HAW HAMBURG

BACHELORTHESIS
Marcel Hoop

Analyse von Validierungsstrategien für Machine Learning Modelle im Bereich der Zeitreihenprognose

Mit Konzeption und Entwicklung einer Umgebung zur Modellevaluation mithilfe generierter synthetischer Zeitreihen

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Marcel Hoop

Analyse von Validierungsstrategien für Machine Learning Modelle im Bereich der Zeitreihenprognose

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuende Prüferin: Prof. Dr. Marina Tropmann-Frick
Zweitgutachter: Prof. Dr. Axel Wagenitz

Eingereicht am: 31. August 2023

Marcel Hoop

Thema der Arbeit

Analyse von Validierungsstrategien für Machine Learning Modelle im Bereich der Zeitreihenprognose

Stichworte

Zeitreihen, Maschinelles Lernen, Validierung, Modellauswahl, Modellbewertung, Prognose, Synthetische Daten, Bedarfsprognose

Kurzzusammenfassung

Das Ziel dieser Arbeit ist die Analyse von Validierungsverfahren für Machine Learning Modelle, die zur Anwendung in der Zeitreihenprognose geeignet sind. Denn die zufällige Unterteilung von Daten durch allgemeine Validierungsverfahren, entfernt zum Teil die Reihenfolge der geordneten Zeitpunkte, die bei Zeitreihen essenziell sind.
Nach grundlegender Einführung in die Thematik wird die Vorgehensweise zur Bewertung eines Modells, der gängigen Fehlermaße und unterschiedlicher Validierungsverfahren aufgeführt. Anknüpfend wird beleuchtet, ob diese für Zeitreihen geeignet sind und speziell für Zeitreihen geeignete Validierungsverfahren aufgezeigt. Im Anschluss wird, als Grundlage empirischer Experimente, die Entwicklung eines Frameworks zur synthetischen Zeitreihengenerierung und automatisierten Ausführung der vorher gezeigten Validierungsverfahren erläutert. Daraufhin werden mehrere Versuche mithilfe des Frameworks durchgeführt, in denen untersucht wird wie sich eines der Validierungsverfahren mit unterschiedlichen Parameterwerten, zu einer saisonalen Periodenlänge innerhalb von synthetischen Zeitreihen anhand exemplarischer ML-Modelle verhält.
Die abschließende Auswertung der Ergebnisse zeigt auf, dass die eingesetzten Modelle unterschiedliche Resultate produzieren und die Ursache nicht eindeutig festgestellt werden kann. Dennoch wird ersichtlich, wie ausschlaggebend die Parameterwahl des Validierungsverfahrens in Bezug zur Modellbewertung ist und dass nicht nur Daten und Modell für eine Bewertung relevant sind.

Marcel Hoop

Title of Thesis

Analysis of validation strategies for machine learning models in time series forecasting

Keywords

Time Series, Machine Learning, Validation, Model Selection, Model Assessment, Forecasting, Synthetic Data, Demand Forecast

Abstract

The goal of this work is to analyze validation methods for machine learning models that are suitable for use in time series forecasting. This is because the random partitioning of data by general validation methods partially removes the order of ordered time points, which are essential in time series.

After a basic introduction to the topic, the procedure for the evaluation of a model, the common error measures and different validation methods are presented. Following this, it is examined whether these are suitable for time series and suitable validation methods especially for time series are discussed. Subsequently, as a basis for empirical experiments, the development of a framework for synthetic time series generation and automated execution of the previously shown validation methods is described. Thereafter, several experiments are conducted using the framework, in which it is investigated how one of the validation methods behaves with different parameter values, to a seasonal period length within synthetic time series using exemplary ML models.

The final evaluation of the results shows that the models used produce different results and that the cause cannot be clearly determined. Nevertheless, it becomes obvious how crucial the parameter choice of the validation procedure is in relation to the model evaluation and that not only data and model are relevant for an evaluation.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Abkürzungsverzeichnis

RQ	Research questions (Forschungsfragen)
ML	Machine Learning
KI	Künstliche Intelligenz
KDE	Kernel Density Estimation (Kerndichteschätzung)
ECD	Empirical Cumulative Distribution Function (empirische Verteilungsfunktion)

Glossar

Einflussvariablen Zusätzliche Informationsvariablen über äußere Einflussfaktoren, wie die Jahreszeit oder ein Börsencrash, die nicht direkt in den Daten enthalten sind.

Framework Entwicklungsrahmen oder Grundstruktur zur Nutzung von Funktionalitäten.

Genauigkeit Bezeichnung wie gut ein Modell Daten vorhersagt

Gewicht Veränderbare interne Parameter, meistens Koeffizienten, eines Modells zur Anpassung an Daten

Interface Schnittstelle zur Kommunikation zwischen zwei Komponenten

Konsekutiv Zeitlich folgend

Multivariat Mehrere Variablen betreffend

Prognose Vorhersage von zukünftigen Ereignissen

Residuen Differenz zwischen den tatsächlichen beobachteten Werten und den vorhergesagten Werten einer statistischen Analyse

Univariat	Nur eine Variable betreffend
Volatilität	Schwankung von Zeitreihen

1 Einleitung

1.1 Motivation

Zeit bestimmt unser Leben. Jeder würde gerne wissen, was als nächstes passiert. Nur wird heute kein Wahrsager, sondern künstliche Intelligenz befragt.

Andrew Ng sagte „Artificial Intelligence is the New Electricity“ („Künstliche Intelligenz ist die neue Elektrizität“) (Andrew Ng 25.01.2017) und vergleicht ihren Einfluss mit dem der Elektrizität vor 100 Jahren.

Bei beidem stellt sich die Frage, wie gut kann künstliche Intelligenz (KI) heutzutage die Zukunft vorhersagen und wie wird überhaupt gemessen, ob eine Vorhersage gut oder schlecht ist. Eingesetzte Methoden des maschinellen Lernens (ML) gibt es für viele Anwendungsmöglichkeiten wie Bilder verarbeiten und generieren, steuern von Robotern, stellen von Krankheitsdiagnosen, schreiben und verstehen von Texten, spielen von Spielen oder beweisen mathematischer Theoreme. (Russell und Norvig 2004; Görz 2003)
So gibt es auch für die Vorhersage von zeitlichen Daten ML-Methoden. Das Spezielle an solchen Daten ist die Berücksichtigung von Zeit und damit einer Reihenfolge. Denn was in der Zukunft passiert, hängt von der Vergangenheit ab und genauer dessen zeitlicher Ordnung.

Zudem sind zeitliche Daten bzw. Zeitreihen überall präsent. Ob es die Anzahl verkaufter Eisbecher am Wochenende, das Wetter, Autounfälle pro Jahr, sekündliche Börsenkurse oder Überwachungssensoren im Krankenhaus betrifft. In so gut wie allen Fällen ist die Vorausplanung unabdingbar, wobei auch kritische Entscheidungen getroffen werden müssen und eine helfende Vorhersage enorm wertvoll sein kann. (Box et al. 1994)

Aber auch wenn eine Vorhersage erstellt wird, stellt sich die Frage wie zuverlässig sie ist oder auch wie weit sie abweichen könnte. Grundsätzlich gibt es dafür Fehlermaße und

Validierungsverfahren, die ein konkretes Modell einer ML-Methode prüfen. Deren Ergebnisse können zum Vergleichen und Bewerten genutzt werden und dies ist einer der wichtigsten Schritte bei der Entwicklung. (Hastie et al. 2009)
Aber sind solche Verfahren auch für die spezielle Art von Zeitreihen geeignet?

1.2 Ziele & Forschungsfragen

Ein wichtiger Grund für den Einsatz von Vorhersagen bzw. der Zeitreihenprognose ist in vielen Bereichen die ausschlaggebende Planung. So hat sie beispielsweise in Form der Bedarfsprognose großes Potential in der Logistik. Denn umso genauer z.B. der zukünftige Bedarf eines Produktes geschätzt wird, desto besser können Vorbereitungen getroffen werden. Die Frage nach der zukünftigen Unsicherheit und wie man sie verringern kann, war schon immer entscheidend für Lieferketten. Die Aushandlung von Vertragsvolumen, präzise Planung von Lagerung, Bestandsmanagement, Verfügbarkeit und Sortimentsplanung hängt von der Frage ab, wie die Zukunft aussieht. Hinzu kommen schwankende Kundenanforderungen, ein harter Wettbewerb, komplexe globale Liefernetzwerke und die Anforderung von Effizienz und Kostenminimierung. (Falatouri et al. 2022; Vandeput und Makridakis 2021)

Des Weiteren machen das Wachstum datengetriebener Prozesse und die Verwendung von Machine Learning Methoden deutlich, wie essenziell die korrekte Einschätzung und Bewertung bzw. Validierung letzterer ist.

Deshalb stellt sich die Frage, welche allgemeinen Validierungsverfahren es zur Einschätzung und Bewertung von ML-Modellen gibt.
In Zusammenhang zu den speziellen Eigenschaften von Zeitreihen soll analysiert werden, ob solche Validierungsverfahren zur Anwendung mit Modellen zur Zeitreihenprognose geeignet sind und ob es entsprechende Alternativen gibt.

Weiterführend soll eines der Verfahren in Bezug zur saisonalen Periodenlänge in einer Zeitreihe, mithilfe von exemplarischen ML-Modellen, untersucht werden. Wie beispielsweise eine schwankende Produktnachfrage je Saison.

Um eigene empirische Untersuchungen durchführen zu können, stellt sich die Aufgabe zur Entwicklung eines Frameworks, das mehrere für Zeitreihen geeignete Validierungsverfahren

beinhaltet und mit dessen Anwendung Modelle bewerten werden können.
Damit die genutzten Daten für die Untersuchung beliebig angepasst werden können, soll das Framework die weitere Funktionalität bieten synthetische Zeitreihen anhand von Vorgaben zu generieren.

Daraus ergeben sich die folgenden Forschungsfragen:

Recherche

RQ1: Welche Validierungsverfahren für ML-Modelle gibt es im Allgemeinen für Supervised Learning?

RQ2: Sind Validierungsverfahren wie in RQ1 für die Verwendung mit Zeitreihen geeignet?

RQ3: Gibt es Validierungsverfahren, die speziell für Zeitreihen geeignet sind?

Praktische Umsetzung

RQ4: Wie kann ein möglichst allgemeines Spezifikationsschema für die Generierung unterschiedlicher univariater synthetischer Zeitreihen aussehen, mit dem die Struktur der Zeitreihen vorgegeben werden kann?

RQ5: Wie können beliebige ML-Modelle zur Zeitreihenprognose mit Validierungsverfahren aus RQ3 und univariaten Zeitreihen bewertet werden?

Empirische Untersuchung

RQ6: Wie verhält sich eines der Validierungsverfahren aus RQ3, mit unterschiedlichen Parameterwerten, zu einer saisonalen Periodenlänge innerhalb von synthetischen Zeitreihen anhand exemplarischer ML-Modelle?

1.3 Struktur der Arbeit

Zuerst wird in den Grundlagen das nötige Verständnis für die restliche Arbeit aufgebaut. Dazu gibt es eine Erläuterung zu Zeitreihen und was sie ausmacht. Ein kurzer Einblick in Machine Learning, gefolgt von einer Einführung in die Zeitreihenprognose und wie dabei vorgegangen

wird, vervollständigen die Einführung in die grundlegende Methodik. Danach gibt es eine Übersicht mit anschließendem Vergleich von üblichen Bewertungsmaßen bzw. Fehlermaßen, die sich zur Modelbewertung eignen. Abschließend wird aufgeführt wie Modelle mit Fehlermaßen bewertet werden können und was es für übliche Validierungsverfahren gibt.

Im Anschluss der Grundlagen wird anknüpfend an die Validierungsverfahren analysiert, ob diese auch für Zeitreihen geeignet sind und welche Alternativen es gibt.

Hierauf wird die praktische Umsetzung des Frameworks aufgeführt, indem Konzept, Anforderungen, Entwurf und Implementierung einzeln erläutert werden.

Abschließend wird die empirische Untersuchung eines für Zeitreihen geeigneten Validierungsverfahrens und der zusammenhängenden Auswirkungen mit Saisonalität beschrieben. Dies umfasst den Versuchsaufbau, dessen Ergebnisse, zwei weitere daraus folgende Versuche und eine zusammenfassende Auswertung.

1.4 Abgrenzung der Arbeit

Andere Arbeiten beschäftigen sich oft mit dem empirischen Vergleich von allgemeinen Validierungsverfahren und spezielleren Validierungsverfahren. Beispielweise wurden in (Schnaubelt 2019) dazu 8 übliche Validierungsverfahren miteinander verglichen. Der Fokus lag auf den Daten. Genauer lag er auf dem Unterschied von global stationären Daten und Daten, bei denen sich der datenerzeugende Prozess im Laufe der Zeit entwickelt und so die Stärke von Stationarität schrittweise erhöht wird. So war das Kernziel ein Leistungsvergleich von Validierungsmethoden in Bezug auf solche Daten. Genutzte Daten gliederten sich in selbst generierte synthetische Zeitreihen mithilfe autoregressiver Prozesse, in der die Stationarität künstlich verändert wurde und echten Finanzdaten. Die Ergebnisse zeigten, dass die Leistung der verschiedenen Validierungsverfahren von der Stationarität der Daten abhängt und die Verwendung von allgemeiner Validierung mit großem Risiko verbunden sein kann.
Auch in der Arbeit von (Cerqueira et al. 2017) belief sich der Fokus ganz ähnlich auf einen Leistungsvergleich verschiedenen Validierungsverfahren. Dabei fungierte die Validierung, die für unabhängig- und identisch verteilte Daten am häufigsten genutzt wird, als Ausgangspunkt zum Vergleich mit anderen angepassten Verfahren. Die verwendeten Daten bestanden aus 53

echten Zeitreihen aus verschiedenen Domänen und generierten synthetischen Daten, produziert mithilfe autoregressiver-, Moving Average- und saisonaler autoregressiver Prozesse.
Die Auswertung ist ähnlich zu (Schnaubelt 2019), dass in er Welt aufgezeichnete Daten oft nicht stationär sind und für solche Einsatzszenarios andere Validierungsverfahren besser geeignet sind, als das nicht angepasste Verfahren.

Gegenüber solchen Arbeiten ist hier nicht das Ziel empirisch zu vergleichen, wie sich normale und für Zeitreihen geeignete Validierungsverfahren unterscheiden und welche davon bessere Ergebnisse liefern. Generell sollen die unterschiedlichen Verfahren erklärt werden und nur anhand der Vor- und Nachteile miteinander verglichen werden. Die empirische Untersuchung richtet sich auf das Verhalten eines Validierungsverfahrens mit verschiedenen Parametern und den Auswirkungen bei vorhandener Saisonalität in den Daten. Aus den Arbeiten überschneidend ist die Nutzung synthetischer Daten, wobei die Möglichkeiten der Generierung hier etwas erweitert werden.

Da der Fokus nicht auf der Entwicklung und Anwendung von ML-Modellen liegt, wird zum Verständnis zwar eine Beschreibung des Vorgehens gegeben und relevante Begrifflichkeiten erklärt, aber der Detailgrad dazu in Grenzen gehalten. Erläuterungen zu ML-Modellen beziehen sich in erster Linie auf Modelle zur Zeitreihenprognose.

Außerdem wird in Bezug zu Daten bzw. Zeitreihen der Einfachheit halber von multivariaten Daten abgesehen. Sofern nicht anders angegeben, beziehen sich Formulierungen in der Arbeit auf univariate Daten. Für die Definition der beiden Arten siehe Abschnitt 2.1 im Kapitel Grundlagen.

Die Beschränkung auf univariate Daten betrifft auch das Framework. Die synthetischen Zeitreihen bestehen aus univariate Daten und Bewertungen erwarten Daten in der Form der Generierung, so z.B. ohne fehlende Einträge etc.

2 Grundlagen

2.1 Was sind Zeitreihen?

Eine Zeitreihe ist eine Sequenz aus konsekutiven Beobachtungen. Einfach ausgedrückt ist es eine Folge von Werten. Ausschlaggebend ist dabei der Zeitpunkt, an dem diese beobachtet wurden, wie auch, in der Regel, die Abhängigkeit benachbarter Werte. (Box et al. 1994) Damit ist nicht nur ihre zeitliche Ordnung relevant, sondern auch Vorkommnisse wie fehlende Einträge oder Verzögerungen.

Zeitreihen entstehen meistens durch regelmäßiges Sammeln und Aufzeichnen, wodurch die Daten zu einem bestimmten Zeitpunkt beginnen und zu einem anderen enden. (Hamilton 1994) Es gibt viele verschiedene Arten von Zeitreihen. Dazu gehören zyklische Aufzeichnungen wie jährliche, monatliche, wöchentliche und tägliche Wiederholungen, wie zum Beispiel die Verkaufszahlen eines Produkts, aber auch Echtzeitdaten wie der Börsenkurs, das Wetter, der Energieverbrauch und die Besuchszahlen einer Webseite.

Dabei ist zu beachten, dass eine Zeitreihe mehr oder weniger komplex sein kann. Wenn mehr als zwei Variablen einen einzelnen Zeitpunkt definieren, nennt man das multivariate Daten. Diese liegen zum Beispiel dann vor, wenn äußere Einflüsse mit in den Daten vorkommen. Im Fall von Produktverkäufen, wären das die zusätzlichen Informationen, ob es ein Feiertag war, welche Jahreszeit vorlag etc. neben der eigentlichen Information über die Anzahl von Verkäufen. Ein anderes Beispiel wäre das Wetter, was für einen bestimmten Ort nicht nur vom Tag abhängt, sondern viele andere Faktoren, wie die Höhenlage, Luftfeuchtigkeit und Windrichtung, einbezogen werden könnten. Entsprechend bezeichnet bivariat genau zwei und univariat genau eine definierende Variable. So bestehen univariate Daten nur aus einem Wert für jeden Zeitpunkt in einer Zeitreihe. (Rob J Hyndman und George Athanasopoulos 2023; Armstrong und Grohman 1972; Crawford 1989)

2.2 Strukturelle Elemente von Zeitreihen

Trend

Ein Trend oder auch Trendverlauf ist die durchschnittliche Veränderung des Mittelwerts innerhalb einer Zeitreihe zwischen zwei aufeinanderfolgenden Zeiträumen oder anders ausgedrückt ein linearer Anstieg über Zeit, wie in Abbildung 1 dargestellt. (Vandeput und Makridakis 2021; Rob J Hyndman und George Athanasopoulos 2023)

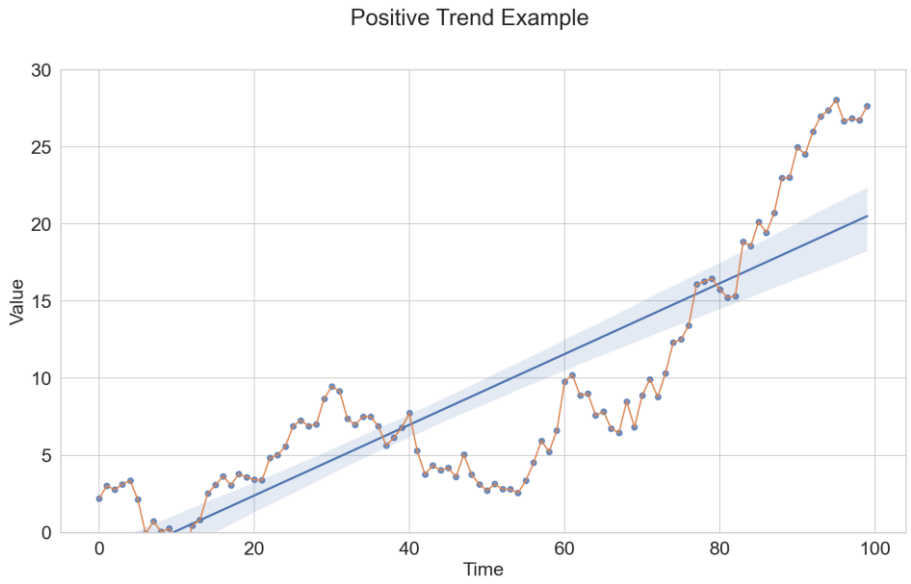


Abbildung 1: Positiver Trend Beispiel

Saisonaler Verlauf

Wenn in festen Zeitintervallen ein wiederkehrendes Muster bzw. Verlauf in einer Zeitreihe auftritt, wird dies meistens als saisonaler Verlauf bezeichnet. Solche Muster können beispielsweise durch Jahreszeiten, wechselnde Trends, verschiedene Wochentage, Schulferien oder andere regelmäßige Ereignisse hervorgerufen werden. Ein simples Beispiel sind Temperaturen, die über das Jahr ansteigen und wieder abfallen. Ein beispielhafter Verlauf ist in Abbildung 2 zu sehen. (Rob J Hyndman und George Athanasopoulos 2023)

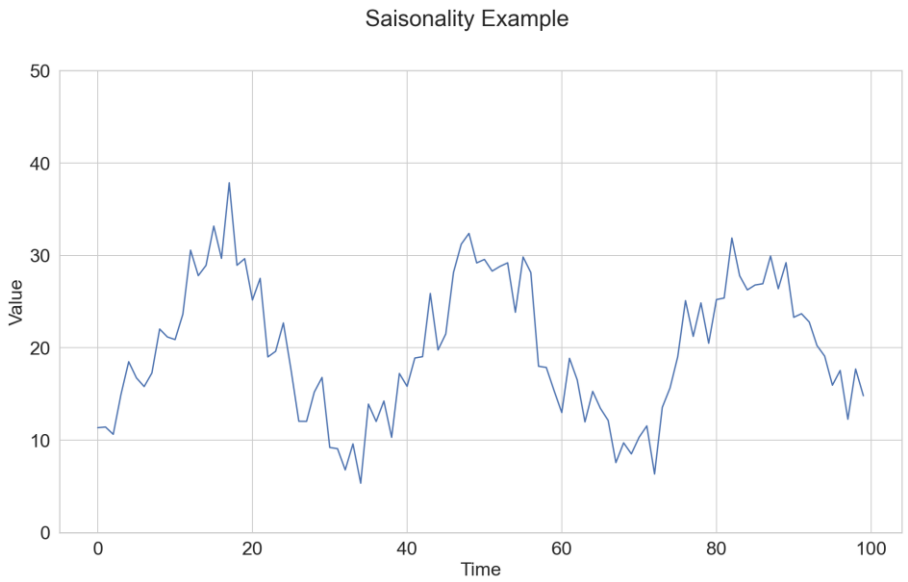


Abbildung 2: Saisonaler Verlauf Beispiel

Rauschen / Noise

In der Statistik ist Rauschen eine unerklärliche Variation in den Daten, die oft auf die Zufälligkeit verschiedener Prozesse zurückzuführen ist. (Vandeput und Makridakis 2021) Rauschen beeinträchtigt damit die Klarheit der Daten und erschwert teilweise den Informationsgewinn. Siehe Abbildung 3.

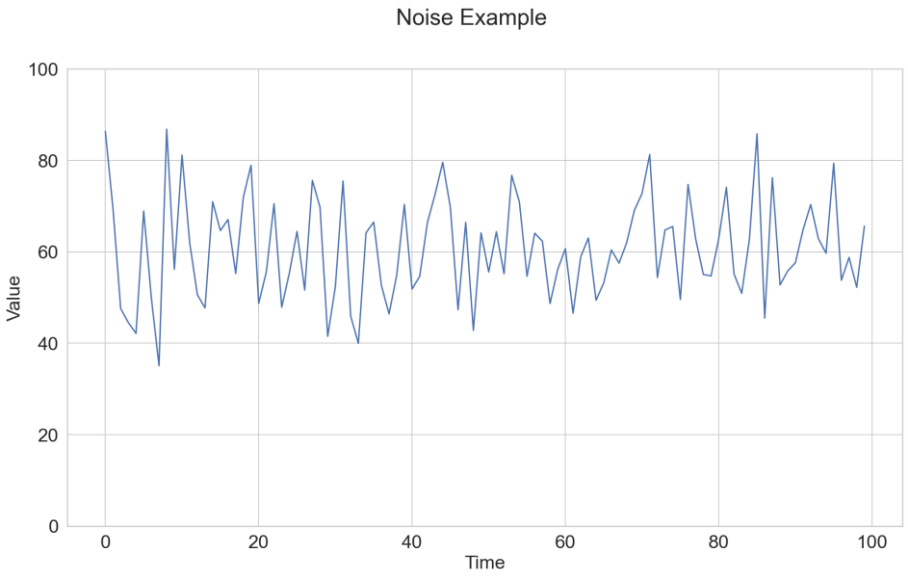


Abbildung 3: Rauschen Beispiel

Ausreißer / Outliers

In einer Zeitreihe identifizieren sich Ausreißer als außergewöhnlich hohe oder niedrige Werte, die nicht dem Verlauf entsprechen, wie in Abbildung 4 gezeigt.. Vereinfacht ist ein Ausreißer ein Wert, der nicht erwartet wird. (Vandeput und Makridakis 2021; Vandeput und Makridakis 2021, 121)

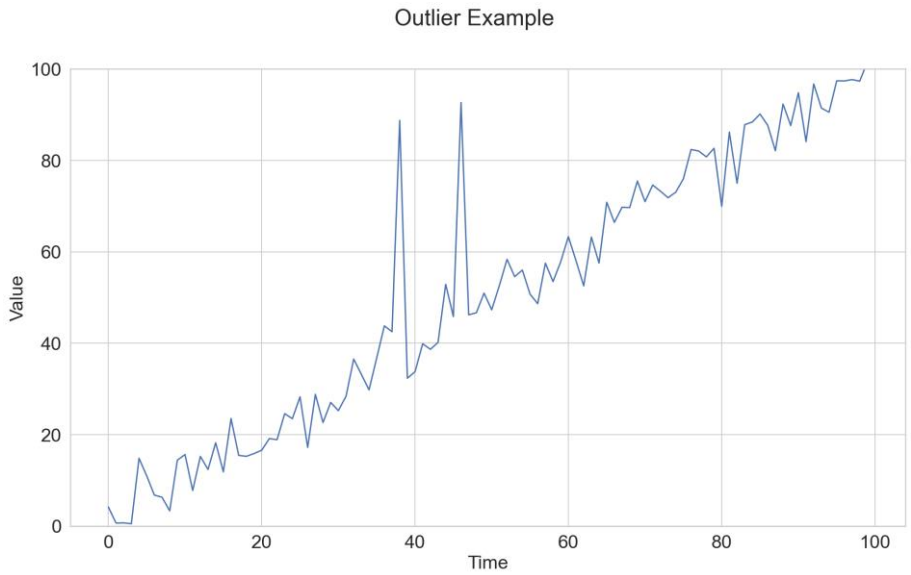


Abbildung 4: Ausreißer Beispiel

2.3 Machine Learning

In diesem Abschnitt wird kurz Machine Learning erklärt und was davon für die Zeitreihenprognose Verwendung findet. Außerdem werden zwei Hauptproblematiken von Machine Learning Under- und Overfitting erläutert.

2.3.1 Kurzübersicht und Begrifflichkeiten

Machine Learning (ML) ist ein Teilbereich von künstlicher Intelligenz (KI), der sich damit befasst Computer dazu zu nutzen automatisch Zusammenhänge aus Erfahrungen, meist in Form von Daten, zu erkennen und auf neue Situationen anzuwenden. (Neupert et al. 2020)

Im Bereich des ML gibt es eine Vielzahl von unterschiedlichen Techniken/Methoden. Eine konkrete Methode wird dabei auch oft als ein ML-Modell bezeichnet. Die meisten solcher Modelle haben interne Parameter, die Gewichte genannt werden. Beim Vorgang der Verarbeitung von Daten passt ein Modell diese internen Gewichte wiederholt an, um sich den Daten immer besser anzupassen. Dieser Prozess wird auch als Lernen oder Training bezeichnet.
Gegenüber internen gibt es auch Parameter, die von außen angegeben werden können, um die Verarbeitung von Daten zu justieren. Dabei wird von Hyperparametern gesprochen, die von den Benutzern üblicherweise gut gewählt werden müssen, um das bestmögliche Ergebnis mit einem Modell zu erzielen. Welches das beste Modell ist wird oft durch die sogenannte Genauigkeit bestimmt, in der je nach Problemstellung getestet wird, wie oft oder wie sehr das Modell richtig liegt. (Neupert et al. 2020)

2.3.2 Supervised Learning

Für ML-Methoden gibt es hauptsächlich drei Kategorien, für die die Problemstellung unterschieden werden kann. Supervised Learning (überwachtes Lernen), Unsupervised Learning (unüberwachtes Lernen) und Reinforcement Learning (Verstärkendes Lernen). Im Bereich der Zeitreihenprognose geht es hauptsächlich um Supervised Learning. Dabei wird, ausgehend von bekannten Ein- und Ausgabedaten, eine Generalisierung angestrebt, um für unbekannte bzw. neue Eingabedaten anhand von Ähnlichkeiten die passenden Ausgabedaten korrekt zu produzieren. (Vandeput und Makridakis 2021; Ertel 2018; Hastie et al. 2009)

Eine weitere Unterteilung von Supervised Learning ergibt die zwei Hauptkategorien Klassifikation und Regression.

In der Klassifikation geht es um die Zuordnung von Eingabedaten zu vorgegebenen diskreten Werten, die verschiedene Klassen oder Kategorien abbilden. Zum Beispiel im Bereich der Bilderkennung, wenn Bilder von Hunden und Katzen der jeweiligen Klasse Hund oder Katze zugeordnet werden sollen. Es wird dann ein Bild nach dem anderen mit bereits bekannter Zuordnung als Eingabedaten verwendet und die korrekte Klasse als gewünschte Ausgabe angegeben. So soll ein Modell Gemeinsamkeiten von allen Hundebildern und allen Katzenbildern erkennen und Unterschiede zwischen Bildern von Hunden und Katzen lernen. Anschließend soll das

Modell neue Bilder von Hunden und Katzen, die es noch nie gesehen hat, denselben Klassen zuordnen können. (Hastie et al. 2009; Ertel 2018)

Bei der Regression geht es um die Vorhersage von kontinuierlichen Werten, basierend auf Eingabedaten. Sie zielt darauf ab, bei bekannten Daten, eine Beziehung zwischen den Eingabevariablen und Ausgabevariablen mathematisch abzubilden, um so neue Daten schätzen zu können. Zum Beispiel soll ein Regressionsmodell so den Preis einer Wohnung oder eines Hauses anhand von Größe, Ort, Zimmeranzahl, Baujahr, Lage und Poolgröße schätzen. Dafür bekommt es Daten von Häusern, bei denen diese Informationen vorliegen und der Preis bekannt ist, um anschließend unbekannte Preise anhand der Informationen anderer Häuser zu schätzen. (Hastie et al. 2009; Ertel 2018)

Die Vorhersage von zukünftigen Werten anhand vergangener Werte in Zusammenhang mit kontinuierlichen Werten, wie bei einer Zeitreihe, ist somit auch ein Regressionsproblem, wodurch nur solche ML-Techniken für diese Problemstellung in Frage kommen und später behandelt werden.

2.3.3 Underfitting

Begriffserklärung

Underfitting tritt auf, wenn ein Modell nicht in der Lage ist, die vorhandenen Daten angemessen zu erfassen, wodurch es unzureichend angepasst ist und die Realität nicht genau genug abbilden kann. (Vandeput und Makridakis 2021)

Man könnte sagen, dass ein underfitted Modell den Trainingsdatensatz nicht richtig versteht. Da das Modell im Trainingsdatensatz nicht richtig funktioniert, wird es auch im Testdatensatz nicht gut abschneiden. Im Fall einer Nachfrageprognose wird ein Modell, das bei der historischen Nachfrage keine gute Genauigkeit erreicht, auch bei der zukünftigen Nachfrage keine gute Leistung erzielen. (Vandeput und Makridakis 2021)

Ursachen

Ein Grund für Underfitting kann eine zu geringe innere Modellkomplexität sein. Mit anderen Worten kann das Modell die entsprechenden Muster aus den Daten nicht verstehen und lernen,

wenn das Model zu wenig Möglichkeiten der eigenen Anpassung hat. (Vandeput und Makridakis 2021)

Ein anderer Grund kann das Fehlen von zusätzlichen Einflussvariablen sein, die Informationen über äußere einflussnehmende Ereignisse bereitstellen. Bei mehrfacher Veränderung des Verlaufs in einer Zeitreihe ohne identifizierbaren Grund innerhalb der Daten, kann ein Modell diese Zusammenhänge nicht gut adaptieren. Sogar bei wiederkehrenden Schwankungen durch bestimmte Ereignisse, wie ein starker Anstieg am Wochenende, reagiert ein Modell zwar darauf, aber könnte trotzdem eine bessere Genauigkeit erreichen, wenn diese Ereignisse als zusätzliche Information in den Daten angegeben sind. Zum Beispiel könnte der starke Anstieg am Wochenende dazu führen, dass das Modell zum Anfang der Woche immer zu hoch schätzt, weil es den Zeitraum des Wochenendes nicht identifiziert bekommt. (Vandeput und Makridakis 2021)

Maßnahmen

Lösungen für Underfitting kann demnach das Verwenden eines komplexeren Modells oder die Steigerung der Komplexität durch Veränderung der Hyperparameter sein. Bei einem neuronalen Netz zum Beispiel durch das Erhöhen der Anzahl Schichten und Knoten.
Hinzukommend kann der Einsatz von zusätzlichen Einflussvariablen die Genauigkeit verbessern. Falls Modelle solche Einflussvariablen nicht direkt verarbeiten können, besteht die Möglichkeit für jeden, aus den Einflussvariablen hervorgehenden, Ereigniszeitraum ein eigenes Modell zu verwenden. Für das Beispiel oben würde es dann ein Modell für normale Wochentage und eines für Wochenendtage geben. (Vandeput und Makridakis 2021)

2.3.4 Overfitting

Begriffserklärung

Gegenüber dem Underfitting gibt es auch das Overfitting, was bedeutet, dass ein Modell die Trainingsdaten zu genau übernimmt. Das Modell lernt dann nicht nur die vorhandenen Muster auswendig, anstatt sie zu verallgemeinern, sondern übernimmt auch die zufälligen Schwankungen des Rauschens. Anders gesagt werden Muster gelernt, die zufälligerweise in der Trainingsmenge funktionieren. Aber da diese Muster höchstwahrscheinlich in zukünftigen Daten nicht

wieder vorkommen, werden dann falsche Vorhersagen getroffen. Problematisch ist dabei die trügerische (sehr) gute Genauigkeit des Modells mit den historischen Trainingsdaten, aber schlechterer Leistung für unbekannte neue Daten. Eine sehr gute Genauigkeit im Training darf also nicht einfach akzeptiert und für wahr gehalten werden. Da Testdaten nicht im Training vorkommen sollten, ist Overfitting an dem hohen Unterschied der Genauigkeit zwischen Training und Test zu erkennen. Weiteres zur Datentrennung in 2.6.3 als In-Sample vs. Out-of-Sample. (Vandeput und Makridakis 2021; Hastie et al. 2009)

Reale Beispiele

Overfitting kann sich auch anders äußern, weshalb es wichtig ist Zufall, Korrelation und Kausalität voneinander zu unterscheiden. In (Silver 2020) gibt es ein perfektes Beispiel für Overfitting bei dem die Wirtschaftsleistung des restlichen Jahres vom Gewinnerteam des Super Bowls ausging. Von 1967 bis 1997 gewann der Aktienmarkt für den Rest des Jahres durchschnittlich 14 Prozent, wenn ein Team aus der National Football League (NFL) gewann. Er fiel jedoch um fast 10%, wenn stattdessen ein Team aus der American Football League (AFL) siegte. Bis 1997 hat dieser Indikator die Richtung des Aktienmarktes in achtundzwanzig von einunddreißig Jahren korrekt vorhergesagt. Sogar ein standardmäßiger statistischer Signifikanztest hätte ergeben, dass die Wahrscheinlichkeit von reinem Zufall dieser korrekten Vorhersage nur bei etwa 1:4.700.000 liegt.

