FERDINAND PORSCHE
FERN FH

Modellselektion bei Prognosen: Ein Vergleich von Kreuzvalidierungsverfahren im Rahmen von Zeitreihenprognosen

Masterarbeit

Eingereicht von: **Patricia Kovács, B.Sc (WU) M.Sc.**

Matrikelnummer: 01551351

im Fachhochschul-Masterstudiengang Wirtschaftsinformatik

der Ferdinand Porsche FernFH GmbH

zur Erlangung des akademischen Grades

Master of Arts in Business

Betreuung und Beurteilung: Mag. Dr. Christoph Krall

Zweitgutachten: Christoph Jungbauer, BA. MA. MA.

Wiener Neudorf, Mai 2023

Ehrenwörtliche Erklärung

Ich versichere hiermit,

1. dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Inhalte, die direkt oder indirekt aus fremden Quellen entnommen sind, sind durch entsprechende Quellenangaben gekennzeichnet.
2. dass ich diese Masterarbeit bisher weder im Inland noch im Ausland in irgendeiner Form als Prüfungsarbeit zur Beurteilung vorgelegt oder veröffentlicht habe.
3. dass die vorliegende Fassung der Arbeit mit der eingereichten elektronischen Version in allen Teilen übereinstimmt.

Wiener Neudorf, 21.05.2023 _____

Unterschrift

Kurzzusammenfassung: Modellselektion bei Prognosen: Ein Vergleich von Kreuzvalidierungsverfahren im Rahmen von Zeitreihenprognosen

Das Kreuzvalidierungsverfahren wird verwendet, um den Prognosefehler eines Modells abzuschätzen. Das Verfahren wird sowohl zur Modellvalidierung, als auch zur Modellselektion bei Prognosen eingesetzt. Diese Arbeit befasst sich mit der Anwendung der Kreuzvalidierung bei stationären Zeitreihen. Der Einsatz der Kreuzvalidierung bei Zeitreihen wirft aufgrund der bestehenden Autokorrelation theoretische Probleme auf. Infolgedessen wurden mehrere Kreuzvalidierungstechniken speziell für den Fall entwickelt, dass die Daten abhängig sind. Doch welches dieser Kreuzvalidierungsverfahren die robustesten und aussagekräftigsten Prognosefehlerabschätzungen in stationären Zeitreihendaten bietet, konnte bislang in der Literatur nicht beantwortet werden. Für die Beantwortung der Forschungsfrage wurde ein empirisches Experiment verwendet. Es wurden vier Arten der Kreuzvalidierung in drei Anwendungsszenarien anhand von synthetischen und realen Daten untersucht. Die Ergebnisse zeigen, dass die geblockte und die hv-geblockte Kreuzvalidierung die robustesten Prognosefehlerschätzungen liefern, wenn synthetische Daten verwendet werden und das Prognosemodell geeignet oder nicht ganz optimal ist. Diese Erkenntnis ist unabhängig von der Länge der Zeitreihe.

Schlagwörter:

Kreuzvalidierung, Modellselektion, Modellvalidierung, Prognose, Zeitreihen, stationär, Holdout

Abstract: Model Selection for Predictions: A comparison of cross-validation methods in the context of time series forecasts

Cross-validation is used to estimate the forecast error of a given model. The method is used both for model validation and for model selection for predictions. This thesis deals with the application of cross-validation for stationary time series. The use of cross-validation for time series raises theoretical caveats due to the existing autocorrelation. Following, several cross-validation techniques have been developed for the case that the underlying data is dependent. However, the existing literature cannot answer the question, which of these cross-validation methods offers the most robust and meaningful forecast error estimates in stationary time series data. An empirical experiment was used to answer this research question. Therefore, four types of cross-validation have been examined in three application scenarios using synthetic and real-world data. The received results show that blocked and hv-blocked cross-validation provide the most robust forecast error estimates when synthetic data are used and the forecast model is appropriate or sub-optimal. This finding is independent of the length of the time series.

Keywords:

Cross-validation, Model Selection, Model Validation, forecast, time series, stationary, Holdout

Danksagung

An dieser Stelle möchte ich mich herzlich bei meinem Betreuer Mag. Dr. Christoph Krall für die angenehme Zusammenarbeit und die Unterstützung dieser Masterarbeit bedanken. Sein konstruktives Feedback trug wesentlich zum Gelingen dieser Arbeit bei. Während der gesamten Arbeit unterstützte er mich mit wertvollen Denkanstößen.

Meinen Studienkollegen:innen Lisa und Thomas danke ich für die motivierenden Worte und das ständige Aufmuntern. Ich könnte mir keine besseren Kollegen:innen für mein Studium hier an der Ferdinand Porsche FernFH wünschen.

Mein besonderer Dank gebührt meiner Familie. In erster Linie meinen Eltern, Großeltern und meiner Schwester: *Danke, dass es euch gibt.*

Zuletzt möchte ich meinem Freund Stefan danken, für seine aufbauenden Worte und seine Unterstützung: *Vielen Dank, dass du mich immer wieder an die wirklich wichtigen Dinge im Leben erinnerst.*

Wiener Neudorf, im Mai 2023

Inhaltsverzeichnis

1 Einleitung

Eines der Kernthemen von Internet of Things (IoT) ist die sinnvolle Verarbeitung der generierten Daten [Ha17, S. 206]. Mit Hilfe von unterschiedlichen Modellen können diese Daten mitunter dafür verwendet werden Prognosen für die Zukunft abzugeben. Das Ziel von statistischen Modellen ist die mathematische Erfassung realer Zusammenhänge, welche eine Abschätzung von Zielgrößen in Abhängigkeit von bekannten Werten anderer Variablen erlaubt. Exakte Vorhersagen von Einzelbeobachtungen anhand von statistischen Modellen sind prinzipiell nicht möglich, da der Zufall hier eine zu große Rolle spielt. Doch selbst für die deterministische Komponente von Prozessen existiert praktisch kein Modell, das zukünftige Werte exakt über den gesamten Anwendungsbereich darstellt. Wie der Statistiker George Box formulierte: „All models are wrong, but some are useful." [BD87, S.74].

Im Kontext der Prognose von Zeitreihendaten bedeutet dies, dass jedes Prognosemodell Fehler in der Vorhersage von zukünftigen Beobachtungen aufweist. In der Modellselektion bei Prognosen gilt es daher, jenes Prognosemodell mit seinen Parametern zu wählen, dass den kleinsten Vorhersageverlust besitzt. Mit anderen Worten, sollte jenes Modell unter allen Kandidatenmodellen gewählt werden, das den kleinsten Prognosefehler besitzt [DTJ18].

1.1 Problemstellung

Zur Modellselektion bei Prognosen gibt es viele verschiedene Techniken. Eine hiervon ist die validierungsbasierte Modellselektion mittels Kreuzvalidierungsverfahren. Dieses bewertet die Leistung des Prognosemodells anhand einer Schätzung des Prognosefehlers [DTJ18]. Folglich wird das Kreuzvalidierungsverfahren für die folgenden zwei Aufgaben verwendet: (i) um eine Schätzung der Zuverlässigkeit und Genauigkeit des Modells abzugeben und (ii) diese Bewertung als Modellselektionskriterium zu verwenden, um das beste Prognosemodel unter den Kandidaten zu wählen [BB12, S. 192].

Der Statistiker Seymour Geisser bewies, dass die standardmäßige k-fache Kreuzvalidierung eines der zuverlässigsten Modellvalidierungsverfahren ist, wenn Beobachtungen unabhängig und identisch verteilt (u.i.v.) sind [Ge75]. Einer der Gründe hierfür ist die effiziente Nutzung der vorhandenen Daten [AC10, S.61]. Das Problem ist jedoch, dass viele reale Datensätze nicht u.i.v. sind. Ein Beispiel für solch einen Datensatz sind Zeitreihen. Zeitreihendaten sind eine sortierte Folge von Daten, die in regelmäßigen

Intervallen erfasst werden. Ein typisches Merkmal von Zeitreihen ist, dass die Daten oft abhängig voneinander sind [CTM20, S.2]. Diese Abhängigkeit wirft theoretische Probleme für die Verwendung der standardmäßigen k-fachen Kreuzvalidierung als Modellvalidierungsverfahren in Zeitreihendaten auf. Infolgedessen wurden mehrere Kreuzvalidierungstechniken speziell für den Fall entwickelt, dass die Daten abhängig sind: die geblockte Kreuzvalidierung [Sn88], die modifizierte Kreuzvalidierung [MT98] und die hv-geblockte Kreuzvalidierung [Ra00]. Welche dieser Varianten der Kreuzvalidierung sich am besten als Modellvalidierungs- und Modellselektionstechnik in Zeitreihendaten eignet, konnte bislang in der Theorie nicht beantwortet werden. Diese Lücke versucht die Masterarbeit zu verkleinern.

Eine zuverlässige Beurteilung des statistischen Modells ist ein kritischer Punkt bei der Modellsuche. Die Wahl des richtigen Kreuzvalidierungsverfahrens ist daher entscheidend, um zuverlässige Prognosen zu erhalten.

1.2 Ziel und Hypothese

Das Arbeitsziel der Masterarbeit ist die Qualifikation der vier Kreuzvalidierungsmethoden als Modellvalidierungsverfahren in Zeitreihenprognoseaufgaben zu vergleichen und diese empirisch auf ihre Eignung zu testen. Anhand eines Experimentes mit synthetischen und realen Zeitreihendaten soll eine Antwort auf die folgende Forschungsfrage gefunden werden:

Welches Kreuzvalidierungsverfahren bietet die robustesten und aussagekräftigsten Prognosefehlerabschätzungen in stationären Zeitreihendaten?

Darüber hinaus wird untersucht, ob die theoretischen Mängel der k-fachen Kreuzvalidierung in der Praxis relevant sind und welchen Einfluss die Datenlänge der Zeitreihe auf die Leistung der vier Kreuzvalidierungsverfahrens hat.

1.3 Struktur der Arbeit

Die Arbeit bietet zunächst einen Überblick über die theoretischen Grundlagen für die Analyse und Prognose von Zeitreihen, die im Rahmen dieser Masterarbeit relevant sind. Dies umfasst eine Einführung in grundlegende Begriffe der Zeitreihenanalyse, sowie eine Beschreibung des Autoregressiven Modells und des Moving Average Modells. Anschließend wird die Modellselektion bei Prognosen näher erläutert und in Kapitel 3 die theoretische Grundlage für das validierungsbasierte Modellselektionsverfahren gesetzt.

Dabei wird erklärt, wie die Modelselektion mithilfe des Holdout-Verfahrens und der vier untersuchten Kreuzvalidierungsverfahren (k-fache Kreuzvalidierung, geblockte Kreuzvalidierung, modifizierte Kreuzvalidierung und die hv-geblockte Kreuzvalidierung) erfolgt. Im Anschluss daran wird auf die Vor- und Nachteile der Kreuzvalidierung eingegangen, sowie einige Anwendungsbeispiele des Verfahrens in der Praxis aufgezeigt. Kapitel 2 bis 6 stellen daher den Literaturteil der Arbeit dar. Ziel ist den aktuellen Wissensstand zum Thema „Modellselektion bei Prognosen anhand des Kreuzvalidierungsverfahrens“ zusammenzufassen und die bestehenden Lücken hervorzuheben, welche diese Arbeit versucht zu minimieren.

Die Beantwortung der gestellten Forschungsfrage erfolgt anhand eines empirischen Experimentes. In Kapitel 7 wird der Aufbau dieses Experimentes ausführlich vorgestellt. Dabei wird insbesondere auf die verwendeten synthetischen und realen Datensätze, das angewandte Prognosemodell, die verwendeten Fehlermaße, sowie die drei Anwendungsszenarien und die untersuchten Modellvalidierungsverfahren eingegangen. Dieses Kapitel soll eine umfassende Verständnisgrundlage für das durchgeführte Experiment in *R* schaffen. Das darauffolgende Kapitel wertet die erzielten Ergebnisse des empirischen Experimentes aus, welche anschließend in Kapitel 9 noch auf den Einfluss der Datenlänge der Zeitreihe überprüft werden. Zum Abschluss erfolgt eine Zusammenfassung der Resultate, gefolgt von einer Diskussion der Methode und einem Ausblick auf offene Fragestellungen zu dieser Arbeit.

2 Lineare Modellierung von Zeitreihen

Die Verwendung der linearen Zeitreihenanalyse ist in vielen Bereichen der Informatik etabliert. So werden Zeitreihenmodelle beispielsweise für die Vorhersage von Systemausfällen, in der Erkennung von Anomalien oder in der Überwachung von Netzwerkverkehr verwendet. Weitere Anwendungsgebiete der Zeitreihenanalyse befinden sich im Bereich künstliche Intelligenz, Machine Learning und Internet of Things, um Muster in den Zeitreihendaten zu erkennen oder um Prognosen zu treffen. Diese Masterarbeit beschäftigt sich mit der Prognose von Zeitreihendaten. Das grundlegende Konzept bei der linearen Prognose von Zeitreihendaten besteht darin, dass ein zukünftiger Wert x_{t+1} eine lineare Beziehung zu anderen Zeitreihendaten aufweist.

Dieses Kapitel enthält eine Einführung in Grundbegriffe und Aufgaben der Zeitreihenanalyse, sowie eine Darstellung der wichtigsten linearen Prozessmodelle für Zeitreihen. Es werden die theoretischen Grundlagen für die Analyse und Prognose von Zeitreihendaten dargestellt, die in den nachfolgenden Kapiteln dieser Masterarbeit verwendet und vorausgesetzt werden. Da es eine Vielzahl an Bücher gibt, die das Thema Zeitreihenanalyse detailliert aufrollen (siehe [BD16, Ha11, Ha94]) werden einige Aspekte nur kurz zusammengefasst. Jene Aspekte, die im Zusammenhang mit der Masterarbeit stehen, werden hingegen ausführlicher beleuchtet. Hierzu gehören Begrifflichkeiten wie stationär und Autokorrelation.

2.1 Grundlagen der Zeitreihenanalyse

Eine Zeitreihe ist eine Reihe von Beobachtungen x_t, die jeweils zu einem bestimmten Zeitpunkt $t = 0, 1, 2, \ldots$ aufgezeichnet werden. Erfolgt die Aufzeichnung in regelmäßigen Intervallen (stündlich, täglich, monatlich, usw.) so handelt es sich um eine diskrete Zeitreihe. Da auch Prozesse in kontinuierlicher Zeit nur zu diskreten Zeitpunkten gemessen werden, werden in der Praxis diskrete Zeitreihen betrachtet, wie beispielsweise bei der Wetterprognose oder der Analyse von Verkaufszahlen [BD16, S. 1-2].

Das grundlegende Ziel in der Zeitreihenanalyse ist die Bestimmung eines Modells, das das Muster der Zeitreihen optimal beschreibt. Das gefundene Modell wird anschließend verwenden, um

- die wichtigsten Merkmale des Zeitreihenmusters zu beschreiben;
- zukünftige Werte der Zeitreihe zu prognostizieren;

- zu erklären, wie sich die Vergangenheit auf die Zukunft auswirkt; oder
- um als Kontrollstandard zu dienen, damit Abweichungen schnell entdeckt werden [Jo22, S. 3-4].

Diese Masterarbeit beschränkt sich auf die Prognose von diskreten Zeitreihenwerten. Bei der Prognose von Zeitreihendaten werden zukünftige Beobachtungen anhand von historischen Daten vorhergesagt. Die historischen Daten werden nach Trends und Muster untersucht, um diese Informationen dann für die Vorhersage zukünftiger Werte zu verwenden. Diese Aufgabe erweist sich jedoch oft als schwierig, da Zeitreihendaten komplexe Muster aufweisen, wie beispielsweise Trends, Saisonalitäten, Zyklen und unregelmäßige Schwankungen. Auch der Zufall spielt eine entscheidende Rolle, weshalb dies in den stochastischen Prozessen ebenfalls beachtet wird [HA18]. Um Zeitreihendaten bestmöglich prognostizieren zu können, sollten

- die Zeitreihendaten stationär sein;
- aktuelle Beobachtungen der Zeitreihe von den vergangenen Beobachtungen abhängen;
- keine Werte in der Zeitreihe fehlen; und
- die zufälligen Fehlerterme ε_t normalverteilt sein [Jo22, S.4].

Schwach stationäre Zeitreihen besitzen einen konstanten Mittelwert, eine konstante Varianz und die Kovarianz ist unabhängig von der Zeit. Mit anderen Worten hängen die ersten und zweiten Momente der Zeitreihe nicht vom Zeitpunkt der Beobachtung ab. Eine Verschiebung entlang der Zeitachse hat keine Auswirkung auf den Mittelwert, die Varianz und die Kovarianz. Für eine stationäre Zeitreihe $\{x_1, x_2, \dots, x_t\}$ müssen für alle t und p folgende zwei Bedingungen erfüllt sein:

$$\text{(i)} \qquad \mathrm{E}(x_1) = \mathrm{E}(x_2) = \cdots = \mathrm{E}(x_t) = \mu; \tag{1}$$

$$\text{(ii)} \qquad \mathrm{Cov}(x_1, x_{1+p}) = \mathrm{Cov}(x_2, x_{2+p}) = \cdots = \mathrm{Cov}(x_t, x_{t+p}), \tag{2}$$

wobei t einen gegebenen Zeitpunkt angibt und p eine beliebige Verschiebung entlang der Zeitachse ist. Bedingung (ii) impliziert, dass die Varianz konstant ist und daher unabhängig von der Zeit t ist: $\mathrm{Var}(x_1) = \mathrm{Var}(x_2) = \cdots = \mathrm{Var}(x_t) = \sigma^2$ [Ha94, S. 34]. Die Annahme der Stationarität ist für Zeitreihenprognosen wichtig, da sie den Modellierungsprozess deutlich vereinfachen. Sind die Zeitreihendaten stationär, so bleiben die statistischen Eigenschaften der Zeitreihe auch in der Zukunft gleich. Mithilfe von historischen Daten können so genauere Prognosen über zukünftige Beobachtungen erzielt werden. Sind die Zeitreihendaten nicht stationär, so ändert sich die statistische

Eigenschaft der Zeitreihe bei Verschiebung entlang der Zeitachse, was eine genaue Prognose deutlich erschwert. Für die Überprüfung der Stationarität einer Zeitreihe kann beispielsweise der Augmented Dickey-Fuller (ADF)-Test verwendet werden. Im Fall, dass die Daten nicht stationär sind, können sie durch eine (saisonale) Differenzierung transformiert werden, um stationäre Daten zu erhalten [HA18].

Die Autokorrelation ist ebenfalls eine wichtige Annahme, um Zeitreihen bestmöglich prognostizieren zu können. Sie misst die vorhandene Abhängigkeit zwischen einer aktuellen Beobachtung und einer vergangenen Beobachtung in einer Zeitreihe. Die p-te Autokorrelation eines schwach stationären Prozesses, bezeichnet als ρ_p, ist definiert als die p-te Autokovarianz geteilt durch die Varianz,

$$\rho_p = \frac{\mathrm{Cov}(x_t, x_{t+p})}{\sqrt{\mathrm{Var}(x_t)}\sqrt{\mathrm{Var}(x_{t+p})}} = \frac{\mathrm{Cov}(x_t, x_{t+p})}{\sigma^2}. \quad (3)$$

Gleichung 3 verdeutlicht, dass die Autokorrelation bei einem stationären Prozess ausschließlich von der zeitlichen Distanz p (auch als Lag bezeichnet) abhängt. Zum Beispiel misst ρ_1 die lineare Abhängigkeit zwischen x_1 und x_2, und ρ_6 die Abhängigkeit zwischen x_1 und x_7 [Ha94, S. 49].

Zeitreihen, die keine Autokorrelation aufweisen, werden als White Noise-Prozesse bezeichnet. Der oft als auch Zufallsprozess bezeichnete stochastische Prozess ε_t besitzt die Eigenschaften, dass

- ε_t identisch verteilt ist;
- $\mathrm{E}(\varepsilon_t) = 0$; und
- $\mathrm{Cov}(\varepsilon_t, \varepsilon_{t+p}) = 0$ für alle $p \neq 0$.

White Noise-Prozesse sind oft ein wichtiger Bestandteil von verschiedenen Modellen in der Zeitreihenanalyse, darunter auch bei Autoregressiven- und Moving Average-Modellen [HA18].

Für die Prognose von Zeitreihen gib es viele verschiedene statistische Modelle. Die Wahl des Prognosemodells hängt hierbei insbesondere von den Merkmalen in den Zeitreihendaten ab. Zu den wichtigsten Prozessmodellen für Zeitreihen gehört der White Noise-Prozess, der Random Walk-Prozess, der Moving Average-Prozess, sowie der Autoregressive-Prozess [Bi23]. Diese Masterarbeit verwendet ein Autoregressives-Modell und ein Moving Average-Modell, weshalb diese Modelle in den nächsten Unterkapiteln näher dargestellt werden.

2.2 AR-Modellierung von Zeitreihen

Autoregressive-Modelle (AR) basieren auf der Idee, dass der aktuelle Wert der Zeitreihe x_t als Funktion von p vergangenen Werten $x_t, x_{t-2}, \ldots., x_{t-p}$ erklärt werden kann, wobei p als Lag bezeichnet wird und die Anzahl der Schritte in der Vergangenheit bestimmt. Beobachtungen aus früheren Zeitschritten werden als Eingangsparameter genutzt, um den Wert des nächsten Zeitschritts vorherzusagen. Formal entspricht ein AR-Modell einem linearen Regressionsmodell für x_t mit den vergangene Werten als Regressoren [SS11, S. 84]. Handelt es sich beispielsweise um ein AR(1) Modell, so bedeutet dies, dass $p = 1$ ist. Der aktuelle Wert x_t kann anhand der vergangenen Beobachtung x_{t-1} erklärt werden. Mathematisch kann dies durch Gleichung 4 dargestellt werden,

$$x_t = c + \Phi x_{t-1} + \varepsilon_t. \tag{4}$$

c ist eine Konstante und Φ ist ein Koeffizient der bestimmt, in welchem Maß der Prozess von seinen, eigenen vergangenen Werten beeinflusst wird. ε_t repräsentiert den Fehlerterm und ist ein White Noise-Prozess mit $\mathrm{E}(\varepsilon_t) = 0$ und $\mathrm{Var}(\varepsilon_t) = \sigma_\varepsilon^2$. Der Fehlerterm ε_t stellt die unvorhersehbare Komponente des Prozesses zum gegebenen Zeitpunkt t dar [SS11, S.85].

Damit ein AR(1)-Prozess schwach stationär ist, muss der Betrag des Koeffizienten Φ kleiner als eins sein. Ist $|\Phi| \geq 1$, so akkumulieren sich die Einflüsse von $\varepsilon_t, \varepsilon_{t-1,}, \varepsilon_{t-2}, \ldots$ für die Zeitreihe x, was zu einem nicht-stationären Prozess führt. Gilt $|\Phi| < 1$, so verschwinden die Einflüsse von $\varepsilon_t, \varepsilon_{t-1,}, \varepsilon_{t-2}, \ldots$ für die Zeitreihe x und der Prozess ist stationär [Ha, S. 53].

Abbildung 1 veranschaulicht im oberen Teil (a) einen stationären AR(1)-Prozess mit $\Phi = -0.7$ und zeigt im unteren Teil (b) einen nicht stationären AR(1)-Prozess mit $\Phi = -1.03$. In beiden Fällen ist $\sigma^2 = 1$ und der Koeffizient Φ ist negativ. Ein negativer Wert von Φ bedeutet, dass zwei aufeinander folgende Beobachtungen negativ korreliert sind, aber Beobachtungen, die zwei Zeitpunkte auseinander liegen, positiv korreliert sind. Ist beispielsweise x_t positiv, so ist die nächste Beobachtung x_{t+1} in der Regel negativ und die übernächste Beobachtung x_{t+2} typischerweise erneut positiv. Dieses Muster ist in beiden Zeitdiagrammen in Abbildung 1 deutlich erkennbar. Der entscheidende Unterschied zwischen den zwei Diagrammen ist der Wert von Φ, der bei einem AR(1)-Prozess verantwortlich für die Annahme der Stationarität ist. Im oberen Zeitdiagramm (a) ist der Betrag des Koeffizienten Φ kleiner als eins. Der dazugehörige Prozess weist keine langfristigen Trends oder systematische Veränderungen im Zeitverlauf auf. Der

Mittelwert und die Varianz des Prozesses bleiben über den gesamten Beobachtungsraum konstant. Solch ein stationärer Prozess wird auch als stabil bezeichnet, da der Prozess nicht unbegrenzt wächst oder zerfällt. Im unteren Zeitdiagramm (b) ist der Betrag des Koeffizienten Φ größer als eins und der Prozess daher nicht stationär. Konkret bedeutet dies in diesem Fall, dass die Varianz des Prozesses im Laufe der Zeit immer größer wird. Die Beobachtungen werden sukzessive in absoluten Werten immer größer, weshalb solch ein Prozess auch als explosiv bezeichnet wird. Dieses Verhalten ist charakteristisch für einen nicht-stationären AR(1)-Prozess mit einem negativen Φ-Koeffizienten [SS11].

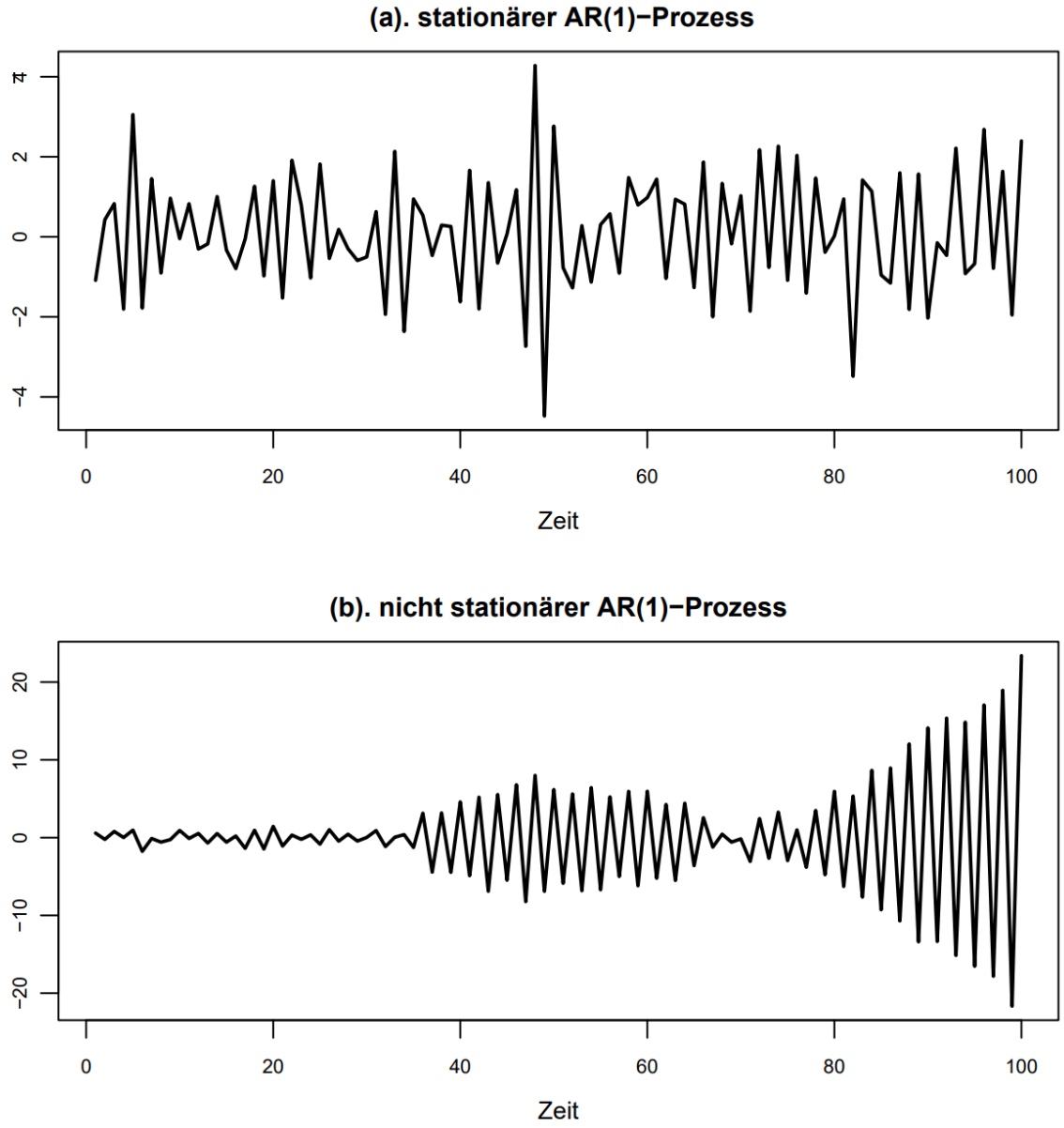

Abbildung 1: Veranschaulichung eines (a) stationären AR(1) Prozesses mit $\Phi = -0.7$ *und eines (b) nicht stationären AR(1) Prozesses mit* $\Phi = -1.03$.

Bei einem AR-Prozess zweiter Ordnung ist der aktuelle Wert der Zeitreihe x_t von den vergangenen zwei Beobachtungen x_{t-1} und x_{t-2} abhängig. Die Gleichung eines AR(2) Modells hat daher die Form

$$x_t = \Phi_1 x_{t-1} + \Phi_2 x_{t-2} + \varepsilon_t. \tag{5}$$

Damit ein AR(2)-Prozess stationär ist, müssen die Koeffizienten Φ_1 und Φ_2 die Bedingungen erfüllen, dass

- $\Phi_1 + \Phi_2 < 1$;
- $\Phi_1 - \Phi_2 < 2$; und
- $|\Phi_2| < 1$.

Nur wenn alle drei Bedingungen erfüllt sind, ist ein AR(2)-Prozess stationär [SS11, S.96]. Ob ein AR(2)-Prozess stationär ist oder nicht, hat bei der Prognose von Zeitreihen eine zentrale Bedeutung.

