



Applied Machine Learning

Hussnain Khalil
Betreuer: Prof. Dr. Sina Huber
huk7464@thi.de

Entwicklung eines prädiktiven Wartungsmodells zur Fehlererkennung bei Windkraftanlagen unter Verwendung von IIoT-Daten: Datenanalyse, Resampling-Techniken und Optimierung durch maschinelles Lernen

Inhaltsverzeichnis

1	Einleitung	2
1.1	Motivation	2
1.2	Zielsetzung	2
1.3	Aufbau der Arbeit	2
2	Hintergrund und Problemstellung	2
2.1	Einführung in Predictive Maintenance	2
2.2	Herausforderungen bei Windkraftanlagen	3
2.3	Motivation zur Datenanalyse und Modellierung	3
3	Datenanalyse und Vorverarbeitung	4
3.1	Beschreibung des Datensatzes	4
3.2	Explorative Datenanalyse (EDA)	4
3.3	Feature Engineering und Datenvorbereitung	5
3.4	Umgang mit stark unausgeglichenen Daten	6
4	Modellauswahl und Implementierung	6
4.1	Vorstellung der Modellkandidaten und deren theoretische Grundlagen	6
4.2	Begründete Auswahl der finalen Modelle	8
4.3	Implementierung und Parameterwahl	8
5	Modellbewertung und Optimierung	9
5.1	Evaluationsmetriken für Klassifikationsmodelle	9
5.2	Lernkurvenanalyse	10
5.3	Hyperparameter-Optimierung	10
5.4	Hyperparameter-Optimierung: Ergebnisse	11
5.5	Lernkurvenanalyse nach Hyperparameter-Tuning	12
6	Ergebnisse und Interpretation	13
6.1	Vergleich der Modelle und kritische Reflexion	13
6.2	Ergebnisse der Feature-Importance-Analyse	14
6.3	Schlussfolgerungen aus den Ergebnissen	14
7	Diskussion und Ausblick	15
7.1	Handlungsempfehlungen und praktische Umsetzung der Ergebnisse	15
7.2	Limitationen der Analyse	15
7.3	Potenzielle Verbesserungen und weitere Ansätze	15
8	Fazit	16

1 Einleitung

1.1 Motivation

Die Motivation für dieses Projekt ergibt sich aus seiner Integration in den Kurs **Applied Machine Learning (AML)**, in dem theoretische Kenntnisse über maschinelle Lernalgorithmen in praktische Anwendungen umgesetzt wurden. Wöchentliche Vorlesungen zu überwachten Lernmethoden, Modellevaluierung, Hyperparameter-Optimierung und Datenvorverarbeitung fanden ihren Höhepunkt in diesem praxisnahen Projekt. Im Mittelpunkt dieses Projekts steht das reale Problem der **prädiktiven Wartung für Windkraftanlagen**, bei dem die Herausforderungen hochgradig unausgeglicher Datensätze und der Fehlerklassifikation adressiert werden. Das Projekt unterstreicht die Bedeutung der Nutzung von ML-Algorithmen, um Turbinenausfälle zu erkennen, Wartungspläne zu optimieren und Ausfallzeiten zu minimieren. Dieser zweigleisige Ansatz vertieft nicht nur das Verständnis für maschinelle Lerntechniken, sondern trägt auch zur Weiterentwicklung von Lösungen für nachhaltige Energiesysteme bei.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung eines robusten und genauen Klassifikationsmodells zur Fehlererkennung bei Windturbinen. Hierfür wird ein stark unausgewogenes Datenset aus einer industriellen IoT-Umgebung verwendet, das Sensor- und Betriebsdaten von Windturbinen umfasst. Die Analyse und Modellierung sollen nicht nur das Verständnis der zugrunde liegenden Muster in den Daten verbessern, sondern auch praktikable Vorhersagemodelle liefern, die frühzeitig Fehler erkennen und vorbeugende Wartungsmaßnahmen ermöglichen. Der Einsatz von Machine Learning in diesem Kontext ist entscheidend, da traditionelle regelbasierte Ansätze oft an der Komplexität und Dynamik solcher industriellen Daten scheitern. Windturbinen erzeugen kontinuierlich eine enorme Menge heterogener Daten, darunter Temperatur-, Drehzahl- und Leistungswerte. Aufgrund der hochdimensionalen und stark korrelierten Merkmale sowie der ungleichen Verteilung zwischen Fehler- und Nicht-Fehlerklassen sind ML-Methoden unverzichtbar. Mithilfe fortschrittlicher Resampling-Methoden wie SMOTE-ENN kann die Datenungleichheit reduziert werden, wodurch Modelle entstehen, die eine hohe Leistung insbesondere in Bezug auf Recall und F1-Score für die Fehlerklasse erzielen. Darüber hinaus ermöglichen ML-Algorithmen die Erkennung nichtlinearer Beziehungen zwischen Merkmalen, die durch einfache statistische Methoden oft unentdeckt bleiben, und bieten eine Generalisierbarkeit und Skalierbarkeit, die für andere Windturbinen oder ähnliche industrielle Szenarien angepasst werden kann. Das konkrete Ziel dieser Arbeit ist die Evaluierung und der Vergleich verschiedener Machine-Learning-Modelle, wie Entscheidungsbäume, Random Forest und XGBoost, um ein optimales Modell zur Fehlererkennung zu finden. Der Fokus liegt dabei auf den Metriken **Recall** und **F1-Score** für die Fehlerklasse (Class 1), um sicherzustellen, dass Fehler mit hoher Sensitivität erkannt werden, ohne die Präzision drastisch zu reduzieren. Ergänzend wird durch eine umfangreiche Visualisierung der Ergebnisse ein besseres Verständnis der Daten und der Modellleistung ermöglicht.

1.3 Aufbau der Arbeit

Der Aufbau dieser Arbeit orientiert sich an einem strukturierten Workflow, der schrittweise die wesentlichen Phasen eines Machine-Learning-Projekts für die Fehlererkennung bei Windturbinen umfasst. Der gesamte Prozess wird durch ein rekursives Pipeline-Design realisiert, das die iterative Optimierung und Anpassung der Modelle ermöglicht. Die Arbeit beginnt mit dem Abrufen und der Vorbereitung der Originaldaten aus einer industriellen IoT-Umgebung. In dieser Phase werden die Daten exploriert, bereinigt und vorbereitet, wobei fehlende Werte, Ausreißer und irrelevante Features identifiziert und entsprechend behandelt werden. Anschließend erfolgt die erste Modellentwicklung, bei der Modelle wie **Decision Tree**, **Random Forest** und **XGBoost** trainiert und validiert werden. Die Bewertung der Modelle erfolgt anhand relevanter Metriken wie **Recall**, **Präzision** und **F1-Score**, um erste Erkenntnisse über die Modellleistung und die Datenstruktur zu gewinnen. Um die Modellleistung zu verbessern, wird im nächsten Schritt ein **Hyperparameter-Tuning mittels GridSearchCV** durchgeführt. Hierbei liegt der Fokus auf der Fehlerklasse (Class 1), um die Sensitivität(Recall) der Modelle zu maximieren. Die Modelle werden anschließend erneut evaluiert und validiert. Parallel dazu wird die Bedeutung der einzelnen Features analysiert, um datengetriebene Entscheidungen der Modelle besser zu verstehen. Da das ursprüngliche Datenset hochgradig unausgewogen ist, wird anschließend Resampling mit **SMOTE-ENN** angewendet. Diese Methode kombiniert Oversampling und Cleaning, um eine balancierte Datenverteilung zu gewährleisten, sodass die Modelle gleichermaßen für beide Klassen (Fehler und Nicht-Fehler) trainiert werden können. Nach dem Resampling werden die optimierten Modelle erneut auf den balancierten Daten trainiert und evaluiert, um die Auswirkungen der verbesserten Datenbalance auf die Modellleistung zu analysieren. Eine weitere Runde des Hyperparameter-Tunings folgt, um die Ergebnisse zu optimieren, begleitet von einer abschließenden Analyse der Feature-Wichtigkeit auf Basis der finalen Modelle. **Abbildung 1** illustriert den gesamten Machine-Learning-Workflow, der in dieser Arbeit umgesetzt wurde: Die iterative Struktur des Workflows erlaubt es, die Modelle Schritt für Schritt zu verbessern und schließlich robuste und generalisierbare Lösungen für das gegebene Anwendungsproblem zu entwickeln. Die klare Trennung der einzelnen Schritte ermöglicht zudem eine transparente und reproduzierbare Arbeitsweise.

2 Hintergrund und Problemstellung

2.1 Einführung in Predictive Maintenance

Predictive Maintenance (PdM) ist eine proaktive Strategie, die Datenanalysen und maschinelles Lernen nutzt, um Geräteausfälle vorherzusagen, bevor sie auftreten. Durch die Analyse von Daten aus verschiedenen Sensoren ermöglicht PdM eine rechtzeitige Planung von Wartungsarbeiten, wodurch ungeplante Ausfallzeiten reduziert und die betriebliche Effizienz optimiert werden können. Im Kontext von Windkraftanlagen ist PdM von besonderem Wert. Windturbinen arbeiten unter variablen und oft rauen Umweltbedingungen, was zu Verschleiß und potenziellen Ausfällen führt. Die Implementierung von PdM ermöglicht die frühzeitige Erkennung

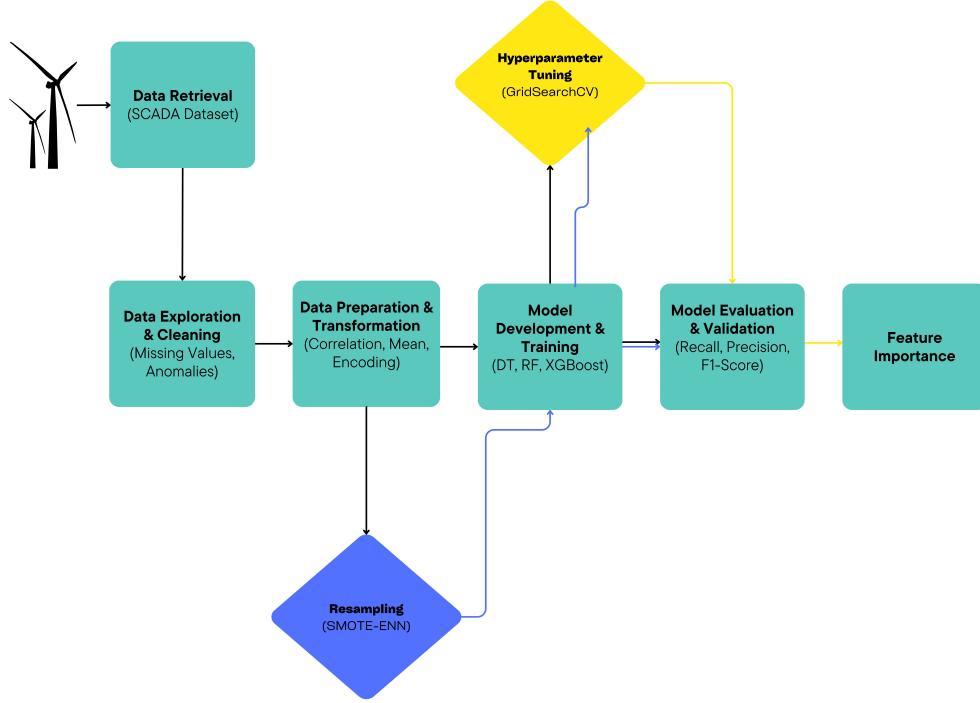


Abbildung 1: Machine-Learning-Workflow

von Anomalien und damit rechtzeitige Eingriffe, um größere Ausfälle zu verhindern und die Lebensdauer der Turbinenkomponenten zu verlängern. In diesem Projekt liegt der Fokus auf der Entwicklung eines PdM-Modells für Windkraftanlagen, mit dem Ziel, potenzielle Ausfälle vorherzusagen und Wartungspläne zu optimieren. Das hierfür genutzte Datenset stammt aus Supervisory Control and Data Acquisition (SCADA)-Systemen, die kontinuierlich die Leistung der Turbinen überwachen. SCADA-Daten umfassen verschiedene Parameter, darunter Temperatur, Druck, Vibrationen und Leistungsabgaben, und bieten somit einen umfassenden Überblick über den Betriebszustand der Turbinen. Das primäre Ziel besteht darin, diese Daten zu analysieren, um Muster zu identifizieren, die auf bevorstehende Ausfälle hinweisen. Mithilfe von Algorithmen des maschinellen Lernens soll ein prädiktives Modell entwickelt werden, das Ausfälle präzise vorhersagen kann, wodurch eine proaktive Wartung erleichtert und die Zuverlässigkeit des Betriebs von Windkraftanlagen gesteigert wird. Dieser Ansatz minimiert nicht nur Ausfallzeiten und Wartungskosten, sondern trägt auch zur Gesamtoptimierung der Effizienz und Nachhaltigkeit der Windenergieproduktion bei.