Ein ähnlicher Effekt tritt häufig bei politischen Wahlen auf. Es heißt dann zum Beispiel, dass jemand mehr als 100 Variablen getestet hat, um vorherzusagen, wer die Wahl gewinnen würde und dabei ein Modell gefunden wurde, was die Ergebnisse der letzten zehn Wahlen perfekt widerspiegelt. Wenn man sich allerdings 100 Variablen ansieht, wird man mit Sicherheit feststellen, dass mindestens eine von ihnen zu jedem Ergebnis passt, das man vorhersagen möchte. (Vandeput und Makridakis 2021)

Auf der Webseite (Tyler Vigen 2023) sind weitere anschauliche Visualisierungen von solchen zufälligen Zusammenhängen. In Abbildung 5 kann zum Beispiel eine Korrelation zwischen Filmen mit Nicolas Cage und Personen, die im Pool ertrunken sind, erkannt werden.

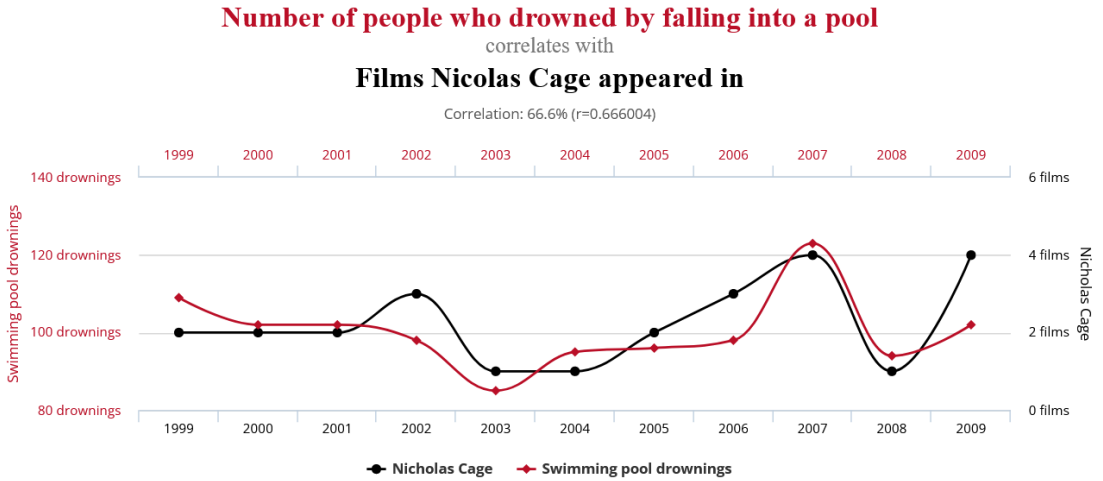


Abbildung 5: Korrelation von im Pool ertrunkenen Personen und Filmen von Nicolas Cage (Tyler Vigen 2023)

Maßnahmen

Wie beim Underfitting können zusätzliche Einflussvariablen auch einen Effekt auf Overfitting haben. Denn desto mehr ein Modell davon hat, umso mehr Möglichkeiten hat es Muster zu erkennen, egal ob sie sinnvoll sind oder nicht. Somit kann die Reduzierung solcher Variablen oder die Verwendung eines simpleren Modells eine Verbesserung bewirken. (Vandeput und Makridakis 2021)

Eine einfache Möglichkeit Overfitting entgegenzuwirken, ist die generelle Herangehensweise so viele Daten wie möglich zu verwenden, damit ein Modell gut abstrahieren kann. Der Einsatz von mehr Daten kann also bei dem Auftreten von Overfitting eine Veränderung bewirken. (Vandeput und Makridakis 2021)

Weniger eine Maßnahme, sondern stattdessen von Anfang an vermieden werden, sollte die Verwendung der Testdaten zum Trainieren sein, wie später in 2.6.3 als In-Sample vs. Out-of-Sample erklärt. Denn einem Modell, das nur mit einem Trainingsdatensatz gut abschneidet, kann man nicht trauen, weil nicht davon auszugehen ist, dass sich die damit erzielten

Ergebnisse auf ungesehene Daten übertragen lassen. Mit den Testdaten sollte also nur die finale Genauigkeit eines Modells ermittelt werden. (Vandeput und Makridakis 2021)

2.4 Einsatz synthetischer Daten

Wenn ein Modell für die Prognose entwickelt wird, sind dazu Daten nötig. Am besten solche, die den vorherzusagenden Daten so ähnlich wie möglich sind. Bei der Verwendung andersartiger Daten zur Entwicklung eines Modells, könnte dessen Performance für diese Daten sehr gut werden und anschließend bei der Prognose von den gewollten Daten sich erheblich unterscheiden. (Emam et al. 2020; Bahrpeyma et al. 2021; Zhang et al. 2018)

Neben der Entwicklung eines Modells für einen bestimmten Anwendungsfall und dessen Daten, gibt es andere Verwendungszwecke für Daten. Zum Beispiel das Entwickeln oder Testen von Algorithmen und generell das Experimentieren mit ML-Verfahren benötigt Beispieldaten, mit denen sie durchgeführt werden können. Je nachdem wie speziell das Vorhaben ist und was untersucht werden soll, können die Anforderungen an die Daten unterschiedlich sein.
Es kann auch eine enorme Einschränkung sein, wenn benötigte Beispieldaten nicht öffentlich zugänglich sind. Zum Beispiel wenn es um personenbezogene Daten geht, die durch Datenschutzgründe nicht von jedem genutzt werden dürfen oder die Aufzeichnungen von Verkaufszahlen aus betrieblichen Gründen von Firmen nicht veröffentlich werden. (Emam et al. 2020; Bahrpeyma et al. 2021; Zhang et al. 2018)

Allerdings kann es auch fast perfekt passende öffentliche Datensätze geben. So kann ein Datensatz viele Eigenschaften aufweisen, die oft erwünscht sind, wie zum Beispiel die Sauberkeit der Daten, in Form wenig vorhandener Lücken, eine große Menge von Daten, wobei mehr Daten nie verkehrt sind und die gegebenen Merkmale, aus denen die Daten bestehen.
Durch das Erfüllen genügend solcher Eigenschaften, kann derselbe Datensatz für sehr viele unterschiedliche Vorhaben genutzt werden und wird auch immer wieder genutzt. Um aber Untersuchungen mit mehreren und vielleicht auch verschiedenen Datensätzen durchzuführen, reicht ein einzelner guter Datensatz nicht aus und falls nur bestimmte Unterscheidungen in den Daten erforderlich sind und die restliche Struktur gleich sein soll, wird es immer schwieriger passende Beispieldaten zu finden. (Emam et al. 2020; Bahrpeyma et al. 2021; Zhang et al. 2018)

Für manche solcher Fälle werden dann stattdessen synthetische Daten verwendet, da sie einiges gegenüber echten Daten vereinfachen. Sie können in beliebiger Größe und Anzahl erstellt und angepasst werden. Es gibt nur Störungen, wie Rauschen, Ausreißer oder fehlende Werte, wenn dies gewollt ist. So kann ein Modell einmal mit und einmal ohne diese getestet werden, um herauszufinden wie gut es damit zurechtkommt. Zudem können viele unterschiedliche Variationen von Datensätzen generiert und ausprobiert werden, um Stärken und Schwächen von einem Modell kenntlich zu machen. Auch Overfitting anhand eines einzelnen statischen Datensatzes kann so vermieden werden, wenn pro Untersuchung neue etwas abweichende Daten genutzt werden. (Emam et al. 2020; Bahrpeyma et al. 2021; Zhang et al. 2018)

Allerdings ist zu beachten, dass selbst erzeugte Daten ungenügend aus echten Vorkommnissen entstandene Daten abbilden können, wodurch sie zum Entwickeln von Modellen für einen konkreten Einsatz nur bedingt geeignet sind. Auch echte Zusammenhänge, wie der Einfluss von äußeren Indikatoren, sind nur mit entsprechendem Aufwand nachzustellen, aber für manche Modellarten unabdingbar. Wie in (Armstrong und Grohman 1972) als naiv und kausal beschriebene Methoden, gibt es auch Modelle, die sich nur anhand der abhängigen Variablen orientieren bzw. nur an den Variablen der äußeren Einflüsse. Als Beispiel wäre eine abhängige Variable die Anzahl von verkauftem Eis an einem Tag und die Angabe der Jahreszeit eine Variable über äußere Einflüsse. Solche äußeren Einflussfaktoren werden im Verlauf dieser Arbeit auch als Einflussvariablen bezeichnet. Hinzu kommen globale Ereignisse, die einen Verlauf bis ins extreme beeinflussen können, wie z.B. ein Börsencrash oder Vergleichbares. Um auch solche Vorkommnisse in synthetische Daten einzupflegen, müsste an jede Eventualität im Voraus gedacht werden und was für Auswirkungen daraus in den Daten entstehen. (Emam et al. 2020; Bahrpeyma et al. 2021; Zhang et al. 2018)

Ob synthetische Daten für ein Vorhaben geeignet sind, hängt schlussendlich von dessen Anforderungen ab. Normalerweise werden Daten zuerst analysiert, um die Struktur, bestimmte Eigenschaften, die Verteilung und für die Modellwahl bzw. das Modell wichtige Informationen herauszufinden. Bei synthetischen Daten ist dies alles bereits bekannt und wird viel mehr vorgegeben und nach Bedarf manipuliert.
In dem Fall dieser Arbeit sollen Variationen synthetischer Zeitreihen anhand eines Frameworks generiert werden können. Diese sollen für Modelle als zu lernende Daten dienen, woraufhin

Vergleiche der Prognosen möglich sind. Damit dabei die Datenbasis eines Durchgangs immer sehr ähnlich ist, sind synthetische Daten geeignet. So gibt es z.B. keine Beeinflussung eines Modells durch zufällige Vorkommnisse in echten Daten, für die ein Modell vielleicht empfindlicher ist als das andere. Eine ähnliche Verwendung von selbst generierten Zeitreihen erfolgt z.B. auch in (Schnaubelt 2019).

2.5 Zeitreihenanalyse und Prognose

2.5.1 Begriffserklärung

Ein wesentliches Merkmal einer Zeitreihe ist, dass benachbarte Beobachtungen in der Regel voneinander abhängig sind. Die Art dieser Abhängigkeit zwischen den Beobachtungen einer Zeitreihe ist von großem praktischen Interesse. Die Zeitreihenanalyse befasst sich mit Techniken zur Analyse dieser Abhängigkeit. Dies umfasst die Entwicklung von stochastischen und dynamischen Modellen für Zeitreihendaten und die Anwendung solcher Modelle in wichtigen Anwendungsbereichen. (Box et al. 1994)

Das Ziel der Zeitreihenprognose ist der Aufbau von Verständnis über den Verlauf und Zusammenhang der historischen Daten, so dass das weitere Verhalten der Zeitreihe auf dieser Grundlage vorhergesagt und zukünftige Werte geschätzt werden können. (Box et al. 1994; Rob J Hyndman und George Athanasopoulos 2023)

2.5.2 Einsatzgebiet Bedarfsprognose

Auch wenn mithilfe einer Zeitreihenprognose viele verschiedene Anwendungsmöglichkeit existieren, ist der Grundgedanke für die praktische Umsetzung im Rahmen dieser Arbeit die Bedarfsprognose. Dabei handelt es sich zum Beispiel um die Vorhersage wie viele Exemplare eines Produkts in einer bestimmten Zeit, z.B. innerhalb eines Tages, verkauft werden. Solche Zahlen im Voraus zu wissen, ist vor allem im Bereich der Logistik relevant. So müssen komplexe globale Wertschöpfungsketten effizient genutzt und Lagerbestände vorher sorgfältig geplant werden, damit weder zu viel Platz belegt und damit verschwendet wird noch zu wenig Produkte zur Verfügung stehen, so dass eine steigende Kundennachfrage nicht bedient werden kann. Eine präzise Bedarfsprognose ermöglicht fundierte Entscheidungen zur Bestandsführung

und Sortimentsplanung. Es gibt dabei viele unterschiedliche Einflussfaktoren - nicht nur vergangene Verkaufszahlen. Zusatzinformationen, die den Markt bzw. Verkauf beeinflussen, können ausschlaggebend sein. Ein starker Wettbewerb in der Handelsbranche führt beispielsweise zu aggressivem Marketing, woraus eine spontan stark erhöhte Nachfrage entstehen kann. So werden die Prognose immer anspruchsvoller und komplexer, je nach Produktvielfalt und geografischer Größenordnung. In (Hewage und Perera 2021) werden beispielsweise verschiedene Arten von Promotions (Webeaktionen) und deren Auswirkungen auf die Zeitreihenprognose in Zusammenhang verschiedener Modelle untersucht und wie damit umgegangen werden kann. (Falatouri et al. 2022; Hewage und Perera 2021)

Somit ist die Bedarfsprognose ein gutes und anschauliches Beispiel für den Anwendungsfall einer Zeitreihenprognose.

2.5.3 M-Competitions

Um aktuelle Fortschritte in der Entwicklung von Methoden zur Zeitreihenprognose zu erfassen, werden die von Spyros Makridakis ins Leben gerufenen und nach ihm benannten M-Competitions seit 1982 regelmäßig ausgetragen.

Da der aktuelle Stand der Technik oft nicht klar zu erkennen ist, wurden die M-Competitions als ein Anhaltspunkt dessen verwendet. So wird sich in verschiedenen Kapiteln auf manche M-Competitions berufen, weshalb hier erläutert wird worum es sich dabei handelt.

Die bekannten internationalen M-Competitions sollen die Entwicklung und Forschung von Prognosetechniken fördern und vergleichen. Dazu wird in jeder Competition eine konkrete Aufgabenstellung vorgegeben, für die die Teilnehmer Modelle zur Lösung entwickeln. Die abschließende Auswertung und der Vergleich zwischen den unterschiedlichen Techniken ist neben der Benennung des Siegers oft die Grundlage für Forschungspublikationen wie z.B. (Makridakis et al. 2022) für die aktuelle M5-Competition. Sie zeigte, dass ML-Methoden in den Mainstream der Prognoseanwendungen vorgedrungen sind, wobei auch in der M4-Competition und anderen Wettbewerben erkannt wurde, dass sie immer genauer werden als statistische Modelle. Der Anwendungsfall betraf dabei Verkaufszahlen von Produkten, also einer Bedarfsprognose.

Ein wichtiger Aspekt der Competitions ist der Vergleich und die Untersuchung von Vorteilen und Nachteilen der eingesetzten Techniken. Dies führt auch dazu, dass sich gut bewährte Methoden etablieren und zum Standard werden.
Für weitere Informationen zu M-Competitions kann deren Internetseite (MOFC 2023) dienen.

2.5.4 Grundlegende Schritte und Begrifflichkeiten

Um einen Überblick über die generelle Vorgehensweise für die Entwicklung einer Zeitreihenprognose zu geben, werden im Folgenden die dabei üblichen Schritte erläutert. Da keine Allgemeinlösung zum Vorgehen für alle Eventualitäten existiert, soll dies als Übersicht dienen, bei der einzelne Schritte von Fall zu Fall abweichen können.

Schritt 1: Problemdefinition

Der erste und oft schwierigste Schritt ist die Definition des Problems und damit einhergehend die Beschreibung des Ziels, was mit der Zeitreihenprognose erreicht werden soll. Dies betrifft viele Fragen, wie z.B.:

- Wer benötigt die Vorhersagen?
- Wofür sollen sie eingesetzt werden?
- Müssen bestimmte Anforderungen erfüllt sein?
- Worauf liegt der Fokus bezüglich Qualität und Sicherheit der Prognose?
- Grundsätzlich was soll vorhergesagt werden?

Dazu kommt die Absprache und Organisation mit allen beteiligten Bereichen, die je nach Projektgröße und Infrastruktur unterteilt sein können. Zum Beispiel muss zu Anfang klar sein wie Daten gesammelt, gespeichert und abgerufen werden können oder die Vorhersagen zukünftig nutzbar sein sollen. (Rob J Hyndman und George Athanasopoulos 2023)

Schritt 2: Datenbeschaffung

Die eigentliche Sammlung oder Beschaffung von Daten legt den Grundstein für zukünftige Prognosen. Abhängig vom Anwendungszweck können öffentliche Daten genau die Eigenschaften aufweisen, die benötigt werden. Falls es eher um interne Vorgänge geht, können

Aufzeichnungen dazu existieren, die die Anforderungen erfüllen. Ansonsten kann es sein, dass die Sammlung von Daten erst geplant und umgesetzt werden muss, um mit gewisser Zeit genügend Daten zu produzieren. Dabei ist es wichtig nur mit genügend Daten die weiteren Schritte fortzuführen, da mit zu wenig Daten keine gute Prognose entstehen kann. (Rob J Hyndman und George Athanasopoulos 2023)

Schritt 3: Datenanalyse und Bereinigung

Sobald Daten zur Verfügung stehen, werden sie auf ihre Eigenschaften untersucht. Für viele ist dazu immer als erste Aktion eine erste Visualisierung der Daten wichtig. Dies hilft, um erste Erkenntnisse über den Verlauf zu gewinnen, merkwürdig erscheinende Vorkommnisse zu identifizieren und eine generelle Übersicht zur Struktur der Daten zu erhalten. Auch wiederkehrende Muster, ein vorkommender Trend, regelmäßige Saisonalität und enthaltene Ausreißer können so von Menschen schnell erkannt und eingeschätzt werden. Des Weiteren können statistische Analysen durchgeführt werden, um weitere Informationen, wie den Wertebereich und die Verteilung, zu erfahren. So können auch einzelne Abschnitte für eine genauere Analyse vorgemerkt werden. Dadurch aufkommende Fragen, wie die Ursache für Ausreißer oder allgemein auftretende Muster, können daraufhin mit entsprechendem Wissen über den Datenhintergrund beantwortet werden. So können auch nicht erwartete Daten identifiziert und gegebenenfalls bereinigt werden. Allgemein findet in diesem Schritt der erste Wissensaufbau über die zu prognostizierenden Daten statt, der die Grundlage für weitere Schritte ist.
Zu weiteren Vorbereitungen gehört die Datenbereinigung. Da es so gut wie immer störende Vorkommnisse in den Daten gibt, die einer erfolgreichen Prognose eher entgegenarbeiten, müssen sie im Vorfeld gesäubert werden. Dazu gehören z.B. nicht erwartete fehlende Beobachtungen, wie bei einem Ausfall der Datenaufzeichnung oder zu extreme Ausreißer, durch defekte Sensoren. Je nach Anomalie kann es mehrere Wege geben, mit ihnen umzugehen. (Rob J Hyndman und George Athanasopoulos 2023; Vandeput und Makridakis 2021; Ertel 2018)

Schritt 4: Auswahl und Training von Modellen

Als nächstes wird ein Modell gesucht, das für die gegebenen Daten geeignet ist und mit ihnen eine präzise Vorhersage treffen kann. Genaueres zur Kategorie eines Modells für Zeitreihenprognosen ist in 2.3.2 in Form von Supervised Learning beschrieben und konkrete Modelle

werden in 2.5.5 als Modellarten für die Zeitreihenprognose vorgestellt.
Zur Wahl des geeigneten Modells kommt die Wahl geeigneter Hyperparameter. Sie hängt von verschiedenen Faktoren ab, wie der Art der Zeitreihe, der Menge an verfügbaren Daten und den spezifischen Anforderungen der Anwendung. Eine gründliche Analyse der Daten, einschließlich der Identifikation von Trend-, Saisonalitäts- oder Ausreißermustern, wie in Schritt 3, ist oft eine wichtige Hilfestellung für die Modellwahl.
Nach dem Festlegen von Modell und Hyperparametern erfolgt das Trainieren mit den Daten. Die so erreichte Generalisierung der Daten vom Modell kann anschließend auf ihre Genauigkeit getestet werden, was in Abschnitt 2.6 als Modellbewertung detaillierter erläutert wird.

Weiterführend ist es üblich mehrere unterschiedliche Modelle auszuprobieren und zu vergleichen, um das Beste herauszufiltern. Mehr dazu später in Abschnitt 2.7, der Validierungsverfahren. (Rob J Hyndman und George Athanasopoulos 2023; Tashman 2000)

Da die Suche nach geeigneten Hyperparametern oft aufwendig sein kann, gibt es mehrere Herangehensweisen. Eine davon ist das automatisierte Ausprobieren einer Liste von Werten. Dabei können beliebige Werte pro Parameter vorgemerkt werden, die mit allen möglichen Kombinationen der vorgemerkten Werte anderer Parameter ausprobiert werden. Alternativ können auch Grenzwerte von Minimum und Maximum mit einer Schrittweite überlegt werden, woraus iterativ alle möglichen Werte und damit Kombinationen erzeugt werden. Diese systematische Vorgehensweise wird auch als Grid-Search bezeichnet und ist eines von mehreren verschiedenen solcher Verfahren. (Vandeput und Makridakis 2021)

Schritt 5: Prognose und Bewertung des Modells

Nachdem das beste Modell und dessen Hyperparameter wie im vorherigen Schritt ausgewählt wurden, kann es für Vorhersagen genutzt werden.
Eine abschließende Bewertung des finalen Modells, wie später in Abschnitt 2.7 zu Validierungsverfahren erläutert, kann eine Einschätzung erbringen, ob das Modell die Anforderungen erfüllt und für den vorgesehenen Einsatzzweck außerhalb der Entwicklungsschritte verwendet werden kann. (Rob J Hyndman und George Athanasopoulos 2023)

Zeitreihenspezifische Begriffe

Im Folgenden werden speziell definierte Begriffe im Thema der Zeitreihenprognose aus der Literatur und in dieser Arbeit erläutert.

Für die Angabe der Vorhersageweite, also wie weit ein Modell in die Zukunft blicken soll bzw. wie viele Vorhersagen erzeugt werden sollen, wird unter Anderem der Begriff *Forecasting Horizon* oder auch *Lead Time* verwendet. (Tashman 2000)

So würden bei den Beobachtungen $\{B_{t-3}, B_{t-2}, B_{t-1}, B_t, B_{t+1}, B_{t+2}, B_{t+3}\}$, dem aktuellen Zeitpunkt t und einem Forecasting Horizon von 2, die Beobachtungen $\{B_{t+1}, B_{t+2}\}$ prognostiziert werden.

Wenn es um die Anzahl von Beobachtungen geht, die ein Modell als Eingabe/Input bzw. Vorlaufzeit bekommt, um damit eine Vorhersage zu treffen, wird dies auch *Lag* oder *Backshift* genannt. Vereinfacht wird damit angegeben, wie weit ein Modell in die Vergangenheit gucken darf, um die Zukunft vorherzusagen. (Rob J Hyndman und George Athanasopoulos 2023)

Das heißt, analog zu oben, bei den Beobachtungen $\{B_{t-3}, B_{t-2}, B_{t-1}, B_t, B_{t+1}, B_{t+2}, B_{t+3}\}$, dem aktuellen Zeitpunkt t und einem Lag von 3, würden die Beobachtungen $\{B_{t-2}, B_{t-1}, B_t\}$ als Eingabe dienen.

Auch wenn es nicht nur auf Zeitreihendaten zutrifft, ist die Unterscheidung von *stationären* und *nicht stationären* Daten für dieses Thema oft wichtig. Stationär bedeutet, dass sich die statistischen Eigenschaften einer Zeitreihe nicht über die Zeit verändern. Die zugrundeliegende Verteilung muss über alle Beobachtungen zeitlich konstant bleiben. So sind Zeitreihen mit Trend oder Saisonalität nicht stationär, weil diese die Beobachtungen zu verschiedenen Zeiten unterschiedlich beeinflussen. Rauschen hingegen belässt die Zeitreihe stationär, da die Daten, egal in welchem Abschnitt, sehr ähnlich zu jedem anderen Zeitpunkt aussehen. (Rob J Hyndman und George Athanasopoulos 2023)

2.5.5 Modellarten für die Zeitreihenprognose

Im Bereich der Zeitreihenprognose gibt es viele unterschiedliche Herangehensweisen. Eine eher nur charakteristische Unterteilung in die beiden Kategorien statistische- und ML-

Verfahren entsteht durch die Art und Weise wie ein Verfahren funktioniert bzw. wie die bekannten Daten verarbeitet werden. So eine Einordnung von Verfahren dient oft, so wie hier, eher einer Unterteilung zu Gliederungszwecken und kann je nach Betrachtungswinkel anders ausfallen. So gibt es im Allgemeinen keine konsistente Aussage zur Kategorie vieler Verfahren. Dies kommt vor allem dadurch, dass so gut wie alle ML-Techniken auf vorheriger statischer Arbeit basieren und es dadurch auch Überschneidungen von Merkmalen aus Statistik und ML gibt. Dennoch kann die Einordnung eines Verfahrens in Statistik oder ML auch hilfreich sein, um das passende Modell für eine Problemstellung zu wählen.

Statistische Modelle

Statistische Verfahren zeichnen sich dadurch aus, dass sie die statistischen Eigenschaften der Daten mit einbeziehen. So orientieren sich viele solcher Verfahren beispielsweise an der direkten Herleitung mathematischer Informationen der letzten vorhandenen Beobachtungen, um den nächsten daraus vorherzusagen. Zum Beispiel bildet das Verfahren des statistischen Moving Average den Durchschnitt der letzten Datenpunkte, was auch gleitender Mittelwert genannt wird und produziert aus dem daraus folgenden Wert die nächste Vorhersage. Allerdings ist dieses Verfahren unfähig bestimmte Verläufe in einer Zeitreihe wie einen Trend zu extrapolieren. Durch solche Eigenschaften können Verfahren je nach Zeitreihenstruktur ungeeignet sein. (Vandeput und Makridakis 2021; Neupert et al. 2020)

Ein anderes Beispiel eines statistischen Verfahrens ist Exponential Smoothing, das sich je nach Variante aus mehreren Komponenten zusammensetzen kann, die jeweils den Einfluss von Durchschnitt, Trendverlauf und Saisonalität der letzten Datenpunkte steuern, um durch die Kombination den nächsten Wert vorherzusagen. Dabei wird durch entsprechende Hyperparameter gesteuert, wieviel Bedeutung z.B. jüngere Beobachtungen gegenüber älteren haben, was beim statistischen Moving Average oft immer gleich gewichtet wird. Mit der Vorgabe dieser Parameter sollte es der Mensch, mithilfe von Kenntnissen über die Daten, dem Modell leichter machen, die Genauigkeit zu verbessern. Wie zu sehen ist, kann bei statistischen Verfahren Wissen über die Daten bei der Modellentwicklung aktiv helfen.
Statt auf einer Beschreibung von Trend und Saisonalität zu basieren, zielt das Verfahren Autoregressive Integrated Moving Average (ARIMA) auf die Korrelation in den Daten ab. Es baut auf dem ARMA-Modell auf, was mit dem autoregressiven (AR) Teil die lineare

Abhängigkeit einer Beobachtung zu ihren Vorherigen abbildet und einem Moving Average Modell (MA) die Residuen der vergangenen Vorhersagen für die nächste Beobachtung mit einbezieht. Das Moving Average Modell ist nicht zu verwechseln mit dem Verfahren des statistischen Moving Average (gleitendem Durchschnitt), da sie zwar vom Namen oft nicht differenziert werden, aber je nach Kontext ein bestimmtes der beiden Verfahren gemeint ist.
Auch das ARMA-Modell kann keinen Trend abbilden und erfordert stationäre Zeitreihen. Deshalb wurde das ARIMA-Modell entwickelt, das mit der zusätzlichen Differenzierung und Integration (I) Trends entgegenwirken und die Daten stationär machen kann. So können auch teilweise nicht stationäre Zeitreihen damit analysiert werden. Jedoch kann eine Zeitreihe auch durch eine Saisonalität, statt eines Trends, nicht stationär sein. Für diesen Fall gibt es weitere Fortführungen des Verfahrens wie z.B. das saisonale ARIMA (SARIMA), bei dem zusätzlich ein saisonales Muster mit einbezogen werden kann. Für ARMA, ARIMA und SARIMA müssen Parameter für jeweils enthaltene Teile von AR, I, MA und Saisonalität angegeben werden, wodurch wieder Wissen über die Zeitreihe vorhanden sein muss, um ein brauchbares Modell zu entwickeln. (Hamilton 1994; Vandeput und Makridakis 2021; Hewage und Perera 2021; Brockwell und Davis 2016; Rob J Hyndman und George Athanasopoulos 2023)

Machine Learning Modelle

Gegenüber statistischen Modellen haben die meisten ML-Verfahren die Eigenheit, dass sie keine Erklärung über die Vorhersage geben, sondern nur eine Antwort. Bei statistischen Modellen kann anhand der aktuellen Parameter oft leicht erkannt werden, warum das Modell einen Wert vorhergesagt hat. Hingegen ist der Prozess der internen Parameterfindung bzw. Modellanpassung von ML-Modellen überwiegend undurchsichtig. So ist es bei einem ML-Modell nicht immer ersichtlich, ob es einen Trend oder saisonalen Verlauf erkannt hat. Es wird abstrakt gesehen eine Beziehung der Eingabevariablen zu den Ausgabevariablen direkt anhand der historischen Daten entwickelt, anstatt diese Beziehung wie bei statistischen Modellen von außen als Hilfestellung zu beschreiben. Durch die wiederholte Anpassung interner Gewichte, anhand der Muster in den Daten, ist im Nachhinein nicht klar, wie der Endwert eines Gewichts zustande gekommen ist. Dennoch gibt es wichtige Entscheidungen von außen zu treffen. Zum Beispiel welche Features aus den Daten verwendet werden sollen und welches ML-Verfahren genutzt wird. Denn auch hier gibt es viele verschiedene Techniken mit unterschiedlichen

Stärken und Schwächen. Des Weiteren haben auch ML-Modelle Hyperparameter, die erheblichen Einfluss auf das Training haben und dementsprechend sorgfältig gewählt werden müssen.

Ein weiteres Merkmal für ML-Modelle ist, dass sie üblicherweise mehr und aufwendigere Rechenoperationen durchführen müssen und damit an eine maschinelle Ausführung gebunden sind. (Vandeput und Makridakis 2021; Makridakis et al. 2018)

Ein Beispiel, dass eher zur Statistik als ML eingeordnet wird, ist die lineare Regression (LR). Dennoch wird sie hier, in Bezug zu verwendeten Gewichten, die wiederholt angepasst werden und keiner Angabe über Trend oder Saisonalität von außen, als ML-Modell vorgestellt. Die LR nimmt eine lineare Beziehung zwischen ein oder mehreren Eingabedaten (Prädiktoren) und vorherzusagender abhängiger Variable (Zielvariable) an. Das Ziel ist eine lineare Funktion zu finden, die die Beziehung am besten abbildet. Dies geschieht indem in der Funktion für jeden Prädiktor ein Koeffizient als Gewicht verwendet wird. Diese werden wiederholt angepasst, um die Differenz von Vorhersagen und wahren Werten zu minimieren. Vorhersagen entstehen dabei durch die Kombination von Prädiktoren und deren Gewichten, also der aktuellen linearen Funktion. (Hastie et al. 2009; Hamilton 1994)

Eine komplett andere Herangehensweise bieten die ML-Modelle Decision Tree (Entscheidungsbaum) und Random Forest. Anders als LR sind Decision Trees in der Lage nicht-lineare Beziehungen zwischen Eingabe- und Zielvariable zu erkennen. Das liegt an der Funktionsweise, die vereinfacht ausgedrückt aus einer hierarchischen Sammlung von if-else-Bedingungen besteht. Dem zugrunde liegt eine Baumstruktur, bei dem jeder Knoten eine Bedingung darstellt. Bei Anwendung zu Regressionszwecken werden im Training die Beobachtungen durch den Aufbau dieser Bedingungen so gut es geht in Untergruppen aufgeteilt, indem numerische Vergleiche mit dem Knotenwert und dem Eingabewert erfolgen. Wie bei der LR wird die Unterteilung so vorgenommen, dass die Differenz zwischen Vorhersagen und deren wahren Werten minimiert wird. Wie groß der Baum wird bzw. wie viele rekursive immer kleinere Untergruppen möglich sind, setzt der Hyperparameter für die Tiefe fest. Sobald diese erreicht ist, enthalten die Endknoten, auch Blätter (Leafs) genannt, den jeweiligen Vorhersagewert für dessen Untergruppe. Es resultiert eine komplexe Struktur von Entscheidungsregeln, bei der ein einzelner Pfad einer spezifischen Kombination von Entscheidungen entspricht.