Aus Gleichung 4 und 5 lässt sich die allgemeine Form eines AR-Modells der Ordnung p ableiten. Ein AR(p)-Modell hat daher die Form

$$x_t = c + \Phi_1 x_{t-1} + \Phi_2 x_{t-2} + \Phi_3 x_{t-3} + \cdots + \ + \Phi_\mathrm{p} x_{t-p} + \varepsilon_t. \tag{6}$$

Wenn $p \geq 3$ ist, so sind die Bedingungen für einen stationären AR(p)-Prozess viel komplexer. Eine detaillierte Erklärung hierfür bieten die Referenzen [Ha11, Ha94, SS11].

2.3 MA-Modellierung von Zeitreihen

Ein Moving Average-Modell (MA) ist ein regressionsähnliches Modell und verwendet den aktuellen Fehlerterm ε_t und eine lineare Kombination von vergangenen Fehlertermen $\varepsilon_t, \varepsilon_{t-1}, \varepsilon_{t-2}, \dots, \varepsilon_{t-q}$, um den aktuellen Wert der Zeitreihe x_t zu bestimmen. Der Fehlerterm ε ist ein White Noise-Prozess mit $\mathrm{E}(\varepsilon_t) = 0$, $\mathrm{Var}(\varepsilon_t) = \sigma_\varepsilon^2$ und $\mathrm{Cov}(\varepsilon_t, \varepsilon_{t+p}) = 0$ für alle $p \neq 0$. Die Ordnung q gibt die Anzahl der Schritte in der Vergangenheit an. Somit bestimmt q die Anzahl an vergangenen Fehlertermen in dem MA-Modell. Bei einem MA-Prozess der Ordnung 1, hängt beispielsweise der aktuelle Werte x_t von den Fehlertermen ε_t und ε_{t-1} ab

$$x_t = c + \varepsilon_t + \theta \varepsilon_{t-1}. \tag{7}$$

Gleichung 7 veranschaulicht einen MA(1)-Prozess, wobei c eine Konstante ist und θ den Koeffizienten des verzögerten Fehlerterms darstellt [SS11, S. 90].

Der Erwartungswert von x_t ist

$$\mathrm{E}(x_t) = \mathrm{E}(c + \varepsilon_t + \theta\varepsilon_{t-1}) = c + \mathrm{E}(\varepsilon_t) + \mathrm{E}(\theta\varepsilon_{t-1}) = c + 0 + 0 = c, \tag{8}$$

da ε ein White Noise-Prozess mit Erwartungswert null ist.

Die Varianz von x_t ist gegeben durch

$$\begin{aligned} \mathrm{Var}(x_t) = \mathrm{E}(x_t - c)^2 &= \mathrm{E}(\varepsilon_t - \theta\varepsilon_{t-1})^2 = \\ &= \mathrm{E}(\varepsilon_t^2 - 2\theta\varepsilon_t\varepsilon_{t-1} + \theta^2\varepsilon_{t-1}^2) = \\ &= \sigma_\varepsilon^2 + 0 + \theta^2\sigma_\varepsilon^2 = \\ &= (1 + \theta^2)\sigma_\varepsilon^2. \end{aligned} \tag{9}$$

Die erste Autokovarianz ist

$$\begin{aligned} \mathrm{Cov}(x_t, x_{t-1}) &= \mathrm{E}\big((x_t - c)(x_{t-1} - c)\big) = \\ &= \mathrm{E}((\varepsilon_t + \theta\varepsilon_{t-1})(\varepsilon_{t-1} + \theta\varepsilon_{t-2})) = \\ &= \mathrm{E}(\varepsilon_t\varepsilon_{t-1} + \theta\varepsilon_{t-1}^2 + \theta\varepsilon_t\varepsilon_{t-2} + \theta^2\varepsilon_{t-1}\varepsilon_{t-2} = \theta\sigma_\varepsilon^2. \end{aligned} \tag{10}$$

Gleichung 8, 9 und 10 zeigen, dass sowohl der Erwartungswert, als auch die Varianz und die Kovarianz des MA(1)-Prozesses Konstanten sind. Alle drei Momente sind keine Funktionen der Zeit t, sondern immer konstant. Dies impliziert, dass ein MA(1)-Prozess immer eine schwach stationäre Zeitreihe ist. Der Mittelwert, die Varianz und die Kovarianz sind unabhängig von der Zeit. Im Gegensatz zu einem AR-Modell, spielt bei einem MA-Modell die Wahl des Koeffizienten θ keine tragende Rolle in der Entscheidung, ob der Prozess stationär ist oder nicht [Ha94, S. 48].

Im Allgemeinen ist ein MA-Modell der Ordnung q gegeben als,

$$x_t = c + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_q\varepsilon_{t-q}. \tag{11}$$

Auch hier sind Erwartungswert, Varianz und Kovarianz unabhängig von der Zeit t. Ein MA(q)-Prozess ist daher für beliebige Werte von $(\theta_1, \theta_2, \ldots, \theta_q)$ stationär. An dieser Stelle sei jedoch angemerkt, dass ein MA(q)-Prozess nur stationär ist, wenn der Wert von q endlich ist, sprich die Koeffizienten $(\theta_1, \theta_2, \ldots, \theta_\infty)$ summierbar sind [Ha94, S. 51-52].

Zwischen AR- und MA-Modellen gibt es enge Zusammenhänge, da jeder stationäre AR(p)-Prozess, als stationärer MA(∞)-Prozess dargestellt werden kann. Gleichung 12 veranschaulicht dies durch wiederholte Substitution für ein AR(1)-Modell.

$$
\begin{aligned}
x_t &= c + \Phi x_{t-1} + \varepsilon_t = \\
&= c + \varepsilon_t + \Phi(\mathrm{c} + \varepsilon_{t-1}) + \ \Phi^2 x_{t-2} \ = \\
&= c + \varepsilon_t + \Phi(c + \varepsilon_{t-1}) + \Phi^2(c + \varepsilon_{t-2}) + \ \Phi^3 x_{t-3} = \\
&= c + \varepsilon_t + \Phi(c + \varepsilon_{t-1}) + \Phi^2(c + \varepsilon_{t-2}) + \ \Phi^3(c + \varepsilon_{t-3}) + \cdots.
\end{aligned} \tag{12}
$$

Unter der stationären Bedingung für einen AR(1)-Prozess, dass $|\Phi| < 1$, kann der Prozess daher auch in Form eines MA(∞)-Modells ausgedrückt werden,

$$
x_t = \frac{c}{1-\Phi} + \ \varepsilon_t + \ \Phi\varepsilon_{t-1} + \ \Phi^2\varepsilon_{t-2} + \ \Phi^3\varepsilon_{t-3} + \cdots. \tag{13}
$$

Umgekehrt ist es auch möglich jeden invertierbaren MA(q)-Prozess, als einen AR(∞)-Prozess darzustellen [HA18].

3 Modellselektion bei Prognosen

Bei der Modellselektion steht meist die Frage: „Welches Modell ist das Beste?“ im Vordergrund. Die Bewertung von „das Beste“ ist von verschiedenen Kriterien abhängig und ist mit dem Modellierungszweck verbunden. Ist das Ziel des Modells beispielsweise ein natürliches Phänomen zu erklären, so werden zur Modellselektion andere Kriterien herangezogen, als wenn das Ziel des Modells ist, eine Prognose zu treffen [Sy11]. Die vorliegende Masterarbeit konzentriert sich auf die Modellselektion bei Prognosen, weshalb im weiteren Verlauf die Kriterien hierfür näher beleuchtet werden.

In der Modellselektion bei Prognosen gilt es, jenes Prognosemodell (mit seinen Parametern) zu wählen, das die beste out-of-sample Vorhersagekraft bietet. Mit anderen Worten, jenes Modell, das unter allen Kandidatenmodellen den kleinsten Prognosefehler besitzt. In der Praxis kann der Prognosefehler jedoch nur mit Hilfe der In-Set-Daten geschätzt werden, da die Out-Set-Daten zukünftige Beobachtungen beinhalten und daher nicht verwendet werden können.[1] Das Kriterium in der Modellselektion bei Prognosen ist daher die Minimierung des geschätzten Prognosefehlers [DTJ18].

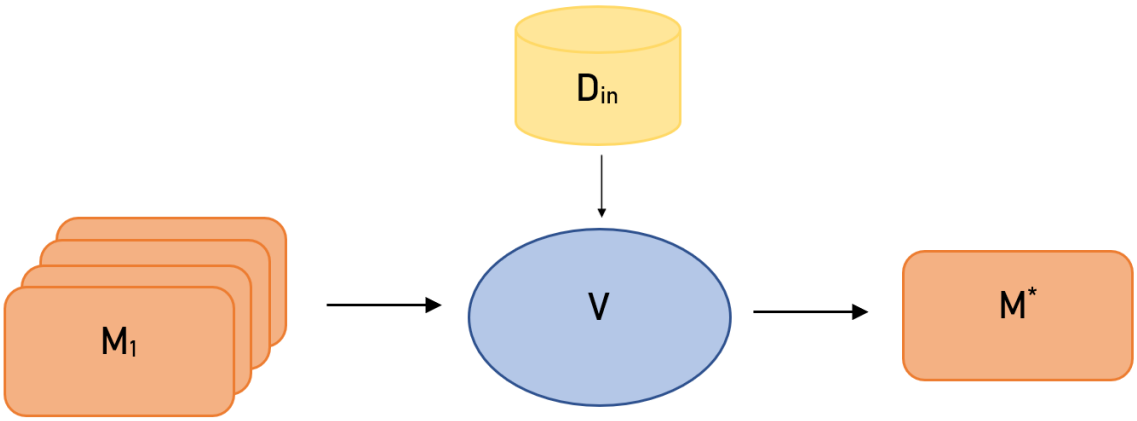

Abbildung 2: Typischer Verlauf einer Modellselektion bei Prognosen, in Anlehnung an [CTS21, S.5].

Abbildung 2 zeigt den typischen Verlauf einer Modellselektion bei Prognosen in der Praxis. Für die Modellselektion werden ausschließlich die verfügbaren In-Set-Daten D_{in} verwendet. Darüber hinaus ist eine Auswahl an Vorhersagemodellen M von n Alternativen $M = \{M_1, \dots, M_n\}$ gegeben. Mit Hilfe eines gewählten Modellvalidierungsverfahrens V wird jede Alternative anhand der In-Set-Daten D_{in} bewertet. Jenes Modell, das die Vorhersagekraft gemäß dem Modellvalidierungsverfahren

[1] Kapitel 7.1 bietet eine ausführliche Erklärung für die Begrifflichkeiten In-Set- und Out-Set-Daten.

maximiert, wird ausgewählt und in zukünftigen Beobachtungen verwendet. Das Ziel besteht darin $M^* \in M$ zu finden, das Modell welches die beste out-of-sample Vorhersagekraft besitzt. Formal lässt sich das Modellselektionsproblem wie folgt darstellen [CTS21, S.5]:

$$M^* \in \operatorname{argmin}\left(\widehat{PE}(M, D_{in}, V)\right), \tag{14}$$

wobei $\widehat{PE}$ das Fehlermaß des geschätzten Prognosefehlers darstellt (um den erwarteten Fehler eines Prognosemodells zu quantifizieren), V das Modellvalidierungsverfahren und D_{in} die In-Set Daten bezeichnet. Wie Gleichung 14 veranschaulicht, stellt der geschätzte Prognosefehler $\widehat{PE}$ eine Funktion dar, für deren Berechnung als Input das Prognosemodell M, die In-Set-Daten D_{in} und das Modellvalidierungsverfahren V benötigt werden. Der Output von $\widehat{PE}$ ist der erwartete Fehler des gewählten Prognosemodells. Gemäß dem Modellselektionskriterium wird jenes Prognosemodell M^* gewählt, dass den erwarteten Fehler minimiert. Im Idealfall repräsentiert M^* auch das Prognosemodell, dass den kleinsten, tatsächlichen Prognosefehler PE besitzt [CTS21, S.5].

Gewünscht ist daher, dass $\widehat{PE}$ so genau wie möglich den tatsächlichen Prognosefehler PE approximiert, da nur dann davon ausgegangen werden kann, dass das gewählte Prognosemodell M^* auch das beste Modell unter Verwendung von zukünftigen Daten darstellt. Hierfür wird jedoch eine verlässliche und genau Bewertung der Performance eines gegeben Prognosemodells benötigt. Anders formuliert, wird ein Modellvalidierungsverfahren benötigt, das robuste Abschätzungen des Prognosefehlers bietet. Die Wahl des richtigen Modellvalidierungsverfahrens ist daher in der Modellselektion bei Prognosen entscheidend.

Zwei typische validierungsbasierte Modellselektionsverfahren sind das Holdout-Verfahren und die Kreuzvalidierung [CTM20, S.2]. Im anschließenden Kapitel erfolgt eine detaillierte Beschreibung dieser beiden Modellvalidierungsverfahren.

4 Validierungsbasierte Modellselektionsverfahren

Um die Prognosefähigkeit eines Modells zu beurteilen, kann das Holdout- oder das Kreuzvalidierungsverfahren angewendet werden. Bei beiden Verfahren wird der Datensatz in zwei Teile geteilt. Ein Teil des Datensatzes wird zur Kalibrierung des Modells verwendet und der zweite Teil des Datensatz für die Überprüfung der Prognosefähigkeit. Je nach gewählter Modellvalidierungsmethode werden die Daten des ursprünglichen Datensatzes unterschiedlich aufgeteilt [Ku07, S.408]. Im folgenden Abschnitt wird ausgeführt, wie die Modellselektion nach dem Holdout-Verfahren erfolgt. Anschließend daran werden die unterschiedlichen Varianten der Kreuzvalidierung beschrieben. Bei beiden Verfahren wird ebenfalls auf die Vor- und Nachteile eingegangen.

4.1 Holdout-Verfahren

Wie in den frühen 30er Jahren von Selmer C. Larson festgestellt wurde, führt das Trainieren eines Modells und das Bewerten seiner statistischen Prognosefähigkeit anhand derselben Daten zu einem überoptimistischen Ergebnis. Das Holdout-Verfahren wurde entwickelt, um dieses Problem zu beheben, ausgehend von der Entwicklung, dass die Überprüfung des Prognosemodells mit den Testdaten erfolgt. Die Testdaten werden bei der Kalibrierung des Modells nicht verwendet und dienen ausschließlich zur Schätzung der Leistung des Prognosemodells. Dieses methodische Vorgehen wird auch als out-of-sample Validierung bezeichnet [AC10, S. 52].

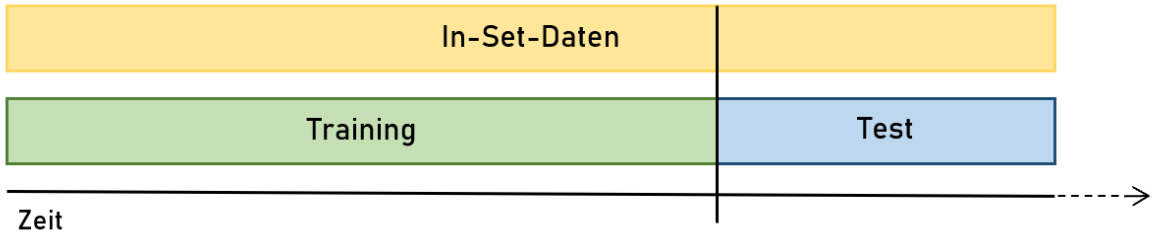

Abbildung 3: Prozess des Holdout-Verfahrens: Einteilung in Trainings- und Testdaten unter der Verwendung von Zeitreihendaten, in Anlehnung an [CTM20, S.3].

Abbildung 3 demonstriert den Prozess des Holdout-Verfahrens. Hierbei wird der ursprüngliche Datensatz in zwei disjunkte Teildatensätze aufgeteilt: (i) einem Trainingsdatensatz, der zur Schätzung der Modellparameter verwendet wird und (ii) einem Testdatensatz, mit Hilfe dessen die Prognosefähigkeit des Modells überprüft wird. Der Testdatensatz entspricht der Rolle neuer Daten und wird daher explizit bei der

Kalibrierung des Modells nicht eingesetzt. Die Überprüfung der Prognosegüte des Modells erfolgt, indem die geschätzten Werte (anhand der Trainingsdaten) mit den tatsächlichen Beobachtungen (Testdaten) verglichen werden. Bei der Aufteilung des Datensatzes muss beachtet werden, dass sowohl die Trainings- als auch die Testdaten repräsentativ für den Gesamtdatensatz sind. Zeitreihendaten stellen eine Besonderheit dar, da es sich hier nicht um unabhängige und identisch verteilte (u.i.v.) Zufallsvariablen handelt. Bei Zeitreihendaten sollte der Datensatz daher nicht zufällig, sondern nach der Zeit aufgeteilt werden [Ku07, S.408].

Eine typische validierungsbasierte Modellselektion mit Hilfe des Holdout-Verfahrens beinhaltet folgende vier Schritte:

1) Die Gesamtdaten werden in die Trainings- und Testdaten aufgeteilt. Die Zerlegung erfolgt meist im Verhältnis von 80/20 oder 70/30. Das bedeutet, dass 80% (70%) der Gesamtdaten dem Trainingsdatensatz zugeordnet werden und die verbleibenden 20% (30%) dem Testdatensatz angehören.
2) Der Trainingsdatensatz wird verwendet um die Parameter des Prognosemodells zu schätzen. Dieser Vorgang wird auch als „Trainieren vom Modell“ bezeichnet.
3) Für die Überprüfung der Prognosefähigkeit des Modells kommt der Testdatensatz zur Anwendung. Der Prognosefehler wird anhand der prognostizierten und tatsächlich beobachteten Werte berechnet.
4) Es wird jenes Prognosemodell gewählt, dass den kleinsten, geschätzten Prognosefehler $\widehat{PE}$ besitzt [Ku07].

Der Vorteil des Holdout-Verfahrens ist, dass die Datenaufteilung nur einmal ausgeführt werden muss und diese Methode daher einen geringen Rechenaufwand besitzt. Die Bewertung der Prognosefähigkeit eines Modells kann jedoch einer höheren Varianz unterliegen als andere Modellvalidierungsverfahren. Dies liegt daran, dass die Bewertung stark davon abhängt, welche Datenpunkte in dem Trainingsdatensatz und welche in dem Testdatensatz landen. Die Schätzung des Prognosefehlers kann daher je nach Aufteilung deutlich voneinander abweichen [Sc23].

4.2 Kreuzvalidierungsverfahren

Die Kreuzvalidierung ist ebenfalls ein out-of-sample Validierungsverfahren und ähnelt in ihrem Ablauf dem Holdout-Verfahren. Im Gegensatz zum Holdout-Verfahren erfolgt die Datenaufteilung beim Kreuzvalidierungsverfahren nicht nur einmal, sondern wird k-Mal durchgeführt. Das Hauptinteresse an der Kreuzvalidierung liegt in der Universalität der Datenaufteilungsheuristik. Es wird lediglich angenommen, dass die Daten u.i.v sind. Daher kann die Kreuzvalidierung auf fast jeden Algorithmus in fast jedem Framework angewendet werden, wie beispielsweise bei Regressionen, Dichteschätzungen oder

Klassifikationen [AC10, S. 52]. Ein Anwendungsfall, wo der Datensatz nicht u.i.v. ist, sind Zeitreihen. Zeitreihendaten sind eine sortierte Folge von Daten, die in regelmäßigen Intervallen erfasst werden. Typisches Charaktermerkmal von Zeitreihen ist, dass die Daten abhängig voneinander sind [CTM20, S. 2]. Diese Abhängigkeit wirft theoretische Probleme für die Verwendung des standardmäßigen k-fachen Kreuzvalidierungsverfahrens als Modellvalidierungsverfahren in Zeitreihendaten auf. Daher wurden spezielle Kreuzvalidierungstechniken entwickelt, um den Fall abzudecken, in dem die Daten voneinander abhängig sind. Hierzu zählen die geblockte Kreuzvalidierung, die modifizierte Kreuzvalidierung und die hv-geblockte Kreuzvalidierung [BB11, S. 851]. Die k-fache Kreuzvalidierung kann daher als Standardprozedur bezeichnet werden. Die geblockte, modifizierte und die hv-geblockte Kreuzvalidierung sind Varianten die entwickelt wurden, um die zeitliche Abhängigkeit zwischen den Beobachtungen zu bewältigen.

Die folgenden Unterkapitel gehen näher auf die einzelnen Varianten der Kreuzvalidierung ein. Im speziellen sollen diese Unterkapitel Aufschluss auf die Fragestellung geben, wie die Aufteilung der Beobachtungen des Gesamtdatensatzes in Trainings- und Testdatensatz erfolgt. Denn dies ist der entscheidende Unterschied zwischen den einzelnen Kreuzvalidierungsverfahren. Kapitel 4.2.1 stellt die standardmäßige k-fache Kreuzvalidierung vor. Da dieses Verfahren aufgrund der theoretischen Mängel in der Theorie nicht gut für Zeitreihen geeignet ist, wurden abgewandelte Methoden der k-fachen Kreuzvalidierung entlang der Literatur vorgestellt. Diese für Zeitreihen angepasste Kreuzvalidierungsverfahren sind in Kapitel 4.2.2 bis 4.2.4 detailliert erklärt.

4.2.1 K-fache Kreuzvalidierung

Die Grundidee der k-fachen Kreuzvalidierung ist, dass der Gesamtdatensatz in k Blöcke geteilt wird. Jeder Block wird einmal als Testdatensatz verwendet und die übriggebliebenen $k-1$ Blöcke als Trainingsdatensatz, für die Kalibrierung des Prognosemodells, eingesetzt. Die Bewertung der Prognosefähigkeit des Modells erfolgt anhand des durchschnittlichen, geschätzten Prognosefehlers über die k-Blöcke hinweg [Ad23, S.5]. Abbildung 4 zeigt das Verfahren der k-fachen Kreuzvalidierung mit drei Blöcken. Die zwölf Beobachtungen sind als Symbole dargestellt, welche im Zuge der Kreuzvalidierung in drei Blöcke eingeteilt werden.

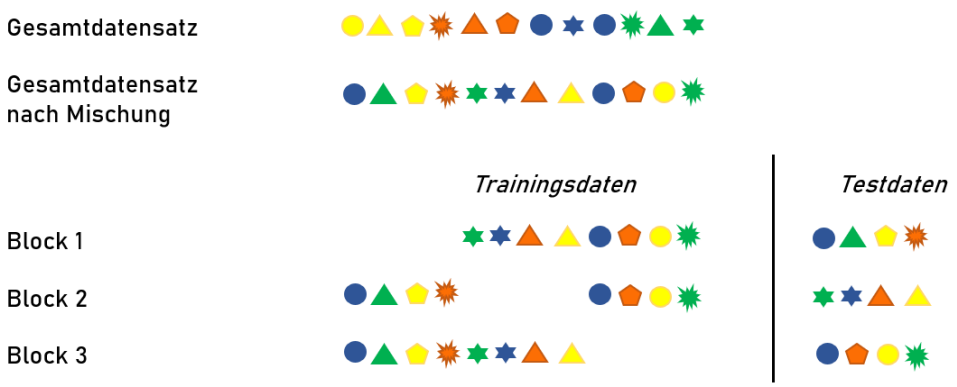

Abbildung 4: Beispielhafte Darstellung einer 3-fachen Kreuzvalidierung in Anlehnung an [KJ13, S. 71].

Ein typischer Ablauf der k-fachen Kreuzvalidierung als Modellselektionstechnik beinhaltet folgende fünf Schritte:

1) Der Gesamtdatensatz wird zufällig gemischt und in k gleich große Blöcke aufgeteilt. Jeder Block stellt daher eine Teilmenge der Gesamtdaten dar und beinhaltet $\frac{n}{k}$ zufällig zugeteilte Beobachtungen, wobei n die Gesamtanzahl an Beobachtungen angibt.
2) Block 1 kommt zur Anwendung: Es wird ein einfaches Holdout-Verfahren durchgeführt, wobei Block 1 den Testdatensatz widerspiegelt und die übrig gebliebenen $k-1$ Blöcke den Trainingsdatensatz bilden. Anhand des Trainingsdatensatzes wird das Prognosemodell kalibriert und zur Schätzung des Prognosefehlers werden die Testdaten (entspricht den Daten aus Block 1) herangezogen.
3) Schritt 2) wird für alle $K = 1,2,\dots,k$ Blöcke wiederholt. Am Ende dieses Schrittes sind folglich k Prognosefehlerschätzungen vorhanden.
4) Der Durchschnitt der erhaltenen k geschätzten Prognosefehlern wird berechnet.
5) Es wird jenes Prognosemodell gewählt, dass den kleinsten, durchschnittlichen, geschätzten Prognosefehler $\widehat{PE}$ besitzt [HTF08, S.242].

Die Wahl der Anzahl an Blöcken k ist normalerweise fünf oder zehn, jedoch gibt es keine formale Regel hierfür. Mit steigender Größe von k, nimmt der Bias (statistische Verzerrung) des Validierungsverfahrens ab. In diesem Kontext ist der Bias die Differenz zwischen den geschätzten und den tatsächlichen Werten des Prognosefehlers [KJ13, S. 70]. Automatisch könnte hieraus die Schlussfolgerung gezogen werden, dass die k-fache Kreuzvalidierung umso genauer ist, je größer k ist. Im Sinne des Bias ist diese Schlussfolgerung auch richtig. Ist $k = n$, so ist der Bias des (erwarteten) Prognosefehlers annähernd bei null. Mit steigender Größe von k, steigt aber auch die Varianz der Kreuzvalidierung, da die gebildeten Trainingssätze dann weniger stark voneinander abweichen. Ein Prognosemodell mit hoher Varianz schenkt den Trainingsdaten zu viel Aufmerksamkeit, weshalb die Verallgemeinerung auf noch unbekannte Daten darunter

leiden kann. Im Sinne der Varianz ist daher die Schlussfolgerung falsch [HTF08, S.243]. Studien von Breiman, Friedman [BF97] und Kohavi [Ko95] zeigen, dass die fünf- oder zehn-fache Kreuzvalidierung einen guten Kompromiss zwischen Bias und Varianz darstellt.

4.2.2 Geblockte Kreuzvalidierung

Die geblockte Kreuzvalidierung [Sn88] ist sehr ähnlich zur standardmäßigen k-fachen Kreuzvalidierung. Der Unterschied besteht darin, dass der ursprüngliche Gesamtdatensatz nicht zufällig gemischt wird. Bei Zeitreihen entstehen dadurch k Blöcke zusammenhängender Beobachtungen. Die natürliche Reihenfolge der Beobachtungen wird innerhalb jedes Blocks beibehalten, aber über die Blöcke hinweg gebrochen [CTM20, S. 5]. Dieser Ansatz der Kreuzvalidierung ist in Abbildung 5 dargestellt. Der einzige Unterschied zu Abbildung 4 ist, dass kein zufälliges Mischen des Gesamtdatensatzes stattfindet. Ein typischer Ablauf der geblockten Kreuzvalidierung als Modellselektionstechnik setzt sich daher aus den Schritten 2) bis 5) der k-fachen Kreuzvalidierung zusammen.

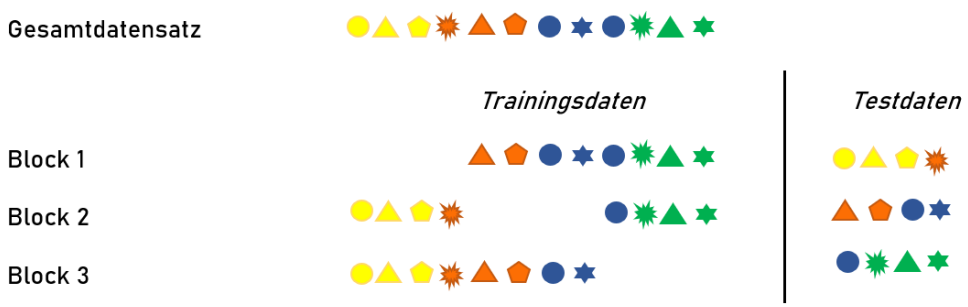

Abbildung 5: Beispielhafte Darstellung einer geblockten, 3-fachen Kreuzvalidierung.

4.2.3 Modifizierte Kreuzvalidierung

Aufgrund der Annahme in der standardmäßigen k-fachen Kreuzvalidierung, dass die Daten u.i.v. sind, stellen die Forscher Allan McQuarrie und Chih-Ling Tsai in ihrem publizierten Paper [MT98] eine abgewandelte Methode für Zeitreihendaten vor: die modifizierte Kreuzvalidierung. Voraussetzung für die modifizierten Kreuzvalidierung ist, dass es sich um eine stationäre Reihe handelt. Die Grundidee der modifizierten Kreuzvalidierung ist Beobachtungen aus dem Trainingsdatensatz zu entfernen, die mit dem Testdatensatz korrelieren, um so die Unabhängigkeit zwischen dem Trainings- und

den Testdatensatz zu gewährleisten. Die Korrelation wird hierbei anhand der Ordnung p des Autoregressiven Prozesses $AR(p)$ bestimmt [BCB14, S. 4].

Ein kurzes Beispiel soll zum besseren Verständnis dieser Datenselektion dienen: Angenommen die Beobachtung y_5 befindet sich nach Zuordnung im Testdatensatz und beim Prognosemodell handelt es sich um ein Autoregressives Modell mit zwei Lags $AR(2)$. Dies bedeutet, dass aus dem Trainingsdatensatz jene zwei Beobachtungen entfernt werden müssen, die im ursprünglichen Gesamtdatensatz die letzte und vorletzte Beobachtung vor y_5 sind, sprich y_4 und y_3. Die Beobachtungen y_4 und y_3 sind mit der Beobachtung y_5 korreliert und werden daher aus den Trainingsdaten entfernt.

Abbildung 6 veranschaulicht den Ansatz der modifizierten Kreuzvalidierung. Hier ist eine modifizierte, 3-fache Kreuzvalidierung dargestellt, mit der Annahme, dass es sich beim Prognosemodell um ein AR(1)-Modell handelt. Die Symbole stellen die Beobachtungen dar, welche im Zuge der Kreuzvalidierung in drei Blöcke eingeteilt werden. Jene Beobachtungen, die sich innerhalb der zeitlichen Distanz von einer (entspricht dem Lag p) Beobachtung eines Testdatenpunktes befinden, werden aus dem Trainingsdatensatz entfernt.

