Automatisierte Fahrzeugbalance-Analyse im Motorsport: Entwicklung und Evaluation von Gradient-Boosting-Modellen auf Basis von Porsche LMDh Telemetriedaten

2. Projektarbeit

vorgelegt am 24.11.2025

Fakultät Wirtschaft und Gesundheit

Studiengang Wirtschaftsinformatik

Kurs WWI2023A

von

Jo Imping

Betreuung in der Ausbildungsstätte:

Dr. Ing. h.c. F. Porsche AG
Paul, Stiegele
IT Product Manager

Unterschrift

DHBW Stuttgart:

Prof. Dr. Alexander Brandt

Sperrvermerk

Die vorliegende Projektarbeit enthält zum Teil Informationen, die nicht für die Öffentlichkeit bestimmt sind. Während einer Sperrzeit von 5 Jahren ab dem Abgabedatum liegt das alleinige Recht zur Verwertung, insbesondere zur Verbreitung der Projektarbeit, auch auf elektronischen Medien, bei der Dr. Ing. h.c. F. Porsche AG.

Während dieser Sperrzeit darf die Projektarbeit - sei es in Teilen oder als Ganzes - nur mit der ausdrücklichen schriftlichen Genehmigung der Dr. Ing. h.c. F. Porsche AG an Dritte weitergegeben werden.

Die vorliegende Projektarbeit ist zur Vorlage zur Anerkennung der Prüfungsleistung freigegeben.

Stuttgart, den 29.10.2025

Unterschrift Fachabteilungsleiter

Inhaltsverzeichnis

Abkürzungsverzeichnis

DSR Design Science Research

EDA Explorative Datenanalyse

GBDT Gradient Boosting Decision Trees

KQL Kusto Query Language

ML Machine Learning

MAE Mean Absolute Error

MSE Mean Squared Error

LMDh Le Mans Daytona hybrid

RMSE Root Mean Squared Error

SVR Support Vector Regression

IMSA International Motor Sports Association

WEC World Endurance Championship

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Problemstellung

Der Motorsport ist ein hochkompetitives Umfeld, in dem Verbesserungen der Fahrzeugperformance oft nur Bruchteile von Sekunden bringen, aber entscheidend sind. Moderne Rennfahrzeuge sind mit Tausenden von Sensoren ausgestattet, die kontinuierlich Telemetriedaten erfassen und übertragen. Diese Datenmengen ermöglichen eine beispiellose Sichtbarkeit in Fahrzeugverhalten, Streckenbedingungen und Fahrerdynamik. Trotz dieser technologischen Verfügbarkeit verlässt sich die Telemetrie-Analyse in der Praxis stark auf manuelle Prozesse: Renningenieure sichten Dashboards, identifizieren Auffälligkeiten und leiten daraus Setup-Anpassungen ab. Dies ist zeitintensiv und anfällig für Übersehungen subtiler Muster, die erst bei Kombination mehrerer Parameter sichtbar werden. Diese Situation repräsentiert ein klassisches Problem der angewandten Informatik: Ein großes Datenvolumen, klare Geschäftsziele, aber unzureichende Automatisierung zur systematischen Mustererkennung. Machine Learning (ML) verspricht hier die Fähigkeit, aus historischen Daten Muster zu erkennen und auf neue Situationen zu generalisieren.

Die Fahrzeugbalance ist eine Schlüsselgröße für Rennfahrer und Renningenieure. Sie beschreibt die Tendenz des Fahrzeugs, in Kurvenfahrt weniger oder stärker zu lenken als vom Fahrer über das Lenkrad eingegeben. Die Fahrzeugbalance beeinflusst direkt Fahrbarkeit, Sicherheit und Renngeschwindigkeit. Die Balance hängt von zahlreichen, komplex verflochtenen Einflussfaktoren ab. Das zentrale Problem besteht darin, dass Renningenieure diese Zusammenhänge derzeit nicht systematisch erfassen. Sie können nicht vorhersagen, welche spezifischen Parameterkombinationen zu welchen Balance-Änderungen führen. Dies zwingt sie zu reaktivem Handeln: Erst wenn Abweichungen in den Daten offensichtlich werden, werden Anpassungen vorgenommen. Ein proaktiver, vorhersagender Ansatz wäre wertvoll, da er die Fähigkeit bietet, frühzeitig zu signalisieren, dass sich die Balance-Charakteristik verschiebt und damit setup-Optimierungen proaktiv zu planen.

Damit ergibt sich die zentrale Forschungsmotivation: Können ML-Modelle aus Telemetriedaten lernen, Fahrzeugbalance vorherzusagen? Und wenn ja, unter welchen Bedingungen generalisieren diese Modelle zuverlässig auf neue Renn-Events mit veränderten Kontexten?

1.2 Zielsetzung

Diese Arbeit verfolgt das Ziel, ein ML-Modell zur Vorhersage der Fahrzeugbalance auf Basis aggregierter Telemetrie-Metriken zu entwickeln und systematisch zu evaluieren. Das Modell soll dazu dienen, Renningenieure zu unterstützen, indem es automatisiert Vorhersagen liefert, statt dass Ingenieure manuell Daten analysieren müssen.

Die Entwicklung folgt der Design Science Research (DSR) Methodologie, die in Kapitel 2 eingeführt wird. Dies bedeutet konkret: (1) Systematische Anforderungsanalyse aus der Anwendungsdomäne, (2) Rigorous Build des Artefakts, (3) Umfassende Evaluierung anhand wissenschaftlicher Metriken, und (4) Reflexion von Design Knowledge für die Wissensbasis.

1.3 Forschungsansatz und Aufbau der Arbeit

Die vorliegende Arbeit strukturiert sich in sechs Kapitel, die den DSR-Prozess vom Problem zur Lösung abbilden. Kapitel 2 vermittelt die theoretischen Grundlagen, Kapitel 3 analysiert Anforderungen, Kapitel 4 dokumentiert das Design und die Entwicklung des Artefakts, Kapitel 5 evaluiert das Modell-Artefakt und Kapitel 6 schließt den DSR-Zyklus mit Reflexion und Ausblick.

Diese Arbeit verbindet technische Artefakt-Entwicklung mit methodisch fundierter Evaluation. So wird das entwickelte ML-Modell anhand etablierter Metriken bewertet, und die gewonnenen Erkenntnisse tragen zur Wissensbasis über ML-Anwendungen im Motorsport bei.

2 Theoretische Grundlagen

Zur Entwicklung eines ML-Modells für Motorsport-Telemetrie-Analyse müssen mehrere ineinandergreifende Konzepte verstanden werden. Dieses Kapitel vermittelt zunächst methodologische Grundlagen (Design Science Research, Datenvorverarbeitung), dann die Theorie von Regressionsproblemen und speziell Gradient Boosting Decision Trees, und abschließend praktische Aspekte von Optimierung und Validierung. Diese Grundlagen bilden die wissenschaftliche Basis für die in Kapitel 4 beschriebene Artefakt-Entwicklung.

2.1 Design Science Research Methodologie

DSR ist ein etablierter Forschungsansatz der Wirtschaftsinformatik, der sich grundlegend von deskriptiven Forschungsmethoden unterscheidet. Während traditionelle empirische Forschung primär darauf ausgerichtet ist, bestehende Phänomene zu verstehen und zu erklären, zielt DSR darauf ab, praktische Probleme durch systematische Entwicklung und rigorose Evaluierung von Artefakten zu lösen. Damit schafft DSR eine Brücke zwischen wissenschaftlicher Fundierung und praktischer Problemlösung.[1]

Das Grundwerk von Hevner et al. (2004) prägt bis heute das Verständnis von DSR und etabliert ein Framework, das auf sieben präskriptiven Richtlinien basiert. Das Framework schreibt vor, dass DSR-Projekte in drei ineinandergreifenden Zyklen durchgeführt werden sollten: Der Relevance Cycle beginnt mit Problemidentifikation aus der Anwendungsdomäne; der Rigor Cycle verankert die Entwicklung in wissenschaftlichem Wissen und etablierten Theorien; der Design Cycle orchestriert iterative Phasen von Artefakt-Konzeption, Entwicklung und Evaluierung. Diese drei Zyklen ermöglichen eine systematische und nachvollziehbare Forschungsvorgehensweise, bei der wissenschaftliche Strenge nicht auf Kosten von Praxisrelevanz geht.[2]

Der DSR-Prozess gliedert sich typischerweise in sechs sequenzielle Phasen. Problem Identification and Motivation beginnt mit der Analyse der Problemdomäne und Begründung ihrer wissenschaftlichen und praktischen Relevanz. Definition of Objectives spezifiziert die Anforderungen, die das entwickelte Artefakt erfüllen muss. In der Design and Development Phase wird das Artefakt konzipiert und prototypisch implementiert. Die Demonstration Phase dokumentiert, dass das Artefakt das Problem tatsächlich lösen kann, typischerweise durch Fallstudien oder kontrollierte Szenarien. Die Evaluation Phase bewertet das Artefakt systematisch gegen die vordefinierten Anforderungen und Ziele. Abschließend erfolgt die Communication Phase, in der Erkenntnisse und Design Knowledge der wissenschaftlichen Gemeinschaft mitgeteilt werden.[3]

Artefakte in DSR können verschiedene Formen annehmen. Constructs sind konzeptionelle Vokabularien und Abstraktionen, die Probleme präzise definieren. Models stellen Zusammenhänge

[1] Vgl. Hevner et al. 2004, S. 77-83
[2] Vgl. Hevner et al. 2004, S. 77-92
[3] Vgl. Peffers et al. 2007, S. 45-77

und Strukturen in vereinfachter Form dar. Methods sind Algorithmen und systematische Verfahrensweisen zur Problemlösung. Instantiations schließlich sind konkrete Implementierungen oder Prototypen.[4] In ML-Projekten ist die Instantiation typischerweise ein trainiertes Modell mit vollständiger Pipeline (Datenvorverarbeitung, Feature Engineering, trainierte Parameter). Die vorliegende Arbeit entwickelt eine Instantiation: ein evaluiertes ML-Modell für Fahrzeugbalance-Vorhersage.

Die Evaluierung in DSR erfüllt mehrere funktionale Rollen. Sie stellt fest, ob und inwieweit das Artefakt die definierten Anforderungen erfüllt. Sie identifiziert Verbesserungspotenziale für weitere Iterationen. Vor allem trägt sie zur wissenschaftlichen Wissensbasis bei, indem Design Principles und generalisierbare Lessons Learned dokumentiert werden.[5] Bewährte Evaluationsmethoden in DSR sind observational (Feldbeobachtung), analytical (logische Deduktion und Proof-of-Concept), experimental (kontrollierte Experimente mit Baseline-Vergleich), testing (systematische Funktionsprüfung) und descriptive (qualitative Bewertung durch Experten).[6] Für ML-Artefakte dominieren analytische und experimentelle Evaluationen mittels etablierter Performance-Metriken.[7]

2.2 Experteninterviews zur Anforderungsermittlung

Die Anforderungsanalyse in Kapitel 3 basiert auf zwei informellen Gesprächen mit einem erfahrenen Performance Engineer aus dem Porsche Motorsport-Team. Das erste Gespräch fand am 29.08.2025 statt, das zweite am 12.09.2025. Ziel war es, durch offene Diskussion die tägliche Arbeitsweise im Telemetrie-Management nachzuvollziehen, zentrale technische Herausforderungen zu identifizieren und realistische Anforderungen an ein automatisiertes Vorhersagemodell zu formulieren. Der interviewte Experte agiert als Teamleiter der Performance-Abteilung im Porsche LMDh-Programm und ist verantwortlich für die Fahrzeugentwicklung, Simulationsmodellierung und insbesondere die datenbasierte Optimierung des Rennwagen-Setups während des operativen Betriebs, wodurch er über tiefgehendes, direkt anwendbares Wissen im Telemetrie-Management verfügt.[8]

Beide Gespräche folgten einem offenen, exploratorischen Format ohne strukturierten Fragenkatalog, um eine natürliche Konversation zu fördern und implizites Wissen des Experten zugänglich zu machen. Die gewonnenen Erkenntnisse bestätigten, dass die aktuelle Telemetrie-Analyse stark manuell erfolgt und mehrere Stunden pro Rennwochenende erfordert. Dies validierte die Problemrelevanz und leitete die Anforderungsableitung in Kapitel 3.[9]

[4] Vgl. Hevner et al. 2004, S. 80
[5] Vgl. Venable, Pries-Heje, Baskerville 2016, S. 79–81; vgl. dazu auch Gregor, Jones 2007, S. 322–325, sowie Hevner et al. 2004, S. 84–85
[6] Vgl. Hevner et al. 2004, S. 85-86
[7] Vgl. Friedman, J., Hastie, Tibshirani 2009, S. 420–430
[8] Vgl. Experteninterview 1, 29.08.2025, Z. 3-27
[9] Vgl. Experteninterview 1 und 2, dokumentiert in Anhang A.1

2.3 Maschinelle Lernverfahren für Regressionsprobleme

ML wird allgemein als automatische Induktion von vorhersagenden Modellen aus Daten definiert, ohne dass Algorithmen explizit programmiert werden müssen.[10] Im praktischen Kontext bedeutet dies: Ein Lernalgorithmus erhält Trainingsdaten, erkennt Muster in diesen Daten und extrahiert eine verallgemeinerbare mathematische Struktur, die auf neue Daten angewendet werden kann.

Das vorliegende Projekt verfolgt ein Regressionsziel: Vorhersage einer kontinuierlichen Zielvariable (Fahrzeugbalance-Wert) aus einer Menge strukturierter Input-Features (Telemetrie-Metriken). Dies unterscheidet sich von Klassifikation, bei der diskrete Kategorien vorhergesagt werden. Regression ist ein Supervised-Learning-Problem: Jede Trainingsinstanz hat ein bekanntes, korrektes Label (den gemessenen Fahrzeugbalance-Wert), gegen das das Modell seine Vorhersagen abgleichen kann.[11]

Das zentrale Problem beim Modelllernen wird durch den Bias-Variance Trade-off beschrieben. Ein Modell mit niedriger Komplexität wie lineare Regression hat hohen Bias: Es macht systematisch vereinfachte Vorhersagen, die die tatsächliche nichtlineare Beziehung zwischen Features und Zielvariable nicht erfassen und somit zu Underfitting führen. Umgekehrt weist ein hochkomplexes Modell wie ein überparametrisiertes Polynom hohe Varianz auf: Es memoriert Trainingsrauschen und besondere Trainingsfälle und generalisiert dadurch schlecht auf neue Daten, was als Overfitting bezeichnet wird.[12] Das Ziel ist ein gutes Gleichgewicht: Ein Modell mit ausreichender Komplexität, um die echte Struktur der Daten zu erfassen, aber nicht so komplex, dass es Zufallsrauschen memoriert.

Die Vorbereitung der Daten ist entscheidend für späteren Erfolg. Exploratory Data Analysis (EDA) untersucht zunächst die Rohverteilungen der Features, identifiziert Korrelationen zwischen Variablen und erkennt potenzielle Ausreißer oder Anomalien.[13] Data Cleaning adressiert praktische Datenqualitätsprobleme: fehlende Werte werden behandelt durch Imputation oder Ausschluss, anomale Messwerte werden gefiltert, und sachlogische Schwellwerte werden gesetzt, etwa der Ausschluss von Messungen mit unplausiblen Sensorwerten.[14]

Feature Engineering ist der kreative Schritt, in dem Rohdaten in aussagekräftige Prädiktoren transformiert werden. Feature Selection wählt von vielen möglichen Kandidaten diejenigen aus, die am meisten zur Vorhersage beitragen und reduziert dadurch Overfitting sowie Trainingszeit.[15] Aggregation kombiniert hochkorrelierte Sensoren zu zusammengefassten Features, beispielsweise der Durchschnitt mehrerer Temperatur-Sensoren, um Rauschredundanz zu minimieren.[16] Encoding wandelt kategoriale Variablen wie Track-Identifikatoren in numerische Repräsentationen

[10] Vgl. Mitchell 1997, S. 1-2
[11] Vgl. Hastie et al. 2009, S. 1-25
[12] Vgl. Hastie et al. 2009, S. 23-31
[13] Vgl. Tukey 1977, S. 1–50
[14] Vgl. Chapman et al. 2000, S. 23–28
[15] Vgl. Guyon, Elisseeff 2003, S. 1157-1182
[16] Vgl. Guyon, Elisseeff 2003, S. 1157-1182

um.[17] Für Telemetriedaten ist Zeitreihen-Glättung wie Moving Averages essentiell: Sie reduziert hochfrequentes Sensorrauschen, ohne Trends zu zerstören, und ermöglicht es, echte physikalische Änderungen von Messfehlern zu unterscheiden.[18]

Nach der Datenvorbereitung entsteht eine zentrale Designfrage: Welcher Regressionsalgorithmus ist am besten geeignet? Lineare Modelle wie Linear Regression, Ridge und Lasso bieten exzellente Interpretierbarkeit, können aber intrinsisch nur lineare Beziehungen erfassen. Interpretierbarkeit hierbei, dass nachvollzogen werden kann, welche Features wie stark die Vorhersage beeinflussen.[19] Support Vector Regression nutzt mathematische Tricks, die sogenannten Kernel-Tricks, um implizit nichtlineare Feature-Transformationen durchzuführen, bleibt aber schwer zu interpretieren.[20] Decision Trees sind intuitiv: Sie partitionieren den Feature-Raum sequenziell anhand scharfer Grenzen, etwa wenn Feature X größer als 5 ist, dann gehe linken Ast. Sie sind robust und können komplexe nichtlineare Muster erfassen, leiden aber unter Overfitting, da ein einzelner Baum Trainingsrauschen memoriert.[21] Random Forests verbessern einzelne Bäume durch Ensemble-Averaging: Viele verschiedene Bäume werden auf zufällig gestörten Datensätzen trainiert, und ihre Vorhersagen werden gemittelt, was Varianz reduziert und Generalisierung verbessert.[22] Boosting-Methoden folgen einem anderen Ensemble-Prinzip: Sie trainieren Bäume sequenziell, wobei jeder neue Baum systematisch die Fehler vorheriger Bäume korrigiert.[23] Neuronale Netze können beliebig komplexe Funktionen approximieren und funktionieren gut bei großen Datenmengen, erfordern aber typischerweise mehr Trainingsdaten als traditionelle ML-Methoden und sind weniger interpretierbar.[24]