Erweiternd gibt es den Random Forest als Wald mit mehreren Entscheidungsbäumen. Diese Art Modelle werden als Ensemble-Modelle bezeichnet, da das Ergebnis durch eine Gruppe von einzelnen Modellen zustande kommt. So auch bei dem Random Forest, bei dem mehrere Decision Trees den gleichen Aufbau haben und am Ende jeweils eine Vorhersage abgeben. Der resultierende Mittelwert wird dann die abschließende Vorhersage. (Vandeput und Makridakis 2021; Hastie et al. 2009; Ertel 2018)

Neuronale Netze

Als weitere kategorische Unterteilung von Machine Learning gibt es noch das Deep Learning. Dabei geht es hauptsächlich um neuronale Netze, die von der Funktionsweise des menschlichen Gehirns inspiriert sind. Es besteht aus miteinander verbundenen künstlichen Neuronen, in Form von simplen mathematischen Funktionen, die in Schichten (Layern) organisiert sind. Die Schichten unterteilen sich in die drei Kategorien Input, Hidden und Output, von denen es mehrere Hidden-Schichten aber nur eine Input- und Output-Schicht gibt. Die Verarbeitung erfolgt durch Annahme der Eingabedaten in die Input-Schicht, dessen Ausgabe die Eingabe der ersten Hidden-Schicht ist. Dies wiederholt sich innerhalb der Hidden-Schicht, bis die Ausgabe an die Output-Schicht übergeben und dort final bestimmt wird. In der Input-Schicht gibt es dabei so viele Neuronen, wie es Eingabedaten bzw. Features gibt. Wogegen in jeder Hidden-Schicht beliebig viele Neuronen vorhanden sein können. Die Anzahl der Neuronen in der Output-Schicht ist abhängig von der Aufgabe des Netzes. Bei einer Regressionsaufgabe bündelt alles in nur ein Neuron, das den endgültigen Wert angibt. Im Normalfall ist anhand dieser Struktur ein Neuron mit allen Neuronen der nächsten Schicht verbunden, was sich Feed-Forward-Network nennt. Dadurch kommen komplexe nicht-lineare Berechnungen zustande, indem die Ausgabe jedes Neurons einer Schicht, an jedes der Nächsten weitergegeben wird. Denn alle Neuronen enthalten anpassbare Gewichte, die via Trainingsverfahren mit speziellen Algorithmen verändert werden, um die Differenz zwischen Vorhersage und Beobachtung zu minimieren. Da dies allgemein als Lernen bezeichnet wird, ist bei einem großen neuronalen Netz mit enorm vielen Neuronen von Deep Learning die Rede.
Es gibt noch weitere wichtige Einzelheiten, die hier aber den Rahmen sprengen würden und deshalb nicht weiter erklärt werden. (Hastie et al. 2009; Vandeput und Makridakis 2021; Skansi 2018)

Weitere neuronale Netzarchitekturen entstanden für spezifischere Thematiken. So gibt es auch eine für die Verarbeitung von sequenziellen Daten, wie bei der Zeitreihenprognose, wie z.B. rekurrente neuronale Netze (RNN). Anders als bei Feed-Forward-Netzen werden die Informationen nicht nur in eine Richtung von Schicht zu Schicht gereicht, sondern zusätzlich über eine Rückkopplungsschleife die Zustände vergangener Zeitschritte in die Berechnungen mit einbezogen. Dies geschieht, indem Kopien der vorherigen Werte einer Schicht, die die rekurrenten Knoten enthält, gespeichert und als zusätzliche Eingaben im nächsten Schritt verwendet werden. Dadurch entsteht die Fähigkeit weiter zurückliegende Beobachtungen und damit Abhängigkeiten im sequenziellen Verlauf der Daten zu erfassen. (Makridakis et al. 2018; Falatouri et al. 2022; Skansi 2018)

2.6 Modellbewertung

Dieser Abschnitt gibt einen Überblick zur Bewertung bzw. Evaluierung von Modellen. Des Weiteren werden exemplarisch üblich verwendete Verlustfunktionen anhand ihrer jeweiligen Eigenschaften und Unterschiede und abschließend die Relevanz der Auswahl von Testdaten erläutert.

Die enthaltenen Variablen wurden zur besseren Lesbarkeit vereinheitlicht. In Tabelle 1 ist eine Übersicht der Variablen und zugehörigen Bedeutungen.

Tabelle 1: Übersicht der in Verlustfunktionen verwendeten Variablen und Bedeutungen

Variable	Bedeutung
y	Wert eines Datenpunkts, auch Beobachtung genannt
$\hat{y}$	Geschätzter Wert der Prognose/Vorhersage zu y
n	Anzahl von Datenpunkten (für die ein y und $\hat{y}$ vorliegt)
e	Fehlerwert
t	Zeitpunkt eines Datenpunkts in einer Zeitreihe

2.6.1 Benötigtes Maß für Genauigkeit

Sobald ein Modell ausgewählt und trainiert wurde, gilt es zu testen, wie gut oder schlecht dessen Vorhersagen sind bzw. wie weit sie von den wahren Werten abweichen. Diese Bewertung ist grundsätzlich erforderlich, um Vergleiche zwischen verschiedenen Modellen oder ein und demselben Modell, aber mit unterschiedlichen Parametern durchzuführen. Denn es gibt keine allgemeingültigen Einstellungen eines Modells, die immer gute Ergebnisse liefern. (Vandeput und Makridakis 2021)

Um einschätzen zu können, ob ein Modell gut oder schlecht prognostiziert, werden die Werte der vorliegenden Daten mit den Vorhersagewerten des Modells verglichen und die Differenz als Fehlerwert festgehalten, der in diesem Kontext als Genauigkeit des Modells bezeichnet werden kann. (Vandeput und Makridakis 2021)

Definition für Fehlerwert

Der Fehlerwert e für einen einzelnen Zeitpunkt t ergibt sich aus der Differenz der Werte aus Vorhersage $\hat{y}$ und den wirklichen Daten y.

$$e_t = y_t - \hat{y}_t$$

Formel 1: Definition des Fehlerwerts eines Vorhersagezeitpunkts (Hyndman und Koehler 2006)

Eine Einschätzung des Modells durch den damit ermittelten Fehler für einen einzelnen Vorhersagepunkt ist zwar bereits nutzbar, kann aber in ihrer Aussagekraft noch verbessert werden. Dazu werden die Fehlerwerte von mehreren Zeitpunkten, statt nur einem, einbezogen. Für die Art und Weise wie mehrere Fehlerwerte zu einem resultierenden Ergebnis aggregiert werden gibt es verschiedene Funktionen, die sich in ihrer Berechnung und Interpretation unterscheiden. Solche Funktionen werden meistens Fehler- oder Verlustfunktionen genannt und dienen für Modelle als Optimierungsziel, das minimiert oder maximiert werden soll.

2.6.2 Konkrete Verlustfunktionen

Bias

Der Bias oder auch die Verzerrung stellt die Gesamtrichtung des historischen Durchschnittsfehlers dar. Er gibt an, ob die Prognosen im Durchschnitt zu hoch (überschätzt) oder zu niedrig (unterschätzt) waren. (Vandeput und Makridakis 2021)

Definiert wird er durch den Durchschnitt aller Fehler. (Vandeput und Makridakis 2021)

$$bias = \frac{1}{n}\sum_{n} e_t$$

Formel 2: Bias (Vandeput und Makridakis 2021)

Der Bias allein reicht allerdings nicht aus, um die Genauigkeit zu beurteilen. Da ein positiver Fehler einen negativen Fehler zu einem anderen Zeitpunkt ausgleichen kann, könnte ein Modell einen sehr geringen Bias erreichen und gleichzeitig nicht genau sein. Dennoch bedeutet ein hoher Bias eine stark verzerrte Prognose und ist ein Hinweis darauf, dass mit dem Modell etwas nicht stimmt. (Vandeput und Makridakis 2021)

Zu beachten ist die begrenzte Aussagekraft, da die Formel einen absoluten Wert produziert. Beispielsweise wäre ohne weitere Informationen über die Zeitreihe nicht klar, ob das Ergebnis 42 einen hohen oder niedrigen Fehler bedeutet. Verglichen wäre er bei Daten im sechsstelligen Bereich als besser einzustufen als im Zweistelligen. (Vandeput und Makridakis 2021)

Natürlich könnte die interpretierende Person die Daten einsehen und kennt dadurch dessen Wertebereich. In diesem Fall ist ein absoluter Fehler ein gutes und sehr aussagekräftiges Genauigkeitsmaß, weil exakt abgelesen werden kann um wieviel Einheiten die Vorhersage durchschnittlich abweicht.

Um diese Beurteilung direkt in die Berechnung zu integrieren, kann der absolute Fehler mit den Werten der Zeitreihe in Beziehung gesetzt werden. Das kann mit der Gesamtsumme beider bzw. dem Durchschnitt beider erfolgen. Daraus resultiert ein skalierter Bias. (Vandeput und Makridakis 2021)

$$bias\% = \frac{\frac{1}{n}\sum e_t}{\frac{1}{n}\sum y_t} = \frac{\sum e_t}{\sum y_t}$$

Formel 3: Skalierter Bias (bias%) (Vandeput und Makridakis 2021)

Mean Absolute Error (MAE)

Der MAE ergibt sich, wie der Name sagt, aus dem Durchschnitt aller absoluten Fehlerwerte.

$$MAE = \frac{1}{n}\sum_{n}|e_t|$$

Formel 4: Mean Absolute Error (Vandeput und Makridakis 2021)

Wie beim Bias handelt sich um eine absolute Zahl, wodurch sie nur im Bezug zum Durchschnitt aller Werte der Zeitreihe als großer oder kleiner Fehler eingeordnet werden kann. Deshalb wird sie üblicherweise dadurch geteilt, um einen skalierten prozentualen Fehlerwert zu erhalten. (Vandeput und Makridakis 2021)

$$MAE\% = \frac{\frac{1}{n}\sum|e_t|}{\frac{1}{n}\sum y_t} = \frac{\sum|e_t|}{\sum y_t}$$

Formel 5: Skalierter Mean Absolute Error (Vandeput und Makridakis 2021)

Wissenswert ist, wodurch die Minimierung des MAE erreicht werden kann. Zum einen gibt es die offensichtliche Möglichkeit, dass alle Vorhersagen mit den echten Beobachtungen übereinstimmen. Eine Minimierung wird laut (Vandeput und Makridakis 2021) aber auch durch Vorhersagen erreicht, bei denen genauso viele $\hat{y}$ kleiner y sind, wie $\hat{y}$ größer y sind bzw. so oft unterschätzt wie überschätzt wird. Anders gesagt kann der MAE auch minimiert werden, indem die gesamte Prognose den gleichen Median annimmt, wie in den Daten vorliegt. Diese Eigenschaft ist zu erwähnen, weil bei Daten mit verschobener Verteilung der Median oft niedriger als der Durchschnitt liegt, wodurch eine Vorhersage aufbauend auf der Minimierung so einer

Verlustfunktion auch verschoben wird und meistens in einer Unterschätzung resultiert, wie in Abbildung 6 zu sehen. (Vandeput und Makridakis 2021)

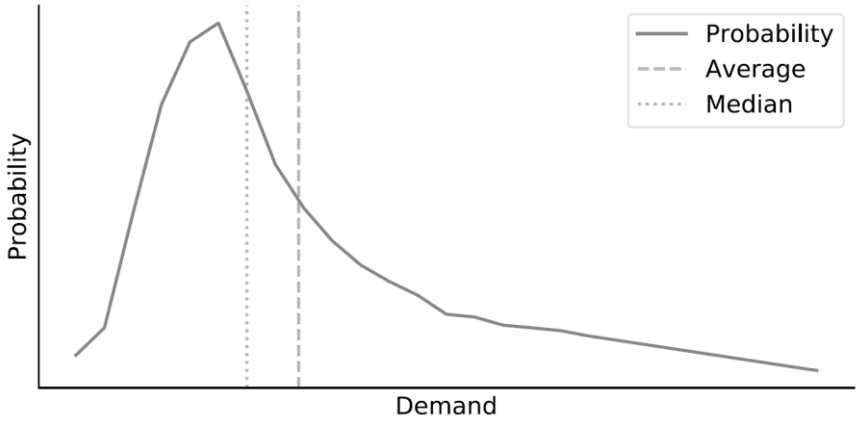


Abbildung 6: Vergleich Median vs. Durchschnitt (Vandeput und Makridakis 2021, 24)

Außerdem ist bei absoluten Fehlern, die zum Median optimiert sind, zu beachten, dass sie für Zeitreihen mit vielen Werten nahe Null und nur sporadisch hohen Anstiegen, den Prognosen mit niedrigen Werten eine bessere Genauigkeit zuschreiben, als tatsächlich vorliegt. (Makridakis et al. 2022)

Mean Absolute Percentage Error (MAPE)

Neben absoluten gibt es auch direkt zu den Daten skalierte Fehlermaße. Einer davon ist der MAPE, der aus dem Durchschnitt der Division von jedem einzelnen absoluten Fehlerwert und zugehörigem Datenpunkt gebildet wird. Anders ausgedrückt bestimmt er den Durchschnitt der prozentualen absoluten Fehler. (Vandeput und Makridakis 2021)

$$MAPE = \frac{1}{n}\sum_{n}\frac{|e_t|}{y_t}$$

Formel 6: Mean Absolute Percentage Error (Vandeput und Makridakis 2021)

Der MAPE ist allerdings mit Vorsicht als Genauigkeitsmaß zu benutzten, denn wie man anhand der Formel erkennen kann, wird jeder Fehler durch den zugehörigen Datenpunkt geteilt. Daraus resultiert, dass hohe Werte für e bei niedrigen y einen großen Einfluss auf die Berechnung haben. Dadurch wird eine, durch den MAPE optimierte, Vorhersage in einem solchen Fall die Daten wahrscheinlich unterschätzen. Beispielsweise kann bei einer Vorhersage von Null

maximal 1 als Quotient bzw. 100% als Fehler herauskommen. Wogegen eine hohe Vorhersage einen Fehler auch über 100% produzieren könnte, je nachdem um wieviel größer $\hat{y}$ gegenüber dem y ist. (Vandeput und Makridakis 2021; Hyndman und Koehler 2006)

Nicht zu verwechseln ist der MAPE mit dem skalierten MAE%. Teilweise wird der MAE normalisiert und dann als MAPE bezeichnet, was zu viel Verwirrung bei Diskussionen führen kann. Um so etwas zu vermeiden, sollte immer klar angegeben werden, wie der Fehler berechnet wurde. (Vandeput und Makridakis 2021)

Mean Squared Error (MSE)

Ein häufig bei Algorithmen, insbesondere bei ML, als Verlustfunktion verwendetes Fehlermaß ist der MSE. Dies kommt von seiner schnellen Berechenbarkeit und einfachen Manipulation gegenüber anderen Maßen. Wie der Name sagt, geht es hierbei um den Durchschnitt der quadrierten Fehlerwerte. Da er nicht auf den ursprünglichen Fehler skaliert wird, kann er nicht ohne Weiteres mit der Skala der eigentlichen Daten in Beziehung gesetzt werden. (Vandeput und Makridakis 2021)

$$MSE = \frac{1}{n}\sum_{n} e_t^2$$

Formel 7: Mean Squared Error (Vandeput und Makridakis 2021)

Durch die Definition ist zu erkennen, dass das Hauptmerkmal des MSE die Quadrierung des Fehlers ist. Resultierend daraus werden höhere Fehlerwerte stärker gewichtet als niedrige, verglichen beispielsweise mit dem MAE bei dem alle Fehler gleich gewichtet werden. Ein großer Wert für den MSE muss somit keine gänzlich schlechte Vorhersage bedeuten, sondern kann durch wenige, bis sogar nur eine einzige, aber extremere Abweichungen verursacht werden. (Vandeput und Makridakis 2021)

Wie beim MAE gibt es auch beim MSE eine weitere allgemeine Möglichkeit eine Minimierung zu erreichen. Anders als der MAE wird der MSE laut (Vandeput und Makridakis 2021) durch Vorhersagen minimiert, die den gleichen Durchschnitt wie die Daten haben, statt des Medians. Somit kann durch eine schiefe Verteilung in den Daten die Verwendung des MSE nicht zu einer verschobenen Prognose führen.

Root Mean Squared Error (RMSE)

Der RMSE basiert auf dem MSE und erweitert ihn durch die Quadratwurzel. Dadurch wird der quadratische Fehlerwert wieder in die ursprüngliche Einheit der Daten zurück transformiert und ist wie der MAE als absoluter Wert zu verwenden. (Vandeput und Makridakis 2021)

$$RMSE = \sqrt{\frac{1}{n}\sum_{n} e_t^2}$$

Formel 8: Root Mean Squared Error (Vandeput und Makridakis 2021)

Wie auch der MAE ist der RMSE als absoluter Wert nicht sehr aussagekräftig und kann dafür zu einem relativen skalierten Wert umgeformt werden. (Vandeput und Makridakis 2021)

$$RMSE\% = \frac{\sqrt{\frac{1}{n}\sum e_t^2}}{\frac{1}{n}\sum y_t}$$

Formel 9: Skalierter Root Mean Squared Error (Vandeput und Makridakis 2021)

Eigenschaften, wie die Minimierung anhand des übereinstimmenden Durchschnitts bei Daten und Prognosen, übernimmt der RMSE vom MSE. (Vandeput und Makridakis 2021)

Relative Fehlerwerte und Fehlermaße

Ähnlich zu der prozentualen Skalierung eines Fehlerwerts in Bezug zu den Daten, wie beim MAPE, gibt es auch die Möglichkeit einer Skalierung des Fehlerwerts in Bezug zum selben Fehlermaß aber eines anderen Modells. Damit wird der Fehler zum Unterschied zweier Modelle mit den Prognosen zu denselben Daten relativiert. (Hyndman und Koehler 2006)

$$re_t = \frac{e_t}{e_t^*}$$

Formel 10: Relativer Fehler zu einem anderen Modell (Hyndman und Koehler 2006)

Dabei steht e_t^* für den Fehlerwert des Referenzmodells zu dem Zeitpunkt t. Zur Integration in den Fehlerterm wird der relative Fehlerwert wie vorher der normale Fehlerwert e_t in die jeweilige Funktion eingesetzt. Diese Abwandlung kann auf mehrere Verlustfunktionen angewandt

werden, um zum Beispiel den Mean Relative Absolute Error (MRAE) zu bestimmen. (Hyndman und Koehler 2006)

Die zweite Skalierungsoption besteht analog zum Bias%, MAE% und RMSE% in der Division des endgültigen Fehlerwerts durch den des Referenzmodells. Auch diese Variante kann dadurch auf mehrere verschiedene Fehlermaße angewandt werden. (Hyndman und Koehler 2006)
Beispielhaft für den MAE:

$$relative\ MAE = \frac{MAE}{MAE^*}$$

Formel 11: Relativer MAE zu einem anderen Modell (Hyndman und Koehler 2006)

Die zweite Option hat des Weiteren den Vorteil der einfachen Interpretation. Bei einem Wert unter 1 ist das aktuelle Modell besser als das Referenzmodell und gegenteilig ist das aktuelle Modell schlechter bei einem Wert über 1. (Hyndman und Koehler 2006)

Als Referenzmodell oder auch Benchmarkmodell für den Referenzfehler wird üblicherweise ein naives Modell gewählt, welches als Vorhersage $\hat{y}_t$ gleich dem vorigen Datenpunkt y_{t-1} wählt. (Hyndman und Koehler 2006)

Mean Absolute Scaled Error (MASE) und Root Mean Scaled Error (RMSSE)

Den Ansatz der relative Fehlerwerte haben (Hyndman und Koehler 2006) noch weiter geführt, indem die Vorhersagen des naiven Referenzmodells (Random Walk) in das jeweilige Fehlermaß integriert wird. So wird jeder Fehlerwert anhand des Fehlermaßes vom Referenzmodell skaliert und ist von der Skala der Daten unabhängig.

Der Mean Absolute Scaled Error (MASE) entsteht beispielsweise durch die Abwandlung des MAE, indem der MAE des naiven Modells als Nenner eingesetzt wird.

$$se_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|} \qquad MASE = \frac{1}{n}\sum_{n}|se_t|$$

Formel 12: Mean Absolute Scaled Error (Hyndman und Koehler 2006)

Zur Interpretation gilt wie bei den anderen relativen Fehlermaßen, dass ein Modell mit einem MASE unter 1, über den Durchschnitt, kleinere Fehlerwerte gegenüber dem Naiven macht. (Hyndman und Koehler 2006)

Analog zu den anderen relativen Fehlern kann der skalierte Fehlerwert se_t wieder für mehrere unterschiedliche Verlustfunktionen, wie auch dem RMSE, adaptiert werden. (Hyndman und Koehler 2006)

Hier steht die zusätzliche Variable h für den Vorhersageweite/Forecasting-Horizon, der ohne eine solche Angabe bei 1 liegt und damit einen Schritt in die Zukunft prognostiziert.

$$RMSSE = \sqrt{\frac{\frac{1}{h}\sum_{t=n+1}^{n+h}(y_t - \hat{y}_t)^2}{\frac{1}{n-1}\sum_{t=2}^{n}(y_t - y_{t-1})^2}}$$

Formel 13: Root Mean Squared Scaled Error (Makridakis et al. 2022)

Im Gegensatz zu anderen Maßen mit ähnlichen statistischen Eigenschaften, wie relativen oder prozentualen Fehlermaßen wie dem MAPE, kann der RMSSE sicher auf Daten mit Werten nahe oder gleich Null verwendet werden, da er nicht auf einer Division dieser niedrigen Werte basiert. (Makridakis et al. 2022)

Vergleich und Überblick

Zusammenfassend können einige Eigenschaften von Fehlermaßen miteinander verglichen werden. Diese unterscheidenden Merkmale sollten bei der Wahl, ausgehend von der Beschaffenheit der Daten und dem Anwendungsfall, mit beachtet werden, um klar zu interpretierende Ergebnisse zu erhalten. Außerdem sind Gegebenheiten wie Ausreißer, Skalierung und Wertebereich der Daten ausschlaggebend.

Zum Thema Ausreißer lässt sich generell sagen, dass Fehlermaße, die durch Annäherung an den Median der Daten minimiert werden, wie der MAE, robuster gegenüber Ausreißern sind, als welche, die durch den Durchschnitt minimiert werden, wie beim RMSE. (Vandeput und Makridakis 2021) Außerdem geben Verlustfunktionen mit Quadrierung des Fehlerwerts, wie der MSE und RMSE, Ausreißern zusätzlich noch mehr Gewicht, wogegen sie sich bei der

Verbesserung des niedrigsten Vorhersagewertes kaum verändern. (Vandeput und Makridakis 2021; Hyndman und Koehler 2006)

Allerdings ist Robustheit gegenüber Ausreißern nicht immer zielführend, denn bei Zeitreihen mit hauptsächlich niedrigen Werten und nur sporadischen, kurzen und hohen Anstiegen, sind die Ausreißer entscheidend für eine genaue Vorhersage. In so einem Fall von unregelmäßigen Daten ist ein Fehlermaß, wie der MAE, nicht gut geeignet. (Vandeput und Makridakis 2021)

Im Hinblick auf Skalierungsabhängigkeiten geht es oft um den Vergleich der Genauigkeit zwischen mehreren Modellen anhand ein und derselben Zeitreihe. Dies kann mithilfe skalenabhängiger Fehlermaße, wie dem MAE, MSE oder RMSE, problemlos durchgeführt werden. Dies ändert sich jedoch, sobald Fehlermaße zwischen mehreren unterschiedlichen Zeitreihen verglichen werden sollen. Da verschiedene Zeitreihen unterschiedliche Maßstäbe haben können, beispielsweise wird die Einheit in Millionen und in einer anderen in Tausend angegeben, sollten Maße, wie der MAE und RMSE bzw. allgemein skalenabhängige Fehlermaße, nicht zum Vergleichen benutzt werden. Falls einer der Fehlerwerte anhand des unterschiedlichen Faktors einfach umskaliert werden würde, wie zum Angleichen eines Fehlers zugehörig zu einer Währung zum Fehler einer anderen Währung, würde der Fehlerwert nicht nur skaliert, sondern auch verändert werden. (Hyndman und Koehler 2006; Tashman 2000)

Stattdessen sind in solchen Anwendungsfällen prozentuale Fehlermaße, wie der MAPE, vorzuziehen, da diese bei Daten mit einer natürlichen Null skalenunabhängig sind. Allerdings kann die Verteilung bei solchen Maßen stark verzerrt sein. Insbesondere, wenn Werte nahe Null in den Daten vorkommen. (Hyndman und Koehler 2006; Tashman 2000) Für solche Fälle können wiederum andere Fehlermaße verwendet werden, die hier nicht weiter erklärt sind. Beispielsweise der in (Vokurka et al. 1996) verwendete Median Absolute Percentage Error (MdAPE) oder symmetrische Fehlermaße, wie der symmetrische MAPE (Armstrong 1985). In Bezug zu prozentualen Fehlermaßen existiert zusätzlich die Möglichkeit von negativen Fehlern, durch zum Beispiel Temperaturdaten mit negativen Werten. Das Problem dabei ist die oft in prozentualen Fehlermaßen enthaltene Division durch den Wert y, durch die der Fehlerwert selbst negativ wird. Um solch ein Maß trotzdem zu nutzen, könnte der Nenner zu einem absoluten Betrag geändert werden. (Hyndman und Koehler 2006)

Darüber hinaus sollten relative Fehlermaße mit der Verwendung eines naiven Modells als Referenz genutzt werden, wenn über unterschiedliche Zeitreihen gemittelt werden soll, die sich in ihrer Volatilität unterscheiden. So können Vergleiche der Genauigkeit von mehreren Modellen über verschiedene Arten von Zeitreihen durchgeführt werden. (Tashman 2000; Hyndman und Koehler 2006)

Zusammenfassend gibt es in (Hyndman und Koehler 2006) folgende beispielhafte Ausschlussstrategie zur Wahl des Fehlermaßes. Wenn die verwendeten Zeitreihen die gleiche Skalierung bzw. Einheitsgröße haben, kann der MAE bevorzugt werden, weil er einfach zu interpretieren ist. Wenn sich die Skalierung unterscheidet, aber alle Daten positiv und weit größer als Null sind, kann der MAPE gewählt werden. In Situationen, in denen es sehr unterschiedliche Skalen gibt und zusätzlich Werte, die nahe Null oder negativ sind, wird der MASE als das beste verfügbare Maß zur Prognosegenauigkeit vorgeschlagen. Natürlich gibt es auch Situationen, in denen einige der anderen bestehenden Fehlermaße immer noch vorzuziehen sind.

Schlussendlich kommt es auf die Gegebenheiten der Daten, Anforderungen und gewünschten Ziele an, welche Fehlermaße geeignet sind. Zur darauf folgenden Eingrenzung können zwei Fragen von (Tashman 2000) in Bezug zu den arithmetischen Unterschieden gestellt werden. Die eine betrifft die Wahl des Fehlermaßes, indem entschieden werden muss, ob der quadratische, prozentuale oder relative Fehler gemittelt werden soll. Die Zweite dreht sich um den geeigneten statistischen Operator. Ob der Median, das arithmetische Mittel oder das geometrische Mittel (hier nicht weiter aufgeführt) verwendet werden soll. Durch die bekannten Unterschiede dieser Kategorien und anhand der oben genannten Gegebenheiten, kann ein geeignetes Fehlermaß ausgewählt werden.

Abschließend als ein Beispiel kam für die M5-Competition 2020, siehe 2.5.3, die Ansicht auf, dass auf skalierten Fehlern basierende Maße wahrscheinlich die geeignetsten statistischen Eigenschaften aufweisen. Des Weiteren war das Ziel des Wettbewerbs den durchschnittlichen Umsatz genau vorherzusagen, was für ein Fehlermaß sprach, welches auf quadrierten Fehlern basiert, die für den Mittelwert optimiert sind. Außerdem war eine Robustheit gegen Werte nahe oder gleich Null durch die verwendeten Zeitreihen gefordert. Durch diese Anforderungen wurde dort der RMSSE verwendet. (Makridakis et al. 2022)

2.6.3 In-Sample vs. Out-of-Sample

Ein wichtiger Unterschied bei der Bewertung eines Modells liegt in dem Ursprung der Daten, die für die Ermittlung des Fehlermaßes genutzt werden. Zum einen können im Training verwendete Daten auch als Testdaten wiederverwendet werden, was in der Literatur oft als in-sample Daten bezeichnet wird, da die Daten im Trainingssample enthalten sind. Zum anderen kann vor dem Training ein gewisser Anteil an Daten aus den Trainingsdaten entfernt und als Testdaten vorgehalten werden, welche meistens als out-of-sample Daten bezeichnet werden. (Tashman 2000)

Welche dieser zwei Arten genutzt wird, beeinflusst die Bewertung des Modells erheblich. Denn wenn ein Modell mit Daten getestet wird, die bereits mit im Training vorkamen, wird eine resultierende Bewertung wahrscheinlich besser ausfallen, als sie tatsächlich ist, was erstmals von (S. Larson 1931) als zu optimistisch beschrieben wurde. Schließlich ist es unwahrscheinlich, dass die Nuancen der Vergangenheit auch in der Zukunft bestehen bleiben. Deshalb ist es sehr wichtig ein Modell mithilfe von Daten zu testen, die es noch nicht gesehen hat. Nur so wird die Generalisierungsfähigkeit für neue unbekannte Daten authentisch geprüft. (Tashman 2000)

Generell sind sich Prognostiker im Allgemeinen einig, dass die Genauigkeit von Prognosemethoden anhand von out-of-sample Tests und nicht anhand der Anpassung an vergangene Daten, in-sample Tests, beurteilt werden sollte. Zudem wurde in der ersten M-Competition 1982 und mehreren empirischen Studien gezeigt, dass Prognosefehler für out-of-sample Daten die Fehler von in-sample Daten auch bei recht kurzen Zeithorizonten übersteigen. (Tashman 2000; Spyros Makridakis et al. 1982; Schnaubelt 2019)

Obwohl es deutlich ist, dass Testdaten von Trainingsdaten separiert werden sollten, gibt es oft das Problem einer sehr geringen Datenmenge. So kann die Aufteilung des Datensatzes dazu führen, dass ein Modell die Daten nicht ausreichend abstrahieren kann und damit Underfitting oder Overfitting auftritt. Eine Möglichkeit ist das Warten und Bewerten durch neue Daten in Echtzeit, was allerdings meistens erhebliche praktische Einschränkungen mit sich bringt, auch wenn dies z.B. in der M2-Competition (Makridakis et al. 1993) für mehrere Monate durchgeführt wurde.

Aber auch wenn nur wenig Daten vorhanden sind, sollte davon abgesehen werden mit in-sample Daten zu Bewerten. Denn durch das Entwickeln eines Modells anhand der Trainingsdaten wird es diese nicht generalisieren, sondern übertrainieren womit Overfitting eintritt. Zum Beispiel wenn die Modellkomplexität erhöht wird, bis der Fehler bei den Trainingsdaten gegen Null tendiert, gezeigt in Abbildung 7. In dem Fall würde die Performance bei zukünftigen Daten sehr wahrscheinlich erheblich schlechter werden. (Hastie et al. 2009)

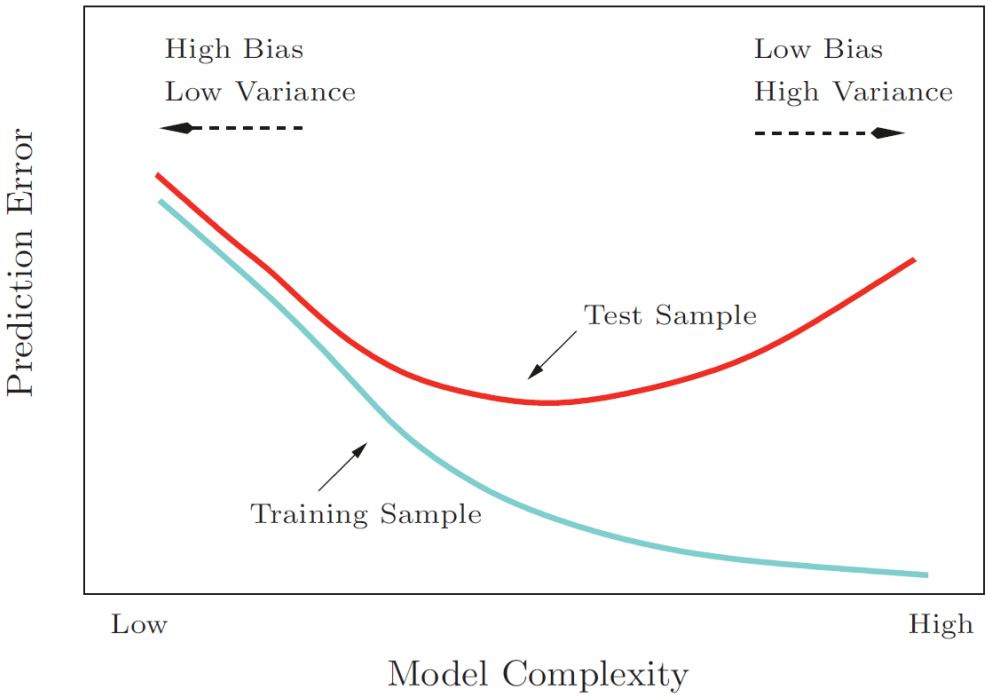


Abbildung 7: Verlauf von Trainingsfehler und Testfehler in Bezug zur Modelkomplexität (Hastie et al. 2009, 38)

Das Testen durch in-sample Daten ist zwar nicht für die endgültige Schätzung der Genauigkeit geeignet, wird aber oft zur Einschätzung während des Trainings genutzt. So kann die Entwicklung eines Modells beobachtet und ggf. angepasst werden, bevor ein Problem, wie etwa Overfitting, erst bei der abschließenden Leistungsbewertung auffällt oder gar unentdeckt bleibt. (Tashman 2000)

Daher können in-sample und out-of-sample am besten kombiniert genutzt werden, um ein umfassendes Verständnis der Modellleistung zu erhalten. In-sample für die Performance während

des Trainings und out-of-sample für die Schätzung der Leistungsfähigkeit in realen Anwendungen. (Schnaubelt 2019)

Abschließend kann allgemein bei der Rede von Testdaten und keiner weiteren Angabe mehrheitlich davon ausgegangen werden, dass sie als out-of-sample Daten nicht in den Trainingsdaten vorkommen, wie auch des Weiteren in dieser Arbeit.

2.7 Validierungsverfahren

2.7.1 Was ist Validierung?

Wie im vorigen Abschnitt erläutert, kann ein Modell mit Fehlermaßen basierend auf einem Testdatensatz bewertet werden. Dies wird unter anderem Validierung genannt und wird hauptsächlich für zwei Zwecke eingesetzt. Erstens dient es der Auswahl eines Modells aus mehreren Kandidaten, auch *Model Selection* genannt. Dabei geht es grundsätzlich um die Wahl des geeignetsten Modells aus verschiedenen ML-Methoden und/oder das anschließende Herausfinden der Kombination von geeigneten Hyperparametern, mit denen das Modell die beste Leistung erzielt. Zweitens wird die eigentliche Modellbewertung, auch *Model Assessment*, in Bezug zu den Daten gebildet, indem z.B. zur feineren Einschätzung auch mehr Datensätze als bei der Model Selection einbezogen werden können. (Schnaubelt 2019; Hastie et al. 2009)

2.7.2 Testdaten vs. Validierungsdaten

Wie erwähnt werden Model Selection und Model Assessment voneinander unterschieden, obwohl ihr Ablauf fast identisch ist. Diese Differenzierung ist sehr wichtig und betrifft wieder die Wahl der Daten. Für die Bewertung eines finalen Modells und Schätzen dessen Prognosefehlers wird ein Testdatensatz von den Gesamtdaten abgetrennt und verwendet. Jedoch sollten dieselben Testdaten nicht zur Suche nach den optimalen Hyperparametern genutzt werden, da diese dann speziell nur für die Testdaten optimiert werden könnten. In anderen Worten würde das Overfitting der Hyperparameter für die Testdaten bedeuten, was von der Verwendung der Testdaten zur Entwicklung, statt Bewertung des Modells ausgelöst wird. (Vandeput und Makridakis 2021)

Deshalb sollte zum Validieren der Hyperparameter immer ein zusätzlicher Validierungsdatensatz genutzt werden. Dieser sollte, wie die Testdaten, vor dem Training von den Gesamtdaten entnommen werden, damit er als out-of-sample Datensatz weder zum Training noch zum Testen beeinflusst wird oder beeinflussen kann. (Vandeput und Makridakis 2021)

Wenn ausreichend Daten vorhanden sind, ist das beste Vorgehen die Unterteilung des Datensatzes in drei Teile: Trainingsdaten, Validierungsdaten und Testdaten. Die Trainingsdaten werden genutzt, um das Modell zu trainieren, mit den Validierungsdaten wird der Prognosefehler zur Model Selection geschätzt und anhand der Testdaten entsteht erst am Ende, wenn das finale Modell feststeht, die Bewertung via Model Assessment. (Hastie et al. 2009)

Allerdings ist es schwierig zu sagen wie viele Datenpunkte in jedem der drei Datensätze enthalten sein sollten, weil es auf die Beschaffenheit und Menge der vorliegenden Daten ankommt. Außerdem hat die Komplexität des Modells Einfluss auf die Mindestgröße des Trainingssatzes bezüglich Underfitting. Eine typische Aufteilung ist 50% für Training und jeweils 25% für Validation und Testen siehe Abbildung 8. (Hastie et al. 2009)

Abbildung 8: Typische Datenaufteilung in Training, Validation und Test nach (Hastie et al. 2009, 222)

Allgemein üblich und auch im weiteren Verlauf dieser Arbeit wird ein Validierungsdatensatz auch als Testdatensatz bezeichnet, wenn es um das Thema Validierung geht. Das kommt von der starken Überschneidung des Einsatzzwecks und Vorgangs. Dies ist nicht zu verwechseln mit dem eigentlichen Testdatensatz, der erst nach der Validierung zur endgültigen Bewertung eines Modells herangezogen wird und vom Prozess der Validation ausgeschlossen ist.