Abbildung 6: Beispielhafte Darstellung einer modifizierten, 3-fachen Kreuzvalidierung mit einem AR(1)-Prognosemodell.

Ein typischer Verlauf der modifizierten Kreuzvalidierung als Modellselektionstechnik setzt sich aus den folgenden fünf Schritten zusammen:

1) Der Gesamtdatensatz wird zufällig gemischt und in k gleichgroße Blöcke aufgeteilt, ident zur k-fachen Kreuzvalidierung.
2) Block 1 kommt zur Anwendung: Wie auch bei der k-fachen Kreuzvalidierung bildet Block 1 den Trainingsdatensatz und die restlichen $k-1$ Blöcke den Trainingsdatensatz. Nun werden jene Beobachtungen aus den Trainingsdaten entfernt, die mit den Testdaten korrelieren. Anhand der übrig gebliebenen

Trainingsdaten werden die Parameter des Prognosemodells geschätzt und mit Hilfe der Trainingsdaten der Prognosefehler geschätzt.

3) Schritt 2) wird für alle $K = 1,2, \dots, k$ Blöcke wiederholt. Somit sind am Ende dieses Schrittes k Prognosefehlerschätzungen vorhanden.
4) Der Durchschnitt der erhaltenen k geschätzten Prognosefehlern wird ermittelt.
5) Es wird jenes Prognosemodell gewählt, dass den kleinsten, durchschnittlichen, geschätzten Prognosefehler $\widehat{PE}$ besitzt [HTF08, S.242].

Wie in Abbildung 6 zu erkennen ist, führt die modifizierte Kreuzvalidierung dazu, dass sehr viele Daten aus dem Trainingsdatensatz entfernt werden. Die Aussagekraft dieses Verfahrens ist daher eng mit der Länge des Gesamtdatensatzes, der Länge des Testdatensatzes und der Ordnung p verbunden [BCB14, S. 4].

4.2.4 Hv-geblockte Kreuzvalidierung

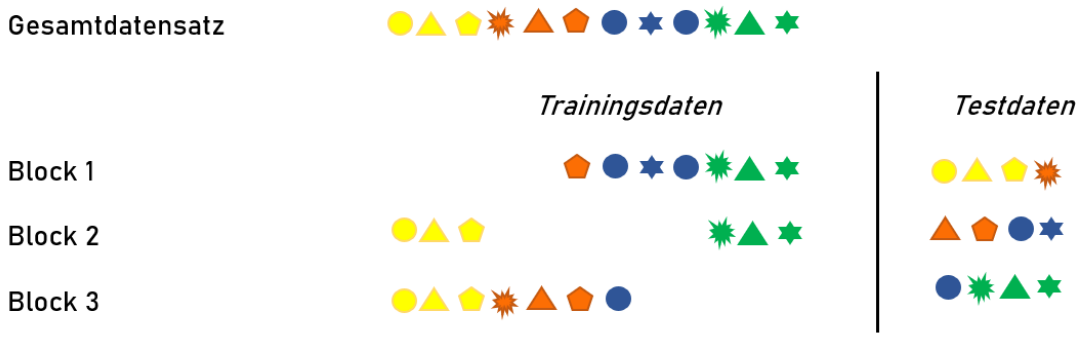

Abbildung 7: Beispielhafte Darstellung einer hv-geblockten, 3-fachen Kreuzvalidierung mit einem AR(1)-Prognosemodell.

Wird der Gesamtdatensatz zufällig gemischt und anschließend die mit dem Testdatensatz korrelierten Beobachtungen aus dem Trainingsdatensatz eliminiert, so kann dies zu einem erheblichen Datenverlust führen. In einigen Fälle kann die Datenmenge im Trainingssatz sogar so gering sein, dass eine Modellvalidierung nicht durchgeführt werden kann [BCB14, S.4]. Die hv-geblockte Kreuzvalidierung, vorgeschlagen von Jeffrey S. Racine [Ra00], bietet eine Lösung für dieses Problem. Um den starken Datenverlust entgegenzuwirken, erfolgt keine zufällige Mischung des Gesamtdatensatzes. Die ursprüngliche Reihenfolge der Beobachtungen wird beibehalten, wie bei der geblockten Kreuzvalidierung. Das Eliminieren von abhängigen Beobachtungen ist daher nur an den Grenzen des gewählten Blocks k erforderlich [BCB14, S.4]. Die Idee der hv-geblockten Kreuzvalidierung ist in Abbildung 7 dargestellt, wo die zwölf Beobachtungen erneut durch Symbole dargestellt sind. Es ist erkennbar, dass im Vergleich zur modifizierten Kreuzvalidierung (siehe Abbildung 6) deutlich mehr Beobachtungen im Trainingsdatensatz enthalten sind.

Das Verfahren der hv-geblockten Kreuzvalidierung ist beinahe ident zu jenem von der modifizierten Kreuzvalidierung. Einziger Unterschied ist, dass der Gesamtdatensatz nicht gemischt wird. Ein typischer Ablauf der hv-geblockten Kreuzvalidierung beinhaltet die Schritte 2) bis 5) der modifizierte Kreuzvalidierung.

5 Vor- und Nachteile des Kreuzvalidierungsverfahrens

Sind ausreichend Daten vorhanden, so stellt die Holdout-Methode eine einfache, rechnerisch kostengünstige Variante dar, um eine Modellselektion bei Prognosen durchzuführen. Ist die Gesamtdatenmenge begrenzt, so bietet das Holdout-Verfahren möglicherweise keine aussagekräftigen, robusten Ergebnisse [BCB14, S. 2]. Ein Vorteil der Kreuzvalidierung ist, dass die Daten vollständig ausgenützt werden. Insbesondere wenn die Datenlänge kurz ist, ist diese Eigenschaft ein entscheidender Vorteil. Dementsprechend wird die Kreuzvalidierung gegenüber dem Holdout-Verfahren vor allem dann präferiert, wenn der Datensatz wenige Beobachtungen beinhaltet. Der Vorteil, dass das Kreuzvalidierungsverfahren den Datensatz voll ausnützt, liegt insbesondere in der Eigenschaft, dass bei diesem Verfahren jeder Datenpunkt genau einmal in dem Testdatensatz und $k-1$ mal in dem Trainingsdatensatz enthalten sein muss. Dadurch ist es weniger entscheidend, wie der ursprüngliche Datensatz aufgeteilt wird. Die Kreuzvalidierung ermöglicht, dass die geschätzte Fehlerprognose nicht von besonderen Ereignissen innerhalb der Zeitreihe abhängig ist. Die Wahrscheinlichkeit, dass der geschätzte Prognosefehler dem tatsächlichen Prognosefehler entspricht, steigt dementsprechend [BB12, S. 193].

Ein Nachteil der Kreuzvalidierung ist die Rechenintensivität, da das Holdout-Verfahren praktisch k-mal durchgeführt wird. Bei einigen Prognosemodellen kann sich dies, aufgrund der benötigten Rechenleistung, als problematisch erweisen [Bi06, S. 33]. Hinzu kommt ein theoretischer Nachteil des Kreuzvalidierungsverfahrens im Rahmen von Zeitreihenprognosen: Eine grundlegende Annahme des Kreuzvalidierungsverfahrens ist, dass die Daten u.i.v. sind [BB12, S.193]. Zeitreihen halten diese Annahme nicht ein, da eine Autokorrelation zwischen den Daten besteht. Zeitreihendaten sind folglich nicht unabhängig. Um diesen theoretischen Mangel entgegenzuwirken, wurden die in Kapitel 4.2 erläuterten Varianten des Kreuzvalidierungsverfahren vorgestellt.

Zusammengefasst ist der größte Vorteile des Kreuzvalidierungsverfahrens, dass jede Beobachtung im Laufe der k-Wiederholungen einmal im Testdatensatz enthalten ist. Mit steigendem Wert von k, sinkt der Bias der Kreuzvalidierung. Einen Nachteil bei Zeitreihendaten stellt die Grundannahme dar, dass die Daten u.i.v. sind. Um diesen theoretischen Mangel entgegenzuwirken, wurden mehrere abgeänderte Verfahren der Kreuzvalidierung entwickelt (siehe Kapitel 4.2).

6 Anwendungsbeispiele des Kreuzvalidierungsverfahrens in der Praxis

Die Kreuzvalidierung spielt im Bereich Datenanalyse eine entscheidende Rolle, um die Prognosefähigkeit von Modellen und Methoden zu bewerten. Anhand des Kreuzvalidierungsverfahrens soll sichergestellt werden, dass die Vorhersagekraft des Modells oder der Methode auch für neue, nicht bekannte Daten hoch ist. Ein häufiges Einsatzgebiet der Kreuzvalidierung in der Praxis stellen Machine Learning Aufgaben dar. Die Aufgabe der Kreuzvalidierung im Machine Learning Workflow ist, die Leistung des Machine Learning Modells zu überprüfen, mit dem Ziel das geeignete Prognosemodell für das spezifische Vorhersageproblem auszuwählen [LY23].

Doch die Kreuzvalidierung ist nicht ausschließlich auf Machine Learning Aufgaben beschränkt. Das Validierungsverfahren wird auch in anderen Bereichen wie Predictive Analytics, Statistik und anderen Datenanalyse-Anwendungen eingesetzt. Im Prinzip findet die Kreuzvalidierung überall Anwendung, wo die Prognoseleistung von Modellen oder Methoden bewertet werden soll [SK11, S.560]. Die praktischen Anwendungsbeispiele sind dementsprechend in der Praxis sehr vielfältig, da Prognosen in vielen Bereichen benötigt werden. So werden beispielsweise Prognose in der Informatik verwendet, um Systemausfälle vorherzusagen, in der Industrie, um den Wartungsbedarf von Maschinen zu antizipieren und in der Wirtschaft, für die Schätzung der Ausfallwahrscheinlichkeit bei Kreditvergaben. Das Einsatzgebiet der Kreuzvalidierung ist dementsprechend sehr breit gefächert.

Ein Beispiel aus der Cloud-Computing-Branche soll das Aufgabengebiet und die Bedeutung des Kreuzvalidierungsverfahren in der Praxis näher darstellen. Konkret handelt es sich hierbei um die Anwendung bei einem Cloud-Service-Level-Agreement (Cloud-SLA). Ein Cloud-SLA ist eine rechtliche Vereinbarung zwischen einem Cloud-Service-Provider und einem Kunden, um sicherzustellen, dass ein Mindestmaß an Service aufrechterhalten wird. Die Verletzung einer Verpflichtung aus dem Cloud-SLA führt aus Sicht des Cloud-Service-Providers zu einer Geldstrafe und zu Reputationseinbußen, weshalb versucht wird dies zu vermeiden. Ein zentraler Aspekt für die Vermeidung ist die zukünftige Dienstgüte (Quality of Service) der Cloud-Parameter zu prognostizieren, um festzustellen, ob sie zu Verstößen führen. Hierfür werden verschiedene Prognosemethoden eingesetzt, deren Ergebnisse jedoch stark von den Inputdaten abhängen. Die Auswahl eines falschen Vorhersagealgorithmus kann zu einer Verletzung des Cloud-SLA führen.

Aus diesem Grund ist es entscheidend, die Genauigkeit jeder Prognosemethode zu überprüfen. Eine eingesetzte Möglichkeit zur Bewertung der Genauigkeit der Prognosemethode ist das Kreuzvalidierungsverfahrens [Hu18].

7 Vorstellung des Experimentes

Das durchgeführte Experiment dient dazu, die folgende Forschungsfrage empirisch zu beantworten: Welches Kreuzvalidierungsverfahren bietet die robustesten und aussagekräftigsten Fehlerabschätzungen in stationären Zeitreihendaten? Mit anderen Worten wird untersucht, welches Kreuzvalidierungsverfahren den genausten Schätzer für den tatsächlichen Prognosefehler bietet. Unter welchem Kreuzvalidierungsverfahren entspricht der geschätzte Prognosefehler $\widehat{PE}$ dem tatsächlichen Prognosefehler PE? In der Theorie herrscht Uneinigkeit über die Beantwortung dieser Forschungsfrage. Aus diesem Grund dient das folgende Experiment zur empirischen Untersuchung der Forschungsfrage. An dieser Stelle sei darauf hingewiesen, dass sich die Masterarbeit ausschließlich auf stationäre Zeitreihendaten beschränkt. Sind Zeitreihendaten stationär, so ändern sich ihre statistischen Eigenschaften über die Zeit nicht. Der Mittelwert und die Varianz der Zeitreihe bleibt über den gesamten Zeitraum hinweg konstant [BB12, S. 197]. Um eine breite Palette an Anwendungsszenarien abzudecken, werden die Experimente unter Verwendung von synthetischen und realen Datensätzen, sowie unter der Zuhilfenahme von verschiedenen Fehlermaßen durchgeführt.

Im Folgenden erfolgt eine Beschreibung des allgemeinen Aufbaus des Experimentes, der verwendeten Daten, des Vorhersagealgorithmus und des Prognosemodells, der drei Anwendungsszenarien, der verwendeten Fehlermaße, sowie der untersuchten Modellselektionsverfahren.. Die Experimente werden mit Hilfe der Open-Source Software *R* durchgeführt.

Im Anhang befindet sich ein kurzes numerisches Beispiel der Durchführung des Experimentes. Dieses veranschaulicht anhand eines Datensatzes mit einer Datenlänge von zehn, wie die standardmäßige 5-fache Kreuzvalidierung funktioniert. Das numerische Beispiel soll zu einem besseren Verständnis des Aufbaus und der Durchführung des Experimentes dienen.

7.1 Allgemeine Aufbau des Experimentes

Das experimentelle Design basiert auf dem Aufbau von Bergmeir, Benitez [BB12] und Bergmeir, Costantini, Benitez [BCB14]. Der vorhandene Zeitreihendatensatz wird zuerst in zwei Teile aufgeteilt: eine In-Set Menge und eine Out-Set Menge. Das In-Set enthält 80% der Daten, das Out-Set 20%. In der Praxis können wir keine Schätzungen an der Out-Set Menge vornehmen, da es zukünftige Beobachtungen darstellt. Daher wird für die

Schätzung des Prognosefehlers $\widehat{PE}$ ausschließlich auf das In-Set zurückgegriffen. Hiermit wird beurteilt, welches das beste Prognosemodell ist. Abbildung 8 veranschaulicht die Aufteilung des Gesamtdatensatzes grafisch. Hier ist deutlich zu sehen, dass im ersten Schritt der Gesamtdatensatz gemäß einer 80/20 Aufteilung in In- und Out-Set Daten geteilt wird. Anschließend werden die In-Set Daten gemäß des gewählten validierungsbasierten Modellselektionsverfahrens (siehe Kapitel 4) in einen Trainings- und Testdatensatz gesplittet und das Prognosemodelle evaluiert. Die Aufteilung des In-Sets erfolgt ebenfalls anhand eines 80/20 Verhältnisses. Das Ziel der In-Set Daten ist den Prognosefehler $\widehat{PE}$ zu schätzen, den ein Prognosemodell durch ungesehene, zukünftige Beobachtungen erleiden wird. Die Out-Set Daten werden verwendet, um den tatsächlichen Prognosefehler PE zu berechnen, den das Prognosemodell erlitten hat.

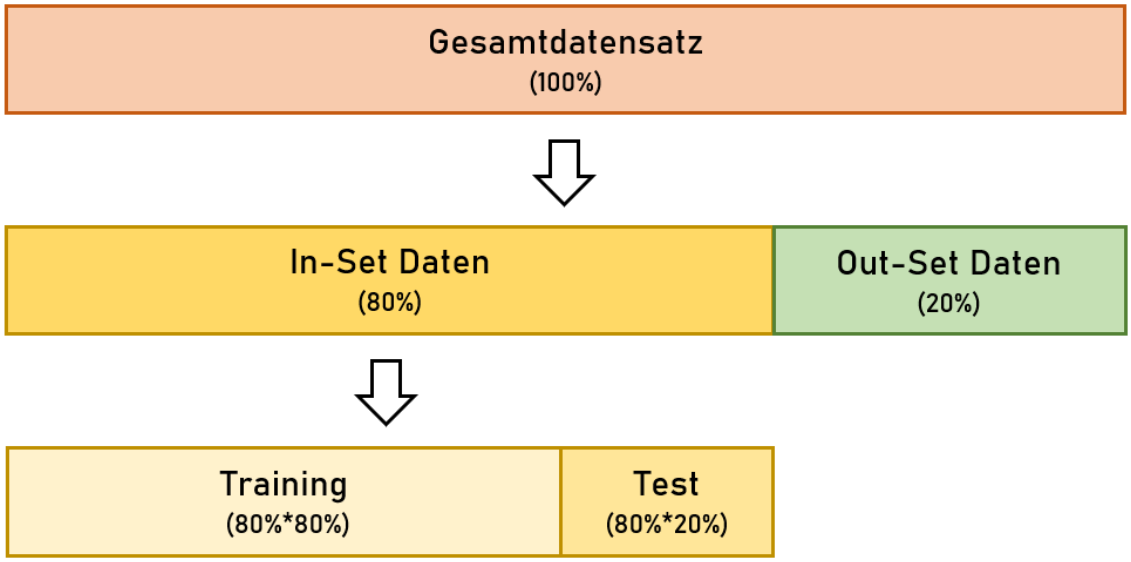

Abbildung 8: Aufteilung des Gesamtdatensatzes in In- und Out-Set Daten.

Das Ziel des validierungsbasierten Modellselektionsverfahrens ist die Leistung des Prognosemodells richtig zu bewerten. Dies bedeutet, dass jenes validierungsbasierte Modellselektionsverfahren die robustesten Ergebnisse liefert, wo der geschätzte Prognosefehler $\widehat{PE}$ so gut wie möglich den tatsächlichen Prognosefehler PE approximiert [CTM20, S.15]. In Kapitel 7.5 „Fehlermaße“ erfolgt eine Beschreibung wie diese Approximation quantifiziert wird.

7.2 Daten

Für die Untersuchung der Forschungsfrage werden sowohl synthetische, als auch reale Daten verwendet. Die synthetischen Daten werden anhand eines Autoregressiven Modells mit zwei Lags AR(2) und eines Moving Average-Modells der Ordnung eins MA(1) generiert. Das Ziel ist ein generelles Resultat zu erlangen. Mit anderen Worten, ein Ergebnis zu bekommen, welches unabhängig von einem bestimmten Datengenerierungsprozess (DGP) ist. Aus diesem Grund werden insgesamt 1.000 Monte Carlo Simulationen durchgeführt, sowohl bei der Generierung der AR(2) Daten, als auch bei den MA(1) Daten. Das hierfür verwendete stochastische Design basiert auf dem publizierten Paper von Bergmeir, Benitez [BB12], so dass für jede Monte Carlo Simulation neue Parameter für den DGP gewählt werden.

Ein synthetischer Datensatz enthält 198.000 Beobachtungen. Bei den realen Datensätzen, werden zwei Datensätze untersucht, die jeweils 3.648 Beobachtungen beinhalten. Anhand der Daten werden drei Anwendungsszenarien im Rahmen des Experimentes definiert, welche in Kapitel 7.4 näher erläutert werden.

7.2.1 Synthetische Daten

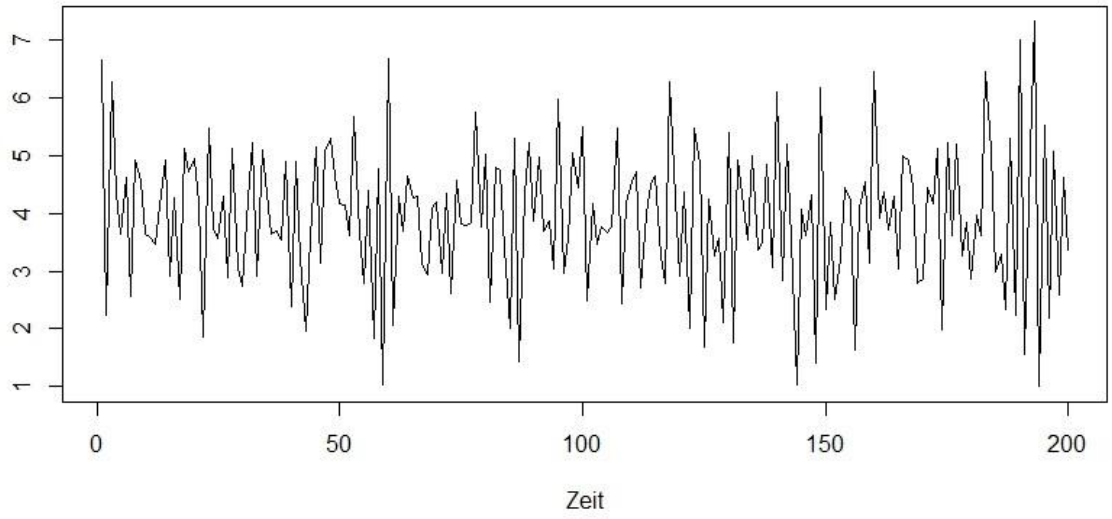

Abbildung 9: Beispiel eines stationären AR(2) Prozesses.

Für Anwendungsszenario eins werden Daten von einem AR(2) Prozess $y_t = \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \varepsilon_t$ mit einer Datenlänge t von $t = 200$ generiert. Abbildung 9 zeigt einen beispielhaften Prozess, welcher im Rahmen des Experimentes untersucht wird. Im Rahmen der durchgeführten Monte Carlo Simulationen werden unterschiedliche

Konstanten Φ_1, Φ_2 für den DGP gewählt. Die hierfür benötigten Parameter werden so gewählt, dass es sich immer um einen stationären AR(2) Prozess handelt. Dies bedeutet, dass die Koeffizienten Φ_1 und Φ_2 die drei Bedingungen erfüllen müssen, dass

- $\Phi_1 + \Phi_2 < 1$;
- $\Phi_1 - \Phi_2 < 2$; und
- $|\Phi_2| < 1$ [HA18].

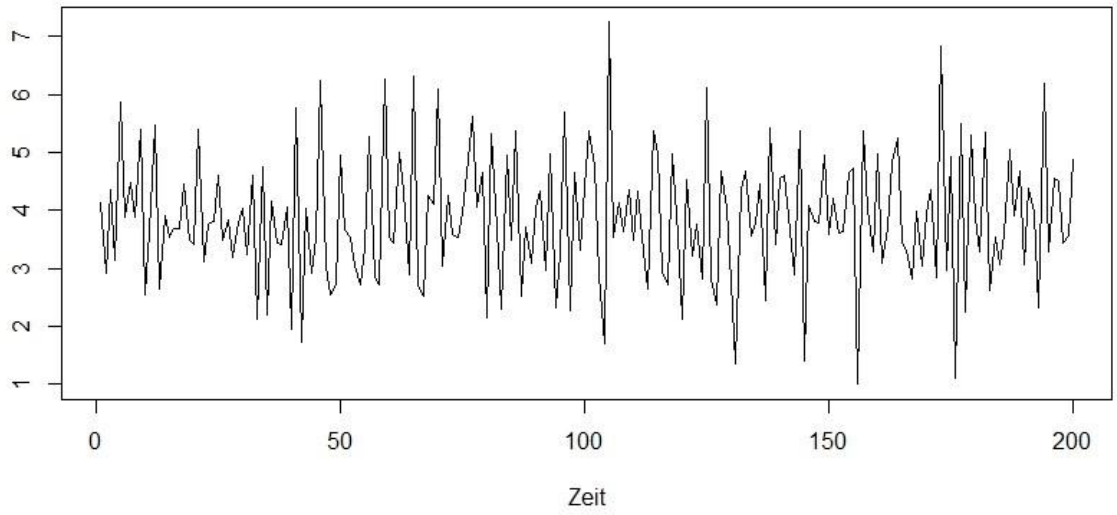

Abbildung 10: Beispiel eines stationären MA(1) Prozesses.

Anwendungsszenario zwei benötigt Daten von einem MA(1) Prozess $y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1}$ mit einer Datenlänge t von $t = 200$. Ident zu Anwendungsszenario eins werden 1000 Monte Carlo Simulationen mit verschiedenen Werten für θ_1 durchgeführt. Unabhängig von der Wahl von θ_1 ist ein MA(1)-Prozess immer stationär.

Abbildung 10 ist ein Zeitreihenplot und zeigt einen typischen, stationären MA(1)-Prozess. Es ist deutlich zu sehen, dass keine klaren Trends oder Muster im Zeitreihenprozess gegeben sind, sondern die Beobachtungen um den Wert 4 schwanken.

7.2.2 Reale Daten

Anwendungsszenario drei basiert auf realen, stationären Daten. Die hierfür benötigten Zeitreihendaten werden von dem Statistiker Rob Hyndman bereitgestellt und können unter der Website https://pkg.yangzhuoranyang.com/tsdl/articles/tsdl.html heruntergeladen werden. In der Mastarbeit werden die beiden stationären Zeitreihendaten „Daily maximum temperatures in Melbourne“ und „Daily minimum temperatures in Melbourne“ verwendet. Für Anwendungsszenario drei werden die Zeitreihendaten in eine Liste von Data Frames mit jeweils 200 Zeilen aufgeteilt.[2]

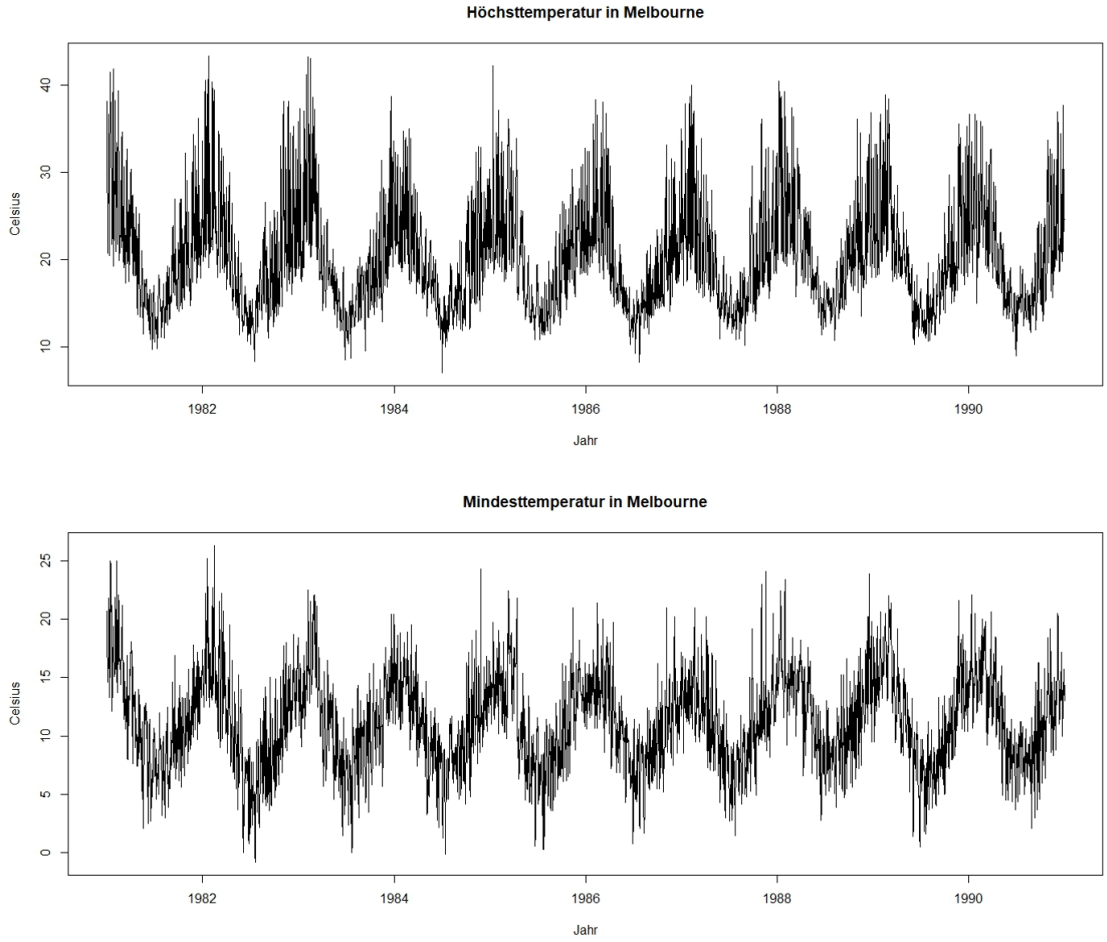

Abbildung 11: Tägliche Höchst- und Mindesttemperatur in Celsius in Melbourne.

Abbildung 11 veranschaulicht die tägliche Temperatur in Celsius in Melbourne von Jänner 1981 bis Dezember 1990. Der obere Subplot zeigt die Höchsttemperatur, während der untere Subplot die Mindesttemperatur darstellt. Bei beiden Zeitreihenprozessen ist das zugrunde liegende Zeitreihenmodell nicht bekannt. Es ist ausschließlich bekannt,

[2] Die 200 Zeilen spiegeln die Datenlänge von 200 wider.

dass der Prozess stationär ist. Auffällig sind die saisonalen Muster in den Daten. In den Monaten von Oktober bis März sind die Temperaturen deutlich höher, als in den Monaten von April bis September. Die Höchsttemperatur in Melbourne schwankt zwischen 7 und 43.3 Celsius. Die Mindesttemperatur zwischen -0.8 und 26.3 Celsius. Bei beiden Subplots gibt es keine offensichtlichen Ausreißer oder fehlende Werte in der Zeitreihe. Dies weist darauf hin, dass der Datensatz zuverlässig und konsistent ist.