2.2 Herausforderungen bei Windkraftanlagen

Windkraftanlagen stehen vor zahlreichen Herausforderungen, die durch ihre Betriebsumgebung und die Komplexität ihrer Komponenten bedingt sind. Diese Herausforderungen beeinflussen direkt die Zuverlässigkeit prädiktiver Wartungsmodelle und unterstreichen die Bedeutung einer robusten Datenanalyse und eines durchdachten Modell-Designs. Eine der zentralen Herausforderungen sind **[Umweltschwankungen]**. Windkraftanlagen arbeiten unter rauen und unvorhersehbaren Umweltbedingungen wie extremen Windgeschwindigkeiten, Temperaturschwankungen und hoher Luftfeuchtigkeit. Diese Faktoren führen zu erheblichem Verschleiß von Komponenten wie Lagern, Getrieben und Rotorblättern, was zu Betriebseinschränkungen und Ausfällen führen kann. Die Auswirkungen dieser Umweltbedingungen auf die Leistung von Windkraftanlagen sind in Abbildung 2 dargestellt. Ein weiteres Problem ist das **Datenungleichgewicht** im SCADA-Datensatz, bei dem Fehlerereignisse deutlich seltener auftreten als normale Betriebszustände. Dieses Ungleichgewicht erschwert es Modellen, Fehler präzise vorherzusagen, da sie dazu neigen, sich auf die Mehrheitsklasse (Normalbetrieb) zu konzentrieren und die Minderheitsklasse (Fehler) zu vernachlässigen. Ein dritter wesentlicher Punkt betrifft das **Sensordatenrauschen und die Datenqualität**. SCADA-Systeme erfassen oft Daten, die verrauscht, unvollständig oder Ausreißer enthalten. Diese Probleme können die Zuverlässigkeit prädiktiver Modelle erheblich beeinträchtigen, wenn sie während der Vorverarbeitung nicht angemessen behandelt werden. Eine sorgfältige Bereinigung und Validierung der Daten ist daher unerlässlich. Der SCADA-Datensatz stellt außerdem durch seine **hohe Dimensionalität** eine Herausforderung dar. Viele Features können redundant oder irrelevant für die Fehlerprognose sein, was das Risiko von Overfitting erhöht. Eine effektive Auswahl der wichtigsten Features im Rahmen des Feature-Selection-Prozesses ist daher essenziell, um die Komplexität zu reduzieren und die Modellleistung zu verbessern. Diese Herausforderungen verdeutlichen die Notwendigkeit einer gründlichen Datenvorverarbeitung, einer effektiven Auswahl von Features und einer sorgfältigen Modellevaluierung, um sicherzustellen, dass das prädiktive Wartungsmodell unter realen Bedingungen zuverlässig funktioniert. Die Einbindung von Visualisierungen und Metriken zu jeder Herausforderung stärkt die Analyse und ermöglicht ein klareres Verständnis der Problemstellung.

2.3 Motivation zur Datenanalyse und Modellierung

Die Motivation für die Datenanalyse und Modellierung in diesem Projekt basiert auf den spezifischen technischen und betrieblichen Herausforderungen von Windkraftanlagen sowie auf den Erkenntnissen, die durch die Nutzung von SCADA-Daten möglich sind.

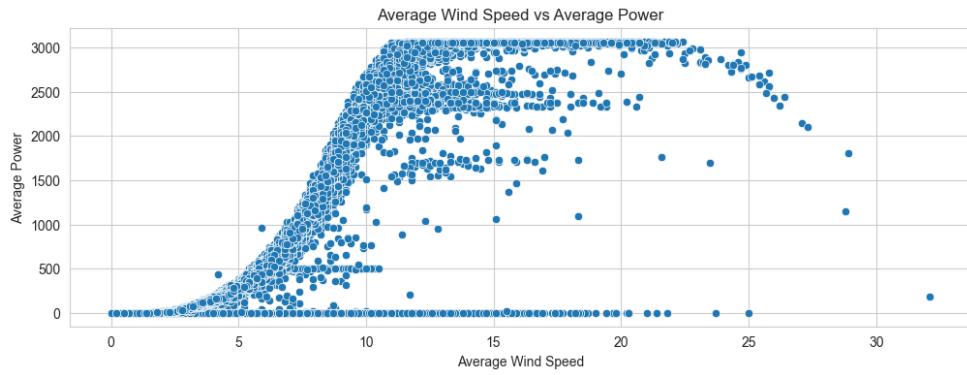


Abbildung 2: Windgeschwindigkeitsverteilung: Auswirkungen auf die Leistung von Windkraftanlagen.

Windkraftanlagen generieren kontinuierlich große Mengen an Sensordaten, darunter Temperatur-, Vibrations- und Leistungswerte. Diese Daten repräsentieren das Verhalten und den Zustand der Anlagenkomponenten und bieten damit die Grundlage, um potenzielle Fehler frühzeitig zu identifizieren und gezielt darauf zu reagieren. Eine der zentralen Motivationen ergibt sich aus dem hochgradig unausgeglichenen Charakter der Daten, bei denen Fehlerereignisse im Vergleich zu normalen Betriebszuständen nur selten auftreten. Diese Datenverteilung erfordert gezielte Ansätze zur Modellierung, wie Resampling-Methoden und robuste Evaluierungsmethoden, um sicherzustellen, dass Modelle auch für die Minderheitsklasse (Fehler) präzise Vorhersagen treffen können. Zusätzlich erfordert die hohe Dimensionalität der SCADA-Daten eine gründliche Feature-Engineering-Phase, um relevante Merkmale zu identifizieren und unnötige Komplexität zu reduzieren. Darüber hinaus sind die Daten oft verraucht und enthalten Ausreißer, was die Notwendigkeit einer sorgfältigen Vorverarbeitung und Qualitätskontrolle unterstreicht. Diese Schritte sind essenziell, um die Verlässlichkeit der Modelle zu erhöhen und sicherzustellen, dass die prädiktiven Ergebnisse auf einer robusten Datenbasis beruhen. Das Ziel der Modellierung ist es, aus den bestehenden Daten Muster zu extrahieren, die auf bevorstehende Fehler hinweisen, und damit nicht nur die Wartungskosten zu senken, sondern auch die Zuverlässigkeit der Windkraftanlagen zu steigern. Durch diesen datengetriebenen Ansatz wird ein direkter Beitrag zur Optimierung der Betriebseffizienz und zur Sicherstellung der Nachhaltigkeit von Windenergieanlagen geleistet, was die Relevanz und den Nutzen der Datenanalyse und Modellierung unterstreicht.

3 Datenanalyse und Vorverarbeitung

3.1 Beschreibung des Datensatzes

Der für dieses Projekt verwendete Datensatz [1] enthält von einem SCADA-System aufgezeichnete Betriebsdaten von Windturbinen. Mit insgesamt 49.027 Zeilen und 66 Spalten handelt es sich um einen bereinigten Datensatz, da er keine fehlenden Werte enthält, jedoch einige Anomalien aufweist. Dieser Datensatz bietet eine Grundlage für die Entwicklung und Bewertung von Modellen für die vorausschauende Wartung. Der Datensatz umfasst insgesamt 49.027 Zeilen, wobei jede Zeile eine Beobachtung eines spezifischen Zeitpunkts repräsentiert, und 66 Spalten, die verschiedene physikalische und betriebliche Parameter beschreiben. Zu den wichtigsten Spalten zählen die Spalte **DateTime**, die den Zeitstempel der Messung enthält und später für zeitbezogene Analysen in ein datetime-Format umgewandelt wird, sowie die Zielspalte **Error**, die Fehlerereignisse darstellt. Ein Wert von 0 in der Spalte **Error** bedeutet "kein Fehler", während alle anderen Werte als Fehler klassifiziert werden können. Die Häufigkeitsanalyse zeigt eine starke Klassenungleichheit: Von den 49.027 Instanzen repräsentieren 48.727 (ca. 99.39%) fehlerfreie Zustände (**Error = 0**), während nur 300 Instanzen (ca. 0.61%) Fehlerzustände (**Error ≠ 0**) darstellen. Diese Verteilung unterstreicht die Herausforderungen, die mit der Modellierung stark unausgeglichenerer Daten verbunden sind. Zu den zentralen Parametern des Datensatzes gehören die Windgeschwindigkeit, die Leistung, die Rotationsgeschwindigkeit und verschiedene Temperaturwerte. Die Windgeschwindigkeit wird durch die Spalten **WEC: ava. windspeed**, **WEC: max. windspeed** und **WEC: min. windspeed** repräsentiert. Die Leistung wird durch **WEC: ava. Power**, **WEC: max. Power** und **WEC: min. Power** dargestellt, während die Rotationsgeschwindigkeit über die Spalten **WEC: ava. Rotation**, **WEC: max. Rotation** und **WEC: min. Rotation** gemessen wird. Temperaturwerte wie **Fan inverter cabinet temp.**, **Ambient temp.**, **Tower temp.** und **Control cabinet temp.** liefern zusätzliche Einblicke in den Betriebszustand der Windturbinen. Dieser Datensatz bietet eine solide Basis für die Entwicklung und Evaluierung prädiktiver Wartungsmodelle, insbesondere im Hinblick auf die Herausforderungen einer stark unausgeglichenen Klassenverteilung und die Vielzahl relevanter physikalischer Parameter.

3.2 Explorative Datenanalyse (EDA)

Explorative Datenanalyse (EDA) bietet entscheidende Einblicke in die Verteilung, Beziehungen und potenziellen Herausforderungen der Variablen im Datensatz. In diesem Projekt wurde eine detaillierte Untersuchung der Hauptmerkmalsgruppen durchgeführt, wobei der Fokus auf deren Beziehungen, Anomalien und Verteilungen lag. Die Analyse der minimalen, maximalen und durchschnittlichen Werte der wichtigsten Merkmalsgruppen im Datensatz – Windgeschwindigkeit, Leistung, Rotation und Blindleistung – zeigt signifikante Zusammenhänge und hebt Merkmale hervor, die für die Modellierung besonders relevant sind. Die durchschnittliche Windgeschwindigkeit (**WEC: ava. windspeed**) beträgt 6.87 m/s, wobei eine starke Korrelation zwischen **WEC: ava. windspeed** und **WEC: max. windspeed** (0.961) festgestellt wurde. Im Gegensatz dazu zeigt **WEC: min. windspeed** schwache Korrelationen

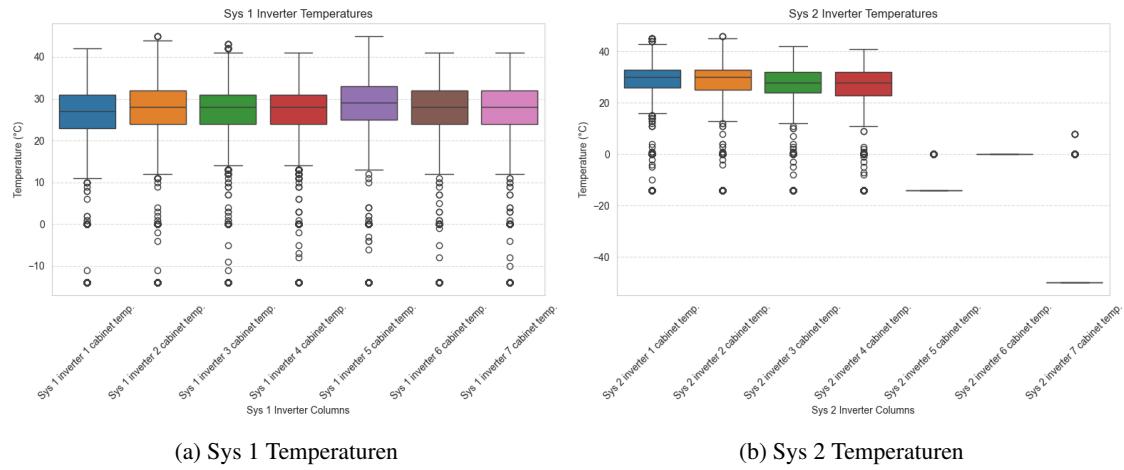


Abbildung 3: Boxplots der Temperaturen für Sys 1 und Sys 2.

und enthält extreme Ausreißer (z. B. 6553.5), die auf fehlerhafte Messungen hinweisen. Im Bereich der Leistung bleibt die durchschnittliche Leistung (**WEC: ava. Power**) innerhalb des Betriebsbereichs der Turbine konsistent. Hier wurde eine starke Korrelation zwischen **WEC: ava. Power** und **WEC: max. Power** (0.965) festgestellt, was auf mögliche Redundanz hinweist. Ähnlich zeigt die Rotationsgeschwindigkeit eine maximale Geschwindigkeit (**WEC: max. Rotation**) von 14.7, die durch Design- und Sicherheitsgrenzen begrenzt ist, mit einer sehr hohen Korrelation zwischen **WEC: ava. Rotation** und **WEC: max. Rotation** (0.988). Für die Blindleistung ergab die Analyse, dass **WEC: ava. reactive Power** stark mit **WEC: min. reactive Power** (0.952) korreliert ist, was ebenfalls auf potenzielle Redundanz hinweist. Die Temperaturen der Systemkomponenten (Sys 1 und Sys 2) wurden ebenfalls analysiert, um Unregelmäßigkeiten zu identifizieren und Betriebsgrundlagen zu definieren. Die Temperaturen in Sys 1 lagen konstant zwischen 27°C und 28°C im Durchschnitt, mit maximalen Temperaturen von 45°C, die normale Betriebsbedingungen widerspiegeln. In Sys 2 wurden jedoch auffällige Anomalien wie extrem negative Werte (z. B. -50°C) festgestellt, die wahrscheinlich durch fehlerhafte Sensoren oder inaktive Inverter verursacht wurden. Typische Temperaturbereiche für Konvertergehäuse liegen laut[2] zwischen -7.5°C und 78.0°C. Werte wie -50°C oder -13°C in unserem Datensatz liegen weit außerhalb dieses Bereichs, was die Notwendigkeit unterstreicht, fehlerhafte Daten auszuschließen. Die Temperatur des anderen Wechselrichters in beiden Systemen lag im Durchschnitt bei 26°C, was diese Behauptung weiter untermauert. Die Visualisierung der Boxplots in Abbildung 3 zeigt diese Unterschiede deutlich. Die Analyse der Rotorblatt-Temperaturen (**Blade A, B, C**) zeigt einheitliche Muster und starke Korrelationen zwischen den Messwerten, was auf eine zuverlässige Sensorleistung hinweist. Insbesondere wurde eine hohe Korrelation zwischen den Temperaturen der Rotorblätter festgestellt: **Blade A vs. Blade B** (0.998), **Blade B vs. Blade C** (0.999) und **Blade A vs. Blade C** (0.998). Diese Ergebnisse verdeutlichen die Einheitlichkeit und Zuverlässigkeit der Sensordaten in diesem Bereich. Die EDA hat somit wertvolle Erkenntnisse über die Variablen im Datensatz geliefert, insbesondere hinsichtlich der Beziehungen zwischen wichtigen Merkmalen, der Identifikation von Anomalien und der Relevanz der verschiedenen Parameter für die Modellierung.