2.7.3 Kreuzvalidierung

Allgemeines Verfahren der Kreuzvalidierung

Neben der manuellen Einteilung in die unterschiedlichen Datensätze gibt es verschiedene statistische Verfahren zur systematischen Organisation der Daten zur Validierung. Die, vor allem

für unabhängige und identisch verteilte Daten, wahrscheinlich am häufigsten genutzten sind die Kreuzvalidierungsverfahren. Diese haben eine lange Geschichte mit solider theoretischer Grundlage. (Cerqueira et al. 2017)

Die Kreuzvalidierung lässt sich auf die Arbeiten von (Stone 1974), (Allen 1974) und (S. Geisser 1975) zurückführen. Hierbei wird kein zusammenhängender Teilabschnitt der Trainingsdaten als Testdatensatz separiert, sondern die gesamten Trainingsdaten für verschiedene Unterteilungen in Trainings- und Testdaten innerhalb des Verfahrens genutzt. Zuerst werden die Daten zufällig in zwei Teilmengen, auch *Folds* genannt, aufgeteilt. Einer der Folds wird als Validierungsdatensatz F_v bestimmt. Es erfolgt das Training des Modells mit dem anderen Fold als Trainingsdaten F_t und dem validierenden Testen anhand F_v unter Gebrauch eines Fehlermaßes. Dieser Ablauf der Unterteilung von Training und Testen kann je nach Verfahren mehrfach wiederholt werden, wobei sich die Aufteilung der Daten systematisch ändert. Abschließend wird aus den gesammelten Ergebnissen der Durchschnitt gebildet, der die Endbewertung der Validation darstellt. (J. Shao 1993; Schnaubelt 2019)

Da sich die Teilmengen bei der Kreuzvalidierung beliebig verschieden zusammenstellen lassen, gibt es unterschiedliche Verfahren der beschriebenen allgemeinen Kreuzvalidierung. Es existieren die beiden Kategorien *exhaustive data splitting* und *partial data splitting* in die sich die verschiedenen Verfahren einordnen lassen. Das Erste schließt alle Validierungen ein, die alle möglichen Aufteilungen der Daten für Training und Testen berücksichtigt. So, dass jeder einzelne Datenpunkt einmal zum Testen verwendet wird, während alle anderen zum Training dienen. Beim *partial data splitting* hingegen wird nur eine Teilmenge der möglichen Aufteilungen der Daten für Training und Testen berücksichtigt. Damit wird nur eine bestimmte Anzahl von ausgewählten Aufteilungen verwendet, um die Rechenkomplexität gegenüber Exhaustive Data Splitting zu verringern. (Arlot und Celisse 2010)

Der Sinn solcher Verfahren ergibt sich aus den Nachteilen der manuellen Abspaltung eines Teils der potenziellen Trainingsdaten. Falls nur wenig Daten vorhanden sind und der Testdatensatz für die Validierung dadurch sehr klein ausfällt, können damit nur unzureichend Informationen über die Leistung eines Modells geliefert werden. Außerdem könnte es aufgrund zufälliger Datenverteilungen in den aufgeteilten Datensätzen zu einer inkorrekten Einschätzung

führen. Durch ihre effizientere Nutzung der Daten und eine zuverlässigere Bewertung, über mehr Daten, wird die Kreuzvalidierung oft bevorzugt. (Hastie et al. 2009)

Vorteile und Nachteile von Exhaustive-/Partial Data Splitting

Bei der Ausschöpfung aller Datenpunkte zur Validation, durch Exhaustive Data Splitting, bietet sich der Vorteil einer vollständigen und umfassenden Einschätzung der Modellleistung, was wiederum zu einer zuverlässigen Beurteilung führt.
Allerdings wird so ein Verfahren je nach Größe des Datensatzes auch enorm rechenintensiv. Genau dieser Nachteil kann bei Partial Data Splitting besser kontrolliert werden, indem nur eine Teilmenge der möglichen Kombinationen einbezogen wird. Da diese Anzahl präzise bestimmt werden kann, ist dies der größte Vorteil solcher Verfahren. Jedoch besteht die Möglichkeit bestimmte Muster in den Teilmengen zu vernachlässigen, was eine Verzerrung der Modellbewertung verursachen könnte. (Schnaubelt 2019)

Welche Art von Verfahren am besten geeignet ist, hängt damit von der Größe des Datensatzes, der aufzuwendenden Rechenressourcen und dem gewünschten Grad an Zuverlässigkeit der Modellbewertung ab.

Beispielverfahren

Da die detaillierten Abläufe sich mit der beschriebenen allgemeinen Kreuzvalidierung überschneiden, wird hier nur auf den Vorgang der Datenaufteilung eingegangen und von der anschließenden Auswertung des Fehlermaßdurchschnitts ausgegangen.

Ein prominentes Beispiel für Exhaustive Data Splitting ist das *leave-one-out* Verfahren von (Stone 1974), (Allen 1974) und (S. Geisser 1975). Hier besteht jede Testteilmenge aus genau einem Datenpunkt. Somit wird das Modell bei n Datenpunkten pro Durchgang auf $n - 1$ Datenpunkten trainiert und auf einem Datenpunkt getestet. Über alle Durchgänge wird das Modell dann n mal trainiert und getestet.

Ein weiteres bekanntes Verfahren der Kategorie Exhaustive Data Splitting ist *leave-p-out* oder auch *leave-n_v-out*. (J. Shao 1993) entwickelte es unter anderem, weil leave-one-out oft unnötig große/komplexe Modelle bei der Model Selection wählte und die Wahrscheinlichkeit das

optimale Modell zu wählen sehr gering sein kann. Leave-p-out erweitert das Vorgehen von leave-one-out, indem die Größe der Testteilmengen anhand von p variabel ist, statt auf 1 festgesetzt zu sein. Beim Verfahren wird also jede mögliche Kombination von p Datenpunkten aus den Daten ohne Wiederholung als einzelner Testdatensatz verwendet, während alle anderen zum Training dienen. Damit steigt die Anzahl Folds von n auf $\binom{n}{p}$ an, wodurch der Rechenaufwand stark zunimmt, aber die Wahrscheinlich das optimale Modell zu wählen höher ausfiel. Um den Rechenaufwand eingrenzen zu können, war es üblicher das leave-one-out Verfahren zu nutzten.

Als Vergleich der Anzahl Folds gibt es bei $n = 5$ Elementen mit leave-one-out 5 Aufteilungen gegenüber leave-p-out mit $p = 2$ insgesamt 10. Dies ergibt sich aus $\binom{n}{p}$, durch die Rechnung $\binom{n}{p} = \frac{n!}{p! * (n-p)!} = \frac{5!}{2! * (5-2)!} = \frac{120}{2 * 6} = 10$. Um zu veranschaulichen wie doppelt so viele Folds mit leave-p-out entstehen, wird in Tabelle 2 und Tabelle 3 ein vergleichendes Beispiel für den Ablauf der Datenunterteilung mit den Werten $\{1, 2, 3, 4, 5\}$ aufgezeigt.

Tabelle 2: Beispielablauf von leave-one-out

leave-one-out		
Durch-lauf	Test	Training
1	$\{1\}$	$\{2, 3, 4, 5\}$
2	$\{2\}$	$\{1, 3, 4, 5\}$
3	$\{3\}$	$\{1, 2, 4, 5\}$
4	$\{4\}$	$\{1, 2, 3, 5\}$
5	$\{5\}$	$\{1, 2, 3, 4\}$

Tabelle 3: Beispielablauf von leave-p-out

leave-p-out mit $p = 2$		
Durch-lauf	Test	Training
1	$\{1, 2\}$	$\{3, 4, 5\}$
2	$\{1, 3\}$	$\{2, 4, 5\}$
3	$\{1, 4\}$	$\{2, 3, 5\}$
4	$\{1, 5\}$	$\{2, 3, 4\}$
5	$\{2, 3\}$	$\{1, 4, 5\}$
6	$\{2, 4\}$	$\{1, 3, 5\}$
7	$\{2, 5\}$	$\{1, 3, 4\}$
8	$\{3, 4\}$	$\{1, 2, 5\}$
9	$\{3, 5\}$	$\{1, 2, 4\}$
10	$\{4, 5\}$	$\{1, 2, 3\}$

Eine andere Erweiterung von leave-one-out entwickelte (BURMAN et al. 1994) und stellte sie als *h-block* Validation vor. Diese war speziell für voneinander abhängige Daten gedacht. Da leave-one-out die ursprüngliche Reihenfolge der Datenpunkte beibehält, kann es im Fall von Zusammenhängen in den Daten zu Abhängigkeiten beim Übergang von Trainings- zu Testsatz für einzelne Unterteilungen aller Kombinationen kommen. Die h-block Validation gehört auch zu den Exhaustive Data Splitting Verfahren und nutzt jeden Datenpunkt einzeln zum Testen. Um Abhängigkeiten zwischen Trainings- und Testdaten zu minimieren, werden beim Testen einer Beobachtung die davor und dahinter liegenden Trainingsdatenpunkte jeweils der Anzahl h entfernt. So reduziert sich das Trainingsset allerdings auf $n - 2h - 1$ Datenpunkte.

Um die Merkmale von leave-p-out und h-block zu kombinieren, entwickelte (Racine 2000) das *hv-block* Verfahren. Hierbei wird der Datensatz wieder systematisch durchlaufen. Wenn ein Datenpunkt zum Validieren gewählt wird, wird adaptiv zu leave-p-out nicht nur der eine Datenpunkt getestet, sondern auch die jeweils Nebenliegenden mit in den Testsatz aufgenommen. Wie viele anliegende Datenpunkte wird durch v, analog zu p aus leave-p-out, bestimmt. So beträgt die Größe jedes Testsatzes $2v + 1$. Zusätzlich werden verbindend zur h-block Validation h Datenpunkte vor und hinter dem Testsatz aus dem Trainingssatz entfernt, was daraus resultierend jeweils eine Größe von $n - 2v - 2h - 1$ besitzt.

Im Bereich Partial Data Splitting ist die *k-fold* Kreuzvalidierung von (S. Geisser 1975) ein sehr bekanntes Verfahren und kann als Standardvalidierungsverfahren in den meisten ML Anwendungen betrachtet werden. Zuerst werden die Datenpunkte zufällig gemischt, wonach eine Aufteilung in k gleich große Teilmengen/Folds erfolgt. Jeder Fold ist damit eine Teilmenge bestehend aus n/k zufällig zugeordneten Datenpunkten, bei n gesamten Datenpunkten. Anschließend wird nacheinander jeder Fold zum Testen ausgewählt, während die restlichen $k - 1$ Folds zum Trainieren des Modells eingesetzt werden. Ein in der Literatur üblicher Wert für k ist 10. (Schnaubelt 2019; Cerqueira et al. 2017)

In Abbildung 9 ist dieses Vorgehen verdeutlicht, indem die ursprüngliche Reihenfolge von Datenpunkten als Farbverlauf abgebildet und die zufällige Unterteilung in Folds ersichtlich ist.

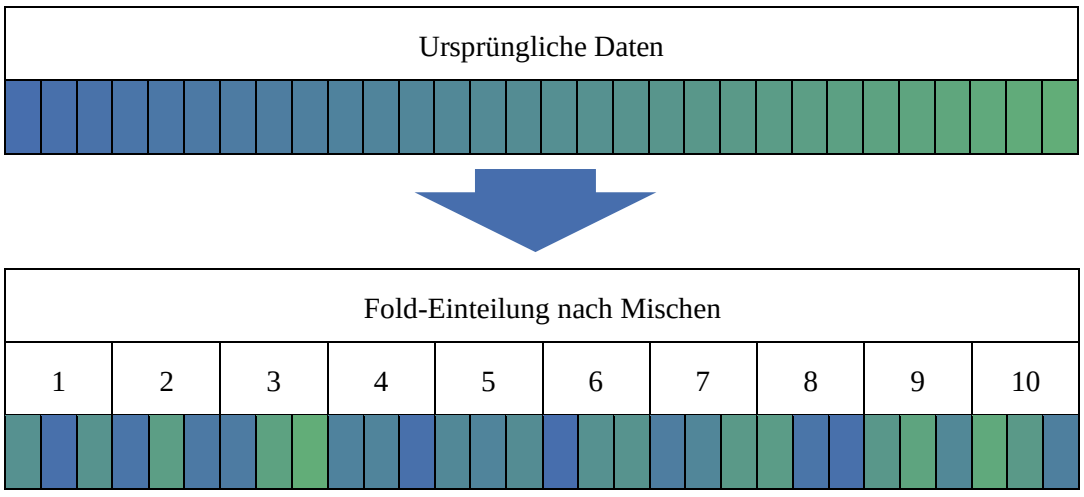


Abbildung 9: Beispiel des Vorgehens bei k-fold Kreuzvalidierung

3 Validierung von Zeitreihen

3.1 Die Problematik mit Kreuzvalidierung

Die normale Kreuzvalidierung kann auch zur Anwendung auf Zeitreihen vorteilhaft erscheinen, weil die verfügbaren Daten effizient genutzt werden und jede Beobachtung zur endgültigen Einschätzung beiträgt. Jedoch limitiert das Merkmal der zeitlichen Ordnung und dessen entsprechende Abhängigkeiten von Datenpunkten die Einsatzfähigkeit sehr. Da bei der Kreuzvalidierung von unabhängig und identisch verteilten Daten ausgegangen wird, erfasst sie die Merkmale von Beobachtungen, die von Vergangenen abhängen können, nicht vollständig. Dies wird durch die zufällige Einordnung der einzelnen Datenpunkte in Trainings- und Testsätze deutlich, was solche Abhängigkeiten zwischen zwei Datenpunkten zum Großteil entfernt. Zum Beispiel würde das Testen der ersten Beobachtung auf dem Training aller Nachfolgenden basieren, wodurch ein Modell die Vergangenheit anhand der Zukunft prognostizieren müsste. Demnach erscheint die normale Kreuzvalidierung für Zeitreihen ungeeignet, weshalb andere Verfahren genutzt werden sollten. (Schnaubelt 2019; Arlot und Celisse 2010)

Weiterhin sollten auch Validierungsverfahren für Zeitreihen Out-Of-Sample Tests verwenden. Wünschenswerte Eigenschaften von Out-Of-Sample Tests für Zeitreihen sind entsprechende Eignung für die Validierung, genügend Prognosen für jede in Frage kommende Vorlaufzeit/Lag und Diversität für die Desensibilisierung der Bewertung für besondere Ereignisse oder sporadische Phasen. (Tashman 2000)

3.2 Abwandlung von Kreuzvalidierungsverfahren

Es gibt die Möglichkeit Verfahren der Kreuzvalidierung, die die zeitliche Ordnung der Daten vollends entfernen, abzuwandeln und sie so zumindest teilweise für Zeitreihen anwendbar zu machen.

Ein Ansatz für die bekannte k-fold Kreuzvalidierung wurde von (Snijders 1988) beschrieben, indem statt der zufälligen Unterteilung aller Datenpunkte, nur nicht unterbrochene Teile der Zeitreihe als Validierungsblöcke verwendet werden. Dies wird erreicht, indem die initiale zufällige Mischung aller Datenpunkte ausgelassen wird. So bestehen alle aufgeteilten Blöcke aus zusammenhängenden Beobachtungen. Dieses Vorgehen ist durch (Bergmeir und Benítez 2012) auch als *blocked cross-validation* bekannt. (Cerqueira et al. 2017)

Auch andere Kreuzvalidierungsverfahren, die die Reihenfolge der Daten nicht manipulieren, können ähnlich wie die blocked cross-validation bedingt für Zeitreihen eingesetzt werden. Aber es bleibt das Problem, dass zum Testen vorheriger Beobachtungen teilweise dessen Nachfolger zum Trainieren verwendet werden. Um auch dem entgegenzuwirken sind andere Methoden erforderlich.

3.3 Forward-Validation Verfahren

3.3.1 Begriffserklärung

Um die zeitliche Abhängigkeit von Beobachtungen angemessen zu berücksichtigen, greift die empirische Forschung in der Regel auf Validierungsverfahren zurück, die die zeitliche Reihenfolge zwischen Trainings- und Validierungssätzen beibehalten. Im Vergleich zur Kreuzvalidierung wird bei solchen Verfahren strikt verlangt, dass die Beobachtungen der Validierungsmenge auf alle Beobachtungen der jeweiligen Trainingsmenge folgen. So gibt es keinerlei Verzerrung der Bewertung durch einen Blick in die Zukunft relativ zur Validierungsmenge. Methoden mit dieser Herangehensweise werden unter anderem *Forward-Validation* Verfahren genannt. (Schnaubelt 2019; Cerqueira et al. 2017)

Durch die Anforderungen der Forward-Validation entstehen aber auch gewisse Nachteile. So werden zum Beispiel, durch die Einschränkung der Trainingsdaten, die verfügbaren Daten

nicht so effizient genutzt, wie bei der Kreuzvalidierung. Des Weiteren entsteht die notwendige Festlegung einer Mindestgröße für Trainingssätze, um die Einteilung einer Zeitreihe, vor allem die ersten Beobachtungen betreffend, zu bestimmen. Diese Mindestgröße sollte zur Vermeidung von Underfitting nicht zu klein sein, aber auch nicht zu groß, wodurch sonst zu viele Trainingseinteilungen verloren gehen. (Schnaubelt 2019)

3.3.2 Holdout und Last-Block-Out

Wenn eine der simpelsten Varianten einer Einteilung in Training und Test betrachtet wird, die die ursprüngliche Reihenfolge beibehält, trifft man auf die einzelne Trennung des Datensatzes in zwei Teile. (L. Devroye 1976) beschreibt so ein Vorgehen als *holdout*, wobei die Trennung an einer beliebigen Position erfolgt und die resultierenden Größen beider Teilmengen jeweils ungleich Null sein müssen. Hier stellt sich die Frage, wo die Trennung am besten erfolgen sollte. Ob sie zufällig gewählt werden sollte oder mit Rücksicht auf die Struktur und den Verlauf der Daten. (Arlot und Celisse 2010)

Da durch die zufällige Wahl einer Position zur Aufteilung ein unausgeglichenes Verhältnis zwischen Trainings- und Testsatz entstehen kann, ist es üblicher einen Teil vom Ende der Zeitreihe zum Testen festzulegen. Dies wird auch *last-block-out* genannt und wird in Abbildung 10 gezeigt. Die grundlegende und ausschlaggebende Entscheidung der genauen Aufteilung der Daten bleibt auch hier nicht aus. Sie ist entscheidet über die Datenmenge, die dem Modell zum Training zur Verfügung steht, sowie wie viele Daten zur Bewertung dienen. (Tashman 2000; Schnaubelt 2019)

Abbildung 10: Beispiel einer Last-Block-Out Datenaufteilung

Durch nur einen Satz für Training und Testen sind diese Verfahren nur unzureichend zur Bewertung von Modellen geeignet. Maßgeblich für so eine Bewertung wäre die Modellgenauigkeit für ausschließlich den letzten Abschnitt der Zeitreihe, was keine sehr zuverlässige Einschätzung der Gesamtleistung wäre. Um ein punktuell schlechtes Ergebnis durch die

Anwendung mehrerer Testzeiträume zu normalisieren, gibt es weitere Forward-Validation Verfahren. (Bergmeir und Benítez 2012; Tashman 2000)

3.3.3 Rolling-Origin

Im Gegensatz zu Validierungen mit einem festem Übergangspunkt von Trainings- zum Testsatz, auch *origin* genannt, gibt es die Möglichkeit einer systematischen iterativen Versetzung dessen. Diese Verfahrensweise ist unter der Bezeichnung *rolling-origin* oder auch *rolling out-of-sample* bekannt und wurde erstmals von (Armstrong und Grohman 1972) beschrieben. (Tashman 2000)

In der ersten Iteration wird der Endpunkt der ersten Trainingsmenge als erste Position des Origin festgelegt $origin = 1$. Die Validierung erfolgt mit dem Trainingssatz $T = \{D_1, D_2, D_3, \dots, D_{origin}\}$ und dem Testsatz $V = \{D_{origin+1}, D_{origin+2}, D_{origin+3}, \dots, D_n\}$, wobei n die Anzahl der Datenpunkte D ist. Anschließend folgt die nächste Iteration womit der Origin um eine Position weiter verschoben wird $origin = origin + 1$ und sich die Zusammensetzung neuer Trainings- Testdatenmengen wiederholt. Die Datensätze von Training und Test erweitern bzw. verringern sich um einen Wert, entsprechend der Verknüpfung zum Origin. Dies wird bis $origin = n - 1$ wiederholt, womit nur noch ein Wert im Testsatz verbleibt. Wenn der Trainingsdatensatz eine gewisse Größe nicht unterschreiten soll, gibt es eine abgewandelte Variante, in der der Origin zu Beginn statt mit 1 mit einer Mindestgröße s initialisiert wird $origin = s$. So entstehen allerdings s weniger Datensatzpaare von Training und Validierung. (Armstrong und Grohman 1972; Tashman 2000; Schnaubelt 2019)

Da die Anzahl an Iterationen bei großen Zeitreihen aber auch einen hohen Rechenaufwand bedeutet, können sie reduziert werden, indem die Verschiebungsweite, wie in Abbildung 11, von Einzelschritten zu blockweise, z.B. im Fall der Verwendung um die Mindestgröße s, erhöht wird. Dagegen kann die hohe Anzahl Iterationen und damit eine intensivere Verarbeitung der Daten bei kleinen Zeitreihen wiederum ein Vorteil sein. (Tashman 2000; Schnaubelt 2019)

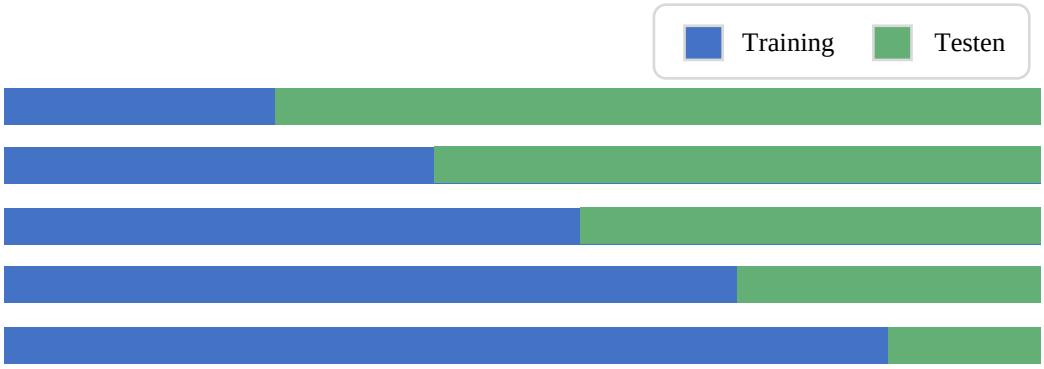


Abbildung 11: Beispiel des Ablaufs von Rolling-Origin Validation mit einer Mindest-trainingsgröße

Bei Einzelschritten entstehen ab der zweiten Iteration mehrfache Prognosewerte für jede Beobachtung, desto weiter hinten sie in der Zeitreihe liegt. Dies hat den Vorteil, dass eine Einschätzung genauer wird und neuere Beobachtungen nahe dem Zeitreihenende am häufigsten einbezogen und damit stärker in der Bewertung gewichtet werden, was als Übergang zu zukünftigen Prognosen am repräsentativsten für künftige Dynamiken sein kann. Andererseits sollen Beobachtungen am Ende der Zeitreihe teilweise mit einem Modell vorhergesagt werden, das nur mit sehr frühen Daten vom Anfang der Aufzeichnungen und auch klein ausfallenden Trainingssätzen trainiert wurde. Dafür entstehen dadurch Prognosen derselben Beobachtung anhand unterschiedlicher Vorlaufzeiten/Lags, so dass die Prognosegenauigkeit auch für jede auftretende Vorlaufzeit beurteilt werden könnte. Beispielsweise wenn für die Vorhersage der 10ten Beobachtung untersucht werden soll, ob die Vorlaufzeit von Beobachtung 1-9, 1-8, 1-7 usw. am genauesten abschneidet. (Tashman 2000; Schnaubelt 2019)

Zum Vergleich mit Holdout und Last-Block-Out verringert Rolling-Origin die Möglichkeit, dass die willkürliche Wahl des Origins die Ergebnisse übermäßig beeinflussen könnte. Dadurch basieren auch andere Validierungen auf dieser rollenden Kernidee. (Fildes 1992; Tashman 2000)

3.3.4 Growing-Window

Ein Verfahren, das auf Rolling-Origin aufbaut, ist die *growing-window* Validation von (Makridakis 1990). Es greift die Problematik der großen Entfernung zwischen den Trainingswerten und den Werten weit hinten im Testsatz auf, indem Letzterer auf eine feste Anzahl von

Beobachtungen beschränkt wird. So entsteht ein Block fester Größe, der die jeweiligen Testdaten definiert. Zeitlich liegt dieser weiterhin nach dem Trainingssatz und verschiebt sich weiter in jeder Iteration anhand des Origin. Die Trainingsdaten hingegen werden wie in Rolling-Origin in jeder Iteration erweitert und wachsen sozusagen an. Dies wird in Abbildung 12 verdeutlicht. (Schnaubelt 2019)

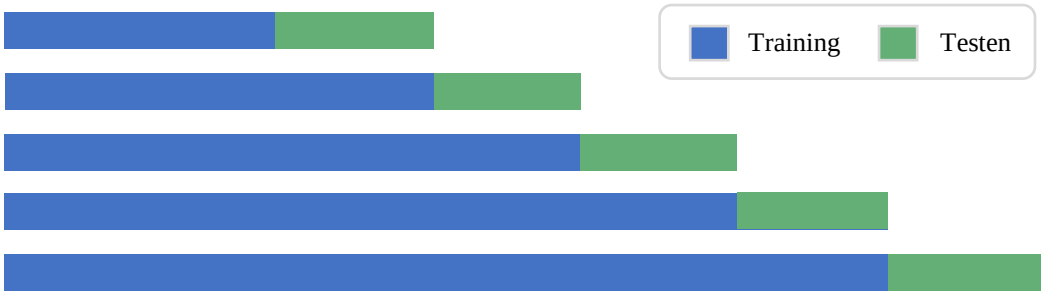


Abbildung 12: Beispiel des Ablaufs von Growing-Window Validation

3.3.5 Rolling-Window

Ein anderes Verfahren, welches auf Rolling-Origin bzw. Growing-Window basiert, ist die *rolling-window* oder auch als *rolling sample* mit fester Größe bekannte Validation, z.B. genutzt von (Callen et al. 1996) und (Swanson und White 1997). Sie schränkt zusätzlich zum Testsatz auch die Größe des Trainingssatzes auf eine feste Anzahl ein, so dass die ältesten Beobachtungen im Trainingssatz bei jeder Iteration entfernt werden, wie in Abbildung 13 zu sehen ist.

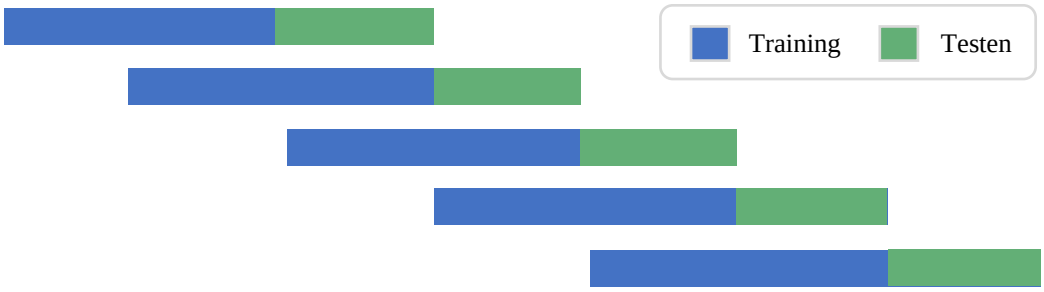


Abbildung 13: Beispiel des Ablaufs von Rolling-Window Validation

Dies hat unter anderem den Grund, dass ältere Beobachtungen vom Training ausgeschlossen werden, um sich nicht zu viel an veralteten Mustern der Vergangenheit zu orientieren. Auch von (Swanson 1998) wurde neben anderen Validierungsverfahren Rolling-Window genutzt, um dem Modell die Anpassungsmöglichkeit an sich verändernde Daten zu geben. Andere

Methoden bewirken dies zum Beispiel alternativ durch eine schwächer werdende Gewichtung, desto weiter die Eingabedaten in der Vergangenheit liegen und so an die Vorlaufzeit/Lag gekoppelt ist. Ein anderer Anwendungsfall wäre die Suche nach der optimalen Vorlaufzeit für ein Modell. Denn falls das Modell keine Einschränkung der Vorlaufzeit hat, kann mit Rolling-Window eine komplette Bewertung anhand einer bestimmten Vorlaufzeit durchgeführt werden. (Tashman 2000)

4 Framework zur Generierung und Bewertung

In diesem Kapitel wird die praktische Umsetzung des Frameworks behandelt, indem auf die Anforderungen, Designentscheidungen und Implementierung eingegangen wird. Um die Ausführungen kompakt und nicht zu ausführlich zu halten, werden vor allem bei der Implementierung nur die wichtigsten Aspekte erläutert und detaillierte Feinheiten ausgelassen.

4.1 Konzept

4.1.1 Verwendungszweck

Als Grundlage für empirische Untersuchungen von Validierungsverfahren für Zeitreihen und speziell für Demand Forecasting wird ein Framework entwickelt, mit dem automatisiert synthetische Zeitreihen generiert und Modelle mithilfe Forward-Validierungsverfahren evaluiert werden können. Dazu teilt sich das Framework in die zwei Bausteine Generator und Bewertungssystem auf.

4.1.2 Generator

Mit dem Generator sollen verschiedene Arten von Zeitreihen erzeugt werden können, deren Eigenschaften wie Länge, Verlauf, Schwankungen, Rauschen oder Ausreißer im Vorfeld festgelegt werden können.

Dazu sollen vorher ein oder mehrere Muster (Patterns) vorgegeben werden, zu dem eine Zeitreihe erstellt wird. So ein Muster besteht aus einer mathematischen Funktion, die als Grundlage der Zeitreihe dient. Jede dieser Funktionen (Pattern-Functions) liefert für einen Zeitpunkt t den Wert einer Beobachtung y auf die jeweilige Berechnungsweise, wie z.B. durch eine quadratische Funktion. Zusätzlich sollen Rauschen, Ausreißer und die Eintrittswahrscheinlichkeit pro Zeitreihe eines Patterns vorgegeben werden können.

Um eine Vielzahl von Kombinationsmöglichkeiten zu ermöglichen, soll die Angabe von mehreren Patterns möglich sein, um zum Beispiel einen saisonalen Verlauf (Pattern 1) mit linearem Anstieg (Pattern 2) zu erzeugen. Um diese Kombination von gewählten Patterns zusammenzufassen, werden sie als ein Demand definiert, was beispielsweise die Beobachtungen von Verkaufszahlen eines Produkts darstellen könnte.

Die eigentliche Generierung einer Zeitreihe soll dann in folgenden Schritten erfolgen:

1. Eine grundlegende Zeitreihe wird als Sequenz von 1 bis zur gewünschten Länge erstellt.
2. Für das Demand wird anhand der enthaltenden Patterns:
    a. Durch die Eintrittswahrscheinlichkeit des Patterns ermittelt, ob es in der Zeitreihe vorkommen wird (weiter mit Schritt b) oder nicht (Schritt 2 mit nächstem Pattern)
    b. Die enthaltene Pattern-Function angewandt: $y = f(t)$, wobei t der aktuelle Index der Zeitreihe ist und y den Wert der erzeugten Beobachtung darstellt.
    c. Rauschen anhand einer parametrisierten Normalverteilung für die Anzahl von Werten der Zeitreihe erzeugt und auf jeden Wert addiert.
    d. Ausreißer durch die parametrisierte Auftrittswahrscheinlichkeit pro Wert der Zeitreihe hinzugefügt. Hierbei dient der Interquartilsabstand der aktuellen Zeitreihe als Ausgangswert, der geknüpft an Wahrscheinlichkeiten negiert und multipliziert wird, um den Wert eines Ausreißers zu bilden.
3. Über alle so erzeugten Zeitreihen der Patterns wird aus den Werten des zugehörigen Zeitpunkts die Summe gebildet, wodurch eine vereinte Zeitreihe bleibt.
4. Eine optionale Umskalierung erfolgt anhand der vorzukommenden Menge des Demands, so dass die Gesamtsumme aller Werte der Zeitreihe die Menge nicht überschreitet.
5. Optional werden abschließend alle negativen Werte der Zeitreihe auf 0 gesetzt, um beispielweise keine negative Anzahl von Demands/Produkten vorkommen zu lassen.

4.1.3 Bewertungssystem

Das Bewertungssystem soll unterschiedliche Forward-Validierungsverfahren auf beliebige Modelle anwenden können, so dass am Ende eine Bewertung entsteht. Das Modell und die Zeitreihendaten liegen dabei außerhalb des Systems und werden zur Verarbeitung übergeben.