Die Frage stellt sich: Welche Methode eignet sich für das vorliegende Motorsport-Problem mit strukturierten, tabulischen Daten von moderater Größe, in denen konkurrierende komplexe Effekte zwischen Features existieren und Interpretierbarkeit gewünscht ist, um Renningenieure zu unterstützen? Unter diesen Bedingungen haben sich Gradient Boosting Decision Trees als Methode der Wahl etabliert, weil sie nichtlineare Muster erfassen, auf moderaten Datenmengen gut funktionieren und ein gutes Gleichgewicht zwischen Vorhersagegenauigkeit und Interpretierbarkeit bieten.[25]

2.4 Gradient Boosting Decision Trees

Gradient Boosting Decision Trees (GBDT) sind eine Ensemble-Methode, die sequenzielle schwache Lerner, das heißt einfache Decision Trees, zu einem starken Vorhersagemodell kombiniert.[26]

[17] Vgl. Kuhn, Johnson 2019, S. 139–170
[18] Vgl. Box et al. 2015, S. 25–45; vgl. dazu auch Cleveland 1979, S. 829–836
[19] Vgl. Hastie et al. 2009, S. 58-85
[20] Vgl. Hastie et al. 2009, S. 290-310
[21] Vgl. Breiman 2001, S. 5-32
[22] Vgl. Breiman 2001, S. 5-32
[23] Vgl. Friedman 2001, S. 1189-1232
[24] Vgl. Goodfellow et al. 2016, S. 164-223
[25] Vgl. Chen, Guestrin 2016, S. 785-794
[26] Vgl. Friedman 2001, S. 1189-1232

Das Kernprinzip ist Residual Learning: Der erste Baum wird auf die Rohdaten trainiert. Der zweite Baum wird trainiert, um die Fehler des ersten Baums vorherzusagen, die sogenannten Residuen. Der dritte Baum korrigiert dann die kombinierten Fehler von Baum 1 und 2, und so weiter. Die finale Vorhersage ergibt sich aus einer gewichteten Summe aller Baum-Ausgaben.

Formalisiert wird dies durch Gradient Descent: Eine Loss-Funktion, typischerweise Mean Squared Error für Regression, quantifiziert, wie schlecht die Vorhersagen sind. Jeder neue Baum wird so konstruiert, dass er in Richtung des negativen Gradienten dieser Loss-Funktion läuft, also gezielt Fehler reduziert.[27] Dieser Gradient-Ansatz führt zu schnellerer Konvergenz als alternatives Ensemble-Averaging.

Warum ist GBDT für Regressionsprobleme bevorzugt? Erstens erfasst es nichtlineare Beziehungen zwischen Features und Zielvariable, was lineare Modelle nicht können. Zweitens funktioniert es auf moderaten Datenmengen gut, im Gegensatz zu neuronalen Netzen. Drittens ist es vergleichsweise schnell zu trainieren. Viertens bietet es Feature-Importance-Schätzungen, die zeigen, welche Telemetrie-Signale am meisten zur Fahrzeugbalance-Vorhersage beitragen.

Zwei prominente Implementierungen konkurrieren: XGBoost (eXtreme Gradient Boosting) ist die ältere Implementierung, seit 2014 verfügbar, und bietet hochoptimierte Algorithmen mit integrierten Regularisierungsmechanismen wie L1- und L2-Penalisierungen, die Overfitting kontrollieren.[28] Sie nutzt Histogram-basiertes Split-Finding: Statt Splits für alle kontinuierlichen Werte zu evaluieren, diskretisiert XGBoost Features in Histogramm-Bins, was zu Geschwindigkeitsvorteil führt. Sie hat native Unterstützung für kategoriale Features, wodurch manuelle Encoding-Schritte entfallen. XGBoost wächst Bäume level-wise: Bei jedem Schritt werden alle Blätter auf derselben Tiefe erweitert.

LightGBM (Light Gradient Boosting Machine) ist eine neuere Implementierung von Microsoft ab 2016, die speziell für Geschwindigkeit auf großen Datensätzen optimiert ist.[29] Sie verwendet Leaf-wise Wachstum: Statt alle Blätter auf gleicher Tiefe zu halten, erweitert sie bei jedem Schritt das Blatt mit dem höchsten Fehlerreduktionspotenzial, was zu tieferen, schmäleren Bäumen führt. Sie nutzt Gradient-based One-Side Sampling: Trainingsinstanzen mit großen Gradienten, die schlecht vorhergesagten, behalten volle Gewichtung, während Instanzen mit kleinen Gradienten, die gut vorhergesagten, untersampled werden. Sie nutzt Exclusive Feature Bundling: Hochkorrelierte Features werden kombiniert, um Dimensionalität zu reduzieren. Diese Optimierungen machen LightGBM oft schneller, zum Preis leicht erhöhten Overfitting-Risikos.

Empirisch zeigen beide Implementierungen überlegene Performance auf strukturierten tabularen Daten im Vergleich zu tiefen neuronalen Netzen.[30] Die Wahl zwischen ihnen ist oft pragmatisch: XGBoost für Balance, LightGBM wenn Geschwindigkeit kritisch ist.

[27] Vgl. Friedman 2001, S. 1200-1210
[28] Vgl. Chen, Guestrin 2016, S. 785-794
[29] Vgl. Ke et al. 2017, S. 1-10.
[30] Vgl. Chen, Guestrin 2016, S. 788-792

2.5 Hyperparameter-Optimierung und Modellvalidierung

Um ein GBDT-Modell zu trainieren, müssen vor dem Training viele Hyperparameter gesetzt werden. Hyperparameter unterscheiden sich fundamental von Modell-Parametern, den internen Gewichten des Modells, die während Training gelernt werden. Hyperparameter sind konfigurierbare Knöpfe, die Architekt oder Architektin des Modells kontrolliert.[31]

Für GBDT sind mehrere Hyperparameter besonders wichtig: Die Anzahl der Estimators bestimmt, wie viele Bäume insgesamt trainiert werden, während die Learning Rate als Schrittweite beim Gradientenverfahren fungiert – kleine Werte führen zu langsameren, stabileren Verbesserungen, große Werte zu schnellerem, aber riskanterem Lernen. Die Max Depth definiert die maximale Tiefe jedes Baums und erhöht mit steigenden Werten die Modellkomplexität. Das Min Child Weight (bzw. die Mindestanzahl an Samples pro Blatt) bewirkt bei höheren Werten konservativere Splits und damit weniger Overfitting, während Subsample den Anteil der Trainingsinstanzen pro Baum festlegt (z. B. 0,8 für 80%). Schließlich kontrollieren Regularisierungsparameter wie L2-Penalisierung die Modellkomplexität.[32] Da manuelle Anpassung dieser Parameter ineffizient ist, nutzt man systematische Hyperparameter-Optimierung. Grid Search definiert ein vorgegebenes Gitter von Parameterwerten und probiert alle Kombinationen systematisch durch, garantiert zwar eine gründliche Suche, führt aber zu exponentiellem Rechenaufwand mit der Anzahl der Parameter.[33] Nach der Hyperparameter-Optimierung ergibt sich eine weitere zentrale Herausforderung: Generalisiert das Modell auf unbekannte Daten? Diese wird durch Modellvalidierung beantwortet. Der Standard k-fold Cross-Validation partitioniert den Datensatz in k gleiche Teile und trainiert das Modell k-mal, jeweils mit k minus eins Teilen als Training und einem Teil als Validierung. Die gemittelten k Validierungs-Scores geben eine robuste Schätzung der Generalisierungsperformance mit geringerer Varianz als ein einzelner Train-Test-Split.[34]

Im vorliegenden Projekt ist jedoch die kritischste Generalisierungsfrage der Transfer auf komplett neue Rennevents, da eine neue Veranstaltung völlig andere Umgebung, andere Fahrer und anderes Setup aufweist. Standard-Cross-Validation löst diese Herausforderung nicht, weshalb hier Leave-One-Event-Out verwendet wird. Dafür wird das Modell auf einem Trainingsdatensatz trainiert, der alle Veranstaltungen außer einer enthält, und dann auf der zurückgehaltenen Veranstaltung validiert.[35] Dies stellt einen extremen, aber realistischen Stress-Test für echte Domänen-Generalisierung dar und offenbart, ob das Modell echte physikalische Strukturen gelernt hat oder nur Muster des Trainingsdatensatzes memoriert hat.

Diese drei Komponenten Hyperparameter-Optimierung, Cross-Validation Strategie und Evaluationsmetrik sind untrennbar: Zusammen stellen sie sicher, dass ein entwickeltes Modell tatsächlich verallgemeinerbar ist, nicht nur auf Trainingsdaten overfittet.

[31] Vgl. Bergstra, Bengio 2012, S. 281-305
[32] Vgl. Chen, Guestrin 2016, S. 787–788
[33] Vgl. Bergstra, Bengio 2012, S. 281-305; vgl. dazu auch Chen, Guestrin 2016, S. 787–788
[34] Vgl. Kohavi 1995, S. 1137-1145
[35] Vgl. Kapitel 4.1

2.6 Evaluationsmetriken für Regression

Regressionsergebnisse werden durch mehrere etablierte Fehlermetriken quantifiziert. Der Mean Squared Error (MSE) berechnet das Durchschnitt der quadrierten Abweichungen zwischen Vorhersagen und Ist-Werten und bestraft größere Fehler überproportional. Der Root Mean Squared Error (RMSE) ist die Quadratwurzel des MSE und hat dieselbe Einheit wie die Zielvariable, was die Interpretation erleichtert.[36]

Der Mean Absolute Error (MAE) quantifiziert den Durchschnitt der absoluten Abweichungen und ist robuster gegenüber Ausreißern, da er Fehler linear (nicht quadratisch) gewichtet.[37] Die Wahl zwischen RMSE und MAE sollte sich nach der erwarteten Fehlerverteilung richten: RMSE ist optimal bei normalverteilten Fehlern, MAE bei Laplace-verteilten Fehlern.

Das Bestimmtheitsmaß R^2 (Coefficient of Determination) gibt an, welcher Anteil der Varianz der Zielvariable durch das Modell erklärt wird. $R^2 = 1$ signalisiert perfekte Vorhersagen, $R^2 = 0$ bedeutet, dass das Modell nicht besser als die Mittelwert-Baseline ist. Negative R^2-Werte sind möglich und deuten auf schlechtere Performance als die Baseline hin.[38] In der Ingenieurpraxis und insbesondere im Motorsport-Datenkontext werden für Validierungsmodelle üblicherweise Schwellwerte von $R^2 \geq 0{,}7$ angestrebt, da dies eine Fehlerreduktion von etwa 50% gegenüber einem Baseline-Modell bedeutet und damit praktische Einsatzfähigkeit gewährleistet.[39]

Diese drei Metriken (RMSE, MAE, R^2) bilden den internationalen Standard in der Regressionsanalyse und ermöglichen Vergleichbarkeit mit etablierten Benchmarks in der Fachliteratur.

[36] Vgl. Hodson 2022, S. 5481-5482
[37] Vgl. Chai, Draxler 2014, S. 1247-1250
[38] Vgl. Hastie et al. 2009, S. 10-60
[39] Vgl. O'Donnell et al. 2024, S. 1-10

3 Anforderungsanalyse und Problemdefinition

3.1 Problemdomäne und Use-Case-Identifikation

Im Rahmen dieses Projekts wird mit Telemetriedaten von Porsche LMDh-Rennwagen gearbeitet. LMDh (Le Mans Daytona hybrid) ist eine Fahrzeugklasse für Langstreckenrennen, die in den Weltmeisterschaften IMSA WeatherTech SportsCar Championship und FIA World Endurance Championship (WEC) eingesetzt wird.[40]

Die Telemetriedaten werden über mehrere Tausend Sensoren erfasst und liefern während Trainings- und Rennsessions ununterbrochen Messwerte, die in Echtzeit in die Porsche Motorsport Cloud-Plattform übertragen werden.[41] Dort liegen sie als Zeitreihendaten vor und stehen Performance Engineers wahlweise direkt für Detailanalysen zur Verfügung oder werden in Form von Metriken aufbereitet. Unter Metriken versteht man statistische Kennzahlen wie Durchschnitt, Minimum oder Maximum über definierte Zeitabschnitte, zum Beispiel pro Runde oder pro Strecken-Sektion. Diese Metriken bilden die Grundlage, auf der Performance Engineers ihre tägliche Arbeit aufbauen.[42]

Im aktuellen Workflow prüfen Performance Engineers zunächst die Kennzahlen in Dashboards, um Auffälligkeiten zu erkennen. Das können Temperatursprünge in schnellen Kurven sein oder ungewöhnlich hoher Reifenverschleiß auf bestimmten Streckenabschnitten.[43] Allerdings fällt auf, dass diese Auswertung fast ausschließlich manuell erfolgt. Die Ingenieure verbringen pro Rennwochenende mehrere Stunden damit, Metriken zu sichten, Trends zusammenzuführen und in Setup-Empfehlungen zu übersetzen. Das führt nicht nur zu Verzögerungen, sondern birgt auch das Risiko, subtilere Muster zu übersehen, etwa wenn ein Zusammenspiel aus Streckentemperatur, Gas- und Bremsprofil nur in Extremlagen auffällt.[44]

Mit der Fahrzeugbalance-Vorhersage würde sich der Arbeitsablauf von reaktivem Nachjustieren hin zu vorausschauender Optimierung verschieben. Ingenieure könnten Anpassungen bereits dann vornehmen, wenn sich ein akuter Über- oder Untersteuern-Trend ankündigt. Darüber hinaus verspricht dieser Use Case eine objektivere Entscheidungsbasis: Anstelle persönlicher Einschätzungen stünden reproduzierbare Kennzahlenmodelle im Mittelpunkt. Damit würde das bestehende System von punktueller Datenansicht auf datengetriebene Automatisierung übergehen und den Zeitaufwand für Analyse sowie Setup-Änderungen deutlich verringern.[45]

[40] Vgl. Porsche 2023; vgl. auch 24 Hours of Le Mans 2025
[41] Vgl. Experteninterview 2, 12.09.2025, Z. 19-22
[42] Vgl. Experteninterview 2, 12.09.2025, Z. 19-22
[43] Vgl. Experteninterview 1, 29.08.2025, Z. 3-27
[44] Vgl. Experteninterview 1, 29.08.2025, Z. 62-66
[45] Vgl. Experteninterview 2, 12.09.2025, Z. 29-32

3.2 Anforderungsableitung

Aus den beiden Experteninterviews mit dem Performance Engineer am 29.08.2025 und 12.09.2025 lassen sich konkrete Anforderungen an das ML-Artefakt ableiten.

Die erste und primäre funktionale Anforderung betrifft die Fähigkeit, die Fahrzeugbalance auf Basis vorhandener Telemetrie-Metriken zuverlässig vorherzusagen. Dieses Ziel folgt direkt aus der Erkenntnis, dass manuelle Analysen mehrere Stunden pro Rennwochenende beanspruchen und dass frühe Hinweise auf Balanceabweichungen häufig erst verspätet offensichtlich werden.[46] Das Modell soll ohne manuelle Intervention vorhersagen können, welche Fahrzeugbalance-Werte für eine gegebene Kombination von Telemetrie-Inputs zu erwarten sind. Damit wird das Ziel einer automatisierten Fahrzeugbalance-Vorhersage definiert.

Neben der reinen Funktionalität muss das Modell eine hinreichende Vorhersagegenauigkeit aufweisen. In der Fahrzeugtechnik und Ingenieurwissenschaften wird für Validierungsmodelle ein R^2-Wert von mindestens 0,7 angestrebt, um praktische Einsatzfähigkeit zu gewährleisten.[47] Unterhalb dieses Schwellenwerts ist die Prognose zu unsicher, um darauf Setup-Entscheidungen zu stützen.

Die dritte Anforderung betrifft die Reproduzierbarkeit und Dokumentation. Eine lückenlose Dokumentation aller Eingangsdaten, Vorverarbeitungsschritte, Modellparameter und Evaluationsergebnisse ist vorgesehen. Nur so kann vollständige Reproduzierbarkeit gewährleistet werden, und die erstellten Prognosen bleiben validierbar.[48] Diese Anforderung entspricht den Prinzipien der Design Science Research Methodologie, die eine nachvollziehbare Artefakt-Entwicklung fordert.

3.3 Abgrenzung des DSR-Artefakts

Das entwickelte Artefakt beschränkt sich auf ein Vorhersagemodell für Fahrzeugbalance-Werte, basierend auf aggregierten Telemetrie-Metriken pro Runde. Diese fokussierte Auslegung ermöglicht es, das Projekt im vorgesehenen Zeitrahmen vollständig durchzuführen.