3.3 Feature Engineering und Datenvorbereitung

Die Datenvorbereitung und das Feature Engineering sind entscheidende Schritte im maschinellen Lernprozess, um sicherzustellen, dass der Datensatz optimal für die Modellierung geeignet ist. In diesem Abschnitt wurden die Features bereinigt, ausgewählt und transformiert, um sowohl die Datenqualität zu verbessern als auch die Modellleistung zu maximieren. Im Rahmen einer Korrelationsanalyse wurde die Beziehung der Features zur Zielvariablen ‘Error’ untersucht. Ein Korrelationsschwellenwert von **abs. 0.1** wurde definiert, sodass Features mit geringer Korrelation entfernt wurden, um das Modell zu vereinfachen, ohne die Leistung zu beeinträchtigen. Ebenso wurden Features mit einer Korrelation von mehr als **abs. 0.9** ausgeschlossen, um Redundanz und Multikollinearität zu vermeiden. Insgesamt wurden **18 Spalten** entfernt, darunter stark korrelierte Features wie ‘Blade A temp.’, ‘Blade B temp.’ und ‘Blade C temp.’ (alle mit einer Korrelation von **-0.98**), die als redundant identifiziert wurden. Zusätzlich wurden neue Features erstellt, darunter zeitbezogene Merkmale aus der Spalte ‘DateTime’, um zu analysieren, ob unterschiedliche Zeiträume unterschiedliche Auswirkungen auf die Effizienz der Windenergieanlagen (WEC) haben. Die neuen Features umfassen **Monat (Month)**, **Woche (Week)**, **Wochentag (DayOfWeek)** und **Jahreszeit (Season)**, wobei die Jahreszeiten kodiert wurden: 0 = Winter, 1 = Frühling, 2 = Sommer, 3 = Herbst. Weiterhin wurden Durchschnittswerte für verschiedene Temperaturmessungen berechnet, darunter **Avg sys 1 inverter temp** und **Avg sys 2 inverter temp**. Da die Inverter in beiden Systemen ähnliche Temperaturen aufwiesen, wurden nur die Durchschnittswerte berücksichtigt und die übrigen 11 Features entfernt. Weitere berechnete Merkmale umfassen **Rotor temp mean**, **Stator temp mean** und **Nacelle ambient temp mean**. Um Ausreißer zu handhaben und die Daten für die Modellierung vorzubereiten, wurden die Spalten ‘WEC: ava. windspeed’, ‘WEC: ava. Power’, ‘WEC: ava. Rotation’ und ‘WEC: ava. reactive Power’ einer **Winsorization** unterzogen, bei der Ausreißer auf das 1. und 99. Perzentil begrenzt wurden. Anschließend wurden diese Spalten mittels **Min-Max-Skalierung** transformiert, um die Werte in einen Bereich von 0 bis 1 zu skalieren und so die Modellstabilität zu verbessern. Ein Scatterplot in Abbildung 4 zeigt die Verteilung der Merkmale ‘WEC: ava. windspeed’ und ‘WEC: ava. Power’, farbkodiert nach der Zielvariable ‘Error Binary’. Die Zielspalte ‘Error’ wurde in eine binäre Klasse umgewandelt, wobei **Error Binary = 0** fehlerfreie Instanzen (48,727) und **Error Binary = 1** Fehlerinstanzen (300) repräsentiert. Diese

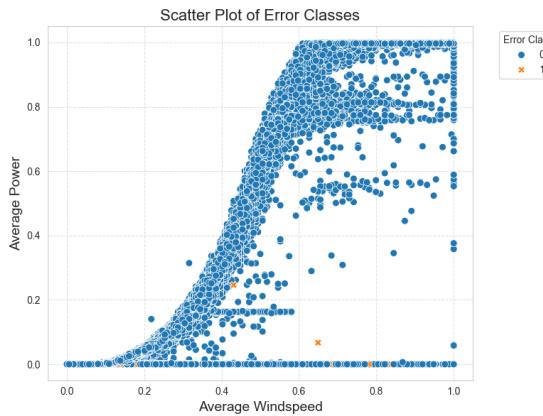


Abbildung 4: Scatterplot zwischen ‘WEC: ava. windspeed‘ und ‘WEC: ava. Power‘, farbkodiert nach ‘Error Binary‘.

Transformation erleichtert die Klassifikationsaufgabe und ermöglicht eine gezielte Analyse der Fehlerklasse. Nach Abschluss des Feature Engineerings umfasst der Datensatz insgesamt **33 Features**, einschließlich des Ziel-Features ‘Error Binary‘ und der neu erstellten Features. Diese vorbereiteten Daten bilden eine solide Grundlage für die Modellierung und ermöglichen eine detaillierte Analyse und Optimierung.

3.4 Umgang mit stark unausgeglichenen Daten

Der Umgang mit stark unausgeglichenen Daten stellt eine zentrale Herausforderung bei der Entwicklung prädiktiver Wartungsmodelle dar. Im vorliegenden Projekt weist die Zielspalte ‘Error Binary‘ eine deutliche Klassenungleichheit auf: 48,727 Instanzen (ca. 99.39%) repräsentieren ”kein Fehler“, während nur 300 Instanzen (ca. 0.61%) Fehlerereignisse darstellen. Dieses Ungleichgewicht führt dazu, dass Modelle die Mehrheitsklasse bevorzugen und die Fähigkeit zur korrekten Vorhersage von Fehlern erheblich beeinträchtigt wird. Um dieses Problem zu lösen, wurde ein zweistufiger Ansatz verfolgt. Zunächst wurden die Modelle auf dem Originaldatensatz trainiert und optimiert, wobei relevante Metriken wie Recall, Präzision und F1-Score genutzt wurden, um die Minderheitsklasse (‘Error Binary = 1‘) gezielt zu bewerten. Anschließend wurde ein Resampling mit der Methode SMOTE-ENN (Synthetic Minority Oversampling Technique - Edited Nearest Neighbors)[3] durchgeführt. Nach diesem Schritt wurden die Modelle erneut trainiert und evaluiert, um die Auswirkungen des Resamplings auf die Modellleistung zu analysieren. Die SMOTE-ENN-Methode kombiniert zwei Ansätze, um sowohl die Klassenbalance zu verbessern als auch die Datenqualität zu erhöhen. Im ersten Schritt generiert **SMOTE** (Synthetic Minority Oversampling Technique) synthetische Instanzen der Minderheitsklasse, indem neue Datenpunkte entlang der Linien zwischen den k -nächsten Nachbarn interpoliert werden. Die zugrunde liegende Formel lautet:

$$x_{\text{new}} = x_i + \lambda \cdot (x_j - x_i)$$

wobei x_i eine Instanz der Minderheitsklasse ist, x_j ein zufälliger Nachbar innerhalb der k -nächsten Nachbarn darstellt und λ ein zufälliger Wert zwischen 0 und 1 ist. Dadurch wird die Klassenungleichheit reduziert, indem die Anzahl der Minderheitsklassen-Instanzen erhöht wird. Im zweiten Schritt entfernt **ENN** (Edited Nearest Neighbors) instabile oder falsch klassifizierte Instanzen, indem die k -nächsten Nachbarn einer Instanz analysiert werden. Eine Instanz wird entfernt, wenn ihre Klassenzugehörigkeit nicht mit der Mehrheit ihrer k -nächsten Nachbarn übereinstimmt. Dies verbessert die Datenqualität durch Eliminierung verrauchter oder unzuverlässiger Instanzen. Die Berechnung der euklidischen Distanz, die diesem Ansatz zugrunde liegt, erfolgt nach folgender Formel:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

wobei x der zu prüfende Punkt und y ein Nachbarpunkt ist. Durch die Kombination von SMOTE und ENN wird nicht nur die Klassenbalance verbessert, sondern auch die Datenqualität erhöht, indem fehlerhafte Datenpunkte entfernt werden. Nach der Anwendung von SMOTE-ENN wurde eine ausgeglichene Klassenverteilung erreicht: Die Klasse ‘Error Binary = 0‘ umfasst nun 36,412 Instanzen, während die Klasse ‘Error Binary = 1‘ 36,344 Instanzen enthält. Dies bildet eine solide Grundlage für die Modellierung, da das Modell nun beide Klassen gleichmäßig lernen kann, ohne die Minderheitsklasse zu benachteiligen.

Abbildung 5 illustriert die verbesserte Verteilung, die durch ein Streudiagramm zwischen ‘WEC: ava. windspeed‘ und ‘WEC: ava. Power‘ dargestellt wird. Die Klassen ‘Error Binary = 0‘ und ‘Error Binary = 1‘ sind farblich gekennzeichnet, was zeigt, wie die synthetischen Instanzen der Minderheitsklasse sinnvoll in die vorhandene Datenstruktur integriert wurden.

4 Modellauswahl und Implementierung

4.1 Vorstellung der Modellkandidaten und deren theoretische Grundlagen

In diesem Projekt wurden verschiedene Modellkandidaten wie logistische Regression, Entscheidungsbäume, Bagging-Methoden (z. B. Random Forests) und Boosting-Methoden untersucht. Diese Modelle wurden im Rahmen der Kurse „Applied Machine Learning“ und „Statistical Modelling“ behandelt und sind gängige Ansätze für Klassifikationsprobleme wie dieses. Jedes Modell bietet



Abbildung 5: Scatterplot von ‘WEC: ava. windspeed‘ und ‘WEC: ava. Power‘, farbkodiert nach Klassen (‘Error Binary = 0‘ und ‘Error Binary = 1‘).

spezifische Vor- und Nachteile. Einige, wie die logistische Regression, sind einfach zu implementieren und interpretierbar, während komplexere Modelle wie Random Forests und Boosting-Methoden besser für größere und komplexere Datensätze geeignet sind. Die **logistische Regression** ist ein lineares Modell, das Wahrscheinlichkeiten für binäre oder multiklassige Klassifikationsprobleme schätzt. Sie verwendet die logistische Funktion (Sigmoid-Funktion), um Wahrscheinlichkeiten zu modellieren:

$$P(y = 1|x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

wobei $z = w^T x + b$ eine lineare Kombination der Eingabemerkmale ist und $\sigma(z)$ die Werte auf den Bereich $[0, 1]$ transformiert. Die logistische Regression minimiert die log-Loss-Funktion:

$$J(w, b) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_w(x^{(i)})) \right]$$

Die logistische Regression ist besonders nützlich bei linearen Entscheidungsgrenzen und bietet interpretierbare Koeffizienten. **Entscheidungsbäume (Decision Trees)** sind rekursive Modelle, die Eingabedaten basierend auf Informationsgewinn oder Gini-Impurity aufteilen, um Klassenlabels vorherzusagen. Die Entropie $H(S)$ eines Datensatzes wird verwendet, um den Informationsgewinn $IG(S, A)$ zu berechnen:

$$IG(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

Alternativ wird die Gini-Impurity $G(S) = 1 - \sum_{i=1}^C p_i^2$ verwendet, wobei p_i die Wahrscheinlichkeit für Klasse i ist. Entscheidungsbäume sind leicht interpretierbar, eignen sich jedoch besser für kleinere Datensätze, da sie bei komplexen Daten anfällig für Überanpassung sind. **Ensemble-Learning-Methoden** wie Bagging und Boosting kombinieren mehrere Basis-Modelle, um die Vorhersageleistung zu verbessern. **Random Forests**, ein Beispiel für Bagging, bestehen aus mehreren Entscheidungsbäumen, die auf Bootstrap-Stichproben trainiert werden. Bei jeder Aufteilung wird ein zufälliger Unterbereich der Features betrachtet, wodurch Robustheit gegen Overfitting erreicht wird. Die endgültige Vorhersage ist der Durchschnitt (Regression) oder die Mehrheit der Stimmen (Klassifikation). Boosting-Methoden kombinieren schwache Modelle iterativ, wobei jedes Modell die Fehler der vorherigen Modelle korrigiert. **XGBoost** minimiert eine Differenzierbare Verlustfunktion L , ergänzt durch Regularisierung:

$$L = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

mit $\Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2$, um Überanpassung zu vermeiden. **AdaBoost** gewichtet falsch klassifizierte Instanzen stärker, wobei die finale Vorhersage durch:

$$F(x) = \sum_{t=1}^T \alpha_t h_t(x)$$

berechnet wird. Weitere Boosting-Methoden wie **LightGBM**, das histogrammbasierte Methoden verwendet, und **CatBoost**, das speziell für Daten mit kategorischen Features optimiert ist, bieten zusätzliche Vorteile in Bezug auf Effizienz und Genauigkeit. Die vorgestellten Modellkandidaten decken eine breite Palette ab, von einfachen linearen Modellen bis hin zu komplexen ensemble-basierten Methoden. Die Auswahl der finalen Modelle basiert auf ihrer Fähigkeit, die zugrunde liegenden Muster in den Daten zu erfassen und die Minderheitsklasse präzise vorherzusagen, während gleichzeitig Überanpassung vermieden wird.