Während einer Validierung soll eines der möglichen Validierungsverfahren und dessen Einstellungsmöglichkeiten vorgegeben werden können. Es werden die einzelnen Schritte der Validierung durchgeführt,

Der grundsätzliche Ablauf einer Validierung soll dann in folgenden Schritten erfolgen:

1. Eine Zeitreihe und Modell wird übergeben.
2. Für jedes ausgewählte Validierungsverfahren wird:
    a. Das vorgegeben Validierungsverfahren wird auf das Modell mit der Zeitreihe und vorgegebener Vorhersagweite (Forecasting-Horizon) angewandt.
    b. Die resultierenden Paare von Vorhersagewerten und den zugehörigen korrekten Werten der Beobachtungen werden anhand von Fehlermaßen evaluiert.
3. Es werden alle Fehlermaße pro durchgeführtem Validierungsverfahren als Bewertungsergebnis zurückgegeben.

4.2 Anforderungen

4.2.1 Funktionale Anforderungen

Im Folgenden werden jeweils für den Generator und das Bewertungssystem die erforderlichen Funktionalitäten als User Stories aufgeführt. Dabei wird aus Sicht des Anwenders eine kurze Aktion beschrieben, die durchgeführt werden kann. Der Anwender ist in dem Fall der Benutzer des Frameworks.

Generator

Als Benutzer kann ich…

- Eine Zeitreihe anhand von Vorgaben generieren.

- In der Konfiguration vorgeben können…
    - Wie lang die Zeitreihe ist
    - Wie das Demand heißt
    - Optional eine maximale Menge des Demands für die Zeitreihe angeben, die nicht überschritten wird
    - wie viele Patterns es gibt
        - welche Pattern-Function im jeweiligen Pattern genutzt wird
        - welche Werte für die Parameter der Pattern-Function genutzt werden
        - welche Eintrittswahrscheinlichkeit das Pattern hat
        - welche Varianz das Rauschen des Patterns hat
        - welche Wahrscheinlichkeit für Ausreißer pro Datenpunkt besteht
- Die Konfiguration per YAML-Datei oder in Python direkt als Parameter übergeben
- Mit einem CLI-Skript den Pfad einer YAML-Datei angeben und eine Generierung starten
- Eine optionale Visualisierung der generierten Zeitreihe sehen
- Die erzeugte Zeitreihe losgelöst vom Framework weiter benutzen.
    - Die erzeugte Zeitreihe als Datei speichern lassen.
    - Die erzeugte Zeitreihe als Rückgabewert weiterverwenden.
- Mit einem CLI-Skript eine Liste aller Pattern-Functions mit dessen Parametern im Format HTML, Markdown oder JSON erstellen lassen.

Bewertungssystem

Als Benutzer kann ich…

- In der Konfiguration vorgeben können…
    - welche vordefinierten Forward-Validierungsverfahren für die Validierung genutzt werden.
    - welche jeweiligen möglichen Parameter die Validierungsverfahren nutzen (beispielweise Mindesttrainingsgröße, Window-Size, Anzahl Folds, etc.)
    - welche Vorhersageweite (Forecasting-Horizon) für die Bewertung genutzt wird.

  - Welche vordefinierten Fehlermaße genutzt werden
  - Ob und welche optionale Gewichtung der Fehlermaße anhand Vorhersageweite (Forecasting-Horizon) durchgeführt wird
  - Ob und welches vordefinierte Modell und dessen Parameter zur Bewertung genutzt werden
- Die Konfiguration per YAML-Datei oder in Python direkt als Parameter übergeben
- Mit einem CLI-Skript den Pfad einer YAML-Datei angeben und eine Bewertung starten
- Via Python API ein externes Modell unter Voraussetzungen zur Bewertung verbinden
- Via Python API die Endwerte der Fehlermaße zu jedem Validierungsverfahren als Rückgabe weiterverwenden.

4.2.2 Nicht funktionale Anforderungen

Wartbarkeit:

Um eine Wiederverwendbarkeit und einfache Erweiterungsmöglichkeit nach den empirischen Versuchen dieser Arbeit zu erreichen, stehen eine modulare Architektur mit entkoppelten Komponenten mit genügend Dokumentation innerhalb des Frameworks beim Systemdesign im Vordergrund. Ausgehend davon werden der Generator und das Bewertungssystem jeweils als eigenständige Teile Frameworks innerhalb des Gesamtprojekts entwickelt, um auch einzeln verwendet werden zu können.

Benutzbarkeit:

Zur Verwendung des Frameworks während der Entwicklung wird es als Python Modul importiert und über dessen API angesprochen. Auch bei späterer Nutzung in Zusammenhang mit Python kann es so verwendet werden.

Für die Nutzung abseits einer bestimmten Programmiersprache gibt es Command Line Interface (CLI), über die wiederum die Python API angesteuert werden kann. Zum Beispiel kann das Framework so auch in einen Ablauf von Pipelines integriert werden.

Zuverlässigkeit

Um die Nutzbarkeit ohne Wissen über den Aufbau des Frameworks einfach zu gestalten, werden mögliche auftretende Fehler möglichst innerhalb des Frameworks behandelt und mit einer sprechenden Meldung der vermuteten Ursache nach außen kommuniziert. Zum Beispiel sich wiedersprechende Konfigurationswerte.

Des Weiteren wird durch die Anfälligkeit vieler möglicher Parameter für die Generierung einer Zeitreihe bei dem Generator, durch einzelne Negativ- und Positiv-Unit-Tests, sichergestellt, dass die Grundfunktionalität einsatzfähig ist.

4.3 Entwurf

4.3.1 Generator für synthetische Zeitreihen

Begleitend der folgenden Ausführungen sind in Abbildung 14 die Hauptkomponenten des Entwurfs zu sehen.

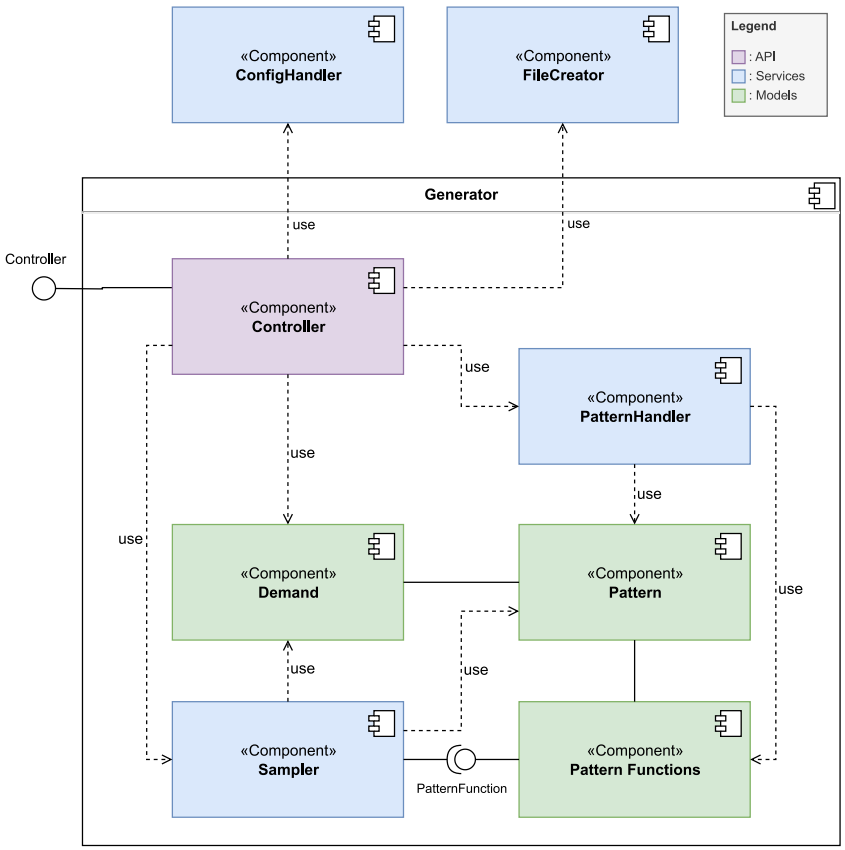


Abbildung 14: Generator Komponentendiagramm

Als Ausgangspunkt gibt es eine Konfiguration, die alle Anpassungsmöglichkeiten der Generierung einer Zeitreihe enthält und als Steuerinstrument von außerhalb dient.

Eine zentrale Kontrolleinheit dient zur Sicherstellung der Austauschbarkeit anschließender Komponenten und steuert den Ablauf und die Koordination zwischen diesen. Auch das Einlesen der Konfiguration und die Zuordnung einzelner Teile davon, zu den korrekten Endpunkten, liegt in dessen Zuständigkeit. Zudem erfüllt sie das Entwurfsmuster einer Fassade, wodurch die möglichen API-Endpunkte für Benutzer zentral aufgeführt sind.

Damit der Generator unterschiedliche Zeitreihen erzeugen kann, gibt es verschiedene vordefinierte Pattern-Functions zwischen denen gewählt werden kann. Durch Vorgabe der Parameter für jede dieser Funktionen, wie den Steigungskoeffizienten bei einer linearen Funktion, können unterschiedlichste Zeitreihenverläufe entstehen. Außerdem halten sich alle vorhandenen Pattern-Functions an eine Schnittstellendefinition (Interface), um schnell und ohne tieferen Eingriff ins System neue gewünschte Funktionen hinzufügen zu können.

Die Konstruktion eines Patterns und dessen Pattern-Function erfolgt über eine eigene Komponente, die auch die Anbindung an die zentrale Kontrolleinheit übernimmt. Angesteuert bündelt sie die Registrierung aller verfügbaren Pattern-Functions und die Zusammensetzung aus den angegebenen Vorgaben.
Um die Bedienbarkeit von außerhalb des Frameworks zu gewährleisten, kann diese Komponente auch eine Liste aller verfügbarer Pattern-Functions mit ihren zugehörigen Parametern und deren Bedeutungen auf Abruf erzeugt werden. So können ohne Kenntnis über den Quellcode die gewünschten Parameter in die Konfiguration eingesetzt werden.

Die konkrete Erzeugung der Zeitreihe wird von einer eigenen Komponente durchgeführt, die anhand des übergebenen Demands mit enthaltenen Patterns nach der in 4.1.2 aufgeführten Vorgehensweise eine neue Zeitreihe aufbaut.

Des Weiteren gibt es zur Förderung der Modularität und unkomplizierter Ersetzbarkeit einzelne Komponenten für:

- Speicherung von Dateien für die generierte Zeitreihe und Pattern-Functions Erklärung
- Verwaltung der Konfiguration als Anbindung der zentralen Kontrolleinheit

4.3.2 Bewertungssystem

Auch hier dient die Darstellung in Abbildung 15 der Hauptkomponenten als Unterstützung zu den Erläuterungen.

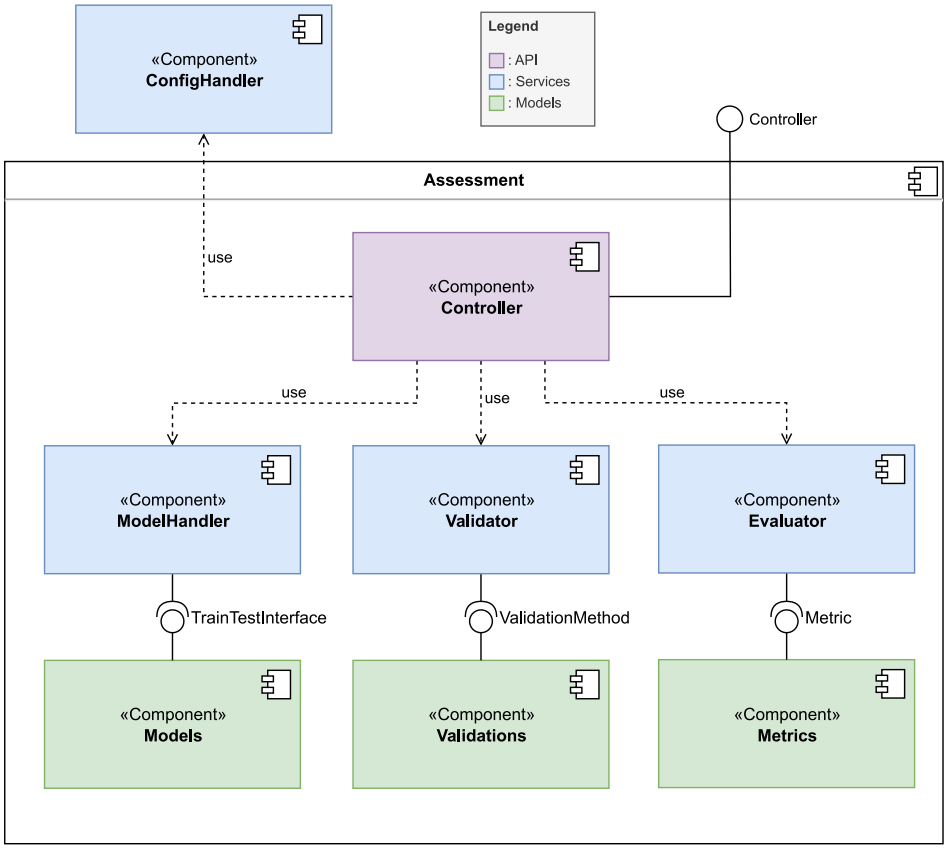


Abbildung 15: Bewertungssystem Komponentendiagramm

Wie beim Generator ist eine eigene Konfiguration der Ausgangspunkt aller änderbaren Einstellungen und Angaben zur konkreten Angabe der durchzuführenden Validierungen.

Auch hier wird die Herangehensweise einer zentralen Kontrolleinheit umgesetzt, die alle weiteren Komponenten ansteuert, den Ablauf koordiniert und als Fassade nach außen dient.

Da zur Modellbewertung ein Modell benötigt wird, gibt es, neben der Auswahl von eingebauten parametrisierbaren Modellen, die Möglichkeit ein externes Modell an das Framework mithilfe eines Interfaces anzuschließen. Die Ansteuerung eines Modells via Interface erfolgt seitens der Kontrolleinheit über einen separaten Manager, der als Ansprechstelle fungiert.

Verschiedene Validierungsverfahren sind über ein gemeinsames Interface implementiert, um das Hinzufügen neuer Verfahren zu erleichtern. Die Nutzung dieses Interface und damit Durchführung eines Validierungsalgorithmus übernimmt eine eigene Steuereinheit, welche sich auch um die Registrierung der verfügbaren Verfahren kümmert. Zur besseren Trennung von Validierungsalgorithmus und Fehlermaß werden die Paare von wahrem Wert und vorhergesagtem Wert der Zeitreihe als Validierungsergebnisse an die Kontrolleinheit zurückgegeben.

Zur Auswertung der Validierungsergebnisse anhand von Verlustfunktionen gibt es mehrere umgesetzte Fehlermaße. Sie halten sich an ein gemeinsames Interface, um auch hier leicht neue hinzufügen zu können. Die Ansteuerung von in der Konfiguration gewählten Fehlermaßen erfolgt über eine weitere eigene Steuereinheit. Diese leitet die Validierungsergebnisse an die jeweiligen Verlustfunktionen weiter, woraus jeweils ein einzelner Wert als Bewertung resultiert.

4.4 Implementierung

4.4.1 Randbedingungen

Eine äußerst beliebte Programmiersprache für Data Science Projekte ist Python. Die leserliche Syntax und umfassende Bibliotheken ermöglichen vielseitige Einsatzmöglichkeiten.
Auch in Bezug zu Machine Learning gibt es von den meisten ML-Bibliotheken eine Möglichkeit zur Nutzung mit Python.

Deshalb wird dieses Projekt in der Programmiersprache Python entwickelt, wobei Visual Studio Code mit mehreren installierten Erweiterungen als Editor dient.

Neben den Standardbibliotheken von Python werden für den Umgang und Visualisierung mit numerischen und tabellarischen Daten Bibliotheken wie numpy, pandas, matplotlib und scikit-learn genutzt. Außerdem wird mit dem Testframework pytest ein Maß an Codequalität mithilfe von Unit-Tests erreicht. Neben diesen hauptsächlich verwendeten Bibliotheken gibt es noch Weitere, die hier nicht weiter aufgeführt sind.

Um für die Weiterentwicklung nach der Umsetzung eine Versionierung des Projektes vorliegen zu haben, wird die Versionsverwaltungssoftware Git während der Entwicklung verwendet.

Zur Einhaltung von Konventionen und Code-Style in Python, werden Flake8 und der Black Formatter zur statischen Code-Analyse (Linten) eingesetzt. Automatische Kontrollen und Einhaltung dieser werden durch Commit-Hooks sichergestellt.

Als Grundlage der Projektstruktur dient ein mit PyScaffold und der für Data Science Projekte zusätzlichen Erweiterung dsproject erzeugtes Projekt.

4.4.2 Generator für synthetische Zeitreihen

Weniger begleitend als mehr der detaillierteren Betrachtung dienend, ist für die Beschreibung der Implementierung das zugehörige Klassendiagramm des Generators in Anhang G.1 zu finden.

Analog zu dem Entwurf gibt es eine Klasse *ConfigHandler*, die die aktuellen Werte der Konfiguration hält und abrufbar macht. Eine neue Konfiguration kann über sie direkt als Dictionary oder per YAML-Datei von außen eingelesen werden.

Als zentrale Kontrolleinheit existiert eine Klasse *Controller*, die alle weiteren Klassen anspricht und den gesamten Ablauf steuert. Sie dient als Bindeglied zwischen allen Komponenten, was diese unter anderem bei Bedarf leicht austauschbar macht. Zusätzlich sind die öffentlichen Hauptfunktionen über ein CLI-Skript ansteuerbar, wodurch sie auch als Fassade (Entwurfsmuster) dient.

Um Pattern-Functions abzubilden gibt es die abstrakte Klasse *PatternFunction* als Interface. Von ihr erbend existieren ein paar solcher Funktionen wie z.B. die *LinearPatternFunction*, für einen linearen Verlauf in der Zeitreihe anhand der Parameter $y = slope * x + y\_intercept$ und die *RandomWalkPatternFunction*, mit der durch Angabe von Mittelwert (dist_mean), Standardabweichung (dist_std_dev) und Startpunkt (start_point) ein eindimensionaler Random Walk hervorgerufen werden kann. Als Beispiel ist ein so generierter Random Walk in Abbildung 16 zu sehen.

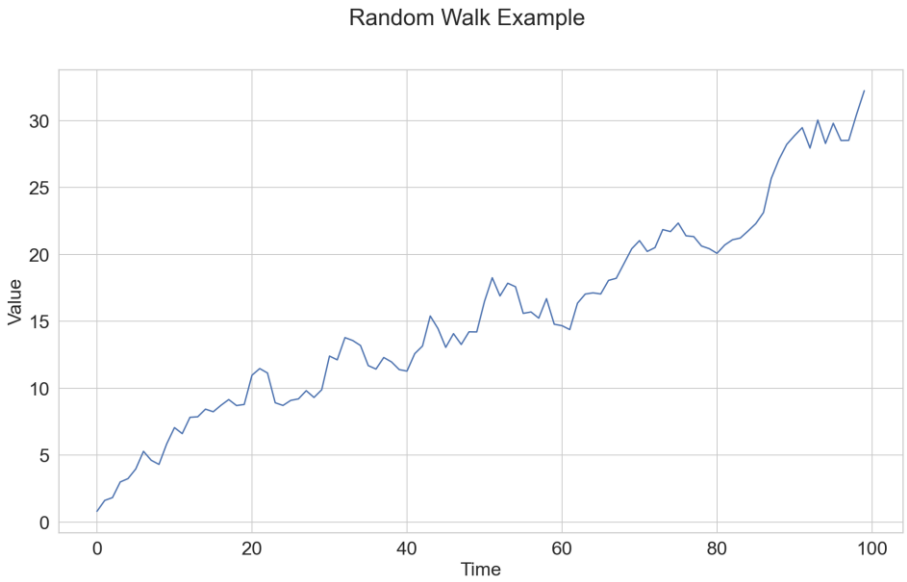


Abbildung 16: Random Walk Beispielverlauf mit $dist\_mean = 0.15$ und $dist\_std\_dev = 1.0$

Eine Liste aller verfügbaren *Pattern-Function*-Klassen ist im Konfigurationsordner hinterlegt, um eine kategorische Unterteilung offen zu lassen.

Eine automatisierte Erstellung einer Beschreibung aller *Pattern-Function*-Klassen inklusive Parameter zur Verständnis nach außen funktioniert über die Verarbeitung der Doc-Strings, die über den *Controller* im Format HTML, JSON oder Markdown als Datei gespeichert werden können.

Die Klasse *Pattern* besteht als Containerklasse aus einer *Pattern-Function*, der Eintrittswahrscheinlichkeit, der Varianz für Rauschen und der Wahrscheinlichkeit von Ausreißern pro Datenpunkt.

Die Verwaltung von *Pattern* und *Pattern-Function*-Klassen übernimmt die Klasse *Pattern-Handler*. Sie registriert alle verfügbaren *Pattern-Function*-Klassen und macht sie durch ihre jeweiligen Identifizierungsnamen von außen erreichbar. So agiert der *Controller* nur mit dieser Klasse und gibt Namen und Parameter weiter, woraus ein *Pattern* erstellt wird.

Die Klasse *Demand* enthält einen Namen, wie z.B. eine Produktbezeichnung und dient neben der optionalen Angabe eines maximalen Kontingents innerhalb der Zeitreihe als Container

einer *Pattern* Liste. Außerdem wird eine optionale spätere Weiterentwicklung in Richtung multivariate Daten mit der möglichen Existenz von mehreren Demands pro Zeitreihe vereinfacht.

Als Recheneinheit gibt es die Klasse *Sampler*, die die konkrete Zeitreihe erzeugt. Sie bekommt vom *Controller* die gewünschte Länge und das zu verwendende *Demand*, was wiederum alle weiteren erforderlichen Einstellungen enthält.

Neben den genannten, gibt es noch weitere sekundäre Klassen, für zum Beispiel die Dateierstellung und eigene Exceptions, welche hier nicht weiter erläutert werden.

Schnittstellen

Die öffentlichen Funktionen vom *Controller* unterteilen sich in API-Funktionen, die mit der Generierung einer Zeitreihe zu tun haben und allgemeinen Utility-Funktionen. Letztere werden intern genutzt und sind nur öffentlich, damit ihre Funktionalität auch anderen Zwecken dienen könnte.

Die Hauptfunktionen sind:

- Konstruktor
  - Initialisiert das System.
- load_generation_config
  - Lädt eine Konfiguration bzw. überschreibt die Aktuelle.
- generate_series
  - Startet die Erzeugung einer neuer Zeitreihe.
- generate_pattern_functions_usage
  - Erzeugt eine Datei mit enthaltender Auflistung aller Pattern-Functions und der Bedeutung ihrer Parameter.

4.4.3 Bewertungssystem

Auch hier ist zur detaillierten Einsicht das Klassendiagramm des Bewertungssystems im Anhang unter G.2 zu finden.

Da wie beim Generator eine Verwaltung der Konfiguration nötig ist, wird die Klasse *Config-Handler* von beiden Systemen in gleicher Funktion genutzt.

Die zentrale Kontrolleinheit übernimmt genauso wie beim Generator eine eigene Klasse *Controller* mit den gleichen Eigenschaften.

Als vordefinierte parametrisierbare Modelle gibt es die Klassen *LinearRegression*, *ARIMA* und *LSTM* (ein RNN), die ihrem Namen entsprechend das jeweilige Modell implementieren, wobei die Klasse *TrainTestInterface* als vorgebendes Interface die erforderlichen Methoden bestimmt. Dieses Interface wird genauso zum Ansteuern externer Modelle genutzt, die es implementiert haben müssen, um verwendet werden zu können. Die Anbindung der Modelle an den *Controller* entsteht durch die Klasse *ModelHandler*, die auftretende Aufrufe weiterleitet.

In ähnlicher Funktion agiert die Klasse *Validator* als Bindeglied zu den verfügbaren Validierungsverfahren. Dabei dient die Klasse *ValidationMethod* als Interface für Validierungsverfahren, um neue Verfahren einheitlich hinzufügen zu können.
Implementierte Validierungsverfahren sind die Last-Block-, Rolling-Origin-, Growing-Window- und Rolling-Window Validation.
Zur besseren Entkopplung werden vom *Controller* nur die Methoden zum Trainieren und Vorhersagen, statt des vollständigen Modellobjekts mit allen zugehörigen Abhängigkeiten, an den *Validator* und damit die Validierungsverfahren übergeben. So erhält ein Validierungsverfahren die Zeitreihe, eine Trainingsfunktion, eine Vorhersagefunktion, die Vorhersagweite bzw. Forecast Horizon und weitere konfigurierende Parameter wie die Mindestgröße für Trainingssätze. Die Rückgabe besteht aus der Auflistung der resultierenden Paare von wahren Werten der Zeitreihe und den zugehörigen produzierten Vorhersagewerten aus dem Validierungsverfahren. Um diese Rückgabe zu vereinheitlichen, gibt es sie anhand der Containerklasse *ValidationReturn*, welche im Interface *ValidationMethod* als fester Rückgabetyp jeder Validierung angegeben ist.

Die schlussendliche Bewertung erfolgt durch ein oder mehrere Fehlermaße, die mit der Klasse *Metric* ein einheitliches Interface implementieren. Implementierte Fehlermaße sind z.B. der *MAE*, *MSE*, *RMSE* und *MAPE*, siehe 2.6.2. Angesprochen werden sie über die Klasse *Evaluator*, der Ergebnisse via *ValidationReturn* vom *Controller* bekommt und an die jeweilige *Metric* übergibt. So entsteht pro konfiguriertem Fehlermaß eine Bewertung, die abschließend vom *Controller* zurück an den Systemaufruf gibt. Optional ist dabei eine konfigurierbare

Gewichtung für die Vorhersageweite, also ob der Fehler vom ersten Vorhersagewert genauso wie z.B. der fünfte in die Berechnung eingeht.

Schnittstellen

Die öffentlichen Funktionen vom *Controller* bieten als API-Funktionen die Ausführung von Validierungsverfahren mit anschließender Auswertung via Verlustfunktionen mit einem beliebigen Modell an.

Die Hauptfunktionen sind:

- Konstruktor
  - Initialisiert das System.
- load_assessment_config
  - Lädt eine Konfiguration bzw. überschreibt die Aktuelle.
- assess
  - Startet in der Konfiguration eingestellte Validierungsverfahren anhand übergebener Zeitreihe und Modell.

5 Experimente

5.1 Untersuchungsziele

Anders als in anderen verwandten Arbeiten, wie bei (Schnaubelt 2019) und (Cerqueira et al. 2017), soll kein Vergleich zwischen normalen Validierungsverfahren und Forward-Validierungsverfahren angestrebt werden, sondern der Fokus auf einem einzelnen Forward-Validierungsverfahren liegen und dessen Parameterauswirkungen. Ausgewählt wurde das Rolling-Window Verfahren, wobei die Auswirkungen von unterschiedlichen Größen des sich verschiebenden Trainingssatzes (Window-Size) in Verbindung mit Saisonalität untersucht werden. Die Annahme dabei ist, dass die Saisonalität durch eine zu kleine Window-Size nicht ausreichend erfasst werden kann und die Fehlerwerte von längeren Window-Sizes niedriger ausfallen sollten.

Um die Vergleichbarkeit zu gewährleisten, werden eine gewisse Anzahl von Zeitreihen pro Vergleich verwendet und jeweils dieselben Zeitreihen für die Durchführung verschiedener Validierungen mit den unterschiedlichen Window-Sizes genutzt. Dabei wird der Verlauf aller Zeitreihen, was Trend und Saisonalität betrifft, ähnlich gehalten.
Zudem wird jede Validierung mit zwei verschiedenen Modellarten durchgeführt, um ein Mindestmaß an Diversität zu erhalten.

Außerdem dient der Versuchsaufbau der beispielhaften Nutzung des entwickelten Frameworks, da dieses zur Erzeugung der Zeitreihen und Durchführung der Validierung genutzt wird.

5.2 Versuchsaufbau

5.2.1 Zeitreihen

Konzept

Als repräsentativer Aufbau einer Zeitreihe für Verkäufe eines saisonalen Produkts, wird ein linearer Anstieg als Trend in Kombination mit einer regelmäßigen saisonalen Schwankung zur Generierung verwendet. Dabei wird die saisonale Periode auf ca. 30 Beobachtungen eingestellt, weil Tests im Vorfeld gezeigt haben, dass z.B. jährliche Zyklen in Kombination mit der nötigen größeren Zeitreihenlänge für manche Modelle schwieriger erfasst werden können und zeitlich sehr aufwendig sind. Ein gleichbleibendes Maß an Rauschen sorgt für geringe Schwankungen, wogegen Ausreißer bewusst ausgeschlossen werden, um zufällige Einflüsse in die Validierung zu vermeiden.
Die Größe der Zeitreihen beträgt 1000 Beobachtungen, damit sie nicht zu klein ist, um genügend Wiederholungen der Saisons zu enthalten. Aber auch nicht zu groß ist, um die Durchführungsdauer von Versuchen akzeptabel zu halten. Aus selbem Grund ist die Anzahl an Zeitreihen für jeden Versuch auf 100 begrenzt.

Mathematischer Aufbau und Parameter

Die Zeitreihen sollen durch den in Kapitel 4 vorgestellten Generator erzeugt werden. Dazu werden eine lineare Funktion für den Trend und eine Sinusfunktion für die Saisonalität verwendet. Deren vorgegebene Funktionsparameter, die an den Generator übergeben werden müssen, setzen sich wie folgt zusammen.

Lineare Funktion: $y = slope * x + y\_intercept$

Sinusfunktion: $y = amplitude * \sin(period * x + x\_shift) + y\_shift$

Um unterschiedliche, aber ähnliche Zeitreihen zu erzeugen, werden die genutzten Werte mancher Parameter über Normalverteilungen abgebildet, aus denen pro Zeitreihe neue Werte gezogen werden. Um dabei keine zu starken Schwankungen zu produzieren, ist die jeweilige

Verteilung jedes Parameters einzeln justiert. Für die übrigen Parameter sind statische Werte vorgesehen.

Da für jeden Durchlauf innerhalb der Versuchspipeline neue Zeitreihen und damit auch neue Parameterwerte verwendet werden, aber die Verteilungen identisch sind, folgt eine Auflistung der jeweiligen Verteilungen oder statischen Werte.

Lineare Funktion

Parameter aus Normalverteilung:

slope mit Erwartungswert von 1,0 und Varianz von 0,2

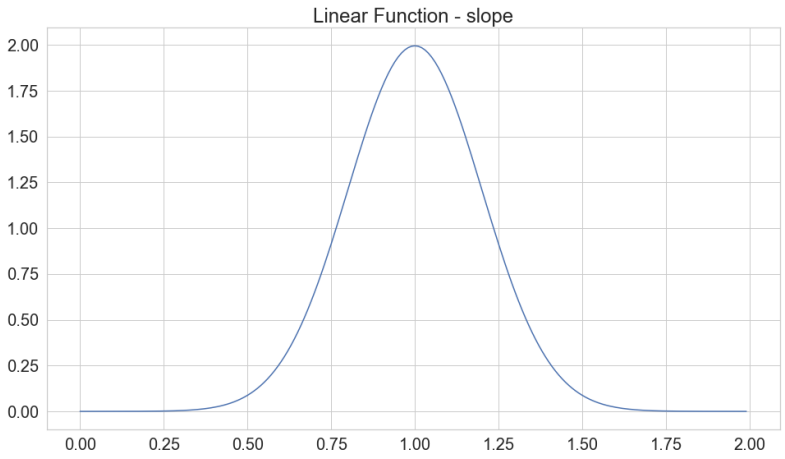


Abbildung 17: Verteilung des Parameters slope der linearen Funktion

Statische Parameter:

y_intercept $= 0$

Sinus Funktion

Parameter aus Normalverteilung:

period mit Erwartungswert von 0,2 und Varianz von 0,0005

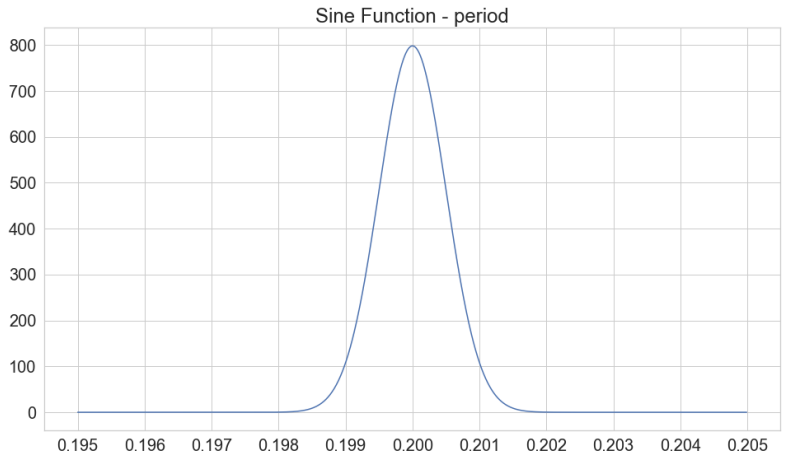


Abbildung 18: Verteilung des Parameters period der Sinus Funktion

Statische Parameter:

$amplitude = 50$
$x\_shift = -7{,}5$
$y\_shift = 50$

Rauschen

Zusätzlich erhält jede der beiden Funktionen pro Zeitreihe ein gleichbleibendes Rauschen. Da das Rauschen selbst auch durch eine Normalverteilung mit Erwartungswert 0 entsteht, kann dessen Varianz als Parameter angegeben werden. Um auch das Rauschen zwischen Zeitreihen etwas zu variieren, wird dafür auch eine Verteilung, statt festem Wert, gewählt und für beide Funktionen verwendet.

Die Varianz des Rauschens wird durch die Normalverteilung mit Erwartungswert 2 und Varianz 0,5 abgebildet.

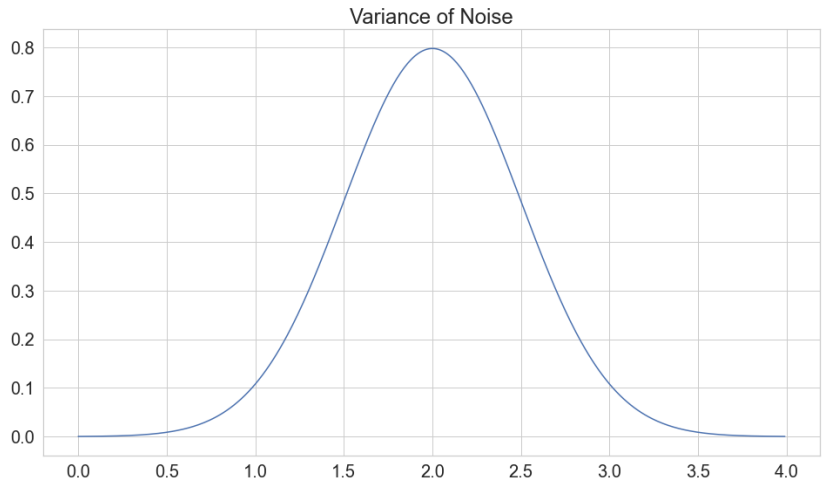


Abbildung 19: Verteilung der Variance des Rauschens

Generierte Zeitreihen

Durch die Neugenerierung und Verarbeitung von Zeitreihen pro Versuchsdurchlauf, können die exakten Zeitreihen, die in den Versuchen verwendet wurden, nicht gezeigt werden. Trotzdem sind in den folgenden Abbildungen beispielhaft generierte Zeitreihen abgebildet, die anhand der oben beschriebenen Parameter erzeugt wurden und für einen Versuch verwendet werden könnten.