7.3 Vorhersagealgorithmus und Prognosemodell

In den Experimenten wird eine One-Step-Ahead-Vorhersage berücksichtigt. Das Prognosemodell ist eine multiple lineare Regression der Zeitreihe auf sich selbst mit zwei Regressoren, der Form:

$$y_{t+1} = \beta_0 + \beta_1 y_t + \beta_2 y_{t-1}. \tag{15}$$

β_0 ist eine Konstante und β_1, β_2 sind die Regressionskoeffizienten der unabhängigen Variablen y_t und y_{t-1}. y_t stellt den aktuellen Wert zum Zeitpunkt t dar, y_{t-1} ist der Wert einen Zeitschritt vor t. Anhand dieser beiden Regressoren soll der zukünftige Wert der Zeitreihe y_{t+1} prognostiziert werden. Die benötigten Parameter werden mit Hilfe der Methode der kleinsten Quadrate (OLS-Methode) gefunden.

Die Wahl des Prognosemodells basiert auf den Referenzen [BB12, CTM20] und beruht auf der Tatsache, dass der synthetischen DGP in Anwendungsszenario eins einem AR(2)-Prozess folgt. AR-Modelle basieren auf der Idee, dass der aktuelle Wert der Zeitreihe y_t als lineare Funktion von p vergangenen Werten $y_{t-1}, y_{t-2}, \ldots, y_{t-p}$ erklärt werden kann, wobei p als Lag bezeichnet wird und die Anzahl der Schritte in der Vergangenheit bestimmt. Der vorhandene, lineare Zusammenhang in AR-Modellen lässt darauf schließen, dass die Vorhersage einer Beobachtung von einem AR(2)-Modell der Form $y_t = \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \varepsilon_t$ durch eine lineare Regression erfolgen kann:

$$y_{t+1} = c + \Phi_1 y_t + \Phi_2 y_{t-1}, \tag{16}$$

wo y_{t+1} die Prognose basierend auf den beobachteten Daten $y_1, y_2, \ldots, y_t$ darstellt [SS11, S. 84].

Um die Methode der kleinsten Quadrate bei AR-Modellen anzuwenden, müssen Zeitreihen durch einen Einbettungsschritt vorverarbeitet werden [BB12, S.197]. Tabelle 1 zeigt ein numerisches Beispiel einer eingebetteten Zeitreihe mit Dimension fünf. Jede Zeile beinhaltet einen Zielwert, der anhand der vergangenen fünf Beobachtungen (Lag 1

bis 5) vorhergesagt werden soll. Im Zuge der Masterarbeit wird eine eingebettete Zeitreihe der Dimension zwei verwendet, da das Prognosemodell eine multiple lineare Regression mit zwei Regressoren ist. Ziel der eingebetteten Zeitreihen ist die passenden Koeffizienten der Gleichung 15 mit Hilfe der Methode der kleinsten Quadrate zu finden.

Datum	Zielwert	Lag 1	Lag 2	Lag 3	Lag 4	Lag 5
1981-01-04	34.5	32.4	38.1	36.5	34.5	31.1
1981-01-05	20.7	34.5	32.4	38.1	36.5	34.5
1981-01-06	21.5	20.7	34.5	32.4	38.1	36.5
1981-01-07	23.1	21.5	20.7	34.5	32.4	38.1
1981-01-08	29.7	23.1	21.5	20.7	34.5	32.4
1981-01-09	36.6	29.7	23.1	21.5	20.7	34.5
1981-01-10	20.6	36.6	29.7	23.1	21.5	20.7

Tabelle 1: Numerisches Beispiel einer eingebetteten Zeitreihe mit Dimension fünf.

7.4 Anwendungsszenarien

In dem Experiment werden drei Anwendungsszenarien (AS) untersucht:

1. **AS.1:** Die generierten, synthetischen Daten stammen von einem stationären AR(2) Prozess.

Der Zweck von AS.1 ist zu veranschaulichen, wie sich die unterschiedlichen Kreuzvalidierungsverfahren verhalten, wenn ein geeignetes Prognosemodell (multiple lineare Regression mit zwei Regressoren) für die Vorhersage verwendet wird.

2. **AS.2:** Die generierten, synthetischen Daten stammen von einem MA(1) Prozess.

AS.2 demonstriert eine Situation, in der die generierte Zeitreihe nicht optimal anhand des Prognosemodells vorhergesagt werden kann. In der Praxis reicht normalerweise eine relativ geringe Anzahl an AR-Lags aus, um die Daten von einem MA(1) Prozess zu modellieren [BB12, S. 204]. Das verwendete Prognosemodell kann deshalb einigermaßen gut an die Daten angepasst werden und so eine Vorhersage getroffen werden.

3. **AS.3:** Es werden reale, stationäre Daten verwendet.

AS.3 spiegelt eine Situation wider, in der ein schlecht spezifiziertes Modell betrachtet wird. Die Temperaturdaten (siehe Abbildung 11) zeigen eine starke saisonale Komponente, weshalb das gewählte Prognosemodell falsch ist. Das Ziel ist zu untersuchen, wie sich die unterschiedlichen Kreuzvalidierungsverfahren verhalten, wenn das geeignete Prognosemodell nicht bekannt ist.

7.5 Fehlermaße

Die zu untersuchende Frage ist, wie gut der geschätzte Prognosefehler $\widehat{PE}$ den tatsächlichen Prognosefehler PE approximiert. Um diese Frage zu beantworten, erfolgt eine Messung des Fehlers zwischen $\widehat{PE}$ und PE über alle Datenreihen im Experiment. Es gibt viele verschiedene Fehlermaße die hierfür herangezogen werden können. Im Rahmen dieser Masterarbeit werden zwei verschiedene Gruppen an Fehlermaßen herangezogen: Relative Fehler und Skalenabhängige Metriken.

Für die Verlustfunktion wird der Root Mean Squared Error (RMSE) als Fehlermaß verwendet.

7.5.1 Relative Fehler

Bei dem relativen Fehlermaß wird der tatsächliche Prognosefehler PE in Relation zum geschätzten Prognosefehler $\widehat{PE}$ gesetzt [Ri21]. Der relative Fehler RE ist definiert als:

$$RE_i = \frac{PE_i}{\widehat{PE}_i}, \tag{17}$$

wobei i die jeweilig durchgeführte Monte Carlo Simulation bezeichnet.

7.5.2 Skalenabhängige Fehlermaße

Skalenabhängig bedeutet, dass das Fehlermaß in den Einheiten der zugrunde liegenden Daten ausgedrückt wird (z.B.: Celsius, Euro oder Liter). Diese Gruppe an Fehlermaß eignet sich daher nicht, wenn Zeitreihen mit verschiedenen Einheiten verglichen werden. Dies ist in der vorliegenden Masterarbeit nicht der Fall, weshalb diese theoretische Einschränkung kein Problem darstellt. Der Hauptvorteil skalenabhängiger Fehlermaße ist, dass sie einfach zu berechnen sind und ein intuitives, leicht vergleichbares Ergebnis liefern [Ri22].

Für die Untersuchung der Forschungsfrage werden drei skalenabhängigen Fehlermaße verwendet:

- Mean Absolute Predictive Accuracy Error (MAPAE);
- Mean Predictive Accuracy Error (MPAE); und
- Root Mean Squared Error (RMSE).

Das Fehlermaß MAPAE wird berechnet, indem die absolute Differenz zwischen prognostizierten und tatsächlichen Werten gezogen wird, wobei $\widehat{PE}_i$ den erwarteten Wert und PE den tatsächlichen Wert darstellt. Die Anzahl an durchgeführten Monte Carlo Simulationen entspricht m. MAPAE berechnet die durchschnittliche Größe des Fehlers von einem Validierungsverfahren [Le21].

$$MAPAE = \frac{1}{m}\sum_{i=1}^{m}\left|\widehat{PE}_i - PE\right| \tag{18}$$

Das Fehlermaß MPAE ist definiert als die Differenz zwischen prognostizierten und tatsächlichen Werten für den Prognosefehler. Es gibt an, ob ein Validierungsverfahren den Prognosefehler über- oder unterschätzt [Le21].

$$MPAE = \frac{1}{m}\sum_{i=1}^{m}(\widehat{PE}_i - PE) \tag{19}$$

Beim Fehlermaß RMSE werden die Fehler quadriert, bevor der Durchschnitt ermittelt wird. Der RMSE gibt großen Fehlern daher ein relativ hohes Gewicht. Das Ergebnis dieses Fehlermaßes ist eine positive Zahl. Je niedriger der Wert des RMSE, desto besser ist die Genauigkeit des geschätzten Prognosefehlers $\widehat{PE}_i$ [Le21].

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\widehat{PE}_i - PE\right)^2} \tag{20}$$

7.6 Modellselektionsverfahren

Es werden folgende Kreuzvalidierungsverfahren verglichen:

- K-fache Kreuzvalidierung
- Geblockte Kreuzvalidierung
- Modifizierte Kreuzvalidierung
- Hv-geblockte Kreuzvalidierung

Bei allen Varianten der Kreuzvalidierung beträgt die Anzahl der Blöcke fünf.

Darüber hinaus erfolgt ein Vergleich mit dem Holdout-Verfahren. Das Holdout-Verfahren ist kein Kreuzvalidierungsverfahren, sondern repräsentiert das typische out-of-sample Verfahren, wo ausschließlich der letzte Teil der Zeitreihe für die Testung verwendet wird. Dieser Vergleich dient zur Überprüfung der Annahme von Bergmeir, Benitez [BB12], dass die k-fache Kreuzvalidierung das Holdout-Verfahren auch outperformt, wenn die Daten abhängig sind.

8 Ergebnisse des Experimentes

Das Ziel von Kreuzvalidierungsverfahren ist das verlässliche und genaue Bewerten der Performance von gegeben Prognosemodellen. Dieser Vorgang wird in jeder Machine Learning Aufgabe, als Bestandteil der Modellselektion, durchgeführt.

Die vorliegende Masterarbeit konzentriert sich auf die Schätzung des (tatsächlichen) Prognosefehlers von Modellen in stationären Zeitreihendaten. In der wissenschaftlichen Literatur wurden verschiedene Varianten der standardmäßigen k-fachen Kreuzvalidierung für Zeitreihendaten vorgestellt, um die theoretischen Mängel dieses Verfahrens zu umgehen. Hierzu zählen die geblockte Kreuzvalidierung, die modifizierte Kreuzvalidierung und die hv-geblockte Kreuzvalidierung (siehe Kapitel 4.2). Die Frage, welches Kreuzvalidierungsverfahren die robustesten Prognosefehlerabschätzungen bei stationären Zeitreihendaten bietet, konnte bislang in der Theorie nicht beantwortet werden. Der folgende Teil der Masterarbeit soll diese Frage, mithilfe des in Kapitel 7 vorgestellten Experimentes, empirisch beantworten.

Es folgt nun eine genaue Beschreibung der Ergebnisse des durchgeführten Experimentes. Die Auswertung der Ergebnisse basiert auf den in Kapitel 7.5 vorgestellten Fehlermaßen. Das bedeutet, dass sowohl der relative Fehler, als auch das skalenbasierte Fehlermaße für die Interpretation der Ergebnisse herangezogen werden.

Die Darstellung der Ergebnisse wird in zwei Unterkapitel aufgeteilt. Kapitel 8.1 beschäftigt sich mit den Resultaten, die auf den synthetischen Daten basieren und Kapitel 8.2 mit jenen Resultaten, die anhand der realen Daten erzielt wurden. Kapitel 8.3 fasst die erzielten Ergebnisse zusammen.

8.1 Resultate basierend auf synthetischen Daten

Ein Teil der Ergebnisse von AS.1, wo ein AR(2) Prozess als DGP verwendet wird, ist in Abbildung 12 dargestellt. Der Boxplot zeigt die Verteilung der relativen Fehler-Daten von den fünf untersuchten validierungsbasierten Modellselektionsverfahren (k-fache Kreuzvalidierung, geblockte Kreuzvalidierung, modifizierte Kreuzvalidierung, hv-geblockte Kreuzvalidierung und die Holdout-Methode). Der relative Fehler ist definiert als der Quotient aus dem tatsächlichen Prognosefehler (Out-Set Fehler) und dem geschätzten Prognosefehler (In-Set Fehler). Wenn der relative Fehler größer als eins ist, so unterschätzt das Modellvalidierungsverfahren den Prognosefehler. Eine

Überschätzung des Prognosefehlers liegt vor, wenn der relative Fehler kleiner als eins ist. Sind die relativen Fehler gleichmäßig um eins verteilt und sind kaum Ausreißer vorhanden, so ist dies ein Indikator dafür, dass ein Validierungsverfahren robust ist.

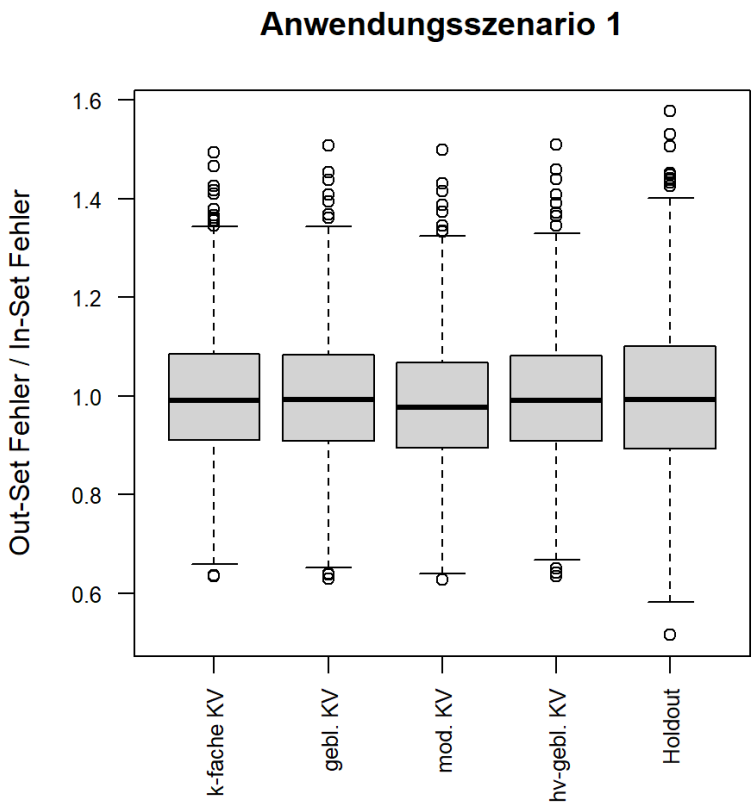

Abbildung 12: Boxplot zur Veranschaulichung der Ergebnisse von AS.1.

Der relative Fehler ist auf der y-Achse des Diagramms abgebildet. Auf den ersten Blick lassen sich zwischen den untersuchten validierungsbasierten Modellselektionsverfahren keine systematischen Unterschiede feststellen. Ein Bias, in Form einer systematischen Über- oder Unterschätzung des tatsächlichen Prognosefehlers, liegt bei keinem Verfahren vor, da alle Boxen um den Wert eins zentriert sind. Dies ist etwas überraschend, da aufgrund der vorhandenen Abhängigkeiten in den Zeitreihendaten, eine Unterschätzung des Fehlers bei den Kreuzvalidierungsverfahren zu erwarten gewesen wäre [BB12, S. 192]. Bei näherer Betrachtung ist erkennbar, dass einzig alleine das modifizierte Kreuzvalidierungsverfahren den tatsächlichen Prognosefehler minimal überschätzt, da hier der Median knapp unter eins ist (der genaue Wert des Median ist 0.9767). Auffällig ist, dass der Boxplot der Holdout-Methode einen größeren Interquartilsabstand und eine größere Spannweite aufweist, als jene Boxplots von den Kreuzvalidierungsverfahren. Dies bedeutet, dass die Holdout-Methode stärker gestreute und folglich weniger robuste Ergebnisse liefert.

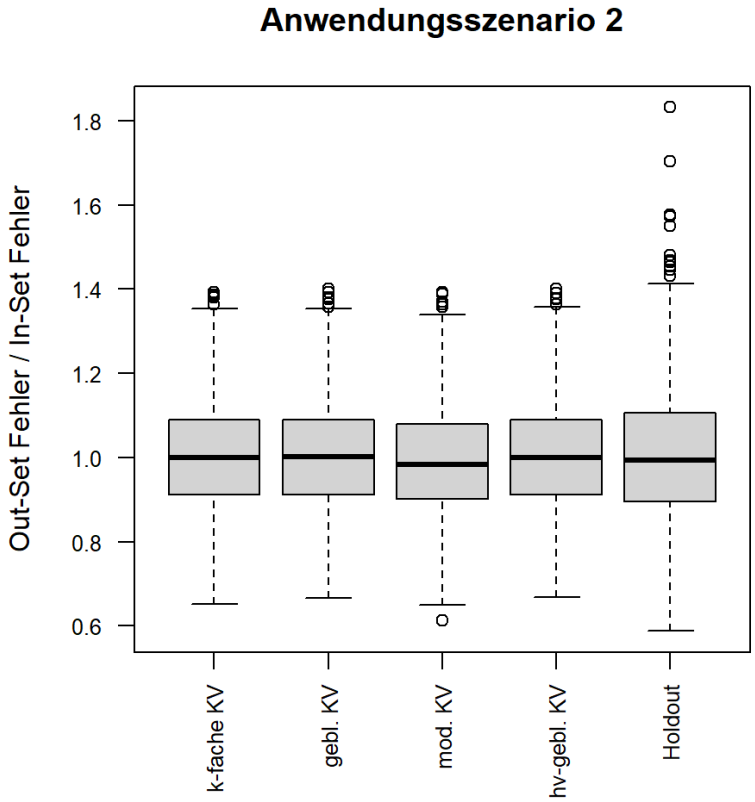

Abbildung 13: Boxplot zur Veranschaulichung der Ergebnisse von AS.2.

Abbildung 13 bezieht sich auf AS.2 und spiegelt ebenfalls die Verteilung des relativen Fehlers der untersuchten Modellvalidierungsverfahren in Form von Boxplots wider. In AS.2 wird ein MA(1) Prozess als DGP verwendet. Ziel ist zu untersuchen, wie sich die unterschiedlichen Verfahren verhalten, wenn kein optimales Prognosemodell für die Vorhersage verwendet wird. Wie in Abbildung 13 zu sehen ist, sind die erzielten Ergebnisse ähnlich zu jenen von AS.1, dargestellt in Abbildung 12. Es liegt keine systematische Über- oder Unterschätzung des tatsächlichen Prognosefehlers vor und die Holdout-Methode liefert erneut im Vergleich zu den Kreuzvalidierungsverfahren deutlich stärker gestreute Ergebnisse. Auffallend ist, dass die Ausreißer bei allen Validierungsverfahren, mit Ausnahme eines Fehlers beim modifizierten Kreuzverfahren, über dem oberen Whisker liegen. Dies deutet darauf hin, dass die Verteilungen etwas rechtsschief sind. Der Mittelwert ist in diesem Fall größer als der Median. Darüber hinaus stechen die starken Ausreißer auf der y-Achse des Boxplots beim Holdout-Verfahren sofort ins Auge. Dies kann darauf hinweisen, dass das Holdout-Verfahren Schwierigkeiten hat, bestimmte Prognosefehler richtig zu schätzen. Starke Ausreißer sprechen bei der Schätzung von Prognosefehlern nicht für ein robustes Validierungsverfahren. Ein robustes Modellvalidierungsverfahren zeichnet sich durch eine gute

Vorhersagegenauigkeit des Prognosefehlers aus und minimiert das Auftreten von Ausreißern.

Sowohl Abbildung 12, als auch Abbildung 13 verdeutlicht, dass die Holdout-Methode stärker gestreute Ergebnisse liefert als das Kreuzvalidierungsverfahren. Die Stärke der Streuung ist ein Indikator für die Robustheit einer Methode. Je stärker die Streuung ist, desto größer ist das Risiko von Fehlern und Ungenauigkeiten in den Ergebnissen. Folglich liefern die synthetischen Daten einen Beweis dafür, dass die untersuchten Arten der Kreuzvalidierung robuster sind als das Holdout-Verfahren (gemäß dem relativen Fehler). Die höhere Robustheit der Kreuzvalidierungsverfahren ist auf die Tatsache zurückzuführen, dass die Datenaufteilung nicht nur einmal, sondern k-Mal durchgeführt wird. Das mögliche Risiko einer Über- oder Unteranpassung des Modells an bestimmte Daten wird dadurch minimiert [AC10, S. 52]. Im Gegensatz dazu erfolgt die Datenaufteilung und folglich auch die Anpassung des Prognosemodells bei der Holdout-Methode nur einmal, was zu größeren Schwankungen in der korrekten Schätzung des Prognosefehlers führen kann [SC23]. Interessant hierbei ist die Tatsache, dass die hier empirisch gewonnene Erkenntnis sich auf abhängige Daten bezieht. Diese Erkenntnis unterstützt das Ergebnis von Bergmeir, Hyndman, Koo [BHK18]. In ihrem veröffentlichen Paper liefern die Wissenschaftler einen mathematischen Beweis, dass die klassische k-fache Kreuzvalidierung auch bei stationären Zeitreihendaten eingesetzt werden kann, trotz Nichteinhaltung der Annahme, dass die Daten u.i.v. sind.

Das relative Fehlermaß zeigt, dass das Kreuzvalidierungsverfahren robuster ist, als das klassische Holdout-Verfahren. Welche spezifische Art des Kreuzvalidierungsverfahren jedoch die aussagekräftigsten und robustesten Fehlerabschätzungen in stationären Zeitreihendaten liefert, kann aus Abbildung 12 und 13 nicht entnommen werden. In beiden Anwendungsszenarien sind die Boxplots der vier Kreuzvalidierungsverfahren beinahe ident. Eine signifikante Aussage kann anhand des relativen Fehlermaßes nicht getroffen werden. Aus diesem Grund werden auch drei skalenabhängige Fehlermaße (RMSE, MPAE und MAPAE) für die Beantwortung der Forschungsfrage herangezogen. Tabelle 2 zeigt das Ergebnis unter Verwendung der skalenabhängigen Fehlermaße.

Anwendungsszenario 1			
Modellselektionsverfahren	**RMSE**	**MAPAE**	**MPAE**
k-fache Kreuzvalidierung	0.1289	0.1031	0.0026
Geblockte Kreuzvalidierung	0.1287	0.1026	0.0033
Modifizierte Kreuzvalidierung	0.1309	0.1052	0.0178
Hv-geblockte Kreuzvalidierung	0.1290	0.1028	0.0041
Holdout	0.1561	0.1239	0.0110
Anwendungsszenario 2			
Modellselektionsverfahren	**RMSE**	**MAPAE**	**MPAE**
k-fache Kreuzvalidierung	0.1304	0.1045	-0.0022
Geblockte Kreuzvalidierung	0.1300	0.1040	-0.0033
Modifizierte Kreuzvalidierung	0.1340	0.1069	0.0121
Hv-geblockte Kreuzvalidierung	0.1297	0.1038	-0.0032
Holdout	0.1606	0.1281	0.0040

Tabelle 2: Resultate von AS.1 und AS.2 anhand der skalenbasierten Fehlermaße.

Der obere Teil von Tabelle 2 zeigt die Ergebnisse für AS.1, wo ein AR(2)-Prozesse als DGP verwendet wird. Es ist erkennbar, dass der RMSE bei allen Kreuzvalidierungsverfahren um den Wert 0.13 schwankt und beim Holdout-Verfahren einen deutlich höheren Wert von 0.1561 aufweist. Je niedriger der Wert des RMSE, desto besser ist die Genauigkeit des geschätzten Prognosefehlers. Das erzielte Ergebnis unterstützt die erzielte Erkenntnis anhand des relativen Fehlers, dass das Kreuzvalidierungsverfahren bei Zeitreihendaten robustere Fehlerschätzungen erzielt, als das Holdout-Verfahren. Das modifizierte Kreuzvalidierungsverfahren schneidet unter den Kreuzvalidierungsverfahren am schlechtesten ab. Eine mögliche Ursache hierfür ist, dass die Modellanpassung anhand deutlich weniger Daten erfolgt, da bei diesem Verfahren sehr viele Beobachtungen aus dem Trainingsdatensatz entfernt werden. In Kapitel 9 wird überprüft, wie sich dieser Sachverhalt ändert, wenn die Datenlänge von 200 auf 500

erhöht wird. Gemäß dem RMSE, erzielt die geblockte Kreuzvalidierung das beste Ergebnis.

Beim Fehlermaß MAPAE liegen die Fehler zwischen 0.1026 und 0.1239. MAPAE gibt die durchschnittliche Größe des Fehlers von einem Validierungsverfahren an. Die geblockte Kreuzvalidierung erreicht erneut die beste Schätzung mit einem Wert von 0.1026, gefolgt von der hv-geblockten Kreuzvalidierung mit einem Fehlerwert von 0.1028. Das Holdout-Verfahren und die modifizierte Kreuzvalidierung schneiden mit absoluten Werten über 0.1050 deutlich schlechter ab.

Die Ergebnisse gelten auch für das Fehlermaß MPAE in einer ähnlichen Weise. MPAE wird verwendet, um zu untersuchen, ob ein Validierungsverfahren den Prognosefehler über- oder unterschätzt. Da alle Werte größer sind als null, deutet dies darauf hin, dass in AS.1. alle Validierungsverfahren im Durchschnitt den Prognosefehler überschätzen. Hier erzielen die geblockte- und die k-fache Kreuzvalidierung die niedrigsten Werte, wohingegen das Holdout-Verfahren und die modifizierte Kreuzvalidierung erneut die höchsten Fehlerwerte aufweisen. Im Gegensatz zum Fehlermaß MAPAE liegt der Wertebereich für MPAE zwischen 0.0026 und 0.0178.

Der untere Teil von Tabelle 2 illustriert die Ergebnisse für AS.2. Hier wird ein MA(1)-Prozesse als DGP verwendet. Bezüglich dem Fehlermaß MAPAE weist die hv-geblockte Kreuzvalidierung mit einem Wert von 0.1038 die beste Performance. Die Erkenntnis, dass die Kreuzvalidierungsverfahren genauere Fehlerschätzungen liefern als das Holdout-Verfahren, wird erneut bekräftigt. Während die Fehlerwerte der Kreuzvalidierungsverfahren bei etwa 0.10 liegen, weist die Holdout-Methode einen Fehlerwert von 0.1280 auf. In Bezug auf das Fehlermaß RMSE outperformt die hv-geblockte Kreuzvalidierung minimal die geblockte Kreuzvalidierung mit einem Fehlerwert von 0.1297 bzw. 0.1299. Ident zu AS.1 sind die modifizierte Kreuzvalidierung und das Holdout-Verfahren nicht konkurrenzfähig, wobei das Holdout-Verfahren mit einem Wert von 0.1606 am schlechtesten den Prognosefehler einschätzt. Interessant sind die Werte des Fehlermaßes MPAE, da die k-fache Kreuzvalidierung, die geblockte Kreuzvalidierung und die hv-geblockte Kreuzvalidierung negative Werte aufweisen. Dies bedeutet, dass in AS.2 der Prognosefehler von diesen Validierungsverfahren im Durchschnitt unterschätzt wird. Das Holdout-Verfahren und die modifizierte Kreuzvalidierung weisen durchschnittlich eine Überschätzung auf, da der Wert von MPAE positiv ist.

Die erzielten Ergebnisse anhand der synthetischen Daten bestätigen die Ergebnisse von [BKH18], dass das Kreuzvalidierungsverfahren bei Zeitreihendaten robustere Fehlerabschätzungen erzielt als das Holdout-Verfahren. Dieses Ergebnis ist unabhängig davon, ob das Prognosemodell für die Vorhersage optimal ist oder nicht. Unter der Gruppe der Kreuzvalidierungsverfahren hat die geblockte Kreuzvalidierung und die hv-geblockte Kreuzvalidierung eine sehr ähnliche, gute Performance. Ist das Prognosemodell für die Vorhersage optimal (siehe AS.1), so bietet gemäß den skalenabhängigen Fehlermaßen die geblockte Kreuzvalidierung die robustesten Fehlerabschätzungen. Wenn das Prognosemodell nicht ganz optimal ist (siehe AS.2) so erzielt die hv-geblockte Kreuzvalidierung den niedrigsten Fehler bei der Schätzung des tatsächlichen Prognosefehlers. Die Resultate zeigen ebenfalls, dass die standardmäßige k-fache Kreuzvalidierung trotz ihrer theoretischen Mängel, robuste Resultate in stationären Zeitreihendaten liefert. Das modifizierte Kreuzvalidierungsverfahren ist in beiden Anwendungsszenarien nicht konkurrenzfähig. Eine mögliche Ursache hierfür ist die deutlich niedrigere Anzahl an Daten im Trainingsdatensatz.

An dieser Stelle sei jedoch angemerkt, dass die erzielten Ergebnisse nicht auf alle Situationen oder Datensätze verallgemeinert werden können. Es wäre falsch aufgrund von AS.1 und AS.2 Schlüsse auf die Allgemeinheit zu zielen. Die hier erzielten Ergebnisse unterstützen lediglich bislang untersuchte Ergebnisse (siehe [BKH18]) und geben eine erste Antwort auf die Fragestellung, welches Kreuzvalidierungsverfahren die robustesten und aussagekräftigsten Fehlerabschätzungen liefert, wenn das Prognosemodell geeignet oder zumindest optimal ist.

8.2 Resultate basierend auf realen Daten

Für AS.3 werden echte Daten verwendet, welche die täglichen Höchst- und Mindesttemperaturen in Melbourne abbilden (siehe Abbildung 11). Beide Datensätze weisen saisonale Muster auf. Wenn auf der Südhalbkugel Sommer ist, so sind die Temperaturen in Melbourne deutlich höher als im Winter. Das gewählte Prognosemodell, eine lineare Regression mit zwei Regressoren, ist aufgrund fehlender Berücksichtigung der saisonalen Effekte in AS.3 daher nicht geeignet. Abbildung 14 zeigt den tatsächlichen Prognosefehler der Höchsttemperaturdaten in Melbourne über einen Zeitraum von 730 Tagen. Es ist erkennbar, dass im Falle einer Temperaturänderung, die Prognosefehler deutlich größer werden. Dies liegt mitunter daran, dass das gewählte Prognosemodell die saisonalen Effekte nicht beachtet. Das lineare Regressionsmodell ist daher kein gutes

Prognosemodell für die zugrundeliegenden Daten. Besser wäre ein saisonales Modell, wie beispielsweise SARIMA, für die Modellierung und Vorhersage der Temperaturdaten.

