_{svm}$	100	0.35	3	–	–	–	–	–	–	–	0.823
SVM$_{linear}$	–	–	–	$10^{.25}$	–	1.0	–	–	–	–	0.783
SVM$_{rbf}$	–	–	–	2^7	2^{-7}	1.0	–	–	–	–	0.784
CART$_{gini}$	–	–	–	–	–	0.8	0.03	–	–	30	0.788
CART$_{info}$	–	–	–	–	–	0.6	0.05	–	–	30	0.786
C5.0	–	–	–	–	–	1.2	–	0.01	1	50	0.754

The best result is highlighted in bold. The numbers ℓ in the column *sampsize*$_{-1}$ indicate that *sampsize*$_{-1}$ is set to $\ell \cdot$*sampsize*$_{+1}$. For the RF and ORF variants, n_{tree} is fixed to 300 and *cutoff*$_{-1}$ is set to $1 -$ *cutoff*$_{+1}$.

3.5.2 Results

This section presents and discusses the results that have been achieved on the studied datasets 1a) and 2a).

Analysis of dataset 1a)

First of all, the results of the analysis of dataset 1a) are presented. Table 3.7 shows for each of the studied algorithm the result of the parameter optimization. Additionally, it contains a performance estimate which is obtained by a stratified 5-fold CV and denoted by CV$_{out}$-BAC. In particular, identical folds are used in this CV for each method to make the estimates as comparable as possible.

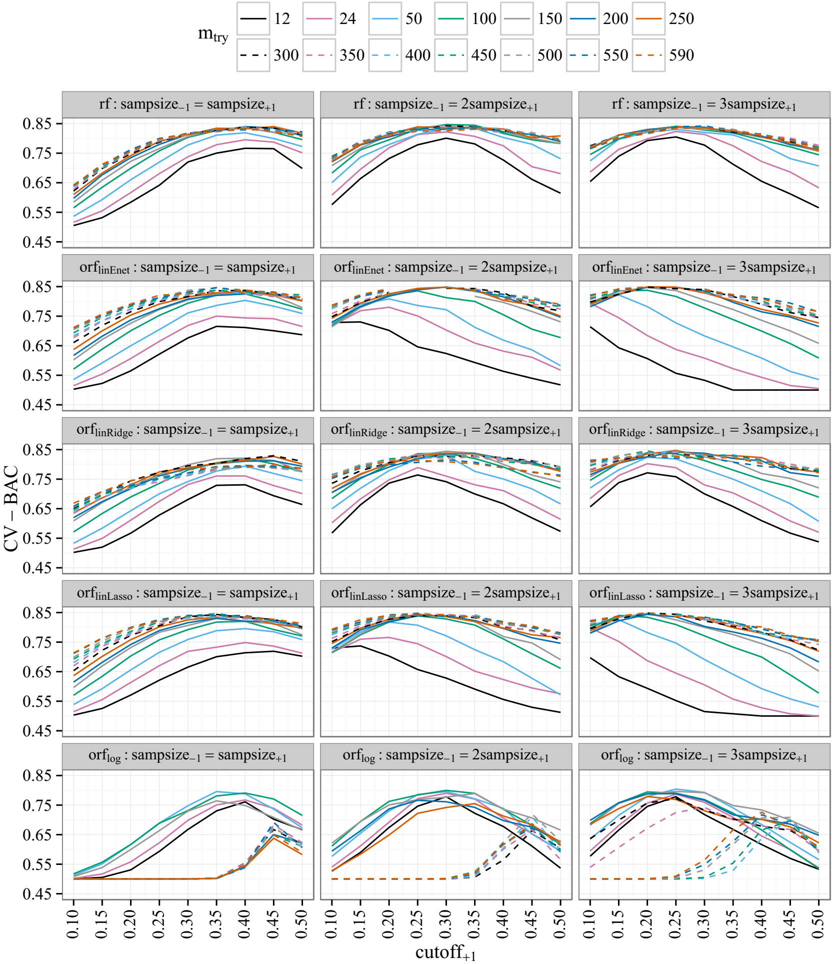


Figure 3.8: Visualization of the parameter optimization for the final models of *rf*, $orf_{linEnet}$, $orf_{linRidge}$, $orf_{linLasso}$, and orf_{log} (cf. [9])

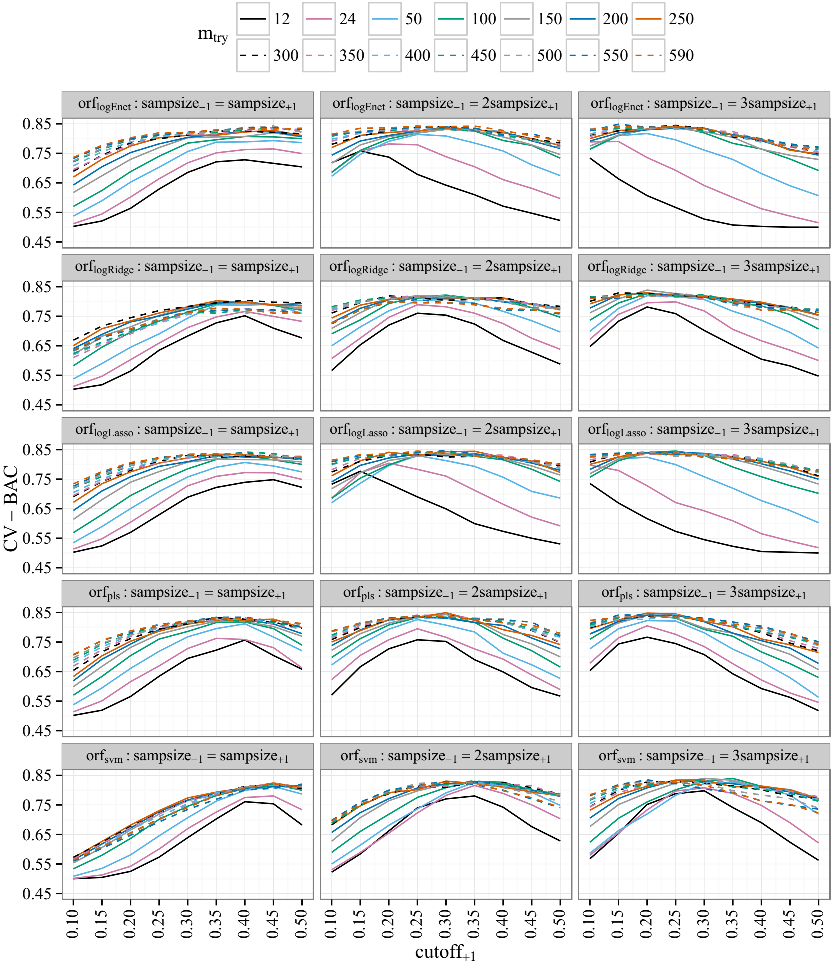


Figure 3.9: Visualization of the parameter optimization for the final models of $orf_{logEnet}$, $orf_{logRidge}$, $orf_{logLasso}$, orf_{pls}, and orf_{svm} (cf.[9])

It has to be noted that no explicit feature selection is performed so far, i.e., neither RFE is applied to SVM_{linear} and SVM_{rbf}, respectively, nor the forward feature selection strategy of the newly proposed framework is conducted for the RF and ORF models. Hence, if all available variables are used, then *rf* clearly outperforms the classification trees and the SVM models, while also being at least slightly better than its ORF competitors using multivariate node splits.

Figures 3.8 and 3.9 show for all studied RF and ORF variants the effect of tuning the parameter m_{try}, $cutoff_{+1}$ and $sampsize_{-1}$ on the performance of the model. The latter quantity is measured by a stratified 5-fold CV that is applied to each training fold of the outer CV loop. It is designated by CV-BAC.

It is interesting to see that the classification performance of each ensemble is strongly dependent on the choice of these three parameters. In particular, it varies a lot with m_{try}, independent on the setting of the other two mentioned parameters. It is notable that high values of m_{try}, i.e., values greater than 300, seem to be more robust against the choice of $cutoff_{+1}$ and $sampsize_{-1}$ than models using a smaller number of variables as candidates for a node split. The only exception is variant *log* which exhibits an inverse behaviour. In general, this result is not surprising because the strong influence of m_{try} on the performance of a random forest model is already known [18].

The dependence of the value of CV-BAC on the choice of the parameters $cutoff_{+1}$ and $sampsize_{-1}$ is even more interesting because it has not been exploited extensively in the literature yet. The plots illustrate that a higher value of $sampsize_{-1}$ only results in high performance values if $cutoff_{+1}$ is decreased at the same time. Moreover, Table 3.7 reveals that the majority of the ORF models achieves the best performance for the highest possible choice of $sampsize_{-1}$.

Another interesting observation is that even for completely balanced stratified bootstrap samples, i.e., if parameter $sampsize_{-1}$ is set equal to $sampsize_{+1}$, the determined optimal value of $cutoff_{+1}$ is less than 0.5 for all variants. Similar results are obtained in [31] for the standard random forest.

Figure 3.10 illustrates for the weighted SVM variants SVM_{linear} and SVM_{rbf} the dependence of CV-BAC on the setting of the regularization parameter C and the kernel parameter σ. It has to be noted that the x-axis of both plots is presented on a logarithmic scale. It can be observed that until a certain point,

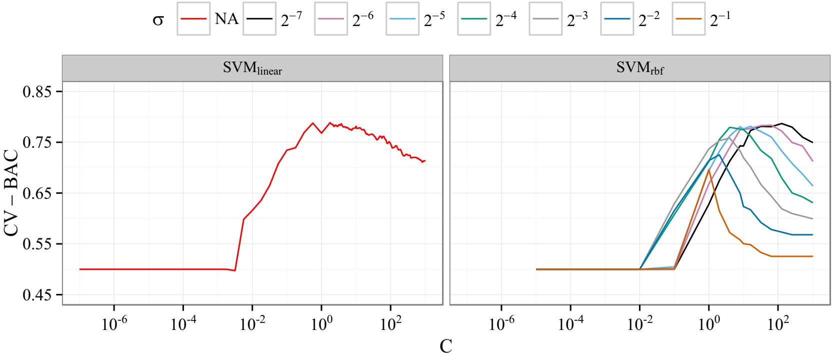


Figure 3.10: Visualization of the parameter optimization for the final models of SVM_{linear} and SVM_{rbf}

an increase of the value of parameter C also leads to a growth of the performance value. However, after this point CV-BAC decreases again. For SVM_{rbf}, this phenomenon remains valid, even if the value of σ is varied. Smaller values of σ seem to lead to higher performance values if C is tuned properly.

Figures 3.11 and 3.12 exhibit the dependence of the estimated classification performance of a CART model employing the Gini Index and the Information Gain as impurity measure, respectively. It can be seen that in both cases lower values of the complexity parameter c_p lead to higher CV-BAC values. Since lower values of c_p correspond to a less aggressive pruning strategy, it can be deduced that larger and more complex trees result in a higher classification performance for this dataset. Regarding the choice of the penalization parameter w_{+1}, it can be noticed that with exception of very low values, i.e., for $w_{+1} \leq 0.2$, CV-BAC is not as sensitive to this parameter as to c_p. As long as c_p is set to a low value, the majority of the studied values of w_{+1} lead to comparably high CV-BAC values. However, when c_p is set to a value greater than approximately 0.2, the estimated classification performance begins to vary a lot with the setting of w_{+}. Thereby, values around $w_{+1} = 1$, i.e., using a penalty for the misclassification of a positive class sample which is approximately $\frac{N_{-1}}{N_{+1}}$ times as high as the one used for the negative class, seems to achieve the best performance.

The parameter *minSplit* does not seem to have a remarkable influence on the classification performance. The value of CV-BAC seems to be very robust

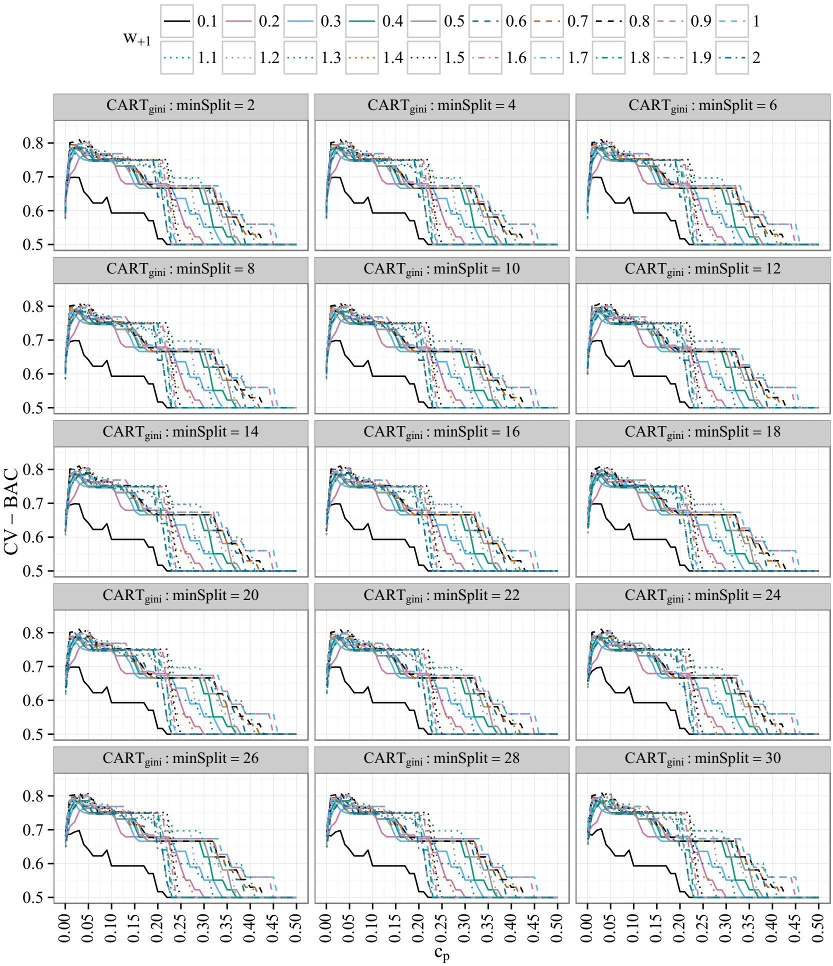


Figure 3.11: Visualization of the parameter optimization for CART using the Gini Index as impurity measure

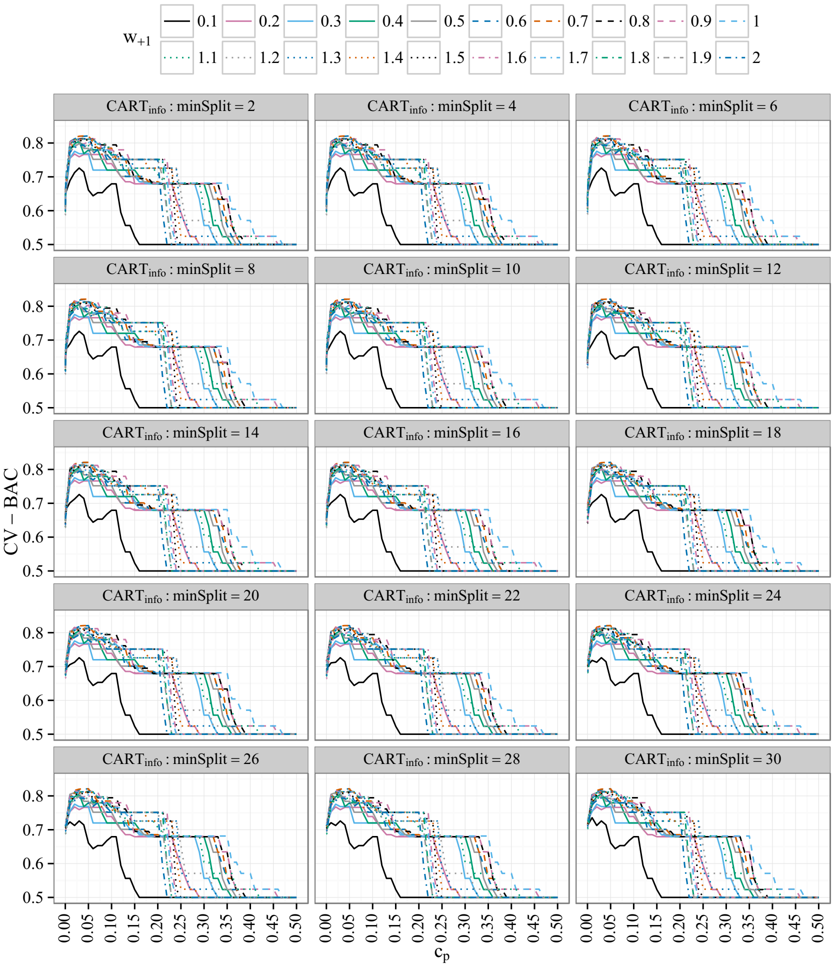


Figure 3.12: Visualization of the parameter optimization for CART using the Information Gain as impurity measure

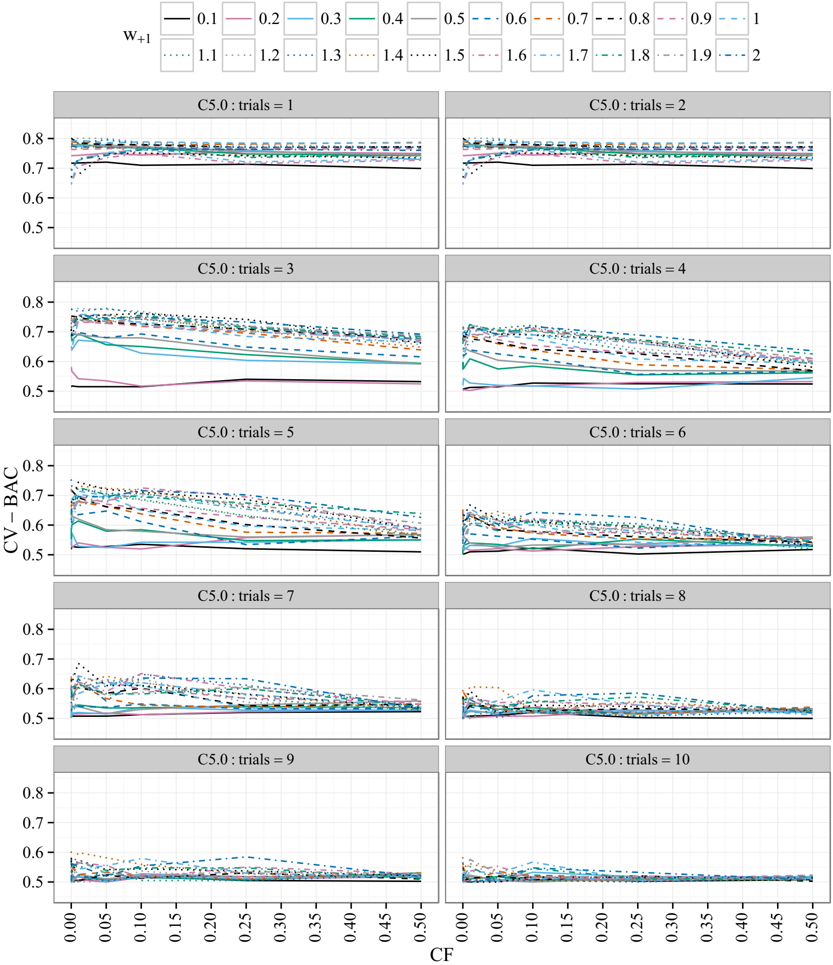


Figure 3.13: Visualization of the parameter optimization for C5.0 (*minCases* = 50)

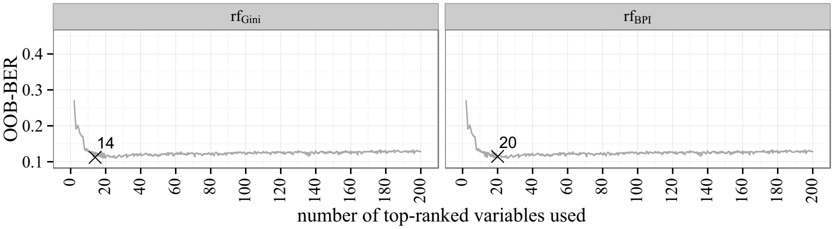


Figure 3.14: The minimal OOB-BER values achieved by each studied RF variant in dependence on the number of top-ranked variables used

against the setting of this parameter for $CART_{gini}$ and $CART_{info}$. The same applies to the choice of the impurity measure. The difference in performance between these two tree models is marginal.

Since the estimated performance of model *C5.0* also seems to be very robust against the choice of parameter *minCases* which is comparable to *minSplit* in *CART*, Figure 3.13 only shows the dependence of the parameter tuning process of *C5.0* for the optimal determined value for *minCases*, i.e., for *minCases* = 50. First, it can be observed that the value of CV-BAC decreases when the number of boosting iterations is increased. This number is specified by the parameter $trials$. Compared to *CART*, the performance of *C5.0* seems to be less sensitive to the choice of its pruning parameter which is denoted by CF. However, at least for a larger number of boosting iterations, higher values of CF, i.e., less aggressive pruning, seem to result in a decreased classification performance. Thus, less complex *C5.0* models seem to perform better when applied to the studied dataset. Again, setting the penalization parameter w_{+1} to low values, results in a worse classification performance, while values around 1 seem to lead to the best models.

Figures 3.14, 3.15, and 3.16 visualize the process of variable selection for the different RF and ORF variants as well as for the two SVM-RFE approaches. It has to be noted that for the forest variants the prediction error is estimated by the OOB-BER values, while for the SVM-RFE models it is given by CV-BER. Moreover, the proposed selection process for the forest variants stops after having considered the 200 top-ranked variables, whereas the recursive feature selection strategy that is used in the SVM-RFE approaches considers successively all available variables.

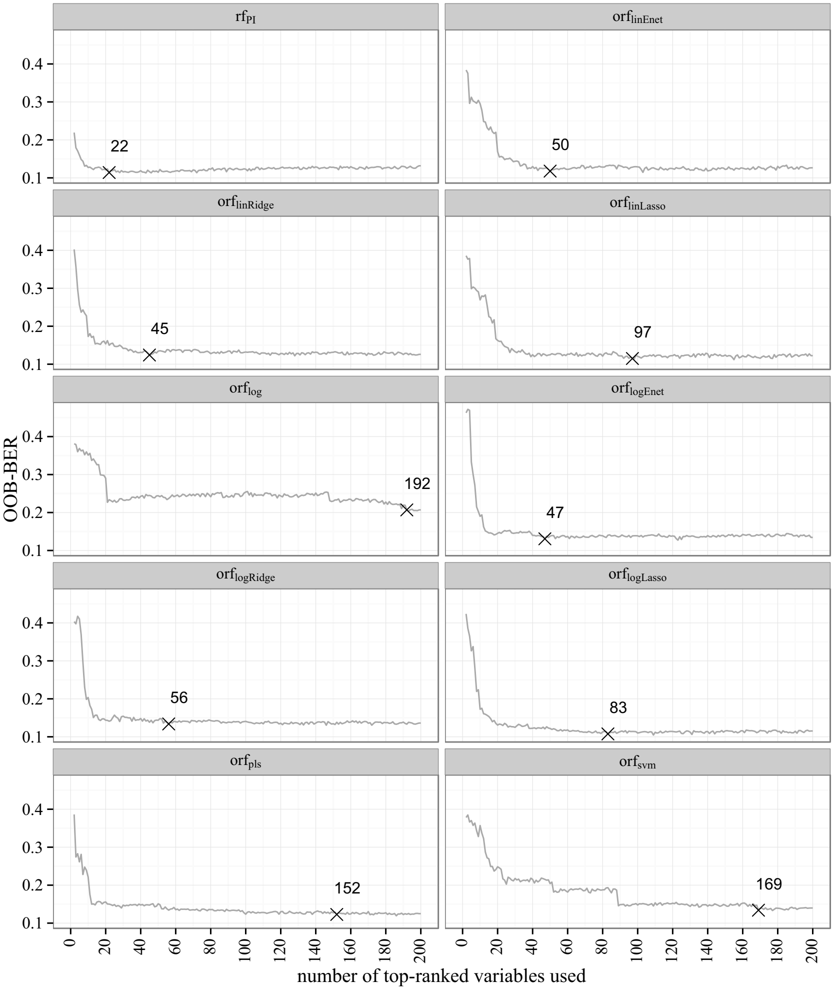


Figure 3.15: The minimal OOB-BER values achieved by each studied ORF variant in dependence on the number of top-ranked variables used (cf.[9])

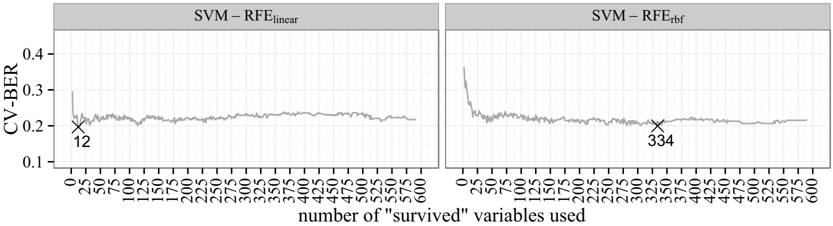


Figure 3.16: The CV-BER values achieved by each studied SVM-RFE variant in dependence on the number of top-ranked variables used

Notably, both for the RF and ORF variants as well as for the SVM-RFE models a small subset of variables is sufficient to obtain an error rate, which is at the level of the overall achieved minimum error of each method. Afterwards, the estimated errors either begin to rise slightly again or stay at approximately the same level.

However, the ORF variants *log* and *svm* are exceptions. Their error rates seem to continuously decrease with increasing numbers of top-ranked variables used for building these models. When comparing the error estimates of the forest variants with the one obtained by the SVM-RFE approaches, it can be seen that the latter two methods are generally operating at a higher error level, i.e., perform worse. The only exception is the ORF model *log* which achieves similar error rates.

In all these figures, the determined optimal number of variables that is used for building the final models is marked by a black cross and labelled by the respective number. It is notable that for the majority of the studied algorithms less than 100 out of the 590 possible, non-zero variables are finally selected. The only exceptions are the ORF variants orf_{log}, orf_{pls}, orf_{svm}, and $SVM\text{-}RFE_{rbf}$.

Table 3.8 summarizes for all studied classification algorithms the final values of the model-specific parameters which have been tuned during the feature selection process and the final number of selected attributes. It also contains the performance estimates of the final models. It is remarkable that the estimated performance of 9 out of the 14 algorithms which apply an external feature selection improves after reducing the number of variables.

The best performance is achieved by method rf_{Gini}. It produces a CV$_{out}$-BAC value of 0.862, while using only 14 variables. This is the fourth lowermost

Table 3.8: Optimal determined parameters and 5-fold cross-validated BAC values (CV$_{out}$-BAC) for each of the 17 studied classifiers, **after** applying the distinct variable selection strategies

	Parameter				
Method	m_{try}	C	σ	**# variables used**	**CV$_{out}$-BAC**
rf_{Gini}	8	–	–	14	***0.862**
rf_{BPI}	4	–	–	20	*0.861
rf_{PI}	9	–	–	22	*0.852
$orf_{linEnet}$	32	–	–	50	*0.837
$orf_{linRidge}$	42	–	–	45	*0.834
$orf_{linLasso}$	96	–	–	97	*0.840
orf_{log}	184	–	–	192	0.751
$orf_{logEnet}$	22	–	–	47	0.822
$orf_{logRidge}$	41	–	–	56	*0.829
$orf_{logLasso}$	49	–	–	83	0.829
orf_{pls}	131	–	–	152	*0.832
orf_{svm}	145	–	–	169	0.814
$SVM\text{-}RFE_{linear}$	–	$2^{9.9}$	–	12	*0.785
$SVM\text{-}RFE_{rbf}$	–	2^{4}	2^{-5}	334	0.781
$CART_{gini}$	–	–	–	**2**	0.788
$CART_{info}$	–	–	–	3	0.786
C5.0	–	–	–	16	0.754

The best result is highlighted in bold. The variants where feature selection improves the classification performance are marked with *.

number of selected attributes. However, the algorithms which employ less variables, namely $SVM\text{-}RFE_{linear}$, $CART_{gini}$, and $CART_{info}$, only achieve CV$_{out}$-BAC values which are less than 0.790. Thus, they perform considerably worse than rf_{Gini}.