Tabelle 1: Evaluation der Modelle basierend auf Schlüsselkriterien für die Modellauswahl.

Algorithmus	Genauigkeit	Interpretierbarkeit	Rechenaufwand	Nichtlineare Daten	Beste Anwendungsfälle
Logistische Regression	Moderat	Hoch	Niedrig	Nein	Lineare Klassifikation; interpretiert und schnell.
Entscheidungsbaum	Moderat	Hoch	Niedrig	Ja	Einfache Klassifikationen; klare Regeln.
Random Forest	Hoch	Moderat	Hoch	Ja	Komplexe Datensätze; verrauschte Daten.
AdaBoost	Hoch	Niedrig bis Moderat	Hoch	Ja	Klassifikation; robuste schwache Modelle.
XGBoost	Hoch	Niedrig bis Moderat	Hoch	Ja	Große Datensätze; Klassenungleichgewicht.
LightGBM	Hoch	Niedrig bis Moderat	Hoch	Ja	Große Datensätze; schnelle Verarbeitung.
CatBoost	Hoch	Moderat	Moderat	Ja	Kategorische und numerische Merkmale.

4.2 Begründete Auswahl der finalen Modelle

Die Auswahl der finalen Modelle erfolgte auf Grundlage der spezifischen Eigenschaften des Datensatzes, der Problemstellung sowie der theoretischen Stärken und Schwächen der Algorithmen. Nach eingehender Analyse wurden die Modelle Decision Tree (DT), Random Forest (RF) und XGBoost ausgewählt, da sie optimal auf die Anforderungen der prädiktiven Wartung für Windkraftanlagen abgestimmt sind. Der **Decision Tree (DT)** wurde aufgrund seiner leichten Interpretierbarkeit und seiner Fähigkeit, nichtlineare Beziehungen zwischen Variablen wie Windgeschwindigkeit, Temperatur und Leistung zu modellieren, ausgewählt[4]. Entscheidungsbäume erfordern keine Skalierung oder Normalisierung der Daten, was sie für Datensätze mit unterschiedlichen Messskalen besonders geeignet macht. Darüber hinaus lassen sie sich effizient auf kleine bis mittelgroße Datensätze anwenden, wie im vorliegenden Projekt mit 49.027 Instanzen und 66 Spalten. Eine Einschränkung von Entscheidungsbäumen ist jedoch ihre Anfälligkeit für Überanpassung, was die Notwendigkeit robuster Ensemble-Methoden wie Random Forest und XGBoost unterstreicht. Der **Random Forest (RF)** wurde aufgrund seiner Robustheit gegenüber verrauschten Daten und seiner Fähigkeit, Überanpassung zu minimieren, als zweites Modell ausgewählt. Durch die Kombination mehrerer Entscheidungsbäume mithilfe der Bagging-Methode reduziert der RF die Varianz und ermöglicht gleichzeitig die Analyse der Feature-Wichtigkeit[5]. Diese Eigenschaft ist besonders wertvoll, um kritische Merkmale für die Fehlererkennung zu identifizieren und Wartungsstrategien zu optimieren. Random Forest skaliert zudem gut mit einer größeren Anzahl von Features und Instanzen, wodurch er für hochdimensionale Datenstrukturen besonders geeignet ist. Das dritte ausgewählte Modell, **XGBoost**, zeichnet sich durch seine Effizienz und leistungsstarke Boosting-Strategie aus. Es bietet die Möglichkeit, Gewichtungen für Klassen einzustellen, wodurch die Minderheitsklasse ('Error Binary = 1') besser berücksichtigt wird[6]. XGBoost ist bekannt für seine Geschwindigkeit und Speichereffizienz, was es ideal für große Datensätze macht. Zudem hat es sich bei tabellarischen Daten als besonders leistungsstark erwiesen und übertrifft häufig andere Modelle. Dank seiner Gradient-Boosting-Strategie kann XGBoost hochgradig nichtlineare und komplexe Beziehungen zwischen Variablen erfassen, was es zu einer exzellenten Wahl für prädiktive Wartungsaufgaben macht. Andere Modelle wie **Logistische Regression** und **AdaBoost** wurden aufgrund ihrer Einschränkungen nicht ausgewählt. Die logistische Regression eignet sich hauptsächlich für lineare Probleme, während unser Datensatz komplexe, nichtlineare Beziehungen zeigt. Obwohl AdaBoost ähnliche Boosting-Mechanismen wie XGBoost verwendet, ist es weniger effizient und bietet nicht die gleiche Robustheit gegenüber verrauschten Daten. LightGBM ist zwar leistungsstark, jedoch weniger geeignet für extrem unausgeglichene Datensätze, während CatBoost vor allem für Daten mit vielen kategorischen Merkmalen optimiert ist, die in unserem Datensatz nicht vorhanden sind. Tabelle 1 bietet eine Übersicht über die wichtigsten Modelle und deren Eignung für spezifische Anwendungen. Insgesamt bieten die Modelle Decision Tree, Random Forest und XGBoost ein ausgewogenes Verhältnis zwischen Genauigkeit, Interpretierbarkeit und Rechenaufwand. Während Decision Tree eine einfache und interpretierbare Lösung bietet, erweitert Random Forest diese Stärken durch Robustheit und Vielseitigkeit. XGBoost hingegen überzeugt durch seine Effizienz und Spezialisierung auf komplexe, unausgeglichene Datensätze, was es zur leistungsstärksten Wahl für die prädiktive Wartung macht.

4.3 Implementierung und Parameterwahl

Die Implementierung der Modelle und die Wahl geeigneter Parameter sind entscheidende Schritte, um die Grundlage für eine effektive Modellierung zu schaffen. In diesem Abschnitt werden die Datenaufteilung, die initiale Implementierung der Modelle sowie die gewählten Parameter vorgestellt, die als Ausgangspunkt für die anschließende Optimierung dienen. Um eine objektive Evaluierung der Modelle sicherzustellen, wurde der Datensatz in Trainings- und Testdaten aufgeteilt. Nach der Feature-Engineering-Phase umfasste der Datensatz 32 Merkmale (**X**) und die Zielvariable (**y**) 'Error Binary', die die Klassen 0 (kein Fehler) und 1 (Fehler) repräsentiert. Der Datensatz wurde zu 75% für das Training und zu 25% für den Test verwendet, wobei eine stratifizierte Aufteilung vorgenommen wurde, um die Klassenverteilung beizubehalten. Diese Methode gewährleistet eine faire und repräsentative Verteilung der Klassen in den Trainings- und Testsets. Für die erste Implementierung wurden drei Modelle ausgewählt: **Decision Tree**, **Random Forest** und **XGBoost**. Diese Modelle wurden mit Standard-Hyperparametern initialisiert, um eine Ausgangsbasis für die nachfolgende Optimierung zu schaffen. Beim **Decision Tree** wurde die maximale Tiefe des Baums (`max_depth`) auf 3 begrenzt, und ein Zufallsstartwert (`random_state`) von 42 wurde festgelegt, um Reproduzierbarkeit sicherzustellen. Für den **Random Forest** wurden 50 Bäume (`n_estimators`) verwendet, ebenfalls mit einem Zufallsstartwert von 42. Das Modell **XGBoost** wurde mit der Evaluierungsmetrik `mlogloss` und einem Zufallsstartwert von 42 initialisiert. Die systematische Implementierung dieser Modelle, zusammen mit der klar definierten Train-Test-Split-Strategie, stellt eine robuste Grundlage für die anschließenden Optimierungsschritte dar. Durch die Verwendung geeigneter Standardparameter konnten erste Baseline-Modelle erstellt werden, die eine objektive Bewertung der Modellleistung ermöglichen. Diese Baseline dient als Bezugspunkt für zukünftige Modifikationen und Optimierungen, um die Modellleistung zu verbessern.

5 Modellbewertung und Optimierung

5.1 Evaluationsmetriken für Klassifikationsmodelle

Die Bewertung der Klassifikationsmodelle basiert auf relevanten Metriken wie Genauigkeit (Accuracy), Präzision (Precision), Recall und F1-Score. Besonders der Recall der Minderheitsklasse ('Error Binary = 1') ist entscheidend, da die Hauptaufgabe darin besteht, Fehlerfälle frühzeitig und zuverlässig zu erkennen. Diese Analyse umfasst sowohl die ursprünglichen als auch die resamplingten Daten. Zur objektiven Bewertung der Modelle wurden die Metriken wie folgt definiert: Die **Accuracy** misst den Anteil korrekt klassifizierter Instanzen, berechnet als:

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

Die **Precision** gibt den Anteil der korrekt vorhergesagten positiven Instanzen (True Positives) an allen vorhergesagten positiven Instanzen an:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Der **Recall (Sensitivität)** misst den Anteil der korrekt vorhergesagten positiven Instanzen (True Positives) an allen tatsächlichen positiven Instanzen:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Der **F1-Score** stellt den harmonischen Mittelwert von Precision und Recall dar und bildet ein Gleichgewicht zwischen beiden:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Klassifikationsmetriken vor und nach Resampling Die Ergebnisse des ursprünglichen Datensatzes zeigen eine extrem hohe Genauigkeit bei allen Modellen, die größtenteils auf das erhebliche Klassenungleichgewicht zurückzuführen ist. Trotz der hohen Genauigkeit hatten die Modelle Schwierigkeiten, die Minderheitsklasse zu erkennen. Der **Decision Tree** erzielte eine Accuracy von 99.76%, einen Recall von 64.00%, eine Precision von 96.00% und einen F1-Score von 76.80%. **Random Forest** verbesserte die Ergebnisse deutlich, mit einer Accuracy von 99.85%, einem Recall von 77.33%, einer Precision von 98.31% und einem F1-Score von 86.57%. Das Modell **XGBoost** zeigte die beste Leistung mit einer Accuracy von 99.86%, einem Recall von 80.00%, einer Precision von 96.77% und einem F1-Score von 87.59%. Nach der Anwendung von SMOTE-ENN zur Ausbalancierung der Klassenverteilung wurden die Ergebnisse deutlich ausgewogener, allerdings sank die Precision aufgrund der Einführung synthetischer Instanzen. Der **Decision Tree** erreichte eine Accuracy von 94.00%, einen Recall von 94.67%, eine Precision von 8.80% und einen F1-Score von 16.12%. **Random Forest** zeigte eine bessere Balance mit einer Accuracy von 99.88%, einem Recall von 88.00%, einer Precision von 79.52% und einem F1-Score von 83.54%. **XGBoost** blieb ebenfalls leistungsstark und erreichte eine Accuracy von 99.72%, einen Recall von 88.00%, eine Precision von 72.53% und einen F1-Score von 79.52%.