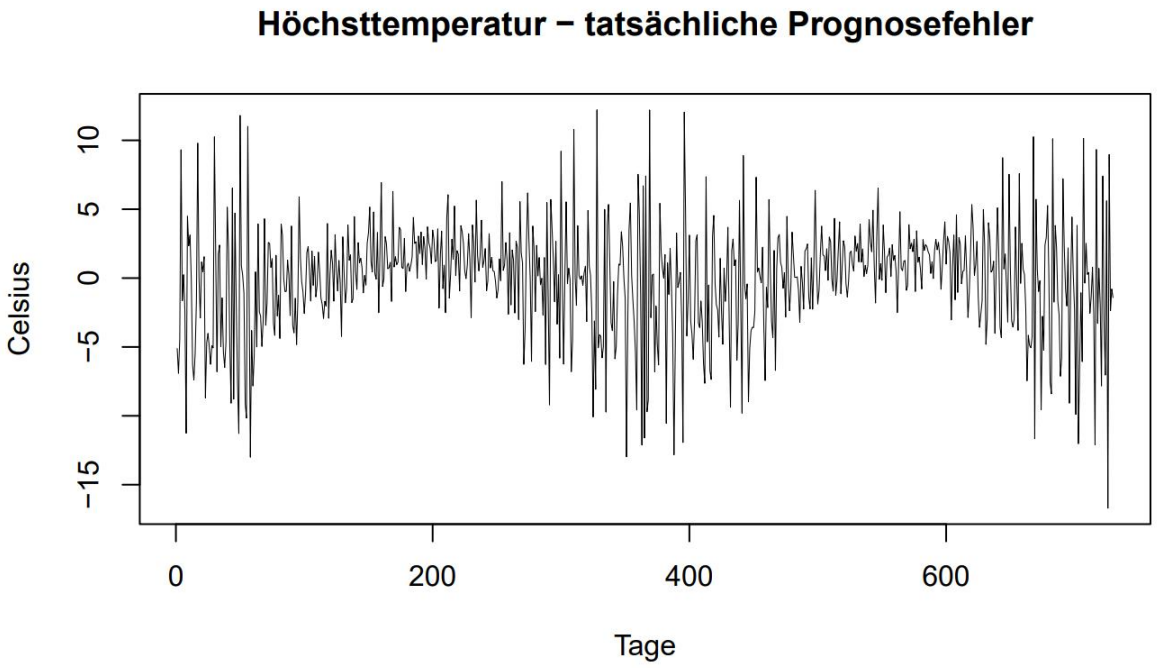

Abbildung 14: Darstellung des tatsächlichen Prognosefehlers (Out-Set Fehler).

Das Ziel dieser Arbeit ist jedoch nicht das passende Prognosemodell zu finden, sondern jenes Modellvalidierungsverfahren zu finden, dass die robustesten Prognosefehlerabschätzungen liefert. AS.3 stellt daher eine Situation dar, in der ein schlecht spezifiziertes Modell betrachtet wird. Der Zweck von AS.3 ist zu untersuchen, wie sich die validierungsbasierten Modellselektionsverfahren verhalten, wenn das geeignete Prognosemodell nicht bekannt ist und ein falsches Modell gewählt wird. Ist das gewählte Validierungserfahren auch robust, wenn das geeignet Prognosemodell nicht bekannt ist? Eine erste Antwort auf diese Fragestellung geben die Boxplots in Abbildung 15.

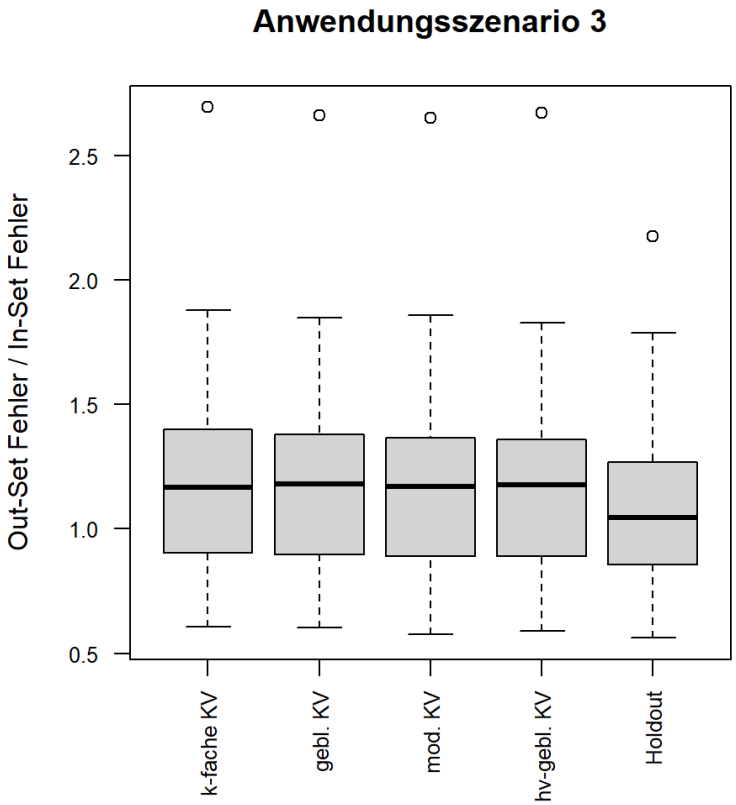

Abbildung 15: Boxplot zur Veranschaulichung der Ergebnisse von AS.3.

Ein deutlicher Unterschied zu AS.1 und AS.2 ist in Abbildung 15 gleich erkennbar: Das Holdout-Verfahren liefert in AS.3 die robustesten Ergebnisse, gemäß dem relativen Fehlermaß. Dies deutet darauf hin, dass bei einem schlecht spezifizierten Prognosemodell das klassische Holdout-Verfahren genauerer Fehlerabschätzungen in Zeitreihendaten bereitstellt, als die Gruppe der Kreuzvalidierungsverfahren. Abbildung 15 zeigt, dass die Box und der Median des Holdout-Verfahrens um den Wert eins zentriert ist. Bei den Kreuzvalidierungsverfahren ist dies nicht der Fall. Hier liegt ein Bias, im Sinne einer systematischen Unterschätzung des tatsächlichen Prognosefehlers, vor. Dieser Bias ist darauf zurückzuführen, dass der In-Set Fehler kleiner ist als der Out-Set Fehler. Das Kreuzvalidierungsverfahren neigt folglich dazu, den Prognosefehler tendenziell zu niedrig einzuschätzen. Diese Resultate bestätigen die Ergebnisse von Bergmeir, Benitez [BB12, S.192], welche ebenfalls auf eine systematische Unterschätzung der Kreuzvalidierungsverfahren verweisen. Die Wissenschaftler führen diese Erkenntnis auf die vorhandene Korrelation in den Zeitreihendaten zurück.

Das relative Fehlermaß verweist auf keine bestehenden, systematischen Unterschiede unter den vier untersuchten Kreuzvalidierungsverfahren. Bei allen Verfahren liegt der

Median ungefähr in der Mitte des Boxplots, was auf eine symmetrische Verteilung der relativen Fehler schließen lässt. Die Streuung der Fehlerwerte ist ebenfalls beinahe ident.

Modellselektionsverfahren	RMSE	MAPAE	MPAE
k-fache Kreuzvalidierung	1.3438	1.0438	-0.4442
Geblockte Kreuzvalidierung	1.3415	1.0247	-0.3991
Modifizierte Kreuzvalidierung	1.3601	1.0585	-0.4002
Hv-geblockte Kreuzvalidierung	1.3396	1.0242	-0.3708
Holdout	1.1409	0.8596	-0.1269

Tabelle 3: Resultate von AS.3 anhand der skalenbasierten Fehlermaße.

Für eine genauere Untersuchung von AS.3 werden die drei skalenabhängige Fehlermaße (RMSE, MAPAE und MPAE) heranzogen. Tabelle 3 zeigt die Ergebnisse für den Fall, dass das Prognosemodell stark falsch spezifiziert ist. Die Resultate unterstützen die gewonnene Erkenntnis anhand der Boxplots, dass in AS.3 das Holdout-Verfahren die robustesten Fehlerabschätzungen liefert. Der in AS.1 und AS.2 gefundene Vorteil der Kreuzvalidierungsverfahren verschwindet, wenn das Modell für die Prognose nicht geeignet ist.

Bezüglich dem RMSE weist die hv-geblockte Kreuzvalidierung mit einem Fehlerwert von 1.3396 die beste Leistung unter den Kreuzvalidierungsverfahren auf, dicht gefolgt von der geblockten Kreuzvalidierung und der k-fachen Kreuzvalidierung. Die modifizierte Kreuzvalidierung schneidet am schlechtesten ab. Dieses Muster ist auch auf das Fehlermaß MAPAE anwendbar. Die durchschnittliche Größe des Fehlers bei den Kreuzvalidierungsverfahren liegt über dem Wert von eins. Zum Vergleich weist die Holdout-Methode einen Fehlerwert von 0.8596 auf. Das Fehlermaß MPAE gibt Auskunft darüber, ob das gewählte Validierungsverfahren den Prognosefehler über- oder unterschätzt. Tabelle 3 zeigt, dass alle Validierungsverfahren den tatsächlichen Prognosefehler tendenziell zu niedrig einschätzen. Dieses Resultat steht in Einklang mit den Ergebnissen des relativen Fehlermaßes (siehe Abbildung 15). Das k-fache Kreuzvalidierungsverfahren unterschätzt hierbei am meisten den tatsächlichen Prognosefehler, mit einem Wert von -0.4442.

Ist das geeignete Prognosemodell nicht bekannt und daher falsch spezifiziert, so bietet das klassische Holdout-Verfahren aussagekräftigere und robustere Fehlerabschätzungen als die vier untersuchten Varianten des Kreuzvalidierungsverfahrens. Das

Kreuzvalidierungsverfahren neigt in AS.3 zu einer systematischen Unterschätzung des tatsächlichen Prognosefehlers. Unter den Varianten der Kreuzvalidierungsverfahren weist die hv-geblockte Kreuzvalidierung die niedrigsten Fehlerwerte auf. Die geblockte Kreuzvalidierung hat minimal schlechtere Werte.

8.3 Zusammenfassung der Resultate

Die drei Anwendungsszenarien verdeutlichen, dass es nicht möglich ist eine allgemein gültige Aussage zu treffen, betreffend der Beurteilung, welches Modellvalidierungsverfahren die robustesten und aussagekräftigsten Fehlerabschätzungen in stationären Zeitreihendaten bietet. Welches Validierungsverfahren sich am besten für die Modellselektion eignet, hängt von vielen verschiedenen Faktoren ab, wie beispielsweise der Art des Prognosemodells, der Qualität der Daten und der Länge der Daten [Do12, S.82]. Die erzielten Resultate bestätigen, dass die Wahl des Prognosemodells ein entscheidender Faktor, bei der Wahl des Modellvalidierungsverfahrens ist. Ist das Prognosemodell passend oder zumindest optimal, so sind die vier untersuchten Varianten der Kreuzvalidierung robuster als das klassische Holdout-Verfahren. Die theoretischen Mängel der standardmäßigen k-fachen Kreuzvalidierung sind in diesen Fällen in der Praxis zu vernachlässigen. Ist das Prognosemodell falsch spezifiziert oder ist man sich über das passende Prognosemodell nicht einig, so sollte das Holdout-Verfahren als Validierungsverfahren gewählt werden.

Die Resultate zeigen ebenfalls, dass die geblockte und die hv-geblockte Kreuzvalidierung die robustesten Prognosefehlerabschätzungen unter den Kreuzvalidierungsverfahren bei stationären Zeitreihendaten bieten. Die beiden Verfahren weisen sehr ähnliche Fehlerwerte entlang der drei Anwendungsszenarien auf, weshalb eine eindeutige Entscheidung zugunsten eines Modellvalidierungsverfahrens schwer ist. Ist das Modell für die Vorhersage der Zeitreihendaten geeignet, so liefert die geblockte Kreuzvalidierung die etwas robusteren Resultate. Wenn das Prognosemodell allerdings nicht optimal oder falsch spezifiziert ist, so zeigen die erzielten Ergebnisse, dass die hv-geblockte Kreuzvalidierung die robusteren Prognosefehlerschätzungen liefert. Beide Verfahren umgehen die theoretischen Mängel der k-fachen Kreuzvalidierung. Der Unterschied besteht lediglich in dem verwendetem Trainingsdatensatz, da bei der hv-geblockten Kreuzvalidierung abhängige Beobachtungen entfernt werden.

Die modifizierte Kreuzvalidierung ist in keinem der drei untersuchten Anwendungsszenarien konkurrenzfähig. Eine mögliche Ursache für die schlechte

Performance ist, dass bei der Datenaufteilung sehr viele Beobachtungen aus dem Trainingsdatensatz entfernt werden. Wenn das Modell auf zu wenig Daten trainiert wird, kann es zu einer Überanpassung (Overfitting) oder zu einer Unteranpassung (Underfitting) kommen. Beide grundlegende Probleme führen dazu, dass das Prognosemodell auf neuen Daten schlecht abschneidet und unzuverlässige Vorhersagen trifft [Ko18, S. 6].

9 Prüfung der Robustheit

Ein bedeutender Vorteil der Kreuzvalidierung ist, dass dieses Verfahren den Datensatz vollständig ausnützt. Aus diesem Grund wird insbesondere bei einer kurzen Datenlänge das Kreuzvalidierungsverfahren gegenüber dem Holdout-Verfahren präferiert. Zeitreihendaten stellen einen Sonderfall dar, da die Daten nicht u.i.v. sind, weshalb der Einsatz des Kreuzvalidierungsverfahren theoretische Probleme aufwirft [BB12, S.193]. Doch die Ergebnisse von AS.1 und AS.2 in Kapitel 8 zeigen, dass diese theoretischen Mängel in der Praxis nicht relevant sind, wenn das Prognosemodell passend oder nicht ganz optimal ist. Denn in diesem Fall liefern alle Varianten des Kreuzvalidierungsverfahren robustere Ergebnisse als das Holdout-Verfahren. Die Ergebnisse dieses Experimentes beruhen auf einer Datenlänge von $T = 200$. Nun stellt sich die Frage, ob dieser Sachverhalt auch für andere Datenlängen T zutrifft. Um eine Zeitreihenanalyse durchführen zu können, sollte die Zeitreihe mindestens 50 Beobachtungen beinhalten [Wa98, S. 2-3]. Die untersuchten Zeitreihen weisen daher eine Datenlänge von 50, 100, 200 und 500 auf. Ziel ist anhand der vier Datenlängen zu untersuchen, ob das in AS.1 und AS.2 erzielte Ergebnis, dass das Kreuzvalidierungsverfahren robustere Fehlerabschätzungen liefert als das Holdout-Verfahren, unabhängig von der Datenlänge T ist.

Darüber hinaus wird untersucht, welchen Einfluss die Datenlänge auf die Robustheit der vier untersuchten Varianten des Kreuzvalidierungsverfahren hat. Wenn die Anzahl an Beobachtungen in einer Zeitreihe gering ist, können die Validierungsmethoden unzuverlässige Schätzungen des Prognosefehlers liefern. Dies liegt daran, dass es in einer kurzen Datenlänge möglicherweise zu wenige Beobachtungen gibt, um eine genaue Schätzung des tatsächlichen Prognosefehlers zu erhalten. Im Allgemeinen gilt: Je mehr Beobachtungen verfügbar sind, desto genauer ist die Schätzung des tatsächlichen Prognosefehlers durch das Validierungsverfahren. Diese Tatsache ist nicht überraschend, da statistische Methoden meistens schlechter abschneiden, wenn sie auf weniger Beobachtungen trainiert werden [Ja13, S. 181]. Durch die Untersuchung des Zusammenhanges zwischen Datenlänge und der Robustheit von Kreuzvalidierungsverfahren soll die Frage beantwortet werden, welchen Einfluss die Länge der Zeitreihendaten auf die Wahl des robustesten und aussagekräftigsten Kreuzvalidierungsverfahrens hat.

9.1 Robustheitstest von AS.1

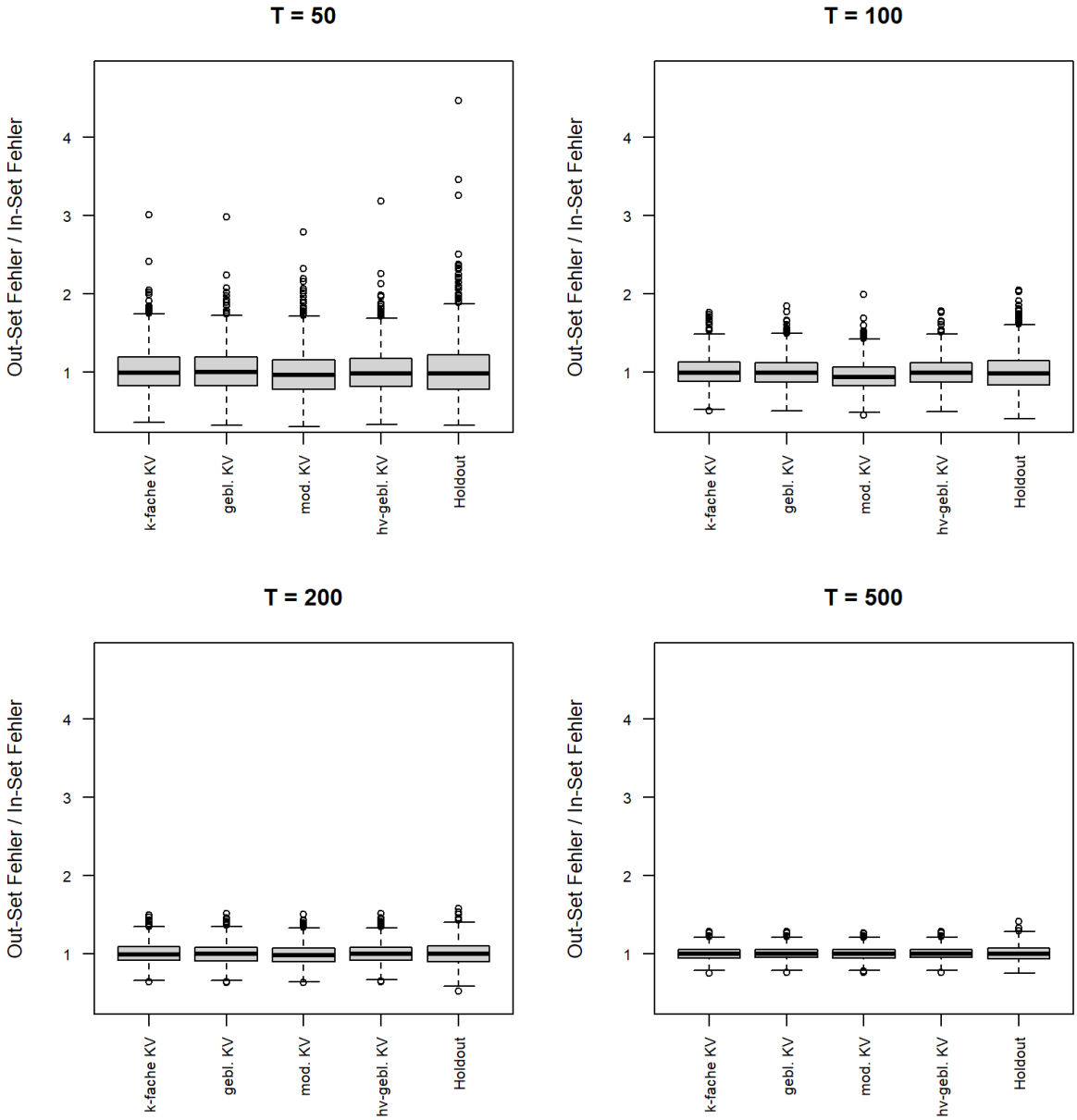

Abbildung 16: Boxplot zur Veranschaulichung der Ergebnisse von AS.1 für die vier Datenlängen $T = 50$, 100, 200 und 500.

Abbildung 16 zeigt die Verteilung des relativen Fehlers $\left(\frac{Out-Set\ Fehler}{In-Set\ Fehler}\right)$ für die fünf untersuchten Modellvalidierungsverfahren für vier verschiedene Datenlängen ($T = 50, 100, 200$ und 500) in Form von Boxplots. Die Abbildung bezieht sich auf AS.1, wo das Prognosemodell für die Vorhersage der Zeitreihendaten geeignet ist. Es ist deutlich erkennbar, dass die Varianz der relativen Fehlerverteilung mit steigender Datenlänge T bei allen Validierungsverfahren kleiner wird. Während die Spannweite bei $T = 50$ von 0.32 bis 4.47 reicht, so ist sie bei $T = 500$ nur von 0.75 bis 1.41. Doch auch der

Interquartilsabstand wird mit steigender Größe der Datenlänge T kleiner. Dies ist unabhängig von der Wahl des Validierungsverfahren. Dieses Ergebnis bekräftigt die Theorie, dass mit steigender Anzahl an Beobachtungen im Datensatz, die Schätzung des tatsächlichen Prognosefehlers durch das Validierungsverfahren genauer wird.

Eine weitere Gemeinsamkeit aller fünf Validierungsverfahren über alle vier Datenlängen ist, dass die Verteilung der relativen Fehler symmetrisch ist. Der Median entspricht bei allen untersuchten Verfahren in etwa dem Mittelwert und ist, mit Ausnahme der modifizierten Kreuzvalidierung, um den Wert eins zentriert. Es besteht daher keine systematische Über- oder Unterschätzung des tatsächlichen Prognosefehlers. Auffallend ist, dass die Ausreißer Großteils über dem oberen Whisker liegen. Mit steigender Datenlänge T nimmt die Anzahl an Ausreißer ebenfalls ab. Dieser Sachverhalt ist beispielsweise deutlich zu sehen, wenn ein Vergleich der Boxplots von $T = 50$ mit $T = 500$ erfolgt.

AS.1 in Kapitel 8 zeigt, dass das Kreuzvalidierungsverfahren robustere Fehlerabschätzungen liefert als das Holdout-Verfahren, wenn das Prognosemodell passend ist. Abbildung 16 veranschaulicht, dass dieses Ergebnis unabhängig von der Datenlänge T ist. Das Holdout-Verfahren weist über alle vier Datenlängen eine größere Spannweite und einen größeren Interquartilsabstand auf, als die vier Kreuzvalidierungsverfahren. Die Verteilung der relativen Fehlerdaten beim Holdout-Verfahren sind daher stärker gestreut und weisen eine höhere Varianz auf. Folglich ist das Kreuzvalidierungsverfahren robuster als das Holdout-Verfahren, wenn das Prognosemodell für die Vorhersage der Zeitreihe geeignet ist.

Ob die Länge der Zeitreihendaten einen Einfluss auf die Wahl des robustesten Modellvalidierungsverfahren hat, kann anhand der Boxplots nicht beurteilt werden, da die k-fache Kreuzvalidierung, die geblockte Kreuzvalidierung und die hv-geblockte Kreuzvalidierung sehr ähnliche Boxplots, über alle vier Datenlängen T aufweisen. Einzig das modifizierte Kreuzverfahren weicht etwas ab. Visuell ist erkennbar, dass bei $T = 100$ und $T = 200$ der Median unter eins liegt und das Verfahren daher den tatsächlichen Prognosefehler minimal überschätzt. Eine mögliche Ursache hierfür ist, dass bei der modifizierten Kreuzvalidierung abhängige Beobachtungen aus dem Trainingsdatensatz entfernt werden und das Trainieren des Prognosemodells daher anhand deutlich weniger Beobachtungen stattfindet [BCB14, S.4]. Interessant hierbei ist, dass bei $T = 50$ das modifizierte Kreuzvalidierungsverfahren einen ähnlichen Boxplot zu den weiteren drei Kreuzvalidierungsverfahren aufweist. Basierend auf der Erkenntnis, dass die

modifizierte Kreuzvalidierung bei kurzen Datensätzen eine deutlich schlechtere Leistung liefert im Vergleich zu anderen Kreuzvalidierungsverfahren, wäre vorhersehbar gewesen, dass der Median der relativen Fehlerverteilung bei $T = 50$ deutlich von dem Wert 1.0 abweicht. Dies ist darauf zurückzuführen, dass bei der modifizierten Kreuzvalidierung es beispielsweise möglich ist, dass 20 abhängige Beobachtungen aus dem Trainingsdatensatz entfernt werden. Bei einer Datengröße von $T = 50$ hat dies einen deutlich größeren Einfluss, als bei $T = 100, 200$ oder 500. Angesichts der Theorie wäre eine kontinuierliche Annäherung des Medians mit steigender Datenlänge T an den Wert eins zu erwarten gewesen.

Tabelle 4 demonstriert die Ergebnisse von AS.1 unter Verwendung der skalenbasierten Fehlermaße RMSE, MAPAE und MPAE für die Datenlängen $T = 100, 200, 350$ und 500. Bei Betrachtung der Spalte für den RMSE fällt unmittelbar auf, dass das Holdout-Verfahren über alle vier Datenlängen den höchsten Wert aufweist. Dieser ist im Vergleich zur geblockten Kreuzvalidierung stets um mindestens 20% höher. Das beste Ergebnis hinsichtlich des RMSE-Fehlermaßes erzielt die k-fache Kreuzvalidierung und die geblockte Kreuzvalidierung. Die Abweichungen zwischen beiden Verfahren liegen bei allen Datenlängen unter 1%, weshalb keine eindeutige Entscheidung getroffen werden kann, welches Verfahren als das robustere angesehen werden kann. Im Vergleich dazu liefert die hv-geblockte Kreuzvalidierung ebenfalls gute Ergebnisse, jedoch mit einem höheren RMSE-Wert über alle untersuchten Datenlängen als die k-fache und die geblockte Kreuzvalidierung. Das Fehlermaß MAPAE liefert vergleichbare Resultate. Hinsichtlich des Fehlermaßes MPAE zeigt sich lediglich ein Unterschied darin, dass das Holdout-Verfahren im Vergleich zur modifizierten Kreuzvalidierung bessere Ergebnisse erzielt.

Die Untersuchung legt nahe, dass der Einfluss der Datenlänge bei der Wahl des Kreuzvalidierungsverfahrens kaum relevant ist. Ist das Prognosemodell für die Vorhersage der Zeitreihendaten geeignet, so liefert die geblockte und die k-fache Kreuzvalidierung die robustesten Fehlerabschätzungen. Dieses Ergebnis ist unabhängig von der Anzahl an Beobachtungen im Gesamtdatensatz.

Modellselektionsverfahren	Länge	RMSE	MAPAE	MPAE
k-fache Kreuzvalidierung	50	0.2789	0.2178	-0.0061
	100	0.1888	0.1503	0.0037
	200	0.1289	0.1031	0.0026
	500	0.0794	0.0639	0.0020
Geblockte Kreuzvalidierung	50	0.2807	0.2205	-0.0062
	100	0.1902	0.1522	0.0064
	200	0.1287	0.1026	0.0033
	500	0.0797	0.0640	0.0018
Modifizierte Kreuzvalidierung	50	0.3166	0.2441	0.0454
	100	0.2088	0.1652	0.0672
	200	0.1309	0.1052	0.0178
	500	0.0806	0.0648	0.0082
Hv-geblockte Kreuzvalidierung	50	0.2855	0.2246	0.0093
	100	0.1904	0.1524	0.0093
	200	0.1290	0.1028	0.0041
	500	0.0797	0.0640	0.0019
Holdout	50	0.3501	0.2756	0.0141
	100	0.2496	0.1956	0.0303
	200	0.1561	0.1239	0.0110
	500	0.0979	0.0789	0.0022

Tabelle 4: Resultate von AS.1 anhand der skalenbasierten Fehlermaße für die Datenlängen T =50, 100, 200 und 500.

9.2 Robustheitstest von AS.2

AS.2 untersucht die Modellvalidierungsverfahren unter der Annahme, dass das Prognosemodell nicht optimal ist. Zur Überprüfung des Einflusses der Datenlänge *T* werden auch hier für alle fünf Verfahren relative und skalenbasierte Fehler für die vier verschiedenen Datenlängen berechnet und ausgewertet.