In general, the studied RF variants slightly outperform their ORF competitors and are clearly better performing than the SVM-RFE and classification tree approaches, which are on a lower prediction level.

The second goal of this case study is to find out, whether the finally selected variables can be related to failures of the hybrid car battery from an engineering point of view. Since all HEV in the considered dataset are equipped with a Li-ion battery, the most important stress factors that are related to aging of this kind of battery are mentioned, first.

Table 3.9: The 14 variables which are selected by the final model of variant rf_{Gini} and the stress factors to which they can be linked to

Variable	Stress Description
X_{720}	percentage of operating time at a very high temperature of the electric machine
X_{420}	percentage of active short circuits of the electric machine at moderate speed of the machine
X_{412}	percentage of hard starts of the internal combustion engine, i.e., starts significantly delayed in time
X_{545}	total number of active short circuits of the electric machine
X_{415}	operating time of DC-DC converter in boost mode
X_{417}	operating time of DC-DC converter in standby mode
X_{418}	percentage of active short circuits of the electric machine at low speed of the machine
X_{527}	idle time of hybrid car battery
X_{526}	operating time of hybrid car battery
X_{220}	percentage of operating time with a low state-of-charge of the hybrid car battery at a normal battery temperature
X_{542}	operating time of the hybrid car battery with a very high recuperation current flow at low, medium, and high battery temperatures
X_{153}	percentage of operating time of the hybrid car battery with a very high recuperation current flow at high battery temperatures
X_{734}	percentage of operating time of the hybrid car battery at moderate battery temperatures
X_{414}	percentage of intended starts of the internal combustion engine when the voltage of the hybrid car battery is too low

Referring to [51], among those are its operating temperature, depth of discharge, current peaks, temperature during standstill, maximal charging, average discharge and charge current, average state-of-charge, and total discharge. Especially, the battery temperature, its state-of-charge, and the strength of current are identified to be closely related to battery aging.

Table 3.9 presents the 14 attributes that are selected by the best performing model rf_{Gini}. Moreover, it specifies to which kind of stress these variables can be linked. From an engineering point of view, it is interesting that the list of selected features does not contain any variable that can not be associated to stress factors of the studied hybrid component such as the angle of the throttle of the internal combustion engine.

Furthermore, it attracts attention that it is possible to categorize the 14 features into two major groups: The first one is formed by the attributes which can be causal for the failure, while the second one rather contains those that store the effect of an already faulty battery.

Variable X_{153}, for example, belongs to the first category, while attribute X_{414} is included in the second one. Further representatives of the first group are variables X_{527} and X_{526}, i.e., load spectra counting the total idle and operating time of the hybrid car battery.

On the contrary, variable X_{412} which counts the fraction of “hard” starts of the internal combustion engine out the total number of starts of this machine can be assigned to the second category. Thereby, a “hard” start signifies an engine start that is significantly delayed in time. The internal combustion engine employed by the HEV in the dataset is started directly by the electric machine, which in turn receives its power from the hybrid car battery. Thus, it is plausible that a faulty Li-ion battery may result in “hard” starts of the internal combustion engine.

Additionally, active short circuits of the electric machine usually coincide with some misbehaviour of this component. That is why also variables X_{420}, X_{545}, and X_{418} are consistently contained in the second category.

Hence, the newly proposed RF based classification and features selection framework seems to have the capability to identify loads of a vehicle which are harmful or at least related to a failure of the hybrid car battery.

Analysis of dataset 2a)

In order to back up the results obtained for dataset 1a), the same analysis is repeated for dataset 2a). Before discussing the results of this second analysis, it is emphasized that the main difference between the two datasets is that all the faulty vehicles of dataset 2a) are known to suffer from the same failure of the hybrid car battery. For dataset 1a this is not the case. As a consequence, it is expected that even better results can be achieved using the load spectrum data which has been recorded for the second HEV fleet.

Table 3.10: Optimal determined parameters and corresponding CV$_{out}$-BAC values achieved on the *outer* stratified 5-fold CV for each of the studied algorithms, **before** applying additionally feature selection

	Parameter										
Method	m_{try}	*cut-off*$_{+1}$	*samp-size*$_{-1}$	C	σ	w_{+1}	c_p	*CF*	*tri-als*	*min-Cases/ Split*	**CV$_{out}$-BAC**
rf	500	0.50	3	–	–	–	–	–	–	–	**0.969**
orf$_{linEnet}$	50	0.25	3	–	–	–	–	–	–	–	0.950
orf$_{linRidge}$	14	0.25	3	–	–	–	–	–	–	–	0.958
orf$_{linLasso}$	450	0.30	1	–	–	–	–	–	–	–	0.950
orf$_{log}$	28	0.45	3	–	–	–	–	–	–	–	0.953
orf$_{logEnet}$	50	0.30	2	–	–	–	–	–	–	–	0.951
orf$_{logRidge}$	14	0.40	3	–	–	–	–	–	–	–	0.950
orf$_{logLasso}$	200	0.35	1	–	–	–	–	–	–	–	0.949
orf$_{pls}$	28	0.30	2	–	–	–	–	–	–	–	0.952
orf$_{svm}$	50	0.45	3	–	–	–	–	–	–	–	0.951
SVM$_{linear}$	–	–	–	$10^{-.75}$	–	1.0	–	–	–	–	0.950
SVM$_{rbf}$	–	–	–	4	2^{-5}	1.0	–	–	–	–	0.966
CART$_{gini}$	–	–	–	–	–	1.2	0.5	–	–	30	0.926
CART$_{info}$	–	–	–	–	–	0.6	0.5	–	–	30	0.927
C5.0	–	–	–	–	–	1.2	–	0.01	1	50	0.935

The best result is highlighted in bold. The numbers ℓ in the column *sampsize*$_{-1}$ indicate that *sampsize*$_{-1}$ is set to ℓ·*sampsize*$_{+1}$. For the RF and ORF variants, n_{tree} is fixed to 300 and *cutoff*$_{-1}$ is set to $1 -$ *cutoff*$_{+1}$.

Table 3.10 summarizes the results for each algorithm before any external feature selection strategy is applied. As expected, the overall estimated classification performance is significantly better than the one that has been achieved on dataset 1a). All studied classifiers produce a CV$_{out}$-BAC value greater than 0.92. The standard RF model *rf* again performs best. Its CV$_{out}$-BAC value is 0.969, i.e., it classifies the dataset almost perfectly. This time, however, it is closely followed by *SVM-RFE*$_{rbf}$ with an estimated performance of 0.966. Again, the classification trees perform worst, but at a pretty high level. Regarding the determined optimal parameters, no noteworthy differences are spotted in comparison with the analysis of dataset 1a).

Table 3.11 shows the results for each algorithm after the discussed feature selection strategies are applied. Exactly one half of the studied RF and ORF variants can improve the classification performance using feature selection, while

Table 3.11: Optimal determined parameters and 5-fold cross-validated BAC values (CV$_{out}$-BAC) for each of the 17 studied classifiers, **after** applying the distinct variable selection strategies

	Parameter				
Method	m_{try}	C	σ	**# variables used**	**CV$_{out}$-BAC**
rf_{Gini}	12	–	–	50	***0.970**
rf_{BPI}	4	–	–	57	0.961
rf_{PI}	2	–	–	**15**	0.961
$orf_{linEnet}$	18	–	–	137	*0.955
$orf_{linRidge}$	2	–	–	101	*0.964
$orf_{linLasso}$	123	–	–	188	0.948
orf_{log}	20	–	–	72	*0.968
$orf_{logEnet}$	10	–	–	37	0.947
$orf_{logRidge}$	69	–	–	176	0.905
$orf_{logLasso}$	20	–	–	31	*0.964
orf_{pls}	6	–	–	21	0.932
orf_{svm}	9	–	–	54	*0.959
$SVM\text{-}RFE_{linear}$	–	$10^{-0.5}$	–	187	0.941
$SVM\text{-}RFE_{rbf}$	–	10	2^{-5}	33	0.946
$CART_{gini}$	–	–	–	1	0.926
$CART_{info}$	–	–	–	1	0.927
C5.0	–	–	–	3	0.935

The best result is highlighted in bold. The variants where feature selection improves the classification performance are marked with *.

the other half achieves slightly worse results after performing this data reduction step. Moreover, RFE has a negative effect on the classification performance both for SVM_{linear} and SVM_{rbf}.

As before, rf_{Gini} outperforms the other algorithms with a CV$_{out}$-BAC value of 0.970, while using only 50 out of the 823 non-zero variables variables. However, it is remarkable that rf_{PI} achieves a comparable performance while exploiting only 15 attributes. Furthermore, it is interesting to see that the classification trees also perform noticeably well, while requiring only 1 respectively 3 variables. Since these kind of algorithms are interpretable in contrast to the SVM, RF, and ORF based approaches, they seem to be a true alternative for some load spectrum datasets. However, their applicability depends on the goal of the analysis, i.e., whether the model has to be interpretable or if the best possible classification performance is desired.

Table 3.12: 8 out of the 50 variables that are selected by the final model of variant rf_{Gini} and the stress factors to which they can be linked to

Variable	Stress Description
X_8	total idle time of hybrid car battery
X_{72}, X_{73}	percentage of total number of rainflow cycles that run through a very low SoC level of the hybrid car battery
X_{131}, X_{132}	percentage of rainflow cycles that start at a moderate SoC level of the hybrid car battery and end with low one
X_{509}	percentage of time with the electric machine operating at regular, medium temperature
X_{835}	percentage of very short parking times
X_{840}	percentage of long parking times

Finally, Table 3.12 presents 8 out of the 50 variables that have been selected by rf_{Gini} for this dataset and the stress factors to which these features can be linked to. Hence, the duration of the idle and parking times seem to be part of the stress patterns that allows the classifier to distinguish between the failed and “healthy” vehicles. Also the nature of charging and discharging cycles seem to have some influence on the result, which is not really surprising, since this stress factor is among the known ones for Li-ion batteries [51]. For data confidentiality reasons, the known causes for the failure of the hybrid car battery of the studied vehicles can not be mentioned in this work. However, it is anticipated that the listed variables will be part of some failure relevant patterns in the data that will be learned from this dataset in Chapter 5. Hence, rf_{Gini} is capable to select variables that may be related to particular failures of hybrid components.

3.6 Conclusion

In this chapter, a huge case study has been performed to find out whether it is possible to distinguish between vehicles having a faulty component and healthy ones using only load spectrum data. Therefore, 17 variants of different classification algorithms have been applied to load spectrum data that has been recorded for two distinct HEV fleet. Thereby, each HEV fleet contained some vehicles that suffered from a failure of the hybrid car battery. Among the studied methods are various ORF and RF models, SVM approaches, and distinct types of classification trees.

Furthermore, a new framework has been proposed that facilitates not only to automatically optimize the parameters of arbitrary RF and ORF variants, but also to select the most important variables for the classification task. It has been shown that the classification performance can be improved frequently using this new framework. Moreover, it has been demonstrated empirically that the standard, univariate RF employing a Gini Index based variable importance measure outperforms the other studied classifiers on this kind of automotive data with respect to the classification performance.

Moreover, the obtained results have shown that the features which are selected by the proposed framework can be related to failures of the hybrid car battery from an engineering point of view. Thus, the new approach enables engineers to focus on the most relevant load spectrum classes by filtering out a huge amount of attributes, which are irrelevant for these failures.

It has been also demonstrated that if all the faulty vehicles suffer from the same type of failure, then the classification problem can almost be solved completely by the new approach. However, since often the information about the specifics of the failure is not available or even known, a new purely data based method has to be developed that indicates which faulty vehicles may suffer from the same and which ones from another failure type. In other words, an algorithm is required that is at least able to identify which vehicles are stressed in the same manner. Thus, such a method could be used to reason that these equally stressed cars suffer from the same type of failure if the same component fails. For this purpose, the next chapter proposes a visualization approach that has the potential to solve this task.

Finally, the proposed classification and feature selection framework does not produce interpretable models, i.e., although the obtained classifiers may be able to separate the faulty from the healthy HEV, it is still not known, what are the patterns that enable these methods to do this. Thus, later in this work, approaches will be discussed which extract rules from load spectrum data which help to describe these patterns.

4 Visualizing different kinds of vehicle stress and usage

At the end of the previous chapter a motivation is given, why an algorithm is strongly needed that is able to identify whether two faulty vehicles are likely to suffer from the same type of failure or not. Thereby, it is only known that the same component of these two cars has failed. Moreover, the method should draw its conclusion exclusively on the basis of the load spectrum data of these vehicles. In other words, it may not rely on additional information from the workshops, because the latter is frequently not available.

Such a technique may not only help to improve the results of classification tasks, as conducted in Chapter 3, but also has the potential to save a lot of money, as follows. On the one hand, it is a common practise of the OEMs to urge the workshops to return faulty components to them such that they are able to investigate them thoroughly in the laboratory. Thus, the reasons for a failure may be understood better. As a consequence, this knowledge can help to improve future versions of the considered component. On the other hand, the budget and man-power, which is available for these examinations, is usually limited, i.e., not all faulty components of interest can be investigated thoroughly. However, a purely data based algorithm with the desired capabilities could solve this dilemma, because it would enable the OEMs to selectively buy only those components back, which suffered from distinct types of failures. Thus, unnecessary costs may be avoided, while gaining the maximum possible knowledge about the misbehaviour of the components at the same time.

Hence, in this chapter a new approach for visualizing load spectrum data is proposed which incorporates a RF based dissimilarity measure in the prizewinning dimensionality reduction technique t-Distributed Stochastic Neighbour Embedding. Its effectiveness and its superiority compared to other state-of-the-art dimensionality reduction techniques is demonstrated by conducting two diverse case studies. Therefore, the fundamentals of the studied approaches are provided in the beginning of this chapter. Then, their applicability is tested using real-world datasets.

This chapter is an extension of the two papers [7] and [8], which have been published earlier by the author of this Thesis.

P. Bergmeir, *Enhanced Machine Learning and Data Mining Methods for Analysing*
Large Hybrid Electric Vehicle Fleets based on Load Spectrum Data, Wissenschaftliche
Reihe Fahrzeugtechnik Universität Stuttgart, https://doi.org/10.1007/978-3-658-20367-2_4

4.1 Distance and dissimilarity measures

The goal of this chapter is to develop a method that is able to find out whether two arbitrary vehicles are stressed in a similar manner. Hence, a measure is required which quantifies the alikeness of the load spectra that have been recorded for these cars. In Data Mining and Unsupervised Learning, there are several *dissimilarity* or *distance* measures for this purpose [50]. Commonly, these pairwise quantities are stored in a $N \times N$ matrix $\mathbf{D}$, where each element $d_{ih} = d(x_i, x_h)$ records the dissimilarity between the i^{th} and h^{th} instance. Moreover, many dissimilarity matrices are symmetric, i.e., $d_{ih} = d_{hi}$, and have zero diagonal elements, i.e., $d_{ii} = 0$ for all samples i. In general, the smaller the value of d_{ih} is, the more alike the objects i and h are.

In this work, the focus lies on the probably most popular dissimilarity measure, i.e., the *Euclidean distance* [50], and a more recent one that takes advantage of a RF classifier.

4.1.1 Euclidean distance

The *Euclidean distance* [10] is probably the most natural and intuitive way to measure the proximity between two objects in space, because it is equal to the length of the direct connection of these two data points. Formally, it is given by

$$d_E(x_i, x_h) = \|x_i - x_h\|_2 = \sqrt{\sum_{j=1}^{p} \left(x_{ij} - x_{hj}\right)^2}. \qquad \text{Eq. 4.1}$$

As this distance measure mainly accumulates the squared differences between the observed attribute values of the considered objects i and h, it is dominated by features lying on large scales. However, usually each variable should have a priori the same influence on the pairwise dissimilarity. Thus, the attributes have to be normalized or rescaled, such that all feature values are distributed equally or fall into the same range.

Table 4.1: Example for generating a synthetic dataset $\widetilde{\mathbf{X}}$ from the original one, $\mathbf{X}$, by random sampling from the univariate distributions of $\mathbf{X}$

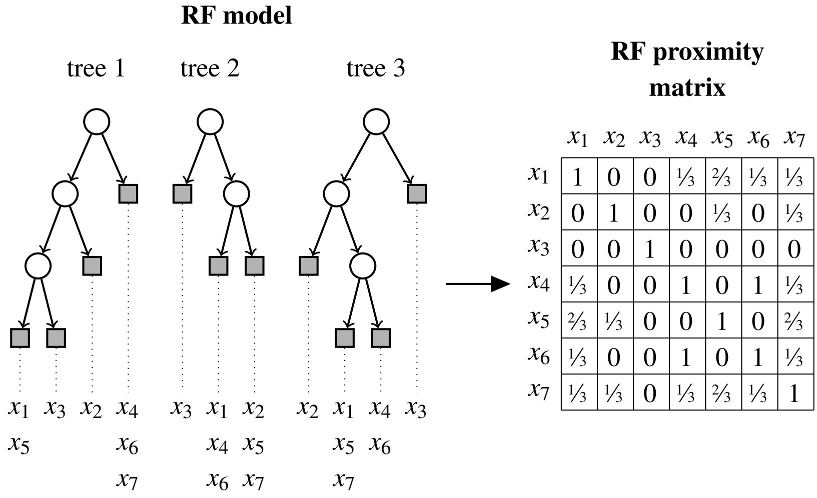
$\mathbf{X}$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$		$\widetilde{\mathbf{X}}$	$\widetilde{\mathbf{x}}_1$	$\widetilde{\mathbf{x}}_2$	$\widetilde{\mathbf{x}}_3$	$\widetilde{\mathbf{x}}_4$
x_1	3.0	15	0.7	111		$\widetilde{x}_1$	3.0	12	1.0	113
x_2	4.5	10	1.0	100		$\widetilde{x}_2$	3.0	15	0.2	111
x_3	9.0	20	0.2	150		$\widetilde{x}_3$	8.0	22	0.5	150
x_4	3.0	13	0.2	143	$\Longrightarrow$	$\widetilde{x}_4$	4.5	13	0.5	188
x_5	2.7	18	0.5	188		$\widetilde{x}_5$	8.0	13	0.5	157
x_6	8.0	12	0.4	157		$\widetilde{x}_6$	2.7	10	0.7	143
x_7	3.0	22	0.5	113		$\widetilde{x}_7$	3.0	12	0.2	100

4.1.2 Random forest dissimilarity

In Chapter 3.2.3, the RF classification algorithm is explained in detail. Another benefit of this method is that it can also be used to calculate proximities between pairs of samples lying in a high-dimensional space. Hence, a scale-independent dissimilarity measure can be inferred by this method, which can serve as input for an unsupervised learning task, e.g., for computing a low-dimensional embedding of a high-dimensional dataset.

The basic idea is to generate a "synthetic" dataset on the basis of the original one first, and to use the RF classifier to distinguish the samples from these two datasets, afterwards. This is proposed by Breiman and Cutler in [18]. Until today, RF dissimilarity measures have been successfully employed in practice, e.g., for tumor profiling based on tissue microarray data [108] or for analysing genomic sequence data [2].

Thereby, the required synthetic data $\widetilde{\mathbf{X}} = (\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_N)^T$ is generated by random sampling from the univariate distributions of the original one, which is denoted by $\mathbf{X}$. In other words, the value of the r^{th} variable of a synthetic sample $\widetilde{x}_i$ is drawn at random from the set $\{x_{1,r}, x_{2,r}, \ldots, x_{N,r}\}$. This leads to the destruction of potential dependencies between the variables $X_1, X_2, \ldots, X_N$. Thus, the attributes of $\widetilde{\mathbf{X}}$ have the same univariate distributions as the corresponding ones in $\mathbf{X}$, but $\widetilde{\mathbf{X}}$ has the distribution of independent random variables at the same time [19].



	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	0	0	⅓	⅔	⅓	⅓
x_2	0	1	0	0	⅓	0	⅓
x_3	0	0	1	0	0	0	0
x_4	⅓	0	0	1	0	1	⅓
x_5	⅔	⅓	0	0	1	0	⅔
x_6	⅓	0	0	1	0	1	⅓
x_7	⅓	⅓	0	⅓	⅔	⅓	1

Figure 4.1: Determining the pairwise RF proximities between seven samples that have been classified by a random forest model, which is formed by three trees

Table 4.1 shows an example dataset $\mathbf{X}$ that consists of seven objects, namely $x_1, x_2, \ldots, x_7$. Each of these samples is characterised by the four variables X_1, X_2, X_3, and X_4. Furthermore, Table 4.1 presents the possible outcome for a synthetic dataset $\widetilde{\mathbf{X}}$ that is generated from $\mathbf{X}$ by applying the sampling strategy described above.

After creating the synthetic dataset $\widetilde{\mathbf{X}}$ from $\mathbf{X}$, each of its samples $\widetilde{x_i}$ is labelled as class 2, while all the original objects are labelled as class 1. Then, the RF classifier is applied to this artificially created binary classification problem to distinguish between these two classes. The resulting RF model is then used to compute proximity values between every possible pair of the original samples, as follows: The two considered records are run down each single tree of the forest and whenever both samples end up in the same leaf node of a tree their pairwise proximity value is incremented by one. Finally, these counters are normalized by dividing them by the total number of trees in the forest.

Figure 4.1 illustrates with a small example the process of computing the RF proximity matrix from a RF model that has been trained on a dataset containing the seven samples $x_1, x_2, \ldots, x_7$. Each leaf node is connected with the objects that fall into it after running down the corresponding tree. In tree 1, for example, the instances x_1 and x_5 end up in the same leaf node, i.e., the leftmost one. Since these two samples also fall into the same leaf node of tree 3, but into different ones in tree 2, their normalized pairwise RF proximity value is ⅔. Moreover, there is no sample that ends up in the same leaf nodes with x_3. Therefore, the pairwise RF proximities between x_3 and all other six samples are 0. In contrast, the objects x_4 and x_6 always reach the same leaf nodes. That is why their pairwise RF proximity is 1.

In order to improve the stability of the RF proximities, the forest should contain a large number of trees, e.g., a few thousand, and the whole procedure, including the generation of the synthetic data, should be repeated several times. Thereby, the risk of creating a somehow biased synthetic dataset, which leads to undesired effects, is minimized. Then, the final RF proximity between two arbitrary objects x_i and x_h is obtained by aggregating the corresponding proximity values resulting from n_{forest} RF models, $n_{forest} \geq 1$, as follows:

$$prox_{RF}\left(x_i, x_h\right) = \frac{1}{n_{forest}} \sum_{f=1}^{n_{forest}} prox_f\left(x_i, x_h\right), \qquad \text{Eq. 4.2}$$

where $prox_f\left(x_i, x_h\right)$ denotes the proximity between x_i and x_h determined with the f^{th} RF model.

However, many dimensionality reduction techniques require a dissimilarity measure as input, as will be described in the upcoming sections. Therefore, by following [107], a dissimilarity measure can be obtained from the RF proximity values, as follows:

$$dist_{RF}\left(x_i, x_h\right) = \sqrt{1 - prox_{RF}\left(x_i, x_h\right)}. \qquad \text{Eq. 4.3}$$

4.2 Dimensionality reduction methods

Since it is nowadays possible to record and store huge amounts of complex data such as speech signals, automotive data, or genome data, handling such high-dimensional data becomes a challenging task. However, often the relevant information for a task is contained in a comparatively small subset of the available features. In that case, the number of variables can be reduced without producing a significant loss of information, i.e., the characteristics of the data can be preserved by keeping only a small number of variables. This has been already shown through the case study of Chapter 3.

Therefore, in the last few decades, many *dimensionality reduction techniques* have been developed to facilitate tasks, like the exploration, the visualization, or the denoising of high-dimensional data. Feature selection, as explained in subchapter 3.3, is one way to reduce the dimensionality of dataset. However, if a visual presentation of a high-dimensional dataset is desired, then the dataset has to be mapped to a space of a predefined number of dimensions u. Common choices are $u = 2$ or $u = 3$. This kind of dimensionality reduction is also known as *data compression* [49]. Following the terminology of [75], the term *dimensionality reduction techniques* refers to approaches that convert high- to low-dimensional data in this work.

The main goal of dimensionality reduction is to find a low-dimensional data representation $\mathscr{M} = (m_1, m_2, \ldots, m_N)^T$ which preserves as much significant structure of the original, high-dimensional data $\mathbf{X} = (x_1, x_2, \ldots, x_N)^T$ as possible. The dataset $\mathscr{M} \in \mathbb{R}^{N \times u}$ is also called a *map* and its elements $m_i \in \mathbb{R}^u$ are consequently named *map points* [75]. For simplification, in the following it is not distinguished between the data and the space where its samples are lying in, i.e., the high-dimensional space is also denoted by the symbol $\mathbf{X}$.

Figure 4.2 shows a taxonomy of the dimensionality reduction techniques that are studied in this work. Generally, these algorithms can be grouped into *linear* and *non-linear* methods. Thereby, the former ones rely on the assumption that the data lies on a linear or at least close to a linear subspace of the high-dimensional input space, while the latter ones are not so restrictive [76]. Thus, non-linear techniques are able to identify more complex embeddings of the high-dimensional space.

In the following, the techniques which are investigated in this work are briefly explained and discussed. These are namely the traditional linear dimensionality reduction method *Principal Components Analysis (PCA)* and its non-linear competitors *Sammon mapping*, *Locally Linear Embedding (LLE)*, *Isometric Feature Mapping (Isomap)*, and the recently proposed method *t-Distributed Stochastic Neighbour Embedding (t-SNE)*.

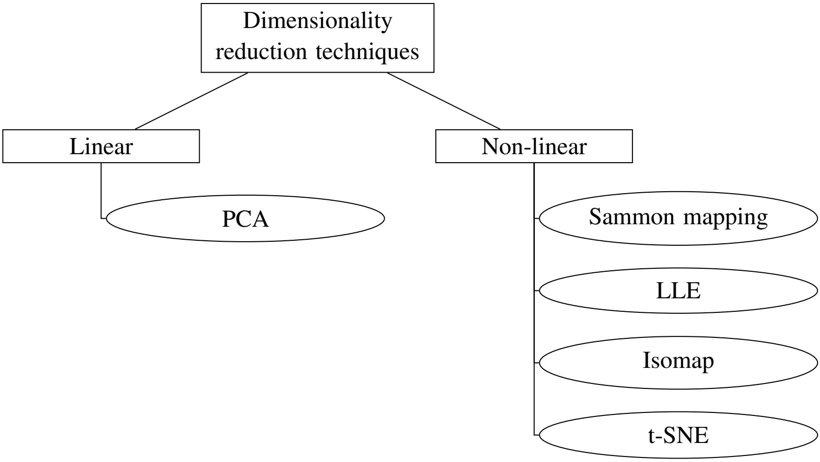


Figure 4.2: Categorization of the studied dimensionality reduction techniques

4.2.1 Principal Components Analysis

Since its development in the beginning of the last century, the statistical procedure *Principal Components Analysis (PCA)* [54, 92] has become probably the most popular linear dimensionality reduction technique. The main goal of this algorithm is to project the high-dimensional data to a low-dimensional subspace in such a way that as much of the variance in the original data is explained as possible. Therefore, it minimizes the correlation between the features by mapping the data into a vector space that is spanned by a linear basis, in which the variance is maximized [76].