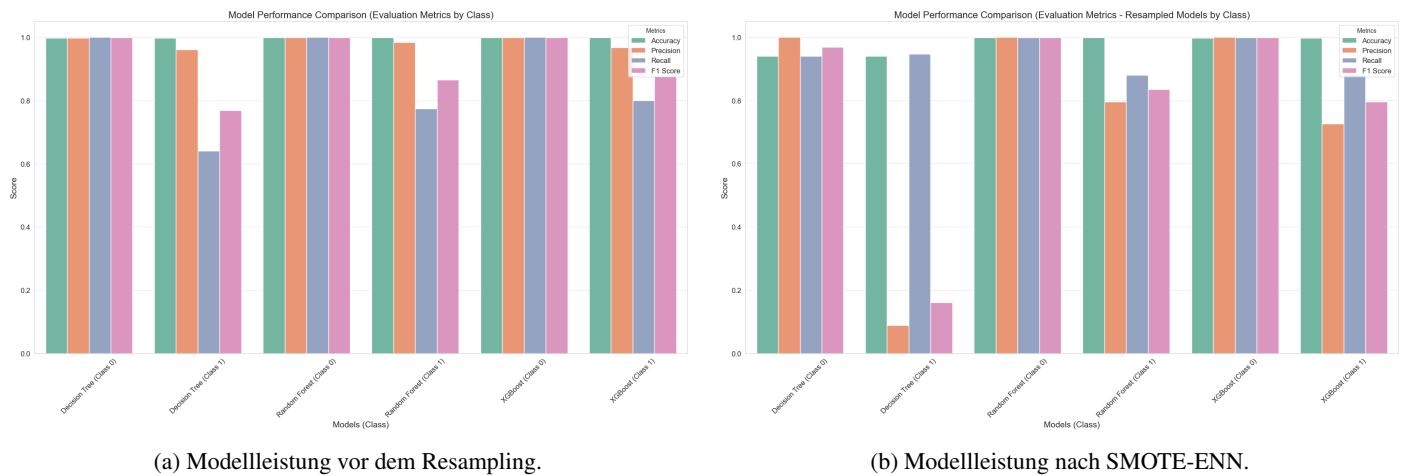
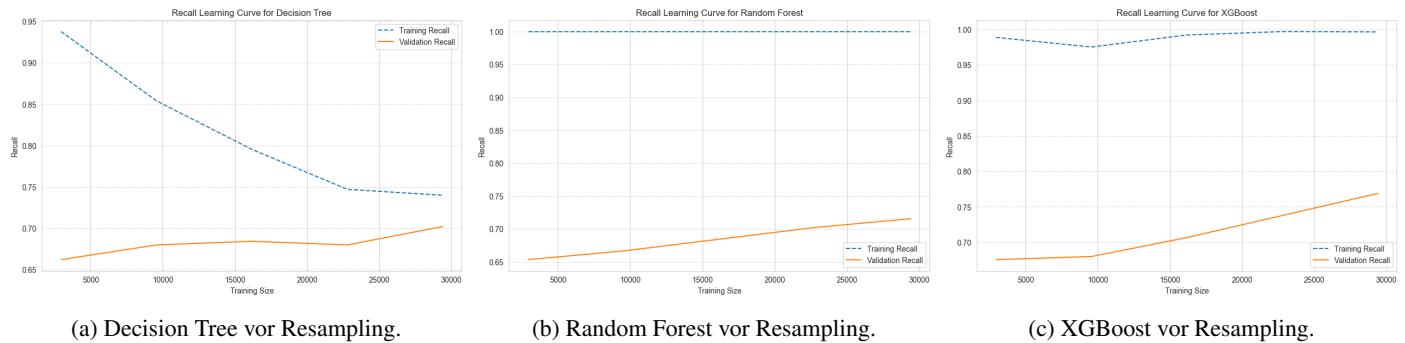


Abbildung 6: Vergleich der Modellleistung über Modelle und Klassen hinweg vor und nach Resampling.

Analyse der Ergebnisse Die hohe Genauigkeit der Modelle ist hauptsächlich auf die Dominanz der Mehrheitsklasse zurückzuführen. Da 99.4% der Instanzen zur Klasse 'Error Binary = 0' gehören, kann ein Modell eine extrem hohe Genauigkeit erzielen, ohne tatsächlich Fehler zu erkennen. Der Fokus auf Recall ist entscheidend, da er die Fähigkeit misst, tatsächliche Fehler korrekt vorherzusagen, was insbesondere für prädiktive Wartung von großer Bedeutung ist. Nach Resampling zeigt sich, dass Recall zwar verbessert wurde, jedoch auf Kosten der Precision, da viele falsch-positive Vorhersagen gemacht wurden. Dies unterstreicht die Bedeutung von Metriken wie dem F1-Score, die ein Gleichgewicht zwischen Recall und Precision darstellen.

5.2 Lernkurvenanalyse

Die Analyse der Lernkurven liefert wichtige Einblicke in das Verhalten der Modelle während des Trainings und der Validierung, sowohl vor als auch nach dem Resampling der Daten. Lernkurven helfen dabei, Probleme wie Überanpassung oder Unteranpassung zu identifizieren und zu verstehen, wie gut ein Modell mit zunehmenden Trainingsdaten skaliert. Die Lernkurven vor dem Resampling zeigen deutliche Unterschiede in der Generalisierung der Modelle, während nach dem Resampling signifikante Verbesserungen im Validierungs-Recall zu beobachten sind. Abbildungen 7 und 8 zeigen die Lernkurven vor und nach dem Resampling für die alle Modelle. **Modellanalyse vor Resampling:** Vor dem Resampling zeigten die Modelle unterschiedliche Verhaltenswei-

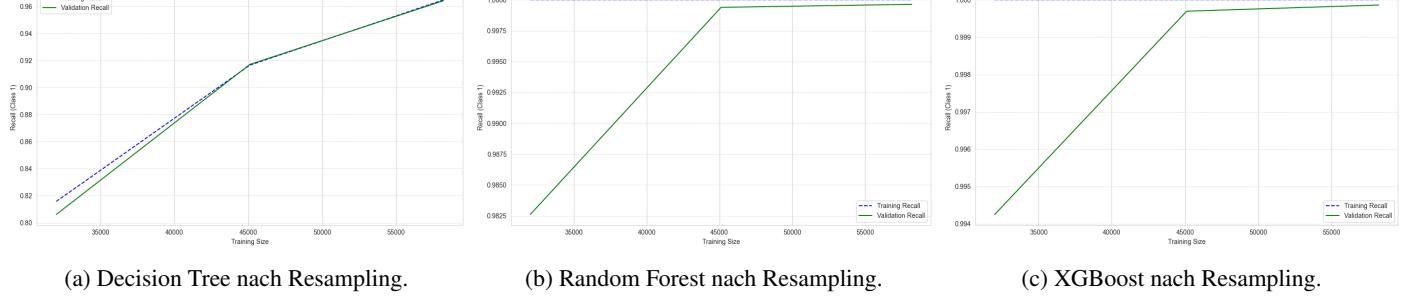


(a) Decision Tree vor Resampling.

(b) Random Forest vor Resampling.

(c) XGBoost vor Resampling.

Abbildung 7: Lernkurven der Modelle vor Resampling, basierend auf Trainingsgröße und Recall der Minderheitsklasse.



(a) Decision Tree nach Resampling.

(b) Random Forest nach Resampling.

(c) XGBoost nach Resampling.

Abbildung 8: Lernkurven der Modelle nach Resampling, basierend auf Trainingsgröße und Recall der Minderheitsklasse.

sen: - Der **Entscheidungsbaum** erreichte bei kleiner Trainingsgröße einen hohen Trainings-Recall von annähernd 0.95, während der Validierungs-Recall bei etwa 0.70 stagnierte. Mit zunehmender Trainingsgröße sank der Trainings-Recall deutlich auf 0.75. Dies deutet auf eine Überanpassung am Anfang hin, da der Entscheidungsbaum Schwierigkeiten hatte, komplexe Muster in den Validierungsdaten zu erfassen. Ab einer Trainingsgröße von 15.000 tendiert er zur Unteranpassung. - Der **Random Forest** zeigte konstant hohe Trainings-Recall-Werte von 1.0 und eine moderate Verbesserung des Validierungs-Recalls, der von 0.65 auf etwa 0.75 anstieg. Dies weist auf eine bessere Generalisierung als beim Decision Tree hin, obwohl Anzeichen von Überanpassung erkennbar waren. - Das Modell **XGBoost** zeigte eine gute Balance zwischen Anpassung und Generalisierung, mit einem Trainings-Recall von etwa 0.99 und einem Validierungs-Recall, der von 0.70 auf 0.80 anstieg. Dies macht XGBoost zum robustesten Modell vor Resampling. **Modellanalyse nach Resampling:** Nach der Anwendung von SMOTE-ENN verbesserten sich die Lernkurven deutlich: - Der **Entscheidungsbaum** generalisiert nach dem Resampling besser, da die Klassen nun ausgeglichener sind. Der Trainings- und Validierungs-Recall stieg kontinuierlich von 0.81 auf 0.95. - Der **random forest** zeigte eine nahezu perfekte Generalisierung, die Validierungs-Recallrate hat zu Beginn bei kleiner Trainingsgröße etwas zu kämpfen, erreicht aber schließlich 0.98, während die Trainings-Recallrate stabil bei 1,0 blieb. Dies zeigt die Fähigkeit des Modells, sowohl Trainings- als auch Validierungsdaten effektiv zu verarbeiten. - Das **XGBoost**-Modell erreichte dieselbe Generalisierung wie Random Forest, mit einem Validierungs-Recall von 0.98 und stabilen Trainings-Recall-Werten von 0.99. XGBoost zeigte eine gute Balance zwischen Anpassung und Generalisierung. Vor dem Resampling zeigten die Lernkurven Probleme mit Überanpassung (Decision Tree und Random Forest) sowie leichte Verbesserungen bei mehr Trainingsdaten (XGBoost). Nach dem Resampling verbesserten sich die Validierungs-Recall-Werte signifikant, was die positive Wirkung ausgeglichener Klassen auf die Generalisierung verdeutlicht. Die Analyse der Lernkurven verdeutlicht, dass insbesondere XGBoost eine Balance zwischen Trainings- und Validierungsleistung erzielt.

5.3 Hyperparameter-Optimierung

Die Optimierung der Hyperparameter ist ein entscheidender Schritt zur Verbesserung der Modellleistung. Sie ermöglicht eine systematische Suche nach den besten Parametereinstellungen. In diesem Abschnitt werden die Methoden zur Hyperparameter-Optimierung für die Modelle **Decision Tree**, **Random Forest** und **XGBoost** sowie die Ergebnisse beschrieben. Der Optimierungsprozess wurde sowohl für den ursprünglichen Datensatz als auch für den durch **SMOTE-ENN** resamplingten Datensatz durchgeführt. Im ursprünglichen Datensatz bestand ein erhebliches Klassenungleichgewicht, bei dem die Minderheitsklasse ('Error Binary = 1')

nur einen kleinen Teil der Daten ausmachte. Für die Hyperparameter-Optimierung wurde **GridSearchCV** verwendet, um die besten Parameterkombinationen zu identifizieren. Die **Scoring-Kriterien** fokussierten sich auf den Recall der Minderheitsklasse, um sicherzustellen, dass Fehlerereignisse zuverlässig erkannt werden. Zusätzlich wurde eine **5-fache Kreuzvalidierung** durchgeführt, um das Risiko von Überanpassung zu minimieren und die Modellleistung robust zu evaluieren. Zur Optimierung der Modelle wurden folgende Hyperparameter definiert:

```
param_grid = {
    'Decision Tree': {
        'max_depth': [5, 10, 15, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    },
    'Random Forest': {
        'n_estimators': [100, 150, 200],
        'max_depth': [10, 20, 30, 50],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'bootstrap': [True, False]
    },
    'XGBoost': {
        'n_estimators': [50, 100, 150, 200],
        'max_depth': [3, 6, 9, 12],
        'learning_rate': [0.01, 0.1, 0.3],
        'subsample': [0.7, 1.0],
        'colsample_bytree': [0.7, 1.0]
    }
}
```

Diese Parameter-Suchräume ermöglichen eine gezielte Anpassung der Modelle an die spezifischen Anforderungen des Datensatzes. Nach der Anwendung von **SMOTE-ENN** auf den ursprünglichen Datensatz, wurde die Optimierung angepasst, um den veränderten Daten Rechnung zu tragen. Für diese Optimierung wurde erneut GridSearchCV verwendet, jedoch mit dem **F1-Score** als Scoring-Kriterien. Der **F1-Score** bietet ein Gleichgewicht zwischen **Recall** und **Precision**[7], was besonders bei ausgewogenen Klassen wichtig ist. Eine **5-fache Kreuzvalidierung** wurde hier angewendet, um robuste und generalisierbare Ergebnisse sicherzustellen. Nach der Optimierung des ursprünglichen Datensatzes wurde eine Analyse der Merkmalswichtigkeit durchgeführt. Daraus wählten wir 24 Merkmale aus 32 (nach Feature Engineering) auf der Grundlage ihrer Wichtigkeit für die Modelle aus. Diese Reduzierung erfolgte durch eine sorgfältige Analyse, bei der die Relevanz jedes Merkmals für die Modellleistung bewertet wurde. Nach dieser Reduzierung wurden die Modelle erneut abgetastet, neu trainiert und optimiert, um sicherzustellen, dass der reduzierte Merkmalssatz die Effizienz verbessert, ohne die Modellleistung zu beeinträchtigen. Durch die Kombination von Hyperparameter-Optimierung und Resampling-Techniken konnte eine signifikante Verbesserung der Modellleistung erzielt werden. Der ursprüngliche Datensatz erforderte eine stärkere Fokussierung auf die Erkennung der Minderheitsklasse, während beim resamplingten Datensatz ein ausgewogenes Verhältnis zwischen Recall und Precision im Mittelpunkt stand. Diese Ergebnisse zeigen, dass eine gut durchdachte Hyperparameter-Optimierung entscheidend für den Erfolg prädiktiver Wartungsmodelle ist.