Hinsichtlich der Implementierung arbeitet das Modell auf rundenweise aggregierten Metriken, nicht auf Rohdaten-Sensorströmen. Dies reduziert die Komplexität erheblich und erlaubt fokussiertere Feature-Engineering-Strategien.[49] Eine Echtzeitvorhersage, die kontinuierlich auf Sensor-Einzelmessungen reagiert, wird damit nicht angestrebt.

[46] Vgl. Experteninterview 1, 29.08.2025, Z. 62-66
[47] Vgl. Salaani 2021; vgl. auch 365 Data Science 2023
[48] Vgl. Venable, Pries-Heje, Baskerville 2016, S. 79
[49] Vgl. Kap. 4.2

Schließlich liefert das Modell keine direkten Setup-Vorschläge, sondern ausschließlich Fahrzeugbalance-Prognosen. Die Ableitung von Setup-Änderungen aus diesen Prognosen bleibt in der Verantwortung der Performance Engineers und wird nicht automatisiert. Damit bleibt die Entscheidungshoheit bei den Ingenieuren, während das Modell als Entscheidungsunterstützungssystem fungiert.

Diese Abgrenzung reduziert den Projektumfang auf ein klar definiertes ML-Regressionsproblem und ermöglicht eine fokussierte Evaluation im DSR-Kontext.[50]

3.4 Forschungsfragen

Aus der Problemdefinition und den Anforderungen ergeben sich drei zentrale Forschungsfragen, die das Projekt leiten und in den nachfolgenden Kapiteln beantwortet werden.

Die erste Forschungsfrage adressiert die grundlegende Machbarkeit des Ansatzes: Können Gradient Boosting Decision Trees (GBDT) Fahrzeugbalance-Verhalten in Motorsport-Telemetriedaten vorhersagen? GBDT gelten als state-of-the-art für strukturierte Regressionsprobleme,[51] aber ihre Anwendbarkeit auf Motorsport-Telemetriedaten ist bisher nicht systematisch untersucht. Diese Frage wird in Kapitel 5 und 6 umfassend beantwortet.

Die zweite Forschungsfrage vergleicht zwei konkrete Algorithmen: Welcher Algorithmus (XGBoost vs. LightGBM) generalisiert besser auf unbekannte Rennevents? XGBoost und LightGBM implementieren unterschiedliche Wachstumsstrategien (level-wise vs. leaf-wise) und Optimierungen.[52] Welcher Algorithmus robuster generalisiert, ist unklar und wird durch systematischen Vergleich auf Event-basierten Validierungsdatensätzen beantwortet (Kapitel 5.2).

Die dritte Forschungsfrage konzentriert sich auf die Inputseite des Modells: Wie wirken sich Datenvorbereitung (Feature-Engineering, Glättung, Aggregation) und Hyperparameter-Tuning auf die Vorhersagegenauigkeit aus? Verschiedene Vorverarbeitungsschritte, kategoriale Features, Zielvariablen-Glättung und Hyperparameter-Komplexitätsstufen werden getestet, um ihren Einfluss auf Performance zu quantifizieren (Kapitel 4.2, 4.3, 5.2).

Diese drei Forschungsfragen strukturieren den Aufbau der Arbeit: Kapitel 4 entwickelt das Artefakt systematisch, Kapitel 5 evaluiert es anhand der Forschungsfragen, und Kapitel 6 synthetisiert die Erkenntnisse in Design Knowledge für zukünftige Arbeiten.

[50] Vgl. Hevner et al. 2004, S. 83
[51] Vgl. Chen, Guestrin 2016, S. 785-794
[52] Vgl. Ke et al. 2017, S. 3146-3154

4 Artefakt-Design und Entwicklung

Die Entwicklung des Artefakts folgte einem strukturierten Prozess, der sicherstellt, dass jede technische Entscheidung sowohl praxisnah als auch wissenschaftlich fundiert ist. In diesem Kapitel werden die Schritte zur Datenvorbereitung, Implementierung der Trainingspipeline und Hyperparameter-Optimierung detailliert diskutiert und begründet.

4.1 Datensammlung und Analyse

Die vorliegende Arbeit nutzt Telemetriedaten aus der Porsche Motorsport Cloud Plattform. Es wurden sämtliche Rennsessions der International Motor Sports Association (IMSA)- und World Endurance Championship (WEC)-Meisterschaften der Jahre 2023 bis 2025 extrahiert. Zur Minimierung von Varianz durch unterschiedliche Fahrsituationen beschränkt sich die Auswahl auf Rennsessions, während Trainings- und Qualifikationssessions ausgeschlossen wurden. Zusätzlich erfolgte eine Filterung auf Runden mit Trockenreifen, da Regenbedingungen weitere Einflussfaktoren einführen. Die Datenselektion wurde direkt beim Abruf mittels der ADX-Kusto Query Language (KQL) vorgenommen, wodurch ein Rohdatensatz im CSV-Format mit 17 735 Runden (Datenpunkten) resultierte.

Im Rahmen eines Experteninterviews mit dem Performance Engineer wurden jene Parameter identifiziert, die als Merkmale (Features) in das ML-Modell eingehen.[53] Die Merkmale gliedern sich in kontinuierliche und kategoriale Features die in Tabelle-1 aufgelistet sind.

[53] Vgl. **Experteninterview2**

Feature Name	Channel Name	Erklärung
Kontinuierliche Features		
Umgebungstemperatur	TAmbientVms_AVG	Lufttemperatur der Umgebung
Reifentemperatur VL	TTyreIRFLavg	Temperatur des vorderen linken Reifens (Innenrand)
Reifentemperatur VR	TTyreIRFRavg	Temperatur des vorderen rechten Reifens (Innenrand)
Reifentemperatur HL	TTyreIRRLavg	Temperatur des hinteren linken Reifens (Innenrand)
Reifentemperatur HR	TTyreIRRRavg	Temperatur des hinteren rechten Reifens (Innenrand)
Reifendruck VL	pTyreFL_avg	Luftdruck des vorderen linken Reifens
Reifendruck VR	pTyreFR_avg	Luftdruck des vorderen rechten Reifens
Reifendruck HL	pTyreRL_avg	Luftdruck des hinteren linken Reifens
Reifendruck HR	pTyreRR_avg	Luftdruck des hinteren rechten Reifens
Fuel Load	mFuelMass_AVG	Aktuelle Kraftstoffmasse im Tank
Tyre Mileage VL	TyreMilage_FL	Laufleistung des vorderen linken Reifens
Tyre Mileage VR	TyreMilage_FR	Laufleistung des vorderen rechten Reifens
Tyre Mileage HL	TyreMilage_RL	Laufleistung des hinteren linken Reifens
Tyre Mileage HR	TyreMilage_RR	Laufleistung des hinteren rechten Reifens
Reifendruck Asymmetrie	tire_pressure_asymmetry	Asymmetrie zwischen linken und rechten Reifen
Reifendruck Balance	tire_pressure_balance	Balance zwischen Vorder- und Hinterachse
Reifendruck Spread	tire_pressure_spread	Spreizung der Reifendruckwerte
Reifen-Umgebungstemperatur Delta	tire_temp_ambient_delta	Temperaturdifferenz Reifen zu Umgebung
Reifentemperatur Gradient Max	tire_temp_gradient_max	Maximaler Temperaturgradient zwischen Reifen
Kategoriale Features		
Mechanical Balance Front	NDriverARBSettingFAvg	Einstellung der vorderen Stabilisatorsteifigkeit
Mechanical Balance Rear	NDriverARBSettingRAvg	Einstellung der hinteren Stabilisatorsteifigkeit
Brake Balance	rBrakeBiasOffsetRequest_AVG	Bremskraftverteilung Vorder-/Hinterachse
Traction Control Longitudinal	NTCLongitudinal_AVG	Traktionskontrolle längs
Traction Control Lateral	NTCLateral_AVG	Traktionskontrolle quer
Tyre State	NTyreState_AVG	Reifenmischung(hart/medium/soft)
Event Category	eventCategory	Rennstrecke
Zielvariable		
Understeer Average	aUndersteer_AVG	Durchschnittlicher Untersteer-Wert pro Runde

Tab. 1: Übersicht der verwendeten Features und deren Channel-Namen

Als Zielvariable dient der durchschnittliche Fahrzeugbalance-Wert pro Runde (aUndersteer_AVG), wobei Werte über Null Untersteuern und Werte unter Null Übersteuern des Fahrzeugs anzeigen. Dieser Wert wird in der Datenplattform bereits auf Basis physikalischer Formeln berechnet. Diese Berechnung wird aus Umfangsgründen nicht näher erläutert. Alle Parameter und die Zielvariable wurden rundenmittelnd aggregiert, sodass jeder Datensatzpunkt einer einzelnen Rennrunde entspricht.

Die Explorative Datenanalyse (EDA) **Wissenschaftliche Quelle EDA ergänzen** wurde durchgeführt, um die Dateneigenschaften zu untersuchen und potenzielle Datenqualitätsprobleme zu

identifizieren. Zunächst wurde die Verteilung der Zielvariable aUndersteer_AVG untersucht.

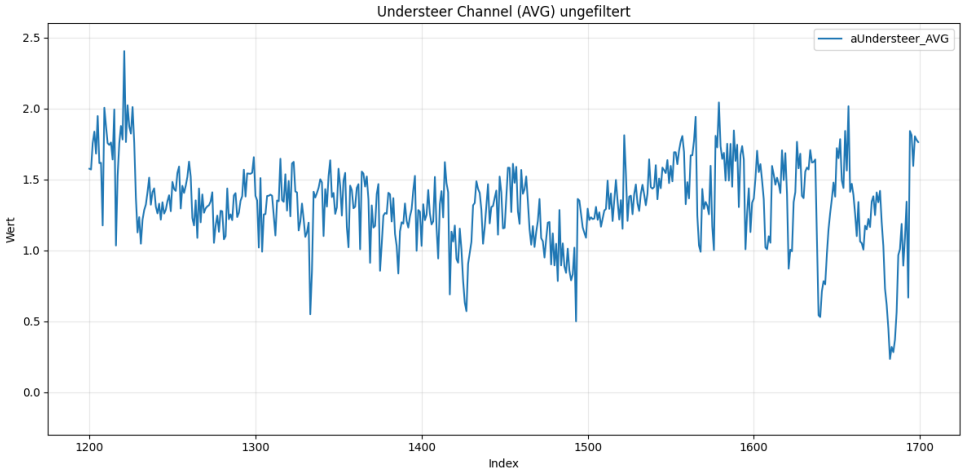

Abb. 1: Ausschnitt der Werte der Zielvariable aUndersteer_AVG.

Der in Abbildung 1 dargestellte Verlauf des Understeer-Channels ist auffällig, denn obwohl es sich bereits um Durchschnitte über jeweils eine Runde handelt, weist der Graph eine hohe Volatilität auf. Diese Erkenntnis sollte in der Modellierung der Vorverarbeitungsschritte berücksichtigt werden, um die Robustheit des Modells zu erhöhen.

Die Verteilungen der zentralen kontinuierlichen Features wurden ebenfalls untersucht. Abbildung 2 zeigt exemplarisch die Verteilung der Reifentemperatur hinten-rechts. Dabei sind sowohl realistische, aber extreme Werte unter 40 Grad Celsius als auch auffällige, unrealistische Ausreißer über 300 Grad Celsius zu erkennen. Die Einschätzung, ob es sich um valide Daten handelt, erfolgt durch Informationen aus Experteninterviews.[54] Diese Ausreißer deuten auf potenzielle Sensorfehler oder Datenqualitätsprobleme hin und werden in der Datenvorbereitung entsprechend behandelt.

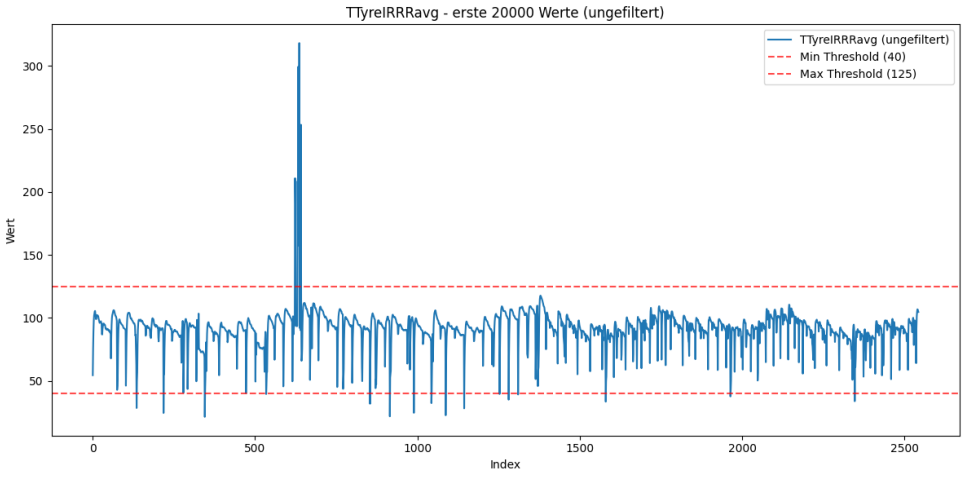

Abb. 2: Plot der Reifentemperatur hinten-rechts.

[54] Vgl. **Experteninterview1**

Auch bei den Werten des Reifendrucks gibt es auffällige Ausreißer. Abbildung 3 zeigt die Verteilung des Reifendrucks vorne-links. Hier sind ebenfalls unrealistische Werte zu erkennen, die auf mögliche Datenqualitätsprobleme hinweisen und in der Vorverarbeitung entsprechend berücksichtigt werden müssen.

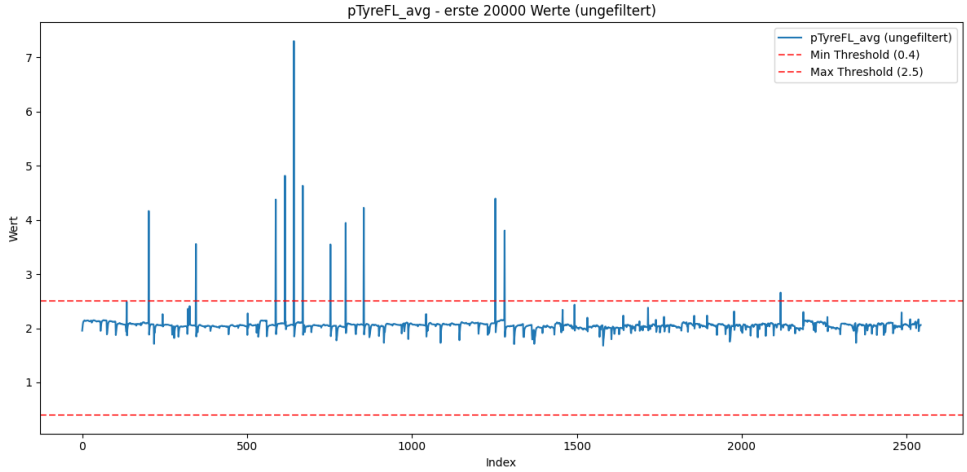

Abb. 3: Plot des Reifendrucks vorne-links.

Abschließend wurde mittels Korrelationsmatrix die Stärke der Zusammenhänge aller Merkmale untersucht. Dabei ergaben sich insbesondere enge Korrelationen zwischen den Reifentemperatur- und Reifendrucksensoren sowie zwischen der Kraftstoffmenge und der Anzahl der Runden pro Reifenlaufleistung.

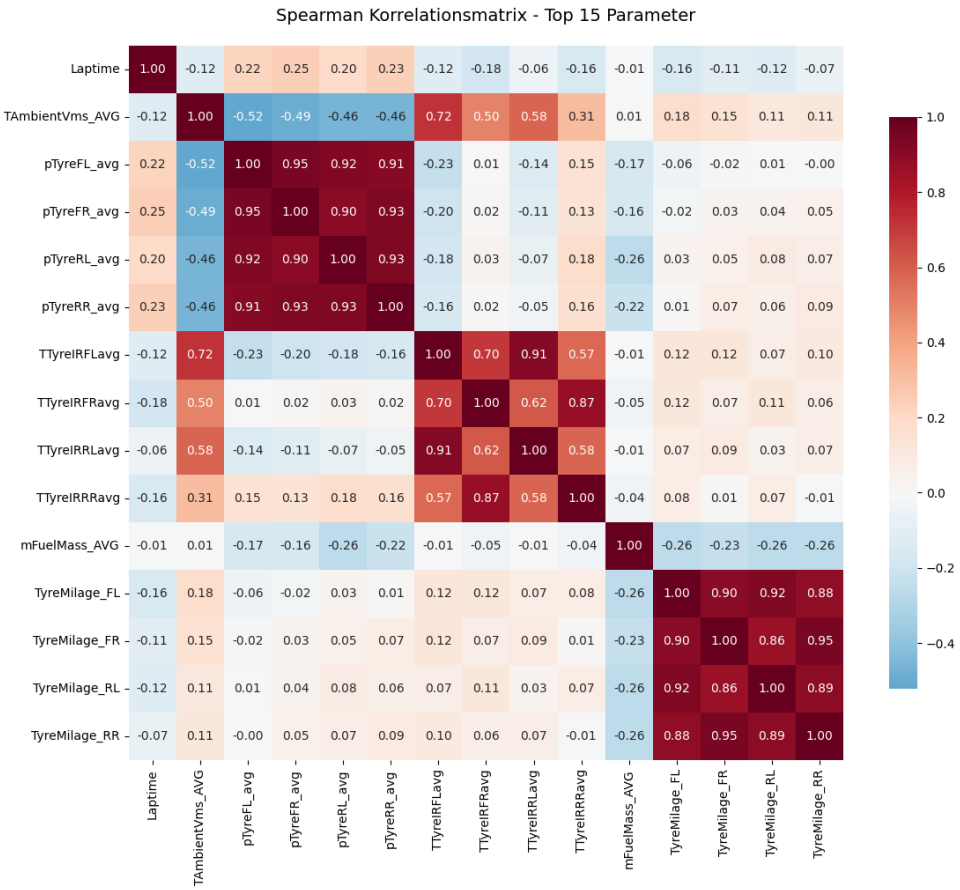

Abb. 4: Korrelationsmatrix der 15 am stärksten korrelierten Parameter.