Mathematically, PCA aims at determining an orthogonal $p \times u$ matrix $\mathbf{A}$, $u << p$, that maximizes the term $\mathbf{A}^T cov(\tilde{\mathbf{X}}, \tilde{\mathbf{X}})\mathbf{A}$, where $cov(\tilde{\mathbf{X}}, \tilde{\mathbf{X}})$ denotes the covariance matrix of the zero mean input data $\tilde{\mathbf{X}}$, i.e., $\tilde{\mathbf{X}}$ is derived from $\mathbf{X}$ by scaling each of its attribute to zero mean. Thus, PCA solves the following eigenproblem [73]:

$$cov(\tilde{\mathbf{X}}, \tilde{\mathbf{X}})\mathbf{A} = \lambda \mathbf{A}. \qquad \text{Eq. 4.4}$$

If a u-dimensional data representation is desired, this problem has to be solved for the u largest eigenvalues, i.e., the columns of $\mathbf{A}$ are given by the corresponding u eigenvectors. Finally, the low-dimensional map points $m_i \in \mathbb{R}^u$ can be obtained by mapping the corresponding data points x_i onto the linear basis given by $\mathbf{A}$ [10]:

$$m_i = (x_i - \bar{x})\,\mathbf{A}, \qquad \text{Eq. 4.5}$$

where $\bar{x}$ denotes the vector of empirical mean values, i.e., $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$.

4.2.2 Sammon Mapping

The dimensionality reduction technique *Sammon mapping* [103] belongs to a collection of non-linear methods called *Multidimensional Scaling (MDS)* [30]. MDS algorithms aim at finding a low-dimensional data representation $\mathscr{M}$, while retaining the pairwise distances between the given data objects as much as possible [76].

Thereby, several different stress functions can be used to measure the discrepancy between the pairwise distances between the objects in the high-dimensional space and those between their low-dimensional data representations.

Referring to [10], the original stress function is given by

$$\Phi(\mathscr{M}) = \sum_{i=1}^{N} \sum_{h=i+1}^{N} \left(d_E^{(X)}(x_i, x_h) - d_E^{(\mathscr{M})}(m_i, m_h) \right)^2, \qquad \text{Eq. 4.6}$$

where the functions $d_E^{(X)}$ and $d_E^{(\mathscr{M})}$ designate the Euclidean distance in the high- and low-dimensional space, respectively. It has to be minimized to obtain good data representations in $\mathscr{M}$.

In other words, it measures the deviation between the original distances and those in the low-dimensional space using the sum of squared differences. A major disadvantage of this measure is that its value is dominated by large pairwise distances between objects of $\mathbf{X}$, i.e., minor ones only have a marginal influence.

Therefore, in order to try to overcome this disadvantage, i.e., to put more emphasis on preserving also smaller pairwise distances, the dimensionality reduction technique Sammon mapping uses a modification of the stress function that is given by Eq. 4.6:

$$\Phi(\mathscr{M}) = \frac{1}{\sum_{i=1}^{N} \sum_{h=i+1}^{N} d^{(X)}\left(x_i, x_h\right)} \cdot \sum_{i=1}^{N} \sum_{h=i+1}^{N} \frac{\left(d^{(X)}\left(x_i, x_h\right) - d^{(\mathscr{M})}\left(m_i, m_h\right)\right)^2}{d^{(X)}\left(x_i, x_h\right)}, \qquad \text{Eq. 4.7}$$

where $d^{(X)}$ and $d^{(\mathscr{M})}$ denote the chosen distance measures of the original and the projected space, respectively [10]. The distance has not necessarily to be the Euclidean distance.

These stress functions can be minimized using either gradient methods, or eigendecomposition of the pairwise dissimilarity matrices, or a pseudo-Newton method [30].

4.2.3 Locally Linear Embedding

The *non-linear* dimensionality reduction technique *Locally Linear Embedding (LLE)* is introduced in [101], where it is used successfully to reasonably arrange images of similar human faces in 2D as well as words in a continuous semantic space.

The basic assumption of LLE is that each data point and its neighbours lie on or at least close to a locally linear patch of the manifold [101]. Thereby, this local structure is captured by linear coefficients w_{ih} that allow to represent each data point x_i as a linear combination of its k nearest neighbours $x_h \in \mathscr{N}_k(x_i)$, where $\mathscr{N}_k(x_i)$ designates the set of the k nearest neighbours of x_i. The overall error that is made by the reconstructions of samples x_i is measured by the following cost function:

$$err(w) = \sum_i \left| x_i - \sum_h w_{ih} x_h \right|^2, \qquad \text{Eq. 4.8}$$

where the weights w_{ih} reflect the contribution of neighbour x_h to the reconstruction of instance x_i.

Optimal values for the weights w_{ih} are obtained by solving the following optimization problem:

$$
\begin{aligned}
&\min_{w} err(w) && \text{Op. 4.1}\\
&\text{subject to:} &&\\
&\sum_{h} w_{ih} = 1 &&\\
&w_{ih} = 0 \qquad \forall x_h \notin \mathscr{N}_k(x_i)\,. &&
\end{aligned}
$$

The corresponding low-dimensional data representations $m_1, m_2, \ldots, m_N$ of the high-dimensional objects $x_1, x_2, \ldots, x_N$ are finally determined by minimizing the following embedding cost function, while fixing the weights w_{ih}:

$$\Phi(\mathscr{M}) = \sum_{i} \left| m_i - \sum_{h} w_{ih} m_h \right|^2 . \qquad \text{Eq. 4.9}$$

This cost function can be minimized by solving a sparse $N \times N$ eigenvalue problem, where the bottom u non-zero eigenvectors provide an ordered set of orthogonal coordinates centred on the origin [101]. However, in order to make sure that the problem is well posed, the low-dimensional data representations must have unit covariance. In other words, the embedding cost function has to be optimized subject to some constraints, which are out of scope of this work.

Is has to be noted that the only parameters which have to be set by the user are the number k of nearest neighbours that define the neighbour of each data point as well as the distance function that measures the closeness between the data objects.

However, as pointed out in [75], a disadvantage of LLE is that the only condition that tries to avoid that all objects collapse onto a single point in the low-dimensional embedding $\mathscr{M}$ is the constraint on the covariance of $\mathscr{M}$, as mentioned before. Though, this constraint is often fulfilled by simply mapping the majority of the data points close to the centre of the map, while scattering the few other points in the empty regions of the low-dimensional space. Thereby, usually little insights into the data structure are provided.

Moreover, since LLE is a neighbourhood-graph based method, the data which has to be visualized must necessarily induce a connected neighbourhood graph.

However, this is not the case for data lying on at least two widely separated submanifolds. Therefore, this kind of data can not be explored properly by using LLE.

4.2.4 Isomap

In [112], another neighbour-graph based method is proposed at approximately the same time as LLE: *Isometric Feature Mapping (Isomap).* It is an enhancement of classical metric-MDS and its main idea is to preserve the intrinsic geometry of the data by estimating the geodesic manifold distances between all pairs of data points. For data points that lie in the same neighbourhood, a sufficient approximation of the geodesic pairwise distances is to simply use their distances in the input space $\mathbf{X}$. However, estimating the geodesic distance between points lying far apart from each other is more complicated. Isomap approximates these distances by computing shortest paths in a graph, where neighbouring points are connected by edges with weights equal to their distance measured in $\mathbf{X}$.

The main steps of Isomap are briefly explained in the following: First, the pairwise distances $d^{(X)}(x_i,x_h)$ are computed between all pairs of objects x_i and x_h in $\mathbf{X}$. Thereby, the user can specify the choice of this distance measure. Afterwards, a neighbourhood-graph G is created by connecting each point x_i with either its k nearest neighbours or all points which are lying within a hypersphere of fixed radius ε around x_i. Next, the edges between two neighbours x_i and x_h are weighted in G with the distance $d^{(X)}(x_i,x_h)$.

Secondly, the geodesic distances of the manifold are approximated by computing the shortest path distance $d^{(G)}(x_i,x_h)$ in G. For this task, shortest path algorithms like Dijkstra's [33] or Floyd's algorithm [38] can be used.

Finally, MDS is applied to the matrix of pairwise, estimated geodesic distances to obtain a low-dimensional embedding $\mathscr{M}$.

4.2.5 t-Distributed Stochastic Neighbour Embedding

Since van der Maaten proposed *t-Distributed Stochastic Neighbour Embedding (t-SNE)* in [75] in 2008 for the first time, this dimensionality reduction method has been successfully used to visually explore many real-world datasets of different kinds, e.g., mouse brain data [57] and metagenomics data [67]. It can be seen as an enhancement of the formerly developed *Stochastic Neighbour Embedding (SNE)* [52]. It is because it employs a symmetrized version of the cost function of the latter one, which has simpler gradients that lead to easier solutions of the underlying optimization problem in turn [75]. Moreover, it tries to avoid a phenomenon which is called the *crowding problem* [75]. This problem is also described briefly later in this work.

The basic idea behind t-SNE is to measure the similarities between two arbitrary high-dimensional objects $x_i, x_h \in \mathbf{X}$ by a probability distribution $\mathscr{P}$ that is defined over pairs of data points. Thereby, the probability of choosing a particular pair of points is proportional to the similarity of these samples. In other words, the distribution $\mathscr{P}$ assigns high probabilities to pairs of similar objects, i.e., to data points that are lying nearby in $\mathbf{X}$, while dissimilar points, i.e., objects that are far apart from each other in $\mathbf{X}$, have a low probability under $\mathscr{P}$. Formally, $\mathscr{P}$ is given by

$$p_{ih} = \frac{p_{i|h} + p_{h|i}}{2N}, \qquad \text{Eq. 4.10}$$

where the conditional probabilities $p_{i|h}$ and $p_{h|i}$ are defined, as follows:

$$p_{i|h} = \frac{\exp\left(-d^{(X)}\left(x_h, x_i\right)^2 / 2\sigma_h^2\right)}{\sum_{k \neq h} \exp\left(-d^{(X)}\left(x_h, x_k\right)^2 / 2\sigma_h^2\right)}, \qquad \text{Eq. 4.11}$$

$$p_{h|i} = \frac{\exp\left(-d^{(X)}\left(x_i, x_h\right)^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-d^{(X)}\left(x_i, x_k\right)^2 / 2\sigma_i^2\right)}, \qquad \text{Eq. 4.12}$$

where $p_{i|i} = p_{h|h} = 0$ and σ_i, σ_h denote the bandwidths of the Gaussian kernels, while $d^{(X)}\left(x_i, x_h\right)$ denominates the pairwise dissimilarity of x_i and x_h in $\mathbf{X}$. In the basic variant of t-SNE, the Euclidean distance d_E is chosen for the function $d^{(X)}$. In the latter case, $p_{i|h}$ can be interpreted as the probability that x_h selects x_i as its neighbour, when neighbours are picked in proportion to their probability density under a Gaussian distribution, which is centred at x_h.

Moreover, for each data object $x_h \in X$ a different bandwidth σ_h is set in such a way that the conditional distribution has a fixed perplexity, $Perp\left(P_h\right)$, which has to be defined by the user. This can be seen as scaling the bandwidth of the

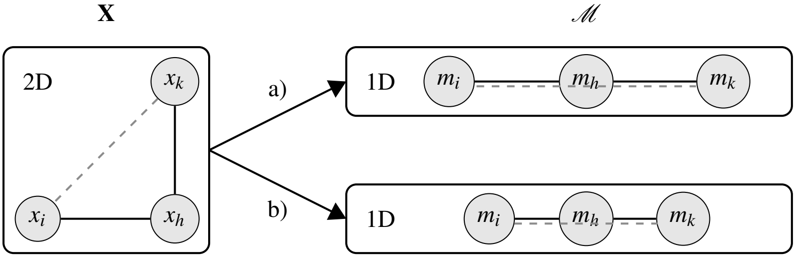


Figure 4.3: Illustrative example showing the impossibility of preserving both the a) local and the b) global structure of high-dimensional data **X** when embedding it in a lower-dimensional space $\mathscr{M}$

Gaussian in such a manner that a user-defined number of data points fall in the mode of this Gaussian. Since it is very likely that the high-dimensional space **X** contains both sparse and dense regions, using individual bandwidths is a possibility to adapt to these different densities. Usually, selecting a smaller value of σ_h is more appropriate in dense than in sparse regions. t-SNE performs a binary search to determine the values of σ_h, where the perplexity is given by

$$Perp(P_h) = 2^{-\sum_i p_{i|h} \log_2 p_{i|h}}, \qquad \text{Eq. 4.13}$$

where P_h denotes the probability distribution that is induced by σ_h over all other data objects x_k with $k \neq h$. The developers of t-SNE recommend to set *Perp* to a value that lies in the range from 5 to 50. Additionally, they note that the outcome of t-SNE usually is very robust against the choices of this parameter [75].

In order to preserve in the low-dimensional data representation $\mathscr{M}$ as much of the local structure of the high-dimensional data **X** as possible, it is obligatory to model distant points of **X** even farther apart in $\mathscr{M}$. Otherwise, it may happen that these points collapse into a single point in $\mathscr{M}$. This is what the term *crowding problem* refers to. Figure 4.3 illustrates with a simple example why it is impossible to find a mapping from **X** to $\mathscr{M}$ that accurately preserves both the local and the global structure of **X**.

The black solid distances $d^{(X)}(x_i, x_h)$ and $d^{(X)}(x_h, x_k)$ are of equal length, but shorter than the one between the data points x_i and x_k. The latter distance is visualized by a grey, dashed line. Thus, preserving the local structure of the three objects in the two-dimensional space $\mathbf{X}$ is equivalent to retaining the short black distances. However, modelling the length of these two distances accurately in the one-dimensional space $\mathscr{M}$ leads to an increase in the length of the larger, grey distance, i.e., the corresponding points m_i and m_k are modelled to far apart from each other in 1D (cf. subplot a) in Figure 4.3). If the distance between the latter two points would have been preserved exactly, the three observations would have moved closer together, i.e., they would have occupied a smaller region of the low-dimensional space $\mathscr{M}$ compared to the high-dimensional space $\mathbf{X}$ (cf. subplot b) in Figure 4.3). Thereby, in the worst case, the local structure of the data may be obscured completely, leading to the above mentioned crowding problem.

The algorithm t-SNE tries to circumvent this problem by using a heavy-tailed normalized Student's t-Distribution with a single degree of freedom to measure the similarity q_{ih} between objects m_i and m_h in the low-dimensional embedding $\mathscr{M}$. These pairwise similarities are formally given by

$$q_{ih} = \frac{\left(1 + \|m_i - m_h\|^2\right)^{-1}}{\sum_{k \neq \ell}\left(1 + \|m_k - m_\ell\|^2\right)^{-1}}, \qquad \text{Eq. 4.14}$$

where q_{ii} is zero. The tails of a Student's t-Distribution, which occupy more probability mass compared to a Gaussian distribution, enable t-SNE to model distant objects of the original space $\mathbf{X}$ even farther apart in the low-dimensional space $\mathscr{M}$.

Finally, the low-dimensional coordinates $m_1, m_2, \ldots, m_N$ are obtained by minimizing a cost function that is equal to the *Kullback-Leiber divergence (KL-divergence)* between the two induced joint probability distributions $\mathscr{P}$ and $\mathscr{Q}$:

$$\min_{\mathscr{Q}} KL(\mathscr{P} \,\|\, \mathscr{Q}) = \sum_i \sum_{h \neq i} p_{ih} \log \frac{p_{ih}}{q_{ih}}. \qquad \text{Eq. 4.15}$$

On the one hand, the asymmetry of the KL-divergence leads to high costs if large values of p_{ih} are modelled by small values of q_{ih}, i.e., if nearby data objects of the high-dimensional space $\mathbf{X}$ are mapped to distant points in $\mathscr{M}$. On the other hand, it does not really account for the correct modelling of small values of p_{ih} in $\mathscr{M}$. Hence, t-SNE mainly focuses on modelling similar objects in $\mathbf{X}$ by similar objects in $\mathscr{M}$, i.e., on preserving the local structure of the data, as mentioned earlier. However, since it is able to model distant points in $\mathbf{X}$ even farther apart in $\mathscr{M}$, as explained above, at least some of the global structure of the original data can be captured.

Computational complexity is beyond the focus of this work. However, it has to be mentioned that in [74] an implementation of t-SNE is introduced which employs a variant of the *Barnes-Hut* algorithm that can learn embeddings in $\mathscr{O}(N \log N)$ time, while requiring only $\mathscr{O}(N)$ space at the same time.

t-Distributed Stochastic Neighbour Embedding using a RF dissimilarity

As discussed, t-SNE uses a probability distribution to model the pairwise similarities between arbitrary high-dimensional objects in $\mathbf{X}$. Referring to Eq. 4.10, this probability distribution symmetrizes conditional probabilities which incorporate the Euclidean distances in their default definitions.

However, the Euclidean distance does not perform intrinsic feature selection in contrast to the RF dissimilarity, as introduced in Chapter 4.1.2. Thus, it may be deteriorated by noise variables. Moreover, it requires a scaling of the attributes such that all of them have a priori the same influence on this distance measure, whereas the RF dissimilarity is scale-independent. Furthermore, in Chapter 3 it has been shown that the RF algorithm can be a successful technique if it is applied to load spectrum data.

As a consequence, it is interesting to study whether the default t-SNE implementation may be improved by replacing the Euclidean distance by the RF dissimilarity in Eq. 4.11 and Eq. 4.12. The acronym *RF-t-SNE* is used for this newly proposed variant of t-SNE in the following.

4.3 Case study: Dependence of vehicle usage on operating country

In the first case study of this chapter, it is investigated which of the presented dimensionality reduction techniques is capable to visualize appropriately the different types of vehicle usage and stress that are prevalent in the studied datasets 1b) and 2b). Since it is known that the considered HEV are stressed differently in distinct operating countries because of the heterogeneous traffic and environmental conditions, this analysis serves as a benchmark study for the discussed techniques.

As in the previous chapter, all required implementations are based on `R` packages. While PCA is included in `R` by default, an implementation of Sammon mapping can be found in the package `MASS` [119]. The algorithm LLE is contained in the package `lle` [32] and the package `vegan` [89] offers an implementation of Isomap. Finally, the two variants RF-t-SNE and t-SNE rely on the packages `Rtsne` [63] and `randomForest` [69].

4.3.1 Preprocessing and parametrization

Before any of the discussed dimensionality reduction techniques is applied, the following preprocessing steps are conducted. In Chapter 4.1.1 it is explained why the Euclidean distance requires a scaling of the used variables. Therefore, each observed attribute vector $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$ is scaled to zero mean and unit variance, before applying any of those dimensionality reduction technique that employs this distance measure. Among them, there are all studied methods with exception of RF-t-SNE. For the latter technique the unscaled data is used as input, because the RF dissimilarity is scale-independent, as discussed in Chapter 4.1.2.

Moreover, if $\mathbf{x}_j = (c, c, \ldots, c)^T$ for any $j \in \{1, 2, \ldots, p\}$, with $c \in \mathbb{R}$ being a constant, then $\mathbf{x}_j$ is filtered out, because it does not provide any information that could be useful for detecting differences in the stress or usage of the vehicles.

Table 4.2 shows the parameter settings for each studied dimensionality reduction technique with respect to each of the two analysed datasets. First, PCA does not require any parameters. On the contrary, as the utilized implementation of Sammon mapping employs an iterative Newton method to optimize its objective function, the maximum number of iterations have to be specified. This parameter is set to 1000 for both datasets.

Moreover, since each country in dataset 1b) is represented by at least 25 HEV, the number of nearest neighbours k is fixed to these two values in the algorithms LLE and Isomap. Also the *perplexity* in t-SNE and RF-t-SNE is set to this value, because it is also related to the neighbourhood of an object. However, since the authors of t-SNE recommend to use a value in the range from 5 to 50 for the perplexity, this parameter is fixed to 50 for dataset 2b),

Table 4.2: Parameter setting for each of the studied dimensionality reduction techniques, specified separately for the two analysed datasets

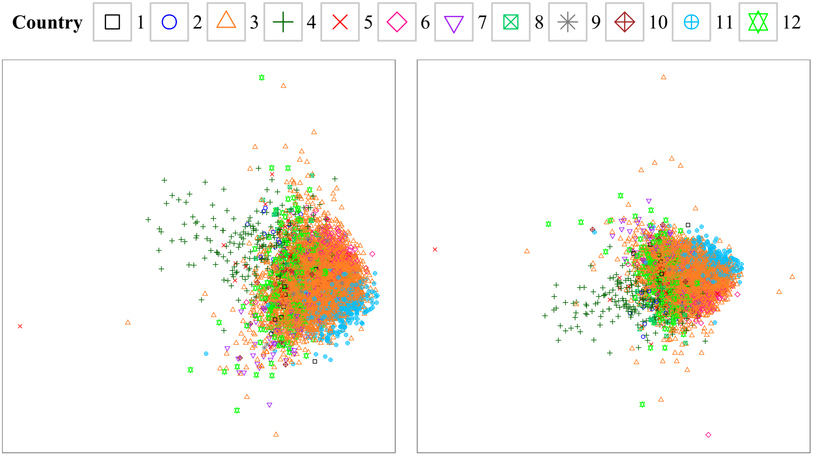
Dim. reduction technique	**Dataset 1b)**	**Dataset 2b)**
PCA	–	–
Sammon mapping	$n_{iter} = 1000$	$n_{iter} = 1000$
LLE	$k = 25$	$k = 50$
Isomap	$k = 25$	$k = 50$
t-SNE	$perplexity = 25$	$perplexity = 50$
	$n_{iter} = 1000$	$n_{iter} = 1000$
RF-t-SNE	$n_{tree} = 5000$	$n_{tree} = 5000$
	$n_{forest} = 50$	$n_{forest} = 50$
	$perplexity = 25$	$perplexity = 50$
	$n_{iter} = 1000$	$n_{iter} = 1000$

although each country is represented by at least 100 HEV in this dataset. In order to be consistent, the same value is used for parameter k in LLE and Isomap, respectively.

Since the two studied t-SNE approaches apply a gradient descent algorithm to minimize the KL divergence between the induced probability distributions, the number of optimization iterations n_{iter} has to be specified. As in Sammon mapping, this parameter is set to 1000 for both datasets. Finally, the number of RF models n_{forest} and the number of trees n_{tree} forming each of these models have to be specified for the computation of the RF dissimilarity that is employed in the newly proposed variant RF-t-SNE. In this work, this measure is based on 50 RF models, where each of them is built by 5000 trees.

4.3.2 Results

In the next two subchapters the results that have been achieved on the studied datasets 1b) and 2b) are presented and discussed in detail.



(a) PCA

(b) Sammon mapping

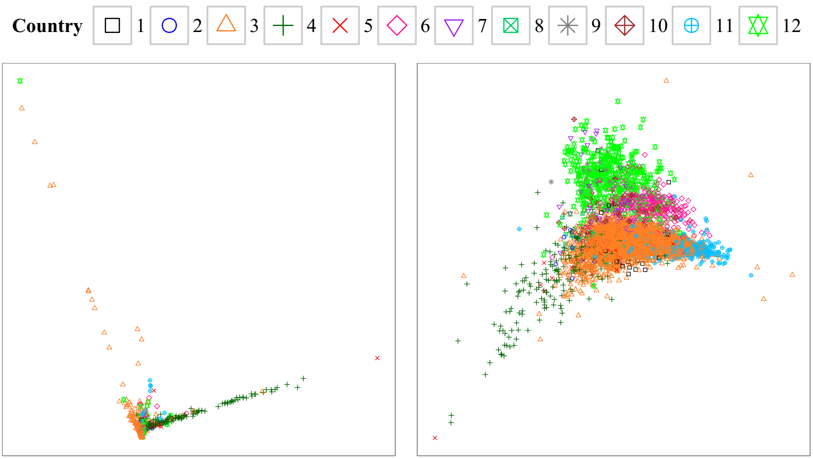
Figure 4.4: Visualizations of the 2D data representations of dataset 1b) obtained by PCA and Sammon mapping (cf. [8])

Analysis of dataset 1b)

Figures 4.4, 4.5, and 4.6 show six distinct two-dimensional maps of the 6670 vehicles of dataset 1b), where each vehicle is represented by the computed projection of all its load spectrum data. The six charts result from applying PCA, Sammon mapping, LLE, Isomap, t-SNE, and RF-t-SNE to this dataset, respectively. In each map, vehicles that are mainly driven in the same country are represented by the same colour and symbol, which are shown in the legend above each figure. However, it has to be emphasized that the information about the main operating country of each vehicle is not used for determining the map points that are depicted in the visualizations.

For data confidentiality reasons, these 12 different countries are not named explicitly, but are encoded by distinct numbers. Moreover, since the values of these map points are only used for the purpose of visualizing the dataset and since they do not have a meaningful (physical) interpretation, the coordinate axes are hidden in each plot. Thereby, misinterpretations shall be avoided.

The data representations obtained by PCA and Sammon mapping are very similar, as Figure 4.4 reveals. Both techniques produce compact point clouds of



(a) LLE

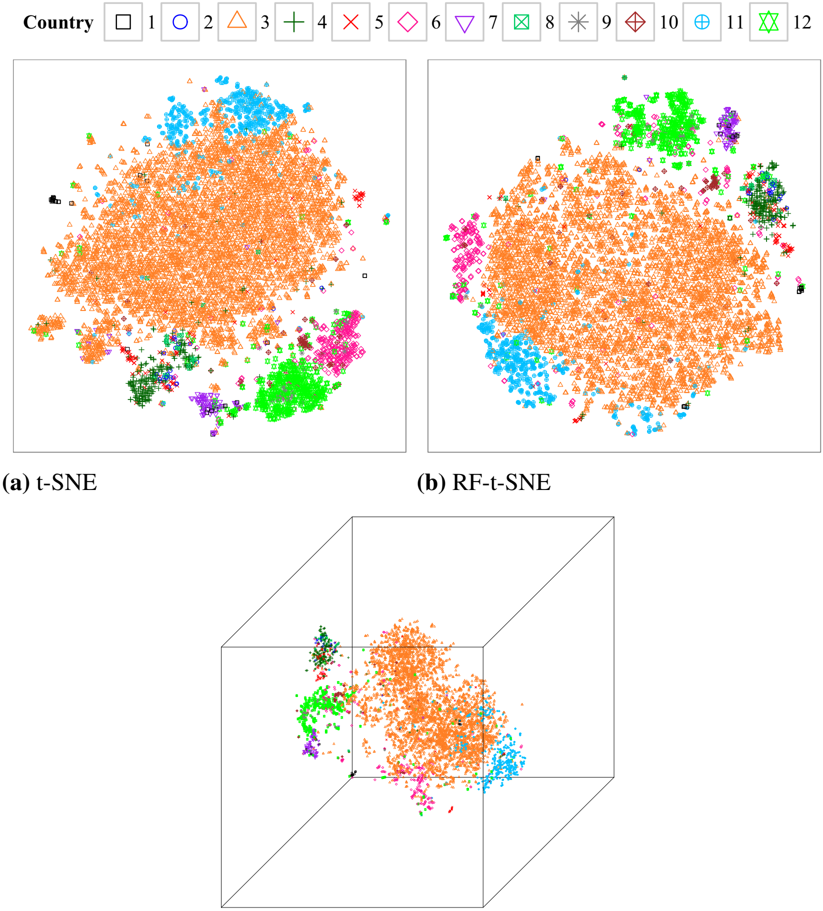
(b) Isomap

Figure 4.5: Visualizations of the 2D data representations of dataset 1b) obtained by LLE and Isomap (cf.[8])

high density. Since the vehicles of distinct countries are located in overlapping, hardly distinguishable regions, no clear clustering structure can be identified with respect to the operating countries of the vehicles. Nevertheless, the colouring gives the impression that there is some country-related structure in this dataset. However, if the map points would not be colourised in these two plots, almost no groupings could be spotted in the visualizations of the data, i.e., one would get the impression that the majority of the vehicles is stressed in a similar way.

Figure 4.5 shows the diverse two-dimensional maps produced by LLE and Isomap, respectively. LLE results in a very uninformative data representation, where almost all vehicles are aligned on two connected lines. Only the usage of the vehicles of countries 3 and 4 seems to be different because the corresponding cars can be found on distinct lines. However, there are also some outliers in the plot, i.e., the load spectrum data underlying these particular vehicles seem to be of a different nature compared to those of the remaining ones, which are projected onto a small, dense region, close to the bottom of the chart.