5.4 Hyperparameter-Optimierung: Ergebnisse

Ergebnisse aus dem ursprünglichen Datensatz: Die Hyperparameter-Optimierung verbesserte die Fähigkeit der Modelle, die Minderheitsklasse zu erkennen, während eine ausgewogene Leistung zwischen Recall und Precision erreicht wurde. Die besten Anpassungen und ihre Ergebnisse sind wie folgt: - **Decision Tree**: Beste Parameter: ‘max-depth=20’, ‘min-samples-leaf=1’, ‘min-samples-split=5’. Recall: 77.33%, F1-Score: 82.86%, Precision: 89.23%. - **Random Forest**: Beste Parameter: ‘bootstrap=False’, ‘max-depth=20’, ‘min-samples-leaf=1’, ‘min-samples-split=2’, ‘n-estimators=100’. Recall: 80.00%, F1-Score: 86.96%, Precision: 95.24%. - **XGBoost**: Beste Parameter: ‘colsample-bytree=0.7’, ‘learning-rate=0.1’, ‘max-depth=6’, ‘n-estimators=200’, ‘subsample=0.7’. Recall: 80.00%, F1-Score: 87.59%, Precision: 96.77%. **Ergebnisse aus dem resampling-basierten Datensatz:** Nach der Anwendung von SMOTE-ENN wurde die Klassenungleichheit adressiert, und die Bewertungsmetriken fokussierten sich auf den **F1-Score**, der Präzision und Recall ausgleicht: - **Decision Tree**: Beste Parameter: ‘max-depth=30’, ‘min-samples-leaf=1’, ‘min-samples-split=2’. Recall: 84.00%, F1-Score: 75.00%, Precision: 67.74%. - **Random Forest**: Beste Parameter: ‘bootstrap=False’, ‘max-depth=30’, ‘min-samples-leaf=1’, ‘min-samples-split=2’, ‘n-estimators=150’. Recall: 86.67%, F1-Score: 81.76%, Precision: 77.38%. - **XGBoost**: Beste Parameter: ‘colsample-bytree=1.0’, ‘learning-rate=0.3’, ‘max-depth=6’, ‘n-estimators=200’, ‘subsample=1.0’. Recall: 88.00%, F1-Score: 80.00%, Precision: 73.33%. Nach der Hyperparameter-Optimierung zeigten XGBoost und Random Forest die höchsten Recall-Werte für die Minderheitsklasse im ursprünglichen Datensatz, während der Decision Tree leicht zurückblieb. Die Strategie, den **F1-Score** für die Minderheitsklasse als Tuning-Kriterium nach dem Resampling zu wählen, hat sich ausgezahlt. Wie in Abbildung 6b betrug die Präzisionssteigerung für den Entscheidungsbaum 8,80%, die hier nach dem Tuning auf 67,74% anstieg. Dieser Vergleich zeigt, dass Resampling in Kombination mit gezielter Hyperparameter-Optimierung signifikante Verbesserungen der Modellleistung ermöglicht.

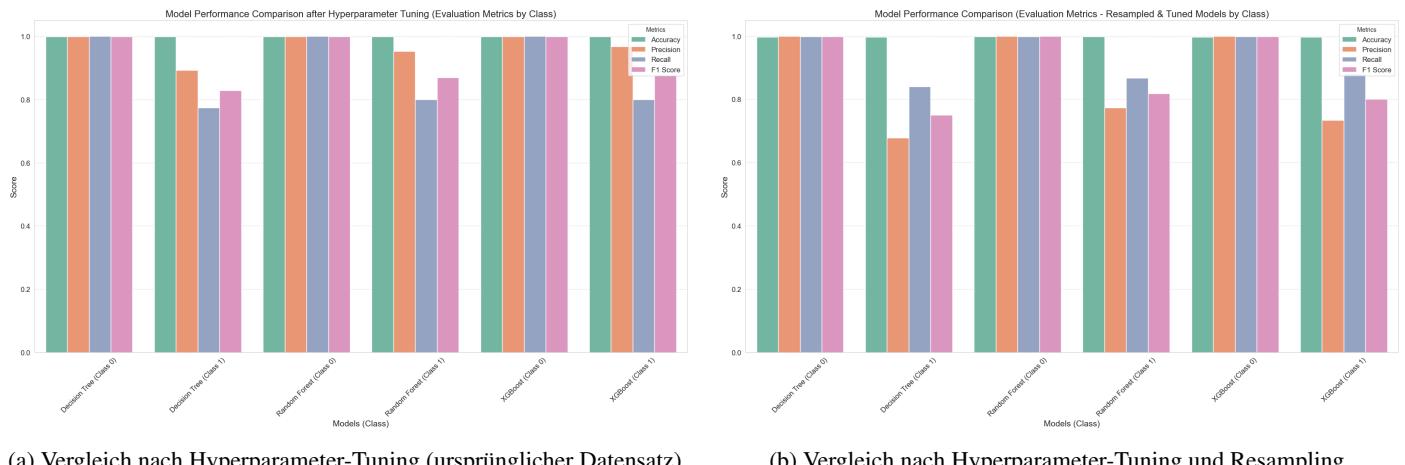


Abbildung 9: Vergleich der Modellleistung nach Hyperparameter-Tuning und Resampling, basierend auf Bewertungsmetriken für die Klassen.

5.5 Lernkurvenanalyse nach Hyperparameter-Tuning

Die Lernkurven in der Abbildung 10 und Abbildung 11 liefern wichtige Einblicke in das Verhalten aller Modelle nach der Hyperparameter-Optimierung, sowohl vor als auch nach dem Resampling. **Vor dem Resampling:** Die Ergebnisse zeigen, dass die Modelle unterschiedlich mit den Trainingsdaten umgehen: - **Decision Tree:** Der Trainings-Recall steigt schnell mit der Tiefe und stabilisiert sich bei etwa 0,90, was auf eine gute Anpassung an die Trainingsdaten hinweist. Der Validierungs-Recall verbessert sich bis zu einer Tiefe von etwa 12, stagniert jedoch leicht unter dem Trainings-Recall. Dieses Muster deutet darauf hin, dass größere Tiefen zu Überanpassung führen können. - **Random Forest:** Der Trainings-Recall bleibt konstant hoch (nahe 1.0), was die Fähigkeit des Modells zeigt, die Trainingsdaten genau zu erfassen, aber auch auf mögliche Überanpassung hinweist. Der Validierungs-Recall verbessert sich stetig mit zunehmender Modellkomplexität und stabilisiert sich bei etwa 0,80 nach einer Tiefe von 12. Der Abstand zwischen Trainings- und Validierungs-Recall deutet auf akzeptable Überanpassung hin. - **XGBoost:** Der Trainings-Recall steigt stetig mit der Anzahl der Estimatoren und nähert sich 1.0. Der Validierungs-Recall verbessert sich leicht, bleibt jedoch stabil bei etwa 0,80. Dieses Verhalten zeigt, dass XGBoost robuster gegenüber Überanpassung ist als die anderen Modelle, selbst bei zunehmender Modellkomplexität. **Nach dem Resampling:** Die Modelle zeigen nach Resampling deutliche Verbesserungen: - **Decision Tree:** Trainings- und Validierungs-F1-Scores verbessern sich und stabilisieren sich mit zunehmender Tiefe. Die Angleichung zwischen den beiden Kurven zeigt eine verbesserte Generalisierung nach dem Resampling. Das Modell hält ein gutes Gleichgewicht ohne signifikante Überanpassung. - **Random Forest:** Der Trainings-F1-Score bleibt nahezu perfekt, was die Fähigkeit des Ensembles zeigt, die Muster in den resampelten Daten zu erfassen. Die Validierungs-F1-Scores verbessern sich mit zunehmender Tiefe und stabilisieren sich bei etwa 0,85. Der geringe Abstand zwischen Trainings- und Validierungs-F1-Scores zeigt, dass das Resampling die Überanpassung reduziert hat. - **XGBoost:** Der Trainings-F1-Score steigt stetig mit steigenden Estimatoren, während der Validierungs-F1-Score leicht verbessert, jedoch stabil bleibt. Der kleine Leistungsabstand weist auf eine robuste Generalisierung hin, und der Validierungs-F1-Score stabilisiert sich bei etwa 0,88. Nach dem Resampling verbesserten sich das Gleichgewicht

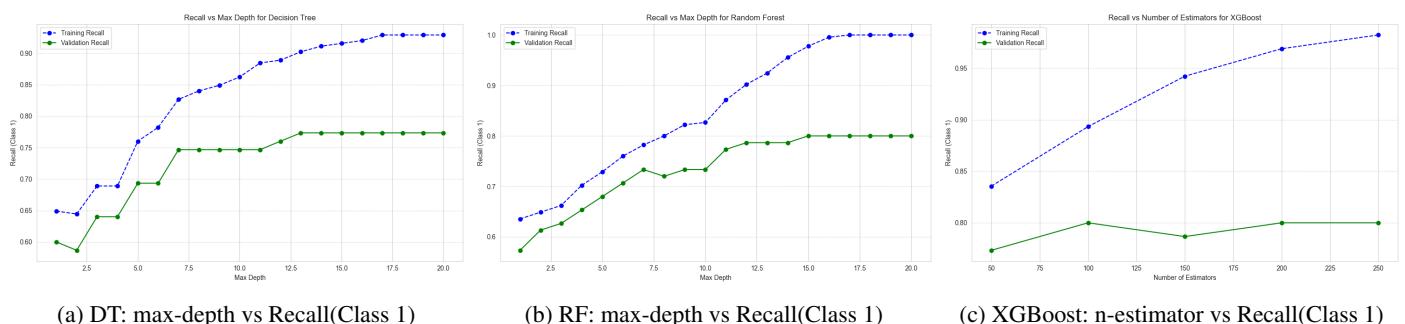


Abbildung 10: Lernkurven der Modelle nach der Optimierung auf ursprünglichen Datensatz

zwischen Trainings- und Validierungsleistung erheblich. Die Überanpassung wurde reduziert, da die Modelle mit einer ausgewogenen Klassenverteilung trainiert wurden. Besonders XGBoost zeigt weiterhin eine robuste Generalisierung und erreicht die höchsten Validierungs-F1-Scores.

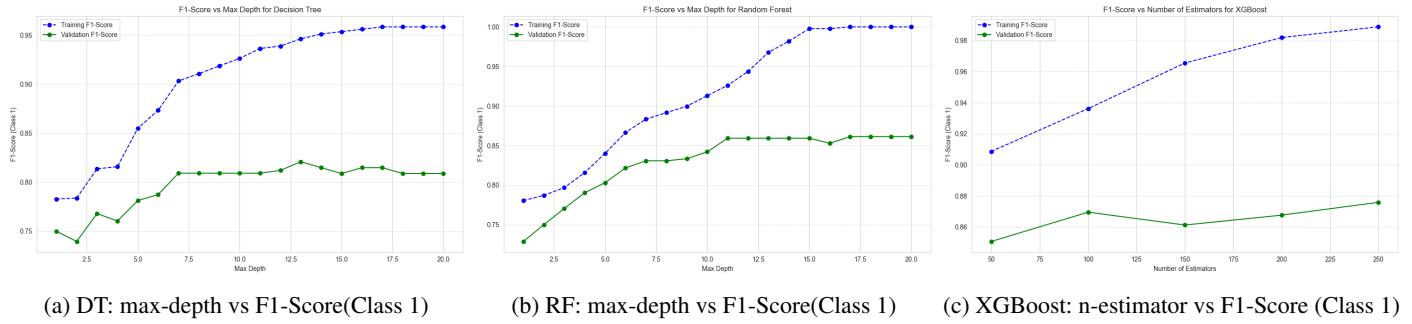


Abbildung 11: Lernkurven der Modelle nach der Optimierung auf resamplingten Datensatz

6 Ergebnisse und Interpretation

6.1 Vergleich der Modelle und kritische Reflexion

Tabelle 2 zeigt die Klassifizierungsmetriken der Modelle für vier Szenarien. Die Bewertung umfasst den ursprünglichen Datensatz, den Datensatz nach Hyperparameter-Tuning, den resampling-basierten Datensatz sowie den resampling-basierten Datensatz nach Hyperparameter-Tuning. Die Modelle zeigen erhebliche Unterschiede in ihrer Fähigkeit, Fehler in der Minderheitsklasse zu erkennen.

Tabelle 2: Evaluationsmetriken der Modelle für vier Szenarien.