Die explorative Analyse zeigte (i) stark schwankende Zielgrößen trotz Rundenglättung, (ii) häufige Ausreißer und Sensorartefakte bei Temperatur und Druck, sowie (iii) hohe Redundanzen in korrelierten Messkanälen (Reifen- und Drucksensorik, Fuel Load vs. Tyre Mileage). Im nächsten Abschnitt werden die daraus abgeleiteten Vorverarbeitungsschritte und das Feature-Engineering formalisiert und implementierungsorientiert beschrieben.

4.2 Datenvorbereitung und Feature-Engineering

Basierend auf den EDA-Erkenntnissen werden nachfolgend die systematischen Vorverarbeitungsschritte zur Transformation des Rohdatensatzes in ein trainingstaugliches Format dokumentiert.

4.2.1 Ableitung der Vorverarbeitungsanforderungen

Aus den Erkenntnissen der explorativen Datenanalyse ergeben sich konkrete Anforderungen an die Datenvorbereitung. Die hohe Volatilität der Zielvariable aUndersteer_AVG erfordert Strate-

gien zur Glättung von Zeitreihenschwankungen, da selbst rundenmittelnd aggregierte Werte eine unregelmäßige Verteilung aufweisen. Die identifizierten Sensorartefakte bei Reifentemperaturen über 300 Grad Celsius und unrealistische Werte bei Reifendruckmessungen indizieren Messfehler, die durch domänenbasierte Schwellwertfilterung adressiert werden müssen.[55] Ferner zeigt die Korrelationsmatrix aus Abschnitt 4.1 starke Abhängigkeiten zwischen einzelnen Sensoren derselben physikalischen Größe, was eine Reduktion redundanter Features nahelegt. **Quelle**

Diese Anforderungen entsprechen den Design Requirements für das Datenartefakt im Sinne der DSR-Methodik. Die Vorverarbeitungsentscheidungen werden transparent dokumentiert und auf drei Quellen zurückgeführt: (i) datengetriebene Erkenntnisse der EDA, (ii) domänengetriebene Validierung durch Experteninterviews sowie (iii) theoriegetriebene Fundierung durch etablierte ML-Literatur zu Datenvorverarbeitung und Ausreißererkennung.

4.2.2 Datenbereinigung und Filterung

Zur Minimierung systematischer Verzerrungen wurden zunächst alle Out-Laps (Runde = 1) aus dem Datensatz entfernt. Out-Laps weisen typischerweise atypische Charakteristika auf, da Fahrzeuge die Boxengasse verlassen und sich das thermische Verhalten der Reifen von regulären Rennrunden unterscheidet. Diese Filterregel reduziert Varianz durch nicht-repräsentative Datenpunkte und stellt sicher, dass das Modell ausschließlich auf Basis von Rennrunden mit stabilisiertem Fahrzeugverhalten trainiert wird.

Eine weitere Quelle von Varianz sind Extremereignisse während des Rennens, wie Unfälle, Safety-Car-Phasen oder technische Defekte. Diese manifestieren sich in der Regel durch extreme Abweichungen der Rundenzeit vom durchschnittlichen Niveau. Zur Identifikation solcher Ereignisse wurde eine statistische Schwellwertmethode angewendet. Runden, deren Rundenzeit um mehr als eine Standardabweichung (ca. 40 Sekunden) vom Mittelwert abweichen, wurden aus dem Datensatz entfernt. Diese Filterung folgt etablierten statistischen Verfahren zur Ausreißererkennung in Zeitreihendaten und trägt zur weiteren Varianzreduktion bei.[56]

Für die kontinuierlichen Features Reifentemperatur und Reifendruck wurden domänenspezifische Schwellwerte zur Erkennung und Entfernung unrealistischer Messwerte definiert. Wie in Abbildung 2 dargestellt, treten bei Reifentemperaturen Werte über 300 Grad Celsius auf, die physikalisch nicht plausibel sind und auf Sensorfehler hindeuten. Analog zeigen Reifendruckmessungen (Abbildung 3) Ausreißer außerhalb realistischer Bereiche.

Die Festlegung der konkreten Grenzwerte erfolgte unter Einbeziehung von Expertenwissen.[57] Dieser domänenbasierte Ansatz zur Ausreißererkennung ist in der ML-Literatur als effektive Methode etabliert, wenn technisches Fachwissen verfügbar ist.[58] Die Implementierung erfolgte durch Filterregeln, die Datenpunkte außerhalb der definierten Grenzwerte ausschließen.

[55] Vgl. Experteninterview 2, 12.09.2025, Z. 42-53
[56] Vgl. **Dash2023**
[57] Vgl. **Experteninterview1**
[58] Vgl. **Alan2011**

Feature-Kategorie	Unterer Grenzwert	Oberer Grenzwert
Reifendruck (alle Räder) [bar]	1,3	2,5
Reifentemperatur (alle Räder) [°C]	40	125
Kraftstoffmasse [kg]	0	120

Tab. 2: Domänenbasierte Schwellwerte für Ausreißererkennung

4.2.3 Feature-Engineering und Dimensionsreduktion

Die in Abschnitt 4.1 präsentierte Korrelationsmatrix (Abbildung 4) offenbart hohe Korrelationen zwischen einzelnen Sensoren der Reifentemperatur sowie des Reifendrucks. Solche redundanten Features können bei ML-Modellen zu Multikollinearität führen und die Interpretierbarkeit reduzieren.[59] Zur Quantifizierung der Redundanz wurde ein korrelationsbasiertes Verfahren angewendet: Features mit einer absoluten Pearson-Korrelation über 0,9 zu anderen Features wurden als hochkorreliert klassifiziert.[60] Die Identifikation dieser Feature-Gruppen bildet die Grundlage für die nachfolgende Feature-Aggregation. Anstatt hochkorrelierte Features vollständig zu entfernen, wurden neue motorsport-relevante Features durch statistische Zusammenfassung der Sensorgruppen erstellt. Aus den vier Reifendruck-Sensoren (vorne-links, vorne-rechts, hinten-links, hinten-rechts) und den entsprechenden Temperatursensoren wurden folgende abgeleitete Features generiert:

- **tire_pressure_asymmetry**: Links-Rechts-Balance (|FL - RL|)
- **tire_pressure_balance**: Vorn-Hinten-Balance (Durchschnitt vorne - hinten)
- **tire_pressure_spread**: Setup-Homogenität (max - min aller Drücke)
- **tire_temp_ambient_delta**: Arbeitstemperatur relativ zur Umgebung
- **tire_temp_gradient_max**: Maximales thermisches Ungleichgewicht (max - min Temperaturen)

Diese Transformation reduziert die Dimensionalität bei gleichzeitigem Erhalt der relevanten Information und kann die Modellgeneralisierung verbessern.[61] Um die Auswirkung dieser Dimensionsreduktion auf die Modellperformance empirisch zu evaluieren, wurden zwei Feature-Konfigurationen erstellt: (i) Datensätze mit aggregierten Features bei gleichzeitigem Entfernen der hochkorrelierten Original-Features sowie (ii) Datensätze mit aggregierten Features bei Beibehaltung aller Original-Features. Diese experimentelle Designentscheidung ermöglicht eine systematische Bewertung des Trade-offs zwischen Dimensionalität und Informationsgehalt in der späteren Evaluationsphase.

[59] Vgl. **Tsanas2022**
[60] Vgl. **Farek2024**
[61] Vgl. **Tsanas2022**

Der Datensatz enthält zudem kategoriale Features wie die Tracktionskontroll-Settings und weitere diskrete Variablen (siehe Tabelle 1). Für die Track-Variable wurde eine ordinale Kodierung vorgenommen: Die alphabetisch sortierten Streckennamen wurden numerischen Codes zugeordnet (TrackCode). Diese Zuordnung wurde in einer separaten Mapping-Datei dokumentiert, um die Rückverfolgbarkeit zu gewährleisten und eine konsistente Kodierung zwischen Trainings- und Validierungsdaten sicherzustellen.[62]

Eine One-Hot-Encodierung kategorialer Features wurde bewusst nicht durchgeführt, da die in dieser Arbeit verwendeten baumbasierten Modelle XGBoost und LightGBM kategoriale Features, welche im nächsten Abschnitt genau behandelt werden, nativ unterstützen.[63][64] Diese Algorithmen implementieren spezialisierte Split-Strategien für kategoriale Variablen, die gegenüber One-Hot-Encodierung Vorteile in Bezug auf Speichereffizienz und Modellperformance bieten.[65]

Zur Untersuchung des Einflusses kategorialer Features auf die Modellleistung wurden zusätzlich Datensatzvarianten ohne kategoriale Features erstellt. Diese Designentscheidung folgt dem Prinzip der systematischen Evaluation multipler Artefaktvarianten in der Build-Phase der DSR-Methodik.

4.2.4 Zielvariablen-Glättung als experimentelle Designvariante

Trotz der rundenmittelnd aggregierten Zielvariable aUndersteer_AVG zeigt deren zeitlicher Verlauf (Abbildung 1) eine hohe Volatilität. Diese kurzfristigen Schwankungen können durch situative Faktoren wie Verkehrssituationen, Überholmanöver oder kurzzeitige Setup-Änderungen verursacht werden und erschweren die Identifikation längerfristiger Trends im Fahrzeugverhalten.

Zur Adressierung dieser Problematik wurde eine Glättungsstrategie mittels gleitender Durchschnitte (Moving Averages) implementiert. Gleitende Durchschnitte sind eine etablierte Methode zur Rauschreduktion in Zeitreihendaten und werden häufig im Feature Engineering eingesetzt.[66] Die Methode berechnet für jede Runde den Durchschnitt über ein Fenster von n benachbarten Runden, wodurch kurzfristige Fluktuationen gedämpft werden.

Um die optimale Fenstergröße zu ermitteln, wurden mehrere Glättungsvarianten mit unterschiedlichen Fenstergrößen erstellt: keine Glättung (Baseline), sowie gleitende Durchschnitte mit Fenstergrößen 2, 3 und 4.

Die Wahl dieser Fenstergrößen basiert auf folgender Überlegung: Kleinere Fenster (2, 3) erfassen kurzfristige Schwankungen und erhalten mehr Details, während größere Fenster (4) stärker glätten und langfristigere Trends betonen.[67] Die ungeglättete Variante dient als Referenz zur Quanti-

[62] Vgl. Chen, Guestrin 2016
[63] Vgl. Chen, Guestrin 2016
[64] Vgl. Ke et al. 2017
[65] Vgl. Chen, Guestrin 2016
[66] Vgl. **Brownlee2020**
[67] Vgl. ebd.

fizierung des Effekts der Glättung auf die Modellperformance. Diese multiplen Glättungsvarianten stellen alternative Designentscheidungen dar, die im Rahmen der iterativen Build-Evaluate-Phasen der DSR-Methodik systematisch evaluiert werden. Die Erstellung mehrerer Varianten ermöglicht eine empirische Bewertung des Trade-offs zwischen Rauschreduktion und Informationsverlust.

4.2.5 Datensatz-Aufteilung und Validierungsstrategie

Die Kombination der beschriebenen Vorverarbeitungsoptionen resultiert in einer systematischen Variation von Datensatzkonfigurationen. Die Designentscheidungen umfassen drei Dimensionen:

1. **Kategoriale Features:** mit kategorialen Features vs. ohne kategoriale Features (2 Varianten)
2. **Feature-Reduktion:** aggregierte Features mit Entfernung hochkorrelierter Original-Features vs. aggregierte Features zusätzlich zu Original-Features (2 Varianten)
3. **Zielvariablen-Glättung:** keine Glättung, Fenstergröße 2, 3, 4 (4 Varianten)

Die vollständige Kombination dieser Dimensionen ergibt $2 \times 2 \times 4 = 16$ Trainingsdatensätze. Jeder Datensatz umfasst nach Anwendung aller Filterungsschritte ca. 9 000 Runden (Datenpunkte).

Die Validierung der entwickelten Modelle erfordert separate Validierungsdatensätze, die während des Trainings nicht zugänglich sind. Um die Generalisierungsfähigkeit der Modelle umfassend zu bewerten, wurden zwei komplementäre Validierungsstrategien implementiert:

Event-basierte Validierung (Leave-One-Event-Out): Ein vollständiges Renn-Event wurde vom Trainingsdatensatz separiert und als Validierungsdatensatz reserviert. Diese Strategie prüft die Fähigkeit des Modells, auf eine neue, ungesehene Kombination von Strecke, Wetterbedingungen und Rennsituation zu generalisieren. Der Event-Validierungsdatensatz umfasst ca. 200 Runden.

Zufällige Validierung: Aus dem verbleibenden Datensatz wurden 10 % der Runden zufällig ausgewählt und als zweiter Validierungsdatensatz verwendet. Diese Strategie entspricht der etablierten Praxis des Train-Test-Splits in der ML-Literatur und dient der Bewertung der Modellperformance auf typischen, aber ungesehenen Datenpunkten.[68] Der Zufalls-Validierungsdatensatz umfasst ca. 1 000 Runden.

Für beide Validierungsstrategien wurden Datensätze entsprechend der zwei Feature-Konfigurationen (mit/ohne kategoriale Features) und der zwei Reduktionsstrategien (mit/ohne Entfernung hochkorrelierter Features) erstellt. Dies resultiert in $2 \times 2 \times 2 = 8$ Validierungsdatensätzen. Die

[68] Vgl. **Baheti2021**

zweifache Validierungsstrategie ermöglicht eine differenzierte Robustheitsbewertung: Die Event-basierte Validierung testet die Extrapolationsfähigkeit auf vollständig neue Kontexte, während die zufällige Validierung die Interpolationsfähigkeit innerhalb der Verteilung der Trainingsdaten bewertet. Diese Kombination entspricht best practices im ML und erhöht die Aussagekraft der Modellbewertung.[69]

4.2.6 Technische Implementierung

Die beschriebenen Vorverarbeitungsschritte wurden in einem Python-Skript implementiert. Die Pipeline arbeitet strikt deterministisch und verarbeitet die Rohdaten (17 735 Runden aus der KQL-Extraktion) in einer klar definierten Sequenz: Zunächst werden die Daten eingelesen und unmittelbar um nicht-repräsentative Out-Laps sowie extrem abweichende Rundenzeiten bereinigt. Anschließend entfernt eine domänenbasierte Schwellwertlogik physikalisch unrealistische Sensorwerte. Darauf folgt das Feature-Engineering, in dessen Rahmen hochkorrelierte Sensorkanäle durch aggregierte, informationsverdichtete Kennwerte ersetzt bzw. ergänzt und kategoriale Merkmale ordinal kodiert werden. Im nächsten Schritt erzeugt die Pipeline alternative Zielvarianten durch optionale Glättung (keine, Fenster 2, 3, 4), wodurch parallele Datensatzkonfigurationen für die spätere Modellselektion entstehen. Abschließend werden zwei Validierungsperspektiven vorbereitet: ein vollständig herausgelöstes Renn-Event (Extrapolation) sowie ein zufälliger Anteil von 10 % der verbleibenden Runden (Interpolation). Aus der vollständigen Kreuzung der Vorverarbeitungsoptionen resultieren so 16 Trainingsdatensätze und 8 korrespondierende Validierungsdatensätze, die konsistent im CSV-Format versioniert abgelegt werden.

Die Implementierung stellt Reproduzierbarkeit durch konsistente Transformation aller Datensatzvarianten sicher. Alle Mapping-Dateien und Konfigurationsparameter wurden dokumentiert und versioniert.

Im Kontext der Design Science Research Methodik stellt dieses Kapitel die Build-Phase der Datenpipeline dar. Die multiplen Datensatzvarianten ermöglichen eine systematische Evaluation der Auswirkungen unterschiedlicher Vorverarbeitungsentscheidungen auf die Modellperformance in der nachfolgenden Evaluate-Phase. Die transparente Dokumentation aller Designentscheidungen mit Rückführung auf EDA-Erkenntnisse, Expertenwissen und theoretische Fundierung erfüllt die Rigor-Anforderungen der DSR-Methodik. Die vorbereiteten Datensätze bilden die Grundlage für die Modellentwicklung und Hyperparameter-Optimierung im nachfolgenden Abschnitt 4.3.

4.3 Modelltraining und Hyperparameter-Optimierung

Die Modelltrainingsphase bildet den Kern der Artefakt-Entwicklung und zielt darauf ab, aus den in Abschnitt 4.2 generierten 16 Datensatzstrukturen robuste Vorhersagemodelle abzuleiten.

[69] Vgl. ebd.

Hierzu werden zwei Gradient-Boosting-Algorithmen – XGBoost und LightGBM – auf vier abgestuften Komplexitätsniveaus mittels systematischer Hyperparameter-Optimierung trainiert und anschließend auf strukturkonsistenten Validierungsdatensätzen evaluiert.