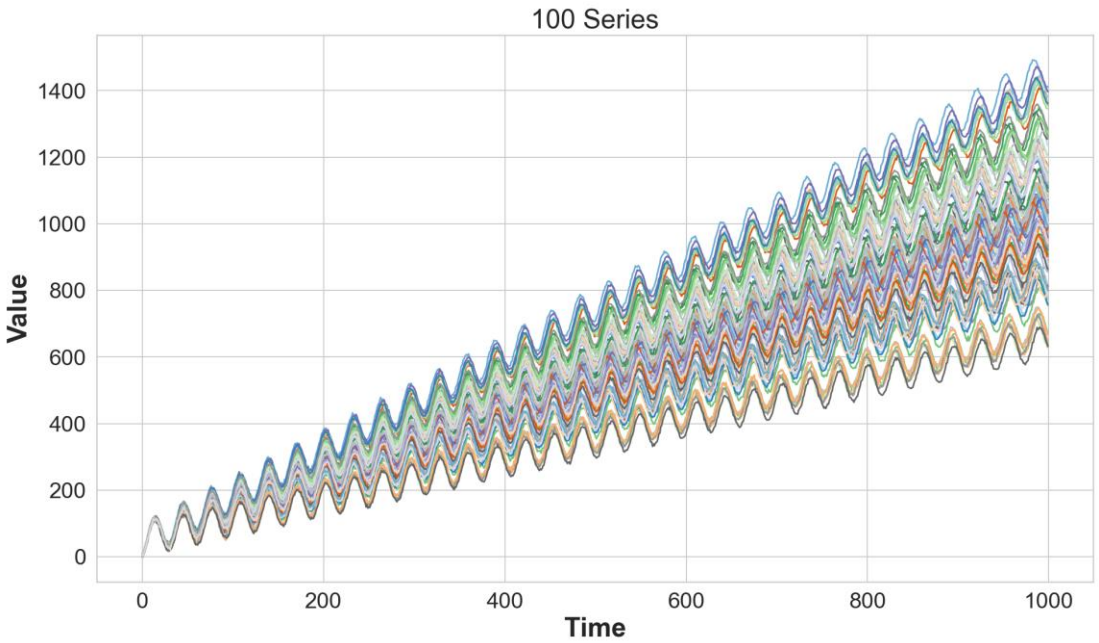


Abbildung 20: Visualisierung 100 exemplarischer generierter Zeitreihen

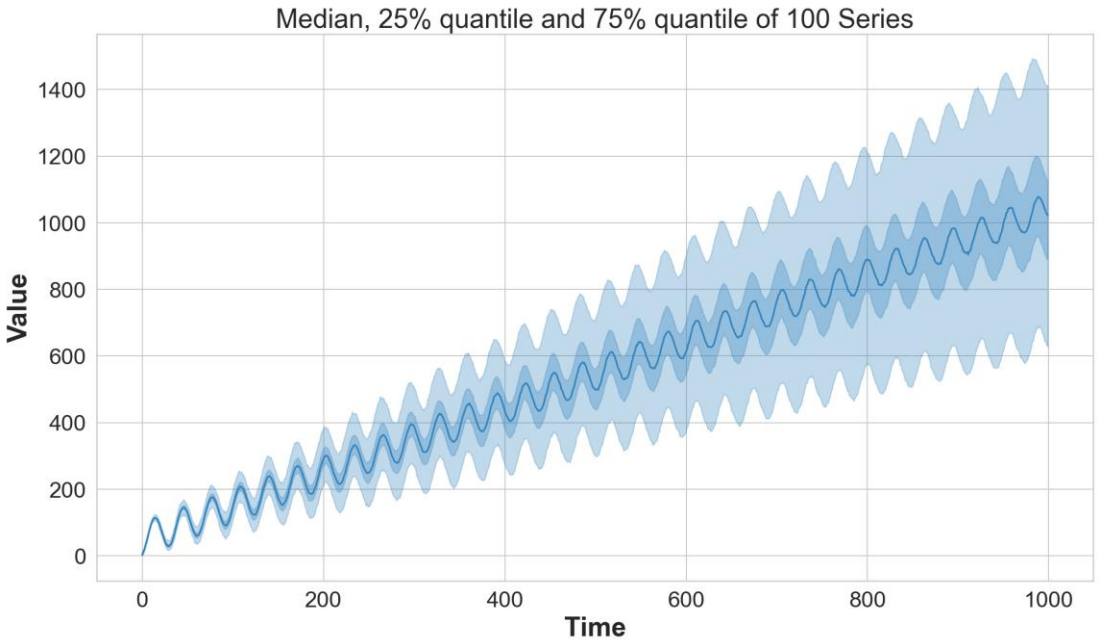


Abbildung 21: Visualisierung der Verteilung der 100 generierten Zeitreihen

5.2.2 Modelle

Auswahl

Bei den eingesetzten Modellen wird sich auf zwei verschiedene beschränkt, wobei Unterschiede der Validierungsergebnisse aufkommen und untersucht werden können, statt nur mit einem Modell.

Als erstes Modell wird das saisonale ARIMA verwendet, weil es grundsätzlich für die Komplexität der Daten geeignet ist, häufig für univariate Daten genutzt wird und oft zum Vergleichen mit ML-Modellen eingesetzt wird, wie z.B. in (Makridakis et al. 2018). (Hewage und Perera 2021; Rob J Hyndman und George Athanasopoulos 2023)

Für das zweite Modell wird sich an dem aktuellen Stand der Technik und erfolgreicher Anwendung im Hinblick auf Zeitreihenprognosen gerichtet. Da die aktuelle M6-Competition noch nicht ausgewertet ist, richtet sich die Auswahl auf die M5-Competition von 2022 (Makridakis et al. 2022). Sie und die M4-Competition zeigten, dass ML-Methoden immer mehr für Prognoseanwendungen eingesetzt werden und immer genauer werden, als statistische Modelle.

Deshalb wird als zweites ein ML-Modell gewählt und weiter nach der M5-Competition gerichtet. Der Sieger des Wettbewerbs nutzte eine Kombination von insgesamt 220 LightGBM Modellen, die zusammen für die endgültige Vorhersage sorgten. Das LightGBM (Light Gradient Boosting Machine) (Ke et al. 2017) ist ein Modell, das speziell für die Technik Gradient-Boosting entwickelt wurde und es in Kombination mit Entscheidungsbäumen nutzt. Dadurch wird nicht ein einzelner Entscheidungsbaum trainiert, der das Modell darstellt und Vorhersagen trifft, sondern iterativ eine Sequenz von schwachen Bäumen erstellt, die jeweils den Fehler des vorherigen korrigieren. So wird das Gesamtmodell schrittweise verbessert und zu einem leistungsstarken Modell kombiniert.
Neben der M5-Competition zeigte LightGBM seine Stärken auch in anderen Wettbewerben und war oft unter den Gewinnern vertreten, wodurch es teilweise sogar als die aktuelle Standardmethode für Prognosewettbewerbe bezeichnet wird. (Hastie et al. 2009)

Auch wenn die Wahl damit auf LightGBM fällt, wird aus Ressourcengründen nur ein Modell für die Versuche eingesetzt werden, statt einer Kombination von wie erwähnt bis zu 220 oder ähnlich.

Modellentwicklung

Um brauchbare Modelle für den Vergleich von Validierungsergebnissen zu erhalten, wurden passende Hyperparameter anhand Model Selection in Kombination zu den Zeitreihen gesucht.

Zur Ermittlung geeigneter Werte für die Parameter p, d und q des saisonalen ARIMA-Modells, gibt es verschiedene Möglichkeiten. p ist dabei dem Autoregressivem, d der Integration und q dem Moving Average zugehörig. Zu diesen kommen die weiteren Parameter P, D, Q und m für den saisonalen Aspekt des Modells hinzu, bei denen es sich um die gleiche Bedeutung nur für den saisonalen Abschnitt handelt und m die Länge einer dieser Abschnitte definiert.

Zum einen kann schrittweise die Box-Jenkins-Methode von (Box et al. 1994) durchgeführt werden, indem die Hyperparameter identifiziert, eine Schätzung der internen Modellparameter anhand von Daten folgt, eine Validierung zur abschließenden Einschätzung der Modellleistung dient und es für Prognosen angewandt werden kann. Die Identifikation kann durch manuelle Auswertung einzelner Tests durchgeführt werden. Dazu wird mit dem erweiterten Dickey-Fuller-Test (Einheitswurzeltest; engl. Augmented Dickey-Fuller-Test) die Stationarität der Zeitreihe festgestellt, um einen Trend zu identifizieren. Je nach Ergebnis wird dadurch die passende Differenzordnung für d bestimmt. Zur Bestimmung von p und q werden auf diesem Weg üblicherweise die Autokorrelationsfunktion und partielle Autokorrelationsfunktion genutzt, anhand dessen die Weite für relevante Korrelationen identifiziert und für p und q eingesetzt werden. (Rob J Hyndman und George Athanasopoulos 2023)

Diese Vorgehensweise wurde für erste kleinere Tests zum besseren Verständnis der Modellart zwar durchgeführt, aber für die relevanten Versuche nicht beibehalten, weil die Voraussetzungen der Model Selection für beide gewählten Modelle ähnlich sein sollten.

Weiterführend wurde ein systematischer Ansatz mithilfe einer Grid-Search, siehe vierten Schritt in 2.5.4, getestet, der sich aber durch lange Laufzeiten als ungeeignet herausgestellt hat.

Schlussendlich wurde das Framework Optuna (Akiba et al. 2019) zur effizienten Suche nach Werten verwendet, in dem eine Zielfunktion (objective) definiert wird, welche Training, Testen und Bewertung ausführt. Die Bewertung erfolgt über ein beliebiges Fehlermaß, wodurch eine Minimierung oder Maximierung vom Framework angestrebt wird.
Die auszuprobierenden Wertebereiche werden im Vorfeld angegeben und vom Framework mithilfe von Tree-Structured Parzen Estimator Schätzverfahren (für weitere Informationen siehe (Ozaki et al. 2022)) und weiterer Techniken effizient eingeschränkt.

Der Vorteil in Bezug zu dem Versuchsaufbau liegt in der modularen Einsatzfähigkeit, da die Zielfunktion selbst definiert wird. So kann für beide Modelle derselbe Ablauf und dieselbe Fehlerfunktion zur Model Selection verwendet werden.

Aufgrund dessen wurde auch für LightGBM die Hyperparametersuche mit Optuna aufgebaut. Die zugehörigen variablen Parameter, nach denen dabei gesucht wird, sind lag wieviele Beobachtungen in die Vergangenheit werden als Eingabe des Modells genutzt, $num\ leaves$ für die maximale Anzahl der Blätter jedes Entscheidungsbaumes, $learning\ rate$ wodurch die Lernrate festgelegt und bestimmt wird, wieviel jeder Entscheidungsbaum zum Gesamtmodell beiträgt und $num\ trees$ für die Anzahl Iterationen, in denen jeweils ein neuer Baum hinzugefügt wird.
Es gibt noch weitere begrenzende Parameter, die hier nicht genutzt werden, um die Möglichkeiten des Algorithmus von LightGBM nicht stark einzuschränken, aber in vielen Fällen sinnvoll sein können.

Angemerkt sei der Unterschied wie die beiden Modelle den Trainings- und Testprozess durchführen. Dem ARIMA-Modell werden die zu trainierende Daten als eine vollständige Sequenz übergeben, um eine Generalisierung über die gesamte Zeitreihe anzustreben. Dagegen muss für das LightGBM-Modell die Sequenz zu einzelnen Paaren von Eingabe x zu Ausgabe y vorbereitet werden. Im Training wird dann eine Generalisierung über alle $y = f(x)$ angestrebt. Hierbei kann x aus einem oder mehreren Werten bestehen, je nachdem wie weit das Modell in die Vergangenheit gucken darf bzw. wie groß die Vorlaufzeit ist.
Durch diesen Unterschied findet auch eine Vorhersage anders statt. Dabei wird dem ARIMA-Modell nur ein Start- und Endpunkt vorgegeben, welche jeweils als numerischer Wert, Datum oder auch eine Zeichenkette definiert sein kann und zusammen einen Zeitraum darstellen. Das

Modell prognostiziert daraufhin die zugehörigen Vorhersagewerte.
Ähnlich zum Trainingsprozess muss das LightGBM-Modell auch zur Prognose vorbereitete Eingaben bekommen. So werden für jede gewünschte Vorhersage y die entsprechenden Eingabedaten x erstellt, die das Modell nacheinander anhand der gelernten Funktion $y = f(x)$ verarbeitet und vorhersagt. Was in der Model Selection noch möglich ist, geht für reale Anwendungen nicht, da für die 10te Vorhersage in die Zukunft vielleicht die Beobachtungen 1 bis 9 als Eingabedaten für das Modell benötigt werden, aber noch gar nicht bekannt sind. In solchen Fällen und so auch in der Validierung der Versuche, wird der erste Wert vorhergesagt, der wiederum an die Eingabedaten angehängt und für die Vorhersage des nächsten Werts genutzt wird.

Um mit diesem Aufbau die Validierung zu starten, wurde eine exemplarische Zeitreihe anhand fester Parameter für den Generator erzeugt. Die verwendeten Parameter entsprachen dabei möglichen Werten der oben aufgeführten Verteilungen. Von der erzeugten Zeitreihe wurde anschließend ein Testsatz von 10% entnommen und vom Rest nochmal 20% zur Validierung reserviert. Die übrigen Daten definieren den Trainingssatz. In der folgenden Abbildung 22 ist diese Aufteilung beispielhaft dargestellt.

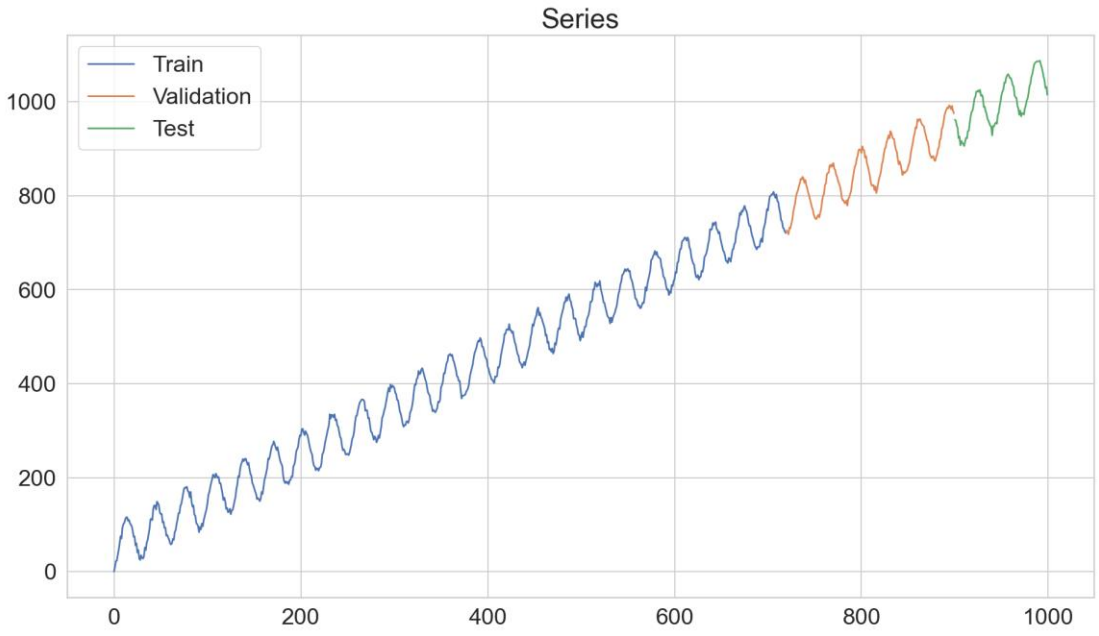


Abbildung 22: Beispielzeitreihe der Modellentwicklung

Der Trainings- und Validierungssatz wird innerhalb der Zielfunktion für Training und Validierung der jeweiligen aktuellen Hyperparameter genutzt. Die Testdaten dienen dem Model Assessment nach der Festlegung geeigneter Hyperparameter.

Der letzte Teil des Aufbaus ist durch das Fehlermaß gegeben. Hierfür soll ein gutes Vergleichsmaß zu den Werten der Zeitreihe und zwischen den Modellen dienen. Eine besondere Robustheit gegen Ausreißer ist nicht nötig, da keine vorgesehen sind. Da es bei einer Zeitreihe bleibt, gibt es auch keine Einschränkungen bezüglich unterschiedlicher Skalen. Trotzdem sollen große Abweichungen stärker bestraft werden, um eine Annäherung zu beschleunigen. Aus diesen Gründen wurde der RMSE gewählt.

Modell Parameter

Saisonales ARIMA

Aus der Model Selection der Modelle ergehen folgende Hyperparameter mit den jeweiligen Fehlerwerten zu den Datensätzen. Außerdem zeigen Abbildung 23 und Abbildung 24 die Prognosen am Ende der Model Selection.

Parameter für saisonales ARIMA: $p = 2, d = 0, q = 2, P = 1, D = 0, Q = 1, m = 30$

RMSE via Validierungsdaten $\approx$ 4,460
RMSE via Testdaten $\approx$ 4,990

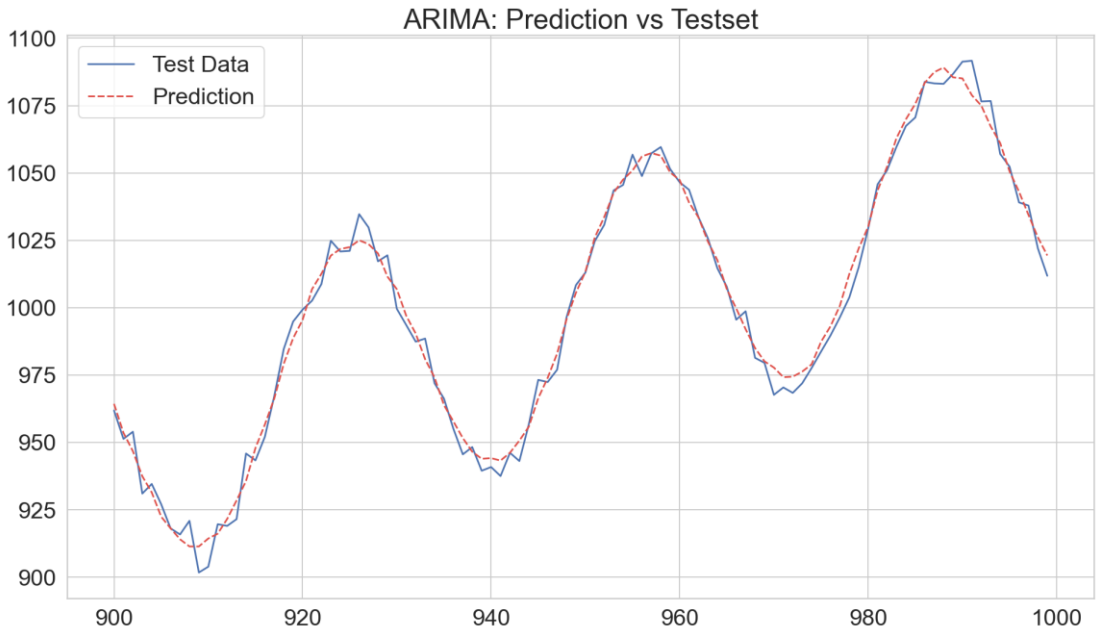


Abbildung 23: ARIMA Vorhersagen für den Testdatensatz der Model Selection

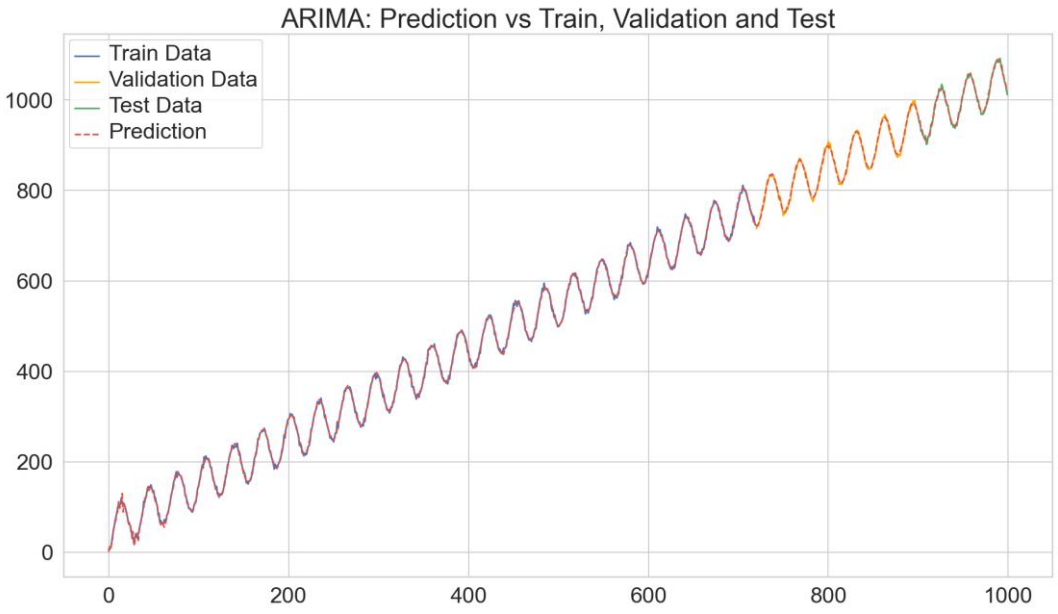


Abbildung 24: ARIMA Vorhersagen für die einzelnen Datensätze der Model Selection

LightGBM

Es folgen die aus der Model Selection hervorgegangenen Hyperparameter mit den zugehörigen Fehlerwerten der getrennten Datensätze, sowie Abbildungen der Prognosen vom Abschluss der Model Selection.

Parameter für LightGBM: $lag = 5, num\ leaves = 144,\ learning\ rate = 0{,}168,$ $num\ trees = 70$

RMSE via Validierungsdaten ≈ 8,184

RMSE via Testdaten ≈ 7,521

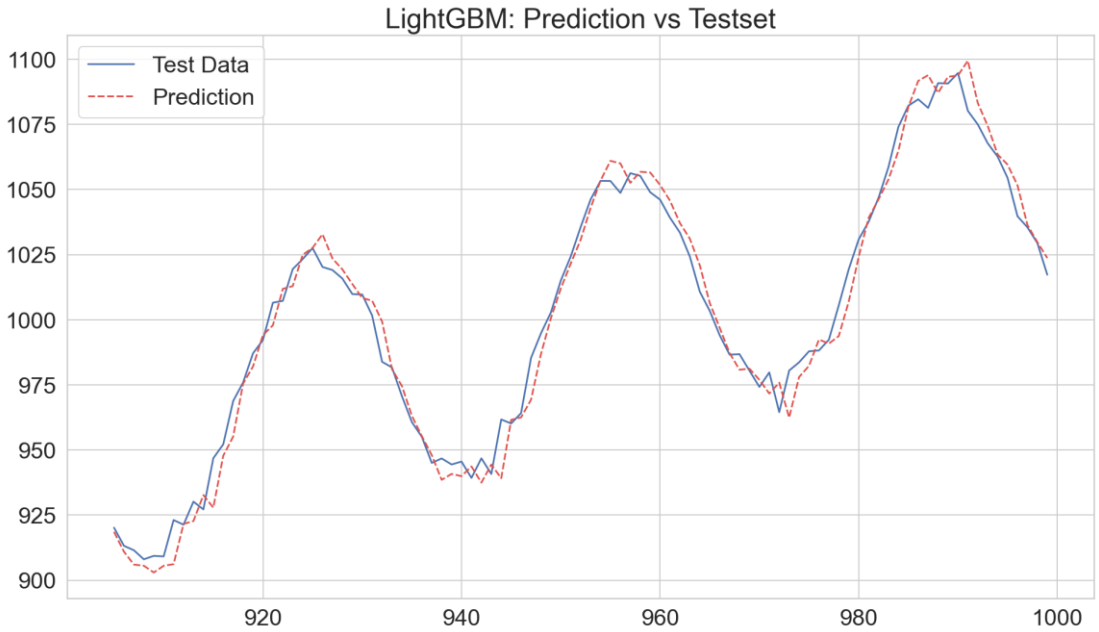


Abbildung 25: LightGBM Vorhersagen für den Testdatensatz der Model Selection

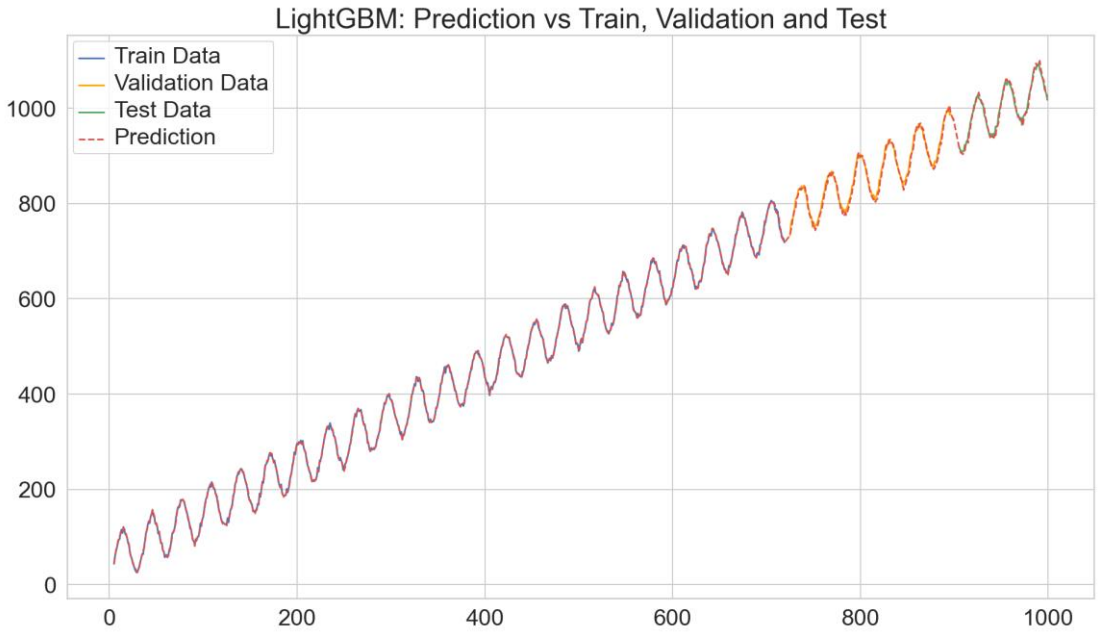


Abbildung 26: LightGBM Vorhersagen für die einzelnen Datensätze der Model Selection

5.2.3 Validierungsverfahren

Window-Sizes

Das gewählte Validierungsverfahren ist, wie in Abschnitt 5.1 erwähnt, das Rolling-Window Verfahren. Um die Auswirkungen von unterschiedlich großen Window-Sizes in Bezug zum saisonalen Verlauf von 30 Beobachtungen zu untersuchen, sind mehrere Größen relativ zur Saisonalität vorgesehen. Davon ausgehend, dass eine zu kleine Window-Size und damit wenig Trainingsdaten dazu führen, dass die Saisonalität nicht ausreichend adaptiert werden kann, gibt es zur Überprüfung dazu Window-Sizes, die kürzer als eine Saison sind. Dazu kommt die normale Länge einer Saison und Verlängerungen, bei denen ein besseres Ergebnis und Unterschiede untereinander erwartet werden, wie zum Beispiel durch verschiedene Anzahlen von Folds durch unterschiedlich große Window-Sizes. Die genauen Größen der verwendeten Window-Sizes sind in Tabelle 4 aufgelistet.

Tabelle 4: Auflistung der Window-Sizes des Versuchsaufbaus

Faktor	Saisonlänge	Window-Size
0,25	30	$7 = \lfloor 0{,}25 * 30 \rfloor$
0,5	30	$15 = \lfloor 0{,}5 * 30 \rfloor$
1	30	$30 = \lfloor 1 * 30 \rfloor$
1,5	30	$45 = \lfloor 1{,}5 * 30 \rfloor$
2	30	$60 = \lfloor 2 * 30 \rfloor$

Forecasting Horizon

Damit zusätzlich zu den unterschiedlichen Window-Sizes noch weitere Untersuchungsoptionen im Verlauf der Versuche produziert werden, wird ein Forecasting Horizon hinzugefügt und gewichtet. Das bedeutet, dass jedes Modell mehr als einen Schritt in die Zukunft vorhersagen wird. Als fester Wert wurde ein Forecasting Horizon von 7 Beobachtungen gewählt, um analog zu täglichen Produktverkäufen jeden Tag der folgenden Woche zu prognostizieren.
Durch die zusätzliche Gewichtung entstehen Validierungsergebnisse für bestimmte Kombinationen der einzelnen Tage. So gibt es eine Validierung, bei der nur die erste Vorhersage voll gewichtet wird, eine Validierung bei der nur die Zweite vorhanden ist usw. und es gibt Validierungen bei denen von 1 bis 2 enthalten sind, von 1 bis 3 usw.

5.3 Versuche

5.3.1 Versuch 1

Methodik

Der erste Versuch wurde anhand des in Abschnitt 5.2 erläuterten Versuchsaufbau durchgeführt. Da für einen ausreichenden Stichprobenumfang eine gewissen Anzahl von Zeitreihen notwendig ist, um eine aussagefähige Auswertung durchführen zu können, aber der zeitliche Aufwand

der Validierungen in jedem Versuch eingeschränkt werden soll, wurde die Anzahl an Zeitreihen des Versuchs auf 100 festgelegt.

Für die Ausführung werden zuerst die Zeitreihen anhand der Parametrisierung generiert. Anschließend erfolgt die Definition der, außerhalb des Bewertungsframeworks genutzten, Modelle und dessen Angabe der Hyperparameter. Danach werden die durchzuführenden Window-Sizes und Gewichtungen aller Forecasting-Horizons für die verschiedenen Konfigurationen der Bewertung definiert. Nach den anschließend ausgeführten Validierungen aller Zeitreihen mit allen Modellen für jede Window-Size Konstellation und der Berechnung eingestellter Fehlermaße, liegen pro Zeitreihe für jedes Modell die jeweiligen Fehlerwerte vor.

Da der RMSE für die Model Selection und Untersuchung gewählt wurde, werden nur dessen Ergebnisse betrachtet.

Ergebnisse

Um die Unterschiede zwischen den verschiedenen Window-Sizes in den Fokus zu rücken, werden in der Auswertung mehrere Visualisierungen genutzt, die die Verteilungen der Fehlerwerte anhand jeder Window-Size darstellen.
Des Weiteren werden der Übersicht wegen von den Gewichtungen des Forecasting-Horizon nur die betrachtet, bei der nur die erste Vorhersage voll gewichtet und der Rest ausgeschlossen wird. Ausgenommen ist die letzte Darstellung des Variationskoeffizienten, die alle Gewichtungen enthält, um einen Eindruck der Schwankungen aufzuzeigen.

Die Visualisierungen der Resultate erfolgen zur besseren Vergleichbarkeit pro Diagrammart paarweise.