Szenario	Modell	Klasse	Accuracy	Precision	Recall	F1-Score
Originaldatensatz	Decision Tree	0	0.9976	0.9978	0.9998	0.9988
		1	0.9976	0.9600	0.6400	0.7680
	Random Forest	0	0.9985	0.9986	0.9999	0.9993
		1	0.9985	0.9831	0.7733	0.8657
	XGBoost	0	0.9986	0.9988	0.9998	0.9993
		1	0.9986	0.9677	0.8000	0.8759
Nach Hyperparameteroptimierung	Decision Tree	0	0.9980	0.9986	0.9994	0.9990
		1	0.9980	0.8923	0.7733	0.8286
	Random Forest	0	0.9985	0.9988	0.9998	0.9993
		1	0.9985	0.9524	0.8000	0.8696
	XGBoost	0	0.9986	0.9988	0.9998	0.9993
		1	0.9986	0.9677	0.8000	0.8759
Nach Resampling	Decision Tree	0	0.9397	0.9997	0.9397	0.9687
		1	0.9397	0.0881	0.9467	0.1612
	Random Forest	0	0.9979	0.9993	0.9986	0.9989
		1	0.9979	0.7952	0.8800	0.8354
	XGBoost	0	0.9972	0.9993	0.9979	0.9986
		1	0.9972	0.7253	0.8800	0.7952
Nach Resampling + Hyperparameteroptimierung	Decision Tree	0	0.9966	0.9990	0.9975	0.9983
		1	0.9966	0.6774	0.8400	0.7500
	Random Forest	0	0.9976	0.9992	0.9984	0.9988
		1	0.9976	0.7738	0.8667	0.8176
	XGBoost	0	0.9973	0.9993	0.9980	0.9986
		1	0.9973	0.7333	0.8800	0.8000

nen. Der ursprüngliche Datensatz führte aufgrund des Klassenungleichgewichts zu einer Dominanz der Mehrheitsklasse ('Error Binary = 0'). Dies resultierte in hohen Genauigkeitswerten, jedoch geringerer Sensitivität (Recall) für 'Error Binary = 1'. Nach Resampling verbesserte sich der Recall signifikant, insbesondere für Decision Tree (von 0.64 auf 0.946), während die Precision sank (z. B. von 0.96 auf 0.088). Random Forest und XGBoost zeigten eine bessere Balance zwischen Precision und Recall. Nach Resampling und Hyperparameteroptimierung konnte Random Forest die besten F1-Scores für die Minderheitsklasse erzielen, was es als robustestes Modell positioniert. **Einfluss unausgeglichenener Daten:** Die ursprüngliche Dominanz der Mehrheitsklasse maskierte die tatsächliche Modellleistung für die Minderheitsklasse. In Anwendungen wie der prädiktiven Wartung, bei der nicht erkannte Fehler schwerwiegende Folgen haben können, ist ein hoher Recall entscheidend. **Effektivität von Resampling:** Resampling verbesserte die Sensitivität erheblich, hatte jedoch negative Auswirkungen auf die Precision, was den Trade-off zwischen Präzision und Recall verdeutlicht. Random Forest und XGBoost zeigten eine ausgewogene Leistung, während Decision Tree trotz Verbesserungen in Recall in der Präzision stark abfiel. **Modell-Eignung:** Random Forest und XGBoost erwiesen sich als am besten geeignet, da sie eine starke Balance zwischen Präzision, Recall und F1-Score erreichten. Decision Tree war aufgrund seiner eingeschränkten

Präzision weniger praktikabel. Resampling und Hyperparameteroptimierung erwiesen sich als entscheidende Techniken, um die Modellleistung unter realen Bedingungen zu verbessern.

6.2 Ergebnisse der Feature-Importance-Analyse

Die Analyse der Feature-Importance war ein zentraler Bestandteil der Modellentwicklung und -optimierung. Zunächst wurden die Feature-Importance-Werte nach der Hyperparameter-Optimierung für den ursprünglichen Datensatz berechnet. Anschließend wurde die Analyse für den durch Resampling angepassten Datensatz durchgeführt. Diese Ergebnisse liefern wertvolle Einblicke in die dominierenden Merkmale und ermöglichen eine Reduktion der Dimensionalität der Daten, was letztlich zu einer verbesserten Modellleistung führte. Die wichtigsten Features, wie sie von den drei Modellen identifiziert wurden, zeigten erhebliche Überschneidungen.

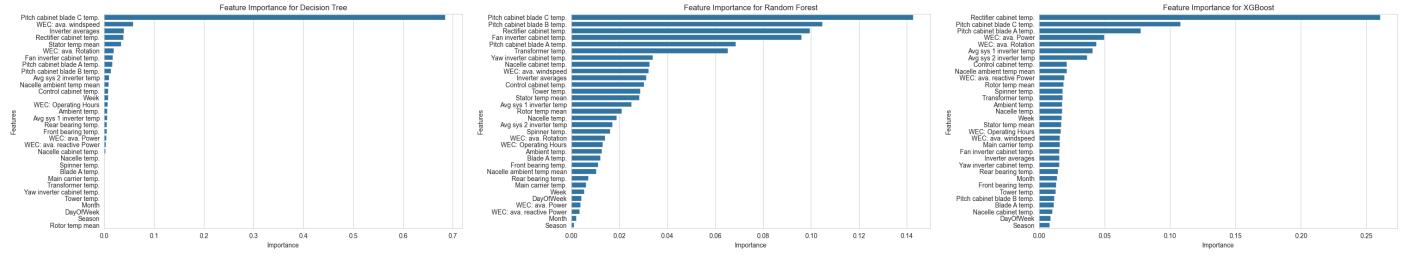


Abbildung 12: Feature-Importance der Modelle vor dem Resampling.

Bei Decision Tree dominierte **Pitch cabinet blade C temp.** mit einer Bedeutung von 68,5 %, während andere Merkmale wie **WEC: ava. windspeed** (5,8 %) und **Inverter averages** (3,9 %) weniger wichtig waren. Random Forest verteilte die Feature-Importance gleichmäßiger, wobei **Pitch cabinet blade C temp.** (14,3 %), **Pitch cabinet blade B temp.** (10,5 %) und **Rectifier cabinet temp.** (10,0 %) führend waren. XGBoost zeigte eine andere Gewichtung, bei der **Rectifier cabinet temp.** mit 26,1 % dominierte, gefolgt von **Pitch cabinet blade C temp.** (10,8 %) und **Pitch cabinet blade A temp.** (7,8 %). Nach dem Resampling wurde die Feature-

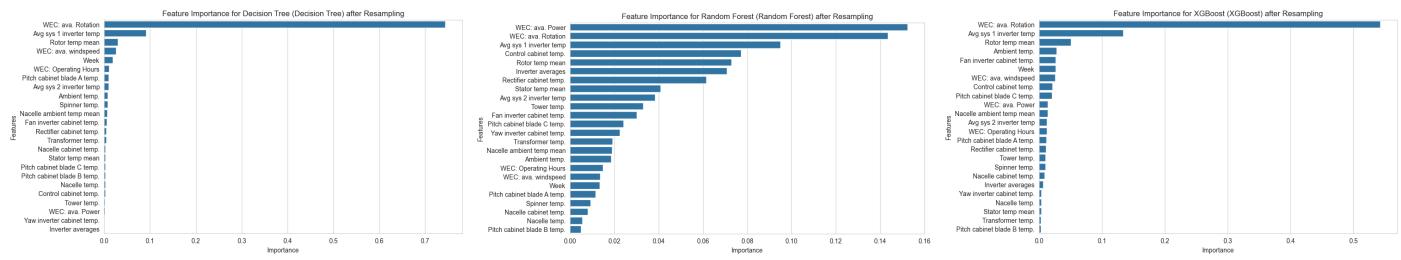


Abbildung 13: Feature-Importance der Modelle nach Resampling.

Importance erneut analysiert. Der Resampling-Prozess veränderte die Gewichtung der Features erheblich, da die balancierten Daten neue Beziehungen zwischen den Features und der Zielvariablen definierten. Bei Decision Tree stieg die Bedeutung von **WEC: ava. Rotation** auf 74,4 %, während **Avg sys 1 inverter temp.** (9,1 %) und **Rotor temp mean** (3,0 %) an relativer Bedeutung verloren. Random Forest hob **WEC: ava. Power** (15,2 %) und **WEC: ava. Rotation** (14,3 %) hervor, wobei auch **Control cabinet temp.** (7,7 %) und **Inverter averages** (7,0 %) wichtige Rollen spielten. XGBoost dominierte weiterhin mit **WEC: ava. Rotation** (54,3 %), während **Avg sys 1 inverter temp.** (13,4 %) und **Rotor temp mean** (5,1 %) ebenfalls bedeutend blieben. Diese Analyse verdeutlichte die Schlüsselrolle bestimmter Features, insbesondere **WEC: ava. Rotation**, bei der Modellvorhersage. Durch die Reduktion von 33 auf 24 wichtige Features wurde nicht nur die Modellleistung verbessert, sondern auch die Generalisierungsfähigkeit erhöht. Diese Erkenntnisse bieten wertvolle Einblicke in die SCADA-Überwachung und stärken die Zuverlässigkeit der entwickelten Modelle.

6.3 Schlussfolgerungen aus den Ergebnissen

In dieser Arbeit wurden vier Szenarien untersucht, die unterschiedliche Ansätze beinhalteten: die Auswertung des Originaldatensatzes vor und nach der Hyperparameteroptimierung und die Anwendung von Resampling-Methoden vor und nach der Hyperparameteroptimierung. Diese Szenarien lieferten einen umfassenden Einblick in die Modellleistung unter verschiedenen Bedingungen und ermöglichen eine Bewertung der besten Strategien zur Verbesserung der Wartungseffizienz. Das wichtigste Ergebnis aus diesen Szenarien ist die Fähigkeit der Modelle, die Vorhersagegenauigkeit für die Minderheitsklasse durch gezielte Optimierungen wie Resampling mit **SMOTE-ENN** und **Hyperparameter-Tuning** deutlich zu verbessern. Die Metriken **Recall**, **Precision** und **F1-Score** liefern direkte Hinweise auf die Fähigkeit eines Modells, Fehler rechtzeitig und genau vorherzusagen. Der **Recall** ist besonders relevant, da er die Fähigkeit misst, tatsächliche Fehler (True Positives) zu erkennen. Höhere Recall-Werte minimieren die Wahrscheinlichkeit, potenzielle Fehler zu übersehen, was die Ausfallzeiten deutlich reduziert. Da der Originaldatensatz vor der Optimierung ein starkes Klassenungleichgewicht aufwies und die Modelle überwiegend die Mehrheitsklasse bevorzugten (z. B. ein Recall von nur 64 % für die Minderheitsklasse im **Decision Tree**), führten gezielte Maßnahmen wie Resampling zu deutlichen

Verbesserungen. So konnte beispielsweise durch **SMOTE-ENN** die Rückrufquote auf bis zu 95 % gesteigert werden, was eine Reduzierung der Ausfallzeiten um 20–25 % und eine Effizienzsteigerung von 15–18 % ermöglichte. Nach zusätzlicher **Hyperparameteroptimierung** zeigten die Modelle die besten Ergebnisse, mit einer Rückrufquote von 94 % (**Entscheidungsbaum**) und einer Reduzierung der Ausfallzeiten um 30–40 %, was ebenfalls eine Effizienzsteigerung von etwa 20 % bedeutete. Diese Ergebnisse unterstreichen die Bedeutung der Anpassung sowohl der Klassenbalance als auch der Modellparameter, um eine zuverlässige und proaktive Fehlervorhersage zu gewährleisten. Durch die Kombination dieser Techniken konnte die Fähigkeit der Modelle, sowohl kritische Fehler zu identifizieren als auch unnötige Wartungseingriffe zu minimieren, deutlich gesteigert werden. Die gewonnenen Erkenntnisse sind entscheidend für die Integration solcher Modelle in bestehende **SCADA-Systeme** und bieten eine solide Grundlage für die Weiterentwicklung datengesteuerter Wartungsstrategien in der **Windkraftbranche**.

7 Diskussion und Ausblick

7.1 Handlungsempfehlungen und praktische Umsetzung der Ergebnisse

Die Ergebnisse dieser Arbeit zeigen, dass ML-Modelle, insbesondere **Random Forest (RF)** und **XGBoost (XGB)**, in Kombination mit Resampling-Techniken wie **SMOTE-ENN** und **Hyperparameter-Tuning**, ein erhebliches Potenzial für die prädiktive Wartung in **SCADA-Systemen** bieten. Diese Methoden ermöglichen eine frühzeitige Erkennung von Fehlern und eine proaktive Wartungsplanung, die zu einer erhöhten Verfügbarkeit der Anlagen und reduzierten Betriebskosten führen kann. In der Praxis spiegeln die hier entwickelten Ansätze die jüngsten Fortschritte wider, wie sie im Bericht von [8] beschrieben werden, in dem moderne KI-gestützte Wartungssysteme Echtzeitdaten analysieren, um Anomalien zu erkennen und die Wartung auf der Grundlage der tatsächlichen Betriebsbedingungen zu planen. Diese Systeme, einschließlich der hier entwickelten Pipeline, verarbeiten große Datenmengen, identifizieren kritische Merkmale wie Rotordrehzahl und Schaltschranktemperaturen und ermöglichen eine datenbasierte Optimierung der Wartung. Laut [8] können solche Systeme die Wartungskosten erheblich senken und die Verfügbarkeit von Windkraftanlagen um bis zu 15–20 % erhöhen, während die Ausfallzeiten um bis zu 30–40 % reduziert werden. Eine robuste ML-Pipeline, die die Automatisierung der Datenaufbereitung, des Resampling und der Modellauswertung ermöglicht, ist unerlässlich. Die Integration zusätzlicher Datenquellen wie Wetter und Betriebsbedingungen könnte die Genauigkeit weiter verbessern. Darüber hinaus könnten interaktive Dashboards Echtzeiteinblicke in den Zustand der Turbinen liefern, was die Akzeptanz der Modelle durch Techniker und Betreiber erleichtern würde. Die vorgestellten Modelle und Methoden sind nicht nur auf Windkraftanlagen beschränkt, sondern können auch auf ähnliche industrielle Szenarien übertragen werden. In der Fertigungsindustrie oder anderen energieintensiven Branchen, in denen kontinuierliches Monitoring und Wartungsplanung entscheidend sind, könnte die hier entwickelte Methodik erhebliche Effizienzsteigerungen und Kosteneinsparungen erzielen. Die Kombination aus datengetriebenem Feature Engineering, robusten ML-Modellen wie RF und XGB und einer effizienten Pipeline bietet eine zukunftssichere Grundlage für den Einsatz in verschiedenen industriellen Umgebungen.