On the contrary, the map yielding from Isomap facilitates a visual discrimination of the vehicles of the five countries 3, 4, 6, 11, and 12. The cars of these



(a) t-SNE

(b) RF-t-SNE

(c) RF-t-SNE (3D)

Figure 4.6: Visualizations of the 2D and 3D data representations of dataset 1b) obtained by t-SNE and RF-t-SNE (cf.[8])

countries occupy overlapping regions, which form together a massive, almost triangular shaped point cloud. Thus, Isomap produces the most informative data visualization compared to PCA, Sammon mapping, and LLE, so far.

Figure 4.6 illustrates the two-dimensional maps determined by t-SNE and its newly proposed variant RF-t-SNE. It demonstrates immediately that none of the four evaluated dimensionality reduction techniques can compete with these two approaches. The maps resulting from t-SNE and RF-t-SNE exhibit a clear dependence of the loads of the vehicles on the countries where they are driven. Both methods project the vast majority of the vehicles of the same country onto two-dimensional data points which lie in the same neighbourhood in the map. Thus, clearly distinguishable groups of countries can be identified, such as the pink one on the left hand side of the map produced by RF-t-SNE that is formed by the cars of country 6.

Furthermore, the plots reveal that in some countries the vehicles seem to be used in a similar manner, e.g., those of countries 2, 4, 5, and 8. Interestingly, it has to be mentioned that the latter countries are on the same continent, while none of the remaining eight countries belongs to this particular continent.

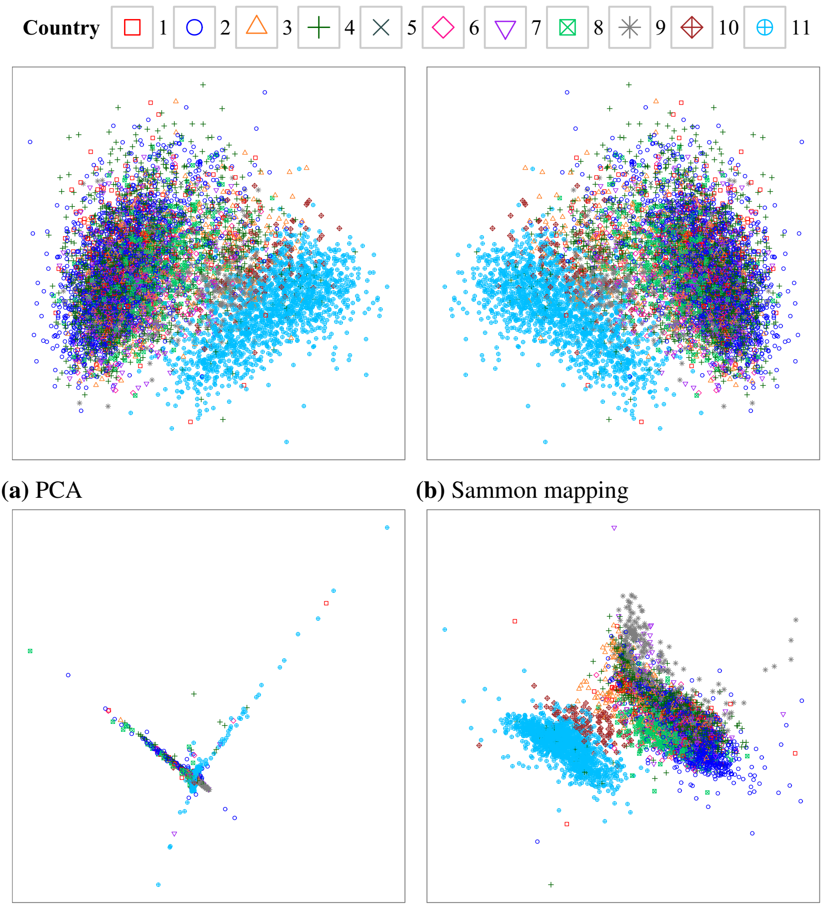
At first glance, the maps obtained by t-SNE and RF-t-SNE seem to be very similar. However, there are some reasons why the outcome of RF-t-SNE is slightly more informative than the one produced by t-SNE. First, in contrast to t-SNE, RF-t-SNE succeeds to project the vehicles of the countries 6 and 12 onto clearly distinguishable clusters. Secondly, the four clusters in the upper right quarter of the plot of RF-t-SNE are separated more clearly compared to the corresponding groups in the map obtained by t-SNE. Thus, even without the colouring, Figure 4.6b would facilitate a visual identification of several, separated clusters.

Finally, a three-dimensional map is computed by RF-t-SNE with the goal of studying whether it uncovers even more structure of the data than its two-dimensional counterpart. Figure 4.6c illustrates this map. The clusters that have already been identified in Figure 4.6b can be also spotted in this three-dimensional plot. However, besides these already known groups, no new insights into the data are gained by incorporating the additional dimension.

Thus, there is no advantage of computing a three-dimensional data representation over determining a two-dimensional one for this dataset. Nevertheless, these maps may be used to support the conclusion that are drawn from the corresponding two-dimensional map.

Analysis of dataset 2b)

Conducting the same analysis on dataset 2b) yields similar results. The computed maps of the corresponding load spectra are shown in Figures 4.7 and 4.8.



(a) PCA

(b) Sammon mapping

(c) LLE

(d) Isomap

Figure 4.7: Visualizations of the 2D data representations of dataset 2b) obtained by PCA, Sammon mapping, LLE, and Isomap (cf. [7])

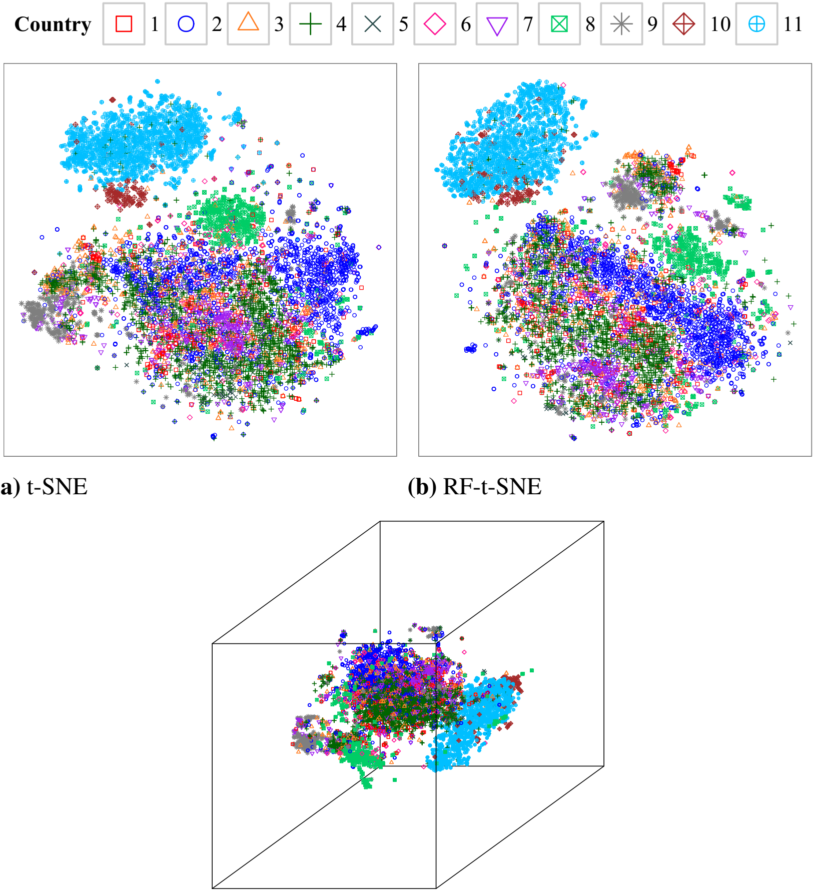
The two dimensionality reduction techniques PCA and Sammon mapping lead to two almost identical data representations. The only difference is that the map produced by PCA has to be reflected along an imaginary y-axis to obtain the outcome of Sammon mapping, as illustrated by Figures 4.7a and 4.7b. Each plot shows two major dense and overlapping point clouds: One is dominated by the vehicles of country 11, while the other one contains almost the rest of the cars.

The result of LLE on dataset 2b) is similar to the one achieved on dataset 1b), as can be seen in Figure 4.7c. Thus, it is still not very informative. The vast majority of the vehicle representations collapse into two narrow lines, which cross each other near the midpoint of the plot.

In contrast, Isomap does a better job, as shown in Figure 4.7d. Its outcome is similar, but slightly better than the one achieved by PCA and Sammon mapping, because the two major clusters are separated more clearly. Moreover, the region in between these two groups is mainly occupied by the vehicles of country 10. On the contrary, this region contains cars of many different countries in the maps produced by PCA and Sammon mapping, respectively.

Nevertheless, Figure 4.8 shows that the most informative results are again achieved by t-SNE and its newly proposed variant RF-t-SNE. The data representations, which are computed by these two techniques, allow a clear distinction of the vehicles of countries 10 and 11 from the remaining ones. Thus, a strong indication is given that the vehicles of these two main groups are stressed in a different manner. Interestingly, it has to be noted that the countries 10 and 11 are not only situated on the same continent, but are also the only two countries out of the eleven studied ones that lie on this particular continent. Furthermore, the remaining nine countries are in turn also part of a single continent. Moreover, the predominant environmental conditions of this continent are pretty similar in each of its countries. Therefore, it is not surprising that the vehicles of these countries seem to be stressed in a similar manner. However, the outcome of RF-t-SNE, which is shown in Figure 4.8b, uncovers a little bit more structure of the latter cluster by subdividing it into one huge and three minor groups that are situated in the upper right part of the plot. On the contrary, in the map produced by t-SNE, there are no minor, separable groups within this large cluster identifiable, as shown in Figure 4.8a. Hence, RF-t-SNE delivers again the most insight into the structure of the analysed dataset.

Moreover, Figure 4.8c shows the visualization of a three-dimensional data representation that is computed by RF-t-SNE. However, the map does not uncover any new structure that is not visible in the corresponding two-dimensional map, which is shown in Figure 4.8b. Therefore, incorporating the additional dimension does not yield any remarkable benefit for this dataset.



(a) t-SNE

(b) RF-t-SNE

(c) RF-t-SNE (3D)

Figure 4.8: Visualizations of the 2D and 3D data representations of dataset 2b) obtained by t-SNE and RF-t-SNE (cf. [7])

4.4 Case study: Visual distinction of component failures

Next, the applicability of the discussed dimensionality reduction techniques is studied with the goal of identifying, whether a vehicle is likely to have failed because of a certain type of usage or stress pattern which is hidden in the recorded load spectrum data. Since it is known that the faulty HEV of dataset

Table 4.3: Parameter setting for each of the studied dimensionality reduction techniques that are applied to dataset 2a)

Dim. reduction technique	**Dataset 2a)**
PCA	–
Sammon mapping	$n_{iter} = 1000$
LLE	$k = 47$
Isomap	$k = 47$
t-SNE	$perplexity = 47$, $n_{iter} = 1000$
RF-t-SNE	$n_{tree} = 5000$, $n_{forest} = 50$, $perplexity = 47$, $n_{iter} = 1000$

2a) all failed because of the same stress pattern, in contrast to those of dataset 1a), only the former dataset is analysed.

Moreover, in Chapter 3.5.2 it has been shown that the method rf_{Gini} achieves the best classification performance on dataset 2a), while using only 50 variables that it has automatically chosen from the 823 available ones. Hence, it is additionally analysed whether the results of the studied dimensionality reduction techniques can be improved if these methods are also applied to these 50 attributes, exclusively.

The two-dimensional data representations of dataset 2a) are computed using the same R implementations of the methods PCA, Sammon mapping, LLE, Isomap, t-SNE, and RF-t-SNE, as introduced in Chapter 4.3.

4.4.1 Preprocessing and parametrization

The same preprocessing as explained in Chapter 4.3.1 is applied to dataset 2a). Table 4.3 shows the parameter setting for each studied dimensionality reduction technique. The only difference to the parameter settings of the previous case study concern the value of the parameter k and the *perplexity*. Since dataset 2a) contains exactly 47 vehicles that suffer from a failure of the hybrid car battery, and because 47 lies within the value range that is recommended by the developers of t-SNE, both mentioned parameters are set to this value.

4.4.2 Results

First, the two-dimensional maps are presented, which result from applying the discussed dimensionality reduction techniques to the full, unreduced dataset 2a). Afterwards, the same analysis is repeated, while using only those 50 features that have been selected by method rf_{Gini} in Chapter 3.5.2.

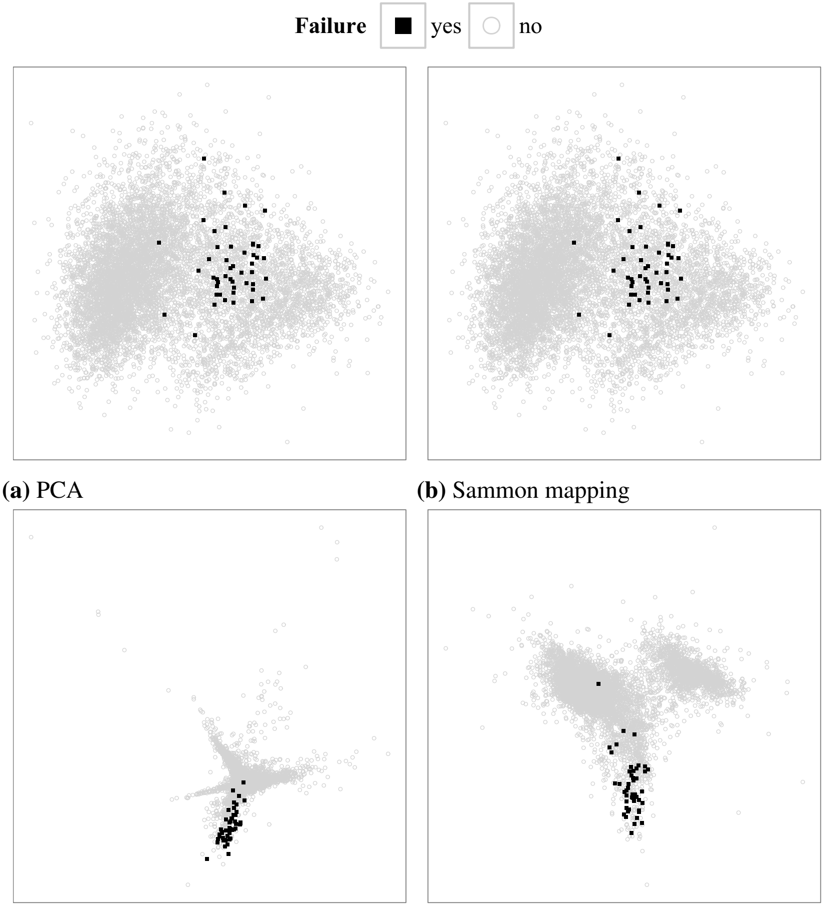
The determined two-dimensional map points of the load spectrum data that reflect the data status of the vehicles suffering from a failure of the hybrid car battery at the time of failure are depicted by black filled squares in all upcoming figures. Otherwise, the remaining data representations of the “healthy” vehicles are visualized by light grey coloured circles. Again, it has to be emphasized that the information about the health status of the hybrid car battery of the vehicles is not used during the dimensionality reduction process, but is only utilized to colourize the determined map points, afterwards.

Analysis of the full dataset 2a)

The computed two-dimensional data representations of the full dataset 2a) using the methods PCA, Sammon mapping, LLE, Isomap, t-SNE, and RF-t-SNE are illustrated by Figures 4.9 and 4.10.

Figures 4.9a and 4.9b present the maps created by PCA and Sammon mapping, respectively. Both maps are visually identical with the failed vehicles being spread over a large region in the middle of each plot. Thus, there is no indication that the vehicles suffering from a failure of the hybrid car battery are stressed in a different manner compared to their “healthy” counterparts.

The results of LLE and Isomap are shown in Figures 4.9c and 4.9d, respectively. LLE produces a map where the majority of all vehicles is concentrated in a small region. Also the failed cars can be found there, while most of them are lying in the lower half of the dense, star-shaped cloud of points. However, they do not form any recognizable cluster on their own. The visualization of the outcome of Isomap exhibits two almost separated regions with high density. Thereby, the failed vehicles only occupy the southern part of the larger of these two clusters. However, again these vehicles do not form a dense region on their own. Nevertheless, since in both maps the data representations of the faulty cars are situated outside the regions with the highest densities, at least it can be supposed that these vehicles are stressed in a different way then the majority of the “healthy” cars.



(a) PCA

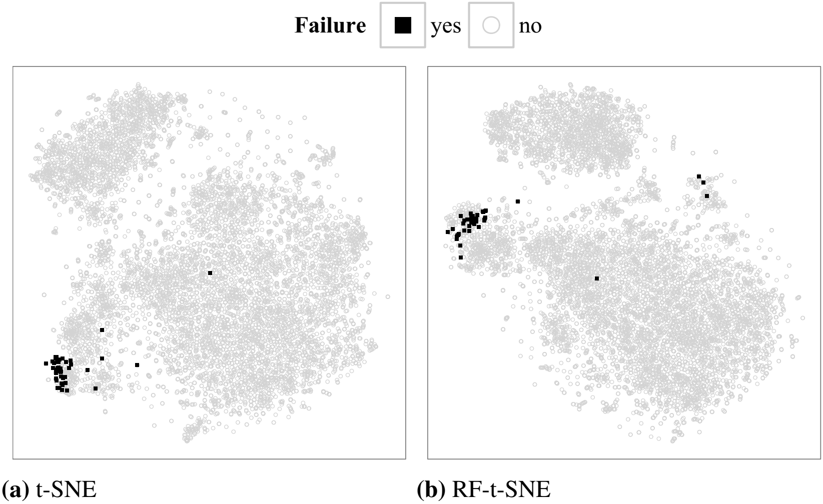
(b) Sammon mapping

(c) LLE

(d) Isomap

Figure 4.9: Visualizations of the 2D data representations of the full dataset 2a) obtained by PCA, Sammon mapping, LLE, and Isomap

In Figures 4.10a and 4.10b the results of t-SNE and RF-t-SNE are presented, respectively. In contrast to the former outcomes, most of the failed vehicles occupy a dense region in each of the two plots. Thus, both plots indicate that these vehicles have been exposed to different patterns of stress compared to the vast majority of the “healthy” cars. Moreover, both maps show that some



(a) t-SNE **(b)** RF-t-SNE

Figure 4.10: Visualizations of the 2D data representations of the full dataset 2a) obtained by t-SNE and RF-t-SNE

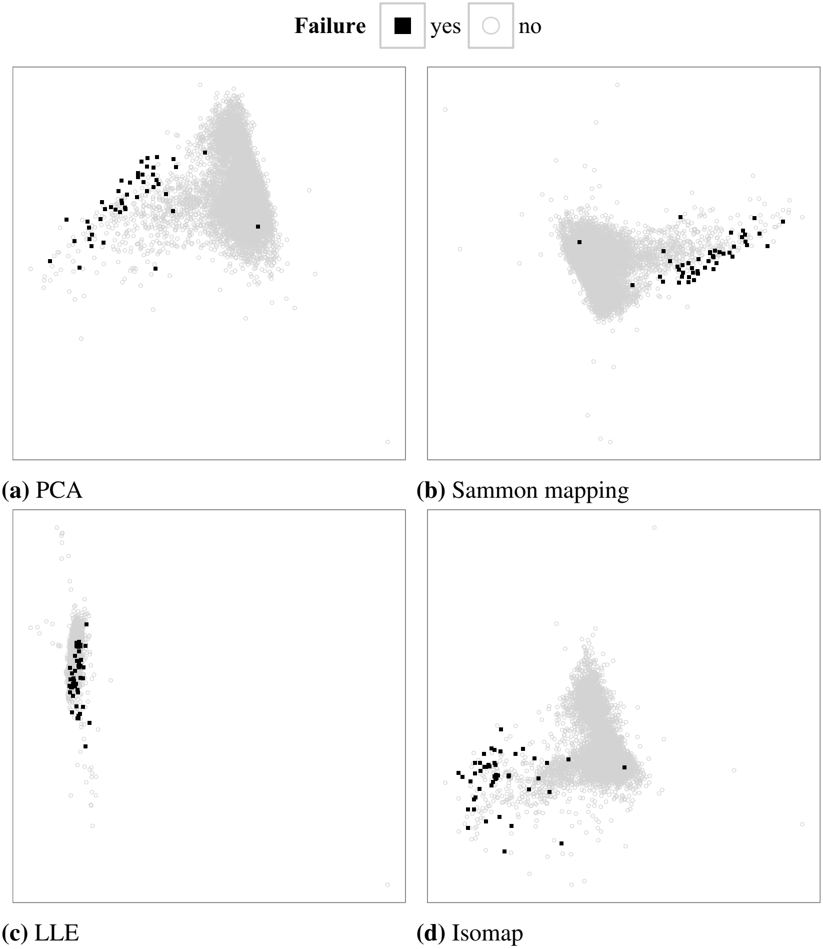
faulty cars seem to be stressed in a slightly different manner than the other failed vehicles. In the map produced by t-SNE there are 6 such outliers, as shown in Figure 4.10a, while Figure 4.10b illustrates that the one obtained by RF-t-SNE contains 5 of these outliers. However, RF-t-SNE projects 3 out of the 5 outliers onto a narrow neighbourhood, while these kind of data points are spread over a less dense region in the map of t-SNE.

In summary, in this case study the two variants of t-SNE outperform the other discussed dimensionality reduction techniques, while producing similar results.

Analysis of the reduced dataset 2a)

Now, the dimensionality of dataset 2a) is reduced by selecting only those 50 variables that have been finally selected by the classification method rf_{Gini} in the case study conducted in Chapter 3.5.2. The new results are illustrated by Figures 4.11 and 4.12.

Figures 4.11a and 4.11b present the maps created by PCA and Sammon mapping. In contrast to the corresponding Figures 4.9a and 4.9b, where the failed



(a) PCA

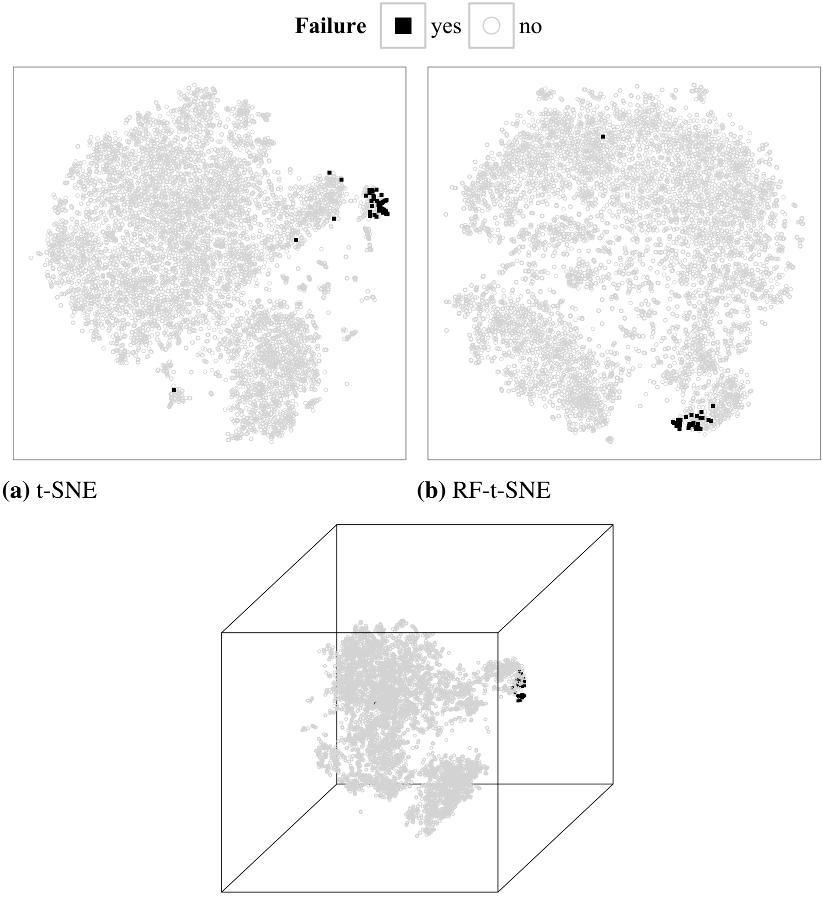
(b) Sammon mapping

(c) LLE

(d) Isomap

Figure 4.11: Visualizations of the 2D data representations of the reduced dataset 2a) obtained by PCA, Sammon mapping, LLE, and Isomap

vehicles are situated in the middle of the large point cloud that is formed by the "healthy" cars, now the faulty cars are projected onto points lying outside the area which is occupied by the vast majority of the "healthy" cars. However, they are still spread over a larger region of the chart, i.e., they do not form a dense cluster. Nevertheless, the new maps indicate that the faulty vehicles seem to be stressed differently than the majority of the remaining HEV fleet.



(a) t-SNE

(b) RF-t-SNE

(c) RF-t-SNE (3D)

Figure 4.12: Visualizations of the 2D and 3D data representations of the reduced dataset 2a) obtained by t-SNE and RF-t-SNE

The outcome of LLE is shown in Figure 4.11c. The map is slightly worse than Figure 4.9c, because now almost all cars are collapsed into a narrow, dense point cloud. Thus, faulty cars can not be distinguished from "healthy" ones.

The new map produced by Isomap is presented by Figure 4.11d. In contrast to the corresponding map of the previous analysis that is illustrated by Figure 4.9d, the faulty vehicles occupy a less compact region now. However, the majority of these cars lies outside the point cloud that is formed by the vast majority of “healthy” vehicles. For this reason, reducing the dimensionality of the dataset also leads to slightly better results for this method.

Finally, Figure 4.12 exhibits the new results of t-SNE and RF-t-SNE, respectively. Comparing these maps to the corresponding former ones, which are shown in Figure 4.10, it can be noticed that the outcome of RF-t-SNE has improved, while the one of t-SNE almost has stayed the same. RF-t-SNE now projects all faulty vehicles except one onto a compact and dense cluster, while in the map produced by t-SNE there are still 5 outliers with respect to the class of failed cars.

Furthermore, also for this dataset, it is not worth to compute a three-dimensional data representation of the selected load spectrum data of dataset 1a) in comparison with its two-dimensional counterpart, as shown in Figure 4.12c. The cluster that is formed by the failed vehicles does not become better separable from the remaining cars by simply adding one dimension. It only affirms the conclusions that have been drawn on the basis of Figure 4.12b.

In summary, RF-t-SNE produces the most informative map on the reduced version of dataset 2a). Moreover, using only the variables that have been selected by the final model of classifier rf_{Gini}, improves most of the two-dimensional data representations that are obtained by the discussed dimensionality reduction techniques.

4.5 Conclusion

In this chapter, the applicability of several dimensionality reduction techniques has been studied when they are applied to load spectrum data. Thereby, the following two different aims have been pursued.

First, these methods have been used to analyse whether the stress and usage patterns of HEV of the same car type depend on the countries, in which these vehicles are driven in. In particular the results of the newly proposed approach RF-t-SNE have shown that there is an evident, country-related structure in the two studied load spectrum datasets. In general, these results coincide with the expectations of the author, because the characteristics of the countries in the datasets, e.g., the predominant environmental conditions, are very heterogeneous and can influence directly or indirectly the driving or usage behaviour of the vehicles. For example, the countries are very versatile regarding their congestion level, according to the traffic index listed in [116]. Some of them suffer from many traffic jams, while this is rather a rare event in other countries. Moreover, the climate can vary a lot, e.g., some countries have a very dry and hot climate during the whole year, whereas others have a moderate or cold one.