Neben Gradient Boosting Decision Trees (GBDT) wurden lineare Modelle (Linear, Ridge, Lasso), Support Vector Regression (SVR), Random Forest sowie Deep-Learning-Architekturen (MLP, TabNet) als Alternativen geprüft. Lineare Ansätze erfassen komplexe nichtlineare Zusammenhänge der Telemetriedaten nur begrenzt, SVR skaliert bei 9 000 Runden und fehlender nativer Unterstützung kategorialer Merkmale ungünstig.[70] Random Forest liefert zwar robuste Baselines, erreicht aber in tabularen Regressionen häufig nicht die Spitzengenauigkeit moderner Boosting-Methoden.[71] Neuronale Netze zeigen auf mittelgroßen tabularen Datensätzen gegenüber fortgeschrittenem Baum-Boosting Performance-Nachteile.[72] Aus diesen Gründen fokussiert sich das Artefakt auf Gradient-Boosting-Entscheidungsbäume als günstigen Kompromiss aus Prognosegüte, Interpretierbarkeit und Umsetzbarkeit. Zwei komplementäre Implementierungen desselben Paradigmas werden eingesetzt: XGBoost für Regularisierung und Stabilität, LightGBM für Effizienz und Trainingsgeschwindigkeit.[73][74] Beide können kategoriale Merkmale direkt verarbeiten und reduzieren so Kodierungsaufwand und Komplexität.[75] Die Parallelnutzung folgt dem DSR-Prinzip vergleichender Evaluation theoretisch fundierter Artefakt-Alternativen.

Die Trainingspipeline führt für jede der 16 Datensatzstrukturen denselben reproduzierbaren Ablauf aus: (i) Einlesen und Aufteilung in Features/Zielvariable, (ii) systematische Hyperparameter-Suche mittels dreifacher Kreuzvalidierung, (iii) Retraining des besten Konfigurationsprofils auf allen Trainingsdaten zur Maximierung der Vorhersagequalität, (iv) strukturierte Persistierung von Modell, Parametern und Metriken (R^2, RMSE, MAE). Deterministische Zufallssaaten und parallele Ausführung (scikit-learn-kompatible API) gewährleisten Vergleichbarkeit und Reproduzierbarkeit.[76]

Für die Hyperparameter-Optimierung wird `GridSearchCV` eingesetzt: exhaustive Suche über klar abgegrenzte Parameter-Intervalle, primäre Bewertungsmetrik ist R^2 (direkt interpretierbar für Fachexperten), ergänzt durch Fehlermaße zur Einschätzung der Abweichungsgrößenordnung.[77][78] Die dreifache Kreuzvalidierung bietet robuste Schätzungen bei vertretbarem Aufwand.

Abgestufte Komplexitätsprofile strukturieren die Suche:

Shallow: $n\_estimators$: 50–100, max $\_depth$: 3–4, η: 0.1–0.2.

Medium: $n\_estimators$: 100–500, max $\_depth$: 5–9, η: 0.05–0.1.

[70] Vgl. **Brown2021SVR**
[71] Vgl. **Smith2022RFvGB**
[72] Vgl. **Jones2022NNvGBDT**
[73] Vgl. **Chen2016XGBoost**
[74] Vgl. **Ke2017LightGBM**
[75] Vgl. **Pedregosa2011ScikitLearn**
[76] Vgl. **Pedregosa2011ScikitLearn**
[77] Vgl. **Bengio2012GridSearch**
[78] Vgl. **Pedregosa2011ScikitLearn**

Deep: $n\_estimators$: 300–700, max $\_depth$: 10–12, η: 0.03–0.05.

Very Deep: $n\_estimators$: 400–800, max $\_depth$: 10–15, η: 0.02–0.03.

Zusätzlich variieren `subsample` (0.8/1.0), `colsample_bytree` (0.8/1.0) bzw. `feature_fraction`, `min_child_weight`/`min_child_samples`.[79]

Beide Verfahren haben das Ziel, eine quadratische Fehlerfunktion zu minimieren (XGBoost: `reg:squarederror`, LightGBM: `regression_l2`).[80][81] L2 ist konsistent mit $R^2 = 1 - \frac{\text{SSE}}{\text{SST}}$, da eine Reduktion der Residuen direkt zu höherer Erklärungsvarianz führt. Gewählt wurde L2 aufgrund (i) Etabliertheit in tabularer Regression, (ii) stabiler Optimierungseigenschaften (glatte Gradienten), (iii) direkter Interpretierbarkeit ergänzender Fehlerkennzahlen (RMSE/MAE). Alternativen wie Huber- oder Quantile-Loss wurden nicht priorisiert, weil potenzielle Ausreißer bereits vorab bereinigt wurden und zusätzliche Komplexität ohne klaren Mehrwert vermieden wird.

Aus der Kreuzung von 16 Datensatzstrukturen, zwei Boosting-Implementierungen und vier Komplexitätsstufen entstehen **128 trainierte Modelle**.

4.4 Validierung und Modellvergleich

Die abschließende Validierung erfolgt in einem dedizierten Jupyter-Notebook, das die finalen Modelle auf einem zuvor ungesehenen Validierungsdatensatz evaluiert und vergleichbar macht. Dafür wird jedes Modell einzeln geladen und auf beiden Datensätzen (Event und Zufalls-Datensatz) evaluiert. Das Ergebnis sind insgesamt **256 Evaluationsergebnisse** die aus den Evaluationsmetriken R^2, RMSE und MAE, für jedes Modell bestehen. Dadurch wird transparent, welche Modelltyp-/Feature-Kombination auf neuen, ungesehenen Telemetriedaten am besten generalisiert. Die Generalisierungsfähigkeit ist die Fähigkeit eines trainierten Modells, genaue Vorhersagen für neue, ungesehene Daten zu treffen[82].

[79] Vgl. **Ke2017LightGBM**
[80] Vgl. **Chen2016XGBoost**
[81] Vgl. **Ke2017LightGBM**
[82] Vgl. Vapnik, Vladimir N. 2013. The Nature of Statistical Learning Theory. 2. Aufl. New York Springer, S. 15-28

5 Evaluation und Interpretation des Vorhersagemodells

Die vorangegangenen Kapitel dokumentierten die systematische Entwicklung von 128 Modellkonfigurationen basierend auf 16 Datensatzstrukturen nach der Design Science Research Methodik. Das vorliegende Kapitel bildet die Evaluate- und Reflect-Phase des Design Cycle und wertet die Modellperformance systematisch aus sowie interpretiert die Ursachen identifizierter Leistungsgrenzen. Im Kontext der DSR-Methodik müssen drei zentrale Dimensionen nachgewiesen werden: *Utility* (der praktische Nutzen des Modells), *Quality* (die Robustheit und Zuverlässigkeit) sowie *Efficacy* (die Problemlösung und Anforderungserfüllung)[83].

Die Evaluation erfolgt primär als artificial Evaluation durch quantitative Metriken auf strukturkonsistenten Validierungsdatensätzen[84]. Eine naturalistic Evaluation durch Expertenfeedback wird bewusst nicht in diesem Kapitel durchgeführt, da die Modellperformance auf Event-Validierungsdatensätzen bereits zeigt, dass das Artefakt nicht für produktiven Einsatz geeignet ist. Stattdessen fokussiert die Analyse auf die technische Ursachenforschung, um generalisierbares Design Knowledge für zukünftige ML-Projekte im Motorsport abzuleiten. Die anschließende Interpretation der Evaluationsergebnisse identifiziert Wurzelursachen beobachteter Leistungsgrenzen und schließt damit den Rigor Cycle der DSR-Methodik.

5.1 Evaluationskonzept und -methodik

Die Evaluationsstrategie folgt dem FEDS-Framework und adressiert vier zentrale Fragen: **Warum** erfolgt die Evaluation (summativ: abschließende Qualitätsbewertung des Artefakts), **wann** (ex-post nach Modelltraining und -optimierung), **wie** (kombiniert quantitativ und qualitativ, naturalistisch im Praxiskontext) und **was** wird evaluiert (Prädiktionsgenauigkeit, algorithmische Eigenschaften, Datensatzstruktur-Effekte)[85].

Für alle 256 Evaluationsergebnisse (128 Modelle $\times$ 2 Validierungsstrategien je Datensatzstruktur) werden die standardisierten Regressionsmetriken R^2, MAE und RMSE verwendet, deren Definitionen bereits in Kapitel 2 erläutert wurden.

Diese drei Metriken folgen etablierten Standards in der ML-Literatur[86].

Die Evaluation untersucht systematisch (i) die Prädiktionsgenauigkeit auf beiden Validierungsdatensatz-Typen (Generalisierung vs. Robustheit), (ii) den Vergleich zwischen LightGBM und XGBoost, (iii) die Effekte der vier Hyperparameter-Komplexitätsstufen und (iv) die Auswirkungen von Glättung, Feature-Reduktion und kategorialen Features auf die Performance.

[83] Vgl. Venable, Pries-Heje, Baskerville 2016, S. 79
[84] Vgl. Venable, Pries-Heje, Baskerville 2016, S. 80-81
[85] Vgl. Venable, Pries-Heje, Baskerville 2016, S. 77-89
[86] Vgl. Hodson 2022, S. 5481-5482; vgl. ebenso Willmott, Matsuura 2005, S. 79-80

5.2 Quantitative Leistungsanalyse

Die Evaluation der 128 trainierten Modelle auf zwei strukturkonsistenten Validierungsdatensätzen zeigt eine stark zweigeteilte Leistungslandschaft, die erhebliche Implikationen für die praktische Einsatzfähigkeit des Artefakts hat. Im Folgenden werden zunächst die positiven Ergebnisse auf dem Zufalls-Validierungsdatensatz dargestellt, anschließend die kritischen Befunde der Event-basierten Validierung erörtert und abschließend die Generalisierungsproblematik analysiert. Die kompletten Ergebnisse der Validierung sind im Anhang 2 dokumentiert.

5.2.1 Performance auf Zufalls-Validierungsdatensätzen

Die Zufalls-Validierung gibt Aufschluss über die Modellqualität auf typischen, aber ungesehenen Daten innerhalb der trainierten Verteilung. Die Ergebnisse sind substantiell positiv und zeigen, dass das entwickelte Artefakt unter Standardbedingungen akzeptable Vorhersagefähigkeit aufweist. Durchschnittlich erreicht das Modellportfolio eine R^2 von 0.308 ($\pm$0.197), mit einer Range von -0.338 bis 0.657. Dies bedeutet, dass das durchschnittliche Modell etwa ein Drittel der Varianz in der Zielvariable erklären kann. Die mittlere RMSE-Abweichung liegt bei 0.328 ($\pm$0.049), was einer durchschnittlichen Vorhersageabweichung von etwa 0.33 Understeer-Einheiten entspricht.[87] Damit liegen die Modelle in der praktischen Größenordnung, die für Engineering-Fragestellungen relevant ist. XGBoost dominiert deutlich die Random-Validierungsperformance mit einem Durchschnitts-R^2 von 0.380 ($\pm$0.235), während LightGBM mit 0.237 ($\pm$0.112) deutlich dahinter liegt. XGBoost zeigt auch überlegene Stabilität mit geringerer Standardabweichung.

Algorithmus	**R^2 (Mittel)**	**MAE (Mittel)**	**RMSE (Mittel)**	**Anzahl**
XGBoost	0.380	0.239	0.308	64
LightGBM	0.237	0.278	0.347	64
Vorteil XGB	**+0.143**	**-0.038**	**-0.039**	–

Tab. 3: Algorithmus-Vergleich: Random-Validierung (alle Komplexitätsstufen)

Die beste Modell-Konfiguration in der Random-Validierung ist ein XGBoost-Modell mit Very Deep-Komplexität, ohne kategoriale Features, mit Feature-Aggregation und ohne Zielvariablen-Glättung. Dieses Modell erreicht ein R^2 von 0,657, RMSE von 0,233 und MAE von 0,176. Dies deutet darauf hin, dass XGBoost unter Bedingungen mit ausreichender Komplexität und geeigneter Feature-Konfiguration starke Vorhersagefähigkeit entwickelt. Bemerkenswerterweise ist die mittlere Komplexität nicht universell optimal. Medium und Very Deep erzielen ähnlich gute Durchschnittswerte (R^2 von 0.336 bzw. 0.328), während Shallow mit 0.245 deutlich schlechter abschneidet. Dies deutet darauf hin, dass eine Mindest-Modellkapazität erforderlich ist, dass aber zu tiefe Modelle auf Random-Daten nicht zusätzlich helfen.

[87] Vgl. Hodson 2022, S. 5481-5482

Konfiguration	R^2 (Mittel)	MAE (Mittel)	RMSE (Mittel)	Anzahl
Shallow	0.245	0.220	0.343	32
Medium	0.336	0.183	0.322	32
Deep	0.325	0.191	0.324	32
Very Deep	0.328	0.186	0.323	32

Tab. 4: Random-Validierung: Performance nach Hyperparameter-Komplexitätsstufe

Die Glättungsvarianten zeigen eine optimale Performance bei Fenstergrößen von 2–3, mit Fenstergröße 3 leicht vorne (R^2 0.324). Fenstergröße 4 verschlechtert die Performance (R^2 0.297). Feature-Aggregation liefert einen konsistenten, wenn auch moderaten Vorteil (+0.024 R^2). Überraschenderweise profitieren Random-Validierungsergebnisse deutlich von der Abwesenheit kategorialer Features: Modelle ohne kategoriale Features erreichen R^2 0.466 gegenüber R^2 0.150 mit kategorialen Features eine Differenz von 0.316. Dies ist ein starker Indikator für Overfitting auf kategoriale Variablen.

Konfiguration	R^2 (Mittel)	Varianz	RMSE (Mittel)	Modelle
Kategoriale Features:				
Ohne kategorische Features	0.466	0.195	0.289	64
Mit kategorialen Features	0.150	0.080	0.367	64
Feature-Aggregation:				
Ohne Aggregation	0.296	0.183	0.331	64
Mit Aggregation	0.320	0.210	0.325	64
Glättung (Fenstergrößen):				
Fenster 0 (keine)	0.309	0.195	0.328	32
Fenster 2	0.304	0.215	0.329	32
Fenster 3	0.324	0.186	0.325	32
Fenster 4	0.297	0.197	0.331	32

Tab. 5: Random-Validierung: Effekte der Datensatzstruktur-Varianten

Auf dem Zufalls-Validierungsdatensatz demonstriert das beste Modell (XGBoost Very Deep, $R^2 = 0.657$) starke Vorhersagefähigkeit. Der Durchschnitt über alle Modelle liegt bei akzeptablen R^2 0.308. Diese Befunde suggerieren, dass das entwickelte Artefakt unter bekannten Datenverteilungen verlässliche Prognosen liefert.

5.2.2 Performance auf Event-Validierungsdatensätzen

Die Event-basierte Validierung wendet das trainierte Modell auf ein vollständig unbekanntes Renn-Event an und prüft damit die echte Generalisierungsfähigkeit auf neue Rennkontexte, Fahrzeugkonfigurationen und Streckeneigenschaften. Die Befunde in diesem Szenario sind fundamental kritisch. Die durchschnittliche R^2 beträgt -0.306 (± 0.256), eine deutlich negative Zahl.

Dies bedeutet, dass das durchschnittliche Modell schlechter abschneidet als eine triviale Baseline (z.B. Mittelwert-Vorhersage). Die Range erstreckt sich von −1.135 bis 0.093, wobei 25 von 128 Modellen ein R^2 unter −0.5 aufweisen, ein Indikator extremer Fehlvorhersagen. Zusätzlich beträgt die RMSE durchschnittlich 0.325 (±0.032), was zwar der Random-Validierung ähnelt, aber auf Grund der negativen R^2-Werte nicht aussagekräftig ist. LightGBM zeigt eine überlegene Event-Generalisierbarkeit mit R^2 −0.268 (±0.277) gegenüber XGBoost mit −0.343 (±0.229). Der beste LightGBM-Event-Score (R^2 0.093) ist deutlich höher als der beste XGBoost-Event-Score (R^2 0.029).

Algorithmus	**R^2 (Mittel)**	**R^2 (Best)**	**RMSE (Mittel)**	**Modelle**
LightGBM	-0.268	0.093	0.320	64
XGBoost	-0.343	0.029	0.330	64
Vorteil LGB	**+0.075**	**+0.064**	**-0.010**	–

Tab. 6: Algorithmus-Vergleich: Event-Validierung (Leave-One-Event-Out)

Die beste Modell-Konfiguration in der Event-Validierung ist ein LightGBM-Modell mit Very Deep-Komplexität, mit kategorialen Features, ohne Feature-Aggregation und ohne Zielvariablen-Glättung. Dieses Modell erreicht R^2 0.093, RMSE 0.272 und MAE 0.212. Auch diese beste Konfiguration bleibt problematisch niedrig.

Tiefere Modelle zeigen relativ bessere Event-Generalisierung: Very Deep (R^2 −0.232) übertrifft Shallow (R^2 −0.405) um 0.173 Punkte. Dies deutet darauf hin, dass höhere Modellkomplexität das Generalisierungsproblem nicht verschärft. Allerdings sind **alle Komplexitätsstufen im absoluten Sinne unzureichend**. Der beobachtete Trend sollte daher nicht als Beleg für robustere Modelle interpretiert werden, sondern lediglich als Hinweis, dass Overfitting durch Komplexität in diesem Fall nicht der limitierende Faktor ist.

Komplexitätsstufe	**R^2 (Mittel)**	**Varianz**	**RMSE (Mittel)**	**Modelle**
Shallow	-0.405	0.317	0.337	32
Medium	-0.318	0.233	0.327	32
Deep	-0.267	0.235	0.321	32
Very Deep	-0.232	0.204	0.317	32

Tab. 7: Event-Validierung: Performance nach Komplexitätsstufe (Trend zu besserer Generalisierung)

Bei der Event-Generalisierung zeigt Glättung (Fenster 3: R^2 −0.281) einen marginalen Vorteil gegenüber keiner Glättung (R^2 −0.330). Kategoriale Features sind ambivalent. Sie verbessern Event-Generalisierung erheblich (mit: R^2 −0.140 vs. ohne: R^2 −0.471, Delta-R^2 +0.331), schaden aber massiv in der Random-Validierung (mit: R^2 0.150 vs. ohne: R^2 0.466, Delta-R^2 −0.316).