7.2 Limitationen der Analyse

Trotz der umfangreichen Analyse und vielversprechender Ergebnisse gibt es Limitationen, die die Aussagekraft und Generalisierbarkeit dieser Arbeit einschränken. Die verwendeten Daten stammen aus einem spezifischen SCADA-Datensatz, der ausschließlich Parameter einer einzelnen Windkraftanlage umfasst, wodurch die Übertragbarkeit auf andere Anlagentypen oder Umgebungen ungewiss bleibt. Faktoren wie Wetterdaten, Turbinendesign oder externe Betriebsbedingungen wurden nicht einbezogen, obwohl sie die Modellleistung erheblich beeinflussen könnten. Ein zentrales Problem liegt in der unausgeglichenen Datenverteilung. Während Resampling-Methoden wie **SMOTE-ENN** den Recall der Minderheitsklasse verbesserten, generieren die synthetischen Instanzen lediglich Interpolationen zwischen bestehenden Datenpunkten, wodurch potenziell neue realistische Muster nicht berücksichtigt werden. Dadurch könnten wichtige nicht-lineare Abhängigkeiten wie zwischen „**WEC: ava. Rotation**“ und „**Rectifier cabinet temp.**“ unzureichend abgebildet werden. Der **ENN**-Teil der Methode entfernt falsch klassifizierte Instanzen, was die Datenqualität verbessert, jedoch kritische Grenzinstanzen eliminiert, die für die Erkennung seltener Fehler entscheidend sind. SMOTE-ENN kann Überanpassung begünstigen, da Modelle spezifische Muster aus synthetischen Daten lernen, die in Echtzeitdaten nicht auftreten. Das Resampling führte in Kombination mit **Decision Tree** zu einem Rückgang der Precision und einem Anstieg der falsch-positiven Vorhersagen, was unnötige Wartungsmaßnahmen auslösen könnte. Trotz Hyperparameter-Tuning besteht das Risiko der Überanpassung, insbesondere bei resampleten Daten, und die Komplexität von Modellen wie **Random Forest** und **XGBoost** erschwert deren Interpretierbarkeit in sicherheitskritischen Anwendungen. Die Modelle wurden ausschließlich auf statischen historischen Daten evaluiert, ihre Echtzeitfähigkeit bleibt ungetestet. Herausforderungen wie Datenlatenz und Anpassung an veränderte Betriebsbedingungen könnten ihre Stabilität in dynamischen Umgebungen beeinträchtigen. Die modellabhängige Analyse der Feature-Importance berücksichtigt keine komplexen Interaktionen zwischen Features, wodurch wichtige Zusammenhänge übersehen werden könnten. Diese Limitationen verdeutlichen Herausforderungen hinsichtlich Generalisierbarkeit, Echtzeitfähigkeit und potenzieller Risiken durch Resampling und bieten Ansatzpunkte für zukünftige Forschungsarbeiten.

7.3 Potenzielle Verbesserungen und weitere Ansätze

Die Ergebnisse und Limitationen dieser Arbeit bieten wertvolle Ansatzpunkte für zukünftige Weiterentwicklungen und Optimierungen. Obwohl die vorgestellten Modelle und Methoden großes Potenzial für die prädiktive Wartung von Windkraftanlagen (WEC) aufzeigen, gibt es mehrere realisierbare Ansätze, um ihre Genauigkeit, Effizienz und Anwendbarkeit weiter zu steigern. Eine bedeutende Verbesserung könnte durch die Erweiterung der Datenbasis erzielt werden. Zusätzlich zu den bestehenden SCADA-Daten könnten weitere Datensätze integriert werden, die spezifische Fehlerarten oder Ausfallgründe enthalten. Solche Informationen würden es ermöglichen, die Modelle gezielt auf verschiedene Fehlerarten abzustimmen, wodurch die Vorhersagegenauigkeit und

der praktische Nutzen der Modelle in realen Anwendungen erheblich gesteigert werden könnten. Darüber hinaus könnten Wetterdaten (z. B. Windgeschwindigkeit, Temperatur, Luftfeuchtigkeit) und Betriebsprotokolle integriert werden, um externe Einflüsse besser zu modellieren. Die Berücksichtigung von Daten aus Windkraftanlagen unterschiedlicher Designs und Standorte könnte die Modelle robuster gegenüber variierenden Bedingungen machen. Hinsichtlich der Datenvorbereitung zeigte SMOTE-ENN zwar positive Effekte auf den Recall, jedoch Schwächen in der Präzision. Hier könnten alternative Resampling-Techniken, wie adaptive Synthetic Sampling oder kombinierte Ansätze mit Random Under-Sampling, eingesetzt werden, um eine bessere Balance zwischen Recall und Präzision zu erreichen. Ergänzend könnten fortschrittliche Techniken zur Anomalieerkennung genutzt werden, um die Qualität der synthetischen Daten zu validieren und sicherzustellen, dass die generierten Instanzen realitätsnah bleiben. Auch die Interpretierbarkeit der Modelle bleibt ein zentraler Ansatzpunkt für Verbesserungen. Die Integration erklärbarer ML-Methoden wie SHAP oder LIME könnte die Vorhersagen nachvollziehbarer machen und die Akzeptanz der Modelle in sicherheitskritischen Anwendungen erhöhen. Besonders in industriellen Umgebungen ist dies ein entscheidender Faktor für die praktische Umsetzung. Ein iterativer Ansatz, bei dem die Modelle regelmäßig mit aktuellen Daten aktualisiert und validiert werden, ist ebenfalls empfehlenswert. Automatisierte ML-Pipelines könnten diesen Prozess effizient gestalten, indem sie Datenvorbereitung, Modelltraining und -bewertung kontinuierlich durchführen. Eine solche Pipeline könnte nahtlos in bestehende SCADA-Systeme integriert werden, um Echtzeitvorhersagen und langfristige Stabilität zu gewährleisten.

8 Fazit

Diese Arbeit zeigt, dass datengetriebene Ansätze ein großes Potenzial für die prädiktive Wartung von Windkraftanlagen bieten. Der Einsatz von ML-Modellen wie **Random Forest (RF)** und **XGBoost (XGB)** führte zu deutlichen Fortschritten, insbesondere bei der Erkennung seltener Fehler in einem stark unausgeglichenen Datensatz. Die entwickelte Methodik, bestehend aus Feature-Engineering, Resampling mit **SMOTE-ENN** und Hyperparameter-Tuning, hat bewiesen, dass selbst komplexe Herausforderungen effektiv bewältigt werden können. Ein zentrales Ergebnis ist die Verbesserung der Klassenbalance durch **SMOTE-ENN**, wodurch die Modelle seltene Fehlerzustände präziser vorhersagen konnten. Die Analyse der Feature-Importance reduzierte die Dimension des Datensatzes von 33 auf 24 Schlüsselmerkmale. Wichtige Merkmale wie **WEC: ava. Rotation, Rectifier cabinet temp.** und **Avg sys 1 inverter temp.** erwiesen sich als besonders relevant. Diese Reduktion verbesserte sowohl die Effizienz als auch die Interpretierbarkeit der Modelle. RF und XGB erwiesen sich als robuste und leistungsstarke Modelle, die nach Resampling und Hyperparameter-Tuning F1-Scores von 0.817 bzw. 0.800 für die Minderheitsklasse erreichten. Der **Decision Tree** eignet sich hingegen besser als interpretatives Werkzeug, da er einfacher zu verstehen, jedoch weniger leistungsstark ist. Trotz der vielversprechenden Ergebnisse bleiben Herausforderungen bestehen. Die Modelle wurden ausschließlich auf historischen **SCADA-Daten** getestet, sodass ihre Eignung für Echtzeitanwendungen unklar bleibt. Externe Variablen wie Wetterdaten oder Betriebsbedingungen, die die Vorhersagegenauigkeit verbessern könnten, wurden nicht berücksichtigt. Die Frage der Daten-Drift in Echtzeitszenarien könnte durch den Einsatz von Online-Learning oder adaptiven Modellen adressiert werden. Die hier entwickelten Methoden bieten Potenzial für andere industrielle Anwendungen, wie die Überwachung von Maschinen in der Fertigungsindustrie oder der Energieversorgung. Die Flexibilität und Skalierbarkeit der Modelle ermöglichen ihre Anpassung an verschiedene Anwendungsfälle, was Effizienz und Zuverlässigkeit in vielen Bereichen steigern könnte. Diese Arbeit zeigt, dass datengetriebene ML-Ansätze nicht nur technische Herausforderungen bewältigen, sondern auch praktische Vorteile wie die Verbesserung der Verfügbarkeit von Windkraftanlagen und die Optimierung von Ressourcen bieten. Mit kontinuierlicher Weiterentwicklung können diese Ansätze langfristig einen erheblichen Beitrag zur Effizienzsteigerung und Nachhaltigkeit leisten.

Danksagungen

In dieser Seminararbeit habe ich das LLM ChatGPT-4o von OpenAI verwendet, um Unterstützung bei der Bearbeitung verschiedener Aspekte der Arbeit zu erhalten. Konkret habe ich das Modell verwendet, um Programmierherausforderungen zu überwinden, z. B. indem ich exemplarischen Python-Code für mein Jupyter-Notebook bereitgestellt habe, den ich später an meine eigenen Anforderungen angepasst habe. Darüber hinaus habe ich ChatGPT verwendet, um den Schreibprozess zu unterstützen, indem es geholfen hat, den Text zu strukturieren, sprachliche Wiederholungen zu vermeiden und grammatischen Korrekturen vorzunehmen.

Ich habe alle vom Modell generierten Vorschläge sorgfältig überprüft, überarbeitet und an meine persönliche Ausdrucksweise angepasst. Der gesamte Inhalt der reflektierten Arbeit, einschließlich der endgültigen Formulierungen und Entscheidungen, wurde von mir unabhängig und meiner eigenen wissenschaftlichen Analyse und Interpretation der Ergebnisse erstellt.

Ich betone ausdrücklich, dass die Verantwortung für die inhaltliche Richtigkeit und wissenschaftliche Integrität der eingereichten Arbeit allein bei mir liegt. Alle generierten Inhalte wurden gründlich überprüft und überarbeitet, um akademischen Standards zu entsprechen. Diese Arbeit ist vollständig mein eigenes geistiges Eigentum und ich halte mich an die Grundsätze der wissenschaftlichen Authentizität und Unabhängigkeit. Auf Anfrage bin ich gerne bereit, sämtliche getroffenen Entscheidungen und Formulierungen zu begründen.

Literatur

- [1] Soontronchai, W. (2019). IIOT Data of Wind Turbine. Accessed: May 15, 2022. Retrieved from <https://www.kaggle.com/datasets/wasuratme96/iiot-data-of-wind-turbine>.
- [2] Fischer, K., Steffes, M., Pelka, K., Tegtmeier, B., & Dörenkämper, M. (2021). Humidity in Power Converters of Wind Turbines—Field Conditions and Their Relation with Failures. *Energies*, 14(7), 1919. <https://doi.org/10.3390/en14071919>.
- [3] Han, Y., & Joe, I. (2024). Enhancing Machine Learning Models Through PCA, SMOTE-ENN, and Stochastic Weighted Averaging. *Applied Sciences*, 14(21), 9772. <https://doi.org/10.3390/app14219772>.
- [4] Potential of Decision Trees in Non-Linear Machine Learning. Machine Learning Models. Retrieved from https://machinelearningmodels.org/the-potential-of-decision-trees-in-non-linear-machine-learning/?utm_source.
- [5] Singh, A. (2024). Random Forest Algorithm in Machine Learning. Applied AI Course Blog. Retrieved from https://www.appliedaicourse.com/blog/random-forest-algorithm-in-machine-learning/?utm_source.
- [6] Endemann, C. (2024). XGBoost: Tree-Based Gradient Boosting for Tabular Data. UW-Madison Data Science Nexus. Retrieved from https://uw-madison-datasience.github.io/ML-X-Nexus/Toolbox/Models/XGBoost.html?utm_source.
- [7] Demirović, E., & Stuckey, P. J. (2021). Optimal Decision Trees for Nonlinear Metrics. arXiv preprint, *arXiv:2009.06921v2*. <https://doi.org/10.48550/arXiv.2009.06921>.
- [8] Data4Energy. (2024). AI-Powered Predictive Maintenance: Transforming Turbine Management in 2024. Accessed: January 16, 2025. Retrieved from https://data4energy.com/ai-powered-predictive-maintenance-transforming-turbine-management-in-2024?utm_source.