Secondly, it has been demonstrated with another case study that the newly proposed technique RF-t-SNE may support the decision whether a particular component of a vehicle failed because of a certain stress or usage pattern. Thus, it may also help to identify, solely on the basis of the recorded load spectrum data, whether any two cars that suffer from a failure of the same component of the power-train are also likely to suffer from the same type of failure. Hence, it may not only help to improve the results of a classification task, as conducted in Chapter 3, but also may support the decision which faulty components should be bought back to get examined thoroughly in the laboratory.

5 Identifying usage and stress patterns in a vehicle fleet

In Chapter 3, methods have been proposed that facilitate a load spectrum based distinction between vehicles with and those without a failure of a hybrid component, whereas algorithms that allow a visual detection of structure such as clusters in the data have been discussed in Chapter 4. However, a common disadvantage of the approaches, which performed best on the studied datasets, is that they are all “black box” models, i.e., they do not allow to gain interpretable insights into the analytical relationship between the data and the obtained results. More precisely, it remains unknown which patterns in the data provoke the classifier rf_{Gini} to assign a certain label to an instance on the one hand, and which ones induce particular objects to form a cluster in the low-dimensional maps, produced by method RF-t-SNE, while others do not, on the other hand.

These kinds of information are essential for engineers though, e.g., in order to understand what stress and usage patterns are harmful for a considered hybrid component to be able to improve its reliability. Hence, this chapter discusses prominent methods of the category of so-called *rule learning* [43] algorithms. Not only their usability for gaining insights into load spectrum data is studied, but also new approaches are proposed that aim at overcoming the known sensitivity of these rule models towards highly correlated variables and noise in the data [10]. Hence, this chapter presents approaches to make the results better understandable that are obtained by the algorithms of the previous two chapters.

As in the previous two chapters, the required fundamentals are offered first, before the chapter continues with the explanation of common used as well as the newly proposed algorithms of this work. Finally, the applicability of the studied methods is evaluated using both an artificially created and a real-world dataset.

P. Bergmeir, *Enhanced Machine Learning and Data Mining Methods for Analysing Large Hybrid Electric Vehicle Fleets based on Load Spectrum Data*, Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart, https://doi.org/10.1007/978-3-658-20367-2_5

5.1 Fundamentals of rule learning

Referring to [68], a *rule R* is defined as

$$R : Body \rightarrow Head, \qquad \text{R. 5.1}$$

where *Body* denominates the *antecedent* and *Head* the *consequent* of *R*. Both are non-empty sets of *conditions*. Hence, a rule is an *implication*, where the consequent becomes true if all the conditions that form the antecedent are fulfilled.

In this work, only a special kind of rules, i.e., *classification rules* are considered. In that case, the consequent *Head* is given exclusively by the class assignment, i.e., *Head* is of the form $class = g$ with class label $g \in \mathscr{G}$. In contrast, the antecedent *Body* is given by at least one condition of the form $X_j \Theta q_j$, where $\Theta \in \{<, =, >, \leq, \geq\}$ is a relational operator, X_j is a variable, and q_j is a constant value. If *Body* contains several conditions, then these “variable-value comparisons” are combined with the logical conjunction operator that is denoted by “$\wedge$”. Moreover, it is said that a rule *R* “applies to” or “covers” an instance if its observed values satisfy all the conditions that are specified in the antecedent of *R*. Then, the statement given in *Head* becomes true for such an instance, i.e., it is assigned the class label *g* that is itemized in *Head*.

Therefore, this kind of rules can simply be read as “IF *Body* is true, THEN *Head* becomes also true”. However, if an object does not satisfy all the conditions that are specified in *Body*, no prediction about its class can be made at all, i.e., any class $g \in \mathscr{G}$ can be the correct one [10].

A classification rule *R* that is learned from some automotive data, for example, may have the following appearance:

$$\begin{aligned}&(T_{batt} > 70\,^{\circ}\mathrm{C}) \wedge (I_{batt} \leq -300A) \wedge (SoC_{batt} \leq 10\%) \\ &\rightarrow (class = failure)\,,\end{aligned} \qquad \text{R. 5.2}$$

where T_{batt}, I_{batt}, and SoC_{batt} designate the temperature, the current, and the state of charge of the hybrid car battery, respectively. In simple terms, this rule claims that a hybrid vehicle battery is likely to be or become faulty if it is exposed to a temperature higher than 70 °C and to a current less than $-300A$,

Table 5.1: Contingency table for an arbitrary rule $R : Body \rightarrow Head$ [68]

	B	$\bar{B}$	$\sum$
H	$n(B \cap H)$	$n(\bar{B} \cap H)$	$n(H)$
$\bar{H}$	$n(B \cap \bar{H})$	$n(\bar{B} \cap \bar{H})$	$n(\bar{H})$
$\sum$	$n(B)$	$n(\bar{B})$	N

when its state of charge is less than 10% at the same time. It has to be noted that this rule serves only as an illustrative example to be able to explain the meaning and syntax of classification rules, i.e., it is negligible at the moment if it makes sense from an engineering point of view.

Finally, the *length* of a rule is equal to the number of conditions forming its antecedent.

5.1.1 Rule evaluation measures

Before the rule mining algorithms that are studied and proposed in this work are explained in detail, it has to be clarified briefly how the quality of classification rules can be assessed. Similar to Chapter 3.1.1, rule evaluation measures can be defined on the basis of a contingency table. Table 5.1 presents such a table for an arbitrary rule, given as $R : Body \rightarrow Head$. Let B and H designate the sets of objects for which the antecedent and consequent of the rule are fulfilled, respectively. Moreover, let $\bar{B}$ and $\bar{H}$ denote the complements of these two sets, i.e., the sets of instances for which the antecedent and consequent conditions of R are not satisfied, respectively.

Furthermore, let the cardinality of a set S be denoted by $n(S)$. Thus, the term $n(B \cap H)$ stands for the number of instances that are covered by the rule R and that are correctly assigned the class label that is specified in the consequent of R. Additionally, the proportion of instances in a set S according to the total number of objects N is denoted by $\hat{p}(S) = \frac{n(S)}{N}$.

Using this notation, the *coverage (cov)* [68] of a rule R is defined, as follows:

$$cov(R) = \hat{p}(B). \qquad \text{Eq. 5.1}$$

It measures the proportion of instances that fulfil all the antecedent conditions of R. It may be interpreted as a measure of generality of R because a low coverage value indicates that there are only few objects in the studied dataset to which the considered rule applies to.

Moreover, the *support (supp)* of a rule R is given by

$$supp\,(R) = \hat{p}\,(B \cap H)\,. \qquad \text{Eq. 5.2}$$

It is a measure for the fraction of objects that are covered by R and that truly belong to the class g that is assigned to them by the rule.

Finally, these two measures can be combined to define the *confidence (conf)* [122] or *accuracy* of a rule R, as follows:

$$conf(R) = \frac{supp\,(R)}{cov\,(R)} \qquad \text{Eq. 5.3}$$

It is a measure of the “pureness” or “correctness” of a rule R, i.e., it tells you how sure you can be that the prediction made by the consequent is true, if R applies to an instance. Thereby, a confidence value of 1.0 indicates for a rule R that it covers only instances of the class that is specified in its consequent.

It has to be noted that the mentioned measures only assess the quality or properties of a *single* rule, i.e., they are not defined for evaluating the quality of an entire set of rules. However, in this chapter the focus only lies on evaluating the quality of individual rules, because the primary goal is to find rules that describe interesting patterns in the data. In particular, it is out of scope of this work to build rule based classification models, i.e., it is not aimed at creating rule sets that classify a dataset correctly.

5.2 Rule learning methods

Now, the rule learners that are investigated in this thesis are briefly explained. Namely, these are *Repeated incremental pruning to produce error reduction (RIPPER)*, *C5.0rules*, and some newly proposed methods that try to extract rules from or at least learn them on the basis of RF classification models.

5.2.1 RIPPER

Cohen proposed in [29] the popular rule learning method named *Repeated incremental pruning to produce error reduction (RIPPER).* It is an enhanced version of the formerly developed approach called *Incremental reduced error pruning (IREP)* [42]. This algorithm generates successively rules for the different, classes $g \in \mathscr{G}$ that are given in a dataset $\mathscr{D}$. Thereby, classes are processed in ascending order with respect to their sizes, i.e., RIPPER always starts with the smallest class, it ends after processing the second largest one, while the largest class becomes the default one, i.e., no rules are induced for this class. Hence, instances that do not satisfy any of the rules that are learned for the smaller classes are assigned the label of the largest class by default.

More precisely, RIPPER applies IREP to create an initial set of rules for each class g [124]. Therefore, it begins with randomly dividing the dataset into a *growing* and a *pruning* set, where the former one is used to learn rules, while the latter one is utilized to simplify these rules, afterwards [88]. In its original implementation, two third of the instances are put in the growing, and one third in the pruning set. However, the ratio between these two sets may also be seen as a tunable parameter of RIPPER.

A rule is grown by adding consecutively conditions of the form $X_j \leq q_j$ or $X_j \geq q_j$ to the antecedent of the rule until it covers only objects of the current class g from the growing set. Thereby, every value x_{ij} of each instance i in the growing set is tested as a candidate for the lower or upper bound value q_j that may restrict each attribute X_j. Here, the condition that maximizes the following information gain [124] is repeatedly added:

$$a\left(\log\left(\frac{a}{b} - \log\frac{A}{B}\right)\right),$$

where a and A denote the number of instances of the current class g, while b and B designate the total number of objects in the growing set that are covered by the rule *after* and *before* adding the new condition, respectively.

As soon as a rule has been created, the pruning phase starts, i.e., it is tried to simplify the rule again by employing the pruning set. Therefore, the final antecedent condition of the rule is eliminated iteratively as long as this leads to an increase in the value of

$$\frac{a+1}{b+2},$$

where a and b have the same meaning, as before.

The described growing and pruning phases are repeated for class g until at least one of the following stopping conditions is satisfied [124]:

- all instances of class g are covered by the generated rules; *or*
- the error rate exceeds 0.5 on the pruning set; *or*
- the *description length (DL)* [124] of the rule set and the objects is 64 bits greater than the smallest one that has been found so far.

Then, after a set of rules $\mathscr{R}_g = \{R_1, R_2, \ldots, R_k\}$ has been created for class g, an additional optimization step takes place. Therein, two alternative rules are produced for each rule $R_m \in \mathscr{R}_g$: a *replacement for* R_m and a *revision of* R_m [29]. For this purpose, the dataset is newly split into a growing and pruning set. However, all objects that satisfy any of the other rules $R_n \neq R_m \in \mathscr{R}_g$ are eliminated from the pruning set, this time. Then, a replacement rule R'_m for R_m is constructed from the scratch using the new growing set. Afterwards, it is pruned back in such a way that the error rate of the entire rule set $\{R_1, \ldots, R'_m, \ldots, R_k\}$ is minimized on the new pruning set.

Analogously, a revision rule R''_m for R_m is formed by adding new conditions to the antecedent of R_m instead of to the empty rule [29]. Finally, rule R_m is preserved or replaced by either rule R'_m or R''_m, depending on which of these three rules has the smallest DL [124]. If there still remain uncovered instances of class g, then the entire growing and pruning process is repeated for these objects. In general, this optimization step can be repeated a specified number of times.

5.2.2 C5.0rules

Another possibility is to deduce classification rules from a *C5.0* tree [64, 97], which has been explained in Chapter 3.2.2 in detail.

For this purpose, an unpruned *C5.0* tree is grown, first. Then, a rule is created for each path from the root to a leaf node of the tree. Thereby, all the variable splits of the path are combined by the logical *AND* ("$\wedge$") to form the antecedent of the rule. Moreover, the class label that is assigned to the leaf node becomes the consequent of the rule. Next, a logical *OR* is used to link all these rules to build a classifier that is equivalent to the original classification tree. Hence, only the form of representation of the classification model changes from a tree structure to a list of rules, so far.

Afterwards, it is checked for each rule whether it can be condensed by eliminating conditions from its antecedent. Therefore, a pessimistic estimate of the error rate for the complete rule is conducted analogously to the pessimistic pruning process of *C5.0* trees. This error rate serves as a benchmark. Next, it is tested for each condition in isolation whether its removal leads to a lower error rate than the benchmark. Hence, the error rate of the rule is iteratively recomputed after removing successively each condition from the antecedent of the rule. If the removal of any of these conditions leads to an improvement of the benchmark, then the condition whose elimination leads to the lowest error rate is removed. This pruning process is repeated until there are no more conditions that can be excluded from the antecedent to beat the benchmark.

After having pruned each individual rule in that manner, a global pruning step is performed on the whole set of rules to eliminate redundant and ineffective rules. Therefore, a measure based on the *minimum description length (MDL)* principle [100] is used to evaluate the performance and the complexity of a rule set at the same time. Thereby, the simpler collection of rules is favoured by this measure, if two rule sets perform equally. However, referring to [97], skipping this global pruning step may be advantageous for some applications.

Moreover, for each class, an initial rule set is formed in a such a way that each training instance satisfies at least one of the rules in this set. Then, an optimization method such as *simulated annealing* [60] is employed to remove or add rules until the rule set can not be improved with respect to the MDL metric [64].

Finally, when it comes to the classification of an instance using the obtained rule set, it may happen that the considered object satisfies more than a single rule of the set at the same time. This may provoke a conflict if these rules predict distinct classes. In order to resolve this conflict, the final class assignment is achieved by applying a weighted voting scheme. Thereby, each rule votes for a class with a weight that is equal to the confidence value of the rule. Then, these votes are aggregated and the class that achieves the highest total number of votes is assigned to the considered instance. Otherwise, if none of the rules in the final set applies to an instance, it gets assigned a default class, which is equal to the class that contains the most training samples that are not covered by any of the rules of the final rule set [49].

5.2.3 Random forest based rule learning methods

Although a RF model pools together several interpretable classification trees, itself belongs to the category of black box models. This is due to the fact that through the combination of many of these trees the comprehensibility of the ensemble is obscured. Therefore, extracting informative rules from RF models is a challenging task.

Hence, three approaches are proposed in this work that extract or at least learn rules on the basis of RF models. Thereby, all of them are motivated by the ideas and results given in [71].

Combined rule extraction and feature elimination

Since a RF is an ensemble of classification trees and since each of these trees can be transformed into a set of rules, as described in Chapter 5.2.2, a RF can also be converted into a set of rules by applying this transformation to each of its trees. Thereby, each individual path from the root to a leaf node of a tree forms a single rule, as before. Moreover, each classification tree is constructed in such a way that an arbitrary instance traverses it from the root to exactly one leaf node, i.e., an instance does never reach two distinct leaf nodes of such a tree. Thus, the leaf node assignment of the trees in a RF may be used to define a *binary encoding* of each instance $x_i \in \mathscr{D}$, as follows [71]:

1. Create a d-dimensional vector $b_i \in \{0,1\}^d$, where d is equal to the total number of leaf nodes in the forest, i.e., the elements of b_i have a one-to-one correspondence with the leaf nodes in the forest. Thereby, all leaf nodes of the entire forest are enumerated from 1 to d, starting with the leftmost leaf node of the first tree and ending with the rightmost one of the n_{tree}^{th} tree.

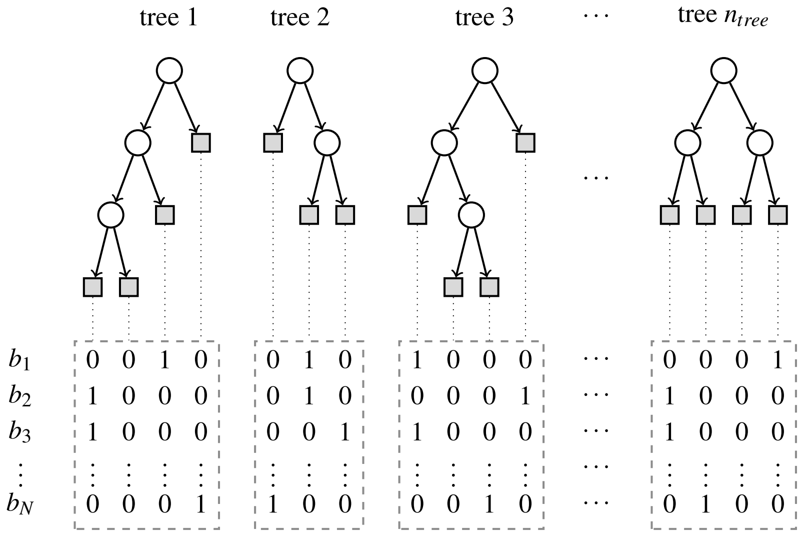

tree 1
tree 2
tree 3
· · ·
tree n_{tree}
b_1 0 0 1 0 | 0 1 0 | 1 0 0 0 · · · 0 0 0 1
b_2 1 0 0 0 | 0 1 0 | 0 0 0 1 · · · 1 0 0 0
b_3 1 0 0 0 | 0 0 1 | 1 0 0 0 · · · 1 0 0 0
b_N 0 0 0 1 | 1 0 0 | 0 0 1 0 · · · 0 1 0 0

Figure 5.1: Binary encoding of a RF model (cf. Fig. 1 in [71])

2. Let the element $b_{i,j}$ be one, if x_i falls into the j^{th} leaf node and zero otherwise. As a consequence, each vector b_i satisfies the following constraint: $\sum_{j=1}^{d} b_{ij} = n_{tree}$.

Figure 5.1 illustrates the described binary encoding process with a notional example, where each instance is assigned a randomly selected leaf node of each of the n_{tree} trees in the forest. Here, tree 1 determines the first four elements of the binary vector b_i, tree 2 the next three ones, and so forth. Furthermore, it is assumed that instance x_1 falls into the third leaf node of tree 1, into the second one of tree 2, and so on, resulting in $b_{1,3} = b_{1,6} = \ldots = 1$. Moreover, object x_2 ends up, e.g., in the first leaf of tree 1 and in the fourth leaf of tree 3. Thus, the elements $b_{2,1}$ and $b_{2,11}$ are also one, whereas, e.g., $b_{2,2}$, $b_{2,3}$, and $b_{2,4}$ are zero.

Moreover, for each tree that is depicted in Figure 5.1, its leaf nodes are connected with dotted lines to the elements of b_i to which they correspond to. These elements are additionally framed with a dashed, grey rectangle to highlight on which tree they are based on.

Now, the idea is to extract useful rules from the RF model by applying an embedded feature selection method, e.g., *linEnet*, to the newly encoded dataset $\mathscr{D}_B = (\mathbf{B}, \mathbf{y})$, with $\mathbf{B} = (b_1, b_2, \ldots, b_N)^T$, that employs a classifier that computes weights for each variable B_j, which are related to the relevance of B_j for the classification result. Since variable B_j corresponds with a rule R_j in this learning task, the latter rule is removed if its assigned weight is zero. In that case, it is not relevant for the classifier to distinguish between the instances of the two classes. It is recommended to use a classifier that employs a lasso or elastic-net penalty for implicit feature selection because these methods promote sparse weight vectors, i.e., small and consequently easier to interpret rule sets.

Figure 5.2 illustrates the main steps of the rule learning method named *Combined Rule Extraction and Feature Elimination (CRF)* that is a slightly modified version of the approach presented in [71]. In the beginning, the current feature set F and the so far optimal attribute set F_{opt} contain all of the available p variables. Furthermore, the entire rule set $\mathscr{R}$ as well as the current optimal rule set $\mathscr{R}_{opt}$ are initialised as empty sets, while the accuracy value *acc* of the best embedded feature selection model is set to zero.

Then, each iteration starts with building a RF model for the given binary classification problem. Thereby, only the features in F are used and the RF parameters are tuned on a predefined parameter grid using CV, as precisely described in Chapter 3. Afterwards, the resulting RF model is employed to create a binary encoding $\mathbf{B}$ for the input matrix $\mathbf{X}$, as explained above. Next, an embedded feature selection method such as *linEnet* (cf. Chapter 3.2.4) is applied to the transformed dataset $\mathscr{D}_B = (\mathbf{B}, \mathbf{y})$ to distinguish between the instances of the two classes $g \in \mathscr{G}$ and to compute a weight β_j for each attribute B_j that corresponds with the RF rule R_j. Again, CV is used to tune the model specific parameters and to assess the model's performance on the data. Then, all rules R_j with $\beta_j \neq 0$ are inserted in the current rule set $\mathscr{R}_{new}$ and the variables that are contained in the antecedents of these rules build the current feature set F_{new}.

Next, a couple of pruning strategies are applied to simplify each individual rule $R_j \in \mathscr{R}_{new}$ as well as to remove irrelevant or redundant rules from this set. In order to prune an individual rule R_j, either the OOB samples of the tree in the RF model that induced R_j or the entire training data may be used, in this newly proposed version of CRF. Thereby, it is tested for each condition of the antecedent of R_j if its removal results in an higher confidence value of the current rule. If this is the case, then the condition is eliminated. Otherwise the next unchecked condition is tested, where the conditions of the antecedent of R_j are evaluated in reverse order, i.e., the last condition is checked first,

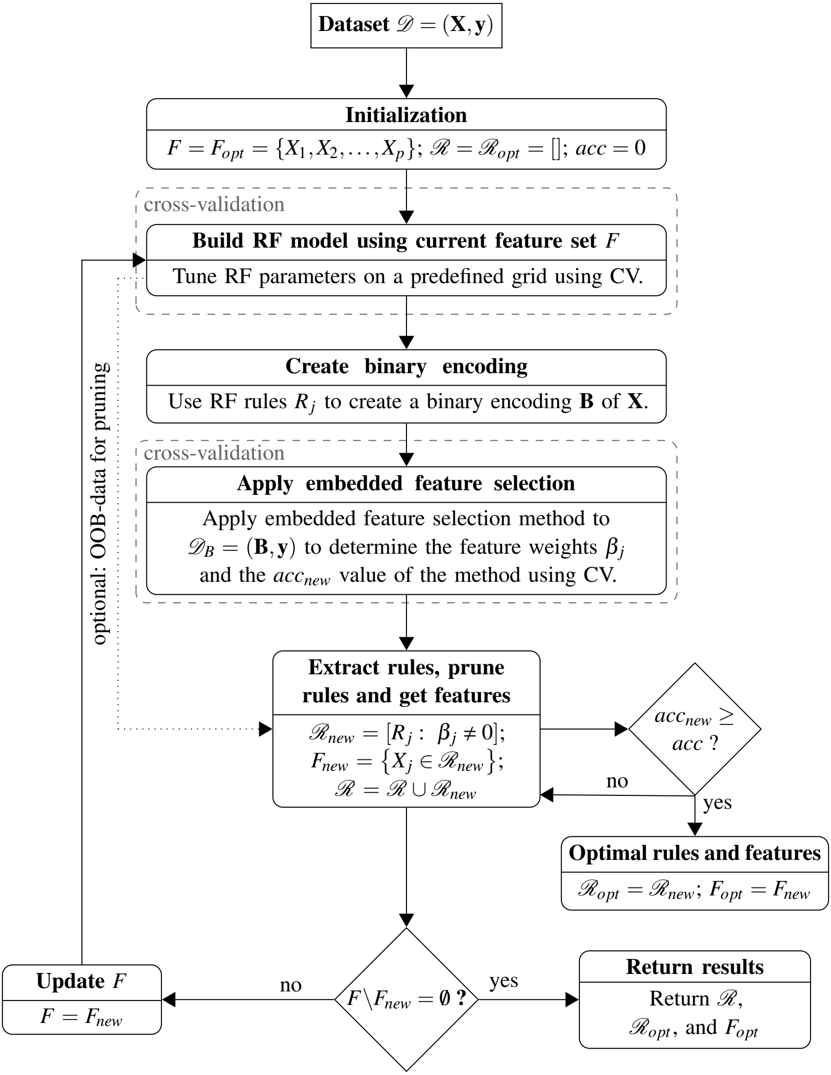

Dataset $\mathscr{D} = (\mathbf{X}, \mathbf{y})$
Initialization
$F = F_{opt} = \{X_1, X_2, \ldots, X_p\}$; $\mathscr{R} = \mathscr{R}_{opt} = []$; $acc = 0$
cross-validation
Build RF model using current feature set F
Tune RF parameters on a predefined grid using CV.
optional: OOB-data for pruning
Create binary encoding
Use RF rules R_j to create a binary encoding $\mathbf{B}$ of $\mathbf{X}$.
cross-validation
Apply embedded feature selection
Apply embedded feature selection method to $\mathscr{D}_B = (\mathbf{B}, \mathbf{y})$ to determine the feature weights β_j and the acc_{new} value of the method using CV.
Extract rules, prune rules and get features
$\mathscr{R}_{new} = [R_j : \beta_j \neq 0]$; $F_{new} = \{X_j \in \mathscr{R}_{new}\}$; $\mathscr{R} = \mathscr{R} \cup \mathscr{R}_{new}$
$acc_{new} \geq acc$?
no
yes
Optimal rules and features
$\mathscr{R}_{opt} = \mathscr{R}_{new}$; $F_{opt} = F_{new}$
Update F
$F = F_{new}$
no
$F \setminus F_{new} = \emptyset$?
yes
Return results
Return $\mathscr{R}$, $\mathscr{R}_{opt}$, and F_{opt}

Figure 5.2: Workflow for extracting optimal rule and feature sets from RF models using the rule learning method CRF [71]

then the second last, and so forth. After each rule $R_j \in \mathscr{R}_{new}$ is pruned, the entire rule set $\mathscr{R}_{new}$ is reduced, as follows: If the antecedents $Body_j$ and $Body_k$ of two arbitrary rules $R_j, R_k \in \mathscr{R}_{new}$, which both predict the same class, are formed by conditions on exactly the same variables, i.e., if all variables X_j that are element of $Body_j$ are also contained in $Body_k$ and vice versa, then the rule is eliminated that has the lower support value, while achieving a less or equal confidence value.

After the pruning phase, it is tested whether the achieved performance of the current embedded feature selection model is equal or better than the one of the best model, so far. If this is the case, then $\mathscr{R}_{new}$ becomes the current optimal rule set $\mathscr{R}_{opt}$. The same applies to the feature set F_{new} and F_{opt}. Moreover, if additionally the cardinality of the new feature set F_{new} is smaller than the one of the current feature set F, then F is replaced by F_{new} and the next iteration is performed.

Finally, the algorithm stops as soon as the feature set F does not change any more, i.e., if $F \backslash F_{new} = \emptyset$. Then, all the rules that are extracted during the iterative process are returned as well as the optimal set of rules $\mathscr{R}_{opt}$ and features $\mathscr{F}_{opt}$.

This version of CRF enhances the algorithm proposed in [71] with incorporating the described pruning strategies into the iterative learning process. Furthermore, different embedded feature selection methods are employed in this work, namely *linEnet*, *logEnet*, *linLasso*, and *logLasso*. The corresponding CRF variants are denominated as $CRF_{linEnet}$, $CRF_{logEnet}$, $CRF_{linLasso}$, and $CRF_{logLasso}$. Moreover, if the OOB samples of each tree in the RF are used for rule pruning, then the term *OOB* is incorporated in the name of the corresponding method, e.g., $CRF^{OOB}_{linEnet}$.

Additionally, besides extracting only the determined optimal sets of rules and features, all rules that are extracted during the iterative process are returned to be able to additionally analyse the entire rule set, if desired. All these rules are evaluated in the case study that is conducted at the end of this chapter.

It has to be noted that the given dataset $\mathscr{D}$ is not split into a training and test set because the goal is not to build a generalizable, rule-based classification model, but to learn informative characteristics of the studied datasets. Loosely spoken, it may be seen more like a statistical analysis of a dataset, in which also the entire data is analysed.

A RF based RIPPER approach

The idea behind this newly proposed rule learning method named RF_{RIPPER} is to employ the RIPPER algorithm to learn rules not only from the entire training set, but also from each subspace of the feature space that is determined by the variables defining the split points of each classification tree in the RF. Thus, it is tried to avoid to learn uninformative rules from noisy or irrelevant features.