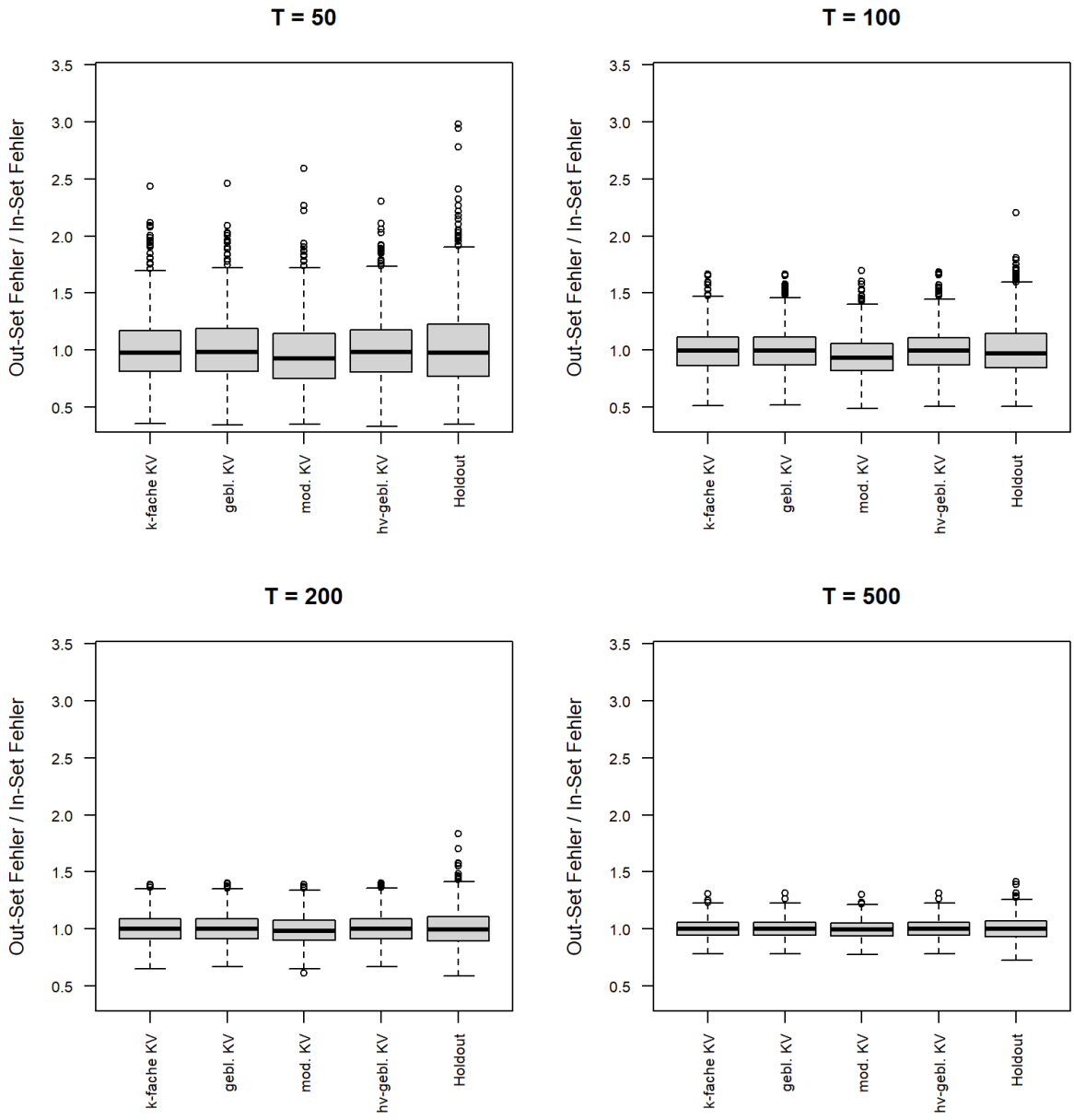

Abbildung 17: Boxplot zur Veranschaulichung der Ergebnisse von AS.2 für die vier Datenlängen T = 50, 100, 200 und 500.

Abbildung 17 demonstriert die Verteilung der relativen Prognosefehler für AS.2 in Form von Boxplots. Es ist erkennbar, dass die Varianz und die Anzahl an Ausreißern in den relativen Fehlern mit zunehmender Datenlänge *T* abnimmt und dass alle

Validierungsverfahren eine symmetrische Verteilung der relativen Fehlerdaten aufweisen, wobei die Boxplots, mit Ausnahme der modifizierten Kreuzvalidierung, um den Wert eins zentriert sind. Zusätzlich bestätigen die vorliegenden Ergebnisse die geringere Leistungsfähigkeit des Holdout-Verfahrens im Vergleich zu den Kreuzvalidierungsverfahren, wenn das Prognosemodell nicht optimal ist. Dies ist darauf zurückzuführen, dass die Spannweite und der Interquartilsabstand des Holdout-Verfahrens stets größer ist, als jene der vier untersuchten Kreuzvalidierungsverfahren. Diese Erkenntnis gilt unabhängig von der Datenlänge T und ist daher in Übereinstimmung mit den Resultaten von AS.1. Folglich kann die Erkenntnis gezogen werden, dass die Ergebnisse aus AS.2 die Schlussfolgerungen aus AS.1 bestätigen.

Tabelle 5 enthält die Ergebnisse für die skalenbasierten Fehlermaße RMSE, MAPAE und MPAE. Die Tabelle zeigt, dass die hv-geblockte und die geblockte Kreuzvalidierung die niedrigsten RMSE- und MAPAE-Werte aufweisen. Bei $T = 50$ und $T = 100$ schneidet die geblockte Kreuzvalidierung minimal besser ab, wohingegen die hv-geblockte Kreuzvalidierung bei $T = 200$ und $T = 500$ minimal niedrigere Werte hat. An dieser Stelle sei jedoch angemerkt, dass die Differenzen stets unter dem 1%-Bereich liegen. Bereits in Kapitel 8 wurde gezeigt, dass die hv-geblockte und die geblockte Kreuzvalidierung eine vergleichbare Performance hinsichtlich der Fehlerabschätzung aufweisen. Die Ergebnisse der vorliegenden Untersuchung bestätigen diese Beobachtung und zeigen, dass die Wahl des robustesten Validierungsverfahren für die Prognosefehlerabschätzung nicht signifikant von der Datenlänge T beeinflusst wird.

Modellselektionsverfahren	Länge	RMSE	MAPAE	MPAE
k-fache Kreuzvalidierung	50	0.2843	0.2254	0.0086
	100	0.1871	0.1484	0.0066
	200	0.1304	0.1045	-0.0022
	500	0.0814	0.0658	-0.0004
Geblockte Kreuzvalidierung	50	0.2826	0.2246	0.0041
	100	0.1866	0.1476	0.0055
	200	0.1299	0.1040	-0.0033
	500	0.0812	0.0657	-0.0008
Modifizierte Kreuzvalidierung	50	0.3336	0.2625	0.0745
	100	0.2122	0.1699	0.0722
	200	0.1340	0.1069	0.0121
	500	0.0820	0.0661	0.0062
Hv-geblockte Kreuzvalidierung	50	0.2851	0.2264	0.0127
	100	0.1865	0.1478	0.0070
	200	0.1297	0.1038	-0.0032
	500	0.0813	0.0657	-0.0008
Holdout	50	0.3502	0.2782	0.0245
	100	0.2328	0.1856	0.0201
	200	0.1606	0.1281	0.0040
	500	0.1022	0.0822	-0.0003

Tabelle 5: Resultate von AS.2 anhand der skalenbasierten Fehlermaße für die Datenlängen T=50, 100, 200 und 500.

9.3 Robustheitstest von AS.3

Für die Untersuchung des Einflusses der Datenlänge T auf die Wahl des robustesten Validierungsverfahrens wird ebenfalls AS.3 untersucht. In AS.3 herrscht Uneinigkeit über das passende Prognosemodell und ist daher möglicherweise falsch spezifiziert.

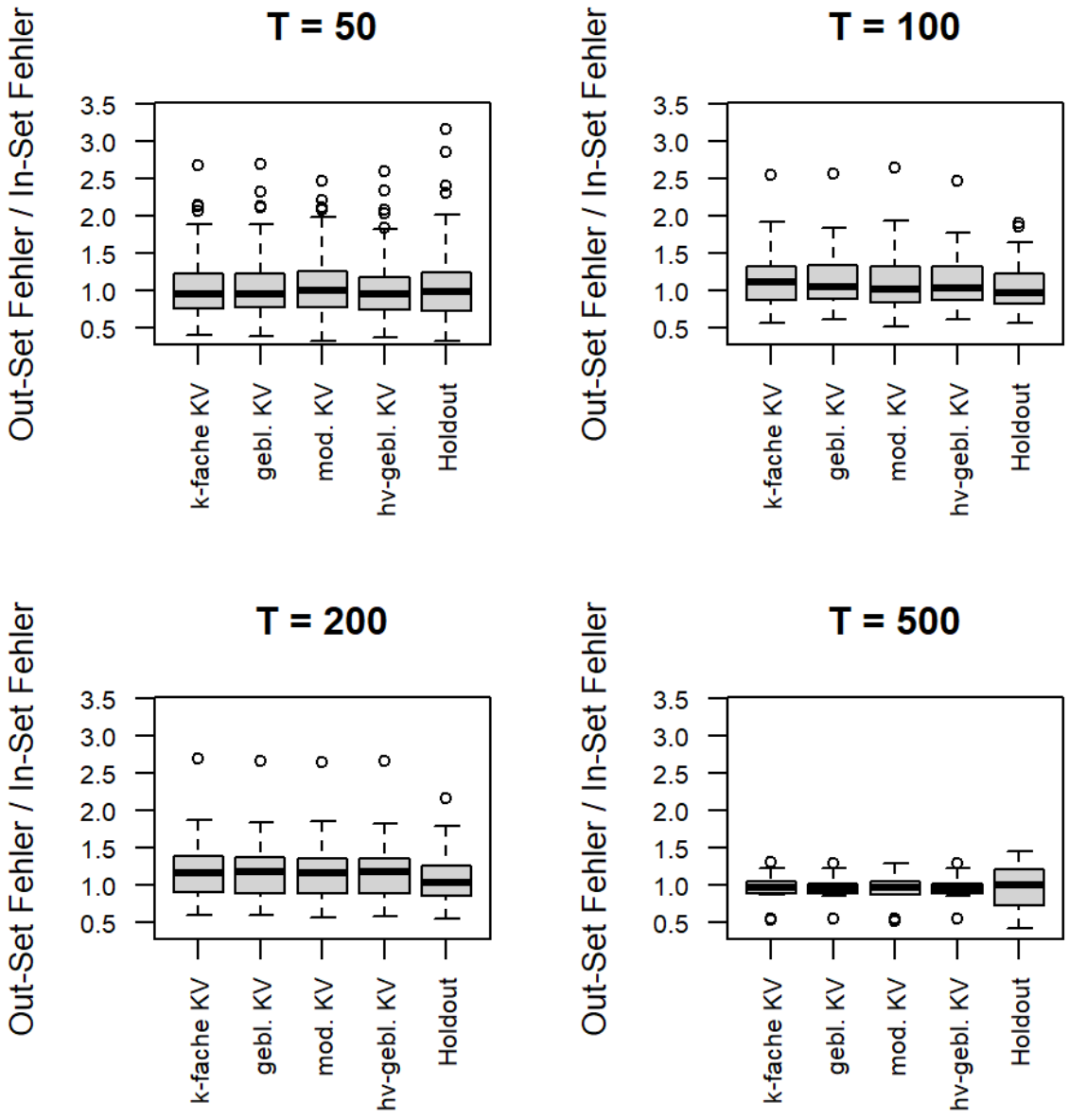

Abbildung 18: Boxplot zur Veranschaulichung der Ergebnisse von AS.3 für die vier Datenlängen T=50, 100, 200 und 500.

Bei einer Datenlänge von $T = 200$ haben die Ergebnisse in Kapitel 8 gezeigt, dass das Holdout-Verfahren robustere Fehlerabschätzungen bietet als die Gruppe der Kreuzvalidierungsverfahren, sowie dass die Kreuzvalidierungsverfahren zu einer Überschätzung des tatsächlichen Prognosefehlers tendieren. Abbildung 18 weist hingegen

darauf hin, dass die in AS.3 erzielten Resultate nur für eine Datenlänge von $T = 200$ gültig sind. Eine Änderung der Datenlänge kann zu abweichenden Resultaten führen. Zur Unterstützung dieser Feststellung werden zwei Beispiele mit den Datenlängen von $T = 50$ und $T = 500$ angeführt. Im ersten Beispiel, bei einer Datenlänge von $T = 50$, zeigt sich, dass der Median aller Kreuzvalidierungsverfahren in etwa bei dem Wert ein liegt und dass die Spannweite und der Interquartilsabstand kleiner sind als beim Holdout-Verfahren. Demnach ist das Holdout-Verfahren nicht robuster als die Gruppe der Kreuzvalidierungsverfahren. Die Veranschaulichung vom zweiten Beispiel, bei einer Datenlänge von $T = 500$, erfolgt anhand von Abbildung 19. Abbildung 19 spiegelt die gleichen Boxplots wie in Abbildung 18 wider, mit dem Unterschied dass die y-Achse in Abbildung 19 von 0.40 bis 1.45 reicht (und nicht von 0.40 bis 3.50). Anhand der Boxplots in Abbildung 19 ist erkennbar, dass die Kreuzvalidierungsverfahren eher zu einer Unterschätzung, als zu einer Überschätzung, des tatsächlichen Prognosefehlers tendieren.

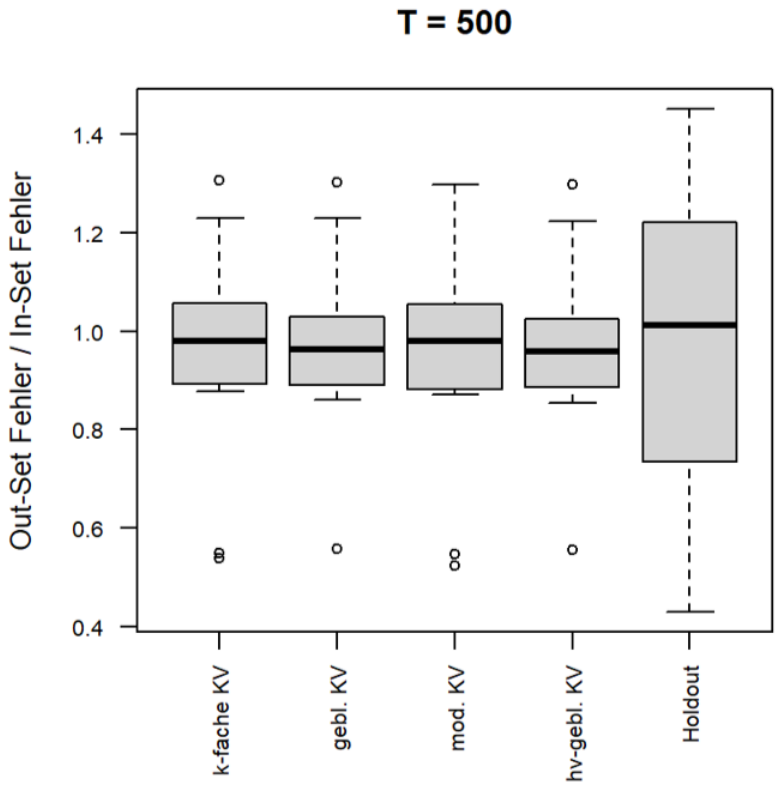

Abbildung 19: Boxplot zur detaillierten Veranschaulichung der Ergebnisse von AS.3 für die Datenlänge T=500.

Ein systematischer Zusammenhang, der sich über alle vier Datenlängen erstreckt, ist in AS.3 nicht ersichtlich. Ein möglicher Grund hierfür könnte sein, dass die Anzahl an Wiederholungen nicht ausreichend ist, um eine zuverlässige Beurteilung der Fehlerabschätzungsfähigkeit der Validierungsverfahren zu erhalten. In AS.3 variiert die Wiederholungsanzahl zwischen 144 (für $T = 50$) und 14 (für $T = 500$). Im Vergleich hierzu werden in AS.1 und AS.3 1000 Wiederholungen durchgeführt. Ist die Anzahl an Wiederholungen zu gering, so kann dies zu einem verzerrten Bild der tatsächlichen Verhaltensweise des Validierungsverfahrens führen. Eine höhere Anzahl an Wiederholungen verbessert hingegen die statistische Aussagekraft der Ergebnisse und ermöglicht daher verlässlichere Aussagen über die Leistungsfähigkeit des Validierungsverfahrens [Ki11, S. 24].

Tabelle 6 zeigt die Ergebnisse von AS.3 anhand der skalenbasierten Fehlermaße RMSE, MAPAE und MPAE. Die skalenbasierten Fehlermaße bestätigen die Beobachtung des relativen Fehlermaßes, dass die erzielten Ergebnisse von AS.3 mit einer gewissen Vorsicht zu betrachten sind. Dies lässt sich darauf zurückführen, dass alle drei Fehlermaße bei einer Datenlänge von $T = 50$ oder $T = 100$ niedrigere Fehlerwerte aufweisen, als bei einer Datenlänge von $T = 200$. Der RMSE-Wert der k-fachen Kreuzvalidierung beträgt bei $T = 50$ beispielsweise 1.1136 und hat bei $T = 200$ einen Wert von 1.3438. Diese Ergebnisse stehen im Widerspruch zu AS.1 und AS.2, welche besagen, dass die Fehlerwerte von RMSE und MAPAE mit steigender Datenlänge T bei allen Validierungsverfahren kleiner werden. Darüber hinaus weisen die vorliegenden Ergebnisse in Tabelle 6 keine Anhaltspunkte für systematische Zusammenhänge auf. Aus diesem Grund wäre es unangemessen, allein auf Basis des realen Datensatzes von AS.3 zu beurteilen, welches Kreuzvalidierungsverfahren die robustesten Fehlerabschätzungen bietet, wenn das Prognosemodell falsch spezifiziert ist oder das geeignete Modell nicht bekannt. Auf eine genauere Untersuchung dieser Fragestellung anhand der drei skalenbasierten Fehlermaße wird deshalb verzichtet.

Die dargestellten Ergebnisse in Tabelle 6 deuten darauf hin, dass das Holdout-Verfahren nicht zwingend besser geeignet ist als alle Kreuzvalidierungsverfahren in AS.3. Der RMSE-Wert des Holdout-Verfahrens ist beispielsweise nur bei einer Datenlänge von $T = 200$ am niedrigsten. Bei $T = 50$ und $T = 500$ hat die geblockte-Kreuzvalidierung und bei $T = 100$ die hv-geblockte Kreuzvalidierung den niedrigsten RMSE-Wert. Ähnliche Ergebnisse zeigen sich auch beim Fehlermaß MAPAE.

Die Überprüfung des Einflusses der Datenlänge T auf die Leistungsfähigkeit der Validierungsverfahren impliziert, dass die erzielten Ergebnisse für AS.3 in Kapitel 8 mit Vorsicht zu interpretieren sind. Um verlässlichere Schlussfolgerungen ziehen zu können, bedarf es einer genaueren Untersuchung von AS.3. Eine Möglichkeit hierfür wäre die Verwendung eines synthetischen Datensatzes oder die Verwendung von mehr als zwei realen Zeitreihendatensätze.

Modellselektionsverfahren	**Länge**	**RMSE**	**MAPAE**	**MPAE**
k-fache Kreuzvalidierung	50	1.1136	0.8081	0.0001
	100	1.1554	0.8379	-0.3456
	200	1.3438	1.0438	-0.4442
	500	0.8991	0.6030	0.1914
Geblockte Kreuzvalidierung	50	1.1088	0.8107	0.0116
	100	1.1458	0.8265	-0.3313
	200	1.3415	1.0247	-0.3991
	500	0.8643	0.6004	0.2116
Modifizierte Kreuzvalidierung	50	1.4224	0.9250	0.0477
	100	1.2018	0.8580	-0.2022
	200	1.3601	1.0585	-0.4002
	500	0.9295	0.6227	0.2156
Hv-geblockte Kreuzvalidierung	50	1.1286	0.8226	0.0636
	100	1.1274	0.8181	-0.3061
	200	1.3396	1.0242	-0.3708
	500	0.8689	0.6063	0.2244
Holdout	50	1.3409	0.9592	0.1190
	100	1.1616	0.8400	0.0415
	200	1.1409	0.8596	-0.1269
	500	1.2087	0.9147	0.2200

Tabelle 6: Resultate von AS.3 anhand der skalenbasierten Fehlermaße für die Datenlängen T=50, 100, 200 und 500.

10 Conclusio

10.1 Beantwortung der Forschungsfrage

In der Modellselektion bei Prognosen gilt es jenes Modell mit seinen Parametern zu wählen, dass den kleinsten out-of-sample Prognosefehler besitzt. In der wissenschaftlichen Literatur gibt es derzeit keine etablierte Vorgehensweise zur Schätzung des out-of-sample Prognosefehlers in Zeitreihendaten. Die gestellte Forschungsfrage: „Welches Kreuzvalidierungsverfahren bietet die robustesten und aussagekräftigsten Prognosefehlerabschätzungen bei stationären Zeitreihendaten?“, konnte anhand der Literatur nicht beantwortet werden. Das Ziel dieser Masterarbeit war, mit Hilfe eines empirischen Experimentes eine Antwort auf diese Fragestellung zu finden.

Die Ergebnisse zeigten, dass es keine eindeutige Antwort auf die gestellte Forschungsfrage gibt, da die Robustheit der Kreuzvalidierungsverfahren von dem verwendeten Prognosemodell und von den Daten abhängig ist. Ist das Prognosemodell geeignet (AS.1) oder zumindest optimal (AS.2), so lieferte die geblockte und die hv-geblockte Kreuzvalidierung die robustesten und aussagekräftigsten Fehlerabschätzungen bei stationären Zeitreihenprognosen. Darüber hinaus bekräftigten die erzielten Resultate von AS.1 und AS.2 die Erkenntnis von Bergmeir, Hyndmann, Koo [BKH18], dass die k-fache Kreuzvalidierung robuster ist, als das Holdout-Verfahren. Die theoretischen Mängel der k-fachen Kreuzvalidierung sind daher in der Praxis zu vernachlässigen. Eine Überprüfung des Einflusses der Datenlänge auf die Leistung der vier Kreuzvalidierungsverfahren zeigte, dass sich die Varianz der Fehlerabschätzungen im Allgemeinen mit zunehmender Länge der Zeitreihendaten verringert. Die untersuchten Validierungsverfahren können somit die Prognosegenauigkeit genauer schätzen, je mehr Beobachtungen zur Verfügung stehen. Die Experimente lieferten jedoch keinen Hinweis darauf, dass die Datenlänge der Zeitreihe einen signifikanten Einfluss auf die Wahl der Validierungsmethode hat. Es ist jedoch zu beachten, dass die Ergebnisse von AS.1 und AS.2 unter synthetischen Daten erzielt wurden. AS.3 untersuchte anhand von realen Temperaturdaten den Fall, dass das geeignete Prognosemodell nicht bekannt ist oder falsch spezifiziert ist. Die Resultate in Kapitel 8 zeigten, dass das Holdout-Verfahren als Validierungsverfahren gewählt werden sollte, wenn das passende Prognosemodell nicht bekannt ist. Eine Überprüfung des Einflusses der Datenlänge auf die Leistungsfähigkeit der Validierungsverfahren implizierte, dass dieses Ergebnis jedoch nur auf eine

Datenlänge von $T = 200$ beschränkt ist und die Ergebnisse in AS.3 daher mit Vorsicht zu interpretieren sind.

Das empirische Experiment zeigte, dass die Wahl des verlässlichsten Kreuzvalidierungsverfahrens von den zugrundeliegenden Daten und vom Prognosemodell abhängt. Für den Fall, dass synthetische Daten verwendet werden und das Prognosemodell passend oder nicht ganz optimal ist, so ist die Antwort auf die gestellte Forschungsfrage, dass die geblockte und die hv-geblockte Kreuzvalidierung die robustesten und aussagekräftigsten Prognosefehlerschätzungen in stationären Zeitreihendaten liefern.

10.2 Diskussion der Methode

Für die Beantwortung der Forschungsfrage wurde ein empirisches Experiment verwendet. Dieses vergleicht in drei unterschiedlichen Anwendungsszenarien vier Arten der Kreuzvalidierung und zusätzlich die Holdout-Methode. AS.1 und AS.2 basieren auf synthetische Daten, während AS.3 ein reales Szenario mit zwei Temperaturzeitreihendaten darstellt. Der Literaturteil dieser Arbeit dient zum besseren Verständnis des Themas, bietet jedoch keine Antwort auf die gestellte Forschungsfrage.

Anhand des empirischen Experimentes konnten Antworten auf einige Aspekte der gestellten Forschungsfrage gefunden werden. Sie ist jedoch viel zu komplex, um sie im Rahmen einer Masterarbeit umfassend beantworten zu können. Für die Schließung der Forschungslücke und einer generellen Beantwortung der gestellten Forschungsfrage, bedarf es einer umfassenderen Untersuchung, die auf einer größeren Menge an realen Daten basiert, mehrere Prognosemodelle abdeckt, verschiedene Vorhersagealgorithmen einbezieht und zusätzliche Fehlermaße berücksichtigt. Die Masterarbeit hat jedoch geschafft, die Forschungslücke zu reduzieren. Es wäre interessant dieses Thema in zukünftigen Forschungen weiter zu vertiefen, mit dem Ziel eine generelle Antwort auf die Forschungsfrage zu erhalten.

10.3 Ausblick

Eine mögliche Frage die sich aus den erzielten Ergebnissen ableitet ist, ob AS.1 (untersucht den Fall, dass das Prognosemodell geeignet ist) und AS.2 (falls das Prognosemodell sub-optimal ist) unter realen Daten auch andere Ergebnisse liefern würden. Die Frage lässt sich auch in die umgekehrte Richtung stellen: Wie würden sich die Validierungsverfahren in AS.3 (wenn das Prognosemodell falsch spezifiziert ist)

verhalten, wenn die Untersuchung anhand von synthetischen Daten erfolgt? Zukünftige Forschung sollte daher weitere Anwendungsszenarien unter Zuhilfenahme verschiedener Datensätze abdecken, insbesondere durch die Verwendung von realen Daten.

Eine weitere offene Frage in diesem Zusammenhang ist, wie sich die Kreuzvalidierungsverfahren verhalten, wenn die Zeitreihendaten nicht stationär sind. In der Praxis handelt es sich oft um Zeitreihen mit komplexen Strukturen, die kein stationäres Verhalten aufweisen. Diese beinhalten zukünftige Muster, die in vergangenen Beobachtungen nicht enthalten sind [CTM20, S. 22].

Die offenen Fragestellungen und Diskussionspunkte zeigen auf, dass noch viel Forschungsbedarf in diesem Thema besteht. Es ist von Bedeutung dieses Thema weiter zu erforschen, da viele Unternehmen, die mit Datensätzen arbeiten, vermehrt auf Machine Learning setzten. Durch den Einsatz von Machine Learning Ansätzen können Unternehmen zuverlässiger und effektiver arbeiten und einen Wettbewerbsvorteil erlangen [He21, S.1]. Die Kreuzvalidierung ist ein wesentlicher Bestandteil des Machine Learning Workflows und kann daher für viele Unternehmen eine entscheidende Rolle sein.

11 Literaturverzeichnis

[AC10] Arlot, Sylvain; Celisse, Alain:A survey of cross-validation procedures for model selection. In (Statistics Surveys, 4): S.40-79, 2010.

[Ad23] Adams, Ryan P.: Model Selection and Cross Validation. https://www.cs.princeton.edu/courses/archive/fall18/cos324/files/model-selection.pdf, 2023, Stand: 16.03.2023.

[BB11] Bergmeir, Christoph; Benitez, José M.: Forecaster performance evaluation with cross-validation and variants. In (11th International Conference on Intelligent Systems Design and Applications): S. 849-854, 2011.

[BB12] Bergmeir, Christoph; Benitez, José M.: On the use of cross-validation for time series predictor evaluation. In (Information Sciences, 191): S. 192-213, 2012.

[BCB14] Bergmeir, Christoph; Costantini, Mauro; Benitez, José M: On the usefulness of cross-validation for directional forecast evaluation. In (Computational Statistics & Data Analysis, 76): S. 132-143, 2014.

[BD16] Brockwell, Peter J.; Davis, Richard. A.: Introduction to time series and forecasting, 2. Springer, New York, 2016.

[BD87] Box, George E.; Draper, Norman R.: Empirical model-building and response surfaces. John Wiley & Sons, 1987.

[BHK18] Bergmeir, Christoph; Hyndman, Rob J.; Koo, Bonsoo: A note on the validity of cross-validation for evaluating autoregressive time series prediction. In (Computational Statistics & Data Analysis, 120): S. 70-83, 2018.

[Bi06] Bishop, Christopher M.: Pattern Recognition and Machine Learning, 4. Springer, New York, 2006.

[Bi23] Bickel, Steffen: Zeitreihenanalyse. https://www.cs.uni-potsdam.de/ml/teaching/ws08/zeitreihenanalyse, 2023, Stand: 06.05.2023.

[BF97] Breiman, Leo; Friedman, Jerome H.: Predicting multivariate responses in multiple linear regression. In (Journal of the Royal Statistical Society: Series B (Statistical Methodology), 59(1)): S. 3-54, 1997.

[CTM20] Cerqueira, Vitor; Torgo, Luis; Mozetič, Igor: Evaluating time series forecasting models, An empirical study on performance estimation methods. In (Machine Learning, 109): S. 1997-2028, 2020.

[CTS21] Cerqueira, Vitor; Torgo, Luis; Soares, Carlos: Model selection for time series forecasting, Empirical analysis of different estimators. 2021.

[Do12] Domingos, Pedro: A few useful things to know about machine learning. In (Communications of the ACM, 55(10)): S. 78-87, 2012.

[DTJ18] Ding, Jie; Tarokh, Vahid; Yang, Yuhong: Model selection techniques, An overview. In (IEEE Signal Processing Magazine, 35(6)): S. 16-34, 2018.

[Ge75] Geisser, Seymour: The predictive sample reuse method with applications. In (Journal of the American statistical Association, 70(350)): S. 320-328, 1975.

[Ha11] Hayashi, Fumio: Econometrics. Princeton University Press, 2011.

[Ha17] Hanes, David; Salgueiro, Gonzalo; Grossetete, Patrick; Barton, Robert; Henry, Jerome: IoT fundamentals, Networking technologies, protocols, and use cases for the internet of things. Indianapolis: Cisco Press, 2017.

[HA18] Hyndman, Rob J.; Athanasopoulos, George: Forecasting, Principles and Practice. OTexts, 2018.

[Ha94] Hamilton, James D.: Time Series Analysis. Princeton University Press, New Jersey, 1994.

[He21] He, Gang; Ahmad, Khwaja M.; Yu, Wenxin; Xu, Xiaochuan; Kumar, Jay: A Comparative Analysis of Machine Learning and Grey Models. 2021.

[HTF08] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome: The Elements of Statistical Learning, 2. Springer, Stanford, 2008.

[Hu18] Hussain, Walayat; Hussain, Farookh K.; Saberi, Morteza; Hussain, Omar K.; Chang, Elizabeth: Comparing time series with machine learning-based prediction approaches for violation management in Cloud SLAs. In (Future Generation Computer Systems, 89): S. 464-477, 2018.

[Ja13] James, Gareth; Hastie, Trevor; Tibshirani, Robert; Witten, Daniela: An Introduction to Statistical Learning, with Applications in R. Springer, New York, 2013.

[Jo22] Jose, Jonath: Introduction to time series analysis and its applications, https://www.researchgate.net/publication/362389180, 2022, Stand: 05.05.2023.