Konfiguration	R^2 (Mittel)	Varianz	RMSE (Mittel)	Modelle
Kategoriale Features:				
Mit kategorialen Features	-0.140	0.207	0.304	64
Ohne kategorische Features	-0.471	0.184	0.346	64
Feature-Aggregation:				
Mit Aggregation	-0.300	0.248	0.325	64
Ohne Aggregation	-0.311	0.266	0.326	64
Glättung (Fenstergrößen):				
Fenster 0 (keine)	-0.330	0.316	0.328	32
Fenster 2	-0.296	0.259	0.324	32
Fenster 3	-0.281	0.219	0.323	32
Fenster 4	-0.315	0.229	0.327	32

Tab. 8: Event-Validierung: Effekte der Datensatzstruktur-Varianten

Die Event-Validierung offenbart ein fundamentales Generalisierungsproblem. Mit durchschnittlich negativer R^2 sind die Modelle in diesem Szenario nicht praktisch einsetzbar. Das beste Modell mit R^2 0.093 ist gerade noch marginale besser als eine Baseline.

5.2.3 Interpretation des Gesamtergebnisses

Die massive Diskrepanz zwischen Random- (R^2-Mittel 0.308) und Event-Validierung (R^2-Mittel -0.306) ist nicht auf typisches Overfitting zurückzuführen, sondern offenbart ein fundamentales Problem der Domänen-Generalisierung. Im Folgenden werden die wahrscheinlichen Ursachen und Implikationen dieser Diskrepanz erörtert.

Ein zentrales Problem besteht darin, dass die Telemetriedaten ausschließlich Fahrzeugzustände und Sensorwerte erfassen, jedoch die entscheidenden Kontextfaktoren, die das Understeer maßgeblich beeinflussen, nicht berücksichtigen. So sind beispielsweise individuelle Fahrereffekte wie Lenk-Aggression, Bremspunkt-Variabilität oder die Wahl der Fahrlinie in den Rohdaten mit physikalischen Fahrzeug-Eigenschaften vermischt und lassen sich nicht voneinander trennen. Das Modell erlernt dadurch eine Kombination aus Fahrzeugverhalten und Fahrercharakteristik, die bei neuen Fahrern nicht übertragbar ist. Hinzu kommt, dass das Fahrzeug-Setup, etwa Chassis-Steifigkeit, Aero-Balance oder Federung, sich zwischen den Events und sogar innerhalb eines Events verändert. Diese Parameter sind in den Telemetriedaten nicht explizit enthalten und können allenfalls indirekt über die Reaktion des Fahrzeugs auf die Fahrbahn abgeleitet werden, was jedoch eine unzureichende und wenig robuste Methode darstellt. Darüber hinaus spielen Umgebungsfaktoren wie Streckentemperatur, Luftdichte, Feuchtigkeit und Windverhältnisse eine fundamentale Rolle für Aerodynamik und Reifenverhalten. Da diese Einflussgrößen im Datensatz nicht enthalten sind, können sie auch nicht rekonstruiert werden. Insgesamt erklärt das Fehlen dieser Kontextfaktoren, weshalb das Modell zwar auf bekannten Daten eine hohe Performance erzielt, jedoch beim Transfer auf neue Events deutlich an Vorhersagekraft verliert.

Diese fehlenden Faktoren erklären, warum das Modell eine hohe Random-Validierungs-Performance erreicht (es memoriert Training-Event-Muster) aber bei Event-Transfer kollabiert (neue Kombinationen aus Fahrer, Setup, Reifen, Umwelt sind unbekannt).

Das zentrale Problem ist ein Covariate-Shift: Die Verteilung der Eingabedaten unterscheidet sich zwischen Training und Event-Test, während das Modell unter der Annahme konstanter Kontextfaktoren trainiert wurde. Bei neuen Fahrern, Setups oder Umwelteinflüssen versagt diese Annahme, und die Generalisierung bricht ein. Der Event-Validierungsdatensatz dient somit als Stress-Test für echte Domain Generalization. Für zukünftige ML-Projekte gilt: Die Datenqualität ist entscheidender als die Wahl des Algorithmus, und die Verwendung kategorialer Features sollte mit Blick auf die Ziel-Domäne erfolgen. Robustere Generalisierung erfordert gezieltes Feature-Engineering und Transfer-Learning-Ansätze, da Modelle ohne explizite Berücksichtigung neuer Kontexte nicht zuverlässig auf unbekannte Strecken übertragbar sind.

6 Fazit, Erkenntnisse und Forschungsausblick

6.1 Erfüllung der Anforderungen und Beantwortung der Forschungsfragen

Die vorliegende Arbeit verfolgte das Ziel, ein ML-Modell zur automatisierten Vorhersage von Understeer-Verhalten in Motorsport-Telemetriedaten zu entwickeln. Zur Bewertung des Erfolgs werden sowohl die technischen Anforderungen als auch die formulierten Forschungsfragen gegen die erhaltenen Ergebnisse abgewogen.

Anforderung 1: Automatisierte Understeer-Vorhersage

Die Anforderung, ein ML-Modell zur Vorhersage von Understeer-Werten auf Basis von Telemetriedaten zu entwickeln, ist partiell erfüllt. Das beste Modell auf dem Random-Validierungsdatensatz (XGBoost, Very Deep, $R^2 = 0.657$) demonstriert, dass Understeer-Vorhersage unter kontrollierten Bedingungen möglich ist. Die durchschnittliche Random-Performance über alle 128 Modelle beträgt $R^2 = 0.308$, was darauf hindeutet, dass die Modelle etwa ein Drittel der Varianz in der Zielvariablen erfassen. Allerdings ist diese Performance auf die spezifische Datenverteilung des Trainingsdatensatzes beschränkt. Bei der Event-Validierung, dem realistischeren Szenario mit vollständig unbekannten Rennkontexten, kollabiert die Performance dramatisch. Das beste Event-Modell (LightGBM, Very Deep, $R^2 = 0.093$) erreicht nur marginal bessere Ergebnisse als eine Baseline-Vorhersage, und der Durchschnitt über alle Modelle beträgt $R^2 = -0.306$, was bedeutet, dass die Modelle schlechter abschneiden als triviale Vergleichsmodelle.[88] Diese Diskrepanz offenbart, dass das Ziel einer produktionsreifen, generalisierbaren Vorhersage nicht erreicht wurde.

Anforderung 2: Hohe Vorhersagegenauigkeit

Die angestrebte Vorhersagegenauigkeit von $R^2 > 0{,}7$ wurde nicht erreicht. Das beste Random-Modell erreicht $R^2 = 0{,}657$, knapp unterhalb der Zielvorgabe, scheitert aber bei Event-Generalisierung ($R^2 = 0{,}093$). Die Ursachenanalyse in Kapitel 5.3 identifiziert als Primärproblem nicht algorithmische Limitationen, sondern fehlende Kontextfaktoren: Fahrercharakteristiken, Setup-Parameter, Reifen-Degradation und Umgebungsdaten sind in den verfügbaren Telemetriedaten nicht erfasst und können nicht aus Sensorwerten rekonstruiert werden.[89] Diese fehlenden Variablen determinieren das Understeer-Verhalten fundamental und liefern eine mögliche Erklärung für die schwache Event-Generalisierung.

[88] Vgl. Kap. 5.2
[89] Vgl. Kap. 5.3

6.2 Beantwortung der Forschungsfragen

Forschungsfrage 1: Können Gradient Boosting Decision Trees Understeer-Verhalten in Motorsport-Telemetriedaten vorhersagen?

Die Antwort lautet: Teilweise ja, aber nur unter stark einschränkenden Bedingungen. GBDT-Modelle funktionieren auf trainierten Datensätzen gut (durchschnittliche Random-Validierung: R^2 = 0,308) und zeigen damit, dass Understeer grundsätzlich aus Telemetriedaten extrahierbar ist. Allerdings generalisieren diese Modelle nicht auf unbekannte Rennkontexte (durchschnittliche Event-Validierung: R^2 = -0,306). Das Problem liegt nicht in der Algorithmuswahl, sondern in der fehlenden Datengrundlage. GBDT memorisieren Muster aus trainierten Events (Track-spezifische Telemetrie-Signaturen), können diese aber nicht auf neue Tracks, neue Fahrer oder neue Setup-Konfigurationen übertragen.[90]

Forschungsfrage 2: Welcher Algorithmus (XGBoost vs. LightGBM) generalisiert besser?

Weder XGBoost noch LightGBM zeigt konsistente Überlegenheit. Bei Event-Generalisierung ist LightGBM leicht besser (durchschnittliche R^2 = -0,268 vs. XGBoost -0,343), mit dem besten Event-Modell bei R^2 = 0,093 (LightGBM) vs. R^2 = 0,029 (XGBoost). Bei Random-Validierung dominiert XGBoost deutlich (durchschnittliche R^2 = 0,380 vs. LightGBM 0,237), mit dem besten Modell bei R^2 = 0,657 (XGBoost) vs. R^2 = 0,458 (LightGBM).[91] Diese gegensätzliche Leistung unterstreicht, dass algorithmische Wahl sekundär ist. Beide Algorithmen leiden unter denselben fundamentalen Limitationen: fehlende Kontextfaktoren.

Forschungsfrage 3: Wie wirken sich Datenvorbereitung und Hyperparameter-Tuning auf Performance aus?

Der größte Effekt stammt nicht aus Hyperparameter-Tuning, sondern aus der Feature-Konfiguration, insbesondere der Verwendung kategorialer Features. In der Event-Validierung helfen kategoriale Features (primär Track-Information) um Delta-R^2 = +0,331 (mit Kategorisch: R^2 = -0,140 vs. ohne: R^2 = -0,471). In der Random-Validierung schaden sie um Delta-R^2 = -0,316 (mit Kategorisch: R^2 = 0,150 vs. ohne: R^2 = 0,466). Diese gegensätzliche Wirkung erklärt sich dadurch, dass Track-Variablen als Proxy für nicht erfasste Event-Charakteristiken fungieren, nicht als echte Kausalvariablen.[92] Glättungsvarianten zeigen moderate Effekte (optimal: Fenster 3, Delta-R^2 ca. 0,03–0,05), und Feature-Aggregation hat minimalen Effekt (Delta-R^2 < 0,03). Hyperparameter-Komplexitätsstufen zeigen, dass tiefere Modelle bei Event-Generalisierung helfen (Very Deep besser als Shallow um Delta-R^2 = +0,173), was gegen klassisches Overfitting-Verständnis spricht und eher auf Domain-Shift-Probleme hindeutet.[93]

[90] Vgl. Kap. 5.3
[91] Vgl. Kap. 5.2
[92] Vgl. Kap. 5.3
[93] Vgl. Kap. 5.2

6.3 Design Knowledge und kritische Selbsteinschätzung

Bedauernswerterweise konnte das Ziel, ein ML-Modell zur Vorhersage der Fahrzeugbalance zu entwickeln, nicht erreicht werden. Trotz intensiver Bemühungen in der Datensammlung (17.735 Runden aus 40 Events), Datenvorbereitung (16 Strukturvarianten) und Modellierung (128 trainierte Modelle) blieb die angestrebte Prognosegenauigkeit aus, sodass diese Arbeit kein in der Praxis nutzbares Artefakt hervorgebracht hat. Im Normalfall würde man nun mehrere Iterationen im Design Cycle durchlaufen, was im Rahmen dieser Arbeit aufgrund von Zeit- und Ressourcenbeschränkungen nicht möglich war.[94]

Aus Sicht der Design Science Research Methodologie ist die Evaluate-Phase vollständig dokumentiert: Der Rigor Cycle wurde mit systematischen Evaluationen, etablierten Metriken und 256 Evaluationsergebnissen erfüllt.[95] Der Relevance Cycle wurde teilweise erfüllt und die zugrunde liegenden Probleme aufgezeigt, die Lösung erfordert aber Daten, die außerhalb des Projektumfangs liegen.

Aus den Erkenntnissen des Entwicklungsprozesses ergeben sich folgende Design Principles für zukünftige Arbeiten: Die zentralen Erkenntnisse aus dem Entwicklungsprozess lassen sich wie folgt zusammenfassen: Die algorithmische Wahl (XGBoost vs. LightGBM) ist für die Vorhersagegenauigkeit nachrangig; entscheidend ist die Datenqualität und die Berücksichtigung relevanter Kontextfaktoren. Kategoriale Features wie Track-Informationen verbessern die Modellleistung innerhalb bekannter Datenbereiche, verschlechtern jedoch die Generalisierung auf neue Events, da sie lediglich Trainingsmuster memorisieren. Die Analyse zeigt, dass Domain Shift, also Unterschiede in der Verteilung der Eingabedaten zwischen Trainings- und Testevents, größere Auswirkungen hat als klassisches Overfitting; tiefere Modelle können unter diesen Bedingungen sogar besser generalisieren. Event-basierte Validierung ist daher essenziell, da sie die tatsächliche Generalisierungsfähigkeit offenbart und Random-Validierung zu optimistischen Einschätzungen führt. Insgesamt bleibt die fehlende Erfassung von Kontextfaktoren wie Fahrercharakteristiken, Setup-Parametern, Reifenstatus und Umgebungsbedingungen die zentrale Limitation, die mit reiner Telemetrie nicht überwunden werden kann.

Was dennoch bleibt sind die Erkenntnisse aus dem Entwicklungsprozess, die wertvolle Einblicke in die Herausforderungen und Limitationen bei der Anwendung von ML im Motorsport-Kontext bieten und einen Grundstein für zukünftige Arbeiten legen.

[94] Vgl. Hevner et al. 2004, S. 83
[95] Vgl. Venable, Pries-Heje, Baskerville 2016, S. 77-89

Anhang

Anhangverzeichnis

Anhang 1: Rohtranskripte der Experteninterviews

Anhang 1/1: Erstes Meeting mit Performance-Ingenieur (29.08.2025)

Experte: Teamleiter Performance Engineering (LMDh-Programm), Porsche Motorsport **Datum:** 29.08.2025

Interviewer: Meeting-Eröffnung Vielen Dank für deine Zeit. Könntest du dich bitte kurz vorstellen?

Ingenieur: Vorstellung und Aufgabengebiet Gerne. Ich bin schon länger bei Porsche und leite das LMDh Performance Team. Ich war bereits in der LMP und Formel E tätig. Unsere Aufgaben umfassen Performance, besonders in der Entwicklung, die Simulations- und Kennwertvorgabe für andere Abteilungen (z.B. wie viel Drag und Abtrieb das Auto benötigt), die Reifenentwicklung mit Michelin, die Rundenzeitberechnung inklusive Modellierung, sowie die Datenanalyse im operativen Betrieb. Auch performancerelevante Software im Auto kommt von uns, etwa die Funktionskontrolle.

Interviewer: Definition und Aufgaben des Performance Engineers Du hast die Rolle bereits kurz angeschnitten. Könntest du die Jobrolle des Performance Engineers bitte noch einmal grundlegend erklären und einordnen, wie die typische Datenanalyse an einem Rennwochenende abläuft?

Ingenieur: Rolle und Setup-Optimierung Performance-Ingenieure sind primär für die Performance des Autos verantwortlich. Das heißt, das Auto muss möglichst optimal auf der Strecke eingesetzt werden, logischerweise bezüglich der Rundenzeit. Ein zweiter wichtiger Punkt ist die Betriebssicherheit des Autos. Wir überwachen kritische Parameter wie Bremstemperatur oder Fahrhöhen. Bei Abweichungen melden wir Probleme, damit der Renningenieur das Auto zur Reparatur reinholen kann. Unsere Hauptaufgabe ist die Setup-Arbeit am Auto, die wir während des Rennwochenendes bis zum Rennen optimieren. Dies geschieht in Abstimmung mit dem Renningenieur und dem Fahrer.

Ingenieur: Daten und Setup-Entscheidung Nach jeder Setup-Änderung ist das Feedback des Fahrers essenziell. Dieses fließt zusammen mit den Daten (Telemetrie oder Kabeldaten) in die Entscheidungsfindung ein. Wir analysieren die Daten, um die Balance des Autos (Unter- oder Übersteuern) zu prüfen und die Fahrhöhen zu optimieren. Das ist die Kernaufgabe: Das Auto zusammen mit dem Renningenieur und dem Fahrer zu optimieren. Wir tragen die Verantwortung für das Setup.

Interviewer: Prozess am Rennwochenende Wie sieht dieses Zusammenspiel an einem Rennwochenende genau aus? Es werden Daten gesammelt, die Autos sind auf der Strecke. Gibt es mehrere Performance Engineers, die spezifische Datenbereiche überwachen?

Ingenieur: Simulation und Validierung Der Performance Engineer nutzt Simulationstools. Wenn ich beispielsweise die Feder ändere, simuliert das Tool, wie sich das auf Fahrhöhe oder

Abtrieb auswirkt. Diese Theorie wird dann mit den Streckendaten abgeglichen. Der Performance Engineer bereitet sich intensiv vor und legt vorab eine Auswahl an Setups fest (einen "Blumenstraußän Optionen), die auf Erfahrungswerten und Fahrsimulatordaten basieren. An der Strecke werden diese Optionen gefahren, um die Theorie zu validieren und das beste Setup für das Qualifying und das Rennen auszuwählen.