Therefore, the RIPPER method is applied to the entire training data, first. Secondly, a binary RF classifier is built with the goal to distinguish between the training instances of the two distinct classes $g \in \mathscr{G}$. Then, the variables which specify the splits of each of its n_{tree} classification trees are extracted separately from each tree. Afterwards, the RIPPER algorithm is applied to learn rules from each of the n_{tree} resulting subsets of attributes. Thereby, several possible parameter settings of this method are evaluated, while all the rules that result from each of these settings are recorded. Finally, all obtained sets of rules are merged into a single, large rule set $\mathscr{R}$. Since, $\mathscr{R}$ usually contains many redundant rules, it has to be simplified, again. For this purpose, the discussed pruning and simplification approaches that are proposed in Chapter 5.2.3, are applied to $\mathscr{R}$. Depending on the set of instances which is used for rule pruning, two variants of the new approach are studied in this Thesis, namely RF_{RIPPER} and RF_{RIPPER}^{OOB}.

In the end, if the remaining rule set is still too large, then it may be iteratively reduced by keeping only the best performing rules that cover those minority class instances that do not satisfy any of the other rules yet being in the final rule set. In each iteration, the rule that leads to the highest confidence value, while covering most of the minority class samples is added to the final rule set. Afterwards, the minority class instances that are covered by this rule are removed from the training set. The iteration process stops, when all minority class objects satisfy at least one of the rules that are contained in this rule set.

It is emphasised again that RF_{RIPPER} and RF_{RIPPER}^{OOB} may lead to rules that overfit the data and may consequently not be appropriate for building rule based classification models. However, this is no problem in this work, since here the goal of rule learning is only to find predominant patterns in the data. Hence, it is not important that these rules generalize well on unseen data.

A RF based C5.0rules approach

Instead of using *RIPPER* as rule learning method in the previous discussed approach, it is also possible to develop a RF based algorithm that employs *C5.0rules* in a similar manner. This new algorithm is denoted by $RF_{C5.0rules}$ and $RF^{OOB}_{C5.0rules}$ in the remainder of this Thesis, respectively. The major difference to RF_{RIPPER} is that *C5.0rules* does not only create rules for the minority, but also for the majority class data. On the contrary, *RIPPER* simply assigns each instance that is not covered by any minority class rule to the majority class by default. However, since the focus of this work lies on learning stress and usage patterns that may relate to failures of a particular hybrid component, it is out of interest to extract rules for "healthy" vehicles. Thus, rules which predict the latter class are simply discarded in this work.

5.3 Case study: Identifying stress patterns for component failures

Now, a case study is conducted to assess the applicability of the presented rule learning algorithms to identify interesting stress patterns in the load spectrum data of a large HEV fleet that may be related to failures of particular components of the hybrid power-train. Therefore, all the discussed rule learning methods are successively applied to a synthetic dataset and to the real-world dataset 2a), that has been introduced in Chapter 2.3. Before the results of each algorithm on both datasets are shown and discussed in detail, it is first explained how the synthetic dataset is generated and what preprocessing and parameter optimization steps are undertaken.

5.3.1 Synthetic data

In order to be able to assess the performance of the introduced rule learning approaches on load spectrum data, a synthetic dataset is created on the basis of the real-world dataset 2a). Therefore, 50 different vehicles and 5 distinct variables are selected randomly from the latter dataset. Then, an artificial vehicle is generated for each of these 50 instances. Thereby, the original feature vector is modified, as follows: The observed value of each of the selected

Table 5.2: Main characteristics of the generated synthetic dataset that has been created on the basis of the real-world load spectrum dataset 2a)

Characteristic	**Synthetic Dataset**
Number of vehicles	8131
Total number of load spectrum classes	823
Number of vehicles with synthetic failure type 1 (FT_1)	50
Number of vehicles with synthetic failure type 2 (FT_2)	25

5 variables X_j is set to a random, numeric value that lies either in the interval $[\mathscr{Q}_{0.10}(\mathbf{x}_j), \mathscr{Q}_{0.15}(\mathbf{x}_j)]$ or in $[\mathscr{Q}_{0.85}(\mathbf{x}_j), \mathscr{Q}_{0.90}(\mathbf{x}_j)]$. Here, $\mathscr{Q}_k(\mathbf{x}_j)$ with $0 < k < 1$ denotes the $(100 \cdot k)\,\%$-quantile value of the empirical distribution of X_j.

In that way, an artificial failure pattern is constructed that is given as a combination of comparatively low and high entries in distinct load spectrum classes with respect to the entire dataset, i.e., the whole HEV fleet. Thereby, it is assumed that extreme observed values for the selected variables are harmful, i.e., they provoke failures of a particular component such as the hybrid car battery. Since very high temperatures, for example, are known to be bad for hybrid car batteries, this assumption is far away from being unrealistic.

Moreover, a second failure type is induced by repeating the whole process for further 25 randomly selected vehicles and 3 different variables, chosen by chance.

Finally, all 75 modified feature vectors are labelled as “faulty”, whereas all the remaining real ones are assigned the label “healthy”, i.e., the information about the two different failure types is suppressed on purpose.

The reason for that is that usually there are different stress patterns which lead to a failure of the same hybrid component. However, often the only information that is provided by the workshop is that a particular component failed, i.e., no failure type distinction is offered.

Table 5.3 presents the resulting two failure types that are abbreviated as FT_1 and FT_2, respectively. It has to be noted that the interval boundaries vary slightly around the specified quantile values. This is due to the fact that the specified quantile values are determined using only the original data. As soon as

Table 5.3: The two distinct failure patterns that are inherent to the faulty vehicles in the synthetic dataset

Failure Type 1 (FT_1)	**Failure Type 2** (FT_2)
$\mathscr{Q}_{0.845}(\mathbf{x}_{240}) \leq X_{240} \leq \mathscr{Q}_{0.900}(\mathbf{x}_{240})$	$\mathscr{Q}_{0.848}(\mathbf{x}_{432}) \leq X_{432} \leq \mathscr{Q}_{0.899}(\mathbf{x}_{432})$
$\wedge$	$\wedge$
$\mathscr{Q}_{0.847}(\mathbf{x}_{407}) \leq X_{407} \leq \mathscr{Q}_{0.901}(\mathbf{x}_{407})$	$\mathscr{Q}_{0.849}(\mathbf{x}_{540}) \leq X_{540} \leq \mathscr{Q}_{0.900}(\mathbf{x}_{540})$
$\wedge$	$\wedge$
$\mathscr{Q}_{0.102}(\mathbf{x}_{437}) \leq X_{437} \leq \mathscr{Q}_{0.155}(\mathbf{x}_{437})$	$\mathscr{Q}_{0.102}(\mathbf{x}_{819}) \leq X_{819} \leq \mathscr{Q}_{0.152}(\mathbf{x}_{819})$
$\wedge$	
$\mathscr{Q}_{0.846}(\mathbf{x}_{488}) \leq X_{488} \leq \mathscr{Q}_{0.900}(\mathbf{x}_{488})$	
$\wedge$	
$\mathscr{Q}_{0.101}(\mathbf{x}_{495}) \leq X_{495} \leq \mathscr{Q}_{0.155}(\mathbf{x}_{495})$	

the original observations are replaced by the synthetic ones, these boundaries may be shifted a little bit. Moreover, they are presented in terms of quantiles with respect to the empirical distribution of the corresponding variable taken over the entire vehicle fleet rather than as numerical values. Thus, they are better interpretable because one immediately knows whether a failure is related to a stress pattern which is given by the combination of comparatively low or high values of some variables.

For notational convenience, $\mathscr{Q}_k$ is used for $\mathscr{Q}_k(\mathbf{x}_j)$ in rule conditions by now, because quantiles are always computed on the basis of the empirical distributions of the variables they restrict.

Hence, the goal of the later conducted analysis is to find out whether the studied rule learning methods are able to autonomously identify the patterns of FT_1 and FT_2 by applying them to the newly generated dataset.

Table 5.4: Parameter grid that is used for optimizing *RIPPER* and that is exploited by RF^{OOB}_{RIPPER} and RF_{RIPPER}

Parameter	**Values**
numFolds	$1,2,3,4,5$
minWeights	$1,2,5,10,20,25,\min\{\#\ minority\ class\ instances,50\}$
numOpt	$1,2,3,\ldots,10$
prune	*yes*, *no*
errorCheck	*yes*, *no*

5.3.2 Preprocessing and parametrization

The parameters of the two studied rule-based classification models *RIPPER* and *C5.0rules* as well as those of the proposed CRF variants are optimized over predefined parameter grids, while the remaining rule learning approaches exploit all possible settings that are specified in these grids. They are given by Tables 5.4, 5.5, and 5.6, respectively. Like in Chapter 3, CV is used to determine the optimal parameter settings, if required, where BAC is employed as performance measure for the two classifiers.

Table 5.4 shows the parameter grid for method *RIPPER* and its RF based variants RF^{OOB}_{RIPPER} and RF_{RIPPER}. Therein, parameter *numFolds* specifies the proportion of instances that build the growing and the pruning set, respectively. Here, one fold is used for pruning, while the remaining folds together form the growing set. Thus, if *numFolds* is set to 5 then 80% of the training data are contained in the growing and 20% in the pruning set. Furthermore, parameter *minWeights* specifies the minimum number of instances that have to be covered by a rule. In this study, it is allowed to take the values 1, 2, 5, 10, 20, 25, and $\min\{\#\ minority\ class\ instances,50\}$, analogue to parameter *minCases* of method *C5.0*, as explained in Chapter 3. Also the number of performed optimization runs may be set through parameter *numOpt*. It is adapted among the integers 1 to 10. Finally, the two boolean parameters *prune* and *errorCheck* allow to specify whether the *RIPPER* pruning strategy is performed and if the stopping criterion that checks the error rate on the pruning set is applied.

Table 5.5 presents the predefined parameter values which are used to optimize the rule-based classifier *C5.0rules* and that are exploited by the newly proposed

Table 5.5: Parameter grid that is used for optimizing *C5.0rules* and that is exploited by $RF_{C5.0rules}^{OOB}$ and $RF_{C5.0rules}$

Parameter	**Values**
w_{+1}	$0.1, 0.2, 0.3, \ldots, 2$
trials	$1, 2, 3, \ldots, 10$
minCases	$1, 2, 5, 10, 20, 25, \min\{\textit{\# minority class instances}, 50\}$
CF	$10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.05, 0.10, 0.25, 0.50$

approaches $RF_{C5.0rules}^{OOB}$ and $RF_{C5.0rules}$, respectively. Like in Chapter 3, the pruning parameter *CF* of *C5.0rules* is adjusted on the values 10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.05, 0.10, 0.25, and 0.50. Moreover, the parameter *minCases* that specifies the minimum number of cases that have to be covered by a rule is adapted among the values 1, 2, 5, 10, 20, 25, and $\min\{\textit{\# minority class instances}, 50\}$, similar to parameter *minWeights* in the *RIPPER* based models. Moreover, 1 to 10 boosting iterations are tried and a cost of one is used for false positives, whereas the cost C_{+1} for a false negative is set to $w_{+1} \cdot \frac{N_{-1}}{N_{+1}}$ with $w_{+1} = 0.1, 0.2, 0.3, \ldots, 2$, as in Chapter 3.

Finally, Table 5.6 shows the studied parameter grid for the CRF variants. It is very similar to the one that has been defined for the RF and ORF variants in Chapter 3. Since the RF model has to be re-fit in each iteration of CRF, parameter m_{try} is optimized over a set of dynamically adapted values here that depend on the current number of available variables p. Here, besides the default value of $\sqrt{p}$ [69], also $\frac{\sqrt{p}}{2}$ and the sequence $\left[\frac{p}{10}\right], \left[\frac{2p}{10}\right], \left[\frac{3p}{10}\right], \ldots, p$ are tried, where $[.]$ denominates the nearest integer function.

Moreover, the class-specific parameters $\textit{sampsize}_g$ and $\textit{cutoff}_g$ are only tuned once, i.e., in the first iteration of CRF, to speed up this time demanding algorithm. Afterwards, it is fixed to the determined optimal values.

The number of trees, n_{tree}, is again fixed to 300, like in Chapter 3. Since the goal is to eliminate a significant number of features in each iteration, i.e., rules in this case, only elastic net ($\alpha = 0.5$) and lasso ($\alpha = 1$) regularized linear and logistic regression are studied, respectively. For the latter models, the complexity parameter λ is optimized employing again the proposal made in [39] that has been explained briefly in Chapter 3.

Table 5.6: Predefined parameter grid for optimizing the studied CRF variants

Parameter	**Values**	**CRF variant**
m_{try}	$\frac{\sqrt{p}}{2}$, $\sqrt{p}$, $\left[\frac{p}{10}\right]$, $\left[\frac{2p}{10}\right]$, $\left[\frac{3p}{10}\right]$, $\ldots, p$	All
$cutoff_{+1}$	$0.10, 0.15, \ldots, 0.50$	All
$cutoff_{-1}$	$1 - cutoff_{+1}$	All
$sampsize_{+1}$	number of minority samples in current cv-training fold	All
$sampsize_{-1}$	$sampsize_{+1}$, $2 \cdot sampsize_{+1}$, $3 \cdot sampsize_{+1}$	All
n_{tree}	300	All
λ	cf. Section 2.5 in [39]	$CRF_{linEnet}$
		$CRF_{logEnet}$
		$CRF_{linLasso}$
		$CRF_{logLasso}$
α	0.5	$CRF_{linEnet}$
	0.5	$CRF_{logEnet}$
	1	$CRF_{linLasso}$
	1	$CRF_{logLasso}$

5.3.3 Results

The next two subchapters present the results that have been achieved on the synthetic dataset, which contains 75 vehicles that suffer from two artificially created failure types, and on the real-world dataset 2a), which covers 47 cars that have failed because of a particular failure type of the hybrid car battery. For both datasets it is studied which of the discussed rule learning algorithms creates the rule that achieves the highest confidence value among its competitors, while covering also as many faulty vehicles, as possible. Moreover, it is additionally analysed whether it is beneficial to apply the approach rf_{Gini}, which has outperformed its rivals in the very first case study of this Thesis, introduced in Chapter 3, to heavily reduce the number of variables first, i.e., before learning any rules.

For the RF and CRF variants, respectively, all rules that are generated during its iterative process are considered as candidates for the best rules, while only the most predictive rules with respect to classification performance are evaluated

Table 5.7: Identified variables of corresponding failure type per rule learning variant if all available attributes are used

	Failure Type 1 (FT_1)					**Failure Type 2 (FT_2)**		
Method	X_{240}	X_{407}	X_{437}	X_{488}	X_{495}	X_{432}	X_{540}	X_{819}
RIPPER	✓			✓	✓	✓		
RF^{OOB}_{RIPPER}	✓	✓	✓	✓	✓	✓	✓	✓
RF_{RIPPER}	✓	✓	✓	✓	✓	✓	✓	✓
C5.0rules	✓		✓	✓	✓	✓	✓	
$RF^{OOB}_{C5.0rules}$	✓	✓	✓	✓	✓	✓	✓	✓
$RF_{C5.0rules}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF^{OOB}_{linEnet}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF_{linEnet}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF^{OOB}_{linLasso}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF_{linLasso}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF^{OOB}_{logEnet}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF_{logEnet}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF^{OOB}_{logLasso}$	✓	✓	✓	✓	✓	✓	✓	
$CRF_{logLasso}$	✓	✓	✓	✓	✓	✓	✓	✓

for the rule-based classification algorithms *RIPPER* and *C5.0rules*. Thus, it is studied if it is sufficient to create rule-based models to identify stress related patterns in load spectrum data or if it is better to extract as much information, as possible.

Analysis of the synthetic dataset

It is started with the presentation of the results that are obtained on the synthetic dataset, which has been introduced in Chapter 5.3.1. As described in the latter chapter, there are two failure types that are abbreviated as FT_1 and FT_2 in this dataset. However, it is emphasized again that all vehicles that suffer either from FT_1 or FT_2 are assigned the same label, i.e., "faulty".

Thus, it is possible to study whether the rule learning methods are nevertheless able to generate useful rules that describe the two distinct failure patterns, which are inherent to FT_1 and FT_2, respectively. As mentioned earlier in this work, there is often no information about the distinct failure types of a particular failed component available. Hence, a good rule learning algorithm has to be capable to learn rules for different failure patterns autonomously. Table 5.3

Table 5.8: Rule R that maximises $conf_{FT_1}(R)$ per studied rule learning method if the entire variable set is exploited

Method	Antecedent of best rule R for FT_1	$n(B \cap \mathrm{FT}_1)$	$conf_{FT_1}(R)$
$RIPPER$	$(X_{240} \geq \mathscr{Q}_{0.845}) \wedge (X_{488} \geq \mathscr{Q}_{0.823}) \wedge (X_{495} \leq \mathscr{Q}_{0.155})$	**50**	**1.000**
RF^{OOB}_{RIPPER}	$(X_{240} \geq \mathscr{Q}_{0.816}) \wedge (X_{488} \geq \mathscr{Q}_{0.846}) \wedge (X_{495} \leq \mathscr{Q}_{0.153})$	49	1.000
RF_{RIPPER}	$(X_{437} \leq \mathscr{Q}_{0.155}) \wedge (X_{488} \geq \mathscr{Q}_{0.846}) \wedge (X_{495} \leq \mathscr{Q}_{0.155})$	**50**	**1.000**
$C5.0rules$	$(X_{240} > \mathscr{Q}_{0.560}) \wedge (X_{437} \leq \mathscr{Q}_{0.275}) \wedge (X_{488} > \mathscr{Q}_{0.823})$	50	0.676
$RF^{OOB}_{C5.0rules}$	$(X_{240} > \mathscr{Q}_{0.845}) \wedge (X_{488} > \mathscr{Q}_{0.844}) \wedge (X_{495} \leq \mathscr{Q}_{0.171})$	**50**	**1.000**
$RF_{C5.0rules}$	$(X_{240} > \mathscr{Q}_{0.798}) \wedge (X_{488} > \mathscr{Q}_{0.846}) \wedge (X_{495} \leq \mathscr{Q}_{0.166})$	**50**	**1.000**
$CRF^{OOB}_{linEnet}$	$(X_{240} > \mathscr{Q}_{0.845}) \wedge (X_{488} > \mathscr{Q}_{0.825}) \wedge (X_{495} \leq \mathscr{Q}_{0.227})$	50	0.980
$CRF_{linEnet}$	$(X_{240} > \mathscr{Q}_{0.840}) \wedge (X_{279} > \mathscr{Q}_{0.513}) \wedge (X_{445} > \mathscr{Q}_{0.954}) \wedge (X_{488} > \mathscr{Q}_{0.830})$	3	1.000
$CRF^{OOB}_{linLasso}$	$(X_{488} > \mathscr{Q}_{0.828}) \wedge (X_{495} \leq \mathscr{Q}_{0.159})$	50	0.848
$CRF_{linLasso}$	$(X_{437} \leq \mathscr{Q}_{0.157}) \wedge (X_{488} > \mathscr{Q}_{0.758}) \wedge (X_{495} \leq \mathscr{Q}_{0.162})$	50	0.943
$CRF^{OOB}_{logEnet}$	$(X_{437} \leq \mathscr{Q}_{0.155}) \wedge (X_{488} > \mathscr{Q}_{0.807}) \wedge (X_{495} \leq \mathscr{Q}_{0.277})$	50	0.943
$CRF_{logEnet}$	$(X_{437} \leq \mathscr{Q}_{0.157}) \wedge (X_{488} > \mathscr{Q}_{0.846}) \wedge (X_{495} \leq \mathscr{Q}_{0.171})$	49	1.000
$CRF^{OOB}_{logLasso}$	$(X_{240} > \mathscr{Q}_{0.534}) \wedge (X_{437} \leq \mathscr{Q}_{0.374}) \wedge (X_{488} > \mathscr{Q}_{0.823})$	50	0.595
$CRF_{logLasso}$	$(X_{240} > \mathscr{Q}_{0.560}) \wedge (X_{437} \leq \mathscr{Q}_{0.275}) \wedge (X_{488} > \mathscr{Q}_{0.823})$	50	0.676

The best results are highlighted in bold.

in Chapter 5.3.1 has introduced the patterns which characterize FT_1 and FT_2, in particular, which variables form the conditions of the corresponding rules. Hence, first of all, it is interesting to see which of these failure related variables are contained in the antecedents of the rules that are generated by each of the studied rule learning variant, if these are applied to the entire set of available variables. In other words, no prior feature selection is conducted.

Table 5.7 lists both all failure related attributes and all analysed rule learning algorithms. Therein, the symbol "✓" shows for each method which variable is contained in any of its created rules that predict a vehicle as a faulty one.

With exception of the methods *RIPPER*, *C5.0rules*, and $CRF^{OOB}_{logLasso}$, all the other studied algorithms create rule conditions that account for all variables that describe the failure pattern of FT_1 and FT_2, respectively. However, it has to be noted that these conditions are not necessarily pooled together to form the antecedent of a single rule. Furthermore, this comes at a cost of producing larger rule sets in most cases, which may also contain a couple of uninformative rules. Since the RF based methods learn rules both from the entire feature space and from all the subspaces that are given by the variables which build

Table 5.9: Rule R that maximises $conf_{FT_2}(R)$ per studied rule learning method if the entire variable set is exploited

Method	Antecedent of best rule R for FT_2	$n(B \cap \mathrm{FT}_2)$	$conf_{FT_2}(R)$
RIPPER	$(X_{431} \geq \mathcal{Q}_{0.261}) \wedge (X_{432} \geq \mathcal{Q}_{0.848})$	22	1.000
RF^{OOB}_{RIPPER}	$(X_{431} \geq \mathcal{Q}_{0.180}) \wedge (X_{432} \geq \mathcal{Q}_{0.848})$	**24**	**1.000**
RF_{RIPPER}	$(X_{431} \geq \mathcal{Q}_{0.180}) \wedge (X_{432} \geq \mathcal{Q}_{0.848})$	**24**	**1.000**
C5.0rules	$(X_{431} > \mathcal{Q}_{0.117}) \wedge (X_{432} > \mathcal{Q}_{0.848}) \wedge (X_{540} > \mathcal{Q}_{0.849})$	25	0.581
$RF^{OOB}_{C5.0rules}$	$(X_{431} > \mathcal{Q}_{0.164}) \wedge (X_{432} > \mathcal{Q}_{0.848})$	**24**	**1.000**
$RF_{C5.0rules}$	$(X_{431} > \mathcal{Q}_{0.163}) \wedge (X_{432} > \mathcal{Q}_{0.848})$	**24**	**1.000**
$CRF^{OOB}_{linEnet}$	$(X_{431} > \mathcal{Q}_{0.167}) \wedge (X_{432} > \mathcal{Q}_{0.846})$	**24**	**1.000**
$CRF_{linEnet}$	$(X_{431} > \mathcal{Q}_{0.167}) \wedge (X_{432} > \mathcal{Q}_{0.846})$	**24**	**1.000**
$CRF^{OOB}_{linLasso}$	$(X_{431} > \mathcal{Q}_{0.167}) \wedge (X_{432} > \mathcal{Q}_{0.846})$	**24**	**1.000**
$CRF_{linLasso}$	$(X_{431} > \mathcal{Q}_{0.167}) \wedge (X_{432} > \mathcal{Q}_{0.846})$	**24**	**1.000**
$CRF^{OOB}_{logEnet}$	$(X_{431} > \mathcal{Q}_{0.172}) \wedge (X_{432} > \mathcal{Q}_{0.846})$	**24**	**1.000**
$CRF_{logEnet}$	$(X_{431} > \mathcal{Q}_{0.172}) \wedge (X_{432} > \mathcal{Q}_{0.846})$	**24**	**1.000**
$CRF^{OOB}_{logLasso}$	$(X_{431} > \mathcal{Q}_{0.167}) \wedge (X_{432} > \mathcal{Q}_{0.846})$	**24**	**1.000**
$CRF_{logLasso}$	$(X_{431} > \mathcal{Q}_{0.167}) \wedge (X_{432} > \mathcal{Q}_{0.846})$	**24**	**1.000**

The best results are highlighted in bold.

the split points in each of the 300 trees of the underlying RF model, this result is not surprising. Moreover, these methods as well as the CRF variants include all learned rules in their final rule sets rather than returning only those that are computed by an individual *RIPPER* or *C5.0rules* model. Thus, the only real surprise is that $CRF^{OOB}_{logLasso}$ filters variable X_{819} out completely.

Next, it is analysed for FT_1 and FT_2 individually which rule learning algorithm is able to create two rules $R : Body \rightarrow FT$, with $FT \in \{FT_1, FT_2\}$, that describe each of these two failure types best. Thereby, a rule R is regarded as being optimal within a set of rules $\mathcal{R}$, if it fulfils several requirements.

First of all, it has to achieve the highest confidence value, because it is aimed at generating rules that are significant for failures. However, since rules that cover only very few faulty vehicles are too specific in general, optimal rules have to apply to at least three vehicles that failed, in this study. Moreover, if two rules achieve exactly the same confidence value, then the rule is chosen that covers more faulty vehicles, because it generalizes better. If there is still no unique solution, then the shorter rule is selected, i.e., the one having less conditions in its antecedent, because it is easier to understand. Afterwards, if there are still more than one optimal rules, then, for simplicity, the rule is

Table 5.10: Optimal determined parameters and corresponding CV$_{out}$-BAC value, which is achieved on the *outer* stratified 5-fold CV for *rf* and rf_{Gini}, i.e., *before* and *after* applying feature selection.

	Parameter				
Method	m_{try}	$cutoff_{+1}$	$sampsize_{-1}$	**# variables used**	**CV$_{out}$-BAC**
rf	750	0.4	2	823	0.988
rf_{Gini}	10	0.4	2	58	0.998

The numbers ℓ in the column $sampsize_{-1}$ indicate that $sampsize_{-1}$ is set to $\ell \cdot sampsize_{+1}$, whereas n_{tree} and $cutoff_{-1}$ are fixed to 300 and $1 - cutoff_{+1}$.

picked that has been added first to the final rule set $\mathscr{R}$. In summary, it is evaluated which of the studied rule learning algorithms achieve the most "pure" rule that applies to the highest number of faulty vehicles at the same time.

Table 5.8 shows for each algorithm the antecedent of its best rule R for predicting FT_1, the number $n\left(B \cap \mathrm{FT}_1\right)$ of faulty vehicles that are covered by R, and the confidence value $conf_{FT_1}(R)$ that is obtained by R. The best results, i.e., the statistics of the rules that achieve the maximum confidence values, while also applying to the highest number of vehicles that suffer from a failure are highlighted in bold. Here, the methods *RIPPER*, RF_{RIPPER}, $RF^{OOB}_{C5.0rules}$, and $RF_{C5.0rules}$ are able to create a rule that covers exclusively all 50 vehicles that suffer from FT_1. Thus, they outperform the other algorithms in that case. Thereby, the antecedent of each of these best performing rules is formed by conditions on three of the five variables which describe the pattern of FT_1.

Furthermore, almost all of the remaining approaches create also rules which cover at least 49 of the 50 vehicles that failed because of FT_1, but they are less accurate. Moreover, it is interesting to see that the antecedents of almost all best rules are formed exclusively by conditions on failure relevant variables. The only exception is the rule antecedent produced by $CRF_{linEnet}$, which also includes constraints on other variables. However, since it creates a "pure" rule that applies only to three faulty vehicles, it is among the worst performing approaches for predicting FT_1, anyway.