Histogramm

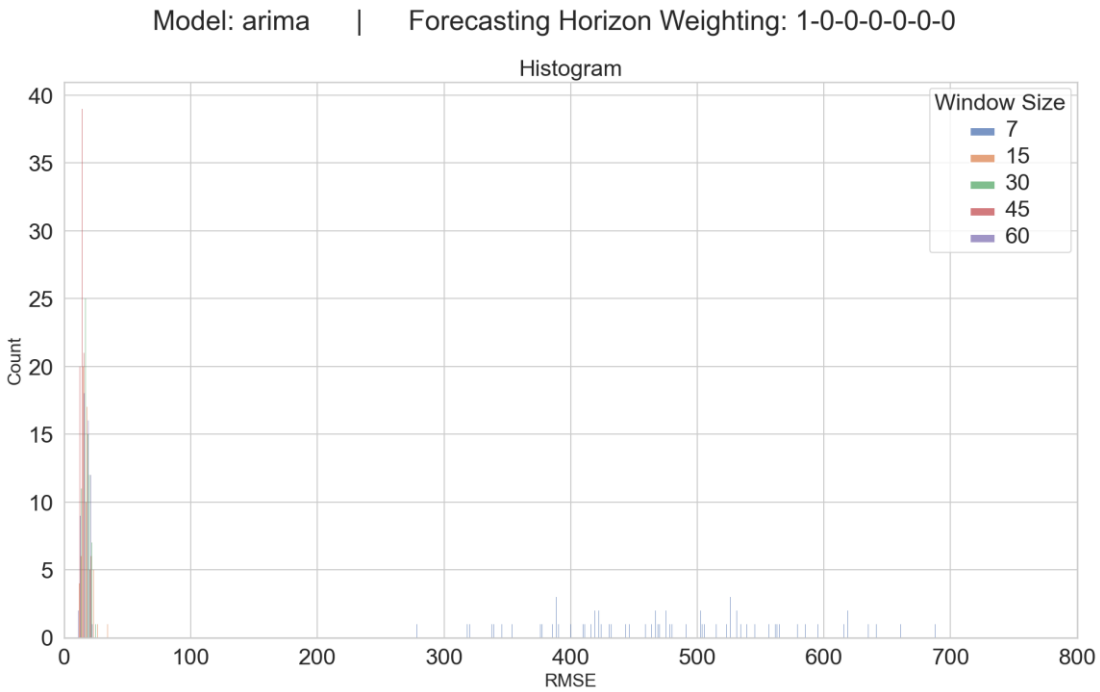


Abbildung 27: Versuch 1 | ARIMA | Histogramm

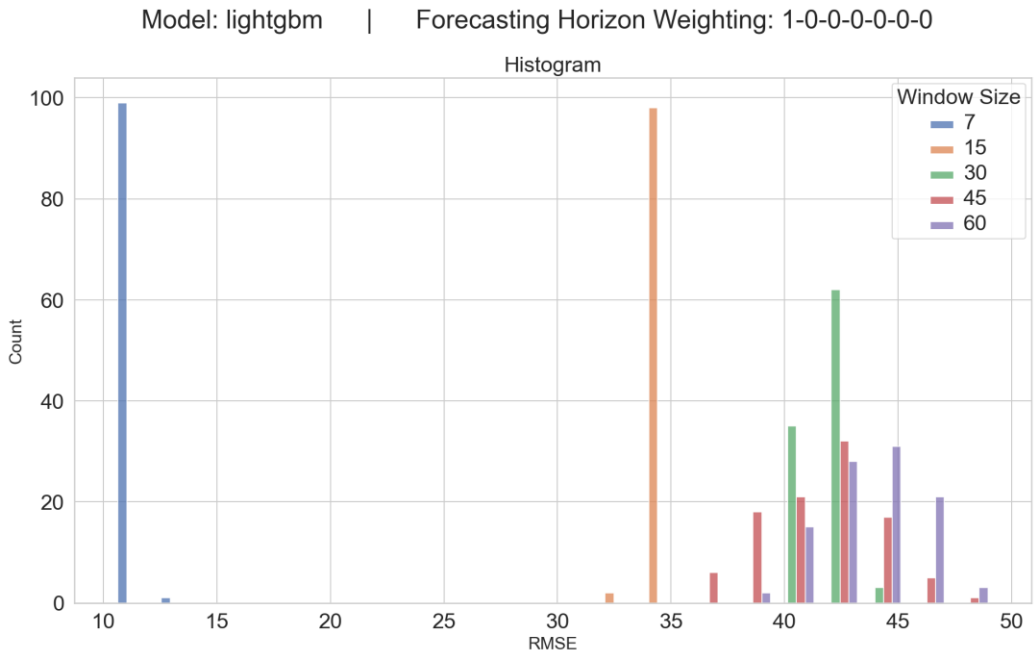


Abbildung 28: Versuch 1 | LightGBM | Histogramm

Boxplot / Kastengrafik

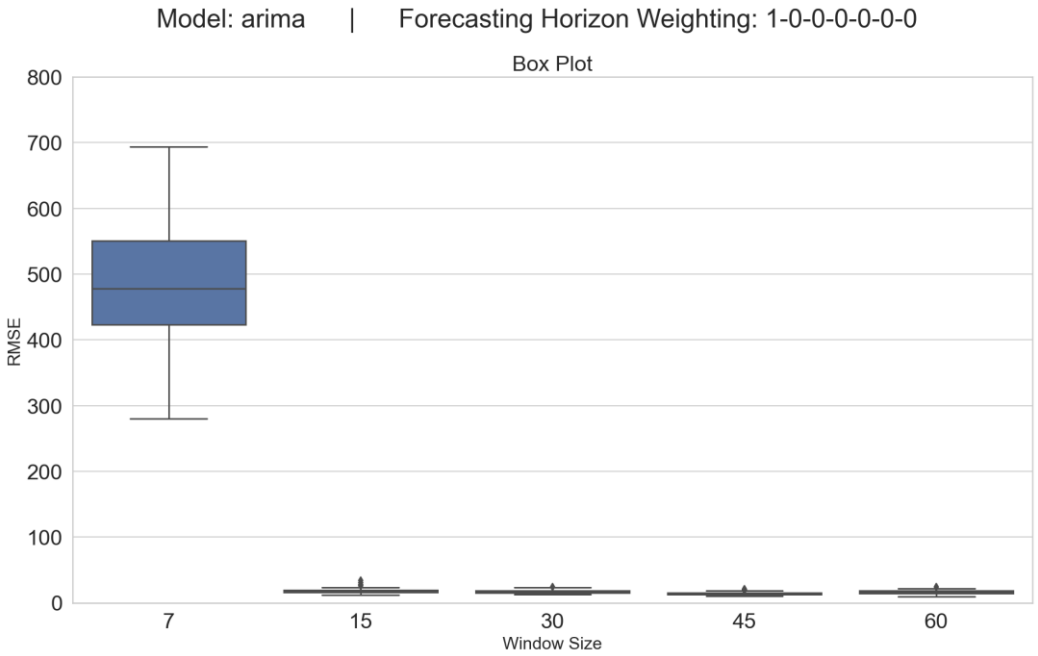


Abbildung 29: Versuch 1 | ARIMA | Boxplot

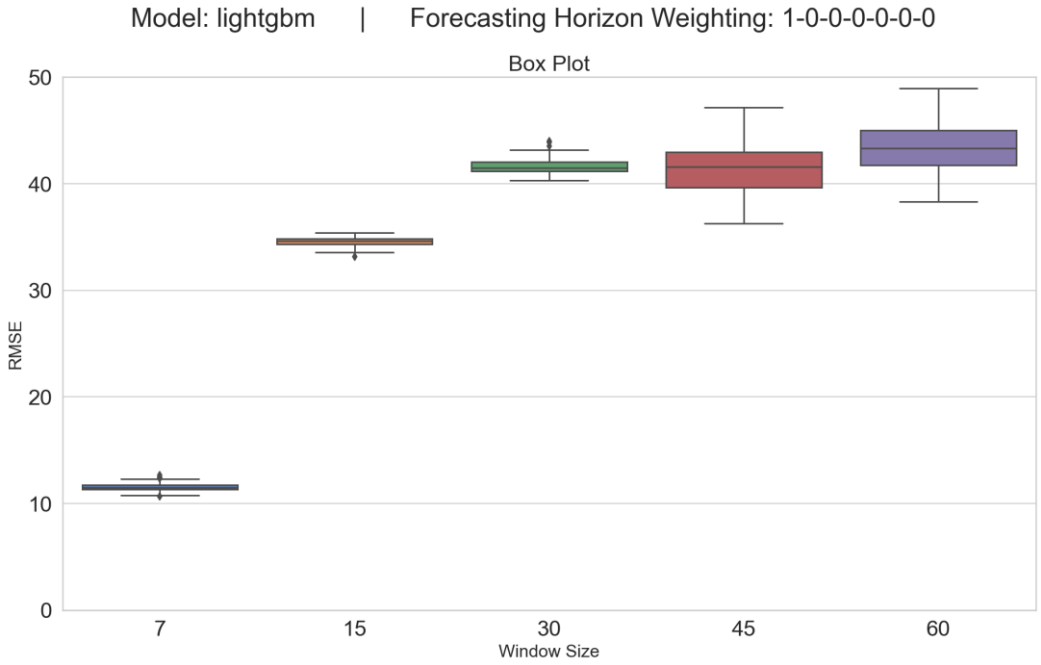


Abbildung 30: Versuch 1 | LightGBM | Boxplot

Kerndichteschätzung

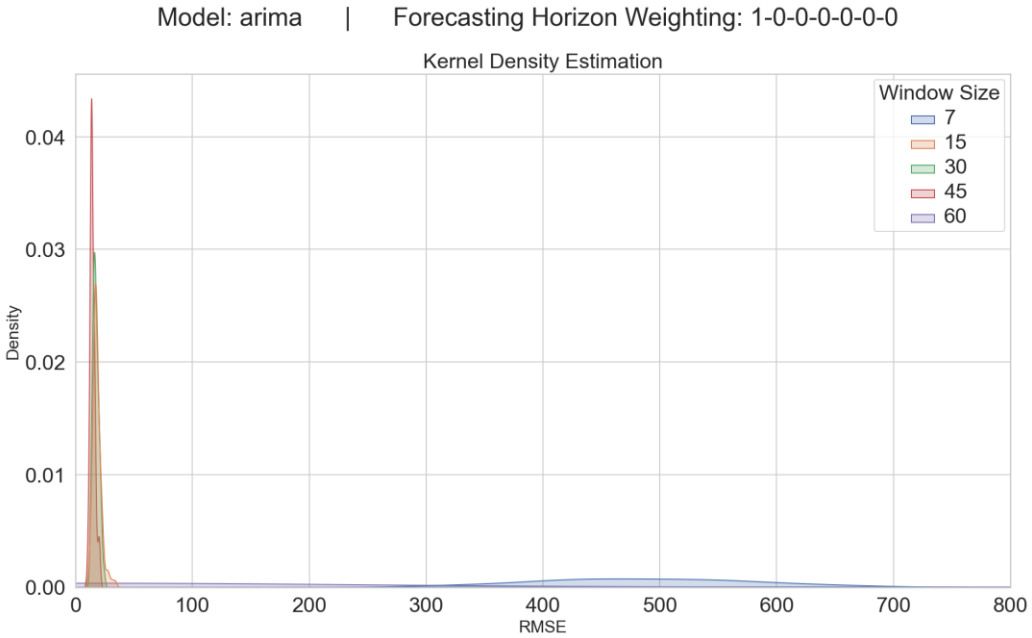


Abbildung 31: Versuch 1 | ARIMA | KDE

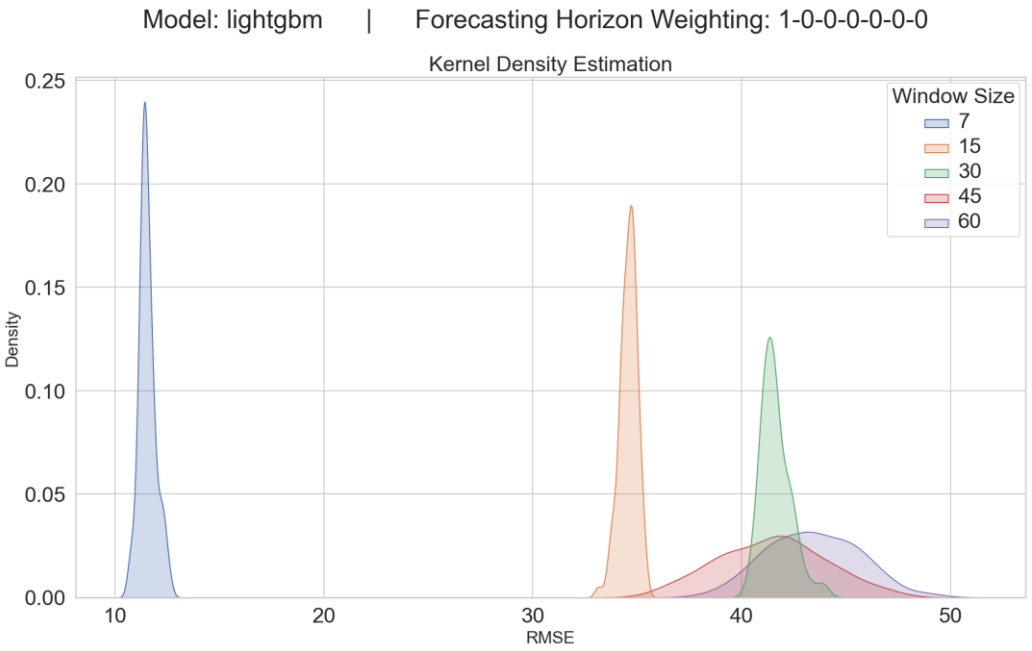


Abbildung 32: Versuch 1 | LightGBM | KDE

Empirische Verteilungsfunktion (ECD)

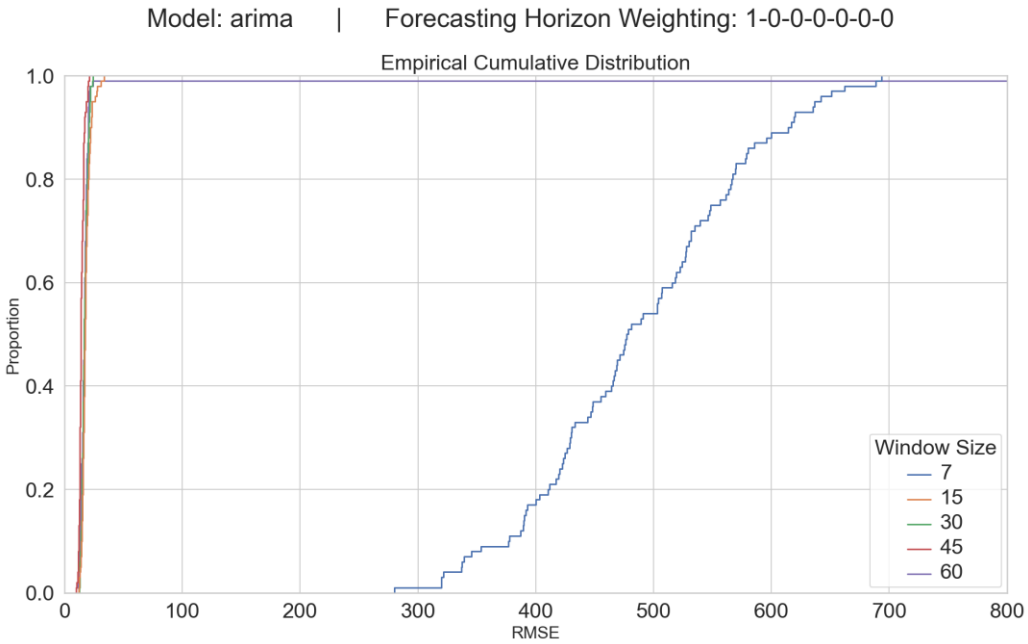


Abbildung 33: Versuch 1 | ARIMA | ECD

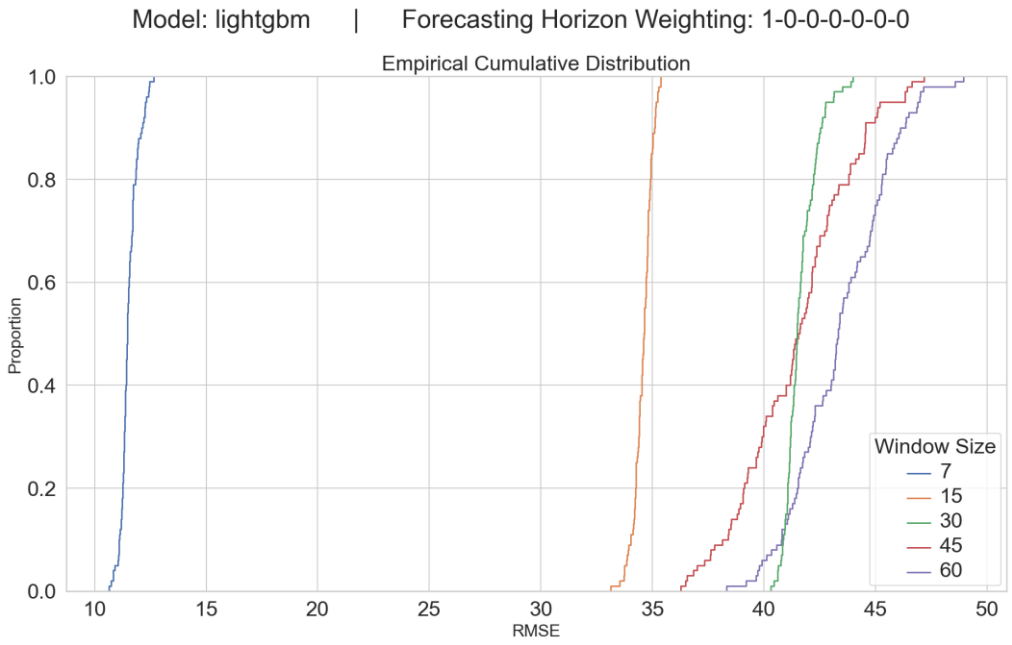


Abbildung 34: Versuch 1 | LightGBM | ECD

Variationskoeffizient / Abweichungskoeffizient

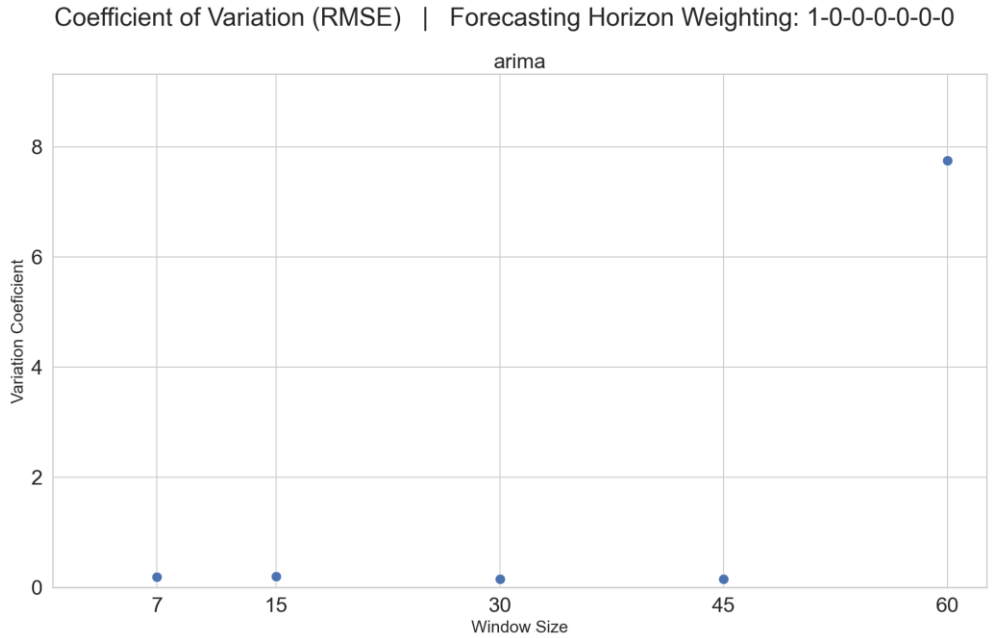


Abbildung 35: Versuch 1 | ARIMA | Variationskoeffizienten

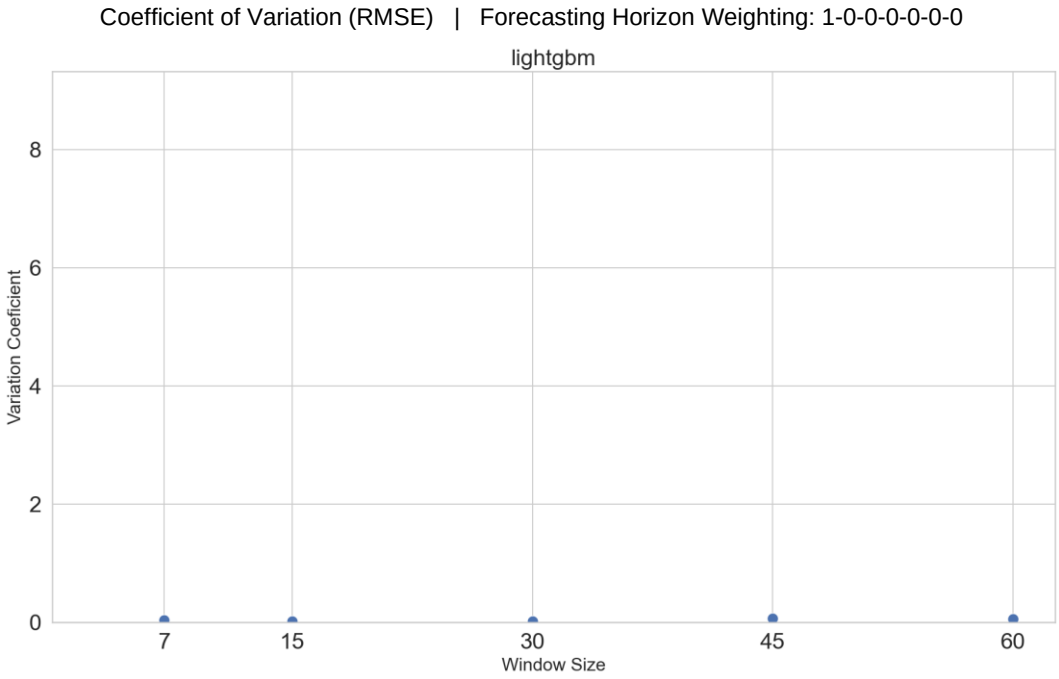


Abbildung 36: Versuch 1 | LightGBM | Variationskoeffizienten

Variationskoeffizient für alle Forecasting-Horizon Gewichtungen

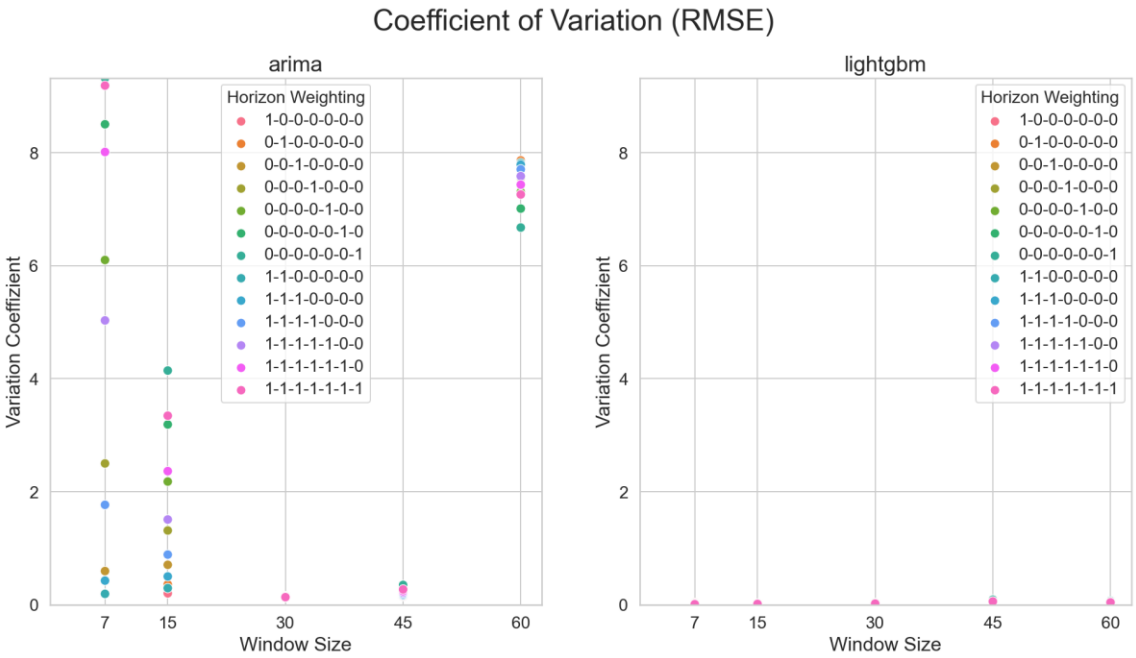


Abbildung 37: Versuch 1 | alle Variationskoeffizienten

Erläuterung

Wie am Box Plot von Abbildung 29 und Abbildung 30 und ECD des ARIMA-Modells in Abbildung 33 deutlich zu erkennen ist, bewirkt die kleinste Window-Size von 7 den im Vergleich zu den anderen Window-Sizes größten Fehlerwert für die Validierung. Die Vermutung und Erwartung war, dass der Trainingsdatenabschnitt zu klein ausfällt und dadurch die Saisonalität nicht gut erfasst werden kann, im Gegensatz zu den größeren Window-Sizes, die alle relativ ähnlich und mit kleinerem Fehlerwert ausgefallen sind.

Fast schon gegenteilig sieht es beim LightGBM-Modell aus. In allen Diagrammen ist zu sehen, dass die kürzeste Window-Size sehr viel besser abzuschneiden scheint als die Restlichen. Dabei liegen die anderen im Fehlerbereich zwischen 30 und 50 gegenüber der Kürzesten mit unter 15. Wird die kleinste Window-Size vom Vergleich ausgeschlossen, scheint es allgemein so zu sein, dass der Fehlerwert höher wird, desto größer die Window-Size. Zum Beispiel wenn Window-Size 15 mit 60 verglichen wird, schneidet die kleinere mit einem niedrigeren Fehlerwert und deutlich weniger Schwankung, siehe Variationskoeffizienten, ab.

Die erhöhte Schwankung bei längerer Window-Size könnte von der geringeren Anzahl Folds verursacht werden, da bei einer Window-Size von 60 in einer Zeitreihe der Länge 1000 nur $16{,}667 \approx \frac{1000}{60}$ Trainings- und Testsätze pro Zeitreihe entstehen. Bei stärker abweichenden Fehlerwerten führt dies zu instabileren Endresultaten. Wogegen die Ergebnisse von $142{,}968 \approx \frac{1000}{7}$ Folds zu einem robusten Endwert führen.

Obwohl, oder eher weil, diese Resultate unerwartet sind, wird im nächsten Versuch probiert dieses Verhalten besser einzuschätzen und Ursachen dafür zu erkennen.

5.3.2 Versuch 2

Methodik

Ausgehend von Versuch 1 soll untersucht werden, warum bei dem LightGBM-Modell kürzere Window-Sizes kleinere Fehlerwerte bei der Validierung produzieren als längere. Ein Gedanke war, dass durch zu wenig Daten unerwartete Entwicklungen stattfinden könnten, weil LightGBM als ML-Modell üblicherweise mit mehr Daten arbeitet. Deshalb wird als nächstes ermittelt, ob eine Datenerhöhung andere Resultate ergibt, indem die Zeitreihenlänge und alle Window-Sizes um den Faktor 10 erhöht werden.

Da das ARIMA-Modell in diesem Versuch kein Untersuchungsobjekt ist und weil es für die angegebene Länge von Zeitreihen einen enormen Zeitbedarf benötigt, wird dieser Versuch nur anhand des LightGBM-Modells durchgeführt.

Ergebnisse

Für die Resultate wurden wieder die gleichen Verteilungsinformationen wie in Versuch 1 herangezogen. Um die Visualisierungen ab hier in Grenzen zu halten, werden nur zur Herleitung benötigte Grafiken aufgeführt.

Boxplot / Kastengrafik

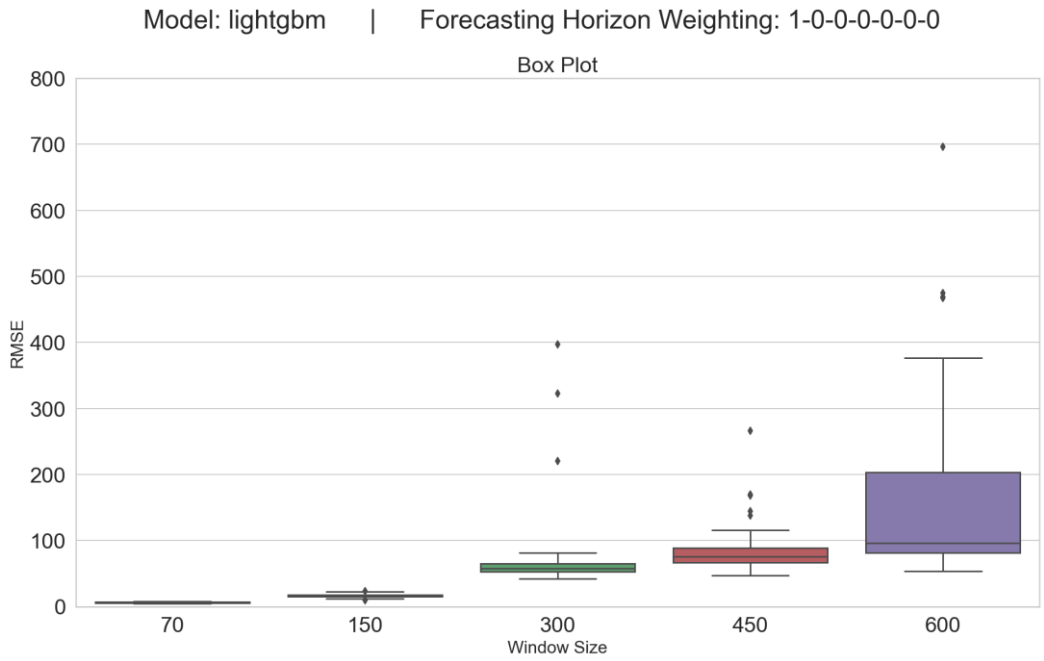


Abbildung 38: Versuch 2 | LightGBM | Boxplot

Empirische Verteilungsfunktion (ECD)

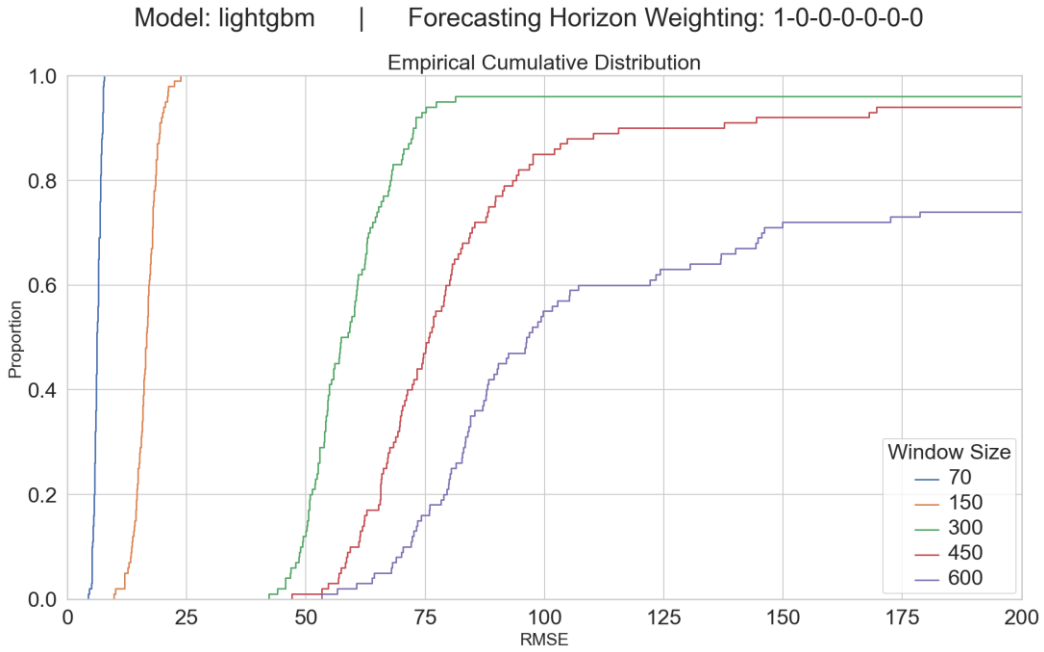


Abbildung 39: Versuch 2 | LightGBM | ECD

Variationskoeffizient / Abweichungskoeffizient

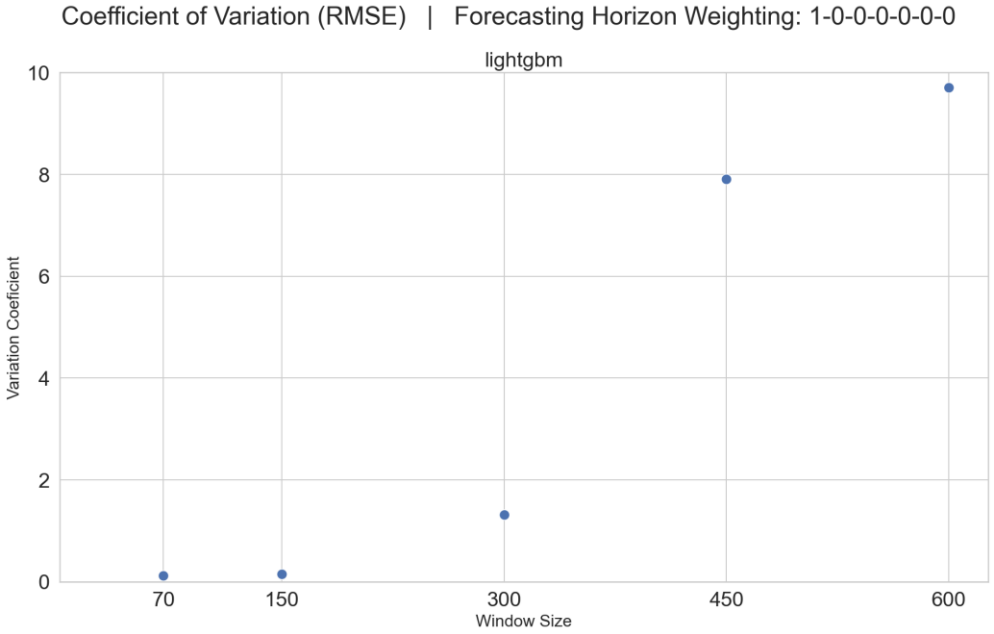


Abbildung 40: Versuch 2 | LightGBM | Variationskoeffizienten

Erläuterung

Obwohl die Zeitreihen jeweils eine Länge von 10.000 Beobachtungen aufweisen und auch die Window-Sizes erheblich erhöht wurden, scheint es die Aufstellung, dass kleinere Window-Sizes einen geringeren Fehlerwert produzieren, nicht davon beeinflusst zu sein. Die kleinste Window-Size ist weiterhin genauer als alle anderen, wobei die Zweitkleinste sich aus dem Fehlerbereich 30 bis 50 und damit von den längeren Window-Sizes entfernt und verbessert hat.

Auf jeden Fall waren die Resultate in Versuch 1 kein zufälliges Ergebnis von zu wenig Daten für ein ML-Modell.

Obwohl die Anzahl Folds jeder Window-Size analog zu Versuch 1 gleichgeblieben sind, entstehen bei größerer Window-Size in diesem Versuch deutlich mehr Schwankungen. Um zu prüfen, ob dies Auswirkungen von zu wenig Validierungen ist, wird ein weiterer Versuch durchgeführt.

5.3.3 Versuch 3

Methodik

Wie aus Versuch 2 hervorging, scheint die Schwankung des Fehlerwerts zu steigen, desto länger eine Window-Size ist. Um zu prüfen, ob sich dies durch eine höhere Anzahl von Validierungen verändert, werden die Längen von Zeitreihe und Window-Sizes beibehalten, aber die Anzahl von Zeitreihen um den Faktor 10 erhöht.