Ingenieur: Setup-Umsetzung Der Performance Engineer definiert das Setup. Über ein Tablet erhalten die Mechaniker die Anweisung für die Setup-Änderungen (z.B. Federwechsel, Fahrhöhe anpassen). Der Chefmechaniker koordiniert die Umsetzung. Sobald die Änderungen umgesetzt sind, geht das Auto auf die Strecke, liefert neue Daten, und der Prozess der Optimierung läuft iterativ weiter.

Interviewer: Überleitung zum Machine Learning (ML) Projekt Vielen Dank, das war ein sehr hilfreicher Überblick. Ich möchte jetzt kurz unser Projekt vorstellen. Wir arbeiten an einem Proof of Concept für Machine Learning, bei dem wir ein Modell auf Telemetriedaten trainieren wollen.

Interviewer: Erläuterung ML-Typen (Klassifikation) Ganz grundlegend: Bei Klassifikationsmodellen werden viele Telemetrie-Inputs verarbeitet, um eine diskrete Kategorie auszugeben. Das könnte beispielsweise die Reifenmischung (Soft, Medium) oder das Fahrverhalten (aggressiv, optimal) sein.

Interviewer: Erläuterung ML-Typen (Regression/Vorhersage) Der zweite Typ sind Vorhersage- oder Regressionsmodelle. Diese werden trainiert, um einen Wert vorherzusagen, wie zum Beispiel die Rundenzeit. Das Modell würde auf Basis der aktuellen Telemetriedaten in den ersten Sektoren die prognostizierte Endrundenzeit voraussagen.

Interviewer: Erläuterung Trainingsdaten und Labeling Für das Training benötigen wir den Input (Telemetriedaten) und den wahren Output (das Label). Das Label muss entweder berechnet werden können oder in den Daten bereits vorhanden sein, da wir den manuellen Aufwand gering halten wollen.

Ingenieur: Rundenzeit-Vorhersage und Alternativen Die Predicted Laptime haben wir bereits im Auto. Das ist ein simpler, aber gut funktionierender, nicht-ML-basierter Ansatz. Deshalb sollten wir uns auf etwas konzentrieren, das es noch nicht gibt.

Ingenieur: Herausforderung und Ideen (Reifen/Setup) Wir generieren sehr viele Daten, auch Kennzahlen (KPIs). Die Schwierigkeit ist, die Zusammenhänge in kurzer Zeit zu verstehen. Zum Beispiel: Wie beeinflusst das aggressive Aufwärmen des Reifens in den ersten Runden den Grip in Runde 20? Oder die schnelle Entscheidung, welcher optimale Compound bei 40 Grad Streckentemperatur zu wählen ist.

Ingenieur: Fahrer-Feedback und Reifenüberhitzung Ein sehr interessanter Ansatz wäre die Guidance für den Fahrer. Wenn der Fahrer durch zu viel Schlupf den Reifen überfährt und dieser dann abbaut, könnte das Modell ihm mitteilen: "Fahre konservativer, wir sehen in den Daten,

du überfährst den Reifen gerade."Hierzu müssten wir den Schlupf oder die Reifentemperatur überwachen.

Interviewer: Potential des ML-Modells Die Idee, aus verschiedenen Telemetrie-Channels den Output "überfahren / nicht überfahrenßu generieren, ist sehr vielversprechend. Hier liegt ein großer Benefit, da dies momentan auf jahrelanger Erfahrung und Bauchgefühl basiert und nicht einfach in Code abzubilden ist.

Ingenieur: Labelling als Problem und alternative Kennwerte Das Hauptproblem wäre das Training und die Label-Generierung. Wir müssten definieren: "Hier, zu diesem Zeitpunkt wurde der Reifen überfahren."Wenn dieses Label nicht einfach aus den Telemetriedaten ableitbar ist, müsste jemand manuell die Daten durchsehen und markieren, was für euch nicht praktikabel ist.

Ingenieur: Kennwert SSnaps" Wir haben alternative Kennwerte. Zum Beispiel die Anzahl der Snaps pro Runde: wie häufig das Heck beim Beschleunigen ausbricht. Wenn der Reifen abgebaut ist, passiert dies häufiger.

Ingenieur: Long-Run-Daten und Lebensdauer-Prognose Ein weiteres Kriterium ist die Rundenzeit über einen Long Run. Man sieht, wie die Rundenzeit mit der Anzahl der Runden schlechter wird. Hier könnte man visualisieren, wann der Reifen tatsächlich seine Leistung verliert.

Interviewer: Präzisierung der Prognose-Idee Man könnte auf Basis des aktuellen Fahrverhaltens eine Lebensdauer prognostizieren. Beispiel: "Basierend auf den letzten zwei Sektoren hält der Reifen noch drei Runden."Fährt der Fahrer konservativer, steigt der Wert auf fünf.

Ingenieur: Einschränkung der Datenbasis Gibt man dem Modell nur die Rundenzeit, weiß es nicht, *warum* die Zeit schlechter wird (war es das Setup, der Fahrer, der Fahrstil?). Das Modell benötigt auch Input über den Fahrer und die genauen Setup-Einstellungen.

Ingenieur: Empfehlung für Proof of Concept (POC) Für den POC sollten wir Daten von Le Mans oder einem Dauerlauf verwenden, bei denen das Setup konstant war. Dann sind nur noch Fahrer- oder Fahrstilunterschiede relevant.

Interviewer: Abschluss und nächster Termin Die Idee mit dem Dauerlauf ist ein sehr guter Startpunkt für einen Proof of Concept. Ich werde mich jetzt einarbeiten und in zwei Wochen einen neuen Termin vorschlagen.

Ingenieur: Wunsch für nächstes Meeting Es wäre gut, wenn du für das nächste Meeting visualisierst, wie du dir den Prozess vorstellst: Hier kommen Daten rein, das passiert, das kommt als Output raus.

Interviewer: Abschluss Vielen Dank für deine Zeit.

Anhang 1/2: Zweites Meeting mit Performance-Ingenieur (12.09.2025)

Experte: Teamleiter Performance Engineering (LMDh-Programm), Porsche Motorsport **Datum:** 12.09.2025

Interviewer: Problemstellung Reifendegradation und neuer Ansatz Das Thema Reifendegradation ist sehr interessant, aber das Labeln ist eine zu große Hürde. Es ist zu wenig Zeit, um euer langjähriges Bauchgefühl in manuell definierte Schwellwerte zu überführen.

Interviewer: Neue Idee: Fahrerfeedback / Explainability AI Die neue Idee ist, ein Modell zu trainieren, um Fahrerfeedback zu generieren. Wir trainieren das Modell auf die verschiedenen Fahrer und nutzen dann Explainability AI. Das würde die KI-Black-Box aufbrechen und zeigen, wie die KI zu ihrer Entscheidung kommt. Der Mehrwert: Die KI könnte die Unterschiede zwischen den Fahrern aufzeigen und erklären, warum Fahrer 1 in Sektor X schneller war. Der Vorteil für mich ist, dass das Label ("welcher Fahrer fährt gerade") einfach aus den Daten zu ziehen ist.

Ingenieur: Vorschlag: Fokus auf Fahrzeugbalance (Car Balance) Was uns aktuell sehr beschäftigt, ist die Fahrzeugbalance. Ist das Auto eher neutral oder untersteuernd? Die Balance hängt von vielen Parametern ab, deren Zusammenhänge für uns Menschen schwierig zu verstehen sind.

Ingenieur: Modellierung der Balance Man könnte sich das als eine Gleichung vorstellen: Das Y (Fahrzeug Balance) hängt von vielen Eingangswerten X ab (Streckentemperatur, Reifentemperatur, -druck, Fahrer, etc.). Dazwischen liegt eine Funktion, die wir nicht genau kennen.

Interviewer: Tool-Nutzung Welche Tools verwendet ihr live an der Strecke? PowerBi Dashboard oder die genauen Telemetriedaten in Wintax?

Ingenieur: Balance-Definition und Metriken Beides. Wintax ist der reine Telemetriedatenstrom. Zusätzlich haben wir Auswertungen und Kennwerte. Wir haben Cluster für den Kurveneingang, die Kurvenmitte und den Kurvenausgang, und dazu die Balance als Mittelwert über alle Kurven.

Interviewer: Definition "Balance" Was genau ist diese Balance?

Ingenieur: Erklärung der Car Balance Die Balance beschreibt, ob das Auto viel Untersteuern hat. Mathematisch hängt es vom Lenkwinkel (Input vom Fahrer) und der Gierrate (wie schnell sich das Auto dreht) ab. Ist die Reaktion der Gierrate auf den Lenkwinkel direkt, ist das Auto neutral. Muss der Fahrer extrem viel lenken ohne Reaktion, ist es untersteuernd. Reagiert das Auto zu schnell, ist es übersteuernd.

Ingenieur: Der Kern der Fragestellung Interessant ist: Das ist unser Y (die Balance), aber wir verstehen nicht, warum dieser Zustand eintritt. Liegt es daran, dass der Fahrer etwas verstellt hat, sich die Streckentemperatur geändert hat oder der Reifen mehr Kilometer Laufleistung hat? Das ist die Frage, die uns am Ende interessiert.

Ingenieur: Beispiel Streckentemperatur Wenn wir den Einfluss der Streckentemperatur auf die Balance wüssten, wäre das hilfreich. Sagt der Fahrer im FP2 (20 Grad heißer) die Balance sei schlecht, könnten wir sagen: "Das liegt an der Strecke. Im Rennen wird es kälter, das passt dann alles."

Interviewer: Bewertung des Car Balance Themas Verstehe. Hier hätten wir das Labeling (die Balance) also schon. Das ist ein komplett anderes, aber sehr vielversprechendes Thema.

Ingenieur: Update zum Degradations-Kennwert Der Verschleiß-Kennwert ist noch nicht verfügbar. Die Kollegen bekommen die Berechnung der Degradation (die wie die Balance berechnet werden soll) nicht mehr bis Ende November in die Datenplattform.

Ingenieur: Modellierung des Degradations-Kennwerts Dahinter steckt ein Reifenmodell. Wir modellieren einen neuen Reifen und optimieren dann die Modellparameter (z.B. den Grip-Parameter), damit sie zu den Messdaten passen. Diese Optimierung wird aktuell noch händisch in Matlab durchgeführt, bevor sie in die Datenplattform integriert wird.

Interviewer: Rückkehr zum Car Balance POC Die Car Balance ist ein guter Ansatz für den POC. Das wäre die Zielvariable, zum Beispiel die Highspeed-Balance in der Kurvenmitte.

Ingenieur: Empfehlung für POC-Umfang Ich würde vorschlagen, es pro Auto zu machen, da die Setups unterschiedlich sind. Man könnte auch die Daten aller Autos zusammenwerfen, um generelle Regeln zu finden, aber man sollte mit einem Auto starten.

Ingenieur: Datenverfügbarkeit und "Mileage" Wir haben fast alle benötigten Inputs (X-Werte) im Dashboard, außer die Mileage (Laufleistung des Reifens), die aber wichtig ist und als Anzahl der gefahrenen Runden in den Daten existiert.

Ingenieur: Allgemeiner Problemkern Es gibt verschiedene Themen, bei denen wir eine Metrik (Y) und die Eingangsvariablen (X) kennen, aber nicht wissen, wie diese zusammenhängen. Das gilt für die Degradation und die Balance. Unser Ansatz wäre: Wir nutzen die enormen Datenmengen, um die Zusammenhänge zu verstehen. Das wäre wirklich sehr hilfreich.

Interviewer: Abschluss Vielen Dank für das Gespräch, ich glaube, das hat sehr geholfen.

Methodische Anmerkungen zu den Interviews

Die Interviews wurden als unstrukturierte Experteninterviews geführt und digital aufgezeichnet. Die vorliegenden Transkripte sind Rohtranskripte, die zur besseren Lesbarkeit minimal geglättet wurden, jedoch den originalen Gesprächsinhalt und -verlauf authentisch wiedergeben. Die beibehaltenen Zeilennummern dienen der präzisen Zitierfähigkeit.

Die Gespräche dienten der:

- Anforderungsanalyse für das Machine Learning Projekt
- Identifikation relevanter Telemetriedaten und Kenngrößen
- Bewertung verschiedener Ansätze (Fokusverlagerung von Reifendegradation zu Car Balance)
- Klärung technischer Umsetzbarkeit und Datenverfügbarkeit

Anhang 2: Modell-Ergebnisse

Anhang 2/1: Ergebnisse der Event-Validierung

Modelltyp,Parametertiefe,Kategorisch,Aggregate,Smoothed,MAE,RMSE,R2
xgb,shallow,True,True,0,0.272,0.344,-0.443
lgb,shallow,True,True,0,0.25,0.318,-0.239
xgb,deep,False,True,4,0.259,0.33,-0.332
lgb,deep,False,True,4,0.316,0.394,-0.895
xgb,deep,True,False,2,0.242,0.308,-0.162
lgb,deep,True,False,2,0.222,0.284,0.012
xgb,medium,False,True,3,0.277,0.344,-0.443
lgb,medium,False,True,3,0.267,0.335,-0.372
xgb,shallow,False,False,2,0.307,0.38,-0.761
lgb,shallow,False,False,2,0.256,0.33,-0.327
xgb,very-deep,False,True,0,0.267,0.335,-0.372
lgb,very-deep,False,True,0,0.296,0.365,-0.625
xgb,medium,True,False,2,0.259,0.325,-0.291
lgb,medium,True,False,2,0.216,0.28,0.04
xgb,deep,True,True,3,0.23,0.297,-0.074
lgb,deep,True,True,3,0.237,0.303,-0.125
xgb,deep,True,True,2,0.226,0.292,-0.042
lgb,deep,True,True,2,0.235,0.301,-0.108
xgb,medium,False,False,3,0.266,0.336,-0.379
lgb,medium,False,False,3,0.309,0.384,-0.804

xgb,shallow,False,True,4,0.298,0.368,-0.656
lgb,shallow,False,True,4,0.247,0.314,-0.206
xgb,shallow,False,False,0,0.325,0.393,-0.884
lgb,shallow,False,False,0,0.325,0.394,-0.899
xgb,shallow,True,True,4,0.25,0.313,-0.196
lgb,shallow,True,True,4,0.231,0.3,-0.1
xgb,very-deep,False,False,3,0.259,0.328,-0.315
lgb,very-deep,False,False,3,0.258,0.329,-0.322
xgb,deep,True,False,3,0.24,0.307,-0.155
lgb,deep,True,False,3,0.223,0.288,-0.013
xgb,very-deep,True,False,3,0.239,0.306,-0.142
lgb,very-deep,True,False,3,0.21,0.274,0.081
xgb,shallow,False,False,3,0.29,0.362,-0.597
lgb,shallow,False,False,3,0.267,0.342,-0.433
xgb,shallow,True,False,2,0.267,0.345,-0.456
lgb,shallow,True,False,2,0.212,0.276,0.067
xgb,medium,True,False,4,0.287,0.36,-0.584
lgb,medium,True,False,4,0.219,0.281,0.038
xgb,very-deep,True,False,4,0.246,0.314,-0.204
lgb,very-deep,True,False,4,0.22,0.283,0.02
xgb,deep,False,True,2,0.279,0.346,-0.467
lgb,deep,False,True,2,0.296,0.37,-0.672
xgb,shallow,True,False,0,0.34,0.418,-1.135
lgb,shallow,True,False,0,0.211,0.28,0.045
xgb,deep,True,False,4,0.262,0.334,-0.366
lgb,deep,True,False,4,0.223,0.288,-0.01
xgb,very-deep,False,False,4,0.253,0.324,-0.282
lgb,very-deep,False,False,4,0.282,0.354,-0.529
xgb,deep,False,True,0,0.263,0.33,-0.334
lgb,deep,False,True,0,0.299,0.366,-0.638
xgb,medium,False,True,0,0.273,0.341,-0.425
lgb,medium,False,True,0,0.322,0.394,-0.895
xgb,deep,False,True,3,0.266,0.335,-0.373
lgb,deep,False,True,3,0.262,0.33,-0.329
xgb,shallow,False,True,0,0.285,0.356,-0.551
lgb,shallow,False,True,0,0.249,0.316,-0.219
xgb,medium,True,True,0,0.239,0.308,-0.163
lgb,medium,True,True,0,0.238,0.303,-0.119
xgb,shallow,True,False,3,0.248,0.317,-0.225
lgb,shallow,True,False,3,0.218,0.282,0.032
xgb,very-deep,True,False,0,0.261,0.327,-0.305