Table 5.9 presents the analogue results for FT_2. It exhibits that 12 of the 14 studied approaches are able to create a rule that applies to 24 of the 25 vehicles

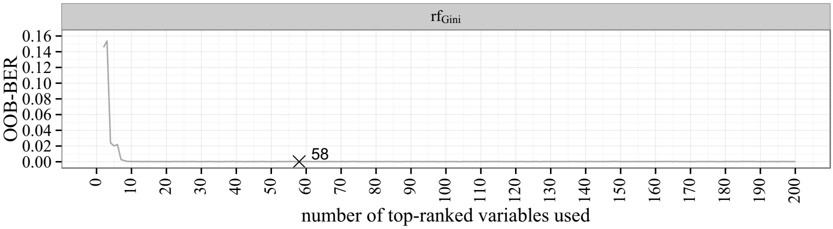


Figure 5.3: The minimal OOB-BER values achieved by rf_{Gini} in dependence on the number of top-ranked variables used

that suffer from FT_2 with a confidence value of one. While all of them contain a condition setting a lower bound for variable 432 that is close to the true one, they also produce a constraint on variable 431 that is not part of the true, artificially created pattern of FT_2. However, since the synthetic data is derived from real-world dataset 2a), it is an unintended, random characteristic of these synthetic, faulty cars that they do not have low observed values in attribute 431. The rule-based classification model *C5.0rules* performs worst in that case. A reason for that may be that only the rules for predicting failed vehicles are evaluated, but a good classification result, which is the main objective of this method, is in general also achieved by learning significant rules for the “healthy” cars.

In summary, methods RF_{RIPPER}, $RF^{OOB}_{C5.0rules}$, and $RF_{C5.0rules}$ perform best on average on the synthetic dataset, if the entire variable space is exploited. They are the only two methods whose best rules for FT_1 and FT_2, respectively, allow to identify in combination 74 out of the 75 faulty vehicles in the synthetic dataset. Thereby, none of the “healthy” cars is covered by any of these two rules. Moreover, the rule-based classification method *RIPPER* is among the well performing approaches, while its competitor *C5.0rules* is among the worst ones.

Next, it is analysed whether the results obtained by the studied rule learning methods may be improved through a prior feature selection. More precisely, it is checked if the results may be improved if these algorithms are only applied to the variables that are selected by the best performing classification model of Chapter 3, namely rf_{Gini}.

The results that are achieved by rf_{Gini} on the synthetic dataset are shown in Table 5.10. It reduces the number of variables from 823 to 58, while improving

Table 5.11: Identified variables of corresponding failure type per rule learner if only the 58 variables of the synthetic dataset are used that are selected by rf_{Gini}

	Failure Type 1 (FT_1)					**Failure Type 2 (FT_2)**		
Method	X_{240}	X_{407}	X_{437}	X_{488}	X_{495}	X_{432}	X_{540}	X_{819}
RIPPER	✓			✓	✓	✓	✓	✓
RF^{OOB}_{RIPPER}	✓	✓	✓	✓	✓	✓	✓	✓
RF_{RIPPER}	✓	✓	✓	✓	✓	✓	✓	✓
C5.0rules	✓	✓	✓	✓	✓	✓		✓
$RF^{OOB}_{C5.0rules}$	✓	✓	✓	✓	✓	✓	✓	✓
$RF_{C5.0rules}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF^{OOB}_{linEnet}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF_{linEnet}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF^{OOB}_{linLasso}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF_{linLasso}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF^{OOB}_{logEnet}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF_{logEnet}$	✓	✓	✓	✓	✓	✓	✓	✓
$CRF^{OOB}_{logLasso}$	✓	✓	✓	✓	✓	✓	✓	
$CRF_{logLasso}$	✓	✓	✓	✓	✓	✓	✓	✓

the cross-validated BAC value from 0.988 to 0.998. Furthermore, it is interesting to see that the most important 8 variables according to this approach are the 8 variables describing the patterns of FT_1 and FT_2, i.e., X_{488}, X_{240}, X_{407}, X_{432}, X_{437}, X_{495}, X_{540}, and X_{819}.

On the other hand, rf_{Gini} finally selects 58 variables, i.e., further 50 attributes that have not been manipulated during the creation process of FT_1 and FT_2. Figure 5.3 reveals that the "elbow", which is intended to be found in the graph "minimal OOB-BER value against number of top-ranked variables used", is not approximated well for this dataset. Otherwise, only the above mentioned 8 top-ranked variables should have been selected. The reason for this bad approximation is that the tolerance criterion, given in Equation 3.49, is not working well if the OOB-BER decreases to values that are almost zero. Thus, it may be improved in a future work by extending it with an additional condition on the allowed absolute deviance from the minimal OOB-BER value.

As before, Table 5.11 shows which rule learning methods create rules that incorporate the five and three variables that are relevant for FT_1 and FT_2, re-

Table 5.12: Rule R that maximises $conf_{FT_1}(R)$ per studied rule learner if only those 58 variables of the synthetic dataset are exploited that are selected by rf_{Gini}

Method	Antecedent of best rule R for FT_1	$n(B \cap \mathrm{FT}_1)$	$conf_{FT_1}(R)$
$RIPPER$	$(X_{240} \geq \mathcal{Q}_{0.845}) \wedge (X_{488} \geq \mathcal{Q}_{0.846}) \wedge (X_{495} \leq \mathcal{Q}_{0.155})$	**50**	***1.000**
RF^{OOB}_{RIPPER}	$(X_{240} \geq \mathcal{Q}_{0.845}) \wedge (X_{488} \geq \mathcal{Q}_{0.824}) \wedge (X_{495} \leq \mathcal{Q}_{0.155})$	**50**	***1.000**
RF_{RIPPER}	$(X_{240} \geq \mathcal{Q}_{0.845}) \wedge (X_{488} \geq \mathcal{Q}_{0.824}) \wedge (X_{495} \leq \mathcal{Q}_{0.155})$	**50**	***1.000**
$C5.0rules$	$(X_{426} > \mathcal{Q}_{0.157}) \wedge (X_{432} \leq \mathcal{Q}_{0.848}) \wedge (X_{437} \leq \mathcal{Q}_{0.155})$	39	*1.000
$RF^{OOB}_{C5.0rules}$	$(X_{240} > \mathcal{Q}_{0.844}) \wedge (X_{488} > \mathcal{Q}_{0.846}) \wedge (X_{495} \leq \mathcal{Q}_{0.171})$	**50**	***1.000**
$RF_{C5.0rules}$	$(X_{240} > \mathcal{Q}_{0.844}) \wedge (X_{488} > \mathcal{Q}_{0.846}) \wedge (X_{495} \leq \mathcal{Q}_{0.171})$	**50**	***1.000**
$CRF^{OOB}_{linEnet}$	$(X_{432} \leq \mathcal{Q}_{0.760}) \wedge (X_{437} \leq \mathcal{Q}_{0.157}) \wedge (X_{476} > \mathcal{Q}_{0.158})$	35	*1.000
$CRF_{linEnet}$	$(X_{432} \leq \mathcal{Q}_{0.760}) \wedge (X_{437} \leq \mathcal{Q}_{0.157}) \wedge (X_{476} > \mathcal{Q}_{0.158})$	35	*1.000
$CRF^{OOB}_{linLasso}$	$(X_{437} \leq \mathcal{Q}_{0.163}) \wedge (X_{488} > \mathcal{Q}_{0.741}) \wedge (X_{495} \leq \mathcal{Q}_{0.162})$	50	*0.926
$CRF_{linLasso}$	$(X_{488} > \mathcal{Q}_{0.842}) \wedge (X_{495} \leq \mathcal{Q}_{0.170}) \wedge (X_{540} \leq \mathcal{Q}_{0.832})$	41	0.891
$CRF^{OOB}_{logEnet}$	$(X_{432} \leq \mathcal{Q}_{0.760}) \wedge (X_{437} \leq \mathcal{Q}_{0.157}) \wedge (X_{476} > \mathcal{Q}_{0.158})$	35	*1.000
$CRF_{logEnet}$	$(X_{432} \leq \mathcal{Q}_{0.760}) \wedge (X_{437} \leq \mathcal{Q}_{0.157}) \wedge (X_{476} > \mathcal{Q}_{0.158})$	35	1.000
$CRF^{OOB}_{logLasso}$	$(X_{240} > \mathcal{Q}_{0.608}) \wedge (X_{488} > \mathcal{Q}_{0.852}) \wedge (X_{488} \leq \mathcal{Q}_{0.904})$	50	0.543
$CRF_{logLasso}$	$(X_{437} \leq \mathcal{Q}_{0.156}) \wedge (X_{488} > \mathcal{Q}_{0.809}) \wedge (X_{488} \leq \mathcal{Q}_{0.931}) \wedge (X_{495} \leq \mathcal{Q}_{0.520})$	50	*0.746

The best results are highlighted in bold. Improved performance values are marked with *.

spectively. It reveals that the prior applied feature selection is beneficial for almost all approaches, because it enables the majority of the algorithms to generate rules that cover at least the same number of relevant variables, as before.

Table 5.12 shows again the best rules for predicting FT_1 per method. However, this time only the 58 attributes that are finally selected by rf_{Gini} and the final RF model that is created by the latter approach are exploited by the studied algorithms, respectively. Moreover, if the prior application of rf_{Gini} results in an improvement of the confidence value, then the corresponding computed values for $conf_{FT_1}(R)$ are marked with an asterisk. Also, if the formerly achieved confidence value can be preserved by the new rule R, while covering at least as many of the failed vehicles, as before, then the corresponding new value of $conf_{FT_1}(R)$ is marked with an asterisk. Additionally, best results are again highlighted in bold.

After applying rf_{Gini}, 11 of the 14 studied methods create a rule that achieves an at least as high confidence value as the best rule that has been created without feature selection. Now, there are five algorithms that generate a per-

Table 5.13: Rule R that maximises $conf_{FT_2}(R)$ per studied rule learner if only those 58 variables of the synthetic dataset are exploited that are selected by rf_{Gini}

Method	Antecedent of best rule R for FT_2	$n(B \cap \mathrm{FT}_2)$	$conf_{FT_2}(R)$
$RIPPER$	$(X_{13} \geq \mathscr{Q}_{0.178}) \wedge (X_{590} \geq \mathscr{Q}_{0.156}) \wedge (X_{819} \leq \mathscr{Q}_{0.152})$	20	1.000
RF^{OOB}_{RIPPER}	$(X_{431} \geq \mathscr{Q}_{0.180}) \wedge (X_{432} \geq \mathscr{Q}_{0.848})$	**24**	***1.000**
RF_{RIPPER}	$(X_{431} \geq \mathscr{Q}_{0.180}) \wedge (X_{432} \geq \mathscr{Q}_{0.848})$	**24**	***1.000**
$C5.0rules$	$(X_{431} > \mathscr{Q}_{0.164}) \wedge (X_{432} > \mathscr{Q}_{0.848})$	**24**	***1.000**
$RF^{OOB}_{C5.0rules}$	$(X_{431} > \mathscr{Q}_{0.164}) \wedge (X_{432} > \mathscr{Q}_{0.848})$	**24**	***1.000**
$RF_{C5.0rules}$	$(X_{431} > \mathscr{Q}_{0.164}) \wedge (X_{432} > \mathscr{Q}_{0.848})$	**24**	***1.000**
$CRF^{OOB}_{linEnet}$	$(X_{321} > \mathscr{Q}_{0.571}) \wedge (X_{532} \leq \mathscr{Q}_{0.703}) \wedge (X_{540} > \mathscr{Q}_{0.852})$	4	0.800
$CRF_{linEnet}$	$(X_{431} > \mathscr{Q}_{0.160}) \wedge (X_{432} > \mathscr{Q}_{0.846})$	**24**	***1.000**
$CRF^{OOB}_{linLasso}$	$(X_{432} > \mathscr{Q}_{0.846}) \wedge (X_{532} \leq \mathscr{Q}_{0.814}) \wedge (X_{540} > \mathscr{Q}_{0.871})$	13	1.000
$CRF_{linLasso}$	$(X_{451} > \mathscr{Q}_{0.213}) \wedge (X_{819} \leq \mathscr{Q}_{0.157})$	19	0.679
$CRF^{OOB}_{logEnet}$	$(X_{431} > \mathscr{Q}_{0.160}) \wedge (X_{432} > \mathscr{Q}_{0.848})$	**24**	***1.000**
$CRF_{logEnet}$	$(X_{431} > \mathscr{Q}_{0.160}) \wedge (X_{432} > \mathscr{Q}_{0.848})$	**24**	***1.000**
$CRF^{OOB}_{logLasso}$	$(X_{432} > \mathscr{Q}_{0.845}) \wedge (X_{432} \leq \mathscr{Q}_{0.907}) \wedge (X_{488} > \mathscr{Q}_{0.837})$	25	0.325
$CRF_{logLasso}$	$(X_{437} > \mathscr{Q}_{0.391}) \wedge (X_{540} > \mathscr{Q}_{0.830}) \wedge (X_{819} \leq \mathscr{Q}_{0.156})$	12	0.667

The best results are highlighted in bold. Improved performance values are marked with *.

fect rule for predicting FT_1, while there only were four, beforehand. However, some methods, like $CRF_{linEnet}$, now extract a rule with a higher confidence value, but at the cost of covering less faulty vehicles than before. Thus, comparing the results before and after feature selection has to be done with caution.

In general, the advantage of a prior feature selection does not seem to be as big for the CRF variants as for the other algorithms. However, since CRF internally applies feature selection on its own, there is no urgent need to additionally select features externally.

Finally, Table 5.13 presents the returned rules for predicting FT_2, if rf_{Gini} is applied first again. In contrast to the results that are obtained for FT_1, the performance of the models, measured by $conf_{FT_2}(R)$, is only improved for eight of the studied rule learning algorithms. Here, there are also less methods that are able to achieve the best result of creating a rule that covers 24 of the 25 vehicles suffering from FT_2 with confidence one.

In summary, the newly proposed variants RF^{OOB}_{RIPPER}, RF_{RIPPER}, $RF^{OOB}_{C5.0rules}$, and $RF_{C5.0rules}$, respectively, perform best on average, after applying the described feature selection strategy. Thus, selecting features before learning any rules seem to be the adequate strategy for these kind of algorithms. For their CRF based competitors, no clear statement about the benefit of applying rf_{Gini} first can be made.

Moreover, both before and after feature selection, the rule-based classification method *RIPPER* is competitive with, but slightly worse than the newly proposed RF based approaches on the studied dataset, while *C5.0rules* performs worse.

Analysis of dataset 2a)

In order to study the applicability of the discussed rule learning methods to real-world problems, the analysis of the previous subchapter is repeated for dataset 2a), which contains 47 vehicles suffering from a particular failure of the hybrid car battery. In the first step, rules for predicting this failure are learned from the entire dataset 2a) again, i.e., all available variables are exploited by each of the studied algorithms. Afterwards, in the second step, only the 50 variables that have been selected by rf_{Gini} in the case study, performed in Chapter 3, are used as input for the algorithms. Moreover, the RF and CRF variants start with the exploration of the final RF model that is produced by rf_{Gini}.

Table 5.14 presents the results of the first part of the analysis, while using the same table structure, as in the previous study. It is notable that there are only four approaches that are able to create rules that apply to the failed cars with a confidence value of one. Thereby, the three newly proposed methods RF_{RIPPER}, RF^{OOB}_{RIPPER}, and $RF_{C5.0rules}$ perform best, because they do not only create “pure” rules for the failed class, but also cover 27 of the 47 instances belonging to the latter category. It is also interesting that, according to the rule conditions, the majority of the failed vehicles that are covered by their rules have either comparatively pretty low or high observed values in the restricted variables, if compared to the entire vehicle fleet that consists of 8131 HEV. (The only exception is condition $(X_{135} \leq \mathscr{Q}_{0.920})$ in the antecedent of the rule built by $RF_{C5.0rules}$). This is an indicator that the usage of the covered failed vehicles or at least of some components of their hybrid power-trains is different from the common one. Furthermore, the prior application of the newly proposed method rf_{Gini} to select important variables before applying any rule learning method to dataset 2a), has a positive influence on the confidence value

Table 5.14: Rule R that maximises $conf_{FT}\,(R)$ per studied rule learner if the entire real-world dataset dataset 2a) is exploited

Method	Antecedent of best rule R for FT	$n\,(B \cap \mathrm{FT})$	$conf_{FT}\,(R)$
RIPPER	$(X_8 \leq \mathscr{Q}_{0.064}) \wedge (X_{132} \geq \mathscr{Q}_{0.930}) \wedge (X_{509} \geq \mathscr{Q}_{0.982})$	32	0.842
RF^{OOB}_{RIPPER}	$(X_8 \leq \mathscr{Q}_{0.048}) \wedge (X_{132} \geq \mathscr{Q}_{0.948}) \wedge (X_{509} \geq \mathscr{Q}_{0.974})$	**27**	**1.000**
RF_{RIPPER}	$(X_8 \leq \mathscr{Q}_{0.048}) \wedge (X_{132} \geq \mathscr{Q}_{0.948}) \wedge (X_{509} \geq \mathscr{Q}_{0.974})$	**27**	**1.000**
C5.0rules	$(X_{509} \geq \mathscr{Q}_{0.989})$	33	0.375
$RF^{OOB}_{C5.0rules}$	$(X_1 > \mathscr{Q}_{0.930}) \wedge (X_{97} \geq \mathscr{Q}_{0.988})$	6	1.000
$RF_{C5.0rules}$	$(X_8 \leq \mathscr{Q}_{0.041}) \wedge (X_{135} \leq \mathscr{Q}_{0.920}) \wedge (X_{509} > \mathscr{Q}_{0.982})$	**27**	**1.000**
$CRF^{OOB}_{linEnet}$	$(X_8 \leq \mathscr{Q}_{0.132}) \wedge (X_{25} > \mathscr{Q}_{0.743}) \wedge (X_{509} > \mathscr{Q}_{0.966})$	40	0.520
$CRF_{linEnet}$	$(X_{99} > \mathscr{Q}_{0.979}) \wedge (X_{820} > \mathscr{Q}_{0.978}) \wedge (X_{860} > \mathscr{Q}_{0.802})$	4	0.667
$CRF^{OOB}_{linLasso}$	$(X_8 \leq \mathscr{Q}_{0.067}) \wedge (X_{25} > \mathscr{Q}_{0.745}) \wedge (X_{134} > \mathscr{Q}_{0.850})$	36	0.563
$CRF_{linLasso}$	$(X_8 \leq \mathscr{Q}_{0.016}) \wedge (X_{509} > \mathscr{Q}_{0.974})$	18	0.818
$CRF^{OOB}_{logEnet}$	$(X_{97} > \mathscr{Q}_{0.996}) \wedge (X_{262} > \mathscr{Q}_{0.445}) \wedge (X_{838} \leq \mathscr{Q}_{0.087})$	3	0.750
$CRF_{logEnet}$	$(X_{99} \leq \mathscr{Q}_{0.977}) \wedge (X_{132} > \mathscr{Q}_{0.984}) \wedge (X_{509} > \mathscr{Q}_{0.969})$	3	0.750
$CRF^{OOB}_{logLasso}$	$(X_{51} > \mathscr{Q}_{0.484}) \wedge (X_{99} > \mathscr{Q}_{0.995}) \wedge (X_{135} \leq \mathscr{Q}_{0.920}) \wedge (X_{509} > \mathscr{Q}_{0.967}) \wedge (X_{545} > \mathscr{Q}_{0.124})$	3	0.750
$CRF_{logLasso}$	$(X_8 \leq \mathscr{Q}_{0.082}) \wedge (X_{158} > \mathscr{Q}_{0.495}) \wedge (X_{410} \leq \mathscr{Q}_{0.450}) \wedge (X_{508} \leq \mathscr{Q}_{0.016})$	19	0.704

The best results are highlighted in bold.

of the best extracted rules for the majority of the studied algorithms, as shown in Table 5.15. Like in the previous analysis, the new RF based methods outperform the other approaches, whereat the two variants employing *C5.0rules* as rule learner perform best on this dataset. Reducing the feature set first, allows each of these two algorithms to create a rule that covers three more failed vehicles than the one generated before. In particular, the rule-based classification methods *RIPPER* and *C5.0rules* are clearly outperformed on this dataset by the two mentioned, new approaches.

Finally, the two distinct, best rules that are created by $RF_{C5.0rules}$, $RF^{OOB}_{C5.0rules}$, and the slightly worse performing approach RF_{RIPPER} are studied, in detail.

The best rule covers 30 failed vehicles and its antecedent is formed by five conditions, with two of them being rather weak. Ignoring the latter two ones, the following conditions remain:

$$(X_8 \leq \mathscr{Q}_{0.048}), (X_{72} > \mathscr{Q}_{0.926}), \text{ and } (X_{73} > \mathscr{Q}_{0.824}).$$

Table 5.15: Rule R that maximises $conf_{FT}(R)$ per studied rule learner, if only those 50 variables of the real-world dataset 2a) are exploited that are selected by rf_{Gini}

Method	Antecedent of best rule R for FT	$n(B \cap \mathrm{FT})$	$conf_{FT}(R)$
RIPPER	$(X_8 \le \mathscr{Q}_{0.048}) \wedge (X_{97} \ge \mathscr{Q}_{0.988})$	23	*0.920
RF^{OOB}_{RIPPER}	$(X_8 \le \mathscr{Q}_{0.048}) \wedge (X_{132} \ge \mathscr{Q}_{0.948}) \wedge (X_{509} \ge \mathscr{Q}_{0.974})$	27	*1.000
RF_{RIPPER}	$(X_8 \le \mathscr{Q}_{0.064}) \wedge (X_{100} \le \mathscr{Q}_{0.982}) \wedge (X_{131} \ge \mathscr{Q}_{0.942}) \wedge$ $(X_{318} \ge \mathscr{Q}_{0.585}) \wedge (X_{835} \ge \mathscr{Q}_{0.950})$	28	*1.000
C5.0rules	$(X_{509} > \mathscr{Q}_{0.989})$	33	*0.375
$RF^{OOB}_{C5.0rules}$	$(X_8 \le \mathscr{Q}_{0.048}) \wedge (X_{72} > \mathscr{Q}_{0.926}) \wedge (X_{73} > \mathscr{Q}_{0.824}) \wedge$ $(X_{100} \le \mathscr{Q}_{0.992}) \wedge (X_{349} > \mathscr{Q}_{0.392})$	**30**	***1.000**
$RF_{C5.0rules}$	$(X_8 \le \mathscr{Q}_{0.048}) \wedge (X_{72} > \mathscr{Q}_{0.926}) \wedge (X_{73} > \mathscr{Q}_{0.824}) \wedge$ $(X_{100} \le \mathscr{Q}_{0.992}) \wedge (X_{349} > \mathscr{Q}_{0.392})$	**30**	***1.000**
$CRF^{OOB}_{linEnet}$	$(X_{99} > \mathscr{Q}_{0.977}) \wedge (X_{378} \le \mathscr{Q}_{0.566}) \wedge (X_{687} \le \mathscr{Q}_{0.020})$	26	*0.703
$CRF_{linEnet}$	$(X_{508} \le \mathscr{Q}_{0.036}) \wedge (X_{877} > \mathscr{Q}_{0.993})$	3	*0.750
$CRF^{OOB}_{linLasso}$	$(X_8 \le \mathscr{Q}_{0.128}) \wedge (X_{99} > \mathscr{Q}_{0.977})$	37	*0.569
$CRF_{linLasso}$	$(X_8 \le \mathscr{Q}_{0.128}) \wedge (X_{99} > \mathscr{Q}_{0.977})$	37	0.569
$CRF^{OOB}_{logEnet}$	$(X_{97} \le \mathscr{Q}_{0.987}) \wedge (X_{98} > \mathscr{Q}_{0.983}) \wedge (X_{132} > \mathscr{Q}_{0.980})$	3	*1.000
$CRF_{logEnet}$	$(X_{378} \le \mathscr{Q}_{0.094}) \wedge (X_{509} > \mathscr{Q}_{0.972})$	18	0.720
$CRF^{OOB}_{logLasso}$	$(X_{378} \le \mathscr{Q}_{0.094}) \wedge (X_{509} > \mathscr{Q}_{0.972})$	18	0.720
$CRF_{logLasso}$	$(X_{378} \le \mathscr{Q}_{0.094}) \wedge (X_{509} > \mathscr{Q}_{0.972})$	18	*0.720

The best results are highlighted in bold. Improved performance values are marked with *.

After having a look at Table 3.12 again, it becomes evident that a characteristic of many failed vehicles in dataset 2a) is that the cumulated idle time of their hybrid car batteries, which is stored in variable X_8, is short, compared to those of the remaining cars of the entire vehicle fleet. Furthermore, the condition on variables X_{72} and X_{73} point out that these cars are driven in such a way that the percentage of the total number of rainflow cycles that run through very low SoC levels of the hybrid car battery is higher than those of the vehicles that are not covered by this rule.

This observation is backed up by the conditions that form the antecedent of the second best rule. After neglecting weak restrictions again, the following conditions survive:

$$(X_8 \le \mathscr{Q}_{0.064}), (X_{131} \ge \mathscr{Q}_{0.942}), \text{ and } (X_{835} \ge \mathscr{Q}_{0.950}).$$

Hence, not only the mentioned properties with respect to the total idle time and the SoC of the hybrid car battery are typical for the failed cars in dataset 2a), but these vehicles also tend to have mainly short parking times, referring to the meaning of variable X_{835}. In particular, this coincides with the observed short idle times of the batteries.

Due to confidentiality reasons, it is not possible to provide more details about the failure of the hybrid car battery of these vehicles. However, it is notable that on the basis of this autonomously extracted pattern, a battery expert has been able to infer the true reason for the considered failure type.

5.4 Conclusion

In this chapter the applicability of several rule learning methods has been studied with the goal to identify stress patterns in load spectrum data of large HEV fleets that are inherent to a particular group of interest, e.g., to vehicles suffering from a failure of a particular component of the hybrid power-train.

For this purpose, the considered approaches have been used to analyse a synthetic dataset first, which contains vehicles that suffer from two distinct, artificially created failure types. It has been demonstrated empirically that especially the newly proposed methods $CRF_{C5.0rules}$, $CRF^{OOB}_{C5.0rules}$ and CRF_{RIPPER} are able to detect the important characteristics of the stress patterns describing these two failure types. Afterwards, similar results have been achieved for a real-world dataset that contains vehicles that suffer from a specific failure type of the hybrid car battery. Also in that case, the mentioned approaches have been able to learn patterns that provide important information about the reasons for the considered component failure.

Moreover, it has been shown empirically that it is often beneficial to first apply the classification and feature selection approach rf_{Gini}, which has been newly proposed in Chapter 3. Thereby, the feature space may be reduced heavily first, before the discussed algorithms may be applied to learn rules from the data.

As a conclusion, rule learning methods may support engineers to gain knowledge about the nature of stress patterns that are related to particular component failures or that are inherent to a usage cluster that has been detected with the visualization techniques, provided in Chapter 4. Hence, this chapter has shown ways to generate interpretable information on the basis of the results that are achieved by the algorithms, discussed in Chapter 3 and 4, respectively.

6 Conclusion

This thesis addressed the problem of analysing a huge amount of a load spectrum data, i.e., a special kind of automotive data that are recorded and computed on-board in modern vehicles such as HEVs. The aim has been manifold, where the main goal has been to determine usage and stress patterns that are related to failures of selected components of the hybrid power-train, like the hybrid car battery. The identified patterns can help the engineers to find out the reasons for component failures and, thus, to improve the dimensioning as well as the reliability of future versions of these vehicle parts.

For this purpose, rule learning algorithms from the field of Machine Learning have been proposed and evaluated on both artificially created and real-world load spectrum datasets. Moreover, a random forest based classification and feature selection approach has been developed that helps to reduce the dimensionality of these datasets heavily. Thus, it allows to localise load spectrum classes that are related to the considered component failures, on the one hand. On the other hand, this technique allows to check if a vehicle is stressed similarly to vehicles that suffer from a failure of a particular component.

Since the components of a power-train may fail due to distinct types of defect, where the information about these different types is often not provided by the workshops, a visualisation technique has been developed additionally that may support the decision process for determining whether a particular component of two vehicles failed because of the same error type or not. Moreover, this newly proposed algorithm can help to identify heterogeneous kinds of usage clusters within a large vehicle fleet.