[Ki11] Kiviet, Jan F.: Monte Carlo simulation for econometricians. In (Foundations and Trends in Econometrics, 5(1–2)): S. 1-181, 2011.

[KJ13] Kuhn, Max; Johnson, Kjell: Applied predictive modeling. Springer, New York, 2013.

[Ko18] Koehrsen, Will: Overfitting vs. underfitting, A complete example. In (Towards Data Science): S. 1-12, 2018.

[Ko95] Kohavi, Ron: A study of cross-validation and bootstrap for accuracy estimation and model selection. In (Ijcai, 14(2)): S. 1137-1145, 1995.

[Ku07] Kuhlmann, Jan: Ausgewählte Verfahren der Holdout- und Kreuzvalidierung. Gabler, Wiesbaden, 2007.

[Le21] Lendave, Vijaysinh: A Guide to Different Evaluation Metrics for Time Series Forecasting Models. https://analyticsindiamag.com/a-guide-to-different-evaluation-metrics-for-time-series-forecasting-models/, 2021, Stand: 23.04.2023

[LY23] Lyashenko, Vladimir; Jha, Abhishek: Cross-Validation in Machine Learning, How to Do It Right. https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right, Stand: 12.05.2023.

[MT98] McQuarrie, Allan; Tsai, Chih-Ling: Regression and time series model selection. In (World Scientific), 1998.

[Ra00] Racine, Jeff: Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. In (Journal of econometrics, 99(1)): S.39-61, 2000.

[Ri21] Rink, Konstantin: Time Series Forecast Error Metrics you should know. https://towardsdatascience.com/time-series-forecast-error-metrics-you-should-know-cc88b8c67f27, 2021, Stand: 30.04.2023.

[Sc23] Schneider, Jeff: Cross Validation. https://www.cs.cmu.edu/~schneide/tut5/node42.html , 2023, Stand: 16.03.2023.

[SK11] Shmueli, Galit; Koppius, Otto R.: Predictive analytics in information systems research. In (MIS quarterly): S. 553-572, 2011.

[Sn88] Snijders, Tom: On cross-validation for predictor evaluation in time series. In (On Model Uncertainty and its Statistical Implications: Proceedings of a Workshop): S. 56-69, 1988.

[SS11] Shumway, Robert H.; Stoffer, David S.: Time Series Analysis and its Applications, 3. Springer, New York, 2011.

[Sy11] Syed, Ali R.: A Review of Cross Validation and Adaptive Model Selection. Veröffentlichte Masterarbeit an der Georgia State University, https://scholarworks.gsu.edu/cgi/viewcontent.cgi?article=1100&context=math_theses, 2011, Stand: 15.03.2023.

[Wa98] Warner, Rebecca M.: Spectral analysis of time-series data. Guilford Press, 1998.

[WG16] Wickham, Hadley; Grolemund, Garrett: R for data science, Import, Tidy, Transform, Visualize, and Model Data. O'Reilly Media, Inc., Sebastopol, 2016.

12 Abbildungsverzeichnis

13 Tabellenverzeichnis

14 Abkürzungsverzeichnis

ADF	Augmented Dickey-Fuller
AR	Autoregressiv
AS.1	Anwendungsszenario 1
AS.2	Anwendungsszenario 2
AS.3	Anwendungsszenario 3
DGP	Datengenerierungsprozess
MA	Moving-Average
MAPAE	Mean Absolute Predictive Accuracy Error
MPAE	Mean Predictive Accuracy Error
IoT	Internet of Things
SLA	Service-Level-Agreement
RMSE	Root Mean Squared Error
u.i.v	unabhängig und identisch verteilt

Anhang A

Das folgende kurze, numerische Beispiel spiegelt die Durchführung des empirischen Experimentes wider. Hierfür wird ein Datensatz mit einer Datenlänge von $T = 10$ gewählt. Im Speziellen wird die Methode der 5-fache Kreuzvalidierung dargestellt. Ziel dieses numerischen Beispiels ist, ein besseres Verständnis des Aufbaus und der Durchführung des Experimentes zu gewährleisten.

Ziel der Modellvalidierungsverfahren ist den tatsächlichen Prognosefehler PE so genau wie möglich anhand des In-Sets zu schätzen. Das In-Set enthält 80% der Daten, das Out-Set 20%. Anhand der In-Set Daten erfolgt eine Schätzung des (tatsächlichen) Prognosefehler $\widehat{PE}$. Die Out-Set Daten dienen zur Berechnung des tatsächlichen Prognosefehlers PE. Ist die Differenz zwischen PE und $\widehat{PE}$ gering, so liegt eine gute Leistung des Modellvalidierungsverfahrens vor.

1) Berechnung des tatsächlichen Prognosefehlers $\boldsymbol{PE}$ anhand der **Out-Set Daten**:

Der Gesamtdatensatz X wird in Trainings- und Testdaten geteilt:

Train (80%)				Test (20%)			
	target	Tm1	Tm2		target	Tm1	Tm2
1981-01-03	34.5	32.4	38.1	1981-01-11	20.4	20.6	36.1
1981-01-04	20.7	34.5	32.4	1981-01-12	30.1	20.4	20.6
1981-01-05	21.5	20.7	34.5				
1981-01-06	23.1	21.5	20.7				
1981-01-07	29.7	23.1	21.5				
1981-01-08	36.6	29.7	23.1				
1981-01-09	36.1	36.6	29.7				
1981-01-10	20.6	36.1	36.6				

Anschließend erfolgt eine ganz normale lineare Regression. Anhand der Trainingsdaten wird das Prognosemodell (lineare Regression mit zwei Regressoren) gebildet und der tatsächliche Prognosefehler anhand der Testdaten berechnet:

Prognose von y:

$$26.8565 + \begin{pmatrix} 20.6 \\ 20.4 \end{pmatrix} * \ 0.4941 + \begin{pmatrix} 36.1 \\ 20.6 \end{pmatrix} * (-0.4563) = \ \begin{pmatrix} 20.56 \\ 27.54 \end{pmatrix}$$

Berechnung des Fehlers (RMSE):

➔ $\sqrt[2]{\frac{1}{2} * ((20.56 - 20.4)^2 + (27.54 - 30.1)^2)} = \sim 1.81 \ = \ PE$

2) Berechnung des geschätzten Prognosefehlers $\widehat{PE}$ anhand der **In-Set Daten**:

Hier wird NUR der Trainingsdatensatz verwendet und nicht der komplette Datensatz, da der Trainingsdatensatz in diesem Fall die In-Set Daten widerspiegelt. Dies bedeutet:

„ursprüngliche“ Traininingsdaten = In-Set Daten (X_in)	target Tm1 Tm2 1981-01-03 34.5 32.4 38.1 1981-01-04 20.7 34.5 32.4 1981-01-05 21.5 20.7 34.5 1981-01-06 23.1 21.5 20.7 1981-01-07 29.7 23.1 21.5 1981-01-08 36.6 29.7 23.1 1981-01-09 36.1 36.6 29.7 1981-01-10 20.6 36.1 36.6

Der erste Schritt der k-fachen Kreuzvalidierung ist den Gesamtdatensatz (hier X_in) zu mischen. Nach zufälligem Mischen des Datensatzes entsteht folgender Datensatz X_in_neu:

```
           target  Tm1  Tm2
1981-01-09   36.1 36.6 29.7
1981-01-10   20.6 36.1 36.6
1981-01-05   21.5 20.7 34.5
1981-01-08   36.6 29.7 23.1
1981-01-04   20.7 34.5 32.4
1981-01-06   23.1 21.5 20.7
1981-01-07   29.7 23.1 21.5
1981-01-03   34.5 32.4 38.1
```

X_in_neu wird anschließend in 5 Blöcke eingeteilt, da die 5-fache Kreuzvalidierung gewählt worden ist. Bei einer 10-fachen Kreuzvalidierung beispielweise würde der Datensatz in 10 Blöcke eingeteilt werden. Die Daten gehören zu folgendem Block:

```
[1] 1 1 2 3 3 4 5 5
```

Die Aufteilung des Gesamtdatensatzes in Training- und Testdaten erfolgt nun gemäß der Blöcke. Ein Block spiegelt immer den Testdatensatz wider, während die verbliebenen $k-1$ Blöcke (hier 4 Blöcke) den Trainingsdatensatz bilden.

Die folgenden Schritte a) bis c) werden für alle 5 Blöcke durchgeführt. Demonstrativ wird nur der Rechenvorgang für Block 1 dargestellt:

a) Aufteilung in Trainings- und Testdatensatz

Angenommen Block 1 wird als Testdatensatz gewählt, so erfolgt folgende Aufteilung in Trainings- und Testdatensatz:

Trainingsdaten „neu“ für In-Set, Block 1	Testdaten „neu“ für In-Set, Block 1
target Tm1 Tm2 1981-01-05 21.5 20.7 34.5 1981-01-08 36.6 29.7 23.1 1981-01-04 20.7 34.5 32.4 1981-01-06 23.1 21.5 20.7 1981-01-07 29.7 23.1 21.5 1981-01-03 34.5 32.4 38.1	target Tm1 Tm2 1981-01-09 36.1 36.6 29.7 1981-01-10 20.6 36.1 36.6

b) Koeffizienten für Prognosemodell (hier lineare Regression mit zwei Regressoren) finden.

Die Regressionskoeffizienten werden mit Hilfe der OLS-Methode gefunden. Für das numerische Beispiele erhalten wir folgende Koeffizienten:

```
Coefficients:
(Intercept)         Tm1          Tm2
14.4675          0.5147       -0.2704
```

c) RMSE berechnen

Prognose für y:

```
         [,1]
[1,] 32.27748
[2,] 30.15464
```

Der RMSE beträgt daher:

$$RMSE = \sqrt[2]{\frac{1}{2} * ((32.28 - 36.1)^2 + (30.15 - 20.6)^2)} = \sim 7.2767$$

Die Schritte a) bis c) werden daher für alle 5 Blöcke wiederholt. Dies bedeutet, dass wir bei einer f-fachen Kreuzvalidierung eigentlich 5 Prognosemodelle und 5 Fehlerschätzungen haben. Aus den 5 Root-Mean Squared Errors wird der Durchschnitt gebildet, um eine Schätzung für den Prognosefehler zu erhalten:

$$(7.2767 \ldots + 0.56856 + 8.13019 \ldots + 8.3748.. + 9.6841\ldots) * \frac{1}{5} = \sim 6.81 = \widehat{PE}$$

In diesem numerischen Beispiel beträgt der tatsächliche Prognosefehler $PE = 1.81$ und der geschätzte Prognosefehler $\widehat{PE} = 6.81$.

Für die Messung des Fehlers zwischen $\widehat{PE}$ und PE über alle Datensätze im Experiment werden die in Kapitel 7.5 vorgestellten Fehlermaße verwendet.

Anhang B

Nachfolgend ist der R-Code um die Ergebnisse des empirischen Experimentes reproduzieren zu können. Für die Durchführung wurde die Version 2022.12.0+353 von **R** verwendet.
Die Hauptergebnisse der Masterarbeit befinden sich im R-Skript "Main_real.R" und „Main_syn.R“. Diese sind eigentlich die einzig erforderlichen R-Skript für die Replikation der Ergebnisse aus Kapitel 8. „Funktionen_Ergebnisse.R“, „Validierungsverfahren.R“ und „Verlustfunktion.R“ sind R-Skripte, die notwendige Funktionen für die Ausführung von „Main_real.R“ und „Main_syn.R“ beinhalten.
Darüber hinaus befindet sich im R-Skript „Daten.R“ der Code, um alle benötigten Daten für die Masterarbeit zu reproduzieren. Hierfür werden synthetische Daten aus einem AR(2) und einem MA(1) Prozess generiert, beziehungsweise reale Daten von der Time Series Data Library (tsdl) verwendet. „Daten_Funktionen.R“ und „simulateTS.R“ beinhalten notwendige Funktionen für die erfolgreiche Ausführung von „Daten.R“.
Für Kapitel 9 „Prüfen der Robustheit“ mussten weitere Daten generiert werden. Der Code hierfür befindet sich in dem R-Skript „Daten_robustcheck.R“. Die Auswertung der Ergebnisse erfolgt für AS.1 und AS.2 in „Main_Robustcheck_syn.R“, sowie für AS.3 in „Main_Robustcheck_real.R“. Für das Plotten der Boxplots wurde ein eigenes R-Skript, unter dem Namen „Plots_robustcheck.R“ erstellt.

Insgesamt besteht das Experiment aus 12 selbstgeschriebenen R-Skripten:

- B.1 Hauptergebnisse (Kapitel 8)
  - Main_syn.R
  - Main_real.R
  - Funktionen_Ergebnisse.R
  - Validierungsverfahren.R
  - Verlustfunktion.R
  - Maxtemp.plots.R
- B.2 Verwendete Daten
  - Daten.R
  - Daten_Funktionen.R
- B.3 Prüfen der Robustheit (Kapitel 9)
  - Daten_robustcheck.R
  - Main_Robustcheck_syn.R
  - Main_Robustcheck_real.R
  - Plots_robustcheck.R

B1 Hauptergebnisse (Kapitel 8)

Main_syn.R

Dieses R-Skript dient zur Reproduktion der Hauptergebnisse der Masterarbeit für AS.1 und AS.2. Hierzu wird der relative Fehler als Fehlermaß verwendet und die skalenbasierten Fehlermaße (RMSE, MAPAE und MPAE).

```
# muss individuellen Pfad anpassen!
setwd("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code")
################################################################################
# 1) Laden der benötigten Daten und Funktionen

# hier entweder nur einen synthetischen Datensatz AR(2) oder MA(1) wählen
load("Daten/AR2.embedded.rdata")
#load("Daten/MA1.embedded.rdata")

# Laden der benötigten Funktionen
source("Verlustfunktion.R")
source("Validierungsverfahren.R")
source("Funktionen_Ergebnisse.R")

### ACHTUNG! Hier erneut den richtigen Datensatz wählen!!
#embedded_time_series <- embedded.MA1
embedded_time_series <- embedded.AR2
################################################################################
# 2) Gesamtdatensatz teilen
form <- target~.
outer_split <- .8 # Gesamtdatensatz gemäß 80/20 teilen
################################################################################

# 3) realtive Fehler (inkl. Boxplots)
rel_kkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"kkv")
rel_bkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"bkv")
rel_mkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"mkv")
rel_hv_bkv <-
final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"hv_bkv")
rel_h <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"h")

# Boxplot
#ACHTUNG! Bei Beschriftung vom Titel muss dieser an das entsprechende AS adaptiert werden!

#vertikale Beschriftung der einzelnen Boxplots und Schriftgröße auf 0.8 einstellen
par(las =2, cex.axis =0.8)
boxplot(rel(rel_kkv),rel(rel_bkv),rel(rel_mkv),rel(rel_hv_bkv),rel(rel_h),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "Anwendungsszenario 1")
```

```
#Scatterplot
# wurde in Masterarbeit schlussendlich nicht verwendet, da kaum ein Unterschied zwischen
# den Plots zu sehen ist
par(mfrow = c(3, 2)) #Grafikbereich auf 3 Zeilen und 2 Spalte festlegen
#vertikale Beschriftung der einzelnen Boxplots und Schriftgröße auf 0.8 einstellen
par(cex.axis =0.6, cex.main = 0.75)
plot (rel_kkv, xlab = "Out-Set Fehler (PE)", ylab ="In-Set Fehler (PE_hat)", main ="k-
fache KV")
abline(coef = c(0,1))
plot (rel_bkv, xlab = "Out-Set Fehler (PE)", ylab ="In-Set Fehler (PE_hat)", main
="geblockte KV")
abline(coef = c(0,1))
plot (rel_mkv, xlab = "Out-Set Fehler (PE)", ylab ="In-Set Fehler (PE_hat)", main
="modifizierte KV")
abline(coef = c(0,1))
plot (rel_hv_bkv, xlab = "Out-Set Fehler (PE)", ylab ="In-Set Fehler (PE_hat)", main ="hv-
geblockte KV")
abline(coef = c(0,1))
plot (rel_h, xlab = "Out-Set Fehler (PE)", ylab ="In-Set Fehler (PE_hat)", main ="Holdout")
abline(coef = c(0,1))

################################################################################
#  4) skalenbasierte Fehlermaße (RMSE, MAPAE, MPAE)

# 1 Möglichkeit: wenn nur einzelne Daten haben will (kürzere Rechendauer -> max 1 Minute)

# Hier muss das Validierungsverfahren in der Funktion entsprechend angepasst werden:
#   'kkV' = k-fache Kreuzvalidierung
#   'bkv' = geblockte Kreuzvalidierung
#   'mkv' = modifizierte Kreuzvalidierung
#   'hv_bkv' = hv-geblockte Kreuzvalidierung
#   'h' = Holdout

final_results(embedded_time_series,outer_split,form,LRM_rmse,5,"hv_bkv","mapae")
final_results(embedded_time_series,outer_split,form,LRM_rmse,5,"hv_bkv","mpae")
final_results(embedded_time_series,outer_split,form,LRM_rmse,5,"hv_bkv","rmse")

# 2 Möglichkeit: wenn ganze Matrix haben möchte (! Dauer mindestens 7 Minuten!)
# die drei skalenbasierten Fehlermaße werden für alle 5 Validierungsverfahren berechnet:
#   'kkV' = k-fache Kreuzvalidierung
#   'bkv' = geblockte Kreuzvalidierung
#   'mkv' = modifizierte Kreuzvalidierung
#   'hv_bkv' = hv-geblockte Kreuzvalidierung
#   'h' = Holdout

Fehler_skalenb(embedded_time_series, outer_split, form, LRM_rmse)
```

Main_real.R

Dieses R-Skript dient zur Reproduktion der Hauptergebnisse der Masterarbeit für AS.3. Hierzu wird erneut der relative Fehler als Fehlermaß verwendet und die skalenbasierten Fehlermaße (RMSE, MAPAE und MPAE).

```
# muss individuellen Pfad anpassen!
setwd("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code")
################################################################################
# 1) Laden der benötigten Daten und Funktionen

load("Daten/Maxtemp.embedded.rdata")
load("Daten/Mintemp.embedded.rdata")

source("Verlustfunktion.R")
source("Validierungsverfahren.R")
source("Funktionen_Ergebnisse.R")


# Maximmaltemperatur-Daten entsprechend anpassen:
Maxtemp <- embedded.Maxtemp
#letzten 48 Einträge löschen, da ansonsten Dataframe nicht auf eine Liste von Dataframes
# mit einer Länge von 200 aufgeteilt werden kann
Maxtemp <- Maxtemp[1:(nrow(Maxtemp)-48), ]
T <- 200 #Datenlänge
# Dataframe in eine Liste von Dataframes mit jeweils Länge von 200 aufteilen
Maxtemp_list <- split(Maxtemp, rep(1:(nrow(Maxtemp)/T), each=T))
# Minimumtemperatur-Daten entsprechend anpassen:
Mintemp <- embedded.Mintemp
Mintemp <- Mintemp[1:(nrow(Mintemp)-48), ] #letzten 48 Einträge löschen
T <- 200 #Datenlänge
# Dataframe in eine Liste von Dataframes mit jeweils Länge von 200 aufteilen
Mintemp_list <- split(Mintemp, rep(1:(nrow(Mintemp)/T), each=T))

# Maxtemp_list und Mintemp_list zu einem Dataframe zusammenführen
embedded_time_series <- c(Maxtemp_list, Mintemp_list)

################################################################################
# 2) Gesamtdatensatz teilen
form <- target~.
outer_split <- .8 # Gesamtdatensatz gemäß 80/20 teilen

################################################################################
# 3) realtive Fehler (inkl. Boxplots)

rel_kkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"kkv")
rel_bkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"bkv")
rel_mkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"mkv")
rel_hv_bkv <-
final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"hv_bkv")
rel_h <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"h")

# Boxplot
par(las =2, cex.axis =0.8) #vertikale Beschriftung der einzelnen Boxplots und Schriftgröße
auf 0.8 einstellen
boxplot(rel(rel_kkv),rel(rel_bkv),rel(rel_mkv),rel(rel_hv_bkv),rel(rel_h),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "Anwendungsszenario 3")
```

```
#Scatterplot
# wurde in Masterarbeit schlussendlich nicht verwendet, da kaum ein Unterschied zwischen
# den Plots zu sehen ist
par(mfrow = c(3, 2)) #Grafikbereich auf 3 Zeilen und 2 Spalte festlegen
#vertikale Beschriftung der einzelnen Boxplots und Schriftgröße auf 0.8 einstellen
par(cex.axis =0.6, cex.main = 0.75)
plot (rel_kkv, xlab = "Out-Set Fehler (PE)", ylab ="In-Set Fehler (PE_hat)", main ="k-
fache KV")
abline(coef = c(0,1))
plot (rel_bkv, xlab = "Out-Set Fehler (PE)", ylab ="In-Set Fehler (PE_hat)", main
="geblockte KV")
abline(coef = c(0,1))
plot (rel_mkv, xlab = "Out-Set Fehler (PE)", ylab ="In-Set Fehler (PE_hat)", main
="modifizierte KV")
abline(coef = c(0,1))
plot (rel_hv_bkv, xlab = "Out-Set Fehler (PE)", ylab ="In-Set Fehler (PE_hat)", main ="hv-
geblockte KV")
abline(coef = c(0,1))
plot (rel_h, xlab = "Out-Set Fehler (PE)", ylab ="In-Set Fehler (PE_hat)", main ="Holdout")
abline(coef = c(0,1))

##############################################################################
# 4) skalenbasierte Fehlermaße (RMSE, MAPAE, MPAE)

# 1 Möglichkeit: wenn nur einzelne Daten haben will (kürzere Rechendauer -> max 1 Minute)

# Hier muss das Validierungsverfahren in der Funktion entsprechend angepasst werden:
#   'kkV' = k-fache Kreuzvalidierung
#   'bkv' = geblockte Kreuzvalidierung
#   'mkv' = modifizierte Kreuzvalidierung
#   'hv_bkv' = hv-geblockte Kreuzvalidierung
#   'h' = Holdout

final_results(embedded_time_series,outer_split,form,LRM_rmse,5,"hv_bkv","mapae")
final_results(embedded_time_series,outer_split,form,LRM_rmse,5,"hv_bkv","mpae")
final_results(embedded_time_series,outer_split,form,LRM_rmse,5,"hv_bkv","rmse")

# 2 Möglichkeit: wenn ganze Matrix haben möchte
# die drei skalenbasierten Fehlermaße werden für alle 5 Validierungsverfahren berechnet:
#   'kkV' = k-fache Kreuzvalidierung
#   'bkv' = geblockte Kreuzvalidierung
#   'mkv' = modifizierte Kreuzvalidierung
#   'hv_bkv' = hv-geblockte Kreuzvalidierung
#   'h' = Holdout
Fehler_skalenb(embedded_time_series, outer_split, form, LRM_rmse)
```

Funktionen_Ergebnisse.R

Dieses R-Skript beinhaltet alle generellen Funktionen um die Hauptergebnisse aus Kapitel 8 reproduzieren zu können.

```
# Funktion, um Gesamtdatensatz in Trainings- und Testdaten zu teilen
partition <- function(x, hat.ratio) {
  len <- NROW(x)

  if (class(x)[1] == "data.frame") {
    train <- x[ seq_len(hat.ratio * len),]
    test <-  x[-seq_len(hat.ratio * len),]
  } else {
    train <- x[ seq_len(hat.ratio * len)]
    test <-  x[-seq_len(hat.ratio * len)]
  }
  list(train = train, test = test)
}



# funktion Single_error -> für eine Zeitreihe den Fehler berechnen!
Single_error <- function(x,outer_split,form,PM_verlust,nblock, methode_kv){

  ##1) PE (= Fehler von in-Set ermitteln)
  #kompletter Datensatz wird gemäß outer_split aufgeteilt
  xp <- partition(x, outer_split)

  # Teilung in Trainings- und Testdaten
  train <- xp$train
  test <- xp$test

  # Fehler von in-set:
  PE <- PM_verlust(form, train, test)

  ##2) PE_hat (=Fehler von Out-Set ermitteln mit Kreuzvalidierungsverfahren bzw. Holdout)
  # hier wird Trainings-set (und NICHT ganzer Datensatz) verwendet
  Kreuzval <- switch (methode_kv,
                      "kkv" = kkv,
                      "bkv" = blocked_kv,
                      "mkv" = modified_kv,
                      "hv_bkv"= hv_blocked_kv,
                      "h" = holdout)

  # prognostizierte Fehler durch out-set
  PE_hat <- Kreuzval(train,nblock,PM_verlust)


  ##3) Output:
  error <- c(PE_hat,PE)
  error
}
```

```
# Funktion "final_results": alle Zeitreihen (=kompletter Datensatz) wird herangezogen
# PE_hat und PE von jeder einzelner Zeitreihe wird berechnet und zum Schluss das
# gewählte Fehlermaß berechnet

final_results <- function(Daten_embedded, outer_split,form,PM_verlust,nblock, methode_kv,
fehler_messung){

  #für jede einzele Zeitreihe PE_hat und PE berechnen
  erg <- list()
  erg_alle <- lapply(1:length(Daten_embedded),
                     function(i) {
                       x <- Daten_embedded[[i]]
                       erg[[i]]   <-   Single_error(x,outer_split,form,PM_verlust,nblock,
methode_kv)
                     })

  PE_hat <- unlist(lapply(erg_alle, `[[`, 1))
  PE <- unlist(lapply(erg_alle, `[[`, 2))

  # Fehlermaß auswählen
  measure_error <- switch(fehler_messung,
                          "rmse" = rmse,
                          "mapae" = mapae,
                          "mpae" = mpae)

  # berechnen des Prognosefehlers
  fehler <- measure_error(PE_hat,PE)
  fehler
}


# Funktion für das Fehlermaß: relative Fehler

final_results_relativ  <-  function(Daten_embedded,  outer_split,form,PM_verlust,nblock,
methode_kv){

  #für jede einzelne Zeitreihe PE_hat und PE berechnen
  erg <- list()
  erg_alle <- lapply(1:length(Daten_embedded),
                     function(i) {
                       x <- Daten_embedded[[i]]
                       erg[[i]]   <-   Single_error(x,outer_split,form,PM_verlust,nblock,
methode_kv)
                     })

  PE_hat <- unlist(lapply(erg_alle, `[[`, 1))
  PE <- unlist(lapply(erg_alle, `[[`, 2))

  # in Liste darstellen (für relativen Vergleich)
  relativ <- list(x = PE, y = PE_hat)
}
```