lgb,very-deep,True,False,0,0.212,0.272,0.093
xgb,deep,True,True,4,0.23,0.295,-0.065
lgb,deep,True,True,4,0.241,0.308,-0.162
xgb,very-deep,False,False,2,0.252,0.321,-0.258
lgb,very-deep,False,False,2,0.284,0.363,-0.606
xgb,medium,True,True,3,0.234,0.3,-0.099
lgb,medium,True,True,3,0.234,0.298,-0.088
xgb,very-deep,True,False,2,0.245,0.312,-0.186
lgb,very-deep,True,False,2,0.221,0.286,0.002
xgb,deep,False,False,0,0.257,0.326,-0.297
lgb,deep,False,False,0,0.265,0.338,-0.394
xgb,shallow,False,True,2,0.325,0.397,-0.928
lgb,shallow,False,True,2,0.295,0.365,-0.628
xgb,medium,True,False,3,0.269,0.34,-0.411
lgb,medium,True,False,3,0.243,0.313,-0.2
xgb,very-deep,False,True,2,0.282,0.349,-0.492
lgb,very-deep,False,True,2,0.268,0.338,-0.395
xgb,deep,True,True,0,0.222,0.287,-0.009
lgb,deep,True,True,0,0.22,0.284,0.013
xgb,medium,False,False,0,0.263,0.336,-0.377
lgb,medium,False,False,0,0.269,0.34,-0.416
xgb,medium,False,True,2,0.286,0.353,-0.526
lgb,medium,False,True,2,0.268,0.337,-0.389
xgb,deep,False,False,4,0.251,0.321,-0.259
lgb,deep,False,False,4,0.283,0.356,-0.55
xgb,shallow,True,True,2,0.248,0.316,-0.216
lgb,shallow,True,True,2,0.218,0.285,0.008
xgb,shallow,False,False,4,0.28,0.35,-0.496
lgb,shallow,False,False,4,0.273,0.348,-0.476
xgb,very-deep,False,True,4,0.273,0.341,-0.421
lgb,very-deep,False,True,4,0.277,0.345,-0.458
xgb,shallow,True,True,3,0.287,0.363,-0.61
lgb,shallow,True,True,3,0.222,0.285,0.007
xgb,shallow,False,True,3,0.297,0.365,-0.625
lgb,shallow,False,True,3,0.255,0.322,-0.264
xgb,medium,False,False,2,0.263,0.334,-0.361
lgb,medium,False,False,2,0.265,0.339,-0.406
xgb,very-deep,True,True,3,0.224,0.29,-0.026
lgb,very-deep,True,True,3,0.227,0.296,-0.072
xgb,medium,True,True,4,0.241,0.311,-0.182
lgb,medium,True,True,4,0.227,0.297,-0.078

xgb,deep,True,False,0,0.252,0.317,-0.229
lgb,deep,True,False,0,0.216,0.276,0.069
xgb,deep,False,False,2,0.243,0.31,-0.175
lgb,deep,False,False,2,0.272,0.348,-0.483
xgb,medium,False,True,4,0.267,0.335,-0.373
lgb,medium,False,True,4,0.289,0.36,-0.583
xgb,very-deep,False,True,3,0.266,0.335,-0.374
lgb,very-deep,False,True,3,0.261,0.328,-0.317
xgb,very-deep,True,True,4,0.229,0.295,-0.062
lgb,very-deep,True,True,4,0.243,0.309,-0.164
xgb,very-deep,True,True,0,0.217,0.282,0.029
lgb,very-deep,True,True,0,0.217,0.282,0.028
xgb,medium,False,False,4,0.264,0.334,-0.363
lgb,medium,False,False,4,0.285,0.356,-0.551
xgb,medium,True,True,2,0.22,0.283,0.02
lgb,medium,True,True,2,0.244,0.31,-0.174
xgb,shallow,True,False,4,0.273,0.35,-0.497
lgb,shallow,True,False,4,0.22,0.293,-0.052
xgb,very-deep,True,True,2,0.227,0.291,-0.037
lgb,very-deep,True,True,2,0.228,0.296,-0.073
xgb,medium,True,False,0,0.261,0.328,-0.316
lgb,medium,True,False,0,0.214,0.274,0.084
xgb,deep,False,False,3,0.259,0.328,-0.317
lgb,deep,False,False,3,0.286,0.362,-0.598
xgb,very-deep,False,False,0,0.255,0.325,-0.289
lgb,very-deep,False,False,0,0.262,0.334,-0.362

Anhang 2/2: Ergebnisse der Zufalls-Validierung

Modelltyp,Parametertiefe,Kategorisch,Aggregate,Smoothed,MAE,RMSE,R2
xgb,shallow,True,True,0,0.285,0.358,0.193
lgb,shallow,True,True,0,0.319,0.399,-0.0
xgb,deep,False,True,4,0.186,0.243,0.628
lgb,deep,False,True,4,0.269,0.336,0.292
xgb,deep,True,False,2,0.276,0.352,0.219
lgb,deep,True,False,2,0.299,0.373,0.126
xgb,medium,False,True,3,0.177,0.234,0.654
lgb,medium,False,True,3,0.257,0.321,0.352
xgb,shallow,False,False,2,0.218,0.283,0.496
lgb,shallow,False,False,2,0.27,0.334,0.3
xgb,very-deep,False,True,0,0.176,0.234,0.657

lgb,very-deep,False,True,0,0.245,0.311,0.391
xgb,medium,True,False,2,0.272,0.348,0.238
lgb,medium,True,False,2,0.291,0.366,0.156
xgb,deep,True,True,3,0.278,0.353,0.216
lgb,deep,True,True,3,0.284,0.354,0.213
xgb,deep,True,True,2,0.294,0.366,0.158
lgb,deep,True,True,2,0.288,0.36,0.184
xgb,medium,False,False,3,0.19,0.251,0.603
lgb,medium,False,False,3,0.251,0.316,0.373
xgb,shallow,False,True,4,0.198,0.257,0.584
lgb,shallow,False,True,4,0.267,0.33,0.316
xgb,shallow,False,False,0,0.225,0.291,0.468
lgb,shallow,False,False,0,0.268,0.339,0.278
xgb,shallow,True,True,4,0.333,0.407,-0.043
lgb,shallow,True,True,4,0.298,0.375,0.117
xgb,very-deep,False,False,3,0.184,0.244,0.625
lgb,very-deep,False,False,3,0.26,0.324,0.341
xgb,deep,True,False,3,0.277,0.354,0.213
lgb,deep,True,False,3,0.304,0.377,0.107
xgb,very-deep,True,False,3,0.283,0.361,0.18
lgb,very-deep,True,False,3,0.295,0.367,0.152
xgb,shallow,False,False,3,0.22,0.285,0.489
lgb,shallow,False,False,3,0.282,0.349,0.233
xgb,shallow,True,False,2,0.287,0.365,0.16
lgb,shallow,True,False,2,0.304,0.381,0.085
xgb,medium,True,False,4,0.273,0.359,0.191
lgb,medium,True,False,4,0.295,0.37,0.138
xgb,very-deep,True,False,4,0.287,0.37,0.138
lgb,very-deep,True,False,4,0.301,0.375,0.115
xgb,deep,False,True,2,0.179,0.237,0.646
lgb,deep,False,True,2,0.259,0.326,0.332
xgb,shallow,True,False,0,0.289,0.375,0.114
lgb,shallow,True,False,0,0.3,0.375,0.114
xgb,deep,True,False,4,0.278,0.36,0.183
lgb,deep,True,False,4,0.305,0.382,0.08
xgb,very-deep,False,False,4,0.195,0.26,0.574
lgb,very-deep,False,False,4,0.263,0.325,0.335
xgb,deep,False,True,0,0.182,0.243,0.628
lgb,deep,False,True,0,0.23,0.294,0.458
xgb,medium,False,True,0,0.188,0.248,0.612
lgb,medium,False,True,0,0.255,0.323,0.342

xgb,deep,False,True,3,0.184,0.24,0.638
lgb,deep,False,True,3,0.268,0.339,0.278
xgb,shallow,False,True,0,0.193,0.253,0.597
lgb,shallow,False,True,0,0.266,0.335,0.294
xgb,medium,True,True,0,0.298,0.372,0.131
lgb,medium,True,True,0,0.287,0.357,0.196
xgb,shallow,True,False,3,0.295,0.373,0.123
lgb,shallow,True,False,3,0.316,0.398,0.003
xgb,very-deep,True,False,0,0.277,0.352,0.22
lgb,very-deep,True,False,0,0.303,0.376,0.112
xgb,deep,True,True,4,0.281,0.355,0.205
lgb,deep,True,True,4,0.283,0.353,0.215
xgb,very-deep,False,False,2,0.192,0.256,0.588
lgb,very-deep,False,False,2,0.262,0.324,0.339
xgb,medium,True,True,3,0.26,0.332,0.305
lgb,medium,True,True,3,0.294,0.365,0.16
xgb,very-deep,True,False,2,0.28,0.355,0.206
lgb,very-deep,True,False,2,0.295,0.369,0.141
xgb,deep,False,False,0,0.19,0.251,0.603
lgb,deep,False,False,0,0.254,0.324,0.339
xgb,shallow,False,True,2,0.196,0.256,0.589
lgb,shallow,False,True,2,0.26,0.324,0.34
xgb,medium,True,False,3,0.269,0.347,0.241
lgb,medium,True,False,3,0.291,0.365,0.164
xgb,very-deep,False,True,2,0.179,0.238,0.643
lgb,very-deep,False,True,2,0.253,0.32,0.357
xgb,deep,True,True,0,0.294,0.368,0.149
lgb,deep,True,True,0,0.297,0.366,0.158
xgb,medium,False,False,0,0.192,0.253,0.597
lgb,medium,False,False,0,0.259,0.333,0.302
xgb,medium,False,True,2,0.18,0.238,0.644
lgb,medium,False,True,2,0.253,0.317,0.369
xgb,deep,False,False,4,0.193,0.256,0.587
lgb,deep,False,False,4,0.26,0.324,0.342
xgb,shallow,True,True,2,0.372,0.461,-0.338
lgb,shallow,True,True,2,0.303,0.376,0.112
xgb,shallow,False,False,4,0.213,0.275,0.523
lgb,shallow,False,False,4,0.285,0.348,0.24
xgb,very-deep,False,True,4,0.187,0.25,0.608
lgb,very-deep,False,True,4,0.259,0.325,0.336
xgb,shallow,True,True,3,0.278,0.353,0.217

lgb,shallow,True,True,3,0.291,0.367,0.152
xgb,shallow,False,True,3,0.199,0.261,0.571
lgb,shallow,False,True,3,0.252,0.314,0.381
xgb,medium,False,False,2,0.19,0.252,0.602
lgb,medium,False,False,2,0.256,0.322,0.347
xgb,very-deep,True,True,3,0.28,0.354,0.21
lgb,very-deep,True,True,3,0.284,0.353,0.215
xgb,medium,True,True,4,0.268,0.344,0.257
lgb,medium,True,True,4,0.298,0.371,0.135
xgb,deep,True,False,0,0.284,0.356,0.204
lgb,deep,True,False,0,0.31,0.384,0.073
xgb,deep,False,False,2,0.196,0.261,0.572
lgb,deep,False,False,2,0.257,0.323,0.344
xgb,medium,False,True,4,0.184,0.245,0.623
lgb,medium,False,True,4,0.242,0.304,0.417
xgb,very-deep,False,True,3,0.184,0.24,0.638
lgb,very-deep,False,True,3,0.266,0.333,0.301
xgb,very-deep,True,True,4,0.297,0.373,0.126
lgb,very-deep,True,True,4,0.282,0.353,0.218
xgb,very-deep,True,True,0,0.29,0.363,0.173
lgb,very-deep,True,True,0,0.291,0.361,0.18
xgb,medium,False,False,4,0.195,0.26,0.574
lgb,medium,False,False,4,0.258,0.326,0.332
xgb,medium,True,True,2,0.273,0.344,0.254
lgb,medium,True,True,2,0.302,0.376,0.112
xgb,shallow,True,False,4,0.31,0.398,0.005
lgb,shallow,True,False,4,0.3,0.374,0.12
xgb,very-deep,True,True,2,0.292,0.364,0.167
lgb,very-deep,True,True,2,0.28,0.35,0.229
xgb,medium,True,False,0,0.279,0.355,0.209
lgb,medium,True,False,0,0.301,0.375,0.118
xgb,deep,False,False,3,0.184,0.244,0.624
lgb,deep,False,False,3,0.252,0.315,0.377
xgb,very-deep,False,False,0,0.192,0.256,0.589
lgb,very-deep,False,False,0,0.249,0.314,0.379

Literaturverzeichnis

24 Hours of Le Mans (2025): Classes: Le Mans Hypercar and LMDh. URL: https://www.24h-lemans.com/en/lemans/classes (Abruf: 30. 10. 2025).

Bergstra, James; Bengio, Yoshua (2012): Random Search for Hyper-Parameter Optimization. In: *Journal of Machine Learning Research* 13.10, S. 281–305.

Box, George E. P.; Jenkins, Gwilym M.; Reinsel, Gregory C.; Ljung, Greta M. (2015): Time Series Analysis: Forecasting and Control. 5th. John Wiley & Sons. ISBN: 978-1118675021.

Breiman, Leo (2001): Random Forests. In: *Machine Learning* 45.1, S. 5–32. DOI: 10.1023/A:1010933404324.

Chai, Tianfeng; Draxler, Roland R. (2014): Root mean square error (RMSE) or mean absolute error (MAE)? Arguments against avoiding RMSE in the literature. In: *Geoscientific Model Development* 7, S. 1247–1250. DOI: 10.5194/gmd-7-1247-2014.

Chapman, Pete; Clinton, Julian; Kerber, Randy; Khabaza, Thomas; Reinartz, Thomas; Shearer, Colin; Wirth, Rüdiger (2000): CRISP-DM 1.0: Step-by-step data mining guide. Techn. Ber.

Chen, Tianqi; Guestrin, Carlos (2016): XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, S. 785–794. DOI: 10.1145/2939672.2939785.

Cleveland, William S. (1979): Robust Locally Weighted Regression and Smoothing Scatterplots. In: *Journal of the American Statistical Association* 74.368, S. 829–836. DOI: 10.1080/01621459.1979.10481038.

Friedman, Jerome; Hastie, Trevor; Tibshirani, Robert (2009): The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd. Springer.

Friedman, Jerome H. (2001): Greedy Function Approximation: A Gradient Boosting Machine. In: *Annals of Statistics* 29.5, S. 1189–1232. DOI: 10.1214/aos/1013203451.

Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016): Deep Learning. MIT Press. ISBN: 9780262035613.

Gregor, Shirley; Jones, David (2007): The Anatomy of a Design Theory. In: *Journal of the Association for Information Systems* 8.5, S. 312–335. DOI: 10.17705/1jais.00129.

Hevner, Alan R.; March, Salvatore T.; Park, Jinsoo; Ram, Sudha (2004): Design Science in Information Systems Research. In: *MIS Quarterly* 28.1, S. 75–105. DOI: 10.2307/25148625.

Hodson, Timothy O. (2022): Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not. In: *Geoscientific Model Development* 15, S. 5481–5492. DOI: 10.5194/gmd-15-5481-2022.

Ke, Guolin; Meng, Qi; Finley, Thomas; Wang, Taifeng; Chen, Wei; Ma, Weidong; Ye, Qiwei; Liu, Tie-Yan (2017): LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In: *Proceedings of NIPS 2017*.

Kohavi, Ron (1995): A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*. Montreal, Canada, S. 1137–1145.

Kuhn, Max; Johnson, Kjell (2019): Feature Engineering and Selection: A Practical Approach for Predictive Models. 1st. Chapman und Hall/CRC. DOI: 10.1201/9781315108230.

O'Donnell, James et al. (2024): Determination of Multi-Component Failure in Automotive Applications. In: *Journal of Computing and Information Science in Engineering* 24.2. DOI: 10.1115/1.4063003.

Peffers, Ken; Tuunanen, Tuure; Rothenberger, Marcus A.; Chatterjee, Samir (2007): A Design Science Research Methodology for Information Systems Research. In: *Journal of Management Information Systems* 24.3, S. 45–77. DOI: 10.2753/MIS0742-1222240302.

Porsche (2023): LMDh-Reglement: Beginn einer neuen Ära im Langstrecken-Motorsport. URL: https://newsroom.porsche.com/de/pressemappen/le-mans-2023/LMDh-Reglement--Beginn-einer-neuen-%C3%84ra-im-Langstrecken-Motorsport.html (Abruf: 30. 10. 2025).

Tukey, John Wilder (1977): Exploratory Data Analysis. Addison-Wesley Publishing Company. ISBN: 0201076160.

Venable, John R.; Pries-Heje, Jan; Baskerville, Richard L. (2016): FEDS: A Framework for Evaluation in Design Science Research. In: *European Journal of Information Systems* 25.1, S. 77–89. DOI: 10.1057/ejis.2014.36.

Erklärung zur Verwendung generativer KI-Systeme

Bei der Erstellung der eingereichten Arbeit habe ich auf künstlicher Intelligenz (KI) basierte Systeme benutzt:

☒ ja

☐ nein

Falls ja: Die nachfolgend aufgeführten auf künstlicher Intelligenz (KI) basierten Systeme habe ich bei der Erstellung der eingereichten Arbeit benutzt:

1. Perplexity
2. Google Gemini

Ich erkläre, dass ich

- mich aktiv über die Leistungsfähigkeit und Beschränkungen der oben genannten KI-Systeme informiert habe,
- die aus den oben angegebenen KI-Systemen direkt oder sinngemäß übernommenen Passagen gekennzeichnet habe,
- überprüft habe, dass die mithilfe der oben genannten KI-Systeme generierten und von mir übernommenen Inhalte faktisch richtig sind,
- mir bewusst bin, dass ich als Autorin bzw. Autor dieser Arbeit die Verantwortung für die in ihr gemachten Angaben und Aussagen trage.

Die oben genannten KI-Systeme habe ich wie im Folgenden dargestellt eingesetzt:

Arbeitsschritt in der wissenschaftlichen Arbeit	**Eingesetzte(s) KI-System(e)**	**Beschreibung der Verwendungsweise**
Quellen Recherche	Perplexity	Unterstützung bei der Literatursuche und -analyse
Erstellung Fließtext	Perplexity, Google Gemini	Unterstützung bei der sprachlichen Ausformulierung

(Ort, Datum) (Unterschrift)

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema: *Automatisierte Fahrzeugbalance-Analyse im Motorsport: Entwicklung und Evaluation von Gradient-Boosting-Modellen auf Basis von Porsche LMDh Telemetriedaten* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

(Ort, Datum) (Unterschrift)