6.1 Main contributions

This thesis made the following main contributions:

- The applicability of several state-of-the-art classification algorithms to distinguish between non-faulty vehicles and cars that suffer from a failure of a particular component of the hybrid power-train has been investigated, when

P. Bergmeir, *Enhanced Machine Learning and Data Mining Methods for Analysing Large Hybrid Electric Vehicle Fleets based on Load Spectrum Data*, Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart, https://doi.org/10.1007/978-3-658-20367-2_6

these methods are applied exclusively to load spectrum data, which have been recorded for huge HEV fleets (Chapter 3).

- A new Random Forest based feature selection and classification technique has been proposed that helps to improve the above mentioned classification results as well as to reduce the dimensionality of the studied datasets heavily. Thus, it also allows to identify load spectrum classes that may be related to component failures (Chapter 3).
- A Data Mining system that is based on the dimensionality reduction technique t-SNE is developed that allows to automatically identify and to visualize different types of vehicle usage and stress by exploiting the load spectrum data of large HEV fleets (Chapter 4).
- The applicability of rule learning methods for identifying and describing usage and stress patterns that are related to component failures is assessed, whereat also new approaches are proposed that outperform the existing ones on the studied datasets (Chapter 5).

6.2 Limitations

The proposed classification approaches have been developed to work with load spectrum data of vehicles that suffer from a failure of a particular component of the power-train and of cars where this problem has not occurred, yet. Thereby, the information about the exact failure time has not been incorporated in the models. Thus, these algorithms have not been designed for forecasting at what point in time the considered failure is very likely to occur. However, using so-called *random survival forests* [56], instead of standard RF classification methods, may be a possible enhancement of the discussed RF approaches to transform them into models that are able to predict the remaining useful life of a component. First attempts to predict the need for repairs of selected components have been undertaken in [40] and [95].

Furthermore, all discussed classifiers have been optimized and evaluated on load spectrum data with the goal to solve binary classification problems, i.e., to distinguish between vehicles with and those without a particular component failure. The reason for this was that there was no information about the different types of failures available from the workshops. If there will be more details about the error types available in the future, then the approaches have to be adapted to be able to handle the arising multi-class classification problems. These problems are far more complex, in general, because decision functions have to be determined that separate not only the instances from two, but from multiple distinct classes. Also the discussed performance measures have to be generalized to the multi-class case, as done in [109]. However, the good news is that there are approaches such as *one-vs.-rest* [124] that allow to transform multi-class problems into multiple binary classification problems. Moreover, in particular, the newly proposed RF based classification and feature selection technique rf_{Gini}, is applicable directly to multi-class problems, because the used variable importance measures as well as the RF algorithm itself are capable to handle more than two classes without problems.

Also the best performing rule learning methods can be extended naturally to multi-class problems, because *RIPPER*, *C5.0rules*, and RF models can be applied directly to multi-class data. The only small modification that has to be carried out is that rules for all failure types have to be returned then, i.e., only rules for the “healthy” cars can be further discarded.

Since the discussed dimensionality reduction and visualization techniques run totally in unsupervised mode, i.e., do not use any information about the class labels, they work in exactly the same way for multi-class datasets.

Finally, it has to be noted that computing the RF dissimilarity measure that is required by RF-t-SNE as well as the new RF based classification and feature selection framework are computationally burdensome. Hence, they scale not well for very big datasets, i.e., data that are recorded for tens of thousands of vehicles. However, this is regarded as being a minor limitation because it may be overcome by using the modern facilities of high-performance computing.

6.3 Benefits

As shown in Chapter 1.2, there are only few publications about applying Data Mining and Machine Learning methods to load spectrum data available, so far. Hence, this thesis did some pioneer work on analysing this special kind of automotive data with modern techniques from the mentioned research areas. Thereby, a major benefit of this work is that it demonstrates ways to gain useful information by exploiting several load spectra simultaneously. Thereby, it not only assesses the performance of a couple of state-of-the-art classification, dimensionality reduction, and rule learning techniques, but also enhances some of them to work better on the studied real-world load spectrum datasets.

Since this kind of data has been analysed mainly manually and separately for each load spectrum by human experts so far, the proposed approaches, in particular the new visualization and rule learning techniques, allow to get a better understanding of interacting components' stress patterns that are prevalent in complex power-trains, like those of HEVs. Therefore, it especially enhances the possibilities of studying customer field data.

Amongst others, it consequently allows engineers to get answers to the following questions:

- Are there different types of vehicle usage or stress within a vehicle fleet?
- Does the operating country have any influence on the kind of vehicle usage?
- Are vehicles that suffer from a failure of a component of the power-train stressed or driven in a special or abnormal way?
- Which loads may be related to these failures and what does a description of the harmful stress patterns look like?

Moreover, the newly proposed approaches may together build the basis of a future predictive maintenance system. Thereby, the enhanced dimensionality reduction and visualization technique RF-t-SNE may support the decision which components are endangered to fail because they are installed in vehicles that are stressed similarly to cars in which the considered elements have already failed. Additionally, this approach may help to identify if there are different stress patterns that are likely to provoke a failure of a particular component that is employed in each car of a large vehicle fleet. Thus, it can support the decision of which components of which customer cars have to be returned to the manufacturer to be examined thoroughly in a laboratory. In that way, lots of money may be saved because costly returns of many components, where the failure type is already known, may be avoided.

The proposed rule learning methods may be employed additionally in such a predictive maintenance and failure type identification system to provide interpretable descriptions of stress patterns that correlate with a certain failure type.

Finally, the developed classification and feature selection approaches may be applied prior to the mentioned techniques to reduce heavily the set of variables by filtering out irrelevant or noisy attributes such that only features remain that are probably related to component failures. Thus, the performance of these approaches is likely to be boosted, as demonstrated empirically by the case studies of Chapter 3, 4, and 5.

7 Outlook

At the end of this thesis, possible enhancements of the proposed approaches as well as future research directions are discussed briefly.

All the approaches that have been proposed in this work are only applied to and evaluated on load spectrum data. However, this kind of automotive data has some disadvantages, as discussed in Chapter 2. It does, for example, not store any information about the time of occurrence of the transformed signal values of interest. Moreover, its quality is also dependent on the data resolution, e.g., on the number of intervals and the definition of each of them.

Hence, in a future work, it may be beneficial to compensate for this lack of information by enriching load spectrum data with other kinds of data such as the error memory data [99] of the vehicles. This may improve the provided classification results and allow to learn even more significant rules for predicting failure types of certain components.

Furthermore, each individual class or interval of a load spectrum has been used as a separate input variable in this thesis. Thereby, the information that is given in each class description, like the interval specification, is not exploited by the proposed approaches. Thus, in a future analysis some statistics that approximate, e.g., the central tendency and the dispersion of the distribution that is underlying each load spectrum may be calculated. Among these measures are the mean and the variance and so on. This is partly done in [40], but only for load spectrum data resulting from one-parameter counting methods. Moreover, Frisk et al. do not determine higher-order statistics, like the skewness and kurtosis, for each load spectrum, which may provide additional useful information.

Finally, modern telematics services will transmit the recorded load spectrum data of each vehicle to databases, owned by the OEM, in regular time intervals in the future, e.g., every week or month. This will offer new possibilities for analysing this kind of data. For example, the variation of the amount of growth of each load spectrum over time, i.e., the absolute or relative differences between load spectra of consecutive transfer dates, may provide useful information. As a consequence, there will be an urgent need for methods that allow to analyse time-series of load spectra in the future.

P. Bergmeir, *Enhanced Machine Learning and Data Mining Methods for Analysing Large Hybrid Electric Vehicle Fleets based on Load Spectrum Data*, Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart, https://doi.org/10.1007/978-3-658-20367-2_7

Bibliography

[1] Akbani R, Kwek S, Japkowicz N (2004) Applying support vector machines to imbalanced datasets. In: Proceedings of the 15th European Conference on Machine Learning (ECML), pp 39–50

[2] Allen E, Horvath S, Tong F, Kraft P, Spiteri E, Riggs AD, Marahrens Y (2003) High concentrations of long interspersed nuclear element sequence distinguish monoallelically expressed genes. Proceedings of the National Academy of Sciences of the United States of America 100(17):9940–9945

[3] Ambroise C, McLachlan GJ (2002) Selection bias in gene extraction on the basis of microarray gene-expression data. Proceedings of the National Academy of Sciences of the United States of America 99(10):6562–6566

[4] Barros R, Cerri R, Jaskowiak P, de Carvalho A (2011) A Bottom-Up Oblique Decision Tree Induction Algorithm. In: Proceedings of 11th International Conference on Intelligent Systems Design and Applications (ISDA), pp 450–456

[5] Bergmeir P, Nitsche C, Nonnast J, Bargende M, Antony P, Keller U (2014) Klassifikationsverfahren zur Identifikation von Korrelationen zwischen Antriebsstrangbelastungen und Hybridkomponentenfehlern einer Hybridfahrzeugflotte. Tech. rep., Universität Stuttgart

[6] Bergmeir P, Nitsche C, Nonnast J, Bargende M, Antony P, Keller U (2014) Using Balanced Random Forests on Load Spectrum Data for Classifying Component Failures of a Hybrid Electric Vehicle Fleet. In: Proceedings of the 13th International Conference on Machine Learning and Applications (ICMLA), pp 397–404

[7] Bergmeir P, Nitsche C, Nonnast J, Bargende M (2015) Methoden des Data Mining zur Visualisierung unterschiedlicher Belastungsmuster einer Hybridfahrzeugflotte auf Basis von Lastkollektivdaten. Tech. rep., Universität Stuttgart

[8] Bergmeir P, Nitsche C, Nonnast J, Bargende M (2016) A Load Spectrum Data based Data Mining System for Identifying Different Types

P. Bergmeir, *Enhanced Machine Learning and Data Mining Methods for Analysing Large Hybrid Electric Vehicle Fleets based on Load Spectrum Data*, Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart, https://doi.org/10.1007/978-3-658-20367-2

of Vehicle Usage of a Hybrid Electric Vehicle Fleet. SAE International Journal of Alternative Powertrains 5(1):50–57

[9] Bergmeir P, Nitsche C, Nonnast J, Bargende M (2016) Classifying component failures of a hybrid electric vehicle fleet based on load spectrum data. Neural Computing and Applications 27:2289–2304

[10] Berthold M, Borgelt C, Höppner F, Klawonn F (2010) Guide to Intelligent Data Analysis. Springer London

[11] Bishop R (2005) Intelligent Vehicle Technology and Trends. Artech House ITS library, Artech House

[12] Blanz V, Schölkopf B, Bülthoff H, Burges C, Vapnik V, Vetter T (1996) Comparison of view-based object recognition algorithms using realistic 3D models. In: Proceedings of International Conference on Artificial Neural Networks (ICANN), pp 251–256

[13] Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2012) A review of feature selection methods on synthetic data. Knowledge and Information Systems 34(3):483–519

[14] Borgwardt K (2001) Optimierung, Operations Research, Spieltheorie: Mathematische Grundlagen. Birkhäuser Basel

[15] Boulesteix AL, Janitza S, Kruppa J, König IR (2012) Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(6):493–507

[16] Breiman L (2001) Random forests. Machine Learning 45(1):5–32

[17] Breiman L (2004) Consistency for a simple model of random forests. Tech. Rep. 670, Department of Statistics, University of Berkeley, USA

[18] Breiman L, Cutler A (2003) Random Forests Manual v4.0. University of California, Berkeley

[19] Breiman L, Cutler A (2016) Random forests-classification description. Department of Statistics Homepage, http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm, Accessed 15 Feb 2016

[20] Breiman L, Friedman J, Olshen R, Stone C (1984) Classification and Regression Trees. Wadsworth

[21] Brodersen K, Ong CS, Stephan K, Buhmann J (2010) The Balanced Accuracy and Its Posterior Distribution. In: Proceedings of 20th International Conference on Pattern Recognition (ICPR), pp 3121–3124

[22] Brodley CE, Utgoff PE (1995) Multivariate Decision Trees. Machine Learning 19(1):45–77

[23] Buddhakulsomsiri J, Zakarian A (2009) Sequential pattern mining algorithm for automotive warranty data. Computers & Industrial Engineering 57(1):137–147

[24] Burges CJC (1998) A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery 2(2):121–167

[25] Camps-Valls G, Rojo-Álvarez J, Martínez-Ramón M (2007) Kernel methods in bioengineering, signal and image processing. Idea Group Pub.

[26] Chang CC, Lin CJ (2011) LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) 2(27):1–27

[27] Chen C, Liaw A, Breiman L (2004) Using Random Forest to Learn Imbalanced Data. Tech. Rep. 666, Department of Statistics, University of Berkeley

[28] Choudhary AK, Harding JA, Tiwari MK (2008) Data mining in manufacturing: a review based on the kind of knowledge. Journal of Intelligent Manufacturing 20(5):501–521

[29] Cohen WW (1995) Fast Effective Rule Induction. In: Proceedings of the 12th International Conference on Machine Learning, pp 115–123

[30] Cox MAA, Cox TF (2008) Handbook of Data Visualization, Springer Berlin Heidelberg, chap Multidimensional Scaling, pp 315–347

[31] Dahinden C (2006) Classification with Tree-Based Ensembles Applied to the WCCI 2006 Performance Prediction Challenge Datasets. In: Proceedings of International Joint Conference on Neural Networks (IJCNN), pp 1669–1672

[32] Diedrich H, Abel DM (2012) lle: Locally linear embedding. http://CRAN.R-project.org/package=lle, R package version 1.1

[33] Dijkstra EW (1959) A note on two problems in connexion with graphs. Numerische Mathematik 1(1):269–271

[34] Do TN, Lenca P, Lallich S, Pham NK (2010) Classifying Very-High-Dimensional Data with Random Forests of Oblique Decision Trees. In: Guillet F, Ritschard G, Zighed D, Briand H (eds) Advances in Knowledge Discovery and Management, Studies in Computational Intelligence, vol 292, Springer Berlin Heidelberg, pp 39–55

[35] Dowling NE (1971) Fatigue failure predictions for complicated stress-strain histories. Tech. Rep. AD0736583, DTIC Document

[36] Fisher RA (1936) The Use Of Multiple Measurements in Taxonomic Problems. Annals of Eugenics 7(2):179–188

[37] Fix E, Hodges J (1951) Discriminatory analysis, Nonparametric Discrimination: Consistency properties. Tech. Rep. 4, USAF School of Aviation Medicine

[38] Floyd RW (1962) Algorithm 97: Shortest Path. Communications of the ACM 5(6):345

[39] Friedman JH, Hastie T, Tibshirani R (2010) Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software 33(1):1–22

[40] Frisk E, Krysander M, Larsson E (2014) Data-Driven Lead-Acid Battery Prognostics Using Random Survival Forests. In: Proceedings of the 2nd European Conference of the PHM Society (PHME)

[41] Fung G, Mangasarian OL (2001) Proximal Support Vector Machine Classifiers. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 77–86

[42] Fürnkranz J, Widmer G (1994) Incremental Reduced Error Pruning. In: Proceedings of the 8th International Conference on Machine Learning (ICML), pp 70–77

[43] Fürnkranz J, Gamberger D, Lavrač N (2012) Foundations of Rule Learning. Springer-Verlag Berlin Heidelberg

[44] Genuer R, Poggi JM, Tuleau-Malot C (2010) Variable Selection Using Random Forests. Pattern Recognition Letters 31(14):2225–2236

[45] Gusikhin O, Rychtyckyj N, Filev D (2007) Intelligent systems in the automotive industry: applications and trends. Knowledge and Information Systems 12(2):147–168

[46] Guyon I, Elisseeff A (2003) An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3:1157–1182

[47] Guyon I, Weston J, Barnhill S, Vapnik V (2002) Gene Selection for Cancer Classification Using Support Vector Machines. Machine Learning 46(1-3):389–422

[48] Hall MA (1999) Correlation-based Feature Subset Selection for Machine Learning. PhD thesis, University of Waikato

[49] Han J, Kamber M, Pei J (2011) Data Mining: Concepts and Techniques, 3rd edn. The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann

[50] Hastie T, Tibshirani R, Friedman J (2009) The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer Series in Statistics, Springer

[51] Herb F (2010) Alterungsmechanismen in Lithium-Ionen-Batterien und PEM-Brennstoffzellen und deren Einfluss auf die Eigenschaften von daraus bestehenden Hybrid-Systemen. PhD thesis, University of Ulm

[52] Hinton GE, Roweis ST (2003) Stochastic Neighbor Embedding. In: Becker S, Thrun S, Obermayer K (eds) Advances in Neural Information Processing Systems 15, MIT Press, pp 857–864

[53] Hoerl AE, Kennard RW (2000) Ridge Regression: Biased Estimation for Nonorthogonal Problems. Technometrics 42(1):80–86

[54] Hotelling H (1933) Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology 24(6):417–441

[55] Huang L, Murphey YL (2006) Text Mining with Application to Engineering Diagnostics. In: Proceedings of 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE), pp 1309–1317

[56] Ishwaran H, Kogalur UB, Blackstone EH, Lauer MS (2008) Random survival forests. The Annals of Applied Statistics 2(3):841–860

[57] Ji S (2013) Computational genetic neuroanatomy of the developing mouse brain: dimensionality reduction, visualization, and clustering. BMC Bioinformatics 14(1):1–14

[58] Jiang H, Deng Y, Chen HS, Tao L, Sha Q, Chen J, Tsai CJ, Zhang S (2004) Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes. BMC Bioinformatics 5(1):1–12

[59] Karatzoglou A, Smola A, Hornik K, Zeileis A (2004) kernlab – An S4 Package for Kernel Methods in R. Journal of Statistical Software 11(9):1–20

[60] Kirkpatrick S, Gelatt Jr CD, Vecchi MP (1983) Optimization by Simulated Annealing. Science 220:671–680

[61] Köhler M, Jenne S, Pötter K, Zenner H (2012) Zählverfahren und Lastannahme in der Betriebsfestigkeit. Springer

[62] Kononenko I (1994) Estimating attributes: Analysis and extensions of RELIEF. In: Proceedings of European Conference on Machine Learning (ECML), pp 171–182

[63] Krijthe J (2015) Rtsne: T-Distributed Stochastic Neighbor Embedding using Barnes-Hut Implementation. `http://CRAN.R-project.org/package=Rtsne`, R package version 0.10

[64] Kuhn M, Johnson K (2013) Applied Predictive Modeling. Springer

[65] Kuhn M, Wing J, Weston S, Williams A, Keefer C, Engelhardt A, Cooper T, Mayer Z, the R Core Team (2014) caret: Classification and Regression Training. `http://CRAN.R-project.org/package=caret`, R package version 6.0-24

[66] Kuhn M, Weston S, Coulter N, Culp M (2015) C50: C5.0 Decision Trees and Rule-Based Models. `http://CRAN.R-project.org/package=C50`, R package version 0.1.0-24

[67] Laczny CC, Pinel N, Vlassis N, Wilmes P (2014) Alignment-free Visualization of Metagenomic Data by Nonlinear Dimension Reduction. Scientific Reports 4:1–12

[68] Lavrač N, Flach P, Zupan B (1999) Rule Evaluation Measures: A Unifying View. In: Proceedings of 9th International Workshop on Inductive Logic Programming (ILP), pp 174–185

[69] Liaw A, Wiener M (2002) Classification and Regression by randomForest. R News 2(3):18–22

[70] Liebl J, Lederer M, Rohde-Brandenburger K, Biermann J, Roth M, Schäfer H (2014) Energiemanagement im Kraftfahrzeug: Optimierung von CO2-Emissionen und Verbrauch konventioneller und elektrifizierter Automobile. ATZ/MTZ-Fachbuch, Springer Fachmedien Wiesbaden

[71] Liu S, Patel RY, Daga PR, Liu H, Fu G, Doerksen RJ, Chen Y, Wilkins DE (2012) Combined Rule Extraction and Feature Elimination in Supervised Classification. IEEE Transactions on NanoBioscience 11(3):228–236

[72] López V, Fernández A, García S, Palade V, Herrera F (2013) An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Information Sciences 250(0):113–141

[73] van der Maaten L (2007) An Introduction to Dimensionality Reduction Using Matlab. Tech. Rep. 07-07, Maastricht University

[74] van der Maaten L (2014) Accelerating t-SNE Using Tree-based Algorithms. Journal of Machine Learning Research 15(1):3221–3245

[75] van der Maaten L, Hinton G (2008) Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(11):2579–2605

[76] van der Maaten L, Postma E, van den Herik H (2009) Dimensionality Reduction: A Comparative Review. Tech. rep., Tillburg University

[77] Maldonado S, Weber R (2009) A wrapper method for feature selection using Support Vector Machines. Information Sciences 179(13):2208–2217

[78] Marin J, Vazquez D, Lopez A, Amores J, Leibe B (2013) Random Forests of Local Experts for Pedestrian Detection. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), pp 2592–2599

[79] Marscholik C, Subke P (2008) Road Vehicles – Diagnostic Communication: Technology and Applications. Hüthig

[80] Martens H (2001) Reliable and relevant modelling of real world data: a personal account of the development of PLS Regression. Chemometrics and Intelligent Laboratory Systems 58(2):85–95

[81] Matsuishi M, Endo T (1968) Fatigue of metals subjected to varying stress. Japan Society of Mechanical Engineers 1968:37–40

[82] Menze B, Splitthoff N (2012) obliqueRF: Oblique Random Forests from Recursive Linear Model Splits. `http://CRAN.R-project.org/package=obliqueRF`, R package version 0.3

[83] Menze BH, Kelm MB, Masuch R, Himmelreich U, Bachert P, Petrich W, Hamprecht FA (2009) A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data. BMC Bioinformatics 10(1):1–16

[84] Menze BH, Kelm BM, Splitthoff DN, Koethe U, Hamprecht FA (2011) On Oblique Random Forests. In: Machine Learning and Knowledge Discovery in Databases, Springer, pp 453–469

[85] Mevik BH, Wehrens R (2007) The pls Package: Principal Component and Partial Least Squares Regression in R. Journal of Statistical Software 18(2):1–24

[86] Mitchell TM (1997) Machine Learning. McGraw Hill

[87] Murthy SK, Kasif S, Salzberg S (1994) A System for Induction of Oblique Decision Trees. Journal of Artificial Intelligence Research 2(1):1–32

[88] Napierała K (2012) Improving Rule Classifiers For Imbalanced Data. PhD thesis, Poznan University of Technology

[89] Oksanen J, Blanchet FG, Kindt R, Legendre P, Minchin PR, O'Hara RB, Simpson GL, Solymos P, Stevens MHH, Wagner H (2015) vegan: Community Ecology Package. `http://CRAN.R-project.org/package=vegan`, R package version 2.3-0

[90] Osuna E, Freund R, Girosi F (1997) Support Vector Machines: Training and Applications. Tech. rep., Massachusetts Institute of Technology

[91] Parfionovas A (2013) Enhancement of Random Forests Using Trees with Oblique Splits. PhD thesis, Utah State University

[92] Pearson K (1901) On lines and planes of closest fit to systems of points in space. Philosophical Magazine 2(6):559–572

[93] Prytz R (2014) Machine learning methods for vehicle predictive maintenance using off-board and on-board data. Licentiate thesis, Halmstad University

[94] Prytz R, Nowaczyk S, Rögnvaldsson T, Byttner S (2013) Analysis of Truck Compressor Failures Based on Logged Vehicle Data. In: Proceedings of 9th International Conference on Data Mining

[95] Prytz R, Nowaczyk S, Rögnvaldsson T, Byttner S (2015) Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data. Engineering Applications of Artificial Intelligence 41(0):139–150

[96] Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA

[97] Quinlan JR (2016) C5.0: An informal tutorial. RuleQuest Research Website, `https://www.rulequest.com/see5-unix.html`, Accessed 30 Mar 2016

[98] Rajpathak DG (2013) An ontology based text mining system for knowledge discovery from the diagnosis data in the automotive domain. Computers in Industry 64(5):565–580

[99] Reif K, Dietsche K, GmbH R (2010) Kraftfahrtechnisches Taschenbuch. Studium und Praxis, Vieweg+Teubner Verlag

[100] Rissanen J (1978) Modeling by shortest data description . Automatica 14(5):465–471

[101] Roweis ST, Saul LK (2000) Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science 290:2323–2326

[102] Saha B, Goebel K (2007) Battery Data Set. NASA Ames Prognostics Data Repository, `http://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/#battery`, Accessed 12 Jan 2015

[103] Sammon J (1969) A Nonlinear Mapping for Data Structure Analysis. IEEE Transactions on Computers C-18(5):401–409

[104] Schijve J (2009) Fatigue of Structures and Materials. 2, Springer Netherlands

[105] Schneider M, Hirsch S, Weber B, Székely G, Menze BH (2015) Joint 3-D vessel segmentation and centerline extraction using oblique Hough forests with steerable filters. Medical Image Analysis 19(1):220–249

[106] Shen KQ, Ong CJ, Li XP, Hui Z, Wilder-Smith E (2007) A Feature Selection Method for Multilevel Mental Fatigue EEG Classification. IEEE Transactions on Biomedical Engineering 54(7):1231–1237

[107] Shi T, Horvath S (2006) Unsupervised Learning With Random Forest Predictors. Journal of Computational and Graphical Statistics 15(1):118–138

[108] Shi T, Seligson D, Belldegrun AS, Palotie A, Horvath S (2005) Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma. Modern Pathology 18:547–557

[109] Sokolova M, Lapalme G (2009) A Systematic Analysis of Performance Measures for Classification Tasks. Information Processing & Management 45(4):427–437

[110] Strobl C (2008) Statistical Issues in Machine Learning – Towards Reliable Split Selection and Variable Importance Measures. PhD thesis, LMU Munich

[111] Tax DM, Duin RP (2004) Support Vector Data Description. Machine Learning 54(1):45–66

[112] Tenenbaum JB, Silva Vd, Langford JC (2000) A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290(5500):2319–2323

[113] Theissler A (2013) Detecting anomalies in multivariate time series from automotive systems. PhD thesis, Brunel University School of Engineering and Design

[114] Therneau T, Atkinson B, Ripley B (2014) rpart: Recursive Partitioning and Regression Trees. `http://CRAN.R-project.org/package=rpart`, R package version 4.1-8

[115] Tibshirani R (1996) Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society, Series B 58:267–288

[116] TomTom International BV (2016) Tomtom traffic index - measuring congestion worldwide. `https://www.tomtom.com/en_gb/trafficindex/#/list`, Accessed 12 Feb 2016

[117] Truong AKY (2009) Fast Growing and Interpretable Oblique Trees via Logistic Regression Models. PhD thesis, University of Oxford, Accessed 25 Jan 2015

[118] Vapnik VN (1995) The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc.

[119] Venables WN, Ripley BD (2002) Modern Applied Statistics with S, 4th edn. Springer, New York

[120] Wan V, Campbell WM (2000) Support vector machines for speaker verification and identification. In: Proceedings of the IEEE Signal Processing Society Workshop, vol 2, pp 775–784

[121] Wang L (2005) Support Vector Machines: Theory and Applications. Studies in Fuzziness and Soft Computing, Springer

[122] Webb IG, Zhang S (2005) K-Optimal Rule Discovery. Data Mining and Knowledge Discovery 10(1):39–79

[123] Wirsching P, Mohsen Shehata A (1977) Fatigue Under Wide Band Random Stresses Using the Rain-Flow Method. Journal of Engineering Materials and Technology 99:205–211

[124] Witten IH, Frank E, Hall MA (2011) Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann

[125] Wold S (2001) Personal memories of the early PLS development. Chemometrics and Intelligent Laboratory Systems 58(2):83–84

[126] Yu G, Geist A, Ostrouchov G, Samatova NF (2003) An SVM-based algorithm for identification of photosynthesis-specific genome features. In: Proceedings of the 2nd IEEE Computer Society Conference on Bioinformatics (CSB), pp 235–243

[127] Zhang J, Marszalek M, Lazebnik S, Schmid C (2007) Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. International Journal of Computer Vision 73(2):213–238

[128] Zou H, Hastie T (2005) Regularization and variable selection via the Elastic Net. Journal of the Royal Statistical Society: Series B 67:301–320