Ergebnisse

Boxplot / Kastengrafik

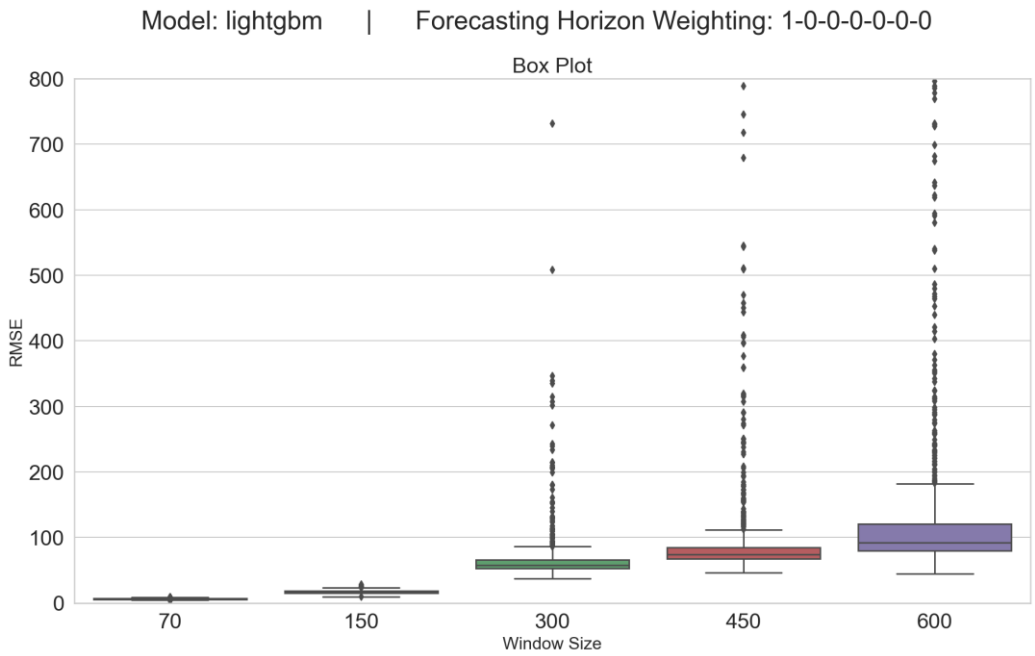


Abbildung 41: Versuch 3 | LightGBM | Boxplot

Empirische Verteilungsfunktion (ECD)

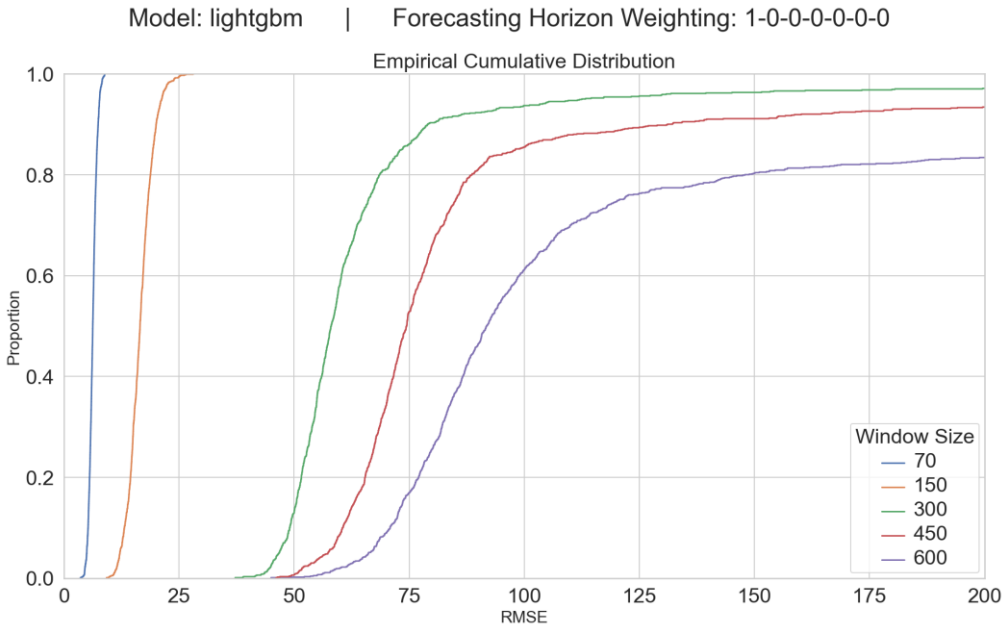


Abbildung 42: Versuch 3 | LightGBM | ECD

Variationskoeffizient / Abweichungskoeffizient

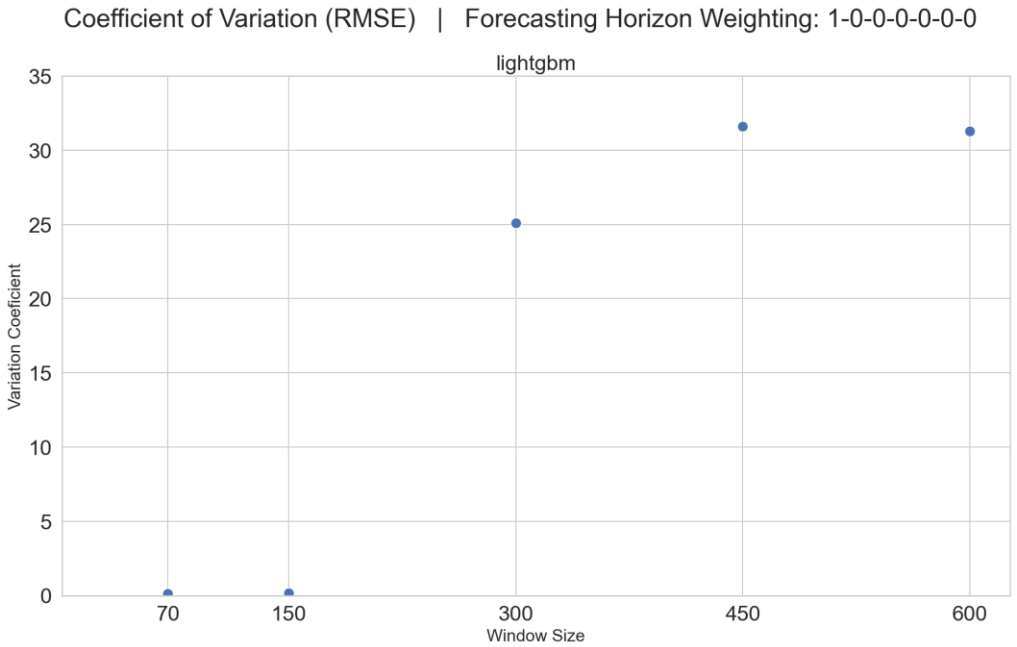


Abbildung 43: Versuch 3 | LightGBM | Variationskoeffizienten

Erläuterung

Was im ECD-Diagramm von Abbildung 42 vielleicht nicht deutlich genug zu sehen ist, wird im Boxplot von Abbildung 41 und dem Variationskoeffizienten in Abbildung 43 klarer. Der Wertebereich der Schwankungen hat stark zugenommen. Beim Vergleich der Skala vom Variationskoeffizienten ist eine Erhöhung des höchsten Wertes von knapp unter 10 in Versuch 2, auf nun einen Höchstbereich von über 30 gestiegen. Zudem sind jetzt die Schwankungen der mittleren Window-Size von 300 um mehr als den Faktor 10 angestiegen, da er mit einem Wert unter 2 relativ niedrig in Versuch 2 war und jetzt im Bereich von 25 liegt.

Somit führte der Einsatz von mehr Zeitreihen nicht zu einer stabileren Gesamtbewertung der Validierungen, sondern bewirkt größere Schwankungen durch mehr Variation in den genutzten Daten.

5.4 Auswertung

Die Annahme, dass beim Einsatz des Validierungsverfahrens Rolling-Window in Kombination mit saisonalen Zeitreihen die gewählte Window-Size relevant für eine aussagekräftige Bewertung sein könnte, kann nicht eindeutig beantwortet werden.

Beim ARIMA-Modell wurden in Versuch 1 bestätigende Ergebnisse produziert, bei denen eine zu kleine Window-Size für deutlich höhere Fehlerwerte sorgte. Dennoch scheint die Window-Size nicht mindestens so groß sein zu müssen, wie die saisonale Periode, da die Window-Size mit der halbierten Länge einer Saison ähnlich gut zu den noch größeren Window-Sizes performte.

Dagegen führten die Versuche mit dem LightGBM-Modell zu widersprechenden Ergebnissen, da die kürzeste Window-Size die beste Bewertung erhielt. Die darauffolgenden Versuche, eine Ursache in der Datenbasis zu identifizieren, änderten daran nichts. Jedoch fiel dadurch auf, dass beim Einsatz von mehr Daten in Form längerer Zeitreihen oder/und einer größeren Anzahl dieser, die Variation der Fehlerwerte mit größerer Window-Size stieg.

Das Verhalten des LightGBM-Modells in Bezug zu den unerwarteten Ergebnissen der Window-Size Größe könnte viele Ursachen haben und konnte in den durchgeführten

Versuchen nicht identifiziert werden.
Generell ist eine Art Overfitting zwar nicht auszuschließen, bei dem die kleinste Window-Size sehr gute Prognosen erzeugt, allerdings würde Overfitting dann auch für die größeren Window-Sizes auftreten und zu einer geringeren Schwankung und wahrscheinlich besseren Fehlerwerten deren führen.
Andere Ursachen könnte beispielsweise die Modellkomplexität betreffen, wofür andere Hyperparameter ausprobiert werden könnten. Auch könnte die Datengrundlage eine Rolle spielen, da LightGBM üblicherweise mehr für multivariate Daten eingesetzt wird.

Auch wenn Annahmen nicht für beide Modelle bestätigt werden konnten, geht hervor, dass die Wahl der Window-Size von mehr als der saisonalen Periode in der Zeitreihe ausgehen sollte. Je nach Daten- und Modelleigenschaften können unterschiedliche Window-Sizes zu großen Unterschieden der Bewertung eines Modells führen.

Diskussion

Eine große Schwierigkeit bei der empirischen Untersuchung war die auftretende Komplexität aller verbundenen Bestandteile. Da sehr viele Kombinationen möglich sind, kann es problematisch sein ein spezielles Verhalten für eine Untersuchung zu provozieren. Die Kombinationsmöglichkeiten ergeben sich dabei aus:

- Validierungsverfahren und zugehörigen Parameter/n
- Fehlermaßen
- Gewichtungsmöglichkeiten für Fehlermaße anhand des Forecasting Horizon
- Zu bewertende Modellart und konkrete Hyperparameter
- Verwendete Zeitreihen bzw. Konfigurationsmöglichkeiten synthetischer Zeitreihen

So war zu Anfang des Versuchsaufbaus angedacht, alle Kombinationen der Fehlermaßgewichtungen in die Ergebnisse aufzunehmen. Das dies aber nicht nur den Rahmen dieser Arbeit gesprengt, sondern auch den Fokus der Untersuchung verschlechtert hätte, musste dort begrenzt werden.

Innerhalb von Versuchen, die nicht Teil dieser Arbeit wurden, trat auch öfter ein Ressourcenproblem auf. So führe ein schlecht ausgewähltes Modell zu größeren Fehlerwerten in der

Validierung, welche in der anschließenden Auswertung durch zu viel Speicherbedarf in manchen Visualisierungen, anhand deren Berechnungen, zu Problemen führte.
Auch kommt es auf das Validierungsverfahren an, wieviel Zeitaufwand für Versuche nötig ist. Zum Beispiel ist bei der originalen Rolling-Origin Validierung die Verschiebungsweite 1 angedacht, wobei der Origin an Position 1 startet. So entstehen $n - 1$ einzelne Validierungsdurchläufe pro Zeitreihe, was bei einer großen Anzahl von langen Zeitreihen den entsprechend hohen Zeitaufwand bedeutet.

Ein anderes Ressourcenproblem betraf die Zeit. In Versuch 2 und 3 wurde für mehr und längere Zeitreihen nur das LightGBM-Modell angewandt, weil der zeitliche Aufwand des ARIMA-Modells stark von den Daten abhing. Ob dies generell der Fall ist oder an dem saisonalen ARIMA oder den gewählten Hyperparametern lag, wurde nicht weiter untersucht. So eine Analyse wäre für diese Arbeit nicht zielführend gewesen, weil die einzelnen Modelle weniger miteinander verglichen werden, als das Validierungsverfahren anhand unterschiedlicher Parameter. Der Grad der Genauigkeit der Modelle ist dabei nicht entscheidend und sollte nur ein Mindestmaß an Generalisierung der Zeitreihe aufweisen.
Dennoch blieben dadurch vergleichbare Resultate des ARIMA-Modells für Versuch 2 und 3 aus, die mit angepasster Rechenzeit nachgeholt werden könnten.

Etwas, was in dieser Arbeit bewusst ausgeschlossen wurde, ist die Verwendung von nicht synthetischen Daten für die Bewertung von Modellen mithilfe der Forward-Validierungsverfahren. Dementsprechende Untersuchungen könnten die Versuchsreihe fortführen, indem der Versuchsaufbau dafür erweitert wird.

Eine andere Richtung für weitere Experimente, die noch nicht weiter untersucht werden konnte, ist der Vergleich von unterschiedlichen Forward-Validierungsverfahren. Ein Versuchsaufbau kann dabei ein oder zwei Modelle enthalten und ähnlich zu dieser Arbeit einen grundlegenden Zeitreihenverlauf vorgeben, zu dem leicht abweichende Zeitreihen erzeugt werden. Je nach Detailgrad könnten mehrere Fehlermaße mit Gewichtung eingesetzt werden.

6 Fazit

6.1 Zusammenfassung

Im Laufe dieser Arbeit wurde gezeigt, dass es für die Bewertung von ML-Modellen in Bezug zu Zeitreihen einiges zu beachten gibt. Die Auswahl des Fehlermaßes, Bestimmung des geeigneten Validierungsverfahrens, eventuelle Parameter dessen und Berücksichtigung der zu verwendeten Daten und des Modells.

Infolgedessen wurde gezeigt, dass es eine Vielzahl von gängigen Fehlermaßen zur Bestimmung der Genauigkeit eines Modells gibt. Die Anwendung von Fehlermaßen aufgreifend, wurde für RQ1 in Abschnitt 2.7 gezeigt, dass die Kreuzvalidierung als das am häufigsten genutzte Verfahren zur Validierung gilt. Um dessen Vorgehensweise aufzuzeigen, wurden die beiden Kategorien exhaustive data splitting und partial data splitting vorgestellt und die Unterschiede beschrieben. Hinzufügend wurden die Funktionsweisen von konkreten Beispielverfahren der Kategorien erklärt.

Weiterführend wurde in Abschnitt 3.1 für RQ2 analysiert, dass die Kreuzvalidierung durch ihre zufällige Anordnung von Datenabschnitten, für Zeitreihen eher ungeeignet ist, weil vorhandene Abhängigkeiten einer Beobachtung zu Vorherigen auf diese Weise entfernt werden.
Dennoch wurde auch gezeigt, dass es Abwandlungen der Kreuzvalidierung gibt, die zum Teil für Zeitreihen verwendet werden können, da sie genau diesen Aspekt der zufälligen Anordnung modifizieren.

RQ3 wurde in Abschnitt 3.3 beantwortet, wobei spezielle Validierungsverfahren erläutert wurden, die die zeitliche Ordnung der Daten von Grund auf unangetastet lassen, indem die Daten schrittweise vom Anfang der Zeitreihe bis zum Ende verarbeitet werden. Es wurde aufgezeigt, wie diese einzelnen Forward-Validierungsverfahren funktionieren, wie sie aufeinander

aufbauen und welche Vor- und Nachteile sie haben. Daraus folgt eine Übersicht von Validierungsverfahren, die für Zeitreihen geeignet sind.

Zur Beantwortung von RQ4 wurde in Kapitel 4 ein Framework entwickelt, mit dem synthetische Zeitreihen mithilfe vorgebender Konfigurationen und modularem Aufbau generiert werden können. Durch die Kombinationsmöglichkeiten innerhalb der Konfiguration können unterschiedlichste Zeitreihen erzeugt und gezielte Anpassungen ohne weiteren Eingriff in den Quellcode bewirkt werden.

Folgend wurde für RQ5 in Kapitel 4 das Framework erweitert, indem übliche Fehlermaße und Forward-Validierungsverfahren implementiert wurden. Zur Vervollständigung wurde die Übergabe von Zeitreihen und die Anbindung beliebiger ML-Modelle anhand einer einzuhaltenden Schnittstelle hinzugefügt.

Zur Untersuchung von RQ6 wurde in Kapitel 5 das Forward-Validierungsverfahren Rolling-Window gewählt, wobei mit dessen Parameter zur Window-Size experimentiert wurde.
Dabei konnte keine eindeutige Aussage getroffen werden, weil von den eingesetzten Modellen stark unterschiedliche Ergebnisse ausgingen, deren Ursache im Weiteren nicht geklärt werden konnte. Obwohl diese unerwarteten Resultate keine eindeutige Aussage hervorbrachten, wird klar, dass Parameter wie die Window-Size nicht nur anhand der Saisonalität gewählt werden sollten und es je nach Modell und Daten zu unterschiedlichen Einschätzungen von Validierungsverfahren kommen kann.

6.2 Ausblick

6.2.1 Analyse von Forward-Validierungsverfahren

Da das ARIMA-Modell nur in Versuch 1 betrachtet wurde, könnten weitere Experimente die Untersuchungen in dessen Richtung fortsetzen.
Zudem bieten sich mehrere direkte Vergleiche von Forward- und Kreuzvalidierungen an, wobei auch untereinander getestet werden könnte, bei welchen generierten Datenstrukturen eine präzise Schätzung erzeugt wird.

Die durchgeführten Versuche könnten mit wesentlich mehr und längeren Zeitreihen wiederholt werden, wobei zusätzlich die Schwankung der Saisonalität erweitert werden könnte. Darüber hinaus könnten durch den Versuchsaufbau weitere Modelle in die Untersuchung integriert werden, um vielleicht mehr über die Ursachen der Resultate von Versuch 2 und 3 zu erfahren.

6.2.2 Offene Weiterentwicklung des Frameworks

Neben programmierspezifischen Verbesserungen gibt es auch weiterführende Pläne und Ideen, für die das bisherige Framework die Grundlage bildet. Ein paar davon möchte ich hier erwähnen:

- Die Entwicklung weiterer Pattern-Functions, Validierungsverfahren, Modelle und Fehlermaße, die direkt im System implementiert werden können.
- Als neues Feature das Testen und Auswerten der Trainingsdaten zusätzlich zu den Testdaten innerhalb von Validierungsverfahren, um den Vergleich zwischen Trainingsfehler und Testfehler auswerten zu können.
- Interaktive Erstellung einer neuen Pattern-Function, indem z.B. der Verlauf einer Linie in ein GUI-Element gezeichnet werden kann.
- Optionale automatisierte Model Selection, statt nur Model Assessment, für die Suche nach optimalen Hyperparametern eines Modells.
- Entwicklung eines interaktiven Dashboards mit dem die Konfiguration direkt in einer Weboberfläche justiert und die Generierung von Zeitreihen mit anschließendem Validieren von Modellen durchgeführt werden kann.

Literaturverzeichnis

Akiba, Takuya/Sano, Shotaro/Yanase, Toshihiko/Ohta, Takeru/Koyama, Masanori (2019). Optuna: A Next-Generation Hyperparameter Optimization Framework. In: Ankur Teredesai (Hg.). Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Anchorage AK USA, 04 08 2019 08 08 2019. New York,NY,United States, Association for Computing Machinery, 2623–2631.

Allen, David M. (1974). The Relationship Between Variable Selection and Data Agumentation and a Method for Prediction. Technometrics 16 (1), 125–127. https://doi.org/10.1080/00401706.1974.10489157.

Andrew Ng (2017). Artificial Intelligence is the New Electricity. Stanford MSx Future Forum, 25.01.2017. Online verfügbar unter https://www.youtube.com/watch?v=21EiKfQYZXc (abgerufen am 24.08.2023). Minute 03:10 bis 03:13.

Arlot, Sylvain/Celisse, Alain (2010). A survey of cross-validation procedures for model selection. Statistics Surveys 4 (none), 40–79. https://doi.org/10.1214/09-SS054.

Armstrong, J. Scott (1985). Long-range forecasting. From crystal ball to computer. 2. Aufl. New York, Wiley.

Armstrong, J. Scott/Grohman, Michael C. (1972). A Comparative Study of Methods for Long-Range Market Forecasting. Management Science 19 (2), 211–221. https://doi.org/10.1287/mnsc.19.2.211.

Bahrpeyma, Fouad/Roantree, Mark/Cappellari, Paolo/Scriney, Michael/McCarren, Andrew (2021). A Methodology for Validating Diversity in Synthetic Time Series Generation. MethodsX 8, 101459. https://doi.org/10.1016/j.mex.2021.101459.

Bergmeir, Christoph/Benítez, José M. (2012). On the use of cross-validation for time series predictor evaluation. Information Sciences 191, 192–213. https://doi.org/10.1016/j.ins.2011.12.028.

Box, George E. P./Jenkins, Gwilym M./Reinsel, Gregory C. (1994). Time series analysis. Forecasting and control. 3. Aufl. Englewood Cliffs, NJ, Prentice-Hall.

Brockwell, Peter J./Davis, Richard A. (2016). Introduction to time series and forecasting. Switzerland, Springer.

BURMAN, PRABIR/CHOW, EDMOND/NOLAN, DEBORAH (1994). A cross-validatory method for dependent data. Biometrika 81 (2), 351–358. https://doi.org/10.1093/biomet/81.2.351.

Callen, Jeffrey L./Kwan, Clarence C.Y/Yip, Patrick C.Y/Yuan, Yufei (1996). Neural network forecasting of quarterly accounting earnings. International Journal of Forecasting 12 (4), 475–482. https://doi.org/10.1016/S0169-2070(96)00706-6.

Cerqueira, Vitor/Torgo, Luis/Smailovic, Jasmina/Mozetic, Igor (2017). A Comparative Study of Performance Estimation Methods for Time Series Forecasting. In: CPS Conference Publishing Services (Hg.). 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Tokyo, Japan, 19-21 Oct., 529–538.

Crawford, Robert M. (1989). A comparative study of univariate and multivariate methodological approaches to educational research. Dissertation. 8923. Ames, Iowa, Iowa State University. https://doi.org/10.31274/rtd-180813-11057.

Emam, Khaled el/Mosquera, Lucy/Hoptroff, Richard (2020). Practical synthetic data generation. Balancing privacy and the broad availability of data. Beijing/Boston/Farnham/Sebastopol/Tokyo, O'Reilly.

Ertel, Wolfgang (2018). Introduction to Artificial Intelligence. 2. Aufl. Cham, Springer.

Falatouri, Taha/Darbanian, Farzaneh/Brandtner, Patrick/Udokwu, Chibuzor (2022). Predictive Analytics for Demand Forecasting – A Comparison of SARIMA and LSTM in Retail SCM. Procedia Computer Science 200, 993–1003. https://doi.org/10.1016/j.procs.2022.01.298.

Fildes, Robert (1992). The evaluation of extrapolative forecasting methods. International Journal of Forecasting 8 (1), 81–98. https://doi.org/10.1016/0169-2070(92)90009-X.

Görz, Günther (Hg.) (2003). Handbuch der Künstlichen Intelligenz. 4. Aufl. München, Oldenbourg Wissenschaftsverlag.

Hamilton, James D. (1994). Time Series Analysis. Princeton , NJ, Princeton University Press.

Hastie, Trevor/Tibshirani, Robert/Friedman, Jerome H. (2009). The elements of statistical learning. Data mining, inference, and prediction. New York, NY, Springer.

Hewage, Harsha Chamara/Perera, H. Niles (2021). Comparing Statistical and Machine Learning Methods for Sales Forecasting During the Post-promotional Period. In: Institute of Electrical and Electronics Engineers (Hg.). 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 13-16 December, 462–466.

Hyndman, Rob J./Koehler, Anne B. (2006). Another look at measures of forecast accuracy. International Journal of Forecasting 22 (4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001.

J. Shao (1993). Linear Model Selection by Cross-validation. Journal of the American Statistical Association 88 (422), 486–494. https://doi.org/10.2307/2290328.

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. Advances in Neural Information Processing Systems 30.

L. Devroye (1976). Nonparametric Discrimination and Density Estimation. Dissertation. Austin, Texas, The University of Texas at Austin. Online verfügbar unter https://apps.dtic.mil/sti/citations/ADA032738 (abgerufen am 29.08.2023).

Makridakis, Spyros (1990). Note—Sliding Simulation: A New Approach to Time Series Forecasting. Management Science 36 (4), 505–512. https://doi.org/10.1287/mnsc.36.4.505.

Makridakis, Spyros/Chatfield, Chris/Hibon, Michèle/Lawrence, Michael/Mills, Terence/Ord, Keith/Simmons, LeRoy F. (1993). The M2-competition: A real-time judgmentally based forecasting study. International Journal of Forecasting 9 (1), 5–22. https://doi.org/10.1016/0169-2070(93)90044-N.

Makridakis, Spyros/Spiliotis, Evangelos/Assimakopoulos, Vassilios (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. PLOS ONE 13 (3), e0194889. https://doi.org/10.1371/journal.pone.0194889.

Makridakis, Spyros/Spiliotis, Evangelos/Assimakopoulos, Vassilios (2022). M5 accuracy competition: Results, findings, and conclusions. International Journal of Forecasting 38 (4), 1346–1364. https://doi.org/10.1016/j.ijforecast.2021.11.013.

MOFC (2023). History of Competitions – MOFC. Online verfügbar unter https://mofc.unic.ac.cy/history-of-competitions/ (abgerufen am 23.08.2023).

Neupert, Titus/Greplova, Eliska/Fischer, Mark H. (2020). Machine Learning kompakt. Ein Einstieg für Studierende der Naturwissenschaften. Wiesbaden, Springer Spektrum.

Ozaki, Yoshihiko/Tanigaki, Yuki/Watanabe, Shuhei/Nomura, Masahiro/Onishi, Masaki (2022). Multiobjective Tree-Structured Parzen Estimator. Journal of Artificial Intelligence Research 73, 1209–1250. https://doi.org/10.1613/jair.1.13188.

Racine, Jeff (2000). Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. Journal of Econometrics 99 (1), 39–61. https://doi.org/10.1016/S0304-4076(00)00030-0.

Rob J Hyndman/George Athanasopoulos (2023). Forecasting: Principles and Practice. Online verfügbar unter https://otexts.com/fpp3/ (abgerufen am 30.08.2023).

Russell, Stuart J./Norvig, Peter (2004). Künstliche Intelligenz. Ein moderner Ansatz. 2. Aufl. München, Pearson Studium.

S. Geisser (1975). The Predictive Sample Reuse Method with Applications. Journal of the American Statistical Association, 320–328. https://doi.org/10.1080/01621459.1975.10479865.

S. Larson (1931). The shrinkage of the coefficient of multiple correlation. Journal of Educational Psychology, 45–55. https://doi.org/10.1037/H0072400.

Schnaubelt, Matthias (2019). A comparison of machine learning model validation schemes for non-stationary time series data. Friedrich-Alexander-Universität Erlangen-Nürnberg. Institute for Economics, Nürnberg. FAU Discussion Papers in Economics No. 11/2019. Online verfügbar unter https://www.econstor.eu/handle/10419/209136.

Silver, Nate (2020). The signal and the noise. Why so many predictions fail - but some don't. New York, Penguin Books.

Skansi, Sandro (2018). Introduction to Deep Learning. From Logical Calculus to Artificial Intelligence. Cham, Springer.

Snijders, Tom A. B. (1988). On Cross-Validation for Predictor Evaluation in Time Series. In: Dr. Theo K. Dijkstra (Hg.). On Model Uncertainty and its Statistical Implications. Springer, Berlin, Heidelberg, 56–69.

Spyros Makridakis/A. Andersen/R. Carbone/R. Fildes/M. Hibon/R. Lewandowski/J. Newton/E. Parzen/R. L. Winkler (1982). The accuracy of extrapolation (time series) methods: Results of a forecasting competition. Journal of Forecasting. https://doi.org/10.1002/for.3980010202.

Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. Journal of the Royal Statistical Society: Series B (Methodological) 36 (2), 111–133. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x.

Swanson, Norman R. (1998). Money and output viewed through a rolling window. Journal of Monetary Economics 41 (3), 455–474. https://doi.org/10.1016/S0304-3932(98)00005-1.

Swanson, Norman R./White, Halbert (1997). Forecasting economic time series using flexible versus fixed specification and linear versus nonlinear econometric models. International Journal of Forecasting 13 (4), 439–461. https://doi.org/10.1016/S0169-2070(97)00030-7.

Tashman, Leonard J. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. International Journal of Forecasting 16 (4), 437–450. https://doi.org/10.1016/S0169-2070(00)00065-0.

Tyler Vigen (2023). Spurious correlations. Online verfügbar unter https://www.tylervigen.com/spurious-correlations (abgerufen am 26.06.2023).

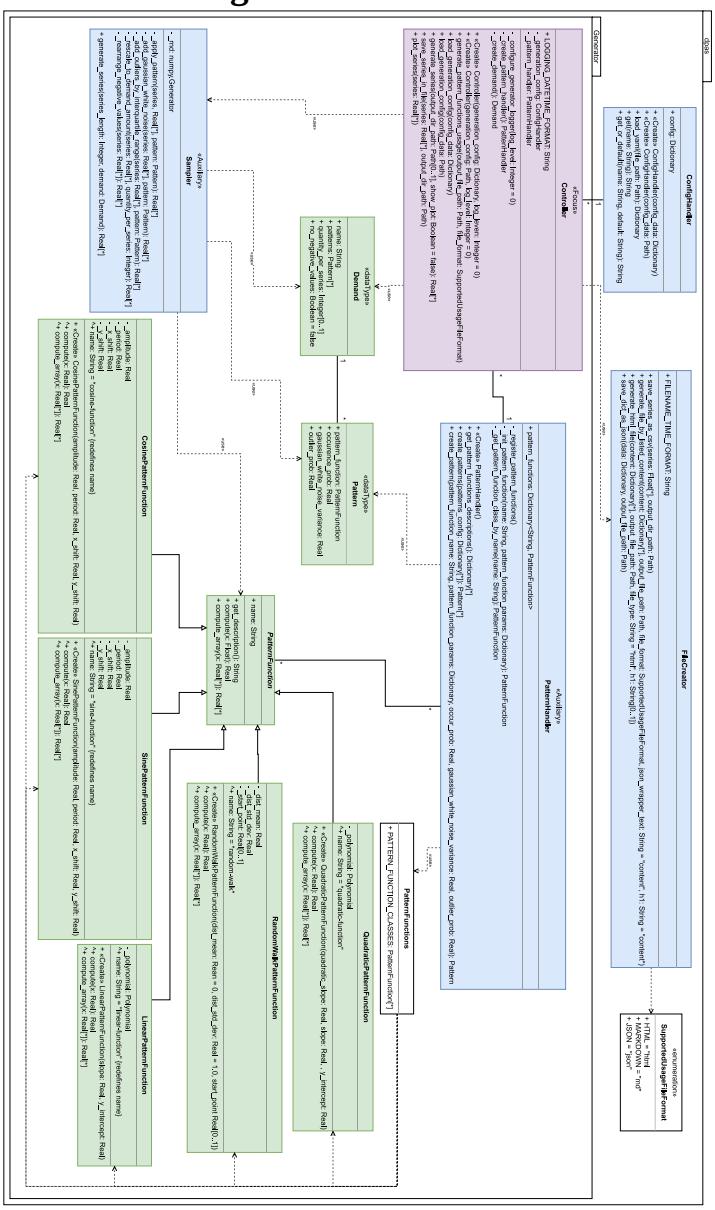
Vandeput, Nicolas/Makridakis, Spyros G. (2021). Data science for supply chain forecasting. 2. Aufl. Berlin/Boston, De Gruyter.

Vokurka, Robert J./Flores, Benito E./Pearce, Stephen L. (1996). Automatic feature identification and graphical support in rule-based forecasting: a comparison. International Journal of Forecasting 12 (4), 495–512. https://doi.org/10.1016/S0169-2070(96)00682-6.

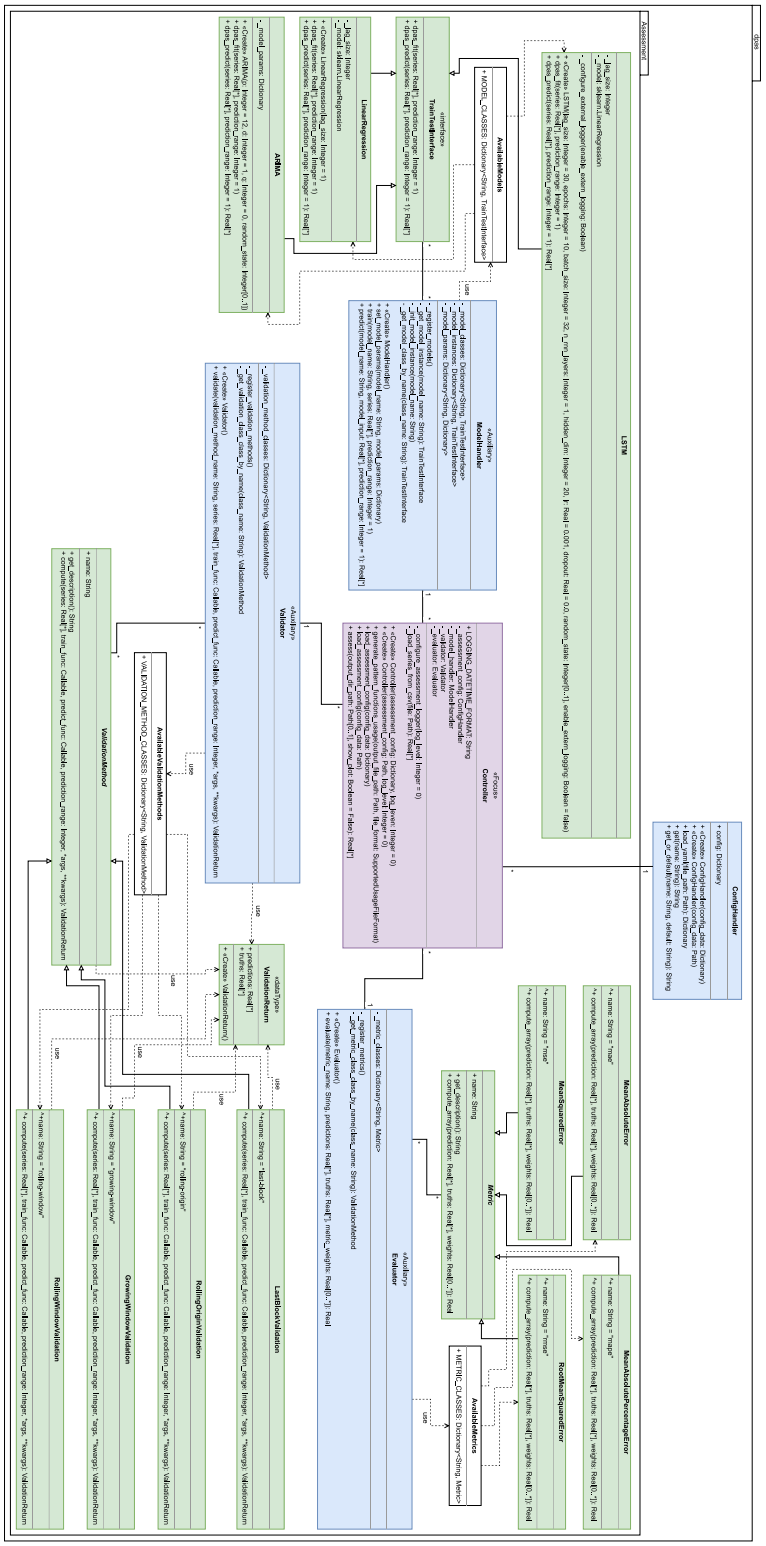
Zhang, Chi/Kuppannagari, Sanmukh R./Kannan, Rajgopal/Prasanna, Viktor K. (2018). Generative Adversarial Network for Synthetic Time Series Data Generation in Smart Grids. In: 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). 29-31 Oct. 2018, 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Aalborg, 10/29/2018 - 10/31/2018. Piscataway, NJ, IEEE, 1–6.

G Anhang

G.1 Generator Klassendiagramm



G.2 Bewertungssystem Klassendiagramm



Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original