Validierungsverfahren.R

In diesem R-Skript befinden sich die Funktionen für die 5 untersuchten Validierungsverfahren: k-fache Kreuzvalidierung, geblockte Kreuzvalidierung, modifizierte Kreuzvalidierung, hv-geblockte Kreuzvalidierung und die Holdout-Methode.

```
####k-fache Kreuzvalidierung (KV)
kkv <- function(x, nblock, PM_verlust){
  #1. Daten zufällig mischen
  set.seed(123) # für Reproduzierbarkeit seed setzen
  x_neu <- x[sample(NROW(x)), ]

  #2. Daten in Blöcke einteilen
  kv.block <- function(x, nblock) {
    cut(seq_len(NROW(x)), breaks = nblock, labels = FALSE)
  }
  f <- kv.block(x_neu, nblock)

  #3. Modell anpassen und Verlust ermitteln
  kkv.erg <- list()
  for (i in seq_len(nblock)) {
    zr.id <- which(f == i) #Block als Test-Set wählen

    train <- x_neu[-zr.id, ] #gewählten Block (= Test-Set) von Daten wegnehmen
    test  <- x_neu[ zr.id, ]

    #Modell an die verbleibenden K-1-Blöcke anpassen & Verlust ermitteln
    kkv.erg[[i]] <- PM_verlust(form, train, test)
  }

  #4. Durchschnitt der k-Verluste bilden
  kkv.erg <- mean(unlist(kkv.erg, use.names = FALSE))

  kkv.erg
}
#kkv(x,5,LRM_rmse)

kv(x,5,LRM_rmse)

#### Geblockte Kreuzvalidierung (KV)
# Standard k-fache Kreuzvalidierung ohne anfänglicher (zufälliger) Mischung der Daten
blocked_kv <- function(x, nblock, PM_verlust){

  #1. Daten in Blöcke einteilen
  kv.block <- function(x, nblock) {
    cut(seq_len(NROW(x)), breaks = nblock, labels = FALSE)
  }
  f <- kv.block(x, nblock)

  #2. Modell anpassen und Verlust ermitteln
  bkv.erg <- list()
  for (i in seq_len(nblock)) {
    zr.id <- which(f == i) #Block als Test-Set wählen

    train <- x[-zr.id, ] #gewählten Block (= Test-Set) von Daten wegnehmen
    test  <- x[ zr.id, ]
```

```
    #Modell an die verbleibenden K-1-Blöcke anpassen & Verlust ermitteln
    bkv.erg[[i]] <- PM_verlust(form, train, test)
  }

  #4. Durchschnitt der k-Verluste bilden
  bkv.erg <- mean(unlist(bkv.erg, use.names = FALSE))

  bkv.erg
}
#blocked_kv(x,5,LRM_rmse))

#### Modifizierte Kreuzvalidierung
# Standard k-fache KV, mit dem Zusatz dass jene Beobachtungen aus dem Train-Set,
# die mit dem Test-Set abhängig sind, entfernt werden.
# Die Anzahl an lags p im AR(p) Prozess gibt die Distanz an, ab welcher Beobachtungen
# unabhängig sind. Im folgenden enthält das Trainingsset daher die Werte:
# {1, ...,Z-p-1,...,Z+p+1,...n}.
modified_kv <- function(x, nblock, PM_verlust){

  p <- ncol(x)-1 #gibt Anzahl an lags p von AR(p)-Prozess an
  x$nr <- seq_len(nrow(x)) #"Nummerierung" hinzufügen (um abhängige Beobachtungen zu
finden)

  #1. Daten zufällig mischen
  set.seed(123) # für Reproduzierbarkeit
  x_neu <- x[sample(NROW(x)), ]

  #2. Daten in Blöcke einteilen
  kv.block <- function(x, nblock) {
    cut(seq_len(NROW(x)), breaks = nblock, labels = FALSE)
  }
  f <- kv.block(x_neu, nblock)

  #3. abhängige Beobachtungen aus Trainingsset entfernen und Modell trainieren
  mkv.erg <- list()
  for (i in seq_len(nblock)) {
    ts.id <- which(f == i)
    test  <- x_neu[ts.id, ] #gewählten Block (z.B: f=1) alle Daten auswählen als Test-Set

    #3.a) abhängige Beobachtungen finden:
    abhBeob <- unlist(lapply(test$nr, function(z){(z-p-1):(z+p+1)}))
    abhBeob <- unique(abhBeob) # Doppelte Werte löschen
    abhBeob <- abhBeob[abhBeob > 0]

    #3.b) Trainingsset bestimmen = Testset und abhängige Beobachtungen von Datensatz
    #entfernen

    train <- x_neu[-abhBeob, ]

    if (nrow(train) < 1) {
      mkv.erg[[i]] <- NA #falls keine Beobachtungen in Trainingsset überbleiben würden...
    } else {
      test$nr <- NULL #Nummerierung entfernen
      train$nr <- NULL

      #3.c) Modell an die verbleibenden K-1-Blöcke anpassen & Verlust ermitteln
      mkv.erg[[i]] <- PM_verlust(form, train, test)
    }
  }
```

```
  #4. Durchschnitt der k-Verluste bilden
  mkv.erg <- mean(unlist(mkv.erg, use.names = FALSE))

  mkv.erg
}
#modified_kv(x,5,LRM_rmse)


#### HV-Blocked-K-fache Kreuzvalidierung (KV)
# = blocked KV, mit Zusatz dass abhängige Beobachtungen zwischen Trainings- und
# Testset entfernt werden (aus Trainingsset).

hv_blocked_kv <- function(x, nblock, PM_verlust){

  p <- ncol(x)-1 #gibt Anzahl an lags p von AR(p)-Prozess an
  #"Nummerierung" hinzufügen (um abhängige Beobachtungen zu finden)
  x$nr <- seq_len(nrow(x))

  #1. Daten in Blöcke einteilen
  kv.block <- function(x, nblock) {
    cut(seq_len(NROW(x)), breaks = nblock, labels = FALSE)
  }
  f <- kv.block(x, nblock)

  #2. abhängige Beobachtungen aus Trainingsset entfernen und Modell trainieren
  hvkv.erg <- list()
  for (i in seq_len(nblock)) {
    ts.id <- which(f == i)
    #für den gewählten Block (z.B: f=1) alle Daten auswählen als Test-Set
    test  <- x[ts.id, ]

    #2.a) abhängige Beobachtungen finden:
    abhBeob <- unlist(lapply(test$nr, function(z){(z-p-1):(z+p+1)}))
    abhBeob <- unique(abhBeob) # Doppelte Werte löschen
    abhBeob <- abhBeob[abhBeob > 0]
    #abhBeob <- abhBeob[abhBeob > nrow(x)]

    #2.b) Trainingsset bestimmen = Testset und abhängige Beobachtungen von Datensatz
    #      entfernen
    train <- x[-abhBeob, ]

    #2.c) Nummerierung entfernen
    test$nr <- NULL
    train$nr <- NULL

    #2.d) Modell an die verbleibenden K-1-Blöcke anpassen & Verlust ermitteln
    hvkv.erg[[i]] <- LRM_rmse(form, train, test)
  }

  #3. Durchschnitt der k-Verluste bilden
  hvkv.erg <- mean(unlist(hvkv.erg, use.names = FALSE))

  hvkv.erg
}
#hv_blocked_kv(x,5,LRM_rmse)
```

```
#### Holdout-Methode
# klassisches Out-of-sample Verfahren
holdout <- function(x, nblock, PM_verlust) {
  nblock <- nblock
  xp <- partition(x, 0.7) # 70% für Trainingsset (30% Testset)

  train <- xp$train
  test <- xp$test

  # Verlust ermitteln
  PM_verlust(form, train, test)
}
#holdout(x,LRM_rmse)
```

Verlustfunktion.R

In diesem R-Skript befinden sich die die notwendige Funktionen, um die Koeffizienten des Prognosemodells (lineare Regression mit zwei Regressoren) zu finden, sowie die Verlustfunktion.

```
# Lineare Regression mit zwei Regressoren wird als Prognosemethode gewählt:

library(tsensembler)
form <- target ~ .

LRM <- function(form, train, test){
  # nur erklärende Variablen X erhalten (abhängige Variable wird durch 1 ersetzt!)
  X <- model.matrix(form, train)
  X <- data.frame(X[,-1])
  Y <- get_y(train, form) # abhängige Variable Y

  #Durchführen von OLS an Trainingsdaten (=Modell trainieren)
  lrm_m <- lm(Y ~ ., data=X) # wähle alle erklärenden Variablen

  Xtest <- model.matrix(form, test) #x-Werte von test-data
  if (ncol(Xtest) == 2) {
    Xtest <- Xtest[,-1]
    Xtest <- as.data.frame(Xtest)
    colnames(Xtest) <- as.character(lrm_m[["terms"]][[3]])
  } else {
    Xtest <- Xtest[,-1]
    Xtest <- as.data.frame(Xtest)
  }

  #Datum brauche ich nicht ... deshalb unname!
  y_hat = as.matrix(unname(predict(lrm_m,Xtest)))
}

#############################################################################
# Fehlermessung- Funktionen:

#1) Root Mean Squared Error
rmse <- function(x_hat,x){
  sqrt(mean(se(x_hat, x), na.rm = TRUE))
}

#2) Mean Absolut Predictive Accuracy Error
mapae <- function(x_hat,x){
  mean(abs(x_hat - x), na.rm=TRUE)
}

#3) Mean Predictive Accuracy Error
mpae <- function(x_hat,x){
  mean(x_hat - x)
}


#############################################################################
```

```
# Verlustfunktion:

# RMSE (Root Mean Squared Error)
LRM_rmse <- function(form, train, test) {
  Y <- get_y(test, form)
  Y_hat <- LRM(form, train, test)

  rmse_erg <- rmse(Y_hat, Y)# Root Mean Squared Error

  rmse_erg
}




LRM_mapae <- function(form, train, test) {
  Y <- get_y(test, form)
  # Y_tr <- get_y(train, form)  (wird nicht benötigt)
  Y_hat <- LRM(form, train, test)

  mapae_erg <- mapae(Y_hat,Y) #Mean Absolut Predictive Accuracy Error

  mapae_erg
}


LRM_mpae <- function(form, train, test) {
  Y <- get_y(test, form)
  # Y_tr <- get_y(train, form)    (wird nicht benötigt)
  Y_hat <- LRM(form, train, test)

  mpae_erg <- mpae(Y_hat,Y) #Mean Predictive Accuracy Error

  mpae_erg
}
```

Maxtemp.plots.R

Dieses R-Skript dient ausschließlich dazu, um Abbdilung 14 der Masterarbeit zu plotten. Abbildung 14 zeigt den tatsächlichen Prognosefehler bei der Höchsttemperatur.

```
#laden der benötigten Daten
load("C:/Users/potik/OneDrive/Dokumente/Fern
fH/Masterarbeit/Code/Daten/Maxtemp.embedded.rdata")

source("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code/Verlustfunktion.R")

embedded_time_series <- embedded.Maxtemp
form <- target~.
outer_split <- .8
partition <- function(x, hat.ratio) {
  len <- NROW(x)

  if (class(x)[1] == "data.frame") {
    train <- x[ seq_len(hat.ratio * len),]
    test <-  x[-seq_len(hat.ratio * len),]
  } else {
    train <- x[ seq_len(hat.ratio * len)]
    test <-  x[-seq_len(hat.ratio * len)]
  }
  list(train = train, test = test)
}

#kompletter Datensatz wird gemäß outer_split aufgeteilt
xp <- partition(embedded_time_series, outer_split)

# Teilung in Trainings- und Testdaten
train <- xp$train
test <- xp$test

# Prognosewerte
Y <- test[1]
Y_hat <- data.frame(LRM(form, train, test)) #prognostizierter Wert out-of-sample


#Prognosefehler plotten
error_OOS <- Y_hat - Y
pdf("PE.pdf", width = 14/2.54, height = 17/2.54)
par(mfrow=c(2,1), cex.axis =0.7, cex.main = 0.9, cex.lab = 0.8)
plot.ts(error_OOS,   xlab="Tage",ylab="Celsius",lwd   =   0.5,   main="Höchsttemperatur   -
tatsächliche Prognosefehler")
dev.off()
```

B2 Daten

Daten.R

Das folgende R-Skript reproduziert alle benötigten Daten für die Masterarbeit. Hierfür werden synthetische Daten generiert: AR(2) und MA(1) Prozess bzw. reale Daten von der Time Series Data Library (tsdl) verwendet.

```
library(tsensembler)
library(tsdl)
library(forecast)
library(xts)
################################################################################
#1) synthetische Daten generieren

# Laden der benötigten Funktionen
setwd("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code/Daten")
source("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code/Daten/simulateTS.R")
source("C:/Users/potik/OneDrive/Dokumente/Fern
fH/Masterarbeit/Code/Daten/Daten_Funktionen.R")


wh <- 1000 #1000 Monte Carlo Simulationen durchführen
WH_T <- 1:wh #Anzahl an Monte Carlo Simulationen
TS_T <- 200 # Datenlänge T der Zeitreihe (Time Series)

### Anwendungsszenario 1: AR(2)
AR2 <- lapply(WH_T,DGP1)

### Anwendungsszenario 2: MA(1)
MA1 <- lapply(WH_T,DGP2)

################################################################################
# 2) reale Daten
# Link für Daten: https://pkg.yangzhuoranyang.com/tsdl/articles/tsdl.html
#install.packages("devtools")
devtools::install_github("FinYang/tsdl")


# 2 gewählte, stationäre Zeitreihendaten
Temp <- subset(tsdl, description = "temperatures in Melbourne")
# Mindest- und Höchsttemperatur in Melbourne
Temperatur <- list(max.Temp=Temp[[1]], min.Temp=Temp[[2]])
# Überprüfen ob wirklich stationär
library(tseries)
adf.test(unlist(Temperatur[["max.Temp"]])) #p-value ist kleiner als 0.01 -> d.h. ja!
adf.test(unlist(Temperatur[["min.Temp"]])) #p-value ist kleiner als 0.01 -> d.h. ja!

# Höchsttemperatur in Melbourne in Zeitreihen-Format umwandeln
Maxtemp <- as.data.frame(Temp[[1]])
#Datum hinzufügen
date = seq(from = as.Date("1981-01-01"), to = as.Date("1990-12-29"), by = 'day')
Maxtemp_ts <- cbind(Maxtemp, date) #Zusammenfügen
Maxtemp <- xts(Maxtemp_ts$x, Maxtemp_ts$date) # in Zeitreihen-Format converten
```

```
# Mindesttemperatur in Melbourne in Zeitreihen-Format umwandeln
Mintemp <- as.data.frame(Temp[[2]])
#Datum hinzufügen
date = seq(from = as.Date("1981-01-01"), to = as.Date("1990-12-29"), by = 'day')
Mintemp_ts <- cbind(Mintemp, date) #Zusammenfügen
Mintemp <- xts(Mintemp_ts$x, Mintemp_ts$date) # in Zeitreihen-Format converten


################################################################################
#3) embedded Daten: Die Dimension ist abhängig von der Annahme des Modells. hier
# wird angenommen, dass das Prognosemodell eine multiple lineare Regression mit
# zwei Regressoren ist. Die Embedding-Dimension ist daher 3!


# Anwendungsszenario 1: AR(2)
embedded.AR2 <- lapply(AR2, embed_timeseries, embedding.dimension = 3)

#Anwendungsszenario 2: MA(1)
embedded.MA1 <- lapply(MA1, embed_timeseries, embedding.dimension = 3)

# Anwendungsszeanrio 3: reale Daten - Höchsttemperatur in Melbourne
embedded.Maxtemp <- embed_timeseries(Maxtemp, embedding.dimension = 3)

# Anwendungsszeanrio 3: reale Daten - Mindesttemperatur in Melbourne
embedded.Mintemp <- embed_timeseries(Mintemp, embedding.dimension = 3)

################################################################################
# 4) Abspeichern der generierten, synthetischen Daten
save(embedded.AR2, file = "AR2.embedded.rdata")
save(embedded.MA1, file = "MA1.embedded.rdata")

#5) Abspeichern der realen, embedded Daten
save(embedded.Maxtemp, file = "Maxtemp.embedded.rdata")
save(embedded.Mintemp, file = "Mintemp.embedded.rdata")
################################################################################
# 6) Plotten eines Beispielsprozesses der synthetischen Daten
plot.ts(AR2[[1]],xlab="Zeit",ylab="",lwd = 1.9, main="AR(2)")
plot.ts(MA1[[1]],xlab="Zeit",ylab="",lwd = 1.9, main="MA(1)")

par(mfrow=c(2,1))
plot.ts(Temperatur[[1]],xlab="Jahr",ylab="Celsius",lwd = 0.5, main="Höchsttemperatur in
Melbourne")
plot.ts(Temperatur[[2]],xlab="Jahr",ylab="Celsius",lwd = 0.5, main="Mindesttemperatur in
Melbourne")

################################################################################
# 7) Für Beispiel in Literaturteil plotten
# stationärer AR(1) Prozess mit phi = -0.9
ar1_s <- arima.sim(model=list(ar=-0.7), n=100, sd=1)
# nicht-stationärer AR(1) Prozess mit phi = 1.6
ar1_ns <- numeric(20) #leeren Vektor erzeugen, um die simulierten Werte zu speichern
ar1_ns[1] <- rnorm(1) #Anfangswert des Prozesses random wählen (zwischen 0 und 1)
phi <- -1.03
# AR(1)-Prozess simulieren
for (i in 2:100) {
  ar1_ns[i] <- phi * ar1_ns[i-1] + rnorm(1)
}

# Abspeichern und plotten der Ergebnisse
pdf("AR(1).pdf", width = 17/2.54, height = 17/2.54)
par(mfrow=c(2,1), cex.axis =0.7, cex.main = 0.9, cex.lab = 0.8, mar=c(4, 3, 2, 2.1),
mgp=c(2,0.7,0))
plot.ts(ar1_s,xlab="Zeit",ylab="",lwd = 1.9, main=" (a). stationärer AR(1)-Prozess")
plot.ts(ar1_ns,xlab="Zeit",ylab="",lwd = 1.9, main="(b). nicht stationärer AR(1)-Prozess")
dev.off()
```

Daten_Funktionen.R

Dieses R-Skript beinhaltet alle Funktionen, die für die Reproduktion der Daten benötigt werden.

```
# Prozess positiv machen, indem Minimum subtrahiert wird und 1 addiert wird
positiv_Proz <- function(y) {
    y - min(y) + 1
  }

### Anwendungsszenario 1: AR(2)
DGP1 <- function(i) {
  positiv_Proz(
    as.vector(simulateLinearTS(TS_T,
                  ar = TRUE,
                  ma = FALSE,
                  lags = 2,
                  seed = i,
                  maxRoot = 5,
                  n.start = 300)[["ts"]]
    )
  )
}

### Anwendungsszenario 2: MA(1)
DGP2 <- function(j) {
  positiv_Proz(
    as.vector(simulateLinearTS(TS_T,
                               ar = FALSE,
                               ma = TRUE,
                               lags = 1,
                               seed = j,
                               maxRoot = 5,
                               n.start = 100)[["ts"]]
    )
  )
}
```

B3 Prüfen der Robustheit (Kapitel 9)

Daten_robustcheck.R

Dieses R-Skript reproduziert alle benötigten Daten für Kapitel 9 „Prüfen der Robustheit" in der Masterarbeit. Hierfür werden synthetische Daten generiert: AR(2) und MA(1) Prozess mit unterschiedlichen Datenlängen (T=50,100,200,350 und 500).

Der Code muss hier für die ausgewählte Datenlänge manuell angepasst werden. Sprich bei einer Wahl von der Datenlänge T=50, muss „TS_T <- 50“ gewählt werden und die restlichen Zeitreihenlängen müssen als Kommentare gelassen werden.

```
#1) benötigte Funktionen laden

#Laden der wichtigsten Funktionen
#Pfad muss wahrscheinlich geändert/adaptiert werden bei Reproduktion der Ergebnisse
setwd("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code/Daten")
source("simulateTS.R")
source("Daten_Funktionen.R")

library(tsensembler)

################################################################################
# 2) synthetische Daten generieren

# in diesem Pfad die Daten abspeichern
setwd("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code/Robustheit")

# Datenlänge variieren! Je nach Wahl werden unterschiedliche Daten generiert
# ACHTUNG! Hier eine Datenlänge wählen und Beschriftungen in 3) entsprechend anpassen!
TS_T <- 50 # Datenlänge T der Zeitreihe (Time Series)
#TS_T <- 100 # Datenlänge T der Zeitreihe (Time Series)
#TS_T <- 200 # Datenlänge T der Zeitreihe (Time Series)
#TS_T <- 350 # Datenlänge T der Zeitreihe (Time Series)
#TS_T <- 500 # Datenlänge T der Zeitreihe (Time Series)

wh <- 1000 #1000 Monte Carlo Simulationen durchführen
WH_T <- 1:wh #Anzahl an Monte Carlo Simulationen


### Anwendungsszenario 1: AR(2)
AR2 <- lapply(WH_T,DGP1)
embedded.AR2 <- lapply(AR2, embed_timeseries, embedding.dimension = 3) #embedded Daten

### Anwendungsszenario 2: MA(1)
MA1 <- lapply(WH_T,DGP2)
embedded.MA1 <- lapply(MA1, embed_timeseries, embedding.dimension = 3) #embedded Daten


################################################################################
# 3) Abspeichern der generierten, synthetischen Daten
# ACHTUNG! Muss hier auch den Namen, wie es abgespeichert werden soll, ändern/adpatieren!
save(embedded.AR2, file = "AR2.50.rdata")
save(embedded.MA1, file = "MA1.50.rdata")
```

Main_Robustcheck_syn.R

Dieses R-Skript beinhaltet die notwendigen Codes, um die Ergebnisse für AS.1 und AS.1 in Kapitel 9 „Überprüfen der Robustheit“ zu erhalten bzw. um sie zu reproduzieren. Hierzu wird der Einfluss der Datenlänge T überprüft. Dabei wird die Datenlänge T auf 50, 100, 200 und 500 eingestellt und so die Ergebnisse untersucht.

Der Code muss in Schritt 1) manuell angepasst werden. Hier muss ausgesucht werden, welche synthetische Datei (z.B.: MA(1)-Prozess mit einer Datenlänge von 50 oder ein AR(1) Prozess mit einer Datenlänge von 500) gewählt wird. Bei der Berechnung und Abspeicherung des relativen Fehlers in Schritt 3) muss der Name entsprechend des gewählten Datensatzes angepasst werden. Dasselbe gilt für Schritt 4), bei der Berechnung der skalenbasierten Fehlermaße.

Die Entscheidung den Code manuell zu ändern wurde getroffen, da die Durchlaufzeit des Codes bei Verwendung einer Schleife sehr lange gedauert hätte (über zwei Stunden). Eine Kopierung des Codes für jeden einzelnen Datensatz wurde als nicht sinnvoll erachtet.

```
# muss individuellen Pfad anpassen!
setwd("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code/Robustheit")

################################################################################
# 1) Laden der benötigten Daten und Funktionen

# hier entweder nur einen synthetischen Datensatz AR(2) oder MA(1) wählen oder BEIDE realen
Datensätze!

# AR(2)-Daten
#load("AR2.50.rdata")
#load("AR2.100.rdata")
#load("AR2.200.rdata")
#load("AR2.500.rdata")

#MA(1)-Daten
load("MA1.50.rdata")
#load("MA1.100.rdata")
#load("MA1.200.rdata")
#load("MA1.500.rdata")

# muss individuellen Pfad anpassen!
source("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code/Verlustfunktion.R")
source("C:/Users/potik/OneDrive/Dokumente/Fern
fH/Masterarbeit/Code/Validierungsverfahren.R")
source("C:/Users/potik/OneDrive/Dokumente/Fern
fH/Masterarbeit/Code/Funktionen_Ergebnisse.R")

### ACHTUNG! Hier erneut den richtigen Datensatz wählen!!
#embedded_time_series <- embedded.AR2
embedded_time_series <- embedded.MA1

################################################################################
```

```
# 2) Gesamtdatensatz teilen
form <- target~.
outer_split <- .8 # Gesamtdatensatz gemäß 80/20 teilen

################################################################################
# 3) relative Fehler

rel_kkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"kkv")
rel_bkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"bkv")
rel_mkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"mkv")
rel_hv_bkv <-
final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"hv_bkv")
rel_h <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"h")

# relativen Fehler für Boxplot in eine Liste zusammenfügen
# ACHTUNG: bei Aspeicherung den Namen entsprechend anpassen! bzw. auch rel_R_50 zum Beispiel
rel_MA1_50 <- list(rel(rel_kkv),rel(rel_bkv),rel(rel_mkv),rel(rel_hv_bkv),rel(rel_h))
names(rel_MA1_50) <- c("kkv","bkv","mkv","hv_bkv","h")

# abspeichern der Ergebnisse
save(rel_MA1_50, file = "rel_MA1.50.rdata")

################################################################################
# 4) skalenbasierte Fehlermaße (RMSE, MAPAE, MPAE)
# die drei skalenbasierten Fehlermaße werden für alle 5 Validierungsverfahren berechnet:
#   'kkV' = k-fache Kreuzvalidierung
#   'bkv' = geblockte Kreuzvalidierung
#   'mkv' = modifizierte Kreuzvalidierung
#   'hv_bkv' = hv-geblockte Kreuzvalidierung
#   'h' = Holdout

# ACHTUNG: Namensanpassung!
sk_MA1_50 <- Fehler_skalenb(embedded_time_series, outer_split, form, LRM_rmse)
save(sk_MA1_50, file = "sk_MA1.50.RData")
```

Main_Robustcheck_real.R

Dieses R-Skript beinhaltet die notwendigen Codes, um die Ergebnisse von AS.3 in Kapitel 9 „Überprüfen der Robustheit“ zu erhalten bzw. um sie zu reproduzieren. Die Datenlänge T wird auf 50, 100, 200 und 500 eingestellt und so die Ergebnisse untersucht.

Der Code muss teilweise manuell angepasst werden. In Schritt 2) muss die Datenlänge gewählt werden, anhand der in Schitt 3) der Datensatz angepasst wird. Bei der Berechnung des relativen Fehlers in Schritt 4), sowie der skalenbasierten Fehlermaße in Schritt 5) muss erneut „nur“ der Name entsprechend modifiziert werden.

```
# muss individuellen Pfad anpassen!
setwd("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code/Robustheit")

################################################################################
# 1) Laden der benötigten Daten und Funktionen
load("C:/Users/potik/OneDrive/Dokumente/Fern
fH/Masterarbeit/Code/Daten/Maxtemp.embedded.rdata")
load("C:/Users/potik/OneDrive/Dokumente/Fern
fH/Masterarbeit/Code/Daten/Mintemp.embedded.rdata")

# muss individuellen Pfad anpassen!
source("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code/Verlustfunktion.R")
source("C:/Users/potik/OneDrive/Dokumente/Fern
fH/Masterarbeit/Code/Validierungsverfahren.R")
source("C:/Users/potik/OneDrive/Dokumente/Fern
fH/Masterarbeit/Code/Funktionen_Ergebnisse.R")


################################################################################
# 2) Wahl der Datenlänge

# ACHTUNG: Datenlänge entsprechend der gewählten Datenlänge anpassen!
T <- 50 #Datenlänge
#T <- 100 #Datenlänge
#T <- 200 #Datenlänge
#T <- 500 #Datenlänge

################################################################################
# 3) Anpassung der Daten

# Beachte: bei einer Datenlänge von T=500 müssen letzten 148 (und nicht 48) Beobachtungen
gelöscht werden!

## Maximmaltemperatur-Daten entsprechend anpassen:
Maxtemp <- embedded.Maxtemp
#letzten 48 Einträge löschen, da ansonsten Dataframe nicht auf eine Liste von Dataframes
# mit einer Länge von 50/100/200/500 aufgeteilt werden kann
Maxtemp <- Maxtemp[1:(nrow(Maxtemp)-48), ]
# Dataframe in eine Liste von Dataframes
Maxtemp_list <- split(Maxtemp, rep(1:(nrow(Maxtemp)/T), each=T))

# Minimumtemperatur-Daten entsprechend anpassen:
Mintemp <- embedded.Mintemp
Mintemp <- Mintemp[1:(nrow(Mintemp)-48), ] #letzten 48 Einträge löschen
# Dataframe in eine Liste von Dataframes
Mintemp_list <- split(Mintemp, rep(1:(nrow(Mintemp)/T), each=T))
```

```
# Maxtemp_list und Mintemp_list zu einem Dataframe zusammenführen
embedded_time_series <- c(Maxtemp_list, Mintemp_list)

################################################################################
# 4) Gesamtdatensatz teilen
form <- target~.
outer_split <- .8 # Gesamtdatensatz gemäß 80/20 teilen

################################################################################
# 5) relative Fehler

rel_kkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"kkv")
rel_bkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"bkv")
rel_mkv <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"mkv")
rel_hv_bkv <-
final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"hv_bkv")
rel_h <- final_results_relativ(embedded_time_series,outer_split,form,LRM_rmse,5,"h")

# relativen Fehler für Boxplot in eine Liste zusammenfügen
# ACHTUNG: Namen gemäß Datenlänge anpassen!
rel_R_50 <- list(rel(rel_kkv),rel(rel_bkv),rel(rel_mkv),rel(rel_hv_bkv),rel(rel_h))
names(rel_R_50) <- c("kkv","bkv","mkv","hv_bkv","h")

# abspeichern der Ergebnisse
save(rel_R_50, file = "rel_R.50.rdata")

################################################################################
# 5) skalenbasierte Fehlermaße (RMSE, MAPAE, MPAE)
# die drei skalenbasierten Fehlermaße werden für alle 5 Validierungsverfahren berechnet:
#   'kkV' = k-fache Kreuzvalidierung
#   'bkv' = geblockte Kreuzvalidierung
#   'mkv' = modifizierte Kreuzvalidierung
#   'hv_bkv' = hv-geblockte Kreuzvalidierung
#   'h' = Holdout

# ACHTUNG: Namen gemäß Datenlänge anpassen!
sk_R_50 <- Fehler_skalenb(embedded_time_series, outer_split, form, LRM_rmse)
save(sk_R_50, file = "sk_R.50.RData")
```

Plots_robustcheck.R

Dieses R-Skript generiert die benötigten Boxplots für Kapitel 9 und zeigt die Ergebnisse der skalenbasierten Fehlermaße.

```
# Laden der benötigten Daten (= AR- und MA-Prozesse mit einer Datenlänge von 50, 100,200
und 500)
setwd("C:/Users/potik/OneDrive/Dokumente/Fern fH/Masterarbeit/Code/Robustheit")

# Laden der relativen-Fehler Daten
load("rel_AR2.50.rdata")
load("rel_AR2.100.rdata")
load("rel_AR2.200.rdata")
load("rel_AR2.500.rdata")

load("rel_MA1.50.rdata")
load("rel_MA1.100.rdata")
load("rel_MA1.200.rdata")
load("rel_MA1.500.rdata")

load("rel_R.50.rdata")
load("rel_R.100.rdata")
load("rel_R.200.rdata")
load("rel_R.500.rdata")

# Plotten in Form eines Boxplots

# Anwendungsszeanrio 1: AR(2)-Prozess
par(mfrow = c(2, 2)) #Grafikbereich auf 2 Zeilen und 2 Spalte festlegen
#vertikale Beschriftung der einzelnen Boxplots und Schriftgröße auf 0.8 einstellen
par(las =2, cex.axis =0.8)
boxplot(rel_AR2_50,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 50")
boxplot(rel_AR2_100,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 100")
boxplot(rel_AR2_200,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 200")
boxplot(rel_AR2_500,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 500")

# Anwendungsszeanrio 2: AR(2)-Prozess
par(mfrow = c(2, 2)) #Grafikbereich auf 2 Zeilen und 2 Spalte festlegen
#vertikale Beschriftung der einzelnen Boxplots und Schriftgröße auf 0.8 einstellen
par(las =2, cex.axis =0.8)
boxplot(rel_MA1_50,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 50")
boxplot(rel_MA1_100,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 100")
```

```
boxplot(rel_MA1_200,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 200")
boxplot(rel_MA1_500,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 500")

# Abbildung  19:
#dev.off()
#vertikale Beschriftung der einzelnen Boxplots und Schriftgröße auf 0.8 einstellen
par(las =2, cex.axis =0.8)
boxplot(rel_MA1_500,
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 500")

# Anwendungsszeanrio 3: reale Daten
par(mfrow = c(2, 2)) #Grafikbereich auf 2 Zeilen und 2 Spalte festlegen
#vertikale Beschriftung der einzelnen Boxplots und Schriftgröße auf 0.8 einstellen
par(las =2, cex.axis =0.8)
boxplot(rel_R_50, ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 50")
boxplot(rel_R_100,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 100")
boxplot(rel_R_200,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 200")
boxplot(rel_R_500,ylim = c(0.4, 3.5),
        names = c("k-fache KV", "gebl. KV", "mod. KV", "hv-gebl. KV","Holdout"),
        ylab = "Out-Set Fehler / In-Set Fehler",
        main = "T = 500")


##############################################################################
# skalenbasierte Fehlermaße
load("sk_AR2.50.rdata")
load("sk_AR2.100.rdata")
load("sk_AR2.200.rdata")
load("sk_AR2.500.rdata")

load("sk_MA1.50.rdata")
load("sk_MA1.100.rdata")
load("sk_MA1.200.rdata")
load("sk_MA1.500.rdata")

load("sk_R.50.rdata")
load("sk_R.100.rdata")
load("sk_R.200.rdata")
load("sk_R.500.rdata")
```